Solution of Week 10 Lab (Prepared by Yuan Yin)

November 6, 2020

1 Exercise 1:

1.1 Levenberg-Marquardt Method:

(a).

Run ‘bandem.m’ to see the output.

(b).

- As one can see from the output, for the good guess, numf = 21 and numg = 11. However, for the bad guess, we need more function evaluations (numf = 77 and numg = 39), i.e. less efficient, to get the solution.
- For the good guess, λ starts from a small value and keeps decreasing. However, we know from lecture that a small λ signals a move like Gauss-Newton step. This actually makes sense since a good guess is close to the real solution, and Gauss-Newton method is expected. While for the bad guess, λ starts from a huge number. This is because initially, our guess is far from the solution, and we need to make the step more like Steepest Descent. As we move increasingly closer to the real root, we need to decrease the value of λ to make the step more like Gauss-Newton
- L-M better than Damped G-N better than Undamped G-N.

1.2 The Optimization toolbox:

(a).

- Run ‘bananaScript.m’ to see the ‘Stopping Criteria Details’.
- Why we can fool MATLAB into minimising the ‘banana’ function by calling *lsqnonlin*? —— This is because the ‘banana’ function happens to be the cost function of some other function! We can use *lsqnonlin* to minimise the cost function, i.e. find the min of the ‘banana’ function!

(b).

Read ‘healthScript.m’.

2 Exercise 2: Euler’s Method for a Scalar 1st Order ODE:

2.1 Error Propagation in Euler’s Method:

Run the codes and try to understand the outputs.

One Interesting Point You May like to Know:

In lecture, we know that when nearby solutions approach each other, $\frac{\partial f}{\partial y} < 0$ and vice versa. However, why we have such relationship?

Explanation:

Suppose we change I.C. from y_0 to $y_0 + \epsilon$, which generates our new solution, y_ϵ. We assume w.l.o.g. that the new solution is larger than the original one, i.e. $y_\epsilon - y = \Delta y > 0$.

$\Rightarrow \frac{d}{dt}(y_\epsilon - y) = \frac{d(\Delta y)}{dt} = f(t, y_\epsilon) - f(t, y) = \frac{\partial f}{\partial y} \times \Delta y$ (A little bit hand waving, but will work if we suppose Δy is small.)

Since $\Delta y > 0$, $\frac{d(\Delta y)}{dt}$ and $\frac{\partial f}{\partial y}$ are of the same sign.

Then $\frac{\partial f}{\partial y} < 0 \Rightarrow \frac{d(\Delta y)}{dt} < 0 \Rightarrow$ nearby solutions approach each other as time evolves (and vice versa).

2.2 Accuracy and Stability:

(a).

Read the code.

(b).

Run the code and try to understand the outputs.

Note that the aim of this part is to check that the output results are consistent with the theory, i.e. GE $\sim$ O(h), where $h = \frac{tspan}{n}$.

(c).

If you are greedy, you can get a small taste of Lect 20 in advance.

$(1 + hJ)$ is called the propagation factor —— the Global Error at t_{k+1} = the Global Error at t_k, propagated with factor (1+hJ), i.e. $GE_{k+1} \sim (1 + hJ)GE_k$. Note that $J = \frac{\partial f}{\partial y}$.

If we numerical errors to decrease, then we require $|1 + hJ| < 1$.

- Since $h = \frac{tspan}{n} > 0$, errors must grow if $J > 0$. This is the reason why in the first figure, errors blow up catastrophically no matter what values we choose for n;
- In the third figure, the error blows up when $n = 10$. This is because for $\dot{y} = -100(y - sin(t)) + cos(t)$, $J = -100$. When $n = 10$, $h = \frac{tspan}{n} = \frac{1}{10}$. $\Rightarrow |1 + hJ| = |1 + \frac{1}{10} \times (-100)| = 9 > 1$.

