



Ciências
ULisboa

Modelação Numérica

Aula 10

Problemas (lineares) com muitos graus de liberdade: tomografia

Problemas **direto** e **inverso**

Problema direto:

Permite calcular “observações” num sistema físico

- Por exemplo: (a) calcular a anomalia da gravidade a partir de uma distribuição de massa; (b) calcular tempos de chegada de um sismo a uma dada estação; (c) calcular trajetórias balísticas sabendo a condições iniciais; etc.

Problema inverso:

Faz o cálculo “inverso”: calcula os parâmetros que deram origem às observações

O problema **inverso**

É, em geral, mais complicado.

Pode estar mal condicionado, não ter solução única.

A solução não é normalmente exata: contém erro.

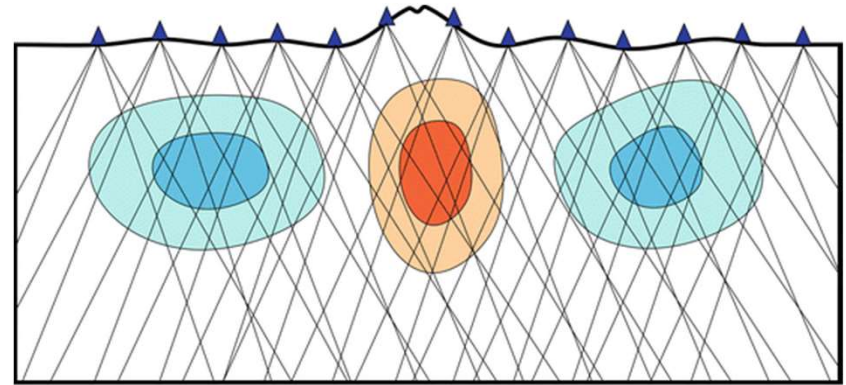
Os dois métodos descritos anteriormente...

Os métodos de **annealing** (modificado) e **genético** permitem resolver (iterativamente) o problema inverso de forma muito geral, a partir da localização de um mínimo de uma função de custo multidimensional.

O espaço multidimensional explorado é o **espaço dos parâmetros**.

A complexidade do problema cresce rapidamente com o número de dimensões ($\propto N!$). Isso limita esses métodos a problemas com poucos parâmetros independentes.

O problema da tomografia



A tomografia permite **mapear estruturas inacessíveis** (subterrâneas ou dentro do corpo) a partir da “inversão” de observações em pontos remotos (som, ondas sísmicas, ondas eletromagnéticas, etc...).

Neste caso, o **número de parâmetros a calcular é muito grande**: tipicamente a distribuição espacial (numa grelha de pontos) de certas propriedades em todo o domínio.

Felizmente, em certos casos, o problema pode ser tratado com uma **aproximação linear**.

Modelo linear

Os problemas lineares podem escrever-se sempre na forma:

$$A\vec{x} = \vec{y}$$

Onde $\vec{x} = \{x_1 \dots x_N\}$ é o vetor de **parâmetros** a estimar, $\vec{y} = \{y_1 \dots y_M\}$ é o vetor de **observações**, e $A = \begin{bmatrix} A_{11} & \dots & A_{1N} \\ \vdots & \ddots & \vdots \\ A_{M1} & \dots & A_{MN} \end{bmatrix}$ é uma matriz de constantes que constitui o modelo linear.

Se $M = N$ e a matriz A não for singular (i.e. $\det(A) \neq 0$) a solução é trivial:

$$\vec{x} = A^{-1}\vec{y}$$

e consiste no cálculo na **matriz inversa**.

(Em python poderia ser `X=np.matmul(np.linalg.inv(A), Y)`)

Caso $M \neq N$

Se $M > N$ (mais observações que incógnitas) o problema é **sobredeterminado**, não existe solução exata, e pode ser resolvido por **mínimos quadrados**, o que consiste em encontrar a solução com o menor erro médio quadrático.

- Exemplo: regressão linear

Se $M < N$ (menos observações que incógnitas) o problema é **subdeterminado**, e existem **múltiplas soluções exatas compatíveis com as observações** e com erro nulo.

Problema subdeterminado $M < N$

Se $M < N$ (menos observações que incógnitas) o problema é **subdeterminado**, e existem **múltiplas soluções exatas compatíveis com as observações** e com erro nulo.

Sendo $A(M, N)$ É possível determinar **univocamente** uma matriz $B(N, M)$, tal que

$$\vec{x} = B\vec{y}$$

corresponde à solução com **menor norma L2** de entre todas as soluções disponíveis.

