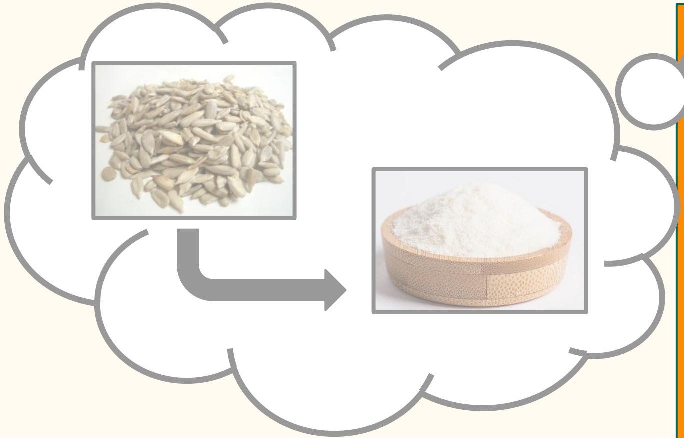
Parte 5

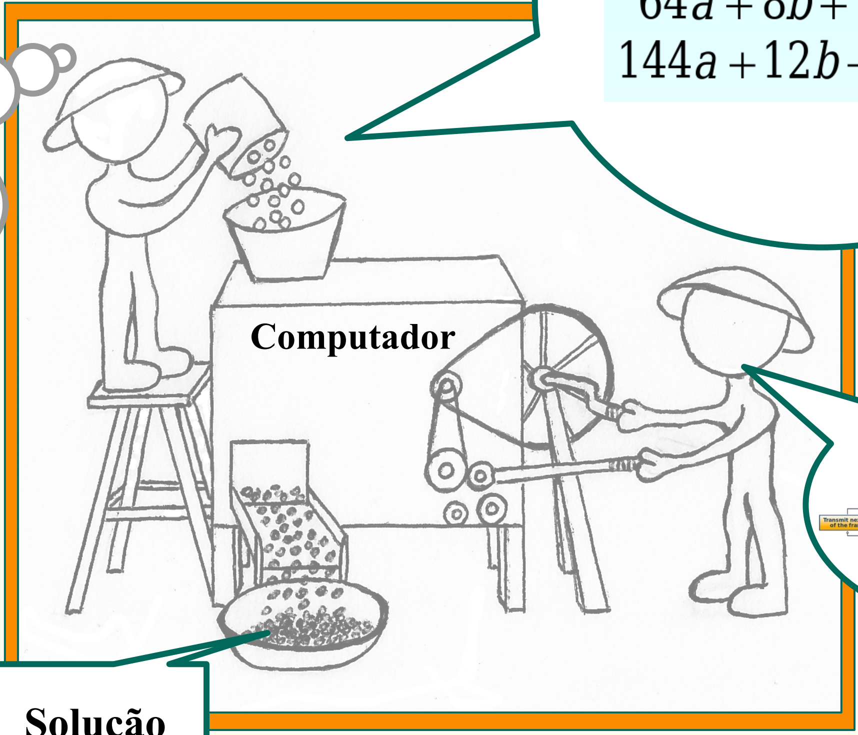
Resolução de Sistemas Lineares - 3

CI1164 - Introdução à Computação Científica
Profs. Armando Delgado e Guilherme Derenievicz
Departamento de Informática - UFPR

Visão geral da disciplina:

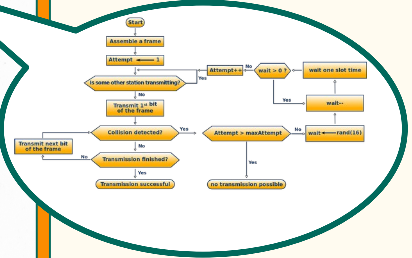


Descrição do Problema



$$25a + 5b + c = 106$$
$$64a + 8b + c = 177$$
$$144a + 12b + c = 600$$

Sistemas Lineares



Método Numérico

Solução

Resolução de Sistemas Lineares

$$A = \begin{bmatrix}
-\alpha & 0 & 0 & 1 & \alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-\alpha & 0 & -1 & 0 & -\alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -\alpha & -1 & 0 & 0 & \alpha & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \alpha & 0 & 1 & 0 & \alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -\alpha & 0 & 0 & 1 & \alpha & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\alpha & 0 & -1 & 0 & -\alpha & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & \alpha & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -\alpha & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\alpha & -1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\alpha & -1
\end{bmatrix}$$

Resolução de Sistemas Lineares

$$
A = \begin{bmatrix}
-\alpha & 0 & 0 & 1 & \alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-\alpha & 0 & -1 & 0 & -\alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -\alpha & -1 & 0 & 0 & \alpha & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \alpha & 0 & 1 & 0 & \alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -\alpha & 0 & 0 & 1 & \alpha & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\alpha & 0 & -1 & 0 & -\alpha & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & \alpha & 0 \\
 & & & & & & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -\alpha & 0 \\
 & & & & & & & 0 & 0 & 0 & 0 & 0 & -\alpha & -1 & 0 & 0 & 1 \\
 & & & & & & & 0 & 0 & 0 & 0 & 0 & \alpha & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\alpha & -1
\end{bmatrix}
$$

