Ciências ULisboa

Modelação Numérica 2022
Aula 19

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.

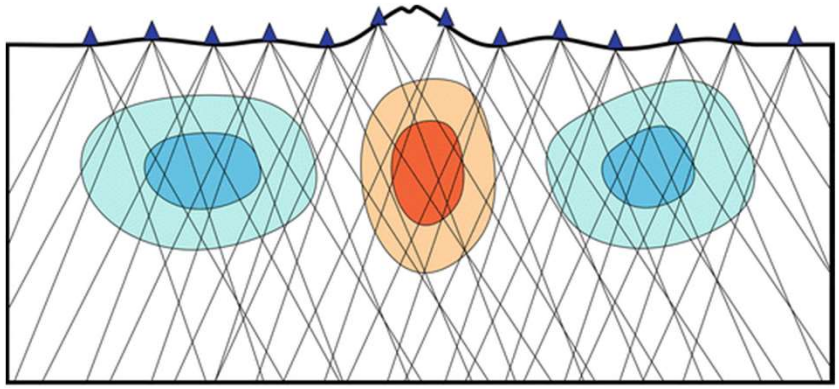
Os dois métodos descritos anteriormente...

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 especial (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 $\mathrm{A} = \begin{bmatrix} A_{11} & \cdots & A_{1N} \\ \vdots & \ddots & \vdots \\ A_{M1} & \cdots & A_{MN} \end{bmatrix}$ 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.

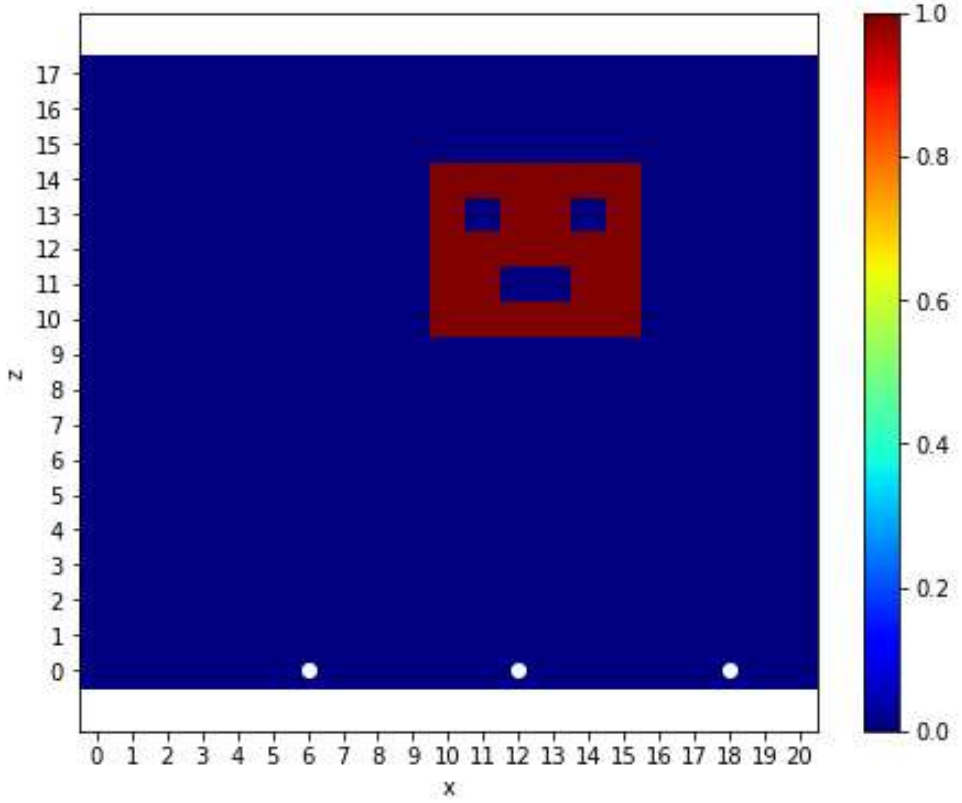
Em **numpy** esse cálculo faz-se de forma imediata:

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

(Moore-Penrose pseudo-inverse)

Tomografia 2D

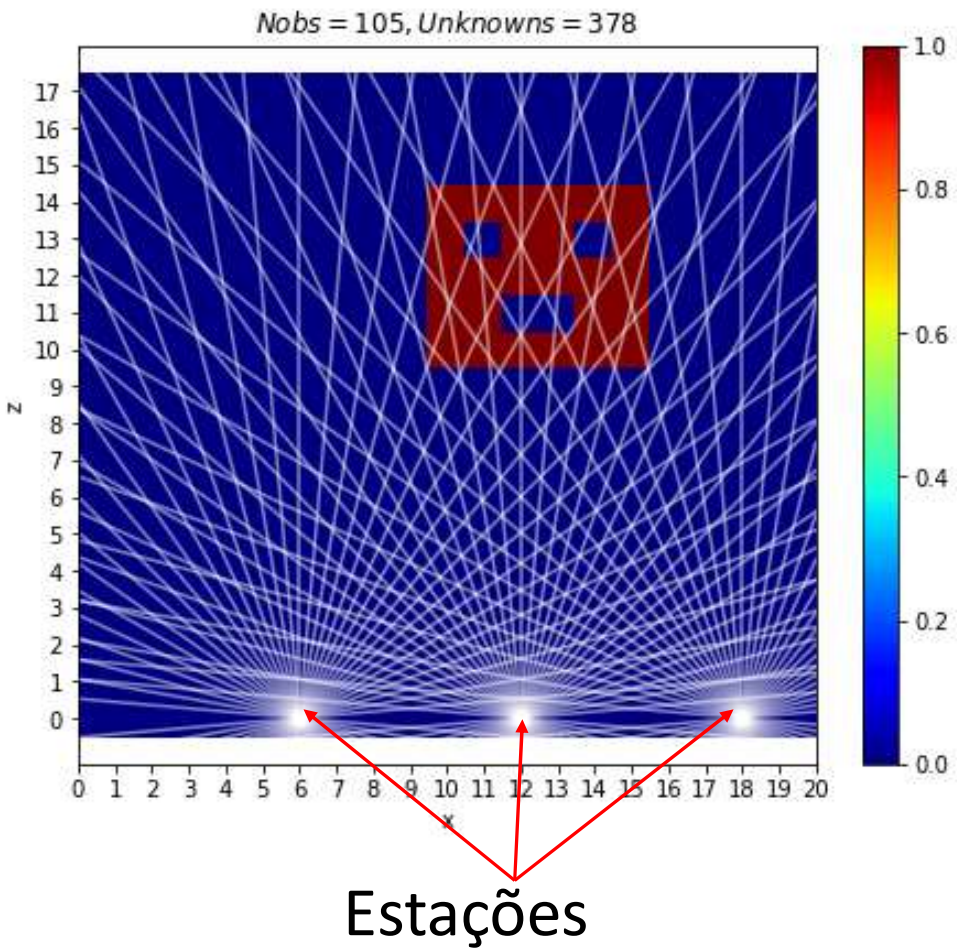
Mapear o domínio (21×18) a partir de observações em 3 pontos na fronteira.



