Assignment 7

Numerical Methods: Gauss Quadrature and Legendre Expansions

Sahil Raj
2301CS41

November 21, 2025

Contents

Problem 1: Numerical Integration of $\frac{8(x+3)}{16+(x+3)^4}$ Using 4-Point Gauss Quadrature

Problem Statement

The objective of this problem is to **numerically integrate** the function

$$f(x) = \frac{8(x+3)}{16+(x+3)^4}$$

over the interval $x \in [-1, 1]$ using the **4-point Gauss-Legendre quadrature** method.

Methodology

Gauss-Legendre Quadrature (4-point)

The Gauss-Legendre quadrature method approximates the definite integral

$$\int_{-1}^{1} f(x)\,dx$$

as a weighted sum of function values at specific nodes:

$$I \approx \sum_{i=0}^{3} w_i\, f(x_i)$$

where x_i are the Gauss-Legendre nodes and w_i are the corresponding weights. For 4-point quadrature, the nodes and weights are precomputed as:

$$\begin{aligned} x_0 &= -0.86114, \quad w_0 = 0.34785 \\ x_1 &= -0.33998, \quad w_1 = 0.65215 \\ x_2 &= 0.33998, \quad w_2 = 0.65215 \\ x_3 &= 0.86114, \quad w_3 = 0.34785 \end{aligned}$$

Implementation Steps

1. Define the function $f(x) = \frac{8(x+3)}{16+(x+3)^4}$.
2. Specify the 4 Gauss-Legendre nodes and weights.
3. Compute the weighted sum $I = \sum_{i=0}^{3} w_i f(x_i)$ to approximate the integral.
4. Display the result.

Results

The numerical integration using 4-point Gauss quadrature gives:

- **Integral Value:** $I \approx 0.540410$

Conclusion

The 4-point Gauss-Legendre quadrature accurately approximates the integral of $f(x) = \frac{8(x+3)}{16+(x+3)^4}$ over $[-1, 1]$. For smooth functions, Gauss quadrature achieves high accuracy with very few nodes compared to methods like the trapezoidal rule.

Problem 2: Numerical Integration of $\frac{1}{x+1}$ in [0,1] Using 4-Point Gauss Quadrature with Variable Change

Problem Statement

The objective of this problem is to **numerically integrate** the function

$$f(x) = \frac{1}{x+1}$$

over the interval $x \in [0, 1]$ using the **4-point Gauss-Legendre quadrature** method. Since Gauss-Legendre nodes are defined over $[-1, 1]$, a **change of variables** is applied to transform the interval $[0, 1]$ to $[-1, 1]$.

Methodology

Change of Variables

The interval $[0, 1]$ is mapped to $[-1, 1]$ using

$$x = \frac{b-a}{2}t + \frac{b+a}{2} = \frac{t+1}{2}, \quad dx = \frac{1}{2}dt$$

where $t \in [-1, 1]$ represents the standard Gauss-Legendre variable. The transformed integral becomes:

$$\int_0^1 \frac{1}{x+1}\,dx = \frac{1}{2}\sum_{i=0}^{3} w_i\, f\Big(\frac{t_i+1}{2}\Big) = \sum_{i=0}^{3} w_i\, \frac{1}{t_i+3}$$

Gauss-Legendre Quadrature (4-point)

The 4-point Gauss-Legendre quadrature approximates the integral as a weighted sum of function values at specific nodes:

$$I \approx \sum_{i=0}^{3} w_i\, f(x_i)$$

with precomputed nodes and weights:

$$
\begin{aligned}
x_0 &= -0.86114, \quad w_0 = 0.34785 \\
x_1 &= -0.33998, \quad w_1 = 0.65215 \\
x_2 &= 0.33998, \quad w_2 = 0.65215 \\
x_3 &= 0.86114, \quad w_3 = 0.34785
\end{aligned}
$$

Implementation Steps

1. Define the mapped function $f(t) = \frac{1}{t+3}$.
2. Specify the 4 Gauss-Legendre nodes and weights.
3. Compute the weighted sum $I = \sum_{i=0}^{3} w_i f(t_i)$ to approximate the integral.
4. Display the result.

