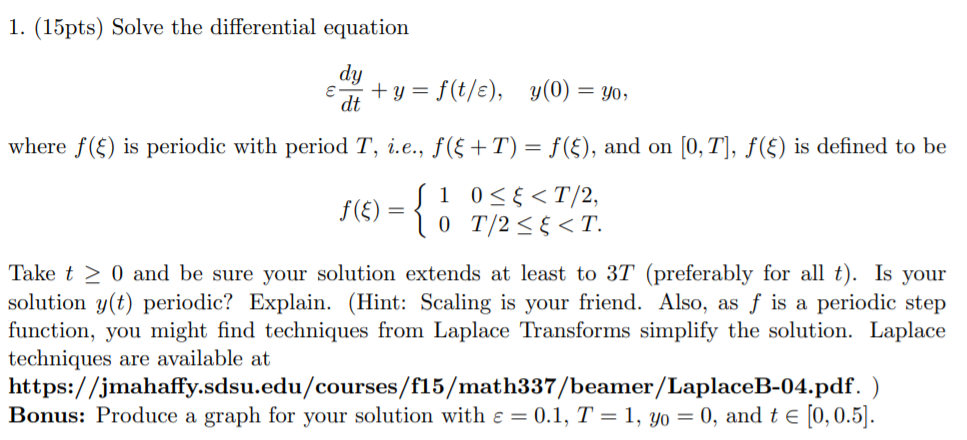
Stefan Cline

Midterm Exam

Math 537 – Ordinary Differential Equations

21 October 2021

**Problem 1**



We first notice that we want to have a function , so we’ll let

Then our equation that’s provided to us becomes:

Solving now for our first interval of ():

Now, utilizing our initial condition:

Hence, the first interval’s solution is:

To find the hand off point of the function (where we know that because if we adjust the interval again, ):

Now, for the other interval:

Now, utilizing the results from the last initial condition (again adjusting the interval we have ):

Placing this back into the second function gives:

So, for our two intervals we have the following expressions:

But, to keep things consistent we can replace again with its original expression as:

Now, to keep going until , we repeat the process, but we have instead new starting conditions because we know the function must be continuous. Effectively, we’ll have:

Where is the end point of the first full period.

Therefore, our function will be continuous again at . This gives:

Then, we know again we have

We’ll call the end point of the interval again, so:

So, plugging this in we have:

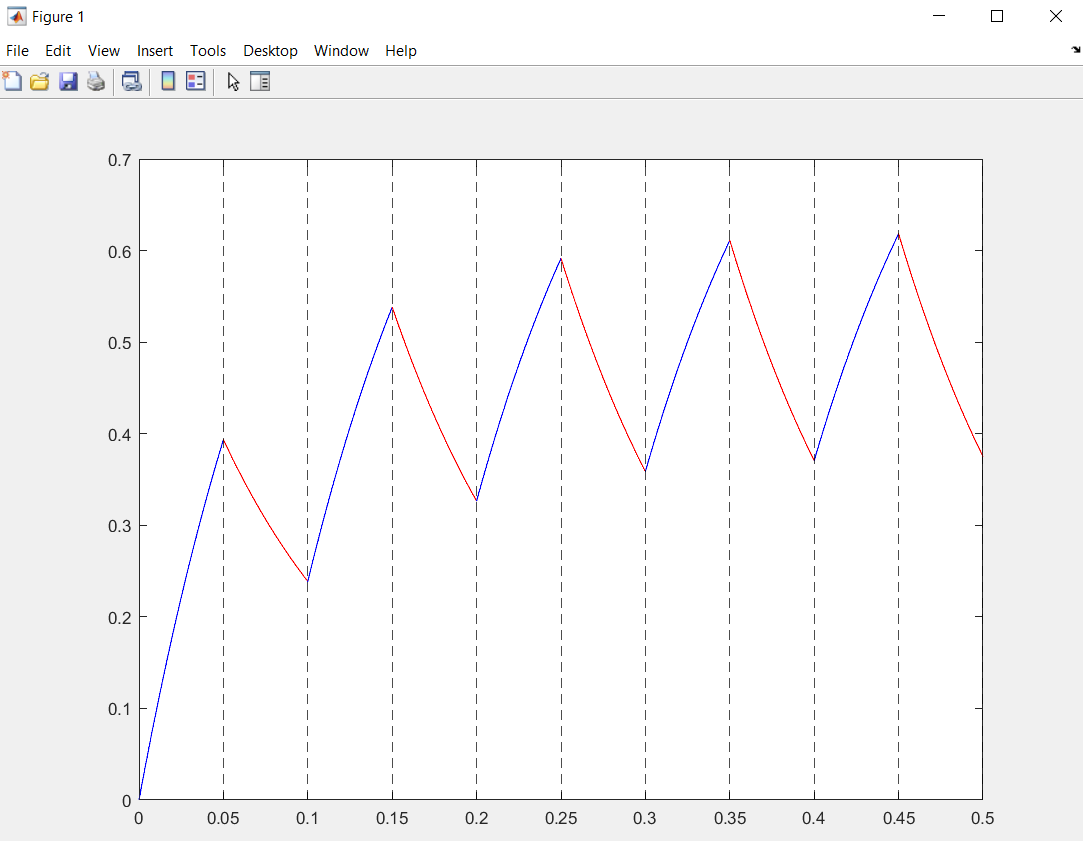
But this will iteratively hold between those two functions for finding the next starting point based off of the last intervals end point. Therefore, we have iteratively (where ):

From here though we can see that the isn’t periodic. This will be seen more easily graphically below given our parameters.

We note that there is a special case, where if we found precisely the right starting values, that we would immediately land on periodic requirements which would make the function periodic for that niche case. But generally no, the function is not periodic.

Bonus:

Utilizing MATLAB, and the values of and :



Code Used:

% Attempt with Actual Equations

t1 = linspace(.0, .05, 1000);

t2 = linspace(.05, .1, 1000);

yp1 = -exp(-(t1/.1))+1;

yp2 = (-exp(-(.1/2)/.1+1/2)+exp(1/2))\*exp(-t2/.1);

t3 = linspace(.1, .15, 1000);

t4 = linspace(.15, .2, 1000);

yp3 = -exp(-((t3-.1\*log(-yp2(end)+1)-.1)/.1))+1;

yp4 = yp3(end)\*exp(3\*1/(2))\*exp(-t4/.1);

t5 = linspace(.2, .25, 1000);

t6 = linspace(.25, .3, 1000);

yp5 = -exp(-((t5-.1\*log(-yp4(end)+1)-(2\*.1))/.1))+1;

yp6 = yp5(end)\*exp(5\*1/(2))\*exp(-t6/.1);

t7 = linspace(.3, .35, 1000);

t8 = linspace(.35, .4, 1000);

yp7 = -exp(-((t7-.1\*log(-yp6(end)+1)-(3\*.1))/.1))+1;

yp8 = yp7(end)\*exp(7\*1/(2))\*exp(-t8/.1);

t9 = linspace(.4, .45, 1000);

t10 = linspace(.45, .5, 1000);

yp9 = -exp(-((t9-.1\*log(-yp8(end)+1)-(4\*.1))/.1))+1;

yp10 = yp9(end)\*exp(9\*1/(2))\*exp(-t10/.1);

plot(t1, yp1, 'b')

hold on

plot(t2, yp2, 'r')

plot(t3, yp3, 'b')

plot(t4, yp4, 'r')

plot(t5, yp5, 'b')

plot(t6, yp6, 'r')

plot(t7, yp7, 'b')

plot(t8, yp8, 'r')

plot(t9, yp9, 'b')

plot(t10,yp10,'r')

xline(0)

xline(.05,'--k')

xline(.1,'--k')

xline(.15,'--k')

xline(.2,'--k')

xline(.25,'--k')

xline(.3,'--k')

xline(.35,'--k')

xline(.4,'--k')

