

DYNAMICS-WIN:

**a numerical toolkit
for exploring and analyzing the dynamics
of low dimensional systems of
differential and difference equations**

Helena E. Nusse and James A. Yorke with
Brian R. Hunt, Eric J. Kostelich and
Aleksey Zimin

© May 2006

CONTENTS

1. Introduction to the program <i>Dynamics</i>	1
The program <i>Dynamics</i>	1
Hardware requirements and recommendations for PC's	1
Running <i>Dynamics</i>	1
<i>Smalldyn</i> : a small version of <i>Dynamics</i>	1
1.1 <i>Getting started with Dynamics</i>	2
Quick start tutorial	3
Enter a menu item or enter a command	4
Retrieving menus	4
Viewing the latest menu while plotting	4
Help with commands	4
1.2 <i>Example: entering commands</i>	5
plotting a trajectory	5
drawing a box	5
viewing the parameter menu	7
refresh the screen and continue plotting	7
clear the screen and continue plotting	7
single stepping through a trajectory	7
plot a cross at current position	8
initializing	8
viewing the y vectors	9
set storage vector y1 and initialize	9
change x scale or y scale	9
quit <i>Dynamics</i>	9
1.3 <i>Some basic commands</i>	10
Basic commands for the screen (in <i>sm</i>)	10
Basic commands for drawing boxes (in <i>bxm</i>)	10
Basic commands for initializing and drawing crosses (in <i>km</i>)	11
Basic commands for drawing axes (in <i>axm</i>)	11
Basic commands for setting parameters (in <i>pm</i>)	12
Basic commands for changing variables (in <i>wvm</i>)	13
Basic commands for differential equations (in <i>dem</i>)	14
Commands for graphs, own, pause, quit and trajectories	15
1.4 <i>Some basis interrupts</i>	17
1.5 <i>Using the mouse</i>	18
2. Examples	19
2.1a <i>Plotting direction field and trajectories</i>	19
2.1b <i>Plotting direction field and trajectories</i>	24
2.2 <i>Basins of attraction</i>	25
2.3 <i>Plotting trajectory versus time</i>	27
2.4 <i>Bifurcation diagram: plotting trajectory versus parameter</i>	29

3. Adding your OWN process to Dynamics	31
3.1 <i>Introduction</i>	31
3.2a <i>Adding a new map</i>	32
Documentation window	33
Map window	33
Initialization window	33
No text in the Initialization window for maps	35
Example 3-1: Adding the Holmes map to <i>Dynamics</i>	37
Example 3-2: Adding a one-dimensional map to <i>Dynamics</i>	39
3.2b <i>Basic rules for adding a new map to Dynamics</i>	40
3.3a <i>Adding a new differential equation</i>	41
Vector field window	42
Initialization window	42
Modulo window	42
Text in the Initialization window for differential equations	42
Example 3-3: Adding the Lotka/Volterra eqs. to <i>Dynamics</i>	43
Example 3-4: Adding a SimBlum equation to <i>Dynamics</i>	45
Example 3-5: Adding the logistic equation to <i>Dynamics</i>	47
Example 3-6: Adding the GoodwiN equation to <i>Dynamics</i>	51
3.3b <i>Basic rules for adding a differential eq. to Dynamics</i>	52
4. Storing and printing pictures that are created by Dynamics	53
4.1 <i>Storing pictures and data</i>	53
Root name of the picture	53
Sending the picture to disk	54
Sending data to disk	55
Reading pictures from disk	55
4.2 <i>Setting the size of the core pictures</i>	56
4.3 <i>Printing pictures</i>	57
4.4 <i>Storing data and pictures of an OWN-process</i>	58
Reference	59

1. INTRODUCTION to DYNAMICS-WIN

The program Dynamics-Win

The computer program *Dynamics-Win* is for the novice and the expert, and helps the novice begin immediately exploring dynamical systems with a broad array of interactive techniques. When investigating a dynamical system, it is of course important to be able to plot trajectories, but that is just the beginning. An array of tools have been developed and combined in a single software package to help visualize what is happening in dynamical systems. Many of these tools such as plotting of "basins of attraction", computing "straddle trajectories", and automatically searching for all periodic orbits of a specified period are in this program *Dynamics*. These tools are elementary in that what they do can be understood by undergraduates and they are fundamental. The algorithms to implement these ideas are sometimes fairly sophisticated. Often, we refer to *Dynamics-Win* simply as *Dynamics*.

Running Dynamics

The program *Dynamics* calls upon the files *y.pic*, *yprom.txt*, *ymenus.txt*, *yps.txt*, *yhelp.txt*, and *yalert.txt*. They should be in the active directory when you start the program *Dynamics*. For example, if you are going to run the program from the directory *C:\DynWin*, use the command "*cd \DynWin*" in the command line to make that the active directory. If you want to add a map or differential equation, then the file *gserver.driv* is needed.

1.1 GETTING STARTED WITH DYNAMICS

To get started, if the directory that contains the program *Dynamics-Win* is C:\DynWin, for example, then type from the Run command line the command
C:\dynwin\dynamics-win <Enter>

It may be handy to activate the directory by typing in the command line "cd C:\DynWin <Enter>" and then Run **dynamics-win <Enter>**. You also may run the program just by clicking the appropriate folders/files via my computer. First, the title page of *Dynamics* appears on the screen for a few seconds and then the **PROcess Menu**. This menu will include the following

MAP MENU 1	
DE	- Differential Equations menu
OWN	- enter your OWN process
MAPS	
C	- complex Cubic map
CW	- CobWeb map, a 1-dim map
H	- Henon map
LOG	- LOGistic map
TT	- TenT map

After selecting the **Differential Equations menu**, hit <Enter> and the differential equations of the process menu appear on the screen and include

DIFFERENTIAL EQUATIONS MENU	
D	- forced double-well Duffing equation
GN	- GoodwiN equation
L	- Lorenz system
LV	- Lotka/Volterra equations
P	- forced damped Pendulum equation
VP	- forced Van der Pol equation

To select the Henon map (H) in the initial process menu (map menu 1), either use the arrow keys to select this process and hit <Enter> or type (both upper case and lower case letters may be used) **H <Enter>**

The **Main Menu** appears on the screen

NumExplM - FileM - ParameterM - VectorM - ScreenM - Help

2 Dynamics

Select NumExplM and hit <Enter> or type **nem**<Enter> and the **Numerical Explorations Menu** appears and includes

```
NEM -- NUMERICAL EXPLORATIONS MENU

T    - plot Trajectory
DYN  - quit & start new map or Differential Eqn.
OWN  - quit & create OWN process

      MENUS
BIFM - BIFurcation diagram Menu
BM   - Basin of attraction Menu
STM  - Straddle Trajectory Menu
UM   - Unstable and stable manifold Menu
```

If you are in the numerical explorations menu, either hit <Esc> or type **mm**<Enter> to return to the main menu. Use the arrow keys to select FileM or type **fm**<Enter> and the **File Menu** appears and includes

```
FM -- FILE MENU

FD  - retrieve picture (From Disk)
PP  - Print Picture
TD  - save picture (To Disk)

      MENUS
SCM - Size of Core picture Menu
DM  - Disk files Menu
```

If you are in the file menu, either hit <Esc> or type **mm**<Enter> to return to the main menu. Use the arrow keys to select Help and hit <Enter> or type **h**<Enter> and the **help menu** appears on the screen. There are two windows in the help menu. The top window contains the commands and the bottom window contains information. Many menus do have this structure. The bottom window is referred to as the **information window**.

Quick start tutorial

If you use the arrow keys to select the command **tut** and hit <Enter>, you will be presented with a quick start tutorial. This tutorial deals with the simplest commands available in *Dynamics* and yet provide a view of the capabilities of the program. *Dynamics* provides a wide range of capabilities to the experienced user and yet is rather easy for the novice to use.

Enter a menu item or enter a command

There are two ways to select the menu item H. One approach is to use the "point and shoot" menu system. When a menu appears, the bottom line of the box that contains the menu presents an item from the menu. That item is high-lighted and enclosed by two asterisks. By hitting the arrow key, you can move from item to item. When the one you want is selected, hit *<Enter>* to implement this selection. You also can click on a command with the left mouse button to high-light an item. When the one you want is selected, click the left mouse button again to implement this selection.

The second approach allows you to ignore which menu, if any, is on the screen. When you remember the command, this approach may be easier. Just type the command and hit *<Enter>*. When typing a command, you can use either upper or lower case letters.

For purposes of exposition, we emphasize the latter approach, but you may choose occasionally to use the "point and shoot" approach. The phrase "enter a command" means type the command (or select the command from a menu on the screen using the arrow keys) and then hit the *<Enter>* key.

Retrieving menus

Start the program and select the **Henon** map. Now the **main menu**, a list of menus, appears on the screen. Once you have selected a process and are working with it, if no menu is on the screen, you can fetch the previous menu by hitting *<Enter>* (assuming you are not in the midst of typing a command). If a menu is on the screen, fetch its parent menu by hitting *<Esc>*. This *<Esc>* can be repeated until the **main menu** appears. You can go directly to the **main menu** by entering command **mm**.

Viewing the latest menu while plotting

Sometimes you want to see the last-called menu while the computer is plotting. If the computer is plotting, then whenever you hit *<Enter>*, the program pauses and the last called menu appears on the screen. In case you do not want to have this latest menu, just hit *<space bar>* and the menu is erased and the computer continues plotting.

Help with the commands

To get help with the command that is high-lighted in a menu on the screen, just hit the *<Tab>* key. ("Hit" a key means gently depress and release the key.) For most commands, a description will appear on the screen. There is another way to get this information and can be used even if the command is not showing on the screen. To get help with a command, hit *<*>* and enter the command. For example, to see what command **b** does, type ***b<Enter>** and the program prints out the description. (Notice there is no space between ***** and **b**.)

1.2 EXAMPLE: entering commands

Let F be a continuous map from the n -dimensional phase space to itself and consider the corresponding discrete time process $\mathbf{x}_{k+1} = F(\mathbf{x}_k)$. For each point \mathbf{x}_0 , the point \mathbf{x}_k is called the k th iterate of the point \mathbf{x}_0 .

The trajectory of any point \mathbf{x} in the phase space is the finite or infinite sequence of consecutive iterates of \mathbf{x} .

Given a pair (x,y) , applying the Henon map once gives a new pair using the formula $(rho - x*x + c1*y, x)$. The symbol '*' in processes of *Dynamics* (maps and differential equations) denotes multiplication. The Jacobian determinant of the Henon map is $-c1$.

Sometimes, for example, when plotting a trajectory, there are two <Enter> keys "←" because after the first ← a menu of hints and options appears. By hitting the second ← you are ignoring these options instructing the program that you do not want to use these options.

```
dynamics-win ← Start the program Dynamics-Win;  
h ← select the Henon map and the main menu appears;  
nem ← get the numerical explorations menu;
```

plotting a trajectory

```
t ← ← plot the trajectory (initial condition y)
```

The initial values for x and y are 0.0 and 2.0 respectively, and the program will apply the process (which is the Henon map) to the point (x,y) and then plot the resulting point. In other words, the program replaces the value of $y = (x,y)$ with the value $(rho - x*x + c1*y, x)$, and it will do this repeatedly, that is, it "iterates" the map. A sequence of dots will appear in rapid succession, hundreds or thousands per second.

drawing a box

The plotting of the trajectory can be paused at any time in order to perform other commands that enhance or change what the program is doing. For example, you can draw a box around the plotted trajectory by locating the "Box" command in the menu system and executing it.

```
← return to numerical explorations menu;  
<Esc> get the main menu on the screen;  
sm ← select the screen menu;  
bxm ← select the box menu;  
b ← draw a box;
```

and a box will be drawn around the screen which frames the trajectory.

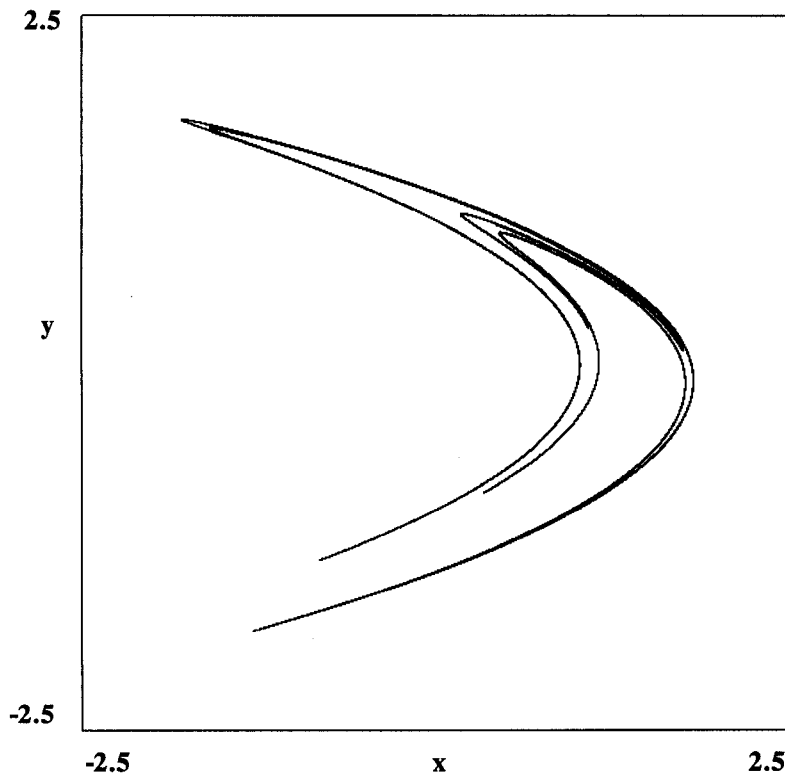


Figure 1-1: Trajectory of the Hénon map

This picture shows a trajectory of the Hénon map (h)

$$(x, y) \rightarrow (1.4 - x^2 + 0.3y, x)$$

In fact, the trajectory of any initial conditions selected from some open set yields a similar picture if the first 20 iterates or so are not plotted. Let $F: \mathbb{R}^2 \rightarrow \mathbb{R}^2$ be a continuous map. A compact set $S \subset \mathbb{R}^2$ is called an **attractor** for F if (a) $F(S) = S$; (b) there exist an open neighborhood U of S such that for every $x \in U$ the distance of x_n and S converges to zero as $n \rightarrow \infty$ (more precisely, for every $\epsilon > 0$, there exists a positive integer N such that for every $n \geq N$ the distance of x_n and S is less than ϵ); (c) the collection of limit points of $\{x_n\}$ is equal to S . In fact, the picture shows an attractor. This attractor is known as the **Hénon attractor**. Frequently, this attractor is called a "strange attractor" because it appears to be a Cantor-like set of curves. Hénon may have chosen $\rho = 1.4$ and $c_1 = 0.3$ because of the beauty of the attractor. What do the attractors look like when the Jacobian c_1 is chosen larger than 0.3 for various ρ values? (There are no attractors if $|c_1| > 1$.)

viewing the parameter menu

<Esc> terminate the plotting of the trajectory
and the box menu appears (since it was the last menu you called). Hit the key <Esc> twice more and the main menu appears on the screen.

pm ←| get the parameter menu to view the parameter settings
The program gives *rho* the default parameter value of 2.12 and *c1* the value -0.3. For this process, the initial setting for the screen is that *x* is plotted horizontally with a scale running from -2.5 to 2.5, and *y* is plotted vertically with the same scale.

