Plotting Themes in Maple 2026

Maple 2026 introduces enhanced support for reusable plotting styles through a new theme option. Maple has long provided extensive control over plotting options, but achieving a consistent visual style typically requires specifying multiple settings. The new theme mechanism allows commonly used combinations of options to be bundled and reused, simplifying the creation of consistently formatted visualizations.

With themes, you can define the look you want once and apply it across multiple plots with a single option. This makes it easier to maintain visual consistency in worksheets, teaching materials, reports, and publications, while still allowing individual plot options to be overridden when needed.

Themes support both 2-D and 3-D plots and can be queried or customized to meet specific requirements.

New Plotting Themes Option

The new plotting theme option allows multiple individual plot settings to be associated with a common name so that they can be easily reused together.

- There are a number of stock themes that can be utilized directly. Custom user-defined themes can also be constructed.
- The various options specified by a theme are grouped according to whether the target plot is two-dimensional (2-D) or three-dimensional (3-D).
- Defined themes can be queried for their contents. The commands used to query or construct a theme are exports of the plots package.

Listing Available Themes

The stock themes can be listed as follows. This returns the names of the available predefined themes.

```
> with(plots):
```

```
> getthemes();
```

```
"DarkGrayGrid", "ExperimentalData", "SpreadsheetPoints", "grayscale", "SpreadsheetLines",      (1)
    "GrayGrid", "DarkGridlines", "reverse"
```

Querying a Theme

Here is an example of querying the contents of a theme. This is one of the stock themes.

