Modelação Numérica 2017

Aula 23, 17/Maio

- Estimativa de parâmetros e optimização.
- Solução de problemas lineares sub- e sobre-determinados.

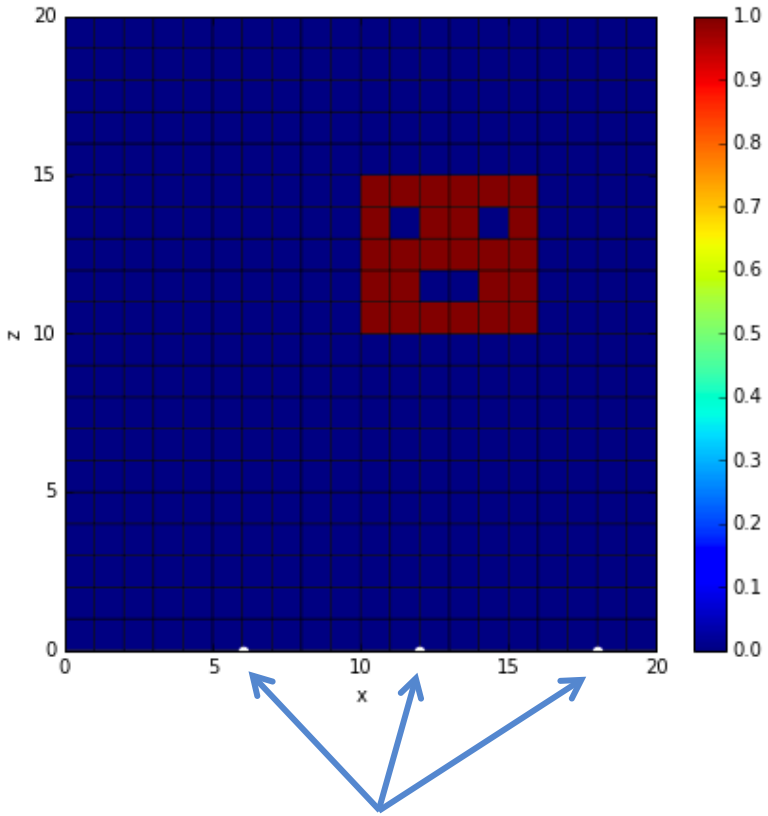
http://modnum.ucs.ciencias.ulisboa.pt

Outro exemplo:

Recuperar a forma de um objecto (nuvem, massa enterrada, …)

Objecto a caracterizar:

nx*nz = 441 parâmetros (0 ou 1)

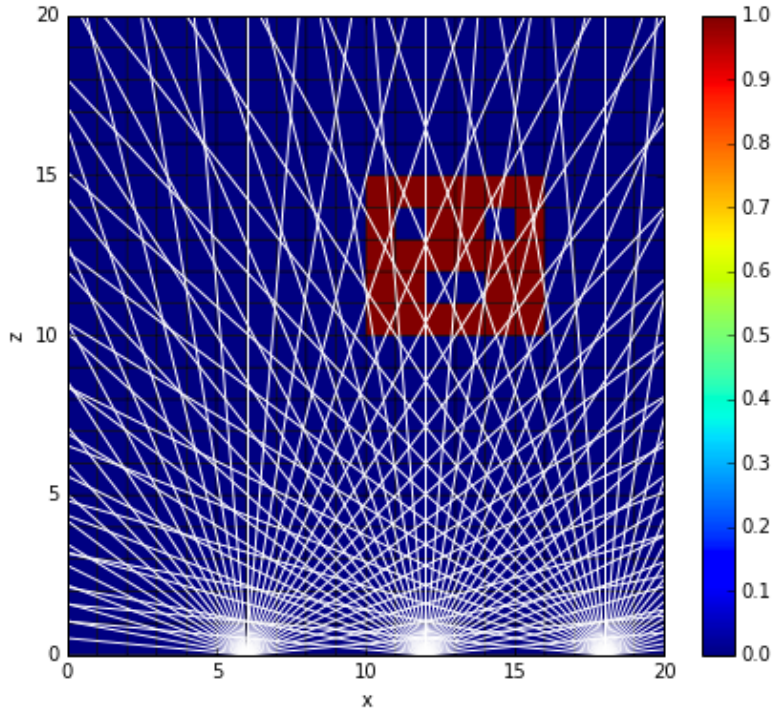


3 estações

Observações:

3 estações*35 ângulos

= 105 observações (0 ou 1)



Cada sensor vê/mede em cada ângulo o integral ao longo do percurso.

