

DataPlotClasses for Xojo

User's Guide v 1.4.5



Roger Meier, January 2025

Table of Contents

DataPlotClasses for Xojo	1
Table of Contents	1
List of Figures	4
List of Tables	4
1. Introduction	5
2. Getting Started	5
3. The Classes	6
3.1. Figure	7
3.1.1. Properties	7
3.1.1.1. BackGoundColor	7
3.1.1.2. EnableMouseEvents	7
3.1.1.3. FrameColor	7
3.1.1.4. GraphCount	7
3.1.1.5. Graphs	7
3.1.1.6. AntiAlias	7
3.1.2. Methods	7
3.1.2.1. DeselectAll	7
3.1.2.2. DrawToGraphics	8
3.1.2.3. Graph	8
3.1.2.4. IndexOfGraph	8
3.1.2.5. Initialize	8
3.1.2.6. Redraw	9
3.1.3. Events	9
3.1.3.1. AutoScale	9
3.1.3.2. BoxClick	9
3.1.3.3. FigureClick	10
3.1.3.4. GraphClick	10
3.1.3.5. MouseDownBox	10
3.1.3.6. MouseDownFigure	10
3.1.3.7. MouseDownGraph	10
3.1.3.8. MouseDownTrace	11
3.1.3.9. MouseOverBox	11
3.1.3.10. MouseOverGraph	11

3.1.3.11.	MouseOverTrace.....	11
3.1.3.12.	TraceClick.....	12
3.2.	Graph.....	13
3.2.1.	Properties	13
3.2.1.1.	AxisLabelSize	13
3.2.1.2.	Box.....	13
3.2.1.3.	BoxColor.....	13
3.2.1.4.	BoxFrame.....	13
3.2.1.5.	BoxFrameColor.....	13
3.2.1.6.	BoxSelected	13
3.2.1.7.	GraphSelected	13
3.2.1.8.	Grid	14
3.2.1.9.	GridColor	14
3.2.1.10.	Legend.....	14
3.2.1.11.	Limits	14
3.2.1.12.	MinorGrid	14
3.2.1.13.	MinorGridColor	14
3.2.1.14.	MinX, MaxX, MinY, MaxY	15
3.2.1.15.	SelectColor.....	15
3.2.1.16.	ShowLegend	15
3.2.1.17.	TextColor	15
3.2.1.18.	TextFont.....	15
3.2.1.19.	TextLabelCount	15
3.2.1.20.	TextLabels.....	15
3.2.1.21.	TickLabelSize	15
3.2.1.22.	Title	16
3.2.1.23.	TitleSize	16
3.2.1.24.	TraceCount.....	16
3.2.1.25.	Traces.....	16
3.2.1.26.	TraceColors.....	16
3.2.1.27.	XLabel, YLabel.....	17
3.2.1.28.	XScaleType, YScaleType	17
3.2.1.29.	XMinorTick, YMinorTick	18
3.2.1.30.	XTick, YTick	18
3.2.1.31.	XTickLabel, YTickLabel.....	18
3.2.1.32.	XTickLabelFormat, YTickLabelFormat.....	19
3.2.2.	Methods.....	19
3.2.2.1.	AddTextLabel	19
3.2.2.2.	AutoFit, AutoFitX, AutoFitY	19
3.2.2.3.	AutoTick, AutoTickX, AutoTickY	19
3.2.2.4.	Initialize	20
3.2.2.5.	IndexOfTrace	20
3.2.2.6.	Plot.....	20
3.2.2.7.	Pixel2ValueX, Pixel2ValueY	20
3.2.2.8.	StemPlot.....	21
3.2.2.9.	SetLimits	21

3.2.2.10.	TextLabel	21
3.2.2.11.	Trace	21
3.2.2.12.	Value2PixelX, Value2PixelY	21
3.3.	Trace	21
3.3.1.	Properties	21
3.3.1.1.	DisplayName.....	21
3.3.1.2.	LineColor	22
3.3.1.3.	LineStyle.....	22
3.3.1.4.	LineWidth	22
3.3.1.5.	MarkerSolid	22
3.3.1.6.	MarkerStyle.....	22
3.3.1.7.	MarkerSize.....	23
3.3.1.8.	Selected	23
3.3.1.9.	Stem	23
3.3.1.10.	x, y.....	23
3.3.1.11.	Visible	23
3.3.1.12.	SkipDecreasingXvalues	23
3.4.	TextLabel	24
3.4.1.	Properties	24
3.4.1.1.	Text	24
3.4.1.2.	TextColor	24
3.4.1.3.	TextFont.....	24
3.4.1.4.	TextSize	24
3.4.1.5.	HorizontalAlignment	24
3.4.1.6.	VerticalAlignment.....	24
3.4.1.7.	Visible	24
3.4.1.8.	x, y.....	25
3.4.1.9.	Rotation.....	25
3.5.	Legend.....	26
3.5.1.	Properties	26
3.5.1.1.	BoxColor.....	26
3.5.1.2.	BoxFrameColor.....	26
3.5.1.3.	Location	26
3.5.1.4.	TextColor	26
3.5.1.5.	TextSize	26
3.5.1.6.	Visible	26
4.	Examples.....	27
4.1.	Example 1	27
4.2.	Example 2	29
4.3.	Example 3	31

List of Figures

Figure 1: Getting Started.....	5
Figure 2: Class Hierarchy.	6
Figure 3: Classes Overview.	6
Figure 4: The Box area of a Graph.	10
Figure 5: Graph Title.	16
Figure 6: Axis Labels.....	17
Figure 7: Axis Scale Types.....	18
Figure 8: Example 1.....	29
Figure 9: Example 2.....	31
Figure 10: Example 3.....	32

List of Tables

Table 1: Style Arguments.	20
Table 2: Line Styles.	22
Table 3: Marker Styles.....	22
Table 4: Legend Locations.....	26

1. Introduction

The DataPlotClasses for Xojo implement an easy way to plot 2D data, similar to the 'plot' and 'stem' functions' in MATLAB.

<http://www.mathworks.com/access/helpdesk/help/techdoc/ref/plot.html>

<http://www.mathworks.com/access/helpdesk/help/techdoc/ref/stem.html>

2. Getting Started

Create a new project and drag the DataPlotClasses folder into your project. Then, drag the "Figure" class, which is a subclass of Canvas, into Window1.

Use the following code to generate some data to be plotted. Note that X and Y data values must be in double arrays, and that the dimensions must match.

```
dim x() as Double
dim y() as Double

for i as integer = 0 to 100
  x.Append i/10
  y.Append 1 + sin(i/10)
next
```

In the simplest case, plotting data only requires three lines of code. Add these lines to your project to plot the data generated by the code above:

```
Figure1.Initialize ' initialize the figure with one graph
call Figure1.Graph(0).Plot(x,y)
Figure1.Redraw
```

This will result in the following plot.

