UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO - INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E COMPUTAÇÃO CIENTÍFICA
MS211A - CÁLCULO NUMÉRICO
PROF. JOÃO FREDERICO DA COSTA AZEVEDO MEYER

PROVA 1

BRYAN WOLFF **RA: 214095**

CAMPINAS
ABRIL DE 2021

Observação: *Todos os gráficos gerados e as questões discutidas foram feitos juntamente com a aluna Wesna Simone Bulla de Araujo RA: 225843. Além disso, os gráficos e os códigos em python estão disponíveis também no Google Colab.*

Questão 1 - *Considere a função dada por* $f(x) = sin[(x^2 - 0.23)^2] - e^{x^2} + 3x$. *com* $x \in [-2,2]$ *Aproxime a raiz ou as raízes justificando sua escolha de método, a precisão de sua aproximação da raiz ou de suas aproximações das raízes.*

Utilizei nesta questão o Método de Newton-Raphson, pois é um dos métodos numéricos mais eficientes para a solução de um problema de determinação de raiz. Além disso, este método possui grande rapidez no processo de convergência, sendo necessário a obtenção de f'(x).

Neste caso, foi utilizado o software Symbolab para obter $f'(x)$:

$$f'(x) = cos((x - 0.23)^2) * 2 * (x - 0.23) - e^{x^2} 2x + 3$$

Além disso, para uma escolha cuidadosa da aproximação inicial que é, em geral, essencial para o bom desempenho do método de Newton, gerei o gráfico da função utilizando a biblioteca matplotlib:

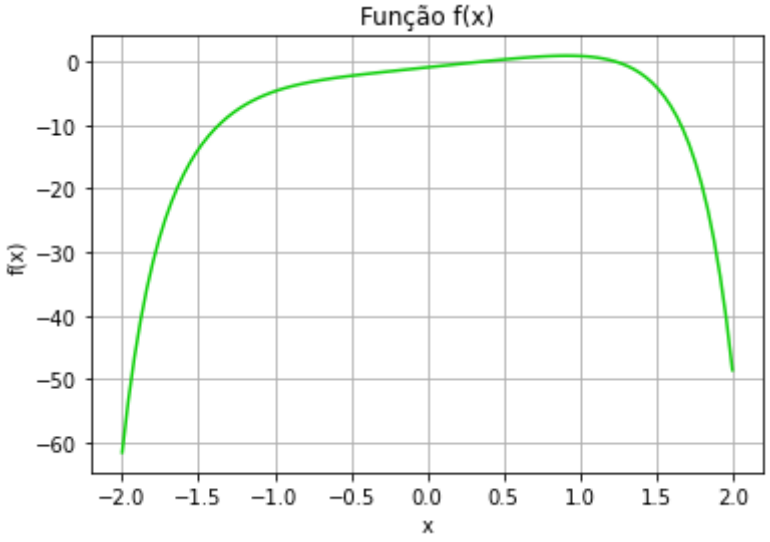

Figura 1 - Função $f(x)$com $x \in [-2,2]$

Vamos trabalhar um valor de erro na ordem de 10^{-3}, definido arbitrariamente, mas maior do que os erros tipicamente utilizados em aula (10^{-7}). A partir do gráfico podemos escolher x = 0 e x = 1 como os valores do chute inicial para utilizar o método proposto (disponível em anexos). Utilizando o primeiro chute de x = 1, e o código utilizado em anexo, temos o seguinte resultado:

Valor da raíz é: 1.22881004

Quantidade de iterações: 7

Erro absoluto: 2.0956242202219144e-05

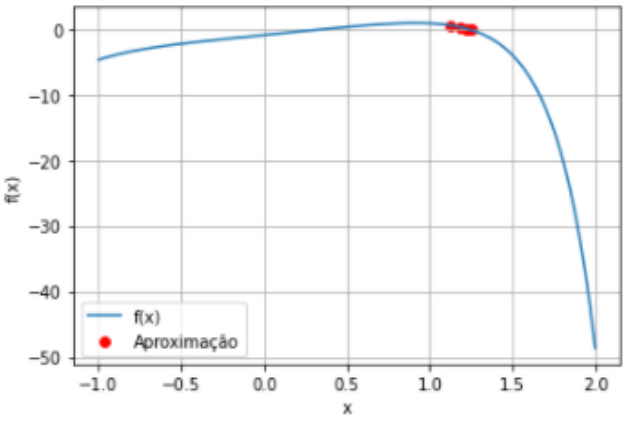

Figura 2 - Trecho da função $f(x)$ que mostra as aproximações solução convergindo para a raíz através do Método de Newton-Raphson a partir do chute inicial dado por $x = 1$.

