GE-4 Numerical Methods Practical File

Name - Anshul Verma
Roll No - 19/78065
Course - BSc (Hons) Computer Science

Practical 1: Bisection Method

Submitted by - Anshul Verma
(19/78065)
BSc (Hons) Computer Science

1 ***Determine the first root of f(x) = cos(x).***
Use initial guesses of x0 = 0 and x1 = 2.0.

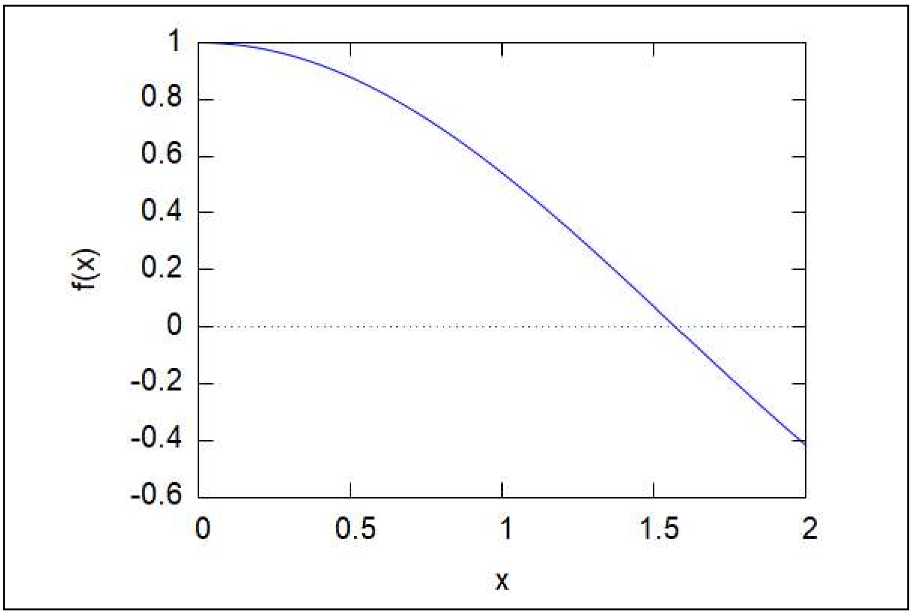
```
/*  Bisection Method  */
kill(all)$
'x0=x0:0$
'x1=x1:2.0$
n:10;
f(x):=cos(x);
if(float(f(x0)·f(x1)>0)) then
 print(" change values")
else
for i:1 thru n do
(a:(x0+x1)/2, if(f(a)·f(x1))>0 then x1:a
 else x0:a,print(i,"iteration gives ",a))$
 print( "after iteration",n,"root is ",a)$
```

(%o3) 10

(%o4) f (x):=cos (x)

1 iteration gives 1.0
2 iteration gives 1.5
3 iteration gives 1.75
4 iteration gives 1.625
5 iteration gives 1.5625
6 iteration gives 1.59375
7 iteration gives 1.578125
8 iteration gives 1.5703125
9 iteration gives 1.57421875
10 iteration gives 1.572265625
after iteration 10 root is 1.572265625

```
wxplot2d(f(x),[x,0,2.0],[grid,1,1],[ylabel,"f(x)"]);
```

(%t7) 

(%o7)

2 Determine the first root of f(x) = −12 − 21x +18x^2 − 2.75x^3. Use initial guesses of x0 = −1 and x1 = 0.

```
kill(all)$
'x0=x0:-1$
'x1=x1:0$
n:10;
f(x):=-12-21·x+18·x^2 - 2.75·x^3;
if(float(f(x0)·f(x1)>0)) then
 print(" change values")
else
for i:1 thru n do
(a:float((x0+x1)/2), if(f(a)·f(x1))>0 then x1:a
 else x0:a,print(i,"iteration gives ",a))$
 print( "after iteration",n,"root is ",a)$
```

