DIFFERENTIAL EQUATIONS
LAB MC207

ANEESH PANCHAL
2K20/MC/021

DEPARTMENT OF APPLIED MATHEMATICS

—

Submitted to

—

Dr. CP Singh and Ms. Simran

VISION

TO BE A WORLD CLASS UNIVERSITY THROUGH EDUCATION, INNOVATION AND RESEARCH FOR THE SERVICE OF HUMANITY.

MISSION

✦ TO ESTABLISH CENTRES OF EXCELLENCE IN EMERGING AREAS OF SCIENCE, ENGINEERING, TECHNOLOGY, MANAGEMENT AND ALLIED AREAS.

✦ TO FOSTER AN ECOSYSTEM FOR INCUBATION, PRODUCT DEVELOPMENT, TRANSFER OF TECHNOLOGY AND ENTREPRENEURSHIP.

✦ TO CREATE ENVIRONMENT OF COLLABORATION, EXPERIMENTATION, IMAGINATION AND CREATIVITY.

✦ TO DEVELOP HUMAN POTENTIAL WITH ANALYTICAL ABILITIES, ETHICS AND INTEGRITY.

✦ TO PROVIDE ENVIRONMENT FRIENDLY, REASONABLE AND SUSTAINABLE SOLUTIONS FOR LOCAL AND GLOBAL NEEDS

DEPARTMENT OF APPLIED MATHEMATICS

VISION

TO EMERGE AS A CENTRE OF EXCELLENCE AND EMINENCE BY IMPARTING FUTURISTIC TECHNICAL EDUCATION WITH SOLID MATHEMATICAL BACKGROUND IN KEEPING WITH GLOBAL STANDARDS, MAKING OUR STUDENTS TECHNOLOGICALLY AND MATHEMATICALLY COMPETENT AND ETHICALLY STRONG SO THAT THEY CAN READILY CONTRIBUTE TO THE RAPID ADVANCEMENT OF SOCIETY AND MANKIND

MISSION

✦TO ACHIEVE ACADEMIC EXCELLENCE THROUGH INNOVATIVE TEACHING AND LEARNING PRACTICES.

✦ TO IMPROVE THE RESEARCH COMPETENCE TO ADDRESS SOCIAL NEEDS.

✦ TO INCULCATE A CULTURE THAT SUPPORTS AND REINFORCES ETHICAL,PROFESSIONAL BEHAVIOURS FOR A HARMONIOUS AND PROSPEROUS SOCIETY.

✦ STRIVE TO MAKE STUDENTS TO UNDERSTAND, APPRECIATE AND GAIN MATHEMATICAL SKILLS AND DEVELOP LOGIC, SO THAT THEY ARE ABLE TO CONTRIBUTE INTELLIGENTLY IN DECISION MAKING WHICH CHARACTERISES OUR SCIENTIFIC AND TECHNOLOGICAL AGE.

PROGRAMME EDUCATIONAL

OUTCOMES

✦ TO PREPARE GRADUATES WITH A SOLID FOUNDATION IN ENGINEERING, MATHEMATICAL SCIENCE AND TECHNOLOGY FOR A SUCCESSFUL CAREER IN MATHEMATICS AND COMPUTING / FINANCE / COMPUTER ENGINEERING FIELDS.

✦ TO PREPARE GRADUATES TO BECOME EFFECTIVE COLLABORATORS / INNOVATORS, WHO COULD ABLY ADDRESS TOMORROW'S SOCIAL, TECHNICAL AND ENGINEERING CHALLENGES.

✦ TO ENRICH GRADUATES WITH INTEGRITY AND ETHICAL VALUES SO THAT THEY BECOME RESPONSIBLE ENGINEERS.

PROGRAMME OUTCOMES

The POs are defined in line with the graduate attributes set by NBA.

✦ ENGINEERING KNOWLEDGE: THE GRADUATE OF MATHEMATICS & COMPUTING MUST HAVE AN ABILITY TO APPLY KNOWLEDGE OF MATHEMATICS, BASIC SCIENCE AND COMPUTER SCIENCE TO SOLVE ENGINEERING AND RELATED PROBLEMS.

✦ PROBLEM ANALYSIS: AN ABILITY TO IDENTIFY, ANALYZE AND FORMULATE COMPLEX ENGINEERING PROBLEMS TO REACH LOGICAL CONCLUSION.

✦ DESIGN/DEVELOPMENT OF SOLUTION: AN ABILITY TO DESIGN AND CONDUCT EXPERIMENTS, ANALYZE AND INTERPRET THE DATA.

✦ CONDUCT INVESTIGATIONS OF COMPLEX PROBLEMS: AN ABILITY TO USE RESEARCH BASED KNOWLEDGE AND APPLY RESEARCH METHODS TO PROVIDE VALID CONCLUSION.

✦ MODERN TOOL USAGES: AN ABILITY TO CREATE, SELECT AND IMPLEMENT APPROPRIATE TECHNIQUES, SUCH AS ARTIFICIAL INTELLIGENCE, NEURAL NETWORK TO MODEL COMPLEX COMPUTER ENGINEERING ACTIVITY.

