UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULADE DE ENGENHARIA MECÂNICA

RELATÓRIO DE ALGORITMO IMPLEMENTADO
MÉTODO DIRETOS PARA RESOLUÇÃO DE SISTEMAS LINEARES COM ELIMINAÇÃO DE GAUSS COM PIVOTAMENTO PARCIAL COM ESCALA

Aluno: Diego Fernando Luque Martin

Matrícula: 191606

Campinas

2017

SUMÁRIO

1. INTRODUÇÃO

Este relatório tem por fundamento a apresentação do Método de Eliminação de Gauss com Pivotamento Parcial com Escala para resolução de Sistema de Equações Lineares. Será estudado também a título de teste o sistema de Pivotamento Completo.

2. MÉTODO

Para resolução de um sistema de equações lineares, devemos trabalhar com uma série de substituições sucessivas, onde mais comumente, utilizamos o recurso de uso de uma matriz de n linhas e m colunas atrelado ao valor dos coeficientes e um vetor (que pode ser eliminado) com as incógnitas, vide representação abaixo:

$$\mathrm{E_1} = \mathrm{a_{11}x_1} + \mathrm{a_{12}x_2} + \cdots + \mathrm{a_{1n}x_n} = \mathrm{a_{1,n+1}}$$

$$\mathrm{E_2} = \mathrm{a_{21}x_1} + \mathrm{a_{22}x_2} + \cdots + \mathrm{a_{2n}x_n} = \mathrm{a_{2,n+1}}$$

...

$$\mathrm{E_n} = \mathrm{a_{n1}x_1} + \mathrm{a_{n2}x_2} + \cdots + \mathrm{a_{nn}x_n} = \mathrm{a_{1,n+1}}$$

$$\mathrm{A} = [\mathrm{a_{ij}}] = \begin{bmatrix} \mathrm{a_{11}} & \mathrm{a_{12}} & \cdots & \mathrm{a_{1n}} \\ \mathrm{a_{21}} & \mathrm{a_{22}} & \dots & \mathrm{a_{2n}} \\ \vdots & \vdots & & \vdots \\ \mathrm{a_{n1}} & \mathrm{a_{n2}} & \dots & \mathrm{a_{nn}} \end{bmatrix} \text{ e } \mathrm{B} = \begin{bmatrix} \mathrm{b_1} \\ \mathrm{b_2} \\ \vdots \\ \mathrm{b_n} \end{bmatrix}$$

$$[\mathrm{A}, \mathrm{b}] = \begin{bmatrix} \mathrm{a_{11}} & \mathrm{a_{12}} & \cdots & \mathrm{a_{1n}} & \vdots & \mathrm{b_1} \\ \mathrm{a_{21}} & \mathrm{a_{22}} & \dots & \mathrm{a_{2n}} & \vdots & \mathrm{b_2} \\ \vdots & \vdots & & \vdots & \vdots & \vdots \\ \mathrm{a_{n1}} & \mathrm{a_{n2}} & \dots & \mathrm{a_{nn}} & \vdots & \mathrm{b_n} \end{bmatrix}$$

Entretanto, o que pode ser notado quando resolvido os sistemas de equação puramente conforme método acima, é que há um erro proveniente das sucessivas substituições devido à troca de linha de matrizes. Seguindo esta linha de raciocínio, foi desenvolvido um método capaz de pivotamento, isto é, troca de linha de forma que o pivô será um valor com módulo superior a 1, visando a redução do erro citado.

Com base nisto, chegamos ao Método de Eliminação de Gauss com Pivotamento Parcial com Escala que consiste, basicamente, o cálculo do fator da escala de cada linha através da seguinte equação:

$$s_i = \max_{1 \le j \le n} |a_{ij}|$$

Sabendo que se para algum i tivermos $s_i = 0$, então o sistema não terá solução única, trabalharemos apenas com todos os casos exceto esses. Desta forma, escolheremos a troca de linha apropriada escolhendo-se o menor número inteiro *p* com:

$$\frac{|a_{p1}|}{s_p} = \max_{1 \le k \le n} \frac{|a_{k1}|}{s_k}$$

Executando a troca das linhas por (E_1) <=> (E_p), onde o intuito da mudança de escala é garantir que o maior elemento em cada linha tenha um módulo relativo 1

2.1. Algoritmo

O livro Analise Numérica (Burden, Análise Numérica, 2015), fornece o seguinte algoritmo (combinando os algoritmos 6.2 e 6.3):

ENTRADA: número de incógnitas e equações n; matriz aumentada A = [a_{ij}] em que $1 \leq i \leq n$ e $1 \leq j \leq n + 1$.

SAÍDA: solução x_1, ..., x_n, ou mensagem que o sistema linear não tem solução única.

Passo 1: Para i = 1, ..., n, faça $s_i = \max_{1 \leq j \leq n} |a_{ij}|$;
Se s_i = 0, então SAÍDA ('Não existe solução única');
PARE.
Senão faça NLINHA(i) = i.

