MATH 4073 Homework 3 Due 09/22/09

Problem 1. Let the sequence $\{p_n\}_{n=1}^{\infty}$ be defined by $p_0 = 0$, $p_n = \exp(-p_{n-1}/2)$ for $n \geq 1$.

(a) Think of the sequence $\{p_n\}_{n=0}^{\infty}$ as a functional iteration, $p_n = g(p_{n-1})$, for an appropriate function g. Write g explicitly. Use Theorem 2.2 (page 54) to prove that g has a fixed point in the interval $[0, 1]$. (In fact, instead of 1, I could have put any arbitrarily large positive number.)

(b) Looking at the explicit expression for $g'(x)$ and thinking about the slope of the graph of $g(x)$ (and comparing it with the slope of the "diagonal", $y = x$), give a simple argument showing that there cannot be more than one positive fixed point of g.

(c) Discuss the uniqueness of the fixed point using the Fixed Point Theorem (Theorem 2.3).

(d) In Mathematica, type the following code:

```
g[x_]=Exp[-x/2];
p = 0;
For[ i = 1, i <= 205, i++,
  { p = g[p],
    Print[i], Print[N[p,50]],
  }
]
exact = N[p,50]
```

then hold down SHIFT and press RETURN to execute it. The value of p at the last step is saved under the name `exact`; we will use it later to study the rate at which p_n approaches the fixed point of g in $[0, 1]$. What do you observe about the behavior of the sequence $\{p_n\}_{n=0}^{\infty}$?

Remark: To check that you indeed obtained the correct answer, you may type

```
N[Exp[-exact/2], 50]
N[exact, 50]
```

and compare the two values.

(e) Now type in Mathematica

```
p = 0;
For[ i = 1, i <= 115, i++,
  { pold = p, p = g[p],
    Print[i], Print[N[p - exact, 50],
    Print[N[(p - exact)/(pold - exact), 50]],
  }
]
```

At each step of the iteration, you see the value of the difference $(p_n - p)$ and of the ratio $\frac{p_n - p}{p_{n-1} - p}$ (p stands for the exact value of the fixed point). What do you observe about the behavior of these two sequences of numbers?

(f) The ratios $\frac{p_n - p}{p_{n-1} - p}$ seem to converge to $-0.351733711249195826024909300929951065\ldots$. What is this number? Could you predict this value without computing it?

Hint: Look at Corollary 2.4.

Problem 2. Consider the function $g(x) = -x^3 + 3x^2 - 2x + 1$. The Mathematica command

```
Plot[{-x^3+3*x^2-2*x+1, x}, {x,-0.1,2.4}]
```

would display the graphs of g and the diagonal $y = x$ (there is no need to attach a printout).

(a) Show (by hand) that $x = 1$ is a fixed point of the function g.

(b) One can easily calculate that $g(0.5) = 0.625$ and $g(1.5) = 1.375$, and can show that g maps the interval $[0.5, 1.5]$ in the interval $[0.625, 1.375]$, which is entirely contained in the interval $[0.5, 1.5]$. Symbolically, one can write $g([0.5, 1.5]) = [0.625, 1.375] \subseteq [0.5, 1.5]$. Therefore, according to Theorem 2.2(a) (page 54), there is a fixed point of g in the interval $[0.5, 1.5]$.

Explain briefly why you can *not* apply Theorem 2.2(b) to show that the fixed point $x = 1$ is the only fixed point in the interval $[0.5, 1.5]$?

(c) Since you could not use Theorem 2.2(b) to show the uniqueness of the fixed point $x = 1$ of the function g, try something else. Define the function $f(x) := g(x) - x$. Show that f is strictly decreasing everywhere except at the point $x = 1$, and use this fact to prove that f cannot have more than one zero.

Hint: Show that the first derivative of f can be written as $-3(x-1)^2$.

(d) Use the Matlab program `fixedpoint.m` (available at the class web-site) fo find the fixed point of g with tolerances 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, and 10^{-6}, with initial value $p_0 = 0.5$. To see the number of iterations the code will perform, set the variable `verbose` to be equal to 1. The stopping criterion this program uses is $|p_n - p_{n-1}| <$ `tol`. Display your results in a table:

Desired tolerance	Value obtained	Number of iterations
10^{-2}	0.8006567083	9
$\vdots$	$\vdots$	$\vdots$

Look at the computed values of the fixed point. Do they look correct within the desired tolerance?

Hint: Make sure that the parameter `nmax` that you pass to the program (the maximum number of iterations allowed) is large enough.

(e) In your opinion, why did the program need such a large number of iterations before the stopping criterion was met and the program stopped? (And recall that the precision obtained was far from the desired tolerance).

