All the commands

I have been pretty bad at keeping the documentation up to date. So, rather than edit that, I am providing this shorter document that includes all the commands on all the menus of XPP. This also will tell you the file structures. *Every file in XPP is ASCII readable!! Thus, you can look at any file that XPP produces or reads in any plain text editor.* Okay, the `.gif` files are not readable, but all others are.

Some general features of the interface

- All the main menu commands have one-key shortcuts. These are indicated below. E.g. to quit type FQY. (File Quit Yes)
- Generally `ESC` will get you out of most stuff.
- As you slide over menu items, some tips will appear at the bottom.
- In the dialog boxes that come up, some items have a (*) next to them. If you click on the (*), you can get a list of allowable options such as variable names, colors, etc.
- By default `Return` moves between entries and `Tab` is like hitting `Ok`. If you want the opposite behavior, use the command line `-ee`. This mimics Windows and others.
- On any 2D graphics window and AUTO window, hold the mouse down and move and the coordinates are reported. In the 3D view, moving the mouse while holding down performs rotations.
- There are six buttons across the main window that bring up new windows allowing you to manipulate initial data `ICs`, boundary conditions `BCs`, delay initial conditions `Delay`, parameters `Param`, and the numerical values of the data `Data`. The `Eqns` button shows you the ODEs.
  - The `ICs` window has buttons `xvst, xvsy,arry` that allow you to quickly choose and plot stuff. Click on several check boxes and `xvst` and all the curves will be plotted vs time. Click on two check boxes and `xvsy` to get a phaseplane view. For PDEs and networks, click on a first item and a last item and then `array` and the `Array` window comes up (see (V)iew axes, below) with all the variables between the first and last plotted vs time.
  - The `ICs, BCs, Param, Delay` windows all have buttons across the top. `Default` restores values in the ODE file; `Cancel` cancels what you have typed `Ok` accepts it, and `Go` accepts it and runs the integrator. Arrow keys on side and on keyboard allow you to move between entries. You can resize the windows, but it is quirky and may crash. Use the `Delay` window to edit initial data for delay equations for $t < 0$. Note that clicking `OK` in the `ICs` window will replace the $t < 0$ data by the most recent data computed, so always go back and click `OK` in the `Delay` window to reset it after you change the `ICs` stuff.
- At the bottom of the main window are three slider buttons which let you vary parameters and initial data. Nullclines and direction fields are redrawn as well as trajectories. Click on the `Par/Var` part and fill it it, Then use the slider. Nullclines, direction fields, trajectories, and color coding are all redrawn. You can preset the sliders with `OPTIONS`; see `xpp_sum.pdf`.

Main Menu commands

- `(I)nitconds` You can exit a single integration by typing `ESC`.
  - (R)ange - bring up range dialog; choose the parameter you want to range over, the number of steps, start/end values, whether you want to reset the storage each time (otherwise it is appended), use old initial data (otherwise it uses the end of the last integration),cycle color (to see multiple plots), movie (erases each time and creates a movie to be played back with (K)inescope). You can exit a range integration if it is going by clicking the `/` key.
  - (2)par range - range over two parameters/variables; set the steps, etc like in above. Crv - means vary both parameters simultaneously along a one-dimensional line; Array- vary them independently (producing Steps*Steps2 solutions). *Note that when you use 1-parameter ranges; this is reset to Crv; so beware.*
  - (L)ast - start the new integration at the end of the last one. Very useful in getting rid of transients
  - (O)ld - default using the current initial conditions
  - (G)o - same as old, I think
  - (M)ouse - select initial data in a two-variable window using the mouse

  - (S)hift- like (L)ast, the time is also shifted. Useful for non-autonomous equations
  - (N)ew - type in initial conditions from the command window - type Esc to finish early
  - s(H)oot - use as initial conditions the stable/unstable manifolds found with the singulatr point command. Choose 1-4 according to the order they were computed. For stable manifolds, make sure you make DT negative
  - (F)ile - read initial conditions from a file. This is a plain ascii file that contains the values for all the initial conditions in order.
  - form(U)la - type in a formula. This is good for systems of PDEs etc. For example type in `u[0..99]` and then type in `sin(2*pi*[j]/100)` and this will initialize to those values. Hit return when done or type in another variable
  - m(I)ce - use mouse to put in multiple initial conditions
  - (D)AE guess - reset the guess for differential algebraic systems
  - (B)ackward - like (G)o but integrate backward in time
- (C)ontinue - add more time to the current solution extending the integrating time
- (N)ullcline
  - (N)ew - draw new nullclines
  - (R)estore - redraw previously drawn ones
  - (A)uto - automatically draws then after integrating
  - (M)anual - only draw if you use restore
  - (F)reeze
    - (F)reeze - save these nullclines on the screen
    - (D)elete all - delete all the saved nullclines
    - (R)ange - range over a parameter and draw all the nullclines
    - (A)nimate - draw the nullclines from the range integration above successively erasing after each redraw.
  - (S)ave - save the stored nullclines. They are stored as pairs of line segments with the X nullcline first and the Y-nullcline second. If you save the frozen and ranged ones, they will be saved in a separate file for each set.
- (D)ir.field/flow
  - (D)irect Field - prompts you for a grid (10-16 is good) and will draw unscaled direction fields. (These are saved and thus plotted)
  - (F)low - prompts for a grid (5 -8 is good) and draws trajectories forward and backward in time. They are not saved so they wont be printed, but you can capture the plot to save with the (K)inescope.
  - (N)o dir. field - don't draw it
  - (C)olorize - Use the parameters from (N)umerics (C)olor code to colorize the whole phase plane. A good grid is 100.
  - (S)caled Dir.Fld - direction field with unit length.
- (W)indow/zoom
  - (W)indow - set the view for the two-d axes (see (V)iew axes to set 3D stuff)
  - (Z)oom in - use the mouse to zoom in
  - Zoom (O)ut - use the mouse to zoom out
  - (F)it - use the current set of graphs to fit the window to hold everything. A very useful command!
  - (D)efault - set the window to the values when you first start
  - (S)croll - allows you to use the mouse to move the view around. Press and hold the mouse. Click ESC to get out of the mode.
- ph(A)sespace - Differential equations that lie on the circle can be readily plotted and handled with this command. This allows you to select the size of the circle (default 2π) and which are folded. Can do this within the ODE file using the `@ fold=name` option.