Results

The numerical integration using 4-point Gauss quadrature gives:

- **Integral Value:** $I \approx 0.693146$

Conclusion

The 4-point Gauss-Legendre quadrature accurately approximates the integral of $f(x) = \frac{1}{x+1}$ over $[0, 1]$. Using the change of variables, standard Gauss-Legendre nodes and weights can be applied to any finite interval, demonstrating the flexibility and efficiency of the Gauss quadrature method.

Problem 3: Expansion of a Radial Function in Terms of Legendre Polynomials

Problem Statement

The objective of this problem is to **represent a given radial function** $R(\theta)$ in terms of **Legendre polynomials** and reconstruct it using a finite number of terms. The input data consists of angles θ (in degrees) and corresponding radial distances $R(\theta)$ stored in 'Guitar_Theta_R.txt'.

Methodology

Data Preparation

1. Load the data file containing angles A and radial distances D.
2. Convert angles to radians: $\theta = \text{deg2rad}(A)$.
3. Compute $x = \cos(\theta)$, which maps the angular data to the domain of Legendre polynomials $[-1, 1]$.

Legendre Polynomial Expansion

The function $R(\theta)$ is expanded in terms of Legendre polynomials $P_l(x)$ as:

$$R(x) \approx \sum_{l=0}^{L} C_l \, P_l(x)$$

where C_l are the expansion coefficients.

Coefficient Calculation using Gauss-Legendre Quadrature:

1. Use 10-point Gauss-Legendre quadrature nodes ξ_i and weights w_i over $[-1, 1]$.
2. Interpolate the radial function at the nodes using 'pchip' for stability: $R(\xi_i)$.
3. Compute each coefficient:

$$C_l = \frac{2l+1}{2} \sum_{i=1}^{10} w_i \, R(\xi_i) \, P_l(\xi_i)$$

4. Maximum Legendre order used: $L = 15$.

Reconstruction

1. Create a fine grid of angles θ_{plot} and corresponding $x_{\text{plot}} = \cos(\theta_{\text{plot}})$.
2. Reconstruct $R_{\text{rec}}(\theta)$ from the Legendre expansion:

$$R_{\text{rec}}(x) = \sum_{l=0}^{L} C_l \, P_l(x)$$

Implementation Notes

- Legendre polynomials are computed using the recurrence relation:

$$P_0(x) = 1, \quad P_1(x) = x, \quad P_{n+1}(x) = \frac{(2n+1)xP_n(x) - nP_{n-1}(x)}{n+1}$$

- Interpolation using 'pchip' ensures smooth reconstruction even if data points are unevenly spaced.

Results

- The Legendre coefficients C_l capture the contribution of each polynomial to the overall function.
- The reconstructed function $R_{\text{rec}}(\theta)$ closely matches the original data $R(\theta)$, demonstrating the accuracy of the expansion.