✦ THE ENGINEER AND SOCIETY: AN ABILITY TO EXPLORE THE IMPACT OF ENGINEERING SOLUTIONS ON THE SOCIETY AND ALSO ON CONTEMPORARY ISSUES ON SOCIETAL AND ENVIRONMENTAL CONTEXT.

✦ ENVIRONMENT AND SUSTAINABILITY: AN ABILITY TO DESIGN A FEASIBLE SYSTEM, COMPONENT OR PROCESS WITHOUT VIOLATING NORMS FOR PUBLIC HEALTH AND SAFETY, CULTURAL, SOCIAL AND ENVIRONMENTAL ISSUES.

✦ ETHICS: AN ABILITY TO UNDERSTAND AND PRACTICE PROFESSIONAL AND ETHICAL RESPONSIBILITIES. 9. INDIVIDUAL AND TEAM WORKS : AN ABILITY TO FUNCTION EFFECTIVELY AS AN INTEGRAL MEMBER OR A LEADER IN A MULTIDISCIPLINARY TEAM.

✦ COMMUNICATION: AN ABILITY TO COMMUNICATE EFFECTIVELY IN BOTH ORAL AND WRITTEN FORM FOR EFFECTIVE TECHNICAL DECISION MAKING, REPORT MAKING AND PRESENTATION.

✦ PROJECT MANAGEMENT AND FINANCE: AN ABILITY TO DEMONSTRATE PRINCIPLE OF MANAGEMENT AND APPLY THEM TO SUITABLE PROJECTS.

✦ LIFELONG LEARNING: AN ABILITY TO RECOGNIZE THE NEED FOR AND TO READY FOR LIFE LONG LEARNING TO KEEP UPDATED ON TECHNOLOGICAL CHANGES.

ACKNOWLEDGEMENT

We would sincerely like to thank Prof. Dr. C.P Singh who encouraged me in carrying out the experiments and has a guiding spirit behind completion of this course.

I am very thankful to him for putting tremendous efforts from his side to assist me as much as possible.

CONTENTS

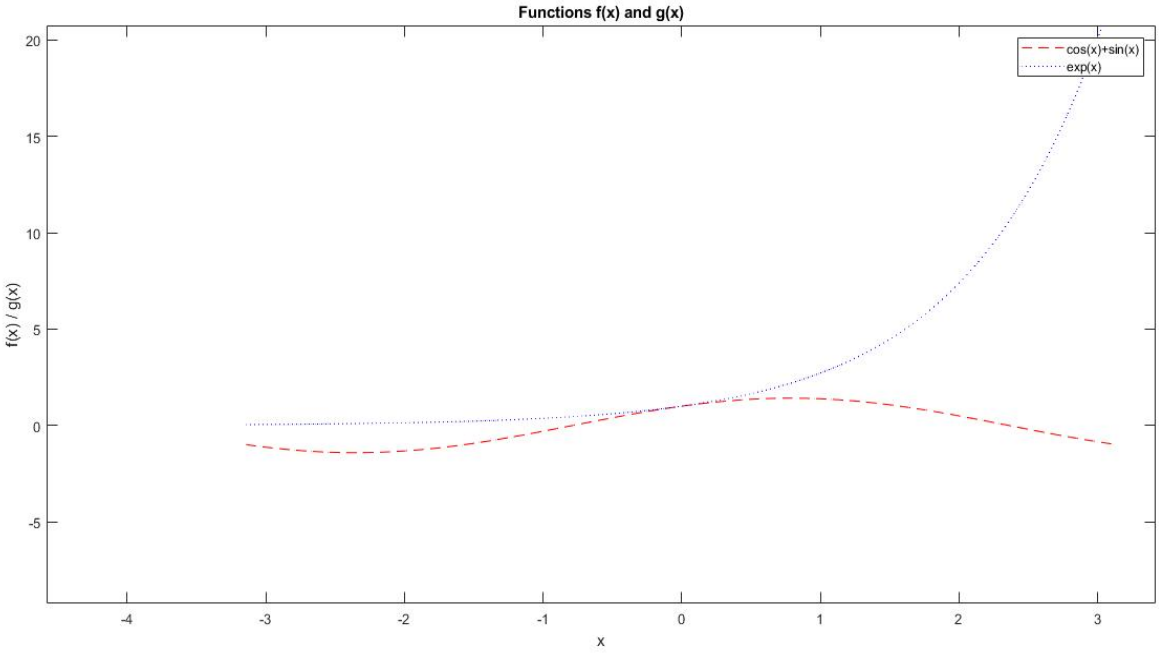
MATLAB Tutorial Lab (10/08/2021)

```
>> a = [2, 4, 6; 3, 2, 1; 7, 8, 4]

a =
2     4     6
3     2     1
7     8     4

>> det(a)
ans =
40

>> inv(a)
ans =
0.0000    0.8000   -0.2000
-0.1250   -0.8500    0.4000
0.2500    0.3000   -0.2000

>> for i=1:3
for j=1:3
A(i,j) = a(i,j)^2;
end
end
>> A

A =
4     16    36
9     4     1
49    64    16

>> rref(a)
```

```
ans =
     1     0     0
     0     1     0
     0     0     1

>> e = eig(a)
e =
11.8345
-2.4613
-1.3732

>> [V,D,W] = eig(a)
V =
    -0.5771    -0.8236     0.7427
    -0.2549     0.4898    -0.6690
    -0.7759     0.2859     0.0284

D =
    11.8345     0     0
     0    -2.4613     0
     0     0    -1.3732

W =
    -0.5632    -0.5483    -0.3097
    -0.6471    -0.5832    -0.8108
    -0.5139     0.5994     0.4967

>> for x =1:10
fprintf('Value of x: %d\n',x)
end
Value of x: 1
Value of x: 2
```

```
Value of x: 3
Value of x: 4
Value of x: 5
Value of x: 6
Value of x: 7
Value of x: 8
Value of x: 9
Value of x: 10

>> for i=1:3
for j=1:3
fprintf('Element of matrix: %d\n',a(i,j));
end
end
Element of matrix: 2
Element of matrix: 4
Element of matrix: 6
Element of matrix: 3
Element of matrix: 2
Element of matrix: 1
Element of matrix: 7
Element of matrix: 8
Element of matrix: 4

>> x = -pi:0.01:pi;
>> f = sin(x)+cos(x);
>> g = exp(x);
>> plot(x, f,'r--', x, g,'b:')
>> legend('cos(x)+sin(x)','exp(x)')
>> xlabel('x')
>> ylabel('f(x) / g(x)')
>> title('Functions f(x) and g(x)')
```

