

疎行列による Krylov 部分空間法の実装

鈴木 厚¹

¹ 理研計算科学研究センター 大規模並列数値計算技術研究チーム
atsushi.suzuki.aj@a.riken.jp

疎行列格納方法 : 1/3

n : # 行

nnz : # 非零要素

$[A]_{ij}$: (i, j) 成分にある非零要素の値

- ▶ COO (Coordinate) format MUMPS

```
structure COOformat {  
  int n, nnz;  
  int irow[nnz];  
  int jcol[nnz];  
  double coef[nnz];  
};
```

- ▶ CSR (Compressed Sparse Row) /
CRS (Compressed Row Storage) 形式 Pardiso

```
structure CSRformat {  
  int n, nnz;  
  int pthrow[n+1];  
  int indcol[nnz];  
  double coef[nnz];  
}
```

$[A]_{ij} = \text{coef}[k]$
 $j = \text{indcol}[k], \text{ptrow}[i] \leq k < \text{ptrow}[i+1]$

疎行列格納方法 : 2/3

5×5 の非対称行列の例 : $n = 5, nnz = 15$.

1.1	1.2		1.4	
2.1	2.2	2.3		2.5
	3.2	3.3		
4.1			0.0	4.5
	5.2		5.4	5.5

	0	1	2	3	4
0	0	1		2	
1	3	4	5		6
2		7	8		
3	9			10	11
4		12		13	14

i	0		1					2		3			4		5
ptrow[i]	0		3					7		9			12		15
indcol[k]	0	1	3	0	1	2	4	1	2	0	3	4	1	3	4
coef[k]	1.1	1.2	1.4	2.1	2.2	2.3	2.5	3.2	3.3	4.1	0.0	4.5	5.2	5.4	5.5

- ▶ diagonal entry should exist even if the value is 0
- ▶ indcol[] should be in ascending order in each row

疎行列格納方法 : 2/3

5×5 の対称行列の上三角部分の例 : $n = 5, nnz = 10$.

1.1	1.2		1.4	
	2.2	2.3		2.5
		3.3		
			0.0	4.5
				5.5

	0	1	2	3	4
0	0	1		2	
1		3	4		5
2			6		
3				7	8
4					9

	<i>i</i>	0		1			2	3		4	5
ptrow[<i>i</i>]		0		3			6	7		9	10
indcol[<i>k</i>]		0	1	3	1	2	4	2	3	4	4
coef[<i>k</i>]		1.1	1.2	1.4	2.2	2.3	2.5	3.3	0.0	4.5	5.5

- ▶ 値が 0 であっても対角成分を記憶する
- ▶ indcol[] はそれぞれの行毎に昇順
- ▶ 上三角行列の保存形式は Pardiso 直接法で採用されている

SpMV : 疎行列ベクトル積演算 : 1/3

A : CSRformat { ptrow, indcol, coef }, $\vec{x}, \vec{y} \in \mathbb{R}^N$
 $\vec{y} = A\vec{x}$ の計算は次の様に実行される

$$\begin{aligned} [A\vec{x}]_i &= \sum_j [A]_{ij} [\vec{x}]_j \\ &= \sum_{j \in \{j; [A]_{ij} \neq 0\}} [A]_{ij} [\vec{x}]_j \\ &= \sum_{\text{ptrow}[i] \leq k < \text{ptrow}[i+1]} \text{coef}[k] \times [\vec{x}]_{\text{indcol}[k]} \end{aligned}$$

```
for (i = 0; i < n; i++) {  
    y[i] = 0.0;  
    for (k = ptrow[i]; k < ptrow[i + 1]; k++) {  
        int j = indcol[k];  
        y[i] += coef[k] * x[j];  
    }  
}
```

SpMV : 疎行列ベクトル積演算 : 2/3

A : CSRformat { ptrow, indcol, coef }, 対称行列のうち上三角部分を記憶,

$\vec{x}, \vec{y} \in \mathbb{R}^N$

$\vec{y} = A\vec{x}$ の演算を実行するため次を仮定する

- ▶ 行列 A の対角成分を記憶する
- ▶ i -行目の最初の非零成分は対角である : `indcol[ptrow[i]] == i`

```
for (i = 0; i < n; i++) {
    y[i] = 0.0;
}
for (i = 0; i < n; i++) {
    for (k = ptrow[i + 1]; k < ptrow[i + 1]; k++) {
        int j = indcol[k];
        y[i] += coef[k] * x[j];
        y[j] += coef[k] * x[i];
    }
    int k = ptrow[i];
    //     int j = indcol[k] ( == i )
    y[i] += coef[k] * x[i];
}
```

