User Guide

Contents

1 Introduction

TeraPlot LT brings powerful 2D and 3D graphing capabilities to a Windows Store app. Using TeraPlot LT, you can create 2D line plots and 3D surface plots in various coordinate systems, based on either mathematical expressions or supplied data.

The program comes with a wide range of graph/plot tailoring and layout features to create visually appealing graphs. Using TeraPlot LT, you can:

- Create multi-page projects containing your graphs.
- Use different row/column layouts on each page.
- Mix 2D and 3D graphs on the same page.
- Cut/paste graphs within a page and between pages.
- Rearrange graphs on a page after they've been created.
- Rearrange project pages after they've been created.
- Customize graph colors (background, axis lines, axis labels, etc.).
- Export graphs as images to a file or the clipboard.
- Specify graph axis tick positioning and labelling.
- Add graph annotation in the form of captions, color keys, legends.
- Zoom, pan, and animate 3D graphs.
- Create cartesian, polar, cylindrical or parametric 3D plots with various surface coloring options, such as solid color, colormapped, or image overlay.
- Create cartesian, polar, or parametric 2D plots with various line style options.

The following sections describe how to use the various TeraPlot LT features.

2 Overview

A TeraPlot LT project consists of a set of pages, with each page containing a number of graphs. The number of pages in the project and the number and layout of graphs on each page are determined by you, the only limitations being that the number of pages can't be less than 1, and the number of graphs per page can't be more than 16. Graphs can be laid out on a page in any row, column format up to 4x4, and 2D and 3D graphs can be placed on the same page. Pages are added, deleted, and reordered via the controls on the top app bar. All other options are accessed via the menus on the bottom app bar.

Project creation, loading, and saving is handled via the menu items on the File menu. The names of recently opened projects are added to the bottom of the File menu,

allowing quick access to commonly used projects. A Demo Project menu item is also available on the File menu. This can be used to open a demo project containing several pages of example graphs. Opening and examining this project can be useful in learning how to use the software. You're free to edit the demo project, but you can't overwrite the preinstalled demo project with the edited version; Save As.. must be used to save the edited project to a new location.

When a new project is first created, it contains a single page, with a single graph holder occupying the whole page. A graph holder represents the area occupied by an as yet uninitialized graph, and contains two buttons, labelled 2D Graph and 3D Graph, as shown in Fig.1 below.

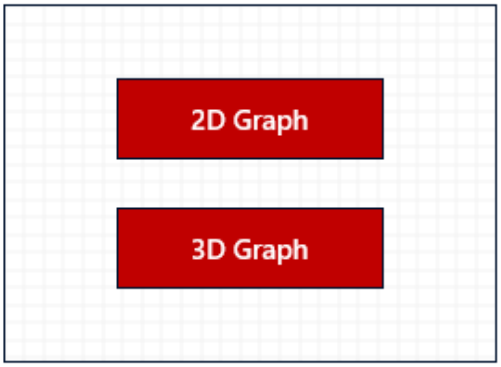

Fig. 1 – Graph Selector.

Pressing one of these buttons causes an empty (i.e. axes only, no plots) graph of the requested type to be created, as shown in fig. 2 below.

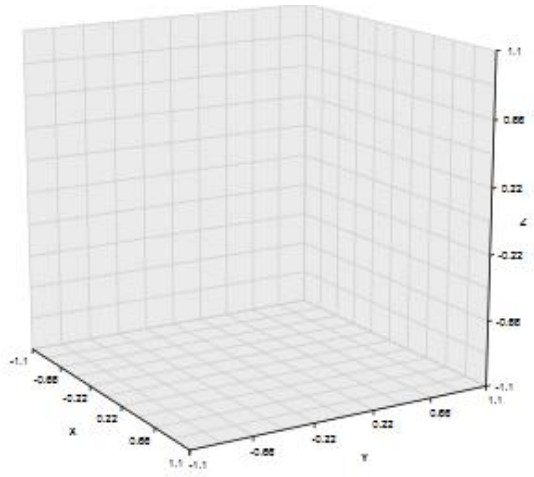

Fig.2 – After pressing 3D Graph button

2.1 Graphs and Plots

Once a graph has been created, it can be modified in various ways, using mainly the menu items from the Graph and Axes/Grid menus.