Outro exemplo:

Recuperar a forma de um objecto (nuvem, massa enterrada, …)

Objecto a caracterizar:

nx*nz = 441 parâmetros (0 ou 1)

m=441

Observações:

3 estações*35 ângulos

= 105 observações (0 ou 1)

n=105

Vector das observações

Modelo (matriz que relaciona as observações com os parâmetros)

Vector dos parâmetros

$$\vec{o} = G\vec{p}$$

$$\begin{bmatrix} o_1 \\ o_2 \\ \dots \\ o_n \end{bmatrix} = \begin{bmatrix} G_{11} & G_{12} & \dots & G_{1m} \\ G_{21} & G_{22} & \dots & G_{2m} \\ G_{\dots 1} & G_{\dots 2} & \dots & G_{\dots m} \\ G_{n1} & G_{n2} & \dots & G_{nm} \end{bmatrix} \begin{bmatrix} p_1 & p_2 & p_{\dots} & p_m \end{bmatrix}$$

Outro exemplo:

Recuperar a forma de um objecto (nuvem, massa enterrada, …)

Objecto a caracterizar:

nx*nz = 441 parâmetros (0 ou 1)

m=441

Observações:

3 estações*35 ângulos

= 105 observações (0 ou 1)

n=105

Vector das observações

Modelo (matriz que relaciona as observações com os parâmetros)

Vector dos parâmetros

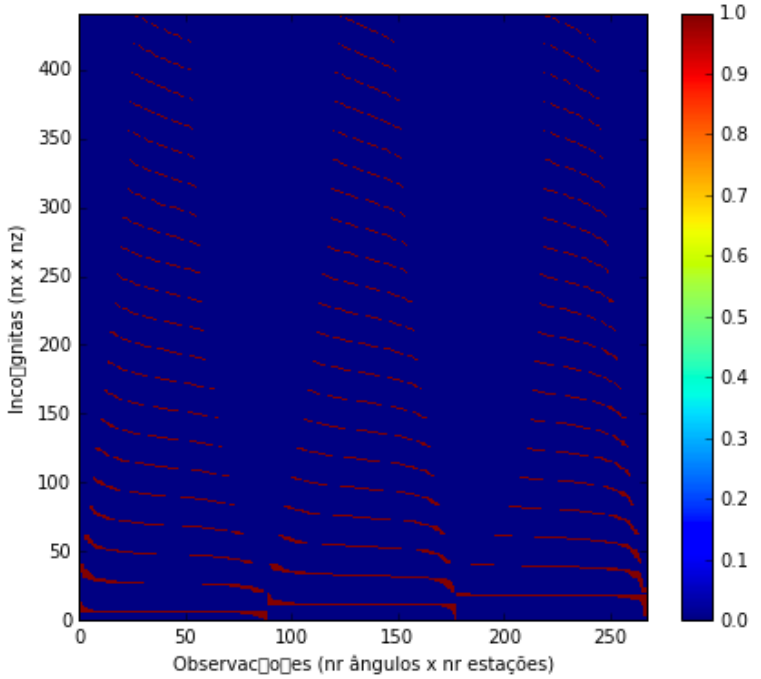
$$\vec{o} = G\vec{p}$$

Cada elemento de G descreve o efeito de cada parâmetro em cada observação.

$$\begin{bmatrix} o_1 \\ o_2 \\ \dots \\ o_n \end{bmatrix} = \begin{bmatrix} G_{11} & G_{12} & \dots & G_{1m} \\ G_{21} & G_{22} & \dots & G_{2m} \\ G_{\dots 1} & G_{\dots 2} & \dots & G_{\dots m} \\ G_{n1} & G_{n2} & \dots & G_{nm} \end{bmatrix} \begin{bmatrix} p_1 & p_2 & p_{\dots} & p_m \end{bmatrix}$$

Outro exemplo:
Recuperar a forma de um objecto (nuvem, massa enterrada, …)

