Chapter 2
A Very Brief Tour of *XPPAUT*

Touring can make you crazy.
—Frank Zappa

In this chapter, I will show you the way to get up and running. Many of the types of problems most people need to solve can be done after going through this short session with the program. I will assume that you are at least somewhat familiar with differential equations.

I will run through two quick tutorials on using *XPPAUT* for a linear differential equation and a nonlinear differential equation. These should be sufficient for anyone to set up differential equations and solve them. For more advanced topics, the user should study the rest of the book.

Note. In the following tutorials, menu commands appear in computer modern font like this: `Command`. Single letter keyboard shortcuts will appear in bold roman like this: **A**. *Do not use the CapsLock key*; all shortcuts are lowercase. Every command can be accessed by a series of keystrokes. To make sure key clicks are interpreted correctly, click on the title bar of the window for which the shortcut is intended.

2.1 Creating the ODE file

Consider the simple linear differential equation,

$$\frac{dx}{dt} = ax + by,$$

$$\frac{dy}{dt} = cx + dy, \tag{2.1}$$

where a, b, c, d are parameters. We will explore the behavior of this two-dimensional system using *XPPAUT* (even though it is easy to obtain a closed form solution). To analyze a differential equation using *XPPAUT*, you must create an input file that tells the program the names of the variables, parameters, and equations. By convention, these files have the file extension ode and I will call them ODE (ordinary differential equation) files. Here is an ODE file for (2.1):

```
# linear2d.ode
#
# right hand sides
x'=a*x+b*y
y'=c*x+d*y
#
# parameters
par a=0,b=1,c=-1,d=0
#
# some initial conditions
init x=1,y=0
#
# we are done
done
```

I have included some comments denoted by lines starting with #; these are not necessary but can make the file easier to understand. The rest of the file is fairly straightforward (I hope). The values given to the parameters are optional; by default they are set to zero. The `init` statement is also optional. The minimal file for this system has four lines:

```
x'=a*x+b*y
y'=c*x+d*y
par a,b,c,d
done
```

Of course, in this minimal file all parameters are set to zero, as are the initial conditions. Use a text editor to type in the first file name exactly as it is shown, or download the files. Name the file `linear2d.ode` and save it. That's it—you have written an ODE file. I give a number of practice examples at the end of this section. The minimal steps are as follows:

- Use an editor to open up a text file.
- Write the differential equations in the file; one per line.
- Use the `par` statement to declare all the parameters in your system. Optionally define initial conditions with the `init` statement.
- End the file with the statement `done`.
- Save and close the file.

Note. The equation reader is case insensitive so that `AbC` and `abC` are treated identically.

WARNING: In statements declaring initial conditions and parameters, **do not ever** put spaces between the variable and the "=" sign and the number. *XPPAUT* uses spaces as a delimiter. Always write `a=2.5` and **never** write `a = 2.5`.

