UNIVERSIDADE FEDERAL FLUMINENSE
EEIMVR - ESCOLA DE ENGENHARIA INDUSTRIAL
METALÚRGICA DE VOLTA REDONDA
PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM
COMPUTACIONAL CIÊNCIA E TECNOLOGIA

RESOLUÇÃO DE SISTEMAS NÃO - LINEARES, AUTOVALORES E AUTOVETORES

CAÍQUE DA SILVA MACHADO
PRISCILA CARRARO DE SOUZA
RAFAEL DE JESUS QUEIROZ

Prof.: Gustavo Benitez

VOLTA REDONDA
ABRIL DE 2019

SUMÁRIO

1 INTRODUÇÃO

Uma ampla gama de problemas na ciência e engenharia, frequentemente, resultam em sistemas de equações não- lineares, que consistem em, um sistema de n equações que possuem n incógnitas, sendo que, estas não podem ser escritas na forma $Ax + By + C = 0$. A forma genérica deste tipo de sistema apresenta todos os termos não nulos a esquerda para todas as equações:

$$\begin{cases} f_1(x_1, x_2, ..., x_n) = 0 \\ f_2(x_1, x_2, ..., x_n) = 0 \\ \qquad\qquad ... \\ f_n(x_1, x_2, ..., x_n) = 0 \end{cases} \tag{1}$$

O conjunto de equações não- lineares pode ser representado na forma matricial $f(x) = 0$

com $f = \begin{bmatrix} f_1 \\ f_2 \\ ... \\ f_n \end{bmatrix}$ e $x = \begin{bmatrix} x_1 \\ x_2 \\ ... \\ x_n \end{bmatrix}$

Além das soluções, outra questão de interesse, a respeito dos sistemas de equações sejam estes lineares, ou não, é a obtenção de seus respectivos autovalores e autovetores.

Neste trabalho será estudado e implementado 3 métodos numéricos para a obtenção das raízes de um sistema de equações não- lineares, sendo que, os métodos utilizados serão o de Newton, Jacobi ou iteração e o método do Gradiente. Para a obtenção dos autovalores e autovetores será utilizado técnicas de aproximação com o objetivo de se evitar a expansão do determinante em polinômios.

2 PROBLEMA PROPOSTO

Para este trabalho foram propostos dois problemas, que serão resolvidos através do uso de uma ferramenta matemática adequada, sendo que, o primeiro consiste na aplicação de métodos numéricos na resolução de um sistema não- linear e o segundo problema solicita o calculo dos autovalores e autovetores de um sistema linear de equações, observando- se que este é definido e solucionado numericamente no trabalho sobre resolução de sistemas lineares [1].

Para o primeiro problema, tem- se que, seja o seguinte sistema não- linear:

$$\begin{cases} (x-1)^2 + (y-1)^2 + (z-1)^2 = 1 \\ 2x^2 + (y-1)^2 = 4z \\ 3x^2 + 2z^2 = 4y \end{cases} \tag{2}$$

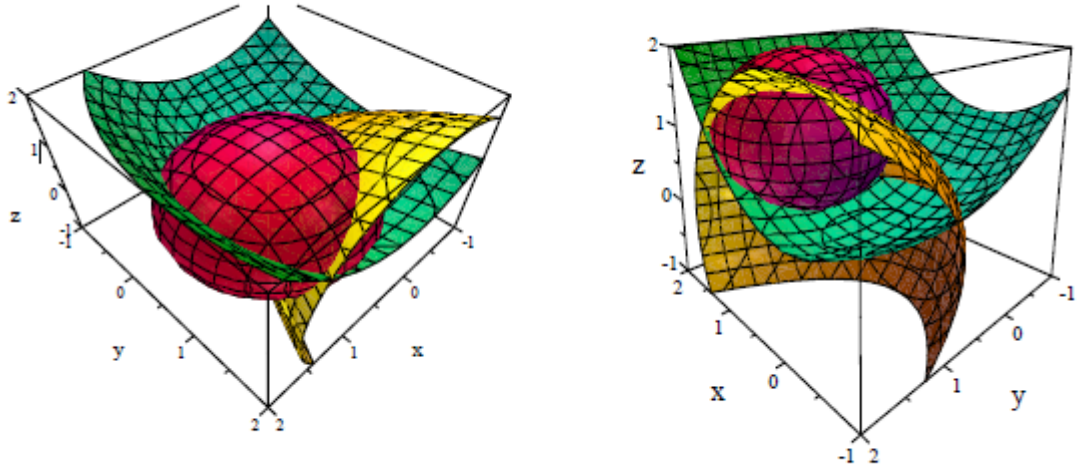

Figura 1 – Gráfico gerado pelo $MatLab$ da equação 2

Para este deve- se encontrar a sua solução utilizando os métodos: de Newton, da Iteração e do Gradiente.

No segundo problema deve- se usar métodos e técnicas que permitam encontrar os autovalores e autovetores correspondentes sem realizar a expansão do determinante.

3 MÉTODOS NUMÉRICOS

3.1 MÉTODO DE NEWTON

Seja $f(x) = 0$ um sistema não-linear escrito de forma abreviada, de acordo com a definição apresentada, tem- se, que este pode ser solucionado através de um método de aproximações sucessivas [2]. Suponha que tem- se p aproximações $x^{(p)} = (x_1^{(p)}, x_2^{(p)}, ..., x_n^{(p)})$ de uma das raízes isoladas $x = (x_1, x_2, ..., x_n)$ do vetor deste sistema. A raiz exata deste pode ser representada como:

$$x = x^{(p)} + \epsilon^{(p)} \tag{3}$$

onde $\epsilon^{(p)} = (\epsilon_1^{(p)}, \epsilon_2^{(p)}, ..., \epsilon_n^{(p)})$ é a correção de erro [2]. Substituindo 3 na definição apresentada, tem- se:

$$f(x^{(p)} + \epsilon^{(p)}) = 0 \tag{4}$$

Na suposição que $f(x)$ é continuamente diferenciável em algum domínio convexo contendo x e $x^{(p)}$, expandi- se o lado esquerdo da equação 4 em série de potência e desprezando-se as potências maiores que 1 [2]:

$$f(x^{(p)} + \epsilon^{(p)}) = f(x^{(p)}) + f'(x^{(p)})\epsilon^{(p)} = 0 \tag{5}$$

Expandindo a equação 5:

$$
\left.
\begin{aligned}
& f_1\left(x_1^{(p)}+\varepsilon_1^{(p)},\ x_2^{(p)}+\varepsilon_2^{(p)},\ \ldots,\ x_n^{(p)}+\varepsilon_n^{(p)}\right)= \\
&\quad = f_1\left(x_1^{(p)},\ x_2^{(p)},\ \ldots,\ x_n^{(p)}\right)+f'_{1x_1}\left(x_1^{(p)},\ x_2^{(p)},\ \ldots,\ x_n^{(p)}\right)\varepsilon_1^{(p)}+ \\
&\quad\quad + f'_{1x_2}\left(x_1^{(p)},\ x_2^{(p)},\ \ldots,\ x_n^{(p)}\right)\varepsilon_2^{(p)}+\ldots \\
&\quad\quad\quad \ldots+f'_{1x_n}\left(x_1^{(p)},\ x_2^{(p)},\ \ldots,\ x_n^{(p)}\right)\varepsilon_n^{(p)}=0, \\
& f_2\left(x_1^{(p)}+\varepsilon_1^{(p)},\ x_2^{(p)}+\varepsilon_2^{(p)},\ \ldots,\ x_n^{(p)}+\varepsilon_n^{(p)}\right)= \\
&\quad = f_2\left(x_1^{(p)},\ x_2^{(p)},\ \ldots,\ x_n^{(p)}\right)+f'_{2x_1}\left(x_1^{(p)},\ x_2^{(p)}\ \ldots,\ x_n^{(p)}\right)\varepsilon_1^{(p)}+ \\
&\quad\quad + f'_{2x_2},\ \left(x_1^{(p)},\ x_2^{(p)},\ \ldots,\ x_n^{(p)}\right)\varepsilon_2^{(p)}+\ldots \\
&\quad\quad\quad \ldots+f'_{2x_n}\left(x_1^{(p)},\ x_2^{(p)},\ \ldots,\ x_n^{(p)}\right)\varepsilon_n^{(p)}=0, \\
& \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
& f_n\left(x_1^{(p)}+\varepsilon_1^{(p)},\ x_2^{(p)}+\varepsilon_2^{(p)},\ \ldots,\ x_n^{(p)}+\varepsilon_n^{(p)}\right)= \\
&\quad = f_n\left(x_1^{(p)},\ x_2^{(p)},\ \ldots,\ x_n^{(p)}\right)+f'_{nx_1}\left(x_1^{(p)},\ x_2^{(p)},\ \ldots,\ x_n^{(p)}\right)\varepsilon_1^{(p)}+ \\
&\quad\quad + f'_{nx_2}\left(x_1^{(p)},\ x_2^{(p)},\ \ldots,\ x_n^{(p)}\right)\varepsilon_2^{(p)}+\ldots \\
&\quad\quad\quad \ldots+f'_{nx_n}\left(x_1^{(p)},\ x_2^{(p)},\ \ldots,\ x_n^{(p)}\right),\ \varepsilon_n^{(p)}=0
\end{aligned}
\right\}
$$

