

Math 216 Differential Equations

Lab 3: Higher-Order Numerical Methods

Goals

In this lab you will compare the order of accuracy of various numerical methods; that is, you will test-drive some new solvers. You will implement an improved Euler's method and a fourth-order Runge-Kutta method, in addition to your Euler's method program from Lab 2. The example from Lab 2 of an RC circuit with AC voltage provides a case-study for comparing these three methods.

Prelab assignment

Before arriving in the lab, answer the following questions. You will need your answers in lab to work the problems, and your recitation instructor may check that you have brought them. These problems are to be handed in as part of your lab report.

Consider the initial-value problem

$$Q'(t) = -\frac{1}{2}Q(t) + 5, \quad Q(0) = 2. \quad (1)$$

1. Solve this problem for $Q(t)$ by noting that the differential equation is separable.
2. Table 1 below shows the resulting values from N steps of some numerical method to approximate $Q(t)$ over the interval $0 \leq t \leq 2$. Copy the table and fill in the exact values using your solution.
3. Calculate the errors and write them in Table 1. Note that error means $|\text{approximate} - \text{exact}|$. Also copy Table 2 and fill in the errors (only for $t = 2$).
4. The step size h is obtained by dividing the interval over which the solution is computed into N equal parts. Calculate the step sizes h for the three rows in Table 2.
5. Plug in the values from Table 2 into the following equations,

$$E_1 = Ch_1^\alpha \quad \text{and} \quad E_2 = Ch_2^\alpha$$

and solve them for the exponent α . Hint: taking logarithms makes the equations easier to solve.

6. Repeat the computations of question 5 using the values h_2, h_3 and E_2, E_3 instead of h_1, h_2 and E_1, E_2 . The value of α you obtain should be close to that obtained in question 5; you are repeating the computation as a check on your previous calculation.

7. Based on your answers to questions 5 and 6 which numerical method do you think was used? (Refer to sections 2.5 and 2.6 of the textbook.)

t	Approximate Q	Exact Q	Error
$N = 10$			
1.0	5.143393877		
2.0	7.051672121		
$N = 60$			
1.0	5.147640987		
2.0	7.056826501		
$N = 360$			
1.0	5.147751596		
2.0	7.056960678		

Table 1: Approximate Solution to the initial-value problem (1)

N	h	Error
10	$h_1 =$	$E_1 =$
60	$h_2 =$	$E_2 =$
360	$h_3 =$	$E_3 =$

Table 2: Errors for $t = 2$

In the lab

In this lab we will see how higher-order numerical methods can be implemented in Matlab as easily as Euler's method was implemented in Lab 2. We are still concerned with the general initial-value problem

$$\frac{dy}{dt} = f(t, y), \quad y(a) = y_0. \quad (2)$$

In Lab 2 you implemented a solver for this initial-value problem using Euler's method. In this lab you will implement solvers using the improved Euler method and the fourth-order Runge-Kutta method. To begin Lab 3, launch Matlab and make sure your working directory is the subdirectory of `~/Desktop/IFS` in which you saved your files from Lab 2. You should see the names of the `.m` files you saved in Lab 2 in the window pane to the left of the Command Window.

Entering in a Matlab program for the improved Euler method

Create a new `.m` file in Matlab called `impEULER.m` that will contain your solver implementing the improved Euler method. Enter into `impEULER.m` the following commands:

```
clear t                % Clears old time steps and
clear yI              % yI values from previous runs
a=0;                  % Initial time
b=1;                  % Final time
N=10;                 % Number of time steps
y0=1;                 % Initial value y(a)
h=(b-a)/N;           % Time step
t(1)=a;
yI(1)=y0;
for n=1:N              % For loop, sets next t,yI values
    k1=f(t(n),yI(n)); % Slope 1
    u(n)=yI(n)+h*k1;  % Predictor
    t(n+1)=t(n)+h;
    k2=f(t(n+1),u(n)); % Slope 2
    yI(n+1)=yI(n)+h*(k1+k2)/2; % Corrector
end
plot(t,yI)
title(['Improved Euler Method using N=',num2str(N)...
,' steps, by MYNAME'])
```

Note: the ellipsis (...) in the last line tells Matlab to continue the command onto the next line of text. If included in the middle of a line of text, it will cause an error.

In this program, substitute your own name for `MYNAME`. Notice that `impEULER.m` refers to the same file `f.m` that `EULER.m` does, so you can solve the same differential equation with both `EULER.m` and `impEULER.m`. While the vector of approximate solutions in `EULER.m`

was called y , the vector of approximate solutions in `impEULER.m` is called yI , so you can distinguish and compare the output of the two numerical methods.

Entering in a Matlab program for the fourth-order Runge-Kutta method

Next we implement a Runge Kutta method. Create a new `.m` file called `RK.m` and enter in the following commands:

```
clear t
clear yRK
a=0;
b=1;
N=10;
y0=1;
h=(b-a)/N;
t(1)=a;
yRK(1)=y0;
for n=1:N                                % Define the slopes k
    k1=f(t(n),yRK(n));                    % Slope 1
    k2=f(t(n)+h/2,yRK(n)+h*k1/2);        % Slope 2
    k3=f(t(n)+h/2,yRK(n)+h*k2/2);        % Slope 3
    k4=f(t(n)+h,yRK(n)+h*k3);            % Slope 4
    k=(k1+2*k2+2*k3+k4)/6;                % Weighted average of slopes
    t(n+1)=t(n)+h;                        % Update time
    yRK(n+1)=yRK(n)+h*k;                  % Update y
end
plot(t,yRK)
title(['Runge Kutta Method using N=',num2str(N)...
,' steps, by MYNAME'])
```

Once again, you should substitute your own name for `MYNAME`. This program again refers to `f.m` and saves the approximate solution in a vector called `yRK` which may be compared with y produced by `EULER.m` and yI produced by `impEULER.m`.

A Matlab program for visualizing the errors

Finally, to compare the errors of the three methods you have implemented in Matlab, it will be helpful to graph all three errors on the same axes. You will now write a Matlab program called `ErrorPlot.m` to do this. The errors will be plotted on a logarithmic scale, where the logarithm of the error is plotted versus time. A “semilog” plot such as this is particularly helpful when you wish to investigate a function, or several functions, that take on values spanning many orders of magnitude. After creating a `.m` file called `ErrorPlot.m`, enter the following commands:

```

EULER                % Runs Euler
impEULER             % Runs impEULER
RK                  % Runs RK
yexact=yE(t);       % Compute exact solution
errE=abs(yexact-y); % Euler method error
errI=abs(yexact-yI); % Improved Euler error
errRK=abs(yexact-yRK); % Runge Kutta error
semilogy(t,errE,':',t,errI,'-.',t,errRK,'-')
legend('Euler','impEULER','RK')
xlabel('t')          % Labels 'x' axis
ylabel('Error')      % Labels 'y' axis
title(['Errors using N=',num2str(N),' steps, by MYNAME'])

```

Substitute your name for MYNAME so your plots will be labelled with your name. Make sure at this time that all of your .m files are saved.

Lab problems

In this lab, you will numerically approximate solutions to the following two initial-value problems:

$$\frac{dy_1}{dt} = y_1, \quad y_1(0) = 1, \quad (3)$$

$$\frac{dy_2}{dt} = -5y_2 + .00585 \sin(120\pi t), \quad y_2(0) = 0. \quad (4)$$

1. Run EULER.m, impEULER.m and RK.m for $N = 10$, to solve the initial-value problem (3) on the interval $0 \leq t \leq 1$. (This is in preparation for Problem 2 to check that all three programs are running correctly; you do not have to print out the results.) To do this problem, make sure that for all three programs, the beginning includes the lines

```

a=0;
b=1;
N=10;
y0=1;

```

You must also make sure that the line of your program f.m which defines the value of f (that is, the right-hand side $f(t, y)$ of the differential equation for the initial-value problem (3)) is

```
f=y;
```

and the line in the file yE.m which defines the exact solution to the initial-value problem (3) is

```
yE=exp(t);
```

2. Run the program `ErrorPlot.m` to compare the errors created by using each method in approximating the solution to the initial-value problem (3). Print the graph of the error using 10 steps. Enter the errors at the indicated times in the appropriate column of Table 3 below.
3. Repeat Problem 2 using $N = 100$ and $N = 1000$ steps. Print these graphs of the error and use the data to complete Table 3. Note: you will need to modify the line `N=10;` in *all three* programs `EULER.m`, `impEULER.m`, and `RK.m`.
4. Conclude for each method that the error is proportional to h^α for some constant exponent α . The constant α is called the *order* of the method. Based on your calculations, what is the order for each method? (Recall the prelab calculations.)
5. Repeat Problems 2 and 3 for the second initial-value problem, (4). Solve on the interval $0 \leq t \leq 0.1$, using $N = 100$, $N = 200$, $N = 400$, and $N = 800$. To do this, you must change your file `f.m` to implement the right-hand side of the differential equation for the initial-value problem (4) using the line

```
f=-5*y+.00585*sin(120*pi*t);
```

and also the file `yE.m` should contain the exact solution of the initial-value problem (4):

```
omg=120*pi;
A=117*omg*.04/(20000*(1+(omg*.2)^2));
B=117*.2/(20000*(1+(omg*.2)^2));
yE=A*(exp(-t/.2)-cos(omg*t))+B*sin(omg*t);
```

Note that modified in this way, these files are the same as used in the second part of Lab 2; you may have saved them as `f2.m` and `yE2.m`.

Finally, the beginning of `EULER.m`, `impEULER.m`, and `RK.m` must be modified to incorporate the new stopping time and initial condition, as well as the number of steps:

```
a=0;
b=.1;
N=100;    % or 200, or 400, or 800
y0=0;
```

Plot the error as before, and include your data in Table 4 on the last page. Print the graphs of the error. Are these errors consistent with your conclusions from Problem 4? Why or why not?

6. When using the Runge-Kutta method to solve the initial-value problem (3) with $N = 1000$, what happened in your graph of the error for small values of t ? About how large was this error? How large is the roundoff error in Matlab? (To find out, type `eps` in the Matlab Command window.) Explain the possible role of roundoff error in that part of the graph.

7. Which method is the most efficient for solving initial-value problems for differential equations? Explain clearly how you interpreted the word “efficient” in answering the previous question (for example, some methods may be easier to program, and other methods may be able to achieve more accuracy using less computer time).

Time t	Number of Steps, N		
	10	100	1000
Euler's Method			
0.5			
0.86			
1.0			
Improved Euler Method			
0.5			
0.86			
1.0			
Runge-Kutta Method			
0.5			
0.86			
1.0			

Table 3: Error from solving the initial-value problem (3)

Note: your lab report for Lab 3 should consist of your solutions to the prelab problems, your solutions to lab problems 1–7 (including any graphs printed out), and a brief, original, conclusions paragraph summarizing what you have learned.

Time t	Number of Steps, N		
	100	200	400
Euler's Method			
0.05			
0.086			
0.1			
Improved Euler Method			
0.05			
0.086			
0.1			
Runge-Kutta Method			
0.05			
0.086			
0.1			

Table 4: Error from solving the initial-value problem (4)