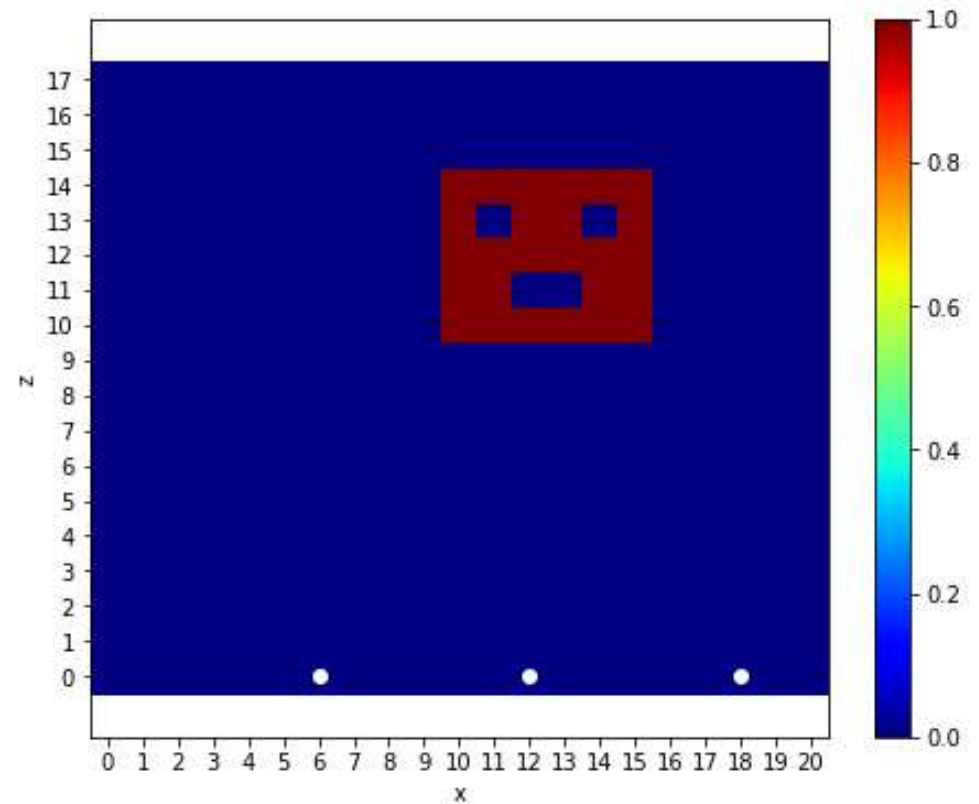
Em **numpy** esse cálculo faz-se de forma imediata:

$$B = np.linalg.pinv(A);$$

(Moore-Penrose pseudo-inverse)

Tomografia 2D

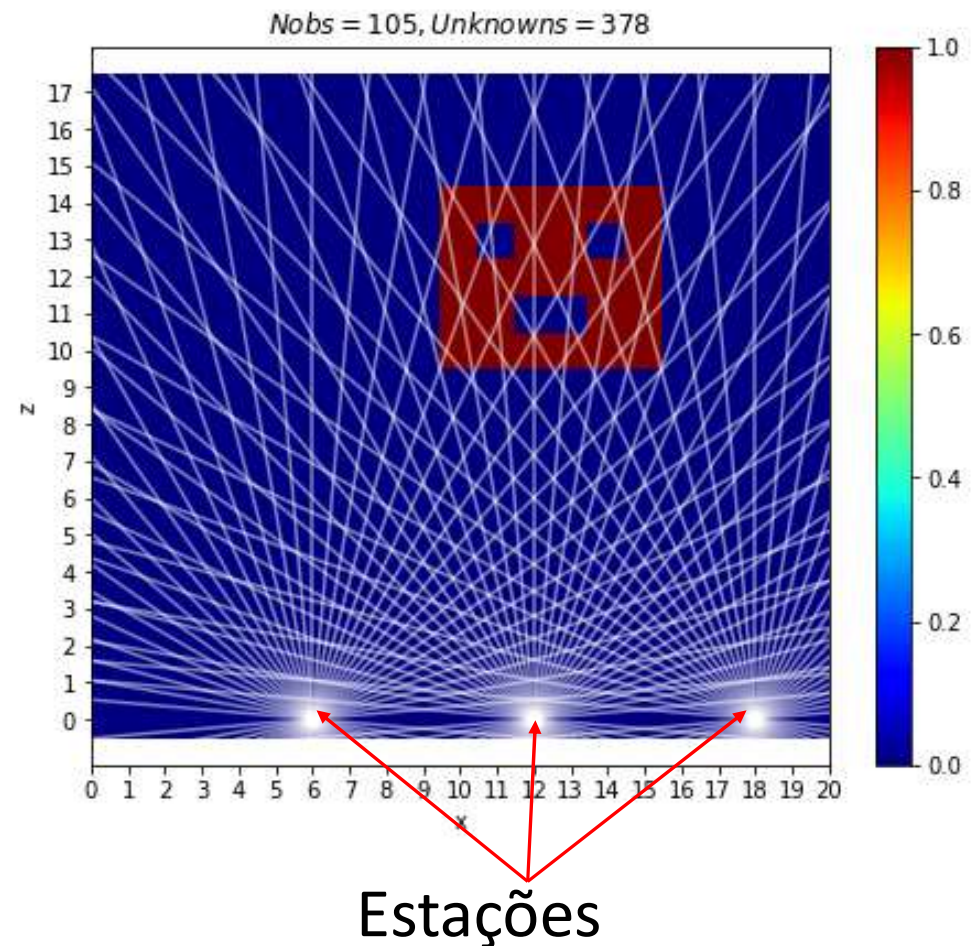
Mapear o domínio (21×18) a partir de observações em 3 pontos na fronteira.