Figura 2 – Expansão da equação 5. Fonte: Demidovich, pag.:460, eq.:4

Deve- se entender $f'(x)$ como sendo a matriz Jacobiana das funções $f_1, f_2, ..., f_n$ com respeito as variáveis $x_1, x_2, ..., x_n$ [2].

$$
f'(x) = W(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & ... & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & ... & \frac{\partial f_2}{\partial x_n} \\ ... & ... & ... & ... \\ \frac{\partial f_1}{\partial x_1} & \frac{\partial f_n}{\partial x_n} & ... & \frac{\partial f_n}{\partial x_n} \end{bmatrix}
$$

O sistema expandido é um sistema linear das correções $\epsilon^{(p)}(i = 1, 2, ..., n)$ para a matriz $W(x)$, sendo que este pode ser escrito como [2]:

$$
f(x^{(p)}) + W(x^{(p)})\epsilon^{(p)} = 0 \tag{6}
$$

onde, assumi- se que a matriz $W(x^{(p)})$ é não singular, assim [2]:

$$
\epsilon^{(p)} = -W^{-1}(x^{(p)})f(x^{(p)}) \tag{7}
$$

consequentemente o método de newton é definido como [2]:

$$
x^{(p+1)} = x^{(p)} - W^{-1}(x^{(p)})f(x^{(p)}) \qquad (p = 0, 1, 2, ...) \tag{8}
$$

3.1.1 Convergência do método de newton

Para este método, observa- se que, foram desenvolvidos teoremas com o objetivo de avaliar a existência de raízes no sistema de interesse e avaliar também a possibilidade de

convergência deste.

Teorema 1: Seja um sistema não linear de equações com coeficientes reais, onde as funções $f(x)$ são definidas e continuas junto com suas derivadas parciais de primeira e segunda ordem num domínio Ω. Ou seja, $f(x) \in C^2(\Omega)$ [3].

$$\text{f(x) = 0 com f = } \begin{bmatrix} f_1 \\ f_2 \\ ... \\ f_n \end{bmatrix} \text{ e x = } \begin{bmatrix} x_1 \\ x_2 \\ ... \\ x_n \end{bmatrix}$$

Suponha x^0 e o fecho de sua vizinhança $\overline{V}(x^0) = ||x - x^0||_m \leq \overline{\Delta x} \subset \Omega$ serem pontos de Ω,

onde $||\ ||_m$ é a m-norma e as seguintes condições serem válidas [3]:

1. A matriz Jacobiana $W(x)$ em x^0 tem inversa $W(x^0)^{-1}$ e $||w(x^0)^{-1}||_m \leq A_0$, com $||W(x^0)||_m = max_i \frac{1}{|\det(W(x^0))|} \sum_{j=1}^{n} |\overline{W_{ij}}|$, $\overline{W_{ij}}$ são os cofatores do elemento W_{ij}
2. $||W(x^0)^{-1} f(x^0)||_m \leq B_0 \leq \frac{\overline{\Delta x}}{2}$
3. $\sum_{j=1}^{n} |\frac{\partial^2 f_i(x)}{\partial x_j \partial x_k}| \leq C$ com $(i, j = 1, ..., n)$ e $x \in \overline{V}(x^0)$
4. as constantes A_0, B_0 e C satisfazem a desigualdade $\mu_0 = 2nA_0B_0C \leq 1$.

Então o método de Newton $x^{p+1} = x^p - W(x^p)^{-1} f(x^p), p = 0, 1, 2, ...$ converge para esta escolha de aproximação inicial x^0 e $x^* = \lim_{p \to \infty} x^p$ é a solução do sistema $f(x) = 0$ tal que $||x^* - x^0||_m \leq 2B_0 \leq \overline{\Delta x}$ [3].

Note que, sempre que sejam verificadas todas as hipóteses do teorema o método de Newton converge para uma solução x^* que é raiz do sistema na vizinhança de x^0.

Teorema 2: Se as condições do teorema 1 são verificadas, então existirá no domínio $||x - x^0||_m \leq 2B_0 \leq \overline{(}\Delta x)$ uma única solução do sistema $f(x) = 0$ [3].

Teorema 3: Se as condições do Teorema 1 são verificadas, então a seguinte desigualdade se verifica para as aproximações sucessivas x^p [3]:

$$||x^* - x^p||_m \leq \left(\frac{1}{2}\right)^{p-1} (\mu_0)^{2p-1} (B_0)$$

onde x^* é a solução do sistema $f(x^*) = 0$ e $\mu_0 = 2nA_0B_0C \leq 1$.

Teorema 4: Se as condições do teorema 1 são verificadas e $\frac{2}{\mu_0}B_0C \leq \overline{\Delta x}$ onde $\mu_0 = 2nA_0B_0C < 1$, então o Método de Newton converge para qualquer escolha da aproximação inicial x_0 que pertença ao domínio $||\tilde{x}^0 - x^0||_m \leq \frac{1-\mu_0}{2\mu_0}B_0$ [3].

Note que se $2B_0 < \overline{\Delta x}$ e $\mu_0 < 1$ então para a aproximação inicial x^0 sempre há uma vizinhança e qualquer ponto desta pode ser escolhido como aproximação inicial para que o Método de Newton seja convergente para a solução procurada x^* [3].

3.2 MÉTODO DE JACOBI

Dado um sistema não linear de equações de um tipo especifico:

$$\begin{cases} x_1 = \varphi(x_1, x_2, ..., x_n), \\ x_2 = \varphi(x_1, x_2, ..., x_n), \\ \qquad ... \\ x_n = \varphi(x_1, x_2, ..., x_n) \end{cases} \tag{9}$$

onde as funções $\varphi_1, \varphi_2, ..., \varphi_n$ são reais, definidas e continuas em alguma vizinhança δ de uma isolada solução $({x_1}^*, {x_2}^*, ..., {x_n}^*)$ deste sistema [2].

