|  |
| --- |
| Delhi Technological University |
| Differential Equations |
| MC – 207 (Lab) |
|  |
| **Anish Sachdeva** |
| **MC/2K16/013** |



VanderPol Equation

# Code

## VanderPol Equation (µ = 1)

type vanderpoldemo

function dydt = vanderpoldemo(t, y, Mu)

%VANDERPOLDEMO Defines the van der Pol equation for ODEDEMO.

%Copyright 1984 – 2014 The Mathworks, Inc.

dydt = [y(2) ; Mu\*(1-y(1)^2)\*y(2) – y(1)];

tspan = [0, 20];

Y0 = [2; 0];

Mu = 1;

ode = @(t, y) vanderpoldemo(t, y, Mu);

[t, y] = ode45(ode, tspan, y0);

%Plot of the solution

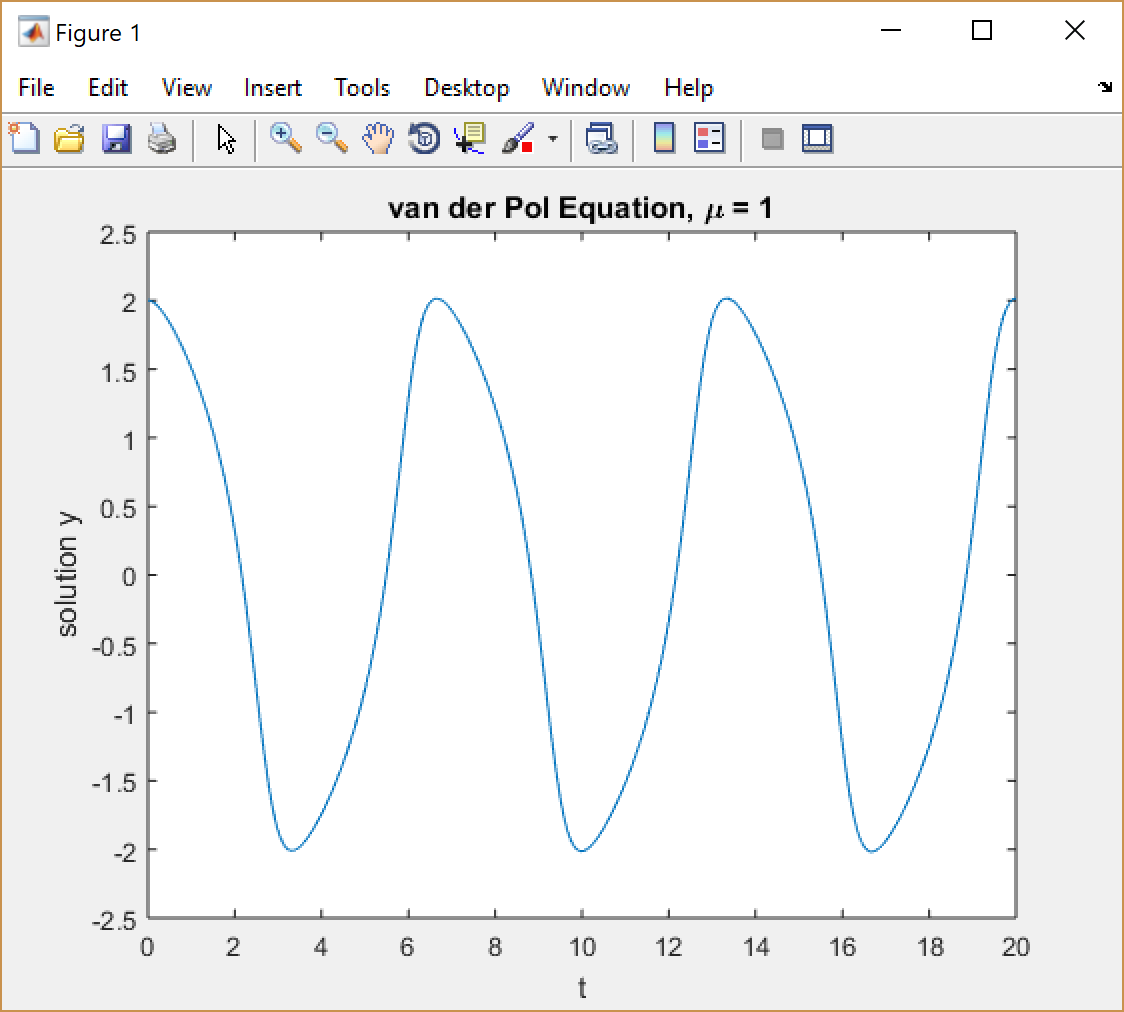
plot(t, y(;, 1))

title(‘van der Pol Equation, \mu = 1’)

