SIS Scuola Interdipartimentale delle Scienze, dell'Ingegneria e della Salute

DiST Dipartimento di Scienze e Tecnologie

Laurea Magistrale in STN

Applicazioni di Calcolo Scientifico e Laboratorio di ACS

(12 cfu)

prof. Mariarosaria Rizzardi

Centro Direzionale di Napoli – Isola C4
stanza: n. 423 – Lato Nord, 4° piano
tel.: 081 547 6545
email: mariarosaria.rizzardi@uniparthenope.it

Argomenti trattati

- Calcolo Numerico in MATLAB:
  - Fitting di dati discreti.
  - Interpolazione di superfici.

Fitting di dati discreti

Richiami

Assegnati i vettori dei campioni (x_i,y_i), di n componenti, si vuole costruire una funzione f, semplice da calcolare, che li descriva.

Ciò può essere realizzato richiedendo che:

f **"passi" per i punti** $y_i=f(x_i)$ *(interpolazione)*

f **"si scosti il meno possibile" dai punti** ? *(m.a. = migliore approssimazione - best fit)*

Che significa *"si scosti il meno possibile"* ?

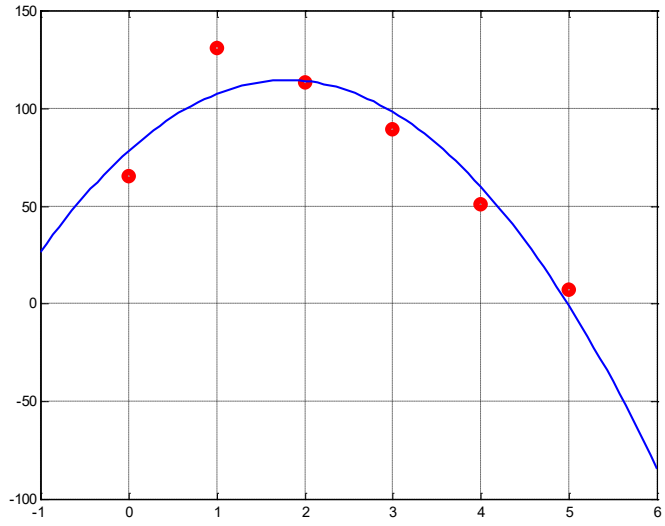
Esempio: Si vuole ***stimare*** (in funzione del tempo) la posizione ***f*(t)** di un corpo lanciato da una certa altezza.

Modello: $f(t) = h_0 + v_0 t + 0.5 g t^2$ (**g** accelerazione di gravità)

dati

x	*y*
0	65
1	131
2	113
3	89
4	51
5	7

Non c'è nessuna parabola "passante" per i dati sperimentali, ma una che **descrive il loro andamento** (modello attendibile) è la seguente…



La parabola f è il *polinomio* (di 2° grado) *dei minimi quadrati* (*pol. approssimante*), detto così perché *minimizza la somma dei quadrati delle lunghezze dei* segmenti verdi, cioè tra tutti i polinomi di 2° grado rende minima la quantità E dove

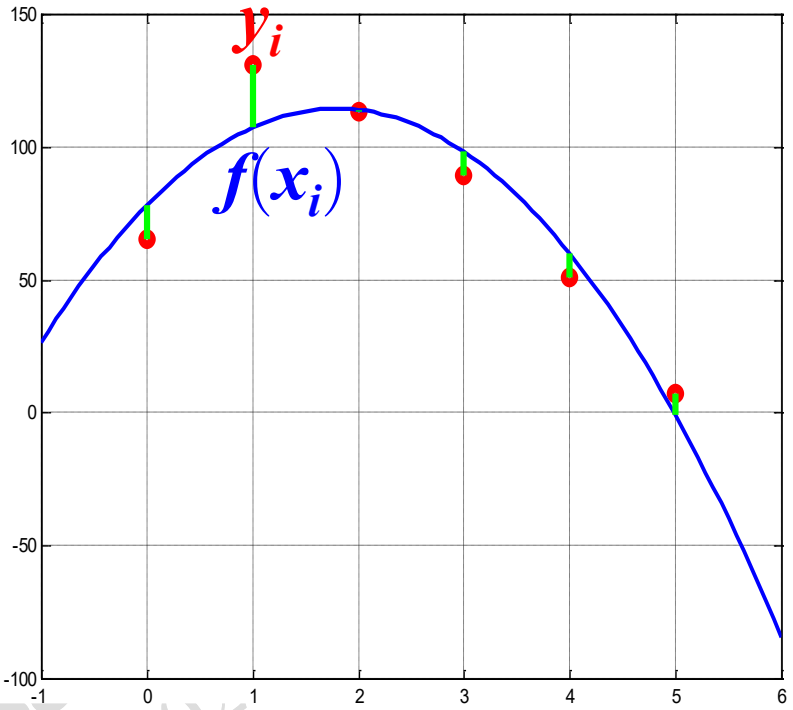
$$E(\alpha,\beta,\gamma) = \| y - F \|_2$$

e

$$F_i = f(x_i)$$

cioè

$$E^2 = \sum_{i=1}^{n} \left[y_i - \alpha x_i^2 + \beta x_i + \gamma \right]^2$$



misura dello scostamento

in MATLAB ...

```
coef=polyfit(x,y,2);
ft=polyval(coef,t);
```

```
x=[0 1 2 3 4 5]; y=[65 131 113 89 51 7];
t=linspace(-1,6,99);
coef=polyfit(x,y,2);
ft=polyval(coef,t);
plot(t,ft,'b',x,y,'or'); grid on
…
```

» coef=polyfit(xi, yi, g);

Restituisce i coefficienti **coef** del polinomio dei minimi quadrati di grado **g** relativo ai dati x_i, y_i, $i=0,1,\ldots,n$.

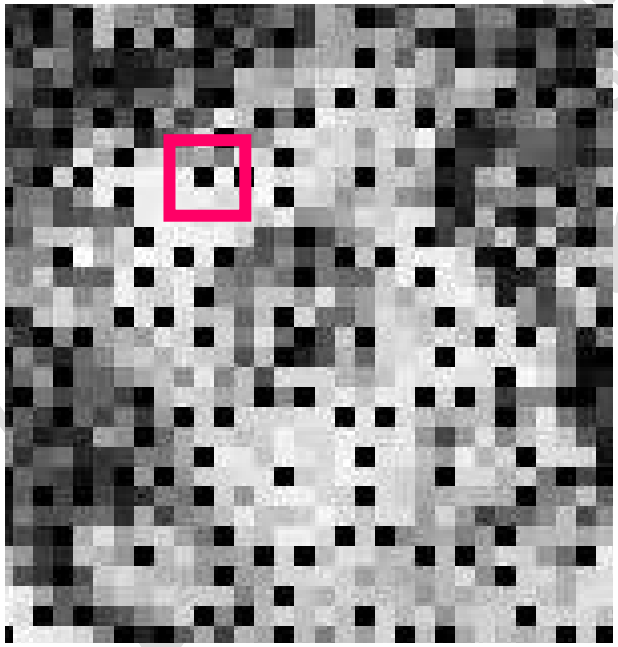
» ft=polyval(c, x);

Restituisce i valori che il polinomio, di coefficienti $\mathbf{c}=[c_1,c_2,\ldots,c_n]$, assume in **x** (usa l'***Algoritmo di Horner***):

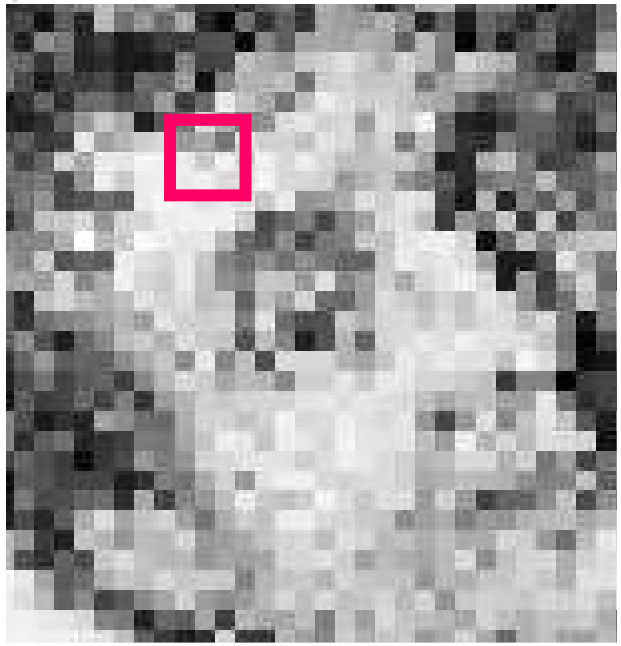
$$P_{n-1}(x) = c_1 x^{n-1} + c_2 x^{n-2} + \ldots + c_{n-1} x + c_n$$

Esempio

Riempire i pixel mancanti di un'immagine



i pixel "neri" circondati dai chiari vengono smorzati



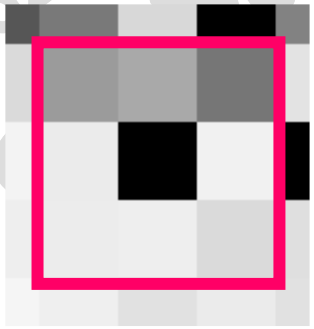
P(i-1,j-1)	P(i-1,j)	P(i-1,j+1)
P(i,j-1)	**P(i,j)**	P(i,j+1)
P(i+1,j-1)	P(i,j)	P(i+1,j+1)

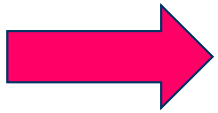
A ciascun pixel "nero" **P(i,j)** è assegnato il colore ottenuto come approssimazione ai minimi quadrati degli 8 pixel adiacenti.

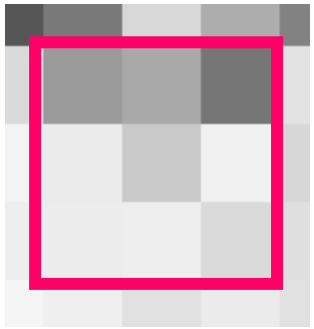
155	169	118
235	**0**	241
236	238	218

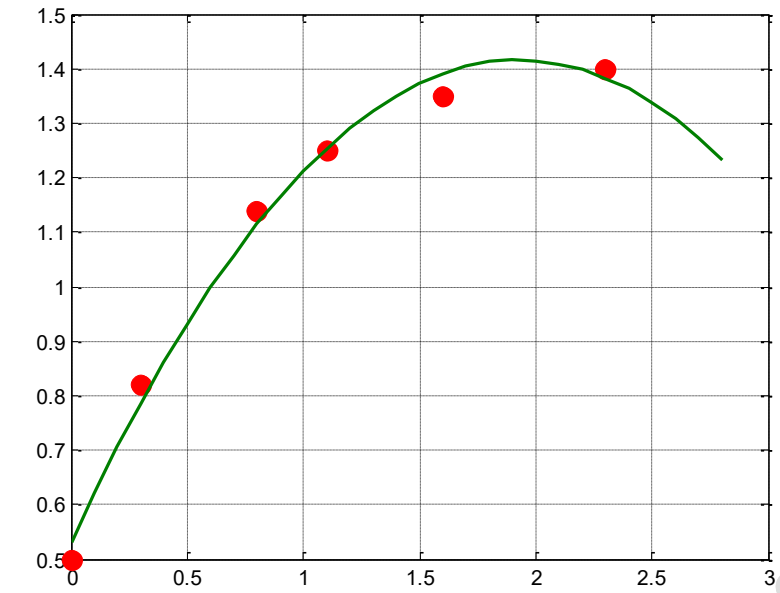
155	169	118
235	**201**	241
236	238	218

```
P(i,j)=polyfit(1:8,[241 118 169 155 235 236 238 218],0)
```










```
t=[0 .3 .8 1.1 1.6 2.3]';
y=[0.5 0.82 1.14 1.25 1.35 1.40]';
t2=0:.1:2.8;
y2=polyval(polyfit(t,y,2),t2);
plot(t,y,'o',t2,y2), grid
```

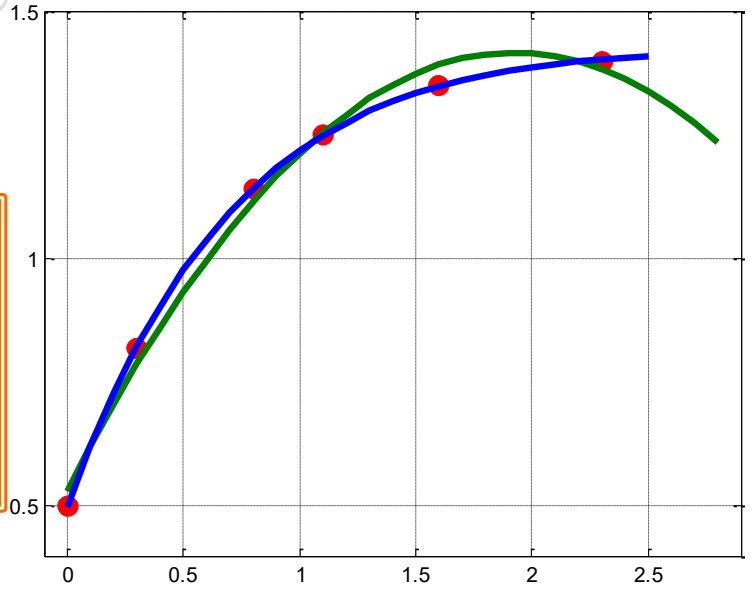
Non sempre il modello polinomiale è attendibile !

Volendo ricostruire una funzione del tipo $f\ t = a_0 + a_1 e^{-t} + a_2 t e^{-t}$

non si può usare polyfit(), ma si può risolvere il sistema lineare con il metodo dei Minimi Quadrati

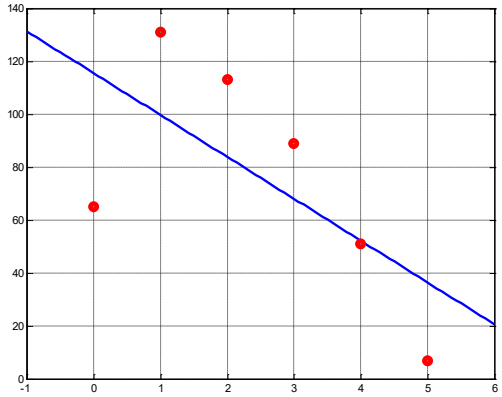
$$f\ t = a_0 + a_1 e^{-t} + a_2 t e^{-t}$$

```
X=[ones(size(t)) exp(-t) t.*exp(-t)];
a=X\y;
T=(0:0.1:2.5)';
Y=[ones(size(T)) exp(-T) T.*exp(-T)]*a;
plot(t,y,'o',t2,y2,T,Y), grid
```

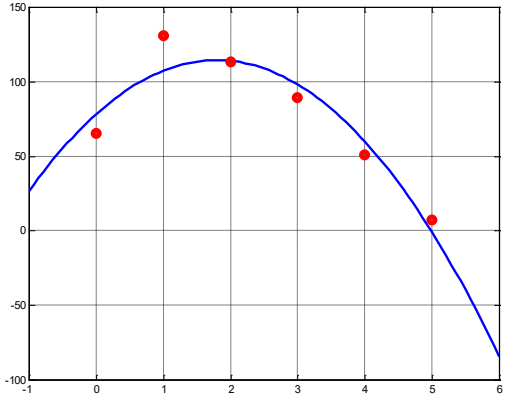


Polinomio dei minimi quadrati di grado g… su 6 nodi (dati di fitting)

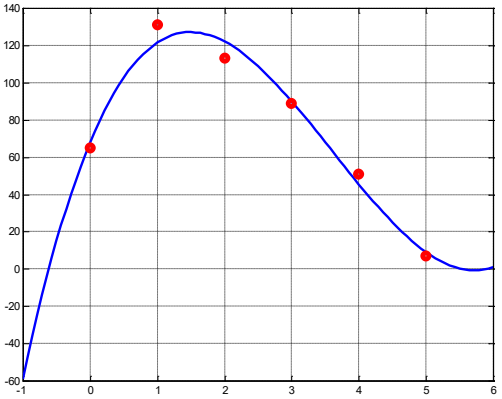
…grado g=1



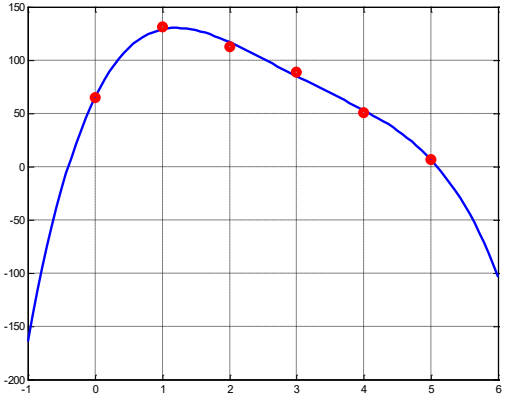
…grado g=2



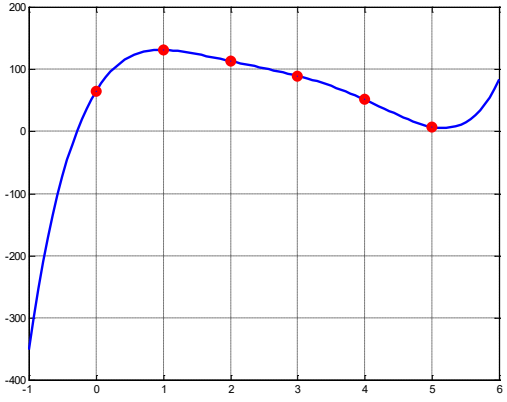
…grado g=3



…grado g=4

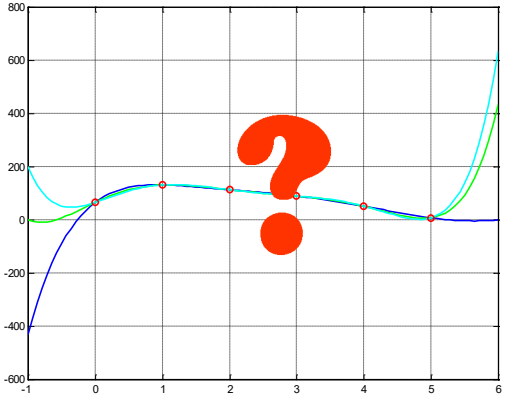


…grado g=5



soluzione non unica!

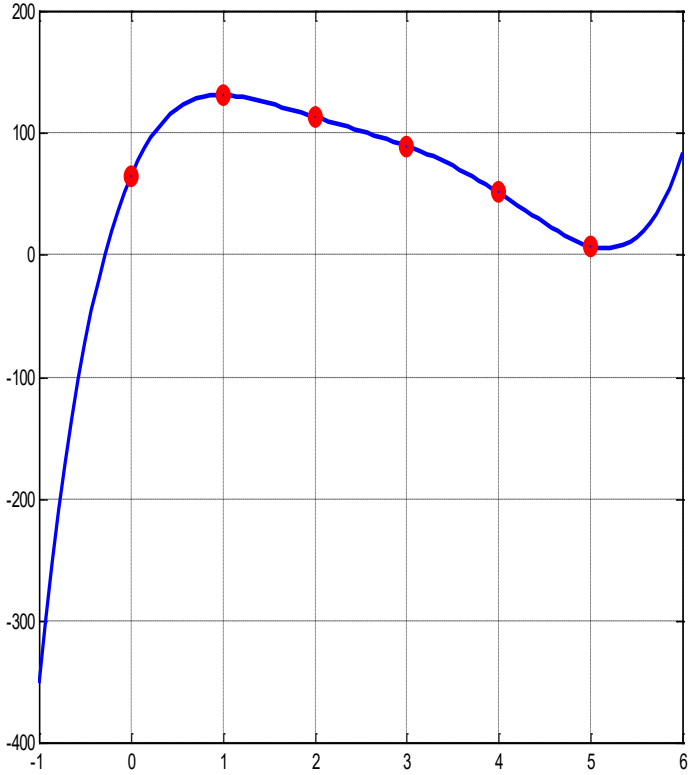
…grado g=6



Interpolazione

…grado 5 = 6 nodi –1

…grado g = n –1
a partire da **n nodi**



Il polinomio dei minimi quadrati passa per gli* n *punti ed è unico

Interpolazione polinomiale di Lagrange

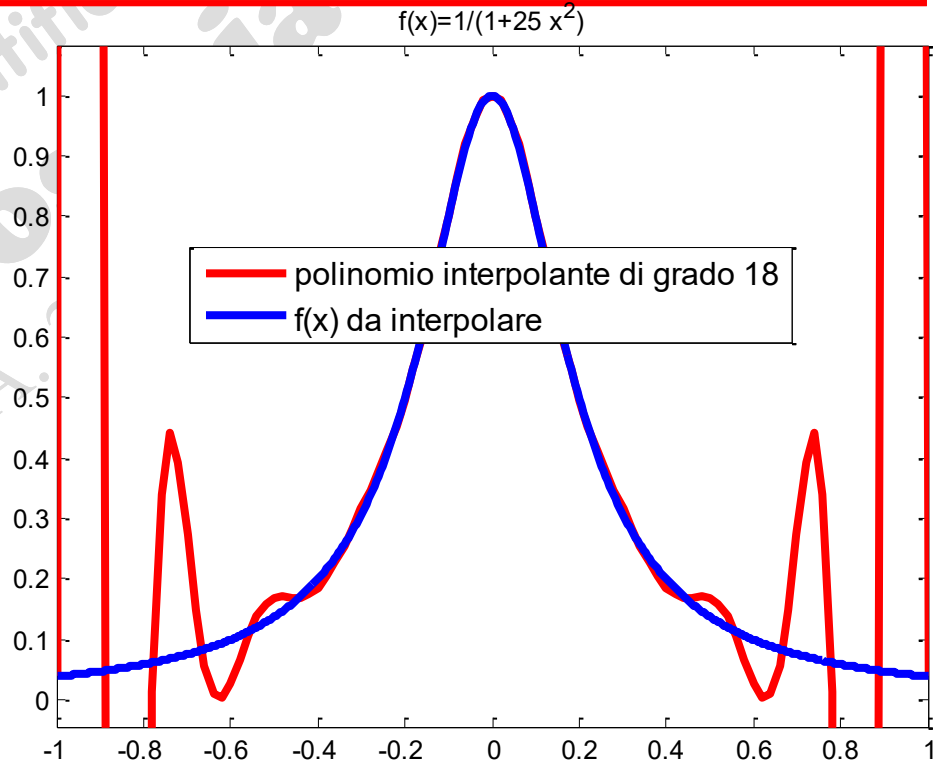
Interpolazione polinomiale di Lagrange

Teorema

Assegnati n punti (x_i, y_i) con $x_i \neq x_j$ per $i \neq j$, esiste un unico polinomio $P_{n-1}(x)$, di grado al più $n-1$, tale che

$$P_{n-1}(x_i) = y_i \quad \forall\, i = 1,2,\ldots,n$$

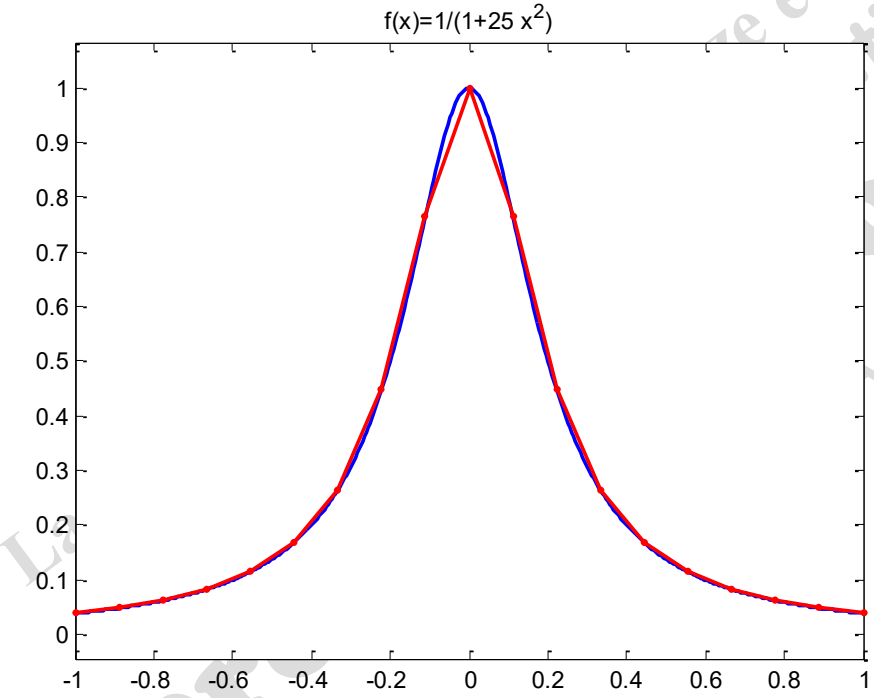
```
f=@(x)1./(1+25*x.^2);
x=linspace(-1,1,19); y=f(x);
coef=polyfit(x,y,length(x)-1);
t=linspace(-1,1,101);
p=polyval(coef,t);
h1=plot(t,p,'r'); hold on
h2=fplot(f,[-1 1],'Color','b');
legend([h1;h2],'polinomio interpolante di grado 18','f(x) da interpolare')
axis([-1 1 0 1]); xlabel('x')
```



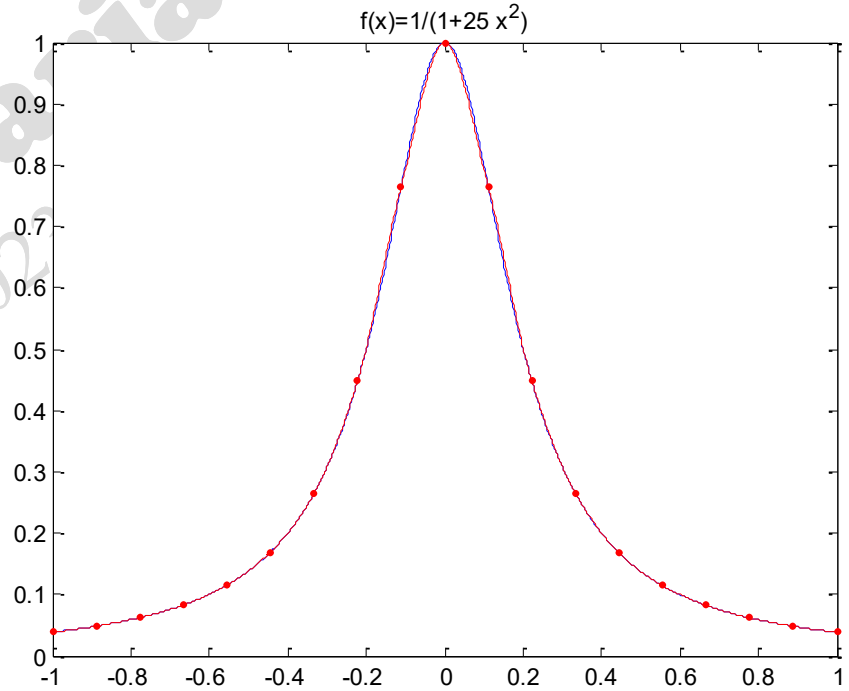
Al crescere del grado il polinomio tende ad oscillare !!!

