FUJITSU

FUJITSU Software

FUJITSU

SSL II スレッド並列機能 使用手引書
（科学用サブルーチンライブラリ）

J2UL-2572-01Z0(01)
2020年6月

ま　え　が　き

本マニュアルは, 科学用サブルーチンライブラリ SSL II スレッド並列機能(Scientific Subroutine Library II Thread-Parallel Capabilities)の機能と使用方法について記述しています.

SSL II スレッド並列機能は, 共用メモリ型スカラ並列計算機システム上で大規模な問題を効率良く計算するための並列アルゴリズムを使用したライブラリです.このため多くのルーチンでは呼出しインタフェースは従来の SSL II, SSL II/VP, SSL II/VPP および SSL II/HPF のインタフェースとは異なります. 本マニュアルでは, それらのルーチンの使用方法について解説します.

また, SSL II スレッド並列機能には, 既存の逐次 SSL II 機能の倍精度ルーチンを並列化することにより, 同一の引数並びで利用できるルーチンも含まれています. ルーチン名は, 逐次倍精度ルーチンの先頭文字の“D”を, “DM_”に変えたものになります. 並列化対象としたルーチン一覧については, 本マニュアルの“SSL II スレッド並列機能一覧表, 2. 逐次 SSL II 機能をスレッド並列化したルーチン”を参照してください. それらのルーチンの使用方法については, 逐次 SSL II のマニュアル群を参照してください.

本書の構成は次のとおりです.

第 I 部　概説
SSL II スレッド並列機能を利用する上での注意事項を示します.

第 II 部　サブルーチンの使用方法
個々のサブルーチンの使用方法を述べます. 各サブルーチンはアルファベット順に編集されています.

SSL II スレッド並列機能を利用する上で OpenMP Fortran の知識を仮定しています. OpenMP Fortran の仕様については “OpenMP Application Program Interface Version 2.5 May 2005” を参照してください.

OpenMP Fortran の処理系の詳細は, Fortran の使用手引書を参照してください. また, スパース行列の格納方法および反復法の収束については, “FUJITSU SSL II 拡張機能使用手引書 II” を参照してください. (文中では “SSL II 拡張機能使用手引書 II” と略記します.)

SSL II スレッド並列機能は, VPP シリーズ向け並列数値計算ライブラリ SSL II/VPP 用に開発されたコードおよびアルゴリズムを全面的もしくは部分的に利用または改造して開発された機能が含まれています.なお, SSL II/VPP はオーストラリア国立大学(The Australian National University: ANU)と富士通株式会社の共同開発によるものです. ANU での開発は, Mike Osborne 教授と Richard Brent 教授により指導され, オーストラリア国立大学スーパーコンピュータセンター長の Bob Gingold 博士が幹事を勤められました.下記の ANU の研究者の方々が SSL II/VPP

の設計と実現に従事されました. 富士通株式会社はこれらの人々の御協力に対して感謝致します.

Professor Richard Peirce Brent
Dr Andrew James Cleary
Dr Murray Leslie Dow
Mr Christopher Robert Dun
Dr Lutz Grosz
Dr David Lawrence Harrar II
Dr Markus Hegland
Ms Judith Helen Jenkinson
Dr Margaret Helen Kahn
Dr Zbigniew Leyk
Mr David John Miron
Professor Michael Robert Osborne
Dr Peter Frederick Price
Dr Stephen Gwyn Roberts
Dr David Barry Singleton
Dr David Edward Stewart
Dr Bing Bing Zhou

本書は旧マニュアル(前版)を元に作成しました. 新しく追加された箇所及び記載内容が旧マニュアルより修正された箇所には, 目次および SSL II スレッド並列機能一覧表に＊印を付けてあります.

輸出管理規制について

本ドキュメントを輸出または第三者へ提供する場合は, お客様が居住する国および米国輸出管理関連法規等の規制をご確認のうえ, 必要な手続きをおとりください.

出版年月および版数

版数	マニュアルコード
2020 年 6 月 第 2 版	J2UL-2572-01Z0(01)
2020 年 2 月 初版	J2UL-2572-01Z0(00)

著作権表示

お 願 い

・本書を無断で他に転載しないようお願いします.
・本書は予告なしに変更されることがあります.

変更履歴

変更内容	変更箇所	版数
製品のレベルアップに伴い、体裁を変更.	-	第 2 版
誤字の訂正	DM_VBCSCC, DM_VBCSD, DM_VRANU5, DM_VSCLU, DM_VSRLU, DM_VSSSLU, DM_VLCSPSXCR1	

SSL II スレッド並列機能一覧表

注)　＊については, まえがき参照

1. SMP 向け並列アルゴリズムを利用したルーチン

SMP 向け並列アルゴリズムを利用した以下の機能については本マニュアルで解説しています.

行列演算

サブルーチン名	項　　目	ページ
DM_VMGGM	行列の積 (実行列)	171
DM_VMVSCC	実スパース行列と実ベクトル積 (圧縮列格納法)	190
DM_VMVSCCC	複素スパース行列と複素ベクトル積 (圧縮列格納法)	194
DM_VMVSD	実スパース行列と実ベクトルの積 (対角形式格納法)	198
DM_VMVSE	実スパース行列と実ベクトルの積 (ELLPACK 形式格納法)	201

連立 1 次方程式(直接法)

サブルーチン名	項　　目	ページ
DM_VLAX	実行列の連立 1 次方程式 (ブロック化された LU 分解法)	122
DM_VALU	実行列の LU 分解 (ブロック化された LU 分解法)	13
DM_VLUX	LU 分解された実行列の連立 1 次方程式	168
DM_VLSX	正値対称行列の連立 1 次方程式 (ブロック化された変形コレスキー分解法)	164
DM_VSLDL	正値対称行列の LDL^T 分解 (ブロック化された変形コレスキー分解法)	365
DM_VLDLX	LDL^T 分解された正値対称行列の連立 1 次方程式	145
DM_VLCX	複素行列の連立 1 次方程式 (ブロック化された LU 分解法)	142
DM_VCLU	複素行列の LU 分解 (ブロック化された LU 分解法)	76
DM_VCLUX	LU 分解された複素行列の連立 1 次方程式	80
DM_VLBX	実バンド行列の連立 1 次方程式 (ガウスの消去法)	125
DM_VBLU	実バンド行列の LU 分解 (ガウスの消去法)	51
DM_VBLUX	LU 分解された実バンド行列の連立 1 次方程式	57
DM_VSSPS	正値対称スパース行列の連立 1 次方程式 (Left-looking な LDL^T 分解法)	428
DM_VSCHOL	正値対称スパース行列の LDL^T 分解 (Left-looking な方法)	272

サブルーチン名	項　　目	ページ
DM_VSCHOLX	LDL^T 分解された正値対称スパース行列の連立 1 次方程式	287
DM_VSRS	実スパース行列の連立 1 次方程式(LU 分解法)	407
DM_VSRLU	実スパース行列の LU 分解	369
DM_VSRLUX	LU 分解された実スパース行列の連立 1 次方程式	390
DM_VSCS	複素スパース行列の連立 1 次方程式(LU 分解法)	339
DM_VSCLU	複素スパース行列の LU 分解	300
DM_VSCLUX	LU 分解された複素スパース行列の連立 1 次方程式	322
DM_VSSSS ＊	構造的に対称な実スパース行列の連立 1 次方程式(LU 分解法)	478
DM_VSSSLU ＊	構造的に対称な実スパース行列の LU 分解	443
DM_VSSSLUX ＊	LU 分解された構造的に対称な実スパース行列の連立 1 次方程式	462

連立 1 次方程式(反復法)

サブルーチン名	項　　目	ページ
DM_VCGD	正値対称スパース行列の連立 1 次方程式 (前処理付き CG 法, 対角形式格納法)	62
DM_VCGE	正値対称スパース行列の連立 1 次方程式 (前処理付き CG 法, ELLPACK 形式格納法)	69
DM_VBCSCC	非対称または不定値のスパース行列の連立 1 次方程式 (BICGSTAB(*l*)法, 圧縮列格納法)	27
DM_VBCSD	非対称または不定値のスパース行列の連立 1 次方程式 (BICGSTAB(*l*)法, 対角形式格納法)	36
DM_VBCSE	非対称または不定値のスパース行列の連立 1 次方程式 (BICGSTAB(*l*)法, ELLPACK 方式格納法)	44
DM_VTFQD	非対称または不定値のスパース実行列の連立 1 次方程式 (TFQMR 法, 対角形式格納法)	504
DM_VTFQE	非対称または不定値のスパース実行列の連立 1 次方程式 (TFQMR 法, ELLPACK 形式格納法)	512
DM_VAMLID	スパースな M-行列の連立 1 次方程式 (代数的マルチレベル反復法[AMLI 法], 対角形式格納法)	17
DM_VMLBIFE	スパース行列の連立 1 次方程式 (不完全ブロック分解に基づくマルチレベル前処理付き反復法, ELLPACK 形式格納法)	176
DM_VLCSPSXCR1	非エルミート複素対称スパース行列の連立 1 次方程式 (不完全 LDL^T 分解に基づく前処理付き共役直交共役残差法 (*COCR* 法), 対称行列用圧縮行格納法)	131

DM_VLSPAXCR2	実非対称スパース行列の連立 1 次方程式 (近似逆行列に基づく前処理付きの演繹的次元縮小法 (IDR 法), 圧縮行格納法)	149

微分方程式

サブルーチン名	項　　目	ページ
DM_VRADAU5	スティフ連立 1 階常微分方程式/微分代数方程式 (陰的ルンゲ・クッタ法)	217

偏微分方程式の離散化

サブルーチン名	項　　目	ページ
DM_VPDE2D	2 次元 2 階偏微分方程式の有限差分法による離散化によるスパース行列の連立 1 次方程式の生成	204
DM_VPDE3D	3 次元 2 階偏微分方程式の有限差分法による離散化によるスパース行列の連立 1 次方程式の生成	210

逆行列

サブルーチン名	項　　目	ページ
DM_VMINV	実行列の逆行列 (ブロック化された Gauss-Jordan 法)	174
DM_VCMINV	複素行列の逆行列(ブロック化された Gauss-Jordan 法)	83

固有値問題

サブルーチン名	項　　目	ページ
DM_VSEVPH	実対称行列の固有値・固有ベクトル (三重対角化, マルチセクション法, 逆反復法)	360
DM_VHEVP	エルミート行列の固有値・固有ベクトル	92
DM_VTDEVC	実三重対角行列の固有値・固有ベクトル	497
DM_VGEVPH	実対称行列の一般化固有値問題 (固有値および固有ベクトル) (三重対角化, マルチセクション法, 逆反復法)	86
DM_VTRID	実対称行列の実対称三重対角行列への変換 (ハウスホルダー法)	519
DM_VHTRID	エルミート行列の実対称三重対角行列への変換 (ハウスホルダー法)	97
DM_VJDHECR	エルミートスパース行列の固有値問題 (Jacobi-Davidson 法, 圧縮行格納法)	101
DM_VJDNHCR	複素スパース行列の固有値問題 (Jacobi-Davidson 法, 圧縮行格納法)	112

フーリエ変換

サブルーチン名	項　　目	ページ
DM_V1DCFT	1 次元離散型複素フーリエ変換 (2,3,5 及び 7 の混合基底)	522
DM_V1DCFT2	1 次元離散型複素フーリエ変換 (2,3,5 及び 7 の混合基底)	526
DM_V1DMCFT	1 次元多重離散型複素フーリエ変換 (2,3,5 及び 7 の混合基底)	529
DM_V2DCFT	2 次元離散型複素フーリエ変換 (2,3,5 及び 7 の混合基底)	533
DM_V3DCFT	3 次元離散型複素フーリエ変換 (2,3,5 及び 7 の混合基底)	537
DM_V3DCFT2	3 次元離散型複素フーリエ変換 (2,3,5 及び 7 の混合基底)	541
DM_V1DRCF	1 次元離散型実フーリエ変換 (2,3,5 及び 7 の混合基底)	545
DM_V1DRCF2	1 次元離散型実フーリエ変換 (2,3,5 及び 7 の混合基底)	549
DM_V2DRCF	2 次元離散型実フーリエ変換 (2,3,5 及び 7 の混合基底)	552
DM_V3DRCF	3 次元離散型実フーリエ変換 (2,3,5 及び 7 の混合基底)	556
DM_V3DRCF2	3 次元離散型実フーリエ変換 (2,3,5 及び 7 の混合基底)	560
DM_V3DCPF	3 次元素因子離散型複素フーリエ変換	564

乱数

サブルーチン名	項　　目	ページ
DM_VRANU4	一様乱数[0,1)の生成	259
DM_VRANU5	一様乱数[0, 1)の生成 (MRG8)	265
DM_VRANN3	正規乱数の生成	249
DM_VRANN4	正規乱数の生成 (Wallace 法)	254

2. 逐次 SSL II 機能をスレッド並列化したルーチン

逐次 SSL II 機能を並列化した以下の機能については逐次 SSL II のマニュアルを参照してください. 引数の与え方は同一であり, ルーチン名は倍精度ルーチン名の先頭文字の“D”を, “DM_”に変えたものになっています.

2.1 標準機能

標準機能ルーチンをスレッド並列化した以下のルーチンの使用方法については,「富士通 SSL II 使用手引書」を参照してください.

行列操作

サブルーチン名	項　　目
DM_MAV	実行列と実ベクトルの積
DM_MCV	複素行列と複素ベクトルの積

最小二乗解

サブルーチン名	項　　目
DM_LAXL	実行列の最小二乗解 (ハウスホルダー変換)
DM_LAXLM	実行列の最小二乗最小ノルム解 (特異値分解法)
DM_GINV	実行列の一般逆行列 (特異値分解法)
DM_ASVD1	実行列の特異値分解 (ハウスホルダー法, QR 法)

固有値固有ベクトル

サブルーチン名	項　　目
DM_EIG1	実行列の固有値及び固有ベクトル (2 段 QR 法)
DM_HSQR	実ヘッセンベルグ行列の固有値 (2 段 QR 法)
DM_HVEC	実ヘッセンベルグ行列の固有ベクトル (逆反復法)
DM_CHVEC	複素ヘッセンベルグ行列の固有ベクトル (逆反復法)
DM_BLNC	実行列の平衡化
DM_CBLNC	複素行列の平衡化
DM_HES1	実行列の実ヘッセンベルグ行列への変換 (ハウスホルダー法)
DM_CHES2	複素行列の複素ヘッセンベルグ行列への変換 (安定化基本相似変換)
DM_HBK1	実行列の固有ベクトルへの逆変換と正規化
DM_CHBK2	複素行列の固有ベクトルへの逆変換
DM_NRML	実行列の固有ベクトルの正規化
DM_CNRML	複素行列の固有ベクトルの正規化

2.2 拡張機能

拡張機能ルーチンをスレッド並列化した以下のルーチンの使用方法については，「FUJITSU SSL II 拡張機能使用手引書 II」を参照してください.

固有値・固有ベクトル

サブルーチン名	項　　目
DM_VLAND	実対称スパース行列の固有値・固有ベクトル (Lanczos 法, 対角形式格納法)

変換

サブルーチン名	項　　目
DM_VMCST	離散型 cosine 変換
DM_VMSNT	離散型 sine 変換
DM_VCCVF	複素データの離散畳み込みおよび離散相関
DM_VRCVF	実データの離散畳み込みおよび離散相関

目　　次

注)　＊については, まえがき参照

第I部　概説

第1章　SSL II スレッド並列機能の概要

SSL II スレッド並列機能は, 共有メモリ型スカラ並列計算機上で並列動作する数値計算ライブラリであり, 単一 CPU では処理できないような大規模な問題を並列処理することで効率よく計算するためのサブルーチンを提供しています. サブルーチンは CALL 文によって呼び出します.

“スレッド並列”とはひとつのプロセスの中で, スレッドと呼ばれる“実行の流れ”を複数個発生させ, 各スレッドは共用メモリ内にあるひとつの CPU を使って計算の一部を担います. CPU の個数が発生させたスレッド の個数分だけあれば, 各スレッドは異なる CPU に割り当てられて並列に実行されます. このように,一つの計算(ただし並列化できる計算)を, 複数スレッドに分担して並列に処理する方式をスレッド並列と呼びます.

SSL II スレッド並列機能の各サブルーチンは内部でスレッドを複数発生して並列化できる計算を, スレッドで並列に計算するライブラリであると言えます. ここで, スレッドの発生, 計算の分担, 同期化, そして消滅は OpenMP Fortran の仕様で記述しています. したがって, SSL II スレッド並列機能のサブルーチンを利用するには, OpenMP Fortran の実行環境を必要とします.

また, サブルーチン内部で発生させるスレッドの個数は, 利用者が OpenMP Fortran の環境変数もしくは実行時ライブラリルーチンで指定できます. したがって , サブルーチンは与えられた任意のスレッド数で動作することができます.

なお, SSL II スレッド並列機能は従来の SSL II や, ベクトル並列計算機の単一プロセッサー用の SSL II/VP, 複数プロセッサー用の SSL II/VPP, SSL II/HPF とは機能範囲, サブルーチン名, 及び呼出し方法が異なります.

第2章　SSL II スレッド並列機能の一般規約

2.1　サブルーチンの演算精度

SSL II スレッド並列機能は, 倍精度で演算を行うサブルーチンをサポートしています.

2.2　サブルーチン名

利用者が CALL 文で呼び出せるサブルーチンの名前は DM_V で始まり, サブルーチン内部で呼ばれるスレーブサブルーチンは DM_U または DL_で始まります. その他, 補助ルーチンとして DMACH があります.

本マニュアルでは, 利用者が呼び出せるサブルーチンの使用方法についてのみ記述します.

2.3　パラメタ

(1) パラメタの並びの順

パラメタの並びの順序は従来の SSL II と同じく, 一般に以下の順に従っています.

(*入出力パラメタの並び*, *入力パラメタの並び*, *出力パラメタの並び*, *ICON*)

(2) パラメタの型

先頭が I, J, K, L, M, N で始まるものは 4 バイト整数型です. その他の文字で始まるものは, 倍精度実数, あるいは倍精度複素数型です.

2.4　SSL II スレッド並列機能の利用方法

(1) サブルーチン呼出しの位置

SSL II スレッド並列機能は OpenMP Fortran のサブルーチンであり, 利用者プログラムでは OpenMP Fortran のパラレルリージョン(Parallel Region) の外および内から呼び出すことができます. また, OpenMP Fortran 文のない逐次プログラムからも呼び出すことができます. さらに自動並列化されたプログラムから呼び出すこともできます.

パラレルリージョンの中から呼び出すときは, パラレルリージョンを実行する各スレッドに対して異なる領域を入出力, 出力および作業域として利用する実引数に指定する必要があります.

上記のいずれの場合も SSL II スレッド並列機能を利用する利用者プログラムを翻訳した後結合するとき, frt コマンドに-Kopenmp オプションを指定します. このオプションにより OpenMP Fortran の実行環境で実行するロードモジュールとなります. 詳細は“Fortran 使用手引書”を参照してください.

(2) スレッド数の指定方法

SSL II スレッド並列機能のサブルーチンの実行は, サブルーチンの内部のパラレルリージョンで多重スレッドによって並列に行われます. 生成されるスレッド数は OpenMP Fortran 仕様で規定された, 環境変数 OMP_NUM_THREADS もしくは実行時ライブラリルーチン (以下ライブラリルーチンと略します.) OMP_SET_NUM_THREADS( )で指定することができます. 普通は,前者の方法でスレッド数を指定します.

ライブラリルーチンでスレッド数を指定するのは, 直後のパラレルリージョンに対して, 特定のスレッド数で実行させる場合に使います. この方法によって, SSL II スレッド並列機能のサブルーチンを呼び出す前に指定すれば, 特定のスレッド数でサブルーチンを実行させることもできます.

OpenMP Fortran の環境変数および実行時ライブラリルーチンの詳細は “Fortran 使用手引書” および “OpenMP Application Program Interface Version 2.5 May 2005” を参照してください.

(3) 各スレッドのスタック領域の大きさ

SSL II スレッド並列機能のサブルーチンの内部で, 各スレッド毎の作業域を自動割付け配列で割り付けています. 生成するスレッドの数を NT, 利用できるメモリ量を M としたとき , 各スレッドのスタック領域の大きさとして環境変数 OMP_STACKSIZE に M/(5*NT) 程度を指定して実行することを勧めます. -Nfjomplib オプションが指定されている場合, 環境変数 THREAD_STACK_SIZE にスタック領域の大きさを指定する事もできます。

OpenMP Fortran の実行環境で動作するプログラムに対するスタックサイズの指定の詳細に関しては, “Fortran 使用手引書” を参照して下さい.

(4) 利用例プログラム

ａ. パラレルリージョンの外側からの呼出し

環境変数 OMP_NUM_THREADS を 4 に設定して以下のプログラムを 4 プロセッサーのシステムで実行すると 4000×4000 の実係数行列の連立 1 次方程式を 4 つのスレッドで並列に解きます.

```
      implicit real*8 (a-h,o-z)
      parameter(nord=4000,ld=nord+1)
c
      real*8  a(ld,nord),b(nord)
      integer ip(nord),is
c
      c=sqrt(2.0d0/dble(1+nord))
      t=datan(1.0d0)*4./(1+nord)
c
      do j=1,nord
      do i=1,nord
      a(i,j)=c*sin(t*i*j)
      enddo
      enddo
c
      do i=1,nord
      s=0.
      do j=1,nord
      s=s+sin(t*i*j)
      b(i)=s*c
      enddo
      enddo
c
      k=ld
      n=nord
      epsz=0.0d0
      isw=1
      call dm_vlax(a,k,n,b,epsz,isw,is,ip,icon)
      print*,'icon=',icon
      print*,'n=',n,', b(1)=',b(1),', b(n)=',b(n)
      stop
      end
```

例 2.1　SSL II スレッド並列機能のルーチンの利用例

b．パラレルリージョンの内側からの呼出し

以下の例は, 独立した 2 組の実係数行列の連立 1 次方程式を解きます. 環境変数 OMP_NUM_THREADS を 2 に,そして環境変数 OMP_NESTED を TRUE に設定して以下のプログラムを 4 プロセッサーのシステムで実行します. このとき 4000×4000 の実係数行列の連立 1 次方程式を 2 つのスレッドで, 4200×4200 の実係数行列の連立 1 次方程式を 2 つのスレッドで合計 4 つのスレッドを使って並列に解きます.

```
      implicit real*8 (a-h,o-z)
      parameter(nord1=4000,ld1=nord1+1)
      parameter(nord2=4200,ld2=nord2+1)
c
      real*8  a1(ld1,nord1),b1(nord1),
     &        a2(ld2,nord2),b2(nord2),epsz1,epsz2
      integer ip1(nord1),ip2(nord2),is1,is2,
     &        icon1,icon2,n1,n2,k1,k2,num,
     &        omp_get_thread_num
c
      c=sqrt(2.0d0/dble(1+nord1))
      t=datan(1.0d0)*4./(1+nord1)
c
      do j=1,nord1
      do i=1,nord1
      a1(i,j)=c*sin(t*i*j)
      enddo
      enddo
c
      do i=1,nord1
      s=0.
      do j=1,nord1
      s=s+sin(t*i*j)
      b1(i)=s*c
      enddo
      enddo
c
      c=sqrt(2.0d0/dble(1+nord2))
      t=datan(1.0d0)*4./(1+nord2)
c
      do j=1,nord2
      do i=1,nord2
      a2(i,j)=c*sin(t*i*j)
      enddo
      enddo
```

```
c
      do i=1,nord2
      s=0.
      do j=1,nord2
      s=s+sin(t*i*j)
      b2(i)=s*c
      enddo
      enddo
c
!$OMP PARALLEL default(shared)
!$OMP+      private(num)
      num=omp_get_thread_num()
      if(num.eq.0)then
      k1=ld1
      n1=nord1
      epsz1=0.0d0
      isw1=1
      call dm_vlax(a1,k1,n1,b1,epsz1,isw1,is1,ip1,icon1)
      print*,'icon1=',icon1
      else
      k2=ld2
      n2=nord2
      epsz2=0.0d0
      isw2=1
      call dm_vlax(a2,k2,n2,b2,epsz2,isw2,is2,ip2,icon2)
      print*,'icon2=',icon2
      endif
!$OMP END PARALLEL
      print*,'n1=',n1,', b1(1)=',b1(1),', b1(n1)=',b1(n1)
      print*,'n2=',n2,', b2(1)=',b2(1),', b2(n2)=',b2(n2)
      stop
      end
```

例 2.2　SSL II スレッド並列機能のルーチンの利用例

2.5　コンディションコード

SSL II スレッド並列機能の実行後の状態を示すパラメタ ICON が用意されています.
コードは 0 から 49999 の値をとりますが, 結果が保証されているか否かに応じて表 2.1 のように区分されています.

表 2.1　コンディションコードの区分

コード	意　　味	結果の保障	区分
0	正常に処理が終了している.	結果は保障される.	正常
1〜9999	正常に処理が終了しているが, なんらかの補助的な情報を含んでいる.		
10000〜19999	処理の過程で, 内部的な制限を加えることにより, 一応の処理は終了している.	制限付きで結果は保障される.	警告
20000〜29999	処理の過程で異常が生じ, 処理が打ち切られた.	結果は保障されない.	異常
30000〜49999	入力パラメタにエラーがあったため, 処理が打ち切られた.		

第II部　サブルーチンの使用方法

DM_VALU

実行列の LU 分解 (ブロック化された LU 分解法)
CALL DM_VALU(A, K, N, EPSZ, IP, IS, ICON)

(1) 機能

$n \times n$ の正則な実行列をブロック化した外積形ガウスの消去法により LU 分解します.

$$\boldsymbol{PA} = \boldsymbol{LU}$$

ただし, $\boldsymbol{P}$ は部分ピボッティングによる行の入換えを行う置換行列, $\boldsymbol{L}$ は下三角行列, $\boldsymbol{U}$ は単位上三角行列です. ($n \geq 1$)

(2) パラメタ

A.....入力. 行列 $\boldsymbol{A}$ を A(1:N,1:N)に格納してください.
出力. 行列 $\boldsymbol{L}$ と行列 $\boldsymbol{U}$ が A(1:N,1:N)に格納されます.
図 DM_VALU-1 参照.
A(K,N)なる倍精度実数型 2 次元配列.

K.....入力. 格納配列 A の 1 次元目の大きさ(≥ N).

N.....入力. 行列 $\boldsymbol{A}$ の次数 n.

EPSZ.....入力. ピボットの相対零判定値(≥ 0.0)
0.0 のときは標準値が採用されます.
((3)使用上の注意 a. 注意①参照)

IP.....出力. 部分ピボッティングによる行の入換えの履歴を示すトランスポジションベクトル. 大きさ n の 1 次元配列.
((3)使用上の注意 a. 注意②参照)

IS.....出力. 行列 $\boldsymbol{A}$ の行列式を求めるための情報. 演算後の配列 A(1:N,1:N)の n 個の対角要素と IS の値を掛け合わせると行列式が得られます.

ICON.....出力. コンディションコード.
表 DM_VALU-1 参照.

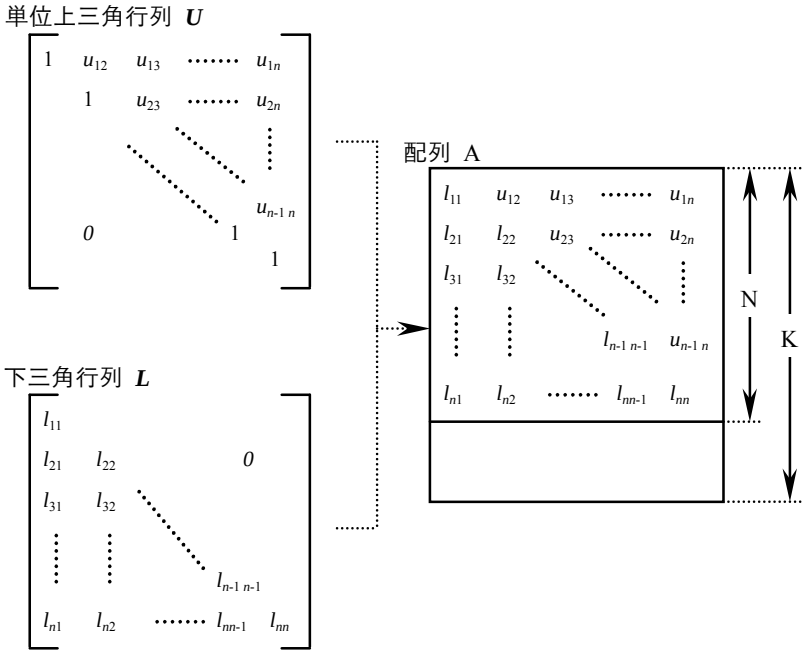


図 DM_VALU-1　演算後の配列 A における ***L*** 及び ***U*** の格納方法

LU 分解された結果の行列は, ***U*** の対角要素を除いた上三角部分と, ***L*** の下三角部分が, 配列 A(1:N,1:N)に格納されます.

表 DM_VALU-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし.	―
20000	行列 ***A*** のある行の要素がすべて零であったか, またはピボットが相対的に零となった. 行列 ***A*** は非正則の可能性が強い.	処理を打ち切る.
30000	K＜N, N＜1, EPSZ＜0.0	

a.　注意

①　EPSZ に値を設定したとすると, この値は次の意味をもっています. すなわち, 選択されたピボット要素の絶対値が EPSZ 以下なら, そのピボットを零と見なし, ICON＝20000 として処理を打ち切ります. EPSZ の標準値は丸め誤差単位 u としたとき, EPSZ＝16 u です. なおピボットが小さくなっても計算を続行させる場合には, EPSZ へ極小の値を与えればよいのですが, その結果は保障されません.

②　トランスポジションベクトルとは, 部分ピボッティングを行う LU 分解

PA＝LU

における置換行列 P に相当します.

本サブルーチンでは, 部分ピボッティングに伴い, 配列 A の内容を実際に交換しています.

すなわち, 分解の J 段階(J = 1, ... , *n*)において第 I 行(I ≥ J)がピボット行として選択された場合には, 配列 A の第 I 行と第 J 行の内容が交換されています. そしてその履歴を示すために IP(J)に I が格納されます.

③ 本サブルーチンに続けて, サブルーチン DM_VLUX を呼び出すことにより, 連立 1 次方程式を解くことができます. 通常はサブルーチン DM_VLAX を呼び出せば, 一度に解が求まります.

b. 使用例

4000 × 4000 の実行列の LU 分解を行います.
(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(4001,4000)
      DIMENSION IP(4000)
C
C
      N=4000
!$OMP PARALLEL DO DEFAULT(PRIVATE) SHARED(A,N)
      DO J=1,N
      DO I=1,N
      A(I,J)=MIN(I,J)
      ENDDO
      ENDDO
!$OMP END PARALLEL DO
C
      K=4001
      CALL DM_VALU(A,K,N,0.0D0,IP,IS,ICON)
      WRITE(6,610)ICON
      IF(ICON.GE.20000)STOP

      S=1.0D0
!$OMP PARALLEL DO DEFAULT(PRIVATE) SHARED(A,N)
!$OMP+         REDUCTION(*:S)
      DO 20 I=1,N
      S=S*A(I,I)
   20 CONTINUE
!$OMP END PARALLEL DO

C
```

```
      DET=IS*S

   40 CONTINUE
      WRITE(6,620)DET
  610 FORMAT(1H0,10X,16HCONDITION CODE =,I5)
  620 FORMAT(1H0,10X,
     *27HDETERMINANT OF THE MATRIX =,D23.16)
      END
```

(3)　手法概要

ブロック化された外積形の LU 分解法については “付録 1 参考文献一覧表” の[1], [30], [54], [55], [56]及び[70]を参照して下さい.

DM_VAMLID

スパースな M-行列の連立 1 次方程式 (代数的マルチレベル反復法[AMLI 法], 対角形式格納法)
CALL DM_VAMLID(A, K, NDIAG, N, NOFST, B, ISW, IGUSS, INFO, EPSOT, EPSIN, X, W, NW, IW, NIW, ICON)

(1) 機能

$n \times n$ の M-行列のスパース行列を係数行列とする連立 1 次方程式を反復法で解きます(M-行列については, (3)使用上の注意 a. 注意①参照).

$$\boldsymbol{Ax} = \boldsymbol{b}$$

$n \times n$ の係数行列 $\boldsymbol{A}$ は, 対角形式の格納法で格納します. ベクトル $\boldsymbol{b}$ および $\boldsymbol{x}$ は n 次元ベクトルです.

反復法は行列 $\boldsymbol{A}$ が対称のときは, ORTHOMIN 法が, 非対称のときは, GMRES 法が使用できます. この反復解法(以下外部反復と呼びます)には代数的マルチレベル法(AMLI 法)による前処理が施されていて, 安定な解法です. 代数的マルチレベル法による前処理では縮小された連立 1 次方程式も反復法(以下内部反復と呼びます)で解かれます. (レベルが深くなるほど, 縮小された連立 1 次方程式の次数は小さくなります.)

(2) パラメタ

A.....入力. 係数行列 $\boldsymbol{A}$ の非零要素を A(1:N, 1:NDIAG)に格納します.
A(K, N)なる 2 次元配列.
対角形式の格納方法については, “SSL II 拡張機能使用手引書 II 第 I 部 概説 3.2.1.1 一般スパース行列の格納方法 b.一般スパース行列の対角形式の格納方法”を参照してください.

K.....入力. 配列 A の 1 次元目の大きさ(≥ N).

NDIAG.....入力. 係数行列 $\boldsymbol{A}$ の非零要素を含む対角ベクトル列の総数.

N..... 入力. 行列 $\boldsymbol{A}$ の次数 n.

NOFST.....入力. 配列 A に格納される対角ベクトルに対応した主対角ベクトルからの距離を格納します. 上対角ベクトル列は正, 下対角ベクトル列は負の値で表します. NOFST(NDIAG)なる 1 次元配列.

B.....入力. 連立 1 次方程式の右辺の定数ベクトルを B(1:N)に格納します.
B(N)なる 1 次元配列.

ISW .....入力. 制御情報.
ISW=1 のとき, 初回の呼び出し.
ISW=2 のとき, 2 回目以降の呼び出し. このとき, A, IW, W の 値は, 初回の呼び出しの結果を変更せずに呼び出す必要があります.
((3)使用上の注意 a. 注意②参照)

IGUSS.....入力. 配列 X に指定された解ベクトルの近似値から反復計算を行うかを指定する制御情報.

IGUSS = 0 のとき, 解ベクトルの近似値を指定しません. このときゼロベクトルから反復を開始します.

IGUSS ≠ 0 のとき, 配列 X に指定された解ベクトルの近似値から反復計算を開始します.

INFO .....入出力. 反復に関する制御情報.

INFO(14)なる 1 次元配列.

行列 ***A*** が対称な場合の指定例.

INFO(1) = -1
INFO(2) = NTHRD × 100
INFO(3) = 0
INFO(5) = 1
INFO(6) = 2000
INFO(10) = 1
INFO(11) = 1000

行列 ***A*** が非対称な場合の指定例.

INFO(1) = -1
INFO(2) = NTHRD × 100
INFO(3) = 0
INFO(5) = 2
INFO(6) = 2000
INFO(7) = 5
INFO(8) = 20
INFO(10) = 2
INFO(11) = 1000
INFO(12) = 10
INFO(13) = 0

NTHRD は, 並列実行するスレッドの数.

INFO(1) = MAXLVL

入力. 代数的マルチレベル法のレベルの最大値.

MAXLVL < 0 のとき, 本ルーチンが評価した最適レベルで行います.

MAXLVL = 0 のとき, マルチレベル法は使われません.

MAXLVL > 0 のとき, 指定された値以上の粗いレベルは使われません.

((3)使用上の注意 a. 注意⑥, ⑦参照)

INFO(2) = MINUK

入力. もっとも深いレベルの縮小された連立 1 次方程式の未知数の最小個数. MINUK は N よりずっと小さく, 並列処理するスレッドの数 NTHRD よりずっと大きな値を指定することをすすめます. 例えば, NTHRD × 100.

INFO(3) = NORM

入力. 行列の正規化の方法を指定します.

NORM < 1 のとき, 行列 ***A*** は ***A*** の対角要素の平方根の逆数を対角要素として持つ対角行列で左右から正規化されます. 対称行列を正規化した行列は対称行列です.

対称な正規化をすすめます. しかし, ある場合は, 非対称な正規化の方が速く収束することもあり得ます.
((3)使用上の注意 a. 注意④参照)
NORM ≥ 1 のとき, 行列 ***A*** は ***A*** の対角要素の逆数からなる対角行列で左から正規化します. ***A*** が対称行列でも正規化されたあとは対称行列ではありません.
((3)使用上の注意 a. 注意⑤参照)

INFO(4)
出力. 使用されたレベル数.

INFO(5) = METHOT
入力. 外部反復で使われる反復法.
METHOT=1 のとき, 前処理付き ORTHOMIN 法が使われます. 行列 ***A*** が対称で対称な正規化を指定した場合に指定してください.
METHOT≠1 のとき, 切り捨て再起動する GMRES 法が使われます. 行列 ***A*** が非対称であるか, 非対称な正規化が使われた場合に指定してください.

INFO(6) = ITMXOT
入力. 外部反復の最大回数. 例えば, 2000.
外部反復回数がこの値に達したとき, 処理を打ち切ります. このとき返却された解は, 保証されません.

INFO(7) = NRESOT
入力. 外部反復で使われる反復法に GMRES 法を指定したとき, 外部反復の直交手続きで使う残差ベクトルの本数を指定します. 例えば 5. つまり GMRES 法で NRESOT 個のベクトルを使い解を更新し, 残りのベクトルによる更新はされません.
((3)使用上の注意 a. 注意⑤参照)

INFO(8) = NRSTOT
入力. 外部反復で使われる反復法に GMRES 法を指定したとき, 再起動(restart)する反復回数を指定します. 例えば 20 回.
NRSTOT < 1 のとき, 再起動は行いません.
((3)使用上の注意 a. 注意⑤参照)

INFO(9) = ITEROT
出力. 外部反復の回数.

INFO(10) = METHIN
入力. 内部反復で使われる反復法.
METHIN = 1 のとき, 前処理付き ORTHOMIN 法が使われます. 行列 ***A*** が対称で対称な正規化を指定した場合に指定してください.
METHIN ≠ 1 のとき, 切り捨て再起動する GMRES 法が使われます. 行列 ***A*** が非対称であるか, 非対称な正規化が使われた場合に指定してください.

INFO(11) = ITMXIN
入力. 内部反復の最大回数. 例えば, 1000.
内部反復回数がこの値に達すると, 外部反復に処理は引き継がれます.

INFO(12) = NRESIN

入力. 内部反復で使われる反復法に GMRES 法を指定したとき, 内部反復の直交手続きで計算する残差ベクトルの数を指定します. 例えば 10. つまり GMRES 法で NRESOT 個のベクトルを使い解を更新し, 残りのベクトルによる更新はされません.

((3)使用上の注意 a. 注意⑤参照)

INFO(13) = NRSTIN

入力. 内部反復で使われる反復法に GMRES 法を指定したとき, 再起動(restart)する反復回数を指定します.

NRSTIN < 1 のとき, 再起動は行いません.

((3)使用上の注意 a. 注意⑤参照)

INFO(14)

出力. 内部反復の平均回数.

EPSOT .....入力. 解の収束判定値.

外部反復の k ステップで求められた解ベクトルが正規化された残差ベクトル $\hat{r}_k = \hat{A}x_k - \hat{b}_k$ に対して以下の条件を満たしたときベクトルは収束したと見なします. $\| \hat{r}_k \| \le \mathrm{EPSOT} \| \hat{b} \|$, $\| y \|^2 = y^{\mathrm{T}} y$ なるユークリッドノルムであり, $\hat{\boldsymbol{A}}$ および $\hat{b}$ は正規化された係数行列および右辺ベクトルです.

EPSIN .....入力. 内部反復での解の収束判定値.

たいていの場合 10^{-3} が最適です.

X.....出力. X(1:N)に解ベクトルが格納されます.

X(N)なる 1 次元配列.

W.....作業域. W(NW)なる 1 次元配列.

NW.....入力. 作業用配列 W の大きさ.

$$\mathrm{NW} \ge \mathrm{NT} \times (3 \times \mathrm{NAMAX} + 5) + 3 \times (\mathrm{NLVL} + 1) \times \mathrm{NBAND} \times \mathrm{MAXT} + \mathrm{MAX}\,(\mathrm{NAMAX} \times \mathrm{NT},\ 7 \times \mathrm{NT} + \mathrm{LR0(NT)})$$

NT = N+MAXT, MAXT は本ルーチンを並列実行する最大スレッド数.

NBAND は下バンド巾と上バンド巾の最大値.

NLVL は代数的マルチレベル法のレベルの数. INFO(1)<0 のときは 10.

NAMAX ≥ NDIAG. ((3)使用上の注意 a. 注意③参照)

ただし, ORTHOMIN 法を使用したとき, LR0(NT) = 4 × NT.

NRES 個の残差の計算のあと切り捨てる GMRES 法を使ったとき,

LR0(NT) = (2 × NRES+1) × NT

NRES = MAX(NRESOT, NRESIN)とすれば十分です.

IW .....作業域. IW(NIW)なる 1 次元配列.

NIW .....入力. 作業用配列 IW の大きさ.

$$\mathrm{NIW} \ge \mathrm{MAXT} \times ((6 \times \mathrm{MAXT} + 12 \times \mathrm{NAMAX}) \times (\mathrm{NLVL}+1) + 8 \times \mathrm{NBAND} + 3000) + 4 \times (\mathrm{N}+\mathrm{MAXT})$$

MAXT は本ルーチンを並列実行する最大スレッド数.

NBAND は下バンド巾と上バンド巾の最大値.

NLVL は代数的マルチレベル法レベルの数. INFO(1)<0 のときは 10.

NAMAX ≥ NDIAG. ((3)使用上の注意 a. 注意③参照)

ICON.....出力. コンディションコード.

表 DM_VAMLID-1 参照.

表 DM_VAMLID-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし	ー
10700	ベクトル v^{pos} が見つからなかった.	v^{pos}=(1, 1, …, 1)として処理を続ける.
10800	GMRES 法で回復可能な break down を起こした.	処理を続ける.
20001	反復回数の上限に達した.	処理を打ち切る. 配列 X には, そのときまでに得られている近似値を出力するが, 精度は保証できない.
20003	GMRES 法で回復不可能な break down を起こした.	処理を打ち切る.
20005	ORTHOMIN 法で $\boldsymbol{p}^{\mathrm{T}} \cdot \boldsymbol{A} \cdot \boldsymbol{p} = 0$ となり回復不可能な break down を起こした.	
20006	ORTHOMIN 法で $\boldsymbol{p}^{\mathrm{T}} \cdot \boldsymbol{r} = 0$ となり回復不可能な break down を起こした.	
30000	N<1, N>K, NDIAG<1, ISW<1, ISW>2	
30104	|NOFST(I)| > N‐1	
30105	主対角要素がない.	
30200	係数行列が M-行列でない.	
30210	行列の圧縮が非正値であるためできない.	
30212	対角要素にゼロ要素がある.	
30310	NIW が小さ過ぎる.	
30320	NW が小さ過ぎる.	

(3)　使用上の注意

a.　注意

①　楕円型の境界値問題を 2 次または 1 次の有限差分離散化を行って生成した係数行列は M-行列です. これらは, 2 階偏微分方程式を離散化するルーチン(DM_VPDE2D, DM_VPDE3D)で生成することができます.

M-行列とは, 以下の条件を意味します.

- すべての対角要素は正 ($a_{i,i} > 0$,　$i = 1, \ldots, \mathrm{N}$), かつ他の要素は非正値 ($a_{i,j} \le 0$,　$i, j = 1, \ldots, \mathrm{N}, i \ne j$).

- 正値のベクトル v^{pos} があって Av^{pos} が正値.
  最初の条件を満たさないとき, また本ルーチンが v^{pos} を見つけることができなかった場合(ICON=10700), 本ルーチンは v^{pos} = (1, ... , 1)として breakdown のリスクをもって処理を続けます. この場合, 外部反復または内部反復で収束が遅かったり, breakdown を起こしたりする可能性(ICON=30212, 30210)があります.
  粗いレベルを定義するために, 係数行列を組み立てるために使われた矩形のグリッドを再構成します. 再構成に失敗したときは, breakdown するか, 粗いレベルの行列の対角ベクトルが過度に増えて収束が遅くなったり, 外部または内部反復で breakdown を起こしたりする可能性があります.

② 同じ係数行列を持つ右辺の異なる多重組みの連立 1 次方程式を解く場合は, 最初の呼び出しで ISW=1 で呼び出して解き, 2 回目以降の呼び出しでは ISW=2 として呼び出すことで解けます. 最初の呼び出しで組み立てられたあらいレベルに対応する行列が 2 回目以降の呼び出しで再利用されます.

③ たいていの場合は, NAMAX = NDIAG で十分です. 与えられた係数行列の対角ベクトルの数よりあらいレベルの行列の対角ベクトルの数が大きい場合が起こり得ます. このとき NAMAX は増やさなければなりません.

④ 行列が対称のとき, ORTHOMIN 法を使うことをすすめます. 本ルーチンは計算のできる未知数を反復が始まる前に行列から除きます. 与えられた行列が対称でなくても実際の反復計算では行列が対称であることが起こり得ます. そのため, 行列が対称となる可能性がある場合は, 最初に対称な正規化を行う ORTHOMIN 法で解くことを試みることをすすめます.

⑤ 行列が対称でない場合は, 非対称な正規化を行って GMRES 法を使うことをすすめます. たいていの場合, 外部反復で NRESOT = 5 として切り捨て 20 ステップ後に再起動すれば十分です. 内部反復では, NRESIN として切り捨てる数をもっと大きくし, もっと大きなステップの後に再起動するか再起動を禁じることが必要です. NRESIN を大きくすると作業域を大きくする必要があります. このため作業域の大きさの計算式の中の NRES を大きくしなければなりません. しかし NRES は NRESOT より小さくできます.
  一般に, NRESIN および NRESOT を大きくすると GMRES 法の精度はよくなりますが, 計算およびメモリ使用量も増えます.

⑥ 本ルーチンは最適なレベルを見つけようとします. 問題によっては, レベル数を指定することで計算時間が減ることもあります. しかし, たいていの場合改善はさほど大きくありません.

⑦ 各レベルの行列を A_n(n= 1, ... , MAXLVL –1)としたとき,

$$A_n = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

$S = A_{22} - A_{21}A_{11}^{-1}A_{12}$ を Schur complement と呼びます. 擬似格子による変数の並びを利用して, 消去する適当な変数の集合を選ぶことでこのようなブロックを形成します. A_{11} の逆行列を対角行列で近似します. この近似的逆行列を使って

Schur complement を計算し, レベル n+1 の行列 A_{n+1} とします.
Schur complement を使い A_n は次のように分解できます.

$$A_n = \begin{bmatrix} A_{11} & 0 \\ A_{21} & I \end{bmatrix} \begin{bmatrix} I & A_{11}^{-1}A_{12} \\ 0 & S \end{bmatrix}$$

A_{11}^{-1} の近似を利用して, 各レベルをこのように近似的に分解し前処理として利用します.

b. 使用例
偏微分方程式を領域 $[0, 1]^2$ で離散化して解きます.

$$-\left(\frac{\partial^2 u}{\partial^2 x_1} + \frac{\partial^2 u}{\partial^2 x_2}\right) + cu = 1$$

Dirichlet 境界条件 $u = 0$ を設定します.
(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT NONE
      INTEGER   MAXT,N1,N2,KA,NA,NLVL,L1,L2,NW,NIW

      PARAMETER(MAXT=4,N1=1281,N2=1537,NLVL=10,
     &          L1=N1,L2=N2,
     &          KA=N1*N2,NA=5,
     &          NW=(3*NA+5)*(KA+MAXT)+3*(NLVL+1)*N1*MAXT
     &             +11*(KA+MAXT),
     &          NIW=((6*MAXT+12*NA)*(NLVL+1)
     &             +8*N1+2000)*MAXT+4*(KA+MAXT))

      INTEGER          NOFST(NA),INFO(100),IW(NIW)
      DOUBLE PRECISION X1(L1),X2(L2),
     &                 A1(L1,L2),A2(L1,L2),B1(L1,L2),B2(L1,L2),
     &                 C(L1,L2),F(L1,L2),
     &                 W(NW),EPSIN,EPSOT

      DOUBLE PRECISION A(KA,NA),B(KA),U(KA),SOL(3*N1*N2),
     &                 RHS(N1*N2),RHSC(N1*N2),TMP

      INTEGER Z1,Z2,NDIAG,N,ICON,ISW,IGUSS,I,Z,NBAND
C
```

```
C-----------------------------------------------------------------
C
C***** CREATE NODE COORDINATES
C
      DO 11 Z1=1,N1
        X1(Z1)=DBLE(Z1-1)/DBLE(N1-1)
11    CONTINUE
      DO 12 Z2=1,N2
        X2(Z2)=DBLE(Z2-1)/DBLE(N2-1)
12    CONTINUE
C
C***** COEFFICIENTS IN THE PARTIAL DIFFERENTIAL EQUATION :
C
      DO 2000 Z2=1,N2

        DO 20 Z1=1,N1
          A1(Z1,Z2)=1
          A2(Z1,Z2)=1
          B1(Z1,Z2)=0
          B2(Z1,Z2)=0
          C (Z1,Z2)=1
          F (Z1,Z2)=1
20    CONTINUE
C
C***** DIRICHLET BOUNDARY CONDITIONS:
C
      C(1,Z2)=1
      F(1,Z2)=0
      C(N1,Z2)=1
      F(N1,Z2)=0
      IF (Z2.EQ.1) THEN
        DO 25 Z1=1,N1
          C(Z1,1)=1
          F(Z1,1)=0
 25     CONTINUE
      END IF
      IF (Z2.EQ.N2) THEN
        DO 26 Z1=1,N1
          C(Z1,N2)=1
          F(Z1,N2)=0
 26     CONTINUE
      END IF
```

```
2000  CONTINUE

      N=N1*N2
      CALL DM_VPDE2D(A1,L1,N1,N2,A2,X1,X2,B1,B2,C,F,A,KA,NA,N,
     &               NDIAG,NOFST,B,ICON)
      PRINT*,'ICON OF DM_VPDE2D =',ICON
      IF (ICON.GT.29999) GOTO 9999
C
      DO Z=1,N
      RHS(Z)=B(Z)
      ENDDO
      NBAND=0
      DO I=1,NDIAG
      NBAND=MAX(NBAND,ABS(NOFST(I)))
      ENDDO
C
C-----------------------------------------------------------------
C
C**** CALL DAMLI:
C
      ISW=1
      IGUSS=0
C
      INFO(1)=-1
      INFO(2)=MAXT*100
      INFO(3)=0
      INFO(5)=1
      INFO(6)=2000
      INFO(10)=1
      INFO(11)=1000
C
      EPSOT=1.D-6
      EPSIN=1.D-3

      CALL DM_VAMLID(A,KA,NDIAG,N,NOFST,B,ISW,IGUSS,
     &     INFO,EPSOT,EPSIN,U,W,NW,IW,NIW,ICON)
      PRINT*,'ICON OF DM_VAMLID = ',ICON
      IF (ICON.GT.29999) GOTO 9999
C
9999  CONTINUE
C
      DO I=1,NBAND
```

```
      SOL(I)=0.0D0
      SOL(NBAND+N+I)=0.0D0
      ENDDO
      DO Z=1,N
      SOL(NBAND+Z)=U(Z)
      ENDDO
      CALL DM_VMVSD(A,KA,NDIAG,N,NOFST,NBAND,SOL,RHSC,ICON)
      TMP=0
      DO Z=1,N
      TMP=MAX(TMP,ABS((RHS(Z)-RHSC(Z))/(RHS(Z)+1.0)))
      ENDDO
C
      PRINT*,' ERROR = ',TMP
C
      END
```

(4)　手法概要

計算を始めるまえに, 線型系は主対角要素が 1 になるように正規化されます. 主対角要素より外側にある値がゼロの要素(Dirichlet 境界条件から派生する)は行列から除かれます. 正規化された系は前処理付きの ORTHOMIN 法か GMRES 法で(“付録 1 参考文献一覧表”の[79]参照.)解かれます. AMLI 法は, 近似的 Schur complements を利用して再帰的にブロック不完全分解を行う方法(“付録 1 参考文献一覧表”の[6]参照.)に基づいています. 同時に消去される変数の集合は, 行列の対角ベクトルから, 擬似格子を再構成してから, 方向を交互にとる方法で定義されます. 最もあらいレベルの線型系は正規化されて, ORTHOMIN 法または GMRES 法で反復的に解かれます.

DM_VBCSCC

非対称または不定値のスパース行列の連立 1 次方程式 (BICGSTAB(*l*)法, 圧縮列格納法)
CALL DM_VBCSCC(A, NZ, NROW, NFCNZ, N, B, ITMAX, EPS, IGUSS, L, X, ITER, W, IW, ICON)

(1) 機能

$n \times n$ の非対称／不定値なスパース行列を係数行列とする連立 1 次方程式を l 次安定化双対共役勾配法(BICGSTAB(l):Bi-Conjugate Gradient Stabilized (l) method)で解きます.

$$\boldsymbol{Ax} = \boldsymbol{b}$$

$n \times n$ の係数行列 $\boldsymbol{A}$ は, 圧縮列格納法で格納します.

ベクトル $\boldsymbol{b}$ および $\boldsymbol{x}$ は n 次元ベクトルです.

反復解法の収束および利用指針に関しては, “SSL II 拡張機能使用手引書 II 第 I 部 概説 第 4 章 連立 1 次方程式の反復解法と収束性” を参照してください.

(2) パラメタ

A.....入力. 係数行列の非零要素を格納します.
A(1:NZ)に係数行列を格納します.
A(NZ)なる 1 次元配列.
圧縮列格納法については, 実スパース行列と実ベクトル(圧縮列格納法), DM_VMVSCC の図 DM_VMVSCC-1 を参照してください.

NZ.....入力. 係数行列 $\boldsymbol{A}$ の非零要素の総数.

NROW .....入力. 圧縮列格納法で使用される行指標で, A に格納される要素が何番目の行ベクトルに属するかを示します.
NROW(NZ)なる 1 次元配列.

NFCNZ .....入力. 行列 $\boldsymbol{A}$ の各列の非零要素を列方向に圧縮して順次配列 A に格納するとき, 対応する列の最初の非零要素が格納される位置を表します.
NFCNZ(N+1) = NZ+1.
NFCNZ(N+1)なる 1 次元配列.

N.....入力. 行列 $\boldsymbol{A}$ の次数 n.

B.....入力. 連立 1 次方程式の右辺の定数ベクトルを B(1:N)に格納します.
B(N)なる 1 次元配列.

ITMAX .....入力. BICGSTAB(l)法の反復回数の上限値. 2000 程度で十分です.

EPS.....入力. 収束判定に用いられる判定値.
EPS が 0.0 以下のとき, 10^{-6} が設定されます.
((3)使用上の注意 a. 注意①参照).

IGUSS.....入力. 配列 X に指定された解ベクトルの近似値から反復計算を行うかを指定する制御情報.
0 のとき, 解ベクトルの近似値を指定しません.

0 以外のとき, 配列 X に指定された解ベクトルの近似値から反復計算を開始します.

L.....入力. BICGSTAB(*l*)法で安定化を行うときの安定化のステップ数 *l*.
1 ≤ L ≤ 8. ほとんどの場合 1 か 2 で十分です.
((3)使用上の注意 a. 注意②参照).

X.....入力. X(1:N)に解ベクトルの近似値を指定することができます.
出力. 解ベクトルが格納されます.
X(N)なる 1 次元配列.

ITER.....出力. BICGSTAB(*l*)法での実際の反復回数.

W.....作業域. W(NZ)なる 1 次元配列.

IW .....作業域. IW(2, NZ)なる 2 次元配列.

ICON.....出力. コンディションコード.
表 DM_VBCSCC-1 参照.

表 DM_VBCSCC-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし	―
20000	Break down を起こした.	処理を打ち切る.
20001	反復回数の上限に達した.	処理を打ち切る. 配列 X には, そのときまでに得られている近似値を出力するが, 精度は保証できない.
30000	N<1, NZ<0, NFCNZ(N+1) ≠ NZ+1, ITMAX ≤0, L<1, L >8	処理を打ち切る.

(3)　使用上の注意

a.　注意

①　本ルーチンは, 残差のユークリッドノルムが最初の残差のユークリッドノルムと EPS の積以下になったとき解が収束したと見なします. 正確な解と求められた近似解の誤差はほぼ行列 ***A*** の条件数と EPS の積に等しくなります.

②　L=1 のとき, BICGSTAB 法と同じアルゴリズムになります. L の値を大きくすると, 反復 1 回あたりのコストは増加しますが, 反復回数が減り, よい収束が得られる場合があります.

b.　使用例

連立 1 次方程式 $\boldsymbol{Ax}=\boldsymbol{f}$ を解きます. 行列 ***A*** は境界条件として立方体の境界で 0 となる楕円型の偏微分方程式に有限差分法を適用して生成されます.

$$-\Delta u + a\nabla u + u = f$$

ここで $a = (a_1, a_2, a_3)$, a_1, a_2 および a_3 はある定数です. 行列 ***A*** はサブルーチン init_mat_diag によって生成されます. これを圧縮列格納法に変換します.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できま

す. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT  REAL*8 (A-H,O-Z)
      PARAMETER (NORD=60,NX = NORD,NY =NORD ,NZ = NORD,
     $      N = NX*NY*NZ)
      PARAMETER (K = N+1)
      PARAMETER (NDIAG = 7)
      PARAMETER (L = 4)

      DIMENSION NOFST(NDIAG)
      DIMENSION DIAG(K,NDIAG),DIAG2(K,NDIAG)
      DIMENSION A(K*NDIAG),NROW(K*NDIAG),NFCNZ(N+1),
     $          W(K*NDIAG),IW(2,K*NDIAG)
      DIMENSION X(N),B(N),SOLEX(N),Y(N)

      PRINT *,'    BICGSTAB(L) METHOD'
      PRINT *,'    COMPRESSED COLUMN STORAGE'
      PRINT *

      SOLEX(1:N)=1.0D0
      PRINT *,'    EXPECTED SOLUTIONS'
      PRINT *,'    X(1) = ',SOLEX(1),' X(N) = ',SOLEX(N)
      PRINT *

      VA1 = 3D0
      VA2 = 1D0/3D0
      VA3 = 5D0
      VC = 1.0
      XL = 1.0
      YL = 1.0
      ZL = 1.0
      CALL INIT_MAT_DIAG(VA1,VA2,VA3,VC,DIAG,NOFST
     &          ,NX,NY,NZ,XL,YL,ZL,NDIAG,N,K)

      DO I=1,NDIAG
C
      IF(NOFST(I).LT.0)THEN
      NBASE=-NOFST(I)
      LENGTH=N-NBASE
      DIAG2(1:LENGTH,I)=DIAG(NBASE+1:N,I)
```

```
      ELSE
      NBASE=NOFST(I)
      LENGTH=N-NBASE
      DIAG2(NBASE+1:N,I)=DIAG(1:LENGTH,I)
      ENDIF
C
      ENDDO
C
      NUMNZ=1
      DO J=1,N
      NTOPCFG=1
      DO I=NDIAG,1,-1
C
      IF(DIAG2(J,I).NE.0.0D0)THEN
C
      NCOL=J-NOFST(I)
      A(NUMNZ)=DIAG2(J,I)
      NROW(NUMNZ)=NCOL
C
      IF(NTOPCFG.EQ.1)THEN
      NFCNZ(J)=NUMNZ
      NTOPCFG=0
      ENDIF
C
      NUMNZ=NUMNZ+1
      ENDIF
C
      ENDDO
      ENDDO
      NFCNZ(N+1)=NUMNZ
      NNZ=NUMNZ-1

      CALL DM_VMVSCC(A,NNZ,NROW,NFCNZ,N,SOLEX,
     $             B,W,IW,ICON)
      ERR1 = ERRNRM(SOLEX,X,N)
C
      X(1:N)=0.0D0
      CALL DM_VMVSCC(A,NNZ,NROW,NFCNZ,N,X,
     $             Y,W,IW,ICON)
      ERR2 = ERRNRM(Y,B,N)

      IGUSS = 0
```

```
      ITMAX = 2000
      EPS=1.0D-8

      CALL DM_VBCSCC(A,NNZ,NROW,NFCNZ,N,B,ITMAX
     &          ,EPS,IGUSS,L,X,ITER,W,IW,ICON)

      ERR3 = ERRNRM(SOLEX,X,N)
      CALL DM_VMVSCC(A,NNZ,NROW,NFCNZ,N,X,
     $             Y,W,IW,ICON)
      ERR4 = ERRNRM(Y,B,N)

      PRINT *,'    COMPUTED VALUES'
      PRINT *,'    X(1) = ',X(1),' X(N) = ',X(N)
      PRINT *
      PRINT *,'    DM_VBCSCC ICON = ',ICON
      PRINT *
      PRINT *,'    N = ',N,' :: NX = ',NX,' NY = ',NY,' NZ = ',NZ
      PRINT *,'    ITER MAX = ',ITMAX
      PRINT *,'    ITER = ',ITER
      PRINT *
      PRINT *,'    EPS = ',EPS
      PRINT *
      PRINT *,'    INITIAL ERROR = ',ERR1
      PRINT *,'    INITIAL RESIDUAL ERROR = ',ERR2
      PRINT *,'    CRITERIA RESIDUAL ERROR = ',ERR2*EPS
      PRINT *
      PRINT *,'    ERROR = ',ERR3
      PRINT *,'    RESIDUAL ERROR = ',ERR4
      PRINT *
      PRINT *

      IF(ERR4.LE.ERR2*EPS*1.1.AND.ICON.EQ.0)THEN
         WRITE(*,*)'********** OK **********'
      ELSE
         WRITE(*,*)'********** NG **********'
      ENDIF

      STOP
      END

C =======================================
C     INITIALIZE COEFFICIENT MATRIX
```

```
C =======================================
      SUBROUTINE INIT_MAT_DIAG(VA1,VA2,VA3,VC,D_L,OFFSET
     &                        ,NX,NY,NZ,XL,YL,ZL,NDIAG,LEN,NDIVP)
      IMPLICIT  REAL*8(A-H,O-Z)
      DIMENSION D_L(NDIVP,NDIAG)
      INTEGER   OFFSET(NDIAG)
C
      IF (NDIAG .LT. 1) THEN
        WRITE (*,*) 'SUBROUTINE INIT_MAT_DIAG:'
        WRITE (*,*) ' NDIAG SHOULD BE GREATER THAN OR EQUAL TO 1'
        RETURN
      ENDIF

!$OMP PARALLEL DEFAULT(PRIVATE)
!$OMP+     SHARED(VA1,VA2,VA3,VC,D_L,OFFSET
!$OMP+      ,NX,NY,NZ,XL,YL,ZL,NDIAG,LEN,NDIVP)

C NDIAG CANNOT BE GREATER THAN 7
      NDIAG_LOC = NDIAG
      IF (NDIAG .GT. 7) NDIAG_LOC = 7

C INITIAL SETTING
      HX = XL/(NX+1)
      HY = YL/(NY+1)
      HZ = ZL/(NZ+1)

!$OMP DO
      DO I = 1,NDIVP
      DO J = 1,NDIAG
      D_L(I,J) = 0.0
      ENDDO
      ENDDO
!$OMP ENDDO

      NXY = NX*NY

C OFFSET SETTING
!$OMP SINGLE
      L = 1
      IF (NDIAG_LOC .GE. 7) THEN
        OFFSET(L) = -NXY
        L = L+1
```

```
      ENDIF
      IF (NDIAG_LOC .GE. 5) THEN
        OFFSET(L) = -NX
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 3) THEN
        OFFSET(L) = -1
        L = L+1
      ENDIF
      OFFSET(L) = 0
      L = L+1
      IF (NDIAG_LOC .GE. 2) THEN
        OFFSET(L) = 1
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 4) THEN
        OFFSET(L) = NX
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 6) THEN
        OFFSET(L) = NXY
      ENDIF
!$OMP END SINGLE

C MAIN LOOP
!$OMP DO
      DO 100 J = 1,LEN
        JS = J

C DECOMPOSE JS-1 = (K0-1)*NX*NY+(J0-1)*NX+I0-1
        K0 = (JS-1)/NXY+1
        IF (K0 .GT. NZ) THEN
        PRINT*,'ERROR; K0.GH.NZ '
        GOTO 100
        ENDIF
        J0 = (JS-1-NXY*(K0-1))/NX+1
        I0 = JS - NXY*(K0-1) - NX*(J0-1)
        L = 1

        IF (NDIAG_LOC .GE. 7) THEN
          IF (K0 .GT. 1) D_L(J,L) = -(1.0/HZ+0.5*VA3)/HZ
          L = L+1
```

```
        ENDIF
        IF (NDIAG_LOC .GE. 5) THEN
          IF (J0 .GT. 1) D_L(J,L) = -(1.0/HY+0.5*VA2)/HY
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 3) THEN
          IF (I0 .GT. 1) D_L(J,L) = -(1.0/HX+0.5*VA1)/HX
          L = L+1
        ENDIF
        D_L(J,L) = 2.0/HX**2+VC
        IF (NDIAG_LOC .GE. 5) THEN
          D_L(J,L) = D_L(J,L) + 2.0/HY**2
          IF (NDIAG_LOC .GE. 7) THEN
            D_L(J,L) = D_L(J,L) + 2.0/HZ**2
          ENDIF
        ENDIF
        L = L+1
        IF (NDIAG_LOC .GE. 2) THEN
          IF (I0 .LT. NX) D_L(J,L) = -(1.0/HX-0.5*VA1)/HX
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 4) THEN
          IF (J0 .LT. NY) D_L(J,L) = -(1.0/HY-0.5*VA2)/HY
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 6) THEN
          IF (K0 .LT. NZ) D_L(J,L) = -(1.0/HZ-0.5*VA3)/HZ
        ENDIF
 100  CONTINUE
!$OMP ENDDO

!$OMP END PARALLEL

      RETURN
      END

C =======================================
* ABSOLUTE ERROR
* | X1 - X2 |
C =======================================
      REAL*8 FUNCTION ERRNRM(X1,X2,LEN)
      IMPLICIT  REAL*8 (A-H,O-Z)
```

```
      DIMENSION X1(*),X2(*)
C
      S = 0D0
      DO 100 I = 1,LEN
        SS = X1(I) - X2(I)
        S = S + SS * SS
 100  CONTINUE
C
      ERRNRM = SQRT( S )
      RETURN
      END
```

(4)　手法概要

BICG アルゴリズムについては“付録 1 参考文献一覧表”の[72]に記述されています.

BICGSTAB(*l*)法は BICGSTAB 法の変形です. “付録 1 参考文献一覧表” の[77]および[32]を参照してください.

DM_VBCSD

非対称または不定値のスパース行列の連立 1 次方程式 (BICGSTAB(*l*)法, 対角形式格納法)
CALL DM_VBCSD(A, K, NDIAG, N, NOFST, B, ITMAX, EPS, IGUSS, L, X, ITER, ICON)

(1)　機能

$n \times n$ の非対称／不定値なスパース行列を係数行列とする連立 1 次方程式を l 次安定化双対共役勾配法(BICGSTAB(l):Bi-Conjugate Gradient Stabilized (l) method)で解きます.

$$\boldsymbol{Ax} = \boldsymbol{b}$$

$n \times n$ の係数行列 $\boldsymbol{A}$ は, 対角形式の格納法で格納します.

ベクトル $\boldsymbol{b}$ および $\boldsymbol{x}$ は n 次元ベクトルです.

反復解法の収束および利用指針に関しては, “SSL II 拡張機能使用手引書 II 第 I 部 概説 第 4 章 連立 1 次方程式の反復解法と収束性” を参照してください.

(2)　パラメタ

A.....入力. 係数行列の非零要素を格納します.
A(1:N, 1:NDIAG)に係数行列を格納します.
A(K, NDIAG)なる 2 次元配列.
対角形式の格納方法については, “SSL II 拡張機能使用手引書 II 第 I 部 概説 3.2.1.1 一般スパース行列の格納方法 b. 一般スパース行列の対角形式の格納方法” を参照してください.

K.....入力. 配列 A の 1 次元目の大きさ(≥ N).

NDIAG.....入力. 係数行列 $\boldsymbol{A}$ の非零要素を含む対角ベクトル列の総数. 配列 A の 2 次元目の大きさ.

N.....入力. 行列 $\boldsymbol{A}$ の次数 n.

NOFST.....入力. 配列 A に格納される対角ベクトルに対応した主対角ベクトルからの距離を格納します. 上対角ベクトル列は正, 下対角ベクトル列は負の値で表します.
NOFST(NDIAG)なる 1 次元配列.

B.....入力. 連立 1 次方程式の右辺の定数ベクトルを B(1:N)に格納します.
B(N)なる 1 次元配列.

ITMAX .....入力. BICGSTAB(l)法の反復回数の上限値. 2000 程度で十分です.

EPS.....入力. 収束判定に用いられる判定値.
EPS が 0.0 以下のとき, 10^{-6} が設定されます.
((3)使用上の注意 a. 注意①参照).

IGUSS.....入力. 配列 X に指定された解ベクトルの近似値から反復計算を行うかを指定する制御情報.
0 のとき, 解ベクトルの近似値を指定しません.

0 以外のとき, 配列 X に指定された解ベクトルの近似値から反復計算を開始します.

L.....入力. BICGSTAB(*l*)法で安定化を行うときの安定化のステップ数 *l*.
1 ≤ L ≤ 8. ほとんどの場合 1 か 2 で十分です.
((3)使用上の注意 a. 注意③参照).

X.....入力. X(1:N)に解ベクトルの近似値を指定することができます.
出力. 解ベクトルが格納されます.
X(N)なる 1 次元配列.

ITER.....出力. BICGSTAB(*l*)法での実際の反復回数.

ICON.....出力. コンディションコード.
表 DM_VBCSD-1 参照.

表 DM_VBCSD-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし	ー
20000	Break down を起こした.	処理を打ち切る.
20001	反復回数の上限に達した.	処理を打ち切る. 配列 X には, そのときまでに得られている近似値を出力するが, 精度は保証できない.
30000	N<1, N>K, NDIAG <1, ITMAX ≤0, L<1, L >8	処理を打ち切る.
32001	| NOFST | > N - 1	

(3)　使用上の注意

a.　注意

①　本ルーチンは, 残差のユークリッドノルムが最初の残差のユークリッドノルムと EPS の積以下になったとき解が収束したと見なします. 正確な解と求められた近似解の誤差はほぼ行列 ***A*** の条件数と EPS の積に等しくなります.

②　対角形式を使う上での注意
係数行列 ***A*** の外側の対角ベクトル(“SSL II 拡張機能使用手引書 II 3.2.1.1 b. 一般スパース行列の対角形式の格納方法” 参照)の要素は零を設定する必要があります.
対角ベクトル列を配列 A に格納する順序に制限は特にありません.
この方法の利点は, 行列ベクトル積が間接指標を使わずに計算できる点です.
しかし, 対角構造を持たない行列は効率よく格納できないという点が欠点です.

③　L=1 のとき, BICGSTAB 法と同じアルゴリズムになります. L の値を大きくすると, 反復 1 回あたりのコストは増加しますが, 反復回数が減り, よい収束が得られる場合があります.

b. 使用例

連立 1 次方程式 $\boldsymbol{Ax} = \boldsymbol{f}$ を解きます. 行列 $\boldsymbol{A}$ は境界条件として立方体の境界で 0 となる楕円型の偏微分方程式に有限差分法を適用して生成されます.

$$-\Delta u + a\nabla u + u = f$$

ここで $a = (a_1, a_2, a_3)$, a_1, a_2 および a_3 はある定数です. 行列 $\boldsymbol{A}$ はサブルーチン init_mat_diag によって生成されます.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT  REAL*8 (A-H,O-Z)
      PARAMETER (NORD=60,NX = NORD,NY =NORD ,NZ = NORD,
     $      N = NX*NY*NZ)
      PARAMETER (K = N+1)
      PARAMETER (NDIAG = 7)
      PARAMETER (L = 4)
      PARAMETER(NVW=3*K)

      DIMENSION NOFST(NDIAG)
      DIMENSION A(K,NDIAG)
      DIMENSION X(N),B(N),SOLEX(N),Y(N)
      DIMENSION VW(NVW)

      PRINT *,'    BICGSTAB(L) METHOD'
      PRINT *,'    DIAGONAL FORMAT'
      PRINT *

      SOLEX(1:N)=1.0D0
      PRINT *,'    EXPECTED SOLUTIONS'
      PRINT *,'    X(1) = ',SOLEX(1),' X(N) = ',SOLEX(N)
      PRINT *

      EPS = 1D-8
      VA1 = 3D0
      VA2 = 1D0/3D0
      VA3 = 5D0
      VC = 1.0
      XL = 1.0
      YL = 1.0
      ZL = 1.0
      CALL INIT_MAT_DIAG(VA1,VA2,VA3,VC,A,NOFST
```

```
&          ,NX,NY,NZ,XL,YL,ZL,NDIAG,N,K)
 NBANDL=0
 NBANDR=0
 DO I=1,NDIAG
 IF(NOFST(I).LT.0)THEN
 NBANDL=MAX(NBANDL,-NOFST(I))
 ELSE
 NBANDR=MAX(NBANDR,NOFST(I))
 ENDIF
 ENDDO

 VW(1+NBANDL:N+NBANDL) = SOLEX(1:N)
 CALL DM_VMVSD(A,K,NDIAG,N,NOFST,NBANDL,VW,B,ICON2)

 X(1:N)=0.0D0
 ERR1 = ERRNRM(SOLEX,X,N)
 VW(1+NBANDL:N+NBANDL) = X(1:N)
 CALL DM_VMVSD(A,K,NDIAG,N,NOFST,NBANDL,VW,Y,ICON2)
 ERR2 = ERRNRM(Y,B,N)

 IGUSS = 0
 ITMAX = 2000

 CALL DM_VBCSD(A,K,NDIAG,N,NOFST,B,ITMAX
&          ,EPS,IGUSS,L,X,ITER,ICON)

 ERR3 = ERRNRM(SOLEX,X,N)
 VW(1+NBANDL:N+NBANDL) = X(1:N)
 CALL DM_VMVSD(A,K,NDIAG,N,NOFST,NBANDL,VW,Y,ICON2)
 ERR4 = ERRNRM(Y,B,N)

 PRINT *,'    COMPUTED VALUES'
 PRINT *,'    X(1) = ',X(1),' X(N) = ',X(N)
 PRINT *
 PRINT *,'    DM_VBCSD ICON = ',ICON
 PRINT *
 PRINT *,'    N = ',N,' :: NX = ',NX,' NY = ',NY,' NZ = ',NZ
 PRINT *,'    NBANDL = ',NBANDL,', NBANDR = ',NBANDR
 PRINT *,'    ITER MAX = ',ITMAX
 PRINT *,'    ITER = ',ITER
 PRINT *
 PRINT *,'    EPS = ',EPS
```

```
      PRINT *
      PRINT *,'    INITIAL ERROR = ',ERR1
      PRINT *,'    INITIAL RESIDUAL ERROR = ',ERR2
      PRINT *,'    CRITERIA RESIDUAL ERROR = ',ERR2*EPS
      PRINT *
      PRINT *,'    ERROR = ',ERR3
      PRINT *,'    RESIDUAL ERROR = ',ERR4
      PRINT *
      PRINT *

      IF(ERR4.LE.ERR2*EPS*1.1.AND.ICON.EQ.0)THEN
         WRITE(*,*)'********** OK **********'
      ELSE
         WRITE(*,*)'********** NG **********'
      ENDIF

      STOP
      END

C =======================================
C     INITIALIZE COEFFICIENT MATRIX
C =======================================
      SUBROUTINE INIT_MAT_DIAG(VA1,VA2,VA3,VC,D_L,OFFSET
     &                        ,NX,NY,NZ,XL,YL,ZL,NDIAG,LEN,NDIVP)
      IMPLICIT  REAL*8(A-H,O-Z)
      DIMENSION D_L(NDIVP,NDIAG)
      INTEGER   OFFSET(NDIAG)
C
      IF (NDIAG .LT. 1) THEN
        WRITE (*,*) 'SUBROUTINE INIT_MAT_DIAG:'
        WRITE (*,*) ' NDIAG SHOULD BE GREATER THAN OR EQUAL TO 1'
        RETURN
      ENDIF

!$OMP PARALLEL DEFAULT(PRIVATE)
!$OMP+     SHARED(VA1,VA2,VA3,VC,D_L,OFFSET
!$OMP+      ,NX,NY,NZ,XL,YL,ZL,NDIAG,LEN,NDIVP)

C NDIAG CANNOT BE GREATER THAN 7
      NDIAG_LOC = NDIAG
      IF (NDIAG .GT. 7) NDIAG_LOC = 7
```

```
C INITIAL SETTING
      HX = XL/(NX+1)
      HY = YL/(NY+1)
      HZ = ZL/(NZ+1)

!$OMP DO
      DO I = 1,NDIVP
      DO J = 1,NDIAG
      D_L(I,J) = 0.0
      ENDDO
      ENDDO
!$OMP ENDDO

      NXY = NX*NY

C OFFSET SETTING
!$OMP SINGLE
      L = 1
      IF (NDIAG_LOC .GE. 7) THEN
        OFFSET(L) = -NXY
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 5) THEN
        OFFSET(L) = -NX
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 3) THEN
        OFFSET(L) = -1
        L = L+1
      ENDIF
      OFFSET(L) = 0
      L = L+1
      IF (NDIAG_LOC .GE. 2) THEN
        OFFSET(L) = 1
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 4) THEN
        OFFSET(L) = NX
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 6) THEN
        OFFSET(L) = NXY
```

```
      ENDIF
!$OMP END SINGLE

C MAIN LOOP
!$OMP DO
      DO 100 J = 1,LEN
        JS = J

C DECOMPOSE JS-1 = (K0-1)*NX*NY+(J0-1)*NX+I0-1
        K0 = (JS-1)/NXY+1
        IF (K0 .GT. NZ) THEN
        PRINT*,'ERROR; K0.GH.NZ '
        GOTO 100
        ENDIF
        J0 = (JS-1-NXY*(K0-1))/NX+1
        I0 = JS - NXY*(K0-1) - NX*(J0-1)
        L = 1

        IF (NDIAG_LOC .GE. 7) THEN
          IF (K0 .GT. 1) D_L(J,L) = -(1.0/HZ+0.5*VA3)/HZ
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 5) THEN
          IF (J0 .GT. 1) D_L(J,L) = -(1.0/HY+0.5*VA2)/HY
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 3) THEN
          IF (I0 .GT. 1) D_L(J,L) = -(1.0/HX+0.5*VA1)/HX
          L = L+1
        ENDIF
        D_L(J,L) = 2.0/HX**2+VC
        IF (NDIAG_LOC .GE. 5) THEN
          D_L(J,L) = D_L(J,L) + 2.0/HY**2
          IF (NDIAG_LOC .GE. 7) THEN
            D_L(J,L) = D_L(J,L) + 2.0/HZ**2
          ENDIF
        ENDIF
        L = L+1
        IF (NDIAG_LOC .GE. 2) THEN
          IF (I0 .LT. NX) D_L(J,L) = -(1.0/HX-0.5*VA1)/HX
          L = L+1
        ENDIF
```

```
        IF (NDIAG_LOC .GE. 4) THEN
          IF (J0 .LT. NY) D_L(J,L) = -(1.0/HY-0.5*VA2)/HY
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 6) THEN
          IF (K0 .LT. NZ) D_L(J,L) = -(1.0/HZ-0.5*VA3)/HZ
        ENDIF
 100  CONTINUE
!$OMP ENDDO

!$OMP END PARALLEL

      RETURN
      END

C =====================================
* ABSOLUTE ERROR
* | X1 - X2 |
C =====================================
      REAL*8 FUNCTION ERRNRM(X1,X2,LEN)
      IMPLICIT  REAL*8 (A-H,O-Z)
      DIMENSION X1(*),X2(*)
C
      S = 0D0
      DO 100 I = 1,LEN
        SS = X1(I) - X2(I)
        S = S + SS * SS
 100  CONTINUE
C
      ERRNRM = SQRT( S )
      RETURN
      END
```

(4)　手法概要

BICG アルゴリズムについては“付録 1 参考文献一覧表”の[72]に記述されています.

BICGSTAB(*l*)法は BICGSTAB 法の変形です. “付録 1 参考文献一覧表” の[77]および[32]を参照してください.

DM_VBCSE

非対称または不定値のスパース行列の連立 1 次方程式 (BICGSTAB(*l*)法, ELLPACK 形式格納法)
CALL DM_VBCSE(A, K, IWIDT, N, ICOL, B, ITMAX, EPS, IGUSS, L, X, ITER, ICON)

(1)　機能

$n \times n$ の非対称／不定値なスパース行列を係数行列とする連立 1 次方程式を l 次安定化双対共役勾配法(BICGSTAB(l):Bi-Conjugate Gradient Stabilized (l) method)で解きます.

$$\boldsymbol{Ax} = \boldsymbol{b}$$

$n \times n$ の係数行列 $\boldsymbol{A}$ は, ELLPACK 形式の格納法で格納します.

ベクトル $\boldsymbol{b}$ および $\boldsymbol{x}$ は n 次元ベクトルです.

反復解法の収束および利用指針に関しては, “SSL II 拡張機能使用手引書 II 第 I 部 概説 第 4 章 連立 1 次方程式の反復解法と収束性” を参照してください.

(2)　パラメタ

A.....入力. 係数行列の非零要素を A(1:N, 1:IWIDT)に格納します.
A(K, IWIDT)なる 2 次元配列.
ELLPACK 形式の格納方法については, “SSL II 拡張機能使用手引書 II 第 I 部 概説 3.2.1.1 一般スパース行列の格納方法” を参照してください.

K.....入力. A および ICOL の 1 次元目の大きさ($\geq n$).

IWIDT.....入力. 係数行列 $\boldsymbol{A}$ の非零要素の行ベクトル方向の最大個数.
A および ICOL の 2 次元目の大きさ.

N.....入力. 行列 $\boldsymbol{A}$ の次数 n.

ICOL .....入力. ELLPACK 形式で使用される列指標で, A の対応する要素がいずれの列ベクトルに属すかを示します.
ICOL(K, IWIDT)なる 2 次元配列.

B.....入力. 連立 1 次方程式の右辺の定数ベクトルを B(1:N)に格納します.
B(N)なる 1 次元配列.

ITMAX .....入力. BICGSTAB(l)法の反復回数の上限値. 2000 程度で十分です.

EPS.....入力. 収束判定に用いられる判定値.
EPS が 0.0 以下のとき, 10^{-6} が設定されます.
((3)使用上の注意 a. 注意①参照).

IGUSS.....入力. 配列 X に指定された解ベクトルの近似値から反復計算を開始するかを示す制御情報.
0 のとき, 解ベクトルの近似値を指定しません.
0 以外のとき, 配列 X に指定された解ベクトルの近似値から反復計算を開始します.

L.....入力. BICGSTAB(*l*)法で安定化を行うときの安定化のステップ数 *l*.
1 ≤ L ≤ 8. ほとんどの場合 1 か 2 で十分です.
((3)使用上の注意 a. 注意②参照)

X.....入力. 解ベクトルの近似値を X(1:N)に指定することができます.
出力. 解ベクトルが X(1:N)に格納されます.
X(N)なる 1 次元配列.

ITER.....出力. BICGSTAB(*l*)法での実際の反復回数.

ICON.....出力. コンディションコード.
表 DM_VBCSE-1 参照.

表 DM_VBCSE-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし	―
20000	Break down を起こした.	処理を打ち切る.
20001	反復回数の上限に達した.	処理を打ち切る. 配列 X には, そのときまでに得られている近似値を出力するが, 精度は保証できない.
30000	K<1, IWIDT<1, N<1, ITMAX≤0, N>K, L<1, L >8	処理を打ち切る.
30001	バンド巾がゼロ.	

(3)　使用上の注意

a.　注意

①　本ルーチンは, 残差のユークリッドノルムが最初の残差のユークリッドノルムと EPS の積以下になったとき収束したと見なします.
正確な解と求められた近似解の誤差はほぼ行列 ***A*** の条件数と EPS の積に等しくなります.

②　L =1 のとき, BICGSTAB 法と同じアルゴリズムになります. L の値を大きくすると, 反復 1 回あたりのコストは増加しますが, 反復回数が減り, よい収束が得られる場合があります.

b.　使用例

連立 1 次方程式 $\boldsymbol{Ax} = \boldsymbol{f}$ を解きます. 行列 ***A*** は境界条件として立方体の境界で 0 となる楕円型の偏微分方程式に有限差分法を適用して生成されます.

$$-\Delta u + a\nabla u + u = f$$

ここで $a = (a_1, a_2, a_3)$, a_1, a_2 および a_3 はある定数です. 行列 ***A*** はサブルーチン init_mat_ell によって生成されます.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT  REAL*8 (A-H,O-Z)
      PARAMETER (NORD=60,NX =NORD ,NY = NORD,NZ = NORD,
     &           N = NX*NY*NZ)
      PARAMETER (K = N+1)
      PARAMETER (IWIDT = 7)
      PARAMETER (L = 4)
      DIMENSION ICOL(K,IWIDT)
      DIMENSION A(K,IWIDT)
      DIMENSION X(N),B(N),SOLEX(N),Y(N)

      PRINT *,'    BICGSTAB(L) METHOD'
      PRINT *,'    ELLPACK FORMAT'
      PRINT *

      SOLEX(1:N)=1.0D0
      PRINT *,'    EXPECTED SOLUTIONS'
      PRINT *,'    X(1) = ',SOLEX(1),' X(N) = ',SOLEX(N)
      PRINT *

      EPS = 1D-8
      VA1 = 3D0
      VA2 = 1D0/3D0
      VA3 = 5D0
      VC = 1.0
      XL = 1.0
      YL = 1.0
      ZL = 1.0
      CALL INIT_MAT_ELL(VA1,VA2,VA3,VC,A,ICOL
     &          ,NX,NY,NZ,XL,YL,ZL,IWIDT,N,K)

      CALL DM_VMVSE(A,K,IWIDT,N,ICOL,SOLEX,B,ICON2)

      X(1:N)=0.0D0
      ERR1 = ERRNRM(SOLEX,X,N)
      CALL DM_VMVSE(A,K,IWIDT,N,ICOL,X,Y,ICON2)
      ERR2 = ERRNRM(Y,B,N)

      IGUSS = 0
      ITMAX = 2000
```

```
      CALL DM_VBCSE(A,K,IWIDT,N,ICOL,B,ITMAX
     &          ,EPS,IGUSS,L,X,ITER,ICON)

      ERR3 = ERRNRM(SOLEX,X,N)
      CALL DM_VMVSE(A,K,IWIDT,N,ICOL,X,Y,ICON2)
      ERR4 = ERRNRM(Y,B,N)

      PRINT *,'    COMPUTED VALUES'
      PRINT *,'    X(1) = ',X(1),' X(N) = ',X(N)
      PRINT *
      PRINT *,'    DM_VBCSE ICON = ',ICON
      PRINT *
      PRINT *,'    N = ',N,' :: NX = ',NX,' NY = ',NY,' NZ = ',NZ
      PRINT *,'    ITER MAX = ',ITMAX
      PRINT *,'    ITER = ',ITER
      PRINT *
      PRINT *,'    EPS = ',EPS
      PRINT *
      PRINT *,'    INITIAL ERROR = ',ERR1
      PRINT *,'    INITIAL RESIDUAL ERROR = ',ERR2
      PRINT *,'    CRITERIA RESIDUAL ERROR =',ERR2*EPS
      PRINT *
      PRINT *,'    ERROR = ',ERR3
      PRINT *,'    RESIDUAL ERROR = ',ERR4
      PRINT *
      PRINT *

      IF(ERR4.LE.ERR2*EPS*1.1.AND.ICON.EQ.0)THEN
         WRITE(*,*)'********** OK **********'
      ELSE
         WRITE(*,*)'********** NG **********'
      ENDIF

      STOP
      END

C ======================================
C INITILIZE COEFFICIENT MATRIX
C ======================================
      SUBROUTINE INIT_MAT_ELL(VA1,VA2,VA3,VC,A_L,ICOL_L,NX,NY,NZ
     &          ,XL,YL,ZL,IWIDTH,LEN,NDIVP)
      IMPLICIT  REAL*8(A-H,O-Z)
```

```
      DIMENSION A_L(NDIVP,IWIDTH)
      DIMENSION ICOL_L(NDIVP,IWIDTH)
C
      IF (IWIDTH .LT. 1) THEN
         WRITE (*,*) 'SUBROUTINE INIT_MAT_ELL:'
         WRITE (*,*) ' IWIDTH SHOULD BE GREATER THAN OR EQUAL TO 1'
         RETURN
      ENDIF
!$OMP PARALLEL DEFAULT(PRIVATE)
!$OMP+  SHARED(VA1,VA2,VA3,VC,A_L,ICOL_L,NX,NY,NZ
!$OMP+        ,XL,YL,ZL,IWIDTH,LEN,NDIVP)

C IWIDTH CANNOT BE GREATER THAN 7
      IWIDTH_LOC = IWIDTH
      IF (IWIDTH .GT. 7) IWIDTH_LOC = 7

C INITIAL SETTING
      HX = XL/(NX+1)
      HY = YL/(NY+1)
      HZ = ZL/(NZ+1)

!$OMP DO
      DO J = 1,IWIDTH
      DO I = 1,NDIVP
      A_L(I,J) = 0.0
      ICOL_L(I,J) = I
      ENDDO
      ENDDO
!$OMP ENDDO

C MAIN LOOP
!$OMP DO
      DO 100 J = 1,LEN
        JS = J
        L = 1

C DECOMPOSE JS-1 = (K0-1)*NX*NY+(J0-1)*NX+I0-1
        K0 = (JS-1)/NX/NY+1
        IF (K0 .GT. NZ) THEN
        PRINT*,' ERROR; K0.GT.NZ '
        GOTO 100
        ENDIF
```

```
J0 = (JS-1-NX*NY*(K0-1))/NX+1
I0 = JS - NX*NY*(K0-1) - NX*(J0-1)
IF (IWIDTH_LOC .GE. 7) THEN
  IF (K0 .GT. 1) THEN
    A_L(J,L) = -(1.0/HZ+0.5*VA3)/HZ
    ICOL_L(J,L) = JS-NX*NY
    L = L+1
  ENDIF
ENDIF
IF (IWIDTH_LOC .GE. 5) THEN
  IF (J0 .GT. 1) THEN
    A_L(J,L) = -(1.0/HY+0.5*VA2)/HY
    ICOL_L(J,L) = JS-NX
    L = L+1
  ENDIF
ENDIF
IF (IWIDTH_LOC .GE. 3) THEN
  IF (I0 .GT. 1) THEN
    A_L(J,L) = -(1.0/HX+0.5*VA1)/HX
    ICOL_L(J,L) = JS-1
    L = L+1
  ENDIF
ENDIF
A_L(J,L) = 2.0/HX**2+VC
IF (IWIDTH_LOC .GE. 5) THEN
  A_L(J,L) = A_L(J,L) + 2.0/HY**2
  IF (IWIDTH_LOC .GE. 7) THEN
    A_L(J,L) = A_L(J,L) + 2.0/HZ**2
  ENDIF
ENDIF
ICOL_L(J,L) = JS
L = L+1
IF (IWIDTH_LOC .GE. 2) THEN
  IF (I0 .LT. NX) THEN
    A_L(J,L) = -(1.0/HX-0.5*VA1)/HX
    ICOL_L(J,L) = JS+1
    L = L+1
  ENDIF
ENDIF
IF (IWIDTH_LOC .GE. 4) THEN
  IF (J0 .LT. NY) THEN
    A_L(J,L) = -(1.0/HY-0.5*VA2)/HY
```

```
            ICOL_L(J,L) = JS+NX
            L = L+1
          ENDIF
        ENDIF
        IF (IWIDTH_LOC .GE. 6) THEN
          IF (K0 .LT. NZ) THEN
            A_L(J,L) = -(1.0/HZ-0.5*VA3)/HZ
            ICOL_L(J,L) = JS+NX*NY
          ENDIF
        ENDIF
 100  CONTINUE
!$OMP ENDDO

!$OMP END PARALLEL

      RETURN
      END

C ======================================
C ABSOLUTE ERROR
C | X1 - X2 |
C ======================================
      REAL*8 FUNCTION ERRNRM(X1,X2,LEN)
      IMPLICIT  REAL*8 (A-H,O-Z)
      DIMENSION X1(*),X2(*)
C
      S = 0D0
      DO 100 I = 1,LEN
        SS = X1(I) - X2(I)
        S = S + SS * SS
 100  CONTINUE
C
      ERRNRM = SQRT( S )
      RETURN
      END
```

(4)　手法概要

BICG アルゴリズムについては “付録 1 参考文献一覧表” の[72]に記述されています.

BICGSTAB(*l*)法は BICGSTAB 法の変形です. “付録 1 参考文献一覧表” の[77]および[32]を参照してください.

DM_VBLU

実バンド行列の LU 分解 (ガウスの消去法)
CALL DM_VBLU(A, K, N, NH1, NH2, EPSZ, IS, IP, ICON)

(1) 機能

$n \times n$, 下バンド幅 h_1, 上バンド幅 h_2 のバンド行列 $\boldsymbol{A}$ をガウスの消去法により LU 分解します.

$$\boldsymbol{PA} = \boldsymbol{LU}$$

ただし, $\boldsymbol{P}$ は行ベクトルの置換行列, $\boldsymbol{L}$ は単位下バンド行列, $\boldsymbol{U}$ は上バンド行列です.
$n > h_1 \geq 0$, $n > h_2 \geq 0$.

(2) パラメタ

A.....入力. バンド係数行列 $\boldsymbol{A}$ を格納します.
A(NH1+1:2×NH1+NH2+1, 1:N)に行列 $\boldsymbol{A}$ を格納します.
A(1:NH1,1:N)でバンドの外側の行列 $\boldsymbol{A}$ の要素は, ゼロを設定します.
図 DM_VBLU-1 参照.
出力. LU 分解された行列 $\boldsymbol{L}$ および $\boldsymbol{U}$ が格納されます.
図 DM_VBLU-2 参照.
A(K, N)なる倍精度実数型の 2 次元配列. A(2×NH1+NH2+2:K, 1:N)の値は保証されません.

K.....入力. 配列 A の整合寸法 (≥ 2×NH1+NH2+1).

N.....入力. 行列 $\boldsymbol{A}$ の次数 n.

NH1.....入力. 下バンド巾の大きさ h_1.

NH2.....入力. 上バンド巾の大きさ h_2.

EPSZ.....入力. ピボットの零判定値(≥0.0).
0.0 のときは, 標準値が設定されます. ((3) 使用上の注意 a. 注意①参照).

IS.....出力. 行ベクトルの入換えの回数を示します.
1 のとき　偶数回の入換え.
-1 のとき　奇数回の入換え.
((3) 使用上の注意 a. 注意④参照).

IP.....出力. 大きさ n の 1 次元配列. 行の入れ換えの履歴情報を格納するトランスポジションベクトルが格納されます.
((3) 使用上の注意 a. 注意②参照).

ICON.....出力. コンディションコード.
表 DM_VBLU-1 参照.

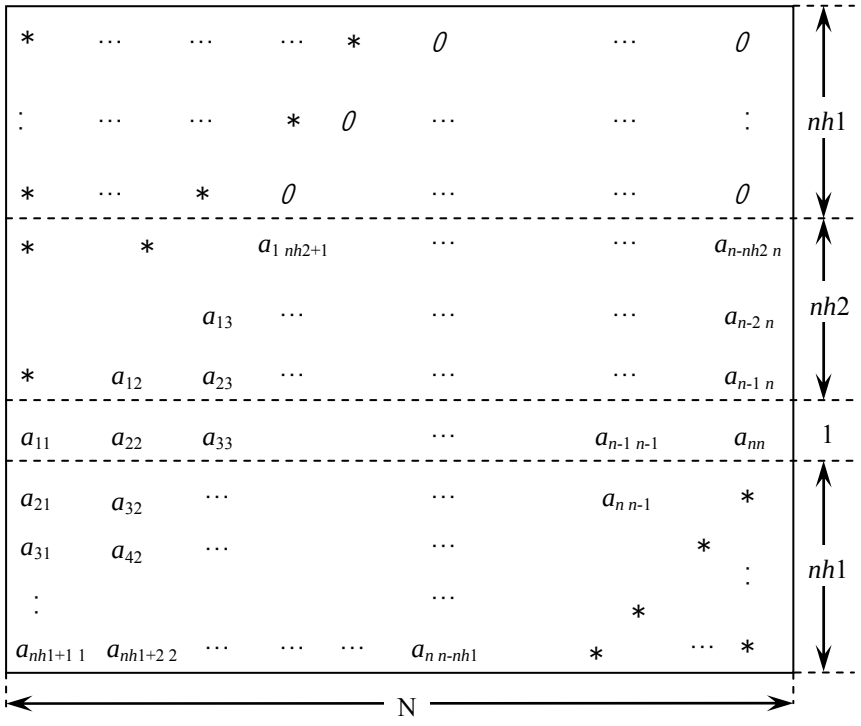


図 DM_VBLU-1　配列 A に行列 ***A*** を格納する方法

バンド行列 ***A*** の対角要素 a_{ii}, i=1, ... , n が A (h_1+h_2+1, 1:n)に, 格納されるように, 行列 ***A*** の列ベクトルを配列 A の列に連続に格納します.

上バンド行列部分, $a_{j-i,\,j}$, i =1, ... , h_2, j=1, ... , n, j - i≥1 を, A(h_1+1:h_1+h_2, 1:n)に格納します.

下バンド行列部分, $a_{j+i,\,j}$, i =1, ... , h_1, j=1, ... , n, j+i≤n を, A(h_1+h_2+2:2×h_1+h_2+1, 1:n)に格納します.

配列 A(1:h_1, 1:n)で, バンドの外側の行列 ***A*** の要素はゼロを設定してください.

*は不定値を示します.

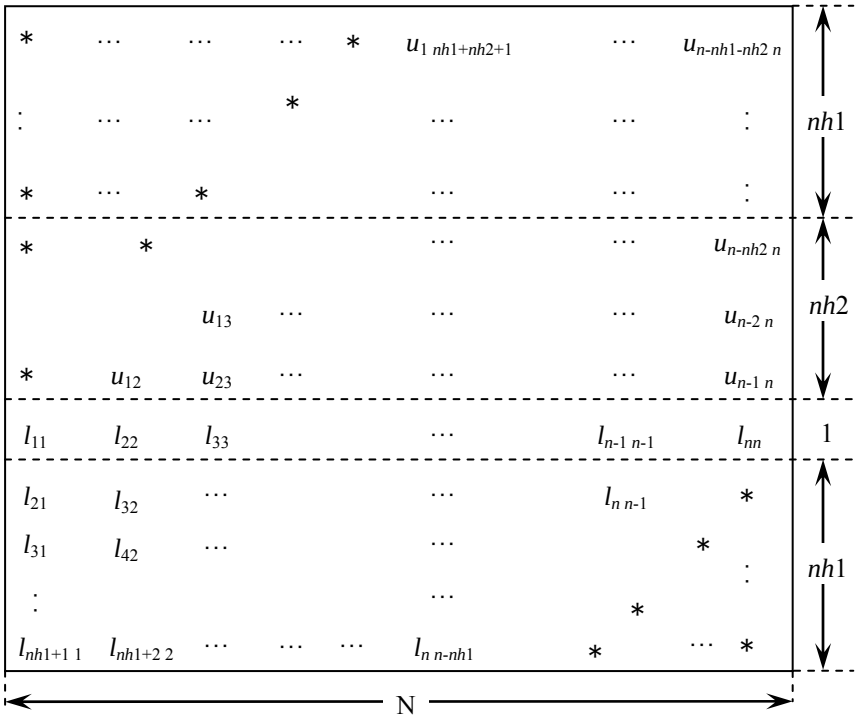


図 DM_VBLU-2　LU 分解された行列 ***L*** および ***U*** の配列 A に格納される方法

LU 分解された単位上バンド行列の対角要素を除いた部分, $u_{j-i+1,\,j}$, i =1, ... , h_1+h_2, j=1, ... , n, j-i+1≥1 が A(1:h_1+h_2, 1:n)に格納されます.
下バンド行列, $l_{j+i,\,j}$, i=0, ... , h_2, j=1, ... , n, j+i≤n が A(h_1+h_2+1:2×h_1+h_2+1, 1:n)に格納されます.
*は不定値を示します.

表 DM_VBLU-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし.	―
20000	行列 ***A*** のある行の要素がすべて零であったか, 又はピボットが相対的に零となった. 行列 ***A*** は非正則の可能性が強い.	処理を打ち切る.
30000	N<1, NH1≥N, NH1<0, NH2≥N, NH2<0, K<2×NH1+NH2+1, EPSZ<0	

(3)　使用上の注意

a.　注意

①　EPSZ が設定されると, LU 分解の過程でピボットが EPSZ 以下のときは相対的に零と見なし, ICON=20000 で処理を打ち切ります. EPSZ の標準値は丸め誤差の単位を u としたとき 16×u です. なおピボットが小さくても処理を続行させたいときには, EPSZ に極小の値を設定すればよいのですが, その結果は保証されません.

② 本サブルーチンでは, 部分ピボッティングに伴い, 行ベクトルの交換を行っています. すなわち, 分解の J 段階(J=1, ... , n)において第 I 行(I≥J)がピボット行として選択された場合には, 第 I 行と第 J 行の内容が交換されています. そしてその履歴を示すために IP(J)に I が格納されます.

③ 本サブルーチンに続けて, サブルーチン DM_VBLUX を呼び出すことにより, 連立 1 次方程式を解くことができます. 通常はサブルーチン DM_VLBX を呼び出せば, 一度に解が求まります.

④ 行列式の値は, IS および A(h_1+h_2+1, i), i=1, ... , n を掛け合わせることで求めることができます.

b. 使用例

n=10000, nh_1=2000, nh_2=3000 の実バンド行列を入力して, バンド行列の連立 1 次方程式を解きます.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
      implicit real*8(a-h,o-z)
      parameter(nh1=2000,nh2=3000,n=10000)
      parameter(ka=2*nh1+nh2+1,n2=n)
      real*8 a(ka,n2),b(n),dwork(4500)
      integer ip(n)
c
      ix=123
      nwork=4500
      nn=nh1+nh2+1
      do i=1,n
      call dvrau4(ix,a(nh1+1,i),nn,dwork,nwork,icon)
      do j=1,nh1+nh2+1
      enddo
      enddo
c
c     zero clear
c
      print*,'nh1=',nh1,',nh2=',nh2,',n=',n
c
c     a(1:nh1,n)=0.0d0
c
      do j=1,n
      do i=1,nh1
      a(i,j)=0.0d0
      enddo
```

```
      enddo
c
c     left upper triangular part
c
      do j=1,nh2
      do i=1,nh2+1-j
      a(i+nh1,j)=0.0d0
      enddo
      enddo
c
c    right rower triangular part
c
      nbase=2*nh1+nh2+1
      do j=1,nh1
      do i=1,j
      a(nbase-i+1,n-nh1+j)=0.0d0
      enddo
      enddo
c
c     set right hand constant vector
c
      do i=1,n
      b(i)=0.0d0
      enddo
c
      do i=1,n
      nptr=i-1
      do j=max(nptr+1-nh2,1),min(n,nptr+nh1+1)
      b(j)=b(j)+a(j-i+nh1+nh2+1,i)
      enddo
      enddo

c
      epsz=0.0d0
      call gettod(tt1)
      call dm_vblu(a,ka,n,nh1,nh2,epsz,is,ip,icon )
      call gettod(tt2)
      print*,'factor time (wall clock)=',(tt2-tt1)*1.0d-6
c
      call gettod(tt1)
      call dm_vblux(b,a,ka,n,nh1,nh2,ip,icon)
      call gettod(tt2)
```

```
      print*,'solve time (wall clock)=',(tt2-tt1)*1.0d-6
c
      tmp=0.0d0
      do i=1,n
      tmp=max(tmp,dabs(b(i)-1))
      enddo
c
      print*,'maximum error =',tmp
c
      stop
      end
```

(4)　手法概要

外積形式のガウスの消去法により LU 分解を行います.

DM_VBLUX

LU 分解された実バンド行列の連立 1 次方程式
CALL DM_VBLUX(B, FA, K, N, NH1, NH2, IP, ICON)

(1) 機能

LU 分解されたバンド行列を係数に持つ連立 1 次方程式を解きます.

$$\boldsymbol{LUx} = \boldsymbol{b}$$

ただし $\boldsymbol{L}$ は下バンド幅 h_1 の単位下バンド行列, $\boldsymbol{U}$ は上バンド幅 $h(=min(h_1+h_2,\ n-1))$の上バンド行列, $\boldsymbol{b}$ は n 次元の実定数ベクトルです. LU 分解された元の行列 $\boldsymbol{A}$ の次数, 上バンド幅, 下バンド幅は n, h_1 および h_2 です. $n>h_1\geq 0$, $n>h_2\geq 0$.

(2) パラメタ

B.....入力. 定数ベクトル $\boldsymbol{b}$.
出力. 解ベクトル $\boldsymbol{x}$.
B(N)なる倍精度実数型の 1 次元配列.

FA .....入力. LU 分解された行列 $\boldsymbol{L}$ および $\boldsymbol{U}$ を格納します.
図 DM_VBLUX-1 参照.
FA(K, N) なる倍精度実数型の 2 次元配列. FA(2×NH1+NH2+2:K, 1:N)の値は保証されません.

K.....入力. 配列 FA の整合寸法 (≥2×NH1+NH2+1).

N.....入力. 行列 $\boldsymbol{A}$ の次数 n.

NH1.....入力. バンド行列 $\boldsymbol{A}$ の下バンド幅 h_1.

NH2.....入力. バンド行列 $\boldsymbol{A}$ の上バンド幅 h_2.

IP.....入力. 大きさ n の 1 次元配列. 行ベクトルの入れ換えの履歴を示すトランスポジションベクトル.

ICON.....出力. コンディションコード.
表 DM_VBLUX-1 参照

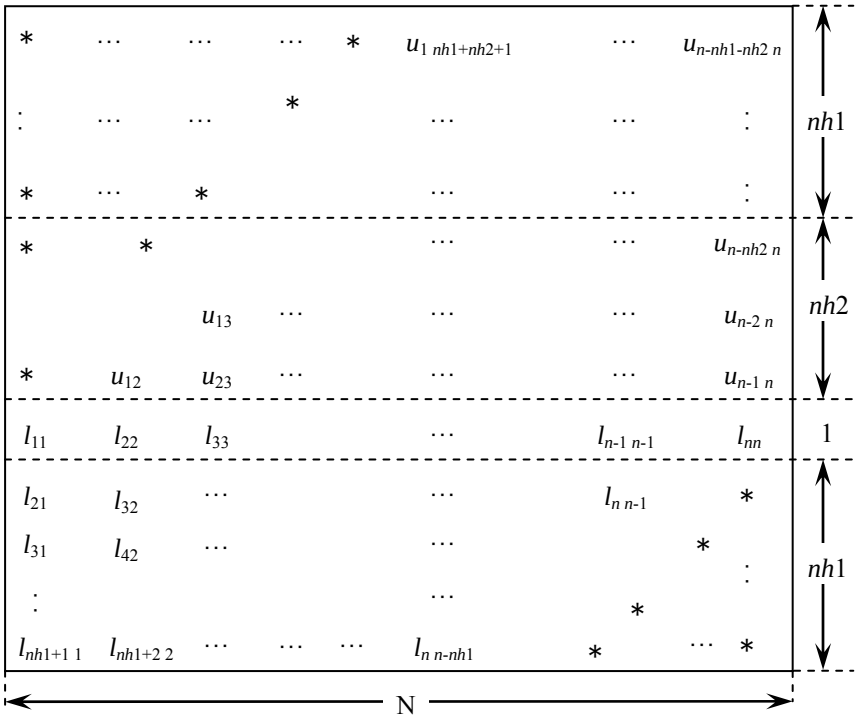


図 DM_VBLUX-1　LU 分解された行列 ***L*** および ***U*** の配列 FA に格納する方法

LU 分解された単位上バンド行列の対角要素を除いた部分, $u_{j-i+1,\,j}$, i =1, ... , h_1+h_2, j=1, ... , n, j-i+1≥1 が FA(1:h_1+h_2, 1:n)に格納されます.
下バンド行列, $l_{j+i,\,j}$, i=0, ... , h_2, j=1, ... , n, j+i≤n が FA(h_1+h_2+1:2×h_1+h_2+1, 1:n)に格納されます.
*は不定値を示します.

表 DM_VBLUX-1 コンディションコード

コード	意　　味	処理内容
0	エラーなし.	―
30000	N<1, NH1≥N, NH1<0, NH2≥N, NH2<0, K<2×NH1+NH2+1. 下バンド行列の対角要素が零であった. IP の内容が正しくない.	処理を打ち切る.

(3)　使用上の注意

a.　注意

①　バンド行列の連立 1 次方程式を解く場合, サブルーチン DM_VBLU を呼び出してから, 本サブルーチンを呼び出すことにより, 連立 1 次方程式を解くことができます. このとき, 本サブルーチンの入力パラメタ(定数ベクトルを除く)はサブルーチン DM_VBLU の出力パラメタをそのまま指定してください. 通常はサブルーチン DM_VLBX を呼び出せば, 一度に解が求まります.

b. 使用例

n=10000, nh_1=2000, nh_2=3000 の実バンド行列を入力して, バンド行列の連立 1 次方程式を解きます.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
      implicit real*8(a-h,o-z)
      parameter(nh1=2000,nh2=3000,n=10000)
      parameter(ka=2*nh1+nh2+1,n2=n)
      real*8 a(ka,n2),b(n),dwork(4500)
      integer ip(n)

c
      ix=123
      nwork=4500
      nn=nh1+nh2+1
      do i=1,n
      call dvrau4(ix,a(nh1+1,i),nn,dwork,nwork,icon)
      do j=1,nh1+nh2+1
      enddo
      enddo
c
c     zero clear
c
      print*,'nh1=',nh1,',nh2=',nh2,',n=',n
c
c     a(1:nh1,n)=0.0d0
c
      do j=1,n
      do i=1,nh1
      a(i,j)=0.0d0
      enddo
      enddo
c
c     left upper triangular part
c
      do j=1,nh2
      do i=1,nh2+1-j
      a(i+nh1,j)=0.0d0
      enddo
      enddo
```

```
c
c    right rower triangular part
c
      nbase=2*nh1+nh2+1
      do j=1,nh1
      do i=1,j
      a(nbase-i+1,n-nh1+j)=0.0d0
      enddo
      enddo
c
c     set right hand constant vector
c
      do i=1,n
      b(i)=0.0d0
      enddo
c
      do i=1,n
      nptr=i-1
      do j=max(nptr+1-nh2,1),min(n,nptr+nh1+1)
      b(j)=b(j)+a(j-i+nh1+nh2+1,i)
      enddo
      enddo

c
      epsz=0.0d0
      call gettod(tt1)
      call dm_vblu(a,ka,n,nh1,nh2,epsz,is,ip,icon )
      call gettod(tt2)
      print*,'factor time (wall clock)=',(tt2-tt1)*1.0d-6
c
      call gettod(tt1)
      call dm_vblux(b,a,ka,n,nh1,nh2,ip,icon)
      call gettod(tt2)
      print*,'solve time (wall clock)=',(tt2-tt1)*1.0d-6

c
      tmp=0.0d0
      do i=1,n
      tmp=max(tmp,dabs(b(i)-1))
      enddo
c
      print*,'maximum error =',tmp
```

```
      c
            stop
            end
```

(4)　手法概要

前進代入および後退代入により, LU 分解された行列を係数とする連立 1 次方程式を解きます.

DM_VCGD

正値対称スパース行列の連立 1 次方程式 (前処理付き CG 法, 対角形式格納法)
CALL DM_VCGD(A, K, NW, N, NDLT, B, IPC, ITMAX, ISW, OMEGA, EPS, IGUSS, X, ITER, RZ, W, IW, ICON)

(1)　機能

$n \times n$ の正規化された正値対称なスパース行列を係数行列とする連立 1 次方程式を前処理付き CG 法で解きます.

$$\boldsymbol{Ax} = \boldsymbol{b}$$

$n \times n$ の係数行列は, 対角要素が 1 になるように正規化されており, 対角要素を除いた非零要素を対角形式のスパース行列の格納法で格納します.

正値対称なスパース行列を係数行列とする連立 1 次方程式の正規化および対角形式の格納方式については, “SSL II 拡張使用手引書 II 第 I 部概説 3.2.1.2 正値対称スパース行列の格納方法”を参照してください. 対角形式の格納法は, 係数行列 $\boldsymbol{A}$ の非零要素が主対角ベクトルに平行な幾つかの対角線上のベクトルにのみ存在することを仮定しています.

偏微分方程式を, 特に定義される領域の境界と平行な格子で, 離散化して得られる連立 1 次方程式の場合, このような構造を持ちます. この場合, 各要素が係数行列の何列ベクトルにあるかを示す情報が不要で, 主対角ベクトルからの距離のみを要し, 効率的です.

(2)　パラメタ

A.....入力. A (1:N, 1:NW)に正規化されたスパース行列を格納します.
演算後, A (N+1:K, *)の値は保証されません.
A(K, NW)なる 2 次元配列.
正規化された正値対称スパース行列の係数行列の非零要素を対角形式で格納します.
正規化された正値対称スパース行列の対角形式の格納方法については, “SSL II 拡張使用手引書 II 第 I 部概説 3.2.1.2 正値対称スパース行列の格納方法 b. 正値対称スパース行列の対角形式の格納方法”参照.

K.....入力. 配列 A の 1 次元目の大きさ($\geq n$).

NW.....入力. 対角形式の格納方法で係数行列 A を格納する, 対角方向のベクトルの本数. 偶数. 配列 A の 2 次元目の大きさ.

N.....入力. 行列 $\boldsymbol{A}$ の次数 n.

NDLT.....入力. NDLT(NW)なる 1 次元配列で, 主対角ベクトルからの距離を示します.
正規化された正値対称スパース行列の対角形式の格納方法については, “SSL II 拡張使用手引書 II 第 I 部概説 3.2.1.2 正値対称スパース行列の格納方法 b. 正値対称スパース行列の対角形式の格納方法”参照.

B.....入力. 連立 1 次方程式の右辺の定数ベクトルを B(1:N)に格納します.
B(N)なる 1 次元配列.

IPC .....入力. 前処理の制御情報.
1 のとき前処理なし.
2 のとき Neumann の前処理.
3 のとき Block 不完全コレスキー分解による前処理. このとき OMEGA を指定する必要があります.
((3) 使用上の注意 a. 注意③参照).

ITMAX .....入力. 反復回数の上限(≥ 0).

ISW .....入力. 制御情報.
1 のとき初回の呼出し.
2 のとき 2 回目以降の呼出し. ただし A, NDLT, W, IW の値は初回呼び出しで設定された値を使用するため, 変更してはなりません.
((3) 使用上の注意 a. 注意①参照).

OMEGA.....入力. 不完全コレスキー分解のための修正値. 0 ≤ OMEGA ≤ 1
IPC=3 のとき使用されます.
((3) 使用上の注意 a. 注意③参照).

EPS.....入力. 収束判定に用いられる判定値.
0 が設定されたときは, $\varepsilon|b|$ が EPS として設定されます. εには 10^{-6} が設定されます.
((3) 使用上の注意 a. 注意②参照).

IGUSS.....入力. 配列 X に指定した解ベクトルの近似値から反復計算を開始するかどうかの情報を設定します.
0 のとき　　解ベクトルの近似を指定しません.
0 以外のとき 配列 X に指定された解ベクトルの近似から反復計算を開始します.

X.....入力. 連立 1 次方程式の解ベクトルの近似値を X(1:N)に指定することができます.
出力. 連立 1 次方程式の解ベクトルが X(1:N)に格納されます.
X(N)なる 1 次元配列.

ITER.....出力. 実際行った反復の回数.

RZ .....出力. 収束判定を行った残差 *rz* の平方根の値.
((3) 使用上の注意 a. 注意②参照).

W.....作業域.
IPC = 3 のとき, W(N+MAXT, NW+8)なる 2 次元配列.
その他のとき, W(N+MAXT, 7)なる 2 次元配列.
MAXT は, 並列実行する最大スレッド数.

IW .....作業域.
IPC=3 のとき, IW(N+2 × MAXT, 4)なる 2 次元配列.
その他のとき, IW(MAXT, 2)なる 2 次元配列.
MAXT は, 並列実行する最大スレッド数.

ICON.....出力. コンディションコード.
表 DM_VCGD-1 参照.

表 DM_VCGD-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし.	ー
10000	A の対角ベクトルを U/L の順に対角ベクトルの距離の昇順に格納し直した.	処理は続行する.
20001	反復回数の上限に達した.	処理を打ち切る. 配列 X には, そのときまでに得られている近似値を出力するが精度は保証できない.
20003	ブレークダウンを起こした.	
30003	ITMAX ≤ 0	処理を打ち切る.
30005	K < N	
30006	不完全 LL^T 分解ができなかった.	
30007	ピボットが負になった.	
30089	NW が偶数でない.	
30091	NBAND = 0	
30092	NW ≤ 0	
30093	K ≤ 0, $n \leq 0$	
30096	OMEGA < 0, OMEGA > 1	
30097	IPC < 1, IPC > 3	
30102	上三角部分が正しく格納されていない.	
30103	下三角部分が正しく格納されていない.	
30104	上三角の本数が下三角の本数と等しくない.	
30105	ISW ≠ 1, 2 であった.	
30200	|NDLT(i)| > n - 1 または NDLT(i) = 0	

(3)　使用上の注意

a.　注意

①　同じ係数行列を持ち定数ベクトルの異なる複数組の連立 1 次方程式を IPC=3 で解く場合, 1 回目は ISW=1 で解き, 2 回目以降は ISW=2 で解きます. 2 回目以降では, 1 回目の呼び出しで得られた不完全コレスキー分解の結果を再利用して解きます.

②　収束判定

n 回目の反復で求められた解が収束したかどうかは次のように判定します.

$\boldsymbol{r}$ を残差ベクトル $\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_n$, $\boldsymbol{M}$ を前処理行列, $rz = \boldsymbol{r}^T\boldsymbol{M}^{-1}\boldsymbol{r}$ として,

$$RZ = \sqrt{(rz)} < \mathrm{EPS}$$

を満たすとき, 収束したと判断します.

③ 前処理

2 種類の前処理および前処理なしの機能が提供されています.

楕円型の偏微分方程式を離散化して解く場合は, 不完全コレスキー分解による前処理が有効です.

$\boldsymbol{A} = \boldsymbol{I} - \boldsymbol{N}$ としたとき, 連立 1 次方程式$(\boldsymbol{I} - \boldsymbol{N})\boldsymbol{x} = \boldsymbol{b}$ の前処理 $\boldsymbol{M}$ は以下のようになります.

IPC=1 前処理なし $\boldsymbol{M} = \boldsymbol{I}$

IPC=2 Neumann $\boldsymbol{M}^{-1} = (\boldsymbol{I} + \boldsymbol{N})$

IPC=3 Block 不完全コレスキー分解 $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{L}^{\mathrm{T}}$. $\boldsymbol{A}$ を並列実行するスレッド数でブロックに分割し, 各ブロックを不完全コレスキー分解したものを前処理 $\boldsymbol{M}$ とします.

IPC=2 のとき，前処理行列も正定値であることが必要です．例えば$(\boldsymbol{I}+\boldsymbol{N})$の対角優位性はそのための十分条件になります．また，行列 $\boldsymbol{N}$ の固有値の絶対値で最大の値が 1 より大きく前処理行列が $\boldsymbol{A}$ の逆行列の良い近似にならない場合などでは，前処理を適用しても収束が改善しない可能性があります.

IPC=3 のとき, OMEGA ($0 \leq \mathrm{OMEGA} \leq 1$)に値を設定する必要があります. OMEGA=0 のときは, 不完全コレスキー分解, OMEGA=1 のときは修正不完全コレスキー分解です.

偏微分方程式の離散化から得られる連立 1 次方程式に対しては, OMEGA=0.92 から 1.00 が最適であることが経験から分かっています.

IPC=3 のときは, 方程式は前処理の効率化のためにウェーブフロント順に並び換えられます.

b. 使用例

n=51200 で正値対称スパース行列の連立 1 次方程式を解きます.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER ND,N,KA,WMAX,NDIAG
      PARAMETER (ND=80,MAXT=4,N=ND**3,KA=N)
      PARAMETER (WMAX=8)
      REAL*8  A(KA,WMAX),B(KA),X(KA),OMEGA,EPS
      REAL*8  IW(MAXT,2),W(N+MAXT,7)
      INTEGER DELTA(WMAX),IPREC,ITER,ITMAX
C
      CALL LAP3D(A,DELTA,KA,N,ND,WMAX,NDIAG)
C
      CALL RHS(A,N,KA,NDIAG,W,DELTA,B)
C
```

```
      EPS=1D-6
      ITMAX=2000
      ISW=1
      IGUSS=0
      IPREC=2
C
      CALL DM_VCGD(A,KA,NDIAG,N,DELTA,B,IPREC,ITMAX,ISW,OMEGA,
     &     EPS,IGUSS,X,ITER,RZ,W,IW,ICON)
      PRINT*,'ICON=',ICON
      PRINT*,'X(1)=',X(1)
      PRINT*,'X(N)=',X(N)
      STOP
      END
C
      SUBROUTINE LAP3D(A,DELTA,KA,N,ND,NDMAX,NDIAG)
      INTEGER NDMAX,NDIAG,N,I,J,L
      INTEGER DELTA(NDMAX),ND,NX,NY
      REAL*8  A(KA,NDMAX)

      DO J=1,NDMAX
      DO I=1,KA
      A(I,J)=0D0
      ENDDO
      ENDDO

      DO J=1,NDMAX
      DELTA(J)=0
      ENDDO

C 3D PROBLEM
      NDIAG=6
      NX=ND
      NY=ND
      DO I=1,N
      L=I
      IF((L/NX)*NX.NE.L.AND.L.LE.N-1) THEN
        A(I,1)=-1.D0/6.D0
      ENDIF
      ENDDO
      DO I=1,N
      L=I
      IZ=(L-1)/(NX*NY)
```

```
      IY=(L-1-IZ*NX*NY)/NX
      IF(L.LE.N-NX.AND.IY.NE.NY-1) THEN
        A(I,2)=-1.D0/6.D0
      ENDIF
      ENDDO
      DO I=1,N
      L=I
      IF(L.LE.N-NX*NY) THEN
        A(I,3)=-1.D0/6.D0
      ENDIF
      ENDDO
      DO I=1,N
      L=I
      IF(((L-1)/NX)*NX.NE.L-1.AND.L.GE.2.AND.L.LE.N) THEN
        A(I,4)=-1.D0/6.D0
      ENDIF
      ENDDO
      DO I=1,N
      L=I
      IZ=(L-1)/(NX*NY)
      IY=(L-1-IZ*NX*NY)/NX
      IF(L.GE.NX+1.AND.L.LE.N.AND.IY.NE.0) THEN
        A(I,5)=-1.D0/6.D0
      ENDIF
      ENDDO
      DO I=1,N
      L=I
      IF(L.GE.NX*NY+1.AND.L.LE.N) THEN
        A(I,6)=-1.D0/6.D0
      ENDIF
      ENDDO
      DELTA(1)=1
      DELTA(2)=NX
      DELTA(3)=NX*NY
      DELTA(4)=-1
      DELTA(5)=-NX
      DELTA(6)=-NX*NY
      RETURN
      END
C
      SUBROUTINE RHS(A,N,KA,NDIAG,DP,DELTA,B)
      IMPLICIT NONE
```

```
      INTEGER N,KA,NDIAG,I,J,DSHIFT
      REAL*8 DP(*),A(KA,*),B(KA)
      INTEGER DELTA(*),ICON
C
      DSHIFT=0
      DO J=1,NDIAG
      DSHIFT=MAX(DSHIFT,ABS(DELTA(J)))
      ENDDO
      DO I=1,3*N
      DP(I)=0
      ENDDO
      DO I=1,N
      DP(I+DSHIFT)=1.D0
      ENDDO
      CALL DM_VMVSD(A,KA,NDIAG,N,DELTA,DSHIFT,DP,B,ICON)
      DO I=1,N
      B(I)=B(I)+DP(DSHIFT+I)
      ENDDO
      RETURN
      END
```

(4)　手法概要

標準的な共役勾配法のアルゴリズム(“付録 1 参考文献一覧表”の[30]参照)が使われています.

不完全コレスキー分解による前処理の方法は“付録 1 参考文献一覧表”の[58]を参照してください. また Wavefront ordering によるベクトル化は“付録 1 参考文献一覧表”の[45]を参照してください.

対角形式のスパース行列の格納方法については, “付録 1 参考文献一覧表”の[59]および[52]を参照してください.

(5)　謝辞

ITPACK および NSPCG の著者に, 修正不完全コレスキー分解及び Wavefront ordering のルーチンを利用することを許可していただいたことに感謝致します.

DM_VCGE

正値対称スパース行列の連立 1 次方程式 (前処理付き CG 法, ELLPACK 形式格納法)
CALL DM_VCGE(A, K, NW, N, ICOL, B, IPC, ITMAX, ISW, OMEGA, EPS, IGUSS, X, ITER, RZ, W, IW, ICON)

(1) 機能

$n \times n$ の正規化された正値対称なスパース行列を係数行列とする連立 1 次方程式を前処理付き CG 法で解きます.

$$\boldsymbol{Ax} = \boldsymbol{b}$$

$n \times n$ の係数行列は, 対角要素が 1 になるように正規化されたもので, 対角要素を除く非零要素を ELLPACK 形式の格納法で格納します.

正値対称なスパース行列を係数行列とする連立 1 次方程式の正規化については, “SSL II 拡張使用手引書 II 第 I 部　概説 3.2.1.2 正値対称スパース行列の格納法”を参照してください.

(2) パラメタ

A.....入力. A(1:N, 1:NW)の部分に正規化されたスパース行列を格納します.
A(K, NW)なる 2 次元配列.
正規化された正値対称スパース行列の ELLPACK 形式の格納法については, “SSL II 拡張使用手引書 II 第 I 部　概説 3.2.1.2 正値対称スパース行列の格納法 a. 正値対称スパース行列の ELLPACK 形式の格納方法”を参照してください.
((3) 使用上の注意 a. 注意①参照).

K.....入力. 配列 A, ICOL の 1 次元目の大きさ(≥ N).

NW.....入力. 上三角行列の行ベクトルの非零要素の最大値を NSU とし, 下三角行列の行ベクトルの非零要素の最大値を NSL とするとき, $2 \times max$(NSU, NSL). 詳しくは, “SSL II 拡張使用手引書 II 第 I 部 概説 3.2.1.2 正値対称スパース行列の格納法 a. 正値対称スパース行列の ELLPACK 形式の格納方法”を参照してください.

N.....入力. 行列 $\boldsymbol{A}$ の次数 n.

ICOL .....入力. 非零要素がどの列ベクトルに属するかの情報を ICOL(1:N, 1:NW)の部分に格納します.
ICOL(K, NW)なる 2 次元配列.
正規化された正値対称スパース行列の ELLPACK 形式の格納法については, “SSL II 拡張使用手引書 II 第 I 部　概説 3.2.1.2 正値対称スパース行列の格納法 a. 正値対称スパース行列の ELLPACK 形式の格納方法”を参照してください.

B.....入力. B(1:N)に連立 1 次方程式の右辺の定数ベクトルが格納されます.
B(N)なる 1 次元配列.

IPC .....入力. 前処理の制御情報.
1 のとき前処理なし.
2 のとき Neumann の前処理.
3 のとき Block 不完全コレスキー分解による前処理. このとき OMEGA を指定する必要があります.
((3) 使用上の注意 a. 注意④参照).

ITMAX .....入力. 反復回数の上限(> 0).

ISW .....入力. 制御情報.
1 のとき初回の呼び出し.
2 のとき 2 回目以降の呼出し. ただし A, ICOL, W, IW の値は 1 回目に設定された値を使用するため, 変更してはなりません.
((3) 使用上の注意 a. 注意②参照).

OMEGA.....入力. 不完全コレスキー分解のための修正値. $0 \leq \mathrm{OMEGA} \leq 1$

EPS.....入力. 収束判定に用いられる判定値.
EPS=0 のとき, $\varepsilon|b|$が EPS として設定されます. ε には 10^{-6} が設定されます.
((3) 使用上の注意 a. 注意③参照).

IGUSS.....入力. 配列 X に指定された解ベクトルの近似値から反復計算を開始するかを示す制御情報.
0 のとき　　解ベクトルの近似値は指定しません.
0 以外のとき　配列 X に指定された解ベクトルの近似値から反復計算を開始します.

X.....入力. X(1:N)に連立 1 次方程式の解の近似ベクトルを指定することができます.
出力. 連立 1 次方程式の解ベクトルが格納されます.
X(N)なる 1 次元配列.

ITER.....出力. 実際行った反復の回数.

RZ .....出力. 収束判定を行った残差 *rz* の平方根.
((3) 使用上の注意 a. 注意②参照).

W.....作業域.
IPC = 3 のとき, W(N+MAXT, NW+8)なる 2 次元配列.
その他のとき, W(N+MAXT, 7)なる 2 次元配列.
MAXT は, 並列実行する最大スレッド数.

IW .....作業域.
IPC=3 のとき, IW(N+2 × MAXT, NW+5)なる 2 次元配列.
その他のとき, IW(MAXT, 2)なる 2 次元配列.
MAXT は, 並列実行する最大スレッド数.

ICON.....出力. コンディションコード.
表 DM_VCGE-1 参照.

表 DM_VCGE-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし.	ー
10000	A, ICOL の要素を U/L と並べ換えた.	処理は続行する.
20001	反復回数の上限に達した.	処理を打ち切る. 配列 X には, そのときまでに得られている近似値を出力するが精度は保証できない.
20003	ブレークダウンを起こした.	
30003	ITMAX ≤ 0	処理を打ち切る.
30005	K < N	
30006	不完全 LL^T 分解が出来なかった.	
30007	ピボットが負になった.	
30092	NW≤0	
30093	K≤0, N≤0	
30096	OMEGA<0, OMEGA >1	
30097	IPC<1, IPC >3	
30098	ISW≠1, 2 であった.	
30100	NW≠2 × max(NSU, NSL)	
30104	上三角または下三角部分が正しく格納されていない.	
負の数	第(-icon)行に非零対角要素がある.	

(3)　使用上の注意

a.　注意

①　スパース行列は ELLPACK 形式(“付録 1 参考文献一覧表”の[45], [62]参照)と呼ばれる格納方法で格納されます.

上三角部分を A(*, 1:NW/2)に下三角部分を A(*, NW/2+1:NW)に格納します. ここで NW=2 × max(NSU, NSL).

IPC=3 以外(不完全コレスキー分解による前処理以外の前処理を指定するとき)では, “SSL II 拡張機能使用手引書 II 第 I 部 概説 3.2.1.2 正値対称スパース行列の格納法 a. 正値対称スパース行列の ELLPACK 形式の格納方法”で記述されているより, 条件が緩い格納法でも受け入れます. つまり, 正規化された正値対称スパース行列で対角要素を除いたものを, 一般スパース行列の ELLPACK 形式の格納方法で格納したものも入力として受け入れます. このとき NW の値は, 2 × max(NSU, NSL)である必要はありません.

②　同じ係数行列を持ち定数ベクトルの異なる複数組の連立 1 次方程式を IPC=3 で解く場合, 1 回目は ISW=1 で解き, 2 回目以降は ISW=2 で解きます. 2 回目以降では, 1 回目の呼び出しで得られた不完全コレスキー分解の結果を再利用します.

③　収束判定

n 回目の反復で求められた解が収束したかどうかは次のように判定します.

$\boldsymbol{r}$ を残差ベクトル $\boldsymbol{r}=\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}_n$, $\boldsymbol{M}$ を前処理行列, $rz=\boldsymbol{r}^{\mathrm{T}}\boldsymbol{M}^{-1}\boldsymbol{r}$ として,

$$RZ=\sqrt{(rz)}<\mathrm{EPS}$$

を満たすとき, 収束したと判断します.

④　前処理

2 種類の前処理および前処理なしの機能が提供されています.

$\boldsymbol{A}=\boldsymbol{I}-\boldsymbol{N}$ としたとき, 連立 1 次方程式$(\boldsymbol{I}-\boldsymbol{N})\boldsymbol{x}=\boldsymbol{b}$ の前処理 $\boldsymbol{M}$ は以下のようになります.

IPC=1 前処理なし $\boldsymbol{M}=\boldsymbol{I}$

IPC=2 Neumann $\boldsymbol{M}^{-1}=(\boldsymbol{I}+\boldsymbol{N})$

IPC=3 Block 不完全コレスキー分解 $\boldsymbol{M}=\boldsymbol{L}\boldsymbol{L}^{\mathrm{T}}$. $\boldsymbol{A}$ を並列実行するスレッド数でブロックに分割し, 各ブロックを不完全コレスキー分解したものを前処理 $\boldsymbol{M}$ とします.

IPC=2 のとき，前処理行列も正定値であることが必要です．例えば$(\boldsymbol{I}+\boldsymbol{N})$の対角優位性はそのための十分条件になります．また，行列 $\boldsymbol{N}$ の固有値の絶対値で最大の値が 1 より大きく前処理行列が $\boldsymbol{A}$ の逆行列の良い近似にならない場合などでは，前処理を適用しても収束が改善しない可能性があります.

IPC=3 のとき, OMEGA ($0 \le \mathrm{OMEGA} \le 1$)に値を設定する必要があります. OMEGA=0 のときは不完全コレスキー分解, OMEGA=1 のときは修正不完全コレスキー分解です.

偏微分方程式の離散化から得られる連立 1 次方程式に対しては, OMEGA=0.92 から 1.00 が最適であることが経験から分かっています.

IPC=3 のとき, 方程式は前処理の効率化のためにウェーブフロント順に並び換えます.

b.　使用例

n=51200 の正値対称スパース行列の連立 1 次方程式を解きます.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER NMAX,N,WMAX,W
      PARAMETER (MAXT=4,NORD=80,WMAX=6)
      PARAMETER (NMAX=NORD**3,N=NMAX)

      REAL*8  RS(NMAX),X(NMAX),EPS,OMEGA,AP(NMAX),RZ
      REAL*8  A(NMAX,WMAX),XW(NMAX+MAXT,7)
      INTEGER ICOL(NMAX,WMAX),XIW1(MAXT,2),IPREC,I,ITMAX,ITER
```

```
C
      CALL SET(A,ICOL,NMAX,N,NORD,WMAX)
      DO I=1,N
      AP(I)=1.0D0
      ENDDO
      W=6
      CALL DM_VMVSE(A,NMAX,W,N,ICOL,AP,RS,ICON)
      DO I=1,N
      RS(I)=RS(I)+1.0D0
      ENDDO
      ITMAX=2000
      EPS=1D-6
      ISW=1
      IPREC=2
      IGUSS=0
      CALL DM_VCGE(A,NMAX,W,N,ICOL,RS,IPREC,ITMAX,ISW,OMEGA,EPS,
     &             IGUSS,X,ITER,RZ,XW,XIW1,ICON)
      PRINT*,'ICON = ',ICON
C
      PRINT*,'X(1)=',X(1)
      PRINT*,'X(N)=',X(N)
C
      STOP
      END
C
      SUBROUTINE SET(A,ICOL,NMAX,N,NORD,WMAX)
      INTEGER WMAX,N,I,J
      INTEGER ICOL(NMAX,WMAX),NORD
      REAL*8  A(NMAX,WMAX)
      N=N
      DO J=1,WMAX
      DO I=1,N
       A(I,J)=0D0
       ICOL(I,J)=I
      ENDDO
      ENDDO
C 3D PROBLEM
      NX=NORD
      NY=NORD
C
      DO I=1,N
       IF((I/NX)*NX.NE.I.AND.I.LE.N-1) THEN
```

```
        A(I,1)=-1.0D0/6.0D0
        ICOL(I,1)=I+1
       ENDIF
      ENDDO
C
      DO I=1,N
       IZ=(I-1)/(NX*NY)
       IY=(I-1-IZ*NX*NY)/NX
       IF(I.LE.N-NX.AND.IY.NE.NY-1) THEN
        A(I,2)=-1.0D0/6.0D0
        ICOL(I,2)=I+NX
       ENDIF
      ENDDO
C
      DO I=1,N
       IF(I.LE.N-NX*NY) THEN
        A(I,3)=-1.0D0/6.0D0
        ICOL(I,3)=I+NX*NY
       ENDIF
      ENDDO
C
      DO I=1,N
       IF(((I-1)/NX)*NX.NE.I-1.AND.I.GE.2.AND.I.LE.N) THEN
        A(I,4)=-1.0D0/6.0D0
        ICOL(I,4)=I-1
       ENDIF
      ENDDO
C
      DO I=1,N
       IZ=(I-1)/(NX*NY)
       IY=(I-1-IZ*NX*NY)/NX
       IF(I.GE.NX+1.AND.I.LE.N.AND.IY.NE.0) THEN
        A(I,5)=-1.0D0/6.0D0
        ICOL(I,5)=I-NX
       ENDIF
      ENDDO
C
      DO I=1,N
       IF(I.GE.NX*NY+1.AND.I.LE.N) THEN
        A(I,6)=-1.0D0/6.0D0
        ICOL(I,6)=I-NX*NY
       ENDIF
```

```
      ENDDO
      RETURN
      END
```

(4) 手法概要

標準的な共役勾配法(“付録 1 参考文献一覧表”の[30]を参照)を用いています. 不完全コレスキー分解による前処理の詳細は“付録 1 参考文献一覧表”の[58]を参照してください. また Wavefront ordering によるベクトル化は“付録 1 参考文献一覧表”の[45]を参照してください.

(5) 謝辞

ITPACK および NSPCG の著者達に, 修正不完全コレスキー分解の前処理および Wavefront ordering のルーチンを利用することを許可していただいたことに感謝いたします.

DM_VCLU

複素行列の LU 分解 (ブロック化された LU 分解法)
CALL DM_VCLU(ZA, K, N, EPSZ, IP, IS, ICON)

(1)　機能

$n \times n$ の正則な複素数行列をブロック化した外積形ガウスの消去法により LU 分解します.

$$\boldsymbol{PA} = \boldsymbol{LU}$$

ただし, $\boldsymbol{P}$ は部分ピボッティングによる行の入換えを行う置換行列, $\boldsymbol{L}$ は下三角行列, $\boldsymbol{U}$ は単位上三角行列です. ($n \geq 1$)

(2)　パラメタ

ZA .....入力. 行列 $\boldsymbol{A}$ を ZA(1:N, 1:N)に格納します.
出力. 行列 $\boldsymbol{L}$ と行列 $\boldsymbol{U}$ が ZA(1:N, 1:N)に格納されます.
図 DM_VCLU-1 参照.
ZA(K, N)なる倍精度複素数型の 2 次元配列.

K.....入力. 格納配列 ZA の 1 次元目の大きさ(≥ N).

N.....入力. 行列 $\boldsymbol{A}$ の次数 n.

EPSZ .....入力. ピボットの相対零判定値(≥ 0.0)
0.0 のときは標準値が採用されます. ((3) 使用上の注意 a. 注意①参照)

IP.....出力. 部分ピボッティングによる行の入換えの履歴を示すトランスポジションベクトル. 大きさ n の 1 次元配列. ((3) 使用上の注意 a. 注意②参照).

IS.....出力. 行列 $\boldsymbol{A}$ の行列式を求めるための情報, 演算後の配列 ZA の n 個の対角要素と IS の値を掛け合わせると行列式が得られます.

ICON.....出力. コンディションコード.
表 DM_VCLU-1 参照.

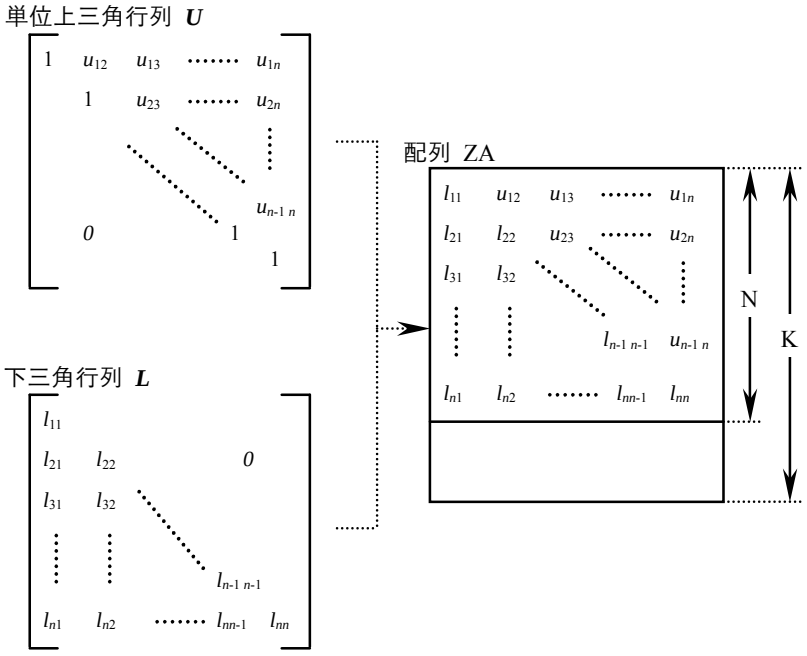


図 DM_VCLU-1　配列 ZA における ***L*** 及び ***U*** の格納方法

LU 分解された行列 ***L*** および ***U*** は, ***U*** の対各要素を除いた上三角部分と ***L*** が配列 ZA(1:N, 1:N)に格納されます.

表 DM_VCLU-1 コンディションコード

コード	意　　味	処理内容
0	エラーなし.	ー
20000	行列 ***A*** のある行の要素がすべて零であったか, またはピボットが相対的に零となった. 行列 ***A*** は非正則の可能性が強い.	処理を打ち切る.
30000	K<N, N<1 である. または, EPSZ<0.0 である.	

(3)　使用上の注意

a.　注意

①　EPSZ に値を設定したとすると, この値は次の意味をもっています. すなわち, 選択されたピボット要素の絶対値が EPSZ 以下なら, そのピボットを零と見なし, ICON=20000 として処理を打ち切ります. EPSZ の標準値は丸め誤差の単位 *u* としたとき, EPSZ=16 *u* です. なおピボットが小さくなっても計算を続行させる場合には, EPSZ へ極小の値を与えればよいのですが, その結果は保障されません.

②　トランスポジションベクトルとは, 部分ピボッティングを行う LU 分解

PA = ***LU***

における置換行列 ***P*** に相当します.

本サブルーチンでは, 部分ピボッティングに伴い, 配列 ZA の内容を実際に交換

しています. すなわち, 分解の J 段階(J＝1, ... , *n*)において第 I 行(I≥J)がピボット行として選択された場合には, 配列 ZA の第 I 行と第 J 行の内容が交換されています. そしてその履歴を示すために IP(J)に I が格納されます.

③　本サブルーチンに続けて, サブルーチン DM_VCLUX を呼び出すことにより, 連立 1 次方程式を解くことができます. 通常はサブルーチン DM_VLCX を呼び出せば, 一度に解が求まります.

b.　使用例

複素行列の連立 1 次方程式を LU 分解して解きます.
(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (N=2000,K=N+1)
C
      COMPLEX*16 A(K,N),B(N)
      REAL*8     C
      INTEGER    IP(N),IS
C
      C=SQRT(1.0D0/DBLE(1+N))
      T=DATAN(1.0D0)*8./(1+N)
C
      DO 100 J=1,N
      DO 100 I=1,N
      A(I,J)=DCMPLX(C*COS(T*I*J),C*SIN(T*I*J))
 100  CONTINUE
C
      DO 200 I=1,N
      S=(0.,0.)
      DO 200 J=1,N
      S=S+DCMPLX(COS(T*I*J),SIN(T*I*J))
      B(I)=S*C
 200  CONTINUE
C
      EPSZ=0.0D0
      CALL DM_VCLU(A,K,N,EPSZ,IP,IS,ICON)
C
      CALL DM_VCLUX(B,A,K,N,IP,ICON)
      PRINT*,'ICON=',ICON
```

```
ERROR=0.0D0
DO I=1,N
ERROR=MAX(ERROR,ABS(1.0D0-B(I)))
ENDDO
PRINT*,'ERROR =',ERROR

PRINT*,'ORDER=',N,' B(1)=',B(1),'B(N)=',B(N)
STOP
END
```

(4) 手法概要

ブロック化された外積形の LU 分解法については“付録 1 参考文献一覧表”の[1], [30], [54], [55],[56]及び[70]を参照して下さい.