<Esc> return to the main menu.
nem ←| get the numerical explorations menu;
t ←| ←| plot the trajectory

and the previous picture returns to the screen and the program continues plotting the trajectory. The picture can be returned to the screen because the program has a separate copy of the picture in core memory, the "core" copy of the picture, in addition to the screen picture. The core copy of the picture usually has higher resolution than the screen. The core copy is 720 dots wide by 720 dots high if there is sufficient memory available, and when the picture is sent to the printer, this higher resolution picture is transmitted. The core picture does not include the words on the screen so pictures can be refreshed (that is, cleaned up) using the command *r*.

refresh the screen and continue plotting

sm ←| call the screen menu;
r ←| refresh the screen;

to get a refreshed picture without all the text that was also on the screen. Hence, the pictures sent to the printer are not muddled up with extraneous text such as menus.

clear the screen and continue plotting

Assuming that a trajectory of the Henon map is still being plotted and that a box has been drawn around the screen,

←| return to screen menu;
c ←| clear the screen and core memory;

The screen clears and the computer continues plotting the trajectory from where it left off.

single stepping through a trajectory

Sometimes the computer plots too fast for what you want to see or do. To see the process iterated at a slower rate, hit <·> and the process will pause. Hit <·> again and the program will iterate the map once and plot the

resulting point. A large cross is plotted at the current (x,y) point. Repeatedly hitting $\langle \cdot \rangle$ you will see how the trajectory moves through the chaotic attractor. Hence, these processes can be walked through as slowly as you want.

$c \leftarrow$ clear the screen and core memory, and immediately
 $\langle \cdot \rangle \dots \langle \cdot \rangle$ hit the period key $\langle \cdot \rangle$ several times
and you will see the process single step, producing one dot for each ' \cdot ', and pausing thereafter. While the program is in this "pause after one dot" mode, you can still enter commands to reinitialize or clear, etc., and while the program is paused, a cross will appear on the screen at the current (x,y) point.
 $\langle \text{space bar} \rangle$ terminate single-stepping pause mode
and the trajectory resumes iterating again as fast as it can.

plot a cross at current position

$k \leftarrow$ plot a cross at current position
and the computer will show the position of each dot it plots with a cross. This command is in the **kruis** menu **km**, which is in the screen menu **sm**. This feature is a toggle. A *toggle* is a command that turns some feature on or off. If it is *on*, entering the command turns it *off* and vice versa. Hence, when the command **k** is entered again, the feature turns off and no crosses are plotted anymore.

$k \leftarrow$ turn the feature of plotting a cross *off*;
 $c \leftarrow$ clear the screen and core memory;

initializing

The current state of the trajectory is a vector called y (or sometimes called y_0). When the trajectory is paused, a large cross appears at that point. Another vector, the "initialization" vector, denoted y_1 , has certain special uses. When the trajectory is paused (or when the vector y_1 is changed using the arrow keys) a small cross appears at its location. It is the initial condition for y .

$i \leftarrow$ initialize

which replaces y by the position of y_1 . After the command **i** has been entered, the trajectory will be restarted from the small cross whose position is given by the initialization vector y_1 . The command **i** is in the **kruis** (cross) menu **km**, which is in the screen menu.

Note. Using a mouse, you may wish to select a new initial condition by pointing the arrow at the desired position and **double** click the left button for initialization.

viewing the y vectors

The values of the coordinates of the vectors y and $y1$ and some of the other storage vectors can be seen by entering the command yv .

yv ← get the y vectors to view the storage vectors
 r ← refresh the screen;
to get rid of the text

set storage vector $y1$ and initialize

$\langle Esc \rangle$ terminate the plotting of the trajectory;
 ρ ← 1.4 ← set ρ to be 1.4;
 $c1$ ← 0.3 ← set $c1$ to be 0.3;
 i ← initialize;
 t ← ← plot the trajectory
and a warning "trajectory is too far from the screen: **process paused**" appears on the screen with a menu (recover menu) of possible corrections. Now set $y1$ to be (0,0).
 sv ← set vector;
 $sv1$ ← set vector $y1$;
 $sv10$ ← 0 ← select $sv10$ to set coordinate #0 of $y1$ equal to 0;
 $sv11$ ← 0 ← select $sv11$ to set coordinate #1 of $y1$ equal to 0;
Notice that if you are using the menu, you should now respond with **ok** twice by hitting $\langle Enter \rangle$ twice. Now initialize, and plot a trajectory.
 i ← initialize;
 t ← ← plot the trajectory
and (if appropriate) hit $\langle space bar \rangle$ to unpause the routine.

change x scale or y scale

In this example, the horizontal or x scale runs from -2.5 to 2.5. We want to change this scale to run from -2 to 2 using command xs .

pm ← fetch the parameter menu;
 xs ← -2 2 ← change the x scale to run from -2 to 2;
with a space between -2 and 2. You can similar change the y scale to run from -2 to 2 using the command ys .
 ys ← -2 2 ← change the y scale to run from -2 to 2;
 c ← clear the screen and core memory.

quit *Dynamics*

There two ways to quit the program. (a) You may quit *Dynamics* such that an options menu appears (command q). For example, you may want to save some data or picture before exiting *Dynamics*. (b) You may quit and exit *Dynamics* without saving anything (command qx).

1.3 SOME BASIC COMMANDS

Basic commands for the screen (in screen menu sm)

C	-	Clear current window
CS	-	Clear Screen (but not core copy)
R	-	Refresh screen

C, CS

The command **c** clears the active window (and the core copy). The command **cs** clears the screen but not the core copy. The refresh command **r** will restore the picture.

R

The program keeps a copy of the picture in core that is or can be of higher resolution than the copy on the screen. The command **r** refreshes the screen, eliminating extraneous text by retrieving it from the core memory copy. The "core" memory copy is the copy of the picture that is used for printing high resolution copies on a printer.

Basic commands for drawing boxes (in box menu bxm)

B	-	draw Box
B1	-	draw Box with tic marks
B2	-	draw Box with double tic marks
BB	-	set Box equal to entire window

B, BB

The command **b** causes a box to be drawn. The default position of the box is the entire screen.

If you enter **b** and no box is drawn, this usually suggests the box is outside the screen area, due to your change in coordinates. Use **bb** to reset the box to be the whole screen, and try **b** again. When you enter command **bb**, no box is drawn, but if you then enter command **b**, a box will be drawn.

B1, B2

Command **b1** draws a box and then draws tic marks, from 4 to 9 tic marks on a side, choosing the number so as to give natural divisions.

Command **b2** draws a box and then draws a double set of tic marks. The first set is identical with the tic marks of command **b1**. The second set is a finer set of small tic marks.

Basic commands for initializing and drawing crosses (in kruis menu km)

I	- Initialize y using y1
K	- big cross along trajectory: OFF
KK	- permanent cross at y
KK1	- permanent cross at y1
Arrow keys move small cross which is at y1 or 2-click left mouse button at desired spot.	

I

The command **i** reinitializes the trajectory vector **y** to be the current position of **y1**; the latter's position is shown by the small cross.

K

The command **k** results in a cross being plotted as each dot is plotted; the cross is erased when a new dot is plotted.

k is a toggle, so if you enter **k** again this feature turns off.

KK, KK1

When command **kk** is entered, a permanent cross turns on at the current position **y** (= **y0**) of the trajectory. "Permanent" means it is also drawn on the core copy of the picture.

The command **kk1** draws the permanent cross at **y1** instead of **y** (= **y0**).

Basic commands for drawing axes (in axes menu axm)

XAX	- X AXis
XAX1	- X Axis with tic marks
XAX2	- X Axis with double tic marks

XAX, XAX1, XAX2

The command **xax** causes the **x** axis to be drawn. The command **xax1** draws the **x** axis and then draws tic marks, from 4 to 9 tic marks on a side, choosing the number so as to give natural divisions.

The command **xax2** draws the **x** axis and then draws a double set of tic marks. The first set is identical with the tic marks of command **xax1** and the second set is a finer set of small tic marks.

YAX, YAX1, YAX2

The commands **yax**, **yax1** and **yax2** are analogous to the **x** axis commands **xax**, **xax1** and **xax2**.

Basic commands for setting parameters (in parameter menu pm)

XCO	-	X COordinate to be plotted:	y[0]
XS	-	X Scale: from -2.5 to 2.5	
YCO	-	Y COordinate to be plotted:	y[1]
YS	-	Y Scale: from -2.5 to 2.5	
C1	-	C1 =	-0.30000000
RHO	-	rho =	2.12000000

The process parameters are set using **c1**, **c2**, ... , **rho**, etc. Only those process parameters that are used in the equations appear in the parameter menu. Enter the parameter's name (like **rho**) and the program will prompt for a new value.

Basic commands for setting parameters not being process parameters

XS, YS

The command **xs** allows you to change the horizontal scale of the screen for plotting dots; the first number is the left hand side and the second is the right one. The two numbers must be entered with a space but no comma between them. (The first number can be larger than the second one if you want a decreasing scale.)

The command **ys** allows you to change the vertical scale of the screen. The first number is the coordinate of the bottom of the screen and the second one is the top.

XCO, YCO

Most variables take on numerical values. The value of the variable **xco** is a name, the name of the horizontal **x** coordinate variable. This command is for specifying the variable or parameter that is to be plotted horizontally. For the Henon map, the default value of **xco** is "y[0]", which is the first coordinate of the vector **y**. With this setting the numerical value of **y[0]** is plotted horizontally. The value of **xco** can either be a coordinate of the process (like **y[0]**) or a process parameter (like **rho** or **c1**). To specify a coordinate like **y[1]** for the horizontal axis, type **xco**<Enter> and then enter just the number of the coordinate. In the case of **y[1]**, type **xco**<Enter> **1**<Enter> . To plot a parameter of the process, either **rho**, or **c1**, **c2**, ... **c9**, type the process parameter and <Enter>. For example, **xco**<Enter> **c1**<Enter> .

If you choose a parameter and plot a trajectory, then the trajectory will appear on a vertical line, the line corresponding to that parameter value.

The value of **yco** is a name, the name of the vertical **y** coordinate variable. This command is for setting the variable or parameter that is to be plotted vertically, and is otherwise similar to **xco**. You can even assign one parameter horizontally (using **xco**) and another vertically in order to plot analogues of Mandelbrot sets (using command **bas** or other related commands in the basin of attraction menu, see **bm**).

Both **xco** and **yco** can be set equal to the same variable. Suppose for example, both coordinates are set to be variable **0**. Then the program plots vertically the specified coordinate of the trajectory, but horizontally it plots that coordinate from the previous time. That is, the vertical coordinate is delayed. To make the delay several time steps instead of 1, use **ipp** to change the number of iterates per plot.

See **prm** for the vertical coordinate in bifurcation diagrams.

Basic commands for changing variables (in when & what menu **wwm**)

CON	-	CONnect consecutive dots:	OFF
PI	-	PreIterates before plotting:	0
PRM	-	PaRaMeter to be varied by +/- :	"rho"
PT	-	T Plots Time horizontally if ON;	now OFF

CON

The command **con** causes successive dots to be connected by straight lines. It can also be used for maps when it is appropriate.

con is a toggle: if **con** is *on*, enter it again and it turns *off*.

PI

The command **pi** is for setting the number of **preiterates**, that is, the number of iterates computed before the first point is plotted. When iterating a process, it may be iterated **pi** times before plotting begins. The number of **preiterates** **pi** should be at least 0. The usual default is 0.

PRM

The command **prm** is for specifying the **parameter**, for example, the parameter to be varied in bifurcation diagrams. The value of the variable **prm** is a name. The value of **prm** is the name of the parameter to be varied.

PT

The command **pt** is a toggle. When **pt** is *on*, the trajectory command **t** plots the time horizontally. Vertically it plots **y[1]** (assuming **l = 0**, that is, the Lyapunov exponents are not being computed). The consecutive dots can be connected by invoking the "connect consecutive dots" command **con**.

Basic commands for differential equations (in differential eqs. menu dem)

EULER	- Euler solver, fixed step size
RK4	- 4th order Runge-Kutta solver, fixed step size
DF	- plot the Direction Field
STEP	- STEP size for differential equation: 0.01

DF

The command **df** is for plotting the direction field of a differential equation. See also command **fg**.

EULER

The command **euler** is for choosing Euler differential equation solver. This command makes the differential equation solver a single step Euler solver.

FG

The command **fg** is for specifying the grid used by **df** (direction field) and **vf** (vector field). This command is in the actions, hints and options menu for **df**.

