Supplement for Problem Set 9

Due to the computationally intensive plots that you need to generate throughout this problem set, it may be best to work with a script file as explained in section 4 of the *solving ODEs on matlab (ps3supp.pdf).*

problem 1

To determine the $\bar{\mu_i}$, you must be creative. One way to do it is fairly obvious. Use the initial condition $x_1 = x^*$, where x^* is a fixed point at the i^{th} bifurcation. (You should know at least one x^* since one is the same at all of the 2^i bifurcation points) Then, starting with the $\bar{\mu_1}$, vary mu in small increments until you notice that $x_{(2^i)} = x_1$. Continue this until the seventh is found.

problems 2,3

To generate the map of the asymptotic values of x_n, for varying μ, you again should write a script that will do this for you. Many of you will figure this out so skip the rest of this supplement, but for those of you who don't, the framework of the script might be:

- start with $\mu = \mu_0$(your choice)
- evaluate iterate.m many times
- Let transients decay. This can be done simply by letting the system run for a very long time and hope it has reached steady state, or by using a more efficient convergence criterion of your choice.
- After the system has reached the steady state oscillation, plot the next N iterations, where N is a large enough to capture as many bifurcations as you want
- Increase μ by a small amount
- Perform entire process again.

That will generate for you a nice map showing branching of the asympotic values, as a function of mu.

A quick and dirty matlab way to implement something like that could be:

```
%script to generate logistic map

%MAP PARAMETERS

%mu range
mu_0=??; Starting mu, your choice
mu_stop=??;  Ending mu, your choice
mu_delta=??;  Mu step, your choice

%convergence parameters
iter_conv=??;  % Guess at number of iterations required for
               % convergence, or use completely different convergence
               % criterion of your choice

%# of points to plot
N=??;

%standard initial condition
x_0=??;

%CALCULATIONS
Ntot=iter_conv+N;

figure(1)

%cycle mu
for mu=[mu_0:mu_delta:mu_stop]
    [i,x]=iterate(mu,x_0,Ntot);

    % determine converged data (using convergence criterion of choice)
    x_conv=x(iter_conv:Ntot);

    % plot converged data
    mu_n=mu*ones(1,length(x_conv))
    plot(mu_n,x_conv,'.') %plot only dots since lines connecting
                          %points would make a mess
    hold on
end

hold off
```