DM_VCLUX

LU 分解された複素数行列の連立 1 次方程式
CALL DM_VCLUX(ZB, ZFA, KFA, N, IP, ICON)

(1)　機能

LU 分解された複素数係数行列を持つ連立 1 次方程式

$$\boldsymbol{LUx} = \boldsymbol{Pb}$$

を解きます. ただし, $\boldsymbol{L}$, $\boldsymbol{U}$はそれぞれ$n \times n$の下三角行列と単位上三角行列, $\boldsymbol{P}$は置換行列(係数行列を LU 分解するときの部分ピボッティングによる行の入換えを行います.), $\boldsymbol{b}$ は n 次元の複素定数ベクトル, $\boldsymbol{x}$ は n 次元の解ベクトルです. ($n \geq 1$)

(2)　パラメタ

ZB .....入力. 定数ベクトル $\boldsymbol{b}$.
　　出力. 解ベクトル $\boldsymbol{x}$.
　　大きさ n の倍精度複素数型の 1 次元配列.

ZFA.....入力. 行列 $\boldsymbol{L}$ と行列 $\boldsymbol{U}$ を ZFA(1:N, 1:N)に格納します.
　　図 DM_VCLUX-1 参照
　　ZFA(KFA, N)なる倍精度複素数型の 2 次元配列.

KFA .....入力. 格納配列 ZFA の 1 次元目の大きさ(≥ N).

N.....入力. 行列 $\boldsymbol{L}$, $\boldsymbol{U}$ の次数 n.

IP.....入力. 部分ピボッティングによる行の入換えの履歴を示すトランスポジションベクトル.
　　大きさ n の 1 次元配列.
　　(サブルーチン DM_VCLU の(3)使用上の注意 a. 注意②参照).

ICON.....出力. コンディションコード.
　　表 DM_VCLUX-1 参照.

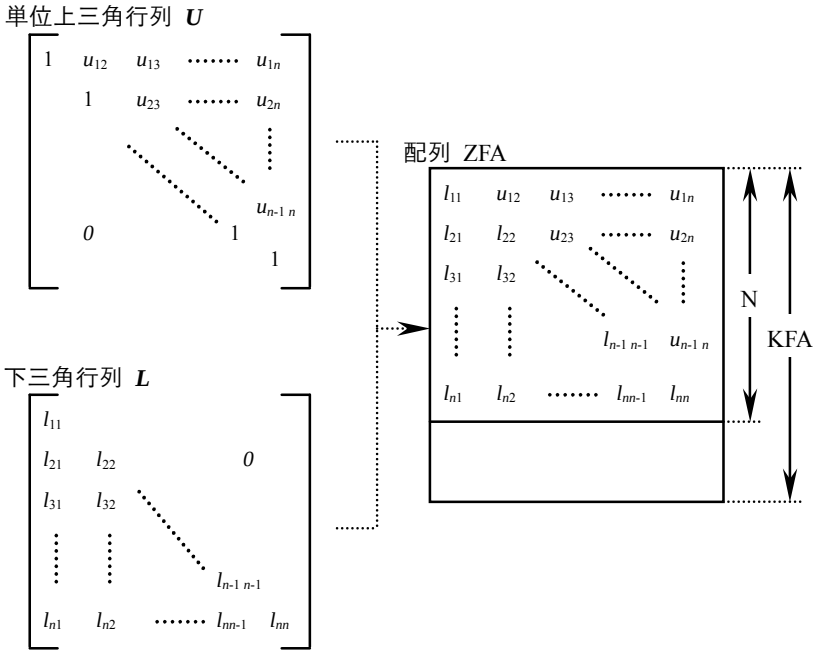


図 DM_VCLUX-1　配列 ZFA における ***L*** 及び ***U*** の格納方法

LU 分解された行列 ***L*** および ***U*** は, ***U*** の対各要素を除いた上三角部分および ***L*** を配列 ZFA(1:N, 1:N)に格納します.

表 DM_VCLUX-1 コンディションコード

コード	意　　味	処理内容
0	エラーなし.	―
20000	係数行列が非正則であった.	処理を打ち切る.
30000	KFA<N, N<1 である. または IP に誤りがあった.	

(3)　使用上の注意

a.　注意

①　連立 1 次方程式を解く場合, サブルーチン DM_VCLU を呼び出して, 係数行列を LU 分解してから, 本サブルーチンを呼び出せば方程式を解くことができます. しかし, 通常はサブルーチン DM_VLCX を呼び出せば, 一度に解が求められます.

b.　使用例

複素行列の連立 1 次方程式を LU 分解して解きます.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (N=2000,K=N+1)
C
      COMPLEX*16 A(K,N),B(N)
      REAL*8     C
      INTEGER    IP(N),IS
C
      C=SQRT(1.0D0/DBLE(1+N))
      T=DATAN(1.0D0)*8./(1+N)
C
      DO 100 J=1,N
      DO 100 I=1,N
      A(I,J)=DCMPLX(C*COS(T*I*J),C*SIN(T*I*J))
 100  CONTINUE
C
      DO 200 I=1,N
      S=(0.,0.)
      DO 200 J=1,N
      S=S+DCMPLX(COS(T*I*J),SIN(T*I*J))
      B(I)=S*C
 200  CONTINUE
C
      EPSZ=0.0D0
      CALL DM_VCLU(A,K,N,EPSZ,IP,IS,ICON)
C
      CALL DM_VCLUX(B,A,K,N,IP,ICON)
      PRINT*,'ICON=',ICON

      ERROR=0.0D0
      DO I=1,N
      ERROR=MAX(ERROR,ABS(1.0D0-B(I)))
      ENDDO
      PRINT*,'ERROR =',ERROR

      PRINT*,'ORDER=',N,' B(1)=',B(1),'B(N)=',B(N)
      STOP
      END
```

(4)　手法概要

LU 分解された複素行列を係数とする連立 1 次方程式を, 前進代入および後退代入により解きます(“付録 1 参考文献一覧表”の[54]を参照) .

DM_VCMINV

複素行列の逆行列 (ブロック化された Gauss-Jordan 法)
CALL DM_VCMINV(ZA, K, N, EPSZ, ICON)

(1) 機能

Gauss-Jordan 法により正則な $n \times n$ 複素行列 $\boldsymbol{A}$ の逆行列 $\boldsymbol{A}^{-1}$ を求めます.

(2) パラメタ

ZA .....入力. 行列 $\boldsymbol{A}$ を ZA(1:N, 1:N)に格納します.
出力. 行列 $\boldsymbol{A}^{-1}$ が ZA(1:N, 1:N)に格納されます.
ZA(K, N)なる倍精度複素数型 2 次元配列.

K.....入力. 格納配列 ZA の 1 次元目の大きさ(≥ N).

N.....入力. 行列 $\boldsymbol{A}$ の次数 n.

EPSZ .....入力. ピボットの相対零判定値(≥ 0.0).
0.0 が入力されたときは, 標準値が採用されます.
((3)使用上の注意 a. 注意①参照).

ICON.....出力. コンディションコード.
表 DM_VCMINV-1 参照.

表 DM_VCMINV-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし.	ー
20000	行列 $\boldsymbol{A}$ のある行の要素がすべて零であったか, またはピボットが相対的に零となった. 行列 $\boldsymbol{A}$ は非正則である可能性が強い.	処理を打ち切る.
30000	N<1, K<N または EPSZ<0.0	

(3) 使用上の注意

a. 注意

① 部分ピボッティングで選択されたピボット要素が 0.0 か絶対値が EPSZ 以下なら, 相対的に零とみなして, ICON=20000 として処理を打ち切ります. EPSZ の標準値は丸め誤差の単位を u としたとき, $16u$ です. EPSZ に極小の値を設定すると, ピボットの絶対値が小さくなっても処理は続行されますが, 計算結果の精度は保証されません.

b.　使用例

逆行列を求めます.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
cc    **example**
      implicit complex*16 (a-h,o-z)
      parameter(n=2000,k=n+1)
c
      complex*16    a(k,n)
      complex*16    as(k,n),tmpz
      real*8    c,t,tmp2,tmp,epsz
c
      c=sqrt(1.0d0/dble(n))
      t=datan(1.0d0)*8.d0/(n)
c
      do 100 j=1,n
      do 100 i=1,n
      a(i,j)=dcmplx(c*cos(t*(i-1)*(j-1)),
     $              c*sin(t*(i-1)*(j-1)))
      as(i,j)=dcmplx(c*cos(t*(i-1)*(j-1)),
     $              -c*sin(t*(i-1)*(j-1)))
 100  continue
c
      epsz=0.0d0
      call  dm_vcminv(a,k,n,epsz,icon)
cc
      tmp=0.0d0
      do j=1,n
      do i=1,n
      tmpz=(a(i,j)-as(i,j))
      tmp2=dabs(dble(tmpz))+dabs(dimag(tmpz))
      if(tmp2.gt.tmp)tmp=tmp2
      enddo
      enddo
      print*,'order=',n,' , error = ',tmp
  99  continue
      stop
      end
```

(4)　手法概要

ブロック化された Gauss-Jordan 法(“付録 1 参考文献一覧表”の[30]を参照)によって逆行列を求めます.

DM_VGEVPH

実対称行列の一般化固有値問題 (固有値・固有ベクトル) (三重対角化, マルチセクション法, 逆反復法)
CALL DM_VGEVPH(A, K, N, B, EPSZ, NF, NL, IVEC, ETOL, CTOL, NEV, E, MAXNE, M, EV, ICON)

(1)　機能

一般化固有値問題の指定された固有値を求めます. 指定に応じて, 固有ベクトルを求めます.

$$\boldsymbol{Ax} = \lambda \boldsymbol{Bx}$$

$\boldsymbol{A}$ は $n \times n$ の実対称行列, $\boldsymbol{B}$ は $n \times n$ の正値対称行列です.

(2)　パラメタ

A.....入力. A(1:N, 1:N)の下三角部分$\{A(i, j) \mid i \geq j\}$に, 実対称行列 $\boldsymbol{A}$ の下三角部分$\{a_{ij} \mid i \geq j\}$を格納します.
演算後の内容は保証されません.
A(K, N)なる倍精度実数型 2 次元配列.

K.....入力. 配列 A の 1 次元目の大きさ(≥ N).

N.....入力. 実対称行列 $\boldsymbol{A}$ の次数 n.

B.....入力. B(1:N, 1:N)の下三角部分$\{B(i, j) \mid i \geq j\}$に正値対称行列 $\boldsymbol{B}$ の下三角部分$\{b_{ij} \mid i \geq j\}$を格納します.
出力. $\boldsymbol{B}$ を LL^T 分解した結果が格納されます.
B(1:N, 1:N)の下三角部分$\{B(i, j) \mid i \geq j\}$に下三角行列 $\boldsymbol{L}\{l_{ij} \mid i \geq j\}$を格納します.
B(K, N)なる倍精度実数型の 2 次元配列.

EPSZ.....入力. $\boldsymbol{B}$ の LL^T 分解におけるピボットの零判定値 (≥ 0.0)
0.0 のときは標準値が採用されます.
((3)使用上の注意 a. 注意①参照).

NF .....入力. 固有値を小さい方から番号付けして(多重固有値には多重度分番号を割り当てる), 求める最初の固有値の番号.

NL .....入力. 固有値を小さい方から番号付けして(多重固有値には多重度分番号を割り当てる), 求める最後の固有値の番号.

IVEC .....入力. 制御情報.
1 のとき, 固有値および対応する固有ベクトルの両方を求めます.
1 以外のとき, 固有値のみ求めます.

ETOL .....入力. 固有値が数値的に異なるか多重かを判定する判定値.
3.0D-16 以下のときこの値が標準値として設定されます.

CTOL .....入力. 近接している固有値が近似的に多重かを式(3.1)を利用して判定する判定値. 近似的多重固有値(cluster)に属する固有値の対応する固有ベクトルの 1 次独立性を保証するために使用されます.

5.0D-12 が一般に適当です. ただし非常に大きなクラスタに対しては, 大きな値が必要です.
1.0D-6 ≥ CTOL ≥ ETOL.
CTOL > 1.0D-6 のときは, CTOL=1.0D-6 と設定されます.
CTOL < ETOL のときは, CTOL=10 × ETOL が標準値として設定されます.
((3)使用上の注意 a. 注意②参照).

NEV .....出力. 求められた固有値の個数に関する情報.
NEV(5)なる 1 次元配列.
詳細は以下のとおり.
NEV(1)は, 異なる固有値の個数.
NEV(2)は, 異なる近似的多重固有値(cluster)の個数.
NEV(3)は, 多重度も含んだすべての固有値の個数.
NEV(4)は, 求めた固有値の内最初の固有値の番号.
NEV(5)は, 求めた固有値の内最後の固有値の番号.

E.....出力. 固有値が格納されます.
求められた固有値は E(1:NEV(3))に格納されます.
E(MAXNE)なる 1 次元配列.

MAXNE.....入力. 計算できる固有値の最大個数.
多重度 m の固有値がいくつかあると考えられるとき, n を上限として NL – NF + 1 + 2 × m を設定する必要があります. 配列 E の 1 次元目の大きさ.
NEV(3) > MAXNE のとき, 固有ベクトルの計算はできません.
((3)使用上の注意 a. 注意③参照).

M.....出力. 求められた固有値の多重度に関する情報.
M(i, 1)は i 番目の固有値λ_iの多重度を, M(i, 2)は i 番目の固有値λ_iに関して, 近接している固有値を近似的多重固有値(cluster)と見なしたときの多重度を示します.
((3)使用上の注意 a. 注意②参照).
M(MAXNE, 2)なる 2 次元配列.

EV .....出力. IVEC=1 のとき, 固有値に対応して固有ベクトルが格納されます.
求められた固有ベクトルは, EV(1:N, 1:NEV(3))に格納されます.
EV(K, MAXNE)なる 2 次元配列.

ICON.....出力. コンディションコード.
表 DM_VGEVPH-1 参照

表 DM_VGEVPH-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし	ー
20000	B の LL^T 分解においてピボットが負となった. 行列 ***B*** は正値ではない.	処理を打ち切る.
20100	B の LL^T 分解においてピボットが相対的に零となった. 行列 ***B*** は非正則の可能性が強い.	
20200	密集固有値の計算中, 固有値の総数が MAXNE を超えた.	処理を打ち切る. 固有ベクトルを求めることはできないが, 異なる固有値自体は求められている. NEV(3)に MAXNE に設定するべき大きさが返却されます. ((3)使用上の注意 a. 注意③参照).
30000	NF < 1, NL > N, NL < NF, N <1, K < N, MAXNE < NL - NF+1, EPSZ < 0.	処理を打ち切る.

(3)　使用上の注意

a.　注意

①　EPSZ が設定されると, LL^T 分解の過程でピボットが EPSZ 以下のときに相対的に零と見なし, ICON = 20100 で処理を打ち切ります. EPSZ の標準値は, まるめ誤差単位を u としたとき $16u$ です. なおピボットが小さくても処理を続行させたいときには, EPSZ に極小の値を設定すればよいですが, その結果は保証されません.

②　本ルーチンは, 求める固有値を交わりのない順序付けられた集合に分けて独立に計算することにより並列化しています.

ε = ETOL としたとき, 連続する固有値 λ_j , $j = s - 1, s, \ldots, s+k, (k \geq 0)$ に対して,

$$\frac{|\lambda_i - \lambda_{i-1}|}{1 + \max\left(|\lambda_{i-1}|, |\lambda_i|\right)} \leq \varepsilon \tag{3.1}$$

$i = s, s+1, \ldots, s+k$ となる i に対して(3. 1)を満たし, $i = s - 1$ および $i = s+k+1$ について(3. 1)を満たさないとき, これらの固有値 λ_j , $j = s - 1, s, \ldots, s+k$ は数値的に多重であるとみなされます.

ETOL の標準値は 3. 0D-16～丸め誤差の単位であり, このとき固有値は計算機で求め得る精度まで分離されます.

(3.1)が ε = ETOL に対して成立しないとき, λ_{i-1} および λ_i は異なる固有値と考えます.

ε = ETOL で異なる固有値とみなされた連続する固有値 λ_m, $m = t - 1, t, \ldots, t+k$, (k≥0)に対して, ε = CTOL としたとき, $i = t, t+1, \ldots, t+k$ について(3.1)を満たし, $i = t - 1$ および $i = t+k+1$ について(3.1)を満たさないとき, これらの異なる固有値 λ_m, $m = t - 1, t, \ldots, t+k$ を近似的に多重(cluster)と見なします. これは, cluster に対応す

る不変部分空間, つまり 1 次独立な固有ベクトルを計算するために利用されます.

③ MAXNE に, 計算できる固有値の最大個数を指定できます. CTOL を大きくすると, cluster の大きさが大きくなり, 計算する固有値の総数が MAXNE を超えるかも知れません. この場合, CTOL を小さくするか, MAXNE を大きくする必要があります.

計算する固有値の総数が MAXNE を超えた場合, ICON=20200 を返却します. このとき, 固有ベクトルの計算を行うよう指定されていたら固有ベクトルの計算はできません. 固有値は求められていますが, 多重度分だけ繰り返して格納はされてはいません.

つまり, 固有値に関しては, 求められた異なる固有値が, E(1:NEV(1))に, および対応する固有値の多重度が M(1:NEV(1), 1)にそれぞれ格納されています.

固有値がすべて異なり, 近似的多重な固有値もない場合, MAXNE は求める固有値の総数を NT(NT=NL-NF+1)として NT で十分です. 多重な固有値がいくつかあり, 多重度が m と考えられるときは, 少なくとも MAXNE は $NT+2\times m$ とする必要があります.

計算する固有値の総数が MAXNE を超えた場合, 計算を続けるために必要な値が NEV(3)に返却されます. この値で領域を確保して再度呼び出すことで計算を続けることができます.

b. 使用例

固有値・固有ベクトルが分かっている一般化固有値問題の指定された固有値・固有ベクトルを求めます.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
cc    **example**
      implicit real*8(a-h,o-z)
      parameter(n=2000 ,k=n+1)
      parameter(nf=1,nl=n,max_nev=nl-nf+1,tau=1.0d0)
      dimension a(k,n),b(k,n),b2(k,n),c(k,n),d(k,n)
      dimension nev(5),mult(max_nev,2)
      dimension eval(max_nev),evec(k,max_nev)
cc
      pai=4.0d0*datan(1.0d0)
      coef=dsqrt(2.0d0/(n+1))
      do j=1,n
      do i=1,n
      d(i,j)=coef*dsin(pai/(n+1)*i*j)
      enddo
      enddo
```

```
cc
      do j=1,n
      do i=1,n
      if(i.eq.j)then
      c(J,J)=DBLE(J)
      else
      c(i,j)=0.0d0
      endif
      enddo
      enddo
cc
cc    d x c -> b
cc
      call dm_vmggm(d,k,c,k,b,k,n,n,n,icon)
cc
cc    b x d -> a
cc
      call dm_vmggm(b,k,d,k,a,k,n,n,n,icon)
cc
cc       B = LL^t , A <- LALt
cc
      do i=1,n
      do j=1,n
      b(j,i)=1.0d0/dsqrt(tau)
      b2(j,i)=min(i,j)/tau
      enddo
      enddo
      call dtrmm('Left','Lower','Not transpose','Not-unit',
     $           n,n,1.0d0,b,k,a,k)
      call dtrmm('Right','Upper','Not transpose','Not-unit',
     $           n,n,1.0d0,b,k,a,k)
cc
      n0x=nf
      n1x=nl
      ivec=1
      etol=1.0d-15
      ctol=1.0d-10
      max_nevx=max_nev
      epsz=0.0d0
      call dm_vgevph( a,k,n,b2,epsz,n0x,n1x,ivec,
     &                     etol,ctol,nev,
     &                     eval,max_nevx,mult,evec,icon )
```

```
      do i=1,nev(3),nev(3)/10
      print*,'eigen value in eval(',i,') = ',eval(i)
      enddo
      stop
      end
```

(4)　手法概要

一般化固有値問題(4.1)を考えます. ここで $\boldsymbol{B}$ は正値対称行列であり, コレスキー分解ができます.

$$\boldsymbol{Ax} = \lambda \boldsymbol{Bx} \tag{4.1}$$

$$\boldsymbol{LL}^{\mathrm{T}} = \boldsymbol{B} \tag{4.2}$$

(4.1)の左辺に L^{-1} を掛けます.

$$\boldsymbol{L}^{-1}\boldsymbol{Ax} = \lambda \boldsymbol{L}^{\mathrm{T}}\boldsymbol{x} \tag{4.3}$$

$$\boldsymbol{y} = \boldsymbol{L}^{\mathrm{T}}\boldsymbol{x} \tag{4.4}$$

とおいて,

$$\boldsymbol{x} = \boldsymbol{L}^{-\mathrm{T}}\boldsymbol{y} \tag{4.5}$$

(4.5)を(4.3)に代入して

$$\boldsymbol{L}^{-1}\boldsymbol{AL}^{-\mathrm{T}}\boldsymbol{y} = \lambda \boldsymbol{y} \tag{4.6}$$

$$\boldsymbol{C} = \boldsymbol{L}^{-1}\boldsymbol{AL}^{-\mathrm{T}} \tag{4.7}$$

$\boldsymbol{C}$ を(4.6)に代入して

$$\boldsymbol{Cy} = \lambda \boldsymbol{y} \tag{4.8}$$

(4.8)の固有値問題を DM_VSEVPH で解きます(DM_VSEVPH の(4)手法概要参照).

DM_VHEVP

エルミート行列の固有値・固有ベクトル
CALL DM_VHEVP(ZA, K, N, NF, NL, IVEC, ETOL, CTOL, NEV, EH, MAXNE, M, ZEV, ICON)

(1)　機能

n 次のエルミート行列の指定されたいくつかの固有値を求めます. 指定に応じて対応する固有ベクトルを計算します.

$$\boldsymbol{Ax} = \lambda \boldsymbol{x} \tag{1.1}$$

(2)　パラメタ

ZA .....入力. ZA(1:N, 1:N)の下三角部分$\{ZA(i, j) \mid i \geq j\}$に固有値・固有ベクトルを求めるエルミート行列 $\boldsymbol{A}$ の下三角部分$\{a_{ij} \mid i \geq j\}$を格納します.
演算後の内容は保証されません.
ZA(K, N)なる倍精度複素数型の 2 次元配列.

K.....入力. 配列 ZA の 1 次元目の大きさ(≥ N).

N.....入力. エルミート行列 $\boldsymbol{A}$ の次数 n.

NF .....入力. 固有値を小さい方から番号付けして(多重固有値には多重度分番号を割り当てる), 求める最初の固有値の番号.

NL .....入力. 固有値を小さい方から番号付けして(多重固有値には多重度分番号を割り当てる), 求める最後の固有値の番号.

IVEC .....入力. 制御情報.
1 のとき, 固有値および対応する固有ベクトルの両方を求めます.
1 以外のとき, 固有値のみ求めます.

ETOL .....入力. 固有値が数値的に異なるか多重かを式(3.1)を利用して判定する判定値.
3.0D-16 以下のときこの値が標準値として設定されます.
((3)使用上の注意 a. 注意①参照).

CTOL .....入力. 近接している固有値が近似的に多重かを式(3.1)を利用して判定する判定値. 近似的多重固有値(cluster)に属する固有値の対応する固有ベクトルの 1 次独立性を保証するために使用されます.
5.0D-12 が一般的に適当です. ただし非常に大きなクラスタに対しては, 大きな値が必要です.
1.0D-6 ≥ CTOL ≥ ETOL.
CTOL > 1.0D-6 のときは, CTOL = 1.0D-6 と設定されます.
CTOL < ETOL のときは, CTOL = 10 × ETOL が標準値として設定されます.
((3)使用上の注意 a. 注意①参照).

NEV .....出力. 求められた固有値の個数に関する情報.
NEV(5)なる 1 次元配列.

詳細は以下のとおり.

NEV(1)は, 異なる固有値の個数.

NEV(2)は, 異なる近似的多重固有値(cluster)の個数.

NEV(3)は, 多重度も含んだすべての固有値の個数.

NEV(4)は, 求めた固有値の内最初の固有値の番号.

NEV(5)は, 求めた固有値の内最後の固有値の番号.

EH.....出力. 固有値が格納されます.

求められた固有値は EH(1:NEV(3))に格納されます.

EH(MAXNE)なる倍精度実数型の 1 次元配列.

MAXNE.....入力. 計算できる固有値の最大個数. 配列 EH の大きさ.

多重度 m の固有値がいくつかあると考えられるとき, n を上限として NL - NF + 1 + 2 × m を設定する必要があります.

NEV(3) > MAXNE のとき, 固有ベクトルの計算はできません.

((3)使用上の注意 a. 注意②参照).

M.....出力.求められた固有値の多重度に関する情報.

M(i, 1)は i 番目の固有値λ_i の多重度を, M(i, 2)は i 番目の固有値λ_i に関して, 近接している固有値を近似的多重固有値(cluster)と見なしたときの多重度を示します.

((3)使用上の注意 a. 注意②参照).

M(MAXNE, 2)なる 2 次元配列.

ZEV.....出力. IVEC=1 のとき, 固有値に対応して固有ベクトルが格納されます.

求められた固有ベクトルは ZEV(1:N, 1:NEV(3))に格納されます.

ZEV(K, MAXNE)なる倍精度複素数型の 2 次元配列.

ICON.....出力. コンディションコード.

表 DM_VHEVP-1 参照

表 DM_VHEVP-1 コンディションコード

コード	意　　味	処理内容
0	エラーなし.	ー
20000	密集固有の計算中, 固有値の総数が MAXNE を超えた.	処理を打ち切る. 固有ベクトルを求めることはできないが, 異なる固有値自体は求められている. NEV(3)に MAXNE に設定するべき大きさが返却される. ((3)使用上の注意 a. 注意②参照).
30000	NF<1, NL>N, NL<NF, K<N, N<1, MAXNE < NL - NF+1	処理を打ち切る.

(3)　使用上の注意

a.　注意

①　本ルーチンは, 求める固有値を交わりのない順序付けられた集合に分けて独立に計算することにより並列化しています.

ε= ETOL としたとき, 連続する固有値λ_j, $j = s - 1, s, \ldots, s + k$, ($k \geq 0$)に対して,

$$\frac{|\lambda_i - \lambda_{i-1}|}{1 + \max\left(|\lambda_{i-1}|, |\lambda_i|\right)} \leq \varepsilon \tag{3.1}$$

$i=s, s+1, \ldots, s+k$ となる i に対して(3.1)を満たし, $i=s-1$ および $i=s+k+1$ について (3.1)を満たさないとき, これらの固有値λ_j, $j= s - 1, s, \ldots, s+k$, は数値的に多重であるとみなされます.

ETOL の標準値は 3.0D-16～丸め誤差の単位であり, このとき固有値は計算機で求め得る精度まで分離されます.

(3.1)がε=ETOL に対して成立しないとき, λ_{i-1}およびλ_iは異なる固有値と考えます.

ε= ETOL で異なる固有値とみなされた連続する固有値λ_m, $m=t - 1, t, \ldots, t + k$, ($k \geq 0$)に対して, ε=CTOL としたとき, $i=t, t+1, \ldots, t+k$ について(3.1)を満たし, $i=t-1$ および $i=t+k+1$ について(3.1)を満たさないとき, これらの異なる固有値λ_m, $m=t-1, t, \ldots, t+k$ を近似的に多重(cluster)と見なします. これは, cluster に対応する不変部分空間, つまり 1 次独立な固有ベクトルを計算するために利用されます.

②　MAXNE に, 計算できる固有値の最大個数を指定できます. CTOL を大きくすると, cluster の大きさが大きくなり, 計算する固有値の総数が MAXNE を超えるかも知れません. この場合, CTOL を小さくするか, MAXNE を大きくする必要があります.

計算する固有値の総数が MAXNE を超えた場合, ICON=20000 を返却します. このとき, 固有ベクトルの計算を行うよう指定されていたら固有ベクトルの計算はできません. 固有値は求められていますが, 多重度分だけ繰り返して格納はされてはいません.

つまり, 固有値に関しては, 求められた異なる固有値が, EH(1:NEV(1))に, および対応する固有値の多重度が M(1:NEV(1), 1)にそれぞれ格納されています.

固有値がすべて異なり, 近似的多重な固有値もない場合, MAXNE は求める固有値の総数を NT(NT=NL-NF+1)として NT で十分です. 多重な固有値がいくつかあり, 多重度が m と考えられるときは, 少なくとも MAXNE は $NT+2\times m$ とする必要があります.

計算する固有値の総数が MAXNE を超えた場合, 計算を続けるために必要な値が NEV(3)に返却されます. この値で領域を確保して再度呼び出すことで計算を続けることができます.

b. 使用例

固有値・固有ベクトルの分かっているエルミート行列の指定された固有値・固有ベクトルを求めます.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER (N=2000,K=N,NE=N,MAX_NEV=NE)
      COMPLEX*16 A(K,N),B(K,N),C(K,N),D(K,N),DH(K,N),ALPHA,BETA,
     &           EVECH(K,MAX_NEV)
      DIMENSION  NEV(5),MULT(MAX_NEV,2)
      DIMENSION  EVAL(MAX_NEV)
CC
      PAI2=8.0D0*DATAN(1.0D0)
      COEF=DSQRT(1.0D0/(N))
      DO J=1,N
      DO I=1,N
      PART1 =COEF*DCOS(PAI2/N*(I-1)*(J-1))
      PART2 =COEF*DSIN(PAI2/N*(I-1)*(J-1))
      D(I,J)=DCMPLX(PART1,PART2)
      DH(I,J)=DCMPLX(PART1,-PART2)
      ENDDO
      ENDDO
CC
      DO J=1,N
      DO I=1,N
      IF(I.EQ.J)THEN
      C(I,J)=DCMPLX(DBLE(I),0.0D0)
      ELSE
      C(I,J)=(0.0D0,0.0D0)
      ENDIF
      ENDDO
      ENDDO
CC
CC    D X C -> B
CC

      ALPHA=(1.0D0,0.0D0)
      BETA=(0.0D0,0.0D0)
      CALL ZGEMM('NO TRANSPOSE','NO TRANSPOSE',N,N,N,ALPHA,
```

```
     $            D,K,C,K,BETA,B,K)
CC
CC    B X D^H -> A
CC
      CALL ZGEMM('NO TRANSPOSE','NO TRANSPOSE',N,N, N,ALPHA,
     &            B,K,DH,K,BETA,A,K)
CC
      IVEC=1
      NF=1
      NL=NE
      EVAL_TOL=1.0D-15
      CLUS_TOL=1.0D-10
      CALL  DM_VHEVP( A,K,N,NF,NL,IVEC,EVAL_TOL,CLUS_TOL,NEV,
     &                EVAL,MAX_NEV,MULT,EVECH,ICON )
      DO I=1,NE,100
      PRINT*,'EIGEN VALUE IN EVEC(',I,') = ',EVAL(I)
      ENDDO
      STOP
      END
```

(4)　手法概要

$n \times n$ のエルミート行列 A=AR+iAI は, AR=AR^T, AI=-AI^T を満たします.

ブロック化された Householder 法でエルミート行列をエルミート三重対角行列に変換して, 対角ユニタリー変換で実三重対角行列に変換します.

三重対角行列の固有値・固有ベクトルはマルチセクション法および逆反復法 (DM_VTDEVC の記述および“付録 1 参考文献一覧表”[61]参照)で計算されます.

この固有ベクトルを利用して, エルミート行列の固有ベクトルが計算されます.

DM_VHTRID

エルミート行列の実対称三重対角行列への変換 (ハウスホルダー法)
CALL DM_VHTRID(ZA, K, N, D, SL, ZS, ICON)

(1) 機能

エルミート行列を Householder 変換でエルミート三重対角行列 $\boldsymbol{H}$ に変換し, 更に対角ユニタリ変換で実三重対角行列 $\boldsymbol{T}$ に変換します.

$$\boldsymbol{H} = \boldsymbol{P}^{*}\boldsymbol{A}\boldsymbol{P}$$

$$\boldsymbol{T} = \boldsymbol{V}^{*}\boldsymbol{H}\boldsymbol{V}$$

$\boldsymbol{A}$ は $n \times n$ エルミート行列, $\boldsymbol{P}$ は $n \times n$ ユニタリ行列, $\boldsymbol{V}$ は $n \times n$ の対角ユニタリ行列, $\boldsymbol{T}$ は実三重対角行列です.

(2) パラメタ

ZA .....入力. ZA(1:N, 1:N)の下三角部分$\{ZA(i, j) | i \geq j\}$にエルミート行列 $\boldsymbol{A}$ の下三角部分$\{a_{ij} | i \geq j\}$を格納します.
ZA(K, N)なる倍精度複素数型の 2 次元配列.
出力. ZA(1:N, 1:N-2)の下三角部分$\{ZA(i, j) | i \geq j\}$にエルミート三重対角化を行うために利用した Householder 変換に関する情報が格納されます. 演算後, 上三角部分の値は保証されません. ((3)使用上の注意 a. 注意①参照)

K.....入力. 配列 ZA の 1 次元目の大きさ(≥ N).

N.....入力. エルミート行列 $\boldsymbol{A}$ の次数 n.

D.....出力. D(N)なる倍精度実数型の 1 次元配列で, 三重対角化された三重対角行列の対角要素を格納します.

SL.....出力. SL(N)なる倍精度実数型の 1 次元配列で, 三重対角化された三重対角行列の下副対角要素を SL(2:N)に格納します. SL(1) = 0.

ZS.....出力. ZS(1:N)に, 対角ユニタリ行列の対角要素が格納されます.
ZS(N)なる倍精度複素数型の 1 次元配列.

ICON.....出力. コンディションコード.
表 DM_VHTRID-1 参照

表 DM_VHTRID-1 コンディションコード

コード	意　　味	処理内容
0	エラーなし.	ー
30000	K<N, N<2	処理を打ち切る.

(3) 使用上の注意

a.　注意

①　k=1, ... , n-2 まで, 以下の変換を繰り返してエルミート三重対角化します.

$$\boldsymbol{A}^{k} = \boldsymbol{P}_k^{*}\boldsymbol{A}^{k-1}\boldsymbol{P}_k \ , \quad \boldsymbol{A}^{0} = \boldsymbol{A}$$

$\boldsymbol{b}^{\mathrm{T}} = (0, \ldots , 0, \boldsymbol{A}^{k}(k+1{:}n, k)^{\mathrm{T}})$とおきます.

$\boldsymbol{b}^{\mathrm{T}} = (0, \ldots , 0, k+1, \ldots , b_n)$

$\boldsymbol{b}^{*}\cdot\boldsymbol{b} = S^2$ として, $\boldsymbol{w}^{\mathrm{T}} = (0, \ldots , 0, \ b_{k+1}\left(1+\dfrac{|S|}{|b_{k+1}|}\right), b_{k+2}, \ldots , b_n)$とおきます.

すると, $\boldsymbol{P}_k = \mathrm{I} - \alpha\boldsymbol{w}\cdot\boldsymbol{w}^{*}, \alpha = \dfrac{1}{S^2 + |b_{k+1}S|}$ と表せます.

$\boldsymbol{A}(k+1{:}n, k)$に $\boldsymbol{w}(k+1{:}n)$, $\boldsymbol{A}(k, k)$にαを格納します.

エルミート三重対角行列は, ユニタリ対角行列で実三重対角化します.

$\boldsymbol{T} = \boldsymbol{V}^{*}\boldsymbol{H}\boldsymbol{V}$

b.　使用例

固有値の分かっているエルミート行列を三重対角化します.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
c     **example**
      implicit real*8(a-h,o-z)
      parameter(n=2000,k=n,ne=n,max_nev=ne)
      complex*16 a(k,n),b(k,n),c(k,n),d(k,n),
     &          dh(k,n),alpha,beta,
     &          tr(n)
      dimension nev(5),mult(max_nev,2)
      dimension eval(max_nev),evec(k,max_nev),dd(n),sld(n),sud(n)
cc
      pai2=8.0d0*datan(1.0d0)
      coef=dsqrt(1.0d0/(n))
      do j=1,n
      do i=1,n
      part1 =coef*dcos(pai2/n*(i-1)*(j-1))
      part2 =coef*dsin(pai2/n*(i-1)*(j-1))
      d(i,j)=dcmplx(part1,part2)
      dh(i,j)=dcmplx(part1,-part2)
      enddo
      enddo
cc
      do j=1,n
```

```
      do i=1,n
      if(i.eq.j)then
      c(i,j)=dcmplx(dble(i),0.0d0)
      else
      c(i,j)=(0.0d0,0.0d0)
      endif
      enddo
      enddo
cc
cc    d x c -> b
cc

      alpha=(1.0d0,0.0d0)
      beta=(0.0d0,0.0d0)
      call zgemm('No transpose','No transpose',n,n,
     $            n,alpha,d,k,c,k,beta,b,k)
cc
cc    b x d^h -> a
cc
      call zgemm('No transpose','No transpose',n,n,
     $            n,alpha,b,k,dh,k,beta,a,k)
cc
      call  dm_vhtrid(a,k,n,dd,sld,tr,icon)
      if(icon.ne.0)then
      print*,' icon of dm_vhtrid =',icon
      stop
      endif
c
      do i=2,n
      sud(i-1)=sld(i)
      enddo
      sud(n)=0.0d0
c
      nf=1
      nl=n
      ivec=0
      eval_tol=1.0d-15
      clus_tol=1.0d-10
      call dm_vtdevc(dd,sld,sud,n,nf,nl,ivec,
     &                eval_tol,clus_tol,nev,
     &                eval,max_nev,evec,k,mult,icon )
      do i=1,ne,n/20
```

```
      print*,'eigen value in eval(',i,') = ',eval(i)
      enddo

      stop
      end
```

(4)　手法概要

$n \times n$ のエルミート行列 A=AR+iAI は, AR=AR^T, AI=-AI^T を満たします.

ブロック化された Householder 法でエルミート行列をエルミート三重対角行列に変換して, 対角ユニタリー変換で実三重対角行列に変換します.

DM_VJDHECR

エルミートスパース行列の固有値問題(Jacobi-Davidson 法，圧縮行格納法)
CALL DM_VJDHECR(ZH, NZ, NCOL, NFRNZ, N, ITRGT, DTRGT, NSEL, NEV, ITMAX, ITER, IFLAG, DPRM, DEVAL, ZEVEC, KV, DHIS, KH, ICON)

(1) 機能

$n \times n$ のエルミートなスパース行列の指定されたいくつかの固有値および対応する固有ベクトルを Jacobi-Davidson 法で求めます.

$$\boldsymbol{Ax} = \lambda \boldsymbol{x}$$

エルミート行列 $\boldsymbol{A}$ は, 下三角部分を圧縮行格納法で格納します. ベクトル $\boldsymbol{x}$ は n 次元ベクトルです.

(2) パラメタ

ZH .....入力. 行列 $\boldsymbol{A}$ の下三角部分の非零要素を格納します.
ZH(NZ)なる 1 次元複素配列. 圧縮行格納法については, 図 DM_VJDHECR-1 を参照してください.

NZ .....入力. 行列 $\boldsymbol{A}$ の下三角部分にある非零要素の総数.

NCOL.....入力. 圧縮行格納法で使用される列指標で, ZH に格納される要素が何番目の列ベクトルに属するかを示します.
NCOL(NZ)なる一次元配列.

NFRNZ .....入力. 行列 $\boldsymbol{A}$ の各行の下三角部分の非零要素を行方向に圧縮して順次配列 ZH に格納するとき, 対応する行の最初の非零要素が格納される位置を表します. NFRNZ(N+1) = NZ+1 として指定します.
NFRNZ(N+1)なる 1 次元配列.

N.....入力. 行列 $\boldsymbol{A}$ の次数 n.

ITRGT.....入力. 求める固有値の選択方法を指定します($0 \le$ ITRGT ≤ 4).
ITRGT=0: DTRGT に指定したターゲット値の近傍の固有値を優先的に選択します.
ITRGT=1: 絶対値の大きい固有値を優先的に選択します.
ITRGT=2: 絶対値の小さい固有値を優先的に選択します.
ITRGT=3: 数値が大きい固有値を優先的に選択します.
ITRGT=4: 数値が小さい固有値を優先的に選択します.
((3)使用上の注意 a. 注意①,②参照).

DTRGT .....入力. ITRGT=0 のとき, 固有値を選択するターゲット値を指定します. また, ITRGT≠0 であっても以下の場合には, 求める固有値の近傍値を DTRGT に指定することで収束が改善することがあります.
1) DPRM(3)=1 で harmonic アルゴリズムを選択しているとき, テスト副空間 $\langle \boldsymbol{W} \rangle = \langle (\boldsymbol{A} - \tau \boldsymbol{I}) \boldsymbol{V} \rangle$ を設定するためのシフト値 τ として DTRGT を使用します. ((3)使用上の注意 a. 注意②参照).

2) DPRM(9) ≥ 1 のとき, 修正方程式で用いる近似固有値として, 反復開始初期に DTRGT への指定値を使用します. ((3)使用上の注意 a. 注意⑤参照).
3) DPRM(15) ≥ 1 のとき, 修正方程式の前処理行列を設定するためのシフト値 τ として DTRGT への指定値を利用します.
((3)使用上の注意 a. 注意⑦参照).
それ以外の場合には, DTRGT の値は参照されません.

NSEL .....入力. 求める固有値個数を指定します($1 \le NSEL \le N$).
((3)使用上の注意 a. 注意①参照).

NEV .....出力. 収束した固有値個数.

ITMAX .....入力. 反復回数の上限を指定します(≥ 0).

ITER.....出力. Jacobi-Davidson 法での実際の反復回数.

IFLAG.....入力. DPRM に動作調整パラメタを明に指定するかどうかを示します.
IFLAG(i) ≠0 のとき, DPRM(i)に指定したパラメタを使用します($i \le 15$).
IFLAG(i) = 0 のとき, DPRM(i)は参照せずデフォルト値を使用します.
IFLAG(16:32)は機能拡張用のリザーブ域のため, 0 を指定しておきます.
IFLAG(32)なる 1 次元配列.

DPRM .....入力. IFLAG で指定された動作調整用パラメタについて, 値を指定します. アルゴリズムにおける各パラメタ変数の定義については“(4)手法概要”を参照してください.
IFLAG(1:32)が全て 0 であれば DPRM は参照されず, デフォルト値が用いられます. デフォルト値のままでは収束しなかった場合に, パラメタを変えて試みることを奨めます.
DPRM(32)なる 1 次元配列.

DPRM(1): リスタート時に縮小する副空間次元 m_{min} を指定します.
($1 \le m_{min} < N$). デフォルト値は $m_{min} = 50$ です.

DPRM(2): 副空間次元の最大次元 m_{max} を指定します($m_{min} < m_{max} \le N$).
デフォルト値は $m_{max} = m_{min}+30$ です.
((3)使用上の注意 a. 注意⑧参照).

DPRM(3): 射影するテスト副空間の設定方法による, アルゴリズムの種別を指定します.
DPRM(3)=0 のとき, 固有値選択ターゲットがスペクトル端の場合に適した standard アルゴリズムを使用します.
DPRM(3)=1 のとき, 固有値選択ターゲットがスペクトル内部の場合に適した harmonic アルゴリズムを使用します.
デフォルトでは, ITRGT=0 および 2 のときに harmonic アルゴリズムを使用し, それ以外のときに standard アルゴリズムを使用します.

DPRM(4): 収束判定許容閾値を指定します. デフォルトは 1.0D-6 です.
((3)使用上の注意 a. 注意④参照).

DPRM(5): 反復中に得られた近似固有値 θ および近似固有ベクトル $\boldsymbol{u}$ に対し, 残差ノルムを計算する方法を指定します.
DPRM(5)=0 のとき, 近似固有値絶対値に対する相対残差ノルム

$|\boldsymbol{Au} - \theta\boldsymbol{u}| / |\theta|$ を使用します.
DPRM(5)=1 のとき, 行列 $\boldsymbol{A}$ の 1-ノルムに対する相対残差ノルム $|\boldsymbol{Au} - \theta\boldsymbol{u}| / |\boldsymbol{A}|_1$ を使用します.
DPRM(5)=2 のとき, 行列 $\boldsymbol{A}$ の Frobenius ノルムに対する相対残差 $|\boldsymbol{Au} - \theta\boldsymbol{u}| / |\boldsymbol{A}|_{Fro}$ を使用します.
DPRM(5)=3 のとき, 行列 $\boldsymbol{A}$ の∞-ノルムに対する相対残差ノルム $|\boldsymbol{Au} - \theta\boldsymbol{u}| / |\boldsymbol{A}|_\infty$ を使用します.
DPRM(5)=4 のとき, 絶対残差ノルム $|\boldsymbol{Au} - \theta\boldsymbol{u}|$ を使用します.
デフォルトは, DPRM(5)=0 です.
((3)使用上の注意 a. 注意③参照).

DPRM(6): 遅延減次処理における閾値を指定します(≤ 1.0).
デフォルトは DPRM(6)=0.9 です.
((3)使用上の注意 a. 注意④参照).

DPRM(7): 反復開始初期ベクトルを ZEVEC(1:N, 1)に指定するかどうかを指定します.
DPRM(7)=0.0 のとき, ルーチン内部で生成した乱数 ベクトルを反復開始初期ベクトルとして使用します.
DPRM(7)=1.0 のとき, ZEVEC(1:N, 1)に反復開始初期ベクトルを指定してください.
デフォルトは乱数ベクトルを使用します.

DPRM(8): 乱数列生成のための種(*seed*)の値を指定します(≥ 1.0).
デフォルトは 1 です.

DPRM(9): 修正方程式で用いる固有値近似値として, 反復毎に得られる近似固有値 θ ではなく, 固定シフト τ を反復開始初期に用います.
反復回数が DPRM(9)までのときに, 固定シフトを用います.
DPRM(3)=0 のとき, デフォルトは DPRM(9) = 0 です.
DPRM(3)=1 のとき, デフォルトは DPRM(9) = m_{max} です.
((3)使用上の注意 a. 注意⑤参照).

DPRM(10): 修正方程式を解く解法種別を指定します.
DPRM(10)=0 のとき, 修正方程式を用いずに $\boldsymbol{t} = \boldsymbol{r}$ とします.
DPRM(10)=1 のとき, GMRES 法を使用します.
DPRM(10)=2 のとき, BiCGstab(L)法を使用します.
DPRM(10)=11 のとき, MINRES 法を使用します.
デフォルトは MINRES 法です. ((3)使用上の注意 a. 注意⑦,⑧参照).

DPRM(11): 修正方程式解法で使用するパラメタを指定します.
BiCGstab(L)法を用いる場合に, L の値を指定します(≤ 10).
デフォルト値は 4 です.

DPRM(12): 修正方程式を連立一次方程式反復解法を用いて解く場合の反復回数上限を指定します(≥ 1). デフォルトは 30 です.

DPRM(13): 修正方程式を連立一次方程式反復解法を用いて解く場合に, 収束判定閾値を決めるパラメタを指定します(> 0.0).

デフォルト値は 0.7 です. ((3)使用上の注意 a. 注意⑥参照).

DPRM(14): 修正方程式を連立一次方程式反復解法を用いて解く場合に, 収束判定閾値を決めるパラメタを指定します. ($0.0 < DPRM(14) \leq 1.0$). 減次処理毎にリセットされる外部反復に応じたカウンタを l としたとき, 修正方程式求解の収束判定閾値は $DPRM(13) \times DPRM(14)^l$ に設定されます.

デフォルト値は 0.7 です. ((3)使用上の注意 a. 注意⑥参照).

DPRM(15): 修正方程式を連立一次方程式反復解法を用いて解く場合に, 前処理方式を指定します(≤ 1).

DPRM(15)=0 のとき, 前処理を行いません.

DPRM(15)=1 のとき, 対角要素スケーリングによる左側前処理を行います. ((3)使用上の注意 a. 注意⑦参照)

デフォルトは DPRM(15)=0 です.

DPRM(16:32):機能拡張用のリザーブ域です.

DEVAL.....出力. 求まった固有値が DEVAL(1:NEV)に格納されます.

DEVAL(NSEL)なる 1 次元配列.

ZEVEC.....出力. 求まった固有ベクトルが ZEVEC(1:N, 1:NEV)に格納されます.

ZEVEC(KV, NSEL)なる 2 次元複素配列.

入力. IFLAG(7)≠0 かつ DPRM(7)=1.0 のとき, 反復開始初期ベクトルを ZEVEC(1:N, 1)に指定します.

KV.....入力. 配列 ZEVEC の 1 次元目の大きさ($\geq$ N).

DHIS .....出力. DHIS(1:*min*(KH, ITER), 1)に Jacobi-Davidson 法の残差ノルム収束履歴を出力します. DHIS(1:*min*(KH, ITER), 2)に反復毎に解いた修正方程式の最終相対残差ノルムを出力します. DHIS(KH, 2)なる 2 次元配列.

KH.....入力. 配列 DHIS の 1 次元目の大きさ(≥ 0). KH=ITMAX であれば十分です. KH=0 を指定すると, DHIS への出力は抑止されます.

ICON.....出力. コンディションコード.

表 DM_VJDHECR-1 参照.

表 DM_VJDHECR-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし	ー
1000	修正方程式の反復解法処理で breakdown が発生した.	それまでの近似解を用いて処理を続ける.
2000	直交化処理などで零ベクトルが発生した.	必要に応じて乱数ベクトルで空間拡大し処理を続ける.
3000	遅延減次処理における前反復情報回復などのリカバリー処理が動作した.	処理を継続する.
10000	反復回数上限までに固有値が NSEL 個まで求まらなかった.	得られた NEV 個までの固有値および固有ベクトルの結果は正常.

20000	射影により縮小された密行列固有値問題の求解処理に失敗した.	処理を打ち切る. NEV>0 であれば, NEV 個までの固有値および固有ベクトルの結果は正常.
21000	固有値が一つも求まらずに反復回数の上限に達した.	処理を打ち切る. 配列 DEVAL(1)および ZEVEC(1:N,1)には, そのときまでに得られている近似値を出力するが, 精度は保証できない.
29000	内部エラーが発生した.	処理を打ち切る.
30000	N<1, ITRGT<0, ITRGT>4, NSEL<1, NSEL>N, ITMAX<0, KV<N, KH<0.	
30001 ～ 30032	IFLAG または DPRM の値が正しくない.	
31000	NZ, NCOL, NFRNZ の値が正しくない.	

$$A = \begin{bmatrix} 1 & 2+4i & 0 & 0 \\ 2-4i & 5 & 7-3i & 6+9i \\ 0 & 7+3i & 8 & 0 \\ 0 & 6-9i & 0 & 10 \end{bmatrix}$$

$$\mathrm{NFRNZ} = \begin{bmatrix} 1 \\ 2 \\ 4 \\ 6 \\ 8 \end{bmatrix}, \quad ZA = \begin{bmatrix} 1 \\ 2-4i \\ 5 \\ 7+3i \\ 8 \\ 6-9i \\ 10 \end{bmatrix}, \quad \mathrm{NCOL} = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 2 \\ 3 \\ 2 \\ 4 \end{bmatrix}$$

図 DM_VJDHECR-1

(3)　使用上の注意

a.　注意

①　Jacobi Davidson アルゴリズムについて

Jacobi-Davidson 法は確定的な手法ではなく, 一般に行列変形を経る密行列向けの解法ほど堅固な方法ではありません. Jacobi-Davidson 法で得られる結果は初期ベクトルの選択にも依存し, 優先的に選択する固有値の出力順も保証されません. また, 求める固有値・固有ベクトルの個数 NSEL が, 行列次数 n に比べてごく少数の場合に適しています.

アルゴリズム内では種々の動作調整パラメタが使用されており, それらのパラメタを調整することで, 収束が改善することがあります. アルゴリズムにおける各パラメタ変数の説明については“(4)手法概要”を参照してください.

②　ITRGT, DTRGT パラメタについて

本サブルーチンでは, ITRGT に設定する固有値選択方法に応じて, 使用する内部アルゴリズムを指定する DPRM(3)のデフォルト値を自動で切り換えています. IFLAG(3)≠0 として明に DPRM(3)の設定をした場合にはその選択が優先されます. すなわちアルゴリズム適性に合う問題設定であれば, ITRGT=0,2 で standard アルゴリズムを使用したり, ITRGT=1,3,4 で harmonic アルゴリズムを使用することもできます.

なお, 絶対値の小さい固有値を優先的に選択する ITRGT=2 でのデフォルトは harmonic アルゴリズムであるため, DTRGT パラメタの値が参照されることにご注意ください. 適切な値が不明な場合は DTRGT=0.0D0 を指定してください.

③　残差ノルム計算方法について

本サブルーチンでは, 収束判定に用いる残差ノルムの値として近似固有値絶対値に対する相対残差ノルムを用いる計算方法をデフォルトとしています. しかし求める固有値が行列ノルムに比べてはるかに小さなゼロ近傍固有値の場合, 収束判定 $|\boldsymbol{Au} - \theta \boldsymbol{u}| / |\theta| <$ DPRM(4)を満たすのが困難な場合があります. その場合には収束判定閾値 DPRM(4)の値を調整するか DPRM(5)で指定する残差ノルム計算方法を変更することで, 収束が得られる可能性があります.

④　遅延減次処理方式について

本サブルーチンでは解の計算精度をあげるために, 収束判定閾値よりも残差ノルムが小さくなった後も減次処理をしないで反復を続ける方法を採用しています. これを遅延減次処理と呼んでいます. この処理が有効な場合, 残差ノルムが収束とみなせる許容閾値 DPRM(4)よりも小さくなった後, 残差ノルムが前反復と比較して DPRM(6)で指定する比率以下に減少する間は反復を続けてから減次処理を行います. この反復継続中に残差ノルムが前の反復より悪化してしまった場合は, 前の反復の近似固有値および近似固有ベクトルを復元してから減次処理を行っています. DPRM(6)=0.0 と指定した場合, この遅延減次処理は行われません.

⑤ 修正方程式に用いる近似固有値について

反復開始初期段階では, 反復毎に得られる近似固有値 θ が, 求める固有値に遠い値である可能性があります. 探索副空間拡大の良否は修正方程式に用いる近似固有値に大きく影響されるため, 本サブルーチンでは反復開始初期段階では θ ではなく, DTRGT に指定された固定シフト τ の値を修正方程式に用いる方法を採用しています. 何反復目までを反復開始初期段階とみなすかは, 動作調整用パラメタの DPRM(9)で指定できます. スペクトル端の固有値を求めるのに適した standard アルゴリズムを用いる場合には, あらかじめ適当な固有値近似値 τ を設定するのが難しいであろうという配慮から, DPRM(3)=0 の場合のデフォルトは本方式を使用しない DPRM(9)=0 としています.

⑥ 修正方程式求解の収束判定閾値

本サブルーチンでは修正方程式求解に, スパース行列を係数行列とする連立一次方程式の反復解法を用いています. 固有ベクトルを求める Jacobi-Davidson アルゴリズムのループを外側反復と呼び, 修正方程式の反復解法でのループを内側反復と呼びます. 外側反復が収束に近づいていない状況では, 内側反復で行われる修正方程式の求解の精度は高くなくても良いと考えられています. 内側反復処理の節約のため, 修正方程式求解の収束判定閾値は外側反復に応じて変化させることができます. 減次処理毎に 0 にリセットされる外部反復に応じたカウンタを l としたとき, 修正方程式求解の収束判定閾値は DPRM(13)×DPRM(14)^l に設定されます. なお, 内側反復回数の上限は DPRM(12)で指定します.

⑦ 修正方程式反復解法の前処理について

修正方程式に対する連立一次方程式反復解法において, 前処理が有効な場合があります. 前処理種別は DPRM(15)で指定します. このとき修正方程式の前処理行列として, 左右の射影操作を含まない部分の $\boldsymbol{M} \fallingdotseq (\boldsymbol{A} - \tau \boldsymbol{I})$ を設定するためのシフト値 τ として DTRGT への指定値を使用します. τ が求める固有値と離れていたり, 前処理行列が修正方程式の係数行列をうまく近似できていないと収束悪化になる場合もありますので注意が必要です. また, 本サブルーチンで対応しているのは左側前処理であり修正方程式の係数行列がエルミート行列でなくなるため, その解法としては非対称行列向けのアルゴリズムを DPRM(10)に指定する必要があります.

⑧ 使用メモリについて

本サブルーチンでは, 内部で使用する作業域を動的確保しています. このため, メモリが不足すると異常終了する可能性があります. そのサイズはパラメタの設定方法によって異なりますが, standard アルゴリズムの場合は $n\times(2\times m_{\max} + 2\times\mathrm{NSEL})\times 16\mathrm{byte}$ 以上, harmonic アルゴリズムでは $n\times(3\times m_{\max} + 2\times\mathrm{NSEL})\times 16\mathrm{byte}$ 以上が基本アルゴリズム部分で必要であり, 修正方程式解法として GMRES 法を用いる場合にはそれに加えて $n\times$DPRM(12)$\times$16byte が大規模問題での主要な作業域サイズになります.

b. 使用例

1 行あたりの非零要素数が 20 個程度の 10000 次元のエルミート行列の絶対値最大固有値を 10 個と, それに対応する固有ベクトルを求めます.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT NONE
      INTEGER NNZMAX,NMAX,LDK,NZC
      PARAMETER (NMAX=10000,NZC=20)
      PARAMETER (NNZMAX=NMAX*NZC)
      PARAMETER (LDK=10)
      COMPLEX*16 ZH(NNZMAX),ZEVEC(NMAX,LDK)
      COMPLEX*16 RVEC(NMAX),ZW(NMAX)
      REAL*8 DTRGT,DEVAL(LDK),DERR,DPRM(32),DHIS(NMAX,2)
      INTEGER NZ,NCOL(NNZMAX),NFRNZ(NMAX+1),N,ITRGT
      INTEGER IFLAG(32),NSEL,NEV,ITMAX,ITER,LDX,LDH,ICON
      INTEGER I,J,K,NCOLJ

      N=NMAX
      CALL MKSPMAT(N,NZC,ZH,NCOL,NFRNZ)
      NZ=NFRNZ(N+1)-1

      ITMAX = 500
      NSEL = 10
      DO I = 1,32
       IFLAG(I)=0
      ENDDO
      LDX = NMAX
      LDH = NMAX
      DTRGT = 0.0D0
      ITRGT = 1

      CALL DM_VJDHECR(ZH,NZ,NCOL,NFRNZ,N,ITRGT,DTRGT,NSEL,
     &                NEV,ITMAX,ITER,IFLAG,DPRM,
     &                DEVAL,ZEVEC,LDX,DHIS,LDH,ICON)

      PRINT *,'DM_VJDHECR ICON=',ICON
      PRINT *,'ITER=',ITER
      DO K = 1,NEV
!$OMP PARALLEL PRIVATE(I,J,ZW,NCOLJ)
       ZW(1:N) = (0.0D0,0.0D0)
!$OMP DO
       DO I=1,N
```

```
        RVEC(I)=(0.0D0,0.0D0)
        DO J=NFRNZ(I),NFRNZ(I+1)-1
         NCOLJ=NCOL(J)
         RVEC(I)=RVEC(I)+ZH(J)*ZEVEC(NCOLJ,K)
         IF(I.NE.NCOLJ)THEN
          ZW(NCOLJ)=ZW(NCOLJ)+DCONJG(ZH(J))*ZEVEC(I,K)
         ENDIF
        ENDDO
       ENDDO
!$OMP CRITICAL
       DO I=1,N
        RVEC(I)=RVEC(I)+ZW(I)
       ENDDO
!$OMP END CRITICAL
!$OMP END PARALLEL
       DERR=0.0D0
       DO I=1,N
        RVEC(I)=RVEC(I)-DEVAL(K)*ZEVEC(I,K)
        DERR=DERR +DREAL(RVEC(I))**2 +DIMAG(RVEC(I))**2
       ENDDO
       DERR=DSQRT(DERR)
       PRINT*,'EIGEN VALUE',K,'=',DEVAL(K)
       PRINT*,'ERROR=',DERR/DABS(DEVAL(K))
      ENDDO
      END

      SUBROUTINE MKSPMAT(N,NZC,ZH,NCOL,NFRNZ)
      IMPLICIT NONE
      INTEGER N,NZC,NCOL(*),NFRNZ(*)
      COMPLEX*16 ZH(*)
      INTEGER I,IC,ICT,J,K,ISEED,LDW,ICON,NNZ
      PARAMETER(LDW=1350)
      REAL*8 DWORK(NZC),RNDWORK(LDW)
      ISEED=1
      NNZ=0
      DO I=1,N
       NFRNZ(I)=NNZ+1
 10    CALL DVRAU4(ISEED,DWORK,NZC,RNDWORK,LDW,ICON)
       IC=0
       DO J=1,NZC
        ICT=N*DABS(DWORK(J))+1
        IF(ICT.LE.I)THEN
```

```
      DO K=1,IC
       IF(ICT.EQ.NCOL(NNZ-K+1))THEN
        NNZ=NNZ-IC
        GO TO 10
       ENDIF
      ENDDO
      NNZ=NNZ+1
      IC=IC+1
      NCOL(NNZ)=ICT
     ENDIF
    ENDDO
   ENDDO
   NFRNZ(N+1)=NNZ+1
   ISEED = 1
   CALL DVRAN4(0.0D0,1.0D0,ISEED,ZH,2*NNZ,RNDWORK,LDW,
  &            ICON)
   DO I=1,N
    DO J=NFRNZ(I),NFRNZ(I+1)-1
     IF(I.EQ.NCOL(J))ZH(J)=DREAL(ZH(J))+DIMAG(ZH(J))
    ENDDO
   ENDDO
   RETURN
   END
```

(4)　手法概要

本サブルーチンでは, スパース行列に対する固有値問題解法として Jacobi-Davidson 法を用いています. n 次元の行列固有値問題に対して, 反復毎に更新される小さな次元の副空間への射影操作を利用して近似解を求めています. 各反復において, 副空間拡大と近似固有ベクトル抽出の二つのフェーズの処理があります. 副空間拡大フェーズは, 何らかの近似固有ベクトルを基にして近似改善のための修正ベクトルを求める処理になっており, 修正方程式と呼ばれる連立一次方程式を解くことで適切な副空間拡大を実現しています. 副空間からの近似固有ベクトル抽出フェーズでは, 副空間次元に縮小された固有値問題を解くことで近似固有ベクトルを抽出します. 求める固有値がスペクトル範囲の内部にある場合には探索副空間とテスト副空間を別に設定することで適切な近似固有ベクトルが抽出できるように工夫がされています.

Jacobi-Davidson アルゴリズムで使用される各種パラメタの説明のために, 以下に反復ループ内の処理概要について示します.

1. 探索副空間拡大ベクトルを $\boldsymbol{t}$ として用意します.
2. $\boldsymbol{t}$ を用いて探索副空間<$\boldsymbol{V}$> とテスト副空間<$\boldsymbol{W}$>を拡大します. ここで $\boldsymbol{V}$,$\boldsymbol{W}$ は $n \times m$ の行列で, <$\boldsymbol{V}$>は $\boldsymbol{V}$ の列ベクトルで張られる線形空間を示します. standard アルゴリズムで

は $\boldsymbol{W}=\boldsymbol{V}$ で, harmonic アルゴリズムでは τ をターゲット値として$<\boldsymbol{W}> = <(\boldsymbol{A}- \tau \boldsymbol{I})\boldsymbol{V}>$です. アルゴリズム種別は DPRM(3)で指定できます.

3. 副空間への射影操作により縮小された m 次元の密行列固有値問題を解きます.
4. Ritz 値あるいは harmonic Ritz 値と呼ばれる縮小固有値問題の解から, ITRGT および DTRGT で指定する選択方法に応じて固有値を選択します.
5. 選択した固有値に対応する固有ベクトルを元の大規模固有値問題の近似固有ベクトル $\boldsymbol{u}$ として抽出し, その Rayleigh 商を近似固有値 θ とします.
6. $\boldsymbol{u}$ と θ から残差ベクトル $\boldsymbol{r} = \boldsymbol{Au}- \theta \boldsymbol{u}$ を計算します.
7. 残差ノルムを, DPRM(5)に指定された方法で計算します.
8. 収束判定許容値 DPRM(4)および遅延減次処理閾値 DPRM(6)で指定される方法で収束と判定できたら, 減次(deflation)処理を行います.
9. 副空間次元 m が $m_{\max}$ よりも大きくなった場合, リスタート処理により副空間を $m_{\min}$ まで縮小します. $m_{\min}$ と $m_{\max}$ は DPRM(1), DPRM(2)で指定できます.
10. 修正方程式 $(\boldsymbol{I}-\boldsymbol{uu}^*)(\boldsymbol{A}- \theta \boldsymbol{I})(\mathrm{I}-\boldsymbol{uu}^*)\ \boldsymbol{t} =-\boldsymbol{r}$ を解くことで, 次の探索空間拡大ベクトル $\boldsymbol{t}$ を計算します. この処理では DPRM(9)～DPRM(15)を使用します.

説明簡略化のため, 減次処理後に必要な射影操作は省略してあります.

アルゴリズムの詳細については, “付録 1 参考文献一覧表” の[7]を参照して下さい.

DM_VJDNHCR

複素スパース行列の固有値問題(Jacobi-Davidson 法，圧縮行格納法)
CALL DM_VJDNHCR(ZA, NZ, NCOL, NFRNZ, N, ITRGT, ZTRGT, NSEL, NEV, ITMAX, ITER, IFLAG, DPRM, ZEVAL, ZEVEC, KV, DHIS, KH, ICON)

(1)　機能

$n \times n$ の複素スパース行列の指定されたいくつかの固有値および対応する固有ベクトルを Jacobi-Davidson 法で求めます.

$$\boldsymbol{Ax} = \lambda \boldsymbol{x}$$

複素行列 $\boldsymbol{A}$ は, 圧縮行格納法で格納します. ベクトル $\boldsymbol{x}$ は n 次元ベクトルです.

(2)　パラメタ

ZA .....入力. 行列 $\boldsymbol{A}$ の非零要素を格納します.
ZA(NZ)なる 1 次元複素配列. 圧縮行格納法については, 図 DM_VJDNHCR-1 を参照してください.

NZ .....入力. 行列 $\boldsymbol{A}$ の非零要素の総数.

NCOL.....入力. 圧縮行格納法で使用される列指標で, ZA に格納される要素が何番目の列ベクトルに属するかを示します.
NCOL(NZ)なる一次元配列.

NFRNZ .....入力. 行列 $\boldsymbol{A}$ の各行の非零要素を行方向に圧縮して順次配列 ZA に格納するとき, 対応する行の最初の非零要素が格納される位置を表します.
NFRNZ(N+1) = NZ+1 として指定します.
NFRNZ(N+1)なる 1 次元配列.

N.....入力. 行列 $\boldsymbol{A}$ の次数 n.

ITRGT.....入力. 求める固有値の選択方法を指定します($0 \le$ ITRGT ≤ 6).
ITRGT=0:　ZTRGT に指定したターゲット値の近傍の固有値を優先的に選択します.
ITRGT=1:　絶対値の大きい固有値を優先的に選択します.
ITRGT=2:　絶対値の小さい固有値を優先的に選択します.
ITRGT=3:　実数部が大きい固有値を優先的に選択します.
ITRGT=4:　実数部が小さい固有値を優先的に選択します.
ITRGT=5:　虚数部が大きい固有値を優先的に選択します.
ITRGT=6:　虚数部が小さい固有値を優先的に選択します.
((3)使用上の注意 a. 注意①,②参照).

ZTRGT.....入力. ITRGT=0 のとき, 固有値を選択するターゲット値を複素変数で指定します. また, ITRGT≠0 であっても以下の場合には, 求める固有値の近傍値を ZTRGT に指定することで収束が改善することがあります.
1) DPRM(3)=1 で harmonic アルゴリズムを選択しているとき, テスト副空間 $\langle \boldsymbol{W} \rangle = \langle (\boldsymbol{A} - \tau \boldsymbol{I}) \boldsymbol{V} \rangle$ を設定するためのシフト値 τ として ZTRGT を使用

します. ((3)使用上の注意 a. 注意②参照).

2) DPRM(9) ≥ 1 のとき, 修正方程式で用いる近似固有値として, 反復開始初期に ZTRGT への指定値を使用します. ((3)使用上の注意 a. 注意⑤参照).

3) DPRM(15) ≥ 1 のとき, 修正方程式の前処理行列を設定するためのシフト値 τ として ZTRGT への指定値を利用します.
((3)使用上の注意 a. 注意⑦参照).

それ以外の場合には, ZTRGT の値は参照されません.

NSEL .....入力. 求める固有値個数を指定します($1 \le NSEL \le N$).
((3)使用上の注意 a. 注意①参照).

NEV .....出力. 収束した固有値個数.

ITMAX .....入力. 反復回数の上限を指定します(≥ 0).

ITER.....出力. Jacobi-Davidson 法での実際の反復回数.

IFLAG.....入力. DPRM に動作調整パラメタを明に指定するかどうかを示します.
IFLAG(i)≠0 のとき, DPRM(i)に指定したパラメタを使用します($i \le 15$).
IFLAG(i) = 0 のとき, DPRM(i)は参照せずデフォルト値を使用します.
IFLAG(16:32)は機能拡張用のリザーブ域のため, 0 を指定しておきます.
IFLAG(32)なる 1 次元配列.

DPRM .....入力. IFLAG で指定された動作調整用パラメタについて, 値を指定します. アルゴリズムにおける各パラメタ変数の定義については“(4)手法概要”を参照してください.
IFLAG(1:32)が全て 0 であれば DPRM は参照されず, デフォルト値が用いられます. デフォルト値のままでは収束しなかった場合に, パラメタを変えて試みることを奨めます.
DPRM(32)なる 1 次元配列.

DPRM(1): リスタート時に縮小する副空間次元 m_{min} を指定します.
($1 \le m_{min} < N$). デフォルト値は $m_{min} = 50$ です.

DPRM(2): 副空間次元の最大次元 m_{max} を指定します($m_{min} < m_{max} \le N$).
デフォルト値は $m_{max} = m_{min}+30$ です.
((3)使用上の注意 a. 注意⑧参照).

DPRM(3): 射影するテスト副空間の設定方法による, アルゴリズムの種別を指定します.
DPRM(3)=0 のとき, 固有値選択ターゲットがスペクトル端の場合に適した standard アルゴリズムを使用します.
DPRM(3)=1 のとき, 固有値選択ターゲットがスペクトル内部の場合に適した harmonic アルゴリズムを使用します.
デフォルトでは, ITRGT=0 および 2 のときに harmonic アルゴリズムを使用し, それ以外のときに standard アルゴリズムを使用します.

DPRM(4): 収束判定許容閾値を指定します. デフォルトは 1.0D-6 です.
((3)使用上の注意 a. 注意④参照).

DPRM(5): 反復中に得られた近似固有値 θ および近似固有ベクトル $\boldsymbol{u}$ に対し, 残差ノルムを計算する方法を指定します.

DPRM(5)=0 のとき, 近似固有値絶対値に対する相対残差ノルム $|\boldsymbol{Au}-\theta\boldsymbol{u}| / |\theta|$ を使用します.
DPRM(5)=1 のとき, 行列 $\boldsymbol{A}$ の 1-ノルムに対する相対残差ノルム $|\boldsymbol{Au}-\theta\boldsymbol{u}| / |\boldsymbol{A}|_1$ を使用します.
DPRM(5)=2 のとき, 行列 $\boldsymbol{A}$ の Frobenius ノルムに対する相対残差 $|\boldsymbol{Au}-\theta\boldsymbol{u}| / |\boldsymbol{A}|_{\mathrm{Fro}}$ を使用します.
DPRM(5)=3 のとき, 行列 $\boldsymbol{A}$ の∞-ノルムに対する相対残差ノルム $|\boldsymbol{Au}-\theta\boldsymbol{u}| / |\boldsymbol{A}|_\infty$ を使用します.
DPRM(5)=4 のとき, 絶対残差ノルム $|\boldsymbol{Au}-\theta\boldsymbol{u}|$ を使用します.
デフォルトは, DPRM(5)=0 です.
((3)使用上の注意 a. 注意③参照).

DPRM(6): 遅延減次処理における閾値を指定します(≤ 1.0).
デフォルトは DPRM(6)=0.9 です.
((3)使用上の注意 a. 注意④参照).

DPRM(7): 反復開始初期ベクトルを ZEVEC(1:N, 1)に指定するかどうかを指定します.
DPRM(7)=0.0 のとき, ルーチン内部で生成した乱数ベクトルを反復開始初期ベクトルとして使用します.
DPRM(7)=1.0 のとき, ZEVEC(1:N, 1)に反復開始初期ベクトルを指定してください.
デフォルトは乱数ベクトルを使用します.

DPRM(8): 乱数列生成のための種(*seed*)の値を指定します(≥ 1.0).
デフォルトは 1 です.

DPRM(9): 修正方程式で用いる固有値近似値として, 反復毎に得られる近似固有値 θ ではなく, 固定シフト τ を反復開始初期に用います.
反復回数が DPRM(9)までのときに, 固定シフトを用います.
DPRM(3)=0 のとき, デフォルトは DPRM(9) = 0 です.
DPRM(3)=1 のとき, デフォルトは DPRM(9) = $m_{\max}$ です.
((3)使用上の注意 a. 注意⑤参照).

DPRM(10): 修正方程式を解く解法種別を指定します.
DPRM(10)=0 のとき, 修正方程式を用いずに $\boldsymbol{t}=\boldsymbol{r}$ とします.
DPRM(10)=1 のとき, GMRES 法を使用します.
DPRM(10)=2 のとき, BiCGstab(L)法を使用します.
デフォルトは GMRES 法です. ((3)使用上の注意 a. 注意⑧参照).

DPRM(11): 修正方程式解法で使用するパラメタを指定します.
BiCGstab(L)法を用いる場合に, L の値を指定します(≤ 10).
デフォルト値は 4 です.

DPRM(12): 修正方程式を連立一次方程式反復解法を用いて解く場合の反復回数上限を指定します(≥ 1). デフォルトは 30 です.

DPRM(13): 修正方程式を連立一次方程式反復解法を用いて解く場合に, 収束判定閾値を決めるパラメタを指定します(> 0.0).
デフォルト値は 0.7 です. ((3)使用上の注意 a. 注意⑥参照).

DPRM(14): 修正方程式を連立一次方程式反復解法を用いて解く場合に, 収束判定閾値を決めるパラメタを指定します. ($0.0 < DPRM(14) \leq 1.0$). 減次処理毎にリセットされる外部反復に応じたカウンタを l としたとき, 修正方程式求解の収束判定閾値は $DPRM(13) \times DPRM(14)^l$ に設定されます.
デフォルト値は 0.7 です. ((3)使用上の注意 a. 注意⑥参照).

DPRM(15): 修正方程式を連立一次方程式反復解法を用いて解く場合に, 前処理方式を指定します(≤ 1).
DPRM(15)=0 のとき, 前処理を行いません.
DPRM(15)=1 のとき, 対角要素スケーリングによる左側前処理を行います. ((3)使用上の注意 a. 注意⑦参照)
デフォルトは DPRM(15)=0 です.

DPRM(16:32):機能拡張用のリザーブ域です.

ZEVAL .....出力. 求まった固有値が ZEVAL(1:NEV)に格納されます.
ZEVAL(NSEL)なる 1 次元複素配列.

ZEVEC.....出力. 求まった固有ベクトルが ZEVEC(1:N, 1:NEV)に格納されます.
ZEVEC(KV, NSEL)なる 2 次元複素配列.
入力. IFLAG(7)≠0 かつ DPRM(7)=1.0 のとき, 反復開始初期ベクトルを ZEVEC(1:N, 1)に指定します.

KV.....入力. 配列 ZEVEC の 1 次元目の大きさ($\geq$ N).

DHIS .....出力. DHIS(1:*min*(KH, ITER), 1)に Jacobi-Davidson 法の残差ノルム収束履歴を出力します. DHIS(1:*min*(KH, ITER), 2)に反復毎に解いた修正方程式の最終相対残差ノルムを出力します. DHIS(KH, 2)なる 2 次元配列.

KH.....入力. 配列 DHIS の 1 次元目の大きさ(≥ 0). KH=ITMAX であれば十分です. KH=0 を指定すると, DHIS への出力は抑止されます.

ICON.....出力. コンディションコード.
表 DM_VJDNHCR-1 参照.

表 DM_VJDNHCR-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし	ー
1000	修正方程式の反復解法処理で breakdown が発生した.	それまでの近似解を用いて処理を続ける.
2000	直交化処理などで零ベクトルが発生した.	必要に応じて乱数ベクトルで空間拡大し処理を続ける.
3000	遅延減次処理における前反復情報回復などのリカバリー処理が動作した.	処理を継続する.
10000	反復回数上限までに固有値が NSEL 個まで求まらなかった.	得られた NEV 個までの固有値および固有ベクトルの結果は正常.

20000	射影により縮小された密行列固有値問題の求解処理に失敗した.	処理を打ち切る. NEV>0 であれば, NEV 個までの固有値および固有ベクトルの結果は正常.
21000	固有値が一つも求まらずに反復回数の上限に達した.	処理を打ち切る. 配列 ZEVAL(1)および ZEVEC(1:N,1)には, そのときまでに得られている近似値を出力するが, 精度は保証できない.
29000	内部エラーが発生した.	処理を打ち切る.
30000	N<1, ITRGT<0, ITRGT>6, NSEL<1, NSEL>N, ITMAX<0, KV<N, KH<0.	
30001 ～ 30032	IFLAG または DPRM の値が正しくない.	
31000	NZ, NCOL, NFRNZ の値が正しくない.	

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 & 0 \\ 4 & 5 & 0 & 6 \\ 0 & 7 & 8 & 9 \\ 0 & 0 & 10 & 11 \end{bmatrix}$$

$$\mathbf{NFRNZ} = \begin{bmatrix} 1 \\ 4 \\ 7 \\ 10 \\ 12 \end{bmatrix}, \quad \mathbf{ZA} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \end{bmatrix}, \quad \mathbf{NCOL} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \\ 2 \\ 4 \\ 2 \\ 3 \\ 4 \\ 3 \\ 4 \end{bmatrix}$$

図 DM_VJDNHCR-1

(3) 使用上の注意

a. 注意

① Jacobi Davidson アルゴリズムについて

Jacobi-Davidson 法は確定的な手法ではなく, 一般に行列変形を経る密行列向けの解法ほど堅固な方法ではありません. Jacobi-Davidson 法で得られる結果は初期ベクトルの選択にも依存し, 優先的に選択する固有値の出力順も保証されません. また, 求める固有値・固有ベクトルの個数 NSEL が, 行列次数 *n* に比べてごく少数の場合に適しています.

アルゴリズム内では種々の動作調整パラメタが使用されており, それらのパラメタを調整することで, 収束が改善することがあります. アルゴリズムにおける各パラメタ変数の説明については“(4)手法概要”を参照してください.

② ITRGT, ZTRGT パラメタについて

本サブルーチンでは, ITRGT に設定する固有値選択方法に応じて, 使用する内部アルゴリズムを指定する DPRM(3)のデフォルト値を自動で切り換えています. IFLAG(3)≠0 として明に DPRM(3)の設定をした場合にはその選択が優先されます. すなわちアルゴリズム適性に合う問題設定であれば, ITRGT=0,2 で standard アルゴリズムを使用したり, ITRGT=1,3,4,5,6 で harmonic アルゴリズムを使用することもできます.

なお, 絶対値の小さい固有値を優先的に選択する ITRGT=2 でのデフォルトは harmonic アルゴリズムであるため, ZTRGT パラメタの値が参照されることにご注意ください. 適切な値が不明な場合は ZTRGT=(0.0, 0.0)を指定してください.

③ 残差ノルム計算方法について

本サブルーチンでは, 収束判定に用いる残差ノルムの値として近似固有値絶対値に対する相対残差ノルムを用いる計算方法をデフォルトとしています. しかし求める固有値が行列ノルムに比べてはるかに小さなゼロ近傍固有値の場合, 収束判定 $|\boldsymbol{Au} - \theta \boldsymbol{u}| / |\theta| <$ DPRM(4)を満たすのが困難な場合があります. その場合には収束判定閾値 DPRM(4)の値を調整するか DPRM(5)で指定する残差ノルム計算方法を変更することで, 収束が得られる可能性があります.

④ 遅延減次処理方式について

本サブルーチンでは解の計算精度をあげるために, 収束判定閾値よりも残差ノルムが小さくなった後も減次処理をしないで反復を続ける方法を採用しています. これを遅延減次処理と呼んでいます. この処理が有効な場合, 残差ノルムが収束とみなせる許容閾値 DPRM(4)よりも小さくなった後, 残差ノルムが前反復と比較して DPRM(6)で指定する比率以下に減少する間は反復を続けてから減次処理を行います. この反復継続中に残差ノルムが前の反復より悪化してしまった場合は, 前の反復の近似固有値および近似固有ベクトルを復元してから減次処理を行っています. DPRM(6)=0.0 と指定した場合, この遅延減次処理は行われません.

⑤　修正方程式に用いる近似固有値について

反復開始初期段階では, 反復毎に得られる近似固有値 θ が, 求める固有値に遠い値である可能性があります. 探索副空間拡大の良否は修正方程式に用いる近似固有値に大きく影響されるため, 本サブルーチンでは反復開始初期段階では θ ではなく, ZTRGT に指定された固定シフト τ の値を修正方程式に用いる方法を採用しています. 何反復目までを反復開始初期段階とみなすかは, 動作調整用パラメタの DPRM(9)で指定できます. スペクトル端の固有値を求めるのに適した standard アルゴリズムを用いる場合には, あらかじめ適当な固有値近似値 τ を設定するのが難しいであろうという配慮から, DPRM(3)=0 の場合のデフォルトは本方式を使用しない DPRM(9)=0 としています.

⑥　修正方程式求解の収束判定閾値

本サブルーチンでは修正方程式求解に, スパース行列を係数行列とする連立一次方程式の反復解法を用いています. 固有ベクトルを求める Jacobi-Davidson アルゴリズムのループを外側反復と呼び, 修正方程式の反復解法でのループを内側反復と呼びます. 外側反復が収束に近づいていない状況では, 内側反復で行われる修正方程式の求解の精度は高くなくても良いと考えられています. 内側反復処理の節約のため, 修正方程式求解の収束判定閾値は外側反復に応じて変化させることができます. 減次処理毎に 0 にリセットされる外部反復に応じたカウンタを l としたとき, 修正方程式求解の収束判定閾値は DPRM(13)×DPRM(14)^l に設定されます. なお, 内側反復回数の上限は DPRM(12)で指定します.

⑦　修正方程式反復解法の前処理について

修正方程式に対する連立一次方程式反復解法において, 前処理が有効な場合があります. 前処理種別は DPRM(15)で指定します. このとき修正方程式の前処理行列として, 左右の射影操作を含まない部分の $\boldsymbol{M} \fallingdotseq (\boldsymbol{A} - \tau \boldsymbol{I})$ を設定するためのシフト値 τ として ZTRGT への指定値を使用します. τ が求める固有値と離れていたり, 前処理行列が修正方程式の係数行列をうまく近似できていないと収束悪化になる場合もありますので注意が必要です.

⑧　使用メモリについて

本サブルーチンでは, 内部で使用する作業域を動的確保しています. このため, メモリが不足すると異常終了する可能性があります. そのサイズはパラメタの設定方法によって異なりますが, standard アルゴリズムの場合は $n\times(2\times m_{\max} + 2\times\mathrm{NSEL} + 5)\times 16\mathrm{byte}$ 以上, harmonic アルゴリズムでは $n\times(3\times m_{\max} + 2\times\mathrm{NSEL} + 5)\times 16\mathrm{byte}$ 以上が基本アルゴリズム部分で必要であり, 修正方程式解法として GMRES 法を用いる場合にはそれに加えて $n\times\mathrm{DPRM(12)}\times 16\mathrm{byte}$ が大規模問題での主要な作業域サイズになります.

b.　使用例

1 行あたりの非零要素数が 20 個の 10000 次元のランダム行列の絶対値最大固有値を 10 個と, それに対応する固有ベクトルを求めます.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは,

OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT NONE
      INTEGER NNZMAX,NMAX,LDK,NZC
      PARAMETER (NMAX=10000,NZC=20)
      PARAMETER (NNZMAX=NMAX*NZC)
      PARAMETER (LDK=10)
      COMPLEX*16 ZA(NNZMAX),ZTRGT,ZEVAL(LDK),ZEVEC(NMAX,LDK)
      COMPLEX*16 RVEC(NMAX)
      REAL*8 DERR,DPRM(32),DHIS(NMAX,2)
      INTEGER NZ,NCOL(NNZMAX),NFRNZ(NMAX+1),N,ITRGT,IFLAG(32)
      INTEGER NSEL,NEV,ITMAX,ITER,I,J,K,ICON,LDX,LDH

      N=NMAX
      CALL MKSPMAT(N,NZC,ZA,NCOL,NFRNZ)
      NZ=NFRNZ(N+1)-1

      ITMAX = 500
      NSEL = 10
      DO I = 1,32
       IFLAG(I)=0
      ENDDO
      LDX = NMAX
      LDH = NMAX
      ZTRGT = (0.0D0,0.0D0)
      ITRGT = 1

      CALL DM_VJDNHCR(ZA,NZ,NCOL,NFRNZ,N,ITRGT,ZTRGT,NSEL,NEV,
     &       ITMAX,ITER,IFLAG,DPRM,ZEVAL,ZEVEC,LDX,DHIS,LDH,ICON)

      PRINT *,'DM_VJDNHCR ICON=',ICON
      PRINT *,'ITER=',ITER
      DO K = 1,NEV
       RVEC(1:N)=(0.0D0,0.0D0)
!$OMP PARALLEL DO PRIVATE(J)
       DO I=1,N
        DO J=NFRNZ(I),NFRNZ(I+1)-1
         RVEC(I)=RVEC(I)+ZA(J)*ZEVEC(NCOL(J),K)
        ENDDO
        RVEC(I)=RVEC(I)-ZEVAL(K)*ZEVEC(I,K)
       ENDDO
```

```
      DERR=0.0D0
      DO I=1,N
       DERR=DERR +DREAL(RVEC(I))**2 +DIMAG(RVEC(I))**2
      ENDDO
      DERR=DSQRT(DERR)
      PRINT*,'EIGEN VALUE',K,'=',ZEVAL(K)
      PRINT*,'ERROR=',DERR/CDABS(ZEVAL(K))
     ENDDO
     STOP
     END

     SUBROUTINE MKSPMAT(N,NZC,ZA,NCOL,NFRNZ)
     IMPLICIT NONE
     INTEGER N,NZC,NCOL(*),NFRNZ(*)
     COMPLEX*16 ZA(*)
     INTEGER I,IC,ICT,J,K,ISEED,LDW,ICON
     PARAMETER(LDW=1350)
     REAL*8 DWORK(NZC),RNDWORK(LDW)
     ISEED=1
     CALL DVRAN4(0.0D0,1.0D0,ISEED,ZA,2*N*NZC,RNDWORK,LDW,ICON)
     ISEED=1
     DO I=1,N
      NFRNZ(I)=(I-1)*NZC+1
10    CALL DVRAU4(ISEED,DWORK,NZC,RNDWORK,LDW,ICON)
      IC=(I-1)*NZC
      DO J=1,NZC
       ICT=N*DABS(DWORK(J))+1
       DO K=1,J-1
        IF(ICT.EQ.NCOL(IC-K+1))GO TO 10
       ENDDO
       IC=IC+1
       NCOL(IC)=ICT
      ENDDO
     ENDDO
     NFRNZ(N+1)=IC+1
     RETURN
     END
```

(4)　手法概要

本サブルーチンでは, スパース行列に対する固有値問題解法として Jacobi-Davidson 法を用いています. n 次元の行列固有値問題に対して, 反復毎に更新される小さな次元の副空間へ

の射影操作を利用して近似解を求めています. 各反復において, 副空間拡大と近似固有ベクトル抽出の二つのフェーズの処理があります. 副空間拡大フェーズは, 何らかの近似固有ベクトルを基にして近似改善のための修正ベクトルを求める処理になっており, 修正方程式と呼ばれる連立一次方程式を解くことで適切な副空間拡大を実現しています. 副空間からの近似固有ベクトル抽出フェーズでは, 副空間次元に縮小された固有値問題を解くことで近似固有ベクトルを抽出します. 求める固有値がスペクトル範囲の内部にある場合には探索副空間とテスト副空間を別に設定することで適切な近似固有ベクトルが抽出できるように工夫がされています.

Jacobi-Davidson アルゴリズムで使用される各種パラメタの説明のために, 以下に反復ループ内の処理概要について示します.

1. 探索副空間拡大ベクトルを $\boldsymbol{t}$ として用意します.
2. $\boldsymbol{t}$ を用いて探索副空間<$\boldsymbol{V}$> とテスト副空間<$\boldsymbol{W}$>を拡大します. ここで $\boldsymbol{V}$,$\boldsymbol{W}$ は $n\times m$ の行列で, <$\boldsymbol{V}$>は $\boldsymbol{V}$ の列ベクトルで張られる線形空間を示します. standard アルゴリズムでは $\boldsymbol{W}$=$\boldsymbol{V}$ で, harmonic アルゴリズムでは τ をターゲット値として<$\boldsymbol{W}$> = <($\boldsymbol{A}-\tau\boldsymbol{I}$)$\boldsymbol{V}$>です. アルゴリズム種別は DPRM(3)で指定できます.
3. 副空間への射影操作により縮小された m 次元の密行列固有値問題を解きます.
4. Ritz 値あるいは harmonic Ritz 値と呼ばれる縮小固有値問題の解から, ITRGT および ZTRGT で指定する選択方法に応じて固有値を選択します.
5. 選択した固有値に対応する固有ベクトルを元の大規模固有値問題の近似固有ベクトル $\boldsymbol{u}$ として抽出し, その Rayleigh 商を近似固有値 θ とします.
6. $\boldsymbol{u}$ と θ から残差ベクトル $\boldsymbol{r}=\boldsymbol{Au}-\theta\boldsymbol{u}$ を計算します.
7. 残差ノルムを, DPRM(5)に指定された方法で計算します.
8. 収束判定許容値 DPRM(4)および遅延減次処理閾値 DPRM(6)で指定される方法で収束と判定できたら, 減次(deflation)処理を行います.
9. 副空間次元 m が $m_{\max}$ よりも大きくなった場合, リスタート処理により副空間を $m_{\min}$ まで縮小します. $m_{\min}$ と $m_{\max}$ は DPRM(1), DPRM(2)で指定できます.
10. 修正方程式 ($\boldsymbol{I}-\boldsymbol{uu}^*$)($\boldsymbol{A}-\theta\boldsymbol{I}$)(I$-\boldsymbol{uu}^*$) $\boldsymbol{t}=-\boldsymbol{r}$ を解くことで, 次の探索空間拡大ベクトル $\boldsymbol{t}$ を計算します. この処理では DPRM(9)～DPRM(15)を使用します.

説明簡略化のため, 減次処理後に必要な射影操作は省略してあります.

アルゴリズムの詳細については, “付録 1 参考文献一覧表” の[7]を参照して下さい.

DM_VLAX

実行列の連立 1 次方程式 (ブロック化された LU 分解法)
CALL DM_VLAX(A, K, N, B, EPSZ, ISW, IS, IP, ICON)

(1)　機能

実係数連立 1 次方程式をブロック化された外積形の LU 分解法で解きます.

ただし $\boldsymbol{A}$ は $n \times n$ の正則な実行列, $\boldsymbol{b}$ は n 次元の実定数ベクトル, $\boldsymbol{x}$ は n 次元の解ベクトルです. ($n \geq 1$)

$$\boldsymbol{Ax} = \boldsymbol{b}$$

(2)　パラメタ

A.....入力. 行列 $\boldsymbol{A}$ を A(1:N, 1:N)に格納します.
出力. 行列 $\boldsymbol{L}$ と行列 $\boldsymbol{U}$ が A(1:N, 1:N)に格納されます.
A(K, N)なる倍精度実数型 2 次元配列.
演算後, A(1:N, 1:N)以外の A の値は保証されません.

K.....入力. 格納配列 A の 1 次元目の大きさ(≥ N).

N.....入力. 行列 $\boldsymbol{A}$ の次数 n.

B.....入力. 定数ベクトル $\boldsymbol{b}$.
出力. 解ベクトル $\boldsymbol{x}$.
B(N)なる倍精度実数型 1 次元配列.

EPSZ.....入力. ピボットの相対零判定値(≥ 0.0)
0.0 のときは標準値が採用されます. ((3)使用上の注意 a. 注意①参照).

ISW.....入力. 制御情報.
同一の係数行列をもつ k (≥ 1)組の方程式を解くとき, 次のように指定します.
ISW=1 のとき 1 組目の方程式を解きます.
ISW=2 のとき 2 組目以降の方程式を解きます. ただし, このとき B の値だけを新しい定数ベクトルの値に変え, それ以外のパラメタはそのままで使います.
((3)使用上の注意 a. 注意②参照).

IS.....出力. 行列 $\boldsymbol{A}$ の行列式を求めるための情報, 演算後の配列 A の n 個の対角要素と IS の値を掛け合わせると行列式が得られます.
((3)使用上の注意 a. 注意②参照).

IP.....出力. 部分ピボッティングによる行の入換えの履歴を示すトランスポジションベクトル. 大きさ n の 1 次元配列.

ICON.....出力. コンディションコード.
表 DM_VLAX-1 参照

表 DM_VLAX-1 コンディションコード

コード	意　　味	処理内容
0	エラーなし	ー
20000	行列 ***A*** のある行の要素がすべて零であったか, またはピボットが相対的に零となった. 行列 ***A*** は非正則の可能性が強い	処理を打ち切る.
30000	K<N, N<1, EPSZ<0.0	

(3)　使用上の注意

a.　注意

①　EPSZ に値を設定した場合, ピボットが, EPSZ 以下なら零とみなし, ICON=20000 で処理を打ち切ります.EPSZ の標準値は丸め誤差単位を u としたとき, EPSZ= 16u です. なおピボットが小さくなっても計算を続行させる場合には, EPSZ へ極小値を与えればよいのですが, その結果は保証されません.

②　同一の係数行列を持つ連立 1 次方程式をいくつか続けて解く場合には, 2 回目以降 ISW=2 として解くと, 係数行列 ***A*** の LU 分解過程を省略するので計算時間が少なくなります. なおこお場合 IS の値は, ISW=1 のときの値が保証されます.

③　本サブルーチンは, 内部で DM_VALU および DM_VLUX を呼び出しています. 本機能をパラレルリージョンで生成されるスレッドから呼び出し, 並列実行するスレッド数を実行環境ルーチン OMP_SET_NUM_THREADS( )で設定するときは, DM_VALU および DM_VLUX を直接呼出し, そのおのおのの直前で OMP_SET_NUM_THREADS( )を呼び出して設定して下さい.

b.　使用例

4000 × 4000 の係数行列を持つ連立 1 次方程式を解きます.
(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(4001,4000)
      DIMENSION IP(4000),B(4000)
C
      N=4000
!$OMP PARALLEL DEFAULT(PRIVATE) SHARED(A,B,N)

!$OMP DO
      DO J=1,N
      DO I=1,N
      A(I,J)=MIN(I,J)
```

```
      ENDDO
      ENDDO
!$OMP END DO

!$OMP DO
      DO I=1,N
      B(I)=I*(I+1)/2+I*(N-I)
      ENDDO
!$OMP END DO

!$OMP END PARALLEL
C
      K=4001
      CALL DM_VLAX(A,K,N,B,0.0D0,1,IS,IP,ICON)
      WRITE(6,610)ICON
      IF(ICON.GE.20000)STOP

      S=1.0D0
!$OMP PARALLEL DO DEFAULT(PRIVATE) SHARED(A,N)
!$OMP+          REDUCTION(*:S)
      DO I=1,N
      S=S*A(I,I)
      ENDDO
!$OMP END PARALLEL DO

      DET=IS*S

C
      WRITE(6,620)(I,B(I),I=1,10)
      WRITE(6,630)DET
  610 FORMAT(1H0,10X,16HCONDITION CODE =,I5)
  620 FORMAT(1H0,10X,15HSOLUTION VECTOR
     */(10X,3(1H(,I3,1H),D23.16)))
  630 FORMAT(1H0,10X,
     *27HDETERMINANT OF THE MATRIX =,D23.16)
      END
```

DM_VLBX

実バンド行列の連立 1 次方程式 (ガウスの消去法)
CALL DM_VLBX(A, K, N, NH1, NH2, B, EPSZ, ISW, IS, IP, ICON)

(1) 機能

バンド行列の連立 1 次方程式をガウスの消去法により解きます.

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

ただし, $\boldsymbol{A}$ は $n \times n$, 下バンド幅 h_1, 上バンド幅 h_2 のバンド行列, $\boldsymbol{b}$ は n 次元の実定数ベクトル, $\boldsymbol{x}$ は n 次元の解ベクトルです. $n>h_1\geq 0$, $n>h_2\geq 0$.

(2) パラメタ

A.....入力. バンド係数行列 $\boldsymbol{A}$ を格納します.
A(NH1+1:2×NH1+NH2+1, 1:N)に行列 $\boldsymbol{A}$ を格納します.
A(1:NH1,1:N)でバンドの外側の行列 A の要素は, ゼロを設定します.
図 DM_VLBX-1 参照
出力. LU 分解された行列 $\boldsymbol{L}$ および $\boldsymbol{U}$ が格納されます.
図 DM_VLBX-2 参照.
A(K, N)なる倍精度実数型の 2 次元配列.
A(2×NH1+NH2+2:K, 1:N)の値は保証されません.

K.....入力. 配列 A の整合寸法 (≥2×NH1+NH2+1).

N.....入力. 行列 $\boldsymbol{A}$ の次数 n.

NH1.....入力. 下バンド巾の大きさ h_1.

NH2.....入力. 上バンド巾の大きさ h_2.

B.....入力. 定数ベクトル $\boldsymbol{b}$.
出力. 解ベクトル $\boldsymbol{x}$.
大きさ n の 1 次元配列.

EPSZ.....入力. ピボットの零判定値(≥0.0).
0.0 のときは, 標準値が設定されます.
((3)使用上の注意 a. 注意①参照).

ISW.....入力. 制御情報.
同一係数行列を k(k≥1)組の方程式を解くとき, 次のとおり指定してください.
ISW=1 のとき　1 組目の方程式を解きます.
ISW=2 のとき　2 組目以降の方程式を解きます.
ただし, このとき B の値だけを新しい定数ベクトル $\boldsymbol{b}$ の値に変え, それ以外のパラメタはそのまま使用してください.

IS.....出力. 行ベクトルの入換えの回数を示します.
1 のとき　偶数回の入換え.
-1 のとき　奇数回の入換え.
((3)使用上の注意 a. 注意③参照).

IP.....出力. 大きさ n の 1 次元配列. 行の入れ換えの履歴情報を格納するトランスポジションベクトルが格納されます.
((3)使用上の注意 a. 注意②参照).

ICON.....出力. コンディションコード.
表 DM_VLBX-1 参照

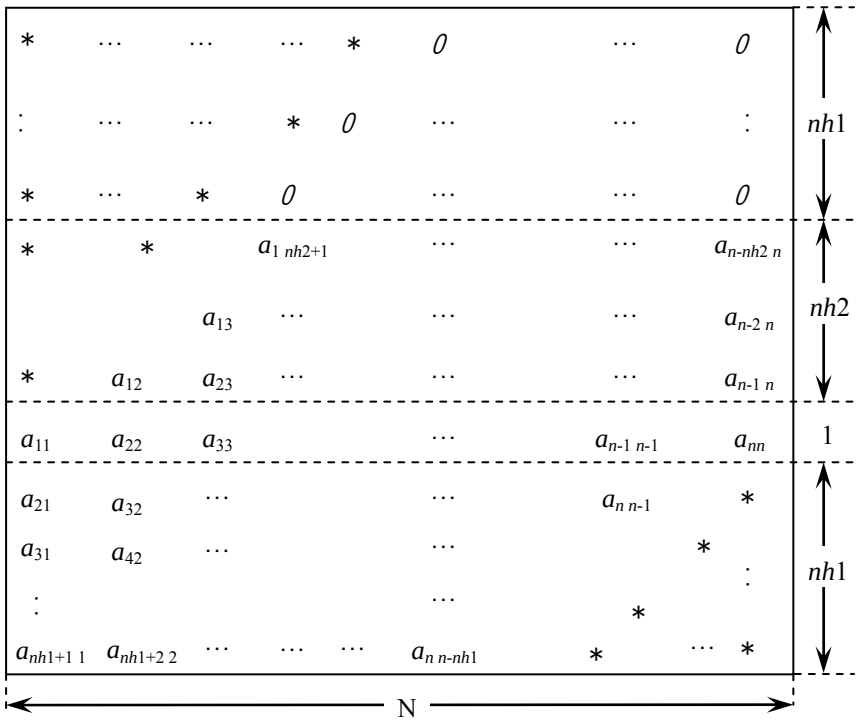


図 DM_VLBX-1　配列 A に行列 ***A*** を格納する方法

バンド行列 ***A*** の対角要素 a_{ii}, i=1, ... , n が A (h_1+h_2+1, 1:n)に, 格納されるように, 行列 ***A*** の列ベクトルを配列 A の列に連続に格納します.
上バンド行列部分, $a_{j-i,\,j}$, i =1, ... , h_2, j=1, ... , n, j-i≥1 を, A(h_1+1:h_1+h_2, 1:n)に格納します.
下バンド行列部分, $a_{j+i,\,j}$, i =1, ... , h_1, j=1, ... , n, j+i≤n を, A(h_1+h_2+2:2×h_1+h_2+1, 1:n)に格納します.
配列 A(1:h_1, 1:n)で, バンドの外側の行列 ***A*** の要素はゼロを設定してください.
*は不定値を示します.

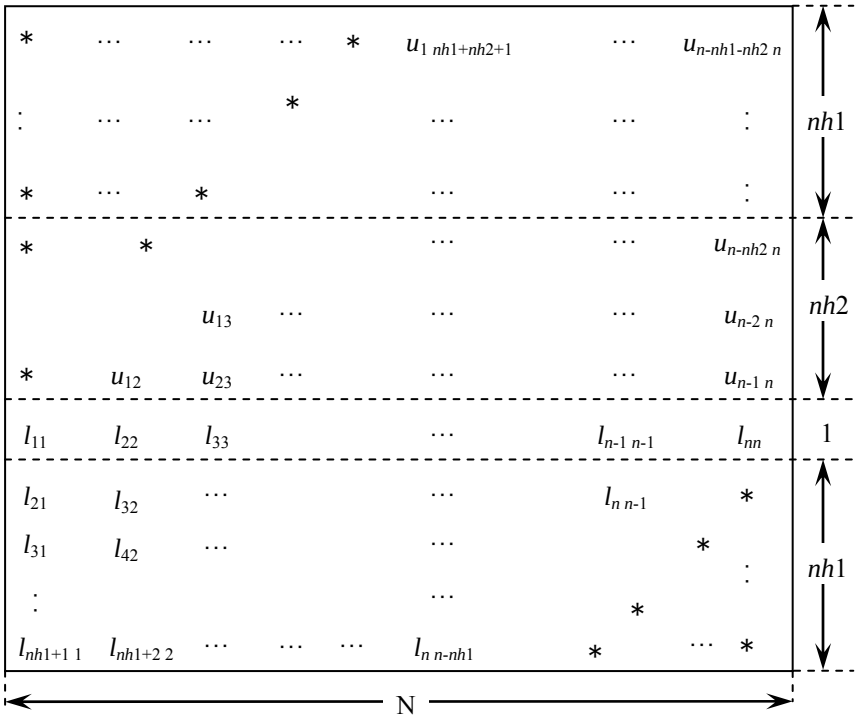


図 DM_VLBX-2　LU 分解された行列 ***L*** および ***U*** の配列 A に格納される方法

LU 分解された単位上バンド行列の対角要素を除いた部分, $u_{j-i+1,\,j}$, i =1, ... , h_1+h_2, j=1, ... , n, j-i+1≥1 が A(1:h_1+h_2, 1:n)に格納されます.
下バンド行列, $l_{j+i,\,j}$, i=0, ... , h_2, j=1, ... , n, j+i≤n が A(h_1+h_2+1:2×h_1+h_2+1, 1:n)に格納されます.
*は不定値を示します.

表 DM_VLBX-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし.	―
20000	行列 ***A*** のある行の要素がすべて零であったか, 又はピボットが相対的に零となった. 行列 ***A*** は非正則の可能性が強い.	処理を打ち切る.
30000	N<1　NH1≥N,　NH1<0,　NH2≥N,　NH2<0, K<2×NH1+NH2, EPSZ<0	

(3)　使用上の注意

a.　注意

①　EPSZ が設定されると, LU 分解の過程でピボットが EPSZ 以下のときは相対的に零と見なし, ICON=20000 で処理を打ち切ります. EPSZ の標準値は丸め誤差の単位を u としたとき 16×u です. なおピボットが小さくても処理を続行させたいときには, EPSZ に極小の値を設定すればよいですが, その結果は保証されません.

② 本サブルーチンでは, 部分ピボッティングに伴い, 行ベクトルの交換を行っています. すなわち, 分解の J 段階(J=1, ... , n)において第 I 行(I≥J)がピボット行として選択された場合には, 第 I 行と第 J 行の内容が交換されています. そしてその履歴を示すために IP(J)に I が格納されます.

③ 行列式の値は, IS および A(h_1+h_2+1, i), i=1, ... , n を掛け合わせることで求めることができます.

b. 使用例

n=10000, h_1=2000, h_2=3000 の実バンド行列を入力して, バンド行列の連立 1 次方程式を解きます.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
      implicit real*8(a-h,o-z)
      parameter(nh1=2000,nh2=3000,n=10000)
      parameter(ka=2*nh1+nh2+1,n2=n)
      real*8 a(ka,n2),b(n),dwork(4500)
      integer ip(n)

c
      ix=123
      nwork=4500
      nn=nh1+nh2+1
      do i=1,n
      call dvrau4(ix,a(nh1+1,i),nn,dwork,nwork,icon)
      do j=1,nh1+nh2+1
      enddo
      enddo
c
c     zero clear
c
      print*,'nh1=',nh1,',nh2=',nh2,',n=',n
c
c     a(1:nh1,n)=0.0d0
c
      do j=1,n
      do i=1,nh1
      a(i,j)=0.0d0
      enddo
      enddo
c
```

```
c     left upper triangular part
c
      do j=1,nh2
      do i=1,nh2+1-j
      a(i+nh1,j)=0.0d0
      enddo
      enddo
c
c    right rower triangular part
c
      nbase=2*nh1+nh2+1
      do j=1,nh1
      do i=1,j
      a(nbase-i+1,n-nh1+j)=0.0d0
      enddo
      enddo
c
c     set right hand constant vector
c
      do i=1,n
      b(i)=0.0d0
      enddo
c
      do i=1,n
      nptr=i-1
      do j=max(nptr+1-nh2,1),min(n,nptr+nh1+1)
      b(j)=b(j)+a(j-i+nh1+nh2+1,i)
      enddo
      enddo


c
      epsz=0.0d0
      isw=1
      call gettod(tt1)
      call dm_vlbx(a,ka,n,nh1,nh2,b,epsz,isw,is,ip,icon)
      call gettod(tt2)
      print*,'time (wall clock)=',(tt2-tt1)*1.0d-6
c
      tmp=0.0d0
      do i=1,n
      tmp=max(tmp,dabs(b(i)-1))
      enddo
```

```
c
      print*,'maximum error =',tmp
c
      stop
      end
```

DM_VLCSPSXCR1

非エルミート複素対称スパース行列の連立 1 次方程式 (不完全 $\boldsymbol{LDL^T}$ 分解に基づく前処理付き共役直交共役残差法 (*COCR* 法), 対称行列用圧縮行格納法)
CALL DM_VLCSPSXCR1(ZSA,NZ,NCOL,NFRNZ,N,ZB, ISW, ZX,IPAR,RPAR, ZVW,ICON)

(1) 機能

$n \times n$ の非エルミート複素対称なスパース行列を係数行列とする連立 1 次方程式を, 不完全 $\boldsymbol{LDL^T}$ 分解に基づく前処理付きの共役直交共役残差法(*COCR* 法)で解きます.

$$\boldsymbol{Ax = b}$$

$n \times n$ の複素係数行列 $\boldsymbol{A}$ は, 対称行列用の圧縮行格納法で格納します.
定数ベクトル $\boldsymbol{b}$ および解ベクトル $\boldsymbol{x}$ は n 次元ベクトルです.

(2) パラメタ

ZSA.....入力. 係数行列の非零要素を格納します.
ZSA(1:NZ)に係数行列を格納します. ZSA(NZ)なる 1 次元複素配列.
対称行列用の圧縮行格納法については, 図 DM_VCLSPSXCR1-1 を参照してください.

NZ.....入力. 係数行列 $\boldsymbol{A}$ の非零要素の総数. ($1 \le$ NZ)

NCOL.....入力. 圧縮行格納法で使用される列指標で, 配列 ZSA に格納される要素が何番目の列ベクトルに属するかを示します.
NCOL(NZ)なる 1 次元配列.

NFRNZ .....入力. 行列 $\boldsymbol{A}$ の上三角行列部分の各行の非零要素を, 行方向に圧縮して順次配列 ZSA に格納するとき, 対応する行の最初の非零要素が格納される位置を表します.
NFRNZ(N+1)=NZ+1. NFRNZ(N+1)なる 1 次元配列.

N.....入力. 行列 $\boldsymbol{A}$ の次数 n. ($1 \le$ N)

ZB .....入力. 連立 1 次方程式の右辺の定数ベクトル $\boldsymbol{b}$ を格納します.
ZB(N)なる 1 次元複素配列.

ISW .....入力. 方程式を解く方法を指定します(ISW=1 又は ISW=3).
ISW=1: 初回目に方程式を解く場合に選択します.
ISW=3: 直前に解いた方程式と同一の行列 $\boldsymbol{A}$ を持ち, ベクトル $\boldsymbol{b}$ の要素が異なる方程式を解く場合に選択します.
ISW=3 で本ルーチンを呼び出す場合, 新たに設定するパラメタ ZB および初期値 ZX など変更のあるパラメタだけを修正し, それ以外のパラメタの内容は, 直前に呼び出した際の内容を変更せずに呼び出します.

ZX .....入力. 解ベクトル $\boldsymbol{x}$ の初期値を格納します.
出力. 解ベクトル $\boldsymbol{x}$ が格納されます.
ZX(N)なる 1 次元複素配列.

IPAR .....動作調整用パラメタ(整数)を指定します. 演算結果の指標値が設定される場合もあります. 何れのパラメタも 0 を指定すれば, 内部でデフォルト値を設定します. デフォルト値で収束しない場合は, パラメタ値を変えて試みることを奨めます. IPAR(20)なる 1 次元配列.

IPAR(1:5): 機能拡張用のリザーブ域で, 0 を指定しておきます.

IPAR(6): 入力. *COCR* 法の反復回数の上限を指定します($0 \le$ IPAR(6)). デフォルト値は 2000 です.

IPAR(7): 出力. 実際の反復回数が設定されます.

IPAR(8): 出力. *COCR* 法の反復で, 係数行列 $\boldsymbol{A}$ と反復過程のベクトル $\boldsymbol{v}$ の積 $\boldsymbol{Av}$ の評価回数が設定されます.

IPAR(9:10): 機能拡張用のリザーブ域で, 0 を指定しておきます.

IPAR(11): 入力. 前処理のための不完全 $\boldsymbol{LDL^T}$ 分解で, 新たな非零要素は破棄しますが, その補償を対角要素に反映するか否かを指定します. IPAR(11)= 0 を指定すると補償しません. IPAR(11)=1 を指定すると補償します. デフォルト値は 0 です. ((3)使用上の注意 a. 注意①参照).

IPAR(12): 出力. 不完全 $\boldsymbol{LDL^T}$ 分解で破棄した非零の要素数が設定されます.

IPAR(13:20): 機能拡張用のリザーブ域で, 0 を指定しておきます.

RPAR .....動作調整用パラメタ(実数)を指定します. 演算後, 結果の指標値が設定される場合もあります. 何れのパラメタも 0.0 を指定すれば, 内部でデフォルト値を設定します. デフォルト値で収束しない場合は, パラメタ値を変えて試みることを奨めます. RPAR(20)なる 1 次元配列.

RPAR(1): 機能拡張用のリザーブ域で, 0.0 を指定しておきます.

RPAR(2): 入力. *COCR* 法の反復で, 解の収束判定値 *epst* を指定します ($0.0 \le epst$). デフォルト値は 10^{-8} です.

RPAR(3): 出力. 収束判定条件を満たす解の, 残差ベクトルに対する相対残差ノルムが設定されます.

RPAR(4): 出力. 不完全 $\boldsymbol{LDL^T}$ 分解で破棄した非零要素の累積値 *zdrp* の実部が設定されます. ((3)使用上の注意 a. 注意①参照).

RPAR(5): 出力. 不完全 $\boldsymbol{LDL^T}$ 分解で破棄した非零要素の累積値 *zdrp* の虚部が設定されます. ((3)使用上の注意 a. 注意①参照).

RPAR(6:20): 機能拡張用のリザーブ域で, 0.0 を指定しておきます.

ZVW .....作業領域. 大きさ NZ の 1 次元複素配列.

ICON.....出力. コンディションコード.
表 DM_VLCSPSXCR1-1 参照.

表 DM_VLCSPSXCR1-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし.	ー
20000	指定した反復回数では, 収束条件を満たさなかった.	配列 ZX にはその時までに得られた近似値が格納される. 対応する相対残差ノルムも出力される.
29000	行列 ***A*** が非正則であった.	処理を打ち切る.
30000	パラメタに誤りがあった. N<1, NZ<1, NZ≠NFRNZ(N+1)-1, ISW<1, ISW=2,ISW>3, IPAR(6)<0, IPAR(11)<0,IPAR(11)>1, RPAR(2)<0.0.	処理を打ち切る.

$$\boldsymbol{A} = \begin{bmatrix} 1 & 2 & 3 & 0 \\ 2 & 5 & 0 & 6 \\ 3 & 0 & 8 & 9 \\ 0 & 6 & 9 & 11 \end{bmatrix}$$

$$\downarrow$$

$$\mathrm{NFRNZ} = \begin{bmatrix} 1 \\ 4 \\ 6 \\ 8 \\ 9 \end{bmatrix}, \quad \mathrm{ZSA} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 5 \\ 6 \\ 8 \\ 9 \\ 11 \end{bmatrix}, \quad \mathrm{NCOL} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 2 \\ 4 \\ 3 \\ 4 \\ 4 \end{bmatrix}$$

図 DM_VLCSPSXCR1-1 行列 A の対称行列用の圧縮行格納法

(3)　使用上の注意

a.　注意

①　新非零要素の破棄と補償について

本サブルーチンでは, 不完全 $\boldsymbol{LDL^T}$ 分解の過程で新たに発生した非零要素は通常破棄します. 破棄した要素の影響を緩和するために, IPAR(11)によってその補償を制御することができます. IPAR(11)=1 と指定すると, 破棄した要素分を当該行の対角要素に反映し補償します. この補償によって, 前処理行列としての特性が向上する場合があります.

更に本サブルーチンでは, IPAR(11)指定の内容に関わらず, 破棄した要素の累積値 *zdrp* を指標として設定します. RPAR(4)と RPAR(5)には, 累積値の実部と虚部の値がそれぞれ設定されます.

b.　使用例

複素対称行列を読み込み, 本サブルーチンで連立 1 次方程式の解を求めます. (並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C=====================================================
C  TEST PROGRAM FOR KRYLOV ITERATION METHODS
C  FOR SPARSE LINEAR EQUATIONS
C  WITH NON-HERMIT COMPLEX SYMMETRIC MATRIX.
C=====================================================
```

```
      PARAMETER (NZMAX=500 000, NMAX=10 000)
      IMPLICIT   REAL*8 (A-H,O-Y)
      IMPLICIT   COMPLEX*16 (Z)
      REAL*8     CNORM2
      DIMENSION  ZSA(NZMAX),NFRNZ(NMAX+1),
     1           NCOL(NZMAX),ZX(NMAX),ZB(NMAX),
     2           ZSAT(NZMAX),NFRNZT(NMAX+1),
     3           NCOLT(NZMAX),ZXT(NMAX),ZBT(NMAX),
     4           ZVW(NZMAX)
     5          ,IPAR(20),RPAR(20)
      CHARACTER  TITLE*72
C-----------------------------------------------------
C     INPUT MATRIX FROM UF SPARSE MATRIX COLLECTION
C-----------------------------------------------------
      CALL CREADMAT(TITLE,ZSAT,N,NFRNZT,NCOLT,ZSA)
      CALL CVECGEN(ZSAT,N,NFRNZT,NCOLT,ZXT,ZBT)
      CALL CMATCOPY(ZSAT,N,NFRNZT,NCOLT,ZXT,ZBT,
     +              ZSA,NFRNZ,NCOL,ZX,ZB)
C
      WRITE(6,600) TITLE
  600 FORMAT(
     */'-------------------------------------------'
     */'TEST MATRIX : '/A36/A36)
C-----------------------------------------------------
      ISW=1
      DO II=1,20
         IPAR(II)=0
         RPAR(II)=0.0D0
      END DO
      NZ=NFRNZ(N+1)-1
      CALL DM_VLCSPSXCR1(ZSA,NZ,NCOL,NFRNZ,N,ZB,
     *                   ISW,ZX,IPAR,RPAR,ZVW,ICON)
C
      IC    =IPAR(7)
      ICMAV =IPAR(8)
      MDRP  =IPAR(11)
      NZDRP =IPAR(12)
      EPST  =RPAR(2)
      RELRES=RPAR(3)
      DRPR  =RPAR(4)
      DRPI  =RPAR(5)
      REL   =CNORM(ZB,N)
```

```
      CALL CMSVCR1(ZSA,N,NFRNZ,NCOL,ZX,ZB,0)
      RELERR=CNORM(ZB,N)/REL
C
      WRITE(6,601)
  601 FORMAT(
     */'-----------------------------------------'
     */' SOLUTION RESULTS BY "DM_VLCSPSXCR1"')
      WRITE(6,605) N,NFRNZ(N+1)-1,MDRP
      WRITE(6,606) ICON,IC,ICMAV,NZDRP,DRPR,
     *             DRPI,EPST,RELRES,RELERR
  605 FORMAT(/' N           =',I12
     *       /' NZ          =',I12
     *       /' MDRP        =',I12)
  606 FORMAT(/' ICON        =',I12
     *       /' IC          =',I12
     *       /' ICMAV       =',I12
     *       /' NZDRP       =',I12
     *       /' DRPR        =',D12.2
     *       /' DRPI        =',D12.2
     *       /' EPST        =',D12.2
     *       /' RELRES      =',D12.2
     *       /' RELERR      =',D12.2
     */'-----------------------------------------')
      IF(RELERR.LE.EPST*1.1D0.AND.ICON.EQ.0)THEN
         WRITE(*,*)' ********** OK **********'
      ELSE
         WRITE(*,*)' ********** NG **********'
      ENDIF
      STOP
      END
C=======================================================
C     READ TEST MATRIX FOR COMPLEX SYMMETRIC MATRIX.
C=======================================================
      SUBROUTINE CREADMAT(TITLE,A,NCOL,IS,JS,W)
C
C THIS ROUTINE READS MATRIX DATA OF RB SPARSE FORM.
C THE FOLLOWING SAMPLE CODE IS ORIGINATED FROM MATRIX
C MARKET;
C
      IMPLICIT NONE
      CHARACTER TITLE*72,KEY*8,MXTYPE*3,RHSTYP*3,
     1         PTRFMT*16,INDFMT*16,VALFMT*20,RHSFMT*20
```

```
      INTEGER  TOTCRD,PTRCRD,INDCRD,VALCRD,RHSCRD,
     1         NROW,NCOL,NNZERO,NELTVL,
     2         NRHS,NRHSIX
      INTEGER  IS(*),JS(*),I
      REAL*8   A(*),W(*)
      INTEGER  IX
C     ------------------------
C     READ IN HEADER BLOCK
C     ------------------------
      READ(5,1000) TITLE,KEY,TOTCRD,PTRCRD,INDCRD,
     +VALCRD,RHSCRD,MXTYPE,NROW,NCOL,NNZERO,NELTVL,
     +PTRFMT,INDFMT,VALFMT,RHSFMT
 1000 FORMAT(A72,A8/5I14/A3,11X,4I14/2A16,2A20)
C
      IF(RHSCRD.GT.0) READ(5,1001) RHSTYP,NRHS,NRHSIX
 1001 FORMAT(A3,11X,2I14)
C     ------------------------
C     READ MATRIX STRUCTURE
C     ------------------------
      READ(5,PTRFMT) (IS(I),I=1,NCOL+1)
      READ(5,INDFMT) (JS(I),I=1,NNZERO)
C
      IF(VALCRD.GT.0) THEN
C     ----------------------
C     READ MATRIX VALUES
C     ----------------------
      IF(MXTYPE(1:1).EQ.'R') THEN
         READ(5,VALFMT) (A(I),I=1,NNZERO)
      ELSE
         READ(5,VALFMT) (A(I),I=1,2*NNZERO)
      END IF
      END IF
      RETURN
      END
C=======================================================
C     COPY COMPLEX MATRIX AND VECTORS.
C=======================================================
      SUBROUTINE CMATCOPY(ZSAT,N,NFRNZT,NCOLT,
     +                ZXT,ZBT,ZSA,NFRNZ,NCOL,ZX,ZB)
      IMPLICIT   REAL*8 (A-H,O-Y)
      IMPLICIT   COMPLEX*16 (Z)
      DIMENSION  ZSAT(*),NFRNZT(*),NCOLT(*),
```

```
     +           ZXT(*),ZBT(*),ZSA(*),NFRNZ(*),
     +           NCOL(*),ZX(*),ZB(*)
C
      NZ=NFRNZT(N+1)-1
      DO I=1,N+1
         NFRNZ(I)=NFRNZT(I)
      END DO
      DO I=1,NZ
         ZSA(I)=ZSAT(I)
         NCOL(I)=NCOLT(I)
      END DO
C
      DO I=1,N
         ZX(I)=ZXT(I)
         ZB(I)=ZBT(I)
      END DO
      RETURN
      END
C=====================================================
C     GENERATE COMPLEX B AND X VECTORS.
C=====================================================
      SUBROUTINE CVECGEN(ZSAT,N,NFRNZT,NCOLT,ZXT,ZBT)
      IMPLICIT   REAL*8 (A-H,O-Y)
      IMPLICIT   COMPLEX*16 (Z)
      DIMENSION  ZSAT(*),NFRNZT(*),NCOLT(*),
     +           ZXT(*),ZBT(*)
C
C     COMPUTE RIGHT HAND SIDE VECTOR B.
      DO II=1,N
         ZXT(II)=1.0D0+DFLOAT(II)/DFLOAT(N)
      END DO
      CALL CMSVCR1(ZSAT,N,NFRNZT,NCOLT,ZXT,ZBT,1)
C
C     SET INITIAL VALUE
      DO II=1,N
         ZXT(II)=0.0D0
      END DO
      RETURN
      END
C=====================================================
C     MATRIX VECTOR MULTIPLICATION.
C     COMPLEX SYMMETRIC MATRIX STORED IN CSR FORM.
```

```
C=====================================================
      SUBROUTINE CMSVCR1(ZSA,N,NFRNZ,NCOL,ZX,ZB,ISW)
      IMPLICIT   REAL*8 (A-H,O-Y)
      IMPLICIT   COMPLEX*16 (Z)
      DIMENSION  ZSA(*),ZB(*),ZX(*),NFRNZ(*),NCOL(*)
C
      IF(ISW.EQ.1) THEN !*** MULTIPLICATION (AX=>B)
C
         DO I=1,N
            ZB(I)=0.0D0
         END DO
         DO I=1,N
            K1=NFRNZ(I)
            K2=NFRNZ(I+1)-1
            IF(ZX(I).NE.0.0D0) THEN
               DO J=K1,K2
                  ZB(NCOL(J))=ZSA(J)*ZX(I)+ZB(NCOL(J))
                  IF(NCOL(J).NE.I)
     +               ZB(I)=ZSA(J)*ZX(NCOL(J))+ZB(I)
               END DO
            ELSE
               DO J=K1,K2
                  ZB(I)=ZSA(J)*ZX(NCOL(J))+ZB(I)
               END DO
            END IF
         END DO
C
      ELSE           !*** RESIDUAL VECTOR (B-AX=>B)
C
         DO I=1,N
            K1=NFRNZ(I)
            K2=NFRNZ(I+1)-1
            IF(ZX(I).NE.0.0D0) THEN
               DO J=K1,K2
                 ZB(NCOL(J))=-ZSA(J)*ZX(I)+ZB(NCOL(J))
                  IF(NCOL(J).NE.I)
     +               ZB(I)=-ZSA(J)*ZX(NCOL(J))+ZB(I)
               END DO
            ELSE
               DO J=K1,K2
                  ZB(I)=-ZSA(J)*ZX(NCOL(J))+ZB(I)
               END DO
```

```
            END IF
         END DO
      END IF
      RETURN
      END
C=====================================================
C     L2 NORM OF A COMPLEX VECTOR.
C=====================================================
      FUNCTION  CNORM(ZX,N)
      IMPLICIT  REAL*8 (A-H,O-Y)
      IMPLICIT  COMPLEX*16 (Z)
      DIMENSION ZX(N)
      CNORM=0.0D0
      DO I=1,N
         CNORM=ZX(I)*DCONJG(ZX(I))+CNORM
      END DO
      IF(CNORM.NE.0.0D0) CNORM=DSQRT(CNORM)
      RETURN
      END
```

(4)　手法概要

本サブルーチンでは, 非エルミート複素対称スパース行列に対する連立 1 次方程式解法として, 不完全 $\boldsymbol{LDL^T}$ 分解に基づく前処理付きの共役直交共役残差法 (*COCR* 法) を用いています.

a.　不完全 $\boldsymbol{LDL^T}$ 分解

前処理法は反復法の反復回数を縮小する方法であり, その手法として不完全分解法が著名です. この不完全分解法は, アルゴリズムが簡素で条件の良い行列では極めて効率的な特性を持ちます.

本サブルーチンでは, 新たな非零要素の発生を抑止した不完全 $\boldsymbol{LDL^T}$ 分解に基づく前処理技法を採用しています.

b.　共役直交共役残差法 (*COCR* 法)

非エルミート複素対称行列に対する解法は, ランチョスプロセスに基づく *BiCG* 法や *CGS* 法, *BiCGSTAB* などの積型解法, アーノルディプロセスに基づく *GMRES* 法などが一般的な解法です. しかしこれらは行列の対称性を活かせず, 反復法の中核となる行列ベクトル積が反復あたり 2 回となります.

本サブルーチンでは, 行列の対称性を活かし, 残差ノルム最小化の特徴と 安定的な収束特性を持つ共役直交共役残差法 (*COCR* 法)を採用しています.

c. 前処理付き *COCR* 法のアルゴリズム

$$
\begin{aligned}
&\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0, \boldsymbol{z}_0 = \boldsymbol{M}^{-1}\boldsymbol{r}_0, \beta_{-1} = 0,\\
&\textit{for } k = 0,1,\ldots \textit{until } \|\boldsymbol{r}_k\|_2 \le \varepsilon\|\boldsymbol{b}\|_2 \textit{ do};\\
&\quad \boldsymbol{p}_k = \boldsymbol{z}_k + \beta_{k-1}\boldsymbol{p}_{k-1},\\
&\quad \boldsymbol{A}\boldsymbol{p}_k = \boldsymbol{A}\boldsymbol{z}_k + \beta_{k-1}\boldsymbol{A}\boldsymbol{p}_{k-1},\\
&\quad \alpha_k = (\bar{\boldsymbol{z}}_k, \boldsymbol{A}\boldsymbol{z}_k)/(\boldsymbol{M}^{-1}\boldsymbol{A}\bar{\boldsymbol{p}}_k, \boldsymbol{A}\boldsymbol{p}_k),\\
&\quad \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k\boldsymbol{p}_k, \boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k\boldsymbol{A}\boldsymbol{p}_k,\\
&\quad \boldsymbol{z}_{k+1} = \boldsymbol{M}^{-1}\boldsymbol{r}_{k+1}, \beta_k = (\bar{\boldsymbol{z}}_{k+1}, \boldsymbol{A}\boldsymbol{z}_{k+1})/(\bar{\boldsymbol{z}}_k, \boldsymbol{A}\boldsymbol{z}_k)\\
&\textit{end}
\end{aligned}
$$

ここで複素ベクトル ***x*** , ***y*** の内積は, 複素共役に基づき

$$(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{n} \bar{x}_i y_i$$

と定義します.

d. 収束判定法

反復解法としての収束判定として, 以下の条件を満たした時に, 解に収束したと見なします.

$$\|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k\|_2 \le eps\|\boldsymbol{b}\|_2$$

ここで *epst* は収束判定値で, パラメタ RPAR(2)に指定します. デフォルト値は 10^{-8} です. パラメタ RPAR(3)には, 残差ベクトルの相対ノルム

$$\|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k\|_2 / \|\boldsymbol{b}\|_2$$

が設定されます. 収束判定を満たさない場合でも, 本サブルーチンでの終了までに得られた解に対するノルムが設定されます.

この残差ベクトル $\boldsymbol{b}$ - $\boldsymbol{A}\boldsymbol{x}_k$ は, 反復公式に於ける漸化式に基づく公式で計算されます.

アルゴリズムの詳細については, “付録 1 参考文献一覧表” の[66], [74]を参照して下さい.

DM_VLCX

複素行列の連立 1 次方程式 (ブロック化された LU 分解法)
CALL DM_VLCX(ZA, K, N, ZB, EPSZ, ISW, IS, IP, ICON)

(1)　機能

複素数係数連立 1 次方程式

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

をブロック化した外積形の LU 分解法で解きます. ただし $\boldsymbol{A}$ は $n \times n$ の正則な複素行列, $\boldsymbol{b}$ は n 次元の複素定数ベクトル, $\boldsymbol{x}$ は n 次元の解ベクトルです. ($n \geq 1$)

(2)　パラメタ

ZA .....入力. 行列 $\boldsymbol{A}$ を ZA(1:N, 1:N)に格納します.
出力. 行列 $\boldsymbol{L}$ と行列 $\boldsymbol{U}$ が ZA(1:N, 1:N)に格納されます.
ZA(K, N)なる倍精度複素数型の 2 次元配列.

K.....入力. 格納配列 ZA の 1 次元目の大きさ(≥ N).

N.....入力. 行列 $\boldsymbol{A}$ の次数 n.

ZB .....入力. 定数ベクトル $\boldsymbol{b}$.
出力. 解ベクトル $\boldsymbol{x}$.
ZB(N)なる倍精度複素数型の 1 次元配列.

EPSZ .....入力. ピボットの相対零判定値(≥ 0.0)
0.0 のときは標準値が採用されます.
((3)使用上の注意 a. 注意①参照).

ISW .....入力. 制御情報.
同一の係数行列をもつ k (≥ 1)組の方程式を解くとき, 次のように指定します.
ISW=1 のとき 1 組目の方程式を解きます.
ISW=2 のとき 2 組目以降の方程式を解きます. ただし, このとき ZB の値だけを新しい定数ベクトルの値に変え, それ以外のパラメタはそのままで使います.
((3)使用上の注意 a. 注意②参照).

IS.....出力. 行列 $\boldsymbol{A}$ の行列式を求めるための情報, 演算後の配列 ZA の N 個の対角要素と IS の値を掛け合わせると行列式が得られます.
((3)使用上の注意 a. 注意②参照).

IP.....出力. 部分ピボッティングによる行の入換えの履歴を示すトランスポジションベクトル. 大きさ n の 1 次元配列.

ICON.....出力. コンディションコード.
表 DM_VLCX-1 参照

表 DM_VLCX-1 コンディションコード

コード	意　　味	処理内容
0	エラーなし.	ー
20000	行列 ***A*** のある行の要素がすべて零であったか, またはピボットが相対的に零となった. 行列 ***A*** は非正則の可能性が強い.	処理を打ち切る.
30000	K<N, N<1 である. または EPSZ<0.0 である.	

(3)　使用上の注意

a.　注意

①　EPSZ に値を設定した場合, ピボットが, EPSZ 以下なら零とみなし, ICON=20000 で処理を打ち切ります. EPSZ の標準値は丸め誤差単位を u としたとき, EPSZ=16u です. なおピボットが小さくなっても計算を続行させる場合には, EPSZ へ極小値を与えればよいのですが, その結果は保証されません.

②　同一の係数行列を持つ連立 1 次方程式をいくつか続けて解く場合には, 2 回目以降 ISW=2 として解くと, 係数行列 ***A*** の LU 分解過程を省略するので計算時間が少なくなります. なおこの場合 IS の値は, ISW=1 のときの値が保証されます.

b.　使用例

複素行列の連立 1 次方程式を解きます.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (N=2000,K=N+1)
C
      COMPLEX*16 A(K,N),B(N)
      REAL*8     C
      INTEGER    IP(N),IS
C
      C=SQRT(1.0D0/DBLE(1+N))
      T=DATAN(1.0D0)*8./(1+N)
C
      DO 100 J=1,N
      DO 100 I=1,N
      A(I,J)=DCMPLX(C*COS(T*I*J),C*SIN(T*I*J))
 100  CONTINUE
C
      DO 200 I=1,N
```

```
      S=(0.,0.)
      DO 200 J=1,N
      S=S+DCMPLX(COS(T*I*J),SIN(T*I*J))
      B(I)=S*C
 200  CONTINUE
C
      ISW=1
      EPSZ=0.0D0
      CALL DM_VLCX(A,K,N,B,EPSZ,ISW,IS,IP,ICON)
      PRINT*,'ICON=',ICON

      ERROR=0.0D0
      DO I=1,N
      ERROR=MAX(ERROR,ABS(1.0D0-B(I)))
      ENDDO
      PRINT*,'ERROR =',ERROR

      PRINT*,'ORDER=',N,' B(1)=',B(1),'B(N)=',B(N)
      STOP
      END
```

DM_VLDLX

LDL^T分解された正値対称行列の連立 1 次方程式
CALL DM_VLDLX(B, FA, KFA, N, ICON)

(1) 機能

LDL^T分解された正値対称な係数行列を持つ連立 1 次方程式,

$$\boldsymbol{LDL}^T\boldsymbol{x} = \boldsymbol{b} \tag{1.1}$$

を解きます. ただし, $\boldsymbol{L}$, $\boldsymbol{D}$ はそれぞれ $n \times n$ の単位下三角行列と対角行列, $\boldsymbol{b}$ は n 次元の実定数ベクトル, $\boldsymbol{x}$ は n 次元の解ベクトルとします. また, $n \geq 1$ とします.

本サブルーチンは, サブルーチン DM_VSLDL により, LDL^T分解された行列を受けて, 求解処理を行います.

(2) パラメタ

B.....入力. 定数ベクトル $\boldsymbol{b}$.
出力. 解ベクトル $\boldsymbol{x}$.
B(N)なる倍精度実数型 1 次元配列.

FA .....入力. LDL^T分解された行列 $\boldsymbol{L}$, $\boldsymbol{D}^{-1}$ および $\boldsymbol{L}^T$ を格納します.
FA(1:N, 1:N)の下三角部分$\{A(i, j) \mid i \geq j\}$ に LDL^T分解された行列 $\boldsymbol{L}$ と $\boldsymbol{D}^{-1}$ を次のように格納します. つまり, FA(i, j)に, 次のように格納します.
l_{ij} $i > j$ のとき,
d_{ii} の逆数 $i = j$ のとき,
(図 DM_VLDLX-1 参照).
FA(KFA, N)なる倍精度実数型 2 次元配列.

KFA .....入力. 配列 FA の整合寸法(≥ N).

N.....入力. 行列 $\boldsymbol{L}$ および $\boldsymbol{D}$ の次数 n.

ICON.....出力. コンディションコード.
表 DM_VLDLX-1 参照

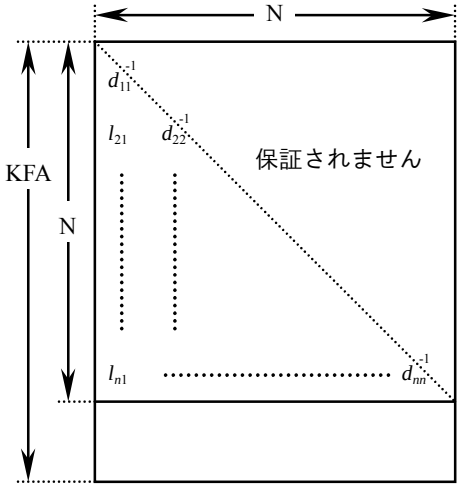


図 DM_VLDLX-1　配列 FA への行列 $\boldsymbol{L}$, $\boldsymbol{D}^{-1}$ を格納する方法

LDL^T 分解された結果は, 行列 $\boldsymbol{D}^{-1}$ を対角部分に, $\boldsymbol{L}$ の対角要素を除いた部分を下三角部分にそれぞれ格納します.

表 DM_VLDLX-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし.	ー
10000	係数行列が正値でなかった.	処理は続行する.
30000	N<1, KFA<N	処理を打ち切る.

(3)　使用上の注意

a.　注意

①　本サブルーチンを, サブルーチン DM_VSLDL に続けて呼び出すことにより, 正値対称な係数行列を持つ連立 1 次方程式を解くことができます. しかし, 通常は, サブルーチン DM_VLSX を呼び出せば, 一度に解が求められます.

b.　使用例

4000 × 4000 の行列を, LDL^T 分解して連立 1 次方程式を解きます.
(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER (N=4000,KFA=N+1)
      REAL*8    A(KFA,N)
```

```
      REAL*8    B(N)
C
!$OMP PARALLEL DEFAULT(PRIVATE) SHARED(A,B)
!$OMP DO
      DO J=1,N
      DO I=J,N
      A(I,J)=MIN(I,J)
      ENDDO
      ENDDO
!$OMP END DO

!$OMP DO
      DO I=1,N
      B(I)=I*(I+1)/2+I*(N-I)
      ENDDO
!$OMP END DO

!$OMP END PARALLEL
C
      CALL DM_VSLDL(A,KFA,N,1.D-13,ICON)
      WRITE(6,610) ICON
      IF(ICON.GE.20000) GO TO 10

      CALL DM_VLDLX(B,A,KFA,N,ICON)
      WRITE(6,630) (B(I),I=1,10)

      S=1.0D0
!$OMP PARALLEL DO DEFAULT(PRIVATE) SHARED(A)
!$OMP+          REDUCTION(*:S)
      DO I=1,N
      S=S*A(I,I)
      ENDDO
!$OMP END PARALLEL DO

      DET=S
      DET=1.D0/DET
      WRITE(6,620) DET
      GO TO 10
  500 FORMAT(I5)
  510 FORMAT(3D22.15)
  600 FORMAT(1H,I5/(10X,3D23.16))
  610 FORMAT(/10X,5HICON=,I5)
```

```
620 FORMAT(/10X
   *,34HDETERMINANT OF COEFFICIENT MATRIX=
   *,D23.16)
630 FORMAT(/10X,15HSOLUTION VECTOR
   *//(10X,3D23.16))
640 FORMAT(/10X,12HINPUT MATRIX)
10  STOP
    END
```

(4)　手法概要

LDL^{T}分解された係数行列の連立 1 次方程式を, 前進代入および後退代入で解きます.

(“付録 1 参考文献一覧表” の[54]を参照)

DM_VLSPAXCR2

実非対称スパース行列の連立 1 次方程式 (近似逆行列に基づく前処理付きの演繹的次元縮小法 (IDR 法), 圧縮行格納法)
CALL DM_VLSPAXCR2 (A,NZ,NCOL,NFRNZ,N,B,ISW,X,AM,NZM,NCOLM,NFRNZM ,NWM,IPAR,RPAR,VW1,IVW1,VW2,IVW2,LMMAX,LNMAX,NUMT,ICON)

(1) 機能

$n \times n$ の実非対称なスパース行列を係数行列とする連立 1 次方程式を, 近似逆行列に基づく前処理付きの演繹的次元縮小法 (*IDRstab(s,l)*法) で解きます.

$$\boldsymbol{Ax} = \boldsymbol{b}$$

$n \times n$ の実係数行列 $\boldsymbol{A}$ は, 圧縮行格納法で格納します.
定数ベクトル $\boldsymbol{b}$ および解ベクトル $\boldsymbol{x}$ は n 次元ベクトルです.
また, s はシャドウ残差の次数, l は加速多項式の次数です.

(2) パラメタ

A.....入力. 係数行列の非零要素を格納します.
A(1:NZ)に係数行列を格納します. A(NZ)なる 1 次元配列.
圧縮行格納法は, 行列 $\boldsymbol{A}$ の転置行列を圧縮列格納法で格納したもの. 圧縮列格納法については, 図 DM_VMVSCC-1 を参照してください.

NZ .....入力. 係数行列 $\boldsymbol{A}$ の非零要素の総数. ($1 \le$ NZ)

NCOL.....入力. 圧縮行格納法で使用される列指標で, A に格納される要素が何番目の列ベクトルに属するかを示します.
NCOL(NZ)なる 1 次元配列.

NFRNZ .....入力. 行列 $\boldsymbol{A}$ の各行の非零要素を行方向に圧縮して順次配列 A に格納するとき, 対応する行の最初の非零要素が格納される位置を表します.
NFRNZ(N+1)=NZ+1. NFRNZ(N+1)なる 1 次元配列.