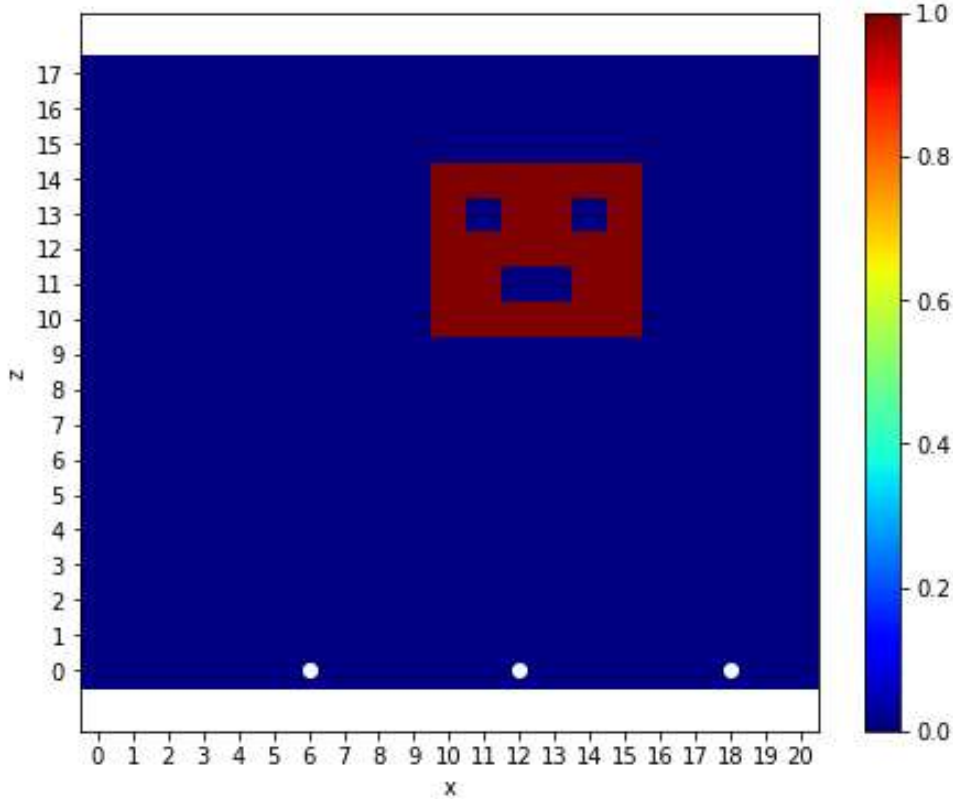
Tomografia 2D

O objeto a mapear é descontínuo (o campo vale 1 no seu interior e 0 no resto do domínio)

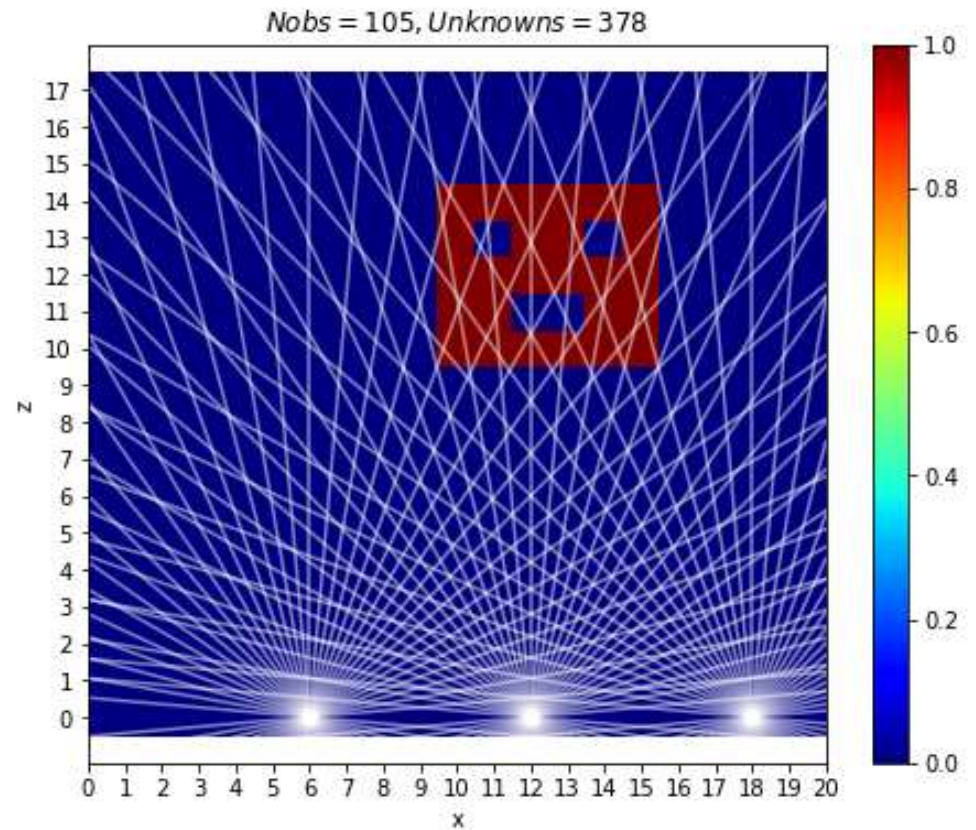
Vamos admitir que o sinal é atenuado mas se propaga em linha reta (**não há refração**).



Domínio a mapear: 378 valores (0,1)



Observações: 3 sensores, cada sensor vê o integral ao longo do percurso, em vários ângulos: 105 valores (3×35)