Passo 2: Para i = 1, ..., n-1 execute Passos 3 a 6. (Processo de Eliminação.)

Passo 3: Faça p ser o menor número inteiro com $i \leq p \leq n$ e

$$\frac{|a(\mathrm{NLINHA}(p), i)|}{s(\mathrm{NLINHA}(p))} = \max_{1 \leq j \leq n} \frac{|a(\mathrm{NLINHA}(j), i)|}{s(\mathrm{NLINHA}(j))}$$

Passo 4: Se $|a(\mathrm{NLINHA}(p), i)| = 0$ então SÁIDA ('Não existe solução única');
PARE.

Passo 5: Se NLINHA(i) ≠ NLINHA(p) então faça NCOPIA = NLINHA(i);
NLINHA(i) = NLINHA(p);
NLINHA(p) = NCOPIA;
(Troca de linha simulada.)

Passo 6: Para j = i + 1, ..., n execute os Passos 7 e 8.

Passo 7: Faça m(NLINHA(j), i) = a(NLINHA(j), i) / a(NLINHA(i),i).

Passo 8: Execute ($E_{NLINHA(j)}$ – m(NLINHA(j),i).$E_{NLINHA(i)}$) → ($E_{NLINHA(i)}$)

Passo 9: Se a(NLINHA(n), n) = 0 então SAÍDA ('Não existe solução única');
PARE.

Passo 10: Faça x_n = a(NLINHA(n), n +1) / a(NLINHA(n), n).
(Começa substituição regressiva.)

Passo 11: Para i = n -1, ..., 1

$$\text{Faça } x_i = \frac{a(NLINHA(i),n+1) - \sum_{j=i+1}^{n} a(NLINHA(i),j)\,.x_j}{a(NLINHA(i),i)}$$

Passo 12: SAÍDA (x_1, ..., x_n); (Procedimento completado com sucesso.)
PARE.

Com base no exemplo dado acima, o algoritmo deverá exigir informações do usuário:

- Número de Incógnitas do Sistema [n];
- Matriz Aumentada [A,b];

Após realizadas os cálculo , o algoritmo deverá apresentar ao final de sua execução uma das seguintes alternativas:

- Vetor Solução [Xn];
- Erro.

Para execução do programa em linguagem de programação, os seguintes parâmetros foram adotados:

- Linguagem de Programação: C;
- Compilador: Code::Blocks 16.01

2.2. Teste

Para teste do código fonte escrito em linguagem de programação C, foi usado exemplo de cálculo dado pelo Livro Análise Numérica (Burden, 2015), página 417:

$$2{,}11x_1 - 4{,}21x_2 + 0{,}921x_3 = 2{,}01$$

$$4{,}01x_1 + 10{,}2x_2 - 1{,}12x_3 = -3{,}09$$

$$1{,}09x_1 + 0{,}987x_2 + 0{,}832x_3 = 4{,}21$$

Como vemos a seguir o algoritmo deve prover o passo a passo das etapas de cálculo até a obtenção da Matriz Triangular Superior de A.

Resultados:

```
Matrizes

A=      +2,11000000             -4,21000000             +0,92100000
        +4,01000000             +10,20000000            -1,12000000
        +1,09000000             +0,98700000             +0,83200000

B=      +2,01000000
        -3,09000000
        +4,21000000

Definição do fator de escala

        s[1]=+4,21000000
        s[2]=+10,20000000
        s[3]=+1,09000000

Realizando as Eliminações

        r[1]= +0,50118765
        r[2]= +0,39313725
        r[3]= +1,00000000
        rmax= +1,00000000 ou seja, linha 3 troca com a linha 1

A[1]=   +1,09000000             +0,98700000             +0,83200000
        +4,01000000             +10,20000000            -1,12000000
        +2,11000000             -4,21000000             +0,92100000

B[1]=   +4,21000000
        -3,09000000
        +2,01000000

        multiplicador[2,1]= +3,67889908
        multiplicador[3,1]= +1,93577982

A[1]=   +1,09000000             +0,98700000             +0,83200000
        +0,00000000             +6,56892661             -4,18084404
        +0,00000000             -6,12061468             -0,68956881

B[1]=   +4,21000000
        -18,57816514
        -6,13963303
```

```
Realizando as Eliminações

        r[2]= +0,64401241
        r[3]= +1,45382771
        rmax= +1,45382771 ou seja, linha 3 troca com a linha 2

A[2]=   +1,09000000             +0,98700000             +0,83200000
        +0,00000000             -6,12061468             -0,68956881
        +0,00000000             +6,56892661             -4,18084404

B[2]=   +4,21000000
        -6,13963303
        -18,57816514

        multiplicador[3,2]= -1,07324623

A[2]=   +1,09000000             +0,98700000             +0,83200000
        +0,00000000             -6,12061468             -0,68956881
        +0,00000000             +0,00000000             -4,92092116

B[2]=   +4,21000000
        -6,13963303
        -25,16750311

Sistema Linear de Incógnitas

A[3]=   +1,09000000             +0,98700000             +0,83200000
        +0,00000000             -6,12061468             -0,68956881
        +0,00000000             +0,00000000             -4,92092116

B[3]=   +4,21000000
        -6,13963303
        -25,16750311

Solução

x[1]= -0,42800441
x[2]= +0,42690323
x[3]= +5,11438861
```