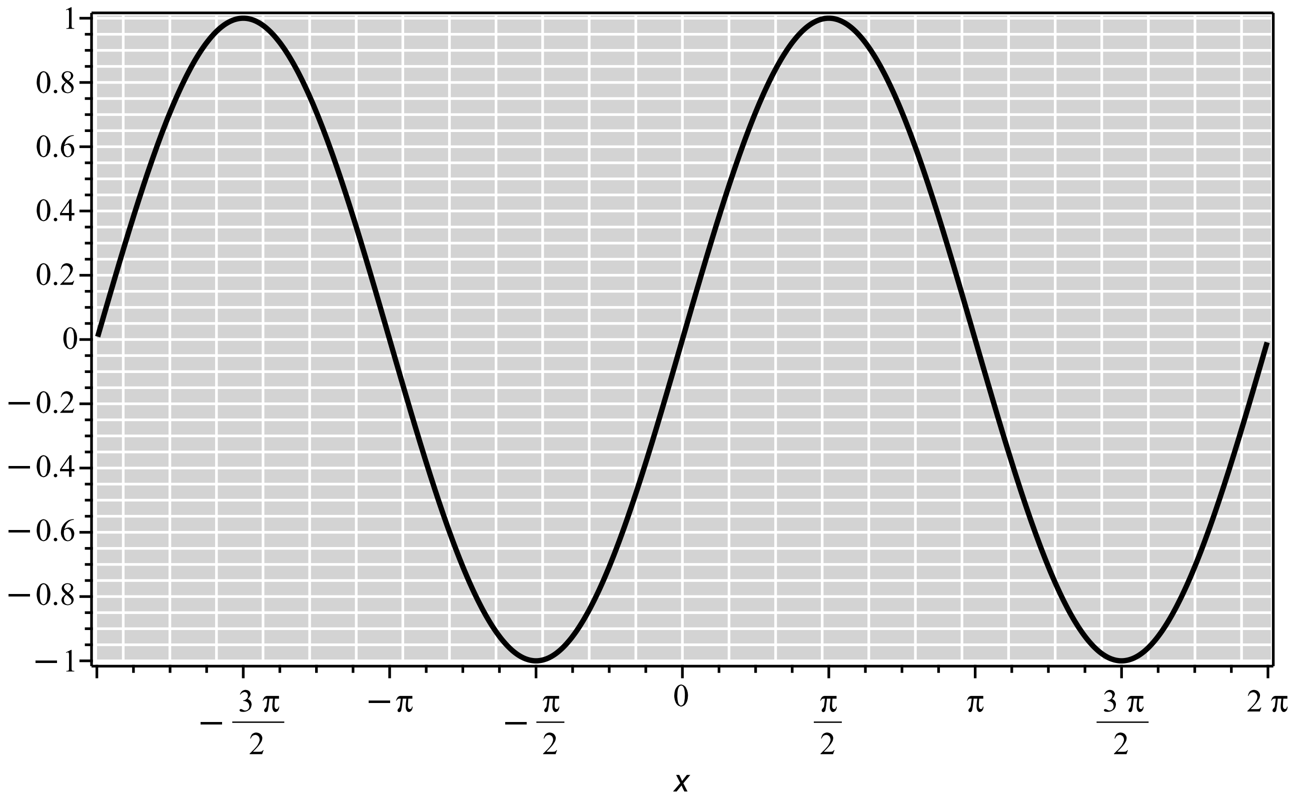
```
> gettheme( GrayGrid );
```

$$\text{table}\left(\left[2 = \left[axes = \text{"box"}, axesfont = [\text{"Arial"}, 10], axis_1 = [tickmarks = [10, subticks = 3], color = black, gridlines = [10, color = \text{"white"}]], axis_2 = [tickmarks = [10, subticks = 3], color = \text{"black"}, gridlines = [10, color = \text{"white"}]], background = \text{"LightGrey"}, color = [\text{"Black"}, \text{"Dalton Orange"}, \text{"Dalton SkyBlue"}, \text{"Dalton BluishGreen"}, \text{"Dalton Yellow"}, \text{"Dalton Blue"}, \text{"Dalton Vermillion"}, \text{"Dalton ReddishPurple"}], labeldirections = [\text{"horizontal"}, \text{"vertical"}], labelfont = [\text{"Arial"}], size = [712.00, 400], thickness = 2\right], 3 = [\ ]\right]\right) \tag{2}$$

Applying a Theme

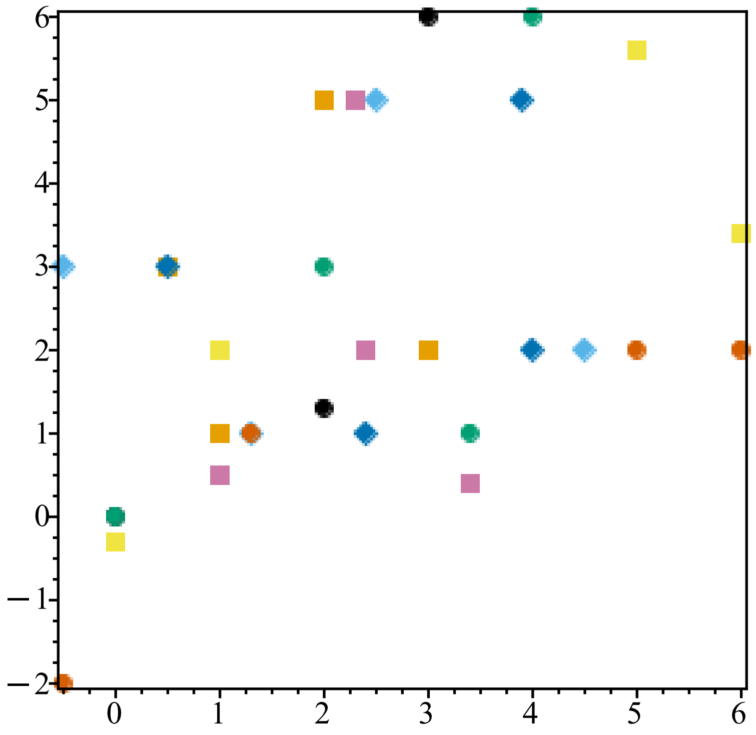
If the theme name is supplied as an option, the individual options it represents are applied automatically. Additional options passed to the plotting command override those defined by the theme.

```
> plot( sin(x), theme=GrayGrid );
```