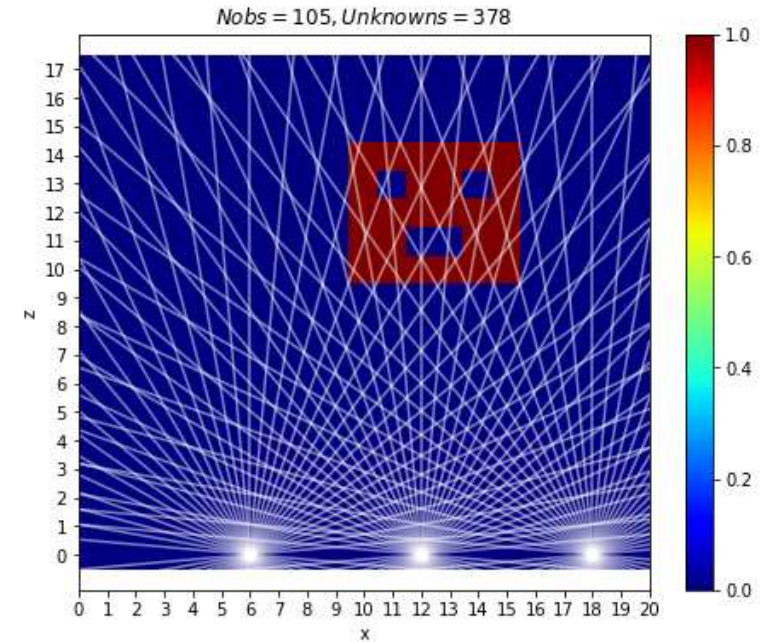
Preliminares

```
import numpy as np;import matplotlib.pyplot as plt
nx=21;nz=18;dx=1;dz=1
x=np.arange(0,nx*dx,dx);z=np.arange(0,nz*dz,dz)
xmin=np.min(x);xmax=np.max(x);zmin=np.min(z);zmax=np.max(z)
xis=np.zeros((nx,nz));zed=np.zeros((nx,nz))
for ix in range(nx):
    zed[ix,:]=z
for iz in range(nz):
    xis[:,iz]=x
CLOUD=np.zeros((nx,nz)) #campo a mapear
for ix in range(10,16):
    for iz in range(10,15):
        CLOUD[ix,iz]=1
CLOUD[11,13]=0;CLOUD[14,13]=0;CLOUD[12:14,11]=0
plt.figure(figsize=(7.5,6))
mm=plt.pcolor(xis,zed,CLOUD,cmap='jet',shading='auto');plt.colorbar(mm)
plt.xlabel('x');plt.ylabel('z');plt.xticks(x);plt.yticks(z)
xE=np.array([6,12,18]);zE=np.zeros(xE.shape) #pontos de observação
plt.scatter(xE,zE,color='white');plt.axis('equal')
```

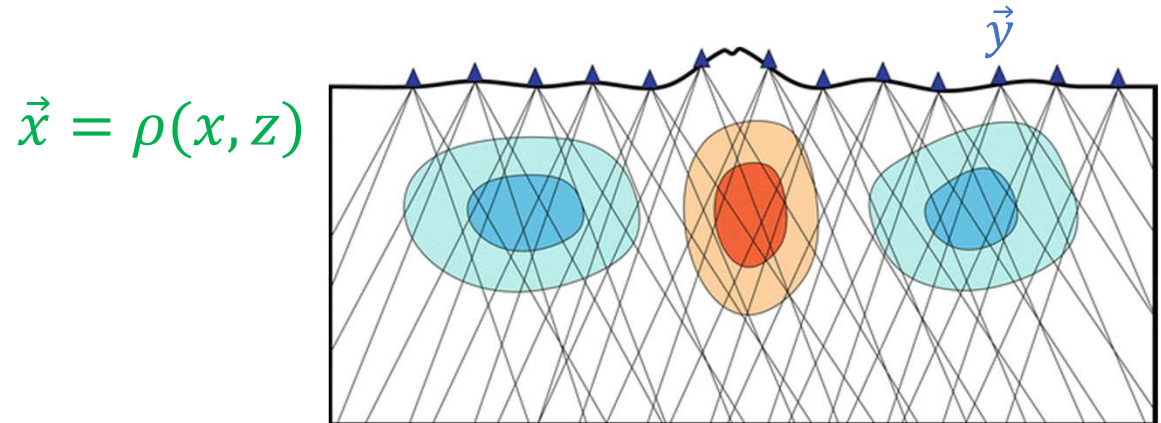
Geração das observações

```
angE=np.arange(175,0,-5) #geometria da observação
slopeE=np.tan(angE*np.pi/180)
nEx=len(xE);nEa=len(angE)
for iE in range(nEx):
    for iang in range(nEa):
        plt.plot([xE[iE],xmax],[zE[iE],zE[iE]+slopeE[iang]*\
                (xmax-xE[iE])],color='white',alpha=0.5)
        plt.plot([xE[iE],xmin],[zE[iE],zE[iE]+slopeE[iang]*\
                (xmin-xE[iE])],color='white',alpha=0.5)
plt.xlim(xmin,xmax)
plt.ylim(zmin,zmax)
plt.scatter(xE,zE,color='yellow',zorder=2)

plt.title(r'$Nobs=%3i,Unknowns=%3i$' % (nEa*nEx,nx*nz))
```



Tomografia



Mapear estruturas inacessíveis a partir da “inversão” de observações em pontos remotos.

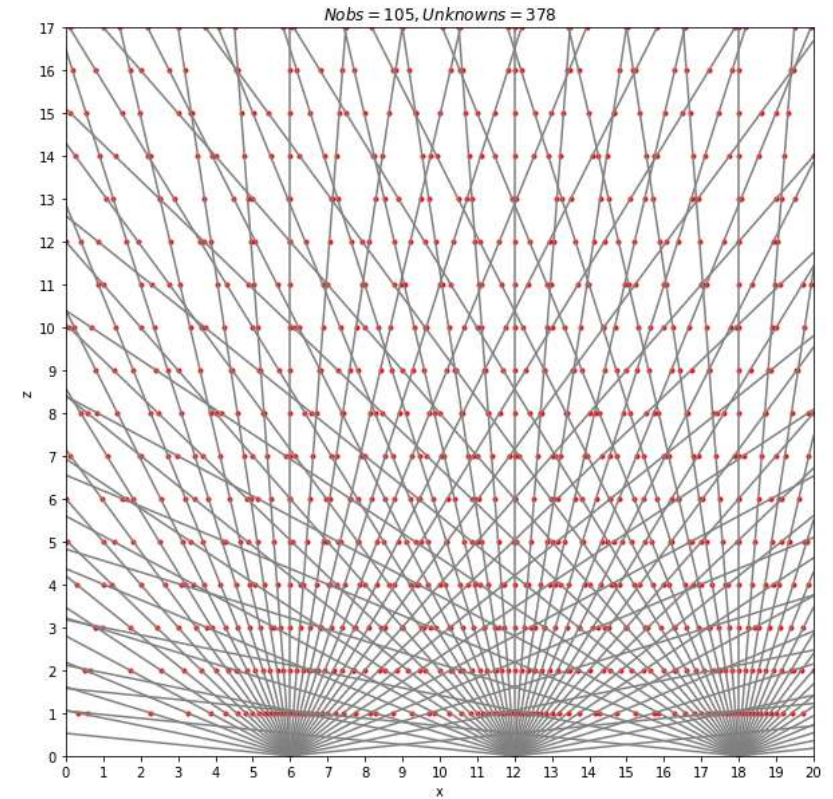
O **número de parâmetros a calcular é muito grande**, mas se so existir atenuação do sinal (mas propagação retilínea, i.e. sem refração) o problema pode ser tratado com uma **aproximação linear**:

$$\vec{y} = A\vec{x}$$

\vec{y} : observações; \vec{x} : parâmetros; A : matriz de coeficientes

Cálculo da matriz A (voxel a voxel)

```
A=np.zeros((nEx*nEa,nx*nz)) #matriz direta
Y=np.zeros((nEx*nEa)) #vetor de observações
zint=np.zeros((nx));xintx=np.zeros((nx))
zintz=np.zeros((nz));xint=np.zeros((nz))
iY=-1
for iEx in range(nEx): #sensores
    for iEa in range(nEa): #ângulos
        xx=xE[iEx];slope=slopeE[iEa] #observação
        for ix in range(nx):
            zint[ix]=zE[iEa]+slope*(x[ix]-xx) #interseta pixel x
            xintx[ix]=x[ix]
        for iz in range(nz):
            zintz[iz]=z[iz]
            xint[iz]=xx+(z[iz]-zE[iEa])/slope #interseta pixel x
            plt.scatter(xint[iz],zintz[iz],color='yellow',marker='.')
        iY=iY+1
    for ix in range(nx):
        if zint[ix]>=zmin and zint[ix]<=zmax:
            iZ=np rint(zint[ix]/dz).astype(int)
            iX=(iZ-1)*nx+ix
            A[iY,iX]=1
    for iz in range(nz):
        if xint[iz]>=xmin and xint[iz]<=xmax:
            ix=np rint(xintx[iz]/dx).astype(int)
            iX=(iz-1)*nx+ix
            A[iY,iX]=1
```



A matriz A só contém '0' e '1'

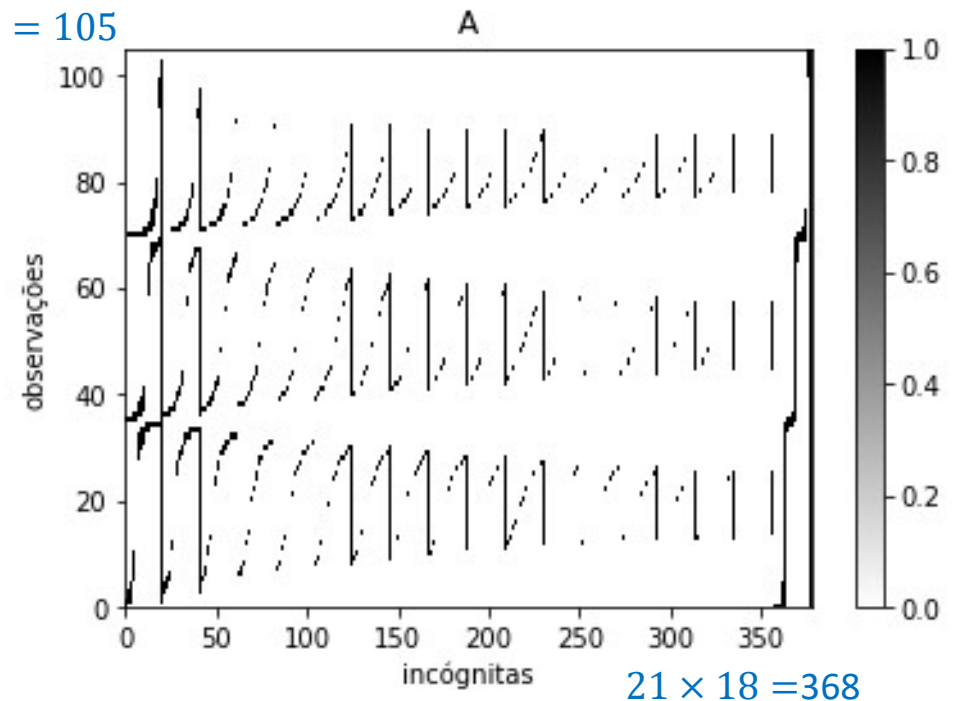
A matriz A só depende da geometria do problema

