1 Trabalho de Computação Científica

O sistema LORAN (LOng RAnge Navigation) calcula a posição de uma embarcação no mar usando sinais a partir de transmissores fixados em determinados lugares. A partir das diferenças de tempo dos sinais recebidos, o barco obtém diferenças de distâncias aos transmissores. Isso leva a duas equações, cada uma representando hipérboles definidas pelas diferenças de distância de dois pontos (focos). O sistema de equações gerado é dado por:

$$\begin{cases} \dfrac{4(x_B - \frac{d_1}{2})^2}{c^2(t_2 - t_1)^2} - \dfrac{4y_B^2}{d_1^2 - c^2(t_2 - t_1)^2} = 1 \\ \\ \dfrac{4(y_B - \frac{d_2}{2})^2}{c^2(t_3 - t_2)^2} - \dfrac{4(x_B - d_1)^2}{d_2^2 - c^2(t_3 - t_2)^2} = 1 \end{cases} \tag{1}$$

onde *c é* a velocidade da luz e os outros parâmetros estão representados na Figura 1.

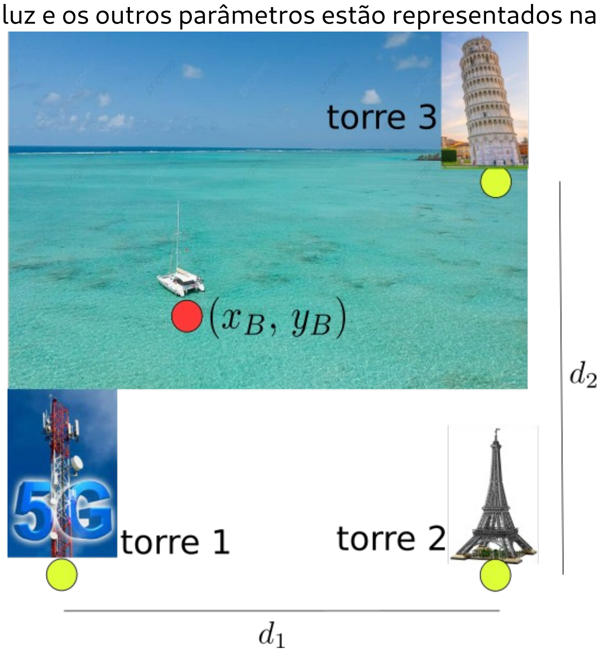

Figure 1: Representação das torres transmissoras e da posição do barco (x_B, y_B).

Observação: Poderíamos resolver o sistema (1) manualmente, mas para que um sistema de navegação funcione bem, ele deve fazer os cálculos de forma automática e numérica. Vale mencionar que o Sistema de Posicionamento Global (GPS) funciona de forma parecida e deve fazer cálculos semelhantes.

1. Resolva o sistema (1) pelo método de Newton. Se a torre 1 está a 3 km da torre 2, e a torre 3 está a 4 km da torre 2, descubra a posição do barco no mar (xB, yB) que recebe os sinais t1 = 0 s, t2 = 5,72 μs e t3 = 8,58 μs. (1 μs = 0,001 ms = 10^{-6} s)

Resolução do trabalho 1

O objetivo do programa é achar duas variáveis (**xB**, **yB**), em um Sistema não linear usando o método de newton.

O programa foi separado em dois arquivos: O programa principal (**main.m**) e e uma função para resolver o sistema não linear usando o método de newton(**fn_newton_method_non_lin_system.m**)

- O programa principal (**main.m**)

$$\begin{cases} \dfrac{4(x_B - \dfrac{d_1}{2})^2}{c^2(t_2-t_1)^2} - \dfrac{4y_B^2}{d_1^2 - c^2(t_2-t_1)^2} & = 1 \\ \\ \dfrac{4(y_B - \dfrac{d_2}{2})^2}{c^2(t_3-t_2)^2} - \dfrac{4(x_B-d_1)^2}{d_2^2 - c^2(t_3-t_2)^2} & = 1 \end{cases}$$

No main.m são passados os parâmetros para construir o sistema de equações

Sendo (xB_initial, yB_initial) o chute inicial, considerando a natureza do problema foi considerado (x inicial igual a 1500 m, y inicial igual a 200 m), pois dessa forma os pontos incias são direcionados ao centro, tendo uma maior precisão no algoritmo.

tol é a tolerância usada como ponto de parada, ao qual as funções se aproximam de zero, nesse caso foi usado 1e-8 = 10^{-8}.

d1 e d2 são as distancias das torres usadas na formula.

t1, t2 e t3 são os tempos usado na formula.

c é uma constante represando a velocidade da luz $\simeq$ $10^8 \, \mathrm{m/s}$

Eq, é um vetor de dois elementos, contendo todo elementos da formula. Eq{1} contem a primeira Equação e Eq{2} contem a segunda equação.