When there is more than one graph on a page, tapping anywhere within that graph makes it the selected graph, and any modifications performed will apply to that graph. Typically, these modifications are performed via a feature-related popup accessed from a menu item on the bottom app bar. For example, graph plots are edited via the Add/Remove Plots popup, accessed from the Graph menu.

2D Graphs contain Line Plots, and 3D graphs contain Surface Plots. In each case, plots can be either Analytical or Tabular. Analytical plots are defined via a script containing one or more lines of mathematical expressions. Tabular plots are defined via a table of data, which can be entered manually (2D plots only), pasted from the clipboard, or loaded from a text file. Once created, various plot features can be modified, e.g. line color and style for line plots, surface color, color map, and texture overlay for surface plots. Besides plots, various other graph features can be modified. These include the color of all graph components such as graph background, grid, axis lines, tick labels, and titles. Axis ranges, intersect (2D graphs only), and tick placement can also be modified.

Figure 3 below shows a graph containing a plot of the function z = x*y.

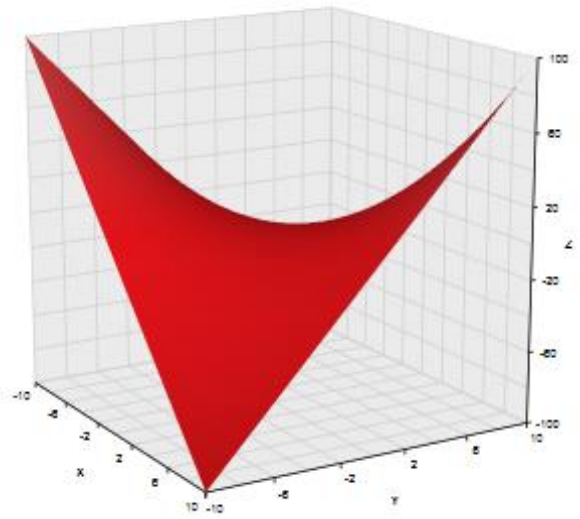

Fig. 3 – Graph with plot of function z = x*y added

2.2 Annotation

Annotation is added to a graph using the Annotation menu. This can take the form of captions, legends, and color keys. Once created, these annotation elements can be selected and dragged to any required position on the graph.

2.3 Layout

Layout of graphs on a page (i.e. no. of graphs per row and column) is specified using the Layout menu. This provides several common layout options plus an option to arbitrarily specify any layout up to 4x4. When a new layout is created with n rows and m columns, n*m graph holders are created on the page. You are then free to initialize each graph as 2D or 3D as required.

2.4 Rearranging Layout

If you create a page with a certain layout, and you then wish to rearrange its graphs, this can be done via the Enter Rearrange Mode menu item on the Layout menu. With rearrange mode activated, you can drag and drop a graph to a destination location, and the graph at the destination will swap places with the dropped graph. Graphs can also be cut/copied and pasted within a page or between pages. While in rearrange mode, all other menu items are disabled, and Exit Rearrange Mode must be pressed to re-enable them.

2.5 Exporting

Using the Export menu, a page can be copied to the clipboard, saved to an image file, or shared via the Share Charm. Any or all of the project pages can be also be sent to the printer.

2.6 Copying/Pasting Graphs

If you have created a graph and you wish to copy it to another location in the project (e.g. to use it as the starting point for a similar graph), selecting it and choosing Graph-Copy Graph will pace it on the program's internal clipboard. To paste, select the new graph location, and choose Graph-Paste Graph.

2.7 Color Schemes

TeraPlot LT has various options for setting the color of component parts of a graph. For example, in a 3D graph, you can set colors for:

- Graph background
- Walls
- Grid
- Axis lines

- Axis tick labels
- Axis titles

and in a 2D graph, you can set:

- Graph background
- Plot area background
- Grid major lines
- Grid minor lines
- Axis tick labels
- Axis titles

In addition, in both 2D and 3D graphs, the colors of annotation items can be set.

