xGRAPH Manual: v3.4

September 3, 2021

1 Introduction

xGRAPH is a batch graphics function. It is intended to process quantities of graphics data, typically input as a cell array of multi-dimensional data. Theoretical or experimental data is passed to the *xgraph* multidimensional graphics program, including the complete *data* cell array, and a cell array of graphics *parameters* for plotting each graph:

- Calling xgraph: $xgraph(data, parameters)$

The output of *xgraph* is unlimited, but memory limits mean $100-1000$ individual plots, due to memory limits. The program also generates error comparisons and chi-squared values if required.

The data structure for input is as follows:

1. The input *data* is a cell array of *datasets,* although this can be collapsed to a single dataset
2. The *parameters* are also a cell array of parameter structures, which can be collapsed or omitted
3. The *dataset* is a cell array of multidimensional *graphs*, each with arbitrary dimensionality. A single graph is also possible.
4. Each *graph* is a real multidimensional array that can generate multiple plots, defined by the corresponding parameters.
5. The first index of each graph allows multiple line outputs, with different line-styles
6. The last index of each graph can optionally be used for multiple error and comparison fields.

When error fields are used, an input parameter is required to indicate the maximum error index.

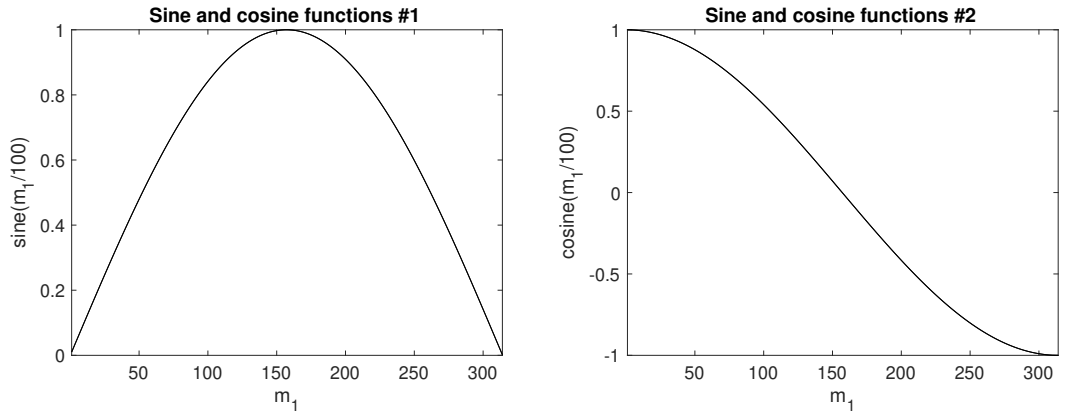

Figure 1: Example: xgraph output of two plots

1.1 Example

```
in.name     ='Sine and cosine functions';
in.olabels = {'sine(m_1/100)','cosine(m_1/100)'};
data        = {sin([1:100*pi]/100),cos([1:100*pi]/100)};
xgraph(data,in);
```

2 xGRAPH data

The data input to *xGRAPH* can come from a file, or from data generated directly from any compatible program.

The data is stored in a cell array *data* with structure:

$$data\{seq\}\{graphs\}(\ell, \mathbf{n}, e)$$

Each member of the outer cell array is a dataset cell array. These contain numerical graphics arrays, defining one independent set of graphical data. Comparisons and errors are plotted if there are errors and comparison data in the input. This generates comparison plots, as well as error totals and χ- squared error estimate when there are statistical variances available.

Multiple different graphs with different dimensionality can be obtained from each dataset, either through projections and slices or by generating additional datasets defined through graphics functions. Either or both alternatives areavailable.

Note that:

- *seq* is the length of a sequence of ouputs. If $seq = 1$, this outer cell array can be omitted.
- *graphs* is the number of graphs in one instance of the seqence. If $graphs = 1$, this inner cell array can be omitted.

The graphics data for a single dataset is held in a multidimensional real array, where:

- ℓ is the number of lines in the graph. Even if $\ell = 1$, this first dimension must still be retained.
- $\mathbf{n} = n_1, \ldots n_d$ is the number of plot-points in each dimension, where $d \geq 1$.
- e is the number of error fields and comparisons in the data. If $e = 1$, this last dimension can be omitted.
- If $e > p.errors$, the extra fields are used as comparison inputs, where $p.errors = 1$ is the default.