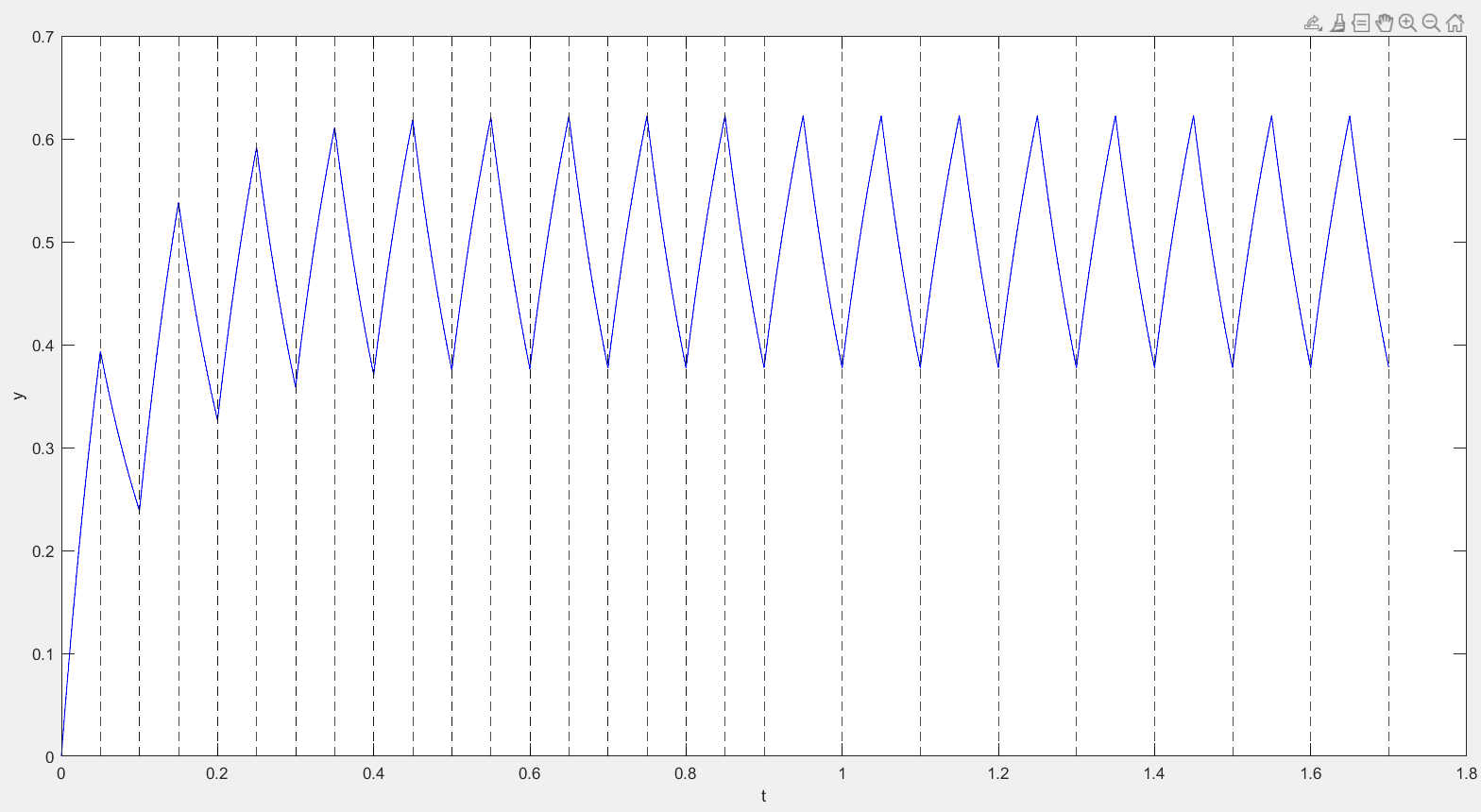
xline(.45,'--k')

xline(.5,'--k')

yline(0)

hold off

Our first image helps us see that this “arc” seems to level out, but finding only the first few iterations of this isn’t as convincing for the the long-term behavior. Therefore, we’ll generate some code and plot this out for a larger number of iterations. Here, we observe that the function itself isn’t periodic, but effectively mimics periodic behavior after some number of iterations have passed and the curve “settles” into this periodic mimicking behavior.



Code Used:

n = 2;

endset = 50;

t1 = linspace(.0, .05, 1000);

t2 = linspace(.05, .1, 1000);

yp1 = -exp(-(t1/.1))+1;

yp2 = (-exp(-(.1/2)/.1+1/2)+exp(1/2))\*exp(-t2/.1);

yp\_first\_last = yp1(end);

yp\_second\_last = yp2(end);

plot(t1, yp1, 'b')

hold on

plot(t2, yp2, 'r')

xline(0)

yline(0)

xline(.05,'--k')

xline(.1,'--k')

xlabel('t')

ylabel('y')

while n < endset

t\_first = linspace(.1\*(n-1), .1\*(n-1)+.05, 1000);

t\_second = linspace(.1\*(n-1)+.05, .1\*n, 1000);

yp\_odd\_current = -exp(-((t\_first-.1\*log(-yp\_second\_last+1)-((n-1)\*.1))/.1))+1;

yp\_even\_current = yp\_odd\_current(end)\*exp((2\*(n-1)+1)\*1/(2))\*exp(-t\_second/.1);

yp\_first\_last = yp\_odd\_current(end);

yp\_second\_last = yp\_even\_current(end);

plot(t\_first, yp\_odd\_current,'-b');

plot(t\_second, yp\_even\_current,'-b');

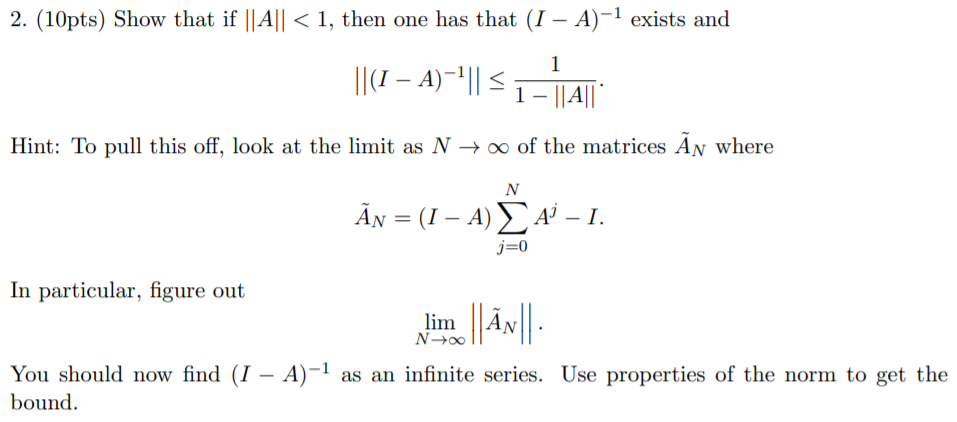
xline(.05\*n,'--k')

xline(.1\*n,'--k')

n = n + 1;

end

**Problem 2**



First, we note that if , then every entry must be less than one. To show this, we note that all of our different norms are equivalent. Viewing the simplest cases, namely and , we observe that sums in any direction (across rows or columns) won’t exceed 1, meaning that we must have for all .

Now, to show that exists for any , we’ll first assume that it doesn’t. Namely, we’ll let:

This would imply that the matrix is singular and therefore not invertible. Here, we suppose that is an eigenvalue for some vector such that:

Observing that if we add and subtract from we get

Multiply through by :

But we know that from above, hence:

However, this would imply that an eigenvalue of the above line would be one. But this isn’t possible given that we know that as it would require some to be greater than or equal to one. As such, we have that

by contradiction. Therefore, is nonsingular and must be invertible.

