Project/Test 1

Michael McDonnell
Virginia Polytechnic and State University

Math 4414 Issues in Scientific Computing

Dr. Tao Lin

October 18, 2015

1)

Define a function $f(x)$ by $f(x) = x^3 - (1.35 - e^{-8x})x^2 + 0.5x - 0.0375$. To find the zeros of $f(x)$ in the interval $[a, b] = [0, 1]$, the preferred method (given it is a single variable function and the consequent simplicity) to find the zeros of the function (where $f(x)$ intercepts the x-axis) is to plot sufficient points, $(z_i, f(z_i)$ where $a \leq z_i \leq b$.

To employ the bisection root finding method, we observe the approximate points of intersection of $f(x)$ with the x-axis, and select an interval (α, β) encompassing the x-axis intersection. This interval (α, β) must satisfy the bisection criterion $f(\alpha)f(\beta) < 0$. This rigorously insures that the selected interval indeed contains a root for $f(x)$.

This procedure was conducted, and via the plot there were three roots on our given interval $[0, 1]$. The following table presents the initial interval (α, β), the corresponding root subject to an error tolerance of 10^{-12}, the residual (the value of $f(x)$ evaluated at found zero), and the number of iterations required to satisfy the error tolerance.

Interval	Zero	Residual	Iteration
$(0, 0.2)$	0.086379665521235	$2.475381011279865e-13$	37
$(0.5, 0.6)$	0.541834901824040	$-4.905797990062411e-15$	36
$(0.7, 0.8)$	0.740477203870978	$3.032296636007459e-15$	36

2)

(a) Define a function $f(x) = sin(0.1 + (4.5 - e^{-5x})x^2 - x^3)$. As per instruction, the Newton method is used to find the zero of the function closest to $x = 0.9$. This method requires the function, an initial guess $x^{(0)} = 0.9$, and the derivative of the function. Using MATLAB's diff command:

```
f =
    @(x)sin(0.1+(4.5-exp(-5*x))*x^2-x^3)

diff(f,x)
ans =
-cos(x^2*(exp(-5*x) - 9/2) + x^3 - 1/10)*(2*x*(exp(-5*x) - 9/2) - 5*x^2*exp(-5*x)...
 + 3*x^2)
```

Which is the derivative of $f(x)$. We can now prepare a function in MATLAB describing the derivative. With the error tolerance, $tol = 10^{-12}$, and $n = 2$ which will yield $x^{(2)}$, we can call the newtonsys function.

Zero	Error	Residual	Iterations
0.923463778942576	$3.131619790868825e-04$	$1.652376421131729e-07$	2

Based on the residual and how close to zero $f(x)$ is when evaluated at our second iteration root, $x^{(2)}$, one could surmise that this is an acceptable root. However, our error tolerance is far from satisfied. Ultimately, the precision of this root does not meet our error tolerance and must be rejected.

(b) Using the same function $f(x)$; initial guess for the root $x^{(0)}$; and error tolerance $tol = 10^{(-12)}$, we can set the limit for our iteration count n to something that we think will be sufficient for convergence. In this case, let $n = 100$. Running the newtonsys function with these new conditions,

Zero	Error	Residual	Iterations
0.923463750338398	$2.863041713090813e-16$	$2.863041713090813e-16$	4

3.

(a) The following MATLAB code was used to write the given system of equations in a form that satisfies

$$\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} Q_1 \\ Q_2 \\ Q_3 \\ p \end{pmatrix}$$

```
function [f] = proj3(x,gamma,c1,c2,c3,z1,z2,z3)
n = length(x);
f = zeros(n,1);
f(1) = x(1) + x(2) + x(3);
f(2) = x(4)./gamma + c1.*sign(x(1)).*(x(1)).^2 + z1;
f(3) = x(4)./gamma + c2.*sign(x(2)).*(x(2)).^2 + z2;
f(4) = x(4)./gamma + c3.*sign(x(3)).*(x(3)).^2 + z3;
end
```

Which yields the system of equations $\vec{F}(\vec{x})$:

$$\begin{aligned} f_1(\vec{x}) &= x_1 + x_2 + x_3 = 0 \\ f_2(\vec{x}) &= \tfrac{x_4}{\gamma} + c_1 sign(x_1)(x_1)^2 + z1 = 0 \\ f_3(\vec{x}) &= \tfrac{x_4}{\gamma} + c_2 sign(x_2)(x_2)^2 + z2 = 0 \\ f_4(\vec{x}) &= \tfrac{x_4}{\gamma} + c_3 sign(x_3)(x_3)^2 + z3 = 0 \end{aligned}$$

(b) The following is the matrix of partial derivatives, the Jacobian, of the system of equations $\vec{F}(\vec{x})$:

$$\begin{pmatrix} 1 & 1 & 1 & 0 \\ c_1\Big((2x_1^2)\delta(x_1) + 2x_1 sign(x_1)\Big) & 0 & 0 & \frac{1}{\gamma} \\ 0 & c_2\Big((2x_2^2)\delta(x_2) + 2x_2 sign(x_2)\Big) & 0 & \frac{1}{\gamma} \\ 0 & 0 & c_3\Big((2x_3^2)\delta(x_3) + 2x_3 sign(x_3)\Big) & \frac{1}{\gamma} \end{pmatrix}$$