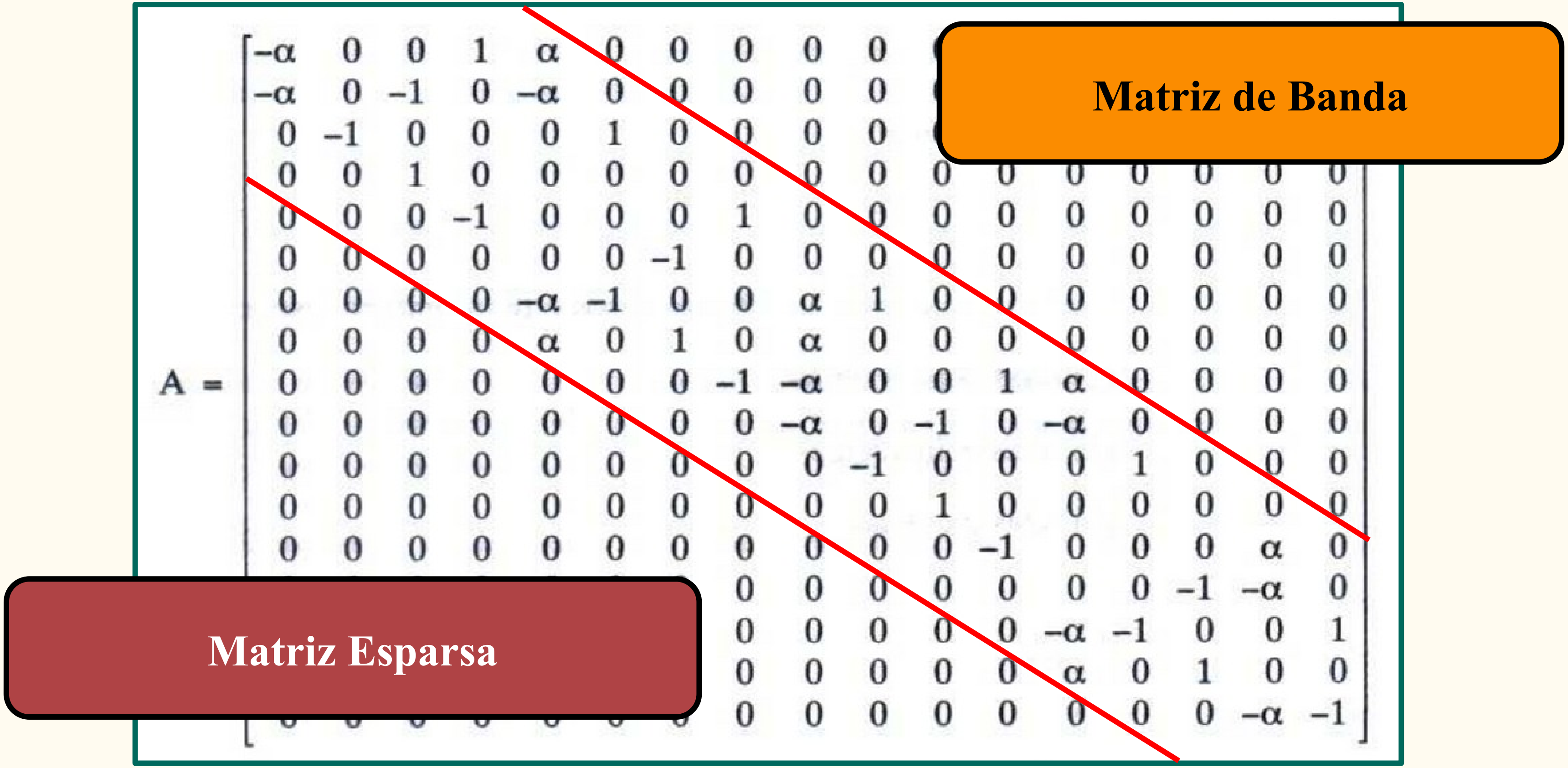
Matriz Esparsa

Resolução de Sistemas Lineares

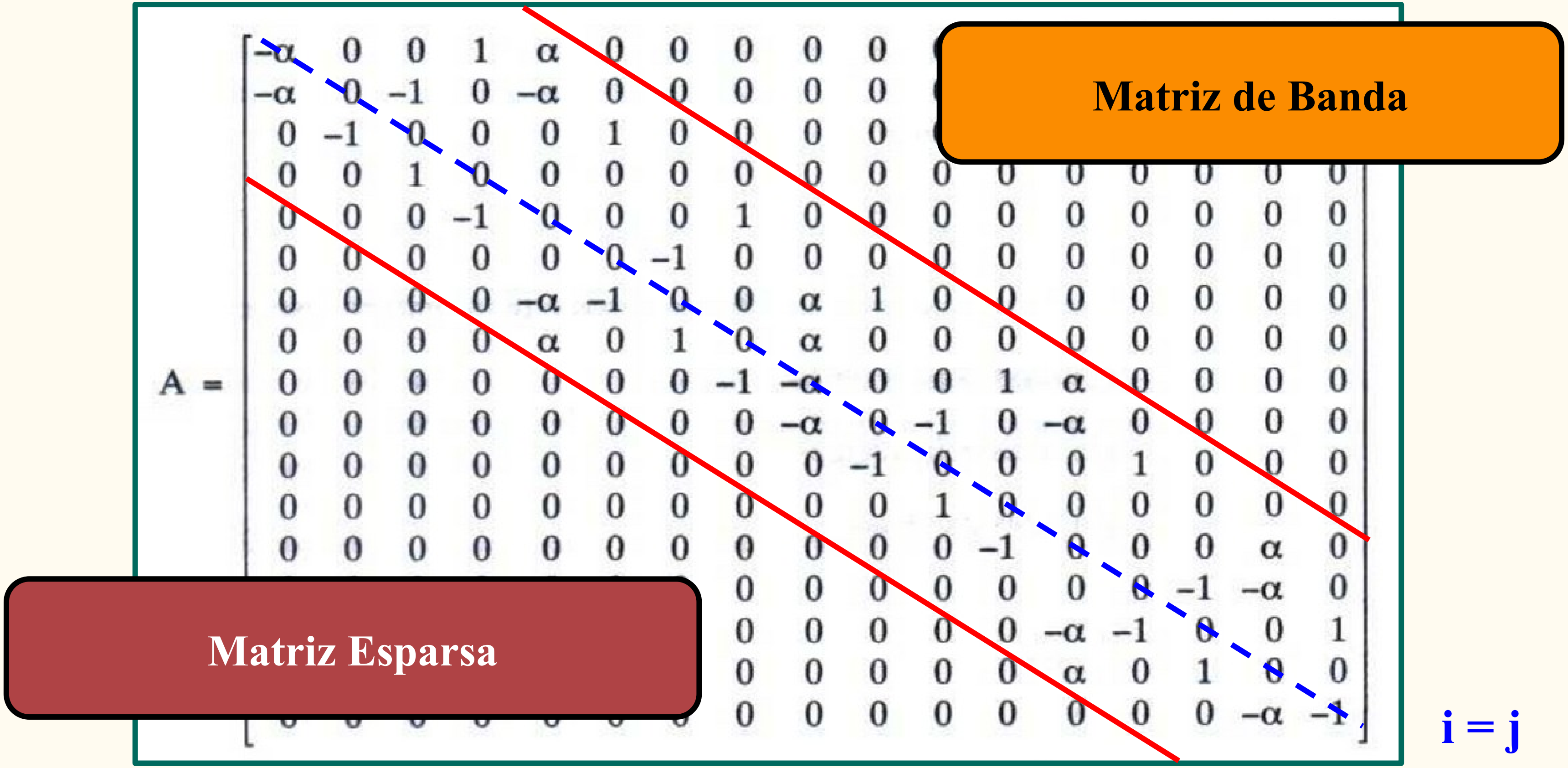
$$
A = \begin{bmatrix}
-\alpha & 0 & 0 & 1 & \alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-\alpha & 0 & -1 & 0 & -\alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -\alpha & -1 & 0 & 0 & \alpha & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \alpha & 0 & 1 & 0 & \alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -\alpha & 0 & 0 & 1 & \alpha & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\alpha & 0 & -1 & 0 & -\alpha & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & \alpha & 0 \\
 & & & & & & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -\alpha & 0 \\
 & & & & & & & 0 & 0 & 0 & 0 & 0 & -\alpha & -1 & 0 & 0 & 1 \\
 & & & & & & & 0 & 0 & 0 & 0 & 0 & \alpha & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\alpha & -1
\end{bmatrix}
$$