Introduzindo os vetores $x = (x_1, x_2, ..., x_n)$ e $\varphi(x) = (\varphi_1, \varphi_2, ..., \varphi_n)$, pode- se escrever o sistema 9 abreviadamente como [2]:

$$x = \varphi(x) \tag{10}$$

Na procura de uma raiz para o vetor $x^* = ({x_1}^*, {x_2}^*, ..., {x_n}^*)$ da equação 10 é frequentemente conveniente usar o método de Jacobi ou Iteração [2]:

$$x^{(p+1)} = \varphi(x^{(p)}) \qquad (p = 0, 1, 2, ...) \tag{11}$$

Onde a aproximação inicial $x^{(0)} \approx x^*$. Observe que, se este processo iterativo converge o valor do limite:

$$\xi = \lim_{p\to\infty} x^{(p)} \tag{12}$$

é definitivamente a raiz da equação 10. De fato, assumindo que a relação 12 é valida, e passando o limite em 11 como $p \to \infty$, tem- se, por virtude da continuidade da função $\varphi(x)$ [2]:

$$\lim_{p\to\infty} x^{(p+1)} = \varphi(\lim_{p\to\infty} x^{(p)})$$

isto é:

$$\xi = \varphi(\xi)$$

Portanto, ξ é a raiz da equação vetorial 10. Se, por outro lado, todas as aproximações $x^{(p)}(p = 0, 1, 2, ...)$ pertencem ao domínio ω e x^* é a única raiz do sistema 10 em ω, então, claramente $\xi = x^*$ [2]. O método da iteração ou Jacobi pode, também, ser aplicado para sistemas genéricos:

$$f(x) = 0 \tag{13}$$

onde $f(x)$ é uma função vetorial definida e continua na vizinhança ω de um isolado vetor de raízes x^*. Por exemplo, reescrevendo este sistema como $x = x + \Lambda f(x)$ onde Λ é uma matriz não singular. Utilizando- se a notação [2]:

$$x + \Lambda f(x) = \varphi(x) \tag{14}$$

tem se que,

$$x = \varphi(x) \tag{15}$$

O método básico da iteração 11 é diretamente aplicável para esta equação. Se a função $f(x)$ possui derivada continua $f'(x)$ em ω, então através da formula 14, tem- se [2]:

$$\varphi'(x) = E + \Lambda f'(x) \tag{16}$$

e pode ser provado que o processo iterativo converge rapidamente se a norma de $\varphi(x)$ é pequena [2]. Logo é conveniente escolher a matriz Λ tal que $\varphi'(x^0) = E + \Lambda f'(x^0) = 0$ e se a matriz $f'(x^0)$ é não singular segue que $\Lambda = -[f'(x^0)]^{-1} \longrightarrow \varphi = x - [f'(x^0)]^{-1} f(x) \longrightarrow x = x - [f'(x^0)]^{-}1 f(x)$ [3]. Aplicando o método da iteração a este sistema segue:

$$x^{k+1} = x^k - [f'(x^0)]^{-1} f(x^k) \tag{17}$$

Note que a equação anterior coincide com o método de Newton modificado para o sistema 13, já que $[f'(x^0)]^{-1} = W(x^0)^{-1}$(Jacobiana) [3]. Ou seja, $x^{k+1} = x^k - W(x^0)^{-1} f(x^k)$. Se a matriz $f'(x^0)$ é singular $(det(f'(x^0)) = 0)$, então deve ser escolhida uma aproximação inicial diferente de x^0 [3].

3.2.1 Convergência do Método de Jacobi

Teorema: Seja um sistema não linear de equações $x = \varphi(x)$ com funções $\varphi(x)$ e $\varphi'(x)$ continuas em Ω tais que em Ω se verifica a desigualdade $||\varphi'(x)||_i \leq q \leq 1$ (q é uma constante). Se todas as aproximações sucessivas $X^{k+1} = \varphi(x^k)$ pertencem a Ω, então este processo iterativo

converge e seu limite é uma solução do sistema $x = \varphi(x)$ em Ω [3].
Ou seja, $\lim_{k\to\infty} x^k = x^* = \varphi(x^*)$ [3].
Aqui as normas são $||\varphi'(x)||_i = \max_{x\in\Omega} ||\varphi'(x)||_m$ e $||\varphi'(x)||_m = \max_i \sum_{j=1}^{n} \left|\frac{\partial\varphi_i(x)}{\partial x_j}\right|$.
Corolário: O método da iteração converge se $\sum_{j=1}^{n} \left|\frac{\partial\varphi_i(x)}{\partial x_j}\right| \leq q \leq 1 \quad \forall i = 1, ..., n$ quando $x \in \Omega$.
Estimativa de erro na aproximação k para o Método de Jacobi ou Iteração: $||x^* - x^k||_m \leq \frac{q^k}{1-q}||x^1 - x^0||_m = \frac{q^k}{1-q}||\varphi(x^0) - x^0||_m$, sendo que, $x^1 = \varphi(x^0)$.

3.3 MÉTODO DO GRADIENTE

Suponha uma sistema de equações:

$$\begin{cases} f_1(x_1, x_2, ..., x_n) = 0 \\ f_2(x_1, x_2, ..., x_n) = 0 \\ \quad\quad ... \\ f_n(x_1, x_2, ..., x_n) = 0 \end{cases} \tag{18}$$

ou na forma matricial:

$$f(x) = 0 \text{ onde } f = \begin{bmatrix} f_1 \\ f_2 \\ \cdot \\ \cdot \\ f_n \end{bmatrix} \tag{19}$$

Supondo- se que as funções f_i são reais e continuamente diferenciáveis em seu domínio comum de definição [2]. Considere a função:

$$U(x) = \sum_{i=1}^{n} [f_i(x)]^2 = (f(x), f(x)) \tag{20}$$

Note que cada solução do sistema 3 anula a função $U(x)$ e vice-versa, cada x que anula $U(x)$ é solução do sistema 3. Suponha que 3 tem somente uma solução isolada que corresponde ao ponto de mínimo absoluto de $U(x)$. Logo o problema se reduz a encontrar o mínimo desta função [3]. De maneira breve, pode- se observar- se que, executando $x^{(0)}$ como solução aproximada inicial de $U(x)$ e $x^{(p)}$ as aproximações, temos a seguinte equação:

$$x^{(p+1)} = x^{(p)} - \lambda^p \nabla U(x^{(p)}) \qquad (p = 0,1,...) \tag{21}$$

O surgimento do gradiente é devido a $x^{(p)}$ mover- se na normal, ao longo do caminho próximo á superfície de nível. Após avaliar- se algumas propriedades tem- se que $x^{(p+1)} = x^{(p)} - \mu_p W'_p f^p$. Sabendo- se que, W'_p é conhecido como wronskiano e está transposto em x^p, $f^{(p)}$ é cada função f analisada no ponto $x^{(p)}$, da mesma forma [3]:

$$\mu_p = \frac{(f^p, W_p W'_p f^{(p)})}{(W_p W'_p f^{(p)}, W_p W'_p f^{(p)})} \tag{22}$$

3.4 AUTOVALORES E AUTOVETORES

Problemas de autovalores e autovetores são comuns em contextos envolvendo condições de contorno na engenharia incluindo vibrações, elasticidade e outros sistemas oscilantes. As técnicas descritas a seguir apresentam soluções para tais problemas.

Dada uma matriz $A_{m\times n}$, um vetor não nulo x é dito autovetor da matriz $A_{m\times n}$ caso exista um escalar λ denominado autovalor da matriz $A_{m\times n}$ e correspondente ao autovetor x, obedecendo a equação I [4].

$$\begin{aligned} &A_{m\times n}x = \lambda x && (I) \\ &A_{m\times n} - \lambda E)x = 0 && (II) \\ &\det(A_{m\times n} - \lambda E)x = 0 && (III) \end{aligned} \tag{23}$$

Para encontrar os autovalores da matriz $A_{m\times n}$, é necessário determinar as raízes do polinômio característico. Esta técnica é conhecida como expansão do determinante em polinômios [4].

$$P_n(\lambda) = \det(A_{m\times n} - \lambda E) = 0 \qquad (VI) \tag{24}$$

Então, encontrar os autovetores $x \neq 0$ associados a cada autovalor que são solução do sistema linear homogêneo representado na equação III [4].

A segunda técnica consiste no método das potências para encontrar o maior autovalor em valor absoluto e seu autovetor correspondente sem expandir o determinante. Supondo que $[\lambda 1] \geq [\lambda 2] \geq ... \geq [\lambda n]$ e adotando y como um vetor arbitrário. Assim, a cada iteração, o vetor y é multiplicado pela matriz $A_{m\times n}$ e normalizado. De acordo com as premissas:

A matriz $A_{m\times n}$em um autovalor que é estritamente maior em magnitude que os outros autovalores.