Dessa forma, conseguimos obter uma boa aproximação do valor da raiz de x_1. Além disso, o valor de f(x) para a raíz x = 1.22881004 obtida neste método é -8.489858593918598e-09 e o erro absoluto da solução obtida indica uma boa aproximação da solução encontrada. Para obter a outra raiz, o segundo chute será de x = 0, e a partir do código em anexo, temos o seguinte resultado:

Valor da raíz é: 0.37704867

Quantidade de iterações: 3

Erro absoluto: 3.5373714823982993e-06

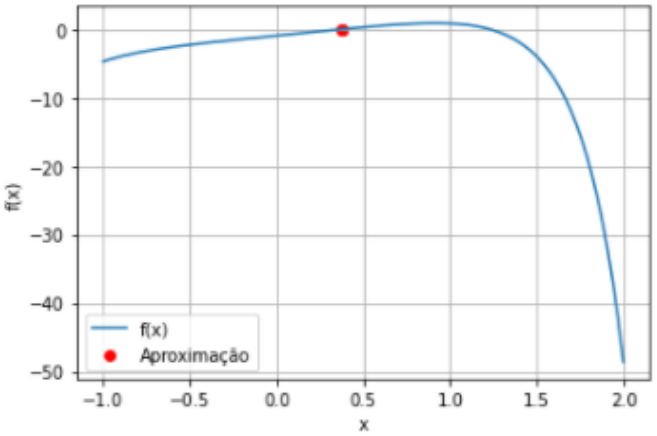

Figura 3 - Trecho da função $f(x)$ que mostra as aproximações solução convergindo para a raíz através do Método de Newton-Raphson a partir do chute inicial dado por $x = 0$.

Dessa forma, conseguimos obter uma ótima aproximação do valor da raiz de x_2. Além disso, o valor de f(x) para a raíz x = 0.37704867 obtida neste método é -6.0276228452949e-12 e o erro absoluto da solução obtida indica uma boa aproximação da solução encontrada. Nesta perspectiva, foi possível obter uma boa precisão para os valores das raízes (x_1 e x_2).

Questão 2 - *Considere a função* $f(x) = x + e^{-x^2}$ *na reta Real. Como identificar se esta função f(x) tem raiz ou raízes reais? Qual seria o método mais adequado para se identificar rapidamente uma boa aproximação dessa ou dessas raízes? Qual o controle de aproximação que você usaria?*

Para identificar se a função $f(x)$ possui raízes, vamos primeiro gerar o gráfico da função para analisar se passa pelo eixo $f(x) = 0$.

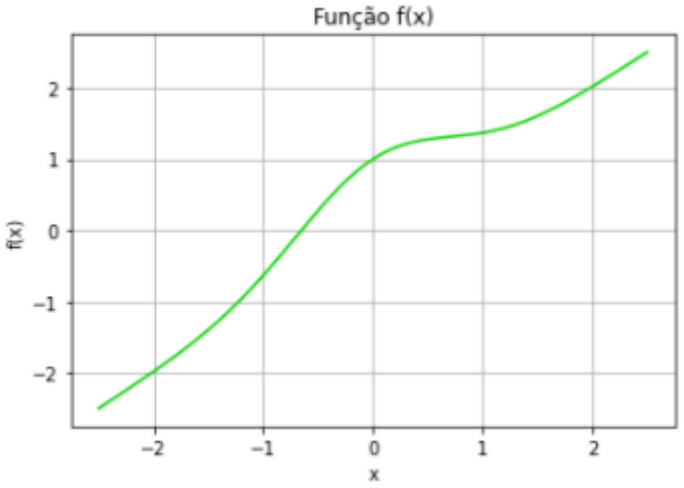

Figura 3 - Função $f(x)$com $x \in [-2,2]$

Dessa forma, é possível identificar a existência de uma raíz próximo de x = -0,5. Um dos métodos mais adequados e mais eficientes conhecido para a solução de um problema de determinação de raiz de maneira rápida é o Método de Newton-Raphson, devido a sua alta rapidez no seu processo de convergência, fazendo com que tenha um elevado desempenho.

