UNIVERSIDADE FEDERAL DO ABC

MCBM003-23 Aproximação Teórica e Numérica I (3º 2025)
Prof. André Pierro de Camargo
Data de entrega: até 02/11/2025

Projeto 1: Um estudo sobre a convergência de splines cúbicos interpoladores

1 Orientações gerais

- A linguagem de programação utilizada na implementação dos métodos é de livre escolha.
- Peça ajuda ao professor sempre que necessário.
- Realize testes preliminares com cada método para identificar possíveis erros de implementação. Por exemplo: teste o método de eliminação de Gauss em sistemas de equações cuja solução seja previamente conhecida e veja se o resultado obtido é coerente.
- Discuta com o professor caso encontre resultados aparentemente sem sentido. Isso pode (ou não) ser um erro de implementação, ou mesmo um indicativo de que o método utilizado possui as suas limitações.

Cada grupo deverá entregar

- Um relatório **em pdf** contendo a resolução dos exercícios teóricos e **TUDO** o que foi solicitado nos exercícios práticos (leiam a descrição com bastante atenção).
- O arquivo contendo o código fonte utilizado nos exercícios práticos.

Enviar o material por e-mail para andre.camargo@ufabc.edu.br, especificando o número do grupo.

Observação: Na Seção 3 encontram-se alguns dos algoritmos que serão utilizados nos exercícios práticos.

1.1 Demais instruções (IMPORTANTE)

A implementação dos métodos deve ser feita **pelos interantes do grupo** e nunca por assistentes de Inteligência Artificial (AI). Para inibir o uso de IAs, os algoritmos devem ser implementados conforme os speudo códigos descritos na Seção 3, **SEMPRE QUE SOLICITADO**. **Lembrem-se**: você são os autores e, portanto, responsáveis pelos seus códigos. Dessa forma, **TUDO** o que for entregue será avaliado e **TUDO** o que constar no código/relatório que não foi visto em sala de aula (e que não constar nos algoritmos da Seção 3) e que não for devidamente explicado implicará em descontos na nota.

- O código fonte deverá ser entregue em formato legível que possa ser aberto por um editor de texto.
- A entrega do relatório **em pdf** é imprescindível. **NÃO SERÂO ACEITOS** trabalhos contendo apenas o código fonte, mas esse pode ser anexado ao relatório.
- Os exercícios são sequênciais, ou seja, a resolução do próximo depende da resolução do anterior. Dessa forma, a correção também será sequêncial. Dessa forma, para pontuar na Tarefa 3, por exemplo, é necessário ter pontuado nas Tarefas 1 e 2.
- Paralelo à entrega do trabalho, também poderá ser solicitado ao grupo explicações adicionais sobre a implementação dos métodos utilizados.

2 Splines cúbicos interpoladores

Dado um conjunto de dados observados $\Delta : \{(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)\}$, um Spline cúbico interpolador de Δ é uma função $S_\Delta : \mathbb{R} \to \mathbb{R}$ tal que

- S_Δ é duas vezes continuamente diferenciável.
- Para cada $i = 0, 1, \ldots, n-1$, S_Δ coincide com um polinômio de grau menor ou igual a 3 no intervalo $[x_i, i_{i+1}]$.
- $S_\Delta(x_i) \ = \ y_i, \ i = 0, 1, \ldots, n.$

É possível mostrar (veja, por exemplo, *Introduction to Numerical Analysis* de Stoer. J., and Bulirsch, R.) que, dados números y_0'' e y_n'', existe um único Spline cúbico interpolador de Δ satisfazendo

$$S_\Delta(x_0)'' = y_0'' \text{ e } S_\Delta(x_n)'' = y_n''.$$

Existem inúmeras aplicações de splines, por exemplo, reconstrução de imagens e aproximação de funções. O foco desse projeto é fazer uma análise empírica da ordem de convergência dos splines cúbicos interpoladores quando os dados amostrais $\{(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)\}$ vem de uma função suave $f : [x_0, x_n] \to \mathbb{R}$, isto é,

$$y_i \;=\; f(x_i), \quad i = 0, 1, \ldots, n.$$

2.1 Exercício 1: cálculo de splines cúbicos

O objetivo desse exercício é implementar a função spline cúbico, que recebe como dados de entrada os dados amostrais $\Delta : \{(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)\}$ e também o ponto x onde o spline S_Δ será avaliado.