O vetor inicial tem um componente não-nulo na direção de um autovetor associado com o autovalor dominante dos autovalores.

4 RESULTADOS NUMÉRICOS

4.1 SOLUÇÃO DO SISTEMA PROPOSTO

Neste trabalho o problema proposto será numericamente solucionado, começando- se primeiramente, pelo sistema de equações não- lineares definido por:

$$\begin{aligned} &(x-1)^2+(y-1)^2+(z-1)^2=1 \\ &2x^2+(y-1)^2=4z \\ &3x^2+2z^2=4y \end{aligned} \tag{25}$$

Por motivos de praticidade, a partir das definições estudadas foram desenvolvidos programas, em um ambiente computacional adequado, que realizam a aplicação direta dos métodos estudados. Dentro deste ambiente foram obtidos as informações necessárias para se avaliar o desempenho de cada método no calculo das raízes. Os algoritmos desenvolvidos encontram- se no apêndice deste trabalho. As raízes do sistema 25 são obtidas após aplicar- se diretamente os métodos estudados. Para a primeira aproximação foi estimado o valor 0.2 para a raiz procurada. O primeiro método a ser aplicado é o de Newton e as raízes encontradas estão dispostas na tabela 1, abaixo:

Tabela 1 – Solução do sistema proposto através do método de Newton

Iteração	x	y	z
0	0.200000000	0.200000000	0.200000000
1	0.200000000	0.200000000	0.200000000
2	0.690000000	0.211666667	0.273333333
3	0.644506997	0.356034279	0.305122289
4	0.649396784	0.364822768	0.311689356

Tabela 2 – Valores da função avaliados no ponto x_k

f(x)	f(y)	f(z)
0.920000000	-0.080000000	-0.600000000
0.920000000	-0.080000000	-0.600000000
0.245613889	0.480336111	0.731055556
0.023922158	0.024981234	0.008229915
0.000144274	0.000125058	0.000157983

Tabela 3 – Erro de aproximação em cada iteração

Iteração	0	1	2	3	4
Erro	1.101271992	0.908565456	0.035553667	0.000247816	0.000000010

Abaixo na figura 3, tem- se o gráfico do erro para cada iteração do método de Newton.

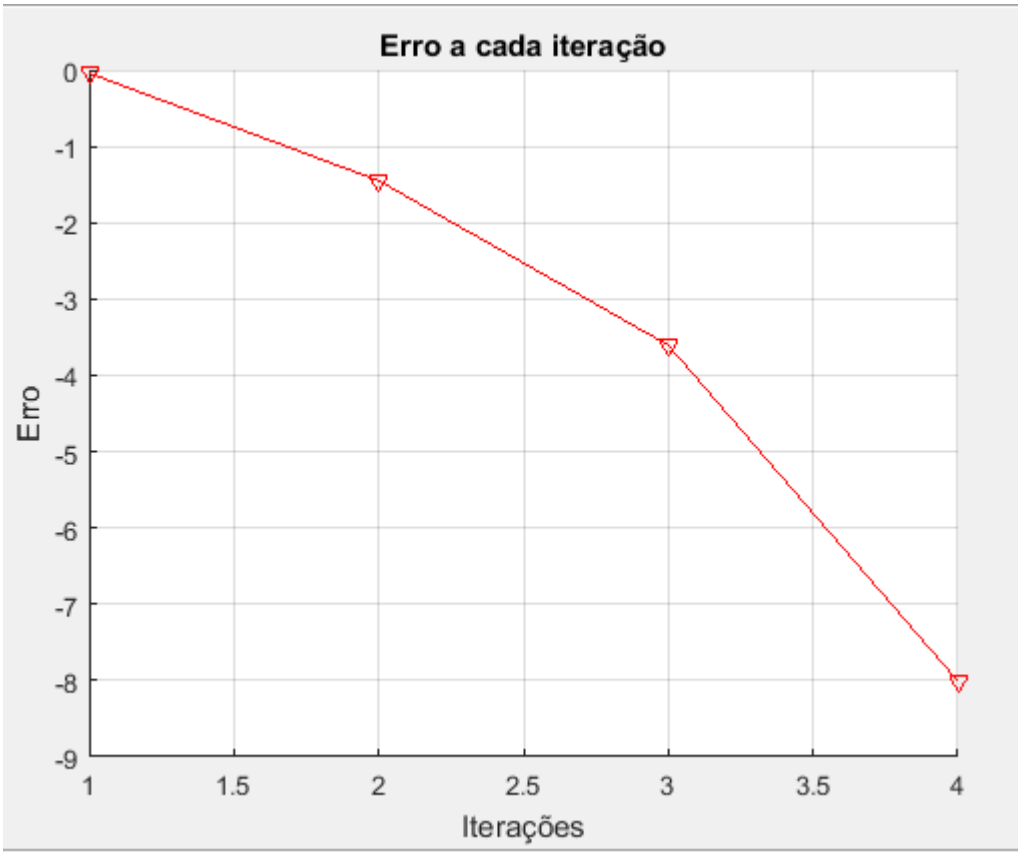

Figura 3 – Gráfico gerado pelo $MatLab$ do Erro de aproximação do método de Newton

O tempo médio apresentado pelo método foi de 4 segundos e este converge após 5 iterações.

O segundo método aplicado é o de Jacobi ou iteração e o valor das raízes encontradas pelo programa para este estão dispostos na tabela 4, abaixo:

Tabela 4 – Solução do sistema proposto através do método de Jacobi

iteração	x	y	z
0	0.200000000	0.200000000	0.200000000
1	0.200000000	0.200000000	0.200000000
2	0.690000000	0.211666667	0.273333333
3	0.631353472	0.384667477	0.312487731
4	0.655172437	0.354580021	0.310760907
5	0.647245253	0.368052574	0.311792741
6	0.650177742	0.363653142	0.311648867
7	0.649146955	0.365317423	0.311728009
8	0.649520003	0.364738803	0.311707436
9	0.649387298	0.364949906	0.311716332
10	0.649435017	0.364875186	0.311713519

Tabela 5 – Valores da função avaliados no ponto x_k

f(x)	f(y)	f(z)
0.920000000	-0.080000000	-0.600000000
0.245613889	0.480336111	0.731055556
-0.012792504	-0.074102398	-0.147551122
0.010523524	0.032025165	0.062577366
-0.002577308	-0.009960577	-0.021001616
0.001140218	0.003804048	0.007830753
-0.000361834	-0.001306524	-0.002745679
0.000139671	0.000479514	0.000996543
-0.000047703	-0.000168980	-0.000353893
0.000017618	0.000061137	0.000127415

Tabela 6 – Erro de aproximação em cada iteração

Iteração	0	1	2	3	4	5
Erro	1.10127199	0.90856546	0.16560842	0.07107941	0.02338640	0.00878018
Iteração	6	7	8	9	10	
Erro	0.00306214	0.00111469	0.00039506	0.00014242	0.00005077	

Abaixo na figura 4, tem- se o gráfico do erro para cada iteração do método de Jacobi ou Iteração.

