Decomposição de Cholesky OpenMP

Larissa Trindade

1. Contexto

Definição

Decomposição de Cholesky ou Fatoração de Cholesky é um método de álgebra linear para *resoluções de sistemas lineares.*

Para utilizar este método e necessário que a matriz do sistema linear seja **quadrada** (n x n), **simétrica** e **definida positiva.**

Para utilizar a decomposição de Cholesky é utilizado a equação (1) onde **A** é a *matriz inicial* e **L** é uma *matriz triangular inferior* com elementos da diagonal principal estritamente positivos.

$$A = LL^T$$

Especificações da máquina

Intel® Core™ i5-7200	
Core	2
Threads	4
Cache	3 MB Intel® Smart Cache
Memória	16 GB

2. Algoritmo

Algoritmo

anterior

```
void cholesky(double** A, int n) {
  for (int i = 0; i < n; i++) //linha
    for (int j = 0; j < (i + 1); j++) { //coluna
      double s = 0;
      for (int k = 0; k < j; k++)  s += A[i][k] * A[j][k];
      A[i][j] = (i == j) ? sqrt(A[i][i] - s) : (1.0 / A[j][j] * (A[i][j] - s));
    }
}
```

novo

```
void cholesky(double** A, int n) {
  double s = 0;
  int diagonal = 0;
  int i, j, k;
  for (i = 0; i < n; i++) { //coluna
    s = 0;
    if (diagonal == 0) {
      diagonal = 1;
      for (k = 0; k < i; k++)  s += A[i][k] * A[i][k];
      A[i][i] = sqrt(A[i][i] - s);
    }
    for (j = i + 1; j <  n; j++) { //linha
      for (k = 0; k < i; k++)  s += A[j][k] * A[i][k];
      A[j][i] = (1.0 / A[i][i] * (A[j][i] - s));
      A[i][j] = A[j][i];
    }
    diagonal = 0;
  }
}
```

Algoritmo

$$A = \begin{pmatrix} \boxed{a_{11}} & a_{12} & a_{13} \\ a_{21} & \boxed{a_{22}} & a_{23} \\ a_{31} & a_{32} & \boxed{a_{33}} \end{pmatrix} = \begin{pmatrix} l_{11}^2 & l_{11}l_{21} & l_{11}l_{31} \\ l_{11}l_{21} & l_{21}^2 + l_{22}^2 & l_{21}l_{31} + l_{22}l_{32} \\ l_{11}l_{31} & l_{21}l_{31} + l_{22}l_{32} & l_{31}^2 + l_{32}^2 + l_{33}^2 \end{pmatrix}$$

$$L = \begin{pmatrix} \boxed{l_{11}} & 0 & 0 \\ l_{21} & \boxed{l_{22}} & 0 \\ l_{31} & l_{32} & \boxed{l_{33}} \end{pmatrix}$$

Diagonal Principal

$$l_{11} = \sqrt{a_{11}} \qquad l_{22} = \sqrt{a_{22} - l_{21}^2} \qquad l_{33} = \sqrt{a_{33} - l_{31}^2 - l_{32}^2}$$

$$\boxed{l_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2}}$$

Algoritmo

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ \boxed{a_{21}} & a_{22} & a_{23} \\ \boxed{a_{31}} & \boxed{a_{32}} & a_{33} \end{pmatrix} = \begin{pmatrix} l_{11}^2 & l_{11}l_{21} & l_{11}l_{31} \\ l_{11}l_{21} & l_{21}^2 + l_{22}^2 & l_{21}l_{31} + l_{22}l_{32} \\ l_{11}l_{31} & l_{21}l_{31} + l_{22}l_{32} & l_{31}^2 + l_{32}^2 + l_{33}^2 \end{pmatrix}$$

$$L = \begin{pmatrix} l_{11} & 0 & 0 \\ \boxed{l_{21}} & l_{22} & 0 \\ \boxed{l_{31}} & \boxed{l_{32}} & l_{33} \end{pmatrix}$$