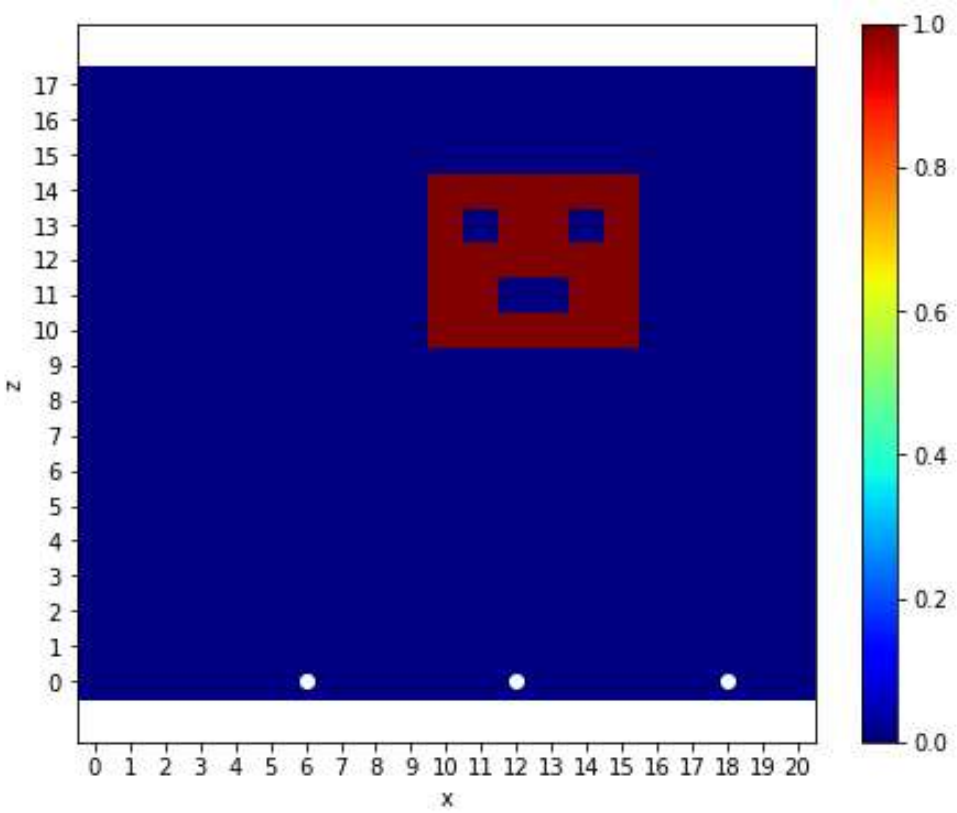
Tomografia 2D

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

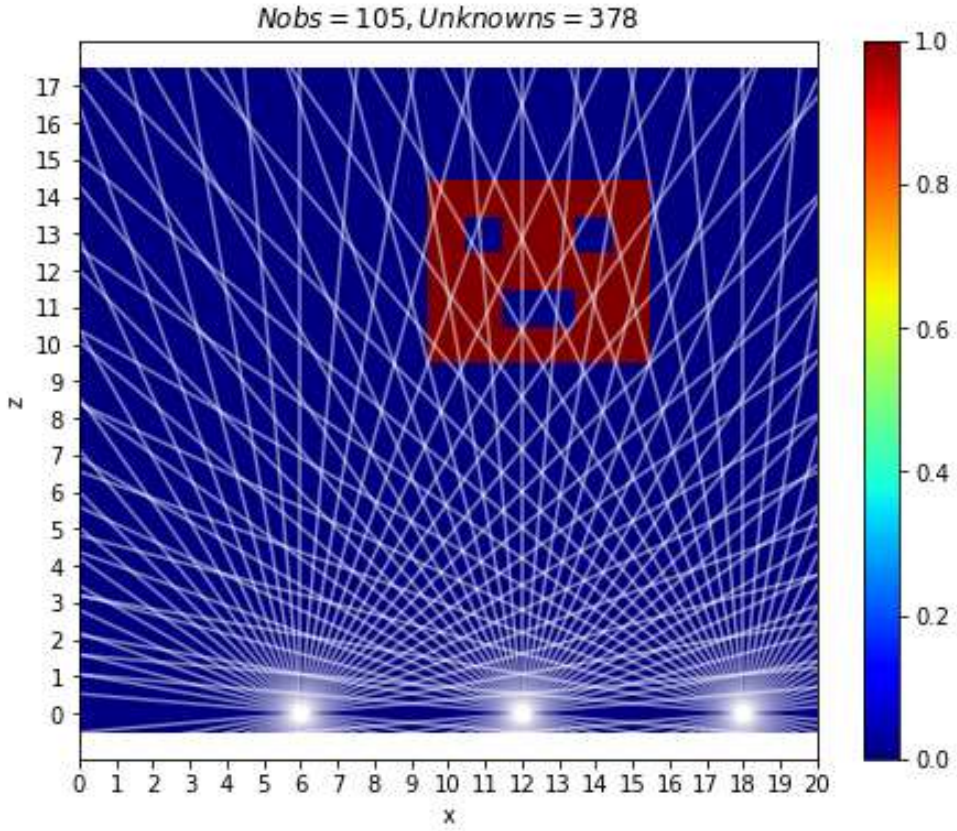
Vamos admitir que o sinal é atenuado mas se propaga em linha reta (não há refração).



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)



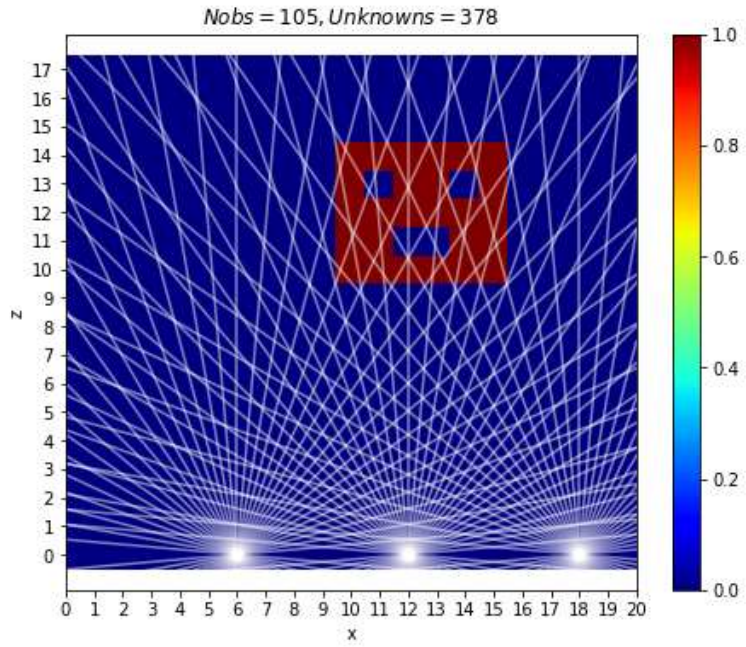
Preliminares

```
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

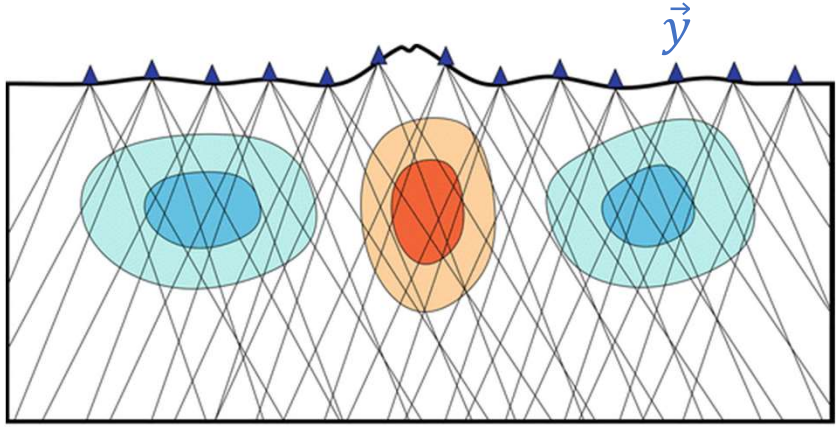
```
angE=np.arange(175,0,-5) #geometria da observação
slopeE=np.tan(angE*np.pi/180)
nEx=len(xE);nEa=len(angE)
for iE in range(nEx):
    for iang in range(nEa):
        plt.plot([xE[iE],xmax],[zE[iE],zE[iE]+slopeE[iang]*\
                 (xmax-xE[iE])],color='white',alpha=0.5)
        plt.plot([xE[iE],xmin],[zE[iE],zE[iE]+slopeE[iang]*\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