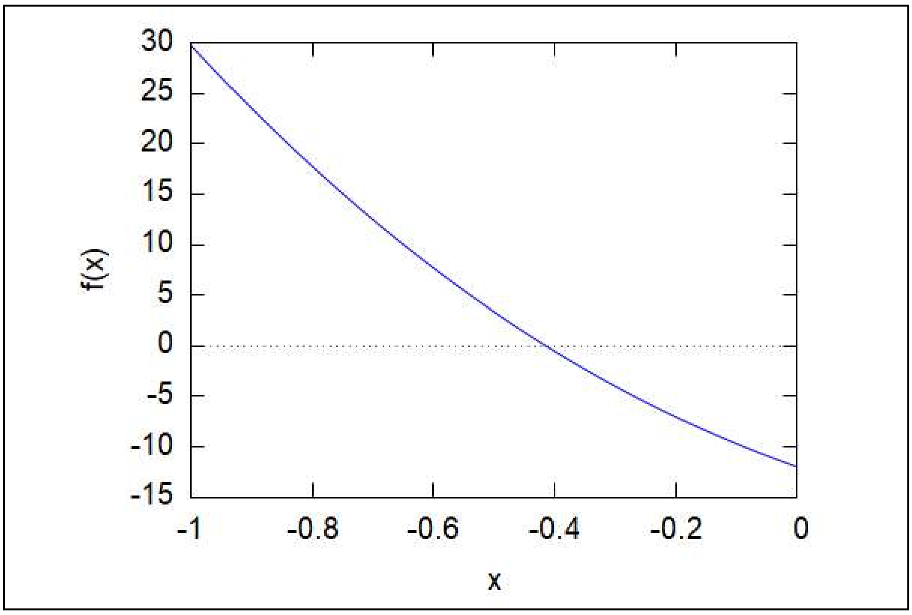
(%o3) 10

(%o4) $f(x):=-12-21\,x+18\,x^2+(-2.75)\,x^3$

```
1  iteration gives  -0.5
2  iteration gives  -0.25
3  iteration gives  -0.375
4  iteration gives  -0.4375
5  iteration gives  -0.40625
6  iteration gives  -0.421875
7  iteration gives  -0.4140625
8  iteration gives  -0.41796875
9  iteration gives  -0.416015625
10  iteration gives  -0.4150390625
after iteration 10 root is  -0.4150390625
```

```
wxplot2d(f(x),[x,-1.0,0],[grid,1,1],[ylabel,"f(x)"]);
```

(%t7)



(%o7)

3 Use bisection method, graphical method and locate the root of f(x) = x^10 − 1 between x = 0 and 1.3.

```
→   kill(all)$
    'x0=x0:0$
    'x1=x1:1.3$
    n:10;
    f(x):= x^10 - 1;
    if(float(f(x0)·f(x1)>0)) then
     print(" change values")
    else
    for i:1 thru n do
    (a:(x0+x1)/2, if(f(a)·f(x1))>0 then x1:a
     else x0:a,print(i,"iteration gives ",a))$
     print("after iteration",n,"root is ",a)$
```