(c) For the given parameters:

$$\gamma = 9790,\ z_1 = -19,\ z_2 = -7,\ z_3 = 2$$
$$c_1 = 3.7220,\ c_2 = 64.5452,\ c_3 = 21.2734$$

The MATLAB output is as follows:

```
zero =
   1.0e+05 *
   0.000011850208651
  -0.000003239424598
  -0.000008610784052
   1.348405139437306
res =
     3.552713678800501e-15
niter =
     9
err =
     3.720137721947402e-12
```

That is, the zeros x_i are:

$$x_1 = (1.0e+05) * 0.000011850208651$$
$$x_2 = (1.0e+05) * -0.000003239424598$$
$$x_3 = (1.0e+05) * -0.000008610784052$$
$$x_4 = (1.0e+05) * 1.348405139437306$$

Where $1.0e+05 = 100000$.

Even though each initial $\{Q_i, p\} = \{x_i\}$ is only an approximation, the resulting root is a zero precise up to MATLAB's precision with respect to the initial approximation values. Thus each x_i is *exact* in regard to initial approximate data.

4. The following table presents the results for various forms of polynomial interpolation evaluated at specific points (latitude 17 and latitude 60) to give an extrapolated estimation based on trends present in the data set.

Methods	Temp. at latitude 17	Temp. at latitude 60
LI, partial data	5.146160000000000	6.155865805041152
LI, all data	5.057404599374643	6.181254744529711
CSI, partial data	5.183957142857143	6.021325231481481
CSI, all data	5.057830773785799	6.040235973708933

The Lagrange interpolation technique using partial data produced results that could be reasonably accepted as 'close'. For example, at Latitude 15 the temperature is 5.02. At latitude 25 the temperature is 5.3. The partial data Lagrange interpolation produced a result of approximately 5.14 for latitude 17. Since 17 is between 15 and 25, and the extrapolated temperature 5.14 is between 5.02 and 5.3, one can permit this an acceptable answer since the general trend is increasing in the neighborhood of those data points. However, this argument does not hold for latitude 60, as it exceeds the temperature given for latitude 65 significantly.

For the all-data Lagrange interpolation, by similar observation, the latitude 17 extrapolation is more acceptable than the partial data interpolation.(since 17 is closer to 15 than 23, and temp. for latitude 17 is closer to temp. for latitude 15). In agreement with the larger oscillations from more data points under Lagrange interpolation, latitude 60 under all-data LI is even further outside the expected temperature range. It should be noted that a larger node set for LI will generally produce 'closer' results towards the center of the ordered data set - it is not until the boundary values of the data set are approached when the wild oscillations manifest from too many nodes. This can be countered by the use of Chebyshev nodes.

For the partial data cubic spline interpolation (PD-CSI), one can draw similarities between PD-LI for latitude 17. There just isn't enough data provided to give an outstanding result; however, still acceptable.

CSI does not fall prey to the catch-22 of more data points for interpolation. The all data CSI produces great results, and a graph shows no indication of unprecedented oscillations towards the endpoints.

5.

(a) For N = 5, the Jacobian for the nonlinear system was found by the MATLAB code:

```
function [J] = proj5_Jacobian(u,N)
J = zeros(N,N);
h = 1/(N+1);

J(1,1) = -2 + 2*(h^2)*u(1);
J(1,2) = 1;
```

```
for row = 2:N-1
    J(row,row-1) = 1;
    J(row,row) = -2 + 2*(h^2)*u(row);
    J(row,row+1) = 1;
end

J(N,N-1) = 1;
J(N,N) = -2 + 2*(h^2)*u(N);

end
```

Which yields the following Jacobian matrix:

$$\begin{pmatrix} -2+2u_1h^2 & 1 & 0 & 0 & 0 \\ 1 & -2+2u_2h^2 & 1 & 0 & 0 \\ 0 & 1 & -2+2u_3h^2 & 1 & 0 \\ 0 & 0 & 1 & -2+2u_4h^2 & 1 \\ 0 & 0 & 0 & 1 & -2+2u_5h^2 \end{pmatrix}$$

(b) Let N = 400.

Method	u_{172}	Residual	NOI
Broyden Method	0.975182390137037	1.330411598710643e-15	7
Newton Method	0.975182390137073	1.344373691338019e-15	5

6.

(a) Using the LI on the data set given, we can find the approximate x-y coordinate values predicted by the interpolation for various t-values not given as data points. The following table presents those x-y coordinates for when $t = 1.5$, and $t = 2.5$.

Time	x	y
1.5	2.250000000000000	3
2.5	3.541666666666667	2.291666666666667

(b) See the attached plot.

The plots are similar. The only noticeable difference is that on my plot, the lower trajectory returning from $(4, 4)$ to $(0, 0$ is slightly more bulbous on the underside and is much closer to $y = -1.5$ between $x = 1$ and $x = 1.5$. Otherwise, they are similar enough by eye not to be told apart.

(c) The following two t-values are for when the x-coordinate is 2.75. By evalutating the LI polynomial at these t values, we can recover the approximate (x, y) point. $t_1 = 1.658312395178109$, $t_2 = 3.208099243548531$. For t_1, $x = 2.750000000001356$, and $y = 3.316624790356218$. For t_2, $x = 2.749999999999137$, $y = 0.585206051616077$.