Project 3 - Numerical Differential Equation Solvers

Chris Newey

February 14, 2020

Abstract In this project we explore predator-prey and epidemic models described by systems of differential equations. We solve these systems of differential equations numerically using an explicit fourth order Runge-Kutta method. After numerically solving the models we plot the data and analyze the physical meaning of the results.

Motivation Both simulations use an explicit fourth order Runge-Kutta method (RK4) capable of solving a system of first order differential equations. The function, `rk4_sys`, accepts a function, the number of equations of the system, initial conditions, endpoints, and the number of steps to be taken. Next, the function calculates $h = (b - a)/n\_steps$, h/2. Time steps are initialized with t= [a:h:b]. Then the function initializes the size of $f = zeros(n\_steps + 1, nvars)$ and puts the initial conditions in the first row with $f(1, :) = init$. The matrix f consists of column vectors for the solutions at each time step plus the initial condition for each equation in the system. Now, we iterate:

```
k1 = halfh*func(nvars, t(j), f(j,:));
k2 = halfh*func(nvars, t(j)+h/2.0,f(j,:)  + k1);
k3 = halfh*func(nvars, t(j)+h/2.0,f(j,:)  + k2);
k4 = halfh*func(nvars, t(j)+h, f(j,:)  + 2.0*k3);

f(j+1,:)  = f(j,:)  + (k1 + 2.0*k2 + 2.0*k3 + k4)/3.0;
```

The call to `func` evaluates the system of differential equations stored in a separate function. The (RK4) code passes parameters t and n_vars to `func`, which are not necessary in this project, to keep the code robust. Finally, we combine the time vector with the solution matrix f and output the matrix y. Variable y consists of one column vector for the time steps and column vectors of the solutions for each system.

Problem 1: For problem one we examine a simple predator-prey model using the system of differential equations:

$$\begin{aligned} \frac{dy}{dt} &= y(\ a - bz) \\ \frac{dz}{dt} &= z(-c + dy) \end{aligned} \tag{1}$$

Variable y represents the prey population and z denotes the predator population. The parameters a,c are the natural birth and death rates of the prey and predator in isolation. Parameters c and d, describe the effects that interactions between the predator and prey species generate. Note, a,b,c, and d > 0. With no interactions between the species, the prey population grows unboundedly, and the predator population decreases to zero. Interactions between the species cause the prey population to decrease while the predator population increases until the prey population is too small to sustain predator population growth. At this point, the predator population decreases, which allows the prey population to grow. The growth in the prey population, eventually leads to a growth of predator population and the cycle continues. The parameters and system **??** are in the function, `pred_prey`.

For the first part of problem one we implement the fourth-order `rk4_sys`, and test it on system **??** with a=1, b=0.1, c=0.5, d=0.02, and initial populations y(0)=100, z(0)=10, $t_m = 25$, and 500 time-steps. The parameters a,b,c,d are initialized in `pred_prey`. The driver program, `driver_problem1` initializes the number of variables, initial conditions, start and stop times, the number of steps, and calls the RK4 method using:

```
b = rk4_sys(@pred_prey,nvars,init,t_0,t_m,n_steps);
```

The results are then plotted and can be seen in figure **??**.

