Practical-1(a)

Sum of 10 natural numbers

```
s=0
for i=1:10
    s=s+i
end

%Average of first 10 natural numbers
s=0
for i=1:10
    s=s+i;
    t=s/i
end
```

```
s =

     0


s =

     1


s =

     3


s =

     6


s =

    10


s =

    15


s =

    21
```

```
s =

    28

s =

    36

s =

    45

s =

    55

s =

     0

t =

     1

t =

    1.5000

t =

     2

t =

    2.5000

t =

     3

t =

    3.5000
```

```
t =

     4

t =

    4.5000

t =

     5

t =

    5.5000
```

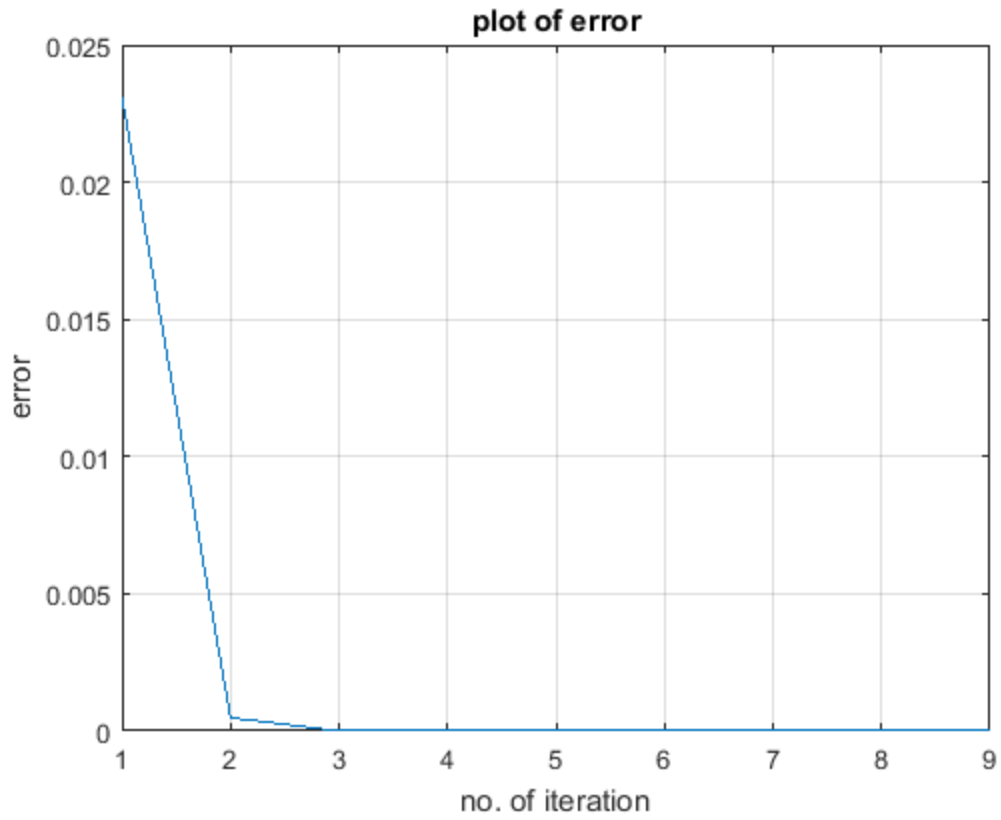
Published with MATLAB® R2015b

Practical no-2(a)