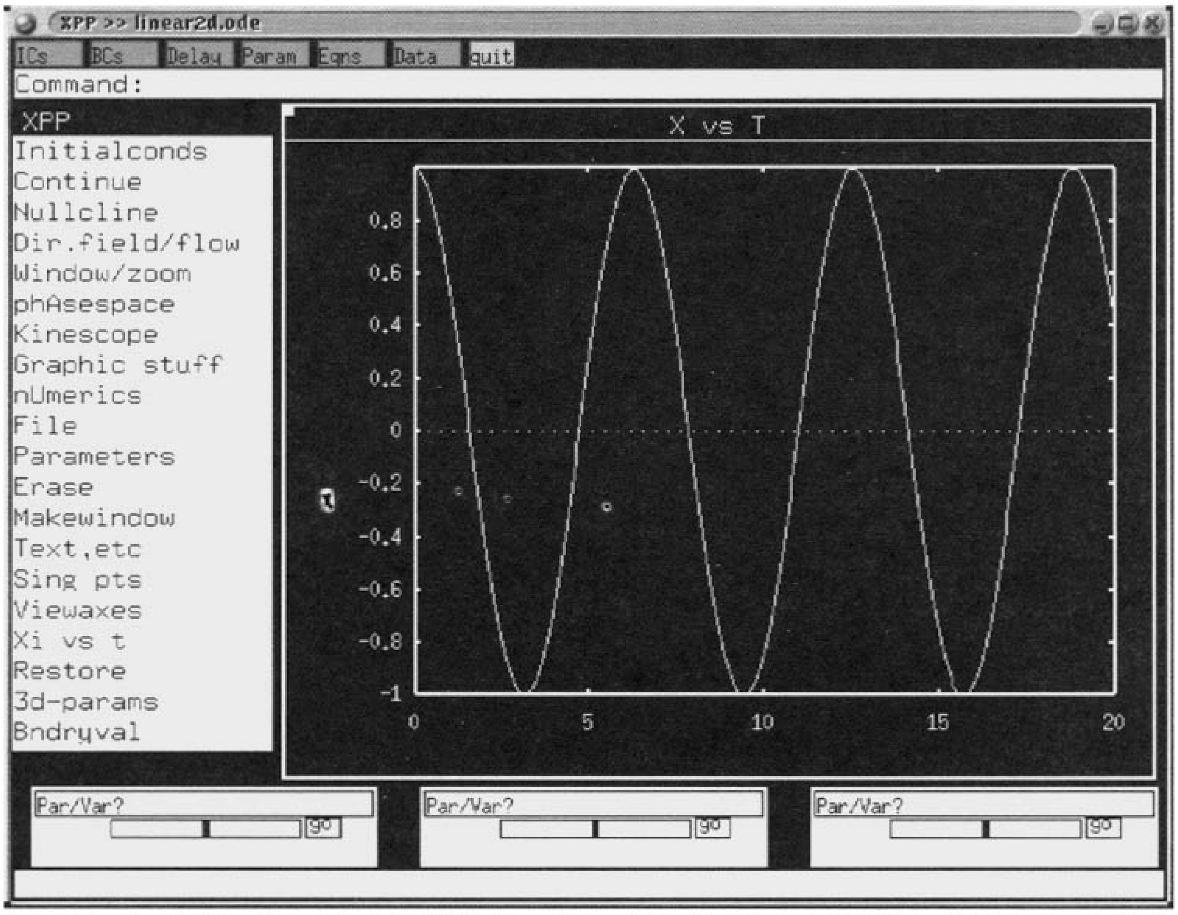


Figure 2.1. *The main XPPAUT window.*

2.2 Running the program

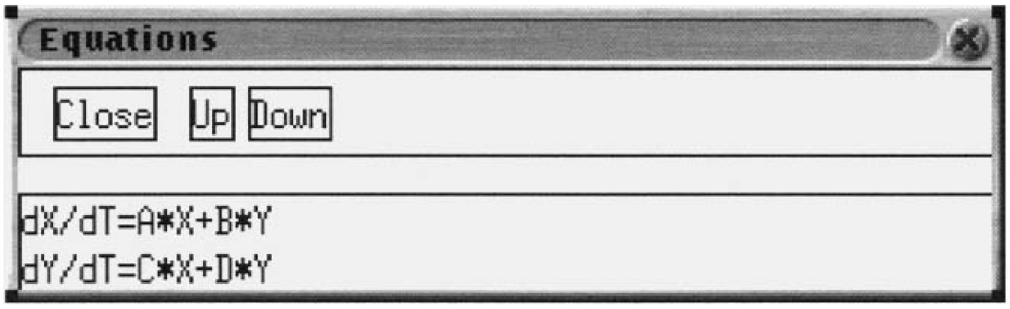
Run *XPPAUT* by typing

```
xpp linear2d.ode
```

Replace xpp with whatever you have decided to call the executable file, including all the desired command line options. (*If you are using winpp, click on the* **winpp** *icon; then choose the file from the file selection dialog box.*) A single window will appear, unless you start *XPPAUT* with all the windows visible (using the command `xppaut -allwin`; see Chapter 1).

2.2.1 The main window

The **Main Window** contains a large region for graphics, menus, and various other gadgets. It is illustrated in Figure 2.1. Commands are given either by clicking on the menu items in the left column with the mouse or tapping keyboard shortcuts. After a while, as you become more familiar with *XPPAUT*, you will use the keyboard shortcuts more often. I will tell you the full commands and the keyboard shortcuts. In general, the keyboard shortcut is the first letter of the command, unless there is ambiguity (such as between `Nullcline` and



```
dX/dT=A*X+B*Y
dY/dT=C*X+D*Y
```

Figure 2.2. *The equation window.*

nUmerics), in which case the shortcut is just the capitalized letter (**N** and **U**, respectively). Unlike Windows keyboard shortcuts, the letter key alone is sufficient and it is not necessary to press the Alt and Ctrl keys at the same time. The top region of the **Main Window** is for typed input such as parameter values. The bottom of the **Main Window** displays information about various things as well as a short description of the highlighted menu item. The three little boxes labeled parameter are sliders to let you change parameters and initial data. Across the top of the **Main Window** you will see a menu bar of turquoise buttons. Clicking on these opens up a variety of windows. In addition, you may or may not have several other yellow buttons which are shortcuts to commands. These can be defined by the user in the ODE file as options (see Appendix B) or in the *XPPAUT* resource file (see Chapter 1).

Click on the button labeled Eqns and the **Equation Window** will appear as shown in Figure 2.2. This allows you to see the differential equations that you are solving. We will describe the other windows as the tutorial progresses. Clicking on the Close button closes this window.

2.2.2 Quitting the program

To exit *XPPAUT*, click File Quit Yes (**F Q Y**).

2.3 Solving the equation, graphing, and plotting

Here, we will solve the ODE, use the mouse to select different initial conditions, save plots of various types, and create files for printing.

Computing the solution

In the **Main Window**, you should see a box with axes numbers. The title in the window should say X vs T which tells you that the variable X is along the vertical axis and T along the horizontal. The plotting range is from 0 to 20 along the horizontal and -1 to 1 along the vertical axis. When a solution is computed, this view will be shown. Click on Initial Conds Go (**I G**) in the **Main Window**. A solution will be drawn followed by a beep. As one would expect, given the differential equations the solution looks like a few cycles of a cosine wave.

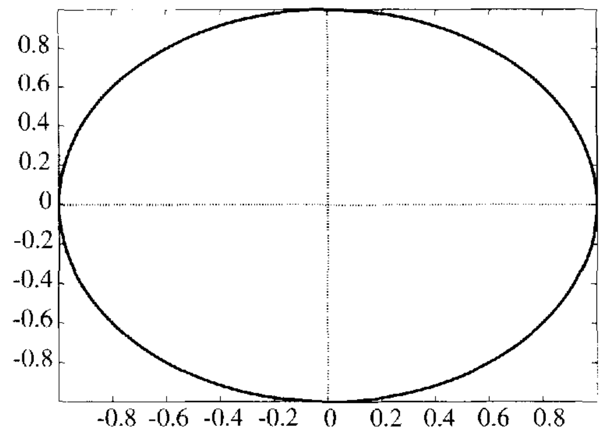


Figure 2.3. *Phase-plane for the linear two-dimensional problem.*

Changing the view

To plot Y versus T instead of X versus T, just click on the command Xi vs t (**X**) and choose Y by backspacing over X, typing in Y and clicking **Enter**.