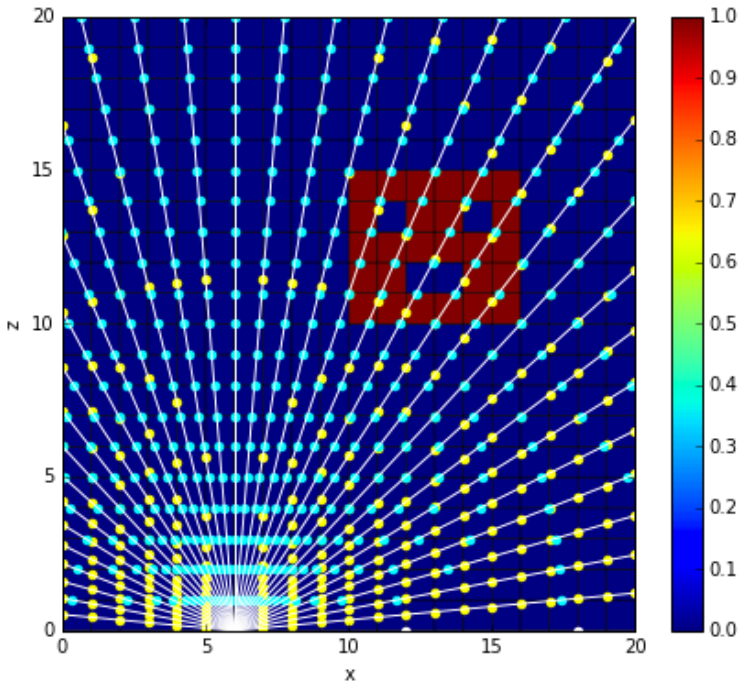
- m = 441 parâmetros
- n = 105 observações
- Cada linha de G tem 441 membros (tantos quantos os pixeis a calcular 21x21)
  - Foi preciso “vetorizar” (reshape) a matriz de parâmetros.
- Cada coluna de G tem 105 membros (3x35):
  - Por vetorização das observações.
- A matriz G depende da geometria (número de sensores, ângulos) e da física do problema (relação entre os valores no espaço físico e as observações remotas): é o modelo.
- A matriz G não depende das observações nem da solução.



$$\begin{bmatrix} o_1 \\ o_2 \\ \dots \\ o_n \end{bmatrix} = \begin{bmatrix} G_{11} & G_{12} & \dots & G_{1m} \\ G_{21} & G_{22} & \dots & G_{2m} \\ G_{\dots 1} & G_{\dots 2} & \dots & G_{\dots m} \\ G_{n1} & G_{n2} & \dots & G_{nm} \end{bmatrix} \begin{bmatrix} p_1 & p_2 & p_{\dots} & p_m \end{bmatrix}$$

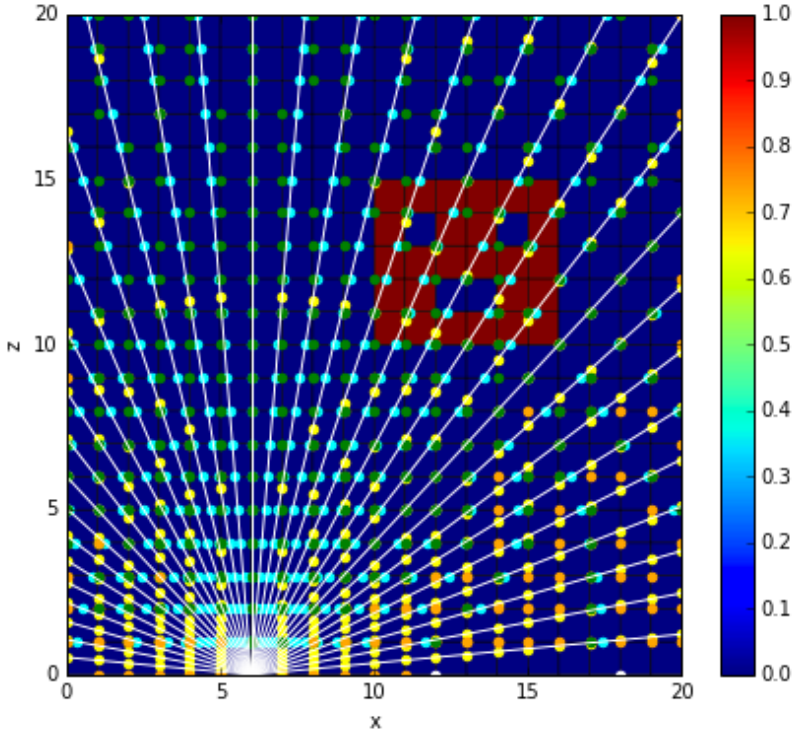
Determinação da matriz G

- Inicializar a matriz a 0.
- Cada elemento da matroiz G é
  - (1 observação vs 1 pixel)
  - Ou seja: (1 sensor + 1 ângulo vs 1 nx + 1 ny)
- Para cada sensor: Para cada ângulo localizar os pixeis atravessados e marcá-los a 1.



Determinação da matriz G

- Inicializar a matriz a 0.
- Cada elemento da matroiz G é
  - (1 observação vs 1 pixel)
  - Ou seja: (1 sensor + 1 ângulo vs 1 nx + 1 ny)
- Para cada sensor: Para cada ângulo localizar os pixeis atravessados e marcá-los a 1.
- Aproximar na grelha regular.