Abaixo da Diagonal

$$l_{21} = \frac{1}{l_{11}} a_{21} \qquad l_{31} = \frac{1}{l_{11}} a_{31} \qquad l_{32} = \frac{1}{l_{22}} (a_{31} - l_{31} l_{21})$$

$$\boxed{l_{ji} = \frac{1}{l_{ii}} a_{ji} - \sum_{k=1}^{j-1} l_{jk} l_{ik}}$$

Algoritmo

```
void cholesky(double** A, int n) {
  double s = 0;
  int diagonal = 0;
  int i, j, k;
  for (i = 0; i < n; i++) { //coluna
      s = 0;
      if (diagonal == 0) {
        diagonal = 1;
        for (k = 0; k < i; k++)  s += A[i][k] * A[i][k];
        A[i][i] = sqrt(A[i][i] - s);
      }
     for (j = i + 1; j <  n; j++) { //linha
         for (k = 0; k < i; k++)  s += A[j][k] * A[i][k];
         A[j][i] = (1.0 / A[i][i] * (A[j][i] - s));
         A[i][j] = A[j][i];
      }
      diagonal = 0;
   }
}
```

Diagonal Principal

$$l_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2}$$

Algoritmo

```
void cholesky(double** A, int n) {
  double s = 0;
  int diagonal = 0;
  int i, j, k;
  for (i = 0; i < n; i++) { //coluna
     s = 0;
     if (diagonal == 0) {
        diagonal = 1;
        for (k = 0; k < i; k++)  s += A[i][k] * A[i][k];
        A[i][i] = sqrt(A[i][i] - s);
     }
    for (j = i + 1; j <  n; j++) { //linha
        for (k = 0; k < i; k++)  s += A[j][k] * A[i][k];
        A[j][i] = (1.0 / A[i][i] * (A[j][i] - s));
        A[i][j] = A[j][i];
    }
     diagonal = 0;
  }
}
```

Abaixo da Diagonal

$$l_{ji} = \frac{1}{l_{ii}} a_{ji} - \sum_{k=1}^{j-1} l_{jk} l_{ik}$$

Algoritmo paralelizado

```c
void cholesky(double** A, int n) {
  double s = 0;
  int diagonal = 0;
  int i, j, k;
  for (i = 0; i < n; i++) { //coluna
      s = 0;
      if (diagonal == 0) {
         diagonal = 1;
         for (k = 0; k < i; k++)  s += A[i][k] * A[i][k];
         A[i][i] = sqrt(A[i][i] - s);
      }
    #pragma omp parallel for shared(A, i, n) private(j, k, s)
    for (j = i + 1; j <  n; j++) { //linha
         for (k = 0; k < i; k++)  s += A[j][k] * A[i][k];
         A[j][i] = (1.0 / A[i][i] * (A[j][i] - s));
         A[i][j] = A[j][i];
    }
      diagonal = 0;
  }
}
```

$$l_{ji} = \frac{1}{l_{ii}} a_{ji} - \sum_{k=1}^{j-1} l_{jk} l_{ik}$$

Exemplo com 2 Threads

$$A = \begin{pmatrix} 25 & 15 & -5 \\ 15 & 18 & 0 \\ -5 & 0 & 11 \end{pmatrix}$$

i=0

$$A = \begin{pmatrix} \boxed{\begin{matrix} 25 \\ 15 \\ -5 \end{matrix}} & \begin{matrix} 15 \\ 18 \\ 0 \end{matrix} & \begin{matrix} -5 \\ 0 \\ 11 \end{matrix} \end{pmatrix}$$

diagonal = 0

$$A = \begin{pmatrix} \boxed{25} & 15 & -5 \\ 15 & 18 & 0 \\ -5 & 0 & 11 \end{pmatrix}$$