vimx main.m

```matlab
% main.m

function main()
    % Initial guess for (xB, yB)
    xB_initial = 1500; % meters
    yB_initial = 2000; % meters
    tol = 1e-8; % tolerance

    % m (Note: dnt confuse meters(m) with kilometers(1 km = 1e3 = 10³ m))
    d1 = 3e3; % meters
    d2 = 4e3; % meters

    % seconds
    t1 = 0;
    t2 = 5.72e-6;
    t3 = 8.58e-6;

    initial_guess = [xB_initial; yB_initial]; % vector of xB and yB

    Eq = equations(d1, d2, t1, t2, t3);

    % Call Newton's method (fn_newthon_method_non-lin-system.m)
    [xB, yB] = fn_newton_method_non_lin_system(Eq, initial_guess, tol);

    printf("xB = %f, yB = %f\n", xB, yB);
end


function Eq = equations(d1, d2, t1, t2, t3)
    c = 3e8; % speed of light (m/s)

    % Equation 1 and Equation 2 | v(1) = xB and v(2) = yB
    Eq{1} = @(v) (4*(v(1) - d1/2)^2)/(c^2*(t2-t1)^2) - (4*(v(2)^2))/(d1^2 - c^2*(t2-t1)^2) - 1;
    Eq{2} = @(v) (4*(v(2) - d2/2)^2)/(c^2*(t3-t2)^2) - (4*(v(1) - d1)^2)/(d2^2 - c^2*(t3-t2)^2) - 1;
end

main();

main.m [+]                                                  38,0-1        All
```

- Função do método de newton (**fn_newton_method_non_lin_system.m**)

No arquivo **fn_newton_method_non_lin_system.m** existem duas funções: **numerical_jacobian** e **fn_newton_method_non_lin_system**.

numerical_jacobian: É a função que dados um sistema de equações(Eq) e par (x, y)(pos) calcula a matriz jacobiana usando a diferenciação numérica e retornando uma matriz jacobiana (J) .

$f'(x) \approx \frac{f(x+h) - f(x)}{h}$ onde h é um número muito próximo de zero, (ex: h $=10^{-6}$)

fn_newton_method_non_lin_system: é a função na qual calcula o método de newton usando a função **numerical_jacobian**, resolve o sistema usando a matriz jacobina (J) com a Equações(Eq{1}, Eq{2}) usando os valores da posições (pos) e atribuindo isso ao delta, e em seguida atribuindo o valor da posição(pos) + (delta) a nova posição, e isso se repete enquanto o determinante de delta (err) for maior que a tolerância (tol). No final de cada repetição é mostrada o valor de cada variável junto a um contador que mostra as mudanças ocorrias em cada passo.

No final da função é retornado os valores de xB e yB.

Repositório onde está o código do exemplo 1 :

1. https://github.com/6Dren/newton_method_for_non_lin_system_matlab/tree/main/example_1

2 Trabalho de Computação Científica

Descreva no seu trabalho a tolerância utilizada, o critério de parada e tudo mais que julgar necessário. Lembre-se dos comentários feitos em sala de aula.
Resolva o sistema não-linear (2) pelo método de Newton.

$$\begin{cases} 3x_1 - cos(x_2x_3) & = \dfrac{1}{2} \\ x_1^2 - 81(x^2+0.1)^2 + sen(x_3) & = -1.06 \\ e^{-x_1x_2} + 20x_3 & = -\dfrac{10\pi - 3}{3} \end{cases}$$

Para x1,x2 e x3. Utilize como chute inicial a aproximação x(0)=(0.1,0.1,−0.1). Descreva no seu trabalho a tolerância utilizada, o critério de parada e tudo mais que julgar necessário. Lembre-se dos comentários feitos em sala de aula.

4 Trabalho de Computação Científica

Resolva o sistema não-linear (4) pelo método de Newton para x1,x2 e x3.

$$\begin{cases} x_1^2 + 2x_2^2 - x_2 - 2x_3 & = 0 \\ x_1^2 - 8x_2^2 + 10x_3 & = 0 \\ \dfrac{x_1^2}{7x_2x_3} - 1 & = 0 \end{cases}$$

Descreva no seu trabalho a tolerância utilizada, o critério de parada e tudo mais que julgar necessário. Lembre-se dos comentários feitos em sala de aula.

5 Trabalho de Computação Científica

Resolva o sistema não-linear (5) pelo método de Newton para x1,x2 e x3.

$$\begin{cases} x_1 + cos(x_1x_2x_3) - 1 & = 0 \\ (1-x_1)^{\frac{1}{4}} + x_2 + 0.05x_3^2 - 0.15x_3 - 1 & = 0 \\ -x_1^2 - 0.1x_2^2 + 0.01x_2 + x_3 - 1 & = 0 \end{cases}$$

Descreva no seu trabalho a tolerância utilizada, o critério de parada e tudo mais que julgar necessário. Lembre-se dos comentários feitos em sala de aula.