Para calcular $S_\Delta(x)$, utilizamos a expressão polinomial de $S_\Delta(x)$ em cada um dos intervalos $[x_i, x_{i+1}],\ i = 0, 1, \ldots, n-1$. Defina

$$h_{i+1} := x_{i+1} - x_i,\ i = 0, 1, \ldots, n-1.$$

Vale que $S_\Delta(x) \;=$

$$\tfrac{M_i}{6h_{i+1}}(x_{i+1} - x)^3 + \tfrac{M_{i+1}}{6h_{i+1}}(x - x_i)^3 + A_i(x - x_i) + B_i, \qquad \text{em} \;\; [x_i, x_{i+1}], \tag{1}$$

sendo M_i, A_i e B_i constantes que dependem de Δ e podem ser determinadas da seguinte forma: $M_0, M_1, \ldots, M_n$ formam o vetor solução do seguinte sistema de equações

$$
\underbrace{\begin{bmatrix} 2 & \lambda_0 & 0 & \dots & \dots & 0 \\ \mu_1 & 2 & \lambda_1 & \dots & \dots & 0 \\ 0 & \mu_2 & 2 & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & \lambda_{n-1} \\ 0 & \dots & \dots & \dots & \mu_n & 2 \end{bmatrix}}_{T} \times \begin{bmatrix} M_0 \\ M_1 \\ M_2 \\ \vdots \\ \vdots \\ M_{n-1} \\ M_n \end{bmatrix} = \underbrace{\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ \vdots \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix}}_{v}, \tag{2}
$$

sendo $\lambda_0 = 0,\ \mu_n = 0,\ d_0 \ = \ 2y_0'',\ d_n = 2y_n''$ e

$$
\begin{cases} \mu_i \ = \ \dfrac{h_i}{h_i + h_{i+1}}, \quad \lambda_i \ = \ \dfrac{h_{i+1}}{h_i + h_{i+1}}, \\ d_i \ = \ \dfrac{6}{h_i + h_{i+1}} \left(\dfrac{y_{i+1} - y_i}{h_{i+1}} - \dfrac{y_i - y_{i-1}}{h_i} \right), \end{cases}
$$

$i = 1, 2, \dots, n-1$. As constantes A_i e B_i são calculadas a partir de $M_0, M_1, \dots, M_n$:

$$
\begin{cases} B_i \ = \ y_i - \dfrac{M_i}{6} h_{i+1}^2 \\ A_i \ = \ \dfrac{y_{i+1} - y_i}{h_{i+1}} - \dfrac{M_{i+1} - M_i}{6} h_{i+1},\ i = 0, 1, \dots, n-1. \end{cases} \tag{3}
$$

2.1.1 Tarefa 1

Considere o seguinte conjunto de dados:

i =	0	1	2	3	4	5	6	7
x_i	-0.9	-0.83	-0.6	-0.49	0	0.2	0.6	0.83
y_i	0.0	1	2.4	4.1	6	8.2	10.6	13.4

Tabela 1: Um conjunto de dados observados Δ.

Para o conjunto de dados Δ da Tabela 1, com $y_0'' = 1$ e $y_n'' = -1$,

1. (0.5) Calcule a matriz T e o vetor de termos independentes v do sistema de equações (2). Exiba os valores calculados com **todas as casas decimais disponíveis**.

   Importante: O conjunto de dados da Tabela 1 é apenas um conjunto de dados para teste (outros conjuntos de dados serão utilizados posteriormente). Dessa forma, os dados de entrada para o seu programa deverão ser dois vetores x e y genéricos (os que formam Δ) de mesmo tamanho, e também os números y_0'' e y_n''.

2. (2.0) Ulizando os valores calculados no item anterior, resolva o sistema de equações resultante pelo método da Eliminação de Gauss. Implemente o método: use o pseudo-código da Seção 3. Exiba a matriz e o vetor de termos independentes após a etapa de triangularização (eliminação). Exiba as soluções obtidas com **TODAS** as casas decimais disponíveis.

   Observação: o uso de solvers, isto é, métodos prontos resolver sistemas de equações, implicará em descontos na pontuação.