RK4

This command makes the differential equation solver a fourth order fixed-step-size Runge-Kutta solver. This is the usual default solver.

STEP

The command **step** sets the **step** size of the differential equation solvers **rk4** and **euler**. Depending on the differential equation(s) you are exploring, in order to get reliable numerical results, you may have to set **step** to a much smaller value; for example, you may have to set **step** to 0.0001 or 0.00001.

Commands for graphs, own, pause, quit and trajectories

DYN

The command **dyn** is used to quit and to restart *Dynamics*. This command allows selection of another differential equation or map; most parameters are reset to the default values.

G

The "plot graph" command **g** plots the graph of 1-dimensional functions. The routine **g** calculates the function value for a fixed number of points (see **wide** in the size of core menu to find this value).

If the iterates per plot **ipp** is changed from 1 to 3, for example, then **g** will graph the third iterate of the function. You may wish to **connect** the consecutive dots by first entering the command **con**. See command **con** in **wwm**. Enter **con** again after creating the graph to turn that feature off.

OWN

The command **own** is used to quit the program *Dynamics* for creating your **own** process (map or differential equation).

P

The command **P** causes the program to Pause. Hit the <space bar> or enter the "un-pause" command **up** to end pause mode. If you are going to start a process like **t** or **um** and you want it to be paused initially, then this is the command to use. The pause initiated by <·> while plotting has a different effect: the program will automatically un-pause from the <·> pause if you start a new process.

Q, QX

The command **q** is for quitting the program. Enter this command and a menu appears on the screen before exiting the program, to allow saving of the data.

The command **qx** is for quitting and exiting *Dynamics*. This command terminates the program; data that has not been saved will be lost.

T

The "trajectory" command instructs the computer to plot the dots of a trajectory. The maximum number is specified by the command **dots**. Usually **dots** is set to such a large value that it will not be reached. Use **x** coordinate and **y** coordinate commands **xco** and **yco** to specify which coordinates will be plotted against which horizontally and vertically. See **xco**. Usually **xco** and **yco** are set to be the space coordinates of the process. For example, in Henon the default setting of **xco** is 0, which means **y[0]** is plotted horizontally, and **yco** is 1 which means **y[1]** is plotted

vertically. However as shown below **xco** and/or **yco** can be set to be parameters of the map.

After entering the trajectory command **t**, an options menu appears. One of the options is to follow a number (**tn**) of trajectories simultaneously. Usually routine **t** plots a single trajectory, but if the trajectory number **tn** is greater than 1, the routine **t** plots **tn** trajectories simultaneously. If you set **tn** to be 100 for example, 100 initial points will be chosen. See commands **tn** and **tnb** for more information.

You can specify that one of the axes is a parameter of the system, for example the horizontal axis could be for the parameter *rho* (if that is one of the parameters of the currently selected process), and then *rho* would be varied over a range specified using the "x scale" command **xs**. Of course, the trajectory would then lie on a single line corresponding to that parameter value. The sequence of commands

xco<Enter> **rho**<Enter> **yco**<Enter> **1**<Enter>

says that **xco** is *rho*, that is, *rho* will be plotted horizontally, that is the *x* coordinate, while coordinate 1 of the state vector will be plotted vertically.

T, TN, TNB

The "trajectory number" command **tn** specifies the number of trajectories that the routine **t** will follow simultaneously. The "tn box" command **tnb** is a toggle; it is either "on" or "off".

If **tn** = 1, then the routine **t** plots a single trajectory starting from the point **y[]**. If **tn** > 1, then the routine **t** plots **tn** trajectories simultaneously; each time you enter command **t**, these **tn** initial points are chosen anew. The initial conditions can be either on the line from **ya** to **yb** if **tnb** is "off", or if **tnb** is "on", they are chosen in the "small box".

When **tn** > 1 and **tnb** is *off*, the trajectory command **t** chooses **tn** initial points equally spaced points are chosen on the segment from **ya** to **yb**. These points include **ya** and **yb**, and they are used as initial points for trajectories. If you set **tn** to be 100 for example, 100 initial points will be chosen on the line from **ya** to **yb**. (See **y** vectors command **yv** to see what these values are at any time. Initially they are in the lower left and upper right corners of the screen so the line is the diagonal connecting them.) Then **t** will follow 100 trajectories simultaneously. You can see the process more clearly if before entering the command **t** you first enter the pause command **p**. Then when **t** is entered, the program will be paused. Hit the period key < · > and all 100 will be advanced one iterate. Hit < · > again and all will advance one more step.

When **tn** > 1 and **tnb** is *on*, the trajectory command **t** chooses **tn** points at random from a rectangle which is either the current window (or whole screen) or if the small box has been set, it chooses the initial points in that box. You can see the small box, if it has been set, using command **b**.

1.4 SOME BASIC INTERRUPTS

Recall that "Enter a command" means type the command (using upper or lower case) and then hit the <Enter> key. A few commands (called **interrupts**) are executed without hitting <Enter> and are called by a single key stroke such as <·>, <Tab>, <Esc>, <space bar>, and the arrow keys.

<·>

The key <·> (one dot) pauses the program *Dynamics* after plotting one dot. Each time <·> is hit, the program computes one more dot and plots it and pauses. The large cross appears at the current location of the trajectory. Holding <·> down will produce a string of iterates; <space bar> returns the program to normal continuous plotting of dots.

<space bar>

<space bar> ends the pause mode; see also command **p** (pause) and interrupts <·> and <Enter> which each initiate a pause mode.

<Esc>

When <Esc> is hit, the current routine terminates or if a menu is on the screen, it terminates the menu and retrieves the parent menu. Hit it again and the program calls its parent menu.

<Tab>

If a process is being run and you hit <Tab>, the program prints the speed (in dots per second) and a selection of parameter values.

If a menu is on the screen, hitting <Tab> will fetch information about the high-lighted command with information.

<F1>, <F2>, <F3>, <F4>, <F10>

The Function keys <F1>, <F2>, <F3>, and <F4> can be used to switch back and forth between windows that have been opened. Use <F10> to switch back to the whole screen.

Each window is a quadrant of the screen, situated as follows:

<F1>	<F2>
<F3>	<F4>

<F7>, <F8>, <F9>

The color numbers on the screen are usually numbered 0 to 15. The "color table" command **ct** displays the color table. The program has an active color number that is used in all plotting (and in refreshing the screen). Function key interrupt <F9> is used for setting the value and the program will prompt the user for a color number; <F8> increases that number while <F7> decreases it.

1.5 USING THE MOUSE

The mouse can be used for a variety of purposes. It is not essential for *Dynamics*, but it can simplify procedures.

Selecting items in menus (*left mouse button*)

If there is a menu on the screen, you can select an item in that menu using the left mouse button. If you click on a command that is not illuminated, it will become illuminated. If it is illuminated and you click on it, then that command will be executed.

Picking a new initial point of a trajectory (*left mouse button*)

If no menu is displayed on the screen, then the mouse can be used to set the initializer y_1 . (If a menu is displayed and you want it to vanish, just hit the <space bar>). Move the mouse and the mouse arrow will appear. Move it to the position on the screen that you select to be the new initial point. When you have selected a new initial point, **double** click the left mouse button. If you are plotting a trajectory, this will reinitialize the trajectory and plotting will continue from this newly selected point.

If you are not using a mouse, the same can be accomplished by using the arrow keys to move the small cross, and then entering command **i** (initialize).

2. EXAMPLES

2.1a PLOTTING DIRECTION FIELD AND TRAJECTORIES

The purpose of this example is to plot some trajectories together with the direction field of the Lotka/Volterra differential equations (lv)

$$x' = (c_1 + c_3*x + c_5*y)*x, \quad y' = (c_2 + c_4*y + c_6*x)*y$$

where the parameter values are given by $c_1 = 0.5$, $c_2 = 0.5$, $c_3 = -0.0005$, $c_4 = -0.0005$, $c_5 = -0.00025$, and $c_6 = -0.00025$.

dynamics-win ← | Start the program *Dynamics-Win*;
de ← | fetch the differential equations menu;
lv ← | get the Lotka/Volterra equations;
pm ← | fetch the parameter menu and verify the values above;
dem ← | fetch the differential equations menu;
df ← | and plot the direction field.

A resulting picture is given in Figure 2-1a.

For plotting trajectories, mark the initial conditions by a small cross.

c ← | clear the screen;
km ← | fetch the kruis (cross) menu,
kks ← | **5 5** ← | and set the scale **kks** of the permanent cross to be 5 pixels by 5 pixels;

To get continuous trajectories, connect the consecutive computed dots.

wwm ← | fetch the when and what to plot menu, and
con ← | connect consecutive dots;
p ← | pause the program;
t ← | ← | plot the trajectory.

i ← | **kk** ← | initialize and draw a permanent cross at **y**;
<space bar> hit the <space bar> to un-pause;
<←,↓,→,↑> use the arrow keys to move the small cross to a new initial condition for a trajectory, and

p ← | pause the program.
Repeat the 4 previous steps (**i** ← | **kk** ← |, <space bar>, <←,↓,→,↑>, **p** ← |). A resulting picture is given in Fig. 2-1b.

Note. Instead of going through the 4 steps repeatedly, an easy way to initialize is to move the mouse arrow to the desired position and then **double** click the left mouse button. To have a cross drawn at this position of **y1**, enter command **kk1**. You can plot several trajectories simultaneously (command **tn**, or commands **tnb** and **tn**), but no crosses can be plotted.

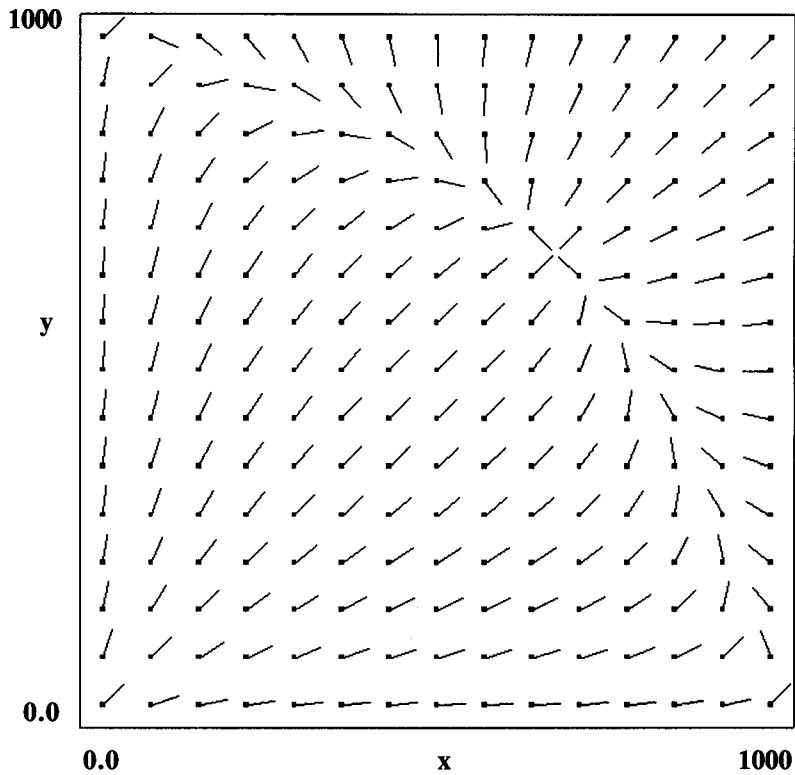


Figure 2-1a: Direction field

This figure shows a direction field for the Lotka/Volterra differential equations

$$\begin{aligned}x' &= (c_1 + c_3*x + c_5*y)*x \\y' &= (c_2 + c_4*y + c_6*x)*y\end{aligned}$$

where $c_1 = 0.5$, $c_2 = 0.5$, $c_3 = -0.0005$, $c_4 = -0.0005$, $c_5 = -0.00025$, and $c_6 = -0.00025$.

Topic of discussion. This figure suggests there are "at least" four equilibria. What are they?

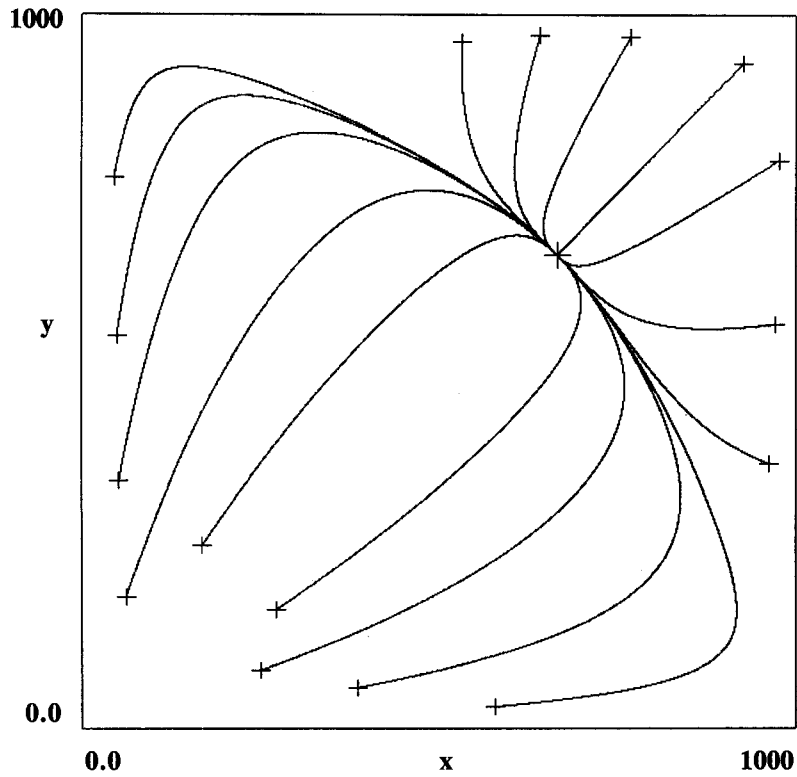


Figure 2-1b: Trajectories of an ODE

This figure shows some trajectories for the Lotka/Volterra differential equations, with parameters as in Figure 2-1a. The initial value of each trajectory is indicated with a small cross.

Topic of discussion. *There is an asymptotically stable equilibrium where the trajectories converge. What does this picture suggest about the eigenvalues and eigenvectors of that equilibrium?*

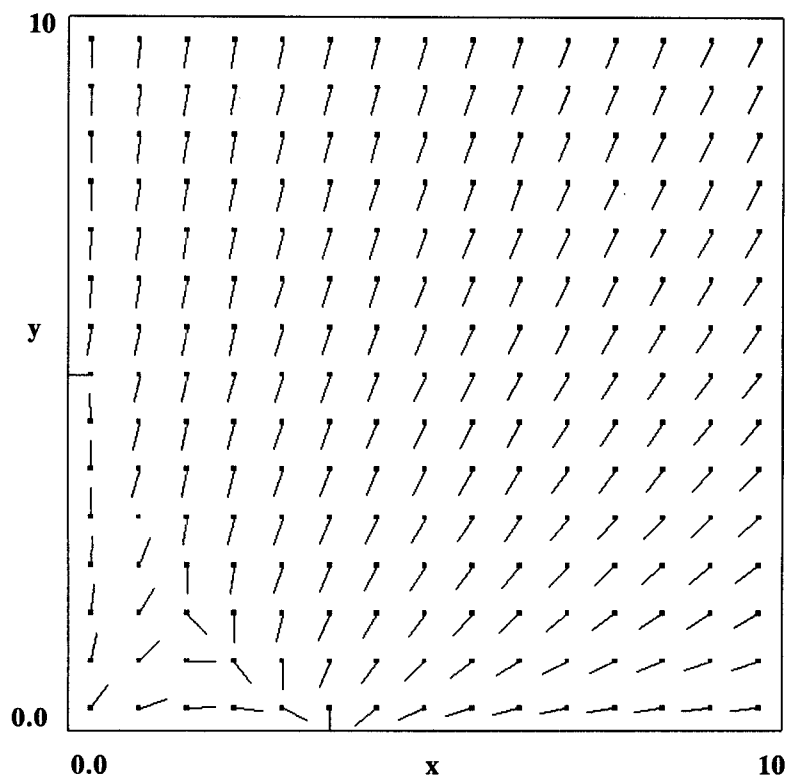


Figure 2-1c: Direction field

This figure shows a direction field for the Lotka/Volterra differential equations

$$\begin{aligned}x' &= (c_1 + c_3*x + c_5*y)*x \\y' &= (c_2 + c_4*y + c_6*x)*y\end{aligned}$$

where $c_1 = 4$, $c_2 = 6$, $c_3 = -1$, $c_4 = -1$, $c_5 = -1$, and $c_6 = -3$.

*If you wish, you may change the grid (command **fg**, which is in the Actions, Hints and Options menu for **df**). The default setting of a (15 by 15) grid (**fg = 15**) has been used in creating this picture.*

Topic of discussion. This figure suggests there are "at least" four equilibria. What are they?

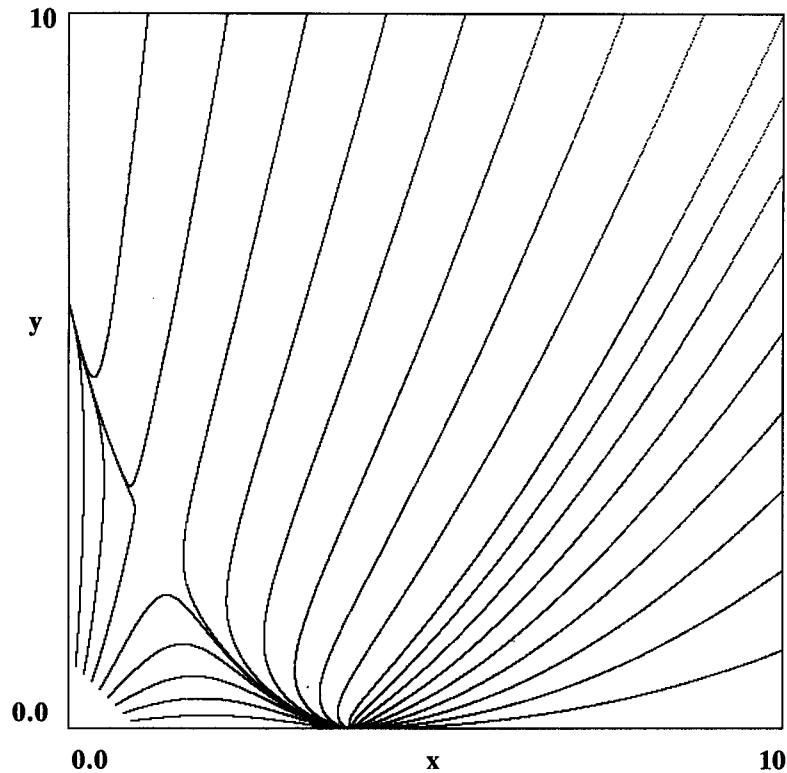


Figure 2-1d: Trajectories of an ODE

This figure shows some trajectories for the Lotka/Volterra differential equations, with parameters as in Figure 2-1c. Ten = 10 trajectories are plotted simultaneously from (10,0)-(10,10), (0,10)-(10,10) and (1,0)-(1,0).

Topic of discussion. There are two asymptotically stable equilibria where the trajectories converge. What does this picture suggest about the eigenvalues and eigenvectors of those equilibria?

To create a part of this picture, carry out the steps in Example 2-1b through entering the step size. Then enter the following commands.

```

tn ← 10 ← plot 10 trajectories simultaneously if t is entered;
diag ← set ya and yb to the opposite corners of the screen,
that is, ya = (0,0) and yb = (10,10);
yv ← get the y vectors and verify the settings for ya, yb;
sv ← sva ← set vector and set vector ya;
sva0 ← 10 ← select sva0 to set coordinate #0 of ya equal to 10;
t ← ← plot the 10 trajectories starting at ya-yb.

```

2.1b PLOTTING DIRECTION FIELD AND TRAJECTORIES

The purpose of this example is to plot some trajectories together with the direction field of the Lotka/Volterra differential equations (lv)

$$x' = (c_1 + c_3*x + c_5*y)*x, \quad y' = (c_2 + c_4*y + c_6*x)*y$$

where the parameter values are given by $c_1 = 4$, $c_2 = 6$, $c_3 = -1$, $c_4 = -1$, $c_5 = -1$, and $c_6 = -3$, and the scales of the variables $0 \leq x, y \leq 10$. To get reliable numerical results, set the step size of the DE-solver to 0.0001.

```
dynamics-win ← Start the program Dynamics-Win;  
de ← fetch the differential equations menu;  
lv ← get the Lotka/Volterra equations;  
pm ← fetch the parameter menu;  
c1 ← 4 ← set the value of  $c_1$  to 4;  
c2 ← 6 ← set the value of  $c_2$  to 6;  
c3 ← -1 ← set the value of  $c_3$  to -1;  
c4 ← -1 ← set the value of  $c_4$  to -1;  
c5 ← -1 ← set the value of  $c_5$  to -1;  
c6 ← -3 ← set the value of  $c_6$  to -3;  
xs ← 0 10 ← set the x scale to run from 0 to 10;  
ys ← 0 10 ← set the y scale to run from 0 to 10;  
dem ← fetch the differential equations menu;  
step ← .0001 ← set the step size to 0.0001;  
df ← ← and plot the direction field.
```

A resulting picture is given in Figure 2-1c.

For plotting trajectories, mark the initial conditions by a small cross.

```
c ← clear the screen;  
km ← fetch the kruis (cross) menu,  
kks ← 5 5 ← and set the scale kks of the permanent cross to be 5  
pixels by 5 pixels.
```

To get continuous trajectories, connect the consecutive computed dots.

```
wwm ← fetch the when and what to plot menu, and  
con ← connect consecutive dots.  
p ← pause the program;  
t ← ← plot the trajectory.
```

Either carry out repeatedly the 4 steps (i ← kk ←, <space bar>, <←,↓,→,↑>, p ←) in Example 2-1a or use the mouse (move the mouse arrow to the desired position and then double click the left mouse button to initialize and enter command **kk1** to draw a cross at this position of y_1). You can plot several trajectories simultaneously (commands **tnb** and **tn**), but no crosses can be plotted. A resulting picture is given in Fig. 2-1d.

2.2 BASINS OF ATTRACTION

The **basin of an attractor** is the set of points whose iterates tend to that attractor. In this example we want to plot all the basins and attractors for the **Henon map** (**h**)

$$(x,y) \rightarrow (rho - x*x + c1*y, x)$$

where the parameter values are given by $rho = 0.5$ and $c1 = 0.9$.

dynamics-win ←	start the program <i>Dynamics-Win</i> ;
h ←	select the H enon map and the main menu appears;
pm ←	select the p arameter menu;
rho ← 0.5 ←	set the value of rho to 0.5;
c1 ← 0.9 ←	set the value of $c1$ to 0.9;
← <Esc>	retrieve the p arameter menu and then the main menu;
nem ←	select the n umerical e xplorations menu;
bm ←	fetch the b asin of attraction menu;
ba ←	plot the b asins and attractors.

A 100 by 100 grid of points will be tested. The colors used to color the grid boxes indicate something that is in the box. Initially all grid boxes are uncolored but as the routine **ba** proceeds, all grid boxes will eventually be colored. The **ba** routine selects an uncolored grid box and tests the box by examining the trajectory that starts at the center of the grid box. The trajectory may or may not pass through grid boxes that have previously been colored, but what it encounters will determine how the initial grid box is colored. First, pixels are colored dark blue, indicating that the trajectories of the centers of these boxes diverge. After a while, some boxes are colored light blue followed by boxes which are colored green. The dark blue region is the basin of infinity. There is one other attractor, namely an asymptotically stable period 2 orbit (green). The basin of the period 2 orbit is colored light blue.

If you print the current picture on paper, you just get a solid black plot. The program has the feature to erase a color. The basin of the period 2 attractor is colored with color number 3.

cm ←	fetch the c olor menu;
em ←	select the e rase color menu;
e3 ←	erase color 3.

The resulting picture is similar to Figure 2-2.

Note. The resolution of the picture that appears on the screen, is low. By first invoking a command like **rh** (**h**igh **r**esolution), the command **ba** generates a picture of high resolution.

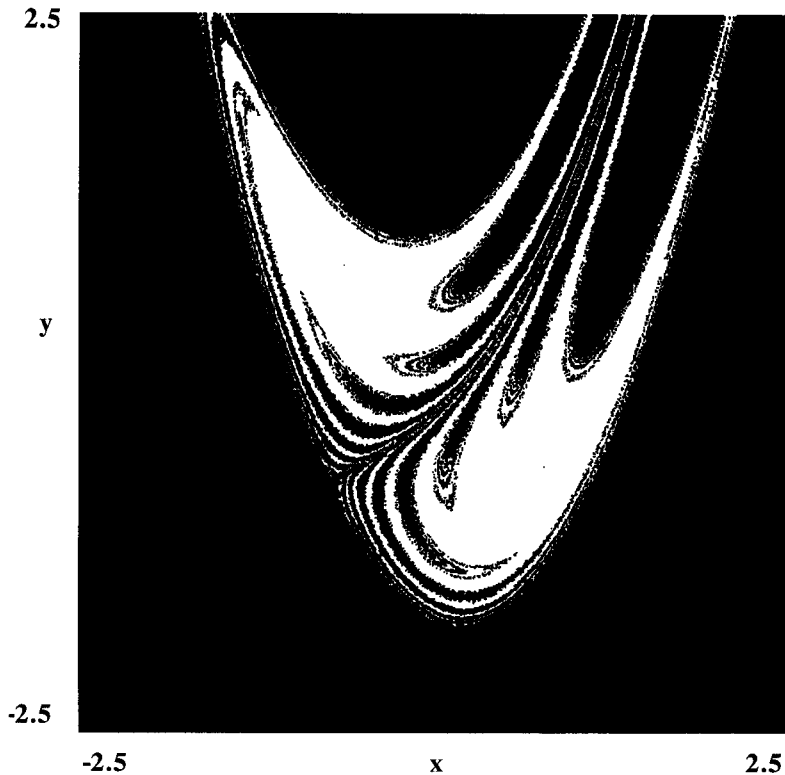


Figure 2-2: Basins of attraction and Attractors

The black area and white areas are two basins of the Henon map (h)

$$(x,y) \rightarrow (0.5 - x*x + 0.9*y, x)$$

The black area shows points whose trajectories diverge (to infinity). There are two attractors for this choice of parameters. The routine `ba` plots all attractors and their basins. It has plotted the basin of infinity, and the basin of a stable period 2 orbit. The white area is the basin of attraction of the period 2 attractor, and is obtained by erasing the color of the basin of the period 2 attractor. We used command `e3` to erase color 3. The isolated dots (in the white region) are on the period 2 attractor. Its basin has been erased and so is white.

There are several ways for finding out what kind of simple attractors may exist. The easiest way may be the following. Plot the basin in high resolution (commands `rh`, `ba`). After the picture has been completed, enter command `cnt` to count the pixels and observe there are two pixels colored with color 2 indicating that a period-2 attractor exists.

2.3 PLOTTING TRAJECTORY VERSUS TIME

The purpose of this example is to plot the trajectory (iterates 1 through 100) of $x_0 = 0.4$ versus "time" of the logistic map (**log**)

$$x \rightarrow 3.83*x*(1-x)$$

dynamics-win ← start the program *Dynamics-Win*;
log ← select the **logistic map** and the main menu appears;
pm ← select the **parameter menu**;
rho ← **3.83** ← set the value of *rho* to 3.83;
con ← turn the toggle **con on** to connect consecutive dots;
Set Vector y1 to be 0.4:
sv ← set vector;
sv1 ← set vector **y1**;
sv10 ← **0.4** ← select **sv10** to set coordinate #0 of **y1** equal to 0.4;
Notice that if you are using the menu, you should now respond with **ok** twice by hitting <Enter> twice.