N.....入力. 行列 $\boldsymbol{A}$ の次数 n. ($1 \le$ N)

B.....入力. 連立 1 次方程式の右辺の定数ベクトル $\boldsymbol{b}$ を格納します.
B(N)なる 1 次元配列.

ISW .....入力. 方程式を解く方法を指定します. ($1 \le$ ISW ≤ 3)
ISW=1: 初回目に方程式を解く場合に選択します.
ISW=2: 直前に解いた方程式と同一のスパース構造を持ち, 行列 $\boldsymbol{A}$ とベクトル $\boldsymbol{b}$ の要素が異なる方程式を解く場合に選択します.
ISW=3: 直前に解いた方程式と同一の行列 $\boldsymbol{A}$ を持ち, ベクトル $\boldsymbol{b}$の 要 素だけが異なる方程式を解く場合に選択します.
ISW=2 又は 3 で本ルーチンを呼び出す場合, 新たに設定するパラメタ A, B および初期値 X など変更のあるパラメタだけを修正し, それ以外のパラメタの内容は, 直前に呼び出した際の内容を変更せずに呼び出します.

X.....入力. 解ベクトル $\boldsymbol{x}$ の初期値を格納します.
出力. 解ベクトル $\boldsymbol{x}$ が格納されます.
X(N)なる 1 次元配列.

AM.....入力. 初期近似逆行列 $\boldsymbol{M_0}$ を指定する場合, その非零要素を圧縮行格納法で AM(1:NZM) に格納します. AM(NWM)なる 1 次元配列.
圧縮行格納法は, 行列 $\boldsymbol{A}$ の格納法と同じです.
出力. 計算された近似逆行列が格納されます.

NZM.....入力. 初期近似逆行列 $\boldsymbol{M_0}$ の非零要素の総数. ($0 \le$ NZM)
初期近似逆行列を指定しない場合は, NZM=0 と指定します. この場合は初期近似逆行列として単位行列が内部で採用されます.
出力. 計算された近似逆行列 $\boldsymbol{M}$ の非零要素の総数が格納されます.

NCOLM.....入力. 初期近似逆行列 $\boldsymbol{M_0}$ を指定する場合, 圧縮行格納法の列指標で, AM に格納される要素が何番目の列ベクトルに属するかを示します.
NCOLM(NWM)なる 1 次元配列.
出力. 計算された近似逆行列 $\boldsymbol{M}$ の, 圧縮行格納法の列指標.

NFRNZM.....入力. 初期近似逆行列 $\boldsymbol{M_0}$ を指定する場合, 各行の非零要素を行方向に圧縮して順次配列 AM に格納するとき, 対応する行の最初の非零要素が格納される位置を表します. NFRNZM(N+1)=NZM+1. NFRNZM(N+1)なる 1 次元配列.
出力. 計算された近似逆行列 $\boldsymbol{M}$ の, 各行の最初の非零要素が格納される位置.

NWM.....入力. 近似逆行列を求める領域の最大サイズを指定します. (N $\le$ NWM)
行列 $\boldsymbol{A}$ の第 k 列の非零要素数を nz_k として, 近似逆行列のサイズ nzm を下記の式で算定します.

$$nzm = \sum_{k=1}^{n} \max(1, nz_k \times IPAR(2) / 100)$$

この値より, NWM を下記の式に従って指定します.

$$NWM = \max(nzm , nz)$$

((3)使用上の注意 a. 注意①参照)

IPAR.....動作調整用パラメタ(整数)を指定します. 演算結果の指標値が設定される場合もあります. 何れのパラメタも 0 を指定すれば, 内部でデフォルト値を設定します. デフォルト値で収束しない場合は, パラメタ値を変えて試みることを奨めます. IPAR(20)なる 1 次元配列.

IPAR(1): 機能拡張用のリザーブ域で, 0 を指定しておきます.

IPAR(2): 入力. 近似逆行列の非零要素数の総数の上限として, 行列 $\boldsymbol{A}$ の非零要素数の総数 NZ に対する比率(%)を指定します. ($0 \le$ IPAR(2))
例えば IPAR(2)=50 と指定した場合, 行列 $\boldsymbol{A}$ の非零要素数のおおよそ 50%の非零要素数を上限に持つ近似逆行列が生成されます.
デフォルト値は 100 です. ((3)使用上の注意 a. 注意③参照)

IPAR(3): 入力. 近似逆行列の各列ベクトルの計算で, 順次追加する新たなインデックスの増量を指定します. ($0 \le$ IPAR(3) $\le n$)
例えば IPAR(3)=2 と指定した場合, 生成する近似逆行列の 非 零要素数を, 効果の高い順番に 2 個づつ順次増加させます. デフォルト値は 1 です. ((3)使用上の注意 a. 注意④参照)

IPAR(4): 入力. 演繹的次元縮小法 *IDRstab(s,l)* 法のシャドウ残差の次数 s を指定します. $(0 \le s \le n)$ デフォルト値は 4 です.

IPAR(5): 入力. 演繹的次元縮小法 *IDRstab(s,l)* 法の加速多項式の次数 l を指定します. $(0 \le l \le n)$ デフォルト値は 1 です.

IPAR(6): 入力. *IDRstab(s,l)* 法の反復回数の上限を指定します.
$(0 \le \text{IPAR(6)})$ デフォルト値は 2000 です.

IPAR(7): 出力. 実際の反復回数が設定されます.

IPAR(8): 出力. *IDRstab(s,l)* 法の反復で, 係数行列 $\boldsymbol{A}$ と反復過程のベクトル $\boldsymbol{v}$ の積 $\boldsymbol{Av}$ の評価回数が設定されます.

IPAR(9): 出力. 近似逆行列の領域 AM, NCOLM のサイズ NWM の予測値が設定されます. ((3)使用上の注意 a. 注意①参照)

IPAR(10:12):機能拡張用のリザーブ域で, 0 を指定しておきます.

IPAR(13): 出力. 作業用配列 VW2, IVW2 で 実際に使用した LMMAX の値.

IPAR(14): 出力. 作業用配列 VW2 で 実際に使用した LNMAX の値.

IPAR(15:20):機能拡張用のリザーブ域で, 0 を指定しておきます.

RPAR.....動作調整用パラメタ(実数)を指定します. 演算後, 結果の指標値が設定される場合もあります. 何れのパラメタも 0.0 を指定すれば, 内部でデフォルト値を設定します. デフォルト値で収束しない場合は, パラメタ値を変えて試みることを奨めます. RPAR(20)なる 1 次元配列.

RPAR(1): 入力. 近似逆行列の計算で, 列ベクトルを求める収束判定値 *eps* を指定します. $(0.0 \le eps)$ デフォルト値は 0.3 です.
((4) 手法概要 a. 近似逆行列　参照)

RPAR(2): 入力. *IDRstab(s,l)* 法の反復で, 解の収束判定値 *epst* を指定します. $(0.0 \le epst)$ デフォルト値は 10^{-8} です.
((4) 手法概要 c. 収束判定法　参照)

RPAR(3): 出力. 収束判定条件を満たす解の, 残差ベクトルに対する相対残差ノルムが設定されます.

RPAR(4:20): 機能拡張用のリザーブ域で, 0.0 を指定しておきます.

VW1.....作業領域. VW1(NWM)なる 1 次元配列.

IVW1 .....作業領域. IVW1(NWM)なる 1 次元配列.

VW2.....作業領域. VW2(LMMAX,LNMAX+3,NUMT)なる 3 次元配列.

IVW2 .....作業領域. IVW2(LMMAX,3,NUMT)なる 3 次元配列.

LMMAX .....入力. 作業用配列の第 1 添字の大きさ. $(1 \le \text{LMMAX})$
行列 A の非零要素に関連する値ですが, 特定の列に着目した時, その列に含まれる非零要素の数と, その非零要素に連動する他の列に含まれる非零要素の総数を定義します. LMMAX には, 列毎の総数の中で最大の値を指定します. 通常 1000 程度で充分ですが, 解が得られない場合は値を拡大し試みることを奨めます. ((3)使用上の注意 a. 注意⑤参照)

LNMAX.....入力. 作業用配列の第 2 添字に関連する大きさ. $(1 \le \text{LNMAX})$
近似逆行列 M の各列ベクトルに含まれる非零要素の数の中で, 最大の値に比例した値を指定します. 標準的な使用の場合は, 行列 $\boldsymbol{A}$ の各列ベクトルの

中で, 非零要素の数の最大値を目安に設定します. 解が得られない場合は, 値を拡大し試みることを奨めます. ((3)使用上の注意 a. 注意⑤参照)

NUMT.....入力. 作業用配列の第 3 添字の大きさ. (1 ≤ NUMT)
本サブルーチンで並列実行を行う最大のスレッド数を指定します.

ICON.....出力. コンディションコード.
表 DM_VLSPAXCR2-1 参照.

表 DM_VLSPAXCR2-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし.	ー
11000	行列 ***A*** に非正則の可能性があった.	処理を続行する.
19000	対角要素が無い行があった.	
20000	指定した反復回数では, 収束条件を満たさなかった.	処理を打ち切る. 配列 X にはその時までに得られた近似値が格納される. 対応する相対残差ノルムも出力される.
25000	NWM の値が小さく, 近似逆行列の領域 AM, NCOLM が不足した.	処理を打ち切る. IPAR(9)に必要となる最低の大きさの予測値 NWM が設定される.
26000	LMMAX の値が小さく, 作業領域 VW2, IVW2 が不足した.	処理を打ち切る.
27000	LNMAX の値が小さく, 作業領域 VW2 が不足した.	
29000	行列 ***A*** が非正則であった.	
30000	パラメタに誤りがあった. N<1, NZ<1, NZ≠NFRFZ(N+1)-1, ISW<1, ISW>3, NWM<N, NZM<0, IPAR(2)<0, IPAR(3)<0, IPAR(4)<0, *n*< IPAR(4), IPAR(5)<0, *n*< IPAR(5), IPAR(6)<0, LMMAX<1, LNMAX<1, NUMT<1, RPAR(1)<0.0, RPAR(2)<0.0	
30011	行列 ***A*** に関するパラメタに誤りがあった. 何れかのインデックスに以下の関係があった. NFRNZ(*k*)>NFRNZ(*k*+1), *k*=1,...,*n*	

30012	行列 ***A*** に関するパラメタに誤りがあった. 何れかのインデックスに以下の関係があった. NCOL(l)>NCOL(l+1), l=NFRNZ(k),...,NFRNZ(k+1), k=1,...,n	処理を打ち切る.
30021	初期近似逆行列 $\boldsymbol{M_0}$ に関するパラメタに誤りがあった. 何れかのインデックスに以下の関係があった. NFRNZM(k)>NFRNZM(k+1), k=1,...,n	
30022	初期近似逆行列 $\boldsymbol{M_0}$ に関するパラメタに誤りがあった. 何れかのインデックスに以下の関係があった. NCOLM(l)>NCOLM(l+1), l=NFRNZM(k),...,NFRNZM(k+1), k=1,...,n	

(3) 使用上の注意

a. 注意

① 近似逆行列に関する領域のサイズについて

近似逆行列 $\boldsymbol{M}$ のサイズ nzm は, 行列 $\boldsymbol{A}$ の第 k 列の非零要素数を nz_k とすると,

$$nzm = \sum_{k=1}^{n} \max(1, nz_k \times IPAR(2)/100)$$

で算定されます. 確保する領域のサイズ NWM は,

$$NWM = \max(nzm, nz)$$

となります. 近似逆行列の非零要素数の上限に関するパラメタ IPAR(2)を, 標準的な値として使用する場合(IPAR(2)=0)は, NWM=NZ と指定します. 上式による NWM の算定が困難な場合, 十分に大きな値 (NWM=2×NZ など) を設定して呼び出します. 本サブルーチンの呼び出しの結果, 予測されるサイズが IPAR(9)に設定されます. この値は, 類似した行列 $\boldsymbol{A}$ に関する方程式を解く際に, 有効な目安を与えます. スパース構造が同一で, 近似逆行列の非零要素の比率も同等の場合は, 後続する呼び出しで, IPAR(9)の値を NWM の目安とします. 一方, 行列 $\boldsymbol{A}$ の非零要素数が多くなる場合や, 近似逆行列 $\boldsymbol{M}$ の非零要素の比率を上げる場合は, IPAR(9)にそれぞれの増加の比率を掛けた量を NWM の目安とします.

② 初期近似逆行列の設定について

本サブルーチンを呼び出すにあたって, 前処理のための近似逆行列の初期行列 $\boldsymbol{M_0}$ を設定することができます. この場合, 初期行列の非零要素の総数を NZM に指定し, 初期行列を圧縮行格納法でパラメタ AM, NCOLM, NFRNZM にそれぞれ設定します. この初期近似行列の設定によって, 本サブルーチンを用いて以下の様な問題を解く場合に, 効率的に処理することができます.

1) スパース構造が同一で, 値が異なる係数行列 $\boldsymbol{A}$ や定数ベクトル b を持つ 複数組みの方程式を解く場合.

2) 類似したスパース構造を持つ複数組みの方程式を解く場合.

これらの処理は ISW と合わせて制御します. この処理では, A と関連するパラメタや B,X の値など変更のあるパラメタだけを修正し, パラメタ AM や作業用配列などその他のパラメタの内容は, 本ルーチンを直前に呼び出した結果を保持したままで呼び出します. この場合, 初期行列の非零要素の総数の上限 IPAR(2)を増加させることが可能です.

③ 近似逆行列 $\boldsymbol{M}$ の非零要素数について

本サブルーチンでは, 近似逆行列 M に基づく前処理付き連立 1 次方程式

$\boldsymbol{AMy = b, x = My}$

を解きます. ここで行列 $\boldsymbol{M}$ は, $\boldsymbol{AM} \fallingdotseq \boldsymbol{I}$ となる様に近似しますが, 行列 $\boldsymbol{M}$ の非零要素数は, 反復の過程で頻繁に実行する行列ベクトル積 $\boldsymbol{x = My}$ の効率とのトレードオフがあります. 本サブルーチンでは, パラメタ IPAR(2)によって, 生成する非零要素数を制御します. 一般的には, 行列 $\boldsymbol{A}$ の非零要素数 nz と同等程度であること, 即ち IPAR(2)=100 が推奨されます.

本サブルーチンでは, 近似逆行列 $\boldsymbol{M}$ を列ベクトル $\boldsymbol{m}_k$, $k=1,\ldots,n$ 毎に計算します. 行列 $\boldsymbol{A}$ の第 k 列の非零要素数を nz_k とすると, 近似逆行列の第 k 列の非零要素数の上限を nz_k×IPAR(2)/100 として求めます. その過程で, 収束条件

$\left\| \boldsymbol{Am}_k - \boldsymbol{e}_k \right\|_2 \le eps$

を満たした場合は, $\boldsymbol{m}_k$ の非零要素の数が上限 nz_k×IPAR(2)/100 に達していなくても, その時点で近似逆行列の該当する列ベクトル $\boldsymbol{m}_k$ の計算を終了します.

④ 近似逆行列 $\boldsymbol{M}$ の計算におけるインデックスの増分について

列ベクトル $\boldsymbol{m}_k$ の計算は, 以下の最小二乗問題を解くことで求めます.

$$\min_{m_k} \left\| \boldsymbol{Am}_k - \boldsymbol{e}_k \right\|_2 , k = 1,\ldots,n$$

ここで $\boldsymbol{e}_k$ は単位ベクトルです. この最小二乗問題の解に基づく残差ベクトルで, $\boldsymbol{m}_k$ に新たに発生する非零要素の中から, 残差ベクトルを最小化する最も貢献度が大きい要素を優先的に選択します. インデックスの増分は, 優先的に選択する個数を指定するものです. 列ベクトル $\boldsymbol{m}_k$ のスパース性を維持するために, 徐々に非零要素を増加させること, 即ち IPAR(3)=1 が推奨されます.

⑤ 作業領域 VW2, IVW2 について

作業用配列 VW2, IVW2 は, それぞれ 3 次元配列です. これらの配列は, 近似逆行列 $\boldsymbol{M}$ の列ベクトル $\boldsymbol{m}_k$ を計算する最小二乗問題を解くための領域として使用されます. 一般に $\boldsymbol{m}_k$ は, スパースベクトルですが, 近似逆行列を求める過程でそのスパース性が変化します. ④項の式に従って最小二乗問題を定義しますが, 残差ベクトル $\boldsymbol{A}\boldsymbol{m}_k - \boldsymbol{e}_k$ はスパースベクトル $\boldsymbol{m}_k$ の非零要素に連動した行列 $\boldsymbol{A}$ の列ベクトルだけが関与し, その非零要素だけからなる矩形の最小二乗問題が定義されます. この矩形の方程式のサイズの最大値を LMMAX, LNMAX として設定し, 配列 VW2, IVW2 を確保します. 本サブルーチン内部で実際に必要とするこの矩形領域のサイズは, 行列 $\boldsymbol{A}$ の特性や IPAR(2)など近似逆行列を求めるための各種パラメタの設定方法に応じて変化します. 従って, 以下に述べる目安で設定して本サブルーチンを呼び出すこと, 解が得られない場合は, 更に値を拡大し試みることを奨めます.

LMMAX は, 行列 A の非零要素に関連した値ですが, 第 k 列に着目した時, この第 k 列に含まれる非零要素の数と, 第 k 列の非零要素に連動する他の列に含まれる非零要素の総数を定義します. 各列に対応した総数の中で最大の大きさに比例した値を LMMAX に指定します. 通常, 1000 程度で充分ですが, 非零要素の密度が高い場合や要素間の相関が強い場合, 更に特定の列に非零要素を多く持つ場合などは大きくします.

LNMAX は, 近似逆行列 $\boldsymbol{M}$ の各列ベクトルに含まれる非零要素の中で, 最大の値に比例した値を指定します. 最大の値は, 下記の式で算定します.

$$\max_k \left[\max(\ 1, nz_k \times IPAR\ (2)/100\ \ )\right]$$

LNMAX には, 最大値予測の 1.2 倍程度の値を指定します.

正常に処理を終了した場合, 実際に使用した領域の大きさが, LMMAX と LNMAX に対応してそれぞれ IPAR(13)と IPAR(14)に格納されます.

b. 使用例

連立 1 次方程式 $\boldsymbol{Ax=f}$ を解きます. 行列 $\boldsymbol{A}$ は境界条件として立方体の境界で 0 となる楕円型の偏微分方程式に有限差分法を適用して生成されます.

$-\Delta \boldsymbol{u} + \boldsymbol{a}\nabla \boldsymbol{u} + \boldsymbol{u} = \boldsymbol{f}$

ここで $\boldsymbol{a} = (a_1, a_2, a_3)$, a_1, a_2 および a_3 はある定数です. 行列 $\boldsymbol{A}$ はサブルーチン INIT_MAT_DIAG によって生成されます. これを圧縮型格納法に変換します. (並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C **EXAMPLE**
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (NORD=60)
      PARAMETER (NX=NORD,NY=NORD,NZ=NORD)
      PARAMETER (N=NX*NY*NZ)
```

```
      PARAMETER (K=N+1,NDIAG=7,L=4)
      PARAMETER (LMMAX=1000,LNMAX=200,NUMT=4)
      DIMENSION  NOFST(NDIAG),DIAG(K,NDIAG),DIAG2(K,NDIAG)
      DIMENSION  A(K*NDIAG),NROW(K*NDIAG),NFCNZ(N+1),
     +           W(K*NDIAG),IW(2,K*NDIAG)
      DIMENSION  X(N),B(N),SOLEX(N),Y(N),IVW(N)
      DIMENSION  VW2(LMMAX,LNMAX+3,NUMT),IVW2(LMMAX,3,NUMT)
      DIMENSION  IPAR(20),RPAR(20)
      DIMENSION  NFRNZ(N+1),NFRNZM(N+1)
C
      REAL*8,ALLOCATABLE  :: AA(:),AM(:),VW1(:)
      INTEGER,ALLOCATABLE :: NCOL(:),JVWB(:),
     +                       NCOLM(:),IVW1(:)
C
      PRINT *,' *** SPARSE LINEAR EQUATIONS BY IDR METHOD',
     +        ' WITH PRECONDITIONING'
      PRINT *,' *** COMPRESSED ROW STORAGE.'
      PRINT *
      SOLEX(1:N)=1.0D0
      PRINT *,' *** EXPECTED SOLUTIONS'
      PRINT *,' X(1) = ',SOLEX(1),' X(N) = ',SOLEX(N)
      PRINT *
      VA1 = 3.0D0
      VA2 = 1.0D0/3D0
      VA3 = 5.0D0
      VC = 1.0D0
      XL = 1.0D0
      YL = 1.0D0
      ZL = 1.0D0
      CALL INIT_MAT_DIAG(VA1,VA2,VA3,VC,DIAG,NOFST
     & ,NX,NY,NZ,XL,YL,ZL,NDIAG,N,K)
      DO I=1,NDIAG
C
      IF(NOFST(I).LT.0)THEN
         NBASE=-NOFST(I)
         LENGTH=N-NBASE
         DIAG2(1:LENGTH,I)=DIAG(NBASE+1:N,I)
      ELSE
         NBASE=NOFST(I)
         LENGTH=N-NBASE
         DIAG2(NBASE+1:N,I)=DIAG(1:LENGTH,I)
      ENDIF
```

```
C
      ENDDO
C
      NUMNZ=1
      DO J=1,N
         NTOPCFG=1
         DO I=NDIAG,1,-1
            IF(DIAG2(J,I).NE.0.0D0)THEN
               NCOLL=J-NOFST(I)
               A(NUMNZ)=DIAG2(J,I)
               NROW(NUMNZ)=NCOLL
               IF(NTOPCFG.EQ.1)THEN
                  NFCNZ(J)=NUMNZ
                  NTOPCFG=0
               ENDIF
               NUMNZ=NUMNZ+1
            ENDIF
         ENDDO
      ENDDO
      NFCNZ(N+1)=NUMNZ
      NNZ=NUMNZ-1
      CALL DM_VMVSCC(A,NNZ,NROW,NFCNZ,N,SOLEX,B,W,IW,ICON)
      ERR1 = ERRNRM(SOLEX,X,N)
C
      X(1:N)=0.0D0
      CALL DM_VMVSCC(A,NNZ,NROW,NFCNZ,N,X,Y,W,IW,ICON)
      ERR2 = ERRNRM(Y,B,N)
C
      ALLOCATE (AA(NNZ),NCOL(NNZ),AM(NNZ),NCOLM(NNZ)
     +         ,VW1(NNZ),IVW1(NNZ))
      ISW=1
      DO I=1,20
         IPAR(I)=0
         RPAR(I)=0.0D0
      END DO
      NWM=NNZ
      NZM=0
C
      CALL CONVGCR(A,N,NFCNZ,NROW,AA,NFRNZ,NCOL,IVW)
      CALL DM_VLSPAXCR2(AA,NNZ,NCOL,NFRNZ,N,B,ISW,X
     +    ,AM,NZM,NCOLM,NFRNZM,NWM,IPAR,RPAR
     +    ,VW1,IVW1,VW2,IVW2,LMMAX,LNMAX,NUMT,ICON)
```

```
C
      EPS=RPAR(2)
      ITMAX=2000
      ERR3 = ERRNRM(SOLEX,X,N)
      CALL DM_VMVSCC(A,NNZ,NROW,NFCNZ,N,X,Y,W,IW,ICONT)
      ERR4 = ERRNRM(Y,B,N)
      PRINT *,' *** COMPUTED SOLUTIONS'
      PRINT *,' X(1) = ',X(1),' X(N) = ',X(N)
      PRINT *
      PRINT *,' DM_VLSPAXCR2 ICON = ',ICON
      PRINT *
      PRINT *,' N        = ',N
      PRINT *,'       NX = ',NX
      PRINT *,'       NY = ',NY
      PRINT *,'       NZ = ',NZ
      PRINT *,' ITER MAX = ',ITMAX
      PRINT *,' ITER     = ',IPAR(7)
      PRINT *,' ICMAV    = ',IPAR(8)
      PRINT *
      PRINT *,' EPS      = ',RPAR(2)
      PRINT *
      PRINT *,' INITIAL ERROR           = ',ERR1
      PRINT *,' INITIAL RESIDUAL ERROR  = ',ERR2
      PRINT *,' CRITERIA RESIDUAL ERROR = ',ERR2*EPS
      PRINT *
      PRINT *,' ERROR                   = ',ERR3
      PRINT *,' RESIDUAL ERROR          = ',ERR4
      PRINT *
      PRINT *
      IF(ERR4.LE.ERR2*EPS*1.1.AND.ICON.EQ.0)THEN
      WRITE(*,*)' ********** OK **********'
      ELSE
      WRITE(*,*)' ********** NG **********'
      ENDIF
      STOP
      END
C =======================================
C INITIALIZE COEFFICIENT MATRIX
C =======================================
      SUBROUTINE INIT_MAT_DIAG(VA1,VA2,VA3,VC,D_L,OFFSET
     & ,NX,NY,NZ,XL,YL,ZL,NDIAG,LEN,NDIVP)
      IMPLICIT REAL*8(A-H,O-Z)
```

```
      DIMENSION D_L(NDIVP,NDIAG)
      INTEGER OFFSET(NDIAG)
C
      IF (NDIAG .LT. 1) THEN
         WRITE (*,*) 'SUBROUTINE INIT_MAT_DIAG:'
         WRITE (*,*) ' NDIAG SHOULD BE GREATER THAN OR EQUAL TO 1'
      RETURN
      ENDIF
!$OMP PARALLEL DEFAULT(PRIVATE)
!$OMP+ SHARED(VA1,VA2,VA3,VC,D_L,OFFSET
!$OMP+ ,NX,NY,NZ,XL,YL,ZL,NDIAG,LEN,NDIVP)
C NDIAG CANNOT BE GREATER THAN 7
      NDIAG_LOC = NDIAG
      IF (NDIAG .GT. 7) NDIAG_LOC = 7
C INITIAL SETTING
      HX = XL/(NX+1)
      HY = YL/(NY+1)
      HZ = ZL/(NZ+1)
!$OMP DO
      DO I = 1,NDIVP
      DO J = 1,NDIAG
         D_L(I,J) = 0.0
      ENDDO
      ENDDO
!$OMP ENDDO
      NXY = NX*NY
C OFFSET SETTING
!$OMP SINGLE
      L = 1
      IF (NDIAG_LOC .GE. 7) THEN
         OFFSET(L) = -NXY
         L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 5) THEN
         OFFSET(L) = -NX
         L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 3) THEN
         OFFSET(L) = -1
         L = L+1
      ENDIF
      OFFSET(L) = 0
```

```
      L = L+1
      IF (NDIAG_LOC .GE. 2) THEN
         OFFSET(L) = 1
         L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 4) THEN
         OFFSET(L) = NX
         L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 6) THEN
         OFFSET(L) = NXY
      ENDIF
!$OMP END SINGLE
C MAIN LOOP
!$OMP DO
      DO 100 J = 1,LEN
         JS = J
C DECOMPOSE JS-1 = (K0-1)*NX*NY+(J0-1)*NX+I0-1
         K0 = (JS-1)/NXY+1
         IF (K0 .GT. NZ) THEN
            PRINT*,'ERROR; K0.GH.NZ '
            GOTO 100
         ENDIF
         J0 = (JS-1-NXY*(K0-1))/NX+1
         I0 = JS - NXY*(K0-1) - NX*(J0-1)
         L = 1
         IF (NDIAG_LOC .GE. 7) THEN
            IF (K0 .GT. 1) D_L(J,L) = -(1.0/HZ+0.5*VA3)/HZ
            L = L+1
         ENDIF
         IF (NDIAG_LOC .GE. 5) THEN
            IF (J0 .GT. 1) D_L(J,L) = -(1.0/HY+0.5*VA2)/HY
            L = L+1
         ENDIF
         IF (NDIAG_LOC .GE. 3) THEN
            IF (I0 .GT. 1) D_L(J,L) = -(1.0/HX+0.5*VA1)/HX
            L = L+1
         ENDIF
         D_L(J,L) = 2.0/HX**2+VC
         IF (NDIAG_LOC .GE. 5) THEN
            D_L(J,L) = D_L(J,L) + 2.0/HY**2
            IF (NDIAG_LOC .GE. 7) THEN
```

```
                  D_L(J,L) = D_L(J,L) + 2.0/HZ**2
               ENDIF
            ENDIF
            L = L+1
            IF (NDIAG_LOC .GE. 2) THEN
               IF (I0 .LT. NX) D_L(J,L) = -(1.0/HX-0.5*VA1)/HX
               L = L+1
            ENDIF
            IF (NDIAG_LOC .GE. 4) THEN
               IF (J0 .LT. NY) D_L(J,L) = -(1.0/HY-0.5*VA2)/HY
               L = L+1
            ENDIF
            IF (NDIAG_LOC .GE. 6) THEN
               IF (K0 .LT. NZ) D_L(J,L) = -(1.0/HZ-0.5*VA3)/HZ
            ENDIF
  100 CONTINUE
!$OMP ENDDO
!$OMP END PARALLEL
      RETURN
      END
C =======================================
C     ABSOLUTE ERROR : | X1 - X2 |
C =======================================
      REAL*8 FUNCTION ERRNRM(X1,X2,LEN)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X1(*),X2(*)
      S = 0D0
      DO 100 I = 1,LEN
         SS = X1(I) - X2(I)
         S = S + SS * SS
  100 CONTINUE
      ERRNRM = SQRT( S )
      RETURN
      END
C=============================================================
C     MODE CONV UNSYM MATRIX FROM COMPRESSED COLUMN TO ROW.
C=============================================================
      SUBROUTINE CONVGCR(AC,N,IC,JC,AR,IR,JR,IW)
      IMPLICIT   REAL*8 (A-H,O-Z)
      DIMENSION  AC(*),IC(N+1),JC(*),AR(*),IR(N+1),JR(*),IW(N)
      NZ=IC(N+1)-1
      DO I=1,N+1
```

```
      IR(I)=0
   END DO
   DO J=1,NZ
      IR(JC(J)+1)=IR(JC(J)+1)+1
   END DO
   IR(1)=1
   DO I=2,N+1
      IR(I)=IR(I)+IR(I-1)
   END DO
   DO I=1,N
      IW(I)=IR(I)
   END DO
   ICOL=1
   DO J=1,NZ
      IF(J.EQ.IC(ICOL+1)) ICOL=ICOL+1
      JR(IW(JC(J)))=ICOL
      AR(IW(JC(J)))=AC(J)
      IW(JC(J))=IW(JC(J))+1
   END DO
   RETURN
   END
```

(4)　手法概要

本サブルーチンでは, 実非対称スパース行列の連立 1 次方程式解法として, 近似逆行列に基づく前処理付きの演繹的次元縮小法 (*IDRstab(s,l)* 法) を用いています.

a.　近似逆行列

一般的に反復法の収束は保証されないか, または収束が極めて緩やかになります. このために与えられた連立 1 次方程式を, より扱い易い形式に変換する前処理を施しますが, これにより反復回数を縮小し収束を速められる可能性があります. この前処理の手法としては, 不完全分解法 (*ILU* 法)などが著名です. この不完全分解法は, アルゴリズムが簡素で条件の良い行列では極めて効率的な特性を持つ反面, 反復過程で多用する三角方程式の解法での漸化的な計算が, 並列処理で非効率になる場合があります. 本サブルーチンではこの問題を改善し, 並列処理向けに適合した前処理技法として, 近似逆行列法を採用しています.

本サブルーチンの反復解法は, 前処理を導入した連立 1 次方程式

$\boldsymbol{AMy = b,\ x = My}$

により解きます. ここで $\boldsymbol{M}$ は近似逆行列です. この $\boldsymbol{M}$ を計算するために,

$\|\boldsymbol{AM - I}\|$

の最小化問題を解きますが, そこでフロベニウスノルムを導入します. このノルムを選択することで, 行列 $\boldsymbol{M}$ の列 $\boldsymbol{m}_k$ を独立して並列に計算することが出来ます. 即ち,

$$\|\boldsymbol{AM} - \boldsymbol{I}\|_F^2 = \sum_{k=1}^{n} \|(\boldsymbol{AM} - \boldsymbol{I})\boldsymbol{e}_k\|_2^2$$

より, 近似逆行列の列ベクトル $\boldsymbol{m}_k$ 毎の最小二乗問題

$$\min_{m_k} \|\boldsymbol{A}\boldsymbol{m}_k - \boldsymbol{e}_k\|_2, k = 1,...,n$$

により, 並列的に求めます.

$\boldsymbol{m}_k$の初期値としては, デフォルトとして単位行列を採用します. この残差ミニマムの最小二乗問題を QR 法で解きます. その結果の残差ベクトルには, 従来 $\boldsymbol{m}_k$には無かった部分に非零要素が発生しますが, その新インデックスの中から最も残差ミニマムに貢献するインデックスを選択し, 順次 $\boldsymbol{m}_k$を改訂します. その際に残差ベクトルのノルムが収束判定値 RPAR(1)を満たした場合, 即ち

$\|\boldsymbol{A}\boldsymbol{m}_k - \boldsymbol{e}_k\|_2 \le eps$

または IPAR(2)に基づく要素数の上限に達した場合に, 近似逆行列のベクトル $\boldsymbol{m}_k$ が収束したと見なします.

b. 演繹的次元縮小法 (*IDRstab(s,l)*法)

演繹的次元縮小法は, クリロフ部分空間法の一種です. 本サブルーチンでは, 加速多項式が 1 次である初期の *IDR(s)*の欠点を改善し, 複数次の加速多項式を持つ方法 *IDRstab(s,l)*法を採用しています. ここで *s* はシャドウ残差の次数, *l* は安定化の加速多項式の次数です.

本サブルーチンでは, パラメタ *s, l* を任意に選択することが出来ますが, *l*=1 の場合は, *s* が大きい時の安定化を改善するために双直交化を図った *BIDR(s)*法を併設し, 自動的に切り換えています.

c. 収束判定法

反復解法としての収束判定として, 以下の条件を満たした時に, 解に収束したと見なします.

$$\|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k\|_2 \le epst\|\boldsymbol{b}\|_2$$

ここで *epst* は収束判定値で, パラメタ RPAR(2)に指定します. デフォルト値は 10^{-8} です. パラメタ RPAR(3)には, 残差ベクトルの相対ノルム

$$\|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k\|_2 / \|\boldsymbol{b}\|_2$$

が設定されます. 収束判定を満たさない場合でも, 本サブルーチンでの終了までに得られた解に対するノルムが設定されます.

この残差ベクトル $\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k$ は, 反復公式に於ける残差ベクトルに関する漸化式に基づく公式で計算されます.

アルゴリズムの詳細については, “付録 1 参考文献一覧表” の[29], [31], [73]を参照して下さい.

DM_VLSX

正値対称行列の連立 1 次方程式 (ブロック化された変形コレスキー分解法)
CALL DM_VLSX(A, K, N, B, EPSZ, ISW, ICON)

(1)　機能

実係数連立 1 次方程式(1.1)の係数行列 $\boldsymbol{A}$ を, ブロック化された外積形の変形コレスキー分解法で, (1.2)のように分解し, 続いて, 方程式を解きます. ただし, $\boldsymbol{A}$ は $n \times n$ の正値対称行列, $\boldsymbol{b}$ は n 次元の実定数ベクトル, $\boldsymbol{x}$ は n 次元の解ベクトル, $\boldsymbol{L}$ は単位下三角行列, $\boldsymbol{D}$ は対角行列とします. また, $n \geq 1$ とします.

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{1.1}$$

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^{\mathrm{T}} \tag{1.2}$$

(2)　パラメタ

A.....入力. 係数行列 $\boldsymbol{A}$.
入力時には, A(1:N, 1:N)の下三角部分$\{\mathrm{A}(i, j) \mid i \geq j\}$に, $\boldsymbol{A}$ の下三角部分$\{a_{ij} \mid i \geq j\}$を格納します.
出力時には, A(1:N, 1:N)の下三角部分$\{\mathrm{A}(i, j) \mid i \geq j\}$に $\boldsymbol{L}$ と $\boldsymbol{D}^{-1}$ が以下のように格納されます.
l_{ij}　　$i > j$ のとき,
d_{ii} の逆数　$i = j$ のとき,
不定　　$i < j$ のとき,
(図 DM_VLSX-1 参照)
A(K, N)なる倍精度実数型 2 次元配列.

K.....入力. 配列 A の 1 次元目の大きさ(≥ N).

N.....入力. 係数行列 $\boldsymbol{A}$ の次数 n.

B.....入力. 定数ベクトル $\boldsymbol{b}$.
出力. 解ベクトル $\boldsymbol{x}$.
B(N)なる倍精度実数型 1 次元配列.

EPSZ.....入力. ピボットの相対零判定値(≥ 0.0).
0.0 のときは標準値が採用されます.
((3)使用上の注意 a. 注意①参照).

ISW.....入力. 制御情報.
同一の係数行列を持つ複数組の方程式を解くとき, 次のように指定します.
ISW=1 のとき, 1 組目の方程式を解きます.
ISW=2 のとき, 2 組目以降の方程式を解きます.
ISW=2 のときは, 配列 B の値だけを新しい定数ベクトル $\boldsymbol{b}$ の値に変え, それ以外の引数の内容は変更せずに使用してください.
((3)使用上の注意 a. 注意②参照).

ICON.....出力. コンディションコード.
(表 DM_VLSX-1 参照)

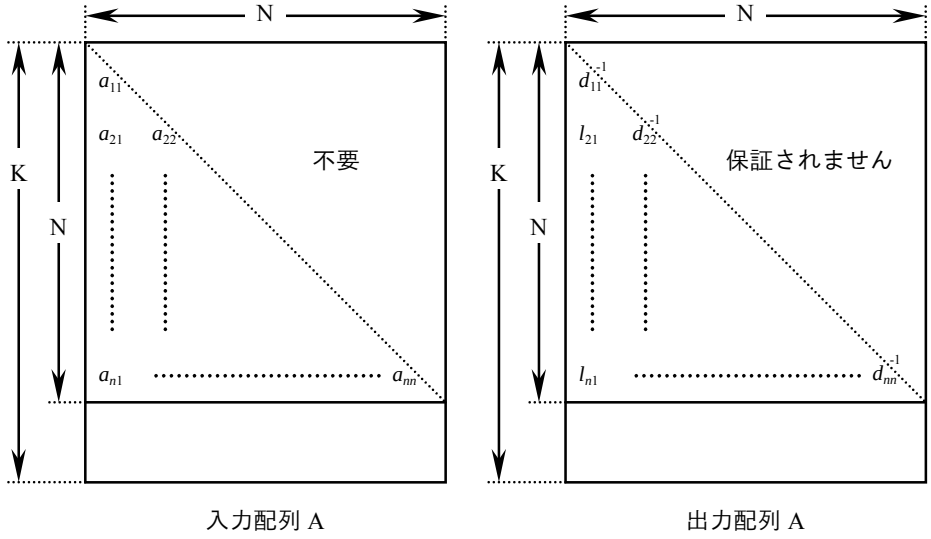


図 DM_VLSX-1 コレスキー分解法のデータの格納方法

LDL^{T}分解を行う正値対称行列の対角要素および下三角部分 a_{ij} を配列 A$(i,j), i=j, \ldots, n, j=1, \ldots, n$ に格納します.
LDL^{T}分解された結果は, 行列 $\boldsymbol{D}^{-1}$ を対角部分に, $\boldsymbol{L}$ の対角要素を除いた部分を下三角部分にそれぞれ格納されます. 上三角部分の値は保証されません.

表 DM_VLSX-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし.	ー
10000	ピボットが負となった. 係数行列は正値でない.	処理は続行する.
20000	ピボットが相対的に零となった. 係数行列は非正則である可能性が強い.	処理を打ち切る.
30000	N<1, EPSZ<0, K<N, ISW≠1, 2.	

(3)　使用上の注意

a.　注意

①　ピボットの相対零判定値にある値を設定したとすると, この値は次の意味を持っています. すなわち, 変形コレスキー分解法による LDL^{T} 分解の過程で, ピボットの絶対値がその値より小さくなった場合に, そのピボットの値を相対的に零と見なし, ICON=20000 として処理を打ち切ります. EPSZ の標準値は, 丸め誤差の単位を u としたとき, EPSZ=16u です.

なお, ピボットの絶対値が小さくなっても, 処理を続行させたい場合には, EPSZ に極小の値を設定すればよいのですが, 結果の精度は保証されません.

② 同一の係数行列を持つ連立 1 次方程式をいくつか続けて解く場合には, 2 回目以降, ISW=2 として解くと, 係数行列 $\boldsymbol{A}$ の LDL^T 分解過程を省略するので, 計算時間を節約できます.

③ 分解の途中でピボットが負となった場合, 係数行列は正値でないことになります. このとき, ICON=10000 としますが, 処理は続行します. ただし, ピボッティングを行っていないため, 計算誤差は大きい可能性があることを考慮してください.

④ 係数行列の行列式の値を求めたいときは, 演算後の配列 A の n 個の対角要素を掛け合わせ, その逆数をとって下さい.

⑤ 本サブルーチンは, 内部で DM_VSLDL および DM_VLDLX を呼び出しています. 本機能をパラレルリージョンで生成されるスレッドから呼び出し, 並列実行するスレッド数を実行環境ルーチン OMP_SET_NUM_THREADS( )で設定したいときは, DM_VSLDL および DM_VLDLX を直接呼出し, そのおのおのの直前で OMP_SET_NUM_THREADS( )を呼び出して設定して下さい.

b. 使用例

4000×4000 の係数行列の連立 1 次方程式を解きます.
(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER (K=4001,N=4000)
      REAL*8    A(K,N),B(N)
C
!$OMP PARALLEL DEFAULT(PRIVATE) SHARED(A,B)
!$OMP DO
      DO J=1,N
      DO I=J,N
      A(I,J)=MIN(I,J)
      ENDDO
      ENDDO
!$OMP END DO

!$OMP DO
      DO I=1,N
      B(I)=I*(I+1)/2+I*(N-I)
      ENDDO
!$OMP END DO
```

```
!$OMP END PARALLEL

      ISW=1
      CALL DM_VLSX(A,K,N,B,1.D-13,ISW,ICON)
      WRITE(6,610) ICON
      IF(ICON.GE.20000) GO TO 100
      WRITE(6,620) (B(I),I=1,10)
C
      S=1.0D0
!$OMP PARALLEL DO DEFAULT(PRIVATE) SHARED(A)
!$OMP+          REDUCTION(*:S)
      DO I=1,N
      S=S*A(I,I)
      ENDDO
!$OMP END PARALLEL DO

      DET=1.0D0
      DET=1.D0/DET
      WRITE(6,630) DET
  100 STOP
  600 FORMAT(1H1/10X,6HORDER=,I5)
  610 FORMAT(1H0,10X,5HICON=,I5)
  620 FORMAT(11X,15HSOLUTION VECTOR
     */(10X,3D23.16))
  630 FORMAT(1H0,10X
     *,34HDETERMINANT OF COEFFICIENT MATRIX=
     *,D23.16)
      END
```

(4) 手法概要

ブロック化された外積形の変形コレスキー分解法については, “付録 1 参考文献一覧表” の [30], [54],[70]を参照して下さい.

DM_VLUX

LU 分解された実行列の連立 1 次方程式
CALL DM_VLUX(B, FA, KFA, N, IP, ICON)

(1)　機能

以下のような LU 分解された実係数行列を持つ連立 1 次方程式を解きます.

$$\boldsymbol{LUx} = \boldsymbol{Pb}$$

ただし, $\boldsymbol{L}$, $\boldsymbol{U}$ はそれぞれ $n \times n$ の下三角行列と単位上三角行列, $\boldsymbol{P}$ は置換行列(係数行列を LU 分解するときの部分ピボッティングによる行の入換えを行います), $\boldsymbol{b}$ は n 次元の実定数ベクトル, $\boldsymbol{x}$ は n 次元の解ベクトルです. ($n \geq 1$)

(2)　パラメタ

B.....入力. 定数ベクトル $\boldsymbol{b}$.
　　出力. 解ベクトル $\boldsymbol{x}$.
　　B(N)なる倍精度実数型 1 次元配列.

FA .....入力. 行列 $\boldsymbol{L}$ と行列 $\boldsymbol{U}$ を FA(1:N, 1:N)に格納します.
　　図 DM_VLUX-1 参照
　　FA(KFA, N)なる倍精度実数型 2 次元配列.

KFA .....入力. 格納配列 FA の 1 次元目の大きさ(≥ N).

N.....入力. 行列 $\boldsymbol{L}$, $\boldsymbol{U}$ の次数 n.

IP.....入力. 部分ピボッティングによる行の入換えの履歴を示すトランスポジションベクトル.
　　大きさ n の 1 次元配列.
　　(サブルーチン DM_VALU の(3)使用上の注意 a. 注意②参照).

ICON.....出力. コンディションコード.
　　表 DM_VLUX-1 参照.

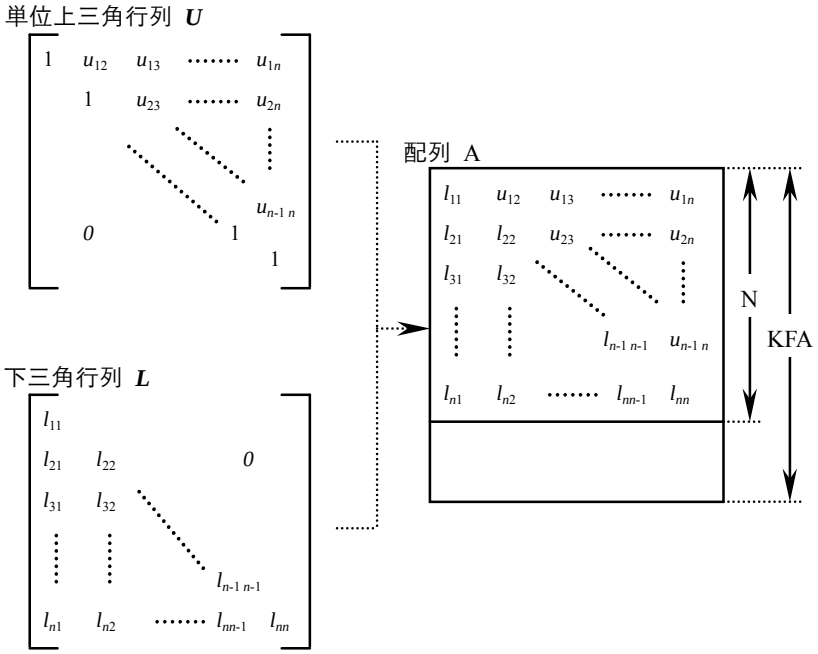


図 DM_VLUX-1 配列 FA における ***L*** 及び ***U*** の格納方法

LU 分解された行列 ***L*** および ***U*** は,***U*** の対角要素を除いた上三角部分と ***L*** を配列 FA(1:N, 1:N)に格納します.

表 DM_VLUX-1 コンディションコード

コード	意　　味	処理内容
0	エラーなし.	ー
20000	係数行列が非正則であった.	処理を打ち切る.
30000	KFA<N, N<1 IP に誤りがあった.	

(3)　使用上の注意

a.　注意

①　連立 1 次方程式を解く場合, サブルーチン DM_VALU を呼び出して, 係数行列を LU 分解してから, 本サブルーチンを呼び出せば方程式を解くことができます. しかし, 通常はサブルーチン DM_VLAX を呼び出せば, 一度に解が求められます.

b.　使用例

4000 × 4000 の係数行列を LU 分解して, 連立 1 次方程式を解きます.
(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (IPN=4)
      DIMENSION  A(4001,4000)
      DIMENSION  B(4000),IP(4000)
C
      N=4000

!$OMP PARALLEL DEFAULT(PRIVATE) SHARED(A,B,N)
!$OMP DO
      DO J=1,N
      DO I=1,N
      A(I,J)=MIN(I,J)
      ENDDO
      ENDDO
!$OMP END DO

!$OMP DO
      DO I=1,N
      B(I)=I*(I+1)/2+I*(N-I)
      ENDDO
!$OMP END DO
!$OMP END PARALLEL
C
      KFA=4001
      CALL  DM_VALU(A,KFA,N,0.0D0,IP,IS,ICON)
      WRITE(6,610)ICON
      IF(ICON.GE.20000)STOP
      CALL  DM_VLUX(B,A,KFA,N,IP,ICON)
      WRITE(6,620)ICON
      WRITE(6,630)(I,B(I),I=1,10)

  610 FORMAT(1H0,10X,26HCONDITION CODE (DM_VALU) =,I5)
  620 FORMAT(1H0,10X,26HCONDITION CODE (DM_VLUX) =,I5)
  630 FORMAT(1H0,10X,17HSOLUTION VECTOR =,
     */(10X,5(1H(,I3,1H),D23.16)))
      END
```

DM_VMGGM

行列の積 (実行列)
CALL DM_VMGGM(A, KA, B, KB, C, KC, M, N, L, ICON)

(1) 機能

$m \times n$ 実行列 ***A*** と, $n \times l$ の実行列 ***B*** の積 ***C*** を求めます.

$$\boldsymbol{C} = \boldsymbol{AB}$$

ここで ***C*** は, $m \times l$ の実行列です. なお, $m, n, l \geq 1$ とします.

(2) パラメタ

A.....入力. 行列 ***A***. データは, A(1:M, 1:N)に格納します.
A(KA, N)なる倍精度実数型 2 次元配列.

KA.....入力. 配列 A の 1 次元目の大きさ(≥ M).

B.....入力. 行列 ***B***. データは, B(1:N, 1:L)に格納します.
B(KB, L)なる倍精度実数型 2 次元配列.

KB.....入力. 配列 B の 1 次元目の大きさ(≥ N).

C.....出力. 行列 ***C***. データは, C(1:M, 1:L)に格納されます.
C(KC, L)なる倍精度実数型 2 次元配列.

KC.....入力. 配列 C の 1 次元目の大きさ(≥ M).

M.....入力. 行列 ***A, C*** の行数 *m*.

N.....入力. 行列 ***A*** の列数及び行列 ***B*** の行数 *n*.

L.....入力. 行列 ***B, C*** の列数 *l*.

ICON.....出力. コンディションコード.
表 DM_VMGGM-1 参照.

表 DM_VMGGM-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし	―
30000	M<1, N<1, L<1, KA<M, KB<N, KC<M	処理を打ち切る.

(3) 使用上の注意

a. 使用例

実行列 ***A, B*** の積を求めます.

本使用例中のサブルーチン PGM は実行列を印刷するものです.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     ** EXAMPLE **
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER (KK=4001,M=4000,N=M,L=M)
      PARAMETER (KA=KK,KB=KK,KC=KK)
      REAL*8    A(KA,N),B(KB,L),C(KC,L)

C
!$OMP PARALLEL DEFAULT(PRIVATE) SHARED(A,B)
!$OMP DO
      DO J=1,M
      DO I=1,N
      IF(J.GT.I)THEN
      A(I,J)=0.0d0
      ELSE
      A(I,J)=1.0d0
      ENDIF
      ENDDO
      ENDDO
!$OMP END DO

!$OMP DO
      DO J=1,M
      DO I=1,N
      IF(J.GE.I)THEN
      B(I,J)=1.0d0
      ELSE
      B(I,J)=0.0d0
      ENDIF
      ENDDO
      ENDDO
!$OMP END DO
!$OMP END PARALLEL

      CALL DM_VMGGM(A,KA,B,KB,C,KC,M,N,L,ICON)
      IF(ICON.NE.0) GOTO 10
      CALL PGM(A,KA,N)
      CALL PGM(B,KB,L)
      CALL PGM(C,KC,L)
      GOTO 10

  150 FORMAT(1H1///10X,27H** MATRIX MULTIPLICATION **)
  10  STOP
```

```
      END

C     ** MATRIX PRINT(REAL NON-SYMMETRIC) **
      SUBROUTINE PGM(A,KA,N)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(KA,N)
      DO 10 I=1,5
      WRITE(6,610) I,(J,A(I,J),J=1,5)
   10 CONTINUE
      RETURN
  610 FORMAT(/5X,I3,3(4X,I3,D23.16),(/8X,3(4X,I3,D23.16)))
      END
```

(4)　手法概要

本サブルーチンでは, ブロック化した行列積手法を用いています. 詳細については, “付録 1 参考文献一覧表” の[30]を参照して下さい.

DM_VMINV

実行列の逆行列 (ブロック化された Gauss-Jordan 法)
CALL DM_VMINV(A, K, N, EPSZ, ICON)

(1)　機能

Gauss-Jordan 法により正則な $n \times n$ 実行列 $\boldsymbol{A}$ の逆行列 $\boldsymbol{A}^{-1}$ を求めます.

(2)　パラメタ

A.....入力. 行列 $\boldsymbol{A}$ を A(1:N, 1:N)に格納します.
　　出力. 行列 $\boldsymbol{A}^{-1}$ が A(1:N, 1:N)に格納されます.
　　A(K, N)なる倍精度実数型 2 次元配列.

K.....入力. 格納配列 A の 1 次元目の大きさ(≥ N).

N.....入力. 行列 $\boldsymbol{A}$ の次数 n.

EPSZ.....入力. ピボットの相対零判定値(≥ 0.0).
　　0.0 が入力されたときは, 標準値が採用されます.
　　((3)使用上の注意 a. 注意①参照).

ICON.....出力. コンディションコード.
　　表 DM_VMINV-1 参照.

表 DM_VMINV-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし.	ー
20000	行列 $\boldsymbol{A}$ のある行の要素がすべて零であったか, またはピボットが相対的に零となった. 行列 $\boldsymbol{A}$ は非正則である可能性が強い.	処理を打ち切る.
30000	N<1, K<N または EPSZ<0.0	

(3)　使用上の注意

a.　注意

①　部分ピボッティングで選択されたピボット要素が 0.0 か絶対値が EPSZ 以下なら, 相対的に零とみなして, ICON=20000 として処理を打ち切ります. EPSZ の標準値は丸め誤差の単位を u としたとき, $16u$ です. EPSZ に極小の値を設定すると, ピボットの絶対値が小さくなっても処理は続行されますが, 計算結果の精度は保証されません.

b. 使用例

逆行列を求めます.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (N=2000,K=N+1)
C
      REAL*8    A(K,N),AS(K,N)
C

      C=SQRT(2.0D0/DBLE(1+N))
      T=DATAN(1.0D0)*4.0D0/(1+N)
C
      DO 100 J=1,N
      DO 100 I=1,N
      A(I,J)=C*SIN(T*I*J)
      AS(I,J)=A(I,J)
 100  CONTINUE
C
      EPSZ=0.0D0
      CALL  DM_VMINV(A,K,N,EPSZ,ICON)
      PRINT*,'ICON=',ICON
C
      TMP=0.0D0
      DO I=1,N
      DO J=1,N
      TMP2=DABS(A(I,J)-AS(I,J))
      IF(TMP2.GT.TMP)TMP=TMP2
      ENDDO
      ENDDO
      PRINT*,'ORDER=',N,' ; ERROR = ',TMP
C
      STOP
      END
```

(4) 手法概要

ブロック化された Gauss-Jordan 法(“付録 1 参考文献一覧表”の[30]を参照)によって逆行列を求めます.

DM_VMLBIFE

スパース行列の連立 1 次方程式 (不完全ブロック分解に基づくマルチレベル前処理付き反復法, ELLPACK 形式格納法)
CALL DM_VMLBIFE (A, K, IWIDT, N, ICOL, B, ISW, IGUSS, INFO, INFOEP, EPSOT, EPSIN, EPSEP, X, W, NW, IW, NIW, ICON)

(1)　機能

$n \times n$ のスパース行列を係数行列とする連立 1 次方程式を反復法で解きます.

$$\boldsymbol{Ax} = \boldsymbol{b}$$

$n \times n$ の係数行列 $\boldsymbol{A}$ は, 対角形式の格納法で格納します. ベクトル $\boldsymbol{b}$ および $\boldsymbol{x}$ は n 次元ベクトルです.

反復法は行列 $\boldsymbol{A}$ が対称のときは, ORTHOMIN 法が, 非対称のときは, GMRES 法が使用できます. この反復解法(以下外部反復と呼びます)にはマルチレベル構造を持つ不完全ブロック分解による前処理が施されていて, 安定な解法です. 各レベルである変数の集合に対して消去を行い, その過程で消去される変数の集合からなる部分行列に対する近似的逆行列を利用します.

消去の手続きは, いわゆる最も粗いレベルの小さな連立 1 次方程式が生成されるまで続きます. 外部反復の各ステップに対して, 最も粗いレベルの連立 1 次方程式は反復法(内部反復)によって解かれます.

(2)　パラメタ

A.....入力. 係数行列の非零要素を A(1:N, 1:IWIDT)に格納します.
A(K, IWIDT)なる 2 次元配列.
ELLPACK 形式の格納方法については, “SSL II 拡張機能使用手引書 II 第 I 部 概説 3.2.1.1 一般スパース行列の格納方法” を参照してください.

K.....入力. A および ICOL の 1 次元目の大きさ($\geq n$).

IWIDT.....入力. 係数行列 $\boldsymbol{A}$ の非零要素の行ベクトル方向の最大個数.
A および ICOL の 2 次元目の大きさ.

N.....入力. 行列 $\boldsymbol{A}$ の次数 n.

ICOL .....入力. ELLPACK 形式で使用される列指標で, A の対応する要素がいずれの列ベクトルに属すかを示します.
ICOL(K, IWIDT)なる 2 次元配列.

B.....入力. 連立 1 次方程式の右辺の定数ベクトルを B(1:N)に格納します.
B(K)なる 1 次元配列.

ISW .....入力. 制御情報.
ISW=1 のとき, 初回の呼び出し.
ISW=2 のとき, 2 回目以降の呼び出し. このとき, A, ICOL, IW, W の値は, 初回の呼び出しの結果を変更せずに呼び出す必要があります.
((3)使用上の注意 a. 注意①参照)

IGUSS.....入力. 配列 X に指定された解ベクトルの近似値から反復計算を行うかを指定する制御情報.

IGUSS = 0 のとき, 解ベクトルの近似値を指定しません. このときゼロベクトルから反復を開始します.

IGUSS ≠ 0 のとき, 配列 X に指定された解ベクトルの近似値から反復計算を開始します.

INFO.....入出力. 反復に関する制御情報.

INFO(14)なる 1 次元配列.

行列 ***A*** が対称な場合の指定例.

INFO(1) = 10
INFO(2) = NTHRD × 100
INFO(3) = 0
INFO(5) = 1
INFO(6) = 2000
INFO(10) = 1
INFO(11) = 1000

行列 ***A*** が非対称な場合の指定例.

INFO(1) = 10
INFO(2) = NTHRD × 100
INFO(3) = 0
INFO(5) = 2
INFO(6) = 2000
INFO(7) = 5
INFO(8) = 20
INFO(10) = 2
INFO(11) = 1000
INFO(12) = 10
INFO(13) = 0

NTHRD は, 並列実行するスレッドの数.

INFO(1) = MAXLVL

入力. 代数的マルチレベル法のレベルの最大値.

MAXLVL < 1 のとき, 前処理は施されません.

MAXLVL > 0 のとき, 指定された値以上の粗いレベルは使われません.

((3)使用上の注意 a. 注意⑤, ⑧参照)

INFO(2) = MINUK

入力. もっとも深いレベルの縮小された連立 1 次方程式の未知数の最小個数.

MINUK は N よりずっと小さく, 並列処理するスレッドの数 NTHRD よりずっと大きな値を指定することをすすめます. 例えば, NTHRD × 100.

INFO(3) = NORM

入力. 行列の正規化の方法を指定します.

NORM < 1 のとき, 行列 ***A*** は ***A*** の対角要素の平方根の逆数を対角要素として持つ対角行列で左右から正規化されます. 対称行列を正規化した行列は対

称行列です.
対称な正規化をすすめます. しかしある場合は, 非対称な正規化の方が速く収束することもあり得ます.
((3)使用上の注意 a. 注意③参照)
NORM ≥ 1 のとき, 行列 ***A*** は ***A*** の主対角要素と同じ符号とした行の絶対値の和の逆数からなる対角行列で左から正規化します. ***A*** が対称行列でも正規化されたあとは対称行列ではありません.
((3)使用上の注意 a. 注意④参照)

INFO(4)
出力. 使用されたレベル数.

INFO(5) = METHOT
入力. 外部反復で使われる反復法.
METHOT=1 のとき, 前処理付き ORTHOMIN 法が使われます. 行列 ***A*** が対称で対称な正規化を指定した場合に指定してください.
METHOT≠1 のとき, 切り捨て再起動する GMRES 法が使われます. 行列 ***A*** が非対称であるか, 非対称な正規化が使われた場合に指定してください.

INFO(6) = ITMXOT
入力. 外部反復の最大回数. 例えば, 2000.
外部反復回数がこの値に達したとき, 処理を打ち切ります. このとき返却された解は, 保証されません.

INFO(7) = NRESOT
入力. 外部反復で使われる反復法に GMRES 法を指定したとき, 外部反復の直交手続きで使う残差ベクトルの本数を指定します. 例えば 5. つまり GMRES 法で NRESOT 個のベクトルを使い解を更新し, 残りのベクトルによる更新はされません.
((3)使用上の注意 a. 注意④参照)

INFO(8) = NRSTOT
入力. 外部反復で使われる反復法に GMRES 法を指定したとき, 再起動(restart)する反復回数を指定します. NRSTOT ≥ NRESOT. 例えば 20 回.
NRSTOT < 1 のとき, 再起動は行いません.
((3)使用上の注意 a. 注意④参照)

INFO(9) = ITEROT
出力. 外部反復の回数.

INFO(10) = METHIN
入力. 内部反復で使われる反復法.
METHIN = 1 のとき, 前処理付き ORTHOMIN 法が使われます. 行列 ***A*** が対称で対称な正規化を指定した場合に指定してください.
METHIN ≠ 1 のとき, 切り捨て再起動する GMRES 法が使われます. 行列 ***A*** が非対称であるか, 非対称な正規化が使われた場合に指定してください.

INFO(11) = ITMXIN
入力. 内部反復の最大回数. 例えば, 1000.
内部反復回数がこの値に達すると, 外部反復に処理は引き継がれます.

INFO(12) = NRESIN

入力. 内部反復で使われる反復法に GMRES 法を指定したとき, 内部反復の直交手続きで計算する残差ベクトルの数を指定します. 例えば 10. つまり GMRES 法で NRESOT 個のベクトルを使い解を更新し, 残りのベクトルによる更新はされません.

((3)使用上の注意 a. 注意④参照)

INFO(13) = NRSTIN

入力. 内部反復で使われる反復法に GMRES 法を指定したとき, 再起動(restart)する反復回数を指定します. NRSTIN ≥ NRESIN.

NRSTIN < 1 のとき, 再起動は行いません.

((3)使用上の注意 a. 注意④参照)

INFO(14)

出力. 内部反復の平均回数.

INFOEP.....入力. 消去する変数の部分行列及び縮小された行列の構成に関する制御情報.

INFOEP の代表的な指定例.

各レベルで, Schur の complement の計算などで使う部分行列の逆行列を近似的に求める方法を指定します.

((3)使用上の注意 a. 注意⑦参照)

1) 逆行列を対角行列で近似する

INFOEP(1) = 1

INFOEP(2) = 5

INFOEP(3) = 2 × NROW

NROW = 係数行列 ***A*** の行の非ゼロ要素の代表的な数.

2) 逆行列を反復法で近似的に求める

INFOEP(1) = NROW

INFOEP(2) = 5

INFOEP(3) = 2 × NROW

INFOEP(3)なる 1 次元配列.

INFOEP(1) = MAXNCV

入力. ブロック分解で消去する変数よりなる部分行列の逆行列の近似行列を構成するとき, 各行ベクトルに存在する非ゼロ要素数の最大個数.

典型的な設定は, MAXNCV = 1 または, MAXNCV = MAXNC です. MAXNCV = 1 のときは, 逆行列は対角要素で近似されます.

INFOEP(2) = MAXITV

入力. 近似的に逆行列を計算する最大ステップ数.

MAXITV は, 精度 TAUV での近似的な逆行列を計算するための最大反復ステップを示します. ステップ数が MAXITV に達したとき, 反復は打ち切られます.

いかなる場合も, 近似には $\frac{\log(\mathrm{TAUV})}{\log(\mathrm{LAMBDA})}$ 以下のステップが必要と考えられます.

MAXITV ≤ 1 のとき, 逆行列は主対角要素のみで近似されます.

INFOEP(3) = MAXNC

入力. ブロック分解の後, Schur の complement として縮小された行列に残る要素を制御します.

MAXNC < 2 のとき, TAU よりも小さな要素は除去されます.

MAXNC > 1 のとき, 各行ベクトルに存在する要素の数は MAXNC により制御されます. 最も大きな MAXNC 個の要素が残ります. 他の要素は TAU より大きくても除去されます.

EPSOT .....入力. 解の収束判定値.

外部反復の k ステップで求められた解ベクトルが正規化された残差ベクトル $\hat{r}_k = \hat{A}x_k - \hat{b}_k$ に対して以下の条件を満たしたときベクトルは収束したと見なします. $\|\hat{r}_k\| \le \mathrm{EPSOT}\left\|\hat{b}\right\|$, $\|y\|^2 = y^{\mathrm{T}}y$ なるユークリッドノルムであり, $\hat{\boldsymbol{A}}$ および $\hat{b}$ は正規化された係数行列および右辺ベクトルです.

((3)使用上の注意 a. 注意⑧参照)

EPSIN .....入力. 内部反復での解の収束判定値.

たいていの場合 10^{-3} が最適です.

EPSEP.....入力. 縮小行列の近似及び逆行列の近似の制御に関係する制御情報.

EPSEP の代表的な指定例.

EPSEP(1) = 1.0D-2

EPSEP(2) = 1.0D-2

EPSEP(3) = 0.2

EPSEP(4) = 1.0D-3

EPSEP(1) = TAU

入力. ブロック分解の後 Schur の complement として縮小された行列が疎行列であることを保つために, TAU より小さな要素は除去します.

TAU を大きくすると, 低いレベルの反復法は速く収束しますが, 前処理の品質は悪くなります.

0 ≤ TAU < 1

EPSEP(2) = TAUV

入力. 近似的逆行列の精度.

TAUV を小さくすると消去にかかる時間は増しますが, 前処理の品質は向上します. 普通は, EPSIN = TAUV が最適です.

EPSEP(3) = LAMBDA

入力. ブロック行列に対する対角しきい値.

ブロック行列の中の除去する変数は, 行の各要素の絶対値の和が, 対角要素の絶対値 × LAMBDA 以下となるように選ばれます.

LAMBDA を大きく指定すると, 除去する変数の集合は小さくなりますが, ブロック行列の近似的な逆行列を計算するコストは増加します.

LAMBDA としては, 0.2 を推奨します.

TAUV ≤ LAMBDA < 1 か LAMBDA = 0 とすべきです.

EPSEP(4) = RHO

入力. 消去の過程で, 主対角要素が小さな値である変数は, 消去する変数から

はずします.

主対角要素は, 行の要素の絶対値の和に RHO を掛けたものより小さいとき小さいと考えます.

RHO = 1.0D-3 を推奨します.

0 < RHO < 1.

X.....出力. X(1:N)に解ベクトルが格納されます.

X(N)なる 1 次元配列.

W.....作業域. W(NW)なる 1 次元配列.

NW.....入力. 作業域 W の大きさ.

NW の大きさの上限目安.

$$NW \le MAX(2 \times MAXLVL + 2, 10) \times NBAND \times MAXT + (4 \times NC + 6) \times (N + MAXT) + MAX(2 \times NC \times (N + MAXT), LR0(N) + MAX(LR0(Nf) + N + MAXT, 6 \times (N + MAXT)))$$

MAXLVL は不完全ブロック分解のレベル数.

NBAND は行列のバンド巾.

NC は行での非ゼロ要素の数の上限. 典型的には, NC = MAXNC.

Nf は最後のレベルの変数の数(典型的には, $2^{-MAXLVL} \times (N + MAXT)$).

MAXT は並列実行する最大スレッド数.

$$LR0(N) = \begin{cases} 4 \times N & : \text{ORTHOMIN法のとき} \\ (2 \times NRES + 1) \times N & : \text{GMRES法のとき} \end{cases}$$

NRES は GMRES 法で使う残差の数.

普通は LR0(Nf)は, 無視できます.

IW .....作業域. 大きさ NIW なる 1 次元配列.

NIW .....入力. 作業域 IW の大きさ.

大きさの上限目安は次のとおり.

$$NIW \le ((4 \times MAXLVL + 10) \times MAXT + 12 \times NBAND) + 3400) \times MAXT + (6 \times NC+11) \times (N + MAXT)$$

MAXLVL は不完全ブロック分解のレベル数.

NBAND は行列のバンド巾.

NC は行での非ゼロ要素の数の上限. 典型的には, NC = MAXNC.

MAXT は並列実行する最大スレッド数.

ICON.....出力. コンディションコード.

表 DM_VMLBIFE-1 参照.

表 DM_VMLBIFE-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし	―
10100	逆行列が十分な精度で計算できなかった.	処理を続ける.
10800	GMRES 法で回復可能な break down を起こした.	
20001	反復回数の上限に達した.	処理を打ち切る. 配列 X には, そのときまでに得られている近似値を出力するが, 精度は保証できない.
20003	GMRES 法で回復不可能な break down を起こした.	処理を打ち切る.
20005	ORTHOMIN 法で $\boldsymbol{p}^{\mathrm{T}} \cdot \boldsymbol{A} \cdot \boldsymbol{p} = 0$ となり回復不可能な break down を起こした.	
20006	ORTHOMIN 法で $\boldsymbol{p}^{\mathrm{T}} \cdot \boldsymbol{r} = 0$ となり回復不可能な break down を起こした.	
30000	N<1, N>K, IWIDT<1 または, ISW<1, ISW>2	
30103	ICOL の値が正しくない.	
30105	主対角要素がない.	
30210	行列の圧縮が非正値であるためできない.	
30213	非ゼロ要素のない行がある.	
30310	NIW が小さ過ぎる.	
30320	NW が小さ過ぎる.	

(3)　使用上の注意

a.　注意

①　同じ係数行列を持つ右辺の異なる多重組みの連立 1 次方程式を解く場合は, 最初の呼び出しで ISW=1 で呼び出して解き, 2 回目以降の呼び出しでは ISW=2 として呼び出すことで解けます. 最初の呼び出しで組み立てられたあらいレベルに対応する行列が 2 回目以降の呼び出しで再利用されます.

②　NW および NIW の大きさで, たいていの場合 NC は IWIDT × 1.5 で十分です. 一般的に, 作業域が足りないときは, NC を増やすことを奨めます. もし係数行列のバンド巾が非常に大きいなら, NBAND を最初に増やすことを奨めます.

③　できる限り ORTHOMIN 法を利用することを奨めます. これは, 行列が対称である必要があります.

たとえ与えられた行列が対称でなくても, 本ルーチンは反復を開始する前に計算できる変数を簡単に除去するので, 実際の反復計算で使う行列は対称となることがあります. そのため, 反復計算に使う行列が対称となる可能性があるときは, 最初に対称的正規化を行う ORTHOMIN 法を試みることを奨めます.

④ 行列が対称でない場合は, 非対称な正規化を行って GMRES 法を使うことを奨めます. たいていの場合, 外部反復で NRESOT=5 として切り捨て 20 ステップ後に再起動すれば十分です. 内部反復では, NRESIN として切り捨てる数をもっと大きくし, もっと大きなステップの後に再起動するか再起動を禁じることが必要です. NRESIN を大きくすると作業域を大きくする必要があります. このため作業域の大きさの計算式の中の NRES を大きくしなければなりません. しかし NRES は NRESOT より小さくできます.

一般に, NRESIN および NRESOT を大きくすると GMRES 法の精度はよくなりますが, 計算およびメモリ使用量も増えます.

⑤ 変数消去は次の条件の 1 つが満たされたとき終了します.

- レベルの数が MAXLVL を超えるか等しくなった.
- 最後のレベルの行列が対角行列である.
- 消去する変数の数が最後のレベルの変数の 10%以下である.

⑥ LAMBDA = 0, RHO = 0.99, TAU = 0, MAXNC = IWIDT としたとき, 古典的な Wavefront ordering による ILUM 前処理になります (“付録 1 参考文献一覧表”の[65]参照.).

LAMBDA = 0, RHO < 1, TAU > 0, MAXNC >> IWIDT としたとき, しきい値付きの ILUM 前処理となります (“付録 1 参考文献一覧表”の[64]参照.).

⑦ 問題に応じて効率のよい前処理を見つけるために, パラメタを変えて試みることを奨めます.

⑧ 各レベルの行列を $A_n = 1, \ldots, \mathrm{MAXLVL}-1$ としたとき,

$$A_n = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

$S = A_{22} - A_{21}A_{11}^{-1}A_{12}$ を Schur complement と呼びます. 消去する変数の集合を選ぶことでこのようなブロックを形成します. A_{11} の逆行列を近似します. この近似的逆行列を使って Schur complement を計算し, レベル n+1 の行列 A_{n+1} とします. Schur complement を使い A_n は次のように分解できます.

$$A_n = \begin{bmatrix} A_{11} & 0 \\ A_{21} & I \end{bmatrix} \begin{bmatrix} I & A_{11}^{-1}A_{12} \\ 0 & S \end{bmatrix}$$

A_{11}^{-1} の近似を利用して, 各レベルをこのように近似的に分解し前処理として利用します.

b. 使用例

偏微分方程式を領域 $[0, 1]^2$ で離散化して解きます.

$$-\left(\frac{\partial^2}{\partial^2 x_1}u + \frac{\partial^2}{\partial^2 x_2}u + \frac{\partial^2}{\partial^2 x_3}u\right) + t\left((x_2 - x_3)\frac{\partial u}{\partial x_1} + (x_3 - x_1)\frac{\partial u}{\partial x_2} + (x_1 - x_2)\frac{\partial u}{\partial x_3}\right) = f$$

t = 1.0, Dirichlet 境界条件 u = 0 を設定します.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT NONE
      INTEGER   MAXT,N1,N2,N3,KA,NA,L1,L2,L3,LGRW,LGIW,
     &          NLBMAX,MAXNC

      PARAMETER(MAXT=2,N1=39,N2=N1,N3=N1,
     &    L1=N1,L2=N2,L3=N3,
     &    KA=N1*N2*N3,NA=7,NLBMAX=N1*N2,
     &    MAXNC=11,
     &    LGRW=(KA+MAXT)*(6*MAXNC+11)+(85*NLBMAX+100)*MAXT,
     &    LGIW=(KA+MAXT)*(6*MAXNC+11)+(13*NLBMAX+200
     &    +61*51+13)*MAXT)

      INTEGER          NDLT(NA),IW(LGIW),
     &                 ICOL(KA,NA)
      DOUBLE PRECISION X1(L1),X2(L2),X3(L3),
     &                 A1(L1,L2,L3),A2(L1,L2,L3),A3(L1,L2,L3),
     &                 B1(L1,L2,L3),B2(L1,L2,L3),B3(L1,L2,L3),
     &                 C(L1,L2,L3),F(L1,L2,L3),
     &                 RW(LGRW)
      REAL*8 EPSIN,EPSOT,EPSEP(10)
      INTEGER INFO(40),INFOEP(10),ISW,IGUSS,IS,NBAND

      DOUBLE PRECISION MAT(KA,NA),RHS(KA),V(KA),
     &                 SOL(3*KA),RHSX(KA),RHSC(KA),TMP

      INTEGER Z1,Z2,Z3,NDIAG,N,ICON,I,Z,NC
      DOUBLE PRECISION ONE,T,HR1,HR2,HR3,HR4,HR6,HR7,HR13
      PARAMETER (ONE=1.D0)
C
C-----------------------------------------------------------------
C
C**** THESE ARE PARAMETERS OF THE TEST PDES. CHANGES OF THE
C     VALUES CAN PRODUCE DIVERGENCE IN THE ITERATIVE SOLVER.
C
      T=1
C
C****** CREATE NODE COORDINATES
```

```
C
        DO 11 Z1=1,N1
          X1(Z1)=DBLE(Z1-1)/DBLE(N1-1)
11      CONTINUE
        DO 12 Z2=1,N2
          X2(Z2)=DBLE(Z2-1)/DBLE(N2-1)
12      CONTINUE
        DO 13 Z3=1,N3
          X3(Z3)=DBLE(Z3-1)/DBLE(N3-1)
13      CONTINUE
C
C   -UX1X1-UX2X2-UX3X3+T*((X2-X3)*UX1+(X3-X1)*UX2+(X1-X2)*UX3)=F
C
C   REMARK: IF T IS TO LARGE THE PDE IS SINGULAR.
C
      DO 203 Z3=1,N3
      DO 203 Z2=1,N2
      DO 203 Z1=1,N1
      A1(Z1,Z2,Z3)=1
      A2(Z1,Z2,Z3)=1
      A3(Z1,Z2,Z3)=1
      B1(Z1,Z2,Z3)=T*(X2(Z2)-X3(Z3))
      B2(Z1,Z2,Z3)=T*(X3(Z3)-X1(Z1))
      B3(Z1,Z2,Z3)=T*(X1(Z1)-X2(Z2))
      C (Z1,Z2,Z3)=0
      HR1 = ONE-X2(Z2)
      HR2 = X2(Z2)*HR1
      HR3 = ONE-X3(Z3)
      HR4 = X3(Z3)*HR3
      HR6 = ONE-X1(Z1)
      HR7 = X1(Z1)*HR6
      HR13 = HR1*X3(Z3)*HR3
      F(Z1,Z2,Z3) = 2*HR2*HR4+2*HR7*HR4+2*HR7*HR2+
     &          T*((X2(Z2)-X3(Z3))*
     &          (HR6*X2(Z2)*HR13-X1(Z1)*X2(Z2)*HR13)+
     &          (X3(Z3)-X1(Z1))*
     &          (HR7*HR13-HR7*X2(Z2)*X3(Z3)*HR3)+
     &          (X1(Z1)-X2(Z2))*
     &          (HR7*HR2*HR3-HR7*HR2*X3(Z3)))
203       CONTINUE
C
C***** DIRICHLET CONDITIONS:
```

```
C
        DO 300  Z3=1,N3
        DO 300  Z2=1,N2
        C(1,Z2,Z3)=1
        B1(1,Z2,Z3)=0
        B2(1,Z2,Z3)=0
        B3(1,Z2,Z3)=0
        F(1,Z2,Z3)=0
        C(N1,Z2,Z3)=1
        B1(N1,Z2,Z3)=0
        B2(N1,Z2,Z3)=0
        B3(N1,Z2,Z3)=0
        F(N1,Z2,Z3)=0
        IF (Z2.EQ.1) THEN
         DO 325 Z1=1,N1
         C(Z1,1,Z3)=1
         B1(Z1,1,Z3)=0
         B2(Z1,1,Z3)=0
         B3(Z1,1,Z3)=0
         F(Z1,1,Z3)=0
325      CONTINUE
        ELSEIF (Z2.EQ.N2) THEN
         DO 326 Z1=1,N1
         C(Z1,N2,Z3)=1
         B1(Z1,N2,Z3)=0
         B2(Z1,N2,Z3)=0
         B3(Z1,N2,Z3)=0
         F(Z1,N2,Z3)=0
326      CONTINUE
        ENDIF
        IF (Z3.EQ.1) THEN
         DO 335 Z1=1,N1
         C(Z1,Z2,1)=1
         B1(Z1,Z2,1)=0
         B2(Z1,Z2,1)=0
         B3(Z1,Z2,1)=0
         F(Z1,Z2,1)=0
335      CONTINUE
        ELSEIF (Z3.EQ.N3) THEN
         DO 336 Z1=1,N1
         C(Z1,Z2,N3)=1
         B1(Z1,Z2,N3)=0
```

```
         B2(Z1,Z2,N3)=0
         B3(Z1,Z2,N3)=0
         F(Z1,Z2,N3)=0
336      CONTINUE
        ENDIF
300     CONTINUE
C
      N=N1*N2*N3
      CALL DM_VPDE3D(A1,L1,L2,N1,N2,N3,
     $               A2,A3,X1,X2,X3,B1,B2,
     $               B3,C,F,MAT,KA,NA,N,
     $               NDIAG,NDLT,RHS,ICON)
      PRINT*,'ICON OF DM_VPDE3D = ',ICON
      IF (ICON.GT.29999) STOP
C
C
      DO Z=1,N
      RHSX(Z)=RHS(Z)
      ENDDO
      NBAND=0
      DO I=1,NDIAG
      NBAND=MAX(NBAND,ABS(NDLT(I)))
      ENDDO
C
C
C**** CHANGE TO ELLPACK FORMAT:
C
      NC=NDIAG
      DO I=1,NC
      DO Z=1,KA
      IS=Z+NDLT(I)
      ICOL(Z,I)=IS
      ENDDO
      ENDDO
C
C****** CALL THE ITERATIVE SOLVER:
C
      ISW=1
      IGUSS=0
      EPSOT=1.D-6
      EPSIN=1.D-3
      INFO(1)=10
```

```
      INFO(2)=MAXT*100
      INFO(3)=1
      INFO(5)=2
      INFO(6)=5000
      INFO(7)=5
      INFO(8)=20
      INFO(11)=5000
      INFO(10)=2
      INFO(12)=20
      INFO(13)=0
      INFOEP(1)=1
      INFOEP(2)=5
      INFOEP(3)=14
      EPSEP(1)=1.D-2
      EPSEP(2)=EPSEP(1)
      EPSEP(3)=0.2
      EPSEP(4)=1.D-3
      CALL DM_VMLBIFE(MAT,KA,NC,N,ICOL,
     &              RHS,ISW,IGUSS,INFO,INFOEP,EPSOT,EPSIN,
     &              EPSEP,V,RW,LGRW,IW,LGIW,ICON)
      PRINT*,'ICON OF DM_VEBIFE = ',ICON
      IF (ICON.GT.29999) STOP
C
      DO I=1,NBAND
      SOL(I)=0.0D0
      SOL(NBAND+N+I)=0.0D0
      ENDDO
      DO Z=1,N
      SOL(NBAND+Z)=V(Z)
      ENDDO
      CALL DM_VMVSD(MAT,KA,NDIAG,N,NDLT,NBAND,SOL,RHSC,ICON)
      TMP=0
      DO Z=1,N
      TMP=MAX(TMP,ABS((RHSX(Z)-RHSC(Z))/(RHSX(Z)+1.0)))
      ENDDO
C
      PRINT*,' ERROR = ',TMP
C
      STOP
      END
```

(4) 手法概要

最初に主対角要素以外がゼロである行を除去します (ディリクレ条件から典型的に生じる.). この結果, 行列が対称行列となる場合があります.

行列は, 各行の和が 1 のオーダーであり, かつ主対角要素が非負となるように正規化されます. 正規化された連立 1 次方程式は ORTHOMIN 法, GMRES 法で解かれます.

前処理は, 近似的な Schur complements を用いたマルチレベル構造の不完全ブロック分解に基づいています. 同時に消去される変数の集合は, 行列の大きな要素からなるグラフの最大独立集合を求めることで定義されます. Schur complement で行列のスパース性を保つために小さな要素は除去されます.

最後のレベルの連立 1 次方程式は, 正規化されて ORTHOMIN 法または GMRES を用いて反復的に解かれます.

DM_VMVSCC

実スパース行列と実ベクトルの積 (圧縮列格納法)
CALL DM_VMVSCC(A, NZ, NROW, NFCNZ, N, X, Y, W, IW, ICON)

(1)　機能

$n \times n$ のスパース行列とベクトルの積を計算します.

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$$

スパース行列 $\boldsymbol{A}$ は, 圧縮列格納法(Compressed Column Storage)で格納されます.
ベクトル $\boldsymbol{x}$ および $\boldsymbol{y}$ は n 次元ベクトルです.

(2)　パラメタ

A.....入力. 係数行列の非零要素を格納します.
A(1:NZ)にスパース行列の非零要素が格納されます.
圧縮列格納法については, 図 DM_VMVSCC-1 参照.
A(NZ)なる 1 次元配列.

NZ .....入力. 係数行列 $\boldsymbol{A}$ の非零要素の総数.

NROW .....入力. 圧縮列格納法で使用される行指標で, A に格納される要素が何番目の行ベクトルに属するかを示します.
NROW(NZ)なる 1 次元配列.

NFCNZ .....入力. 行列 $\boldsymbol{A}$ の各列の非零要素を列方向に圧縮して順次配列 A に格納するとき, 対応する列の最初の非零要素が格納される位置を表します.
NFCNZ(N+1) = NZ+1.
NFCNZ(N+1)なる 1 次元配列.

N.....入力. 行列 $\boldsymbol{A}$ の次数 n.

X.....入力. ベクトル x を X(1:N)に格納します.
X(N)なる 1 次元配列.

Y.....出力. 行列とベクトルの積の結果が Y(1:N)に格納されます.
Y(N)なる 1 次元配列.

W.....作業域. W(NZ)なる 1 次元配列.

IW .....作業域. IW(2, NZ)なる 2 次元配列.

ICON.....出力. コンディションコード.
表 DM_VMVSCC-1 参照.

表 DM_VMVSCC-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし.	ー
30000	N<1, NZ<0, NFCNZ(N+1) ≠ NZ+1	処理を打ち切る.

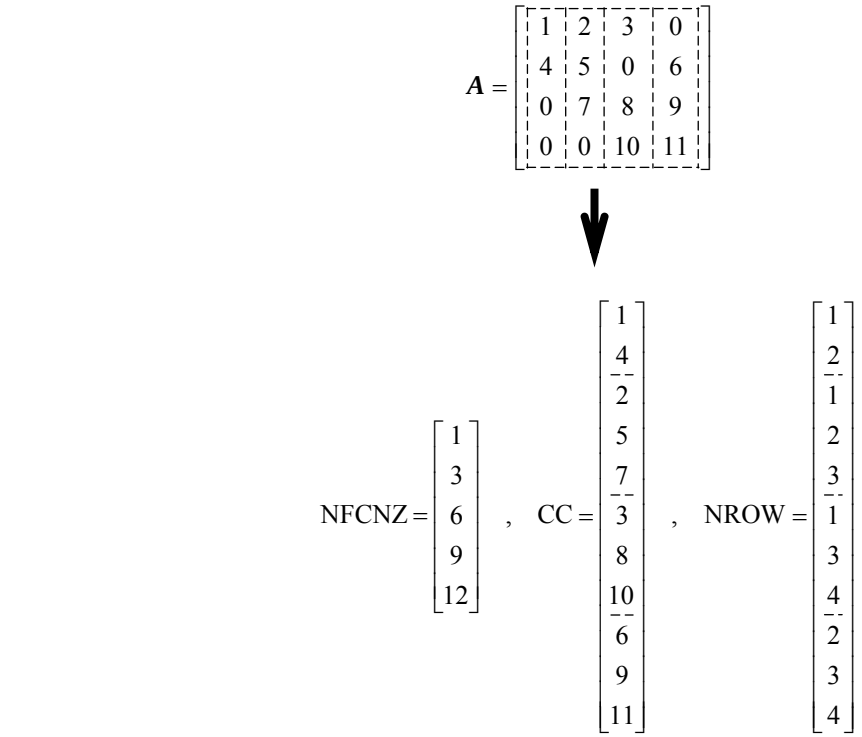


図 DM_VMVSCC-1

行列 ***A*** を圧縮列格納法で格納する方法を示します.

1 次元配列 CC に行列 ***A*** の列の非零要素を圧縮して順次格納します.

NFCNZ(*i*)には, 行列 ***A*** の *i* 番目の列の最初の非零要素を 1 次元配列 CC に格納する位置を設定します. 行列 ***A*** の次数を *n* として, NFCNZ(*n*+1) = *nz* + 1 と設定します. *nz* は, 行列 ***A*** の非零要素の総数です. 配列 CC の *i* 番目の要素 CC(*i*)に格納された行列 ***A*** の非零要素が何番目の行にあるかを NROW(*i*)に設定します.

(3) 使用上の注意

a. 使用例

スパース行列とベクトルの積を計算します.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT  REAL*8 (A-H,O-Z)
      PARAMETER (NORD=60,NX = NORD,NY =NORD ,NZ = NORD,
     $      N = NX*NY*NZ)
      PARAMETER (K = N+1)
      PARAMETER (NDIAG = 7)

      DIMENSION NOFST(NDIAG)
      DIMENSION DIAG(K,NDIAG)
```

```
      DIMENSION A(K*NDIAG),NROW(K*NDIAG),NFCNZ(N+1),
     $          W(K*NDIAG),IW(2,K*NDIAG)
      DIMENSION X(N),B(N),Y(N)

      X(1:N)=1.0D0

      NOFST(1)=-NX*NY
      NOFST(2)=-NX
      NOFST(3)=-1
      NOFST(4)=0
      NOFST(5)=1
      NOFST(6)=NX
      NOFST(7)=NX*NY

      DO I=1,NDIAG
C
      IF(NOFST(I).LT.0)THEN
      NBASE=-NOFST(I)
      LENGTH=N-NBASE
      DIAG(1:LENGTH,I)=DBLE(I)
      ELSE
      NBASE=NOFST(I)
      LENGTH=N-NBASE
      DIAG(NBASE+1:N,I)=DBLE(I)
      ENDIF
C
      ENDDO
C
      NUMNZ=1
      DO J=1,N
      NTOPCFG=1
      DO I=NDIAG,1,-1
C
      IF(DIAG(J,I).NE.0.0D0)THEN
C
      NCOL=J-NOFST(I)
      A(NUMNZ)=DIAG(J,I)
      NROW(NUMNZ)=NCOL
C
      IF(NTOPCFG.EQ.1)THEN
      NFCNZ(J)=NUMNZ
      NTOPCFG=0
```

```
      ENDIF
C
      NUMNZ=NUMNZ+1
      ENDIF
C
      ENDDO
      ENDDO
      NFCNZ(N+1)=NUMNZ
      NNZ=NUMNZ-1

      CALL DM_VMVSCC(A,NNZ,NROW,NFCNZ,N,X,
     $             Y,W,IW,ICON)
C
      B(1:N)=0.0D0
      DO I=1,N
      NS=NFCNZ(I)
      NE=NFCNZ(I+1)-1
      DO J=NS,NE
      II=NROW(J)
      B(II)=B(II)+A(J)*X(I)
      ENDDO
      ENDDO
C
      S=0.0D0
      DO I=1,N
      S=MAX(S,ABS(Y(I)-B(I)))
      ENDDO
C
      PRINT*,'ERROR=',S

      STOP
      END
```

DM_VMVSCCC

複素スパース行列と複素ベクトルの積 (圧縮列格納法)
CALL DM_VMVSCCC(ZA, NZ, NROW, NFCNZ, N, ZX, ZY, ZW, IW, ICON)

(4)　機能

$n \times n$ の複素スパース行列と複素ベクトルの積を計算します.

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$$

スパース行列 $\boldsymbol{A}$ は, 圧縮列格納法(Compressed Column Storage)で格納されます.
ベクトル $\boldsymbol{x}$ および $\boldsymbol{y}$ は n 次元ベクトルです.

(5)　パラメタ

ZA .....入力. 係数行列の非零要素を格納します.
ZA(1:NZ)にスパース行列の非零要素が格納されます.
圧縮列格納法については, 図 DM_VMVSCC-1 参照. 図 DM_VMSCC-1 の実数型の配列 CC を複素数型の配列に変えたものを利用します.
ZA(NZ)なる倍精度複素数型の 1 次元配列.

NZ .....入力. 係数行列 $\boldsymbol{A}$ の非零要素の総数.

NROW .....入力. 圧縮列格納法で使用される行指標で, ZA に格納される要素が何番目の行ベクトルに属するかを示します.
NROW(NZ)なる 1 次元配列.

NFCNZ .....入力. 行列 $\boldsymbol{A}$ の各列の非零要素を列方向に圧縮して順次配列 ZA に格納するとき, 対応する列の最初の非零要素が格納される位置を表します.
NFCNZ(N+1) = NZ+1.
NFCNZ(N+1)なる 1 次元配列.

N.....入力. 行列 $\boldsymbol{A}$ の次数 n.

ZX .....入力. ベクトル x を ZX(1:N)に格納します.
ZX(N)なる倍精度複素数型の 1 次元配列.

ZY .....出力. 行列とベクトルの積の結果が ZY(1:N)に格納されます.
ZY(N)なる倍精度複素数型の 1 次元配列.

ZW .....作業域. ZW(NZ)なる倍精度複素数型の 1 次元配列.

IW .....作業域. IW(2, NZ)なる 2 次元配列.

ICON.....出力. コンディションコード.
表 DM_VMVSCCC-1 参照.

表 DM_VMVSCCC-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし.	ー
30000	N<1, NZ<0, NFCNZ(N+1) ≠ NZ+1	処理を打ち切る.

(6) 使用上の注意

b. 使用例

複素スパース行列と複素ベクトルの積を計算します.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT  REAL*8 (A-H,O-Z)
      PARAMETER (NORD=60,NX = NORD,NY =NORD ,NZ = NORD,
     $      N = NX*NY*NZ)
      PARAMETER (K = N+1)
      PARAMETER (NDIAG = 7)

      DIMENSION NOFST(NDIAG)
      COMPLEX*16 ZDIAG(K,NDIAG),ZA(K*NDIAG),ZW(K*NDIAG)
      DIMENSION NROW(K*NDIAG),NFCNZ(N+1),
     $          IW(2,K*NDIAG)
      COMPLEX*16 ZX(N),ZB(N),ZY(N)

      ZX(1:N)=(1.0D0,0.0D0)

      NOFST(1)=-NX*NY
      NOFST(2)=-NX
      NOFST(3)=-1
      NOFST(4)=0
      NOFST(5)=1
      NOFST(6)=NX
      NOFST(7)=NX*NY

      DO I=1,NDIAG
C
      IF(NOFST(I).LT.0)THEN
      NBASE=-NOFST(I)
      LENGTH=N-NBASE
      ZDIAG(1:LENGTH,I)=DCMPLX(DBLE(I),0.0D0)
      ELSE
      NBASE=NOFST(I)
      LENGTH=N-NBASE
      ZDIAG(NBASE+1:N,I)=DCMPLX(DBLE(I),0.0D0)
      ENDIF
```

```
C
      ENDDO
C
      NUMNZ=1
      DO J=1,N
      NTOPCFG=1
      DO I=NDIAG,1,-1
C
      IF(ZDIAG(J,I).NE.(0.0D0,0.0D0))THEN
C
      NCOL=J-NOFST(I)
      ZA(NUMNZ)=ZDIAG(J,I)
      NROW(NUMNZ)=NCOL
C
      IF(NTOPCFG.EQ.1)THEN
      NFCNZ(J)=NUMNZ
      NTOPCFG=0
      ENDIF
C
      NUMNZ=NUMNZ+1
      ENDIF
C
      ENDDO
      ENDDO
      NFCNZ(N+1)=NUMNZ
      NNZ=NUMNZ-1

      CALL DM_VMVSCCC(ZA,NNZ,NROW,NFCNZ,N,ZX,
     $             ZY,ZW,IW,ICON)
C
      ZB(1:N)=(0.0D0,0.0D0)
      DO I=1,N
      NS=NFCNZ(I)
      NE=NFCNZ(I+1)-1
      DO J=NS,NE
      II=NROW(J)
      ZB(II)=ZB(II)+ZA(J)*ZX(I)
      ENDDO
      ENDDO
C
      S=0.0D0
      DO I=1,N
```

```
      S=MAX(S,CDABS(ZY(I)-ZB(I)))
      ENDDO
C
      PRINT*,'ERROR=',S

      STOP
      END
```

DM_VMVSD

実スパース行列と実ベクトルの積 (対角形式格納法)
CALL DM_VMVSD(A, K, NDIAG, N, NOFST, NLB, X, Y, ICON)

(1)　機能

$n \times n$ のスパース行列とベクトルの積を計算します.

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$$

スパース行列 $\boldsymbol{A}$ は, 対角形式の格納法で格納されます.
ベクトル $\boldsymbol{x}$ および $\boldsymbol{y}$ は n 次元ベクトルです.

(2)　パラメタ

A.....入力. 係数行列の非零要素を格納します.
A(1:N, 1:NDIAG)にスパース行列の非零要素が格納されます.
対角形式の格納方法については, “SSL II 拡張機能使用手引書 II 第 I 部 3.2.1.1 一般スパース行列の格納方法 b. 一般スパース行列の対角形式の格納方法” を参照してください.
A(K, NDIAG)なる 2 次元配列.

K.....入力. 配列 A の 1 次元目の大きさ($\geq n$).

NDIAG.....入力. 配列に格納される係数行列の非零要素を含む対角ベクトル列の総数.
配列 A の 2 次元目の大きさ.

N.....入力. 行列 $\boldsymbol{A}$ の次数 n.

NOFST.....入力. 配列 A に格納される対角ベクトルに対応した主対角ベクトルからの距離が格納されます. 上対角ベクトル列は正, 下対角ベクトル列は負の値で表します.
NOFST(NDIAG)なる 1 次元配列.

NLB .....入力. 行列 $\boldsymbol{A}$ 下バンド幅.

X.....入力. ベクトル x を X(NLB+1:NLB+N)に格納します.
X($n+nlb+nub$)なる 1 次元配列.
nlb:下バンド幅, *nub*:上バンド幅.

Y.....出力. 行列とベクトルの積の結果が Y(1:N)に格納されます.
Y(N)なる 1 次元配列.

ICON.....出力. コンディションコード.
表 DM_VMVSD-1 参照.

表 DM_VMVSD-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし.	ー
30000	N<1, NDIAG<1, K<N, NLB ≠ max(-NOFST(I)), |NOFST(I)| > N-1	処理を打ち切る.

(3)　使用上の注意

a.　注意

①　対角形式を使う上での注意

係数行列 ***A*** の外側の対角ベクトルの要素は零を設定する必要があります.

対角ベクトル列を配列 A に格納する順序に制限は特にありません.

この方法の利点は, 行列ベクトル積が間接指標を使わずに計算できる点です.

しかし, 対角構造を持たない行列を効率よく格納できないことが欠点です.

b.　使用例

スパース行列とベクトルの積を計算します.

スパース行列は, INIT_MAT_DIAG(DM_VBCSD の b. 使用例参照.)で生成します.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT  REAL*8 (A-H,O-Z)
      PARAMETER (NORD=60,NX = NORD,NY =NORD ,NZ = NORD,
     $      N = NX*NY*NZ)
      PARAMETER (K = N+1)
      PARAMETER (NDIAG = 7)
      PARAMETER(NVW=3*K)
      DIMENSION NOFST(NDIAG)
      DIMENSION A(K,NDIAG)
      DIMENSION Y(N),B(N)
      DIMENSION X(NVW)

      VA1 = 3D0
      VA2 = 1D0/3D0
      VA3 = 5D0
      VC = 1.0
      XL = 1.0
      YL = 1.0
      ZL = 1.0
```

```
      CALL INIT_MAT_DIAG(VA1,VA2,VA3,VC,A,NOFST
     &          ,NX,NY,NZ,XL,YL,ZL,NDIAG,N,K)

!$OMP PARALLEL DO DEFAULT(PRIVATE) SHARED(A,B)
      DO I=1,N
      B(I)=0.0D0
      DO J=1,NDIAG
      B(I)=B(I)+A(I,J)
      ENDDO
      ENDDO
!$OMP END PARALLEL DO

      NBANDL=0
      NBANDR=0
      DO I=1,NDIAG
      IF(NOFST(I).LT.0)THEN
      NBANDL=MAX(NBANDL,-NOFST(I))
      ELSE
      NBANDR=MAX(NBANDR,NOFST(I))
      ENDIF
      ENDDO

      X(1+NBANDL:N+NBANDL) = 1.0D0
      CALL DM_VMVSD(A,K,NDIAG,N,NOFST,NBANDL,X,Y,ICON)

      ERROR=0.0D0
!$OMP PARALLEL DO DEFAULT(PRIVATE) SHARED(Y,B)
!$OMP+         REDUCTION(MAX:ERROR)
      DO I=1,N
      ERROR=MAX(ERROR,DABS(Y(I)-B(I)))
      ENDDO
!$OMP END PARALLEL DO

      PRINT*,'ERROR = ',ERROR

      STOP
      END
```

DM_VMVSE

実スパース行列と実ベクトルの積 (ELLPACK 形式格納法)
CALL DM_VMVSE(A, K, NW, N, ICOL, X, Y, ICON)

(1) 機能

$n \times n$ のスパース行列とベクトルの積を計算します.

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$$

$n \times n$ の係数行列は, 2 つの配列を使用する, ELLPACK 形式の格納法で格納されます.

$\boldsymbol{y}$ および $\boldsymbol{x}$ は n 次元ベクトルです.

(2) パラメタ

A.....入力. 係数行列の非零要素を A(1:N, 1:NW)に格納します.
ELLPACK 形式の格納方法については, “SSL II 拡張機能使用手引書 II 第 I 部 概説 3.2.1.1 一般スパース行列の格納方法” を参照してください.
A(K, NW)なる 2 次元配列.

K.....入力. 配列 A の 1 次元目の大きさ($\geq n$).

NW.....入力. 配列 A の 2 次元目の大きさ. 配列 A に格納される行列 $\boldsymbol{A}$ の非零要素の行ベクトル方向の最大個数.

N.....入力. 行列 $\boldsymbol{A}$ の次数.

ICOL.....入力. ELLPACK 形式で使用される列指標で, A に格納される要素が何番目の列ベクトルに属すかを示します.
ICOL(K, NW)なる 2 次元配列.

X.....入力. ベクトル $\boldsymbol{x}$ を X(1:N)に格納します.
X(N)なる 1 次元配列.

Y.....出力. 行列とベクトルの積の結果が Y(1:N)に格納されます.
Y(N)なる 1 次元配列.

ICON.....出力. コンディションコード.
表 DM_VMVSE-1 参照.

表 DM_VMVSE-1 コンディションコード

コード	意　味	処理内容
0	エラーなし.	ー
30000	N<1, NW<1, K<N	処理を打ち切る.

(3)　使用上の注意

a.　注意

①　ELLPACK 格納形式を使う上での注意

ELLPACK 形式でデータを格納する前に, 配列 A および ICOL をおのおの零, 行ベクトルの番号で初期化することを勧めます.

b.　使用例

スパース行列とベクトルの積を計算します.

スパース行列は, INIT_MAT_ELL(DM_VBCSE の b. 使用例参照.)で生成されます.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT  REAL*8 (A-H,O-Z)
      PARAMETER (NORD=60,NX =NORD ,NY = NORD,NZ = NORD,
     &           N = NX*NY*NZ)
      PARAMETER (K = N+1)
      PARAMETER (IWIDT = 7)
      DIMENSION ICOL(K,IWIDT)
      DIMENSION A(K,IWIDT)
      DIMENSION X(N),B(N),Y(N)

      VA1 = 3D0
      VA2 = 1D0/3D0
      VA3 = 5D0
      VC = 1.0
      XL = 1.0
      YL = 1.0
      ZL = 1.0
      CALL INIT_MAT_ELL(VA1,VA2,VA3,VC,A,ICOL
     &          ,NX,NY,NZ,XL,YL,ZL,IWIDT,N,K)

!$OMP PARALLEL DO DEFAULT(PRIVATE) SHARED(A,B)
      DO I=1,N
      B(I)=0.0D0
      DO J=1,IWIDT
      B(I)=B(I)+A(I,J)
      ENDDO
      ENDDO
!$OMP END PARALLEL DO
```

```
      Y(1:N)=1.0D0
      CALL DM_VMVSE(A,K,IWIDT,N,ICOL,Y,X,ICON2)

      ERROR=0.0D0
!$OMP PARALLEL DO DEFAULT(PRIVATE) SHARED(X,B)
!$OMP+         REDUCTION(MAX:ERROR)
      DO I=1,N
      ERROR=MAX(ERROR,DABS(X(I)-B(I)))
      ENDDO
!$OMP END PARALLEL DO

      PRINT*,'ERROR = ',ERROR

      STOP
      END
```

DM_VPDE2D

2 次元 2 階偏微分方程式の有限差分法による離散化によるスパース行列の連立 1 次方程式の生成
CALL DM_VPDE2D(A1, L1, N1, N2, A2, X1, X2, B1, B2, C, F, A, K, NA, N, NDIAG, NOFST, R, ICON)

(1)　機能

本ルーチンは, 長方形領域 B で以下の境界条件を満たす 2 次元 2 階偏微分方程式を, 有限差分法によって離散化して生成される連立 1 次方程式の係数行列と定数ベクトルを生成します.

$$-\left(\frac{\partial}{\partial x_1}a_1\frac{\partial u}{\partial x_1}+\frac{\partial}{\partial x_2}a_2\frac{\partial u}{\partial x_2}\right)+b_1\frac{\partial u}{\partial x_1}+b_2\frac{\partial u}{\partial x_2}+cu=f \tag{1.1}$$

長方形領域 B で偏微分方程式(1.1)を満たし, 長方形領域 B の境界で境界条件(1.2)を満たします.

$$\beta_1\frac{\partial u}{\partial x_1}+\beta_2\frac{\partial u}{\partial x_2}+\gamma u=\phi \tag{1.2}$$

偏微分方程式に現れる関数 a_1, a_2, b_1, b_2, c, f は長方形領域 B で定義されます.
境界条件に現れる $\beta_1, \beta_2, \gamma, \phi$ は長方形領域 B の境界で定義された関数です.
離散化に使う N1 × N2 の格子点は $x_{i,j}$ = (X1(i), X2(j))で定義されます.

i = 1, ... , N1,　j = 1, ... , N2

B = [X(1), X1(N1)] × [X2(1), X(N2)].

偏微分方程式に現れる関数および境界条件は領域の格子点での値で定義されます.
離散化によって生成された係数行列 ***A*** は, 対角形式格納法で格納されます.

(2)　パラメタ

A1.....入力. 係数 $a_1(x_{i,j})$を設定します.
　　A1(L1, N2)なる 2 次元配列.

L1.....入力. 配列 A1, A2, B1, B2, C, F の 1 次元目の大きさ (L1 ≥ N1).

N1.....入力. x_1 方向の格子点の数(N1 > 2).

N2.....入力. x_2 方向の格子点の数(N2 > 2).

A2.....入力. 係数 $a_2(x_{i,j})$を設定します.
　　A2 (L1, N2)なる 2 次元配列.

X1.....入力. X1(1:N1)に格子点での x_1 座標の値を格納します. 座標の値は昇順に格納します.
　　X1(i) < X1(i+1) , i=1, ... , N1 - 1
　　X1(N1)なる 1 次元配列.

X2.....入力. X2(1:N2)に格子点での x_2 座標の値を格納します. 座標の値は昇順に格納します.

X2(i) < X2(i+1) , i=1, ... , N2 - 1

X2(N2)なる 1 次元配列.

B1.....入力. 係数 $b_1(x_{i,j})$, および境界条件 β_1 を以下のように設定します.

$$\mathrm{B1}(i,j)=\begin{cases}\beta_1(x_{1,j}) & i=1\\ \beta_1(x_{\mathrm{N1},j}) & i=\mathrm{N1}\\ \beta_1(x_{i,1}) & j=1\\ \beta_1(x_{i,\mathrm{N2}}) & j=\mathrm{N2}\\ b_1(x_{i,j}) & \text{上記以外}\end{cases}$$

B1(L1, N2)なる 2 次元配列.

B2.....入力. 係数 $b_2(x_{i,j})$, および境界条件 β_2 を以下のように設定します.

$$\mathrm{B2}(i,j)=\begin{cases}\beta_2(x_{1,j}) & i=1\\ \beta_2(x_{\mathrm{N1},j}) & i=\mathrm{N1}\\ \beta_2(x_{i,1}) & j=1\\ \beta_2(x_{i,\mathrm{N2}}) & j=\mathrm{N2}\\ b_2(x_{i,j}) & \text{上記以外}\end{cases}$$

B2(L1, N2)なる 2 次元配列.

C.....入力. 係数 $c(x_{i,j})$, および境界条件 γ を以下のように設定します.

$$\mathrm{C}(i,j)=\begin{cases}\gamma(x_{1,j}) & i=1\\ \gamma(x_{\mathrm{N1},j}) & i=\mathrm{N1}\\ \gamma(x_{i,1}) & j=1\\ \gamma(x_{i,\mathrm{N2}}) & j=\mathrm{N2}\\ c(x_{i,j}) & \text{上記以外}\end{cases}$$

C(L1, N2)なる 2 次元配列.

F .....入力. 係数 $f(x_{i,j})$, および境界条件 ϕ を以下のように設定します.

$$\mathrm{F}(i,j)=\begin{cases}\phi(x_{1,j}) & i=1\\ \phi(x_{\mathrm{N1},j}) & i=\mathrm{N1}\\ \phi(x_{i,1}) & j=1\\ \phi(x_{i,\mathrm{N2}}) & j=\mathrm{N2}\\ f(x_{i,j}) & \text{上記以外}\end{cases}$$

F(L1, N2)なる 2 次元配列.

A.....出力. 偏微分方程式を離散化して生成された係数行列の非零要素が格納されます.

A(1:N, NDIAG)に係数行列が格納されます.

A(K, NA)なる 2 次元配列.

対角形式の格納方法については, “SSL II 拡張機能使用手引書 II 第 I 部 概説 3.2.1.1 一般スパース行列の格納方法 b. 一般スパース行列の対角形式の格納方法” を参照してください.

K.....入力. 配列 A の 1 次元目の大きさ(≥ N = N1 × N2).

NA.....入力. 配列 A の 2 次元目の大きさ(≥ NDIAG = 5).

N.....入力. 行列 $\boldsymbol{A}$ の次数 n(= N1 × N2).

NDIAG.....出力. 係数行列 $\boldsymbol{A}$ の非零要素を含む対角ベクトル列の総数(= 5).

NOFST.....出力. 配列 A に格納される対角ベクトルに対応した主対角ベクトルからの距離を格納します. 上対角ベクトル列は正, 下対角ベクトル列は負の値で表します.
NOFST(NDIAG)なる 1 次元配列.

R.....出力. 偏微分方程式を離散化して生成された連立 1 次方程式の右辺定数ベクトルが R(1:N)に格納されます.
R(K)なる 1 次元配列.

ICON.....出力. コンディションコード.
表 DM_VPDE2D-1 参照.

表 DM_VPDE2D-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし	―
30000	L1<N1, N1<3, N2<3, NA<5, K<N1 × N2.	処理を打ち切る.
30001	格子点の座標が昇順に格納されていない.	

(3)　使用上の注意

a.　注意

①　線型系や固有値問題の解法によって計算された格子点での解の値は, 格子点の配置や数に大きく依存します.

②　隣接する格子点の間の距離の変化は緩やかでなければなりません. 例えば x_1 方向の条件として以下を満たすべきです.

$$0.5 \le \frac{\mathrm{X1}(i)-\mathrm{X1}(i-1)}{\mathrm{X1}(i+1)-\mathrm{X1}(i)} \le 2 \quad ,i=2,...,\mathrm{N1}-1$$

(x_2 方向も同様です.) この条件が満たされていないと係数行列は悪条件となる可能性があります. 係数行列の条件数は格子ばかりでなく係数関数によっても決まることに注意してください.

b.　使用例

領域は channel$[-1, 1]^2$ とします. 偏微分方程式は

$$-\left(\frac{\partial^2 u}{\partial^2 x_1}+\frac{\partial^2 u}{\partial^2 x_2}\right)+v_1\frac{\partial u}{\partial x_1}+v_2\frac{\partial u}{\partial x_2}=0$$

回転している速度場により生じる channel を伝わる量 u の拡散をモデル化したものです.

$$v=(v_1,v_2)=v_0\cdot\left(\frac{x_2}{\sqrt{x_1^2+x_2^2}},\frac{-x_1}{\sqrt{x_1^2+x_2^2}}\right)$$

ここで, v_0 は実定数(例えば, v_0=1)です. 境界条件は次のように設定されます.

$u = 0$　　$x_2 = -1$
$u = 1$　　$x_2 = 1$
$\frac{\partial u}{\partial n} = 0$　上記以外

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT NONE

      INTEGER   N1,N2,KA,NA,L1,L2

      PARAMETER (N1=49,N2=49, L1=N1,L2=N2, KA=N1*N2,NA=5)

      INTEGER   NOFST(NA)
      DOUBLE PRECISION V0,X1(L1),X2(L2),
     &         A1(L1,L2),A2(L1,L2),B1(L1,L2),B2(L1,L2),
     &         C(L1,L2),F(L1,L2),A(KA,NA),R(KA)

      INTEGER Z1,Z2,ICON,I,J,N,NDIAG

      V0=1.
C
C create grid nodes nodes:
C
      DO 11 Z1=1,N1
        X1(Z1)=(2*DBLE(Z1-1)/DBLE(N1-1)-1.)
11    CONTINUE
      DO 13 Z2=1,N2
        X2(Z2)=(2*DBLE(Z2-1)/DBLE(N2-1)-1.)
13    CONTINUE
C
C coefficient functions:
C
      DO 2000 Z2=1,N2

       DO 20 Z1=1,N1
        A1(Z1,Z2)=1
        A2(Z1,Z2)=1
20     CONTINUE
       DO 21 Z1=2,N1-1
```

```
          B1(Z1,Z2)= V0*X2(Z2)/SQRT(X1(Z1)**2+X2(Z2)**2+1.D-10)
          B2(Z1,Z2)=-V0*X1(Z1)/SQRT(X1(Z1)**2+X2(Z2)**2+1.D-10)
          C (Z1,Z2)=0
          F (Z1,Z2)=0
21    CONTINUE
C
C boundary conditions at faces X1=-1 and X1=1:
C
      B1(1,Z2)=-1
      B2(1,Z2)=0
      C (1,Z2)=0
      F (1,Z2)=0

      B1(N1,Z2)=1
      B2(N1,Z2)=0
      C (N1,Z2)=0
      F (N1,Z2)=0
C
C boundary conditions at faces X2=-1 and X2=1:
C
      IF (Z2.EQ.1) THEN
        DO 103 Z1=1,N1
          B1(Z1,1)=0
          B2(Z1,1)=0
          C (Z1,1)=1
          F (Z1,1)=0
103     CONTINUE
      ELSE IF (Z2.EQ.N2) THEN
        DO 113 Z1=1,N1
          B1(Z1,N2)=0
          B2(Z1,N2)=0
          C (Z1,N2)=1
          F (Z1,N2)=1
113     CONTINUE
      END IF
2000  CONTINUE

C
C build the linear system:
C
      N=N1*N2
      CALL DM_VPDE2D(A1,L1,N1,N2,A2,X1,X2, B1,B2,C,F,A,KA,NA,N,
```

```
     &               NDIAG,NOFST,R,ICON)
      PRINT*,'ICON of DM_VPDE2D =',ICON
      IF (ICON.GT.29999) GOTO 9999
C
C write the matrix to a file:
C
      WRITE (6,'(3D23.16)') ((A(I,J),I=1,N,100),J=1,NDIAG)
      WRITE (6,'(3I10)') (NOFST(J),J=1,NDIAG)
      WRITE (6,'(3D23.16)') (R(I),I=1,N,100)
9999  CONTINUE
      END
```

(4) 手法概要

拡散項 $-\nabla a \nabla$ は x_1 および x_2 方向に対して 2 次の中心差分法によって近似されます. 移流項 $b\nabla$ は, 1 次の風上差分法によって近似されます. 詳細は“付録 1 参考文献一覧表”の[75]を参照してください.

DM_VPDE3D

3 次元 2 階偏微分方程式の有限差分法による離散化によるスパース行列の連立 1 次方程式の生成
CALL DM_VPDE3D(A1, L1, L2, N1, N2, N3, A2, A3, X1, X2, X3, B1, B2, B3, C, F, A, KA, NA, N, NDIAG, NOFST, R, ICON)

(1)　機能

本ルーチンは長方形領域 B で以下の境界条件を満たす 3 次元 2 階偏微分方程式を有限差分法によって離散化して生成される連立 1 次方程式の係数行列と定数ベクトルを生成します.

$$-\left(\frac{\partial}{\partial x_1}a_1\frac{\partial u}{\partial x_1}+\frac{\partial}{\partial x_2}a_2\frac{\partial u}{\partial x_2}+\frac{\partial}{\partial x_3}a_3\frac{\partial u}{\partial x_3}\right)+b_1\frac{\partial u}{\partial x_1}+b_2\frac{\partial u}{\partial x_2}+b_3\frac{\partial u}{\partial x_3}+cu=f \tag{1.1}$$

長方形領域 B で偏微分方程式(1.1)を満たし, 長方形領域 B の境界で境界条件(1.2)を満たします.

$$\beta_1\frac{\partial u}{\partial x_1}+\beta_2\frac{\partial u}{\partial x_2}+\beta_3\frac{\partial u}{\partial x_3}+\gamma u=\phi \tag{1.2}$$

偏微分方程式に現れる関数 a_1, a_2, a_3, b_1, b_2, b_3, c, f は長方形領域 B で定義されます.
境界条件に現れる $\beta_1, \beta_2, \beta_3, \gamma, \phi$ は長方形領域 B の境界で定義された関数です.
離散化に使う N1 × N2 × N3 の格子点は $x_{i,j,k}$ = (X1(i), X2(j), X3(k))で定義されます.
i=1, ... , N1, j=1, ... , N2, k=1, ... , N3
B=[X1(1), X1(N1)] × [X2(1), X(N2)] × [X3(1), X3(N3)].
偏微分方程式に現れる関数および境界条件は領域の格子点での値で定義されます.
離散化によって生成された係数行列 $\boldsymbol{A}$ は, 対角形式格納法で格納されます.

(2)　パラメタ

A1.....入力. 係数 $a_1(x_{i,j,k})$を設定します.
　　A1(L1, L2, N3)なる 3 次元配列.
L1 .....入力. 配列 A1, A2, A3, B1, B2, B3, C, F の 1 次元目の大きさ(L1 ≥ N1).
L2 .....入力. 配列 A1, A2, A3, B1, B2, B3, C, F の 2 次元目の大きさ(L2 ≥ N2).
N1.....入力. x_1 方向の格子点の数(N1 > 2).
N2.....入力. x_2 方向の格子点の数(N2 > 2).
N3.....入力. x_3 方向の格子点の数(N3 > 2).
A2.....入力. 係数 $a_2(x_{i,j,k})$を設定します.
　　A2(L1, L2, N3)なる 3 次元配列.
A3.....入力. 係数 $a_3(x_{i,j,k})$を設定します.
　　A3(L1, L2, N3)なる 3 次元配列.

X1.....入力. X1(1:N1)に格子点での x_1 座標の値を格納します. 座標の値は昇順に格納します.

X1(i) < X1(i+1) , i=1, ... , N1 - 1

X1(N1)なる 1 次元配列.

X2.....入力. X2(1:N2)に格子点での x_2 座標の値を格納します. 座標の値は昇順に格納します.

X2(i) < X2(i+1) , i=1, ... , N2 - 1

X2(N2)なる 1 次元配列.

X3.....入力. X3(1:N3)に格子点での x_3 座標の値を格納します. 座標の値は昇順に格納します.

X3(i) < X3(i+1) , i=1, ... , N3 - 1

X3(N3)なる 1 次元配列.

B1.....入力. 係数 $b_1(x_{i,j,k})$, および境界条件 β_1 を以下のように設定します.

$$\mathrm{B1}(i,j,k)=\begin{cases}\beta_1(x_{1,j,k}) & i=1\\ \beta_1(x_{\mathrm{N1},j,k}) & i=\mathrm{N1}\\ \beta_1(x_{i,1,k}) & j=1\\ \beta_1(x_{i,\mathrm{N2},k}) & j=\mathrm{N2}\\ \beta_1(x_{i,j,1}) & k=1\\ \beta_1(x_{i,j,\mathrm{N3}}) & k=\mathrm{N3}\\ b_1(x_{i,j,k}) & \text{上記以外}\end{cases}$$

B1(L1, L2, N3)なる 3 次元配列.

B2.....入力. 係数 $b_2(x_{i,j,k})$, および境界条件 β_2 を以下のように設定します.

$$\mathrm{B2}(i,j,k)=\begin{cases}\beta_2(x_{1,j,k}) & i=1\\ \beta_2(x_{\mathrm{N1},j,k}) & i=\mathrm{N1}\\ \beta_2(x_{i,1,k}) & j=1\\ \beta_2(x_{i,\mathrm{N2},k}) & j=\mathrm{N2}\\ \beta_2(x_{i,j,1}) & k=1\\ \beta_2(x_{i,j,\mathrm{N3}}) & k=\mathrm{N3}\\ b_2(x_{i,j,k}) & \text{上記以外}\end{cases}$$

B2(L1, L2, N3)なる 3 次元配列.

B3.....入力. 係数 $b_3(x_{i,j,k})$, および境界条件 β_3 を以下のように設定します.

$$\mathrm{B3}(i,j,k)=\begin{cases}\beta_3(x_{1,j,k}) & i=1\\ \beta_3(x_{\mathrm{N1},j,k}) & i=\mathrm{N1}\\ \beta_3(x_{i,1,k}) & j=1\\ \beta_3(x_{i,\mathrm{N2},k}) & j=\mathrm{N2}\\ \beta_3(x_{i,j,1}) & k=1\\ \beta_3(x_{i,j,\mathrm{N3}}) & k=\mathrm{N3}\\ b_3(x_{i,j,k}) & \text{上記以外}\end{cases}$$

B3(L1, L2, N3)なる 3 次元配列.

C.....入力. 係数 $c(x_{i,j,k})$, および境界条件 γ を以下のように設定します.

$$\mathrm{C}(i,j,k)=\begin{cases}\gamma(x_{1,j,k}) & i=1\\ \gamma(x_{\mathrm{N1},j,k}) & i=\mathrm{N1}\\ \gamma(x_{i,1,k}) & j=1\\ \gamma(x_{i,\mathrm{N2},k}) & j=\mathrm{N2}\\ \gamma(x_{i,j,1}) & k=1\\ \gamma(x_{i,j,\mathrm{N3}}) & k=\mathrm{N3}\\ c(x_{i,j,k}) & \text{上記以外}\end{cases}$$

C(L1, L2, N3)なる 3 次元配列.

F .....入力. 係数 $f(x_{i,j,k})$, および境界条件 ϕ を以下のように設定します.

$$\mathrm{F}(i,j,k)=\begin{cases}\phi(x_{1,j,k}) & i=1\\ \phi(x_{\mathrm{N1},j,k}) & i=\mathrm{N1}\\ \phi(x_{i,1,k}) & j=1\\ \phi(x_{i,\mathrm{N2},k}) & j=\mathrm{N2}\\ \phi(x_{i,j,1}) & k=1\\ \phi(x_{i,j,\mathrm{N3}}) & k=\mathrm{N3}\\ f(x_{i,j,k}) & \text{上記以外}\end{cases}$$

F(L1, L2, N3)なる 3 次元配列.

A.....出力. 偏微分方程式を離散化して生成された係数行列の非零要素が格納されます.

A(1:N, NDIAG)に係数行列が格納されます.

A(KA, NA)なる 2 次元配列.

対角形式の格納方法については, “SSL II 拡張機能使用手引書 II 第 I 部 概説 3.2.1.1 一般スパース行列の格納方法 b. 一般スパース行列の対角形式の格納方法”を参照してください.

KA.....入力. 配列 A の 1 次元目の大きさ(≥ N=N1 × N2 × N3).

NA.....入力. 配列 A の 2 次元目の大きさ(≥ NDIAG=7).

N.....入力. 行列 $\boldsymbol{A}$ の次数 n(= N1 × N2 × N3).

NDIAG.....出力. 係数行列 $\boldsymbol{A}$ の非零要素を含む対角ベクトル列の総数(= 7).

NOFST.....出力. 配列 A に格納される対角ベクトルに対応した主対角ベクトルからの距離が格納されます. 上対角ベクトル列は正, 下対角ベクトル列は負の値で表します.

NOFST(NDIAG)なる 1 次元配列.

R.....出力. 偏微分方程式を離散化して生成された連立 1 次方程式の右辺定数ベクトルが R(1:N)に格納されます.

R(N)なる 1 次元配列.

ICON.....出力. コンディションコード.

表 DM_VPDE3D-1 参照.

表 DM_VPDE3D-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし	―
30000	L1<N1, L2<N2, N1<3, N2<3, N3<3, NA<7, KA<N1 × N2 × N3	処理を打ち切る.
30001	格子点の座標が昇順に格納されていない.	

(3)　使用上の注意

a.　注意

①　線型系や固有値問題の解法によって計算された格子点での解の値は, 格子点の配置や数に大きく依存します.

②　隣接する格子点の間の距離の変化は緩やかでなければなりません. 例えば x_1 方向の条件として以下を満たすべきです.

$$0.5 \le \frac{\mathrm{X1}(i)-\mathrm{X1}(i-1)}{\mathrm{X1}(i+1)-\mathrm{X1}(i)} \le 2 \quad ,i=2,...,\mathrm{N1}-1$$

(x_2 方向および x_3 方向も同様です.) この条件が満たされていないと係数行列は悪条件となる可能性があります. 係数行列の条件数は格子ばかりでなく係数関数によっても決まることに注意してください.

b.　使用例

領域は channel$[-1, 1]^2 \times [0, 5]$とします. 偏微分方程式は

$$-\left(\frac{\partial^2 u}{\partial^2 x_1}+\frac{\partial^2 u}{\partial^2 x_2}+\frac{\partial^2 u}{\partial^2 x_3}\right)+v_1\frac{\partial u}{\partial x_1}+v_2\frac{\partial u}{\partial x_2}=0$$

回転している速度場により生じる channel を伝わる量 u の拡散をモデル化したものです.

$$v=(v_1,v_2,v_3)=v_0\cdot\left(\frac{x_2}{\sqrt{x_1^2+x_2^2}},\frac{-x_1}{\sqrt{x_1^2+x_2^2}},0\right)$$

ここで, v_0 は実定数(例えば, v_0=1)です. 境界条件は次のように設定されます.

$u=0$　　$x_3=0$
$u=1$　　$x_3=5$
$\frac{\partial u}{\partial n}=0$　上記以外

ここで n は channel の境界での法線場を示しています.
(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT NONE
```

```
      INTEGER   N1,N2,N3,KA,NA,L1,L2,L3

      PARAMETER(N1=49,N2=49,N3=25,L1=N1,L2=N2,L3=N3,
     &          KA=N1*N2*N3,NA=7)

      INTEGER  NOFST(NA)
      DOUBLE PRECISION V0,X1(L1),X2(L2),X3(L3),
     &         A1(L1,L2,L3),A2(L1,L2,L3),A3(L1,L2,L3),
     &         B1(L1,L2,L3),B2(L1,L2,L3),B3(L1,L2,L3),
     &         C(L1,L2,L3),F(L1,L2,L3),A(KA,NA),R(KA)

      INTEGER  Z1,Z2,Z3,ICON,I,J,N,NDIAG

      V0=1.
C
C create grid nodes nodes:
C
      DO 11 Z1=1,N1
        X1(Z1)=(2*DBLE(Z1-1)/DBLE(N1-1)-1.)
11    CONTINUE
      DO 12 Z2=1,N2
        X2(Z2)=(2*DBLE(Z2-1)/DBLE(N2-1)-1.)
12    CONTINUE
      DO 13 Z3=1,N3
        X3(Z3)=DBLE(Z3-1)/DBLE(N3-1)
13    CONTINUE
C
C coefficient functions:
C
      DO 2000 Z3=1,N3

       DO 20 Z2=1,N2
        DO 20 Z1=1,N1
          A1(Z1,Z2,Z3)=1
          A2(Z1,Z2,Z3)=1
          A3(Z1,Z2,Z3)=1
20     CONTINUE
       DO 21 Z2=2,N2-1
        DO 21 Z1=2,N1-1
          B1(Z1,Z2,Z3)= V0*X2(Z2)/SQRT(X1(Z1)**2+X2(Z2)**
     &                 2+1.D-10)
```

```
          B2(Z1,Z2,Z3)=-V0*X1(Z1)/SQRT(X1(Z1)**2+X2(Z2)**
     &                 2+1.D-10)
          B3(Z1,Z2,Z3)=0
          C (Z1,Z2,Z3)=0
          F (Z1,Z2,Z3)=0
21    CONTINUE
C
C boundary conditions at faces X1=-1 and X1=1:
C
      DO 101 Z2=1,N2
        B1(1,Z2,Z3)=-1
        B2(1,Z2,Z3)=0
        B3(1,Z2,Z3)=0
        C (1,Z2,Z3)=0
        F (1,Z2,Z3)=0

        B1(N1,Z2,Z3)=1
        B2(N1,Z2,Z3)=0
        B3(N1,Z2,Z3)=0
        C (N1,Z2,Z3)=0
        F (N1,Z2,Z3)=0
101   CONTINUE
C
C boundary conditions at faces X2=-1 and X2=1:
C
      DO 102 Z1=1,N1
        B1(Z1,1,Z3)=0
        B2(Z1,1,Z3)=-1
        B3(Z1,1,Z3)=0
        C (Z1,1,Z3)=0
        F (Z1,1,Z3)=0

        B1(Z1,N2,Z3)=0
        B2(Z1,N2,Z3)=1
        B3(Z1,N2,Z3)=0
        C (Z1,N2,Z3)=0
        F (Z1,N2,Z3)=0
102   CONTINUE
C
C boundary conditions at faces X3=0 and X3=5:
C
      IF (Z3.EQ.1) THEN
```

```
        DO 103 Z1=1,N1
          DO 103 Z2=1,N2
            B1(Z1,Z2,1)=0
            B2(Z1,Z2,1)=0
            B3(Z1,Z2,1)=0
            C (Z1,Z2,1)=1
            F (Z1,Z2,1)=0
103     CONTINUE
      ELSE IF (Z3.EQ.N3) THEN
        DO 113 Z1=1,N1
          DO 113 Z2=1,N2
            B1(Z1,Z2,N3)=0
            B2(Z1,Z2,N3)=0
            B3(Z1,Z2,N3)=0
            C (Z1,Z2,N3)=1
            F (Z1,Z2,N3)=1
113     CONTINUE
      END IF

2000  CONTINUE
C
C build the linear system:
C
      N=N1*N2*N3
      CALL DM_VPDE3D (A1,L1,L2,N1,N2,N3,A2,A3,X1,X2,X3,B1,B2,B3,
     &                C,F,A,KA,NA,N,NDIAG,NOFST,R,ICON)
      PRINT*,'DM_VPDE3D : ICON=',ICON
      IF (ICON.GT.29999) GOTO 9999
C
C write the matrix to a file:
C
      WRITE (6,'(3D23.16)') ((A(I,J),I=1,N,1000),J=1,NDIAG)
      WRITE (6,'(3I10)') (NOFST(J),J=1,NDIAG)
      WRITE (6,'(3D23.16)') (R(I),I=1,N,1000)
9999  CONTINUE
      STOP
      END
```

(4)　手法概要

拡散項 $-\nabla a \nabla$ は x_1, x_2 および x_3 方向に対して 2 次の中心差分法によって近似されます. 移流項 $b\nabla$ は, 1 次の風上差分法によって近似されます. 詳細は“付録 1 参考文献一覧表”の[75]を参照してください.

DM_VRADAU5

スティフ連立 1 階常微分方程式/微分代数方程式　(陰的ルンゲ・クッタ法)
CALL DM_VRADAU5 (N,FCN,X,Y,XEND,H,RTOL,ATOL,ITOL,JAC,IJAC,MLJAC, MUJAC,MAS,IMAS,MLMAS,MUMAS,SOLOUT,IOUT, WORK,LWORK,IWORK,LIWORK,RPAR,IPAR,ICON)

(1)　機能

スティフな連立 1 階常微分方程式, または微分代数方程式の初期値問題の数値解を陰的ルンゲ・クッタ法で求めます.

$$\boldsymbol{M}\boldsymbol{y}' = \boldsymbol{f}(x,\boldsymbol{y}) \qquad \boldsymbol{y}(x_0) = \boldsymbol{y}_0 \tag{1.1}$$

ここで $\boldsymbol{M}$ は n 次元の定数正方行列 (質量マトリックスと呼びます), $\boldsymbol{y}$ は n 次元の解ベクトル(成分を $y_1, y_2, \ldots, y_n$ とします), $\boldsymbol{f}(x,\boldsymbol{y})$ は n 次元の関数ベクトル(成分を $f_1, f_2, \ldots, f_n$ とします), そして $\boldsymbol{y}_0$ は $x = x_0$ おける初期値ベクトルです(成分を $y_{01}, y_{02}, \ldots, y_{0n}$ とします.) .

$\boldsymbol{M}$ が正則で単位行列 $\boldsymbol{I}$ でない場合は陰的な常微分方程式となります.また $\boldsymbol{M}$ が特異行列のときに微分代数方程式となります.

本サブルーチンは, 一回の呼出しで, 指定された点 $x_{end}\,(\neq x_0)$ での解が求まった時点で復帰します. x_0 から x_{end} に向かって積分していく過程で 1 ステップが完了するたびに解を受け取るには引数 SOLOUT に相当する利用者サブルーチンを用意します.

本サブルーチンは E.Hairer と G.Wanner(2011 年 3 月現在, ジュネーブ大学)によるフリーのプログラム RADAU5 に基づいており, 付録 2 に示すライセンス条文のもとで再配布するものです.

(2)　パラメタ

N.....入力. 方程式の元数($N \geq 1$).

FCN.....入力. $\boldsymbol{f}(x,\boldsymbol{y})$ の値を計算する利用者サブルーチンの名前. 次のような引数をもつサブルーチンとして用意します. EXTERNAL 宣言が必要.

```
SUBROUTINE  FCN(N,X,Y,F,RPAR,IPAR)
REAL*8  X,Y(N),F(N)
```

ここで,

N: 入力. 方程式の元数.

X: 入力. 独立変数 x

Y: 入力. 解ベクトル $\boldsymbol{y}$

F: 出力. $\boldsymbol{f}(x,\boldsymbol{y})$. F(1)= f_1 , F(2)= f_2 ,….. F(N)= f_n

RPAR,IPAR については下記参照.

X.....入力. 初期値 x_0 .

出力. 解が求まった最終点 (正常終了の場合は x_{end})

Y.....入力. 解の初期値. Y(1)= y_{01}, Y(2)= y_{02} ,…,Y(N)= y_{0n} の順に与えます. 大きさ N の一次元配列.

出力. 引数 X の出力値における解.

XEND .....入力. 解を求めたい最終点 x_{end} ($x_{end} - x_0$ は正でも負でも構いません).途中の解は引数 IOUT の指定によってサブルーチン SOLOUT へ渡されます.

H.....入力. 初期ステップ幅の予測値. 初期の過渡期を持ったスティフな方程式においては H= $1.0/(\boldsymbol{f}'(x,\boldsymbol{y})$ のノルム) ---通常 1.D-3 または 1.0D-5 ---が良いとされていますが, 与えられた予測値は本サブルーチン内部で調整されるので予測値の値はさほど重要ではありません. H=0.0 と与えた場合は, デフォルトの H=1.D-6 が採用されます.

出力. 積分の最終ステップで採用されたステップ幅.

RTOL,ATOL .入力. 解に対する要求精度です. RTOL には相対誤差を, ATOL には絶対誤差を与えます. 両者は共にスカラーか(定数ではなく変数で指定), あるいは大きさ N のベクトルの値とします.その選択は下記の ITOL で行います.

なお, ATOL は(またはその成分 ATOL(I)は)正であり, RTOL は(またはその成分 RTOL(I)は) 10 u よりも大であること. ここで u は丸め誤差の単位であり利用者が WORK(1)で指定した値か, デフォルトの DMACH()の値.

ITOL .....入力.RTOL, ATOL に対するスイッチ.

ITOL=0 とすると, RTOL, ATOL は共にスカラーを与えます.このとき解の成分 Y(I) (I=1, …, N) の局所誤差はほぼ RTOL*|Y(I)| + ATOL 以下になるよう積分します.

ITOL ≠ 0 とすると RTOL, ATOL は共に大きさ N のベクトルとして値を与えます. このとき解の成分 Y(I) (I=1, …, N) の局所誤差はほぼ RTOL(I)*|Y(I)| + ATOL(I) 以下になるよう積分します.

JAC .....入力. $\boldsymbol{f}(x,\boldsymbol{y})$ の $\boldsymbol{y}$ に関する偏微分 (ヤコビ行列) を計算するサブルーチンの名前. EXTERNAL 宣言が必要. ただし, 下記に述べる引数 IJAC を 0 とした場合は用意する必要はありません.その場合は, ダミーのサブルーチンを用意して下さい.

IJAC ≠ 0 のときは, 次のような引数を持つサブルーチンとして用意します.

```
  SUBROUTINE  JAC(N,X,Y,DFY,LDFY,RPAR,IPAR)
  REAL*8 X,Y(N),DFY(LDFY,N)
  DFY(1,1)=…….
```

LDFY は JAC を呼出す側のサブルーチンから与えられます.

本サブルーチンは, ヤコビ行列が密行列の場合とバンド行列の場合の二通りのケースを引数 MLJAC を通して区別して扱います.

MLJAC=N の場合, ヤコビ行列を密行列として扱います. サブルーチン JAC の引数 DFY には

$$\mathrm{DFY(I,J)} = \frac{\partial f_i}{\partial y_j}$$

となるように値を入れます.

0 ≤ MLJAC < N の場合, ヤコビ行列をバンド行列として扱います.このとき

MLJAC, MUJAC はバンド行列の下バンド幅, 上バンド幅を意味し, 配列 DFY には

$$\text{DFY}(\ \text{I}-\text{J}+\text{MUJAC}+1,\ \text{J}) = \frac{\partial f_i}{\partial y_j}$$

となるように値をいれます.

例として, N=6 で MLJAC=3, MUJAC=1 の場合の DFY への格納の様子を図 DM_VRADAU5-1 に示します.ここで $a_{ij} = \partial f_i / \partial y_j$. *印の要素は参照されません.

$$\begin{pmatrix} a_{11} & a_{12} & & & & \\ a_{21} & a_{22} & a_{23} & & & \\ a_{31} & a_{32} & a_{33} & a_{34} & & \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & \\ & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} \\ & & a_{63} & a_{64} & a_{65} & a_{66} \end{pmatrix} \Rightarrow \begin{array}{|c|c|c|c|c|c|} \hline * & a_{12} & a_{23} & a_{34} & a_{45} & a_{56} \\ \hline a_{11} & a_{22} & a_{33} & a_{44} & a_{55} & a_{66} \\ \hline a_{21} & a_{32} & a_{43} & a_{54} & a_{65} & * \\ \hline a_{31} & a_{42} & a_{53} & a_{64} & * & * \\ \hline a_{41} & a_{52} & a_{63} & * & * & * \\ \hline \end{array}$$

図 DM_VRADAU5-1

IJAC.....入力. ヤコビ行列の計算法を指定するスイッチ.

IJAC=0 とすると, ヤコビ行列は本サブルーチン内部で差分で近似計算します.

IJAC ≠ 0 とすると,ヤコビ行列は利用者が与えるサブルーチン JAC で計算します.

MLJAC.....入力.ヤコビ行列が密行列かバンド行列かを示すスイッチ.

MLJAC=N とすると, ヤコビ行列は密行列であることを, また 0 ≤ MLJAC<N のときは, ヤコビ行列がバンド行列であることを示すとともに MLJAC には下バンド幅を与えます.

MUJAC.....入力. ヤコビ行列がバンド行列の場合に上バンド幅を与えます. MLJAC=N の場合は入力する必要はありません.

MAS.....入力. 質量マトリックスである $n \times n$ の定数正方行列 $\boldsymbol{M}$ を与えるサブルーチンの名前. EXTERNAL 宣言が必要. ただし $\boldsymbol{M}$ が単位行列の場合を特別扱いするために引数 IMAS で区別します.

IMAS=0 とすると $\boldsymbol{M}$ が単位行列であることを示します. この場合はサブルーチン MAS を与える必要はなく, 単にダミーのサブルーチンを与えます.

IMAS ≠ 0 とすると $\boldsymbol{M}$ を与えるサブルーチンが必要です. 次のような引数を持つサブルーチンとして用意します.

```
SUBROUTINE  MAS(N,AM,LMAS,RPAR,IPAR)
REAL*8  AM(LMAS,N)
AM(1,1)= ……
```

本サブルーチンは, $\boldsymbol{M}$ が密行列の場合とバンド行列の場合の二通りのケー

スを引数 MLMAS を通して区別して扱います.
MLMAS=N の場合, 質量マトリックスを密行列として扱います. サブルーチン MAS の引数 AM には

$$\mathrm{AM(I,J)} = \boldsymbol{M}_{ij}$$

となるように値を入れます.
$0 \le$ MLMAS<N の場合, 質量マトリックスをバンド行列として扱います.このとき MLMAS, MUMAS はバンド行列の下バンド幅, 上バンド幅を意味し, 配列 AM には

$$\mathrm{AM(I-J+MUMAS+1, J)} = \boldsymbol{M}_{ij}$$

となるように値をいれます. ヤコビ行列がバンド行列の場合の格納方法と同じです.
MLMAS $\le$ MLJAC,MUMAS $\le$ MUJAC であること.

IMAS .....入力. 質量マトリックス $\boldsymbol{M}$ についての情報を与えます.
IMAS=0 とすると $\boldsymbol{M}$ は単位行列として見なします.
IMAS≠0 とすると $\boldsymbol{M}$ は単位行列ではないことを意味し, サブルーチン MAS を与えます.

MLMAS.....入力. 質量マトリックス $\boldsymbol{M}$ が密行列かバンド行列かを示すスイッチ.
MLMAS=N とすると, $\boldsymbol{M}$ は密行列であることを, また $0 \le$ MLMAS<N のときは, $\boldsymbol{M}$ がバンド行列であることを示すとともに MLMAS には下バンド幅を与えます. ただし, MLMAS $\le$ MLJAC とします.

MUMAS .....入力. 質量マトリックス $\boldsymbol{M}$ がバンド行列の場合の上バンド幅を与えます.
MLMAS=N の場合は入力する必要はありません. MUMAS $\le$ MUJAC.

SOLOUT.....入力.XEND まで積分する過程で求まった途中の解を受け取るサブルーチンの名前.　途中の解を受け取るか否かは 引数 IOUT で指定します.
IOUT≠0 とすると次の仕様で SOLOUT へ解が渡されます.

```
  SUBROUTINE SOLOUT(NR,XOLD,X,Y,CONT,LRC,N,RPAR,IPAR,
                    IRTRN,WORK2,IWORK2)
  REAL*8  X,Y(N),CONT(LRC)
```

SOLOUT へは積分の NR 番目のステップで求まった X における解 Y が渡されます.初期値 $x_0, \boldsymbol{y}_0$ は NR=1 として最初の解として渡されます.また XEND における解が求まったときも渡されます.
XOLD は一つ前のステップでの x の値です. IRTRN は積分を終了させたい場合に使われ負の値を入れると, 本サブルーチンは積分を終了し, 本サブルーチンの呼出しプログラムに戻ります.

密出力について：
利用者は SOLOUT 内で, 区間 [XOLD, X] 内の任意の点で解を計算することができます.たとえば, あらかじめ意図した等間隔な分点での解を計算したい場合に有益です. 区間 [XOLD, X]内の任意の点での解を計算するには関数副プログラム DM_VCONTR5 を呼び出します. 例えば[XOLD, X]内の点 S

における解の I 番目 ($1 \le I \le N$) の成分を計算したい場合は,

DM_VCONTR5(I,S,CONT,LRC,WORK2,IWORK2)

と呼出します. この関数副プログラムの復帰値は REAL*8 型です.