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

from math import pi as pi
from numpy.linalg import pinv as pinv

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

################### Inicialização

### número de pontos
nx=21
nz=21

### espaçamento entre pontos
dx=1.
dz=1.

### vectores x e z
x=np.arange(nx)*dx
z=np.arange(nz)*dz

### limited do domínio
xmin=np.min(x)
xmax=np.max(x)
zmin=np.min(z)
zmax=np.max(z)

### matrizes 2D em x e y
xx=np.zeros([nx,nz])
zz=np.zeros([nx,nz])

for ix in range(nx):
    for iz in range(nz):
        xx[ix,iz]=x[ix]
        zz[ix,iz]=z[iz];
```

```
### matrizes 2D com a solução a descobrir
SS=np.zeros([nx,nz])
for ix in range(10,16):
    for iy in range(10,15):
        SS[ix,iy]=1

SS[11,13]=0
SS[14,13]=0
SS[12:14,11]=0

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

plt.pcolor(xx,zz,SS, edgecolors='k', linewidths=.5)
plt.colorbar()
plt.xlabel('x')
plt.ylabel('z');
plt.xlim([xmin, xmax])
plt.ylim([zmin, zmax])
##%%
### Observações

### Pontos de observação (estações)
xE=np.array([6,12,18])*dx
zE=np.zeros(len(xE))

plt.scatter(xE,zE, color='white')
##%%
```

```
### ...
#da=5.
da=2.
#da=1.
#da=.5
angE=np.arange(da,180,da)         # ângulos
slopeE=np.tan(angE*pi/180.)      # declive

nEx=len(xE)              # nr de estações
nEa=len(angE)            # nr de ângulos
nobs=nEx*nEa             # nr total de observações
npar=nx*nz               # nr de parâmetros a encontrar/optimizar

## construção da matriz G
G=np.zeros([nobs,npar])      # matriz do modelo (ex: funções de Green)

zintx=np.zeros(nx)
zintz=np.zeros(nz)
xintz=np.zeros(nz)
xintx=np.zeros(nx)
```

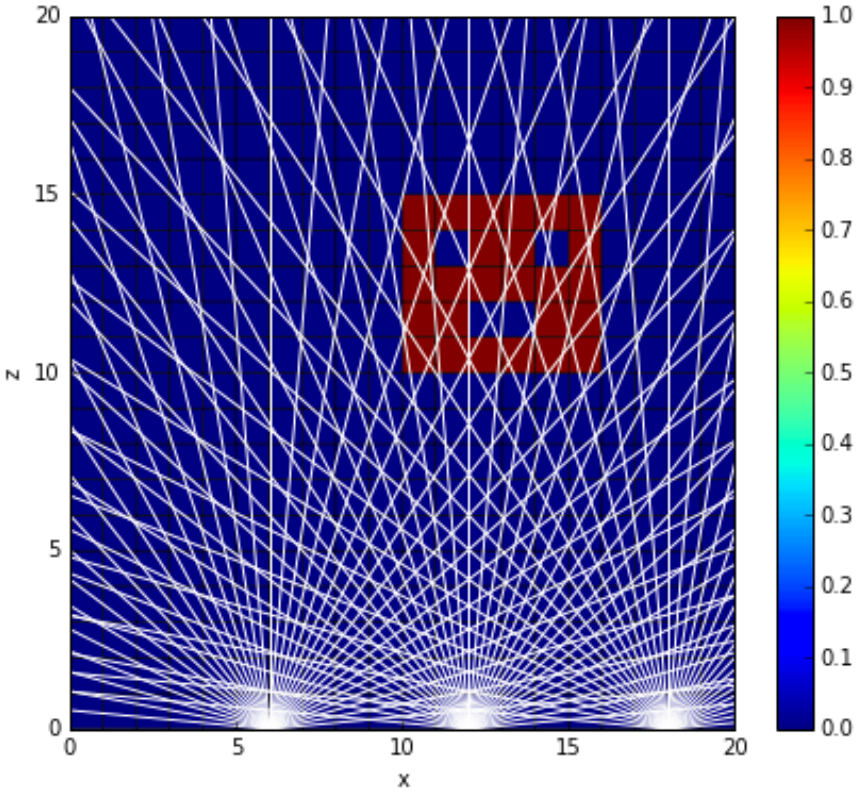
```
iObs=0;
for iEs in range(nEx):          # para cada estação
    for iEa in range(nEa):      # para cada ângulo
        xis=xE[iEs]             # x da estação
        slope=slopeE[iEa]       # declive correspondente ao ângulo (observação)

        for ix in range(nx):        # para cada ponto em x
            zintx[ix] = zE[iEs] + slope*(x[ix]-xis)
            xintx[ix] = x[ix];

        for iz in range(nz):        # para cada ponto em z
            zintz[iz] = z[iz]
            xintz[iz] = xis + (z[iz]-zE[iEs])/slope

        plt.plot(xintx,zintx, color='white')        # linhas
#        plt.scatter(xintx,zintx, color='yellow')    # pontos a x constante
#        plt.scatter(xintz,zintz, color='cyan')      # pontos a z constante