2.3 Try Yourself:

[1]:
```
%%file TryYourselfDriver.m

function TryYourselfDriver

clc

y0 = 1;
tspan = [0, 2];
```

```
fname = @(t, y) y.^2;
n1 = 10;
n2 = 100;

[tvals,yvals1] = FixedEuler(fname,tspan,y0,n1);
yvals1

[tvals,yvals2] = FixedEuler(fname,tspan,y0,n2);
yvals2

fprintf('However, from first year math, y = 1/(1 - t).\n');
exact_sol = 1 / (1 - 2);
fprintf('Thus, our exact solution at t = 2 should be %6.4f.\n\n', exact_sol);


end
```

```
Created file '/Users/RebeccaYinYuan/MAST30028 Tutorial Answers Yuan Yin/WEEK
10/TryYourselfDriver.m'.
```


[2]:
```
TryYourselfDriver
```



```
yvals1 =

   1.0e+05 *

    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0001
    0.0002
    0.0009
    0.0166
    5.5163


yvals2 =

  1.0e+278 *

    0.0000
    0.0000
    0.0000
    0.0000
```

```
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
```

```
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
1.3057
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
```

```
    Inf

However, from first year math, y = 1/(1 - t).
Thus, our exact solution at t = 2 should be -1.0000.
```

What is going on?

From first year mathematics, the exact solution should be $y = \frac{1}{1-t}$. It is important to notice that y exhibits an asymptotic behaviour at t = 1, i.e.

$$\lim_{t \to 1^-} y \to \infty \quad and \quad \lim_{t \to 1^+} y \to -\infty.$$

However, the Jacobian, $J = \frac{\partial f}{\partial y} = 2y$, will explode near $t = 1$, meaning that NO method works at all.

Also, note that the situation for $n = 100$ is even worse than $n = 10$. This is because in $n = 10$ case, our step size is bigger, and we can step across the asymptote more quickly (i.e. The larger the step size, the more points we skip near $t = 1$.). In contrast, if $n = 100$, there are more points/steps distributed near $t = 1$, which makes J huge at many steps and the question becomes terribly conditioned.

3 Exercise 3: Euler's Method for systems:

3.1 Converting to Systems:

(a).

$$\ddot{\theta} + sin(\theta) = 0 \Rightarrow \ddot{\theta} = -sin(\theta)$$

By setting $\omega = \dot{\theta}$, we have a system of first order OEDs:

$$\begin{cases} \dot{\theta} = \omega \\ \dot{\omega} = -sin(\theta) \end{cases}$$

If we go one step further by setting

$$\vec{y} = \begin{bmatrix} \theta \\ \omega \end{bmatrix}$$

Then we have

$$\vec{y'} = \begin{bmatrix} \theta' \\ \omega' \end{bmatrix} = \begin{bmatrix} \omega \\ -sin(\theta) \end{bmatrix} = \begin{bmatrix} \vec{y}(2) \\ -sin(\vec{y}(1)) \end{bmatrix}$$

(b).

$$\dddot{y} + 3x^2y\ddot{y} - sin(y)\dot{y}^2 \Rightarrow \dddot{y} = -3x^2y\ddot{y} + sin(y)\dot{y}^2 + cos(x)$$

Using the similar method as in part(a) above, we have:

$$\begin{cases} \dot{y} = \omega \\ \dot{\omega} = \eta \\ \dot{\eta} = -3x^2y\eta + sin(y)\omega^2 + cos(x) \end{cases}$$

If we go one step further by setting

$$\vec{u} = \begin{bmatrix} y \\ \omega \\ \eta \end{bmatrix}$$

Then we have

$$\vec{u'} = \begin{bmatrix} y' \\ \omega' \\ \eta' \end{bmatrix} = \begin{bmatrix} \omega \\ \eta \\ -3x^2 y\eta + sin(y)\omega^2 + cos(x) \end{bmatrix} = \begin{bmatrix} \vec{u}(2) \\ \vec{u}(3) \\ -3x^2\vec{u}(1)\vec{u}(3) + sin(\vec{u}(1))\vec{u}(2)^2 + cos(x) \end{bmatrix}$$

(c).

Similarly,

$$\begin{cases} \dot{x} = \omega \\ \dot{\omega} = -\frac{x}{r^3} = -\frac{x}{(x^2+y^2)^{\frac{3}{2}}} \\ \dot{y} = \eta \\ \dot{\eta} = -\frac{y}{r^3} = -\frac{y}{(x^2+y^2)^{\frac{3}{2}}} \end{cases}$$

If we go one step further by setting

$$\vec{u} = \begin{bmatrix} x \\ \omega \\ y \\ \eta \end{bmatrix}$$

Then we have

$$\vec{u'} = \begin{bmatrix} x' \\ \omega' \\ y' \\ \eta' \end{bmatrix} = \begin{bmatrix} \omega \\ -\frac{x}{(x^2+y^2)^{\frac{3}{2}}} \\ \eta \\ -\frac{y}{(x^2+y^2)^{\frac{3}{2}}} \end{bmatrix} = \begin{bmatrix} \vec{u}(2) \\ -\frac{\vec{u}(1)}{(\vec{u}(1)^2+\vec{u}(3)^2)^{\frac{3}{2}}} \\ \vec{u}(4) \\ -\frac{\vec{u}(3)}{(\vec{u}(1)^2+\vec{u}(3)^2)^{\frac{3}{2}}} \end{bmatrix}$$