Now, we wish to show that

Let’s assume that

So,

But, clearly as we have that as every element in is less than one, so taking the elements to higher and higher powers will mean they approach zero. As such then we can also see that

Going back to our assumption of and setting it’s limit equal to zero:

Then, if we find their norms:

If we equate this to a geometric sum, we then know the form of a geometric sum is:

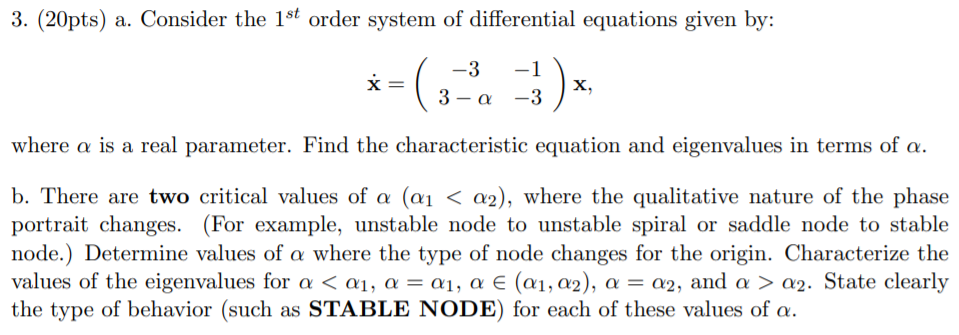
In this case however, . Therefore, we have that (by the triangle inequality):

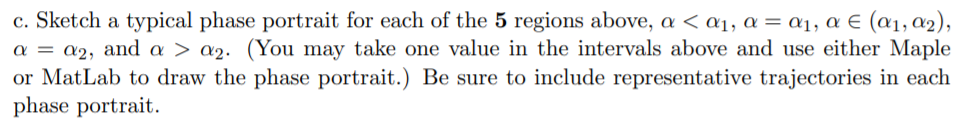
Hence, altogether we then arrive at our desired goal:

As desired, we’ve proven that exists if and that

Q.E.D.

**Problem 3**





**Part a)**

Observing the equation:

We can call matrix . So, finding

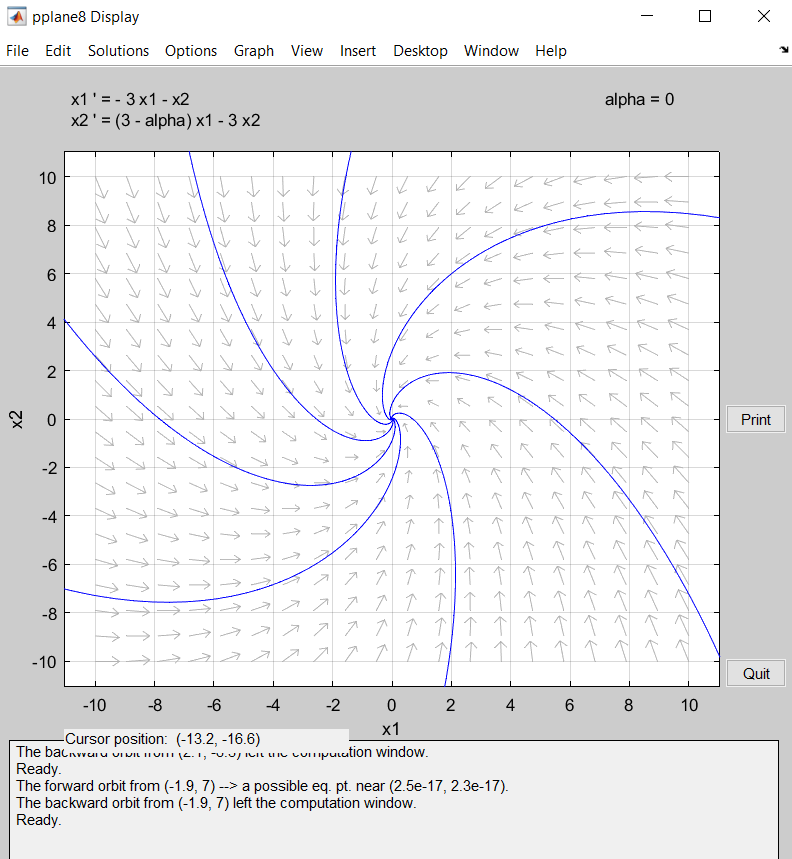
So, our two separate values for are:

**Part b)**

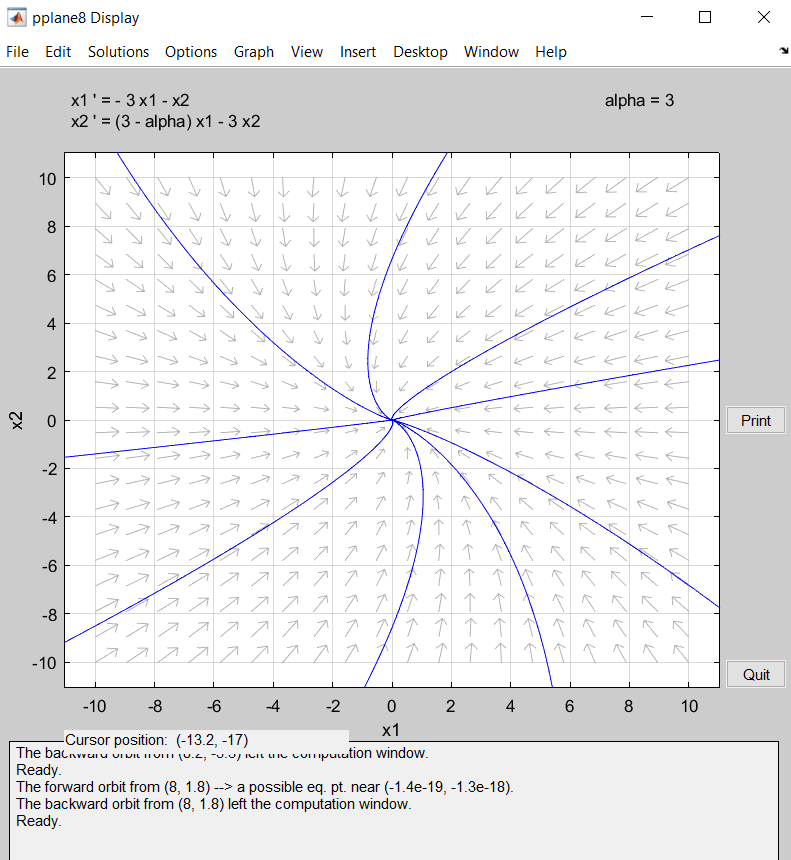
We’ll organize our possibilities into a chart for easier viewing and bookkeeping. Therefore, we have and .

|  |  |
| --- | --- |
| Interval | Description |
| 1. | Here, this is when we have which produces complex eigenvalues. Hence, we expect spiral activity. Now, because the real component of this eigenvalue is negative, it will act as a sink. Also, if we test a point, say , we see that we have:  Therefore, on the vertical axis, we see a pull down and to the left meaning we have a counterclockwise rotation.  Spiral Sink (Counterclockwise) |
| 2. | At 3 we see which is a repeated root. Hence, our eigenvalues are both , meaning our solutions will take on the form  Therefore, we’ll clearly have a stable sink.  Stable Sink |
| 3. | As we move to values greater than 3, we don’t see a change in the behavior of the phase planes. We continue to see sinks.  Stable Sink |
| 4. | At 12 we observe the most unique behavior. At this point we have:  Therefore, our solutions are:  Hence, our phase portrait will show a collapse to the nullcline.  Collapse to Nullcline |
| 5. | For a value greater than 12, we notice that we have the discriminant being greater than 3. Therefore, we’ll have two eigenvalues, one positive and one negative. Therefore, our solution will look like:  We can expect this to always give us a saddle node.  Saddle Node. |