Come ridurre le oscillazioni ?
… mediante l'interpolazione a tratti

Si dividono in blocchi adiacenti di "pochi" punti i dati da interpolare e si interpola su ogni blocco.



interpolazione a tratti lineare (blocchi di 2 punti)



interpolazione a tratti cubica (blocchi di 4 punti)

L'interpolazione a blocchi ricostruisce una funzione (polinomiale a tratti) che risulta continua ma la sua derivata nei nodi di raccordo è in generale discontinua.

Come garantire la continuità nella derivata?

mediante l'interpolazione con spline

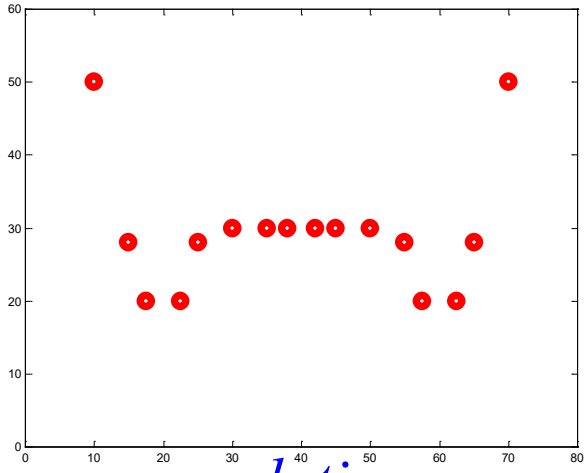
Le **spline** sono particolari polinomiali a tratti di grado **m** che garantiscono la continuità nelle derivate fino all'ordine **m-1**.

Più usata: spline naturale cubica interpolante

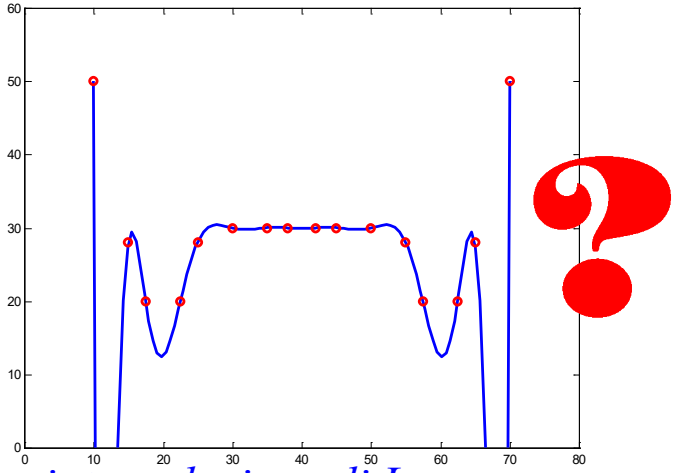
(polinomiale a tratti di grado al più 3 con derivate continue fino all'ordine 2)

Esempio: ricostruzione del profilo del catamarano

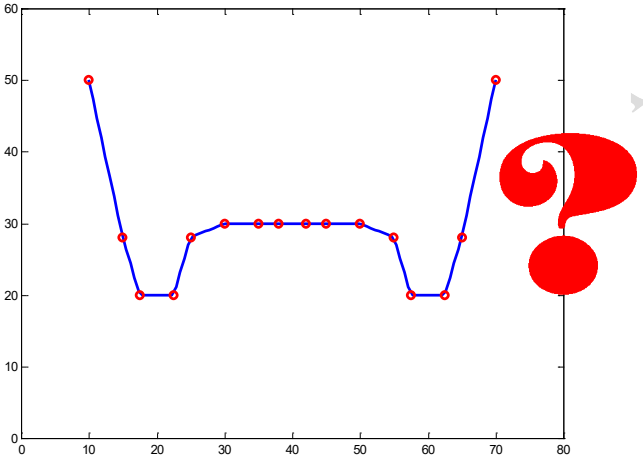
Che *"tipo"* di interpolazione ?



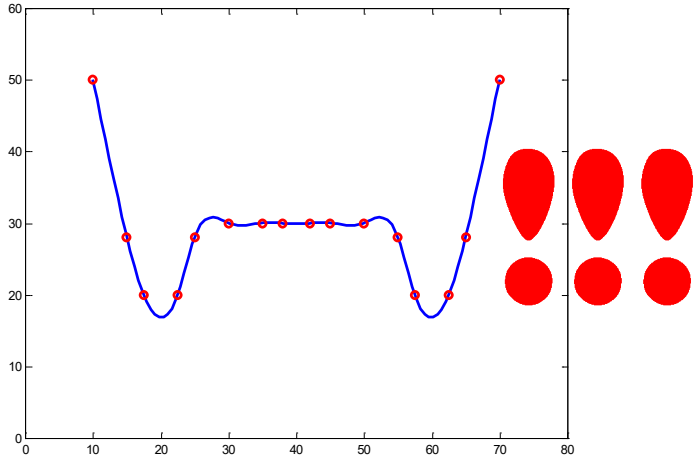
dati



interpolazione di Lagrange



interpolazione a tratti lineare



interp. con spline naturale cubica

Interpolazione in MATLAB

```
ft=interp1(x,y,t,'metodo');
```

Restituisce i valori assunti in **t** dal **polinomio interpolante a tratti** i dati **x,y**, dove

`'metodo'`	= **linear**	*interpolazione lineare*
	= **spline**	*interpolazione mediante* ***spline*** *naturale cubica*
	altri ...	

```
coef=interp1(x,y,'tipo','pp');
pt=ppval(coef,t);
```

pp = piecewise polynomial

valuta il polinomio a tratti

fornisce i coefficienti della funzione a tratti

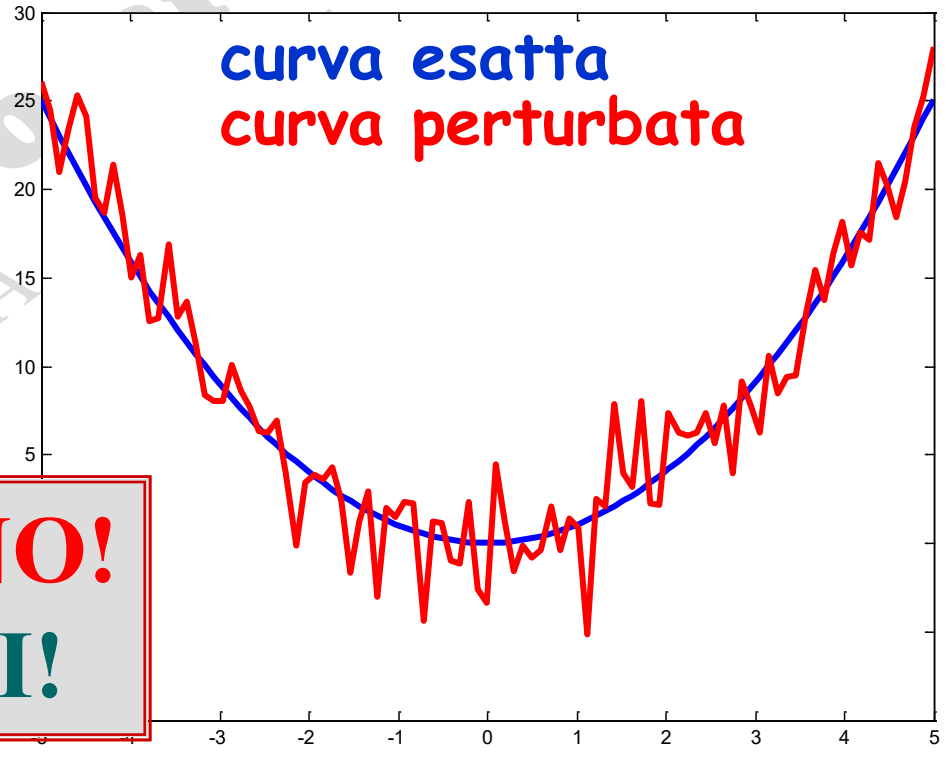
Esempio di best fit

`yi` dati esatti

`yp` dati perturbati

```
np=99; xi=linspace(-5,5,np); yi=xi.^2;
yp=yi+(-1.02).^fix(100*rand(1,np)).*rand(1,np);
plot(xi,yi,'b',xi,yp,'r')
```

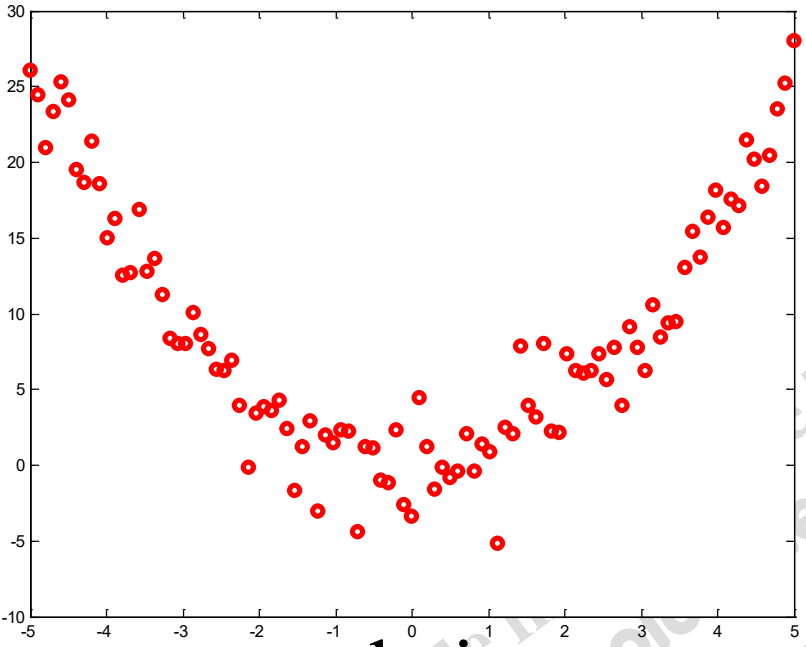
Come eliminare le perturbazioni per ricostruire la funzione esatta?



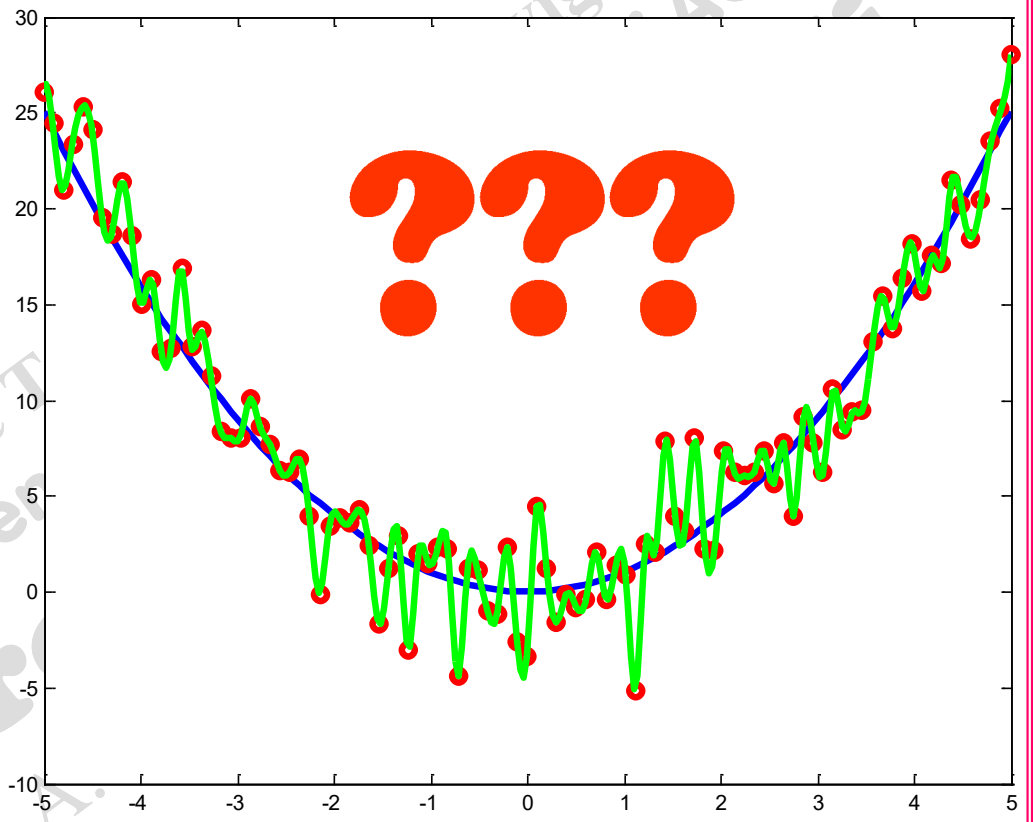
Interpolazione? NO!

Best fit? SI!

Interpolazione mediante splines



dati

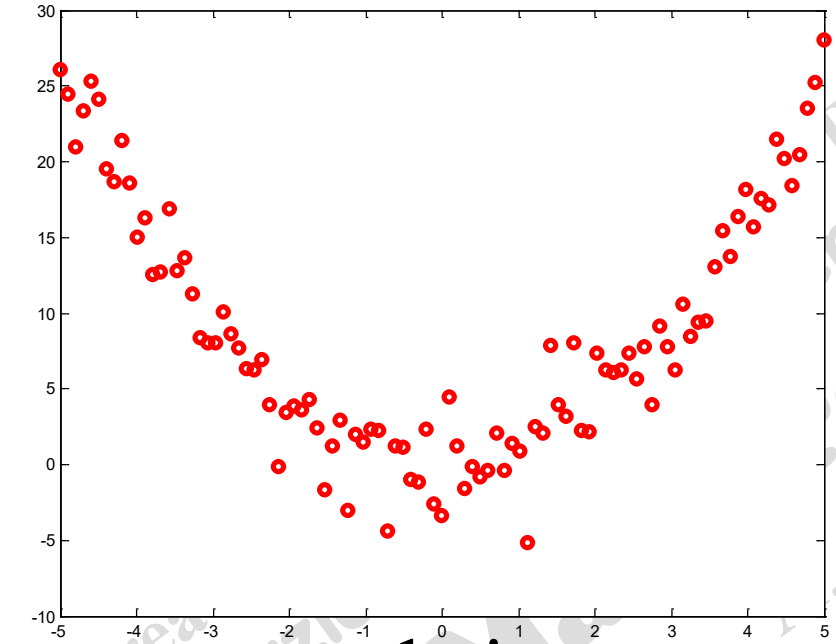



```
xx=linspace(-5,5,299);
yn=interp1(xi,yp,xx,'spline');
plot(xi,yi,'b', xi,yp,'ro', xx,yn,'g')
```

ricostruisce anche le perturbazioni !

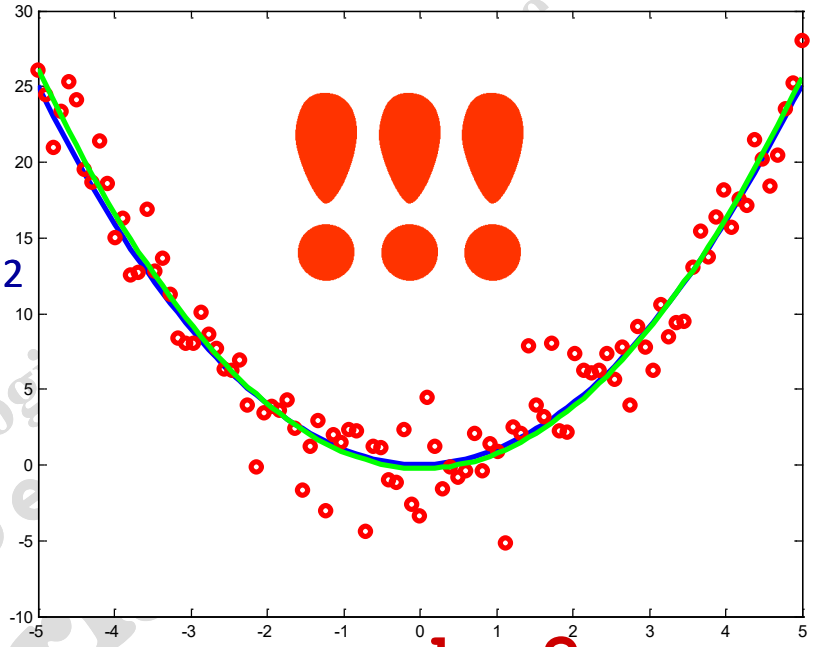
Fitting mediante minimi quadrati* polinomiale

* in parte 2

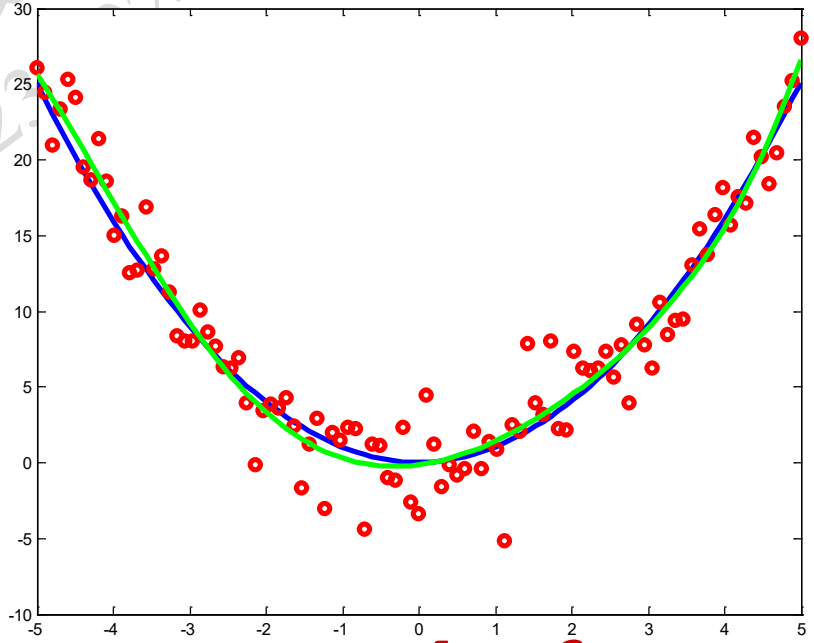


dati

```
coef=polyfit(xi,yp,grado);
yn=polyval(coef,xi);
plot(xi,yi,'b',xi,yp,'ro', ...
     xi,yn,'g')
```



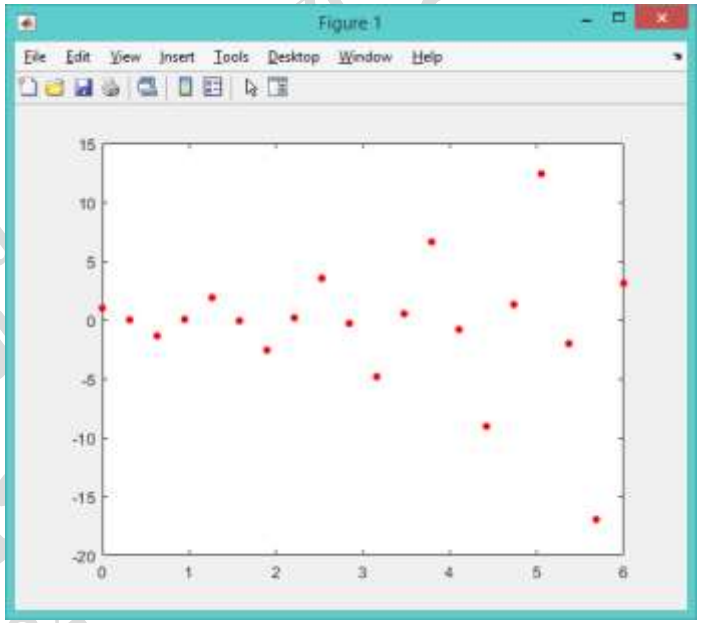
grado=2

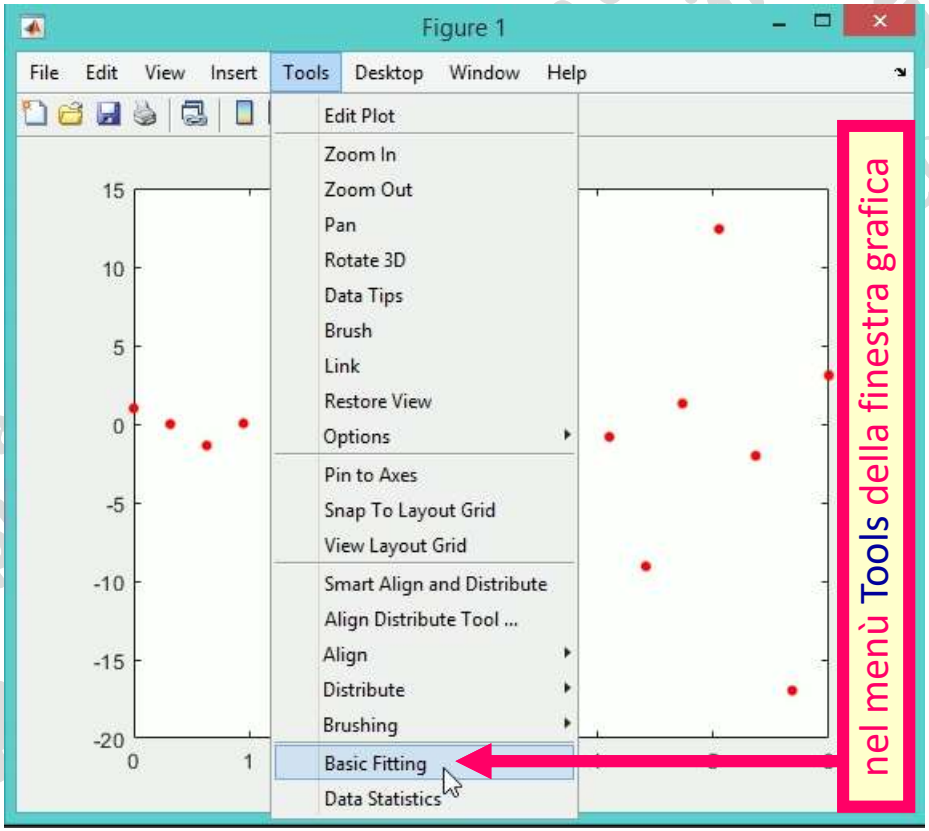


grado=6

Basic Fitting Tool di *MATLAB*

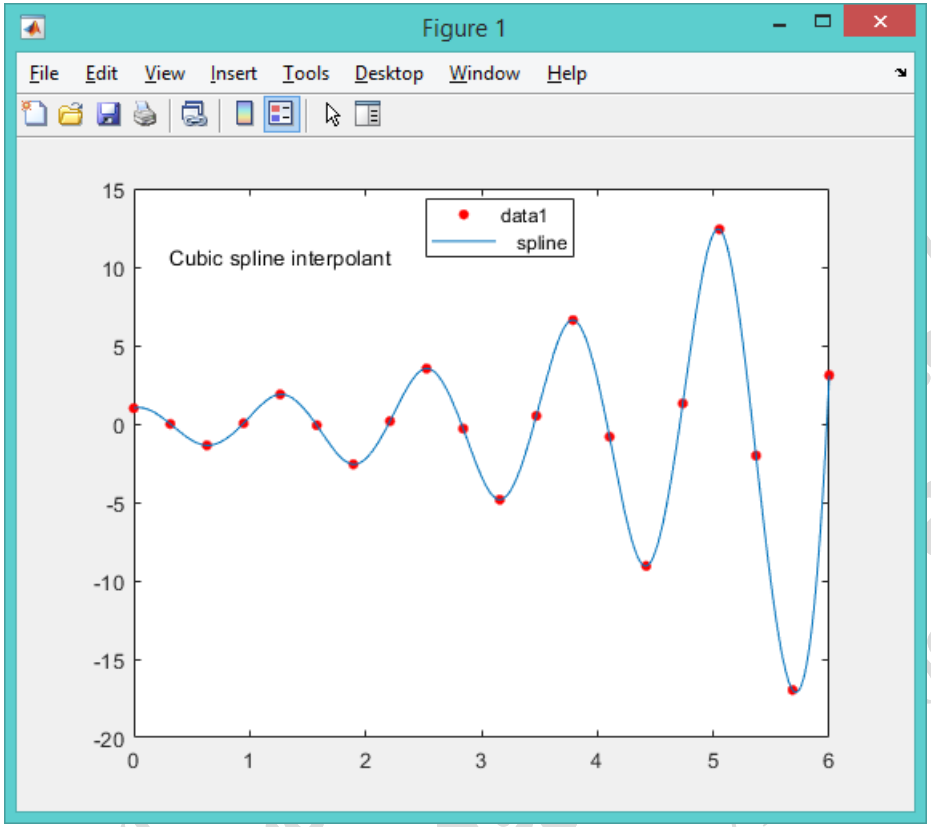
```
xj=linspace(0,6,20)';
fj=exp(xj/2).*cos(5*xj);
plot(xj,fj,'r.','MarkerSize',15)
```

