PHYS 325 Assignment 2:
Some First Numerical Problems

Author: A. J. Ogle
Date: February 23rd, 2015
PHYS 325 Computational Physics

1 Introduction

The world is a changing entity. Part of the beauty and power of physics is its ability to predict how things will change using basic, fundamental laws. These mathematical descriptions of the world can be used to predict how objects fall, fly, and fling, or how particles decay or systems with rates of decay/diffusion stabilize. Traditionally, this predicting happened with pen and paper. In the modern day, however, a computer serves as a more efficient tool to perform tedious and repetitive calculations. One method to approximate differential equations (the kind used to describe how things become different, or change) is called Euler's method.

Euler's method for approximating solutions to first order differential equations is explored for a few real-world problems such as free fall, an object with constant velocity, and dampened and resonant systems of states (radioactive decay, namely). Euler's method is first presented and discussed, then applied to the individual problems in the following sections. Numerical results are compared to analytic counterparts. Euler's method was found to serve as a powerful approximation of first order differential equations, particularly with the aid of computer simulation.

2 Background

The Euler method of solving first order differential equations of the type:

$$\frac{dv}{dt} = a \tag{1}$$

Using Euler's method, we solve this differential equation by first finding its initial value, y(n), at some point x = n. From here, the next term is found by adding some small increment, dx, to the original term. In this way, we approximate the first order differential equation using the following form:

$$y(n+1) = y(n) + \frac{dy}{dx}dx \tag{2}$$

This technique suits computational methods well, since computers are able to calculate a large array of terms very quickly. In the following sections, we numerically and analytically solve first order differentials modeling some real-world conditions, then plot and analyze the results.

3 Problem 1.1: Free Fall

Free fall can be modeled by the following differential:

$$\sum F = ma = m\frac{dv}{dt} = -mg \tag{3}$$

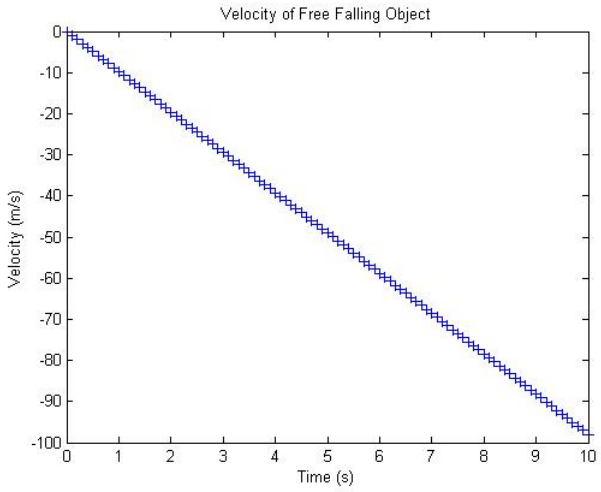

Figure 1: The changing velocity of an object in free fall. It is expected that the velocity of the object continue to increase in magnitude as the acceleration due to gravity acts on it. Since gravity is constant and any wind resistance is ignored, the rate at which the velocity of the object changes is constant.

This is because the only force acting on an object in free fall is assumed to be gravity. Using Euler's method, we find a constant change in velocity, as would be expected in the presence of a constant force. The numerical results agree with those arrived at analytically, as the figure shows a line with constant slope, as would be expected of a function with a constant derivative. Analytically, we expect the velocity of an object in free fall after ten seconds to be 98 meters per second. This can be seen from the equation for velocity given by 4. This is the numerical result produced by Euler's method from the simulation, as shown in figure 1.

$$v_f = v_i + at \tag{4}$$

This would not be the case if instead the position of the object versus time was plotted, as the acceleration constantly changes the velocity, resulting in a non-linear rate of change in displacement.

