CS 510 Computer Problem: Computation of real roots of polynomials due 2/25/08

Write a program which uses Newton's method to compute real roots of an n^{th} degree polynomial

$$p(x) = a_n x^n + \cdots + a_1 x + a_0$$

with real coefficients. The program should accept as input the degree and coefficients of the polynomial, stopping conditions for the Newton iteration, and initial guesses - most conveniently supplied one at a time since not all initial guesses are likely to productive. For each initial iterate supplied to the program, print out the corresponding Newton iterates x_k and $f(x_k)$ values. The polynomial and its derivative should be evaluated by nested multiplication (as described below).

Use your program to compute the real roots of

$$p(x) = 49x^4 - 224x^3 + 165x^2 - 44x + 4$$

to within absolute error 10^{-10}, if possible.

Provide an interpretation of your computational results, including observed convergence rates and inferences about the multiplicities of the roots. Do you feel you attained the desired accuracy in your computed roots? [Note: The important thing in this computer problem is not the program itself, but the computational results it gives and your analysis of those results.]

Nested multiplication for evaluating $p(x) = a_n x^n + \cdots + a_1 x + a_0$ and its first derivative at $x = z$

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

$$p(x) = (a_0 + x(a_1 + x(a_2 + \cdots + x(a_{n-1} + xa_n)\cdots))).$$

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

$b_n \leftarrow a_n$
for $k = n-1, \ldots, 0$
  $b_k \leftarrow a_k + zb_{k+1}$
Then $b_0 = p(z)$.

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 $p(z)$.

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

$z\rfloor$	a_n	a_{n-1}	$\cdots$	a_2	a_1	a_0
		zb_n		zb_3	zb_2	zb_1
$z\rfloor$	b_n	b_{n-1}	$\cdots$	b_2	b_1	$\lfloor b_0 = p(z)$

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

$$q(x) \equiv b_n x^{n-1} + \cdots + b_2 x + b_1.$$

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 $p(z)$, $p'(z)$:

$z\rfloor$	a_n	a_{n-1}	$\cdots$	a_2	a_1	a_0
		zb_n		zb_3	zb_2	zb_1
$z\rfloor$	b_n	b_{n-1}	$\cdots$	b_2	b_1	$\lfloor b_0 = p(z)$
		zc_n		zc_3	zc_2	
	c_n	c_{n-1}	$\cdots$	c_2	$\lfloor c_1 = p'(z)$	

The following algorithm evaluates $p(z)$ and $p'(z)$ simultaneously, without creating any new arrays:

$b \leftarrow a_n$
$c \leftarrow b$
for $k = n - 1, \ldots, 1$
    $b \leftarrow a_k + zb$
    $c \leftarrow b + zc$
$b \leftarrow a_0 + zb$
Then $b = p(z)$ and $c = p'(z)$.