CONT,LRC,WORK2,IWORK2 は SOLOUT に渡された値をそのまま利用して下さい.

IOUT.....入力. SOLOUT で途中の解を受け取るか否かを示すスイッチ.

IOUT=0 とすると SOLOUT には途中の解は渡されません.

IOUT≠0 とすると SOLOUT に途中の解を渡します.

WORK .....作業領域. 大きさ LWORK なる一次元配列.

WORK(1), WORK(2), …. , WORK(20)は本サブルーチン内部で使われる各種パラメタとして使われ, デフォルトのパラメタ値を使う場合は WORK(1), ……., WORK(20) には 0.D0 を入れます. デフォルト以外の値を指定する場合は後述の「パラメタの詳細な指定について」を参照.

WORK(21)から WORK(LWORK)までは内部で生成されるベクトルや行列のための作業領域です. 問題の大きさに依存します.

LWORK は少なくとも以下の大きさです.

N * (LJAC+LMAS+3*LE*12) + 20

ここで

LJAC は

LJAC=N (MLJAC=N の場合)

LJAC=MLJAC+MUJAC+1 (MLJAC < N の場合)

LMAS は

LMAS=0 (IMAS=0 の場合)

LMAS=N (IMAS $\neq 0$ かつ MLMAS=N の場合)

LMAS=MLMAS+MUMAS+1 (MLMAS<N の場合)

LE は

LE=N (MLJAC=N の場合)

LE=2*MLJAC+MUJAC +1 (MLJAC < N の場合)

ヤコビ行列が密行列でかつ $\boldsymbol{M}$ が単位行列の場合は

LWORK=4 * N*N + 12*N + 20

となります.

常微分方程式または微分代数方程式が 2 階以上の高階の問題に対して効率の良い使い方を後述しますが, その際には IWORK(9)に正の整数 M1 を入れます. その場合の作業領域の大きさは

LWORK=N*(LJAC+12)+(N-M1)*(LMAS+3*LE)+20

となります. ただしこの式における LJAC, LMAS, LE はこれらの上記の定義式における N を N-M1 で置き換えた値です.

LWORK.....入力. 作業領域 WORK の大きさ.

IWORK.....大きさ LIWORK なる整数型の作業領域. LIWORK は少なくとも 3×N+20 であること.

IWORK(1), IWORK(2), …., IWORK(20) は本サブルーチン内部で使われる各

種パラメタとして使われ, デフォルトのパラメタ値を使う場合は IWORK(1), ……., IWORK(20) には 0 を入れます. デフォルト以外の値を設定する場合は後述の「**パラメタの詳細な指定について**」を参照してください.
IWORK(21), …, IWORK(LIWORK)は作業領域として使われます.

出力. IWORK(14)から IWORK(20)には XEND までの積分が完了した際の統計情報を出力します.

IWORK(14)	NFCN	サブルーチン FCN の呼出し回数(ただし, IJAC=0 の場合のヤコビ行列近似のための呼出しは含みません).
IWORK(15)	NJAC	ヤコビ行列を計算した回数(IJAC $\neq 0$ の場合はサブルーチン JAC を呼出した回数となります).
IWORK(16)	NSTEP	試行されたステップの回数.
IWORK(17)	NACCPT	成功したステップの回数.
IWORK(18)	NREJCT	受け入れられなかったステップの回数(これは要求精度を満たさなかったときのステップ). ただし積分の開始時はカウントしていません.
IWORK(19)	NDEC	線形方程式の解法で LU 分解した回数.
IWORK(20)	NSOL	LU 分解した行列を使って求解した回数(ただしステップ幅の選択のために要した求解はカウントしていません).

LIWORK.....入力. 整数型作業領域 IWORK の大きさ.

RPAR,IPAR...実数型と整数型のパラメタ (もしくはパラメタ配列).
これら引数の用途は, 本サブルーチンを呼出すプログラムと, サブルーチン FCN, JAC, MAS, SOLOUT とで数値のやり取りをする場合に使います.
例えば, 本サブルーチンを呼出すプログラムで, RPAR,IPAR に何らかの値を設定しておけば, サブルーチン FCN,JAC, MAS, SOLOUT の中でそれを参照することができます.

ICON.....出力. コンディションコード.
表 DM_VRADAU5-1 参照.

パラメタの詳細な指定について：

配列 IWORK, WORK の最初の 20 要素には 0 を指定するのがデフォルトですが, デフォルト値以外の値を設定できます.

IWORK(1) .....入力. IWORK(1) ≠0 と指定すると, ヤコビ行列が密行列のとき, かつ $\boldsymbol{M}$ が単位行列の場合に, ヤコビ行列をヘッセンベルグ行列へ変換します.これは N が大きいときに効率が良くなります.

IWORK(2) .....入力. XEND まで積分するのに要するステップ数の上限を指定します(>0).
IWORK(2) =0 の場合はデフォルトで 100000 が設定されます.

IWORK(3) .....入力. 内部で解かれる連立非線形代数方程式の解法においてニュートン法の反復回数の上限を指定します(>0).
IWORK(3) =0 の場合はデフォルトで 7 が設定されます

IWORK(4) .....入力. IWORK(4)=0 の場合, ニュートン法における初期値として補外された collocation 解を使います. IWORK(4) ≠0 の場合, 通常の初期値が使われます. 後者は, ニュートン法の収束が遅い場合(これは前述の NSTEP が NACCPT+NREJCT よりも大きい場合)に有効です.

下記の 3 つのパラメタは, 指数が 1 よりも大きい微分代数方程式を解く場合に重要です. その場合, 解ベクトルの要素 Y(1),Y(2), …, Y(N) は指数 1, 指数 2, 指数 3 の順に並ぶようにして下さい.本サブルーチンの内部で行われている数値解の誤差評価は, 指数 2 の要素に対しては H が乗ぜられ, 指数 3 の要素に対しては H^2 が乗ぜられます. デフォルトである IWORK(5)=0, IWORK(6)=0, IWORK(7)=0 以外の値を設定するときは, これら 3 つの整数の和は N であること.

IWORK(5) .....入力. 解ベクトルにおける指数 1 の要素数(≥0). $\boldsymbol{M}$ が単位行列かあるいは正則な行列なら, 常微分方程式となるので, すべての要素が指数 1 となり,その場合はデフォルト設定 (IWORK(5)=0) とすれば十分です.

IWORK(6) .....入力. 解ベクトルにおける指数 2 の要素数(≥0). デフォルトは IWORK(6)=0 です.

IWORK(7) .....入力. 解ベクトルにおける指数 3 の要素数(≥0). デフォルトは IWORK(7)=0 です.

IWORK(8) .....入力. ステップ幅の制御に関するパラメタです.
IWORK(8)=1 のとき, 修正予期コントローラ (Gustafsson)によりステップ幅を制御します. IWORK(8)>1 のとき, 標準的ステップ幅コントローラによりステップ幅を制御します.
デフォルト(IWORK(8)=0)は前者の方法です. 前者の方法は, より安定的な制御法であることが実験により観察されています.後者の方法は簡単な問題に対しては, しばしば若干速い制御法となります.

IWORK(9)と IWORK(10)は, 方程式が 1 階ではなく 2 階以上の方程式を解くときに有益な手段を与えます. よく知られるように 2 階以上の方程式は新しい未知関数を導入することで 1 階の方程式に変換できます. そのとき

Y(I)' = Y(I+M2) I=1,2,…,M1

のような特別な式を含むことになります. ここで M1 は M2 の整数倍です. このような場合に IWORK(9), IWORK(10) を設定することで, 効率が著しく向上します.
例として, 2 階の常微分方程式 $\boldsymbol{p}'' = \boldsymbol{g}(x, \boldsymbol{p}, \boldsymbol{p}')$ は次のように 1 階の方程式に変換できます.

$$\boldsymbol{p}' = \boldsymbol{v}$$
$$\boldsymbol{v}' = \boldsymbol{g}(x, \boldsymbol{p}, \boldsymbol{v})$$

ここで, $\boldsymbol{p}$, $\boldsymbol{v}$ は共に N / 2 次元のベクトルで M1=M2=N/2 となります.
このような特別の構造を活かして (M1 > 0) 本サブルーチンを利用する場合は, 上記に述べた引数のいくつかは異なった設定が必要になります:

JAC .....入力. ヤコビ行列の非自明な要素だけを計算するサブルーチンとします.例えば, 上記の 2 階から 1 階に変換された方程式で言えば,

$$\left(\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{p}} \quad \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{v}}\right)$$

なる N/2×N の行列だけを計算します.
一般的な仕様は次のとおりです.以下, F は 1 階に変換したあとの右辺全体, Y は全体の解ベクトルを示すものとします.
MLJAC = N - M1 と設定すると, 非自明なヤコビ行列は密行列とみなし,

$$\mathrm{DFY(I,\,J)} = \frac{\partial \mathrm{F(I+M1)}}{\partial \mathrm{Y(J)}}\ ,\ \mathrm{I=1,\ldots,N-M1,\quad J=1,\ldots,N}$$

と代入します.
0 ≤ MLJAC<N - M1 場合は, 非自明なヤコビ行列はバンド行列とみなし, M1=M2×MM なる整数 MM を使って

$$\mathrm{DFY(I-J+MUJAC+1,J+K\times M2)} = \frac{\partial \mathrm{F(I+M1)}}{\partial \mathrm{Y(J+K\times M2)}}$$

$$\mathrm{I=1,\ldots,N-M1,\quad J=1,\ldots,M2,\quad K=0,..,MM}$$

と代入します. ただしバンド行列として扱う場合は, N=M1+M2 を満たす場合に限ります.

MLJAC .....入力.
MLJAC=N - M1 : ヤコビ行列の非自明な部分が密行列の場合.
0 ≤ MLJAC<N - M1: M1=M2×MM としたとき MM+1 個の部分行列,

$$\frac{\partial \mathrm{F(I+M1)}}{\partial \mathrm{Y(J+K\times M2)}}\ ,\quad \mathrm{I=1,\ldots,N-M1\ ,\quad J=1,\ldots,M2,\quad K=0,\ldots,MM}$$

がバンド行列であり, MLJAC には最大の下バンド幅を設定.

MUJAC.....入力.
上記の部分行列について, 最大の上バンド幅を設定します. MLJAC=N - M1 の場合は設定する必要はありません.

MAS .....入力.
IMAS=0 のときは MAS は単位行列とみなします. この場合は MAS を計算するサブルーチンはダミールーチンを用意してください.
IMAS≠0 のときは, 右下の角に位置する部分行列だけを MAS では与えます. 例を以下に示します.

$$\boldsymbol{M}\boldsymbol{p}'' = \boldsymbol{g}(x, \boldsymbol{p}, \boldsymbol{p}')$$

なる 2 階の方程式は, 次のように 1 階の問題に変換できます.

$$\boldsymbol{p}' = \boldsymbol{v}$$
$$\boldsymbol{M}\boldsymbol{v}' = \boldsymbol{g}(x, \boldsymbol{p}, \boldsymbol{v})$$

これは

$$\begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{M} \end{pmatrix} \begin{pmatrix} \boldsymbol{p}' \\ \boldsymbol{v}' \end{pmatrix} = \begin{pmatrix} \boldsymbol{v} \\ \boldsymbol{g}(x, \boldsymbol{p}, \boldsymbol{v}) \end{pmatrix}$$

と書けます. この場合, 左辺の係数行列は「(1)機能」で表現した $\boldsymbol{M}$ に相当します.改めて左辺の係数行列全体を $\boldsymbol{M}$ で表すと,

MLMAS=N-M1 ときは右下の角の部分行列は密行列とみなし,MAS の引数 AM には

AM(I,J)= M(I+M1, J+M1) , I=1,…,N-M1 , J=1,…,N-M1

のように与えます.

MLMAS≠N-M1 のときは右下の角の部分行列はバンド行列とみなし,

AM(I-J+MUMAS+1, J)= M(I+M1, J+M1)

のように与えます.

MLMAS.....入力.

MLMAS=N-M1 とき, $\boldsymbol{M}$ の非自明な部分行列は密行列.

0 ≤ MLMAS<N-M1 のとき $\boldsymbol{M}$ の非自明な部分行列がバンド行列であり MLMAS は下バンド幅を設定. MLMAS ≤ MLJAC であること.

MUMAS .....入力.

0 ≤ MLMAS<N-M1 のとき, $\boldsymbol{M}$ の非自明なバンド行列部分の上バンド幅を設定. MUMAS ≤ MUJAC であること. MLMAS=N-M1 の場合は, MUMAS の設定は不要です.

IWORK(9) .....入力. M1 の値(≥ 0). デフォルトでは M1=0.

IWORK(10) ...入力. M2 の値(≥ 0). デフォルトでは M1=0 のとき M2=N. M1>0 のとき M2=M1.

IWORK(9) >0 の場合は IWORK(9)+IWORK(10) ≤ N であること.

WORK(1).....入力. 丸め誤差の単位 u. DMACH() ≤ WORK(1)<1.0D0 であること.デフォルトは DMACH().

WORK(2).....入力. ステップ幅の予測に使う安全係数. デフォルトでは 0.9D0. デフォルト以外の値を設定する場合, 0.001D0<WORK(2)<1.0D0 であること.

WORK(3).....入力. ヤコビ行列を再計算するか否かを決める数値.デフォルトは 0.001D0. もしヤコビ行列の計算のコストが高い場合は例えば 0.1D0 程度に上げます. 小さな問題を解く場合は, 0.001D0 くらいに小さくします.

WORK(3)に負の値を入れると, 成功したステップ後に毎回ヤコビ行列を計算します. WORK(3) < 1.0D0 であること.

WORK(4).....入力. ニュートン法の収束判定値. 通常は 1.D0 未満の値. 小さな値を入れるほど遅くなりますが安全になります.

デフォルトは MAX(10u/TOLST,MIN(0.03D0, $\sqrt{\text{TOLST}}$))です.ここで u は丸め誤差の単位, TOLST=0.1·RTOL**(2/3)です. RTOL がベクトルのときは RTOL(1)が使われます. WORK(4)>u/TOLST であること.

WORK(5),

WORK(6).....入力. ステップ幅を変更するか否かを決めるパラメタです.

もし WORK(5)< HNEW/HOLD<WORK(6)のとき, ステップ幅は変更しませ

ん. WORK(5),WORK(6)の設定値によっては, (WORK(3)の設定とともに)大きな問題の場合, 内部での線形方程式解法の時間の節約になります.
小さな問題に対しては WORK(5)=1.D0, WORK(6)=1.2D0 が適当であり, 大きな問題でかつヤコビ行列が密行列になると WORK(5)=0.99D0, WORK(6)=2.D0 が適当です.
デフォルトでは WORK(5)=1.D0, WORK(6)=1.2D0 です.　WORK(5) ≤ 1.0D0, WORK(6) ≥ 1.0D0 であること.

WORK(7).....入力.ステップ幅の上限. デフォルトでは $x_{end} - x_0$ です.

WORK(8),
WORK(9).....入力.ステップ幅の選択についてのパラメタです.
新しいステップ幅は次の制約のもので決定されます.
WORK(8) ≤ HNEW/HOLD ≤ WORK(9)
デフォルトでは WORK(8)=0.2D0, WORK(9)=8.D0 です. WORK(8) ≤ 1.0D0, WORK(9) ≥ 1.0D0 であること.

表 DM_VRADAU5-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし	ー
100	サブルーチン SOLOUT の中で引数 IRTRN に負の値が設定された.	処理を打ち切る. 解は正常.
10000	IWORK(2)で設定されたステップ数の上限に達した.	処理を打ち切る. XEND における解は得られていない. IWORK(2)により大きな整数を設定してやり直すと正常終了する可能性はある.
21000	ステップ幅が極度に小さくなった.	処理を打ち切る.
22000	内部で解かれる線形方程式の係数行列が繰り返し特異になった.	処理を打ち切る.
30000	入力した引数の値が正しくないものがあった.	処理を打ち切る.

(3)　使用上の注意

a.　注意

①　利用者サブルーチン SOLOUT の役割

IOUT≠0 のとき, 本サブルーチンは, x_0 から x_{end} に向かって積分していく過程で 1 ステップが完了するたびにその時点での x の値と数値解 $\boldsymbol{y}$ を SOLOUT へ渡します. 具体的には, 仮に $x_0 < x_{end}$ の場合,

$$x_0 < x_1 < x_2 < \ldots < x_{end}$$

のように 1 ステップ進むごとに x_i とその点での数値解を SOLOUT へ渡します (x_0 と x_{end} も含みます). x_i は要求精度を満たすべく行われるステップ幅制御の結果で決まります.

利用者が意図する点列での数値解を求めたい場合は, SOLOUT の中で密出力用の関数副プログラム DM_VCONTR5 を利用することで可能となります. 例えば一定間隔を持った点列での解を求めたい場合は使用例 1 を参照してください.

意図する点列での解を求めるのに初期値と引数 XEND を繰り返し変えて呼び出すことは本サブルーチンでは適切ではありません.

② 利用者サブルーチン内でのスレッド並列化

利用者サブルーチンである FCN, JAC, MAS, および SOLOUT のいずれも必要なら OpenMP によるスレッド並列化することは可能です (本サブルーチンからは並列リージョンの外からこれらのサブルーチンを call します).

③ 微分代数方程式の「指数」と初期値

モデル $\boldsymbol{M}\boldsymbol{y}' = \boldsymbol{f}(x, \boldsymbol{y})$ において $\boldsymbol{M}$ が正則な場合は常微分方程式となり, 解ベクトル $\boldsymbol{y}$ の成分はすべて指数 1 となり IWORK(5)～IWORK(7)はデフォルトの 0 を設定します. またどんな初期値でも唯一の数値解が得られます.

これに対し, $\boldsymbol{M}$ が特異行列の場合は微分代数方程式となり, 指数と初期値について注意が必要になります. 以下, ガイドラインを述べます.

$\boldsymbol{M}$ が特異行列の場合, $\boldsymbol{M}$ は以下のように分解できます.

$$\boldsymbol{M} = \boldsymbol{S}\begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{pmatrix}\boldsymbol{T}$$

ここで $\boldsymbol{S},\boldsymbol{T}$ は n 次元の正則行列, $\boldsymbol{I}$ は n より小さな次元の単位行列です(そのような $\boldsymbol{S},\boldsymbol{T}$ は例えば, 完全ピボッティングによる $\boldsymbol{M}$ のガウス消去法で得られます). 分解された $\boldsymbol{M}$ をモデルに代入し, 両辺に $\boldsymbol{S}^{-1}$ を乗じ,

$$\boldsymbol{T}\boldsymbol{y} = \begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{w} \end{pmatrix}$$

とおけば, モデルは以下のような形に変換できます.

$$\begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{pmatrix}\begin{pmatrix} \boldsymbol{u}' \\ \boldsymbol{w}' \end{pmatrix} = \boldsymbol{S}^{-1}\boldsymbol{f}(x, \boldsymbol{T}^{-1}\begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{w} \end{pmatrix}) =: \begin{pmatrix} \boldsymbol{g}(x,\boldsymbol{u},\boldsymbol{w}) \\ \boldsymbol{h}(x,\boldsymbol{u},\boldsymbol{w}) \end{pmatrix}$$

または

$$\boldsymbol{u}' = \boldsymbol{g}(x,\boldsymbol{u},\boldsymbol{w})$$
$$\boldsymbol{0} = \boldsymbol{h}(x,\boldsymbol{u},\boldsymbol{w})$$

このように微分方程式と代数方程式に分離できた形をヘッセンベルグ形式と言い, 実用問題ではよく現れます.代数式の拘束条件を持った微分方程式とみなすことができます. 以下, このヘッセンベルグ形式の代表的な指数 1, 指数 2, 指数 3 の問題を挙げます. 本サブルーチンの使用時に参考にしてください. なお, 以下では, 式を簡略化するために独立変数の表記は省略します.

a) 指数 1 の問題

いま

$$\boldsymbol{y}' = \boldsymbol{f}(\boldsymbol{y},\boldsymbol{z}) \qquad (3.1a)$$
$$\boldsymbol{0} = \boldsymbol{g}(\boldsymbol{y},\boldsymbol{z}) \qquad (3.1b)$$

を考えます. ここで $\boldsymbol{y},\boldsymbol{z}$ は未知なる関数ベクトルで長さの和は n とします.

(1.1)でいう $\boldsymbol{M}$ は

$$\boldsymbol{M}=\begin{pmatrix}\boldsymbol{I} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\end{pmatrix}$$

です. (3.1b)を微分し (3.1a)を使うと

$$\boldsymbol{0}=\boldsymbol{g}_y(\boldsymbol{y},\boldsymbol{z})\boldsymbol{f}(\boldsymbol{y},\boldsymbol{z})+\boldsymbol{g}_z(\boldsymbol{y},\boldsymbol{z})\boldsymbol{z}' \qquad (3.1c)$$

となります. ここで $\boldsymbol{g}_y(\boldsymbol{y},\boldsymbol{z}),\boldsymbol{g}_z(\boldsymbol{y},\boldsymbol{z})$ は各々 $\partial\boldsymbol{g}(\boldsymbol{y},\boldsymbol{z})/\partial\boldsymbol{y},\partial\boldsymbol{g}(\boldsymbol{y},\boldsymbol{z})/\partial\boldsymbol{z}$ を意味します.

もし $\boldsymbol{z}'$ の係数である $\boldsymbol{g}_z(\boldsymbol{y},\boldsymbol{z})$ が解の近傍で正則ならば,

$$\boldsymbol{z}'=-\boldsymbol{g}_z^{-1}(\boldsymbol{y},\boldsymbol{z})\boldsymbol{g}_y(\boldsymbol{y},\boldsymbol{z})\boldsymbol{f}(\boldsymbol{y},\boldsymbol{z})$$

が得られます. この場合, $\boldsymbol{y},\boldsymbol{z}$ 共に指数 1 となります. 初期値 $\boldsymbol{y}_0,\boldsymbol{z}_0$ は(3.1b)を満たす必要があります.

b) 指数 2 の問題

$$\boldsymbol{y}'=\boldsymbol{f}(\boldsymbol{y},\boldsymbol{z}) \qquad (3.2a)$$

$$\boldsymbol{0}=\boldsymbol{g}(\boldsymbol{y}) \qquad (3.2b)$$

のように代数式の中に $\boldsymbol{z}$ が含まれていない場合を考えます. ここでも

$$\boldsymbol{M}=\begin{pmatrix}\boldsymbol{I} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\end{pmatrix}$$

です. (3.2b)を微分すると

$$\boldsymbol{0}=\boldsymbol{g}_y(\boldsymbol{y})\boldsymbol{f}(\boldsymbol{y},\boldsymbol{z}) \qquad (3.2c)$$

が得られます.さらに (3.2c)を微分すると $\boldsymbol{z}'$ の係数は

$$\boldsymbol{g}_y(\boldsymbol{y})\boldsymbol{f}_z(\boldsymbol{y},\boldsymbol{z}) \qquad (3.2d)$$

となります.もし解の近傍で(3.2d)が正則であるならば, $\boldsymbol{y}$ は指数 1, $\boldsymbol{z}$ は指数 2 となります. 初期値 $\boldsymbol{y}_0,\boldsymbol{z}_0$ については (3.2b)だけでなく(3.2c)も満たすものである必要があります.

c) 指数 3 の問題

$$\boldsymbol{y}'=\boldsymbol{f}(\boldsymbol{y},\boldsymbol{z}) \qquad (3.3a)$$

$$\boldsymbol{z}'=\boldsymbol{k}(\boldsymbol{y},\boldsymbol{z},\boldsymbol{u}) \qquad (3.3b)$$

$$\boldsymbol{0}=\boldsymbol{g}(\boldsymbol{y}) \qquad (3.3c)$$

を考えます. ここで解ベクトル $\boldsymbol{y},\boldsymbol{z},\boldsymbol{u}$ の長さの和は n とします. また $\boldsymbol{M}$ は

$$\boldsymbol{M}=\begin{pmatrix}\boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0}\end{pmatrix}$$

となります. (3.3c)を微分し, (3.3a)を使うと

$$\boldsymbol{0}=\boldsymbol{g}_y\boldsymbol{f} \qquad (3.3d)$$

を得ます. これを微分して(3.3a,b) を使うと

$$\mathbf{0} = \boldsymbol{g}_{yy}(\boldsymbol{f}, \boldsymbol{f}) + \boldsymbol{g}_y \boldsymbol{f}_y \boldsymbol{f} + \boldsymbol{g}_y \boldsymbol{f}_z \boldsymbol{k} \qquad (3.3e)$$

を得ます. ここで右辺第一項は, 行列 $\boldsymbol{g}_y$ を微分して得られる行列 $\boldsymbol{g}_{yy}$ とベクトル $\boldsymbol{f}$ の積を意味します. 続いて(3.3e)を微分すると $\boldsymbol{u}'$ が現れますがその係数である $\boldsymbol{g}_y \boldsymbol{f}_z \boldsymbol{k}_u$ が解の近傍で正則であるなら, (3.3a,b,c)の問題では, $\boldsymbol{y}$ は指数 1, $\boldsymbol{z}$ は指数 2, $\boldsymbol{u}$ は指数 3 となります. 初期値 $\boldsymbol{y}_0, \boldsymbol{z}_0$ および $\boldsymbol{u}_0$ は(3.3 c,d,e)を満たすものである必要があります.

b. 使用例

■使用例 1： $\boldsymbol{y}' = \boldsymbol{f}(x, \boldsymbol{y})$ の形の問題

常微分方程式の初期値問題,

$$y_1' = y_2$$
$$y_2' = \frac{((1-y_1^2)y_2 - y_1)}{\varepsilon}, \varepsilon = 10^{-6}$$
$$y_1(0) = 2, y_2(0) = 0$$

において, $x = 1,2,\ldots,11$ での解を求め, 印刷するプログラムを以下に示します.この問題のヤコビ行列 $\partial \boldsymbol{f} / \partial \boldsymbol{y}$ は

$$\begin{pmatrix} \frac{\partial f_1}{\partial y_1} & \frac{\partial f_1}{\partial y_2} \\ \frac{\partial f_2}{\partial y_1} & \frac{\partial f_2}{\partial y_2} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ (-2y_1y_2 - 1)/\varepsilon & (1-y_1^2)/\varepsilon \end{pmatrix}$$

であり, 引数 JAC に相当するサブルーチン JVPOL で右辺の行列を 2 次元配列 DFY に代入しています.

```
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (ND=2,LWORK=4*ND*ND+12*ND+20,LIWORK=3*ND+20)
      DIMENSION Y(ND),WORK(LWORK),IWORK(LIWORK)
      DIMENSION RPAR(2)

      EXTERNAL FVPOL,JVPOL,SOLOUT
      RPAR(1)=1.0D-6
      RPAR(2)=0.2D0
      N=ND
      IJAC=1
      MLJAC=N
      IMAS=0
      IOUT=1
      X=0.0D0
      Y(1)=2.0D0
```

```
      Y(2)=-0.66D0
      XEND=11.0D0
      RTOL=1.0D-4
      ATOL=1.0D0*RTOL
      ITOL=0
      H=1.0D-6
      DO I=1,20
         IWORK(I)=0
         WORK(I)=0.D0
      END DO
      CALL DM_VRADAU5(N,FVPOL,X,Y,XEND,H,
     &                RTOL,ATOL,ITOL,
     &                JVPOL,IJAC,MLJAC,MUJAC,
     &                FVPOL,IMAS,MLMAS,MUMAS,
     &                SOLOUT,IOUT,
     &                WORK,LWORK,IWORK,LIWORK,
     &   RPAR,IPAR,ICON)
      WRITE(6,*) 'ICON=', ICON
      WRITE (6,99) X,Y(1),Y(2)
 99   FORMAT(1X,'X =',F5.2,'    Y =',2E18.10)
      STOP
      END
C
C
      SUBROUTINE SOLOUT (NR,XOLD,X,Y,CONT,LRC,N,RPAR,IPAR,IRTRN,
     &   WORK2,IWORK2)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Y(N),CONT(LRC),RPAR(*)
      IF (NR.EQ.1) THEN
         WRITE (6,99) X,Y(1),Y(2),NR-1
      ELSE
 10      CONTINUE
         IF (X.GE.RPAR(2)) THEN
C --- CONTINUOUS OUTPUT FOR RADAU5
      WRITE (6,99) RPAR(2),DM_VCONTR5(1,RPAR(2),CONT,LRC,WORK2,
     &  IWORK2),DM_VCONTR5(2,RPAR(2),CONT,LRC,WORK2,IWORK2),
     &  NR-1
            RPAR(2)=RPAR(2)+0.2D0
            GOTO 10
         END IF
      END IF
 99   FORMAT(1X,'X =',F5.2,'    Y =',2E18.10,'    NSTEP =',I4)
```

```
      RETURN
      END
C
C
      SUBROUTINE FVPOL(N,X,Y,F,RPAR,IPAR)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Y(N),F(N),RPAR(*)
      F(1)=Y(2)
      F(2)=((1-Y(1)**2)*Y(2)-Y(1))/RPAR(1)
      RETURN
      END
C
C
      SUBROUTINE JVPOL(N,X,Y,DFY,LDFY,RPAR,IPAR)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Y(N),DFY(LDFY,N),RPAR(*)
      DFY(1,1)=0.0D0
      DFY(1,2)=1.0D0
      DFY(2,1)=(-2.0D0*Y(1)*Y(2)-1.0D0)/RPAR(1)
      DFY(2,2)=(1.0D0-Y(1)**2)/RPAR(1)
      RETURN
      END
```

■使用例 2： $y' = f(x, y)$ の形の問題. ヤコビ行列がバンド行列の場合.

以下に示す偏微分方程式を考えます. t は時刻, x はスカラーの空間変数とします.

$$\frac{\partial u}{\partial t} = A + u^2 v - (B+1)u + \alpha \frac{\partial^2 u}{\partial x^2}$$

$$\frac{\partial v}{\partial t} = Bu - u^2 v + \alpha \frac{\partial^2 v}{\partial x^2}$$

$$0 \le x \le 1, A = 1, B = 3, \alpha = 1/50$$

x についての境界条件 ： $u(0,t) = u(1,t) = 1, v(0,t) = v(1,t) = 3$

t についての初期条件 ： $u(x,0) = 1 + \frac{1}{2}\sin(2\pi x), v(x,0) = 3$

空間変数に関する 2 階微分を, $x_i = i/(N+1)$ $(1 \le i \le N)$, $\Delta x = 1/(N+1)$ から定まる N 点格子上の差分で置き換えることにより, 上記の偏微分方程式は以下に示す, 独立変数が t で $u_i = u(t, x_i), v_i = (t, x_i)$ を未知関数とする 2N 元の常微分方程式の初期値問題となります. ここで u' などは t についての微分を表すものとします.

$$u_i' = 1 + u_i^2 v_i - 4u_i + \alpha / (\Delta x)^2 \left(u_{i-1} - 2u_i + u_{i+1}\right)$$

$$v_i' = 3u_i - u_i^2 v_i + \alpha / (\Delta x)^2 \left(v_{i-1} - 2v_i + v_{i+1}\right)$$

$$u_0(t) = u_{N+1}(t) = 1, v_0(t) = v_{N+1}(t) = 3$$

$$u_i(0) = 1 + \frac{1}{2}\sin(2\pi x_i), v_i(0) = 3, i = 1,2,\ldots,N$$

この問題を本サブルーチンで解くにあたり, 解ベクトル $\boldsymbol{y}$ を
$\boldsymbol{y} = (u_1, v_1, u_2, v_2, \ldots, u_N, v_N)^{\mathrm{T}}$ とします. すると, ヤコビ行列は下バンド幅, 上バンド幅が共に 2 のバンド行列になります. このため引数 MLJAC と MUJAC は 2 と設定します. 以下のプログラムでは N=500, XEND=10, IOUT=0 (途中の解を SOLOUT へ渡さない) とし, XEND での解ベクトルの一部の成分を印刷しています.

```
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (ND=1000,NL=2,NU=2)
      PARAMETER (LWORK=(7*NL+4*NU+16)*ND+20,LIWORK=3*ND+20)
      DIMENSION Y(ND),WORK(LWORK),IWORK(LIWORK)
      DIMENSION RPAR(2)
      EXTERNAL FBRUS,JBRUS,SOLOUT
      PI=3.14159265358979324D0
      N=500
      N2=2*N
      USDELQ=(DBLE(N+1))**2
      GAMMA=0.02D0*USDELQ
      GAMMA2=2.D0*GAMMA
      RPAR(1)=GAMMA
      RPAR(2)=GAMMA2
      X=0.D0
      XEND=10.D0
      ANP1=N+1
      DO 1 I=1,N
      XI=I/ANP1
      Y(2*I)=3.D0
    1 Y(2*I-1)=1.D0+0.5D0*DSIN(2.D0*PI*XI)
      IJAC=1
C   Jacobian is a banded matrix.
      MLJAC=NL
      MUJAC=NU
      IMAS=0
C   Output Routine is not used.
      IOUT=0
      RTOL=1.0D-6
      ATOL=RTOL
      ITOL=0
      H=1.0D-6
      DO I=1,20
```

```
            WORK(I)=0.D0
            IWORK(I)=0
         END DO
         CALL DM_VRADAU5(N2,FBRUS,X,Y,XEND,H,
     &                 RTOL,ATOL,ITOL,
     &                 JBRUS,IJAC,MLJAC,MUJAC,
     &                 FBRUS,IMAS,MLMAS,MUMAS,
     &                 SOLOUT,IOUT,
     &                 WORK,LWORK,IWORK,LIWORK,
     &  RPAR,IPAR,ICON)
         WRITE(6,*) 'ICON=',ICON
         WRITE(6,99) Y(1),Y(2),Y(N2-1),Y(N2)
 99      FORMAT(1X,4F18.10)
         STOP
         END
C
         SUBROUTINE SOLOUT (NR,XOLD,X,Y,CONT,LRC,N,RPAR,IPAR,IRTRN,
     &  WORK2,IWORK2)
         RETURN
         END
C
         SUBROUTINE FBRUS(N2,X,Y,F,RPAR,IPAR)
         IMPLICIT REAL*8 (A-H,O-Z)
         DIMENSION Y(N2),F(N2),RPAR(*)
         N=N2/2
         GAMMA=RPAR(1)
         GAMMA2=RPAR(2)
         I=1
         IU=2*I-1
         IV=2*I
         UI=Y(IU)
         VI=Y(IV)
             UIM=1.D0
             VIM=3.D0
             UIP=Y(IU+2)
             VIP=Y(IV+2)
         PROD=UI*UI*VI
         F(IU)=1.D0+PROD-4.D0*UI+GAMMA*(UIM-2.D0*UI+UIP)
         F(IV)=3.D0*UI-PROD+GAMMA*(VIM-2.D0*VI+VIP)
         DO 5 I=2,N-1
         IU=2*I-1
         IV=2*I
```

```
      UI=Y(IU)
      VI=Y(IV)
          UIM=Y(IU-2)
          VIM=Y(IV-2)
          UIP=Y(IU+2)
          VIP=Y(IV+2)
      PROD=UI*UI*VI
      F(IU)=1.D0+PROD-4.D0*UI+GAMMA*(UIM-2.D0*UI+UIP)
      F(IV)=3.D0*UI-PROD+GAMMA*(VIM-2.D0*VI+VIP)
    5 CONTINUE
      I=N
      IU=2*I-1
      IV=2*I
      UI=Y(IU)
      VI=Y(IV)
          UIM=Y(IU-2)
          VIM=Y(IV-2)
          UIP=1.D0
          VIP=3.D0
      PROD=UI*UI*VI
      F(IU)=1.D0+PROD-4.D0*UI+GAMMA*(UIM-2.D0*UI+UIP)
      F(IV)=3.D0*UI-PROD+GAMMA*(VIM-2.D0*VI+VIP)
      RETURN
      END
C
      SUBROUTINE JBRUS(N2,X,Y,DFY,LDFY,RPAR,IPAR)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Y(N2),DFY(LDFY,N2),RPAR(*)
      N=N2/2
      GAMMA=RPAR(1)
      GAMMA2=RPAR(2)
      DO 1 I=1,N
      IU=2*I-1
      IV=2*I
      UI=Y(IU)
      VI=Y(IV)
      UIVI=UI*VI
      UI2=UI*UI
      DFY(3,IU)=2.D0*UIVI-4.D0-GAMMA2
      DFY(2,IV)=UI2
      DFY(4,IU)=3.D0-2.D0*UIVI
      DFY(3,IV)=-UI2-GAMMA2
```

```
      DFY(2,IU)=0.D0
      DFY(4,IV)=0.D0
1     CONTINUE
      DO 2 I=1,N2-2
      DFY(1,I+2)=GAMMA
      DFY(5,I)=GAMMA
2     CONTINUE
      RETURN
      END
```

■使用例 3： $\boldsymbol{y}'' = \boldsymbol{f}(x, \boldsymbol{y}, \boldsymbol{y}')$ の形の 2 階の問題.

長方形領域 $\Omega = \{(x, y); 0 \le x \le 2, 0 \le y \le 4/3\}$ で定義された以下に示す偏微分方程式を考えます.

$$\frac{\partial^2 u}{\partial t^2} + \omega \frac{\partial u}{\partial t} + \sigma \Delta \Delta u = f(x, y, t), \text{ ここで } \Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

境界条件: $u|_{\partial\Omega} = 0, \Delta u|_{\partial\Omega} = 0$

初期条件: $u(x, y, 0) = 0, \dfrac{\partial u}{\partial t}(x, y, 0) = 0$

長方形領域を x 方向, y 方向共に幅 $h = 2/9$ で分割してできる 8×5 個の内側格子点での解を求めます. まず,

$$x_i = ih, y_j = jh, i = 1,2,\ldots,8, j = 1,2,\ldots,5$$
$$u_{ij} = u(x_i, y_j, t)$$

として空間変数に関する微分を差分で置き換えます. 次に $v_{ij} = u'_{ij}$ と置けば,

$$u''_{ij} = v_{ij}$$
$$v'_{ij} = -\omega v_{ij} - \frac{\sigma}{h^4}(20u_{ij} - 8u_{i-1\,j} - 8u_{i+1\,j} + 2u_{i+1\,j+1} + 2u_{i+1\,j-1}$$
$$+ 2u_{i-1\,j-1} + 2u_{i-1\,j+1} + u_{i-2\,j} + u_{i+2\,j} + u_{i\,j-2} + u_{i\,j+2}) + f(x_i, y_j, t)$$

なる常微分方程式が得られます. ここで $k = i + 8(j-1)$ なる対応で (i, j) から k にマッピングし, $y_k = u_{ij}, y_{k+40} = v_{ij}$ と置けば,

$$(y_1, y_2, \ldots, y_{40}, y_{41}, \ldots, y_{80})^{\mathrm{T}}$$

を解ベクトルとする常微分方程式となります. これを本サブルーチンで解く場合, IWORK(9)=40 と与えて, ヤコビ行列は非自明な部分だけを与えれば効率よく積分することができます. 以下のプログラムでは,

$\omega = 1000, \sigma = 100$

$$f(x,y,t)=\begin{cases}200(\mathrm{e}^{-5(t-x-2)^2}+\mathrm{e}^{-5(t-x-5)^2}) & y=y_2\text{または}y_4\text{のとき}\\ 0 & y\neq y_2\text{かつ}y\neq y_4\text{のとき}\end{cases}$$

とします. また非自明なヤコビ行列はバンド行列 (上バンド幅, 下バンド幅共に 2×8) となります.

```
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER(MX=8,MY=5)
      PARAMETER (ND=2*MX*MY,LWORK=4*ND*ND+12*ND+20,LIWORK=3*ND+20)
      DIMENSION Y(ND),WORK(LWORK),IWORK(LIWORK)
      EXTERNAL FPLATE,JPLATSB,SOLOUT
      DIMENSION RPAR(4),IPAR(7)

      NX=MX
      NY=MY
      NACHS1=2
      NACHS2=4
      NXM1=NX-1
      NYM1=NY-1
      NDEMI=NX*NY
      OMEGA=1000.D0
      STIFFN=100.D0
      WEIGHT=200.D0
      DENOM=NX+1
      DELX=2.D0/DENOM
      USH4=1.D0/(DELX**4)
      FAC=STIFFN*USH4
      N=ND
      IMAS=0
C --- OUTPUT ROUTINE IS USED DURING INTEGRATION
      IOUT=1
C --- INITIAL VALUES
      X=0.0D0
      DO  I=1,N
      Y(I)=0.D0
      END DO
C --- REQUIRED TOLERANCE
        RTOL=1.0D-6
        ATOL=RTOL*1.0D-3
        ITOL=0
C --- INITIAL STEP SIZE
        H=1.0D-2
```

```
C --- SET DEFAULT VALUES
        DO I=1,20
           WORK(I)=0.D0
           IWORK(I)=0
        END DO
C --- SECOND ORDER OPTION AND BANDED
           IJAC=1
           IWORK(9)=N/2
           MLJAC=2*MX
           MUJAC=2*MX
C --- ENDPOINT OF INTEGRATION
        XEND=7.D0
C --- COMMUNICATION VALUES
      IPAR(1)=NX
      IPAR(2)=NXM1
      IPAR(3)=NY
      IPAR(4)=NYM1
      IPAR(5)=NDEMI
      IPAR(6)=NACHS1
      IPAR(7)=NACHS2
      RPAR(1)=OMEGA
      RPAR(2)=DELX
      RPAR(3)=FAC
      RPAR(4)=WEIGHT

C --- CALL OF THE SUBROUTINE RADAU5
        CALL DM_VRADAU5(N,FPLATE,X,Y,XEND,H,
     &                  RTOL,ATOL,ITOL,
     &                  JPLATSB,IJAC,MLJAC,MUJAC,
     &                  FPLATE,IMAS,MLMAS,MUMAS,
     &                  SOLOUT,IOUT,
     &                  WORK,LWORK,IWORK,LIWORK,
     &   RPAR,IPAR,ICON)
      WRITE(6,*) 'ICON=',ICON
      DO  K=1,N
      WRITE (6,*) Y(K)
      END DO
      STOP
      END
C
      SUBROUTINE SOLOUT (NR,XOLD,X,Y,CONT,LRC,N,RPAR,IPAR,IRTRN,
     &  WORK2,IWORK2)
```

```
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Y(N),CONT(LRC)
      NHALF=N/2
      WRITE (6,991) X,NHALF,Y(1),Y(NHALF),NR-1
 991  FORMAT(1X,'X =',F9.5,' Y(1) and Y(',I3,')=',2F18.10,
     &  ' NSTEP =',I4)
      RETURN
      END
C
      SUBROUTINE FPLATE (N, X, Y, F, RPAR, IPAR)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Y(N), F(N)
      DIMENSION RPAR(*),IPAR(*)
      NX=IPAR(1)
      NXM1=IPAR(2)
      NY=IPAR(3)
      NYM1=IPAR(4)
      NDEMI=IPAR(5)
      NACHS1=IPAR(6)
      NACHS2=IPAR(7)
      OMEGA=RPAR(1)
      DELX=RPAR(2)
      FAC=RPAR(3)
      WEIGHT=RPAR(4)

      DO 1 I=1,NX
      DO 1 J=1,NY
      K=I+NX*(J-1)
C -------- SECOND DERIVATIVE ----
      F(K)=Y(K+NDEMI)
C ------ CENTRAL POINT---
      UC=16.D0*Y(K)
      IF(I.GT.1)THEN
         UC=UC+Y(K)
         UC=UC-8.D0*Y(K-1)
      END IF
      IF(I.LT.NX)THEN
         UC=UC+Y(K)
         UC=UC-8.D0*Y(K+1)
      END IF
      IF(J.GT.1)THEN
         UC=UC+Y(K)
```

```
         UC=UC-8.D0*Y(K-NX)
      END IF
      IF(J.LT.NY)THEN
         UC=UC+Y(K)
         UC=UC-8.D0*Y(K+NX)
      END IF
      IF(I.GT.1 .AND.J.GT.1 )UC=UC+2.D0*Y(K-NX-1)
      IF(I.LT.NX.AND.J.GT.1 )UC=UC+2.D0*Y(K-NX+1)
      IF(I.GT.1 .AND.J.LT.NY)UC=UC+2.D0*Y(K+NX-1)
      IF(I.LT.NX.AND.J.LT.NY)UC=UC+2.D0*Y(K+NX+1)
      IF(I.GT.2)UC=UC+Y(K-2)
      IF(I.LT.NXM1)UC=UC+Y(K+2)
      IF(J.GT.2)UC=UC+Y(K-2*NX)
      IF(J.LT.NYM1)UC=UC+Y(K+2*NX)
      IF(J.EQ.NACHS1.OR.J.EQ.NACHS2)THEN
         XI=I*DELX
         FORCE=EXP(-5.D0*(X-XI-2.D0)**2)+EXP(-5.D0*(X-XI-5.D0)**2)
      ELSE
         FORCE=0.D0
      END IF
      F(K+NDEMI)=-OMEGA*Y(K+NDEMI)-FAC*UC+FORCE*WEIGHT
1     CONTINUE
      RETURN
      END

      SUBROUTINE JPLATSB(N,X,Y,DFY,LDFY,RPAR,IPAR)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Y(N),DFY(LDFY,N)
      DIMENSION RPAR(*),IPAR(*)
      NX=IPAR(1)
      NXM1=IPAR(2)
      NY=IPAR(3)
      NYM1=IPAR(4)
      NDEMI=IPAR(5)
      OMEGA=RPAR(1)
      FAC=RPAR(3)

      DO 1 I=1,LDFY
      DO 1 J=1,N
1     DFY(I,J)=0.D0
      MU=2*NX+1
      FAC2=FAC*2.0D0
```

```
      FAC8=FAC*8.0D0
      FAC16=FAC*16.0D0
      DO 2 I=1,NX
      DO 2 J=1,NY
      K=I+NX*(J-1)
      DFY(MU,K)=-FAC16
      IF(I.GT.1)THEN
         DFY(MU,K)=DFY(MU,K)-FAC
         DFY(MU+1,K-1)=FAC8
      END IF
      IF(I.LT.NX)THEN
         DFY(MU,K)=DFY(MU,K)-FAC
         DFY(MU-1,K+1)=FAC8
      END IF
      IF(J.GT.1)THEN
         DFY(MU,K)=DFY(MU,K)-FAC
         DFY(MU+NX,K-NX)=FAC8
      END IF
      IF(J.LT.NY)THEN
         DFY(MU,K)=DFY(MU,K)-FAC
         DFY(MU-NX,K+NX)=FAC8
      END IF
      IF(I.GT.1 .AND.J.GT.1 )DFY(MU+NX+1,K-NX-1)=-FAC2
      IF(I.LT.NX.AND.J.GT.1 )DFY(MU+NX-1,K-NX+1)=-FAC2
      IF(I.GT.1 .AND.J.LT.NY)DFY(MU-NX+1,K+NX-1)=-FAC2
      IF(I.LT.NX.AND.J.LT.NY)DFY(MU-NX-1,K+NX+1)=-FAC2
      IF(I.GT.2)DFY(MU+2,K-2)=-FAC
      IF(I.LT.NXM1)DFY(MU-2,K+2)=-FAC
      IF(J.GT.2)DFY(MU+2*NX,K-2*NX)=-FAC
      IF(J.LT.NYM1)DFY(MU-2*NX,K+2*NX)=-FAC
      DFY(MU,K+NDEMI)= -OMEGA
    2 CONTINUE
      RETURN
      END
```

■使用例 4： $\boldsymbol{My}' = \boldsymbol{f}(x, \boldsymbol{y})$ の形の微分代数方程式.

t を独立変数とする 8 つの未知関数 $y_1, y_2, \ldots, y_8$ があって, 以下の方程式を満たすものとします.

$$C_5(y_2' - y_1') = y_1/R_9$$
$$-C_5(y_2' - y_1') = \alpha f(y_4 - y_3) - U_b/R_8 + y_2/R_8$$
$$-C_4 y_3' = y_3/R_7 - f(y_4 - y_3)$$
$$C_3(y_5' - y_4') = -U_b/R_6 + y_4(1/R_5 + 1/R_6) + (1-\alpha) f(y_4 - y_3)$$
$$-C_3(y_5' - y_4') = -U_b/R_4 + y_5/R_4 + \alpha f(y_7 - y_6)$$
$$-C_2 y_6' = y_6/R_3 - f(y_7 - y_6)$$
$$C_1(y_8' - y_7') = -U_b/R_2 + y_7(1/R_1 + 1/R_2) + (1-\alpha) f(y_7 - y_6)$$
$$C_1(y_7' - y_8') = y_8/R_0 - U_e(t)/R_0$$

ここで

$$C_k = k \cdot 10^{-6}, k = 1,2,\ldots,5$$
$$R_0 = 1000, R_k = 9000, k = 1,2,\ldots,9$$
$$f(y_i - y_j) = \beta(\mathrm{e}^{(y_i - y_j)/U_F} - 1)$$
$$U_F = 0.026, \alpha = 0.99, \beta = 10^{-6}, U_b = 6$$
$$U_e(t) = 0.1 \cdot \sin(200\pi t)$$

$\boldsymbol{y} = (y_1, y_2, \ldots, y_8)^{\mathrm{T}}$ とすれば, 8 つの方程式の左辺は **M*y***'と表せます. ここで, ***M*** は次のような三重対角行列です.

$$\boldsymbol{M} = \begin{pmatrix} -C_5 & C_5 & & & & & & \\ C_5 & -C_5 & & & & & & \\ & & -C_4 & & & & & \\ & & & -C_3 & C_3 & & & \\ & & & C_3 & -C_3 & & & \\ & & & & & -C_2 & & \\ & & & & & & -C_1 & C_1 \\ & & & & & & C_1 & -C_1 \end{pmatrix}$$

M は明らかに階数 5 の特異行列となり, したがって方程式は微分代数方程式となります. かつ, 指数 1 の微分代数方程式です.

t=0 から t=0.2 まで積分するとします. このとき初期値 ***y***(0)は上記方程式の 8 つの右辺を成分とするベクトルが ***M*** の像空間に入るように選べば無矛盾な解を得ることができます. その初期値は次のとおりです.

$$y_1(0) = 0, y_2(0) = U_b - y_1(0) \cdot R_8/R_9, y_3(0) = y_4(0) = U_b/(R_6/R_5 + 1)$$
$$y_5(0) = U_b, y_6(0) = y_7(0) = U_b/(R_2/R_1 + 1), y_8(0) = 0$$

8 つの方程式の右辺を成分とするベクトルのヤコビ行列は上バンド幅が 2, 下バンド幅が 1 のバンド行列となります. なお, この微分代数方程式では解の成分はすべて指標 1 であることが証明できます. 以下のプログラムでは, 時間幅 0.0025 ごとの解を印刷しています.

```
      IMPLICIT REAL*8 (A-H,O-Z)
```

```
      PARAMETER (ND=8,LJAC=4,LMAS=3,LE=5)
      PARAMETER (LWORK=ND*(LJAC+LMAS+3*LE+12)+20,LIWORK=3*ND+20)
      DIMENSION Y(ND),WORK(LWORK),IWORK(LIWORK),RPAR(16)
      EXTERNAL FAMPL,JBAMPL,BBAMPL,SOLOUT
      UE=0.1D0
        RPAR(1)=UE
      UB=6.0D0
        RPAR(2)=UB
      UF=0.026D0
        RPAR(3)=UF
      ALPHA=0.99D0
        RPAR(4)=ALPHA
      BETA=1.0D-6
        RPAR(5)=BETA
      R0=1000.0D0
        RPAR(6)=R0
      R1=9000.0D0
        RPAR(7)=R1
      R2=9000.0D0
        RPAR(8)=R2
      R3=9000.0D0
        RPAR(9)=R3
      R4=9000.0D0
        RPAR(10)=R4
      R5=9000.0D0
        RPAR(11)=R5
      R6=9000.0D0
        RPAR(12)=R6
      R7=9000.0D0
        RPAR(13)=R7
      R8=9000.0D0
        RPAR(14)=R8
      R9=9000.0D0
        RPAR(15)=R9
      RPAR(16)=0.0025D0
      N=8
      IJAC=1
      MLJAC=1
      MUJAC=2
      IMAS=1
      MLMAS=1
      MUMAS=1
```

```
      IOUT=1
      X=0.0D0
      Y(1)=0.D0
      Y(2)=UB-Y(1)*R8/R9
      Y(3)=UB/(R6/R5+1.D0)
      Y(4)=UB/(R6/R5+1.D0)
      Y(5)=UB
      Y(6)=UB/(R2/R1+1.D0)
      Y(7)=UB/(R2/R1+1.D0)
      Y(8)=0.D0
      XEND=0.2D0
      RTOL=1.0D-5
      ATOL=1.0D-6*RTOL
      ITOL=0
      H=1.0D-6
      DO 10 I=1,20
      IWORK(I)=0
 10   WORK(I)=0.D0
      CALL DM_VRADAU5(N,FAMPL,X,Y,XEND,H,
     &                RTOL,ATOL,ITOL,
     &                JBAMPL,IJAC,MLJAC,MUJAC,
     &                BBAMPL,IMAS,MLMAS,MUMAS,
     &                SOLOUT,IOUT,
     &                WORK,LWORK,IWORK,LIWORK,RPAR,IPAR,ICON)
      WRITE(6,*) 'ICON=',ICON
      WRITE (6,99) X,Y(1),Y(2)
 99   FORMAT(1X,'X =',F7.4,'    Y =',2E18.10)
      STOP
      END
C
C
      SUBROUTINE SOLOUT (NR,XOLD,X,Y,CONT,LRC,N,RPAR,IPAR,IRTRN,
     &  WORK2,IWORK2)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Y(N),CONT(LRC),RPAR(*)
      IF (NR.EQ.1) THEN
         WRITE (6,99) X,Y(1),Y(2),NR-1
      ELSE
 10      CONTINUE
         IF (X.GE.RPAR(16)) THEN
         WRITE (6,99) RPAR(16),
     &     DM_VCONTR5(1,RPAR(16),CONT,LRC,WORK2,IWORK2),
```

```
     &     DM_VCONTR5(2,RPAR(16),CONT,LRC,WORK2,IWORK2),NR-1
           RPAR(16)=RPAR(16)+0.0025D0
           GOTO 10
           END IF
        END IF
 99     FORMAT(1X,'X =',F7.4,'    Y =',2E18.10,'    NSTEP =',I4)
        RETURN
        END
C
C
        SUBROUTINE FAMPL(N,X,Y,F,RPAR,IPAR)
        IMPLICIT REAL*8 (A-H,O-Z)
        REAL*8 Y(N),F(N),RPAR(*)
        UE=RPAR(1)
        UB=RPAR(2)
        UF=RPAR(3)
        ALPHA=RPAR(4)
        BETA=RPAR(5)
        R0=RPAR(6)
        R1=RPAR(7)
        R2=RPAR(8)
        R3=RPAR(9)
        R4=RPAR(10)
        R5=RPAR(11)
        R6=RPAR(12)
        R7=RPAR(13)
        R8=RPAR(14)
        R9=RPAR(15)
        W=2.D0*3.141592654D0*100.D0
        UET=UE*DSIN(W*X)
        FAC1=BETA*(DEXP((Y(4)-Y(3))/UF)-1.D0)
        FAC2=BETA*(DEXP((Y(7)-Y(6))/UF)-1.D0)
        F(1)=Y(1)/R9
        F(2)=(Y(2)-UB)/R8+ALPHA*FAC1
        F(3)=Y(3)/R7-FAC1
        F(4)=Y(4)/R5+(Y(4)-UB)/R6+(1.D0-ALPHA)*FAC1
        F(5)=(Y(5)-UB)/R4+ALPHA*FAC2
        F(6)=Y(6)/R3-FAC2
        F(7)=Y(7)/R1+(Y(7)-UB)/R2+(1.D0-ALPHA)*FAC2
        F(8)=(Y(8)-UET)/R0
        RETURN
        END
```

```
C
C
      SUBROUTINE JBAMPL(N,X,Y,DFY,LDFY,RPAR,IPAR)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 Y(N),DFY(LDFY,N),RPAR(*)
      UE=RPAR(1)
      UB=RPAR(2)
      UF=RPAR(3)
      ALPHA=RPAR(4)
      BETA=RPAR(5)
      R0=RPAR(6)
      R1=RPAR(7)
      R2=RPAR(8)
      R3=RPAR(9)
      R4=RPAR(10)
      R5=RPAR(11)
      R6=RPAR(12)
      R7=RPAR(13)
      R8=RPAR(14)
      R9=RPAR(15)
      FAC14=BETA*DEXP((Y(4)-Y(3))/UF)/UF
      FAC27=BETA*DEXP((Y(7)-Y(6))/UF)/UF
      DO  J=1,8
       DFY(1,J)=0.D0
       DFY(2,J)=0.D0
       DFY(4,J)=0.D0
      END DO
      DFY(3,1)=1.D0/R9
      DFY(3,2)=1.D0/R8
      DFY(2,3)=-ALPHA*FAC14
      DFY(1,4)=ALPHA*FAC14
      DFY(3,3)=1.D0/R7+FAC14
      DFY(2,4)=-FAC14
      DFY(3,4)=1.D0/R5+1.D0/R6+(1.D0-ALPHA)*FAC14
      DFY(4,3)=-(1.D0-ALPHA)*FAC14
      DFY(3,5)=1.D0/R4
      DFY(2,6)=-ALPHA*FAC27
      DFY(1,7)=ALPHA*FAC27
      DFY(3,6)=1.D0/R3+FAC27
      DFY(2,7)=-FAC27
      DFY(3,7)=1.D0/R1+1.D0/R2+(1.D0-ALPHA)*FAC27
      DFY(4,6)=-(1.D0-ALPHA)*FAC27
```

```
      DFY(3,8)=1.D0/R0
      RETURN
      END
C
      SUBROUTINE BBAMPL(N,B,LB,RPAR,IPAR)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 B(LB,N),RPAR(*)
      DO  I=1,8
       B(1,I)=0.D0
       B(3,I)=0.D0
      END DO
      C1=1.D-6
      C2=2.D-6
      C3=3.D-6
      C4=4.D-6
      C5=5.D-6
C
      B(2,1)=-C5
      B(1,2)=C5
      B(3,1)=C5
      B(2,2)=-C5
      B(2,3)=-C4
      B(2,4)=-C3
      B(1,5)=C3
      B(3,4)=C3
      B(2,5)=-C3
      B(2,6)=-C2
      B(2,7)=-C1
      B(1,8)=C1
      B(3,7)=C1
      B(2,8)=-C1
      RETURN
      END
```

(4)　手法概要

本サブルーチンは, スティフな常微分方程式あるいは微分代数方程式に対して安定的な手法である 3 段 5 次の陰的ルンゲ・クッタ法(文献[33],[34]では Radau IIA と定義している公式) を採用しています.

簡単のため, まず ***M=I*** の場合を考えます. いま x_0 からステップ幅 h で積分した x_1 における解 $\boldsymbol{y}_1$ を求める場面を考えます. ここでは x_0 は本サブルーチンでいう初期点ではなく, 1 ステップ積分するときの出発点とします. 3 段の陰的ルンゲ・クッタ公式は以下のように表せます.

$$\boldsymbol{g}_i = \boldsymbol{y}_0 + h\sum_{j=1}^{3} a_{ij}\,\boldsymbol{f}(x_0 + c_j h, \boldsymbol{g}_j) \qquad i = 1,2,3 \tag{4.1a}$$

$$\boldsymbol{y}_1 = \boldsymbol{y}_0 + h\sum_{j=1}^{3} b_j\,\boldsymbol{f}(x_0 + c_j h, \boldsymbol{g}_j) \tag{4.1b}$$

a_{ij}, c_j および b_j は公式の係数であり, 通常, 次のような表で表現されます.

c_1	a_{11}	a_{12}	a_{13}
c_2	a_{21}	a_{22}	a_{23}
c_3	a_{31}	a_{32}	a_{33}
	b_1	b_2	b_3

本サブルーチンで使う 3 段 5 次の Radau IIA の公式では係数は以下のとおりです.

$\frac{4-\sqrt{6}}{10}$	$\frac{88-7\sqrt{6}}{360}$	$\frac{296-169\sqrt{6}}{1800}$	$\frac{-2+3\sqrt{6}}{225}$
$\frac{4+\sqrt{6}}{10}$	$\frac{296+169\sqrt{6}}{1800}$	$\frac{88+7\sqrt{6}}{360}$	$\frac{-2-3\sqrt{6}}{225}$
1	$\frac{16-\sqrt{6}}{36}$	$\frac{16+\sqrt{6}}{36}$	$\frac{1}{9}$
	$\frac{16-\sqrt{6}}{36}$	$\frac{16+\sqrt{6}}{36}$	$\frac{1}{9}$

計算としては, 原理的には(4.1a)の $\boldsymbol{g}_1, \boldsymbol{g}_2$ および $\boldsymbol{g}_3$ についての(すなわち $3n$ 元の)非線形方程式を解いて(4.1b)に代入して $\boldsymbol{y}_1$ を計算するものです. ただし, 実際には計算誤差を少なくするために $\boldsymbol{z}_i = \boldsymbol{g}_i - \boldsymbol{y}_0$ なるベクトルを導入して $\boldsymbol{z}_i$ についての非線形代数方程式を解いており, さらに, (4.1b)で表すような $\boldsymbol{f}$ の計算を必要としない工夫を施しています.

非線形方程式は基本的にはニュートン法で解きます. ただし, ニュートン法における反復の度に $\boldsymbol{f}$ のヤコビ行列を計算し直すことはせず, ヤコビ行列は $x_0, \boldsymbol{y}_0$ における値を終始使うことにしています(これを簡約ニュートン反復(Simplified Newton Iterations)と呼びます).

ステップ幅 h は要求される解の精度を満たす範囲でなるべく大きく選ぶという戦略で制御されます. ここで計算解 $\boldsymbol{y}_1$ の誤差の見積もりが必要となりますが, その方法は, 5 次よりも 1 次だけ低い公式, すなわち 4 次の公式で, かつ埋め込み型(Embedded Formula)によりもう一つの計算解 $\hat{\boldsymbol{y}}_1$ を求め $\boldsymbol{y}_1$ との差に基づいて誤差を見積もります. 埋め込み型の公式とは, 同じ $\boldsymbol{g}_1, \boldsymbol{g}_2$ および $\boldsymbol{g}_3$ を使って, (4.1b)における b_j の代わりに別の $\hat{b}_j$ を使うというものです. こうすることで, 新たな $\boldsymbol{f}$ の計算を要しません.

(1.1) で $\boldsymbol{M} \neq \boldsymbol{I}$ の場合は Radau IIA 公式に現れるすべての $\boldsymbol{f}$ を形式的に $\boldsymbol{M}^{-1}\boldsymbol{f}$ で置き換えた式を作り, それに $\boldsymbol{M}$ を乗ずることにより (1.1) 向けの公式が得られます.
アルゴリズムの詳細については, “付録 1 参考文献一覧表” の[33],[34]を参照して下さい.[33]はルンゲ・クッタ法の一般論が,[34]はスティフ問題および微分代数方程式の解法の一つとして陰的ルンゲ・クッタ法の理論と実装が述べられています. また, [35], [36]はそれ

ぞれ[33],[34]の邦訳書です.

DM_VRANN3

正規乱数の生成
CALL DM_VRANN3(DAM, DSD, IX, DA, K, N, DWORK, NWORK, ICON)

(1) 機能

与えられた平均 m と標準偏差σを持った正規分布の密度関数(1.1)から, 擬似乱数を生成します.

$$f(x)=\frac{1}{\sigma\sqrt{2\pi}}\exp\left(\frac{-(x-m)^2}{2\sigma^2}\right) \tag{1.1}$$

(2) パラメタ

DAM .....入力. 正規分布の平均 m.
倍精度実数型.

DSD .....入力. 正規分布の標準偏差σ(>0).
倍精度実数型.

IX .....入力. 出発値
初回呼び出しでは, IX に正の値を与え, 2 回目以降の呼び出しでは返却値 0 のまま呼び出して下さい. 初回呼び出しで指定する出発値が異なると, 異なる乱数列が生成されます.
((3)使用上の注意 b. 注意①参照).
出力. 0.

DA.....出力. 各スレッドごとに生成される異なる擬似正規乱数.
DA(K, NUMT)なる倍精度実数型 2 次元配列.
ここで NUMT は本ルーチンで並列実行を行うスレッドの数. P 番目(0～NUMT - 1)のスレッドで生成された N 個の擬似乱数が DA(1:N, P+1)に返却されます.((3)使用上の注意 a. 注意⑥参照)

K.....入力. 配列 DA の 1 次元目の大きさ. (≥N)

N.....入力. DA に返却される各スレッドで生成される正規分布擬似乱数の個数.
((3)使用上の注意 b. 注意②参照).

DWORK .....作業領域. DWORK(NWORK, NUMT)なる倍精度実数型 2 次元配列.
本サブルーチンを繰り返し呼び出す場合は, 内容および本ルーチンで並列実行を行うスレッドの数 NUMT を変更しないでください. DWORK には, 本サブルーチンが現在の位置から再度呼び出される場合に必要な, すべての現情報が格納されています. (3)使用上の注意 a. 注意③, ⑥参照)

NWORK .....入力. 配列 DWORK の 1 次元目の大きさ. NWORK≥1156.

ICON.....出力. コンディションコード. 表 DM_VRANN3-1 参照.

表 DM_VRANN3-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし.	ー
30000	K<N または K<1	処理を打ち切る.
30001	NWORK が小さ過ぎた. IX<0, DSD≤0	
30002	内部エラー	
30003～30009	DWORK が変更されていた. 又は, 初回呼び出しで IX に 0 が与えられた.	
40000	IX が大き過ぎる.	

(3)　使用上の注意

a.　注意

①　出発値 IX について

確定的な計算(deterministic program)で擬似乱数列を生成する場合は, 何かランダムな入力が必要になります. したがって, 利用者は出発値 IX を与えなければなりません. この出発値は “たね(*seed*)” と呼ばれます. 本サブルーチンの初回呼び出しでは, “たね” IX は正整数である必要があります(例外事項については, ⑤を参照). 2 回目以降の呼び出しでは, IX には 0 を指定して下さい. これは同じ乱数列から続いて乱数を求めるためです. プログラミングを容易にするため, 本サブルーチンは初回呼び出し後 IX に 0 を返しています. 本サブルーチンは, *seed*=*seed*×OMP_GET_NUM_THREADS( )+OMP_GET_THREAD_NUM( )+1 のように, スレッド番号+1, OMP_GET_THREAD_NUM( )+1 を *seed* に付加しています. このように異なるスレッドで使用される *seed* は, 他のプロセッサーでの *seed* とは全く別であることが保証され, 10^{18} 未満の長さの乱数列であればオーバーラップすることはありません. ((4)手法概要参照).

②　パラメタ N について

本サブルーチンは最初の *seed*(IX)で定義される無限列から, 次の N 個の擬似乱数を返却します. N ≤ 0 の時は擬似乱数を返却しません.

効率良くするためには, 利用者は N を十分大きく(例えば N=100, 000)します. それは, サブルーチン呼び出しのオーバーヘッドが減少するためです. 本サブルーチンを連続的に呼び出す途中で, N が変更される場合は, 配列 DA の 1 次元目の大きさ K を最大の N の大きさ以上に設定する必要があります.

③　作業領域 DWORK について

DWORK は, 複数回本サブルーチンを呼び出す場合, 次の呼び出しのための情報を格納する作業領域として使用されます. したがって, 本サブルーチンを呼び出している間は, 呼び出し側のプログラムで DWORK の内容を変更してはいけません.

④　パラメタ NWORK について

DWORK(1,:) , ... , DWORK(NWORK,:)は, 本サブルーチンによって使用されます. NWORK は, 本サブルーチンの各呼び出しで変えないでください. NWORK には

少なくとも 1156 以上, また十分大きな数(例えば NWORK=100, 000)を与えると効率的です.

⑤ 同じ乱数の再生成について
DWORK(1,:) , ... , DWORK(NWORK,:)が保存される場合, その DWORK を再使用し, IX=0 として本サブルーチンを呼び出すことにより, (DWORK が保存された時点からの)同じ乱数列が再度生成されます.

⑥ 本サブルーチンを並列実行するスレッド数 NUMT は, 環境変数 OMP_NUM_THREADS または実行環境ルーチン OMP_SET_NUM_THREADS( )で指定します.
実行環境ルーチン OMP_SET_NUM_THREADS( )で指定するときは, 2 回目以降の呼び出しの直前でも初回呼び出しと同じスレッド数を OMP_SET_NUM_THREADS( )で指定して下さい.

b. 使用例
10, 000, 000 × 4 個の擬似正規乱数を生成し, 平均と標準偏差を計算します.

```
C     ** EXAMPLE **
      PARAMETER (NUMT=4)
      PARAMETER (NRAN=10000000)
      PARAMETER (NSEED=12345)
      PARAMETER (NWMAX=100000)
      PARAMETER (NBUF=120000,K=NBUF)
      DOUBLE PRECISION DA(K,NUMT)
      DOUBLE PRECISION DWORK(NWMAX,NUMT)
      DOUBLE PRECISION DSUM,DSUM2,DSSUM,DSSUM2
      DOUBLE PRECISION DMEAN,DSIG
      DOUBLE PRECISION DAM,DSD
      INTEGER NTOT
C     Initialize ix,n and nwork
      IX=NSEED
      WRITE (*,*)' Seed ',IX
      DAM=0.0D0
      DSD=1.0D0
      WRITE (*,*)' Mean ',DAM
      WRITE (*,*)' Standard deviation ',DSD
      N=NBUF
      NWORK=NWMAX
      DSUM=0.0D0
      DSSUM=0.0D0
C     ngen counts down to 0
      NGEN=NRAN
      NTOT=NRAN*NUMT
```

```
C     Generate ngen numbers
C     with maximum NBUF at a time.
      KRPT=(NRAN+NBUF-1)/NBUF
      WRITE (*,*)' Generating ',NTOT,' numbers'
      WRITE (*,*)' with ',KRPT,
     $   ' calls to dm-vrann3 on ',NUMT,' threads'
      CALL OMP_SET_NUM_THREADS(NUMT)
      DO 20 IZ=1,KRPT
      N=MIN0(NBUF,NGEN)
      CALL DM_VRANN3(DAM,DSD,IX,DA,K,N,DWORK,NWORK,ICON)
      IF (ICON.NE.0) WRITE (*,*)' ICON ',ICON
C     Accumulate sum of numbers
      DSUM2=0.0D0
      DO 30 J=1,NUMT
      DO 10 I=1,N
      DSUM2=DSUM2+DA(I,J)
  10  CONTINUE
  30  CONTINUE
C     Accumulate sum of numbers globally.
      DSSUM2=0.0D0
      DO 40 J=1,NUMT
      DO 50 I=1,N
       DSSUM2=DSSUM2+DA(I,J)*DA(I,J)
 50   CONTINUE
 40   CONTINUE
      DSUM=DSUM+DSUM2
      DSSUM=DSSUM+DSSUM2
C Count down numbers still to generate
C on each processor
      NGEN=NGEN-N
 20   CONTINUE
C Compute overall mean.
      DMEAN=DSUM/DFLOAT(NTOT)
      WRITE (*,*) ' Sample mean ',DMEAN
C Compute overall sample standard deviation.
      DSIG=DSSUM/DFLOAT(NTOT)
      WRITE (*,*)' Sample standard deviation ',DSIG
      STOP
      END
```

(4)　手法概要

本ルーチンは正規分布する擬似乱数を生成するために, 初等関数の高速な計算法を利用した Polar 法(Polar method)を使用しています. この方法に必要な一様擬似乱数は“SSL II 拡張機能使用手引書 II”DVRAU4 と同じ手法を利用して生成されます.

Polar 法は“付録 1 参考文献一覧表”の[46]に記述されています. 実現方法の詳細および他の方法との比較は“付録 1 参考文献一覧表”の[11]を参照してください.

DM_VRANN4

正規乱数の生成 (Wallace 法)
CALL DM_VRANN4(DAM, DSD, IX, DA, K, N, DWORK, NWORK, ICON)

(1)　機能

与えられた平均 m と標準偏差σを持った正規分布の密度関数(1.1)から, 擬似乱数を生成します.

$$f(x)=\frac{1}{\sigma\sqrt{2\pi}}\exp(\frac{-(x-m)^2}{2\sigma^2}) \tag{1.1}$$

(2)　パラメタ

DAM .....入力. 正規分布の平均 m.
倍精度実数型.

DSD .....入力. 正規分布の標準偏差σ(>0).
倍精度実数型.

IX .....入力. 出発値
初回呼び出しでは, IX に正の値を与え, 2 回目以降の呼び出しでは返却値 0 のまま呼び出して下さい. 初回呼び出しで指定する出発値が異なると, 異なる乱数列が生成されます.
((3)使用上の注意 b. 注意①参照).
出力. 0.

DA.....出力.各スレッドごとに生成される異なる擬似正規乱数.
DA(K, NUMT)なる倍精度実数型 2 次元配列.
ここで NUMT は本ルーチンで並列実行を行うスレッドの数. P 番目(0～NUMT - 1)のスレッドで生成された N 個の擬似乱数が DA(1:N, P+1)に返却されます.((3)使用上の注意 a. 注意⑥参照)

K.....入力. 配列 DA の 1 次元目の大きさ. (≥N)

N.....入力. DA に返却される各スレッドで生成される正規分布擬似乱数の個数.
((3)使用上の注意 b. 注意②参照).

DWORK .....作業領域. DWORK(NWORK, NUMT)なる倍精度実数型 2 次元配列.
本サブルーチンを繰り返し呼び出す場合は, 内容および本ルーチンで並列実行を行うスレッドの数 NUMT を変更しないでください. DWORK には, 本サブルーチンが現在の位置から再度呼び出される場合に必要な, すべての現情報が格納されています. (3)使用上の注意 a. 注意③, ⑥参照)

NWORK .....入力. 配列 DWORK の 1 次元目の大きさ. NWORK≥1350.

ICON.....出力. コンディションコード. 表 DM_VRANN4-1 参照.

表 DM_VRANN4-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし.	ー
30000	K<N または K<1	処理を打ち切る.
30001	NWORK が小さ過ぎた. IX<0, DSD≤0	
30002	内部エラー	
30003～30008	DWORK が変更されていた. 又は, 初回呼び出しで IX に 0 が与えられた.	
30009	IX が大き過ぎる.	
40001～40002	DWORK が変更されていた. 又は, 初回呼び出しで IX に 0 が与えられた.	

(3)　使用上の注意

a.　注意

①　出発値 IX について

確定的な計算(deterministic program)で擬似乱数列を生成する場合は, 何かランダムな入力が必要になります. したがって, 利用者は出発値 IX を与えなければなりません. この出発値は “たね(*seed*)” と呼ばれます. 本サブルーチンの初回呼び出しでは, “たね” IX は正整数である必要があります(例外事項については, ⑤を参照). 2 回目以降の呼び出しでは, IX には 0 を指定して下さい. これは同じ乱数列から続いて乱数を求めるためです. プログラミングを容易にするため, 本サブルーチンは初回呼び出し後 IX に 0 を返しています.

②　パラメタ N について

本サブルーチンは最初の *seed*(IX)で定義される無限列から, 次の N 個の擬似乱数を返却します. N ≤ 0 の時は擬似乱数を返却しません.

効率良くするためには, 利用者は N を十分大きく(例えば N=100, 000)します. それは, サブルーチン呼び出しのオーバーヘッドが減少するためです. 本サブルーチンを連続的に呼び出す途中で, N が変更される場合は, 配列 DA の 1 次元目の大きさ K を最大の N の大きさ以上に設定する必要があります.

③　作業領域 DWORK について

DWORK は, 複数回本サブルーチンを呼び出す場合, 次の呼び出しのための情報を格納する作業領域として使用されます. したがって, 本サブルーチンを呼び出している間は, 呼び出し側のプログラムで DWORK の内容を変更してはいけません.

④　パラメタ NWORK について

DWORK(1,:) , ... , DWORK(NWORK,:)は, 本サブルーチンによって使用されます. NWORK は, 本サブルーチンの各呼び出しで変えないでください. NWORK には少なくとも 1350 以上, また十分大きな数(例えば NWORK=500, 000)を与えると効率的です.

⑤　同じ乱数の再生成について
DWORK(1,:) ,... , DWORK(NWORK,:)が保存される場合, その DWORK を再使用し, IX=0 として本サブルーチンを呼び出すことにより, (DWORK が保存された時点からの)同じ乱数列が再度生成されます.

⑥　本サブルーチンを並列実行するスレッド数 NUMT は, 環境変数 OMP_NUM_THREADS または実行環境ルーチン OMP_SET_NUM_THREADS( )で指定します.
実行環境ルーチン OMP_SET_NUM_THREADS( )で指定するときは, 2 回目以降の呼び出しの直前でも初回呼び出しと同じスレッド数を OMP_SET_NUM_THREADS( )で指定して下さい.

⑦　本ルーチンで使われている Wallace 法は, DM_VRANN3 で使われている Polar 法に比べて約 3 倍高速です.

b.　使用例
10, 000, 000 × 4 個の擬似正規乱数を生成し, 平均と標準偏差を計算します.

```
C     ** EXAMPLE **
      PARAMETER (NUMT=4)
      PARAMETER (NRAN=10000000)
      PARAMETER (NSEED=12345)
      PARAMETER (NWMAX=100000)
      PARAMETER (NBUF=120000,K=NBUF)
      DOUBLE PRECISION DA(K,NUMT)
      DOUBLE PRECISION DWORK(NWMAX,NUMT)
      DOUBLE PRECISION DSUM,DSUM2,DSSUM,DSSUM2
      DOUBLE PRECISION DMEAN,DSIG
      DOUBLE PRECISION DAM,DSD
      INTEGER NTOT
C     Initialize ix,n and nwork
      IX=NSEED
      WRITE (*,*)' Seed ',IX
      DAM=0.0D0
      DSD=1.0D0
      WRITE (*,*)' Mean ',DAM
      WRITE (*,*)' Standard deviation ',DSD
      N=NBUF
      NWORK=NWMAX
      DSUM=0.0D0
      DSSUM=0.0D0
C     ngen counts down to 0
      NGEN=NRAN
      NTOT=NRAN*NUMT
```

```
C     Generate ngen numbers
C     with maximum NBUF at a time.
      KRPT=(NRAN+NBUF-1)/NBUF
      WRITE (*,*)' Generating ',NTOT,' numbers'
      WRITE (*,*)' with ',KRPT,
     $   ' calls to dm-vrann3 on ',NUMT,
     $   ' threads'
      CALL OMP_SET_NUM_THREADS(NUMT)
      DO 20 IZ=1,KRPT
      N=MIN0(NBUF,NGEN)
      CALL DM_VRANN4(DAM,DSD,IX,DA,K,N,DWORK,NWORK,ICON)
      IF (ICON.NE.0) WRITE (*,*)' ICON ',ICON
C     Accumulate sum of numbers
      DSUM2=0.0D0
      DO 30 J=1,NUMT
      DO 10 I=1,N
      DSUM2=DSUM2+DA(I,J)
  10  CONTINUE
  30  CONTINUE
C     Accumulate sum of numbers globally.
      DSSUM2=0.0D0
      DO 40 J=1,NUMT
      DO 50 I=1,N
       DSSUM2=DSSUM2+DA(I,J)*DA(I,J)
 50   CONTINUE
 40   CONTINUE
      DSUM=DSUM+DSUM2
      DSSUM=DSSUM+DSSUM2
C Count down numbers still to generate
C on each processor
      NGEN=NGEN-N
 20   CONTINUE
C Compute overall mean.
      DMEAN=DSUM/DFLOAT(NTOT)
      WRITE (*,*) ' Sample mean ',DMEAN
C Compute overall sample standard deviation.
      DSIG=DSSUM/DFLOAT(NTOT)
      WRITE (*,*)' Sample standard deviation ',DSIG
      STOP
      END
```

(4)　手法概要

本ルーチンは正規分布する擬似乱数を生成するために, Wallace の方法の一種の変形を使用しています. この方法に必要な一様擬似乱数は“SSL II 拡張機能使用手引書 II”DVRAU4 と同じ手法を利用して生成されます.

Wallace の方法は, “付録 1 参考文献一覧表”の[78]に記述されています. 実現方法の詳細および他の方法との比較は“付録 1 参考文献一覧表”の[11]および[12]を参照してください.

DM_VRANU4

一様乱数[0, 1)の生成
CALL DM_VRANU4(IX, DA, K, N, DWORK, NWORK, ICON)

(1) 機能

各スレッドで, 区間[0, 1)上での一様分布からおのおの異なる擬似乱数列を生成します.

(2) パラメタ

IX .....入力. 出発値.
出力. 0.
初回呼び出しでは, IX に正の値を与え, 2 回目以降の呼び出しでは返却値 0 のまま呼び出してください. IX < 8000000.
((3)使用上の注意 a. 注意①参照).

DA.....出力. 区間[0, 1)で各スレッドごとに生成される異なる一様な擬似乱数.
DA(K, NUMT)なる倍精度実数型 2 次元配列.
ここで NUMT は本ルーチンで並列実行を行うスレッドの数. P 番目(0～NUMT-1)のスレッドで生成された N 個の擬似乱数が DA(1:N, P+1)に返却されます.((3)使用上の注意 a. 注意⑥参照)

K.....入力. 配列 DA の 1 次元目の大きさ. (≥ N)

N.....入力. DA に返却される各スレッドで生成される一様分布擬似乱数の個数.
((3)使用上の注意 a. 注意②参照)

DWORK .....作業領域. DWORK(NWORK, NUMT)なる倍精度実数型 2 次元配列.
本サブルーチンを繰り返し呼び出す場合は, 内容および本ルーチンで並列実行を行うスレッドの数 NUMT を変更しないでください. DWORK には, 本サブルーチンが現在の位置から再度呼び出される場合に必要な, すべての現情報が格納されています. (3)使用上の注意 a. 注意③, ⑥参照)

NWORK .....入力. 配列 DWORK の大きさ. NWORK ≥ 388 であること.
作業域の大きさと乱数の周期などの関係については “(4)手法概要” を参照してください.

ICON.....出力. コンディションコード.
表 DM_VRANU4-1 参照.

表 DM_VRANU4-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし	ー
30000	K<N または K<1	処理を打ち切る.
30001	NWORK が小さ過ぎた.	
30002	内部チェックで誤りを検出した.	
30003～30008	DWORK が変更されていた. または, 初回呼び出しで IX に 0 が与えられた.	
30009	IX が大き過ぎた.	

(3)　使用上の注意

a.　注意

①　出発値 IX について

確定的な計算(deterministic program)で擬似乱数列を生成する場合は, 何かランダムな入力が必要になります. したがって, 利用者は出発値 IX を与えなければなりません. この出発値は “たね(*seed*)” と呼ばれます. 本サブルーチンの初回呼び出しでは, “たね” IX は正整数である必要があります(例外事項については, ⑤を参照). 2 回目以降の呼び出しでは, IX には 0 を指定して下さい. これは同じ乱数列から続いて乱数を求めるためです. プログラミングを容易にするため, 本サブルーチンは初回呼び出し後 IX に 0 を返しています. 本サブルーチンは, *seed*=*seed*×OMP_GET_NUM_THREADS( )+OMP_GET_THREAD_NUM( )+1 のように, スレッド番号+1, OMP_GET_THREAD_NUM( )+1 を *seed* に付加しています. このように異なるスレッドで使用される *seed* は, 他のプロセッサーでの *seed* とは全く別であることが保証され, 10^{18} 未満の長さの乱数列であればオーバーラップすることはありません. ((4)手法概要参照).

②　パラメタ N について

本サブルーチンは最初の *seed*(IX)で定義される無限列から, 次の N 個の擬似乱数を返却します. N ≤ 0 の時は擬似乱数を返却しません.

効率良くするためには, 利用者は N を十分大きく(例えば N=100, 000)します. それは, サブルーチン呼び出しのオーバーヘッドが減少するためです. 本サブルーチンを連続的に呼び出す途中で, N が変更される場合は, 配列 DA の 1 次元目の大きさ K を最大の N の大きさ以上に設定する必要があります.

③　作業領域 DWORK について

DWORK は, 複数回本サブルーチンを呼び出す場合, 次の呼び出しのための情報を格納する作業領域として使用されます. したがって, 本サブルーチンを呼び出している間は, 呼び出し側のプログラムで DWORK の内容を変更してはいけません.

④　パラメタ NWORK について

DWORK(1, :) ,..., DWORK(NWORK, :)は, 本サブルーチンによって使用されます.

NWORK は, 本サブルーチンの各呼び出しで常に一定である方が良く, NWORK は少なくとも 388 以上にして下さい.

⑤ チェックポイントについて
DWORK(1, :) ,..., DWORK(NWORK, :)が保存される場合, その DWORK を再使用し, IX=0 として本サブルーチンを呼び出すことにより, (DWORK が保存された時点からの)同じ乱数列が再度生成されます.

⑥ 本サブルーチンを並列実行するスレッド数 NUMT は, 環境変数 OMP_NUM_THREADS または実行環境ルーチン OMP_SET_NUM_THREADS( )で指定します.
実行環境ルーチン OMP_SET_NUM_THREADS( )で指定するときは, 2 回目以降の呼び出しの直前でも初回呼び出しと同じスレッド数を OMP_NUM_THREADS( )で指定して下さい.

b. 使用例
1,000,000 × 4 個の擬似一様乱数を生成し, 平均値を計算します. 出発値は 123 とします.

```
C     **EXAMPLE**
      PARAMETER(NUMT=4)
      PARAMETER(NRAN=1000000)
      PARAMETER(NSEED=123)
      PARAMETER(NWMAX=5000)
      PARAMETER(NBUF=25000)
      DOUBLE PRECISION DA(NBUF,NUMT)
      DOUBLE PRECISION DWORK(NWMAX,NUMT)
      DOUBLE PRECISION DSUM,DSUM2
      DOUBLE PRECISION DMEAN,DSIG
      INTEGER TNO,NTOT

C     Initialize ix, n and nwork
      IX=NSEED
      PRINT *,' Seed ',IX
      N=NBUF
      NWORK=NWMAX
      DSUM=0.0D0
C ngen counts down to 0
      NGEN=NRAN
      NTOT=NRAN*NUMT
C Generate ngen numbers on each thread
C with maximum NBUF at a time
      KRPT=(NRAN+NBUF-1)/NBUF
      PRINT *,' Generating ',NTOT,
```

```
     $        ' numbers'
      PRINT *,' with ',KRPT,
     $        ' calls to dm_vranu4 on ',NUMT,
     $        ' threads'
      DO 20 J=1,KRPT
      N=MIN0(NBUF,NGEN)
      DSUM2=0.0D0
      CALL OMP_SET_NUM_THREADS(NUMT)
      CALL DM_VRANU4(IX,DA,NBUF,N,DWORK,NWORK,ICON)
      IF(ICON.NE.0) PRINT *,
     $        ' Error return,',
     $        ' ICON ',ICON
      DO 30 TNO=1,NUMT
C Accumulate sum of numbers locally
      DO 10 I=1,N
      DSUM2=DSUM2+DA(I,TNO)
   10 CONTINUE
   30 CONTINUE
C Accumulate sum of numbers globally
      DSUM=DSUM+DSUM2
C Count down numbers still to generate
C on each processor
      NGEN=NGEN-N
   20 CONTINUE
C Compute overall mean
      DMEAN=DSUM/DFLOAT(NTOT)
      PRINT *,' Mean ',DMEAN
C Compute deviation from 0.5 normalized
C by expected value 1/sqrt(12*ntot).
c This should be (approximately) normally
C distributed with mean 0, variance 1.
      DSIG=(DMEAN-0.5D0)*DSQRT(12.0D0*NTOT)
      PRINT *,' Normalized deviation ',DSIG

      STOP
      END
```

(4)　手法概要

本サブルーチンは一般化されたフィボナッチ手法(Generalized Fibonacci method)を用いています. 擬似乱数列を X(1), X(2), ..., とする時,

$$X(J) = \alpha \times X(J - r) + \beta \times X(J - s) \ \text{(modulo 1)}$$

J > r > s

ここで, r と s は固定された正整数で, lags と呼ばれます. α と β は小さな奇数の整数です.
初回呼び出し(または, IX > 0 としての呼び出し)において, 本サブルーチンは 2 を法とした原始三項式(a primitive trinomial(mod 2))を定義する組(r, s)と, それに対応する線型再帰式(a linear recurrence)を決めます. ここでは, 14 個の可能な組(r, s)がありますが, N 及び NWORK が十分に大きいという制約に従い, r が最大な組が選択されます. したがって, 利用者は
- 適度に長い周期を持ち, 初期化のオーバーヘッドが小さく, 格納域が節約(例えば NWORK=1000)できる比較的良い生成法
- 極めて長い周期を持ち, 初期化のオーバーヘッドは大きく, 格納域も必要(例えば NWORK=133000)だが, 非常に良い生成法
- 組(r, s)の選択方法の正確な詳細はわからない, ある妥協した中間的な方法が選択できます. 本サブルーチンの使用する組(r, s)を表 DM_VRANU4-2 に示します.
原始三項式の表については, “付録 1 参考文献一覧表” の[41]を参照して下さい.

表 DM_VRANU4-2 組(r, s)

r	s	r	s
127	97	4423	2325
258	175	9689	5502
521	353	19937	10095
607	334	23209	13470
1279	861	44497	23463
2281	1252	110503	56784
3217	2641	132049	79500

本サブルーチンは, r ≤ 1000 の時, $(\alpha, \beta) = (7, 9)$を, r >1000 の時, $(\alpha, \beta) = (1,15)$をそれぞれパラメタとして選択します. その理論的根拠は, α >1 の時, 統計的検定(statistical test)において性能が向上するようですが, それは r がより小さい時のみに限定されることなのです. 大きな r に対しては, たとえ $\alpha = 1$ でも統計的検定の性能は良く, この選択であれば乱数生成の性能は向上します.
r を 127(最小の NWORK に対して)から 132049(N ≥ 264098, NWORK ≥ 132056)の範囲の数とすると, 乱数列の周期は, $W\times(2^r-1)$です. 因子 W は, 語長(word length)に依存して決まります. (Fujitsu VPP シリーズおよび PRIMEPOWER シリーズ(SPARC アーキテクチャ)では, $W=2^{48}$ であり, 周期は最低でも 10^{52} 以上です.)
本サブルーチンの初期化では, 異なる *seed*(IX)に対して返却される擬似乱数列が, 全周期列の中で少なくとも $2^{60} > 10^{18}$ の間隔をもって分かれていることを保証しています. したがってあらゆる実用目的に対して, 異なる *seed*(IX)は異なる擬似乱数列生成をします. 本サブルーチンでは *seed* にスレッド番号+1 を付加し, 異なるスレッドでは異なる *seed* が使用されます.
手法及び実現方法の詳細は, “付録 1 参考文献一覧表” の[9]及び[10]により詳しく記述され

ています. さらに詳しい説明や他の手法との比較については, “付録 1 参考文献一覧表” の[4], [24], [42]及び[53]を参照して下さい.

(5)　一様乱数の検定

表 DM_VRANU4-3 に, NWORK=44504(r=44497, s=23463)の時, DM_VRANU4 が生成した擬似乱数についての統計的仮説検定(testing of statistical hypotheses)の結果を示します.

この表で, χ^2検定の自由度 f は極めて大きく(100 万程度), この場合, 式 $\sqrt{2\chi^2}-\sqrt{2f-1}$ は, 単位分散を持つ正規分布(a normal deviate with unit variance)に極めて近くなります.

表 DM_VRANU4-3　χ^2検定結果(*n* 次元単位超立方体での一様分布)

(注 1) 次元	(注 2) サイズ	(注 3) res_l	(注 4) res_v	(注 5) 密度	(注 6) thrd1	(注 6) thrd2	(注 6) thrd3	(注 6) thrd4
1	10^9	5×10^7	50000000	20.00	1.21	1.37	-0.24	0.90
1	0.8×10^9	1.25×10^7	12500000	64. 00	-0.67	0.79	0.39	-1.04
2	10^9	7071	49999041	10.00	-0.10	0.42	0.30	-0.65
2	2×10^9	3535	12496225	80.02	-0.37	-0.25	1.44	-0.07
3	2×10^9	368	49836032	13. 38	1.40	-0.21	-1.92	-0.47
3	2×10^9	232	12487168	53. 39	-0.96	-0.63	0.46	-0.22
4	2×10^9	84	49787136	10.04	0.76	1.51	1.10	-1.45
4	2×10^9	59	12117361	41.26	-0.38	0.08	0.32	0.16

(注 1)次元：単位超立方体の次元

(注 2)サイズ：生成される擬似乱数の個数

(注 3)res_l：各次元での区間[0, 1)を等分割した部分区間の数

(注 4)res_v単位超立方体を等分割した超立方体の数

(注 5)密度：小超立方体あたりの乱数点(random point)の平均値

(注 6)thrd1～4：各スレッド, 変数値 $\sqrt{2\chi^2}-\sqrt{2f-1}$

DM_VRANU5

一様乱数[0, 1)の生成 (MRG8)
CALL DM_VRANU5(IX, DA, N, J, DWORK, ICON)

(1)　機能

区間[0, 1)上での一様分布から擬似乱数列を 8 次の原始多項式による多重再帰ジェネレータ(MRG8:Multiple Recursive Generator with 8th-order full primitive polynomials)で生成します. 本ルーチンでは使用するスレッド数に関係なく同じ疑似乱数列を生成します. 乱数の再現性が必要な場合には本ルーチンをご利用ください. このため, DM_VRANU4 とはインタフェースが異なります.

また, 本ルーチンでは疑似乱数列の J 個先までジャンプする機能をサポートしています. この機能は並列に乱数を生成する場合に利用できます.

従来ルーチンの DM_VRANU4 は本ルーチンよりも高速に疑似乱数を生成します. 乱数生成の性能を優先する場合は DM_VRANU4 をご利用ください.

本ルーチンと DM_VRANU4 は両方とも著名な乱数検定プログラム TESTU01 の bigCrush テストをパスします.

(2)　パラメタ

IX .....入力. 出発値.
　　出力. 0.
　　初回呼び出しでは, IX に正の値を与え, 2 回目以降の呼び出しでは返却値 0 のまま呼び出してください.
　　((3)使用上の注意 a. 注意①参照).

DA.....出力. 区間[0, 1)で生成される一様な擬似乱数.
　　DA(N)なる倍精度実数型 1 次元配列.

N.....入力. DA に返却される擬似乱数の個数.

J.....入力. 疑似乱数列の先頭を J 個先までジャンプします.
　　8 バイト整数型.
　　疑似乱数列の最初から必要な場合は 0_8 (8 バイト整数型の 0)を設定してください.
　　((3)使用上の注意 a. 注意②参照)

DWORK .....作業領域. DWORK(8)なる倍精度実数型 1 次元配列.
　　本サブルーチンを繰り返し呼び出す場合は, 内容を変更しないでください.
　　DWORK には, 本サブルーチンが現在の位置から再度呼び出される場合に必要な, すべての現情報が格納されています. ((3)使用上の注意 a. 注意③参照).

ICON.....出力. コンディションコード.
　　表 DM_VRANU5-1 参照.

表 DM_VRANU5-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし	ー
30000	IX<0, N<1 または J<0	処理を打ち切る.

(3)　使用上の注意

a.　注意

①　出発値 IX について

確定的な計算(deterministic program)で擬似乱数列を生成する場合は, 何かランダムな入力が必要になります. したがって, 利用者は出発値 IX を与えなければなりません. この出発値は “たね(seed)” と呼ばれます. 本サブルーチンの初回呼び出しでは,“たね”IX は正整数である必要があります(例外事項については, ④を参照).

2 回目以降の呼び出しでは, IX には 0 を指定して下さい. これは同じ乱数列から続いて乱数を求めるためです. プログラミングを容易にするため, 本サブルーチンは初回呼び出し後 IX に 0 を返しています.

②　パラメタ J について

本サブルーチンでは, パラメタ J に 0 以上の値を入力することで, 乱数列を J 個先までジャンプすることができます.

プロセス並列で計算するとき, プロセスごとに同じ IX を与え, J を変えることにより, プロセスごとに異なる疑似乱数を生成することができます. ((3)使用上の注意 b) 使用例 使用例 2, 3 参照).

③　作業領域 DWORK について

DWORK は, 複数回本サブルーチンを呼び出す場合, 次の呼び出しのための情報を格納する作業領域として使用されます. したがって, 本サブルーチンを呼び出している間は, 呼び出し側のプログラムで DWORK の内容を変更してはいけません.

④　チェックポイントについて

DWORK が保存される場合, その DWORK を再使用し, IX=0 として本サブルーチンを呼び出すことにより, (DWORK が保存された時点からの)同じ乱数列が生成されます.

b.　使用例

使用例 1

1,000,000 個の擬似一様乱数を生成し, 平均値を計算します. 出発値は 123 とします. (並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 つのスレッドで並列実行するときは,OMP_NUM_THREADS を 4 に設定して実行します.)

```
C      **EXAMPLE 1**
       INTEGER NRAN,NSEED,NBUF
```

```
      PARAMETER(NRAN=10000000)
      PARAMETER(NSEED=123)
      PARAMETER(NBUF=25000)
      DOUBLE PRECISION DA(NBUF)
      DOUBLE PRECISION DWORK(8)
      DOUBLE PRECISION DSUM,DSUM2
      DOUBLE PRECISION DMEAN
      INTEGER IX,N,ICON
      INTEGER I,J
C
C Generate NRAN numbers with maximum NBUF at a time
      IX=NSEED
      PRINT *,' Seed ',IX
      PRINT *,' Generating ',NRAN,' numbers'
C
      DSUM=0.0D0
      DO J=1,NRAN,NBUF
        N=MIN0(NBUF,NRAN-J+1)
        CALL DM_VRANU5(IX,DA,N,0_8,DWORK,ICON)
        IF(ICON.NE.0) THEN
          PRINT *,' Error return ICON ',ICON
        END IF
        DSUM2=0.0D0
        DO I=1,N
          DSUM2=DSUM2+DA(I)
        END DO
        DSUM=DSUM+DSUM2
      END DO
C Compute mean
      DMEAN=DSUM/DFLOAT(NRAN)
      PRINT *,' Mean ',DMEAN

      STOP
      END
```

使用例 2

MPI で並列化されたプログラムで, 4 つのプロセスでそれぞれ異なる 100,000 個の擬似一様乱数を生成し, 全体の平均値を計算します. 出発値は 123 とします.

このプログラムでは, J に 2^{31}-1 を与えています. 生成する疑似乱数が 1 プロセスあたり 2^{31}-1 個以下なら重複することはありません.

```
C     **EXAMPLE 2**
      INTEGER,PARAMETER::N=10000
      INTEGER(8),PARAMETER::JUMP=2147483647_8   ! =2**31-1
      REAL(8)::X(N)
      REAL(8)::DNALL
      INTEGER::IRANK,NP,IERROR
      INTEGER::IX,ICON
      INTEGER::I
      INTEGER(8)::J
      REAL(8)::WORK(8)
      REAL(8)::DSUM,DSUMALL,DMEAN
      INCLUDE 'mpif.h'
C
      CALL MPI_INIT(IERROR)
      CALL MPI_COMM_RANK( MPI_COMM_WORLD, IRANK, IERROR )
      CALL MPI_COMM_SIZE( MPI_COMM_WORLD, NP, IERROR )
C
      IX=123
      J=IRANK*JUMP
      CALL DM_VRANU5(IX,X,N,J,WORK,ICON)
      IF(ICON.NE.0) THEN
        WRITE(6,*) 'DM_VRANU5 ERROR ICON= ',ICON
      END IF
C
      DSUM=0.0D0
      DO I=1,N
        DSUM=DSUM+X(I)
      END DO
      CALL MPI_REDUCE(DSUM,DSUMALL,1,MPI_REAL8,MPI_SUM,0,
     -                MPI_COMM_WORLD,IERROR)
C Compute overall mean
      DNALL=DFLOAT(N)*DFLOAT(NP)
      IF(IRANK.EQ.0) THEN
        DMEAN=DSUMALL/DNALL
        WRITE(6,*) 'Mean    ',DMEAN
      END IF
C
      CALL MPI_FINALIZE(IERROR)
      END
```

使用例 3

MPI で並列化されたプログラムで, 4 つのプロセスで合わせて 1,000,000 個の擬似一様乱数を 2 つ生成し, 全体の平均値を計算します. 出発値は 123 とします.

このプログラムでは 1,000,000 個の疑似乱数を全プロセスで等分割して生成します. プロセス数を変数 NP で指定しています. プロセス数を変更して実行する場合, NP の値を変更してください. このとき, 全プロセスで生成した 1,000,000 個の疑似乱数はプロセス数に関係なく同じ値になります.

```
C     **EXAMPLE 3**
      INTEGER::NX,NY,NP
      PARAMETER(NX=100000)
      PARAMETER(NY=100000)
      PARAMETER(NP=4)        ! NUMBER OF PROCESS
      REAL(8)::X((NX+NP-1)/NP),Y((NY+NP-1)/NP)
      INTEGER::IRANK,NSIZE,IERROR
      INTEGER::IX,NL,ICON,JUMP
      INTEGER::I
      INTEGER(8)::J0,J
      REAL(8)::WORK(8)
      REAL(8)::DSUM,DSUMALL,DMEAN
      INCLUDE 'mpif.h'
C
      CALL MPI_INIT(IERROR)
      CALL MPI_COMM_RANK( MPI_COMM_WORLD, IRANK, IERROR )
      CALL MPI_COMM_SIZE( MPI_COMM_WORLD, NSIZE, IERROR )
      IF(NP.NE.NSIZE) THEN
        CALL MPI_FINALIZE(IERROR)
        STOP
      END IF
C
      IX=123
      JUMP=(NX+NP-1)/NP
      J=MIN(IRANK*JUMP,NX)
      NL=MIN(JUMP,NX-J)
      IF(NL.GE.1) THEN
        CALL DM_VRANU5(IX,X,NL,J,WORK,ICON)
        IF(ICON.NE.0) THEN
          WRITE(6,*) 'DM_VRANU5 ERROR ICON= ',ICON
        END IF
        J0=NX-(J+NL)
      ELSE
        J0=NX
```

```
      END IF
C
      DSUM=0.0D0
      DO I=1,NL
        DSUM=DSUM+X(I)
      END DO
      CALL MPI_REDUCE(DSUM,DSUMALL,1,MPI_REAL8,MPI_SUM,0,
     *                MPI_COMM_WORLD,IERROR)
C Compute overall mean of X
      IF(IRANK.EQ.0) THEN
        DMEAN=DSUMALL/DFLOAT(NX)
        WRITE(6,*) 'Mean of X   ',DMEAN
      END IF
C
      JUMP=(NY+NP-1)/NP
      J=MIN(IRANK*JUMP,NY)
      NL=MIN(JUMP,NY-J)
      J=J+J0
      IF(NL.GE.1) THEN
        CALL DM_VRANU5(IX,Y,NL,J,WORK,ICON)
        IF(ICON.NE.0) THEN
          WRITE(6,*) 'DM_VRANU5 ERROR ICON= ',ICON
        END IF
      END IF
C
      DSUM=0.0D0
      DO I=1,NL
        DSUM=DSUM+Y(I)
      END DO
      CALL MPI_REDUCE(DSUM,DSUMALL,1,MPI_REAL8,MPI_SUM,0,
     *                MPI_COMM_WORLD,IERROR)
C Compute overall mean of Y
      IF(IRANK.EQ.0) THEN
        DMEAN=DSUMALL/DFLOAT(NY)
        WRITE(6,*) 'Mean of Y   ',DMEAN
      END IF
C
      CALL MPI_FINALIZE(IERROR)
      END
```

(4)　手法概要

本サブルーチンは 8 次の原始多項式による多重再帰ジェネレータ(MRG8:Multiple Recursive Generator with 8th-order full primitive polynomials)を用いています. 擬似乱数列を $x_1, x_2, ...,$ とする時, 次の式で疑似乱数を生成します.

$x_i = (a_1x_{i-1} + a_2x_{i-2} + a_3x_{i-3} + a_4x_{i-4} + a_5x_{i-5} + a_6x_{i-6} + a_7x_{i-7} + a_8x_{i-8}) \bmod p$

$p=2^{31}-1,$

$a_1= 1089656042,\ a_2= 1906537547,\ a_3= 1764115693,\ a_4= 1304127872,$

$a_5= 189748160,\ a_6= 1984088114,\ a_7= 626062218,\ a_8= 1927846343.$

$DA(i)= x_i*(1/p)$

乱数列の周期は $(2^{32}-1)^8-1$ (約 4.5×10^{74})です.

手法の詳細については“付録 1 参考文献一覧表”の[82]を参照してください.

MRG8 は, “付録 1 参考文献一覧表”の[83]でモンテカルロ法において良い結果が得られたことが報告されています.

(5)　一様乱数の検定

本ルーチンは疑似乱数の検証プログラムとして著名な TESTU01 の bigCrush テストをパスします.

TESTU01 の詳細については“付録 1 参考文献一覧表”の[84]を参照してください.

DM_VSCHOL

正値対称スパース行列の LDL^T 分解 (Left-looking な方法)
CALL DM_VSCHOL(A, NZ, NROW, NFCNZ, N, IORDERING, NPERM, ISW, EPSZ, NASSIGN, NSUPNUM, NFCNZFACTOR, PANELFACTOR, NSIZEFACTOR, NFCNZINDEX, NPANELINDEX, NSIZEINDEX, NDIM, NPOSTO, W, IW1, IW2, IW3, ICON)

(1)　機能

$n \times n$ の正値対称スパース行列 $\boldsymbol{A}$ を, 変形コレスキー分解法により LDL^T 分解します.

$$\boldsymbol{QPAP^TQ^T=LDL^T}$$

ただし, $\boldsymbol{P}$ は ordering による行列要素の並び換えを示す置換行列, $\boldsymbol{Q}$ は post order による行列要素の並び換えを示す置換行列を示します. $\boldsymbol{P}$, $\boldsymbol{Q}$ は直交行列です.

$\boldsymbol{L}$ は単位下三角行列, $\boldsymbol{D}$ は対角行列です.

(2)　パラメタ

A..... 入力. 係数行列 $\boldsymbol{A}$ のスパースな正値対称行列の下三角行列部分 $\{a_{ij}|i \geq j\}$ を圧縮列格納法で A(1:NZ)に格納します.
A(NZ)なる 1 次元配列.
圧縮列格納法については,実スパース行列と実ベクトル(圧縮列格納法), DM_VMVSCC の図 DM_VMVSCC-1 を参照してください.

NZ ..... 入力. 係数行列 $\boldsymbol{A}$ のスパースな正値対称行列の下三角行列部分にある非零要素の総数.

NROW..... 入力. 圧縮列格納法で使用される行指標で A に格納される要素が何番目の行ベクトルに属するかを示します.
NROW(NZ)なる 1 次元配列.

NFCNZ..... 入力. 行列 $\boldsymbol{A}$ の各列の非零要素を列方向に圧縮して順次配列 A に格納するとき,対応する列の最初の非零要素が格納される位置を表します.
NFCNZ(N+1)=NZ+1.
NFCNZ(N+1)なる 1 次元配列.

N..... 入力. 行列 $\boldsymbol{A}$ の次数 n.

IORDERING..... 入力. ordering を表す置換行列 $\boldsymbol{P}$ で $\boldsymbol{PAP^T}$ と変換した行列を LDL^T 分解するかを指定します. 置換行列は直交行列です.
1 のとき, $\boldsymbol{PAP^T}$ と変換したものを LDL^T 分解します.
1 以外のとき, 行列 $\boldsymbol{A}$ をそのまま LDL^T 分解します.

NPERM ..... 入力. IORDERING=1 のとき使用する置換行列をベクトルで指定します.
NPERM(N)なる 1 次元配列.
((3)使用上の注意 a.注意①参照).

ISW ..... 入力. 呼び出しに関する制御情報を示します.
1)　1 のとき, 初回呼び出し.

2) 2 のとき, 1 回目の呼び出しでは PANELFACTOR または NPANELINDEX の大きさが不足していた. ICON=31000 で終了した. このとき, NSIZEFACTOR,または NSIZEINDEX に返却された必要な大きさで PANELFACTOR または NPANELINDEX を確保し直して再度呼び出しします.
さらに A, NZ, NROW, NFCNZ, N, IORDERING, NPERM, NASSIGN, NSUPNUM, NFCNZFACTOR, NFCNZINDEX, NPANELINDEX, NPOSTO, NDIM, W, IW1, IW2, IW3 に格納されている値を変更してはなりません.

3) 3 のとき, 同じ非零パターンを持つ次数の同じ行列で,行列要素の値が異なる行列に対して symbolic decomposition の解析結果や必要な大きさが同じになる配列 PANELFACTOR, NPANELINDEX を再利用して LDL^T 分解することを指定します.行列の値を配列要素に格納し直して呼び出します.
このとき, NROW の値を変えずに配列 A に行列要素の値を格納し直すか, 別の配列 B に格納し引数 A として受け渡さなければなりません.
さらに NZ, NROW, NFCNZ, N, IORDERING, NPERM, NASSIGN, NSUPNUM, NFCNZFACTOR, NSIZEFACTOR, NFCNZINDEX, NPANELINDEX, NSIZEINDEX, NPOSTO, NDIM, W, IW1, IW2, IW3 に格納されている値を変更してはなりません.

EPSZ ..... 入力. ピボットの相対判定値 (≥0.0)
0.0 のときは標準値が設定されます.
((3)使用上の注意 a.注意②参照).

NASSIGN..... 出力. 各 supernode は複数の列ベクトルからなり,分解結果に関して共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します.この panel を PANELFACTOR の 1 次元部分配列として順番に割り付けたとき何番目になるかを示します. j=NASSIGN(i)のとき, i 番目の supernode を j 番目に割り付けたことを示します.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
結果の格納方法については図 DM_VSCHOL-1 を参照してください.
NASSIGN(N)なる 1 次元配列.
((3)使用上の注意 a.注意③参照).

NSUPNUM ..... 出力. supernode の総数.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します. (≤n)

NFCNZFACTOR.. 出力. 各 supernode は複数の列ベクトルからなり,分解結果に関して共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します.この panel を PANELFACTOR の 1 次元部分配列として順番に割り付けたとき, i 番目の panel の先頭要素 panel(1,1)が 1 次元配列の PANELFACTOR の何番目の要素になるかを示します.
NFCNZFACTOR(N+1)なる 8 バイト整数型の 1 次元配列.
結果の格納方法については図 DM_VSCHOL-1 を参照してください.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.

PANELFACTOR .. 出力. 各 supernode は複数の列ベクトルからなり,分解結果に関して共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します. panel を順番に割り付けます.
i 番目の supernode に対応する panel が何番目の位置に割り当てられるかは j=NASSIGN(i)から分かります.その先頭位置は NFCNZFACTOR(j)に格納されています. panel ごとに分解結果が格納されます.
i 番目に割付けられた panel の大きさは DIM(1,i)×DIM(2,i)の 2 次元配列と見なせます. i 番目の panel の panel(s,t), s>t, s=1,…,DIM(1,i), t=1,…,DIM(2,i)に単位下三角行列 $\boldsymbol{L}$ の対角要素を除いた対応部分が格納されます.対角部分 panel(t,t)には対角行列 $\boldsymbol{D}$ 対応部分が格納されます.
PANELFACTOR(NSIZEFACTOR)なる 1 次元配列.
結果の格納方法については図 DM_VSCHOL-1 を参照してください.
((3)使用上の注意 a.注意④参照).

NSIZEFACTOR.... 入力. PANELFACTOR の大きさを示す. 8 バイト整数型.
出力. PANELFACTOR の大きさとして必要な大きさが返却されます.
((3)使用上の注意 a.注意④参照).

NFCNZINDEX ..... 出力. 各 supernode は複数の列ベクトルからなり,分解結果に関して共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します.この行指標ベクトルを NPANELINDEX に順番に割り付けたとき i 番目の行指標ベクトルの先頭要素が 1 次元配列である NPANELINDEX の何番目の要素になるかを示します.
NFCNZINDEX(N+1)なる 8 バイト整数型の 1 次元配列.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
結果の格納方法については図 DM_VSCHOL-1 を参照してください.

NPANELINDEX... 出力. 各 supernode は複数の列ベクトルからなり,分解結果に関して共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します.この行指標ベクトルを NPANELINDEX に順番に割り付けます. i 番目の supernode に対応する panel の行の指標ベクトルが何番目に割り当てられるかは j=NASSIGN(i)から分かります.その先頭位置は NFCNZINDEX(j)に格納されています. panel ごとの行の指標ベクトルが格納されます.
この行指標は行列 $\boldsymbol{A}$ を post order で並び換えた行列 $\boldsymbol{QAQ}^{\boldsymbol{T}}$ での行の番号です.
NPANELINDEX(NSIZEINDEX)なる 1 次元配列.
結果の格納方法については図 DM_VSCHOL-1 を参照してください.
((3)使用上の注意 a.注意④参照).

NSIZEINDEX ..... 入力. NPANELINDEX の大きさを示す. 8 バイト整数型.
出力. 必要な大きさが返却されます.
((3)使用上の注意 a.注意④参照).

NDIM ..... 出力. NDIM(1,i)および NDIM(2,i)は i 番目に割り付けられた panel の 1 次元目と 2 次元目の大きさを示します.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.

NDIM(2,N)なる 2 次元配列.
結果の格納方法については図 DM_VSCHOL-1 を参照してください.

NPOSTO ..... 出力. post order で i 番目の node が行列 ***A*** の何番目の列番号に対応するかを表す 1 次元ベクトル.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
NPOSTO(N)なる 1 次元配列.
((3)使用上の注意 a.注意⑤参照).

W ..... 作業域.
出力/入力.
IORDERING=1 のとき大きさ NZ の 1 次元配列.
ISW=1, 2, 3 で続けて呼び出すとき,呼び出しの間で必要なデータを受け渡すために使われます.呼び出しの間で値を変更してはいけません.
IORDERING≠1 のとき,大きさ 1 の 1 次元配列.

IW1 ..... 作業域.
出力/入力.
IORDERING= 1 のとき大きさ NZ+N+1 の 1 次元配列.
ISW=1, 2, 3 で続けて呼び出すとき,呼び出しの間で必要なデータを受け渡すために使われます.呼び出しの間で値を変更してはいけません.
IORDERING≠1 のとき,大きさ 1 の 1 次元配列.

IW2 ..... 作業域.
出力/入力. 大きさ NZ+N+1 の 1 次元配列.
ISW=1, 2, 3 で続けて呼び出すとき,呼び出しの間で必要なデータを受け渡すために使われます.呼び出しの間で値を変更してはいけません.

IW3 ..... 作業域.
出力/入力. 大きさ N×35+35 の 1 次元配列.
ISW=1, 2, 3 で続けて呼び出すとき,呼び出しの間で必要なデータを受け渡すために使われます.呼び出しの間で値を変更してはいけません.
IW3(N*35+35)なる 1 次元配列.

ICON ..... 出力.コンディションコード.
表 DM_VSCHOL-1 参照

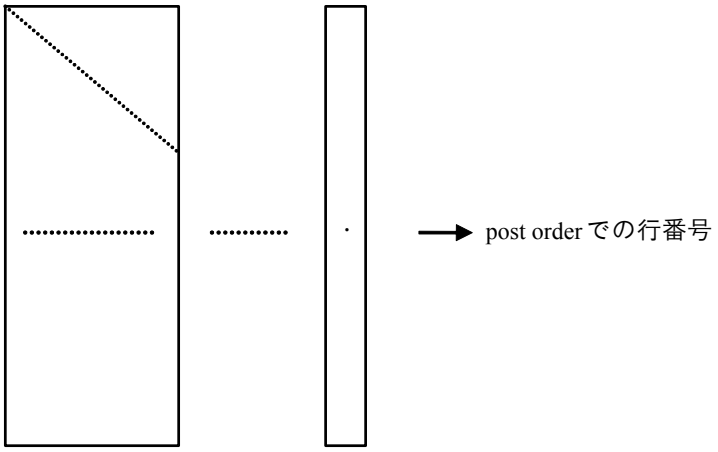


図 DM-VSCHOL-1 分解結果の格納概念図

j = NASSIGN(i)　　→ i 番目の supernode は j 番目に格納されます.
p = NFCNZFACTOR(j)　→ j 番目の panel は PANELFACTOR の p 番目の要素から
DIM(1,j)×DIM(2,j)の長さを占めます.
q = NFCNZINDEX(j)　→ j 番目の panel の行指標を表すベクトルは NPANELINDEX の
q 番目の要素から DIM(1,j)の長さを占めます.

panel は大きさ DIM(1,j)×DIM(2,j)の配列と見なせます.
panel(s,t) , s > t , s=1 , … , DIM(1,j),
t=1 , … , DIM(2,j)
に分解結果の単位下三角行列 $\boldsymbol{L}$ の対角要素を除いた部分が格納されます.
panel(t,t)に対角行列 $\boldsymbol{D}$ の対応部分が格納されます.
行指標の値は行列 $\boldsymbol{A}$ の列番号をもつ node を post order で並び換えた $\boldsymbol{QAQ^T}$ での列番号を表します.

表 DM_VSCHOL-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし	ー
10000	行列が正値でなかった.	処理は続行する.
20000	ピボットが相対的にゼロになった. 行列は非正則の可能性が強い.	処理を打ち切る.
30000	N<1,NZ<0,NFCNZ(N+1)≠NZ+1, NSIZEFACTOR<1, NSIZEINDEX<1, EPSZ<0.0,ISW<1,ISW>3	
30100	NPERM に指定した置換行列が正しくない.	
30200	NROW(j)に格納された i 列目の行指標 k が k<i または k>N	
30300	i 列目の行指標の数 NFCNZ(i+1)-NFCNZ(i)>n-i+1	
30400	対角要素が格納されていない列がある.	
31000	PANELFACTOR の大きさ NSIZEFACTOR または NPANELINDEX の大きさ NSIZEINDEX が小さすぎる.	NSIZEFACTOR または NSIZEINDEX で指定された大きさで PANELFACTOR または NPANELINDEX を割り付けて ISW=2 として再度呼び出す.

(3)　使用上の注意

a.　注意

①　直交行列である置換行列$\boldsymbol{P}$の要素$\boldsymbol{p_{ij}=1}$のとき, NPERM(i)=jと表現します.
逆は以下のようにして求めることができます.

```
DO i=1,n
 j=NPERM(i)
 NPERMINV(j)=i
 ENDDO
```

Fill-Reducing OrderingはMETISなどを使って求めることができます.
詳細は“付録1 参考文献一覧表”の[43],[44]を参照願います.

②　ピボットの相対零判定値にある値を設定したとすると, この値は次の意味を持っています. すなわち, 変形コレスキー分解法によるLDL^T分解の過程で, ピボットの絶対値がその値より小さくなった場合に, そのピボットの値を相対的に零と見なし, ICON=20000として処理を打ち切ります. EPSZの標準値は, 丸め誤差の単位をuとしたとき, EPSZ=16uです.
なお, ピボットの絶対値が小さくなっても, 処理を続行させたい場合には, EPSZに極小の値を与えれば良いのですが, 結果の精度は保証されません.
分解の途中でピボットが負となった場合, 係数行列は正値ではありません. このとき, ICON=10000として処理は続行します. ただし, ピボッティングを行っていないため, 計算誤差は大きい可能性があります.

③　分解結果に関するデータをNASSIGN, NSUPNUM, NFCNZFACTOR, NSIZEFACTOR, NFCNZINDEX, NPANELINDEX, NSIZEINDEX, NPOSTO, NDIM, IW3などをルーチンDM_VSCHOLXに受け渡して引き続き呼び出すことで,連立1次方程式$\boldsymbol{Ax=b}$を変形した$\boldsymbol{LDL^TPQx=PQb}$を解くことができます.

④　分解結果を格納する配列PANELFACTOR, NPANELINDEXの必要な大きさは,事前には分かりません.十分大きな配列を割り当てるか,本ルーチンを呼び出してsymbolic decompositionを行った解析の結果を使って割り当てを行うことができます.
例えば,大きさ1の1次元配列などを割付けます.そしてその大きさ1などの小さな値をNSIZEFACTOR, NSIZEINDEXに指定して, ISW=1で呼び出します.
symbolic decompositionを行い, ICON=31000で終了し, NSIZEFACTORとNSIZEINDEXに必要な大きさが返却されます.必要な大きさの配列を割付け直して, ISW=2で呼び出すことで, symbolic decomposition以降の処理を続けることができます.

⑤　列番号に対応するノードを考えます.これをpost orderで順番を並び換えたときの番号の対応関係がNPOSTOに格納されています. post orderのi番目のノードが並び換える前の何番目のノードに対応するかを表します. j=nposto(i)はj番目であることを表します.
①同様にこれは直交行列である置換行列$\boldsymbol{Q}$を表し,行列$\boldsymbol{A}$を$\boldsymbol{QAQ^T}$と並び換えることに相当します.
逆変換$\boldsymbol{Q^T}$は以下のようにして求めることができます.

```
DO i=1,n
j=NPOSTO(i)
NPOSTOINV(j)=i
ENDDO
```

b.　使用例

連立 1 次方程式 $\boldsymbol{Ax}=\boldsymbol{f}$ を解きます. 行列 $\boldsymbol{A}$ は境界条件として立方体の境界で 0 となる楕円型の偏微分方程式に有限差分法を適用して生成されます.

$$-\Delta u + a\nabla u + cu = f$$

ここで $a=(a_1, a_2, a_3)$, a_1, a_2, a_3 および c はゼロで,ラプラシアンになり行列 $\boldsymbol{A}$ は正値対称です.行列 $\boldsymbol{A}$ はサブルーチン init_mat_diag によって生成されます. これを圧縮列格納法に変換します.
(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT  REAL*8 (A-H,O-Z)
      PARAMETER (NORD=39,NX = NORD,NY =NORD ,NZ = NORD,
     $      N = NX*NY*NZ)
      PARAMETER (K = N+1)
```

```
 PARAMETER (NDIAG = 7,NDIAGH=4)

 DIMENSION NOFST(NDIAG)
 DIMENSION DIAG(K,NDIAG),DIAG2(K,NDIAG)
 DIMENSION C(K*NDIAG),NROWC(K*NDIAG),NFCNZC(N+1),
$          WC(K*NDIAG),IWC(2,K*NDIAG)
 DIMENSION A(NDIAGH*N),NROW(K*NDIAG),NFCNZ(N+1),
$          NPERM(N),NASSIGN(N),W(NDIAGH*N),
$          NPOSTO(N),NDIM(2,N),
$          IW1(NDIAGH*N+N+1),
$          IW2(NDIAGH*N+N+1),
$          IW3(35*N+35)
 REAL*8, DIMENSION(:), ALLOCATABLE ::  PANELFACTOR
 INTEGER*4, DIMENSION(:), ALLOCATABLE :: NPANELINDEX
 REAL*8 DUMMYF
 INTEGER*4 NDUMMYI
 INTEGER*8 NSIZEFACTOR,NSIZEINDEX,
$          NFCNZFACTOR(N+1),
$          NFCNZINDEX(N+1)
 DIMENSION X(N),B(N),SOLEX(N)


 PRINT *,'    LEFT-LOOKING MODIFIED CHOLESKY METHOD'
 PRINT *,'    FOR SPARSE POSITIVE DEFINITE MATRICES'
 PRINT *,'    IN COMPRESSED COLUMN STORAGE'
 PRINT *

 SOLEX(1:N)=1.0D0
 PRINT *,'    EXPECTED SOLUTIONS'
 PRINT *,'    X(1) = ',SOLEX(1),' X(N) = ',SOLEX(N)
 PRINT *

 VA1 = 0.0D0
 VA2 = 0.0D0
 VA3 = 0.0D0
 VC =  0.0D0
 XL = 1.0
 YL = 1.0
 ZL = 1.0
 CALL INIT_MAT_DIAG(VA1,VA2,VA3,VC,DIAG,NOFST
&          ,NX,NY,NZ,XL,YL,ZL,NDIAG,N,K)
```

```
      DO I=1,NDIAG
C
      IF(NOFST(I).LT.0)THEN
      NBASE=-NOFST(I)
      LENGTH=N-NBASE
      DIAG2(1:LENGTH,I)=DIAG(NBASE+1:N,I)
      ELSE
      NBASE=NOFST(I)
      LENGTH=N-NBASE
      DIAG2(NBASE+1:N,I)=DIAG(1:LENGTH,I)
      ENDIF
C
      ENDDO
C
      NUMNZC=1
      NUMNZ=1
      DO J=1,N
      NTOPCFGC=1
      NTOPCFG=1
      DO I=NDIAG,1,-1
C
      IF(DIAG2(J,I).NE.0.0D0)THEN
C
      NCOL=J-NOFST(I)
      C(NUMNZC)=DIAG2(J,I)
      NROWC(NUMNZC)=NCOL
C
      IF(NCOL.GE.J)THEN
      A(NUMNZ)=DIAG2(J,I)
      NROW(NUMNZ)=NCOL
      ENDIF

C
      IF(NTOPCFGC.EQ.1)THEN
      NFCNZC(J)=NUMNZC
      NTOPCFGC=0
      ENDIF
C
      IF(NTOPCFG.EQ.1)THEN
      NFCNZ(J)=NUMNZ
      NTOPCFG=0
      ENDIF
```

```
C
      IF(NCOL.GE.J)THEN
      NUMNZ=NUMNZ+1
      ENDIF
C
      NUMNZC=NUMNZC+1
      ENDIF
C
      ENDDO
      ENDDO
      NFCNZC(N+1)=NUMNZC
      NNZC=NUMNZC-1
      NFCNZ(N+1)=NUMNZ
      NNZ=NUMNZ-1
C

      CALL DM_VMVSCC(C,NNZC,NROWC,NFCNZC,N,SOLEX,
     $             B,WC,IWC,ICON)
C
      X=B
      IORDERING=0
      ISW=1
      EPSZ=0.0D0
      NSIZEFACTOR=1
      NSIZEINDEX=1

      CALL DM_VSCHOL(A,NNZ,NROW,NFCNZ,N,IORDERING,
     $              NPERM,ISW,EPSZ,NASSIGN,NSUPNUM,
     $              NFCNZFACTOR,DUMMYF,
     $              NSIZEFACTOR,NFCNZINDEX,
     $              NDUMMYI,NSIZEINDEX,NDIM,NPOSTO,
     $              W,IW1,IW2,IW3,ICON)

      PRINT *
      PRINT *,'    ICON = ',ICON,' NSIZEFACTOR = ',NSIZEFACTOR,
     $       'NSIZEINDEX = ',NSIZEINDEX
      PRINT *
C
C     ALLOCATE STORAGES IN RETURNED SIZES
C
      ALLOCATE( PANELFACTOR(NSIZEFACTOR) )
      ALLOCATE( NPANELINDEX(NSIZEINDEX) )
```

```
      ISW=2

      CALL DM_VSCHOL(A,NNZ,NROW,NFCNZ,N,IORDERING,
     $              NPERM,ISW,EPSZ,NASSIGN,NSUPNUM,
     $              NFCNZFACTOR,PANELFACTOR,
     $              NSIZEFACTOR,NFCNZINDEX,
     $              NPANELINDEX,NSIZEINDEX,NDIM,NPOSTO,
     $              W,IW1,IW2,IW3,ICON)

      CALL DM_VSCHOLX(N,IORDERING,
     $              NPERM,X,NASSIGN,NSUPNUM,
     $              NFCNZFACTOR,PANELFACTOR,
     $              NSIZEFACTOR,NFCNZINDEX,
     $              NPANELINDEX,NSIZEINDEX,NDIM,NPOSTO,
     $              IW3,ICON)

      ERR = ERRNRM(SOLEX,X,N)

      PRINT *,'    COMPUTED VALUES'
      PRINT *,'    X(1) = ',X(1),' X(N) = ',X(N)
      PRINT *
      PRINT *,'    ICON = ',ICON
      PRINT *
      PRINT *,'    N = ',N,' :: NX = ',NX,' NY = ',NY,' NZ = ',NZ
      PRINT *
      PRINT *,'    ERROR = ',ERR
      PRINT *
      PRINT *

      IF(ERR.LT.1.0D-8.AND.ICON.EQ.0)THEN
         WRITE(*,*)'    ********** OK **********'
      ELSE
         WRITE(*,*)'    ********** NG **********'
      ENDIF

      DEALLOCATE( PANELFACTOR,NPANELINDEX )

      STOP
      END

C =======================================
```

```
C     INITIALIZE COEFFICIENT MATRIX
C =====================================
      SUBROUTINE INIT_MAT_DIAG(VA1,VA2,VA3,VC,D_L,OFFSET
     &                        ,NX,NY,NZ,XL,YL,ZL,NDIAG,LEN,NDIVP)
      IMPLICIT  REAL*8(A-H,O-Z)
      DIMENSION D_L(NDIVP,NDIAG)
      INTEGER   OFFSET(NDIAG)
C
      IF (NDIAG .LT. 1) THEN
        WRITE (*,*) 'SUBROUTINE INIT_MAT_DIAG:'
        WRITE (*,*) ' NDIAG SHOULD BE GREATER THAN OR EQUAL TO 1'
        RETURN
      ENDIF

!$OMP PARALLEL DEFAULT(PRIVATE)
!$OMP+     SHARED(VA1,VA2,VA3,VC,D_L,OFFSET
!$OMP+      ,NX,NY,NZ,XL,YL,ZL,NDIAG,LEN,NDIVP)

C NDIAG CANNOT BE GREATER THAN 7
      NDIAG_LOC = NDIAG
      IF (NDIAG .GT. 7) NDIAG_LOC = 7

C INITIAL SETTING
      HX = XL/(NX+1)
      HY = YL/(NY+1)
      HZ = ZL/(NZ+1)

!$OMP DO
      DO I = 1,NDIVP
      DO J = 1,NDIAG
      D_L(I,J) = 0.0
      ENDDO
      ENDDO
!$OMP ENDDO

      NXY = NX*NY

C OFFSET SETTING
!$OMP SINGLE
      L = 1
      IF (NDIAG_LOC .GE. 7) THEN
        OFFSET(L) = -NXY
```

```
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 5) THEN
        OFFSET(L) = -NX
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 3) THEN
        OFFSET(L) = -1
        L = L+1
      ENDIF
      OFFSET(L) = 0
      L = L+1
      IF (NDIAG_LOC .GE. 2) THEN
        OFFSET(L) = 1
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 4) THEN
        OFFSET(L) = NX
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 6) THEN
        OFFSET(L) = NXY
      ENDIF
!$OMP END SINGLE

C MAIN LOOP
!$OMP DO
      DO 100 J = 1,LEN
        JS = J

C DECOMPOSE JS-1 = (K0-1)*NX*NY+(J0-1)*NX+I0-1
        K0 = (JS-1)/NXY+1
        IF (K0 .GT. NZ) THEN
          PRINT*,'ERROR; K0.GH.NZ '
          GOTO 100
         ENDIF
        J0 = (JS-1-NXY*(K0-1))/NX+1
        I0 = JS - NXY*(K0-1) - NX*(J0-1)
        L = 1

        IF (NDIAG_LOC .GE. 7) THEN
          IF (K0 .GT. 1) D_L(J,L) = -(1.0/HZ+0.5*VA3)/HZ
```

```
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 5) THEN
          IF (J0 .GT. 1) D_L(J,L) = -(1.0/HY+0.5*VA2)/HY
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 3) THEN
          IF (I0 .GT. 1) D_L(J,L) = -(1.0/HX+0.5*VA1)/HX
          L = L+1
        ENDIF
        D_L(J,L) = 2.0/HX**2+VC
        IF (NDIAG_LOC .GE. 5) THEN
          D_L(J,L) = D_L(J,L) + 2.0/HY**2
          IF (NDIAG_LOC .GE. 7) THEN
            D_L(J,L) = D_L(J,L) + 2.0/HZ**2
          ENDIF
        ENDIF
        L = L+1
        IF (NDIAG_LOC .GE. 2) THEN
          IF (I0 .LT. NX) D_L(J,L) = -(1.0/HX-0.5*VA1)/HX
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 4) THEN
          IF (J0 .LT. NY) D_L(J,L) = -(1.0/HY-0.5*VA2)/HY
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 6) THEN
          IF (K0 .LT. NZ) D_L(J,L) = -(1.0/HZ-0.5*VA3)/HZ
        ENDIF
 100  CONTINUE
!$OMP ENDDO

!$OMP END PARALLEL

      RETURN
      END

C =======================================
* SOLUTE ERROR
* | X1 - X2 |
C =======================================
      REAL*8 FUNCTION ERRNRM(X1,X2,LEN)
```

```
      IMPLICIT  REAL*8 (A-H,O-Z)
      DIMENSION X1(*),X2(*)
C
      S = 0D0
      DO 100 I = 1,LEN
        SS = X1(I) - X2(I)
        S = S + SS * SS
 100  CONTINUE
C
      ERRNRM = SQRT( S )
      RETURN
      END
```

(4)　手法概要

symbolic decomposition を行い,列の間のデータ依存関係や修正コレスキー分解の結果の行列 ***L*** の非零要素の情報を解析します.これをもとに複数の列を束ねた supernode を検出します. supernode にまとめるとき,列の非零のパターンが似ているものをまとめますが,分解過程でのキャッシュのデータを効率よく再利用するために,余分な zero 要素を持つ列も束ねノードを構成する列の数を増やします.

supernode を構成する列に対して,修正コレスキー分解の結果の非零要素の行番号を表す行指標の和集合をもとめます. supernode の修正コレスキー分解の結果は, 1 次元目の大きさがこの行指標の集合の要素数となる 2 次元の panel に圧縮して格納します.行指標の集合は vector で表現します.

Left-looking な変形コレスキー分解で LDL^T 分解します.

技術全般に関しては, “付録 1 参考文献一覧表” の[19]を参照して下さい.

DM_VSCHOLX

LDL^T分解された正値対称スパース行列の連立 1 次方程式
CALL DM_VSCHOLX(N, IORDERING, NPERM, B, NASSIGN, NSUPNUM, NFCNZFACTOR, PANELFACTOR, NSIZEFACTOR, NFCNZINDEX, NPANELINDEX, NSIZEINDEX, NDIM, NPOSTO, IW3, ICON)

(1) 機能

変形コレスキー分解法によりLDL^T分解された$n \times n$の正値対称スパース行列の連立1次方程式を解きます.

$$\boldsymbol{LDL^TQPx=QPb}$$

ただし, $\boldsymbol{P}$はorderingによる行列要素の並び換えを示す置換行列, $\boldsymbol{Q}$はpost orderによる行列要素の並び換えを示す置換行列を示します. $\boldsymbol{P}$,$\boldsymbol{Q}$は直交行列です.

$\boldsymbol{L}$は単位下三角行列, $\boldsymbol{D}$は対角行列です.

$\boldsymbol{b}$は右辺定数ベクトル, $\boldsymbol{x}$は解ベクトルです.

(2) パラメタ

N..... 入力. 行列 $\boldsymbol{A}$ の次数 n.

IORDERING..... 入力. LDL^T 分解を行うときに, ordering を表す直交行列である置換行列 $\boldsymbol{P}$ で $\boldsymbol{PAP^T}$ と変換したかを示します.
1 のとき, $\boldsymbol{PAP^T}$ と変換したものを LDL^T 分解した.
1 以外のとき,行列 $\boldsymbol{A}$ をそのまま LDL^T 分解した.

NPERM ..... 入力. IORDERING=1 のとき使用した置換行列をベクトルで指定します.
NPERM(N)なる 1 次元配列.
((3)使用上の注意 a.注意①参照).

B ..... 入力. $\boldsymbol{Ax=b}$ の右辺定数ベクトル.
出力. 解ベクトル.
B(N)なる 1 次元配列.

NASSIGN..... 入力. 各 supernode は複数の列ベクトルからなり,分解結果に関して共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します.この panel を PANELFACTOR の 1 次元部分配列として順番に割り付けたとき何番目になるかを示します. j=NASSIGN(i)のとき, i 番目の supernode を j 番目に割り付けたことを示します.
分解結果の格納方法については図 DM_VSCHOLX-1 を参照してください.
NASSIGN(N)なる 1 次元配列.

NSUPNUM ..... 入力. supernode の総数.

NFCNZFACTOR .. 入力. 各 supernode は複数の列ベクトルからなり,分解結果に関して共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します.この panel を PANELFACTOR の 1 次元部分配列として順番に割り付けたとき, i 番目の panel の先頭要素 panel(1,1)が 1 次元配列の PANELFACTOR

の何番目の要素になるかを示します.
NFCNZFACTOR(N+1)なる 8 バイト整数型の 1 次元配列.
分解結果の格納方法については図 DM_VSCHOLX-1 を参照してください.
((3)使用上の注意 a.注意③参照).

PANELFACTOR .. 入力. 各 supernode は複数の列ベクトルからなり,分解結果に関して共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します. panel を順番に割り付けます.
i 番目の supernode に対応する panel が何番目の位置に割り当てられるかは j=NASSIGN(i)から分かります.その先頭位置は NFCNZFACTOR(j)に格納されています. panel ごとに分解結果が格納されます.
i 番目に割付けられた panel の大きさは DIM(1,i)×DIM(2,i)の 2 次元配列と見なせます. i 番目の panel の panel(s,t), s>t, s=1,⋯,DIM(1,i), t=1,⋯,DIM(2,i)に単位下三角行列 $\boldsymbol{L}$ の対角要素を除いた対応部分を格納します.対角部分 panel(t,t)には対角行列 $\boldsymbol{D}$ の対応部分を格納します.
PANELFACTOR(NSIZEFACTOR)なる 1 次元配列.
分解結果の格納方法については図 DM_VSCHOLX-1 を参照してください.

NSIZEFACTOR.... 入力. PANELFACTOR の大きさを示す.8 バイト整数型.

NFCNZINDEX ..... 入力. 各 supernode は複数の列ベクトルからなり, 分解結果に関して共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します.この行指標ベクトルを NPANELINDEX に順番に割り付けたとき i 番目の行指標ベクトルの先頭要素が 1 次元配列である NPANELINDEX の何番目の要素になるかを示します.
NFCNZINDEX(N+1)なる 8 バイト整数型の 1 次元配列.
分解結果の格納方法については図 DM_VSCHOLX-1 を参照してください.

NPANELINDEX... 入力. 各 supernode は複数の列ベクトルからなり,分解結果に関して共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します.この行指標ベクトルを NPANELINDEX に順番に割り付けます. i 番目の supernode に対応する panel の行の指標ベクトルが何番目に割り当てられるかは j=NASSIGN(i)から分かります.その先頭位置は NFCNZINDEX(j)に格納されています. panel ごとの行の指標ベクトルを格納します.
この行指標は行列 $\boldsymbol{A}$ を post order で並び換えた行列 $\boldsymbol{QAQ^T}$ での行の番号です.
NPANELINDEX(NSIZEINDEX)なる 1 次元配列.
分解結果の格納方法については図 DM_VSCHOLX-1 を参照してください.

NSIZEINDEX ..... 入力. NPANELINDEX の大きさを示す.8 バイト整数型.

NDIM ..... 入力. NDIM(1,i)および NDIM(2,i)は i 番目に割り付けられた panel の 1 次元目と 2 次元目の大きさを示します.
NDIM(2,N)なる 2 次元配列.

分解結果の格納方法については図 DM_VSCHOLX-1 を参照してください.

NPOSTO ..... 入力. post order で i 番目の node が行列 ***A*** の何番目の列番号に対応するかを表す 1 次元ベクトル.
NPOSTO(N)なる 1 次元配列.
((3)使用上の注意 a.注意②参照).

IW3..... 入力. ルーチン DM_VSCHOL の本ルーチンの呼び出しに先立つ呼び出しで使った作業域 IW3 の内容を変更せず指定します.大きさ N×35+35 の 1 次元配列.
IW3(N*35+35)なる 1 次元配列.

ICON ..... 出力. コンディションコード.
表 DM_VSCHOLX-1 参照.

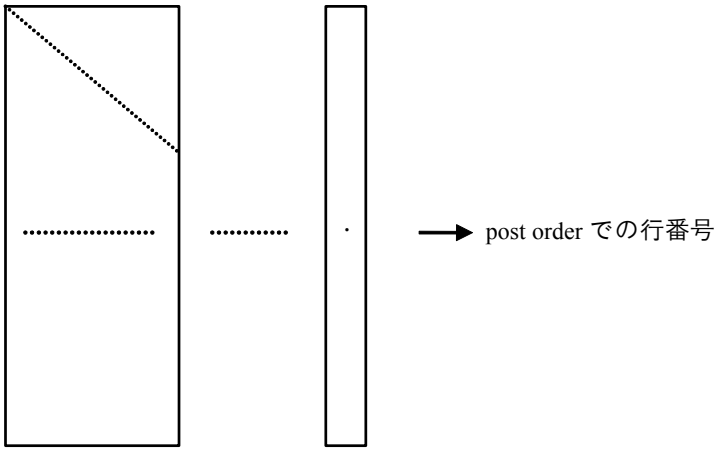


図 DM-VSCHOLX-1 分解結果の格納概念図

j = NASSIGN(i)　　→ i 番目の supernode は j 番目に格納されます.
p = NFCNZFACTOR(j)　→ j 番目の panel は PANELFACTOR の p 番目の要素から
　　DIM(1,j)×DIM(2,j)の長さを占めます.
q = NFCNZINDEX(j)　→ j 番目の panel の行指標を表すベクトルは NPANELINDEX の
　　q 番目の要素から DIM(1,j)の長さを占めます.

panel は大きさ DIM(1,j)×DIM(2,j)の配列と見なせます.
panel(s,t) , s > t , s=1 , … , DIM(1,j),
　　t=1 , … , DIM(2,j)
に分解結果の単位下三角行列 ***L*** の対角要素を除いた部分が格納されます.
panel(t,t)に対角行列 ***D*** の対応部分が格納されます.
行指標の値は行列 ***A*** の列番号をもつ node を post order で並び換えた $\boldsymbol{QAQ}^{\boldsymbol{T}}$ での列番号を表します.

表 DM_VSCHOLX-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし	―
30000	N<1, NSIZEFACTOR<1, NSIZEINDEX<1, NSUPNUM<1	処理を打ち切る.
30100	NPERM に指定した置換行列が正しくない.	

(3) 使用上の注意

a. 注意

① 直交行列である置換行列$\boldsymbol{P}$の要素$\boldsymbol{p_{ij}=1}$のとき, NPERM(i)=jと表現します.
逆は以下のようにして求めることができます.

```
DO i=1,n
j=NPERM(i)
NPERMINV(j)=i
ENDDO
```

② 列番号に対応するノードを考えます.これをpost orderで順番を並び換えたときの番号の対応関係がNPOSTOに格納されています. post orderのi番目のノードが並び換える前の何番目のノードに対応するかを表します. j=nposto(i)はj番目であることを表します.
①同様にこれは直交行列である置換行列$\boldsymbol{Q}$をあらわし,行列$\boldsymbol{A}$を$\boldsymbol{QAQ^T}$と並び換えることに相当します.
逆変換$\boldsymbol{Q^T}$は以下のようにして求めることができます.