4 Problem 1.2: Constant Velocity

It should be noted here that the problem presented in this section is identical to that of problem 1.1, but with different context. In problem 1.1, we wanted to find the rate at which the velocity changed (the acceleration) which was constant. Here, we want to find the rate at which position changes (velocity), which is also constant. Using the methods outlined in 5, we solve the first order differential equation:

$$\frac{dx}{dt} = v \tag{5}$$

Plotting the output, we find:

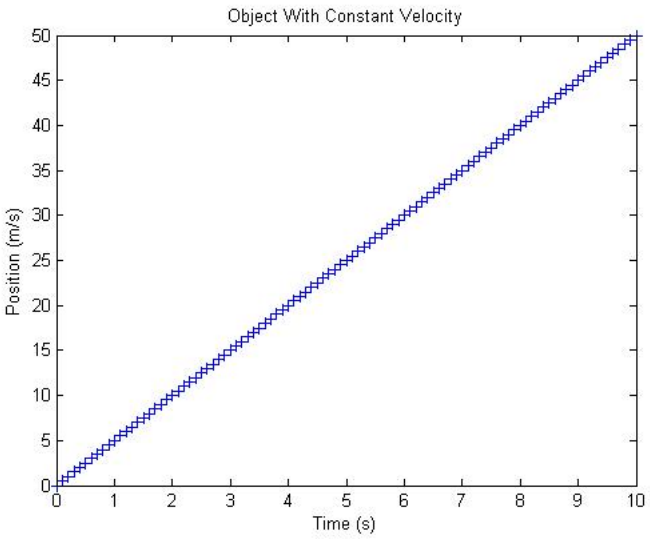

Figure 2: The changing position of an object with constant velocity.

Notice that the plots for both figure 1 and figure 2 look nearly identical, but with figure 1 having a negative slope and figure 2 having a positive slope. In either case, however, the slopes are constant (they are straight lines), meaning the derivative is constant.

5 Problem 1.4: Radioactive Decay

The decay of a particle into a smaller and more stable particle can be modeled by the equation:

$$\frac{dN_a}{dt} = -\frac{N_a}{\tau_a} \tag{6}$$

Where the decay time is represented by the variable Tau(a). Let's consider the case that the A particle (Na) decays into a B particle (Nb) before the B particle then decays into a different particle. In this case, the number of A and B particles can be modeled by the equation:

$$\frac{dN_a}{dt} = -\frac{N_a}{\tau_a} \tag{7}$$

and

$$\frac{dN_b}{dt} = \frac{N_a}{\tau_a} - \frac{N_b}{\tau_b} \tag{8}$$

As A particles decay into B particles, the number of B particles increases until the A particles decay to a point such that the newly created B particles are decaying faster than they are being generated. This explains the initial rise in the number of B and then complete decay of the B and A particle line, as shown in the figure below.