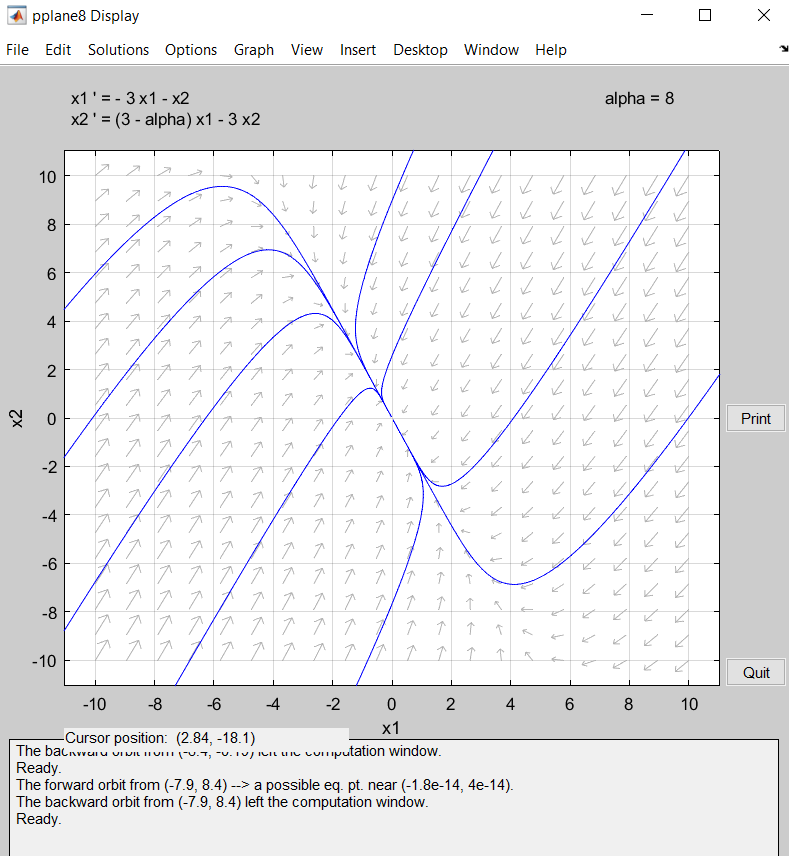
Region 1:



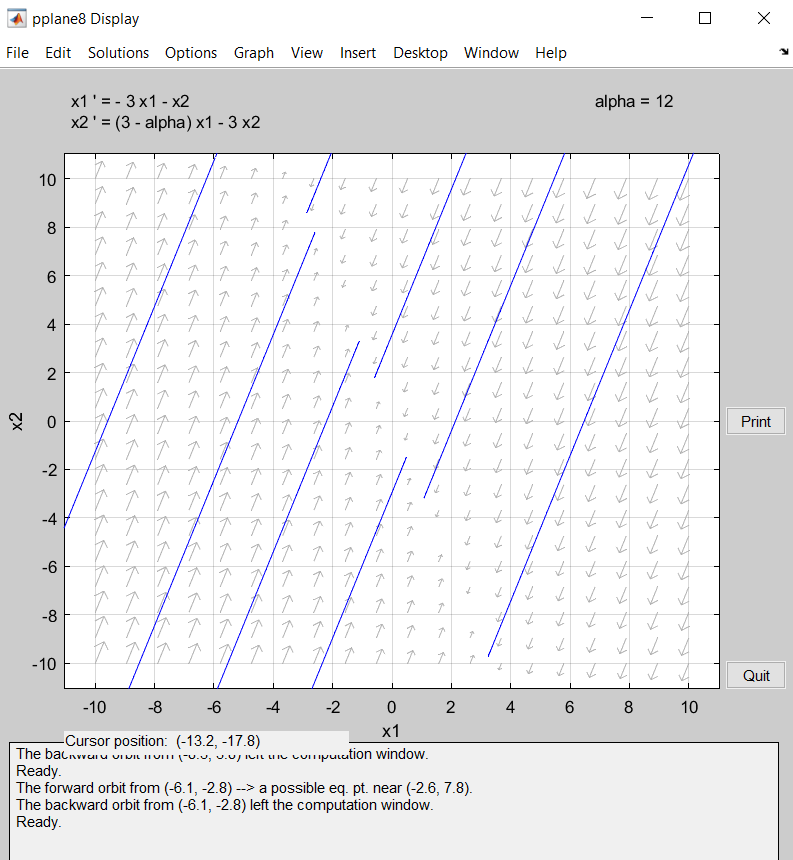
Region 2:



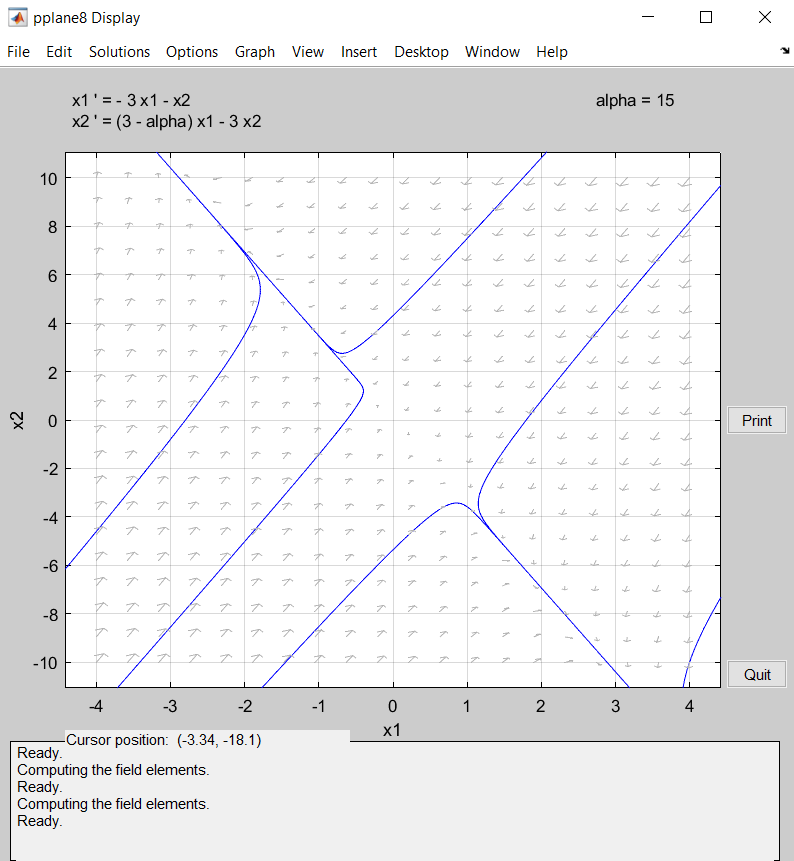
Region 3:



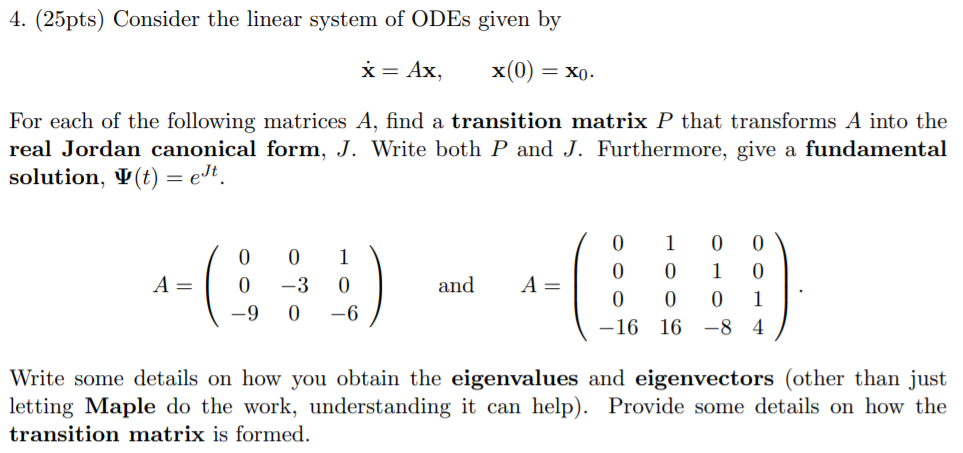
Region 4:



Region 5:



**Problem 4:**



Using the following series of commands in MAPLE gives us our transition matrix and our Jordan Matrix . Effectively here we’re transitioning from :

In order to find the eigenvalues, we’d compute:

Computing this gives the characteristic equation (done in MATLAB):

Now, using MATLAB to find the roots (we also can see graphically that is the only root to the equation for both real and complex values):

>> roots([-1,-9,-27,-27])

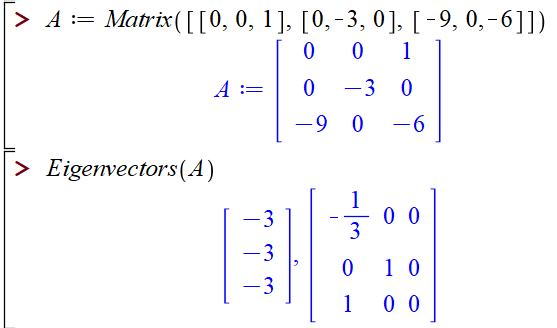
ans =

-3.0000 + 0.0000i

-3.0000 + 0.0000i

-3.0000 - 0.0000i

Hence, the only eigenvalue we get is . This can also be done in MAPLE:



Now, we see that maple gave us . We can verify by computing:

Picking :

So, we have:

Clearly here satisfies so we’re back at our produced value from MAPLE of . Now, using a Jordan Chain to find

We’ll pick . Then:

Solving:

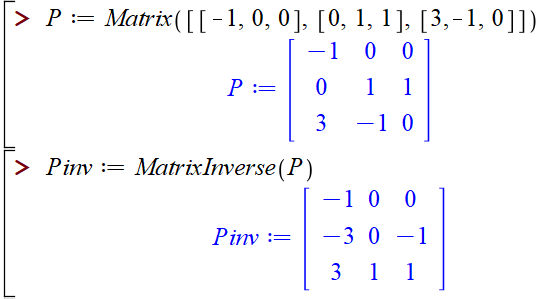
Hence, we have a solution at : So, . Now, for our last vector:

Let’s let . So,

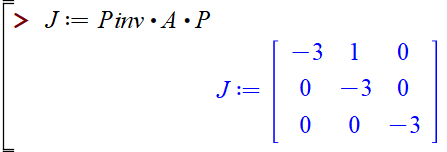
Then, we’ll pick for simplicity:

Therefore, our last vector is: . So, building our matrix from these three eigenvectors gives:

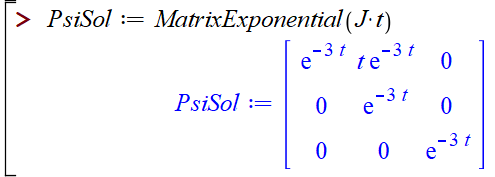
Using MAPLE to find :



Now, utilizing MAPLE we’ll compute :

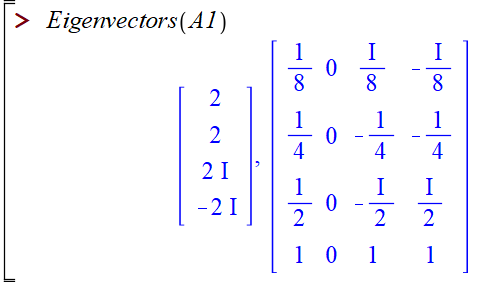


Therefore, we now know that finding a set of solutions will be :



(We note that we can find this by doing where and has the single one in the position. This will then give us Taylor Series expansions that will be and one which will then multiply together to give the above given matrix.)

We’ll now complete the same process again but with our other matrix. We’ll use MAPLE to do the tedious calculations of finding the first few eigenvalues and vectors for us, but knowing that if required the process would be the same as the above:



Therefore, we can build as follows (the second rows being built from the real and imaginary parts of the eigenvectors respectively):

Therefore, we need to find the second vector, in . Utilizing the Jordan Chain again:

Here now, we’ll utilize MATLAB in order to perform row reduced echelon operations that will give us our .

>> rref([A,[1;2;4;8]])

ans =

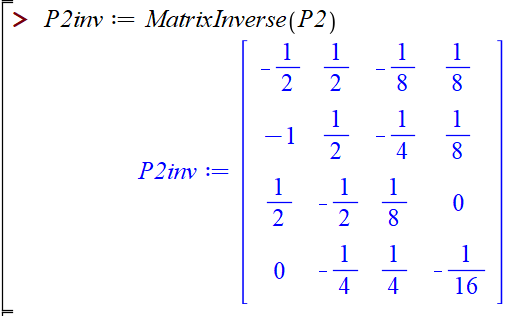
1.0000 0 0 -0.1250 -1.5000

0 1.0000 0 -0.2500 -2.0000

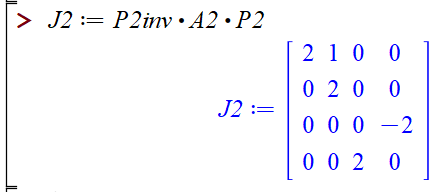
0 0 1.0000 -0.5000 -2.0000

0 0 0 0 0

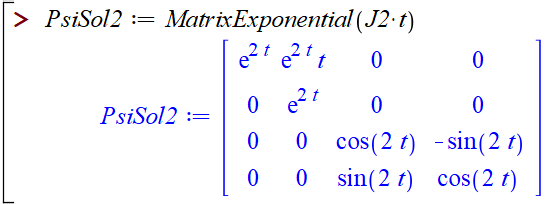
Therefore, we have if we simply pick . Placing this into the missing column of our matrix and finding :



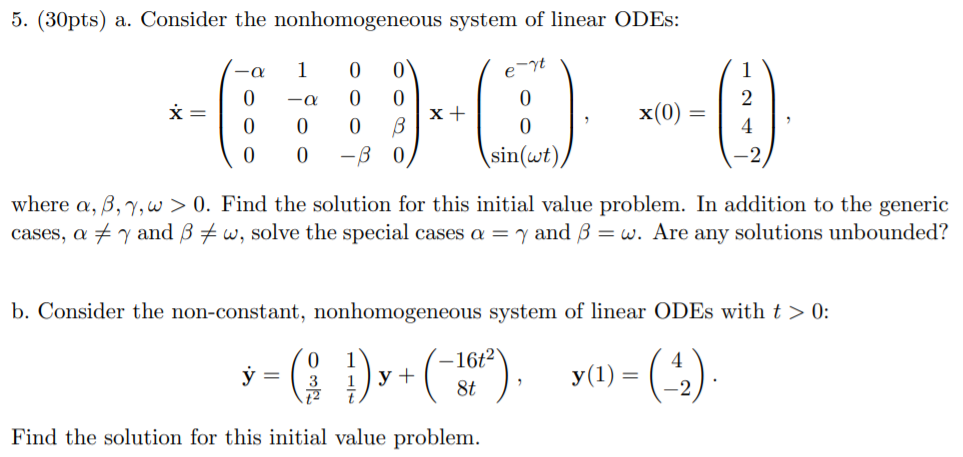
Then, finding :



Lastly, finding :



