Modelação Numérica 2017

Aula 17, 26/Abr

- Estimativa de parâmetros e optimização

http://modnum.ucs.ciencias.ulisboa.pt

Estimativa/optimização de parâmetros

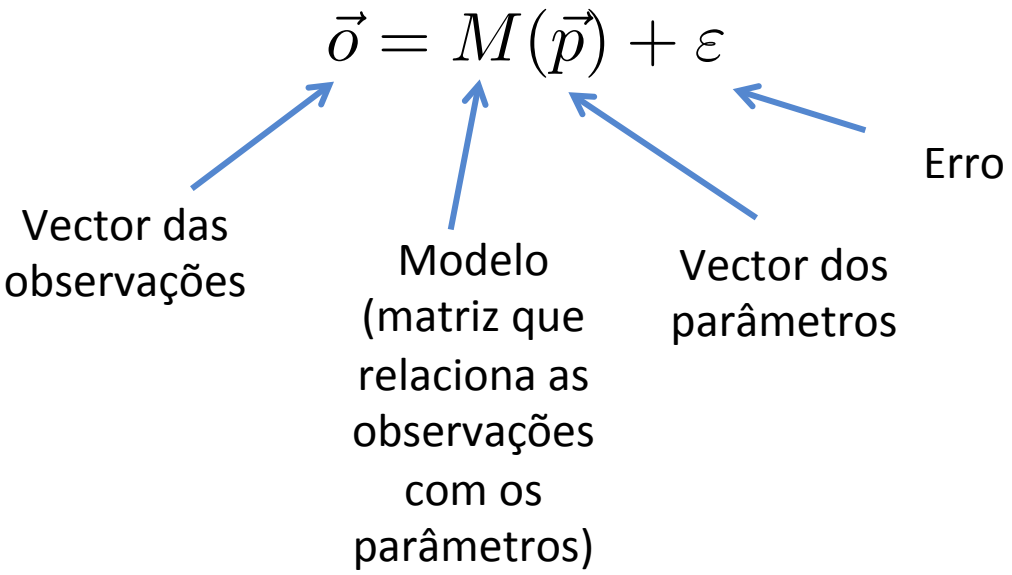
- Muitas vezes queremos estimar parâmetros que não conseguimos medir directamente. No entanto, podemos conseguir relacionar os parâmetros que queremos estimar com medições que conseguimos fazer. Dizemos então que queremos estimar parâmetros de um modelo. [Por exemplo, queremos descobrir a densidade das rochas/minérios a certa profundidade].
- Realizamos então um conjunto de medidas (observações) que sabemos relacionar com os parâmetros a estimar [Por exemplo, vamos medir a aceleração da gravidade à superfície da Terra, para estimar a densidade em profundidade].
- As medições têm erro, e os parâmetros estimados também têm erro.

Estimativa/optimização de parâmetros

- Se eu tiver o mesmo número de observações do que parâmetros a estimar, posso conseguir resolver o problema de forma única (é como ter duas equações a duas incógnitas), se as observações forem consistentes. Mas não tenho dados independentes para verificar a solução obtida.
- De forma mais geral, o meu problema pode ser sobre-determinado (tenho mais observações do que parâmetros a constranger).
- E pode também ser sub-determinado (tenho menos observações do que parâmetros a constranger, e neste caso só vou conseguir determinar como variam conjuntos de parâmetros).
- Em qualquer dos casos, torna-se necessário estabelecer uma metodologia "óptima" para encontrar o "melhor" conjunto de parâmetros que se ajusta aos dados => Optimização ou ajuste de parâmetros.

Estimativa/optimização de parâmetros

- Em linguagem matemática, o problema a resolver é o seguinte:



- Note-se que, de forma geral, o vector das observações e dos parâmetros têm dimensões diferentes.

Estimativa/optimização de parâmetros

- Em linguagem matemática, o problema a resolver é o seguinte:

$$\vec{o} = M(\vec{p}) + \varepsilon$$

Vector das observações

Modelo (que relaciona as observações com os parâmetros)

Vector dos parâmetros

Erro

- Se o modelo fosse linear, ficaríamos com: $\vec{o} = L\vec{p} + \varepsilon$

sendo *L* a matriz dos coeficientes do modelo.

Problema linear sobredeterminado

- O caso mais simples é o da regressão linear:

$$y = ax + b$$

- Os parâmetros a estimar são: a, b.
- As observações são os valores y.