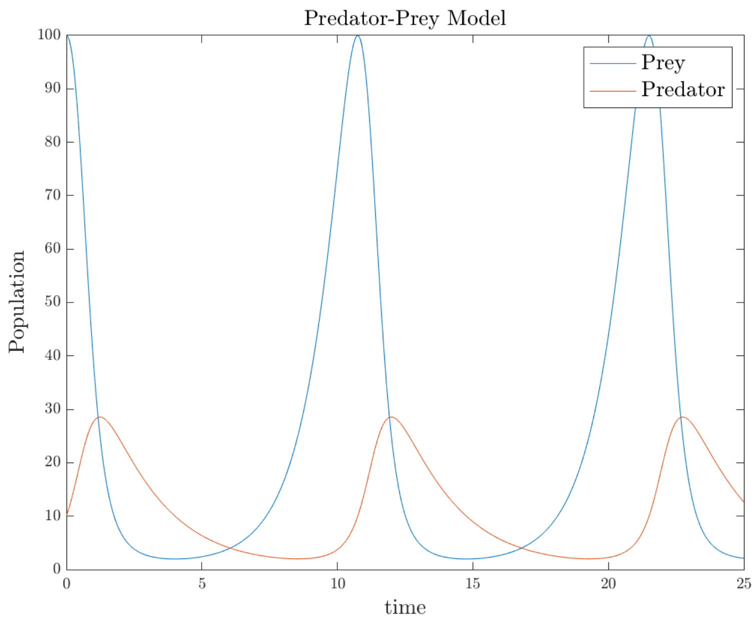

Figure 1: Plot of predator-prey model for y(0) = 100 and z(0)=10

Next, we estimate the period of the cyclic decay/growth pattern. From figure **??** we seen that the cycle begins to repeat itself around t=10.75. This estimate is used to plot y(t) versus z(t) to see an estimated cycle. However, the plot does not fully close until the next time step, t=10.80 where it has now overlapped slightly. So, the true value of one cycle must be in between, call it 10.755. Now, since the plot closes in on itself and the system nearly returns to the initial conditions we see that 10.755 time units is approximately one period. The plot of y(t) versus z(t) for t=0-10.755 figure **??**.

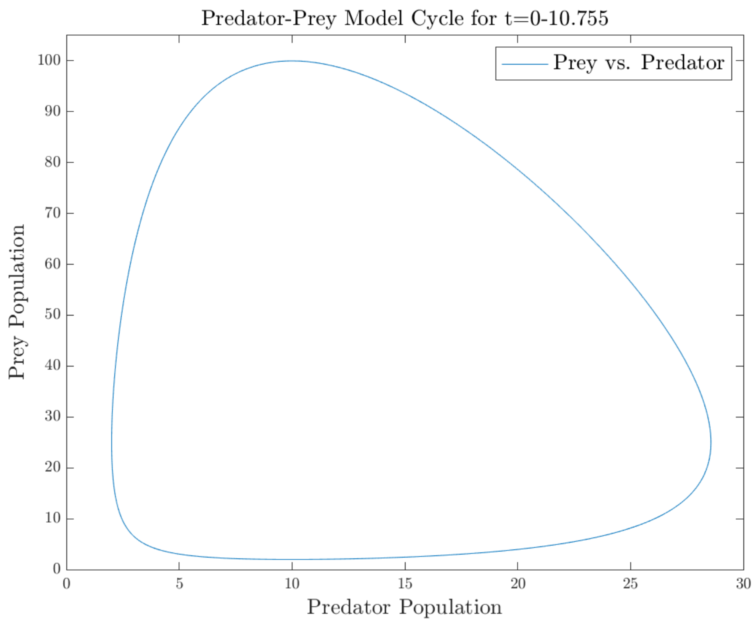

Figure 2: One Cycle for predator-prey model for $t = 0 - 10.755$

The final part of problem one involves determining non-zero initial populations such that the populations never change. Constant solutions implies both derivatives simultaneously equal 0. This gives:

$$\begin{aligned} \frac{dy}{dt} &= y(a - bz) = 0 \\ \frac{dz}{dt} &= z(-c + dy) = 0 \end{aligned} \tag{2}$$

Taking positive values for physically realistic results we get, $y = c/d$ and $z = a/b$. In our case we use $y(0) = c/d = .5/.02 = 25$ and $z(0) = a/b = 1/.1 = 10$. A numerical simulation with these initial conditions is run and plotted in figure **??**.

<u>Problem 2</u> Consider the epidemic model:

$$\begin{aligned} y_1' &= -\,cy_1y_2 \\ y_2' &= \quad cy_1y_2 - dy_2 \\ y_3' &= \quad dy_2 \end{aligned} \tag{3}$$

where y_1, y_2, y_3 represent respectively the percentages of susceptible, infective in circulation, and infective removed by isolation, death, or recovery and immunity. Parameters c,d represent the infection rate and removal rate, respectively. The parameters and system **??** are in the function, `epidemic`.

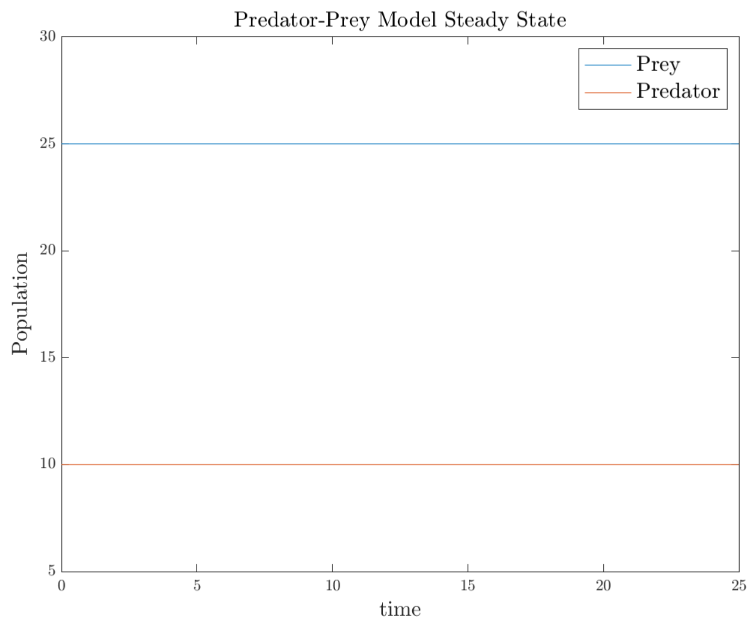

Figure 3: Steady state of predator-prey model for $y(0) = c/d = 25$ and $z(0) = a/b = 10$