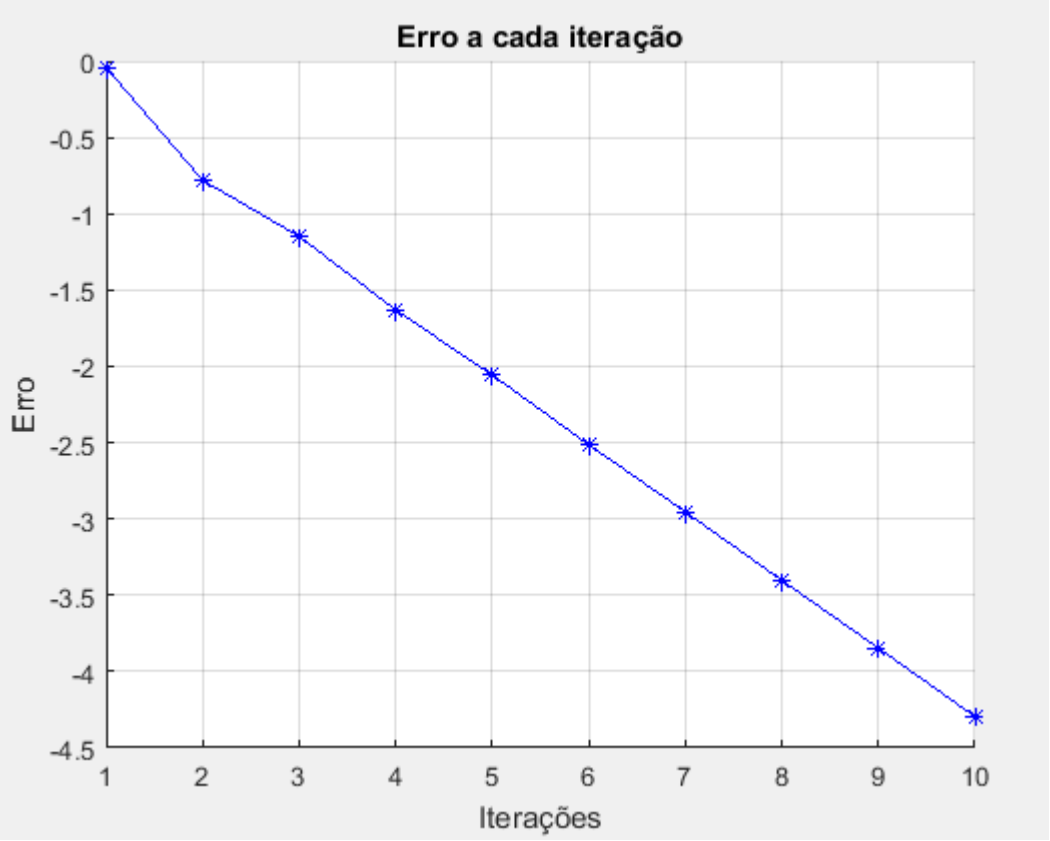

Figura 4 – Gráfico gerado pelo $MatLab$ do Erro de aproximação do método da Iteração

O tempo médio apresentado pelo método de Jacobi foi de 2.5 segundos e este converge após 11 iterações.

O último método aplicado é o do gradiente e o valor das raízes encontradas pelo programa para este estão dispostos na tabela 7, abaixo:

Tabela 7 – Solução do sistema proposto através do método do gradiente

Iteração	x	y	z	Iteração	x	y	z
0	0.200000000	0.200000000	0.200000000	29	0.646272178	0.361827348	0.311297704
1	0.200000000	0.200000000	0.200000000	30	0.647219727	0.362059271	0.311260604
2	0.405027182	0.104029830	0.348317536	31	0.647087262	0.362618480	0.311404112
3	0.426446489	0.219464213	0.302314734	32	0.647789506	0.362791579	0.311377317
4	0.500783679	0.201743070	0.292302249	33	0.647690863	0.363205509	0.311483535
5	0.505593460	0.252385273	0.306519609	34	0.648211515	0.363334515	0.311464054
6	0.551270454	0.250763287	0.296804350	35	0.648138122	0.363641128	0.311542730
7	0.550829830	0.281723820	0.305180928	36	0.648524252	0.363737170	0.311528494
8	0.581563363	0.283313278	0.300458875	37	0.648469680	0.363964408	0.311586801
9	0.579913088	0.303497546	0.305758516	38	0.648756107	0.364035853	0.311576357
10	0.601313053	0.305829554	0.303217806	39	0.648715549	0.364204331	0.311619586
11	0.599576439	0.319527776	0.306773678	40	0.648928051	0.364257448	0.311611901
12	0.614801093	0.321785679	0.305322434	41	0.648897918	0.364382397	0.311643960
13	0.613290452	0.331333400	0.307788230	42	0.649055593	0.364421870	0.311638294
14	0.624269886	0.333267232	0.306921169	43	0.649033211	0.364514555	0.311662074
15	0.623045365	0.340043522	0.308666716	44	0.649150215	0.364543881	0.311657889
16	0.631033575	0.341610475	0.308129224	45	0.649133593	0.364612644	0.311675532
17	0.630074168	0.346480490	0.309381938	46	0.649220423	0.364634426	0.311672436
18	0.635920489	0.347712436	0.309038033	47	0.649208081	0.364685448	0.311685527
19	0.635182819	0.351243529	0.309945566	48	0.649272521	0.364701623	0.311683236
20	0.639478856	0.352194597	0.309719387	49	0.649263357	0.364739484	0.311692950
21	0.638918085	0.354771110	0.310381227	50	0.649311183	0.364751495	0.311691252
22	0.642083811	0.355496758	0.310228903	51	0.649304380	0.364779592	0.311698461
23	0.641660608	0.357385340	0.310713859	52	0.649339875	0.364788509	0.311697203
24	0.643998048	0.357934634	0.310609186	53	0.649334825	0.364809361	0.311702553
25	0.643680209	0.359323559	0.310965753	54	0.649361170	0.364815981	0.311701621
26	0.645408527	0.359737087	0.310892612	55	0.649357421	0.364831457	0.311705591
27	0.645170613	0.360761024	0.311155434	56	0.649376974	0.364836371	0.311704900
28	0.646449851	0.361071142	0.311103626	57	0.649374191	0.364847857	0.311707846

Tabela 8 – Valores da função avaliados no ponto x_k

Iteração	f(x)	f(y)	f(z)	Iteração	f(x)	f(y)	f(z)
0	0.920000000	-0.080000000	-0.600000000	29	0.006698559	-0.002591026	-0.000493687
1	0.920000000	-0.080000000	-0.600000000	30	0.005784249	-0.000287294	0.002209369
2	0.581445235	-0.262413561	0.318671945	31	0.004966900	-0.001917396	-0.000363107
3	0.424964474	-0.236309607	-0.149498632	32	0.004288004	-0.000212207	0.001639084
4	0.387267168	-0.030426284	0.116261808	33	0.003683874	-0.001420010	-0.000267691
5	0.284280660	-0.155901161	-0.054758309	34	0.003179827	-0.000156941	0.001216158
6	0.257197980	-0.018063519	0.084829837	35	0.002732808	-0.001052257	-0.000197691
7	0.200448056	-0.097976035	-0.030383975	36	0.002358605	-0.000116176	0.000902443
8	0.178086862	-0.011763754	0.061945794	37	0.002027572	-0.000780077	-0.000146186
9	0.143559921	-0.065320016	-0.018116077	38	0.001749780	-0.000086059	0.000669700
10	0.126329316	-0.007843841	0.045296023	39	0.001504493	-0.000578483	-0.000108204
11	0.103944208	-0.045068454	-0.011415207	40	0.001298280	-0.000063782	0.000497007
12	0.090929784	-0.005354302	0.033242016	41	0.001116448	-0.000429087	-0.000080147
13	0.075816431	-0.031787540	-0.007490875	42	0.000963374	-0.000047289	0.000368861
14	0.066063968	-0.003726312	0.024470955	43	0.000828537	-0.000318329	-0.000059398
15	0.055579059	-0.022753258	-0.005067225	44	0.000714913	-0.000035072	0.000273763
16	0.048298160	-0.002633386	0.018055454	45	0.000614901	-0.000236191	-0.000044037
17	0.040886179	-0.016453089	-0.003507220	46	0.000530559	-0.000026016	0.000203187
18	0.035461396	-0.001883327	0.013343872	47	0.000456364	-0.000175263	-0.000032659
19	0.030151656	-0.011982880	-0.002469971	48	0.000393760	-0.000019301	0.000150807
20	0.026114659	-0.001359294	0.009873432	49	0.000338711	-0.000130062	-0.000024226
21	0.022274521	-0.008771950	-0.001762470	50	0.000292242	-0.000014322	0.000111932
22	0.019272594	-0.000987940	0.007311774	51	0.000251393	-0.000096523	-0.000017973
23	0.016476106	-0.006445163	-0.001270149	52	0.000216902	-0.000010627	0.000083079
24	0.014245018	-0.000721840	0.005418053	53	0.000186588	-0.000071636	-0.000013336
25	0.012198290	-0.004748287	-0.000922203	54	0.000160987	-0.000007887	0.000061664
26	0.010540702	-0.000529518	0.004016583	55	0.000138490	-0.000053167	-0.000009896
27	0.009037198	-0.003505030	-0.000673331	56	0.000119487	-0.000005853	0.000045769
28	0.007806008	-0.000389599	0.002978592	57	0.000102792	-0.000039460	-0.000007344

