MATLAB Graphics - 3D Plotting

Leigh J. Little SUNY Brockport September 5, 2001

Contents

1 Introduction

This handout illustrates the basic commands for creating 3D figures in MATLAB. There are 2 approaches to conceptually viewing 3D data. The simplest is a set of points in 3D space. This is called a line graph. The more common approach is a generalization of a line graph called a surface. You can view a surface as the graphical representation of some 2D function $z = f(x, y)$ over some region of the x-y plane.

1.1 Relation to 2D Plots

Most of the 2D plotting commands also apply to 3D figures. This includes the `title`, `xlabel`, `ylabel` (and `zlabel`), `axis`, `subplot` and `legend` commands. Some of these (*e.g.*, the `axis` command) require additional arguments to account for the added z direction.

2 Line Graphs

As an example of a line graph, enter the following commands at the MATLAB prompt:

```
>>  t = (0:8*pi/200:8*pi);
>>  x = cos(t); y = sin(t); z = 4*t;
>> plot3(x,y,z)
```

This graph is called a circular helix and is shown in Figure 1. Line graphs can have the same line and marker types as in 2D plots.

2.1 Setting the Viewpoint

A useful feature 3D figures is the ability to rotate your viewpoint. To do this, click on the `rotate` button on the toolbar (the one that looks like a circular arrow). When you click on the graph, you see the outline of the graph limits as a large rectangular box. By moving the mouse you can alter the viewpoint.

If you want to set the viewpoint more precisely, you can use the `view` command. To understand the view command, you need to recall the use of `spherical coordinates`.

Consider point A in Figure 2. This point is located at (x, y, z) in Cartesian coordinates. For some applications, such as orbit mechanics or gyroscopic motion, Cartesian coordinates are cumbersome to use.

These can be transformed to the more convenient spherical coordinates (ρ, θ, ϕ) using the formulas

$$\rho = \sqrt{x^2 + y^2 + z^2}, \quad \theta = \tan^{-1}\left(\frac{y}{x}\right), \quad \phi = \cos^{-1}\left(\frac{z}{\rho}\right).$$

The spherical coordinates of A are also indicated in Figure 2. The `view` command uses the angles θ and ϕ to set the viewpoint. Some examples are:

`>> view(30,60)` - sets viewpoint to $\theta = 30$, $\phi = 60$.

`>> view(150,-40)` - sets viewpoint to $\theta = 150$, $\phi = -40$. Here, you are looking at the graph from underneath.

`>> view(2)` - sets the viewpoint to a 2D flat view, down the z axis. This also called the `plan view`.

`>> view(3)` - sets the viewpoint to the default values of $\theta = -37.5$, $\phi = 30$.

The angles for the viewpoint should be input in degrees.

3 Surface Plots

Surface plots are the most common way to view 3D data. To see a quick example of a surface plot, enter:

```
>> peaks
```

This graph is shown in Figure 3. As can be seen, the graph consists wavy rectangular grid with large peaks in the positive and negative z directions.

Notice that the color of the mesh depends on the height of the surface above (or below) the x-y plane. This coloring is called a *colormap* and can be changed using the `colormap` command:

```
>> colormap('hsv')
>> colormap('summer')
>> colormap('bone')
```

You can get a list of all available colormaps by entering `help graph3d`.

You can also add a *colorbar* to the figure. A colorbar is an index that show the correspondence between color and function value. To place a colorbar to the right of the figure, enter:

```
>> colorbar
```

To add a colorbar at the bottom of the figure, enter:

```
>> colorbar('vertical')
```

Figure 4 shows the peaks function with the colorbar added.

3.1 Uniform Rectangular Grids

Surface plots are appropriate whenever you have a 2D region of discrete points and a corresponding function value. You can think of a surface plot as a sequence of intersecting 3D line graphs. These lines form a *mesh*. To get a better idea of what the mesh is, consider a specific example. Let x and y be the 2 equally spaced vectors (though they do not have to be equally spaced in general) on the interval $[-2, 2]$, each having 21 points.

```
>> close
>> x = (-2:4/20:2); y = x;
```

We would like to plot the function $z = e^{-(x^2+y^2)}$ over the rectangular region $x \in [-2, 2]$, $y \in [-2, 2]$. The function z needs to be evaluated at the discrete points given in the vectors x and y. To do this, an x-y mesh needs to be created. This can be done by entering:

```
>> figure(1)
>>    [X,Y]=meshgrid(x,y);
>>    mesh(x,y,0*X); view(2)
>>    xlabel('x')
>>    ylabel('y')
>>    title('Uniform Mesh')
```

This creates the two mesh matrices X and Y and plots the mesh in a 2D view (see Figure 5). This is an example of a *uniform* mesh, which is characterized by equal spacing of the grid points in each coordinate direction.

