MATH 4093/5093 Homework 5 NOT Due Fri, 10/08/10

Problem 1. Assume that the sequence $\{p_n\}_{n=0}^{\infty}$ is generated by some iterative method for finding a root of an equation. Also assume that we know that the sequence $\{p_n\}_{n=0}^{\infty}$ converges to some number p of some order α with some asymptotic error constant λ, but we don't know the values of α and λ. The goal of this problem is to develop a method for determining the numerical value of α from the numerical values of the members of the sequence $\{p_n\}_{n=0}^{\infty}$.

Let $E_n := |p_n - p|$ be the error at the nth step of the iteration, and define $\ell_n := \ln E_n$.

(a) Show that for large n, the following approximate identity holds:

$$\ell_n - \alpha \ell_{n-1} \approx \ln \lambda \ .$$

Hint: Just look at the definition of order of convergence.

(b) Using the approximate identity derived in (a) show that

$$\alpha \approx \frac{\ell_n - \ell_{n+1}}{\ell_{n-1} - \ell_n} \ .$$

Note that this approximate formula for α does not depend on the base of the logarithms; if ℓ_n is defined as the log base 10 of E_n, the formula will remain the same.

(c) The following Mathematica code

```
f[x_] := Sin[x] + x - 1;
p = N[1, 10000];
For[ i = 1, i <= 20, i++,
     { p = p - f[p] / f’[p],
     }
]
exactvalue = p;
Print[N[exactvalue, 50]]
]
```

uses the Newton's method to compute the root of the equation

$$\sin x + x = 1 \ , \tag{1}$$

starting from the value $p_0 = 1$, and performing 20 steps of Newton's iteration; the computations are performed with 10000 decimal digits of accuracy. The result is save as the variable `exact`, and 50 digits of `exact` are printed, so that the output is the "exact" (with 10000 digits) value of root of $\sin x + x = 1$:

```
exact = 0.51097342938856910952001397114508063204535889262375
```

Having computed the "exact" value of the root of (1), I used it to study the performance of several numerical methods for computing roots of equations.

First I ran the Mathematica code

```
p = N[1, 10000];
For[ i = 1, i <= 16, i++,
     { p = p - f[p] / f'[p],
       error = Abs[p - exactvalue],
       Print[ i, "   ", N[Log[error]/Log[10], 10]]
     }
]
```

which performed Newton's iteration for solving the same equation as above, with starting value $p_0 = 1$, and printed the logarithm (base 10) of the value of the absolute error, $|p_n - p_{\text{exact}}|$ for each step. The results are given in Table 1 below ("Indeterminate" means that the accuracy of 10000 digits was not enough).

The Mathematica code

```
p0 = N[0, 10000];
p1 = N[1, 10000];
For[ i = 1, i  16, i++,
     { p = p1 - f[p1]*(p1 - p0)/(f[p1] - f[p0]),
       error = Abs[p - exactvalue],
       Print[ i, "   ", N[ Log[error]/Log[10]], 10] ],
       p0 = p1,
       p1 = p
     }
]
```

used the secant method for solving (1), with starting values $p_0 = 0$ and $p_1 = 1$; the output has the same format as the one of the Newton's method and is presented in Table 1.

Finally, I wrote equation (1) as a fixed-point problem, $g(p) = p$, where $g(x) = x - f(x) = x - (\sin x + x - 1) = 1 - \sin x$, and solved it using Mathematica with initial value $p_0 = 0.5$:

```
g[x_] := 1 - Sin[x];
p = N[1/2, 10000];
For[ i = 1, i <= 50, i++,
     { p = g[p],
       error = Abs[p - exactvalue],
       Print[i, "   ", N[Log[error]/Log[10], 10]]
     }
]
```

Table 1: Results of the Newton's and secant methods, and an FPI for solving (1).

