Lecture - 01

Date - 03.07.19
Wednesday

Legendre polynomial is define as

$$P_0(x) = 1$$

$$P_1(x) = x$$

with $n P_n(x) = (2n-1) x P_{n-1}(x) - (n-1) P_{n-1}(x)$

Write a function double legendre (double x, int n) that recursively calculates the legendre polynomial of order n at some point x.

Plot the first 4 legendre polynomial for $x \in [-1,1]$ with an increament of 0.01 and save the plot as "legendre.png".

Solution:

```
# include <iostream>
# include <fstream>
using namespace std;
double legendre (double x, int n)
{
 if (n==0)
 return 1;
 else if (n==1)
 return x;
 else
 return ((2*n-1)*x* legendre (x, n-1) - (n-1)*
 legendre (x, n-2)) /n;
}
```

Esonix

Scanned by CamScanner

```
int main()
{
ofstream fout("l.dat");
for(double x=-1; x<=1; x+=0.01)
{
fout << x << "  " << legendre(x,0) << "  " <<
legendre(x,1) << "  " << legendre(x,2) << endl;
}
return 0;
}
```

Scanned by CamScanner

Hermite polynomial is defined as,

$$H_0(x) = 1$$

$$H_1(x) = 2x$$

with $H_n(x) = 2x H_{n-1}(x) - 2(n-1) H_{n-2}(x)$

Write a function double hermite (double x, int n) that recursively calculates the Hermite polynomial of order n at some point x.

Plot the first 4 hermite polynomial for $x \in [-3, 3]$ with an increament of 0.01 and save the plot as "Hermite.png".

Solution:

```
# include <iostream>
# include <fstream>
# include <cmath>
using namespace std;
double hermite (double x, int n)
{
if (n==0)
return 1;
else if (n==1)
return 2*x;
else
return (2*x*hermite (x,n-1) - 2*(n-1)*hermite (x, n-2));
}
```

Esonix®

Scanned by CamScanner

```
int main()
{
ofstream fout("h.dat");
for (double x=-3; x<=3; x+=0.01)
{
fout << x << " " << hermite(x,0) << " " << hermite(x,1)
<< hermite(x,2) << " " << hermite(x,3) << endl;
}
return 0;
}
```

Scanned by CamScanner

lecture - 02

Date - 08.07.19
Monday

Q A series expansion for Gauss's error function is given by

$$\text{erf}(x) = \frac{2e^{-x^2}}{\sqrt{\pi}} \sum_{n=0}^{\infty} \frac{2^K x^{2K+1}}{(2K+1)!!}$$

where, $(2K+1)!! = 1.3.5 \ldots (2K+1)$ is double factorial.

(a) Define a C++ function double gerf (double x, int n) that evaluates the series for the error function up to the nth term in the series for a given value of x.

(b) Plot your function gerf(x, n) with n = 20 for $x \in [-2, 2]$ with an increament of 0.001 and save the plot as "error.png".

Solution:

```
# include <iostream>
# include <fstream>
# include <cmath>
using namespace std;
double dfact (int n)
{
double prod = 1.0;
for (int i=1; i<=n; i++)
{
prod *= i;
}
return prod;
}
```

Scanned by CamScanner

```
double gerf (double x, int n)
{
double sum = 0.0;
for (int k=0; k<=n; k++)
{
sum += (pow(2,k)* pow(x, 2*k+1))/dfact(2*k+1);
double a = (2* exp(-x*x)) / sqrt (M_PI);
}
return sum*a;
}
int main()
{
ofstream fout ("g.dat");
for (double x=-2; x<=2; x+=0.001)
{
fout <<x<<"  " << gerf (x,20) << endl;
}
return 0;
}
```

Scanned by CamScanner

The incomplete gamma function are defined as

$$\gamma(n,x) = (n-1)!\left(1 - e^{-x}\sum_{s=0}^{n-1}\frac{x^s}{s!}\right)$$

$$\Gamma(n,x) = (n-1)!\, e^{-x}\sum_{s=0}^{n-1}\frac{x^s}{s!}$$

(a) Write two C++ function double gamma1 (int n, double x) and double gamma2 (int n, double x), that will return the values of gamma function by using the above two definitions.

(b) The potential produced by a hydrogen electron is given by

$$V(r) = \frac{q}{4\pi\epsilon_0 a_0}\left\{\frac{1}{2r}\gamma(3, 2r) + \Gamma(2, 2r)\right\}$$

Write a C++ function double potential (double r) that evaluates the potential by using the above definition. Plot $V(r)$ vs r for $r \in [0,4]$ with an increament of 0.01. Take $\frac{q}{4\pi\epsilon_0 a_0} = 1$ and

$$V(r) = \frac{1}{4\pi\epsilon_0 a_0}\cdot\frac{1}{r}$$

Solution:

```
#include <iostream>
#include <fstream>
#include <cmath>
using namespace std;
double dfact(int n)
{
double prod = 1.0;
for (int i=1; i<=n; i+=1)
{
prod *= i;
}
```