Cálculo da matriz A

```
#A[i,j]=0: obs i não inclui do pixel j  
#A[I,j]=1: obs i inclui o pixel j
```

```
plt.xticks(x)  
plt.yticks(z)  
plt.figure()  
m2=plt.pcolor(A,cmap='grey_r')  
plt.colorbar(m2)  
plt.xlabel('nx*nz (%3i,%3i)' % (nx,nz))  
plt.ylabel('nEx*nEa (%3i,%3i)' % (nEx,nEa))  
plt.title('A')  
plt.ylabel('observações');plt.xlabel('incógnitas')
```

$$3 \times 35 = 105$$

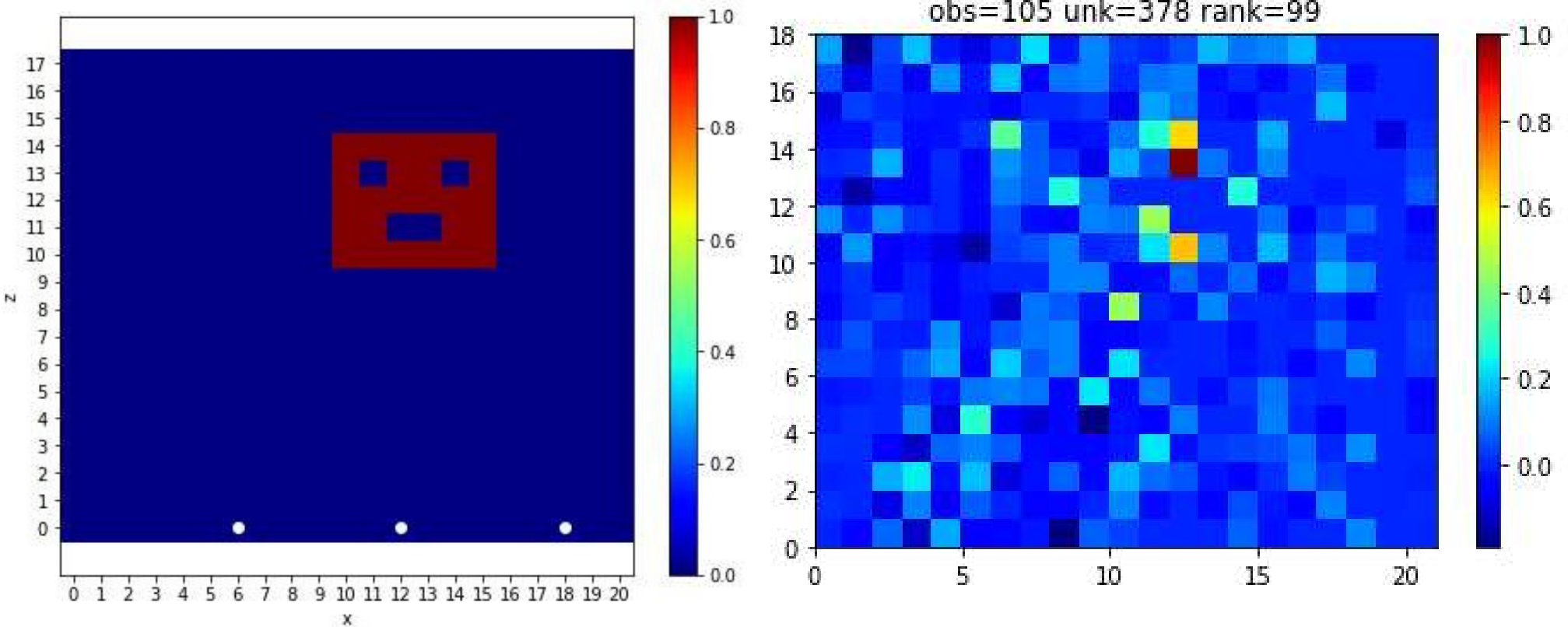


A matriz A só contém '0' e '1'

Solução

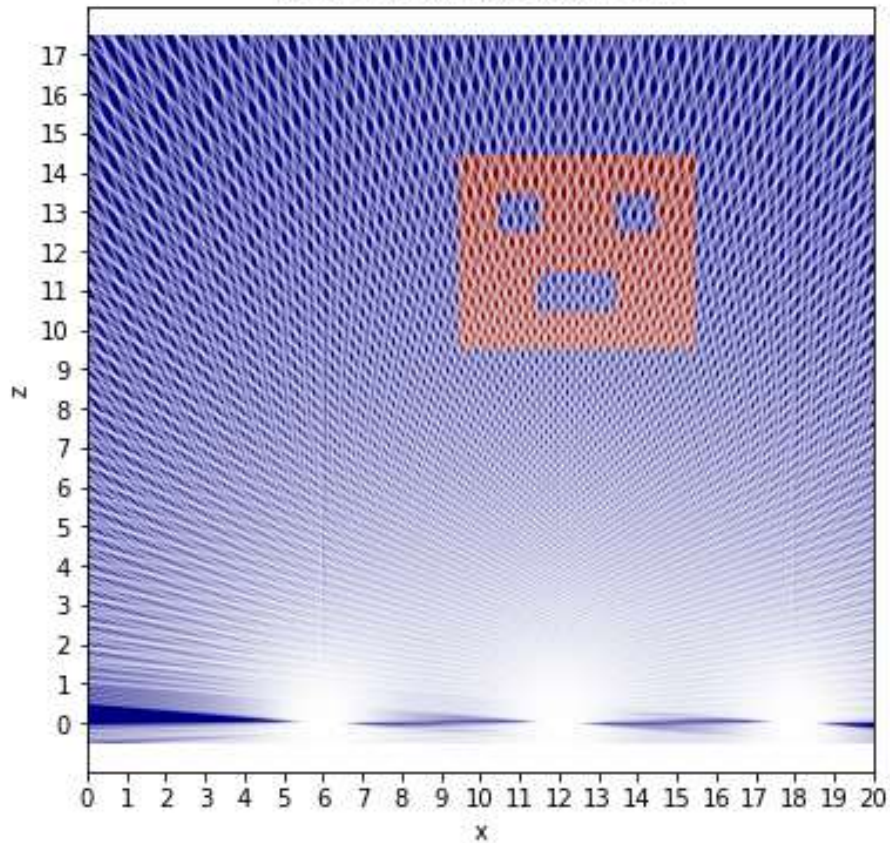
```
Y=np.matmul(A,np.reshape(CLOUD,(nx*nz))) #observações vetorizadas: problema direto
Yy=np.reshape(Y,(nEa,nEx)).transpose() #observações em matriz
xxY=np.empty(Yy.shape);xxY[:]=np.nan
aaY=np.copy(xxY)
for iang in range(nEa):
    for iE in range(nEx):
        aaY[iE,iang]=angE[iang]
for iEx in range(nEx):
    for iang in range(nEa):
        xxY[iEx,iang]=xE[iEx]
pseudoinv=np.linalg.pinv(A) #inversão
Xest=np.matmul(pseudoinv,Y) #solução (em vector)
Xmat=np.reshape(Xest,(nx,nz)) #solução no espaço físico (em matriz)
plt.figure()
m4=plt.pcolor(xis,zed,Xmat,cmap='jet')
plt.colorbar(m4)
plt.title('obs='+str(nEx*nEa)+' unk='+str(nx*nz)\
          +' rank='+str(np.linalg.matrix_rank(A)))
```

