Dynamic Soaring: Assignment 4

Izza (AE18B006)

Patibandla Krishna Vamsi (AE17B036) &

RV Ramkumar (AE20D025)

1 Integration of functions

Gauss Quadrature using Legendre Basis:

The definite integral of any function can be evaluated by quadrature rule which is the weighted sum of the integrand evaluated at quadrature points.

$$\int_a^b f(x)dx = \sum_{i=0}^{N} w_i f(q_i) \tag{1}$$

By use of scaling i.e. $x = a + \frac{(b-a)}{2}(t+1)$ and $dx = \frac{(b-a)}{2}dt$, the integration interval can be converted from $[a, b]$ to $[-1, 1]$ which is given below:

$$\int_a^b f(x)dx = \frac{(b-a)}{2}\int_{-1}^{1} f\left(a + \frac{(b-a)}{2}(t+1)\right) dt$$

Basic idea of Gauss quadrature is to determine quadrature points and weights such that it evaluates as high degree polynomial as possible. By choosing N quadrature points, it is possible to approximate the integrand (function) by using a polynomial of degree $(2N-1)$ which is better approximation than other integration rules using Newton-Cotes families viz. midpoint rule, trapezoidal rule, Simpson's rule, etc.,

If the zeros of the Legendre polynomial is used as quadrature nodes, the weight at each node is estimated by using the expression given below:

$$w_i = \frac{2}{(1-x_i^2)[P_n'(x_i)]^2}$$

where $P_n(x)$ is the nth degree Legendre polynomial expressed by the equation 2 and $P_n^{'}(x_i)$ is the derivative of the nth degree Legendre polynomial evaluated at the ith quadrature point (x_i).

$$P_n(x) = \frac{1}{2^n n!}\frac{d^n}{dx^n}[(x^2-1)^n] \tag{2}$$

The above equation can also be written as a recurence relation as follows:

$$(n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x) \tag{3}$$

with

$$P_1(x) = x; \qquad P_2(x) = \frac{1}{2}(3x^2-1) \tag{4}$$

By using the recurrence relation as per equation 3, the roots of the Legendre polynomial can be obtained numerically using say Newton-Raphson method to find the quadrature points and the corresponding weights.

The table below 1 gives the nodes and weights for Legendre polynomials till 5th degree. The 'n' also represents the number of points (roots of the Legendre polynomial of degree n).

Table 1: The Legendre Quadrature Points and Weights

n	Nodes	Weights
1	0.00000	2.00000
2	$\pm\frac{1}{\sqrt{3}} = \pm 0.55735$	1.00000
3	0; $\pm$0.774597	0.88889; 0.55556
4	$\pm$0.339981; $\pm$0.861136	0.652145 ; 0.347855
5	0 ; $\pm$0.538469; $\pm$0.90618	0.568889; 0.478629; 0.236927

Trapezoidal Rule:

The trapezoidal rule is a two point rule due to the fact that the two end points of the definite integral is used. By considering the points $(a, f(a))$ and $(b, f(b))$, a linear Lagrange polynomial can be formulated and is given by.

$$P(x) = f(a) + \frac{f(b) - f(a)}{b - a}(x - a) \tag{5}$$

therefore, the integral of the function $f(x)$ can be approximated as the integral of the above polynomial function as

$$\int_a^b f(x)dx \approx \int_a^b P(x)dx$$

By performing the integration, we get the following

$$\int_a^b f(x)dx \approx \int_a^b P(x)dx = (b - a)\frac{f(b) + f(a)}{2} \tag{6}$$

Since the trapezoidal method uses a linear interpolation function in the entire domain, it is better to divide domain into multiple small domains and perform the integration for better accuracy.

Evaluation of $\int_1^2 \frac{1}{x}dx$ using Gauss Qaudrature Legendre Method

The Matlab code below evaluate integral using the Gauss Quadrature using Legendre nodes.

```
clear all; close all; clc;

N = 5;               % Polynomial degree;
a = 1.0; b=2.0;      % Definite integral lower and upper limits

% The function lgwt(N,a,b) gives the output of the weights and the node
% point after incorporating the scaling and there is no requirement of
% multiplying

[tau,weight] = lgwt(N,a,b);

% The function to be integrated
func = @(x) 1.0/x;      % The function is f(x) = 1/x

n=length(tau);  % the length is same as that of the polynomial degree
sum(1)=0.0;

for i=1:n
    sum(i+1) = sum(i)+weight(i)*func(tau(i));
end
integral = sum(end);
display(['The value of the Integral of f(x)=1/x bw 1 and 2: ',num2str(integral)])
```

```
 The value of the Integral of f(x)=1/x bw 1 and 2 : 0.69315
```