- (K)inescope - Lets you capture the image in the drawing window and play it back. Lots of other parts of XPP use this to make movies. (See the range commands, for example.)
  - (C)apture - capture the image on the screen
  - (R)eset - delete all captured images
  - (P)layback - cycle through the images using the left and right arrows, Escape to exit
  - (A)utoplay - cycles through the images specificed number times with specified time between frames in milliseconds
  - (S)ave - save the individual images as a series of Gifs.
  - (M)ake anigif - saves the images as one animated gif file. Playback on your browser or convert to some other video format (e.g. with Quicktime Pro)
- (G)raphic stuff
  - (A)dd curve - use this to add curves to the plot. A dialog will come up asking for the details. Colors are 0-10 (click on the (*) to see the actual colors). Linetype is 1,0, or negative integgers which produce filled circles of larger diameters. 0 is just dots.
  - (D)elete last - deletes the most recent curve. You cannot delete the primary curve, which is always on.
  - (R)emove all - removes all the curves
  - (E)dit curve - lets you edit the curve; 0 is the primary curve; added are 1-9. The dialog is the same as with (A)dd curve
  - (P)ostscript - produce postscript output. A dialog comes up first to let you decide the properties. Then the file selector comes up so you can choose the file.
  - S(V)G - produce scaleable vector graphics output. This is an XML-based format that is viewable on a web browser, infinitely zoomable, and is supported by many platforms including iOS. It has the disadvantage of being rather large in some cases.
  - (F)reeze - keeps lots of curves and other stuff on the screen so you can change stuff and still keep the curves.
    - (F)reeze -brings up a dialog box to let you freeze the current *primary curve (that is curve 0)* . Give it a color. The Key is the title for the legend. The name is just for convenience in choices below. Up to 26 curves can be frozen.
    - (D)elete - lets you delete selected curves (their names are listed; ESC gets you out).
    - (R)emove all - removes all frozen curves.
    - (K)ey - turn the key (legends) on/off and if you turn them on, position with the mouse.
    - (B)if.Diag - import a bifurcation diagram from Auto. (You save it in Auto using the Write Pts option.)
    - (C)lr. BD - clear the bifurcation diagram
    - (O)n freeze - each time you integrate, the new curve will be frozen, so you can automatically freeze up to 26 curves. This toggles to (O)ff freeze.
  - (A)xes options - Brings up a dialog. You can set the positions of the axes (thin dotted lines) and whether they appear or not and the postscript font size.
  - exp(O)rt data - save all the primary curves (those that you have added with (A)dd curve) into a text file that is space delimited. First column is the x-axis and the others are the y-axes for the viewed curves. In 3D mode, all three columns are saved.
  - (C)olormap - select the colormap that is used in the animator, the colorize options, and the array plot. There are 7 of them. `Normal` is like the `jet in matlab`.
- (N)umerics - see below; this is a huge menu!!
- (F)ile - see below; this is also big
- (P)arameters - change parameters manually from the command window. Type the name of the parameter and then type in a value. Hit return twice to exit. Type default to get the parameters to their default values (those when loaded). Better is to click the `Param` box in the top main window.
- (E)rase - erases the screen
- (M)akewindow - allows you to create new windows

- (C)reate - make a new window. This copies the current window (all the added curves, but not the frozen curves) and this window becomes the hot window. A small black box occurs in the upper left. You can edit this window as in the main window.
- (K)ill all - deletes all created windows
- (D)estroy - deletes the most recently created window cannot be destroyed.
- (B)ottom puts the active window on the bottom.
- (A)uto turns on a flag so that the window will automatically be redrawn when needed.
- (M)anual turns off the flag and the user must restore the picture manually.
- (S)imPlot on/off lets you plot the solution in all active windows while the simulation is running. This can slow you down abit.

- (T)ext etc - add text, arrows, other widgets. Up to 400 can be added.
  - (T)ext This prompts you for the text you want to add. Then you are asked for the size: 0-8pt, 1-12pt, 2-14pt, 3-18pt, 4-24pt. Text also has several escape sequences:
    * \1 – switches to Greek font
    * \0 – switches to Roman font
    * \s – subscript
    * \S – superscript
    * \n – neither sub nor superscript
    * \{expr} – evaluate the expression in the braces before rendering. This can be quite useful with sliders and range integration as this info is provided in the window.

    So that `\1a=\{alpha}` will produce the text $\alpha = .123$ where .123 was the value of the parameter `alpha`. If you change alpha, it will also change the text, when you redraw. Note that not all X-servers will have these fonts, but the postscript file will still draw them. Finally, place the text with the mouse.
  - (A)rrow This lets you draw an arrow-head to indicate a direction on a trajectory. You will be prompted for the size, which should be some positive number, usually less than 1. Then you must move the the mouse and select a direction and starting point. Click on the starting point and holding the mouse button down, drag the mouse to indicate the direction of the arrow-head. Then release the mouse-button and the arrow will be drawn.
  - (P)ointer This is like an arrow, but draws the stem as well as the arrow head. It can be used to point to important features of your graph. The prompts are like those for `(A)rrow`. Note that if you choose the arrow size to be zero, this is a straight line.
  - (M)arker This lets you draw little markers, such as triangles, squares, etc on the picture. When prompted to position the marker with the mouse, you can over-ride the mouse and manually type in coordinates if you hit the (Tab) key.
  - (E)dit This lets you edit the text, arrows, and pointers in one of three ways (M)ove, (C)hange, (D)elete. Select the object with the mouse.
  - (D)elete All Deletes all the objects in the current window.
  - marker(S) This is similar to the Marker command, but allows you to automatically mark a number of points along a computed trajectory. Choose the marker properties, the starting row of data, the number of markers, and the number of rows to skip between markers. Note each marker is counted separately so only 400 total are allowed. If you change to a different view (different parameters on the axes), they remain, so you may want to delete them.