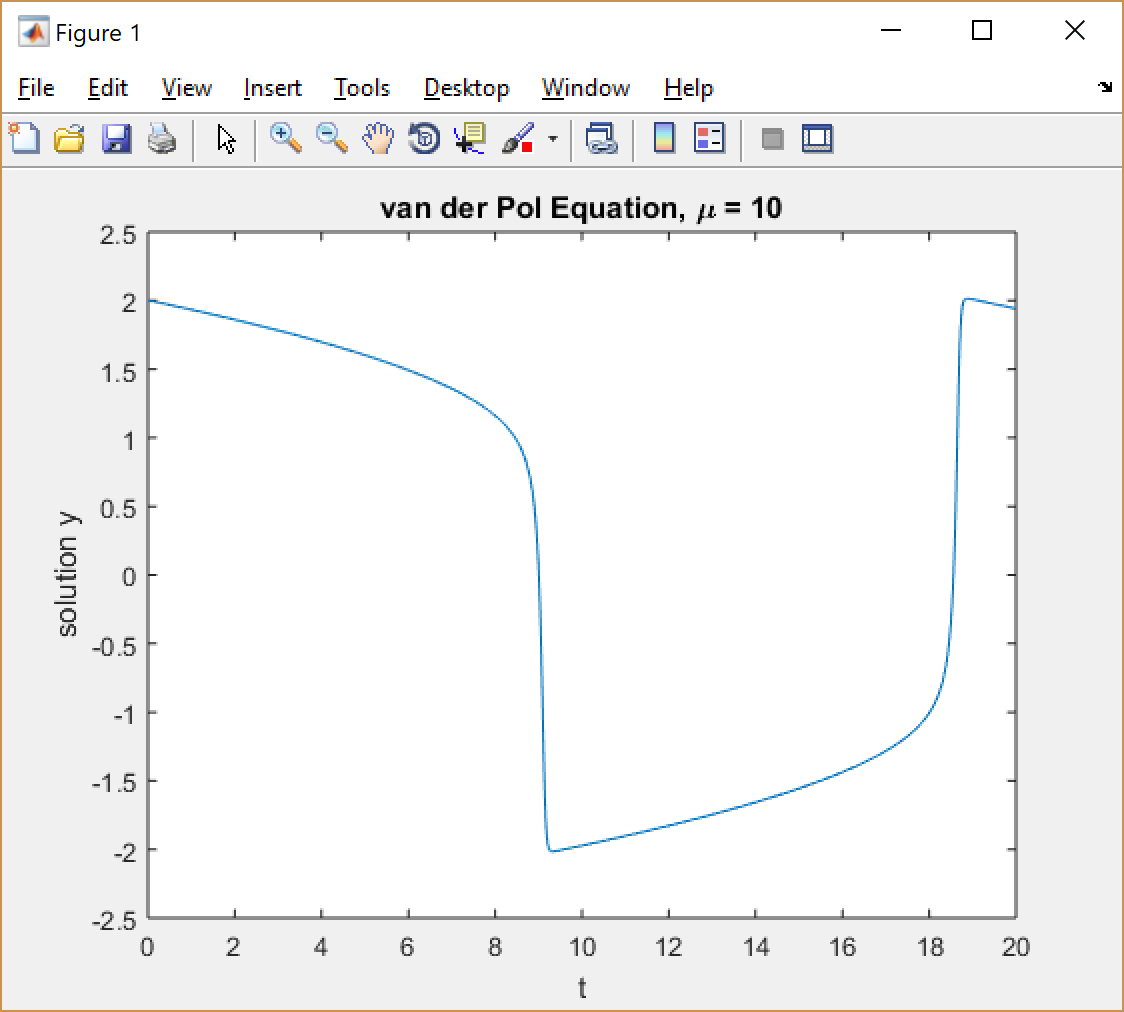
axis([0 300 -3 3])

xlabel(‘t’)

ylabel(‘solution y’)



## For µ=10



## VanderPol Equation (µ = 1000) using ODE15S

type vanderpoldemo

function dydt = vanderpoldemo(t, y, Mu)

%VANDERPOLDEMO Defines the van der Pol equation for ODEDEMO.

%Copyright 1984 – 2014 The Mathworks, Inc.

dydt = [y(2) ; Mu\*(1-y(1)^2)\*y(2) – y(1)];

tspan = [0, 3000];

Y0 = [2; 0];

Mu = 1000;

ode = @(t, y) vanderpoldemo(t, y, Mu);

[t, y] = ode15s(ode, tspan, y0);

%Plot of the solution

title(‘van der pol equation, \’)

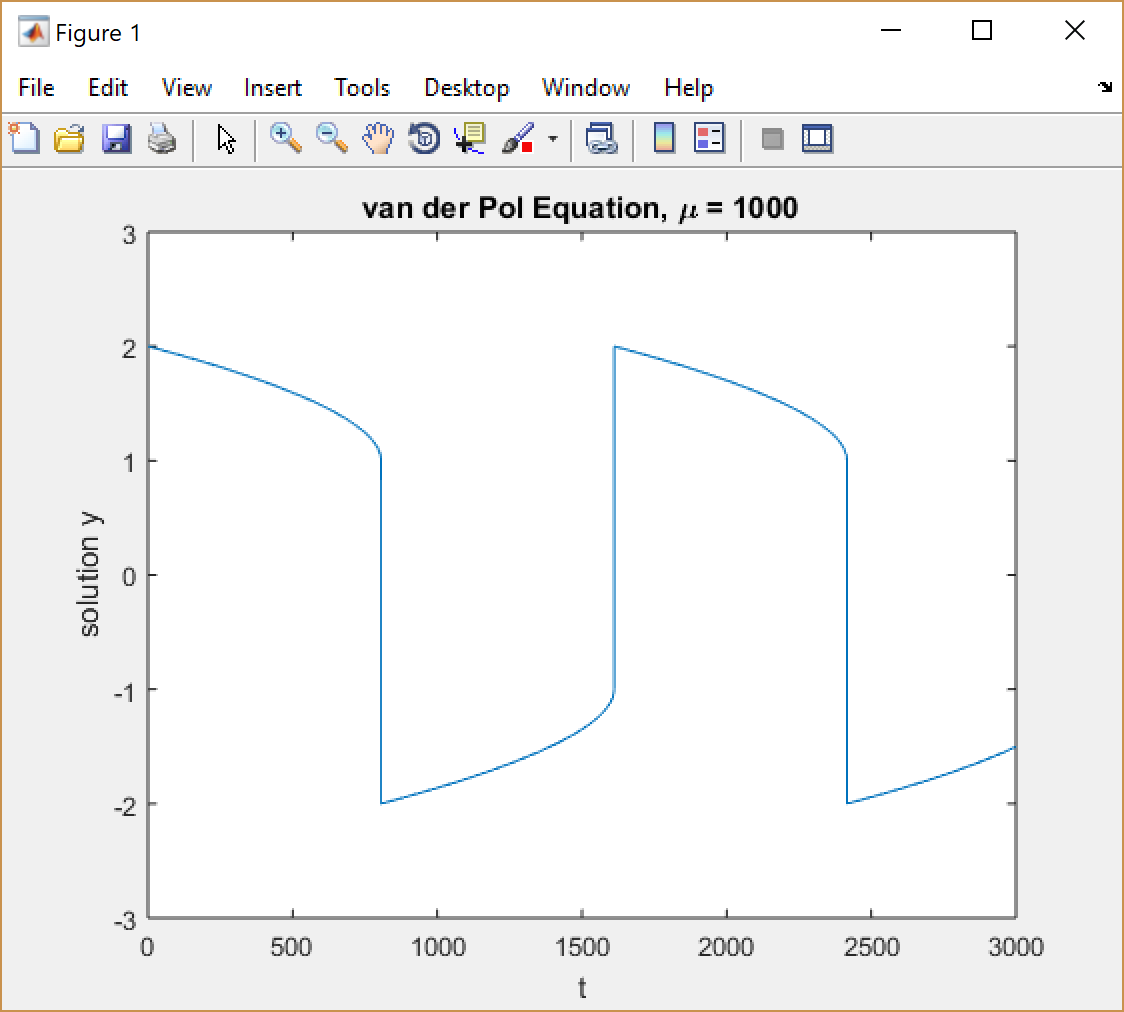
axis([0 3000 -3 3])