Para o controle de aproximação, vamos utilizar um valor de erro na ordem de 10^{-3}, definido arbitrariamente, mas maior do que os erros tipicamente utilizados em aula (10^{-7}). Utilizamos essa tolerância pois estamos abordando este problema a partir de uma perspectiva didática, e por não ser utilizado em um problema real, não precisamos buscar um erro ínfimo.

Assim, para obter a raiz desejada, o chute será de x = -0,5 , e a partir do código em anexo, e da derivada $f'(x) = 1 - 2xe^{-x^2}$ temos o seguinte resultado:

Valor da raíz é: -0.65291864

Quantidade de iterações: 3

Erro absoluto: 7.198292603094458e-07

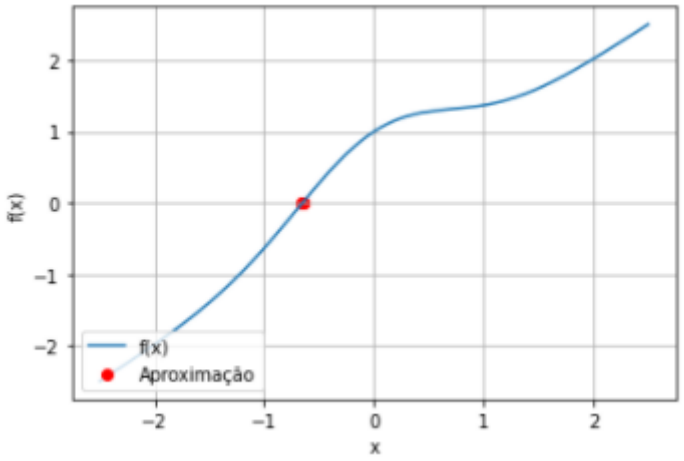

Figura 4 - Trecho da função $f(x)$ que mostra as aproximações solução convergindo para a raíz através do Método de Newton-Raphson a partir do chute inicial dado por $x = -0,5$.

Dessa forma, conseguimos obter uma ótima aproximação do valor da raiz de x_1. Além disso, o valor de f(x) para a raíz x = -0.65291864 obtida neste método é -4.984901380566953e-14 e o erro absoluto da solução obtida indica uma boa aproximação da solução encontrada. Nesta perspectiva, com apenas 3 iterações foi possível obter uma ótima aproximação para o valor da raiz a partir do Método de Newton-Raphson.

Questão 3 - *Dos métodos apresentados para aproximar a raiz de uma função, qual ou quais os que lhe parecem mais úteis e por quê? Há algum método que você acredita que nunca vai usar? Justifique cuidadosamente todas as suas opiniões.*

De todos os métodos apresentados para obter uma raiz aproximada de uma função, os que mais se destacam são o Método da Bissecção e o Método de Newton-Raphson.

O Método da Bissecção garante uma convergência, desde que a função seja contínua num intervalo [a,b], tal que f(a)f(b)<0. Isso garante que uma raiz seja encontrada, mesmo que o método não seja o mais eficiente possível. Se o objetivo for a redução do intervalo que contém a raiz este é o método mais indicado.

Já o Método de Newton-Raphson, é um dos métodos numéricos mais eficientes e conhecidos para a solução de um problema de determinação de raiz, e possui grandes vantagens devido a rapidez no processo de convergência e o seu desempenho elevado. Se a escolha parte de um valor inicial para a raiz e $f'(x)$ for de fácil obtenção, este é o método mais recomendado. Porém, nem sempre é fácil de determinar $f'(x)$, sendo muito difícil ou até mesmo impossível em alguns casos.