Often, you may want to plot a phase-plane instead, that is, X versus Y. To do this, click on Viewaxes 2D (**V 2**) and a dialog box will appear. Fill it in as follows:

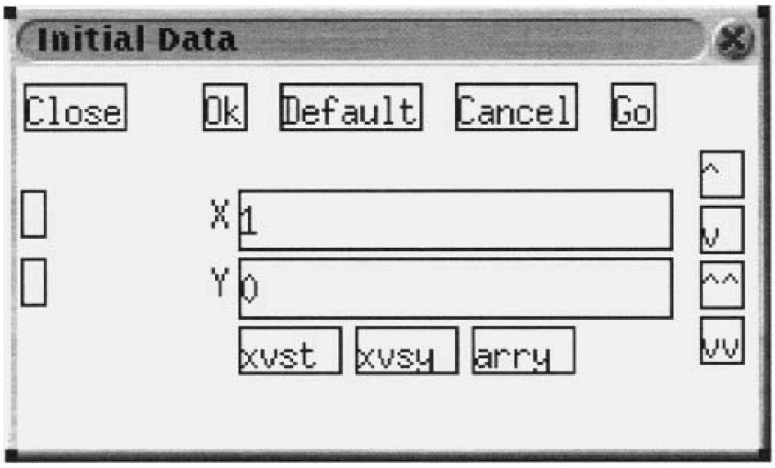
X-axis: X	Xmax: 1
Y-axis: Y	Ymax: 1
Xmin: -1	Xlabel:
Ymin: -1	Ylabel:

Click on OK when you are done. (Note that you could have filled in the labels if you had wanted, but for now, there is no reason to.) You should see a nice elliptical orbit in the window. This is the solution in the phase-plane (cf. Figure 2.3).

Shortcuts

Click on the button ICs in the **Main Window** to bring up the **Initial Data Window**. This window shows the current initial data. There is a very simple way to view the phase-plane or view variables versus time. Look at the **Initial Data Window** (Figure 2.4). You will see that there are little boxes next to the variable names. Check the two boxes next to X and Y. Then at the bottom of the **Initial Data Window**, click the XvsY button. This will plot a phase-plane and automatically size the window to contain the entire trajectory. This is a shortcut and does not give you the control that the menu command does. (For example, the window is always sized for the trajectory, and no labels are added or changed. Nor can you plot auxiliary quantities with this shortcut.) To view one or more variables against time, just check the variables you want to plot (up to 10) and click on the XvsT button in the **Initial Data Window**.

You should have a phase-plane picture in the window. (If not, get one by following the above instructions or using the shortcut.) Click on Initial Conds Mouse (**I M**). Use the mouse to click somewhere in the window. You should see a new trajectory drawn. This,



X: 1

Y: 0

Figure 2.4. *The initial conditions window.*

too, is an ellipse. Repeat this again to draw another trajectory. If you get tired of repeating this, try `Initial Conds mIce` (**I I**) which, being "mice" is many mouses. Keep clicking in the window. When you are bored with this, click either outside the window or press the escape key, `Esc`.

Click on `Erase` and then `Restore` (**E R**). Note that all the trajectories are gone except the latest one. *XPPAUT* only stores the latest one. There is a way to store many of them, but we will not explore that for now.

Printing the picture

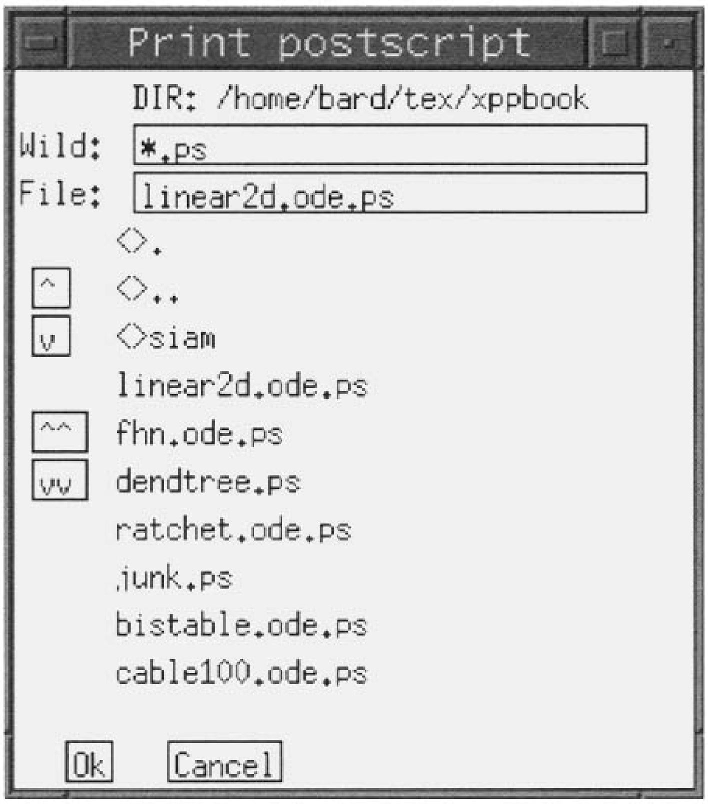
XPPAUT does not directly send a picture to your printer. Rather, it creates a PostScript® file which you can send to your printer. If you don't have PostScript capabilities, then you probably will have to use the alternate method of getting hardcopy. (Note that Microsoft® Word supports the import of PostScript and Encapsulated PostScript, but can only print such pictures to a PostScript printer. You can download a rather large program for Windows called GhostView which enables you to view and print PostScript on printers that are not PostScript compatible. Linux and other UNIX distributions usually have a PostScript viewer included.)

Here is how to make a PostScript file. Click on `Graphics Postscript` (**G P**). A dialog box will appear asking for three choices: (i) black and white or color; (ii) landscape or portrait; (iii) font size for the axes. Accept all the defaults for now by just clicking `Ok`. You will then be asked for a filename. The File Selector box is shown in Figure 2.5. You can move up or down directory trees by clicking on the <>; choose files by clicking on them; scroll up or down by clicking on the up/down arrows on the left or using the arrow keys and the PageUp/PageDown keys on the keyboard; change the wild card; or type in a filename. For now, you can just click on `Ok` and a PostScript plot will be created and saved. The file will be called `linear2d.ode.ps` but you can rename it anything you want.

Once you have the PostScript file, you can type

```
lpr filename
```

on UNIX. If you are in Windows and your computer is connected to a PostScript printer,



```
Print postscript
        DIR: /home/bard/tex/xppbook
Wild:   *.ps
File:   linear2d.ode.ps
        <>.
        <>..
        <>siam
        linear2d.ode.ps
        fhn.ode.ps
        dendtree.ps
        ratchet.ode.ps
        junk.ps
        bistable.ode.ps
        cable100.ode.ps
```

Figure 2.5. *File selector.*

then this should work:

```
copy filename lpt1:
```

If you are in Windows and you don't have a PostScript printer, it is still easy to print the PostScript files to your printer. However, you have to download a program called GhostView. This allows you to print PostScript files from printers that are not PostScript compatible. Most Windows distributions of GhostView come with a command line utility called `gsprint.exe` in the `\Ghostgum\gsview` directory. Either of the following two lines have worked for me:

```
gsprint myfile.ps
gsprint -colour myfile.ps
```

I use the latter for color printouts.