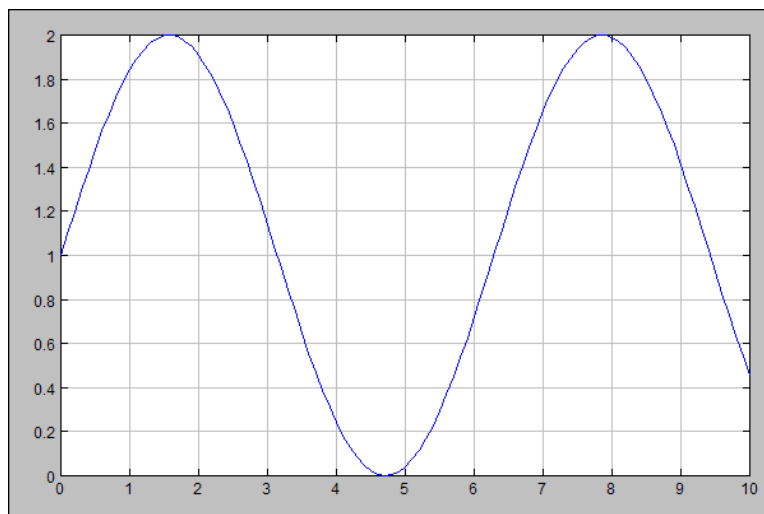


Figure 1: Getting Started

3. The Classes

The DataPlotClasses for Xojo currently consist of 5 classes. The class hierarchy is as follows:

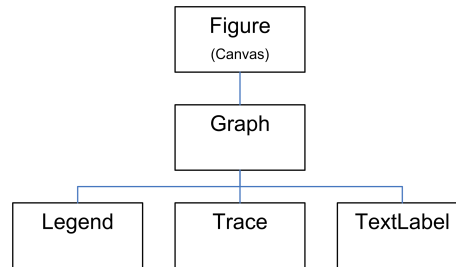


Figure 2: Class Hierarchy.

Figure is a subclass of Canvas, i.e. it can be added to a project simply by dragging it to a window. A **Figure** can have multiple children of type **Graph**. The method **Figure.Initialize** assigns **Graph** classes to **Figure**. The optional arguments determine the number of **Graphs** and their locations.

Graph is a rectangular area with axes, gridlines, tick marks, labels, etc. It can have children of type **Trace** and **Legend**. A **Graph** can have multiple children of type **Trace**, but only one child of type **Legend**.

Trace is a class that contains the actual data to be plotted. The properties of **Trace** determine the style of the plot.

TextLabel is a class that can be used to place text on a **Graph**.

Legend is a rectangular area within a **Graph**, and it displays a list of labels and styles to identify the **Traces** in **Graph**.

Figure 3 depicts a sample plot with three **Graphs**, each with different subclass elements.

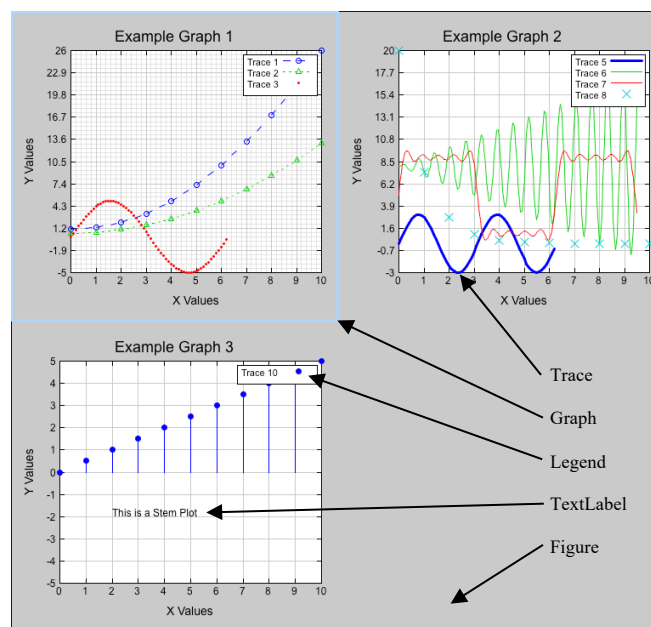


Figure 3: Classes Overview.

3.1. Figure

3.1.1. Properties

Figure inherits all the properties from its super class *Canvas*. In addition to these, it also implements the following properties:

3.1.1.1. BackGroundColor

BackColor as Color = &c00C0C0

Sets or returns the background color of the *Figure*. The default is gray.

3.1.1.2. EnableMouseEvents

EnableMouseEvents as Boolean = true

Determines whether mouse events such as move and click events will fire.

3.1.1.3. FrameColor

FrameColor as Color = &c000000

Sets or returns the frame color of the *Figure*. The default is black.

3.1.1.4. GraphCount

GraphCount as Integer // Read-Only

Returns the number of *Graphs* in the *Figure*.

3.1.1.5. Graphs

Graphs() as Graph // Read-Only

Returns the collection of *Graphs* in the *Figure* as array.

3.1.1.6. AntiAlias

AntiAlias as Boolean = true

Determine whether antialiasing is used to render the *Figure*.

3.1.2. Methods

Figure inherits all the methods from its super class *Canvas*. In addition to these, it also implements the following methods.

3.1.2.1. DeselectAll

DeselectAll

Deselects all selected items in the *Figure* (*Graphs*, *Traces*).

3.1.2.2. DrawToGraphics

DrawToGraphics(g as Graphics, PreserveAspectRatio as Boolean = False)

This can be used to print *Figures* or save them as picture. When printing or saving to pictures, the optional argument *PreserveAspectRatio* can be set to *True*. This can be helpful when printing and the aspect ratio of the displayed figure is not the same as the aspect ratio of the paper onto which the figure is to be printed.

E.g. to save a Figure as picture, use the following code:

```
Function FileSaveAs() As Boolean

    // Create Picture to be saved
    dim p as new Picture(Figure1.Width, Figure1.Height,32)
    dim g as Graphics = p.Graphics

    // Draw Figure
    Figure1.DrawToGraphics(g)

    // Save Picture
    Dim f as FolderItem = GetFolderItem("Graph.png")
    if f <> nil then p.Save(f, picture.SaveAsPNG)

End Function
```

3.1.2.3. Graph

Graph(NoOfGraph as integer) as Graph

Returns the Graph specified by index **NoOfGraph**.

3.1.2.4. IndexOfGraph

IndexOfGraph(gr as Graph) as integer

Returns the index of the *Graph* (**gr**) in the collection of *Graphs* in the *Figure*.

