1. **In order to change which disease you want to run you remove the % sign for it’s corresponding parameter and make sure the others still have % signs**
2. **The title can be changed each time you run a simulation to reflect what we’re actually looking at. Plot elements are at the very end of the code**
3. **Sigma is now going to be the value that we change, so we should be running it at 0,1, and the value taken from the COVID data**

function simulated\_stages=general\_covid19\_sim(initial\_values, parameters, max\_time\_units, dt)

%SARS:

%parameters = [0.25, 0, 0.03521126761, 0.04255319149, 0.04, 0.05361930295, 0.1569858713, 0.206185567, 0.0279, 0.9721, 0.13, 0.87, 1, 0];

%COVID:

parameters = [0.1652892562, 0.75, 0.71548, 0.1072425, 0.1111111111, 0.07142857143, 0.25, 0.2173913043, 0.0089, 0.9911, 0.3, 0.7, 0.023, 0.977];

%Measles:

%parameters = [0.0005, 0, 0, 0.125, 0.0625, 0.25, 0.07143, 0.08333333333, 0.003, 0.997, 0, 1, 0.211, 0.789];

initial\_values = [1000, 0, 1, 0, 0, 0, 0, 0];

dt = 0.5;

max\_time\_units = 100/dt;

%parameters = [beta, f\_a, gamma\_a, gamma\_i, gamma\_h, gamma\_q, eta, sigma, mu, mu\_diff, p\_a, p\_a\_diff, c\_h, c\_h\_diff];

%initial\_values = [S0, E0, I0, A0, Q0, H0, D0, R0];

% Runs a simulation of a general infectious disease model.

% Program assumes the following:

%%%% Runge-Kutta 4 implemented using time increments of size dt.

%%%% Simulation runs over at most m time increments.

%%%% There are n stages in the model.

%

% simulated\_stages is an (m+1)\*n matrix of the simulated values, where the

% element in row i and column j corresponds to the number of individuals in

% stage j at time step i.

% For example, if modeling an SIR model, then simulated\_stages(4, 2) would

% correspond to the population of infected individuals (stage 2) at time 4.

%

% initial\_values is a 1\*n matrix of the initial values for each stage.

% NOTE: initial\_values must be entered as a Matlab-formatted matrix such as

% [a1, a2, a3, ..., an] but commas may be omitted if feeling adventurous.

%

% parameters is a matrix of all of the parameters relevant for the model.

% NOTE: parameters must be entered as a Matlab-formatted matrix.

%

% max\_time\_units is a scalar indicating the number of time units to be

% simulated over.

%

% dt is a scalar indicating the size of the increments.

%

% WARNING: Double check that yprime function is written correctly. There

% is no checker for its formatting since yprime depends so heavily on the

% model.

%

% by Stephen Becklin

% written 2022/04/11

% Clear previous results

clc;

% Check for correct input formats

if ~ismatrix(initial\_values)

disp('Initial values input is not in the form of a Matlab matrix.');

return;

end

if size(initial\_values, 1) ~= 1

disp('Initial values input is not a 1-by-n matrix.');

return;

end

if ~ismatrix(parameters)

disp('Parameters input is not in the form of a Matlab matrix.');

return;

end

if size(parameters, 1) ~= 1

disp('Parameters input is not a 1-by-# matrix.');

return;

end

if ~isnumeric(max\_time\_units) | mod(max\_time\_units,1) ~= 0

disp('Max time units input is not an integer.');

return;

end

if size(max\_time\_units, 1) ~= 1 || size(max\_time\_units, 2) ~= 1

disp('Max time units input must be a single numeric input and not a matrix.');

return;

end

if ~isnumeric(dt)

disp('dt input is not a number.');

return;

end

if size(dt, 1) ~= 1 || size(dt, 2) ~= 1

disp('dt input must be a single numeric input and not a matrix.');

return;

end

% Create output matrix

simulated\_stages = zeros(max\_time\_units + 1, size(initial\_values, 2));

simulated\_stages(1, :) = initial\_values;