If you have configured a graph with a set of colors and you wish to use these colors as the default for all future graphs, the first step is to select the menu item Graph – Save Color Scheme. This displays a popup prompting for the color scheme name. Enter a name in the edit field and press OK. Then, in Settings – Preferences, select the scheme from the appropriate dropdown (color schemes for 2D graphs and 3D graphs are listed in separate dropdowns). Future new graphs will now be created with the specified color scheme.

To remove a color scheme, select it and press the Remove button.

3 2D Graphs

3.1 Overview

By a 2D graph, we mean one which has two axes: x and y. After initializing a 2D graph, the next step is to add one or more line plots to it. This is done using the Line Plots Popup, accessed via the menu item Graph-Add/Remove Plots. The line plots popup lists all plots currently present in the graph, and provides buttons to add new ones and remove existing ones. When an analytical plot is added or selected, a script window is made available for script entry. When a tabular plot is added or selected, a two column grid of cells is provided for data entry.

Line plots in a 2D graph can be defined in 3 coordinate forms. These are:

1. Cartesian, $y = f(x)$. This is the normal x, y coordinate system.
2. Polar, $r = f(\varphi)$. These are plane polar coordinates where r represents the radial distance of a point from the origin, and φ (phi) is the counterclockwise angle between the x axis and a line joining the origin to the point in question.
3. Parametric, $x = f_1(t)$, $y = f_2(t)$. This is the parametric definition of a line where both x and y are specified as functions of a parameter t.

These coordinate forms apply to both analytical and tabular plots. For analytical plots, expressions must contain independent and dependent variable symbols appropriate to the plot form. For tabular plots, the data points must already be in the appropriate coordinates. The plot forms are summarized in the table below, along with example scripts.

Form	Independent Variable	Dependent Variable	Reserved Symbols	Example Script
Cartesian	x	y	x, y	g = sin(x) y = cos(x) + g
Polar	φ	r	r, phi	a = 1.5 r = a*exp(2.1*phi)
Parametric	t	x and y	t, x, y	x = t*t y = 4*t*t

In the example column in the table above, all examples have two lines to illustrate that a script can contain multiple lines, and that any symbol encountered that isn't a reserved symbol is simply treated as a variable you defined for your own use, such as g in the Cartesian example, and a in the polar example. Since scripting is common to both 2D and 3D graphs, it's described under Scripting below. We won't go any further into the script syntax here then, except to point out that although sometimes the Greek letter φ is used in the description of polar coordinates in the table above, the actual English word is used in the script. So the symbol φ should never be used in a script, but rather the word phi.

For tabular 2D Line plots, data can be entered manually, pasted from the clipboard (e.g. after being copied there using Excel), or loaded from a text file. If loading from a text file, the file should contain two columns of numbers only (x column first, then y), with no heading. The numbers may be separated by spaces, commas or tabs.

3.2 Demos – Creating 2D Graphs

For a complete step-by-step guide on how to create 2D graphs (analytical and tabular), see the demo videos at the TeraPlot LT YouTube demo channel: http://www.youtube.com/teraplotlt.

4 3D Graphs

4.1 Overview

By a 3D graph, we mean one which has three axes: x, y and z. After initializing a 3D graph, the next step is to add one or more surface plots to it. This is done using the Surface Plots Popup, accessed via the menu item Graph-Add/Remove Plots. The surface plots popup lists all plots currently present in the graph, and provides buttons to add new ones and remove existing ones. When an analytical plot is added or selected, a script window is made available for script entry. When a tabular plot is added or selected, a multi column grid of cells is provided for data entry.

Surface plots in a 3D graph can be defined in 4 coordinate forms. These are:

1. Cartesian, z = f(x, y). This is the normal x, y, z coordinate system.
2. Spherical Polar, r = f(θ, φ) where r represents the radial distance of a point from the origin, θ (theta) represents elevation and φ (phi) represents azimuth.
3. Cylindrical Polar, r = f(φ, z) where z represents distance along the z axis and φ (phi) represents azimuth.
4. Parametric, x = f_1(u, v), y = f_2(u, v), z = f_3(u, v). This is the parametric definition of a surface where x, y and z are specified as functions of parameters u and v.