Other ways to get hardcopy

Another way to get hardcopy which you can import into documents is to grab the image from the screen. In Windows, click on `Alt+PrtSc` after making the desired window active. In some servers, such as the MiX server, this will grab the entire X desktop window, which you can paste this into the Paint accessory and then use the tools in Paint to cut out what you want. Alternatively, you can download a number of programs that let you capture areas of the screen. In the UNIX environment, you can capture a window using `xv`, an excellent utility that is free and available for most UNIX versions. (All of the screenshots in this tutorial were captured with `xv`.) Finally, you can capture the screen (or a series of screen images) with the `Kinescope Capture` command and then write these to disk with the `Kinescope Save` command. This produces a gif file that is usable by many software packages including most browsers.

Getting a good window

If you have computed a solution and don't have a clue about the bounds of the graph, let *XPPAUT* do all the work. Click on `Window/zoom (F)it` and the window will be resized to a perfect fit. The shortcut is **W F** and most likely you will use it a lot!

2.4 Changing parameters and initial data

There are many ways to vary the parameters and initial conditions in *XPPAUT*. We have already seen how to change the initial data using the mouse. This method works for any n-dimensional system as long as the current view is a phase-plane of two variables. Here are three other ways to change the initial data:

(1) From the main menu, click on `Initial Conds New` and manually insert your data at the prompts. You will be prompted for each variable in order. (For systems with hundreds of variables, this is not a very good way to change the data!) However, if you get tired, just press `Esc` and the remaining variables will assume their current initial conditions.

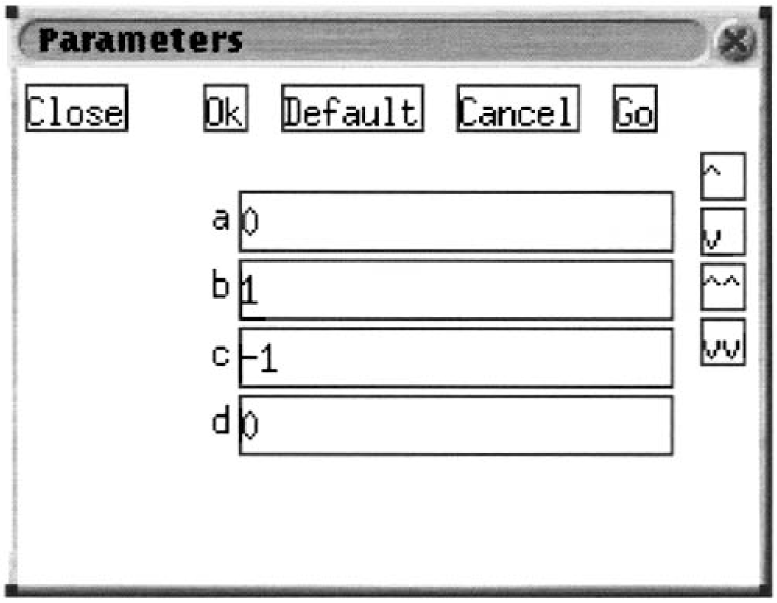
(2) In the **Initial Data Window**, you can edit the particular variable you want to change. Just click in the window next to the variable and edit the value. Then click on the `Go` button in the **Initial Data Window**. If there are many variables, you can use the little scroll buttons on the right to go up and down a line or a page at a time. If you click the mouse in the text entry region for a variable, you can use the `PageUp`, `PageDown`, `Home`, and `End` keys to move around. Clicking `Enter` moves to the next text window. The `Default` button returns the initial data to those with which the program started. If you don't want to run the simulation, but have set the initial data, *you must* click on the `Ok` button in the **Initial Data Window** for the new initial data to be recognized.

(3) Attach the variable to a slider. Sliders are described next.

There are many ways to change parameters as well. Here are three of them:

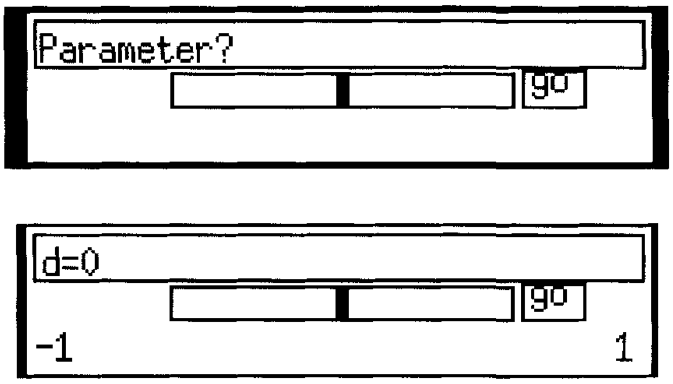
(1) From the **Main Window**, click on `Parameters`. In the command line of the **Main Window**, you will be prompted for a parameter name. Type in the name of a parameter that you want to change. Click on `Enter` to change the value and `Enter` again to change another parameter. Click on `Enter` a few times to get rid of the prompt.

(2) Bring up the **Parameter Window** by clicking on the turquoise `Param` button at the top of the **Main Window**. In the **Parameter Window** (shown in Figure 2.6), type in values next to the parameter you want to change. Use the scroll buttons or the keyboard to scroll around. As in the **Initial Data Window**, there are four buttons



a 0
b 1
c -1
d 0

Figure 2.6. *The parameter window.*



Parameter?

d=0

-1 1

Figure 2.7. *Top: Unused parameter slider. Bottom: Used parameter slider.*

across the top. Click on Go to keep the values and run the simulation; click on Ok to keep the parameters without running the simulation. Click on Cancel to return to the values since you last pressed Go or Ok. The Default button returns the parameters to the values with which you started the program.