$\vec{x} = \rho(x, z)$



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

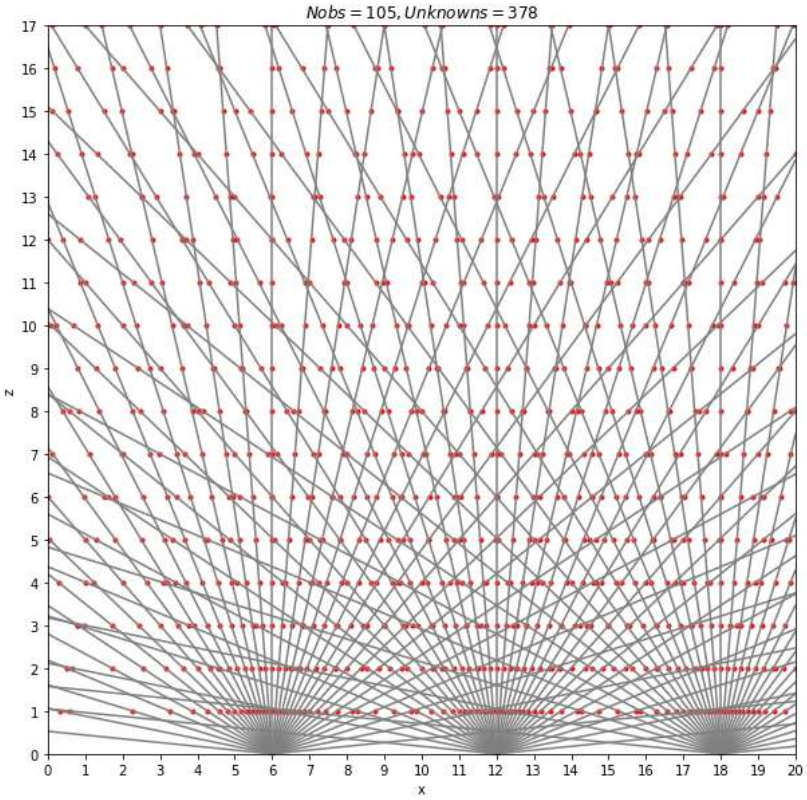
O número de parâmetros a calcular é muito grande, mas se so existir atenuação do sinal (mas propagação retilíne, i.e. sem refração) o problema pode ser tratado com uma **aproximação linear:**

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

$\vec{y}$: observações; $\vec{x}$: parâmetros; A: matriz de coeficientesType equation here.

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[iEx]+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[iEx])/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[ix]/dx).astype(int)
                iX=(iz-1)*nx+ix
                A[iY,iX]=1
```



A matriz A só contém ‘0’ e ‘1’

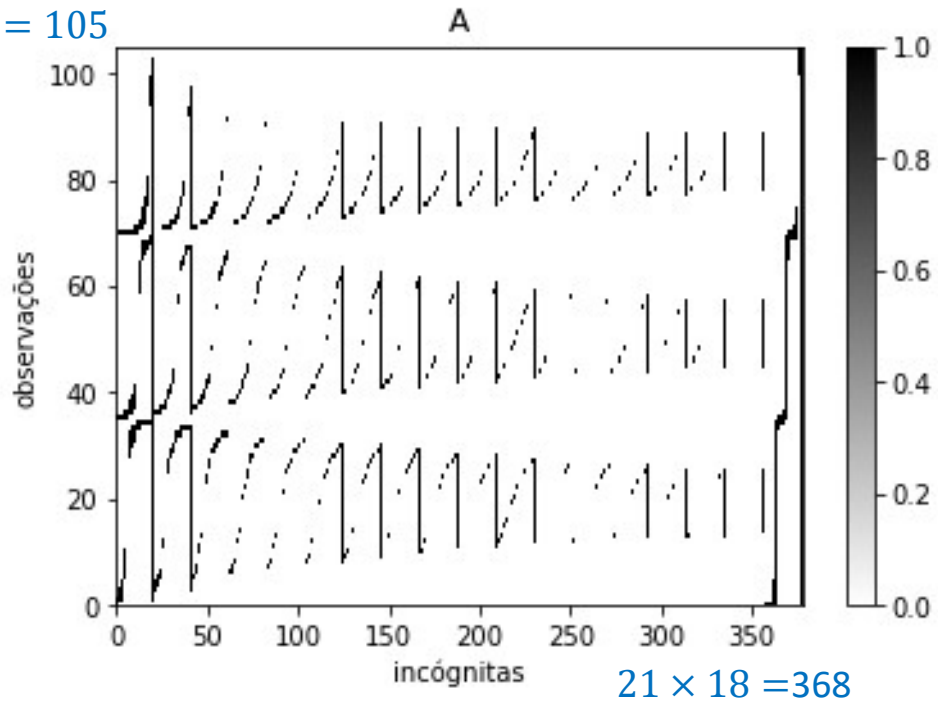
A matriz A só depende da geometria do problema

Cálculo da matriz A

$3 \times 35 = 105$

