Info to SimplexNumerica

Surface Plots

V23

Introduction

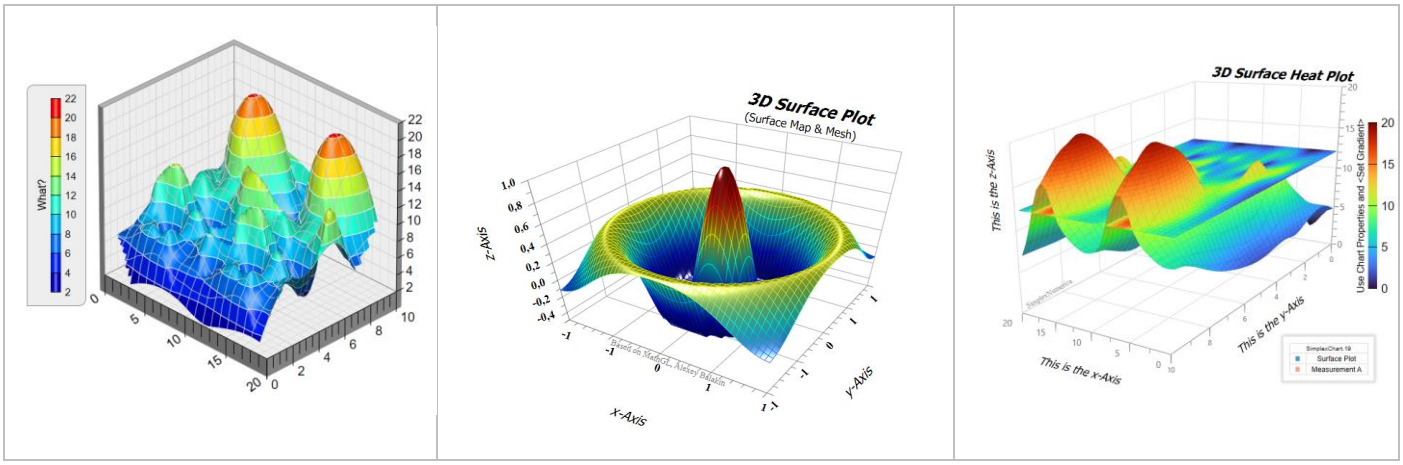
Three-dimensional (3-D) plots can be a useful way to present data that consists of more than two variables. *SimplexNumerica* provides various options for displaying three-dimensional data, like line and wire, surface, mesh plots, and many others. The plots can also be formatted to have a specific appearance and special effects. Only a few of the three-dimensional plotting features are described here in this White Paper. Additional information can be found in the main manual.

Data Visualization

Surface plots are charts of three-dimensional data. Rather than showing the individual data points, surface plots show a functional relationship between a designated dependent variable (y-Axis), and two independent variables (x-Axis and z-Axis). The plot is a companion plot to the Contour Plot.

It is important to understand how these plots are constructed. A two-dimensional grid of x and y is constructed. The range of this grid is equal to the range of the data. Next, a z value is calculated for each grid point. This z value is like a weighted average of all data values that are "near" this grid point. But, to make it more precisely, *SimplexNumerica* used a triangulation or interpolation algorithm to find the best approach to the data. The three-dimensional Surface Plot is constructed using these fit values. Hence, the Surface Plot does not show the variation at each grid point. For that, you should use a 3D Scatter Plot.

Here three different chart types for Surface Plots:

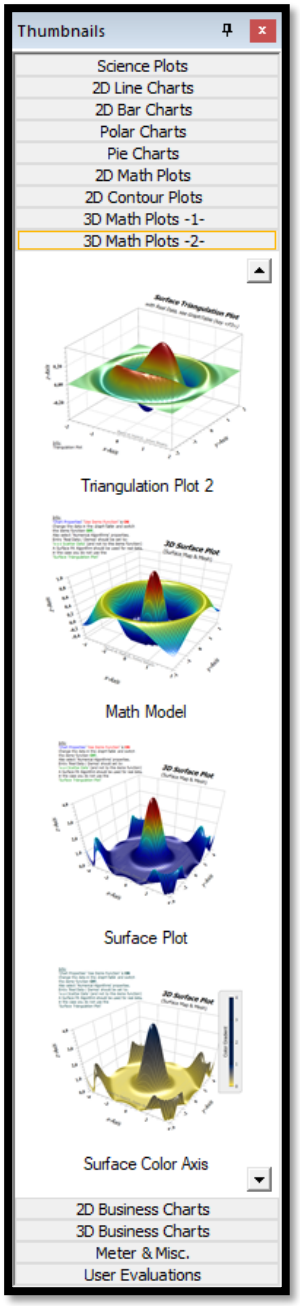


Demo Mode / Real Mode

To have a nicer impression, *SimplexNumerica* shows - in the Thumbnail Window - some of the pictures based on the demo mode. In demo mode, the data are right generated by a mathematical function, like

```
f(x,y) = 0.6 * sin(2 * π * x) * sin(3 * π * y) + 0.4 * cos(3 * π * x * y); // π = PI = 3.14
```

Surface Plots

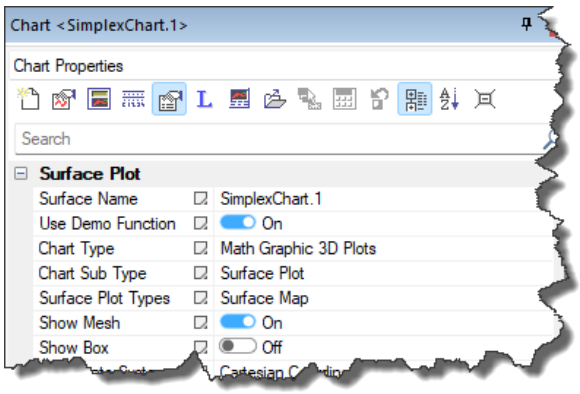


Here the Thumbnail Window (expanded in the width) that shows the rubric *3D Math Plots* (page 2)

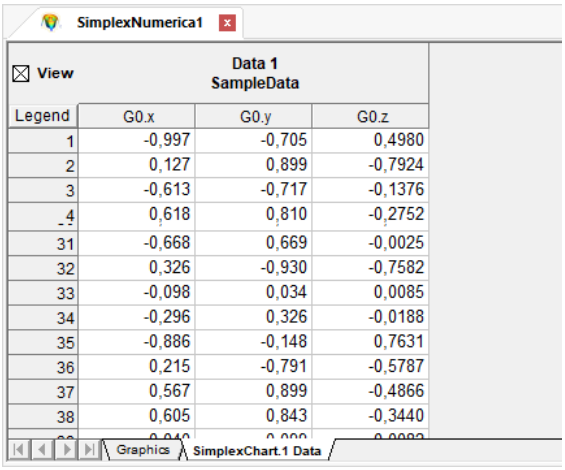
As you can see, there are many symbols with different settings for a Surface Plot and other related chart types.

To switch-off the demo mode, use the properties:

1. If 'Chart Properties' 'Use Demo Function' is ON

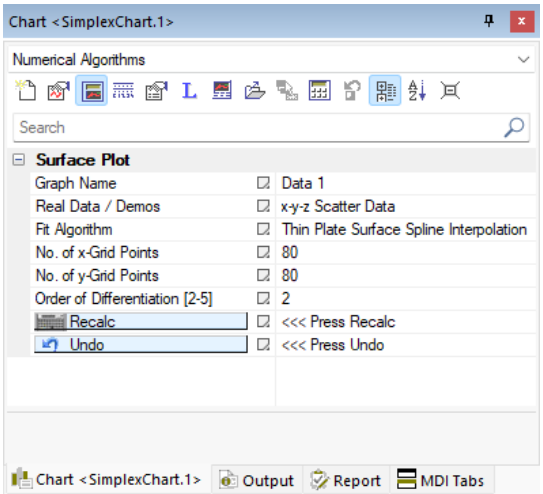


2. Change the data in the GraphTable and then switch the demo function OFF.



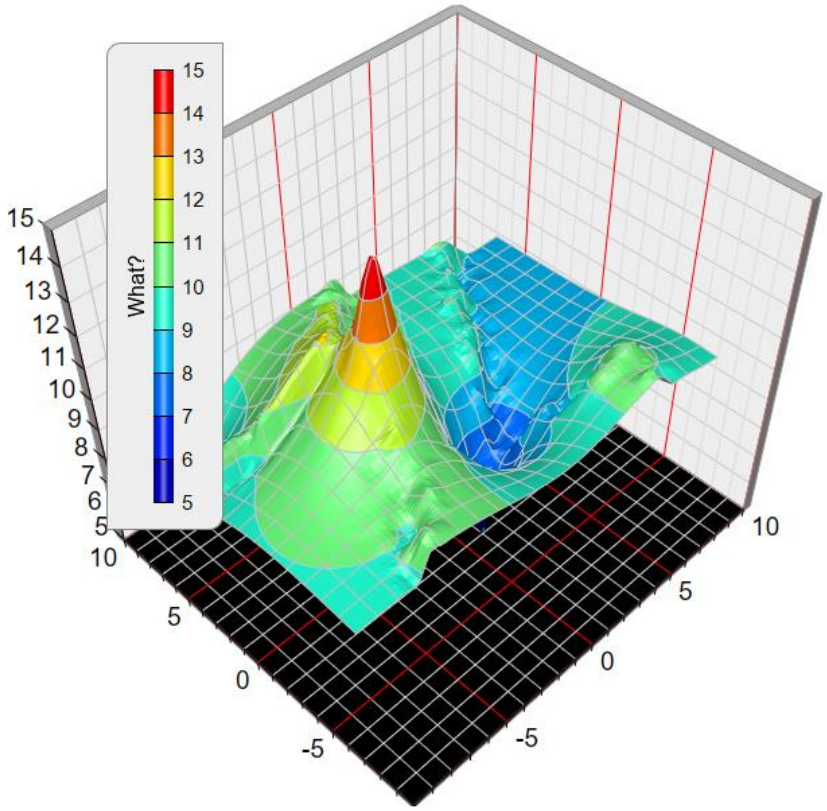
Edit your x/y/z data and go back to the Graphics View (key <F3>)

3. Select 'Numerical Algorithms' properties.

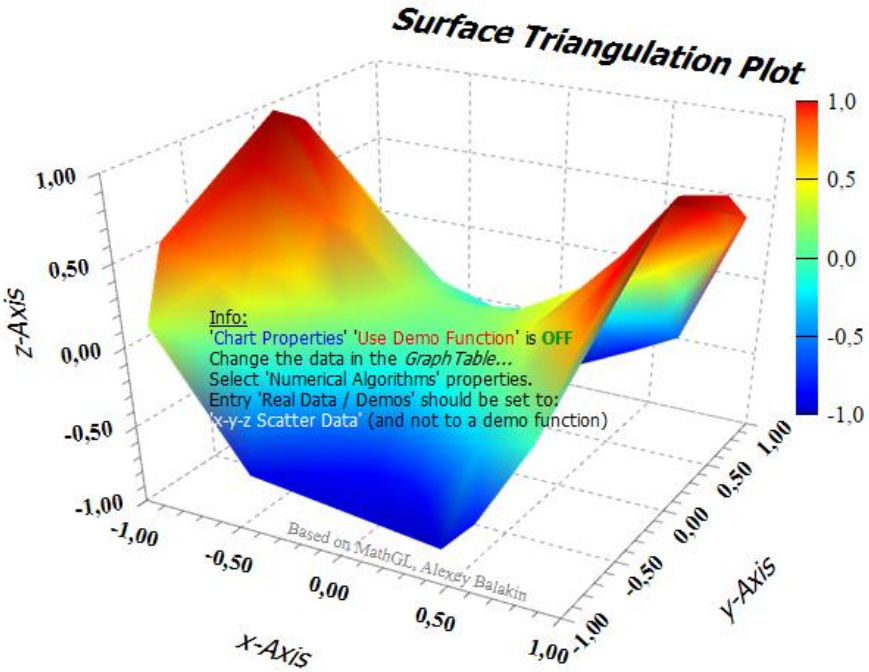


Surface Plots

4. Entry 'Real Data / Demos' should be set to: **'x-y-z Scatter Data'** (and not to this demo function)
5. A Surface Fit Algorithm should be used for real data, in the case you do not use the 'Surface Triangulation Plot'
6. Press button '**Recalc**'



This picture above shows real data interpolated by the '**Built-in Spline Interpolation**'



Another method to fit real data is by using the '**Surface Triangulation Plot**'

Surface Plots

By using the 'Surface Triangulation Plot' no interpolation is needed.