        for ix in range(nx):
            if zintx[ix]>=zmin and zintx[ix]<=zmax:
                iZ=np.round(zintx[ix]/dz)
                iPar=(iZ)*nx+ix
                G[iObs,int(iPar)]=1
#                plt.scatter(x[ix],z[iZ], color='orange')      # pontos a z constante

        for iz in range(nz):
            if xintz[iz]>=xmin and xintz[iz]<=xmax:
                ix=np.round(xintz[iz]/dx)
                iPar=(iz)*nx+ix
                G[iObs,(iPar)]=1
#                plt.scatter(x[ix],z[iz], color='green')      # pontos a z constante
        iObs=iObs+1;
```

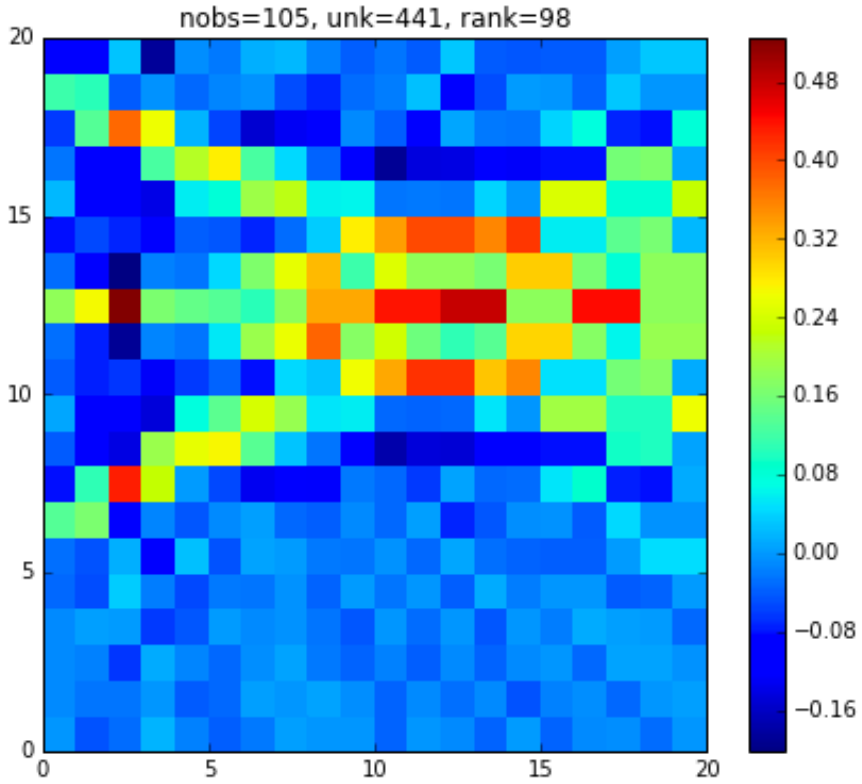
```
#%% plot matriz G
plt.close()
plt.rcParams['figure.figsize'] = 7,6

plt.pcolor(np.transpose(G))
plt.colorbar()
plt.xlabel(u'Observações (nr ângulos x nr estações)')
plt.ylabel(u'Incógnitas (nx x nz)')
plt.xlim([0,nobs])
plt.ylim([0,npar])

#%% Observações sintéticas

S=np.reshape(SS,[nx*nz,1])      # solução vectorizada
obs=np.dot(G,S)                   # observações vectorizadas (na forma de um vector 1D)

## solução

Xest=np.dot(pinv(G),obs)     # solução
Xmat=np.reshape(Xest,[nx,nz])