3.1.2.5. Initialize

Initialize(NoOfGraphs as integer = 0)

Deletes all the items from the *Figure* and adds **NoOfGraphs** (zero-based) to the *Figure*. If **NoOfGraphs** is omitted or equal to 0, one *Graph* is added to the *Figure*. For values greater than 0, the *Graphs* are automatically positioned in the *Figure*, in a way such that the number of rows and columns are equal, or only differ by no more than 1, where the number of columns is the larger number.

Initialize(Rows as integer, Columns as integer)

By passing two integer values (both zero-based), the number of Graphs vertically and horizontally can be defined.

Examples:

```
Figure1.Initialize(5) // creates a figure with 3 columns and 2 rows, automatically
Figure1.Initialize(2,1) // creates a figure with 2 columns and 3 rows
```

3.1.2.6. Redraw**Redraw**

Forces the *Figure* contents to be redrawn. This is a thread-safe call and is the only method that should be used to redraw the figure. Use *DrawToGraphics* to print *Figures* or save them as picture.

3.1.3. Events

Figure inherits nearly all the events from its super class *Canvas*. In addition to these, it also implements the following events.

3.1.3.1. AutoScale**AutoScale**(gr as graph, Axis as integer)

Fires whenever a Graph has been autoscaled. The *Graph* that was scaled as well as the affected *Axis* are passed as arguments. The values for *Axis* are either 0 (for x-axis) or 1 (for y-axis).

3.1.3.2. BoxClick**BoxClick**(gr as graph, x as integer, y as integer, xVal as double, yVal as double) As boolean

Fires when the Box area (the rectangular area containing the traces) of a *Graph* is clicked (mouse button pressed and released). Passes the current *Graph* (*gr*), the pixel coordinates (*x,y*), as well as the data coordinates (*xVal,yVal*) of the location where the click occurred. Returning *True* prevents the *GraphClick* event from firing afterwards.

Figure 4 shows the Box area of a *Graph*, marked red.

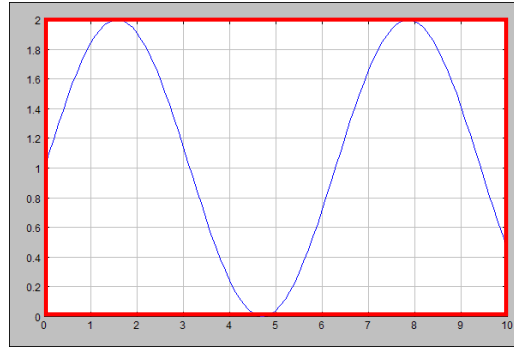


Figure 4: The Box area of a Graph.

3.1.3.3. FigureClick

FigureClick(x as integer, y as integer)

Fires when the **Figure** is clicked (mouse button pressed and released). Passes the pixel coordinates (x,y) of the location where the click occurred.

3.1.3.4. GraphClick

GraphClick(gr as graph, x as integer, y as integer) As Boolean

Fires when a **Graph** is clicked (mouse button pressed and released), outside the Box (Labels, Title, etc) as well as inside the Box (rectangular area with traces). Returning **True** prevents the **FigureClick** event from firing afterwards.

3.1.3.5. MouseDownBox

MouseDownBox(gr as graph, x as integer, y as integer, xVal as double, yVal as double) As boolean

Fires when the mouse button is pressed down inside the area of the box (the rectangular area containing the traces) of a **Graph**. Passes the current **Graph** (gr), the pixel coordinates (x,y), as well as the data coordinates (xVal,yVal) of the location where the mouse was pressed. Returning **True** prevents the **MouseDownGraph** event from firing afterwards.

3.1.3.6. MouseDownFigure

MouseDownFigure(x as integer, y as integer) As boolean

Fires when the mouse button is pressed down inside the **Figure**. Passes the pixel coordinates (x,y) of the location where the mouse button was pressed.

3.1.3.7. MouseDownGraph

MouseDownGraph(gr as graph, x as integer, y as integer, xVal as double, yVal as double) As boolean

Fires when the mouse button is pressed down inside a **Graph**. Passes the current **Graph** (gr), the pixel coordinates (x,y), as well as the data coordinates

(*xVal,yVal*) of the location where the mouse was pressed. Returning *True* prevents the *MouseDownFigure* event from firing afterwards.

3.1.3.8. MouseDownTrace

MouseDownTrace(gr as graph, t as trace, x as integer, y as integer, xVal as double, yVal as double, DPoint as integer) As boolean

Fires when the mouse button is pressed down inside a *Graph*. Passes the current *Graph* (*gr*), the current *Trace* (*t*), the current pixel coordinates (*x,y*), the current data coordinates (*xVal,yVal*) of the location where the mouse was pressed. If the mouse was pressed on an actual data point, *DPoint* contains the index of the data point within the *t.x()* and *t.y()* value arrays of the *Trace* (*t*), otherwise *DPoint* assumes the value -1. Returning *True* prevents the *MouseDownBox* event from firing afterwards.

3.1.3.9. MouseOverBox

MouseOverBox(gr as graph, x as integer, y as integer, xVal as double, yVal as double) As Boolean

Fires when the mouse is moving over the Box area of a *Graph*. Passes the current *Graph* (*gr*), the current pixel coordinates (*x,y*), as well as the current data coordinates (*xVal,yVal*) of the currently displayed data range. Returning *True* prevents the *MouseOverGraph* event from firing.

3.1.3.10. MouseOverGraph

MouseOverGraph(gr as graph, x as integer, y as integer) As Boolean

Fires when the mouse is moving over a *Graph*. Passes the *Graph* (*gr*) and the current pixel coordinates (*x,y*). Returning *True* prevents the *MouseMove* event of from firing afterwards.

3.1.3.11. MouseOverTrace

MouseOverTrace(gr as graph, t as trace, x as integer, y as integer, xVal as double, yVal as double, DPoint as integer) As Boolean

Fires when the mouse is moving over a *Trace* (line or marker). Passes the current *Graph* (*gr*), the current *Trace* (*t*), the current pixel coordinates (*x,y*), as well as the current data coordinates (*xVal,yVal*). If the mouse is over an actual data point, *DPoint* contains the index of the data point within the *t.x()* and *t.y()* value arrays of the *Trace* (*t*), otherwise *DPoint* assumes the value -1. Returning *True* prevents the *MouseOverBox* event from firing.

3.1.3.12. TraceClick

TraceClick (gr as graph, t as trace, x as integer, y as integer, xVal as double, yVal as double, DPoint as integer) As Boolean

Fires when a *Trace* is clicked (mouse button pressed and released). Passes the current **Graph** (**gr**), the current **Trace** (**t**), the current pixel coordinates (**x**,**y**), as well as the current data coordinates (**xVal**,**yVal**). If the click occurred on an actual data point, **DPoint** contains the index of the data point within the **t.x()** and **t.y()** value arrays of the **Trace** (**t**), otherwise **DPoint** assumes the value -1. Returning **True** prevents the **BoxClick** event from firing.