plot(t, y(;, 1))

xlabel(‘t’)

ylabel(‘solution y’)

title(‘van der Pol Equation, \mu = 1’)



Solution of Differential Equations with Parameters Using *dsolve*

Date: 24 August 2017

# Code

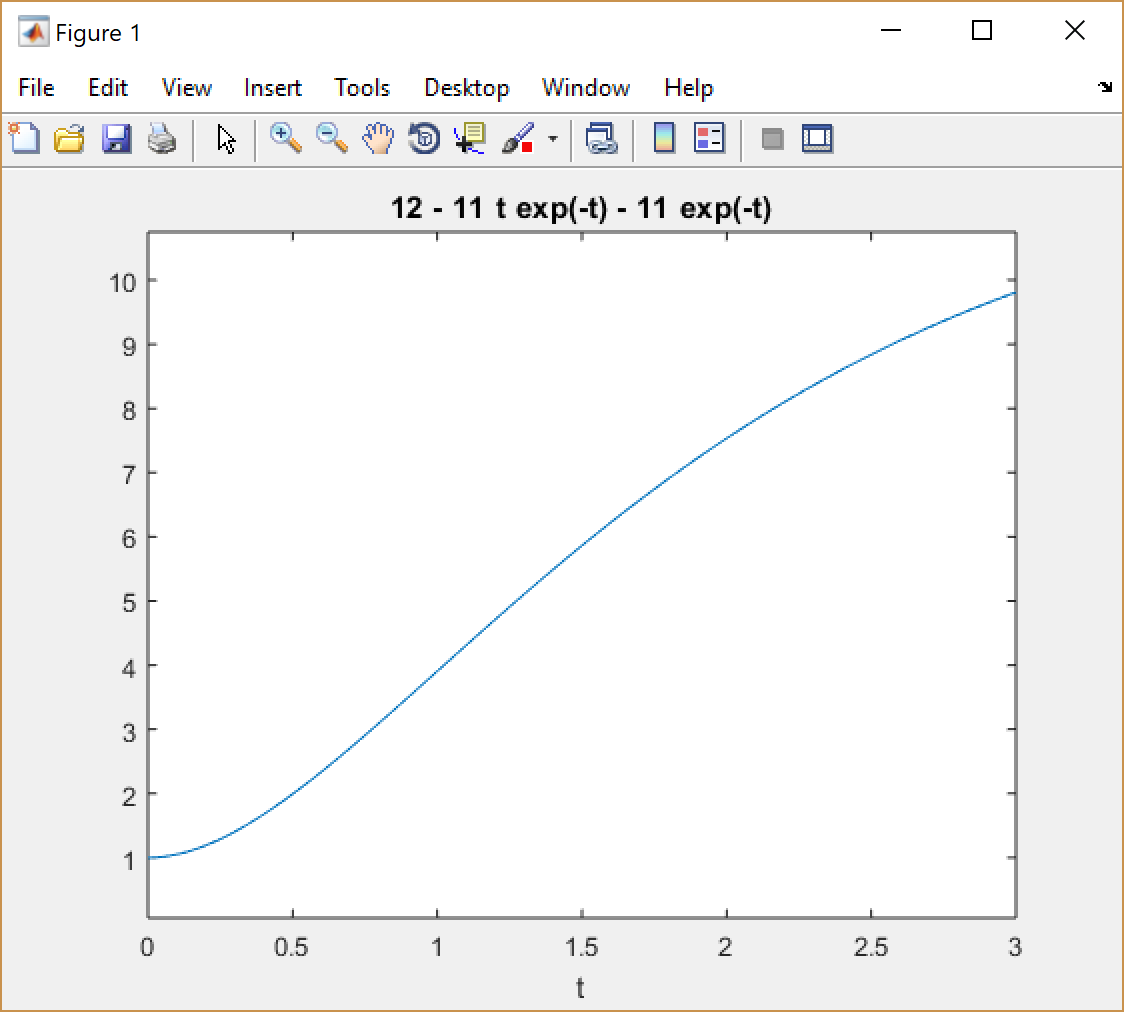
>>dsolve('D2y + 2\*Dy + y = 12', 'y(0) = 1', 'Dy(0) = 0')