2.3. Validação

Para validação do algoritmo escrito foi utilizado o exemplo de cálculo demonstrado no livro Análise Numérica, (Burden, 2015), página 417, onde o autor demostra o método de Eliminação de Gauss com pivotamento parcial com escala com os seguintes parâmentros:

$$2{,}11x_1 - 4{,}21x_2 + 0{,}921x_3 = 2{,}01$$

$$4{,}01x_1 + 10{,}2x_2 - 1{,}12x_3 = -3{,}09$$

$$1{,}09x_1 + 0{,}987x_2 + 0{,}832x_3 = 4{,}21$$

Os resultados obtidos pelo autor estão na tabela 1 a seguir:

Tabela 1

Incógnitas	Resultado
x_1	-0,431
x_2	0,430
x_3	5,12

Utilizando os mesmos métodos do exemplo acima e aplicando ao algoritmo, obteve se os seguintes resultados:

```
Solução

x[1]= -0,42800441
x[2]= +0,42690323
x[3]= +5,11438861
```

Considerando que o autor utiliza se de aritmética de arredondamento de três algarismos significativos, o algoritmo foi considerado válido pois os valores de x_1, x_2 e x_3 tem igualdade em todas as casas decimais apresentadas pelo autor salvo arredondamentos no último algarismo significativo.

2.4. Pivotamento Completo

A título de estudo foi alterado o código fonte do algoritmo implementado introduzindo se o pivotamento de colunas na fase inicial.

O novo algoritmo (Anexo B) substitui as posições das colunas da matriz comparando o maior valor encontrado em módulo em cada coluna e deslocando o maior valor para a coluna da esquerda.

Foi introduzido no código anterior o seguinte algoritmo a fim de realizar a substituição das referências das colunas no algoritmo:

```
/*Pivotamento de colunas*/
for (j = 1; j <= n; j++)
{
        C[j] = j;
```

```
		s1max = 0.0;
		for (i = 1; i <= n; i++){s1max = max (s1max, fabs(A[i][j]));}
		S1[j] = s1max;
	}

	printf("\nDefinição do fator de escala para Colunas\n");
	for (i = 1; i <= n; i++){printf("\n\ts[%d]=%+.8f", i, S1[i]);}

	for (k = 1; k < n; k++)
	{
	printf("\n\nRealizando as Trocas de Colunas\n");

	rmax = 0.0;
		for (j = k ; j <= n; j++)
		{
			_j = C[j];
			if (S1[_j] > rmax)
			{
				rmax = S1[_j];
				i = j;
			}
		}

	/*Troca a referência das colunas*/
		_k = C[i];
		C[i] = C[k];
		C[k] = _k;

		printf("\n\trmax= %+.8f ou seja, coluna %d troca com a coluna %d\n", rmax, i, C[i]);

		printf("\nA[%d]=",k);
		for (i = 1; i <= n; i++){for (j = 1; j <= n; j++){printf("\t%+.8f\t", A[i][C[j]]);}
printf("\n");}
		printf("\nB[%d]=",k);
		for (i = 1; i <= n; i++){printf("\t%+.8f\n", B[i]);}
	}
```