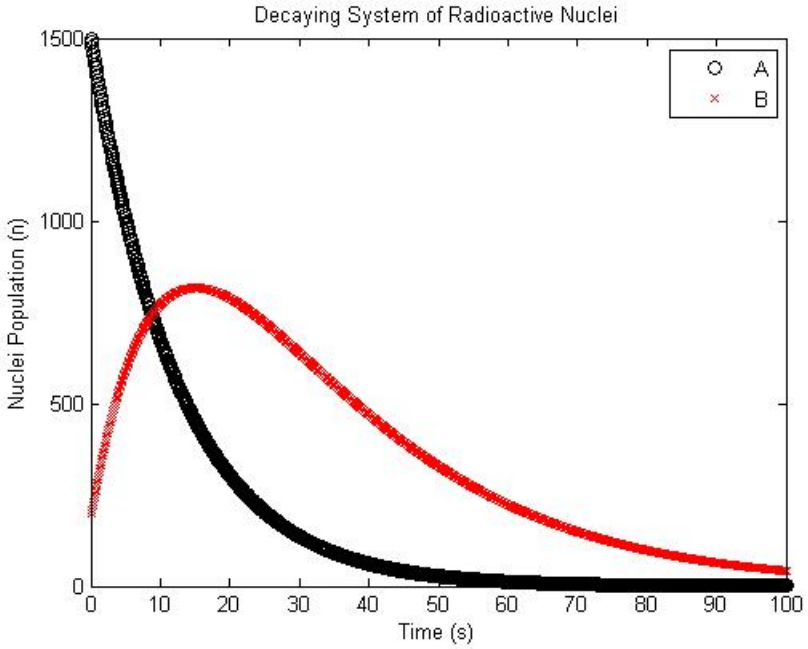

Figure 3: A particles decay into B particles, whereupon the B particles then decay into smaller particles. This simulation has the following initial conditions: Tau(a) = 12.56, Tau(b) = 22.72, N(a) = 15e5, N(b) = 2e2. The maximum of the B particle line represents the point at which the decay rate of A is equal to the decay rate of B. As fewer A particles decay, fewer B particles are generated, until the rate at which B particles are generated becomes less than the rate at which they decay. This simulation is representative of a dampened system.

6 Problem 1.5: Radioactive Decay with Resonance

Similar to problem 1.4, this system represents the transition of a particle between two states, namely A and B. It can be modeled by equations similar to 6, except a new term modeling the transition of B particles back into A particles must be added, and vice versa for the equation determining the number of B particles.

$$\frac{dN_a}{dt} = \frac{N_b}{\tau_b} - \frac{N_a}{\tau_a} \tag{9}$$

$$\frac{dN_b}{dt} = \frac{N_a}{\tau_a} - \frac{N_b}{\tau_b} \tag{10}$$

The decay times of the A (Tau of a) and B (Tau of b) particles can be adjusted to change the levels at which the A and B states settle.

A particles decay into B particles, then B particles decay into A particles, until the rates at which they decay into each state become equal. At this point, the system has reached what is called a "steady state," in which the rate of change for both populations of particles goes to 0. Looking at figure 4, this time happens at about 30 seconds. It is at this time that the A and B lines become nearly horizontal.

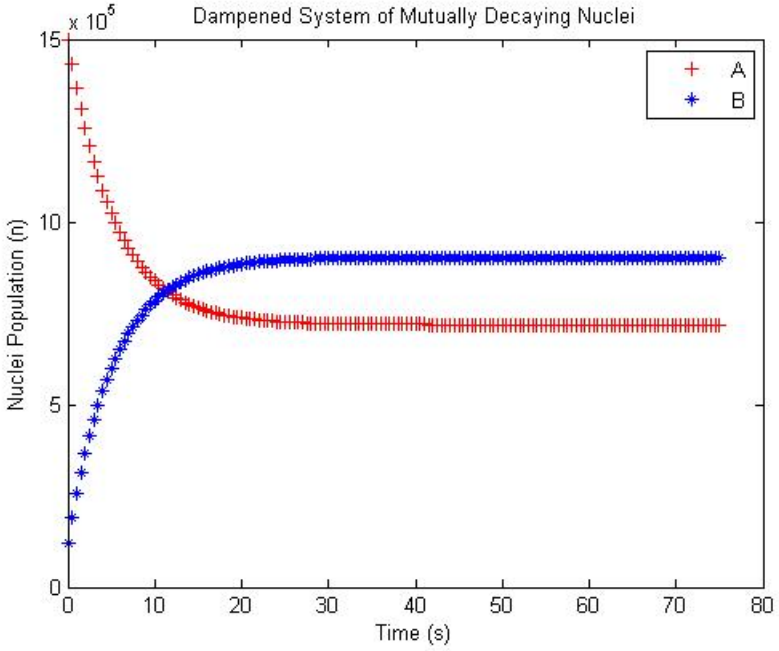

Figure 4: A resonant system of decaying nuclear particles is represented in this figure. This simulation used the following initial conditions: Tau(a) = 10, Tau(b) = 12.56, N(a) = 15e5, N(b) = 12.2e4. Alpha particles decay into beta particles, whereupon the beta particles decay into alpha particles. This transformation from alpha to beta continues until the decay rates for both alpha and beta particles are equivalent, resulting in a stable state where the rate of change in the population of alpha and beta particles approaches 0.