- (S)ing pts - find equilibria
  - (G)o - use the current initial data to try to find equilibria with a Newton solver. If successful, you will be asked if you want the eigenvalues printed to the console; if there any special one-dimensional manifolds, you will be asked if you want them drawn. Then a summary window will appear. It shows the equilibria. It also tells how many complex with positive real parts (c+), etc. The `Import` button lets you load the equilibria as initial data. The initial data for one-dimensional manifolds is remembered, so they can be recreated as trajectories with the (I)nitialconds s(H)oot menu
  - (M)ouse - use the mouse to select a guess

  - (R)ange - range over parameters as you find equilibria. The dialog asks you the parameter, steps, etc. Choosing `Shoot=Y`, will draw the relevant one-dimensional invariant manifolds. `Stability col.` will put the stability info into one of the data browser columns. (You should add an extra column if you don't want to overwrite info like the fixed point. Columns are labeled starting with 1.) `Movie=Y` will create a Kinescope clip at each point - this is only useful if there are invariant manifolds to be drawn. When done, the first column of the browser has the parameter and the remainng, the equilibrium values or stability info (Unstab.stab) Choose the Monte Carlo and it will act like the Monte Carlo below but will run through parameters. This should be used with the Movie feature on. You have to run Monte Carlo once before using this to set up the parameters.
  - monte(C)ar - Randomly guess starting values to try to find all the equilibria. You will be prompted to append (add onto what you already have), `Shoot` (draw one-dimensional manifolds when relevant) , number of guesses, tolerance (if their difference is less than the tolerance, they will be treated as the same point and not stored; otherwise you would have lots of repeats), and the ranges for each variable. This is a great tool to get the full behavior of all the fixed points, etc. Try `wcfun.ode`.

- (V)iew axes - choose some viewing options
  - (2)D view - select the axes, the window, and the labels.
  - (3)D view - select the axes, the max/min of the three axes variables, and the 2D viewing window. (I ignore these and use the convenient (W)indow (F)it; then go back and refine it). The (3)d params will give you rotation options, etc
  - (A)rray - brings up the array plot window which is for large systems such as a discretized PDE. Suppose you have a system with variables called `U0, U1,...U199` and you integrated for 50 time units with a stepsize of 0.05 (so there are 1000 steps) and you want to plot them vs time. Then pick `Column 1` as `U0`; `Ncols=200`; `Row 1:0`; `NRows:201`; `RowSkip:5` and leave the rest as is and click `OK`. This will plot the data colored according to the magnitude with time running down. (The color map used is chosen in the (G)raphic stuff (C)olormap.) If you want to skip everyother column, set `Colskip:2`, etc. There are 7 buttons: `Redraw,Close` are obvious; `Edit` brings up the initial dialog box; `GIF` saves a GIF of the current view (without the scale bar); `Fit` will use the chosen variables to find the best scaling. `Range` is really cool! This allows you to range through parameters, redrawing after each integration. A dialog comes up prompting for the filename, and whether you want the image tagged with the parameter value (`Tag`) whether you want to save one animated GIF (`Still:0`) or a sequence of individual GIFs (`Still:1`). The dialog from `(I)nitconds (R)ange` comes up.
  - `(T)oon` brings up the animation window. See below!

- `(X)i vst t` prompts you for a variable and plots it vs time; it automatically scales the window.

- `Restore` redraws everything

- `(3)d-params` Brings up a dialog allowing you to change the perspective, etc. You can play with these and then click `(W)indow (F)it` to get it in the window. The `Movie` option lets you create a series of Kinescope frames as you vary the angle through various increments. Use the `(K)inescope` to view and save them.

File Menu commands From the main menu, click on `(F)ile` to get this menu.

- `(P)rt Src` brings up the source code of your ODE file. The buttons allow you to navigate if it is a big file. (up/down arrows, PgUp/Dn, Home/End on the keyboard also work.) `Kill` closes the window. There is one cool button called `Action.` This allows you to run a sort of tutorial using the ODE file. To see it work, try the file `lecar.ode` and click on the `Action` button. Most of the text goes away leaving stuff with little asterisks. If you click these, some parameters are changed in the model and you can follow the directions. The author of the ODE file must write the relevant code. Note that even if you turn off the bell, clicking the asterisks makes a beep to let you know something was done.

- `(W)rite set` allows you to store much of the info from your current session for later recall. You will be prompted for a file name. Numerical parameters, ODE parameters, initial data, graphics information, etc are all stored in a human readable ascii file. Use this a lot so that if you have something interesting, you will be able to reproduce it. I have had so many students that could not reproduce their results because they did not remember how they got them! Use it in conjunction with the next command. The standard name for these files is `*.set` and if you use `XPPAUT` from the command line, you can add the line `-setfile bob.set` to automatically start up with the saved set file as your starting parameters etc. Set files are not very smart so that if you change your ODE file, it will probably not work. However, they can be edited to fix this, but you have to know what you are doing!

- `(R)ead set` reads a saved set (see previous command). You will be prompted for a file name.

- (A)uto brings up the AUTO bifurcation package window. See below for its usage.
- (C)alculator brings up a window which will give results of calculations that you type in; e.g., sin(sqrt(3)/pi) will return 0.523819... You can use any parameter names and variable names; in the latter case it returns their current value. (If you have integrated the equations, it is the last value.) You can set parameters and initial conditions as numbers or formulae using the following m:minf(v) for example. Type ESC to exit the calculator.
- (E)dit allows you to edit the functions, etc.
  - (R)HS's Lets you edit the right-hand sides of the equation. It works OK with simple right hand sides and if there are just a few of them. Don't try it for really complex ones as it will either crash or just give an error.
  - (F)unctions lets you edit user-defined functions.
  - (S)ave as seems to be a left over from a vary old version of XPPAUT and will save a version in the old style of the original DOS program PhasePlane; it may or may not work :)
  - (L)oad DLL allows you to load a dynamically linked library and associated function. See the ode tstdll.ode in the ode directory that comes with XPPAUT. You can add options to the ode file to automatically load the desired library and functions.
- (S)ave info provides a summary of the important info about the numerics, parameters, initial data, etc. It cannot be read by XPPAUT but can be easily read by the user.
- (B)ell on/off toggles the annoying bell noise that XPPAUT makes.
- (H)elp Assuming you have told XPPAUT where to find the help directory and what browser you use, this should fire up a rather out of date help file. You have to define two environmental variables, XPPHELP which points to the help directory in the XPP distribution and XPPBROWSER which is the command to open a web browser. On the Mac, in a terminal, I write:

  ```
  export XPPHELP=/Users/bard/xppaut/help/xpphelp.html
  export XPPBROWSER=open
  ```

  since open seems to know how to handle html files by opening Safari.
- (Q)uit Well, duh
- (T)ranspose opens up a dialog box that allows you to transpose the data in the browser. The Time column is replaced the integers, 1 to n where n is the number of columns. For each of the m rows the first m columns after the time column contain the transposed data. This is a useful command for looking at discretized PDEs where you can then look at their spatial profile. Clicking cancel in the dialog will put the data back into the untransposed form.
- t(I)ps turns on/off the tips that are shown at the bottom of the screen.
- (G)et par set loads a predefined parameter set. These can be initial conditions, parameter values, and even controls for the graphs. See lecar.ode for examples, declared via the set command in the ODE file
- c(L)one will attempt to write a new ODE file with the current initial data and parameters. You will be prompted for the file name. If the ODE file uses the array formulation, this will make a pretty big unreadable file.
- .(X)pprc will bring up your .xpprc file if you have defined the environmental variable XPPEDITOR.
- t(U)torial brings up a series of little suggestions. A work in progress to be sure.