3 xGRAPH parameters

The complete cell array of the simulation data is also passed to the *xgraph* program, along with optional graphics parameters for each observable, to create an extended graphics data structure.

Some *xgraph* parameters are universal or default parameters, for every graph. However, most *xgraph* parameters are cell arrays by graph index, which replace the universal parameters like labels. These graphics parameters can be individually set for each output that is plotted, using the cell index $\{n\}$ in a curly bracket.

The axis labels, axis points and axis transformations of each graph are nested cell arrays, as there can be any number of axes. If only a single input is specified, the cell index is automatically supplied. Graphics defaults are also user-modifiable by editing the *xgpreferences* function. In the case of the logs switch, the observable axis is treated as an extra space-time dimension.

The graphics parameters can be individually set for each output that is plotted, using the graph index $\{n\}$ in a curly bracket. The axis labels, axis points and axis transformations of each graph are also cell arrays, as there can be any number of axes: so these are all nested cell arrays. If only a single input is specified, the cell index is automatically supplied.

The plotted result can also be an arbitrary function of the generated average data, by using the optional input *gfunction*.

Label	Typical value	Description
olabels{n}	*'a'*	Cell array of output data labels
minbar{n}	*0.01*	Minimum relative error-bar
gfunction{n}	@(d,~) d{n}	Functions plotted of averages
font{n}	18	Font size for graph labels
headers{n}	"	Graph headers
images{n}	0	Number of movie images
imagetype{n}	0	Type of movie images
transverse{n}	0	Number of transverse plots
pdimension{n}	3	Maximum plot dimensions
xfunctions{n}	-	Axis transformations
axes{n}	{1:m,1:m,..,}	Integer points plotted for each axis
logs{n}	[0,0,1]	Log switch for each axis
diffplot{n}	0	Difference plot switch for comparisons
cutoff	1.e-12	Lower cutoff, used for log plots and chi-squares
xk{n}	{1:m,1:m,..}	Coordinates plotted for each axis
glabels{n}	*{'m' 'x' 'y' 'z'}*	Graph-specific axis labels
graphs	-	Maximum number of graphs
errors	3	Total number of data fields, including errors
esample{n}	1	Controls size and type of sampling errorbars

- Note that up to 6 distinct types of input data can occur, indexed by the last index. The data always has index =1.
- There are can be up to two error bars (I and II), as well as comparison data, also with up to two error bars.
- With differences, the total comparison errors are treated as type (I), total data errors are treated as type (II).
- Type (I) errors have standard error bars, type II error bars - usually from sampling - have two solid lines.
- If *esample* = -1*,* both error bars are combined and RMS values plotted as a single error bar.
- Differences are plotted as unnormalized (*diffplot = 1)* or normalized (*diffplot = 2)* by the total RMS errors.

4 User definable graphics functions

It is possible to simply run **xgraph** as is, without much intervention. However, there are many customization options, including two user defined functions which have useful applications. These are as follows:

gfunction(d,p) is is a cell array of graphics functions whose output is plotted. There is one graphics function for each multidimensional dataset that is plotted. The default value is just the average stored in the input data array.

An arbitrary number of functions of these observables can also be plotted, including vector observables. The input to graphics functions is the observed data averages or functions of averages in a given sequence, stored in a cell array of size $d\{n\}(\ell, \mathbf{j}, e)$. If there are more graphics functions than input data cells, this will simply generate additional data for graphing.

xfunctions(nd,p) is is a cell array of axis functions whose output is plotted. There is one graphics function for each separate graph and plot dimension, nd. The default value is just the coordinate vector $xk\{nd\}$ stored in the input parameter structure, or else the index if this is omitted.

5 xGRAPH output graphs

This routine is prewritten to cover a range of useful graphs, but the generated graphs can be modified to suit the user.