Matriz Esparsa

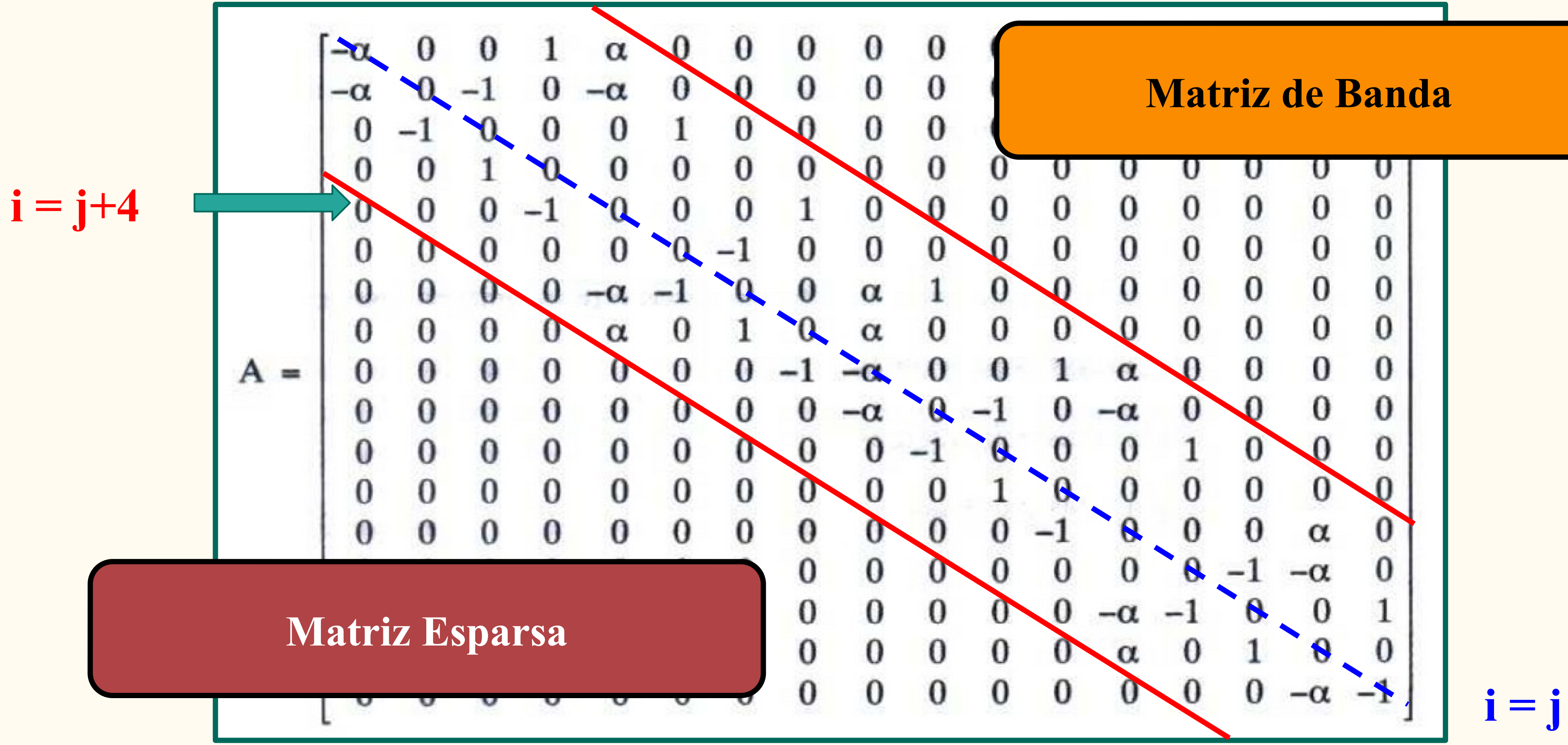
Matrizes k-Diagonais



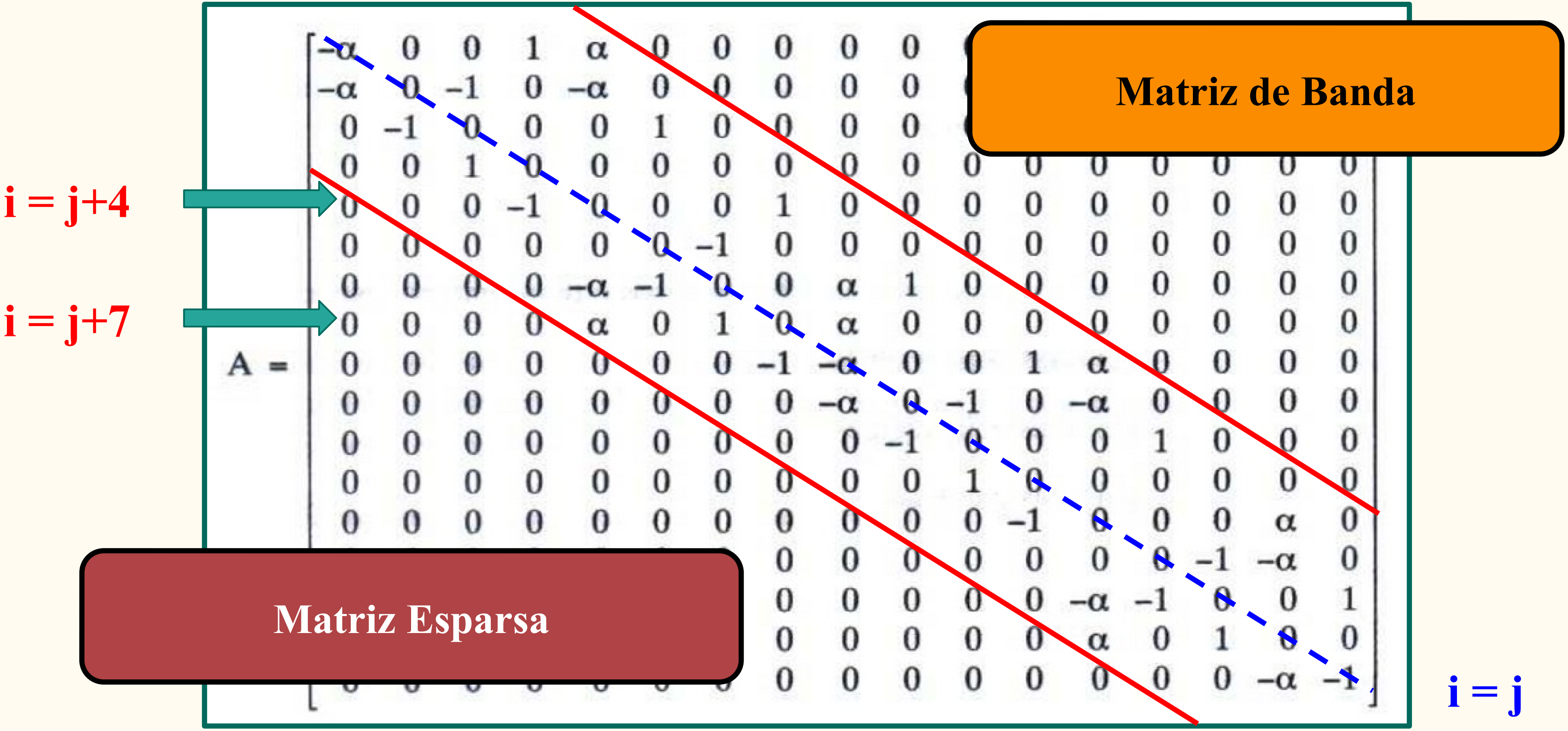
Matrizes k-Diagonais



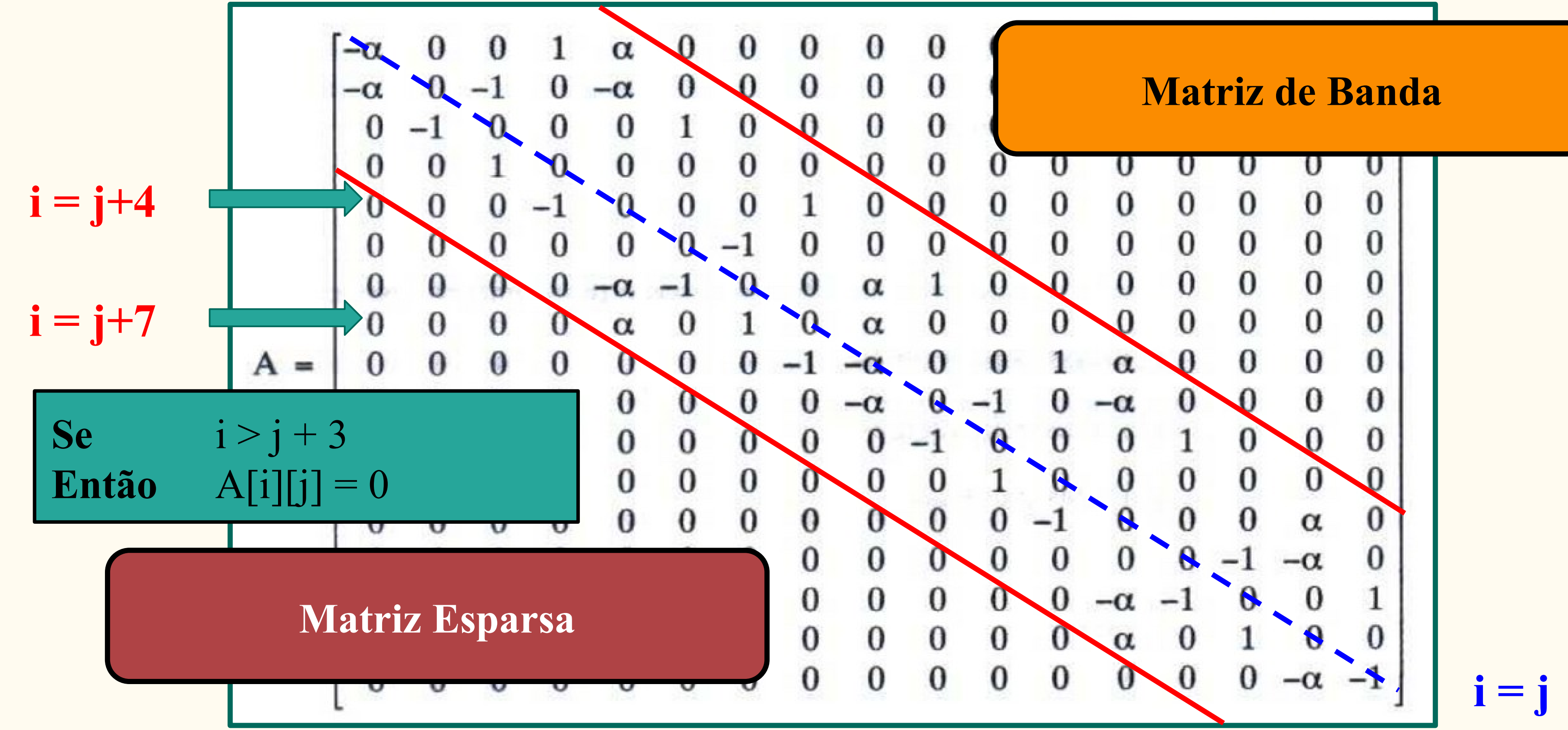
Matrizes k-Diagonais



Matrizes k-Diagonais



Matrizes k-Diagonais



Se $i > j + 3$
Então $A[i][j] = 0$

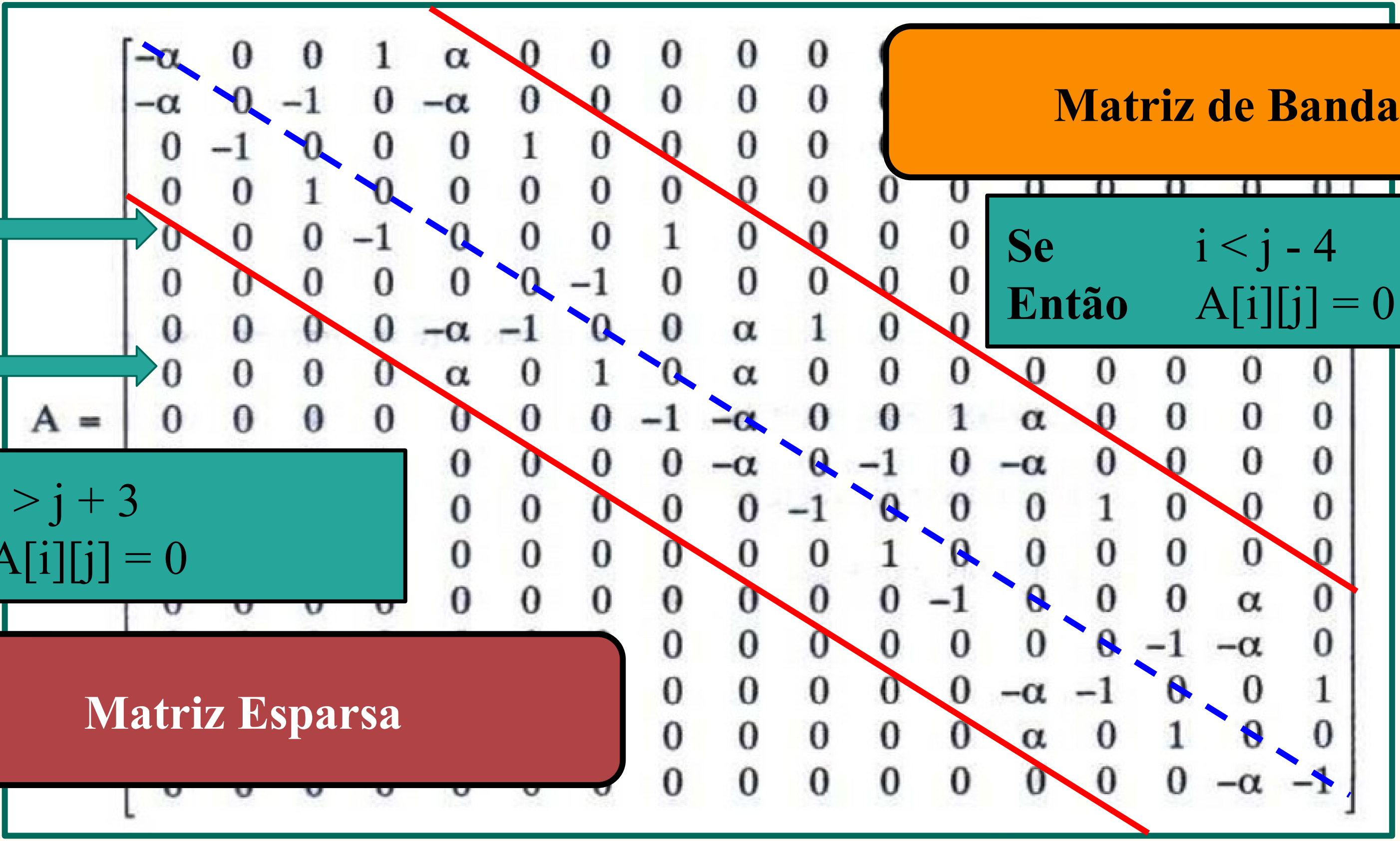
Matrizes k-Diagonais

Matriz de Banda

Se $i < j - 4$
Então $A[i][j] = 0$

$i = j+4$

$i = j+7$

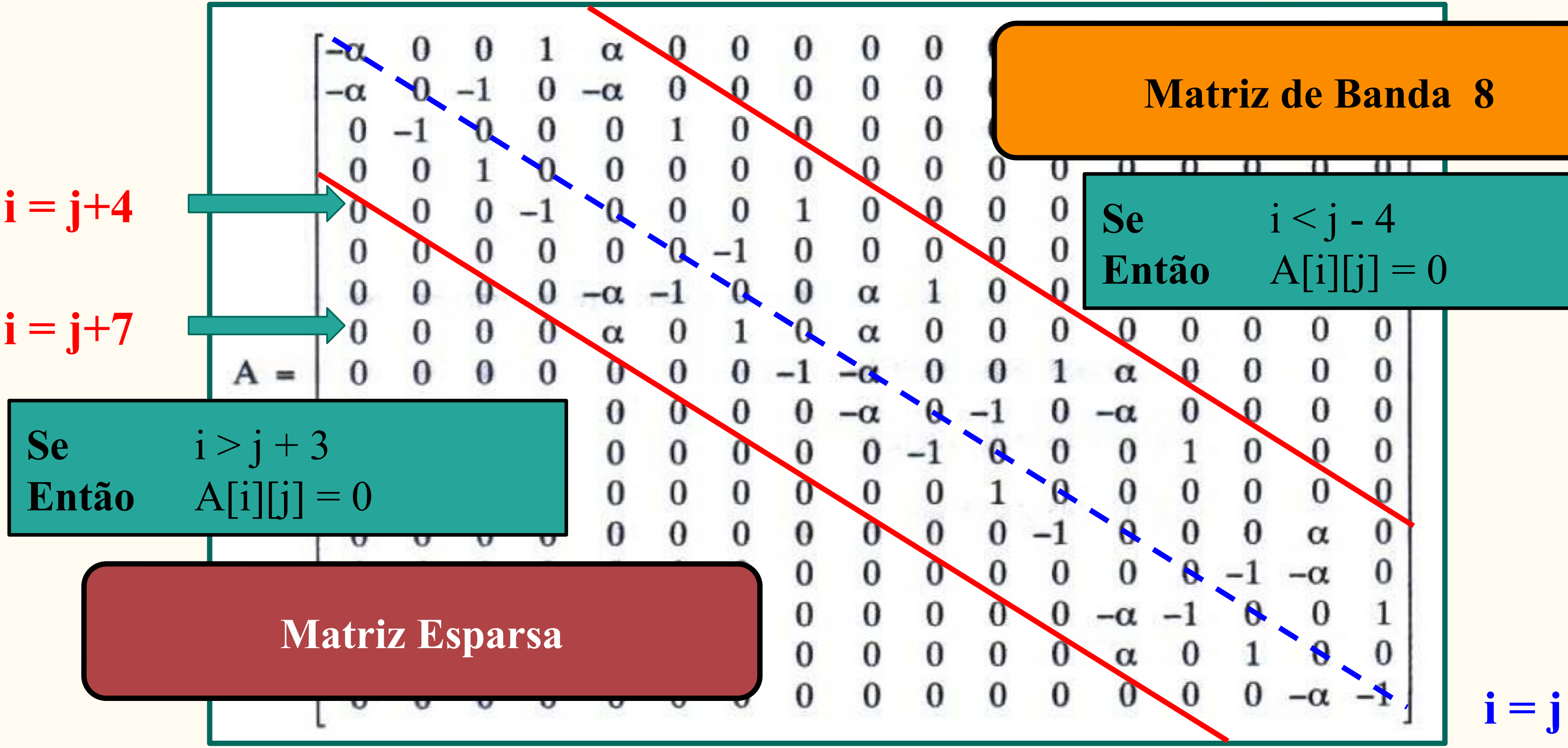


Se $i > j + 3$
Então $A[i][j] = 0$

Matriz Esparsa

$i = j$

Matrizes k-Diagonais



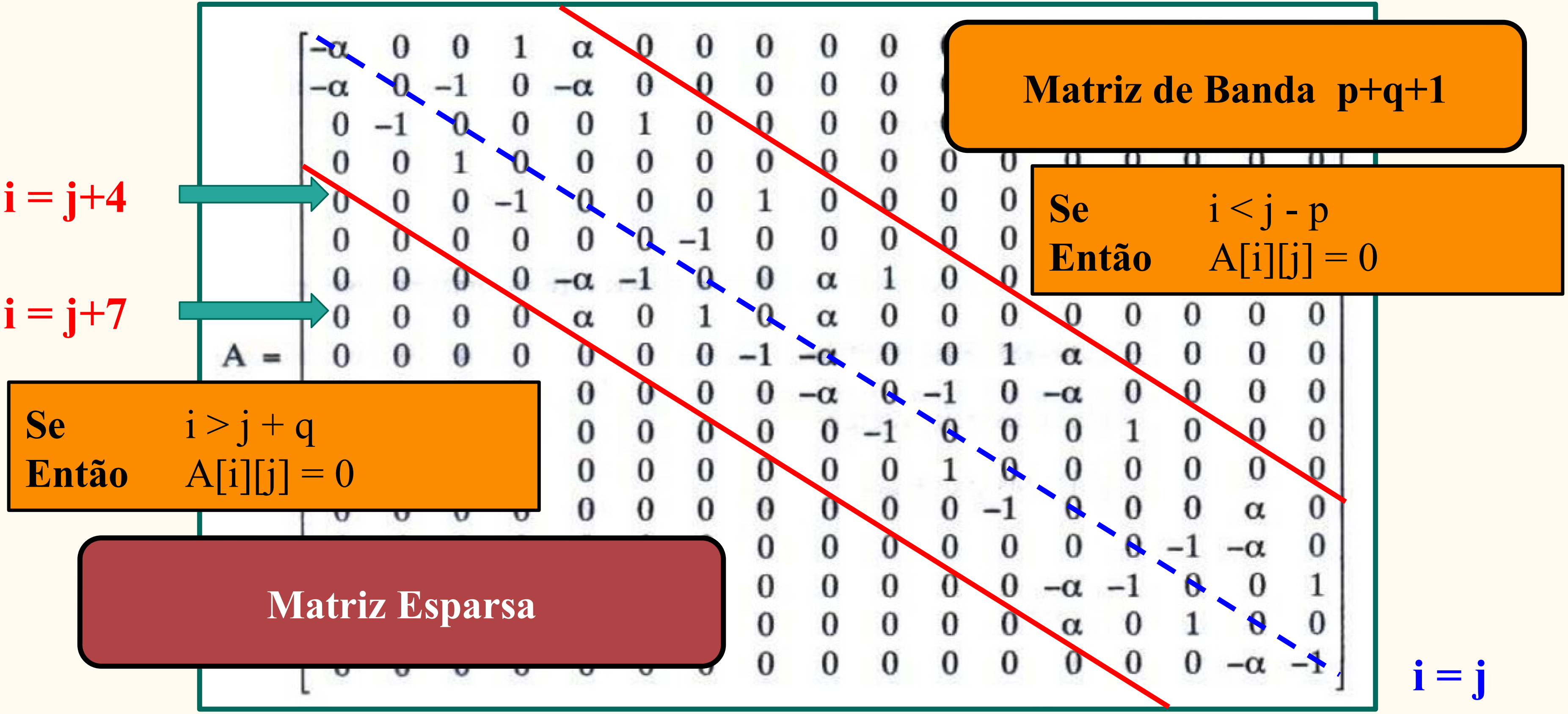
Matriz de Banda 8

Se $i < j - 4$
Então $A[i][j] = 0$

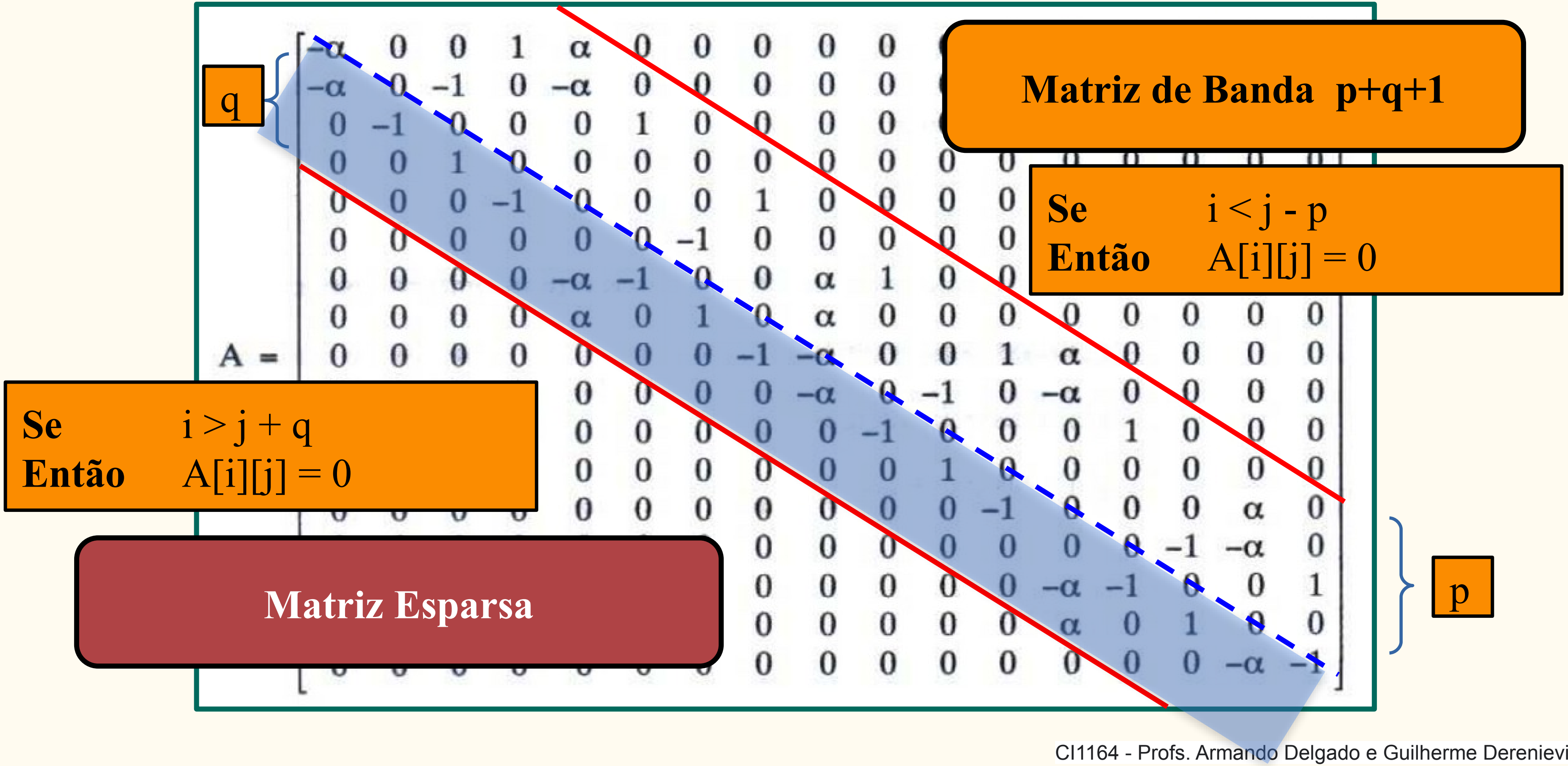
Se $i > j + 3$
Então $A[i][j] = 0$

Matriz Esparsa

Matrizes k-Diagonais



Matrizes k-Diagonais



Simplificação nos métodos

- Armazenamento dos elementos da matriz em memória
  - ➔ Basta as diagonais
- Implementação dos métodos ficam mais simples
  - ➔ Menos laços aninhados
  - ➔ Menos operações em ponto flutuante

Matrizes Tridiagonais (p = q = 1)

$$\begin{bmatrix} d_1 & c_1 & 0 & 0 & 0 \\ a_1 & d_2 & c_2 & 0 & 0 \\ 0 & a_2 & d_3 & c_3 & 0 \\ 0 & 0 & a_3 & d_4 & c_4 \\ 0 & 0 & 0 & a_4 & d_5 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}$$