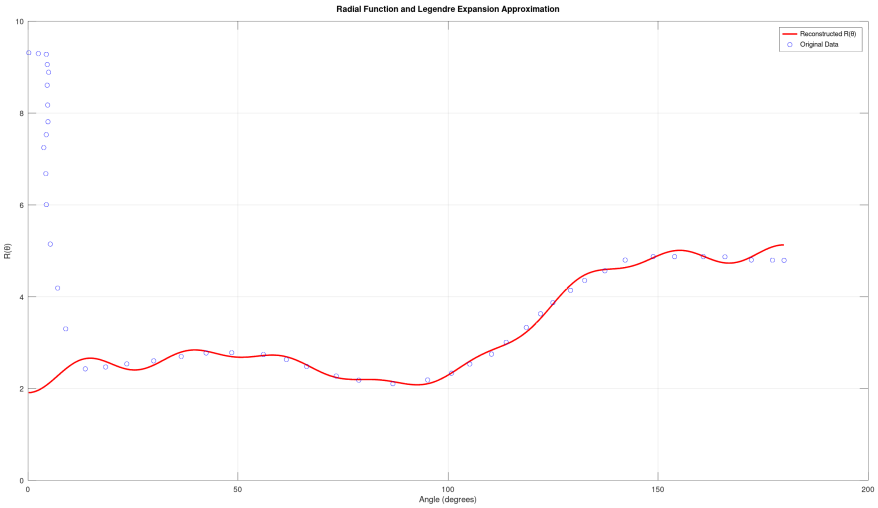

Figure 3.1: Reconstructed radial function $R(\theta)$ using Legendre polynomial expansion (red) compared with original data (blue circles).

Conclusion

The Legendre polynomial expansion successfully approximates the given radial function using a finite number of terms. The Gauss-Legendre quadrature provides an efficient method to compute expansion coefficients. The reconstructed function closely follows the original data, validating the approach for smooth angular functions.

Problem 4: Demonstrating the Orthogonality of the Legenedre Polynomials using Gauss 3-Point Quadrature

Problem Statement

The goal of this experiment is to **verify the orthogonality of Legendre polynomials** using **3-point Gauss–Legendre quadrature**. The known orthogonality condition is:

$$\int_{-1}^{1} P_i(x)\, P_j(x)\, dx = \begin{cases} \dfrac{2}{2i+1}, & \text{if } i = j, \\ 0, & \text{if } i \neq j. \end{cases}$$

A numerical quadrature scheme is used to approximate these integrals and compare them against the theoretical values.

Methodology

Gauss–Legendre Quadrature

A **3-point Gauss–Legendre rule** is used. It integrates exactly all polynomials up to degree:

$$2n - 1 = 5.$$

Thus, the integral of $P_i P_j$ is accurate only when:

$$i + j \leq 5.$$

The nodes and weights are:

$$x = \{-\sqrt{3/5},\ 0,\ \sqrt{3/5}\}, \quad w = \left\{\frac{5}{9},\ \frac{8}{9},\ \frac{5}{9}\right\}.$$

Legendre Polynomial Evaluation

Legendre polynomials up to order 5 are explicitly defined:

$$P_0 = 1, \quad P_1 = x, \quad P_2 = \frac{1}{2}(3x^2 - 1), \quad P_3 = \frac{1}{2}(5x^3 - 3x),$$

$$P_4 = \frac{1}{8}(35x^4 - 30x^2 + 3), \quad P_5 = \frac{1}{8}(63x^5 - 70x^3 + 15x).$$

These expressions form the integrand:

$$f_{ij}(x) = P_i(x)\, P_j(x).$$

Numerical Integration

For each pair (i, j) with $0 \leq i, j \leq 5$, the integral is approximated as:

$$I_{ij} \approx \sum_{k=1}^{3} w_k \, P_i(x_k) \, P_j(x_k).$$

To compare with the analytical orthogonality formula, the diagonal entries are normalized by:

$$\frac{2i+1}{2},$$

so the expected normalized matrix becomes:

$$M_{ij} = \begin{cases} 1, & i = j, \\ 0, & i \neq j. \end{cases}$$

Only entries with $i + j \leq 5$ are meaningful because of the degree limitation of the quadrature rule.

Results

Orthogonality Matrix

The computed matrix shows:

- diagonal entries very close to 1,
- off-diagonal entries near 0,
- expected inaccuracies whenever $i + j > 5$, since the 3-point rule cannot integrate such products exactly.