The routine **t** (trajectory) will plot the time (the number of the iterate) on the horizontal axis if the "plot time" toggle **pt** has to be turned on. See the **when and what to plot menu** for the current status of **pt**.

pm ← fetch the **parameter menu**, and then
wwm ← get the **when and what to plot menu**;
pt ← turn the **plot time toggle on**.
← retrieve the **when and what to plot menu**;
<Esc> ← get the **parameter menu**;
xs ← **0 100** ← set the x scale to run from 0 to 100;
i ← initialize;
t ← plot the trajectory
xax1 ← draw the x axis with tic marks;

The resulting picture is similar to Figure 2-3.

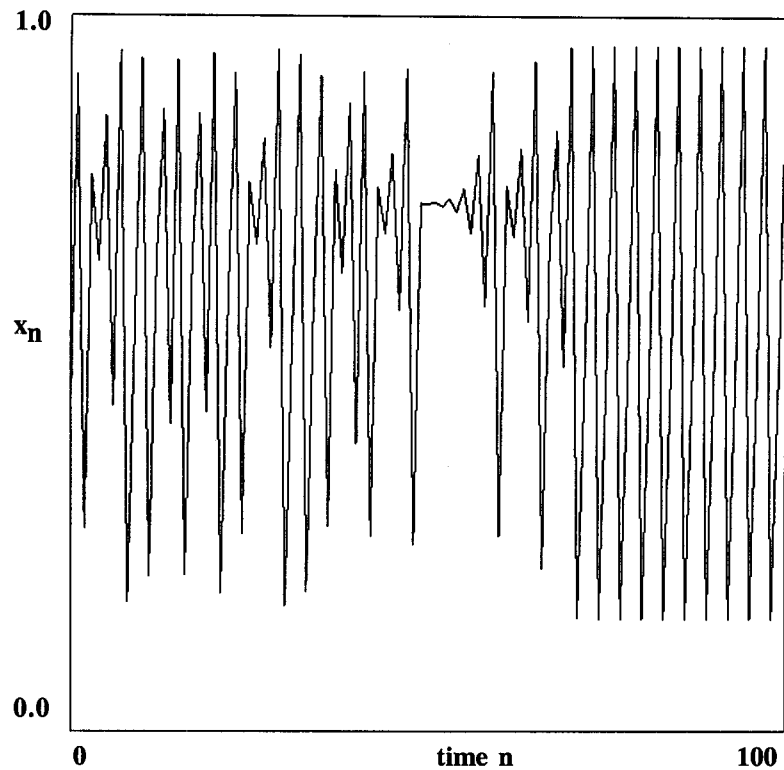


Figure 2-3: Trajectory versus time

This figure shows a trajectory versus time of the logistic map (log)

$$x \rightarrow 3.83 * x * (1 - x)$$

The initial condition of the trajectory is 0.4.

Topic of discussion. Using only this graph and the map, identify where there is a fixed point. What can you conclude about the map near the fixed point?

2.4 BIFURCATION DIAGRAM: plotting trajectory versus parameter

The purpose of this example is to plot a trajectory (iterates 61 through 260) versus a parameter of the logistic map (**log**)

$$x \rightarrow \text{rho} * x * (1 - x)$$

where *rho* varies from 2 to 4. For the minimum value of *rho* (*rho* = 2) and some initial condition x_0 , first calculate the first 60 iterates of x_0 without plotting (that is, calculate x_k with $1 \leq k \leq 60$ without plotting); we say the number of *pre-iterates* is 60. In *Dynamics* this number is called **bifurcation pre-iterates** (**bifpi** is listed in *bifm* and its default value is 60). Then plot the next 200 iterates of x_0 (that is, plot x_k with $61 \leq k \leq 260$). In *Dynamics* this number is called **bifurcation dots** (**bifd** is listed in *bifm* and its default value is 200). Increase *rho* slightly, say by 0.01 to rho_{new} . If the trajectory is not reinitialized at rho_{new} , we say *the attractor is followed* and take for the initial condition at the current value rho_{new} the last point that was plotted using the previous parameter value rho_{prev} . Then calculate 60 iterates of this point without plotting and then plot the next 200 iterates. Increase *rho* again, calculate 60 iterates of the last computed point at the previous parameter value without plotting and then plot the next 200 iterates. Continue increasing the parameter *rho* until it assumes the maximum value of 4.

Note. In *Dynamics*, when creating a bifurcation diagram, the attractor is followed if **bifi** is set to 0 (default setting) and the bifurcation parameter is increased by the value of **bifr** divided by the value of **bifv**.

```
dynamics-win ←| start the program Dynamics-Win;  
log ←| select the logistic map and the main menu appears;  
nem ←| fetch the numerical explorations menu, and then  
bifm ←| get the bifurcation diagram menu, and verify that  
pi = 60 and bifd = 200.
```

The command **bifr** in the **bifurcation diagram** menu is for specifying the range of the bifurcation parameter *rho*. The default setting is from **not set** to **not set**.

```
bifr ←| 2 4 ←| make the parameter rho vary from 2 to 4;  
←| retrieve the bifurcation diagram menu;  
bifs ←| plot the bifurcation diagram on the screen.
```

The resulting picture is called a **bifurcation diagram** and is similar to Figure 2-4.

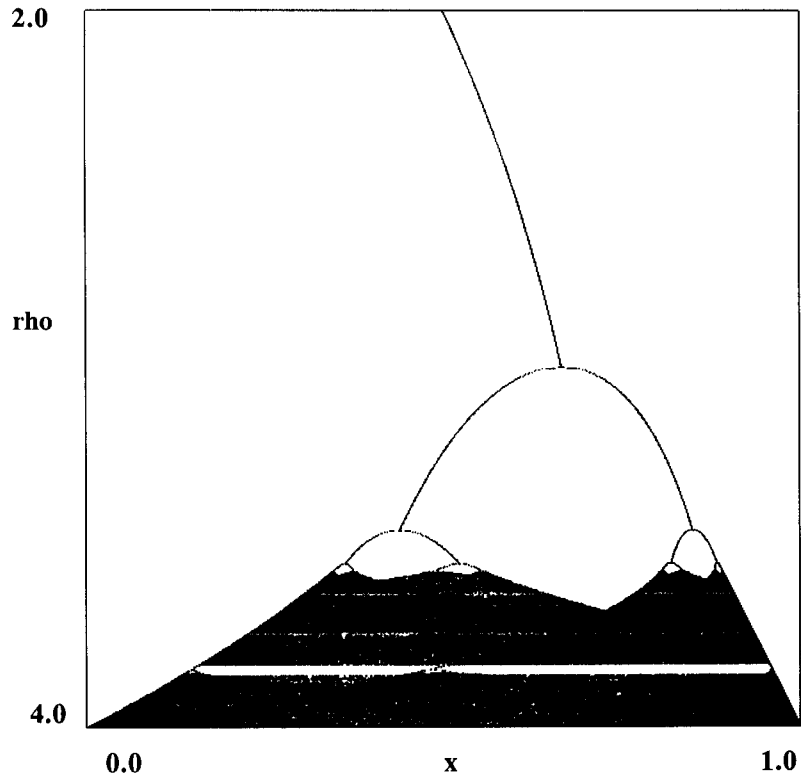


Figure 2-4: Bifurcation diagram created on the screen

In this bifurcation diagram for the logistic map (log)

$$x \rightarrow \text{rho} * x(1-x)$$

the parameter rho varies from 2.0 to 4.0, that is, the bifurcation range (bifr) is from 2.0 to 4.0. This bifurcation diagram is created on the screen using command bifs.

There are two different ways of plotting a bifurcation diagram. If the bifurcation diagram is sent directly to the printer (command bif), then you can set the length of the diagram and you have an option of printing the parameter values (bifp is on). If you create the bifurcation diagram on the screen, then you can set the size of the picture (commands high and wide in scm), but parameter values cannot be printed.

Frequently, in bifurcation diagrams the parameter is varied along the horizontal axis, while one of the space coordinates is on the vertical axis. You can obtain such a graphical representation by rotating the picture three times 90 degrees clockwise (command rot).

3. ADDING YOUR OWN PROCESS TO DYNAMICS

3.1 INTRODUCTION

Although there is a variety of maps and differential equations in the menus of *Dynamics*, you may want to study a process that is not in the menus. You can add your own processes to the program as follows.

We first describe how to add a map. Start the program as usual. Type

dynamics-win <Enter>

and the first menu that appears will have the line:

OWN - enter your OWN process

When you select this entry, four windows will appear on your screen, presenting an example of how to add a new process. Simply change what is in these four windows to define your process. When adding a process, the following keys are important for carrying out this task:

- hit <Esc> to cancel the option of adding a new process to *Dynamics*;
- hit <Tab> to go to the next window;
- hit <F1> to compile after you have inserted the new process;
- hit <F3> for adding a map to *Dynamics*;
- hit <F4> for adding a differential equation to *Dynamics*;

- use <BkSp> for erasing text in any of the windows;
- use <Ctrl-K> for erasing a line of text;
- use <Ctrl-W> for erasing all text in a window.

3.2a ADDING A NEW MAP

When you select the entry "own" from the map menu, four windows will appear on your screen, in approximately the following format, presenting an example of how to add a map to *Dynamics*. First some general comments.

- In each window, the '!' means that the rest of the line is a comment. Delete superfluous text using <BkSp>, <Ctrl-K> or <Ctrl-W>. *Note.* For deleting text, does not work.
- The equations are not case sensitive. Use either upper or lower case letters.
- The computer screen shows only 4-6 lines of each window. You may use more than the number of lines shown. However, if you use more lines, then a part of the information will be hidden whenever you switch to another window, but can be accessed with the arrow keys.
- In the windows you can have more than one equation per line or more than one line per equation. Temporary variables need not be initialized.

Documentation window
Henon map (x,y) --> (rho - x*x + c1*y, x)
Map window
x := rho - x*x + c1*y y := x
Initialization window
x := 0 y := 2 c1 := -0.3 rho := 2.12 x_lower := -2.5 x_upper := 2.5 y_lower := -2.5 y_upper := 2.5
Inverse map window
x := y y := (x - rho + y*y) / c1 ! Warning: do not use c1 = 0
Esc=Cancel Tab=Next F4=Diff Eqs F1=Compile Ctrl-K=EraseLine

Simply change what is in these four windows to define your map. Of course, there are some rules which are described later. We first describe the contents of the windows followed by some detailed examples.

Documentation window

The Documentation window provides text that will appear whenever you call either the **main menu** or the **parameter menu**. It will also be part of printed copies of pictures, when you print them using text level 2 or 3.

- Document your work!
Generally it is best to use four lines of text or less.
- All the text in the Documentation window will appear on the screen when the **main menu** or the **parameter menu** is called. It will also be part of printed copies of pictures, when you print them using text level 2 or 3.

Map window

The map that you want to add must be defined in the Map window.

- Allowed **process parameters**: $c_1, \dots, c_9, \text{phi}, \text{rho}, \text{sigma}, \text{beta}$.
- Allowed **main variables**: r, s, t, \dots, z . The value of the main variables that are used are inserted into $y[0], y[1]$, etc. You can find their values using the command **yv**. The program makes a list in **alphabetical order** of the r, s, t, \dots, z variables you use and assigns them to $y[0], y[1], y[2], \dots$. If you use $r, x,$ and y (listing them alphabetically), their values will be inserted into $y[0], y[1]$, and $y[2]$, respectively.
- Allowed **phase space variables**: u, \dots, z . These variables have a special role. When the quasi-Newton method is used, for example, numerical partial derivatives are computed with respect to these variables, and the Lyapunov exponents indicate how fast these variables move apart. In practice, these six variables will be the only ones used for maps, unless you use $r, s,$ and t for plotting. You might, for example, define r and s to be variables that give a better plot.
- Each main variable used must appear on the left side of an equation exactly once in the Map window.

Initialization window

The initial values that are listed in the Initialization window alert the program to the elements used in your process. Notice that you can have more than one equation per line.

- All the main variables and parameters that appear in the Map window are initialized with value 0 unless you initialize them here.

- C1,..., C9, phi, rho, sigma, beta are the only parameters permitted in equations. Their values will appear in the **parameter menu pm**.

- As described above, the program makes a list in alphabetical order of the variables you use and assigns to them y[0], y[1], y[2], If you use two or more phase space variables, then the first two phase space variables (when listed alphabetically) are assigned to **xco** and **yco**.

If you use only w and z, then w corresponds to y[0] and z to y[1]. If you wish to plot w horizontally and z vertically, then the Initialization window would include

```
xco := 0          ! w is variable # 0
yco := 1          ! z is variable # 1
```

Actually the above assignment is automatic and does not have to be made explicitly.

If you use x, y and z, then x corresponds to y[0], y to y[1] and z to y[2]. If you wish to plot y horizontally and z vertically, then the Initialization window would include

```
xco := 1          ! y is variable # 1
yco := 2          ! z is variable # 2
```

- The default scale both horizontally and vertically is 0 to 1. If you wish to change one of these, then as in the example above define the variables

x_lower x_upper and/or y_lower y_upper

Their values will appear in the **parameter menu pm**.

- You may even wish to define **z_lower z_upper** and **zco** if you plan use three dimensional plots.
- Anything defined in this window can be changed by using standard commands of the program. For example, if you choose the defaults, then you can redefine the scale using the **x scale** and **y scale** commands **xs** and **ys**, which are in the **parameter menu pm**.

Inverse map window

Using the Inverse map window is optional. The inverse is used in calculating stable manifolds and in going backwards in time for **de**'s (see the inverse command **v**). If you want to make use of the inverse (provided it exists) then you must define the inverse in the Inverse map window.

- The program does not check to verify that the map you provide is in fact the inverse. See command **v** (inverse map).

