Eliminação gaussiana com pivoteamento parcial

Arthur Paiva, Lucio Reis, Rafael Campbell

Qual o objetivo?

Resolver **sistemas de equações lineares** computacionalmente.

O que é uma equação linear?

O que é uma equação linear?

São equações onde todos as variáveis têm expoente igual a um e não há multiplicação entre si.

São descritas em sua forma geral:

$$a_1X_1 + a_2X_2 + ... + a_nX_n = b$$

Sistema de equações lineares

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ a_{31}x_1 + a_{32}x_2 + \ldots + a_{3n}x_n = b_3 \end{cases}$$

Como representar computacionalmente?

Como representar computacionalmente?

Matrizes!

$$AX = B$$

Representação matricial

$$AX = B$$

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \mathbf{x} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \mathbf{=} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

Método

Três passos!

Método

1. Obter uma matriz aumentada [A|B]

$$\left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{array}\right]$$

Método

2. Obter uma matriz equivalente [A’|B’], onde A é triangular superior.

$$\left[\begin{array}{cccc|c} a'_{11} & a'_{12} & \cdots & a'_{1n} & b'_1 \\ 0 & a'_{22} & \cdots & a'_{2n} & b'_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & a'_{nn} & b'_n \end{array}\right]$$

Método

3. Resolver o sistema de equações A'X = B' por substituição regressiva.

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} (b'_1 - x_2 a'_{12} - \ldots - x_n a'_{1n}) \div a'_{11} \\ (b'_2 - x_3 a'_{23} - \ldots - x_n a'_{2n}) \div a'_{22} \\ \vdots \\ b'_n \div a'_{nn} \end{bmatrix}$$

Restrições

1. Matriz quadrada
2. Pivôs não nulos
3. Matriz densa

Algoritmo

Algoritmo

1. Obter uma matriz aumentada [A|B]

$$\text{intervaloDecimal} = 100 \text{ /*valor máximo*/} \times 100 \text{ /*duas casas decimais*/}$$
$$\text{Para cada linha } L_i\text{, faça:}$$
$$\quad \text{Para cada coluna } C_j\text{, faça:}$$
$$\quad\quad \text{Constante} = \text{ValorAleatorio(intervaloDecimal)} \div fatorDecimal$$
$$\quad\quad M_{ij} = \text{Constante}$$
$$\quad\quad \text{Soma} = \text{Soma} + \text{Constante} \times \text{X}_j$$
$$\quad B_i = \text{Soma}$$
$$\quad \text{Soma} = 0$$

Algoritmo

2. Obter uma matriz equivalente [A’|B’], onde A é triangular superior.

Para cada coluna i, faça:

Para cada linha L_k, se $k > i$, faça:

$$L'_k \leftarrow L_k - (L_i \times (a_{ki}/a_{ii}))$$

Algoritmo

2. Obter uma matriz equivalente [A’|B’], onde A é triangular superior.

Para cada coluna i, faça:

Para cada linha L_k, se $k > i$, faça:

$$L'_k \leftarrow L_k - (L_i \times (a_{ki} / a_{ii}))$$

E se o pivô for nulo?

Algoritmo

2.a Realizar pivoteamento parcial

p $= i$ //define-se pivô como i
Para cada linha L_j, se $j > i$, faça:
    Se $a_{ji} > a_{pi}$, faça:
        $p = j$ //define-se um novo pivô
Se $p \neq i$, faça:
    $L_i \leftrightarrow L_p$ //permutar linhas

Algoritmo

3. Resolver o sistema de equações A'X = B' por substituição regressiva.

$$\begin{aligned}
&\text{Para cada linha } L_i, \text{ de } n \rightarrow 0, \text{ faça:} \\
&\qquad \text{Para cada coluna } C_j, \text{ de } i+1 \rightarrow n, \text{ faça:} \\
&\qquad\qquad \text{Soma} = \text{Soma} + (a_{ij} \times x_j) \\
&x_i = (b_i - \text{ Soma }) \div a_{ii}
\end{aligned}$$

Algoritmo

Como paralelizar?

Algoritmo

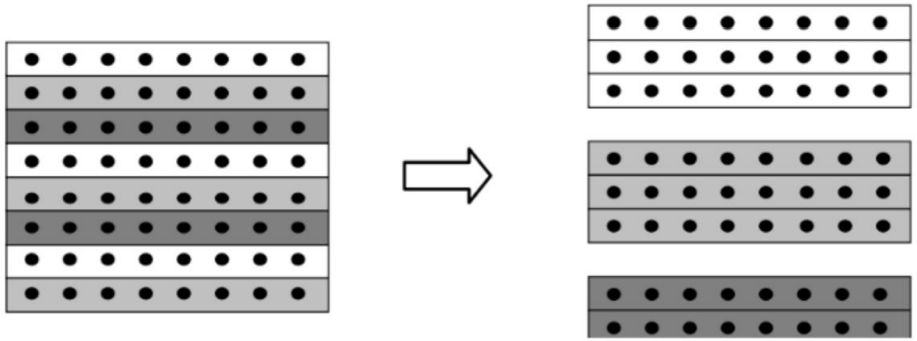
Como paralelizar?

Essencialmente, paraleliza-se a segunda etapa.

Algoritmo

Como paralelizar?