Evaluation of $\int_0^{\pi/2} \sin(x)dx$ using Gauss Qaudrature Legendre Method

The Matlab code below evaluate integral using the Gauss Quadrature using Legendre nodes.

```
clear all; close all; clc;

N = 5;                % Polynomial degree;
a = 0.0; b=pi/2;      % Definite integral lower and upper limits

% The function lgwt(N,a,b) gives the output of the weights and the node
% point after incorporating the scaling and there is no requirement of
% multiplying

[tau,weight] = lgwt(N,a,b);

% The function to be integrated
func = @(x) sin(x);

n=length(tau);  % the length is same as that of the polynomial degree
sum(1)=0.0;

for i=1:n
    sum(i+1) = sum(i)+weight(i)*func(tau(i));
end
integral = sum(end);
display(['The value of the Integral of f(x)= sin(x) bw 0 and pi/2 : ',num2str(integral)])
```

```
The value of the Integral of f(x)= sin(x) bw 0 and pi/2 : 1
```

Evaluation of $\int_1^2 \frac{1}{x}dx$ using Trapezoidal Rule

The Matlab code below evaluate the integral using the Trapezoidal rule.

```
clear all; close all; clc;

a = 1.0; b=2;     % Definite integral lower and upper limits

% The function to be integrated
func = @(x) 1.0/x;

n=5;    % number of nodes similar to the Legendre basis
h = (b-a)/n;
sum1(1) = 0.0;
for i=1:n+1
    f(i) = func(a+(i-1)*h);
end
for i=1:n
    sum1(i+1) = sum1(i) + h*0.5*(f(i)+f(i+1));
end

integral = sum1(end);

display(['The value of the Integral of f(x)= 1/x bw 1 and 2: ',num2str(integral)])
```

```
The value of the Integral of f(x)= 1/x bw 1 and 2: 0.69563
```

Evaluation of $\int_0^{\pi/2} \sin(x)dx$ using Trapezoidal Rule

The Matlab code below evaluate the integral using the Trapezoidal rule.

```
clear all; close all; clc;

a = 0.0; b=pi/2;     % Definite integral lower and upper limits

% The function to be integrated
func = @(x) sin(x);

n=5;                % No. of nodes
h = (b-a)/n;
sum1(1) = 0.0;
for i=1:n+1
    f(i) = func(a+(i-1)*h);
end
for i=1:n
    sum1(i+1) = sum1(i) + h*0.5*(f(i)+f(i+1));
end

integral = sum1(end);

display(['The value of the Integral of f(x)= sin(x) bw 0 and pi/2: ',num2str(integral)])
```

```
The value of the Integral of f(x)= sin(x) bw 0 and pi/2: 0.99176
```

Comparison of the Results

The below mentioned table 2 gives the consolidated results of the integrals evaluated using different method for a total number of nodes of 5. It is evident that the Gauss Quadrature (Legendre) is corroborating well with analytical method.

Table 2: Comparison of the Results

Integral	No. of Nodes	Gauss-Quadrature	Trapezoidal Rule	Analytical
$\int_1^2 \frac{1}{x}dx$	5	0.69315	0.69563	0.69315
$\int_0^{\pi/2} \sin(x)dx$	5	1.0	0.99176	1.0

2 Fitting Interpolation for the Functions

A function can be approximated by a polynomial function using uniform / non-uniform node distribution which equals the function value exactly at those nodes using Lagrange, Van der Monde polynomial, etc.,