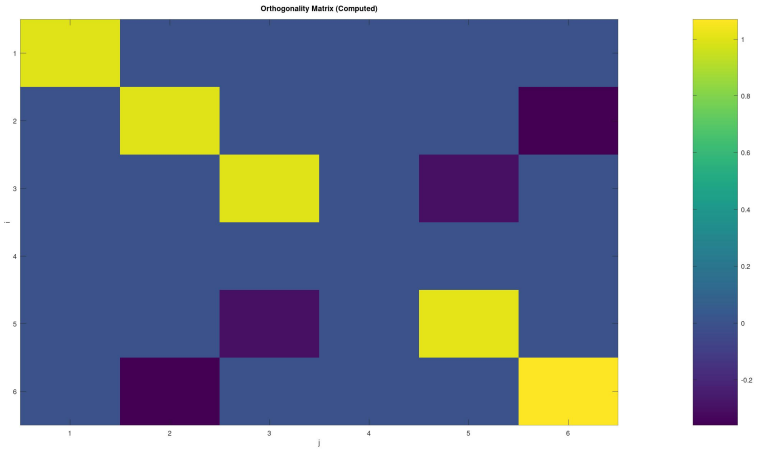

Figure 4.1: Computed orthogonality matrix using 3-point Gauss–Legendre quadrature.

Legendre Polynomial Plots

The Legendre polynomials P_0 through P_5 are plotted over $[-1, 1]$, showing the typical oscillatory structure and symmetry.

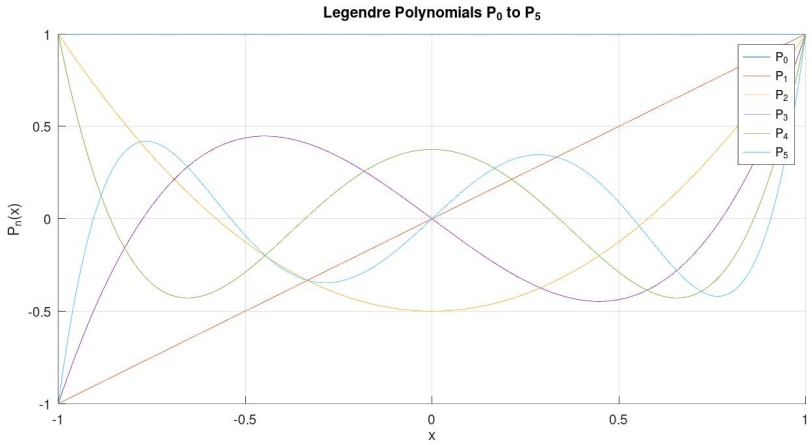

Figure 4.2: Plots of Legendre polynomials $P_0(x)$ to $P_5(x)$.

Conclusion

This experiment confirms the expected orthogonality of Legendre polynomials using a 3-point Gauss–Legendre quadrature rule. The results match theoretical values for all (i, j) such that $i + j \leq 5$. This verifies:

- the correctness of the Legendre polynomial formulas,
- the reliability of Gauss–Legendre quadrature,
- and the orthogonality structure of the polynomial family.

The visualizations further support these findings.

Problem 5: Spherically Symmetric Gaussian Charge Distribution

Problem Statement

A spherically symmetric charge distribution is given by

$$\rho(r) = \rho_0\, e^{-r^2/a^2},$$

where ρ_0 is the central charge density, a is a characteristic length scale, and r is the radial distance from the centre.

(a) Calculate the total charge Q contained in the entire space.

(b) Find the electric field $E(r)$ at a distance r from the centre using Gauss's law.

Analytic Calculations

(a) Total charge Q

Total charge is

$$Q \;=\; \int_{\mathbb{R}^3} \rho(r)\, dV \;=\; 4\pi \int_0^\infty \rho(r)\, r^2\, dr \;=\; 4\pi\rho_0 \int_0^\infty r^2 e^{-r^2/a^2}\, dr.$$

Use the standard integral

$$\int_0^\infty r^2 e^{-\alpha r^2}\, dr \;=\; \frac{\sqrt{\pi}}{4}\, \alpha^{-3/2}, \qquad (\Re\alpha > 0).$$

With $\alpha = 1/a^2$ we get

$$\int_0^\infty r^2 e^{-r^2/a^2}\, dr = \frac{\sqrt{\pi}}{4}\, (1/a^2)^{-3/2} = \frac{\sqrt{\pi}}{4}\, a^3.$$

Therefore the total charge is

$$\boxed{Q \;=\; 4\pi\rho_0 \cdot \frac{\sqrt{\pi}}{4}\, a^3 \;=\; \rho_0\, \pi^{3/2}\, a^3}.$$

This is the exact analytic result.