3.2. Graph

3.2.1. Properties

3.2.1.1. AxisLabelSize

AxisLabelSize As Integer = 14

Sets or returns the font size for Axis Labels.

3.2.1.2. Box

Box As Boolean = true

Turns the box on or off. If this property is *true*, the box is drawn as a filled rectangle in the color defined in *BoxColor*.

3.2.1.3. BoxColor

BoxColor As Color = &cFFFFFF

Sets or returns the color of the Box. Default is white.

3.2.1.4. BoxFrame

BoxFrame As Boolean = true

Turns the Box Frame on or off. If this property is *true*, the box frame is drawn as a rectangle in the color defined in *BoxFrameColor*.

3.2.1.5. BoxFrameColor

BoxFrameColor As color = &c000000
--

Sets or returns the color of the Box Frame. Default is black.

3.2.1.6. BoxSelected

BoxSelected As Boolean = false

If this property is *true*, the box is selected. This can also be used to select or deselect the box. If the box is selected, a selection frame will be drawn around the box.

3.2.1.7. GraphSelected

GraphSelected As Boolean = false

If this property is *true*, the *Graph* is selected. This can also be used to select or deselect the *Graph*. If the *Graph* is selected, a selection frame will be drawn around the *Graph*.

3.2.1.8. Grid

Grid As Boolean = true

Determines whether a grid is drawn or not. If this property is *true*, grid lines will be drawn in the color defined in *GridColor* and at the X and Y values specified in *XTick()* and *YTick()*.

3.2.1.9. GridColor

GridColor As color = &cC0C0C0

Sets or returns the color of the grid lines. The default is gray.

3.2.1.10. Legend

Legend as Legend // Read-Only

Returns the *Legend* of the *Graph* so that its properties can be accessed.

3.2.1.11. Limits

Limits (3) as double

A double array with 4 elements holding the X and Y axis limits:

Limits(0) = MinX

Limits(1) = MaxX

Limits(2) = MinY

Limits(3) = MaxY

Whenever axis limits are assigned using this property, *AutoTick* is executed automatically.

3.2.1.12. MinorGrid

MinorGrid As Boolean = true

Determines whether a minor grid is drawn or not. If this property is *true*, minor grid lines will be drawn in the color defined in *MinorGridColor* and at the X and Y values specified in *XMinorTick()* and *YMinorTick()*.

3.2.1.13. MinorGridColor

MinorGridColor As color = &cDFDFDF

Sets or returns the color of the grid lines. The default is light gray.

3.2.1.14. MinX, MaxX, MinY, MaxY

MinX as double
MaxX as double
MinY as double
MaxY as double

Properties to individually access the axis limits.

3.2.1.15. SelectColor

SelectColor As Color = &c9FCFF

Sets or returns the color used to display selected items in the *Graph*. Default is light blue.

3.2.1.16. ShowLegend

ShowLegend as Boolean = false

Determines whether the *Legend* is drawn or not. If this property is *true*, the *Legend* will be drawn at the location specified by *Legend.Location*.

3.2.1.17. TextColor

TextColor As Color = &c000000

Sets or returns the color used to display text in the *Graph* (Title, Axes, Labels, etc). Default is black.

3.2.1.18. TextFont

TextFont As String = "Arial"

Sets or returns the font used to display text in the *Graph* (Title, Axes, Labels, Legend, etc). Default is "Arial".

3.2.1.19. TextLabelCount

TextLabelCount as integer // Read-Only

Returns the number of children of type *TextLabel*.

3.2.1.20. TextLabels

TextLabels() as Trace // Read-Only

Returns the collection of *TextLabels* in the *Graph*.

3.2.1.21. TickLabelSize

TickLabelSize As Integer = 12

Sets or returns the font size used for tick mark labels.

3.2.1.22. Title

Title As String = ""

Sets or returns the title of the *Graph*. If this *String* is left empty, no title will be drawn and the area otherwise occupied by the title is used to expand the *Graph*, as shown in Figure 5.

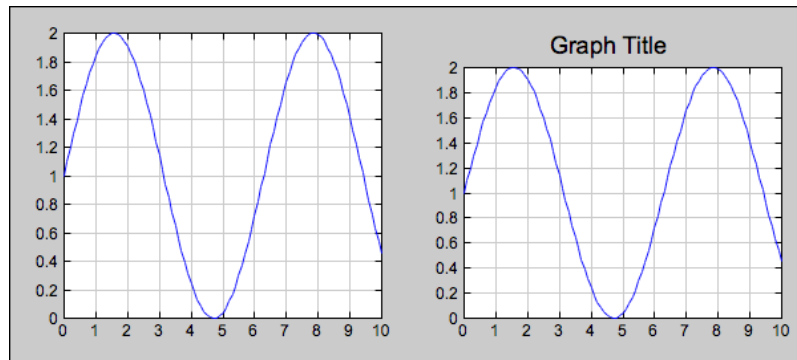


Figure 5: Graph Title.

3.2.1.23. TitleSize

TitleSize As Integer = 18

Sets or returns the font size used to draw the *Graph* title.

3.2.1.24. TraceCount

TraceCount as integer // Read-Only

Returns the number of children of type *Trace*.

3.2.1.25. Traces

Traces() as Trace // Read-Only

Returns the collection of *Traces* in the *Graph*.

3.2.1.26. TraceColors

TraceColors() as Color // Read-Only

Returns the array of default trace colors used in the *Graph*.

3.2.1.27. XLabel, YLabel

XLabel As String = ""
YLabel As String = ""

Sets or returns the labels of the axes of the *Graph*. If these *Strings* are left empty, no labels will be drawn, and the area otherwise occupied by the labels is used to expand the *Graph*, as shown in Figure 6.

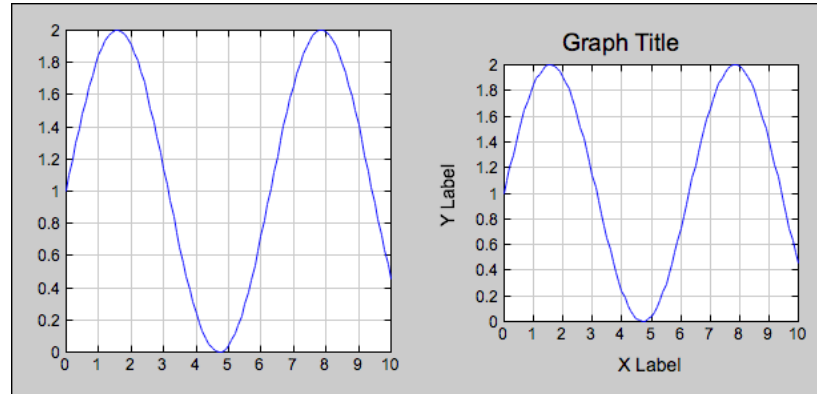


Figure 6: Axis Labels.