Interpolation of $ln(x)$

The code below compares the polynomial interpolation of $ln(x)$ using Lagrange interpolation of degree 10 considering uniform node distribution and Legendre node distribution. The function $ln(x)$ is evaluated between 1 and 2.

```
clear all; close all; clc;

n  = 11;                        % No. of nodes
a = 1;
b = 2;
x = linspace(a,b,n);             % uniform nodes between and b;
[x1,wt] = lgwt(n,a,b);           % nodes and weight using legendre basis;

f = @(x) log(x);               % Given function

y = f(x);
L=zeros(n,n);
for i=1:n
    v = 1;
    for j=1:n
        if i~=j
            v = conv(v,poly(x(j)))/(x(i)-x(j));
        end
    end
    L(i,:) = v*y(i);
end
p=sum(L);                    % polynomial coefficients for uniform nodes

x1 = flip(x1)
y1= f(x1);
L1=zeros(n,n);
for i=1:n
    v1 = 1;
    for j=1:n
        if i~=j
            v1 = conv(v1,poly(x1(j)))/(x1(i)-x1(j));
        end
    end
    L1(i,:) = v1*y1(i);
end
p1=sum(L1);                    % polynomial coefficients for Legendre nodes

xx = linspace(a,b,50);
yy = f(xx);                    % analytical value;
y_Lag = polyval(p,xx);         % Using Lagrange polynomial - uniform node distribution
y_Leg = polyval(p1,xx);        % Using Lagrange polynomial - Legendre node distribution

figure(1);
plot(xx,y_Lag,'r-+'); hold on;
plot(xx,yy,'b-');
ylim([0 1]);
legend('Poly (uniform nodes)', 'Analytical');
fs='fontsize';set(gca,fs,11);
xlabel('x',fs,11); ylabel('log(x)',fs,11);
print -depsc 'poly_uni_log.eps'

figure(2);
plot (xx,y_Leg,'r-+'); hold on;
plot (xx,yy,'b-');
ylim([0 1]);
legend('Poly (Legendre nodes)', 'Analytical');
fs='fontsize'; set(gca,fs,11);
xlabel('x',fs,11); ylabel('log(x)',fs,11);
print -depsc 'poly_leg_log.eps'
```