   Para identificar possíveis erros de implementação, encontre abaixo as saídas esperadas para o programa para o seguinte conjunto de dados (também com $y_0'' = 1$ e $y_n' = -1$)

i =	0	1	2	3	4	5	6	7
x_i	-0.9	-0.83	-0.6	-0.49	0	0.2	0.6	0.83
y_i	0.0	2	2.6	0.1	6	-5	3	1

Tabela 2: Conjunto de dados Δ para teste

```
a. Matriz T e vetor v antes da etapa de eliminação

  T
            [,1]        [,2]        [,3]        [,4]        [,5]        [,6]        [,7]        [,8]
[1,] 2.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
[2,] 0.2333333333 2.0000000000 0.7666666667 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
[3,] 0.0000000000 0.6764705882 2.0000000000 0.3235294118 0.0000000000 0.0000000000 0.0000000000 0.0000000000
[4,] 0.0000000000 0.0000000000 0.1833333333 2.0000000000 0.8166666667 0.0000000000 0.0000000000 0.0000000000
```

```
[5,] 0.0000000000 0.0000000000 0.0000000000 0.7101449275 2.0000000000 0.2898550725 0.0000000000 0.0000000000
[6,] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.3333333333 2.0000000000 0.6666666667 0.0000000000
[7,] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.6349206349 2.0000000000 0.3650793651
[8,] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 2.0000000000
>      v
             [,1]
[1,]    2.0000000
[2,] -519.2546584
[3,] -447.1053243
[4,]  347.6808905
[5,] -582.9636202
[6,]  750.0000000
[7,] -273.2919255
[8,]   -2.0000000

b. Matriz T e vetor v após a etapa de eliminação

        T
     [,1] [,2]         [,3]         [,4]         [,5]         [,6]         [,7]         [,8]
[1,]    2    0 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
[2,]    0    2 0.7666666667 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
[3,]    0    0 1.7406862745 0.3235294118 0.0000000000 0.0000000000 0.0000000000 0.0000000000
[4,]    0    0 0.0000000000 1.9659250915 0.8166666667 0.0000000000 0.0000000000 0.0000000000
[5,]    0    0 0.0000000000 0.0000000000 1.7049980728 0.2898550725 0.0000000000 0.0000000000
[6,]    0    0 0.0000000000 0.0000000000 0.0000000000 1.9433322776 0.6666666667 0.0000000000
[7,]    0    0 0.0000000000 0.0000000000 0.0000000000 0.0000000000 1.7821883431 0.3650793651
[8,]    0    0 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 2.0000000000

v
             [,1]
[1,]    2.0000000
[2,] -519.4879917
[3,] -271.3961507
[4,]  376.2649965
[5,] -718.8806396
[6,]  890.5437834
[7,] -564.2481510
[8,]   -2.0000000

c. solução do sistema

M
[1,]    1.0000000
[2,] -171.0100091
[3,] -231.4799653
[4,]  406.5715296
[5,] -517.9887857
[6,]  566.7978840
[7,] -316.3992593
[8,]   -1.0000000
```

3. (1.0) Utilizando os resultados obtidos no item anterior, calcule as constantes A_i e $B_i, i = 0, 1, \ldots, n-1$ definidas em (3).

Para identificar possíveis erros de implementação, encontre abaixo as saídas esperadas para o programa para o conjunto de dados da Tabela 2 (também com $y_0'' = 1$ e $y_n' = -1$)

```
 A
30.578212011   4.926710639 -34.424883467  87.546575408 -91.159555658  78.879809552 -20.785957112
B
-8.166666667e-04  3.507738247e+00  3.066817930e+00 -1.616963738e+01  9.453258571e+00 -2.011461024e+01
 5.789586802e+00
```

4. (2.0) Uma vez calculados os coeficientes M, A e B que aparecem em (1), temos condicões de calcular o spline interpolador para um dado valor x^* da variável x, utilizando a fórmula (1). Para isso, precisamos saber quais dos coeficientes M_is A_i e B_i devemos utilizar, isto é, devemos descobrir em qual intevalo dos intervalos $[x_i, x_{i+1}], i = 0, 1, \dots, n-1$, está o ponto x^*. A forma mais simples de verificar isso é fazer um laço sobre i e ir verificando se

$$x_i < x^*, \tag{4}$$

parando o laço quando a condição (4) for falsa. Porém, observe que, no pior caso (por exemplo, quando $x^* \in [x_{n-1}, x_n]$), será necessário percorrer todos os intervalos e isso é muito custoso computacionalmente quando n é grande, por exemplo $n = 10^7$, $n = 10^8$, etc.

Para contornar esse problema, iremos utilizar um algoritmo de busca binária. Essencialmente, começamos com os índices $i_0 = 0$ e $I_0 = n$, que correspondem ao intervalo de busca $[x_0, x_n]$, e, a cada passo, descartamos (a grosso modo) metade dos índices comparando o valor x^* com o valor x_{α_0}, sendo o índice α_0 mais ou menos na metade entre i_0 e I_0, ou seja:

$$\alpha_0 = \frac{i_0 + I_0}{2},$$

arrendondado (sempre para cima, ou sempre para baixo).