Tabela 9 – Erro de aproximação em cada iteração

Iteração	Erro	Iteração	Erro	Iteração	Erro	Iteração	Erro
0	1.10127199	15	0.05162990	30	0.00533651	45	0.00056873
1	0.71308572	16	0.04421181	31	0.00459550	46	0.00048995
2	0.50871099	17	0.03793569	32	0.00395715	47	0.00042209
3	0.40548542	18	0.03253940	33	0.00340807	48	0.00036363
4	0.32881475	19	0.02795188	34	0.00293506	49	0.00031327
5	0.27142806	20	0.02400433	35	0.00252803	50	0.00026989
6	0.22517085	21	0.02063664	36	0.00217737	51	0.00023251
7	0.18891955	22	0.01773740	37	0.00187554	52	0.00020031
8	0.15875877	23	0.01525768	38	0.00161550	53	0.00017257
9	0.13443345	24	0.01312231	39	0.00139162	54	0.00014867
10	0.11386778	25	0.01129246	40	0.00119875	55	0.00012809
11	0.09696353	26	0.00971646	41	0.00103266	56	0.00011035
12	0.08255115	27	0.00836406	42	0.00088957	57	0.00009507
13	0.07054900	28	0.00719916	43	0.00076634	60	
14	0.06026956	29	0.00619850	44	0.00066017	61	

Abaixo na figura 5, tem- se o gráfico do erro para cada iteração do método do Gradiente.

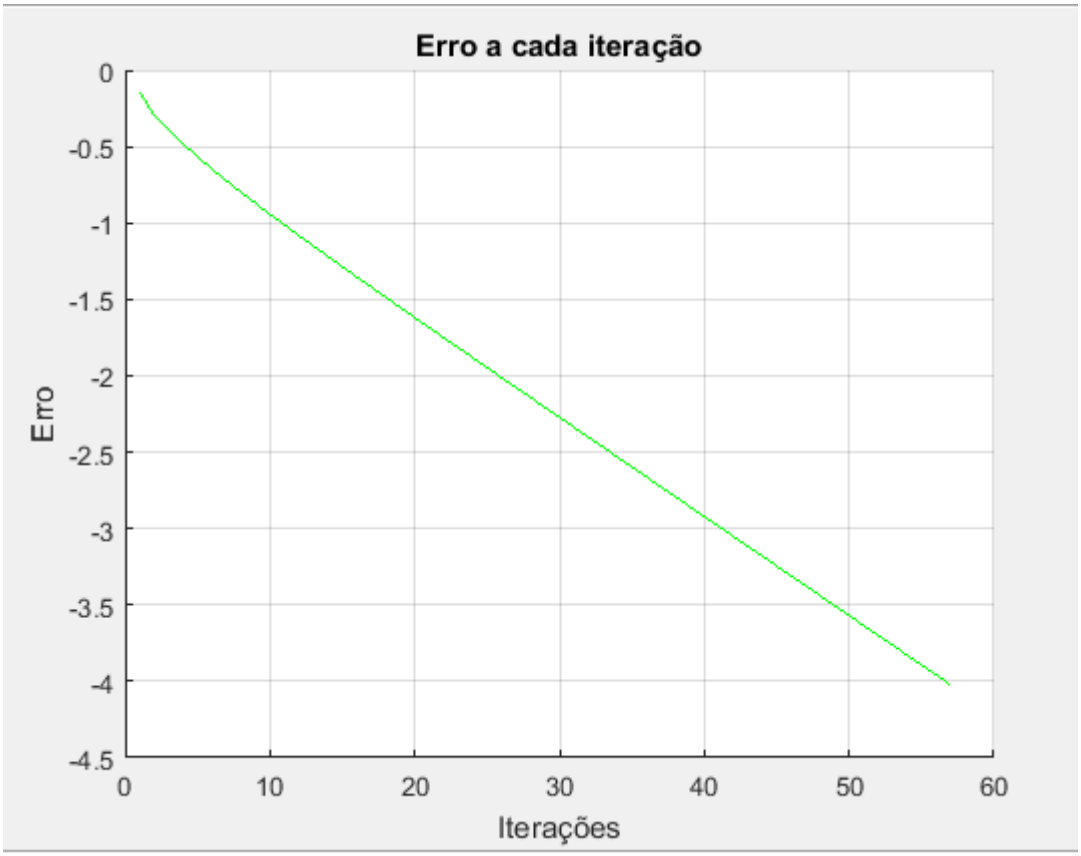

Figura 5 – Gráfico gerado pelo $MatLab$ do Erro de aproximação do método do Gradiente

O tempo médio apresentado pelo método de gradiente foi de 2.1 segundos e este converge após 57 iterações.

Após a aplicação dos métodos, observa- se que, todos apresentaram boa convergência para o sistema 25 e que o método do gradiente apesar de ser o mais rápido é também o que realizou um número maior de iterações entre os métodos avaliados. Comparando- se o desempenho dos métodos a partir do erro de aproximação, tem- se que, o método de Newton mostrou- se como a melhor escolha, uma vez que, o erro deste está muito próximo de zero para o palpite inicial sugerido. Além disso outro fato importante que sugere o método de newton como o mais adequado é o seu baixo número de iterações realizadas. Para um palpite inicial nulo ou maior que 1 observou- se que o erro de aproximação para a solução é significativo, porem se for verificado as condições de convergência dos métodos aplicados, tem- se que mesmo com um palpite inicial ruim ou distante da solução do sistema estes alcançaram a convergência, apesar de um longo tempo de espera.

4.2 AUTOVALORES E AUTOVETORES

Usando- se os métodos estudados em sala de aula para encontrar os autovalores e autovetores da matriz $A_{4\times4}$ do Problema 1 do Trabalho 1 [1]. Esta matriz $A_{4\times4}$ é uma matriz de Hilbert quadrada de ordem 4.

$$A = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \end{bmatrix}$$

Para encontrar os autovalores e autovetores da matriz$A_{4\times4}$, foi implementado um algoritmo, que será apresentado no apêndice deste trabalho, baseado no método das potências, na qual consiste em encontrar primeiramente a aproximação do maior autovalor em valor absoluto e o seu autovetor sem expandir o determinante, o segundo maior autovalor em valor absoluto e assim sucessivamente. Assim nas tabelas a seguir tem- se os valores dos autovalores calculados por este:

Tabela 10 – Autovalores para $\lambda1$

Iteração	Lambda 1				Erro
1	2,083333333	1,283333333	0,950000000	0,759523810	1,269047619
2	1,551142857	1,448453927	1,401963241	1,374988804	0,175089588
3	1,505783608	1,494055472	1,488381977	1,484985800	0,049164507
4	1,500839941	1,499516528	1,498869743	1,498480539	0,006124974
5	1,500284791	1,500135573	1,500062555	1,500018587	0,000698689
6	1,500222229	1,500205406	1,500197172	1,500192214	7,88787E-05
7	1,500215176	1,500213280	1,500212351	1,500211792	8,89449E-06
8	1,500214381	1,500214167	1,500214063	1,500214000	1,00282E-06
9	1,500214291	1,500214267	1,500214256	1,500214248	1,13063E-07
10	1,500214281	1,500214279	1,500214277	1,500214276	1,27470E-08
11	1,500214280	1,500214280	1,500214280	1,500214280	1,43700E-09
12	1,500214280	1,500214280	1,500214280	1,500214280	1,62000E-10
13	1,500214280	1,500214280	1,500214280	1,500214280	1,80000E-11