Scanned by CamScanner

```
return prod;
}
double gamma1 (int n, double x)
{
double sum = 0.0;
for (int s=0; s<=n; s++)
{
sum += (pow(x,s)) / dfact (s);
}
return dfact (n-1) * (1 - exp(-x) * sum);
}
double gamma2 ( int n, double x)
{
double sum = 0.0;
for (int s=0; s <= n, s++)
{
sum += (pow( x,s)/ dfact (s));
}
return dfact (n-1) * exp(-x) * sum;
}
double potential (double r)
{
return {(1/(2*r))*gamma1( 3, 2*r)*gamma2( 2, 2*r)}
}
int main ()
{
ofstream fout ("gamma.dat");
for (double r=0; r <=4 ; r+= 0.001)
{
fout << r << " " << potential (r) << endl;
}
return 0;
}
```

Scanned by CamScanner

4 An object of mass m is dropped from a height h, experiences to air resistance and is acted upon by gravity after a time t (sec) its height is

$$h(t) = h_0 - \frac{mg}{K} t + \frac{m^2 g}{K^2} (1 - e^{-Kt/m})$$

$g = 9.8\ ms^{-1}$, $K = 0.1\ Nms^{-1}$, $h = 300\ m$, $m = 0.25\ kg$

plot $h(t)$ Vs t for $t \in [0, 20]$. From the plot, find the time taken for the object to hit the ground.

Solution:

```
# include <iostream>
# include <fstream>
# include <cmath>
using namespace std;
double g = 9.8;
double K = 0.1;
double h0 = 300;
double m = 0.25;
double height (double t)
{
double s;
s = h0 - ((m*g*t)/K) + (m*m*g)/(K*K) * (1- exp(-K*t/m))
return s;
}
```

Scanned by CamScanner

```
int main()
{
ofstream fout("h.dat");
for(double t=0; t<=20; t+=0.01)
{
fout << t << " " << height(t) << endl;
}
return 0;
}
```

Scanned by CamScanner

Lecture-03

Date - 10.07.19
Wednesday

The Bernoulli numbers for even integeres $m = 2n$ are defined by the following series representation.

$$B_m = B_{2n} = \frac{(-1)^{n-1}\, 2\, (2n)!}{(2\pi)^{2n}} \sum_{p=1}^{\infty} p^{-2n}$$

$m = 2, 4, 6$ or $m = 2n$ with $n = 1, 2, 3 \ldots$

Write a C++ function double bernoulli (int n) then calculate the Bernoulli number using the first 20 terms in the series.

$$\tan(x) = \sum_{n=0}^{\infty} (-1)^{n-1}\, 2^{2n}\, (2^{2n}-1)\, B_{2n}\, x^{2n-1} / 2n!$$

Define a function double mytan (double x, int N) that will call the function bernoulli() to evaluate tan(x). Plot your function mytan() and the function tan() defined in cmath.

Solution:

```
# include <iostream>
# include <fstream>
# include <cmath>
using namespace std;
double fact (int n)
{
if (n==0)
return 1;
else
return n * fact (n-1);
}
double bernoulli (int n)
```

Esonix®

Scanned by CamScanner

```
{
double sum = 0;
for (int p=1; p<=20; p++)
{
sum += pow (p, (-2*n));
}
return (pow ((-1), n-1)*2*fact (2*n)*sum) / pow (2*M_PI, 2*n
}
double mytan (double x, int N)
{
double sum1 = 0;
for (int n=0; n<=N; n++)
{
sum1 += (pow(-1, n-1)*pow(2, 2*n)*(pow(2, 2*n)-1)*bernoulli(
* pow( x, 2*n-1)) / (fact (2*n));
}
return sum1;
}
int main ()
{
ofstream fout ("lo.dat");
for (double x=-M_PI/3; x<M_PI/3; x+=0.001)
{
fout <<x << " " << mytan (x,20) << " " << tan (x)<<end
}
return 0;
}
```

Scanned by CamScanner

$\int_0^{\pi/2} \frac{1}{1+\sin x}\,dx$

Use trapezium, Simpson 1/3 and 3/8 to evaluate the integral.

$$I = \int_a^b f(x)\,dx$$

Trap: $I = \frac{h}{2}\left[f(a) + 2\sum_{K=1}^{N-1} f(a+Kh) + f(b)\right]$

Simp 1/3: $I = \frac{h}{3}\left[f(a) + 4\sum_{K=1,3,5\ldots N-1} f(a+Kh) + 2\sum_{K=2,4,6\ldots N-2} f(a+Kh) + f(b)\right]$

Simp 3/8: $I = \frac{3h}{8}\left[f(a) + 3\sum_{K=1,4,7\ldots} f(a+Kh) + 3\sum_{K=2,5,8\ldots}^{N-1} f(a+Kh) + 2\sum_{K=3,6,9}^{N-3} f(a+Kh) + f(b)\right]$

Solution:

```
#include <iostream>
# include <fstream>
# include <cmath>
using namespace std;
double integ (double x)
{
return 1/(1+sin(x));
}
double trap (double a, double b, int n)
{
double h=(b-a)/n;
double sum=0;
for (int i=1; i< n; i++)
{
sum+= integ (a+i*h);
}
```