3.2.1.28. XScaleType, YScaleType

XScaleType as integer = 0
YScaleType as integer = 0

Determines the scale type of each axis:

0: linear

1: logarithmic

The axis scale type can be set individually for each axis. The different combinations are shown in Figure 7.

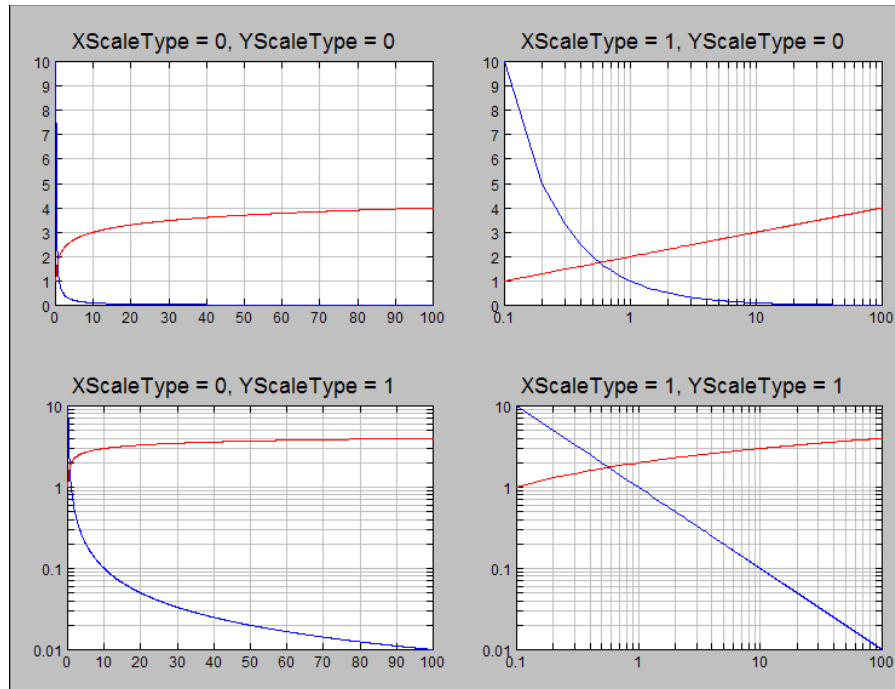


Figure 7: Axis Scale Types.

Whenever the scale type of an axis is set, *AutoFit* is automatically executed for this axis.

3.2.1.29. XMinorTick, YMinorTick

XMinorTick() as double
YMinorTick() as double

Arrays of type *Double*, one for each axis. The array elements determine where tick marks and grid lines are drawn. The *AutoTick* functions can be used to automatically populate these arrays.

3.2.1.30. XTick, YTick

XTick() as double
YTick() as double

Arrays of type *Double*, one for each axis. The array elements determine where tick marks and grid lines are drawn. The *AutoTick* functions can be used to automatically populate these arrays. Whenever these arrays are written (manually or automatically), the corresponding tick label arrays (*XTickLabel()* and *YTickLabel()*) are automatically generated to match the values in *XTick()* and *YTick()*.

3.2.1.31. XTickLabel, YTickLabel

XTickLabel() As String
YTickLabel() As String

Arrays of type *String*, one for each axis. The array elements are *strings* that are drawn next to the corresponding tick marks defined in *XTick()* and *YTick()*. Whenever values are assigned to *XTick()* or *YTick()*, the corresponding tick label arrays are automatically generated.

3.2.1.32. XTickLabelFormat, YTickLabelFormat

XTickLabelFormat As String = ""
YTickLabelFormat As String = ""

If not set to an empty *string*, defines the number format used for the axis tick labels. Refer to Xojo's user reference for the **Str** or **Format** commands for available format specifications.

3.2.2. Methods

3.2.2.1. AddTextLabel

AddTextLabel(Text as String, x as double, y as double) as TextLabel

Creates a new *TextLabel* at the passed coordinates (x,y) with the passed **Text**.

3.2.2.2. AutoFit, AutoFitX, AutoFitY

AutoFit
AutoFitX
AutoFitY

The **AutoFit** methods are used to automatically select the axis limits such that all the data of all the *Traces* assigned to the *Graph* is plotted. Automatic fitting can be done for each axis individually (*AutoFixX* or *AutoFixY*), or both axes simultaneously (*AutoFit*).

For *linear* axis scales, the limits are rounded to the next 2nd most significant digit. For *logarithmic* axis scales, the limits are rounded to the next decade.

Whenever an *AutoFit* Method is executed, the corresponding *AutoTick* method is executed as well.

3.2.2.3. AutoTick, AutoTickX, AutoTickY

AutoTick
AutoTickX
AutoTickY

The *AutoTick* methods are used to automatically populate the *XTick()* and *YTick()* arrays. For *linear* axis scales, the axes are divided into 10 equal segments. For *logarithmic* axis scales, each decade along the axis is divided into 10 equal segments.

Calling an *AutoTick* method causes the corresponding tick label array to be populated automatically.

3.2.2.4. Initialize

Initialize()

Removes the *Legend* and *Traces* from the *Graph*. Returns all the colors and display settings to their defaults.

3.2.2.5. IndexOfTrace

IndexOfTrace(t as trace) as integer

Returns the index of the passed *Trace* in the *Graph's* collection of *Traces*.

3.2.2.6. Plot

Plot(x() as double, y() as double, Style as string = "") as trace Plot(y() as double, Style as string = "") as trace

Plots the values in *y()* versus the values in *x()*. If *x()* is omitted, incrementing integer values starting at 0 are used instead. Similar to its MATLAB counterpart, an optional *Style* argument can be passed. *Style* is a space-separated string of style arguments for both the trace and the markers. The options are as follows:

Line Style	Marker Style	Line Color	Line Width
- solid	. point	b blue	1..9 pixels
: dotted	o circle	g green	
-. slash-dotted	x x	r red	
-- dashed	+ plus	c cyan	
n none	* star	m magenta	
	s square	y yellow	
	d diamond		
	t triangle		

Table 1: Style Arguments.

Example:

```
t = Figure1.Graph(0).Plot(x, y, "- o r 2")
```

This plots a red solid line with circle markers, with a width of 2.

Each time *Graph.Plot* is called, a new *Trace* is added to *Graph*. *Graph.Plot* returns the *Trace (t)* that was added to the *Graph*.