Com base no algoritmo acima, obteve-se o seguinte resultado utilizando o mesmo exemplo de cálculo dado pelo Livro Análise Numérica (Burden, 2015), página 417:

```
Matrizes
A=	+2,11000000		-4,21000000		+0,92100000
	+4,01000000		+10,20000000		-1,12000000
	+1,09000000		+0,98700000		+0,83200000
B=	+2,01000000
	-3,09000000
	+4,21000000

Definição do fator de escala para Colunas
	s[1]=+4,01000000
	s[2]=+10,20000000
	s[3]=+1,12000000

Realizando as Trocas de Colunas
	rmax= +10,20000000 ou seja, coluna 2 troca com a coluna 1

A[1]=	-4,21000000		+2,11000000		+0,92100000
	+10,20000000		+4,01000000		-1,12000000
	+0,98700000		+1,09000000		+0,83200000
B[1]=	+2,01000000
	-3,09000000
	+4,21000000
```

```
Realizando as Trocas de Colunas
        rmax= +4,01000000 ou seja, coluna 2 troca com a coluna 1

A[2]=   -4,21000000             +2,11000000             +0,92100000
        +10,20000000            +4,01000000             -1,12000000
        +0,98700000             +1,09000000             +0,83200000
B[2]=   +2,01000000
        -3,09000000
        +4,21000000

Definição do fator de escala para Linhas
        s[1]=+4,21000000
        s[2]=+10,20000000
        s[3]=+1,09000000

Realizando as Eliminações
        r[1]= +1,00000000
        r[2]= +1,00000000
        r[3]= +0,90550459
        rmax= +1,00000000 ou seja, linha 1 troca com a linha 1

A[1]=   -4,21000000             +2,11000000             +0,92100000
        +10,20000000            +4,01000000             -1,12000000
        +0,98700000             +1,09000000             +0,83200000
B[1]=   +2,01000000
        -3,09000000
        +4,21000000

        multiplicador[2,1]= -2,42280285
        multiplicador[3,1]= -0,23444181

A[1]=   -4,21000000             +2,11000000             +0,92100000
        +0,00000000             +9,12211401             +1,11140143
        +0,00000000             +1,58467221             +1,04792090
B[1]=   +2,01000000
        +1,77983373
        +4,68122803


Realizando as Eliminações
        r[2]= +0,89432490
        r[3]= +1,45382771
        rmax= +1,45382771 ou seja, linha 3 troca com a linha 2

A[2]=   -4,21000000             +2,11000000             +0,92100000
        +0,00000000             +1,58467221             +1,04792090
        +0,00000000             +9,12211401             +1,11140143
B[2]=   +2,01000000
        +4,68122803
        +1,77983373

        multiplicador[3,2]= +5,75646747

A[2]=   -4,21000000             +2,11000000             +0,92100000
        +0,00000000             +1,58467221             +1,04792090
        +0,00000000             +0,00000000             -4,92092116
B[2]=   +2,01000000
        +4,68122803
```

```
        -25,16750311

Sistema Linear de Incógnitas

A[3]=   -4,21000000             +2,11000000             +0,92100000
        +0,00000000             +1,58467221             +1,04792090
        +0,00000000             +0,00000000             -4,92092116
B[3]=   +2,01000000
        +4,68122803
        -25,16750311

Solução

x[1]= -0,42800441
x[2]= +0,42690323
x[3]= +5,11438861
```

A princípio não houve diferença no resultado final pois o algoritmo implementado não utiliza de aritmética de arredondamento suficientemente limitante para que este apresente um erro de arredondamento ou de divisão por um número muito pequeno.

3. EXERCÍCIOS RESOLVIDOS

Exercício 6.2 – 9a (Burden, 2015)

$$0{,}03x_1 + 58{,}9x_2 = 59{,}2$$

$$5{,}31x_1 - 6{,}10x_2 = 47$$

```
Matrizes
A=      +0,03000000             +58,90000000
        +5,31000000             -6,10000000
B=      +59,20000000
        +47,00000000

Sistema Linear de Incógnitas
A[2]=   +5,31000000             -6,10000000
        +0,00000000             +58,93446328
B[2]=   +47,00000000
        +58,93446328

Solução
x[1]= +10,00000000
x[2]= +1,00000000
```

Exercício 6.2 – 9b (Burden, 2015)

$$3{,}03x_1 - 12{,}1x_2 + 14x_3 = -119$$

$$-3{,}03x_1 + 12{,}1x_2 - 7x_3 = 120$$

$$6{,}11x_1 - 14{,}2x_2 + 21x_3 = -139$$

```
Matrizes
A=      +3,03000000             -12,10000000            +14,00000000
        -3,03000000             +12,10000000            -7,00000000
        +6,11000000             -14,20000000            +21,00000000
B=      -119,00000000
        +120,00000000
        -139,00000000

Sistema Linear de Incógnitas
A[3]=   +6,11000000             -14,20000000            +21,00000000
        +0,00000000             +5,05810147             +3,41407529
        +0,00000000             +0,00000000             +7,00000000
B[3]=   -139,00000000
        +51,06873977
        +1,00000000

Solução
x[1]= -0,00000000
x[2]= +10,00000000
x[3]= +0,14285714
```

Exercício 6.2 – 9c (Burden, 2015)

$$1{,}19x_1 + 2{,}11x_2 - 100x_3 + x_4 = 1{,}12$$

$$14{,}2x_1 - 0{,}122x_2 + 12{,}2x_3 - x_4 = 3{,}44$$

$$0x_1 + 100x_2 - 99{,}9x_3 + x_4 = 2{,}15$$

$$015{,}3x_1 + 0{,}110x_2 - 13{,}1x_3 - x_4 = 4{,}16$$

```
Matrizes
A=   +1,19000000         +2,11000000          -100,00000000        +1,00000000
     +14,20000000        -0,12200000          +12,20000000         -1,00000000
     +0,00000000         +100,00000000        -99,90000000         +1,00000000
     +15,30000000        +0,11000000          -13,10000000         -1,00000000
B=   +1,12000000
     +3,44000000
     +2,15000000
     +4,16000000

Sistema Linear de Incógnitas
A[4]=   +14,20000000       -0,12200000        +12,20000000        -1,00000000
        +0,00000000        +100,00000000      -99,90000000        +1,00000000
        +0,00000000        +0,00000000        -26,00386117        +0,07505028
        +0,00000000        +0,00000000        +0,00000000         +0,77715086
B[4]=   +3,44000000
        +2,15000000
        +0,44832994
        -0,91906536

Solução
x[1]= +0,17682530
x[2]= +0,01269269
x[3]= -0,02065405
x[4]= -1,18260870
```