Acredito que um dos métodos que nunca vou usar é o Método do Ponto Fixo, pois mesmo que tenha um desempenho regular e previsível, tem a necessidade de obtenção de uma função de iteração g(x) que pode ser muito difícil de obter, como também é difícil de implementar.

Questão 4 - *Considere a matriz e o vetor dados pelo programa abaixo e resolva um sistema do tipo m*x = b. Justifique cuidadosamente sua escolha de método e critique seu resultado.*

```
%
n=251;
%
% Matriz
m=sparse(n);
for i=1:n
m(i,i) = 2.35;
endfor
for i=1:n-1
m(i,i+1) = -0.78;
m(i+1,i) = -0.41;
endfor
for i=1:n-25
m(i,i+25) = -0.51;
m(i+25,i) = -0.28;
endfor
%
% Termo independente
for i=1:2:n
b(i) = 1.5;
endfor
for i=2:2:n-1
b(i)=0.75;
endfor
b=b'
```

A partir desse código, obtemos as matrizes m e b do sistema linear dado por *m*x = b.* Dessa forma, utilizando o Método de Gauss-Seidel disponibilizado aqui, e também disponível em anexo, foi possível obter os seguintes valores para a solução aproximada do problema com apenas 26 iterações:

x = [1.8655, 2.0171, 2.2752, 2.1036, 2.2941, 2.1079, 2.2951, 2.1081, 2.2952, 2.1081, 2.2952, 2.1081, 2.2952, 2.1082, 2.2954, 2.1086, 2.2963, 2.1107, 2.3010, 2.1215, 2.3264, 2.1825, 2.4777, 2.5715, 2.8461, 2.7054, 2.8863, 2.7166, 2.8893, 2.7173, 2.8895, 2.7174, 2.8895, 2.7174, 2.8895, 2.7174, 2.8895, 2.7174, 2.8896, 2.7175, 2.8898, 2.7182, 2.8914, 2.7216, 2.8986, 2.7365, 2.9280, 2.7901, 3.0114, 2.8629, 3.0458, 2.8757, 3.0500, 2.8770, 3.0504, 2.8771, 3.0504, 2.8771, 3.0504, 2.8771, 3.0504, 2.8771, 3.0504, 2.8770, 3.0503, 2.8769, 3.0502, 2.8770, 3.0509, 2.8789, 3.0551, 2.8869, 3.0678, 2.9031, 3.0813, 2.9102, 3.0841, 2.9111, 3.0844, 2.9112, 3.0844, 2.9112, 3.0844, 2.9112, 3.0844, 2.9112, 3.0844, 2.9111, 3.0841, 2.9107, 3.0835, 2.9098, 3.0822, 2.9080, 3.0800, 2.9055, 3.0776, 2.9034, 3.0759, 2.9019, 3.0744, 2.9008, 3.0738, 2.9005, 3.0736, 2.9004, 3.0736, 2.9004, 3.0736, 2.9004, 3.0736, 2.9003, 3.0734, 2.8999, 3.0727, 2.8988, 3.0709, 2.8960, 3.0668, 2.8900, 3.0586, 2.8796, 3.0464, 2.8670, 3.0353, 2.8592, 3.0309, 2.8571, 3.0301, 2.8568, 3.0300, 2.8567, 3.0299, 2.8567, 3.0299, 2.8566, 3.0298, 2.8564, 3.0293, 2.8555, 3.0277, 2.8529, 3.0234, 2.8460, 3.0128, 2.8304, 2.9913, 2.8032, 2.9610, 2.7751, 2.9411, 2.7649, 2.9368, 2.7632, 2.9362, 2.7630, 2.9362, 2.7630, 2.9361, 2.7630, 2.9361, 2.7628, 2.9358, 2.7623, 2.9347, 2.7604, 2.9314, 2.7546, 2.9216, 2.7384, 2.8956, 2.6991, 2.8407, 2.6311, 2.7719, 2.5822, 2.7499, 2.5741, 2.7472, 2.5733, 2.7470, 2.5732, 2.7469, 2.5732, 2.7469, 2.5731, 2.7468, 2.5728, 2.7462, 2.5718, 2.7443, 2.5683, 2.7378, 2.5563, 2.7162, 2.5183, 2.6518, 2.4157, 2.5048, 2.2436, 2.3792, 2.1994, 2.3661, 2.1958, 2.3651, 2.1955, 2.3650, 2.1955, 2.3650, 2.1954, 2.3649, 2.1953, 2.3647, 2.1950, 2.3640, 2.1934, 2.3610, 2.1876, 2.3495, 2.1648, 2.3045, 2.0771, 2.1386, 1.7825, 1.6935, 1.4368, 1.6229, 1.4218, 1.6196, 1.4210, 1.6195, 1.4210, 1.6194, 1.4210, 1.6194, 1.4210, 1.6194, 1.4209, 1.6192, 1.4205, 1.6184, 1.4189, 1.6151, 1.4119, 1.5999, 1.3778, 1.5214, 1.1907, 1.0584]