Program to find root of eqn using newton raph-son method and error analysis

```
f=@(x) x^3-x-1;      % given equation
df=@(x) 3*x^2-1;       % derivative of given equation
a=1;                   % initial value of a
b=2;                   % initial value of b
if f(a)*f(b)>0         % condition of existence
    disp('wrong choice')
else
    x0=(a+b)/2;
    for i=1:9          % iteration
        c=x0-(f(x0)/df(x0));      % newton raphson
        fprintf('iteration no %d,root %f \n',i,c);
        x0=c;
    end
    fprintf('root of given equation is %f',c);
end
a=1;
b=2;
    r=c;
    x0=(a+b)/2;
    for i=1:9
        c=x0-(f(x0)/df(x0));
        err(i)=abs(c-r);
        x0=c;
    end
    plot(err);
    title('plot of error');
    xlabel('no. of iteration');
    ylabel('error');
    grid on;
```

```
iteration no 1,root 1.347826
iteration no 2,root 1.325200
iteration no 3,root 1.324718
iteration no 4,root 1.324718
iteration no 5,root 1.324718
iteration no 6,root 1.324718
iteration no 7,root 1.324718
iteration no 8,root 1.324718
iteration no 9,root 1.324718
root of given equation is 1.324718
```



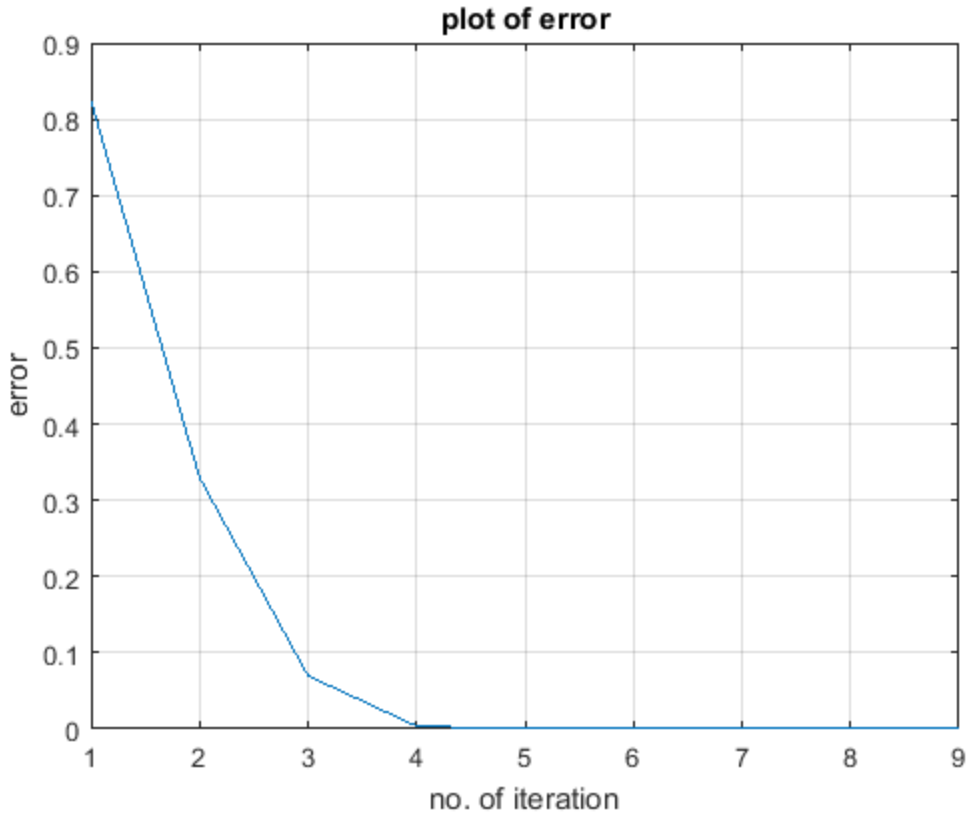
Published with MATLAB® R2015b

Practical no-2(b)

Program to find root of eqn using newton raph-son method and error analysis

```
f=@(x) x*exp(x)-cos(x);      % given equation
df=@(x) exp(x)*(x+1)+sin(x);       % derivative of given equation
a=0;                   % initial value of a
b=4;                   % initial value of b
if f(a)*f(b)>0         % condition of existence
    disp('wrong choice')
else
    x0=(a+b)/2;
    for i=1:9          % iteration
        c=x0-(f(x0)/df(x0));      % newton raphson
        fprintf('iteration no %d,root %f \n',i,c);
        x0=c;
    end
    fprintf('root of given equation is %f',c);
end
a=0;
b=4;
    r=c;
    x0=(a+b)/2;
    for i=1:9
        c=x0-(f(x0)/df(x0));
        err(i)=abs(c-r);
        x0=c;
    end
    plot(err);
    title('plot of error');
    xlabel('no. of iteration');
    ylabel('error');
    grid on;
```

```
iteration no 1,root 1.341569
iteration no 2,root 0.847701
iteration no 3,root 0.587557
iteration no 4,root 0.521581
iteration no 5,root 0.517770
iteration no 6,root 0.517757
iteration no 7,root 0.517757
iteration no 8,root 0.517757
iteration no 9,root 0.517757
root of given equation is 0.517757
```