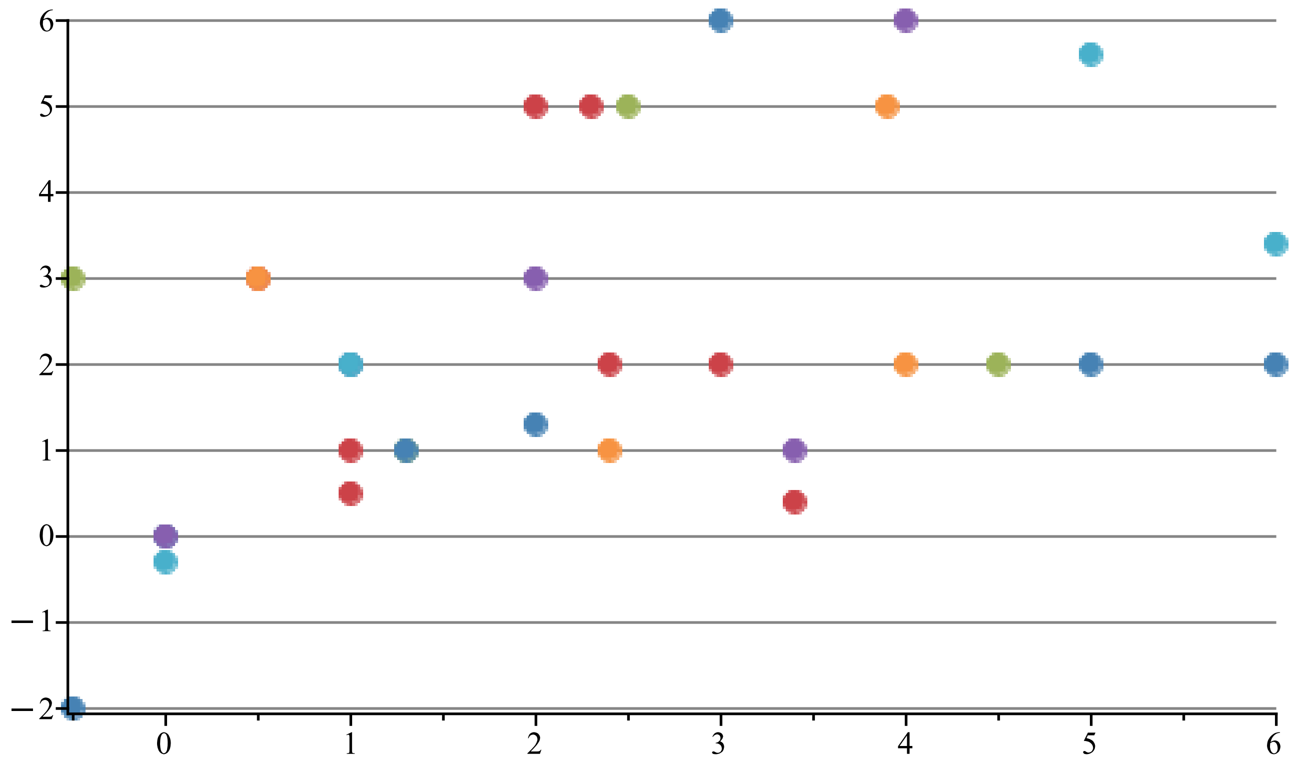


Here are some examples using some of the stock themes:

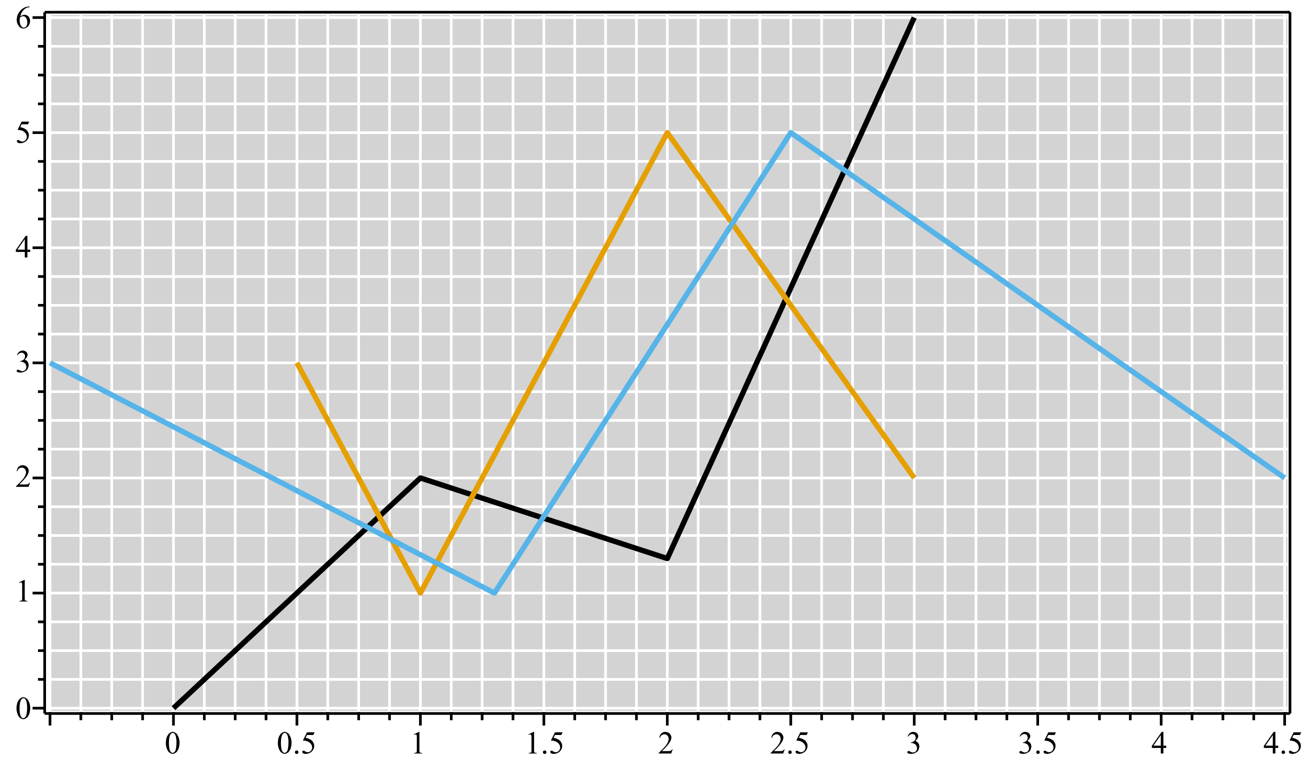
```
> data:=[ [[0,0],[1,2],[2,1.3],[3,6]],
         [[0.5,3],[1,1],[2,5],[3,2]],
         [[-0.5,3],[1.3,1],[2.5,5],[4.5,2]],
         [[0,0],[2,3],[3.4,1],[4,6]],
         [[0,-0.3],[1,2],[5,5.6],[6,3.4]],
         [[0.5,3],[2.4,1],[3.9,5],[4,2]],
         [[-0.5,-2],[1.3,1],[5,2],[6,2]],
         [[1,0.5],[2.3,5],[3.4,0.4],[2.4,2]] ]:

  > plot(data, theme=ExperimentalData);
```