No text in the Initialization window for maps

If you erase all text in the Initialization window of the Henon map, then after you have compiled this process, the parameter menu is of approximately the following format

```
PM -- PARAMETER MENU

OK - parameters are fine as set
XCO - X COordinate to be plotted: y[0]
XS - X Scale: from 0 to 1
YCO - Y COordinate to be plotted: y[1]
YS - Y Scale: from 0 to 1
C1 - C1 = 0.00000000
RHO - rho = 0.00000000
SD - Screen Diameters: 1.00

MENUMS
VM - Vector Menu for initializations, etc.
WWM - When and What to plot Menu
```

The Y Vectors are set as follows:

```
YV -- Y VECTORS

Vectors -- y0 = y = current state, y1 = cursor position
state vec y      storage vec y1      storage vec y2
y[0] = 0;        y1[0] = 0;        y2[0] = NOT SET
y[1] = 0;        y1[1] = 0;        y2[1] = NOT SET

storage vec ya    storage vec yb    storage vec ye
ya[0] = 0;        yb[0] = 1;        ye[0] = NOT SET
ya[1] = 0;        yb[1] = 1;        ye[1] = NOT SET
```

For the case of the Henon map and other maps, you can set parameters like *c1* and *rho*, coordinates *xco* and *yco*, and scales *xs* and *ys* as you wish. You also may set the vectors. Hence, you may leave the Initialization window empty.

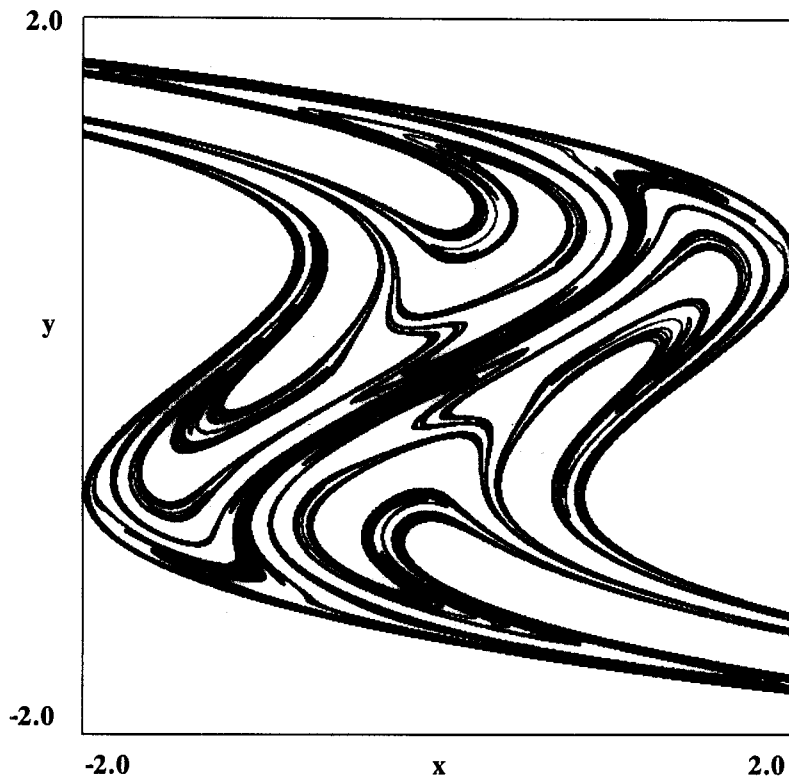


Figure 3-1: Unstable manifold of a fixed point

The stable manifold of a fixed point p (equilibrium point p) of a continuous, invertible map F is the collection of points whose trajectories converge to p under forward iteration of the map F ; the unstable manifold of p is the collection of points whose trajectories converge to p under backward iteration of the map F . This figure shows the unstable manifold of the fixed point $(0,0)$ of the Own-Holmes map

$$(x,y) \rightarrow (1.5*x - x*x*x + 0.95*y, x)$$

You may plot also the stable manifold of the fixed point $(0,0)$. If you do so, the intersection of the unstable manifold and the stable manifold is a complicated set which is an example of a so-called fractal.

Example 3-1. Adding the Holmes map to *Dynamics*

The objective of this example is to show how to add the Holmes map, a cubic map in the plane,

$$(x,y) \rightarrow (\rho*x - x*x*x + c1*y, x)$$

to *Dynamics* using the *own*-feature. For our convenience, we use "Own-Holmes map" in the Documentation window. See also Section 4.5, entitled "Storing data and pictures of an *own*-process". Start *Dynamics* and select *own*, and add the equations as indicated below.

Documentation window
Own-Holmes map (x,y) --> (rho*x - x*x*x + c1*y, x)
Map window
x := rho*x - x*x*x + c1*y y := x
Initialization window
c1 := 0.95 rho := 1.5 x_lower := -2.0 x_upper := 2.0 y_lower := -2.0 y_upper := 2.0
Inverse map window
x := y y := (x - rho*y + y*y*y) / c1 ! Warning: do not use c1 = 0
Esc=Cancel Tab=Next F4=Diff Eqs F1=Compile Ctrl-K=EraseLine

After completing the typing, hit <F1> to compile.

Plot the unstable manifold of the fixed point (0,0) (menu *um*, commands *ul*, *ur*). The resulting picture is shown in Figure 3-1. To store the picture on disk, we recommend to have the file easily recognized as being a picture file created by a process that has been added to *Dynamics* by the *own* feature. For example, store this picture under the file name *o-Ho-um.pic*.

Note. As discussed on page 35, you may leave the Initialization window empty. Then $c1 = \rho = 0 = y1[0] = y1[1]$ and scales are from 0 to 1.

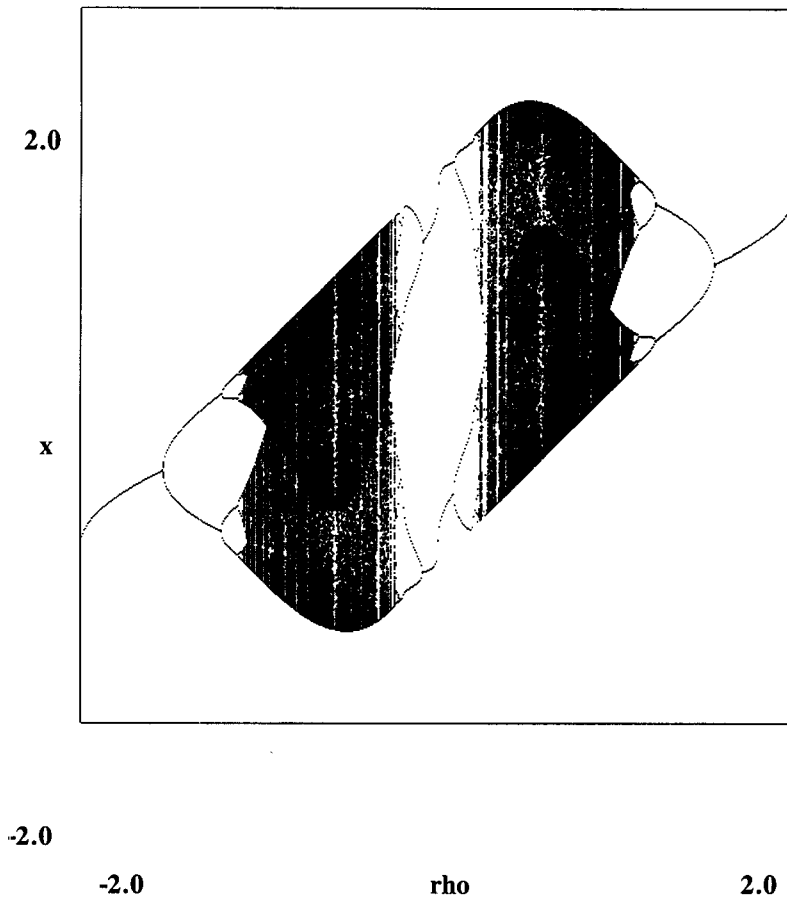


Figure 3-2: Bifurcation diagram created on the screen and rotated
This bifurcation diagram for the cubic map (Own-Cubic)

$$x \rightarrow x*x*x - 2*x + rho$$

where rho varies from -2 to 2. This picture is similar to the one on the screen, but is rotated. The picture is 720 by 720. Square picture can be rotated, while pictures of 640 by 480 (size of VGA screen) cannot be rotated.

To create this picture, enter command `bifr` followed by `-2 2`, and then enter command `bifs`. You may wish to set `bifv` to 720, but is not necessary.

After the diagram was created on the screen, the command `rot` was invoked three times resulting in a rotation of 90 degrees counter-clockwise. Each time when the command `rot` is executed, it will rotate the picture 90 degrees clockwise.

Example 3-2. Adding a one-dimensional map to *Dynamics*

The purpose of this example is showing how to add a one-dimensional real cubic map

$$x \rightarrow x*x*x + c1*x + rho$$

to *Dynamics* using the *own*-feature. For our convenience, we use "Own-RealCubic map" in the Documentation window. See also Section 4.5, entitled "Storing data and pictures of an *own*-process". Start *Dynamics* and select *own*, and add the equations as indicated below.

Documentation window
Own-RealCubic map x --> x*x*x + c1*x + rho Plot x at time n (horizontally) against x at time n+1 (vertically)
Map window
x := x*x*x + c1*x + rho
Initialization window
x := -1.0 c1 := -2.0 rho := 1.0 x_lower := -2.0 x_upper := 2.0
Inverse map window
Esc=Cancel Tab=Next F4=Diff Eqs F1=Compile Ctrl-K=EraseLine

After completing the typing, hit <F1> to compile.

Plot the bifurcation diagram (menu *bifm*, commands *bifr* (-2,2), *bifs*). A variant of the resulting picture is shown in Figure 3-2. You may store this picture (created by a process that was added to *Dynamics* by the *own* feature), for example, under the file name *o-c-bifs.pic*.

Note. Since there is one phase space variable, *Dynamics* will plot *x* vertically and horizontally it will plot *x* at the previous time, that is, at time *n* it plots (x_{n-1}, x_n) where x_n denotes the value of *x* at time *n*.

Note. As discussed on page 35, you may leave the Initialization window empty. Then $c1 = rho = 0 = y1 = y0$ and the scales are from 0 to 1.

3.2b BASIC RULES FOR ADDING A NEW MAP TO *DYNAMICS*

Simply change what is in these four windows to define your map. Of course, there are some rules. Standard algebraic expressions are allowed.

- Use * to denote multiplication
- Arithmetic operators (* / + -) have their conventional precedence.
- Expressions can use the usual algebraic expressions as well

the number pi (π)

and the following functions:

sin, cos, tan, atan, sqrt, exp, mod,

log (log base e), log10 (log base 10), rand

as well as

absolute value $|E|$ of an expression E and

power E^G , where either E is positive or G is an integer.

- If you are in doubt as to whether you need parentheses in an expression, they do no harm so they can be added.
- First, the values of the main variables (r, s, ..., z) are first inserted into the right hand side of all the equations, and then the equations are evaluated sequentially. Consider the case where x and y both have initial values of 0.0 and the process is defined by

$$\begin{aligned}x &:= y + 1 \\ y &:= x\end{aligned}$$

First, the value 0.0 is inserted for x and y in the right hand side of both equations. The x in the second equation is the initial value. Hence after one iterate, x is 1 and y is 0. The sequential evaluation is only important if temporary variables are used.

random number rand()

The function **rand()** selects a pseudo-random number between -1 and 1. For example, `temp := rand()`. Notice that **rand()** is a function with no arguments!

3.3a ADDING A NEW DIFFERENTIAL EQUATION

Equation entry can be in one of two modes, called Map Mode or Differential Equation Mode. Each accepts three windows of inputs in addition to the Documentation window. In Map Mode these are called the map, inverse map, and initialization. In DE Mode these are called vector field, modulo, and initialization.

After selecting **own**, you will be in the Map Mode. To switch to the Differential Equation, press <F4> and four windows will appear on your screen, in approximately the following format, presenting an example of how to add a differential equation to *Dynamics*. In each of the four windows, the '!' means that the rest of the line is a comment.

Documentation window
forced damped Pendulum $x'' + c1*x' + c2*\sin(x) = \text{rho}*(c3 + \cos(\text{phi}*t))$
Vector field window
$s' := 1$! this is time $t' := 1$! this is time mod $2*\text{pi}/\text{phi}$ (see Modulo window) $x' := y$ $y' := \text{rho}*(c3 + \cos(\text{phi}*t)) - c1*y - c2*\sin(x)$
Initialization window
$t := 0$ $x := 1$ $y := 0$ $x_upper := \text{pi}$ $x_lower := -\text{pi}$ $y_upper := 4$ $y_lower := -2$ $c1 := 0.2$ $c2 := 1.0$ $c3 := 0.0$ $\text{rho} := 2.5$ $\text{phi} := 1$ $\text{spc} := 30$ $\text{ipp} := 30$! Take 30 steps per $2*\text{pi}/\text{phi}$ and ! plot once in 30 steps
Modulo window
$t := \text{mod}(t, 2*\text{pi})$ $x := \text{mod}(x, -\text{pi}, \text{pi})$
Esc=Cancel Tab=Next F4=Diff Eqs F1=Compile Ctrl-K=EraseLine

Using the pendulum process P to create a basins picture is slightly faster on PC's than the corresponding picture made with *Own*. For most purposes however, the difference in speed is less important than the convenience of getting the program running. Simply change what is in these four windows to define your map. Of course, there are some rules which are described later. We first describe the contents of the windows followed by some detailed examples.

The four windows follow essentially the same rules as for maps, and use the same variables and functions and parameters. See Section 3.2a for details.

Vector field window

Differential equations to be added are written in the Vector field window.

Initialization window

The step sized used by the differential equation solver is specified in this window.

Modulo window

Using the Modulo window is optional. The equation(s) in this window are applied after each time step of the differential equation solver.

Text in the Initialization window for differential equations

For maps it has been discussed that you are allowed not to enter anything in the Initialization window. All the parameters are set to be zero and the scales run from 0 to 1. The same holds for differential equations like the Lotka/Volterra equations. In addition to these settings of parameters, the step size is set to be 0.01.