Vale destacar que, se plotar o valor da variável $Erro\_value$ do código citado em anexo, vamos obter uma lista de erros absolutos associado a cada valor da solução obtida. O erro é da ordem de grandeza de 10^{-4}a 10^{-6} e indica uma boa precisão da solução encontrada.

Escolhi o método iterativo de Gauss-Seidel para a resolução do problema, pois mesmo que necessitem de um número infinito de operações aritméticas para obter a solução, ele é vantajoso porque pode produzir boas aproximações com relativamente poucas iterações. Além disso, a matriz M nunca é alterada durante o processo iterativo, ao contrário do que acontece com os outros métodos não iterativos. Vale ressaltar que, diferentemente do método de Jacobi, o método de Gauss-Seidel utiliza valores anteriores para o cálculo do valor posterior, afetando positivamente na precisão dos resultados, além de convergir mais rápido que o método de Jacobi. Além disso, os cálculos são simples no método de Gauss-Seidel, e por isso a tarefa de programação é simples e o requisito de memória é menor, sendo útil para sistemas pequenos.

Questão 5 - *Se, em seu trabalho profissional, você tiver que orientar a compra de um software para resolver sistemas lineares, quais seriam suas três primeiras escolhas? Justifique…*

Se eu fosse orientar a compra de um software para resolver sistemas lineares, dentre os métodos estudados eu escolheria o Método iterativo de Newton-Raphson Modificado, Método iterativo de Gauss-Seidel e o Método Direto de Eliminação de Gauss com pivoteamento parcial.

O método iterativo de Newton é considerado por muitos autores o melhor método para encontrar sucessivas melhores aproximações da solução, além do método funcionar para grande parte dos casos e ainda também para sistemas lineares e não lineares. Além de eficiente, apresenta um número reduzido de iterações e com uma boa precisão para encontrar a solução aproximada do sistema. Além disso, o Método de Newton Modificado tem a vantagem de calcular uma única vez a matriz Jacobiana e pode convergir para uma solução quando o ponto inicial está longe desta. E o número de iterações são independentes do tamanho do sistema.

O método de Gauss-Seidel, é um ótimo método para a resolução de sistemas lineares, pois os cálculos por iterações são simples e, portanto, a sua implementação é mais fácil. Dessa forma, o tempo gasto por iteração é menor do que o Método de Newton, e por isso o

armazenamento necessário na memória do computador é relativamente menor, podendo ser aplicados em sistemas menores.

Já o Método de Eliminação de Gauss é o algoritmo de solução mais fundamental, que consiste em aplicar operações elementares sucessivas em um sistema linear (escalonamento), transformando num sistema mais fácil de resolvê-lo. A técnica de pivô parcial é usada para evitar erros de arredondamento. Além disso, pode ser usado onde o número de equações e o número de incógnitas não são iguais.

Questão 6 - *Considere o sistema não linear dado abaixo. Identifique uma terna (x,y,z) para a qual f, g e h se anulam simultaneamente. Avalie seu resultado e comente a precisão.*

$$f(x,y,z) = 3x^2y^2 - z + 2$$
$$g(x,y,z) = x - 2y^2 - 3z^2 + 12$$
$$h(x,y,z) = x + y - 0.5$$

Para a resolução de um sistema não linear, utilizei o método iterativo disponibilizado pelo docente na página do curso, cujo nome é dado por "$Método\ de\ Newton\ Não\ Linear$". O código abordado com as devidas funções $f(x,y,z)$, $g(x,y,z)$ e $h(x,y,z)$ do problema está contido no anexo. Utilizei este código devido a baixa quantidade de iterações realizadas até atingir o resultado desejado.