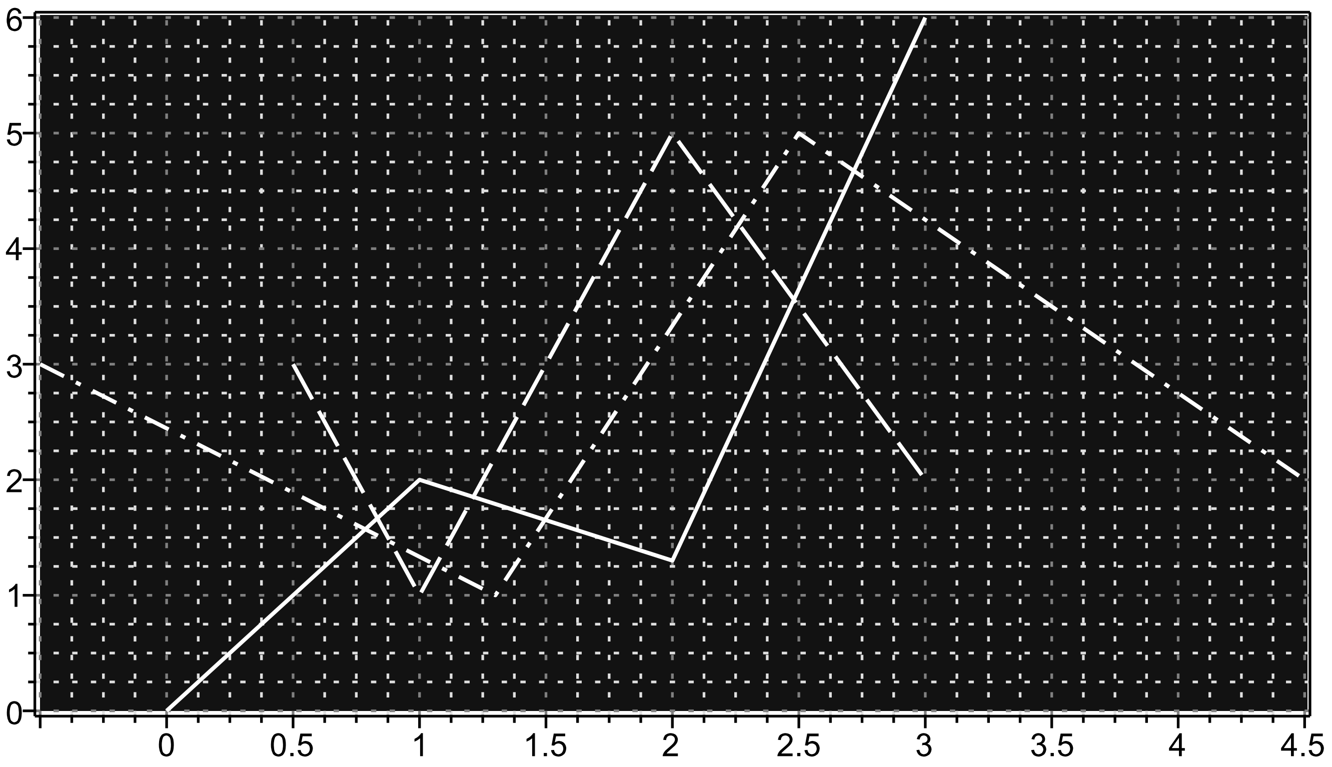
```
> plot(data, theme=SpreadsheetPoints);
```



```
> plot([data[1], data[2], data[3]], theme=GrayGrid);
```



```
> plot([data[1], data[2], data[3]], theme=DarkGridlines);
```



Creating a Custom Theme

Here are some examples of constructing a user-defined theme.