- Exemplo: Quero estimar a velocidade de um carro (v) e a sua posição de origem (x_0) [parâmetros], e para tal vou medir a sua posição (x) [observações] em determinados tempos (t).

$$x = vt + x_0$$

Problema linear sobredeterminado

- Se tiver N=2 observações (y), então vou obter uma solução única com erro nulo (pelo menos, formalmente).
- Se N>2, então o problema é sobre-determinado (tenho mais observações do que parâmetros a determinar). Neste caso, não existe uma solução exacta.
- Tratando-se de um modelo linear, podemos formular o problema na forma de um produto matricial:

$$\vec{o} = L\vec{p} + \varepsilon$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix} = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \dots & 1 \\ x_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \dots \\ \varepsilon_n \end{bmatrix}$$

Regressão linear (com constrangimento)

- A função numy.polyfit calcula os parâmetros (a, b) da regressão linear, ou os de uma regressão polinomial de ordem mais elevada (e.g: $y = ax^2 + bx + c$), minimizando o erro médio quadrático.

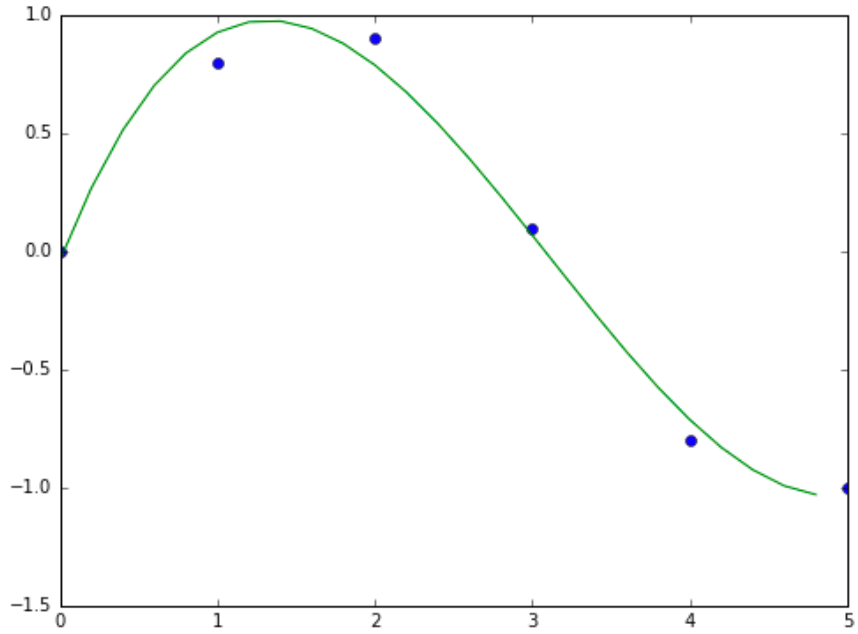
```
import matplotlib.pyplot as plt
import numpy as np

#%%
plt.rcParams['figure.figsize'] = 8, 6

x = np.array([0.0, 1.0, 2.0, 3.0,  4.0,  5.0])
y = np.array([0.0, 0.8, 0.9, 0.1, -0.8, -1.0])
p = np.polyfit(x, y, 3)

x2=np.arange(0,5,.2)
yfit = p[3] + p[2]*x2 + p[1]*x2**2 + p[0]*x2**3

plt.plot(x,y, 'o', x2,yfit)
```



Regressão linear (com constrangimento)

- Por vezes, precisamos de impôr condições aos parâmetros, por exemplo, podemos querer fazer o ajuste:

$$y = ax$$

- Ou seja, queremos impôr b=0. Neste caso, podemos resolver o problema explicitamente (o problema tem solução analítica simples). Para um dado valor de a, o erro médio quadrático é:

$$\overline{\varepsilon^2} = \frac{1}{N}\sum_{k=1}^{N}(y_k - ax_k)^2$$

- Para minimizarmos o erro quadrático médio, podemos derivá-lo e igualá-lo a zero: $\frac{\partial \overline{\varepsilon^2}}{\partial a} = 0$

Regressão linear (com constrangimento)

$$\frac{\partial \overline{\varepsilon^2}}{\partial a} = 0$$

$$\frac{\partial}{\partial a}\left(\frac{1}{N}\sum_{k=1}^{N}(y_k - ax_k)^2\right) = 0$$

$$\frac{\partial}{\partial a}\left(\frac{1}{N}\sum_{k=1}^{N} y_k^2 - 2ax_k y_k + a^2 x_k^2\right) = 0$$

$$\frac{1}{N}\sum_{k=1}^{N} -2x_k y_k + 2ax_k^2 = 0$$

$$\sum_{k=1}^{N} -2x_k y_k + 2ax_k^2 = 0$$

$$\sum_{k=1}^{N} 2x_k y_k = \sum_{k=1}^{N} 2ax_k^2$$

$$\sum_{k=1}^{N} x_k y_k = \sum_{k=1}^{N} ax_k^2$$

$$a = \frac{\sum_{k=1}^{N} x_k y_k}{\sum_{k=1}^{N} x_k^2}$$

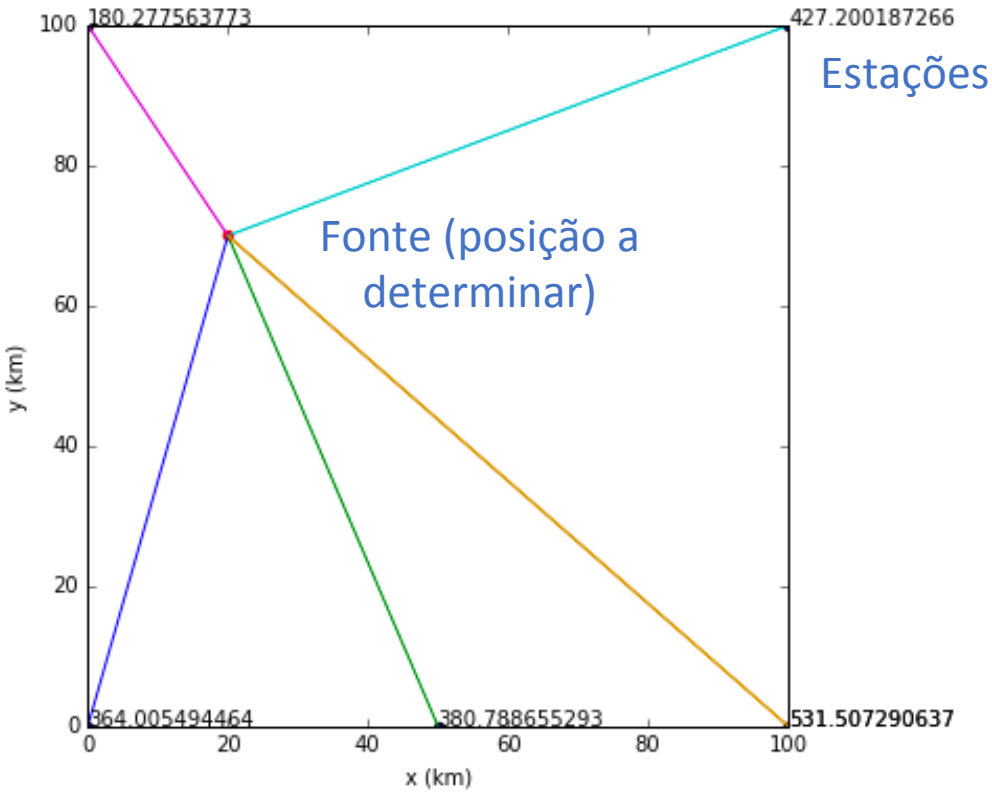
Caso geral: função custo

- Em geral, não é possível obter diretamente (analiticamente) uma solução óptima.
- Nesses casos, é necessário proceder iterativamente.
- Os métodos iterativos requerem:
- Um método para obter potenciais soluções, e de as aceitar se for caso disso.
  - Função custo J, ex:
    - Minimização do erro (norma L1): $J = \varepsilon = |\vec{o} - M(\vec{p})|$
    - Minimização do erro quadrático (norma L2): $J = \varepsilon^2 = |\vec{o} - M(\vec{p})|^2$
  - Aceitar soluções que (pelo menos em média) reduzam J.
- Um critério de paragem:
  - Número de iterações.
  - Valor atingido pela função custo.
  - Falta de progresso em J ou em p

Um problema real:

- Temos uma fonte (sismo, tsunami, emissor GPS, etc) que gera ondas que se propagam e que são registadas em estações.
- Nas estações mais longe da fonte, as chegadas das ondas são registadas mais tarde.