(b) Electric field $E(r)$ via Gauss's law

By spherical symmetry, the electric field points radially and (for $r > 0$) Gauss's law gives

$$E(r)\,(4\pi r^2) \;=\; \frac{Q_{\mathrm{enc}}(r)}{\varepsilon_0},$$

so

$$E(r) \;=\; \frac{1}{4\pi\varepsilon_0}\,\frac{Q_{\mathrm{enc}}(r)}{r^2},$$

where

$$Q_{\mathrm{enc}}(r) \;=\; 4\pi\rho_0 \int_0^r r'^2 e^{-r'^2/a^2}\,dr'.$$

Evaluate the radial integral. Let $t = r'/a$. Then

$$\int_0^r r'^2 e^{-r'^2/a^2}\,dr' = a^3 \int_0^{r/a} t^2 e^{-t^2}\,dt.$$

Use the antiderivative

$$\int t^2 e^{-t^2}\,dt \;=\; \frac{\sqrt{\pi}}{4}\,\mathrm{erf}(t) - \frac{t}{2}e^{-t^2},$$

so

$$\int_0^{r/a} t^2 e^{-t^2}\,dt = \frac{\sqrt{\pi}}{4}\,\mathrm{erf}\Big(\frac{r}{a}\Big) - \frac{r}{2a}e^{-r^2/a^2}.$$

Hence

$$Q_{\mathrm{enc}}(r) = 4\pi\rho_0 a^3 \left[\frac{\sqrt{\pi}}{4}\,\mathrm{erf}\Big(\frac{r}{a}\Big) - \frac{r}{2a}\,e^{-r^2/a^2}\right].$$

This simplifies to

$$Q_{\mathrm{enc}}(r) = \rho_0 \left[\pi^{3/2}a^3\,\mathrm{erf}\Big(\frac{r}{a}\Big) \;-\; 2\pi a^2 r\,e^{-r^2/a^2}\right]$$

Substituting into Gauss's law gives

$$E(r) = \frac{1}{4\pi\varepsilon_0}\frac{Q_{\mathrm{enc}}(r)}{r^2} = \frac{\rho_0}{4\pi\varepsilon_0}\left[\frac{\pi^{3/2}a^3}{r^2}\,\mathrm{erf}\Big(\frac{r}{a}\Big) - \frac{2\pi a^2}{r}e^{-r^2/a^2}\right].$$

This expression reduces to the expected behaviours:

- As $r \to 0$, series expansion shows $E(r) \propto r$ (regular at origin).
- As $r \to \infty$, $\mathrm{erf}(r/a) \to 1$ and the second term vanishes, giving

$$E(r) \xrightarrow{r\to\infty} \frac{1}{4\pi\varepsilon_0}\frac{Q}{r^2} \quad \text{with} \quad Q = \rho_0\pi^{3/2}a^3,$$

  i.e. a point-charge behaviour at large distances.

Numerical Approach (Octave Code)

Integral formulation used in the code

The code numerically evaluates a radial integral of the form

$$I(R) = \int_{-1}^{1} p^2 \, \exp\big(-R^2 p^2/a^2\big) \, dp,$$

using a 4-point Gauss–Legendre quadrature on the interval $[-1, 1]$:

$$I(R) \approx \sum_{k=1}^{4} w_k \, p_k^2 e^{-R^2 p_k^2/a^2},$$

where p_k and w_k are the Gauss–Legendre nodes and weights (exact forms used in the code).

The code forms the electric field as

$$E(R) \;=\; \frac{\rho_0 R}{2\varepsilon_0} \, I(R),$$

which is equivalent to using an appropriate change of variables to reduce the enclosed-charge integral to this form.