Matrizes Tridiagonais (p = q = 1)

$$\begin{bmatrix} d_1 & c_1 & 0 & 0 & 0 \\ a_1 & d_2 & c_2 & 0 & 0 \\ 0 & a_2 & d_3 & c_3 & 0 \\ 0 & 0 & a_3 & d_4 & c_4 \\ 0 & 0 & 0 & a_4 & d_5 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}$$

```
/* Seja um S.L. de ordem 'n'
*/
void eliminacaoGauss( double **A, double *b, u
   /* para cada linha a partir da primeira */
   for (int i=0; i < n; ++i) {
      for(int k=i+1; k < n; ++k) {
         double m = A[k][i] / A[i][i];
         A[k][i] = 0.0;
         for(int j=i+1; j < n; ++j)
            A[k][j] -= A[i][j] * m;
         b[k] -= b[i] * m;
      }
   }
}
```

Matrizes Tridiagonais (p = q = 1)

$$\begin{bmatrix} d_1 & c_1 & 0 & 0 & 0 \\ a_1 & d_2 & c_2 & 0 & 0 \\ 0 & a_2 & d_3 & c_3 & 0 \\ 0 & 0 & a_3 & d_4 & c_4 \\ 0 & 0 & 0 & a_4 & d_5 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}$$

```
/* Seja um S.L. de ordem 'n'
*/
void eliminacaoGauss( double **A, double *b, u
   /* para cada linha a partir da primeira */
   for (int i=0; i < n; ++i) {
      for(int k=i+1; k < n; ++k) {
         double m = A[k][i] / A[i][i];
         A[k][i] = 0.0;
         for(int j=i+1; j < n; ++j)
            A[k][j] -= A[i][j] * m;
         b[k] -= b[i] * m;
      }
   }