The project simulation takes $c = 1$, $d = 5$, $y_1(0) = 95$, $y_2(0) = 5$, $y_3(0) = 0$, and solves system **??** for t=0-1 with 100 time steps using the RK4 method. The results are then plotted in figure **??**.

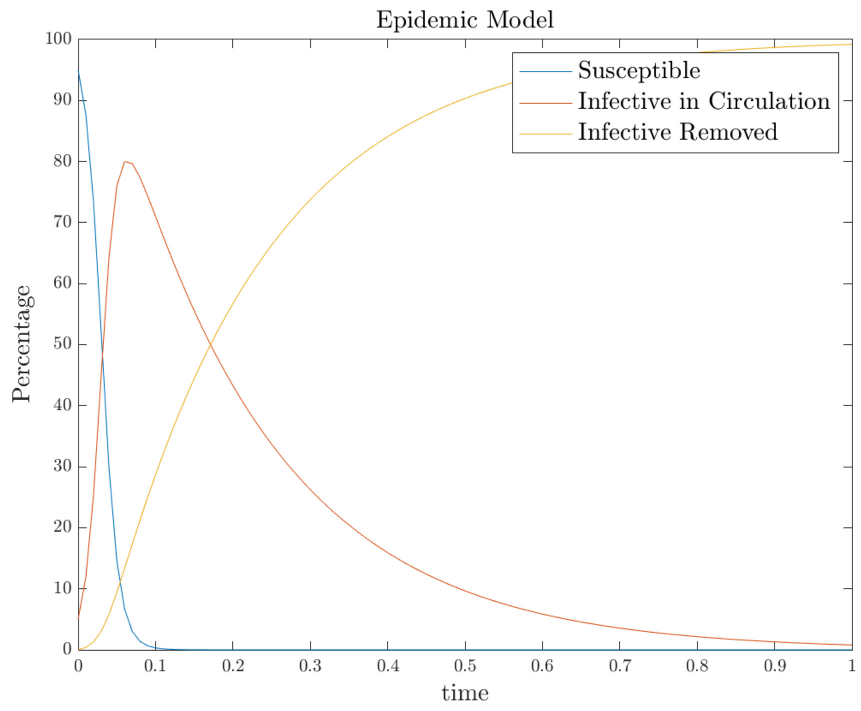

Figure 4: Plot of epidemic model with $y_1(0) = 95$, $y_2(0) = 5$, $y_3(0) = 0$

Now, to interpret the results. First, we see the total population accounted for in our system is 100%. The susceptible population begins at 95%, the infective in circulation begins at 5%, and the infective removed population begins at 0%. Now, we start to see the process begin. First, the susceptible population quickly decreases as the infective in circulation quickly decreases and the infective removed population increases slower than the other two populations. So, we see the susceptible population become infected and the susceptible population decreases. At the same time, infective removed population increases as people are removed by isolation, death, or recovery and immunity. Next, we see a turning point in the infective in circulation population. At this point, the infective in circulation population begins to slow and until starts to decrease. At this same time the susceptible population drops to nearly 0%. The infective removed population continues to increase. As the simulation continues to the end the infective population reaches effectively 0% and the infective removed approaches 100%. So, in the end the entire population is considered infective removed.

Program Listings

RK4 Solver:

Function:

```
function y = rk4_sys(func,nvars,init,a,b,n_steps)
%Input - func is the call to the function with the eqns for the
    systems to
%           be solved
%        - a and b are the endpoints of the interval
%        - init=[x_1(a),...x_nvars(a)] are the initial conditions
%        - n_steps is the number of steps
%Output - h is the step size
%         - t is the vector of steps
%         - f = [x1(t)...xn(t)]; where x_k(t) is the approximation
%                 to the kth dependent variable
%         - y = a matrix of time and solution vectors
h= (b-a)/n_steps;
halfh = h/2.0;
t= [a:h:b];
f= zeros(n_steps+1,nvars);
f(1,:) = init;
for j=1:n_steps

    k1 = halfh*func(nvars, t(j),       f(j,:));
    k2 = halfh*func(nvars, t(j)+h/2.0,f(j,:) + k1);
    k3 = halfh*func(nvars, t(j)+h/2.0,f(j,:) + k2);
    k4 = halfh*func(nvars, t(j)+h,     f(j,:) + 2.0*k3);

    f(j+1,:) = f(j,:) + (k1 + 2.0*k2 + 2.0*k3 + k4)/3.0;
```

```
end
t = transpose(t);
y = [t, f];
```