Example 1

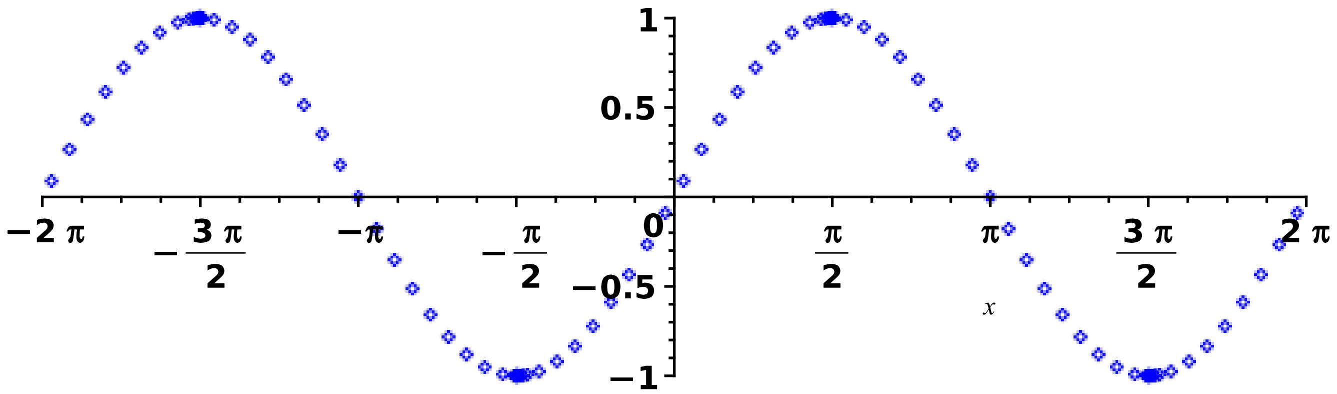
In this example, the two supplied lists correspond to options for 2-D and 3-D plots respectively.

```
> createtheme(MyTheme,
            [color=blue, style=point, size=[700,200], font=
  ["Helvetica",bold,12]],
            [shading=zhue, style=surfacecontour, size=[400,450],
  font=["Helvetica",12]]);
```

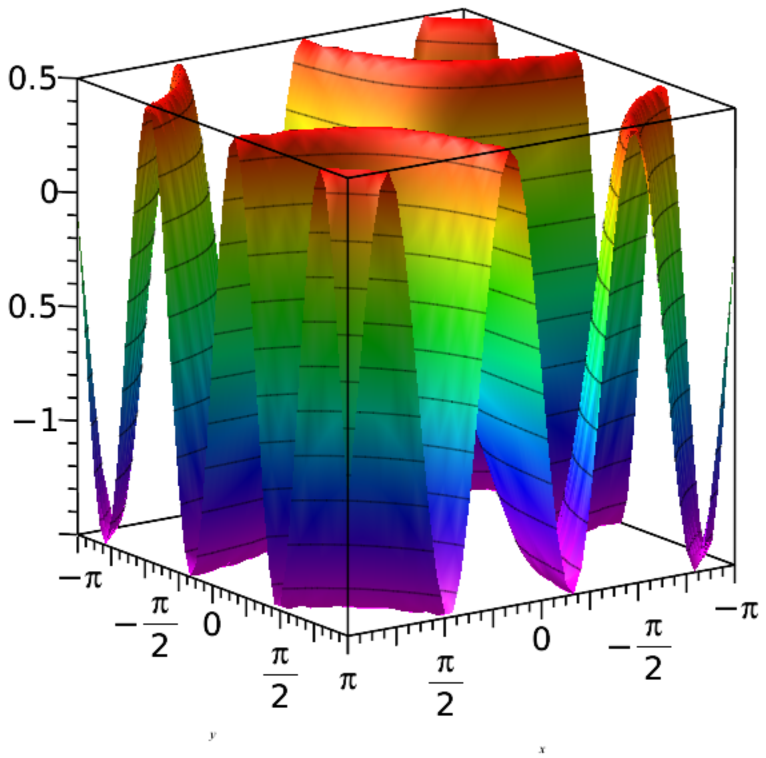
$\mathrm{table}([2 = [\mathit{color} = \mathit{blue}, \mathit{style} = \mathit{point}, \mathit{size} = [700, 200], \mathit{font} = [\text{"Helvetica"}, \mathit{bold}, 12]], 3 = [\mathit{shading} = \mathit{zhue}, \mathit{style} = \mathit{surfacecontour}, \mathit{size} = [400, 450], \mathit{font} = [\text{"Helvetica"}, 12]]])$ **(3)**

