Interpolating Scattered Data

Scattered data consists of a set of points X and corresponding values V, where the points have no structure or order between their relative locations. There are various approaches to interpolating scattered data. One widely used approach uses a Delaunay triangulation of the points.

Create a scattered data set on the surface of a paraboloid.

X = [-1.5 3.2; 1.8 3.3; -3.7 1.5; -1.5 1.3; ...

0.8 1.2; 3.3 1.5; -4.0 -1.0; -2.3 -0.7;

0 -0.5; 2.0 -1.5; 3.7 -0.8; -3.5 -2.9; ...

-0.9 -3.9; 2.0 -3.5; 3.5 -2.25];

V = X(:,1).^2 + X(:,2).^2;

hold on

plot3(X(:,1),X(:,2),zeros(15,1), '\*r')

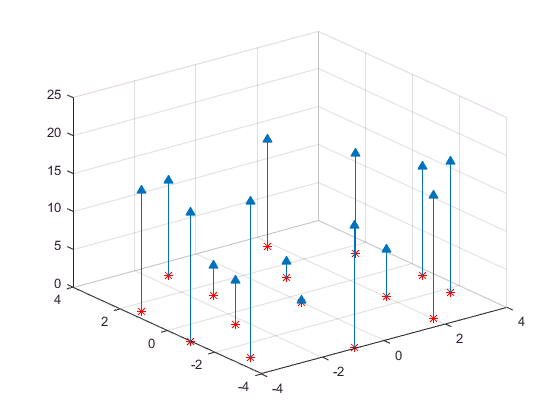
axis([-4, 4, -4, 4, 0, 25]);

grid

stem3(X(:,1),X(:,2),V,'^','fill')

hold off

view(322.5, 30);



Create a Delaunay triangulation, lift the vertices, and evaluate the interpolant at the query point Xq.

figure('Color', 'white')

t = delaunay(X(:,1),X(:,2));

hold on

trimesh(t,X(:,1),X(:,2), zeros(15,1), 'EdgeColor','r', 'FaceColor','none')

defaultFaceColor = [0.6875 0.8750 0.8984];

trisurf(t,X(:,1),X(:,2), V, 'FaceColor', defaultFaceColor, 'FaceAlpha',0.9);

plot3(X(:,1),X(:,2),zeros(15,1), '\*r')

axis([-4, 4, -4, 4, 0, 25]);

grid

plot3(-2.6,-2.6,0,'\*b','LineWidth', 1.6)

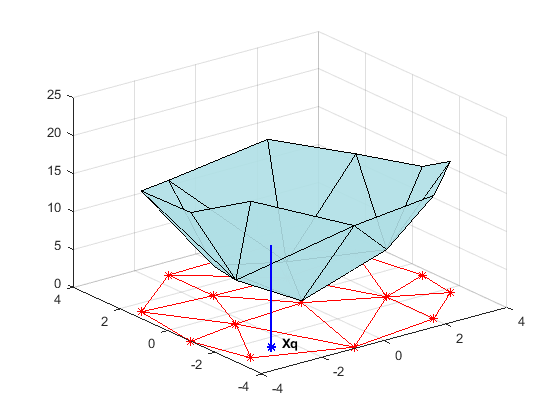
plot3([-2.6 -2.6]',[-2.6 -2.6]',[0 13.52]','-b','LineWidth',1.6)

hold off

view(322.5, 30);

text(-2.0, -2.6, 'Xq', 'FontWeight', 'bold', ...

'HorizontalAlignment','center', 'BackgroundColor', 'none');



**Interpolating Scattered Data Using griddata and griddatan**

The griddata and griddatan functions take a set of sample points, X, corresponding values, V, and query points, Xq, and return the interpolated values, Vq. The calling syntax is similar for each function; the primary distinction is the 2-D / 3–D griddata function lets you define the points in terms of X, Y / X, Y, Z coordinates. These two functions interpolate scattered data at predefined grid-point locations; the intent is to produce gridded data, hence the name. Interpolation is more general in practice. You might want to query at arbitrary locations within the convex hull of the points.

**Example**

Plot the seamount data set (a seamount is an underwater mountain). The data set consists of a set of longitude (x) and latitude (y) locations, and corresponding seamount elevations (z) measured at those coordinates.

load seamount

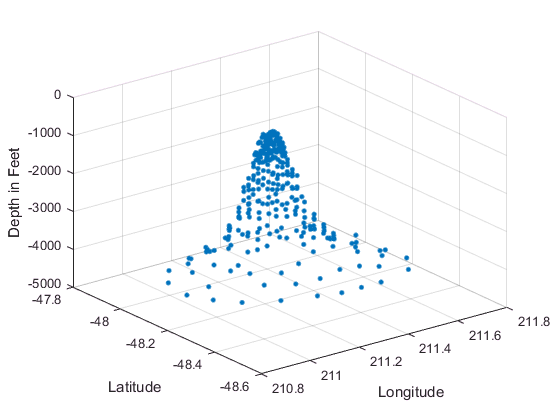
plot3(x,y,z,'.','markersize',12)

xlabel('Longitude')

ylabel('Latitude')

zlabel('Depth in Feet')

grid on



Use meshgrid to create a set of 2-D grid points in the longitude-latitude plane and then use griddata to interpolate the corresponding depth at those points.

figure

[xi,yi] = meshgrid(210.8:0.01:211.8, -48.5:0.01:-47.9);

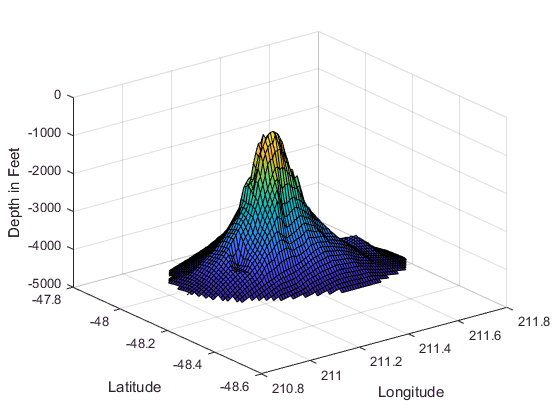
zi = griddata(x,y,z,xi,yi);

surf(xi,yi,zi);

xlabel('Longitude')

ylabel('Latitude')

zlabel('Depth in Feet')



Now that the data is in a gridded format, compute and plot the contours.

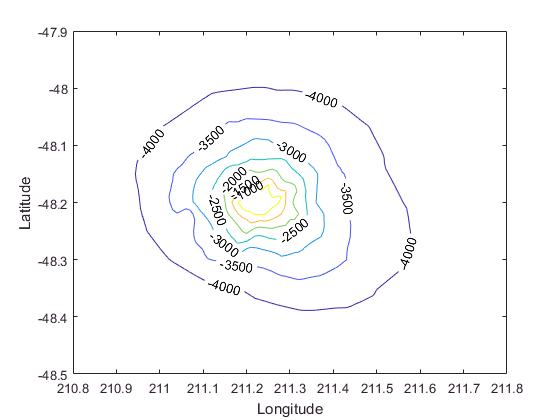
figure

[c,h] = contour(xi,yi,zi);

clabel(c,h);

xlabel('Longitude')

ylabel('Latitude')



You can also use griddata to interpolate at arbitrary locations within the convex hull of the dataset. For example, the depth at coordinates (211.3, -48.2) is given by:

zi = griddata(x,y,z, 211.3, -48.2);

**scatteredInterpolant Class**

The griddata function is useful when you need to interpolate to find the values at a set of predefined grid-point locations. In practice, interpolation problems are often more general, and the scatteredInterpolant class provides greater flexibility.

scatteredInterpolant provides the following interpolation methods:

* 'nearest' — Nearest-neighbor interpolation, where the interpolating surface is discontinuous.
* 'linear' — Linear interpolation (default), where the interpolating surface is C0 continuous.
* 'natural' — Natural-neighbor interpolation, where the interpolating surface is C1 conti

**Example**

**%Create the scattered data set.**

X = -3 + 6.\*gallery('uniformdata',[250 2],0);

V = peaks(X(:,1),X(:,2));

% **Create the interpolant.**

F = scatteredInterpolant(X,V);

%Evaluate the interpolant.

Vq = F([1.5 1.25])

Vq=1.3966

% You can evaluate at a vector of point locations:

Xq = [0.5 0.25; 0.75 0.35; 1.25 0.85];

Vq = F(Xq)

Vq =

1.0880

1.8127

2.3472

%You can evaluate F at grid point locations and plot the result.

[Xq,Yq] = meshgrid(-2.5:0.125:2.5);

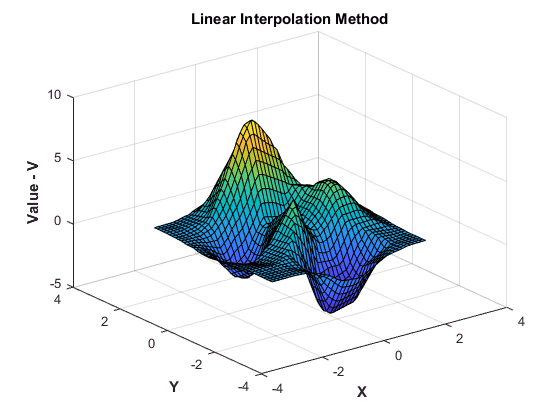
Vq = F(Xq,Yq);

surf(Xq,Yq,Vq);

xlabel('X','fontweight','b'), ylabel('Y','fontweight','b');

zlabel('Value - V','fontweight','b');

title('Linear Interpolation Method','fontweight','b');



% You can change the interpolation method on the fly. Set the method to 'nearest'.

F.Method = 'nearest';

Vq = F(Xq,Yq);

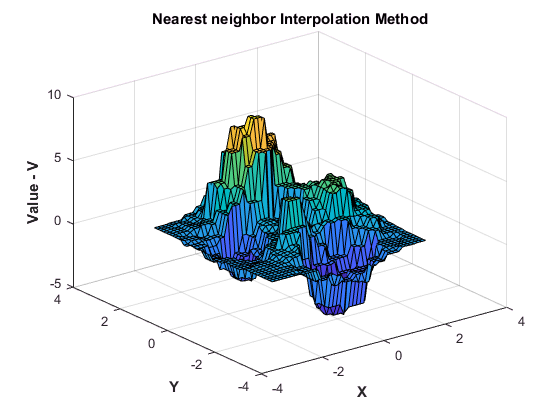
figure

surf(Xq,Yq,Vq);

xlabel('X','fontweight','b'),ylabel('Y','fontweight','b')

zlabel('Value - V','fontweight','b')

title('Nearest neighbor Interpolation Method','fontweight','b');



% Change the interpolation method to natural neighbor, reevaluate, and plot the results.

F.Method = 'natural';

Vq = F(Xq,Yq);

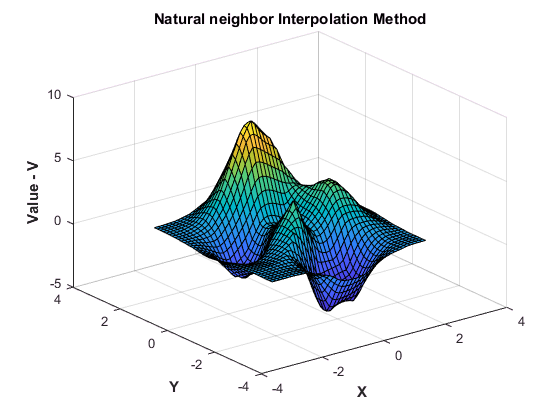
figure

surf(Xq,Yq,Vq);

xlabel('X','fontweight','b'),ylabel('Y','fontweight','b')

zlabel('Value - V','fontweight','b')

title('Natural neighbor Interpolation Method','fontweight','b');



**Addressing Problems in Scattered Data Interpolation**

Many of the illustrative examples in the previous sections dealt with the interpolation of point sets that were sampled on smooth surfaces. In addition, the points were relatively uniformly spaced. For example, clusters of points were not separated by relatively large distances. In addition, the interpolant was evaluated well within the convex hull of the point locations.

When dealing with real-world interpolation problems the data may be more challenging. It may come from measuring equipment that is likely to produce inaccurate readings or outliers. The underlying data may not vary smoothly, the values may jump abruptly from point to point. This section provides you with some guidelines to identify and address problems with scattered data interpolation.

**Input Data Containing NaNs**

You should preprocess sample data that contains NaN values to remove the NaN values as this data cannot contribute to the interpolation. If a NaN is removed, the corresponding data values/coordinates should also be removed to ensure consistency. If NaN values are present in the sample data, the constructor will error when called.

The following example illustrates how to remove NaNs.

Create some data and replace some entries with NaN:

x = rand(25,1)\*4-2;

y = rand(25,1)\*4-2;

V = x.^2 + y.^2;

x(5) = NaN; x(10) = NaN; y(12) = NaN; V(14) = NaN;

This code errors:

F = scatteredInterpolant(x,y,V);

Instead, find the sample point indices of the NaNs and then construct the interpolant:

nan\_flags = isnan(x) | isnan(y) | isnan(V);

x(nan\_flags) = [];

y(nan\_flags) = [];

V(nan\_flags) = [];

F = scatteredInterpolant(x,y,V);