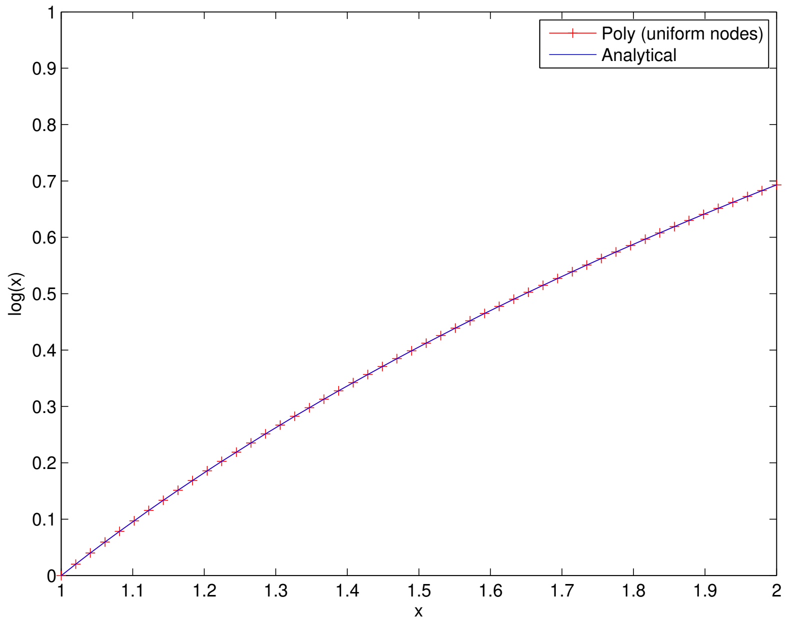

Figure 1: Polynomial fit of $ln(x)$ with Uniform Node Distribution

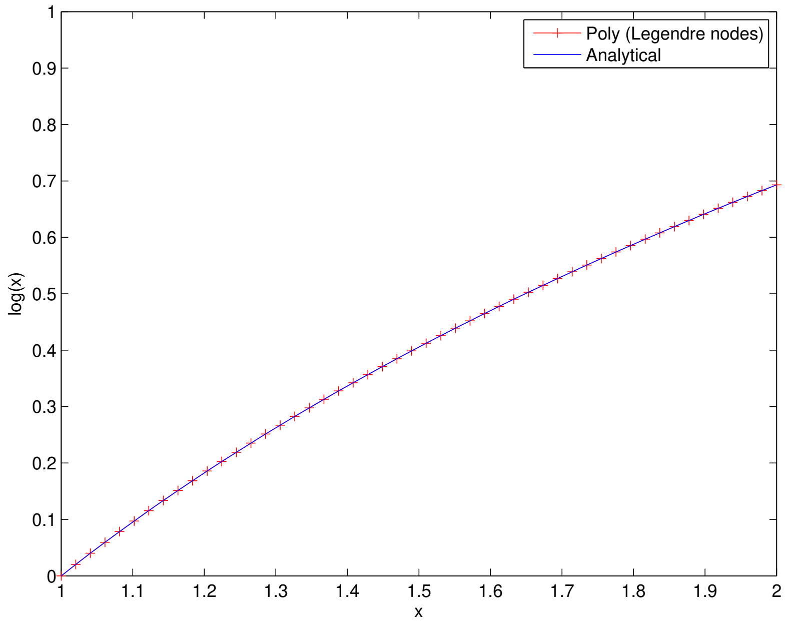

Figure 2: Polynomial fit of $ln(x)$ with Legendre Node Distribution

Interpolation of $\cos(x)$

The code below compares the polynomial interpolation of $\cos(x)$ using Van der Monde interpolation of degree 10 considering uniform node distribution and Legendre node distribution. The function $\cos(x)$ is evaluated between $-\pi$ and π.

```
clear all; close all; clc;

n = 11;                          % No. of nodes
a = -pi;
b = pi;
x = linspace(a,b,n);           % Uniform Node Distribution
[x1,wt] = lgwt(n,a,b);         % Legendre Node Distribution
```

```
f = @(x) cos(x);                % Function

Vinv = inv(vander(x));          % Using Vandermonde Matrix
p = Vinv*f(x)';                 % Uniform Node Distribtuion


Vinv1 = inv(vander(x1));        % Using Vandermonde Matrix
p1 = Vinv1*f(x1);               % Legendre Node Distribution



xx = linspace(a,b,50);
yy = f(xx);                  % Actual function
y  = polyval(p,xx);          % Interpolation function using uniform nodes
y1 = polyval(p1,xx);         % Interpolation function using Legendre nodes

figure(1);
plot(xx,y,'r-+'); hold on;
plot(xx,yy,'b-');
axis equal;
legend('Vandermonde (Uni Nodes)', 'Analytical');
fs='fontsize'; set(gca,fs,11);
xlabel('x',fs,11); ylabel('cos(x)',fs,11);
print -depsc 'vander_uni_cos.eps'

figure(2);
plot (xx,y1,'r-+'); hold on;
plot (xx,yy,'b-');
axis equal;
legend('Vandermonde (Leg Nodes)', 'Analytical');
fs='fontsize'; set(gca,fs,11);
xlabel('x',fs,11); ylabel('cos(x)',fs,11);
print -depsc 'vander_leg_cos.eps'
```

