CS 323 Computer Problem: Solution of Nonlinear Equations
due February 23, 2011

The objective of this assignment is to write a general MATLAB program for computing real roots of polynomials via Newton's method, and then apply it to a particular polynomial.

Guidelines for program

(i) Write a MATLAB function $[p, pprime] = poly(a, z)$, say, which evaluates

$$p(x) = a_1 x^n + \cdots + a_n x + a_{n+1}$$

and its first derivative at $x = z$, returning the values $p(z)$ and $p'(z)$ in p and $pprime$. The evaluation should be based on *nested multiplication*, as described below. Place this function in a separate file *poly.m*, say. Note: There is no need to pass n as an argument to *poly* - it can be obtained within *poly* via the Matlab function *length*. (Type *help length* for more information.)

(ii) The main program should be placed in a file *newton.m*, say. It should:

A. Read in as data...

the coefficients of the polynomial whose roots are to be computed
two parameters *abserr* and *itmax* explained in (iii)
initial guesses for Newton's method

B. Apply Newton's method to $p(x)$ for each initial guess, and print out the following table of iterates and f values:

x_k	$f(x_k)$
~	~
~	~
⋮	⋮

(iii) Stopping criteria for Newton's method:

$$|x_k - x_{k-1}| < abserr \quad \text{(success)}$$
$$k > itmax \quad \text{(failure)}$$

As a debugging check, apply your program to the example problem we used in class: $x^2 - 2 = 0$, $x_0 = 1$.

Once you are confident your program is working properly, apply it to the problem of computing the real roots of

$$p(x) = 49x^4 + 7x^3 - 82x^2 + 51x - 9$$

to within absolute error $< 10^{-10}$. Be sure to print your x_k's with *at least* 10 digits to the right of the decimal point. Use the Matlab command *diary* to save the output from your program. (Type *help diary* for more information.)

What are your approximations to the roots of $p(x)$? Provide a brief explanation of your strategy for choosing initial iterates. Also, provide an interpretation of your computational results, including observed convergence rates. Do you believe you obtained the requested 10^{-10} accuracy in your approximations to the roots? Explain.

<u>Note:</u> If $\overline{x}$ is a multiple root of $p(x)$ (i.e., $p(x)$ contains $(x-\overline{x})^m$ as a factor where $m \geq 2$), then $p'(\overline{x}) = 0$. If $\overline{x}$ is a single root, then $p'(\overline{x}) \neq 0$.

Nested multiplication for evaluating $p(x) = a_1 x^n + \cdots + a_n x + a_{n+1}$
and its first derivative at $x = z$

Write $p(x)$ in "nested" form:

$$p(x) = (a_{n+1} + x(a_n + x(a_{n-1} + \cdots + x(a_2 + x a_1) \cdots))).$$

Evaluating at $x = z$ starting from the innermost pair of parentheses, we obtain the following algorithm:

$b_1 = a_1$
for $k = 2 : n+1$
    $b_k = a_k + z * b_{k-1}$
Then $b_{n+1} = p(z)$.

This algorithm (known also as synthetic division or Horner's rule) requires n multiplications and n additions compared with $2n-1$ multiplications and n additions for "direct" evaluation of $p(z)$.

<u>Schematic description of algorithm for evaluating $p(z)$:</u>

$$\begin{array}{c|cccccc}
z \rfloor & a_1 & a_2 & \cdots & a_{n-1} & a_n & a_{n+1} \\
 & & z b_1 & & z b_{n-2} & z b_{n-1} & z b_n \\
\hline
 & b_1 & b_2 & \cdots & b_{n-1} & b_n & \lfloor b_{n+1} = p(z)
\end{array}$$

Now suppose $p'(z)$ is also desired. Then use the b_k's from above to form

$$q(x) \equiv b_1 x^{n-1} + \cdots + b_{n-1} x + b_n.$$

One may verify that $q(x)$ and b_0 are the quotient and remainder when $p(x)$ is divided by $x - z$, i.e.,

$$p(x) = (x - z) q(x) + b_0.$$

Differentiating this representation for $p(x)$ and evaluating at $x = z$, we see that $p'(z) = q(z)$. Thus $p'(z)$ can be gotten by applying the nested multiplication algorithm to $q(x)$. Moreover, from the standpoint of the schema, the b_k's are laid out in just the right way for doing this.

Schematic description of algorithm for evaluating $p(z)$, $p'(z)$:

$$
\begin{array}{r|cccccc}
z\rfloor & a_1 & a_2 & \cdots & a_{n-1} & a_n & a_{n+1} \\
 & & zb_1 & & zb_{n-2} & zb_{n-1} & zb_n \\
\hline
z\rfloor & b_1 & b_2 & \cdots & b_{n-1} & b_n & \lfloor b_{n+1} = p(z) \\
 & & zc_1 & & zc_{n-2} & zc_{n-1} & \\
\hline
 & c_1 & c_2 & \cdots & c_{n-1} & \lfloor c_n = p'(z) &
\end{array}
$$

The following algorithm evaluates both $p(z)$ and $p'(z)$:

$b_1 = a_1$
for $k = 2 : n+1$
    $b_k = a_k + z * b_{k-1}$
$c_1 = b_1$
for $k = 2 : n$
    $c_k = b_k + z * c_{k-1}$
Then $p(z) = b_{n+1}$ and $p'(z) = c_n$.