Tome, por exemplo, os dados da Tabela 1 e suponha que $x^* = -0.47$. Nesse caso, temos $i_0 = 0, I_0 = 7$ e $\alpha_0 = 4$ (se decidirmoa arredondar para cima). Como $x^* < x_4 = 0$, sabemos, então, que x^* está no intervalo $[x_0, x_4]$ e, assim, reduzimos o intervalo de busca inicial. Daí repetimos o processo definindo $i_1 = 0, I_1 = 4$ e $\alpha_1 = \frac{0+4}{2} = 2$.

Comparando x^* com $x_2 = -0.6$, temos que $x^* \geq -0.6$. Isso nos diz que x^* está no intervalo $[x_2, x_4]$. Repetindo o processo mais uma vez, fazemos $i_2 = 2, I_2 = 4$ e $\alpha_2 = \frac{2+4}{2} = 3$. Como $x^* > x_3 = -0.49$, temos, então, que x^* está no intervalo $[x_3, x_4]$. Logo, o índice "i"é $i = 3$ e as constantes a ser utilizadas para avaliar $S_\Delta(x)$ no ponto x^* são M_3, M_4, A_3 e B_3.

Implemente o método de busca binária descrito na seção 3 e utilize-o para calcular o interpolador S_Δ, com os dados dos da Tabela 1, nos pontos $-0.9, -0.8, -0, 7, \ldots ...0.7, 0.8$.

Para identificar possíveis erros de implementação, encontre abaixo as saídas esperadas para o programa para $x^* = -0.6, x^* = 0.25$ e $x^* = 0.5$

```
x^* = -0.6

j  i_j  I_j
0   1    8
1   1    4
2   2    4
3   3    4

S(-0.6) = 2.4

OBSERVAÇÃO IMPORTANTE: no R, os índices dos vetores começam com ''1'' ao invés de ''zero''.
Assim, no exemplo acima, em linguagens começando do índice ''zero'', as saídas para x^* = -0.6 são:

j  i_j  I_j
0   0    7
1   0    3
2   1    3
3   2    3

############################################

x^* = 0.25

j  i_j  I_j
0    1    8
1    4    8
2    6    8
3    6    7

S(0.25) = 8.63723321
############################################

j  i_j  I_j
0    1    8
1    4    8
2    6    8
3    6    7
```

```
S(0.5) = 9.924920762
```

2.1.2 Tarefa 2

Nessa tarefa, iremos usar pontos igualmente espaçados em $[0,1]$, isto é: $[a,b]=[0,1]$,

$$x_i = a + ih, i = 0, 1, \ldots, n \quad h = \frac{b-a}{n}. \tag{5}$$

Dada uma função contínua $f : [0,1] \to \mathbb{R}$, defina

$$y_i = f(x_i),\ i = 0, 1, \ldots, n. \tag{6}$$

O objetivo dessa tarefa e analisar o quão bem $f(x)$ é aproximada pelo spline cúbico interpolador $S_\Delta(x)$ com Δ definido em (5) e (6) (para as nossas simulações, utilizaremos $f(x) = \exp(x)$). Para $f(x) = \exp(x)$, $n = 10$ e $y_0'' = y_n'' = 0$,

1. (0.5) Para $m = 13$, calcule $S_\Delta(x)$ nos pontos $x = t_j^*$ definidos por

   $$t_j^* = a + jh^*,\quad j = 0, 1, \ldots, m,\quad h^* = \frac{b-a}{m}.$$

   Para identificar possíveis erros de implementação, encontre abaixo as saídas esperadas para a função $f(x) = \cos(x)$

```
 j       0 1.00000000000 2.0000000000 3.0000000000 4.0000000000 5.0000000000 6.0000000000
t_j      0 0.07692307692 0.1538461538 0.2307692308 0.3076923077 0.3846153846 0.4615384615
S(t_j)   1 0.99681868481 0.9883052962 0.9734563393 0.9530387072 0.9269454286 0.8953666581

 j       7.0000000000 8.0000000000 9.0000000000 10.0000000000 11.0000000000 12.0000000000
t_j      0.5384615385 0.6153846154 0.6923076923  0.7692307692  0.8461538462  0.9230769231
S(t_j)   0.8584980783 0.8165527899 0.7697769996  0.7184273380  0.6629305894  0.6032481560

 j       13.0000000000
t_j      1.0000000000
S(t_j)   0.5403023059
```

