Universidade Federal do Ceará
Campus de Quixadá
Matemática Computacional (2017.1)
Prof. Wladimir Araújo Tavares

Trabalho de Implementação
Sistemas Lineares
Data de Entrega: 27/04/2017
Trabalho em equipe máximo 2 alunos
Plágio será punidos com nota zero

Repositório: `https://github.com/WladimirTavares/matematica_computacional2017/tree/master/sistemas_lineares`

No código disponível no repositório acima, temos o processo completo de resolução de sistema linear $Ax = b$ utilizando o método de eliminação de Gauss (MEG).

O método de resolução de sistema linear utilizando o MEG pode ser separado em três etapas:

- Etapa 1: Construção da matriz aumentada $[A|b]$
- Etapa 2: Transformação da matriz aumentada [A|b] em um outra matriz aumentada $[\overline{A}|\overline{b}]$, onde $\overline{A}$ é uma matriz triangular superior.
- Etapa 3: Resolução do sistema linear $[\overline{A}|\overline{b}]$ por retrosubstituição, obtendo-se assim as soluções x do sistema linear original $Ax = b$

1. O processo de resolução de resolução de sistema linear utilizando o MEG pode ser comprometido por causa da escolha de algum pivô nulo ($a_{kk} = 0$). Neste caso, o processo pode continuar simplesmente permutando a linha k com uma outra linha cujo coeficiente na coluna seja diferente de zero. Com o objetivo de minimizar os erros de arredondamento, escolha o coeficiente com o maior valor e em caso de empate, escolha o coeficiente que vem primeiro. Implemente a eliminação de Gauss com pivoteamento parcial que transforma a matriz original em uma matriz triangular superior e devolve o determinante dessa matriz.

   ```
   //Transforma a matriz em uma matriz triangular superior e retorna o determ
   //LD MATRIX::GaussParcialPivot();
   ```

   $$\left[\begin{array}{cccc|c} 2 & 1 & 1 & 0 & 1 \\ 4 & 3 & 3 & 1 & 2 \\ 8 & 7 & 9 & 5 & 4 \\ 6 & 7 & 9 & 8 & 5 \end{array}\right] \rightarrow \left[\begin{array}{cccc|c} 8 & 7 & 9 & 5 & 4 \\ 0 & 7/4 & 9/4 & 17/4 & 2 \\ 0 & 0 & -6/7 & -2/7 & 4/7 \\ 0 & 0 & 0 & 2/3 & 2/3 \end{array}\right]$$

2. O método de Gauss-Jordan consiste transformar a matriz original em uma matriz diagonal unitária utilizando as operações elementares. Implemente o método Gauss-Jordan que realiza essa tarefa.

```
//Transforma a matriz em uma matriz diagonal unitária e
//retorna o determinante da matriz inicial
//LD MATRIX::GaussJordan();
```

$$\left[\begin{array}{ccc|c} 5 & 5 & 0 & 15 \\ 2 & 4 & 1 & 10 \\ 3 & 4 & 0 & 11 \end{array}\right] \rightarrow \left[\begin{array}{ccc|c} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 0 \end{array}\right]$$

3. O método de Gauss-Jordan pode ser utilizado para obtenção da inversa de uma matriz. A inversa de uma matriz pode ser obtida seguindo dois passos:

   - Etapa 1: Construção da matriz aumentada $[A|I]$
   - Etapa 2: Transforma da matriz aumentada $[A|I]$ em uma matriz diagonal unitária $[I|A^{-1}]$ utilizando o método de Gauss-Jordan
   - Etapa 3: Guardar a matriz inversa na matriz original.

