Part 1

For the following temporal scheme,

$$y_{\mathrm{n+1}} = y_n + \frac{h}{2}[3f(y_n) - f(y_{\mathrm{n-1}})]$$
$$y_{\mathrm{n+1}} = y_n + \frac{h}{2}[f(y_n) - f(y_{\mathrm{n+1}})]$$

Derive the local truncation error and region of absolute stability:

Local truncation error provides a computational estimate of error introduced for one iteration of a numerical estimate. The general approach taken to derive a local truncation error LTE for a numerical method, where $LTE = y(t_{n+1}) - y_{n+1}$ is the following:

1. Taylor expand the $y(t_{n+1})$ term. 2. Taylor expand the other term on the right side of the equation. 3. Subtract the two expanded terms in order to get the LTE value.

For the first scheme (Adams-Bashforth explicit method)

$$LTE = \frac{1}{h}(y_{n+1} - y_n) - \frac{1}{2}[3f(y_n) - f(y_{n-1})]$$
$$= \frac{1}{h}(y_{n+1} - y_n) - \frac{1}{2}[3y'_n - y'_{n-1}]$$

Expanding the term on the left:

$$\frac{1}{h}(y_{n+1} - y_n) = y'_n + \frac{y''_n}{2}h + \frac{y'''_n}{6}h^2$$

Expanding the term on the right:

$$\frac{3}{2}y'_n - \frac{1}{2}y'_{n-1} = \frac{3}{2}y'_n - \frac{1}{2}(y'_n - hy''_n + h^2\frac{y'''_n}{2})$$
$$= y'_n + \frac{h}{2}y''_n - \frac{h^2}{4}y'''_n$$

Substituting expanded terms into previous expression:

$$LTE = y'_n + \frac{y''_n}{2}h + \frac{y'''_n}{6}h^2 - [y'_n + \frac{h}{2}y''_n - \frac{h^2}{4}y'''_n]$$
$$LTE = \frac{5}{12}y'''_n h^2 \tag{1}$$

Absolute Stability analysis for this scheme:

$$y_{n+1} - y_n = \frac{h}{2}[3f(y_n) - f(y_{n-1})]$$

Using the relation $y'(t_n) = f(y(t_n)) = \lambda y(t_n)$ and

$$y(t_n) = y_n$$

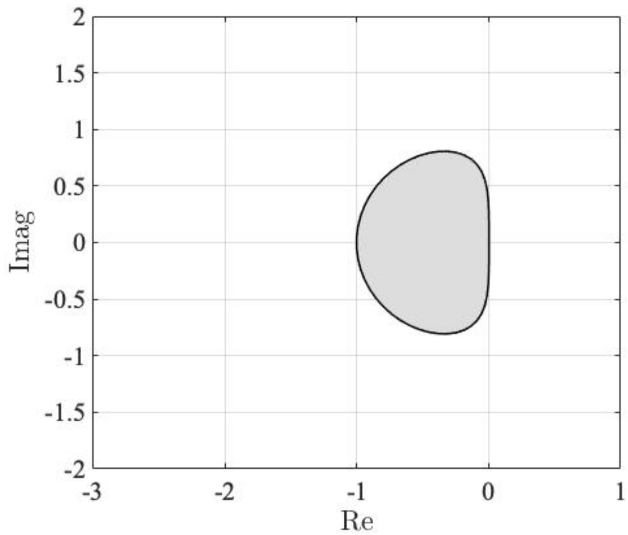
,

$$y_{n+1} - y_n = \frac{\lambda h}{2}[3y_n - y_{n-1}]$$

$$z^2 - z(1 + \frac{3}{2}\lambda h) + \frac{1}{2}\lambda h = 0$$

$$\lambda h = \frac{z^2 - z}{\frac{3}{2}z - 1} \tag{2}$$

Plotting the region of absolute stability for this method:



For the second scheme (Adams-Moulton implicit method):

$$\begin{aligned} LTE &= \frac{1}{h}(y_{n+1} - y_n) - \frac{1}{2}[f(y_n) + f(y_{n+1})] \\ &= \frac{1}{h}(y_{n+1} - y_n) - \frac{1}{2}[y'_n + y'_{n+1}] \end{aligned}$$

Expanding the term on the left:

$$\frac{1}{h}(y_{n+1} - y_n) = y'_n + \frac{y''_n}{2}h + \frac{y'''_n}{6}h^2$$

Expanding the term on the right:

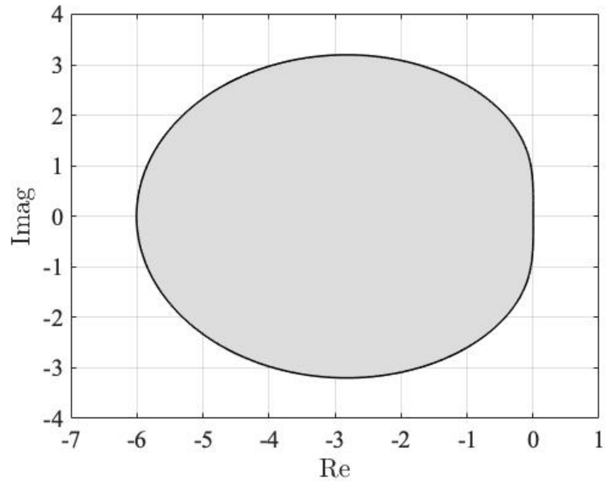
$$\begin{aligned} \frac{1}{2}[y'_n + y'_{n+1}] &= \frac{1}{2}(y'_n + y''_n h + \frac{y'''_n}{2}h^2) + \frac{1}{2}y'_n \\ &= y'_n + fracy''_n 2h + \frac{y'''_n}{4}h^2 \end{aligned}$$

Substituting expanded terms into previous expression:

$$LTE = y'_n + \frac{y''_n}{2}h + \frac{y'''_n}{6}h^2 - (y'_n + fracy''_n 2h + \frac{y'''_n}{4}h^2)$$

$$LTE = -\frac{1}{12}y'''_n h^2 \qquad (3)$$

Plotting the region of absolute stability for this method:



Implement both schemes on computer.

Algorithm 1: Adams-Bashforth

```
function [t,y] = ab(interval, initial, n)

    h = (interval(2) - interval(1)) / n;
    y(1,:) = initial;
    t(1) = inter(1);
    t(2) = t(1) + h;
    y(i+1,:) = x + h*(my_func(t,x) + my_func(t+h,x+h*z1))/2;
    f(i,:) = my_func(t(i),y(i,:));

    for i=2:n
        t(i+1) = t(i) + h;
        f(i,:) = my_func(t(i),y(i,:));
        y(i+1,:) = y(i,:) + h*(3*f(i,:)/2 - f(i-1,:)/2);
    end
end
```

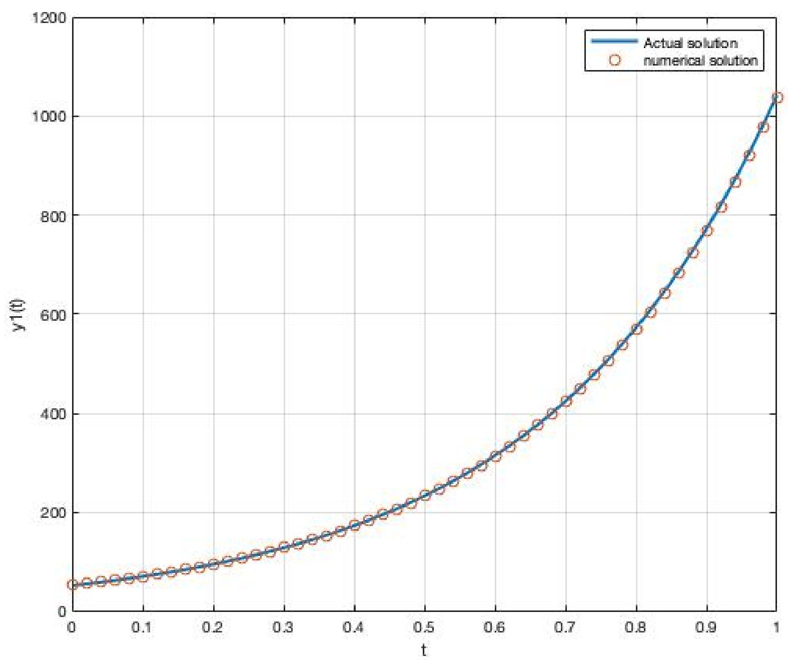
Algorithm 2: Adams-Moulton

```
function [t,y] = am(interval, initial, n)

    h = (interval(2) - interval(1)) / n;
    y(1,:) = initial;
    t(1) = inter(1);
    t(2) = t(1) + h;
    y(i+1,:) = x + h*(my_func(t,x) + my_func(t+h,x+h*z1))/2;
    f(i,:) = my_func(t(i),y(i,:));

    for i=2:n
        t(i+1) = t(i) + h;
        f(i,:) = my_func(t(i),y(i,:));
        y(i+1,:) = y(i,:) + h*(f(i-1,:)/2 - f(i,:)/2);
    end
end
```