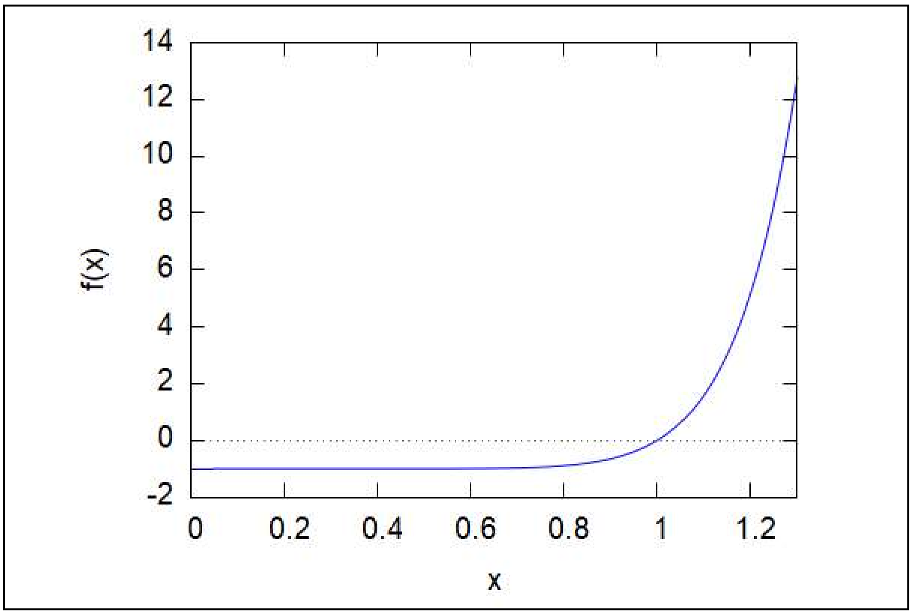
(%o3) 10

(%o4) $f(x):=x^{10}-1$

```
1  iteration gives   0.65
2  iteration gives   0.9750000000000001
3  iteration gives   1.1375
4  iteration gives   1.05625
5  iteration gives   1.015625
6  iteration gives   0.9953125
7  iteration gives   1.00546875
8  iteration gives   1.000390625
9  iteration gives   0.9978515625000001
10  iteration gives   0.9991210937500001
after iteration 10  root is   0.9991210937500001
```

```
→   wxplot2d(f(x),[x,0,1.3],[grid,1,1],[ylabel,"f(x)"]);
```

(%t7)



(%o7)

4 Use bisection and graphical method to determine the mass of the bungee jumper with a drag coefficient of 0.25 kg/m to have a velocity of 36 m/s after 4 s of free fall. Note: The acceleration of gravity is 9.81 m/s^2. Use formula f(m) = sqrt(g*m/cd)*tanh(sqrt(g*cd/m)*t) − v and initial guesses as 50 and 200.

```
kill(all)$
'x0=x0:50$
'x1=x1:200$
n:15;
f(m) := sqrt(9.81·m/0.25)·tanh(sqrt(9.81·0.25/m)·4)-36;
if(float(f(x0)·f(x1)>0)) then
 print(" change values")
else
for i:1 thru n do
(a:float((x0+x1)/2), if(f(a)·f(x1))>0 then x1:a
 else x0:a,print(i,"iteration gives ",a))$
 print("after iteration",n,"root is ",a)$
```