Exercício 6.2 – 9d (Burden, 2015)

$$\pi x_1 - ex_2 + \sqrt{2}x_3 - \sqrt{3}x_4 = \sqrt{11}$$

$$\pi^2 x_1 + ex_2 - e^2 x_3 + 3/7\, x_4 = 0$$

$$\sqrt{5}x_1 - \sqrt{6}x_2 + x_3 - \sqrt{2}x_4 = \pi$$

$$\pi^3 x_1 + e^2 x_2 - \sqrt{7}x_3 + 1/9\, x_4 = \sqrt{2}$$

```
Matrizes

A=     +3,14159265         -2,71828183          +1,41421356          -1,73205081
       +9,86960440         +2,71828183          -7,38905610          +0,42857143
       +2,23606798         -2,44948974          +1,00000000          -1,41421356
       +31,00627668        +7,38905610          -2,64575131           +0,11111111
B=     +3,31662479
       +0,00000000
       +3,14159265
       +1,41421356
```

```
Sistema Linear de Incógnitas
A[4]=   +3,14159265       -2,71828183       +1,41421356       -1,73205081
        +0,00000000       +11,25801607      -11,83193904      +5,86996954
        +0,00000000       +0,00000000       +19,35831445      -0,63531809
        +0,00000000       +0,00000000       +0,00000000       +0,06900168

B[4]=   +3,31662479
        -10,41948409
        +0,34924151
        +0,31444080

Solução
x[1]= +0,78839377
x[2]= -3,12541359
x[3]= +0,16759659
x[4]= +4,55700236
```

Exercício 6.2 – 25c (Burden, 2015)

Deduzindo a s equação para um sistema linear, encontra se:

$$0{,}013i_1 + 3{,}333x_2 - 0x_3 = 22$$

$$0{,}013x_1 - 0x_2 + 4{,}002x_3 = 12$$

$$x_1 - x_2 - x_3 = 0$$

```
Matrizes

A=      +0,01300000         +3,33300000         +0,00000000
        +0,01300000         +0,00000000         +4,00200000
        +1,00000000         -1,00000000         -1,00000000
B=      +22,00000000
        +12,00000000
        +0,00000000

Sistema Linear de Incógnitas
A[3]=   +1,00000000         -1,00000000         -1,00000000
        +0,00000000         +3,34600000         +0,01300000
        +0,00000000         +0,00000000         +4,01494949
B[3]=   +0,00000000
        +22,00000000
        +11,91452481

Solução
x[1]= +9,53102574
x[2]= +6,56348535
x[3]= +2,96754040
```

4. CONCLUSÃO

A Eliminação Regressiva de Gauss se mostra uma ferramenta poderosa para a resolução de Sistemas de Equações Lineares com implementação relativamente simples e pouco requerimento de capacidade computacional, desde que algumas situações sejam observadas como por exemplo a transformação da Matriz Aumentada em uma Matriz Triangular Superior.

Para tal transformação o Pivotamento Parcial com Escala demonstrou ser importante para o ajuste da Matriz onde a Eliminação Regressiva de Gauss será aplicada, não só eliminando possíveis pivôs nulos, que impossibilitariam a continuidade do método, como também ajustando possíveis divisões que podem gerar grandes erros de arredondamento para o resultado final.

Também foi estudado dentro deste relatório o sistema de Pivotamento Completo, ou seja, o ajuste de linhas e colunas para a minimização dos erros citados acima. Não foi encontrada diferença significativa dos valores finais entre um método de Pivotamento e outro dentro dos exercícios apresentados neste relatório. Essa característica pode ser explicada devido aos dois algoritmos implementados não possuírem limitações quanto a aritmética de arredondamento, portanto, erros provenientes de divisões muito pequenas são mitigados pela capacidade de armazenamento das variáveis tipo “double” dos algoritmos em questão.

5. REFERÊNCIAS

Burden, R. L. (2011). *Numerical Analysis* (9th Edition ed.). Boston: Cengage Learning.

Burden, R. L. (2015). *Análise Numérica* (Tradução da 10ª edição norte-americana ed.). São Paulo: Cengage Learning.