```
#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')
```



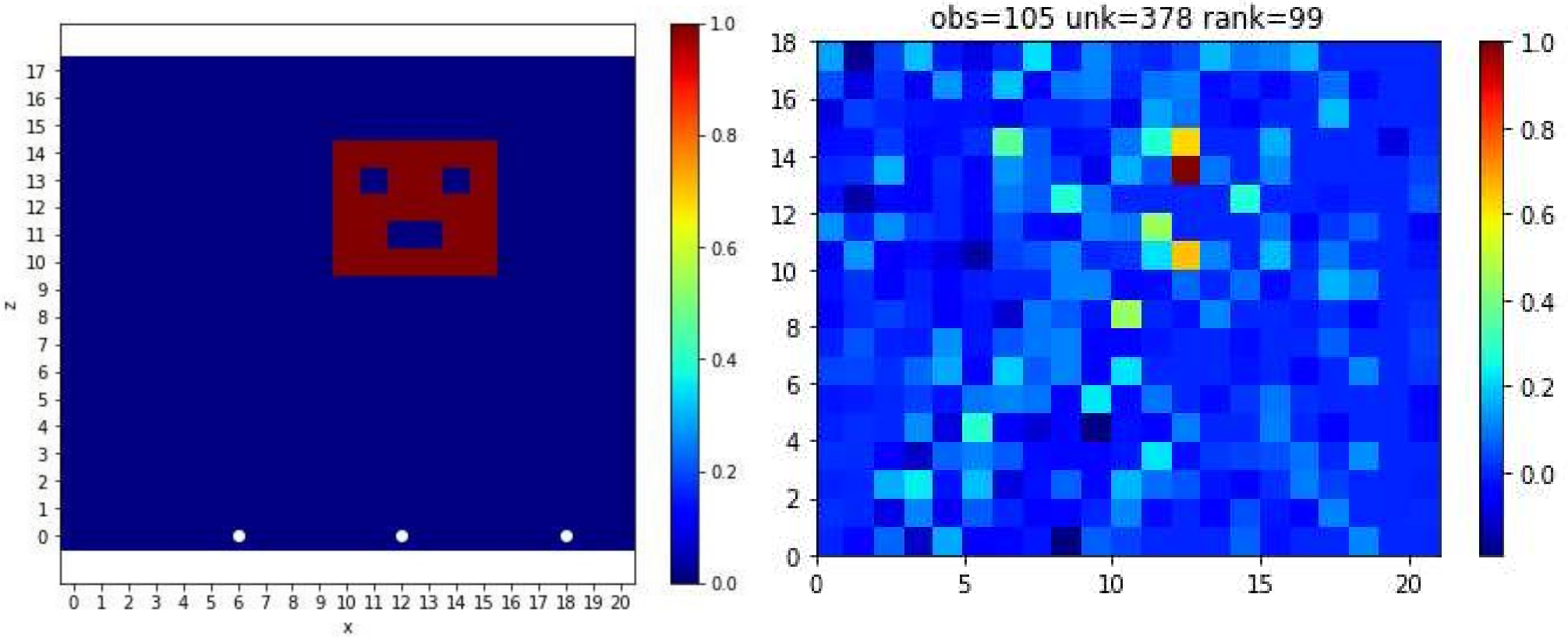
$21 \times 18 = 368$

A matriz A só contém '0' e '1'

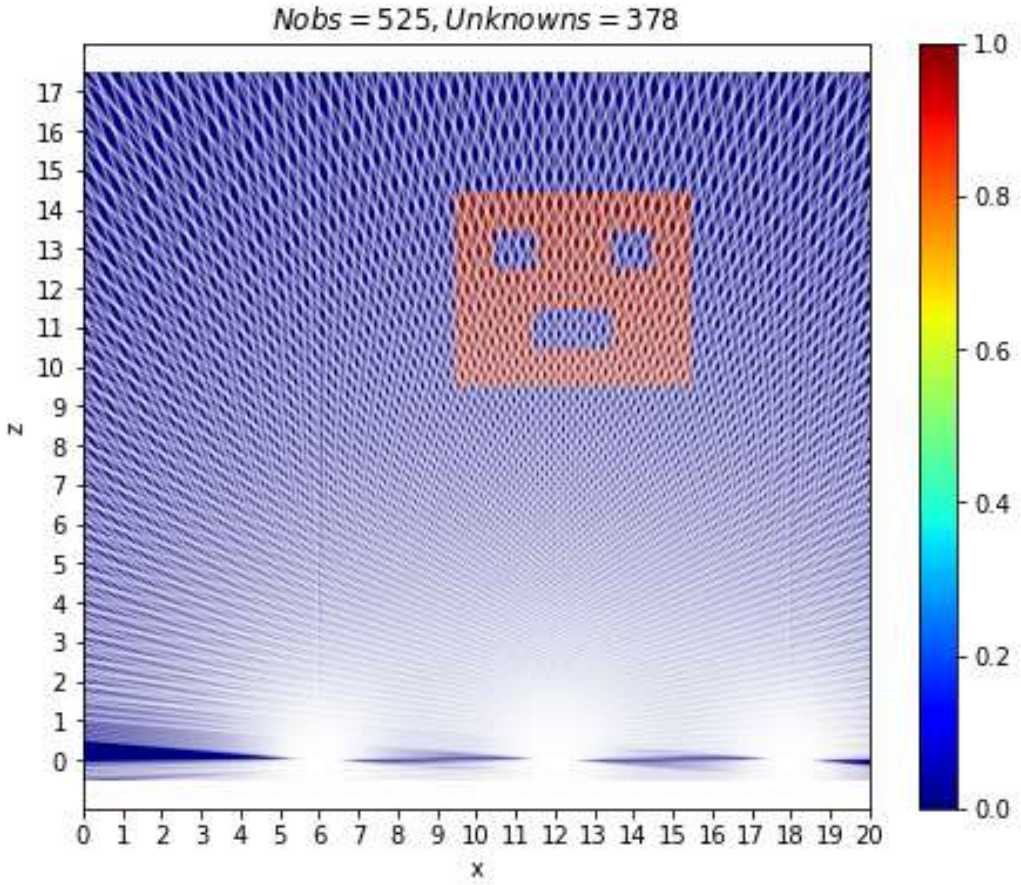
Solução