Tabela 11 – Autovalores para $\lambda2$

Iteração	Lambda 2				Erro
1	0,181954614	0,306277649	0,169640248	0,128409551	0,196570515
2	0,169626236	0,172347157	0,168837445	0,167210303	0,02706523
3	0,169160494	0,169266669	0,169128911	0,169063527	0,000350385
4	0,169141993	0,169146210	0,169140727	0,169138122	1,31370E-05
5	0,169141295	0,169141380	0,169141180	0,169141081	5,29026E-07
6	0,169141610	0,169140880	0,169141039	0,169141075	8,29270E-08

Tabela 12 – Autovalores para $\lambda 3$

Iteração	Lambda 2				Erro
1	0,006895877	0,007161427	-0,102926745	0,006085493	0,020695987

Tabela 13 – Maior Autovalor em valor absoluto

$\lambda 1$	$\lambda 2$	$\lambda 3$	$\lambda 4$
1,5002143	0,1691406	0,0067277	0,0001079

Os valores calculados pelo programa para os autovetores estão dispostos na tabela 14, abaixo:

Tabela 14 – Autovetores

Autovetor 1	Autovetor 2	Autovetor 3	Autovetor 4
0,7926	-0,5821	0,1850	0,0292
0,4519	0,3705	-0,7455	0,3287
0,3224	0,5096	0,0951	0,7914
0,2522	0,5140	0,6332	0,5145

O tempo médio para o programa do método das potências é de 6.54 centésimos de segundo, fato que é aceitável, uma vez que, a matriz do problema solucionado é de ordem pequena.
Foi utilizada além do algoritmo implementado baseado no método das potências para encontrar os autovalores e autovetores da da matriz A_4x4, a rotina $eig()$ fornecida pelo próprio $MatLab$. Nas tabelas 15 e 16 são exibidos os valores obtidos.

Tabela 15 – Autovetores calculados pela rotina do $MatLab$

Autovetor 1	Autovetor 2	Autovetor 3	Autovetor 4
0,0292	0,1792	-0,5821	0,7926
-0,3287	-0,7419	0,3705	0,4519
0,7914	0,1002	0,5096	0,3224
-0,5146	0,6383	0,5140	0,2522

Tabela 16 – Maior Autovalor em valor absoluto calculado pela rotina do $MatLab$

$\lambda 1$	$\lambda 2$	$\lambda 3$	$\lambda 4$
0,0001	0,0067	0,1691	1,5002

A rotina do $MatLab$ gasta em média 9 milésimos de segundo para calcular os valores e comparando- se a rotina com os programas desenvolvidos, observa- se que, há uma boa aproximação entre os valores obtidos pelos programas e que o tempo de processamento de ambos é significativamente baixo para a matriz utilizada.

CONSIDERAÇÕES FINAIS

Enfim, após aplicar- se todos os métodos observa- se que estes mostraram eficiência na resolução do sistema proposto, uma vez que, todos atingiram a convergência, apresentaram baixo valor para o erro de aproximação e realizaram um número pequeno de iteração. Além disso, observa- se também que a tarefa de se obter os autovalores e os autovalores apresenta- se como trabalhosa mesmo realizando- se a aplicação de métodos que evitam a expansão do determinante em polinômios.

REFERÊNCIAS

[1] C. S. Machado, P. C. Souza, R. J. Queiroz, Trabalho 1: Resolução de sistemas lineares. Disponível em<https://github.com/caiqueCederj/trabalhos_mcct.git>. Nenhuma citação no texto.

[2] B. P. Demidovich, I. A. Maron, Computational Mathematics. 2nd Edition, Mir Publishers, 1976. Nenhuma citação no texto.

[3] ALVARES, G. B. Aulas 6 e 7. Disponível em: <http://www.professores.uff.br/gbenitez/atividades-de-ensino/ pos-graduacao-mestrado-multi-interdisciplinar-em-modelagem-computacional-em-ciencia-e-tecnologia/>. Acesso em: 22 de maio de 2019. Nenhuma citação no texto.

[4] ALVAREZ, G. B. Autovalores e Autovetores. Aula 8 e 9. Disponível em: <http://www.professores.uff.br/gbenitez/atividades-de-ensino/pos-graduacao-mestrado-multi-interdisciplinar-em-modelagem-computacional-em-ciencia-e-tecnologia/>. Acesso em: 2 de junho de 2019. Nenhuma citação no texto.

Apêndices

APÊNDICE A – LISTA DE ALGORITMOS

A.1 MÉTODO DE NEWTON

```
close all;
clc;
clear all;

disp(['Problema No. 1:']);
disp(['Resolução de sistema não linear pelo Método de Newton']);
tic
disp(['Aproximação inicial:']);
x0=[0.2;0.2;0.2]  %Aproximação inicial
 disp(['Funçoes do sistema a resolver:']);
 syms  x y z
 sistema=[(x-1)^2+(y-1)^2+(z-1)^2-1;2*x^2+(y-1)^2-4*z;3*x^2-4*y+2*z^2] %Funçoes do
sistema a resolver
 disp(['Matriz Jacobiana do sistema a resolver:']);
 jaco=jacobian(sistema)

 epsilon=1.0e-4;
 maxiter = 30;
 iteracao = 1;
 S=inline(sistema);
 J=inline(jaco);
 erro=norm(S(x0(1),x0(2),x0(3)),2);
  fx0=S(x0(1),x0(2),x0(3));
 fprintf(' Iteração                              raiz x
f(x)                              erro\n');
 fprintf('x                 y              z                 f(x)           f(y)
f(z)             \n');

 fprintf(' %2d (%14.9f,%14.9f,%14.9f)(%14.9f,%14.9f,%14.9f) %7.9f\n', 0,x0(1),x0(2),x0
(3),fx0(1),fx0(2),fx0(3),erro);
 while erro >= epsilon
         fx0=S(x0(1),x0(2),x0(3));
         Wx0=J(x0(1),x0(2),x0(3));
         xk=x0-inv(Wx0)*fx0;
         fxk=S(xk(1),xk(2),xk(3));
         erro=norm((fxk),2);
         fprintf(' %2d (%14.9f,%14.9f,%14.9f)(%14.9f,%14.9f,%14.9f) %7.9f\n',
iteracao,x0(1),x0(2),x0(3),fx0(1),fx0(2),fx0(3),erro);
    if iteracao > maxiter
        fprintf(' Numero maximo de iterações excedido \n');
    return;
    end

    x0=xk;
    iteracao = iteracao+1;
 end
toc
iteracao
```

A.2 MÉTODO DE JACOBI OU ITERAÇÃO

```
close all;
clc;
clear all;

tic
 palpite=[0.2;0.2;0.2]; %Aproximação inicial
 syms  x y z
 sistema=[(x-1)^2+(y-1)^2+(z-1)^2-1;2*x^2+(y-1)^2-4*z;3*x^2+2*z^2-4*y]; %Funçoes do↙
sistema a resolver
 jaco=jacobian(sistema);
 epsilon=1.0e-04;
 maxiter = 300;
 iteracao = 1;
 S=inline(sistema);
 J=inline(jaco);
 erro=norm(S(palpite(1),palpite(2),palpite(3)),2);
 fx0=S(palpite(1),palpite(2),palpite(3));
 Wx0=J(palpite(1),palpite(2),palpite(3));
 fprintf(' Iter                              raiz x↙
f(x)                              erro\n');
 fprintf(' %2d     (%14.9f,%14.9f,%14.9f)     (%14.9f,%14.9f,%14.9f)     %12.8f\n', 0,↙
palpite(1),palpite(2),palpite(3),fx0(1),fx0(2),fx0(3),erro);
 while erro >= epsilon
    fx0=S(palpite(1),palpite(2),palpite(3));
    xk=palpite-inv(Wx0)*fx0;
    fxk=S(xk(1),xk(2),xk(3));
    erro=norm((fxk),2);
    fprintf(' %2d     (%14.9f,%14.9f,%14.9f)     (%14.9f,%14.9f,%14.9f)     %12.8f\n',↙
iteracao,palpite(1),palpite(2),palpite(3),fx0(1),fx0(2),fx0(3),erro);
    if iteracao > maxiter
        fprintf(' Numero maximo de iterações excedido \n');
    return;
    end
    palpite=xk;
    iteracao = iteracao+1;
 end
toc
iteracao
```