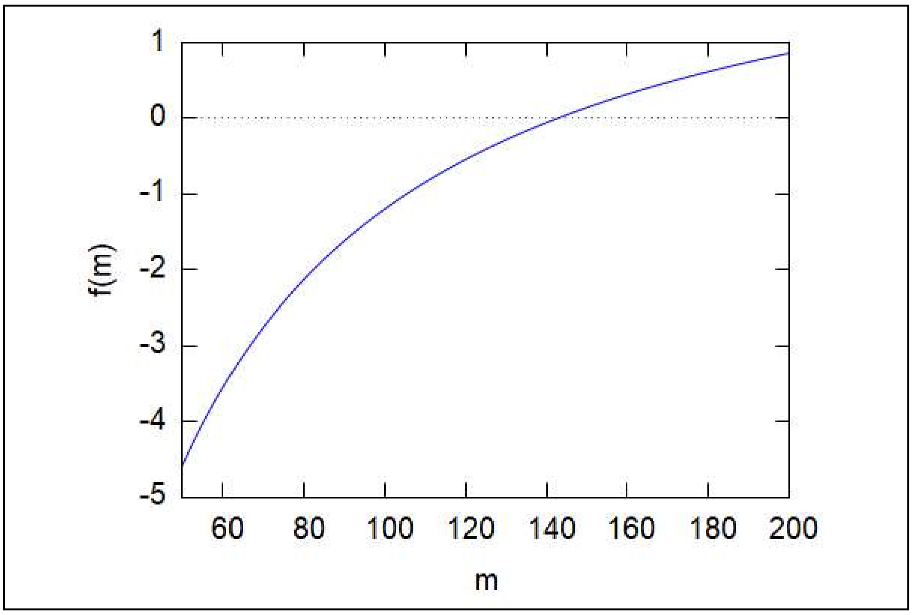
(%o3) 15

(%o4) $f(m):=\sqrt{\frac{9.81\,m}{0.25}}\tanh\left(\sqrt{\frac{9.81\,0.25}{m}}\,4\right)-36$

```
1  iteration gives   125.0
2  iteration gives   162.5
3  iteration gives   143.75
4  iteration gives   134.375
5  iteration gives   139.0625
6  iteration gives   141.40625
7  iteration gives   142.578125
8  iteration gives   143.1640625
9  iteration gives   142.87109375
10  iteration gives   142.724609375
11  iteration gives   142.7978515625
12  iteration gives   142.76123046875
13  iteration gives   142.742919921875
14  iteration gives   142.7337646484375
15  iteration gives   142.7383422851563
after iteration 15 root is   142.7383422851563
```

```
→ wxplot2d(f(m),[m,50.0,200.0],[grid,1,1],[ylabel,"f(m)"]);
```

(%t7)



(%o7)

5 ***You buy a $35,000 vehicle for nothing down at $8,500 per year for 7 years. Use the bisection method function to determine the interest rate that you are paying. Employ initial guesses for the interest rate of 0.01 and 0.3. The formula relating present worth P, annual payments A, number of years n, and interest rate i is***

A = P*(i*(1 + i)^n)/((1 + i)^n − 1)

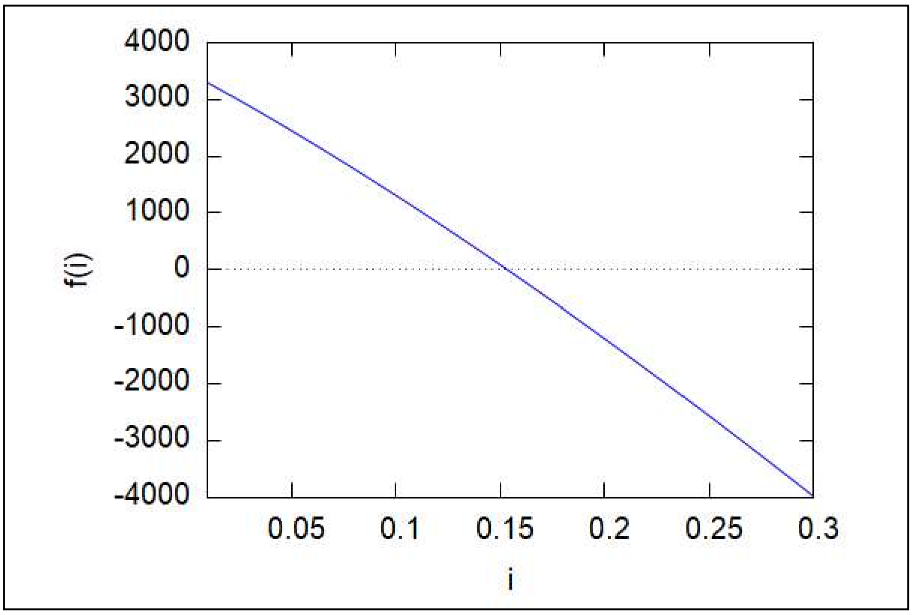
```
kill(all)$
'x0=x0:0.01$
'x1=x1:0.3$
n:10;
f(i):= 8500-35000·(i·(1 + i)^7)/((1 + i)^7 - 1);
if(float(f(x0)·f(x1)>0)) then
 print(" change values")
else
for i:1 thru n do
(a:float((x0+x1)/2), if(f(a)·f(x1))>0 then x1:a
 else x0:a,print(i,"iteration gives ",a))$
 print("after iteration",n,"root is ",a)$
```