```
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)))
```

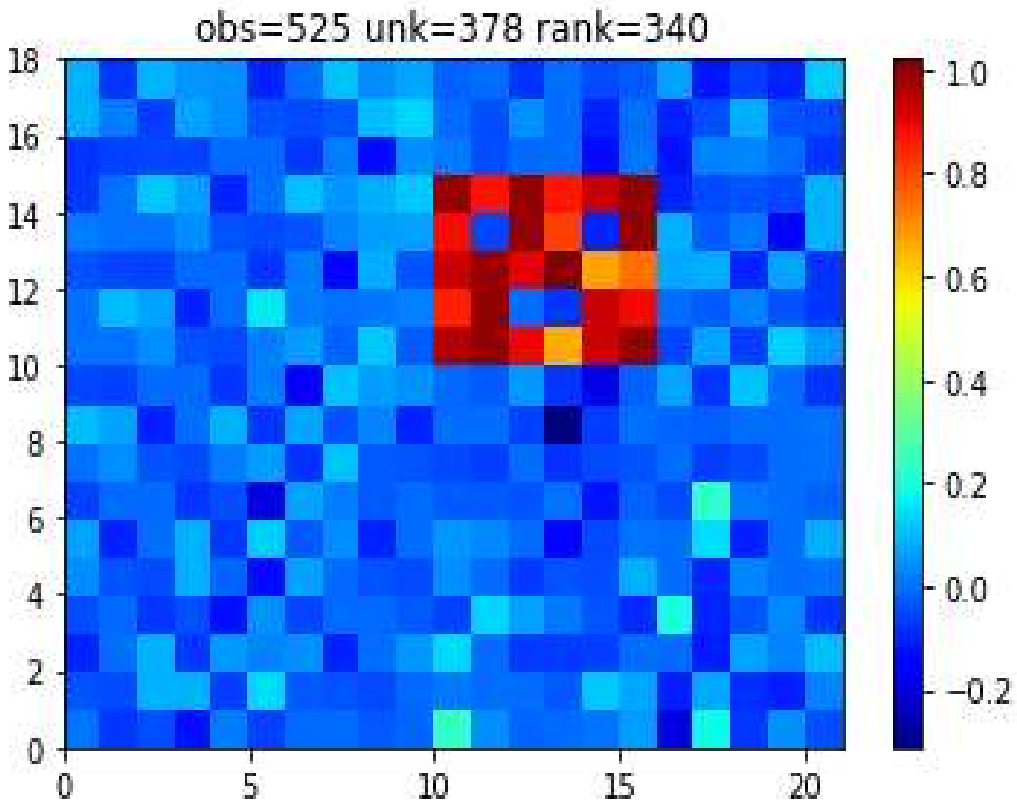
Problema fortemente subdeterminado amostra cada 5°



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

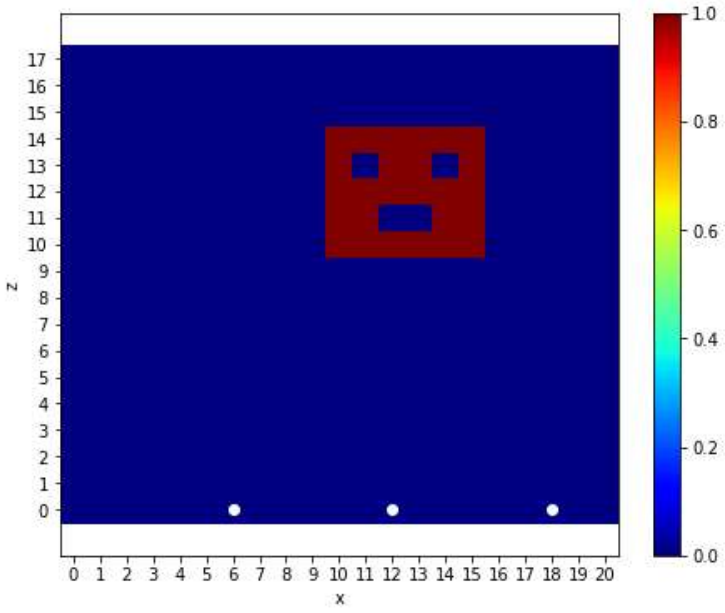


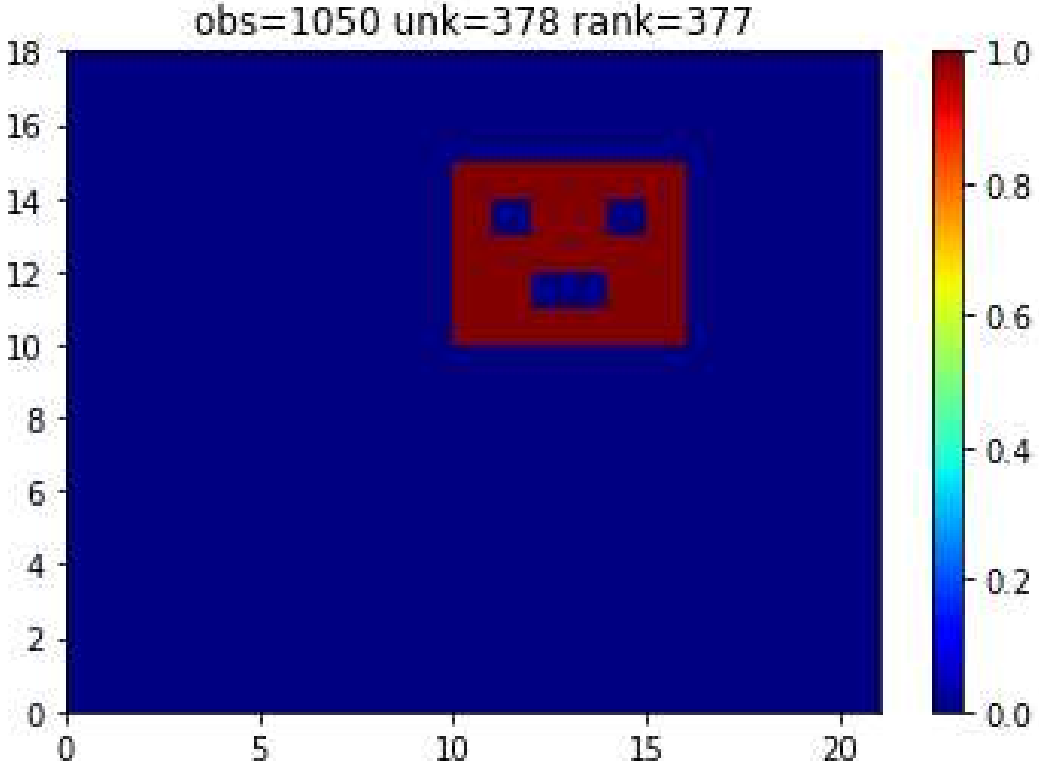
1°



Observações cada $0.5° \ (3 \times 350)$

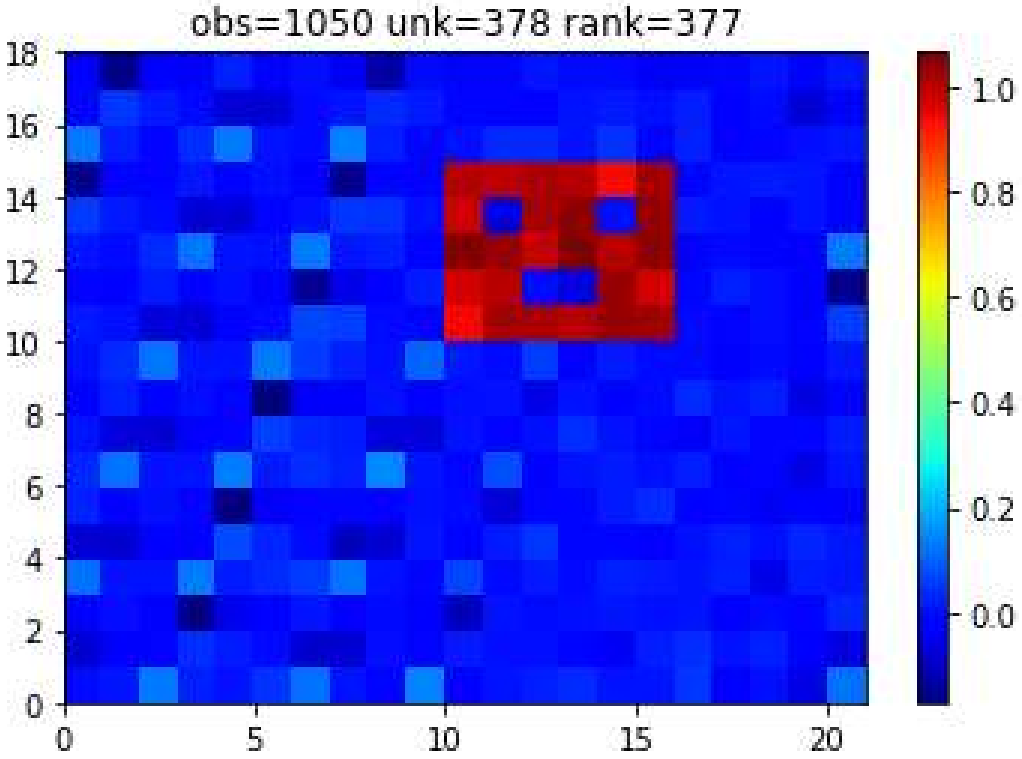
rank: maior submatriz
não singular



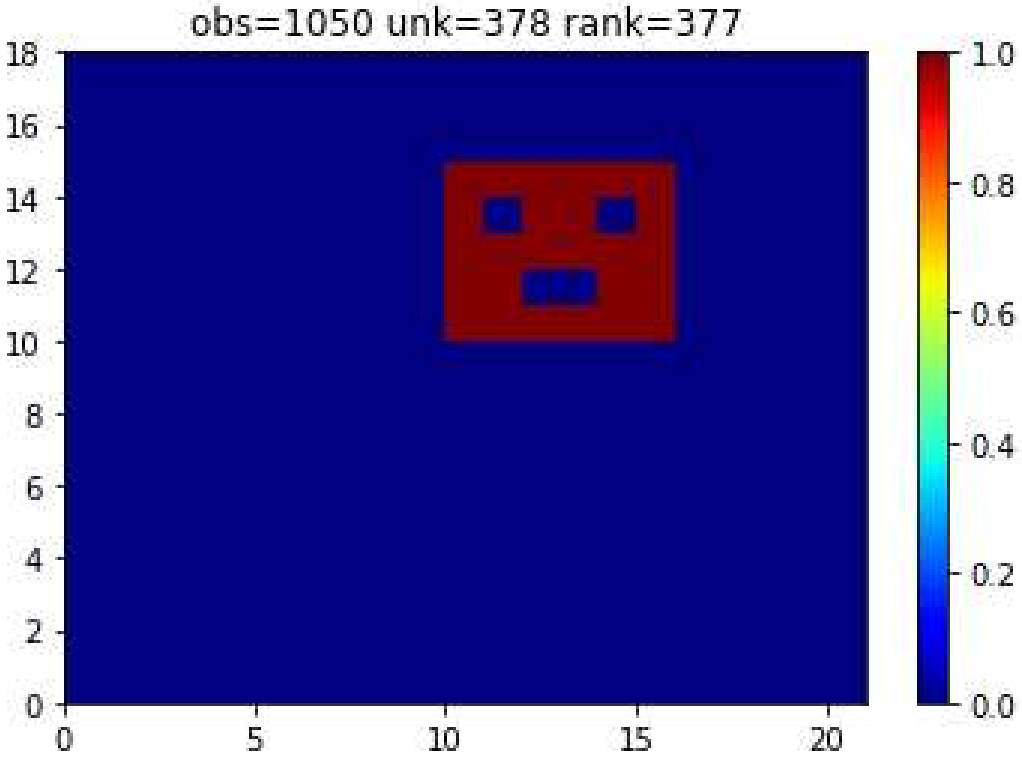