Problema fortemente subdeterminado amostra cada 5°



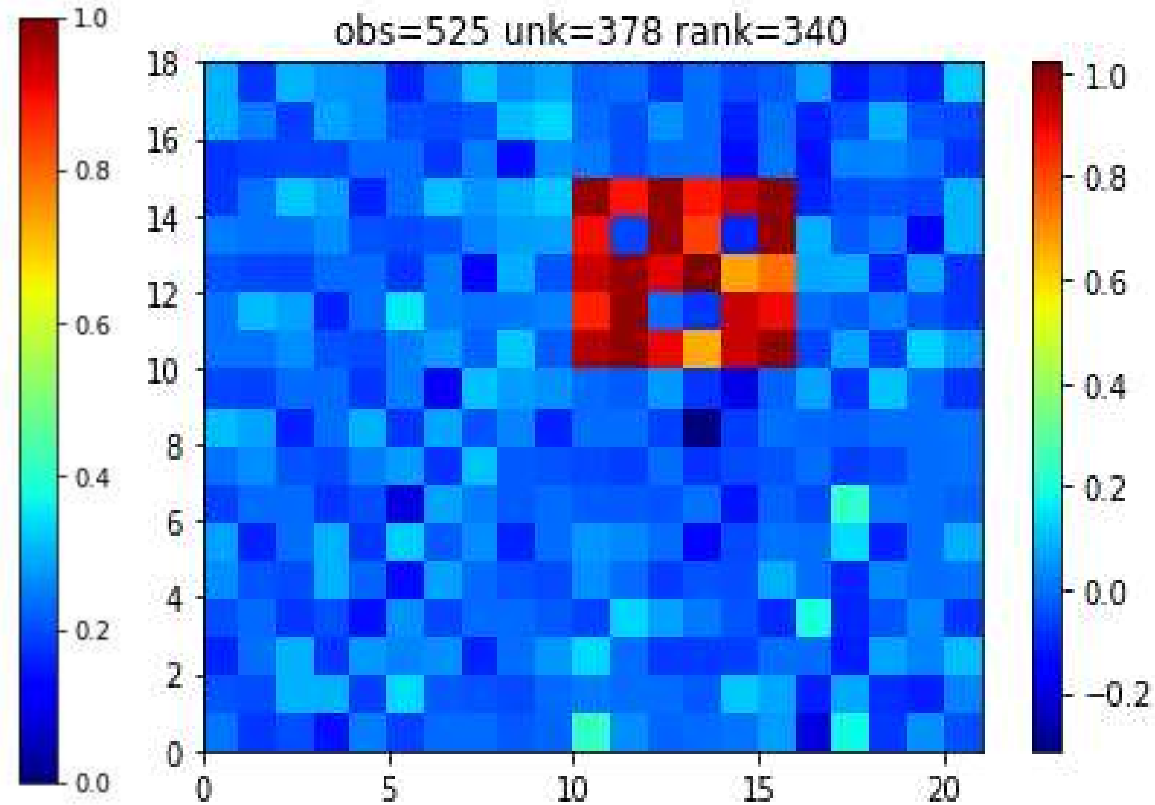
Aumentando o nº de observações 1° (notar que existe redundância: $obs > rank$)