$$l_{11} = \sqrt{a_{11}} = 5$$

$$A = \begin{pmatrix} \boxed{5} & 15 & -5 \\ 15 & 18 & 0 \\ -5 & 0 & 11 \end{pmatrix}$$

```
void cholesky(double** A, int n) {
  double s = 0;
  int diagonal = 0;
  int i, j, k;
  for (i = 0; i < n; i++) { //coluna
     s = 0;
     if (diagonal == 0) {
        diagonal = 1;
        for (k = 0; k < i; k++)  s += A[i][k] * A[i][k];
        A[i][i] = sqrt(A[i][i] - s);
     }
    #pragma omp parallel for shared(A, i) private(j, k, s)
    for (j = i + 1; j <  n; j++) { //linha
        for (k = 0; k < i; k++)  s += A[j][k] * A[i][k];
        A[j][i] = (1.0 / A[i][i] * (A[j][i] - s));
        A[i][j] = A[j][i];
     }
     diagonal = 0;
  }
}
```

Exemplo com 2 Threads

i = 0

$$A = \begin{pmatrix} \boxed{\begin{matrix} 5 \\ 15 \\ -5 \end{matrix}} & \begin{matrix} 15 \\ 18 \\ 0 \end{matrix} & \begin{matrix} -5 \\ 0 \\ 11 \end{matrix} \end{pmatrix}$$

diagonal = 1

Parallel for – 2 Threads

j = i + 1

Thread 0

i = 0 j = 1

$$A = \begin{pmatrix} 5 & 15 & -5 \\ \boxed{15} & 18 & 0 \\ -5 & 0 & 11 \end{pmatrix}$$

$$l_{21} = \frac{1}{l_{11}} a_{21} = \frac{15}{5} = 3$$

Thread 1

i = 0 j = 2

$$A = \begin{pmatrix} 5 & 15 & -5 \\ 15 & 18 & 0 \\ \boxed{-5} & 0 & 11 \end{pmatrix}$$

$$l_{31} = \frac{1}{l_{11}} a_{31} = \frac{-5}{5} = -1$$

```
void cholesky(double** A, int n) {
  double s = 0;
  int diagonal = 0;
  int i, j, k;
  for (i = 0; i < n; i++) { //coluna
      s = 0;
      if (diagonal == 0) {
          diagonal = 1;
          for (k = 0; k < i; k++)  s += A[i][k] * A[i][k];
          A[i][i] = sqrt(A[i][i] - s);
      }
    #pragma omp parallel for shared(A, i) private(j, k, s)
    for (j = i + 1; j <  n; j++) { //linha
         for (k = 0; k < i; k++)  s += A[j][k] * A[i][k];
         A[j][i] = (1.0 / A[i][i] * (A[j][i] - s));
         A[i][j] = A[j][i];
      }
      diagonal = 0;
   }
}
```

Exemplo com 2 Threads

i=1

$$A = \begin{pmatrix} 5 & \boxed{3} & -1 \\ 3 & \boxed{18} & 0 \\ -1 & \boxed{0} & 11 \end{pmatrix}$$

diagonal = 0

$$A = \begin{pmatrix} 5 & 3 & -1 \\ 3 & \boxed{18} & 0 \\ -1 & 0 & 11 \end{pmatrix}$$

$$l_{22} = \sqrt{a_{22} - l_{21}^2} = \sqrt{18 - 3^2} = \sqrt{9} = 3$$

$$A = \begin{pmatrix} 5 & 3 & -1 \\ 3 & \boxed{3} & 0 \\ -1 & 0 & 11 \end{pmatrix}$$

```
void cholesky(double** A, int n) {
  double s = 0;
  int diagonal = 0;
  int i, j, k;
  for (i = 0; i < n; i++) { //coluna
     s = 0;
     if (diagonal == 0) {
        diagonal = 1;
        for (k = 0; k < i; k++)  s += A[i][k] * A[i][k];
        A[i][i] = sqrt(A[i][i] - s);
     }
    #pragma omp parallel for shared(A, i) private(j, k, s)
    for (j = i + 1; j <  n; j++) { //linha
        for (k = 0; k < i; k++)  s += A[j][k] * A[i][k];
        A[j][i] = (1.0 / A[i][i] * (A[j][i] - s));
        A[i][j] = A[j][i];
    }
    diagonal = 0;
  }
}
```