```
Y=Y*(1+(np.random.random(Y.shape)*0.5)*0.05)
```

5% de ruído



0% de ruído

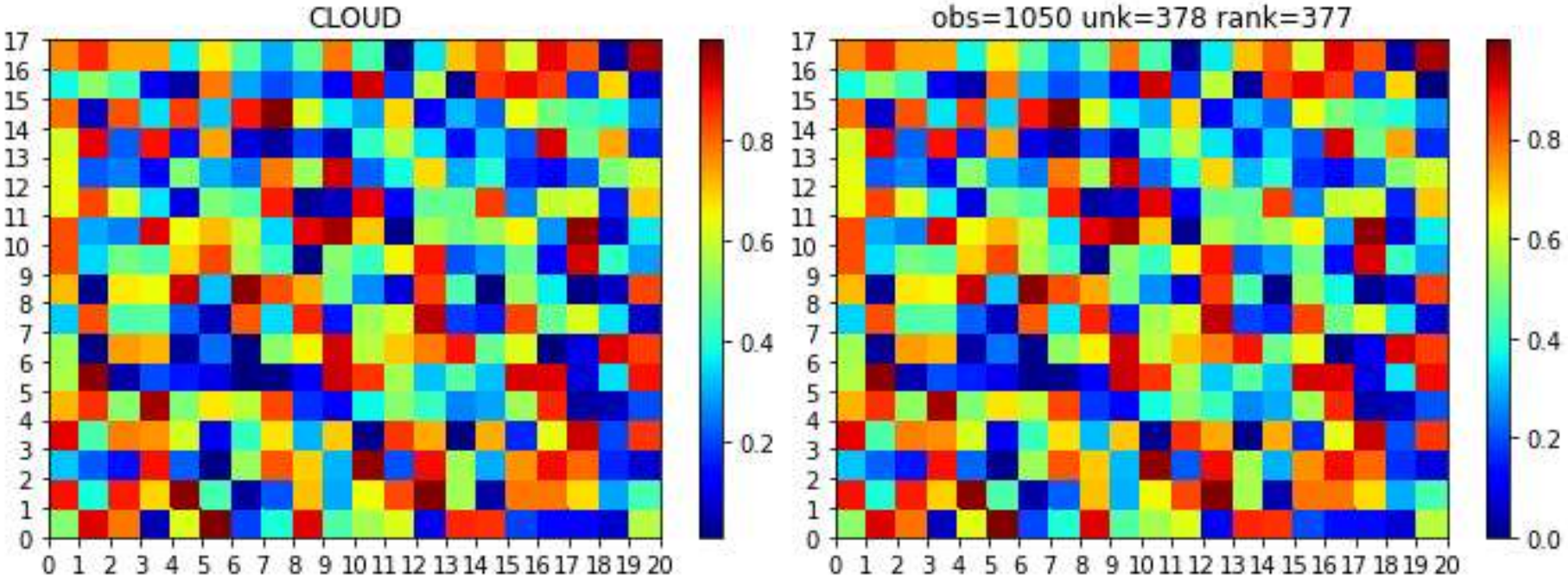


Mantendo a mesma matriz A, novo CLOUD (random)

