MATH 4073 Homework 5 Due Mon, 10/04/10

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 Mathematica notebook `rate_of_convergence1.nb` (which you can download from the class website) computes the value of the solution of the equation $\sin x + x = 1$ using (i) Newton's method, (ii) the secant method, (iii) the fixed point iteration method, (iv) the fixed point iteration method combined with Aitken extrapolation, and (v) the fixed point iteration method combined with Steffensen's method for accelerating convergence. Run all the programs from the notebook and use the formula for α derived in part (b) to find empirically the numerical values of α for all these methods. 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

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

Note: Since we are using a very high accuracy (namely, 10000 digits), you have to be patient when runnig the programs; make sure that a program is finished before you run the next one. Attach the printout of running the programs. In the printout, mark clearly the numbers that you use in your calculations, and show the calculations in detail. Recall that to perform some calculation in Mathematica, you need to put the cursor on that part of the program, press SHIFT, and while holding it down, press RETURN.

Problem 2. In this problem you will develop a modification the Newton's method to find a zero of multiplicity $m \geq 1$ of the equation $f(x) = 0$.

(a) Let the sequence $\{p_n\}_{n\in\mathbb{N}}$ be defined by the recurrence relation $p_n = g(p_{n-1})$ (and p_0 is some initial guess), where $g(x) = x - \dfrac{m\, f(x)}{f'(x)}$. Prove that, if the point p is a zero of multiplicity m of f, then $g'(p) = 0$. This implies that the sequence $\{p_n\}_{n\in\mathbb{N}}$ converges faster than linearly.

Hint: The calculations you need to do are very similar to the ones on pages 80–81 of the book involving the function $\mu(x)$.

(b) Find the multiplicity of the root $\frac{\pi}{2}$ of the equation $(x - \frac{\pi}{2})(1 - \sin x) = 0$.

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

```
p = N[3, 50000];
m = (***put here your answer from part (b)***) ;
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, "   ", Log[1.0*error]]
  }
]
```

can be used (after an appropriate modification) to find empirically the order of convergence of the method. What is the order of the convergence that you observe? Explain briefly your reasoning. (Again, be patient – Mathematica can take a while because you 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. Run the Mathematica code from part (c) after modifying it appropriately and use the output to find the order of convergence in this case. What do you observe?

(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)
```

Note that in Matlab the name of a file must be the same as the name of the code in it.

(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 in Problem 2 of Homework 4)

```
1.000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
4.666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666
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 $2.289428485106664986796\ldots$ in the sequence above, you know the result with 8 correct digits after the decimal point: 2.28942848510666; the next one, $2.289428485106663735616084423880037791\ldots$, has 28 correct digits after the decimal point, namely, 2.28942848510666373561608442388. If you know a number p_n with k digits of accuracy, what can you say about $\log_{10}|p_n - p|$?

Problem 3. The Mathematica code

```
p = N[1, 50];   g[x_] = Exp[-x];
For[i = 1, i <= 9, i++,
  { Print[i], Print[N[p, 13]],
    p = g[p],
  }
]
```

performs a fixed point iteration of the function $g(x) = e^{-x}$ with initial point $p_0 = 1$. The output is given in the table below.

n	p_n
0	1.0000000000000
1	0.3678794411714
2	0.6922006275553
3	0.5004735005636
4	0.6062435350856
5	0.5453957859750
6	0.5796123355034
7	0.5601154613611
8	0.5711431150802
9	0.5648793473910
⋮	⋮

Use the values of p_7, p_8, and p_9 to compute the term $\hat{p}_7$ by using Aitken's Δ^2-method for accelerating the convergence of the sequence $\{p_n\}$. Compare the values of the errors $|p_9 - p|$ and $|\hat{p}_7 - p|$, where p is the exact value of the fixed point, $p = 0.5671432904098\ldots$.

Problem 4. Use the synthetic division algorithm to find consecutively the following ratios:

$$\frac{x^5 - 12x^4 + 64x^3 - 158x^2 + 15x + 250}{x + 1}\,, \qquad \frac{x^5 - 12x^4 + 64x^3 - 158x^2 + 15x + 250}{(x + 1)(x - 2)}\,.$$

Use this result to find all roots (including the non-real ones) of the polynomial equation

$$x^5 - 12x^4 + 64x^3 - 158x^2 + 15x + 250 = 0\,. \tag{1}$$

The built-in Matlab command `roots` finds (possibly complex) roots of polynomials. The argument of `roots` is a vector of all coefficients of the polynomial, starting with the one at the highest power. For example, to find all roots of the polynomial $2x^5 + x^4 + 5x^2 - 13x + 5$, type `roots([2 1 0 5 -13 5])` and press RETURN. Use this command to find the roots of the polynomial equation (1) after you deflate it by eliminating the roots -1 and 2 (so that you have to use `roots` to solve a cubic equation). Attach your Matlab printout.

Remark: One can find out more information about Matlab functions, say `roots`, by typing

```
help roots
```

Use this to see the description of Matlab command `fzero` which is a built-in zero finder. By typing

```
help help
```

you will get information about the command `help` itself, and

```
help /
```

will give you a description of all operators and special characters.