3.2.2.7. Pixel2ValueX, Pixel2ValueY

Pixel2ValueX(x as integer) as double Pixel2ValueY(y as integer) as double

Converts pixel coordinates to data coordinates, according to the axis limits and the axis type.

3.2.2.8. StemPlot

StemPlot(x() as double, y() as double) as trace
StemPlot(y() as double) as trace

Draws a Stem Plot for the values in *y()* versus the values in *x()*. If *x()* is omitted, incrementing integer values starting at 0 are used instead. Each time *Graph.StemPlot* is called, a new *Trace* is added to *Graph*. *Graph.StemPlot* returns the *Trace (t)* that was added to the *Graph*.

3.2.2.9. SetLimits

SetLimits(minX as double, maxX as double, minY as double, maxY as double)

Sets the axis limits. Calling this method also executes *AutoTick*.

3.2.2.10. TextLabel

TextLabel(index as integer) as TextLabel

Returns the *TextLabel* from the *Graph's* collection of *TextLabels*, specified by *index*.

3.2.2.11. Trace

Trace(index as integer) as Trace

Returns the *Trace* from the *Graph's* collection of *Traces*, specified by *index*.

3.2.2.12. Value2PixelX, Value2PixelY

Value2PixelX(xVal as double) as integer
Value2PixelY(yVal as double) as integer

Converts data coordinates to pixels coordinates, according to the axis limits and the axis type.

3.3. Trace

3.3.1. Properties

3.3.1.1. DisplayName

DisplayName As String = ""

This is the name of the *Trace* that is displayed in the *Legend*.

3.3.1.2. LineColor

LineColor As Color

Sets or returns the color of the *Trace*.

3.3.1.3. LineStyle

LineStyle As Integer = 0

Sets or returns the line style of the *Trace*.

Line Style	
0	solid
1	dotted
2	slash-dotted
3	dashed
4	none

Table 2: Line Styles.

3.3.1.4. LineWidth

LineWidth As Integer = 1

Sets or returns the thickness of the *Trace* line in pixels.

3.3.1.5. MarkerSolid

MarkerSolid As Boolean = false

This property determines whether markers that are polygons (circle, square, diamond, triangle) are drawn solid or outlined only.

3.3.1.6. MarkerStyle

MarkerStyle As Integer = 8

Sets or returns the style of the Marker.

Marker Style	
0	point
1	circle
2	x
3	plus
4	star
5	square
6	diamond
7	triangle
8	none

Table 3: Marker Styles.

3.3.1.7. MarkerSize**MarkerSize** As Integer = 7

Sets or returns the size of the Marker in pixels.

3.3.1.8. Selected**Selected** As Boolean = false

If this property is *true*, the *Trace* is selected. This can also be used to select or deselect the *Trace*. If the *Trace* is selected, it will be drawn wider and in a different color.

3.3.1.9. Stem**Stem** As Boolean = false

If this property is *true*, the *Trace* is drawn with stems at each datapoint. When adding a *Trace* to a *Graph* using the *Graph.StemPlot* method, this property is set to *true* for the added *Trace*.

3.3.1.10. x, y**x()** As double
y() As double

These arrays contain the actual data of the trace. The values are usually automatically assigned to these arrays via the *Graph.Plot* method.

3.3.1.11. Visible**Visible** As Boolean = true

If this property is *true*, the *Trace* is visible. If this property is *false*, the *Trace* will be hidden.

3.3.1.12. SkipDecreasingXvalues**SkipDecreasingXvalues** As Boolean = false

If this property is *true*, lines between datapoints with decreasing x-values are skipped.

3.4. TextLabel

3.4.1. Properties

3.4.1.1. Text

Text As String = ""

This is the **Text** of the *TextLabel* that will be drawn in the *Graph*.

3.4.1.2. TextColor

TextColor As Color = &c000000

Sets or returns the color used to display the *TextLabel*. Default is black.

3.4.1.3. TextFont

TextFont As String = "Arial"

Sets or returns the font used to display text in the *Graph* (Title, Axes, Labels, Legend, etc). Default is "Arial".

3.4.1.4. TextSize

TextSize as integer = 12

Sets or returns the font size used for the *TextLabel*.

3.4.1.5. HorizontalAlignment

HorizontalAlignment as integer = <i>TextLabel.kHorizontalAlignmentLeft</i>

Sets or returns the horizontal text alignment used to display the *TextLabel*.

Valid values are: *TextLabel.kHorizontalAlignmentLeft*
TextLabel.kHorizontalAlignmentCenter
TextLabel.kHorizontalAlignmentRight

3.4.1.6. VerticalAlignment

VerticalAlign as integer = <i>TextLabel.kVerticalAlignmentBottom</i>

Sets or returns the vertical text alignment used to display the *TextLabel*. Valid values are:

TextLabel.kVerticalAlignmentBottom
TextLabel.kVerticalAlignmentMiddle
TextLabel.kVerticalAlignmentTop

3.4.1.7. Visible

Visible as integer = false

Determines whether the *TextLabel* is drawn or not. The default is *false*.

3.4.1.8. x, y

```
x as double = 0  
y as double = 0
```

Sets or returns the coordinates of the *TextLabel*.

3.4.1.9. Rotation

```
Rotation as double = 0
```

Sets or returns the rotation (in degrees) of the *TextLabel*.

3.5. Legend

3.5.1. Properties

3.5.1.1. BoxColor

BoxColor As Color = &cFFFFFF

Sets or returns the color of the Box of the *Legend*. Default is white.

3.5.1.2. BoxFrameColor

BoxFrameColor As Color = &c000000
--

Sets or returns the color of the Box Frame of the *Legend*. Default is black.

3.5.1.3. Location

Location as integer = 0

Sets or returns the location of the *Legend* in the *Graph*.

Location	
0	top right
1	top left
2	bottom right
3	bottom left
4	top
5	bottom
6	right
7	left

Table 4: Legend Locations.

3.5.1.4. TextColor

TextColor As Color = &c000000

Sets or returns the color used to display text in the *Legend*. Default is black.

3.5.1.5. TextSize

TextSize as integer = 11

Sets or returns the font size used for the *Legend*.

3.5.1.6. Visible

Visible as integer = false

Determines whether the *Legend* is drawn or not. The default is *false*.