   Implemente o método para encontrar a inversa de uma matriz.

```
//Transforma a matriz original na matriz inversa e
//devolve true se a matriz é inversível
//devolve false caso contrario
bool MATRIX::inversa();
```

$$\left[\begin{array}{ccc|ccc} 5 & 5 & 0 & 1 & 0 & 0 \\ 2 & 4 & 1 & 0 & 1 & 0 \\ 3 & 4 & 0 & 0 & 0 & 1 \end{array}\right] \rightarrow \left[\begin{array}{ccc|ccc} 8/10 & 0 & -1 & 1 & 0 & 0 \\ -6/10 & 0 & 1 & 0 & 1 & 0 \\ 4/5 & 1 & -2 & 0 & 0 & 1 \end{array}\right]$$

4. O process de solução de um sistema linear $Ax = b$ utilizando a matriz inversa pode ser realizada em duas etapas:

   - Encontre a matriz inversa A^{-1}
   - Devolve o vetor $x = A^{-1}b$

   Implemente o método solveByInverse que resolve uma sistema linear seguindo as etapas acima:

```
vector <LD> solveByInverse(MATRIX& M, vector<LD> b)
```

$$Ax = b \rightarrow \begin{bmatrix} 5 & 5 & 0 \\ 2 & 4 & 1 \\ 3 & 4 & 0 \end{bmatrix} x = \begin{bmatrix} 15 \\ 10 \\ 11 \end{bmatrix} \rightarrow x = A^{-1}b \rightarrow$$

$$x = \begin{bmatrix} 8/10 & 0 & -1 \\ -6/10 & 0 & 1 \\ 4/5 & 1 & -2 \end{bmatrix} \begin{bmatrix} 15 \\ 10 \\ 11 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}$$

5. A decomposição LU da matriz A devolver uma matriz triangular inferior com a diagonal unitária L e um uma matriz triangular superior U tal que A = LU.

$$A = L * U = \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

A matriz L e U podem ser codificadas em uma única matriz chamada matriz de decomposição LU.

$$LU = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ l_{21} & u_{22} & u_{23} \\ l_{31} & l_{32} & u_{33} \end{bmatrix}$$

Implemente o método decomposeLU que transforma a matriz original em uma matriz de decomposição LU utilizando o método de eliminação de Gauss sem pivoteamento.

```
//Transforma a matriz original na matriz de decomposição LU
//LD MATRIX::decomposeLU();
```

$$A = \begin{bmatrix} 3 & 2 & 4 \\ 1 & 1 & 2 \\ 4 & 3 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1/3 & 1 & 0 \\ 4/3 & 1 & 1 \end{bmatrix} \begin{bmatrix} 3 & 2 & 4 \\ 0 & 1/3 & 2/3 \\ 0 & 0 & -4 \end{bmatrix}$$

Matriz de decomposição LU

$$\begin{bmatrix} 3 & 2 & 4 \\ 1/3 & 1/3 & 2/3 \\ 4/3 & 1 & -4 \end{bmatrix}$$

6. O processo de solução de uma sistema linear $Ax = b$ utilizando a decomposição LU pode ser realizada em três etapas:

   (a) Encontre a decomposição LU

   (b) Resolva o sistema Ly = b usando substituição progressiva.

   (c) Resolva o sistema Ux = y usando substituição progressiva.

   Implemente o método de solveByLU que que resolve um sistema linear seguindo as etapas acima.

   ```
   vector <LD> solveByLU(MATRIX& A, vector<LD> b)
   ```

7. Implemente o método de Gauss-Jacobi para a obtenção de uma aproximação para a solução de um sistema linear $Ax = b$

   ```
   //xinicial : aproximacao inicial
   //tol : representa a tolerância para que a solução seja aceita
   //MAX_ITER: representa o número máximo de iterações do método
   vector <LD> GaussJacobi(MATRIX& A,
                           vector<LD> b,
                           vector <LD> xinicial,
                           LD tol,
                           int MAX_ITER)
   ```

8. Implemente o método de Gauss-Seidel para a obtenção de uma aproximação para a solução de um sistema linear $Ax = b$

```
//xinicial : aproximacao inicial
//tol : representa a tolerância para que a solução seja aceita
//MAX_ITER: representa o número máximo de iterações do método
vector <LD> GaussSeidel(MATRIX& A,
                        vector<LD> b,
                        vector <LD> xinicial,
                        LD tol,
                        int MAX_ITER)
```