These coordinate forms apply to both analytical and tabular plots. For analytical plots, expressions must contain independent and dependent variable symbols appropriate to the plot form. For tabular plots, the data points must already be in the appropriate coordinates. The plot forms are summarized in the table below, along with example scripts.

Form	Independent Variables	Dependent Variable	Reserved Symbols	Example Script
Cartesian	x, y	z	x, y, z	g = sin(x*y) z = x*y + g
Polar	θ, φ	r	r, theta, phi	a = 1.5 r = sin(a*theta*phi)
Cylindrical	φ, z	r	r, phi, z	g = sin(z) r = g*cos(phi)

Parametric	u, v	x, y, and z	x, y, z, u, v	x = u y = v z = u*v

In the example column in the table above, all examples have at least two lines to illustrate that a script can contain multiple lines, and that any symbol encountered that isn't a reserved symbol is simply treated as a variable you defined for your own use, such as g in the Cartesian and cylindrical examples, and a in the polar example. Since scripting is common to both 2D and 3D graphs, it's described under Scripting below. We won't go any further into the script syntax here then, except to point out that although sometimes the Greek letters φ and θ are used in the description of polar coordinates in the table above, the actual English word is used in the script. So the symbols φ or θ should never be used in a script, but rather the words phi and theta.

For tabular surface plots, the x and y values are defined by four numbers: the x minimum, the x spacing, the y minimum, and the y spacing. Data (z values) can be pasted from the clipboard (e.g. after being copied there using Excel), or loaded from a text file. The data should in the form of n*m z values where n is the number of rows, and m is the number of columns. Each row represents a constant x value with x increasing downwards. Each column represents a constant y value with y increasing to the right. If loading from a text file, the file should contain m columns of numbers with no heading. The numbers may be separated by spaces, commas or tabs.

4.2 Demos – Creating 3D Graphs

For a complete step-by-step guide on how to create 3D graphs (analytical and tabular), see the demo videos at the TeraPlot LT YouTube demo channel: http://www.youtube.com/teraplotlt.

5 Scripting

Defining an analytical plot in the Add/Remove Plots Popup involves creating one or more lines of script text which assign a value to the dependent variable(s) for that plot type, usually as some function of the independent variable(s). Because the script can be multiline, a value needs to be explicitly assigned to the dependent variable, usually on the final line. When the Draw Graph button is pressed, TeraPlot LT then evaluates the script for every value of the independent variable(s) and draws the plot. Let's take a specific example for a Cartesian 2D line plot:

```
t1 = 3*cos(_pi*x)
t2 = sin(x)
y = (t1 + t2)/4.25
```

The first two lines of the above example creates two terms, t1 and t2, based on x, the independent variable. In the third line, t1 and t2 are added together, divided by 4.25, and the result assigned to the independent variable y. The above lines encapsulate the two simple issues you need to be aware of to create almost any script. These are: script arithmetic syntax and reserved symbols, described in the following two sections. A further use scripting feature is conditional evaluation of an expression, and this is also described below.

5.1 Arithmetic Syntax.

When writing math expressions on paper, multiplication is often implied, as is the application of a function to a variable. An example might be:

y = 3sinx.

If the above expression was entered as is, the TeraPlot LT script parser would misinterpret this expression and assume 3sinx was all one symbol. Multiplication therefore needs to be made explicit, and sin needs to be identified as a function by placing its arguments in brackets to give:

y = 3*sin(x).

The use of the other arithmetic symbols +, -, /, requires no further special consideration.

5.2 Reserved Symbols.

In the 3 line script above, we used the reserved symbols x, y and _pi, and effectively declared two temporary symbols, t1 and t2, for our own use. _pi is a built-in constant representing the number π. Another built-in constant is _e, representing the number e. As for t1 and t2, if the script parser encounters any unrecognized symbols in an otherwise valid expression, it will assume they have been defined by you, and incorporate them into its list of symbols. The list of symbols specific to each plot type is given in the tables in 2D Graphs and 3D Graphs. One point to note is that although sometimes Greek letters such as θ, and φ are used in the description of coordinates, the actual English word is used in the script. So the symbols θ, φ and π should never be used in a script, but rather theta, phi and _pi.

5.3 Conditional Statements

Sometimes it's useful to be able to use an **if-then-else** type statement in a script. For example, to plot the function

$$y = \begin{cases} \sin(x) & (x >= 0) \\ -\sin(x) & (x < 0) \end{cases}$$

we would want to be able to say something like "if x is greater than or equal to 0, then y = sin(x), otherwise y = -sin(x)".

Statements like this can be created using the if-then-else operator:

? :

This operator takes three arguments (arg1, arg2, arg3), and returns a single value (val) as shown below:

val = arg1 ? arg2 : arg3

arg1 is a boolean. If arg1 evaluates to true, val is set equal to arg2, otherwise val is set equal to arg3.

To evaluate our example function, the expression would therefore be:

y = x >= 0 ? sin(x) : -sin(x)

This could be made more readable by putting the right hand side inside brackets, but it's not a requirement:

y = (x >= 0 ? sin(x) : -sin(x))

5.4 Dealing with Singularities and Undefined Points

As mentioned above, TeraPlot LT evaluates your script at each of the plot's data points. However, certain functions don't have defined values for all points. For example y = 1/x is infinite at x = 0. So what happens in this case if one of the plot x values is 0? In this case, TeraPlot LT would not be able to evaluate the function at x = 0 since it involves division by 0. The point is therefore marked as invalid, and is omitted from the final plot.

Omitting points that can't be evaluated may or may not be desirable visually. For example, suppose we are plotting y = 1/x from -5 to 5 using 101 data points. This gives 100 divisions over the range, and the 51st point will lie at x = 0. The resulting plot is shown in fig.3 below. In this case the omission of the singular point at x = 0 doesn't impact the plot since we want to see the two disjoint arms of the plot.