>>


Functions f(x) and g(x)
cos(x)+sin(x)
exp(x)
f(x) / g(x)
x
20
15
10
5
0
-5
-4
-3
-2
-1
0
1
2
3

PROGRAM-1 (17/08/2021)

Write a program to find the all the solutions of the equation dX = AX using Eigenvalue-Eigen vector method.

Theory:

In linear algebra, an eigenvector or characteristic vector of a linear transformation is a nonzero vector that changes by a scalar factor when that linear transformation is applied to it. The corresponding eigenvalue is the factor by which the eigenvector is scaled.

Code:

```
function[sols] = linear_DE_SYSTEM_SOLVER(A)
syms t lambda

A = [3 1;-2 1]
%A = [3 4;0 4]; %L1 ~= L2
%A = [2 0;0 2]; %L1 = L2
%A = [8 12 -2;-3 -4 1;-1 -2 2]; %L1 = L2 = L3
%A = [7 -1 6;-10 4 -12;-2 1 -1]; %L1 ~= L2 ~= L3
%A = [0 1 1;1 0 1;1 1 0]; %L1 = L2 ~= L3

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);

   %variable precision arithmetic
   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 = i+1;
        end
    end
end

for i=1:n
    sols(i) = (V(i,:).*exp(eigenvalues'*i))*consts;
end
```

Results:

```
>> linear_DE_SYSTEM_SOLVER

A =

     3     1
    -2     1


eigenvalues =

   2.0000 + 1.0000i
   2.0000 - 1.0000i


ans =

[ - c1*(2346496548713211/562949953421312 - 511437925802507i/562949953421312) -
c2*(2346496548713211/562949953421312 + 511437925802507i/562949953421312), -
c1*(326361337935945/17592186044416 + 1426225066434809i/35184372088832) -
c2*(326361337935945/17592186044416 - 1426225066434809i/35184372088832)]
```

PROGRAM-2 (24/08/2021)

Write a program to solve the initial value problem using Eigen values and Eigen vector method. dX = AX , X(B) = (C)

Theory:

In linear algebra, an eigenvector or characteristic vector of a linear transformation is a nonzero vector that changes by a scalar factor when that linear transformation is applied to it. The corresponding eigenvalue is the factor by which the eigenvector is scaled.

Code:

```
function[sols,vals]=Linear_DE_SYSTEM_SOLVERS(A,B,C)
syms t lambda

A = [3 1 -1;1 3 -1;3 3 -1];
B = [pi/2];
C = [0,1,0];

%A = [-3 -1;2 -1];
%B = [pi/2];
%C = [0,1];

n = length(A);
[V,D] = eig(A);
eigenvalues = diag(D)
const = 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);

   %variable precision arithmetic
   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 = i+1;
      end
   end
end

for i=1:n
   sols(i)=(V(i,:)*exp(eigenvalues.*t))*const(i);
end
vals=vpasolve(subs(sols,t,B)==C);
constnames=fieldnames(vals);

% The final solution is
for i=1:n
   sols=subs(sols,const(i),vals.(constnames{i}));
end
```

Results:

```
>> Linear_DE_SYSTEM_SOLVERS

eigenvalues =

   2.0000
   1.0000
   2.0000


ans =

[ 0, 0.079013825539440955959039219639*exp(2*t) +
0.015908134992895448728021472198971*11^(1/2)*exp(t) -
0.012499248922989281143445442442049*14^(1/2)*exp(2*t), 0]
```

PROGRAM-3 (31/08/2021)

Write a program to find the eigenvalues and eigenvectors of the Sturm-Liouville problem: d^2 X + λX = 0, dX(0) = 0, dX(L) = 0

Theory:

Each such equation together with its boundary conditions constitutes a Sturm-Liouville (S-L) problem. The value of λ is not specified in the equation: finding the λ for which there exists a non-trivial solution is part of the given S-L problem.