(3) Use the little sliders (Figure 2.7). We will attach the parameter d to one of the sliders. Click on one of the unused parameter sliders. Fill in the dialog box as follows:

Parameter: d
Value: 0
Low: -1
High: 1

and click Ok. You have assigned the parameter d to one of the sliders and allowed it to range between -1 and 1. Grab the little slider with the mouse and move it around. Watch how d changes. Now click on the tiny go button in the slider. The equations will be integrated. Move the slider some more and click on the go button to get another solution. The sliders can also be attached to initial data. Just choose a variable instead of a parameter.

2.5 Looking at the numbers: The data viewer

In addition to the graphs that *XPPAUT* produces, it also gives you access to the actual numerical values from the simulation. You can bring up the **Data Viewer** by clicking on the turquoise button labeled Data at the top of the **Main Window**. The **Data Viewer** shown in Figure 2.8 has many buttons, some of which we will use later. The main use of the **Data Viewer** is to look at the actual numbers from a simulation. The independent variable occupies the left-most column and the dependent variables fill in the remaining windows. Click on the top of the **Data Viewer** to make it the active window. The arrow keys and the PageUp, PageDown, Home, and End keys (as well as their corresponding buttons) do all the obvious things. Left and right arrow keys scroll horizontally—a useful feature if you have many variables. I mention the following three particularly useful buttons:

Find brings up a dialog box prompting you for the name of a column and a value. If you click on Ok, *XPPAUT* will find the entry that is closest and bring that row to the top. For example, you can find the maximum and minimum of a variable by choosing a really big number and a really small number in the Find dialog box.

Get loads the top line of the **Data Viewer** as initial data.

Write writes the entire contents of the browser to a text file that you specify.

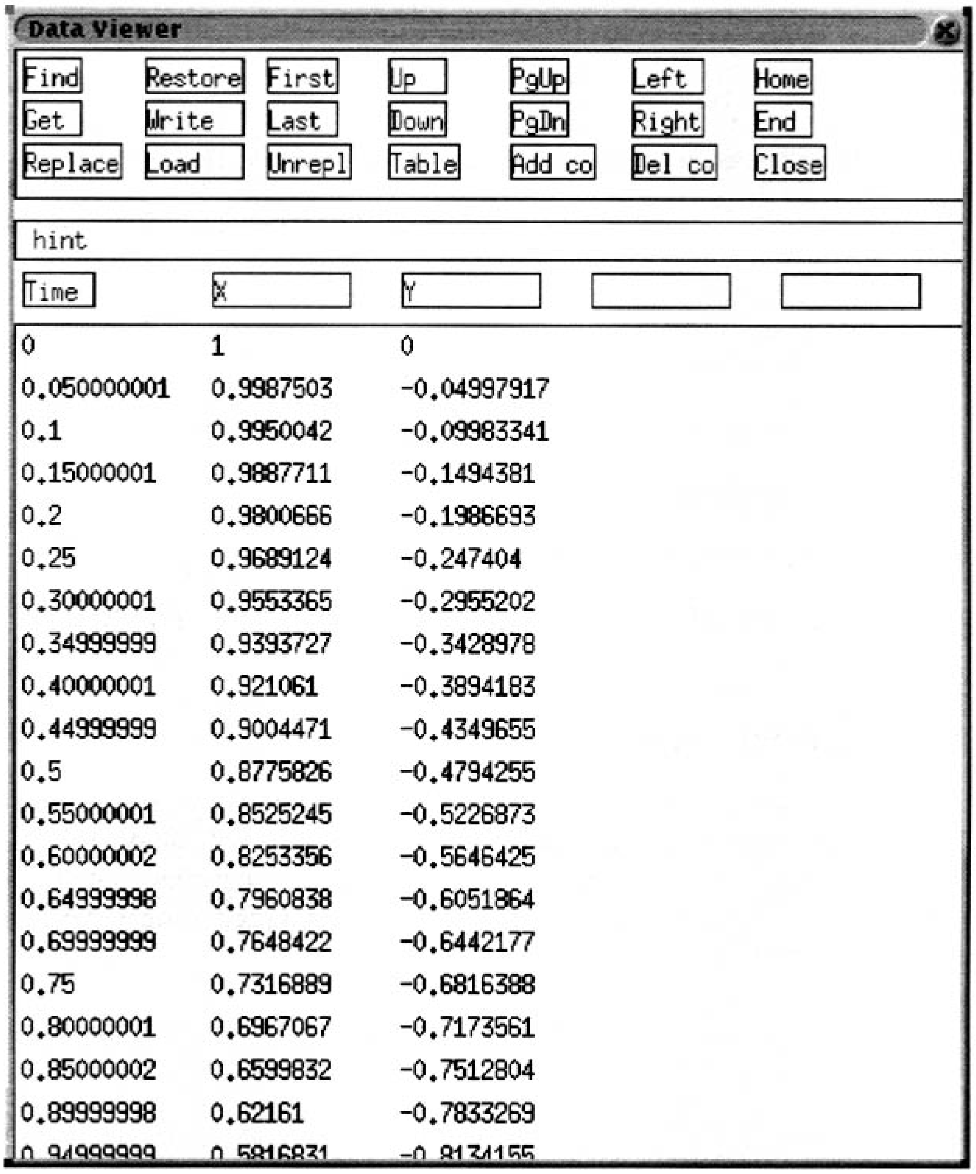
2.6 Saving and restoring the state of *XPPAUT*

Often you will have a view, a set of parameters, and initial data that you want to keep. You can save the current state of *XPPAUT* by clicking on File Write set (**F W**) in the **Main Window**. This bring up a file selection box. Type in a filename—the default extension is .set. The resulting file is an ASCII file that is human and computer readable. The first and last few lines will look like this:

```
## Set file for linear2d.ode on Fri Aug  4 13:53:31 2000
2   Number of equations and auxiliaries
4   Number of parameters
# Numerical stuff
1    nout
40    nullcline mesh


.......

RHS etc ...
dX/dT=A*X+B*Y
dY/dT=C*X+D*Y
```