Numerics Menu commands From the main menu, click on nUmerics to get this menu. It is a permanent menu in the sense that it replaces the main menu until you exit it. This presumes you would want to do a bunch of numerical settings at one time.

- (T)otal allows you to set the total time to integrate.
- (S)tart time sets the initial value of t. This is relevant for nonautonomous systems.
- t(R)ansient is a very useful setting. XPPAUT will only save data once t exceeds transient. So this lets you throw out data that may be a transient before you reach a steady state.

- `(D)t` sets the step size for the integrator. If this is negative, then XPP integrates backwards in time. For adaptive integrators, this sets the frequency that numbers are output.
- `(N)cline control` sets the nullcline mesh. Larger values make for smoother nullclines but will take longer to plot.
- `s(I)ng pt control` sets parameters for finding equilibria: the number of Newton iterates, the Newton tolerence, and the perturbation size for computing the Jacobi matrix.
- `n(O)utput` sets the frequency of output for fixed step size integrators. Adaptive integrators ignore this.
- `(B)ounds` sets the bounds in XPP. If the magnitude of any variable or auxilliary quantity exceed `Bounds` then XPP stops integrating.
- `(M)ethod` lets you choose from many integration methods:
  - `(D)iscrete` is used to discrete dynamical systems
  - `(E)uler` is the standard one step method and is what you should use for stochastic ODEs
  - `(M)od. Euler` is a two-step method also called Heun's methos
  - `(R)unge-Kutta` is the well-known fourth order method
  - `(A)dams` is a fixed step Adams Bashforth predictor corrector
  - `(G)ear` is an adaptive step size stiff solver. You should provide a tolerance (smaller is slower and more accurate), minimum step size and maximum. The maximum should be bigger than `Dt`. Note that with this and other adaptive integrators, `Dt` does not affect output, just its frequency.
  - `(V)olterra` is a fixed step size backward integrator for Volterra integral equations. It asks for tolerance (this is for the newton solver, not accuracy), maximum iterations (ditto), and maximum points (how much history is kept in the buffers; integral equations use all data back in time), and whether you want the convolutions etc automatically re-evaluated whenever you change parameters. (You probably do.)
  - `(B)ackEul` is a stiff version of Euler (first order) but backward solver. It asks for a tolerance and a maximum number of iterations, both related to the Newton solver
  - `(Q)ual. RK4` is a fourth order nonstiff adaptive integrator. Smaller tolerances are more accurate
  - `(S)tiff` is the stiff solver in NRC and is related to Gear's method
  - `(C)vode` is CVODE, an adaptic stiff solver that askes for relative and absolute tolerance. This and Rosebrock are the only stiff solvers that can exploit a banded system; that is one whose Jacobi matrix is zero at some distance from the diagonal. It is the recommended integrator for stiff systems.
  - `DoPri(5),DoPri(8)3` are Dormand-Prince solvers that are related to Runge-Kutta and are adaptive. MatLab uses these as its default.
  - `Rosen(2)3` is a Rosenbrock stiff integrator that is used by MatLab. Like CVODE it can handle banded systems
  - `s(Y)mplectic` is a first order symplectic integrator for conservative systems. ODEs have to have a special form for this. E.g:

$$\begin{aligned} x_1' &= v_1 \\ v_1' &= F_1(x_1,\ldots,x_n) \\ x_2' &= v_2 \\ v_2' &= F_2(x_1,\ldots,x_n) \end{aligned}$$

- `d(E)lay` allows you to set delay equation parameters The maximum delay sets the biggest delay you can have in your equations. Real and imag guess are the guesses for roots to the exponential polynomial equations that you see in delay equations. The grid size determines how the accurate the contour integration is to find instabilities of the linearized equations.
- `(C)olor code` allows you to color your trajectories according to the value of some quantity. The choices are `(N)o color` which turns this off, `(V)elocity` colors according to the maginitude of the vector field, and `(A)nother quantity` which can be another variable, time, or any auxiliary variable such as energy. You are asked for the ranges or whether you want to optimize, which will automatically scale it for you. This is really cool in conjunction with `Dir. Fld Colorize`.

