Universidade Estadual de Maringá
Centro de Tecnologia
Departamento de Informática
Bacharelado em Ciência da Computação
6900-01 - Matemática Computacional

Trabalho de Matemática Computacional

Alunos: Diogo Alves de Almeida RA: 95108

Professor: Airton Marco Polidório

Maringá
2017

1 Cálculo de raiz quadrada pelo método de Newton-Raphson

Nesta seção será apresentado a implementação do cálculo da raiz quadrada de números reais através do método de Newton-Raphson. Esse método funciona a partir de cálculos iterativos, que melhoram a solução ao fim de cada iteração.

No começo do algoritmo é definido quantas casa decimais será a precisão, veremos que quanto maior for as casas decimais, maior será o número de iterações para obtermos a solução desejada.

Abaixo o algoritmo:

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

long double ieee754 (float entrada);

int main (){
  long double xk, fxk, dfxk, xk1, erro, precisao, entrada;
  int i = 0;
  precisao = 0.000000000000001;
  printf("Entrada: ");
  scanf(" %Lf", &entrada);
  xk1 = ieee754(entrada);
  do {
    printf("Iteracao %i \n", i++);
    xk = xk1;
    printf("xk = %.20Lf\n", xk);
    fxk = (xk * xk) - entrada;
    printf("f(xk) = %.20Lf\n", fxk);
    dfxk = 2 * xk;
    printf("f'(Xk) = %.20Lf\n", dfxk);
    xk1 = xk - (fxk/dfxk);
    printf("xk1 = %.20Lf\n", xk1);
    erro = fabs(xk1 - xk);
    printf("Erro: %.25Lf\n\n\n", erro);
  } while(erro > precisao);

  printf("Precisao: %.15Lf\n", precisao);
  printf("xk = %.30Lf \n", xk);
}

long double ieee754 (float entrada){
    int expo;
    long double mantissa;

    expo = floor(log2(entrada));
    mantissa = entrada / pow(2,expo);
    expo = expo + 127;

    if(fmod(entrada, 2) !=0){
      mantissa = mantissa/2;
    }

    return mantissa;
}
```

Foram obtidos alguns testes para a comparação, primeiramente foi obtidos os valores reais das seguintes raízes e em seguida os valores pelo algoritmo:

Valores Reais:	Newton-Raphson:
$\sqrt{7.45} = 2.729468812791236125575$	$\sqrt{7.45} = 2.729468812791236125752$
$\sqrt{18.232} = 4.269894612282602929957$	$\sqrt{18.232} = 4.269894612282602930092$
$\sqrt{34.675} = 5.888548208174914823077$	$\sqrt{34.675} = 5.888548208174914823029$

Podemos perceber que a precisão atinge as 18 casas, após isso, o método começa a falhar. O erro

acontece porque o algoritmo para de calcular a raiz quando ele atinge a precisão especificada, ou seja, após isso ele ignora gerando o erro.

2 Triangulação de pontos no espaço a partir de receptores de sinal

1) Considerando que uma das fontes de imprecisão são erros nos dados, quais são os elementos envolvidos nesse problema que podem comprometer a qualidade dos dados? Por quê?

Resposta: Um dos erros são as interferências do meio externo que afeta o deslocamento do sinal até os receptores como por exemplo o sinal emitidos por outros aparelhos (celulares, controles remotos, roteador wi-fi, micro-ondas entre outros), além das barreiras físicas que enfraquecem o sinal como por exemplo pessoas, animais, construções entre outros.

Mas o principal erro é a transformação de dados que os receptores fazem ao receber esses sinais de um aparelho analógico, sabemos que a perda de dados é grande pois é muito difícil obter 100%. da precisão dos dados enviados devido a falta de bits para o armazenamento desses dados, gerando a imprecisão, ou seja, os erros nos decimais.

2) Apresente (implemente) uma solução para estimar a localização do emissor. Aplique essa solução para os dois casos acima e discuta sobre os erros encontrados.

Resposta: A seguir o algoritmo implementado para o problema:

```
#linguagem: python
import math
import numpy as np

xi = [1.55, -4.02, -4.40, 9.27, 9.15] # Coordenada x dos receptores
yi = [17.63, 0, 9.6, 4.64, 12] # Coordenada y dos receptores
zi = [1.35, 1.35, 1.35, 1.35, 1.35] # Coordenada z dos receptores

ro0 = [-26, -33.8, -29.8, -31.2, -33]
lk = [2.1, 1.8, 1.3, 1.4, 1.5]

rok = [-48.4, -50.6, -32.2, -47.4, -46.3] # Caso 1
xreal = 0 # Caso 1
yreal = 9 # Caso 1
#rok = [-46.9, -46.4, -41.2, -45.8, -48.7] # Caso 2
#xreal = 3 # Caso 2
#yreal = 3 # Caso 2

zreal = 1.24 # Caso1 e Caso2
dk = []
w = []
resultado = []
A = []
B = []

def distancia (ro0, rok, lk):
    dk = pow (10,(ro0 - rok)/(10*lk))
    return dk

def calculow (xi, yi, dk):
    wi = pow( xi,2) + pow(yi,2)  - pow(dk,2)
    return wi

for i in range(5):
    dk.append(distancia(ro0[i], rok[i], lk[i]))
    w.append(calculow(xi[i], yi[i], dk[i]))


A = np.matrix([[xi[1] - xi[0], yi[1] - yi[0]],
               [xi[2] - xi[1], yi[2] - yi[1]],
```

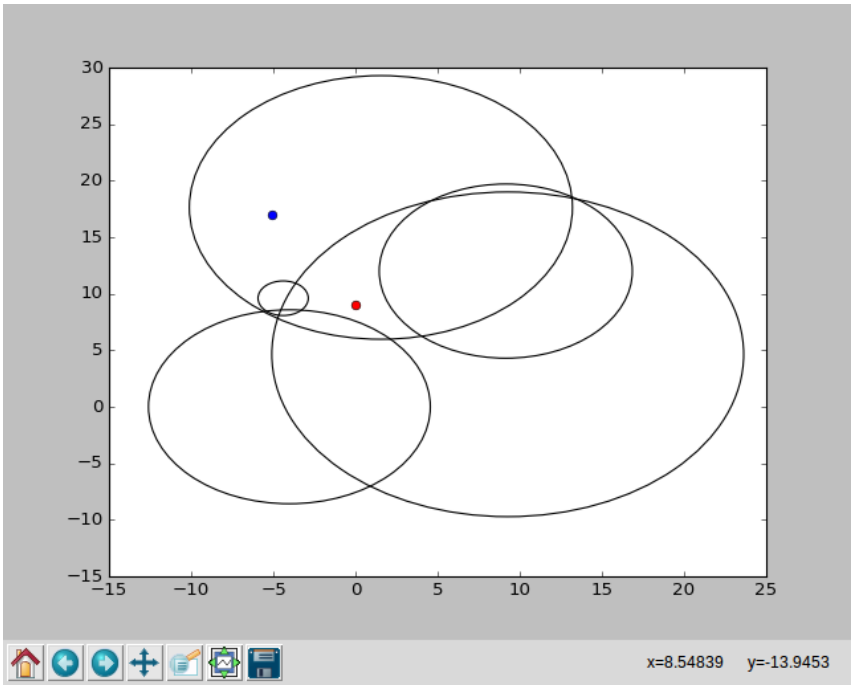
```
                [xi[3] - xi[2], yi[3] - yi[2]],
                [xi[4] - xi[3], yi[4] - yi[3]],
                [xi[0] - xi[4], yi[0] - yi[4]]])