Time	X	Y
0	1	0
0.050000001	0.9987503	-0.04997917
0.1	0.9950042	-0.09983341
0.15000001	0.9887711	-0.1494381
0.2	0.9800666	-0.1986693
0.25	0.9689124	-0.247404
0.30000001	0.9553365	-0.2955202
0.34999999	0.9393727	-0.3428978
0.40000001	0.921061	-0.3894183
0.44999999	0.9004471	-0.4349655
0.5	0.8775826	-0.4794255
0.55000001	0.8525245	-0.5226873
0.60000002	0.8253356	-0.5646425
0.64999998	0.7960838	-0.6051864
0.69999999	0.7648422	-0.6442177
0.75	0.7316889	-0.6816388
0.80000001	0.6967067	-0.7173561
0.85000002	0.6599832	-0.7512804
0.89999998	0.62161	-0.7833269
0.94999999	0.5816831	-0.8134155

Figure 2.8. *The data viewer.*

Once you quit *XPPAUT*, you can start it up again and then use the `File Read set` to load the parameters and data that you saved.

If you haven't already done so, quit *XPPAUT*. To quit, click on `File Quit Yes` (**F Q Y**).

2.6.1 Command summary

`Initialconds Go` computes a trajectory with the initial conditions specified in the **Initial Data Window**. (**I G**)

`Initialconds Mouse` computes a trajectory with the initial conditions specified by the mouse. `Initialconds m(I)ce` lets you specify many initial conditions. (**I M** or **I I**)

`Erase` erases the screen. (**E**)

`Restore` redraws the screen. (**R**)

`Viewaxes 2D` lets you define a new two-dimensional view. (**V 2**)

`Graphic stuff Postscript` allows you to create a PostScript file of the current graphics. (**G P**)

`KinescopeCapture` allows you to capture the current view into memory and `KinescopeSave` writes the current view to disk.

`Window/zoom (F)it` fits the window to include the entire solution. (**W F**)

`File Quit` exits the program. (**F Q**)

`File Write set` saves the state of *XPPAUT*. (**F R**)

`File Read set` restores the state of *XPPAUT* from a saved `.set` file. (**F R**)

2.7 A nonlinear equation

We will look at a nonlinear equation next, find fixed points, and draw some nullclines and direction fields. Here we want to solve a nonlinear equation. We will choose a planar system since there are many nice tools available for analyzing two-dimensional systems. A classic model is the Fitzhugh–Nagumo equation which is used as a model for nerve conduction. The equations are as follows:

$$\frac{dV}{dt} = I + V(1-V)(V-a) - w, \tag{2.2}$$

$$\frac{dw}{dt} = \epsilon(V - \gamma w)$$

with parameters I, a, ϵ, γ. Typical values are $a = .1$, $I = 0$, $\epsilon = .1$, and $\gamma = 0.25$. Let's write the following ODE file for this:

```
# Fitzhugh-Nagumo equations
v'=I+v*(1-v)*(v-a) -w
w'=eps*(v-gamma*w)
par I=0,a=.1,eps=.1,gamma=.25
@ xp=V,yp=w,xlo=-.25,xhi=1.25,ylo=-.5,yhi=1,total=100
@ maxstor=10000
done
```

We have already seen the first four lines: (i) lines beginning with a # are comments; (ii) the next two lines `v'` and `w'` define the differential equations; and (iii) the line beginning with `par` defines the parameters and their default values. The fifth line beginning with the @ sign is a directive to set some of the options in *XPPAUT*. These could all be done within the program, but this way everything is all set up for you. Details of these options are found

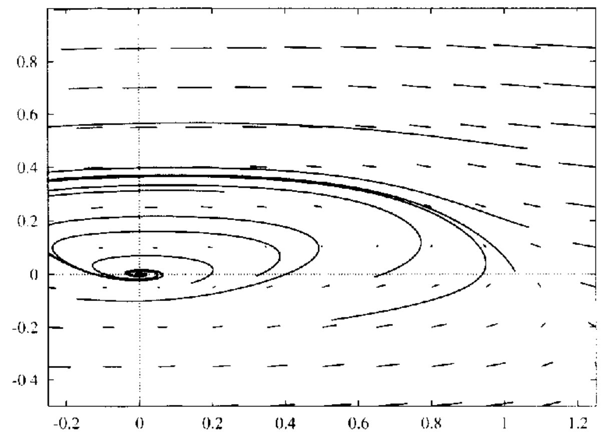


Figure 2.9. *Direction fields and some trajectories for the Fitzhugh–Nagumo equations.*

in Appendix B. (For the curious, these options set the x-axis (xp) to be the V variable, the y-axis (yp) to be the w variable, the plot range to be $[-.25, 1.25] \times [-.5, 1]$, and the total amount of integration time to be 100.) The last option @ `maxstor=10000` is very useful. *XPPAUT* allocates enough storage to keep 4000 time points. You can make it allocate as much as you want with this option. Here I have told *XPPAUT* to allocate storage for 10,000 points. Type in this ODE file (or download it; all of the files are available for downloading at the *XPPAUT* home page) and save it as `fhn.ode`.

2.7.1 Direction fields

Run the new ODE file by typing `xpp fhn.ode`. The usual window will pop up. One of the standard ways to analyze differential equations in the plane is to sketch the *direction fields*. Suppose that the differential equation is

$$x' = f(x, y), \qquad y' = g(x, y).$$

The phase-plane is divided into a grid and at each point (x, y) in the grid, a vector is drawn with (x, y) as the base and $(x + sf(x, y), y + sg(x, y))$ as the terminal point, where s is a scaling factor. This so-called direction field gives you a hint about how trajectories move around in the plane. *XPPAUT* lets you quickly draw the direction field of a system. Click on `Dir.field/flow (D)irect Field` (**D D**) and then accept the default of 10 for the grid size by clicking `Enter`. A bunch of vectors will be drawn on the screen, mainly horizontal. They are horizontal because ϵ is small, which implies that there is little change in the w variable. The length of the vectors is proportional to the magnitude of the flow at each point. At the head of each vector is a little bead. If you want to have scaled direction fields which don't take into account the magnitude of the vector field, just click on `Dir.field (S)caled Dir.Fld` (**D S**) and use the default grid size. (I prefer pure direction fields, but this is a matter of taste.) Click on `Initialconds m(I)ce` to try several different trajectories. Note how the vectors from the direction field are tangent to the trajectories. See Figure 2.9.