- `stoc(H)ast` is a rather large menu that has many things related to the analysis of noisy systems and some signal processing.
  - `(N)ew seed` sets a new seed for the random number generator. As this is a deterministtic rule, if you set the seed to the same number, all simulations will be repeatable.
  - `(D)ata` brings back the results of the simulation. If, for example, you took a histogram or did a spectral analysis, the data is replaced in the Browser. This command gets it back. This command allows you to apply many different analyses to your data without losing it.
  - `(C)ompute` brings up the menu for range integration. After you have integrated over the range (and you can range over any parameter without changing it; if it is a noisy system it will compute many sample paths); the mean and variance are computed pointwise. You access them with the `(M)ean` and `(V)ariance` items that appear in the menu
  - `(H)istogram` allows you to bin any quantity and produce a histogram. The binned data is in the first two columns of the browser. The `condition` choice lets you add additional conditions on other variables, say, in a Markov process with many states, you might want to look at the histogram only in a particular state.
  - `(O)ld hist`, brings back the histogram if you have used the `Data` command. Not sure why I even have this; in the old days, maybe, computers were too slow.
  - `(F)ourier` takes the FFT of the quantity you have chosen. The first column is the frequency, the second the cosine term, the third the sine term.
  - `(P)ower` returns the absolute value of the FFT in the second column and the phase in the third. For example if there is a peak at the frequency 0.133, this means that the period is about 7.5 in the units of the model.
  - `f(I)t data` is quite complicated. See the documentation file for details. Roughly, you specify a data file whose first column has equally spaced time points, and second etc columns contain data. You must specify the total number of columns, even if you don't use all of them. Type a comma or space-separated list of variables to fit, and the corresponding list of columns, e.g. to fit x to column 3 and y to column 5, you write x,y on the left and 3,5 on the right. Give a list of parameters and variables (initial data can be varied) whos values you want to change to fit the data. The numbet of points in the data file should be specified. You should set the integration time to match that of your data. Tolerance determines when the least squares isnt changing significantly; epsilon is for numerically getting the variational equations, and maxiter is the maximum iterations. The default values are OK. If you get failure, increase maxiter or sometime just run a few times. You should always start with a reasonable initial guess.
  - `(S)tat` will return the mean and standard deviation of a column of data in the data browser
  - `(L)iapunov` computes the maximal Liapunov exponent of a computed solutions to your ODE. You can range over parameters and the exponent will be saved along with the value of the parameter.
  - `st(A)utocor` is a specialty function that will plot the spike-time autocorrelation. Try running `qif-noise.ode` and then clicking on this function. Choose 200 bins from -100 to 100 and pick `t` as the variable. Plot `V vs t` and you will see the spike time autocorrelogram.
  - `(X)correl etc` does a auto/cross correlation using a direct method or the FFT. (FFT is fast but sometimes has artifacts). The first two columns contain the result. Choose the bin number and the variable(s). For example, if you have integrated with an output time step of 0.05, then picking a bin of 1001 (`XPP` will make it odd), then you will get a cross correlation with 500 negative and 500 positive times and when plotted will range from -25 (500*.05) to 25.
  - `sp(E)c dns` computes various types of spectral density estimates for your system. You can do a PSD, cross PSD, or coherence (normalized cross PSD). Choose the window size and the window type as well as the variables. Small windows give you more estimates to average over but the spectral resolution is crude. You may want to experiment with various settings.
  - `(2)D histogram` allows you to produce histograms in two variables. The first two columns of the Browser contain the bins and the third column, the fraction of points. 2D histograms are normalized unlike 1D, for some reason :-)
- `(P)oincare map` allows you to compute stroboscopic plots. The choices are `(N)one`, `(S)ection`, `(M)ax/min`, `(P)eriod`. Section will output when a specified quantity crosses the section from above or below; Max/min will output when the derivative of a specified quantity changes sign; Period is like Section, but the time column is replaced by the interval between crossings instead of the time of the crossing. You will get a dialog box that asks for the variable (it can be an auxiliary quantity or a variable), the value of the section, the direction, and whether you want to stop the integration when the section is hit. The direction is +1 if the variable crosses below or +1 for maximum; -1 for crossing above or from minimum; and 0 for both.

- `r(U)elle plot` allows you to plot shifted versions of variables against each other. Three axes are presented. The shift is a nonnegative integear and corresponds to a shift in rows in the Browser.
- `loo(K)up` allows you to edit tables. You can replace them with different ones, etc. The `(V)iew` option will load the table into the first two columns of the Browser. Note you cannot replace a formula style table with a file style and vice versa.
- `bnd(V)al` allows you to set parameters for the BVP solver. The BVP solver uses Newton's method, so the meaning of the three parameters is similar to that in `s(I)ng pt ctrl`.
- `(A)veraging` provides a set of tools for weak coupling analysis.
  - `(N)ew adjoint` command will compute the normalized periodic orbit to the adjoint equation obtained by linearizing about an exponentially stable limit cycle, transposing the Jacobi matrix, and multiplying by -1. The user should integrate precisely one period of the limit cycle. If convergence occurs, `XPP` returns the adjoint in each of the corresponding columns.
  - `(M)ake H` computes the interaction function:

$$H(\phi) := \frac{1}{T}\int_0^T Z(t)\cdot G(U(t),U(t+\phi))\ dt$$

    where $Z(t)$ is the adjoint, $U(t)$ is the periodic orbit, and G is the weak interaction. In typing the right-hand sides, use the variable names for the components of $U(t)$ and their primed counterparts for $U(t+\phi)$. For example, for diffusion of voltage, the right-hand dise is written `V'-V` which means $V(t+\phi)-V(t)$. If you have 4 or more columns, the third and fourth will be replaced by the odd and even parts of the interaction function. The first column is time (ϕ) and the second is H.
  - `(A)djoint,(O)rbit,(H)fun` bring back the previously computed adjoint, orbit, and interaction function, respectively.
  - `(P)arameters` allows you to change the parameters for the adjoint computation.
  - `(R)ange` sets a flag, `AdjRange` to ON for computing averaging functions over a range of parameters. This is tricky to do, so here is how.
    1. First compute the adjoint and the H function for some value of the parameters. This is so you can input the coupling interactions.
    2. Set `tRansient` to some big number and `Total` to even larger. This is so you can relax to the limit cycle.
    3. Set `Poincare map` to `Period` and pick a section that will be crossed over the entire range of parameters that you run. Set `Stop on section=1`. Now if you change parameters and integrate, it should compute the period.
    4. Now do the `Init conds Range`. For each parameter, the orbit, adjoint, and H function will be computed and stored in files of the form `orbit.XXX.dat, adjoint.XXX.dat, hfun.XXX.dat` where `XXX` is a unique name that gives the value of the parameter.
    5. After it is done, the flag `AdjRange` is set to off.
- `Esc` gets you out of the numerics menu

The Browser is a like a spread-sheet that allows you to look at the numbers, manipulate them and do other things that are fairly useful.

- `Up,Down,PgUp,PgDn,Left,Right,Home,End` aid in navigation; they have obvious keyboard equivalents.
- `First, Last, Restore` allow you to mark a segment of data to show in the drawing window. The `Restore` command in the main window ignores this.
- `Find` allows you to find where a chosen variable is closest to a chosen value. The corresponding row is brought to the top line of the Browser. This is useful for finding the maximum and minimum of a variable. Just choose a real big (small) number and it `XPP` will find the closest value.
- `Get` will load the first row of the Browser as the initial data
- `Write` will write the entire contents of the Browser to a specified data file in space delimited ascii
- `Read` will read data into the Browser that is in space delimited ascill

- `Table` will create an `XPP` table for the given column of data. You should choose a name, a column and the x-range
- `Close` Closes the Browser window
- `Del Col` is supposed to delete an added column, but it doesn't work, so don't bother trying :)
- `Replace` allows you to replace one column with some formula relating the other columns. Everything is done row by row. For example, if column 2 is `v`and column 3 is `w`, then you could replace `v` by `v+w` and the contents of column 2 would be replaced by the sum of coumns 2 and 3. You can integrate a colum using the & symbol and differentiate it using the @ symbol. So replacing `w` with `@w` would replace column three by its numerical derivative. `XPP` uses `NOUT*DELTAT` as the step for the derivative and the integral. The `&, @` can only be used on isolated variables and cannot be part of a general expression.
- `Unreplace` undoes the most recent replace
- `Addcol` lets you create a new named column. The new column can be any valid combination of the others as in the `Replace` button.