ans = 12 - 11\*t\*exp(-t) - 11\*exp(-t)

>>y = ans;

>>ezplot(y, [0 3]);

# Plot



To Find the Solution of the Equation X. = AX Using the Eigenvalue-Eigen Vector Method

Date: 31st August’ 2017

# Functions Used

[V, D] = eig(A) – Used to find the eigenvalues and the eigenvectors of the matrix A. V gives a matrix with it’s eigenvectors as it’s columns, D is a diagonal matrix with eigenvalues as the diagonal elements.

# Code

%%linear\_DE\_System\_Solver : Solves a system of linear homogenous DEs

function [ sols ] = linear\_DE\_Sstem\_Solver( A )

    syms t lambda

    n = length(A);

    [V, D] = eig(A);

    eigenvalues = diag(D);

    consts = reshape(sym('c%d', 1:n), n, 1);

    unique\_eigenvalues = unique(eigenvalues);

    mults = histc(eigenvalues, unique\_eigenvalues);

    sols = sym('x%d', [1 n]);

    if length(unique\_eigenvalues) ~= length(eigenvalues)

        %For repeating eigenvalues

        i = 1; ch\_mat = A - lambda \* eye(n);

        V = vpa(v);

        while i <= n

            [pos] = find(unique\_eigenvalues == eigenvalues(i));

            if mults(pos) > 1

                e\_vector = V(:, i);

                a\_mat = subs(ch\_mat, eigenvalues(i));

                for j = 1:mults(pos)

                    V(:, i) = V(:, i) .\* (t ^ (j-1));

                    P = inv(a\_mat ^ (j-1)) \* e\_vector;

                    V(:, i) = V(:, i) + P;

                    i = i+1;

                end;

            else

                i++;

            end;

        end;

    end;

    for i = 1:n

        sols(i) = (V(i, :) .\* exp(eigenvalues' \* t)) \* conts;

    end;

end

# Output

linear\_DE\_Sstem\_Solver([1 2 ; 3 2])

ans =

[ - (2^(1/2)\*c1\*exp(-t))/2 - (2\*13^(1/2)\*c2\*exp(4\*t))/13, (2^(1/2)\*c1\*exp(-t))/2 - (3\*13^(1/2)\*c2\*exp(4\*t))/13]

To Solve an Initial Value Problem of System of Linear Homogenous Differential Equations Using the Eigenvalue-Eigenvector Method

Date: 7th August’ 2017

# Functions Used

[V, D] = eig(A) – Used to find the eigenvalues and the eigenvectors of the matrix A. V gives a matrix with it’s eigenvectors as it’s columns, D is a diagonal matrix with eigenvalues as the diagonal elements.

subs(A, old, new) – Substitutes all instances of symbolic object ‘old’ with symbolic object ‘new’ in the expression A.