Once defined, the theme can be used in plotting commands:

```
> plot( sin(x), theme=MyTheme, numpoints=70 );
```



```
> plot3d( sin(x*y)-1/2, x=-Pi..Pi, y=-Pi..Pi, theme=MyTheme );
```



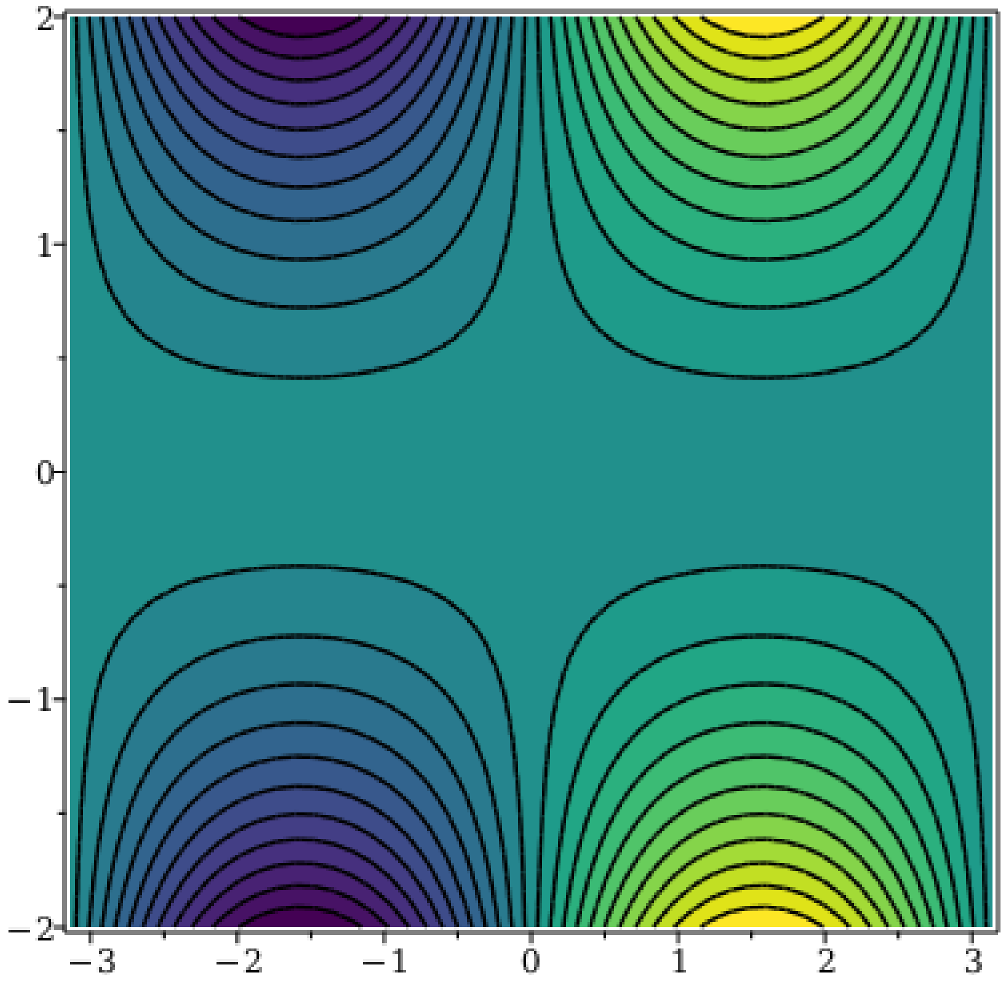
Example 2:

In this example, we apply a custom theme to a contour plot.

```
> createtheme(Cont01,[colorbar=false, contours=22, filled=true,
  color=black, axes=boxed,
                    contourlabels=false, colorscheme="Viridis",
  font=["Times",bold]],[]);
```

table([2 = [*colorbar* = *false*, *contours* = 22, *filled* = *true*, *color* = *black*, *axes* = *boxed*, *contourlabels* = *false*, *colorscheme* = "Viridis", *font* = ["Times", *bold*]], 3 = [ ]]) **(4)**