```
CLOUD=np.random.random(CLOUD.shape)
plt.figure();m1=plt.pcolor(xis,zed,CLOUD,cmap='jet');plt.xticks(x);plt.yticks(z)
plt.colorbar(m1);plt.title('CLOUD')
Y=np.matmul(A,np.reshape(CLOUD,(nx*nz))) #observações vetorizadas
Yy=np.reshape(Y,(nEa,nEx)).transpose()
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
Xmat=np.reshape(Xest,(nx,nz))
plt.figure()
m4=plt.pcolor(xis,zed,Xmat,cmap='jet');plt.colorbar(m4);plt.xticks(x);plt.yticks(z)
plt.title('obs='+str(nEx*nEa)+' unk='+str(nx*nz)+' rank='+str(np.linalg.matrix_rank(A)))
```

Nuvem aleatória

Inversão (amostra 0.5°)

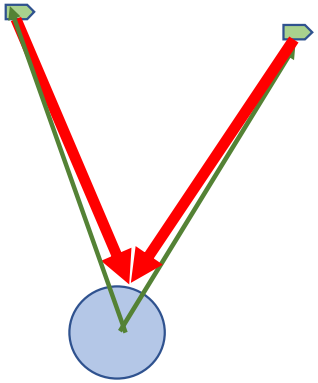


Ciências ULisboa

Modelação Numérica 2022
Aula 20

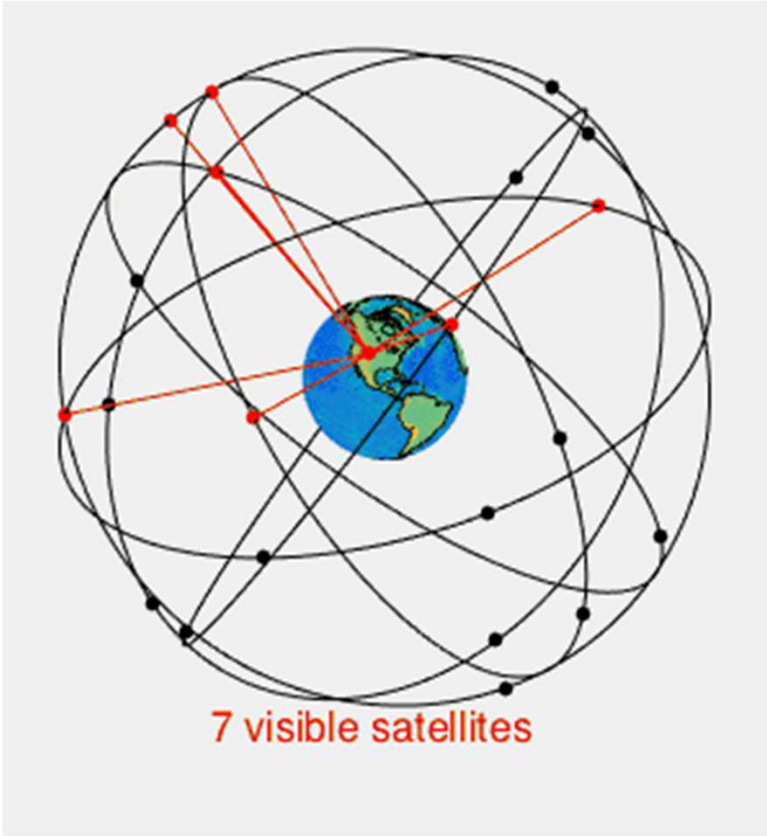
Exercícios de aplicação

Localização 3D por GPS



Vamos esquecer a geometria esférica.
Vamos considerer 2 ângulos: altura e azimute.
Altura: Ângulo com a horizontal
Azimute: Ângulo com o eixo dos xx

$$t = \frac{\sqrt{x^2 + y^2 + z^2}}{c} + \varepsilon$$

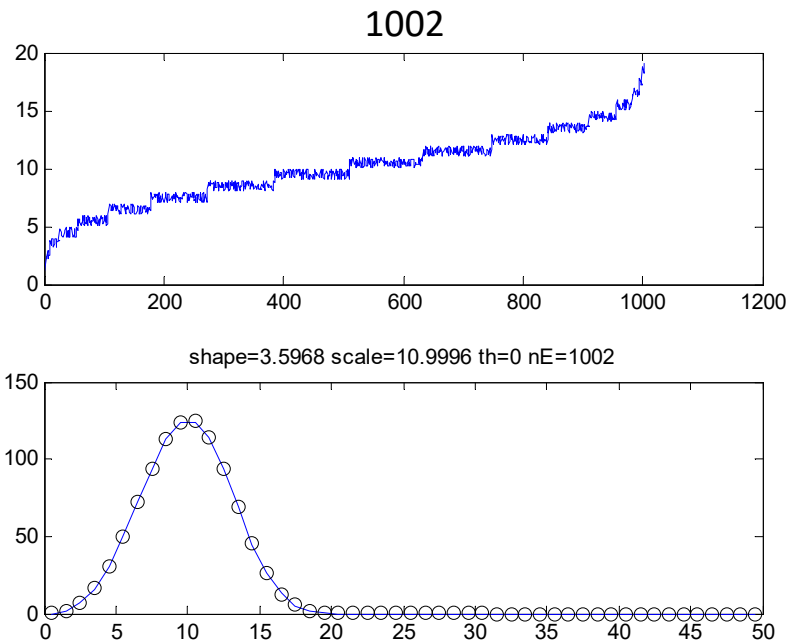


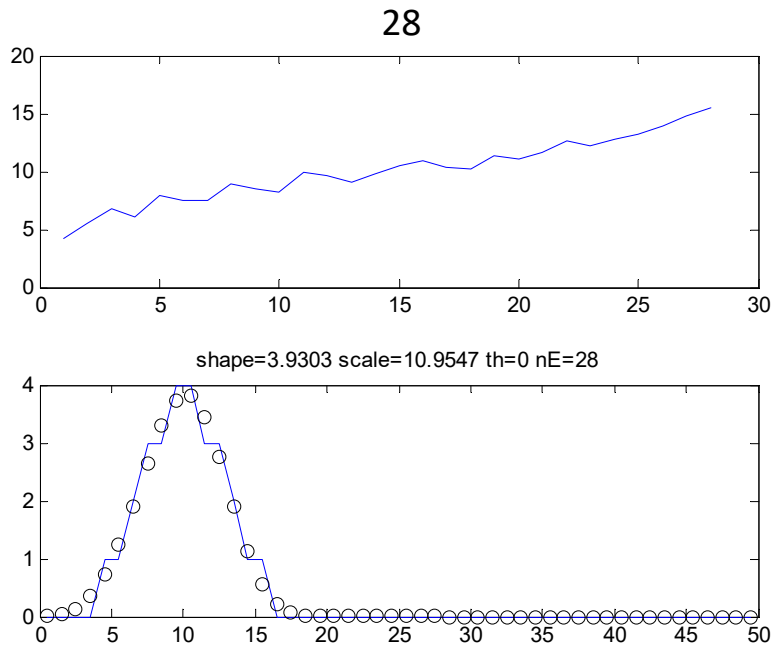
wikipedia

Função de custo para ajuste do histograma bin a bin