(%o3) 10

(%o4) $f(i):=8500-\frac{35000\left(i\,(1+i)^{7}\right)}{(1+i)^{7}-1}$

```
1  iteration gives   0.155
2  iteration gives   0.0825
3  iteration gives   0.11875
4  iteration gives   0.136875
5  iteration gives   0.1459375
6  iteration gives   0.15046875
7  iteration gives   0.152734375
8  iteration gives   0.1538671875
9  iteration gives   0.15330078125
10  iteration gives   0.153583984375
after iteration  10  root is   0.153583984375
```

```
wxplot2d(f(i),[i,0.01,0.3],[grid,1,1],[ylabel,"f(i)"]);
```

(%t7) 

(%o7)

Practical 2(a): Secant Method

Submitted by - Anshul Verma
(19/78065)
BSc (Hons) Computer Science

1 *Determine the root of f(x) = x^3-x-1 using Secant method. Use initial guesses as 1.3 and 1.4.*

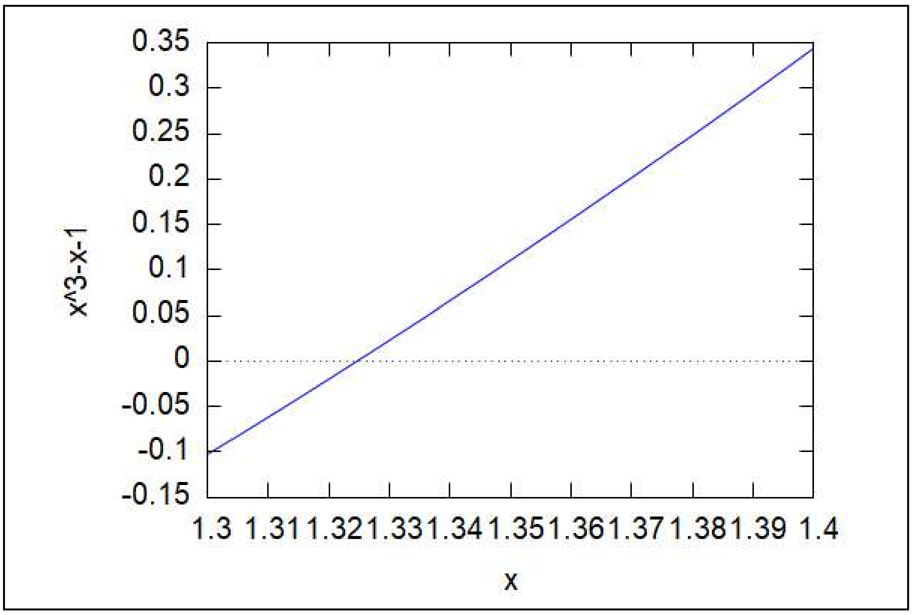
```
kill(all)$
'x0=x0:1.3;
'x1=x1:1.4;
f(x):=x^3-x-1;
for i:1 thru 6 do (
    if(equal(f(x0),f(x1)))
       then return()
    else
    x2:(x0·f(x1)-x1·f(x0))/(f(x1)-f(x0)),
    x0:x1, x1:x2,
print("iteration",i,", root =",x2))$
print("Root is: ",x2)$
wxplot2d(f(x),[x,1.3,1.4]);
```

(%o1) $x0=1.3$

(%o2) $x1=1.4$

(%o3) $f(x):=x^3-x-1$

```
iteration 1 , root = 1.323042505592841
iteration 2 , root = 1.3246060608507
iteration 3 , root = 1.324718132164679
iteration 4 , root = 1.324717957226505
iteration 5 , root = 1.324717957244746
iteration 6 , root = 1.324717957244746
Root is:  1.324717957244746
```

(%t6) 

(%o6)

2 Determine the highest real root of f (x) = x3− 6x2 + 11x − 6.1 using the secant method. (three iterations, xi−1 = 2.5 and xi = 3.5).

```
kill(all)$
'x0=x0:2.5;
'x1=x1:3.5;
f(x):= x^3- 6·x^2 + 11·x - 6.1;
for i:1 thru 3 do (
    if(equal(f(x0),f(x1)))
       then return()
    else
    x2:(x0·f(x1)-x1·f(x0))/(f(x1)-f(x0)),
    x0:x1, x1:x2,
print("iteration",i,", root =",x2))$
print("Root is: ",x2)$
wxplot2d(f(x),[x,2.5,3.5]);
```