**Problem 5:**



**Part a)**

We notice that our first matrix, we’ll call it is already in Jordan Canonical Form, so we can write our fundamental solution as:

, where

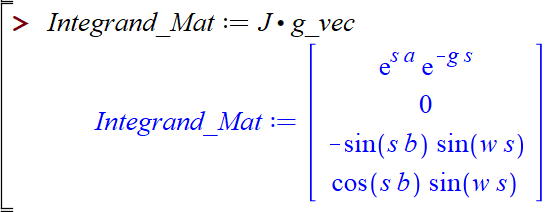
Therefore, we immediately have:

Now, we’ll find the solution to our equation by using the Variation of Constants Formula:

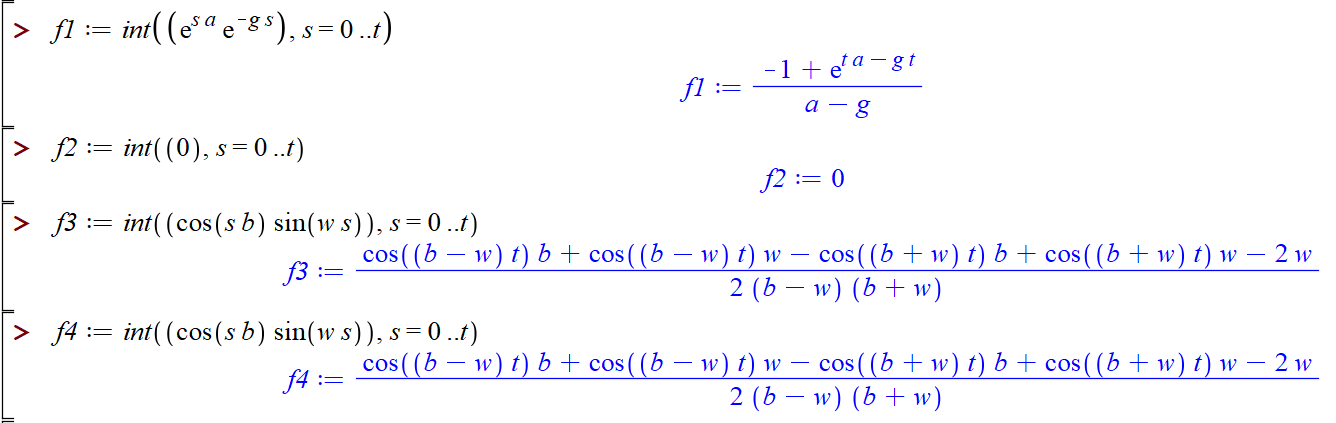
Noticing the negative one times the matrix we see that that becomes:

Then, our term can be expanded to:

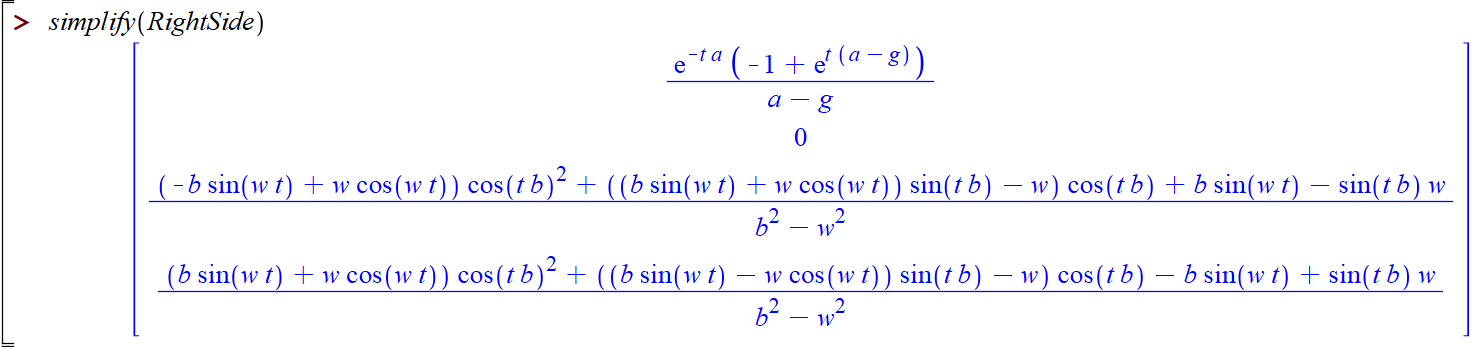
Multiplying the integrand:



Then, we need to find the integral of each of those separate pieces, using MAPLE again:

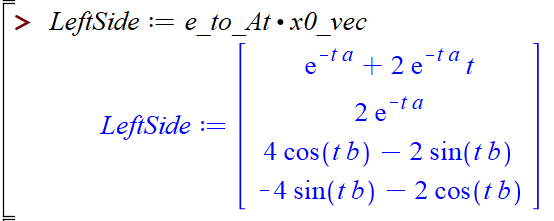


We’ll now make this into a matrix, and then multiply it by :

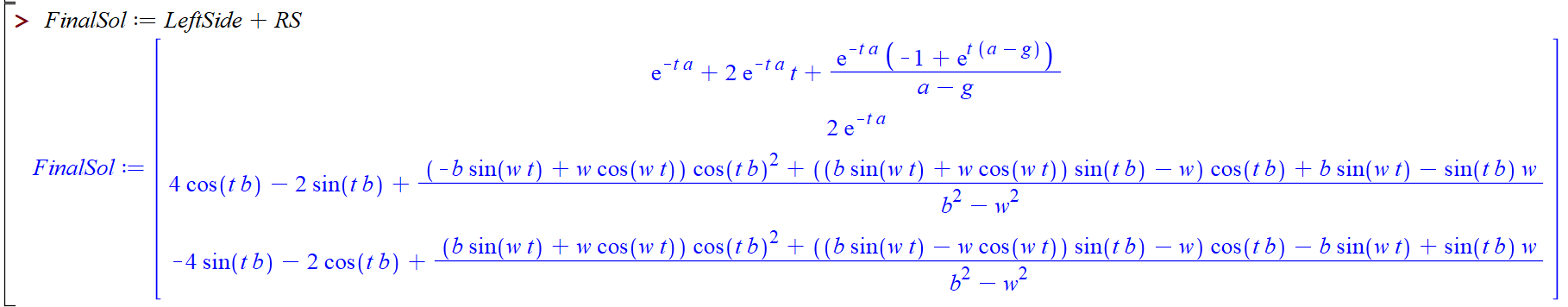


Now, we’ll find the left side of the equation. This will be:

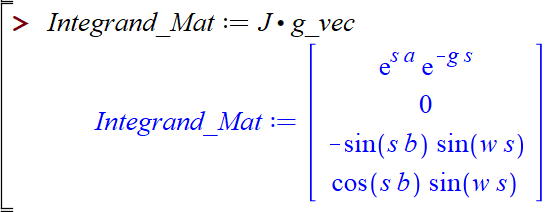
Using MAPLE:



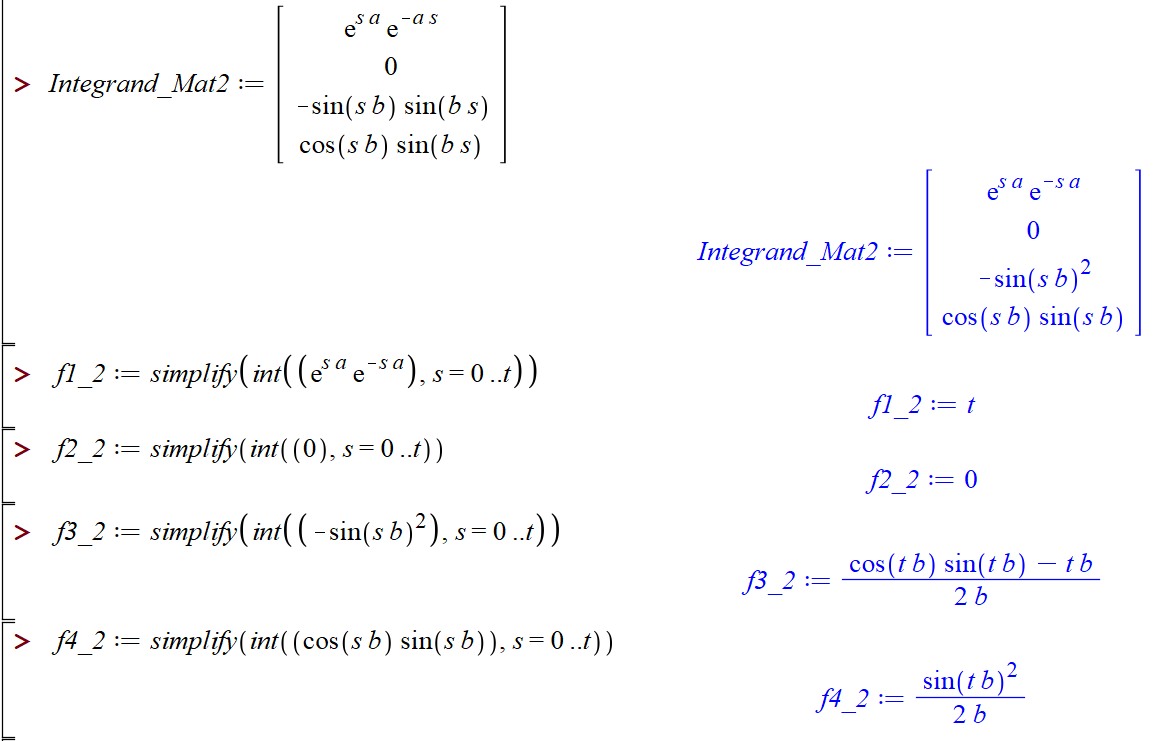
Now, adding those two pieces together will give us our final solution:



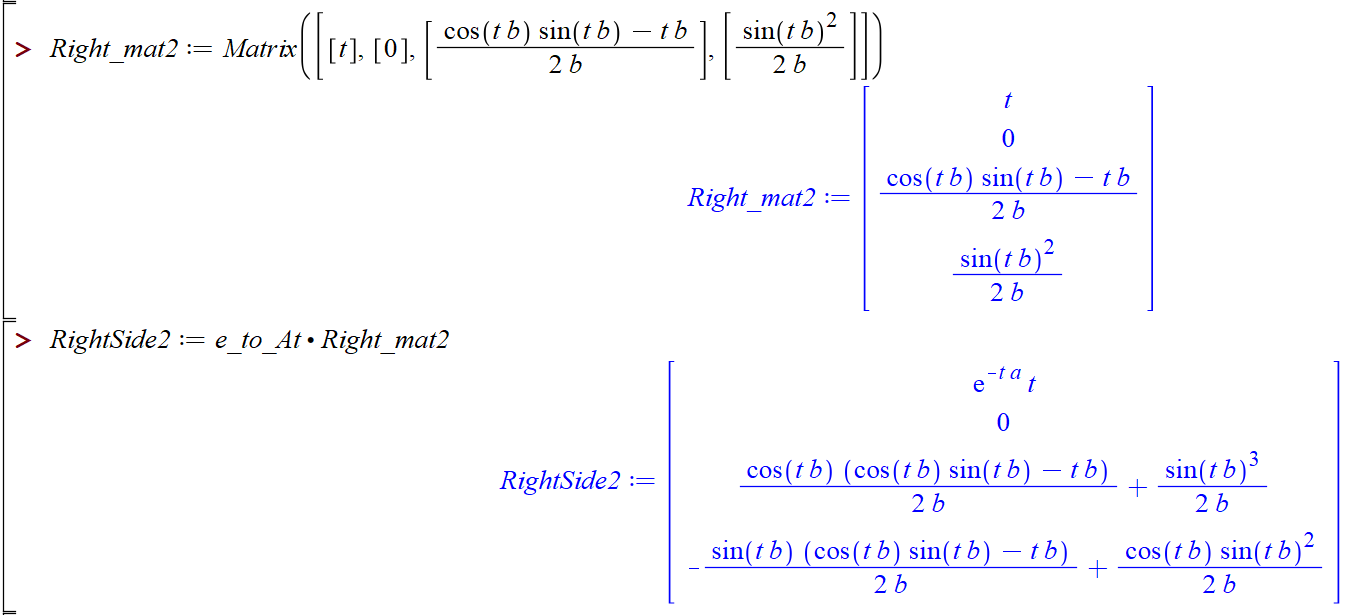
This above solution however we note is for and (this is clear because we’d get zero in the denominator for 3 pieces of our solution and therefore, we’ll treat them as a separate case). Here, if we instead notice earlier on in our computations, we set and then we’ll have:



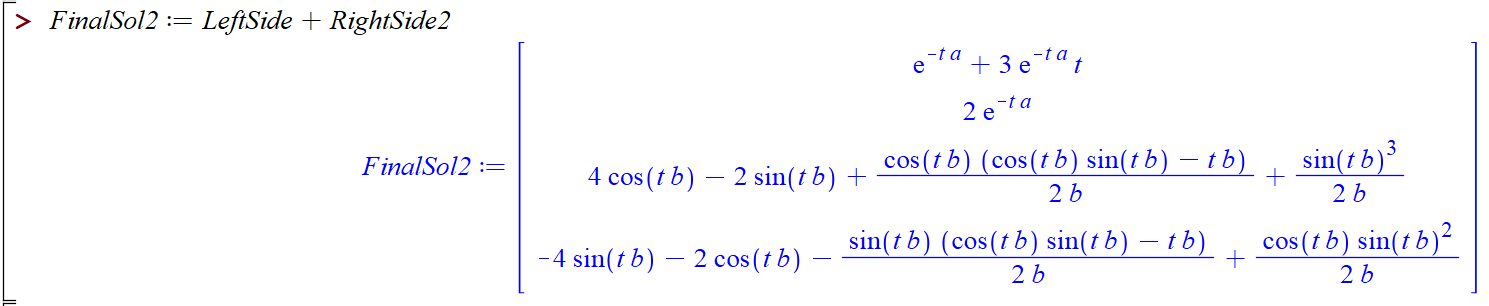
Changing the variables to be equal and then redoing our integrals:



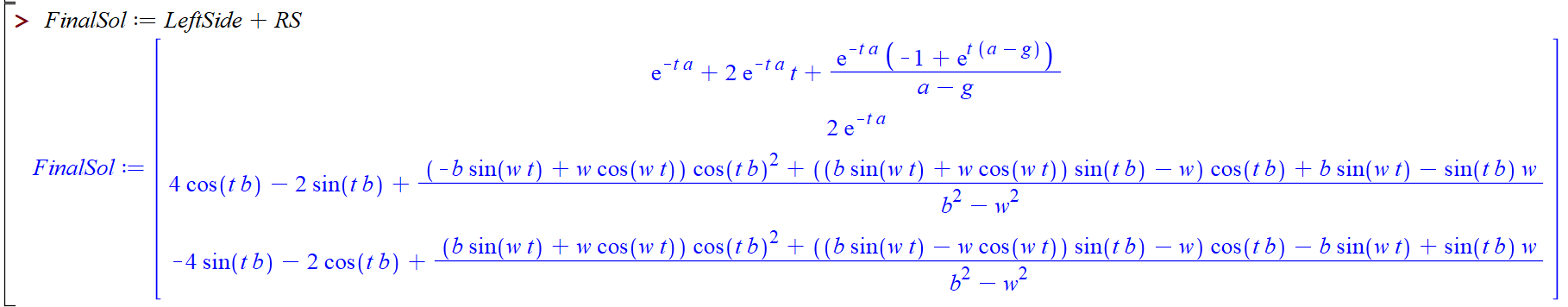
Now, multiplying this by :