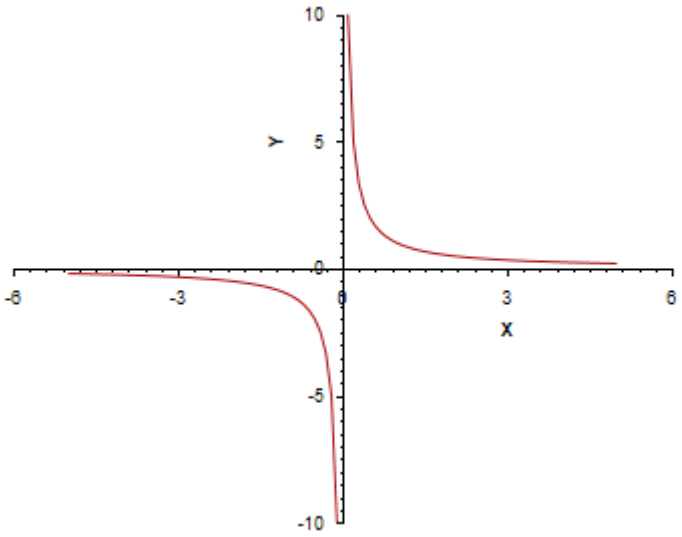

Fig. 3

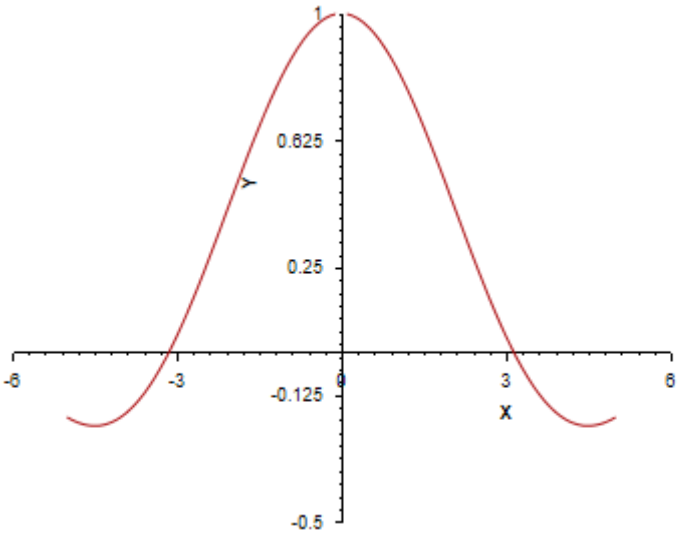

Fig.4

On the other hand, if we are plotting y = sin(x)/x, omitting the point at x = 0 has the effect shown in fig.4 above, i.e. a gap at x = 0. One way to avoid this is to simply make sure that the plot is not calculated at x = 0, e.g. by changing the number of data points from 101 to 100. Another way would be to use a conditional statement in the script (see section 5.3 above) to avoid the point, i.e.:

y = abs(x) > 0.001 ? sin(x)/x : 1.0

5.5 Large Values near Singular Points

Another fact to be aware in relation to plots with singular points is that TeraPlot LT by default sets the dependent variable range in the graph to encompass all data points. For example, in fig. 5 below, the function y = tan(x) has been plotted at 201 points between 0.0 and 10.0. Because some of the x values will fall close to points where the function is singular, the y values for these points will be very large. In the case below, the y value ranges from around -35 to around 250. By default, TeraPlot LT includes all of these points and sets the y axis range accordingly. However a completely satisfactory view of the behaviour of tan(x) would be seen by limiting the y range to say 10 to 10.

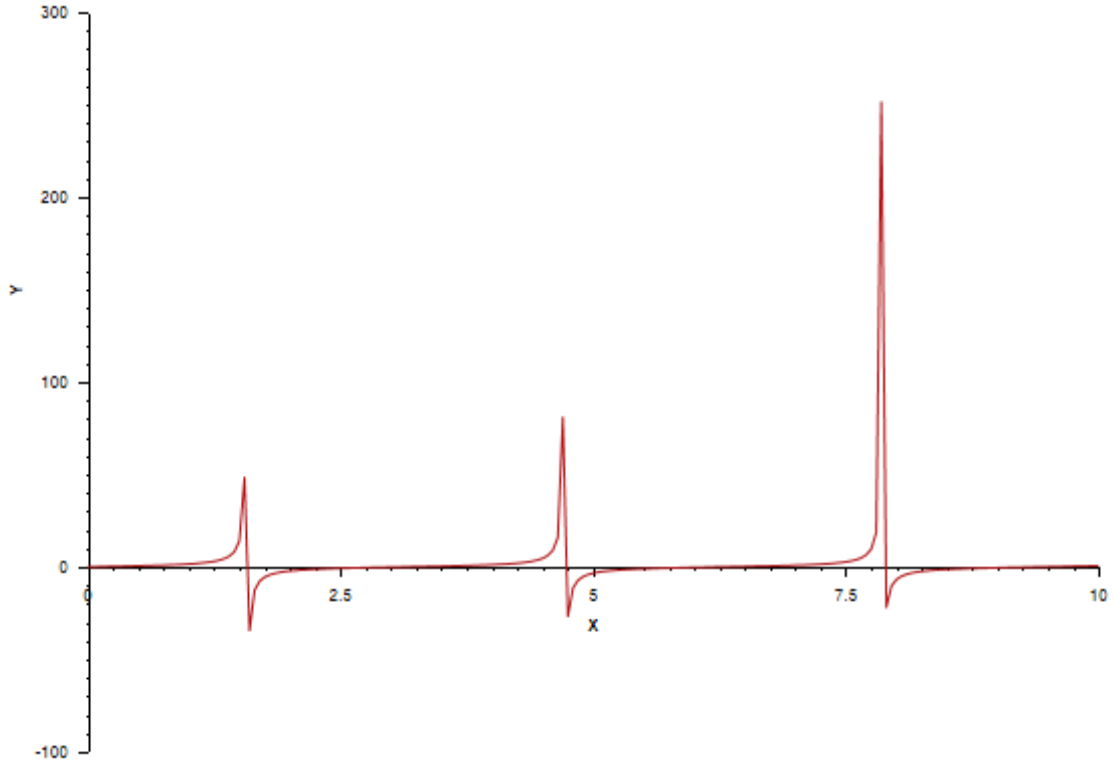

Fig. 5

This has been done in fig. 6 below, producing a graph of tan(x) more like those commonly seen. This was done by simply setting the y axis range to range from -10 to 10 using the Axis Ticks/Ranges popup, accessible from the Axes/Grid menu.