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

plt.pcolor(xx,zz,Xmat)
plt.colorbar()
rank=np.linalg.matrix_rank(G)
plt.title('nobs='+str(nobs)+', unk='+str(npar)+', rank='+str(rank))
```

Solução (nobs=105)

Notar que rank(A)=98<105, logo há observações redundantes.

O problema é subdeterminado...

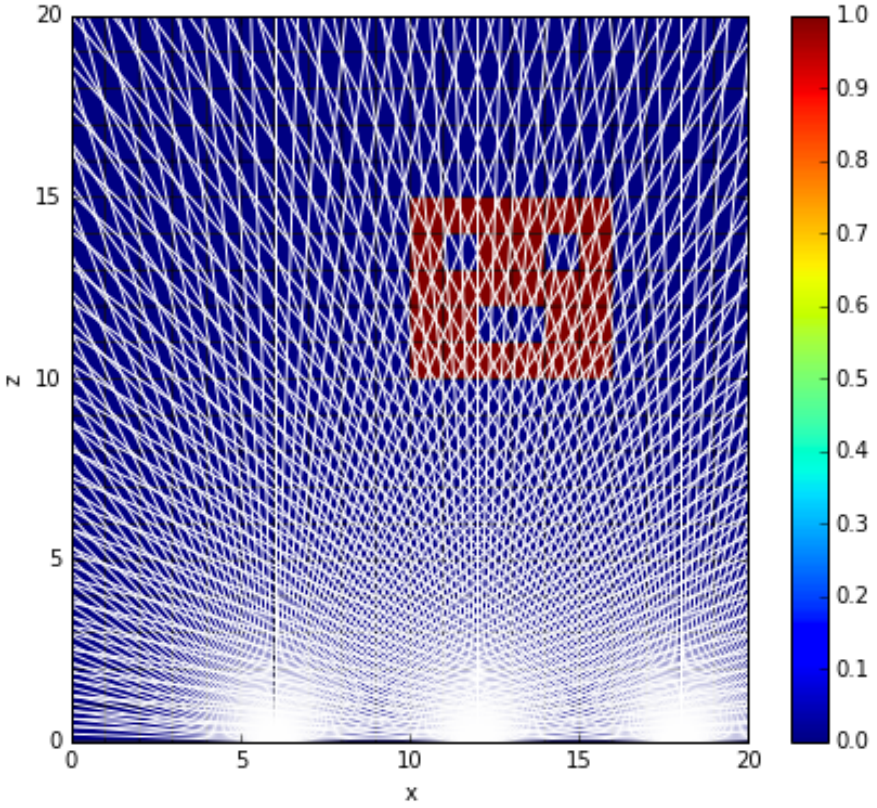


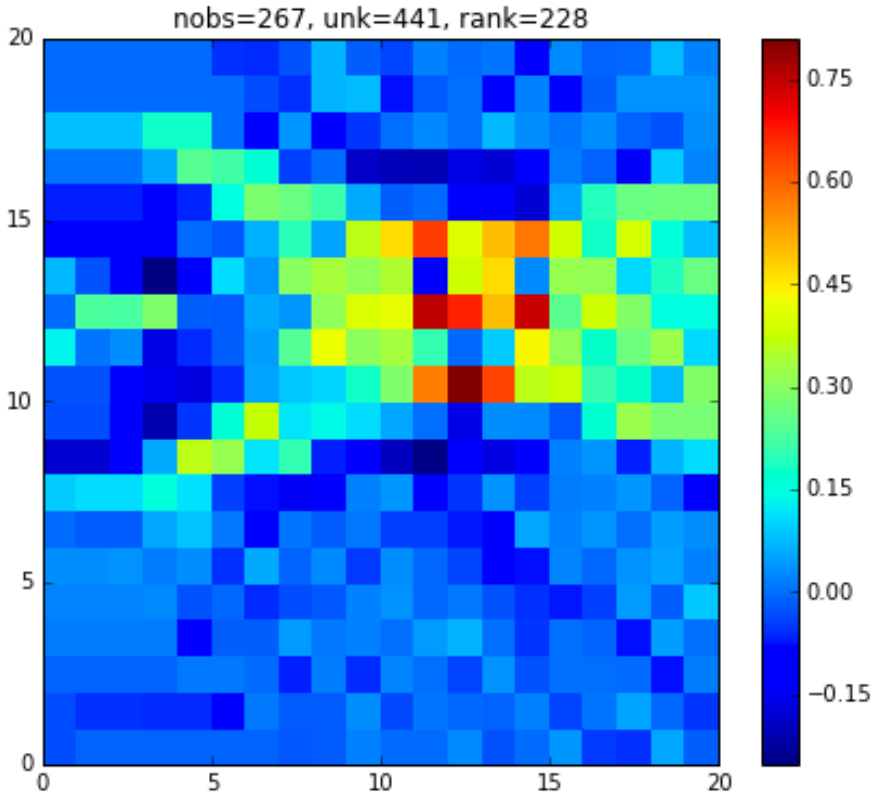


Solução (nobs=267)

Notar que rank(A)=228<267, logo há observações redundantes.

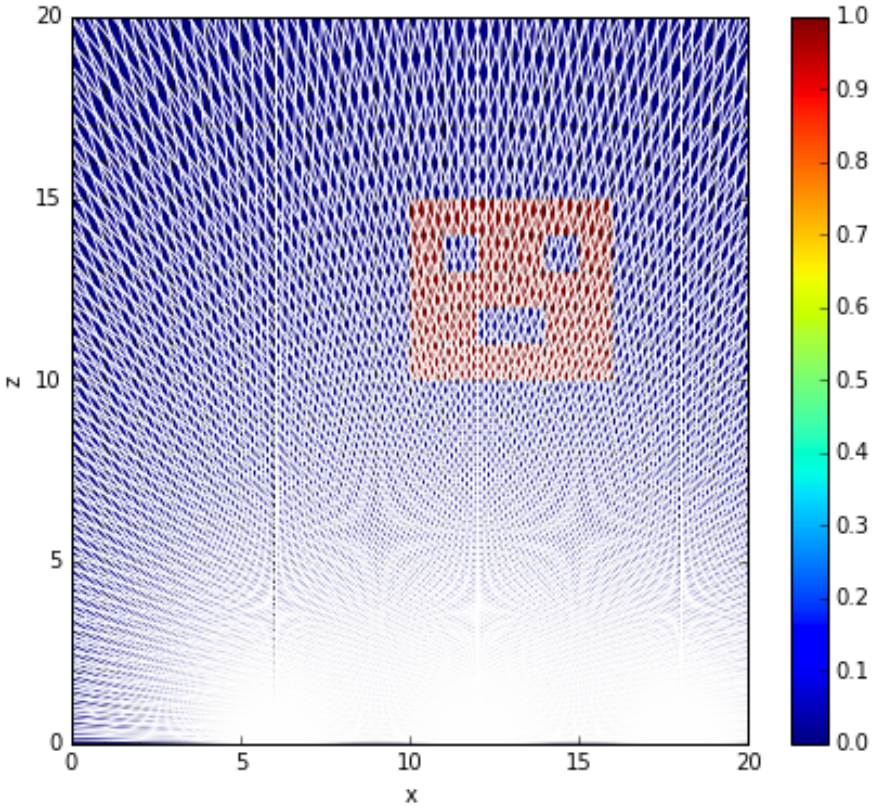