3.2 Vectorized Euler's Method

[1]:
```
%%file DriverVecEuler.m

function DriverVecEuler
%% PART A

clc

f_partA = @(t, y) [y(2); -sin(y(1))];
t_span = [0, 20];
n = 2000;
y0_partA = [1, 0];

[tvals_partA,yvals_partA] = FixedEulerVec(f_partA, t_span, y0_partA,n);

% Is this system periodic?
% We need to produce a phase plot:
```

```
x_partA = yvals_partA(:, 1); % theta values
y_partA = yvals_partA(:, 2); % theta dash values

figure(1);
plot(x_partA, y_partA);
xlabel('theta');
ylabel('theta dash');

fprintf('As one can see from the phase plot, the system A is periodic!\n\n');



%% PART B


f_partB = @(t, y) [-(8 / 3) * y(1) + y(2) * y(3); -10 * y(2) + 10 * y(3); -y(1)
↪* y(2) + 28 * y(2) - y(3)];
t_span = [0, 20];
n = 2000;
y0_partB = [1, 1, 1]

[tvals_partB,yvals_partB] = FixedEulerVec(f_partB, t_span, y0_partB,n);

% Is this system periodic?
% We need to produce a phase plot:
x_partB = yvals_partB(:, 1); % y_(1) values
y_partB = yvals_partB(:, 2); % y_(2) values
z_partB = yvals_partB(:, 3); % y_(3) values

figure(2);
plot3(x_partB, y_partB, z_partB);
xlabel('y_(1)');
ylabel('y_(2)');
zlabel('y_(3)');

fprintf('As one can see from the phase plot, the system B is periodic!\n\n');


%% Note:

% Now you can change n to 20000 to investigate furthur!

end
```

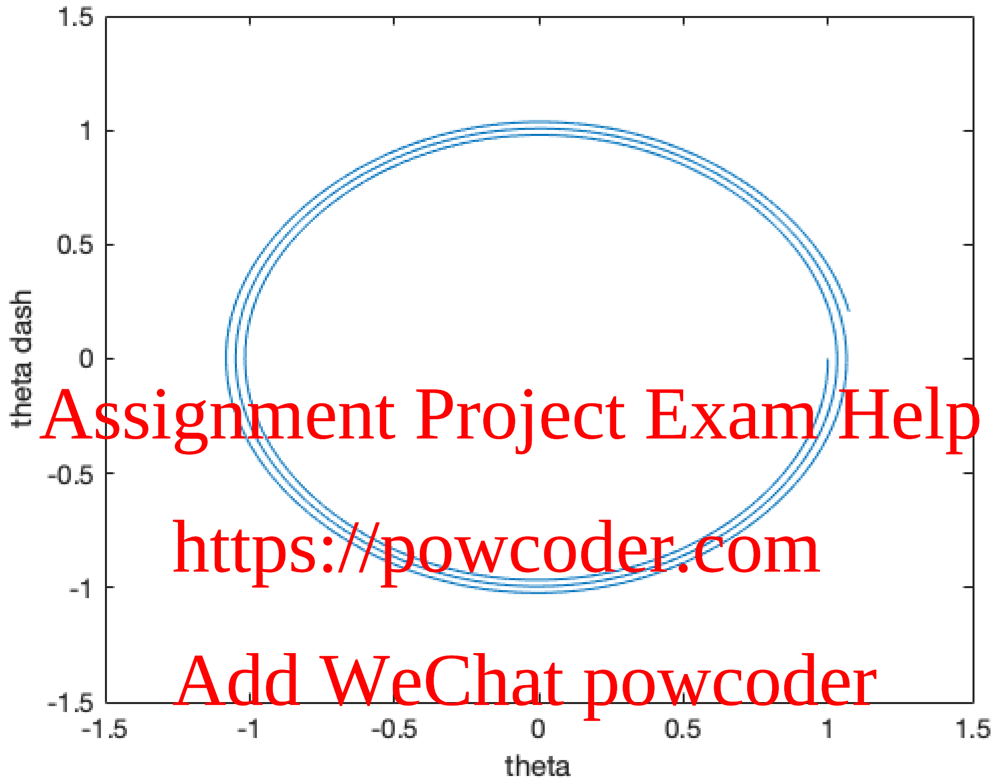


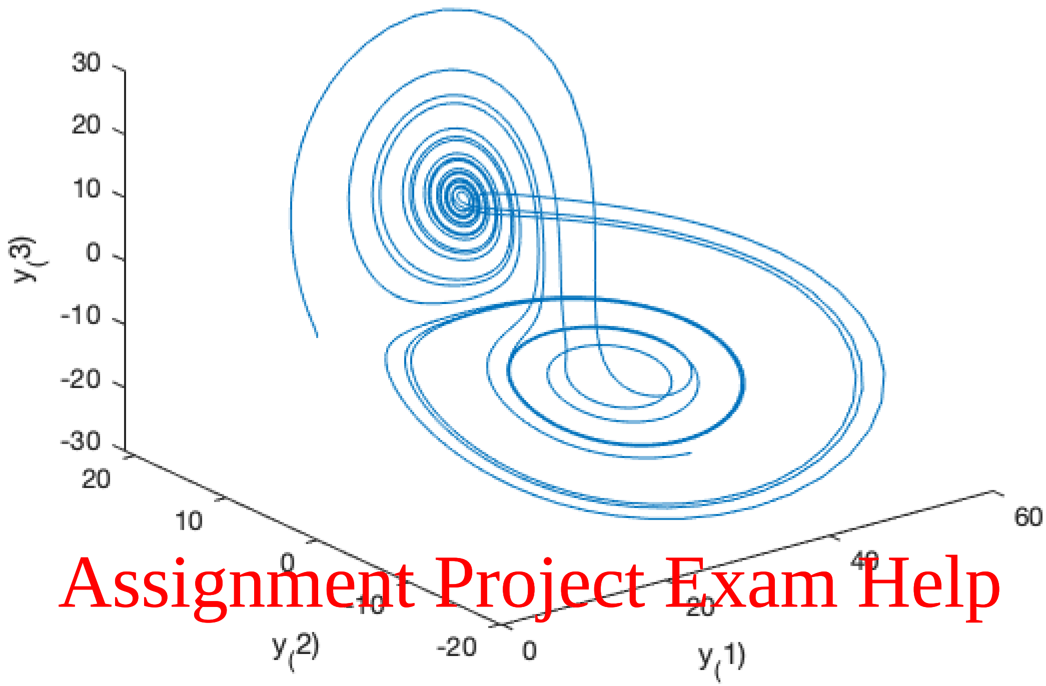
```
Created file '/Users/hailongguo/lib/Dropbox/Teaching/subject/UoM/2020/MAST30028/
other/MAST30028Tutorial Answers Yuan Yin/Tute Answers ipynb
files/DriverVecEuler.m'.
```

[3]:
```
DriverVecEuler
```

```
As one can see from the phase plot, the system A is periodic!

As one can see from the phase plot, the system B is periodic!
```





NOTE:

- For $\ddot{\theta} + sin(\theta) = 0$:

From part (a) in section 3.1, we have:

$$\vec{y'} = \begin{bmatrix} \theta' \\ \omega' \end{bmatrix} = \begin{bmatrix} \omega \\ -sin(\theta) \end{bmatrix} = \begin{bmatrix} \vec{y}(2) \\ -sin(\vec{y}(1)) \end{bmatrix}$$

Therefore, our input 'fname' $= @(t, y)\ [y(2); -sin(y(1))]$

- For

$$\begin{cases} \dot{y}_1 = -\frac{8}{3}y_1 + y_2y_3 \\ \dot{y}_2 = -10y_2 + 10y_3 \\ \dot{y}_3 = -y_1y_2 + 28y_2 - y_3 \end{cases}$$

If we set $\vec{y} = [y_1; y_2; y_3]$, we have:

$$\vec{y'} = \begin{bmatrix} -\frac{8}{3}\vec{y}(1) + \vec{y}(2)\vec{y}(3) \\ \\ -10\vec{y}(2) + 10\vec{y}(3) \\ \\ -\vec{y}(1)\vec{y}(2) + 28\vec{y}(2) - \vec{y}(3) \end{bmatrix}$$

Therefore, our input 'fname' = $@(t, y)$ $[-\frac{8}{3} * y(1) + y(2) * y(3); -10 * y(2) + 10 * y(3); -y(1) * y(2) + 28 * y(2) - y(3)]$