Exemplo com 2 Threads

i = 1

$$A = \begin{pmatrix} 5 & \boxed{3} & -1 \\ 3 & \boxed{3} & 0 \\ -1 & \boxed{0} & 11 \end{pmatrix}$$

diagonal = 1

Parallel for – 2 Threads
j = i + 1

Thread 1
não necessária

Thread 0

i = 1   j = 2

$$A = \begin{pmatrix} 5 & 3 & -1 \\ 3 & 3 & 0 \\ -1 & \boxed{0} & 11 \end{pmatrix}$$

$$l_{32} = \frac{1}{l_{22}}(a_{31} - l_{31} l_{21}) = \frac{1}{3}(0 - (-1 * 3)) = \frac{3}{3} = 1$$

```
void cholesky(double** A, int n) {
  double s = 0;
  int diagonal = 0;
  int i, j, k;
  for (i = 0; i < n; i++) { //coluna
     s = 0;
     if (diagonal == 0) {
        diagonal = 1;
        for (k = 0; k < i; k++)  s += A[i][k] * A[i][k];
        A[i][i] = sqrt(A[i][i] - s);
     }
    #pragma omp parallel for shared(A, i) private(j, k, s)
    for (j = i + 1; j <  n; j++) { //linha
        for (k = 0; k < i; k++)  s += A[j][k] * A[i][k];
        A[j][i] = (1.0 / A[i][i] * (A[j][i] - s));
        A[i][j] = A[j][i];
     }
     diagonal = 0;
  }
}
```

Exemplo com 2 Threads

i=2

$$A = \begin{pmatrix} 5 & 3 & \boxed{-1} \\ 3 & 3 & \boxed{1} \\ -1 & 1 & \boxed{11} \end{pmatrix}$$

diagonal = 0

$$A = \begin{pmatrix} 5 & 3 & -1 \\ 3 & 3 & 1 \\ -1 & 1 & \boxed{11} \end{pmatrix}$$

$$l_{33} = \sqrt{a_{33} - l_{31}^2 - l_{32}^2} = \sqrt{11 - (-1)^2 - 1^2} = \sqrt{9} = 3$$

$$A = \begin{pmatrix} 5 & 3 & -1 \\ 3 & 3 & 1 \\ -1 & 1 & \boxed{3} \end{pmatrix}$$

```
void cholesky(double** A, int n) {
  double s = 0;
  int diagonal = 0;
  int i, j, k;
  for (i = 0; i < n; i++) { //coluna
      s = 0;
      if (diagonal == 0) {
         diagonal = 1;
         for (k = 0; k < i; k++)  s += A[i][k] * A[i][k];
         A[i][i] = sqrt(A[i][i] - s);
      }
     #pragma omp parallel for shared(A, i) private(j, k, s)
     for (j = i + 1; j <  n; j++) { //linha
          for (k = 0; k < i; k++)  s += A[j][k] * A[i][k];
          A[j][i] = (1.0 / A[i][i] * (A[j][i] - s));
          A[i][j] = A[j][i];
      }
      diagonal = 0;
  }
}
```

3. Resultados

3 entradas analisadas e definidas pelo trabalho 1:

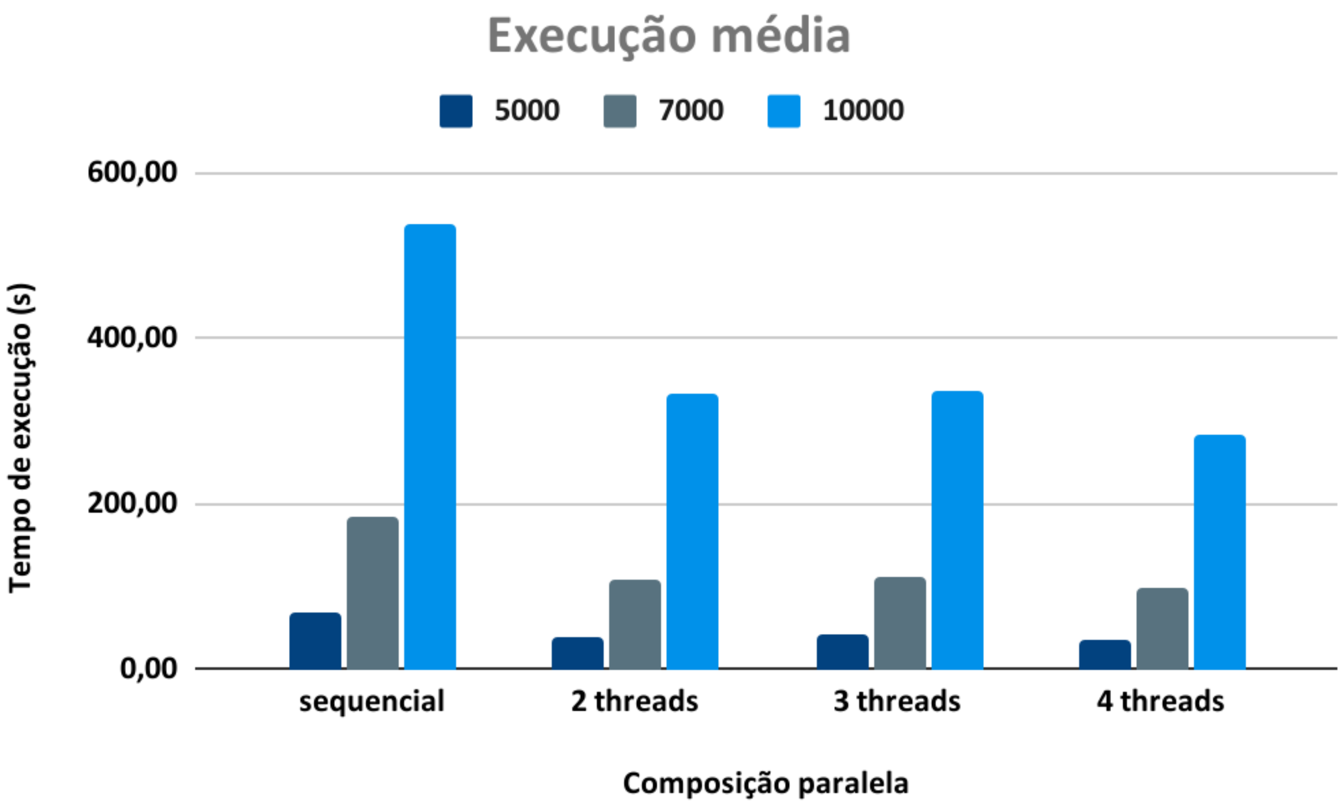
- *cholesky_5000.in*
- *cholesky_7000.in*
- *cholesky_10000.in*

Verificação: Próprio programa da Decomposição de Cholesky (caso algo esteja errado e a matriz não esteja correta, o arquivo .out possui algumas linhas com **#INDO**)