O problema é subdeterminado...

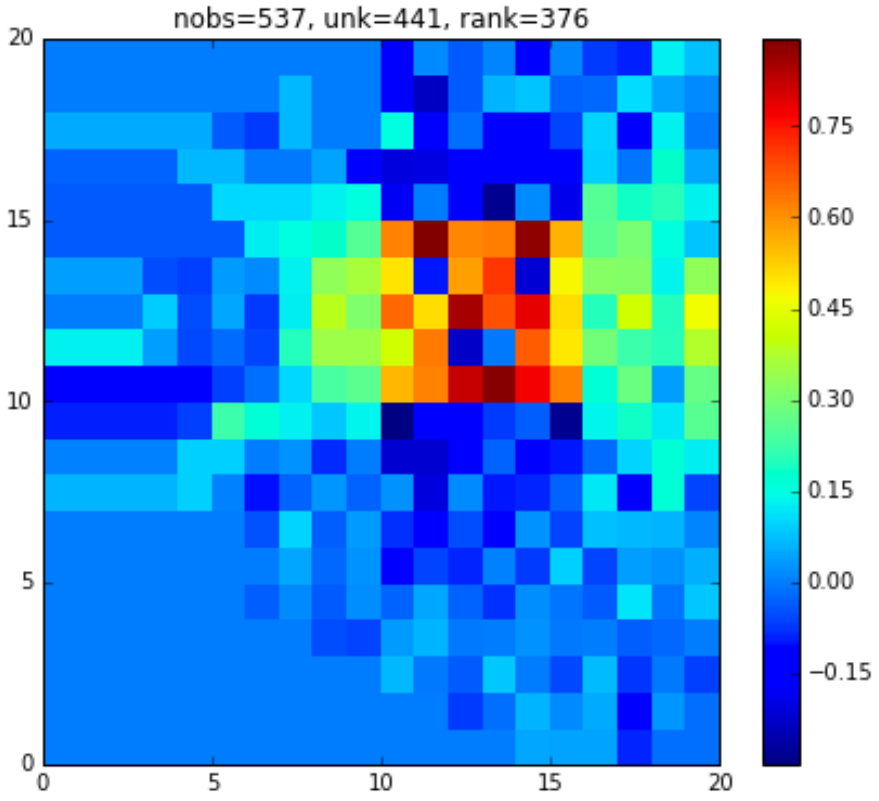




Solução (nobs=537)

Notar que rank(A)=376<537, logo há observações redundantes.

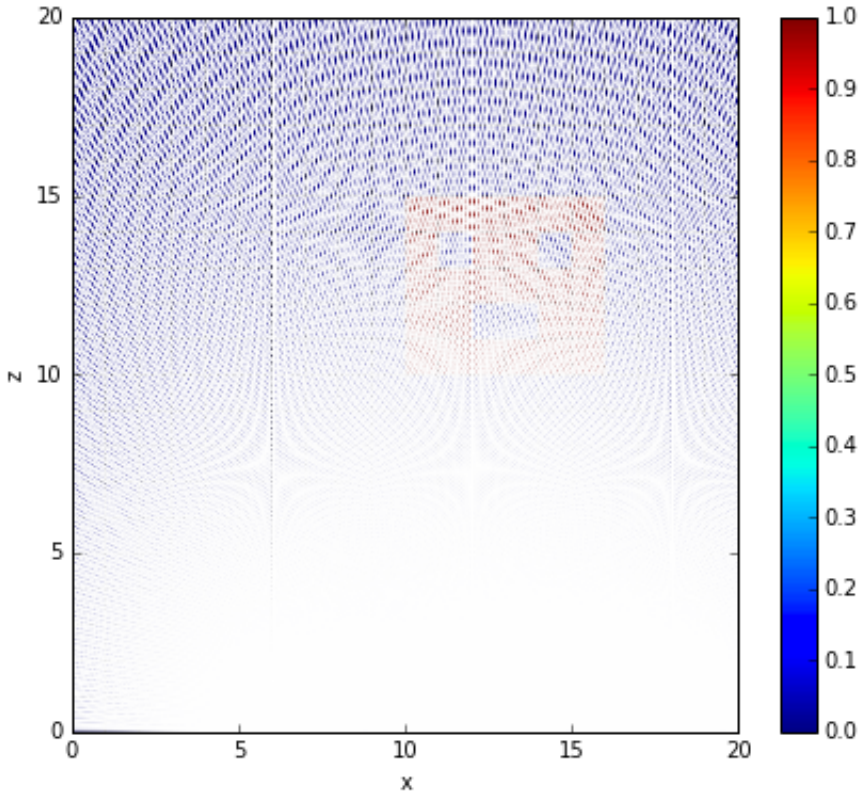


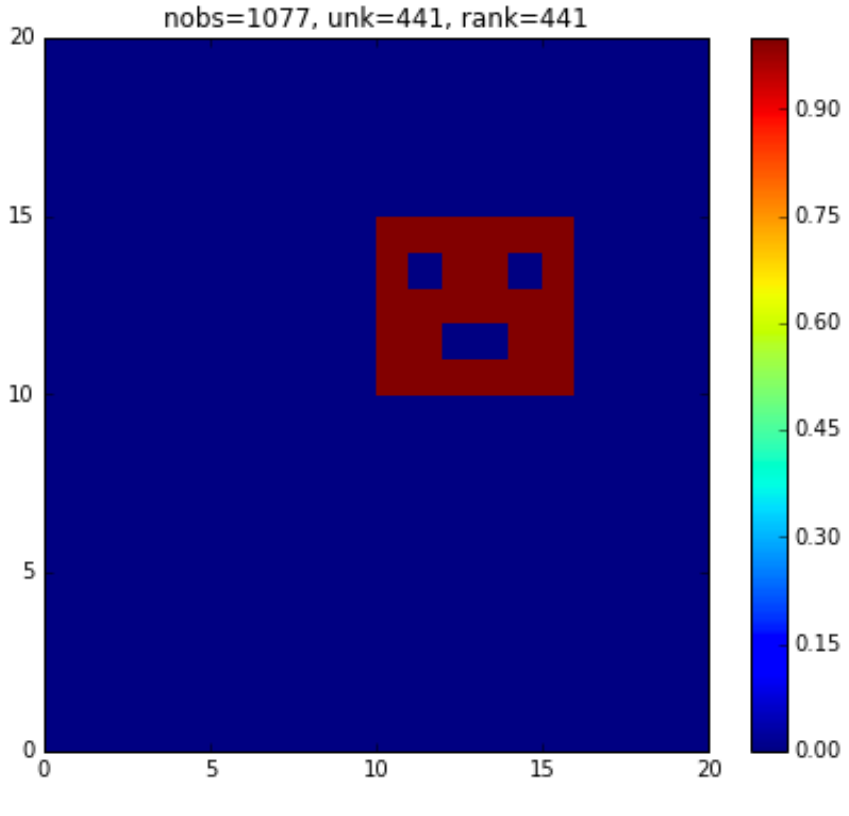


Solução (nobs=1077)

Notar que rank(A)=441.

Perfeito!





Comentários

- O problema resolve-se facilmente, mas a matriz do modelo dá trabalho a construir... O algoritmo funciona também para problemas que não sejam sub-determinados (mas é mais caro que o necessário, mas mais robusto).
- Optou-se por uma matriz de 0 e 1. Seria mais preciso calcular em cada pixel atravessado o percurso do raio através do pixel. Dava mais trabalho mas seria essencial para poder reconstituir objetos de "densidade" variável.
- Em problemas 3D a subdeterminação é frequentemente avassaladora. Só é possível obter uma solução iterativamente a partir de uma solução inicial (first guess) já próxima da solução procurada. O processo descrito seria aplicado em cada iteração para a correção a introduzir à solução.