Auto Window is brought up from the `File Auto` command via the main menu. AUTO is a powerful bifurcation/continuation package for which this window is an interface to `XPP.` You should read the regular documentation for how to use this. When running, you can stop AUTO (usually) by either tapping `ESC` or clicking on the `ABORT` button. The menu commands are

- `(P)arameter` defines the up to eight parameters that AUTO is aware of. You should always set these first. By default, they are the first eight parameters in your ODE file.
- `(A)xes` both sets up the plotting axes and also tells AUTO the parameters you wish to continue with:
  - `(H)i` will plot the maximum value of the variable vs the parameter. A dialog box asking for windows, dependent variables, and also continuation variables is presented. Note that the window you provide does not affect the range of the continuation however the choice of parameters is exactly the set of parameters that AUTO will use to continue.
  - `(N)orm, h(I)-lo,(P)eriod,(A)verage,f(R)equency` are all different things that can be plotted on the y-axis and produce the same dialog box. For `Period,Frequency`, only periodic orbits will be drawn. The `Norm` shows the L^2 norm of periodic orbits and fixed points; the `Average` will plot the equilibrium values or the average over a periodic orbit.
  - `(Z)oom in, zoom (O)ut` let you control the viewing area with the mouse.
  - `(F)it` will automatically choose the dimensions of the plot based on the diagram. It is not particularly smart and is worse than the Fit in the main `XPP` window.
  - `(D)efault` is the window that you started with
  - `(T)wo par` will set AUTO up for a two parameter bifurcation. It requires that you specify the two parameters you wish to continue with.
  - `last (1) par, last (2) par` set the window to the last 1-parameter (2-parameter) continuation that you used.
  - `(S)croll` lets you scroll around the window. Click on ESC to exit.
- `(N)umerics` is the main dialog to set numerical parameters for AUTO. The parameters are
  - `Ntst` is the number of mesh intervals used for discretization of the collocation (Increase this if AUTO won't continue)
  - `Nmax` is maximum number of points for a given branch
  - `Npr` is how often points will be printed out in entirety. All special points are printed out. A point that is printed out entirely can be used to restart a continuation; regular points cannot be.
  - `Ds` is the starting step size for the continuation; the sign determines the direction as well
  - `Dsmin` is the smalled allowable step size before AUTO will quit (Decrease this if AUTO won't continue).
  - `Dsmax` is the maximum step size allowed. (Decrease this to get finer plots).
  - `Par Min, Par max` determine the parameter range over which AUTO will run the continuation

- `Norm mi, Norm max` determine the range of the dependent variable over which AUTO will run the continuation
- `Ncol` is the numberof Gauss collocation points per mesh interval $2 \leq \text{Ncol} \leq 7$
- `EPSL` is the relative tolerance for the parameter in Newton's method (decrease this if AUTO won't continue).
- `EPSU` is tolerance for the dependent variable in Newton's method
- `EPSS` is the tolerance for the arclength in detecting bifurcations
- `The rest` There are 7 other parameters in the last column that are also AUTO parameters but in 20 years of running XPP with AUTO, I have never found a reason to change them. If you have to ask what they mean, then you don't probably want to change them; they are for experts only.

- `(R)un` is a context-dependent menu and the choices depend on the current state of AUTO. Once you have run for the first time, the choices will change. For example, if you find a Hopf bifurcation, then there are 4 choices. Others may have more or less. There are four starting states:
  - `(S)teady state` continues from a fixed point. Make sure that you have initial conditions that are right on the fixed point.
  - `(P)eriodic` continues from a periodic orbit that you have computed for one full cycle.
  - `(B)dry Value` continues from a solution to boundary value problem. You should have used the BVP solver in `XPP` to get one good starting solution.
  - `(H)omoclinic or H(E)teroclinic` will try to solve homo/heteroclinic solutions. This requires that you put approximate values of the equilibrium point (points) and the dimensions of the stable and unstable manifolds into the little dialog and also have run an integration with an approximate solution. A good way to get an approximation is to use the shooting feature of XPP `Sing. Pts.`. This starting point should be run in Two-parameter mode.
  - `Other starting conditions.` AUTO marks special points in several ways and these are the only points that you should start with once you are up and running. They are
    - `EP`, an endpoint of the calculation (say you have reached a boundary). Extend the bound and extend the diagram. Almost all special points can also be extended. Not very common, but you may need to do it.
    - `LP`, a limit point or fold. You can continue this in 2 parameters
    - `HB`, a Hopf bifurcation. You can compute the periodic branch coming from this or do a two-parameter continuation
    - `BR`, a branch point. You can compute the other branches or do a two-parameter continuation.
    - `UZ`, a user defined point. For periodic orbits, you can continue in two parameters as a fixed period.
    - `TR`, a Torus bifurcation. Continue in two-parameters
    - `PD`, a period doubling bifurcation. Either compute the new branch or two-parameter continue.
    - `MX`, the dreaded MX. AUTO cannot continue.
    - **Note** that two parameter continuations produce solid curves with different colors for each type.

- `Grab` allows you to move along the diagram. Use the left/right arrow keys and the `Home, End` keys to navigate. Use the `Tab` key to jump to specially marked points. Use the up/down arrows to navigate to specific types of special points. Use the mouse to move to a specific location. You can mark a portion of a branch and then have `XPP` solve the ODE for all parameters within the marks. Use the `s` key for the start and the `e` for the end. Type `Esc` to exit the Grab without doing anything and type `return/enter` to Grab the point. If you grab a special point (indicated by a little number in the diagram) then you can run the continuation with this as a starting point. If you grab a regular point, then the parameter and initial data are loaded into `XPP` and you can then look at the solutions by simulating them.

- `Usr period` allows you to mark specified values of your parameter or when the oscillation has a specific period. Just choose how many you want and then type in the values; use `T=35` for example to mark all periodic orbits where the period is 35; use `I=20` to mark all solutions where the continuation parameter, `I` takes the value 20, etc.