# Code

function [sols, vals] = linear\_IVP\_system\_solver(A, B, C)

    syms t lambda

    n = length(A);

    [V, D] = eig(A);

    eigenValues = diag(D);

    constants = reshape(sym('c%d', 1:n), n, 1);

    unique\_EigenValues = unique(eigenValues);

    mults = histc(eigenValues, unique\_EigenValues);

    sols = sym('x%d', [1 n]);

    if(length(unique\_EigenValues) ~= length(eigenValues))

        %For representing Eigen Values

        i = 1;

        ch\_Mat = A - lambda \* eye(n);

        V = vpa(V);

        while(i <= n)

            [pos] = find(unique\_EigenValues == eigenValues(i));

            if(mults(pos) > 1)

                EigenVector = V( : , i);

                a\_Mat = subs(ch\_Mat, eigenValues(i));

                for j = 1 : mults(pos)

                    V(:, i) = V(:, i) .\* (t ^ (j-1));

                    P = inv(a\_Mat ^ (j-1)) \* EigenVector;

                    V(:, i) = V(:, i) + P;

                    i = i + 1;

                end

            else

                i = 1+1;

            end

        end

    end

    for i = 1 : n

        sols(i) = (V(i, :) .\* exp(eigenValues' \* t)) \* constants;

    end

    vals = solve(subs(sols, t, B) == C);

    constantNames = fieldnames(vals);

    %The final solution

    for i = i : n

        sols = subs(sols, constants(i), vals.(constantNames{i}));

    end

# Output

linear\_IVP\_system\_solver([1 2 ; 3 2], 0, [0 -4])

ans =

[ - (8\*exp(4\*t))/5 - (2^(1/2)\*c1\*exp(-t))/2, (2^(1/2)\*c1\*exp(-t))/2 - (12\*exp(4\*t))/5]

Strum-Liouville: Calculate the eigenValues and eigenVectors

%% for a strumLioville problem with biundaries [0,L] and EigenValue

%% lambda st. X'(L)=X'(0)=0

function[eValue, eFunction, nonZero] = strum\_liouville(L)

syms y(x)

syms lambda n

sprintf('solving for various conditions')

sprintf('lambda > 0')

assume(lambda > 0);

solution = dsolve(diff(y,2) + lambda\*y == 0);

eFunction =solution;

diff\_sol = diff(solution, x);

vals = solve(subs(diff\_sol, 0)==0, subs(diff\_sol, L)==0);

nonZero = vals;

sprintf('Non Zero values in the solution')

disp(vals);

%%Eigen values for this solution

eValue = [(n \* pi) / L] .^ 2;

sprintf('When lambda = 0')

try

solution = dsolve(subs(diff(y,2) + lambda\*y == 0), lambda, 0);

catch

sprintf('No non trivial solution');

end

sprintf('When lambda < 0')

assume(lambda < 0);

solution = dsolve(diff(y,2) + lambda \* y == 0);

diff\_sol = diff(solution, x);

%%No explicit non-trivial solutions possible

vals = solve(subs(diff\_sol,0) == 0, subs(diff\_sol, L) == 0);

Output

[val fun coeff] = strum\_liouville(pi);

ans =

solving for various conditions

ans =

lambda > 0

ans =

Non Zero values in the solution

C3: [0x1 sym]

lambda: [0x1 sym]

ans =

When lambda = 0

ans =

When lambda < 0

Warning: Cannot find explicit solution.

> In solve (line 318)

In strum\_liouville (line 35)

vals =

C6: [0x1 sym]

lambda: [0x1 sym]

function [ u v ] = lagrange\_solver( )

%lagrange solver: solves a linear partial differential eqution of the

%form Pp + Qq + R = 0 by lagrange's method

syms x y z p q dx dy c1 c2

%the equation to be solved

lhs = (y^2 \*z /x) \* p + z\*x\*q;

rhs = y^2;

C = coeffs(lhs, [q p]);

P = C(1);

Q = C(2);

R = rhs;

%seperate variables

P = P\*x / z;

Q = Q\*x / z;

%Integrating

u = int(P, y) == int(Q,x) + c1;

%consider first and last fractions

P = C(1);

P = y^2 \* z / P;

R = y^2 \* z / R;

v = int(P,x) == int(R,z) + c2;

end

output

[u v] = lagrange\_solver()

u =

y^3/3 == x^3/3 + c1

v =

x^2/2 == z^2/2 + c2

%Charpit\_solver: Solve a non-linear PDE using Charpit's method

function[ans] = charpit\_solver()

syms f(x,y,z,p,q) fx fy fz fp fq a b

f = z -p\*x-q\*y - p\*p - q\*q;