Scanned by CamScanner

```
return (h/2)*(integ(a) + 2*sum + integ(b));
}
double simp13(double a, double b, int n)
{
double h=(b-a)/n;
double sum1=0;
double sum2=0;
for(int k=1; k<n; k+=2)
sum1+=integ(a+k*h);
for(int k=2; k<n; k+=2)
sum2+=integ(a+k*h);
return (h/3)*(integ(a)+4*sum1+2*sum2+integ(b));
}
double sim38(double a, double b, int n)
{
double h=(b-a)/n;
double sum1=0;
double sum2=0;
double sum3=0;
for(int k=1; k<n; k+=3)
sum1+=integ(a+k*h);
for(int k=2; k<n; k+=3)
sum2+=integ(a+k*h);
for(int k=3; k<n; k+=3)
sum3+=integ(a+k*h);
return ((3*h)/8)*(integ(a)+3*sum1+3*sum2+2*sum3+integ(b));
}
```

Scanned by CamScanner

```
int main()
{
ofstream fout ("t.dat");
fout <<" " << trap (0, M_PI/2, 120) << endl;
fout <<" " << simp13 (0, M_PI/2, 120) << endl;
fout <<" " << simp38 (0, M_PI/2, 120) << endl;
return 0;
}
```

Scanned by CamScanner

Q) $T(t) = 100\left(1 - \frac{2}{\sqrt{\pi}}\int_0^{8/\sqrt{t}} e^{-\tau^2}\, d\tau\right)$

Write a c++ programme to evaluate the integral. Let the function double temp(double T) that evaluates the temeperature at a time t.

Write a c++ program to save data for T(t) in the range $t \in [10, 200]$ with an increament of 0.01. Plot the data T vs t using gnuplot.

Solution:

```
#include <iostream>
# include <cmath>
# include <fstream>
using namespace std;
double time (double t, double tau)
{
return 100* (1- 2/sqrt (M_PI)* exp (- tau * tau));
}
double trap (double t)
{
double a=0;
double b= 8/ sqrt(t);
int n=100;
double h=(b-a)/n;
double sum=0;
for (int i=1; i<n ; i++)
{
sum += time (t, a + i* h);
}
return (h/2) * (time (t, a) + 2*sum + time (t, b));
}
```

Scanned by CamScanner

```
int main()
{
ofstream fout ("t.dat");
for (double t=10; t<=200; t+=0.01)
fout << t << "  " << trap(t) << endl;
return 0;
}
```

Scanned by CamScanner

Date - 17.07.19
Wednesday

An integral representation Gauss error function is given by

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$

(a) Define a function errint (double x) that evaluates the error function for the agreement x by using any suitable numerical technique

(b) Plot your function for $x \in [-2, 2]$ with an increament of 0.01,

Solution:

```
#include <iostream>
# include <fstream>
# include <cmath>
using namespace std;
double func (double t)
{
return (2/sqrt (M_PI)) * exp (-t * t);
}
double trap (double a, double b, int n)
{double h = (b-a)/n;
double sum = 0;
for (double i = 1; i <= n; i++)
{
sum += func (a + (i * h));
}
sum = h * (func (a) + func (b) + 2 * sum) / 2;
return sum;
}
int main ()
{
ofstream fout ("g.dat")
double value;
double t,
for (double x = -2; x <= 2; x += 0.01)
fout << x << " " << trap (0, x, 100) << endl;
return 0;
}
```

Scanned by CamScanner

4) The Gaussian curve is defined by

$$f(x) = \frac{1}{\sqrt{2\pi}} \exp(-x^2/2)$$

Write a c++ function mygaussian (double a) that evaluates the integral,

$$I(a) = \frac{1}{\sqrt{2\pi}} \int_{-a}^{a} \exp(-x^2/2)\, dx$$

Plot I(a) vs a for $a \in [-2, 2]$ and from the plot find the value of a for which $I(a) = 1/2$.

Solution

```
#include <iostream>
#include <fstream>
#include <cmath>
using namespace std;
double func (double x)
{
return (1/sqrt(2*M_PI)) * exp(-x*x/2);
}
double mygaussian (double p, double q, int n)
{
double h = (q-p)/n;
double sum = 0;
for (double i=1; i<=n; i++)
{
sum += func (p+(i*h));
}
```

Scanned by CamScanner

```
sum = h* (func (p) + func(q) + 2*sum)/2;
return sum;
}
int main ()
{
ofstream fout ("gc.dat");
doulele value;
doulele t;
for (doulele a=-2; a<=2; a+=0.01)
{
fout << a << "  " << abs (mygaussian (-a,a,100)) << endl;
}
return 0;
}
```

Scanned by CamScanner

The spherical Bessel's function have the integral representation,

$$J_n(z) = \frac{z^n}{2^{n+1} n!} \int_0^{\pi} \cos(z\cos\theta)(\sin\theta)^{2n+1} d\theta$$

(i) Define a function double bessel (int n, double z) that evaluates the spherical Bessel's functions by using the above definition.

(ii) Plot the first four Bessel's function using gnuplot.

Solution:

```
# include <iostream>
# include <fstream>
# include <cmath>
using namespace std;
double fact (int p)
{
double prod = 1;
for (int i=1; i<=p; i++)
{
prod *= i;
}
return prod;
}
double func (double theta, double z, double n)
{
return (pow(z,n)/(pow (2,n+1)*fact(n)))*cos (z*cos(theta))*
pow (sin (theta), (2*n)+1);
}
```

Esonix®

Scanned by CamScanner

```
double bessel (int n, double z)
{
double h, b, a, N;
a = 0;
b = M_PI;
N = 100;
h = (b-a)/N;
double sum = 0;
for (double i = 1; i < N; i++)
{
sum += func(a + i*h, z, n);
}
double su = h*(func(a, z, n) + func(b, z, n) + 2*sum)/2;
return su;
}
int main ()
{
ofstream fout ("sb.dat");
for (double z = 0; z <= 20; z += 0.01)
{
fout << z << " " << bessel(0, z) << " " << bessel(1, z) <
" " << bessel(2, z) << " " << bessel(3, z) << endl;
}
return 0;
}
```

Scanned by CamScanner

(b) The Fresnel integral of the second kind may be defined in terms of the spherical Bessel's function by

$$S(x) = \frac{1}{2}\int_0^x J_0(z)\sqrt{z}\,dz$$

(i) Define a function double fresnel (double x) using the above definition and the function bessel (n, z)

(ii) Plot $S(x)$ vs x for $x \in [0, 20]$

Solution:

```
#include <iostream>
#include <fstream>
#include <cmath>
using namespace std;
double fact (int r)
{
if (r==0)
return 1;
else
return r* fact (r-1);
}
double func (double theta, double z, double n)
{
return (pow (z,n)/(pow (2, n+1)*fact (n)))*cos (z*cos (theta))
* pow (sin (theta), (2*n)+1);
}
double bessel (int n, double z)
{
```

Esonix®

Scanned by CamScanner

```
double b=0;
double c = M_PI;
int m=100;
double h = (c-b)/m;
double sum = 0;
for (int i=1; i<m; i++)
{
sum += func(b+i*h, z, n);
}
return (h/2)*func(b,z,n) + 2*sum + func(c, z, n));
}
double func2 (double z)
{
return 0.5*bessel(0,z)*sqrt(z);
}
double fresnel (double x)
{
double b =0;
double c =x;
int m = 100;
double h =(c-b)/m;
double sum =0;
for (int i=1; i<m; i++)
{
sum += func2 (b+i*h);
}
return (h/2)*(fun2(b) + 2* sum + func2(c));
}
int main()
{
ofstream fout ("bl.dat");
ofstream fou ("b.dat");
for (double z=0; z<25; z +=0.01)
{fout << z<" " << bessel(0,z) << " " << bessel(1,z) <<" " <<
bessel(2,z) << " " << bessel(3,z) << endl;
}
for (double x=0; x<20; x+= 0.001)
{
fou << x << " " fresnel(x) << endl;
}return 0;
```

Scanned by CamScanner

lecture Monday

The Spherical Bessel's function

$$J_n(x) = \frac{x^n}{2^{n+1} n!} \int_0^{\pi} \cos(x\cos\theta)(\sin\theta)^{2n+1}\, d\theta$$

An analysis of the antenna radiation pattern for a system of circular aperture involves the function,

$$g(U) = \int_0^1 f(r)\, J_0(ur)\, r\, dr \; ; \; f(r) = 1 - r^2$$

Plot $g(U)$ vs U for $U \in [0, 25]$

Solution

```
# include <iostream>
# include <fstream>
# include <cmath>
double fact (int p)
{
double prod =1;
for (int i=1; i<=p; i++)
{
prod * = i;
}
return prod;
}
double func (double theta, double z, double )
{
return (pow (z,n)/(pow (2,n+1)*fact (n))) * cos (z*cos(theta
* pow (sin (theta), (2*n)+1);
}
double bessel (int n, double z)
```

Esonix®

Scanned by CamScanner

```
{
doulele h,b,a,N;
a=0;
b=M-PI;
N=100;
h=(b-a)/N;
doulele sum=0;
for (doulele i=1; i<N; i++)
{sum+= func (a+i*h, z, n);
}
doulele su = (h*(func(a; z, n) + f(b, z, n) + 2*sum))/2;
return su;
}
doulele func2 (doulele r)
{return (1-r*r)
}
doulele g (doulele r, doulele u)
{return func 2(r)* bessel 1 0, u*r) * r;
}
doulele rad (doulele u)
{doulele h,b,a,N;
a=0;
b=1;
N=100;
h=(b-a)/N;
doulele sum=0;
for (doulele j=1; j<=N; j++)
{sum+= g(a+j*h, u)
}
doulele sm = (h*(g(a,u) + g(b,u) + 2*sum))/2;
return sm
}
```

Scanned by CamScanner

```
int main()
{
ofstream fout ("ca.dat");
for (doulele u=0; u<=20; u+=0.01)
{
fout << u << " " << rad(u) << endl;
}
return 0;
}
```

Scanned by CamScanner

The complete elliptical integral of the first and second kind are defined respectively as.

$$K(m) = \int_0^{\pi/2} (1 - m\sin^2\theta)^{-1/2}\, d\theta$$

and $$E(m) = \int_0^{\pi/2} (1 - m\sin^2\theta)^{1/2}\, d\theta$$

Plot $K(m)$ vs m and $E(m)$ vs m for $m \in [0,1]$ in the same graph.

Solution:

```
# include <iostream>
# include <fstream>
# include <cmath>
using namespace std;
double K (double m, double theta)
{
return pow((1-m*sin(theta)*sin(theta)), -0.5);
}
double e (double m, double theta)
{
return pow((1-m*sin(theta)*sin(theta)), 0.5);
}
double func1 (double m)
{
double a=0;
double b=M_PI/2;
```

Scanned by CamScanner

```
7  doulele n = 100;
8  doulele h = (b-a)/n;
9  doulele sum = 0.0;
0  for (int i=1; i<n; i++)
   {
   sum + = K(m, (a+i*h));
3  }
4  return ((h/2) * K(m,a) + 2*sum + K(m,b)));
5  }
6  doulele func2 (doulele m)
7  {
8  doulele a = 0;
9  doulele b = M_PI/2;
30 doulele n = 100;
31 doulele h = (b-a)/n;
32 doulele sum = 0.0;
33 for (int i=1; i<n; i++)
34 {
35 sum + = e(m, (a+i*h));
36 }
37 return ((h/2) * (e(m,a) + 2*sum + e(m,b)));
38 }
```

Scanned by CamScanner

```
int main ()
{
ofstream fout ("k.dat");
for (double m=0; m<=1; m+=0.001)
 {
fout << m << " " << func1(m) << " " << func2(m)<<endl;
 }
return 0;
}
```

Scanned by CamScanner

The period of a simple pendulum ... large angle amplitude (θ_m) is given by

$$T = 4\left(L/g\right)^{1/2} K\left(\sin^2 \frac{\theta_m}{2}\right)$$

Plot T vs θ_m for $\theta_m \in [0, \pi/2]$. chose (L/g) such that as $\theta_m \to 0$ then $T = 1s$. Check values: $\theta_m = [10°, 50°, 90°]$

$T = [1.00193, 1.05033, 1.18250]$

$$\boxed{\begin{aligned} 1 &= 4\left(\frac{L}{g}\right)^{1/2} \pi/2 \\ \therefore \left(\frac{L}{g}\right)^{1/2} &= \frac{1}{2\pi} = \frac{1}{4\pi^2} \end{aligned}}$$

Solution:

```
# include <iostream>
# include <fstream>
# include <cmath>
using namespace std;
double K(double m, double theta)
{
return pow((1- m*sin(theta)*sin(theta)), -0.5);
}
double e(double m, double theta)
{
return pow((1- m*sin(theta)*sin(theta)), 0.5);
}
```

Scanned by CamScanner

```
doulele func1 (doulele m)
{
doulele a=0;
doulele b=M_PI/2;
doulele n=100;
doulele h=(b-a)/n;
doulele sum=0.0;
for (int i=1; i<n; i++)
{
sum += K(m, (a+i*h));
}
return ((h/2)*(K(m,a)+2*sum+K(m,b)));
}
doulele func2 (doulele m)
{
doulele a=0
doulele b=M_PI/2;
doulele n=100;
doulele h=(b-a)/n;
doulele sum=0.0;
for (int i=1; i<n; i++)
{
sum += e(m, (a+i*h));
}
```

Scanned by CamScanner

```
return ((h/2)*(e(m,a) + 2* sum + e (m, b)));
}
double f(double theta)
{
return 4* (1/(2 * M_PI)) * fun e1( pow(sin(theta/2)), 2));
}
int main()
{
ofstream fout ("f.dat");
for (double theta=0; theta<= M_PI/2; theta += 0.001)
{
fout << theta <<" " << f(theta) <<" "<<endl;
}
return 0;
}
```

Esonix®

Scanned by CamScanner

Lecture - 06.

Date - 21.08.19
Wednesday

Algorithm for Gaussian Quadrature method:

$$\int_a^b f(x)\,dx \simeq \sum_{k=1}^{N} f(x_k)\,\omega_k$$

x_k = sample points
ω_k = weight.

x_k = sample points between -1 and +1 by using legendre polynomial.

Algorithm:

(1) Find the sample point x_k in the zeroes of the legendre polynomial of degree n.

$$P_n(x) = 0$$
$$\therefore x = x_k \quad (x_k = x_1, x_2 \ldots x_n)$$

(2) Find weights at each of these sample points by using,

$$\omega_k = \left[\frac{2}{(1-x^2)}\left(\frac{dP_n}{dx}\right)^{-2}\right]_{x=x_k}$$

(3) Rescale the values by using the following equations,

$$x'_k = \frac{1}{2}(b-a)\,x_k + \frac{1}{2}(b+a)$$
$$\omega'_k = \frac{1}{2}(b-a)\,\omega_k$$

where a and b are the lower and upper limits of integration respectively.

Scanned by CamScanner

(4) Use these calculated values x_k' and ω_k' into the following equation to get the result of integration.

$$\int_a^b f(x)\,dx \simeq \sum_{k=1}^{N} \omega_k' f(x_k')$$

Zeroes of nth order legendre polynomial:

Tricomi's asymptotic approximation for the kth root of the nth polynomial is,

$$x_{n,k} \approx \left(1 - \frac{1}{8n^2} + \frac{1}{8n^3}\right) \cos\left\{\frac{\pi(4k-1)}{4n+2}\right\}$$

$$\boxed{x_{n,k} \approx \cos\left\{\frac{\pi(k-0.25)}{n+0.5}\right\}}$$

The roots can be improved by using Newton-Raphson method.

Newton-Raphson method

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Example 14:

Esonix®

Scanned by CamScanner

Lecture - 07

Date - 26.08.19
Monday

Case - I:

For infinite limit,

$$I = \int_0^\infty f(x)\,dx$$

$$= \int_0^1 f\left(\frac{z}{1-z}\right) \frac{dz}{(1-z)^2}$$

Let,

$$z = \frac{x}{1+x}$$

$$x = \frac{z}{1-z}$$

$$dx = \frac{dz}{(1-z)^2}$$

x	0	∞
z	0	1

Case - II:

$$I = \int_a^\infty f(x)\,dx$$

$$= \int_0^1 f\left(\frac{z}{1-z} + a\right) \frac{dz}{(1-z)^2}$$

Case - III:

$$I = \int_{-\infty}^a f(x)\,dx$$

$$= \int_{-1}^0 \frac{-1}{(1+z)^2} f\left(\frac{-z}{1+z} + a\right) dz$$

Case - IV:

$$I = \int_{-\infty}^{+\infty} f(x)\,dx$$

$$= \int_{-1}^{+1} \frac{1+z^2}{(1-z^2)^2} \cdot f\left(\frac{z}{1-z^2}\right) dz$$

$$I = \int_0^\infty f(x)\,dx$$

Example 17:

$$I = \int_0^\infty e^{-x^2}\,dx$$

$$= \int_0^1 f\left(\frac{z}{1-z}\right) \frac{dz}{(1-z)^2}$$

$$= \int_0^1 e^{-(z/1-z)^2} \frac{dz}{(1-z)^2}$$

Example 18:

Scanned by CamScanner

lecture-08

Date- 28.08.19
Wednesday

Newton-Raphson method:

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})} = x_n - h;$$

Here, $h = f(x_{n-1}) / f'(x_{n-1})$

Secent method:

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f(x_{n-1}) - f(x_{n-2})}$$

Q Find the root of the following equation by bisection, newton and secent method corrected upto 5 decimal place where $f(x) = 0$

(a) $xe^x = 1$

(b) $\cos x - xe^x = 0$

(c) $x = e^{-x}$

(d) $x \sin x + \cos x$

Solution:

For bisection method:

```
1.#include <iostream>
2.# include <cmath>
3.# include <fstream>
4.# include <iomanip>
5.# define e 0.00001
6.using namespace std;
7. double f1(double x)
8. {
9. return x* exp(x) - 1;
10. }
```

Esonix®

Scanned by CamScanner

```
double f2(double x)
{
return cos(x) - (x* exp(x));
}
double f3(double x)
{
return x - exp(-x);
}
double f4(double x)
{
return x*sin(x) + cos(x);
}
double bisection (double a, double b)
{
double c = 0.0;
while (abs(a-b) >= e)
{
c = (a+b)/2.0;
if (f1(c) == 0)
break;
else if (f1(a) * f1(c) < 0)
b = c;
else a = c;
}
```

Scanned by CamScanner

```
35. return c;
36. int main()
37. {
38. cout << "root of equation by bisection:" << bisection
39. (4,-2) << endl;
40. return 0;
41. }
```

Esonix®

Scanned by CamScanner

For bisection, newton and secent method altogether:

```
# include <iostream>
# include <cmath>
# include <fstream>
# include <iomanip>
# define e 0.0001
using namespace std;
double f1 (double x)
{
return x*exp(x) -1;
}
double f2 ( double x)
{
return cos (x) - ( x* exp(x));
}
double f3 (double x)
{
return x - exp(-x);
}
double f4 ( double x)
{
return x* sin (x) + cos (x);
}
double bisection (double a, double b)
{
double c = 0.0;
while (abs (a - b) >= e).
{
c = (a+b)/2.0;
```

Scanned by CamScanner

```
if (f1(c)==0)
break;
else if (f1(a)*f1(c) <0)
b=c;
else a=c;
}
return c;
}
double df1 (double x)
{
return exp(x)*(1+x);
}
double newton (double x)
{
double h = f1(x)/ df1(x);
while (abs(h)>=e)
{
x=x-h;
h=f1(x)/df1(x);
return x;
}
}
```

Esonix®

Scanned by CamScanner

```
int main()
{
cout << "root of equation by bisection :"<<
bisection (4,-2) << "    " << "root of equation
by newton : " << newton(5) << endl;
return 0;
}
```

Scanned by CamScanner

2. $\beta = \tan \beta$

Find the root?

Solution:

```
# include <iostream>
# include <fstream>
# include <cmath>
# include <iomanip>
# define e 0.00001
using namespace std;
double f1(double b)
{
return b;
}
double f2(double b)
{
double tan(b);
}
int main ()
{
ofstream fout ("optics.dat");
for (double b=-15; b<=15; b+=0.01)
{
fout <<b<<" "<<b<<" "<<tan(b)<<endl;
}
```

Esonix®

Scanned by CamScanner

```
22. return 0;
23. }
```

Scanned by CamScanner

Find the square root of a given number n by Newton-Raphson method;

$$x^2 = n$$

$$\therefore x = \sqrt{n}$$

Solution:

If we rearrange the equation as, $x^2 - n = 0$ the programme will be come:

```
1. # include <iostream>
2. # include <cmath>
3. # include <fstream>
4. # define e 0.0001
5. using namespace std;
6. double f (double x, double n)
7. {
8. return (x*x) - n;
9. }
10. double df (double x, double n)
11. {
12. return 2*x;
13. }
14. double bisection (double a, double b)
15. {
16. double n;
17. double c = 0.0;
```

Esonix®

Scanned by CamScanner

```
while (abs(a-b)>=e)
{
c=(a+b)/2.0;
if (f(c,n)==0)
break;
else if (f(a,n)*f(c,n)<0)
b=c;
else a=c;
}
return c;
}
double newton (double x, double n)
{
double h=f(x,n)/df(x,n);
while (abs(h)>=e)
{
x=x-h;
h=f(x,n)/df(x,n);
}
return x;
}
int main()
{
cout << "Root of the newton raphson: "<<newton(1,19)<<
endl;
return 0;
}
```

Scanned by CamScanner

If we arrange the equation as $x = \sqrt{n}$ the programe become

```
# include <iostream>
# include <fstream>
# include <cmath>
# include <iomanip>
# define e 0.0001
using namespace std;
double x (double n)
{
return sqrt(n);
}
double dx (double n)
{
    return 1/(2* sqrt (n));
}
double newton (double n)
{
double h = x(n) / dx(n);
while (abs(h) > e)
{
n = n - h;
h = x(n) / dx(n);
return n;
}
```

Esonix®

Scanned by CamScanner

```
}
int main()
{
cout << "result:" << newton (5) << endl;
return 0;
}
```

Scanned by CamScanner

$\beta = \frac{\sin\beta}{\cos\beta} \rightarrow \sin\beta - \beta\cos\beta = 0$

Solution.

```
#include <iostream>
#include <fstream>
#include <cmath>
#include <iomanip>
# define e 0.0001.
using namespace std;
double f1 (double x)
{
return sin(x) - (x*cos(x));
}
double df1 (double x)
{
return (cos(x) + x* sin(x) - cos(x));
}
double newton (double x)
{
double h = f1(x) / df1(x);
while (abs (h)>= e)
{
x = x - h;
h = f1(x) / df1(x);
return x;
```

Esonix®

Scanned by CamScanner

```
}
}
int main ()
{
cout << " result of the equation: " << newton(4)
<<endl;
return 0;
}
```

Scanned by CamScanner

lecture - 09

Date - 02.09.19
Monday

In a certain quantum mechanical problem of a particle moving in a potential wall, the eigen energies are given by.

$$-\cot(x) = \frac{1}{x}\sqrt{2mV_0(a^2/\hbar^2) - x^2}$$

where, $2mV_0(a^2/\hbar^2) = 10.498597$

$$F = 0.9525082255x^2 - 10$$

Plot $-\cot(x)$ and $\frac{1}{x}\sqrt{2mV_0(a^2/\hbar^2) - x^2}$ in the same graph. Find the energy eigenvalue using NR method. Use the plot to select an inertial game.

Solution:

```
# include <iostream>
# include <fstream>
# include <cmath>
# define e 0.0001
# define K 10.498597
using namespace std;
double f1 (double x)
{
return (1/x) * sqrt (K-(x*x));
}
double f2 (double x)
{
return - (1/tan(x));
}
```

Esonix®

Scanned by CamScanner

```
double f(double x)
{
return-(1/tan(x)) - (1/x)*sqrt(K-(x*x));
}
double df(double x)
{
return (1/pow(sin(x),2)) + (K/(x*x*sqrt(K-(x*x))));
}
double eigen(double x)
{
while (abs(h)>=e)
{
x=x-h;
h=f(x)/df(x);
}
return x;
}
int main()
{
ofstream fout("g.dat");
double x;
for (x=-20; x<=20; x+=0.01)
{
fout<<x<<" "<<f2(x)<<" "<<f1(x)<<endl;
}
return 0;
}
```

Scanned by CamScanner

Lecture: 10 ODE Date - 11.09.19 Wednesday

$\frac{dy}{dt} + 2y = 2 - e^{-4t}$. $y(0) = 1$

The exact solution $1 + \frac{1}{2}e^{-4t} - \frac{1}{2}e^{-2t}$. Plot $y(t)$ vs t for both the numerical solution and the exact solution in the same graph

Hint: $\frac{dy}{dt} = f(t, y)$. $y_{n+1} = y_n + h\, f(y_n, t_n)$

Solution:

```
# include <iostream>
# include <fstream>
# include <cmath>
using namespace std;
double f1( double y, double t)
{
return 2 - exp(- 4*t) - (2*y);
}
double f2( double t)
{
Return 1+(0.5 * exp(-4*t)) - (0.5 * exp(-2*t));
}
void euler (double y0, double t0)
{
ofstream fout ("0.dat")
double h = 0.001;
double y = y0;
for (double t = t0; t<=10; t+=h)
{
```