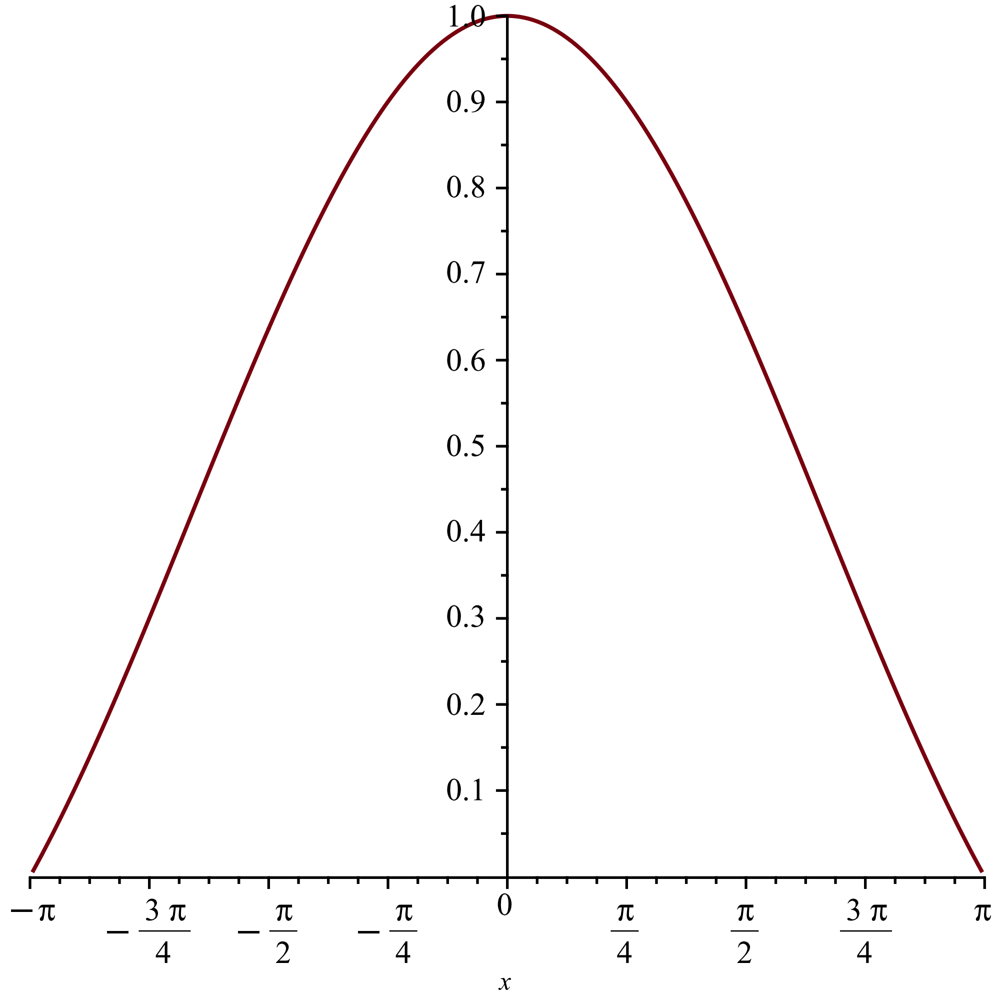
```
> contourplot(sin(x)*y^2, x=-Pi..Pi, y=-2..2, theme=Cont01);
```