B = np.matrix([[w[1] - w[0]],
               [w[2] - w[1]],
               [w[3] - w[2]],
               [w[4] - w[3]],
               [w[0] - w[4]]])

resultado = (A.T * A).I * A.T * B

print("Resultado")
print("x = ", resultado[0])
print("y = ", resultado[1])

import matplotlib.pyplot as plt
circle1 = plt.Circle((xi[0], yi[0]), dk[0], color='black', clip_on= True, fill = False)
circle2 = plt.Circle((xi[1], yi[1]), dk[1], color='black', clip_on= True, fill = False)
circle3 = plt.Circle((xi[2], yi[2]), dk[2], color='black', clip_on= True, fill = False)
circle4 = plt.Circle((xi[3], yi[3]), dk[3], color='black', clip_on= True, fill = False)
circle5 = plt.Circle((xi[4], yi[4]), dk[4], color='black', clip_on= True, fill = False)

fig, ax = plt.subplots()
ax = plt.gca()
ax.cla()

ax.set_xlim((-15,25))
ax.set_ylim((-15,30))
ax.plot(xreal, yreal, 'o', color='red')
ax.plot(resultado[0], resultado[1], 'o', color='blue')

ax.add_patch(circle1)
ax.add_patch(circle2)
ax.add_patch(circle3)
ax.add_patch(circle4)
ax.add_patch(circle5)

plt.show()
```

Resultados:

Caso 1:

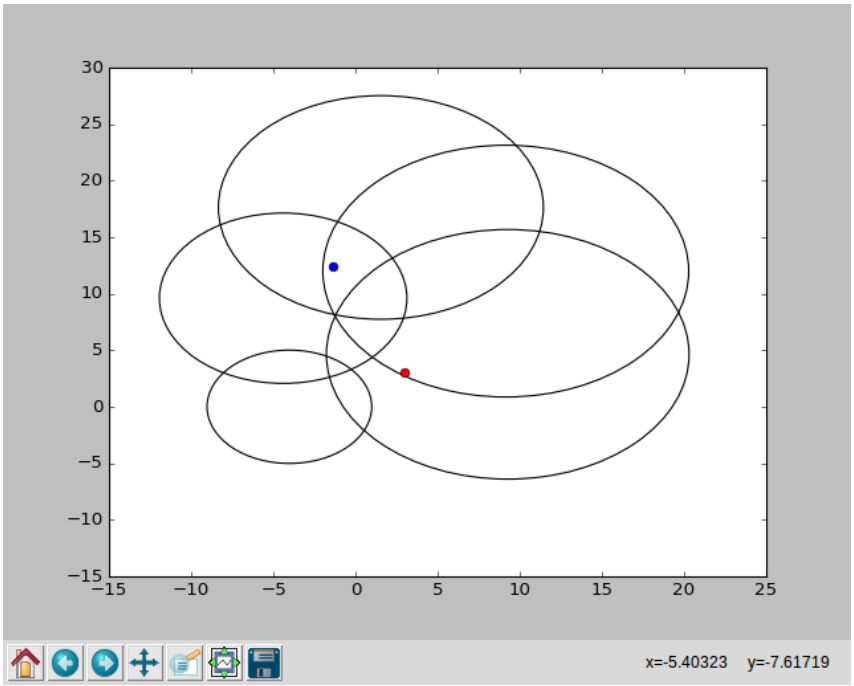
Posição real: (0.00, 9.00) **Círculo Vermelho**

Posição obtida: (-5.1169536, 16.93954403) **Círculo Azul**



Caso 2:

Posição real: (3.00, 3.00) **Círculo Vermelho**
Posição obtida: (-1.34862745, 12.43576921) **Círculo Azul**



Vimos que os resultados apresentados foram bem diferentes do que esperávamos, da posição real.

Isso acontece pelos fatores descritos na questão 1, como também pela maneira como os cálculos são feitos, pela ordem ordem que se calcula as matrizes (matriz A) que também modifica o valor.

Por exemplo, se modificarmos a formula dessa matriz, alterando a ordem dos elementos, ou o índice deles (contando que todos os índices sejam usados) cada uma delas gerará um valor diferente.
Também é muito importante lembrar que os erros também acontecem por arredondamentos criados no truncamento de casas decimais.