Christopher Wong
999234204
MAT 128B

Programming Project 1 **2/1/2017**

Problem 1

(a)

$$f(x) = x + lnx.$$

Using $a_0 = 0.5$ and $b_0 = 1$ for REGFAL, $x_0 = 0.5$ and $x_1 = 1$ as starting values for SECANT, and $x_0 = 1$ as initial guess for NEWTON, we obtain:

Program	Iterations	Evaluations of f	Evaluations of f'	x_k	$f(x_k)$
REGFAL	7	9	NA	5.671432904097840e-01	2.220446049250313e-16
SECANT	6	8	NA	5.671432904097838e-01	-1.110223024625157e-16
NEWTON	5	6	5	5.671432904097838e-01	-1.110223024625157e-16

We see that Newton's method converges the fastest as expected because it has quadratic convergence. After comes the secant method with its order of convergence between linear and quadratic and lastly the regula falsi method whose convergence can be the same as the bisection method (linear convergence) or even slower. In this problem, all three methods converged to the zero of the function.

(b)

$$f(x) = 1 - 10x + 0.01e^x.$$

Using $a_0 = 5$ and $b_0 = 20$ for REGFAL, $x_0 = 5$ and $x_1 = 20$ as starting values for SECANT, and $x_0 = 20$ as initial guess for NEWTON, we obtain:

Program	Iterations	Evaluations of f	Evaluations of f'	x_k	$f(x_k)$
REGFAL	154708	154710	NA	9.105602120447736e+00	-4.649251650334918e-09
SECANT	7	9	NA	1.011063943496088e-01	6.071532165918825e-17
NEWTON	17	18	17	9.105602120505811e+00	-5.684341886080802e-14

We notice something strange about our zero(s) from our programs! There are two zeros to this function and it just so happens that Newton's method converged slower than the secant method to the larger zero of the function. The secant method converged to the smaller zero of the function the fastest of the programs and the regula falsi method converged EXTREMELY slow to the larger zero of the function. The regular falsi method program also terminated differently than the other two because its $|f(x_k)| > FTOL = 10^{-15}$ so it stopped as a result of the x_k values having a difference of less than XTOL $= 10^{-15}$ for each additional iteration after 154,708.

Christopher Wong
999234204
MAT 128A
Programming Project 1 **2/1/2017**

(c)

$$f(x) = x^2 - 2xe^{-x} + e^{-2x}.$$

Running NEWTON with initial guess $x_0 = 1$:

Program	Iterations	Evaluations of f	Evaluations of f'	x_k	$f(x_k)$
NEWTON	25	26	25	5.671433018515338e-01	3.885780586188048e-16

We notice that while Newton's method converged to a zero, it did so slowly. Newton's method has the property of quadratic convergence so we expect a very rapid convergence to the zero especially since we started close to it ($x_0 = 1$).

(d)

$$f(x) = \arctan x - \frac{2x}{1+x^2}.$$

Using $a_0 = 1$ and $b_0 = 2$ for REGFAL, $x_0 = 1$ and $x_1 = 2$ as starting values for SECANT, and $x_0 = 1$ as initial guess for NEWTON, we obtain:

Program	Iterations	Evaluations of f	Evaluations of f'	x_k	$f(x_k)$
REGFAL	6	8	NA	1.391745200270735e+00	2.220446049250313e-16
SECANT	5	7	NA	1.391745200270735e+00	0
NEWTON	4	5	4	1.391745200270735e+00	2.220446049250313e-16

All three programs found the same positive zero of the function. We see that Newton's method converges the fastest as expected because it has quadratic convergence. After comes the secant method with its order of convergence between linear and quadratic and lastly the regula falsi method whose convergence can be the same as the bisection method (linear convergence) or even slower.