The code will cascade down from higher to lower dimensions, generating different types of user-defined graphs. Each type of graph is generated once for each specified graphics function. The graphics axes that are used for plotting, and the points plotted, are defined using the optional ***axes*** input parameters, where $axes\{n\}$ indicates the n-th specified graphics function or set of generated graphical data.

If there are no ***axes*** inputs, or the inputs are zero - for example, $axes\{1\} = \{0, 0, 0\}$ - only the lowest dimensions are plotted, up to 3. If the ***axes*** inputs project out a single point in a given dimension, - for example, $axes\{1\} = \{0, 31, -1, 0\}$, these axes are suppressed in the plots. This reduces the effective dimension of the data - in this case to two dimensions.

Examples:

- $axes\{1\} = \{0\}$ - For function 1, plot all the first dimensional points; higher dimensions get defaults.
- $axes\{2\} = \{-2, 0\}$ - For function 2, plot the maximum value of x_1 (the default), and all higher-dimensional x-points.
- $axes\{3\} = \{1 : 4 : 51, 32, 64\}$ - For function 3, plot every 4-th x_1 point at x_2 point 32, x_3 point 64
- $axes\{4\} = \{0, 2 : 4 : 48, 0\}$ - For function 4, plot every x_1 point , every 4-th x_2 point, and all x_3-points.

Points labelled -1 indicates a default 'typical' point, which is the midpoint. If one uses -2, this is the last point.

The *pdimension* input can also be used to reduce dimensionality, as this sets the maximum effective *plotted dimension*. For example, $pdimension\{1\} = 1$ means that only plots vs x_1 are output for the first function plotted. Note that lower dimensions will always be replaced by corresponding higher dimensions if there are *axes* that are suppressed. Slices can be taken at any desired point, not just the midpoint. Using the standard notation of, for example, $axes\{1\} = \{6 : 3 : 81\}$, can be used to modify the starting, interval, and finishing points for complete control on the plot points.

The graphics results depend on the resulting **effective** dimension, which is equal to the actual input data dimension unless there is an *axes* suppression, described above. Since the plot has to include a data axis, the plot itself will have an extra data axis, as well as the lattice *dimension* axes.

One can plot only three axes directly using standard graphics tools. The strategy to deal with the higher dimensionality is as follows:

dimension=1 For one lattice dimension, a 2D plot of observable *vs t* is plotted, with data at each lattice point in time. Exact results, error bars and sampling error bounds are included if available.

dimension=2 For two lattice dimensions, one 3D image of observable *vs x,t* is plotted. A movie of distinct 2D graphic plots is also possible. Otherwise, only a slice through $x = 0$ is available, reducing the lattice dimension to 1.

dimension=3 For three lattice dimensions, if $images > 1$, a movie of distinct 3D graphic images of observables are plotted as *images* slices versus the first plot dimension. Otherwise, only a slice through the chosen point, or the default, is available for the highest dimension, reducing the lattice dimension to 2.

dimension>3 For higher lattice dimensions, only a slice through the chosen point, or the default midpoint, is available, reducing the lattice dimension to 3.

In addition to graphs versus x_1, the **xGRAPH** function can generate *images* (3D) and *transverse* (2D) plots at specified points, up to a maximum given by the number of points specified. The number of these can be individually specified for each graphics output. The images available are specified as $imagetype = 1, \ldots 4$, giving:

1. 3D perspective plots (the default),
2. grey-scale colour plots,
3. contour plots, and
4. pseudo-color plots respectively.

Eerror bars, sampling errors and multiple lines for comparisons are only graphed for 2D plots. Error-bars are not plotted when they are below a user-specified size, usually with a default of 1% of the maximum range, to improve graphics quality. Higher dimensional graphs do not output the error-bar data, for visibility reasons, but they are still recorded in the data files.

6 Public API

6.1 xGRAPH functions and objects

.. function xgraph (data [,input])

This is the xgraph graphics function. It takes computed simulation "data" and "input". It plots graphs, and returns the maximum difference "diff" from comparisons with user-specified comparison functions. The "data" should have as many cells as "input" cells, for sequences.

If "data = 'filename.h5'" or "data= 'filename.mat'", the specified file is read both for "input" and "data". Here ".h5" indicates an HDF5 file, and ".mat" indicates a Matlab file.

When the "data" input is given as a filename, input parameters in the file are replaced by any of the new "input" parameters that are specified. Any stored "input" can be overwritten when the graphs are generated. This allows graphs of data to be modified retrospectively.

.. _sec-parameters:

6.2 Input parameters and user functions

A sequence of graphs is obtained from inputs in a cell array, as "input = {in1, in2, ...}". The input parameters of each simulation in the sequence are specified in a Matlab structure. The inputs are numbers, vectors, strings, functions and cell arrays. All metadata has preferred values, so only changes from the preferences need to be input. The resulting data is stored internally as a sequence of structures in a cell array, to describe the simulation sequence.

The graphics parameters are also stored in the cell array "input" as a sequence of structures "in". This only need to be input when the graphs are generated, and can be changed at a later time to alter the graphics output. A sequence of simulations is graphed from "input" specifications.

If there is one simulation, just one structure can be input, without the sequence braces.

The standard way to input each parameter value is:

$$in.label = parameter$$

The standard way to input a function is:

$$in.label = @function - name$$

The inputs are scalar or vector parameters or function handles. Quantities relating to graphed averages are cell arrays, indexed by the graph number. The available inputs, with their default values in brackets, are given below.

Simulation metadata, including all preferred default values that were used in a particular simulation, can also stored for reference in data files. This is done in both the ".mat" and the ".h5" output files, so the entire graphics input can be easily reconstructed or changed.

Note that parameter inputs can be numbers, vectors, strings or cells arrays. To simplify the inputs, some conventions are used, as follows:

- All input data has default values
- Vector inputs of numbers are enclosed in square brackets, "[...]".
- Where multiple inputs of strings, functions or vectors are needed they should be enclosed in curly brackets, "{...}", to create a cell array.
- Vector or cell array inputs with only one member don't require brackets.
- Incomplete or partial vector or cell array inputs are filled in with the last applicable default value.
- New function definitions can be just handles pointing elsewhere, or else defined inline.

If any inputs are omitted, there are default values which are set by the internal :func:'xgpreferences' function. The defaults can be changed by editing the :func:'xgpreferences' function.

In the following descriptions, :attr:'graphs' is the total number of graphed variables of all types. The space coordinate, image, image-type and transverse data can be omitted if there is no spatial lattice, that is, if the dimension variable is set to one.

For uniformity, the graphics parameters that reference an individual data object are cell arrays, indexed over the graph number using braces "{}". If a different type of input is used, like a scalar or matrix, xSPDE will attempt to convert the type. The axis labels are cell arrays, indexed over dimension. The graph number used to index these cell arrays refers to the data object, and there can be multiple plots obtained, depending on the graphics input.

6.3 xGRAPH parameters

Paramaeters that apply to all graphs in dataset

name

Default: ' '

Name used to label simulation, usually corresponding to the equation or problem solved. This can be added or removed from graphs using the :attr:'headers' Boolean variable, as explained below.

- p.name = 'your project name'

olabels

Default: {' ', ...}

Cell array of labels for the graph axis objects. These are text labels that are used on the vertical graph axes. The default value is blank. This is overwritten by any subsequent label input when the graphics program is run:

- p.olabels{n} = 'string'

graphs

Default: number of observables

This gives the number of observables computed. The default is the length of the cell array of observe functions. Normally, this is not initialized, as the default is used. Mostly used to reduce graphical output on a long file.

- p.graphs >= 0

functions

Default: number of functional transformations

This gives the maximum number of graphs, which are functions of the observables. The default is the length of the cell array of input data. Normally, this is not initialized, as the default is typically used.

- p.functions >= 0

print

Default: 1

Print flag for output information while running xGRAPH. If "print = 0", most output is suppressed, while "print = 1" displays a progress report, and "print = 2" also generates a readable summary of the parameters as a record.

- p.print >= 0

olabels

Default: {'a', ...}

Cell array of labels for the graph axis observables and functions. These are text labels that are used on the graph axes. The default value is "'a_1'" if the default observable is used, otherwise it is blank. This is overwritten by any subsequent label input when the graphics program is run:

- p.olabels{n} = 'string'

parametric

Default: [0,0]

Cell array that defines parametric plots, if required, for each graph number. The first number is the graph number of the observable plotted on the horizontal axis for the parametric plot. The second number is the axis number where the parametric value is substituted, which can be time (axis 1) or x-coordinate (axis 2).

