Lesson 2: Falling Mass and Euler Methods

2.1 Applied Problem.

Consider a mass, which is falling through a viscous medium. One example is a falling rock, and a second example is a person in parachute. These two masses will have different speeds because of different air resistance. We would like to quantify these air resistances, and use them to predict the speed of the falling object.

2.2 Differential Equation Model.

Here we use Newton's law of motion with two forces: one from gravity, mg, and a second from the viscous medium, \(-ku^2\) (or, some quadratic function of u). The speed of the mass is denoted by \(u\), and the \(k\) is a constant, which reflects the mass and the viscosity of the medium. Newton's law of motion then gives the differential equation

\[
m \frac{du}{dt} = -mg + ku^2.
\]

As time evolves, the speed will approach \(u = (mg/k)^{1/2}\). More generally, the resistive force will be \(c \text{ abs}(u) + ku^2\) where the constants \(c\) and \(k\) are determined from physical experiments.

2.3 Method of Solution.

The general problem of solving or approximating the solution of the differential equation

\[
y' = g(t,y)
\]

can be approached via a variation of Euler method.
**Euler's Numerical Method for Approximating \( y' = g(t,y) \) and \( y(0) \) given.**

Let \( Y(i) \) be an approximation for \( y(i\cdot dt) \) given by

\[
Y(i + 1) = Y(i) + dt \cdot g(i\cdot dt, Y(i)).
\]

In order to formulate an improved variation of Euler's method, consider the integral form of the differential equation by integrating both sides from \( t = i\cdot dt \) to \( t = (i+1)\cdot dt \)

\[
y((i + 1) \cdot dt) - y(i \cdot dt) = \int_{i\cdot dt}^{(i+1)\cdot dt} g(t, y(t)) dt.
\]

Next approximate the integral by the trapezoid method and use the Euler method to approximate \( y((i+1)\cdot dt) \). This gives the following where \( Y(i+1) \) is over written by the second line, and \( Y(i) \) is the approximation of \( y(i\cdot dt) \).

**Improved Euler's Numerical Method for Approximating \( y' = g(t,y) \) and \( y(0) \) given.**

\[
Y(i+1) = Y(i) + dt/2 \cdot (g(i\cdot dt, Y(i)) + g((i+1)\cdot dt, Y(i+1))).
\]

The following Matlab code is stored in the m-file eulerr.m, and it illustrates the errors in the Euler and improved Euler methods for the cooling cup of coffee problem that was discussed in the previous two lessons. The error is the difference in the numerical solution and the exact solution

\[
\text{Error} = Y(i) - y(i\cdot dt).
\]
The table of errors indicates that the error for the Euler method is proportional to the step size, while the error for the improved Euler method is proportional to the square of the step size.

Matlab Code eulerr.m: Euler, Improved Euler and Exact.

```matlab
% This code compares the discretization errors.
% The Euler and improved Euler methods are used.
clear;
maxk = 8;      % number of time steps
T  = 1.0;      % final time
dt = T/maxk;
time(1) = 0;
u0 = 200.;     % initial temperature
c  = 1.;       % insulation factor
usur = 70.;    % surrounding temperature
uexact(1) = u0;
ueul(1)   = u0;
uiueul(1) = u0;
for k = 1:maxk
    time(k+1) = k*dt;
    % exact solution
    uexact(k+1) = usur + (u0 - usur)*exp(-c*k*dt);
    % Euler numerical approximation
    ueul(k+1) = ueul(k) + dt*c*(usur - ueul(k));
    % improved Euler numerical approximation
    utemp = uiueul(k) + dt*c*(usur - uiueul(k));
    uiueul(k+1)= uiueul(k) +
                  dt/2*(c*(usur - uiueul(k)) + c*(usur - utemp));
    err_eul(k+1) = abs(ueul(k+1) - uexact(k+1));
    err_im_eul(k+1) = abs(uiueul(k+1) - uexact(k+1));
end
plot(time, ueul,time, uexact, time, uiueul)
maxk
err_eul_at_T = err_eul(maxk+1)
err_im_eul_at_T = err_im_eul(maxk+1)
```
### Table: Discretization Errors

<table>
<thead>
<tr>
<th>KK</th>
<th>erreul</th>
<th>errimerr</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>3.1152</td>
<td>0.1370</td>
</tr>
<tr>
<td>16</td>
<td>1.5347</td>
<td>0.0326</td>
</tr>
<tr>
<td>32</td>
<td>0.7571</td>
<td>0.0080</td>
</tr>
<tr>
<td>64</td>
<td>0.3761</td>
<td>0.0020</td>
</tr>
</tbody>
</table>

2.4 Matlab Implementation.

The following two lines are in the function file `fmass.m`. This is for the falling mass problem where the mass $m = 1$, the gravitation constant $g = 32$, and the resistive force is $0.1*\text{abs}(y)$ where $y$ represents the speed of the mass.

```matlab
function fmass = fmass(t,y)
    fmass = -32 + 0.1*abs(y);
```

The improved Euler method is implemented in the m-file `imeuler.m`. The initial speed is stored in $y(1)$, and the initial time is stored in $t(1)$. The number of time steps is given by $KK$, the final time is given by $T$, and the step size is $h = T/KK$. The first line in the for loop is just the Euler substep, and the third line in the for loop is the improved Euler calculation.

```matlab
%your name, your student number, lesson number
clear;
y(1) = 0.;  % initial speed stored in first entry of array y
T = 50.;    % final time
KK = 200    % number of time steps
h = T/KK;   % time step size
t(1)= 0.;   % initial time stored in first entry of array t
```
for k = 1:KK
    y(k+1) = y(k) + h*fmass(t(k),y(k));
    t(k+1) = t(k) + h;
    y(k+1) = y(k) + .5*h*(fmass(t(k),y(k)) +
                        fmass(t(k+1),y(k+1)));
end
plot(t,y)
title('your name, your student number, lesson number')
xlabel('time')
ylabel('speed')

2.5 Numerical Experiments.

The first numerical experiment is with the resistive force equal to .1u as indicated in the above Matlab files. Here the speed is negative because the mass is falling down, and it increases from 0 to 320. It requires about 50 units of time to approach this speed.
The second numerical experiment is with the resistive force equal to
\[0.1 \text{abs}(u) + 0.001u^2.\]
So, the second line in the fmass.m must be changed to
\[\text{fmass} = -32 + 0.1\text{abs}(y) + 0.001y^2;\]
Here the speed is still negative because the mass is falling down, but now it only
increases from 0 to a little less than 140. Furthermore, it only takes about 20 units of
time is approach this speed.
2.6 Additional Calculations.

Consider the falling mass problem where \( m = 1, \ g = 32, \ u(0) = 0, \) and variable resistive forces \( .1 \text{abs}(u) + k \ u^2 \) where \( k \) changes.

(a). Consider the resistive force \( .1 \text{abs}(u) + (.001*(1+.S)*0.5)u^2 \). Modify the function file fmass.m and execute the code file imeuler. Note the curve for the speed on the vertical axis and time on the horizontal axis. Use the hold on command so that the following two curves will appear on the same graph.

(b). Consider the resistive force \( .1 \text{abs}(u) + (.001*(1+.S)*1.0)u^2 \). Modify the function file fmass.m and execute the code file imeuler.

(c). Consider the resistive force \( .1 \text{abs}(u) + (.001*(1+.S)*.5)u^2 \). Modify the function file fmass.m and execute the code file imeuler.

(d). Compare the three curves. What happens when \( k \) is increased.

(e). Find \( k \) so that the steady state speed is 12. There are two ways to do this. Either set the right side of the differential equation equal to zero and solve for \( k \), or do a number of numerical experiments with different \( k \).