For the forced damped pendulum and other periodically forced differential equations, it is required that the

```
spc := 30
```

is included. However, it may be better to include "ipp := 30". Since the forced damped Pendulum equation is in fact acting on a cylinder, you should also include "x_upper := pi x_lower := -pi"

Therefore, the minimal contents of the Initialization window for the forced damped pendulum is the following:

```
Initialization window
x_upper := pi  x_lower := -pi
spc := 30  ipp := 30  ! Take 30 steps per 2*pi/phi and
! plot once in 30 steps
```

Example 3-3. Adding the Lotka/Volterra eqs. to *Dynamics*

In this example we add the Lotka/Volterra eqs.

$$\begin{aligned}x' &= (c1 + c3*x + c5*y)*x \\y' &= (c2 + c4*y + c6*x)*y\end{aligned}$$

to *Dynamics* using the *own*-feature. Although the Lotka/Volterra equations are a process of *Dynamics*, we will carry out this example. For our convenience, we use "Own-Lotka/Volterra DE" in the Documentation window. See also Section 4.4 for storing data and pictures. Start *Dynamics* and select *own*, hit <F4> and add the equations as indicated.

Documentation window
Own-Lotka/Volterra DE $x' = (c1 + c3*x + c5*y)*x$ $y' = (c2 + c4*y + c6*x)*y$
Vector field window
$t' := 1$! this is time $x' := (c1 + c3*x + c5*y)*x$ $y' := (c2 + c4*y + c6*x)*y$
Initialization window
step := 0.01 x_lower := 0 x_upper := 1000 y_lower := 0 y_upper := 1000 x := 10 y := 100 c1 := 0.5 c3 := -0.0005 c5 := -0.00025 c2 := 0.5 c4 := -0.0005 c6 := -0.00025
Modulo window
Esc=Cancel Tab=Next F4=Diff Eqs F1=Compile Ctrl-K=EraseLine

After completing the typing, hit <F1> to compile. The differential equation solver will use the "step" that is specified in the Initialization window. You may change its value, which is listed in the menu **dem**.

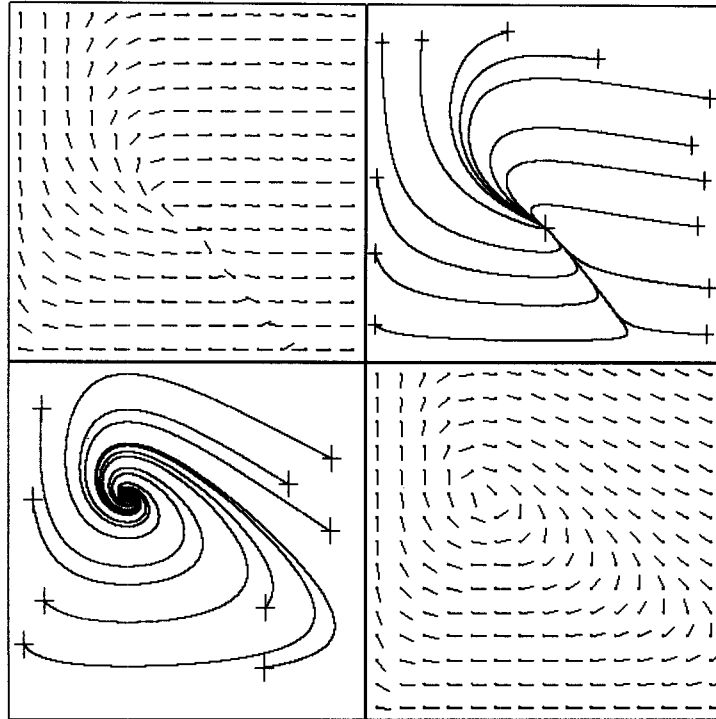


Figure 3-3: Direction field and trajectories of Lotka/Volterra equations
In all the pictures of this figure the direction field and some trajectories are shown for the Lotka/Volterra differential equations

$$\begin{aligned}x' &= (c_1 + c_3*x + c_5*y)*x \\y' &= (c_2 + c_4*y + c_6*x)*y\end{aligned}$$

where x is plotted horizontally and y is plotted vertically. The initial value of each trajectory is indicated with a small cross; a larger cross indicates an equilibrium. Set `con` to be on or set `step = 0.001`.

In the upper windows, $c_1 = 0.5$, $c_2 = -0.1$, $c_3 = -0.000625$, $c_4 = 0$, $c_5 = -0.01$, and $c_6 = 0.0002$. The x scale runs from 0 (left) to 1000 (right), and the y scale runs from 0 (bottom) to 50 (top).

In the lower windows, $c_1 = 0.5$, $c_2 = -0.1$, $c_3 = -0.00025$, $c_4 = 0$, $c_5 = -0.01$, and $c_6 = 0.0002$. The x scale runs from 0 (left) to 1500 (right), and the y scale runs from 0 (bottom) to 60 (top).

Example 3-4. Adding a SimBlum differential eq. to *Dynamics*

In this example we add the SimBlum eq.

$$x'(w) = c1*x*x + c2*w*w + c3*w*x$$

to *Dynamics* using the *own*-feature. For our convenience, we use "Own-SimBlum DE" in the Documentation window. See also Section 4.5 for storing data and pictures. Start *Dynamics* and select *own*, hit <F4> and add the equations as indicated.

Documentation window
Own-SimBlum DE $x'(w) = c1*x*x + c2*w*w + c3*w*x$
Vector field window
$w' := 1$! this is time $x' := c1*x*x + c2*w*w + c3*w*x$
Initialization window
$w := 0$ $x := 1$ $x_lower := -3$ $x_upper := 3$ $y_lower := -3$ $y_upper := 3$ $c1 := 1.0$ $c2 := 1.0$ $c3 := 0.0$ $step := 0.0001$
Modulo window
Esc=Cancel Tab=Next F4=Diff Eqs F1=Compile Ctrl-K=EraseLine

After completing the typing, hit <F1> to compile. Enter command *df*, and the resulting picture is shown in Figure 3-4. The differential equation solver will use the "step" that is specified in the Initialization window. You may change its value, which is listed in the menu *dem*.

Note. If you prefer to use the variable 't' instead of 'w' in the differential equation and the windows, then you must set the x coordinate to be y[0] after you have tapped <F1>, that is, enter

xco ← 0 ←

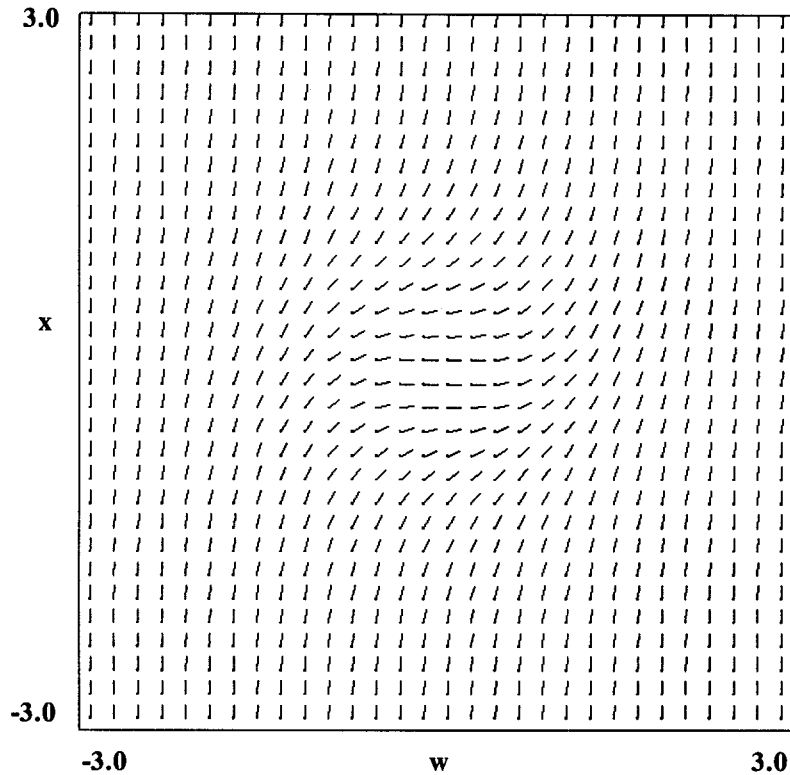


Figure 3-4: Direction field

This figure shows a direction field for the SimBlum differential equation

$$x'(w) = c1*x*x + c2*w*w + c3*w*x$$

where $c1 = 1.0$, $c2 = 1.0$, $c3 = 0.0$

*The grid size of the direction field is set to 30 by 30 (command **fg**).*

Topic of discussion. *This figure suggests almost all solutions diverge to infinity. Which trajectories, if any, are bounded?*

To plot 10 trajectories simultaneously that start on the line segment $ya = (-3,-3)$ to $yb = (3,-3)$, carry out the following steps (step = 0.0001):

```

tn ← 10 ← plot 10 trajectories simultaneously if t is entered;
diag ← set ya and yb to the opposite corners of the screen,
that is,  $ya = (-3,-3)$  and  $yb = (3,3)$ ;
svb1 ← -3 ← set coordinate #1 of yb equal to -3;
t ← ← plot the 10 trajectories starting at  $ya-yb$ .

```

Example 3-5. Adding the logistic differential eq. to Dynamics

In this example we add the logistic eq.

$$x' = \text{rho} * x * (c1 - x)$$

to *Dynamics* using the *own*-feature. For our convenience, we use "Own-logistic DE" in the Documentation window. See also Section 4.4 for storing data and pictures. Start *Dynamics* and select *own*, hit <F4> and add the equations as indicated.

Documentation window
Own-logistic DE $x' = \text{rho} * x * (c1 - x)$
Vector field window
$w' := 1$! this is time $x' := \text{rho} * x * (c1 - x)$
Initialization window
$w := 0$ $x := 1$ $x_lower := 0$ $x_upper := 3$ $y_lower := -2$ $y_upper := 8$ $c1 := 5.0$ $\text{rho} := 0.5$ $\text{step} := 0.0001$
Modulo window
Esc=Cancel Tab=Next F4=Diff Eqs F1=Compile Ctrl-K=EraseLine

After completing the typing, hit <F1> to compile. Enter command **df**, and a resulting picture is shown in Figure 3-5a. The differential equation solver will use the "step" that is specified in the Initialization window. You may change its value, which is listed in the menu *dem*.

Note. If you prefer to use the variable 't' instead of 'w' in the differential equation and the windows, then you must set the x coordinate to be y[0] after you have tapped <F1>, that is, enter

$xco \leftarrow 0 \leftarrow$

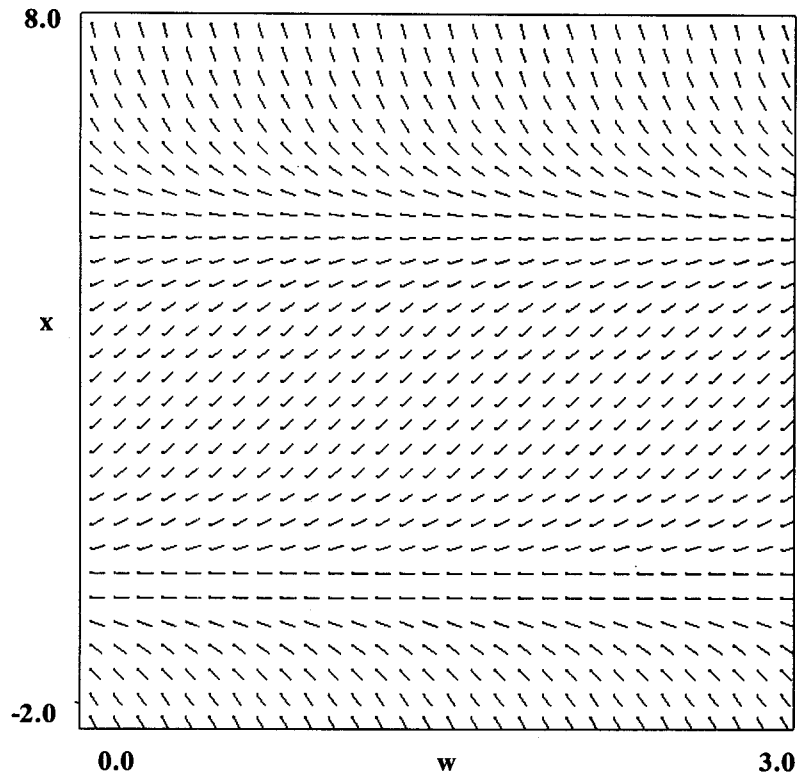


Figure 3-5a: Direction field

This figure shows a direction field for the logistic differential equation

$$x' = \text{rho} * x * (c1 - x)$$

where $c1 = 5.0$, $\text{rho} = 0.5$

The grid size of the direction field is set to 30 by 30 (command `fg`).

Topic of discussion. this figure suggests there are two equilibria. What are they and which of them is asymptotically stable and which one is unstable? Explain why.