```
DO i=1,n
j=NPOSTO(i)
NPOSTOINV(j)=i
ENDDO
```

③ LDL^{T}分解の結果データは, DM_VSCHOL を呼び出した結果を指定して本ルーチンを呼び出すことで連立 1 次方程式を解くことができます.

b. 使用例

連立1次方程式 $\boldsymbol{Ax}=\boldsymbol{f}$ を解きます. 行列$\boldsymbol{A}$は境界条件として立方体の境界で0となる楕円型の偏微分方程式に有限差分法を適用して生成されます.

$$-\Delta u + a\nabla u + cu = f$$

ここで$a=(a_1, a_2, a_3)$, a_1, a_2, a_3およびcはゼロで,ラプラシアンになり行列$\boldsymbol{A}$は正値対称です.行列$\boldsymbol{A}$はサブルーチン init_mat_diagによって生成されます. これを圧縮列格納法に変換します.
(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4プロセッサーのシステムで4つのスレッドで並列実行するときは, OMP_NUM_THREADSを4に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT  REAL*8 (A-H,O-Z)
      PARAMETER (NORD=39,NX = NORD,NY =NORD ,NZ = NORD,
     $      N = NX*NY*NZ)
      PARAMETER (K = N+1)
      PARAMETER (NDIAG = 7,NDIAGH=4)

      DIMENSION NOFST(NDIAG)
```

```
      DIMENSION DIAG(K,NDIAG),DIAG2(K,NDIAG)
      DIMENSION C(K*NDIAG),NROWC(K*NDIAG),NFCNZC(N+1),
     $          WC(K*NDIAG),IWC(2,K*NDIAG)
      DIMENSION A(NDIAGH*N),NROW(K*NDIAG),NFCNZ(N+1),
     $          NPERM(N),NASSIGN(N),W(NDIAGH*N),
     $          NPOSTO(N),NDIM(2,N),
     $          IW1(NDIAGH*N+N+1),
     $          IW2(NDIAGH*N+N+1),
     $          IW3(35*N+35)
      REAL*8, DIMENSION(:), ALLOCATABLE ::  PANELFACTOR
      INTEGER*4, DIMENSION(:), ALLOCATABLE :: NPANELINDEX
      REAL*8 DUMMYF
      INTEGER*4 NDUMMYI
      INTEGER*8 NSIZEFACTOR,NSIZEINDEX,
     $          NFCNZFACTOR(N+1),
     $          NFCNZINDEX(N+1)
      DIMENSION X(N),B(N),SOLEX(N)


      PRINT *,'    LEFT-LOOKING MODIFIED CHOLESKY METHOD'
      PRINT *,'    FOR SPARSE POSITIVE DEFINITE MATRICES'
      PRINT *,'    IN COMPRESSED COLUMN STORAGE'
      PRINT *

      SOLEX(1:N)=1.0D0
      PRINT *,'    EXPECTED SOLUTIONS'
      PRINT *,'    X(1) = ',SOLEX(1),' X(N) = ',SOLEX(N)
      PRINT *

      VA1 = 0.0D0
      VA2 = 0.0D0
      VA3 = 0.0D0
      VC =  0.0D0
      XL = 1.0
      YL = 1.0
      ZL = 1.0
      CALL INIT_MAT_DIAG(VA1,VA2,VA3,VC,DIAG,NOFST
     &          ,NX,NY,NZ,XL,YL,ZL,NDIAG,N,K)

      DO I=1,NDIAG
C
      IF(NOFST(I).LT.0)THEN
```

```
      NBASE=-NOFST(I)
      LENGTH=N-NBASE
      DIAG2(1:LENGTH,I)=DIAG(NBASE+1:N,I)
      ELSE
      NBASE=NOFST(I)
      LENGTH=N-NBASE
      DIAG2(NBASE+1:N,I)=DIAG(1:LENGTH,I)
      ENDIF
C
      ENDDO
C
      NUMNZC=1
      NUMNZ=1
      DO J=1,N
      NTOPCFGC=1
      NTOPCFG=1
      DO I=NDIAG,1,-1
C
      IF(DIAG2(J,I).NE.0.0D0)THEN
C
      NCOL=J-NOFST(I)
      C(NUMNZC)=DIAG2(J,I)
      NROWC(NUMNZC)=NCOL
C
      IF(NCOL.GE.J)THEN
      A(NUMNZ)=DIAG2(J,I)
      NROW(NUMNZ)=NCOL
      ENDIF


C
      IF(NTOPCFGC.EQ.1)THEN
      NFCNZC(J)=NUMNZC
      NTOPCFGC=0
      ENDIF
C
      IF(NTOPCFG.EQ.1)THEN
      NFCNZ(J)=NUMNZ
      NTOPCFG=0
      ENDIF
C
      IF(NCOL.GE.J)THEN
      NUMNZ=NUMNZ+1
```

```
      ENDIF
C
      NUMNZC=NUMNZC+1
      ENDIF
C
      ENDDO
      ENDDO
      NFCNZC(N+1)=NUMNZC
      NNZC=NUMNZC-1
      NFCNZ(N+1)=NUMNZ
      NNZ=NUMNZ-1
C

      CALL DM_VMVSCC(C,NNZC,NROWC,NFCNZC,N,SOLEX,
     $             B,WC,IWC,ICON)
C
      X=B
      IORDERING=0
      ISW=1
      EPSZ=0.0D0
      NSIZEFACTOR=1
      NSIZEINDEX=1

      CALL DM_VSCHOL(A,NNZ,NROW,NFCNZ,N,IORDERING,
     $              NPERM,ISW,EPSZ,NASSIGN,NSUPNUM,
     $              NFCNZFACTOR,DUMMYF,
     $              NSIZEFACTOR,NFCNZINDEX,
     $              NDUMMYI,NSIZEINDEX,NDIM,NPOSTO,
     $              W,IW1,IW2,IW3,ICON)

      PRINT *
      PRINT *,'    ICON = ',ICON,' NSIZEFACTOR = ',NSIZEFACTOR,
     $       'NSIZEINDEX = ',NSIZEINDEX
      PRINT *
C
C     ALLOCATE STORAGES IN RETURNED SIZES
C
      ALLOCATE( PANELFACTOR(NSIZEFACTOR) )
      ALLOCATE( NPANELINDEX(NSIZEINDEX) )

      ISW=2
```

```
      CALL DM_VSCHOL(A,NNZ,NROW,NFCNZ,N,IORDERING,
     $              NPERM,ISW,EPSZ,NASSIGN,NSUPNUM,
     $              NFCNZFACTOR,PANELFACTOR,
     $              NSIZEFACTOR,NFCNZINDEX,
     $              NPANELINDEX,NSIZEINDEX,NDIM,NPOSTO,
     $              W,IW1,IW2,IW3,ICON)

      CALL DM_VSCHOLX(N,IORDERING,
     $              NPERM,X,NASSIGN,NSUPNUM,
     $              NFCNZFACTOR,PANELFACTOR,
     $              NSIZEFACTOR,NFCNZINDEX,
     $              NPANELINDEX,NSIZEINDEX,NDIM,NPOSTO,
     $              IW3,ICON)

      ERR = ERRNRM(SOLEX,X,N)

      PRINT *,'    COMPUTED VALUES'
      PRINT *,'    X(1) = ',X(1),' X(N) = ',X(N)
      PRINT *
      PRINT *,'    ICON = ',ICON
      PRINT *
      PRINT *,'    N = ',N,' :: NX = ',NX,' NY = ',NY,' NZ = ',NZ
      PRINT *
      PRINT *,'    ERROR = ',ERR
      PRINT *
      PRINT *

      IF(ERR.LT.1.0D-8.AND.ICON.EQ.0)THEN
         WRITE(*,*)'    ********** OK **********'
      ELSE
         WRITE(*,*)'    ********** NG **********'
      ENDIF

      DEALLOCATE( PANELFACTOR,NPANELINDEX )

      STOP
      END

C =======================================
C     INITIALIZE COEFFICIENT MATRIX
C =======================================
      SUBROUTINE INIT_MAT_DIAG(VA1,VA2,VA3,VC,D_L,OFFSET
```

```
     &                         ,NX,NY,NZ,XL,YL,ZL,NDIAG,LEN,NDIVP)
      IMPLICIT  REAL*8(A-H,O-Z)
      DIMENSION D_L(NDIVP,NDIAG)
      INTEGER   OFFSET(NDIAG)
C
      IF (NDIAG .LT. 1) THEN
        WRITE (*,*) 'SUBROUTINE INIT_MAT_DIAG:'
        WRITE (*,*) ' NDIAG SHOULD BE GREATER THAN OR EQUAL TO 1'
        RETURN
      ENDIF

!$OMP PARALLEL DEFAULT(PRIVATE)
!$OMP+     SHARED(VA1,VA2,VA3,VC,D_L,OFFSET
!$OMP+      ,NX,NY,NZ,XL,YL,ZL,NDIAG,LEN,NDIVP)

C NDIAG CANNOT BE GREATER THAN 7
      NDIAG_LOC = NDIAG
      IF (NDIAG .GT. 7) NDIAG_LOC = 7

C INITIAL SETTING
      HX = XL/(NX+1)
      HY = YL/(NY+1)
      HZ = ZL/(NZ+1)

!$OMP DO
      DO I = 1,NDIVP
      DO J = 1,NDIAG
      D_L(I,J) = 0.0
      ENDDO
      ENDDO
!$OMP ENDDO

      NXY = NX*NY

C OFFSET SETTING
!$OMP SINGLE
      L = 1
      IF (NDIAG_LOC .GE. 7) THEN
        OFFSET(L) = -NXY
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 5) THEN
```

```
        OFFSET(L) = -NX
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 3) THEN
        OFFSET(L) = -1
        L = L+1
      ENDIF
      OFFSET(L) = 0
      L = L+1
      IF (NDIAG_LOC .GE. 2) THEN
        OFFSET(L) = 1
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 4) THEN
        OFFSET(L) = NX
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 6) THEN
        OFFSET(L) = NXY
      ENDIF
!$OMP END SINGLE

C MAIN LOOP
!$OMP DO
      DO 100 J = 1,LEN
        JS = J

C DECOMPOSE JS-1 = (K0-1)*NX*NY+(J0-1)*NX+I0-1
        K0 = (JS-1)/NXY+1
        IF (K0 .GT. NZ) THEN
          PRINT*,'ERROR; K0.GH.NZ '
          GOTO 100
         ENDIF
        J0 = (JS-1-NXY*(K0-1))/NX+1
        I0 = JS - NXY*(K0-1) - NX*(J0-1)
        L = 1

        IF (NDIAG_LOC .GE. 7) THEN
          IF (K0 .GT. 1) D_L(J,L) = -(1.0/HZ+0.5*VA3)/HZ
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 5) THEN
```

```
          IF (J0 .GT. 1) D_L(J,L) = -(1.0/HY+0.5*VA2)/HY
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 3) THEN
          IF (I0 .GT. 1) D_L(J,L) = -(1.0/HX+0.5*VA1)/HX
          L = L+1
        ENDIF
        D_L(J,L) = 2.0/HX**2+VC
        IF (NDIAG_LOC .GE. 5) THEN
          D_L(J,L) = D_L(J,L) + 2.0/HY**2
          IF (NDIAG_LOC .GE. 7) THEN
            D_L(J,L) = D_L(J,L) + 2.0/HZ**2
          ENDIF
        ENDIF
        L = L+1
        IF (NDIAG_LOC .GE. 2) THEN
          IF (I0 .LT. NX) D_L(J,L) = -(1.0/HX-0.5*VA1)/HX
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 4) THEN
          IF (J0 .LT. NY) D_L(J,L) = -(1.0/HY-0.5*VA2)/HY
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 6) THEN
          IF (K0 .LT. NZ) D_L(J,L) = -(1.0/HZ-0.5*VA3)/HZ
        ENDIF
 100  CONTINUE
!$OMP ENDDO

!$OMP END PARALLEL

      RETURN
      END

C =======================================
* SOLUTE ERROR
* | X1 - X2 |
C =======================================
      REAL*8 FUNCTION ERRNRM(X1,X2,LEN)
      IMPLICIT  REAL*8 (A-H,O-Z)
      DIMENSION X1(*),X2(*)
C
```

```
      S = 0D0
      DO 100 I = 1,LEN
        SS = X1(I) - X2(I)
        S = S + SS * SS
 100  CONTINUE
C
      ERRNRM = SQRT( S )
      RETURN
      END
```

DM_VSCLU

複素スパース行列の LU 分解
CALL DM_VSCLU(ZA, NZ, NROW, NFCNZ, N, IPLEDSM, MZ, ISCLITERMAX, IORDERING, NPERM, ISW, NROWSYM, NFCNZSYM, NASSIGN, NSUPNUM, NFCNZFACTORL, ZPANELFACTORL, NSIZEFACTORL, NFCNZINDEXL, NPANELINDEXL, NSIZEINDEXL, NDIM, NFCNZFACTORU, ZPANELFACTORU, NSIZEFACTORU, NFCNZINDEXU, NPANELINDEXU, NSIZEINDEXU, NPOSTO, SCLROW,SCLCOL, EPSZ, THEPSZ, IPIVOT, ISTATIC, SPEPSZ, NFCNZPIVOT, NPIVOTP, NPIVOTQ, ZW, W, IW1, IW2, ICON)

(1)　機能

$n \times n$ の複素スパース行列 $\boldsymbol{A}$ に対角要素に大きな要素を並べる並び換えを行い, 行および列の均衡化を行うスケーリングを行います. スーパーノードの対角ブロック内で指定した Pivot をとり, LU 分解します.

要素の並び換えや スケーリングおよび Pivot では, 複素数の大きさを表す絶対値は, 実部の絶対値と虚部の絶対値を加えたもので近似します.

複素スパース行列 $\boldsymbol{A}$ は以下のように変換できます.

$$\boldsymbol{A_1=D_rAP_cD_c}$$

ここで $\boldsymbol{P_c}$ は列の入れ換えを行う直交行列, $\boldsymbol{D_r}$ は行のスケーリングを行う対角行列, $\boldsymbol{D_c}$ は列のスケーリングを行う対角行列です.

$$\boldsymbol{A_2=QPA_1P^TQ^T}$$

$\boldsymbol{A_2}$ は, スーパーノードの対角ブロックで指定された pivot を行い, 行および列の入れ換えを行い LU 分解されます.

ただし, $\boldsymbol{P}$ は, $\boldsymbol{SYM}= \boldsymbol{A_1}+\boldsymbol{A_1^T}$ の非ゼロパターンに対して求めた ordering による行列要素の並び換えを示す置換行列で, $\boldsymbol{Q}$ は $\boldsymbol{SYM}$ に対する post order による行列要素の並び換えを示す置換行列を示します. $\boldsymbol{P}$, $\boldsymbol{Q}$ は直交行列です.

$\boldsymbol{L}$ は下三角行列, $\boldsymbol{U}$ は単位上三角行列です.

Pivot処理で, 閾値 THEPSZ より絶対値が大きな Pivot を見つけることが出来なかったとき, 対角ブロック内の絶対値が最大の要素を Pivot の候補とします. この値が小さ過ぎるときに Static Pivot を与えて近似的に LU 分解を続けることができます.

(2)　パラメタ

ZA ..... 入力. 複素スパースな係数行列 $\boldsymbol{A}$ を圧縮列格納法で ZA (1:NZ)に格納します.

ZA(NZ)なる倍精度複素数型の 1 次元配列.

圧縮列格納法については, 実スパース行列と実ベクトル(圧縮列格納法), DM_VMVSCC の図 DM_VMVSCC-1 を参照してください. 実数型の配列 CC を複素数型の配列に変えたものを利用します.
((3)使用上の注意 a. 注意⑤参照).

NZ ..... 入力. 複素スパースな係数行列 $\boldsymbol{A}$ の非零要素の総数.

NROW..... 入力. 圧縮列格納法で使用される行指標で ZA に格納される要素が何番目の行ベクトルに属するかを示します.
NROW(NZ)なる 1 次元配列.

NFCNZ..... 入力. 行列 $\boldsymbol{A}$ の各列の非零要素を列方向に圧縮して順次配列 ZA に格納するとき,対応する列の最初の非零要素が格納される位置を表します.
NFCNZ(N+1)=NZ+1.
NFCNZ(N+1)なる 1 次元配列.

N..... 入力. 行列 $\boldsymbol{A}$ の次数 n.

IPLEDSM..... 入力. 対角要素に大きな要素を並べる列の入れ換えを行うかを指定します.
1 のとき, 列の入れ換えを求めて入れ換えます.
1 以外のとき, 列の入れ換えを行いません.

MZ..... 出力. IPLEDSM に 1 が指定されたとき, 列の入れ換えを表す. MZ(i)=j は行列 $\boldsymbol{a_{ij}}$ の属する j 番目の列を i 番目の列に移動する. $\boldsymbol{a_{ij}}$ は対角要素に並べる大きな要素を表す.
MZ(N)なる 1 次元配列.

ISCLITERMAX.... 入力. 係数行列のスケーリングを行う対角行列 $\boldsymbol{D_r}$ と $\boldsymbol{D_c}$ を反復して求める反復回数.
ISCLITERMAX ≤ 0 のときスケーリングは行わすに, $\boldsymbol{D_r}$ および $\boldsymbol{D_c}$ は単位行列が設定されます.
ISCLITERMAX ≥ 10 のときは 10 回を上限値とします.

IORDERING..... 入力. ordering を表す置換行列 $\boldsymbol{P}$ で $\boldsymbol{PA_1P^T}$ と変換した行列を LU 分解するかを指定します. 置換行列は直交行列です.
10 のとき, ISW=1 で呼び出すと ordering を求める $\boldsymbol{A_1}$ に関する情報を求めて NROWSYM, NFCNZSYM に設定します.
11 のとき, ISW=1 で 10 を指定して呼び出して得た行列を対称化した情報 NROWSYM, NFCNZSYM を使って決めた ordering を NPERM に指定して同じく ISW=1 で呼び出し, 計算を続けることを示します. $\boldsymbol{PA_1P^T}$ を LU 分解する処理を続けます.
10,11 以外のとき, ordering は指定せずに行列 $\boldsymbol{A_1}$ をそのまま LU 分解します.
出力. ISW=1 と IORDERING=10 を指定して呼び出した後, IORDERING に 11 が設定されます. そのため, ordering を NPERM に指定して再び呼び出すときに特に 11 を指定する必要はありません.
((3)使用上の注意 a. 注意①参照).

NPERM ..... 入力. IORDERING=11 のとき使用する置換行列をベクトルで指定します.

NPERM(N)なる 1 次元配列.
((3)使用上の注意 a. 注意①参照).

ISW ..... 入力. 呼び出しに関する制御情報を示します.
1)　1 のとき
対称化および symbolic decomposition を行い, 必要な領域が割り当てられているか調べ計算を行います.
IORDERING=10 で呼び出し, ordering を求めるための情報が NROWSYM, NFCNZSYM に出力されます. これらを使って ***SYM*** に対する ordering を求めた後, NPERM に指定して IORDERING=11 を指定してもう 1 回 ISW=1 で呼び出します.
IORDERING が 10,11 以外のときは ordering は行ないません.
2)　2 のとき
ISW=1 で呼び出したとき, ZPANELFACTORL,
ZPANELFACTORU, NPANELINDEXL または NPANELINDEXU の大きさが不足して ICON=31000 で終了した処理を継続します. このとき, NSIZEFACTORL, NSIZEFACTORU, NSIZEINDEXL または NSIZEINDEXU に返却された必要な大きさで ZPANELFACTORL, ZPANELFACTORU, NPANELINDEXL または NPANELINDEXU を確保し直して指定しなおして再度呼び出します.
これらの引数と実行を制御する引数 ISW 以外の引数に格納されている値は変更してはいけません.

NROWSYM ..... 出力. IORDERING=10 で呼び出したとき, 対称化した $\boldsymbol{SYM}=\boldsymbol{A_1}+\boldsymbol{A_1}^T$ の非ゼロパターンの下三角行列部分の行指標を列圧縮したものが返却されます.
IPLEDSM=1 のときは, $\boldsymbol{A_1}=\boldsymbol{D_r}\boldsymbol{A}\boldsymbol{P_c}\boldsymbol{D_c}$ です.
NROWSYM(NZ+N)なる 1 次元配列.

NFCNZSYM ..... 出力. IORDERING=10 で呼び出したとき, 行列 SYM の下三角部分の各列の非零要素の行指標を列方向に圧縮して順次配列 NROWSYM に格納するとき,対応する列の最初の非零要素が格納される位置を表します.
NFCNZSYM(N+1)=NSYMZ+1 NSYMZ は,総要素数を表します.
NFCNZSYM(N+1)なる 1 次元配列.

NASSIGN..... 出力. 各 supernode に対する L, U は圧縮して 2 次元の panel に格納します. この panel を PANELFACTORL および PANELFACTORU の 1 次元部分配列として順番に割り付けたとき何番目になるかを示します. これらの指標ベクトルも同じように NPANELINDEXL, NPANELINDEXU に割り付けます. j=NASSIGN(i)のとき, i 番目の supernode を j 番目に割り付けたことを示します.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
分解結果の格納方法については図 DM_VSCLU-1 を参照してください.
NASSIGN(N)なる 1 次元配列.

NSUPNUM ..... 出力. supernode の総数.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します. (≤n)

NFCNZFACTORL 出力. 複素スパース行列の LU 分解の結果の行列 ***L*** および ***U*** はスーパーノードに対応しておのおの求めます. 各 supernode に対応する ***L*** の列ベクトルは, 共通の行指標ベクトルを持つように圧縮してブロック対角部分に ***U*** の対応部分を含めて 2 次元の panel に格納します. この panel を ZPANELFACTORL の 1 次元部分配列として順番に割り付けたとき, i 番目の panel の先頭要素 panel(1,1)が 1 次元配列の ZPANELFACTORL の何番目の要素になるかを示します.

NFCNZFACTORL(N+1)なる 8 バイト整数型の 1 次元配列.

結果の格納方法については図 DM_VSCLU -1 を参照してください.

入力. ISW≠1 のとき,初回呼び出しの値を再利用します.

ZPANELFACTORL出力. 各 supernode の分解結果に対応する行列 ***L*** の複数の列ベクトルと行列 ***U*** のブロック対角部分について，共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します. panel を順番に割り付けます. i 番目の supernode に対応する panel が何番目の位置に割り当てられるかは j=NASSIGN(i)から分かります. その先頭位置は NFCNZFACTORL(j)に格納されています. panel ごとに分解結果が格納されます.

i 番目に割付けられた panel の大きさは DIM(1,i)×DIM(2,i)の 2 次元配列と見なせます. i 番目の panel の panel(s,t), $s \geq t$, s=1,…,DIM(1,i), t=1,…,DIM(2,i)に下三角行列 ***L*** が格納されます. panel の panel(s,t), $s < t$, t=1,…,DIM(2,i)には単位上三角行列 ***U*** の対角要素を除いたブロック対角部分が格納されます.

ZPANELFACTORL (NSIZEFACTORL)なる倍精度複素数型の 1 次元配列.

結果の格納方法については図 DM_VSCLU-1 を参照してください.

((3)使用上の注意 a. 注意③参照).

NSIZEFACTORL . 入力. ZPANELFACTORL の大きさを示す. 8 バイト整数型.

出力. ZPANELFACTORL の大きさとして必要な大きさが返却されます.

((3)使用上の注意 a. 注意③参照).

NFCNZINDEXL ... 出力. 各 supernode の分解結果に対応する行列 ***L*** の複数の列ベクトルと行列 ***U*** のブロック対角部分について，共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します. この行指標ベクトルを NPANELINDEXL に順番に割り付けたとき i 番目の行指標ベクトルの先頭要素が 1 次元配列である NPANELINDEXL の何番目の要素になるかを示します.

NFCNZINDEXL(N+1)なる 8 バイト整数型の 1 次元配列.

入力. ISW≠1 のとき,初回呼び出しの値を再利用します.

結果の格納方法については図 DM_VSCLU-1 を参照してください.

NPANELINDEXL 出力. 各 supernode の分解結果に対応する行列 ***L*** の複数の列ベクトルと行列 ***U*** のブロック対角部分について，共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します. この行指標ベクトルを NPANELINDEXL に順番に割り付けます. i 番目の supernode に対応する panel の行の指標ベクトルが何番目に割り当てられるかは j=NASSIGN(i)から分かります. その先頭位置は NFCNZINDEXL(j)に格

納されています.
この行指標は行列 ***SYM*** に対する post order で並び換えたときの行の番号です.
NPANELINDEXL(NSIZEINDEXL)なる 1 次元配列.
結果の格納方法については図 DM_VSCLU-1 を参照してください.
((3)使用上の注意 a. 注意③参照).

NSIZEINDEXL..... 入力. NPANELINDEXL の大きさを示す. 8 バイト整数型.
出力. 必要な大きさが返却されます.
((3)使用上の注意 a. 注意③参照).

NDIM ..... 出力. NDIM(1,i)および NDIM(2,i)は行列 ***L*** に関して i 番目に割り付けられた panel の 1 次元目と 2 次元目の大きさを示します.
NDIM(3,i)は行列 ***U*** に関して割り付けられた panel を転置したときの 1 次元目の大きさに対角ブロックの大きさを加えた大きさを示します.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
NDIM(3,N)なる 2 次元配列.
結果の格納方法については図 DM_VSCLU-1 を参照してください.

NFCNZFACTORU 出力. 複素スパース行列の LU 分解の結果の行列 ***U*** に関しては, 各 supernode に対応する ***U*** の行ベクトルは, ブロック対角部分を除いて共通の列指標ベクトルを持つように圧縮し, 転置したものを 2 次元の panel に格納します. この panel を ZPANELFACTORU の 1 次元部分配列として順番に割り付けたとき, i 番目の panel の先頭要素 panel(1,1)が 1 次元配列の ZPANELFACTORU の何番目の要素になるかを示します.
NFCNZFACTORU(N+1)なる 8 バイト整数型の 1 次元配列.
結果の格納方法については図 DM_VSCLU-1 を参照してください.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.

ZPANELFACTORU 出力. 各 supernode の分解結果に対応する行列 ***U*** のブロック対角部分を除いた複数の行ベクトルについて，共通の列指標ベクトルを持つように圧縮し, 転置して 2 次元の panel に格納します. panel を順番に割り付けます. i 番目の supernode に対応する panel が何番目の位置に割り当てられるかは j=NASSIGN(i) から分かります. その先頭位置は NFCNZFACTORU(j)に格納されています. panel ごとに分解結果が格納されます.
i 番目に割付けられた panel の大きさは{DIM(3,i)-DIM(2,i)}×DIM(2,i)の 2 次元配列と見なせます. i 番目の panel の panel(s,t), s=1,…,DIM(3,i)-DIM(2,i), t=1,…,DIM(2,i)に単位上三角行列 ***U*** のブロック対角部分を除いた部分が行ベクトルを圧縮し, 転置して格納します.
ZPANELFACTORU (NSIZEFACTORU)なる倍精度複素数型の 1 次元配列.
結果の格納方法については図 DM_VSCLU-1 を参照してください.
((3)使用上の注意 a. 注意③参照).

NSIZEFACTORU. 入力. ZPANELFACTORU の大きさを示す. 8 バイト整数型.

出力. ZPANELFACTORU の大きさとして必要な大きさが返却されます.
((3)使用上の注意 a. 注意③参照).

NFCNZINDEXU... 出力. 各 supernode の分解結果に対応する行列 ***U*** は対角ブロック部分を除いて複数の行ベクトルについて, 共通の列指標ベクトルを持つように圧縮し, 転置して 2 次元の panel に格納します. 対角ブロック部分も含んだこの列指標ベクトルを NPANELINDEXU に順番に割り付けたとき i 番目の列指標ベクトルの先頭要素が 1 次元配列である NPANELINDEXU の何番目の要素になるかを示します.
NFCNZINDEXU(N+1)なる 8 バイト整数型の 1 次元配列.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
結果の格納方法については図 DM_VSCLU-1 を参照してください.

NPANELINDEXU 出力. 各 supernode の分解結果に対応する行列 ***U*** の対角ブロックを除いた部分を転置して圧縮して 2 次元の panel に格納します. この列指標ベクトルを対角ブロック部分を含めて NPANELINDEXU に順番に割り付けます. i 番目の supernode に対応する panel の行の指標ベクトルが何番目に割り当てられるかは j=NASSIGN(i)から分かります. その先頭位置は NFCNZINDEXU(j)に格納されています.
この行指標は行列 ***SYM*** に対する post order で並び換えたときの列の番号です.
NPANELINDEXU(NSIZEINDEXU)なる 1 次元配列.
結果の格納方法については図 DM_VSCLU-1 を参照してください.
((3)使用上の注意 a. 注意③参照).

NSIZEINDEXU .... 入力. NPANELINDEXU の大きさを示す. 8 バイト整数型.
出力. 必要な大きさが返却されます.
((3)使用上の注意 a. 注意③参照).

NPOSTO ..... 出力. post order で i 番目の node が行列 ***A*** の何番目の列番号に対応するかを表す 1 次元ベクトル.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
NPOSTO(N)なる 1 次元配列.
((3)使用上の注意 a. 注意④参照).

SCLROW ..... 出力. スケーリング対角行列 $\boldsymbol{D}_r$. 対角要素が 1 次元配列に格納されます.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
SCLROW(N)なる 1 次元配列.

SCLCOL..... 出力. スケーリング対角行列 $\boldsymbol{D}_c$. 対角要素が 1 次元配列に格納されます.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
SCLCOL(N)なる 1 次元配列.

EPSZ ..... 入力. 分解過程で対角要素の大きさの絶対値の相対判定値
出力. EPSZ ≤ 0.0 のときは標準値が設定されます.
((3)使用上の注意 a. 注意②参照).

THEPSZ ..... 入力. ピボットの判定での閾値(threshold). これより大きな値はピボットとして採用します. 見つかればその時点で, ピボット検索は打ち切ります.

例えば 1.0D-2 程度.
出力. THEPSZ ≤ 0.0 のときは 1.0D-2 が設定されます.
EPSZ ≥ THEPSZ > 0.0 のときは, EPSZ が設定されます.

IPIVOT..... 入力. スーパーノード内でピボットを行うか, 行う場合どのようなピボットを行うかを指定します. 例えば, 完全ピボットとして 40 を指定.
IPIVOT < 10 : または IPIVOT ≥ 50: ピボットなし.
10 ≤ IPIVOT < 20 : 部分ピボット
20 ≤ IPIVOT < 30 : 対角ピボット
21 : スーパーノード内で対角ピボットがとれないとき, Rook ピボットに変更します.
22 : スーパーノード内で対角ピボットがとれないときは, Rook ピボットに, Rook ピボットが取れないときは完全ピボットに変更します.
30 ≤ IPIVOT < 40 :Rook ピボット
32 : スーパーノード内で Rook ピボットがとれないとき, 完全ピボットをとります.
40 ≤ IPIVOT < 50 : 完全ピボット

ISTATIC..... 入力. Pivot を指定したとき, Static Pivot を行うかを示します.
1) = 1 のとき
スーパーノード内で指定したピボットが SPEPSZ より大きくないとき, ピボットとして大きさ SPEPSZ の複素数で近似します. ピボットの大きさの値が 0.0d0 なら SPEPSZ に近似します.
このとき, 以下を設定しなければなりません.
a) EPSZ は EPSZ の標準値以下にしなければなりません.
b) ISCLITERMAX は 10 を設定しスケーリングを行う必要があります.
c)THEPSZ ≥ SPEPSZ でなければなりません.
2) ≠1 のとき
Static Pivot は行いません.

SPEPSZ..... 入力. ISTATIC=1 のとき, Static Pivot として使う値.
THEPSZ ≥ SPEPSZ ≥ EPSZ でなければなりません.
出力. SPEPSZ < EPSZ のときは, 1.0D-10 が設定されます.

NFCNZPIVOT ..... 出力. スーパーノード内の相対的な位置でのピボットの行, 列の入れ換えの履歴を格納する位置を示す. i 番目の supernode に関する情報が何番目の位置に割り当てられるかはj=NASSIGN(i)から分かります.その先頭位置は NFCNZPIVOT(j)に格納されています. i 番目のスーパーノードの入れ換え情報は, NPIVOTP, NPIVOTQ の is=NFCNZPIVOT(j) ,…, ie=NFCNZPIVOT(j)+NDIM(2,j)-1 番目の要素に格納されます.
NFCNZPIVOT(NSUPNUM+1)なる 1 次元配列.

NPIVOTP ..... 出力. スーパーノード内の行の入れ換えに関する情報を格納する.
NPIVOTP(N)なる 1 次元配列.

NPIVOTQ ..... 出力. スーパーノード内の列の入れ換えに関する情報を格納する.
NPIVOTQ(N)なる 1 次元配列.

ZW.....作業域.
出力/入力.
大きさ 2*NZ の倍精度複素数型の 1 次元配列.
ISW=1, 2 で続けて呼び出すとき,呼び出しの間で必要なデータを受け渡すために使われます. 呼び出しの間で値を変更してはいけません.

W.....作業域.
出力/入力.
大きさ 4*NZ+6*N の 1 次元配列.
ISW=1, 2 で続けて呼び出すとき,呼び出しの間で必要なデータを受け渡すために使われます. 呼び出しの間で値を変更してはいけません.

IW1.....作業域.
出力/入力.
大きさ 2*NZ+2*(N+1)+16*N の 1 次元配列.
ISW=1, 2 で続けて呼び出すとき,呼び出しの間で必要なデータを受け渡すために使われます. 呼び出しの間で値を変更してはいけません.

IW2.....作業域.
出力/入力.
大きさ 47*N+47+NZ+4*(N+1)+2*(NZ+N)の 1 次元配列.
ISW=1, 2 で続けて呼び出すとき,呼び出しの間で必要なデータを受け渡すために使われます. 呼び出しの間で値を変更してはいけません.

ICON.....出力. コンディションコード.
表 DM_VSCLU-1 参照

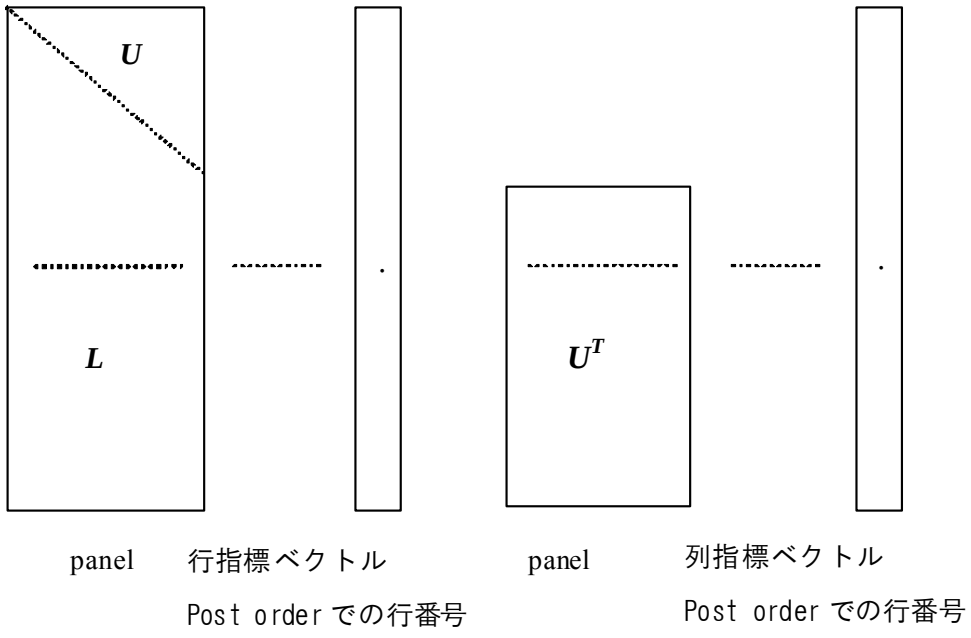


図 DM_VSCLU-1 分解結果の格納概念図

j = NASSIGN(i)　　→ i 番目の supernode は j 番目に格納されます.

p = NFCNZFACTORL(j) → j 番目の panel は ZPANELFACTORL の p 番目の要素から DIM(1,j)×DIM(2,j)の長さを占めます.

q = NFCNZINDEXL(j) → j 番目の panel の行指標を表すベクトルは NPANELINDEXL の q 番目の要素から DIM(1,j)の長さを占めます.

panel は大きさ DIM(1,j)×DIM(2,j)の配列と見なせます.

panel(s,t) ， s ≥ t , s = 1 , … , DIM(1,j), t = 1 , … , DIM(2,j) に分解結果の下三角行列 $\boldsymbol{L}$ が格納されます.

panel(s,t) ， s < t , s=1 , … , DIM(2,j), t=1 , … , DIM(2,j) に単位上三角行列 $\boldsymbol{U}$ の対角ブロック部分が対角要素を除き格納されます.

u = NFCNZFACTORU(j) → j 番目の panel は ZPANELFACTORU の u 番目の要素から (DIM(3,j)-DIM(2,j))×DIM(2,j)の長さを占めます.

v = NFCNZINDEXU(j) → j 番目の panel の列指標を表すベクトルは NPANELINDEXU の v 番目の要素から DIM(3,j)の長さを占めます.

panel は大きさ (DIM(3,j)-DIM(2,j))×DIM(2,j)の配列と見なせます.

panel(x,y) ， x=1 , … , DIM(3,j)-DIM(2,j), y=1 , … , DIM(2,j)

に分解結果の単位上三角行列 $\boldsymbol{U}$ の転置行列 $\boldsymbol{U}^T$ の対角ブロック部分を除いた部分が格納されます.

指標の値は行列 $\boldsymbol{A}$ の列番号をもつ node を post order で並び換えた $\boldsymbol{QAQ}^T$ での列番号を表します.

表 DM_VSCLU-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし	ー
10000	ISTATIC=1 のとき小さすぎるピボットを SPEPSZ で置き換える Static Pivot を行いました.	ー
20000	ピボットが相対的に零になりました.	処理を打ち切ります.
20100	IPLEDSM=1 を指定して,対角要素に絶対値が大きな要素を並べるため maximum matching を求める処理で, 長さ N の maximum matching がみつかりませんでした. 行列が特異である可能性があります.	
20200	行および列の均衡化を行う対角行列の対角要素を求める過程で,元の行列 A の行もしくは列にゼロベクトルがありました. 行列が特異である可能性があります.	

コード	意　　味	処理内容
30000	N<1,NZ<0,NFCNZ(N+1)≠NZ+1, NSIZEFACTORL<1, NSIZEFACTORU<1, NSIZEINDEXL<1, NSIZEINDEXU<1, ISW<1, ISW>2	処理を打ち切ります.
30100	NPERM に指定した置換行列が正しくない.	
30200	NROW(j)に格納された i 列目の行指標 k が k<1 または k>N	
30300	i 列目の行指標の数 NFCNZ(i+1)-NFCNZ(i)>n	
30500	ISTATIC=1 のとき満たすべき条件をみたしていない. EPSZ は標準値 16***u*** より大きかった. または ISCLITERMAX <10 であった. または SPEPSZ>THEPSZ であった.	
31000	ZPANELFACTORL の大きさ NSIZEFACTORL または NPANELINDEXL の大きさ NSIZEINDEXL または ZPANELFACTORU の大きさ NSIZEFACTORU または NPANELINDEXU の大きさ NSIZEINDEXU が小さすぎます.	NSIZEFACTORL または NSIZEINDEXL または NSIZEFACTORU または NSIZEINDEXU で指定された 大きさで ZPANELFACTORL または NPANELINDEXL または ZPANELFACTORU または NPANELINDEXU を割り付けて ISW=2 として再度呼び出す.

(3)　使用上の注意

a.　注意

①　直交行列である置換行列$\boldsymbol{P}$の要素$p_{ij}=1$のとき, NPERM(i)=jと表現します.
逆は以下のようにして求めることができます.

```
DO i=1,n
 j=NPERM(i)
 NPERMINV(j)=i
 ENDDO
```

Fill-Reducing OrderingはMETISなどを使って求めることができます.
詳細は“付録1 参考文献一覧表”の[43],[44]を参照願います.

②　分解過程で対角要素の大きさの絶対値の相対零判定値にある値を設定したとすると, この値は次の意味を持っています. すなわち, LU分解の過程で, 対角要素の大きさの絶対値がその値より小さくなった場合に, その値を相対的に零と見なし, ICON=20000として処理を打ち切ります. EPSZの標準値は, 丸め誤差の単位をuとしたとき, EPSZ=16uです.
Pivotでは, 複素数の大きさを表す絶対値は, 実部の絶対値と虚部の絶対値を加えたもので近似します.

なお, 対角要素の大きさの絶対値が小さくなっても, 処理を続行させたい場合には, EPSZに極小の値を与えれば良いのですが, 結果の精度は保証されません. Static Pivotを指定した場合, 対角要素がSPEPSZより小さいとき, 大きさSPEPSZの複素数で近似してLU分解を行います.

③　分解結果を格納する配列ZPANELFACTORL, NPANELINDEXL, ZPANELFACTORU, NPANELINDEXUの必要な大きさは,事前には分かりません. 十分大きな配列を割り当てるか,本ルーチンを呼び出してsymbolic decompositionを行った解析の結果を使って割り当てを行うことができます. 例えば, 大きさ1の1次元配列などを割付けます. そしてその大きさ1などの小さな値をNSIZEFACTORL, NSIZEINDEXL, NSIZEFACTORU, NSIZEINDEXUに指定して, ISW=1で呼び出します.
symbolic decompositionを行い, ICON=31000で終了し, NSIZEFACTORL, NSIZEINDEXL, NSIZEFACTORU, NSIZEINDEXUに必要な大きさが返却されます. 必要な大きさの配列を割付け直して引数に指定して, ISW=2で呼び出すことで, symbolic decomposition以降の処理を続けることができます. 使用例を参照願います.
LU分解の結果を利用してDM_VSCLUXを利用して連立1次方程式を解く上での注意は(3)使用上の注意a. 注意⑤参照.

④　列番号に対応するノードを考えます. これをpost orderで順番を並び換えたときの番号の対応関係がNPOSTOに格納されています. post orderのi番目のノードが並び換える前の何番目のノードに対応するかを表します. j=nposto(i)はj番目であることを表します.
①同様にこれは直交行列である置換行列$\boldsymbol{Q}$を表し,行列$\boldsymbol{A}$を$\boldsymbol{QAQ^T}$と並び換えることに相当します.
逆変換$\boldsymbol{Q^T}$は以下のようにして求めることができます.

```
 DO i=1,n
  j=NPOSTO(i)
  NPOSTOINV(j)=i
  ENDDO
```

⑤　本ルーチンで得られたLU分解の結果を使い, DM_VSCLUXを引き続いて呼びだすことで連立1次方程式$\boldsymbol{Ax=b}$を解くことができます.
このとき, DM_VSCLUで利用した以下の引数を指定します. 使用例を参照願います.

```
ZA, NZ, NROW, NFCNZ, N,
IPLEDSM, MZ, IORDERING, NPERM,
NASSIGN, NSUPNUM,
NFCNZFACTORL, ZPANELFACTORL,
NSIZEFACTORL, NFCNZINDEXL, NPANELINDEXL,
NSIZEINDEXL, NDIM,
NFCNZFACTORU, ZPANELFACTORU, NSIZEFACTORU,
NFCNZINDEXU, NPANELINDEXU, NSIZEINDEXU, NPOSTO,
SCLROW,SCLCOL,
```

NFCNZPIVOT,
NPIVOTP, NPIVOTQ, IW2

b. 使用例

連立 1 次方程式 $\boldsymbol{Ax} = \boldsymbol{f}$ を解きます.

行列は境界条件として立方体の境界で 0 となる楕円型の偏微分方程式に有限差分法を適用して生成されたものを利用します.

$$-\Delta u + a\nabla u + cu = f$$

ここで $a = (a_1, a_2, a_3)$.

行列はサブルーチン init_mat_diag によって生成されます.

対角形式格納法で生成し, 対角ベクトルの下 6 本を圧縮列格納法で格納します. 結果生成された非対称な行列 $\boldsymbol{A}$ を複素スパース行列に格納して連立 1 次方程式を解きます.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT  REAL*8 (A-H,O-Z)
      PARAMETER (NORD=40,KX = NORD,KY =NORD ,KZ = NORD,
     $      N = KX*KY*KZ)
      PARAMETER (NBORDER=N+1,NOFFDIAG=6)
      PARAMETER (K = N+1)
      PARAMETER (NDIAG = 7)
      INTEGER*4 WL,ZWL
      PARAMETER (NALL=NDIAG*N,
C
     $   ZWL =2*NALL,
     $   WL  =4*NALL+6*N,
     $   IW1L=2*NALL+2*(N+1)+16*N,
     $   IW2L=47*N+47+4*(N+1)+NALL+2*(NALL+N))
C
      DIMENSION NOFST(NDIAG)
      DIMENSION DIAG(K,NDIAG),DIAG2(K,NDIAG)
      COMPLEX*16 ZA(K*NDIAG),ZWC(K*NDIAG),
     $           ZW(ZWL),ZONE
      PARAMETER(ZONE=(1.0D0,0.0D0))
      DIMENSION NROW(K*NDIAG),NFCNZ(N+1),
     $          NROWSYM(K*NDIAG+N),NFCNZSYM(N+1),
     $          IWC(2,K*NDIAG)
      DIMENSION NPERM(N),W(WL),
     $          NPOSTO(N),NDIM(3,N),
     $          NASSIGN(N),
```

```
     $          MZ(N),
     $          IW1(IW1L),IW2(IW2L)
      COMPLEX*16, DIMENSION(:), ALLOCATABLE ::
     $        ZPANELFACTORL,ZPANELFACTORU
      INTEGER*4, DIMENSION(:), ALLOCATABLE :: NPANELINDEXL,NPANELINDEXU
      COMPLEX*8 ZDUMMYFL,ZDUMMYFU
      INTEGER*4 NDUMMYIL,
     $          NDUMMYIU
      INTEGER*8 NSIZEFACTORL,
     $          NSIZEINDEXL,
     $          NSIZEINDEXU,
     $          NSIZEFACTORU,
     $          NFCNZFACTORL(N+1),
     $          NFCNZFACTORU(N+1),
     $          NFCNZINDEXL(N+1),
     $          NFCNZINDEXU(N+1)
      COMPLEX*16 ZB(N),ZSOLEX(N)
      REAL*8 THEPSZ,EPSZ,SPEPSZ,
     $       SCLROW(N),SCLCOL(N)
C
      INTEGER*4     IPIVOT,ISTATIC,NFCNZPIVOT(N+1),
     $              NPIVOTP(N),NPIVOTQ(N),
     $              IREFINE,ITERMAX,ITER,IPLEDSM
C
      PRINT *,'    LU DECOMPOSITION METHOD'
      PRINT *,'    FOR SPARSE UNSYMMETRIC COMPLEX MATRICES'
      PRINT *,'    IN COMPRESSED COLUMN STORAGE'
      PRINT *
C
      DO I=1,N
      ZSOLEX(I)= ZONE
      ENDDO
      PRINT *,'    EXPECTED SOLUTIONS'
      PRINT *,'    X(1) = ',ZSOLEX(1),' X(N) = ',ZSOLEX(N)
      PRINT *
C
      VA1 = 1.0D0
      VA2 = 2.0D0
      VA3 = 3.0D0
      VC =  4.0D0
      XL = 1.0
      YL = 1.0
```

```
      ZL = 1.0
      CALL INIT_MAT_DIAG(VA1,VA2,VA3,VC,DIAG,NOFST
     &          ,KX,KY,KZ,XL,YL,ZL,NDIAG,N,K)
C
      DIAG2=0
C
      DO I=1,NDIAG
C
      IF(NOFST(I).LT.0)THEN
      NBASE=-NOFST(I)
      LENGTH=N-NBASE
      DIAG2(1:LENGTH,I)=DIAG(NBASE+1:N,I)
      ELSE
      NBASE=NOFST(I)
      LENGTH=N-NBASE
      DIAG2(NBASE+1:N,I)=DIAG(1:LENGTH,I)
      ENDIF
C
      ENDDO
C
      NUMNZ=1
C
      DO J=1,N
      NTOPCFG=1
C
      DO I=NDIAG,1,-1
C
      IF(NTOPCFG.EQ.1)THEN
      NFCNZ(J)=NUMNZ
      NTOPCFG=0
      ENDIF
C
      IF(J.LT.NBORDER.AND.I.GT.NOFFDIAG)THEN
      CONTINUE
      ELSE
C
      IF(DIAG2(J,I).NE.0.0D0)THEN
C
      NCOL=J-NOFST(I)
      ZA(NUMNZ)=DCMPLX(DIAG2(J,I),0.0D0)
      NROW(NUMNZ)=NCOL
C
```

```
      NUMNZ=NUMNZ+1
C
      ENDIF
      ENDIF
      ENDDO
      ENDDO
C
      NFCNZ(N+1)=NUMNZ
      NZ=NUMNZ-1
C
      CALL DM_VMVSCCC(ZA,NZ,NROW,NFCNZ,N,ZSOLEX,
     $             ZB,ZWC,IWC,ICON)
C
C     INITIAL CALL WITH IORDER=1
C
      IORDERING= 0
      IPLEDSM=1
      ISCLITERMAX=10
      ISW=1
      NSIZEFACTORL=1
      NSIZEFACTORU=1
      NSIZEINDEXL=1
      NSIZEINDEXU=1
      EPSZ=1.0D-16
      THEPSZ=1.0D-2
      SPEPSZ=0.0D0
      IPIVOT=40
      ISTATIC=0
      IREFINE=1
      EPSR=0.0D0
      ITERMAX=10
C
      CALL DM_VSCLU(ZA,NZ,NROW,NFCNZ,N,
     $              IPLEDSM,MZ,ISCLITERMAX,IORDERING,
     $              NPERM,ISW,
     $              NROWSYM,NFCNZSYM,
     $              NASSIGN,
     $              NSUPNUM,
     $              NFCNZFACTORL,ZDUMMYFL,
     $              NSIZEFACTORL,
     $              NFCNZINDEXL,
     $              NDUMMYIL,NSIZEINDEXL,
```

```
     $              NDIM,
     $              NFCNZFACTORU,ZDUMMYFU,
     $              NSIZEFACTORU,
     $              NFCNZINDEXU,
     $              NDUMMYIU,NSIZEINDEXU,
     $              NPOSTO,
     $              SCLROW,SCLCOL,
     $              EPSZ,THEPSZ,
     $              IPIVOT,ISTATIC,SPEPSZ,NFCNZPIVOT,
     $              NPIVOTP,NPIVOTQ,
     $              ZW,W,IW1,IW2,ICON)
C
      PRINT*,'ICON=',ICON,' NSIZEFACTORL=',NSIZEFACTORL,
     $       ' NSIZEFACTORU=',NSIZEFACTORU,
     $       'NSIZEINDEXL=',NSIZEINDEXL,
     $       'NSIZEINDEXU=',NSIZEINDEXU,
     $       'NSUPNUM=',NSUPNUM
C
      ALLOCATE( ZPANELFACTORL(NSIZEFACTORL) )
      ALLOCATE( ZPANELFACTORU(NSIZEFACTORU) )
      ALLOCATE( NPANELINDEXL(NSIZEINDEXL) )
      ALLOCATE( NPANELINDEXU(NSIZEINDEXU) )
C
      ISW=2
C
      CALL DM_VSCLU(ZA,NZ,NROW,NFCNZ,N,
     $              IPLEDSM,MZ,ISCLITERMAX,IORDERING,
     $              NPERM,ISW,
     $              NROWSYM,NFCNZSYM,
     $              NASSIGN,
     $              NSUPNUM,
     $              NFCNZFACTORL,ZPANELFACTORL,
     $              NSIZEFACTORL,
     $              NFCNZINDEXL,
     $              NPANELINDEXL,NSIZEINDEXL,
     $              NDIM,
     $              NFCNZFACTORU,ZPANELFACTORU,
     $              NSIZEFACTORU,
     $              NFCNZINDEXU,
     $              NPANELINDEXU,NSIZEINDEXU,
     $              NPOSTO,
     $              SCLROW,SCLCOL,
```

```
     $              EPSZ,THEPSZ,
     $              IPIVOT,ISTATIC,SPEPSZ,NFCNZPIVOT,
     $              NPIVOTP,NPIVOTQ,
     $              ZW,W,IW1,IW2,ICON)
C
      CALL DM_VSCLUX(N,
     $              IORDERING,
     $              NPERM,
     $              ZB,
     $              NASSIGN,
     $              NSUPNUM,
     $              NFCNZFACTORL,ZPANELFACTORL,
     $              NSIZEFACTORL,
     $              NFCNZINDEXL,
     $              NPANELINDEXL,NSIZEINDEXL,
     $              NDIM,
     $              NFCNZFACTORU,ZPANELFACTORU,
     $              NSIZEFACTORU,
     $              NFCNZINDEXU,
     $              NPANELINDEXU,NSIZEINDEXU,
     $              NPOSTO,
     $              IPLEDSM,MZ,
     $              SCLROW,SCLCOL,
     $              NFCNZPIVOT,
     $              NPIVOTP,NPIVOTQ,
     $              IREFINE,EPSR,ITERMAX,ITER,
     $              ZA,NZ,NROW,NFCNZ,
     $              IW2,ICON)
C
      ERR = ERRNRM(ZSOLEX,ZB,N)

      PRINT *,'    COMPUTED VALUES'
      PRINT *,'    X(1) = ',ZB(1),' X(N) = ',ZB(N)
      PRINT *
      PRINT *,'    ICON = ',ICON
      PRINT *
      PRINT *,'    N = ',N
      PRINT *
      PRINT *,'    ERROR = ',ERR
      PRINT *,'    ITER=',ITER
      PRINT *
      PRINT *
```

```
      IF(ERR.LT.1.0D-8.AND.ICON.EQ.0)THEN
         WRITE(*,*)'********** OK **********'
      ELSE
         WRITE(*,*)'********** NG **********'
      ENDIF
C
      DEALLOCATE( ZPANELFACTORL,ZPANELFACTORU,
     $            NPANELINDEXL,
     $            NPANELINDEXU )

      STOP
      END

C =======================================
C     INITIALIZE COEFFICIENT MATRIX
C =======================================
      SUBROUTINE INIT_MAT_DIAG(VA1,VA2,VA3,VC,D_L,OFFSET
     &                ,NX,NY,NZ,XL,YL,ZL,NDIAG,LEN,NDIVP)
      IMPLICIT  REAL*8(A-H,O-Z)
      DIMENSION D_L(NDIVP,NDIAG)
      INTEGER   OFFSET(NDIAG)
C
      IF (NDIAG .LT. 1) THEN
        WRITE (*,*) 'SUBROUTINE INIT_MAT_DIAG:'
        WRITE (*,*) ' NDIAG SHOULD BE GREATER THAN OR EQUAL TO 1'
        RETURN
      ENDIF

!$OMP PARALLEL DEFAULT(PRIVATE)
!$OMP+     SHARED(VA1,VA2,VA3,VC,D_L,OFFSET
!$OMP+      ,NX,NY,NZ,XL,YL,ZL,NDIAG,LEN,NDIVP)

C NDIAG CANNOT BE GREATER THAN 7
      NDIAG_LOC = NDIAG
      IF (NDIAG .GT. 7) NDIAG_LOC = 7

C INITIAL SETTING
      HX = XL/(NX+1)
      HY = YL/(NY+1)
      HZ = ZL/(NZ+1)
```

```
!$OMP DO
      DO I = 1,NDIVP
      DO J = 1,NDIAG
      D_L(I,J) = 0.0
      ENDDO
      ENDDO
!$OMP ENDDO

      NXY = NX*NY

C OFFSET SETTING
!$OMP SINGLE
      L = 1
      IF (NDIAG_LOC .GE. 7) THEN
        OFFSET(L) = -NXY
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 5) THEN
        OFFSET(L) = -NX
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 3) THEN
        OFFSET(L) = -1
        L = L+1
      ENDIF
      OFFSET(L) = 0
      L = L+1
      IF (NDIAG_LOC .GE. 2) THEN
        OFFSET(L) = 1
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 4) THEN
        OFFSET(L) = NX
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 6) THEN
        OFFSET(L) = NXY
      ENDIF
!$OMP END SINGLE

C MAIN LOOP
!$OMP DO
```

```
      DO 100 J = 1,LEN
        JS = J

C DECOMPOSE JS-1 = (K0-1)*NX*NY+(J0-1)*NX+I0-1
        K0 = (JS-1)/NXY+1
        IF (K0 .GT. NZ) THEN
          PRINT*,'ERROR; K0.GH.NZ '
          GOTO 100
        ENDIF
        J0 = (JS-1-NXY*(K0-1))/NX+1
        I0 = JS - NXY*(K0-1) - NX*(J0-1)
        L = 1

        IF (NDIAG_LOC .GE. 7) THEN
          IF (K0 .GT. 1) D_L(J,L) = -(1.0/HZ+0.5*VA3)/HZ
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 5) THEN
          IF (J0 .GT. 1) D_L(J,L) = -(1.0/HY+0.5*VA2)/HY
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 3) THEN
          IF (I0 .GT. 1) D_L(J,L) = -(1.0/HX+0.5*VA1)/HX
          L = L+1
        ENDIF
        D_L(J,L) = 2.0/HX**2+VC
        IF (NDIAG_LOC .GE. 5) THEN
          D_L(J,L) = D_L(J,L) + 2.0/HY**2
          IF (NDIAG_LOC .GE. 7) THEN
            D_L(J,L) = D_L(J,L) + 2.0/HZ**2
          ENDIF
        ENDIF
        L = L+1
        IF (NDIAG_LOC .GE. 2) THEN
          IF (I0 .LT. NX) D_L(J,L) = -(1.0/HX-0.5*VA1)/HX
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 4) THEN
          IF (J0 .LT. NY) D_L(J,L) = -(1.0/HY-0.5*VA2)/HY
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 6) THEN
```

```
            IF (K0 .LT. NZ) D_L(J,L) = -(1.0/HZ-0.5*VA3)/HZ
          ENDIF
 100  CONTINUE
!$OMP ENDDO

!$OMP END PARALLEL

      RETURN
      END
C =======================================
* SOLUTE ERROR
* | Z1 - Z2 |
C =======================================
      REAL*8 FUNCTION ERRNRM(Z1,Z2,LEN)
      IMPLICIT  REAL*8 (A-H,O-Z)
      COMPLEX*16 Z1(*),Z2(*),SS
C
      S = 0D0
      DO 100 I = 1,LEN
        SS  = Z1(I) - Z2(I)
        S = S + DREAL(SS * DCONJG(SS))
 100  CONTINUE
C
      ERRNRM = SQRT( S )
      RETURN
      END
```

(4)　手法概要

対角要素に絶対値の大きな要素を並べる並び換えを行います.
並び換えを行った行列に対して行, 列の均衡化を行うスケーリングを施します.
要素の並び換えや スケーリングおよびピボットでは, 複素数の大きさを表す絶対値は, 実部の絶対値と虚部の絶対値を加えたもので近似します.
この行列を LU 分解します.
スーパーノードに対応する非ゼロ要素を 2 次元の panel に格納します.
安定化のためのピボットは対角ブロック部分で検索します.
指定した値より絶対値が大きな値が見つかるとピボットとして採用する閾値 THEPSZ を指定できます. 求めたピボットが小さすぎるとき, SPEPSZ で指定した値で近似して LU 分解を続ける static pivot を指定することも可能です.
詳細は“付録 1 参考文献一覧表”の以下の文献を参照願います.
対角要素に絶対値の大きな要素を並べる方法に関しては,の[23],[57]をマッチングの応用アルゴリズムの詳細は[13]を, フィボナッチヒープに関しては[17]を,スパースな

非対称な複素行列の LU 分解のベースに関しては[19],[2],[22],[48],[68]を行列の均衡化やピボットに関しては[63],[69]をそれぞれ参照願います.

DM_VSCLUX

LU 分解された複素スパース行列の連立 1 次方程式
CALL DM_VSCLUX(N, IORDERING, NPERM ZB, NASSIGN, NSUPNUM, NFCNZFACTORL, ZPANELFACTORL, NSIZEFACTORL, NFCNZINDEXL, NPANELINDEXL, NSIZEINDEXL, NDIM, NFCNZFACTORU, ZPANELFACTORU, NSIZEFACTORU, NFCNZINDEXU, NPANELINDEXU, NSIZEINDEXU, NPOSTO, IPLEDSM, MZ, SCLROW,SCLCOL, NFCNZPIVOT, NPIVOTP, NPIVOTQ, IREFINE, EPSR, ITERMAX, ITER, ZA, NZ, NROW, NFCNZ, IW2, ICON)

(1)　機能

$n \times n$ の複素スパース行列 $\boldsymbol{A}$ に対角要素に大きな要素を並べる並び換えを行い, 行および列の均衡化を行うスケーリングを行い, そしてスーパーノード内で Pivot をとり, 以下のように LU 分解されています. LU 分解の結果を利用して以下の連立 1 次方程式を解きます.
要素の並び換えや スケーリングおよび Pivot では, 複素数の大きさを表す絶対値は, 実部の絶対値と虚部の絶対値を加えたもので近似します.

$$\boldsymbol{Ax=b}$$

行列$\boldsymbol{A}$は以下のように分解されています.

$$\boldsymbol{P_{rs}QPD_rAP_cD_cP^TQ^TP_{cs}=LU}$$

ここで, 複素スパース行列$\boldsymbol{A}$は以下のように変換されています.

$$\boldsymbol{A_1=D_rAP_cD_c}$$

ここで $\boldsymbol{P_c}$ は列の入れ換えを行う直交行列, $\boldsymbol{D_r}$ は行のスケーリングを行う対角行列, $\boldsymbol{D_c}$ は列のスケーリングを行う対角行列です.

$$\boldsymbol{A_2=QPA_1P^TQ^T}$$

$\boldsymbol{A_2}$は, スーパーノード内に閉じて指定された pivot を行い,行および列の入れ換えを行い LU 分解されています.
$\boldsymbol{P_{rs}}$, $\boldsymbol{P_{cs}}$ はそれぞれ行の入れ換えおよび列の入れ換えを示す直交行列です. 実際の入れ換えはスーパーノードに属する限られた行列の部分で行われます.
ただし, $\boldsymbol{P}$ は,$\boldsymbol{SYM}= \boldsymbol{A_1+A_1^T}$ の非ゼロパターンに対して求めた ordering による行列要素の並び換えを示す置換行列で, $\boldsymbol{Q}$ は $\boldsymbol{SYM}$ に対する post order による行列要素の並び換えを示す置換行列を示します. $\boldsymbol{P}$, $\boldsymbol{Q}$ は直交行列です.
$\boldsymbol{L}$ は下三角行列, $\boldsymbol{U}$ は単位上三角行列です.
解の反復改良で精度を改良することを指定できます.

(2)　パラメタ

N..... 入力. 行列 $\boldsymbol{A}$ の次数 n.

IORDERING..... 入力. 11 のとき, NPERM に ordering を指定して LU 分解されたことを示します. $\boldsymbol{PA_1P^T}$ を LU 分解しました.
11 以外のとき, ordering は指定されていません.
((3)使用上の注意 a. 注意①参照).

NPERM ..... 入力. IORDERING=11 のとき使用する置換行列をベクトルで指定します.
NPERM(N)なる 1 次元配列.
((3)使用上の注意 a. 注意②参照).

ZB..... 入力. $\boldsymbol{Ax=b}$ の右辺定数ベクトル.
出力. 解ベクトル.
ZB (N)なる倍精度複素数型の 1 次元配列.

NASSIGN..... 入力. 各 supernode に対する $\boldsymbol{L}$, $\boldsymbol{U}$ は圧縮して 2 次元の panel に格納します. この panel を ZPANELFACTORL および ZPANELFACTORU の 1 次元部分配列として順番に割り付けたとき何番目になるかを示します. これらの指標ベクトルも同じように NPANELINDEXL, NPANELINDEXU に割り付けます. j=NASSIGN(i)のとき, i 番目の supernode を j 番目に割り付けたことを示します.
分解結果の格納方法については図 DM_VSCLUX-1 を参照してください.
NASSIGN(N)なる 1 次元配列.

NSUPNUM ..... 入力. supernode の総数. (≤n)

NFCNZFACTORL 入力. 複素スパース行列の LU 分解の結果の行列 $\boldsymbol{L}$ および $\boldsymbol{U}$ はスーパーノードに対応しておのおの求めます. 各 supernode に対応する $\boldsymbol{L}$ の列ベクトルは, 共通の行指標ベクトルを持つように圧縮してブロック対角部分に $\boldsymbol{U}$ の対応部分を含めて 2 次元の panel に格納します. この panel を ZPANELFACTORL の 1 次元部分配列として順番に割り付けたとき, i 番目の panel の先頭要素 panel(1,1)が 1 次元配列の ZPANELFACTORL の何番目の要素になるかを示します.
NFCNZFACTORL(N+1)なる 8 バイト整数型の 1 次元配列.
結果の格納方法については図 DM_VSCLUX-1 を参照してください.

ZPANELFACTORL 入力. 各 supernode の分解結果に対応する行列 $\boldsymbol{L}$ の複数の列ベクトルと行列 $\boldsymbol{U}$ のブロック対角部分について, 共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します. panel を順番に割り付けます. i 番目の supernode に対応する panel が何番目の位置に割り当てられるかは j=NASSIGN(i)から分かります. その先頭位置は NFCNZFACTORL(j)に格納されています. panel ごとに分解結果が格納されます.
i 番目に割付けられた panel の大きさは DIM(1,i)×DIM(2,i)の 2 次元配列と見なせます. i 番目の panel の panel(s,t), $s \geq t$, $s = 1,\ldots,$DIM(1,i), $t=1,\ldots,$DIM(2,i)に下三角行列 $\boldsymbol{L}$ が格納されます. panel の panel(s,t), $s<t$, $t=1,\ldots,$DIM(2,i)には単位上三角行列 $\boldsymbol{U}$ の対角要素を除いたブロック対角部分が格納されます.
ZPANELFACTORL (NSIZEFACTORL)なる倍精度複素数型の 1 次元配列.
結果の格納方法については図 DM_VSCLUX-1 を参照してください.

NSIZEFACTORL . 入力. ZPANELFACTORL の大きさを示す. 8 バイト整数型.

NFCNZINDEXL ... 入力. 各 supernode の分解結果に対応する行列 ***L*** の複数の列ベクトルと行列 ***U*** のブロック対角部分について，共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します. この行指標ベクトルを NPANELINDEXL に順番に割り付けたとき i 番目の行指標ベクトルの先頭要素が 1 次元配列である NPANELINDEXL の何番目の要素になるかを示します.

NFCNZINDEXL(N+1)なる 8 バイト整数型の 1 次元配列.

NPANELINDEXL 入力. 各 supernode の分解結果に対応する行列 ***L*** の複数の列ベクトルと行列 ***U*** のブロック対角部分について，共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します. この行指標ベクトルを NPANELINDEXL に順番に割り付けます. i 番目の supernode に対応する panel の行の指標ベクトルが何番目に割り当てられるかは j=NASSIGN(i)から分かります. その先頭位置は NFCNZINDEXL(j)に格納されています.

この行指標は行列 ***SYM*** に対する post order で並び換えたときの行の番号です.

NPANELINDEXL(NSIZEINDEXL)なる 1 次元配列.

結果の格納方法については図 DM_VSCLUX-1 を参照してください.

NSIZEINDEXL ..... 入力. NPANELINDEXL の大きさを示す. 8 バイト整数型.

NDIM ..... 入力. NDIM(1,i)および NDIM(2,i)は行列 ***L*** に関して i 番目に割り付けられた panel の 1 次元目と 2 次元目の大きさを示します.

NDIM(3,i)は行列 ***U*** に関して割り付けられた panel を転置したときの 1 次元目の大きさに対角ブロックの大きさを加えた大きさを示します.

NDIM(3,N)なる 2 次元配列.

結果の格納方法については図 DM_VSCLUX-1 を参照してください.

NFCNZFACTORU 入力. 複素スパース行列の LU 分解の結果の行列 ***U*** に関しては, 各 supernode に対応する ***U*** の行ベクトルは, ブロック対角部分を除いて共通の列指標ベクトルを持つように圧縮し, 転置して 2 次元の panel に格納します. この panel を PANELFACTORU の 1 次元部分配列として順番に割り付けたとき, i 番目の panel の先頭要素 panel(1,1)が 1 次元配列の ZPANELFACTORU の何番目の要素になるかを示します.

NFCNZFACTORU(N+1)なる 8 バイト整数型の 1 次元配列.

結果の格納方法については図 DM_VSCLUX-1 を参照してください.

ZPANELFACTORU 入力. 各 supernode の分解結果に対応する行列 ***U*** のブロック対角部分を除いた複数の行ベクトルについて，共通の列指標ベクトルを持つように圧縮し, 転置して 2 次元の panel に格納します. panel を順番に割り付けます. i 番目の supernode に対応する panel が何番目の位置に割り当てられるかは j=NASSIGN(i) から分かります. その先頭位置は NFCNZFACTORU(j)に格納されています. panel ごとに分解結果が格納されます.

i 番目に割付けられた panel の大きさは{DIM(3,i)-DIM(2,i)}×DIM(2,i)の 2

次元配列と見なせます. i 番目の panel の panel(s,t), s=1,…,DIM(3,i)-DIM(2,i), t=1,…,DIM(2,i)に単位上三角行列 ***U*** のブロック対角部分を除いた部分が行ベクトルを圧縮し, 転置したものが格納されます.
ZPANELFACTORU (NSIZEFACTORU)なる倍精度複素数型の 1 次元配列.
結果の格納方法については図 DM_VSCLUX-1 を参照してください.

NSIZEFACTORU. 入力. ZPANELFACTORU の大きさを示す. 8 バイト整数型.

NFCNZINDEXU... 入力. 各 supernode の分解結果に対応する行列 ***U*** は対角ブロック部分を除いて複数の行ベクトルについて, 共通の列指標ベクトルを持つように圧縮し, 転置した形で 2 次元の panel に格納します. 対角ブロック部分も含んだこの列指標ベクトルを NPANELINDEXU に順番に割り付けたとき i 番目の列指標ベクトルの先頭要素が 1 次元配列である NPANELINDEXU の何番目の要素になるかを示します.
NFCNZINDEXU(N+1)なる 8 バイト整数型の 1 次元配列.
結果の格納方法については図 DM_VSCLUX-1 を参照してください.

NPANELINDEXU 入力. 各 supernode の分解結果に対応する行列 ***U*** の対角ブロックを除いた部分を転置して圧縮して 2 次元の panel に格納します. 対応する列指標ベクトルには対角ブロック部分を含めたものを NPANELINDEXU に順番に割り付けます. i 番目の supernode に対応する panel の行の指標ベクトルが何番目に割り当てられるかは j=NASSIGN(i)から分かります. その先頭位置は NFCNZINDEXU(j)に格納されています.
この行指標は行列 ***SYM*** に対する post order で並び換えたときの列の番号です.
NPANELINDEXU(NSIZEINDEXU)なる 1 次元配列.
結果の格納方法については図 DM_VSCLUX-1 を参照してください.

NSIZEINDEXU .... 入力. NPANELINDEXU の大きさを示す. 8 バイト整数型.

NPOSTO ..... 入力. post order で i 番目の node が行列 ***A*** の何番目の列番号に対応するかを表す 1 次元ベクトル.
((3)使用上の注意 a. 注意③参照).

IPLEDSM..... 入力. LU 分解を行うときに対角要素に大きな要素を並べる列の入れ換えを行ったかを指定します.
1 のとき, 列の入れ換えを行いました.
1 以外のとき, 列の入れ換えを行っていません.

MZ..... 入力. IPLEDSM に 1 が指定されたとき, 列の入れ換えを表す. MZ(i)=j は行列 a_{ij} の属する j 番目の列を i 番目の列に移動する. a_{ij} は対角要素に並べる大きな要素を表す.
MZ(N)なる 1 次元配列.

SCLROW ..... 入力. スケーリング対角行列 D_r. 対角要素が 1 次元配列に格納されます.
SCLROW(N)なる 1 次元配列.

SCLCOL..... 入力. スケーリング対角行列 D_c. 対角要素が 1 次元配列に格納されます.
SCLCOL(N)なる 1 次元配列.

NFCNZPIVOT ..... 入力. スーパーノード内の相対的な位置でのピボットの行, 列の入れ換えの履歴を格納する位置を示す. i 番目の supernode に関する情報が何番目の位置に割り当てられるかは j=NASSIGN(i)から分かります. その先頭位置は NFCNZPIVOT(j)に格納されています. i 番目のスーパーノードの入れ換え情報は, NPIVOTP, NPIVOTQ の is=NFCNZPIVOT(j) ,…, ie=NFCNZPIVOT(j)+NDIM(2,j)-1 番目の要素に格納されます.
NFCNZPIVOT(NSUPNUM+1)なる 1 次元配列.

NPIVOTP ..... 入力. スーパーノード内の行の入れ換えに関する情報を格納します.
NPIVOTP(N)なる 1 次元配列.

NPIVOTQ ..... 入力. スーパーノード内の列の入れ換えに関する情報を格納します.
NPIVOTQ(N)なる 1 次元配列.

IREFINE..... 入力. LU 分解結果を利用して解を求めるときに, 解の反復改良を行うかどうかを示す. 残差ベクトルを計算するときに 4 倍精度で計算します.
=1 : 解の反復改良を行う. 反復改良して得られる残差ベクトル $\boldsymbol{r}_k$ の絶対値の差分がひとつ前の差分の 1/4 より大きくなるまで, 反復改良を行います.
≠1: 解の反復改良を行わない.

EPSR ..... 入力. 解の残差ベクトル $\boldsymbol{b}$-$\boldsymbol{Ax}$ の絶対値が, $\boldsymbol{b}$ の絶対値に対して十分小さいかを判定する判定値
EPSR ≤ 0.0 のとき 1.0D-6 が設定されます.

ITERMAX..... 入力. (≥ 1) 反復改良を行うときの最大反復回数.

ITER..... 出力. 反復改良を行った回数.

ZA ..... 入力. 複素スパースな係数行列 $\boldsymbol{A}$ を圧縮列格納法で ZA(1:NZ)に格納します.
ZA(NZ)なる倍精度複素数型の 1 次元配列.
圧縮列格納法については,実スパース行列と実ベクトル(圧縮列格納法), DM_VMVSCC の図 DM_VMVSCC-1 を参照してください. 実数型の配列 CC を複素数型の配列に変えたものを利用します.

NZ ..... 入力. 複素スパースな係数行列 $\boldsymbol{A}$ の非零要素の総数.

NROW..... 入力. 圧縮列格納法で使用される行指標で ZA に格納される要素が何番目の行ベクトルに属するかを示します.
NROW(NZ)なる 1 次元配列.

NFCNZ..... 入力. 行列 $\boldsymbol{A}$ の各列の非零要素を列方向に圧縮して順次配列 ZA に格納するとき,対応する列の最初の非零要素が格納される位置を表します.
NFCNZ(N+1)=NZ+1.
NFCNZ(N+1)なる 1 次元配列.

IW2..... 作業域.
入力.
大きさ 47*N+47+NZ+4*(N+1)+2*(NZ+N)の 1 次元配列.
複素スパース行列の LU 分解を行う DM_VSCLU からのデータの受け渡しに使われます. 呼び出しの間で値を変更してはいけません.

ICON ..... 出力.コンディションコード.

表 DM_VSCLUX-1 参照

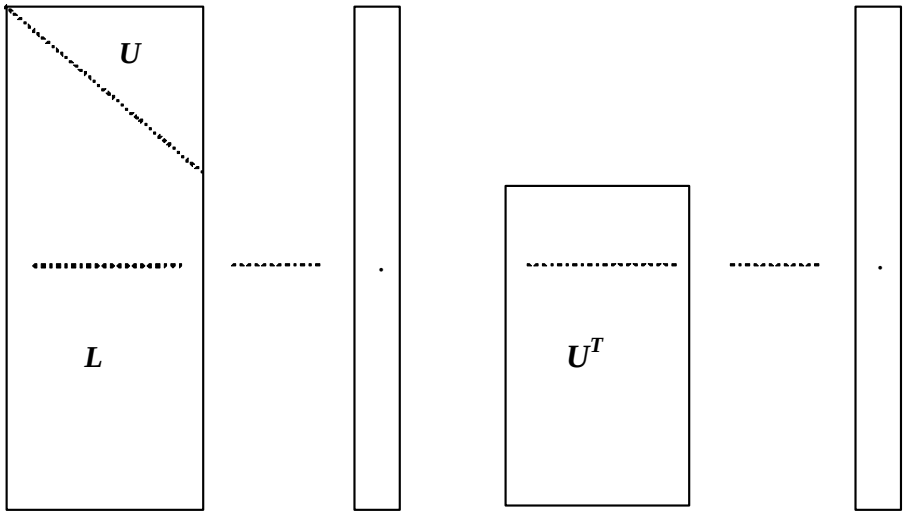


図 DM_VSCLUX-1 分解結果の格納概念図

j = NASSIGN(i)　→ i 番目の supernode は j 番目に格納されます.

p = NFCNZFACTORL(j) → j 番目の panel は ZPANELFACTORL の p 番目の要素から DIM(1,j)×DIM(2,j)の長さを占めます.

q = NFCNZINDEXL(j) → j 番目の panel の行指標を表すベクトルは NPANELINDEXL の q 番目の要素から DIM(1,j)の長さを占めます.

panel は大きさ DIM(1,j)×DIM(2,j)の配列と見なせます.

panel(s,t) , $s \geq t$, s = 1 , … , DIM(1,j), t=1 , … , DIM(2,j) に分解結果の下三角行列 $\boldsymbol{L}$ が格納されます.

panel(s,t) , $s < t$, s = 1 , … , DIM(2,j), t=1 , … , DIM(2,j) に単位上三角行列 $\boldsymbol{U}$ の対角ブロック部分が対角要素を除き格納されます.

u = NFCNZFACTORU(j) → j 番目の panel は ZPANELFACTORU の u 番目の要素から (DIM(3,j)-DIM(2,j))×DIM(2,j)の長さを占めます.

v = NFCNZINDEXU(j) → j 番目の panel の列指標を表すベクトルは NPANELINDEXU の v 番目の要素から DIM(3,j)の長さを占めます.

panel は大きさ(DIM(3,j)-DIM(2,j))×DIM(2,j)の配列と見なせます.

panel(x,y) , x=1 , … , DIM(3,j)-DIM(2,j), y=1 , … , DIM(2,j)

に分解結果の単位上三角行列 $\boldsymbol{U}$ の転置行列 $\boldsymbol{U^T}$ の対角ブロック部分を除いた部分が格納されます.

指標の値は行列 $\boldsymbol{A}$ の列番号をもつ node を post order で並び換えた $\boldsymbol{QAQ^T}$ での列番号を表します.

表 DM_VSCLUX-1　コンディションコード使用上の注意

	意　　味	処理内容
0	エラーなし	ー
20400	LU 分解された行列の対角要素にゼロがあります.	処理を打ち切ります.
20500	求めた解ベクトルに対する残差ベクトルのノルムが方程式 $\boldsymbol{Ax}=\boldsymbol{b}$ の右辺ベクトルのノルムの EPSR 倍より大きい. 係数行列は特異に近い可能性があります.	
30000	N<1,NZ<0,NFCNZ(N+1)≠NZ+1, NSIZEFACTORL<1, NSIZEFACTORU<1, NSIZEINDEXL<1, NSIZEINDEXU < 1, IREFINE=1 のとき ITERMAX<1	
30100	NPERM に指定した置換行列が正しくない.	
30200	NROW(j)に格納された i 列目の列指標 k が k<1 または k>N	
30300	i 列目の行指標の数 NFCNZ(i+1)-NFCNZ(i)>n	

(3)　使用上の注意

a.　注意

①　DM_VSCLUでLU分解を行った結果を利用します.
DM_VSCLUの(3)使用上の注意a. 注意⑤参照やDM_VSCLUXの使用例を参照願います.

②　直交行列である置換行列$\boldsymbol{P}$の要素$\boldsymbol{p_{ij}=1}$のとき, NPERM(i)=jと表現します.
逆は以下のようにして求めることができます.

```
DO i=1,n
  j=NPERM(i)
  NPERMINV(j)=i
  ENDDO
```

③　列番号に対応するノードを考えます. これをpost orderで順番を並び換えたときの番号の対応関係がNPOSTOに格納されています. post orderのi番目のノードが並び換える前の何番目のノードに対応するかを表します. j=nposto(i)はj番目であることを表します.
②同様にこれは直交行列である置換行列$\boldsymbol{Q}$を表し,行列$\boldsymbol{A}$を$\boldsymbol{QAQ^T}$と並び換えることに相当します.
逆変換$\boldsymbol{Q^T}$は以下のようにして求めることができます.

```
 DO i=1,n
  j=NPOSTO(i)
  NPOSTOINV(j)=i
  ENDDO
```

b. 使用例

連立 1 次方程式 $\boldsymbol{Ax} = \boldsymbol{f}$ を解きます.

行列は境界条件として立方体の境界で 0 となる楕円型の偏微分方程式に有限差分法を適用して生成されたものを利用します.

$$-\Delta u + a\nabla u + cu = f$$

ここで $a = (a_1, a_2, a_3)$.

行列はサブルーチン init_mat_diag によって生成されます.

対角形式格納法で生成し, 対角ベクトルの下 6 本を圧縮列格納法で格納します.

結果生成された非対称な行列 $\boldsymbol{A}$ を複素スパース行列に格納して連立 1 次方程式を解きます.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT  REAL*8 (A-H,O-Z)
      PARAMETER (NORD=40,KX = NORD,KY =NORD ,KZ = NORD,
     $      N = KX*KY*KZ)
      PARAMETER (NBORDER=N+1,NOFFDIAG=6)
      PARAMETER (K = N+1)
      PARAMETER (NDIAG = 7)
      INTEGER*4 WL,ZWL
      PARAMETER (NALL=NDIAG*N,
C
     $   ZWL =2*NALL,
     $   WL  =4*NALL+6*N,
     $   IW1L=2*NALL+2*(N+1)+16*N,
     $   IW2L=47*N+47+4*(N+1)+NALL+2*(NALL+N))
C
      DIMENSION NOFST(NDIAG)
      DIMENSION DIAG(K,NDIAG),DIAG2(K,NDIAG)
      COMPLEX*16 ZA(K*NDIAG),ZWC(K*NDIAG),
     $           ZW(ZWL),ZONE
      PARAMETER(ZONE=(1.0D0,0.0D0))
      DIMENSION NROW(K*NDIAG),NFCNZ(N+1),
     $          NROWSYM(K*NDIAG+N),NFCNZSYM(N+1),
     $          IWC(2,K*NDIAG)
      DIMENSION NPERM(N),W(WL),
     $          NPOSTO(N),NDIM(3,N),
     $          NASSIGN(N),
     $          MZ(N),
     $          IW1(IW1L),IW2(IW2L)
```

```
      COMPLEX*16, DIMENSION(:), ALLOCATABLE ::
     $        ZPANELFACTORL,ZPANELFACTORU
      INTEGER*4, DIMENSION(:), ALLOCATABLE :: NPANELINDEXL,NPANELINDEXU
      COMPLEX*8 ZDUMMYFL,ZDUMMYFU
      INTEGER*4 NDUMMYIL,
     $          NDUMMYIU
      INTEGER*8 NSIZEFACTORL,
     $          NSIZEINDEXL,
     $          NSIZEINDEXU,
     $          NSIZEFACTORU,
     $          NFCNZFACTORL(N+1),
     $          NFCNZFACTORU(N+1),
     $          NFCNZINDEXL(N+1),
     $          NFCNZINDEXU(N+1)
      COMPLEX*16 ZB(N),ZSOLEX(N)
      REAL*8 THEPSZ,EPSZ,SPEPSZ,
     $       SCLROW(N),SCLCOL(N)
C
      INTEGER*4     IPIVOT,ISTATIC,NFCNZPIVOT(N+1),
     $              NPIVOTP(N),NPIVOTQ(N),
     $              IREFINE,ITERMAX,ITER,IPLEDSM
C
      PRINT *,'    LU DECOMPOSITION METHOD'
      PRINT *,'    FOR SPARSE UNSYMMETRIC COMPLEX MATRICES'
      PRINT *,'    IN COMPRESSED COLUMN STORAGE'
      PRINT *
C
      DO I=1,N
      ZSOLEX(I)= ZONE
      ENDDO
      PRINT *,'    EXPECTED SOLUTIONS'
      PRINT *,'    X(1) = ',ZSOLEX(1),' X(N) = ',ZSOLEX(N)
      PRINT *
C
      VA1 = 1.0D0
      VA2 = 2.0D0
      VA3 = 3.0D0
      VC =  4.0D0
      XL = 1.0
      YL = 1.0
      ZL = 1.0
      CALL INIT_MAT_DIAG(VA1,VA2,VA3,VC,DIAG,NOFST
```

```
     &          ,KX,KY,KZ,XL,YL,ZL,NDIAG,N,K)
C
      DIAG2=0
C
      DO I=1,NDIAG
C
      IF(NOFST(I).LT.0)THEN
      NBASE=-NOFST(I)
      LENGTH=N-NBASE
      DIAG2(1:LENGTH,I)=DIAG(NBASE+1:N,I)
      ELSE
      NBASE=NOFST(I)
      LENGTH=N-NBASE
      DIAG2(NBASE+1:N,I)=DIAG(1:LENGTH,I)
      ENDIF
C
      ENDDO
C
      NUMNZ=1
C
      DO J=1,N
      NTOPCFG=1
C
      DO I=NDIAG,1,-1
C
      IF(NTOPCFG.EQ.1)THEN
      NFCNZ(J)=NUMNZ
      NTOPCFG=0
      ENDIF
C
      IF(J.LT.NBORDER.AND.I.GT.NOFFDIAG)THEN
      CONTINUE
      ELSE
C
      IF(DIAG2(J,I).NE.0.0D0)THEN
C
      NCOL=J-NOFST(I)
      ZA(NUMNZ)=DCMPLX(DIAG2(J,I),0.0D0)
      NROW(NUMNZ)=NCOL
C
      NUMNZ=NUMNZ+1
C
```

```
      ENDIF
      ENDIF
      ENDDO
      ENDDO
C
      NFCNZ(N+1)=NUMNZ
      NZ=NUMNZ-1
C
      CALL DM_VMVSCCC(ZA,NZ,NROW,NFCNZ,N,ZSOLEX,
     $             ZB,ZWC,IWC,ICON)
C
C     INITIAL CALL WITH IORDER=1
C
      IORDERING= 0
      IPLEDSM=1
      ISCLITERMAX=10
      ISW=1
      NSIZEFACTORL=1
      NSIZEFACTORU=1
      NSIZEINDEXL=1
      NSIZEINDEXU=1
      EPSZ=1.0D-16
      THEPSZ=1.0D-2
      SPEPSZ=0.0D0
      IPIVOT=40
      ISTATIC=0
      IREFINE=1
      EPSR=0.0D0
      ITERMAX=10
C
      CALL DM_VSCLU(ZA,NZ,NROW,NFCNZ,N,
     $              IPLEDSM,MZ,ISCLITERMAX,IORDERING,
     $              NPERM,ISW,
     $              NROWSYM,NFCNZSYM,
     $              NASSIGN,
     $              NSUPNUM,
     $              NFCNZFACTORL,ZDUMMYFL,
     $              NSIZEFACTORL,
     $              NFCNZINDEXL,
     $              NDUMMYIL,NSIZEINDEXL,
     $              NDIM,
     $              NFCNZFACTORU,ZDUMMYFU,
```

```
     $              NSIZEFACTORU,
     $              NFCNZINDEXU,
     $              NDUMMYIU,NSIZEINDEXU,
     $              NPOSTO,
     $              SCLROW,SCLCOL,
     $              EPSZ,THEPSZ,
     $              IPIVOT,ISTATIC,SPEPSZ,NFCNZPIVOT,
     $              NPIVOTP,NPIVOTQ,
     $              ZW,W,IW1,IW2,ICON)
C
      PRINT*,'ICON=',ICON,' NSIZEFACTORL=',NSIZEFACTORL,
     $       ' NSIZEFACTORU=',NSIZEFACTORU,
     $       'NSIZEINDEXL=',NSIZEINDEXL,
     $       'NSIZEINDEXU=',NSIZEINDEXU,
     $       'NSUPNUM=',NSUPNUM
C
      ALLOCATE( ZPANELFACTORL(NSIZEFACTORL) )
      ALLOCATE( ZPANELFACTORU(NSIZEFACTORU) )
      ALLOCATE( NPANELINDEXL(NSIZEINDEXL) )
      ALLOCATE( NPANELINDEXU(NSIZEINDEXU) )
C
      ISW=2
C
      CALL DM_VSCLU(ZA,NZ,NROW,NFCNZ,N,
     $              IPLEDSM,MZ,ISCLITERMAX,IORDERING,
     $              NPERM,ISW,
     $              NROWSYM,NFCNZSYM,
     $              NASSIGN,
     $              NSUPNUM,
     $              NFCNZFACTORL,ZPANELFACTORL,
     $              NSIZEFACTORL,
     $              NFCNZINDEXL,
     $              NPANELINDEXL,NSIZEINDEXL,
     $              NDIM,
     $              NFCNZFACTORU,ZPANELFACTORU,
     $              NSIZEFACTORU,
     $              NFCNZINDEXU,
     $              NPANELINDEXU,NSIZEINDEXU,
     $              NPOSTO,
     $              SCLROW,SCLCOL,
     $              EPSZ,THEPSZ,
     $              IPIVOT,ISTATIC,SPEPSZ,NFCNZPIVOT,
```

```
     $              NPIVOTP,NPIVOTQ,
     $              ZW,W,IW1,IW2,ICON)
C
      CALL DM_VSCLUX(N,
     $              IORDERING,
     $              NPERM,
     $              ZB,
     $              NASSIGN,
     $              NSUPNUM,
     $              NFCNZFACTORL,ZPANELFACTORL,
     $              NSIZEFACTORL,
     $              NFCNZINDEXL,
     $              NPANELINDEXL,NSIZEINDEXL,
     $              NDIM,
     $              NFCNZFACTORU,ZPANELFACTORU,
     $              NSIZEFACTORU,
     $              NFCNZINDEXU,
     $              NPANELINDEXU,NSIZEINDEXU,
     $              NPOSTO,
     $              IPLEDSM,MZ,
     $              SCLROW,SCLCOL,
     $              NFCNZPIVOT,
     $              NPIVOTP,NPIVOTQ,
     $              IREFINE,EPSR,ITERMAX,ITER,
     $              ZA,NZ,NROW,NFCNZ,
     $              IW2,ICON)
C
      ERR = ERRNRM(ZSOLEX,ZB,N)

      PRINT *,'    COMPUTED VALUES'
      PRINT *,'    X(1) = ',ZB(1),' X(N) = ',ZB(N)
      PRINT *
      PRINT *,'    ICON = ',ICON
      PRINT *
      PRINT *,'    N = ',N
      PRINT *
      PRINT *,'    ERROR = ',ERR
      PRINT *,'    ITER=',ITER
      PRINT *
      PRINT *

      IF(ERR.LT.1.0D-8.AND.ICON.EQ.0)THEN
```

```
         WRITE(*,*)'********** OK **********'
       ELSE
         WRITE(*,*)'********** NG **********'
       ENDIF
C
       DEALLOCATE( ZPANELFACTORL,ZPANELFACTORU,
      $            NPANELINDEXL,
      $            NPANELINDEXU )

       STOP
       END

C =======================================
C     INITIALIZE COEFFICIENT MATRIX
C =======================================
       SUBROUTINE INIT_MAT_DIAG(VA1,VA2,VA3,VC,D_L,OFFSET
      &                ,NX,NY,NZ,XL,YL,ZL,NDIAG,LEN,NDIVP)
       IMPLICIT  REAL*8(A-H,O-Z)
       DIMENSION D_L(NDIVP,NDIAG)
       INTEGER   OFFSET(NDIAG)
C
       IF (NDIAG .LT. 1) THEN
         WRITE (*,*) 'SUBROUTINE INIT_MAT_DIAG:'
         WRITE (*,*) ' NDIAG SHOULD BE GREATER THAN OR EQUAL TO 1'
         RETURN
       ENDIF

!$OMP PARALLEL DEFAULT(PRIVATE)
!$OMP+     SHARED(VA1,VA2,VA3,VC,D_L,OFFSET
!$OMP+      ,NX,NY,NZ,XL,YL,ZL,NDIAG,LEN,NDIVP)

C NDIAG CANNOT BE GREATER THAN 7
       NDIAG_LOC = NDIAG
       IF (NDIAG .GT. 7) NDIAG_LOC = 7

C INITIAL SETTING
       HX = XL/(NX+1)
       HY = YL/(NY+1)
       HZ = ZL/(NZ+1)

!$OMP DO
       DO I = 1,NDIVP
```

```
      DO J = 1,NDIAG
      D_L(I,J) = 0.0
      ENDDO
      ENDDO
!$OMP ENDDO

      NXY = NX*NY

C OFFSET SETTING
!$OMP SINGLE
      L = 1
      IF (NDIAG_LOC .GE. 7) THEN
        OFFSET(L) = -NXY
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 5) THEN
        OFFSET(L) = -NX
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 3) THEN
        OFFSET(L) = -1
        L = L+1
      ENDIF
      OFFSET(L) = 0
      L = L+1
      IF (NDIAG_LOC .GE. 2) THEN
        OFFSET(L) = 1
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 4) THEN
        OFFSET(L) = NX
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 6) THEN
        OFFSET(L) = NXY
      ENDIF
!$OMP END SINGLE

C MAIN LOOP
!$OMP DO
      DO 100 J = 1,LEN
        JS = J
```

```
C DECOMPOSE JS-1 = (K0-1)*NX*NY+(J0-1)*NX+I0-1
        K0 = (JS-1)/NXY+1
        IF (K0 .GT. NZ) THEN
          PRINT*,'ERROR; K0.GH.NZ '
          GOTO 100
        ENDIF
        J0 = (JS-1-NXY*(K0-1))/NX+1
        I0 = JS - NXY*(K0-1) - NX*(J0-1)
        L = 1

        IF (NDIAG_LOC .GE. 7) THEN
          IF (K0 .GT. 1) D_L(J,L) = -(1.0/HZ+0.5*VA3)/HZ
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 5) THEN
          IF (J0 .GT. 1) D_L(J,L) = -(1.0/HY+0.5*VA2)/HY
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 3) THEN
          IF (I0 .GT. 1) D_L(J,L) = -(1.0/HX+0.5*VA1)/HX
          L = L+1
        ENDIF
        D_L(J,L) = 2.0/HX**2+VC
        IF (NDIAG_LOC .GE. 5) THEN
          D_L(J,L) = D_L(J,L) + 2.0/HY**2
          IF (NDIAG_LOC .GE. 7) THEN
            D_L(J,L) = D_L(J,L) + 2.0/HZ**2
          ENDIF
        ENDIF
        L = L+1
        IF (NDIAG_LOC .GE. 2) THEN
          IF (I0 .LT. NX) D_L(J,L) = -(1.0/HX-0.5*VA1)/HX
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 4) THEN
          IF (J0 .LT. NY) D_L(J,L) = -(1.0/HY-0.5*VA2)/HY
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 6) THEN
          IF (K0 .LT. NZ) D_L(J,L) = -(1.0/HZ-0.5*VA3)/HZ
        ENDIF
```

```
 100  CONTINUE
!$OMP ENDDO

!$OMP END PARALLEL

      RETURN
      END
C =====================================
* SOLUTE ERROR
* | Z1 - Z2 |
C =====================================
      REAL*8 FUNCTION ERRNRM(Z1,Z2,LEN)
      IMPLICIT  REAL*8 (A-H,O-Z)
      COMPLEX*16 Z1(*),Z2(*),SS
C
      S = 0D0
      DO 100 I = 1,LEN
        SS  = Z1(I) - Z2(I)
        S = S + DREAL(SS * DCONJG(SS))
 100  CONTINUE
C
      ERRNRM = SQRT( S )
      RETURN
      END
```

DM_VSCS

複素スパース行列の連立 1 次方程式 (LU 分解法)
CALL DM_VSCS(ZA, NZ, NROW, NFCNZ, N, IPLEDSM, MZ, ISCLITERMAX, IORDERING, NPERM, ISW, NROWSYM, NFCNZSYM, ZB, NASSIGN, NSUPNUM, NFCNZFACTORL, ZPANELFACTORL, NSIZEFACTORL, NFCNZINDEXL, NPANELINDEXL, NSIZEINDEXL, NDIM, NFCNZFACTORU, ZPANELFACTORU, NSIZEFACTORU, NFCNZINDEXU, NPANELINDEXU, NSIZEINDEXU, NPOSTO, SCLROW, SCLCOL, EPSZ, THEPSZ, IPIVOT, ISTATIC, SPEPSZ, NFCNZPIVOT, NPIVOTP, NPIVOTQ, IREFINE, EPSR, ITERMAX, ITER, ZW, W, IW1, IW2, ICON)

(1) 機能

$n \times n$ の複素スパース行列 $\boldsymbol{A}$ に対角要素に大きな要素を並べる並び換えを行い, 行および列の均衡化を行うスケーリングを行います. スーパーノードの対角ブロック内で指定した Pivot をとり, LU 分解し解きます.

要素の並び換えや スケーリングおよび Pivot では, 複素数の大きさを表す絶対値は, 実部の絶対値と虚部の絶対値を加えたもので近似します.

$$\boldsymbol{Ax=b}$$

複素スパース行列 $\boldsymbol{A}$ は以下のように変換できます.

$$\boldsymbol{A_1=D_rAP_cD_c}$$

ここで $\boldsymbol{P_c}$ は列の入れ換えを行う直交行列, $\boldsymbol{D_r}$ は行のスケーリングを行う対角行列, $\boldsymbol{D_c}$ は列のスケーリングを行う対角行列です.

$$\boldsymbol{A_2=QPA_1P^TQ^T}$$

$\boldsymbol{A_2}$ は, スーパーノードの対角ブロックで指定された pivot を行い,行および列の入れ換えを行い LU 分解されます.

ただし, $\boldsymbol{P}$ は,$\boldsymbol{SYM}= \boldsymbol{A_1}+\boldsymbol{A_1^T}$ の非ゼロパターンに対して求めた ordering による行列要素の並び換えを示す置換行列で, $\boldsymbol{Q}$ は $\boldsymbol{SYM}$ に対する post order による行列要素の並び換えを示す置換行列を示します. $\boldsymbol{P}$, $\boldsymbol{Q}$ は直交行列です.

$\boldsymbol{L}$ は下三角行列, $\boldsymbol{U}$ は単位上三角行列です.

Pivot 処理で, 閾値 THEPSZ より絶対値が大きな Pivot を見つけることが出来なかったとき, 対角ブロック内の絶対値が最大の要素を Pivot の候補とします. この値が小さ過ぎるときに Static Pivot を与えて近似的に LU 分解を続けることができます.

そして LU 分解の結果を利用して, 解を求めます.

また, 解の反復改良で精度を改良することを指定できます.

(2)　パラメタ

ZA ..... 入力. 複素スパースな係数行列 $\boldsymbol{A}$ を圧縮列格納法で ZA(1:NZ)に格納します.
ZA(NZ)なる倍精度複素数型の 1 次元配列.
圧縮列格納法については,実スパース行列と実ベクトル(圧縮列格納法), DM_VMVSCC の図 DM_VMVSCC-1 を参照してください. 実数型の配列 CC を複素数型の配列に変えたものを利用します.

NZ ..... 入力. 複素スパースな係数行列 $\boldsymbol{A}$ の非零要素の総数.

NROW..... 入力. 圧縮列格納法で使用される行指標で ZA に格納される要素が何番目の行ベクトルに属するかを示します.
NROW(NZ)なる 1 次元配列.

NFCNZ..... 入力. 行列 $\boldsymbol{A}$ の各列の非零要素を列方向に圧縮して順次配列 A に格納するとき,対応する列の最初の非零要素が格納される位置を表します.
NFCNZ(N+1)=NZ+1.
NFCNZ(N+1)なる 1 次元配列.

N..... 入力. 行列 $\boldsymbol{A}$ の次数 n.

IPLEDSM..... 入力. 対角要素に大きな要素を並べる列の入れ換えを行うかを指定します.
1 のとき, 列の入れ換えを求めて入れ換えます.
1 以外のとき, 列の入れ換えを行いません.

MZ..... 出力. IPLEDSM に 1 が指定されたとき, 列の入れ換えを表す. MZ(i)=j は行列 $\boldsymbol{a_{ij}}$ の属する j 番目の列を i 番目の列に移動する. $\boldsymbol{a_{ij}}$ は対角要素に並べる大きな要素を表す.
MZ(N)なる 1 次元配列.

ISCLITERMAX.... 入力. 係数行列のスケーリングを行う対角行列 $\boldsymbol{D_r}$ と $\boldsymbol{D_c}$ を反復して求める反復回数.
ISCLITERMAX ≤ 0 のときスケーリングは行わすに, $\boldsymbol{D_r}$ および $\boldsymbol{D_c}$ は単位行列が設定されます.
ISCLITERMAX ≥ 10 のときは 10 回を上限値とします.

IORDERING..... 入力. ordering を表す置換行列 $\boldsymbol{P}$ で $\boldsymbol{PA_1P^T}$ と変換した行列を LU 分解するかを指定します. 置換行列は直交行列です.
10 のとき, ISW=1 で呼び出すと ordering を求める $\boldsymbol{A_1}$ に関する情報を求めて NROWSYM, NFCNZSYM に設定します.
11 のとき, ISW=1 で 10 を指定して呼び出して得た行列を対称化した情報 NROWSYM, NFCNZSYM を使って決めた ordering を NPERM に指定して同じく ISW=1 で呼び出し, 計算を続けることを示します. $\boldsymbol{PA_1P^T}$ を LU 分解する処理を続けます.
10,11 以外のとき, ordering は指定せずに行列 $\boldsymbol{A_1}$ をそのまま LU 分解します.
出力. ISW=1 と IORDERING=10 を指定して呼び出した後, IORDERING に 11 が設定されます. そのため, ordering を NPERM に指定して再び呼び出すときに特に 11 を指定する必要はありません.

((3)使用上の注意 a. 注意①参照).

NPERM ..... 入力. IORDERING=11 のとき使用する置換行列をベクトルで指定します.

NPERM(N)なる 1 次元配列.

((3)使用上の注意 a. 注意①参照).

ISW ..... 入力. 呼び出しに関する制御情報を示します.

1) 1 のとき

対称化および symbolic decomposition を行い, 必要な領域が割り当てられているか調べ計算を行います.

IORDERING=10 で呼び出し, ordering を求めるための情報が NROWSYM, NFCNZSYM に出力されます. これらを使って ***SYM*** に対する ordering を求めた後, NPERM に指定して IORDERING=11 を指定してもう 1 回 ISW=1 で呼び出します.

IORDERING が 10,11 以外のときは ordering は行ないません.

2) 2 のとき

ISW=1 で呼び出したとき, ZPANELFACTORL,

ZPANELFACTORU, NPANELINDEXL または NPANELINDEXU の大きさが不足して ICON=31000 で終了した処理を継続します. このとき, NSIZEFACTORL , NSIZEFACTORU, NSIZEINDEXL または NSIZEINDEXU に返却された必要な大きさで ZPANELFACTORL,

ZPANELFACTORU, NPANELINDEXL または NPANELINDEXU を確保し直して指定しなおして再度呼び出します.

これらの引数と実行を制御する引数 ISW 以外の引数に格納されている値は変更してはいけません.

3) 3 のとき

先立つ呼び出しで LU 分解された同じ係数行列に対する連立 1 次方程式の右辺ベクトル ***b*** を変えて再度解くことを指定します. このとき先立つ呼び出しで LU 分解した結果を使います.

実行を制御する引数 ISW と右辺ベクトル ***b*** を格納する引数 B 以外の引数に格納されている値を変更していけません.

NROWSYM ..... 出力. IORDERING=10 で呼び出したとき, 対称化した $\boldsymbol{SYM}=\boldsymbol{A_1}+\boldsymbol{A_1^T}$ の非ゼロパターンの下三角行列部分の行指標を列圧縮したものが返却されます.

IPLEDSM=1 のときは, $\boldsymbol{A_1}=\boldsymbol{D_r A P_c D_c}$ です.

NROWSYM(NZ+N)なる 1 次元配列.

NFCNZSYM ..... 出力. IORDERING=10 で呼び出したとき, 行列 SYM の下三角部分の各列の非零要素の行指標を列方向に圧縮して順次配列 NROWSYM に格納するとき,対応する列の最初の非零要素が格納される位置を表します.

NFCNZSYM(N+1)=NSYMZ+1 NSYMZ は,総要素数を表します.

NFCNZSYM(N+1)なる 1 次元配列.

ZB..... 入力. ***Ax=b*** の右辺定数ベクトル.
出力. 解ベクトル.
ZB(N)なる倍精度複素数型の 1 次元配列.

NASSIGN..... 出力. 各 supernode に対する ***L***, ***U*** は圧縮して 2 次元の panel に格納します. この panel を ZPANELFACTORL および ZPANELFACTORU の 1 次元部分配列として順番に割り付けたとき何番目になるかを示します. これらの指標ベクトルも同じように NPANELINDEXL, NPANELINDEXU に割り付けます. j=NASSIGN(i)のとき, i 番目の supernode を j 番目に割り付けたことを示します.
入力. ISW≠1 のとき, 初回呼び出しの値を再利用します.
分解結果の格納方法については図 DM_VSCS -1 を参照してください.
NASSIGN(N)なる 1 次元配列.

NSUPNUM ..... 出力. supernode の総数.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します. (≤n)

NFCNZFACTORL 出力. 複素スパース行列の LU 分解の結果の行列 ***L*** および ***U*** はスーパーノードに対応しておのおの求めます. 各 supernode に対応する *L* の列ベクトルは, 共通の行指標ベクトルを持つように圧縮してブロック対角部分に ***U*** の対応部分を含めて 2 次元の panel に格納します. この panel を ZPANELFACTORL の 1 次元部分配列として順番に割り付けたとき, i 番目の panel の先頭要素 panel(1,1)が 1 次元配列の ZPANELFACTORL の何番目の要素になるかを示します.
NFCNZFACTORL(N+1)なる 8 バイト整数型の 1 次元配列.
結果の格納方法については図 DM_VSCS -1 を参照してください.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.

ZPANELFACTORL出力. 各 supernode の分解結果に対応する行列 ***L*** の複数の列ベクトルと行列 ***U*** のブロック対角部分について，共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します. panel を順番に割り付けます. i 番目の supernode に対応する panel が何番目の位置に割り当てられるかは j=NASSIGN(i)から分かります. その先頭位置は NFCNZFACTORL(j)に格納されています. panel ごとに分解結果が格納されます.
i 番目に割付けられた panel の大きさは DIM(1,i)×DIM(2,i)の 2 次元配列と見なせます. i 番目の panel の panel(s,t), $s \geq t$, s=1,…,DIM(1,i), t=1,…,DIM(2,i)に下三角行列 ***L*** が格納されます. panel の panel(s,t), $s < t$, t=1,…,DIM(2,i)には単位上三角行列 ***U*** の対角要素を除いたブロック対角部分が格納されます.
ZPANELFACTORL(NSIZEFACTORL)なる倍精度複素数型の 1 次元配列.
結果の格納方法については図 DM_VSCS -1 を参照してください.
((3)使用上の注意 a. 注意③参照).

NSIZEFACTORL . 入力. ZPANELFACTORL の大きさを示す. 8 バイト整数型.
出力. ZPANELFACTORL の大きさとして必要な大きさが返却されます.
((3)使用上の注意 a. 注意③参照).

NFCNZINDEXL ... 出力. 各 supernode の分解結果に対応する行列 ***L*** の複数の列ベクトルと行列 ***U*** のブロック対角部分について，共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します.この行指標ベクトルを NPANELINDEXL に順番に割り付けたとき i 番目の行指標ベクトルの先頭要素が 1 次元配列である NPANELINDEXL の何番目の要素になるかを示します.
NFCNZINDEXL(N+1)なる 8 バイト整数型の 1 次元配列.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
結果の格納方法については図 DM_VSCS-1 を参照してください.

NPANELINDEXL 出力. 各 supernode の分解結果に対応する行列 ***L*** の複数の列ベクトルと行列 ***U*** のブロック対角部分について，共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します. この行指標ベクトルを NPANELINDEXL に順番に割り付けます. i 番目の supernode に対応する panel の行の指標ベクトルが何番目に割り当てられるかは j=NASSIGN(i)から分かります. その先頭位置は NFCNZINDEXL(j)に格納されています.
この行指標は行列 ***SYM*** に対する post order で並び換えたときの行の番号です.
NPANELINDEXL(NSIZEINDEXL)なる 1 次元配列.
結果の格納方法については図 DM_VSCS-1 を参照してください.
((3)使用上の注意 a. 注意③参照).

NSIZEINDEXL ..... 入力. NPANELINDEXL の大きさを示す. 8 バイト整数型.
出力. 必要な大きさが返却されます.
((3)使用上の注意 a. 注意③参照).

NDIM ..... 出力. NDIM(1,i)および NDIM(2,i)は行列 ***L*** に関して i 番目に割り付けられた panel の 1 次元目と 2 次元目の大きさを示します.
NDIM(3,i)は行列 ***U*** に関して割り付けられた panel を転置したときの 1 次元目の大きさに対角ブロックの大きさを加えた大きさを示します.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
NDIM(3,N)なる 2 次元配列.
結果の格納方法については図 DM_VSCS-1 を参照してください.

NFCNZFACTORU 出力. 複素スパース行列の LU 分解の結果の行列 ***U*** に関しては, 各 supernode に対応する ***U*** の行ベクトルは, ブロック対角部分を除いて共通の列指標ベクトルを持つように圧縮し, 転置したものを 2 次元の panel に格納します. この panel を ZPANELFACTORU の 1 次元部分配列として順番に割り付けたとき, i 番目の panel の先頭要素 panel(1,1)が 1 次元配列の ZPANELFACTORU の何番目の要素になるかを示します.
NFCNZFACTORU(N+1)なる 8 バイト整数型の 1 次元配列.
結果の格納方法については図 DM_VSCS-1 を参照してください.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.

ZPANELFACTORU 出力. 各 supernode の分解結果に対応する行列 ***U*** のブロック対角部分を除いた複数の行ベクトルについて，共通の列指標ベクトルを持つよう

に圧縮し, 転置して 2 次元の panel に格納します. panel を順番に割り付けます. i 番目の supernode に対応する panel が何番目の位置に割り当てられるかは j=NASSIGN(i) から分かります. その先頭位置は NFCNZFACTORU(j)に格納されています. panel ごとに分解結果が格納されます.

i 番目に割付けられた panel の大きさは{DIM(3,i)-DIM(2,i)}×DIM(2,i)の 2 次元配列と見なせます. i 番目の panel の panel(s,t), s=1,…,DIM(3,i)-DIM(2,i), t=1,…,DIM(2,i)に単位上三角行列 $\boldsymbol{U}$ のブロック対角部分を除いた部分が行ベクトルを圧縮し, 転置して格納します.

ZPANELFACTORU(NSIZEFACTORU)なる倍精度複素数型の 1 次元配列.

結果の格納方法については図 DM_VSCS-1 を参照してください.

((3)使用上の注意 a. 注意③参照).

NSIZEFACTORU. 入力. ZPANELFACTORU の大きさを示す. 8 バイト整数型.

出力. ZPANELFACTORU の大きさとして必要な大きさが返却されます.

((3)使用上の注意 a. 注意③参照).

NFCNZINDEXU... 出力. 各 supernode の分解結果に対応する行列 $\boldsymbol{U}$ は対角ブロック部分を除いて複数の行ベクトルについて, 共通の列指標ベクトルを持つように圧縮し, 転置して 2 次元の panel に格納します. 対角ブロック部分も含んだこの列指標ベクトルを NPANELINDEXU に順番に割り付けたとき i 番目の列指標ベクトルの先頭要素が 1 次元配列である NPANELINDEXU の何番目の要素になるかを示します.

NFCNZINDEXU(N+1)なる 8 バイト整数型の 1 次元配列.

入力. ISW≠1 のとき,初回呼び出しの値を再利用します.

結果の格納方法については図 DM_VSCS-1 を参照してください.

NPANELINDEXU 出力. 各 supernode の分解結果に対応する行列 $\boldsymbol{U}$ の対角ブロックを除いた部分を転置して圧縮して 2 次元の panel に格納します. この列指標ベクトルを対角ブロック部分を含めて NPANELINDEXU に順番に割り付けます. i 番目の supernode に対応する panel の行の指標ベクトルが何番目に割り当てられるかは j=NASSIGN(i)から分かります. その先頭位置は NFCNZINDEXU(j)に格納されています.

この行指標は行列 $\boldsymbol{SYM}$ に対する post order で並び換えたときの列の番号です.

NPANELINDEXU(NSIZEINDEXU)なる 1 次元配列.

結果の格納方法については図 DM_VSCS-1 を参照してください.

((3)使用上の注意 a. 注意③参照).

NSIZEINDEXU .... 入力. NPANELINDEXU の大きさを示す. 8 バイト整数型.

出力. 必要な大きさが返却されます.

((3)使用上の注意 a. 注意③参照).

NPOSTO ..... 出力. post order で i 番目の node が行列 $\boldsymbol{A}$ の何番目の列番号に対応するかを表す 1 次元ベクトル.

入力. ISW≠1 のとき,初回呼び出しの値を再利用します.

NPOSTO(N)なる 1 次元配列.
((3)使用上の注意 a. 注意④参照).

SCLROW ..... 出力. スケーリング対角行列 $\boldsymbol{D}_r$. 対角要素が 1 次元配列に格納されます.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
SCLROW(N)なる 1 次元配列.

SCLCOL..... 出力. スケーリング対角行列 $\boldsymbol{D}_c$. 対角要素が 1 次元配列に格納されます.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
SCLCOL(N)なる 1 次元配列.

EPSZ ..... 入力. 分解過程で対角要素の大きさの絶対値の相対判定値
出力. EPSZ ≤ 0.0 のときは標準値が設定されます.
((3)使用上の注意 a. 注意②参照).

THEPSZ ..... 入力. ピボットの判定での閾値(threshold). これより大きな値はピボットとして採用します. 見つかればその時点で, ピボット検索は打ち切ります.
例えば 1.0D-2 程度.
出力. THEPSZ ≤ 0.0 のときは 1.0D-2 が設定されます.
EPSZ ≥ THEPSZ > 0.0 のときは, EPSZ が設定されます.

IPIVOT..... 入力. スーパーノード内でピボットを行うか, 行う場合どのようなピボットを行うかを指定します. 例えば, 完全ピボットとして 40 を指定.
IPIVOT < 10 : または IPIVOT ≥ 50: ピボットなし.
10 ≤ IPIVOT < 20 : 部分ピボット
20 ≤ IPIVOT < 30 : 対角ピボット
21 : スーパーノード内で対角ピボットがとれないとき, Rook ピボットに変更します.
22 : スーパーノード内で対角ピボットがとれないときは, Rook ピボットに, Rook ピボットが取れないときは完全ピボットに変更します.
30 ≤ IPIVOT < 40 :Rook ピボット
32 : スーパーノード内で Rook ピボットがとれないとき, 完全ピボットをとります.
40 ≤ IPIVOT < 50 : 完全ピボット

ISTATIC..... 入力. Pivot を指定したとき, Static Pivot を行うかを示します.
1) = 1 のとき
スーパーノード内で指定したピボットが SPEPSZ より大きくないとき, ピボットとして大きさ SPEPSZ の複素数で近似します. ピボットの値が 0.0d0 なら SPEPSZ に近似します.
このとき, 以下を設定しなければなりません.
a) EPSZ は EPSZ の標準値以下にしなければなりません.
b) ISCLITERMAX は 10 を設定しスケーリングを行う必要があります.
c)THEPSZ ≥ SPEPSZ でなければなりません.
d) 解の反復改良を行うため IREFINE=1 を指定しなければなりません.
2) ≠1 のとき
Static Pivot は行いません.

SPEPSZ ..... 入力. ISTATIC=1 のとき, Statc Pivot として使う値.
1.0D-8 ≥ SPEPSZ ≥ EPSZ でなければなりません.
出力. SPEPSZ<EPSZ のときは, 1.0D-10 が設定されます.

NFCNZPIVOT ..... 出力. スーパーノード内の相対的な位置でのピボットの行, 列の入れ換えの履歴を格納する位置を示す. i 番目の supernode に関する情報が何番目の位置に割り当てられるかは j=NASSIGN(i)から分かります. その先頭位置は NFCNZPIVOT(j)に格納されています. i 番目のスーパーノードの入れ換え情報は, NPIVOTP, NPIVOTQ の is=NFCNZPIVOT(j) ,…, ie=NFCNZPIVOT(j)+NDIM(2,j)-1 番目の要素に格納されます.
NFCNZPIVOT(NSUPNUM+1)なる 1 次元配列.

NPIVOTP ..... 出力. スーパーノード内の行の入れ換えに関する情報を格納する.
NPIVOTP(N)なる 1 次元配列.

NPIVOTQ ..... 出力. スーパーノード内の列の入れ換えに関する情報を格納する.
NPIVOTQ(N)なる 1 次元配列.

IREFINE..... 入力. LU 分解結果を利用して解を求めるときに, 解の反復改良を行うかどうかを示す. 残差ベクトルを計算するときに 4 倍精度で計算します.
=1 : 解の反復改良を行う. 反復改良して得られる残差ベクトル $\boldsymbol{r}_k$ の絶対値の差分がひとつ前の差分の 1/4 より大きくなるまで, 反復改良を行います.
≠1: 解の反復改良を行わない.
ISTATIC=1 のとき, IREFINE=1 でなければなりません.

EPSR ..... 入力. 解の残差ベクトル $\boldsymbol{b}$-$\boldsymbol{Ax}$ の絶対値が, $\boldsymbol{b}$ の絶対値に対して十分小さいかを判定する判定値.
EPSR ≤ 0.0 のとき 1.0D-6 が設定されます.

ITERMAX..... 入力. (≥ 1) 反復改良を行うときの最大反復回数.

ITER..... 出力. 反復改良を行った回数.

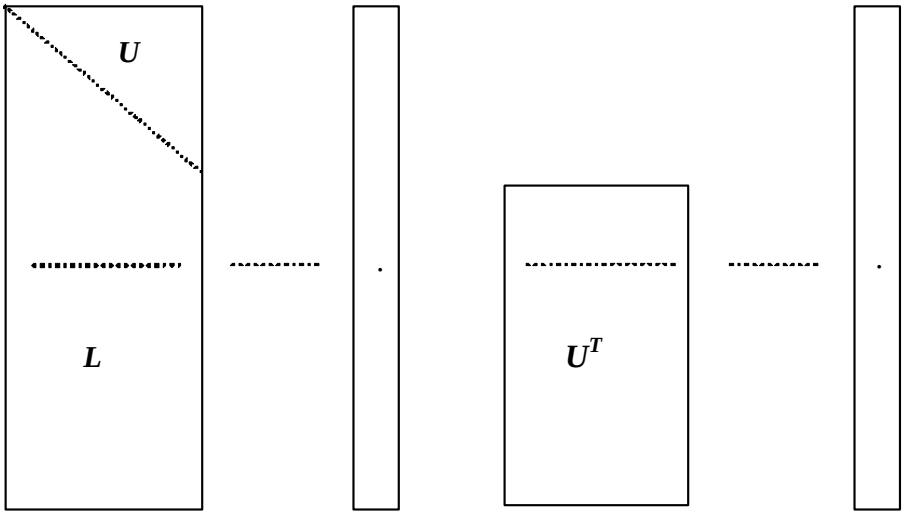
ZW..... 作業域.
出力/入力.
大きさ 2*NZ の倍精度複素数型の 1 次元配列.
ISW=1, 2 で続けて呼び出すとき,呼び出しの間で必要なデータを受け渡すために使われます. 呼び出しの間で値を変更してはいけません.

W..... 作業域.
出力/入力.
大きさ 4*NZ+6*N の 1 次元配列.
ISW=1, 2 で続けて呼び出すとき,呼び出しの間で必要なデータを受け渡すために使われます. 呼び出しの間で値を変更してはいけません.

IW1..... 作業域.
出力/入力.
大きさ 2*NZ+2*(N+1)+16*N の 1 次元配列.
ISW=1, 2 で続けて呼び出すとき,呼び出しの間で必要なデータを受け渡すために使われます. 呼び出しの間で値を変更してはいけません.

IW2..... 作業域.
出力/入力.
大きさ 47*N+47+NZ+4*(N+1)+2*(NZ+N)の 1 次元配列.
ISW=1, 2, 3 で続けて呼び出すとき,呼び出しの間で必要なデータを受け渡すために使われます. 呼び出しの間で値を変更してはいけません.

ICON..... 出力. コンディションコード.
表 DM_VSCS-1 参照



panel 行指標ベクトル Post order での行番号

panel 列指標ベクトル Post order での行番号

図 DM_VSCS-1 分解結果の格納概念図

j = NASSIGN(i) → i 番目の supernode は j 番目に格納されます.

p = NFCNZFACTORL(j) → j 番目の panel は ZPANELFACTORL の p 番目の要素から DIM(1,j)×DIM(2,j)の長さを占めます.

q = NFCNZINDEXL(j) → j 番目の panel の行指標を表すベクトルは NPANELINDEXL の q 番目の要素から DIM(1,j)の長さを占めます.

panel は大きさ DIM(1,j)×DIM(2,j)の配列と見なせます.

panel(s,t) , $s \geq t$, s = 1 , … , DIM(1,j), t=1 , … , DIM(2,j) に分解結果の下三角行列 $\boldsymbol{L}$ が格納されます.

panel(s,t) , $s < t$, s=1 , … , DIM(2,j), t=1 , … , DIM(2,j) に単位上三角行列 $\boldsymbol{U}$ の対角ブロック部分が対角要素を除き格納されます.

u = NFCNZFACTORU(j) → j 番目の panel は ZPANELFACTORU の u 番目の要素から (DIM(3,j)-DIM(2,j))×DIM(2,j)の長さを占めます.

v = NFCNZINDEXU(j) → j 番目の panel の列指標を表すベクトルは NPANELINDEXU の v 番目の要素から DIM(3,j)の長さを占めます.

panel は大きさ (DIM(3,j)-DIM(2,j))×DIM(2,j)の配列と見なせます.

panel(x,y) , x=1 , … , DIM(3,j)-DIM(2,j), y=1 , … , DIM(2,j)
に分解結果の単位上三角行列 $\boldsymbol{U}$ の転置行列 $\boldsymbol{U}^T$ の対角ブロック部分を除いた部分が

格納されます.
指標の値は行列 $\boldsymbol{A}$ の列番号をもつ node を post order で並び換えた $\boldsymbol{QAQ^T}$ で
の列番号を表します.

表 DM_VSCS-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし	ー
20000	ピボットが相対的に零になりました.	処理を打ち切ります.
20100	IPLEDSM=1 を指定して,対角要素に絶対値が大きな要素を並べるため maximum matching を求める処理で, 長さ N の maximum matching がみつかりませんでした.行列が特異である可能性があります.	
20200	行および列の均衡化を行う対角行列の対角要素を求める過程で,元の行列 A の行もしくは列にゼロベクトルがありました. 行列が特異である可能性があります.	
20400	LU 分解された行列の対角要素にゼロがあります.	
20500	求めた解ベクトルに対する残差ベクトルのノルムの大きさが方程式 Ax=b の右辺ベクトルのノルムの EPSR 倍より大きい. 係数行列は特異に近い可能性があります.	
30000	N<1,NZ<0,NFCNZ(N+1)≠NZ+1, NSIZEFACTORL<1, NSIZEFACTORU<1, NSIZEINDEXL<1, NSIZEINDEXU<1, ISW<1, ISW>3, IREFINE=1 のとき ITERMAX<1,	
30100	NPERM に指定した置換行列が正しくない.	
30200	NROW(j)に格納された i 列目の行指標 k が k<1 または k>N	
30300	i 列目の行指標の数 NFCNZ(i+1)-NFCNZ(i)>n	
30500	ISTATIC=1 のとき満たすべき条件をみたしていない. EPSZ は標準値 16$\boldsymbol{u}$ より大きかった. または ISCLITERMAX <10 であった. または IREFINE≠1 であった. または SPEPSZ>THEPSZ であった. または SPEPSZ>1.0D-8 であった.	

コード	意　　味	処理内容
31000	ZPANELFACTORL の大きさ NSIZEFACTORL または NPANELINDEXL の大きさ NSIZEINDEXL または ZPANELFACTORU の大きさ NSIZEFACTORU または NPANELINDEXU の大きさ NSIZEINDEXU が小さすぎます.	NSIZEFACTORL または NSIZEINDEXL または NSIZEFACTORU または NSIZEINDEXU で指定された 大きさで ZPANELFACTORL ま たは NPANELINDEXL または ZPANELFACTORU または NPANELINDEXU を割り付け て ISW=2 として再度呼び出す.

(3)　使用上の注意

a.　注意

①　直交行列である置換行列$\boldsymbol{P}$の要素$\boldsymbol{p_{ij}=1}$のとき, NPERM(i)=jと表現します.
逆は以下のようにして求めることができます.

```
DO i=1,n
  j=NPERM(i)
  NPERMINV(j)=i
  ENDDO
```

Fill-Reducing OrderingはMETISなどを使って求めることができます.
詳細は“付録1 参考文献一覧表”の[43],[44]を参照願います.

②　分解過程で対角要素の大きさの絶対値の相対零判定値にある値を設定したとすると, この値は次の意味を持っています. すなわち, LU分解の過程で, 対角要素の大きさの絶対値がその値より小さくなった場合に, その値を相対的に零と見なし, ICON=20000として処理を打ち切ります. EPSZの標準値は, 丸め誤差の単位をuとしたとき, EPSZ=16uです.
Pivotでは, 複素数の大きさを表す絶対値は, 実部の絶対値と虚部の絶対値を加えたもので近似します.
なお, 対角要素の大きさの絶対値が小さくなっても, 処理を続行させたい場合には, EPSZに極小の値を与えれば良いのですが, 結果の精度は保証されません.
Static Pivotを指定した場合, 対角要素がSPEPSZより小さいとき, 大きさSPEPSZの複素数で近似してLU分解を行います. このとき, 解の反復改良を指定しなければなりません.

③　分解結果を格納する配列ZPANELFACTORL, NPANELINDEXL, ZPANELFACTORU, NPANELINDEXUの必要な大きさは,事前には分かりません. 十分大きな配列を割り当てるか,本ルーチンを呼び出してsymbolic decompositionを行った解析の結果を使って割り当てを行うことができます.
例えば,大きさ1の1次元配列などを割付けます.そしてその大きさ1などの小さな値をNSIZEFACTORL, NSIZEINDEXL, NSIZEFACTORU, NSIZEINDEXUに指定して, ISW=1で呼び出します.
symbolic decompositionを行い, ICON=31000で終了し, NSIZEFACTORL, NSIZEINDEXL, NSIZEFACTORU, NSIZEINDEXUに必要な大きさが返却されま

す. 必要な大きさの配列を割付け直して引数に指定して, ISW=2で呼び出すことで, symbolic decomposition以降の処理を続けることができます. 使用例を参照願います.

④　列番号に対応するノードを考えます.これをpost orderで順番を並び換えたときの番号の対応関係がNPOSTOに格納されています. post orderのi番目のノードが並び換える前の何番目のノードに対応するかを表します. j=nposto(i)はj番目であることを表します.

①同様にこれは直交行列である置換行列$\boldsymbol{Q}$を表し,行列$\boldsymbol{A}$を$\boldsymbol{QAQ^T}$と並び換えることに相当します.

逆変換$\boldsymbol{Q^T}$は以下のようにして求めることができます.

```
DO i=1,n
   j=NPOSTO(i)
   NPOSTOINV(j)=i
   ENDDO
```

⑤　連立1次方程式$\boldsymbol{Ax}=\boldsymbol{b}$は, DM_VSCLUで複素スパース行列$\boldsymbol{A}$をLU分解して, 分解結果を利用して引き続いてDM_VSCLUXを呼び出して解くこともできます.

b.　使用例

連立 1 次方程式 $\boldsymbol{Ax} = \boldsymbol{f}$ を解きます.

行列は境界条件として立方体の境界で 0 となる楕円型の偏微分方程式に有限差分法を適用して生成されたものを利用します.

$$-\Delta u + a\nabla u + cu = f$$

ここで $a = (a_1, a_2, a_3)$.

行列はサブルーチン init_mat_diag によって生成されます.

対角形式格納法で生成し, 対角ベクトルの下 6 本を圧縮列格納法で格納します. 結果生成された非対称な行列 $\boldsymbol{A}$ を複素スパース行列に格納して連立 1 次方程式を解きます.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT  REAL*8 (A-H,O-Z)
      PARAMETER (NORD=40,KX = NORD,KY =NORD ,KZ = NORD,
     $      N = KX*KY*KZ)
      PARAMETER (NBORDER=N+1,NOFFDIAG=6)
      PARAMETER (K = N+1)
      PARAMETER (NDIAG = 7)
      INTEGER*4 WL,ZWL

      PARAMETER (NALL=NDIAG*N,
```

```
C
     $   ZWL =2*NALL,
     $   WL  =4*NALL+6*N,
     $   IW1L=2*NALL+2*(N+1)+16*N,
     $   IW2L=47*N+47+4*(N+1)+NALL+2*(NALL+N))
C
      DIMENSION NOFST(NDIAG)
      DIMENSION DIAG(K,NDIAG),DIAG2(K,NDIAG)
      COMPLEX*16 ZA(K*NDIAG),ZWC(K*NDIAG),
     $           ZW(ZWL),ZONE
      PARAMETER(ZONE=(1.0D0,0.0D0))
      DIMENSION NROW(K*NDIAG),NFCNZ(N+1),
     $          NROWSYM(K*NDIAG+N),NFCNZSYM(N+1),
     $          IWC(2,K*NDIAG)
      DIMENSION NPERM(N),W(WL),
     $          NPOSTO(N),NDIM(3,N),
     $          NASSIGN(N),
     $          MZ(N),
     $          IW1(IW1L),IW2(IW2L)
      COMPLEX*16, DIMENSION(:), ALLOCATABLE ::
     $        ZPANELFACTORL,ZPANELFACTORU
      INTEGER*4, DIMENSION(:), ALLOCATABLE :: NPANELINDEXL,NPANELINDEXU
      COMPLEX*16 ZDUMMYFL,ZDUMMYFU
      INTEGER*4 NDUMMYIL,
     $          NDUMMYIU
      INTEGER*8 NSIZEFACTORL,
     $          NSIZEINDEXL,
     $          NSIZEINDEXU,
     $          NSIZEFACTORU,
     $          NFCNZFACTORL(N+1),
     $          NFCNZFACTORU(N+1),
     $          NFCNZINDEXL(N+1),
     $          NFCNZINDEXU(N+1)
      COMPLEX*16 ZB(N),ZSOLEX(N)
      REAL*8 EPSZ,THEPSZ,SPEPSZ,
     $       SCLROW(N),SCLCOL(N)
C
      INTEGER*4     IPIVOT,ISTATIC,NFCNZPIVOT(N+1),
     $              NPIVOTP(N),NPIVOTQ(N),
     $              IREFINE,ITERMAX,ITER,IPLEDSM
C
      PRINT *,'    LU DECOMPOSITION METHOD'
```

```
      PRINT *,'    FOR SPARSE UNSYMMETRIC COMPLEX MATRICES'
      PRINT *,'    IN COMPRESSED COLUMN STORAGE'
      PRINT *
C
      DO I=1,N
      ZSOLEX(I)=ZONE
      ENDDO
      PRINT *,'    EXPECTED SOLUTIONS'
      PRINT *,'    X(1) = ',ZSOLEX(1),' X(N) = ',ZSOLEX(N)
      PRINT *
C
      VA1 = 1.0D0
      VA2 = 2.0D0
      VA3 = 3.0D0
      VC =  4.0D0
      XL = 1.0
      YL = 1.0
      ZL = 1.0
      CALL INIT_MAT_DIAG(VA1,VA2,VA3,VC,DIAG,NOFST
     &          ,KX,KY,KZ,XL,YL,ZL,NDIAG,N,K)
C
      DIAG2=0
C
      DO I=1,NDIAG
C
      IF(NOFST(I).LT.0)THEN
      NBASE=-NOFST(I)
      LENGTH=N-NBASE
      DIAG2(1:LENGTH,I)=DIAG(NBASE+1:N,I)
      ELSE
      NBASE=NOFST(I)
      LENGTH=N-NBASE
      DIAG2(NBASE+1:N,I)=DIAG(1:LENGTH,I)
      ENDIF
C
      ENDDO
C
      NUMNZ=1
C
      DO J=1,N
      NTOPCFG=1
C
```

```
      DO I=NDIAG,1,-1
C
      IF(NTOPCFG.EQ.1)THEN
      NFCNZ(J)=NUMNZ
      NTOPCFG=0
      ENDIF
C
      IF(J.LT.NBORDER.AND.I.GT.NOFFDIAG)THEN
      CONTINUE
      ELSE
C
      IF(DIAG2(J,I).NE.0.0D0)THEN
C
      NCOL=J-NOFST(I)
      ZA(NUMNZ)=DCMPLX(DIAG2(J,I),0.0D0)
      NROW(NUMNZ)=NCOL
C
      NUMNZ=NUMNZ+1
C
      ENDIF
      ENDIF
      ENDDO
      ENDDO
C
      NFCNZ(N+1)=NUMNZ
      NZ=NUMNZ-1
C
      CALL DM_VMVSCCC(ZA,NZ,NROW,NFCNZ,N,ZSOLEX,
     $             ZB,ZWC,IWC,ICON)
C
C     INITIAL CALL WITH IORDER=1
C
      IORDERING= 0
      IPLEDSM=1
      ISCLITERMAX=10
      ISW=1
      EPSZ=1.0D-16
      NSIZEFACTORL=1
      NSIZEFACTORU=1
      NSIZEINDEXL=1
      NSIZEINDEXU=1
      THEPSZ=1.0D-2
```

```
      SPEPSZ=0.0D0
      IPIVOT=40
      ISTATIC=0
      IREFINE=1
      EPSR=0.0D0
      ITERMAX=10
C
      CALL DM_VSCS(ZA,NZ,NROW,NFCNZ,N,
     $              IPLEDSM,MZ,ISCLITERMAX,IORDERING,
     $              NPERM,ISW,
     $              NROWSYM,NFCNZSYM,
     $              ZB,
     $              NASSIGN,
     $              NSUPNUM,
     $              NFCNZFACTORL,ZDUMMYFL,
     $              NSIZEFACTORL,
     $              NFCNZINDEXL,
     $              NDUMMYIL,NSIZEINDEXL,
     $              NDIM,
     $              NFCNZFACTORU,ZDUMMYFU,
     $              NSIZEFACTORU,
     $              NFCNZINDEXU,
     $              NDUMMYIU,NSIZEINDEXU,
     $              NPOSTO,
     $              SCLROW,SCLCOL,
     $              EPSZ,THEPSZ,
     $              IPIVOT,ISTATIC,SPEPSZ,NFCNZPIVOT,
     $              NPIVOTP,NPIVOTQ,
     $              IREFINE,EPSR,ITERMAX,ITER,
     $              ZW,W,IW1,IW2,ICON)
C
      PRINT*,'ICON=',ICON,' NSIZEFACTORL=',NSIZEFACTORL,
     $       ' NSIZEFACTORU=',NSIZEFACTORU,
     $       'NSIZEINDEXL=',NSIZEINDEXL,
     $       'NSIZEINDEXU=',NSIZEINDEXU,
     $       'NSUPNUM=',NSUPNUM
C
      ALLOCATE( ZPANELFACTORL(NSIZEFACTORL) )
      ALLOCATE( ZPANELFACTORU(NSIZEFACTORU) )
      ALLOCATE( NPANELINDEXL(NSIZEINDEXL) )
      ALLOCATE( NPANELINDEXU(NSIZEINDEXU) )
C
```

```
      ISW=2
C
      CALL DM_VSCS(ZA,NZ,NROW,NFCNZ,N,
     $              IPLEDSM,MZ,ISCLITERMAX,IORDERING,
     $              NPERM,ISW,
     $              NROWSYM,NFCNZSYM,
     $              ZB,
     $              NASSIGN,
     $              NSUPNUM,
     $              NFCNZFACTORL,ZPANELFACTORL,
     $              NSIZEFACTORL,
     $              NFCNZINDEXL,
     $              NPANELINDEXL,NSIZEINDEXL,
     $              NDIM,
     $              NFCNZFACTORU,ZPANELFACTORU,
     $              NSIZEFACTORU,
     $              NFCNZINDEXU,
     $              NPANELINDEXU,NSIZEINDEXU,
     $              NPOSTO,
     $              SCLROW,SCLCOL,
     $              EPSZ,THEPSZ,
     $              IPIVOT,ISTATIC,SPEPSZ,NFCNZPIVOT,
     $              NPIVOTP,NPIVOTQ,
     $              IREFINE,EPSR,ITERMAX,ITER,
     $              ZW,W,IW1,IW2,ICON)
C
      ERR = ERRNRM(ZSOLEX,ZB,N)
C
      PRINT *,'    COMPUTED VALUES'
      PRINT *,'    X(1) = ',ZB(1),' X(N) = ',ZB(N)
      PRINT *
      PRINT *,'    ICON = ',ICON
      PRINT *
      PRINT *,'    N = ',N
      PRINT *
      PRINT *,'    ERROR = ',ERR
      PRINT *,'    ITER=',ITER
      PRINT *
      PRINT *
C
      IF(ERR.LT.1.0D-8.AND.ICON.EQ.0)THEN
         WRITE(*,*)'********** OK **********'
```

```
      ELSE
         WRITE(*,*)'********** NG **********'
      ENDIF
C
      DEALLOCATE( ZPANELFACTORL,ZPANELFACTORU,
     $            NPANELINDEXL,
     $            NPANELINDEXU )
C
      STOP
      END

C ========================================
C     INITIALIZE COEFFICIENT MATRIX
C ========================================
      SUBROUTINE INIT_MAT_DIAG(VA1,VA2,VA3,VC,D_L,OFFSET
     &               ,NX,NY,NZ,XL,YL,ZL,NDIAG,LEN,NDIVP)
      IMPLICIT  REAL*8(A-H,O-Z)
      DIMENSION D_L(NDIVP,NDIAG)
      INTEGER   OFFSET(NDIAG)
C
      IF (NDIAG .LT. 1) THEN
        WRITE (*,*) 'SUBROUTINE INIT_MAT_DIAG:'
        WRITE (*,*) ' NDIAG SHOULD BE GREATER THAN OR EQUAL TO 1'
        RETURN
      ENDIF

!$OMP PARALLEL DEFAULT(PRIVATE)
!$OMP+     SHARED(VA1,VA2,VA3,VC,D_L,OFFSET
!$OMP+      ,NX,NY,NZ,XL,YL,ZL,NDIAG,LEN,NDIVP)

C NDIAG CANNOT BE GREATER THAN 7
      NDIAG_LOC = NDIAG
      IF (NDIAG .GT. 7) NDIAG_LOC = 7

C INITIAL SETTING
      HX = XL/(NX+1)
      HY = YL/(NY+1)
      HZ = ZL/(NZ+1)

!$OMP DO
      DO I = 1,NDIVP
      DO J = 1,NDIAG
```

```
      D_L(I,J) = 0.0
      ENDDO
      ENDDO
!$OMP ENDDO

      NXY = NX*NY

C OFFSET SETTING
!$OMP SINGLE
      L = 1
      IF (NDIAG_LOC .GE. 7) THEN
        OFFSET(L) = -NXY
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 5) THEN
        OFFSET(L) = -NX
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 3) THEN
        OFFSET(L) = -1
        L = L+1
      ENDIF
      OFFSET(L) = 0
      L = L+1
      IF (NDIAG_LOC .GE. 2) THEN
        OFFSET(L) = 1
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 4) THEN
        OFFSET(L) = NX
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 6) THEN
        OFFSET(L) = NXY
      ENDIF
!$OMP END SINGLE

C MAIN LOOP
!$OMP DO
      DO 100 J = 1,LEN
        JS = J
```

```
C DECOMPOSE JS-1 = (K0-1)*NX*NY+(J0-1)*NX+I0-1
        K0 = (JS-1)/NXY+1
        IF (K0 .GT. NZ) THEN
          PRINT*,'ERROR; K0.GH.NZ '
          GOTO 100
        ENDIF
        J0 = (JS-1-NXY*(K0-1))/NX+1
        I0 = JS - NXY*(K0-1) - NX*(J0-1)
        L = 1

        IF (NDIAG_LOC .GE. 7) THEN
          IF (K0 .GT. 1) D_L(J,L) = -(1.0/HZ+0.5*VA3)/HZ
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 5) THEN
          IF (J0 .GT. 1) D_L(J,L) = -(1.0/HY+0.5*VA2)/HY
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 3) THEN
          IF (I0 .GT. 1) D_L(J,L) = -(1.0/HX+0.5*VA1)/HX
          L = L+1
        ENDIF
        D_L(J,L) = 2.0/HX**2+VC
        IF (NDIAG_LOC .GE. 5) THEN
          D_L(J,L) = D_L(J,L) + 2.0/HY**2
          IF (NDIAG_LOC .GE. 7) THEN
            D_L(J,L) = D_L(J,L) + 2.0/HZ**2
          ENDIF
        ENDIF
        L = L+1
        IF (NDIAG_LOC .GE. 2) THEN
          IF (I0 .LT. NX) D_L(J,L) = -(1.0/HX-0.5*VA1)/HX
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 4) THEN
          IF (J0 .LT. NY) D_L(J,L) = -(1.0/HY-0.5*VA2)/HY
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 6) THEN
          IF (K0 .LT. NZ) D_L(J,L) = -(1.0/HZ-0.5*VA3)/HZ
        ENDIF
 100  CONTINUE
```

```
!$OMP ENDDO

!$OMP END PARALLEL

      RETURN
      END

C =======================================
* SOLUTE ERROR
* | Z1 - Z2 |
C =======================================
      REAL*8 FUNCTION ERRNRM(Z1,Z2,LEN)
      IMPLICIT  REAL*8 (A-H,O-Z)
      COMPLEX*16 Z1(*),Z2(*),SS
C
      S = 0D0
      DO 100 I = 1,LEN
        SS  = Z1(I) - Z2(I)
        S = S + DREAL(SS * DCONJG(SS))
 100  CONTINUE
C
      ERRNRM = SQRT( S )
      RETURN
      END
```

(4)　手法概要

対角要素に絶対値の大きな要素を並べる並び換えを行います.
並び換えを行った行列に対して行, 列の均衡化を行うスケーリングを施します.
要素の並び換えや スケーリングおよびピボットでは, 複素数の大きさを表す絶対値は, 実部の絶対値と虚部の絶対値を加えたもので近似します.
この行列をLU分解します.
スーパーノードに対応する非ゼロ要素を2次元のpanelに格納します.
安定化のためのピボットは対角ブロック部分で検索します.
指定した値より絶対値が大きな値が見つかるとピボットとして採用する閾値THEPSZを指定できます. 求めたピボットが小さすぎるとき, SPEPSZで指定した値で近似してLU分解を続けるstatic pivotを指定することも可能です.
詳細は“付録1 参考文献一覧表”の以下の文献を参照願います.
対角要素に絶対値の大きな要素を並べる方法に関しては,の[23],[57]をマッチングの応用アルゴリズムの詳細は[13]を, フィボナッチヒープに関しては[17]を, スパースな非対称な複素行列のLU分解のベースに関しては[19],[2],[22],[48],[68]を行列の均衡化やピボットに関しては[63],[69]をそれぞれ参照願います.

DM_VSEVPH

実対称行列の固有値・固有ベクトル (三重対角化, マルチセクション法, 逆反復法)
CALL DM_VSEVPH(A, K, N, NF, NL, IVEC, ETOL, CTOL, NEV, E, MAXNE, M, EV, ICON)

(1)　機能

実対称行列 $\boldsymbol{A}$ の指定された幾つかの固有値を求めます. 指定に応じて, 固有ベクトルを求めます.

実対称行列 $\boldsymbol{A}$ を Householder 変換で三重対角化し, マルチセクション法により固有値を求めます. 固有ベクトルは逆反復法で求めます.

$$\boldsymbol{Ax} = \lambda \boldsymbol{x}$$

$\boldsymbol{A}$ は $n \times n$ 実対称行列です.

(2)　パラメタ

A.....入力. A(1:N, 1:N)の下三角部分$\{A(i, j)| i \geq j\}$に, 実対称行列$\boldsymbol{A}$の下三角部分$\{a_{ij} | i \geq j\}$を格納します.
演算後の内容は保証されません.
A(K, N)なる倍精度実数型 2 次元配列.

K.....入力. 配列 A の 1 次元目の大きさ(≥ N).

N.....入力. 実対称行列 $\boldsymbol{A}$ の次数 n.

NF .....入力. 固有値を小さい方から番号付けして(多重固有値には多重度分番号を割り当てる), 求める最初の固有値の番号.

NL .....入力. 固有値を小さい方から番号付けして(多重固有値には多重度分番号を割り当てる), 求める最後の固有値の番号.

IVEC .....入力. 制御情報.
1 のとき, 固有値および対応する固有ベクトルの両方を求めます.
1 以外のとき, 固有値のみ求めます.

ETOL .....入力. 固有値が数値的に異なるか多重かを判定する判定値.
3.0D-16 以下のときこの値が標準値として設定されます.

CTOL .....入力. 近接している固有値が近似的に多重かを式(3.1)を利用して判定する判定値. 近似的多重固有値(cluster)に属する固有値の対応する固有ベクトルの 1 次独立性を保証するために使用されます.
5.0D-12 が一般に適当です. ただし非常に大きなクラスタに対しては, 大きな値が必要です.
1.0D-6 ≥ CTOL ≥ ETOL.
CTOL > 1.0D-6 のときは, CTOL=1.0D-6 と設定されます.
CTOL < ETOL のときは, CTOL=10 × ETOL が標準値として設定されます.
((3)使用上の注意 a. 注意①参照).

NEV .....出力. 求められた固有値の個数に関する情報.
NEV(5)なる 1 次元配列.
詳細は以下のとおり.
NEV(1)は, 異なる固有値の個数.
NEV(2)は, 異なる近似的多重固有値(cluster)の個数.
NEV(3)は, 多重度も含んだすべての固有値の個数.
NEV(4)は, 求めた固有値の内最初の固有値の番号.
NEV(5)は, 求めた固有値の内最後の固有値の番号.

E.....出力. 固有値が格納されます.
求められた固有値は E(1:NEV(3))に格納されます.
E(MAXNE)なる 1 次元配列.

MAXNE.....入力. 計算できる固有値の最大個数.
多重度 m の固有値がいくつかあると考えられるとき, n を上限として NL − NF + 1 + 2 × m を設定する必要があります. 配列 E の 1 次元目の大きさ.
NEV(3) > MAXNE のとき, 固有ベクトルの計算はできません.
((3)使用上の注意 a. 注意②参照).

M.....出力. 求められた固有値の多重度に関する情報.
M(i, 1)は i 番目の固有値λ_iの多重度を, M(i, 2)は i 番目の固有値λ_iに関して, 近接している固有値を近似的多重固有値(cluster)と見なしたときの多重度を示します.
((3)使用上の注意 a. 注意①参照).
M(MAXNE, 2)なる 2 次元配列.

EV .....出力. IVEC=1 のとき, 固有値に対応して固有ベクトルが格納されます.
求められた固有ベクトルは, EV(1:N, 1:NEV(3))に格納されます.
EV(K, MAXNE)なる 2 次元配列.

ICON.....出力. コンディションコード.
表 DM_VSEVPH-1 参照

表 DM_VSEVPH-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし	―
20000	密集固有値の計算中, 固有値の総数が MAXNE を超えた.	処理を打ち切る. 固有ベクトルを求めることはできないが, 異なる固有値自体は求められている. NEV(3)に MAXNE に設定するべき大きさが返却されます. ((3)使用上の注意 a. 注意②参照).
30000	NF < 1, NL > N, NL < NF, N <1, K < N, MAXNE < NL - NF+1.	処理を打ち切る.

(3)　使用上の注意

a.　注意

①　本ルーチンは, 求める固有値を交わりのない順序付けられた集合に分けて独立に計算することにより並列化しています.

ε=ETOL としたとき, 連続する固有値λ_j , $j=s-1, s, \ldots, s+k, (k\geq 0)$に対して,

$$\frac{|\lambda_i - \lambda_{i-1}|}{1+\max\left(|\lambda_{i-1}|,|\lambda_i|\right)} \leq \varepsilon \tag{3.1}$$

$i=s, s+1, \ldots, s+k$ となる i に対して(3. 1)を満たし, $i=s-1$ および $i=s+k+1$ について(3. 1)を満たさないとき, これらの固有値λ_j , $j=s-1, s, \ldots, s+k$ は数値的に多重であるとみなされます.

ETOL の標準値は 3. 0D-16～丸め誤差の単位であり, このとき固有値は計算機で求め得る精度まで分離されます.

(3.1)がε=ETOL に対して成立しないとき, λ_{i-1} およびλ_i は異なる固有値と考えます.

ε = ETOL で異なる固有値とみなされた連続する固有値λ_m, $m=t-1, t, \ldots, t+k$, (k≥0)に対して, ε=CTOL としたとき, $i=t, t+1, \ldots, t+k$ について(3.1)を満たし, $i=t-1$ および $i=t+k+1$ について(3.1)を満たさないとき, これらの異なる固有値λ_m, $m=t-1, t, \ldots, t+k$ を近似的に多重(cluster)と見なします. これは, cluster に対応する不変部分空間, つまり 1 次独立な固有ベクトルを計算するために利用されます.

②　MAXNE に, 計算できる固有値の最大個数を指定できます. CTOL を大きくすると, cluster の大きさが大きくなり, 計算する固有値の総数が MAXNE を超えるかも知れません. この場合, CTOL を小さくするか, MAXNE を大きくする必要があります.

計算する固有値の総数が MAXNE を超えた場合, ICON=20000 を返却します. このとき, 固有ベクトルの計算を行うよう指定されていたら固有ベクトルの計算はできません. 固有値は求められていますが, 多重度分だけ繰り返して格納はされてはいません.

つまり, 固有値に関しては, 求められた異なる固有値が, E(1:NEV(1))に, および対応する固有値の多重度が M(1:NEV(1), 1)にそれぞれ格納されています.
固有値がすべて異なり, 近似的多重な固有値もない場合, MAXNE は求める固有値の総数を NT(NT=NL-NF+1)として NT で十分です. 多重な固有値がいくつかあり, 多重度が m と考えられるときは, 少なくとも MAXNE は $NT+2\times m$ とする必要があります.
計算する固有値の総数が MAXNE を超えた場合, 計算を続けるために必要な値が NEV(3)に返却されます. この値で領域を確保して再度呼び出すことで計算を続けることができます.

b. 使用例

固有値・固有ベクトルが分かっている実対称行列の指定された固有値・固有ベクトルを求めます.
(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER (N=2000,K=N)
      PARAMETER (NE=N,MAX_NEV=NE)
      DIMENSION A(K,N),B(K,N),C(K,N),D(K,N),AC(K,N)
      DIMENSION NEV(5),MULT(MAX_NEV,2)
      DIMENSION EVAL(MAX_NEV),EVEC(K,MAX_NEV)
CC
      PAI=4.0D0*DATAN(1.0D0)
      COEF=DSQRT(2.0D0/(N+1))
      DO J=1,N
      DO I=1,N
      D(I,J)=COEF*DSIN(PAI/(N+1)*I*J)
      ENDDO
      ENDDO
CC
      DO J=1,N
      DO I=1,N
      IF(I.EQ.J)THEN
      C(I,J)=I
      ELSE
      C(I,J)=0.0D0
      ENDIF
      ENDDO
      ENDDO
```

```
CC
CC    d x c -> b
CC
      CALL DM_VMGGM(D,K,C,K,B,K,N,N,N,ICON)
CC
CC    b x d -> a
CC
      CALL DM_VMGGM(B,K,D,K,A,K,N,N,N,ICON)
CC
      DO I=1,N
      DO J=I,N
      AC(J,I)=A(J,I)
      ENDDO
      ENDDO
      NF=1
      NL=NE
      IVEC=1
      EVAL_TOL=1.0D-15
      CLUS_TOL=1.0D-10
      CALL DM_VSEVPH( AC,K,N,NF,NL,IVEC,EVAL_TOL,CLUS_TOL,NEV,
     &                EVAL,MAX_NEV,MULT,EVEC,ICON )
      DO I=1,NE,N/20
      PRINT*,'EIGEN VALUE IN EVAL(',I,') = ',EVAL(I)
      ENDDO
C
      STOP
      END
```

(4)　手法概要

本ルーチンは, 実対称行列を三重対角化して三重対角行列の固有値問題を解く方法に基づいています. 三重対角化で, ブロック化した Householder 法で三重対角化する方法(“付録 1 参考文献一覧表”の[30]参照.)を並列化したものです.