A triangulation of a compact surface is a finite collection of triangles that cover the surface in such a way that every point on the surface is in a triangle, and the intersection of any two triangles is either void, a common edge or a common vertex. A triangulated surface is also called tri-surface.

Data Structure

A surface plot is constructed from three variables. The x and y (independent) variables are shown on the horizontal axes. The z variable is shown along the vertical axis. Note that all three variables are (numeric) real values (in C++ the *double* type).

Here is how *SimplexNumerica* calculates a Surface Plot from a mathematical (demo) function (in C++):

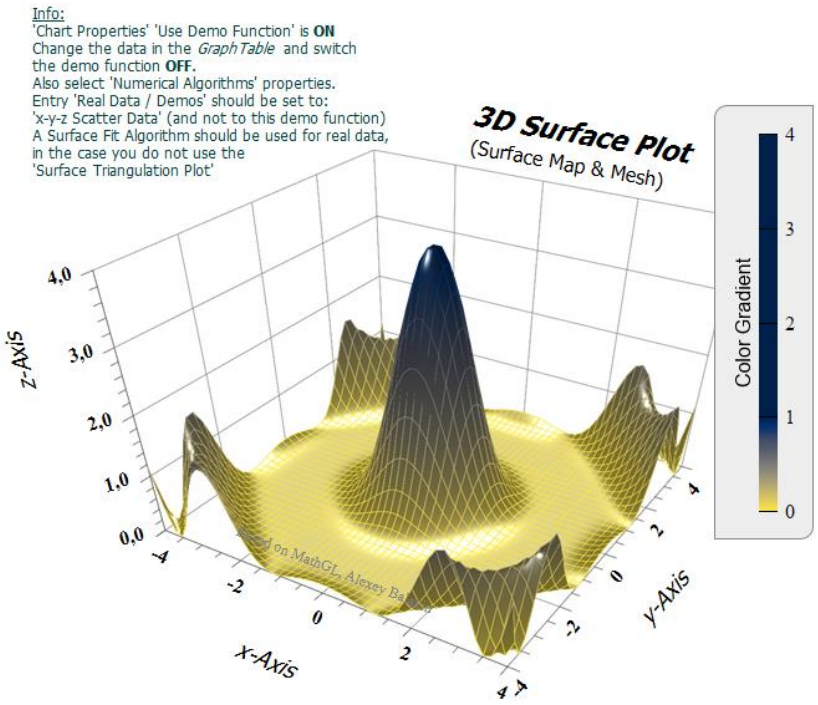
```
//Info: d3_xmin/d3_xmax are the chart border x limits; dto. for y-Axis
// Dim of the Grid Points
long NumberOfXDataPoints = 50; // get this from your program...
long NumberOfYDataPoints = 40;

double d3_xmin, d3_xmax, d3_ymin, d3_ymax;
// get this from your program...

for (long j = 0; j < NumberOfYDataPoints; j++)
{
   double y = d3_ymin + j * ((d3_ymax - d3_ymin) / (NumberOfYDataPoints - 1));

   for (long i = 0; i < NumberOfXDataPoints; i++)
   {
      double x = d3_xmin + i * ((d3_xmax - d3_xmin) / (NumberOfXDataPoints - 1));

      long i0 = i + NumberOfXDataPoints * j;

      double z = Get3DZValueFromFunction(SelectedSurface, x, y);

      //
      // Proceed with x, y, z...
      //
   }
}

double Get3DZValueFromFunction(_tagSurfaceFunctionNames SelectedSurface, double x, double y)
{
   double zValue;

   try
   {
      switch (SelectedSurface)
      {
         case surface_math_a:
            zValue = 0.6 * sin(2 * M_PI * x) * sin(3 * M_PI * y) + 0.4 * cos(3 * M_PI * x * y);
            break;
         case ...
            // more examples....
      }
   }
   catch
   {
      //....
   }
```

Surface Plots

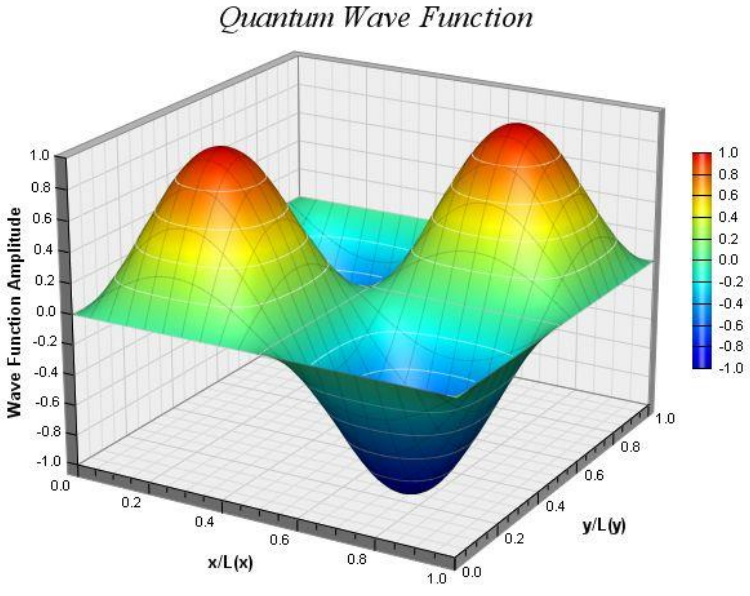
```
   zValue = __max(zValue, g.surface_interval.d3_zmin);
   zValue = __min(zValue, g.surface_interval.d3_zmax);

   return zValue;
}
```

Examples

Here you will find some examples for Surface Plots, integrated in SimplexNumerica, but please have a look to the Thumbnail Window *3D Math Plots* in SimplexNumerica, here you will find lot more of 3D variations.

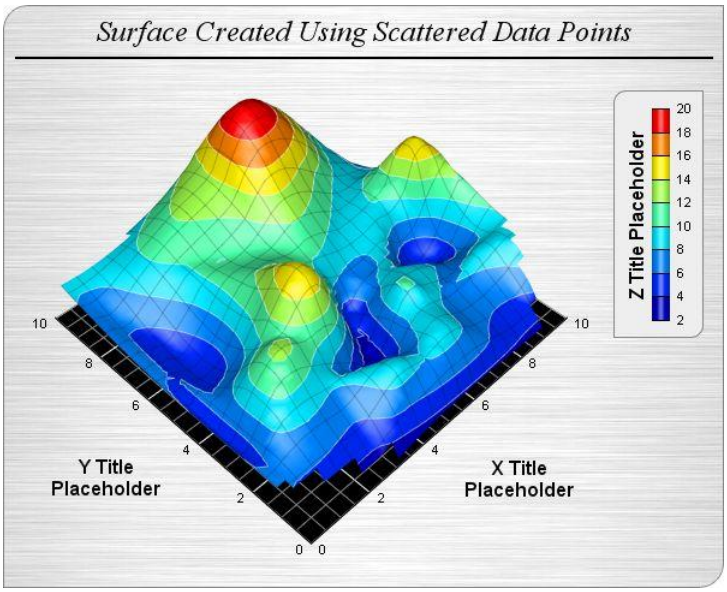


A surface chart with lighting effects. The smooth surface is created using spline surface fitting of the data points.

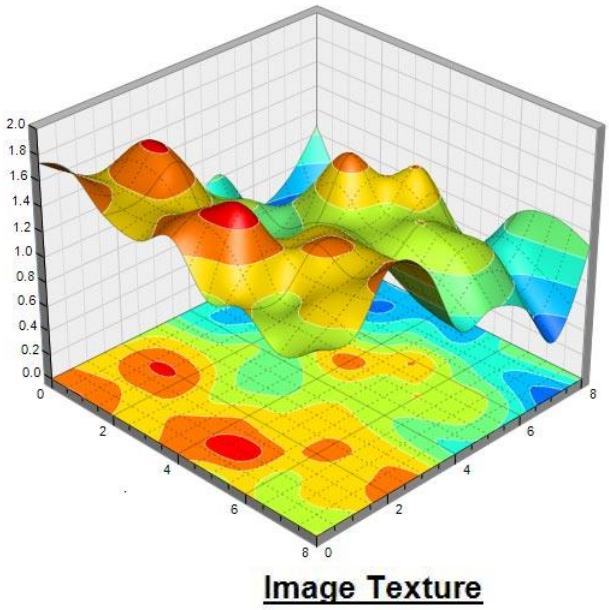


A surface chart with continuous coloring and lighting effects, and with white contour lines and grey grid lines. The smooth surface is created using spline surface fitting of the data points.

Surface Plots

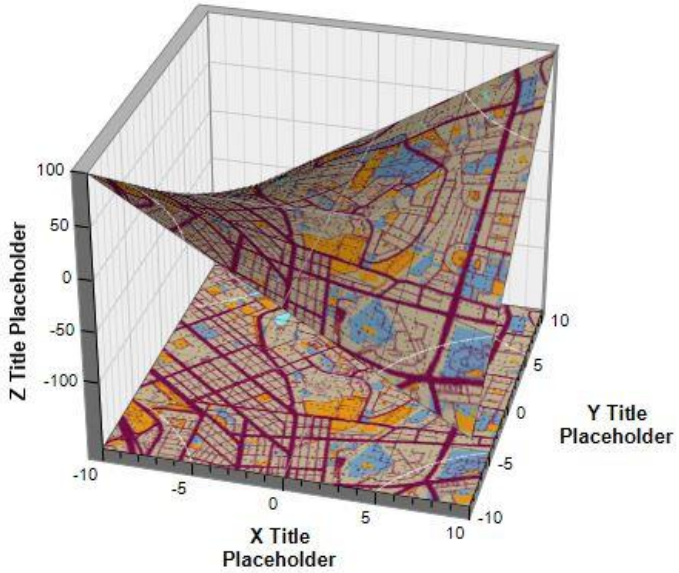


A surface chart created using scattered data (data points are randomly distributed rather than lying on a grid), with the vertical walls hidden.



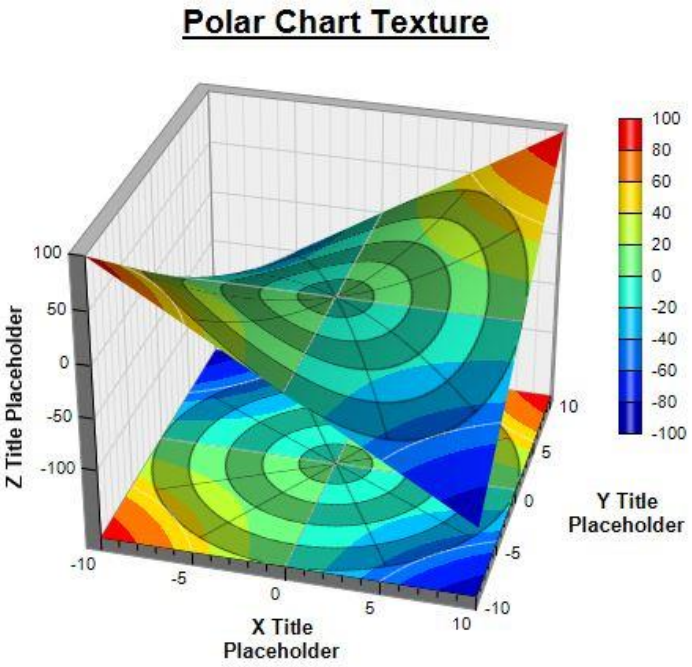
Surface Projection: The surface can be projected onto the XY plane to visualize the contour chart of the surface.

Image Texture



Custom Surface Texture: For maximum flexibility, you can overlay a custom image on the surface. With this method, you can color the surface in any way you like.

Surface Plots



Dynamic Image as Texture: The texture image can be dynamically generated. In this example, a semi-transparent polar chart is overlaid on the surface.

That's it for now!