**Computer Exercises:**

Matlab is using to implement the question.

**1)**

Program to compute the absolute and relative errors in Stirling's approximation:

a)

function y= approximate( x )

%approximate value

y=sqrt(2\*pi\*x).\*((x/exp(1)).^x);

end

b)

function y = exactv( x )

%exact value

y=factorial(x);

end

c)

%error calculation

x=1:10;

% absolute error= approximate value-true value

absoluteerror= approximate(x)-exactv(x);

figure(1);

subplot(2,1,1);

plot(x,absoluteerror);

ylabel('absolute error');

grid;

% relative error=absolute error/true value

relativeerror=absoluteerror./exactv(x);

subplot(2,1,2);

plot(x,relativeerror);

ylabel('relative error');

grid;

Plotting errors:



As we see from above figure, when n increases, absolute error also increases but relative error decreases.

**4)**

Program to compute the mathematical constant e:

% exponential compute

k=1:20;

n=10.^k;

e=(1+1./n).^n;

error=exp(1)-e;

plot(k,error);

Plotting error:



Error does not always decrease as seen from figüre.

**9)**

**a)**

Program to compute the exponential function ex:

function y= ourexponential(x)

y=0;

n=1;

while(1)

if (x.^(n-1))/factorial(n-1)==0

break;

else

y=y+(x.^(n-1))/factorial(n-1);

end

n=n+1;

end

end

**b)**

Stopping criteria is “(x.^(n-1))/factorial(n-1)==0”.

**c)**

Testing program and plotting error:

x=[-20,-15,-10,-5,-1,1,5,10,15,20];

e1=ourexponential(x);

e2=exp(x);

error=e2-e1;

plot(x,error);

grid;