Now that the mesh has been created, z can now be evaluated on the discrete mesh:

```
>> z = exp(-(X.^2 + Y.^2));
```

Note that you *must* use the vectorized version of the computation. The surface for z can now be plotted:

```
>> figure(2)
>>   surf(x,y,z)
>>   xlabel('x')
>>   ylabel('y')
>>   zlabel('z')
>>   title('Simple surface plot')
```

This is shown in Figure 6. You should see a bell shaped surface with its peak at the origin. You can also see the original mesh has been deformed so that it conforms to the function surface.

The spacing between grid points in the x and y directions do not have to be the same in order for the mesh to be uniform. All that is required is for the spacing in each direction to be constant. For example, the mesh generated by the commands below is also a uniform mesh:

```
>> close all
>> x = (-2:4/20:2); y = (-2:4/40:2);
>> [X,Y]=meshgrid(x,y);
>> figure(1)
>>   mesh(x,y,0*X); view(2)
>>   xlabel('x')
>>   ylabel('y')
>>   title('Uniform Computational Mesh')
>>
>> z = exp(-(X.^2 + Y.^2));
>> figure(2)
>>   surf(x,y,z)
>>   xlabel('x')
>>   ylabel('y')
>>   zlabel('z')
>>   title('Simple surface plot, uniform mesh')
```

These plots are shown in Figures 7 and 8 respectively.

3.2 Non-Uniform Rectangular Grids

Now consider the same function as in the previous example. From the figure, it seems like most of the interesting behavior of the graph is near the center of the graph. In order to capture these features more accurately, a grid that puts more discrete points in the center portion of the region can be created:

```
>> close all
>> x = [(-2:.2:-1) (-0.9:.1:1) (1.2:.2:2)];
>> y = x;
>> [X,Y]=meshgrid(x,y);
>> figure(1)
>>   mesh(x,y,0*X); view(2)
```

```
>>   xlabel('x')
>>   ylabel('y')
>>   title('Non-uniform Computational Mesh')
>>
>> z = exp(-(X.^2 + Y.^2));
>> surf(x,y,z)
>> xlabel('x')
>> ylabel('y')
>> zlabel('z')
>> title('Simple surface plot, non-uniform')
```

These plots are shown in Figures 9 and 10 respectively.

3.3 Arbitrary data

The `surf` command is useful for creating 3D plots, but it has big disadvantage. The data must lie on a rectangular grid in order to view the the graph. Sometimes the data is not always available in this form and you need to create a surface plot of an arbitrary set of 3D coordinates (similar to the data for a line graph). You can view this data as a surface by using the `griddata` command. As an example, generate 3 vectors of sorted random numbers to use as the coordinate points.

```
>> close all
>> x = sort(rand(20));
>> y = sort(rand(20));
>> z = sort(rand(20));
>>
>> xmin = min(min(x)); xmax = max(max(x));
>> xnew = (xmin:(xmax-xmin)/10:xmax);
>> ymin = min(min(y)); ymax = max(max(y));
>> ynew = (ymin:(ymax-ymin)/20:ymax);
>> X = xnew; Y = ynew;
>> [X,Y,znew]=griddata(x,y,z,X,Y');
>>
>> figure(1)
>>   mesh(X,Y,0*X)
>>   title('Original points and new uniform grid')
>>   xlabel('x')
>>   ylabel('y')
>>   axis equal
>>
>> figure(2)
>>   surf(X,Y,znew);
>>   xlabel('x')
>>   ylabel('y')
>>   zlabel('z')
>>   title('Interpolated function on the uniform grid')
```

These plots are shown in Figures 11 and 12 respectively. Note that the function is oddly shaped because the data is random. The `griddata` function takes five input values. The first three are the discrete coordinate vectors x, y and z. You also must provide 2 vectors that will be used to create the uniform grid (the vectors `xnew` and `ynew` in the example above). The `griddata` function then interpolates the function z on the scattered data to the function $znew$ on the uniform grid.

3.4 Mesh Plots

Mesh plots are similar to the surface plots created with `surfl` except the squares are not colored. All that is seen is the mesh distortion. Enter the following commands:

```
>> close all
```

```
>> x = (-2:4/20:2); y = x;
>> [X,Y] = meshgrid(x,y);
>> z = exp(-(X.^2+Y.^2));
>> mesh(x,y,z);
>> xlabel('x')
>> ylabel('y')
>> zlabel('z')
>> title('Simple mesh plot')
```

This graph is shown in Figure 13. Just like surface plots, you can add colorbars and change the colormap for the figure.

3.5 Contour Plots

Contour plots are surface plots that are viewed in the plan view (`view(2)`). Height on the surface is represented by *level curves*. Each point on a level curve is at the same elevation. Contour plots are frequently encountered in geography and cartography. To create a contour plot, enter the commands:

```
>> close all
>> x = (-2:4/20:2); y = (-2:4/20:2);
>> [X,Y]=meshgrid(x,y);
>> z = exp(-(X.^2 + Y.^2));
>> figure(1)
>>   contour(x,y,z)
>>   axis square
>>   xlabel('x')
>>   ylabel('y')
>>   title('Simple contour plot')
```

The plot is shown in Figure 14. This is the same bell function from before, except it looks more like a target. This is because the bell is symmetric to the origin and you are looking straight down at the surface from above. This graph is not too helpful because there is no legend. The individual level curves can be labeled by entering:

```
>> close all
>> Z=contour(x,y,z);
>> clabel(Z);
>> xlabel('x')
>> ylabel('y')
>> title('Labeling the contours')
```

The resulting plot is shown in Figure 15. Now each of the level curves has the height value attached to it. By default, MATLAB will draw 9 level curves at equally spaced values between the minimum and maximum function values. You can control the number of contours by including an additional parameter:

```
>> close all
>> Z=contour(x,y,z,20);
>> clabel(Z);
>> xlabel('x')
>> ylabel('y')
>> title('Labeling the contours, 20 levels')
```

The resulting plot is shown in Figure 16. Here, the `contour` function has been instructed to draw 20 contours.

You can also change the individual line types and specify exactly which level curves to draw. See `help contour` for more information.

Like the surface and mesh plots, you must have the data available on a rectangular grid to create a contour plot. If you do not have your data in this form, you can interpolate it to a regular grid as described above.

4 Lab for MATLAB Graphics - 3D Plotting

For each of the problem below, be sure to include appropriate annotation (labels, *etc.*) to your graphs.

1) This problem relates to the function
$$f(x,y) = e^{-(x^2+y^2)}.$$
   a) Create a surface plot of $f(x,y)$ on the region $x \in [-1,1]$, $y \in [-1,1]$.
   b) Create a mesh plot of $f(x,y)$ on the region $x \in [-1,1]$, $y \in [-1,1]$.
   c) Create contour plot of $f(x,y)$ on the region $x \in [-1,1]$, $y \in [-1,1]$. You should display the 10 contour levels $\{0.1, 0.3, 0.5, 0.6, 0.7, 0.80, 0.85, 0.9, 0.95, 1.0\}$.

2) Generate a matrix A of size 20×20 with random entries using the `random` command.
   a) Create a surface plot of A. Note that your x and y axis limits are the indices of the matrix entries.
   b) Create a mesh plot of A.
   c) Create contour plot of A. You should display the 11 contour levels $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$.

3) Take the matrix in Problem 2) and issue the command `A=sort(A)`. Repeat Problem 2) for the sorted matrix. How is the sorted matrix different from the original?

4) Repeat Problem 3) under the assumption that the matrix is plotted over the region $x \in [-1,1]$, $y \in [-1,1]$. Note that this only requires the inclusion of discrete x and y vectors of appropriate length.

5) Create the vectors `x = (0:.2:1); y = x;` then issue the command `[X,Y] = meshgrid(x,y);` Examine the matrices `X` and `Y`. Based on your observation, what does the `meshgrid` command do to the vectors `x` and `y`?

5 Figures

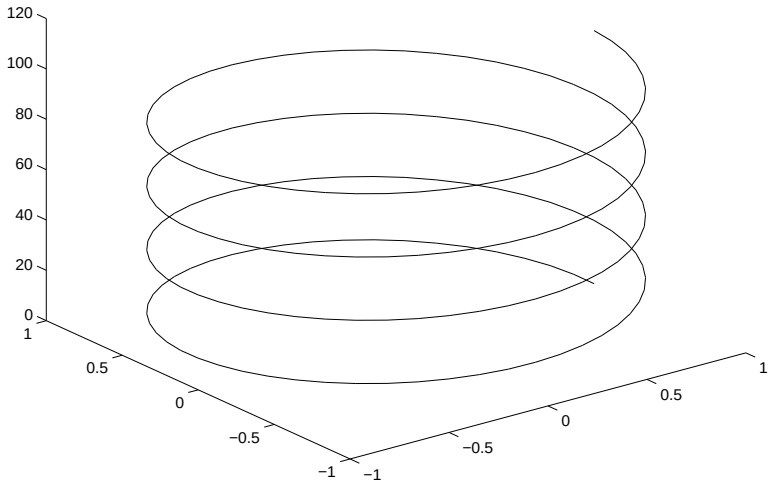

Figure 1: Simple line graph in 3D.