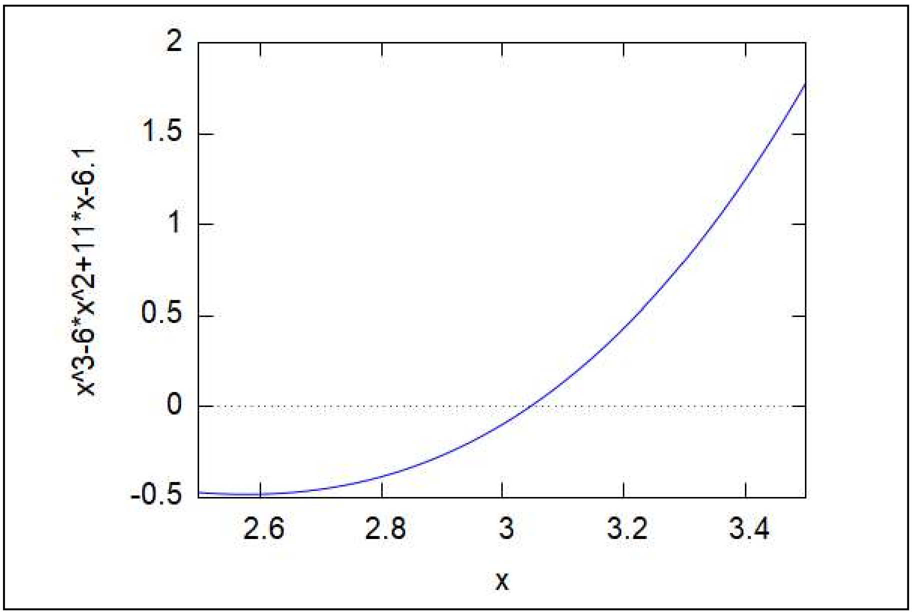
(%o1) $x0=2.5$

(%o2) $x1=3.5$

(%o3) $f(x):=x^3-6x^2+11x-6.1$

iteration 1 , root = 2.711111111111111
iteration 2 , root = 2.871090503477539
iteration 3 , root = 3.221923449437695
Root is: 3.221923449437695

(%t6)



(%o6)

3 Determine the lowest positive root of

f(x) = 7*sin(x)e^−x − 1:
(a) Using the secant method (three iterations, xi−1 = 0.5 and xi = 0.4
(b) Graphically.

```
kill(all)$
'x0=x0:0.5;
'x1=x1:0.4;
f(x):= 7·sin(x)·(%e)^-x - 1;
for i:1 thru 6 do (
    if(equal(f(x0),f(x1)))
       then return()
    else
    x2:(x0·f(x1)-x1·f(x0))/(f(x1)-f(x0)),
    x0:x1, x1:x2,
print("iteration",i,", root =",x2))$
print("Root is: ",x2)$
```

(%o1) $x0=0.5$

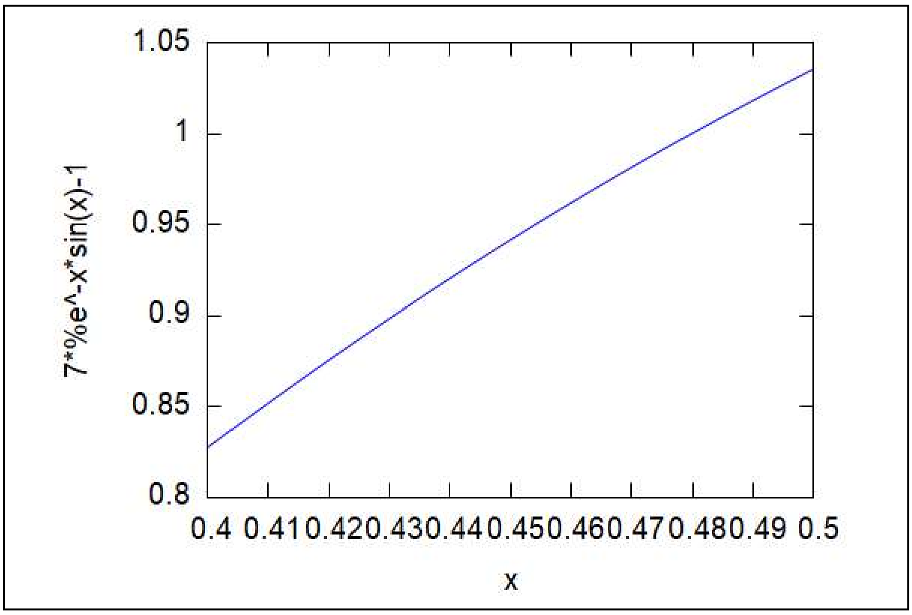
(%o2) $x1=0.4$

(%o3) $f(x):=7\sin(x)\,\%e^{-x}-1$

```
iteration 1 , root = 0.002782023389712249
iteration 2 , root = 0.2182365700522151
iteration 3 , root = 0.1789888834227972
iteration 4 , root = 0.1696440472681687
iteration 5 , root = 0.1701857331972554
iteration 6 , root = 0.1701799974665363
Root is:  0.1701799974665363
```

```
wxplot2d(f(x),[x,0.4,0.5]);
```

(%t6)



(%o6)

4 *Use secant method to determine the mass of the bungee jumper with a drag coefficient of 0.25 kg/m to have a velocity of 36 m/s after 4 s of free fall. Note: The acceleration of gravity is 9.81 m/s^2. Use formula f(m) = sqrt(g*m/cd)*tanh(sqrt(g*cd/m)*t) − v and initial guesses as 50 and 200.*

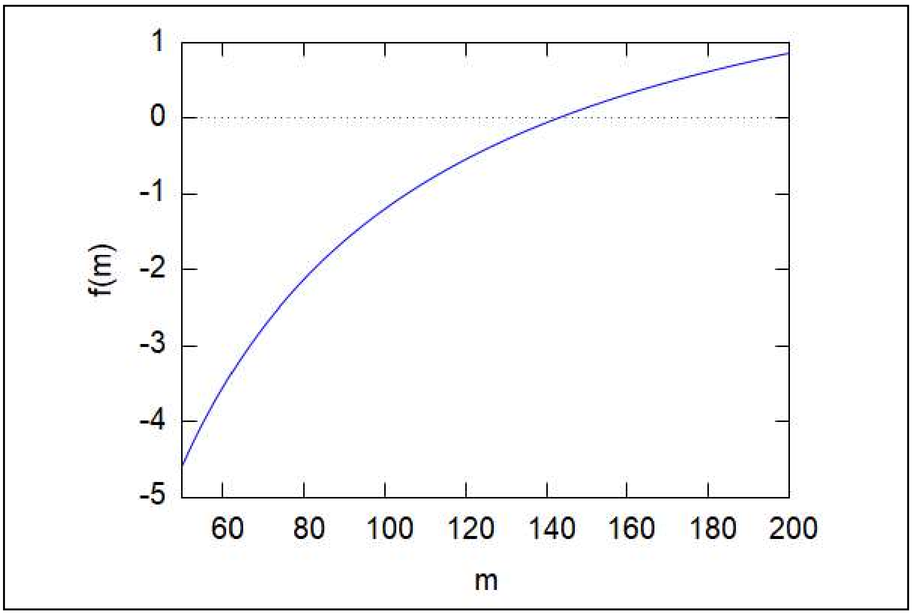
```
kill(all)$
'x0=x0:50;
'x1=x1:200;
f(m) := sqrt(9.81·m/0.25)·tanh(sqrt(9.81·0.25/m)·4)-36;
for i:1 thru 6 do (
    if(equal(f(x0),f(x1)))
       then return()
    else
    x2:(x0·f(x1)-x1·f(x0))/(f(x1)-f(x0)),
    x0:x1, x1:x2,
print("iteration",i,", root =",x2))$
print("Root is: ",x2)$
wxplot2d(f(m),[m,50,200],[ylabel,"f(m)"]);
```