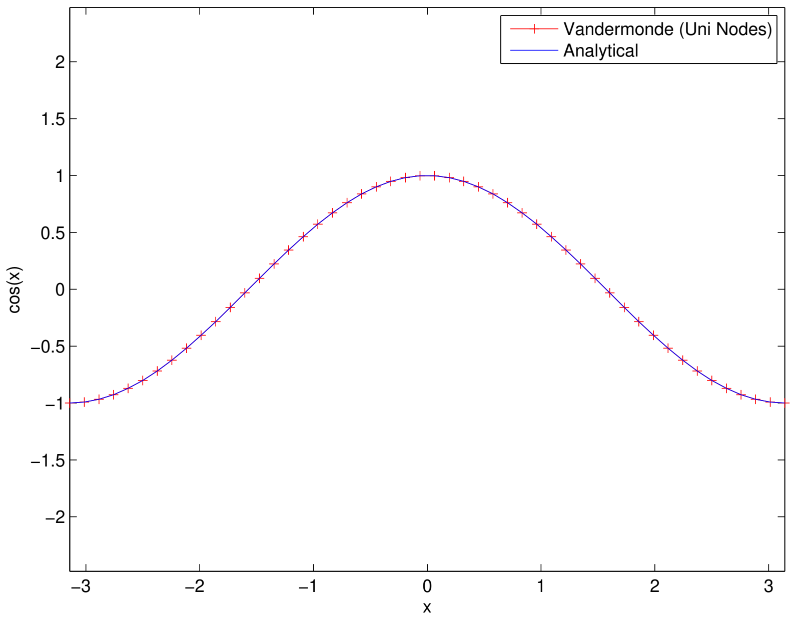

Figure 3: Polynomial fit of $\cos(x)$ with Uniform Node Distribution

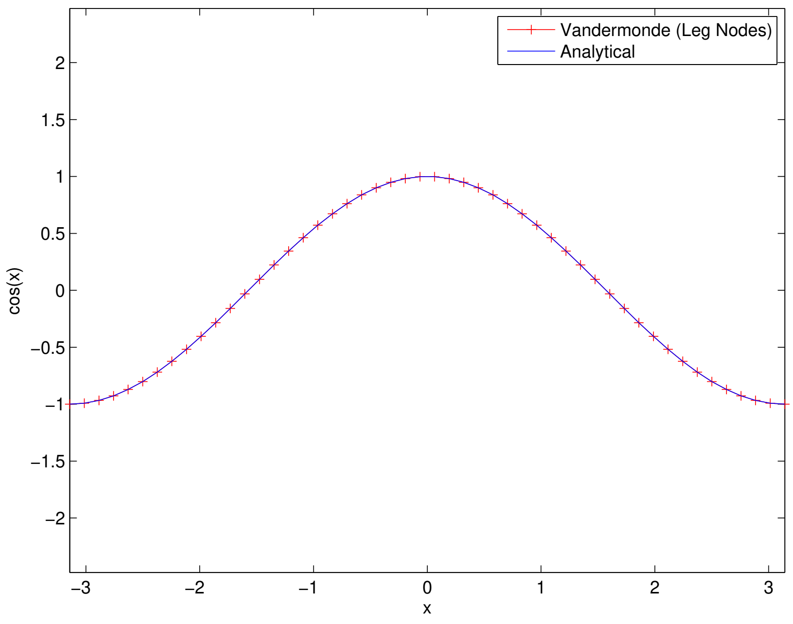

Figure 4: Polynomial fit of cos(x) with Legendre Node Distribution

Evaluating derivative of $ln(x)$ using LGL differentiation matrix

The Matlab code below evaluates the derivative using the differentiation matrix with collocation at the Legendre-Gauss-Lobatto nodes including the end points -1 to 1.

```
clear all; close all; clc;

N = 50;              % Polynomial degree;

% legDc.m
% Computes the Legendre differentiation matrix with collocation at the
% Legendre-Gauss-Lobatto nodes.

[x,D] = legDc(N-1);

f = @(x) log(x);      % The function is f(x) = log(x)
g = @(x) 1.0/x ;      % g = f' , i.e derivative of f

Xdot = D*f(x);
n = length(x);
fdot = zeros(n,1);

for i = 1 : n
    fdot(i) = g(x(i));
end

figure(1);
plot(x,Xdot,'b-o'); hold on;
plot(x,fdot,'r-');
legend('Using Dmat', 'Analytical');
fs='fontsize'; set (gca,fs,11);
print -depsc 'deriv_logx.eps'
```

```
Warning: Imaginary parts of complex X and/or Y arguments ignored
```