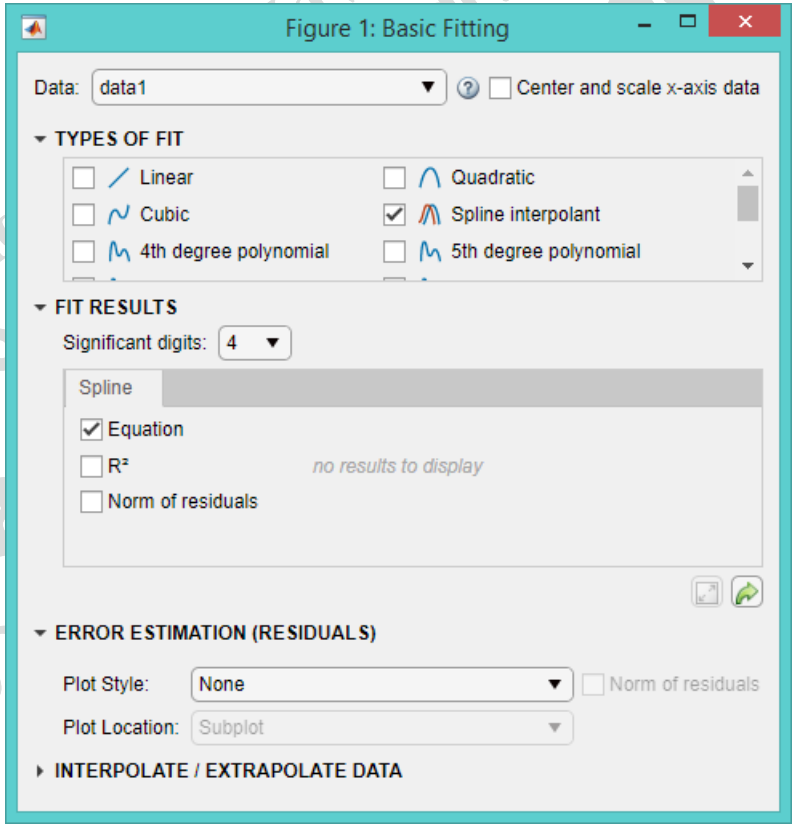




nel menù Tools della finestra grafica

Basic Fitting Tool di *MATLAB*





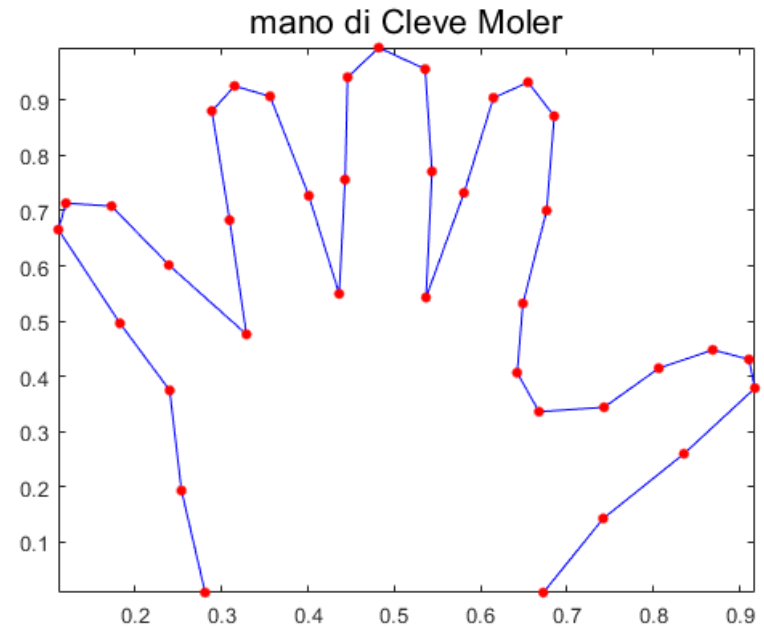
Interpolazione di curve in MATLAB

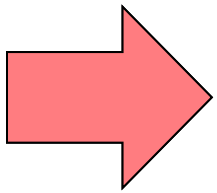
Come ricostruire una curva a partire dai suoi campioni?

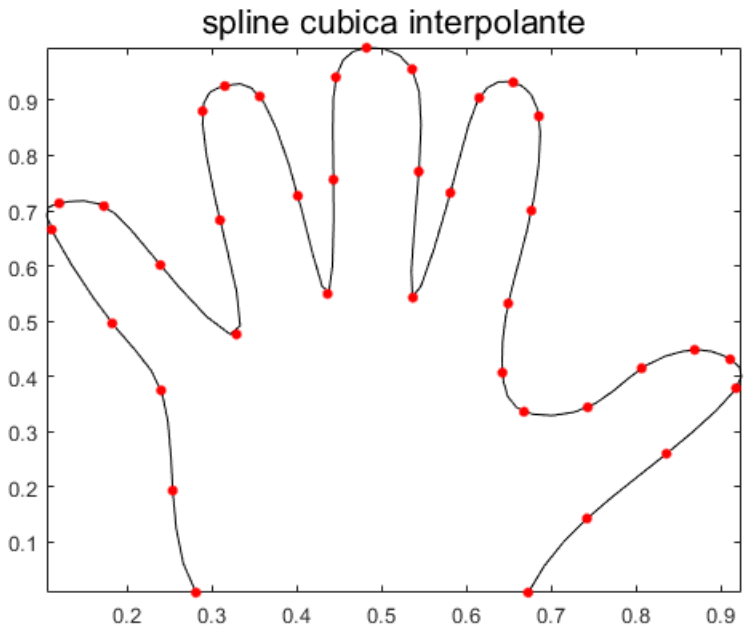
Si interpolano separatamente le ascisse e le ordinate dei campioni, introducendo delle ascisse fittizie: avendo n campioni $(x_i, y_i)_{i=1,\dots,n}$, si interpolano i dati $(1{:}n, x)$ e $(1{:}n, y)$

Esempio

```
load mano; xj=mano(:,1); yj=mano(:,2);
N=3*numel(xj); T=linspace(tj(1),tj(end),N)';
tj=(1:numel(xj))'; % nodi fittizi
X=interp1(tj,xj,T,'spline'); % o X=spline(tj,xj,T);
Y=interp1(tj,yj,T,'spline'); % o Y=spline(tj,yj,T);
plot(X,Y,'k',xj,yj,'.r','MarkerSize',15); axis tight; hold on
```

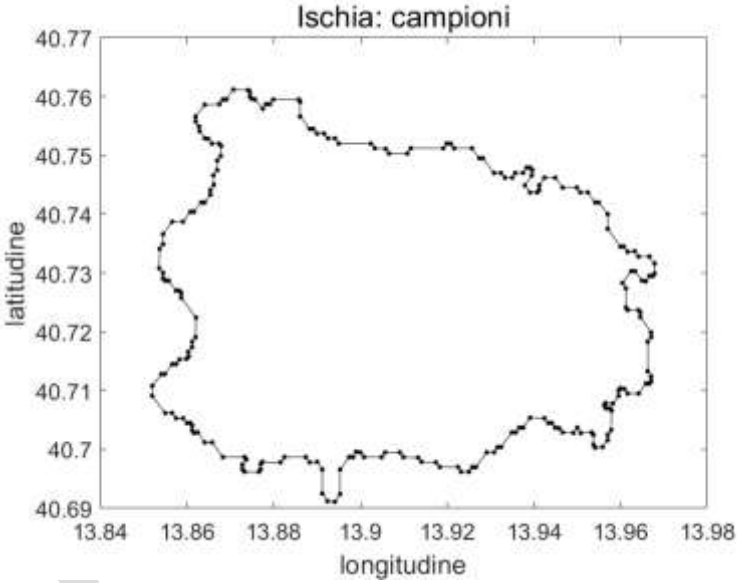


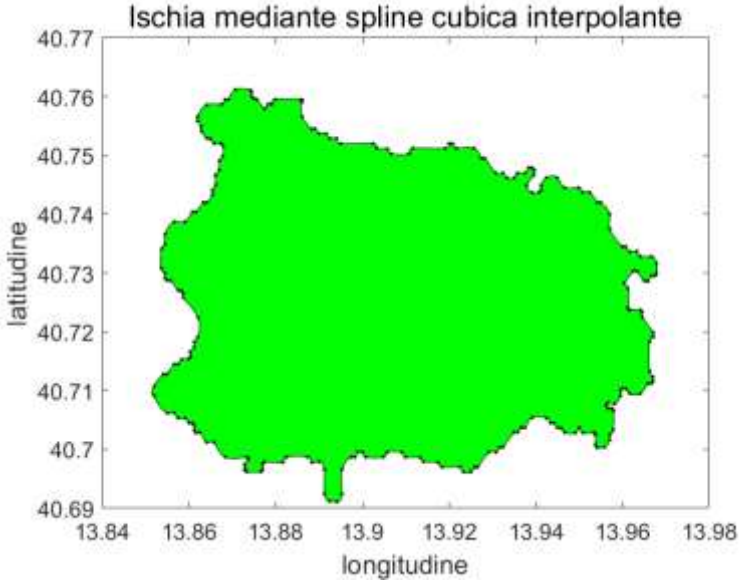




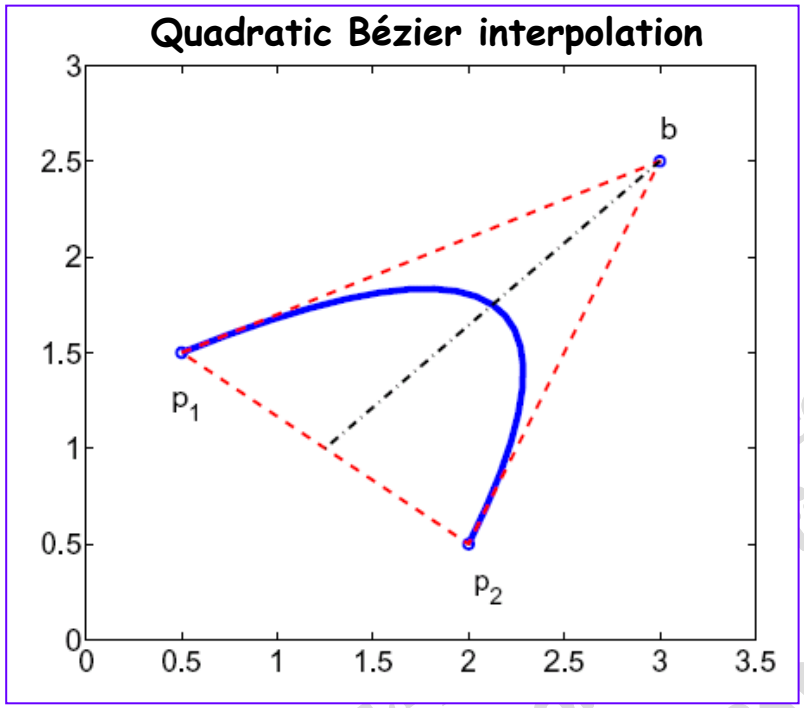
Esempio

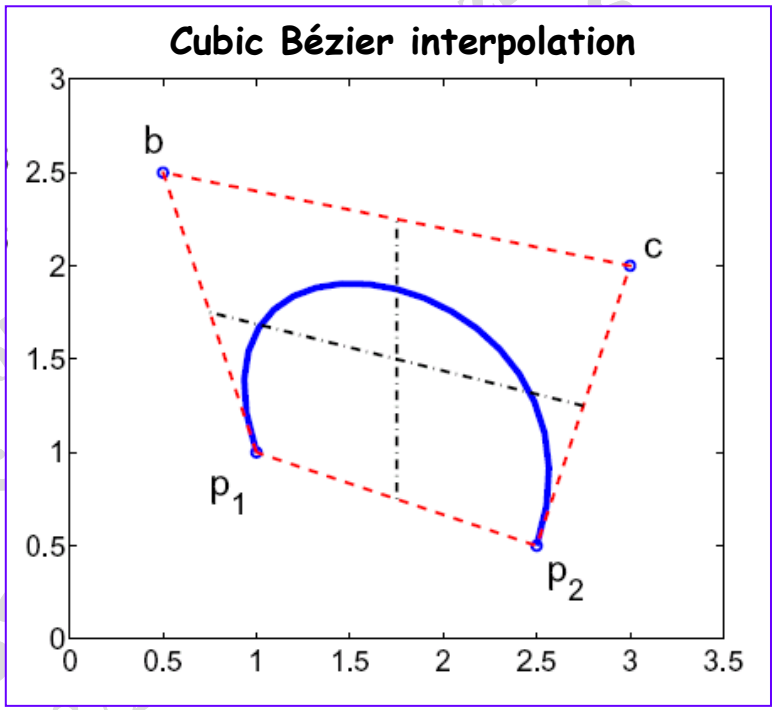

Giardini La Mortella
Spiaggia della Chiaia
Forio
SP270
Casamicciola Terme
Porto d'Ischia
Procida
L'olmo
Spiaggia libera della Chiaiolella
Cretaio
Ischia
Castello Aragonese d'Ischia
Isola d'Ischia
SP270
SP270
Serrara Fontana
Barano D'ischia
Piano Liguori
Sant'Angelo
Sant'Angelo

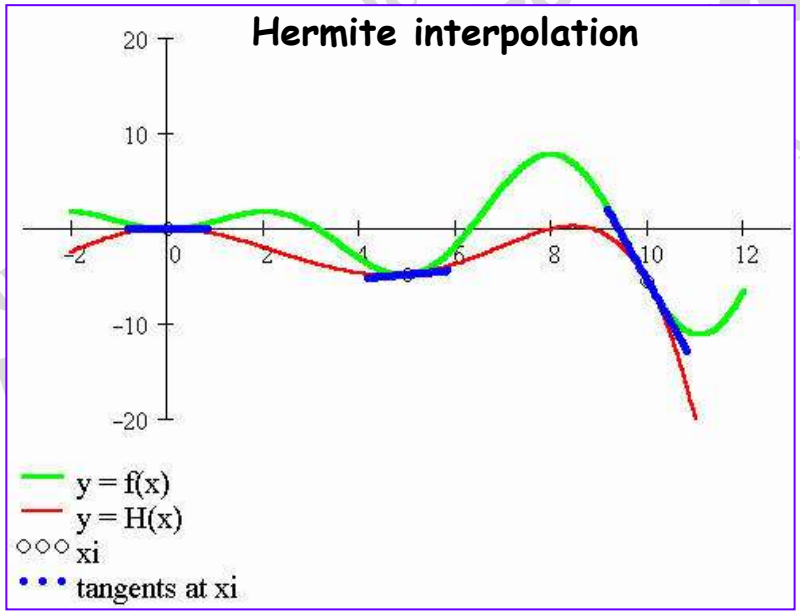

Ischia: campioni
latitudine
longitudine
40.77
40.76
40.75
40.74
40.73
40.72
40.71
40.7
40.69
13.84
13.86
13.88
13.9
13.92
13.94
13.96
13.98


Ischia mediante spline cubica interpolante
latitudine
longitudine
40.77
40.76
40.75
40.74
40.73
40.72
40.71
40.7
40.69
13.84
13.86
13.88
13.9
13.92
13.94
13.96
13.98

Altri tipi di interpolazioni







Interpolazione Trigonometrica

… Fourier (DFT)

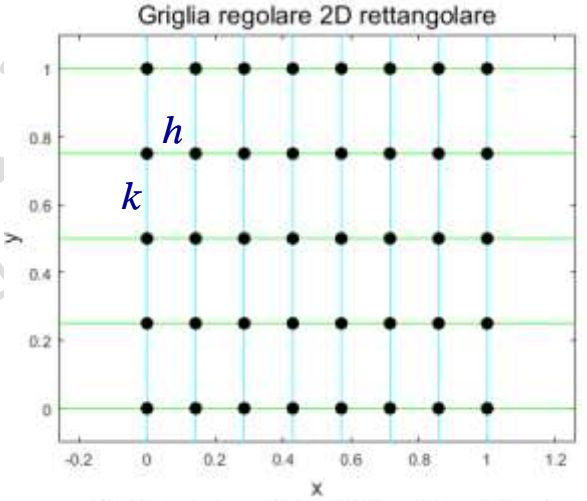
Interpolazione di superfici in MATLAB

Assegnati n punti 3D (x_i, y_i, z_i) si cerca una funzione f (appartenente ad una certa classe di funzioni) tale che:

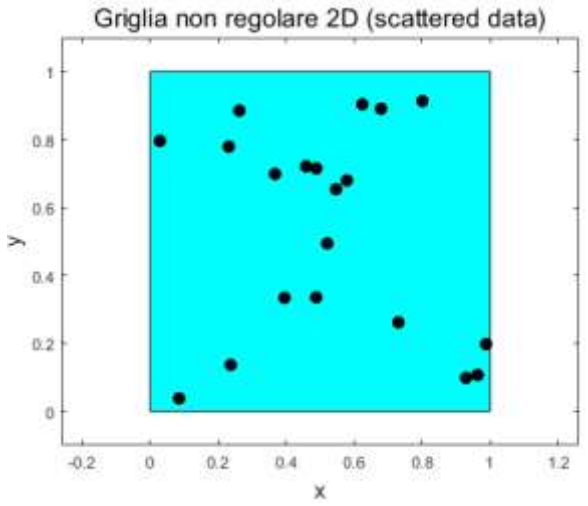
$$f(x_i, y_i) = z_i$$

Dominio (x_i, y_i)

1) Dati su griglia regolare 2D rettangolare o quadrata (gridded data)



2) Dati su griglia non regolare 2D (scattered data)



Interpolazione in più dimensioni con MATLAB

Grid Creation Functions	
meshgrid	**[X,Y]=meshgrid(x,y)** returns 2D grid coordinates based on the coordinates contained in vectors x and y. X is a matrix where each row is a copy of x, and Y is a matrix where each column is a copy of y. The grid represented by the coordinates X and Y has length(y) rows and length(x) columns. **[X,Y,Z]=meshgrid(x,y,z)** returns 3D grid coordinates defined by the vectors x, y, and z. The grid represented by X, Y, and Z has size length(y)-by-length(x)-by-length(z).
ndgrid	**[X1,X2,...,Xn]=ndgrid(x1,x2,...,xn)** replicates the grid vectors x1, x2, ..., xn to produce an n-dimensional full grid.
Gridded Data	
interp2	Interpolation for 2D gridded data in meshgrid format.
interp3	Interpolation for 3D gridded data in meshgrid format.
interpn	Interpolation for 1D, 2D, 3D, and N-D gridded data in ndgrid format.
griddedInterpolant	For interpolation of 1D, 2D, 3D, N-D gridded data set.
Scattered data	
griddata, griddatan, scatteredInterpolant	**zq=griddata(x,y,z,xq,yq,*method*)** fits a surface of the form z = f(x,y) to the scattered data in the vectors (x,y,z). The griddata function interpolates the surface at the query points specified by (xq,yq) and returns the interpolated values, zq. The surface always passes through the data points defined by x and y. *method*='linear','nearest','natural','cubic' are triangulation-based interpolation methods.
delaunay	**T=delaunay(P)** creates a 2D or 3D Delaunay triangulation from the points in a matrix P.
delaunayTriangulation	To create a delaunayTriangulation object, which allows a variety of topological and geometric queries. For example, locate a facet that contains a specific point.

Interpolazione di superfici

1) Dati su griglia regolare 2D (rettangolare o quadrata)

Le funzioni più usate sono i polinomi bilineari, del tipo:

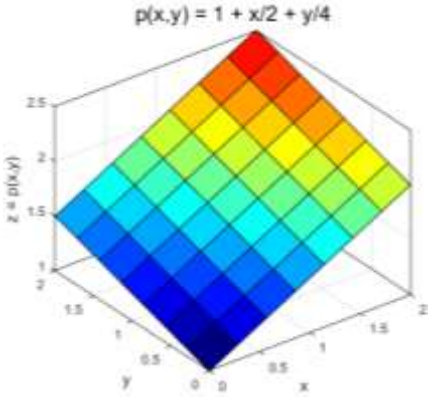
$$P_{BL}(x,y) = a_1 + a_2 x + a_3 y + a_4 xy$$ 4 coefficienti

Essi sono particolari polinomi di 2° grado, lineari rispetto alle singole variabili.

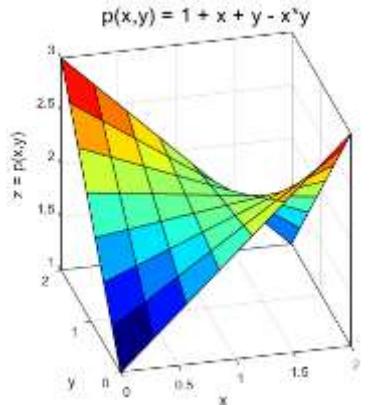
Come caso particolare si ritrovano i polinomi di 1° grado: 3 coefficienti

$$P_L(x,y) = a_1 + a_2 x + a_3 y$$

Il grafico di un polinomio di 1° grado è un piano:



Il grafico di un polinomio bilineare è una superficie:



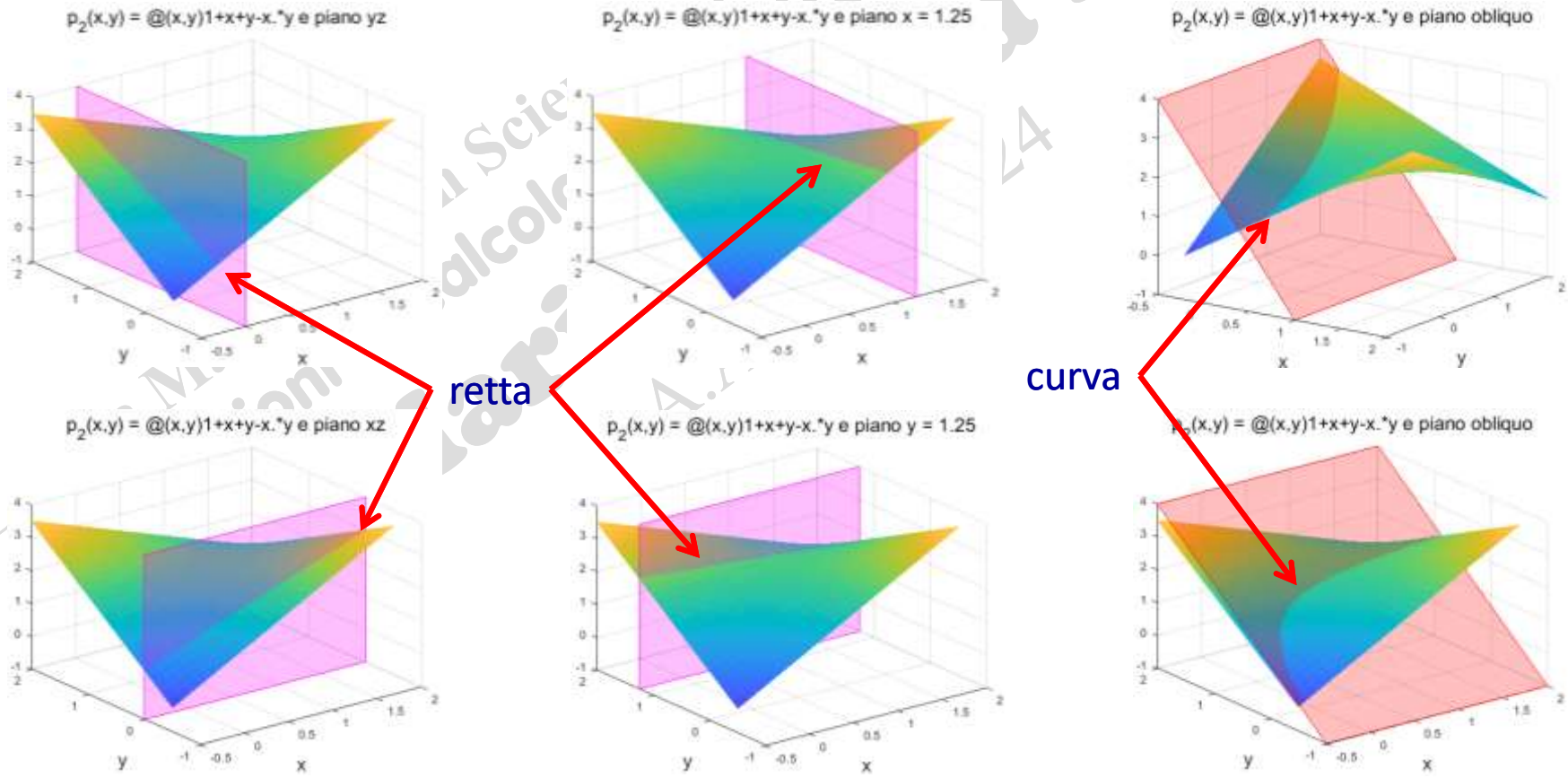
Polinomi bilineari

Proprietà

Se si interseca un polinomio bilineare:

$$P_{BL}(x,y) = a_1 + a_2 x + a_3 y + a_4 xy$$

con un piano parallelo al piano cartesiano (x,z), di equazione y=cost., oppure al piano cartesiano (y,z), di equazione x=cost., si ottiene come intersezione una retta (da qui il nome bilineare). Cosa succede con un piano di eq.: z=cost.?