三重対角行列の固有値問題は固有値をマルチセクション法で計算し, 固有ベクトルを逆反復法で求めます. 詳細は, DM_VTDEVC の記述および“付録 1 参考文献一覧表”の[61]を参照してください.

元の行列の固有ベクトルは, 三重対角行列の固有ベクトルからなる行列に三重対角化を行う変換行列を掛けて求めます.

DM_VSLDL

正値対称行列の LDL^T 分解 (ブロック化された変形コレスキー分解法)
CALL DM_VSLDL(A, K, N, EPSZ, ICON)

(1) 機能

$n \times n$ の正値対称行列 $\boldsymbol{A}$ を, ブロック化された外積形の変形コレスキー分解法により LDL^T 分解します.

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^T$$

ただし, $\boldsymbol{L}$ は単位下三角行列, $\boldsymbol{D}$ は対角行列です.

(2) パラメタ

A.....入力. 係数行列 $\boldsymbol{A}$.
出力. 行列 $\boldsymbol{L}$ 及び $\boldsymbol{D}^{-1}$.
入力時には, A(1:N, 1:N)の下三角部分$\{A(i, j)| i \geq j\}$に, $\boldsymbol{A}$ の下三角部分$\{a_{ij} | i \geq j\}$を格納します.
出力時には, A(1:N, 1:N)の下三角部分$\{A(i, j) | i \geq j\}$に $\boldsymbol{L}$ と $\boldsymbol{D}^{-1}$ が以下のように格納されます.
l_{ij} : $i > j$ のとき,
d_{ii} の逆数 : $i = j$ のとき,
不定 : $i < j$ のとき.
(図 DM_VSLDL-1 参照)
A(K, N)なる倍精度実数型 2 次元配列.

K.....入力. 配列 A の 1 次元目の大きさ(≥ N).

N.....入力. 係数行列 $\boldsymbol{A}$ の次数 n.

EPSZ.....入力. ピボットの相対零判定値(≥ 0.0).
0.0 のときは標準値が設定されます.
((3)使用上の注意 a. 注意①参照).

ICON.....出力. コンディションコード.
(表 DM_VSLDL-1 参照)

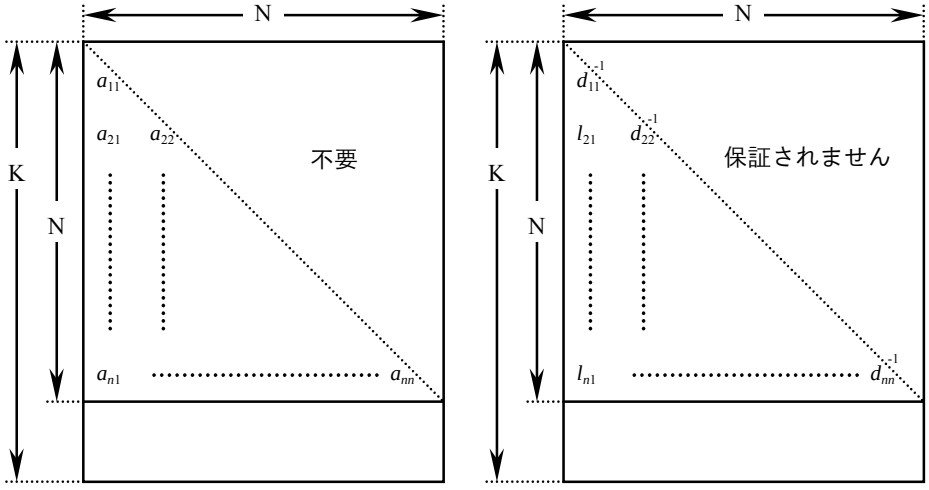


図 DM_VSLDL-1 コレスキー分解法のデータの格納方法

LDL^T 分解を行う正値対称行列の対角要素および下三角部分 a_{ij} を配列 A(i, j), $i=j, \ldots, n, j=1, \ldots, n$ に格納します.

LDL^T 分解された結果は, 行列 $\boldsymbol{D}^{-1}$ が対角部分に, $\boldsymbol{L}$ の対角要素を除いた部分が下三角部分にそれぞれ格納されます.

表 DM_VSLDL-1 コンディションコード

コード	意　　味	処理内容
0	エラーなし	―
10000	ピボットが負となった. 係数行列は正値でない.	処理は続行する.
20000	ピボットが相対的に零となった. 係数行列は非正則である可能性が強い.	処理を打ち切る.
30000	N<1, EPSZ < 0, K < N	

(3)　使用上の注意

a.　注意

①　ピボットの相対零判定値にある値を設定したとすると, この値は次の意味を持っています. すなわち, 変形コレスキー分解法による LDL^T 分解の過程で, ピボットの絶対値がその値より小さくなった場合に, そのピボットの値を相対的に零と見なし, ICON=20000 として処理を打ち切ります. EPSZ の標準値は, 丸め誤差の単位を u としたとき, EPSZ=16u です.

なお, ピボットの絶対値が小さくなっても, 処理を続行させたい場合には, EPSZ に極小の値を与えれば良いのですが, 結果の精度は保証されません.

② 分解の途中でピボットが負となった場合, 係数行列は正値ではありません. このとき, ICON=10000 として処理は続行します. ただし, ピボッティングを行っていないため, 計算誤差は大きい可能性があります.

③ 係数行列の行列式の値を求めるときは, 演算後の配列 A の n 個の対角成分を掛け合わせ, その逆数をとって下さい.

b. 使用例

4000×4000 の行列を, LDL^T 分解します.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER (K=4000,N=4000)
      REAL*8    A(K,N)
C
!$OMP PARALLEL DO DEFAULT(PRIVATE) SHARED(A)
      DO J=1,N
      DO I=J,N
      A(I,J)=MIN(I,J)
      ENDDO
      ENDDO
!$OMP END PARALLEL DO

      CALL DM_VSLDL(A,K,N,1.D-13,ICON)
      WRITE(6,610) ICON
      IF(ICON.GE.20000) GO TO 10
C
      S=1.D0
!$OMP PARALLEL DO DEFAULT(PRIVATE) SHARED(A)
!$OMP+         REDUCTION(*:S)
      DO I=1,N
      S=S*A(I,I)
      ENDDO
!$OMP END PARALLEL DO

      DET=S
      DET=1.D0/DET
      WRITE(6,620) DET
      WRITE(6,640)
      DO J=1,5
```

```
      WRITE(6,600) J,(A(I,J),I=J,5)
      ENDDO
      GO TO 10
600 FORMAT(1H,I5/(10X,3D23.16))
610 FORMAT(/10X,5HICON=,I5)
620 FORMAT(//10X
     *,22HDETERMINANT OF MATRIX=,D23.16)
640 FORMAT(/10X,17HDECOMPOSED MATRIX)
10  stop
      END
```

(4)　手法概要

ブロック化された外積形の変形コレスキー分解法については, “付録 1 参考文献一覧表” の [30], [54], [70]を参照して下さい.

DM_VSRLU

実スパース行列の LU 分解
CALL DM_VSRLU(A, NZ, NROW, NFCNZ, N, IPLEDSM, MZ, ISCLITERMAX, IORDERING, NPERM, ISW, NROWSYM, NFCNZSYM, NASSIGN, NSUPNUM, NFCNZFACTORL, PANELFACTORL, NSIZEFACTORL, NFCNZINDEXL, NPANELINDEXL, NSIZEINDEXL, NDIM, NFCNZFACTORU, PANELFACTORU, NSIZEFACTORU, NFCNZINDEXU, NPANELINDEXU, NSIZEINDEXU, NPOSTO, SCLROW,SCLCOL, EPSZ, THEPSZ, IPIVOT, ISTATIC, SPEPSZ, NFCNZPIVOT, NPIVOTP, NPIVOTQ, W, IW1, IW2, ICON)

(1) 機能

$n \times n$の実スパース行列$\boldsymbol{A}$に対角要素に大きな要素を並べる並び換えを行い. 行および列の均衡化を行うスケーリングを行います. スーパーノードの対角ブロック内で指定した Pivot をとり, LU 分解します.

実スパース行列 $\boldsymbol{A}$ は以下のように変換できます.

$$\boldsymbol{A_1=D_rAP_cD_c}$$

ここで $\boldsymbol{P_c}$ は列の入れ換えを行う直交行列, $\boldsymbol{D_r}$ は行のスケーリングを行う対角行列, $\boldsymbol{D_c}$ は列のスケーリングを行う対角行列です.

$$\boldsymbol{A_2=QPA_1P^TQ^T}$$

$\boldsymbol{A_2}$ は, スーパーノードの対角ブロックで指定された pivot を行い, 行および列の入れ換えを行い LU 分解されます.

ただし, $\boldsymbol{P}$ は, $\boldsymbol{SYM}=\boldsymbol{A_1}+\boldsymbol{A_1^T}$ の非ゼロパターンに対して求めた ordering による行列要素の並び換えを示す置換行列で, $\boldsymbol{Q}$ は $\boldsymbol{SYM}$ に対する post order による行列要素の並び換えを示す置換行列を示します. $\boldsymbol{P}$, $\boldsymbol{Q}$ は直交行列です.

$\boldsymbol{L}$ は下三角行列, $\boldsymbol{U}$ は単位上三角行列です.

Pivot処理で, 閾値 THEPSZ より絶対値が大きな Pivot を見つけることが出来なかったとき, 対角ブロック内の絶対値が最大の要素を Pivot の候補とします. この値が小さ過ぎるときに Static Pivot を与えて近似的に LU 分解を続けることができます.

(2) パラメタ

A..... 入力. 実スパースな係数行列 $\boldsymbol{A}$ を圧縮列格納法で A(1:NZ)に格納します. A(NZ)なる 1 次元配列.
圧縮列格納法については, 実スパース行列と実ベクトル(圧縮列格納法), DM_VMVSCC の図 DM_VMVSCC-1 を参照してください.
((3)使用上の注意 a. 注意⑤参照).

NZ ..... 入力. 実スパースな係数行列 $\boldsymbol{A}$ の非零要素の総数.

NROW..... 入力. 圧縮列格納法で使用される行指標で A に格納される要素が何番目の行ベクトルに属するかを示します.
NROW(NZ)なる 1 次元配列.

NFCNZ..... 入力. 行列 $\boldsymbol{A}$ の各列の非零要素を列方向に圧縮して順次配列 A に格納するとき,対応する列の最初の非零要素が格納される位置を表します.
NFCNZ(N+1)=NZ+1.
NFCNZ(N+1)なる 1 次元配列.

N..... 入力. 行列 $\boldsymbol{A}$ の次数 n.

IPLEDSM..... 入力. 対角要素に大きな要素を並べる列の入れ換えを行うかを指定します.
1 のとき, 列の入れ換えを求めて入れ換えます.
1 以外のとき, 列の入れ換えを行いません.

MZ..... 出力. IPLEDSM に 1 が指定されたとき, 列の入れ換えを表す. MZ(i)=j は行列 $\boldsymbol{a_{ij}}$ の属する j 番目の列を i 番目の列に移動する. $\boldsymbol{a_{ij}}$ は対角要素に並べる大きな要素を表す.
MZ(N)なる 1 次元配列.

ISCLITERMAX.... 入力. 係数行列のスケーリングを行う対角行列 $\boldsymbol{D_r}$ と $\boldsymbol{D_c}$ を反復して求める反復回数.
ISCLITERMAX ≤ 0 のときスケーリングは行わすに, $\boldsymbol{D_r}$ および $\boldsymbol{D_c}$ は単位行列が設定されます.
ISCLITERMAX ≥ 10 のときは 10 回を上限値とします.

IORDERING..... 入力. ordering を表す置換行列 $\boldsymbol{P}$ で $\boldsymbol{PA_1P^T}$ と変換した行列を LU 分解するかを指定します. 置換行列は直交行列です.
10 のとき, ISW=1 で呼び出すと ordering を求める $\boldsymbol{A_1}$ に関する情報を求めて NROWSYM, NFCNZSYM に設定します.
11 のとき, ISW=1 で 10 を指定して呼び出して得た行列を対称化した情報 NROWSYM, NFCNZSYM を使って決めた ordering を NPERM に指定して同じく ISW=1 で呼び出し, 計算を続けることを示します. $\boldsymbol{PA_1P^T}$ を LU 分解する処理を続けます.
10,11 以外のとき, ordering は指定せずに行列 $\boldsymbol{A_1}$ をそのまま LU 分解します.
出力. ISW=1 と IORDERING=10 を指定して呼び出した後, IORDERING に 11 が設定されます. そのため, ordering を NPERM に指定して再び呼び出すときに特に 11 を指定する必要はありません.
((3)使用上の注意 a. 注意①参照).

NPERM..... 入力. IORDERING=11 のとき使用する置換行列をベクトルで指定します.
NPERM(N)なる 1 次元配列.
((3)使用上の注意 a. 注意①参照).

ISW ..... 入力. 呼び出しに関する制御情報を示します.
3)　1 のとき

対称化および symbolic decomposition を行い, 必要な領域が割り当てられているか調べ計算を行います.
IORDERING=10 で呼び出し, ordering を求めるための情報が NROWSYM, NFCNZSYM に出力されます. これらを使って ***SYM*** に対する ordering を求めた後, NPERM に指定して IORDERING=11 を指定してもう 1 回 ISW=1 で呼び出します.
IORDERING が 10,11 以外のときは ordering は行ないません.
4) 2 のとき
ISW=1 で呼び出したとき, PANELFACTORL,
PANELFACTORU, NPANELINDEXL または NPANELINDEXU の大きさが不足して ICON=31000 で終了した処理を継続します. このとき, NSIZEFACTORL, NSIZEFACTORU, NSIZEINDEXL または NSIZEINDEXU に返却された必要な大きさで PANELFACTORL, PANELFACTORU, NPANELINDEXL または NPANELINDEXU を確保し直して指定しなおして再度呼び出します.
これらの引数と実行を制御する引数 ISW 以外の引数に格納されている値は変更してはいけません.

NROWSYM ..... 出力. IORDERING=10 で呼び出したとき, 対称化した $\boldsymbol{SYM}= \boldsymbol{A_1}+\boldsymbol{A_1}^T$ の非ゼロパターンの下三角行列部分の行指標を列圧縮したものが返却されます.
IPLEDSM=1 のときは, $\boldsymbol{A_1}=\boldsymbol{D_r}\boldsymbol{A}\boldsymbol{P_c}\boldsymbol{D_c}$ です.
NROWSYM(NZ+N)なる 1 次元配列.

NFCNZSYM ..... 出力. IORDERING=10 で呼び出したとき, 行列 SYM の下三角部分の各列の非零要素の行指標を列方向に圧縮して順次配列 NROWSYM に格納するとき,対応する列の最初の非零要素が格納される位置を表します.
NFCNZSYM(N+1)=NSYMZ+1 NSYMZ は,総要素数を表します.
NFCNZSYM(N+1)なる 1 次元配列.

NASSIGN..... 出力. 各 supernode に対する L, U は圧縮して 2 次元の panel に格納します. この panel を PANELFACTORL および PANELFACTORU の 1 次元部分配列として順番に割り付けたとき何番目になるかを示します. これらの指標ベクトルも同じように NPANELINDEXL, NPANELINDEXU に割り付けます. j=NASSIGN(i)のとき, i 番目の supernode を j 番目に割り付けたことを示します.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
分解結果の格納方法については図 DM_VSRLU-1 を参照してください.
NASSIGN(N)なる 1 次元配列.

NSUPNUM ..... 出力. supernode の総数.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します. (≤n)

NFCNZFACTORL 出力. 実スパース行列の LU 分解の結果の行列 ***L*** および ***U*** はスーパーノードに対応しておのおの求めます. 各 supernode に対応する ***L*** の列ベクトルは, 共通の行指標ベクトルを持つように圧縮してブロック対角部分に ***U*** の対応部分を含めて 2 次元の panel に格納します. この panel

を PANELFACTORL の 1 次元部分配列として順番に割り付けたとき, i 番目の panel の先頭要素 panel(1,1)が 1 次元配列の PANELFACTORL の何番目の要素になるかを示します.
NFCNZFACTORL(N+1)なる 8 バイト整数型の 1 次元配列.
結果の格納方法については図 DM_VSRLU-1 を参照してください.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.

PANELFACTORL 出力. 各 supernode の分解結果に対応する行列 ***L*** の複数の列ベクトルと行列 ***U*** のブロック対角部分について，共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します. panel を順番に割り付けます. i 番目の supernode に対応する panel が何番目の位置に割り当てられるかは j=NASSIGN(i)から分かります. その先頭位置は NFCNZFACTORL(j)に格納されています. panel ごとに分解結果が格納されます.
i 番目に割付けられた panel の大きさは DIM(1,i)×DIM(2,i)の 2 次元配列と見なせます. i 番目の panel の panel(s,t), s ≥ t, s=1,…,DIM(1,i), t=1,…,DIM(2,i)に下三角行列 ***L*** が格納されます. panel の panel(s,t), s < t, t=1,…,DIM(2,i)には単位上三角行列 ***U*** の対角要素を除いたブロック対角部分が格納されます.
PANELFACTORL(NSIZEFACTORL)なる 1 次元配列.
結果の格納方法については図 DM_VSRLU-1 を参照してください.
((3)使用上の注意 a. 注意③参照).

NSIZEFACTORL . 入力. PANELFACTORL の大きさを示す. 8 バイト整数型.
出力. PANELFACTORL の大きさとして必要な大きさが返却されます.
((3)使用上の注意 a. 注意③参照).

NFCNZINDEXL ... 出力. 各 supernode の分解結果に対応する行列 ***L*** の複数の列ベクトルと行列 ***U*** のブロック対角部分について，共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します. この行指標ベクトルを NPANELINDEXL に順番に割り付けたとき i 番目の行指標ベクトルの先頭要素が 1 次元配列である NPANELINDEXL の何番目の要素になるかを示します.
NFCNZINDEXL(N+1)なる 8 バイト整数型の 1 次元配列.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
結果の格納方法については図 DM_VSRLU-1 を参照してください.

NPANELINDEXL 出力. 各 supernode の分解結果に対応する行列 ***L*** の複数の列ベクトルと行列 ***U*** のブロック対角部分について，共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します. この行指標ベクトルを NPANELINDEXL に順番に割り付けます. i 番目の supernode に対応する panel の行の指標ベクトルが何番目に割り当てられるかは j=NASSIGN(i)から分かります. その先頭位置は NFCNZINDEXL(j)に格納されています.
この行指標は行列 ***SYM*** に対する post order で並び換えたときの行の番号です.
NPANELINDEXL(NSIZEINDEXL)なる 1 次元配列.

結果の格納方法については図 DM_VSRLU-1 を参照してください.
((3)使用上の注意 a. 注意③参照).

NSIZEINDEXL..... 入力. NPANELINDEXL の大きさを示す. 8 バイト整数型.
出力. 必要な大きさが返却されます.
((3)使用上の注意 a. 注意③参照).

NDIM ..... 出力. NDIM(1,i)および NDIM(2,i)は行列 ***L*** に関して i 番目に割り付けられた panel の 1 次元目と 2 次元目の大きさを示します.
NDIM(3,i)は行列 ***U*** に関して割り付けられた panel を転置したときの 1 次元目の大きさに対角ブロックの大きさを加えた大きさを示します.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
NDIM(3,N)なる 2 次元配列.
結果の格納方法については図 DM_VSRLU-1 を参照してください.

NFCNZFACTORU 出力.実スパース行列の LU 分解の結果の行列 ***U*** に関しては, 各 supernode に対応する ***U*** の行ベクトルは, ブロック対角部分を除いて共通の列指標ベクトルを持つように圧縮し, 転置したものを 2 次元の panel に格納します. この panel を PANELFACTORU の 1 次元部分配列として順番に割り付けたとき, i 番目の panel の先頭要素 panel(1,1)が 1 次元配列の PANELFACTORU の何番目の要素になるかを示します.
NFCNZFACTORU(N+1)なる 8 バイト整数型の 1 次元配列.
結果の格納方法については図 DM_VSRLU-1 を参照してください.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.

PANELFACTORU 出力. 各 supernode の分解結果に対応する行列 ***U*** のブロック対角部分を除いた複数の行ベクトルについて, 共通の列指標ベクトルを持つように圧縮し, 転置して 2 次元の panel に格納します. panel を順番に割り付けます. i 番目の supernode に対応する panel が何番目の位置に割り当てられるかは j=NASSIGN(i) から分かります. その先頭位置は NFCNZFACTORU(j)に格納されています. panel ごとに分解結果が格納されます.
i 番目に割付けられた panel の大きさは{DIM(3,i)-DIM(2,i)}×DIM(2,i)の 2 次元配列と見なせます. i 番目の panel の panel(s,t), s=1,…,DIM(3,i)-DIM(2,i), t=1,…,DIM(2,i)に単位上三角行列 ***U*** のブロック対角部分を除いた部分が行ベクトルを圧縮し, 転置して格納します.
PANELFACTORU(NSIZEFACTORU)なる 1 次元配列.
結果の格納方法については図 DM_VSRLU-1 を参照してください.
((3)使用上の注意 a. 注意③参照).

NSIZEFACTORU. 入力. PANELFACTORU の大きさを示す. 8 バイト整数型.
出力. PANELFACTORU の大きさとして必要な大きさが返却されます.
((3)使用上の注意 a. 注意③参照).

NFCNZINDEXU... 出力. 各 supernode の分解結果に対応する行列 ***U*** は対角ブロック部分を除いて複数の行ベクトルについて, 共通の列指標ベクトルを持つように圧縮し, 転置して 2 次元の panel に格納します. 対角ブロック部分も含んだこの列指標ベクトルを NPANELINDEXU に順番に割り付けたとき i

番目の列指標ベクトルの先頭要素が 1 次元配列である NPANELINDEXU の何番目の要素になるかを示します.
NFCNZINDEXU(N+1)なる 8 バイト整数型の 1 次元配列.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
結果の格納方法については図 DM_VSRLU-1 を参照してください.

NPANELINDEXU 出力. 各 supernode の分解結果に対応する行列 ***U*** の対角ブロックを除いた部分を転置して圧縮して 2 次元の panel に格納します. この列指標ベクトルを対角ブロック部分を含めて NPANELINDEXU に順番に割り付けます. i 番目の supernode に対応する panel の行の指標ベクトルが何番目に割り当てられるかは j=NASSIGN(i)から分かります. その先頭位置は NFCNZINDEXU(j)に格納されています.
この行指標は行列 ***SYM*** に対する post order で並び換えたときの列の番号です.
NPANELINDEXU(NSIZEINDEXU)なる 1 次元配列.
結果の格納方法については図 DM_VSRLU-1 を参照してください.
((3)使用上の注意 a. 注意③参照).

NSIZEINDEXU .... 入力. NPANELINDEXU の大きさを示す. 8 バイト整数型.
出力. 必要な大きさが返却されます.
((3)使用上の注意 a. 注意③参照).

NPOSTO ..... 出力. post order で i 番目の node が行列 ***A*** の何番目の列番号に対応するかを表す 1 次元ベクトル.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
NPOSTO(N)なる 1 次元配列.
((3)使用上の注意 a. 注意④参照).

SCLROW ..... 出力. スケーリング対角行列 $\boldsymbol{D}_r$. 対角要素が 1 次元配列に格納されます.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
SCLROW(N)なる 1 次元配列.

SCLCOL..... 出力. スケーリング対角行列 $\boldsymbol{D}_c$. 対角要素が 1 次元配列に格納されます.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
SCLCOL(N)なる 1 次元配列.

EPSZ ..... 入力. 分解過程で対角要素の大きさの絶対値の相対判定値
出力. EPSZ ≤ 0.0 のときは標準値が設定されます.
((3)使用上の注意 a. 注意②参照).

THEPSZ ..... 入力. ピボットの判定での閾値(threshold). これより大きな値はピボットとして採用します. 見つかればその時点で, ピボット検索は打ち切ります.
例えば 1.0D-2 程度.
出力. THEPSZ ≤ 0.0 のときは 1.0D-2 が設定されます.
EPSZ ≥ THEPSZ > 0.0 のときは, EPSZ が設定されます.

IPIVOT..... 入力. スーパーノード内でピボットを行うか, 行う場合どのようなピボットを行うかを指定します. 例えば, 完全ピボットとして 40 を指定.
IPIVOT < 10 : または IPIVOT ≥ 50: ピボットなし.

10 ≤ IPIVOT < 20 : 部分ピボット
20 ≤ IPIVOT < 30 : 対角ピボット
21 : スーパーノード内で対角ピボットがとれないとき, Rook ピボットに変更します.
22 : スーパーノード内で対角ピボットがとれないときは, Rook ピボットに, Rook ピボットが取れないときは完全ピボットに変更します.
30 ≤ IPIVOT < 40 :Rook ピボット
32 : スーパーノード内で Rook ピボットがとれないとき, 完全ピボットをとります.
40 ≤ IPIVOT < 50 : 完全ピボット

ISTATIC..... 入力. Pivot を指定したとき, Static Pivot を行うかを示します.
1) = 1 のとき
スーパーノード内で指定したピボットが SPEPSZ より大きくないとき, ピボットとして DSIGN(SPEPSZ,PIVOT)で近似します. ピボットの値が 0.0d0 なら SPEPSZ に近似します.
このとき, 以下を設定しなければなりません.
a) EPSZ は EPSZ の標準値以下にしなければなりません.
b) ISCLITERMAX は 10 を設定しスケーリングを行う必要があります.
c)THEPSZ ≥ SPEPSZ でなければなりません.
2) ≠1 のとき
Static Pivot は行いません.

SPEPSZ..... 入力. ISTATIC=1 のとき, Static Pivot として使う値.
THEPSZ ≥ SPEPSZ ≥ EPSZ でなければなりません.
出力. SPEPSZ < EPSZ のときは, 1.0D-10 が設定されます.

NFCNZPIVOT..... 出力. スーパーノード内の相対的な位置でのピボットの行, 列の入れ換えの履歴を格納する位置を示す. i 番目の supernode に関する情報が何番目の位置に割り当てられるかはj=NASSIGN(i)から分かります.その先頭位置は NFCNZPIVOT(j)に格納されています. i 番目のスーパーノードの入れ換え情報は, NPIVOTP, NPIVOTQ の is=NFCNZPIVOT(j) ,…, ie=NFCNZPIVOT(j)+NDIM(2,j)-1 番目の要素に格納されます.
NFCNZPIVOT(NSUPNUM+1)なる 1 次元配列.

NPIVOTP..... 出力. スーパーノード内の行の入れ換えに関する情報を格納する.
NPIVOTP(N)なる 1 次元配列.

NPIVOTQ ..... 出力. スーパーノード内の列の入れ換えに関する情報を格納する.
NPIVOTQ(N)なる 1 次元配列.

W..... 作業域.
出力/入力.
大きさ 4*NZ+6*N の 1 次元配列.
ISW=1, 2 で続けて呼び出すとき,呼び出しの間で必要なデータを受け渡すために使われます. 呼び出しの間で値を変更してはいけません.

IW1..... 作業域.
出力/入力.

大きさ 2*NZ+2*(N+1)+16*N の 1 次元配列.
ISW=1, 2 で続けて呼び出すとき,呼び出しの間で必要なデータを受け渡すために使われます. 呼び出しの間で値を変更してはいけません.

IW2..... 作業域.
出力/入力.
大きさ 47*N+47+NZ+4*(N+1)+2*(NZ+N)の 1 次元配列.
ISW=1, 2 で続けて呼び出すとき,呼び出しの間で必要なデータを受け渡すために使われます. 呼び出しの間で値を変更してはいけません.

ICON..... 出力. コンディションコード.
表 DM_VSRLU-1 参照

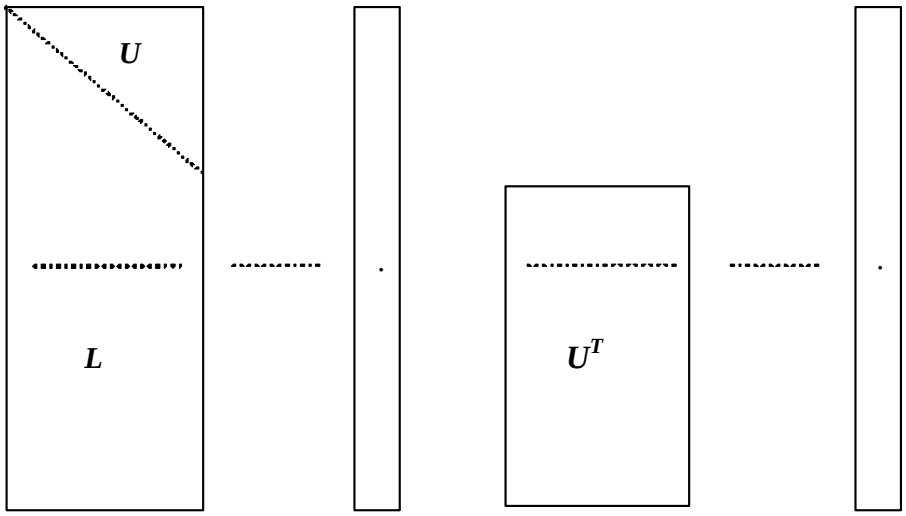


図 DM_VSRLU-1 分解結果の格納概念図

j = NASSIGN(i)　→ i 番目の supernode は j 番目に格納されます.

p = NFCNZFACTORL(j) → j 番目の panel は PANELFACTORL の p 番目の要素から DIM(1,j)×DIM(2,j)の長さを占めます.

q = NFCNZINDEXL(j) → j 番目の panel の行指標を表すベクトルは NPANELINDEXL の q 番目の要素から DIM(1,j)の長さを占めます.

panel は大きさ DIM(1,j)×DIM(2,j)の配列と見なせます.

panel(s,t) , $s \geq t$, s = 1 , … , DIM(1,j), t = 1 , … , DIM(2,j) に分解結果の下三角行列 ***L*** が格納されます.

panel(s,t) , $s < t$, s=1 , … , DIM(2,j), t=1 , … , DIM(2,j) に単位上三角行列 ***U*** の対角ブロック部分が対角要素を除き格納されます.

u = NFCNZFACTORU(j) → j 番目の panel は PANELFACTORU の u 番目の要素から (DIM(3,j)-DIM(2,j))×DIM(2,j)の長さを占めます.

v = NFCNZINDEXU(j) → j 番目の panel の列指標を表すベクトルは NPANELINDEXU の v 番目の要素から DIM(3,j)の長さを占めます.

panel は大きさ(DIM(3,j)-DIM(2,j))×DIM(2,j)の配列と見なせます.

panel(x,y) , x=1 , … , DIM(3,j)-DIM(2,j), y=1 , … , DIM(2,j)

に分解結果の単位上三角行列 $\boldsymbol{U}$ の転置行列 $\boldsymbol{U^T}$ の対角ブロック部分を除いた部分が格納されます.

指標の値は行列 $\boldsymbol{A}$ の列番号をもつ node を post order で並び換えた $\boldsymbol{QAQ^T}$ での列番号を表します.

表 DM_VSRLU-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし	―
10000	ISTATIC=1 のとき小さすぎるピボットを SPEPSZ で置き換える Static Pivot を行いました.	―
20000	ピボットが相対的に零になりました.	処理を打ち切ります.
20100	IPLEDSM=1 を指定して,対角要素に絶対値が大きな要素を並べるため maximum matching を求める処理で, 長さ N の maximum matching がみつかりませんでした. 行列が特異である可能性があります.	
20200	行および列の均衡化を行う対角行列の対角要素を求める過程で,元の行列 A の行もしくは列にゼロベクトルがありました. 行列が特異である可能性があります.	
30000	N<1,NZ<0,NFCNZ(N+1)≠NZ+1, NSIZEFACTORL<1, NSIZEFACTORU<1, NSIZEINDEXL<1, NSIZEINDEXU<1, ISW<1, ISW>2	
30100	NPERM に指定した置換行列が正しくない.	
30200	NROW(j)に格納された i 列目の行指標 k が k<1 または k>N	
30300	i 列目の行指標の数 NFCNZ(i+1)-NFCNZ(i)>n	

コード	意　　味	処理内容
30500	ISTATIC=1 のとき満たすべき条件をみたしていない. EPSZ は標準値 16***u*** より大きかった. または ISCLITERMAX <10 であった. または SPEPSZ>THEPSZ であった.	処理を打ち切ります.
31000	PANELFACTORL の大きさ NSIZEFACTORL または NPANELINDEXL の大きさ NSIZEINDEXL または PANELFACTORU の大きさ NSIZEFACTORU または NPANELINDEXU の大きさ NSIZEINDEXU が小さすぎます.	NSIZEFACTORL または NSIZEINDEXL または NSIZEFACTORU または NSIZEINDEXU で指定された大きさで PANELFACTORL または NPANELINDEXL または PANELFACTORU または NPANELINDEXU を割り付けて ISW=2 として再度呼び出す.

(3)　使用上の注意

a.　注意

①　直交行列である置換行列***P***の要素$p_{ij}=1$のとき, NPERM(i)=jと表現します.
逆は以下のようにして求めることができます.

```
DO i=1,n
 j=NPERM(i)
 NPERMINV(j)=i
 ENDDO
```

Fill-Reducing OrderingはMETISなどを使って求めることができます.
詳細は“付録1 参考文献一覧表”の[43],[44]を参照願います.

②　分解過程で対角要素の大きさの絶対値の相対零判定値にある値を設定したとすると, この値は次の意味を持っています. すなわち, LU分解の過程で, 対角要素の大きさの絶対値がその値より小さくなった場合に, その値を相対的に零と見なし, ICON=20000として処理を打ち切ります. EPSZの標準値は, 丸め誤差の単位をuとしたとき, EPSZ=16uです.
なお, 対角要素の大きさの絶対値が小さくなっても, 処理を続行させたい場合には, EPSZに極小の値を与えれば良いのですが, 結果の精度は保証されません.
Static Pivotを指定した場合, 対角要素がSPEPSZより小さいとき, SPEPSZで近似してLU分解を行います.

③　分解結果を格納する配列PANELFACTORL, NPANELINDEXL, PANELFACTORU, NPANELINDEXUの必要な大きさは,事前には分かりません. 十分大きな配列を割り当てるか,本ルーチンを呼び出してsymbolic decompositionを行った解析の結果を使って割り当てを行うことができます.
例えば, 大きさ1の1次元配列などを割付けます. そしてその大きさ1などの小さな値をNSIZEFACTORL, NSIZEINDEXL, NSIZEFACTORU, NSIZEINDEXU

に指定して, ISW=1で呼び出します.
symbolic decompositionを行い, ICON=31000で終了し, NSIZEFACTORL, NSIZEINDEXL, NSIZEFACTORU, NSIZEINDEXUに必要な大きさが返却されます. 必要な大きさの配列を割付け直して引数に指定して, ISW=2で呼び出すことで, symbolic decomposition以降の処理を続けることができます. 使用例を参照願います.
LU分解の結果を利用してDM_VSRLUXを利用して連立1次方程式を解く上での注意は(3)使用上の注意a. 注意⑤参照.

④ 列番号に対応するノードを考えます. これをpost orderで順番を並び換えたときの番号の対応関係がNPOSTOに格納されています. post orderのi番目のノードが並び換える前の何番目のノードに対応するかを表します. j=nposto(i)はj番目であることを表します.
①同様にこれは直交行列である置換行列$\boldsymbol{Q}$を表し,行列$\boldsymbol{A}$を$\boldsymbol{QAQ^T}$と並び換えることに相当します.
逆変換$\boldsymbol{Q^T}$は以下のようにして求めることができます.

```
DO i=1,n
 j=NPOSTO(i)
 NPOSTOINV(j)=i
 ENDDO
```

⑤ 本ルーチンで得られたLU分解の結果を使い, DM_VSRLUXを引き続いて呼びだすことで連立1次方程式$\boldsymbol{Ax}=\boldsymbol{b}$を解くことができます.
このとき, DM_VSRLUで利用した以下の引数を指定します. 使用例を参照願います.
A, NZ, NROW, NFCNZ, N,
IPLEDSM, MZ, IORDERING, NPERM,
NASSIGN, NSUPNUM,
NFCNZFACTORL, PANELFACTORL,
NSIZEFACTORL, NFCNZINDEXL, NPANELINDEXL,
NSIZEINDEXL, NDIM,
NFCNZFACTORU, PANELFACTORU, NSIZEFACTORU,
NFCNZINDEXU, NPANELINDEXU, NSIZEINDEXU, NPOSTO,
SCLROW,SCLCOL,
NFCNZPIVOT,
NPIVOTP, NPIVOTQ, IW2

b. 使用例

連立 1 次方程式 $\boldsymbol{Ax} = \boldsymbol{f}$ を解きます.
行列は境界条件として立方体の境界で 0 となる楕円型の偏微分方程式に有限差分法を適用して生成されたものを利用します.

$$-\Delta u + a\nabla u + cu = f$$

ここで $a = (a_1, a_2, a_3)$.
行列はサブルーチン init_mat_diag によって生成されます.
対角形式格納法で生成し, 対角ベクトルの下 6 本を圧縮列格納法で格納します.

結果生成された非対称な行列 ***A*** に関する連立 1 次方程式を解きます.
(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT  REAL*8 (A-H,O-Z)
      PARAMETER (NORD=40,KX = NORD,KY =NORD ,KZ = NORD,
     $      N = KX*KY*KZ)
      PARAMETER (NBORDER=N+1,NOFFDIAG=6)
      PARAMETER (K = N+1)
      PARAMETER (NDIAG = 7)
      INTEGER*4 WL
      PARAMETER (NALL=NDIAG*N,
C
     $   WL  =4*NALL+6*N,
     $   IW1L=2*NALL+2*(N+1)+16*N,
     $   IW2L=47*N+47+4*(N+1)+NALL+2*(NALL+N))
C
      DIMENSION NOFST(NDIAG)
      DIMENSION DIAG(K,NDIAG),DIAG2(K,NDIAG)
      DIMENSION A(K*NDIAG),NROW(K*NDIAG),NFCNZ(N+1),
     $          NROWSYM(K*NDIAG+N),NFCNZSYM(N+1),
     $
     $          WC(K*NDIAG),IWC(2,K*NDIAG)
      DIMENSION NPERM(N),W(WL),
     $          NPOSTO(N),NDIM(3,N),
     $          NASSIGN(N),
     $          MZ(N),
     $          IW1(IW1L),IW2(IW2L)
      REAL*8, DIMENSION(:), ALLOCATABLE ::  PANELFACTORL,PANELFACTORU
      INTEGER*4, DIMENSION(:), ALLOCATABLE :: NPANELINDEXL,NPANELINDEXU
      REAL*8 DUMMYFL,DUMMYFU
      INTEGER*4 NDUMMYIL,
     $          NDUMMYIU
      INTEGER*8 NSIZEFACTORL,
     $          NSIZEINDEXL,
     $          NSIZEINDEXU,
     $          NSIZEFACTORU,
     $          NFCNZFACTORL(N+1),
     $          NFCNZFACTORU(N+1),
     $          NFCNZINDEXL(N+1),
```

```
     $          NFCNZINDEXU(N+1)
      DIMENSION B(N),SOLEX(N)
      REAL*8 THEPSZ,EPSZ,SPEPSZ,
     $       SCLROW(N),SCLCOL(N)
C
      INTEGER*4     IPIVOT,ISTATIC,NFCNZPIVOT(N+1),
     $              NPIVOTP(N),NPIVOTQ(N),
     $              IREFINE,ITERMAX,ITER,IPLEDSM
C
      PRINT *,'    LU DECOMPOSITION METHOD'
      PRINT *,'    FOR SPARSE UNSYMMETRIC REAL MATRICES'
      PRINT *,'    IN COMPRESSED COLUMN STORAGE'
      PRINT *
C
      DO I=1,N
      SOLEX(I)=DBLE(1)
      ENDDO
      PRINT *,'    EXPECTED SOLUTIONS'
      PRINT *,'    X(1) = ',SOLEX(1),' X(N) = ',SOLEX(N)
      PRINT *
C
      VA1 = 1.0D0
      VA2 = 2.0D0
      VA3 = 3.0D0
      VC =  4.0D0
      XL = 1.0
      YL = 1.0
      ZL = 1.0
      CALL INIT_MAT_DIAG(VA1,VA2,VA3,VC,DIAG,NOFST
     &          ,KX,KY,KZ,XL,YL,ZL,NDIAG,N,K)
C
      DIAG2=0
C
      DO I=1,NDIAG
C
      IF(NOFST(I).LT.0)THEN
      NBASE=-NOFST(I)
      LENGTH=N-NBASE
      DIAG2(1:LENGTH,I)=DIAG(NBASE+1:N,I)
      ELSE
      NBASE=NOFST(I)
      LENGTH=N-NBASE
```

```
      DIAG2(NBASE+1:N,I)=DIAG(1:LENGTH,I)
      ENDIF
C
      ENDDO
C
      NUMNZ=1
C
      DO J=1,N
      NTOPCFG=1
C
      DO I=NDIAG,1,-1
C
      IF(NTOPCFG.EQ.1)THEN
      NFCNZ(J)=NUMNZ
      NTOPCFG=0
      ENDIF
C
      IF(J.LT.NBORDER.AND.I.GT.NOFFDIAG)THEN
      CONTINUE
      ELSE
C
      IF(DIAG2(J,I).NE.0.0D0)THEN
C
      NCOL=J-NOFST(I)
      A(NUMNZ)=DIAG2(J,I)
      NROW(NUMNZ)=NCOL
C
      NUMNZ=NUMNZ+1
C
      ENDIF
      ENDIF
      ENDDO
      ENDDO
C
      NFCNZ(N+1)=NUMNZ
      NZ=NUMNZ-1
C
      CALL DM_VMVSCC(A,NZ,NROW,NFCNZ,N,SOLEX,
     $             B,WC,IWC,ICON)
C
C     INITIAL CALL WITH IORDER=1
C
```

```
      IORDERING= 0            !
      IPLEDSM=1
      ISCLITERMAX=10
      ISW=1
      NSIZEFACTORL=1
      NSIZEFACTORU=1
      NSIZEINDEXL=1
      NSIZEINDEXU=1
      EPSZ=1.0D-16
      THEPSZ=1.0D-2
      SPEPSZ=0.0D0
      IPIVOT=40
      ISTATIC=0
      IREFINE=1
      EPSR=0.0D0
      ITERMAX=10
C
      CALL DM_VSRLU(A,NZ,NROW,NFCNZ,N,
     $              IPLEDSM,MZ,ISCLITERMAX,IORDERING,
     $              NPERM,ISW,
     $              NROWSYM,NFCNZSYM,
     $              NASSIGN,
     $              NSUPNUM,
     $              NFCNZFACTORL,DUMMYFL,
     $              NSIZEFACTORL,
     $              NFCNZINDEXL,
     $              NDUMMYIL,NSIZEINDEXL,
     $              NDIM,
     $              NFCNZFACTORU,DUMMYFU,
     $              NSIZEFACTORU,
     $              NFCNZINDEXU,
     $              NDUMMYIU,NSIZEINDEXU,
     $              NPOSTO,
     $              SCLROW,SCLCOL,
     $              EPSZ,THEPSZ,
     $              IPIVOT,ISTATIC,SPEPSZ,NFCNZPIVOT,
     $              NPIVOTP,NPIVOTQ,
     $              W,IW1,IW2,ICON)
C
      PRINT*,'ICON=',ICON,' NSIZEFACTORL=',NSIZEFACTORL,
     $       ' NSIZEFACTORU=',NSIZEFACTORU,
     $       'NSIZEINDEXL=',NSIZEINDEXL,
```

```
     $       'NSIZEINDEXU=',NSIZEINDEXU,
     $       'NSUPNUM=',NSUPNUM
C
      ALLOCATE( PANELFACTORL(NSIZEFACTORL) )
      ALLOCATE( PANELFACTORU(NSIZEFACTORU) )
      ALLOCATE( NPANELINDEXL(NSIZEINDEXL) )
      ALLOCATE( NPANELINDEXU(NSIZEINDEXU) )
C
      ISW=2
C
      CALL DM_VSRLU(A,NZ,NROW,NFCNZ,N,
     $              IPLEDSM,MZ,ISCLITERMAX,IORDERING,
     $              NPERM,ISW,
     $              NROWSYM,NFCNZSYM,
     $              NASSIGN,
     $              NSUPNUM,
     $              NFCNZFACTORL,PANELFACTORL,
     $              NSIZEFACTORL,
     $              NFCNZINDEXL,
     $              NPANELINDEXL,NSIZEINDEXL,
     $              NDIM,
     $              NFCNZFACTORU,PANELFACTORU,
     $              NSIZEFACTORU,
     $              NFCNZINDEXU,
     $              NPANELINDEXU,NSIZEINDEXU,
     $              NPOSTO,
     $              SCLROW,SCLCOL,
     $              EPSZ,THEPSZ,
     $              IPIVOT,ISTATIC,SPEPSZ,NFCNZPIVOT,
     $              NPIVOTP,NPIVOTQ,
     $              W,IW1,IW2,ICON)
C
      CALL DM_VSRLUX(N,
     $              IORDERING,
     $              NPERM,
     $              B,
     $              NASSIGN,
     $              NSUPNUM,
     $              NFCNZFACTORL,PANELFACTORL,
     $              NSIZEFACTORL,
     $              NFCNZINDEXL,
     $              NPANELINDEXL,NSIZEINDEXL,
```

```
     $              NDIM,
     $              NFCNZFACTORU,PANELFACTORU,
     $              NSIZEFACTORU,
     $              NFCNZINDEXU,
     $              NPANELINDEXU,NSIZEINDEXU,
     $              NPOSTO,
     $              IPLEDSM,MZ,
     $              SCLROW,SCLCOL,
     $              NFCNZPIVOT,
     $              NPIVOTP,NPIVOTQ,
     $              IREFINE,EPSR,ITERMAX,ITER,
     $              A,NZ,NROW,NFCNZ,
     $              IW2,ICON)
C
      ERR = ERRNRM(SOLEX,B,N)

      PRINT *,'    COMPUTED VALUES'
      PRINT *,'    X(1) = ',B(1),' X(N) = ',B(N)
      PRINT *
      PRINT *,'    ICON = ',ICON
      PRINT *
      PRINT *,'    N = ',N
      PRINT *
      PRINT *,'    ERROR = ',ERR
      PRINT *,'    ITER=',ITER
      PRINT *
      PRINT *

      IF(ERR.LT.1.0D-8.AND.ICON.EQ.0)THEN
         WRITE(*,*)'********** OK **********'
      ELSE
         WRITE(*,*)'********** NG **********'
      ENDIF
C

      DEALLOCATE( PANELFACTORL,PANELFACTORU,
     $            NPANELINDEXL,
     $            NPANELINDEXU )

      STOP
      END
```

```
C ========================================
C     INITIALIZE COEFFICIENT MATRIX
C ========================================
      SUBROUTINE INIT_MAT_DIAG(VA1,VA2,VA3,VC,D_L,OFFSET
     &               ,NX,NY,NZ,XL,YL,ZL,NDIAG,LEN,NDIVP)
      IMPLICIT  REAL*8(A-H,O-Z)
      DIMENSION D_L(NDIVP,NDIAG)
      INTEGER   OFFSET(NDIAG)
C
      IF (NDIAG .LT. 1) THEN
        WRITE (*,*) 'SUBROUTINE INIT_MAT_DIAG:'
        WRITE (*,*) ' NDIAG SHOULD BE GREATER THAN OR EQUAL TO 1'
        RETURN
      ENDIF

!$OMP PARALLEL DEFAULT(PRIVATE)
!$OMP+     SHARED(VA1,VA2,VA3,VC,D_L,OFFSET
!$OMP+      ,NX,NY,NZ,XL,YL,ZL,NDIAG,LEN,NDIVP)

C NDIAG CANNOT BE GREATER THAN 7
      NDIAG_LOC = NDIAG
      IF (NDIAG .GT. 7) NDIAG_LOC = 7

C INITIAL SETTING
      HX = XL/(NX+1)
      HY = YL/(NY+1)
      HZ = ZL/(NZ+1)

!$OMP DO
      DO I = 1,NDIVP
      DO J = 1,NDIAG
      D_L(I,J) = 0.0
      ENDDO
      ENDDO
!$OMP ENDDO

      NXY = NX*NY

C OFFSET SETTING
!$OMP SINGLE
      L = 1
      IF (NDIAG_LOC .GE. 7) THEN
```

```
        OFFSET(L) = -NXY
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 5) THEN
        OFFSET(L) = -NX
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 3) THEN
        OFFSET(L) = -1
        L = L+1
      ENDIF
      OFFSET(L) = 0
      L = L+1
      IF (NDIAG_LOC .GE. 2) THEN
        OFFSET(L) = 1
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 4) THEN
        OFFSET(L) = NX
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 6) THEN
        OFFSET(L) = NXY
      ENDIF
!$OMP END SINGLE

C MAIN LOOP
!$OMP DO
      DO 100 J = 1,LEN
        JS = J

C DECOMPOSE JS-1 = (K0-1)*NX*NY+(J0-1)*NX+I0-1
        K0 = (JS-1)/NXY+1
        IF (K0 .GT. NZ) THEN
          PRINT*,'ERROR; K0.GH.NZ '
          GOTO 100
        ENDIF
        J0 = (JS-1-NXY*(K0-1))/NX+1
        I0 = JS - NXY*(K0-1) - NX*(J0-1)
        L = 1

        IF (NDIAG_LOC .GE. 7) THEN
```

```
          IF (K0 .GT. 1) D_L(J,L) = -(1.0/HZ+0.5*VA3)/HZ
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 5) THEN
          IF (J0 .GT. 1) D_L(J,L) = -(1.0/HY+0.5*VA2)/HY
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 3) THEN
          IF (I0 .GT. 1) D_L(J,L) = -(1.0/HX+0.5*VA1)/HX
          L = L+1
        ENDIF
        D_L(J,L) = 2.0/HX**2+VC
        IF (NDIAG_LOC .GE. 5) THEN
          D_L(J,L) = D_L(J,L) + 2.0/HY**2
          IF (NDIAG_LOC .GE. 7) THEN
            D_L(J,L) = D_L(J,L) + 2.0/HZ**2
          ENDIF
        ENDIF
        L = L+1
        IF (NDIAG_LOC .GE. 2) THEN
          IF (I0 .LT. NX) D_L(J,L) = -(1.0/HX-0.5*VA1)/HX
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 4) THEN
          IF (J0 .LT. NY) D_L(J,L) = -(1.0/HY-0.5*VA2)/HY
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 6) THEN
          IF (K0 .LT. NZ) D_L(J,L) = -(1.0/HZ-0.5*VA3)/HZ
        ENDIF
 100  CONTINUE
!$OMP ENDDO

!$OMP END PARALLEL

      RETURN
      END

C =======================================
* SOLUTE ERROR
* | X1 - X2 |
C =======================================
```

```
      REAL*8 FUNCTION ERRNRM(X1,X2,LEN)
      IMPLICIT  REAL*8 (A-H,O-Z)
      DIMENSION X1(*),X2(*)
C
      S = 0D0
      DO 100 I = 1,LEN
        SS = X1(I) - X2(I)
        S = S + SS * SS
 100  CONTINUE
C
      ERRNRM = SQRT( S )
      RETURN
      END
```

(4) 手法概要

対角要素に絶対値の大きな要素を並べる並び換えを行います.
並び換えを行った行列に対して行, 列の均衡化を行うスケーリングを施します.
この行列を LU 分解します.
スーパーノードに対応する非ゼロ要素を 2 次元の panel に格納します.
安定化のためのピボットは対角ブロック部分で検索します.
指定した値より絶対値が大きな値が見つかるとピボットとして採用する閾値 THEPSZ を指定できます. 求めたピボットが小さすぎるとき, SPEPSZ で指定した値で近似して LU 分解を続ける static pivot を指定することも可能です.
詳細は“付録 1 参考文献一覧表”の以下の文献を参照願います.
対角要素に絶対値の大きな要素を並べる方法に関しては,の[23],[57]をマッチングの応用アルゴリズムの詳細は[13]を, フィボナッチヒープに関しては[17]を,スパースな非対称な実行列の LU 分解に関しては[19],[2],[22],[48],[68]を行列の均衡化やピボットに関しては[63],[69]をそれぞれ参照願います.

DM_VSRLUX

LU 分解された実スパース行列の連立 1 次方程式
CALL DM_VSRLUX(N, IORDERING, NPERM B, NASSIGN, NSUPNUM, NFCNZFACTORL, PANELFACTORL, NSIZEFACTORL, NFCNZINDEXL, NPANELINDEXL, NSIZEINDEXL, NDIM, NFCNZFACTORU, PANELFACTORU, NSIZEFACTORU, NFCNZINDEXU, NPANELINDEXU, NSIZEINDEXU, NPOSTO, IPLEDSM, MZ, SCLROW,SCLCOL, NFCNZPIVOT, NPIVOTP, NPIVOTQ, IREFINE, EPSR, ITERMAX, ITER, A, NZ, NROW, NFCNZ, IW2, ICON)

(1)　機能

$n \times n$ の実スパース行列 $\boldsymbol{A}$ に対角要素に大きな要素を並べる並び換えを行い, 行および列の均衡化を行うスケーリングを行い, そしてスーパーノード内で Pivot をとり, 以下のように LU 分解されています. LU 分解の結果を利用して以下の連立 1 次方程式を解きます.

$$\boldsymbol{Ax=b}$$

行列 $\boldsymbol{A}$ は以下のように分解されています.

$$\boldsymbol{P_{rs}QPD_rAP_cD_cP^TQ^TP_{cs}=LU}$$

ここで, 実スパース行列 $\boldsymbol{A}$ は以下のように変換されています.

$$\boldsymbol{A_1=D_rAP_cD_c}$$

ここで $\boldsymbol{P_c}$ は列の入れ換えを行う直交行列, $\boldsymbol{D_r}$ は行のスケーリングを行う対角行列, $\boldsymbol{D_c}$ は列のスケーリングを行う対角行列です.

$$\boldsymbol{A_2=QPA_1P^TQ^T}$$

$\boldsymbol{A_2}$ は, スーパーノード内に閉じて指定された pivot を行い,行および列の入れ換えを行い LU 分解されています.

$\boldsymbol{P_{rs}}$, $\boldsymbol{P_{cs}}$ はそれぞれ行の入れ換えおよび列の入れ換えを示す直交行列です. 実際の入れ換えはスーパーノードに属する限られた行列の部分で行われます.

ただし, $\boldsymbol{P}$ は,$\boldsymbol{SYM}= \boldsymbol{A_1}+\boldsymbol{A_1^T}$ の非ゼロパターンに対して求めた ordering による行列要素の並び換えを示す置換行列で, $\boldsymbol{Q}$ は $\boldsymbol{SYM}$ に対する post order による行列要素の並び換えを示す置換行列を示します. $\boldsymbol{P}$, $\boldsymbol{Q}$ は直交行列です.

$\boldsymbol{L}$ は下三角行列, $\boldsymbol{U}$ は単位上三角行列です.

解の反復改良で精度を改良することを指定できます.

(2)　パラメタ

N..... 入力. 行列 $\boldsymbol{A}$ の次数 n.

IORDERING..... 入力. 11 のとき, NPERM に ordering を指定して LU 分解されたことを示します. $\boldsymbol{PA_1P^T}$ を LU 分解しました.

11 以外のとき, ordering は指定されていません.
((3)使用上の注意 a. 注意①参照).

NPERM ..... 入力. IORDERING=11 のとき使用する置換行列をベクトルで指定します.
NPERM(N)なる 1 次元配列.
((3)使用上の注意 a. 注意②参照).

B ..... 入力. ***Ax=b*** の右辺定数ベクトル.
出力. 解ベクトル.
B(N)なる 1 次元配列.

NASSIGN..... 入力. 各 supernode に対する ***L***, ***U*** は圧縮して 2 次元の panel に格納します. この panel を PANELFACTORL および PANELFACTORU の 1 次元部分配列として順番に割り付けたとき何番目になるかを示します. これらの指標ベクトルも同じように NPANELINDEXL, NPANELINDEXU に割り付けます. j=NASSIGN(i)のとき, i 番目の supernode を j 番目に割り付けたことを示します.
分解結果の格納方法については図 DM_VSRLUX-1 を参照してください.
NASSIGN(N)なる 1 次元配列.

NSUPNUM ..... 入力. supernode の総数. (≤n)

NFCNZFACTORL 入力. 実スパース行列の LU 分解の結果の行列 ***L*** および ***U*** はスーパーノードに対応しておのおの求めます. 各 supernode に対応する ***L*** の列ベクトルは, 共通の行指標ベクトルを持つように圧縮してブロック対角部分に ***U*** の対応部分を含めて 2 次元の panel に格納します. この panel を PANELFACTORL の 1 次元部分配列として順番に割り付けたとき, i 番目の panel の先頭要素 panel(1,1)が 1 次元配列の PANELFACTORL の何番目の要素になるかを示します.
NFCNZFACTORL(N+1)なる 8 バイト整数型の 1 次元配列.
結果の格納方法については図 DM_VSRLUX-1 を参照してください.

PANELFACTORL 入力. 各 supernode の分解結果に対応する行列 ***L*** の複数の列ベクトルと行列 ***U*** のブロック対角部分について，共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します. panel を順番に割り付けます. i 番目の supernode に対応する panel が何番目の位置に割り当てられるかは j=NASSIGN(i)から分かります. その先頭位置は NFCNZFACTORL(j)に格納されています. panel ごとに分解結果が格納されます.
i 番目に割付けられた panel の大きさは DIM(1,i)×DIM(2,i)の 2 次元配列と見なせます. i 番目の panel の panel(s,t), $s \geq t$, s = 1,…,DIM(1,i), t=1,…,DIM(2,i)に下三角行列 ***L*** が格納されます. panel の panel(s,t), s<t, t=1,…,DIM(2,i)には単位上三角行列 ***U*** の対角要素を除いたブロック対角部分が格納されます.
PANELFACTORL(NSIZEFACTORL)なる 1 次元配列.
結果の格納方法については図 DM_VSRLUX-1 を参照してください.

NSIZEFACTORL . 入力. PANELFACTORL の大きさを示す. 8 バイト整数型.

NFCNZINDEXL ... 入力. 各 supernode の分解結果に対応する行列 ***L*** の複数の列ベクトルと行列 ***U*** のブロック対角部分について，共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します. この行指標ベクトルを NPANELINDEXL に順番に割り付けたとき i 番目の行指標ベクトルの先頭要素が 1 次元配列である NPANELINDEXL の何番目の要素になるかを示します.
NFCNZINDEXL(N+1)なる 8 バイト整数型の 1 次元配列.

NPANELINDEXL 入力. 各 supernode の分解結果に対応する行列 ***L*** の複数の列ベクトルと行列 ***U*** のブロック対角部分について，共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します. この行指標ベクトルを NPANELINDEXL に順番に割り付けます. i 番目の supernode に対応する panel の行の指標ベクトルが何番目に割り当てられるかは j=NASSIGN(i)から分かります. その先頭位置は NFCNZINDEXL(j)に格納されています.
この行指標は行列 ***SYM*** に対する post order で並び換えたときの行の番号です.
NPANELINDEXL(NSIZEINDEXL)なる 1 次元配列.
結果の格納方法については図 DM_VSRLUX-1 を参照してください.

NSIZEINDEXL ..... 入力. NPANELINDEXL の大きさを示す. 8 バイト整数型.

NDIM ..... 入力. NDIM(1,i)および NDIM(2,i)は行列 ***L*** に関して i 番目に割り付けられた panel の 1 次元目と 2 次元目の大きさを示します.
NDIM(3,i)は行列 ***U*** に関して割り付けられた panel を転置したときの 1 次元目の大きさに対角ブロックの大きさを加えた大きさを示します.
NDIM(3,N)なる 2 次元配列.
結果の格納方法については図 DM_VSRLUX-1 を参照してください.

NFCNZFACTORU 入力. 実スパース行列の LU 分解の結果の行列 ***U*** に関しては, 各 supernode に対応する ***U*** の行ベクトルは, ブロック対角部分を除いて共通の列指標ベクトルを持つように圧縮し, 転置して 2 次元の panel に格納します. この panel を PANELFACTORU の 1 次元部分配列として順番に割り付けたとき, i 番目の panel の先頭要素 panel(1,1)が 1 次元配列の PANELFACTORU の何番目の要素になるかを示します.
NFCNZFACTORU(N+1)なる 8 バイト整数型の 1 次元配列.
結果の格納方法については図 DM_VSRLUX-1 を参照してください.

PANELFACTORU 入力. 各 supernode の分解結果に対応する行列 ***U*** のブロック対角部分を除いた複数の行ベクトルについて，共通の列指標ベクトルを持つように圧縮し, 転置して 2 次元の panel に格納します. panel を順番に割り付けます. i 番目の supernode に対応する panel が何番目の位置に割り当てられるかは j=NASSIGN(i) から分かります. その先頭位置は NFCNZFACTORU(j)に格納されています. panel ごとに分解結果が格納されます.
i 番目に割付けられた panel の大きさは{DIM(3,i)-DIM(2,i)}×DIM(2,i)の 2 次元配列と見なせます. i 番目の panel の panel(s,t), s=1,…,DIM(3,i)-

DIM(2,i), t=1,…,DIM(2,i)に単位上三角行列 ***U*** のブロック対角部分を除いた部分が行ベクトルを圧縮し, 転置したものが格納されます.
PANELFACTORU(NSIZEFACTORU)なる 1 次元配列.
結果の格納方法については図 DM_VSRLUX-1 を参照してください.

NSIZEFACTORU. 入力. PANELFACTORU の大きさを示す. 8 バイト整数型.

NFCNZINDEXU... 入力. 各 supernode の分解結果に対応する行列 ***U*** は対角ブロック部分を除いて複数の行ベクトルについて, 共通の列指標ベクトルを持つように圧縮し, 転置した形で 2 次元の panel に格納します. 対角ブロック部分も含んだこの列指標ベクトルを NPANELINDEXU に順番に割り付けたとき i 番目の列指標ベクトルの先頭要素が 1 次元配列である NPANELINDEXU の何番目の要素になるかを示します.
NFCNZINDEXU(N+1)なる 8 バイト整数型の 1 次元配列.
結果の格納方法については図 DM_VSRLUX-1 を参照してください.

NPANELINDEXU 入力. 各 supernode の分解結果に対応する行列 ***U*** の対角ブロックを除いた部分を転置して圧縮して 2 次元の panel に格納します. 対応する列指標ベクトルには対角ブロック部分を含めたものを NPANELINDEXU に順番に割り付けます. i 番目の supernode に対応する panel の行の指標ベクトルが何番目に割り当てられるかは j=NASSIGN(i)から分かります. その先頭位置は NFCNZINDEXU(j)に格納されています.
この行指標は行列 ***SYM*** に対する post order で並び換えたときの列の番号です.
NPANELINDEXU(NSIZEINDEXU)なる 1 次元配列.
結果の格納方法については図 DM_VSRLUX-1 を参照してください.

NSIZEINDEXU .... 入力. NPANELINDEXU の大きさを示す. 8 バイト整数型.

NPOSTO ..... 入力. post order で i 番目の node が行列 ***A*** の何番目の列番号に対応するかを表す 1 次元ベクトル.
((3)使用上の注意 a. 注意③参照).

IPLEDSM..... 入力. LU 分解を行うときに対角要素に大きな要素を並べる列の入れ換えを行ったかを指定します.
1 のとき, 列の入れ換えを行いました.
1 以外のとき, 列の入れ換えを行っていません.

MZ..... 入力. IPLEDSM に 1 が指定されたとき, 列の入れ換えを表す. MZ(i)=j は行列 a_{ij} の属する j 番目の列を i 番目の列に移動する. a_{ij} は対角要素に並べる大きな要素を表す.
MZ(N)なる 1 次元配列.

SCLROW ..... 入力. スケーリング対角行列 D_r. 対角要素が 1 次元配列に格納されます.
SCLROW(N)なる 1 次元配列.

SCLCOL..... 入力. スケーリング対角行列 D_c. 対角要素が 1 次元配列に格納されます.
SCLCOL(N)なる 1 次元配列.

NFCNZPIVOT ..... 入力. スーパーノード内の相対的な位置でのピボットの行, 列の入れ換えの履歴を格納する位置を示す. i 番目の supernode に関する情報が何番目の位置に割り当てられるかは j=NASSIGN(i)から分かります. その先

頭位置は NFCNZPIVOT(j)に格納されています. i 番目のスーパーノードの入れ換え情報は, NPIVOTP, NPIVOTQ の is=NFCNZPIVOT(j) ,…, ie=NFCNZPIVOT(j)+NDIM(2,j)-1 番目の要素に格納されます.
NFCNZPIVOT(NSUPNUM+1)なる 1 次元配列.

NPIVOTP ..... 入力. スーパーノード内の行の入れ換えに関する情報を格納します.
NPIVOTP(N)なる 1 次元配列.

NPIVOTQ ..... 入力. スーパーノード内の列の入れ換えに関する情報を格納します.
NPIVOTQ(N)なる 1 次元配列.

IREFINE..... 入力. LU 分解結果を利用して解を求めるときに, 解の反復改良を行うかどうかを示す. 残差ベクトルを計算するときに 4 倍精度で計算します.
=1 : 解の反復改良を行う. 反復改良して得られる残差ベクトル $\boldsymbol{r}_k$ の絶対値の差分がひとつ前の差分の 1/4 より大きくなるまで, 反復改良を行います.
≠1: 解の反復改良を行わない.

EPSR ..... 入力. 解の残差ベクトル $\boldsymbol{b}-\boldsymbol{Ax}$ の絶対値が, $\boldsymbol{b}$ の絶対値に対して十分小さいかを判定する判定値
EPSR ≤ 0.0 のとき 1.0D-6 が設定されます.

ITERMAX..... 入力. (≥ 1) 反復改良を行うときの最大反復回数.

ITER..... 出力. 反復改良を行った回数.

A..... 入力. 実スパースな係数行列 $\boldsymbol{A}$ を圧縮列格納法で A(1:NZ)に格納します.
A(NZ)なる 1 次元配列.
圧縮列格納法については,実スパース行列と実ベクトル(圧縮列格納法), DM_VMVSCC の図 DM_VMVSCC-1 を参照してください.

NZ ..... 入力. 実スパースな係数行列 $\boldsymbol{A}$ の非零要素の総数.

NROW..... 入力. 圧縮列格納法で使用される行指標で A に格納される要素が何番目の行ベクトルに属するかを示します.
NROW(NZ)なる 1 次元配列.

NFCNZ..... 入力. 行列 $\boldsymbol{A}$ の各列の非零要素を列方向に圧縮して順次配列 A に格納するとき,対応する列の最初の非零要素が格納される位置を表します.
NFCNZ(N+1)=NZ+1.
NFCNZ(N+1)なる 1 次元配列.

IW2..... 作業域.
入力.
大きさ 47*N+47+NZ+4*(N+1)+2*(NZ+N)の 1 次元配列.
実スパース行列の LU 分解を行う DM_VSRLU からのデータの受け渡しに使われます. 呼び出しの間で値を変更してはいけません.

ICON..... 出力.コンディションコード.
表 DM_VSRLUX-1 参照

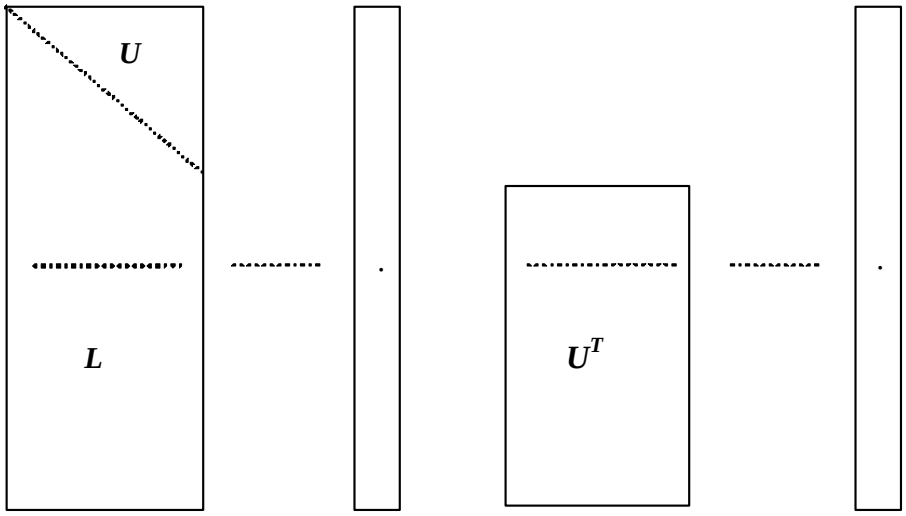


図 DM_VSRLUX-1 分解結果の格納概念図

j = NASSIGN(i)　→ i 番目の supernode は j 番目に格納されます.

p = NFCNZFACTORL(j) → j 番目の panel は PANELFACTORL の p 番目の要素から DIM(1,j)×DIM(2,j)の長さを占めます.

q = NFCNZINDEXL(j) → j 番目の panel の行指標を表すベクトルは NPANELINDEXL の q 番目の要素から DIM(1,j)の長さを占めます.

panel は大きさ DIM(1,j)×DIM(2,j)の配列と見なせます.

panel(s,t) , $s \geq t$, s = 1 , … , DIM(1,j), t=1 , … , DIM(2,j) に分解結果の下三角行列 $\boldsymbol{L}$ が格納されます.

panel(s,t) , $s < t$, s = 1 , … , DIM(2,j), t=1 , … , DIM(2,j) に単位上三角行列 $\boldsymbol{U}$ の対角ブロック部分が対角要素を除き格納されます.

u = NFCNZFACTORU(j) → j 番目の panel は PANELFACTORU の u 番目の要素から (DIM(3,j)-DIM(2,j))×DIM(2,j)の長さを占めます.

v = NFCNZINDEXU(j) → j 番目の panel の列指標を表すベクトルは NPANELINDEXU の v 番目の要素から DIM(3,j)の長さを占めます.

panel は大きさ(DIM(3,j)-DIM(2,j))×DIM(2,j)の配列と見なせます.

panel(x,y) , x=1 , … , DIM(3,j)-DIM(2,j), y=1 , … , DIM(2,j) に分解結果の単位上三角行列 $\boldsymbol{U}$ の転置行列 $\boldsymbol{U}^T$ の対角ブロック部分を除いた部分が格納されます.

指標の値は行列 $\boldsymbol{A}$ の列番号をもつ node を post order で並び換えた $\boldsymbol{QAQ}^T$ での列番号を表します.

表 DM_VSRLUX-1　コンディションコード使用上の注意

	意　　味	処理内容
0	エラーなし	ー
20400	LU 分解された行列の対角要素にゼロがあります.	処理を打ち切ります.
20500	求めた解ベクトルに対する残差ベクトルのノルムが方程式 $\boldsymbol{Ax}=\boldsymbol{b}$ の右辺ベクトルのノルムの EPSR 倍より大きい. 係数行列は特異に近い可能性があります.	
30000	N<1,NZ<0,NFCNZ(N+1)≠NZ+1, NSIZEFACTORL<1, NSIZEFACTORU<1, NSIZEINDEXL<1, NSIZEINDEXU < 1, IREFINE=1 のとき ITERMAX<1	
30100	NPERM に指定した置換行列が正しくない.	
30200	NROW(j)に格納された i 列目の列指標 k が k<1 または k>N	
30300	i 列目の行指標の数 NFCNZ(i+1)-NFCNZ(i)>n	

(3)　使用上の注意

a.　注意

① DM_VSRLUでLU分解を行った結果を利用します.
DM_VSRLUの(3)使用上の注意a. 注意⑤参照やDM_VSRLUXの使用例を参照願います.

② 直交行列である置換行列$\boldsymbol{P}$の要素$\boldsymbol{p_{ij}=1}$のとき, NPERM(i)=jと表現します.
逆は以下のようにして求めることができます.

```
DO i=1,n
  j=NPERM(i)
  NPERMINV(j)=i
  ENDDO
```

③ 列番号に対応するノードを考えます. これをpost orderで順番を並び換えたときの番号の対応関係がNPOSTOに格納されています. post orderのi番目のノードが並び換える前の何番目のノードに対応するかを表します. j=nposto(i)はj番目であることを表します.
②同様にこれは直交行列である置換行列$\boldsymbol{Q}$を表し,行列$\boldsymbol{A}$を$\boldsymbol{QAQ^T}$と並び換えることに相当します.
逆変換$\boldsymbol{Q^T}$は以下のようにして求めることができます.

```
DO i=1,n
  j=NPOSTO(i)
  NPOSTOINV(j)=i
  ENDDO
```

b. 使用例

連立 1 次方程式 $\boldsymbol{Ax} = \boldsymbol{f}$ を解きます.

行列は境界条件として立方体の境界で 0 となる楕円型の偏微分方程式に有限差分法を適用して生成されたものを利用します.

$$-\Delta u + a\nabla u + cu = f$$

ここで $a = (a_1, a_2, a_3)$.

行列はサブルーチン init_mat_diag によって生成されます.

対角形式格納法で生成し, 対角ベクトルの下 6 本を圧縮列格納法で格納します.

結果生成された非対称な行列 $\boldsymbol{A}$ に関する連立 1 次方程式を解きます.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT  REAL*8 (A-H,O-Z)
      PARAMETER (NORD=40,KX = NORD,KY =NORD ,KZ = NORD,
     $      N = KX*KY*KZ)
      PARAMETER (NBORDER=N+1,NOFFDIAG=6)
      PARAMETER (K = N+1)
      PARAMETER (NDIAG = 7)
      INTEGER*4 WL
      PARAMETER (NALL=NDIAG*N,
C
     $   WL  =4*NALL+6*N,
     $   IW1L=2*NALL+2*(N+1)+16*N,
     $   IW2L=47*N+47+4*(N+1)+NALL+2*(NALL+N))
C
      DIMENSION NOFST(NDIAG)
      DIMENSION DIAG(K,NDIAG),DIAG2(K,NDIAG)
      DIMENSION A(K*NDIAG),NROW(K*NDIAG),NFCNZ(N+1),
     $          NROWSYM(K*NDIAG+N),NFCNZSYM(N+1),
     $
     $          WC(K*NDIAG),IWC(2,K*NDIAG)
      DIMENSION NPERM(N),W(WL),
     $          NPOSTO(N),NDIM(3,N),
     $          NASSIGN(N),
     $          MZ(N),
     $          IW1(IW1L),IW2(IW2L)
      REAL*8, DIMENSION(:), ALLOCATABLE ::  PANELFACTORL,PANELFACTORU
      INTEGER*4, DIMENSION(:), ALLOCATABLE :: NPANELINDEXL,NPANELINDEXU
      REAL*8 DUMMYFL,DUMMYFU
      INTEGER*4 NDUMMYIL,
```

```
     $          NDUMMYIU
      INTEGER*8 NSIZEFACTORL,
     $          NSIZEINDEXL,
     $          NSIZEINDEXU,
     $          NSIZEFACTORU,
     $          NFCNZFACTORL(N+1),
     $          NFCNZFACTORU(N+1),
     $          NFCNZINDEXL(N+1),
     $          NFCNZINDEXU(N+1)
      DIMENSION B(N),SOLEX(N)
      REAL*8 THEPSZ,EPSZ,SPEPSZ,
     $       SCLROW(N),SCLCOL(N)
C
      INTEGER*4     IPIVOT,ISTATIC,NFCNZPIVOT(N+1),
     $              NPIVOTP(N),NPIVOTQ(N),
     $              IREFINE,ITERMAX,ITER,IPLEDSM
C
      PRINT *,'    LU DECOMPOSITION METHOD'
      PRINT *,'    FOR SPARSE UNSYMMETRIC REAL MATRICES'
      PRINT *,'    IN COMPRESSED COLUMN STORAGE'
      PRINT *
C
      DO I=1,N
      SOLEX(I)=DBLE(1)
      ENDDO
      PRINT *,'    EXPECTED SOLUTIONS'
      PRINT *,'    X(1) = ',SOLEX(1),' X(N) = ',SOLEX(N)
      PRINT *
C
      VA1 = 1.0D0
      VA2 = 2.0D0
      VA3 = 3.0D0
      VC =  4.0D0
      XL = 1.0
      YL = 1.0
      ZL = 1.0
      CALL INIT_MAT_DIAG(VA1,VA2,VA3,VC,DIAG,NOFST
     &          ,KX,KY,KZ,XL,YL,ZL,NDIAG,N,K)
C
      DIAG2=0
C
      DO I=1,NDIAG
```

```
C
      IF(NOFST(I).LT.0)THEN
      NBASE=-NOFST(I)
      LENGTH=N-NBASE
      DIAG2(1:LENGTH,I)=DIAG(NBASE+1:N,I)
      ELSE
      NBASE=NOFST(I)
      LENGTH=N-NBASE
      DIAG2(NBASE+1:N,I)=DIAG(1:LENGTH,I)
      ENDIF
C
      ENDDO
C
      NUMNZ=1
C
      DO J=1,N
      NTOPCFG=1
C
      DO I=NDIAG,1,-1
C
      IF(NTOPCFG.EQ.1)THEN
      NFCNZ(J)=NUMNZ
      NTOPCFG=0
      ENDIF
C
      IF(J.LT.NBORDER.AND.I.GT.NOFFDIAG)THEN
      CONTINUE
      ELSE
C
      IF(DIAG2(J,I).NE.0.0D0)THEN
C
      NCOL=J-NOFST(I)
      A(NUMNZ)=DIAG2(J,I)
      NROW(NUMNZ)=NCOL
C
      NUMNZ=NUMNZ+1
C
      ENDIF
      ENDIF
      ENDDO
      ENDDO
C
```

```
      NFCNZ(N+1)=NUMNZ
      NZ=NUMNZ-1
C
      CALL DM_VMVSCC(A,NZ,NROW,NFCNZ,N,SOLEX,
     $             B,WC,IWC,ICON)
C
C     INITIAL CALL WITH IORDER=1
C
      IORDERING= 0            !
      IPLEDSM=1
      ISCLITERMAX=10
      ISW=1
      NSIZEFACTORL=1
      NSIZEFACTORU=1
      NSIZEINDEXL=1
      NSIZEINDEXU=1
      EPSZ=1.0D-16
      THEPSZ=1.0D-2
      SPEPSZ=0.0D0
      IPIVOT=40
      ISTATIC=0
      IREFINE=1
      EPSR=0.0D0
      ITERMAX=10
C
      CALL DM_VSRLU(A,NZ,NROW,NFCNZ,N,
     $              IPLEDSM,MZ,ISCLITERMAX,IORDERING,
     $              NPERM,ISW,
     $              NROWSYM,NFCNZSYM,
     $              NASSIGN,
     $              NSUPNUM,
     $              NFCNZFACTORL,DUMMYFL,
     $              NSIZEFACTORL,
     $              NFCNZINDEXL,
     $              NDUMMYIL,NSIZEINDEXL,
     $              NDIM,
     $              NFCNZFACTORU,DUMMYFU,
     $              NSIZEFACTORU,
     $              NFCNZINDEXU,
     $              NDUMMYIU,NSIZEINDEXU,
     $              NPOSTO,
     $              SCLROW,SCLCOL,
```

```
     $              EPSZ,THEPSZ,
     $              IPIVOT,ISTATIC,SPEPSZ,NFCNZPIVOT,
     $              NPIVOTP,NPIVOTQ,
     $              W,IW1,IW2,ICON)
C
      PRINT*,'ICON=',ICON,' NSIZEFACTORL=',NSIZEFACTORL,
     $       ' NSIZEFACTORU=',NSIZEFACTORU,
     $       'NSIZEINDEXL=',NSIZEINDEXL,
     $       'NSIZEINDEXU=',NSIZEINDEXU,
     $       'NSUPNUM=',NSUPNUM
C
      ALLOCATE( PANELFACTORL(NSIZEFACTORL) )
      ALLOCATE( PANELFACTORU(NSIZEFACTORU) )
      ALLOCATE( NPANELINDEXL(NSIZEINDEXL) )
      ALLOCATE( NPANELINDEXU(NSIZEINDEXU) )
C
      ISW=2
C
      CALL DM_VSRLU(A,NZ,NROW,NFCNZ,N,
     $              IPLEDSM,MZ,ISCLITERMAX,IORDERING,
     $              NPERM,ISW,
     $              NROWSYM,NFCNZSYM,
     $              NASSIGN,
     $              NSUPNUM,
     $              NFCNZFACTORL,PANELFACTORL,
     $              NSIZEFACTORL,
     $              NFCNZINDEXL,
     $              NPANELINDEXL,NSIZEINDEXL,
     $              NDIM,
     $              NFCNZFACTORU,PANELFACTORU,
     $              NSIZEFACTORU,
     $              NFCNZINDEXU,
     $              NPANELINDEXU,NSIZEINDEXU,
     $              NPOSTO,
     $              SCLROW,SCLCOL,
     $              EPSZ,THEPSZ,
     $              IPIVOT,ISTATIC,SPEPSZ,NFCNZPIVOT,
     $              NPIVOTP,NPIVOTQ,
     $              W,IW1,IW2,ICON)
C
      CALL DM_VSRLUX(N,
     $              IORDERING,
```

```
     $              NPERM,
     $              B,
     $              NASSIGN,
     $              NSUPNUM,
     $              NFCNZFACTORL,PANELFACTORL,
     $              NSIZEFACTORL,
     $              NFCNZINDEXL,
     $              NPANELINDEXL,NSIZEINDEXL,
     $              NDIM,
     $              NFCNZFACTORU,PANELFACTORU,
     $              NSIZEFACTORU,
     $              NFCNZINDEXU,
     $              NPANELINDEXU,NSIZEINDEXU,
     $              NPOSTO,
     $              IPLEDSM,MZ,
     $              SCLROW,SCLCOL,
     $              NFCNZPIVOT,
     $              NPIVOTP,NPIVOTQ,
     $              IREFINE,EPSR,ITERMAX,ITER,
     $              A,NZ,NROW,NFCNZ,
     $              IW2,ICON)
C
      ERR = ERRNRM(SOLEX,B,N)

      PRINT *,'    COMPUTED VALUES'
      PRINT *,'    X(1) = ',B(1),' X(N) = ',B(N)
      PRINT *
      PRINT *,'    ICON = ',ICON
      PRINT *
      PRINT *,'    N = ',N
      PRINT *
      PRINT *,'    ERROR = ',ERR
      PRINT *,'    ITER=',ITER
      PRINT *
      PRINT *

      IF(ERR.LT.1.0D-8.AND.ICON.EQ.0)THEN
         WRITE(*,*)'********** OK **********'
      ELSE
         WRITE(*,*)'********** NG **********'
      ENDIF
C
```

```
      DEALLOCATE( PANELFACTORL,PANELFACTORU,
     $            NPANELINDEXL,
     $            NPANELINDEXU )

      STOP
      END

C =======================================
C     INITIALIZE COEFFICIENT MATRIX
C =======================================
      SUBROUTINE INIT_MAT_DIAG(VA1,VA2,VA3,VC,D_L,OFFSET
     &               ,NX,NY,NZ,XL,YL,ZL,NDIAG,LEN,NDIVP)
      IMPLICIT  REAL*8(A-H,O-Z)
      DIMENSION D_L(NDIVP,NDIAG)
      INTEGER   OFFSET(NDIAG)
C
      IF (NDIAG .LT. 1) THEN
        WRITE (*,*) 'SUBROUTINE INIT_MAT_DIAG:'
        WRITE (*,*) ' NDIAG SHOULD BE GREATER THAN OR EQUAL TO 1'
        RETURN
      ENDIF

!$OMP PARALLEL DEFAULT(PRIVATE)
!$OMP+     SHARED(VA1,VA2,VA3,VC,D_L,OFFSET
!$OMP+      ,NX,NY,NZ,XL,YL,ZL,NDIAG,LEN,NDIVP)

C NDIAG CANNOT BE GREATER THAN 7
      NDIAG_LOC = NDIAG
      IF (NDIAG .GT. 7) NDIAG_LOC = 7

C INITIAL SETTING
      HX = XL/(NX+1)
      HY = YL/(NY+1)
      HZ = ZL/(NZ+1)

!$OMP DO
      DO I = 1,NDIVP
      DO J = 1,NDIAG
      D_L(I,J) = 0.0
      ENDDO
      ENDDO
```

```
!$OMP ENDDO

      NXY = NX*NY

C OFFSET SETTING
!$OMP SINGLE
      L = 1
      IF (NDIAG_LOC .GE. 7) THEN
        OFFSET(L) = -NXY
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 5) THEN
        OFFSET(L) = -NX
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 3) THEN
        OFFSET(L) = -1
        L = L+1
      ENDIF
      OFFSET(L) = 0
      L = L+1
      IF (NDIAG_LOC .GE. 2) THEN
        OFFSET(L) = 1
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 4) THEN
        OFFSET(L) = NX
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 6) THEN
        OFFSET(L) = NXY
      ENDIF
!$OMP END SINGLE

C MAIN LOOP
!$OMP DO
      DO 100 J = 1,LEN
        JS = J

C DECOMPOSE JS-1 = (K0-1)*NX*NY+(J0-1)*NX+I0-1
        K0 = (JS-1)/NXY+1
        IF (K0 .GT. NZ) THEN
```

```
          PRINT*,'ERROR; K0.GH.NZ '
          GOTO 100
        ENDIF
        J0 = (JS-1-NXY*(K0-1))/NX+1
        I0 = JS - NXY*(K0-1) - NX*(J0-1)
        L = 1

        IF (NDIAG_LOC .GE. 7) THEN
          IF (K0 .GT. 1) D_L(J,L) = -(1.0/HZ+0.5*VA3)/HZ
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 5) THEN
          IF (J0 .GT. 1) D_L(J,L) = -(1.0/HY+0.5*VA2)/HY
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 3) THEN
          IF (I0 .GT. 1) D_L(J,L) = -(1.0/HX+0.5*VA1)/HX
          L = L+1
        ENDIF
        D_L(J,L) = 2.0/HX**2+VC
        IF (NDIAG_LOC .GE. 5) THEN
          D_L(J,L) = D_L(J,L) + 2.0/HY**2
          IF (NDIAG_LOC .GE. 7) THEN
            D_L(J,L) = D_L(J,L) + 2.0/HZ**2
          ENDIF
        ENDIF
        L = L+1
        IF (NDIAG_LOC .GE. 2) THEN
          IF (I0 .LT. NX) D_L(J,L) = -(1.0/HX-0.5*VA1)/HX
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 4) THEN
          IF (J0 .LT. NY) D_L(J,L) = -(1.0/HY-0.5*VA2)/HY
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 6) THEN
          IF (K0 .LT. NZ) D_L(J,L) = -(1.0/HZ-0.5*VA3)/HZ
        ENDIF
 100  CONTINUE
!$OMP ENDDO

!$OMP END PARALLEL
```

```
      RETURN
      END

C =======================================
* SOLUTE ERROR
* | X1 - X2 |
C =======================================
      REAL*8 FUNCTION ERRNRM(X1,X2,LEN)
      IMPLICIT  REAL*8 (A-H,O-Z)
      DIMENSION X1(*),X2(*)
C
      S = 0D0
      DO 100 I = 1,LEN
        SS = X1(I) - X2(I)
        S = S + SS * SS
 100  CONTINUE
C
      ERRNRM = SQRT( S )
      RETURN
      END
```

DM_VSRS

実スパース行列の連立 1 次方程式 (LU 分解法)
CALL DM_VSRS(A, NZ, NROW, NFCNZ, N, IPLEDSM, MZ, ISCLITERMAX, IORDERING, NPERM, ISW, NROWSYM, NFCNZSYM, B, NASSIGN, NSUPNUM, NFCNZFACTORL, PANELFACTORL, NSIZEFACTORL, NFCNZINDEXL, NPANELINDEXL, NSIZEINDEXL, NDIM, NFCNZFACTORU, PANELFACTORU, NSIZEFACTORU, NFCNZINDEXU, NPANELINDEXU, NSIZEINDEXU, NPOSTO, SCLROW, SCLCOL, EPSZ, THEPSZ, IPIVOT, ISTATIC, SPEPSZ, NFCNZPIVOT, NPIVOTP, NPIVOTQ, IREFINE, EPSR, ITERMAX, ITER, W, IW1, IW2, ICON)

(1) 機能

$n \times n$ の実スパース行列 $\boldsymbol{A}$ に対角要素に大きな要素を並べる並び換えを行い, 行および列の均衡化を行うスケーリングを行います. スーパーノードの対角ブロック内で指定した Pivot をとり, LU 分解し解きます.

$$\boldsymbol{Ax=b}$$

実スパース行列 $\boldsymbol{A}$ は以下のように変換できます.

$$\boldsymbol{A_1=D_rAP_cD_c}$$

ここで $\boldsymbol{P_c}$ は列の入れ換えを行う直交行列, $\boldsymbol{D_r}$ は行のスケーリングを行う対角行列, $\boldsymbol{D_c}$ は列のスケーリングを行う対角行列です.

$$\boldsymbol{A_2=QPA_1P^TQ^T}$$

$\boldsymbol{A_2}$ は, スーパーノードの対角ブロックで指定された pivot を行い,行および列の入れ換えを行い LU 分解されます.

ただし, $\boldsymbol{P}$ は,$\boldsymbol{SYM}= \boldsymbol{A_1+A_1^T}$ の非ゼロパターンに対して求めた ordering による行列要素の並び換えを示す置換行列で, $\boldsymbol{Q}$ は $\boldsymbol{SYM}$ に対する post order による行列要素の並び換えを示す置換行列を示します. $\boldsymbol{P}$, $\boldsymbol{Q}$ は直交行列です.

$\boldsymbol{L}$ は下三角行列, $\boldsymbol{U}$ は単位上三角行列です.

Pivot 処理で, 閾値 THEPSZ より絶対値が大きな Pivot を見つけることが出来なかったとき, 対角ブロック内の絶対値が最大の要素を Pivot の候補とします. この値が小さ過ぎるときに Static Pivot を与えて近似的に LU 分解を続けることができます.

そして LU 分解の結果を利用して, 解を求めます.

また, 解の反復改良で精度を改良することを指定できます.

(2) パラメタ

A.....入力. 実スパースな係数行列 $\boldsymbol{A}$ を圧縮列格納法で A(1:NZ)に格納します. A(NZ)なる 1 次元配列.
圧縮列格納法については,実スパース行列と実ベクトル(圧縮列格納法), DM_VMVSCC の図 DM_VMVSCC-1 を参照してください.

NZ .....入力. 実スパースな係数行列 $\boldsymbol{A}$ の非零要素の総数.

NROW.....入力. 圧縮列格納法で使用される行指標で A に格納される要素が何番目の行ベクトルに属するかを示します.
NROW(NZ)なる 1 次元配列.

NFCNZ.....入力. 行列 $\boldsymbol{A}$ の各列の非零要素を列方向に圧縮して順次配列 A に格納するとき,対応する列の最初の非零要素が格納される位置を表します.
NFCNZ(N+1)=NZ+1.
NFCNZ(N+1)なる 1 次元配列.

N.....入力. 行列 $\boldsymbol{A}$ の次数 n.

IPLEDSM.....入力. 対角要素に大きな要素を並べる列の入れ換えを行うかを指定します.
1 のとき, 列の入れ換えを求めて入れ換えます.
1 以外のとき, 列の入れ換えを行いません.

MZ.....出力. IPLEDSM に 1 が指定されたとき, 列の入れ換えを表す. MZ(i)=j は行列 $\boldsymbol{a_{ij}}$ の属する j 番目の列を i 番目の列に移動する. $\boldsymbol{a_{ij}}$ は対角要素に並べる大きな要素を表す.
MZ(N)なる 1 次元配列.

ISCLITERMAX....入力. 係数行列のスケーリングを行う対角行列 $\boldsymbol{D_r}$ と $\boldsymbol{D_c}$ を反復して求める反復回数.
ISCLITERMAX ≤ 0 のときスケーリングは行わすに, $\boldsymbol{D_r}$ および $\boldsymbol{D_c}$ は単位行列が設定されます.
ISCLITERMAX ≥ 10 のときは 10 回を上限値とします.

IORDERING.....入力. ordering を表す置換行列 $\boldsymbol{P}$ で $\boldsymbol{PA_1P^T}$ と変換した行列を LU 分解するかを指定します. 置換行列は直交行列です.
10 のとき, ISW=1 で呼び出すと ordering を求める $\boldsymbol{A_1}$ に関する情報を求めて NROWSYM, NFCNZSYM に設定します.
11 のとき, ISW=1 で 10 を指定して呼び出して得た行列を対称化した情報 NROWSYM, NFCNZSYM を使って決めた ordering を NPERM に指定して同じく ISW=1 で呼び出し, 計算を続けることを示します. $\boldsymbol{PA_1P^T}$ を LU 分解する処理を続けます.
10,11 以外のとき, ordering は指定せずに行列 $\boldsymbol{A_1}$ をそのまま LU 分解します.
出力. ISW=1 と IORDERING=10 を指定して呼び出した後, IORDERING に 11 が設定されます. そのため, ordering を NPERM に指定して再び呼び出すときに特に 11 を指定する必要はありません.
((3)使用上の注意 a. 注意①参照).

NPERM.....入力. IORDERING=11 のとき使用する置換行列をベクトルで指定します.

NPERM(N)なる 1 次元配列.
((3)使用上の注意 a. 注意①参照).

ISW ..... 入力. 呼び出しに関する制御情報を示します.
3) 1 のとき
対称化および symbolic decomposition を行い, 必要な領域が割り当てられているか調べ計算を行います.
IORDERING=10 で呼び出し, ordering を求めるための情報が NROWSYM, NFCNZSYM に出力されます. これらを使って ***SYM*** に対する ordering を求めた後, NPERM に指定して IORDERING=11 を指定してもう 1 回 ISW=1 で呼び出します.
IORDERING が 10,11 以外のときは ordering は行ないません.
4) 2 のとき
ISW=1 で呼び出したとき, PANELFACTORL, PANELFACTORU, NPANELINDEXL または NPANELINDEXU の大きさが不足して ICON=31000 で終了した処理を継続します. このとき, NSIZEFACTORL , NSIZEFACTORU, NSIZEINDEXL または NSIZEINDEXU に返却された必要な大きさで PANELFACTORL, PANELFACTORU, NPANELINDEXL または NPANELINDEXU を確保し直して指定しなおして再度呼び出します.
これらの引数と実行を制御する引数 ISW 以外の引数に格納されている値は変更してはいけません.
3) 3 のとき
先立つ呼び出しで LU 分解された同じ係数行列に対する連立 1 次方程式の右辺ベクトル ***b*** を変えて再度解くことを指定します. このとき先立つ呼び出しで LU 分解した結果を使います.
実行を制御する引数 ISW と右辺ベクトル ***b*** を格納する引数 B 以外の引数に格納されている値を変更していけません.

NROWSYM ..... 出力. IORDERING=10 で呼び出したとき, 対称化した $\boldsymbol{SYM}= \boldsymbol{A_1}+\boldsymbol{A_1}^T$ の非ゼロパターンの下三角行列部分の行指標を列圧縮したものが返却されます.
IPLEDSM=1 のときは, $\boldsymbol{A_1}=\boldsymbol{D_r}\boldsymbol{A}\boldsymbol{P_c}\boldsymbol{D_c}$ です.
NROWSYM(NZ+N)なる 1 次元配列.

NFCNZSYM ..... 出力. IORDERING=10 で呼び出したとき, 行列 SYM の下三角部分の各列の非零要素の行指標を列方向に圧縮して順次配列 NROWSYM に格納するとき,対応する列の最初の非零要素が格納される位置を表します.
NFCNZSYM(N+1)=NSYMZ+1 NSYMZ は,総要素数を表します.
NFCNZSYM(N+1)なる 1 次元配列.

B ..... 入力. $\boldsymbol{Ax}=\boldsymbol{b}$ の右辺定数ベクトル.
出力. 解ベクトル.
B(N)なる 1 次元配列.

NASSIGN..... 出力. 各 supernode に対する $\boldsymbol{L}$, $\boldsymbol{U}$ は圧縮して 2 次元の panel に格納します. この panel を PANELFACTORL および PANELFACTORU の 1 次元部分

配列として順番に割り付けたとき何番目になるかを示します. これらの指標ベクトルも同じように NPANELINDEXL, NPANELINDEXU に割り付けます. j=NASSIGN(i)のとき, i 番目の supernode を j 番目に割り付けたことを示します.
入力. ISW≠1 のとき, 初回呼び出しの値を再利用します.
分解結果の格納方法については図 DM_VSRS-1 を参照してください.
NASSIGN(N)なる 1 次元配列.

NSUPNUM ..... 出力. supernode の総数.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します. (≤n)

NFCNZFACTORL 出力. 実スパース行列の LU 分解の結果の行列 $\boldsymbol{L}$ および $\boldsymbol{U}$ はスーパーノードに対応しておのおの求めます. 各 supernode に対応する L の列ベクトルは, 共通の行指標ベクトルを持つように圧縮してブロック対角部分に $\boldsymbol{U}$ の対応部分を含めて 2 次元の panel に格納します. この panel を PANELFACTORL の 1 次元部分配列として順番に割り付けたとき, i 番目の panel の先頭要素 panel(1,1)が 1 次元配列の PANELFACTORL の何番目の要素になるかを示します.
NFCNZFACTORL(N+1)なる 8 バイト整数型の 1 次元配列.
結果の格納方法については図 DM_VSRS-1 を参照してください.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.

PANELFACTORL 出力. 各 supernode の分解結果に対応する行列 $\boldsymbol{L}$ の複数の列ベクトルと行列 $\boldsymbol{U}$ のブロック対角部分について，共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します. panel を順番に割り付けます. i 番目の supernode に対応する panel が何番目の位置に割り当てられるかは j=NASSIGN(i)から分かります. その先頭位置は NFCNZFACTORL(j)に格納されています. panel ごとに分解結果が格納されます.
i 番目に割付けられた panel の大きさは DIM(1,i)×DIM(2,i)の 2 次元配列と見なせます. i 番目の panel の panel(s,t), $s \geq t$, s=1,…,DIM(1,i), t=1,…,DIM(2,i)に下三角行列 $\boldsymbol{L}$ が格納されます. panel の panel(s,t), $s < t$, t=1,…,DIM(2,i)には単位上三角行列 $\boldsymbol{U}$ の対角要素を除いたブロック対角部分が格納されます.
PANELFACTORL(NSIZEFACTORL)なる 1 次元配列.
結果の格納方法については図 DM_VSRS-1 を参照してください.
((3)使用上の注意 a. 注意③参照).

NSIZEFACTORL . 入力. PANELFACTORL の大きさを示す. 8 バイト整数型.
出力. PANELFACTORL の大きさとして必要な大きさが返却されます.
((3)使用上の注意 a. 注意③参照).

NFCNZINDEXL ... 出力. 各 supernode の分解結果に対応する行列 $\boldsymbol{L}$ の複数の列ベクトルと行列 $\boldsymbol{U}$ のブロック対角部分について，共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します.この行指標ベクトルを NPANELINDEXL に順番に割り付けたとき i 番目の行指標ベクトルの先頭要素が 1 次元配列である NPANELINDEXL の何番目の要素になるかを示します.

NFCNZINDEXL(N+1)なる 8 バイト整数型の 1 次元配列.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
結果の格納方法については図 DM_VSRS-1 を参照してください.

NPANELINDEXL 出力. 各 supernode の分解結果に対応する行列 ***L*** の複数の列ベクトルと行列 ***U*** のブロック対角部分について，共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します. この行指標ベクトルを NPANELINDEXL に順番に割り付けます. i 番目の supernode に対応する panel の行の指標ベクトルが何番目に割り当てられるかは j=NASSIGN(i)から分かります. その先頭位置は NFCNZINDEXL(j)に格納されています.
この行指標は行列 ***SYM*** に対する post order で並び換えたときの行の番号です.
NPANELINDEXL(NSIZEINDEXL)なる 1 次元配列.
結果の格納方法については図 DM_VSRS-1 を参照してください.
((3)使用上の注意 a. 注意③参照).

NSIZEINDEXL..... 入力. NPANELINDEXL の大きさを示す. 8 バイト整数型.
出力. 必要な大きさが返却されます.
((3)使用上の注意 a. 注意③参照).

NDIM ..... 出力. NDIM(1,i)および NDIM(2,i)は行列 ***L*** に関して i 番目に割り付けられた panel の 1 次元目と 2 次元目の大きさを示します.
NDIM(3,i)は行列 ***U*** に関して割り付けられた panel を転置したときの 1 次元目の大きさに対角ブロックの大きさを加えた大きさを示します.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
NDIM(3,N)なる 2 次元配列.
結果の格納方法については図 DM_VSRS-1 を参照してください.

NFCNZFACTORU 出力. 実スパース行列の LU 分解の結果の行列 ***U*** に関しては, 各 supernode に対応する ***U*** の行ベクトルは, ブロック対角部分を除いて共通の列指標ベクトルを持つように圧縮し, 転置したものを 2 次元の panel に格納します. この panel を PANELFACTORU の 1 次元部分配列として順番に割り付けたとき, i 番目の panel の先頭要素 panel(1,1)が 1 次元配列の PANELFACTORU の何番目の要素になるかを示します.
NFCNZFACTORU(N+1)なる 8 バイト整数型の 1 次元配列.
結果の格納方法については図 DM_VSRS-1 を参照してください.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.

PANELFACTORU 出力. 各 supernode の分解結果に対応する行列 ***U*** のブロック対角部分を除いた複数の行ベクトルについて，共通の列指標ベクトルを持つように圧縮し, 転置して 2 次元の panel に格納します. panel を順番に割り付けます. i 番目の supernode に対応する panel が何番目の位置に割り当てられるかは j=NASSIGN(i)から分かります. その先頭位置は NFCNZFACTORU(j)に格納されています. panel ごとに分解結果が格納されます.
i 番目に割付けられた panel の大きさは{DIM(3,i)-DIM(2,i)}×DIM(2,i)の 2

次元配列と見なせます. i 番目の panel の panel(s,t), s=1,…,DIM(3,i)-DIM(2,i), t=1,…,DIM(2,i)に単位上三角行列 $\boldsymbol{U}$ のブロック対角部分を除いた部分が行ベクトルを圧縮し, 転置して格納します.
PANELFACTORU(NSIZEFACTORU)なる 1 次元配列.
結果の格納方法については図 DM_VSRS-1 を参照してください.
((3)使用上の注意 a. 注意③参照).

NSIZEFACTORU. 入力. PANELFACTORU の大きさを示す. 8 バイト整数型.
出力. PANELFACTORU の大きさとして必要な大きさが返却されます.
((3)使用上の注意 a. 注意③参照).

NFCNZINDEXU... 出力. 各 supernode の分解結果に対応する行列 $\boldsymbol{U}$ は対角ブロック部分を除いて複数の行ベクトルについて, 共通の列指標ベクトルを持つように圧縮し, 転置して 2 次元の panel に格納します. 対角ブロック部分も含んだこの列指標ベクトルを NPANELINDEXU に順番に割り付けたとき i 番目の列指標ベクトルの先頭要素が 1 次元配列である NPANELINDEXU の何番目の要素になるかを示します.
NFCNZINDEXU(N+1)なる 8 バイト整数型の 1 次元配列.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
結果の格納方法については図 DM_VSRS-1 を参照してください.

NPANELINDEXU 出力. 各 supernode の分解結果に対応する行列 $\boldsymbol{U}$ の対角ブロックを除いた部分を転置して圧縮して 2 次元の panel に格納します. この列指標ベクトルを対角ブロック部分を含めて NPANELINDEXU に順番に割り付けます. i 番目の supernode に対応する panel の行の指標ベクトルが何番目に割り当てられるかは j=NASSIGN(i)から分かります. その先頭位置は NFCNZINDEXU(j)に格納されています.
この行指標は行列 $\boldsymbol{SYM}$ に対する post order で並び換えたときの列の番号です.
NPANELINDEXU(NSIZEINDEXU)なる 1 次元配列.
結果の格納方法については図 DM_VSRS-1 を参照してください.
((3)使用上の注意 a. 注意③参照).

NSIZEINDEXU .... 入力. NPANELINDEXU の大きさを示す. 8 バイト整数型.
出力. 必要な大きさが返却されます.
((3)使用上の注意 a. 注意③参照).

NPOSTO ..... 出力. post order で i 番目の node が行列 $\boldsymbol{A}$ の何番目の列番号に対応するかを表す 1 次元ベクトル.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
NPOSTO(N)なる 1 次元配列.
((3)使用上の注意 a. 注意④参照).

SCLROW ..... 出力. スケーリング対角行列 $\boldsymbol{D_r}$. 対角要素が 1 次元配列に格納されます.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
SCLROW(N)なる 1 次元配列.

SCLCOL..... 出力. スケーリング対角行列 $\boldsymbol{D}_c$. 対角要素が 1 次元配列に格納されます.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
SCLCOL(N)なる 1 次元配列.

EPSZ ..... 入力. 分解過程で対角要素の大きさの絶対値の相対判定値
出力. EPSZ ≤ 0.0 のときは標準値が設定されます.
((3)使用上の注意 a. 注意②参照).

THEPSZ ..... 入力. ピボットの判定での閾値(threshold). これより大きな値はピボットとして採用します. 見つかればその時点で, ピボット検索は打ち切ります.
例えば 1.0D-2 程度.
出力. THEPSZ ≤ 0.0 のときは 1.0D-2 が設定されます.
EPSZ ≥ THEPSZ > 0.0 のときは, EPSZ が設定されます.

IPIVOT..... 入力. スーパーノード内でピボットを行うか, 行う場合どのようなピボットを行うかを指定します. 例えば, 完全ピボットとして 40 を指定.
IPIVOT < 10 : または IPIVOT ≥ 50: ピボットなし.
10 ≤ IPIVOT < 20 : 部分ピボット
20 ≤ IPIVOT < 30 : 対角ピボット
21 : スーパーノード内で対角ピボットがとれないとき, Rook ピボットに変更します.
22 : スーパーノード内で対角ピボットがとれないときは, Rook ピボットに, Rook ピボットが取れないときは完全ピボットに変更します.
30 ≤ IPIVOT < 40 :Rook ピボット
32 : スーパーノード内で Rook ピボットがとれないとき, 完全ピボットをとります.
40 ≤ IPIVOT < 50 : 完全ピボット

ISTATIC..... 入力. Pivot を指定したとき, Static Pivot を行うかを示します.
1) = 1 のとき
スーパーノード内で指定したピボットが SPEPSZ より大きくないとき, ピボットとして DSIGN(SPEPSZ,PIVOT)で近似します. ピボットの値が 0.0d0 なら SPEPSZ に近似します.
このとき, 以下を設定しなければなりません.
a) EPSZ は EPSZ の標準値以下にしなければなりません.
b) ISCLITERMAX は 10 を設定しスケーリングを行う必要があります.
c)THEPSZ ≥ SPEPSZ でなければなりません.
d) 解の反復改良を行うため IREFINE=1 を指定しなければなりません.
2) ≠1 のとき
Static Pivot は行いません.

SPEPSZ ..... 入力. ISTATIC=1 のとき, Statc Pivot として使う値.
1.0D-10 ≥ SPEPSZ ≥ EPSZ でなければなりません.
出力. SPEPSZ<EPSZ のときは, 1.0D-10 が設定されます.

NFCNZPIVOT ..... 出力. スーパーノード内の相対的な位置でのピボットの行, 列の入れ換えの履歴を格納する位置を示す. i 番目の supernode に関する情報が何番

目の位置に割り当てられるかは j=NASSIGN(i)から分かります. その先頭位置は NFCNZPIVOT(j)に格納されています. i 番目のスーパーノードの入れ換え情報は, NPIVOTP, NPIVOTQ の is=NFCNZPIVOT(j) ,…, ie=NFCNZPIVOT(j)+NDIM(2,j)-1 番目の要素に格納されます.
NFCNZPIVOT(NSUPNUM+1)なる 1 次元配列.

NPIVOTP ..... 出力. スーパーノード内の行の入れ換えに関する情報を格納する.
NPIVOTP(N)なる 1 次元配列.

NPIVOTQ ..... 出力. スーパーノード内の列の入れ換えに関する情報を格納する.
NPIVOTQ(N)なる 1 次元配列.

IREFINE..... 入力. LU 分解結果を利用して解を求めるときに, 解の反復改良を行うかどうかを示す. 残差ベクトルを計算するときに 4 倍精度で計算します.
=1 : 解の反復改良を行う. 反復改良して得られる残差ベクトル $\boldsymbol{r}_k$ の絶対値の差分がひとつ前の差分の 1/4 より大きくなるまで, 反復改良を行います.
≠1: 解の反復改良を行わない.
ISTATIC=1 のとき, IREFINE=1 でなければなりません.

EPSR ..... 入力. 解の残差ベクトル $\boldsymbol{b}-\boldsymbol{Ax}$ の絶対値が, $\boldsymbol{b}$ の絶対値に対して十分小さいかを判定する判定値.
EPSR ≤ 0.0 のとき 1.0D-6 が設定されます.

ITERMAX..... 入力. (≥ 1) 反復改良を行うときの最大反復回数.

ITER..... 出力. 反復改良を行った回数.

W ..... 作業域.
出力/入力.
大きさ 4*NZ+6*N の 1 次元配列.
ISW=1, 2 で続けて呼び出すとき,呼び出しの間で必要なデータを受け渡すために使われます. 呼び出しの間で値を変更してはいけません.

IW1..... 作業域.
出力/入力.
大きさ 2*NZ+2*(N+1)+16*N の 1 次元配列.
ISW=1, 2 で続けて呼び出すとき,呼び出しの間で必要なデータを受け渡すために使われます. 呼び出しの間で値を変更してはいけません.

IW2..... 作業域.
出力/入力.
大きさ 47*N+47+NZ+4*(N+1)+2*(NZ+N)の 1 次元配列.
ISW=1, 2, 3 で続けて呼び出すとき,呼び出しの間で必要なデータを受け渡すために使われます. 呼び出しの間で値を変更してはいけません.

ICON ..... 出力. コンディションコード.
表 DM_VSRS-1 参照

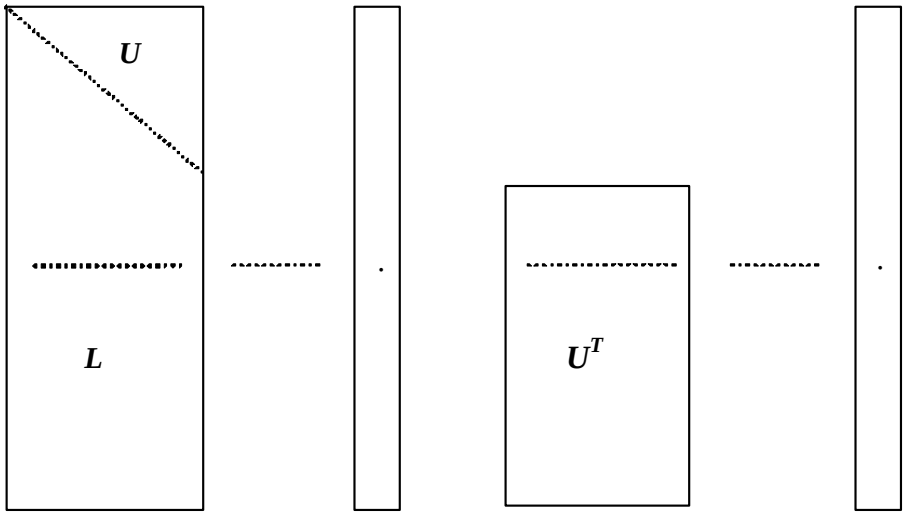


図 DM_VSRS-1 分解結果の格納概念図

j = NASSIGN(i) → i 番目の supernode は j 番目に格納されます.

p = NFCNZFACTORL(j) → j 番目の panel は PANELFACTORL の p 番目の要素から DIM(1,j)×DIM(2,j)の長さを占めます.

q = NFCNZINDEXL(j) → j 番目の panel の行指標を表すベクトルは NPANELINDEXL の q 番目の要素から DIM(1,j)の長さを占めます.

panel は大きさ DIM(1,j)×DIM(2,j)の配列と見なせます.

panel(s,t) , $s \geq t$, s = 1 , … , DIM(1,j), t=1 , … , DIM(2,j) に分解結果の下三角行列 $\boldsymbol{L}$ が格納されます.

panel(s,t) , $s < t$, s=1 , … , DIM(2,j), t=1 , … , DIM(2,j) に単位上三角行列 $\boldsymbol{U}$ の対角ブロック部分が対角要素を除き格納されます.

u = NFCNZFACTORU(j) → j 番目の panel は PANELFACTORU の u 番目の要素から (DIM(3,j)-DIM(2,j))×DIM(2,j)の長さを占めます.

v = NFCNZINDEXU(j) → j 番目の panel の列指標を表すベクトルは NPANELINDEXU の v 番目の要素から DIM(3,j)の長さを占めます.

panel は大きさ (DIM(3,j)-DIM(2,j))×DIM(2,j)の配列と見なせます.

panel(x,y) , x=1 , … , DIM(3,j)-DIM(2,j), y=1 , … , DIM(2,j) に分解結果の単位上三角行列 $\boldsymbol{U}$ の転置行列 $\boldsymbol{U^T}$ の対角ブロック部分を除いた部分が格納されます.

指標の値は行列 $\boldsymbol{A}$ の列番号をもつ node を post order で並び換えた $\boldsymbol{QAQ^T}$ で の列番号を表します.

表 DM_VSRS-1 コンディションコード

コード	意　　味	処理内容
0	エラーなし	ー
20000	ピボットが相対的に零になりました.	処理を打ち切ります.
20100	IPLEDSM=1 を指定して,対角要素に絶対値が大きな要素を並べるため maximum matching を求める処理で, 長さ N の maximum matching がみつかりませんでした.行列が特異である可能性があります.	
20200	行および列の均衡化を行う対角行列の対角要素を求める過程で,元の行列 A の行もしくは列にゼロベクトルがありました. 行列が特異である可能性があります.	
20400	LU 分解された行列の対角要素にゼロがあります.	
20500	求めた解ベクトルに対する残差ベクトルのノルムの大きさが方程式 Ax=b の右辺ベクトルのノルムの EPSR 倍より大きい. 係数行列は特異に近い可能性があります.	
30000	N<1,NZ<0,NFCNZ(N+1)≠NZ+1, NSIZEFACTORL<1, NSIZEFACTORU<1, NSIZEINDEXL<1, NSIZEINDEXU<1, ISW<1, ISW>3, IREFINE=1 のとき ITERMAX<1,	
30100	NPERM に指定した置換行列が正しくない.	
30200	NROW(j)に格納された i 列目の行指標 k が k<1 または k>N	
30300	i 列目の行指標の数 NFCNZ(i+1)-NFCNZ(i)>n	
30500	ISTATIC=1 のとき満たすべき条件をみたしていない. EPSZ は標準値 16***u*** より大きかった. または ISCLITERMAX <10 であった. または IREFINE≠1 であった. または SPEPSZ>THEPSZ であった. または SPEPSZ>1.0D-10 であった.	
31000	PANELFACTORL の大きさ NSIZEFACTORL または NPANELINDEXL の大きさ NSIZEINDEXL または PANELFACTORU の大きさ NSIZEFACTORU または NPANELINDEXU の大きさ NSIZEINDEXU が小さすぎます.	NSIZEFACTORL または NSIZEINDEXL または NSIZEFACTORU または NSIZEINDEXU で指定された大きさで PANELFACTORL または NPANELINDEXL または PANELFACTORU または NPANELINDEXU を割り付けて ISW=2 として再度呼び出す.

(3)　使用上の注意

a.　注意

①　直交行列である置換行列$\boldsymbol{P}$の要素$\boldsymbol{p_{ij}=1}$のとき, NPERM(i)=jと表現します.
逆は以下のようにして求めることができます.

```
DO i=1,n
  j=NPERM(i)
  NPERMINV(j)=i
ENDDO
```

Fill-Reducing OrderingはMETISなどを使って求めることができます.
詳細は“付録1 参考文献一覧表”の[43],[44]を参照願います.

②　分解過程で対角要素の大きさの絶対値の相対零判定値にある値を設定したとすると, この値は次の意味を持っています. すなわち, LU分解の過程で, 対角要素の大きさの絶対値がその値より小さくなった場合に, その値を相対的に零と見なし, ICON=20000として処理を打ち切ります. EPSZの標準値は, 丸め誤差の単位をuとしたとき, EPSZ=16uです.
なお, 対角要素の大きさの絶対値が小さくなっても, 処理を続行させたい場合には, EPSZに極小の値を与えれば良いのですが, 結果の精度は保証されません.
Static Pivotを指定した場合, 対角要素がSPEPSZより小さいとき, SPEPSZで近似してLU分解を行います. このとき, 解の反復改良を指定しなければなりません.

③　分解結果を格納する配列PANELFACTORL, NPANELINDEXL, PANELFACTORU, NPANELINDEXUの必要な大きさは,事前には分かりません. 十分大きな配列を割り当てるか,本ルーチンを呼び出してsymbolic decompositionを行った解析の結果を使って割り当てを行うことができます.
例えば,大きさ1の1次元配列などを割付けます.そしてその大きさ1などの小さな値をNSIZEFACTORL, NSIZEINDEXL, NSIZEFACTORU, NSIZEINDEXUに指定して, ISW=1で呼び出します.
symbolic decompositionを行い, ICON=31000で終了し, NSIZEFACTORL, NSIZEINDEXL, NSIZEFACTORU, NSIZEINDEXUに必要な大きさが返却されます. 必要な大きさの配列を割付け直して引数に指定して, ISW=2で呼び出すことで, symbolic decomposition以降の処理を続けることができます. 使用例を参照願います.

④　列番号に対応するノードを考えます.これをpost orderで順番を並び換えたときの番号の対応関係がNPOSTOに格納されています. post orderのi番目のノードが並び換える前の何番目のノードに対応するかを表します. j=nposto(i)はj番目であることを表します.
①同様にこれは直交行列である置換行列$\boldsymbol{Q}$を表し,行列$\boldsymbol{A}$を$\boldsymbol{QAQ^T}$と並び換えることに相当します.
逆変換$\boldsymbol{Q^T}$は以下のようにして求めることができます.

```
DO i=1,n
  j=NPOSTO(i)
  NPOSTOINV(j)=i
  ENDDO
```

⑤　連立1次方程式$\boldsymbol{Ax}=\boldsymbol{b}$は, DM_VSRLUで実スパース行列$\boldsymbol{A}$をLU分解して, 分解結果を利用して引き続いてDM_VSRLUXを呼び出して解くこともできます.

b. 使用例

連立 1 次方程式 $\boldsymbol{Ax} = \boldsymbol{f}$ を解きます.

行列は境界条件として立方体の境界で 0 となる楕円型の偏微分方程式に有限差分法を適用して生成されたものを利用します.

$$-\Delta u + a\nabla u + cu = f$$

ここで $a = (a_1, a_2, a_3)$.

行列はサブルーチン init_mat_diag によって生成されます.

対角形式格納法で生成し, 対角ベクトルの下 6 本を圧縮列格納法で格納します.

結果生成された非対称な行列 $\boldsymbol{A}$ に関する連立 1 次方程式を解きます.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT  REAL*8 (A-H,O-Z)
      PARAMETER (NORD=40,KX = NORD,KY =NORD ,KZ = NORD,
     $      N = KX*KY*KZ)
      PARAMETER (NBORDER=N+1,NOFFDIAG=6)
      PARAMETER (K = N+1)
      PARAMETER (NDIAG = 7)
      INTEGER*4 WL
      PARAMETER (NALL=NDIAG*N,
C
     $   WL  =4*NALL+6*N,
     $   IW1L=2*NALL+2*(N+1)+16*N,
     $   IW2L=47*N+47+4*(N+1)+NALL+2*(NALL+N))
C
      DIMENSION NOFST(NDIAG)
      DIMENSION DIAG(K,NDIAG),DIAG2(K,NDIAG)
      DIMENSION A(K*NDIAG),NROW(K*NDIAG),NFCNZ(N+1),
     $          NROWSYM(K*NDIAG+N),NFCNZSYM(N+1),
     $
     $          WC(K*NDIAG),IWC(2,K*NDIAG)
      DIMENSION NPERM(N),W(WL),
     $          NPOSTO(N),NDIM(3,N),
     $          NASSIGN(N),
     $          MZ(N),
     $          IW1(IW1L),IW2(IW2L)
```

```
      REAL*8, DIMENSION(:), ALLOCATABLE ::  PANELFACTORL,PANELFACTORU
      INTEGER*4, DIMENSION(:), ALLOCATABLE :: NPANELINDEXL,NPANELINDEXU
      REAL*8 DUMMYFL,DUMMYFU
      INTEGER*4 NDUMMYIL,
     $          NDUMMYIU
      INTEGER*8 NSIZEFACTORL,
     $          NSIZEINDEXL,
     $          NSIZEINDEXU,
     $          NSIZEFACTORU,
     $          NFCNZFACTORL(N+1),
     $          NFCNZFACTORU(N+1),
     $          NFCNZINDEXL(N+1),
     $          NFCNZINDEXU(N+1)
      DIMENSION B(N),SOLEX(N)
      REAL*8 EPSZ,THEPSZ,SPEPSZ,
     $       SCLROW(N),SCLCOL(N)
C
      INTEGER*4     IPIVOT,ISTATIC,NFCNZPIVOT(N+1),
     $              NPIVOTP(N),NPIVOTQ(N),
     $              IREFINE,ITERMAX,ITER,IPLEDSM
C
      PRINT *,'    LU DECOMPOSITION METHOD'
      PRINT *,'    FOR SPARSE UNSYMMETRIC REAL MATRICES'
      PRINT *,'    IN COMPRESSED COLUMN STORAGE'
      PRINT *
C
      DO I=1,N
      SOLEX(I)=DBLE(1)
      ENDDO
      PRINT *,'    EXPECTED SOLUTIONS'
      PRINT *,'    X(1) = ',SOLEX(1),' X(N) = ',SOLEX(N)
      PRINT *
C
      VA1 = 1.0D0
      VA2 = 2.0D0
      VA3 = 3.0D0
      VC =  4.0D0
      XL = 1.0
      YL = 1.0
      ZL = 1.0
      CALL INIT_MAT_DIAG(VA1,VA2,VA3,VC,DIAG,NOFST
     &          ,KX,KY,KZ,XL,YL,ZL,NDIAG,N,K)
```

```
C
      DIAG2=0
C
      DO I=1,NDIAG
C
      IF(NOFST(I).LT.0)THEN
      NBASE=-NOFST(I)
      LENGTH=N-NBASE
      DIAG2(1:LENGTH,I)=DIAG(NBASE+1:N,I)
      ELSE
      NBASE=NOFST(I)
      LENGTH=N-NBASE
      DIAG2(NBASE+1:N,I)=DIAG(1:LENGTH,I)
      ENDIF
C
      ENDDO
C
      NUMNZ=1
C
      DO J=1,N
      NTOPCFG=1
C
      DO I=NDIAG,1,-1
C
      IF(NTOPCFG.EQ.1)THEN
      NFCNZ(J)=NUMNZ
      NTOPCFG=0
      ENDIF
C
      IF(J.LT.NBORDER.AND.I.GT.NOFFDIAG)THEN
      CONTINUE
      ELSE
C
      IF(DIAG2(J,I).NE.0.0D0)THEN
C
      NCOL=J-NOFST(I)
      A(NUMNZ)=DIAG2(J,I)
      NROW(NUMNZ)=NCOL
C
      NUMNZ=NUMNZ+1
C
      ENDIF
```

```
      ENDIF
      ENDDO
      ENDDO
C
      NFCNZ(N+1)=NUMNZ
      NZ=NUMNZ-1
C
      CALL DM_VMVSCC(A,NZ,NROW,NFCNZ,N,SOLEX,
     $             B,WC,IWC,ICON)
C
C     INITIAL CALL WITH IORDER=1
C
      IORDERING= 0            !
      IPLEDSM=1
      ISCLITERMAX=10
      ISW=1
      EPSZ=1.0D-16
      NSIZEFACTORL=1
      NSIZEFACTORU=1
      NSIZEINDEXL=1
      NSIZEINDEXU=1
      THEPSZ=1.0D-2
      SPEPSZ=0.0D0
      IPIVOT=40
      ISTATIC=0
      IREFINE=1
      EPSR=0.0D0
      ITERMAX=10
C
      CALL DM_VSRS(A,NZ,NROW,NFCNZ,N,
     $              IPLEDSM,MZ,ISCLITERMAX,IORDERING,
     $              NPERM,ISW,
     $              NROWSYM,NFCNZSYM,
     $              B,
     $              NASSIGN,
     $              NSUPNUM,
     $              NFCNZFACTORL,DUMMYFL,
     $              NSIZEFACTORL,
     $              NFCNZINDEXL,
     $              NDUMMYIL,NSIZEINDEXL,
     $              NDIM,
     $              NFCNZFACTORU,DUMMYFU,
```

```
     $              NSIZEFACTORU,
     $              NFCNZINDEXU,
     $              NDUMMYIU,NSIZEINDEXU,
     $              NPOSTO,
     $              SCLROW,SCLCOL,
     $              EPSZ,THEPSZ,
     $              IPIVOT,ISTATIC,SPEPSZ,NFCNZPIVOT,
     $              NPIVOTP,NPIVOTQ,
     $              IREFINE,EPSR,ITERMAX,ITER,
     $              W,IW1,IW2,ICON)
C
      PRINT*,'ICON=',ICON,' NSIZEFACTORL=',NSIZEFACTORL,
     $       ' NSIZEFACTORU=',NSIZEFACTORU,
     $       'NSIZEINDEXL=',NSIZEINDEXL,
     $       'NSIZEINDEXU=',NSIZEINDEXU,
     $       'NSUPNUM=',NSUPNUM
C
      ALLOCATE( PANELFACTORL(NSIZEFACTORL) )
      ALLOCATE( PANELFACTORU(NSIZEFACTORU) )
      ALLOCATE( NPANELINDEXL(NSIZEINDEXL) )
      ALLOCATE( NPANELINDEXU(NSIZEINDEXU) )
C
      ISW=2
C
      CALL DM_VSRS(A,NZ,NROW,NFCNZ,N,
     $              IPLEDSM,MZ,ISCLITERMAX,IORDERING,
     $              NPERM,ISW,
     $              NROWSYM,NFCNZSYM,
     $              B,
     $              NASSIGN,
     $              NSUPNUM,
     $              NFCNZFACTORL,PANELFACTORL,
     $              NSIZEFACTORL,
     $              NFCNZINDEXL,
     $              NPANELINDEXL,NSIZEINDEXL,
     $              NDIM,
     $              NFCNZFACTORU,PANELFACTORU,
     $              NSIZEFACTORU,
     $              NFCNZINDEXU,
     $              NPANELINDEXU,NSIZEINDEXU,
     $              NPOSTO,
     $              SCLROW,SCLCOL,
```

```
     $              EPSZ,THEPSZ,
     $              IPIVOT,ISTATIC,SPEPSZ,NFCNZPIVOT,
     $              NPIVOTP,NPIVOTQ,
     $              IREFINE,EPSR,ITERMAX,ITER,
     $              W,IW1,IW2,ICON)
C
      ERR = ERRNRM(SOLEX,B,N)
C
      PRINT *,'    COMPUTED VALUES'
      PRINT *,'    X(1) = ',B(1),' X(N) = ',B(N)
      PRINT *
      PRINT *,'    ICON = ',ICON
      PRINT *
      PRINT *,'    N = ',N
      PRINT *
      PRINT *,'    ERROR = ',ERR
      PRINT *,'    ITER=',ITER
      PRINT *
      PRINT *
C
      IF(ERR.LT.1.0D-8.AND.ICON.EQ.0)THEN
         WRITE(*,*)'********** OK **********'
      ELSE
         WRITE(*,*)'********** NG **********'
      ENDIF
C

      DEALLOCATE( PANELFACTORL,PANELFACTORU,
     $            NPANELINDEXL,
     $            NPANELINDEXU )
C
      STOP
      END

C =======================================
C     INITIALIZE COEFFICIENT MATRIX
C =======================================
      SUBROUTINE INIT_MAT_DIAG(VA1,VA2,VA3,VC,D_L,OFFSET
     &               ,NX,NY,NZ,XL,YL,ZL,NDIAG,LEN,NDIVP)
      IMPLICIT  REAL*8(A-H,O-Z)
      DIMENSION D_L(NDIVP,NDIAG)
      INTEGER   OFFSET(NDIAG)
```

```
C
      IF (NDIAG .LT. 1) THEN
        WRITE (*,*) 'SUBROUTINE INIT_MAT_DIAG:'
        WRITE (*,*) ' NDIAG SHOULD BE GREATER THAN OR EQUAL TO 1'
        RETURN
      ENDIF

!$OMP PARALLEL DEFAULT(PRIVATE)
!$OMP+     SHARED(VA1,VA2,VA3,VC,D_L,OFFSET
!$OMP+      ,NX,NY,NZ,XL,YL,ZL,NDIAG,LEN,NDIVP)

C NDIAG CANNOT BE GREATER THAN 7
      NDIAG_LOC = NDIAG
      IF (NDIAG .GT. 7) NDIAG_LOC = 7

C INITIAL SETTING
      HX = XL/(NX+1)
      HY = YL/(NY+1)
      HZ = ZL/(NZ+1)

!$OMP DO
      DO I = 1,NDIVP
      DO J = 1,NDIAG
      D_L(I,J) = 0.0
      ENDDO
      ENDDO
!$OMP ENDDO

      NXY = NX*NY

C OFFSET SETTING
!$OMP SINGLE
      L = 1
      IF (NDIAG_LOC .GE. 7) THEN
        OFFSET(L) = -NXY
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 5) THEN
        OFFSET(L) = -NX
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 3) THEN
```

```
        OFFSET(L) = -1
        L = L+1
      ENDIF
      OFFSET(L) = 0
      L = L+1
      IF (NDIAG_LOC .GE. 2) THEN
        OFFSET(L) = 1
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 4) THEN
        OFFSET(L) = NX
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 6) THEN
        OFFSET(L) = NXY
      ENDIF
!$OMP END SINGLE

C MAIN LOOP
!$OMP DO
      DO 100 J = 1,LEN
        JS = J

C DECOMPOSE JS-1 = (K0-1)*NX*NY+(J0-1)*NX+I0-1
        K0 = (JS-1)/NXY+1
        IF (K0 .GT. NZ) THEN
          PRINT*,'ERROR; K0.GH.NZ '
          GOTO 100
        ENDIF
        J0 = (JS-1-NXY*(K0-1))/NX+1
        I0 = JS - NXY*(K0-1) - NX*(J0-1)
        L = 1

        IF (NDIAG_LOC .GE. 7) THEN
          IF (K0 .GT. 1) D_L(J,L) = -(1.0/HZ+0.5*VA3)/HZ
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 5) THEN
          IF (J0 .GT. 1) D_L(J,L) = -(1.0/HY+0.5*VA2)/HY
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 3) THEN
```

```
          IF (I0 .GT. 1) D_L(J,L) = -(1.0/HX+0.5*VA1)/HX
          L = L+1
        ENDIF
        D_L(J,L) = 2.0/HX**2+VC
        IF (NDIAG_LOC .GE. 5) THEN
          D_L(J,L) = D_L(J,L) + 2.0/HY**2
          IF (NDIAG_LOC .GE. 7) THEN
            D_L(J,L) = D_L(J,L) + 2.0/HZ**2
          ENDIF
        ENDIF
        L = L+1
        IF (NDIAG_LOC .GE. 2) THEN
          IF (I0 .LT. NX) D_L(J,L) = -(1.0/HX-0.5*VA1)/HX
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 4) THEN
          IF (J0 .LT. NY) D_L(J,L) = -(1.0/HY-0.5*VA2)/HY
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 6) THEN
          IF (K0 .LT. NZ) D_L(J,L) = -(1.0/HZ-0.5*VA3)/HZ
        ENDIF
 100  CONTINUE
!$OMP ENDDO

!$OMP END PARALLEL

      RETURN
      END

C ======================================
* SOLUTE ERROR
* | X1 - X2 |
C ======================================
      REAL*8 FUNCTION ERRNRM(X1,X2,LEN)
      IMPLICIT  REAL*8 (A-H,O-Z)
      DIMENSION X1(*),X2(*)
C
      S = 0D0
      DO 100 I = 1,LEN
        SS = X1(I) - X2(I)
        S = S + SS * SS
```

```
 100  CONTINUE
C
      ERRNRM = SQRT( S )
      RETURN
      END
```

(4)　手法概要

対角要素に絶対値の大きな要素を並べる並び換えを行います.
並び換えを行った行列に対して行, 列の均衡化を行うスケーリングを施します.
この行列をLU分解します.
スーパーノードに対応する非ゼロ要素を2次元のpanelに格納します.
安定化のためのピボットは対角ブロック部分で検索します.
指定した値より絶対値が大きな値が見つかるとピボットとして採用する閾値THEPSZを指定できます. 求めたピボットが小さすぎるとき, SPEPSZで指定した値で近似してLU分解を続けるstatic pivotを指定することも可能です.
詳細は“付録1 参考文献一覧表”の以下の文献を参照願います.
対角要素に絶対値の大きな要素を並べる方法に関しては,の[23],[57]をマッチングの応用アルゴリズムの詳細は[13]を, フィボナッチヒープに関しては[17]を,スパースな非対称な実行列のLU分解に関しては[19],[2],[22],[48],[68]を行列の均衡化やピボットに関しては[63],[69]をそれぞれ参照願います.

