DYNAMICAL SYSTEMS
Tutorial 17

Iterated Maps

```
sysid
```

```
Mathematica  6.0.3, DynPac 11.01, 1/13/2009
```

```
plotreset;
```

```
intreset;
```

▪ Functions and Variables Used in This Tutorial

asprat, axon, bifurcmap, bifurc3Dmap, bimap, boxrat, classifymap, cobweb, colorvec, eigsysmap, eigvalmap, findpolyfix, frameon, imsize, intreset, iterate, jacob, jacobval, mapcomp, mapval, nfindfix, nfindpolyfix, outbound, parmval, parmvec, periodmap, phaser, phaser3D, plotreset, plrange, plrange3D, pointcon, portraitmap, portrait3Dmap, ptsize, rangeflag, ranger, residualfix, setback, setcolor, setde, setmap, setparm, setstate, show, slopevec, staterange, statevec, stripsol, sysid, sysreport, timeplot, and viewmap.

▪ Description of Systems Used in This Tutorial

In this tutorial, our objective is to illustrate the use of the functions defined for iterated mappings. For examples, we will use the logistic map for a 1D case, the Henon map for a 2D case, and a combination of the two for a 3D case. In many cases, it is useful to apply some of the functions directly to iterates of the map. For example, if we are studying a map f[x], then one way of finding orbits of period two is to look for fixed points of f[f[x]]. Most of the functions used in DynPac for mappings allow an optional final argument which is the level of composition desired. We will see a number of examples of this below.

▪ Logistic Map

The logistic map is discussed in many references. A very complete and readable discussion is given in Chapter 10 of **Nonlinear Dynamics and Chaos** by Steven Strogatz, Addison-Wesley, 1994. Many of the interesting properties of the map were discovered by the mathematical biologist Robert May ("Simple Mathematical Models with Very Complicated Dynamics," Nature **261**, 459, 1976.) The basic form of the map is

$$x_{n+1} = r x_n (1 - x_n) \quad .$$

As is well-known this map exhibits a wide and interesting range of behavior as r is varied. We define the system for

DynPac, starting by the setmap command. This command tells DynPac that we are working with a mapping rather than a differential equation.

```
setmap;
```

```
setstate[{x}]; setparm[{r}]; parmval = {3.2}; slopevec = {r * x * (1 - x)};
```

```
sysreport
```

```
SYSTEM DEFINITION (11.01)

     System name: sysname = System

     State vector: statevec = {x}

     State units: stateunits = {}

     Slope vector: slopevec = {r (1 - x) x}

     Parameter vector: parmvec = {r}

     Parameter values: parmval = {3.2}

     Parameter units vector: parmunits = {}

     Time unit: timeunit =

     System Type: sysmode = mapping
```

We could use this same function as the slope for a differential equation. The command setde switches back to differential equation mode. The primary difference in the two modes is the actual stepping algorithm used in constructing solutions -- a Runge-Kutta step for a differential equation, and a map iteration for the mapping. It is only at that basic level of code that the two modes differ.

```
setde;
```

```
sysreport
```

```
SYSTEM DEFINITION (11.01)
```

```
System name: sysname = System

State vector: statevec = {x}

State units: stateunits = {}

Slope vector: slopevec = {r (1 - x) x}

Parameter vector: parmvec = {r}

Parameter values: parmval = {3.2}

Parameter units vector: parmunits = {}

Time unit: timeunit =

System Type: sysmode = differential equation
```

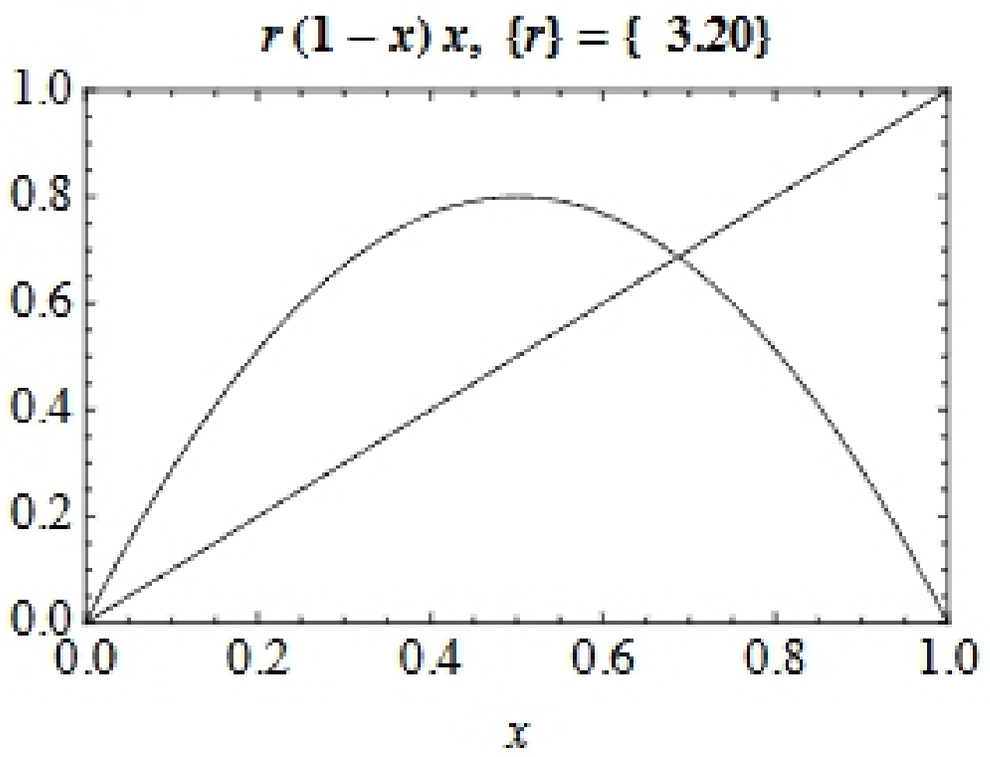
We return to the map setting.

```
setmap;
```

We start by viewing the map.

```
imsize = 250;

viewmap[]
```



The picture suggests that there are two fixed points -- one at 0 and one between 0.6 and 0.8. We find these. Because the mapping is a polynomial, we can use findpolyfix or nfindpolyfix. We can also use the more general nfindfix, which requires an initial guess.