ANEXO A. CÓDIGO FONTE PIVOTAMENTO PARCIAL

```
/*********************************************
 * \brief   Programa realizado para a Aula de Métodos Númericos em Fenômenos de Transporte
 *          Análise Numérica - Burden, Richard - 10ª edição - Pág. 417
 *          Algoritmo 6.3 - Eliminação de Gauss com Pivotamento Parcial com Escala
 *
 * \param   Número de Incógnitas do Sistema n
 * \param   Matriz Aumentada AB
 * \return  Vetor Solução Xn
 *
 *********************************************/
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include <locale.h>

#define max(a,b) (a > b) ? a : b

void esc_piv(int, double **, int *, double *);
void solucao(int, double **, int *, double *, double *);

int main(void)
{
        setlocale(LC_ALL,"");

        double **A, *B, *X;
        int *L;
        int i, j, n;

        /*Entrada de parâmetros*/
        printf("Digite o número de incógnitas do sistema [n]: "); scanf("%d", &n);

        A = calloc((n+1), sizeof(double *));
        for (i=0; i<=n; i++){A[i] = calloc((n+1), sizeof(double));}
        B = calloc((n+1), sizeof(double));
        X = calloc((n+1), sizeof(double));
        L = calloc((n+1), sizeof(int));


        /*Entrada de parâmetros dependentes do número de equações*/
        printf("\nDigite os valores da Matriz A[i][j]: \n");
        for (i=1; i<=n; i++){for (j=1; j<=n; j++){printf("A[%d][%d]= ", i, j); scanf ("%lf", &A[i][j]);}}

        /*exemplo para debugação
        A[1][1] = 2.11  ;A[1][2] = -4.21   ;A[1][3] = 0.921;
        A[2][1] = 4.01  ;A[2][2] = 10.2    ;A[2][3] = -1.12;
        A[3][1] = 1.09  ;A[3][2] = 0.987   ;A[3][3] = 0.832;*/

        /*Entrada de parâmetros dependentes do número de equações*/
        printf("\nDigite os valores dos Resultados B[i]: \n");
        for (i=1; i<=n; i++){printf("B[%d]= ", i); scanf ("%lf", &B[i]);}

        /*exemplo para debugação
        B[1]    = 2.01  ;B[2]    = -3.09   ;B[3]    = 4.21;*/

        printf("\nMatrizes\n");
        printf("\nA=");
        for (i = 1; i <= n; i++){for (j = 1; j <= n; j++){printf("\t%+.8f\t", A[i][j]);} printf("\n");}
        printf("\nB=");
        for (i = 1; i <= n; i++){printf("\t%+.8f\t", B[i]); printf("\n");}

        esc_piv(n, A, L, B);
        solucao(n, A, L, B, X);

        printf("\nSolução\n");

        for (i = 1; i <= n; i++){printf("\nx[%d]= %+.8f", i, X[i]);}
        printf("\n\nFim do programa. ");

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

```
        free(A); free(X); free(B); free(L);

        system("PAUSE");
        return 0;
}

void esc_piv(int n, double **A, int *L, double *B)
{
        int i, j, k, _i, _k;
        double S[n+1];
        double xmult, smax, rmax, ratio;

        /*Definição do fator de escala s*/
        for (i = 1; i <= n; i++)
        {
                L[i] = i;
                smax = 0.0;
                for (j = 1; j <= n; j++){smax = max (smax, fabs(A[i][j]));}
                S[i] = smax;
        }

        printf("\nDefinição do fator de escala\n");
        for (i = 1; i <= n; i++){printf("\n\ts[%d]=%+.8f", i, S[i]);}

        /*Pivotamento de Colunas com escala */
        for (k = 1; k < n; k++)
        {
                printf("\n\nRealizando as Eliminações\n");

                rmax = 0.0;
                for (i = k ; i <= n; i++)
                {
                        _i = L[i];
                        ratio = fabs(A[_i][k]) / S[_i];
                        printf("\n\tr[%d]= %+.8f", i, ratio);
                        if (ratio > rmax)
                        {
                                rmax = ratio;
                                j = i;
                        }
                }

        /*Troca a referência das linhas*/
        _k = L[j];
        L[j] = L[k];
        L[k] = _k;

        printf("\n\trmax= %+.8f ou seja, linha %d troca com a linha %d\n", rmax, j, L[j]);

        printf("\nA[%d]=",k);
        for (i = 1; i <= n; i++){for (j = 1; j <= n; j++){printf("\t%+.8f\t", A[L[i]][j]);} printf("\n");}
        printf("\nB[%d]=",k);
        for (i = 1; i <= n; i++){printf("\t%+.8f\n", B[L[i]]);}

                for (i = k+1; i <= n; i++)
                {
                        _i = L[i];
                        xmult = A[_i][k] / A[_k][k];
                        printf("\n\tmultiplicador[%d,%d]= %+.8f", i, k, xmult);
                        A[_i][k] = 0.0;
                        for (j = k+1; j<= n; j++){A[_i][j] -= xmult * A[_k][j];}
                        B[_i] -= xmult * B[_k];
                }

        printf("\n\nA[%d]=",k);
        for (i = 1; i <= n; i++){for (j = 1; j <= n; j++){printf("\t%+.8f\t", A[L[i]][j]);} printf("\n");}
        printf("\nB[%d]=",k);
        for (i = 1; i <= n; i++){printf("\t%+.8f\t", B[L[i]]); printf("\n");}
        }
}