7 Conclusions

As can be seen from the solutions of the previous problems, the Euler method of approximating solutions to first order differential equations is more than sufficient for use in modeling real-world phenomena such as free fall, motion at constant velocity, and dampened and resonant radioactivity.

References

[1] Giordano, Nicholas J., and Hisao Nakanishi. "1. A First Numerical Problem." Computational Physics. Upper Saddle River, NJ: Pearson/Prentice Hall, 2006. N. pag. Print.

[2] Barker, Christopher A. "Euler's Method." Numerical Methods–Euler's Method. San Joaquin Delta College, 2009. Web. 28 Feb. 2015. ¡http://calculuslab.deltacollege.edu/ODE/7-C-1/7-C-1-h-b.html¿.

8 MATLAB Code

```
%% Begin Simulation

%clear everything before beginning the run
clear all

%Politely ask user which simulation they would like to run
prompt = {'Run 1.1 Simulation? (1=yes, 0=no)',
'Run 1.2 Simulation? (1=yes, 0=no)',
'Run 1.4 Simulation? (1=yes, 0=no)',
'Run 1.5 Simulation? (1=yes, 0=no)'};
dlg_title = 'Input';
num_lines = 1;
def = {'0','0','0','1'};
begin = inputdlg(prompt,dlg_title,num_lines,def);

for i = 1:length(begin)
    if(str2double(begin(i)) == 1)
       run(i) = 1;
    else
       run(i) = 0;
    end
end

run1_1 = run(1);
run1_2 = run(2);
run1_4 = run(3);
run1_5 = run(4);

%% Problem 1.1
if(run1_1)

%Request information from user
prompt = {'Step size? (dx):','dx/dt:','Offset?'};
dlg_title = 'Input';
num_lines = 1;
def = {'0.1','5','0'};
answer = inputdlg(prompt,dlg_title,num_lines,def);

% define the step size
dx = str2double(answer(1));

%initialize array of x values
x = (0.0:dx:10);

%define dydx
dydx = str2double(answer(2));
```

```
%define the offset
c = str2double(answer(3));

%begin Euler's method for calculation, store in new array
for i = 1:length(x)
    %define initial conditions, initialize y array
    if(i == 1)
       a(i) = 0.0 + c;
    else
    a(i) = a(i-1)-dydx*dx + c;
    end
end

clc
disp('Simulation 1.1 completed.')

plot(x,a,'+');
hold on
xlabel('Time (s)')
ylabel('Velocity (m/s)')
title('Velocity of Free Falling Object')

end

%% Problem 1.2
if(run1_2)


%Request information from user
prompt = {'Step size? (dx):','dx/dt:','Offset?'};
dlg_title = 'Input';
num_lines = 1;
def = {'0.1','5','0'};
answer = inputdlg(prompt,dlg_title,num_lines,def);

% define the step size
dx = str2double(answer(1));

%initialize array of x values
x = (0.0:dx:10);

%define dydx
dydx = str2double(answer(2));

%define the offset
c = str2double(answer(3));

%begin Euler's method for calculation, store in new array
for i = 1:length(x)
    %define initial conditions, initialize y array
```

```
    if(i == 1)
       a(i) = 0.0 + c;
    else
    a(i) = a(i-1)+dydx*dx + c;
    end
end

clc
disp('Simulation 1.2 completed.')

clf
plot(x,a,'+');
hold on
xlabel('Time (s)')
ylabel('Position (m/s)')
title('Object With Constant Velocity')

end

%% Problem 1.4

if(run1_4)

%Request information from user
prompt = {'Step size? (dt):'
,'dNa/dt:','dNb/dt',
'Initial # of A  Isotopes',
'Decay Rate of A?',
'Initial # of B  Isotopes',
'Decay Rate of B?',
'Total Time?'};
dlg_title = 'Input';
num_lines = 1;
def = {'0.1','5','5','15e2','12.56','2e2','22.72','100'};
answer = inputdlg(prompt,dlg_title,num_lines,def);

%Define initial variables
dt = str2double(answer(1)); %step size
dnadt = str2double(answer(2)); %rate of change for A isotopes