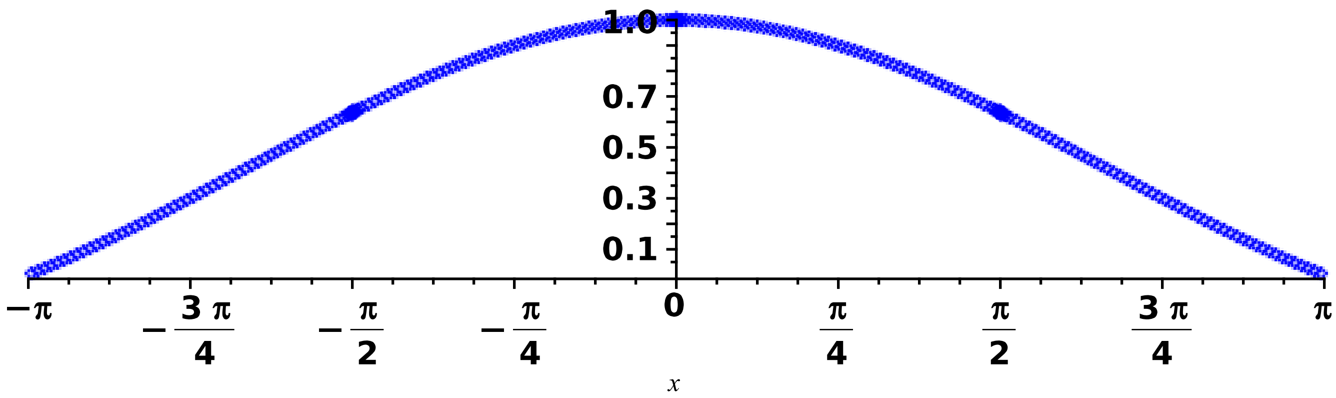
Applying a Theme to Existing Plots

The theme can also be supplied to the **plots:-display** command, to modify an existing plot that was otherwise created with using the theme.

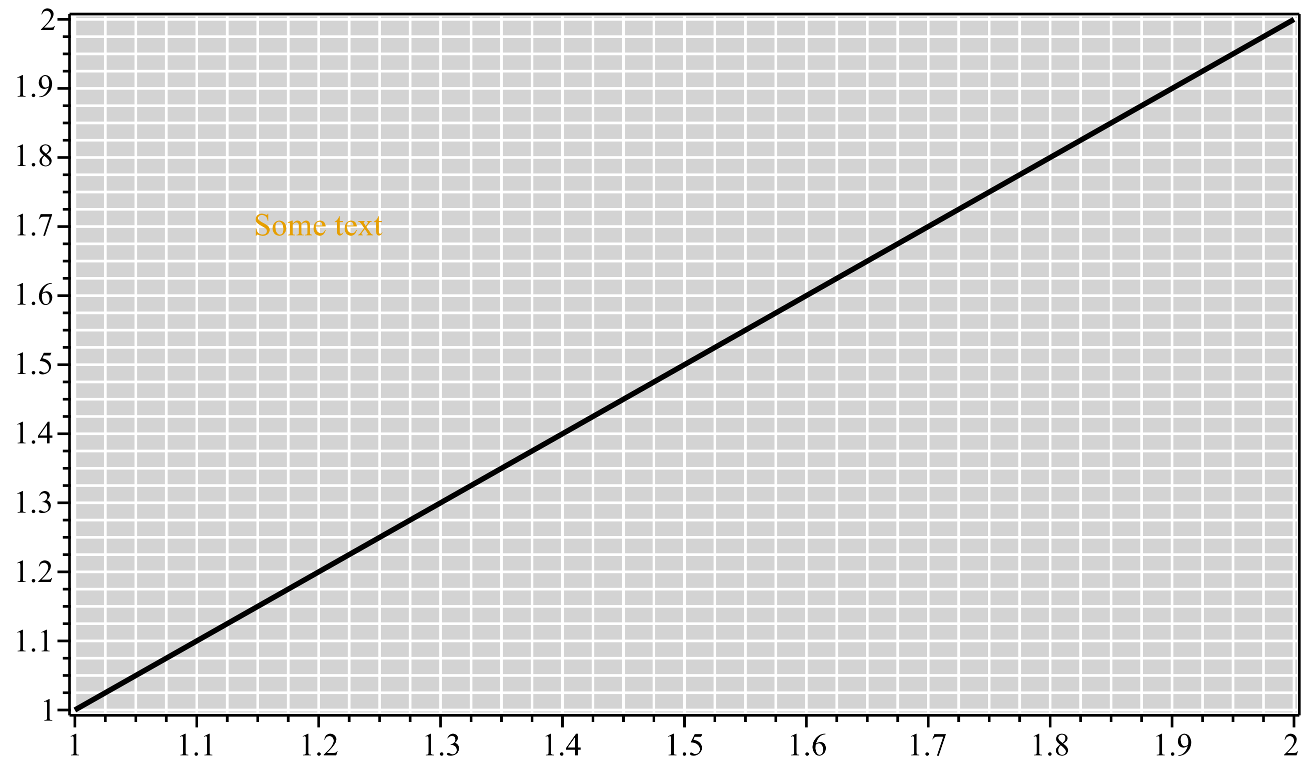
```
> P := plot(sin(x)/x, x=-Pi..Pi);
```



```
> display(P, theme=MyTheme);
```



```
> display(plottools:-line([1,1],[2,2]),
        textplot([1.2,1.7,"Some text"]),
        theme=GrayGrid);
```



Supported Commands

At present the plotting commands which support themes include plot, plot3d, implicitplot, densityplot, and plots:-display.