/*Calculo da solução X*/
void solucao(int n, double **A, int *L, double *B, double *X)
{
```

```
        int i, j, k, _i, _k, _n;
        double soma;

        for (k = 1; k < n ; k++)
        {
                _k = L[k];
                for (i = k+1; i <= n; i++)
                {
                        _i = L[i];
                        B[_i] -= A[_i][k] * B[_k];
                }
        }
        _n = L[n];
        X[n] = B[_n] / A[_n][n];

        printf("\nSistema Linear de Incógnitas\n");
        printf("\nA[%d]=",k);
        for (i = 1; i <= n; i++){for (j = 1; j <= n; j++){printf("\t%+.8f\t", A[L[i]][j]);} printf("\n");}
        printf("\nB[%d]=",k);
        for (i = 1; i <= n; i++){printf("\t%+.8f\t", B[L[i]]); printf("\n");}

        for (i = n-1; i>= 1; i--)
        {
                _i = L[i];
                soma = B[_i];
                for (j = i+1; j <= n ; j++){soma -= A[_i][j] * X[j];}
                X[i] = soma / A[_i][i];
        }
}
```

ANEXO B. CÓDIGO FONTE PIVOTAMENTO COMPLETO

```
/*********************************************
 * \brief   Programa realizado para a Aula de Métodos Númericos em Fenômenos de Transporte
 *         Análise Numérica - Burden, Richard - 10ª edição - Pág. 417
 *         Algoritmo 6.3 - Eliminação de Gauss com Pivotamento Parcial com Escala
 *
 * \param   Número de Incógnitas do Sistema n
 * \param   Matriz Aumentada AB
 * \return  Vetor Solução Xn
 *
 *********************************************/
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include <locale.h>

#define max(a,b) (a > b) ? a : b

void esc_piv(int, double **, int *, int *, double *);
void solucao(int, double **, int *, int *, double *, double *);

int main(void)
{
    setlocale(LC_ALL,"");

        double **A, *B, *X;
        int *L, *C;
        int i, j, n;

    printf("Digite o número de incógnitas do sistema [n]: "); scanf("%d", &n);

        A = calloc((n+1), sizeof(double *));
    for (i=0; i<=n; i++){A[i] = calloc((n+1), sizeof(double));}
        B = calloc((n+1), sizeof(double));
        X = calloc((n+1), sizeof(double));
        L = calloc((n+1), sizeof(int));
    C = calloc((n+1), sizeof(int));


    /*Entrada de parâmetros dependentes do número de equações*/
    printf("\nDigite os valores da Matriz A[i][j]: \n");
    for (i=1; i<=n; i++){for (j=1; j<=n; j++){printf("A[%d][%d]= ", i, j); scanf ("%lf", &A[i][j]);}}

        /*exemplo para debugação
        A[1][1] = 2.11  ;A[1][2] = -4.21   ;A[1][3] = 0.921;
        A[2][1] = 4.01  ;A[2][2] = 10.2    ;A[2][3] = -1.12;
        A[3][1] = 1.09  ;A[3][2] = 0.987   ;A[3][3] = 0.832;*/

        /*Entrada de parâmetros dependentes do número de equações*/
    printf("\nDigite os valores dos Resultados B[i]: \n");
        for (i=1; i<=n; i++){printf("B[%d]= ", i); scanf ("%lf", &B[i]);}

        /*exemplo para debugação
        B[1]    = 2.01  ;B[2]    = -3.09   ;B[3]    = 4.21;*/

        printf("\nMatrizes\n");
        printf("\nA=");
        for (i = 1; i <= n; i++){for (j = 1; j <= n; j++){printf("\t%+.8f\t", A[i][j]);} printf("\n");}
    printf("\nB=");
        for (i = 1; i <= n; i++){printf("\t%+.8f\t", B[i]); printf("\n");}

        esc_piv(n, A, L, C, B);
        solucao(n, A, L, C, B, X);

        printf("\nSolução\n");

        for (i = 1; i <= n; i++){printf("\nx[%d]= %+.8f", i, X[C[i]]);}
    printf("\n\nFim do programa. ");