Code:

```
function [e_value, e_function, non_zero] = sturm_liouville(L)
syms y(x)
syms lambda n

%Equation is, y'' + (lambda)y = 0

L = 1;
%L = pi/4;
%L = pi/2;

%sprintf stores the string in buffer
sprintf('Solving for various conditions...')
sprintf('When lambda > 0')
assume(lambda > 0);

%dsolve computes symbolic solutions to ordinary differential equations
%sol = diff(X,n) applies diff recursively n times, resulting in the nth difference
solution = dsolve(diff(y, 2) + lambda * y == 0);
e_function = solution
diff_sol = diff(solution, x);

%subs(s) returns a copy of s , replacing symbolic variables in s , with their values obtained from the calling function
vals = solve(subs(diff_sol, x,0) == 0, subs(diff_sol, x,L) == 0);
non_zero = vals;
sprintf('Non-zero values in the solution')
vals;

%Eigen Value for this solution
e_value = [(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 solution exists
vals = solve(subs(diff_sol, 0) == 0, subs(diff_sol, L) == 0);
```

Results:

```
>> sturm_liouville

ans =

   'Solving for various conditions...'


ans =

   'When lambda > 0'


e_function =

C1*cos(lambda^(1/2)*x) - C2*sin(lambda^(1/2)*x)


ans =

   'Non-zero values in the solution'


ans =

   'When lambda = 0'


ans =

   'No non-trivial solution'
```

```
ans =

    'When lambda < 0'

ans =

n^2*pi^2
```

PROGRAM-4 (14/09/2021 – 21/09/2021)

Write a program to solve Lagrange's equation using Lagrange's method.

Theory:

In mathematical optimization, the method of Lagrange multipliers is a strategy for finding the local maxima and minima of a function subject to equality constraints (i.e., subject to the condition that one or more equations have to be satisfied exactly by the chosen values of the variables).

Code (14/09/2021):

```
function [] = Lagrange() %Function Name
syms x y z p q dx dy c1 c2

lhs=((y^2)*z/x)*p+(x*z)*q; %LHS Side
rhs=(y^2); %RHS Side

C=coeffs(lhs,[q p]); %Coefficients of equation
P=C(1); %Extracting value of P
Q=C(2); %Extracting value of Q
R=rhs; %Extracting value of R

%int() is an integrating function
%1st and 2nd expression
P=P*x/z;
Q=Q*x/z;
U = int(P,y) == int(Q,x) + c1; %Solving for U
S = int(P,y) - int(Q,x);
U = simplify(S)

%1st and last expression
P=y^2*z/P;
R=y^2*z/R;
V = int(P,x) == int(R,z) + c2; %Solving for V
T = int(P,x) - int(R,z);
V = simplify(T)

disp("Solution is given as f(U,V): ")
disp('f(')
disp(U)
disp(', ')
disp(V)
disp(') == 0')
```

Results:

```
>> Lagrange
```

```
U =

y^3/3 - x^3/3


V =

(z*(2*x - z))/2

Solution is given as f(U,V):
f(
y^3/3 - x^3/3

,
(z*(2*x - z))/2

) == 0

>>
```

Code (21/09/2021):

```
function [] = LagrangeEqn() %Function Name
syms x y z p q dx dy c1 c2 sol deno num sol

lhs = (x^2-y^2-z^2)*p + (2*x*y)*q; %LHS Side
rhs = 2*x*z; %RHS Side

C = coeffs(lhs,[q p]); %Coefficients of equation
P=C(1); %Extracting value of P
Q=C(2); %Extracting value of Q
R=rhs; %Extracting value of R

%int() is an integrating function
%2nd and 3rd expression
Q=2*x/Q;
R=2*x/R;
V = int(Q,y) == int(R,z) + c2; %Solving for V
T = exp(int(Q,y) - int(R,z));
V = simplify(T)

%3rd and last expression
R=1/R;
Q=1/Q;
P=P/2;
deno = 2*(P + y*Q + z*R);
num = 2/3*(int(x/deno,x) + int(y/deno,y) +int(z/deno,z));
U = num == int(1/R,z) +c1;
sol = exp(num - int(1/R,z));
```

```
U = simplify(sol)
disp("Solution is given as f(U,V)=0: ")
disp('f(')
disp(U)
disp(', ')
disp(V)
disp(') == 0')
```

Results:

```
>> LagrangeEqn

V =

y/z


U =

(x^2 + y^2 + z^2)/z

Solution is given as f(U,V)=0:
f(

(x^2 + y^2 + z^2)/z


,
y/z

) == 0
```

PROGRAM-5 (28/09/2021)

Write a program to solve non-linear PDE using Charpit's Method: f(x,y,z,p,q) = 0

Theory:

Charpit's Method. This is general method for solving partial differential equations (PDE) with two independent variables. This method is applied to solve those partial differential equations which cannot be solved by already known methods. The basic idea in Charpit's method is the introduction of another partial differential equation of order one of the form

$$g(x, y, z, p, q) = 0 \quad \ldots(2)$$

involving the two independent variables x, y and the dependent variable z, along with p and q. After introducing (2), we solve equations (1) and (2) for p and q and then substitute these values in the equation

Code I:

```
%Function Name
function [] = CharpitDE()

% f = 2q(z-px- qy) - (1 + q^2)
% Declaring syms type varibles
syms x y z p q dx dy c1 c2 a b

lhs = 2*q*(z-(p*x)-(q*y));
rhs = 1+(q*q);

% Given equation
f = lhs - rhs

% Finding partial derivatives
fx = diff(f,x);
fy = diff(f,y);
fz = diff(f,z);
fp = diff(f,p);
fq = diff(f,q);

% denominator of dp, dq, dz
f1 = fx + p* fz
f2 = fy + q * fz
f3 = simplify(-p*fp-q*fq)

% Solving for given condition
if f2==0 %condtion for q=constant
   Q = a
   eqn = subs(f, q, Q);
   eq2 = solve(eqn, p);
   P = eq2
```

```
elseif f1==0 %condtion for p=constant
   P = a
   eqn = subs(f, p, P);
   eq2 = solve(eqn, q);
   Q = eq2
end

% Solution after Integration
sol=int(Q,y)+int(P,x);

disp("Solution is given as f(x,y)= ")
disp(sol)
```

Results:

```
>> CharpitDE

f =

- 2*q*(p*x - z + q*y) - q^2 - 1


f1 =

0


f2 =

0


f3 =

2*q*(q - z + 2*p*x + 2*q*y)


Q =

a


P =

-(a^2 - 2*a*(z - a*y) + 1)/(2*a*x)

Solution is given as f(x,y)=
a*y - (log(x)*(2*a^2*y - 2*a*z + a^2 + 1))/(2*a)


>>
```

Code II:

```
%Function Name
function [] = CharpitDiffEqn()

% f = ((p*p)+(q*q))*y - q*z
% Declaring syms type varibles
syms x y z p q dx dy c1 c2 a b c

lhs = ((p*p)+(q*q))*y;
rhs = q*z;

% Given equation
f = lhs - rhs

% Finding partial derivatives
fx = diff(f,x);
fy = diff(f,y);
fz = diff(f,z);
fp = diff(f,p);
fq = diff(f,q);

% denominator of dp, dq, dz
f1 = fx + p* fz
f2 = fy + q * fz
f3 = simplify(-p*fp-q*fq)

% Solving for given condition
if f2==0 %condtion for q=constant
   Q = a
   eqn = subs(f, q, Q);
   eq2 = solve(eqn, p);
   P = eq2

elseif f1==0 %condtion for p=constant
   P = a
   eqn = subs(f, p, P);
   eq2 = solve(eqn, q);
   Q = eq2

else
   int1 = 2*f1/p;
   int2 = 2*f2/p;
   intsol = int(int2,p)-int(int1,q)-c*c;
   P1 = solve(intsol,p);
   eqn = subs(f,p,P1);
   eqn2 = solve(eqn,q);
   Q = eqn2
   eqn3 = subs(intsol,q,Q);
   eqn4 = solve(eqn3,p);
   P = eqn4

end
```

```
% Solution after Integration
sol = int(Q,y)+int(P,x);

disp("Solution is given as f(x,y)= ")
disp(sol)
```

Results:

```
>> CharpitDiffEqn

f =

y*(p^2 + q^2) - q*z


f1 =

-p*q


f2 =

p^2


f3 =

- 2*y*p^2 + q*(z - 2*q*y)


Q =

(c^2*y)/z


P =

  (c*(z + c*y)^(1/2)*(z - c*y)^(1/2))/z
 -(c*(z + c*y)^(1/2)*(z - c*y)^(1/2))/z

Solution is given as f(x,y)=
 (c^2*y^2)/(2*z) + (c*x*(z + c*y)^(1/2)*(z - c*y)^(1/2))/z
 (c^2*y^2)/(2*z) - (c*x*(z + c*y)^(1/2)*(z - c*y)^(1/2))/z
```

PROGRAM-6 (12/10/2021 – 10/11/2021)

Write a program to solve (aD^2 + bDD’ + cD’^2)z = f(x,y)

Theory:

A linear partial differential equation with constant coefficients is known as non homogeneous linear partial differential equation with constant coefficients if the orders of all the partial derivatives involved in the equation are not equal. Here we are dealing with only homogeneous PDE with constant coefficients.

Code (12/10/2021):

```
%Function Name
function [] = HomoPDE()
syms D d x y z m f1 f2 f3

%D is wrt x and d is wrt y
%We take f1, f2 and f3 inplace of phi1, phi2 and phi3

%F is LHS and f is RHS
F = (D*D*D - 6*D*D*d + 11*D*d*d - 6*d*d*d)*z;
f = exp(5*x + 6*y);

%Substitute D with m and d with 1
eqn1 = subs(F, D, m);
eqn2 = subs(eqn1, d, 1)

%Values of m as a 3*1 matrix
mval = solve(eqn2, m)

%Find CF
if mval(1)~=mval(2)~=mval(3)
   CF = f1*(y+mval(1)*x) + f2*(y+mval(2)*x) + f3*(y+mval(3)*x)
elseif mval(1) == mval(2) ~= mval(3)
   CF = f1*(y+mval(1)*x) + x*f2*(y+mval(2)*x) + f3*(y+mval(3)*x)
elseif mval(1) ~= mval(2) == mval(3)
   CF = f1*(y+mval(1)*x) + x*f2*(y+mval(2)*x) + f3*(y+mval(3)*x)
elseif mval(1) == mval(3) ~= mval(2)
   CF = x*f1*(y+mval(1)*x) + f2*(y+mval(2)*x) + f3*(y+mval(3)*x)
else
   CF = f1*(y+mval(1)*x) + x*f2*(y+mval(2)*x) + x*x*f3*(y+mval(3)*x)
end

%Find equation of variables D and d
F1 = F/z;
F2 = f;

%Find values of constants of x and y
fpi = log(f);
valx = diff(fpi,x)
valy = diff(fpi,y)
```

```
%Substitute values of x and y in F1
while true
   eqn3 = subs(F1,D,valx);
   eqn4 = subs(eqn3,d,valy);
   if eqn4~=0
      break
   else
      F1 = diff(F1,D);
      F2 = F2*x;
   end
end

%Find PI
PI = F2/eqn4

%Final answer
final_ans = CF+PI
```

Results:

```
>> HomoPDE

eqn2 =

z*(m^3 - 6*m^2 + 11*m - 6)


mval =

 1
 2
 3


CF =

f1*(x + y) + f2*(2*x + y) + f3*(3*x + y)


valx =

5


valy =

6
```

```
PI =

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


final_ans =

f1*(x + y) - exp(5*x + 6*y)/91 + f2*(2*x + y) + f3*(3*x + y)

>>
```

Code (26/10/2021):
Code I:

```
%Function Name
function [] = HomoPDE_cos()
syms D d f1 f2 f3 y x z m c

%D is wrt x and d is wrt y
%We take f1 and f2 inplace of phi1 and phi2

%F is LHS and f is RHS
F = (D*D - 3*D*d + 2*d*d)*z;
f = 2*cos(x+3*y);

%Substitute D with m and d with 1
eqn = subs(subs(F, D, m), d, 1);

%Values of m as a 3*1 or 2*1 matrix
mval = solve(eqn, m)

%Find CF
if mval(1)~=mval(2)
   CF = f1*(y+mval(1)*x) + f2*(y+mval(2)*x)
else
   CF = f1*(y+mval(1)*x) + x*f2*(y+mval(2)*x)
end

%Find coeff of x and y
fn = f/2
eqnx = acos(eval(subs(subs(fn,y,0),x,1)))
eqny = acos(eval(subs(subs(fn,x,0),y,1)))

%Find PI
PIF = F/z;
PI_deno = subs(subs(subs(PIF,D*D,(-1)*eqnx*eqnx) ,D*d,(-1)*eqnx*eqny) ,d*d,(-1)*eqny*eqny)
PI = f/PI_deno

%Final answer
final_ans = CF + PI
```

Results:

```
>> HomoPDE_cos

mval =

 1
 2


CF =

f1*(x + y) + f2*(2*x + y)


fn =

cos(x + 3*y)


eqnx =

    1


eqny =

   3.0000


PI_deno =

-10


PI =

-cos(x + 3*y)/5


final_ans =

f1*(x + y) - cos(x + 3*y)/5 + f2*(2*x + y)

>>
```

Code II:

```
%Function Name
function [] = HomoPDE_sin()
syms D d f1 f2 f3 y x z m c
```

```
%D is wrt x and d is wrt y
%We take f1 and f2 inplace of phi1 and phi2

%F is LHS and f is RHS
F = (D*D - 5*D*d + 4*d*d)*z;
f = sin(2*x+3*y);

%Substitute D with m and d with 1
eqn = subs(subs(F, D, m), d, 1);

%Values of m as a 3*1 or 2*1 matrix
mval = solve(eqn, m)

%Find CF
if mval(1)~=mval(2)
   CF = f1*(y+mval(1)*x) + f2*(y+mval(2)*x)
else
   CF = f1*(y+mval(1)*x) + x*f2*(y+mval(2)*x)
end

%Find coeff of x and y
%sin(x) = sin(pi - x)
eqnx = pi - asin(simplify(subs(subs(f,y,0),x,1)))
eqny = pi - asin(simplify(subs(subs(f,x,0),y,1)))

%Find PI
PIF = F/z;
PI_deno = subs(subs(subs(PIF,D*D,(-1)*eqnx*eqnx) ,D*d,(-1)*eqnx*eqny) ,d*d,(-1)*eqny*eqny)
PI = f/PI_deno

%Final answer
final_ans = CF + PI
```

Results:

```
>> HomoPDE_sin

mval =

 1
 4


CF =

f1*(x + y) + f2*(4*x + y)


eqnx =

2
```

```
eqny =

3


PI_deno =

-10


PI =

-sin(2*x + 3*y)/10


final_ans =

f1*(x + y) - sin(2*x + 3*y)/10 + f2*(4*x + y)

>>
```

Code (10/11/2021):

```
%Function Name
function [] = HomoPDEgen_sol()

%t is any random variable
syms D d f1(t) f2(t) f3(t) y x z m c

%D is wrt x and d is wrt y
%We take f1, f2 and f3 inplace of phi_1, phi_2 and phi_3

%F is LHS and f is RHS
F = (D*D*D + 2*D*D*d - D*d*d- 2*d*d*d)*z;
f = (y+2)*exp(x);

%Making the constant of D to 1
Fconst = subs(subs(subs(F,D,1),z,1),d,0);
F = F/Fconst;
f = f/Fconst;

%Substitute D with m and d with 1
eqn = subs(subs(F, D, m), d, 1);

%Values of m as a 3*1 or 2*1 matrix
mval = solve(eqn, m)
n = numel(mval);

%Find CF
if n==2
```

```
    if mval(1)~=mval(2)
        CF = f1(y+mval(1)*x) + f2(y+mval(2)*x)
    else
        CF = f1(y+mval(1)*x) + x*f2(y+mval(2)*x)
    end

elseif n==3
    if mval(1)~=mval(2)~=mval(3)
        CF = f1(y+mval(1)*x) + f2(y+mval(2)*x) + f3(y+mval(3)*x)
    elseif mval(1) == mval(2) ~= mval(3)
        CF = f1(y+mval(1)*x) + x*f2(y+mval(2)*x) + f3(y+mval(3)*x)
    elseif mval(1) ~= mval(2) == mval(3)
        CF = f1(y+mval(1)*x) + x*f2(y+mval(2)*x) + f3(y+mval(3)*x)
    elseif mval(1) == mval(3) ~= mval(2)
        CF = x*f1(y+mval(1)*x) + f2(y+mval(2)*x) + f3(y+mval(3)*x)
    else
        CF = f1(y+mval(1)*x) + x*f2(y+mval(2)*x) + x*x*f3(y+mval(3)*x)
    end
end

%Find PI
PIn = f;
while n>0
    PIn = subs(PIn,y,c-mval(n)*x);
    PIn = int(PIn,x);
    Value_m = mval(n);
    PIn = subs(PIn,c,y+mval(n)*x);
    n = n-1;
end

%Final Particular Integral
PI = simplify(PIn)

%Final answer
final_ans = CF + PI
```

Results:

```
>> HomoPDEgen_sol

mval =

 -2
 -1
  1


CF =

f1(y - 2*x) + f2(y - x) + f3(x + y)
```

PI =

y*exp(x)

final_ans =

f1(y - 2*x) + f2(y - x) + f3(x + y) + y*exp(x)

PROGRAM-7 (16/11/2021)

Write a program to solve heat equation subject to boundary conditions and the initial conditions.

Theory:

Let u(x,t) denote the temperature at point x at time t. The equation governing this setup is the so-called one-dimensional heat equation: u_t(x,t) = c^2*u_xx(x,t), where k>0 is a constant (the thermal conductivity of the material).

Boundary Conditions:
u(0,t) = u(a,t) = 0 for all t

Initial Condition:
u(x,0) = f(x)
0<x<a , t>0

Code (16/11/2021):

sturm_liouville.m

```
function [e_value, e_function] = sturm_liouville(L)
syms y(x)
syms lambda n

%Equation is, y'' + (lambda)y = 0
L = 1;
%L = pi/4;

%sprintf stores the string in buffer
sprintf('Solving for various conditions...')
sprintf('When lambda > 0')
assume(lambda > 0);

%dsolve computes symbolic solutions to ordinary differential equations
%sol = diff(X,n) applies diff recursively n times, resulting in the nth difference
solution = dsolve(diff(y, 2) + lambda * y == 0);
e_function = solution
diff_sol = diff(solution, x);

%subs(s) returns a copy of s , replacing symbolic variables in s , with their values obtained from the calling function
vals = solve(subs(diff_sol, x,0) == 0, subs(diff_sol, x,L) == 0);
non_zero = vals;
sprintf('Non-zero values in the solution')
vals;

%Eigen Value for this solution
e_value = [(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 solution exists
vals = solve(subs(diff_sol, 0) == 0, subs(diff_sol, L) == 0);
```

Heat_eqn.m

```
syms  x T B Bn C C1 C2 Ux lambda

%INCLUDE FUNCTION
%e_value is the p term which is equal to root(lambda)
[e_value, e_function] = sturm_liouville()
e_value = sqrt(e_value);

%SOLUTION
sol1 = subs(e_function,lambda,e_value)

%we can also use dsolve for this
sol = int(1/T,T)==int(-e_value*e_value*k,t);
sol2 = C*solve(sol,T)
sol1 = subs(sol1,C1,0)

Ux = sol1*sol2;
Ux = subs(Ux,-C*C2,Bn)
%Bn is half range fourier sine series
```

Results:

```
>> Heat_eqn

ans =

   'Solving for various conditions...'


ans =

   'When lambda > 0'


e_function =
```

```
C1*cos(lambda^(1/2)*x) - C2*sin(lambda^(1/2)*x)


ans =

    'Non-zero values in the solution'


ans =

    'When lambda = 0'


ans =

    'No non-trivial solution'


ans =

    'When lambda < 0'

Warning: Unable to find explicit solution. For options, see help.
> In sym/solve (line 317)
  In sturm_liouville (line 44)
  In Heat_eqn (line 5)

e_value =

n^2*pi^2


e_function =

C1*cos(lambda^(1/2)*x) - C2*sin(lambda^(1/2)*x)



sol1 =

C1*cos(x*(pi*(n^2)^(1/2))^(1/2)) - C2*sin(x*(pi*(n^2)^(1/2))^(1/2))

Warning: Solutions are only valid under certain conditions. To include parameters
and conditions in the solution, specify the 'ReturnConditions' value as 'true'.
> In sym/solve>warnIfParams (line 478)
  In sym/solve (line 357)
  In Heat_eqn (line 13)


sol2 =
```

```
C*exp(-k*n^2*t*pi^2)


sol1 =

-C2*sin(x*(pi*(n^2)^(1/2))^(1/2))


Ux =

Bn*exp(-k*n^2*t*pi^2)*sin(x*(pi*(n^2)^(1/2))^(1/2))

>>
```

PROGRAM-8 (16/11/2021)

Write a program to solve wave equation subject to boundary and the initial conditions.

Theory:

The wave equation is an important second-order linear partial differential equation for the description of waves—as they occur in classical physics—such as mechanical waves (e.g. water waves, sound waves and seismic waves) or light waves. It arises in fields like acoustics, electromagnetics, and fluid dynamics.

Wave Equation:
u_tt(x,t) = c^2*u_xx(x,t)

Boundary Conditions:
u(0,t) = u(a,t) = 0

Initial Condition:
u(x,0) = f(x) and u_t(x,0) = g(x)
0<x<a , t>0

Code (16/11/2021):

sturm_liouville.m

```
function [e_value, e_function] = sturm_liouville(L)
syms y(x)
syms lambda n

%Equation is, y'' + (lambda)y = 0

L = 1;
%L = pi/4;
%L = pi/2;

%sprintf stores the string in buffer
sprintf('Solving for various conditions...')
sprintf('When lambda > 0')
assume(lambda > 0);

%dsolve computes symbolic solutions to ordinary differential equations
%sol = diff(X,n) applies diff recursively n times, resulting in the nth difference
solution = dsolve(diff(y, 2) + lambda * y == 0);
e_function = solution
diff_sol = diff(solution, x);

%subs(s) returns a copy of s , replacing symbolic variables in s , with their values obtained from the calling function
vals = solve(subs(diff_sol, x,0) == 0, subs(diff_sol, x,L) == 0);
non_zero = vals;
```

```
sprintf('Non-zero values in the solution')
vals;

%Eigen Value for this solution
e_value = [(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 solution exists
vals = solve(subs(diff_sol, 0) == 0, subs(diff_sol, L) == 0);
```

Wave_eqn.m

```
syms  u(x,t) x k c t f(x) g(x) X(x) T(t) A B Bn C D C1 C2 lambda

%INCLUDE FUNCTION FOR DISTANCE
[e_value, e_function] = sturm_liouville();
e_value = sqrt(e_value);

%SOLUTION FOR X
sol_X = simplify(subs(e_function,lambda^(1/2),k));

%INCLUDE FUNCTION FOR TIME
[e_value2, e_function2] = sturm_liouville();
e_value2 = sqrt(e_value2);

%SOLUTION FOR T
sol_T = subs(subs(e_function,x,t),lambda^(1/2),k*c);

X(x) = subs(subs(sol_X,C1,A),-C2,B)
T(t) = subs(subs(sol_T,C1,C),-C2,D)

%Applying Boundary condition and Initial condition
fprintf('\n\nAfter applying Boundary and Initial Conditions:\n')

%Bn is half range fourier sine series
X(x) = subs(subs(X(x),A,0),B,Bn)

%f(x) and g(x) are half range fourier sine series
T(t) = subs(subs(T(t),C,f(x)),D,g(x))
```

```
fprintf('\nAs we have assumed initially: u(x,t) = X(x)*T(t):\n')
u(x,t) = X(x)*T(t)
```

Results:

```
>> Wave_eqn

ans =

   'Solving for various conditions...'


ans =

   'When lambda > 0'


e_function =

C1*cos(lambda^(1/2)*x) - C2*sin(lambda^(1/2)*x)


ans =

   'Non-zero values in the solution'


ans =

   'When lambda = 0'


ans =

   'No non-trivial solution'


ans =

   'When lambda < 0'

Warning: Unable to find explicit solution. For options, see help.
> In sym/solve (line 317)
  In sturm_liouville (line 44)
  In Wave_eqn (line 4)

ans =

   'Solving for various conditions...'
```

```
ans =

    'When lambda > 0'


e_function =

C1*cos(lambda^(1/2)*x) - C2*sin(lambda^(1/2)*x)


ans =

    'Non-zero values in the solution'


ans =

    'When lambda = 0'


ans =

    'No non-trivial solution'


ans =

    'When lambda < 0'

Warning: Unable to find explicit solution. For options, see help.
> In sym/solve (line 317)
  In sturm_liouville (line 44)
  In Wave_eqn (line 11)


X(x) =

A*cos(k*x) + B*sin(k*x)


T(t) =

C*cos(c*k*t) + D*sin(c*k*t)



After applying Boundary and Initial Conditions:

X(x) =

Bn*sin(k*x)
```

T(t) =

cos(c*k*t)*f(x) + sin(c*k*t)*g(x)

As we have assumed initially: u(x,t) = X(x)*T(t):

u(x, t) =

Bn*sin(k*x)*(cos(c*k*t)*f(x) + sin(c*k*t)*g(x))

>>