Para utilizar o método proposto, é necessário calcular a matriz jacobiana J_{3X3}, cuja matriz está destacada no código em anexo. Devido a dificuldade de gerar o gráfico da função $f(x,y,z)$, $g(x,y,z)$ e $h(x,y,z)$ de forma a estimar um chute inicial para a resolução do sistema, utilizei como chute $x = 1$, $y = -1$, $z = 2$, isto é, valores inicialmente baixos devido ao fato de que, se passar coordenadas com valores baixos, a função já estará próxima de zero.

Dessa forma, dado o chute inicial, a matriz jacobiana e o sistema linear, já é possível executar o código de forma a obter o resultado. Ao executar o código dado em anexo, obtemos os seguintes valores:

$$x = 0.6889$$
$$y = -0.1889$$
$$z = 2.0508$$

Iterações: 9

Para avaliar o resultado obtido, calculamos o valor das funções no resultado encontrado:

$$f(0.6889, -0.1889, 2.0508) = 3.9570e-06$$

$$g(0.6889, -0.1889, 2.0508) = 1.9166e-04$$

h(0.6889,-0.1889,2.0508) = -5.5511e-17

Nessa perspectiva, dado o resultado obtido e o valor das funções avaliada nas coordenadas da solução, obtemos um resultado satisfatório na resolução do problema, ou seja, foi possível identificar uma terna (x, y, z) no qual as funções $f(x, y, z)$, $g(x, y, z)$ e $h(x, y, z)$ se anulam simultaneamente. Além disso, o erro absoluto do problema se encontra na ordem de grandeza de 10^{-11}, isto indica uma boa aproximação da solução encontrada

Questão 7 - *Comente o que lhe pareceu mais útil no que foi visto até aqui de MS211.*

Em geral, tudo o que aprendemos até agora pode ser utilizado posteriormente em problemas profissionais. Os enfatizados Métodos Numéricos me surpreenderam pelo fato de que é possível formular e resolver problemas matemáticos complexos de diversas áreas da ciências exatas usando operações aritméticas menos complexas a partir de aplicações de algoritmos simples. De maneira geral, esses métodos podem ser utilizados para diversos problemas da vida cotidiana, sendo um dos assuntos que pretendo carregar comigo pela vida toda.

Questão 8 - *Comente algum tópico que não foi abordado nas questões anteriores e justifique o(s) motivo(s) de sua escolha.*

Existem dois tópicos que me recordo que não foram abordados nas questões anteriores, que são os erros nas representações de números reais e aritmética de ponto flutuante. Acredito que, mesmo que sejam tópicos importantes no processo de aprendizagem de Cálculo Numérico, o foco dado nos Métodos Numéricos é compreensível, visto que é um assunto que podemos utilizar na resolução de problemas matemáticos no mundo profissional e acadêmico.

ANEXOS

- ## Método de Newton-Raphson

```
def newtonRaphson(x0 = 1, e = 0.001, func = f, dev = dev):
  vetor = [] #vetor das aproximações
  Dr = 0 #Erro absoluto
  qtd_iter = 0
  h = func(x0)/dev(x0)
  while abs(h) >= e:
    h = func(x0)/dev(x0)
    x = x0 - h
    Dr = abs(x - x0)
    x0 = x
    qtd_iter += 1
    vetor.append(x)
  print("Valor da raíz é: {:.8f}".format(x))
  print("Quantidade de iterações: {}".format(qtd_iter))
  print("Erro absoluto é: ", Dr)
  print("O valor de f(x) para a raíz x = {:.8f} obtida neste método é
{}".format(vetor[-1],func(vetor[-1])))
  return vetor
```