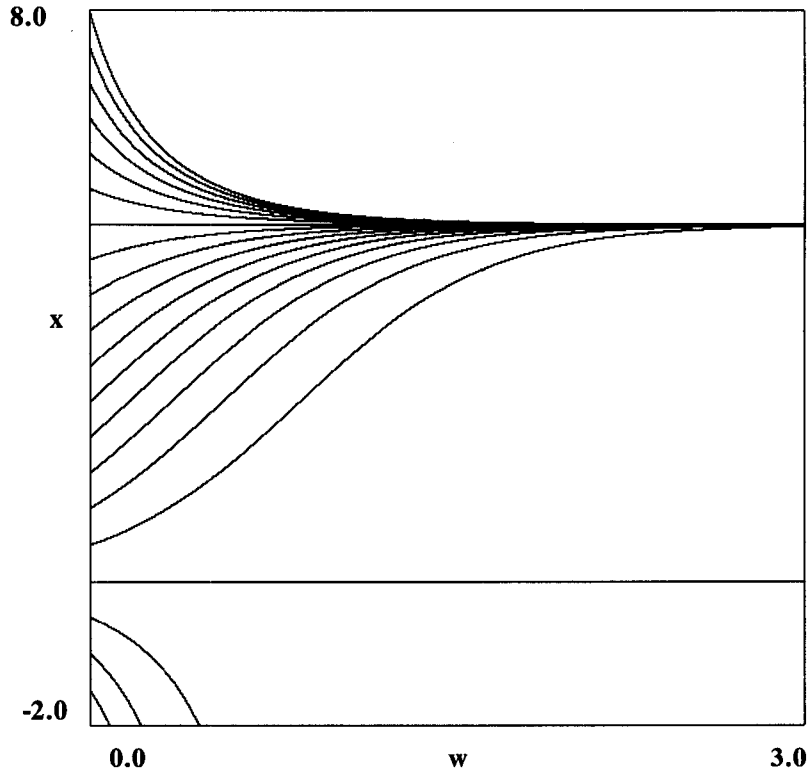


Figure 3-5b: Trajectories plotted simultaneously

This figure shows trajectories for the logistic differential equation

$$x' = \text{rho} * x * (c1 - x)$$

where $c1 = 5.0$, $\text{rho} = 0.5$ that are plotted simultaneously.

Topic of discussion. This figure suggests there are two equilibria. What are they and which of them is asymptotically stable and which one is unstable? Explain why.

*To plot 21 trajectories simultaneously that start on the line segment $y_a = (0, -2)$ to $y_b = (0, 8)$, carry out the following steps (**step** = 0.0001):*

tn ←	21 ←	<i>plot 21 trajectories simultaneously if t is entered;</i>
diag ←		<i>set y_a and y_b to the opposite corners of the screen,</i>
		<i>that is, $y_a = (0, -2)$ and $y_b = (3, 8)$;</i>
svb0 ←	0 ←	<i>set coordinate #0 of y_b equal to 0;</i>
t ←		<i>plot the 10 trajectories starting at y_a-y_b.</i>

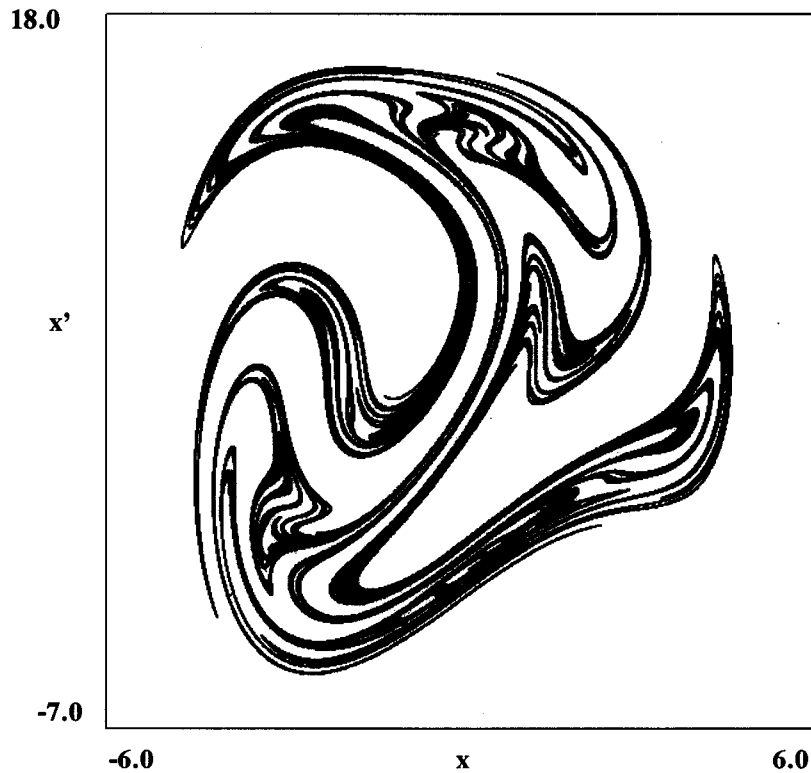


Figure 3-6: A strange attractor

This figure shows the attractor of the time- 2π map of the Goodwin differential equation (gn)

$$x'' + c_1 \left(\frac{x^2 x - 1}{x^2 x + 1} \right) x' - c_2 x + c_3 x^2 x^2 = \rho \sin(\phi t)$$

The parameter values are $c_1 = 0.1$, $c_2 = c_3 = 0.5$, $\phi = 1$, and $\rho = 37$.

This attractor is called a "strange attractor" because it appears to have infinitely many layers of curves. Each blow-up of the attractor reveals curves that split into multiple curves.

3.3b BASIC RULES FOR ADDING A DIFFERENTIAL EQ. TO *DYNAMICS*

The rules are essentially the same as for maps; see Section 3.2b.

In addition:

- For each main variable used, there must be precisely one differential eq.

Temporary variables

In writing equations, you can also use temporary variables. They can have any name beginning with a letter using up to 8 letters and digits, except the names cannot be the names of parameters c_1, \dots, c_9 or main variables r, s, t, \dots, z or function names \sin, \exp, \dots or reserved words $\pi, \text{AND}, \text{NOT}, \dots$ mentioned earlier in this chapter. Each temporary variable must be defined in each window in which it is used, so "temp := temp+1" can not be the first line of a window.

Special parameter phi

Adding a differential equation is quite similar to the process for adding a map. Here the parameter ϕ has a special role and should be used only when there is a time dependent periodic forcing. It appears in arguments of periodic forcing functions. The forced damped Pendulum and the Goodwin equation are examples of these type.

The program knows that a differential equation is going to be used when it finds the differential equation step size has been set, that is the variable "step" (for example, $\text{step} = .01$). The Lotka/Volterra equations is an example of this type. See also "Rules for periodically forced differential equations" below.

Rules for periodically forced differential equations

The pendulum example $X'' + c_1 X' + c_2 \sin(X) = \rho(c_3 + \cos(\phi T))$ above has a time periodic term $\rho(c_3 + \cos(\phi T))$ in its equation. For such examples, it is common to plot points once every period. The program allows such plotting only if the variable ϕ is defined. The period is assumed to be $2\pi/\phi$. Hence even if you have a fixed period in your example, ϕ should be defined.

In the Initialization window you may also assign `steps_per_cycle spc`. If ϕ is defined and `spc` is not, it is given the default values 50, and the time step `step` is given the value `step = 2*pi/(phi*spc)` for differential equations. The default values are .01 and 50. If you set `phi`, it should be greater than 0.

Another variable that can be set in the Initialization window is `ipp`. It has a default value of 1. If you wish the trajectory to plotted once each period, give it the same value you give to `phi`. Do **not** write `ipp = phi` because `phi` does not have a value in the Initialization window.

4. STORING AND PRINTING PICTURES THAT ARE CREATED BY DYNAMICS

The Disk Menu displays an overview of using the disk. Enter the command **dm**, and the **Disk Menu** will appear on the screen. It includes

DM	--	DISK MENU
RN	-	Root Name (to change root h)
DD	-	Dump Data to disk file h.dd
PCX	-	copy picture to PCX file h.pcx readable by MS Windows
REC	-	RECORD all keystrokes in file h.rec: OFF
TD	-	copy picture and parameters To Disk file h.pic
Commands DD and TD save current data/picture in h.dd or h.pic. Data can be later restored by starting the program by DYNAMICS-Win h.dd <Enter> or DYNAMICS-Win h.pic <Enter>		

4.1 STORING PICTURES and DATA

Root name of the picture (*command rn*)

Pictures and data can be stored on disk with all current parameter values. The root name must be permissible by the operating system. *Dynamics* allows a root name up to 8 letters, then a period followed by up to 3 letters of an extension provided by *Dynamics*.

If you want to store a picture on the disk, you have to specify the root name or use the default name. The "Root Name" command **rn** is for changing the disk file's root name. You provide the root name and *Dynamics* provides the extension. You can precede the name by the disk designation. For example, you may use a root name like "A:h1" and the program will use A:h1.pic or A:h1.dd or A:h1.prn depending on what is in the file. Substitute for "A:" above the appropriate disk drive. If the picture is in the current disk drive directory, no drive designation is necessary.

Sending the picture to disk

The "to disk" command **td** copies the current picture to a disk file. If the root name is "h1", then the resulting file name of the picture is "h1.pic", where ".pic" is added by *Dynamics* to denote the file contains a picture. (Use command **rn** for the root name of the file.) The picture also remains in the program. The disk file includes almost all current values of the parameters and is not actually created unless you use the command **td**.

Warning. If a file with that name already exists, the program will write over the old file, destroying the old one.

If you have created a file named h1.pic, the only way to get *Dynamics* to access the parameter settings in this file is to use the file to start the program from the Run command line using the command

dynamics-win h1.pic <Enter>

This resets the parameters and vectors to their previous value according to the values in the file. See also the commands **rn** (root name), **dd** (dump data) and **fd** (from disk). You can use commands **fd** and **afd** (add from disk) to retrieve the picture while the program is running.

The command **pcx** instructs the program to create a file called "root.pcx", a copy of the current picture, in **pcx** format. This format is used by Microsoft Windows and Microsoft Paintbrush. Windows can then be used for printing the picture.

Sending data to disk

The "dump data" command **dd** creates a batch file that contains the process name and the values of most constants and vectors. If you used "Own" to create a map or differential equation, it is also included. The file is just like the one created using command **td** (to disk) except the picture is not saved. To use such a file, you must not be using *Dynamics*. This dump data file is a batch file for starting *Dynamics*. Ordinarily you would start the program from the Run command line with the command:

dynamics-win <Enter>

Now if you have created a data dump file named **lorenz.dd** the program can be started from the Run command line using the command

dynamics-win lorenz.dd <Enter>

This resets the parameters and vectors to their previous value according to the values in the file. The picture is **not** contained in this file. See also the commands **rn** (root name) and **fd** (from disk).

The "record" command **rec** is a toggle, and can be invoked before or after a process is selected. When **rec** is turned *on*, it instructs the program to store all the keystrokes in a file. The file name ends in ".rec" and is printed in the **disk menu dm**.

Reading pictures from disk

Enter command **rdm** and the **Read Disk files Menu** appears. It includes

RDM	--	READ DISK FILES MENU
AFD	-	Add picture From Disk file h.pic into core picture
FD	-	read picture From the Disk file h.pic

The "add from disk" command **afd** adds the picture from disk file into core picture, so that the two pictures are overlaid. You will be prompted for the name of the file.

The disk file copy remains on the disk unaltered. This command enables you to create a number of files with different parts of a picture, and then add them together.

The "from disk" command **fd** makes the program read the picture from the disk into core memory. You will be prompted for the name of the file. If the file name is set to be **A:h.pic** for example, then the program expects to find the file on the disk in the "A-drive", and it will not look elsewhere.

Invoking **fd** does not change any parameters of the current process.

4.2 SETTING THE SIZE OF THE CORE PICTURES

If you enter the command **scm**, the **Size of Core Menu** appears on the screen.

```
SCM  --  SIZE OF CORE MENU

Changing Size of Core destroys current picture
(Core copy of a picture is what gets printed)

SQ5   -  core picture 512 pixels wide by 512 high
SQ7   -  core picture 720 pixels wide by 720 high
HIGH  -  core picture is 720 dots HIGH
WIDE  -  core picture is 720 dots WIDE
```

Commands for changing the core size

HIGH

The command **high** sets the height (in pixels) of the internal core picture. The default value is 720.

Changing the core size destroys the current picture.

WIDE

The command **wide** sets the number of columns of dots in the core picture. That is, it sets the width (in pixels) of the internal core picture. The default value is 720. Hewlett-Packard printers will not accept a value greater than 960.

Changing the core size destroys the current picture.

SQ5

Command **sq5** makes the core picture 512 pixels high by 512 pixels wide. Changing the core size destroys the current picture.

SQ7

Command **sq7** makes the core picture 720 dots high and 720 dots wide. Changing the core size destroys the current picture.

4.3 PRINTING PICTURES

To print a picture, we recommend <Alt>-<PrtSc> to copy the *Dynamics* screen to the clipboard. Then after opening a program like MS Word, <Ctrl>-<V> pastes the picture into the current document. It can be printed using the program's print commands.

Before creating the picture, you may wish to apply the command WHI (white) which sets the background to color white, which often creates pictures that are better for printing.

4.4 STORING DATA AND PICTURES OF AN OWN-PROCESS

After tapping <F1> and successfully entering the stage for iteration, you can store your equations using the **dump data** command **dd** or the **to disk** **td** (which stores the picture as well as the data); see also Section 4.1.

Root name

If you have created your own process using *Dynamics*'s **own** feature, the default root name is "own". Hence, if you save data using command **dd**, then the data is saved in file **own.dd**. If you store a picture that has been created by this process, then it is stored in file **own.pic**.

We recommend to change, for example, the root name of the Holmes map to **o-ho**, and the root name of the GoodwiN equation to **o-gn**. Of course, you may want variants for these. As an example, the picture that has been created contains, for example, a trajectory of the Holmes map, can be stored in the file "o-ho.pic", but a file name like "o-ho-t.pic" might be better to indicate that the picture is a trajectory for the Holmes map that was created by the own-feature of *Dynamics*.

Recommendation. We recommend to save the data of an *own*-process immediately after you have compiled, for example, in the file named *own-sb.dd*. After exiting *Dynamics*, you can get access to these data by the command **dynamics own-sb.dd**. Hit <F1>, and continue working with this process.

Similarly, if you have saved a picture with file name "*own-sb.pic*", you can retrieve this picture by the command **dynamics own-sb.pic**. Hit <F1>, and the picture appears on the screen.

Reference

H.E. Nusse and J.A. Yorke, *Dynamics: Numerical Explorations*, Second, Revised and Expanded Edition, 608 + xvi pp., accompanied by software program *Dynamics* (Version 2), an interactive program for IBM compatible PC's and Unix computers (the Unix version of the program is by B.R. Hunt and E.J. Kostelich), Springer-Verlag, New York, 1998