Nobs = 525, Unknowns = 378



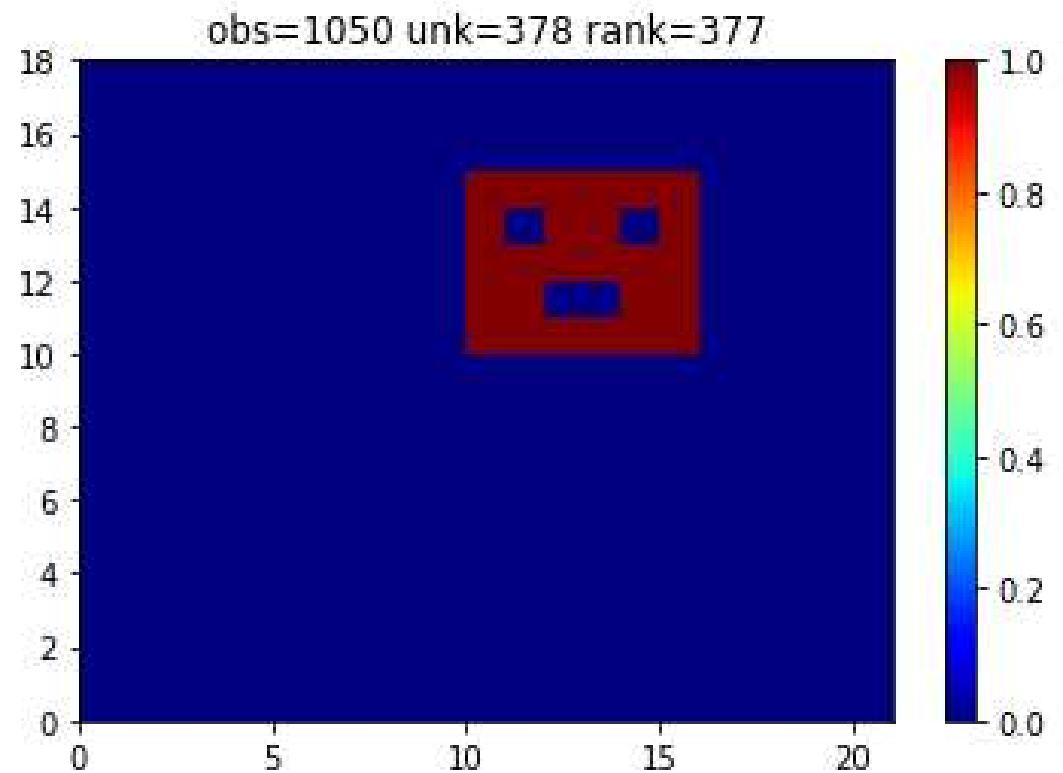
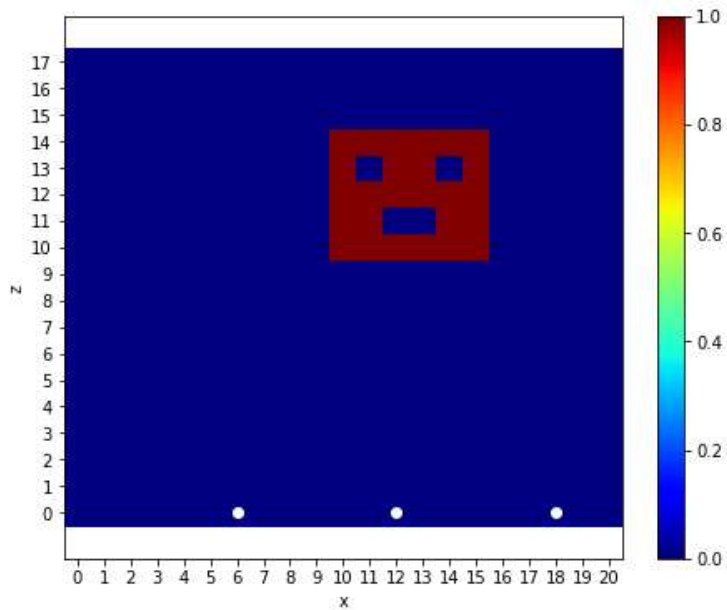
1°

obs=525 unk=378 rank=340



Observações cada 0.5° (3×350)

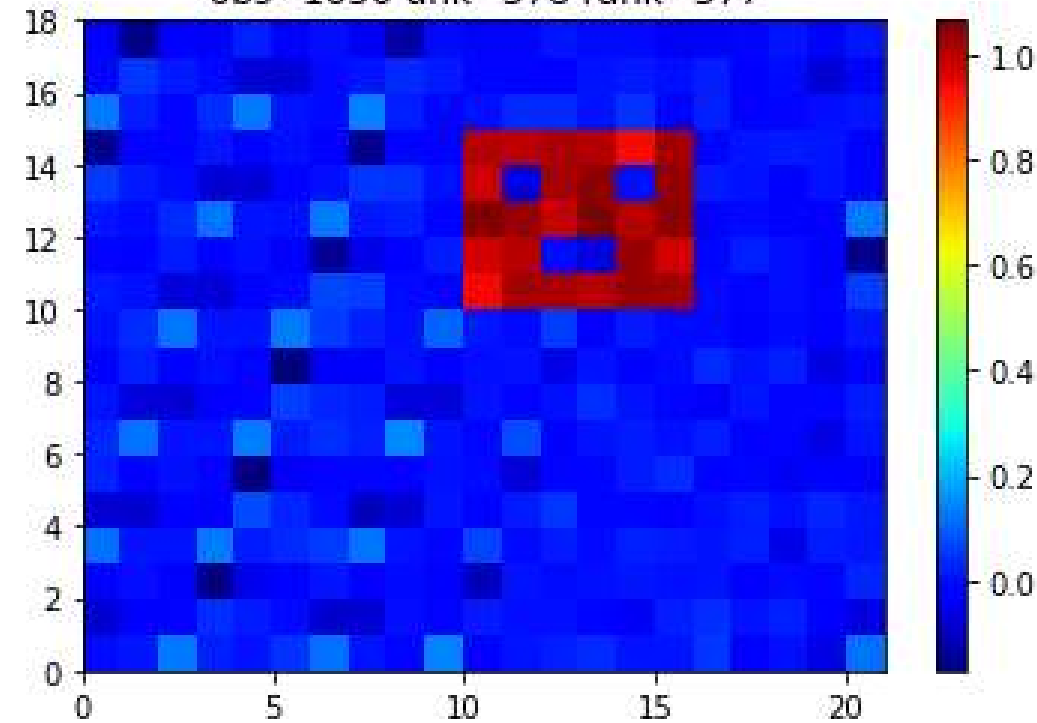
rank: maior submatriz
não singular



$$Y = Y * (1 + (\text{np.random.random}(Y.\text{shape}) * 0.5) * 0.05)$$

5% de ruído

obs=1050 unk=378 rank=377



0% de ruído

obs=1050 unk=378 rank=377