Interpolazione 2D con polinomi bilineari su 4 punti

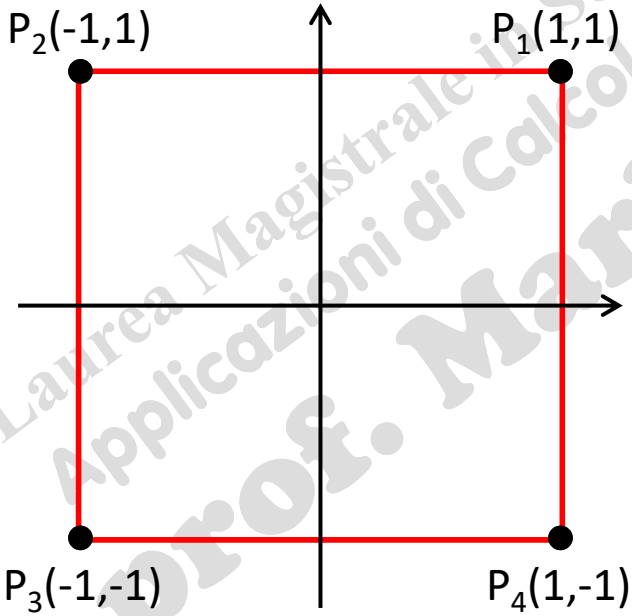
Per determinare i 4 coefficienti di un polinomio bilineare servono 4 punti che forniscano condizioni linearmente indipendenti affinché il sistema sia compatibile determinato: per $i=1, 2, 3, 4$

$$z_i = p(x_i, y_i) = a_1 + a_2 x_i + a_3 y_i + a_4 x_i y_i$$

Bisogna assegnare 4 punti 3D $P_i(x_i,y_i,z_i)$ ed imporre

$$p(x_i, y_i) = z_i$$

Esempio 1

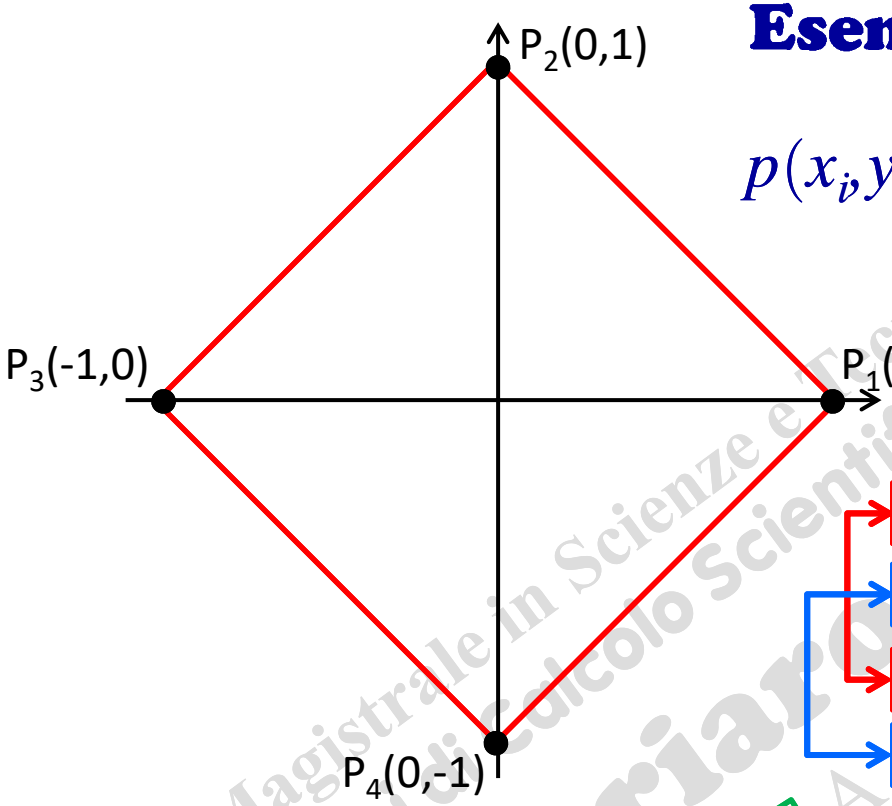


$$\begin{cases} a_1 + a_2 + a_3 + a_4 = z_1 \\ a_1 - a_2 + a_3 - a_4 = z_2 \\ a_1 - a_2 - a_3 + a_4 = z_3 \\ a_1 + a_2 - a_3 - a_4 = z_4 \end{cases}$$

Esiste un'unica soluzione

```
z=sym('z',[4 1]);
A=[1  1  1  1;
   1 -1  1 -1;
   1 -1 -1  1;
   1  1 -1 -1];
disp(det(A))
  -16
a=A\z
a =
z1/4 + z2/4 + z3/4 + z4/4
z1/4 - z2/4 - z3/4 + z4/4
z1/4 + z2/4 - z3/4 - z4/4
z1/4 - z2/4 + z3/4 - z4/4
```

Interpolazione 2D con polinomi bilineari su 4 punti



Esempio 2

$$p(x_i,y_i) = a_1 + a_2 x_i + a_3 y_i + a_4 x_i y_i = z_i$$

$$\begin{cases} a_1 + a_2 = z_1 \\ a_1 + a_3 = z_2 \\ a_1 - a_2 = z_3 \\ a_1 - a_3 = z_4 \end{cases}$$

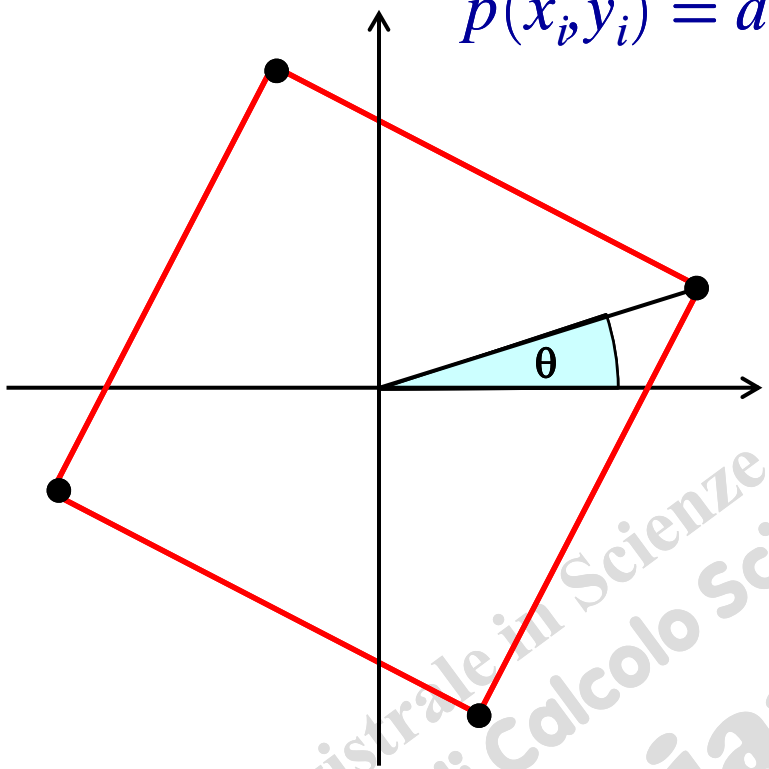
$$\begin{cases} 2a_1 = z_1 + z_3 \\ 2a_1 = z_2 + z_4 \end{cases}$$

```
z=sym('z',[4 1]);
A=[1  1  0  0;
   1  0  1  0;
   1 -1  0  0;
   1  0 -1  0];
disp(det(A))
  0
a=A\z
Warning: Solution does not
exist because the system is
inconsistent
a =
Inf
Inf
Inf
Inf
```

sistema incompatibile se $z_1 + z_3 \neq z_2 + z_4$

Interpolazione 2D con polinomi bilineari su 4 punti

$$p(x_i, y_i) = a_1 + a_2 x_i + a_3 y_i + a_4 x_i y_i = z_i$$

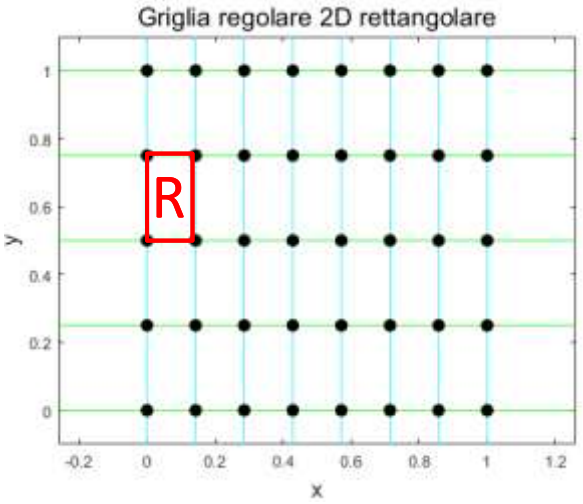


Si dimostra che il sistema ammette un'unica soluzione solo nei casi

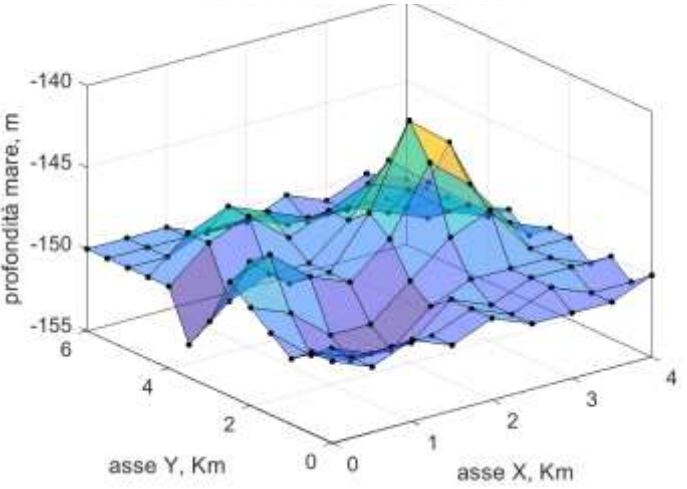
$$\theta \neq k\frac{\pi}{2}, \qquad k = 0, 1, \ldots$$

Nel caso di dati su griglia regolare 2D **esiste sempre** un unico polinomio bilineare interpolante i vertici del rettangolo R.

Interpolazione locale su ogni rettangolo: *polinomio bilineare a pezzi* (o *a tratti*).



dati di interpolazione `'seamount2.mat'`




```
ZZ=interp2(x,y,z,XX,YY,metodo);
```

metodo:

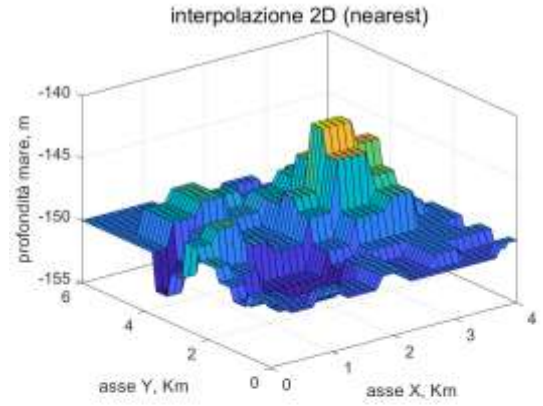
`'linear'` based on bilinear interpolation of the values at neighboring grid points in each respective dimension (default).

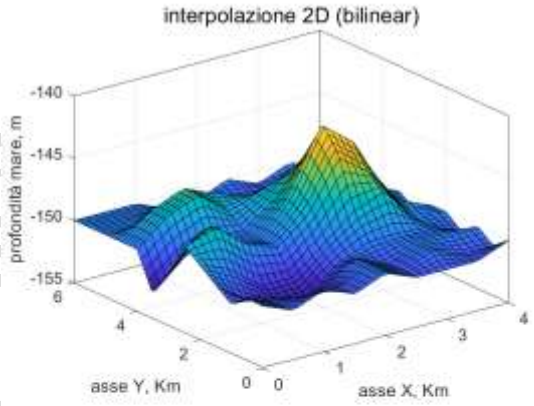
`'nearest'` the interpolated value at a query point is the value at the nearest sample grid point.

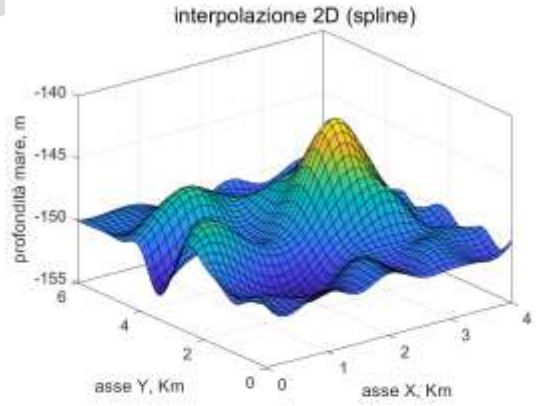
`'cubic'` based on a cubic interpolation of the values at neighboring grid points in each respective dimension.

`'spline'` based on a cubic spline with not-a-knot end conditions.

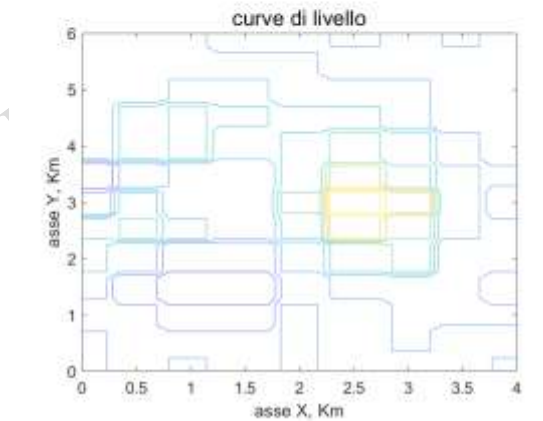
per infittire la griglia da `(x,y,z)` a `(XX,YY,ZZ)`



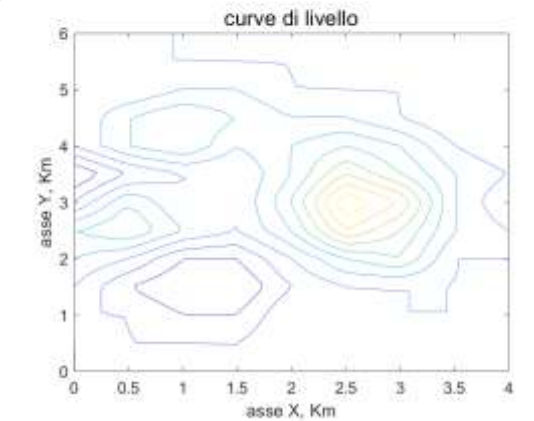




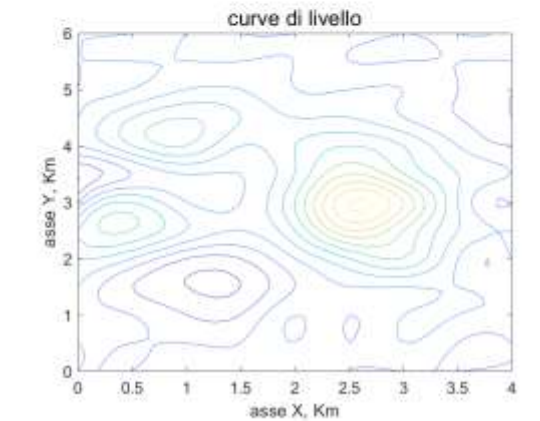
`interp2(x,y,z,XX,YY,'nearest')`



`interp2(x,y,z,XX,YY,'linear')`



`interp2(x,y,z,XX,YY,'spline')`



Ancora: per infittire la griglia da (x,y,z) a (XX,YY,ZZ)

crea oggetto interpolatore col metodo selezionato

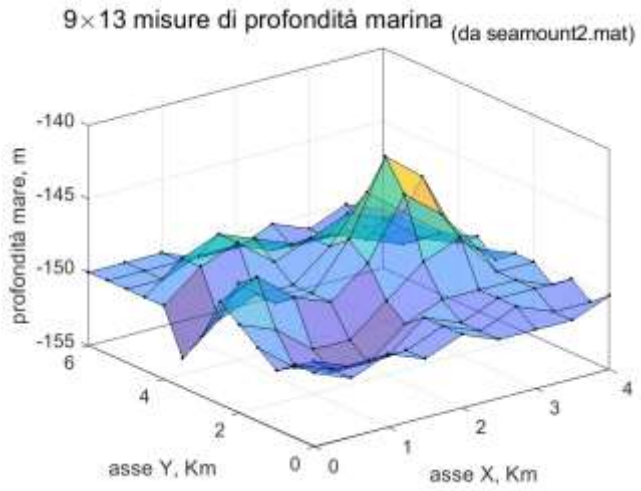
richiede griglia creata con **ndgrid()** invece di **meshgrid()**

```
F=griddedInterpolant(x,y,z,metodo);
```

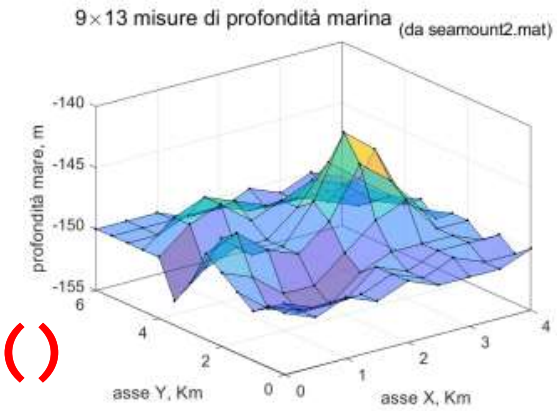
metodo:
'linear' based on bilinear interpolation of the values at neighboring grid points in each respective dimension (default).
'nearest' the interpolated value at a query point is the value at the nearest sample grid point.
'cubic' based on a cubic interpolation of the values at neighboring grid points in each respective dimension.
'spline' based on a cubic spline with not-a-knot end conditions.
...

```
load seamount2.mat % dati (x,y,z)
[X,Y]=ndgrid(x,y); z=z'; % griddedInterpolant richiede ndgrid
xx=linspace(min(x),max(x), 4*numel(x));
yy=linspace(min(y),max(y), 4*numel(y));
[XX,YY]=ndgrid(xx,yy);
method=…
% crea un oggetto interpolatore con i vari metodi
F=griddedInterpolant(X,Y,z,method);
ZZinterp=F(XX,YY); % interpola i dati
...
```

method= { 'linear' | 'nearest' | 'cubic' | 'makima' | 'spline' }

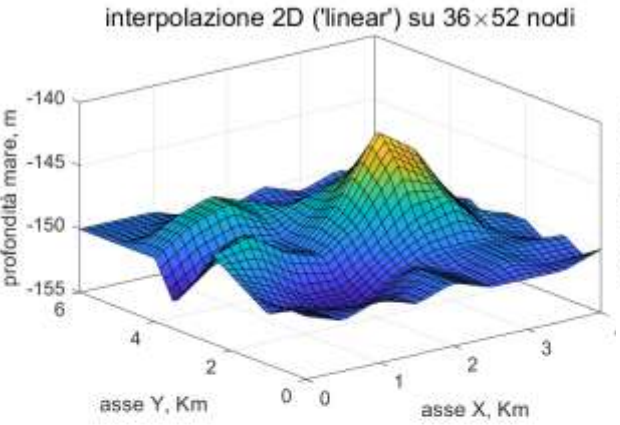


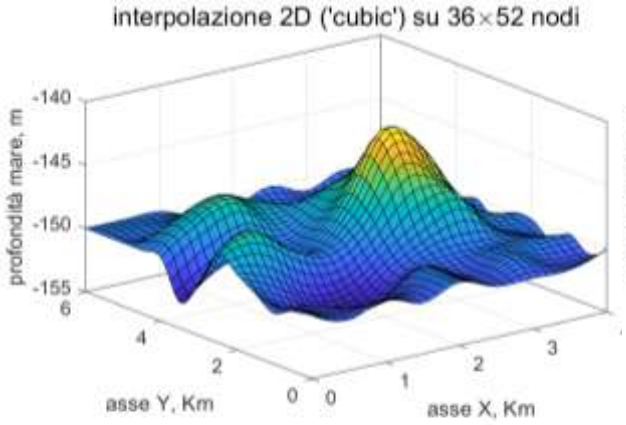
```
F=griddedInterpolant(x,y,z,metodo);
```

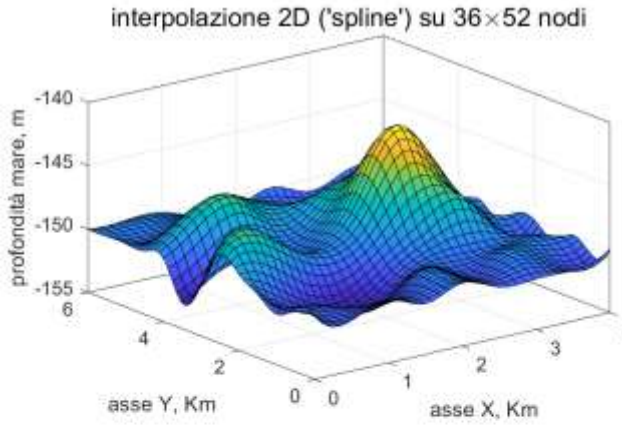


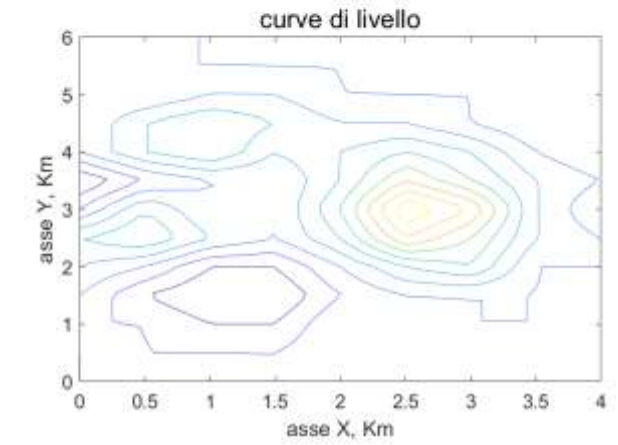
richiede griglia creata
con **ndgrid()**
invece di **meshgrid()**

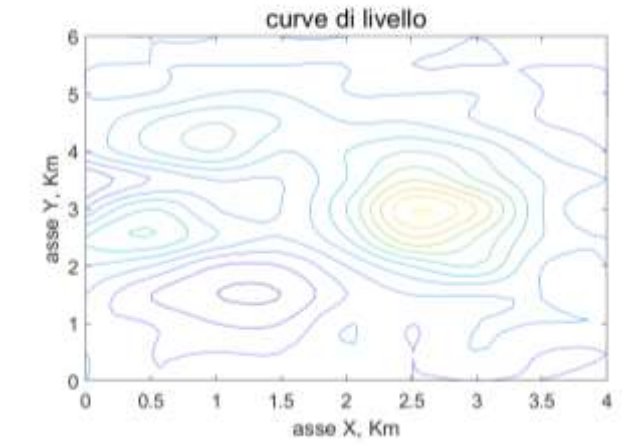
confrontare con **interp2()**

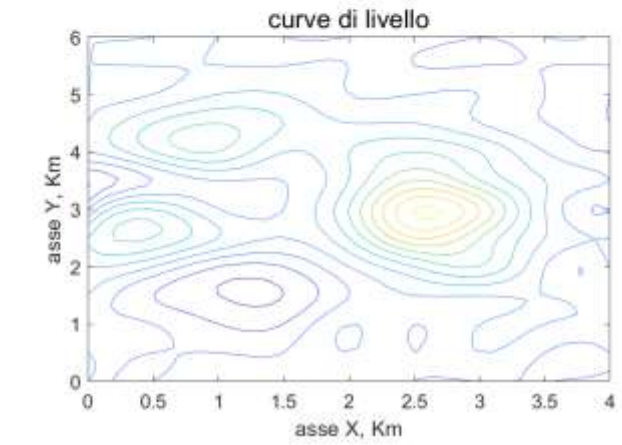












Richiami: differenza tra meshgrid() e ndgrid()

```
[x1,y1]=meshgrid(1:3,4:5)
x1 =
     1     2     3
     1     2     3
y1 =
     4     4     4
     5     5     5
[x2,y2]=ndgrid(1:3,4:5)
x2 =
     1     1
     2     2
     3     3
y2 =
     4     5
     4     5
     4     5
```