DM_VSSPS

正値対称スパース行列の連立 1 次方程式 (Left-looking な LDL^T 分解法)
CALL DM_VSSPS(A, NZ, NROW, NFCNZ, N, IORDERING, NPERM, ISW, EPSZ, B, NASSIGN, NSUPNUM, NFCNZFACTOR, PANELFACTOR, NSIZEFACTOR, NFCNZINDEX, NPANELINDEX, NSIZEINDEX, NDIM, NPOSTO, W, IW1, IW2, IW3, ICON)

(1)　機能

$n \times n$の正値対称スパース行列$\boldsymbol{A}$を変形コレスキー分解法によりLDL^T分解して解きます.

$$\boldsymbol{Ax=b}$$

正値対称スパース行列$\boldsymbol{A}$は以下のように分解されます.

$$\boldsymbol{QPAP^TQ^T=LDL^T}$$

ただし, $\boldsymbol{P}$はorderingによる行列要素の並び換えを示す置換行列, $\boldsymbol{Q}$はpost orderによる行列要素の並び換えを示す置換行列を示します. $\boldsymbol{P}$,$\boldsymbol{Q}$は直交行列です.

$\boldsymbol{L}$は単位下三角行列, $\boldsymbol{D}$は対角行列です.

(2)　パラメタ

A ..... 入力. 係数行列$\boldsymbol{A}$のスパースな正値対称行列の下三角行列部分$\{a_{ij}|i \geq j\}$を圧縮列格納法で A(1:NZ)に格納します.
A(NZ)なる 1 次元配列.
圧縮列格納法については,実スパース行列と実ベクトル(圧縮列格納法), DM_VMVSCC の図 DM_VMVSCC-1 を参照してください.

NZ ..... 入力. 係数行列$\boldsymbol{A}$のスパースな正値対称行列の下三角行列部分にある非零要素の総数.

NROW ..... 入力. 圧縮列格納法で使用される行指標で A に格納される要素が何番目の行ベクトルに属するかを示します.
NROW(NZ)なる 1 次元配列.

NFCNZ ..... 入力. 行列$\boldsymbol{A}$の各列の非零要素を列方向に圧縮して順次配列 A に格納するとき,対応する列の最初の非零要素が格納される位置を表します.
NFCNZ(N+1)=NZ+1.
NFCNZ(N+1)なる 1 次元配列.

N ..... 入力. 行列$\boldsymbol{A}$の次数 n.

IORDERING ..... 入力. ordering を表す置換行列$\boldsymbol{P}$で$\boldsymbol{PAP^T}$と変換した行列を LDL^T 分解するかを指定します.置換行列は直交行列です.
1 のとき, $\boldsymbol{PAP^T}$と変換したものを LDL^T 分解します.
1 以外のとき, 行列$\boldsymbol{A}$をそのまま LDL^T 分解します.

NPERM ..... 入力. IORDERING=1 のとき使用する置換行列をベクトルで指定します.
NPERM(N)なる 1 次元配列.
((3)使用上の注意 a.注意①参照).

ISW ..... 入力. 呼び出しに関する制御情報を示します.

1) 1 のとき,初回呼び出し.
2) 2 のとき, 1 回目の呼び出しでは PANELFACTOR または NPANELINDEX の大きさが不足していた. ICON=31000 で終了した. このとき, NSIZEFACTOR,または NSIZEINDEX に返却された必要な大きさで PANELFACTOR または NPANELINDEX を確保し直して再度呼び出しします.
   さらに A, NZ, NROW, NFCNZ, N, IORDERING, NPERM, NASSIGN, NSUPNUM, NFCNZFACTOR, NFCNZINDEX, NPANELINDEX, NPOSTO, NDIM, W, IW1, IW2, IW3 に格納されている値を変更してはなりません.
3) 3 のとき,同じ非零パターンを持つ次数の同じ行列で,行列要素の値が異なる行列に対して symbolic decomposition の解析結果や必要な大きさが同じになる配列 PANELFACTOR, NPANELINDEX を再利用して LDL^T 分解を行って方程式を解くことを指定します.行列の値を配列要素に格納し直して呼び出します.
   このとき, NROW の値を変えずに配列 A に行列要素の値を格納し直すか,別の配列 C に格納し引数 A として受け渡さなければなりません.
   さらに NZ, NROW, NFCNZ, N, IORDERING, NPERM, NASSIGN, NSUPNUM, NFCNZFACTOR, NSIZEFACTOR, NFCNZINDEX, NPANELINDEX, NSIZEINDEX, NPOSTO, NDIM, W, IW1, IW2, IW3 に格納されている値を変更してはなりません.
4) 4 のとき,先立つ呼び出しで LDL^T 分解された同じ係数行列に対する連立 1 次方程式の右辺ベクトル $\boldsymbol{b}$ を変えて再度解くことを指定します.このとき先立つ呼び出しで LDL^T 分解した結果を使います.
   さらに N, IORDERING, NPERM, NASSIGN, NSUPNUM, NFCNZFACTOR, NSIZEFACTOR, NFCNZINDEX, NPANELINDEX, NSIZEINDEX, NPOSTO, NDIM, IW3 に格納されている値を変更してはなりません.

EPSZ ..... 入力. ピボットの相対判定値 (≥0.0)
0.0 のときは標準値が設定されます.
((3)使用上の注意 a.注意②参照).

B ..... 入力. $\boldsymbol{Ax=b}$ の右辺定数ベクトル.
出力. 解ベクトル.
B(N)なる 1 次元配列.

NASSIGN..... 出力. 各 supernode は複数の列ベクトルからなり,分解結果に関して共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します. この panel を PANELFACTOR の 1 次元部分配列として順番に割り付けたとき何番目になるかを示します. j=NASSIGN(i)のとき, i 番目の supernode を j 番目に割り付けたことを示します.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.

分解結果の格納方法については図 DM_VSSPS-1 を参照してください.
NASSIGN(N)なる 1 次元配列.
((3)使用上の注意 a.注意③参照).

NSUPNUM ..... 出力. supernode の総数.
入力. ISW≠1 のとき, 初回呼び出しの値を再利用します. (≤n)

NFCNZFACTOR .. 出力. 各 supernode は複数の列ベクトルからなり,分解結果に関して共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します. この panel を PANELFACTOR の 1 次元部分配列として順番に割り付けたとき, i 番目の panel の先頭要素 panel(1,1)が 1 次元配列の PANELFACTOR の何番目の要素になるかを示します.
NFCNZFACTOR(N+1)なる 8 バイト整数型の 1 次元配列.
分解結果の格納方法については図 DM_VSSPS-1 を参照してください.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.

PANELFACTOR .. 出力. 各 supernode は複数の列ベクトルからなり,分解結果に関して共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します. panel を順番に割り付けます.
i 番目の supernode に対応する panel が何番目の位置に割り当てられるかは j=NASSIGN(i)から分かります.その先頭位置は NFCNZFACTOR(j)に格納されています. panel ごとに分解結果が格納されます.
i 番目に割付けられた panel の大きさは DIM(1,i)×DIM(2,i)の 2 次元配列と見なせます. i 番目の panel の panel(s,t), s>t, s=1,⋯,DIM(1,i), t=1,⋯, DIM(2,i)に単位下三角行列 $\boldsymbol{L}$ の対角要素を除いた対応部分が格納されます.対角部分 panel(t,t)には対角行列 $\boldsymbol{D}$ の対応部分が格納されます.
PANELFACTOR(NSIZEFACTOR)なる 1 次元配列.
分解結果の格納方法については図 DM_VSSPS-1 を参照してください.
((3)使用上の注意 a.注意③参照).

NSIZEFACTOR.... 入力. PANELFACTOR の大きさを示す. 8 バイト整数型.
出力. PANELFACTOR の大きさとして必要な大きさが返却されます.
((3)使用上の注意 a.注意③参照).

NFCNZINDEX ..... 出力. 各 supernode は複数の列ベクトルからなり,分解結果に関して共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します. この行指標ベクトルを NPANELINDEX に順番に割り付けたとき i 番目の行指標ベクトルの先頭要素が 1 次元配列である NPANELINDEX の何番目の要素になるかを示します.
NFCNZINDEX(N+1)なる 8 バイト整数型の 1 次元配列.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
分解結果の格納方法については図 DM_VSSPS-1 を参照してください.

NPANELINDEX... 出力. 各 supernode は複数の列ベクトルからなり,分解結果に関して共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します. この行指標ベクトルを NPANELINDEX に順番に割り付けます. i 番目の supernode に対応する panel の行の指標ベクトルが何番目に割り当てられるかは j=NASSIGN(i)から分かります.その先頭位置は

NFCNZINDEX(j)に格納されています. panel ごとの行の指標ベクトルが格納されます.
この行指標は行列 $\boldsymbol{A}$ を post order で並び換えた行列 $\boldsymbol{QAQ}^{T}$ での行の番号です.
NPANELINDEX(NSIZEINDEX)なる 1 次元配列.
分解結果の格納方法については図 DM_VSSPS-1 を参照してください.
((3)使用上の注意 a.注意③参照).

NSIZEINDEX ..... 入力. NPANELINDEX の大きさを示す. 8 バイト整数型.
出力. 必要な大きさが返却されます.
((3)使用上の注意 a.注意③参照).

NDIM ..... 出力. NDIM(1,i)および NDIM(2,i)は i 番目に割り付けられた panel の 1 次元目と 2 次元目の大きさを示します.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
NDIM(2,N)なる 2 次元配列.
分解結果の格納方法については図 DM_VSSPS-1 を参照してください.

NPOSTO ..... 出力. post order で i 番目の node が行列 $\boldsymbol{A}$ の何番目の列番号に対応するかを表す 1 次元ベクトル.
入力. ISW≠1 のとき, 初回呼び出しの値を再利用します.
NPOSTO(N)なる 1 次元配列.
((3)使用上の注意 a.注意④参照).

W ..... 作業域.
出力/入力.
IORDERING＝1 のとき大きさ NZ の 1 次元配列.
ISW=1, 2, 3 で続けて呼び出すとき，呼び出しの間で必要なデータを受け渡すために使われます．呼び出しの間で値を変更してはいけません.
IORDERING≠1 のとき,大きさ 1 の 1 次元配列.

IW1..... 作業域.
出力/入力.
IORDERING＝1 のとき大きさ NZ+N+1 の 1 次元配列.
ISW=1, 2, 3 で続けて呼び出すとき,呼び出しの間で必要なデータを受け渡すために使われます.呼び出しの間で値を変更してはいけません.
IORDERING≠1 のとき,大きさ 1 の 1 次元配列.

IW2..... 作業域.
出力/入力. 大きさ NZ+N+1 の 1 次元配列.
ISW=1, 2, 3 で続けて呼び出すとき,呼び出しの間で必要なデータを受け渡すために使われます.呼び出しの間で値を変更してはいけません.

IW3..... 作業域.
出力/入力. 大きさ N×35+35 の 1 次元配列.
ISW=1, 2, 3, 4 で続けて呼び出すとき,呼び出しの間で必要なデータを受け渡すために使われます.呼び出しの間で値を変更してはいけません.
IW3(N*35+35)なる 1 次元配列.

ICON ..... 出力. コンディションコード.
表 DM_VSSPS-1 参照.

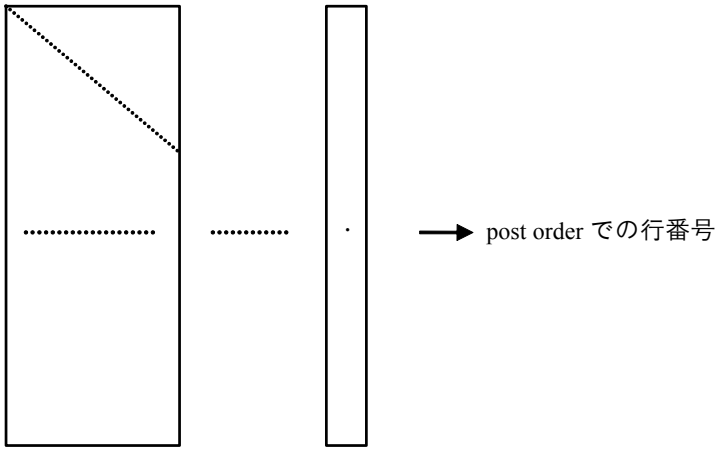


図 DM-VSSPS-1 分解結果の格納概念図

j = NASSIGN(i)　→ i 番目の supernode は j 番目に格納されます.
p = NFCNZFACTOR(j)　→ j 番目の panel は PANELFACTOR の p 番目の要素から DIM(1,j)×DIM(2,j)の長さを占めます.
q = NFCNZINDEX(j)　→ j 番目の panel の行指標を表すベクトルは NPANELINDEX の q 番目の要素から DIM(1,j)の長さを占めます.

panel は大きさ DIM(1,j)×DIM(2,j)の配列と見なせます.
panel(s,t) , s > t , s=1 , … , DIM(1,j),
t=1 , … , DIM(2,j)
に分解結果の単位下三角行列 $\boldsymbol{L}$ の対角要素を除いた部分が格納されます.
panel(t,t)に対角行列 $\boldsymbol{D}$ の対応部分が格納されます.
行指標の値は行列 $\boldsymbol{A}$ の列番号をもつ node を post order で並び換えた $\boldsymbol{QAQ^T}$ での列番号を表します.

表 DM_VSSPS-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし	ー
10000	行列が正値でなかった.	処理は続行する.
20000	ピボットが相対的にゼロになった. 行列は非正則の可能性が強い.	処理を打ち切る.
30000	N<1, NZ<0, NFCNZ(N+1)≠NZ+1, NSIZEFACTOR<1, NSIZEINDEX<1, EPSZ<0.0, ISW<1, ISW>4	
30100	NPERM に指定した置換行列が正しくない.	
30200	NROW(j)に格納された i 列目の行指標 k が k<i または k>N	
30300	i 列目の行指標の数 NFCNZ(i+1)-NFCNZ(i)>n-i+1	
30400	対角要素が格納されていない列がある.	
31000	PANELFACTOR の大きさ NSIZEFACTOR または NPANELINDEX の大きさ NSIZEINDEX が小さすぎる.	NSIZEFACTOR または NSIZEINDEX で指定された大きさで PANELFACTOR または NPANELINDEX を割り付けて ISW=2 として再度呼び出す.

(3)　使用上の注意

a.　注意

①　直交行列である置換行列$\boldsymbol{P}$の要素$\boldsymbol{p_{ij}=1}$のとき, NPERM(i)=jと表現します.
逆は以下のようにして求めることができます.

```
DO i=1,n
 j=NPERM(i)
 NPERMINV(j)=i
 ENDDO
```

Fill-Reducing OrderingはMETISなどを使って求めることができます.
詳細は“付録1 参考文献一覧表”の[43],[44]を参照願います.

②　ピボットの相対零判定値にある値を設定したとすると, この値は次の意味を持っています. すなわち, 変形コレスキー分解法による LDL^T 分解の過程で, ピボットの絶対値がその値より小さくなった場合に, そのピボットの値を相対的に零と見なし, ICON=20000 として処理を打ち切ります. EPSZ の標準値は, 丸め誤差の単位を u としたとき, EPSZ=16u です.
なお, ピボットの絶対値が小さくなっても, 処理を続行させたい場合には, EPSZ に極小の値を与えれば良いのですが, 結果の精度は保証されません.
分解の途中でピボットが負となった場合, 係数行列は正値ではありません. このとき, ICON=10000として処理は続行します. ただし, ピボッティングを行っていないため, 計算誤差は大きい可能性があります.

③　分解結果を格納する配列PANELFACTOR, NPANELINDEXの必要な大きさは,事前には分かりません.十分大きな配列を割り当てるか,本ルーチンを呼び出してsymbolic decompositionを行った解析の結果を使って割り当てを行うことができます.
例えば,大きさ1の1次元配列などを割付けます.そしてその大きさ1などの小さな値をNSIZEFACTOR, NSIZEINDEXに指定して, ISW=1で呼び出します. symbolic decompositionを行い, ICON=31000で終了し, NSIZEFACTORとNSIZEINDEXに必要な大きさが返却されます.必要な大きさの配列を割付け直して, ISW=2で呼び出すことで, symbolic decomposition以降の処理を続けることができます.

④　列番号に対応するノードを考えます.これをpost orderで順番を並び換えたときの番号の対応関係がNPOSTOに格納されています. post orderのi番目のノードが並び換える前の何番目のノードに対応するかを表します. j=nposto(i)はj番目であることを表します.
①同様にこれは直交行列である置換行列$\boldsymbol{Q}$をあらわし,行列$\boldsymbol{A}$を$\boldsymbol{QAQ^T}$と並び換えることに相当します.
逆変換$\boldsymbol{Q^T}$は以下のようにして求めることができます.

```
DO i=1,n
 j=NPOSTO(i)
 NPOSTOINV(j)=i
 ENDDO
```

b.　使用例

連立1次方程式 $\boldsymbol{Ax}=\boldsymbol{f}$ を解きます. 行列$\boldsymbol{A}$は境界条件として立方体の境界で0となる楕円型の偏微分方程式に有限差分法を適用して生成されます.

$$-\Delta u + a\nabla u + cu = f$$

ここで$a=(a_1, a_2, a_3)$, a_1, a_2, a_3およびcはゼロで,ラプラシアンになり行列$\boldsymbol{A}$は正値対称です.行列$\boldsymbol{A}$はサブルーチン init_mat_diagによって生成されます. これを圧縮列格納法に変換します.
(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4プロセッサーのシステムで4つのスレッドで並列実行するときは, OMP_NUM_THREADSを4に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT  REAL*8 (A-H,O-Z)
      PARAMETER (NORD=39,NX = NORD,NY =NORD ,NZ = NORD,
     $      N = NX*NY*NZ)
      PARAMETER (K = N+1)
      PARAMETER (NDIAG = 7,NDIAGH=4)

      DIMENSION NOFST(NDIAG)
      DIMENSION DIAG(K,NDIAG),DIAG2(K,NDIAG)
      DIMENSION C(K*NDIAG),NROWC(K*NDIAG),NFCNZC(N+1),
```

```
     $          WC(K*NDIAG),IWC(2,K*NDIAG)
      DIMENSION A(NDIAGH*N),NROW(K*NDIAG),NFCNZ(N+1),
     $          NPERM(N),NASSIGN(N),W(NDIAGH*N),
     $          NPOSTO(N),NDIM(2,N),
     $          IW1(NDIAGH*N+N+1),
     $          IW2(NDIAGH*N+N+1),
     $          IW3(35*N+35)
      REAL*8, DIMENSION(:), ALLOCATABLE ::  PANELFACTOR
      INTEGER*4, DIMENSION(:), ALLOCATABLE :: NPANELINDEX
      REAL*8 DUMMYF
      INTEGER*4 NDUMMYI
      INTEGER*8 NSIZEFACTOR,NSIZEINDEX,
     $          NFCNZFACTOR(N+1),
     $          NFCNZINDEX(N+1)
      DIMENSION X(N),B(N),SOLEX(N)


      PRINT *,'    LEFT-LOOKING MODIFIED CHOLESKY METHOD'
      PRINT *,'    FOR SPARSE POSITIVE DEFINITE MATRICES'
      PRINT *,'    IN COMPRESSED COLUMN STORAGE'
      PRINT *

      SOLEX(1:N)=1.0D0
      PRINT *,'    EXPECTED SOLUTIONS'
      PRINT *,'    X(1) = ',SOLEX(1),' X(N) = ',SOLEX(N)
      PRINT *

      VA1 = 0.0D0
      VA2 = 0.0D0
      VA3 = 0.0D0
      VC =  0.0D0
      XL = 1.0
      YL = 1.0
      ZL = 1.0
      CALL INIT_MAT_DIAG(VA1,VA2,VA3,VC,DIAG,NOFST
     &          ,NX,NY,NZ,XL,YL,ZL,NDIAG,N,K)

      DO I=1,NDIAG
C
      IF(NOFST(I).LT.0)THEN
      NBASE=-NOFST(I)
      LENGTH=N-NBASE
```

```
      DIAG2(1:LENGTH,I)=DIAG(NBASE+1:N,I)
      ELSE
      NBASE=NOFST(I)
      LENGTH=N-NBASE
      DIAG2(NBASE+1:N,I)=DIAG(1:LENGTH,I)
      ENDIF
C
      ENDDO
C
      NUMNZC=1
      NUMNZ=1
      DO J=1,N
      NTOPCFGC=1
      NTOPCFG=1
      DO I=NDIAG,1,-1
C
      IF(DIAG2(J,I).NE.0.0D0)THEN
C
      NCOL=J-NOFST(I)
      C(NUMNZC)=DIAG2(J,I)
      NROWC(NUMNZC)=NCOL
C
      IF(NCOL.GE.J)THEN
      A(NUMNZ)=DIAG2(J,I)
      NROW(NUMNZ)=NCOL
      ENDIF

C
      IF(NTOPCFGC.EQ.1)THEN
      NFCNZC(J)=NUMNZC
      NTOPCFGC=0
      ENDIF
C
      IF(NTOPCFG.EQ.1)THEN
      NFCNZ(J)=NUMNZ
      NTOPCFG=0
      ENDIF
C
      IF(NCOL.GE.J)THEN
      NUMNZ=NUMNZ+1
      ENDIF
C
```

```
      NUMNZC=NUMNZC+1
      ENDIF
C
      ENDDO
      ENDDO
      NFCNZC(N+1)=NUMNZC
      NNZC=NUMNZC-1
      NFCNZ(N+1)=NUMNZ
      NNZ=NUMNZ-1
C

      CALL DM_VMVSCC(C,NNZC,NROWC,NFCNZC,N,SOLEX,
     $             B,WC,IWC,ICON)
C
      X=B
      IORDERING=0
      ISW=1
      EPSZ=0.0D0
      NSIZEFACTOR=1
      NSIZEINDEX=1

      CALL DM_VSSPS(A,NNZ,NROW,NFCNZ,N,IORDERING,
     $              NPERM,ISW,EPSZ,X,NASSIGN,NSUPNUM,
     $              NFCNZFACTOR,DUMMYF,
     $              NSIZEFACTOR,NFCNZINDEX,
     $              NDUMMYI,NSIZEINDEX,NDIM,NPOSTO,
     $              W,IW1,IW2,IW3,ICON)

      PRINT *
      PRINT *,'    ICON = ',ICON,' NSIZEFACTOR = ',NSIZEFACTOR,
     $       'NSIZEINDEX = ',NSIZEINDEX
      PRINT *
C
C     ALLOCATE STORAGES IN RETURNED SIZES
C
      ALLOCATE( PANELFACTOR(NSIZEFACTOR) )
      ALLOCATE( NPANELINDEX(NSIZEINDEX) )

      ISW=2

      CALL DM_VSSPS(A,NNZ,NROW,NFCNZ,N,IORDERING,
     $              NPERM,ISW,EPSZ,X,NASSIGN,NSUPNUM,
```

```
     $              NFCNZFACTOR,PANELFACTOR,
     $              NSIZEFACTOR,NFCNZINDEX,
     $              NPANELINDEX,NSIZEINDEX,NDIM,NPOSTO,
     $              W,IW1,IW2,IW3,ICON)

      ERR = ERRNRM(SOLEX,X,N)

      PRINT *,'    COMPUTED VALUES'
      PRINT *,'    X(1) = ',X(1),' X(N) = ',X(N)
      PRINT *
      PRINT *,'    ICON = ',ICON
      PRINT *
      PRINT *,'    N = ',N,' :: NX = ',NX,' NY = ',NY,' NZ = ',NZ
      PRINT *
      PRINT *,'    ERROR = ',ERR
      PRINT *
      PRINT *

      IF(ERR.LT.1.0D-8.AND.ICON.EQ.0)THEN
         WRITE(*,*)'    ********** OK **********'
      ELSE
         WRITE(*,*)'    ********** NG **********'
      ENDIF

      DEALLOCATE( PANELFACTOR,NPANELINDEX )

      STOP
      END

C ======================================
C     INITIALIZE COEFFICIENT MATRIX
C ======================================
      SUBROUTINE INIT_MAT_DIAG(VA1,VA2,VA3,VC,D_L,OFFSET
     &                        ,NX,NY,NZ,XL,YL,ZL,NDIAG,LEN,NDIVP)
      IMPLICIT  REAL*8(A-H,O-Z)
      DIMENSION D_L(NDIVP,NDIAG)
      INTEGER   OFFSET(NDIAG)
C
      IF (NDIAG .LT. 1) THEN
        WRITE (*,*) 'SUBROUTINE INIT_MAT_DIAG:'
        WRITE (*,*) ' NDIAG SHOULD BE GREATER THAN OR EQUAL TO 1'
        RETURN
```

```
      ENDIF

!$OMP PARALLEL DEFAULT(PRIVATE)
!$OMP+     SHARED(VA1,VA2,VA3,VC,D_L,OFFSET
!$OMP+      ,NX,NY,NZ,XL,YL,ZL,NDIAG,LEN,NDIVP)

C NDIAG CANNOT BE GREATER THAN 7
      NDIAG_LOC = NDIAG
      IF (NDIAG .GT. 7) NDIAG_LOC = 7

C INITIAL SETTING
      HX = XL/(NX+1)
      HY = YL/(NY+1)
      HZ = ZL/(NZ+1)

!$OMP DO
      DO I = 1,NDIVP
      DO J = 1,NDIAG
      D_L(I,J) = 0.0
      ENDDO
      ENDDO
!$OMP ENDDO

      NXY = NX*NY

C OFFSET SETTING
!$OMP SINGLE
      L = 1
      IF (NDIAG_LOC .GE. 7) THEN
        OFFSET(L) = -NXY
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 5) THEN
        OFFSET(L) = -NX
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 3) THEN
        OFFSET(L) = -1
        L = L+1
      ENDIF
      OFFSET(L) = 0
      L = L+1
```

```
      IF (NDIAG_LOC .GE. 2) THEN
        OFFSET(L) = 1
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 4) THEN
        OFFSET(L) = NX
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 6) THEN
        OFFSET(L) = NXY
      ENDIF
!$OMP END SINGLE

C MAIN LOOP
!$OMP DO
      DO 100 J = 1,LEN
        JS = J

C DECOMPOSE JS-1 = (K0-1)*NX*NY+(J0-1)*NX+I0-1
        K0 = (JS-1)/NXY+1
        IF (K0 .GT. NZ) THEN
          PRINT*,'ERROR; K0.GH.NZ '
          GOTO 100
        ENDIF
        J0 = (JS-1-NXY*(K0-1))/NX+1
        I0 = JS - NXY*(K0-1) - NX*(J0-1)
        L = 1

        IF (NDIAG_LOC .GE. 7) THEN
          IF (K0 .GT. 1) D_L(J,L) = -(1.0/HZ+0.5*VA3)/HZ
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 5) THEN
          IF (J0 .GT. 1) D_L(J,L) = -(1.0/HY+0.5*VA2)/HY
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 3) THEN
          IF (I0 .GT. 1) D_L(J,L) = -(1.0/HX+0.5*VA1)/HX
          L = L+1
        ENDIF
        D_L(J,L) = 2.0/HX**2+VC
        IF (NDIAG_LOC .GE. 5) THEN
```

```
          D_L(J,L) = D_L(J,L) + 2.0/HY**2
          IF (NDIAG_LOC .GE. 7) THEN
            D_L(J,L) = D_L(J,L) + 2.0/HZ**2
          ENDIF
        ENDIF
        L = L+1
        IF (NDIAG_LOC .GE. 2) THEN
          IF (I0 .LT. NX) D_L(J,L) = -(1.0/HX-0.5*VA1)/HX
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 4) THEN
          IF (J0 .LT. NY) D_L(J,L) = -(1.0/HY-0.5*VA2)/HY
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 6) THEN
          IF (K0 .LT. NZ) D_L(J,L) = -(1.0/HZ-0.5*VA3)/HZ
        ENDIF
 100  CONTINUE
!$OMP ENDDO

!$OMP END PARALLEL

      RETURN
      END

C ======================================
* SOLUTE ERROR
* | X1 - X2 |
C ======================================
      REAL*8 FUNCTION ERRNRM(X1,X2,LEN)
      IMPLICIT  REAL*8 (A-H,O-Z)
      DIMENSION X1(*),X2(*)
C
      S = 0D0
      DO 100 I = 1,LEN
        SS = X1(I) - X2(I)
        S = S + SS * SS
 100  CONTINUE
C
      ERRNRM = SQRT( S )
      RETURN
      END
```

(4)　手法概要

symbolic decomposition を行い,列の間のデータ依存関係や修正コレスキー分解の結果の行列 $\boldsymbol{L}$ の非零要素の情報を解析します.これをもとに複数の列を束ねた supernode を検出します. supernode にまとめるとき,列の非零のパターンが似ているものをまとめますが,分解過程でのキャッシュのデータを効率よく再利用するために,余分な zero 要素を持つ列も束ねノードを構成する列の数を増やします.

supernode を構成する列に対して,修正コレスキー分解の結果の非零要素の行番号を表す行指標の和集合をもとめます. supernode の修正コレスキー分解の結果は, 1 次元目の大きさがこの行指標の集合の要素数となる 2 次元の panel に圧縮して格納します.行指標の集合は vector で表現します.

Left-looking な変形コレスキー分解で LDL^T 分解します.

技術全般に関しては, “付録 1 参考文献一覧表” の[19]を参照して下さい.

DM_VSSSLU

構造的に対称な実スパース行列の LU 分解
CALL DM_VSSSLU(A, NZ, NROW, NFCNZ, N, ISCLITERMAX, IORDERING, NPERM, ISW, NASSIGN, NSUPNUM, NFCNZFACTORL, PANELFACTORL, NSIZEFACTORL, NFCNZINDEXL, NPANELINDEXL, NSIZEINDEX, NDIM, NFCNZFACTORU, PANELFACTORU, NSIZEFACTORU, NFCNZINDEXU, NPANELINDEXU, NPOSTO, SCLROW,SCLCOL, EPSZ, THEPSZ, IPIVOT, ISTATIC, SPEPSZ, W, IW, ICON)

(1) 機能

$n \times n$ の構造的に対称な実スパース行列 $\boldsymbol{A}$ に行および列の均衡化を行うスケーリングを行います. スーパーノードの対角ブロック内で指定した Pivot をとり, LU 分解します.
(構造的に対称な実スパース行列の非零要素はそれと対称な位置に要素を持ち、その値は必ずしも等しくはない行列です.)
構造的に対称な実スパース行列 $\boldsymbol{A}$ は以下のように変換できます.

$$\boldsymbol{A_1 = D_r A D_c}$$

ここで $\boldsymbol{D_r}$ は行のスケーリングを行う対角行列, $\boldsymbol{D_c}$ は列のスケーリングを行う対角行列です.

$$\boldsymbol{A_2 = Q P A_1 P^T Q^T}$$

$\boldsymbol{A_2}$ は, スーパーノードの対角ブロックで指定された pivot を行い, 行および列の入れ換えを行い LU 分解されます.
ただし, $\boldsymbol{P}$ は, $\boldsymbol{A}$ の非ゼロパターンに対して求めた ordering による行列要素の並び換えを示す置換行列で, $\boldsymbol{Q}$ は $\boldsymbol{SYM}$ に対する post order による行列要素の並び換えを示す置換行列を示します. $\boldsymbol{P}$, $\boldsymbol{Q}$ は直交行列です.

LU 分解の結果の行列 $\boldsymbol{L}$ と $\boldsymbol{U}$ の非零要素のパターンはそれぞれに対して対称になります. $\boldsymbol{L}$ は下三角行列, $\boldsymbol{U}$ は単位上三角行列です.

Pivot処理で, 閾値 THEPSZ より絶対値が大きな Pivot を見つけることが出来なかったとき, 対角ブロック内の絶対値が最大の要素を Pivot の候補とします. この値が小さ過ぎるときに Static Pivot を与えて近似的に LU 分解を続けることができます.

(2) パラメタ

A..... 入力.構造的に対称な実スパースな係数行列 $\boldsymbol{A}$ を圧縮列格納法で A(1:NZ)に格納します.
A(NZ)なる 1 次元配列.
圧縮列格納法については, 実スパース行列と実ベクトル(圧縮列格納法),

DM_VMVSCC の図 DM_VMVSCC-1 を参照してください.
((3)使用上の注意 a. 注意⑤参照).

NZ ..... 入力. 構造的に対称な実スパースな係数行列 $\boldsymbol{A}$ の格納する非零要素の総数.

NROW..... 入力. 圧縮列格納法で使用される行指標で A に格納される要素が何番目の行ベクトルに属するかを示します.
NROW(NZ)なる 1 次元配列.

NFCNZ..... 入力. 行列 $\boldsymbol{A}$ の各列の格納する非零要素を列方向に圧縮して順次配列 A に格納するとき,対応する列の最初の格納する非零要素が格納される位置を表します.
NFCNZ(N+1)=NZ+1.
NFCNZ(N+1)なる 1 次元配列.

N..... 入力. 行列 $\boldsymbol{A}$ の次数 n.

ISCLITERMAX .... 入力. 係数行列のスケーリングを行う対角行列 $\boldsymbol{D_r}$ と $\boldsymbol{D_c}$ を反復して求める反復回数.
ISCLITERMAX ≤ 0 のときスケーリングは行わすに, $\boldsymbol{D_r}$ および $\boldsymbol{D_c}$ は単位行列が設定されます.
ISCLITERMAX ≥ 10 のときは 10 回を上限値とします.

IORDERING..... 入力. ordering を表す置換行列 $\boldsymbol{P}$ で $\boldsymbol{PAP^T}$ と変換した行列を LU 分解するかを指定します. 置換行列は直交行列です.
1 のとき, $\boldsymbol{PAP^T}$ と変換した行列を LU 分解します.
1 以外のとき, 行列をそのまま LU 分解します.
((3)使用上の注意 a. 注意①参照).

NPERM ..... 入力. IORDERING=1 のとき使用する置換行列をベクトルで指定します.
NPERM(N)なる 1 次元配列.
((3)使用上の注意 a. 注意①参照).

ISW ..... 入力. 呼び出しに関する制御情報を示します.
1)　1 のとき
初回の呼び出し.Symbolic decomposition を行い, 必要な領域が割り当てられているか調べ計算を行います.
2)　2 のとき
ISW=1 で呼び出したとき, PANELFACTORL, PANELFACTORU, NPANELINDEXL または NPANELINDEXU の大きさが不足して ICON=31000 で終了した処理を継続します. このとき, NSIZEFACTORL, NSIZEFACTORU または NSIZEINDEX に返却された必要な大きさで PANELFACTORL, PANELFACTORU, NPANELINDEXL または NPANELINDEXU を確保し直して指定しなおして再度呼び出します.
これらの引数と実行を制御する引数 ISW 以外の引数に格納されている値は変更してはいけません.

NASSIGN..... 出力. 各 supernode に対する L, U は圧縮して 2 次元の panel に格納します. この panel を PANELFACTORL および PANELFACTORU の 1 次元部分

配列として順番に割り付けたとき何番目になるかを示します. これらの指標ベクトルも同じように NPANELINDEXL, NPANELINDEXU に割り付けます. j=NASSIGN(i)のとき, i 番目の supernode を j 番目に割り付けたことを示します.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
分解結果の格納方法については図 DM_VSSSLU-1 を参照してください.
NASSIGN(N)なる 1 次元配列.

NSUPNUM ..... 出力. supernode の総数.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します. (≤n)

NFCNZFACTORL 出力. 構造的に対称な実スパース行列の LU 分解の結果の行列 ***L*** および ***U*** はスーパーノードに対応しておのおの求めます. 各 supernode に対応する ***L*** の列ベクトルは, 共通の行指標ベクトルを持つように圧縮してブロック対角部分に ***U*** の対応部分を含めて 2 次元の panel に格納します. この panel を PANELFACTORL の 1 次元部分配列として順番に割り付けたとき, i 番目の panel の先頭要素 panel(1,1) が 1 次元配列の PANELFACTORL の何番目の要素になるかを示します.
NFCNZFACTORL(N+1)なる 8 バイト整数型の 1 次元配列.
結果の格納方法については図 DM_VSSSLU-1 を参照してください.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.

PANELFACTORL 出力. 各 supernode の分解結果に対応する行列 ***L*** の複数の列ベクトルと行列 ***U*** のブロック対角部分について，共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します. panel を順番に割り付けます. i 番目の supernode に対応する panel が何番目の位置に割り当てられるかは j=NASSIGN(i)から分かります. その先頭位置は NFCNZFACTORL(j)に格納されています. panel ごとに分解結果が格納されます.
i 番目に割付けられた panel の大きさは NDIM(1,i)×NDIM(2,i)の 2 次元配列と見なせます. i 番目の panel の panel(s,t), $s \geq t$, s=1,…,NDIM(1,i), t=1,…,NDIM(2,i)に下三角行列 ***L*** が格納されます. panel の panel(s,t), $s < t$, t=1,…,NDIM(2,i)には単位上三角行列 ***U*** の対角要素を除いたブロック対角部分が格納されます.
PANELFACTORL(NSIZEFACTORL)なる 1 次元配列.
結果の格納方法については図 DM_VSSSLU-1 を参照してください.
((3)使用上の注意 a. 注意③参照).

NSIZEFACTORL . 入力. PANELFACTORL の大きさを示す. 8 バイト整数型.
出力. PANELFACTORL の大きさとして必要な大きさが返却されます.
((3)使用上の注意 a. 注意③参照).

NFCNZINDEXL ... 出力. 各 supernode の分解結果に対応する行列 ***L*** の複数の列ベクトルと行列 ***U*** のブロック対角部分について，共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します. この行指標ベクトルを NPANELINDEXL に順番に割り付けたとき i 番目の行指標ベクトルの先頭要素が 1 次元配列である NPANELINDEXL の何番目の要素になるかを示します.

NFCNZINDEXL(N+1)なる 8 バイト整数型の 1 次元配列.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
結果の格納方法については図 DM_VSSSLU-1 を参照してください.

NPANELINDEXL 出力. 各 supernode の分解結果に対応する行列 ***L*** の複数の列ベクトルと行列 ***U*** のブロック対角部分について，共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します. この行指標ベクトルを NPANELINDEXL に順番に割り付けます. i 番目の supernode に対応する panel の行の指標ベクトルが何番目に割り当てられるかは j=NASSIGN(i)から分かります. その先頭位置は NFCNZINDEXL(j)に格納されています.
この行指標は post order で並び換えたときの行の番号です.
NPANELINDEXL(NSIZEINDEX)なる 1 次元配列.
結果の格納方法については図 DM_VSSSLU-1 を参照してください.
((3)使用上の注意 a. 注意③参照).

NSIZEINDEX ..... 入力. NPANELINDEXL および NPANELINDEXU の大きさを示す.
8 バイト整数型.
出力. 必要な大きさが返却されます.
((3)使用上の注意 a. 注意③参照).

NDIM ..... 出力. NDIM(1,i)および NDIM(2,i)は行列 ***L*** に関して i 番目に割り付けられた panel の 1 次元目と 2 次元目の大きさを示します.
NDIM(1,i)-NDIM(2,i)および NDIM(2,i)は行列 ***U*** に関して割り付けられた panel を転置したときの 1 次元目と 2 次元目の大きさを示します.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
NDIM(2,N)なる 2 次元配列.
結果の格納方法については図 DM_VSSSLU-1 を参照してください.

NFCNZFACTORU 出力. 構造的に対称な実スパース行列の LU 分解の結果の行列 ***U*** に関しては, 各 supernode に対応する ***U*** の行ベクトルは, ブロック対角部分を除いて共通の列指標ベクトルを持つように圧縮し, 転置したものを 2 次元の panel に格納します. この panel を PANELFACTORU の 1 次元部分配列として順番に割り付けたとき, i 番目の panel の先頭要素 panel(1,1)が 1 次元配列の PANELFACTORU の何番目の要素になるかを示します.
NFCNZFACTORU(N+1)なる 8 バイト整数型の 1 次元配列.
結果の格納方法については図 DM_VSSSLU-1 を参照してください.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.

PANELFACTORU 出力. 各 supernode の分解結果に対応する行列 ***U*** のブロック対角部分を除いた複数の行ベクトルについて，共通の列指標ベクトルを持つように圧縮し, 転置して 2 次元の panel に格納します. panel を順番に割り付けます. i 番目の supernode に対応する panel が何番目の位置に割り当てられるかは j=NASSIGN(i)から分かります. その先頭位置は NFCNZFACTORU(j)に格納されています. panel ごとに分解結果が格納されます.
i 番目に割付けられた panel の大きさは

{NDIM(1,i)-NDIM(2,i)}×NDIM(2,i)の 2 次元配列と見なせます. i 番目の panel の panel(s,t), s=1,…,NDIM(1,i)- NDIM(2,i), t=1,…,NDIM(2,i)に単位上三角行列 ***U*** のブロック対角部分を除いた部分が行ベクトルを圧縮し, 転置して格納します.
PANELFACTORU(NSIZEFACTORU)なる 1 次元配列.
結果の格納方法については図 DM_VSSSLU-1 を参照してください.
((3)使用上の注意 a. 注意③参照).

NSIZEFACTORU. 入力. PANELFACTORU の大きさを示す. 8 バイト整数型.
出力. PANELFACTORU の大きさとして必要な大きさが返却されます.
((3)使用上の注意 a. 注意③参照).

NFCNZINDEXU... 出力. 各 supernode の分解結果に対応する行列 ***U*** は対角ブロック部分を除いて複数の行ベクトルについて, 共通の列指標ベクトルを持つように圧縮し, 転置して 2 次元の panel に格納します. 対角ブロック部分も含んだこの列指標ベクトルを NPANELINDEXU に順番に割り付けたとき i 番目の列指標ベクトルの先頭要素が 1 次元配列である NPANELINDEXU の何番目の要素になるかを示します.
NFCNZINDEXU(N+1)なる 8 バイト整数型の 1 次元配列.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
結果の格納方法については図 DM_VSSSLU-1 を参照してください.

NPANELINDEXU 出力. 各 supernode の分解結果に対応する行列 ***U*** の対角ブロックを除いた部分を転置して圧縮して 2 次元の panel に格納します. この列指標ベクトルを対角ブロック部分を含めて NPANELINDEXU に順番に割り付けます. i 番目の supernode に対応する panel の行の指標ベクトルが何番目に割り当てられるかは j=NASSIGN(i)から分かります. その先頭位置は NFCNZINDEXU(j)に格納されています.
この行指標は post order で並び換えたときの列の番号です.
NPANELINDEXU(NSIZEINDEX)なる 1 次元配列.
結果の格納方法については図 DM_VSSSLU-1 を参照してください.
((3)使用上の注意 a. 注意③参照).

NPOSTO ..... 出力. post order で i 番目の node が行列 ***A*** の何番目の列番号に対応するかを表す 1 次元ベクトル.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
NPOSTO(N)なる 1 次元配列.
((3)使用上の注意 a. 注意④参照).

SCLROW ..... 出力. スケーリング対角行列 $\boldsymbol{D}_r$. 対角要素が 1 次元配列に格納されます.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
SCLROW(N)なる 1 次元配列.

SCLCOL..... 出力. スケーリング対角行列 $\boldsymbol{D}_c$. 対角要素が 1 次元配列に格納されます.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
SCLCOL(N)なる 1 次元配列.

EPSZ ..... 入力. 分解過程で対角要素の大きさの絶対値の相対判定値
出力. EPSZ ≤ 0.0 のときは標準値が設定されます.

((3)使用上の注意 a. 注意②参照).

THEPSZ..... 入力. ピボットの判定での閾値(threshold). これより大きな値はピボットとして採用します. 見つかればその時点で, ピボット検索は打ち切ります.
例えば 1.0D-2 程度.
出力. THEPSZ ≤ 0.0 のときは 1.0D-2 が設定されます.
EPSZ ≥ THEPSZ > 0.0 のときは, EPSZ が設定されます.

IPIVOT..... 入力. スーパーノード内でピボットを行うか, 行う場合どのようなピボットを行うかを指定します. 例えば, 完全ピボットとして 40 を指定.
IPIVOT < 10 : または IPIVOT ≥ 50: ピボットなし.
10 ≤ IPIVOT < 20 : 部分ピボット
20 ≤ IPIVOT < 30 : 対角ピボット
21 : スーパーノード内で対角ピボットがとれないとき, Rook ピボットに変更します.
22 : スーパーノード内で対角ピボットがとれないときは, Rook ピボットに, Rook ピボットが取れないときは完全ピボットに変更します.
30 ≤ IPIVOT < 40 :Rook ピボット
32 : スーパーノード内で Rook ピボットがとれないとき, 完全ピボットをとります.
40 ≤ IPIVOT < 50 : 完全ピボット

ISTATIC..... 入力. Pivot を指定したとき, Static Pivot を行うかを示します.
1) = 1 のとき
スーパーノード内で指定したピボットが SPEPSZ より大きくないとき, ピボットとして copysign(spepsz,pivot) DSIGN(SPEPSZ,PIVOT)で
近似します. ピボットの値が 0.0d0 なら SPEPSZ に近似します.
このとき, 以下を設定しなければなりません.
a) EPSZ は EPSZ の標準値以下にしなければなりません.
b) ISCLITERMAX は 10 を設定しスケーリングを行う必要があります.
c)THEPSZ ≥ SPEPSZ でなければなりません.
2) ≠1 のとき
Static Pivot は行いません.

SPEPSZ..... 入力. ISTATIC=1 のとき, Static Pivot として使う値.
THEPSZ ≥ SPEPSZ ≥ EPSZ でなければなりません.
出力. SPEPSZ < EPSZ のときは, 1.0D-10 が設定されます.

W..... 作業域.
出力/入力.
大きさ 4*NZ+6*N の 1 次元配列.
ISW=1, 2 で続けて呼び出すとき,呼び出しの間で必要なデータを受け渡すために使われます. 呼び出しの間で値を変更してはいけません.

IW..... 作業域.
出力/入力.
大きさ 36*N+36+2*NZ+3*(N+1)の 1 次元配列.

ISW=1, 2 で続けて呼び出すとき,呼び出しの間で必要なデータを受け渡すために使われます. 呼び出しの間で値を変更してはいけません.

ICON ..... 出力. コンディションコード.
表 DM_VSSSLU-1 参照

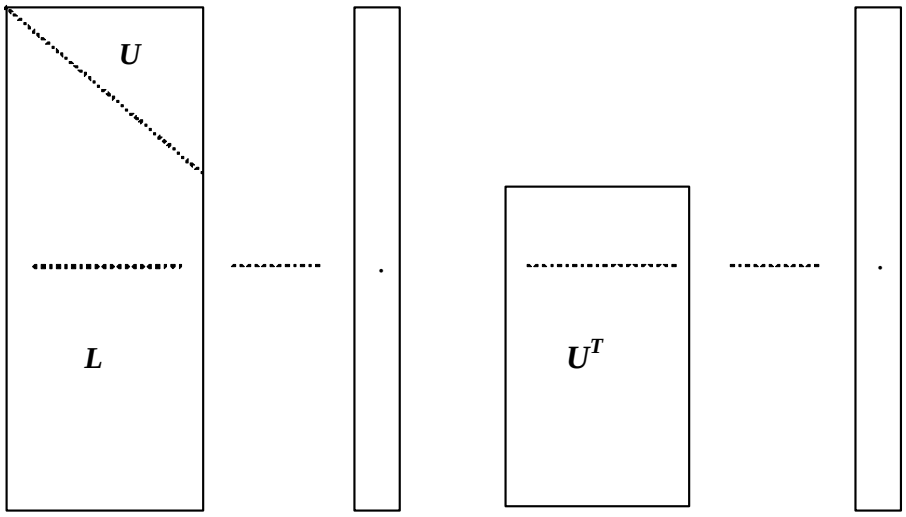


図 DM_VSSSLU-1 分解結果の格納概念図

j = NASSIGN(i) → i 番目の supernode は j 番目に格納されます.

p = NFCNZFACTORL(j) → j 番目の panel は PANELFACTORL の p 番目の要素から NDIM(1,j)×NDIM(2,j)の長さを占めます.

q = NFCNZINDEXL(j) → j 番目の panel の行指標を表すベクトルは NPANELINDEXL の q 番目の要素から NDIM(1,j)の長さを占めます.

panel は大きさ NDIM(1,j)×NDIM(2,j)の配列と見なせます.

panel(s,t) , $s \geq t$, s = 1 , … , NDIM(1,j), t = 1 , … , NDIM(2,j) に分解結果の下三角行列 $\boldsymbol{L}$ が格納されます.

panel(s,t) , $s < t$, s=1 , … , NDIM(2,j), t=1 , … , NDIM(2,j) に単位上三角行列 $\boldsymbol{U}$ の対角ブロック部分が対角要素を除き格納されます.

u = NFCNZFACTORU(j) → j 番目の panel は PANELFACTORU の u 番目の要素から (NDIM(1,j)-NDIM(2,j))×NDIM(2,j)の長さを占めます.

v = NFCNZINDEXU(j) → j 番目の panel の列指標を表すベクトルは NPANELINDEXU の v 番目の要素から NDIM(1,j)の長さを占めます.

panel は大きさ (NDIM(1,j)-NDIM(2,j))×NDIM(2,j)の配列と見なせます.

panel(x,y) , x=1 , … , NDIM(1,j)-NDIM(2,j), y=1 , … , NDIM(2,j)
に分解結果の単位上三角行列 $\boldsymbol{U}$ の転置行列 $\boldsymbol{U}^T$ の対角ブロック部分を除いた部分が

格納されます.
指標の値は行列 $\boldsymbol{A}$ の列番号をもつ node を post order で並び換えた $\boldsymbol{QAQ^T}$ で
の列番号を表します.

表 DM_VSSSLU-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし	ー
10000	ISTATIC=1 のとき小さすぎるピボットを SPEPSZ で置き換える Static Pivot を行いました.	ー
20000	ピボットが相対的に零になりました.	処理を打ち切ります.
20200	行および列の均衡化を行う対角行列の対角要素を求める過程で,元の行列 A の行もしくは列にゼロベクトルがありました. 行列が特異である可能性があります.	
30000	N<1,NZ<0,NFCNZ(N+1)≠NZ+1, NSIZEFACTORL<1, NSIZEFACTORU<1, NSIZEINDEX<1, ISW<1, ISW>2	
30100	NPERM に指定した置換行列が正しくない.	
30200	NROW(j)に格納された i 列目の行指標 k が k<1 または k>N	
30300	i 列目の行指標の数 NFCNZ(i+1)-NFCNZ(i)>n	

コード	意　　味	処理内容
30500	ISTATIC=1 のとき満たすべき条件をみたしていない. EPSZ は標準値 16***u*** より大きかった. または ISCLITERMAX <10 であった. または SPEPSZ>THEPSZ であった.	処理を打ち切ります.
30700	行列 ***A*** が構造的に対称でない.	
31000	PANELFACTORL の大きさ NSIZEFACTORL または NPANELINDEXL および NPANELINDEXU の大きさ NSIZEINDEX または PANELFACTORU の大きさ NSIZEFACTORU が小さすぎます.	NSIZEFACTORL または NSIZEINDEX または NSIZEFACTORU で指定された大きさで PANELFACTORL または NPANELINDEXL および NPANELINDEXU または PANELFACTORU を割り付けて ISW=2 として再度呼び出す.

(3)　使用上の注意

a.　注意

①　直交行列である置換行列***P***の要素$p_{ij}=1$のとき, NPERM(i)=jと表現します.
逆は以下のようにして求めることができます.

```
DO i=1,n
 j=NPERM(i)
 NPERMINV(j)=i
 ENDDO
```

Fill-Reducing OrderingはMETISなどを使って求めることができます.
詳細は“付録1 参考文献一覧表”の[43],[44]を参照願います.

②　分解過程で対角要素の大きさの絶対値の相対零判定値にある値を設定したとすると, この値は次の意味を持っています. すなわち, LU分解の過程で, 対角要素の大きさの絶対値がその値より小さくなった場合に, その値を相対的に零と見なし, ICON=20000として処理を打ち切ります. EPSZの標準値は, 丸め誤差の単位を*u*としたとき, EPSZ=16*u*です.
なお, 対角要素の大きさの絶対値が小さくなっても, 処理を続行させたい場合には, EPSZに極小の値を与えれば良いのですが, 結果の精度は保証されません.
Static Pivotを指定した場合, 対角要素がSPEPSZより小さいとき, SPEPSZで近似してLU分解を行います.

③　分解結果を格納する配列PANELFACTORL, NPANELINDEXL, PANELFACTORU, NPANELINDEXUの必要な大きさは,事前には分かりません. 十分大きな配列を割り当てるか,本ルーチンを呼び出してsymbolic decompositionを行った解析の結果を使って割り当てを行うことができます.
例えば, 大きさ1の1次元配列などを割付けます. そしてその大きさ1などの小さな値をNSIZEFACTORL, NSIZEINDEX, NSIZEFACTORUに指定して, ISW=1

で呼び出します.
symbolic decompositionを行い, ICON=31000で終了し, NSIZEFACTORL, NSIZEINDEX, NSIZEFACTORUに必要な大きさが返却されます. 必要な大きさの配列を割付け直して引数に指定して, ISW=2で呼び出すことで, symbolic decomposition以降の処理を続けることができます. 使用例を参照願います.
LU分解の結果を利用してDM_VSSSLUXを利用して連立1次方程式を解く上での注意は(3)使用上の注意a. 注意⑤参照.

④　列番号に対応するノードを考えます. これをpost orderで順番を並び換えたときの番号の対応関係がNPOSTOに格納されています. post orderのi番目のノードが並び換える前の何番目のノードに対応するかを表します. j=nposto(i)はj番目であることを表します.
①同様にこれは直交行列である置換行列$\boldsymbol{Q}$を表し,行列$\boldsymbol{A}$を$\boldsymbol{QAQ^T}$と並び換えることに相当します.
逆変換$\boldsymbol{Q^T}$は以下のようにして求めることができます.

```
 DO i=1,n
  j=NPOSTO(i)
  NPOSTOINV(j)=i
  ENDDO
```

⑤　本ルーチンで得られたLU分解の結果を使い, DM_VSSSLUXを引き続いて呼びだすことで連立1次方程式$\boldsymbol{Ax}=\boldsymbol{b}$を解くことができます.
このとき, DM_VSSSLUで利用した以下の引数を指定します. 使用例を参照願います.
A, NZ, NROW, NFCNZ, N,
IORDERING, NPERM,
NASSIGN, NSUPNUM,
NFCNZFACTORL, PANELFACTORL,
NSIZEFACTORL, NFCNZINDEXL, NPANELINDEXL,
NSIZEINDEX, NDIM,
NFCNZFACTORU, PANELFACTORU, NSIZEFACTORU,
NFCNZINDEXU, NPANELINDEXU, NPOSTO,
SCLROW,SCLCOL,
IW

b.　使用例

連立 1 次方程式 $\boldsymbol{Ax} = \boldsymbol{f}$ を解きます.
行列は境界条件として立方体の境界で 0 となる楕円型の偏微分方程式に有限差分法を適用して生成されたものを利用します.

$$-\Delta u + a\nabla u + cu = f$$

ここで $a = (a_1, a_2, a_3)$.
行列はサブルーチン init_mat_diag によって生成されます.
対角形式格納法で生成し,圧縮列格納法で格納します. 結果生成された構造的に対称な行列 $\boldsymbol{A}$ に関する連立 1 次方程式を解きます.
(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できま

す. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT  REAL*8 (A-H,O-Z)
      PARAMETER (NORD=39,NX = NORD,NY =NORD ,NZ = NORD,
     $      N = NX*NY*NZ,NXY=NX*NY)
      PARAMETER (K = N+1)
      PARAMETER (NDIAG = 7)
      PARAMETER (NALL=NDIAG*N,
     $   IWL=36*N+36+2*NALL+3*(N+1))
      PARAMETER(IPRINT=0)
      DIMENSION NOFST(NDIAG)
      DIMENSION DIAG(K,NDIAG),DIAG2(K,NDIAG)
      DIMENSION C(K*NDIAG),NROWC(K*NDIAG),NFCNZC(N+1),
     $          WC(K*NDIAG),IWC(2,K*NDIAG)
      DIMENSION A(NDIAG*N),NCOLUMN(K*NDIAG),NFCNZ(N+1),
     $          NPERM(N),W(NDIAG*N+N),
     $          NPOSTO(N),NDIM(2,N),
     $          NASSIGN(N),
     $          IW(IWL)
      REAL*8, DIMENSION(:), ALLOCATABLE ::  PANELFACTORL,PANELFACTORU
      INTEGER*4, DIMENSION(:), ALLOCATABLE :: NPANELINDEXL,
     $                                        NPANELINDEXU
      REAL*8 DUMMYFL,DUMMYFU
      INTEGER*4 NDUMMYIL,NDUMMYIU
      INTEGER*8 NSIZEFACTORL,NSIZEINDEX,
     $          NSIZEFACTORU,
     $          NFCNZFACTORL(N+1),
     $          NFCNZFACTORU(N+1),
     $          NFCNZINDEXL(N+1),
     $          NFCNZINDEXU(N+1)
      DIMENSION X(N),B(N),SOLEX(N),NPERM1(N)
C
      REAL*8 THEPSZ,
     $       EPSR,
     $       SEPSZ,
     $       SCLROW(N),SCLCOL(N)

      INTEGER*4     IPIVOT,ISTATIC,
     $              ISCLITERMAX,
     $              IREFINE,ITERMAX,ITER
```

```
      PRINT *,'    DIRECT METHOD'
      PRINT *,'    FOR SPARSE STRUCTURALLY SYMMETRIC REAL MATRICES'
      PRINT *,'    IN COMPRESSED COLUMN STORAGE'
      PRINT *

      DO I=1,N
      SOLEX(I)=1.0D0
      ENDDO
      PRINT *,'    EXPECTED SOLUTIONS'
      PRINT *,'    X(1) = ',SOLEX(1),' X(N) = ',SOLEX(N)
      PRINT *

      VA1 = 1.0D0
      VA2 = 2.0D0
      VA3 = 3.0D0
      VC =  4.0D0
      XL = 1.0
      YL = 1.0
      ZL = 1.0
      CALL INIT_MAT_DIAG(VA1,VA2,VA3,VC,DIAG,NOFST
     &          ,NX,NY,NZ,XL,YL,ZL,NDIAG,N,K)
C
      DIAG2=0
C
      DO I=1,NDIAG
C
      IF(NOFST(I).LT.0)THEN
      NBASE=-NOFST(I)
      LENGTH=N-NBASE
      DIAG2(1:LENGTH,I)=DIAG(NBASE+1:N,I)
      ELSE
      NBASE=NOFST(I)
      LENGTH=N-NBASE
      DIAG2(NBASE+1:N,I)=DIAG(1:LENGTH,I)
      ENDIF
C
      ENDDO
C
      NUMNZC=1
```

```
C
      DO J=1,N
      NTOPCFGC=1
C
      DO I=NDIAG,1,-1
C
      IF(DIAG2(J,I).NE.0.0D0)THEN
C
      NCOL=J-NOFST(I)
      C(NUMNZC)=DIAG2(J,I)
      NROWC(NUMNZC)=NCOL
C
      IF(NTOPCFGC.EQ.1)THEN
      NFCNZC(J)=NUMNZC
      NTOPCFGC=0
      ENDIF
C
      NUMNZC=NUMNZC+1
C
      ENDIF
      ENDDO
      ENDDO
C
      NFCNZC(N+1)=NUMNZC
      NNZC=NUMNZC-1
C
      CALL DM_VMVSCC(C,NNZC,NROWC,NFCNZC,N,SOLEX,
     $             B,WC,IWC,ICON)
C
C
      X=B
      IORDERING=0
      ISCLITERMAX=10
      ISW=1
      EPSZ=1.0D-16
      NSIZEFACTORL=1
      NSIZEFACTORU=1
      NSIZEINDEX=1
      THEPSZ=1.0D-2
      EPSR=1.0D-8
      SEPSZ=1.0D-10
      IPIVOT=40
```

```
      ISTATIC=1
      IREFINE=1
      ITERMAX=10

      CALL DM_VSSSLU(C,NNZC,NROWC,NFCNZC,N,
     $              ISCLITERMAX,IORDERING,
     $              NPERM,ISW,
     $              NASSIGN,
     $              NSUPNUM,
     $              NFCNZFACTORL,DUMMYFL,
     $              NSIZEFACTORL,NFCNZINDEXL,
     $              NDUMMYIL,NSIZEINDEX,NDIM,
     $              NFCNZFACTORU,DUMMYFU,
     $              NSIZEFACTORU,
     $              NFCNZINDEXU,NDUMMYIU,
     $              NPOSTO,
     $              SCLROW,SCLCOL,
     $              EPSZ,
     $              THEPSZ,
     $              IPIVOT,ISTATIC,SEPSZ,
     $              W,IW,ICON)
      PRINT*,'    ICON=',ICON,' NSIZEFACTORL=',NSIZEFACTORL,
     $       ' NSIZEFACTORU=',NSIZEFACTORU,
     $       'NSIZEINDEX=',NSIZEINDEX
      PRINT*,'    NSUPNUM=',NSUPNUM
      PRINT *
C
      ALLOCATE( PANELFACTORL(NSIZEFACTORL) )
      ALLOCATE( PANELFACTORU(NSIZEFACTORU) )
      ALLOCATE( NPANELINDEXL(NSIZEINDEX) )
      ALLOCATE( NPANELINDEXU(NSIZEINDEX) )
C
      ISW=2
      CALL DM_VSSSLU(C,NNZC,NROWC,NFCNZC,N,
     $              ISCLITERMAX,IORDERING,
     $              NPERM,ISW,
     $              NASSIGN,
     $              NSUPNUM,
     $              NFCNZFACTORL,PANELFACTORL,
     $              NSIZEFACTORL,NFCNZINDEXL,
     $              NPANELINDEXL,NSIZEINDEX,NDIM,
     $              NFCNZFACTORU,PANELFACTORU,
```

```
     $              NSIZEFACTORU,
     $              NFCNZINDEXU,NPANELINDEXU,
     $              NPOSTO,
     $              SCLROW,SCLCOL,
     $              EPSZ,
     $              THEPSZ,
     $              IPIVOT,ISTATIC,SEPSZ,
     $              W,IW,ICON)
      CALL GETTOD(T3)
C
      CALL DM_VSSSLUX(N,
     $              IORDERING,
     $              NPERM,
     $              X,
     $              NASSIGN,
     $              NSUPNUM,
     $              NFCNZFACTORL,PANELFACTORL,
     $              NSIZEFACTORL,NFCNZINDEXL,
     $              NPANELINDEXL,NSIZEINDEX,NDIM,
     $              NFCNZFACTORU,PANELFACTORU,
     $              NSIZEFACTORU,
     $              NFCNZINDEXU,NPANELINDEXU,
     $              NPOSTO,
     $              SCLROW,SCLCOL,
     $              IREFINE,EPSR,ITERMAX,ITER,
     $              C,NNZC,NROWC,NFCNZC,
     $              IW,
     $              ICON)

C
      ERR = ERRNRM(SOLEX,X,N)

      PRINT *,'    COMPUTED VALUES'
      PRINT *,'    X(1) = ',X(1),' X(N) = ',X(N)
      PRINT *
      PRINT *,'    ICON = ',ICON
      PRINT *
      PRINT *,'    N = ',N,' :: NX = ',NX,' NY = ',NY,' NZ = ',NZ
      PRINT *
      PRINT *,'    ERROR = ',ERR
      PRINT *,'    ITER=',ITER
      PRINT *
```

```
      PRINT *

      IF(ERR.LT.1.0D-8.AND.ICON.EQ.0)THEN
         WRITE(*,*)'    ********** OK **********'
      ELSE
         WRITE(*,*)'    ********** NG **********'
      ENDIF

      DEALLOCATE( PANELFACTORL,PANELFACTORU,NPANELINDEXL,
     $            NPANELINDEXU )

      STOP
      END

C =======================================
C     INITIALIZE COEFFICIENT MATRIX
C =======================================
      SUBROUTINE INIT_MAT_DIAG(VA1,VA2,VA3,VC,D_L,OFFSET
     &               ,NX,NY,NZ,XL,YL,ZL,NDIAG,LEN,NDIVP)
      IMPLICIT  REAL*8(A-H,O-Z)
      DIMENSION D_L(NDIVP,NDIAG)
      INTEGER   OFFSET(NDIAG)
C
      IF (NDIAG .LT. 1) THEN
        WRITE (*,*) 'SUBROUTINE INIT_MAT_DIAG:'
        WRITE (*,*) ' NDIAG SHOULD BE GREATER THAN OR EQUAL TO 1'
        RETURN
      ENDIF

!$OMP PARALLEL DEFAULT(PRIVATE)
!$OMP+     SHARED(VA1,VA2,VA3,VC,D_L,OFFSET
!$OMP+      ,NX,NY,NZ,XL,YL,ZL,NDIAG,LEN,NDIVP)

C NDIAG CANNOT BE GREATER THAN 7
      NDIAG_LOC = NDIAG
      IF (NDIAG .GT. 7) NDIAG_LOC = 7

C INITIAL SETTING
      HX = XL/(NX+1)
      HY = YL/(NY+1)
      HZ = ZL/(NZ+1)
```

```
!$OMP DO
      DO I = 1,NDIVP
      DO J = 1,NDIAG
      D_L(I,J) = 0.0
      ENDDO
      ENDDO
!$OMP ENDDO

      NXY = NX*NY

C OFFSET SETTING
!$OMP SINGLE
      L = 1
      IF (NDIAG_LOC .GE. 7) THEN
        OFFSET(L) = -NXY
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 5) THEN
        OFFSET(L) = -NX
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 3) THEN
        OFFSET(L) = -1
        L = L+1
      ENDIF
      OFFSET(L) = 0
      L = L+1
      IF (NDIAG_LOC .GE. 2) THEN
        OFFSET(L) = 1
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 4) THEN
        OFFSET(L) = NX
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 6) THEN
        OFFSET(L) = NXY
      ENDIF
!$OMP END SINGLE

C MAIN LOOP
!$OMP DO
```

```
      DO 100 J = 1,LEN
        JS = J

C DECOMPOSE JS-1 = (K0-1)*NX*NY+(J0-1)*NX+I0-1
        K0 = (JS-1)/NXY+1
        IF (K0 .GT. NZ) THEN
  PRINT*,'ERROR; K0.GH.NZ '
  GOTO 100
  ENDIF
        J0 = (JS-1-NXY*(K0-1))/NX+1
        I0 = JS - NXY*(K0-1) - NX*(J0-1)
        L = 1

        IF (NDIAG_LOC .GE. 7) THEN
          IF (K0 .GT. 1) D_L(J,L) = -(1.0/HZ+0.5*VA3)/HZ
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 5) THEN
          IF (J0 .GT. 1) D_L(J,L) = -(1.0/HY+0.5*VA2)/HY
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 3) THEN
          IF (I0 .GT. 1) D_L(J,L) = -(1.0/HX+0.5*VA1)/HX
          L = L+1
        ENDIF
        D_L(J,L) = 2.0/HX**2+VC
        IF (NDIAG_LOC .GE. 5) THEN
          D_L(J,L) = D_L(J,L) + 2.0/HY**2
          IF (NDIAG_LOC .GE. 7) THEN
            D_L(J,L) = D_L(J,L) + 2.0/HZ**2
          ENDIF
        ENDIF
        L = L+1
        IF (NDIAG_LOC .GE. 2) THEN
          IF (I0 .LT. NX) D_L(J,L) = -(1.0/HX-0.5*VA1)/HX
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 4) THEN
          IF (J0 .LT. NY) D_L(J,L) = -(1.0/HY-0.5*VA2)/HY
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 6) THEN
```

```
            IF (K0 .LT. NZ) D_L(J,L) = -(1.0/HZ-0.5*VA3)/HZ
          ENDIF
 100  CONTINUE
!$OMP ENDDO

!$OMP END PARALLEL

      RETURN
      END

C =======================================
* SOLUTE ERROR
* | X1 - X2 |
C =======================================
      REAL*8 FUNCTION ERRNRM(X1,X2,LEN)
      IMPLICIT  REAL*8 (A-H,O-Z)
      DIMENSION X1(*),X2(*)
C
      S = 0D0
      DO 100 I = 1,LEN
        SS = X1(I) - X2(I)
        S = S + SS * SS
 100  CONTINUE
C
      ERRNRM = SQRT( S )
      RETURN
      END
```

(4) 手法概要

行列に対して行, 列の均衡化を行うスケーリングを施します.
この行列を LU 分解します.
スーパーノードに対応する非ゼロ要素を 2 次元の panel に格納します.
安定化のためのピボットは対角ブロック部分で検索します.
指定した値より絶対値が大きな値が見つかるとピボットとして採用する閾値 THEPSZ を指定できます. 求めたピボットが小さすぎるとき, SPEPSZ で指定した値で近似して LU 分解を続ける static pivot を指定することも可能です.
詳細は“付録 1 参考文献一覧表”の以下の文献を参照願います.
スパースな実行列の LU 分解に関しては[19],[2],[22],[48],[68]を行列の均衡化やピボットに関しては[63],[69]をそれぞれ参照願います.

DM_VSSSLUX

LU 分解された構造的に対称な実スパース行列の連立 1 次方程式
CALL DM_VSSSLUX(N, IORDERING, NPERM, B, NASSIGN, NSUPNUM, NFCNZFACTORL, PANELFACTORL, NSIZEFACTORL, NFCNZINDEXL, NPANELINDEXL, NSIZEINDEX, NDIM, NFCNZFACTORU, PANELFACTORU, NSIZEFACTORU, NFCNZINDEXU, NPANELINDEXU, NPOSTO, SCLROW,SCLCOL, IREFINE, EPSR, ITERMAX, ITER, A, NZ, NROW, NFCNZ, IW, ICON)

(1)　機能

$n \times n$ の構造的に対称な実スパース行列 $\boldsymbol{A}$ に行および列の均衡化を行うスケーリングを行い, そしてスーパーノード内で Pivot をとり, 以下のように LU 分解されています. LU 分解の結果を利用して以下の連立 1 次方程式を解きます.

$$\boldsymbol{Ax=b}$$

行列$\boldsymbol{A}$は以下のように分解されています.

$$\boldsymbol{P_{rs}QPD_rAD_cP^TQ^TP_{cs}=LU}$$

ここで, 構造的に対称な実スパース行列$\boldsymbol{A}$は以下のように変換されています.

$$\boldsymbol{A_1= D_rAD_c}$$

ここで $\boldsymbol{D_r}$ は行のスケーリングを行う対角行列, $\boldsymbol{D_c}$ は列のスケーリングを行う対角行列です.

$$\boldsymbol{A_2=QPA_1P^TQ^T}$$

$\boldsymbol{A_2}$は, スーパーノード内に閉じて指定された pivot を行い,行および列の入れ換えを行い LU 分解されています.

$\boldsymbol{P_{rs}}$, $\boldsymbol{P_{cs}}$ はそれぞれ行の入れ換えおよび列の入れ換えを示す直交行列です. 実際の入れ換えはスーパーノードに属する限られた行列の部分で行われます.

ただし, $\boldsymbol{P}$ は,$\boldsymbol{A}$ の非ゼロパターンに対して求めた ordering による行列要素の並び換えを示す置換行列で, $\boldsymbol{Q}$ は post order による行列要素の並び換えを示す置換行列を示します. $\boldsymbol{P}$, $\boldsymbol{Q}$ は直交行列です.

$\boldsymbol{L}$ は下三角行列, $\boldsymbol{U}$ は単位上三角行列です.

解の反復改良で精度を改良することを指定できます.

(2)　パラメタ

N..... 入力. 行列 $\boldsymbol{A}$ の次数 n.

IORDERING..... 入力. 1 のとき, NPERM に ordering を指定して LU 分解されたことを示します. $\boldsymbol{PA_1P^T}$ を LU 分解しました.

1 以外のとき, ordering は指定されていません.
((3)使用上の注意 a. 注意①参照).

NPERM ..... 入力. IORDERING=1 のとき使用する置換行列をベクトルで指定します.
NPERM(N)なる 1 次元配列.
((3)使用上の注意 a. 注意②参照).

B ..... 入力. ***Ax=b*** の右辺定数ベクトル.
出力. 解ベクトル.
B(N)なる 1 次元配列.

NASSIGN..... 入力. 各 supernode に対する ***L***, ***U*** は圧縮して 2 次元の panel に格納します. この panel を PANELFACTORL および PANELFACTORU の 1 次元部分配列として順番に割り付けたとき何番目になるかを示します. これらの指標ベクトルも同じように NPANELINDEXL, NPANELINDEXU に割り付けます. j=NASSIGN(i)のとき, i 番目の supernode を j 番目に割り付けたことを示します.
分解結果の格納方法については図 DM_VSSSLUX-1 を参照してください.
NASSIGN(N)なる 1 次元配列.

NSUPNUM ..... 入力. supernode の総数. (≤n)

NFCNZFACTORL 入力. 構造的に対称な実スパース行列の LU 分解の結果の行列 ***L*** および ***U*** はスーパーノードに対応しておのおの求めます. 各 supernode に対応する ***L*** の列ベクトルは, 共通の行指標ベクトルを持つように圧縮してブロック対角部分に ***U*** の対応部分を含めて 2 次元の panel に格納します. この panel を PANELFACTORL の 1 次元部分配列として順番に割り付けたとき, i 番目の panel の先頭要素 panel(1,1) が 1 次元配列の PANELFACTORL の何番目の要素になるかを示します.
NFCNZFACTORL(N+1)なる 8 バイト整数型の 1 次元配列.
結果の格納方法については図 DM_VSSSLUX-1 を参照してください.

PANELFACTORL 入力. 各 supernode の分解結果に対応する行列 ***L*** の複数の列ベクトルと行列 ***U*** のブロック対角部分について, 共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します. panel を順番に割り付けます. i 番目の supernode に対応する panel が何番目の位置に割り当てられるかは j=NASSIGN(i)から分かります. その先頭位置は NFCNZFACTORL(j)に格納されています. panel ごとに分解結果が格納されます.
i 番目に割付けられた panel の大きさは NDIM(1,i)×NDIM(2,i)の 2 次元配列と見なせます. i 番目の panel の panel(s,t), $s \geq t$, s = 1,…,NDIM(1,i), t=1,…,NDIM(2,i)に下三角行列 ***L*** が格納されます. panel の panel(s,t), s<t, t=1,…,NDIM(2,i)には単位上三角行列 ***U*** の対角要素を除いたブロック対角部分が格納されます.
PANELFACTORL(NSIZEFACTORL)なる 1 次元配列.
結果の格納方法については図 DM_VSSSLUX-1 を参照してください.

NSIZEFACTORL . 入力. PANELFACTORL の大きさを示す. 8 バイト整数型.

NFCNZINDEXL ... 入力. 各 supernode の分解結果に対応する行列 ***L*** の複数の列ベクトルと行列 ***U*** のブロック対角部分について，共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します. この行指標ベクトルを NPANELINDEXL に順番に割り付けたとき i 番目の行指標ベクトルの先頭要素が 1 次元配列である NPANELINDEXL の何番目の要素になるかを示します.
NFCNZINDEXL(N+1)なる 8 バイト整数型の 1 次元配列.

NPANELINDEXL 入力. 各 supernode の分解結果に対応する行列 ***L*** の複数の列ベクトルと行列 ***U*** のブロック対角部分について，共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します. この行指標ベクトルを NPANELINDEXL に順番に割り付けます. i 番目の supernode に対応する panel の行の指標ベクトルが何番目に割り当てられるかは j=NASSIGN(i)から分かります. その先頭位置は NFCNZINDEXL(j)に格納されています.
この行指標は post order で並び換えたときの行の番号です.
NPANELINDEXL(NSIZEINDEX)なる 1 次元配列.
結果の格納方法については図 DM_VSSSLUX-1 を参照してください.

NSIZEINDEX ..... 入力. NPANELINDEXL および NPANELINDEXU の大きさを示す.
8 バイト整数型.

NDIM ..... 入力. NDIM(1,i)および NDIM(2,i)は行列 ***L*** に関して i 番目に割り付けられた panel の 1 次元目と 2 次元目の大きさを示します.
NDIM(1,i)-NDIM(2,i)および NDIM(2,i)は行列 ***U*** に関して割り付けられた panel を転置したときの 1 次元目と 2 次元目の大きさを示します.
NDIM(2,N)なる 2 次元配列.
結果の格納方法については図 DM_VSSSLUX-1 を参照してください.

NFCNZFACTORU 入力. 構造的に対称な実スパース行列の LU 分解の結果の行列 ***U*** に関しては, 各 supernode に対応する ***U*** の行ベクトルは, ブロック対角部分を除いて共通の列指標ベクトルを持つように圧縮し, 転置して 2 次元の panel に格納します. この panel を PANELFACTORU の 1 次元部分配列として順番に割り付けたとき, i 番目の panel の先頭要素 panel(1,1)が 1 次元配列の PANELFACTORU の何番目の要素になるかを示します.
NFCNZFACTORU(N+1)なる 8 バイト整数型の 1 次元配列.
結果の格納方法については図 DM_VSSSLUX-1 を参照してください.

PANELFACTORU 入力. 各 supernode の分解結果に対応する行列 ***U*** のブロック対角部分を除いた複数の行ベクトルについて，共通の列指標ベクトルを持つように圧縮し, 転置して 2 次元の panel に格納します. panel を順番に割り付けます. i 番目の supernode に対応する panel が何番目の位置に割り当てられるかは j=NASSIGN(i) から分かります. その先頭位置は NFCNZFACTORU(j)に格納されています. panel ごとに分解結果が格納されます.
i 番目に割付けられた panel の大きさは {NDIM(1,i)-NDIM(2,i)}×NDIM(2,i)の 2 次元配列と見なせます. i 番目の

panel の panel(s,t), s=1,…,NDIM(1,i)- NDIM(2,i), t=1,…,NDIM(2,i)に単位上三角行列 ***U*** のブロック対角部分を除いた部分が行ベクトルを圧縮し, 転置したものが格納されます.
PANELFACTORU(NSIZEFACTORU)なる 1 次元配列.
結果の格納方法については図 DM_VSSSLUX-1 を参照してください.

NSIZEFACTORU . 入力. PANELFACTORU の大きさを示す. 8 バイト整数型.

NFCNZINDEXU... 入力. 各 supernode の分解結果に対応する行列 ***U*** は対角ブロック部分を除いて複数の行ベクトルについて, 共通の列指標ベクトルを持つように圧縮し, 転置した形で 2 次元の panel に格納します. 対角ブロック部分も含んだこの列指標ベクトルを NPANELINDEXU に順番に割り付けたとき i 番目の列指標ベクトルの先頭要素が 1 次元配列である NPANELINDEXU の何番目の要素になるかを示します.
NFCNZINDEXU(N+1)なる 8 バイト整数型の 1 次元配列.
結果の格納方法については図 DM_VSSSLUX-1 を参照してください.

NPANELINDEXU 入力. 各 supernode の分解結果に対応する行列 ***U*** の対角ブロックを除いた部分を転置して圧縮して 2 次元の panel に格納します. 対応する列指標ベクトルには対角ブロック部分を含めたものを NPANELINDEXU に順番に割り付けます. i 番目の supernode に対応する panel の行の指標ベクトルが何番目に割り当てられるかは j=NASSIGN(i)から分かります. その先頭位置は NFCNZINDEXU(j)に格納されています.
この行指標は post order で並び換えたときの列の番号です.
NPANELINDEXU(NSIZEINDEX)なる 1 次元配列.
結果の格納方法については図 DM_VSSSLUX-1 を参照してください.

NPOSTO ..... 入力. post order で i 番目の node が行列 ***A*** の何番目の列番号に対応するかを表す 1 次元ベクトル.
((3)使用上の注意 a. 注意③参照).

SCLROW ..... 入力. スケーリング対角行列 $\boldsymbol{D}_r$. 対角要素が 1 次元配列に格納されます.
SCLROW(N)なる 1 次元配列.

SCLCOL..... 入力. スケーリング対角行列 $\boldsymbol{D}_c$. 対角要素が 1 次元配列に格納されます.
SCLCOL(N)なる 1 次元配列.

IREFINE..... 入力. LU 分解結果を利用して解を求めるときに, 解の反復改良を行うかどうかを示す. 残差ベクトルを計算するときに 4 倍精度で計算します.
=1 : 解の反復改良を行う. 反復改良して得られる残差ベクトル $\boldsymbol{r}_k$ の絶対値の差分がひとつ前の差分の 1/4 より大きくなるまで, 反復改良を行います.
≠1: 解の反復改良を行わない.

EPSR ..... 入力. 解の残差ベクトル ***b-Ax*** の絶対値が, ***b*** の絶対値に対して十分小さいかを判定する判定値
EPSR ≤ 0.0 のとき 1.0D-6 が設定されます.

ITERMAX..... 入力. (≥ 1) 反復改良を行うときの最大反復回数.

ITER..... 出力. 反復改良を行った回数.

A.....入力. 構造的に対称な実スパースな係数行列 ***A*** を圧縮列格納法で A(1:NZ)に格納します.
A(NZ)なる 1 次元配列.
圧縮列格納法については,実スパース行列と実ベクトル(圧縮列格納法), DM_VMVSCC の図 DM_VMVSCC-1 を参照してください.

NZ .....入力. 構造的に対称な実スパースな係数行列 ***A*** の格納する非零要素の総数.

NROW.....入力. 圧縮列格納法で使用される行指標で A に格納される要素が何番目の行ベクトルに属するかを示します.
NROW(NZ)なる 1 次元配列.

NFCNZ.....入力. 行列 ***A*** の各列の格納する非零要素を列方向に圧縮して順次配列 A に格納するとき,対応する列の最初の格納する非零要素が格納される位置を表します.
NFCNZ(N+1)=NZ+1.
NFCNZ(N+1)なる 1 次元配列.

IW.....作業域.
入力.
大きさ 36*N+36+2*NZ+3*(N+1)の 1 次元配列.
構造的に対称な実スパース行列の LU 分解を行う DM_VSSSLU からのデータの受け渡しに使われます. 呼び出しの間で値を変更してはいけません.

ICON.....出力.コンディションコード.
表 DM_VSSSLUX-1 参照

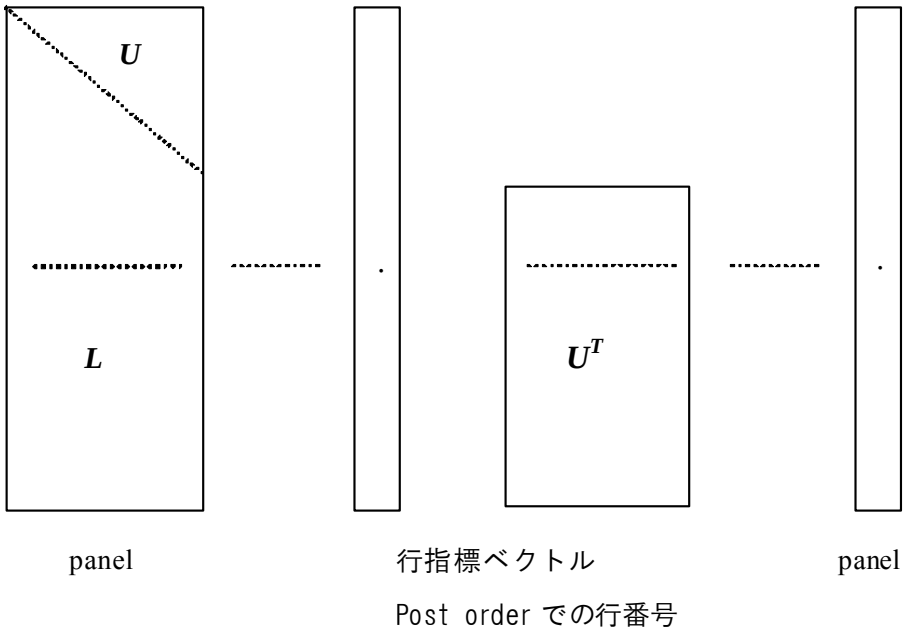


図 DM_VSSSLUX-1 分解結果の格納概念図

j = NASSIGN(i)　　→ i 番目の supernode は j 番目に格納されます.

p = NFCNZFACTORL(j) → j 番目の panel は PANELFACTORL の p 番目の要素から NDIM(1,j)×NDIM(2,j)の長さを占めます.

q = NFCNZINDEXL(j) → j 番目の panel の行指標を表すベクトルは NPANELINDEXL の q 番目の要素から NDIM(1,j)の長さを占めます.

panel は大きさ NDIM(1,j)×NDIM(2,j)の配列と見なせます.

panel(s,t) , $s \geq t$, s = 1 , … , NDIM(1,j), t=1 , … , NDIM(2,j) に分解結果の下三角行列 $\boldsymbol{L}$ が格納されます.

panel(s,t) , $s < t$, s = 1 , … , NDIM(2,j), t=1 , … , NDIM(2,j) に単位上三角行列 $\boldsymbol{U}$ の対角ブロック部分が対角要素を除き格納されます.

u = NFCNZFACTORU(j) → j 番目の panel は PANELFACTORU の u 番目の要素から (NDIM(1,j)-NDIM(2,j))×NDIM(2,j)の長さを占めます.

v = NFCNZINDEXU(j) → j 番目の panel の列指標を表すベクトルは NPANELINDEXU の v 番目の要素から NDIM(1,j)の長さを占めます.

panel は大きさ(NDIM(1,j)-NDIM(2,j))×NDIM(2,j)の配列と見なせます.

panel(x,y) , x=1 , … , NDIM(1,j)-NDIM(2,j), y=1 , … , NDIM(2,j)

に分解結果の単位上三角行列 $\boldsymbol{U}$ の転置行列 $\boldsymbol{U}^T$ の対角ブロック部分を除いた部分が格納されます.

指標の値は行列 $\boldsymbol{A}$ の列番号をもつ node を post order で並び換えた $\boldsymbol{QAQ}^T$ での列番号を表します.

表 DM_VSSSLUX-1　コンディションコード使用上の注意

	意　　味	処理内容
0	エラーなし	ー
20400	LU 分解された行列の対角要素にゼロがあります.	処理を打ち切ります.
20500	求めた解ベクトルに対する残差ベクトルのノルムが方程式 $\boldsymbol{Ax}=\boldsymbol{b}$ の右辺ベクトルのノルムの EPSR 倍より大きい. 係数行列は特異に近い可能性があります.	
30000	N<1,NZ<0,NFCNZ(N+1)≠NZ+1, NSIZEFACTORL<1, NSIZEFACTORU<1, NSIZEINDEX<1, IREFINE=1 のとき ITERMAX<1	
30100	NPERM に指定した置換行列が正しくない.	
30200	NROW(j)に格納された i 列目の列指標 k が k<1 または k>N	
30300	i 列目の行指標の数 NFCNZ(i+1)-NFCNZ(i)>n	

(3)　使用上の注意

a.　注意

①　DM_VSSSLUでLU分解を行った結果を利用します.
DM_VSSSLUの(3)使用上の注意a. 注意⑤参照やDM_VSSSLUXの使用例を参照願います.

②　直交行列である置換行列$\boldsymbol{P}$の要素$p_{ij}=1$のとき, NPERM(i)=jと表現します.
逆は以下のようにして求めることができます.

```
DO i=1,n
  j=NPERM(i)
  NPERMINV(j)=i
  ENDDO
```

③　列番号に対応するノードを考えます. これをpost orderで順番を並び換えたときの番号の対応関係がNPOSTOに格納されています. post orderのi番目のノードが並び換える前の何番目のノードに対応するかを表します. j=nposto(i)はj番目であることを表します.
②同様にこれは直交行列である置換行列$\boldsymbol{Q}$を表し,行列$\boldsymbol{A}$を$\boldsymbol{QAQ}^T$と並び換えることに相当します.
逆変換$\boldsymbol{Q}^T$は以下のようにして求めることができます.

```
DO i=1,n
  j=NPOSTO(i)
  NPOSTOINV(j)=i
  ENDDO
```

b.　使用例

連立 1 次方程式 $\boldsymbol{Ax}=\boldsymbol{f}$ を解きます.
行列は境界条件として立方体の境界で 0 となる楕円型の偏微分方程式に有限差分法を適用して生成されたものを利用します.

$$-\Delta u + a\nabla u + cu = f$$

ここで $a=(a_1, a_2, a_3)$.
行列はサブルーチン init_mat_diag によって生成されます.
対角形式格納法で生成し, 圧縮列格納法で格納します. 結果生成された構造的に対称な行列 $\boldsymbol{A}$ に関する連立 1 次方程式を解きます.
(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT  REAL*8 (A-H,O-Z)
      PARAMETER (NORD=39,NX = NORD,NY =NORD ,NZ = NORD,
     $      N = NX*NY*NZ,NXY=NX*NY)
      PARAMETER (K = N+1)
      PARAMETER (NDIAG = 7)
      PARAMETER (NALL=NDIAG*N,
```

```
$   IWL=36*N+36+2*NALL+3*(N+1))
 PARAMETER(IPRINT=0)
 DIMENSION NOFST(NDIAG)
 DIMENSION DIAG(K,NDIAG),DIAG2(K,NDIAG)
 DIMENSION C(K*NDIAG),NROWC(K*NDIAG),NFCNZC(N+1),
$          WC(K*NDIAG),IWC(2,K*NDIAG)
 DIMENSION A(NDIAG*N),NCOLUMN(K*NDIAG),NFCNZ(N+1),
$          NPERM(N),W(NDIAG*N+N),
$          NPOSTO(N),NDIM(2,N),
$          NASSIGN(N),
$          IW(IWL)
 REAL*8, DIMENSION(:), ALLOCATABLE ::  PANELFACTORL,PANELFACTORU
 INTEGER*4, DIMENSION(:), ALLOCATABLE :: NPANELINDEXL,
$                                        NPANELINDEXU
 REAL*8 DUMMYFL,DUMMYFU
 INTEGER*4 NDUMMYIL,NDUMMYIU
 INTEGER*8 NSIZEFACTORL,NSIZEINDEX,
$          NSIZEFACTORU,
$          NFCNZFACTORL(N+1),
$          NFCNZFACTORU(N+1),
$          NFCNZINDEXL(N+1),
$          NFCNZINDEXU(N+1)
 DIMENSION X(N),B(N),SOLEX(N),NPERM1(N)
C
 REAL*8 THEPSZ,
$       EPSR,
$       SEPSZ,
$       SCLROW(N),SCLCOL(N)

 INTEGER*4     IPIVOT,ISTATIC,
$              ISCLITERMAX,
$              IREFINE,ITERMAX,ITER



 PRINT *,'    DIRECT METHOD'
 PRINT *,'    FOR SPARSE STRUCTURALLY SYMMETRIC REAL MATRICES'
 PRINT *,'    IN COMPRESSED COLUMN STORAGE'
 PRINT *

 DO I=1,N
 SOLEX(I)=1.0D0
```

```
      ENDDO
      PRINT *,'    EXPECTED SOLUTIONS'
      PRINT *,'    X(1) = ',SOLEX(1),' X(N) = ',SOLEX(N)
      PRINT *

      VA1 = 1.0D0
      VA2 = 2.0D0
      VA3 = 3.0D0
      VC =  4.0D0
      XL = 1.0
      YL = 1.0
      ZL = 1.0
      CALL INIT_MAT_DIAG(VA1,VA2,VA3,VC,DIAG,NOFST
     &          ,NX,NY,NZ,XL,YL,ZL,NDIAG,N,K)
C
      DIAG2=0
C
      DO I=1,NDIAG
C
      IF(NOFST(I).LT.0)THEN
      NBASE=-NOFST(I)
      LENGTH=N-NBASE
      DIAG2(1:LENGTH,I)=DIAG(NBASE+1:N,I)
      ELSE
      NBASE=NOFST(I)
      LENGTH=N-NBASE
      DIAG2(NBASE+1:N,I)=DIAG(1:LENGTH,I)
      ENDIF
C
      ENDDO
C
      NUMNZC=1
C
      DO J=1,N
      NTOPCFGC=1
C
      DO I=NDIAG,1,-1
C
      IF(DIAG2(J,I).NE.0.0D0)THEN
C
      NCOL=J-NOFST(I)
      C(NUMNZC)=DIAG2(J,I)
```

```
      NROWC(NUMNZC)=NCOL
C
      IF(NTOPCFGC.EQ.1)THEN
      NFCNZC(J)=NUMNZC
      NTOPCFGC=0
      ENDIF
C
      NUMNZC=NUMNZC+1
C
      ENDIF
      ENDDO
      ENDDO
C
      NFCNZC(N+1)=NUMNZC
      NNZC=NUMNZC-1
C
      CALL DM_VMVSCC(C,NNZC,NROWC,NFCNZC,N,SOLEX,
     $             B,WC,IWC,ICON)
C
C
      X=B
      IORDERING=0
      ISCLITERMAX=10
      ISW=1
      EPSZ=1.0D-16
      NSIZEFACTORL=1
      NSIZEFACTORU=1
      NSIZEINDEX=1
      THEPSZ=1.0D-2
      EPSR=1.0D-8
      SEPSZ=1.0D-10
      IPIVOT=40
      ISTATIC=1
      IREFINE=1
      ITERMAX=10

      CALL DM_VSSSLU(C,NNZC,NROWC,NFCNZC,N,
     $              ISCLITERMAX,IORDERING,
     $              NPERM,ISW,
     $              NASSIGN,
     $              NSUPNUM,
     $              NFCNZFACTORL,DUMMYFL,
```

```
     $              NSIZEFACTORL,NFCNZINDEXL,
     $              NDUMMYIL,NSIZEINDEX,NDIM,
     $              NFCNZFACTORU,DUMMYFU,
     $              NSIZEFACTORU,
     $              NFCNZINDEXU,NDUMMYIU,
     $              NPOSTO,
     $              SCLROW,SCLCOL,
     $              EPSZ,
     $              THEPSZ,
     $              IPIVOT,ISTATIC,SEPSZ,
     $              W,IW,ICON)
      PRINT*,'    ICON=',ICON,' NSIZEFACTORL=',NSIZEFACTORL,
     $       ' NSIZEFACTORU=',NSIZEFACTORU,
     $       'NSIZEINDEX=',NSIZEINDEX
      PRINT*,'    NSUPNUM=',NSUPNUM
      PRINT *
C
      ALLOCATE( PANELFACTORL(NSIZEFACTORL) )
      ALLOCATE( PANELFACTORU(NSIZEFACTORU) )
      ALLOCATE( NPANELINDEXL(NSIZEINDEX) )
      ALLOCATE( NPANELINDEXU(NSIZEINDEX) )
C
      ISW=2
      CALL DM_VSSSLU(C,NNZC,NROWC,NFCNZC,N,
     $              ISCLITERMAX,IORDERING,
     $              NPERM,ISW,
     $              NASSIGN,
     $              NSUPNUM,
     $              NFCNZFACTORL,PANELFACTORL,
     $              NSIZEFACTORL,NFCNZINDEXL,
     $              NPANELINDEXL,NSIZEINDEX,NDIM,
     $              NFCNZFACTORU,PANELFACTORU,
     $              NSIZEFACTORU,
     $              NFCNZINDEXU,NPANELINDEXU,
     $              NPOSTO,
     $              SCLROW,SCLCOL,
     $              EPSZ,
     $              THEPSZ,
     $              IPIVOT,ISTATIC,SEPSZ,
     $              W,IW,ICON)
      CALL GETTOD(T3)
C
```

```
      CALL DM_VSSSLUX(N,
     $              IORDERING,
     $              NPERM,
     $              X,
     $              NASSIGN,
     $              NSUPNUM,
     $              NFCNZFACTORL,PANELFACTORL,
     $              NSIZEFACTORL,NFCNZINDEXL,
     $              NPANELINDEXL,NSIZEINDEX,NDIM,
     $              NFCNZFACTORU,PANELFACTORU,
     $              NSIZEFACTORU,
     $              NFCNZINDEXU,NPANELINDEXU,
     $              NPOSTO,
     $              SCLROW,SCLCOL,
     $              IREFINE,EPSR,ITERMAX,ITER,
     $              C,NNZC,NROWC,NFCNZC,
     $              IW,
     $              ICON)

C
      ERR = ERRNRM(SOLEX,X,N)

      PRINT *,'    COMPUTED VALUES'
      PRINT *,'    X(1) = ',X(1),' X(N) = ',X(N)
      PRINT *
      PRINT *,'    ICON = ',ICON
      PRINT *
      PRINT *,'    N = ',N,' :: NX = ',NX,' NY = ',NY,' NZ = ',NZ
      PRINT *
      PRINT *,'    ERROR = ',ERR
      PRINT *,'    ITER=',ITER
      PRINT *
      PRINT *

      IF(ERR.LT.1.0D-8.AND.ICON.EQ.0)THEN
         WRITE(*,*)'    ********** OK **********'
      ELSE
         WRITE(*,*)'    ********** NG **********'
      ENDIF

      DEALLOCATE( PANELFACTORL,PANELFACTORU,NPANELINDEXL,
     $            NPANELINDEXU )
```

```
      STOP
      END

C =======================================
C     INITIALIZE COEFFICIENT MATRIX
C =======================================
      SUBROUTINE INIT_MAT_DIAG(VA1,VA2,VA3,VC,D_L,OFFSET
     &               ,NX,NY,NZ,XL,YL,ZL,NDIAG,LEN,NDIVP)
      IMPLICIT  REAL*8(A-H,O-Z)
      DIMENSION D_L(NDIVP,NDIAG)
      INTEGER   OFFSET(NDIAG)
C
      IF (NDIAG .LT. 1) THEN
        WRITE (*,*) 'SUBROUTINE INIT_MAT_DIAG:'
        WRITE (*,*) ' NDIAG SHOULD BE GREATER THAN OR EQUAL TO 1'
        RETURN
      ENDIF

!$OMP PARALLEL DEFAULT(PRIVATE)
!$OMP+     SHARED(VA1,VA2,VA3,VC,D_L,OFFSET
!$OMP+      ,NX,NY,NZ,XL,YL,ZL,NDIAG,LEN,NDIVP)

C NDIAG CANNOT BE GREATER THAN 7
      NDIAG_LOC = NDIAG
      IF (NDIAG .GT. 7) NDIAG_LOC = 7

C INITIAL SETTING
      HX = XL/(NX+1)
      HY = YL/(NY+1)
      HZ = ZL/(NZ+1)

!$OMP DO
      DO I = 1,NDIVP
      DO J = 1,NDIAG
      D_L(I,J) = 0.0
      ENDDO
      ENDDO
!$OMP ENDDO

      NXY = NX*NY
```

```
C OFFSET SETTING
!$OMP SINGLE
      L = 1
      IF (NDIAG_LOC .GE. 7) THEN
        OFFSET(L) = -NXY
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 5) THEN
        OFFSET(L) = -NX
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 3) THEN
        OFFSET(L) = -1
        L = L+1
      ENDIF
      OFFSET(L) = 0
      L = L+1
      IF (NDIAG_LOC .GE. 2) THEN
        OFFSET(L) = 1
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 4) THEN
        OFFSET(L) = NX
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 6) THEN
        OFFSET(L) = NXY
      ENDIF
!$OMP END SINGLE

C MAIN LOOP
!$OMP DO
      DO 100 J = 1,LEN
        JS = J

C DECOMPOSE JS-1 = (K0-1)*NX*NY+(J0-1)*NX+I0-1
        K0 = (JS-1)/NXY+1
        IF (K0 .GT. NZ) THEN
  PRINT*,'ERROR; K0.GH.NZ '
  GOTO 100
  ENDIF
        J0 = (JS-1-NXY*(K0-1))/NX+1
```

```
          I0 = JS - NXY*(K0-1) - NX*(J0-1)
          L = 1

          IF (NDIAG_LOC .GE. 7) THEN
            IF (K0 .GT. 1) D_L(J,L) = -(1.0/HZ+0.5*VA3)/HZ
            L = L+1
          ENDIF
          IF (NDIAG_LOC .GE. 5) THEN
            IF (J0 .GT. 1) D_L(J,L) = -(1.0/HY+0.5*VA2)/HY
            L = L+1
          ENDIF
          IF (NDIAG_LOC .GE. 3) THEN
            IF (I0 .GT. 1) D_L(J,L) = -(1.0/HX+0.5*VA1)/HX
            L = L+1
          ENDIF
          D_L(J,L) = 2.0/HX**2+VC
          IF (NDIAG_LOC .GE. 5) THEN
            D_L(J,L) = D_L(J,L) + 2.0/HY**2
            IF (NDIAG_LOC .GE. 7) THEN
              D_L(J,L) = D_L(J,L) + 2.0/HZ**2
            ENDIF
          ENDIF
          L = L+1
          IF (NDIAG_LOC .GE. 2) THEN
            IF (I0 .LT. NX) D_L(J,L) = -(1.0/HX-0.5*VA1)/HX
            L = L+1
          ENDIF
          IF (NDIAG_LOC .GE. 4) THEN
            IF (J0 .LT. NY) D_L(J,L) = -(1.0/HY-0.5*VA2)/HY
            L = L+1
          ENDIF
          IF (NDIAG_LOC .GE. 6) THEN
            IF (K0 .LT. NZ) D_L(J,L) = -(1.0/HZ-0.5*VA3)/HZ
          ENDIF
 100  CONTINUE
!$OMP ENDDO

!$OMP END PARALLEL

          RETURN
          END
```

```
C ========================================
* SOLUTE ERROR
* | X1 - X2 |
C ========================================
      REAL*8 FUNCTION ERRNRM(X1,X2,LEN)
      IMPLICIT  REAL*8 (A-H,O-Z)
      DIMENSION X1(*),X2(*)
C
      S = 0D0
      DO 100 I = 1,LEN
        SS = X1(I) - X2(I)
        S = S + SS * SS
 100  CONTINUE
C
      ERRNRM = SQRT( S )
      RETURN
      END
```

DM_VSSSS

構造的に対称な実スパース行列の連立 1 次方程式 (LU 分解法)
CALL DM_VSSSS(A, NZ, NROW, NFCNZ, N, ISCLITERMAX, IORDERING, NPERM, ISW, B, NASSIGN, NSUPNUM, NFCNZFACTORL, PANELFACTORL, NSIZEFACTORL, NFCNZINDEXL, NPANELINDEXL, NSIZEINDEX, NDIM, NFCNZFACTORU, PANELFACTORU, NSIZEFACTORU, NFCNZINDEXU, NPANELINDEXU, NPOSTO, SCLROW, SCLCOL, EPSZ, THEPSZ, IPIVOT, ISTATIC, SPEPSZ, IREFINE, EPSR, ITERMAX, ITER, W, IW, ICON)

(1)　機能

$n \times n$ の構造的に対称な実スパース行列 $\boldsymbol{A}$ に行および列の均衡化を行うスケーリングを行います. スーパーノードの対角ブロック内で指定した Pivot をとり, LU 分解し解きます.
(構造的に対称な実スパース行列の非零要素はそれと対称な位置に非零要素を持ち, その値は必ずしも等しくはない行列です.)

$$\boldsymbol{Ax=b}$$

構造的に対称な実スパース行列 $\boldsymbol{A}$ は以下のように変換できます.

$$\boldsymbol{A_1= D_rAD_c}$$

ここで $\boldsymbol{D_r}$ は行のスケーリングを行う対角行列, $\boldsymbol{D_c}$ は列のスケーリングを行う対角行列です.

$$\boldsymbol{A_2=QPA_1P^TQ^T}$$

$\boldsymbol{A_2}$ は, スーパーノードの対角ブロックで指定された pivot を行い,行および列の入れ換えを行い LU 分解されます.
ただし, $\boldsymbol{P}$ は,$\boldsymbol{A}$ の非ゼロパターンに対して求めた ordering による行列要素の並び換えを示す置換行列で, $\boldsymbol{Q}$ は post order による行列要素の並び換えを示す置換行列を示します. $\boldsymbol{P}$, $\boldsymbol{Q}$ は直交行列です.

LU 分解の結果の行列 $\boldsymbol{L}$ と $\boldsymbol{U}$ の非零要素のパターンはそれぞれに対して対称になります. $\boldsymbol{L}$ は下三角行列, $\boldsymbol{U}$ は単位上三角行列です.

Pivot 処理で, 閾値 THEPSZ より絶対値が大きな Pivot を見つけることが出来なかったとき, 対角ブロック内の絶対値が最大の要素を Pivot の候補とします. この値が小さ過ぎるときに Static Pivot を与えて近似的に LU 分解を続けることができます.
そして LU 分解の結果を利用して, 解を求めます.
また, 解の反復改良で精度を改良することを指定できます.

(2)　パラメタ

A.....入力. 構造的に対称な実スパースな係数行列 $\boldsymbol{A}$ を圧縮列格納法で A(1:NZ)に格納します.
A(NZ)なる 1 次元配列.
圧縮列格納法については,実スパース行列と実ベクトル(圧縮列格納法), DM_VMVSCC の図 DM_VMVSCC-1 を参照してください.

NZ .....入力. 構造的に対称な実スパースな係数行列 $\boldsymbol{A}$ の格納する非零要素の総数.

NROW.....入力. 圧縮列格納法で使用される行指標で A に格納される要素が何番目の行ベクトルに属するかを示します.
NROW(NZ)なる 1 次元配列.

NFCNZ.....入力. 行列 $\boldsymbol{A}$ の各列の格納する非零要素を列方向に圧縮して順次配列 A に格納するとき,対応する列の最初の格納する非零要素が格納される位置を表します.
NFCNZ(N+1)=NZ+1.
NFCNZ(N+1)なる 1 次元配列.

N.....入力. 行列 $\boldsymbol{A}$ の次数 n.

ISCLITERMAX....入力. 係数行列のスケーリングを行う対角行列 $\boldsymbol{D_r}$ と $\boldsymbol{D_c}$ を反復して求める反復回数.
ISCLITERMAX ≤ 0 のときスケーリングは行わすに, $\boldsymbol{D_r}$ および $\boldsymbol{D_c}$ は単位行列が設定されます.
ISCLITERMAX ≥ 10 のときは 10 回を上限値とします.

IORDERING.....入力. ordering を表す置換行列 $\boldsymbol{P}$ で $\boldsymbol{PAP^T}$ と変換した行列を LU 分解するかを指定します. 置換行列は直交行列です.
1 のとき, PAP^T と変換した行列を LU 分解します.
1 以外のとき, 行列をそのまま LU 分解します.
((3)使用上の注意 a. 注意①参照).

NPERM .....入力. IORDERING=1 のとき使用する置換行列をベクトルで指定します.
NPERM(N)なる 1 次元配列.
((3)使用上の注意 a. 注意①参照).

ISW .....入力. 呼び出しに関する制御情報を示します.
1) 1 のとき
初回の呼び出し.Symbolic decomposition を行い, 必要な領域が割り当てられているか調べ計算を行います.
2) 2 のとき
ISW=1 で呼び出したとき, PANELFACTORL,
PANELFACTORU, NPANELINDEXL または NPANELINDEXU の大きさが不足して ICON=31000 で終了した処理を継続します. このとき, NSIZEFACTORL，NSIZEFACTORU, NSIZEINDEX に返却された必要な大きさで PANELFACTORL， PANELFACTORU, NPANELINDEXL, NPANELINDEXU を確保し直して指定しなおして再度呼び出します.
これらの引数と実行を制御する引数 ISW 以外の引数に格納されている値は変更してはいけません.

3) 3 のとき
先立つ呼び出しで LU 分解された同じ係数行列に対する連立 1 次方程式の右辺ベクトル ***b*** を変えて再度解くことを指定します. このとき先立つ呼び出しで LU 分解した結果を使います.
実行を制御する引数 ISW と右辺ベクトル ***b*** を格納する引数 B 以外の引数に格納されている値を変更していけません.

B ..... 入力. ***Ax=b*** の右辺定数ベクトル.
出力. 解ベクトル.
B(N)なる 1 次元配列.

NASSIGN..... 出力. 各 supernode に対する ***L***, ***U*** は圧縮して 2 次元の panel に格納します. この panel を PANELFACTORL および PANELFACTORU の 1 次元部分配列として順番に割り付けたとき何番目になるかを示します. これらの指標ベクトルも同じように NPANELINDEXL, NPANELINDEXU に割り付けます. j=NASSIGN(i)のとき, i 番目の supernode を j 番目に割り付けたことを示します.
入力. ISW≠1 のとき, 初回呼び出しの値を再利用します.
分解結果の格納方法については図 DM_VSSSS-1 を参照してください.
NASSIGN(N)なる 1 次元配列.

NSUPNUM ..... 出力. supernode の総数.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します. (≤n)

NFCNZFACTORL 出力. 構造的に対称な実スパース行列の LU 分解の結果の行列 ***L*** および ***U*** はスーパーノードに対応しておのおの求めます. 各 supernode に対応する *L* の列ベクトルは, 共通の行指標ベクトルを持つように圧縮してブロック対角部分に ***U*** の対応部分を含めて 2 次元の panel に格納します. この panel を PANELFACTORL の 1 次元部分配列として順番に割り付けたとき, i 番目の panel の先頭要素 panel(1,1) が 1 次元配列の PANELFACTORL の何番目の要素になるかを示します.
NFCNZFACTORL(N+1)なる 8 バイト整数型の 1 次元配列.
結果の格納方法については図 DM_VSSSS-1 を参照してください.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.

PANELFACTORL 出力. 各 supernode の分解結果に対応する行列 ***L*** の複数の列ベクトルと行列 ***U*** のブロック対角部分について，共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します. panel を順番に割り付けます. i 番目の supernode に対応する panel が何番目の位置に割り当てられるかは j=NASSIGN(i)から分かります. その先頭位置は NFCNZFACTORL(j)に格納されています. panel ごとに分解結果が格納されます.
i 番目に割付けられた panel の大きさは NDIM(1,i)×NDIM(2,i)の 2 次元配列と見なせます. i 番目の panel の panel(s,t), $s \geq t$, s=1,…,NDIM(1,i), t=1,…,NDIM(2,i)に下三角行列 ***L*** が格納されます. panel の panel(s,t), $s < t$, t=1,…,NDIM(2,i)には単位上三角行列 ***U*** の対角要素を除いたブロック対角部分が格納されます.
PANELFACTORL(NSIZEFACTORL)なる 1 次元配列.

結果の格納方法については図 DM_VSSSS-1 を参照してください.
((3)使用上の注意 a. 注意③参照).

NSIZEFACTORL . 入力. PANELFACTORL の大きさを示す. 8 バイト整数型.
出力. PANELFACTORL の大きさとして必要な大きさが返却されます.
((3)使用上の注意 a. 注意③参照).

NFCNZINDEXL ... 出力. 各 supernode の分解結果に対応する行列 ***L*** の複数の列ベクトルと行列 ***U*** のブロック対角部分について，共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します.この行指標ベクトルを NPANELINDEXL に順番に割り付けたとき i 番目の行指標ベクトルの先頭要素が 1 次元配列である NPANELINDEXL の何番目の要素になるかを示します.
NFCNZINDEXL(N+1)なる 8 バイト整数型の 1 次元配列.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
結果の格納方法については図 DM_VSSSS-1 を参照してください.

NPANELINDEXL 出力. 各 supernode の分解結果に対応する行列 ***L*** の複数の列ベクトルと行列 ***U*** のブロック対角部分について，共通の行指標ベクトルを持つように圧縮して 2 次元の panel に格納します. この行指標ベクトルを NPANELINDEXL に順番に割り付けます. i 番目の supernode に対応する panel の行の指標ベクトルが何番目に割り当てられるかは j=NASSIGN(i)から分かります. その先頭位置は NFCNZINDEXL(j)に格納されています.
この行指標は post order で並び換えたときの行の番号です.
NPANELINDEXL(NSIZEINDEX)なる 1 次元配列.
結果の格納方法については図 DM_VSSSS-1 を参照してください.
((3)使用上の注意 a. 注意③参照).

NSIZEINDEX ..... 入力. NPANELINDEXL および NPANELINDEXU の大きさを示す.
8 バイト整数型.
出力. 必要な大きさが返却されます.
((3)使用上の注意 a. 注意③参照).

NDIM ..... 出力. NDIM(1,i)および NDIM(2,i)は行列 ***L*** に関して i 番目に割り付けられた panel の 1 次元目と 2 次元目の大きさを示します.
NDIM(1,i)-NDIM(2,i)および NDIM(2,i)は行列 ***U*** に関して割り付けられた panel を転置したときの 1 次元目と 2 次元目の大きさを示します.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
NDIM(2,N)なる 2 次元配列.
結果の格納方法については図 DM_VSSSS-1 を参照してください.

NFCNZFACTORU 出力. 構造的に対称な実スパース行列の LU 分解の結果の行列 ***U*** に関しては, 各 supernode に対応する ***U*** の行ベクトルは, ブロック対角部分を除いて共通の列指標ベクトルを持つように圧縮し, 転置したものを 2 次元の panel に格納します. この panel を PANELFACTORU の 1 次元部分配列として順番に割り付けたとき, i 番目の panel の先頭要素 panel(1,1)が 1 次元配列の PANELFACTORU の何番目の要素になるかを示します.

NFCNZFACTORU(N+1)なる 8 バイト整数型の 1 次元配列.
結果の格納方法については図 DM_VSSSS-1 を参照してください.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.

PANELFACTORU 出力. 各 supernode の分解結果に対応する行列 $\boldsymbol{U}$ のブロック対角部分を除いた複数の行ベクトルについて，共通の列指標ベクトルを持つように圧縮し, 転置して 2 次元の panel に格納します. panel を順番に割り付けます. i 番目の supernode に対応する panel が何番目の位置に割り当てられるかは j=NASSIGN(i) から分かります. その先頭位置は NFCNZFACTORU(j)に格納されています. panel ごとに分解結果が格納されます.
i 番目に割付けられた panel の大きさは {NDIM(1,i)-NDIM(2,i)}×NDIM(2,i)の 2 次元配列と見なせます. i 番目の panel の panel(s,t), s=1,…,NDIM(1,i)-NDIM(2,i), t=1,…,NDIM(2,i)に単位上三角行列 $\boldsymbol{U}$ のブロック対角部分を除いた部分が行ベクトルを圧縮し, 転置して格納します.
PANELFACTORU(NSIZEFACTORU)なる 1 次元配列.
結果の格納方法については図 DM_VSSSS-1 を参照してください.
((3)使用上の注意 a. 注意③参照).

NSIZEFACTORU. 入力. PANELFACTORU の大きさを示す. 8 バイト整数型.
出力. PANELFACTORU の大きさとして必要な大きさが返却されます.
((3)使用上の注意 a. 注意③参照).

NFCNZINDEXU... 出力. 各 supernode の分解結果に対応する行列 $\boldsymbol{U}$ は対角ブロック部分を除いて複数の行ベクトルについて, 共通の列指標ベクトルを持つように圧縮し, 転置して 2 次元の panel に格納します. 対角ブロック部分も含んだこの列指標ベクトルを NPANELINDEXU に順番に割り付けたとき i 番目の列指標ベクトルの先頭要素が 1 次元配列である NPANELINDEXU の何番目の要素になるかを示します.
NFCNZINDEXU(N+1)なる 8 バイト整数型の 1 次元配列.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
結果の格納方法については図 DM_VSSSS-1 を参照してください.

NPANELINDEXU 出力. 各 supernode の分解結果に対応する行列 $\boldsymbol{U}$ の対角ブロックを除いた部分を転置して圧縮して 2 次元の panel に格納します. この列指標ベクトルを対角ブロック部分を含めて NPANELINDEXU に順番に割り付けます. i 番目の supernode に対応する panel の行の指標ベクトルが何番目に割り当てられるかは j=NASSIGN(i)から分かります. その先頭位置は NFCNZINDEXU(j)に格納されています.
この行指標は post order で並び換えたときの列の番号です.
NPANELINDEXU(NSIZEINDEX)なる 1 次元配列.
結果の格納方法については図 DM_VSSSS-1 を参照してください.
((3)使用上の注意 a. 注意③参照).

NPOSTO ..... 出力. post order で i 番目の node が行列 $\boldsymbol{A}$ の何番目の列番号に対応するかを表す 1 次元ベクトル.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
NPOSTO(N)なる 1 次元配列.
((3)使用上の注意 a. 注意④参照).

SCLROW ..... 出力. スケーリング対角行列 $\boldsymbol{D}_r$. 対角要素が 1 次元配列に格納されます.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
SCLROW(N)なる 1 次元配列.

SCLCOL..... 出力. スケーリング対角行列 $\boldsymbol{D}_c$. 対角要素が 1 次元配列に格納されます.
入力. ISW≠1 のとき,初回呼び出しの値を再利用します.
SCLCOL(N)なる 1 次元配列.

EPSZ ..... 入力. 分解過程で対角要素の大きさの絶対値の相対判定値
出力. EPSZ ≤ 0.0 のときは標準値が設定されます.
((3)使用上の注意 a. 注意②参照).

THEPSZ ..... 入力. ピボットの判定での閾値(threshold). これより大きな値はピボットとして採用します. 見つかればその時点で, ピボット検索は打ち切ります.
例えば 1.0D-2 程度.
出力. THEPSZ ≤ 0.0 のときは 1.0D-2 が設定されます.
EPSZ ≥ THEPSZ > 0.0 のときは, EPSZ が設定されます.

IPIVOT..... 入力. スーパーノード内でピボットを行うか, 行う場合どのようなピボットを行うかを指定します. 例えば, 完全ピボットとして 40 を指定.
IPIVOT < 10 : または IPIVOT ≥ 50: ピボットなし.
10 ≤ IPIVOT < 20 : 部分ピボット
20 ≤ IPIVOT < 30 : 対角ピボット
21 : スーパーノード内で対角ピボットがとれないとき, Rook ピボットに変更します.
22 : スーパーノード内で対角ピボットがとれないときは, Rook ピボットに, Rook ピボットが取れないときは完全ピボットに変更します.
30 ≤ IPIVOT < 40 :Rook ピボット
32 : スーパーノード内で Rook ピボットがとれないとき, 完全ピボットをとります.
40 ≤ IPIVOT < 50 : 完全ピボット

ISTATIC..... 入力. Pivot を指定したとき, Static Pivot を行うかを示します.
1) = 1 のとき
スーパーノード内で指定したピボットが SPEPSZ より大きくないとき, ピボットとして DSIGN(SPEPSZ,PIVOT)で近似します. ピボットの値が 0.0d0 なら SPEPSZ に近似します.
このとき, 以下を設定しなければなりません.
a) EPSZ は EPSZ の標準値以下にしなければなりません.
b) ISCLITERMAX は 10 を設定しスケーリングを行う必要があります.
c)THEPSZ ≥ SPEPSZ でなければなりません.

d) 解の反復改良を行うため IREFINE=1 を指定しなければなりません.
2) ≠1 のとき
Static Pivot は行いません.

SPEPSZ ..... 入力. ISTATIC=1 のとき, Statc Pivot として使う値.
1.0D-10 ≥ SPEPSZ ≥ EPSZ でなければなりません.
出力. SPEPSZ<EPSZ のときは, 1.0D-10 が設定されます.

IREFINE..... 入力. LU 分解結果を利用して解を求めるときに, 解の反復改良を行うかどうかを示す. 残差ベクトルを計算するときに 4 倍精度で計算します.
=1 : 解の反復改良を行う. 反復改良して得られる残差ベクトル $\boldsymbol{r_k}$ の絶対値の差分がひとつ前の差分の 1/4 より大きくなるまで, 反復改良を行います.
≠1: 解の反復改良を行わない.
ISTATIC=1 のとき, IREFINE=1 でなければなりません.

EPSR ..... 入力. 解の残差ベクトル $\boldsymbol{b-Ax}$ の絶対値が, $\boldsymbol{b}$ の絶対値に対して十分小さいかを判定する判定値.
EPSR ≤ 0.0 のとき 1.0D-6 が設定されます.

ITERMAX..... 入力. (≥ 1) 反復改良を行うときの最大反復回数.

ITER..... 出力. 反復改良を行った回数.

W..... 作業域.
出力/入力.
大きさ NZ+N の 1 次元配列.
ISW=1, 2 で続けて呼び出すとき,呼び出しの間で必要なデータを受け渡すために使われます. 呼び出しの間で値を変更してはいけません.

IW..... 作業域.
出力/入力.
大きさ 36*N+36+2*NZ+3*(N+1)の 1 次元配列.
ISW=1, 2, 3 で続けて呼び出すとき,呼び出しの間で必要なデータを受け渡すために使われます. 呼び出しの間で値を変更してはいけません.

ICON..... 出力. コンディションコード.
表 DM_VSSSS-1 参照

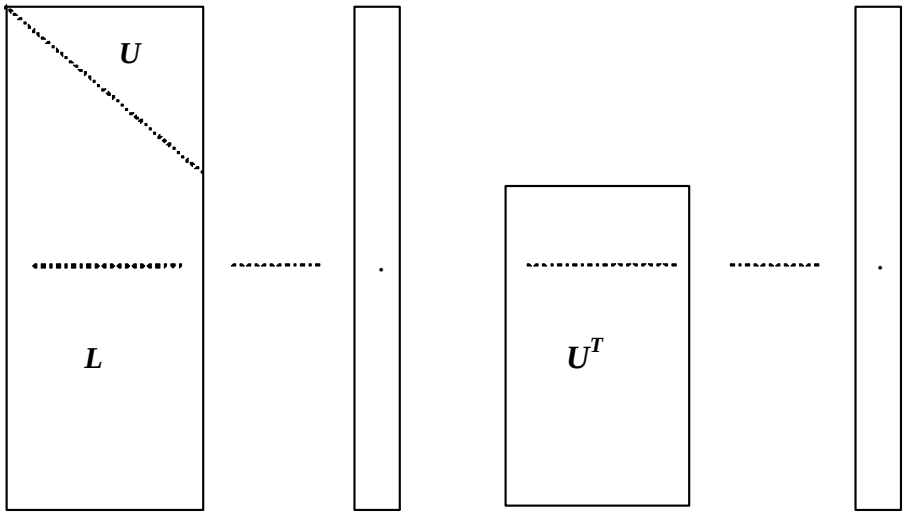


図 DM_VSSSS-1 分解結果の格納概念図

j = NASSIGN(i) → i 番目の supernode は j 番目に格納されます.

p = NFCNZFACTORL(j) → j 番目の panel は PANELFACTORL の p 番目の要素から NDIM(1,j)×NDIM(2,j)の長さを占めます.

q = NFCNZINDEXL(j) → j 番目の panel の行指標を表すベクトルは NPANELINDEXL の q 番目の要素から NDIM(1,j)の長さを占めます.

panel は大きさ NDIM(1,j)×NDIM(2,j)の配列と見なせます.

panel(s,t) , $s \geq t$, s = 1 , … , NDIM(1,j), t=1 , … , NDIM(2,j) に分解結果の下三角行列 $\boldsymbol{L}$ が格納されます.

panel(s,t) , $s < t$, s=1 , … , NDIM(2,j), t=1 , … , NDIM(2,j) に単位上三角行列 $\boldsymbol{U}$ の対角ブロック部分が対角要素を除き格納されます.

u = NFCNZFACTORU(j) → j 番目の panel は PANELFACTORU の u 番目の要素から (NDIM(1,j)-NDIM(2,j))×NDIM(2,j)の長さを占めます.

v = NFCNZINDEXU(j) → j 番目の panel の列指標を表すベクトルは NPANELINDEXU の v 番目の要素から NDIM(1,j)の長さを占めます.

panel は大きさ(NDIM(1,j)-NDIM(2,j))×NDIM(2,j)の配列と見なせます.

panel(x,y) , x=1 , … , NDIM(1,j)-NDIM(2,j), y=1 , … , NDIM(2,j)

に分解結果の単位上三角行列 $\boldsymbol{U}$ の転置行列 $\boldsymbol{U^T}$ の対角ブロック部分を除いた部分が格納されます.

指標の値は行列 $\boldsymbol{A}$ の列番号をもつ node を post order で並び換えた $\boldsymbol{QAQ^T}$ で

の列番号を表します.

表 DM_VSSSS-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし	ー
20000	ピボットが相対的に零になりました.	処理を打ち切ります.
20200	行および列の均衡化を行う対角行列の対角要素を求める過程で,元の行列 A の行もしくは列にゼロベクトルがありました. 行列が特異である可能性があります.	
20400	LU 分解された行列の対角要素にゼロがあります.	
20500	求めた解ベクトルに対する残差ベクトルのノルムの大きさが方程式 Ax=b の右辺ベクトルのノルムの EPSR 倍より大きい.　係数行列は特異に近い可能性があります.	
30000	N<1,NZ<0,NFCNZ(N+1)≠NZ+1, NSIZEFACTORL<1, NSIZEFACTORU<1, NSIZEINDEX<1, ISW<1, ISW>3, IREFINE=1 のとき ITERMAX<1,	
30100	NPERM に指定した置換行列が正しくない.	
30200	NROW(j)に格納された i 列目の行指標 k が k<1 または k>N	
30300	i 列目の行指標の数 NFCNZ(i+1)-NFCNZ(i)>n	
30500	ISTATIC=1 のとき満たすべき条件をみたしていない. EPSZ は標準値 16***u*** より大きかった. または ISCLITERMAX <10 であった. または IREFINE≠1 であった. または SPEPSZ>THEPSZ であった. または SPEPSZ>1.0D-10 であった.	
30700	行列 ***A*** が構造的に対称でない.	
31000	PANELFACTORL の大きさ NSIZEFACTORL または NPANELINDEXL および NPANELINDEXU の大きさ NSIZEINDEX または PANELFACTORU の大きさ NSIZEFACTORU が小さすぎます.	NSIZEFACTORL または NSIZEINDEX または NSIZEFACTORU で指定された大きさで PANELFACTORL または NPANELINDEXL および NPANELINDEXU または PANELFACTORU を割り付けて ISW=2 として再度呼び出す.

(3)　使用上の注意

a.　注意

① 直交行列である置換行列$\boldsymbol{P}$の要素$p_{ij}=1$のとき, NPERM(i)=jと表現します.
逆は以下のようにして求めることができます.

```
DO i=1,n
  j=NPERM(i)
  NPERMINV(j)=i
ENDDO
```

Fill-Reducing OrderingはMETISなどを使って求めることができます.
詳細は“付録1 参考文献一覧表”の[43],[44]を参照願います.

② 分解過程で対角要素の大きさの絶対値の相対零判定値にある値を設定したとすると, この値は次の意味を持っています. すなわち, LU分解の過程で, 対角要素の大きさの絶対値がその値より小さくなった場合に, その値を相対的に零と見なし, ICON=20000として処理を打ち切ります. EPSZの標準値は, 丸め誤差の単位をuとしたとき, EPSZ=16uです.
なお, 対角要素の大きさの絶対値が小さくなっても, 処理を続行させたい場合には, EPSZに極小の値を与えれば良いのですが, 結果の精度は保証されません.
Static Pivotを指定した場合, 対角要素がSPEPSZより小さいとき, SPEPSZで近似してLU分解を行います. このとき, 解の反復改良を指定しなければなりません.

③ 分解結果を格納する配列PANELFACTORL, NPANELINDEXL, PANELFACTORU, NPANELINDEXUの必要な大きさは,事前には分かりません. 十分大きな配列を割り当てるか,本ルーチンを呼び出してsymbolic decompositionを行った解析の結果を使って割り当てを行うことができます.
例えば,大きさ1の1次元配列などを割付けます.そしてその大きさ1などの小さな値をNSIZEFACTORL, NSIZEINDEX, NSIZEFACTORUに指定して, ISW=1で呼び出します.
symbolic decompositionを行い, ICON=31000で終了し, NSIZEFACTORL, NSIZEINDEX, NSIZEFACTORUに必要な大きさが返却されます. 必要な大きさの配列を割付け直して引数に指定して, ISW=2で呼び出すことで, symbolic decomposition以降の処理を続けることができます. 使用例を参照願います.

④ 列番号に対応するノードを考えます.これをpost orderで順番を並び換えたときの番号の対応関係がNPOSTOに格納されています. post orderのi番目のノードが並び換える前の何番目のノードに対応するかを表します. j=nposto(i)はj番目であることを表します.
①同様にこれは直交行列である置換行列$\boldsymbol{Q}$を表し,行列$\boldsymbol{A}$を$\boldsymbol{QAQ^T}$と並び換えることに相当します.
逆変換$\boldsymbol{Q^T}$は以下のようにして求めることができます.

```
DO i=1,n
  j=NPOSTO(i)
  NPOSTOINV(j)=i
  ENDDO
```

⑤ 連立1次方程式$\boldsymbol{Ax}=\boldsymbol{b}$は, DM_VSSSLUで構造的に対称な実スパース行列$\boldsymbol{A}$をLU分解して,分解結果を利用して引き続いてDM_VSSSLUXを呼び出して解くこともできます.

b. 使用例

連立 1 次方程式 $\boldsymbol{Ax} = \boldsymbol{f}$ を解きます.

行列は境界条件として立方体の境界で 0 となる楕円型の偏微分方程式に有限差分法を適用して生成されたものを利用します.

$$-\Delta u + a\nabla u + cu = f$$

ここで $a = (a_1, a_2, a_3)$.

行列はサブルーチン init_mat_diag によって生成されます.

対角形式格納法で生成し,圧縮列格納法で格納します. 結果生成された構造的に対称な行列 $\boldsymbol{A}$ に関する連立 1 次方程式を解きます.(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT  REAL*8 (A-H,O-Z)
      PARAMETER (NORD=39,NX = NORD,NY =NORD ,NZ = NORD,
     $      N = NX*NY*NZ,NXY=NX*NY)
      PARAMETER (K = N+1)
      PARAMETER (NDIAG = 7)
      PARAMETER (NALL=NDIAG*N,
     $   IWL=36*N+36+2*NALL+3*(N+1))
      PARAMETER(IPRINT=0)
      DIMENSION NOFST(NDIAG)
      DIMENSION DIAG(K,NDIAG),DIAG2(K,NDIAG)
      DIMENSION C(K*NDIAG),NROWC(K*NDIAG),NFCNZC(N+1),
     $          WC(K*NDIAG),IWC(2,K*NDIAG)
      DIMENSION A(NDIAG*N),NCOLUMN(K*NDIAG),NFCNZ(N+1),
     $          NPERM(N),W(NDIAG*N+N),
     $          NPOSTO(N),NDIM(2,N),
     $          NASSIGN(N),
     $          IW(IWL)
      REAL*8, DIMENSION(:), ALLOCATABLE ::  PANELFACTORL,PANELFACTORU
      INTEGER*4, DIMENSION(:), ALLOCATABLE :: NPANELINDEXL,
     $                                        NPANELINDEXU
      REAL*8 DUMMYFL,DUMMYFU
      INTEGER*4 NDUMMYIL,NDUMMYIU
      INTEGER*8 NSIZEFACTORL,NSIZEINDEX,
     $          NSIZEFACTORU,
     $          NFCNZFACTORL(N+1),
     $          NFCNZFACTORU(N+1),
     $          NFCNZINDEXL(N+1),
```

```
     $          NFCNZINDEXU(N+1)
      DIMENSION X(N),B(N),SOLEX(N),NPERM1(N)
C
      REAL*8 THEPSZ,
     $       EPSR,
     $       SEPSZ,
     $       SCLROW(N),SCLCOL(N)

      INTEGER*4     IPIVOT,ISTATIC,
     $              ISCLITERMAX,
     $              IREFINE,ITERMAX,ITER



      PRINT *,'    DIRECT METHOD'
      PRINT *,'    FOR SPARSE STRUCTURALLY SYMMETRIC REAL MATRICES'
      PRINT *,'    IN COMPRESSED COLUMN STORAGE'
      PRINT *

      DO I=1,N
      SOLEX(I)=1.0D0
      ENDDO
      PRINT *,'    EXPECTED SOLUTIONS'
      PRINT *,'    X(1) = ',SOLEX(1),' X(N) = ',SOLEX(N)
      PRINT *

      VA1 = 1.0D0
      VA2 = 2.0D0
      VA3 = 3.0D0
      VC =  4.0D0
      XL = 1.0
      YL = 1.0
      ZL = 1.0
      CALL INIT_MAT_DIAG(VA1,VA2,VA3,VC,DIAG,NOFST
     &          ,NX,NY,NZ,XL,YL,ZL,NDIAG,N,K)
C
      DIAG2=0
C
      DO I=1,NDIAG
C
      IF(NOFST(I).LT.0)THEN
      NBASE=-NOFST(I)
```

```
      LENGTH=N-NBASE
      DIAG2(1:LENGTH,I)=DIAG(NBASE+1:N,I)
      ELSE
      NBASE=NOFST(I)
      LENGTH=N-NBASE
      DIAG2(NBASE+1:N,I)=DIAG(1:LENGTH,I)
      ENDIF
C
      ENDDO
C
      NUMNZC=1
C
      DO J=1,N
      NTOPCFGC=1
C
      DO I=NDIAG,1,-1
C
      IF(DIAG2(J,I).NE.0.0D0)THEN
C
      NCOL=J-NOFST(I)
      C(NUMNZC)=DIAG2(J,I)
      NROWC(NUMNZC)=NCOL
C
      IF(NTOPCFGC.EQ.1)THEN
      NFCNZC(J)=NUMNZC
      NTOPCFGC=0
      ENDIF
C
      NUMNZC=NUMNZC+1
C
      ENDIF
      ENDDO
      ENDDO
C
      NFCNZC(N+1)=NUMNZC
      NNZC=NUMNZC-1
C
      CALL DM_VMVSCC(C,NNZC,NROWC,NFCNZC,N,SOLEX,
     $             B,WC,IWC,ICON)
C
C
      X=B
```

```
      IORDERING=0
      ISCLITERMAX=10
      ISW=1
      EPSZ=1.0D-16
      NSIZEFACTORL=1
      NSIZEFACTORU=1
      NSIZEINDEX=1
      THEPSZ=1.0D-2
      EPSR=1.0D-8
      SEPSZ=1.0D-10
      IPIVOT=40
      ISTATIC=1
      IREFINE=1
      ITERMAX=10
C
      CALL DM_VSSSS(C,NNZC,NROWC,NFCNZC,N,
     $              ISCLITERMAX,IORDERING,
     $              NPERM,ISW,
     $              X,
     $              NASSIGN,
     $              NSUPNUM,
     $              NFCNZFACTORL,DUMMYFL,
     $              NSIZEFACTORL,NFCNZINDEXL,
     $              NDUMMYIL,NSIZEINDEX,NDIM,
     $              NFCNZFACTORU,DUMMYFU,
     $              NSIZEFACTORU,
     $              NFCNZINDEXU,NDUMMYIU,
     $              NPOSTO,
     $              SCLROW,SCLCOL,
     $              EPSZ,
     $              THEPSZ,
     $              IPIVOT,ISTATIC,SEPSZ,
     $              IREFINE,EPSR,ITERMAX,ITER,
     $              W,IW,ICON)
C
      PRINT*,'    ICON=',ICON,' NSIZEFACTORL=',NSIZEFACTORL,
     $       'NSIZEFACTORU=',NSIZEFACTORU,
     $       'NSIZEINDEX=',NSIZEINDEX
      PRINT*,'    NSUPNUM=',NSUPNUM
      PRINT *
C
      ALLOCATE( PANELFACTORL(NSIZEFACTORL) )
```

```
      ALLOCATE( PANELFACTORU(NSIZEFACTORU) )
      ALLOCATE( NPANELINDEXL(NSIZEINDEX) )
      ALLOCATE( NPANELINDEXU(NSIZEINDEX) )
C
      ISW=2
      CALL DM_VSSSS(C,NNZC,NROWC,NFCNZC,N,
     $              ISCLITERMAX,IORDERING,
     $              NPERM,ISW,
     $              X,
     $              NASSIGN,
     $              NSUPNUM,
     $              NFCNZFACTORL,PANELFACTORL,
     $              NSIZEFACTORL,NFCNZINDEXL,
     $              NPANELINDEXL,NSIZEINDEX,NDIM,
     $              NFCNZFACTORU,PANELFACTORU,
     $              NSIZEFACTORU,
     $              NFCNZINDEXU,NPANELINDEXU,
     $              NPOSTO,
     $              SCLROW,SCLCOL,
     $              EPSZ,
     $              THEPSZ,
     $              IPIVOT,ISTATIC,SEPSZ,
     $              IREFINE,EPSR,ITERMAX,ITER,
     $              W,IW,ICON)

C
      ERR = ERRNRM(SOLEX,X,N)

      PRINT *,'    COMPUTED VALUES'
      PRINT *,'    X(1) = ',X(1),' X(N) = ',X(N)
      PRINT *
      PRINT *,'    ICON = ',ICON
      PRINT *
      PRINT *,'    N = ',N,' :: NX = ',NX,' NY = ',NY,' NZ = ',NZ
      PRINT *
      PRINT *,'    ERROR = ',ERR
      PRINT *,'    ITER=',ITER
      PRINT *
      PRINT *

      IF(ERR.LT.1.0D-8.AND.ICON.EQ.0)THEN
         WRITE(*,*)'    ********** OK **********'
```

```
      ELSE
         WRITE(*,*)'    ********** NG **********'
      ENDIF

      DEALLOCATE( PANELFACTORL,PANELFACTORU,NPANELINDEXL,
     $            NPANELINDEXU )

      STOP
      END

C =======================================
C     INITIALIZE COEFFICIENT MATRIX
C =======================================
      SUBROUTINE INIT_MAT_DIAG(VA1,VA2,VA3,VC,D_L,OFFSET
     &               ,NX,NY,NZ,XL,YL,ZL,NDIAG,LEN,NDIVP)
      IMPLICIT  REAL*8(A-H,O-Z)
      DIMENSION D_L(NDIVP,NDIAG)
      INTEGER   OFFSET(NDIAG)
C
      IF (NDIAG .LT. 1) THEN
        WRITE (*,*) 'SUBROUTINE INIT_MAT_DIAG:'
        WRITE (*,*) ' NDIAG SHOULD BE GREATER THAN OR EQUAL TO 1'
        RETURN
      ENDIF

!$OMP PARALLEL DEFAULT(PRIVATE)
!$OMP+     SHARED(VA1,VA2,VA3,VC,D_L,OFFSET
!$OMP+      ,NX,NY,NZ,XL,YL,ZL,NDIAG,LEN,NDIVP)

C NDIAG CANNOT BE GREATER THAN 7
      NDIAG_LOC = NDIAG
      IF (NDIAG .GT. 7) NDIAG_LOC = 7

C INITIAL SETTING
      HX = XL/(NX+1)
      HY = YL/(NY+1)
      HZ = ZL/(NZ+1)

!$OMP DO
      DO I = 1,NDIVP
      DO J = 1,NDIAG
      D_L(I,J) = 0.0
```

```
      ENDDO
      ENDDO
!$OMP ENDDO

      NXY = NX*NY

C OFFSET SETTING
!$OMP SINGLE
      L = 1
      IF (NDIAG_LOC .GE. 7) THEN
        OFFSET(L) = -NXY
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 5) THEN
        OFFSET(L) = -NX
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 3) THEN
        OFFSET(L) = -1
        L = L+1
      ENDIF
      OFFSET(L) = 0
      L = L+1
      IF (NDIAG_LOC .GE. 2) THEN
        OFFSET(L) = 1
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 4) THEN
        OFFSET(L) = NX
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 6) THEN
        OFFSET(L) = NXY
      ENDIF
!$OMP END SINGLE

C MAIN LOOP
!$OMP DO
      DO 100 J = 1,LEN
        JS = J

C DECOMPOSE JS-1 = (K0-1)*NX*NY+(J0-1)*NX+I0-1
```

```
      K0 = (JS-1)/NXY+1
      IF (K0 .GT. NZ) THEN
 PRINT*,'ERROR; K0.GH.NZ '
 GOTO 100
 ENDIF
      J0 = (JS-1-NXY*(K0-1))/NX+1
      I0 = JS - NXY*(K0-1) - NX*(J0-1)
      L = 1

      IF (NDIAG_LOC .GE. 7) THEN
        IF (K0 .GT. 1) D_L(J,L) = -(1.0/HZ+0.5*VA3)/HZ
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 5) THEN
        IF (J0 .GT. 1) D_L(J,L) = -(1.0/HY+0.5*VA2)/HY
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 3) THEN
        IF (I0 .GT. 1) D_L(J,L) = -(1.0/HX+0.5*VA1)/HX
        L = L+1
      ENDIF
      D_L(J,L) = 2.0/HX**2+VC
      IF (NDIAG_LOC .GE. 5) THEN
        D_L(J,L) = D_L(J,L) + 2.0/HY**2
        IF (NDIAG_LOC .GE. 7) THEN
          D_L(J,L) = D_L(J,L) + 2.0/HZ**2
        ENDIF
      ENDIF
      L = L+1
      IF (NDIAG_LOC .GE. 2) THEN
        IF (I0 .LT. NX) D_L(J,L) = -(1.0/HX-0.5*VA1)/HX
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 4) THEN
        IF (J0 .LT. NY) D_L(J,L) = -(1.0/HY-0.5*VA2)/HY
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 6) THEN
        IF (K0 .LT. NZ) D_L(J,L) = -(1.0/HZ-0.5*VA3)/HZ
      ENDIF
 100  CONTINUE
!$OMP ENDDO
```

```
!$OMP END PARALLEL

      RETURN
      END

C ======================================
* SOLUTE ERROR
* | X1 - X2 |
C ======================================
      REAL*8 FUNCTION ERRNRM(X1,X2,LEN)
      IMPLICIT  REAL*8 (A-H,O-Z)
      DIMENSION X1(*),X2(*)
C
      S = 0D0
      DO 100 I = 1,LEN
        SS = X1(I) - X2(I)
        S = S + SS * SS
 100  CONTINUE
C
      ERRNRM = SQRT( S )
      RETURN
      END
```

(4)　手法概要

行列に対して行, 列の均衡化を行うスケーリングを施します.
この行列をLU分解します.
スーパーノードに対応する非ゼロ要素を2次元のpanelに格納します.
安定化のためのピボットは対角ブロック部分で検索します.
指定した値より絶対値が大きな値が見つかるとピボットとして採用する閾値THEPSZを指定できます. 求めたピボットが小さすぎるとき, SPEPSZで指定した値で近似してLU分解を続けるstatic pivotを指定することも可能です.
詳細は“付録1 参考文献一覧表”の以下の文献を参照願います.
スパースな実行列のLU分解に関しては[19],[2],[22],[48],[68]を行列の均衡化やピボットに関しては[63],[69]をそれぞれ参照願います.

DM_VTDEVC

実三重対角行列の固有値・固有ベクトル
CALL DM_VTDEVC(D, SL, SU, N, NF, NL, IVEC, ETOL, CTOL, NEV, E, MAXNE, EV, K, M, ICON)

(1) 機能

実三重対角行列の指定された固有値を求めます. 指定に応じて対応する固有ベクトルを求めます.

$$\boldsymbol{T}\boldsymbol{x} = \lambda \boldsymbol{x} \tag{1.1}$$

$\boldsymbol{T}$はn次元の実三重対角行列です. なお, 三重対角行列$\boldsymbol{T}$は次の条件を満たすものとします.

$$l_i u_{i-1} > 0, \quad i = 2, \ldots, n \tag{1.2}$$

ここで, 三重対角行列 $\boldsymbol{T}$ の要素 t_{ij} としたとき, d_i は対角要素を,
$l_i = t_{i,\,i-1}$, $u_i = t_{i,\,i+1}$, は副対角要素を表します. ただし $l_1 = u_n = 0$.

$$(\boldsymbol{T}\boldsymbol{v})_i = l_i v_{i-1} + d_i v_i + u_i v_{i+1}, \quad i = 1, 2, \ldots, n \tag{1.3}$$

(2) パラメタ

D.....入力. D(N)なる倍精度実数型の 1 次元配列で対角要素 d_i を格納します.