n	$\log_{10}\lvert p_n - p_{\text{exact}}\rvert$, Newton's	$\log_{10}\lvert p_n - n - p_{\text{exact}}\rvert$, secant	$\log_{10}\lvert p_n - n - p_{\text{exact}}\rvert$, FPI
1	-1.242027932	-1.493891618	-2.017682082
2	-3.405246446	-2.540518867	-2.078208795
3	-7.694806441	-4.909346984	-2.136547863
4	-16.27367865	-8.334835543	-2.196791228
5	-33.43142307	-14.12824440	-2.255370626
6	-67.74691190	-23.34714570	-2.315399191
7	-136.3778896	-38.35945587	-2.374162006
8	-273.6398449	-62.59066733	-2.434027590
9	-548.1637555	-101.8341890	-2.492930284
10	-1097.211577	-165.3089221	-2.552672122
11	-2195.307219	-268.0271768	-2.611681434
12	-4391.498504	-434.2201646	-2.671329270
13	-8783.881074	-703.1314071	-2.730419812
14	Indeterminate	-1138.235637	-2.789996212
15	Indeterminate	-1842.251110	-2.849148622
16	Indeterminate	-2981.370814	-2.908670720

Use the formula for α derived in part (b) to find empirically the numerical values of α for the three methods above, using the results from Table 1. Does the values of α you have obtained match the theoretical predictions? Discuss briefly. One can prove (but you do *not* need to do this!) that the order of convergence of the secant method is

$$\alpha_{\text{secant}}^{(\text{theor})} = \frac{1}{2}(\sqrt{5}+1) \approx 1.61803398874\ldots\ .$$

Problem 2. Recall that the multiplicity of a zero p of the function f is defined as the number m such that

$$f(x) = (x-p)^m\, q(x)\ ,$$

where q is a function satisfying $\lim_{x\to p} q(x) \neq 0$.

Recall also that Newton's method for finding a zero of the function f (or, equivalently, a root of the equation $f(x)=0$) is based on the iterative procedure $p_{n+1} = g(p_n)$, where p_0 is some starting value, and

$$g(x) = x - \frac{f(x)}{f'(x)}\ .$$

We stated in class that, if p is a simple zero of f (i.e., a zero of multiplicity 1) and the Newton's method converges to p, then the convergence is at least quadratic, i.e., or order $\alpha \geq 2$.

If, however, the zero of f is non-simple, then the Newton's method converges only linearly.

In class we proved that, if p is a fixed point of the function g and $g'(p) \neq 0$, then if the iteration $p_{n+1} = g(p_n)$ converges to p, then the convergence is linear (and $\lambda = |g'(p)|$).

In this problem you will show that, indeed, the Newton's method converges linearly for $m \geq 2$, and will find a modification of Newton's method that works with multiple zeros (but one needs to know the multiplicity of the zero and pass it to the program as one of the arguments).

Let p be a zero of multiplicity $m \geq 2$ of f. Then the Newton's iteration for finding a zero of f has the form

$$\begin{aligned} g(x) &= x - \frac{f(x)}{f'(x)} \\ &= x - \frac{(x-p)^m q(x)}{[(x-p)^m q(x)]'} \\ &= x - \frac{(x-p)^m q(x)}{m(x-p)^{m-1} q(x) + (x-p)^m q'(x)} \\ &= x - (x-p)\frac{q(x)}{mq(x) + (x-p)q'(x)} \ , \end{aligned}$$

therefore

$$g'(x) = 1 - \frac{q(x)}{mq(x) + (x-p)q'(x)} - (x-p)\,\frac{\mathrm{d}}{\mathrm{d}x}\left(\frac{q(x)}{mq(x) + (x-p)q'(x)}\right) \ .$$

This implies that

$$g'(p) = 1 - \frac{1}{m} \neq 0 \ ,$$

hence the convergence of Newton's method is only linear.