%f = z\*p\*q - p - q;

fx = diff(f,x);

fy = diff(f,y);

fz = diff(f,z);

fp = diff(f,p);

fq = diff(f,q);

if fx + p\*fz == fy + q\*fz && fy + q\*fz == 0

ans = simplify(subs(subs(f,p,a),q,b) == 0);

elseif fx + p\*fz / (fy + q\*fz) == p/q

S = solve(subs(f, p, a\*q) == 0, subs(f,q,p/a) == 0);

%seperation of variables after substituting solution struct

end;

ans =

a^2 + x\*a + b^2 + y\*b == z

%homogenous linear PDE solver: solves a homogenous linear partial

%differential equation, when f(x,y) is exponential without repeating

%factors

function[ans] = homogenous\_linear\_PDE\_solver()

syms F(D, Dp) f(x, y) v

F = D^3 - 6 \* D^2 \* Dp + 11\*D\*Dp^2 - 6\*Dp^3;

f = exp(5\*x + 6\*y);

c1 = log(subs(subs(f, y, 0), x, 1));

c2 = log(subs(subs(f, x, 0), y, 1));

ans = f / subs(subs(F, D, c1), Dp, c2);

end

homogenous\_linear\_PDE\_solver

ans =

-exp(5\*x + 6\*y)/91

%%heat\_exact: Compute the exact symbolic rpresentation of the solution of

%%the heat equation for a uniform bar with diffusivity k, initial

%%temperature distribution f and insulated boundaries 0 < x < a, with

%%initial temperature distribution along the rod given by f(x)

function[sol, eValue] = heat\_exact(k, a)

%%using seperation of variables

syms G(t) phi(x) lambda

[eValue phi coeff] = strum\_liouville(a);

phi = subs(phi, lambda, eValue);

t\_ode = diff(G, t) == -k\*lambda\*G;

t\_sol = dsolve(t\_ode);

sol = t\_sol \* phi;

end

heat\_exact(6, pi)

ans =

solving for various conditions

ans =

lambda > 0

ans =

Non Zero values in the solution

C3: [0x1 sym]

lambda: [0x1 sym]

ans =

When lambda = 0

ans =

When lambda < 0

Warning: Cannot find explicit solution.

> In solve (line 318)

In strum\_liouville (line 35)

In heat\_exact (line 9)

vals =

C6: [0x1 sym]

lambda: [0x1 sym]

ans =

C9\*exp(-6\*lambda\*t)\*(C3\*cos(x\*(n^2)^(1/2)) + C4\*sin(x\*(n^2)^(1/2)))

>>

Output

wave(4, pi)

ans =

solving for various conditions

ans =

lambda > 0

ans =

Non Zero values in the solution

lambda: [0x1 sym]

sig: [0x1 sym]

ans =

When lambda = 0

ans =

When lambda < 0

vals =

lambda: [0x1 sym]

sig: [0x1 sym]

ans =

solving for various conditions

ans =

lambda > 0

ans =

Non Zero values in the solution

lambda: [0x1 sym]

sig: [0x1 sym]

ans =

When lambda = 0

ans =

When lambda < 0

vals =

lambda: [0x1 sym]

sig: [0x1 sym]

ans =

(C3\*cos(lambda^(1/2)\*t) + C4\*sin(lambda^(1/2)\*t))\*(C3\*cos(lambda^(1/2)\*x) + C4\*sin(lambda^(1/2)\*x))

>>

Code

%%wave: Compute the exact symbolic representation of the wave equation for

%%waves along a string of path of length a, with given initial and boundary

%%conditions

function[sol, eValue] = wave(k, a)

%%using seperaton of variables

syms X(x) T(t) sig c1 c2 d1 d2 c3 c4

[eValue\_t X coeff\_X] = strum\_liouville(sig);

[eValue\_t T coeff\_t] = strum\_liouville(k\*sig);

T = subs(subs(subs(T, x, t), c3, d1), c4, d2);

sol = [T \* X];

end