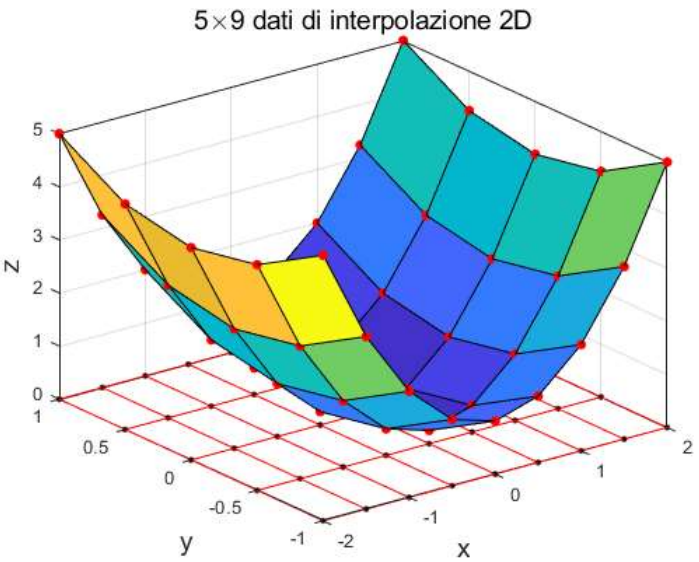
matrici trasposte delle precedenti

Come funziona l'interpolazione 2D di `interp2()`

Esempio

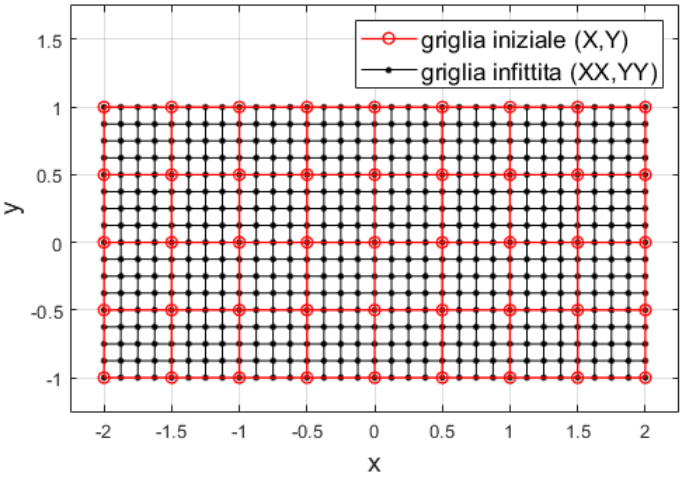
Dati (X,Y,Z) su griglia regolare 2D

```
xA=-2; xB=+2; Nx=9;  x=linspace(xA,xB,Nx);
yA=-1; yB=+1; Ny=5;  y=linspace(yA,yB,Ny);
[X,Y]=meshgrid(x,y); Z=X.^2 + Y.^2;
step=diff(x(1:2)); % stesso passo lungo gli assi x e y
surf(X,Y,Z); grid on; hold on; plot3(X,Y,Z,'.r')
```



infittisce la griglia

```
sstep=0.25*step; xx=xA:sstep:xB; yy=yA:sstep:yB;
[XX,YY]=meshgrid(xx,yy); ZZ=zeros(size(XX));
[m,n]=size(X); [mm,nn]=size(XX);
plot(XX,YY,'.k-',XX',YY','k', X,Y,'or-',X',Y','r')
```

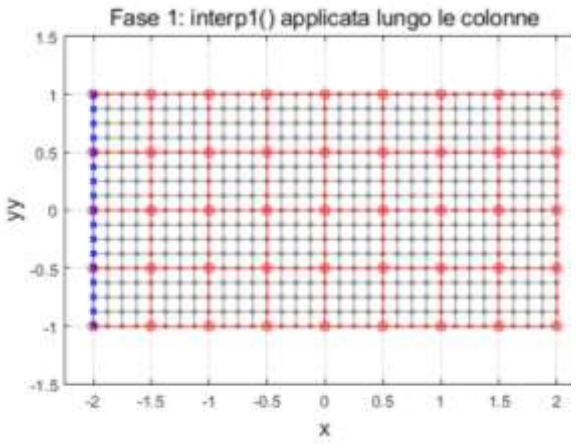


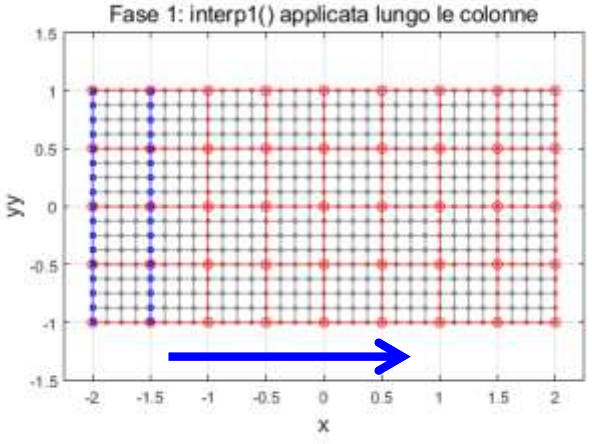
L'idea è quella di usare l'interpolazione 1D (**`interp1`**) una volta lungo la direzione dell'asse y ed una seconda volta lungo la direzione dell'asse x.

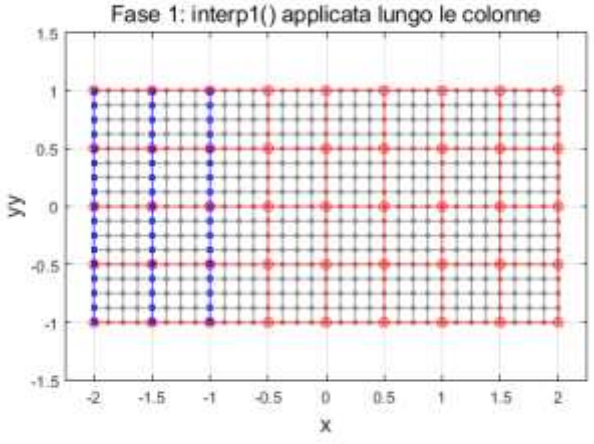
Esempio: interpolazione 2D mediante quella 1D

ogni interpolazione può essere eseguita in parallelo

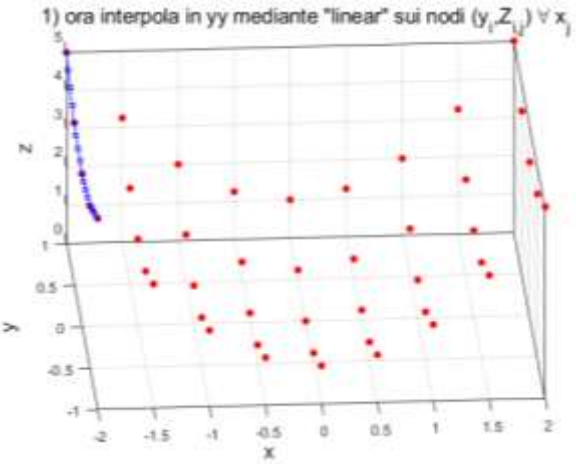
Fase 1

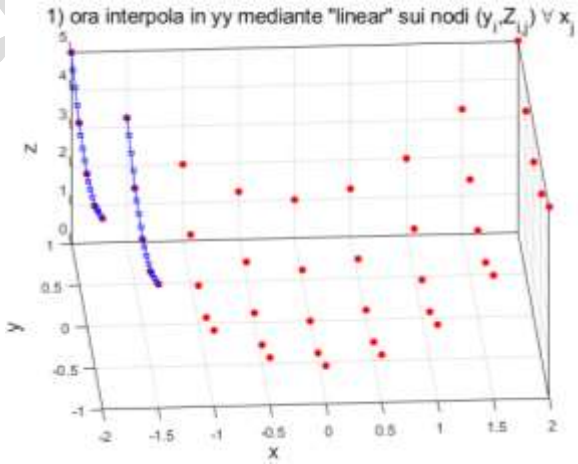


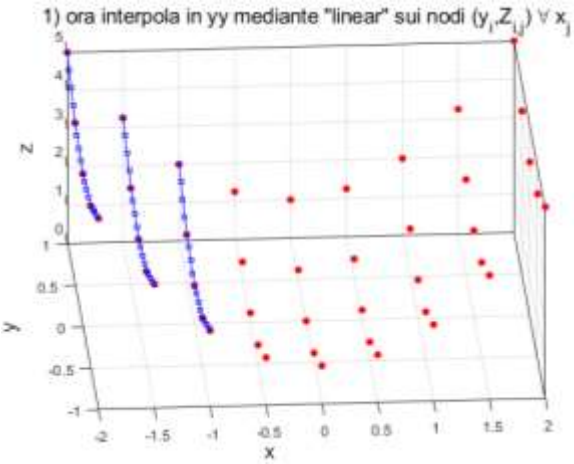




Per ogni x, usa **`interp1()`** sui nodi della griglia di partenza (y,z), lungo le sue colonne, per ottenere le quote dei punti con le ordinate della griglia infittita (x,yy,zz)

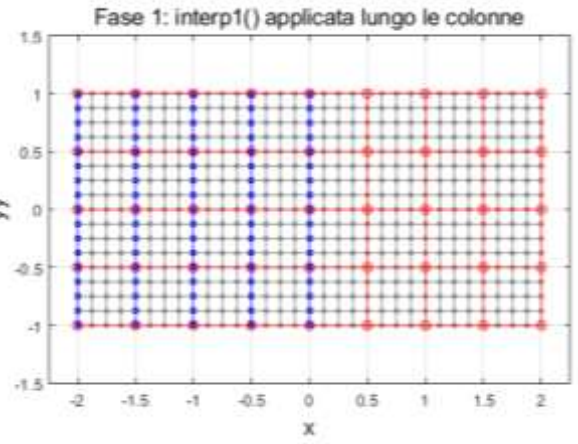


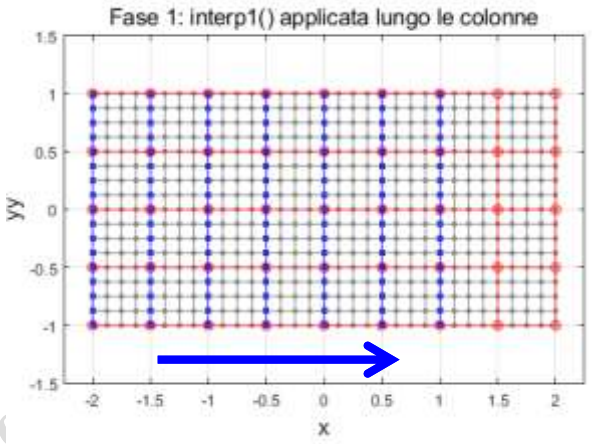


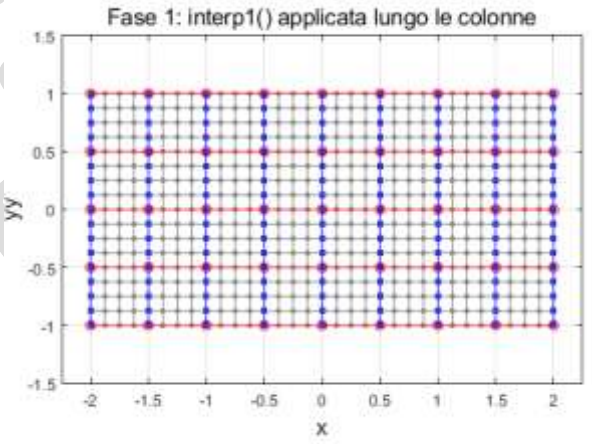


Esempio: interpolazione 2D mediante quella 1D

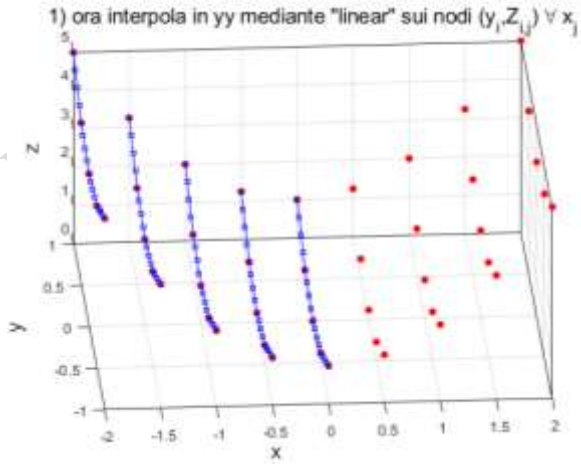
Fase 1

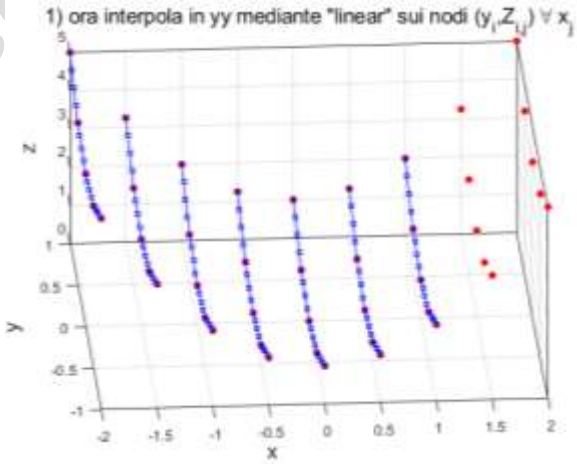


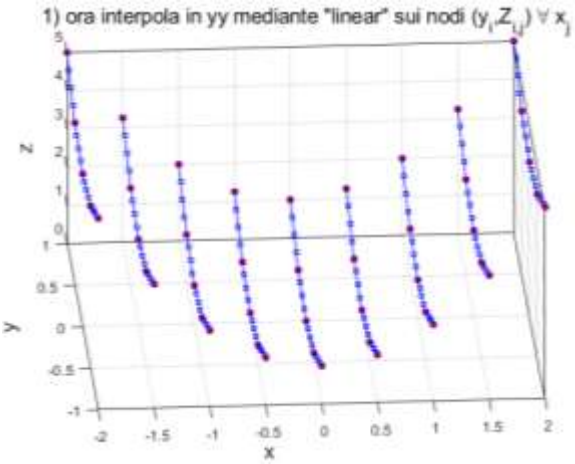




Per ogni x, usa **`interp1()`** sui nodi della griglia di partenza (y,z), lungo le sue colonne, per ottenere le quote dei punti con le ordinate della griglia infittita (x,yy,zz)

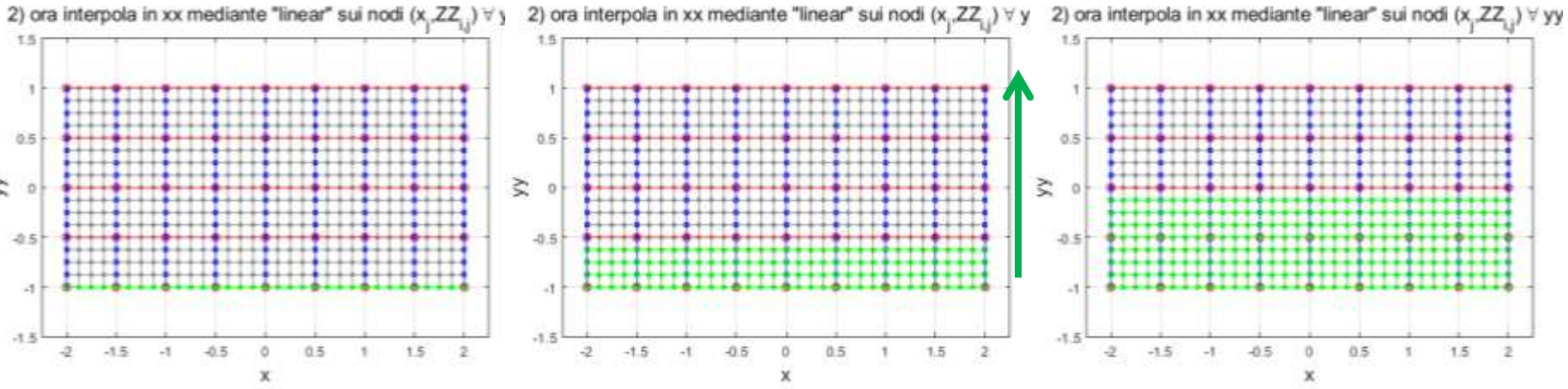




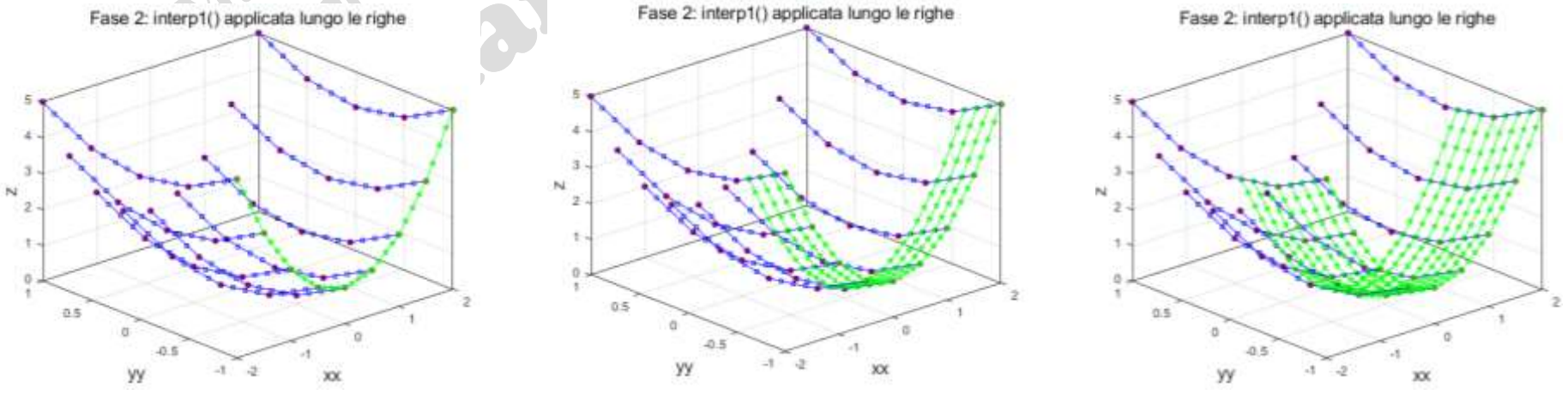


Esempio: interpolazione 2D mediante quella 1D

Fase 2

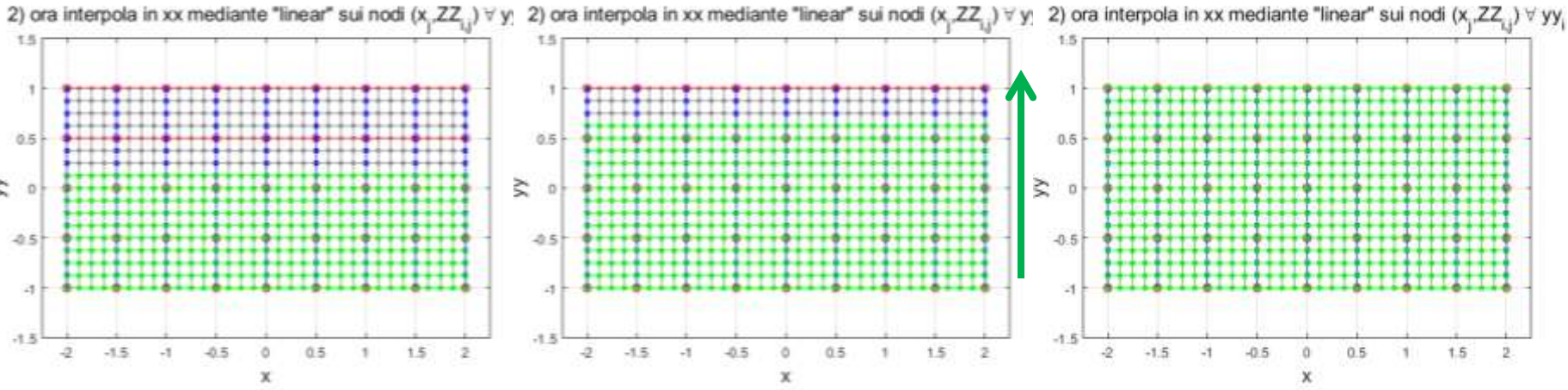


Per ogni yy, usa **`interp1()`** sui nodi della griglia parzialmente infittita (x,zz), lungo le sue righe, per ottenere le quote dei punti con le ordinate della griglia infittita (xx,yy,zzz)

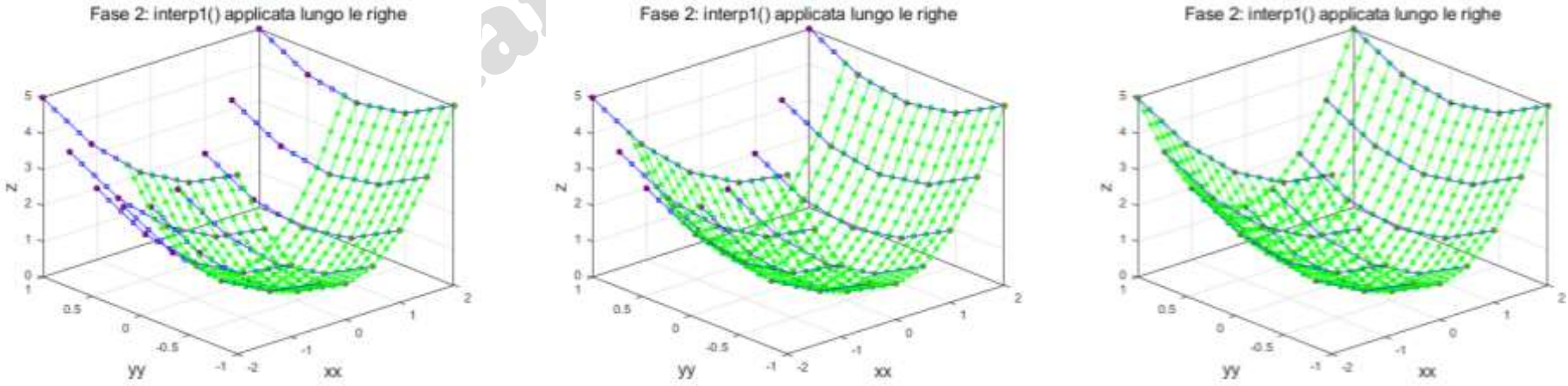


Esempio: interpolazione 2D mediante quella 1D

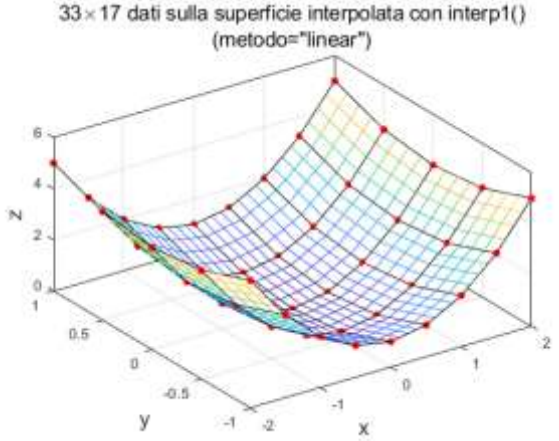
Fase 2

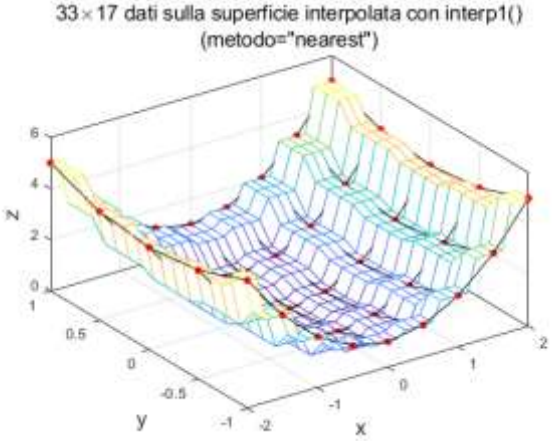


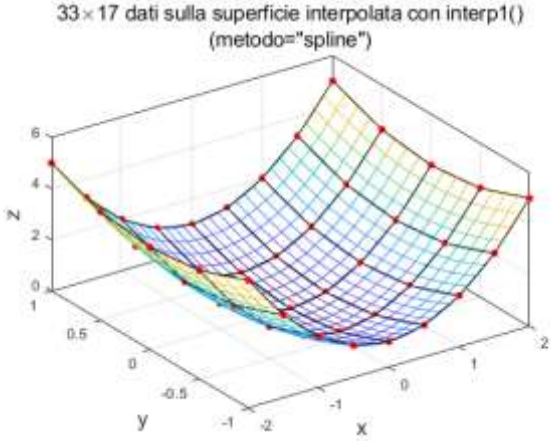
Per ogni yy, usa **`interp1()`** sui nodi della griglia parzialmente infittita (x,zz), lungo le sue righe, per ottenere le quote dei punti con le ordinate della griglia infittita (xx,yy,zzz)