Quadrature rules used

- 4-point Gauss–Legendre: exact for polynomials up to degree 7; exact nodes/weights used in closed form.
- 2-point Gauss–Hermite (used for checking the total charge numerically): nodes $\pm 1/\sqrt{2}$ with weights $\sqrt{\pi}/2$.

Results

Analytic values

Using the analytic formulas with $\rho_0 = 1$, $a = 1$ and $\varepsilon_0 = 1$ (the same nondimensional choices used in the code) we have:

$$Q_{\text{analytic}} = \pi^{3/2} \, a^3 \rho_0 = \pi^{3/2} \approx 5.568327 \quad (\text{for } a = \rho_0 = 1).$$

The analytic electric field is given by the boxed expression above; it can be evaluated numerically for a grid of r values for comparison with the code.

Numerical evaluation (from the code)

The provided Octave script computes:

- a numerical estimate of the total charge Q (via a quadrature used in the code),
- $E(R)$ on a grid $R \in [0.1, 10]$ using the 4-point Gauss–Legendre evaluation described above,
- and plots $E(R)$ versus R.

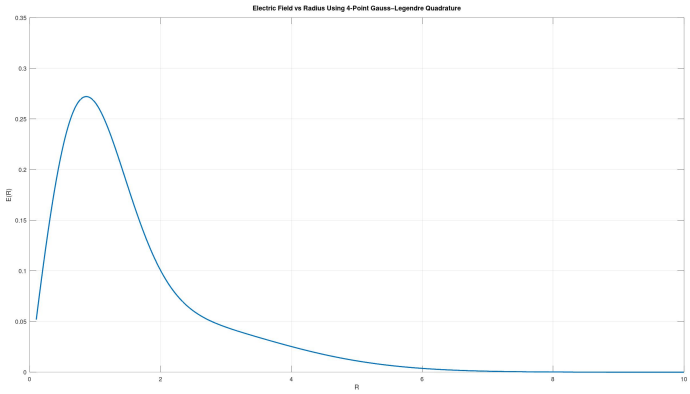

Figure 5.1: Electric field $E(r)$ (computed numerically).

Appendix A: Octave Code for Problem 1

```
%% Program to compute the definite integral of the function using
%% Gauss Quadrature Technique (4-points)
%%
%% Author: Sahil Raj
%% Assignment 7 Problem 1

clc; clear all;

% Precomputed Node points from the table
x0 = -0.86114;
x1 = -0.33998;
x2 =  0.33998;
x3 =  0.86114;

% Precomputed weights from the table
w0 = 0.34785;
w1 = 0.65215;
w2 = 0.65215;
w3 = 0.34785;

function y = f(x)
   y = (8 * (x+3)) / (16 + power(x+3, 4));
endfunction

I = w0*f(x0) + w1*f(x1) + w2*f(x2) + w3*f(x3);

printf("Integral of the function in [-1, 1] is: %f\n", I);
```

Appendix B: Octave Code for Problem 2

```
%% Program to compute the definite integral of the function using
%% Gauss Quadrature Technique (4-points)
%%
%% Author: Sahil Raj
%% Assignment 7 Problem 2

%% For function 1/x+1 in range [0, 1] using change of variables

clc; clear all;

% Precomputed Node points from the table
x0 = -0.86114;
x1 = -0.33998;
x2 =  0.33998;
x3 =  0.86114;

% Precomputed weights from the table
w0 = 0.34785;
w1 = 0.65215;
w2 = 0.65215;
w3 = 0.34785;

function y = f(x)
   y = 1 / (x+3);
endfunction

I = w0*f(x0) + w1*f(x1) + w2*f(x2) + w3*f(x3);

printf("Integral of the function in [-1, 1] is: %f\n", I);
```