Problem 1:

Driver:

```
function driver_problem1
%This function calls an rk4 algo to solve a system of eqns
    describing a
%predator-prey system, z(t) and y(t) respectively.
%Plots: y(t), z(t) vs time then y(t) vs z(t).
%Input - nvars: number of variables in the system of diff eqs
%       - init = [x_1(a),...x_nvars(a)] are the initial conditions
%       - t_0 and t_m are the stop and start times
%       - n_steps is the number of steps
%Output - b: first column is t_k and the next culumns are y(t_k),z
    (t_k)
%        - f = plots of results for the solution of the predator-
    prey system
nvars = 2;
y_0 = 100;
z_0 = 10;
init = zeros(nvars,1);
init(1) = y_0;
init(2) = z_0;
t_0 = 0;
t_m = 25.0;
n_steps = 500;

b = rk4_sys(@pred_prey,nvars,init,t_0,t_m,n_steps);

figure();
str0 = ['Prey'];
str1 = ['Predator'];
plot(b(:,1),b(:,2),b(:,1),b(:,3))
title('Predator-Prey Model')
xlabel('time')
ylabel('Population')
legend({str0,str1})

b = rk4_sys(@pred_prey,nvars,init,t_0,10.755,n_steps);
```

```
figure();
str0 = ['Prey vs. Predator'];
plot(b(:,3),b(:,2))
ylim([0 105])
title('Predator-Prey Model Cycle for t=0-10.755')
xlabel('Predator Population')
ylabel('Prey Population')
legend({str0})

end
```

<u>Driver:</u>

```
function driver_problem1p3

a=1.0;
b=.1;
c=.5;
d=.02;
nvars = 2;
y_0 = c/d;
z_0 = a/b;
init = zeros(nvars,1);
init(1) = y_0;
init(2) = z_0;
t_0 = 0;
t_m = 25.0;
n_steps = 500;

b = rk4_sys(@pred_prey,nvars,init,t_0,t_m,n_steps);

figure();
str0 = ['Prey'];
str1 = ['Predator'];
plot(b(:,1),b(:,2),b(:,1),b(:,3))
ylim([5 30])
title('Predator-Prey Model Steady State')
xlabel('time')
ylabel('Population')
legend({str0,str1})

end
```

<u>Function:</u>

```
function yp = pred_prey(nvars,t,y)
%Input - nvars: number of variables in the system of diff eqns
```

```
%       - t is time step
%       - y are the current values of the soln
%Output - yp: the values for diff eqns at the current time step
    a = 1.0;
    b = 0.1;
    c = 0.5;
    d = 0.02;
    yp = zeros(1,nvars);
    yp(1) = y(1)*( a-b*y(2));
    yp(2) = y(2)*(-c+d*y(1));

end
```

<u>**Problem 2:**</u>

<u>Driver:</u>

```
function driver_problem2

nvars = 3;
y1_0 = 95.0;
y2_0 = 5.0;
y3_0 = 0.0;
t_0  = 0.0;
t_m  = 1.0;
n_steps = 100;

init = zeros(nvars,1);
init(1) = y1_0;
init(2) = y2_0;
init(3) = y3_0;

b = rk4_sys(@epidemic,nvars,init ,t_0,t_m,n_steps);

figure();
str0 = ['Susceptible'];
str1 = ['Infective in Circulation'];
str2 = ['Infective Removed'];
plot(b(:,1),b(:,2),b(:,1),b(:,3),b(:,1),b(:,4))
title('Epidemic Model')
xlabel('time')
ylabel('Percentage')
legend({str0,str1,str2})

end
```

Function:

```
function yp = epidemic(nvars,t,y)
%Input - nvars: number of variables in the system of diff eqns
%       - t is time step
%       - y are the current values of the soln
%Output - yp: the values for diff eqns at the current time step
    c = 1.0;
    d = 5.0;
    yp     = zeros(1,nvars);
    yp(1) = -c*y(1)*y(2);
    yp(2) =  c*y(1)*y(2)-d*y(2);
    yp(3) =  d*y(2);
end
```