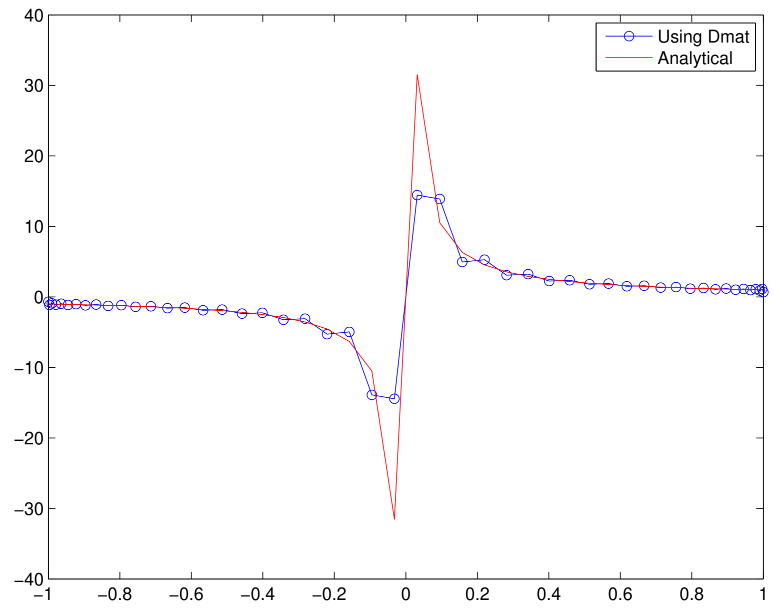

Figure 5: Derivative of log(x) using LGL Node Distribution

Evaluating derivative of $ln(x)$ using uniform Node Distribution

The matlab code below evaluates the derivative of $ln(x)$ between -1 and 1 including the end points using the finite difference approach using the uniform nodes distribution. Though a Differentiation Matrix using finite difference method can be obtained but the code is written using simple approach. At the boundaries, the forward Euler and backward Euler of first order and in the nodes between the central difference scheme is used.

```
    clear all; close all; clc;

N=49;               % No. of points-1
a=-1; b=1;          % Boundary value;
delx = (b-a)/N;

for i=1:N+1
    x(i) = a + (i-1)*delx;  % Uniform nodes;
end

f = @(x) log(x);
g = @(x) 1.0/x;

for i=1:N+1
    f1(i) = f(x(i));
end

dfdx = zeros(N+1,1);

dfdx(1) = (f1(2)-f1(1))/delx;           % Forward Euler at Boundary Value
dfdx(N+1) = (f1(N+1)-f1(N))/delx;       % Backward Euler at Boundar Value

for i=2:N
    dfdx(i) = (f1(i+1)-f1(i-1))/(2*delx);   % Central differencing in the interior nodes
end

xx = linspace(a,b,50);
for i=1:length(xx)
    g1(i) = g(xx(i));
end
```

```
figure(1);
plot (x,dfdx,'b-o'); hold on;
plot (xx,g1,'r-');
legend('Using FD', 'Analytical');
fs = 'fontsize'; set (gca,fs,11);
print -depsc 'deriv_logx_uni.eps'
```

```
Warning: Imaginary parts of complex X and/or Y arguments ignored
```

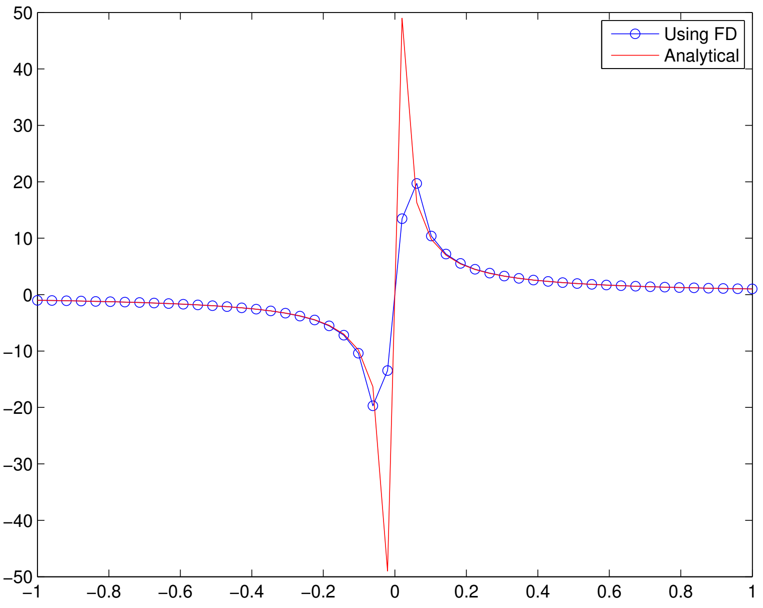

Figure 6: Derivative of log(x) using Uniform Node Distribution

Evaluating derivative of $cos(x)$ using LGL Differentiation Matrix

The Matlab code below evaluates the derivative using the differentiation matrix with collocation at the Legendre-Gauss-Lobatto nodes.

```
clear all; close all; clc;

N = 20;               % Polynomial degree;

% legDc.m
% Computes the Legendre differentiation matrix with collocation at the
% Legendre-Gauss-Lobatto nodes.

[x,D] = legDc(N-1);

f = @(x) cos(x);              % The function is f(x) = cos(x)
g = @(x) -sin(x) ;            % g = f' , i.e derivative of f

a = -pi; b = pi;
tau = 0.5*( (b-a)*x + (b+a));     % Scaling of the function

Xdot = 2/(b-a)*D*f(tau);      % derivative at discrete points

xx = linspace(a,b,50);
n  = length(xx);
```

```
fdot = zeros(n,1);

for i = 1 : n
    fdot(i) = g(xx(i));
end

plot(tau,Xdot,'b-o'); hold on;
plot(xx,fdot,'r-'); axis equal;
legend('Using Dmat', 'Analytical');
fs = 'fontsize'; set (gca,fs,12);
print -depsc 'deriv_cosx.eps'
```

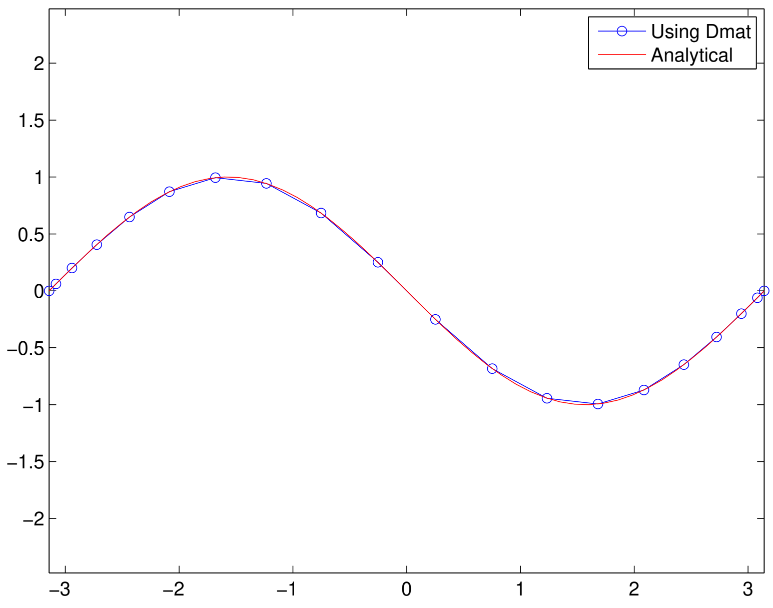

Figure 7: Derivative of cos(x) in the interval $[-\pi, \pi]$ using LGL nodes

Evaluating derivative of $\cos(x)$ using uniform Node Distribution

The matlab code below evaluates the derivative of $cos(x)$ between $-\pi$ and π using the finite difference approach using the uniform nodes distribution. Though a Differentiation Matrix using finite difference method can be obtained but the code is written using simple approach. At the boundaries, the forward Euler and backward Euler of first order and in the nodes in between the central difference scheme is used.

```
clear all; close all; clc;

N=20;               % No. of Nodes
a=-pi; b=pi;        % Boundary value;
delx = (b-a)/N;

for i=1:N+1
    x(i) = a + (i-1)*delx;              % Uniform nodes.
end

f = @(x) cos(x);
g = @(x) -sin(x);

for i=1:N+1
    f1(i) = f(x(i));
end
```

```
dfdx = zeros(N+1,1);

dfdx(1) = (f1(2)-f1(1))/delx;           % Forward Euler at Boundary Value
dfdx(N+1) = (f1(N+1)-f1(N))/delx;       % Backward Euler at Boundar Value

for i=2:N
    dfdx(i) = (f1(i+1)-f1(i-1))/(2*delx);   % Central differencing in the interior nodes
end

figure(1);
plot (x,dfdx,'r-+'); hold on;

xx = linspace(a,b,50);
plot (xx,g(xx),'b-');
legend('Using FD', 'Analytical');
fs = 'fontsize'; set (gca,fs,12);
print -depsc 'deriv_cosx_uni.eps'
```

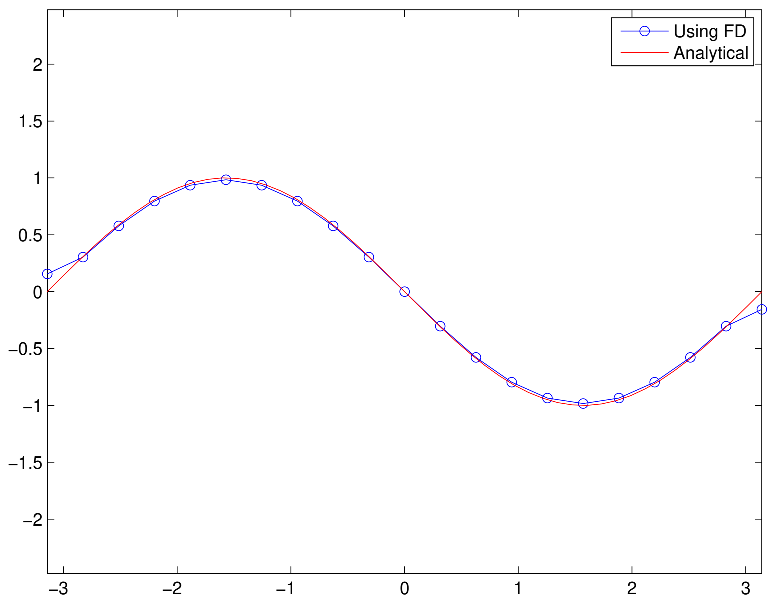

Figure 8: Derivative of cos(x) in the interval $[-\pi, \pi]$ using Uniform Node Distribution

Comment on the Derivatives

To have better appreciation the derivatives of $ln(x)$ and $\cos(x)$ obtained is given below.

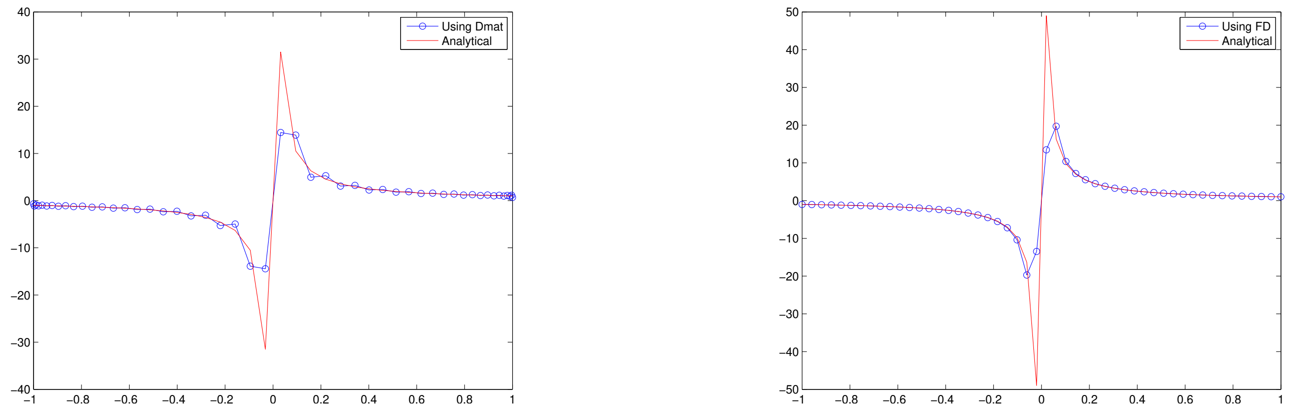

Figure 9: Derivative of $ln(x)$ in the interval [-1, 1]

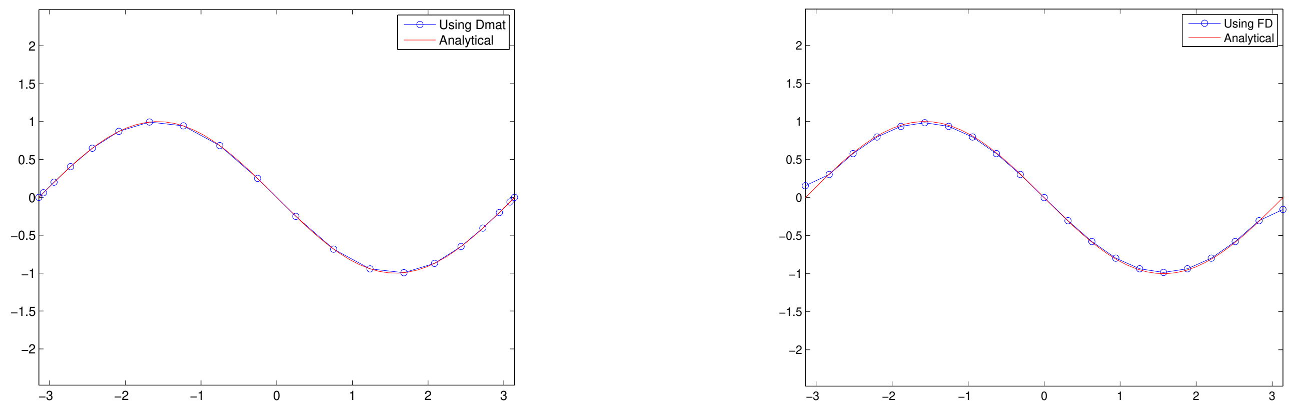

Figure 10: Derivative of $\cos(x)$ in the interval $[-\pi,\, \pi]$

The total number of nodes used for the derivatives among the Differentiation matrix and the Finite Difference method are the same. However, the Legendre-Gauss-Lobatto nodes are used to obtain the differentiation matrix. Whereas uniform node distribution is used in the finite difference approach. As can be seen, both the methods are reasonably corroborating well except near the singularity. With regard to the derivative of the $\cos(x)$ in the interval $[-\pi,\, \pi]$, it can be observed that the LGL nodes differentiation matrix is corroborating well with the analytic value whereas the FD approach do not match well in the boundaries. It was also observed that when LGL nodes are used even for FD approach, the derivative was matching well with the analytical value even in the boundaries.