Autumn Quarter 2005
Math. Methods Problem Set 2: Iterated maps

October 11, 2005

1 Analysis of a new iterated map

Consider the system defined by $x_{n+1} = f(x_n, a)$, where

$$f(x, a) = ax\cos(x) \tag{1}$$

You are to perform a complete analysis of the behavior of this system. Using a `mapExplorer` object, and the `orbitDiagram.py` script will help you in your exploration. This exploration requires a combination of analytic work (i.e. using mathematics on paper) and computer simulations. Note that in some cases, you may not be able to find an analytic expression for some of the answers you need (e.g. the maxima of the function $x\cos(x)$). In such cases, feel free to get an approximation by writing a Python script that helps you get an answer, e.g. by printing out a table of values or using a Newton Method iteration.

The following provides some guidelines for your exploration. In all cases, it is implicit that you should discuss how things change as the parameter a is varied.

- Where are the fixed points? Discuss their stability
- If x is initially in the interval $[0, \frac{\pi}{2}]$ under what circumstances will it stay there? What can you say if x is outside this interval?
- When does a stable period-2 orbit appear? When does this go unstable? Discuss what kinds of other periodic orbits appear after the period-2 orbit becomes unstable

- When does the system appear to have a dense set of unstable periodic orbits?
- Show that this system exhibits "unpredictability," once it goes chaotic, in the sense that initially close orbits diverge. Estimate the Lyapunov exponent. Show an example of a chaotic orbit.

Hints and Tips: If you are using a Mac, an easy way to get a figure produced by `plotOrbit` or anything else into your write-up is to use the `Grab` utility, located in the Applications-¿Utilities on OS X Macs. Click on Grab, then follow the instructions to "grab" the contents of any window on your screen. Then you can save this in a convenient format, or copy and paste it into a word processor document. Linux and Windows offer similar screen-capture utilities.

2 Programming project: Computing a histogram

When analyzing data, it is often convenient to make a *histogram.*. A histogram is a plot of the number of data points that fall in certain ranges. For example, if analyzing a dataset containing the mass of bugs found in a field, you might want to know how many have mass between .1 grams and .2 grams, between .3 grams and .4 grams, and so forth.

In this project, you are to write a function to take a list of data, contained in an argument `data`, and a list of the boundaries of the intervals (or "bins"), contained in the list `bins`, and produce a list which contains the number of items in each interval. Both these arguments are Python lists, which are presumed to be lists of integers or floating point numbers. For example, in the bugs example, we might have

```
bugsData = [.01,.14,.15,.21,.01,.11,.25]
bugsBins = [.1,.2,.3,.4]
```

and then then `histo(bugsBins,bugsData)` would return the list `[3,2,0]`. Note that since there is one fewer "bin" than there are bin-boundaries, the length of the list returned is one less than the length of the list `bins`. Note also that, though we used equally space bins in this example, the bins do not have to be equally spaced. (If you knew in advance that the bins were going

to be equally spaced, it would actually be possible to do the calculation in a much more efficient way than is suggested below).

A good way to write this function is to first define an auxiliary function `count(x1,x2,data)`, which returns the number of data items between `x1` and `x2` inclusive. This function can be written simply using one `for` loop and an `if` statement which increments a counter if the data is in the interval. Once you have defined this auxiliary function, it will be easy to generate a list giving the histogram by simply looping over the elements of `bins`. *Note:* You should be able to do this in a single line using list comprehension, without needing a `for` block.

Try out your function by making a histogram of a data set consisting of 100 values of $\sin(x)$ computed on an equally spaced set of points between 0 and 2π. What interval should the set of bins cover? What is a reasonable number of bins to use?

If you want to plot your histogram, you can do it using the code below:

```
from ClimateUtilities import *
#Generate a list of bin centers
centers = [(bins[i]+bins[i+1])/2. for i in range(len(bins)-1)]
#Stuff the data in a curve
c = Curve()
c.addCurve(centers)
c.addCurve(myHisto)
#Plot the curve
plot(c)
```

In the above, I've assumed that your histogram is called `myHisto`, such as might be generated from `myHisto = histo(bins,myData)`. Try it out!