4. Examples

The following examples use the following routine to create linear arrays of type double:

```
Function MakeRange(StartVal as double, StopVal as double, Interval as double) As double()
    // Creates an array of doubles, starting at StartVal and ending at StopVal,
    // with increments of Interval

    dim r() as double

    if Interval = 0 then
        r.Append StartVal
        r.Append StopVal
        return r
    end if

    dim NoOfSteps as integer = Floor((StopVal-StartVal) / Interval)

    for i as integer = 0 to NoOfSteps
        r.Append StartVal + i*interval
    next

    return r

End Function
```

4.1. Example 1

Code:

```
// Generate Some Data
dim x1() as double
dim x2() as double
dim x3() as double
dim y1() as double
dim y2() as double
dim y3() as double
dim y4() as double
dim y5() as double
dim y6() as double
dim y7() as double
dim y8() as double
dim y9() as Double
dim y10() as Double

x1 = MakeRange(0,10,1)
for i as integer = 0 to UBound(x1)
    y1.Append 1 + (x1(i)*0.5)^2
    y2.append y1(i) / 2
next

x2 = MakeRange(0,6.3,0.1)
for i as integer = 0 to UBound(x2)
    y3.Append 5*sin(x2(i))
    y4.Append 3*sin(x2(i)*2)
next
```

```
x3 = MakeRange(0,9.5,0.05)
for i as integer = 0 to UBound(x3)
  y5.Append sin(x3(i)*10) * x3(i) + 8
  y6.Append 5 * (sin(x3(i)) + 1/3*sin(3*x3(i)) + 1/5*sin(5*x3(i)) _
    + 1/7*sin(7*x3(i)) + 1/9*sin(9*x3(i)) ) + 5
next

for i as integer = 0 to UBound(x1)
  y7.Append 20 * exp(-x1(i))
  y8.Append Sqrt(40*x1(i))
  y9.Append 20 - 2*i
  y10.Append 20
next

// Plot data
Figure1.Initialize(0)

dim t as trace
dim g as graph = Figure1.Graph(0)
t = g.Plot(x1,y1," 0 o --")
t.DisplayName = "Trace 1"
t = g.Plot(x1,y2,"t :")
t.DisplayName = "Trace 2"
t = g.Plot(x2,y3,"n .")
t.DisplayName = "Trace 3"
t = g.Plot(x2,y4,"3")
t.DisplayName = "Trace 4"
t = g.Plot(x3,y5)
t.DisplayName = "Trace 5"
t = g.Plot(x3,y6)
t.DisplayName = "Trace 6"
t = g.Plot(x1,y7,"n x")
t.DisplayName = "Trace 7"
t.MarkerSize = 11 ' custom marker size
t = g.Plot(x1,y8,"- s")
t.DisplayName = "Trace 8"
t.LineColor = &c007700 ' custom color
t = g.Plot(x1,y9,"n +")
t.DisplayName = "Trace 9"
t = g.Plot(x1,y10,"*")
t.DisplayName = "Trace 10"

// Labels and Title
g.XLabel = "X Values"
g.YLabel = "Y Values"
g.Title = "Example Graph"

// Show Legend
g.Legend.Location = 1
g.ShowLegend = true

Figure1.Redraw
```

Result:

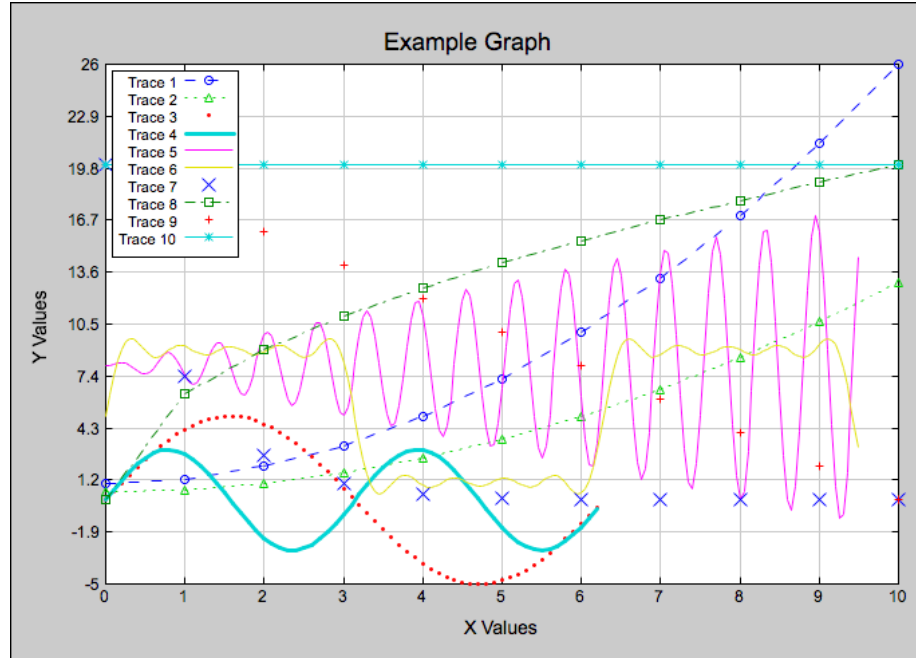


Figure 8: Example 1.

4.2. Example 2

Code:

```
// Generate Some Data
dim x1() as double
dim x2() as double
dim x3() as double
dim y1() as double
dim y2() as double
dim y3() as double
dim y4() as double
dim y5() as double
dim y6() as double
dim y7() as double
dim y8() as double
dim y9() as Double
dim y10() as Double

x1 = MakeRange(0,10,1)
for i as integer = 0 to UBound(x1)
    y1.Append 1 + (x1(i)*0.5)^2
    y2.append y1(i) / 2
next

x2 = MakeRange(0,6.3,0.1)
for i as integer = 0 to UBound(x2)
    y3.Append 5*sin(x2(i))
    y4.Append 3*sin(x2(i)*2)
next
```

```

x3 = MakeRange(0,9.5,0.05)
for i as integer = 0 to UBound(x3)
  y5.Append sin(x3(i)*10) * x3(i) + 8
  y6.Append 5 * (sin(x3(i)) + 1/3*sin(3*x3(i)) + 1/5*sin(5*x3(i)) _
    + 1/7*sin(7*x3(i)) + 1/9*sin(9*x3(i)) ) + 5
next

for i as integer = 0 to UBound(x1)
  y7.Append 20 * exp(-x1(i))
  y8.Append Sqrt(40*x1(i))
  y9.Append 20 - 2*i
  y10.Append 20
next

// Plot data
Figure1.Initialize(2) // We want 3 Graphs

dim t as trace
dim g as graph = Figure1.Graph(0)
t = g.Plot(x1,y1," o --")
t.DisplayName = "Trace 1"
t = g.Plot(x1,y2,"t :")
t.DisplayName = "Trace 2"
t = g.Plot(x2,y3,"n .")
t.DisplayName = "Trace 3"
t = g.Plot(x2,y4,"3")
g = Figure1.Graph(1)
t.DisplayName = "Trace 4"
t = g.Plot(x3,y5)
t.DisplayName = "Trace 5"
t = g.Plot(x3,y6)
g = Figure1.Graph(2)
t.DisplayName = "Trace 6"
t = g.Plot(x1,y7,"n x")
t.DisplayName = "Trace 7"
t.MarkerSize = 11 ' custom marker size
t = g.Plot(x1,y8,"- s")
t.DisplayName = "Trace 8"
t.LineColor = &c007700 ' custom color
t = g.Plot(x1,y9,"n +")
t.DisplayName = "Trace 9"
t = g.Plot(x1,y10,"*")
t.DisplayName = "Trace 10"

// Labels, Title and Legend
for i as integer = 0 to 2
  g = Figure1.Graph(i)
  g.XLabel = "X Values"
  g.YLabel = "Y Values"
  g.Title = "Example Graph"
  g.Legend.Location = 1
  g.ShowLegend = true
next

Figure1.Redraw