SpMV : 疎行列ベクトル積演算 : 3/3

スカラ α と β に対して CSRformat で格納している一般の非対称行列 A をベクトルに作用させる演算 $y = \alpha A\vec{x} + \beta\vec{y}$

```
void SparseGEMV(struct CSRformat &A,
                const double &alpha, std::vector<double> &x,
                const double &beta, std::vector<double> &y)
{
    int nrow = A.n;
    for (int i = 0; i < nrow; i++) {
        y[i] *= beta;
    }
    double tmp;
    for (int i = 0; i < nrow; i++) {
        tmp = 0.0;
        for (k = ptrow[i]; k < ptrow[i + 1]; k++) {
            int j = indcol[k];
            tmp += coef[k] * x[j];
        }
        y[i] += alpha * tmp;
    }
}
```

内積演算

二つのベクトル: $\vec{x}, \vec{y} \in \mathbb{R}^N$ に対して

$$(\vec{x}, \vec{y}) = \sum_{1 \leq i \leq N} [\vec{x}]_i [\vec{y}]_i$$

```
double tmp = 0.0;
for (i = 0; i < n; i++) {
    tmp += x[i] * y[i];
}
```

BLAS レベル 1 サブルーチン ddot

```
double cblas_ddot(const int n,
                  const double *x, const int incx,
                  const double *y, const int incy);

tmp = cblas_ddot(n, &x[0], 1, &y[0], 1);
```

C++ STL `std::vector< >` クラス

```
#include <vector>
std::vector<double> x(100); // allocation with size = 100
x.resize(200); // enlarging array with keeping 100 entries
x.clean(); // deallocation
&x[0]; // double pointer to the first entry
```


疎行列の格納形式の変換 : COO 形式から CSR 形式へ

- ▶ 各行の記憶のため動的なメモリー確保が不可欠

```
std::vector<std::list<int> > pcol(nrow);
std::vector<std::list<double> > pcoef(nrow);
for (int k = 0; k < nnz; k++) {
    int i = Acoo.irow[k], j = Acoo.jcol[k];
    double coef = Acoo.coef[k];

    if (pcol[i].empty()) {
        pcol[i].push_back(j); pcoef[i].push_back(coef); }
    else {
        if (pcol[i].back() < j) {
            pcol[i].push_back(j); pcoef[i].push_back(coef); }
        else {
            typename std::list<double> iv = pcoef[i].begin();
            typename std::list<int> it = pcol[i].begin();
            for (; it != pcol[i].end(); ++it, ++iv) {
                if ((*it) > j) {
                    pcol[i].insert(it, j); pcoef[i].insert(iv, coef); } } }
    }
}
Acsr.ptrow[0] = 0;
int k = 0;
for (int i = 0; i < n; i++) {
    ptrow[i + 1] = ptrow[i] + pcol[i].size();
    typename std::list<double> iv = pcoef[i].begin();
    typename std::list<int> it = pcol[i].begin();
    for (; it != pcol[i].end(); ++it, ++iv, k++) {
        Acsr.pcol[k] = (*it); Acsr.coef[k] = (*iv); } }
```

疎行列の格納形式の変換 : CSSR 形式から COO 形式へ

```
int nrow = Acsr.n;
for (int i = 0; i < n; i++) {
    for (int k = Acsr.ptrow[i]; k < Acsr.ptrow[i + 1]; k++) {
        Acoo.irow[k] = i;
        Acoo.jcol[k] = Acsr.indcol[k];
        Acoo.coef[k] = Acsr.coef[k];
    }
}
```

実習 : IML++ で C++ で記述された GMRES を実装する

IML++, <https://math.nist.gov/impl++/gmres.h.txt> では GMRES は C++ で記述されている

最初の実装は前処理 `const Preconditioner &M` 無しで `Operator`, `Vector` と `Matrix` クラスを簡単なもので置き換える

- ▶ `Real` は `double`
- ▶ `Vector` は `std::vector<double>`
- ▶ `Operator` は `structure CSRformat`
- ▶ `Matrix` は 一次元配列 `std::vector<double>` を二つの添え字で参照するものとして

```
std::vector<double> HH((max_iter + 1) * (max_iter + 1));  
#define H(i, j) HH((i) + (j) * (max_iter + 1))
```

残差ベクトルを計算する

```
r = b - A * x;
```

は C++ の演算子で記述されているが、疎行列の GEMV ルーチンで置き換える

```
std::vector<double> r(b);  
SparseGEMV(A, (-1.0), x, 1.0, r);
```