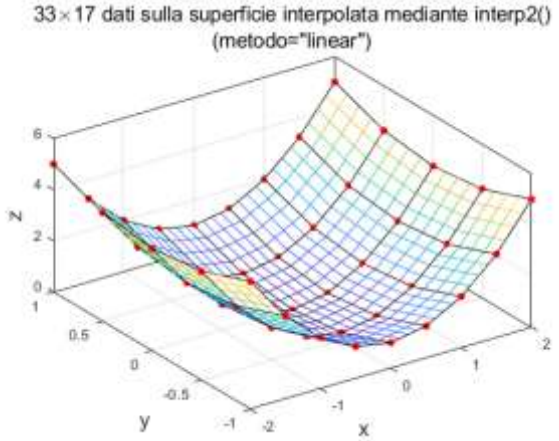


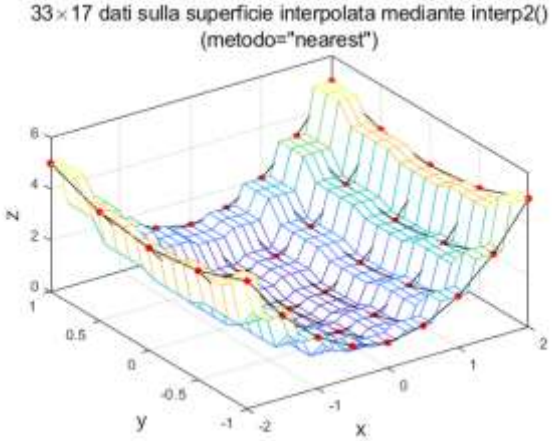
Esempio: interpolazione 2D mediante quella 1D

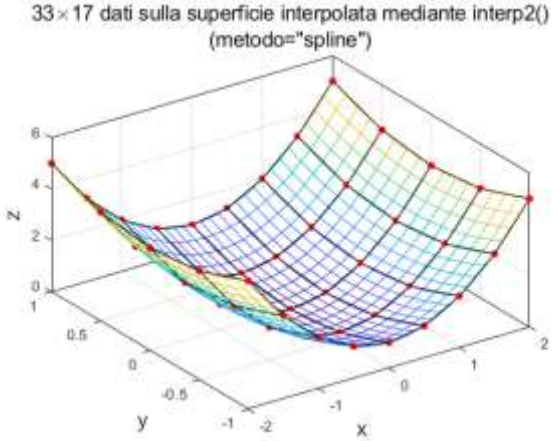


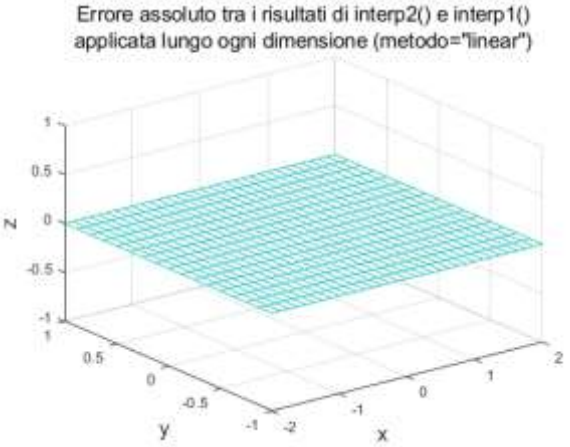


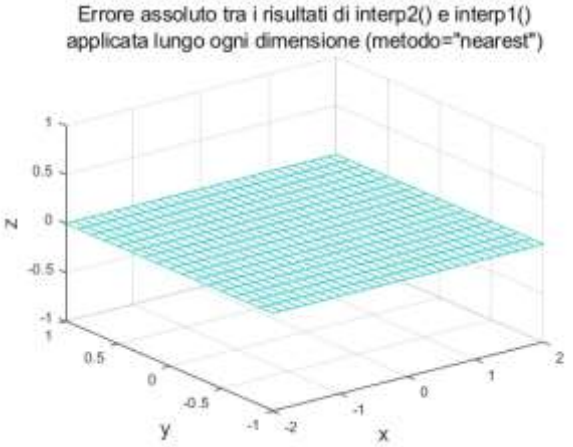


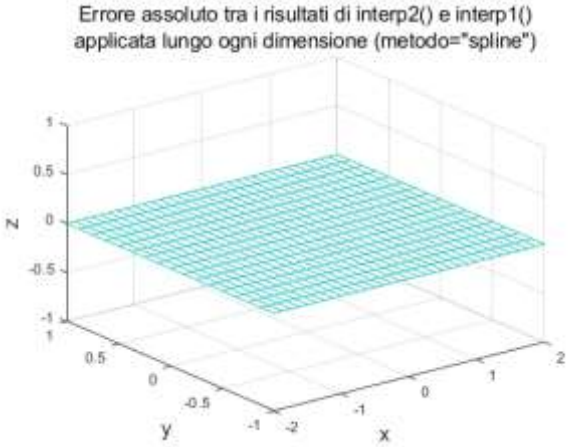












Interpolazione di superfici

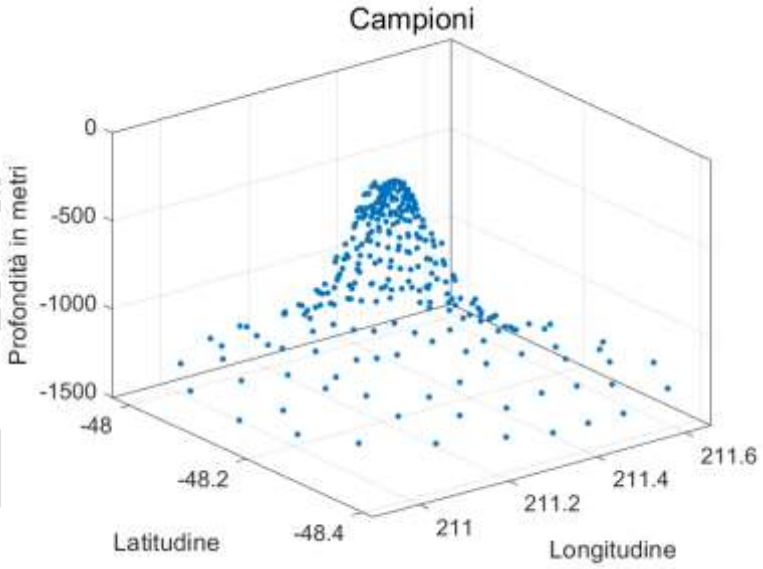
2) Dati su griglia non regolare 2D (scattered data)

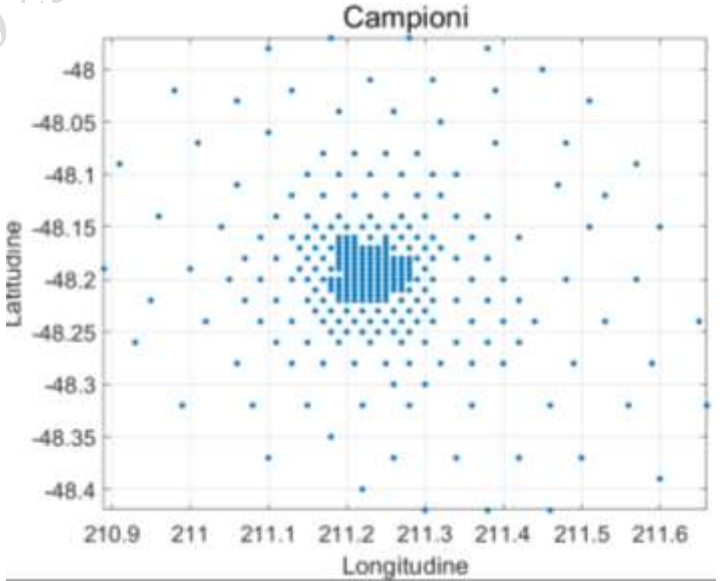
Esempio: `griddata` (lenta per più interpolazioni)

```
load seamount; z=0.3048*z; plot3(x,y,z,'.','markersize',10); grid on; box on
xlabel('Longitudine'); ylabel('Latitudine'); zlabel('Profondità in metri')
title('Campioni','FontWeight','normal','FontSize',16)
[xnew,ynew] = meshgrid(210.8:0.01:211.8, -48.5:0.01:-47.9);
method=...;  znew=griddata(x,y,z,xnew,ynew,method);
figure(2); [c,h]=contour(xnew,ynew,znew); clabel(c,h);
xlabel('Longitudine'); ylabel('Latitudine')
title("Contour plot ('" + method + "')",'FontWeight','normal','FontSize',16)
figure(3); surf(xnew,ynew,znew); hold on; grid on; box on
plot3(x,y,z,'.r','markersize',8)
xlabel('Longitudine'); ylabel('Latitudine'); zlabel('Profondità in metri')
title("Metodo di interpolazione: '" + method + "'",'FontWeight','normal','FontSize',16)
```

method= { 'linear' | 'nearest' | 'natural' | 'cubic'

esempio_griddata.m

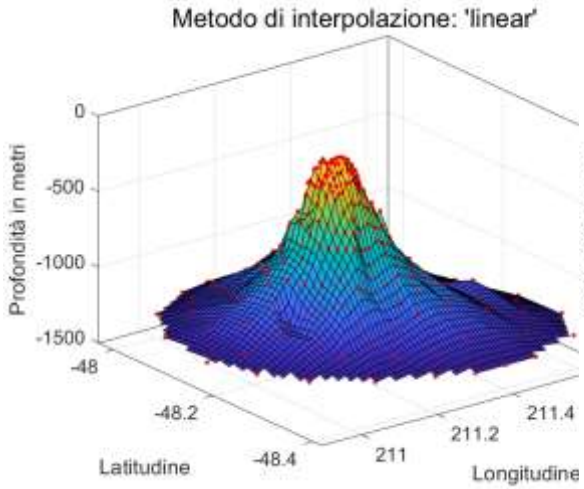


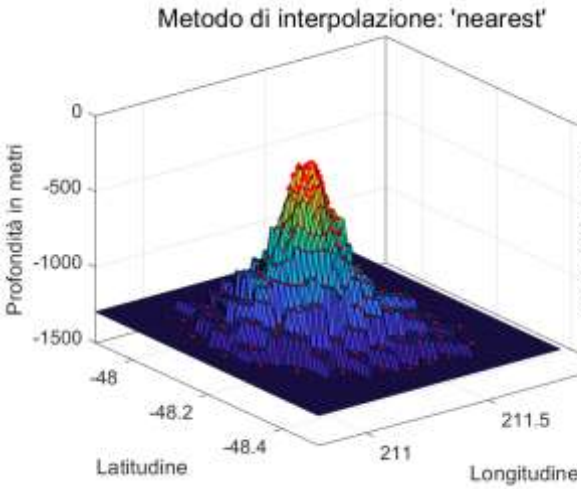


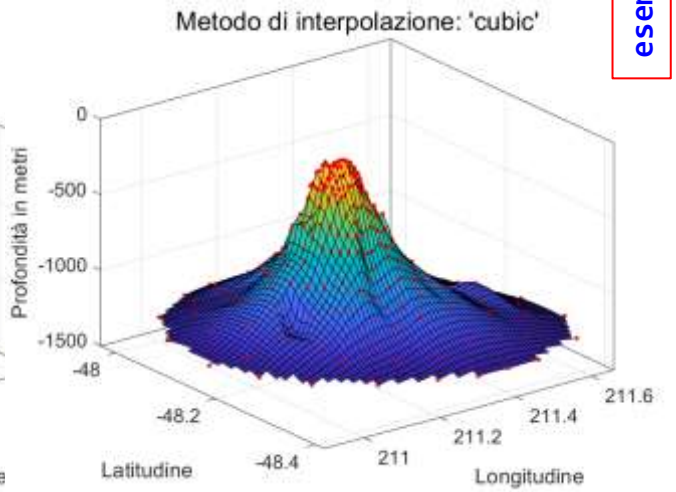
Interpolazione di superfici

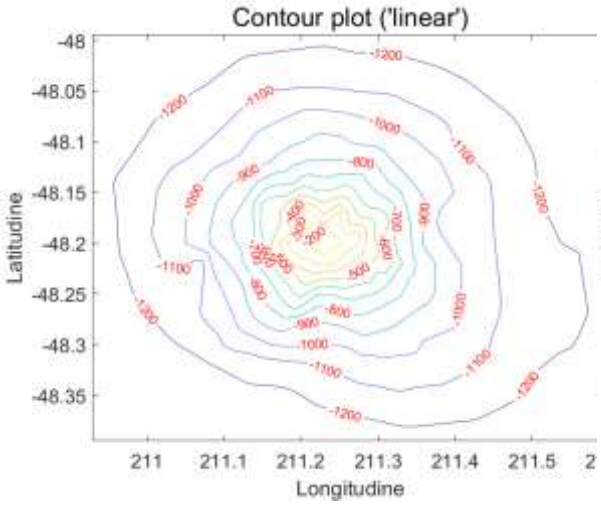
2) Dati su griglia non regolare 2D (scattered data)

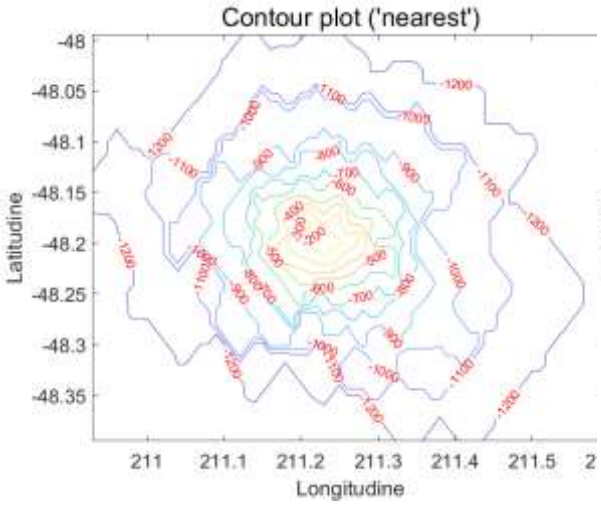
`griddata()`

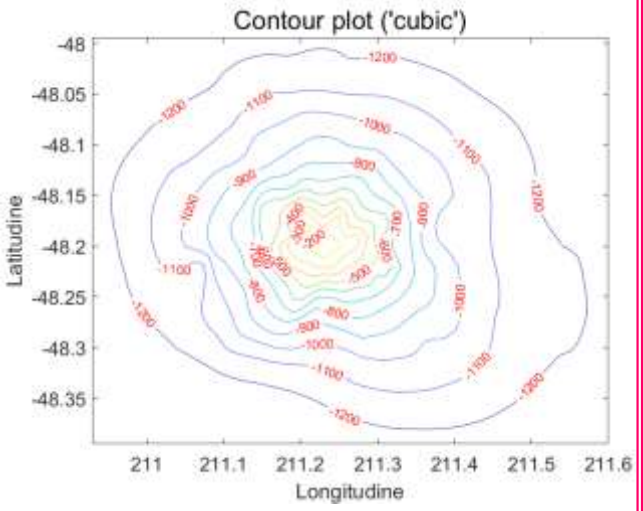






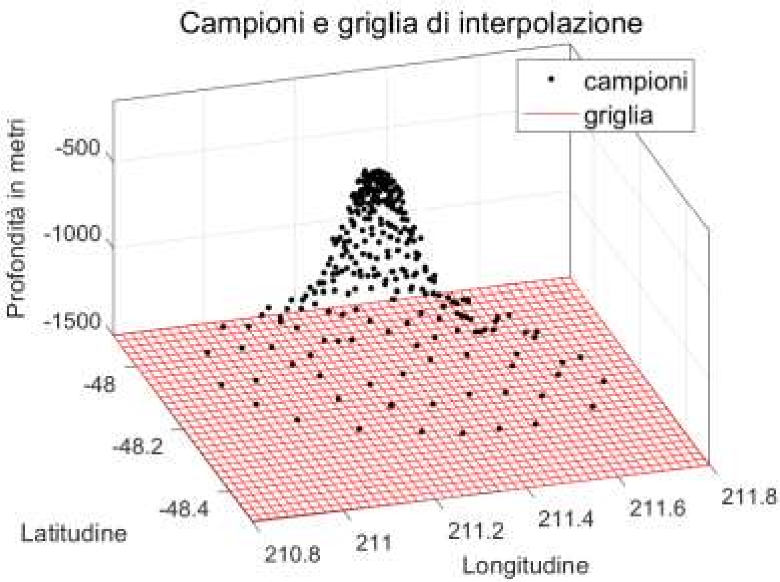


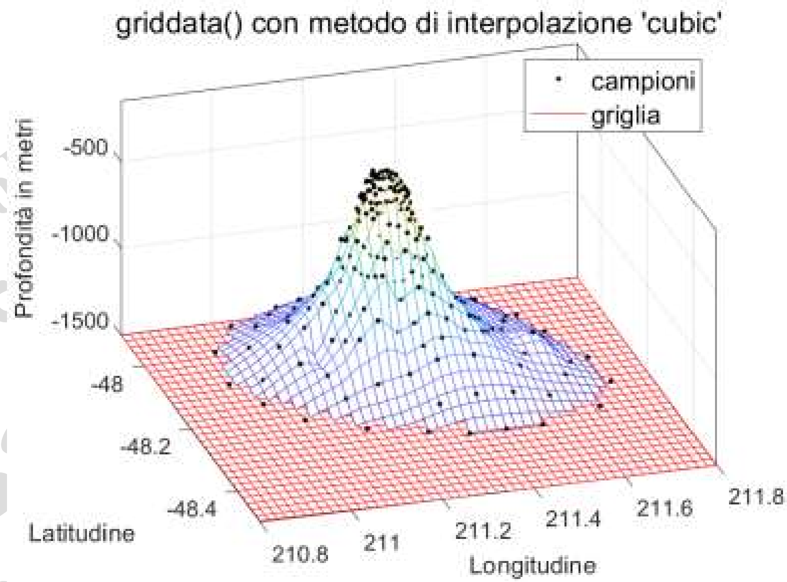




Interpolazione di superfici

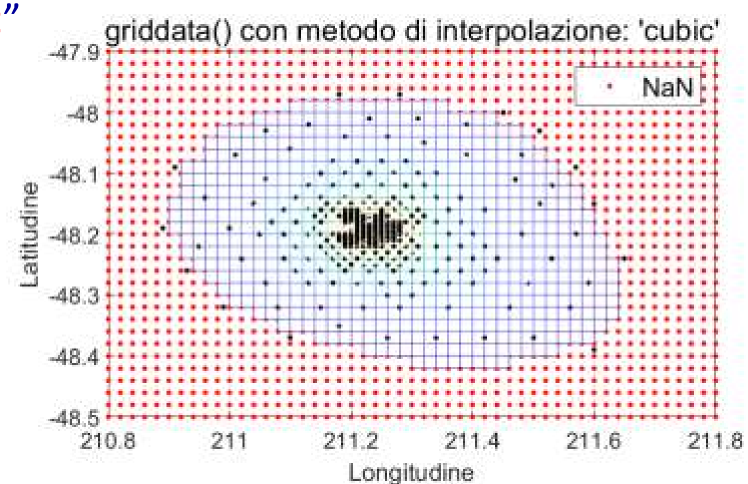
2) Dati su griglia non regolare 2D (scattered data)





`griddata()` restituisce valori che non ricoprono tutta la griglia
tranne che col metodo "`nearest`"

Le quote dei punti mancanti contengono il valore NaN che nei grafici non viene visualizzato



Interpolazione di superfici

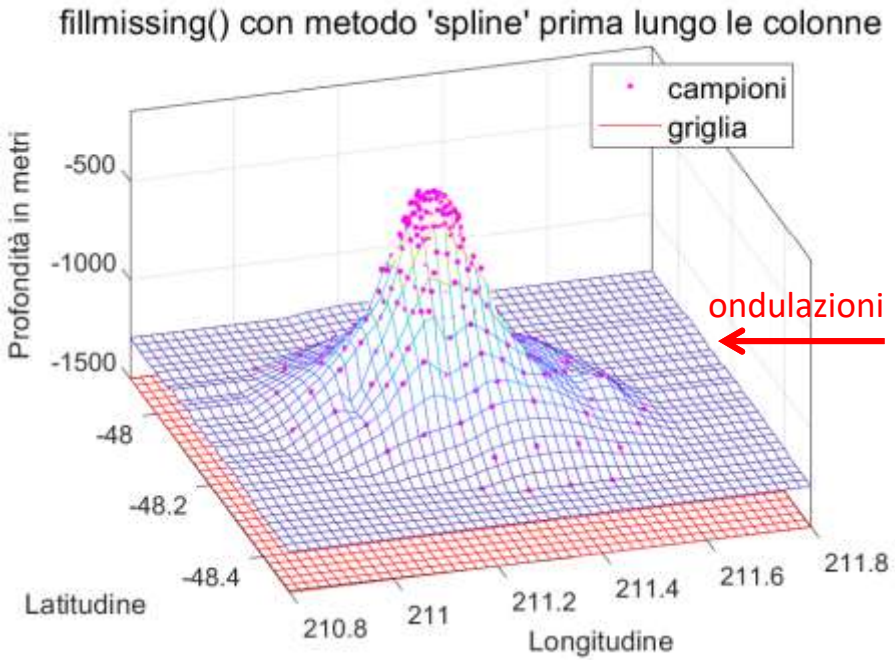
2) Dati su griglia non regolare 2D (scattered data)

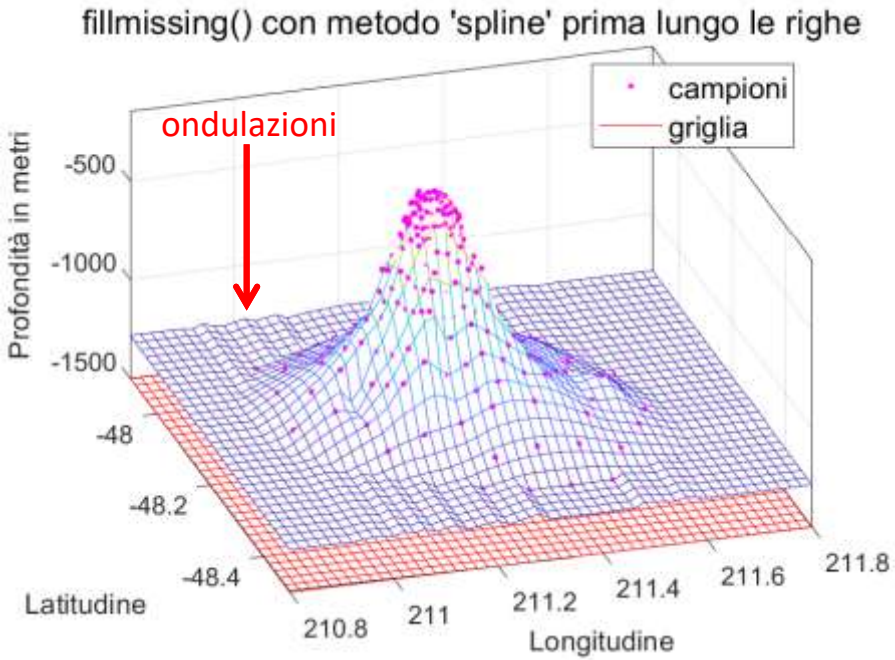
Per riempire i NaN con qualche valore usare la funzione `fillmissing()` che ha molte opzioni d'uso.

Esempio

```
load seamount; ... znew=griddata(x,y,z,xnew,ynew,method);
Znew=znew;
Znew=fillmissing(Znew,'spline',2,'EndValues','nearest');
Znew=fillmissing(Znew,'spline',1,'EndValues','nearest');
```

riempie lungo una dim prima:
lungo le colonne
poi: lungo le righe





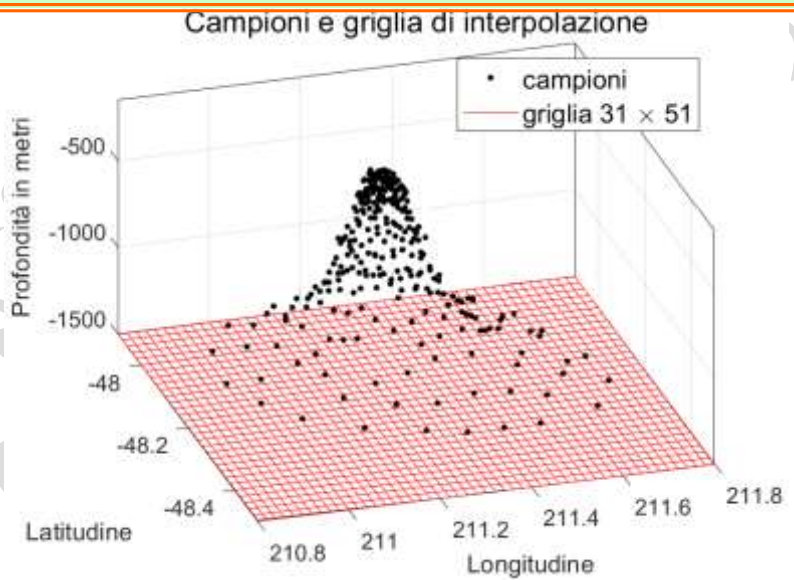
Interpolazione di superfici

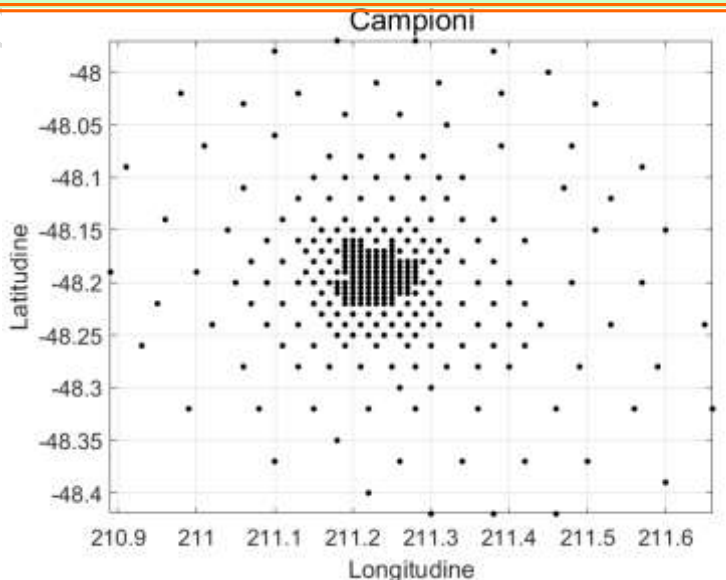
2) Dati su griglia non regolare 2D (scattered data)

Esempio: `scatteredInterpolant` (veloce per più interpolazioni)

```
load seamount; z=0.3048*z;
plot3(x,y,z,'.','markersize',10); grid on; box on
xlabel('Longitudine'); ylabel('Latitudine'); zlabel('Profondità in metri')
title('Campioni','FontWeight','normal','FontSize',16)
[xnew,ynew] = meshgrid(210.8:0.02:211.8, -48.5:0.02:-47.9);
method=...; F=scatteredInterpolant(x,y,z, method);
znew=F(xnew,ynew);
figure; [c,h]=contour(xnew,ynew,znew); axis tight; clabel(c,h);
xlabel('Longitudine'); ylabel('Latitudine')
title("Contour plot ('" + method + "')",'FontWeight','normal','FontSize',16)
figure; mesh(xnew,ynew,znew); hold on; grid on; box on; plot3(x,y,z,'.r','markersize',8)
xlabel('Longitudine'); ylabel('Latitudine'); zlabel('Profondità in metri')
title("Metodo di interpolazione: '" + method + "'",'FontWeight','normal','FontSize',16)
```

method= { 'linear' | 'nearest' | 'natural' }

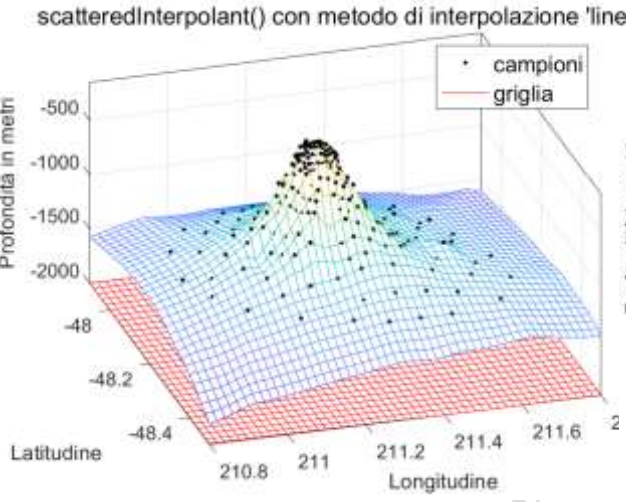


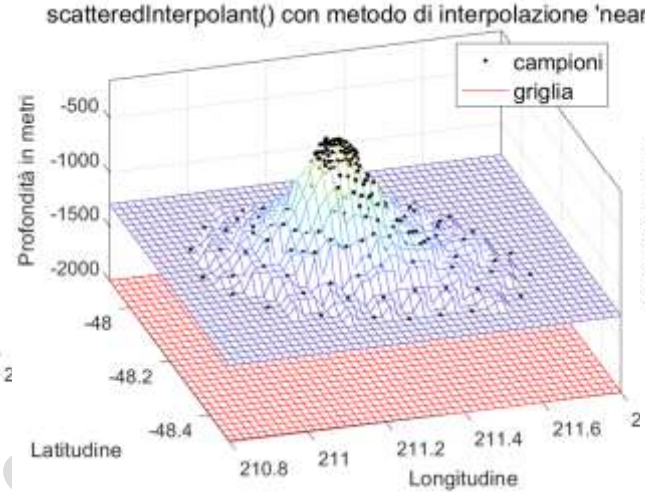


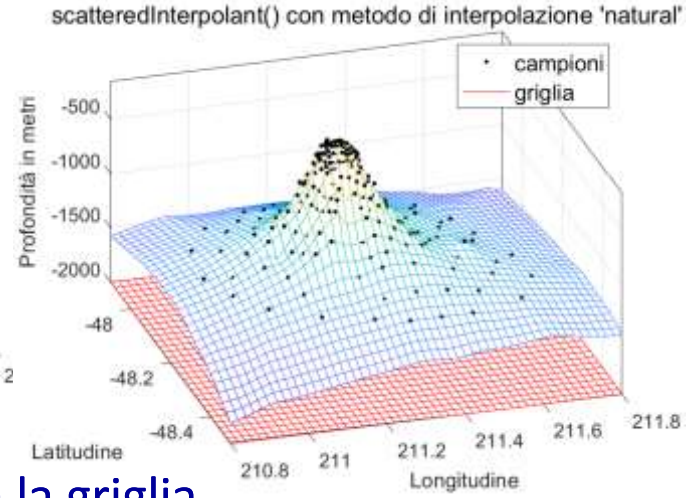
Interpolazione di superfici

2) Dati su griglia non regolare 2D (scattered data)

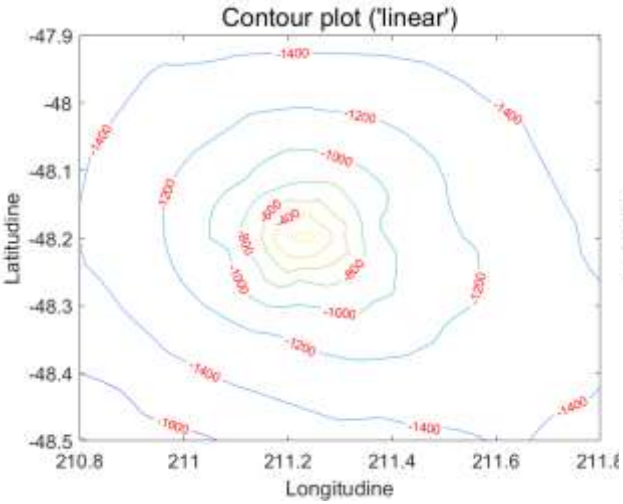
`scatteredInterpolant()`

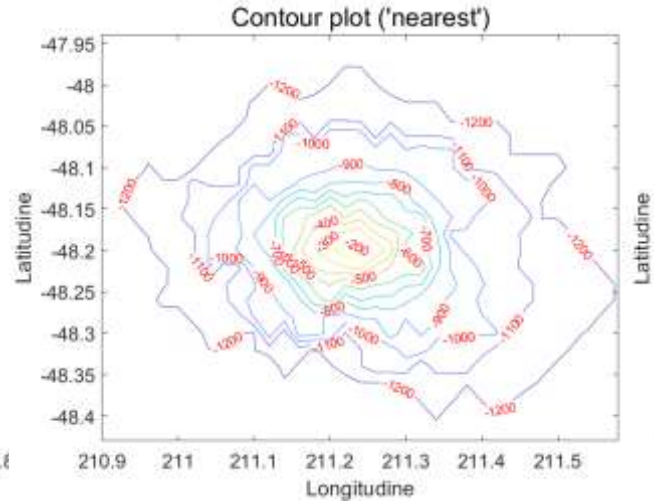


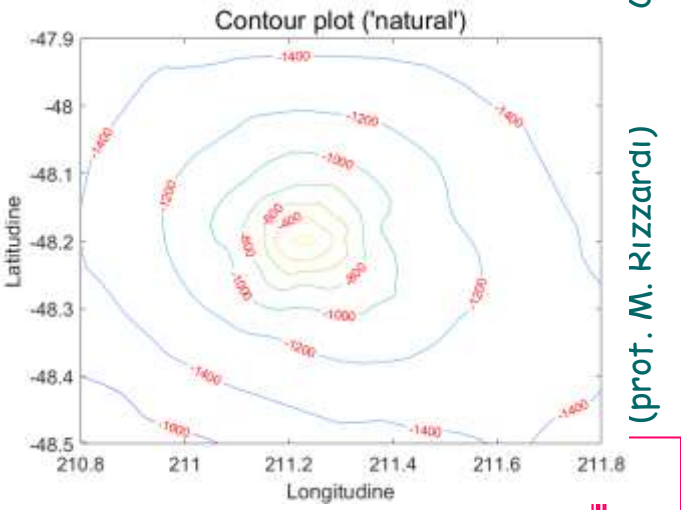




`scatteredInterpolant()` ricopre tutta la griglia





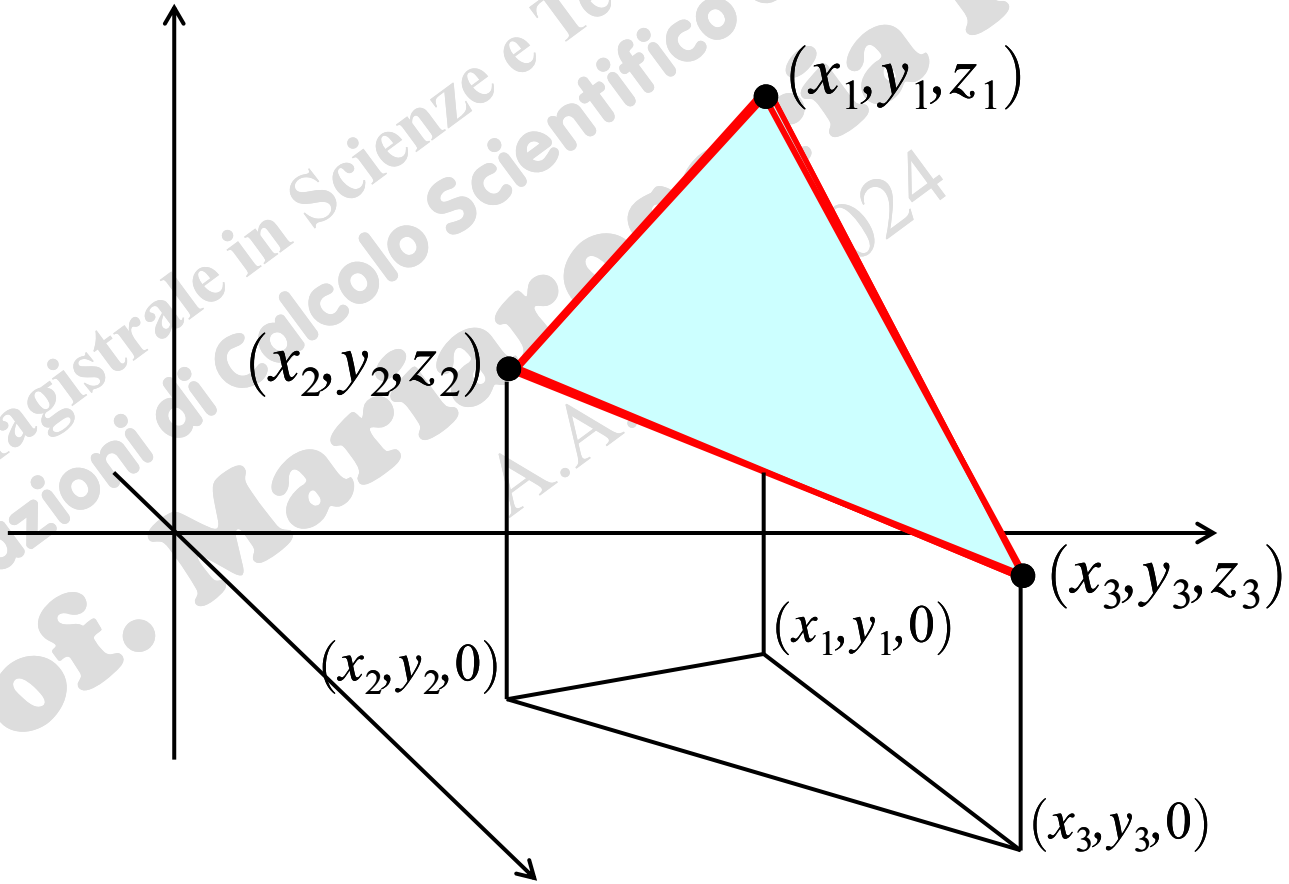


Interpolazione 2D con polinomi lineari su 3 punti

$$p(x_i, y_i) = a_1 + a_2 x_i + a_3 y_i = z_i$$

Assegnati 3 punti non allineati (i vertici di un triangolo), esiste un unico polinomio lineare interpolante.

3 punti non allineati dello spazio 3D individuano univocamente un piano: interpolazione locale su ogni triangolo (*polinomio lineare a pezzi*).

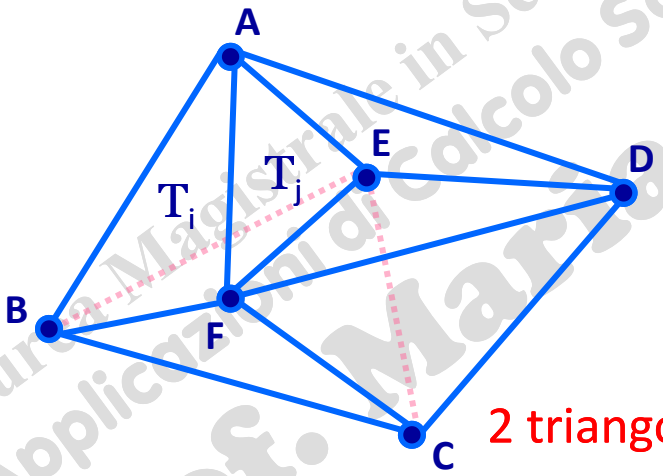


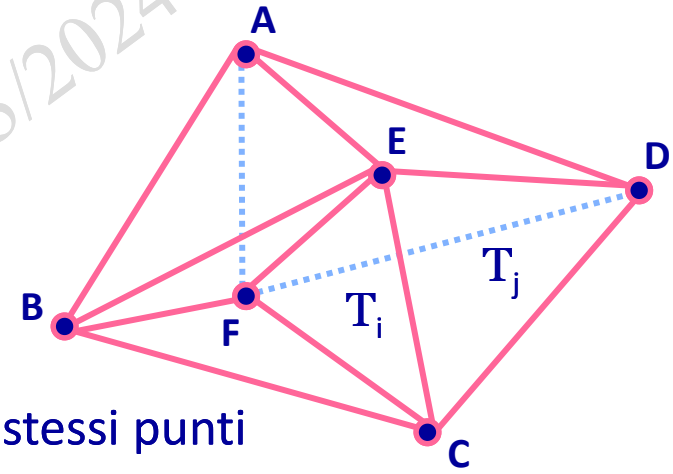
Triangolazione di un insieme di punti

Una triangolazione di un insieme finito $\mathbf{V}$ di punti del piano (nodi) è una collezione finita di triangoli (il triangolo è una figura convessa), i cui vertici sono i punti di $\mathbf{V}$, tale che l'unione di tali triangoli è il convex hull dei punti. Per ogni coppia di triangoli T_i e T_j deve verificarsi una, ed una sola, delle seguenti situazioni:

- T_i e T_j sono disgiunti ($T_i \cap T_j = \varnothing$);
- T_i e T_j hanno in comune un solo vertice ($T_i \cap T_j = P_k$);
- T_i e T_j hanno in comune due vertici ed il lato che li connette ($T_i \cap T_j = P_h P_k$).

Problema: la triangolazione in generale **non** è unica!





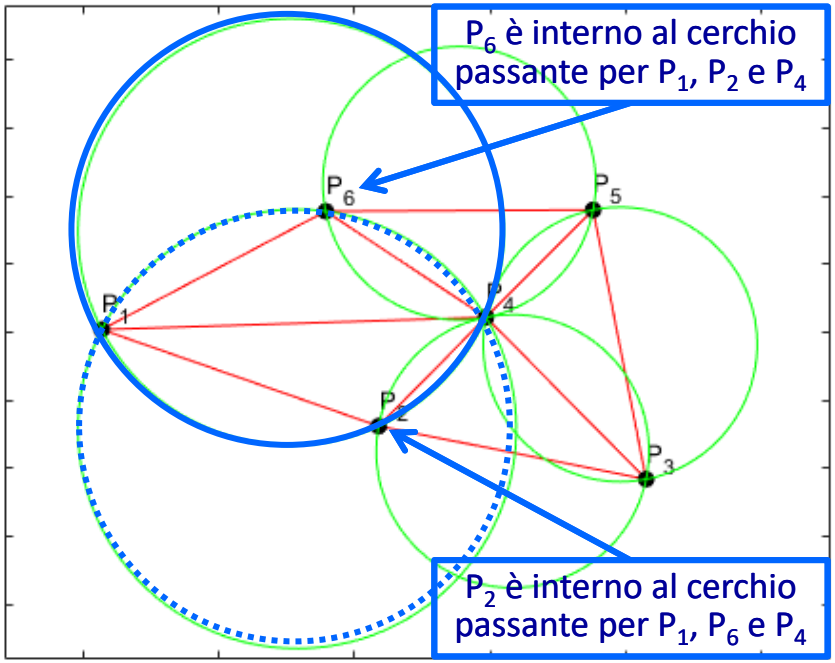
2 triangolazioni sugli stessi punti

In 3D il triangolo è sostituito dal tetraedro; 

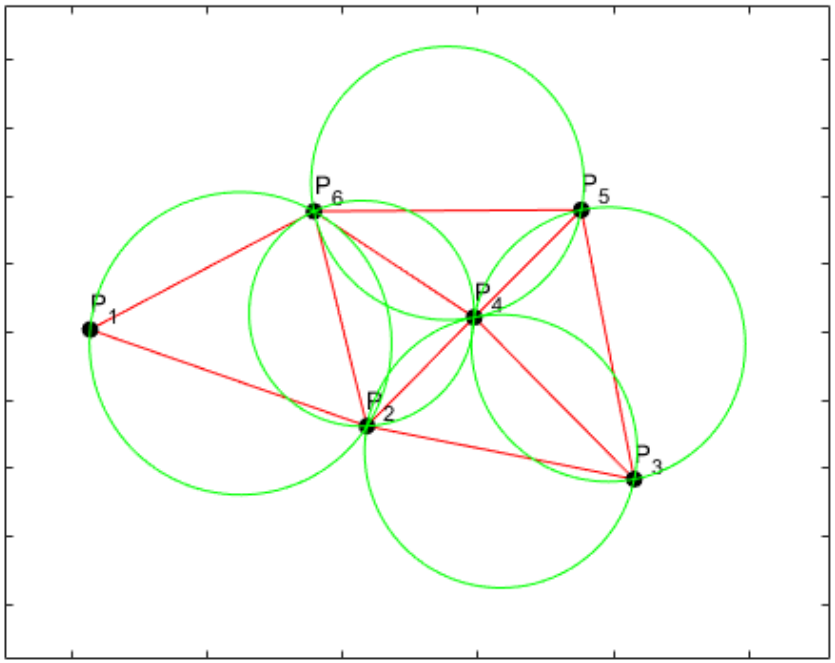
in n dimensioni si parla di simplesso.

Triangolazione di Delaunay

Una triangolazione di un insieme finito **V** di punti del piano (nodi) è detta *triangolazione di Delaunay* se il cerchio circoscritto ad ogni triangolo è vuoto, cioè se nessun punto di **V** (oltre i 3 vertici) vi giace all'interno.



Non è una triangolazione di Delaunay



È una triangolazione di Delaunay

- Ogni insieme di punti non collineari ha un'unica triangolazione di Delaunay.
- Ogni triangolazione di Delaunay massimizza il più piccolo angolo interno tra tutte le triangolazioni possibili.
- La triangolazione di Delaunay è il "duale" di un'altra costruzione geometrica nota come Diagramma di Voronoi.

Triangolazione di Delaunay in MATLAB

```
P=[ 14, 202;  % punto 1
   218, 131;  % punto 2
   415,  92;  % punto 3
   297, 211;  % punto 4
   376, 290;  % punto 5
   179, 289]; % punto 6
Np=size(P,1); X=P(:,1); Y=P(:,2);
T=delaunay(X,Y); % triangolazione
trep=triangulation(T,X,Y)
```

`trep` oggetto triangolazione

```
trep =
  triangulation with properties:
             Points: [6×2 double]
   ConnectivityList: [5×3 double]
```

```
[cc,rr]=circumcenter(trep)
figure; axis equal; hold on
t=linspace(-pi,pi,201);
for k=1:numel(rr)
    plot(cc(k,1),cc(k,2),'+b')
    plot(cc(k,1)+rr(k)*cos(t), ...
         cc(k,2)+rr(k)*sin(t),'c')
end
triplot(T,X,Y,'Color','r')
plot(X,Y,'ok','MarkerFaceColor','k')
```

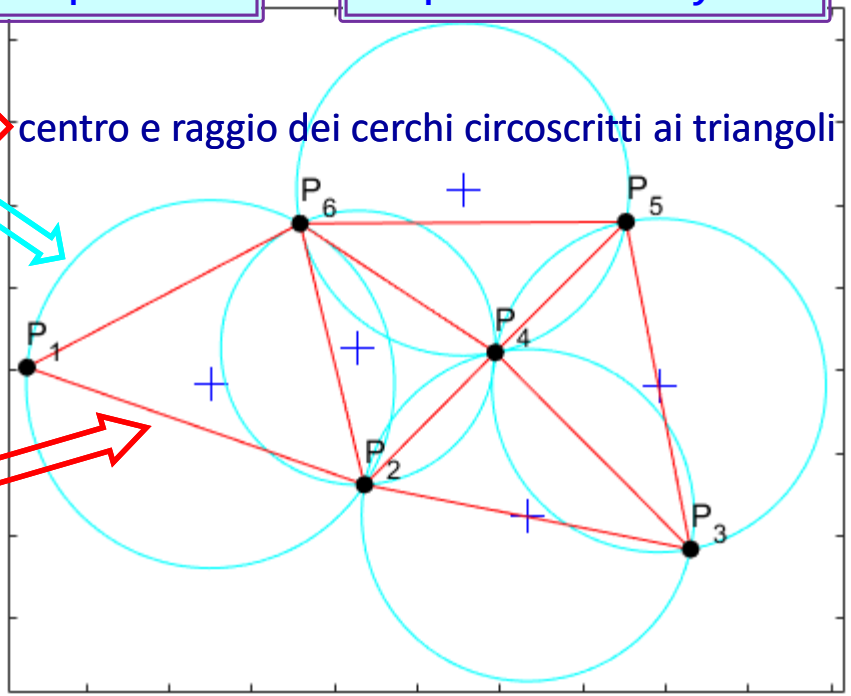
```
T=delaunay(x,y);
```

`T` — indici dei vertici di ogni triangolo

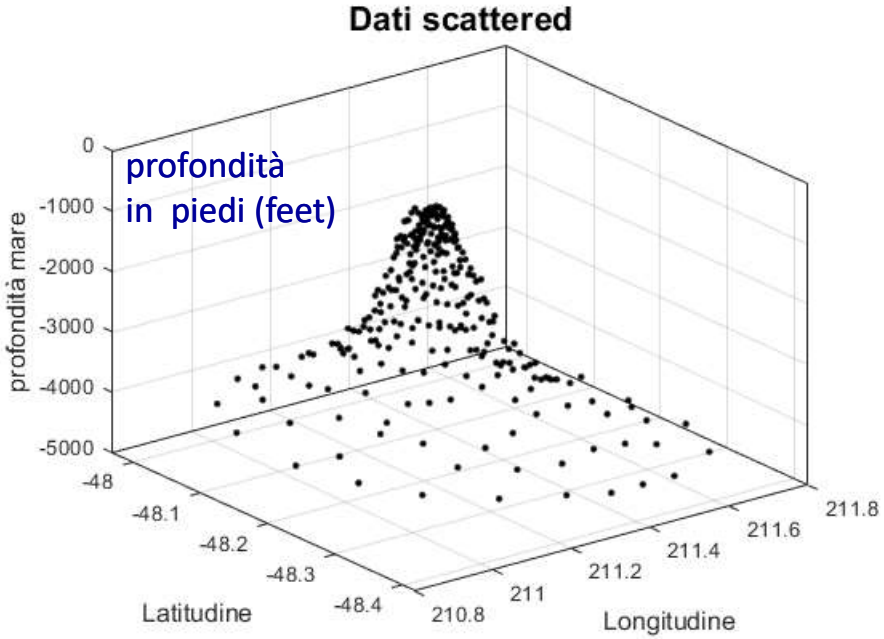
```
T =
     1     2     6
     4     5     6
     2     4     6
     2     3     4
     4     3     5
```

`trep.Points` `trep.ConnectivityList`

centro e raggio dei cerchi circoscritti ai triangoli



Triangolazione di Delaunay in MATLAB

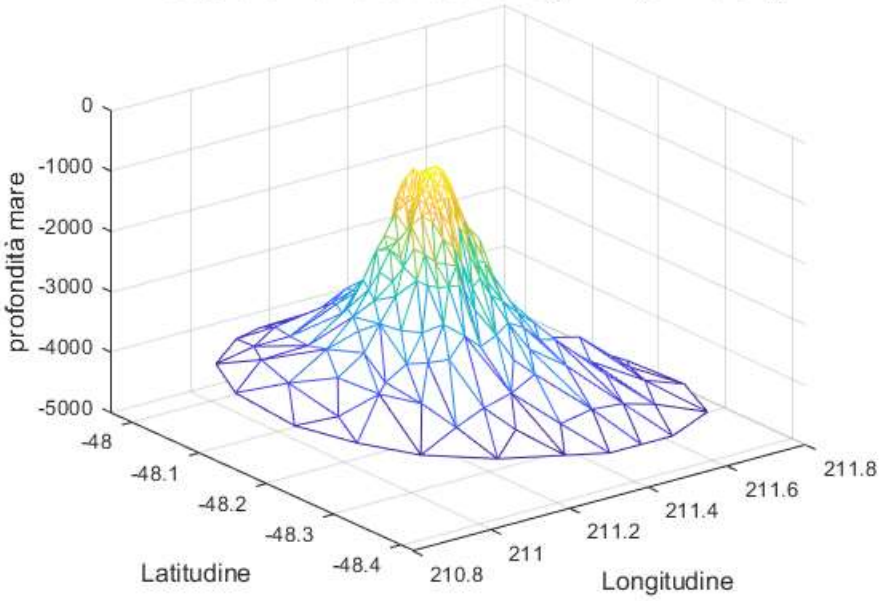


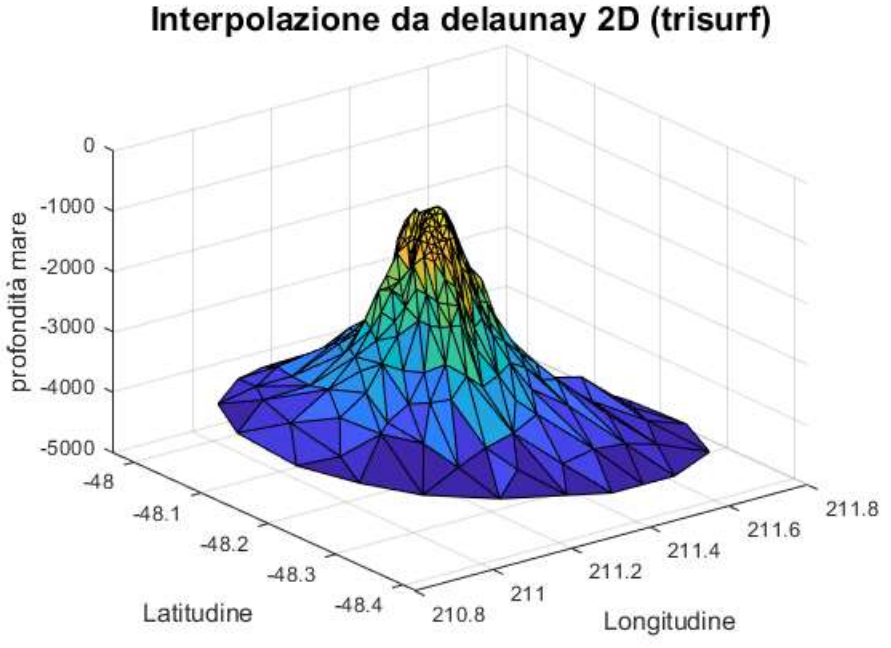
per convertire da piedi (feet) a metri:

- moltiplicare per 0.3048
- usare `convlength()` in Aerospace Toolbox

Delaunay 2D con dati 3D

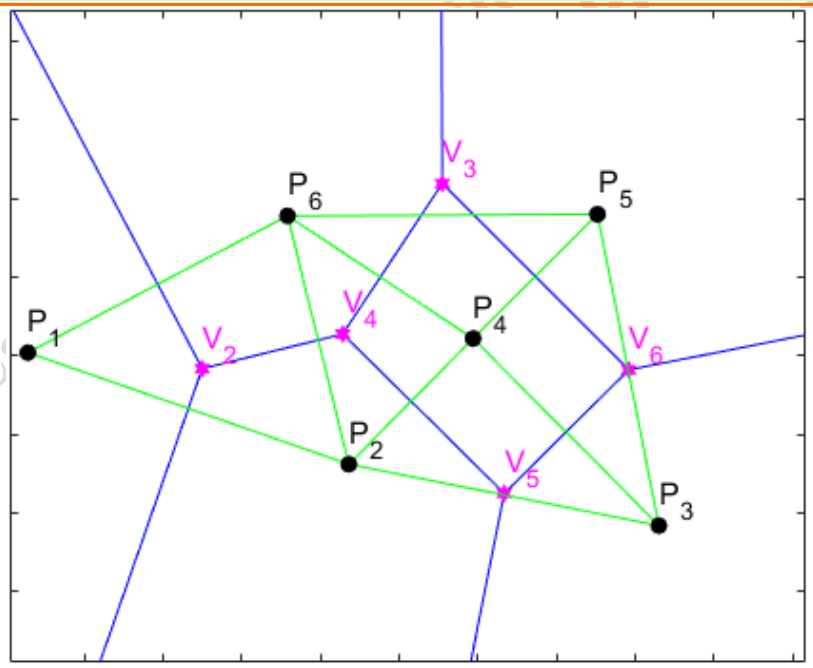
```
load seamount
T=delaunay(x,y);
figure; trimesh(T,x,y,z)
figure; trisurf(T,x,y,z)
```

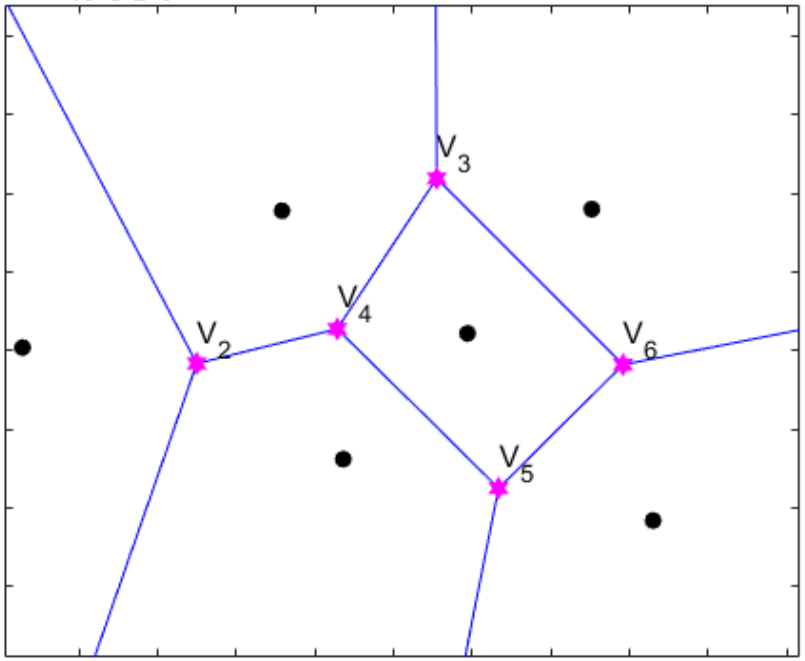




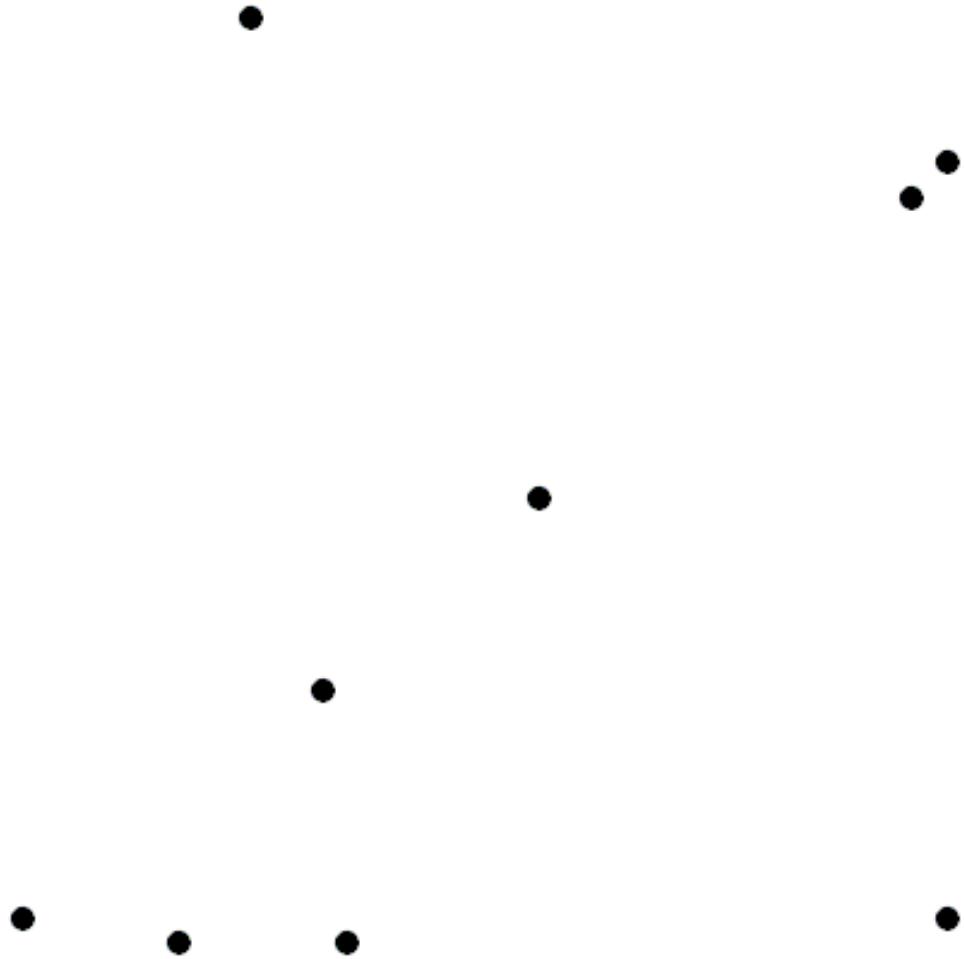
Tassellazione di Voronoi in MATLAB

```
P=[14, 202; 218, 131; 415, 92; 297, 211; 376, 290; 179, 289];
Np=size(P,1); X=P(:,1); Y=P(:,2);
figure; axis equal; hold on
DT=delaunayTriangulation(P); % crea un oggetto triangolazione di Delaunay
[vx,vy]=voronoi(DT); plot(vx,vy,'b') % disegna la tassellazione di Voronoi
[V,r]=voronoiDiagram(DT); % per i vertici V della tassellazione di Voronoi
Nv=size(V,1); plot(V(:,1),V(:,2),'hm','MarkerFaceColor','m','MarkerSize',8)
text(V(:,1),V(:,2), arrayfun(@(n) {sprintf('V_%d', n)}, (1:Nv)'), ...
                    'VerticalAlignment','bottom','FontSize',12,'Color','k')
triplot(DT,'-g'); % aggiunge la triangolarizzazione di Dalaunay
plot(X,Y,'ok','MarkerFaceColor','k')
```



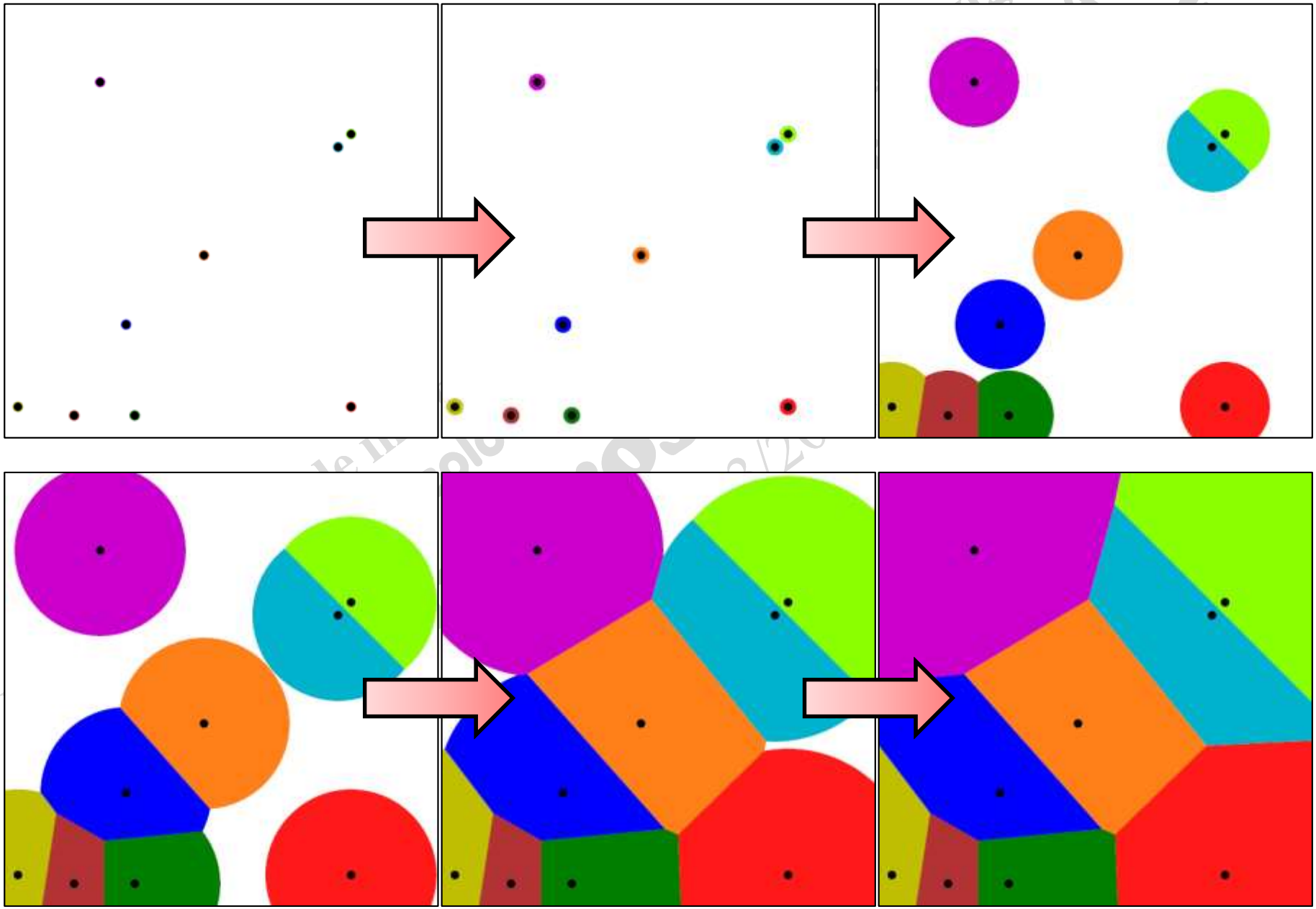


Tassellazione (o tassellatura) di Voronoi



Ogni regione attorno ad un centroide contiene i punti che sono più vicini a quel centroide

Tassellazione (o tassellatura) di Voronoi



Tassellazione di Voronoi in MATLAB (Thiessen polygons)

topoieti ← ArcGIS Pro

La Tassellazione di Voronoi è utilizzata in meteorologia e idrologia per trovare i pesi dei dati di precipitazione delle stazioni su un'area (bacino idrografico). I punti che generano i poligoni sono le varie stazioni che registrano i dati di precipitazione.

Il metodo (grafico) dei poligoni di Thiessen (o dei topoieti) consiste nel dividere l'area di interesse (il bacino) in regioni di influenza, una per ogni stazione. La regione di influenza S_i individua la zona dei punti che sono più vicini alla stazione i-sima di qualsiasi altra stazione.

Le regioni di influenza possono essere individuate disegnando gli assi dei segmenti di giunzione di stazioni adiacenti. In tal modo si formano dei poligoni attorno alle stazioni, detti aree di influenza della stazione.

La precipitazione media areale è calcolata con la formula

$$P_{media} = \sum_{i=1,..,n} \lambda_i \, p_i$$

dove p_i è il dato di precipitazioni e λ_i il peso della stazione i-sima, che viene calcolato come rapporto tra l'area della zona d'influenza e l'area totale del bacino.

L'individuazione dei poligoni col metodo grafico non è univoca, mentre con Delaunay e Voronoi lo è.

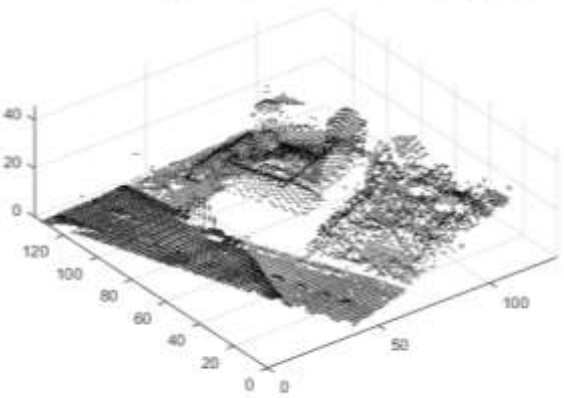
Esempio di applicazione: TIN

Le strutture dati digitali *Triangulated Irregular Network* (TIN) sono utilizzate in molte applicazioni, come, ad esempio, i sistemi di informazione geografica (GIS), la progettazione assistita da computer (CAD) per la rappresentazione visiva di una superficie topografica.

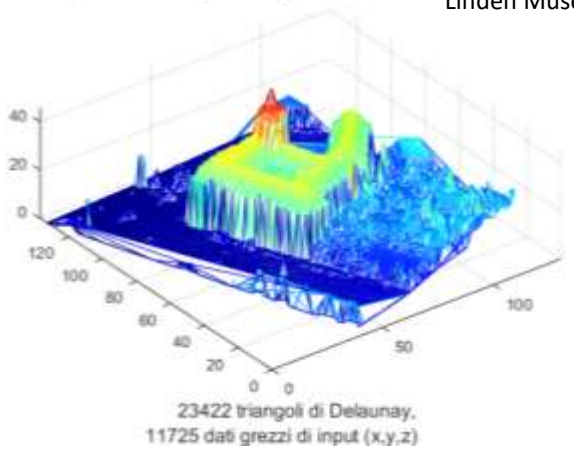
Un TIN è una rappresentazione vettoriale della superficie terrestre fisica o del fondale marino, costituita da nodi e linee distribuiti irregolarmente, con coordinate tridimensionali (x,y,z), e disposti in una rete di triangoli non sovrapposti.

Un TIN utilizzato per rappresentare il terreno è spesso chiamato modello digitale di elevazione (DEM), che può essere ulteriormente utilizzato per produrre modelli digitali di superficie (DSM) o modelli digitali del terreno (DTM).

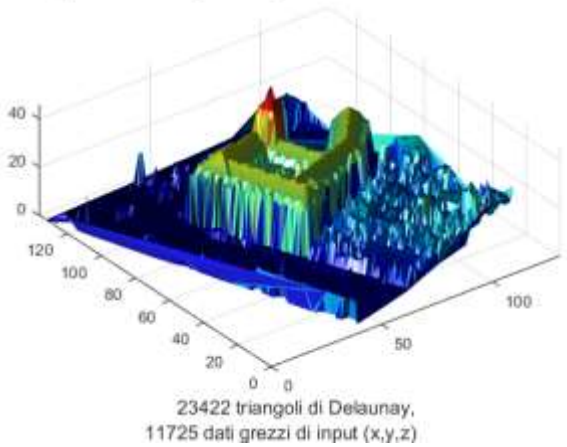
11725 dati grezzi ["raw data"] di input (x,y,z)



trimesh() di Delaunay Triangulation su raw data: Stoccarda Linden Museum



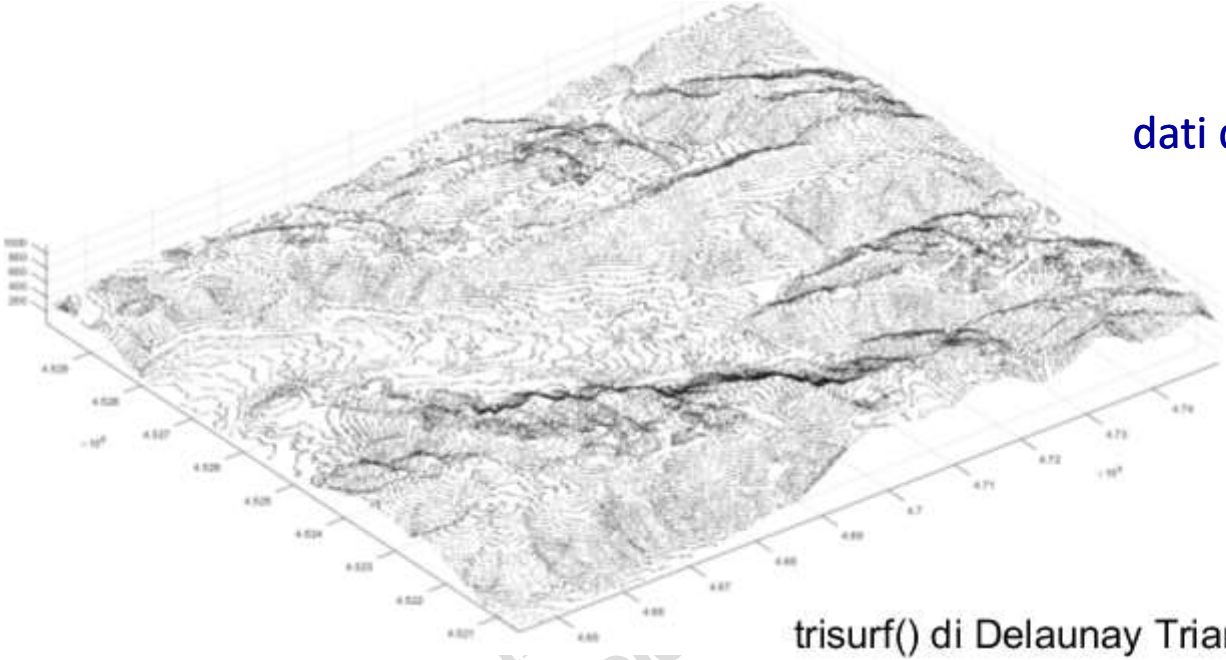
trisurf() di Delaunay Triangulation su raw data: Stoccarda



dati da “airborne laser scanning”

DTM da Delaunay Triangulation

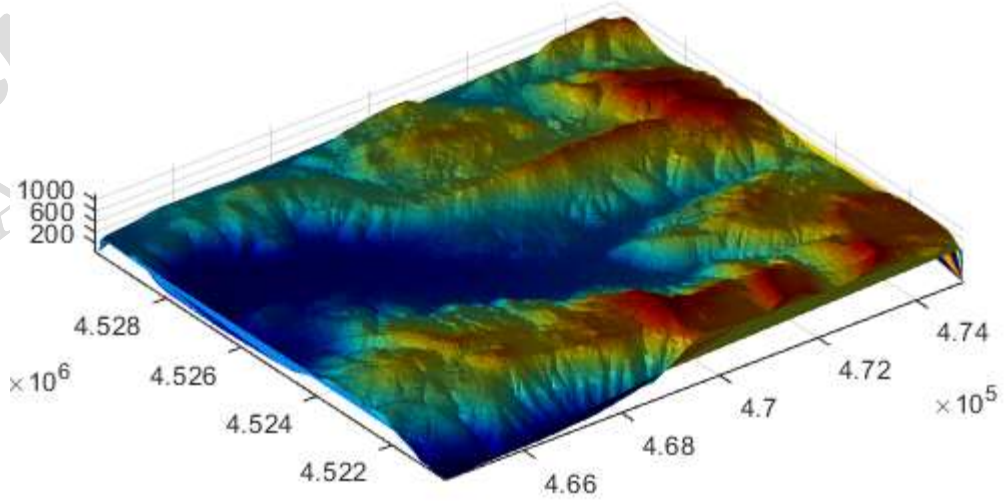
Esempio di applicazione: TIN



dati da “airborne laser scanning”

45985 raw data

trisurf() di Delaunay Triangulation su raw data: foceSarno



DTM da Delaunay Triangulation

91952 triangoli di Delaunay,
45985 dati grezzi di input (x,y,z)