% Computation

Y = initial\_values;

y = transpose(Y);

for t = 1:max\_time\_units

y = rk4(dt,y,parameters);

Y = transpose(y);

simulated\_stages(t+1, :) = Y;

%disp(simulated\_stages); %% comment out?

end

% Function for Runge-Kutta 4

%%% DO NOT EDIT

function y=rk4(dt,y0,parameters)

% dt is the time increment

% y0 is a column vector of stages at time t

% y is a column vector of projected stages at t + dt

k1 = yprime(y0,parameters);

k2 = yprime(y0+0.5\*dt\*k1,parameters);

k3 = yprime(y0+0.5\*dt\*k2,parameters);

k4 = yprime(y0+dt\*k3,parameters);

y = y0+dt\*(k1+2\*k2+2\*k3+k4)/6;

end

% Function for derivatives in differential equation

function yp = yprime(y,parameters)

% y is a column vector of stages at time t

% yp is a column vector of the derivatives for each stage at time t

% Sample code for SIR model:

%% dS/dt = -beta \* S \* I

%% dI/dt = beta \* S \* I - alpha \* I

%% dR/dt = alpha \* I

%% Resulting code assuming parameters = [alpha, beta]:

%%% yp(1) = -1\*parameters(2)\*y(1)\*y(2);

%%% yp(2) = parameters(2)\*y(1)\*y(2) - parameters(1)\*y(2);

%%% yp(3) = parameters(1)\*y(2);

% Create derivative vector

yp = sqrt(-1)\*ones([size(y, 1) 1]);

% Write differential equation code below

%%% INSERT CODE HERE

%y(1) is S, y(2) is E , y(3) is I, y(4) is A, y(5) is Q, y(6) is H, y(7)

% is D and y(8) is R

yp(1) = -1\*parameters(1)\*y(1)\*(y(3)+parameters(2)\*y(4));

yp(2) = parameters(1)\*y(1)\*parameters(2) - parameters(7)\*y(2);

yp(3) = parameters(12)\*parameters(7)\*y(2)-parameters(4)\*y(3)-parameters(8)\*y(3);

yp(4) = parameters(11)\*parameters(7)\*y(2)-parameters(3)\*y(4);

yp(5) = parameters(8)\*y(3)-parameters(6)\*y(5);

yp(6) = parameters(6)\*y(5)\*parameters(13)+parameters(4)\*y(3)\*parameters(14)-parameters(5)\*y(6);

yp(7) = parameters(9)\*parameters(5)\*y(6);

yp(8) = parameters(4)\*y(3)\*parameters(14)+parameters(3)\*y(4)+parameters(10)\*parameters(5)\*y(6)+parameters(6)\*y(5)\*parameters(14);

S = simulated\_stages(:,1);

E = simulated\_stages(:,2);

I = simulated\_stages(:,3);

A = simulated\_stages(:,4);

Q = simulated\_stages(:,5);

H = simulated\_stages(:,6);

D = simulated\_stages(:,7);

R = simulated\_stages(:,8);

% Days = [];

% for t = 1:101

% Days = [t; Days];

% end

% Days = Days;

time = 0:dt:max\_time\_units\*dt;

newcolors = [ 0 0 0

0 0.4470 0.7410

0.8500 0.3250 0.0980

0.9290 0.6940 0.1250

0.4940 0.1840 0.5560

0.4660 0.6740 0.1880

0.3010 0.7450 0.9330

0.6350 0.0780 0.1840];

colororder(newcolors);

plot(S, '-', 'LineWidth',2);

hold on

plot(E, '-', 'LineWidth',2);

plot(I, '-', 'LineWidth',2);

plot(A, '-', 'LineWidth',2);

plot(Q, '-', 'LineWidth',2);

plot(H, '-', 'LineWidth',2);

plot(D, '-', 'LineWidth',2);

plot(R, '-', 'LineWidth',2);

hold off

legend('S', 'E', 'I', 'A', 'Q', 'H', 'D', 'R')

title('COVID Simulation with X= 0.1')

xlabel('Time')

ylabel('Population')

end

end