laboratório 01: Eliminação Gaussiana

Operações elementares

Seja M uma matriz qualquer. As seguintes operações chamadas de operações elementares sobre a matriz:

– trocar duas linhas da matriz

$$\begin{bmatrix} 1 & -2 & 2 & 1 \\ 2 & 1 & -3 & 5 \\ 4 & -7 & 1 & -1 \end{bmatrix} \xrightarrow{L_0 \leftrightarrow L_2} \begin{bmatrix} 4 & -7 & 1 & -1 \\ 2 & 1 & -3 & 5 \\ 1 & -2 & 2 & 1 \end{bmatrix}$$

– multiplicar uma linha por um número ($\neq 0$)

$$\begin{bmatrix} 1 & -2 & 2 & 1 \\ 2 & 1 & -3 & 5 \\ 4 & -7 & 1 & -1 \end{bmatrix} \xrightarrow{L_1 \leftarrow 3L_1} \begin{bmatrix} 1 & -2 & 2 & 1 \\ 6 & 3 & -9 & 15 \\ 4 & -7 & 1 & -1 \end{bmatrix}$$

– substituir uma linha pela soma dela mesma com outra multiplicada por um número ($\neq 0$)

$$\begin{bmatrix} 1 & -2 & 2 & 1 \\ 2 & 1 & -3 & 5 \\ 4 & -7 & 1 & -1 \end{bmatrix} \xrightarrow{L_1 \leftarrow L_1 + 2L_0} \begin{bmatrix} 1 & -2 & 2 & 1 \\ 3 & -1 & -1 & 6 \\ 4 & -7 & 1 & -1 \end{bmatrix}$$

tarefa 1: operações elementares

Implemente as operações elementares para matrizes quaisquer $n \times m$.

$$\begin{aligned} L_i &\leftrightarrow L_j \\ L_i &\leftarrow \alpha L_i \\ L_i &\leftarrow L_i + \alpha L_j \end{aligned}$$

Você deve criar os métodos:

```c
void trocar_linhas(int i, int j, float** M, int n, int m){
    // Troca as linhas i e j da matriz M de dimensões n × m

    ...
}

void multiplicar_escalar_linha(int i, float a, float** M, int n, int m){
    // Multiplica a linha i da matriz M de dimensões n × m

    ...
}

void somar_linhas(int i, int j, float a, float** M, int n, int m){
    // Soma à linha i o produto da linha j por a
    // na matriz M de dimensões n × m

    ...
}
```

tarefa 2: zerando uma posição abaixo

Usando operações elementares, podemos alterar os valores em uma matriz.

Considere a seguinte matriz:

$$\begin{bmatrix} \boldsymbol{a_{00}} & a_{01} & a_{02} & a_{03} \\ a_{10} & \boldsymbol{a_{11}} & a_{12} & a_{13} \\ a_{20} & a_{21} & \boldsymbol{a_{22}} & a_{23} \\ a_{30} & a_{31} & a_{32} & \boldsymbol{a_{33}} \end{bmatrix}$$

Se realizarmos a operação elementar

$$L_2 \leftarrow L_2 - \frac{a_{21}}{a_{11}} L_1$$

Obtemos uma matriz da seguinte forma:

$$\begin{bmatrix} \boldsymbol{a_{00}} & a_{01} & a_{02} & a_{03} \\ a_{10} & \textcircled{\boldsymbol{a_{11}}} & a_{12} & a_{13} \\ a'_{20} & \boxed{0} & \boldsymbol{a'_{22}} & a'_{23} \\ a_{30} & a_{31} & a_{32} & \boldsymbol{a_{33}} \end{bmatrix}$$

O elemento a_{11} da diagonal principal é chamado de pivô.

Em geral, quando temos uma matriz e uma posição (i, j) abaixo da diagonal principal, o pivô é o elemento (j, j).

Se fizermos a operação

$$L_i \leftarrow L_i - \frac{a_{ij}}{a_{jj}} L_j$$

a posição (i, j) da matriz será zerada

$$\begin{bmatrix} & \vdots & \\ \dots & \boldsymbol{a_{jj}} & \\ & \vdots & \\ & a_{ij} & \end{bmatrix} \xrightarrow{L_i \leftarrow L_i - \frac{a_{ij}}{a_{jj}} L_j} \begin{bmatrix} & \vdots & \\ \dots & \boldsymbol{a_{jj}} & \\ & \vdots & \\ & \boxed{0} & \end{bmatrix}$$

Implemente um método que, dada uma posição (i, j) da matriz que esteja abaixo, zera o elemento $A[i][j]$ utilizando o elemento o elemento da diagonal principal como pivô.

Você deve criar o método:

```
void zerar_posicao_abaixo(int i, int j, float** M, int n, int m){
    // Zera a posição M[i][j] da matriz M, onde i > j

    ...
}
```

tarefa 3: zerando coluna abaixo

Implemente um método que zera todas as posições de uma determinada coluna j que ficam abaixo da diagonal principal matriz.

Você deve criar o método:

```
void zerar_coluna_abaixo(int j, float** M, int n, int m){
    // Zera as posições da coluna j da matriz M abaixo da diagonal principal

    ...
}
```

tarefa 4: matriz triangular superior

Implemente um método que zera todas as posições abaixo da diagonal principal da matriz. (Uma matriz em que todas as posições abaixo da diagonal principal estão zeradas é chamada de triangular superior.)

Você deve criar o método:

```
void triangular_superior(int j, float** M, int n, int m){
    // Zera as posições da matriz M abaixo da diagonal principal

    ...
}
```

tarefa 5: zerando uma posição acima

Implemente um método que zera uma posição acima da diagonal principal (semelhante ao

método que zera as posições abaixo da diagonal principal).

Você deve criar o método:

```
void zerar_posicao_acima(int i, int j, float** M, int n, int m){
    // Zera a posição M[i][j] da matriz M, onde i < j

    ...
}
```

tarefa 6: zerando coluna abaixo

Implemente um método que zera todas as posições de uma determinada coluna j que ficam acima da diagonal principal matriz.

Você deve criar o método:

```
void zerar_coluna_abaixo(int j, float** M, int n, int m){
    // Zera as posições da coluna j da matriz M abaixo da diagonal principal

    ...
}
```

tarefa 7: matriz triangular inferior

Implemente um método que zera todas as posições acima da diagonal principal da matriz. (Uma matriz em que todas as posições acima da diagonal principal estão zeradas é chamada de triangular inferior.)

Você deve criar o método:

```
void triangular_inferior(float** M, int n, int m){
    // Zera as posições da matriz M acima da diagonal principal

    ...
}
```

tarefa 8: matriz diagonal

Implemente um método que transforma a matriz em uma matriz diagonal. (Uma matriz diagonal é uma matriz cujas únicas posições não nulas estão na diagonal principal.)

Você deve criar o método:

```
void diagonal(float** M, int n, int m){
    // Zera as posições da matriz M acima da diagonal principal

    ...
}
```

tarefa 9: matriz inversa

Imagine que tenhamos uma matriz, por exemplo

$$\begin{bmatrix} 1 & -2 & 2 \\ 4 & 2 & -6 \\ -4 & 7 & -1 \end{bmatrix}$$

Se expandirmos a matriz da seguinte forma:

$$\left[\begin{array}{ccc|ccc} 1 & -2 & 2 & 1 & 0 & 0 \\ 4 & 2 & -6 & 0 & 1 & 0 \\ -4 & 7 & -1 & 0 & 0 & 1 \end{array}\right]$$

E realizarmos uma sequência de operações elementares de forma que a parte da esquerda se torne uma matriz identidade

$$\left[\begin{array}{ccc|ccc} 1 & -2 & 2 & 1 & 0 & 0 \\ 4 & 2 & -6 & 0 & 1 & 0 \\ -4 & 7 & -1 & 0 & 0 & 1 \end{array}\right] \quad \rightarrow \quad \ldots \quad \rightarrow \quad \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & b_{00} & b_{01} & b_{02} \\ 0 & 1 & 0 & b_{10} & b_{11} & b_{12} \\ 0 & 0 & 1 & b_{20} & b_{21} & b_{22} \end{array}\right]$$

a parte da direita será a inversa da matriz original.

Implemente um algoritmo que calcula a inversa de uma matriz quadrada, inversível dada.

Alocação dinâmica de matriz em C

Na linguagem C, podemos alocar uma matriz M de dimensões $n \times m$ dinamicamente da seguinte maneira:

```
float** M = malloc(n*sizeof(float*));
for (int i = 0; i < n; i++){
    M[i] = malloc(m*sizeof(float));
}
```

Desta forma, a matriz M fica alocada na memória e pode ser usada normalmente:

```
M[i][j] = 3.12;
```

Uma vez criada a matriz dessa forma, o tipo dela na linguagem C é

```
float**
```

Além disso, as dimensões da matriz devem ser mantidas em variáveis, pois, uma vez criada a matriz dinamicamente, não é possível (em C padrão) recuperar as suas dimensões.

Finalmente, pode ser conveniente liberar o espaço alocado para a matriz uma vez que ela não é mais necessária. Isso pode ser feito desta forma:

```
for (int i = 0; i < n; i++){
    free(M[i]);
}
free(M);
```

Os métodos `malloc()` e `free` fazem parte da biblioteca `stdlib.h`

Alocação dinâmica de matriz em C++

A alocação dinâmica de matrizes em C++ é semelhante à forma feita em C. A principal diferênça é que em C++ usaremos as palavras reservadas `new` e `delete` ao invés dos métodos `malloc()` e `free()`.

Para alocar uma matriz M de dimensões $n \times m$, fazemos o seguinte:

```
float** M = new float*[n];
for (int i = 0; i < n; i++){
    M[i] = new float[m];
}
```

Para desalocar:

```
for (int i = 0; i < n; i++){
    delete [] M[i];
}
delete [] M;
```

Novamente, se lembre de manter as dimensões da matriz de alguma forma acessível.

Sistemas de equações lineares

Considere o seguinte sistema de equações:

$$\begin{aligned} x - 2y + 2z &= 1 \\ 2x + \ y - 3z &= 5 \\ 4x - 7y + 1z &= -1 \end{aligned} \tag{I}$$

Esse sistema dá origem à seguinte matriz:

$$\left[\begin{array}{ccc|c} 1 & -2 & 2 & 1 \\ 2 & 1 & -3 & 5 \\ 4 & -7 & 1 & -1 \end{array}\right]$$

Imagine que executamos algumas operações elementares sobre a matriz, por exemplo:

$$\begin{bmatrix} 1 & -2 & 2 & 1 \\ 2 & 1 & -3 & 5 \\ 4 & -7 & 1 & -1 \end{bmatrix} \xrightarrow{L_2 \leftarrow 3 \times L_2} \begin{bmatrix} 1 & -2 & 2 & 1 \\ 6 & 3 & -9 & 15 \\ 4 & -7 & 1 & -1 \end{bmatrix} \quad \dots \quad \begin{bmatrix} 6 & 3 & -9 & 15 \\ -2 & 4 & -4 & -2 \\ 2 & -3 & -3 & -3 \end{bmatrix}$$

A matriz obtida corresponde ao seguinte sistema:

$$\begin{aligned} 6x + 3y - 9z &= 15 \\ -2x + 4y - 4z &= -2 \\ 2x - 3y - 3z &= -3 \end{aligned} \tag{II}$$

fato: os sistemas (I) e (II) possuem as mesmas soluções.