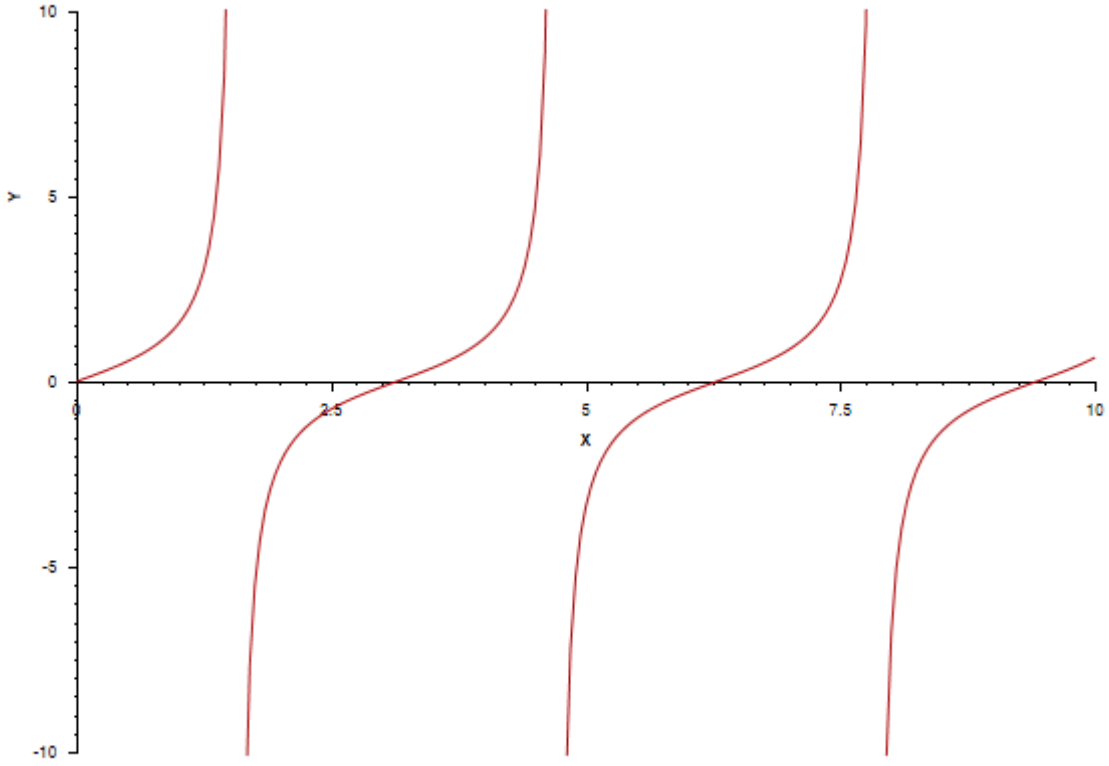

Fig. 6

5.6 List of Built-In Functions

The table below lists the functions available for use in describing analytical plots.

Function	Description	Example
sin	sine	b = sin(a)
cos	cosine	b = cos(a)
tan	tangent	b = tan(a)
asin	inverse sine	b = asin(a)
acos	inverse cosine	b = acos(a)
atan	inverse tangent	b = atan(a)
atan2	inverse tan with two args	c = atan2(a, b)
sinh	hyberbolic sine	b = sinh(a)
cosh	hyberbolic cosine	b = cosh(a)
tanh	hyberbolic tangent	b = tanh(a)
log	log to base 10	b = log(a)
ln	natural log	b = ln(a)
exp	e raised to a power	b = exp(a)
sqrt	square root	b = sqrt(a)
sign	sign of a number	b = sign(a)
abs	absolute value of a number	b = abs(a)