dnbdt = str2double(answer(3)); %rate of change for B isotopes
Na = str2double(answer(4)); %number of initial A isotopes
ta = str2double(answer(5)); %decay rate of A
Nb = str2double(answer(6)); %number of initial B isotopes
tb = str2double(answer(7)); %decay rate of B
ft = str2double(answer(8)); %total simulated time

%Create an array for the time variable (independent variable)
t = (0.0:dt:ft);

%Loop for calculating the decay of alpha particle
```

```
for i = 1:length(t)
    %define initial conditions, initialize y array
    if(i == 1)
       a(i) = Na;
       b(i) = Nb;
       dnadt = -Na/ta;
       dnbdt = -Nb/tb;
    else
        %calculate the number of isotopes after next dt
        a(i) = a(i-1)+dnadt*dt;
        dnadt = -a(i)/ta;

        %calculate the number of isotopes after next dt
        b(i) = b(i-1)+dnbdt*dt;
        dnbdt = a(i)/ta - b(i)/tb;
    end
end

clc
disp('Simulation 1.4 completed.');

clf
plot(t,a,'ok');
hold on;
plot(t,b,'xr');
hold on;
xlabel('Time (s)')
ylabel('Nuclei Population (n)')
title('Decaying System of Radioactive Nuclei')

end

%% Simulation 1.5

if(run1_5)

%This simulation is very similar to 1.4, except the system is more
%representative of a resonance between two states, A and B.

%Request information from user
prompt = {'Step size? (dt):',
'dNa/dt:','dNb/dt',
'Initial # of A  Isotopes',
'Decay Rate of A?',
'Initial # of B  Isotopes',
'Decay Rate of B?','Total Time?'};
dlg_title = 'Input';
num_lines = 1;
def = {'0.5','0','0','15e5','10','12.2e4','12.56','50'};
answer = inputdlg(prompt,dlg_title,num_lines,def);
```

```
%Define initial variables
dt = str2double(answer(1)); %step size
dnadt = str2double(answer(2)); %rate of change for A isotopes
dnbdt = str2double(answer(3)); %rate of change for B isotopes
Na = str2double(answer(4)); %number of initial A isotopes
ta = str2double(answer(5)); %decay rate of A
Nb = str2double(answer(6)); %number of initial B isotopes
tb = str2double(answer(7)); %decay rate of B
ft = str2double(answer(8)); %total simulated time

%Create an array for the time variable (independent variable)
t = (0.0:dt:ft);

%Loop for calculating the decay of alpha particle
for i = 1:length(t-1)
    %define initial conditions, initialize a and b array
    if(i == 1)
       a(1) = Na;
       b(1) = Nb;
       dnadt = Na/ta;
       dnbdt = Nb/tb;
       tN(1) = Na + Nb;
    else
        %calculate the rates of change after next dt
        dnadt = a(i-1)/ta;
        dnbdt = b(i-1)/tb;

        %calculate the number of isotopes after next dt
        a(i) = a(i-1) - dnadt*dt + dnbdt*dt;
        b(i) = b(i-1) - dnbdt*dt + dnadt*dt;

        %create array that counts total particles over time
        %This can be plotted to ensure the "conservation of particles"
        %in this closed system.
        tN(i) = a(i) + b(i);
    end
end

clc
disp('Simulation 1.5 completed.');

clf
plot(t,a,'+r');
hold on;
plot(t,b,'*b');
hold on;
legend('Alpha','Beta')
xlabel('Time (s)')
ylabel('Nuclei Population (n)')
```

```
title('Dampened System of Mutually Decaying Nuclei')
end

%% No Simulation Selected Error Message

%Tells the user that no simulation was selected. Will make message more
%sassy at a later date.
if(~run1_1 && ~run1_2 && ~run1_4 && ~run1_5)
    disp('Error occured with running simulation;
     No simulation was selected to be run');
end
```