Hint: Look at Corollary 2.4 (page 59). Why doesn't it work in our problem? Think about the values of g'.

Problem 3. Consider the one-parameter family of functions $g_a(x) = 1 - ax^2$. Here the real number $a > 0$ is a positive parameter, and x is the independent variable. We are interested in finding fixed points of $g_a(x)$ in the interval $x \in [0, 1]$.

(a) Find (by hand) all fixed points of the recursion relation $p_n = g_a(p_{n-1})$; recall that $a > 0$ by assumption.

(b) Show that: (i) one of the fixed points found in part (a) is always negative (and therefore not interesting for us), and that (ii) the other is always positive and is always in $(0, 1)$.

(c) In this and the next several parts of the problem you will study the behavior of the iterates of g_a. For this purpose you can either use the Mathematica code

```
a = 0.6;
p = 0.2;
g[x_] = 1 - a*x^2;
For[ i = 1 , i <= 200, i++,
  { p = g[p],
    Print[i], Print[p],
  }
]
```

or the Matlab code `fixedpoint.m`, running it as follows:

```
fixedpoint( inline( '1 - 0.6*x^2' ), 0.2, 1.0e-15, 200, 1)
```

For this and the following parts of this problem, there is no need to attach the printouts, just describe what you see! Run the codes with $a = 0.6$ and $p_0 = 0.2$. Do the iterates p_n tend to a limit? What is the numerical value of this limit? Compare it with your theoretical prediction for the value of the limit from part (a).

(d) Now run the code with $a = 1.1$ (again with $p_0 = 0.2$). Do the iterates tend to a limit? Look closely at the last iterates (with $n \approx 200$) – what do you observe?

(e) Run the code with $a = 1.35$ (again with $p_0 = 0.2$). Again, look closely at the last iterates (hint: look at p_{196} and p_{200}).

(f) Run the code with $a = 1.38$ (again with $p_0 = 0.2$). Again, describe the asymptotic behavior of the sequence $\{p_n\}_{n=0}^{\infty}$ and describe what you see (look at p_{192} and p_{200}).

It turns out that, if the parameter a keeps growing, at some values of a the asymptotic behavior of the iterates of g_a changes abruptly – the terminology is that at these values g_a undergoes *period-doubling bifurcations*. The discovery in mid-1970s of some striking properties of this infinite sequence of bifurcations by Mitchell Feigenbaum (a physicist, back then at the Los Alamos National Laboratory) led to a rapid development of the modern *Theory of Dynamical Systems* (which studies the asymptotic behavior of high iterates of maps). A famous early article on simple ecological models that exhibit interesting phenomena is "Simple mathematical models with very complicated dynamics" by Robert May (published in *Nature* **261** (1976), 459–467), which is attached to my e-mail with this homework. It is a pleasure to read – take a look at it when you have time.

Problem 4. When I was a kid, I really wanted to have a calculator, but did not. A friend of mine did, and we played with it a lot. But the calculator could only add, subtract, multiply and divide, and we wanted to do more complicated things, like computing square roots. (At the time we did not even know that functions like "sin", "cos" and "exp" existed).
So we thought of the following trick to compute, say, $\sqrt{10}$. If we take a number, say, $p_0 = 5$, and divide 10 by this number, we would obtain something smaller or something bigger than 5, or, if we were extremely lucky, we would obtain 5, which would mean that our initial guess that $\sqrt{10}$ was equal to 5 was correct (but, of course, such nice things never happen). So, ignoring the unlikely case of guessing the value of $\sqrt{10}$ right away, we have the following two possibilities:

(i) if our initial guess p_0 was smaller than the true value of $\sqrt{10}$, then $\frac{10}{p_0}$ would be greater than $\sqrt{10}$ (why?);

(ii) if our initial guess p_0 was greater than the true value of $\sqrt{10}$, then $\frac{10}{p_0}$ would be smaler than $\sqrt{10}$.

In any case, the true value of $\sqrt{10}$ would be bracketed between p_0 and $\frac{10}{p_0}$. So let us take the average of these two numbers, $p_1 := \frac{1}{2}\left(p_0 + \frac{10}{p_0}\right)$, as a better approximation to $\sqrt{10}$. Then we can repeat the same procedure with p_1 and obtain an even better approximation p_2, etc. To our great satisfaction, the method worked very well!

(a) Write explicitly the described procedure as a fixed-point iteration.

(b) Run verbosely `fixedpoint.m` to find $\sqrt{10}$ with accuracy 10^{-10}. Attach your printout.

(c) Construct a Newton's iterative procedure for finding $\sqrt{a}$ numerically. Compare the Newton method for this problem with above procedure.

(d) Run verbosely the Matlab code `newton.m` (available at the class web-site) to find $\sqrt{10}$. Attach your printout.