Esonix®

Scanned by CamScanner

```
cout << t << " " << y << " " << f2(t) << endl;
y += h * f1(y, t);
}
}
int main()
{
euler(1, 0);
}
```

Scanned by CamScanner

$$\frac{d^2\theta}{dt^2} + g/L\,\theta = 0 \Rightarrow \boxed{\frac{d\theta}{dt} = \omega \Rightarrow \frac{\theta(t+dt) - \theta(t)}{dt} = \omega}$$

$$\boxed{\Rightarrow \theta(t+dt) = \theta(t) + \omega(t) * dt}$$

$$\therefore \frac{d\omega}{dt} + g/L\,\theta = 0 \Rightarrow \frac{d\omega}{dt} = -g/L\,\theta \Rightarrow \omega(t+dt) = \omega t - (g/L) * \theta * dt$$

Plot θ vs t and ω vs t in the same graph. where $\theta(0)=0, \omega(0)=1$
$L = 1\,m$

Solution:

```
1. # include <iostream>
2. # include <fstream>
3. # include <cmath>
4. using namespace std;
5. void euler (double t0, double theta 0, double omega 0)
6. {
7. double h=0.001, g=9.8, l=1.0;
8. double theta = theta 0, omega = omega 0;
9. ofstream fout ("02.dat")
10. for (double t=t0; t<=10; t+=h)
11. {
12. fout << t << " " << theta << " " << omega << endl;
13. omega -= (g/l)* theta *h;
14. }
15. }
16. int main()
17. {
19. euler (0,0,1);
20. return 0;
21. }
```

Esonix®

Scanned by CamScanner

RK4 method:

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$k_1 = hf(t_i, y_i)$$

$$k_2 = hf(t_i + h/2, y_i + \frac{k_1}{2})$$

$$k_3 = hf(t_i + h/2, y_i + \frac{k_2}{2})$$

$$k_4 = hf(t_i + h, y_i + k_3)$$

$\frac{dy}{dt} = t\sqrt{y}$, $y(0) = 1$

exact solution is, $y = (t^2+4)^2/16$. Plot them in a graph.

Solution:

```
#include <iostream>
# include <fstream>
# include <cmath>
using namespace std;
double f1(double t, double y)
{
return t*sqrt(y);
}
double f2(double t0, double y0)
{
return pow((t*t+4), 2)/16;
}
void rk4(double t0, double y0)
{
ofstream fout("rk.dat");
```

Scanned by CamScanner

```
double h = 0.001;
double y = y0;
double k1, k2, k3, k4;
for (double t = t0; t <= 10; t += h)
{
cout << t << " " << y << " " << f2(t) << endl;
k1 = h* f1(t, y);
k2 = h* f1 (t + (h/2), y + (k1/2));
k3 = h*f1 (t+(h/2) , y + (k2/2));
k4 = h* f1 (t+h, y+k3);
y += (k1 + 2*k2 + 2*k3 + k4)/6;
}
}
int main()
{
rk4 (0,1);
}
```

Esonix®

Scanned by CamScanner

$$\frac{d^2x}{dt^2} = -\frac{kx}{m} \Rightarrow \frac{dv}{dt} = -kx/m \qquad \boxed{\frac{dx}{dt} = v_x}$$

Hint: Void Rx (double t0, double x0, double v0)

$k_1 = h * f(t, x, v)$

$l_1 = h * g(t, x, v)$

$k_2 = h * f((t + h/2), x + (k1/2); v + (l1/2));$

$l_2 = h * g((t + h/2), x + (k1/2), v + (l1/2));$

Solution:

```
1. # include <iostream>
2. # include <cmath>
3. # include <fstream>
4. using namespace std;
5. double f1(double t, double x, double v)
6. {
7. return v;
8. }
9. double g (double t, double x, double v)
10. {
11. double k=150;
12. double m=150;
13. return (-k*x)/m;
14. }
15. void rk4 (double t0; double x0, double v0)
16. {
17. ofstream fout ("0.dat");
18. double h=0.001;
19. double x=x0, v=v0;
20. double k1, k2, k3, k4, l1, l2, l3, l4;
21. for (double t=t0; t<=100; t+=h)
22. {
23. fout << t << " " << x << " " << v << endl;
```

Scanned by CamScanner

```
k1 = h*f1(t,x,v);
l1 = h*g(t,x,v);
k2 = h*f1(t+(h/2.0), x+(k1/2.0); v+(l1/2.0));
l2 = h*g(t+(h/2.0), x+(k1/2.0); v+(l1/2.0));
k3 = h*f1(t+(h/2.0), x+(k2/2.0); v+(l2/2.0));
l3 = h*g(t+(h/2.0), x+(k2/2.0); v+(l2/2.0));
k4 = h*f1(t+h, x+k3; v+l3);
l4 = h*g(t+h, x+k3, v+l3);
x+= (k1+2*k2+2*k3+k4)/6;
v+= (l1+2*l2+2*l3+l4)/6;
}
}
int main()
{
rk4(0, 0.5, 4);
}
```

Esonix®

Scanned by CamScanner