- `(C)lear, re(D)raw` erases/redraws the diagram

- `(F)ile` produces an additional menu:
  - `(I)mport orbit` loads a specially marked orbit into the Browser so you can plot it, etc. Note that you do not have to run the integrator; the orbit is already computed by AUTO.

- (S)ave diagram saves all the information in the diagram in a special form that you can later reload. I use this without starting the continuation to save the numerical settings as these are also kept.
- (L)oad diagram Loads a saved diagram
- (P)ostscript,S(V)G will produce postcript (SVG) of the current window.
- (R)eset diagram deletes the entire diagram.
- (C)lear grab removes the point from the grab buffer. This is useful if you want to restart with a new starting point, but do not want to delete the diagram. AUTO always checks the grab buffer when asked to Run.
- (W)rite pts writes the x, y coordinates out to a files of the current diagram along with information about the points. Specifically, what is plotted out is x,y1,y2,type,br,twop where x is the x-axis parameter, and y1,y2 are the max and min of the y-axis (if you have chosen Hi-lo axes). br is the branch number. type=1,2,3,4 for stable equilibria, unstable equilibria, stable periodics, unstable periodics. twop gives information about what kind of two-parameter bifurcation is plotted. 0 means 1-parameter. These files can be loaded into the main XPP window using the Freeze Load diagram command. I have tried to make it fairly smart so that only the data that is shown in the plot window will be saved.
- All info will write out lots of information into a file. Specifically, each row has the following:

  type br twop par1 par2 period uhigh[1..n] ulow[1..n] evr[1] evm[1] ...evr[n] evm[n]

  Here type, br, twop are as above; par1 par2 are the values of the two parameters (one will not vary in one-parameter plots, but will in two parameters); per is the period of any oscillations; uhigh[1..n] are the maxima of the n variables; ulow, the minima. The pairs evr,evm are the real and imaginary parts of the eigenvalues.
- init (D)ata will create a file with the initial conditions corresponding to each of the points on the diagram. The first column is the type, the second the branch, the third the parameter, and the remaining columns are the initial data for each variable.
- (T)oggle redraw will turn on/off redrawing of the diagram. Sometimes (in Windows, especially), XPP will redraw the diagram dozens of times which can be really time consuming. This sort of fixes it.
- auto ra(N)ge will let you run a range of simulations in XPP using data from one of the branches in your diagram that you have marked (see Grab).When you click on this, the Range Dialog will come up. Ignore the questions about the range of parameters etc, a this will be done automatically based on the part of the diagram you grabbed. Then let it rip and it will do a standard range integration using the parameters and initial data from the diagram
- se(L)ect 2Par pt allows you to pick pairs of parameters from the two-parameter view. Click anywhere in the window and as you move the mouse, the parameter values are shown at the bottom of the window. When you release the button, the last values are stored. Clicking on this menu command will copy them to the XPP parameter window.
- draw laBled sets flags that will import the orbits from labeled points and automatically draw them in the XPP window. You can erase before drawing each one or just overwrite. You can also toggle this off. Note that if this is on and you have a very complex diagram with many labeled periodic orbits, it will often crash, so use it for simple diagrams!

Animation window

The animation window is evoked with the Viewaxes Toon click. The animator reads an animation script, usually called something.ani. I will first decribe the scripting commands and then the controls. Most of the commands put simple geometric structures on the screen and their size, shape, color, and position can all be dependent on the dynamic variables in your ODE. Several other commands determine the global nature of the animation. The grab command may be the coolest one in the box, as this lets you manipulate objects in the animation window such that they can change the parameters and initial data in the ODE.

dimension xlo;ylo;xhi;yho

speed delay

transient

permanent

line x1;y1;x2;y2;color;thickness

```
rline x1;y1;color;thickness

rect x1;y1;x2;y2;color;thickness

frect x1;y1;x2;y2;color

circ x1;y1;rad;color;thickness

fcirc x1;y1;rad;color

ellip x1;y1;rx;ry;color;thickness

fellip x1;y1;rx;ry;color

comet x1;y1;type;n;color

text x1;y1;s

vtext x1;y1;s;z

settext size;font;color

xnull x1;y1;x2;y2;color;id

ynull x1;y1;x2;y2;color;id

grab x1;x2;d
    {u1=f1(mouse_x,mouse_y);...;un=fn(mouse_x,mouse_y)}
    {v1=g1(mouse_x,mouse_y,mouse_vx,mouse_vy);...;runnow=[0|1]}

done
```

All commands can be abbreviated to their first three letters and case is ignored. At startup the dimension of the animation window in user coordinates is (0,0) at the bottom left and (1,1) at the top right. Thus the point (0.5,0.5) is the center no matter what the actual size of the window on the screen. `Color` is described by either a floating point number between 0 and 1 with 0 corresponding to red and 1 to violet (if the default color map is used). When described as a floating point number, it can be a formula that depends on the variables. In all the commands, the color is optional *except* `settext.` The other way of describing color is to use names which all start with the $ symbol. The names are: `$WHITE, $RED, $REDORANGE, $ORANGE, $YELLOWORANGE, $YELLOW, $YELLOWGREEN, $GREEN, $BLUEGREEN, $BLUE,$PURPLE, $BLACK`.

The `transient` and `permanent` declarations tell the animator whether the coordinates have to be evaluated at every time or if they are fixed for all time. The default when the file is loaded is `transient.` Thus, these are just toggles between the two different types of objects.

The number following the `speed` declaration must be a nonnegative integer. It tells the animator how many milliseconds to wait between pictures.

The `dimension` command requires 4 numbers following it. They are the coordinates of the lower left corner and the upper right. The defaults are (0,0) and (1,1).

The `settext` command tells the animator what size and color to make the next text output. The size must be an integer, { `0,1,2,3,4` } with 0 the smallest and 4 the biggest. The font is either `roman` or `symbol.` The color must be a named color and not one that is evaluated.

Most of the rest of the commands put something on the screen.

- `line x1;y1;x2;y2;color;thickness` draws a line from `(x1,y1)` to `(x2,y2)` in user coordinates. These four numbers can be any expression that involves variables and fixed variables from your simulation. They are evaluated at each time step (unless the line is `permanent`) and this is scaled to be drawn in the window. The `color` is optional and can either be a named color or an expression that is to be evaluated. The `thickness` is also optional but if you want to include this, you must include the `color` as well. `thickness` is any nonnegative integer and will result in a thicker line.