(e)

$$f(x) = \arctan x.$$

Using the zero found in part (d) as initial guess for NEWTON we obtain an error stating that the derivative of our zero estimate $f'(x_k) = 0$ after 46 iterations. So we conclude that Newton's method diverges for this function and initial guess. The estimates before the error are given below:

Iterations	Evaluations of f	Evaluations of f'	x_k	$f(x_k)$
46	46	45	-3.359067911765232e+181	-1.570796326794897e+00

Christopher Wong
999234204
MAT 128B

Programming Project 1 **2/1/2017**

(f)

$$f(x) = \arctan x.$$

Using NEWTON with the specified initial guesses x_0 gives:

x_0	Iterations	Evaluations of f	Evaluations of f'	x_k	$f(x_k)$
1	5	6	5	0	0
5	ERROR AT: 9	9	8	1.309270537307534e+214	1.570796326794897e+00
50	ERROR AT: 8	8	7	-4.625029064936211e+241	-1.570796326794897e+00
100	ERROR AT: 8	8	7	-3.570421266828119e+280	-1.570796326794897e+00

We notice that starting at $x_0 = 1$ is the only initial guess that makes Newton's method converge to a zero. The remaining initial guesses $x_0 = 5$, 50, 100 make Newton's method diverge after 8 or 9 iterations when the derivative at the estimate x_k, $f'(x_k) < 10^{-16} = 0$.

Christopher Wong
999234204
MAT 128A

Programming Project 1 **2/1/2017**

Problem 2

(a)

$$f(x) = \arctan x - \frac{2x}{1+x^2}.$$

Using NEWTON with the specified initial guesses x_0 gives:

x_0	Iterations	Evaluations of f	Evaluations of f'	x_k	$f(x_k)$
-40	ERROR AT: 7	7	6	-2.633572261133008e+83	-1.570796326794897e+00
-30	ERROR AT: 8	8	7	3.536231819534596e+149	1.570796326794897e+00
-5	5	6	5	-1.391745200270735e+00	-2.220446049250313e-16
10	ERROR AT: 8	8	7	-1.379680173705174e+78	-1.570796326794897e+00
20	ERROR AT: 8	8	7	-6.696780523894109e+124	-1.570796326794897e+00

Using NEWTON_DAMPING with the specified initial guesses x_0 gives:

x_0	Iterations	Evaluations of f	Evaluations of f'	x_k	$f(x_k)$
-40	67	68	67	-1.391745200270737e+00	-1.221245327087672e-15
-30	63	64	63	-1.391745200270737e+00	-1.221245327087672e-15
-5	50	51	50	-1.391745200270737e+00	-1.221245327087672e-15
10	53	54	53	1.391745200270737e+00	1.221245327087672e-15
20	58	59	58	1.391745200270737e+00	1.221245327087672e-15

In terms of convergence, we find that when Newton's method without damping only converged if our initial guess x_0 was close enough to the zero of the function and did so quadratically as expected. For all other initial values, Newton's method without damping diverged. On the other hand, Newton's method with damping converged for all of our initial guesses albeit a lot slower than the expected quadratic convergence of Newton's method. We also notice that when our initial guess converged to the closest zero of the function (i.e. negative initial guesses will converge to the negative zero of f and positive initial guesses will converge to the positive zero of f).

Christopher Wong
999234204
MAT 128B
Programming Project 1 **2/1/2017**

(b)

$$f(x) = \frac{1-x}{1+x}, \quad x \ -1.$$

Using NEWTON with the specified initial guesses x_0 gives:

x_0	Iterations	Evaluations of f	Evaluations of f'	x_k	$f(x_k)$
2	6	7	6	1	0
5	ERROR AT: 10	10	9	-2.681561585988510e+154	-1
10	ERROR AT: 9	9	8	-3.337594124014076e+167	-1
100	ERROR AT: 8	8	7	-1.623661373928269e+217	-1

Using NEWTON_DAMPING with the specified initial guesses x_0 gives:

x_0	Iterations	Evaluations of f	Evaluations of f'	x_k	$f(x_k)$
2	1	2	1	2	-3.333333333333333e-01
5	1	2	1	5	-6.666666666666666e-01
10	1	2	1	10	-8.181818181818182e-01
100	1	2	1	100	-9.801980198019802e-01

We find that Newton's method without damping converged if our initial guess x_0 (in this case $x_0 = 2$) was close enough to the zero of the function and did so quadratically as expected. For all other initial values, Newton's method without damping diverged. For this function, Newton's method with damping does not change the estimate of the zero from the initial guess for these values because the next estimate was too close to our initial guess. That is, $|x_k - x_{k-1}| \leq (1 + |x_k|)$XTOL. This is a result of the small derivative at these points forcing the damping factor λ_k to be very small. Because of this, we find

$$x_{k+1} = x_k - \lambda_k \frac{f(x_k}{f'(x_k)} \approx x_k.$$

So our program for Newton's method with damping immediately terminates and returns the initial estimate.