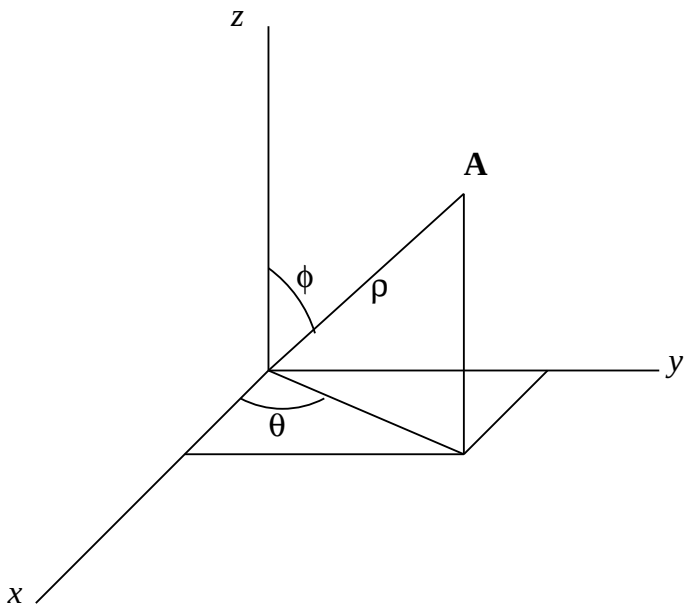

Figure 2: Cartesian and spherical coordinates.

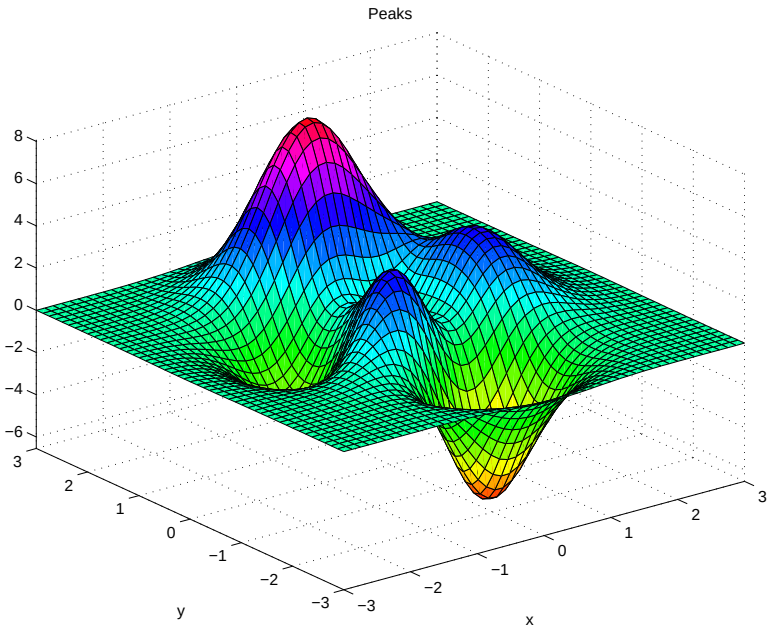

Figure 3: Simple surface plot.

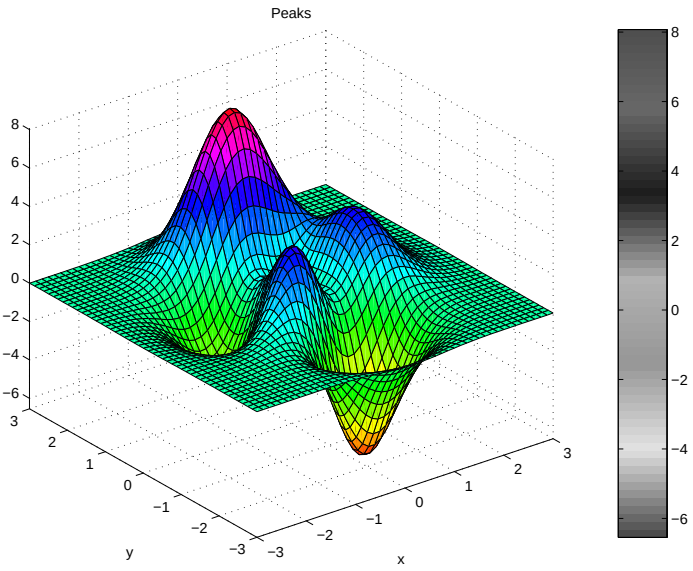

Figure 4: Simple surface plot with colorbar.

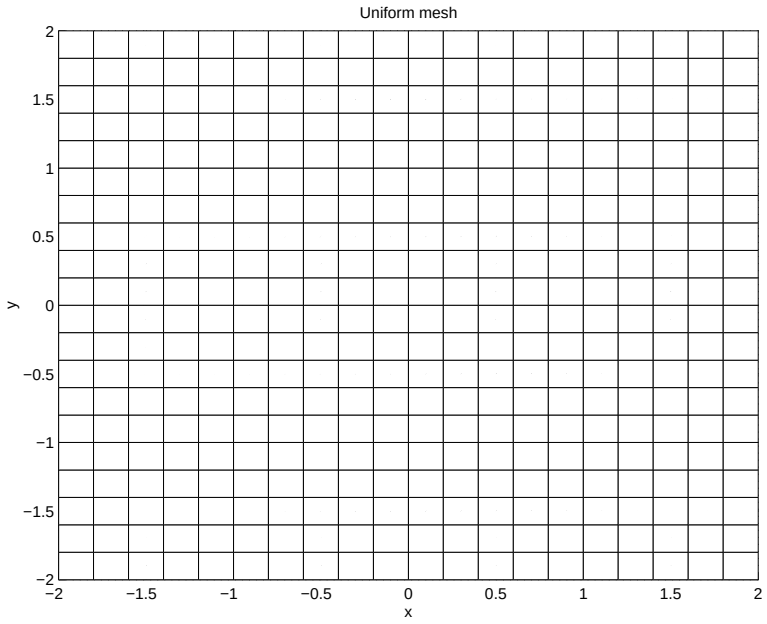

Figure 5: Uniform computational mesh.

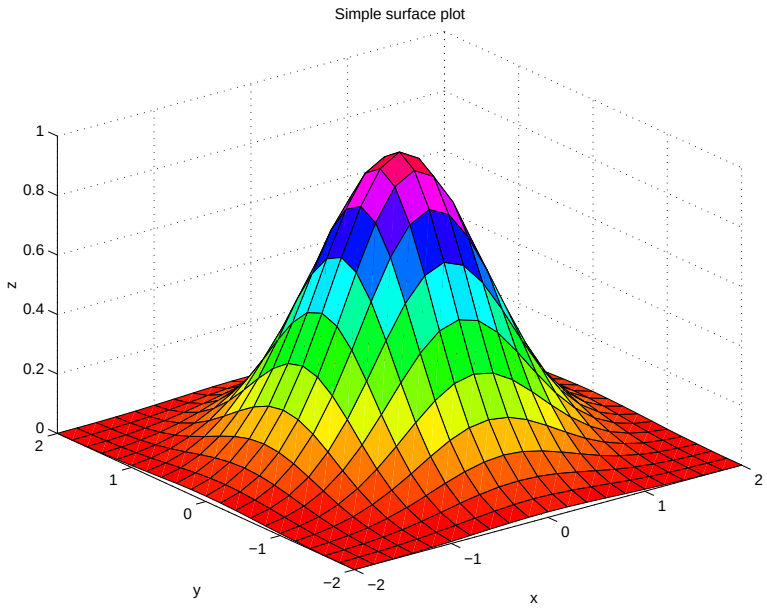

Figure 6: Simple surface plot, uniform computational mesh.

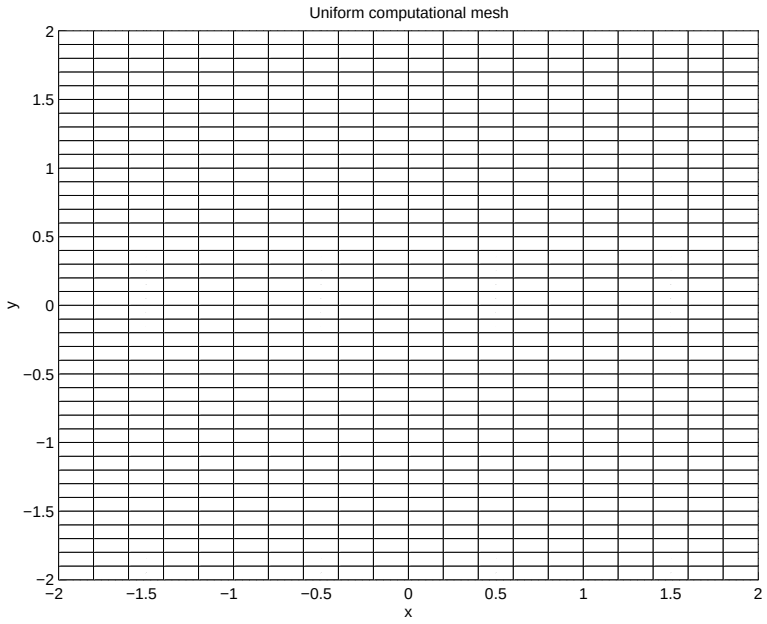

Figure 7: Another uniform computational mesh.

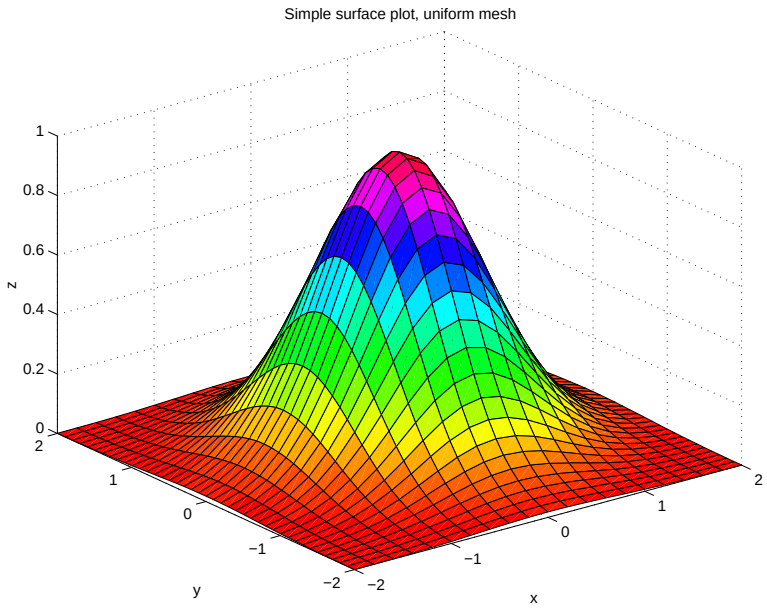

Figure 8: Surface plot on new uniform computational mesh.

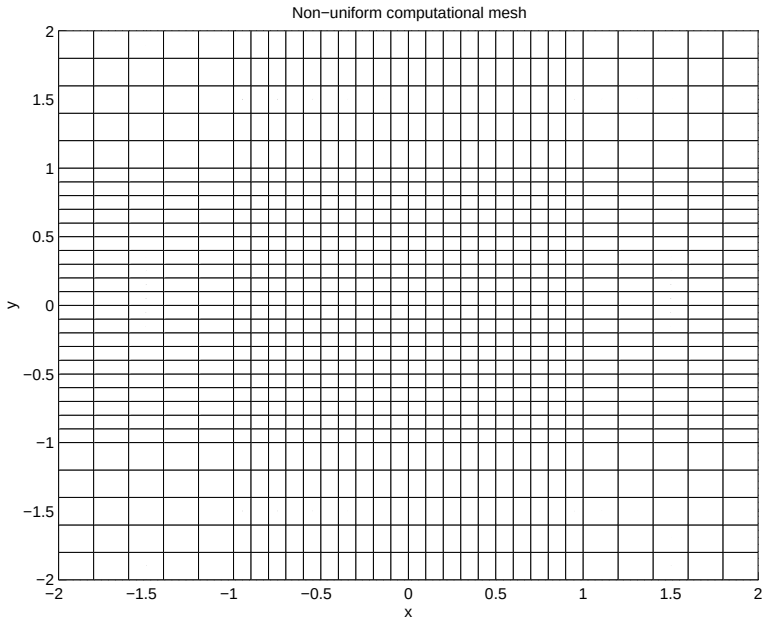

Figure 9: Non-uniform computational mesh.

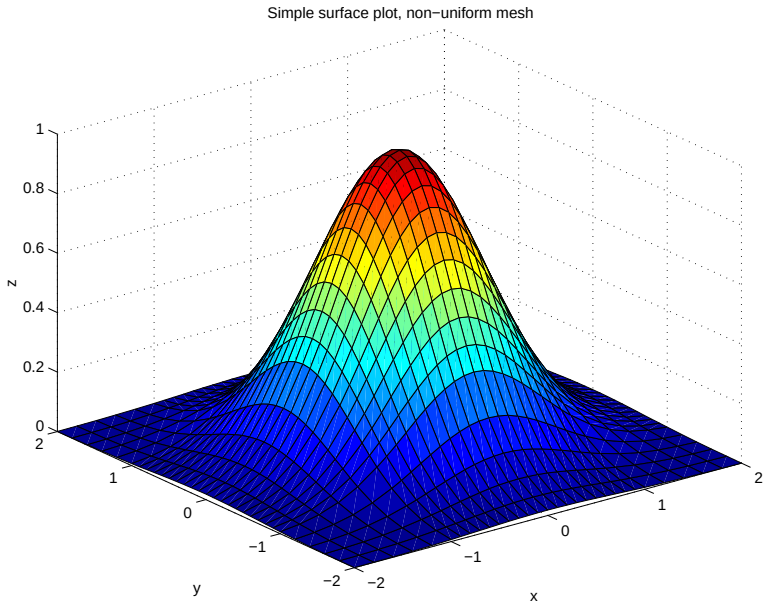

Figure 10: Surface plot on non-uniform computational mesh.

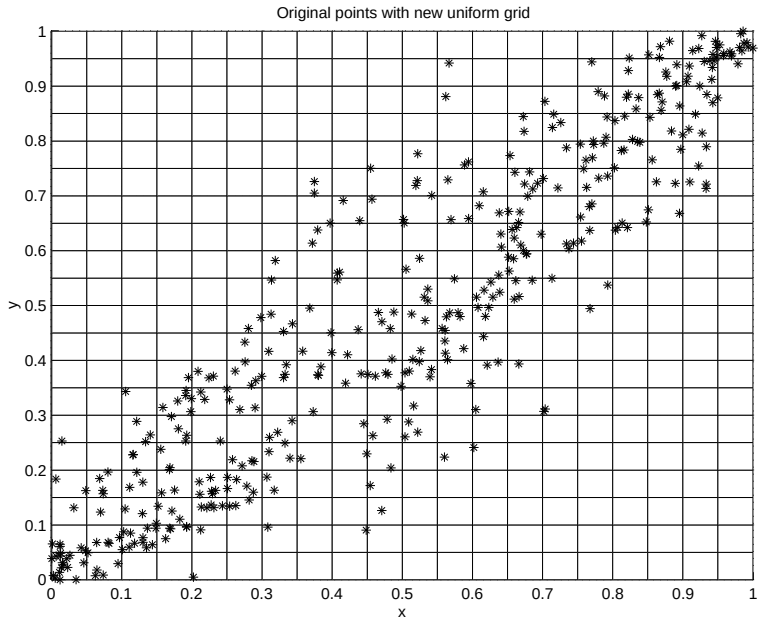

Figure 11: Irregular data and new uniform mesh.

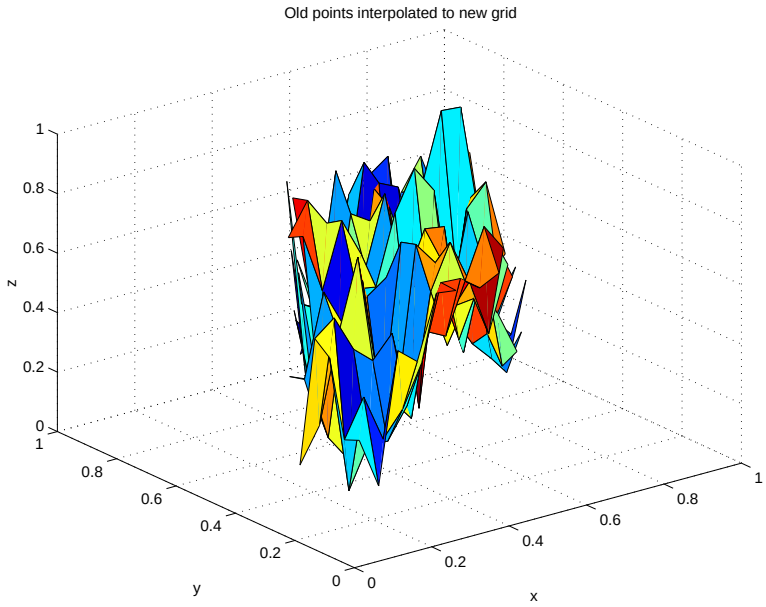

Figure 12: Interpolated function on the new uniform mesh.

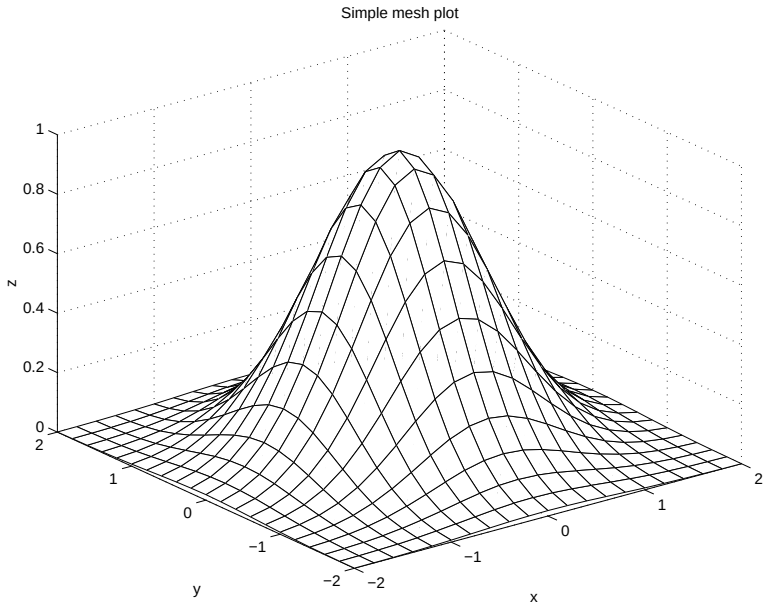

Figure 13: Simple mesh plot.

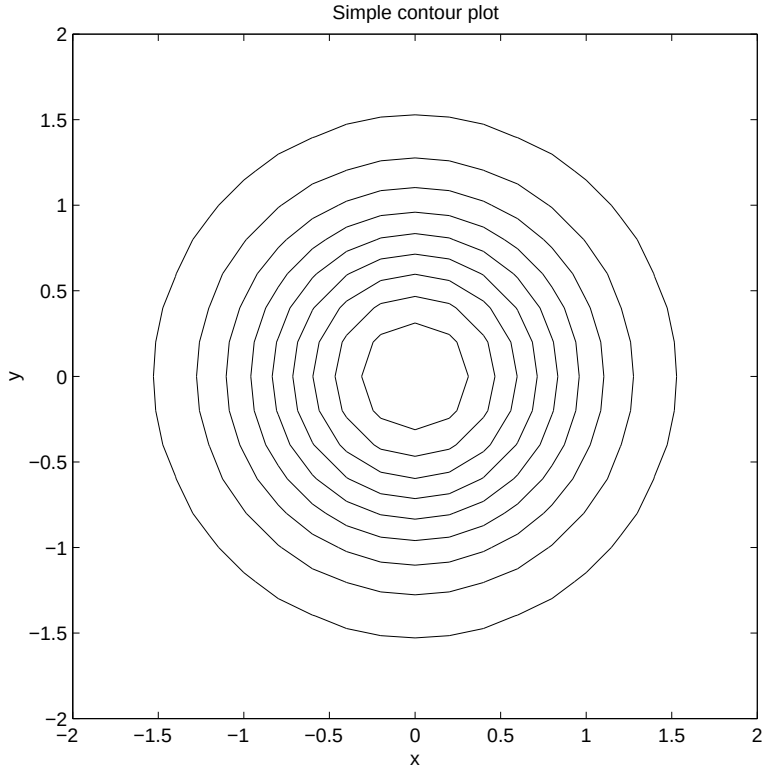

Figure 14: Simple contour plot.

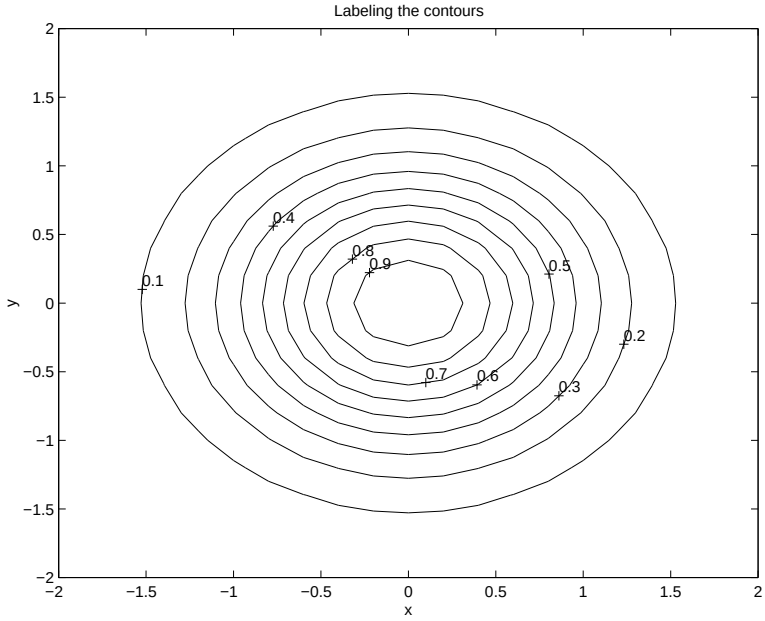

Figure 15: Simple contour plot with contours labeled.

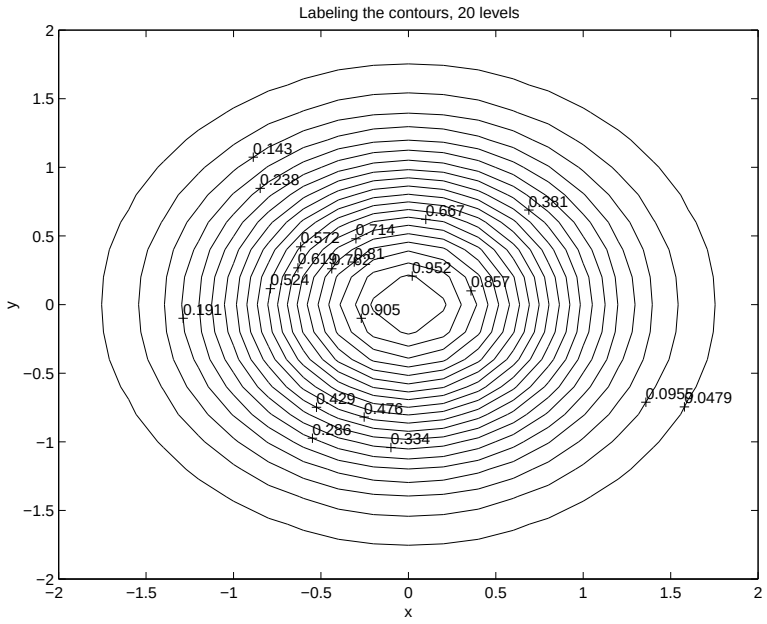

Figure 16: Simple contour plot with contours labeled, 20 levels.