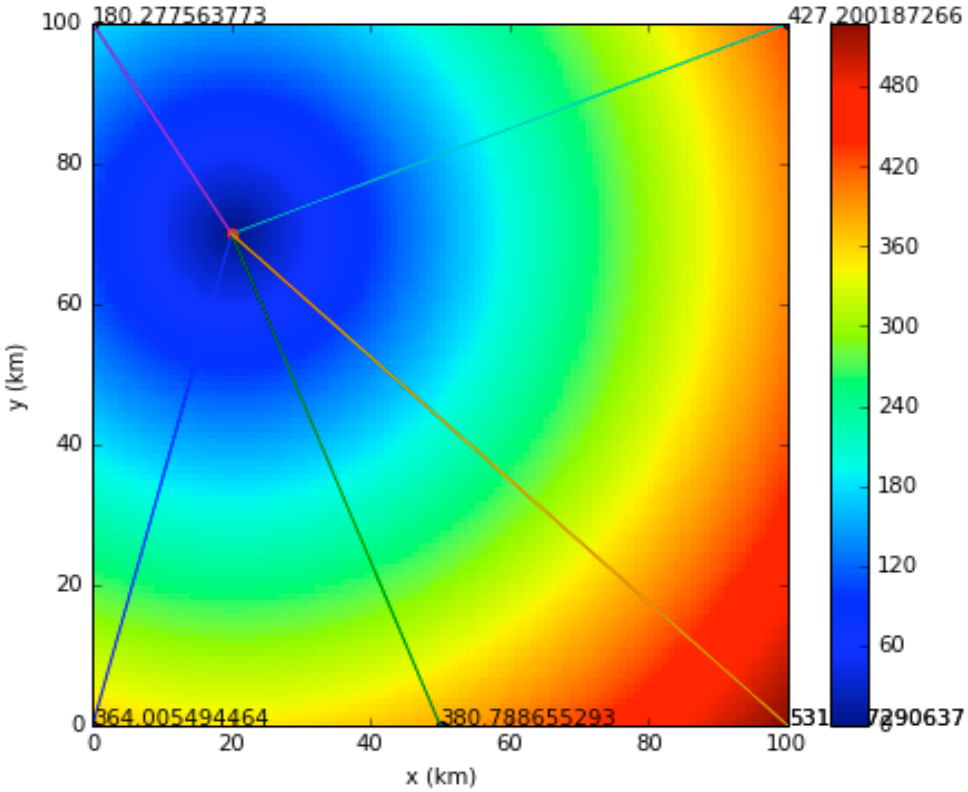
- Queremos localizar a fonte (sismo, tsunami, emissor GPS)
- Conhecendo os tempos de chegada das ondas às estações.
- E sabendo a velocidade de propagação das ondas (caso simples, c=contante)



Um problema real:

- Temos uma fonte (sismo, tsunami, emissor GPS, etc) que gera ondas que se propagam e que são registadas em estações.
- Nas estações mais longe da fonte, as chegadas das ondas são registadas mais tarde.

- Queremos localizar a fonte (sismo, tsunami, emissor GPS)
- Conhecendo os tempos de chegada das ondas às estações.
- E sabendo a velocidade de propagação das ondas (caso simples, c=contante)



```python
import matplotlib.pyplot as plt
import numpy as np

#%%
plt.rcParams['figure.figsize'] = 10, 6

################## Inicializar

#%% Setup

# Limites do domínio espacial
xmin=0.
xmax=100e3
ymin=0.
ymax=100e3

# Espaçamento entre pontos
dx=1e3
dy=1e3

# Pontos em x e y
x=np.arange(xmin,xmax+dx,dx)
y=np.arange(ymin,ymax+dy,dy)

# Nr de pontos
nx=len(x)
ny=len(y)

# velocidade de propagação do sinal
c=200
```

```
# Matrizes 2D de x, y, t
xx=np.zeros([nx,ny])
yy=np.zeros([nx,ny])
tt=np.zeros([nx,ny])

for ix in range(nx):
    for iy in range(ny):
        xx[ix,iy]=ix*dx
        yy[ix,iy]=iy*dy

xr=np.array([xmin,xmax])/1e3      # x range
yr=np.array([ymin,ymax])/1e3      # y range

#%% Estações e fonte

# Coordenadas das estações
xE=np.array([0, 50, 100, 100,0, 100])*1e3
yE=np.array([0, 0, 0, 100, 100,0])*1e3

# Coordenadas da fonte
xF=20e3
yF=70e3;

# Propagação do sinal
distE=np.sqrt((xE-xF)**2+(yE-yF)**2)    # Distância Fonte-Estações
tE=distE/c;                             # Tempo de chegada da onda às estações

# Tempo de chegada da onda a cada ponto da grelha
dd=np.sqrt((xx-xF)**2+(yy-yF)**2)       # Distância
tt=dd/c                                 # Tempo
```

```
#%% Plot

plt.rcParams['figure.figsize'] = 7,6
plt.close()

plt.pcolor(xx/1e3,yy/1e3,tt)           # matriz dos tempos de propagação
plt.colorbar()
plt.scatter(xF/1e3,yF/1e3, color='r');      # fonte

# estações
for iE in range(len(xE)):
    plt.plot([xE[iE]/1000, xF/1000], [yE[iE]/1000, yF/1000])
    plt.scatter(xE[iE]/1e3,yE[iE]/1e3)
    plt.text(xE[iE]/1e3,yE[iE]/1e3, str(tE[iE]))

plt.xlabel('x (km)')
plt.ylabel('y (km)')
plt.xlim(xr)
plt.ylim(yr)

plt.savefig('p1-TempoPropagacao.png')
```