2. (0.5) Calcule o erro máximo E_n entre f e S_Δ nos pontos t_j^*:

   $$E_n = \max_{0 \le j \le m} e_j, \qquad e_j = |f(t_j^*) - S_\Delta(t_j^*)|$$

Para identificar possíveis erros de implementação, encontre abaixo as saídas esperadas para a função $f(x) = \cos(x)$

```
j    0 1.000000000000 2.0000000000000 3.000000000e+00 4.000000000e+00 5.000000000e+00
t_j  0 0.076923076923 0.1538461538462 2.307692308e-01 3.076923077e-01 3.846153846e-01
e_j  0 0.000224193888 0.0001162923291 3.440032321e-05 3.692223952e-06 2.618663206e-06

j    6.000000000e+00 7.000000000e+00 8.000000000e+00 9.000000000e+00 1.000000000e+01
t_j  4.615384615e-01 5.384615385e-01 6.153846154e-01 6.923076923e-01 7.692307692e-01
e_j  1.782177783e-06 5.661994570e-07 1.310503716e-06 1.968864934e-06 1.860780959e-05

j    1.100000000e+01 1.200000000e+01    13
t_j  8.461538462e-01 9.230769231e-01     1
e_j  6.279218821e-05 1.211316962e-04     0

E_n = e_1 =  0.000224193888
```

2.2 Tarefa 3

Pode-se verificar que a segunda derivadas do spline cúbico $S''_\Delta(x)$ satisfaz $S''_\Delta(x_0) = y''_0$ e $S''_\Delta(x_n) = y''_n$. É esperado, portanto, que $S''_\Delta(x)$ aproxime melhor f quando $y''_0 = f''(x_0)$ e $y''_n = f''(x_n)$.

1. (1.0) Calcule os erros E_n no item 2 da Tarefa 2 (com a mesma f) com $m = 10117$ fixado e $n = 10, 20, 30, \ldots, 200$ em dois cenários:

$$\text{caso 1 } : y''_0 = y''_n = 0;$$
$$\text{caso 2 } : y''_0 = f''(x_0) \text{ e } y''_n = f''(x_n).$$

Em qual caso o erro fica menor para valores grandes de n?

Para identificar possíveis erros de implementação, encontre abaixo as saídas esperadas para a função $f(x) = \cos(x)$ e $n = 10, 20, 30$ e 40

```
# caso 1: y''_0 = y''_n = 0
n    1.000000e+01 2.000000e+01 3.000000e+01 4.000000e+01
E_n  4.915632e-04 1.227706e-04 5.455482e-05 3.068514e-05

# caso 2:  y''_0 = f''(x_0)  e  y''_n = f''(x_n)

n    1.000000e+01 2.000000e+01 3.000000e+01 4.000000e+01
E_n  6.551877e-07 4.093091e-08 8.084419e-09 2.557849e-09
```

2.2.1 Tarefa 4

Sob algumas hipóteses sobre a função interpolada f, pode-se provar que existem constantes positivas K e ρ tais que, para n suficientemente grande, o erro E_n calculado na tarefa anterior se comporta como $K\frac{1}{n^\rho}$. As constantes K e ρ dependem apenas de f e de como são escohidos y_0'' e y_n''. Em geral, o número

$$\rho \tag{7}$$

é um número inteiro positivo e é chamado de a ordem de convergência. Para estimar a ordem de convergência ρ, escrevemos

$$E_n \approx K\frac{1}{n^\rho}$$

e aplicamos o logaritmo (natural) dos dois lados:

$$\log(E_n) \approx \log(K) - \rho\log(n). \tag{8}$$

1. (0.5) Calcule os valores

$$(\log(n_1), \log(E_{n_1})), (\log(n_2), \log(E_{n_2})), \ldots, (\log(n_{20}), \log(E_{n_{20}})) \tag{9}$$

   utilizando os erros E_n calculados na Tarefas 3 para os casos 1 e 2 $(n_1 = 10, n_2 = 20, \ldots n_{20} = 200)$.

2. (2.0)

   Observe que, se o erro E_n obedece aproximadamente (8), então os pontos calculados em (9) deverão estar próximos a uma reta com coeficiente angular próximo a $-\rho$. Verifique essa afirmação ajustando os dados calculados (9) nos casos 1 e 2 por duas retas pelo método dos mínimos quadrados discreto. Faça um gráfico exibindo os valores (9) e também as retas de mínimos quadrados, assim, como na Figura 1. Qual é a orden de convergência sugerida pelos ajustes de mínimos quadrados em cada caso?