```
function iniCOST(tipo,dados)
global tE wE wH nE
shape=3.6;scale=11;threshold=0;nE=10000;tE=[0.5:49.5];nbin=length(tE);
wB=nearest(weibull(shape,scale,threshold,tE)*nE);wB=wB'; %N°amostras
nE=sum(wB);wE=zeros(nE,1);kE=0; %correção nE (nearest)
for ibin=1:nbin
    New=wB(ibin);
    wE(kE+1:kE+New)=tE(ibin)+rand(New,1)-0.5;
    kE=kE+New;
end
wH=hist(wE,tE); %Histograma sintético
end
function custo=cost(V)
global tE wE wH nE
custo=mean((weibull(shape,scale,threshold,tE)*nE-wH).^2);
end
function w=weibull(shape,scale,threshold,t)
w=real(shape/scale*((t-threshold)/scale).^(shape-1)…
      .*exp(-((t-threshold)/scale).^shape));
end
```

Sensibilidade à dimensão da amostra





Ajuste por máxima verosimilhança (*maximum likelihood)*

Escolher a solução mais provável (o conjunto de parâmetros a que tem maior probabilidade de produzir a amostra)

Maximizar: $J(\lambda, k, x) = \prod \; p(\lambda, k|x) \Longrightarrow$

$$Minimizar: \quad (-ln(J)) = -\sum ln(p(x))$$

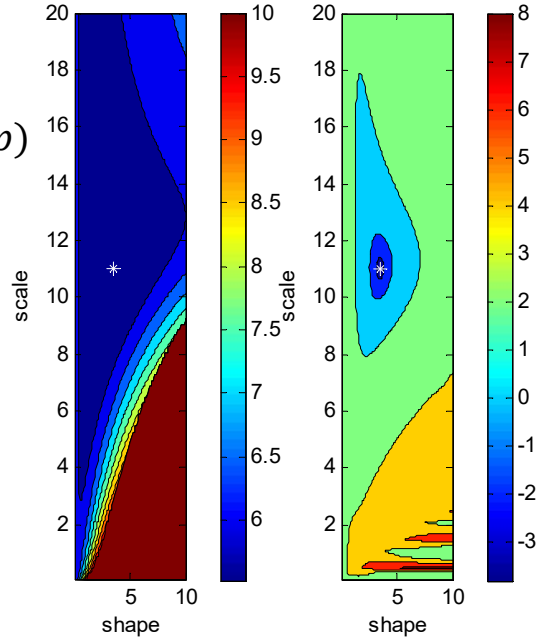
```
function custo=cost(V)
global tE wE wH nE
ww=weibull(shape,scale,threshold,wE);
custo=-sum(log(ww));
end
```

wE é a amostra, não o cálculo por bin (1000 termos e não 50)

Comparação ML vs bin fit

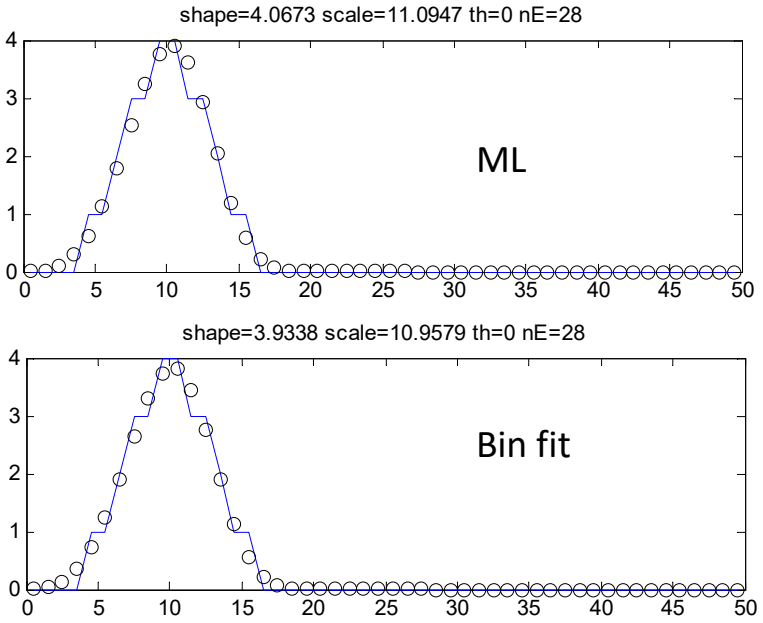
ML

$$-\sum \ln(p)$$



Bin fit

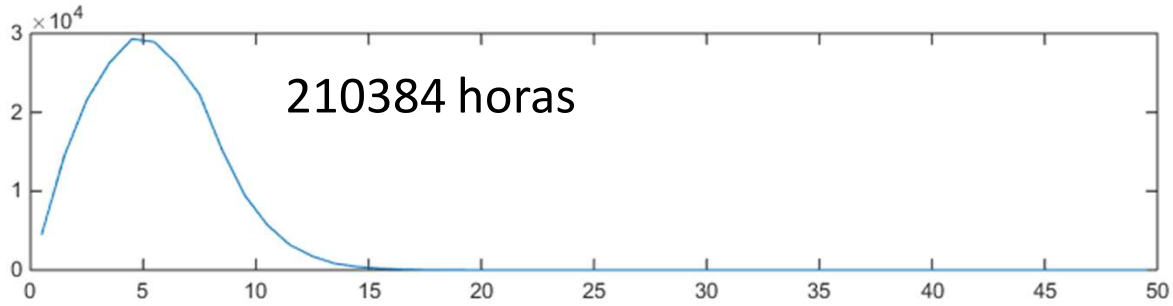
RMSE



Estima de k, λ, τ: por ajuste a mínimos quadrados do histograma

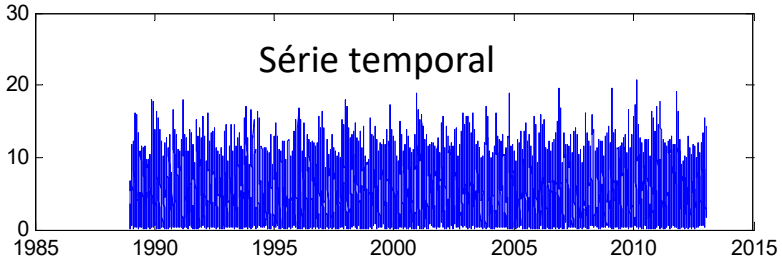
Vamos considerar uma amostra realista de dados de vento constituída por observações horárias durante 20 anos.

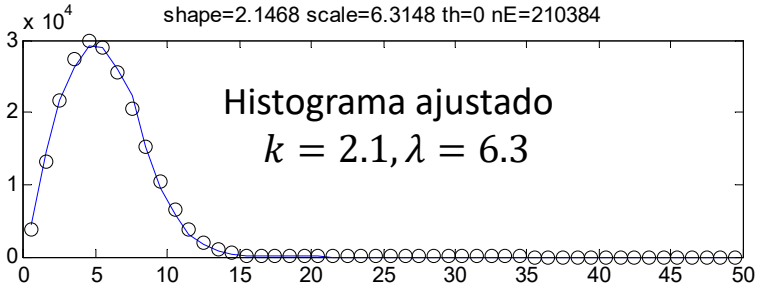
Vamos admitir que conhecemos o domínio do histograma: entre 0 e uma velocidade máxima, possivelmente superior à velocidade máxima registada na amostra. O nosso objetivo é calcular a curva de Weibull que melhor se ajusta ao histograma da amostra. Vamos usar o método de optimização já descrito a 3 dimensões.



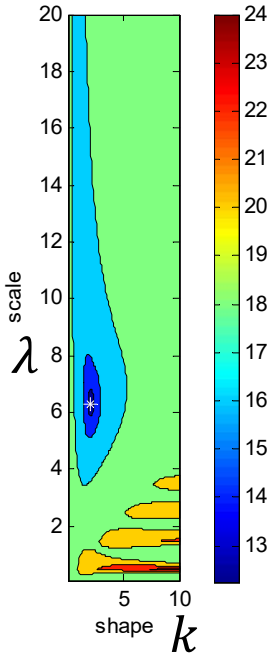
Ajuste de dados “reais”

```
function iniCOST(tipo,dados)
global tE wE wH nE %measurements
global windE windB binW
wrfdata=load('wrf_9km_IGIDL.dat'…
    ,'-ascii');
windE=sqrt(wrfdata(:,3).^2…
    +wrfdata(:,4).^2);
nE=length(windE);
binW=[0.5:49.5]'; %bins
windB=hist(windE,binW)';
clear wrfdata;
end
function custo=cost(V)
global windE windB binW nE
shape=V(1);scale=V(2);threshold=V(3);
custo=mean((weibull(shape,scale,threshold,binW)*nE-windB).^2);
end
```





Função de custo



Dados de temperatura

Ajuste de Weibull