}
```

Matrizes Tridiagonais (p = q = 1)

$$\begin{bmatrix} d_1 & c_1 & 0 & 0 & 0 \\ a_1 & d_2 & c_2 & 0 & 0 \\ 0 & a_2 & d_3 & c_3 & 0 \\ 0 & 0 & a_3 & d_4 & c_4 \\ 0 & 0 & 0 & a_4 & d_5 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}$$

```
/* Seja um S.L. de ordem 'n'
*/
void eliminacaoGauss( double **A, double *b, u
   /* para cada linha a partir da primeira */
   for (int i=0; i < n; ++i) {
      for(int k=i+1; k < n; ++k) {
         double m = A[k][i] / A[i][i];
         A[k][i] = 0.0;
         for(int j=i+1; j < n; ++j)
            A[k][j] -= A[i][j] * m;
         b[k] -= b[i] * m;
      }
   }
}
```

Matrizes Tridiagonais (p = q = 1)

$$\begin{bmatrix} d_1 & c_1 & 0 & 0 & 0 \\ a_1 & d_2 & c_2 & 0 & 0 \\ 0 & a_2 & d_3 & c_3 & 0 \\ 0 & 0 & a_3 & d_4 & c_4 \\ 0 & 0 & 0 & a_4 & d_5 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}$$

```
/* Seja um S.L. de ordem 'n'
*/
void eliminacaoGauss( double **A, double *b, u
   /* para cada linha a partir da primeira */
   for (int i=0; i < n; ++i) {
      for(int k=i+1; k < n; ++k) {
         double m = A[k][i] / A[i][i];
         A[k][i] = 0.0;
         for(int j=i+1; j < n; ++j)
            A[k][j] -= A[i][j] * m;
         b[k] -= b[i] * m;
      }
   }
}
```