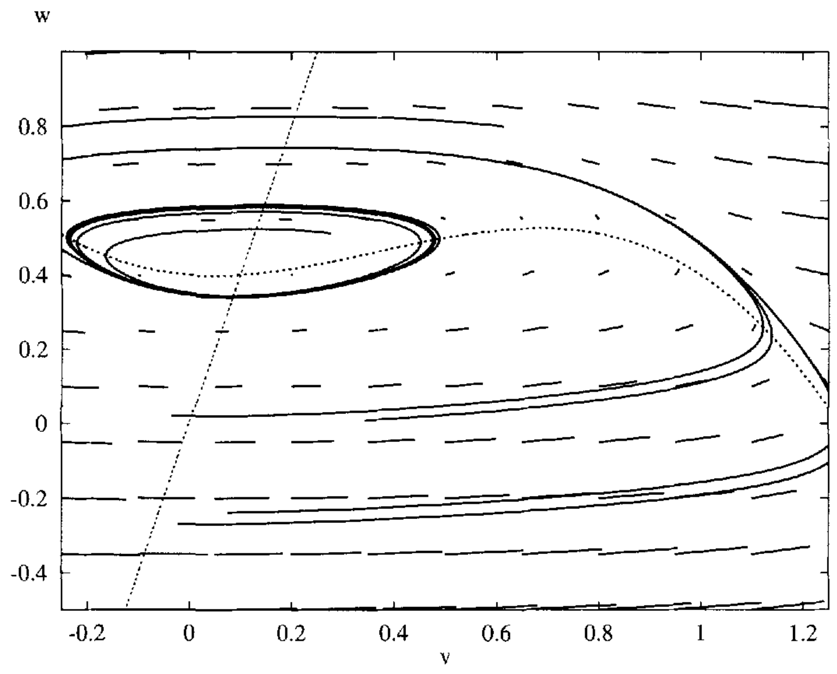


Figure 2.10. *Nullclines, direction fields, and trajectories for $I = 0.4$ in the Fitzhugh–Nagumo equations.*

2.7.2 Nullclines and fixed points

A powerful technique for the analysis of planar differential equations, which is related to the calculation of direction fields, is the use of *nullclines*. Nullclines are curves in the plane along which the rate of change of a particular variable is zero. The x-nullcline is the curve where $dx/dt = 0$, that is, $f(x, y) = 0$. Similarly, the y-nullcline is the curve where $g(x, y) = 0$. The usefulness of these curves is that they break up the plane into regions along which the derivatives of each variable have a constant sign. Thus, the general direction of the flow is easy to determine. Furthermore, any place that the nullclines intersect is a fixed point of the differential equation.

XPPAUT can compute the nullclines for planar systems. To do this, just click on `Nullcline New` (**N N**). You should see two curves appear; a red one representing the v-nullcline and a green one representing the w-nullcline. The green one is a straight line and the red is a cubic. They intersect just once: there is a single fixed point. Move the mouse into the phase-plane area and hold it down as you move it. At the bottom of the **Main Window** the x and y coordinates will appear as you move the mouse. The intersection of the nullclines will appear to be at (0,0). Figure 2.10 shows a printout along with a representative trajectory when the parameter $I = 0.3$.

The stability of fixed points is determined by linearizing about them and finding the eigenvalues of the resulting linear matrix. *XPPAUT* will determine this for you quite easily. *XPPAUT* uses Newton's method to find the fixed points and then numerically linearizes about them to determine stability. To use Newton's method, a decent guess first needs to be provided. For planar systems, this is easy to do—just guess the intersection of the nullclines. In *XPPAUT*, fixed points and their stability are found using the `Sing pts` command, as "singular points" is a term sometimes used for fixed points or equilibrium points. Click on

Sing pts Mouse (**S M**) and move the mouse close to the intersection of the nullclines. Click either mouse button and a message box will appear on the screen. Click on No since we don't need the eigenvalues. A new window will appear that contains information about the fixed points. The stability is shown at the top of the window. The nature of the eigenvalues is as follows: **c+** denotes the number of complex eigenvalues with positive real part; **c–** is the number of complex eigenvalues with negative real part; **im** is the number of purely imaginary eigenvalues; **r+** is the number of positive real eigenvalues; and **r–** is the number of negative real eigenvalues. Recall that a fixed point is linearly stable if all the eigenvalues have negative real parts. Finally, the value of the fixed points is shown under the line. As can be seen from this example, there are two complex eigenvalues with negative real parts: the fixed point is (0,0). (*XPPAUT* reports a very small nonzero fixed point due to numerical error.) Integrate the system using the mouse, starting with initial conditions near the fixed point. (In the **Main Window**, type **I I**.) Note how solutions spiral into the origin, as is expected when there are complex eigenvalues with negative real parts.

For nonplanar systems of differential equations, you must provide a direct guess. Bring up the **Initial Data Window**, type in your guess, and click on Ok in the **Initial Data Window**. Then from the **Main Window**, click on Sing Pts Go (**S G**).

Another way to get fixed points is to use the Sing Pts monte(C)ar (**S C**) command, which randomly picks starting guesses in a range and then tries to find them with Newton's method. You have to provide the ranges for starting guesses as well as the number of guesses. The fixed points found are listed in the **Data Viewer**. You should use this method only if you have no clue where the fixed points lie.

Bring up the **Parameter Window**. In this window, change the parameter I from 0 to 0.4 and click on Ok. In the **Main Window**, erase the screen and redraw the nullclines as follows: Erase Nullclines New (**E N N**). The fixed point has moved up. Check its stability by using the mouse (Sing pts Mouse). The fixed point should be (0.1,0.4) and it is unstable since there are two complex eigenvalues with positive real parts (**C+=2**). Use the mouse to choose several initial conditions in the plane. All solutions go to a limit cycle. That is, they converge to a closed curve in the plane representing a stable periodic solution.