- `rline x1;y1;color;thickness` is similar to the `line` command, but a line is drawn from the endpoints of the last line drawn to `(xold+x1,yold+y1)` which becomes then new last point. All other options are the same. This is thus a "relative" line.

- `rect x1;y1;x2;y2;color;thickness` draws a rectangle with lower corner `(x1,y1)` to upper corner `(x2,y2)` with optional color and thickness.

- `frect x1;y1;x2;y2;color` draws a filled rectangle with lower corner `(x1,y1)` to upper corner `(x2,y2)` with optional color.
- `circ x1;y1;rad;color;thick` draws a circle with radius `rad` centered at `(x1,y1)` with optional color and thickness.
- `fcirc x1;y1;rad;color` draws a filled circle with radius `rad` centered at `(x1,y1)` with optional color.
- `ellip x1;y1;rx;ry;color` draws an ellipse with radii `rx,ry` centered at `(x1,y1)` with optional color and thickness.
- `fellip x1;y1;rx;ry;color` draws a filled ellipse with radii `rx,ry` centered at `(x1,y1)` with optional color.
- `comet x1;y1;type;n;color` keeps a history of the last `n` points drawn and renders them in the optional `color`. If `type` is non-negative, then the last n points are drawn as a line with thickness in pixels of the magnitude of `type`. If `type` is negative, filled circles are drawn with a radius of `-thick` in pixels.
- `text x1;y1;s` draws a string `s` at position `(x1,y1)` with the current color and text properties. Only the coordinates can depend on the current values.
- `vtext x1;y1;s;z` draws a string `s` followed by the floating point value `z` at position `(x1,y1)` with the current color and text properties. Thus, you can print out the current time or value of a variable at any given time.
- `xnull x1;y1;x2;y2;color;id` uses the nullclines that you have already computed in your animation. You can use the static nullclines by just choosing `-1` for the `id` parameter. To use dynamic nullclines, you must compute a range of nullclines using the `Nullcline Freeze Range` command. The parameter `id` runs from 0 to N where N is the number of nullclines that you have computed in the range dialog. The animator converts `id` to an integer and tests whether it is in the range and then loads the appropriate nullcline. The `ynull` command is identical. The parameters `x1,y1,x2,y2` tell the animator the window in which the nullclines are defined. These should be the lower-left and upper right corners of the phaseplane where the nullclines were computed.
- `grab x;y;d` allows you to manipulate an object on the screen. Typically the object will be determined by `x,y`. The parameter `d` draws an X at the position `x,y` of diameter `d` so when you invoke the Grab button in the animation window you will see the X and use the mouse to click on the object. The first line that follows the `grab` command tells the animator what to change while you are dragging the mouse (with the button down). You have access to the mouse coordinates in the frame of the animation window. The second line tells you what to do when the mouse is released. You have access to the mouse velocity as well. You can set `runnow` equal to 1 if you want to run the ODE as soon as you release the mouse. Set `runnow=0` to not run immediately. As examples of `grab`, see the odel files, `pendx.ode,idoublepend.ode` and the corresponding animation files `pendx.ani,idoublepend.ani`

Animation controls

When you bring up the animation window `Viewaxes Toon`, a new window appears. The first thing you should do is load an animation file. (You can load the file at the same time as you load the ODE file using the `-anifile filename.ani` command line argument.) Use the `File` button. Don't worry if the animation file you make has an error. Generally, you can load a file over and over agan to experiment and build it up.

In the upper right is a checkbox. If you check this, then whenever you run a simulation, the animation will run simultaneously. It slows the integration down, so you probably don't want to do this all the time.

There is a slider as well to let you move the animation back and forth.

Assuming you have run the ODE and loaded an animation, the `Go` button will start the animation. The `Fast,Slow` buttons speed up or slow down the animation. The `Pause` stops it in which case the `Go` picks up where you paused. `Reset` moves to the start of the simulation. The `>>>>,<<<<` step forward/back one frame. The `Close` button kills the animation window. The `Skip` button lets you skip frames (useful to make smaller animation files).

The `MPEG` button lets you make a series of PPM files that you could then sew together to make a movie of the animation. More usefully, by choosing the animated gif option, you can make a movie directly. Once you have clicked this option, `Reset` and then `Run` the animation. When completed, there will be a file called `anim.gif`. If you want to make more animations, rename it.

The `Grab` button allows you to interact with the animation, if you have added the appropriate code to the `.ani` file. When you click `Grab`, all the grabble objects show up with a cross on them. Use you mouse to click and hold one of them and move it; the object will move as you move the mouse. When you release it, depending on the code in the file, the ODE will run using, possibly, the position and the velocity of the mouse motion to determine initial data.

Array plot

The array plot is a way to view many variables as once as they vary in time; it is useful for PDEs etc. There are two ways to get the array plot: `Viewaxes Array` from the main menu; or, from the `Initial Data` window. The latter is the easiest. Suppose that you have variables named `u0,u1,...,u199`. In the `Init data` window, each variable has a little box next to it. If you wanted to plot all of them as a function of time, click next to `u0`, scroll all the way down and click next to `u199`; then click the `arry` button. This will bring up the array plot. Time runs down, color codes the magnitude, and the variables run across. To edit the ranges, etc, click on the `Edit` button. This will bring up a dialog box. If you invoke the array plot via the `Viewaxes Array` method, you will get the same dialog box. The entries in the dialog box tell you how to set up the plot: `Column 1` is the first variable; `Ncols` is the total number of columns (`Colskip` lets you plot every other one or every third etc if your ODE is organized as, say `u0,v0,w0,u1,v1,w1,...` in which case, you'd set the number of columns to 600 and the skip to 3). `Nrows` refers to the number of rows you will plot and `Rowskip` lets you skip over them. `Zmin,Zmax` define the color scale range; the maximum and minimum of the variables being plotted. `Autoplot` will redraw automatically if needed. The other buttons are: `Redraw` (obvious), `Close` (obvious); `GIF` will make a gif of the current view; `Print` lets you create a postscript file of the current plot (you tell it the labels and the file name; Render determines the color code which I forget); `Fit` will automatically adjust the `Zmax,Zmin`. `Range` allows you to range through parameters or intial conditions to produce either a series of still gifs or a single animated gif file. You can choose the name, whether stills (`Still=1`) or movie (`Still=0`) as well as whether you want the parameter written on the image (`Tag=1`). The `Range integrate` dialog comes up and you can fill that in as usual.