A.3 MÉTODO DO GRADIENTE

```
close all;
clc;
clear all;

tic
x0=[0.2;0.2;0.2]; %Aproximação inicial
 syms  x y z
 sistema=[(x-1)^2+(y-1)^2+(z-1)^2-1;2*x^2+(y-1)^2-4*z;3*x^2+2*z^2-4*y]; %Funçoes do sistema a resolver
 jaco=jacobian(sistema);
 epsilon=1.0e-4;
 maxiter = 200;
 iteracao = 1;
 S=inline(sistema);
 J=inline(jaco);
 erro=norm(S(x0(1),x0(2),x0(3)),2);
 fx0=S(x0(1),x0(2),x0(3));
 fprintf(' Iter          raiz x                f(x)              erro\n');
 fprintf(' %2d (%14.9f,%14.9f,%14.9f)(%14.9f,%14.9f,%14.9f) %12.8f\n', 0,x0(1),x0(2),x0(3),fx0(1),fx0(2),fx0(3),erro);
 while erro >= epsilon
    fx0=S(x0(1),x0(2),x0(3));
    jaco=J(x0(1),x0(2),x0(3));
    Denx = jaco'*fx0;
    xk=x0-[(Denx)'*Denx*Denx]/[(jaco*Denx)'*(jaco*Denx)];
    fxk=S(xk(1),xk(2),xk(3));
    erro=norm((fxk),2);
    fprintf(' %2d (%14.9f,%14.9f,%14.9f)(%14.9f,%14.9f,%14.9f) %12.8f\n', iteracao,x0(1),x0(2),x0(3),fx0(1),fx0(2),fx0(3),erro);
    if iteracao > maxiter
        fprintf(' Numero maximo de iterações excedido \n');
    return;
    end
    x0=xk;
    iteracao = iteracao+1;
end
toc
```

A.4 AUTOVALORES E AUTOVETORES

```
close all;
clc;
clear all;
A = [1 1/2 1/3 1/4; 1/2 1/3 1/4 1/5; 1/3 1/4 1/5 1/6; 1/4 1/5 1/6 1/7];
[m,n] = size(A);
tol = 1.0E-4;
tic
% CALCULO DO PRIMEIRO AUTOVALOR LAMBDA1 E DO PRIMEIRO AUTOVETOR V1
Y0 = ones(n,1);
lambda0 = 0;
erro = inf;
iter=1;
fprintf(' Iter                                                lambda1↙
erro\n');
while erro >= tol

    Y = A*Y0;
    lambda1 = Y./Y0;
    erro = abs(mean(lambda1)-mean(lambda0));
    fprintf(' %2d (%18.12f,%18.12f,%18.12f,%18.12f,) %16.12f\n', iter,lambda1(1),↙
lambda1(2),lambda1(3),lambda1(4),erro);
    Y0 = Y;
    lambda0 = lambda1;
    iter = iter+1;
end
LAMBDA1 = mean(lambda1);
L1 = norm(Y,2);
V1 = Y./L1;
```

```
% CALCULO DO SEGUNDO AUTOVALOR LAMBDA2 E DO SEGUNDO AUTOVETOR V2
Y0 = ones(n,1);
lambda0 = 0;
erro0 = inf;
erro1 = inf;
iter1 = 1;
fprintf(' Iter                                                lambda2↙
erro\n');
while erro1 >= tol

    Y1 = A*Y0;
    Y2 = A*Y1;
    lambda2 = (Y2 - LAMBDA1*Y1)./(Y1 - LAMBDA1*Y0);
    Z2 = Y2 - LAMBDA1*Y1;
    erro1 = abs(mean(lambda2)-mean(lambda0));

    if erro0 < erro1
        break
    end
    fprintf(' %2d (%18.12f,%18.12f,%18.12f,%18.12f,) %16.12f\n', iter1,lambda2(1),↙
lambda2(2),lambda2(3),lambda2(4),erro1);
    Y0 = Y1;
    erro0 = erro1;
    lambda0 = lambda2;
    iter1 = iter1+1;
end
```

```
LAMBDA2 = mean(lambda2);
L2 = norm(Z2,2);
V2 = -Z2./L2;
%fprintf(' Autovetor 2: \n',V2 )
%V2

% CALCULO DO TERCEIRO AUTOVALOR LAMBDA3 E DO TERCEIRO AUTOVETOR V3
Y0 = [0.01;0.01;0.01;0.01];
lambda0 = 0;
erro0 = inf;
erro2 = inf;
iter2 = 1;
fprintf(' Iter                                          lambda3↙
erro\n');
while erro2 >= tol

    Y1 = A*Y0;
    Y2 = A*Y1;
    Y3 = A*Y2;
    lambda3 = ((Y3 - LAMBDA1*Y2) - LAMBDA2*(Y2 - LAMBDA1*Y1))./((Y2 - LAMBDA1*Y1) -↙
LAMBDA2*(Y1 - LAMBDA1*Y0));
    Z3 = (Y3 - LAMBDA1*Y2) - LAMBDA2*(Y2 - LAMBDA1*Y1);
    erro2 = abs(mean(lambda3)-mean(lambda0));

    if erro0 < erro2
        break
    end
    fprintf(' %2d     (%10.12f,%10.12f,%10.12f,%10.12f,)     %16.12f\n', iter2,lambda3↙
(1),lambda3(2),lambda3(3),lambda3(4),erro2);
    Y0 = Y1;
    erro0 = erro2;
    lambda0 = lambda3;
    iter2 = iter2+1;
end
LAMBDA3 = mean(lambda3);

L3 = norm(Z3,2);
V3 = Z3./L3;

%fprintf(' Autovetor 3: \n',V3 )
%V3

% CALCULO DO QUARTO AUTOVALOR LAMBDA4 E DO QUARTO AUTOVETOR V4

LAMBDA4 = trace(A)-LAMBDA1-LAMBDA2-LAMBDA3;

Z = [V1';V2';V3'];

X1 = [Z(:,2),Z(:,3),Z(:,4)];
X2 = [Z(:,1),Z(:,3),Z(:,4)];
X3 = [Z(:,1),Z(:,2),Z(:,4)];
X4 = [Z(:,1),Z(:,2),Z(:,3)];

Z4 = [det(X1),det(X2),det(X3),det(X4)];
```

```
L4 = norm(Z4,2);
V4 = Z4./L4;

fprintf('       LAMBDA1         LAMBDA2         LAMBDA3         LAMBDA4\n');
fprintf(' %14.7f,%14.7f,%14.7f,%14.7f\n', LAMBDA1,LAMBDA2,LAMBDA3,LAMBDA4);

fprintf('     AUTOVETOR1\n');...
fprintf(' %10.4f,%10.4f,%10.4f,%10.4f\n',V1');

fprintf('     AUTOVETOR2\n');...
fprintf(' %10.4f,%10.4f,%10.4f,%10.4f\n',V2');

fprintf('     AUTOVETOR3\n');
fprintf(' %10.4f,%10.4f,%10.4f,%10.4f\n',V3');

fprintf('     AUTOVETOR4\n');
fprintf(' %10.4f,%10.4f,%10.4f,%10.4f\n',V4');
toc
```