Let's make a picture that has the nullclines, the direction fields, and a few representative trajectories. Since *XPPAUT* keeps only the last trajectory computed, we will "freeze" the solutions we compute. You can freeze all trajectories automatically or freeze them one at a time. We will do the former. Click on Graphic stuff (F)reeze (O)n freeze (**G F O**) to permanently save computed curves. Up to 26 can be saved in any window. Now use the mouse to compute several trajectories. Draw the direction fields by clicking Dir.field/flow (D)irect Field (**D D**). Finally, label the axes. Click on Viewaxes 2D (**V 2**) and the two-dimensional view dialog box will appear. Change nothing but the labels (the last two entries), and set V as the **Xlabel** and w as the **Ylabel**. Click on Ok to close the dialog box. Finally, since the axes are confusing in the already busy picture, click on Graphic stuff aXes opts (**G X**), and in the dialog box change the 1's in the entries **X-org(1=on)** and **Y-org(1=on)** to 0's to turn off the plotting of the X and Y axes. Click Ok when you are done. Now create a PostScript file (Graphic stuff (P)ostscript (**G P**)) and accept all the defaults. Name the file whatever you want and click on Ok in the file selection box. Figure 2.10 shows the version that I have made. Yours will be slightly different. If you want to play around some more, turn off the automatic freeze option (`Graphic stuff Freeze Off freeze` (**G F O**)) and delete all the frozen curves (`Graphic stuff Freeze Remove all` (**G F R**)).

2.7.3 Command summary

`Nullcline New` draws nullclines for a planar system. (**N N**)

`Dir.field/flow (D)irect Field` draws direction fields for a planar system. (**D D**)

`Sing pts Mouse` computes fixed points for a system with initial guess specified by the mouse. (**S M**)

`Sing pts Go` computes fixed points for a system with initial guess specified by the current initial conditions. (**S G**)

`Graphic stuff Freeze On freeze` will permanently keep computed trajectories in the current window. (**G F O**)

`Graphic stuff Freeze Off freeze` will toggle off the above option. (**G F O**)

`Graphic stuff Freeze Remove all` deletes all the permanently stored curves. (**G F R**)

`Graphic stuff aXes opts` lets you change the axes. (**G X**)

`Viewaxes 2D` allows you to change the two-dimensional view of the current graphics window and to label the axes. (**V 2**)

2.8 The most important numerical parameters

XPPAUT has many numerical routines built into it and, thus, there are many numerical parameters that you can set. These will be dealt with in subsequent sections of the book where necessary. However, the most common parameters you will want to change are the total amount of time to integrate and the step size for integration. You may also want to change the method of integration from the default fixed step Runge–Kutta algorithm. To alter the numerical parameters, click on `nUmerics` (**U**) which produces a new menu. This is a top-level menu which allows you to change many parameters before you return to the main menu. To return to the main menu, just click on `[Esc]-exit` or press the `Esc` key.

There are many entries on the numerics menu. The following four are the most commonly used:

`Total` sets the total amount of time to integrate the equations. (**T**)

`Dt` sets the size of the time step for the fixed step size integration methods and sets the output times for the adaptive integrators. (**D**)

`Noutput` sets the number of steps to take before *plotting* an output point. Thus, to plot every fourth point, change `Nout` to 4. For the variable step size integrators, this should be set to 1. (**O**)

`Method` sets the integration method. There are currently 15 available. They are described in Appendix C. (**M**)

When you are done setting the numerical parameters, just click on `Esc-exit` or press the `Esc` key.

2.9 Exercises

1. Write a differential equation file for the following damped pendulum:

$$\ddot{x} + a\dot{x} + \sin x = 0.$$

Hint: Rewrite this as a system of two first order equations:

$$x' = v, \qquad v' = -av - \sin x.$$

Look at the phase-plane (plot x along the horizontal and v along the vertical axis) for this in the case of no damping ($a = 0$) and small positive damping. A good window to use is $[-8, 8] \times [-3, 3]$. Draw the direction field and classify all the fixed points with and without damping. (In any message boxes related to the computation of fixed points, answer `No` in every case. If, by mistake, you click on `Yes`, then repeatedly press the `Esc` key.) Draw some representative trajectories with and without damping. Plot your work in the damped and undamped case as a PostScript file.

2. Fire up the Fitzhugh–Nagumo equation. Increase the parameter I from zero and determine the value at which the fixed point becomes unstable. Continue to increase I and figure out the value at which the fixed point again becomes stable. Next, set $I = 0$ and change γ to 10, erase the screen, and recompute the nullclines. How many fixed points are there? Determine their stability. Choose several initial conditions and see if you can detect a difference between the long-time behavior of the trajectory.

3. Write a differential equation file for the Lorenz equations

$$x' = s(-x + y),$$
$$y' = rx - y - xz,$$
$$z' = -bz + xy$$

with initial conditions $x = -7.5$, $y = -3.6$, $z = 30$, and starting parameter values $r = 27$, $s = 10$, $b = 2.66$. (Don't forget to include the multiplication sign * wherever there are products!) Run *XPPAUT* and plot x against time. Then plot x and z together. (**Hint:** click on the variables x and y in the **Initial Data Window** and then click the `xvsy` button in the **Initial Data Window**.) You might want to go into the numerics menu and make `Dt` smaller and the total amount of time longer, e.g., set `Dt=.025` and `Total=40`.

4. Sketch phase-planes for the following systems in the window $[-3, 3] \times [-3, 3]$:

   1. $x' = y - 2x$, $y' = x + y$;
   2. $x' = x$, $y' = x^2 + y^2 - 1$;
   3. $x' = x(1 - x/2 - y)$, $y' = y(x - 1 - y/2)$
   4. $x' = y^2$, $y' = x$;
   5. $x' = 2 - x - y^2$, $-y(x^2 + y^2 - 3x + 1)$.

   (**Hint:** Only make one ODE file. Within *XPPAUT*, click on `File` `Edit` `RHS's` to edit the right-hand sides. Click on `OK` when you are done. Make sure you include the `*` for multiplication!)