Resolução do trabalho 2, 4 e 5

Como ele ele muito parecido com o trabalho 1, e é o mesmo algoritmo do trabalho 2, 4 e 5, a resolução dele não sera muito diferente do anterior

o objetivo do programa é achar três variáveis (**xB**, **yB**, **zB**), em um Sistema não linear usando o método de newton.

O programa foi separado em dois arquivos: O programa principal (**main.m**) e e uma função para resolver o sistema não linear usando o método de newton(**fn_newton_method_n3**)

- O programa principal (**main.m**)

No main.m são passados os parâmetros para construir o sistema de equações

Sendo (xB_initial, yB_initial, zB_initial) o chute inicial, foi considerando os valores pedidos para cada trabalho, no caso do 2º são (0.1,0.1,−0.1), no 4º e 5º não são especificados, então zera considerado (1, 1, 1) para ambos.

tol é a tolerância usada como ponto de parada, ao qual as funções se aproximam de zero, nesse caso foi usado 1e-8 = 10^{-8}.

Eq, é um vetor de três elementos, contendo as equações usadas em cada questão. Eq{1} contem a primeira Equação, Eq{2} contem a segunda equação e Eq{3} contem a terceira equação.

- Função do método de newton (**fn_newton_method_n3 .m**)

No arquivo **fn_newton_method_n3.m** existem duas funções: **numerical_jacobian** e **fn_newton_method_n3**.

numerical_jacobian: É a função que dados um sistema de equações(Eq) e um trio (x, y, z) (pos) calcula a matriz jacobiana usando a diferenciação numérica e retornando uma matriz jacobiana (J) .

$f'(x) \approx \frac{f(x+h)-f(x)}{h}$ onde h é um número muito próximo de zero, (ex: h =10^{-6})

fn_newton_method_non_lin_system: é a função na qual calcula o método de newton usando a função **numerical_jacobian**, resolve o sistema usando a matriz jacobina (J) com a Equações(Eq{1}, Eq{2}, Eq{3})usando os valores da posições (pos) e atribuindo isso ao delta, e em seguida atribuindo o valor da posição(pos) + (delta) a nova posição, e isso se repete enquanto o determinante de delta (err) for maior que a tolerância (tol). No final de cada repetição é mostrada o valor de cada variável junto a um contador que mostra as mudanças ocorrias em cada passo.

No final da função é retornado os valores de xB, yB, zB.

Repositório onde está o código do exemplo 2, 4 e 5:

2. https://github.com/6Dren/newton_method_for_non_lin_system_matlab/tree/main/example_2

4. https://github.com/6Dren/newton_method_for_non_lin_system_matlab/tree/main/example_4

5. https://github.com/6Dren/newton_method_for_non_lin_system_matlab/tree/main/example_5

3 Trabalho de Computação Científica

O sistema LORAN (LOng RAnge Navigation) calcula a posição de uma embarcação no mar usando sinais a partir de transmissores fixados em determinados lugares. A partir das diferenças de tempo dos sinais recebidos, o barco obtém diferenças de distâncias aos transmissores. Isso leva a duas equações, cada uma representando hipérboles definidas pelas diferenças de distância de dois pontos (focos). Considere o sistema de equações não-lineares gerado para alguma situação:

$$\begin{cases} \dfrac{x^2}{186^2} - \dfrac{y^2}{300^2 - 186^2} & = 1 \\ \dfrac{(y-500)^2}{279^2} - \dfrac{(x-300)^2}{500^2 - 279^2} & = 1 \end{cases}$$

Resolva o sistema (3) pelo método de Newton.
Descreva no seu trabalho a tolerância utilizada, o critério de parada e tudo mais que julgar necessário. Lembre-se dos comentários feitos em sala de aula.

Resolução do trabalho 3

O objetivo do programa é achar duas variáveis (**xB**, **yB**), em um Sistema não linear usando o método de newton.

O programa foi separado em dois arquivos: O programa principal (**main.m**) e e uma função para resolver o sistema não linear usando o método de newton(**fn_newton_method_n2.m**).

A resolução do **trabalho 3** é muito similar ao do **trabalho 1**, com poucas diferenças

se desconsiderar as variáveis: distancia (d1, d2), tempo (t1, t2 e t3) e a velocidade da luz (c), que são omitidas no trabalho 3, a mesma resolução do trabalho 1 se aplica no trabalho 3

Repositório onde está o código do exemplo 3:

3. https://github.com/6Dren/newton_method_for_non_lin_system_matlab/tree/main/example_3