Appendix C: Octave Code for Problem 3

```
%% Program to compute the expansion of the given
%% function in terms of legendre polynomials
%%
%% Author: Sahil Raj
%% Assignment 7 Problem 3

clc; clear all;

% Load data
T = dlmread("Guitar\_Theta\_R.txt", "\t");
A = T(:,1);   % angles in degrees
D = T(:,2);   % distances

theta = deg2rad(A);
x = cos(theta);

% Gauss-Legendre quadrature nodes and weights (10-point)
xi = [
   -0.9739065285;
   -0.8650633666;
   -0.6794095682;
   -0.4333953941;
   -0.1488743389;
    0.1488743389;
    0.4333953941;
    0.6794095682;
    0.8650633666;
    0.9739065285
];
wi = [
    0.0666713443;
    0.1494513491;
    0.2190863625;
    0.2692667193;
    0.2955242247;
    0.2955242247;
    0.2692667193;
    0.2190863625;
    0.1494513491;
    0.0666713443
];
```

```
42
43 % Interpolation function R(x)
44 function Rval = computeR(xval, xdata, Ddata)
45     % Interpolate using pchip for stability
46     Rval = interp1(xdata, Ddata, xval, 'pchip', 'extrap');
47 endfunction
48
49 % Legendre polynomial using recurrence
50 function Pl = legendre\_numeric(l, x)
51     if l == 0
52         Pl = 1;
53         return;
54     elseif l == 1
55         Pl = x;
56         return;
57     end
58     P0 = 1;
59     P1 = x;
60     for n = 1:l-1
61         Pn1 = ((2*n+1) * x .* P1 - n * P0)/(n+1);
62         P0 = P1;
63         P1 = Pn1;
64     end
65     Pl = P1;
66 endfunction
67
68 % Compute coefficients
69 maxL = 15;      % maximum Legendre order
70 C = zeros(maxL+1,1);
71 for l = 0:maxL
72     sumval = 0.0;
73     for i = 1:length(xi)
74         Rval = computeR(xi(i), x, D);
75         Pl = legendre\_numeric(l, xi(i));
76         sumval = sumval + wi(i) * Rval * Pl;
77     end
78     C(l+1) = (2*l+1)/2 * sumval;   % normalization factor for
       P\_l
79 end
80
81 % Reconstruct R(theta) from Legendre expansion
82 theta\_plot = linspace(min(theta), max(theta), 300);
83 x\_plot = cos(theta\_plot);
84 R\_rec = zeros(size(x\_plot));
85 for l = 0:maxL
86     R\_rec = R\_rec + C(l+1) * legendre\_numeric(l, x\_plot);
87 end
88
89 % Plot
90 figure;
```

```
plot(rad2deg(theta\_plot), R\_rec, 'r-', 'LineWidth', 2); hold
    on;
plot(A, D, 'bo');
xlabel('Angle (degrees)');
ylabel('R(\theta)');
title('Radial Function and Legendre Expansion Approximation');
legend('Reconstructed R(\theta)','Original Data');
grid on;
```

Appendix D: Octave Code for Problem 4

```
clc; clear all;
%%  Verification of the Orthogonality of Legendre Polynomials
%%  Using 3-Point Gauss-Legendre Quadrature
%%
%%  The 3-point quadrature formula is exact for polynomials of
%%  degree <= 5. Therefore, the integral of P\_i * P\_j is
   reliable
%%  only when i + j <= 5.
%%
%%  Author : Sahil Raj
%%  Course : Assignment 7 - Problem 4

N = 5;                  % Maximum order to test

%%  Quadrature Data (3-Point Gauss-Legendre)
%%  These weights and nodes integrate exactly up to degree 5.
global nP;    nP    = 3;
global lambda; lambda = [ 5/9; 8/9; 5/9 ];
global nodeps; nodeps = [ -sqrt(3/5); 0; sqrt(3/5) ];


%%  Custom Legendre Polynomials P\_0 ... P\_5
function y = LegendreCustom(n, x)
    switch n
        case 0, y = 1.0;
        case 1, y = x;
        case 2, y = (3*x*x - 1)/2;
        case 3, y = (5*x\^3 - 3*x)/2;
        case 4, y = (35*x\^4 - 30*x\^2 + 3)/8;
        case 5, y = (63*x\^5 - 70*x\^3 + 15*x)/8;
        otherwise
            error("LegendreCustom: order not implemented.");
    end
endfunction


%%  Gauss Quadrature Integrator
function I = gaussQuad(fn)
    global nP lambda nodeps;
    I = 0.0;
    for i = 1:nP
```

```
        I += lambda(i) * fn(nodeps(i));
    end
endfunction


%%  Compute Orthogonality Matrix
intmatrix = zeros(N+1, N+1);

for i = 0:N
    for j = 0:N
        fn = @(x) LegendreCustom(i,x) * LegendreCustom(j,x);
        I = gaussQuad(fn);

        % Apply normalization for diagonal entries:
        if i == j
            I *= (2*i + 1) / 2;
        endif

        intmatrix(i+1, j+1) = I;
    endfor
endfor

%% Display matrix
disp("Orthogonality Matrix:");
disp(intmatrix);


%%  PLOTTING SECTION

% Plot Legendre polynomials P0..P5
x = linspace(-1, 1, 400);
figure(1); hold on; grid on;
title("Legendre Polynomials P\_0 to P\_5");
xlabel("x"); ylabel("P\_n(x)");

for n = 0:5
    plot(x, arrayfun(@(t) LegendreCustom(n,t), x));
endfor
legend("P\_0","P\_1","P\_2","P\_3","P\_4","P\_5");


% Heatmap-like plot of orthogonality matrix
figure(2);
imagesc(intmatrix);
title("Orthogonality Matrix (Computed)");
xlabel("j"); ylabel("i");
colorbar();
set(gca, "XTick", 1:N+1, "YTick", 1:N+1);
```

Appendix E: Octave Code for Problem 5

```
1
2  ng = 4;
3  nodes = [
4      -sqrt((3/7) - (2/7)*sqrt(6/5));
5      -sqrt((3/7) + (2/7)*sqrt(6/5));
6       sqrt((3/7) + (2/7)*sqrt(6/5));
7       sqrt((3/7) - (2/7)*sqrt(6/5))
8  ];
9
10 weights = [
11     (18 - sqrt(30)) / 36;
12     (18 + sqrt(30)) / 36;
13     (18 + sqrt(30)) / 36;
14     (18 - sqrt(30)) / 36
15 ];
16
17
18 %% Physical constants
19 rho0 = 1.0;
20 eps0 = 1.0;
21 a    = 1.0;
22
23
24 %% Integrand: p\^2 * e\^{-(R\^2 p\^2)/a\^2}
25 function y = fun(p, R, a)
26     y = exp(-(R*R)*(p*p)/(a*a)) * (p*p);
27 endfunction
28
29
30 function E = computeE(R, nodes, weights, rho0, eps0, a)
31     E = (rho0 * R) / (2 * eps0);
32
33     I = 0;
34     for i = 1:length(nodes)
35         I += weights(i) * fun(nodes(i), R, a);
36     end
37
38     E *= I;
39 endfunction
40
41
```

```
%% Total charge using 2-pt Gauss-Hermite rule
x  = [-1/sqrt(2); 1/sqrt(2)];
wH = [sqrt(pi)/2; sqrt(pi)/2];

Q = 2*pi*a\^3*rho0 * (wH(1)*x(1)\^2 + wH(2)*x(2)\^2);
fprintf("Total charge in space: %.6f\n", Q);


%% Compute E(R) over range
Rs = 0.1:0.01:10;
Es = arrayfun(@(R) computeE(R, nodes, weights, rho0, eps0, a),
   Rs);


%% PLOT
figure;
plot(Rs, Es, "linewidth", 2);
grid on;
xlabel("R");
ylabel("E(R)");
title("Electric Field vs Radius Using 4-Point Gauss-Legendre
   Quadrature");
```