Joseph Ingenito

5/10/19

Project

Part I

1)

(a)

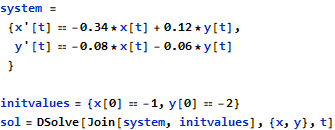






(b)

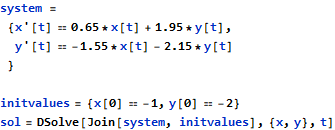






(c)





{{x -> Function[{t}, (-1. +

0. i) e^(-0.75 t) ((1. + 0. i) Cos[

1.03078 t] + (5.14176 + 3.1486\*10^-17 i) Sin[1.03078 t])],

y -> Function[{t}, -2. e^(-0.75 t) (1. Cos[

1.03078 t] - (2.11006 + 5.50981\*10^-17 i) Sin[1.03078 t])]}}

I couldn’t preserve Mathematica’s Formatting for this solution because the output is too long.

2)

For each system I made functions that also return the true solution found on Mathematica so making and editing the plots can be done easily.

(a)

function [t,x,y] = EulSystem\_1(h, t0, x0, y0, tf)

k = 1;

t(k) = t0;

x(k) = x0;

y(k) = y0;

while((t(k) + h <= tf + 10^-10))

t(k+1) = t(k) + h;

x(k+1) = x(k) + h\*(-0.5\*x(k) + 0.6\*y(k));

y(k+1) = y(k) + h\*(-0.4\*x(k) + 0.9\*y(k));

k = k + 1;

end

end

function [t,x,y] = TrueSolution\_1()

t = 0:0.1:4;

x = -exp(0.7.\*t);

y = -2.\*exp(0.7.\*t);

end

(b)

function [t,x,y] = EulSystem\_2(h, t0, x0, y0, tf)

k = 1;

t(k) = t0;

x(k) = x0;

y(k) = y0;

while((t(k) + h <= tf + 10^-10))

t(k+1) = t(k) + h;

x(k+1) = x(k) + h\*(-0.34\*x(k) + 0.12\*y(k));

y(k+1) = y(k) + h\*(-0.08\*x(k) - 0.06\*y(k));

k = k + 1;

end

end

function [t,x,y] = TrueSolution\_2()

t = 0:0.1:4;

x = -exp(-0.1.\*t);

y = -2.\*exp(-0.1.\*t);

end

(c)

function [t,x,y] = EulSystem\_3(h, t0, x0, y0, tf)

k = 1;

t(k) = t0;

x(k) = x0;

y(k) = y0;

while((t(k) + h <= tf + 10^-10))

t(k+1) = t(k) + h;

x(k+1) = x(k) + h\*(0.65\*x(k) + 1.95\*y(k));

y(k+1) = y(k) + h\*(-1.55\*x(k) - 2.15\*y(k));

k = k + 1;

end

end

function [t,x,y] = TrueSolution\_3()

t = 0:0.1:4;

x = -exp(-0.75.\*t).\*(cos(1.03078.\*t) + 5.14176.\*sin(1.03078.\*t));

y = -2.\*exp(-0.75.\*t).\*(cos(1.03078.\*t)-2.11006.\*sin(1.03078.\*t));

end

3)

function CreateSubplots(t1,x1,y1,t2,x2,y2,tvec,xvec,yvec)

ax1 = subplot(4,2,1);

ax2 = subplot(4,2,2);

ax34 = subplot(4,2,[3,4]);

ax5 = subplot(4,2,5);

ax6 = subplot(4,2,6);

ax78 = subplot(4,2,[7,8]);

plot(ax1,t1,x1,"blue",tvec,xvec,"black")

title(ax1,"x(t) vs t with h=0.25")

xlabel(ax1,"t Axis")

ylabel(ax1,"x(t) Axis")

plot(ax2,t1,y1,"blue",tvec,yvec,"black")

title(ax2,"y(t) vs t with h=0.25")

xlabel(ax2,"t Axis")

ylabel(ax2,"y(t) Axis")

plot(ax34,x1,y1,"blue",xvec,yvec,"black")

title(ax34,"y(t) vs x(t) with h=0.25")

xlabel(ax34,"x(t) Axis")

ylabel(ax34,"y(t) Axis")

plot(ax5,t2,x2,"blue",tvec,xvec,"black")

title(ax5,"x(t) vs t with h=0.1")

xlabel(ax5,"t Axis")

ylabel(ax5,"x(t) Axis")

plot(ax6,t2,y2,"blue",tvec,yvec,"black")

title(ax6,"y(t) vs t with h=0.1")

xlabel(ax6,"t Axis")

ylabel(ax6,"y(t) Axis")

plot(ax78,x2,y2,"blue",xvec,yvec,"black")

title(ax78,"y(t) vs x(t) with h=0.1")

xlabel(ax78,"x(t) Axis")

ylabel(ax78,"y(t) Axis")

end

(a)