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

```
        free(A); free(X); free(B); free(L); free(C);

        system("PAUSE");
    return 0;
}

void esc_piv(int n, double **A, int *L, int *C, double *B)
{
        int i, j, k, _i, _j, _k;
        double S1[n+1], S2[n+1];
        double xmult, s1max, s2max, rmax, ratio;

    /*Pivotamento de colunas*/
    /*Definição do fator de escalas para colunas*/
        for (j = 1; j <= n; j++)
        {
                C[j] = j;
                s1max = 0.0;
                for (i = 1; i <= n; i++){s1max = max (s1max, fabs(A[i][j]));}
        S1[j] = s1max;
    }

    printf("\nDefinição do fator de escala para Colunas\n");
    for (i = 1; i <= n; i++){printf("\n\ts[%d]=%+.8f", i, S1[i]);}

    for (k = 1; k < n; k++)
        {
        printf("\n\nRealizando as Trocas de Colunas\n");

            rmax = 0.0;
                for (j = k ; j <= n; j++)
                {
                        _j = C[j];
                        if (S1[_j] > rmax)
                        {
                                rmax = S1[_j];
                                i = j;
                        }
                }

        /*Troca a referência das colunas*/
                _k = C[i];
                C[i] = C[k];
                C[k] = _k;

        printf("\n\trmax= %+.8f ou seja, coluna %d troca com a coluna %d\n", rmax, i, C[i]);

        printf("\nA[%d]=",k);
        for (i = 1; i <= n; i++){for (j = 1; j <= n; j++){printf("\t%+.8f\t", A[i][C[j]]);} printf("\n");}
        printf("\nB[%d]=",k);
        for (i = 1; i <= n; i++){printf("\t%+.8f\n", B[i]);}
        }


    /*Pivotamento para Linhas*/
        /*Definição do fator de escalas para linhas*/
        for (i = 1; i <= n; i++)
        {
                L[i] = i;
                s2max = 0.0;
                for (j = 1; j <= n; j++){s2max = max (s2max, fabs(A[i][C[j]]));}
        S2[i] = s2max;
    }

    printf("\nDefinição do fator de escala para Linhas\n");
    for (i = 1; i <= n; i++){printf("\n\ts[%d]=%+.8f", i, S2[i]);}

        /*Pivotamento de Colunas com escala */
        for (k = 1; k < n; k++)
        {
        printf("\n\nRealizando as Eliminações\n");

            rmax = 0.0;
                for (i = k ; i <= n; i++)
```

```
                {
                        _i = L[i];
                        ratio = fabs(A[_i][C[k]]) / S2[_i];
                        printf("\n\tr[%d]= %+.8f", i, ratio);
                        if (ratio > rmax)
                        {
                                rmax = ratio;
                                j = i;
                        }
                }

        /*Troca a referência das linhas*/
                _k = L[j];
                L[j] = L[k];
                L[k] = _k;

        printf("\n\trmax= %+.8f ou seja, linha %d troca com a linha %d\n", rmax, j, L[j]);

        printf("\nA[%d]=",k);
        for (i = 1; i <= n; i++){for (j = 1; j <= n; j++){printf("\t%+.8f\t", A[L[i]][C[j]]);}
printf("\n");}
        printf("\nB[%d]=",k);
        for (i = 1; i <= n; i++){printf("\t%+.8f\n", B[L[i]]);}

                for (i = k+1; i <= n; i++)
                {
                        _i = L[i];
                        xmult = A[_i][C[k]] / A[_k][C[k]];
            printf("\n\tmultiplicador[%d,%d]= %+.8f", i, k, xmult);
                        A[_i][C[k]] = 0.0;
                        for (j = k+1; j<= n; j++){A[_i][C[j]] -= xmult * A[_k][C[j]];}
                        B[_i] -= xmult * B[_k];
                }

        printf("\n\nA[%d]=",k);
        for (i = 1; i <= n; i++){for (j = 1; j <= n; j++){printf("\t%+.8f\t", A[L[i]][C[j]]);}
printf("\n");}
        printf("\nB[%d]=",k);
        for (i = 1; i <= n; i++){printf("\t%+.8f\t", B[L[i]]); printf("\n");}
    }
}

/*Calculo da solução X*/
void solucao(int n, double **A, int *L, int *C, double *B, double *X)
{
        int i, j, k, _i, _k, _n;
        double soma;

        for (k = 1; k < n ; k++)
        {
                _k = L[k];
                for (i = k+1; i <= n; i++)
                {
                        _i = L[i];
                        B[_i] -= A[_i][C[k]] * B[_k];
                }
        }
        _n = L[n];
        X[C[n]] = B[_n] / A[_n][C[n]];

    printf("\nSistema Linear de Incógnitas\n");
    printf("\nA[%d]=",k);
    for (i = 1; i <= n; i++){for (j = 1; j <= n; j++){printf("\t%+.8f\t", A[L[i]][C[j]]);} printf("\n");}
    printf("\nB[%d]=",k);
    for (i = 1; i <= n; i++){printf("\t%+.8f\t", B[L[i]]); printf("\n");}

        for (i = n-1; i>= 1; i--)
        {
                _i = L[i];
                soma = B[_i];
                for (j = i+1; j <= n ; j++){soma -= A[_i][C[j]] * X[C[j]];}
                X[C[i]] = soma / A[_i][C[i]];
        }
}
```