```

Result:

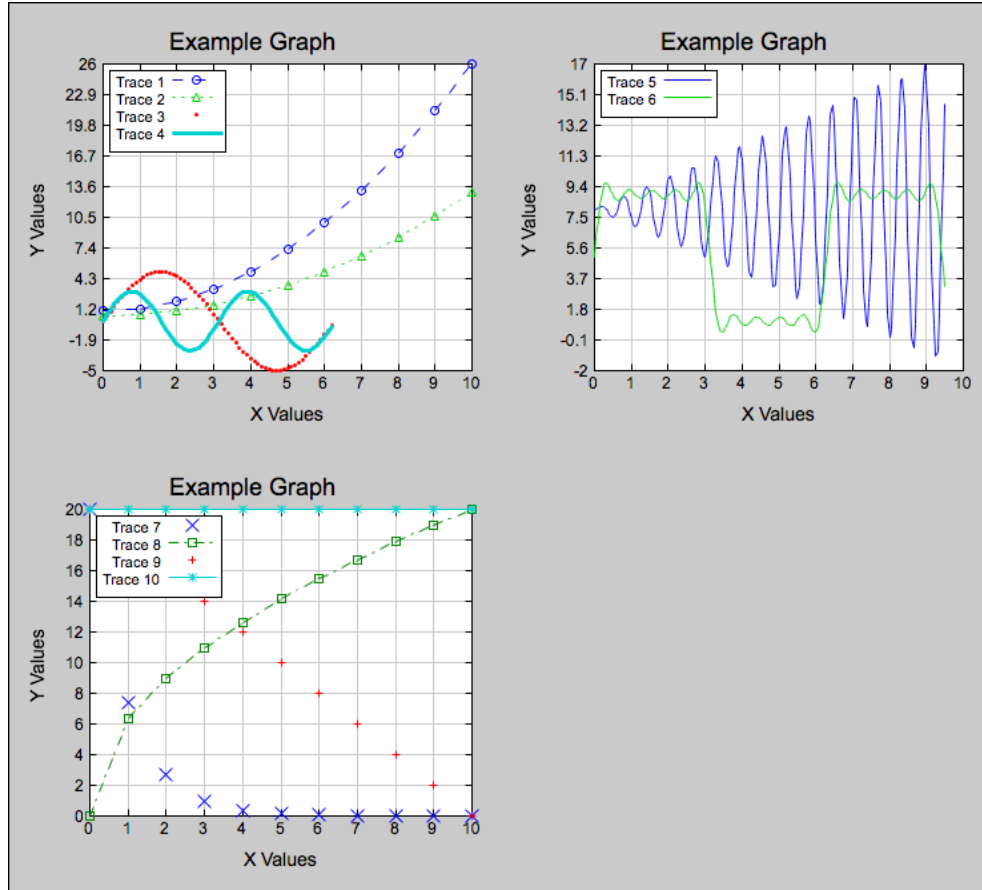


Figure 9: Example 2.

4.3. Example 3

Code:

```
// Generate Some Data
dim x1() as double
dim y1() as double
dim y2() as Double

x1 = MakeRange(0.1,999.9,0.1)
for i as integer = 0 to UBound(x1)
    y1.Append 1/x1(i)
    y2.Append 2 + log(x1(i))/log(10)
next

// Plot data
Figure1.Initialize(0)

dim t as trace
dim g as Graph = Figure1.Graph(0)
g.XScaleType = 1
g.YScaleType = 1
```

```
t = g.Plot(x1,y1,"o --")
t.DisplayName = "Trace 1"
t = g.Plot(x1,y2,"r")
t.DisplayName = "Trace 2"

// Labels and Title
g.XLabel = "X Values"
g.YLabel = "Y Values"
g.Title = "Example Graph"

// Show Legend
g.ShowLegend = true

Figure1.Redraw
```

Result:

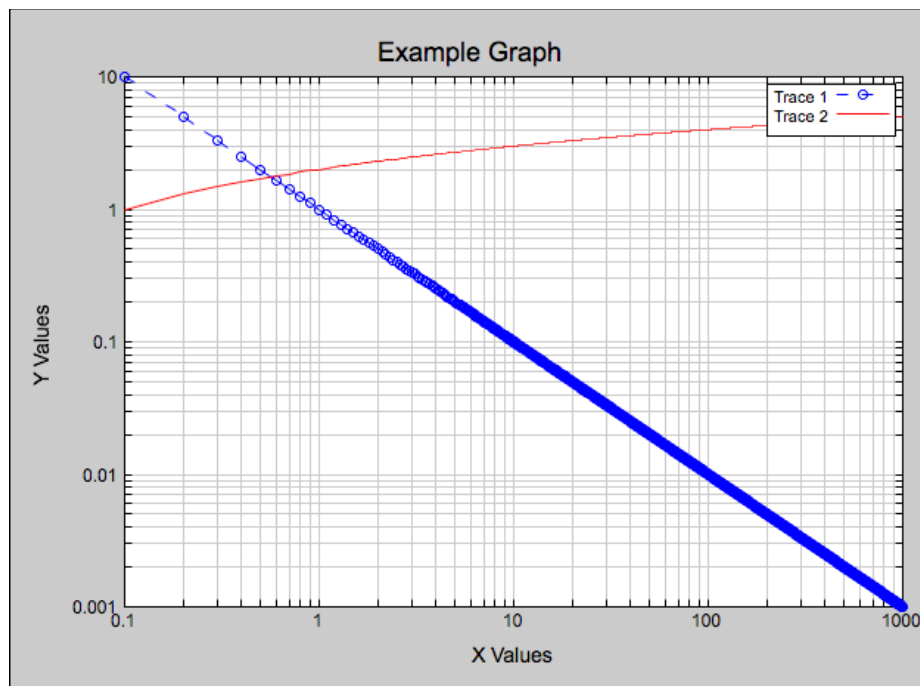


Figure 10: Example 3.