Published with MATLAB® R2015b

Practical no- 3(a)

Practical to find roots using Muller Method

```
p0=-1;
p1=0;
p2=1;
TOL=10^(-5);
format long;
f=@(x) cos(x)-x*exp(x);
for i=1:100
 lambda=(p2-p1)/(p1-p0);
 delta=(p2-p0)/(p1-p0);
 u=f(p0)*(lambda)^2-f(p1)*(delta)^2+f(p2)*(lambda+delta);
 D=(u^2-4*f(p2)*lambda*delta*(f(p0)*lambda-f(p1)*delta+f(p2)))^(1/2);
 if abs(-u-D)<abs(-u+D)
     deno=-u+D;
 else
     deno=-u-D;
 end
 lambda1=(2*f(p2)*delta)/deno;
 p=p2+lambda1*(p2-p1);
 if abs(f(p))<TOL
     disp(p);
     break;
 end
 p0=p1;
 p1=p2;
 p2=p;
end
```

```
   0.517757371045046
```

Published with MATLAB® R2015b

Practical no- 3(b)

Practical to find roots using Muller Method

```
p0=1;
p1=1.5;
p2=2;
TOL=10^(-5);
format long;
f=@(x) x^3-2*x-1;
for i=1:100
 lambda=(p2-p1)/(p1-p0);
 delta=(p2-p0)/(p1-p0);
 u=f(p0)*(lambda)^2-f(p1)*(delta)^2+f(p2)*(lambda+delta);
 D=(u^2-4*f(p2)*lambda*delta*(f(p0)*lambda-f(p1)*delta+f(p2)))^(1/2);
 if abs(-u-D)<abs(-u+D)
     deno=-u+D;
 else
     deno=-u-D;
 end
 lambda1=(2*f(p2)*delta)/deno;
 p=p2+lambda1*(p2-p1);
 if abs(f(p))<TOL
     disp(p);
     break;
 end
 p0=p1;
 p1=p2;
 p2=p;
end
```

```
   1.618033978147904
```

Published with MATLAB® R2015b

Practical -4a

Lagrange Interpolation

```
n=5;
X=[5;7;11;13;17];
Y=[150;392;1452;2366;5202];
x=9;
y=0;
for i=1:n
    dr=1;
    nr=1;
    for j=1:n
        if(j~=i)
            nr=nr*(x-X(j));
            dr=dr*(X(i)-X(j));
        end
    end
    y=y+((nr)/(dr))*Y(i)
end
fprintf('when x=%f,y=%f',x,y)
```

```
y =

 -16.666666666666664


y =

     1.924000000000000e+02


y =

     1.483066666666667e+03


y =

     6.944000000000000e+02


y =

   810

when x=9.000000,y=810.000000
```

Published with MATLAB® R2015b

Practical -4b

Lagrange Interpolation Method

```
n=4;
X=[0;1;2;5];
Y=[2;3;12;147];
x=3;
y=0;
for i=1:n
    dr=1;
    nr=1;
    for j=1:n
        if(j~=i)
            nr=nr*(x-X(j));
            dr=dr*(X(i)-X(j));
        end
    end
    y=y+((nr)/(dr))*Y(i)
end
fprintf('when x=%f,y=%f',x,y)
```

```
y =

   0.800000000000000


y =

  -3.700000000000000


y =

  20.300000000000001


y =

    35

when x=3.000000,y=35.000000
```

Published with MATLAB® R2015b