Christopher Wong
999234204
MAT 128A

Code Appendix

REGFAL.m

```
% Problem 1 REGFAL (regula falsi method)
% clear
format long e

% cd 'C:\Users\Christopher\Desktop\MAT 128B\Project 1';

% Taking the input and converting it into a function
input_func = input('Evaluate f(x) = ', 's');
f = str2func(strcat('@(x)', input_func));

a0 = input('Input the left endpoint of the initial interval: ');
b0 = input('Input the right endpoint of the initial interval: ');
KTOL = input('Input the iteration bound: ');
FTOL = input('Input the function convergence tolerance: ');
XTOL = input('Input the interval convergence tolerance: ');

if (a0 >= b0 || f(a0)*f(b0) >= 0)
  error('Interval endpoints are not valid.');
end

a = a0;
b = b0;
fa = f(a);
fb = f(b);
f_eval = 2;

% Note that in the lecture notes k starts at 0, so we have to shift the
% index by 1
for k = 1:(KTOL)
  % Storing the x_k's in c(k)
  c(k) = (b*fa - a*fb) / (fa - fb);
  fc = f(c(k));
  f_eval = f_eval + 1;
  if (abs(fc) <= FTOL)
    break;
  end
  if (k > 1)
    if (abs(c(k) - c(k-1)) <= (1 + abs(c(k)))*XTOL)
      break;
    end
  end
  % Although this check should theoretically never happen...
  if (k > (KTOL+1))
    fprintf('Method failed.');
```

```
  end
  if (sign(fa) <= sign(fc))
    a = c(k);
    fa = fc;
  else
    b = c(k);
    fb = fc;
  end
end

fprintf('Number of iterations: %d\n', k);
fprintf('Number of function evaluations: %d\n', f_eval);
fprintf('Estimated root x = %d and f(x) = %d\n', c(k), fc);
```

Christopher Wong
999234204
MAT 128A

SECANT.m

```
% Problem 1 SECANT (secant method)
% clear
format long e

% cd 'C:\Users\Christopher\Desktop\MAT 128B\Project 1';

% Taking the input and converting it into a function
input_func = input('Evaluate f(x) = ', 's');
f = str2func(strcat('@(x)', input_func));

x0 = input('Input the first starting value: ');
x1 = input('Input the second starting value: ');
KTOL = input('Input the iteration bound: ');
FTOL = input('Input the function convergence tolerance: ');
XTOL = input('Input the interval convergence tolerance: ');

if (f(x0) == f(x1))
  error('Starting values are not valid.');
end

% Note that x0 is in x(1), x1 is in x(2) and so on
x = [x0, x1];

fx_2 = f(x0);
fx_1 = f(x1);
f_eval = 2;

for k = 3:(KTOL+2)
  % Storing the x_k's in x(k) according to the index offset mentioned above
  x(k) = x(k-1) - (fx_1 * (x(k-1) - x(k-2))) / (fx_1 - fx_2);
  fx = f(x(k));
  f_eval = f_eval + 1;
  if (fx == fx_1)
    % Function value of iteration is the same as previous iteration
    break;
  end
  if (abs(fx) <= FTOL)
    break;
  end
  if (abs(x(k) - x(k-1)) <= (1 + abs(x(k)))*XTOL)
      break;
  end
  % Although this check should theoretically never happen...
  if (k > (KTOL+3))
    fprintf('Method failed.');
  end
```

```
  fx_2 = fx_1;
  fx_1 = fx;
end

fprintf('Number of iterations: %d\n', k-2);
fprintf('Number of function evaluations: %d\n', f_eval);
fprintf('Estimated root x = %d and f(x) = %d\n', x(k), fx);
```

Christopher Wong
999234204
MAT 128A
Programming Project 1 **2/1/2017**

NEWTON.m

```
% Problem 1 NEWTON (Newton's method)
% clear
format long e

% cd 'C:\Users\Christopher\Desktop\MAT 128B\Project 1';

% Taking the input and converting it into a function
input_func = input('Evaluate f(x) = ', 's');
f = str2func(strcat('@(x)', input_func));

input_func_prime = input('f''(x) = ', 's');
f_prime = str2func(strcat('@(x)', input_func_prime));

x0 = input('Input the initial guess: ');
KTOL = input('Input the iteration bound: ');
FTOL = input('Input the function convergence tolerance: ');
XTOL = input('Input the interval convergence tolerance: ');

if (f_prime(x0) == 0)
  error('Derivative at initial guess is 0');
else
  x = x0;
  f_prime_eval = 0;
end

fx_1 = f(x0);
f_eval = 1;

for k = 2:(KTOL+1)
  if (f_prime(x(k-1)) == 0)
    fprintf('Stopped at iteration: %d\n', k-1);
    error('Derivative at estimate is zero.')
  end
  % Storing the x_k's in x(k) according to the index offset mentioned above
  x(k) = x(k-1) - (fx_1 / f_prime(x(k-1)));
  fx = f(x(k));
  f_eval = f_eval + 1;
  f_prime_eval = f_prime_eval + 1;
  if (abs(fx) <= FTOL)
    break;
  end
  if (abs(x(k) - x(k-1)) <= (1 + abs(x(k)))*XTOL)
      break;
  end
  % Although this check should theoretically never happen...
  if (k > (KTOL+2))
```

```
    fprintf('Method failed.');
  end
  fx_1 = fx;
end

fprintf('Number of iterations: %d\n', k-1);
fprintf('Number of function evaluations: %d\n', f_eval);
fprintf('Number of derivative evaluations: %d\n', f_prime_eval);
fprintf('Estimated root x = %d and f(x) = %d\n', x(k), fx);
```

Christopher Wong
999234204
MAT 128A

NEWTON_DAMPING.m

```
% Problem 1 NEWTON_DAMPING (Newton's method with damping)
% clear
format long e

% cd 'C:\Users\Christopher\Desktop\MAT 128B\Project 1';

% Taking the input and converting it into a function
input_func = input('Evaluate f(x) = ', 's');
f = str2func(strcat('@(x)', input_func));

input_func_prime = input('f''(x) = ', 's');
f_prime = str2func(strcat('@(x)', input_func_prime));

x0 = input('Input the initial guess: ');
KTOL = input('Input the iteration bound: ');
FTOL = input('Input the function convergence tolerance: ');
XTOL = input('Input the interval convergence tolerance: ');

if (f_prime(x0) == 0)
  error('Derivative at initial guess is 0');
else
  x = x0;
  f_prime_eval = 0;
end

fx_1 = f(x0);
f_eval = 1;

% Setting the initial (and best) value of lambda
lambda = 1;

for k = 2:(KTOL+1)
  if (f_prime(x(k-1)) == 0)
    fprintf('Stopped at iteration: %d\n', k-1);
    error('Derivative at estimate is zero.')
  end
  lambda = min(1, 2*lambda);
  for j = 1:100
    if (abs(fx_1 - lambda * (fx_1) / f_prime(x(k-1))) <= ...
        (1 - lambda/2)*abs(fx_1))
      x(k) = x(k-1) - (lambda * (fx_1) / f_prime(x(k-1)));
    else
      lambda = lambda/2;
    end
  end
  fx = f(x(k));
```

```
  f_eval = f_eval + 1;
  f_prime_eval = f_prime_eval + 1;
  if (abs(fx) <= FTOL)
    break;
  end
  if (abs(x(k) - x(k-1)) <= (1 + abs(x(k)))*XTOL)
      break;
  end
  % Although this check should theoretically never happen...
  if (k > (KTOL+2))
    fprintf('Method failed.');
  end
  fx_1 = fx;
end

fprintf('Number of iterations: %d\n', k-1);
fprintf('Number of function evaluations: %d\n', f_eval);
fprintf('Number of derivative evaluations: %d\n', f_prime_eval);
fprintf('Estimated root x = %d and f(x) = %d\n', x(k), fx);
```