SL.....入力. SL(N)なる倍精度実数型の 1 次元配列で下副対角行列 l_i を SL(2:N)に格納します. SL(1) = 0.

SU .....入力. SU(N)なる倍精度実数型の 1 次元配列で上副対角行列 u_i を SU(1:N-1)に格納します. SU(N) = 0.

N.....入力. 三重対角行列の次数 n.

NF .....入力. 固有値を小さい方から番号付けして(多重固有値には多重度分番号を割り当てる), 求める最初の固有値の番号.

NL .....入力. 固有値を小さい方から番号付けして(多重固有値には多重度分番号を割り当てる), 求める最後の固有値の番号.

IVEC .....入力. 制御情報.

1 のとき　　固有値および固有ベクトルの両方を求めます.

1 以外のとき　固有値のみ求めます.

ETOL .....入力. 固有値が数値的に異なるか多重かを式(3.4)を利用して判定する判定値.

3.0D-16 以下のときこの値が標準値として設定されます.

((3)使用上の注意 a. 注意②参照).

CTOL .....入力. 近接している固有値が近似的に多重かを式(3.4)を利用して判定する判定値.

1.0D-6 ≥ CTOL ≥ ETOL.

CTOL > 1.0D-6 のときは, CTOL = 1.0D-6 と設定されます.

CTOL < ETOL のときは, CTOL=10 × ETOL が標準値として設定されます.
((3)使用上の注意 a. 注意②参照).

NEV .....出力. 求められた固有値の個数に関する情報.
NEV(5)なる 1 次元配列.
詳細は以下のとおり.
NEV(1)は, 求めた異なる固有値の個数.
NEV(2)は, 求めた異なる近似的多重固有値(cluster)の個数.
NEV(3)は, 求めた多重度も含んだすべての固有値の個数.
NEV(4)は, 求めた固有値の内最初の固有値の番号.
NEV(5)は, 求めた固有値の内最後の固有値の番号.

E.....出力. 固有値が格納されます.
求められた固有値は E(1:NEV(3))に格納されます.
E(MAXNE)なる 1 次元配列.

MAXNE.....入力. 計算できる固有値の最大個数.
多重度 m の固有値がいくつかあると考えられるとき, n を上限として NL - NF + 1 + 2 × m を設定する必要があります. 配列 E の大きさ.
NEV(3) > MAXNE のとき, 固有ベクトルの計算はできません.
((3)使用上の注意 a. 注意③参照).

EV .....出力. IVEC = 1 のとき, 固有値に対応して固有ベクトルが格納されます.
求められた固有ベクトルは EV(1:N, 1:NEV(3))に格納されます.
EV(K, MAXNE)なる 2 次元配列.

K.....入力. EV の 1 次元目の大きさ(≥N).

M.....出力. 求められた固有値の多重度に関する情報.
M(i, l)は i 番目の固有値λ_iの多重度を, M(i, 2)は i 番目の固有値λ_iに関して, 近接している固有値を近似的多重固有値(cluster)と見なしたときの多重度を示します.
((3)使用上の注意 a. 注意②参照).
M(MAXNE, 2)なる 2 次元配列.

ICON.....出力. コンディションコード.
表 DM_VTDEVC-1 参照

表 DM_VTDEVC-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし	―
20000	多重固有値の計算中, 固有値の総数が MAXNE を超えた.	処理を打ち切る. 固有ベクトルを求めることはできないが, 異なる固有値自体は求められている. NEV(3)に MAXNE に設定するべき大きさが返却される. ((3)使用上の注意 a. 注意③参照).
30000	N<1, K<1, NF<1, NL>N, NL<NF, MAXNE<NL - NF+1, N>K.	処理を打ち切る
30100	$SL(i)\times SU(i-1)\le 0$, 行列が対称化できなかった.	

(3)　使用上の注意

a.　注意

①　本ルーチンの適用条件について

本ルーチンは $l_i u_{i-1} > 0$ のみを要求します. このため, 例えば(3.1)左側の一般化固有値問題は, 対角行列 $\boldsymbol{D}$ の要素が全て正ならば, 両辺に左から $\boldsymbol{D}^{-1}$ をかけて, 本ルーチンで扱える三重対角行列の固有値問題の形に変換できます.

$$\boldsymbol{T}\boldsymbol{x} = \lambda \boldsymbol{D}\boldsymbol{x} \qquad \longrightarrow \qquad \boldsymbol{T}'\boldsymbol{x} = \lambda \boldsymbol{x},\ \boldsymbol{T}' = \boldsymbol{D}^{-1}\boldsymbol{T} \tag{3.1}$$

また, 非対称三重対角行列の固有値問題 $\boldsymbol{T}\boldsymbol{x} = \lambda \boldsymbol{x}$ は対角行列を使って以下にしめす, 対称な一般化固有値問題に変換することができます.

$$\boldsymbol{D}\boldsymbol{T}\boldsymbol{x} = \lambda \boldsymbol{D}\boldsymbol{x} \tag{3.2}$$

ここで, 対角行列 $\boldsymbol{D}$ の対角要素は $d_1 = 1,\ d_i = u_{i-1} d_{i-1} / l_i,\ i = 2, \ldots, n$. このことは, 対称三重対角行列の一般化固有値問題も本ルーチンで扱う形に変換できることを意味します. ただ, 変換に際し $\boldsymbol{D}^{-1}$ を乗することでスケーリング問題を引き起こす場合は, (3.2)と等価な以下の対称な問題を考えるほうが望ましいです.

$$\boldsymbol{D}^{1/2}\boldsymbol{T}\boldsymbol{D}^{-1/2}\boldsymbol{w} = \lambda \boldsymbol{w}, \quad \boldsymbol{w} = \boldsymbol{D}^{1/2}\boldsymbol{x} \tag{3.3}$$

②　本ルーチンは, 求める固有値を交わりのない順序付けられた集合に分けて独立に計算することにより並列化しています.

ε = ETOL としたとき, 連続する固有値 $\lambda_j,\ j = s-1, s, \ldots, s+k,\ (k \ge 0)$に対して,

$$\frac{\left|\lambda_i - \lambda_{i-1}\right|}{1 + \max\left(\left|\lambda_{i-1}\right|, \left|\lambda_i\right|\right)} \le \varepsilon \tag{3.4}$$

$i = s, s+1, \ldots, s+k$ となる i に対して(3.4)を満たし, $i = s-1$ および $i = s+k+1$ について(3.4)を満たさないとき, これらの固有値 $\lambda_j,\ j = s-1, s, \ldots, s+k$ は数値的に多重であるとみなされます.

ETOL の標準値は 3.0D-16～丸め誤差の単位であり, このとき固有値は計算機で求め得る精度まで分離されます.
(3.4)がε=ETOL に対して成立しないとき, λ_{i-1} およびλ_iは異なる固有値と考えます.
ε=ETOL で異なる固有値とみなされた連続する固有値λ_m, $m = t - 1, t, \ldots, t+k$, ($k \geq 0$)に対して, ε=CTOL としたとき, $i = t, t+1, \ldots, t+k$ について(3.4)を満たし, $i = t - 1$ および $i = t+k+1$ について(3.4)を満たさないとき, これらの異なる固有値λ_m, $m = t - 1, t, \ldots, t+k$ を近似的に多重(cluster)と見なします. これは cluster に対応する不変部分空間, つまり 1 次独立な固有ベクトルを計算するために利用されます.

③ MAXNE に, 計算できる固有値の最大個数を指定できます. CTOL を大きくすると, cluster の大きさが大きくなり, 計算する固有値の総数が MAXNE を超えるかも知れません. この場合, CTOL を小さくするか, MAXNE を大きくする必要があります.
計算する固有値の総数が MAXNE を超えた場合, ICON=20000 を返却します. このとき, 固有ベクトルの計算を行うよう指定されていたら固有ベクトルの計算はできません. 固有値は求められていますが, 多重度分だけ繰り返して格納はされてはいません.
つまり, 固有値に関しては, 求められた異なる固有値が, E(1:NEV(1))に, および対応する固有値の多重度が M(1:NEV(1), 1)にそれぞれ格納されています.
固有値がすべて異なり, 近似的多重な固有値もない場合, MAXNE は求める固有値の総数を NT(NT=NL-NF+1)として NT で十分です. 多重な固有値がいくつかあり, 多重度が m と考えられるときは, 少なくとも MAXNE は $NT + 2 \times m$ とする必要があります.
計算する固有値の総数が MAXNE を超えた場合, 計算を続けるために必要な値が NEV(3)に返却されます. この値で領域を確保して再度呼び出すことで計算を続けることができます.

b. 使用例
数値的多重固有値を持つことが知られている Wilkinson による変形(“付録 1 参考文献一覧表”の[81]参照)に基づくモデル問題の $ne = nf - nl + 1$ 個の固有値と対応する固有ベクトルを計算する問題を解きます.
(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT REAL*8(A-H,O-Z)
C
      INTEGER K,N,N0,N1,NE,MAX_CLUS,MAX_NEV,NWR,P1,Q1,IVEC
C
      PARAMETER (K=7001)
```

```
      PARAMETER (P1=70,Q1=100,N=P1*Q1,N0=6001,N1=7000,
     &           NE=N1-N0+1)
      PARAMETER (MAX_CLUS=2*Q1,MAX_NEV=NE+MAX_CLUS)
      PARAMETER (NWR=2*N+2)
C
C
      REAL*8  EVAL_TOL,CLUS_TOL
      REAL*8  A(N),B(N),C(N),EVAL(MAX_NEV),
     &        EVEC(K,MAX_NEV),WR(NWR)
      INTEGER MULT(MAX_NEV,2),NEV(3),ICON,I,J,II,L
      INTEGER N0X,N1X
C
C     W^+_n (Wilkinson): Pathologically close eigenvalues
C
      J = ( P1 + 1 ) / 2
      B(J) = 0.D0
      DO 40 I=1,J-1
         A(I+1) = 1.D0
         C(I) = 1.D0
         A(J+I) = 1.D0
         C(J+I-1) = 1.D0
         B(I) = DFLOAT( J - I )
   40    B(2*J-I) = B(I)
      A(1) = 0.D0
      C(P1) = 0.D0
      DO 45 L=2,Q1
         II = (L-1) * P1
         DO 45 I=1,P1
            A(II+I) = A(I)
            C(II+I) = C(I)
            B(II+I) = B(I)
   45 CONTINUE
C
      A(1)=0.D0
      C(N)=0.D0
C
      EVAL_TOL=3.D-16
      CLUS_TOL=5.D-12
C
      N0X=N0
      N1X=N1
      IVEC=1
```

```
C
      CALL DM_VTDEVC(B,A,C,N,N0X,N1X,IVEC,EVAL_TOL,CLUS_TOL,NEV,
     &               EVAL,MAX_NEV,EVEC,K,MULT,ICON)
C
      CALL CHECK(A,B,C,N,EVEC,K,EVAL,NEV,WR,WR(N+3))
C
      STOP
      END

      SUBROUTINE CHECK(SL,D,SU,N,EV,LD,E,NEV,W,W2)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION SU(*),D(*),SL(*),EV(LD,*),E(*),NEV(3),
     &          W(N+2),W2(N)
C
      TMP=0.0
      DO I=1,NEV(3)
C
      DO J=1,N
      W(J+1)=EV(J,I)
      ENDDO
      W(1)=0.0
      W(N+2)=0.0
      DO J=1,N
      W2(J)=SL(J)*W(J)+D(J)*W(J+1)+SU(J)*W(J+2)-E(I)*W(J+1)
      TMP=MAX(TMP,ABS(W2(J)/(ABS(E(I))+1)))
      ENDDO
      ENDDO
C
      PRINT*,'== maximum element error in ||T*x-eig*x||= ',TMP,' ==='

      RETURN
      END
```

(4)　手法概要

各プロセッサーで固有値を区間の細分化によって求めるとき, Sturm 列の値は 1 つの固有値に対しておおよそ, *npts* / *nev* 点で計算されます. ただし, $npts \geq \max(msect, 3 \times \mathrm{MAXNE}) + \mathrm{MAXNE}$, *nev* は求める固有値の個数です.

npts の最適な最小値の選択は“付録 1 参考文献一覧表”の[71]に従って決定されます.

混成的なデータ構造が利用されます. 固有値の順序付けおよび多重区間分けの両方の処理を行うために LIFO(*last in*, *first out*)リスト構造と Array 構造が結合されています. これに関しては“付録 1 参考文献一覧表”の[61]を参照してください. この処理は, ETOL で与えられる区間の細分化の限界まで行われます. (多重区間分けは固定の回数行われます.) 標準

値が選定されるとき, 固有値の精度は行列のスケールに相対的な計算機精度となります.
Sturm カウントに関しては“付録 1 参考文献一覧表”の[80]を参照してください. 符号のカウントは IEEE 浮動小数点演算で, 固有値をパラメタに持つ単調関数になる特性があります(“付録 1 参考文献一覧表”の[20]参照).
固有ベクトルは, 逆反復法で計算されます.
数値的に多重(または, 近似的に多重)な固有値が検出されたときを除いて, 初期ベクトルは Sturm 列の符号構造を利用して決定されます.
数値的に多重(または, 近似的に多重)な固有値に対してはランダムな初期ベクトルが追加生成され, クラスタの他の固有ベクトルと直交化されます. たいていの場合逆反復の反復回数は 1 回で十分です. クラスタ固有値に対応する固有ベクトルも逆反復の後に再直交化されます.

DM_VTFQD

非対称または不定値のスパース行列の連立 1 次方程式 (TFQMR 法, 対角形式格納法)
CALL DM_VTFQD (A, K, NDIAG, N, NOFST, B, ITMAX, EPS, IGUSS, X, ITER, ICON)

(1)　機能

$n \times n$ の非対称／不定値なスパース行列を係数行列とする連立 1 次方程式を転置なし準最小残差法(TFQMR : Transpose-free Quasi Minimal Residual method)で解きます.

$$\boldsymbol{Ax} = \boldsymbol{b}$$

$n \times n$ の係数行列 $\boldsymbol{A}$ は, 対角形式の格納法で格納します.
ベクトル $\boldsymbol{b}$ および $\boldsymbol{x}$ は n 次元ベクトルです.
反復解法の収束および利用指針に関しては, “SSL II 拡張機能使用手引書 II 第 I 部 概説 第 4 章 連立 1 次方程式の反復解法と収束性” を参照してください.

(2)　パラメタ

A.....入力. 係数行列の非零要素を格納します.
A(1:N, 1:NDIAG)に係数行列を格納します.
A(K, NDIAG)なる 2 次元配列.
対角形式の格納方法については, “SSL II 拡張機能使用手引書 II 第 I 部 概説 3.2.1.1 一般スパース行列の格納方法 b. 一般スパース行列の対角形式の格納方法” を参照してください.

K.....入力. 配列 A の 1 次元目の大きさ(≥ N).

NDIAG.....入力. 係数行列 $\boldsymbol{A}$ の非零要素を含む対角ベクトル列の総数. 配列 A の 2 次元目の大きさ.

N.....入力. 行列 $\boldsymbol{A}$ の次数 n.

NOFST.....入力. 配列 A に格納される対角ベクトルに対応した主対角ベクトルからの距離を格納します. 上対角ベクトル列は正, 下対角ベクトル列は負の値で表します.
NOFST(NDIAG)なる 1 次元配列.

B.....入力. 連立 1 次方程式の右辺の定数ベクトルを B(1:N)に格納します.
B(N)なる 1 次元配列.

ITMAX .....入力. TFQMR 法の反復回数の上限値. 2000 程度で十分です.

EPS.....入力. 収束判定に用いられる判定値.
EPS が 0.0 以下のとき, 10^{-6} が設定されます.
((3)使用上の注意 a. 注意①参照).

IGUSS.....入力. 配列 X に指定された解ベクトルの近似値から反復計算を行うかを指定する制御情報.
0 のとき, 解ベクトルの近似値を指定しません.
0 以外のとき, 配列 X に指定された解ベクトルの近似値から反復計算を開始します.

X.....入力. X(1:N)に解ベクトルの近似値を指定することができます.
出力. 解ベクトルが格納されます.
X(N)なる 1 次元配列.
ITER.....出力. TFQMR 法での実際の反復回数.
ICON.....出力. コンディションコード.
表 DM_VTFQD-1 参照.

表 DM_VTFQD-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし	ー
20000	Break down を起こした.	処理を打ち切る.
20001	反復回数の上限に達した.	処理を打ち切る. 配列 X には, そのときまでに得られている近似値を出力するが, 精度は保証できない.
30000	N<1, N>K, NDIAG <1, ITMAX ≤0	処理を打ち切る.
32001	| NOFST | > N - 1	

(3)　使用上の注意

a.　注意

①　本ルーチンは, 残差のユークリッドノルムが最初の残差のユークリッドノルムと EPS の積以下になったとき解が収束したと見なします. 正確な解と求められた近似解の誤差はほぼ行列 ***A*** の条件数と EPS の積に等しくなります.

②　対角形式を使う上での注意

係数行列 ***A*** の外側の対角ベクトル(“SSL II 拡張機能使用手引書 II 3.2.1.1 b. 一般スパース行列の対角形式の格納方法” 参照)の要素は零を設定する必要があります.

対角ベクトル列を配列 A に格納する順序に制限は特にありません.

この方法の利点は, 行列ベクトル積が間接指標を使わずに計算できる点です.

しかし, 対角構造を持たない行列は効率よく格納できないという点が欠点です.

b.　使用例

連立 1 次方程式 $\boldsymbol{Ax} = \boldsymbol{f}$ を解きます. 行列 ***A*** は境界条件として立方体の境界で 0 となる楕円型の偏微分方程式に有限差分法を適用して生成されます.

$$-\Delta u + a\nabla u + u = f$$

ここで $a = (a_1, a_2, a_3)$, a_1, a_2 および a_3 はある定数です. 行列 ***A*** はサブルーチン init_mat_diag によって生成されます.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT  REAL*8 (A-H,O-Z)
      PARAMETER (EPS = 1D-8)
      PARAMETER (NORD=60,NX = NORD,NY =NORD ,NZ = NORD,
     $      N = NX*NY*NZ)
      PARAMETER (K = N+1)
      PARAMETER (NDIAG = 7)
      PARAMETER(NVW=3*K)

      DIMENSION NOFST(NDIAG)
      DIMENSION A(K,NDIAG)
      DIMENSION X(N),B(N),SOLEX(N),Y(N)
      DIMENSION VW(NVW)

      PRINT *,'    BICGSTAB(L) METHOD'
      PRINT *,'    DIAGONAL FORMAT'
      PRINT *

      SOLEX(1:N)=1.0D0
      PRINT *,'    EXPECTED SOLUTIONS'
      PRINT *,'    X(1) = ',SOLEX(1),' X(N) = ',SOLEX(N)
      PRINT *

      VA1 = 3D0
      VA2 = 1D0/3D0
      VA3 = 5D0
      VC = 1.0
      XL = 1.0
      YL = 1.0
      ZL = 1.0
      CALL INIT_MAT_DIAG(VA1,VA2,VA3,VC,A,NOFST
     &          ,NX,NY,NZ,XL,YL,ZL,NDIAG,N,K)
      NBANDL=0
      NBANDR=0
      DO I=1,NDIAG
      IF(NOFST(I).LT.0)THEN
      NBANDL=MAX(NBANDL,-NOFST(I))
      ELSE
      NBANDR=MAX(NBANDR,NOFST(I))
      ENDIF
      ENDDO
```

```
      VW(1+NBANDL:N+NBANDL) = SOLEX(1:N)
      CALL DM_VMVSD(A,K,NDIAG,N,NOFST,NBANDL,VW,B,ICON2)

      X(1:N)=0.0D0
      ERR1 = ERRNRM(SOLEX,X,N)
      VW(1+NBANDL:N+NBANDL) = X(1:N)
      CALL DM_VMVSD(A,K,NDIAG,N,NOFST,NBANDL,VW,Y,ICON2)
      ERR2 = ERRNRM(Y,B,N)

      IGUSS = 0
      ITMAX = 2000

      CALL DM_VTFQD(A,K,NDIAG,N,NOFST,B,ITMAX
     &          ,EPS,IGUSS,X,ITER,ICON)

      ERR3 = ERRNRM(SOLEX,X,N)
      VW(1+NBANDL:N+NBANDL) = X(1:N)
      CALL DM_VMVSD(A,K,NDIAG,N,NOFST,NBANDL,VW,Y,ICON2)
      ERR4 = ERRNRM(Y,B,N)

      PRINT *,'    COMPUTED VALUES'
      PRINT *,'    X(1) = ',X(1),' X(N) = ',X(N)
      PRINT *
      PRINT *,'    DM_VTFQD ICON = ',ICON
      PRINT *
      PRINT *,'    N = ',N,' :: NX = ',NX,' NY = ',NY,' NZ = ',NZ
      PRINT *,'    NBANDL = ',NBANDL,', NBANDR = ',NBANDR
      PRINT *,'    ITER MAX = ',ITMAX
      PRINT *,'    ITER = ',ITER
      PRINT *
      PRINT *,'    EPS = ',EPS
      PRINT *
      PRINT *,'    INITIAL ERROR = ',ERR1
      PRINT *,'    INITIAL RESIDUAL ERROR = ',ERR2
      PRINT *,'    CRITERIA RESIDUAL ERROR = ',ERR2*EPS
      PRINT *
      PRINT *,'    ERROR = ',ERR3
      PRINT *,'    RESIDUAL ERROR = ',ERR4
      PRINT *
      PRINT *

      IF(ERR4.LE.ERR2*EPS*1.1.AND.ICON.EQ.0)THEN
```

```
         WRITE(*,*)'********** OK **********'
      ELSE
         WRITE(*,*)'********** NG **********'
      ENDIF

      STOP
      END

C =======================================
C     INITIALIZE COEFFICIENT MATRIX
C =======================================
      SUBROUTINE INIT_MAT_DIAG(VA1,VA2,VA3,VC,D_L,OFFSET
     &                        ,NX,NY,NZ,XL,YL,ZL,NDIAG,LEN,NDIVP)
      IMPLICIT  REAL*8(A-H,O-Z)
      DIMENSION D_L(NDIVP,NDIAG)
      INTEGER   OFFSET(NDIAG)
C
      IF (NDIAG .LT. 1) THEN
        WRITE (*,*) 'SUBROUTINE INIT_MAT_DIAG:'
        WRITE (*,*) ' NDIAG SHOULD BE GREATER THAN OR EQUAL TO 1'
        RETURN
      ENDIF

!$OMP PARALLEL DEFAULT(PRIVATE)
!$OMP+     SHARED(VA1,VA2,VA3,VC,D_L,OFFSET
!$OMP+      ,NX,NY,NZ,XL,YL,ZL,NDIAG,LEN,NDIVP)

C NDIAG CANNOT BE GREATER THAN 7
      NDIAG_LOC = NDIAG
      IF (NDIAG .GT. 7) NDIAG_LOC = 7

C INITIAL SETTING
      HX = XL/(NX+1)
      HY = YL/(NY+1)
      HZ = ZL/(NZ+1)

!$OMP DO
      DO I = 1,NDIVP
      DO J = 1,NDIAG
      D_L(I,J) = 0.0
      ENDDO
      ENDDO
```

```
!$OMP ENDDO

      NXY = NX*NY

C OFFSET SETTING
!$OMP SINGLE
      L = 1
      IF (NDIAG_LOC .GE. 7) THEN
        OFFSET(L) = -NXY
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 5) THEN
        OFFSET(L) = -NX
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 3) THEN
        OFFSET(L) = -1
        L = L+1
      ENDIF
      OFFSET(L) = 0
      L = L+1
      IF (NDIAG_LOC .GE. 2) THEN
        OFFSET(L) = 1
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 4) THEN
        OFFSET(L) = NX
        L = L+1
      ENDIF
      IF (NDIAG_LOC .GE. 6) THEN
        OFFSET(L) = NXY
      ENDIF
!$OMP END SINGLE

C MAIN LOOP
!$OMP DO
      DO 100 J = 1,LEN
        JS = J

C DECOMPOSE JS-1 = (K0-1)*NX*NY+(J0-1)*NX+I0-1
        K0 = (JS-1)/NXY+1
        IF (K0 .GT. NZ) THEN
```

```
          PRINT*,'ERROR; K0.GH.NZ '
          GOTO 100
        ENDIF
        J0 = (JS-1-NXY*(K0-1))/NX+1
        I0 = JS - NXY*(K0-1) - NX*(J0-1)
        L = 1

        IF (NDIAG_LOC .GE. 7) THEN
          IF (K0 .GT. 1) D_L(J,L) = -(1.0/HZ+0.5*VA3)/HZ
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 5) THEN
          IF (J0 .GT. 1) D_L(J,L) = -(1.0/HY+0.5*VA2)/HY
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 3) THEN
          IF (I0 .GT. 1) D_L(J,L) = -(1.0/HX+0.5*VA1)/HX
          L = L+1
        ENDIF
        D_L(J,L) = 2.0/HX**2+VC
        IF (NDIAG_LOC .GE. 5) THEN
          D_L(J,L) = D_L(J,L) + 2.0/HY**2
          IF (NDIAG_LOC .GE. 7) THEN
            D_L(J,L) = D_L(J,L) + 2.0/HZ**2
          ENDIF
        ENDIF
        L = L+1
        IF (NDIAG_LOC .GE. 2) THEN
          IF (I0 .LT. NX) D_L(J,L) = -(1.0/HX-0.5*VA1)/HX
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 4) THEN
          IF (J0 .LT. NY) D_L(J,L) = -(1.0/HY-0.5*VA2)/HY
          L = L+1
        ENDIF
        IF (NDIAG_LOC .GE. 6) THEN
          IF (K0 .LT. NZ) D_L(J,L) = -(1.0/HZ-0.5*VA3)/HZ
        ENDIF
 100  CONTINUE
!$OMP ENDDO

!$OMP END PARALLEL
```

```
      RETURN
      END

C =======================================
* ABSOLUTE ERROR
* | X1 - X2 |
C =======================================
      REAL*8 FUNCTION ERRNRM(X1,X2,LEN)
      IMPLICIT  REAL*8 (A-H,O-Z)
      DIMENSION X1(*),X2(*)
C
      S = 0D0
      DO 100 I = 1,LEN
        SS = X1(I) - X2(I)
        S = S + SS * SS
 100  CONTINUE
C
      ERRNRM = SQRT( S )
      RETURN
      END
```

(4)　手法概要

TFQMR 法については“付録 1 参考文献一覧表”の[26]を参照してください.

DM_VTFQE

非対称または不定値のスパース行列の連立 1 次方程式 (TFQMR 法, ELLPACK 形式格納法)
CALL DM_VTFQE (A, K, IWIDT, N, ICOL, B, ITMAX, EPS, IGUSS, X, ITER, ICON)

(1)　機能

$n \times n$ の非対称／不定値なスパース行列を係数行列とする連立 1 次方程式を転置なし準最小残差法(TFQMR : Transpose-free Quasi Minimal Residual method)で解きます.

$$\boldsymbol{Ax} = \boldsymbol{b}$$

$n \times n$ の係数行列 $\boldsymbol{A}$ は, ELLPACK 形式の格納法で格納します.

ベクトル $\boldsymbol{b}$ および $\boldsymbol{x}$ は n 次元ベクトルです.

反復解法の収束および利用指針に関しては, “SSL II 拡張機能使用手引書 II 第 I 部 概説 第 4 章 連立 1 次方程式の反復解法と収束性” を参照してください.

(2)　パラメタ

A.....入力. 係数行列の非零要素を A(1:N, 1:IWIDT)に格納します.
A(K, IWIDT)なる 2 次元配列.
ELLPACK 形式の格納方法については, “SSL II 拡張機能使用手引書 II 第 I 部 概説 3.2.1.1 一般スパース行列の格納方法” を参照してください.

K.....入力. A および ICOL の 1 次元目の大きさ($\geq n$).

IWIDT.....入力. 係数行列 $\boldsymbol{A}$ の非零要素の行ベクトル方向の最大個数.
A および ICOL の 2 次元目の大きさ.

N.....入力. 行列 $\boldsymbol{A}$ の次数 n.

ICOL .....入力. ELLPACK 形式で使用される列指標で, A の対応する要素がいずれの列ベクトルに属すかを示します.
ICOL(K, IWIDT)なる 2 次元配列.

B.....入力. 連立 1 次方程式の右辺の定数ベクトルを B(1:N)に格納します.
B(N)なる 1 次元配列.

ITMAX .....入力. TFQMR 法の反復回数の上限値. 2000 程度で十分です.

EPS.....入力. 収束判定に用いられる判定値.
EPS が 0.0 以下のとき, 10^{-6} が設定されます.
((3)使用上の注意 a. 注意①参照).

IGUSS.....入力. 配列 X に指定された解ベクトルの近似値から反復計算を開始するかを示す制御情報.
0 のとき, 解ベクトルの近似値を指定しません.
0 以外のとき, 配列 X に指定された解ベクトルの近似値から反復計算を開始します.

X.....入力. 解ベクトルの近似値を X(1:N)に指定することができます.
出力. 解ベクトルが X(1:N)に格納されます.
X(N)なる 1 次元配列.
ITER.....出力. TFQMR 法での実際の反復回数.
ICON.....出力. コンディションコード.
表 DM_VTFQE-1 参照.

表 DM_VTFQE-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし	ー
20000	Break down を起こした.	処理を打ち切る.
20001	反復回数の上限に達した.	処理を打ち切る. 配列 X には, そのときまでに得られている近似値を出力するが, 精度は保証できない.
30000	K<1, IWIDT<1, N<1, ITMAX≤0, N>K	処理を打ち切る.
30001	バンド巾がゼロ.	

(3)　使用上の注意

a.　注意

①　本ルーチンは, 残差のユークリッドノルムが最初の残差のユークリッドノルムと EPS の積以下になったとき収束したと見なします.
正確な解と求められた近似解の誤差はほぼ行列 $\boldsymbol{A}$ の条件数と EPS の積に等しくなります.

b.　使用例

連立 1 次方程式 $\boldsymbol{Ax}=\boldsymbol{f}$ を解きます. 行列 $\boldsymbol{A}$ は境界条件として立方体の境界で 0 となる楕円型の偏微分方程式に有限差分法を適用して生成されます.

$$-\Delta u + a\nabla u + u = f$$

ここで $a = (a_1, a_2, a_3)$, a_1, a_2 および a_3 はある定数です. 行列 $\boldsymbol{A}$ はサブルーチン init_mat_ell によって生成されます.
(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT  REAL*8 (A-H,O-Z)
      PARAMETER (EPS = 1D-8)
      PARAMETER (NORD=60,NX =NORD ,NY = NORD,NZ = NORD,
     &           N = NX*NY*NZ)
      PARAMETER (K = N+1)
```

```
      PARAMETER (IWIDT = 7)
      DIMENSION ICOL(K,IWIDT)
      DIMENSION A(K,IWIDT)
      DIMENSION X(N),B(N),SOLEX(N),Y(N)

      PRINT *,'    BICGSTAB(L) METHOD'
      PRINT *,'    ELLPACK FORMAT'
      PRINT *

      SOLEX(1:N)=1.0D0
      PRINT *,'    EXPECTED SOLUTIONS'
      PRINT *,'    X(1) = ',SOLEX(1),' X(N) = ',SOLEX(N)
      PRINT *

      VA1 = 3D0
      VA2 = 1D0/3D0
      VA3 = 5D0
      VC = 1.0
      XL = 1.0
      YL = 1.0
      ZL = 1.0
      CALL INIT_MAT_ELL(VA1,VA2,VA3,VC,A,ICOL
     &          ,NX,NY,NZ,XL,YL,ZL,IWIDT,N,K)

      CALL DM_VMVSE(A,K,IWIDT,N,ICOL,SOLEX,B,ICON2)

      X(1:N)=0.0D0
      ERR1 = ERRNRM(SOLEX,X,N)
      CALL DM_VMVSE(A,K,IWIDT,N,ICOL,X,Y,ICON2)
      ERR2 = ERRNRM(Y,B,N)

      IGUSS = 0
      ITMAX = 2000

      CALL DM_VTFQE(A,K,IWIDT,N,ICOL,B,ITMAX
     &          ,EPS,IGUSS,X,ITER,ICON)

      ERR3 = ERRNRM(SOLEX,X,N)
      CALL DM_VMVSE(A,K,IWIDT,N,ICOL,X,Y,ICON2)
      ERR4 = ERRNRM(Y,B,N)

      PRINT *,'    COMPUTED VALUES'
```

```
      PRINT *,'    X(1) = ',X(1),' X(N) = ',X(N)
      PRINT *
      PRINT *,'    DM_VTFQE ICON = ',ICON
      PRINT *
      PRINT *,'    N = ',N,' :: NX = ',NX,' NY = ',NY,' NZ = ',NZ
      PRINT *,'    ITER MAX = ',ITMAX
      PRINT *,'    ITER = ',ITER
      PRINT *
      PRINT *,'    EPS = ',EPS
      PRINT *
      PRINT *,'    INITIAL ERROR = ',ERR1
      PRINT *,'    INITIAL RESIDUAL ERROR = ',ERR2
      PRINT *,'    CRITERIA RESIDUAL ERROR =',ERR2*EPS
      PRINT *
      PRINT *,'    ERROR = ',ERR3
      PRINT *,'    RESIDUAL ERROR = ',ERR4
      PRINT *
      PRINT *

      IF(ERR4.LE.ERR2*EPS*1.1.AND.ICON.EQ.0)THEN
         WRITE(*,*)'********** OK **********'
      ELSE
         WRITE(*,*)'********** NG **********'
      ENDIF

      STOP
      END

C ======================================
C INITILIZE COEFFICIENT MATRIX
C ======================================
      SUBROUTINE INIT_MAT_ELL(VA1,VA2,VA3,VC,A_L,ICOL_L,NX,NY,NZ
     &          ,XL,YL,ZL,IWIDTH,LEN,NDIVP)
      IMPLICIT  REAL*8(A-H,O-Z)
      DIMENSION A_L(NDIVP,IWIDTH)
      DIMENSION ICOL_L(NDIVP,IWIDTH)
C
      IF (IWIDTH .LT. 1) THEN
         WRITE (*,*) 'SUBROUTINE INIT_MAT_ELL:'
         WRITE (*,*) ' IWIDTH SHOULD BE GREATER THAN OR EQUAL TO 1'
         RETURN
      ENDIF
```

```
!$OMP PARALLEL DEFAULT(PRIVATE)
!$OMP+  SHARED(VA1,VA2,VA3,VC,A_L,ICOL_L,NX,NY,NZ
!$OMP+        ,XL,YL,ZL,IWIDTH,LEN,NDIVP)

C IWIDTH CANNOT BE GREATER THAN 7
      IWIDTH_LOC = IWIDTH
      IF (IWIDTH .GT. 7) IWIDTH_LOC = 7

C INITIAL SETTING
      HX = XL/(NX+1)
      HY = YL/(NY+1)
      HZ = ZL/(NZ+1)

!$OMP DO
      DO J = 1,IWIDTH
      DO I = 1,NDIVP
      A_L(I,J) = 0.0
      ICOL_L(I,J) = I
      ENDDO
      ENDDO
!$OMP ENDDO

C MAIN LOOP
!$OMP DO
      DO 100 J = 1,LEN
        JS = J
        L = 1

C DECOMPOSE JS-1 = (K0-1)*NX*NY+(J0-1)*NX+I0-1
        K0 = (JS-1)/NX/NY+1
        IF (K0 .GT. NZ) THEN
          PRINT*,' ERROR; K0.GT.NZ '
          GOTO 100
        ENDIF
        J0 = (JS-1-NX*NY*(K0-1))/NX+1
        I0 = JS - NX*NY*(K0-1) - NX*(J0-1)
        IF (IWIDTH_LOC .GE. 7) THEN
          IF (K0 .GT. 1) THEN
            A_L(J,L) = -(1.0/HZ+0.5*VA3)/HZ
            ICOL_L(J,L) = JS-NX*NY
            L = L+1
          ENDIF
```

```
ENDIF
IF (IWIDTH_LOC .GE. 5) THEN
  IF (J0 .GT. 1) THEN
    A_L(J,L) = -(1.0/HY+0.5*VA2)/HY
    ICOL_L(J,L) = JS-NX
    L = L+1
  ENDIF
ENDIF
IF (IWIDTH_LOC .GE. 3) THEN
  IF (I0 .GT. 1) THEN
    A_L(J,L) = -(1.0/HX+0.5*VA1)/HX
    ICOL_L(J,L) = JS-1
    L = L+1
  ENDIF
ENDIF
A_L(J,L) = 2.0/HX**2+VC
IF (IWIDTH_LOC .GE. 5) THEN
  A_L(J,L) = A_L(J,L) + 2.0/HY**2
  IF (IWIDTH_LOC .GE. 7) THEN
    A_L(J,L) = A_L(J,L) + 2.0/HZ**2
  ENDIF
ENDIF
ICOL_L(J,L) = JS
L = L+1
IF (IWIDTH_LOC .GE. 2) THEN
  IF (I0 .LT. NX) THEN
    A_L(J,L) = -(1.0/HX-0.5*VA1)/HX
    ICOL_L(J,L) = JS+1
    L = L+1
  ENDIF
ENDIF
IF (IWIDTH_LOC .GE. 4) THEN
  IF (J0 .LT. NY) THEN
    A_L(J,L) = -(1.0/HY-0.5*VA2)/HY
    ICOL_L(J,L) = JS+NX
    L = L+1
  ENDIF
ENDIF
IF (IWIDTH_LOC .GE. 6) THEN
  IF (K0 .LT. NZ) THEN
    A_L(J,L) = -(1.0/HZ-0.5*VA3)/HZ
    ICOL_L(J,L) = JS+NX*NY
```

```
          ENDIF
        ENDIF
 100  CONTINUE
!$OMP ENDDO

!$OMP END PARALLEL

      RETURN
      END

C =====================================
C ABSOLUTE ERROR
C | X1 - X2 |
C =====================================
      REAL*8 FUNCTION ERRNRM(X1,X2,LEN)
      IMPLICIT  REAL*8 (A-H,O-Z)
      DIMENSION X1(*),X2(*)
C
      S = 0D0
      DO 100 I = 1,LEN
        SS = X1(I) - X2(I)
        S = S + SS * SS
 100  CONTINUE
C
      ERRNRM = SQRT( S )
      RETURN
      END
```

(4)　手法概要

TFQMR 法については“付録 1 参考文献一覧表”の[26]を参照してください.

DM_VTRID

実対称行列の実対称三重対角行列への変換 (ハウスホルダー法)
CALL DM_VTRID(A, K, N, D, SL, ICON)

(1) 機能

実対称行列 $\boldsymbol{A}$ を Householder 変換で三重対角化します.

$$\boldsymbol{T} = \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{Q}$$

$\boldsymbol{A}$ は $n \times n$ 実対称行列, $\boldsymbol{Q}$ は $n \times n$ 直交行列, $\boldsymbol{T}$ は三重対角行列です.

(2) パラメタ

A.....入力. A(1:N, 1:N)の下三角部分$\{A(i, j)| i \geq j\}$に, 実対称行列$\boldsymbol{A}$の下三角部分$\{a_{ij} | i \geq j\}$を格納します.
出力. A(1:N, 1:N-2)の下三角部分$\{A(i, j)| i \geq j\}$に三重対角化を行うために利用した Householder 変換に関する情報が格納されます. 演算後, 上三角部分の値は保証されません. ((3)使用上の注意 a. 注意①参照).
A(K, N)なる倍精度実数型 2 次元配列.

K.....入力. 配列 A の 1 次元目の大きさ(≥ N).

N.....入力. 実対称行列 $\boldsymbol{A}$ の次数 n.

D.....出力. D(N)なる倍精度実数型の 1 次元配列で, 三重対角化された三重対角行列の対角要素を格納します.

SL.....出力. SL(N)なる倍精度実数型の 1 次元配列で, 三重対角化された三重対角行列の下副対角要素を SL(2:N)に格納します. SL(1) = 0.

ICON.....出力. コンディションコード.
表 DM_VTRID-1 参照

表 DM_VTRID-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし	ー
30000	N <2, K < N.	処理を打ち切る.

(3) 使用上の注意

a. 注意

① k=1, ... , n-2 まで, 以下の変換を繰り返して三重対角化します.

$$\boldsymbol{A}^{k} = \boldsymbol{Q}_k^{\mathrm{T}}\boldsymbol{A}^{k-1}\boldsymbol{Q}_k\ ,\quad \boldsymbol{A}^{0} = \boldsymbol{A}$$

$\boldsymbol{b}^{\mathrm{T}} = (0, \ldots , 0, A^{k-1}(k{+}1{:}n, k)^{\mathrm{T}})$とおきます.

$\boldsymbol{b}^{\mathrm{T}} = (0, \ldots , 0, b_{k+1}, \ldots , b_n)$

$\boldsymbol{b}^{\mathrm{T}}\cdot\boldsymbol{b} = S^2$ とし, $\boldsymbol{w}^{\mathrm{T}} = (0, \ldots , 0, b_{k+1}{+}S, b_{k+2}, \ldots , b_n)$とおきます.

b_{k+1} と S は同符号となるように S の符合を選びます.

すると, $\boldsymbol{Q}_k = \mathrm{I} - \alpha \boldsymbol{w} \cdot \boldsymbol{w}^{\mathrm{T}}, \alpha = \dfrac{1}{S^2 + |b_{k+i} S|}$ と表せます.

$\boldsymbol{A}(k+1{:}n, k)$に $\boldsymbol{w}(k+1{:}n)$, $\boldsymbol{A}(k, k)$にαを格納します.

b. 使用例

固有値が分かっている実対称行列の三重対角化を行います.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
c     **example**
      implicit real*8(a-h,o-z)
      parameter(n=2000,k=n)
      parameter(ne=n,max_nev=ne)
      dimension a(k,n),b(k,n),c(k,n),d(k,n),ac(k,n)
      dimension dd(n),sld(n),sud(n)
      dimension nev(5),mult(max_nev,2)
      dimension eval(max_nev),evec(k,max_nev)
cc
      pai=4.0d0*datan(1.0d0)
      coef=dsqrt(2.0d0/(n+1))
      do j=1,n
      do i=1,n
      d(i,j)=coef*dsin(pai/(n+1)*i*j)
      enddo
      enddo
cc
      do j=1,n
      do i=1,n
      if(i.eq.j)then
      c(i,j)=i
      else
      c(i,j)=0.0d0
      endif
      enddo
      enddo
cc
cc    d x c -> b
cc
      call dm_vmggm(d,k,c,k,b,k,n,n,n,icon)
cc
```

```
cc    b x d -> a
cc
      call dm_vmggm(b,k,d,k,a,k,n,n,n,icon)
cc
      do i=1,n
      do j=i,n
      ac(j,i)=a(j,i)
      enddo
      enddo
c
      call dm_vtrid( ac,k,n,dd,sld,icon )
      if(icon.ne.0)then
      print*,' icon of dm_vtrid =',icon
      stop
      endif
c
      do i=2,n
      sud(i-1)=sld(i)
      enddo
      sud(n)=0.0d0
c
      nf=1
      nl=n
      ivec=0
      eval_tol=1.0d-15
      clus_tol=1.0d-10
      call dm_vtdevc( dd,sld,sud,n,nf,nl,ivec,
     &                eval_tol,clus_tol,nev,
     &                eval,max_nev,evec,k,mult,icon )
      do i=1,ne,n/20
      print*,'eigen value in eval(',i,') = ',eval(i)
      enddo
c
      stop
      end
```

(4)　手法概要

本ルーチンは, 三重対角化で, ブロック化した Householder 法で三重対角化する方法(“付録 1 参考文献一覧表”の[30]参照.)を並列化したものです.

DM_V1DCFT

1 次元離散型複素フーリエ変換 (2, 3, 5 及び 7 の混合基底)
CALL DM_V1DCFT(X, KX, Y, KY, N1, N2, ISN, ICON)

(5)　機能

1 次元複素数フーリエ変換の正変換または逆変換を, 混合基底 FFT により行います.
変換データ長 $n(=n_1 \times n_2)$は, 2, 3, 5 及び 7 の巾の積で表すことのできる数です.

a.　1 次元フーリエ変換
$\{x_j\}$を入力し, (1.1)で定義する変換を行い, $\{n\ \alpha_k\}$を求めます.

$$n\alpha_k = \sum_{j=0}^{n-1} x_j \omega_n^{-jk} \quad , k = 0,1,...,n-1 \qquad (1.1)$$
$$, \omega_n = \exp(2\pi i / n)$$

b.　1 次元フーリエ逆変換
$\{\alpha_k\}$を入力し, (1.2)で定義する変換を行い, $\{x_j\}$を求めます.

$$x_j = \sum_{j=0}^{n-1} \alpha_k \omega_n^{jk} \quad , j = 0,1,...,n-1 \qquad (1.2)$$
$$, \omega_n = \exp(2\pi i / n)$$

データ長が十分大きくない場合, “SSL II 拡張使用手引書 II”の DVCFM1 の利用を勧めます.

(6)　パラメタ

X.....入力. 複素数データ. データは, X(1:N1, 1:N2)に格納します.
図 DM_V1DCFT-1 参照.
X(KX, N2)なる倍精度複素数型 2 次元配列.
((3)使用上の注意 a. 注意①参照)

KX.....入力. 入力データ配列 X の 1 次元目の大きさ. (≥ N1)
整数(INTEGER*4).

Y.....出力. 変換された複素数データ. データは, Y(1:N2, 1:N1)に格納されます.
図 DM_V1DCFT-1 参照.
Y(KY, N1)なる倍精度複素数型 2 次元配列.
((3)使用上の注意 a. 注意①参照).

KY.....入力. 配列 Y の 1 次元目の大きさ. (≥ N2)

N1.....入力. 変換データ長(n = N1 × N2)を 2 次元データとして見なしたときの, 1 次元目の大きさ. N1 は 2, 3, 5 及び 7 の巾の積で表すことのできる数.
((3)使用上の注意 a. 注意①参照).

N2.....入力. 変換データ長(n = N1 × N2)を 2 次元データとして見なしたときの, 2 次元目の大きさ. N2 は 2, 3, 5 及び 7 の巾の積で表すことのできる数.
((3)使用上の注意 a. 注意①参照).

ISN.....入力. 変換か逆変換かを指定.
変換　：ISN = 1
逆変換：ISN = -1

ICON.....出力. コンディションコード.
表 DM_V1DCFT-1 参照.

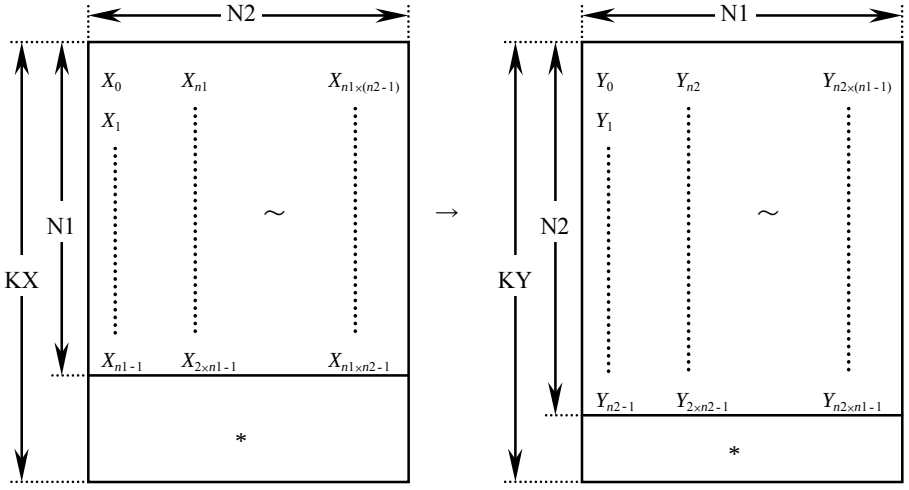


図 DM_V1DCFT-1　入力配列及び出力配列の格納形式

表 DM_V1DCFT-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし.	ー
30001	配列の次元が 0 または 0 以下であった.	処理を打ち切る.
30002	先導する次元の大きさが実際の次元の大きさより小さかった.	
30008	変換数が 2, 3, 5, 7 を基底としていない.	
30016	ISN の値が適切でない.	

(7)　使用上の注意

a.　注意

①　n=n_1× n_2個の 1 次元データを k=0, ... , n -1 と番号付けした時, 次のように書けます.

$$k = k_1 + k_2 \times n_1 \ , k_1 = 0,...,n_1 - 1$$
$$, k_2 = 0,...,n_2 - 1$$

$$k = i_1 + i_2 \times n_2 \ , i_1 = 0,...,n_2 - 1$$
$$, i_2 = 0,...,n_1 - 1$$

入力のデータは(k_1, k_2), 出力のデータは(i_1, i_2)を添字とする 2 次元データと見なしています. (図 DM_V1DCFT-1 参照)

② 一般的なフーリエ変換の定義

1 次元離散型複素フーリエ変換および, 逆変換は一般的に(3.1), (3.2)で定義されます.

$$\alpha_k = \frac{1}{n}\sum_{j=0}^{n-1} x_j \omega_n^{-jk}, k = 0,1,...,n-1 \tag{3.1}$$

$$x_j = \sum_{j=0}^{n-1} \alpha_k \omega_n^{jk}, j = 0,1,...,n-1 \tag{3.2}$$

ここで, $\omega_n = \exp(2\pi i / n)$

本サブルーチンでは, (3.1), (3.2)の左辺に対応して$\{n\alpha_k\}$または$\{x_j\}$を求めます. したがって, 結果の正規化は必要に応じて行って下さい.

b. 使用例

1 次元 FFT を計算します.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (N1=4000,N2=3000)
      PARAMETER (KX=N1+1,KY=N2+1)
      COMPLEX*16 X(KX,N2),Y(KY,N1)
      INTEGER ISN
*
*     Set up the input data arrays
*
!$OMP PARALLEL DO DEFAULT(PRIVATE) SHARED(X)
      DO I=1,N2
      DO J=1,N1
      X(J,I)=DCMPLX(FLOAT(J)+FLOAT(N1)*(I-1),0.0)
      ENDDO
      ENDDO
!$OMP END PARALLEL DO
*
*     Do the forward transform
*
```

```
      ISN=1
      CALL DM_V1DCFT(X,KX,Y,KY,N1,N2,ISN,ICON)
      IF(ICON.NE.0) THEN
        WRITE(*,*) 'error occurred : ',ICON
      ENDIF
*
*     Do the reverse transform
*
      ISN=-1
      CALL DM_V1DCFT(Y,KY,X,KX,N2,N1,ISN,ICON)
      IF(ICON.NE.0) THEN
        WRITE(*,*) 'error occurred : ',ICON
      ENDIF
*
*     Find the error after the forward and
*     inverse transform.
*
      ERROR=0

!$OMP PARALLEL DO DEFAULT(PRIVATE) SHARED(X)
!$OMP+         REDUCTION(MAX:ERROR)
      DO I=1,N2
      DO J=1,N1
        ERROR=MAX(ABS(DBLE(X(J,I))/N2/N1)-
     &  (FLOAT(J)+FLOAT(N1)*(I-1)),ERROR)
        ERROR=MAX(ABS(DIMAG(X(J,I))/N2/N1),
     &                              ERROR)
      ENDDO
      ENDDO
!$OMP END PARALLEL DO

      WRITE(*,*) 'Error ', ERROR
      STOP
      END
```

(8)　手法概要

本サブルーチンは, スカラ計算機向けの高性能な 1 次元フーリエ変換 DVCFM1(詳細は“SSL II 拡張機能使用手引書 II”参照.)を利用して実現しています.

DM_V1DCFT2

1 次元離散型複素フーリエ変換 (2, 3, 5 及び 7 の混合基底)
CALL DM_V1DCFT2(X, N, Y, ISN, ICON)

(1)　機能

1 次元複素数フーリエ変換の正変換または逆変換を, 混合基底 FFT により行います.
変換データ長 n は, 2, 3, 5 及び 7 の巾の積で表すことのできる数です.

a.　1 次元フーリエ変換

$\{x_j\}$を入力し, (1.1)で定義する変換を行い, $\{n\ \alpha_k\}$を求めます.

$$n\alpha_k = \sum_{j=0}^{n-1} x_j \omega_n^{-jk} \quad ,k = 0,1,...,n-1 \qquad ,\omega_n = \exp(2\pi i / n) \tag{1.1}$$

b.　1 次元フーリエ逆変換

$\{\alpha_k\}$を入力し, (1.2)で定義する変換を行い, $\{x_j\}$を求めます.

$$x_j = \sum_{k=0}^{n-1} \alpha_k \omega_n^{jk} \quad ,j = 0,1,...,n-1 \qquad ,\omega_n = \exp(2\pi i / n) \tag{1.2}$$

データ長が十分大きくない場合, “SSL II 拡張使用手引書 II”の DVCFM1 の利用を勧めます.

(2)　パラメタ

X.....入力. 複素数データ. データは, X(1:N)に格納します.
　　X(N)なる倍精度複素数型 1 次元配列.
N.....入力. 変換データ長(n).
Y.....出力. 変換された複素数データ. データは, Y(1:N)に格納されます.
　　Y(N)なる倍精度複素数型 1 次元配列.
ISN.....入力. 変換か逆変換かを指定.
　　変換　：ISN = 1
　　逆変換：ISN = -1
　　整数(INTEGER*4).
ICON.....出力. コンディションコード.
　　表 DM_V1DCFT2-1 参照.

表 DM_V1DCFT2-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし.	ー
30008	変換数が 2, 3, 5, 7 を基底としていない.	処理を打ち切る.
30016	ISN の値が適切でない.	

(3)　使用上の注意

a.　注意

①　一般的なフーリエ変換の定義

1 次元離散型複素フーリエ変換および, 逆変換は一般的に(3.1), (3.2)で定義されます.

$$\alpha_k = \frac{1}{n}\sum_{j=0}^{n-1} x_j \omega_n^{-jk}, k = 0,1,...,n-1 \tag{3.1}$$

$$x_j = \sum_{k=0}^{n-1} \alpha_k \omega_n^{jk}, j = 0,1,...,n-1 \tag{3.2}$$

ここで, $\omega_n = \exp(2\pi i / n)$

本サブルーチンでは, (3.1), (3.2)の左辺に対応して$\{n\alpha_k\}$または$\{x_j\}$を求めます. したがって, 結果の正規化は必要に応じて行って下さい.

b.　使用例

1 次元 FFT を計算します.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER (N1=1024,N2=N1,N=N1*N2)
      COMPLEX*16 X(N),Y(N),XX(N)
C
      DO I=1,N
      X(I)=DBLE(I)
      XX(I)=X(I)
      ENDDO
C
      CALL DM_V1DCFT2(X,N,Y,1,ICON)
      PRINT*,'ICON =',ICON
C
      CALL DM_V1DCFT2(Y,N,X,-1,ICON)
      PRINT*,'ICON =',ICON
```

```
C
      TMP=0.0D0
      DO I=1,N
      TMP=MAX(ABS(X(I)/DBLE(N)-XX(I)),TMP)
      ENDDO
      PRINT*,' ERROR =',TMP
C
      STOP
      END
```

(4)　手法概要

本サブルーチンは, スカラ計算機向けの高性能な 1 次元フーリエ変換 DVCFM1(詳細は“SSL II 拡張機能使用手引書 II”参照.)を利用して実現しています.

DM_V1DMCFT

1 次元多重離散型複素フーリエ変換 (2, 3, 5 及び 7 の混合基底)
CALL DM_V1DMCFT(X, KX, N, M, ISN, ICON)

(1) 機能

1 次元複素数フーリエ変換の正変換または逆変換を, 混合基底 FFT により多重に行います.
変換データ長 *n* は, 2, 3, 5 及び 7 の巾の積で表すことのできる数です.

a. 1 次元フーリエ変換

$\{x_j\}$を入力し, (1.1)で定義する変換を行い, $\{n\,\alpha_k\}$を求めます.

$$n\alpha_k = \sum_{j=0}^{n-1} x_j \omega_n^{-jk} \quad , k = 0,1,...,n-1 \qquad (1.1)$$
$$, \omega_n = \exp(2\pi i / n)$$

b. 1 次元フーリエ逆変換

$\{\alpha_k\}$を入力し, (1.2)で定義する変換を行い, $\{x_j\}$を求めます.

$$x_j = \sum_{j=0}^{n-1} \alpha_k \omega_n^{jk} \quad , j = 0,1,...,n-1 \qquad (1.2)$$
$$, \omega_n = \exp(2\pi i / n)$$

(2) パラメタ

X.....入力. 複素数データ. データは, X(1:N, 1:M)に格納します.
出力. 変換された複素数データ.
結果は, X(1:N, 1:M)に格納されます.
X(KX, M)なる倍精度複素数型 2 次元配列.
((3)使用上の注意 a. 注意①参照)

KX.....入力. 入力データ配列 X の 1 次元目の大きさ. (≥ N)

N.....入力. 変換データ長. N は 2, 3, 5 及び 7 の巾の積で表すことのできる数.

M.....入力. 変換するデータの多重度.

ISN.....入力. 変換か逆変換かを指定.
変換　：ISN = 1
逆変換：ISN = -1

ICON.....出力. コンディションコード.
表 DM_V1DMCFT-1 参照.

表 DM_V1DMCFT-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし.	ー
30001	配列の次元が 0 または 0 以下であった.	処理を打ち切る.
30002	先導する次元の大きさが実際の次元の大きさより小さかった.	
30008	変換数が 2, 3, 5, 7 を基底としていない.	
30016	ISN の値が適切でない.	

(3)　使用上の注意

a.　注意

①　一般的なフーリエ変換の定義

1 次元離散型複素フーリエ変換および, 逆変換は一般的に(3.1), (3.2)で定義されます.

$$\alpha_k = \frac{1}{n}\sum_{j=0}^{n-1} x_j \omega_n^{-jk}, k = 0,1,...,n-1 \tag{3.1}$$

$$x_j = \sum_{j=0}^{n-1} \alpha_k \omega_n^{jk}, j = 0,1,...,n-1 \tag{3.2}$$

ここで, $\omega_n = \exp(2\pi i / n)$

本サブルーチンでは, (3.1), (3.2)の左辺に対応して$\{n\alpha_k\}$または$\{x_j\}$を求めます. したがって, 結果の正規化は必要に応じて行って下さい.

b.　使用例

1 次元 FFT を多重に計算します.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (N1=2048,M=256)
      PARAMETER (KX=N1+1)
      COMPLEX*16 X(KX,M)
      INTEGER ISN

*
*     Set up the input data arrays
*
!$OMP PARALLEL DO DEFAULT(PRIVATE) SHARED(X)
```

```
      DO I=1,M
      DO J=1,N1
      X(J,I)=dcmplx(FLOAT(J)+FLOAT(N1)*(I-1),0.0)
      ENDDO
      ENDDO
!$OMP END PARALLEL DO

*
*     Do the forward transform
*
      ISN=1
      CALL dm_v1dmcft(X,KX,N1,M,ISN,ICON)
      IF(ICON.NE.0) THEN
        WRITE(*,*) 'error occurred : ',ICON
      ENDIF
*
*     Do the reverse transform
*
       ISN=-1
       CALL dm_v1dmcft(X,KX,N1,M,ISN,ICON)
       IF(ICON.NE.0) THEN
         WRITE(*,*) 'error occurred : ',ICON
       ENDIF
*
*     Find the error after the forward and
*     inverse transform.
*
      ERROR=0

!$OMP PARALLEL DO DEFAULT(PRIVATE) SHARED(X)
!$OMP+         REDUCTION(MAX:ERROR)
      DO I=1,M
      DO J=1,N1
      ERROR=MAX(ABS(dble(X(J,I))/N1-
     &  (FLOAT(J)+FLOAT(N1)*(I-1))),ERROR)
      ERROR=MAX(ABS(dimag(X(J,I))/N1),
     &                              ERROR)
      ENDDO
      ENDDO
!$OMP END PARALLEL DO

      WRITE(*,*) 'Error ', ERROR
```

```
      STOP
      END
```

(4)　手法概要

本サブルーチンは, スカラ計算機向けの高性能な 1 次元フーリエ変換 DVCFM1(詳細は“SSL II 拡張機能使用手引書 II”参照.)を利用して実現しています.

DM_V2DCFT

2 次元離散型複素フーリエ変換 (2, 3, 5 及び 7 の混合基底)
CALL DM_V2DCFT(X, KX, N1, N2, ISN, ICON)

(1) 機能

2 次元の複素フーリエ変換の正変換または逆変換を, 混合基底 FFT により行います.

2 次元データ(n_1, n_2)の各次元の大きさは, 2, 3, 5 及び 7 の巾の積で表すことのできる数です.

a. 2 次元フーリエ変換

$\{x_{j1j2}\}$を入力し, (1.1)で定義する変換を行い, $\{n_1 n_2 \alpha_{k1k2}\}$を求めます.

$$n_1 n_2 \alpha_{k1k2} = \sum_{j1=0}^{n1-1} \sum_{j2=0}^{n2-1} x_{j1j2} \omega_{n1}^{-j1k1} \omega_{n2}^{-j2k2} \qquad (1.1)$$

$$,k_1 = 0,1,...,n_1 - 1$$
$$,k_2 = 0,1,...,n_2 - 1$$
$$,\omega_{n1} = \exp(2\pi i / n_1)$$
$$,\omega_{n2} = \exp(2\pi i / n_2)$$

b. 2 次元フーリエ逆変換

$\{\alpha_{k1k2}\}$を入力し, (1.2)で定義する変換を行い, $\{x_{j1j2}\}$を求めます.

$$x_{j1j2} = \sum_{k1=0}^{n1-1} \sum_{k2=0}^{n2-1} \alpha_{k1k2} \cdot \omega_{n1}^{j1k1} \omega_{n2}^{j2k2}$$
$$, j_1 = 0,1,...,n_1 - 1$$
$$, j_2 = 0,1,...,n_2 - 1 \qquad (1.2)$$
$$,\omega_{n1} = \exp(2\pi i / n_1)$$
$$,\omega_{n2} = \exp(2\pi i / n_2)$$

(2) パラメタ

X.....入力. 複素数データ.
　　データは, X(1:N1, 1:N2)に格納します.
　　出力. 変換された複素数データ.
　　結果は, X(1:N1, 1:N2)に格納されます.
　　X(KX, N2)なる倍精度複素数型 2 次元配列.
　　((3)使用上の注意 a. 注意①参照).

KX.....入力. 入力データ配列 X の 1 次元目の大きさ. (≥ N1)

N1.....入力. 変換する 2 次元データの 1 次元目の大きさ n_1.
　　n_1は 2, 3, 5 及び 7 の巾の積で表すことのできる数.

N2.....入力. 変換する 2 次元データの 2 次元目の大きさ n_2.
　　n_2は 2, 3, 5 及び 7 の巾の積で表すことのできる数.

ISN.....入力. 変換か逆変換かを指定.
　　変換　：ISN = 1
　　逆変換：ISN = -1
ICON.....出力. コンディションコード.
　　表 DM_V2DCFT-1 参照.

表 DM_V2DCFT-1 コンディションコード

コード	意　　味	処理内容
0	エラーなし	ー
30001	配列の次元が 0 または 0 以下であった.	処理を打ち切る
30002	先導する次元の大きさが実際の次元の大きさより小さかった.	
30008	変換数が 2, 3, 5, 7 を基底としていない.	
30016	ISN の値が適切でない.	

(3)　使用上の注意

a.　注意

①　一般的なフーリエ変換の定義

2 次元離散型複素フーリエ変換および, 逆変換は一般的に(3.1), (3.2)で定義されます.

$$\alpha_{k1k2} = \frac{1}{n_1 n_2} \sum_{j1=0}^{n1-1} \sum_{j2=0}^{n2-1} x_{j1j2} \omega_{n1}^{-j1k1} \omega_{n2}^{-j2k2} \quad (3.1)$$
$$, k_1 = 0,1,...,n_1 - 1$$
$$, k_2 = 0,1,...,n_2 - 1$$

$$x_{j1j2} = \sum_{k1=0}^{n1-1} \sum_{k2=0}^{n2-1} \alpha_{k1k2} \omega_{n1}^{j2k2} \omega_{n2}^{j2k2} \quad (3.2)$$
$$, j_1 = 0,1,...,n_1 - 1$$
$$, j_2 = 0,1,...,n_2 - 1$$

ここで, $\omega_{n1}=\exp(2\pi i/n_1)$, $\omega_{n2}=\exp(2\pi i/n_2)$

本サブルーチンでは, (3.1), (3.2)の左辺に対応して$\{n_1 n_2 \alpha_{k1k2}\}$または$\{x_{j1j2}\}$を求めます. したがって, 結果の正規化は必要に応じて行って下さい.

b.　使用例

2 次元 FFT を計算します.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C      **EXAMPLE**
       IMPLICIT REAL*8 (A-H,O-Z)
```

```
      PARAMETER (N1=4000,N2=3000)
      PARAMETER (KX=4400)
      COMPLEX*16 X(KX,N2)
      INTEGER ISN

*
*      Set up the input data arrays
*
!$OMP PARALLEL DO DEFAULT(PRIVATE) SHARED(X)
       DO I=1,N2
       DO J=1,N1
       X(J,I)=DCMPLX(FLOAT(J)+FLOAT(N1)*(I-1),0.0)
       ENDDO
       ENDDO
!$OMP END PARALLEL DO

*
*     Do the forward transform
*
       ISN=1
       CALL DM_V2DCFT(X,KX,N1,N2,ISN,ICON)
       IF(ICON.NE.0) THEN
         WRITE(*,*) 'error occurred : ',ICON
       ENDIF
*
*     Do the reverse transform
*
       ISN=-1
       CALL DM_V2DCFT(X,KX,N1,N2,ISN,ICON)
       IF(ICON.NE.0) THEN
         WRITE(*,*) 'error occurred : ',ICON
       ENDIF
*
*     Find the error after the forward and
*     inverse transform.
*
      ERROR=0

!$OMP PARALLEL DO DEFAULT(PRIVATE) SHARED(X)
!$OMP+         REDUCTION(MAX:ERROR)
      DO I=1,N2
      DO J=1,N1
```

```
            ERROR=MAX(ABS(DBLE(X(J,I))/(N2*N1)-
           &  (FLOAT(J)+FLOAT(N1)*(I-1))),ERROR)
            ERROR=MAX(ABS(DIMAG(X(J,I))/(N2*N1)),
           &                              ERROR)
          ENDDO
          ENDDO
!$OMP END PARALLEL DO

          WRITE(*,*) 'Error ', ERROR
          STOP
          END
```

(4)　手法概要

本サブルーチンは, スカラ計算機向けの高性能な 1 次元フーリエ変換 DVCFM1(詳細は“SSL II 拡張機能使用手引書 II”参照.)を利用して実現しています.

DM_V3DCFT

3 次元離散型複素フーリエ変換 (2, 3, 5 及び 7 の混合基底)
CALL DM_V3DCFT(X, KX, N1, N2, N3, ISN, ICON)

(1) 機能

3 次元複素フーリエ変換の正変換または逆変換を, 混合基底 FFT により行います.

3 次元データ(n_1, n_2, n_3)の各次元の大きさは, 2, 3, 5 及び 7 の巾の積で表すことのできる数です.

a. 3 次元フーリエ変換

$\{x_{j1j2j3}\}$を入力し, (1.1)で定義する変換を行い, $\{n_1n_2n_3\alpha_{k1k2k3}\}$を求めます.

$$n_1n_2n_3\alpha_{k1k2k3} = \sum_{j1=0}^{n1-1}\sum_{j2=0}^{n2-1}\sum_{j3=0}^{n3-1} x_{j1j2j3}\omega_{n1}^{-j1k1}\omega_{n2}^{-j2k2}\omega_{n3}^{-j3k3} \qquad (1.1)$$

$$,k_1 = 0,1,...,n_1-1$$
$$,k_2 = 0,1,...,n_2-1$$
$$,k_3 = 0,1,...,n_3-1$$
$$,\omega_{n1} = \exp(2\pi i/n_1)$$
$$,\omega_{n2} = \exp(2\pi i/n_2)$$
$$,\omega_{n3} = \exp(2\pi i/n_3)$$

b. 3 次元フーリエ逆変換

$\{\alpha_{k1k2k3}\}$を入力し, (1.2)で定義する変換を行い, $\{x_{j1j2j3}\}$を求めます.

$$x_{j1j2j3} = \sum_{k1=0}^{n1-1}\sum_{k2=0}^{n2-1}\sum_{k3=0}^{n3-1}\alpha_{k1k2k3}\cdot\omega_{n1}^{j1k1}\omega_{n2}^{j2k2}\omega_{n3}^{j3k3}$$

$$,j_1 = 0,1,...,n_1-1$$
$$,j_2 = 0,1,...,n_2-1$$
$$,j_3 = 0,1,...,n_3-1 \qquad (1.2)$$
$$,\omega_{n1} = \exp(2\pi i/n_1)$$
$$,\omega_{n2} = \exp(2\pi i/n_2)$$
$$,\omega_{n3} = \exp(2\pi i/n_3)$$

(2) パラメタ

X.....入力. 複素数データ.
データは, X(1:N1, 1:N2, 1:N3)に格納します.
出力. 変換された複素数データ.
結果は, X(1:N1, 1:N2, 1:N3)に格納されます.
X(KX, N2, N3)なる倍精度複素数型 3 次元配列.

KX.....入力. 入力データ配列 X の 1 次元目の大きさ. (≥ N1)

N1.....入力. 変換する 3 次元データの 1 次元目の大きさ n_1.
n_1は, 2, 3, 5 及び 7 の巾の積で表すことのできる数であること.

N2.....入力. 変換する 3 次元データの 2 次元目の大きさ n_2.
n_2 は, 2, 3, 5 及び 7 の巾の積で表すことのできる数であること.
N3.....入力. 変換する 3 次元データの 3 次元目の大きさ n_3.
n_3 は, 2, 3, 5 及び 7 の巾の積で表すことのできる数であること.
ISN.....入力. 変換か逆変換かを指定.
変換　：ISN = 1
逆変換：ISN = -1
ICON.....出力. コンディションコード.
表 DM_V3DCFT-1 参照.

表 DM_V3DCFT-1 コンディションコード

コード	意　　味	処理内容
0	エラーなし.	ー
30001	配列の次元が 0 または 0 以下であった.	処理を打ち切る.
30002	先導する次元の大きさが実際の次元の大きさより小さかった.	
30008	変換数が 2, 3, 5, 7 を基底としていない.	
30016	ISN の値が適切でない.	

(3)　使用上の注意

a.　注意

①　一般的なフーリエ変換の定義

3 次元離散型複素フーリエ変換および, 逆変換は一般的に(3.1), (3.2)で定義されます.

$$\alpha_{k1k2k3} = \frac{1}{n_1 n_2 n_3} \sum_{j1=0}^{n1-1} \sum_{j2=0}^{n2-1} \sum_{j3=0}^{n3-1} x_{j1j2j3} \omega_{n1}^{-j1k1} \omega_{n2}^{-j2k2} \omega_{n3}^{-j3k3} \quad (3.1)$$
$$, k_1 = 0,1,...,n_1 - 1$$
$$, k_2 = 0,1,...,n_2 - 1$$
$$, k_3 = 0,1,...,n_3 - 1$$

$$x_{j1j2j3} = \sum_{k1=0}^{n1-1} \sum_{k2=0}^{n2-1} \sum_{k3=0}^{n3-1} \alpha_{k1k2k3} \omega_{n1}^{j1k1} \omega_{n2}^{j2k2} \omega_{n3}^{j3k3} \quad (3.2)$$
$$, j_1 = 0,1,...,n_1 - 1$$
$$, j_2 = 0,1,...,n_2 - 1$$
$$, j_3 = 0,1,...,n_3 - 1$$

ここで, $\omega_{n1} = \exp(2\pi i / n_1)$, $\omega_{n2} = \exp(2\pi i / n_2)$, $\omega_{n3} = \exp(2\pi i / n_3)$

本サブルーチンでは, (3.1), (3.2)の左辺に対応して $\{n_1 n_2 n_3 \alpha_{k1k2k3}\}$ または $\{x_{j1j2j3}\}$ を求めます. したがって, 結果の正規化は必要に応じて行って下さい.

b. 使用例

3 次元 FFT を計算します.
(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (N1=400,N2=100,N3=200)
      PARAMETER (KX=440)
      COMPLEX*16 X(KX,N2,N3)
      INTEGER ISN
*
*     Set up the input data arrays
*
!$OMP PARALLEL DO DEFAULT(PRIVATE) SHARED(X)
      DO K=1,N3
      DO I=1,N2
      DO J=1,N1
      X(J,I,K)=DCMPLX(FLOAT(J)+FLOAT(N1)*(I-1),0.0)
      ENDDO
      ENDDO
      ENDDO
!$OMP END PARALLEL DO

*
*     Do the forward transform
*
      ISN=1
      CALL DM_V3DCFT(X,KX,N1,N2,N3,ISN,ICON)
      IF(ICON.NE.0) THEN
        WRITE(*,*) 'error occurred : ',ICON
      ENDIF
*
*    Do the reverse transform
*
      ISN=-1
      CALL DM_V3DCFT(X,KX,N1,N2,N3,ISN,ICON)
      IF(ICON.NE.0) THEN
        WRITE(*,*) 'error occurred : ',ICON
      ENDIF
*
```

```
*    Find the error after the forward and
*    inverse transform.
*
      ERROR=0
!$OMP PARALLEL DO DEFAULT(PRIVATE) SHARED(X)
!$OMP+         REDUCTION(MAX:ERROR)

      DO K=1,N3
      DO I=1,N2
      DO J=1,N1
      ERROR=MAX(ABS(DBLE(X(J,I,K))/(N3*N2*N1)-
     &      (FLOAT(J)+FLOAT(N1)*(I-1))),ERROR)
      ERROR=MAX(ABS(DIMAG(X(J,I,K))/(N3*N2*N1)),
     &                           ERROR)
      ENDDO
      ENDDO
      ENDDO
!$OMP END PARALLEL DO

      WRITE(*,*) 'Error ', ERROR
      STOP
      END
```

(4)　手法概要

本サブルーチンは, スカラ計算機向けの高性能な 1 次元フーリエ変換 DVCFM1(詳細は“SSL II 拡張機能使用手引書 II”参照.)を利用して実現しています.

DM_V3DCFT2

3 次元離散型複素フーリエ変換 (2, 3, 5 及び 7 の混合基底)
CALL DM_V3DCFT2(X, K1, K2, N1, N2, N3, ISN, ICON)

(1) 機能

3 次元複素フーリエ変換の正変換または逆変換を行います.

3 次元データ(n_1, n_2, n_3)の各次元の大きさは, 2, 3, 5, 7 の巾の積として表される数でなければなりません.

a. 3 次元フーリエ変換

$\{x_{j1j2j3}\}$を入力し, (1.1)で定義する変換を行い, $\{n_1n_2n_3\alpha_{k1k2k3}\}$を求めます.

$$n_1n_2n_3\alpha_{k1k2k3} = \sum_{j1=0}^{n1-1}\sum_{j2=0}^{n2-1}\sum_{j3=0}^{n3-1} x_{j1j2j3}\omega_{n1}^{-j1k1}\omega_{n2}^{-j2k2}\omega_{n3}^{-j3k3} \qquad (1.1)$$

$$,k_1 = 0,1,...,n_1-1$$
$$,k_2 = 0,1,...,n_2-1$$
$$,k_3 = 0,1,...,n_3-1$$
$$,\omega_{n1} = \exp(2\pi i/n_1)$$
$$,\omega_{n2} = \exp(2\pi i/n_2)$$
$$,\omega_{n3} = \exp(2\pi i/n_3)$$

b. 3 次元フーリエ逆変換

$\{\alpha_{k1k2k3}\}$を入力し, (1.2)で定義する変換を行い, $\{x_{j1j2j3}\}$を求めます.

$$x_{j1j2j3} = \sum_{k1=0}^{n1-1}\sum_{k2=0}^{n2-1}\sum_{k3=0}^{n3-1}\alpha_{k1k2k3}\cdot\omega_{n1}^{j1k1}\omega_{n2}^{j2k2}\omega_{n3}^{j3k3}$$

$$,j_1 = 0,1,...,n_1-1$$
$$,j_2 = 0,1,...,n_2-1$$
$$,j_3 = 0,1,...,n_3-1 \qquad (1.2)$$
$$,\omega_{n1} = \exp(2\pi i/n_1)$$
$$,\omega_{n2} = \exp(2\pi i/n_2)$$
$$,\omega_{n3} = \exp(2\pi i/n_3)$$

(2) パラメタ

X.....入力. 複素数データ.
データは, X(1:N1, 1:N2, 1:N3)に格納します.
出力. 変換された複素数データ.
結果は, X(1:N1, 1:N2, 1:N3)に格納されます.
X(K1, K2, N3)なる倍精度複素数型 3 次元配列.

K1.....入力. 入力データ配列 X の 1 次元目の大きさ. (≥ N1)

K2.....入力. 入力データ配列 X の 2 次元目の大きさ. (≥ N2)

N1.....入力. 変換する 3 次元データの 1 次元目の大きさ n_1.

N2.....入力. 変換する 3 次元データの 2 次元目の大きさ n_2.

N3.....入力. 変換する 3 次元データの 3 次元目の大きさ n_3.

ISN.....入力. 変換か逆変換かを指定.

変換　：ISN = 1

逆変換：ISN = -1

ICON.....出力. コンディションコード.

表 DM_V3DCFT2-1 参照.

表 DM_V3DCFT2-1 コンディションコード

コード	意　　味	処理内容
0	エラーなし.	ー
30000	n_1, n_2, n_3 が 0 または 0 以下であった. K1<N1, K2<N2 または ISN の値が適切でない.	処理を打ち切る.
30008	変換数が 2, 3, 5, 7 を基底としていない.	

(3)　使用上の注意

a.　注意

①　一般的なフーリエ変換の定義

3 次元離散型複素フーリエ変換および, 逆変換は一般的に(3.1), (3.2)で定義されます.

$$\alpha_{k1k2k3} = \frac{1}{n_1 n_2 n_3} \sum_{j1=0}^{n1-1} \sum_{j2=0}^{n2-1} \sum_{j3=0}^{n3-1} x_{j1j2j3} \omega_{n1}^{-j1k1} \omega_{n2}^{-j2k2} \omega_{n3}^{-j3k3} \quad (3.1)$$

$$, k_1 = 0,1,..., n_1 - 1$$

$$, k_2 = 0,1,..., n_2 - 1$$

$$, k_3 = 0,1,..., n_3 - 1$$

$$x_{j1j2j3} = \sum_{k1=0}^{n1-1} \sum_{k2=0}^{n2-1} \sum_{k3=0}^{n3-1} \alpha_{k1k2k3} \omega_{n1}^{j1k1} \omega_{n2}^{j2k2} \omega_{n3}^{j3k3} \quad (3.2)$$

$$, j_1 = 0,1,..., n_1 - 1$$

$$, j_2 = 0,1,..., n_2 - 1$$

$$, j_3 = 0,1,..., n_3 - 1$$

ここで, $\omega_{n1} = \exp(2\pi i / n_1)$, $\omega_{n2} = \exp(2\pi i / n_2)$, $\omega_{n3} = \exp(2\pi i / n_3)$

本サブルーチンでは, (3.1), (3.2)の左辺に対応して$\{n_1 n_2 n_3 \alpha_{k1k2k3}\}$または$\{x_{j1j2j3}\}$を求めます. したがって, 結果の正規化は必要に応じて行って下さい.

b.　使用例

3 次元 FFT を計算します.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
c     **example**
      implicit real*8 (a-h,o-z)
      parameter (n1=128,n2=128,n3=128)
      parameter (k1=n1+1,k2=n2)
      complex*16 x(k1,k2,n3)
      integer isn
*
*     set up the input data arrays
*
!$omp parallel do default(private) shared(x)
      do k=1,n3
      do i=1,n2
      do j=1,n1
      x(j,i,k)=dcmplx(float(j)+float(n1)*(i-1),0.0)
      enddo
      enddo
      enddo
!$omp end parallel do

*
*     do the forward transform
*
      isn=1
      call dm_v3dcft2(x,k1,k2,n1,n2,n3,isn,icon)
      if(icon.ne.0) then
        write(*,*) 'error occurred : ',icon
      endif
*
*    do the reverse transform
*
      isn=-1
      call dm_v3dcft2(x,k1,k2,n1,n2,n3,isn,icon)
      if(icon.ne.0) then
        write(*,*) 'error occurred : ',icon
      endif
*
*    find the error after the forward and
*    inverse transform.
*
      error=0
!$omp parallel do default(private) shared(x)
!$omp+         reduction(max:error)
```

```
      do k=1,n3
      do i=1,n2
      do j=1,n1
      error=max(abs(dble(x(j,i,k))/(n3*n2*n1)-
     &      (float(j)+float(n1)*(i-1))),error)
      error=max(abs(dimag(x(j,i,k))/(n3*n2*n1)),
     &                          error)
      enddo
      enddo
      enddo
!$omp end parallel do

      write(*,*) 'error=', error
      stop
      end
```

(4)　手法概要

本サブルーチンは, スカラ計算機向けの高性能な 1 次元フーリエ変換 DVCFM1(詳細は“SSL II 拡張機能使用手引書 II”参照.)を利用して実現しています.

DM_V1DRCF

1 次元離散型実フーリエ変換 (2, 3, 5 及び 7 の混合基底)
CALL DM_V1DRCF(X, KX, Y, KY, N1, N2, ISIN, ISN, ICON)

(1) 機能

1 次元実フーリエ変換の正変換, 逆変換を, 混合基底 FFT により行います.
データ数 $n(=n_1 \times n_2)$は, 2, 3, 5 及び 7 の巾の積で表すことのできる数です.

a. 1 次元フーリエ変換

$\{x_j\}$を入力し, (1.1)で定義する変換を行い, $\{n\alpha_k\}$を求めます.

$$n\alpha_k = \sum_{j=0}^{n-1} x_j \omega_n^{-jkr} \quad , k = 0,1,...,n-1 \qquad , \omega_n = \exp(2\pi i / n) \tag{1.1}$$

$r=1$ または $r=-1$

b. 1 次元フーリエ逆変換

$\{\alpha_k\}$を入力し, (1.2)で定義する変換を行い, $\{x_j\}$を求めます.

$$x_j = \sum_{k=0}^{n-1} \alpha_k \omega_n^{jkr} \quad , j = 0,1,...,n-1 \qquad , \omega_n = \exp(2\pi i / n) \tag{1.2}$$

$r=1$ または $r=-1$

本ルーチンは, n を適当な数 n_1 及び n_2 に分解するとき, DM_V1DRCF2 より 30%以上高速です.

(2) パラメタ

X.....入力／出力. 実データ.
データは, X(1:N1, 1:N2)に格納します.
実数から複素数への変換のとき(ISN = 1)入力となり, 複素数から実数への変換のとき(ISN = -1)出力となります. ISN = 1 のとき, 入力データは保存されません.
X(KX, N2)なる倍精度実数型 2 次元配列.
図 DM_V1DRCF-1 参照.
((3)使用上の注意 a. 注意①参照).

KX.....入力. 配列 X の 1 次元目の大きさ(≥ N1).
整数(INTEGER*4).

Y.....出力／入力. 変換された複素数データ.
データは, Y(1:N2/2+1, 1:N1)に格納します.
実数から複素数への変換のとき(ISN = 1)出力となり, 複素数から実数への変換のとき(ISN = -1)入力となります.

ISN = -1 のとき入力データは保存されません.

Y (KY, N1)なる倍精度複素数型 2 次元配列.

図 DM_V1DRCF-1 参照.

((3)使用上の注意 a. 注意③参照).

KY.....入力. 配列 Y の 1 次元目の大きさ. (KY ≥ N2/2+1)

整数(INTEGER*4).

N1.....入力. 変換する実データ($n=n1 \times n2$)を 2 次元データとみなしたときの, 1 次元目の大きさ.

N1 は 2, 3, 5 及び 7 の巾の積で表すことのできる数であること.

N1 × N2 は変換されるデータ列の長さです.

整数(INTEGER*4).

((3)使用上の注意 a. 注意①, ④参照).

N2.....入力. 変換する実データ($n=n1 \times n2$)を 2 次元データとみなしたときの, 2 次元目の大きさ.

N2 は 2, 3, 5 及び 7 の巾の積で表すことのできる数であること.

N1 × N2 は変換されるデータ列の長さです.

整数(INTEGER*4).

((3)使用上の注意 a. 注意①, ④参照).

ISIN.....入力. 変換の方向を表します.

1 のとき　$r = 1$

-1 のとき　$r = -1$

ISN.....入力. 変換か逆変換かを指定.

変換　：ISN = 1

逆変換：ISN = -1

整数(INTEGER*4).

ICON.....出力. コンディションコード.

表 DM_V1DRCF-1 参照.

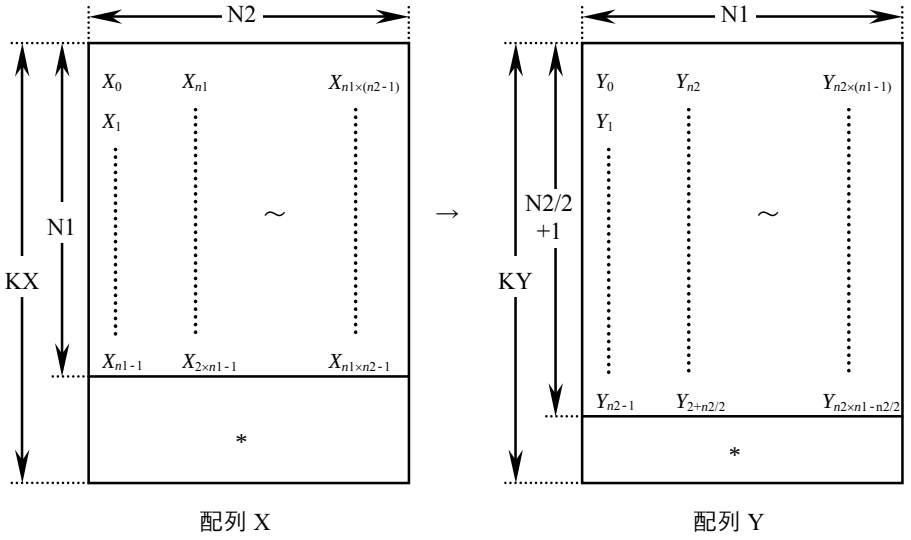


図 DM_V1DRCF-1　データの配列への格納形式

表 DM_V1DRCF-1　コンディションコード

コード	意　　味	処理内容
0	エラーなし.	ー
30000	KX<N1, KY<N2/2+1, N1<1, N2<1, ISIN≠1, -1, ISN≠1, -1.	処理を打ち切る.
30008	変換数が 2, 3, 5, 7 を基底としていない.	

(3)　使用上の注意

a.　注意

①　$n=n_1\times n_2$ 個の 1 次元データを $k=0, \ldots, n-1$ と番号付けした時,

$$k = k_1 + k_2 \times n_1 \ , k_1 = 0,\ldots,n_1 - 1$$
$$\qquad\qquad ,k_2 = 0,\ldots,n_2 - 1$$
$$k = i_1 + i_2 \times n_2 \ , i_1 = 0,\ldots,n_2 - 1$$
$$\qquad\qquad ,i_2 = 0,\ldots,n_1 - 1$$

と書けます. 実データは(k_1, k_2), 複素データは(i_1, i_2)を添字とする 2 次元データと見なしています. ただし $i_1=0, \ldots, n_2/2$ が Y に格納されます.

(図 DM_V1DRCF-1 参照)

②　一般的なフーリエ変換の定義

1 次元離散型複素フーリエ変換および, 逆変換は一般的に(3.1), (3.2)で定義されます.

$$\alpha_k = \frac{1}{n}\sum_{j=0}^{n-1} x_j \omega_n^{-jk} \quad , k=0, 1, \ldots, n-1 \tag{3.1}$$

$$x_j = \sum_{k=0}^{n-1} \alpha_k \omega_n^{jk} \quad , j=0, 1, \ldots, n-1 \tag{3.2}$$

ここで, $\omega_n = \exp(2\pi i/n)$

本サブルーチンでは, (3.1), (3.2)の左辺に対応して$\{n\alpha_k\}$または$\{x_j\}$を求めます. したがって, 結果の正規化は必要に応じて行って下さい.

③　1 次元実フーリエ変換の結果は次の複素共役(¯ で示します)の関係があります.

$$\alpha_k = \overline{\alpha_{n-k}} , k = 1,\ldots,n-1$$

$n=n_1\times n_2$

$i_1=0, 1, \ldots, n_2-1$

$i_2=0, 1, \ldots, n_1-1$

$k=i_1+i_2\times n_2$ とすると,

$n-k=n_2-i_1+(n_1-1-i_2)\times n_2$

であり, $i_1=1, \ldots, n_2/2$ までのデータ(0 を除いた前半部分)から残りのデータは求めることができます.

④　n がほぼ等しい十分大きな数 n_1 および n_2 の積に分解できるとき, 最も性能が良くなります.

b.　使用例

1 次元実 FFT を計算します.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER (N1=1024,N2=1024,KX=N1+1,KY=N2/2+1+1)
      REAL*8     X(KX,N2), XX(N1,N2)
      COMPLEX*16 Y(KY,N1)
CC
      DO I=1,N2
      DO J=1,N1
      X(J,I)=J+N1*(I-1)
      XX(J,I)=X(J,I)
      ENDDO
      ENDDO
      ISW=1
      CALL DM_V1DRCF(X,KX,Y,KY,N1,N2,1,ISW,ICON)
      PRINT*,' ICON =',ICON
CC
      ISW=-1
      CALL DM_V1DRCF(X,KX,Y,KY,N1,N2,1,ISW,ICON)
      PRINT*,' ICON =',ICON
CC
      TMP=0.0D0
      DO I=1,N2
      DO J=1,N1
      TMP=MAX(DABS(DBLE(X(J,I))/DBLE(N1)/DBLE(N2)
     $        -DBLE(XX(J,I))),TMP)
      ENDDO
      ENDDO
CC
      PRINT*,' ERROR  =',TMP
      STOP
      END
```

(4)　手法概要

本サブルーチンは, スカラ計算機向けの高性能な 1 次元フーリエ変換 DVCFM1(詳細は “SSL II 拡張機能使用手引書 II”参照.)を利用して実現しています.

DM_V1DRCF2

1 次元離散型実フーリエ変換 (2, 3, 5 及び 7 の混合基底)
CALL DM_V1DRCF2(X, N, Y, ISIN, ISN, ICON)

(1) 機能

1 次元の実フーリエ変換の正変換および逆変換を, 混合基底 FFT により行います.

1 次元データの大きさ n は, 2, 3, 5, 7 の巾の積で表せる数です.

a. フーリエ変換

実数列$\{x_j\}$を入力して, 複素数列$\{n\alpha_k\}$を求めます.

$$n\alpha_k = \sum_{j=0}^{n-1} x_j \omega_n^{-jkr} \quad , k = 0,1,...,n-1 \qquad , \omega_n = \exp(2\pi i / n) \tag{1.1}$$

$r=1$ または $r=-1$

b. フーリエ逆変換

複素数列$\{\alpha_k\}$を入力して, 実数列$\{x_j\}$を求めます.

$$x_j = \sum_{k=0}^{n-1} \alpha_k \omega_n^{jkr} \quad , j = 0,1,...,n-1 \qquad , \omega_n = \exp(2\pi i / n) \tag{1.2}$$

$r=1$ または $r=-1$

(2) パラメタ

X.....入力／出力. 実数列が X(1:N)に格納されます.
変換(実数から複素数への変換)のとき入力となり, 逆変換(複素数から実数への変換)のとき出力となります.
X(N)なる倍精度実数型の 1 次元配列.

N.....入力. 変換する実数データの数.
N は 2, 3, 5, 7 の巾乗の積に分解できる偶数であること.

Y.....出力／入力. 約半分の複素数データが Y(1:N/2+1)に格納されます.
((3)使用上の注意 a. 注意①参照).
変換(実数から複素数への変換)のとき出力となり, 逆変換(複素数から実数への変換)のとき入力となります.
Y(N/2+1)なる倍精度複素数型の 1 次元配列.

ISIN.....入力. フーリエ変換の符号を表します. 1 または-1

ISN.....入力. 実数から複素数への変換か, 複素数から実数への変換かを示します.
変換(実数から複素数への変換)の場合 : 1
逆変換(複素数から実数への変換)の場合 : -1

ICON.....出力. コンディションコード.
表 DM_V1DRCF2-1 参照.

表 DM_V1DRCF2-1 コンディションコード

コード	意　　味	処理内容
0	エラーなし.	ー
30000	N が 2 の倍数でない. または, N が 2, 3, 5 及び 7 の巾乗で表現できない. または, ISW, ISIN の値が正しくない.	処理を打ち切る.

(3)　使用上の注意

a.　注意

①　1 次元実フーリエ変換した結果は, 次の複素共役の関係(¯ で示す)があります.
α_k, $k = 0, \ldots, n-1$ としたとき,

$\alpha_k = \overline{\alpha}_{n-k}$　0 を除いた, $k = 1, \ldots, n-1$

②　1 次元実フーリエ変換及び, 逆変換は一般的に(3.1), (3.2)と定義されます.
実数列$\{x_j\}$を入力して, 複素数列$\{\alpha_k\}$を求めます.

$$\alpha_k = \frac{1}{n}\sum_{j=0}^{n-1} x_j \omega_n^{-jk} \quad , k=0, 1, \ldots, n-1 \tag{3.1}$$

複素数列$\{\alpha_k\}$を入力して, 実数列$\{x_j\}$を求めます.

$$x_j = \sum_{k=0}^{n-1} \alpha_k \omega_n^{jk} \quad , j=0, 1, \ldots, n-1 \tag{3.2}$$

本サブルーチンでは, (3.1), (3.2)の左辺に対して$\{n\alpha_k\}$, $\{x_j\}$を求めるので, 正規化は必要に応じて行ってください.

b.　使用例

1 次元実 FFT を計算します.
(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER (N1=1024,N2=N1,N=N1*N2)
      REAL*8     X(N)
      COMPLEX*16 Y(N/2+1),XX(N)
C
      DO I=1,N
      X(I)=DBLE(I)
      XX(I)=X(I)
      ENDDO
C
```

```
      CALL DM_V1DRCF2(X,N,Y,1,1,ICON)
      PRINT*,'ICON =',ICON
C
      CALL DM_V1DRCF2(X,N,Y,1,-1,ICON)
      PRINT*,'ICON =',ICON
C
      TMP=0.0D0
      DO I=1,N
      TMP=MAX(ABS(X(I)/DBLE(N)-XX(I)),TMP)
      ENDDO
      PRINT*,' ERROR =',TMP
C
      STOP
      END
```

DM_V2DRCF

2 次元離散型実フーリエ変換 (2, 3, 5 及び 7 の混合基底)
CALL DM_V2DRCF(X, K, N1, N2, ISIN, ISN, ICON)

(1)　機能

2 次元実フーリエ変換の正変換または逆変換を, 混合基底 FFT により行います.

2 次元データ(n_1, n_2)の各次元数の大きさは, 2, 3, 5 及び 7 の巾の積で表すことのできる数です.

a.　2 次元フーリエ変換

$\{x_{j1j2}\}$を入力し, (1.1)で定義する変換を行い, $\{n_1 n_2 \alpha_{k1k2}\}$を求めます.

$$
\begin{aligned}
n_1 n_2 \alpha_{k1k2} &= \sum_{j1=0}^{n1-1} \sum_{j2=0}^{n2-1} x_{j1j2} \omega_{n1}^{-j1k1r} \omega_{n2}^{-j2k2r} \\
&, k_1 = 0,1,...,n_1 - 1 \\
&, k_2 = 0,1,...,n_2 - 1 \\
&, \omega_{n1} = \exp(2\pi i / n_1) \\
&, \omega_{n2} = \exp(2\pi i / n_2) \\
&, r = 1 \text{ または } r = -1
\end{aligned}
\tag{1.1}
$$

b.　2 次元フーリエ逆変換

$\{\alpha_{k1k2}\}$を入力し, (1.2)で定義する変換を行い, $\{x_{j1j2}\}$を求めます.

$$
\begin{aligned}
x_{j1j2} &= \sum_{k1-0}^{n1-1} \sum_{k2=0}^{n2-1} \alpha_{k1k2} . \omega_{n1}^{j1k1r} \omega_{n2}^{j2k2r} \\
&, j_1 = 0,1,...,n_1 - 1 \\
&, j_2 = 0,1,...,n_2 - 1 \\
&, \omega_{n1} = \exp(2\pi i / n_1) \\
&, \omega_{n2} = \exp(2\pi i / n_2) \\
&, r = 1 \text{ または } r = -1
\end{aligned}
\tag{1.2}
$$

(2)　パラメタ

X.....入力／出力. 2 次元実データ.

データを, X(1:N1, 1:N2)に格納します.

実数から複素数への変換(ISN=1)のとき入力に, 複素数から実数への変換(ISN=-1)のとき出力になります.

出力／入力. 変換された複素データの実部, 虚部を次のように格納します.

結果は, X を X(2, K/2, N2)なる配列とみなして, 実部は X(1, 1:N1/2+1, N2)に, 虚部は X(2, 1:N1/2+1, N2)に格納されます.

実数から複素数への変換(ISN = 1)のとき出力に, 複素数から実数への変換(ISN = -1)のとき入力になります.

フーリエ変換された複素数データには共役関係があり, 約半分が格納され

ます.

((3)使用上の注意 a. 注意②参照).

X(K, N2)なる倍精度実数型 2 次元配列.

K.....入力. 配列 X の整合寸法($\geq 2 \times (n_1/2+1)$). 偶数.

整数(INTEGER*4).

N1.....入力. 変換する 2 次元データの 1 次元目の大きさ n_1.

n_1 は 2, 3, 5 及び 7 の巾の積で表すことのできる数.

整数(INTEGER*4).

N2.....入力. 変換する 2 次元データの 2 次元目の大きさ n_2.

n_2 は 2, 3, 5 及び 7 の巾の積で表すことのできる数.

整数(INTEGER*4).

ISIN.....入力. 変換の方向を示します.

1 のとき　$r = 1$

-1 のとき　$r = -1$

整数(INTEGER*4).

ISN.....入力. 変換か逆変換かを指定.

変換　：ISN = 1

逆変換：ISN = -1

整数(INTEGER*4).

ICON.....出力. コンディションコード

表 DM_V2DRCF-1 参照.

表 DM_V2DRCF-1 コンディションコード

コード	意　　味	処理内容
0	エラーなし	ー
30000	K<2×(N1/2+1), K が偶数でない. N1<1, N2<1, ISIN≠1, -1, ISN≠1, -1.	処理を打ち切る.
30008	変換数が 2, 3, 5, 7 を基底としていない.	

(3)　使用上の注意

a.　注意

①　一般的なフーリエ変換の定義

2 次元離散型複素フーリエ変換および, 逆変換は一般的に(3.1), (3.2)で定義されます.

$$\alpha_{k1k2} = \frac{1}{n_1 n_2} \sum_{j1=0}^{n1-1} \sum_{j2=0}^{n2-1} x_{j1j2} \omega_{n1}^{-j1k1} \omega_{n2}^{-j2k2} \quad (3.1)$$
$$, k_1 = 0,1,..,n_1 - 1$$
$$, k_2 = 0,1,...,n_2 - 1$$

$$x_{j1j2} = \sum_{k1=0}^{n1-1}\sum_{k2=0}^{n2-1}\alpha_{k1k2}\omega_{n1}^{j1k1}\omega_{n2}^{j2k2} \quad (3.2)$$
$$, j_1 = 0,1,...,n_1 - 1$$
$$, j_2 = 0,1,...,n_2 - 1$$

ここで, $\omega_{n1}= \exp(2\pi i/n_1)$, $\omega_{n2}= \exp(2\pi i/n_2)$

本サブルーチンでは, (3.1), (3.2)の左辺に対応して$\{n_1 n_2 \alpha_{k1k2}\}$または$\{x_{j1j2}\}$を求めます. したがって, 結果の正規化は必要に応じて行って下さい.

② 2 次元の実データのフーリエ変換結果には次の複素共役(¯ で示します)の関係があります.

$$\alpha_{k1k2} = \overline{\alpha_{n1-k1n2-k2}} \quad (3.3)$$

k_1= 0, ... , n_1/2, k_2= 0, ... , n_2-1 のデータから残りのデータを計算することができます.

b. 使用例

2 次元実 FFT を計算します.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER (N1=2048,N2=2048,K=(N1/2+1)*2)
      REAL*8    X(K,N2), Y(N1,N2)
CC
      DO I=1,N2
      DO J=1,N1
      X(J,I)=J+N2*(I-1)
      Y(J,I)=X(J,I)
      ENDDO
      ENDDO
      ISW=1
      CALL DM_V2DRCF(X,K,N1,N2,1,ISW,ICON)
      PRINT*,' ICON =',ICON
CC
      ISW=-1
      CALL DM_V2DRCF(X,K,N1,N2,1,ISW,ICON)
      PRINT*,' ICON =',ICON
CC
      TMP=0.0D0
      DO I=1,N2
      DO J=1,N1
```

```
      TMP=MAX(DABS(DBLE(X(J,I))/DBLE(N1)/DBLE(N2)
     $             -DBLE(Y(J,I))),TMP)
      ENDDO
      ENDDO
CC
      PRINT*,' ERROR  =',TMP
      STOP
      END
```

(4) 手法概要

本サブルーチンは, スカラ計算機向けの高性能な 1 次元フーリエ変換 DVCFM1(詳細は“SSL II 拡張機能使用手引書 II”参照.)を利用して実現しています.

DM_V3DRCF

3 次元離散型実フーリエ変換 (実数から複素数へ)(2, 3, 5 及び 7 の混合基底)
CALL DM_V3DRCF(X, K, N1, N2, N3, ISIN, ISN, ICON)

(1)　機能

3 次元実フーリエ変換の正変換または逆変換を, 混合基底 FFT により行います.

3 次元データ(n_1, n_2, n_3)の各次元の大きさは, 2, 3, 5 及び 7 の巾の積で表すことのできる数です.

a.　3 次元フーリエ変換

$\{x_{j1j2j3}\}$を入力し, (1.1)で定義する変換を行い, $\{n_1n_2n_3\alpha_{k1k2k3}\}$を求めます.

$$
\begin{aligned}
n_1n_2n_3\alpha_{k1k2k3} = &\sum_{j1=0}^{n1-1}\sum_{j2=0}^{n2-1}\sum_{j3=0}^{n3-1} x_{j1j2j3}\omega_{n1}^{-j1k1r}\omega_{n2}^{-j2k2r}\omega_{n3}^{-j3k3r} \\
&, k_1 = 0,1,...,n_1-1 \\
&, k_2 = 0,1,...,n_2-1 \\
&, k_3 = 0,1,...,n_3-1 \\
&, \omega_{n1} = \exp(2\pi i/n_1) \\
&, \omega_{n2} = \exp(2\pi i/n_2) \\
&, \omega_{n3} = \exp(2\pi i/n_3) \\
&, r = 1 \text{ または } r = -1
\end{aligned}
\tag{1.1}
$$

b.　3 次元フーリエ逆変換

$\{\alpha_{k1k2k3}\}$を入力し, (1.2)で定義する変換を行い, $\{x_{j1j2j3}\}$を求めます.

$$
\begin{aligned}
x_{j1j2j3} = &\sum_{k1=0}^{n1-1}\sum_{k2=0}^{n2-1}\sum_{k3=0}^{n3-1} \alpha_{k1k2k3}\omega_{n1}^{j1k1r}\omega_{n2}^{j2k2r}\omega_{n3}^{j3k3r} \\
&, j_1 = 0,1,...,n_1-1 \\
&, j_2 = 0,1,...,n_2-1 \\
&, j_3 = 0,1,...,n_3-1 \\
&, \omega_{n1} = \exp(2\pi i/n_1) \\
&, \omega_{n2} = \exp(2\pi i/n_2) \\
&, \omega_{n3} = \exp(2\pi i/n_3) \\
&, r = 1 \text{ または } r = -1
\end{aligned}
\tag{1.2}
$$

(2)　パラメタ

X.....入力／出力. 3 次元実データ.

データは, X(1:N1, 1:N2, 1:N3)に格納します.

実数から複素数への変換(ISN=1)のとき入力, 複素数から実数への変換(ISN=-1)のとき出力となります.

出力／入力. 変換された複素データの実部および虚部.

実数から複素数への変換(ISN=1)のとき出力, 複素数から実数への変換

(ISN=-1)のとき入力となります.
実数データからフーリエ変換された複素数データは共役関係があり, 約半分が格納されます.
((3)使用上の注意 a. 注意②参照)
結果は, X を X(2, K/2, N2, N3)なる配列とみなして, 実部が X(1, 1:N1/2+1, 1:N2, 1:N3)に, 虚部が X(2, 1:N1/2+1, 1:N2, N3)に格納されます.
X(K, N2, N3)なる倍精度実数型 3 次元配列.

K.....入力. 入力データ配列 X の 1 次元目の大きさ($\geq 2 \times (n_1/2+1)$). 偶数.
整数(INTEGER*4).

N1.....入力. 変換する実データの 1 次元目の大きさ n_1.
n_1 は, 2, 3, 5 及び 7 の巾の積で表すことのできる数.
整数(INTEGER*4).

N2.....入力. 変換する実データの 2 次元目の大きさ n_2.
n_2 は, 2, 3, 5 及び 7 の巾の積で表すことのできる数.
整数(INTEGER*4).

N3.....入力. 変換する実データの 3 次元目の大きさ n_3.
n_3 は, 2, 3, 5 及び 7 の巾の積で表すことのできる数.
整数(INTEGER*4).

ISIN.....入力. 変換の方向を表します.
1 のとき　$r = 1$
-1 のとき　$r = -1$
整変数(INTEGER*4)であること.

ISN.....入力. 変換か逆変換かを指定.
変換　：ISN = 1
逆変換：ISN = -1
整変数(INTEGER*4)であること.

ICON.....出力. コンディションコード.
表 DM_V3DRCF-1 参照.

表 DM_V3DRCF-1 コンディションコード

コード	意　　味	処理内容
0	エラーなし	ー
30000	K<2 × (N1/2+1), K が偶数でない. N1<1, N2<1, N3<1, ISIN≠1, -1, ISN≠1, -1.	処理を打ち切る.
30008	変換数が 2, 3, 5, 7 を基底としていない.	

(3)　使用上の注意

a.　注意

①　一般的なフーリエ変換の定義

3 次元離散型複素フーリエ変換および, 逆変換は一般的に(3.1), (3.2)で定義されます.

$$\alpha_{k1k2k3} = \frac{1}{n_1 n_2 n_3} \sum_{j1=0}^{n1-1} \sum_{j2=0}^{n2-1} \sum_{j3=0}^{n3-1} x_{j1j2j3} \omega_{n1}^{-j1k1} \omega_{n2}^{-j2k2} \omega_{n3}^{-j3k3}$$
$$, k_1 = 0,1,..., n_1 - 1$$
$$, k_2 = 0,1,..., n_2 - 1$$
$$, k_3 = 0,1,..., n_3 - 1 \qquad (3.1)$$

$$x_{j1j2j3} = \sum_{k1=0}^{n1-1} \sum_{k2=0}^{n2-1} \sum_{k3=0}^{n3-1} \alpha_{k1k2k3} \omega_{n1}^{j1k1} \omega_{n2}^{j2k2} \omega_{n3}^{j3k3}$$
$$, j_1 = 0,1,.., n_1 - 1$$
$$, j_2 = 0,1,..., n_2 - 1$$
$$, j_3 = 0,1,..., n_3 - 1 \qquad (3.2)$$

ここで, $\omega_{n1} = \exp(2\pi i / n_1)$, $\omega_{n2} = \exp(2\pi i / n_2)$, $\omega_{n3} = \exp(2\pi i / n_3)$

本サブルーチンでは, (3.1), (3.2)の左辺に対応して$\{n_1 n_2 n_3 \alpha_{k1k2k3}\}$または$\{x_{j1j2j3}\}$を求めます. したがって, 結果の正規化は必要に応じて行って下さい.

② 3 次元実データのフーリエ変換の結果には, 次の複素共役(¯で示します)の関係があります.

$$\alpha_{k1k2k3} = \overline{\alpha_{n1-k1n2-k2n3-k3}} \qquad (3.3)$$

つまり, k_1=0, ... , n_1/2, k_2=0, ... , n_2-1, k_3=0, ... , n_3-1
のデータから残りのデータは求めることができます.

b. 使用例

3 次元実 FFT を計算します.
(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
C     **EXAMPLE**
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER (N1=128,N2=128,N3=128,K=(N1/2+1)*2)
      DIMENSION YY(K,N2,N3), YR(K,N2,N3)
C
      DO I3=1,N3
      DO I2=1,N2
      DO I1=1,N1
      YY(I1,I2,I3)=DBLE(I1+N1*(I2-1)+N1*N2*(I3-1))
      YR(I1,I2,I3)=YY(I1,I2,I3)
      ENDDO
      ENDDO
      ENDDO
C
      ISW=1
      CALL DM_V3DRCF(YY,K,N1,N2,N3,1,ISW,ICON)
```

```
      PRINT*,'ICON =',ICON
C
      ISW=-1
      CALL DM_V3DRCF(YY,K,N1,N2,N3,1,ISW,ICON)
      PRINT*,'ICON =',ICON
C
      TMP=0.0D0
      DO I3=1,N3
      DO I2=1,N2
      DO I1=1,N1
      TMP=MAX(DABS(YY(I1,I2,I3)/DBLE(N1)/DBLE(N2)/DBLE(N3)
     $             -YR(I1,I2,I3)),TMP)
      ENDDO
      ENDDO
      ENDDO
      PRINT*,' ERROR =',TMP
C
      STOP
      END
```

(4)　手法概要

本サブルーチンは, スカラ計算機向けの高性能な 1 次元フーリエ変換 DVCFM1(詳細は“SSL II 拡張機能使用手引書 II”参照.)を利用して実現しています.

DM_V3DRCF2

3 次元離散型実フーリエ変換 (実数から複素数へ)(2, 3, 5 及び 7 の混合基底)
CALL DM_V3DRCF2(X, K1, K2, N1, N2, N3, ISIN, ISN, ICON)

(1)　機能

3 次元実フーリエ変換の正変換または逆変換を, 混合基底 FFT により行います.

3 次元データ(n_1, n_2, n_3)の各次元の大きさは, 2, 3, 5 及び 7 の巾の積で表すことのできる数です.

a.　3 次元フーリエ変換

$\{x_{j1j2j3}\}$を入力し, (1.1)で定義する変換を行い, $\{n_1n_2n_3\alpha_{k1k2k3}\}$を求めます.

$$
\begin{aligned}
n_1n_2n_3\alpha_{k1k2k3} = &\sum_{j1=0}^{n1-1}\sum_{j2=0}^{n2-1}\sum_{j3=0}^{n3-1} x_{j1j2j3}\omega_{n1}^{-j1k1r}\omega_{n2}^{-j2k2r}\omega_{n3}^{-j3k3r} \\
&, k_1 = 0,1,...,n_1-1 \\
&, k_2 = 0,1,...,n_2-1 \\
&, k_3 = 0,1,...,n_3-1 \\
&, \omega_{n1} = \exp(2\pi i/n_1) \\
&, \omega_{n2} = \exp(2\pi i/n_2) \\
&, \omega_{n3} = \exp(2\pi i/n_3) \\
&, r = 1 \text{ または } r = -1
\end{aligned}
\tag{1.1}
$$

b.　3 次元フーリエ逆変換

$\{\alpha_{k1k2k3}\}$を入力し, (1.2)で定義する変換を行い, $\{x_{j1j2j3}\}$を求めます.

$$
\begin{aligned}
x_{j1j2j3} = &\sum_{k1=0}^{n1-1}\sum_{k2=0}^{n2-1}\sum_{k3=0}^{n3-1} \alpha_{k1k2k3}\omega_{n1}^{j1k1r}\omega_{n2}^{j2k2r}\omega_{n3}^{j3k3r} \\
&, j_1 = 0,1,...,n_1-1 \\
&, j_2 = 0,1,...,n_2-1 \\
&, j_3 = 0,1,...,n_3-1 \\
&, \omega_{n1} = \exp(2\pi i/n_1) \\
&, \omega_{n2} = \exp(2\pi i/n_2) \\
&, \omega_{n3} = \exp(2\pi i/n_3) \\
&, r = 1 \text{ または } r = -1
\end{aligned}
\tag{1.2}
$$

(2)　パラメタ

X.....入力／出力. 3 次元実データ.

データは, X(1:N1, 1:N2, 1:N3)に格納します.

実数から複素数への変換(ISN=1)のとき入力, 複素数から実数への変換(ISN=-1)のとき出力となります.

出力／入力. 変換された複素データの実部および虚部.

実数から複素数への変換(ISN=1)のとき出力, 複素数から実数への変換

(ISN=-1)のとき入力となります.
実数データからフーリエ変換された複素数データは共役関係があり, 約半分が格納されます.
((3)使用上の注意 a. 注意②参照)
結果は, X を X(2, K1/2, K2, N3)なる配列とみなして, 実部が X(1, 1:N1/2+1, 1:N2, 1:N3)に, 虚部が X(2, 1:N1/2+1, 1:N2, N3)に格納されます.
X(K1, K2, N3)なる倍精度実数型 3 次元配列.

K1.....入力. 入力データ配列 X の 1 次元目の大きさ($\geq 2 \times (n_1/2+1)$). 偶数.
整数(INTEGER*4).

K2.....入力. 入力データ配列 X の 2 次元目の大きさ($\geq n_2$)

N1.....入力. 変換する実データの 1 次元目の大きさ n_1.
n_1 は, 2, 3, 5 及び 7 の巾の積で表すことのできる数.
整数(INTEGER*4).

N2.....入力. 変換する実データの 2 次元目の大きさ n_2.
n_2 は, 2, 3, 5 及び 7 の巾の積で表すことのできる数.
整数(INTEGER*4).

N3.....入力. 変換する実データの 3 次元目の大きさ n_3.
n_3 は, 2, 3, 5 及び 7 の巾の積で表すことのできる数.
整数(INTEGER*4).

ISIN.....入力. 変換の方向を表します.
1 のとき　$r = 1$
-1 のとき　$r = -1$
整変数(INTEGER*4)であること.

ISN.....入力. 変換か逆変換かを指定.
変換　：ISN = 1
逆変換：ISN = -1
整変数(INTEGER*4)であること.

ICON.....出力. コンディションコード.
表 DM_V3DRCF2-1 参照.

表 DM_V3DRCF2-1 コンディションコード

コード	意　味	処理内容
0	エラーなし	ー
30000	K1<2 × (N1/2+1), K1 が偶数でない. K2<N2, N1<1, N2<1, N3<1, ISIN≠1, -1, ISN≠1, -1.	処理を打ち切る.
30008	変換数が 2, 3, 5, 7 を基底としていない.	

(3)　使用上の注意

a.　注意

①　一般的なフーリエ変換の定義

3 次元離散型複素フーリエ変換および, 逆変換は一般的に(3.1), (3.2)で定義されます.

$$\alpha_{k1k2k3} = \frac{1}{n_1 n_2 n_3} \sum_{j1=0}^{n1-1} \sum_{j2=0}^{n2-1} \sum_{j3=0}^{n3-1} x_{j1j2j3} \omega_{n1}^{-j1k1} \omega_{n2}^{-j2k2} \omega_{n3}^{-j3k3}$$
$$, k_1 = 0,1,\ldots, n_1 - 1$$
$$, k_2 = 0,1,\ldots, n_2 - 1$$
$$, k_3 = 0,1,\ldots, n_3 - 1 \qquad (3.1)$$

$$x_{j1j2j3} = \sum_{k1=0}^{n1-1} \sum_{k2=0}^{n2-1} \sum_{k3=0}^{n3-1} \alpha_{k1k2k3} \omega_{n1}^{j1k1} \omega_{n2}^{j2k2} \omega_{n3}^{j3k3}$$
$$, j_1 = 0,1,\ldots, n_1 - 1$$
$$, j_2 = 0,1,\ldots, n_2 - 1$$
$$, j_3 = 0,1,\ldots, n_3 - 1 \qquad (3.2)$$

ここで, $\omega_{n1} = \exp(2\pi i / n_1)$, $\omega_{n2} = \exp(2\pi i / n_2)$, $\omega_{n3} = \exp(2\pi i / n_3)$

本サブルーチンでは, (3.1), (3.2)の左辺に対応して$\{n_1 n_2 n_3 \alpha_{k1k2k3}\}$または$\{x_{j1j2j3}\}$を求めます. したがって, 結果の正規化は必要に応じて行って下さい.

②　3 次元実データのフーリエ変換の結果には, 次の複素共役(¯で示します)の関係があります.

$$\alpha_{k1k2k3} = \overline{\alpha_{n1-k1n2-k2n3-k3}} \qquad (3.3)$$

つまり, k_1=0, ... , n_1/2, k_2=0, ... , n_2-1, k_3=0, ... , n_3-1
のデータから残りのデータは求めることができます.

b.　使用例

3 次元実 FFT を計算します.
(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
c     **example**
      implicit real*8(a-h,o-z)
      parameter(n1=128,n2=128,n3=128,k1=(n1/2+1)*2,k2=n2+1)
      dimension yy(k1,k2,n3),yr(k1,k2,n3)
c
      do i3=1,n3
      do i2=1,n2
      do i1=1,n1
      yy(i1,i2,i3)=dble(i1+n1*(i2-1)+n1*n2*(i3-1))
      yr(i1,i2,i3)=yy(i1,i2,i3)
      enddo
```

```
      enddo
      enddo
c
      isw=1
      call dm_v3drcf2(yy,k1,k2,n1,n2,n3,1,isw,icon)
      print*,'icon =',icon
c
      isw=-1
      call dm_v3drcf2(yy,k1,k2,n1,n2,n3,1,isw,icon)
      print*,'icon =',icon
c
      tmp=0.0d0
      do i3=1,n3
      do i2=1,n2
      do i1=1,n1
      tmp=max(dabs(yy(i1,i2,i3)
     $              /dble(n1)/dble(n2)/dble(n3)
     $             -yr(i1,i2,i3)),tmp)
      enddo
      enddo
      enddo
      print*,' error =',tmp
c
      stop
      end
```

(4) 手法概要

本サブルーチンは, スカラ計算機向けの高性能な 1 次元フーリエ変換 DVCFM1(詳細は“SSL II 拡張機能使用手引書 II”参照.)を利用して実現しています.

DM_V3DCPF

3 次元素因子離散型複素フーリエ変換
CALL DM_V3DCPF (X, K1, K2, N1, N2, N3, ISN, ICON)

(1)　機能

3 次元複素フーリエ変換の正変換または逆変換を行います.

3 次元データ(n_1, n_2, n_3)の各次元の大きさは, 次の条件を満たす数です.

ー　次の数の中で互いに素な因子の積で表せること.

因子 p($p \in \{2, 3, 4, 5, 7, 8, 9, 16, 25\}$)

a.　3 次元フーリエ変換

$\{x_{j1j2j3}\}$を入力し, (1.1)で定義する変換を行い, $\{n_1n_2n_3\alpha_{k1k2k3}\}$を求めます.

$$n_1n_2n_3\alpha_{k1k2k3} = \sum_{j1=0}^{n1-1}\sum_{j2=0}^{n2-1}\sum_{j3=0}^{n3-1} x_{j1j2j3}\omega_{n1}^{-j1k1}\omega_{n2}^{-j2k2}\omega_{n3}^{-j3k3} \qquad (1.1)$$

$$,k_1 = 0,1,...,n_1-1$$
$$,k_2 = 0,1,...,n_2-1$$
$$,k_3 = 0,1,...,n_3-1$$
$$,\omega_{n1} = \exp(2\pi i/n_1)$$
$$,\omega_{n2} = \exp(2\pi i/n_2)$$
$$,\omega_{n3} = \exp(2\pi i/n_3)$$

b.　3 次元フーリエ逆変換

$\{\alpha_{k1k2k3}\}$を入力し, (1.2)で定義する変換を行い, $\{x_{j1j2j3}\}$を求めます.

$$x_{j1j2j3} = \sum_{k1=0}^{n1-1}\sum_{k2=0}^{n2-1}\sum_{k3=0}^{n3-1}\alpha_{k1k2k3}\cdot\omega_{n1}^{j1k1}\omega_{n2}^{j2k2}\omega_{n3}^{j3k3}$$

$$,j_1 = 0,1,...,n_1-1$$
$$,j_2 = 0,1,...,n_2-1$$
$$,j_3 = 0,1,...,n_3-1 \qquad (1.2)$$
$$,\omega_{n1} = \exp(2\pi i/n_1)$$
$$,\omega_{n2} = \exp(2\pi i/n_2)$$
$$,\omega_{n3} = \exp(2\pi i/n_3)$$

(2)　パラメタ

X.....入力. 複素数データ.
　　データは, X(1:N1, 1:N2, 1:N3)に格納します.
　　出力. 変換された複素数データ.
　　結果は, X(1:N1, 1:N2, 1:N3)に格納されます.
　　X(K1, K2, N3)なる倍精度複素数型 3 次元配列.

K1.....入力. 入力データ配列 X の 1 次元目の大きさ. (≥ N1)

K2.....入力. 入力データ配列 X の 2 次元目の大きさ. (≥ N2)
N1.....入力. 変換する 3 次元データの 1 次元目の大きさ n_1.
N2.....入力. 変換する 3 次元データの 2 次元目の大きさ n_2.
N3.....入力. 変換する 3 次元データの 3 次元目の大きさ n_3.
ISN.....入力. 変換か逆変換かを指定.
変換　：ISN = 1
逆変換：ISN = -1
ICON.....出力. コンディションコード.
表 DM_V3DCPF-1 参照.

表 DM_V3DCPF-1 コンディションコード

コード	意　　味	処理内容
0	エラーなし.	ー
20000	n_1, n_2 または n_3 が 2, 3, 4, 5, 7, 8, 9, 16, 25 の中の互いに素な因子に分解できなかった.	処理を打ち切る.
30000	n_1, n_2, n_3 が 0 または 0 以下であった. K1<N1, K2<N2 または ISN の値が適切でない.	

(3)　使用上の注意

a.　注意

①　一般的なフーリエ変換の定義

3 次元離散型複素フーリエ変換および, 逆変換は一般的に(3.1), (3.2)で定義されます.

$$\alpha_{k1k2k3} = \frac{1}{n_1 n_2 n_3} \sum_{j1=0}^{n1-1} \sum_{j2=0}^{n2-1} \sum_{j3=0}^{n3-1} x_{j1j2j3} \omega_{n1}^{-j1k1} \omega_{n2}^{-j2k2} \omega_{n3}^{-j3k3} \quad (3.1)$$
$$, k_1 = 0,1,...,n_1 - 1$$
$$, k_2 = 0,1,...,n_2 - 1$$
$$, k_3 = 0,1,...,n_3 - 1$$

$$x_{j1j2j3} = \sum_{k1=0}^{n1-1} \sum_{k2=0}^{n2-1} \sum_{k3=0}^{n3-1} \alpha_{k1k2k3} \omega_{n1}^{j1k1} \omega_{n2}^{j2k2} \omega_{n3}^{j3k3} \quad (3.2)$$
$$, j_1 = 0,1,...,n_1 - 1$$
$$, j_2 = 0,1,...,n_2 - 1$$
$$, j_3 = 0,1,...,n_3 - 1$$

ここで, $\omega_{n1} = \exp(2\pi i / n_1)$, $\omega_{n2} = \exp(2\pi i / n_2)$, $\omega_{n3} = \exp(2\pi i / n_3)$

本サブルーチンでは, (3.1), (3.2)の左辺に対応して$\{n_1 n_2 n_3 \alpha_{k1k2k3}\}$または$\{x_{j1j2j3}\}$を求めます. したがって, 結果の正規化は必要に応じて行って下さい.

b.　使用例

3 次元 FFT を計算します.

(並列実行を行うスレッド数は環境変数(OMP_NUM_THREADS)などで指定できます. 例えば, 4 プロセッサーのシステムで 4 つのスレッドで並列実行するときは, OMP_NUM_THREADS を 4 に設定して実行します.)

```
c     **example**
      implicit real*8 (a-h,o-z)
      parameter (n1=40,n2=240,n3=90)
      parameter (k1=n1,k2=n2)
      complex*16 x(k1,k2,n3)
      integer isn
*
*     set up the input data arrays
*
!$omp parallel do default(private) shared(x)
      do k=1,n3
      do i=1,n2
      do j=1,n1
      x(j,i,k)=dcmplx(float(j)+float(n1)*(i-1),0.0)
      enddo
      enddo
      enddo
!$omp end parallel do

*
*     do the forward transform
*
      isn=1
      call dm_v3dcpf(x,k1,k2,n1,n2,n3,isn,icon)
      if(icon.ne.0) then
        write(*,*) 'error occurred : ',icon
      endif
*
*    do the reverse transform
*
      isn=-1
      call dm_v3dcpf(x,k1,k2,n1,n2,n3,isn,icon)
      if(icon.ne.0) then
        write(*,*) 'error occurred : ',icon
      endif
*
```

```
*    find the error after the forward and
*    inverse transform.
*
      error=0
!$omp parallel do default(private) shared(x)
!$omp+         reduction(max:error)

      do k=1,n3
      do i=1,n2
      do j=1,n1
      error=max(abs(dble(x(j,i,k))/(n3*n2*n1)-
     &      (float(j)+float(n1)*(i-1))),error)
      error=max(abs(dimag(x(j,i,k))/(n3*n2*n1)),
     &                           error)
      enddo
      enddo
      enddo
!$omp end parallel do

      write(*,*) 'error ', error
      stop
      end
```

付録1　参考文献一覧表

[1] P.AMESTOY, M.DAYDE and I.DUFF
Use of computational kernels in the solution of full and sparse linear equations, M.COSNARD, Y.ROBERT, Q.QUINTON and M.RAYNAL, PARALLEL & DISTRIBUTED ALGORITHMS, North-Holland, 1989, pp.13-19.

[2] P.R.AMESTOY and C.PUGLISH
AN UNSYMMETRIZED MULTIFRONTAL LU FACTORIZATION, SIAM J. MATRIX ANAL. APPL. Vol. 24, No. 2, pp. 553-569, 2002

[3] A.A.Anda and H.Park
Fast Plane Rotations with Dynamic Scaling, to appear in SIAM J. Matrix Analysis and Applications, 1994.

[4] S.L.Anderson
Random number generators on vector supercomputers and other advanced architectures, SIAM Rev. 32 (1990), 221-251.

[5] C.Ashcraft
The distributed solution of linear systems using the torus wrap data mapping, Tech. Report ECA-TR-147, Boeing Computer Services, October 1990.

[6] O.Axelsson and M.Neytcheva
Algebraic multilevel iteration method for Stieltjes matrices. Num. Lin. Alg. Appl., 1:213-236, 1994.

[7] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, editors.
"Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide." SIAM, Philadelphia, 2000.

[8] Ă.Björck
Solving linear least squares problems by Gram-Schmidt orthogonalisation, BIT, 7:1-21,1967.

[9] R.P.Brent
Uniform random number generators for supercomputers, Proc. Fifth Australian Supercomputer Conference, Melbourne, Dec. 1992, 95-104.

[10] R.P.Brent
Uniform random number generators for vector and parallel computers, Report TR-CS-92-02, Computer Sciences Laboratory, Australian National University, Canberra, March 1992.

[11] R.P.Brent
Fast normal random number generators on vector processors, Technical Report TR-CS-93-04, Computer Sciences Laboratory, Australian National University, Canberra, March 1993.

[12] R.P.Brent
A Fast Vectorised Implementation of Wallace's Normal Random Number Generator, Technical Report, Computer Sciences Laboratory, Australian National University, to appear.

[13] R.Burkard, M.Dell'Amico and S.Martello
Assignment Problems, SIAM Philadelphia, 2009

[14] J.Choi, J.Dongarra, R.Pozo, and D.Walker
ScaLAPACK : A scalable linear algebra library for distributed memory concurrent computers., Technical Report 53, LAPACK Working Note, 1993.

[15] A.Cleary
A comparison of algorithms for Cholesky factorization on a massively parallel MIMD computer, Parallel Processing for Scientific Computing, 1991.

[16] A.Cleary
A Scalable Algorithm for Triangular System Solution Using the Torus Wrap Mapping, ANU-CMA Tech Report, series 1994.

[17] T.H.CORMEN, C.E.LEISERSON, R.L.RIVEST and C.STEIN
INTRODUCTION TO ALGORITHMS, SECOND EDITION, The MIT Press, 2001

[18] J.K.Cullum and R.A.Willoughby
"Lanczos algorithm for large symmetric eigenvalue computations", Birkhauser, 1985.

[19] T.Davis
Direct Methods for Sparse Linear Systems, SIAM 2006.

[20] J.Demmel and W.Kahan
Accurate singular values of bidiagonal matrices, SISSC 11, 873-912, 1990.

[21] J.J.Dongarra and R.A.Van de Geijn
Reduction to condensed form for the eigenvalue problem on distributed memory architectures, Parallel Computing, 18, pp.973-982, 1992.

[22] I.S.DUFF, A.M.ERISMAN and J.K.REID
Direct Methods for Sparse Matrices, OXFORD SCIENCE PUBLICATIONS, 1986

[23] I.S.DUFF and J.KOSTER
ON ALGORITHMS FOR PERMUTING LARGE ENTRIES TO THE DIAGONAL OF A SPARSE MATRIX, SIAM J. MATRIX ANAL. APPL. Vol. 22, No. 4, pp. 973-996, 2001

[24] A.M.Ferrenberg, D.P.Landau and Y.J.Wong
Monte Carlo simulations: Hidden errors from good" random number generators, Phys. Rev. Lett. 69 (1992), 3382-3384.

[25] G.Fox
Square matrix decomposition ― Symmetric, local, scattered, CalTech Publication Hm-97, California Institute of Technology, Pasadena, CA, 1985.

[26] R.Freund
"A transpose-free quasi-minimal residual algorithm for nonhermitian linear systems", SIAM J.Sci.Comput. 14, 1993, pp.470-482.

[27] R.Freund and N.Nachtigal
"QMR: a quasi minimal resudual method for non-Hermitian linear systems", Numer. Math. 60, 1991, pp.315-339.

[28] K.A.Gallivan, R.J.Plemmons, and A.H.Sameh
Parallel Algorithms for Dense Linear Algebra Computations, SIAM Review, 1990.

[29] Martin B. van Gijzen and Peter Sonneveld
"An elegant IDR(s) variant that efficiently exploits bi-orthogonality properties",
Delft university of technology, Report 08-21, 2008.

[30] G.H.Golub, C.F.van Loan
Matrix Computations Second Edition, The Johns Hopkins University Press, 1989.

[31] Marcus J. Grote and Thomas Huckle
"Parallel preconditioning with sparse approximate inverse",
SIAM J. Sci. Comput., Vol.18, No.3, pp838-853, May 1997.

[32] M.H.Gutknecht
Variants of BiCGStab for matrices with complex spectrum,IPS Research report No. 91-14, 1991.

[33] E. Hairer, S.P.Norsett, and G. Wanner
"Solving Ordinary Differential Equations I: Nonstiff Problems." Second Revised Edition, Springer, 2000.

[34] E. Hairer, and G. Wanner
“Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems.” Second Revised Edition, Springer, 2002.

[35] E.ハイラー/S.P.ネルセット/G.ヴァンナー著, 三井斌友 監訳
常微分方程式の数値解法 I : 基礎編 , Springer , 2007

[36] E.ハイラー/G.ヴァンナー著, 三井斌友 監訳
常微分方程式の数値解法 II : 発展編 , Springer , 2008

[37] Markus Hegland
An implementation of multiple and multi-variate Fourier transforms on vector processors, submitted to SIAM J.Sci. Comput., 1992.

[38] Markus Hegland
Block Algorithms for FFTs on Vector and Parallel Computers. PARCO 93, Grenoble, 1993.

[39] Markus Hegland
On the parallel solution of tridiagonal systems by wrap-around partitioning and incomplete LU factorization, Numer. Math. 59, 453-472, 1991.

[40] B.Hendrickson and D.Womble
The torus-wrap mapping for dense matrix calculations on massively parallel computers, SAND Report SAND 92-0792, Sandia National Laboratories, Albuquerque, NM, 1992.

[41] J.R.Heringa, H.W.J.Blöte and A.Compagner
New primitive trinomials of Mersenne-exponent degrees for random-number generation, International J. of Modern Physics C 3 (1992), 561-564.

[42] F.James
A review of pseudorandom number generators, Computer Physics Communications 60 (1990), 329-344.

[43] G.KARYPIS AND V.KUMAR
A fast and high quality multilevel scheme for partitioning irregular graphs, SIAM J. Sci. Comput., 20 pp.359-392, 1998

[44] G.KARYPIS AND V.KUMAR
METIS
A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices
Version 4.0
University of Minnesota, Department of Computer Science / Army HPC Research Center
Minneapolis, MN 55455
September 20, 1998

[45] D.Kincaid, T.Oppe
ITPACK on supercomputers, Numerical methods, Lecture Notes in Mathematics 1005 (1982).

[46] D.E.Knuth
The Art of Computer Programming, Volume 2: Seminumerical Algorithms (second edition). Addison-Wesley, Menlo Park, 1981, Sec. 3.4.1, Algorithm P.

[47] Z.Leyk
Modified generalized conjugate residuals for nonsymmetric systems of linear equations, in Proceedings of the 6th Biennial Conference on Computational Techniques and Applications: CTAC93, D.Stewart, H.Gardner and D.Singleton, eds., World Scientific, 1994, pp.338-344. Also published as CMA Research Report CMA-MR33-93, Australian National University, 1993.

[48] X.S.Li AND J.W.DEMMEL
A scalable sparse direct solver using static pivoting, in Proceedings of the Ninth SIAM Conference on Parallel Processing for Scientific Computing, San Antonio, Texas, 1999, CD-ROM, SIAM, Philadelphia, PA, 1999

[49] Charles Van Loan
Computational Frameworks for the Fast Fourier Transform, SIAM, 1992.

[50] F.T.Luk
Computing the Singular-Value Decomposition on the ILIAC IV, ACM Trans. Math. Softw., 6, 1980, pp.259-273.

[51] F.T.Luk and H.Park
On Parallel Jacobi Orderings, SIAM J.Sci. Comput., 10, 1989, pp.18-26.

[52] N.K.Madsen, G.h.Rodrigue, and J.I.Karush
“Matrix multiplication by diagonals on a vector/parallel processor”, Information Processing Letters, vol.5, 1976, pp.41-45.

[53] G.Marsaglia
A current view of random number genetators, Computer Science and Statistics: The Interface (edited by L.Billard), Elsevier Science Publishers B.V. (North-Holland), 1985, 3-10.

[54] M.Nakanishi, H.Ina, K.Miura
A high performance linear equation solver on the VPP500 parallel supercomputer, Proceedings of Supercomputing’94, Washington D.C., Nov. 1994.

[55] 中西誠, 三上次郎
ブロッキングLU分解法のVP2000シリーズ向けチューニングについて情報処理学会第42回全国大会, 1991

[56] 中西誠, 三上次郎
線型方程式の高速化技術, 情報処理学会研究報告, 91-OS-54, 情処研報 Vol.92, No.22, pp.33-40, 1992

[57] M.OLSCHOWKA and A.NEUMAIER
A new pivoting strategy for Gaussian elimination, Linear Algebra Appl., 240(1996), pp.131-151

[58] T.Oppe, W.Joubert and D.Kincaid
An overview of NSPCG: a nonsymmetric preconditioned conjugate gradient package, Computer Physics communications 53 p283 (1989).

[59] T.C.Oppe and D.R.Kincaid
“Are there iterative BLAS?”, Int. J. Sci. Comput. Modeling (to appear or has appeared).

[60] M.R.Osborne
Solving least squares problems on parallel vector processors, Area 4 working notes no. 17, 1994.

[61] M.R.Osborne
Computing the eigenvalues of tridiagonal matrices on parallel vector processors, Mathematics Research Report No. MRR 044-94, Australian National University, 1994.

[62] J.R.Rice and R.F.Boisvert
Solving Elliptic Problems Using Ellpack, Springer-Verlag, New York, 1985.

[63] D. Ruiz
A scaling algorithm to equilibrate both rows and columns norms in matrices, Tech. rep. RAL-TR-2001-034, Rutherford Appleton Laboratory, Chilton, U.K., 2001

[64] Y.Saad
ILUT: A dual threshold incomplete LU factorization.Research Report UMSI 92/38, University of Minnesota, Supercomputer Institute, 1200 Washington Avenue South, Minneapolis, Minnesota 55415, USA, 1992.

[65] Y.Saad
ILUM: A multi-elimination ILU preconditioner for general sparse 591 matrices. SIAM J. Sci. Comput., 17:830-847, 1996.

[66] Y.Saad
"Iterative methods for sparse linear systems, second edition",
Univ.Minnesota,SIAM, 2003

[67] Y.Saad and M.H.Schultz
“GMRES : a generalized minimal residual algorithm for solving nonsymmetric linear systems”, SIAM J. Sci. Stat. Comput. 7, 1986, p.856-869.

[68] O.Schenk , K.Gärtner
Solving unsymmetric sparse systems of linear equations with PARDISO, Future Genration Computer Systems 20(2004)475-487

[69] J.A.SCOTT
Scaling and Pivoting in an Out-of-Core Sparese Direct Solver
ACM Transactions on Mathematical Software, Vol. 37, No. 2, Article 19, April 2010

[70] 島崎眞昭
スーパーコンピュータとプログラミング, 共立出版株式会社, 1989

[71] H.D.Simon
Bisection is not optimal on vector processors, SISSC 10, 205-209, 1989.
[72] G. Sleijpen, D. Fokkema
BCG for linear equations involving unsymmetric matrices with complex spectrum, Electronic Transactions on Numerical Analysis, 1 p11 1993
[73] Gerard L.G. Sleijpen and Martin B. van Gijzen
"Exploiting BICGSTAB(l) Strategies to Induce Dimension Reduction",
Delft university of technology, Report 09-02, 2009.
[74] Tomohiro Sogabe, Shao-Liang Zhang
"A COCR method for solving complex symmetric linear systems",
Journal of Computational and SIAM Applied Mathematics, 199(2007)297-303.
[75] J.C. Strikwerda
Finite Difference Schemes and Partial Differential Equations. Wadsworth and Brooks/Cole, Pacific Grove, 1989.
[76] Paul N.Swarztrauber
Multiprocessor FFTs. Parallel Comput. 5, 197-210, 1987.
[77] H.A.Van Der Vorst
“BCG: A fast and smoothly converging variant of BI-CG for the solution of non-symmetric linear systems”, SIAM J. Sci. Statist. Comput., 13 p631 1992
[78] C.S.Wallace
“Fast Pseudo-Random Generators for Normal and Exponential Variates”,
ACM Trans. on Mathematical Software 22 (1996), 119-127.
[79] R.Weiss
Parameter-Free Iterative Linear Solvers. Mathematical Research, vol. 97. Akademie Verlag, Berlin, 1996.
[80] J.H.Wilkinson
The Algebraic Eigenvalue Problem, O.U.P., 1965.
[81] B.B.Zhou and R.P.Brent
A Parallel Ordering Algorithm for Efficient One-Sided Jacobi SVD Computations, to appear in Proc. Sixty IASTED-ISMM International Conference on Parallel and Distributed Computing Systems, 1994.
[82] K. Miura
Full Polynomial Multiple Recursive Generator(MRG) Revisited, MCQMC 2006, Ulm, Germany
[83] Kenta Hongo, Ryo Maezono, and Kenichi Miura
Random Number Generators Tested on Quantum Monte Carlo Simulations, Journal of Computational Chemistry, 31, 2186-2194, 2010
[84] P. L'Ecuyer and R. Simard
TestU01: A C Library for Empirical Testing of Random Number Generators, ACM Transactions on Mathematical Software, Vol. 33, article 22, 2007.

付録2　著作者氏名と著作物の索引

機能を実現する上で全面的または部分的に利用または改造した SSL II/VPP のコードまたはアルゴリズムの著作者氏名と著作物

著作者氏名	SSL II/VPP のサブルーチン名 (スレッド並列機能のサブルーチン名)	項　　目
Richard Peirce Brent Peter Frederick Price	DP_VRANU4 (DM_VRANU4)	一様乱数 [0,1)の生成
Richard Peirce Brent Margaret Helen kahn	DP_VRANN3 (DM_VRANN3)	正規乱数の生成
Richard Peirce Brent Margaret Helen kahn	DP_VRANN4 (DM_VRANN4)	正規乱数の生成 (Wallace 法)
Murry Leslie Dow	DP_VBCSD (DM_VBCSD)	非対称または不定値のスパース実行列の連立 1 次方程式 (BICGSTAB(*l*)法, 対角形式格納法)
	DP_VBCSE (DM_VBCSE)	非対称または不定値のスパース実行列の連立 1 次方程式 (BICGSTAB(*l*)法, ELLPACK 形式格納法)
	DP_VCGD (DM_VCGD)	正値対称スパース行列の連立 1 次方程式 (前処理付き CG 法, 対角形式格納法)
	DP_VCGE (DM_VCGE)	正値対称スパース行列の連立 1 次方程式 (前処理付き CG 法, ELLPACK 形式格納法)
Lutz Grosz	DP_VAMLID (DM_VAMLID)	スパースな M-行列の連立 1 次方程式 (代数的マルチレベル反復法 [AMLI 法], 対角形式格納法)
	— (DM_VMLBIFE)	スパース行列の連立 1 次方程式 (不完全ブロック分解に基づくマルチレベル前処理付き反復法, ELLPACK 形式格納法)
	DP_VPDE2D (DM_VPDE2D)	2 次元 2 階偏微分方程式の有限差分法による離散化によるスパース行列の連立 1 次方程式の生成

著作者氏名	SSL II/VPP の サブルーチン名 (スレッド並列機能 のサブルーチン名)	項　　目
	DP_VPDE3D (DM_VPDE3D)	3 次元 2 階偏微分方程式の有限差分法による離散化によるスパース行列の連立 1 次方程式の生成
Zbigniew Leyk	DP_VMVSD (DM_VMVSD)	スパース実行列と実ベクトルの積 (対角形式格納法)
	DP_VMVSE (DM_VMVSE)	スパース実行列と実ベクトルの積 (ELLPACK 形式格納法)
	DP_VTFQD (DM_VTFQD)	非対称または不定値のスパース実行列の連立 1 次方程式 (TFQMR 法, 対角形式格納法)
	DP_VTFQE (DM_VTFQE)	非対称または不定値のスパース実行列の連立 1 次方程式 (TFQMR 法, ELLPACK 形式格納法)
Michael Robert Osborne David Lawrence Harrar II	DP_VTDEVC (DM_VTDEVC)	実三重対角行列の固有値・固有ベクトル

サブルーチン DM_VRADAU5 は, フリーソフトウェア RADAU5 に基づいて作られており, 以下の条件で提供するものです.

Copyright (c) 2004, Ernst Hairer

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN

CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

索　　引

さ

し

す

せ

た

ち

て

と

な

に

は

ひ

ふ

へ

ま

ゆ

ら