- **A busca pelo pivô é dividida em blocos ou ciclos**
- **A resolução das linhas é dividida em ciclos**



Algoritmo Serial

Para cada coluna C_i, faça:

Seleciona o valor mais significativo da coluna

Permuta a linha L_i com a do pivô

Zera todos os valores abaixo da diagonal principal

OpenMP

1. As equações serão divididas em ciclos
2. A busca pelo pivô será dividida em blocos com **redução**

OpenMP

Como são divididos os ciclos?

```
#pragma omp parallel for
        private(term,k,j)
        schedule(static,1)
```

OpenMP

Como é feita a redução?

```
typedef struct Compare {
        double val;
        int index;
} compare;
```

OpenMP

Como é feita a redução?

```
#pragma omp declare reduction
        (maximum : compare :
                omp_out = omp_in.val > omp_out.val ?
                        omp_in :
                        omp_out)
```

Pthread

1. As equações serão divididas em ciclos
2. A busca pelo pivô será dividida em ciclos
3. Usa-se uma estrutura **gauss_elimination_param** para passar informação entre threads.
4. Usa-se os métodos **pthread_join** e **pthread_create** para gerenciar criação e finalização dos processos

Pthread

Como são divididos os ciclos?

```
for(int i = column + 1 + threadID;
        i < matrixSize;
        k += NUM_THREADS)
```

Pthread

Como é a estrutura?

```
struct gauss_elimination_param {
        int col; //coluna atual
        int threadNumber; //ID da thread aberta
        int matrixSize; //tamanho da matriz
        double **matrix; //matriz A
        double *arrayB; //vetor B
};
```

MPI

1. As equações serão divididas em ciclos com **Send/Recv**
2. A busca pelo pivô será dividida em blocos com **Scatter e Redução**

MPI

Como são divididos os ciclos?

```
Para cada ciclo, faça:
        Para cada processo, faça:
                Envia linha
        Para cada processo, faça:
                Recebe linha
```

MPI

Como é feita a redução?

```
typedef struct pivotValueIndex {
        double val;
        int index;
} PIVOTVALUEINDEX;
```

MPI

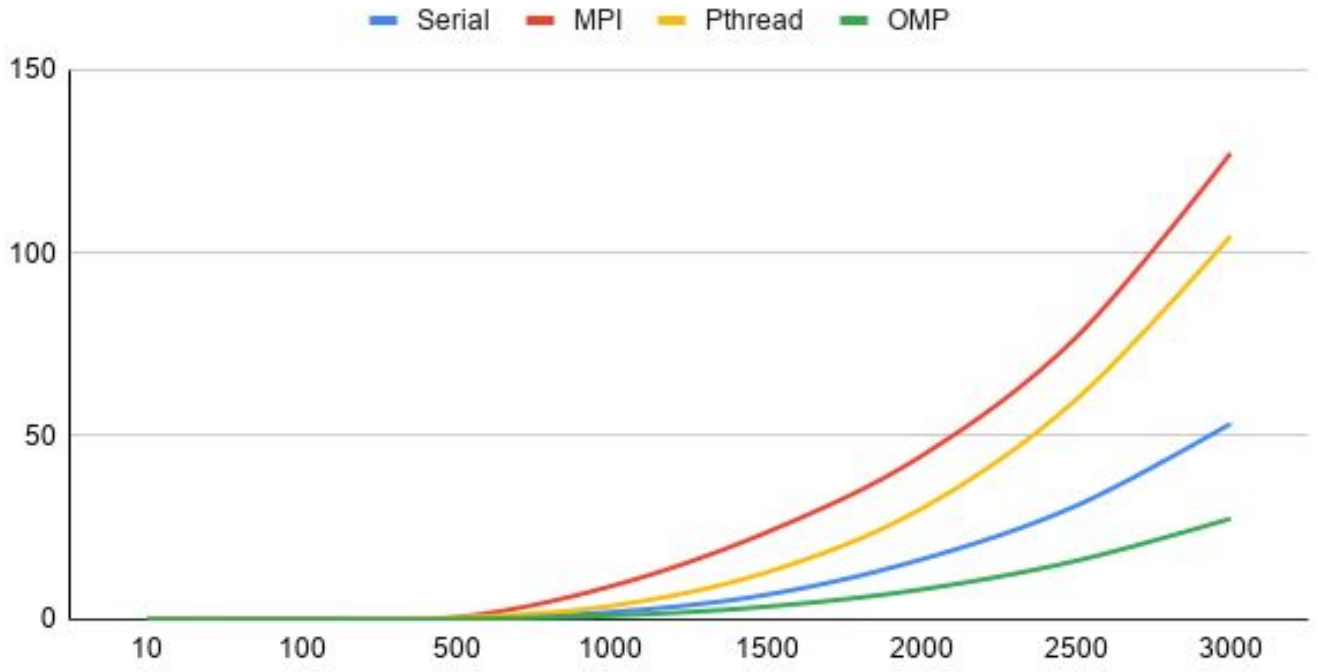
Como é feita a redução?

```
MPI_Reduce(&bigger,
           &pivot,
           1,
           MPI_DOUBLE_INT,
           MPI_MAXLOC,
           0,
           MPI_COMM_WORLD)
```

Resultados

Tamanho	Serial	MPI	Pthread	OMP
10	0.000005	0.000838	0.001445	0.001980
100	0.002155	0.044342	0.020939	0.006778
500	0.251302	0.568266	0.529727	0.130077
1000	1.983525	9.063898	3.682738	1.023678
1500	6.699230	23.693881	12.798478	3.459354
2000	16.310575	44.483196	30.115219	8.148721
2500	30.926184	76.815634	59.849367	15.947293
3000	53.377539	127.159432	104.564415	27.473075

Resultados


Serial
MPI
Pthread
OMP
150
100
50
0
10
100
500
1000
1500
2000
2500
3000