>> [t1,x1,y1] = EulSystem\_1(0.25,0,-1,-2,4);

>> [t2,x2,y2] = EulSystem\_1(0.1,0,-1,-2,4);

>> [tvec,xvec,yvec] = TrueSolution\_1();

>> CreateSubplots(t1,x1,y1,t2,x2,y2,tvec,xvec,yvec);



Figure 1: Plots of the solutions to the first system with step sizes 0.25, and 0.1

(b)

>> [t1,x1,y1] = EulSystem\_2(0.25,0,-1,-2,4);

>> [t2,x2,y2] = EulSystem\_2(0.1,0,-1,-2,4);

>> [tvec,xvec,yvec] = TrueSolution\_2();

>> CreateSubplots(t1,x1,y1,t2,x2,y2,tvec,xvec,yvec);



Figure 2: Plots of the solutions for the second system with step sizes 0.25, and 0.1

(c)

>> [t1,x1,y1] = EulSystem\_3(0.25,0,-1,-2,4);

>> [t2,x2,y2] = EulSystem\_3(0.1,0,-1,-2,4);

>> [tvec,xvec,yvec] = TrueSolution\_3();

>> CreateSubplots(t1,x1,y1,t2,x2,y2,tvec,xvec,yvec);



Figure 3: Plots of the solutions for the third system with step sizes 0.25, and 0.1

4)

(a)

Using a step size of 0.25 deviated further from the true solution as x approaches 4 as expected, but as x gets arbitrarily close to 4 the estimated solution deviates too far from the true solution to give a meaningful result. With a step size of 0.1 the estimated solution is close enough to the true solution to give a meaningful approximation. Also the solution for y(t) is exactly 2 \* x(t) so plotting y(t) vs x(t) actually produces a linear graph.

(b)

The approximated solutions with a step size of 0.25 and 0.1 were both almost identical to the true solution. For this system even a step size of only 0.25 is close enough to the true solution to give a meaningful approximation. Similar to the first system the solution for y(t) is exactly 2 \* x(t) so plotting y(t) vs x(t) also produces a linear graph. This result can most likely be attributed to the initial conditions for x(t) and y(t) since x(0) = -1 and y(0) = -2, and since the solutions for these two systems don’t involve cosines or sines it can almost be expected for y(t) to be exactly double x(t).

(c)

The solutions for the third system involved cosines and sines, and it seems that for both a step size of 0.25 and 0.1 the inflection points seem to be the same for the estimated solution and the true solution. The end points of the interval also are both approximated close to the true solution for both step sizes. What differs between the two step sizes are the approximated portions of the curve in between the two inflection points, in which a step size of 0.1 is close enough to the true solution for a meaningful approximation.

5)

>> a1 = [-0.5 0.6; -0.4 0.9];

>> a2 = [-0.34 0.12; -0.08 -0.06];

>> a3 = [0.65 1.95; -1.55 -2.15];

>> eig(a1)

ans =

-0.3000

0.7000

>> eig(a2)

ans =

-0.3000

-0.1000

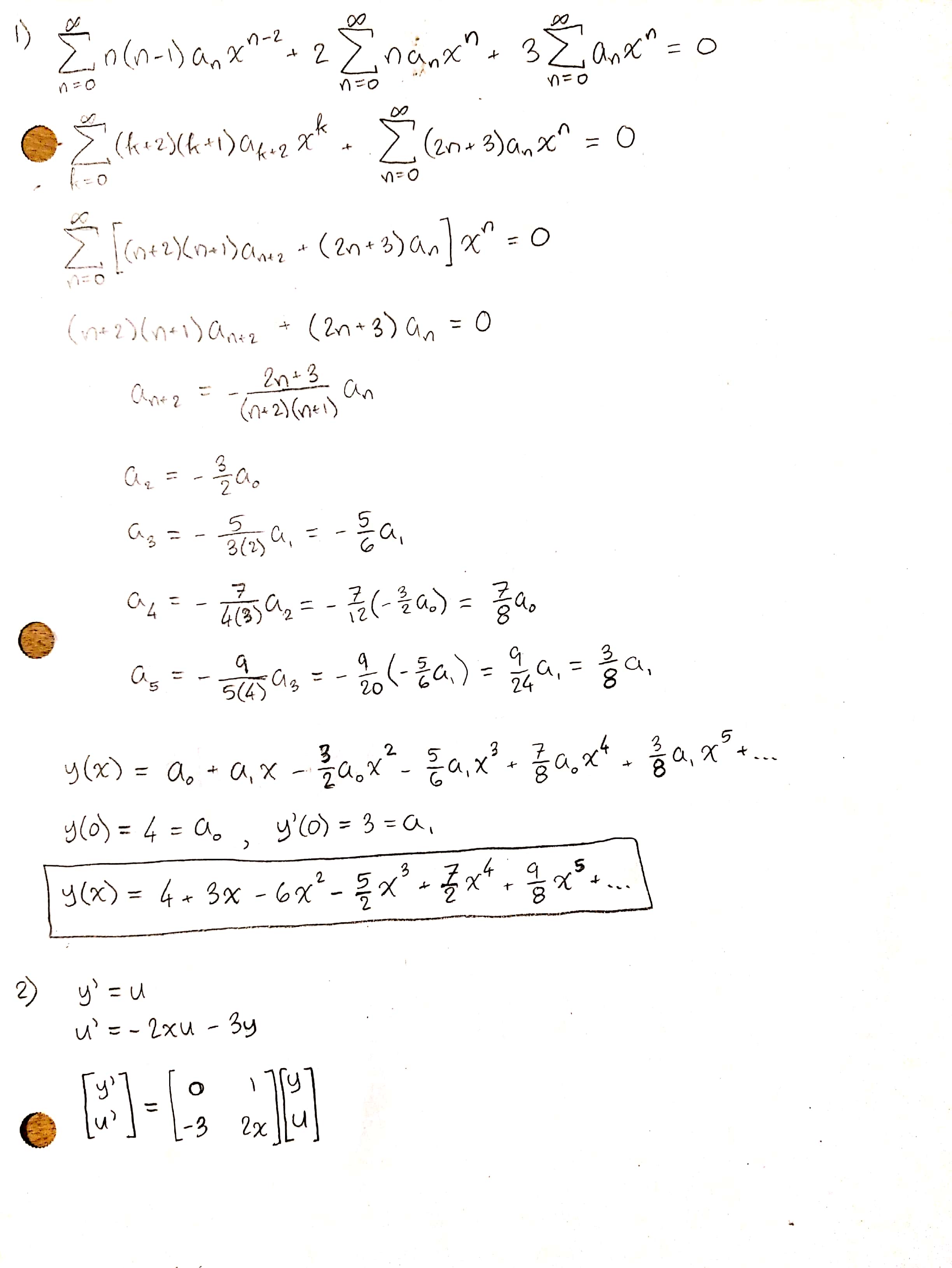
>> eig(a3)

ans =

-0.7500 + 1.0308i

-0.7500 - 1.0308i

Part II



3)

function [x,u,y] = EulSystem\_Part2(h, x0, y0, u0, xf)

k = 1;

x(k) = x0;

y(k) = y0;

u(k) = u0;

while((x(k) + h <= xf + 10^-10))

x(k+1) = x(k) + h;

y(k+1) = y(k) + h\*u(k);

u(k+1) = u(k) + h\*(-3\*y(k) - 2\*x(k)\*u(k));

k = k + 1;

end

end

function [x,y] = SeriesSolution(y0, yprime0)

x = 0:0.1:2;

y = y0.\*(1 - (3/2).\*x.^2 + (7/8).\*x.^4) + yprime0.\*(x - (5/6).\*x.^3 + (3/8).\*x.^5);

end

function CreateSubplot(x1,y1,x2,y2,xseries,yseries)

ax1 = subplot(2,1,1);

ax2 = subplot(2,1,2);

plot(ax1,x1,y1,"blue",xseries,yseries,"black")

plot(ax2,x2,y2,"blue",xseries,yseries,"black")

title(ax1,"Plot of y(x) vs x with h=0.2")

title(ax2,"Plot of y(x) vs x with h=0.1")

xlabel(ax1,"x Axis")

xlabel(ax2,"x Axis")

ylabel(ax1,"y(x) Axis")

ylabel(ax2,"y(x) Axis")

end

4)

>> [x1,u1,y1] = EulSystem\_Part2(0.2,0,4,3,2);

>> [x2,u2,y2] = EulSystem\_Part2(0.1,0,4,3,2);

>> [xseries,yseries] = SeriesSolution(4,3);

>> CreateSubplot(x1,y1,x2,y2,xseries,yseries);



5)

The Euler’s Method approximation with a step size of 0.2 was closer to the approximated series solution than expected. With a step size of 0.1 the Euler’s Method approximation lines up almost exactly with the first 6 terms of the approximated series solution. If the ratio test is used to determine the interval of convergence from the recurrence relationship, the series seems to converge on , but the approximated series only seems to converge on . This can be contributed to the fact that only the first 6 terms were used from the series approximation.

