MATLAB: 3-D Graphics

Last revised : March, 2003

Overview

To get an overview of available graphics functions in MATLAB, look at the help pages for the following topics:

`graph2d`	Two-dimensional graphs.
`graph3d`	Three-dimensional graphs.
`specgraph`	Specialized graphs.
`graphics`	Handle graphics.

This article illustrates the use of the *easy to use* graphics commands listed under the topic `specgraph`. At the end of the writeup two examples are given showing how one can set up the necessary data arrays to use the more primitive commands `contour`, `mesh`, and `surf`.

`ezplot`	Easy to use function plotter.
`ezplot3`	Easy to use 3-D parametric curve plotter.
`ezcontour`	Easy to use contour plotter.
`ezcontourf`	Easy to use filled contour plotter.
`ezmesh`	Easy to use 3-D mesh plotter.
`ezmeshc`	Easy to use combination mesh/contour plotter.
`ezsurf`	Easy to use 3-D colored surface plotter.
`ezsurfc`	Easy to use combination surf/contour plotter.

This brief document mainly consists of examples of MATLAB code. It is suggested that you have MATLAB running while reading so that you can verify the examples. Each of the examples includes an image of the expected result. Just click on the `View Result` tag to the right of the MATLAB code.

Plotting curves: `ezplot`, `ezplot3`

We begin with an example of 2-D graphics using `ezplot` to plot the graph $y = x + 2\sin(2x)$ on the interval $0 \leq x \leq 10$. For simple functions that can be easily written in one line of code we pass the MATLAB expression, enclosed in single quotation marks, as the first input argument to `ezplot`. The optional second argument specifies the plotting limits for the independent variable. These "easy-to-use" functions place the equation in the figure title and label the horizontal axes with the independent variables.

```
>> ezplot('x + 2*sin(2*x)', [0 10]);                (Ex 1) See Results
```

For more complicated functions, first define the function through an ***m-file***, then pass the name of the function to the plotting routine. Note that such a function must be written to operate on a vector of input values (in MATLAB lingo, the function must be ***vectorized***). For example, consider the following piecewise function defined in the file `func1.m`:

```
function y = func1(x,a)
for i = 1 : length(x)
  if ( x(i) < a )
    y(i) = a + (x(i) - a)^2;
  else
    y(i) = a - (x(i) - a)^2;
  end
end
```

Now plot $y = f(x)$ for several choices of the parameter a, overlaying the plots on a single axis.

```
>> ezplot('func1(x,0)', [-1 5]);
>> hold on
>> ezplot('func1(x,1)', [-1 5]);
>> ezplot('func1(x,2)', [-1 5]);
>> ezplot('x', [-1 5]);
>> axis([-1 5 -4 6]);                            (Ex 2) See Results
```

Plot the parametric curve, $x(t) = t + 2\sin(2t),\ y(t) = t + 2\cos(5t)$,

```
>> ezplot('t+2*sin(2*t)','t+2*cos(5*t)',[-10 10]);  (Ex 3) See Results
```

In xyz-space, use `ezplot3` to plot the parametric curve defined by, $\{x(t) = \cos t,\ y(t) = \sin t,\ z(t) = \sin(5t)\}$.

```
>> ezplot3('cos(t)','sin(t)','sin(5*t)',[-pi pi]);  (Ex 4) See Results
```

***Note*:** When using the `ez` functions, there does not appear to be any simple way to control plot attributes such as line color or style. The input arguments for `ezplot` do not include a color/style specification as permitted in the case of the `plot` command. More importantly, these `ez` functions do not return a graphics handle. There are methods for retrieving the handle and manipulating the attributes through the `set` function, but these techniques are too advanced for this introductory document. If editing is needed, the most direct solution is to turn on **Enable Plot Editing** from the menu bar along the top of the Figure window and use the mouse to select the line to be edited. Holding down the right mouse button over the object brings up a menu of attributes to be edited.

The exact sequence for editing graphics windows is not the same across all operating systems and versions of MATLAB*. Some of these editing features first appeared in version 5 but underwent significant changes in version 6. Moreover, the Linux version appears to be different than the Windows version.*

Contour plots, level curves: `ezcontour`, `ezplot`

The function `ezcontour` plots level curves for functions of two variables $z = f(x, y)$. The second and third arguments are the plotting limits for each of the independent variables x and y. One drawback to `ezcontour` is that it does identify the contour levels. Adding a colorbar to the graph will give you an approximate idea of the level associated with each curve. [The function `ezcontourf` produces *filled* contour plots.]

```
>> ezcontour('y^2 - x^2', [-3 3],[-3 3]);
>> colorbar                                      (Ex 5) See Results
```

To plot specific level curves, it is more effective to use `ezplot`. The command `ezplot('g(x,y)',[..],[..])` plots the level curve $g(x, y) = 0$. To plot the level curve $f(x, y) = k$, define $g(x, y) = f(x, y) - k$ in the previous command. In the following sequence, level curves for $z = 0, \pm 2, \pm 4$ are plotted on a single graph. The lines have been edited to make the negative levels ($z < 0$) dashed blue lines and the positive levels ($z > 0$) solid red lines. The labels are added by selecting **Insert** -> **Text**.

```
>> ezplot('y^2 - x^2 + 4', [-3 3],[-3 3]);
>> hold on
>> ezplot('y^2 - x^2 + 2', [-3 3],[-3 3]);
>> ezplot('y^2 - x^2',     [-3 3],[-3 3]);
>> ezplot('y^2 - x^2 - 2', [-3 3],[-3 3]);
>> ezplot('y^2 - x^2 - 4', [-3 3],[-3 3]);
>> grid on                                       (Ex 6) See Results
```

Surfaces as graphs: `ezmesh`, `ezsurf`

To introduce three-dimensional viewing of surfaces, we first consider examples of surfaces that can be plotted as graphs, $z = f(x, y)$.

The `mesh` functions represent the surface with a mesh where the four edges of each mesh element are colored according to the height of the function, but the faces of the mesh elements are left uncolored.

```
>> ezmesh('y^2 - x^2', [-3 3],[-3 3]);                    (Ex 7) See Results
```

The function `ezmeshc` also includes the level contours of the function $f(x, y)$ plotted on the "floor" of the 3-D plot. That is, if the plotting limits in the z-direction are $z_{\min} \le z \le z_{\max}$, then the level curves are plotted in the plane $z = z_{\min}$.

The `hidden` command controls whether or not to show hidden lines (this only applies to `mesh` plots). Use `hidden off` to make hidden lines visible.

The viewing angle for 3-D graphics can be changed by dragging the mouse inside the figure window if `rotate3d` has been turned on, either from the menu bar or by typing **`rotate3d on`** from the command line. From the command line, **`view(125,30)`** will set the viewing angle with horizontal rotation 125° and vertical elevation 30°. This is very similar to the viewing perspective typically used in Stewart's text.

```
>> ezmeshc('y^2 - x^2', [-3 3],[-3 3]);
>> hidden off
>> view(125, 30);                                         (Ex 8) See Results
```

The `surf` commands produce a colored surface with the faces of each mesh element colored according to the height of the surface $z = f(x, y)$.

```
>> ezsurf('-10*x*y*exp(-(x*x+y*y))', [-2.5 2.5], [-2.5 2.5]);
>> view(125, 30)                                          (Ex 9) See Results
```

The choice of colors for surface plots is determined by the current *colormap*. To see a list of available colormaps and a one-line description of each, run `helpwin graph3d`. The default colormap is *jet*.

The `shading` command controls the color shading of the mesh elements making up a surface. There are three choices: *flat* uses a single color within each mesh element; *faceted* uses *flat* shading with superimposed black mesh lines (this is the default); *interp* uses linear interpolation to make the shading change smoothly across the surface.

```
>> ezsurf('-10*x*y*exp(-(x^2 + y^2))', [-2.5 2.5], [-2.5 2.5]);
>> colormap(hsv)
>> shading('flat')
>> view(110,-20)                                         (Ex 10) See Results
```

The following example graphs the upper half of the sphere $x^2 + y^2 + z^2 = 9$ and overlays the tangent plane at $(x, y, z) = (0.5, 0.5, 2.915)$.

```
>> ezsurf('sqrt(9 - x^2 - y^2)', [-4 4], [-4 4]);
>> view(110,30);
>> hold on
>> ezmesh('2.915 - 0.172*(x - .5) - 0.172*(y - .5)', [-3 3], [-3 3]);
>> hidden off
>> hp = plot3(.5, .5, 2.915, 'r.');
>> set(hp, 'markersize', 20);
>> axis([-4 4 -4 4 0 4]);                                (Ex 11) See Results
```

Parametric surfaces: `ezmesh`, `ezsurf`

[*See Section 10.5 of Stewart's text for an introduction to parametric surfaces.*]

In the previous example for the upper half-sphere, note the ragged edge where the sphere meets the xy-plane. Here is an example of a surface that is best represented in parametric form,

$$x = f(u,v), \qquad y = g(u,v) \qquad z = h(u,v) \quad \text{for } \; a \le u \le b, \; c \le v \le d.$$

The functions `ezmesh` and `ezsurf` accept parametric representations; in this case the first three input arguments are the MATLAB expressions for $x(u,v)$, $y(u,v)$, and $z(u,v)$. In this first example, we graph the upper halves of two concentric spheres with a section removed from the outer sphere.

```
>> ezsurf('sin(u)*cos(v)', 'sin(u)*sin(v)',...
>>        'cos(u)', [0 pi/2 0 2*pi]);
>> hold on
>> ezsurf('2*sin(u)*cos(v)', '2*sin(u)*sin(v)',...
>>        '2*cos(u)', [0 pi/2 0 7*pi/4]);
>> axis([-2.1 2.1 -2.1 2.1 0 2.1]);
>> set(gca, 'DataAspectRatio', [1 1 1]);
>> view(50,25);                                      (Ex 12) See Results
```

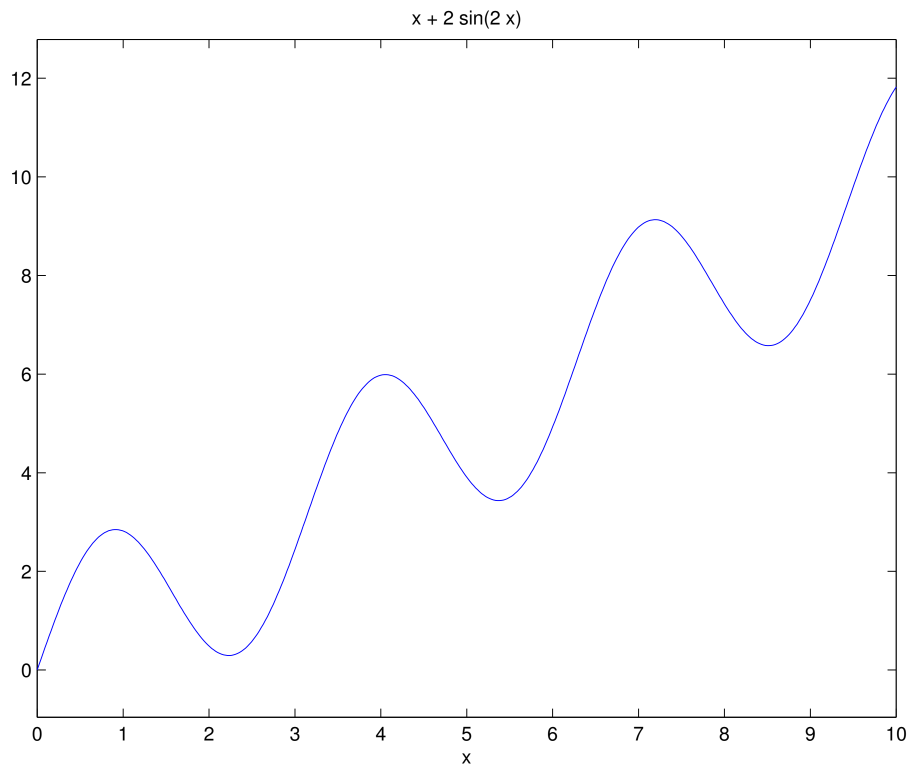
Surfaces of revolution are best represented in parametric form. In the following example, the curve $x = \sin^2(\pi x)$ is rotated about the z-axis on the interval $-1 \le z \le 1$.

```
>> ezsurf('sin(pi*u)*sin(pi*u)*cos(v)',...
>>        'sin(pi*u)*sin(pi*u)*sin(v)', 'u', [-1 1 0 2*pi]);
>> set(gca, 'DataAspectRatio', [1 1 1]);
>> view(135,15);                                     (Ex 13) See Results
```

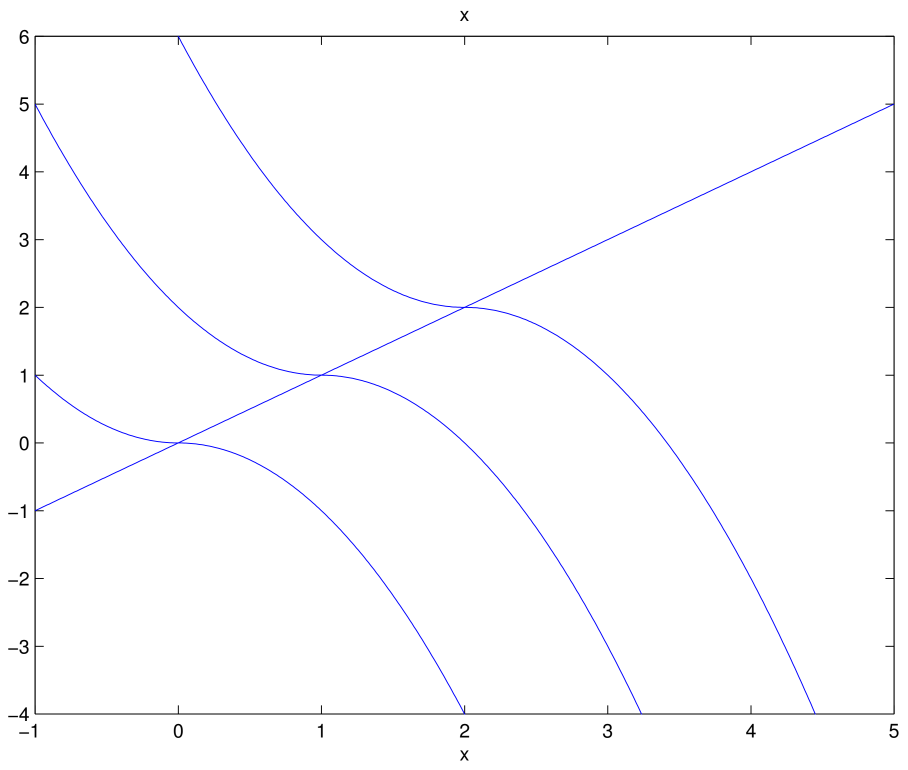
An example using `contour` and `surf`

To achieve better control over the graphics output, it is best to work directly with the more primitive functions, `plot`, `plot3`, `contour`, `mesh`, `surf`,...In this case the user must first create the data sets needed for plotting. This final example represents the function $z = \sin(\pi x)\sin(\pi x)$ both as a family of contour plots and as a surface using the matlab function `surfc`. The example shows how one can add labels to the contours using the `clabel` function, and also illustrates how to combine multiple plots into a single graphics window using the `subplot` function.

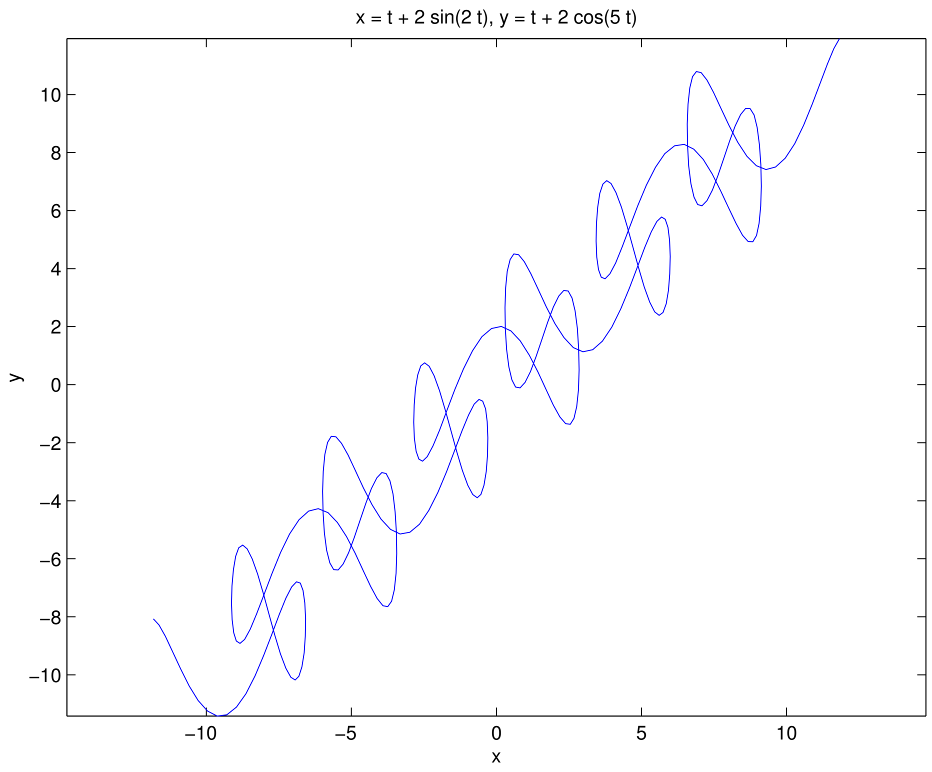
```
>> x = [-1 : 1/16 : 1];
>> y = [-1 : 1/16 : 1];
>> [x2,y2] = meshgrid(x,y);
>> z = sin(pi*x2) .* sin(pi*y2);
>> subplot(1,2,1);
>> [c,h] = contour(x, y, z, [-1 : 0.125 : 1]);
>> clabel(c, h, [-1 : .25 : 1]);
>> set(gca, 'DataAspectRatio', [1 1 1]);
>> title('sin(\pi x) sin(\pi y)');
>> subplot(1,2,2);
>> surfc(x, y, z);
>> view(125,30);
>> xlabel('x');  ylabel('y');  zlabel('z');
>> set(gca, 'DataAspectRatio', [1 1 1]);             (Ex 14) See Results
```



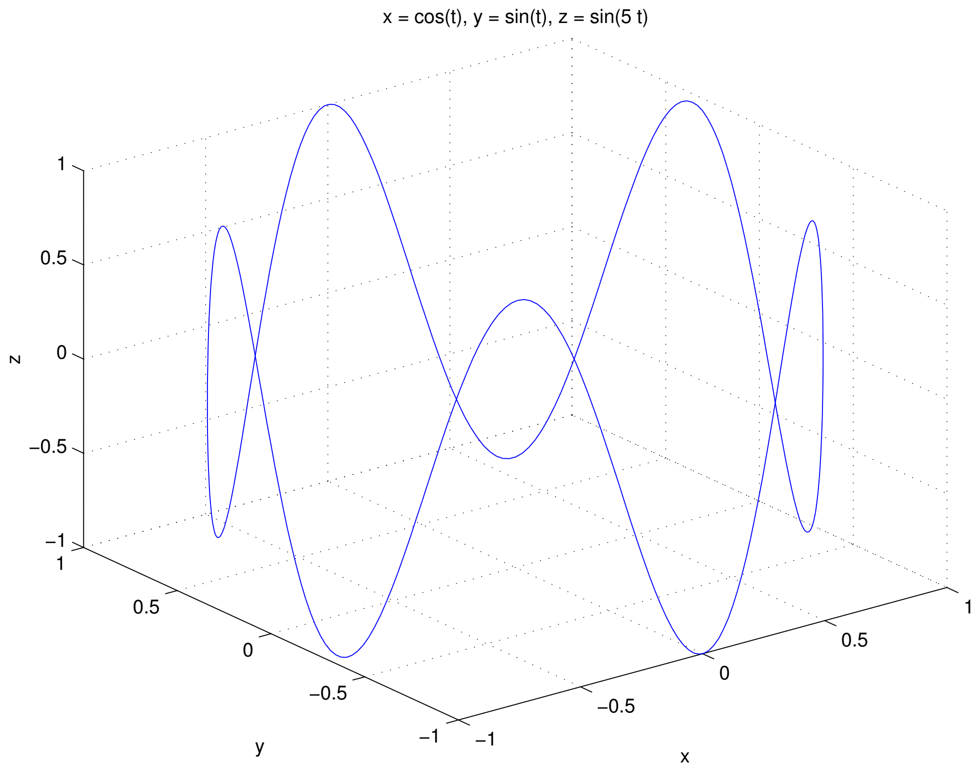
Result for Example 1. *Go Back*



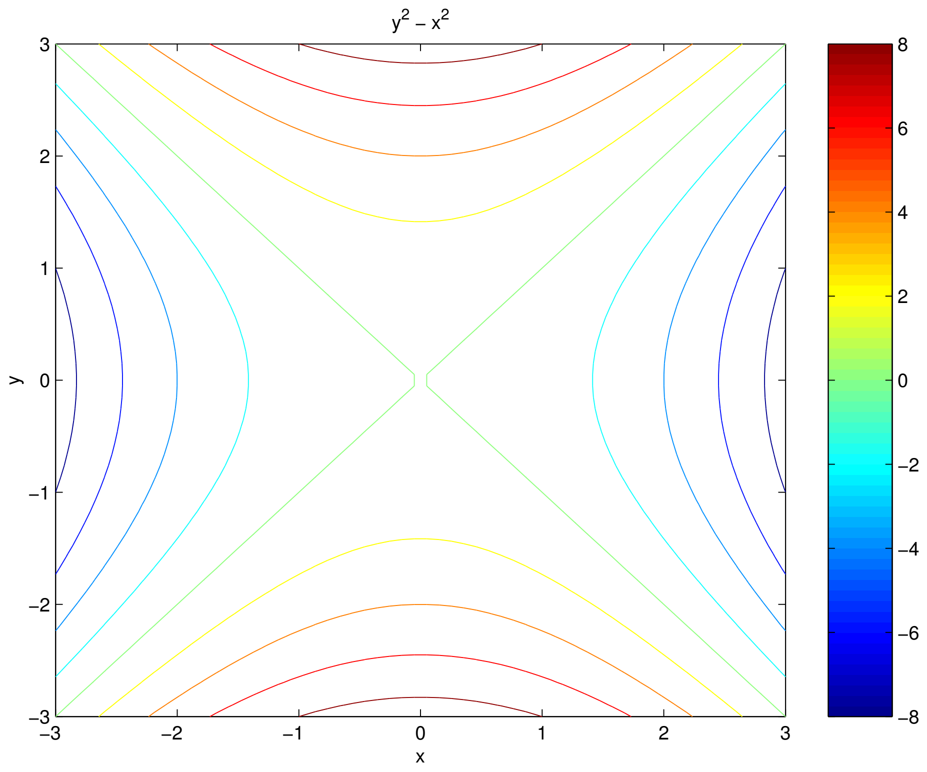
Result for Example 2. *Go Back*



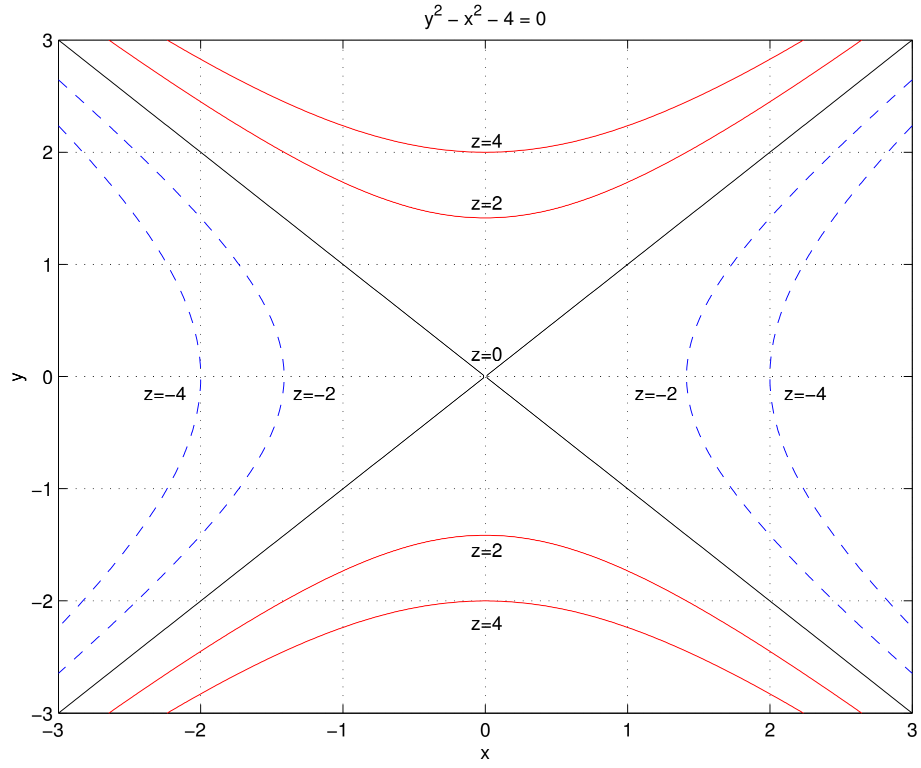
Result for Example 3. *Go Back*



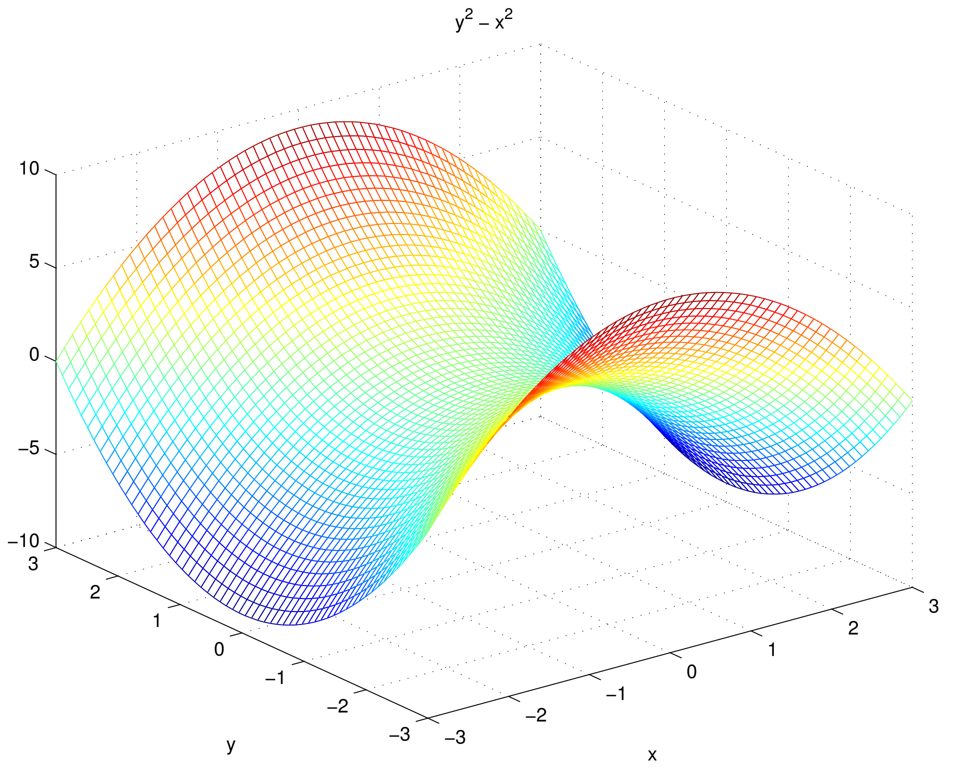
Result for Example 4. *Go Back*



Result for Example 5. *Go Back*

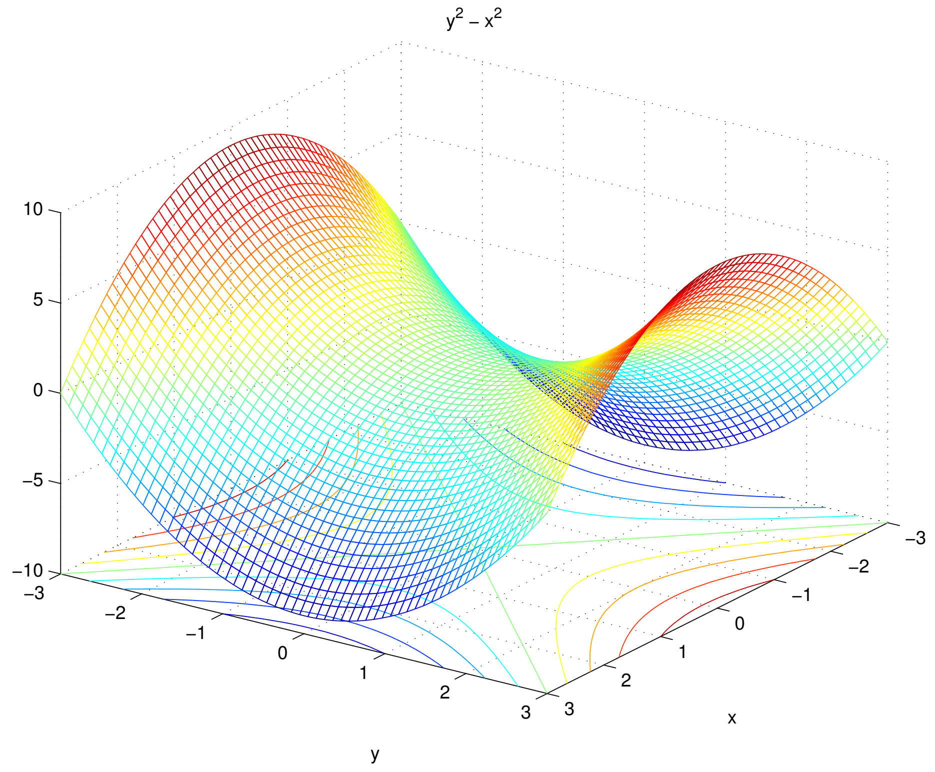


Result for Example 6. *Go Back*



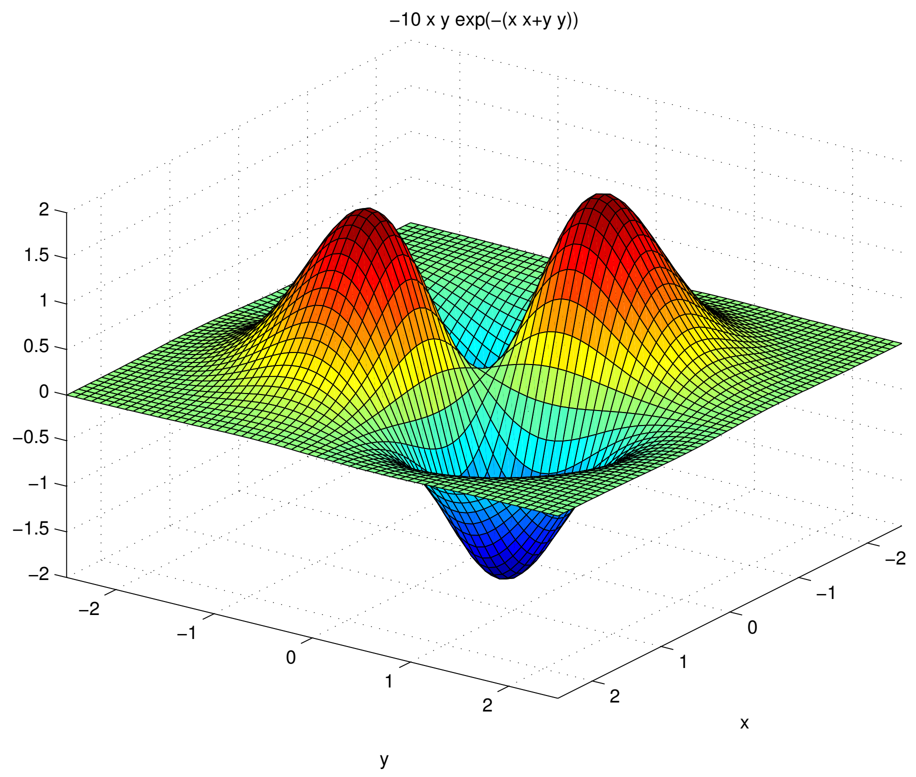
Result for Example 7.

Go Back

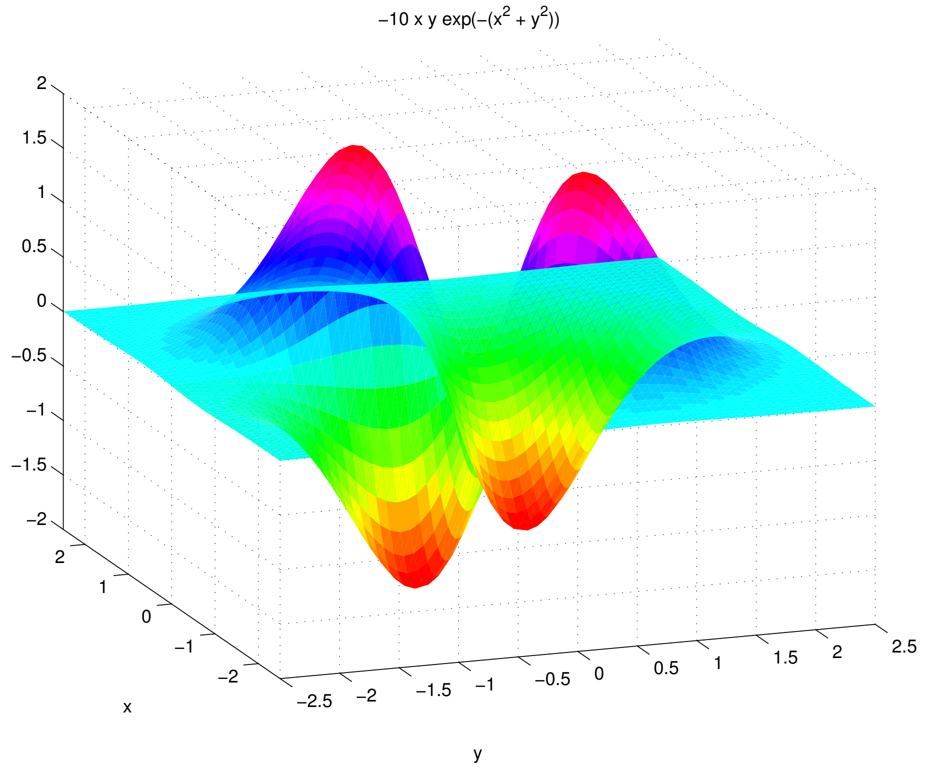


Result for Example 8.

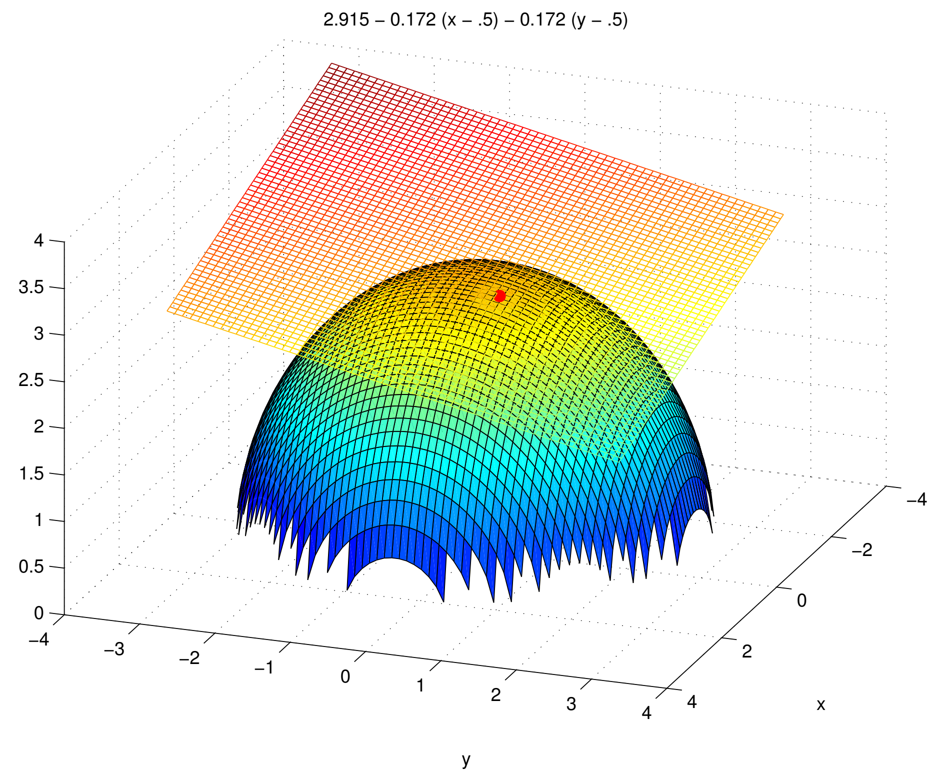
Go Back



Result for Example 9. *Go Back*

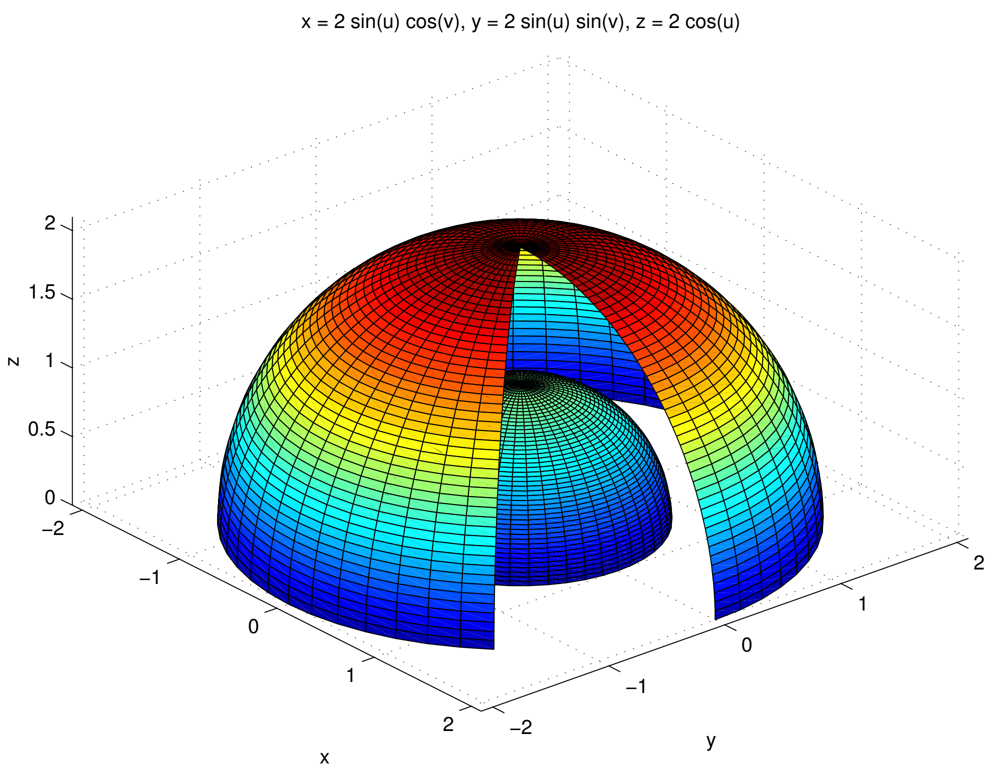


Result for Example 10. *Go Back*



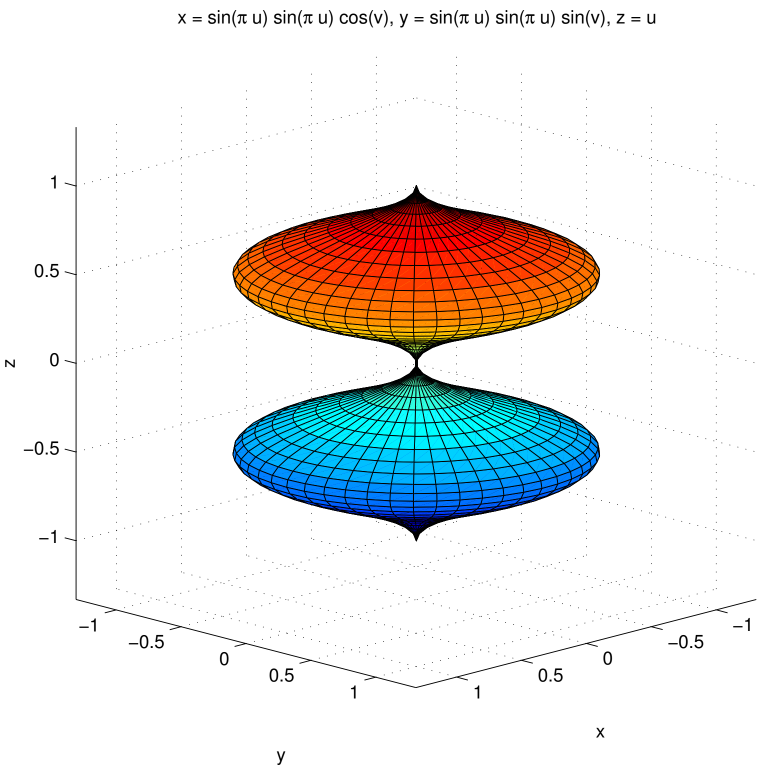
Result for Example 11.

Go Back

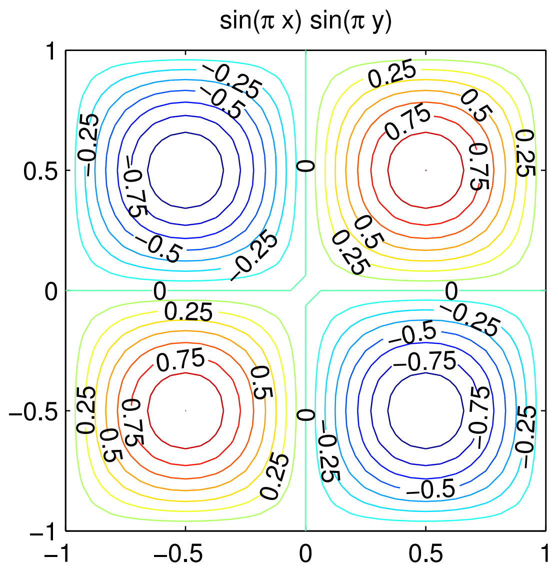


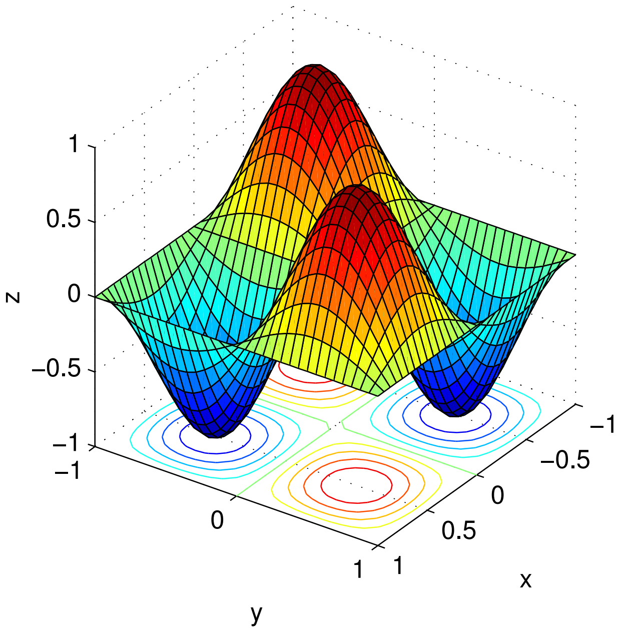
Result for Example 12.

Go Back



Result for Example 13. *Go Back*





Result for Example 14. *Go Back*