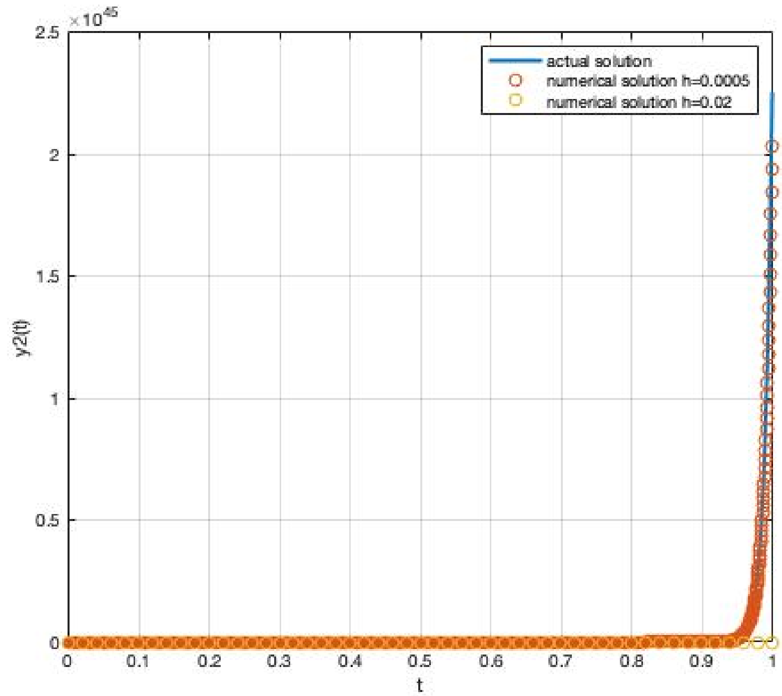
Solve the following ODEs using both schemes:

$$\frac{dy_1}{dt} = -5y_1 + 3y_2$$
$$\frac{dy_2}{dt} = 100y_1 - 301y_2$$

Solving the first ODE with the Adams-Bashforth explicit method ($h = 0.02$):



Solving the second ODE with the Adams-Bashforth explicit method:



In the first example, low error is achieved with $h = 0.02$, while with the second example using the same h value yields a much higher error, and an inability of the function to estimate the behavior of the real solution. A much smaller h value of $h = 0.0005$ was used to get low error. This makes sense within the paradigm of LTE, in that a smaller h value can give smaller error.

Part 2

Task: Recreate simulations from figures 1 and 2 from the 2013 Nature paper "Nonlinear Growth Kinetics of Breast Cancer Stem Cells: Implications for Cancer Stem Cell Targeted Therapy"

Four systems of ODEs were proposed in this paper for studying cancer stem cell growth kinetics. In order to recreate these figures, I primarily used the Manipulate function within Mathematica so I could easily change parameters within these systems. In this papers, some of the parameters were reported as actual values, and others were reported as ratios. The systems of interest are the following:

System 1: no feedback control

$$\frac{dx_0(t)}{dt} = (p_0 - q_0)v_0 x_0(t) - d_0 x_0(t)$$

$$\frac{dx_1(t)}{dt} = (1 - p_0 + q_0)v_0 x_0(t) + (p_1 - q_1)v_1 x_1(t) - d_1 x_1(t)$$

$$\frac{dx_2(t)}{dt} = (1 - p_1 + q_1)v_1 x_1(t) - d_2 x_2(t)$$

System 2: type 1 feedback control

$$\frac{dx_0(t)}{dt} = (p_0 - q_0)\frac{v_0}{1 + \beta_0(x_2(t-\tau))^2} x_0(t) - d_0 x_0(t)$$

$$\frac{dx_1(t)}{dt} = (1 - p_0 + q_0)\frac{v_0}{1 + \beta_0(x_2(t-\tau))^2} x_0(t) + (p_1 - q_1)\frac{v_1}{1 + \beta_0(x_2(t-\tau))^2} x_1(t) - d_1 x_1(t)$$

$$\frac{dx_2(t)}{dt} = (1 - p_1 - q_1)\frac{v_1}{1 + \beta_0(x_2(t-\tau))^2}x_1(t) - d_2x_2(t)$$

System 3: type 2 feedback control

$$\frac{dx_0(t)}{dt} = (\frac{p_0}{1 + \gamma_1^0(x_2(t-\tau))^2} - \frac{q_0}{1 + \gamma_2^0(x_2(t-\tau))^2})v_0x_0(t) - d_0x_0(t)$$

$$\frac{dx_1(t)}{dt} = (1 - \frac{p_0}{1 + \gamma_1^0(x_2(t-\tau))^2} + \frac{q_0}{1 + \gamma_2^0(x_2(t-\tau))^2})v_0x_0(t) + (\frac{p_1}{1 + \gamma_1^1(x_2(t-\tau))^2} - \frac{q_1}{1 + \gamma_2^1(x_2(t-\tau))^2})v_1x_2(t) - d_1x_1(t)$$

$$\frac{dx_2(t)}{dt} = (1 - \frac{p_1}{1 + \gamma_1^1(x_2(t-\tau))^2} - \frac{q_1}{1 + \gamma_2^1(x_2(t-\tau))^2})v_1x_1(t) - d_2x_2(t)$$