Resultados – Tempo médio

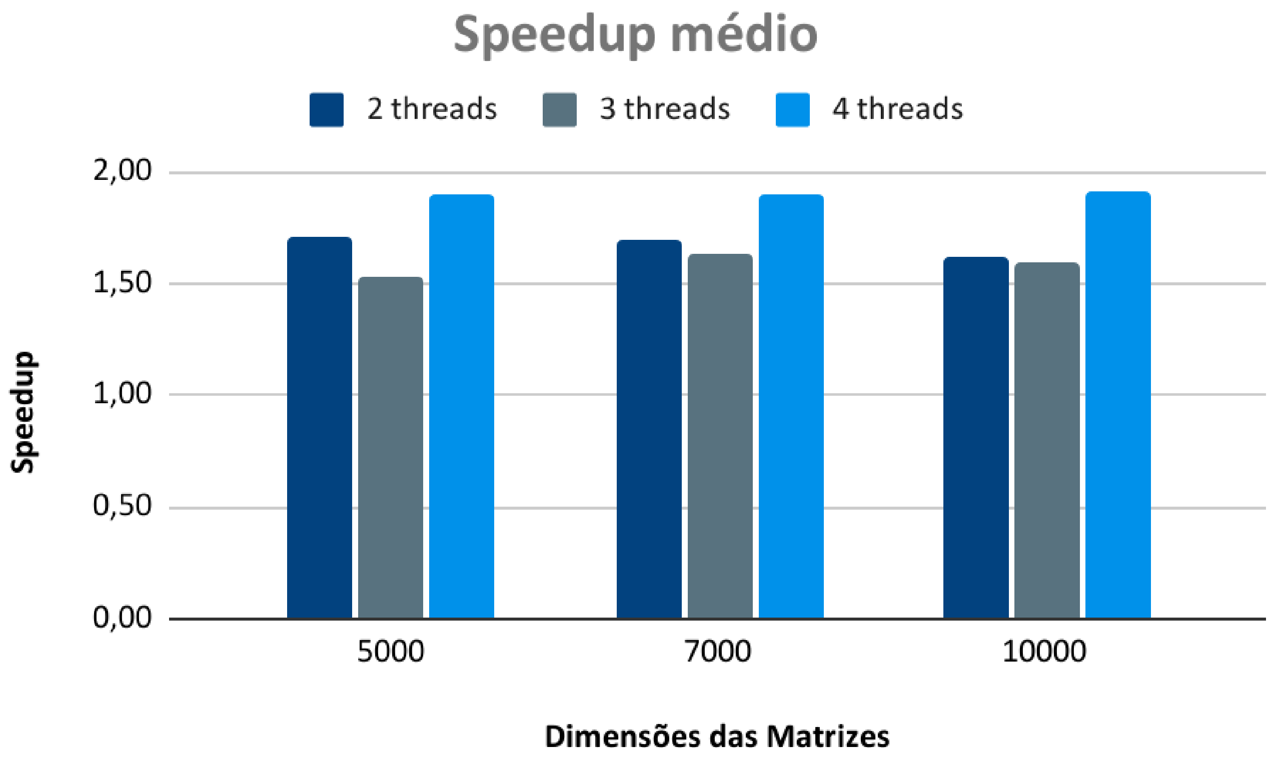
Tamanho	Sequencial Antigo (min)	Sequencial Novo (min)	2 Threads (s)	3 Threads (s)	4 Threads (s)
5000	71,4	68,12	39,98	44,42	35,83
7000	189	185,41	109,52	113,67	97,91
10000	547,8	539,32	332,08	337,29	282,64

Média realizada com a remoção dos outliers


Execução média
5000
7000
10000
600,00
400,00
200,00
0,00
Tempo de execução (s)
sequencial
2 threads
3 threads
4 threads
Composição paralela

Speedup - Médio com omp

Tamanho	2 Threads	3 Threads	4 Threads
5000	1,70	1,53	1,90
7000	1,69	1,63	1,89
10000	1,62	1,60	1,91


Speedup médio
2 threads
3 threads
4 threads
2,00
1,50
1,00
0,50
0,00
Speedup
5000
7000
10000
Dimensões das Matrizes

Obrigada!

lala.trindade.2008@gmail.com

https://github.com/LarissaTrin/CholeskyDecomposition