- ## Método de Gauss-Seidel

```
%% Solution of x in Ax=b using Gauss Seidel Method
%Disponível em:
https://www.mathworks.com/matlabcentral/fileexchange/32051-gauss-seidel-method
n=251;
%
% Matriz
m=sparse(n);
for i=1:n
m(i,i) = 2.35;
endfor
for i=1:n-1
m(i,i+1) = -0.78;
m(i+1,i) = -0.41;
endfor
for i=1:n-25
m(i,i+25) = -0.51;
m(i+25,i) = -0.28;
endfor
%
% Termo independente
for i=1:2:n
b(i) = 1.5;
endfor
for i=2:2:n-1
b(i)=0.75;
endfor
b=b';
%Utilizaremos agora o método Gauss-Seidel
A = m;
```

```
C = b;% constants vector
n = length(C);
X = zeros(n,1);
Error_eval = ones(n,1);
%% Check if the matrix A is diagonally dominant
for i = 1:n
    j = 1:n;
    j(i) = [];
    B = abs(A(i,j));
    Check(i) = abs(A(i,i)) - sum(B); % Is the diagonal value greater than the
remaining row values combined?
    if Check(i) < 0
        fprintf('The matrix is not strictly diagonally dominant at row %2i\n\n',i)
    end
end
%% Start the Iterative method
iteration = 0;
while max(Error_eval) > 0.001
    iteration = iteration + 1;
    Z = X;  % save current values to calculate error later
    for i = 1:n
        j = 1:n; % define an array of the coefficients' elements
        j(i) = [];  % eliminate the unknow's coefficient from the remaining
coefficients
        Xtemp = X;  % copy the unknows to a new variable
        Xtemp(i) = [];  % eliminate the unknown under question from the set of values
        X(i) = (C(i) - sum(A(i,j) * Xtemp)) / A(i,i);
    end
    Xsolution(:,iteration) = X;
    Error_eval = sqrt((X - Z).^2);
end
%% Display Results
GaussSeidelTable = [1:iteration;Xsolution]'
MaTrIx = [A X C]
```

- # Método de Newton Não Linear

```
% Método de Newton não-Linear para
% o sistema dado por
% 1ª equação f(x,y,z) = 3*(x^2)*(y^2) -z + 2
% 2ª equação g(x,y,z) = x - 2*(y^2) - 3*(z^2) + 12
% 3ª equação h(x,yz) = x + y - 0.5
clc
clear all
%
% Preparo do cálculo da função vetorial F = (f(x,y,z),g(x,y,z),h(x,y,z))
f = @ (x,y,z) 3*(x^2)*(y^2) -z + 2
g = @ (x,y,z) x - 2*(y^2) - 3*(z^2) + 12
h = @ (x,y,z) x + y - 0.5
%
% Preparo para construção do Jacobiano
a11 = @ (x,y,z) 6*x*(y^2)
a12 = @ (x,y,z) 6*(x^2)*y
a13 = @ (x,y,z) -1
a21 = @ (x,y,z) 1
a22 = @ (x,y,z) -4*y
```

```
a23 = @ (x,y,z) -6*z
a31 = @ (x,y,z) 1
a32 = @ (x,y,z) 1
a33 = @ (x,y,z) 0

% Aproximação inicial do vetor x0
x0 = [1 -1 2]';
% N° máximo de iterações K, contador icont
% e tolerância tol
K=25;
tol = 1e-10; err=1;
icont = 1;
%
% Procedimento repetitivo/iterativo

%
while icont<K && err>tol
J=[a11(x0(1),x0(2),x0(3)) a12(x0(1),x0(2),x0(3)) a13(x0(1),x0(2),x0(3));
a21(x0(1),x0(2),x0(3)) a22(x0(1),x0(2),x0(3)) a23(x0(1),x0(2),x0(3));
a31(x0(1),x0(2),x0(3)) a32(x0(1),x0(2),x0(3)) a33(x0(1),x0(2),x0(3))];
F=[f(x0(1),x0(2),x0(3)); g(x0(1),x0(2),x0(3)); h(x0(1),x0(2),x0(3))];
delt = J\(-F);
err=norm(delt);
x0=x0.+delt;
icont=icont+1;
endwhile
display('resultado')
x0
display('n° de iterações')
icont
display('norma de F')
norm(F)
```