- p.parametric{n} = [p1,p2] >= 0

If both are zero, the plot against an independent space-time coordinate is calculated as usual. If nonzero, a parametric plot is made, for two-dimensional plots only. In both cases the vertical axis is used to plot the graph variable. The horizontal axis is used for either the independent variable or the parametric variable. In this version, only vertical error-bars are available. Can be usefully combined with p.scatters{n} to plot individual trajectories, but the number of scatters should be the same in each of the two graphs that are parametrically plotted against each other.

axes

Default: {{0,0,0,..}}

Gives the axis and points plotted for each plotted function. As special cases, "p = 0", is the default value that gives the entire axis, while "p = -1" generates a default point on the axis, at the midpoint. Other values are vector range indicators, for example "p = 5" plots the fifth point, while "p = 1:4:41" plots every fourth point. For each graph type "n" the axes can be individually specified. If more than three axes are specified, only the first three are used. The others are set to default values.

- p.axes{n} = {p1,p2,p3,..pd}

font

Default: {18, ...}

This sets the default font sizes for the graph labels, indexed by graph. This can be changed per graph.

- p.font{n} > 0

esample

Default: {1, 1 ...}

This sets the type and size of sampling errors that are plotted. If esample = 0, no sampling error lines are plotted, just the mean. If esample = -n, $\pm n\sigma$ sampling errors are included in the errorbars. If esample = n, separate upper and lower $\pm n\sigma$ sampling error lines are plotted. In both cases, the magnitude of esample sets the number of standard deviations used.

- p.esample{n} = e

minbar

Default: {0.01, ...}

This is the minimum relative error-bar that is plotted. Set to a large value to suppress unwanted error-bars, although its best not to ignore the error-bar information! This can be changed per graph.

- p.minbar{n} >= 0

images

Default: {0, 0, 0, ...}

This is the number of 3D, transverse o-x-y movie images plotted as discrete time slices. Only valid if the input data dimension is greater than 2. Note that, if present, the coordinates not plotted are set to their central value, when plotting the transverse images. This input should have a value from zero up to a maximum value of the number of plotted time-points. It has a vector length equal to *graphs*:

- p.images{n} = 0 ... p.points(1)

imagetype

Default: “{1, 1, ...}“

This is the type of transverse o-x-y movie images plotted. If an element is “1“, a perspective surface plot is output, for “2“, a gray plot with colours is output, for “3“ a contour plot with 10 equally spaced contours is generated, and for “4“ a pseudocolor map is generated.This has a vector length equal to :attr:‘graphs‘.

- p.imagetype{n} = 1, 2, 3, 4

transverse

Default: “{0, 0, ...}“

This is the number of 2D, transverse o-x images plotted as discrete time slices. Only valid if :attr:‘dimension‘ is greater than 2. Note that, if present, the y,z-coordinates are set to their central values, when plotting the transverse images. Each element should be from “0“ up to a maximum value of the number of plotted time-points. It has a vector length equal to :attr:‘graphs‘:

- p.transverse{n}=0 ... p.points(1)

headers

Default: “{’head1’, ’head2’, ...}“

This is a string variable giving the graph headers for each type of function plotted. The default value is an empty string “”“, which gives the overall simulation heading. Use a space “’ ’“ to suppress graphics headers entirely. It is useful to include simulation headers - which is the default - to identify graphs in preliminary stages, while they may not be needed in a final result.

- p.headers{n} = ’my_graph_header’

pdimension

Default: “{3, 3, ...}“

This is the maximum space-time grid dimension for each plotted quantity. The purpose is eliminate unwanted graphs. For example, it may be useful to reduce the maximum dimension when averaging in space. Higher dimensional graphs are not needed, as the data is duplicated. Averaging can be useful for checking conservation laws, or for averaging over homogeneous data to reduce sampling errors. All graphs are suppressed if it is set to zero. Any three dimensions can be chosen using the axes command.