   A fim de validar o seu código, encontre na Figura 1 os valores obtidos aplicando o mesmo procedimento para a função $f(x) = \cos(x)$.

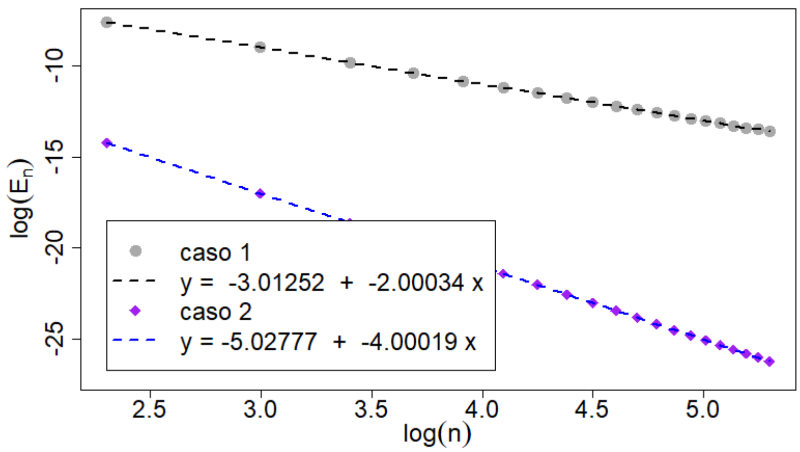

Figura 1: O erro $\log(E_n)$ nos casos 1 e 2 e as suas aproximações de mínimos quadrados para $f(x) = \cos(x)$.

3 Algoritmos

Considere um sistema linear de m equações com m incógnitas dado na forma matricial por

$$A \times x = y, \tag{10}$$

sendo A a matriz (quadrada $m \times m$) do sistema e $y \in \mathbb{R}^m$ o vetor de termos independentes. A seguir são listados os algoritmos para resolver o sistema de equações (10). Nesses algoritmos, os índices da matriz e do vetor são representados por números de 1 até m. Para implementação em linguagens que utilizam outro tipo de indexação (a maioria das linguagens começa com o índice zero) será preciso fazer os ajustes necessários.

3.1 Método da eliminação de Gauss

Entrada: A, y, m = ordem da matriz A.
Saída: O vetor x, solução do sistema (10).

Parte 1: Escalonar (triangularizar o sistema)

Para j de 1 até m-1, faça

- Se $A_{j,j} = 0$, encontre k tal que $A_{k,j} \neq 0$ e troque as linhas j e k da Matriz A. Se isso não for possível, então a matriz A é singular e convém exibir uma mensagem de erro.
- Para i de j+1 até m, faça // elimina todos os elementos de A da // coluna j que estão abaixo da diagonal
  - Defina $\mu = -\frac{A_{i,j}}{A_{j,j}}$ // define o multiplo da linha j a ser somado à linha i da matriz A.
  - Para k de j até m, faça
    - $A_{i,k} = A_{ik} + \mu\ A_{j,k}$ // faz a operação desejada entre as linhas // da matriz A.
  - $y_i = y_i + \mu\ y_j$ // atualiza o vetor de termos independentes.

Parte 2: Resolver o sistema triangular resultante

- Para i de m até 1, faça
  - $x_i = y_i$.
  - Para j de $i+1$ até m faça
    - $x_i = x_i - A_{i,j}\ x_j$.
  - $x_i = x_i / A_{i,i}$.

3.2 Algoritmo de busca binária

No nosso problema, dado um valor x^*, devemos encontrar a qual intervalo $]x_i; x_{i+1}]$ o ponto x^* pertence, isto é devemos encontrar o índice i tal que

$$x_i \leq x^* < x_{i+1}. \tag{11}$$

Dado um vetor ordenado $X = (x_0, x_1, \ldots x_n)$, com $x_0 < x_1 < \ldots < x_n$ e um valor de x^* (supondo $t_0 < x^* \leq t_n$), o seguinte algoritmo retorna o índice i que satisfaz (11).

- Faça $m = 0; M = n$;
- Enquanto $|M - m| > 1$, faça
  - k = valor arredondado (inteiro) de $(M + m)/2$;
  - se $x^* > x_k$, faça $m = k$ e $M = M$;
  - se $x^* \leq x_k$, faça $m = m$ e $M = k$;
- retorne $i = m$.