(a) Let p be a zero of multiplicity $m \geq 2$ of f. Consider the following modification of the Newton's method: $p_{n+1} = g(p_n)$, where

$$g(x) = x - m\frac{f(x)}{f'(x)} \ .$$

Show that in this case $g'(p) = 0$, hence the convergence is faster than linear.

(b) Show that the multiplicity of the root $\frac{\pi}{2}$ of the equation $(x - \frac{\pi}{2})(1 - \sin x) = 0$ is $m = 3$. *Hint:* Expand $\sin x$ in a Taylor series around $x_0 = \frac{\pi}{2}$.

(c) The following Mathematica code (similar to the codes from Problem 1)

```
p = N[3, 50000];
m = 3;
f[x_] := (x - Pi/2) * (1-Sin[x]);
For[i = 1, i <= 10, i++,
  { p = p - m*f[p]/f'[p],
    error = Abs[p - Pi/2],
    Print[i, "   ", N[Log[error],10]]
  }
]
```

can be used to find empirically the order of convergence of the method. Here is the output:

```
1      -0.7204139049
2      -3.415376010
3      -11.50140053
4      -35.75947410
5      -108.5336948
6      -326.8563569
7      -981.8243432
8      -2946.728302
9      -8841.440179
10      -26525.57581
```

What is the order of the convergence that you observe? Explain briefly your reasoning. (Note that we are doing the calculations with accuracy or 50000 decimal digits!)

(d) The number $\frac{\pi}{2}$ is a root of the equation $(x - \frac{\pi}{2})^3\,(1 - \sin x) = 0$ of multiplicity 5. The Mathematica code from part (c) can be modified appropriately find empirically the order of convergence in this case. The output is given below. What do you observe?

```
1      -0.9648058725
2      -4.371283436
3      -14.59097156
4      -45.25003594
5      -137.2272291
6      -413.1588085
7      -1240.953547
8      -3724.337761
9      -11174.49041
10      Indeterminate
```

(e) Modify the Matlab code `newton.m` to write a code `newton_m.m` that implements the algorithm from part (a) for finding zeros of multiplicity m of the equation $f(x) = 0$. The first line of your code must be

```
function r = newton_m( fun, funder, m, xinit, tol, nmax, verbose)
```

(f) One can estimate roughly the order of convergence m of an iterative method as follows. If after iterating enough number of times the error is significantly smaller than 1 (say, 0.01 or smaller), then the number of correct digits of p_{n+1} is roughly m times the number of correct digits of p_n. You can observe this, in the following quadratically convergent sequence (coming from using Newton's method to find $\sqrt[3]{12}$ as a root of the equation $x^3 - 12 = 0$).

```
1.000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
4.666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666
3.294784580498866213151927437641723356009070294784580498866213151927437641723356009070294784580498866
2.564996283933496822279640898870787662335200127188373868695090191534036878441096931206491314332028214
2.317973627725310965894671183077646390006759422097600376823322683758217378024541415220574391246824662
2.289778566911922285264949690307277524808339353864854224824331811491750945243930217290781459861111391
2.289428538627570193775919893742118572719030429524354378801546408311322845023342839603272971787615841
2.289428485106664986796174761541083407108687502501926456729710419471508178918399015462717660858495387
2.289428485106663735616084423880037791808000926685328369417780029499568633251106146440076011857659331
2.289428485106663735616084423879354017831813841575862144198104552351246075611950964542191974634843603
2.289428485106663735616084423879354017831813841575862144198104348131348598048428300875216322061834091
2.289428485106663735616084423879354017831813841575862144198104348131348598048428300875216322061834091
```

Give a theoretical explanation of this rule of thumb.

Hint: For example, in the number 2.289428485106664986796 . . . in the sequence above, you know the result with 8 correct digits after the decimal point: 2.28942848510666; the next one, 2.2894284851066637356160844238800377 9 . . ., has 28 correct digits after the decimal point, namely, 2.2894284851066637356160844238. If you know a number p_n with k digits of accuracy, what can you say about $\log_{10} |p_n - p|$?