- p.pdimension{n} ≥ 0

xlabels

Default: “{’t’, ’x’, ’y’, ’z’}“ or “{’x_1’, ’x_2’, ’x_3’, ’x_4’}“

Labels for the graph axis independent variable labels, vector length of :attr:‘dimension‘. The numerical labeling default is used when the “p.numberaxis“ option is set. Note, these are typeset in Latex mathematics mode!

- p.xlabels = {p.xlabels(1), ..., p.xlabels(p.dimension)}

klabels

Default: “{’\\omega’, ’k_x’, ’k_y’, ’k_z’}“ or “{’k_1’, ’k_2’, ’k_3’, ’k_4’}“

Labels for the graph axis Fourier transform labels, vector length of :attr:‘dimension‘. The numerical labeling default is used when the “p.numberaxis“ option is set. Note, these are typeset in Latex mathematics mode!

- p.klabels = {p.klabels(1), ..., p.klabels(p.dimension)}

glabels

Default: “{{’t’, ’x’, ’y’, ’z’}}“ or “{{’\omega’, ’k_x’, ’k_y’, ’k_z’}}“

Graph-dependent labels for the independent variable labels, nested cell array with first dimension :attr:‘graphs‘, second dimension :attr:‘dimension‘. This is useful if the axis labels

change from graph to graph, for example, if the coordinates have a functional transform.

- p.glabels{n} = {p.xlabels(1), ..., p.xlabels(p.dimension)}

limits

Default: “{[0,0;0,0; ...]}“

Graph-dependent limits specified as a cell array with dimension :attr:‘graphs‘. Each entry is a matrix of graph limits with the first index the dimension, the second index 1,2 for the lower and

upper limit respectively. This is useful if the limits required change from graph to graph.

- p.limits{n} = [t1,t2;x1,x2;y1,y2; ...,}

legends

Default: {{”,”}}

Graph-dependent legends, specified as a nested cell array of strings for each line.

- p.legends{n} = {labels(1), ..., labels(lines)}

lines

Default: {{’-k’,’--k’,’:k’,’-.k’,’-ok’,’--ok’,’:ok’,’-.ok’,’-+k’,’--+k’}}

Line types for each line in every two-dimensional graph plotted. If a given line on a two-dimensional line is to be removed completely, set the relevant line-style to zero. For example, to remove the first line from graph 3, set p.lines{3} ={0}. This is useful when generating and changing graphics output from a saved data file.

- p.lines{n} = {linetype{1}, ..., linetype{nl}}

6.4 xGRAPH user functions

gfunction (data,in)

This is a cell array of graphics function handles. Use when a graph is needed that is a functional transformation of the observed averages. The default value generates all the averages that are in the simulated data. The input is the data cell array of averages, and the output is the data array that is plotted. Note that in general the cell index is used to describe a given graph, while the first vector index in the graphed data indexes a line in the graph. For multidimensional data, the graphics program automatically generates several different projections of a given graph to allow a complete picture.

xfunctions (x_nd,in)

This is a nested cell array of graphics axis transformations. Use when a graph is needed with an axis that is a function of the original axes. The input of the function is the original axis coordinates, and the output is the new coordinate set. The default value generates the input axes. Called as p.xfunctions{n}{nd}(x_nd,in) for the n-th graph and axis nd, where x_nd is a vector of axis coordinate points for that axis dimension.

6.5 xGRAPH internal functions

xgraph(data,input)

xGRAPH(data,input) graphs multidimensional data files.

- Input: data cells 'data', input parameter cells 'input'.
- Output: graphs.
- If no numeric 'data' present, reads data from a file named 'data'. If data and no parameters exist, plots from default parameters.
- First data dimension is a line index, last dimension are the error-bars and comparisons
- Needs: xread, xmakecell, xgpreferences, xmultiplot

input = xgpreferences (input,oldinput)

- input = XGPREFERENCES(input) sets default values for graphics inputs.
- Input: 'input' cell array, and any previous input from a datafile, 'oldinput'.
- Output: the 'input' cell array with updated and default graphics values.
- Called by: xgraph
- Needs: xprefer, xcprefer