Part III

Functions:

function [t,x1,x2] = DataPoints()

t = [0 7.143 14.286 21.429 28.571 35.714 42.857 50 57.143 64.286 71.429 78.571 85.714 92.857 100];

x1 = [0.623 0.374 0.249 0.183 0.145 0.12 0.103 0.089 0.078 0.068 0.06 0.053 0.047 0.041 0.037];

x2 = [0 0.113 0.151 0.157 0.15 0.137 0.124 0.11 0.098 0.087 0.077 0.068 0.06 0.053 0.047];

end

function CreateSubplots(t,x1,x2)

ax1 = subplot(2,2,1);

ax2 = subplot(2,2,2);

ax3 = subplot(2,2,3);

ax4 = subplot(2,2,4);

plot(ax1,t,x1);

plot(ax2,t,x2);

plot(ax3,t,log(x1));

plot(ax4,t,log(x2));

title(ax1,"Plot of x1(t)");

title(ax2,"Plot of x2(t)");

title(ax3,"Plot of ln(x1(t))");

title(ax4,"Plot of ln(x2(t))");

xlabel(ax1,"t Axis");

xlabel(ax2,"t Axis");

xlabel(ax3,"t Axis");

xlabel(ax4,"t Axis");

ylabel(ax1,"x1(t) Axis");

ylabel(ax2,"x2(t) Axis");

ylabel(ax3,"ln(x1(t)) Axis");

ylabel(ax4,"ln(x2(t)) Axis");

end

function [p1,p2] = PolyFitDataSet(t,x1,x2,tmax)

tvec = 0:0.1:tmax;

n = numel(t) - 1;

p1 = polyfit(t,log(x1),n);

p2 = polyfit(t,log(x2),n);

y1 = polyval(p1,tvec);

y2 = polyval(p2,tvec);

ax1 = subplot(2,1,1);

ax2 = subplot(2,1,2);

plot(ax1,t,log(x1),'o',tvec,y1);

plot(ax2,t,log(x2),'o',tvec,y2);

title(ax1,"Polynomial of Least Squares for ln(x1(t))");

title(ax2,"Polynomial of Least Squares for ln(x2(t))");

ylabel(ax1,"ln(x1(t)) Axis");

ylabel(ax2,"ln(x2(t)) Axis");

xlabel(ax1,"t Axis");

xlabel(ax2,"t Axis");

end

1)

2)

represents the rate of change of the concentration of the drug in the Central Compartment (Blood). The which represents the amount of the drug in the Tissue Compartment at rate c. The which represents the amount of drug in the Central Compartment at rate . Since the rate of change of concentrations are being modeled by , becomes negative.

represents the rate of change of the concentration of the drug in the Tissue Compartment. The which represents the amount of the drug in the Central Compartment at rate b. The which represents the amount of drug in the Tissue Compartment at rate c. Since the rate of change of concentrations are being modeled by , becomes negative.

3)

To show that the coefficient matrix has 2 distinct, real, and negative eigenvalues, we first show that the discriminant of the characteristic polynomial is positive which implies 2 distinct, real roots of the characteristic polynomial. Shown below is the characteristic polynomial of the coefficient matrix:

We now show that the discriminant of the characteristic polynomial is positive:

Since , the above inequality is true and the discriminant must be positive. Since the discriminant is positive, there are 2 distinct, real roots to the characteristic polynomial.

We now show that the 2 roots of the characteristic polynomial are negative by using relationship between the eigenvalues of a matrix and its trace, and determinant. We know that both eigenvalues must add to the trace of the matrix, and the must multiply to the determinant of the matrix.

Since and , the only possible signs for both eigenvalues are

4)

Since and are distinct eigenvectors of A, the concentrations of the drug in each compartment tend to 0 for large values of t.

5)

Since the eigenvalues satisfy , and so is transient:

6)

For large values of t for which ,

Since , , which is approximately a straight line with a slope of .

To find the y – intercept in terms of the defined parameters, substitute 0 in for t which gives:

So the y – intercept is

7)

>> [t,x1,x2] = DataPoints();

>> CreateSubplots(t,x1,x2);



8)

The only data point that didn’t seem to fit the rest of the data points in the set when plotted was the first data point (t = 0, x1 = 0.623, x2 = 0), so the first data point was removed in order to obtain a smooth Polynomial of Least Squares to fit the data set.

>> [p1,p2] = PolyFitDataSet(t,x1,x2,100)

p1 =

Columns 1 through 11

0.0000 -0.0000 0.0000 -0.0000 0.0000 -0.0000 0.0000 -0.0000 0.0007 -0.0163 0.2486

Columns 12 through 14

-2.4148 13.1866 -31.0206

p2 =

Columns 1 through 11

0.0000 -0.0000 0.0000 -0.0000 0.0000 -0.0000 0.0000 -0.0000 0.0005 -0.0121 0.1878

Columns 12 through 14

-1.8586 10.4563 -26.9367

From the vector returned from the PolyFitDataSet function:

The PolyFitDataSet function also creates the subplots with each approximation, the Figure created is shown on the next page:

