Engineering Analysis And Design(DE)

MC-207

Lab File

Submitted To:

L. N. Das
Department of Applied Mathematics
Delhi Technological University

Submitted By:

Devesh Singh
2k17/MC/039
Group : P2

<u>INDEX</u>

QUESTION 1

Write a program to find the solutions of the equation $\dot{X} = AX$ using the Eigenvalue-Eigen vector method

Functions used

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

Code

```
%% linear_DE_system_solver: Solve a system of linear homogenoous DEs

function [sols] = linear_DE_system_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 = i + 1;
end;         end;    end;
for i = 1:n
     sols(i) = (V(i, :) .* exp(eigenvalues' * t)) * consts;     end;
```

OUTPUT:

```
>> linear_DE_system_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]
```

QUESTION 2

Write a program to solve an Initial value problem of a system of linear homogenous differential equations using the eigenvalue-eigenvector method

Functions used

[V, D] = eig(A) – Used to find the eigenvalues and eigenvectors of the matrix A. V gives a matrix with the eigenvectors as its columns, D is a diagonal matrix with eigenvalues as the diagonal elements.
subs(A, old, new) – Substitutes all instances of symbolic object old with new in the expression A.

Code

```
%% linear_DE_IVP_solver:   Solve a system of linear homogenoous DE IVP
function [sols, vals] = linear_DE_IVP_solver(A, B, C)     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 = i +
1;              end;
end;     end;     for i =
1:n
      sols(i) = (V(i, :) .* exp(eigenvalues' * t)) * consts;
end;     vals = solve(subs(sols, t, B) == C);     constnames
= fieldnames(vals);     % The final solutions
    for i = 1:n
        sols = subs(sols, consts(i), vals.(constnames{i}));     end;
```

Invocation

```
>> linear_IVP_system_solver([1 2; 3 2], 0, [0 -4])

ans =

[ (8*exp(-t))/5 - (8*exp(4*t))/5, - (8*exp(-t))/5 - (12*exp(4*t))/5]
```

QUESTION 3

Write a program to find the Eigenvalues and Eigenfunctions of the Sturm-Liouville problem given by X’’ + λX = 0 with boundary conditions X’(0) = X’(L) = 0

Functions used

syms x – Declares a symbolic object x
assume (condition)– Declare assumption on symbolic objects for the equation solver

Code

```
%% sturm_liouville: Calculate the eigenvalues and eigenvectors
%% For a Sturm Lioville problem with boundaries
%% [0, L] and eigenvalue lambda st. X'(L) = X'(0) = 0
function [e_value, e_function, non_zero] = sturm_liouville(L)
syms y(x)
syms lambda n
sprintf('Solving for various conditions...')
sprintf('When lambda > 0')
assume(lambda > 0);
solution = dsolve(diff(y, 2) + lambda * y == 0);
e_function = solution;
diff_sol = diff(solution, x);
vals = solve(subs(diff_sol, 0) == 0, subs(diff_sol, L) == 0);
non_zero = vals;
sprintf('Non-zero values in the solution')
vals
%% Eigenvalues 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 solutions possible
vals = solve(subs(diff_sol, 0) == 0, subs(diff_sol, L) == 0);
```

Invocation

```
>> [val fun coeff] = sturm_liouville(pi);
ans =
Solving for various conditions...
ans =
When lambda > 0
ans =
Non-zero values in the solution
vals =
C3: [0x1 sym]
lambda: [0x1 sym]
ans =
When lambda = 0
ans =
No non-trivial solution
ans =
When lambda < 0
Warning: Cannot find explicit solution.
> In solve (line 318)
In sturm_liouville (line 30)
>> val
val =
n^2
>> fun
fun =
```

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


>> coeff
coeff =

C3: [0x1 sym]
lambda: [0x1 sym]
```

QUESTION 4

Aim

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

Functions used

coeffs(P, x) – Obtain the coefficients of symbol x in the polynomial P

Code

```
%% lagrange_solver: Solves a linear partial differential equation
%% of the form Pp + Qq + R by Lagrange's method
function [u v] = lagrange_solver()
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;
% Separate 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
```

Invocation

```
>> [u v] = lagrange_solver()
u =
y^3/3 == x^3/3 + c1
v =
x^2/2 == z^2/2 + c2
```

QUESTION 5

Aim

Write a program to solve non-linear PDE using Charpit’s method

Functions used

coeffs(P, x) – Obtain the coefficients of symbol x in the polynomial P
diff(f, x) – Differentiate f w.r.t. x

Code

```
%% 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;
```

Invocation

```
>> charpit_solver()
fx =
-p
fy =
-q
fz =
1
fp =
- 2*p - x
fq =
- 2*q - y
ans =
a^2 + x*a + b^2 + y*b == z
```

QUESTION 6

Aim

Write a program to solve

$$(aD^2 + bD'D + CD'^2)\,z = f(x, y)$$

Functions used

coeffs(P, x) – Obtain the coefficients of symbol x in the polynomial P

Code

```
%% homogeneous_linear_PDE_solver: Solves a homogeneous linear partial
%% differetial equation, when f(x, y) is exponential
%%without repeating factors
function [ans] = homogeneous_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);
```

Invocation

```
>> homogenous_linear_PDE_solver()
ans =
-exp(5*x + 6*y)/91
```

QUESTION 7

Aim

Write a program to solve the heat equation

$$\frac{\partial u}{\partial t} = k\frac{\partial^2 u}{\partial x^2}$$

Functions used

dsolve(eqn) – Solves the ordinary differential equation given by `eqn`
solve(sys) – Solves the system of linear equations given

Code

```
%% heat_exact: Compute the exact symbolic representation 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, e_value] = heat_exact(k, a)
%% Using seperation of variables
syms G(t) phi(x) lambda
[e_value phi coeff] = sturm_liouville(a);
phi = subs(phi, lambda, e_value);
t_ode = diff(G, t) == -k * lambda * G;
t_sol = dsolve(t_ode);
sol = t_sol * phi;
```

Invocation

```
>> heat_exact(6, pi)
ans =
Solving for various conditions...
ans =
When lambda > 0
ans =
Non-zero values in the solution
vals =
C3: [0x1 sym]
lambda: [0x1 sym]
ans =
When lambda = 0
ans =
No non-trivial solution
ans =
When lambda < 0
Warning: Cannot find explicit solution.
> In solve (line 318)
In sturm_liouville (line 30)
In heat_exact (line 10)
ans =
C11*exp(-6*lambda*t)*(C3*cos(x*(n^2)^(1/2)) + C4*sin(x*(n^2)^(1/2)))
>> syms C11 C3 C4 bn
>> sol = ans;
>> sol = subs(subs(subs(sol, C3, 0), C11, 1), C4, bn)
sol =
bn*exp(-6*lambda*t)*sin(x*(n^2)^(1/2))
```

QUESTION 8

Aim

Write a program to solve the wave equation

Functions used

dsolve(eqn) – Solves the ordinary differential equation given by `eqn`
solve(sys) – Solves the system of linear equations given

Code

```
%% wave_exact: Compute the exact symbolic representation of the
%% Solution of the wave equation for waves along a
%% string of length a, with given initial and boundary
%% conditions
function [sol, e_value] = wave_exact(k, a)
%% Using seperation of variables
syms X(x) T(t) sig c1 c2 d1 d2 C3 C4
[e_value_t X coeff_x] = sturm_liouville(sig);
[e_value_t T coeff_t] = sturm_liouville(k * sig);
T = subs(subs(subs(T, x, t), C3, d1), C4, d2);
sol = [T * X];
```

Invocation

```
>> wave_exact(4, pi)
ans =
Solving for various conditions...
ans =
When lambda > 0
ans =
When lambda = 0
```

```
ans =
When lambda < 0
ans =
Solving for various conditions...
ans =
When lambda > 0
ans =
When lambda = 0
ans =
When lambda < 0
ans =
[ (C3*cos(x*((pi^2*n^2)/sig^2)^(1/2)) +
C4*sin(x*((pi^2*n^2)/sig^2)^(1/2)))*(d1*cos(t
```