System 4: type 1 and type 2 feedback control

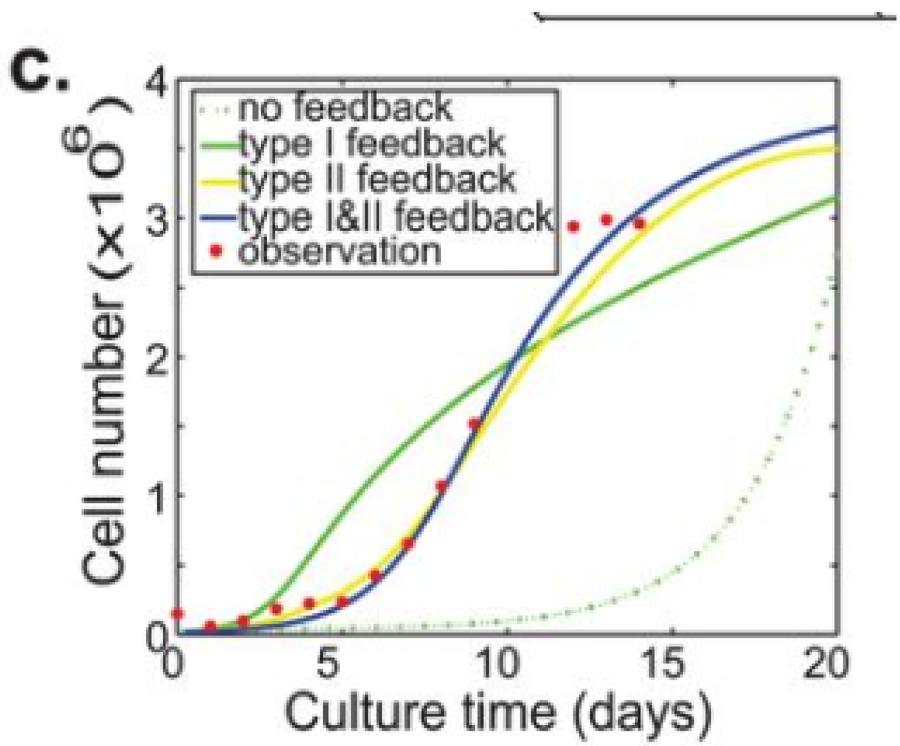
$$\frac{dx_0(t)}{dt} = (\frac{p_0}{1 + \gamma_1^0(x_2(t-\tau))^2} - \frac{q_0}{1 + \gamma_2^0(x_2(t-\tau))^2})\frac{v_0}{1 + \beta_0(x_2(t-\tau))^2}x_0(t) - d_0x_0(t$$

$$\frac{dx_1(t)}{dt} = (1 - \frac{p_0}{1 + \gamma_1^0(x_2(t-\tau))^2} + \frac{q_0}{1 + \gamma_2^0(x_2(t-\tau))^2})\frac{v_0}{1 + \beta_0(x_2(t-\tau))^2}x_0(t)$$
$$+(\frac{p_1}{1 + \gamma_1^1(x_2(t-\tau))^2} - \frac{q_1}{1 + \gamma_2^1(x_2(t-\tau))^2})\frac{v_1}{1 + \beta_0(x_2(t-\tau))^2}x_1(t) - d_1x_1(t)$$

$$\frac{dx_2(t)}{dt} = (1 - \frac{p_1}{1 + \gamma_1^1(x_2(t-\tau))^2} - \frac{q_1}{1 + \gamma_2^1(x_2(t-\tau))^2})\frac{v_1}{1 + \beta_0(x_2(t-\tau))^2}x_1(t) - d_2x_2(t)$$

Once these systems were put into Mathematica, NDSolve was used to numerically solve them in order to recreate the plots from the 2013 Nature paper. Parameter exploration was done manually using the Mathematica function Manipulate. This is an exercise in patiently and delicately balancing all these parameters in order to obtain the required output plots. Below is my attempt at recreation of the figure 1c.

The plot from the Nature 2013 paper that we are tasked with recreating:



Exact same equations put into Mathematica, hours of manipulating parameters. With more time and determination one ought to be able to get them to match up.