k = i+1

Matrizes Tridiagonais (p = q = 1)

$$\begin{bmatrix} d_1 & c_1 & 0 & 0 & 0 \\ a_1 & d_2 & c_2 & 0 & 0 \\ 0 & a_2 & d_3 & c_3 & 0 \\ 0 & 0 & a_3 & d_4 & c_4 \\ 0 & 0 & 0 & a_4 & d_5 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}$$

```
/* Seja um S.L. de ordem 'n'
*/
void eliminacaoGauss( double **A, double *b, u
   /* para cada linha a partir da primeira */
   for (int i=0; i < n; ++i) {
      for(int k=i+1; k < n; ++k) {
         double m = A[k][i] / A[i][i];
         A[k][i] = 0.0;
         for(int j=i+1; j < n; ++j)
            A[k][j] -= A[i][j] * m;
         b[k] -= b[i] * m;
      }
   }
}
```

k = i+1

Matrizes Tridiagonais (p = q = 1)

$$\begin{bmatrix} d_1 & c_1 & 0 & 0 & 0 \\ a_1 & d_2 & c_2 & 0 & 0 \\ 0 & a_2 & d_3 & c_3 & 0 \\ 0 & 0 & a_3 & d_4 & c_4 \\ 0 & 0 & 0 & a_4 & d_5 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}$$

```
/* Seja um S.L. de ordem 'n'
*/
void eliminacaoGauss( double **A, double *b, u
  /* para cada linha a partir da primeira */
  for (int i=0; i < n; ++i) {
    for(int k=i+1; k < n; ++k) {
        double m = A[k][i] / A[i][i];
        A[k][i] = 0.0;
        for(int j=i+1; j < n; ++j)
            A[k][j] -= A[i][j] * m;
        b[k] -= b[i] * m;
    }
  }
}
```

k = i+1

j = i+1

Matrizes Tridiagonais (p = q = 1)

$$\begin{bmatrix} d_1 & c_1 & 0 & 0 & 0 \\ a_1 & d_2 & c_2 & 0 & 0 \\ 0 & a_2 & d_3 & c_3 & 0 \\ 0 & 0 & a_3 & d_4 & c_4 \\ 0 & 0 & 0 & a_4 & d_5 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}$$

```
double b[5];
double d[5];
double a[4];
double c[4];
```

```
/* Seja um S.L. de ordem 'n'
*/
void eliminacaoGauss( double **A, double *b, u
   /* para cada linha a partir da primeira */
   for (int i=0; i < n; ++i) {
      for(int k=i+1; k < n; ++k) {          k = i+1
         double m = A[k][i] / A[i][i];
         A[k][i] = 0.0;
         for(int j=i+1; j < n; ++j)          j = i+1
            A[k][j] -= A[i][j] * m;
         b[k] -= b[i] * m;
      }
   }
}
```

Matrizes Tridiagonais (p = q = 1)

$$\begin{bmatrix} d_1 & c_1 & 0 & 0 & 0 \\ a_1 & d_2 & c_2 & 0 & 0 \\ 0 & a_2 & d_3 & c_3 & 0 \\ 0 & 0 & a_3 & d_4 & c_4 \\ 0 & 0 & 0 & a_4 & d_5 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}$$

```
double b[5];
double d[5];
double a[4];
double c[4];
```

```
double *d, double *a, double *c
```

```
/* Seja um S.L. de ordem
*/
void eliminacaoGauss( double **A, double *b, u
   /* para cada linha a partir da primeira */
   for (int i=0; i < n; ++i) {
      for(int k=i+1; k < n; ++k) {
         double m = A[k][i] / A[i][i];
         A[k][i] = 0.0;
         for(int j=i+1; j < n; ++j)
            A[k][j] -= A[i][j] * m;
         b[k] -= b[i] * m;
      }
   }
}
```

Matrizes Tridiagonais (p = q = 1)

$$\begin{bmatrix} d_1 & c_1 & 0 & 0 & 0 \\ a_1 & d_2 & c_2 & 0 & 0 \\ 0 & a_2 & d_3 & c_3 & 0 \\ 0 & 0 & a_3 & d_4 & c_4 \\ 0 & 0 & 0 & a_4 & d_5 \end{bmatrix} \begin{bmatrix} b_1 \\ \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}$$

```
double *d, double *a, double *c
```

```
double m = a[i] / d[i];
```

```
                   ordem
void elimi      Gauss( double **A, double *b, u
   /* para c    linha a partir da primeira */
   for (int i=0   i < n; ++i) {
      for(int k=i+1; k < n; ++k) {
         double m = A[k][i] / A[i][i];
         A[k][i] = 0.0;
         for(int j=i+1; j < n; ++j)
            A[k][j] -= A[i][j] * m;
         b[k] -= b[i] * m;
      }
   }
}
```

```
double b[5];
double d[5];
double a[4];
double c[4];
```

Matrizes Tridiagonais (p = q = 1)

$$\begin{bmatrix} \boxed{d_1} & c_1 & 0 & 0 & 0 \\ a_1 & d_2 & c_2 & 0 & 0 \\ 0 & a_2 & d_3 & c_3 & 0 \\ 0 & 0 & a_3 & d_4 & c_4 \\ 0 & 0 & 0 & a_4 & d_5 \end{bmatrix} \begin{bmatrix} b_1 \\ \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}$$

```
double b[5];
double d[5];
double a[4];
double c[4];
```

```
double *d, double *a, double *c
```

```
double m = a[i] / d[i];
```

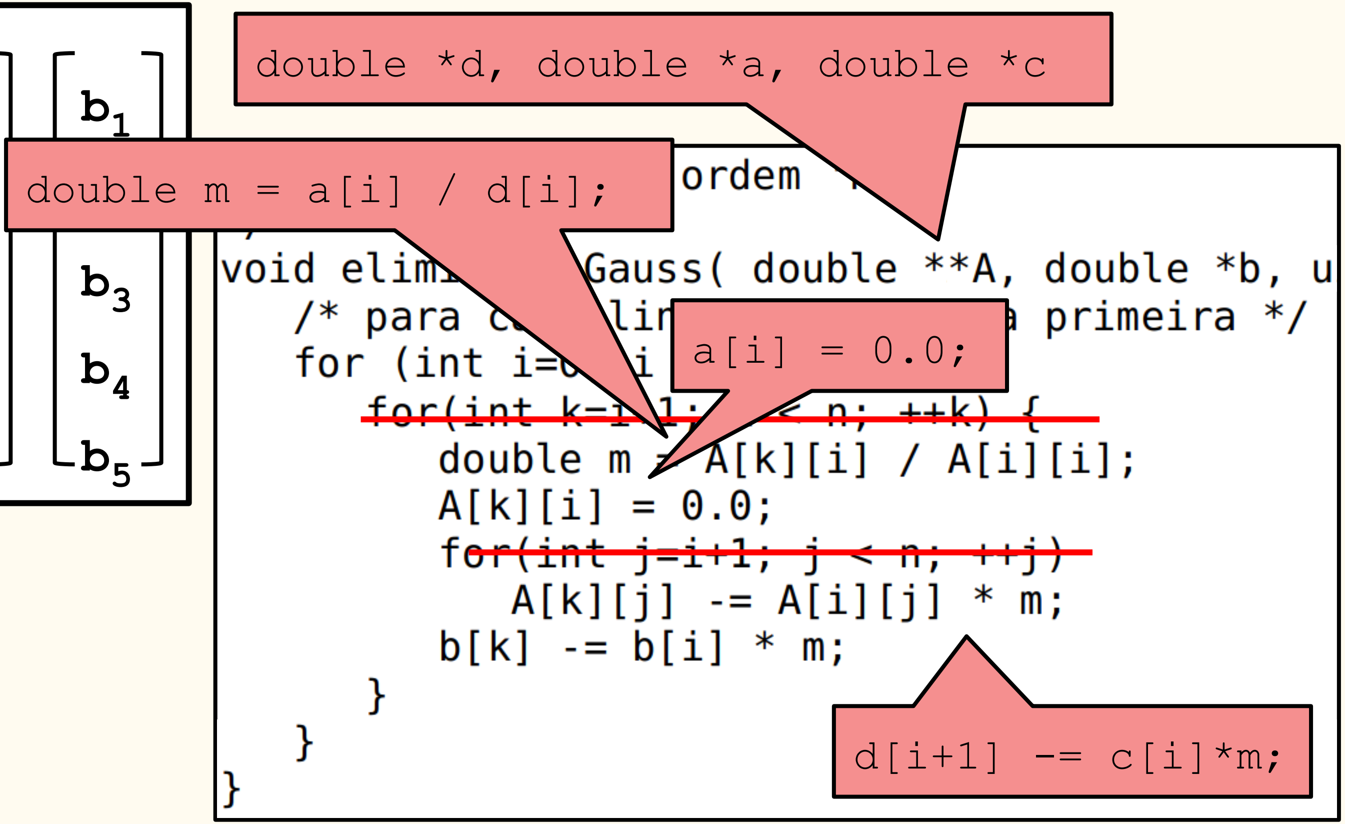
```
a[i] = 0.0;
```

```
                    ordem
void elimi          Gauss( double **A, double *b, u
  /* para c      lin              a primeira */
  for (int i=0      i
    for(int k=i+1;      < n; ++k) {
      double m = A[k][i] / A[i][i];
      A[k][i] = 0.0;
      for(int j=i+1; j < n; ++j)
        A[k][j] -= A[i][j] * m;
      b[k] -= b[i] * m;
    }
  }
}
```

Matrizes Tridiagonais (p = q = 1)

$$
\begin{bmatrix}
d_1 & c_1 & 0 & 0 & 0 \\
a_1 & d_2 & c_2 & 0 & 0 \\
0 & a_2 & d_3 & c_3 & 0 \\
0 & 0 & a_3 & d_4 & c_4 \\
0 & 0 & 0 & a_4 & d_5
\end{bmatrix}
\begin{bmatrix}
b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5
\end{bmatrix}
$$

```c
double b[5];
double d[5];
double a[4];
double c[4];
```



Matrizes Tridiagonais (p = q = 1)

$$\begin{bmatrix} d_1 & c_1 & 0 & 0 & 0 \\ a_1 & d_2 & c_2 & 0 & 0 \\ 0 & a_2 & d_3 & c_3 & 0 \\ 0 & 0 & a_3 & d_4 & c_4 \\ 0 & 0 & 0 & a_4 & d_5 \end{bmatrix} \begin{bmatrix} b_1 \\ \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}$$

double *d, double *a, double *c

double m = a[i] / d[i];

a[i] = 0.0;

```
ordem *
/
void elimi     Gauss( double **A, double *b, u
  /* para c      lin          a primeira */
  for (int i=0   i
    for(int k=i+1;    < n; ++k) {
      double m = A[k][i] / A[i][i];
      A[k][i] = 0.0;
      for(int j=i+1; j < n; ++j)
        A[k][j] -= A[i][j] * m;
      b[k] -= b[i] * m;
    }
  }
}
```

b[i+1] -= b[i]*m;

d[i+1] -= c[i]*m;

double b[**5**];
double d[**5**];
double a[**4**];
double c[**4**];

SL Tridiagonais ➔ Eliminação de Gauss

$$\begin{bmatrix} d_0 & c_0 & 0 & 0 & 0 \\ a_0 & d_1 & c_1 & 0 & 0 \\ 0 & a_1 & d_2 & c_2 & 0 \\ 0 & 0 & a_2 & d_3 & c_3 \\ 0 & 0 & 0 & a_3 & d_4 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}$$

```
double b[5];
double d[5];
double a[4];
double c[4];
```

```
void eliminacaoGauss(double *d, double *a,
                     double *c, double *b, double *x,
                     uint n)
{
   // TRIANGULARIZAÇÃO:  5(n-1) operações
   for (int i=0; i < n-1; ++i) {
      double m = a[ i ] / d[ i ];
      a[ i ] = 0.0;
      d[ i+1 ] -= c[ i ] * m;
      b[ i+1 ] -= b[ i ] * m;
   }
   // RETROSUBSTITUIÇÃO: ≈ 3n operações (n grande)
   x[ n-1 ] = b[ n-1 ] / d[ n-1 ];
   for (int i=n-2; i >= 0; --i)
      x[ i ] = (b[ i ] - c[ i ] * x[ i+1 ]) / d[ i ];
}
```

SL Tridiagonais ➔ Gauss-Seidel

$$\begin{bmatrix} d_0 & c_0 & 0 & 0 & 0 \\ a_0 & d_1 & c_1 & 0 & 0 \\ 0 & a_1 & d_2 & c_2 & 0 \\ 0 & 0 & a_2 & d_3 & c_3 \\ 0 & 0 & 0 & a_3 & d_4 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}$$

```
double b[5];
double d[5];
double a[4];
double c[4];
```

```
void gaussSeidel (double *d, double *a, double *c,
                  double *b, double *x, uint n, double tol)
{
  double erro = 1.0 + tol;
   while (erro < tol) {
     // 5(n-2)+6 ≈ 5n operações / iteração

     X[ 0 ] = (b[ 0 ] – c[ 0 ] * x[ 1 ]) / d[ 0 ];

     for (int i=1; i < n-1; ++i)
       X[ i ] = (b[ i ] – a[ i-1 ] * x[ i-1] – c[ i ] * x[ i+1 ]) / d[ i ];

     X[ n-1 ] = (b[ n-1 ] – a[ n-2 ] * x[ n-2 ] ) / d[ n-1 ];

     // Calcula erro
   }
}
```

SL k-diagonais com valores parametrizados

- Em alguns tipos de problemas, os valores das diagonais podem ser parametrizados:

  ➔ Equações diferenciais ordinárias ou parciais
    - Método de Diferenças Finitas

  ➔ Sistemas não-lineares → Matriz de Broyden

- Não é necessário alocar vetores para as diagonais
  ➔ Os valores são calculados durante a execução do método

SL Tridiagonais com valores parametrizados

Comuns na solução de Equações Diferenciais Ordinárias por Diferenças Finitas

$$\begin{bmatrix} d_0 & c_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ a_0 & d_1 & c_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & a_1 & d_2 & c_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & a_2 & d_3 & c_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & a_3 & d_4 & c_4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & a_4 & d_5 & c_5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & a_5 & d_6 & c_6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & a_6 & d_7 & c_7 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_7 & d_8 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \\ b_8 \end{bmatrix}$$

- $d_i \rightarrow 2\ h^2$
- $a_i \rightarrow 2+h$
- $c_i \rightarrow 2-h$
- $b_i \rightarrow 2\ h\ f(i)$

SL Tridiagonais ➔ Gauss-Seidel

```
void gaussSeidel (double *d, double *a, double *c,
                  double *b, double *x, uint n, double tol)
{
   ......
   X[ 0 ] = (b[ 0 ] – c[ 0 ] * x[ 1 ]) / d[ 0 ];

   for (int i=1; i < n-1; ++i) {
      X[ i ] = (b[ i ] - a[ i-1 ] * x[ i-1] – c[ i ] * x[ i+1 ]) / d[ i ];
   }
   X[ n-1 ] = (b[ n-1 ] - a[ n-2 ] * x[ n-2] ) / d[ n-2 ];
   .....
}
```

double h

2hf(0)

2-h

$2h^2$

2-h

2hf(i)

2+h

$2h^2$

2hf(n-1)

2+h

$2h^2$

- $\mathbf{d_i} \rightarrow 2\ h^2$
- $\mathbf{a_i} \rightarrow 2+h$
- $\mathbf{c_i} \rightarrow 2-h$
- $\mathbf{b_i} \rightarrow 2\ h\ f(i)$

SL Tridiagonais ➔ Gauss-Seidel

```
void gaussSeidel (double h, double *x, uint n, double tol)
{
    double d = 2*h*h,   a = 2+h,   c = 2-h;

    ……
    X[ 0 ] = (2*h*f(0) – c * x[ 1 ]) / d;

    for (int i=1; i < n-1; ++i) {
       X[ i ] = (2*h*f(i) - a * x[ i-1] – c * x[ i+1 ]) / d;
    }

    X[ n-1 ] = (2*h*f(n-1) - a * x[ n-2] ) / d;
   …….
}
```

- $d_i \rightarrow 2\ h^2$
- $a_i \rightarrow 2+h$
- $c_i \rightarrow 2-h$
- $b_i \rightarrow 2\ h\ f(i)$

Referências

- Leituras Complementares do Tópico no Moodle

- Daniel Weingaertner; notas de aula da disciplina **Introdução à Computação Científica** (UFPR/DINF)

- M. Cristina C. Cunha; **Métodos Numéricos**. Editora Unicamp.

- A. Kaw, E. Kalu; **Numerical Methods with Applications**. Disponível em https://nm.mathforcollege.com/textbook-numerical-methods-with-applications/

Créditos

Este documento é de autoria do Prof. Armando Luiz N. Delgado (UFPR/DINF), para uso na disciplina Introdução à Computação Científica (CI1164).