Then adding this to the same thing as before as the left side of our equation gives us a new final solution of:



We’ll now evaluate which solutions, and their requisite pieces, are bounded or not. So, starting with the first solution:

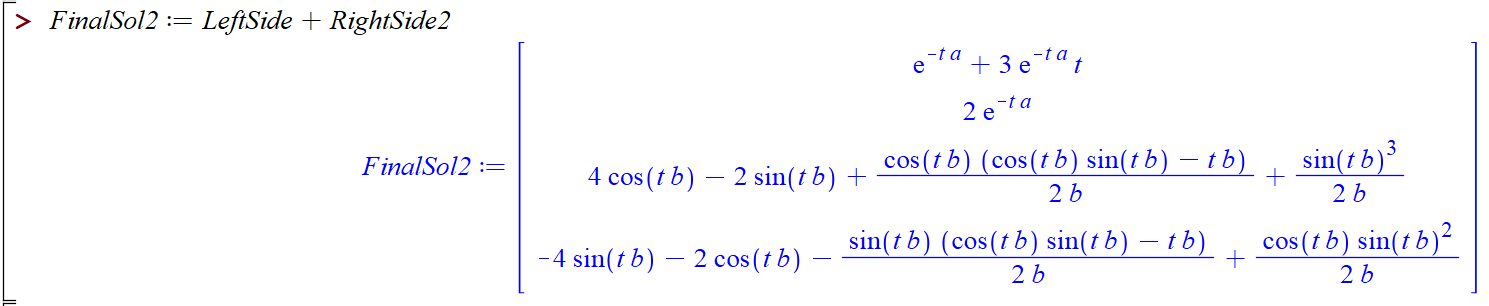


Note, below we’ll replace with and with .

The first solution, clearly has no solutions at (hence the finding the second solution set). Now, simplifying the solution and seeing when the limit causes the expression to tend to positive or negative infinity:

We know that we have: , from the problem statement. So, because both and are greater than zero, we’ll have exponential decay which will outpace any polynomial expression, making it clear that:

Now, we’ll take a look at the second solution possibility from when we have and .



Here we have a much simpler case to analyze. The only variables we have left are alpha and beta. But we notice only alpha is in the first set of equations and beta plays no role in boundedness for any of the equations. Hence, we see that:

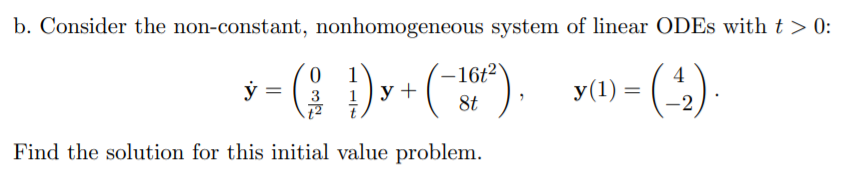
bounded

bounded

unbounded regardless due to the term

unbounded regardless due to the term

**Part b)**



First, we’ll need to find the homogenous solution to the above non-homogenous ODE. So, we let:

Therefore, we have:

Multiplying through by and substituting :

So, guessing that is a solution, and fining the derivatives for :

Plugging these into our equation:

Dividing by :

Therefore (we note again that after a single derivative is ),

Now, utilizing our initial conditions we have:

Utilizing MATLAB to avoid mistakes:

>> A = [1 1 4; 3 -1 -2]

A =

1 1 4

3 -1 -2

>> rref(A)

ans =

1.0000 0 0.5000

0 1.0000 3.5000

Therefore, we have:

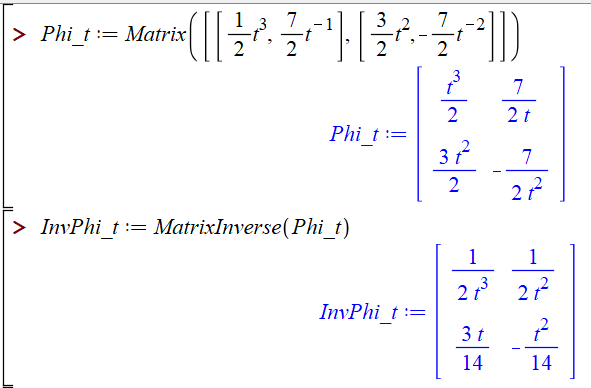
Then, we know that our linear combination of solutions becomes:

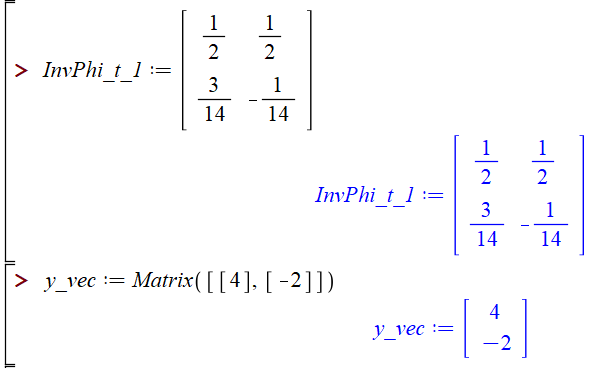
So, this then gives:

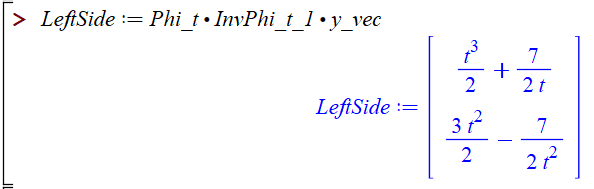
Checking then that and

We’ll now utilize this to compute our final solution using the variation of constants formula.

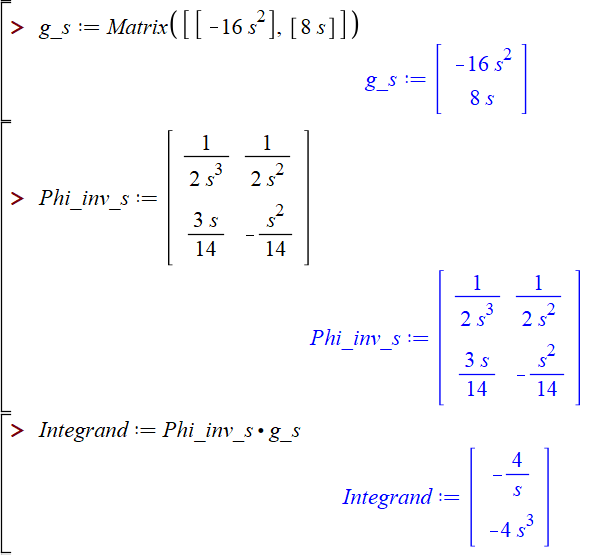
Placing the pieces into MAPLE, we find the left side of the equation:

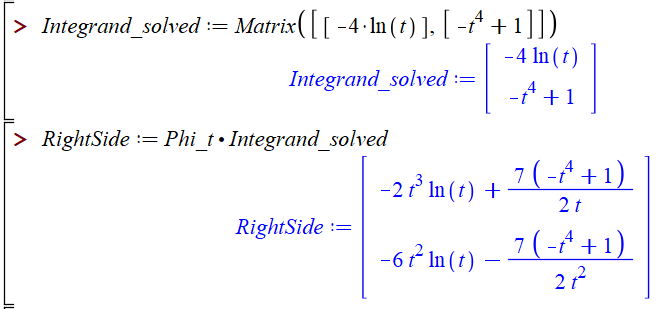






Now, to find the right side of the equation.





Lastly, we add both sides of the equation together and get our final solution:

