Projeto de Georreferenciamento com Três Satélites

Computação I - B - 12588

Prof[a]. Anna Rafaela Silva Ferreira

8 de junho de 2024

1 Objetivo do Projeto

O objetivo deste projeto é introduzir os alunos ao princípio matemático da trilateração, aplicado no posicionamento utilizando GPS. Através de uma série de atividades, os alunos irão aprender a resolver problemas práticos de determinação de localização a partir de dados de distância entre um receptor (telefone celular) e três estações base. Isso tudo com a aplicação prática em programação em Python.

2 Data de Entrega e Avaliação

03/07/2024

3 Entrega do trabalho

- O trabalho deverá ser feito em trios.
- O programa solicitado deverá ser entregue em um arquivo que deverá ser nomeado com o nome e sobrenome dos alunos conforme o exemplo abaixo. **Arquivos com outros nomes não serão aceitos**.
  Exemplo: NomeAluno1_NomeAluno2_NomeAluno3.py ou
  NomeAluno1_NomeAluno2_NomeAluno3.ipynb
- O programa DEVE estar na extensão ".py"ou ".ipynb". Programas com qualquer outra extensão ou sem extensão **não serão aceitos**. Fiquem atentos ao arquivo que ir˜ao enviar.
- Trabalhos com estruturas e/ou organizações semelhantes (plágio) serão penalizados com a nota zero.
- O programa deve ser passível de compilação e de posterior execução sem erros.
- O programa que não obedecer às restrições estabelecidas receberá zero.
- Este trabalho possui nota total igual a 10,0 (dez pontos).
- Este trabalho corresponde a 30% da Média Parcial da disciplina.
- Antes de escrever o código, faça o estudo do problema e o planejamento da sua solução.
- Lembre-se de documentar seu código (exemplo 1, exemplo 2).

4 Descrição do Problema

Há três estações base capazes de enviar e receber sinais do seu telefone celular, denominado P. No contexto de um sistema de coordenadas retangulares, as localizações dadas das três estações base A, B e C são (x_A,y_A), (x_B,y_B) e (x_C,y_C) respectivamente, onde cada unidade representa 1 km. As distâncias entre P e as estações base A, B e C são encontradas como PA km, PB km e PC km respectivamente. Considerando que A, B, C e P estão no mesmo plano horizontal, o objetivo é determinar as coordenadas do ponto P (x_P,y_P). As distâncias entre um satélite e um celular pode ser encontrada multiplicando o tempo de transmissão do sinal pela velocidade do sinal (que é a velocidade da luz $\simeq 299792,458$ km/s)

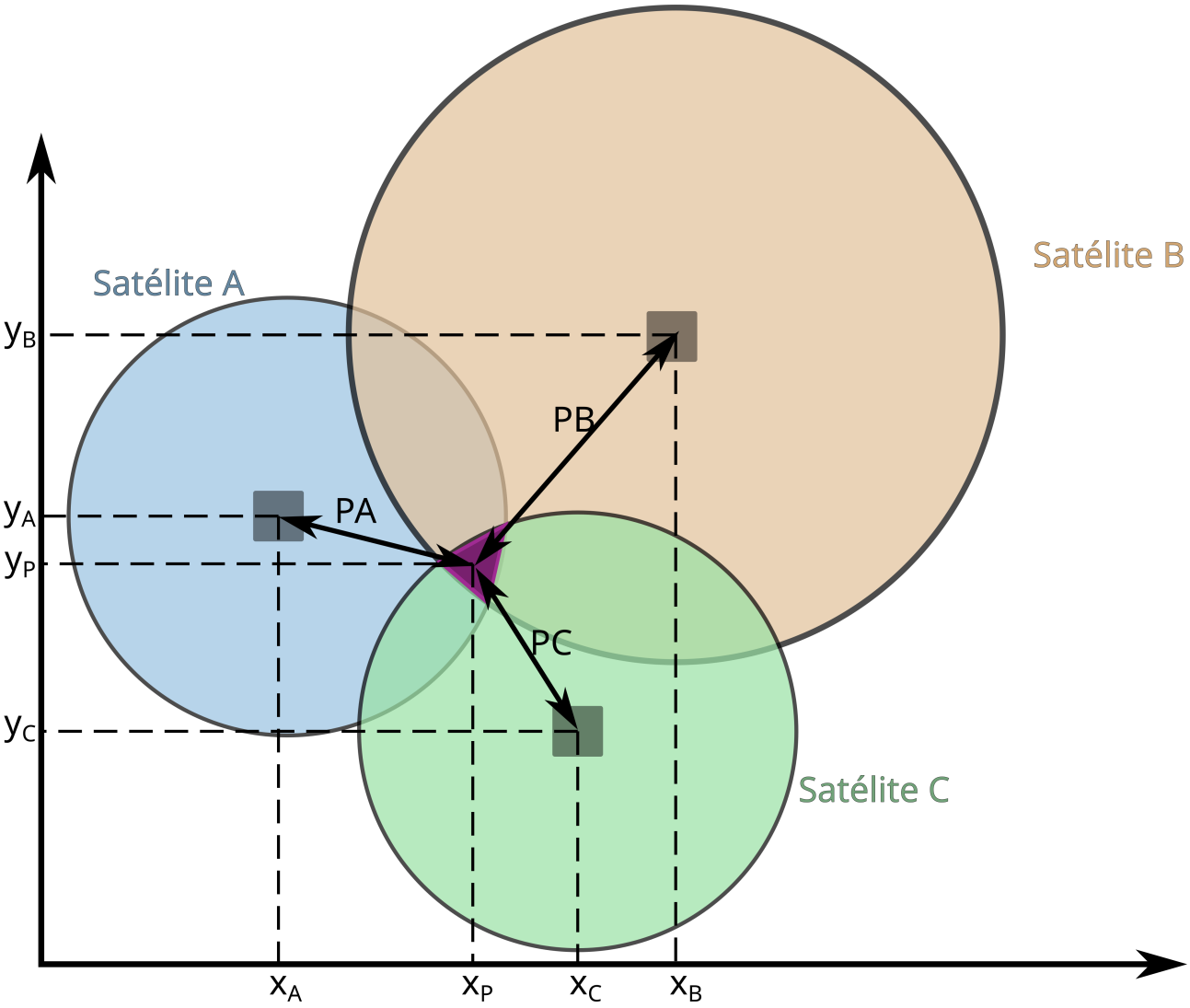

Figura 1: Trilateração de satélites para encontrar um ponto.

5 Entradas

- Posição de cada satélite em longitude e latitude. Se o satélite no sul ou no oeste, colocar sinal negativo no valor.
- Tempo para os sinais chegarem ao receptor GPS de cada satélite em segundos.

Esses dados devem ser **armazenados** em uma matriz $3x3$, onde as linhas são relativas a cada satélite e as colunas são, na ordem, abscissa x, ordenada y e tempo. As coordenadas dos

satélites devem estar em uma tupla. Ou seja:

$$[[(x_A,\ y_A),\ tempo_A],\\ [(x_B,\ y_B),\ tempo_B],\\ [(x_C,\ y_C),\ tempo_C]]$$

6 Saídas

- Posição do ponto P, em longitude e latitude. Se o satélite no sul ou no oeste, colocar sinal negativo no valor.
- Gráfico mostrando os pontos dos satélites e do ponto P

7 Restrições

- Implementação de funções (pelo menos três funções, fora a função de gráfico dada).
- Obrigatoriedade do módulo *math* e/ou módulo *numpy*. Outros módulos podem ser acrescidos.
- Não permitir tempo negativo ou igual a zero segundos.
- Não permitir posição igual para os satélites.
- Os dados entregues não devem ser tratados ao longo do programa como uma variável do tipo *float*, ou seja, não devem ser armazenadas em variáveis primitivas em nenhum momento do seu programa.
- O programa deve continuar rodando até que o usuário digite uma posição $(0, 0)$ para qualquer um dos satélites.
- O código deve estar documentado (exemplo 1, exemplo 2).

8 Testes

- Exemplo 1:

  Neste exemplo, o satélite A está numa longitude 153.3 S e em uma latitude 23.8 O, o satélite B está numa longitude 61.9 S e em uma latitude 54.8 O e o satélite C está numa longitude 121.9 S e em uma latitude 27.7 L

  Entradas:

  Dados satélite A: $-153.3,\ -23.8,\ 2.3396413371271625\ s$
  Dados satélite B: $-61.9,\ 54.8,\ 0.7857048249926033\ s$
  Dados satélite C: $-121.9,\ 27.7,\ 1.9881647472319892\ s$

  Saída: $(-42.8, -5.09)$

  Gráfico:

- Exemplo 2:

  Neste exemplo, o satélite A está numa longitude 25.6 S e em uma latitude 21.7 O, o satélite B está numa longitude 49 S e em uma latitude 38.3 O e o satélite C está numa longitude 78.6 S e em uma latitude 8.8 L

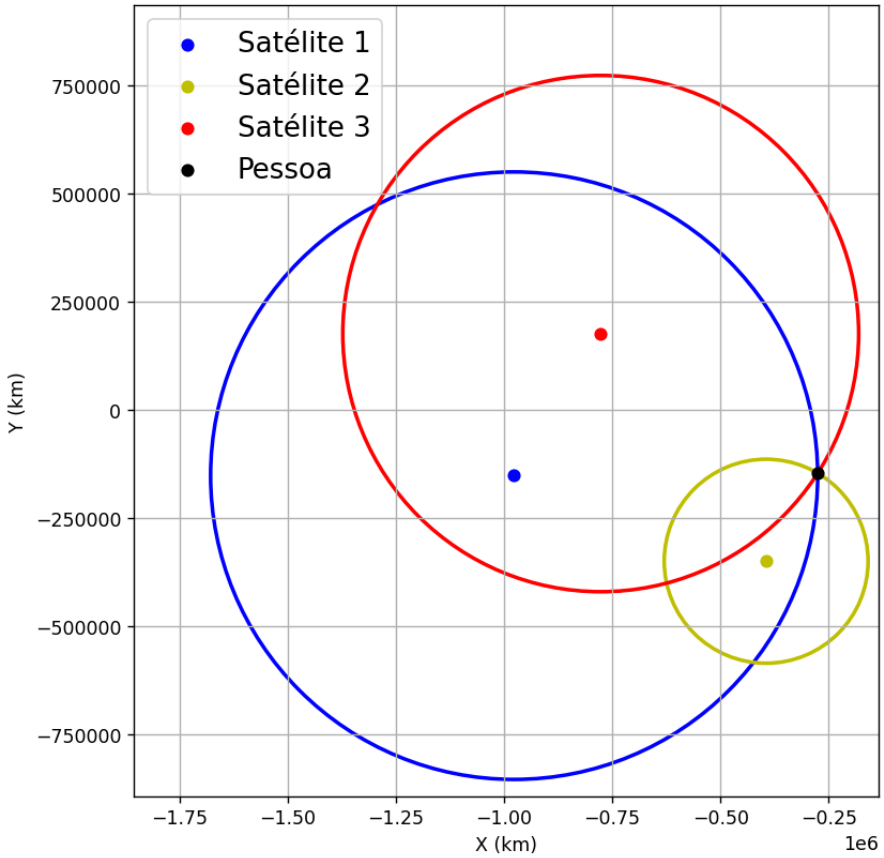

Figura 2: Exemplo 1

Entradas:

Dados satélite A: $-25.6,\ -21.7,\ 0.5081397743277849\ s$
Dados satélite B: $-49,\ -38.3,\ 0.7179516554333171\ s$
Dados satélite C: $-78.6,\ 8.8,\ 0.8160558987853649\ s$

Saída: $(-43.21, -22.90)$

Gráfico:

- Exemplo 3:

  Neste exemplo, o satélite A está numa longitude 61 N e em uma latitude 54.7 L, o satélite B está numa longitude 31.3 S e em uma latitude 49.2 L e o satélite C está numa longitude 23.2 S e em uma latitude 2.9 O

  Entradas:

  Dados satélite A: $61,\ 54.7,\ 1.2525775454508472\ s$
  Dados satélite B: $-31.3,\ 49.2,\ 0.7151472310610769\ s$
  Dados satélite C: $-23.2,\ -2.9,\ 1.2264939709557763\ s$

  Saída: $(2.35, 48.85)$

  Gráfico:

9 Referências Sugeridas

- `https://www.gps.gov/multimedia/tutorials/trilateration/`
- `https://gisgeography.com/trilateration-triangulation-gps/`

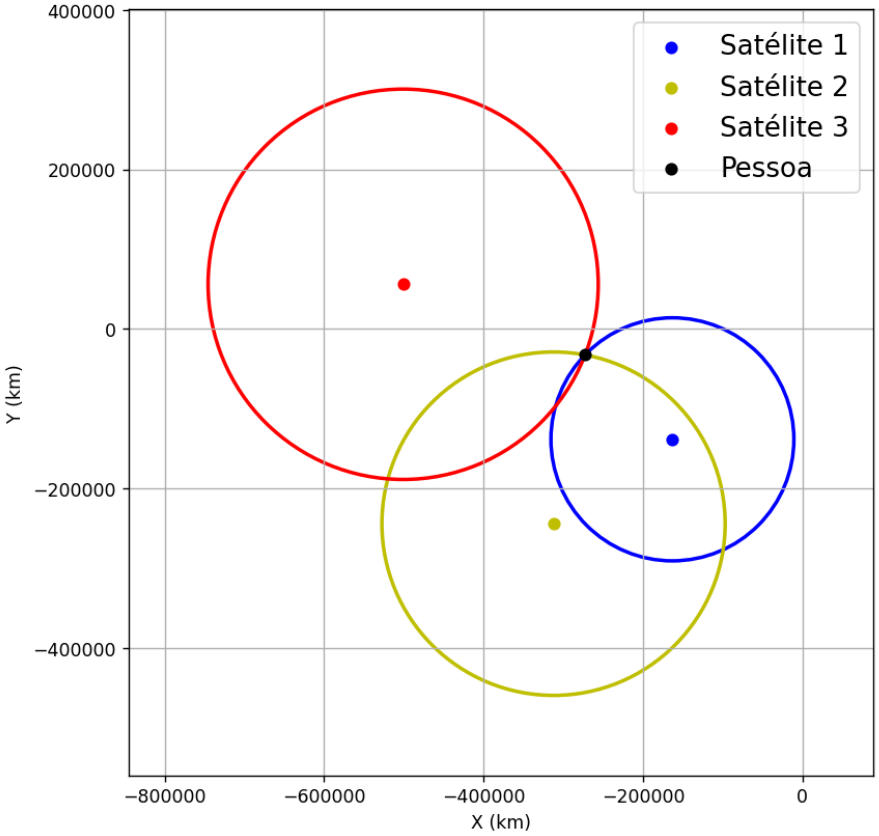

Figura 3: Exemplo 2

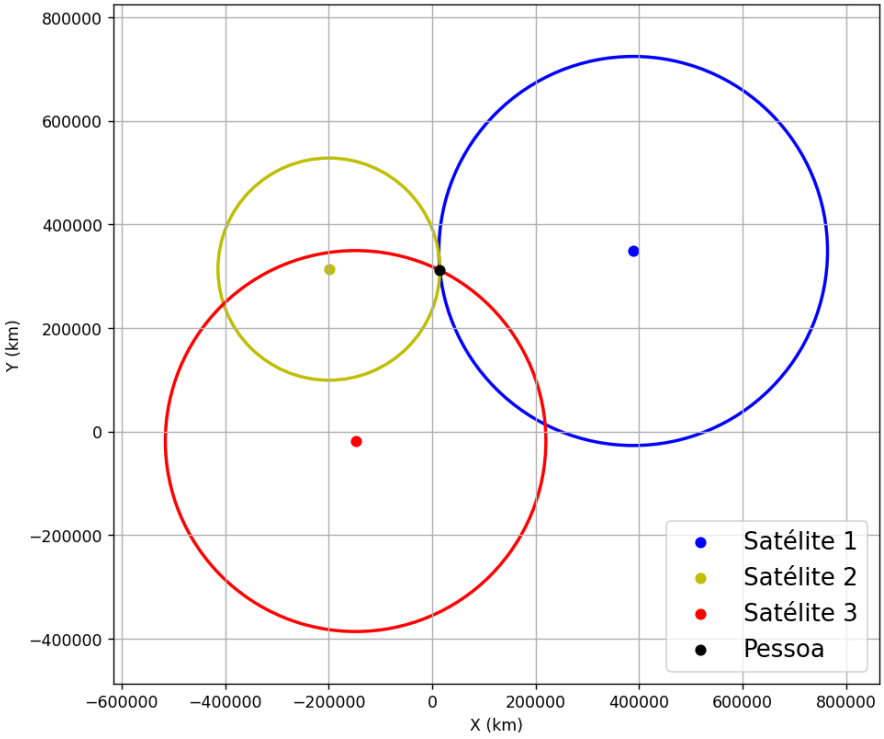

Figura 4: Exemplo 3

- https://paginas.fe.up.pt/~ee02108/trilatera%C3%83%C2%A7%C3%83%C2%A3o.pdf
- https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.fsolve.html
- https://in-the-sky.org/satmap_worldmap.php
- https://www.n2yo.com/?s=43567
- https://www.edb.gov.hk/attachment/en/curriculum-development/kla/ma/res/STEM%20example_sec_trilateration%20GPS_eng.pdf
- https://letstalkscience.ca/educational-resources/backgrounders/math-behind-gps
- https://www.w3schools.com/python/python_lambda.asp

10 Apêndice

```
import matplotlib.pyplot as plt

def plotar_circulos(pontos, distancias):
    """
    Plota circulos representando as distancias de pontos
    especificos e maximiza a janela do grafico.
    Essa funcao recebe os pontos em x e y (e nao em longitude e
    latitude)

    Args:
        pontos (list of tuples): Lista de coordenadas (x, y) para
        os pontos.
        distancias (list of floats): Lista de distancias correspondentes
        a cada ponto.

    """
    SIZE = 15

    # Criacao da figura e dos eixos
    fig, ax = plt.subplots()

    # Maximizar a janela do grafico
    figManager = plt.get_current_fig_manager()
    figManager.window.showMaximized()

    # Configuracoes de tamanho de fonte para diferentes elementos
    # do grafico
    plt.rc('font', size=SIZE)          # Tamanho padrao do texto
    plt.rc('axes', titlesize=SIZE)     # Tamanho da fonte do titulo
                                       # dos eixos
    plt.rc('axes', labelsize=SIZE)     # Tamanho da fonte dos rotulos
                                       # dos eixos x e y
```

```
plt.rc('xtick', labelsize=SIZE)     # Tamanho da fonte dos rotulos
                                    # dos ticks do eixo x
plt.rc('ytick', labelsize=SIZE)     # Tamanho da fonte dos rotulos
                                    # dos ticks do eixo y
plt.rc('legend', fontsize=SIZE)     # Tamanho da fonte da legenda

# Definicao de cores para os pontos e circulos
cores = ['b', 'y', 'r', 'k']

# Plotar os pontos e os circulos correspondentes
for i, ponto in enumerate(pontos[:-1]):
  ax.scatter(ponto[0],ponto[1],color=cores[i],label=
                                        f'Satelite {i+1}')
  circulo = plt.Circle((ponto[0], ponto[1]), distancias[i],
                              color=cores[i], fill=False, linewidth=2)
  ax.add_artist(circulo)

# Plotar o ultimo ponto (pessoa)
ax.scatter(pontos[-1][0], pontos[-1][1], color=cores[-1],
                                        label='Pessoa')

# Determinar os limites dos eixos com base nas distancias
x_min = min(p[0] - d for p, d in zip(pontos[:-1], distancias))
x_max = max(p[0] + d for p, d in zip(pontos[:-1], distancias))
y_min = min(p[1] - d for p, d in zip(pontos[:-1], distancias))
y_max = max(p[1] + d for p, d in zip(pontos[:-1], distancias))

# Ajustar os limites dos eixos
ax.set_xlim(x_min - 1e5, x_max + 1e5)
ax.set_ylim(y_min - 1e5, y_max + 1e5)

# Configuracoes dos rotulos e do aspecto dos eixos
ax.set_xlabel('X (km)')
ax.set_ylabel('Y (km)')
ax.set_aspect('equal', adjustable='box')

# Adicionar legenda e grid ao grafico
plt.legend()
plt.grid(True)

# Exibir o grafico
plt.show(block=False)
```