(%o1) $x0=50$

(%o2) $x1=200$

(%o3) $f(m):=\sqrt{\frac{9.81\,m}{0.25}}\tanh\left(\sqrt{\frac{9.81\;0.25}{m}}\,4\right)-36$

iteration 1 , root = 176.2773459668288
iteration 2 , root = 130.6111872022427
iteration 3 , root = 145.3057954093966
iteration 4 , root = 142.9342845415285
iteration 5 , root = 142.7344441086685
iteration 6 , root = 142.7376370683808
Root is: 142.7376370683808

(%t6) 

(%o6)

5 Determine the highest real root of f (x) = cos(x) + 2*sin(x) + x^2 using the secant method. (six iterations, xi−1 = 0 and xi = -0.1).

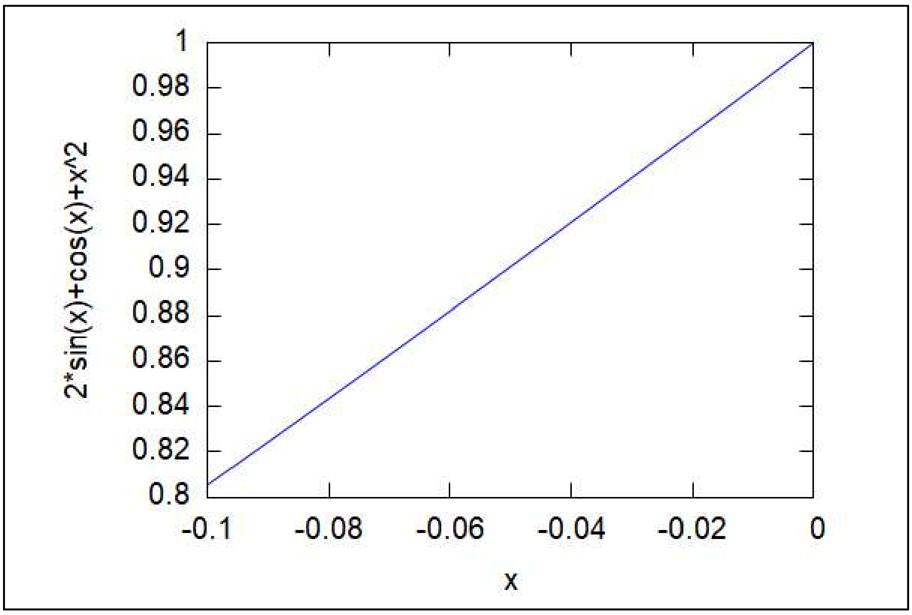
```
kill(all)$
'x0=x0:0;
'x1=x1:-0.1;
f(x) := cos(x) + 2·sin(x) + x^2;
for i:1 thru 6 do (
    if(equal(f(x0),f(x1)))
       then return()
    else
    x2:(x0·f(x1)-x1·f(x0))/(f(x1)-f(x0)),
    x0:x1, x1:x2,
print("iteration",i,", root =",x2))$
print("Root is: ",x2)$
wxplot2d(f(x),[x,-0.1,0]);
```

(%o1) $x0=0$

(%o2) $x1=-0.1$

(%o3) $f(x):=\cos(x)+2\sin(x)+x^2$

```
iteration 1 , root = -0.513709182245311
iteration 2 , root = -0.609961481945027
iteration 3 , root = -0.6517970946457313
iteration 4 , root = -0.658798769946493
iteration 5 , root = -0.6592612540851915
iteration 6 , root = -0.6592660426540474
Root is:  -0.6592660426540474
```

(%t6) 

(%o6)

Practical 2(b): Regula-Falsi Method

Submitted by - Anshul Verma
(19/78065)
BSc (Hons) Computer Science

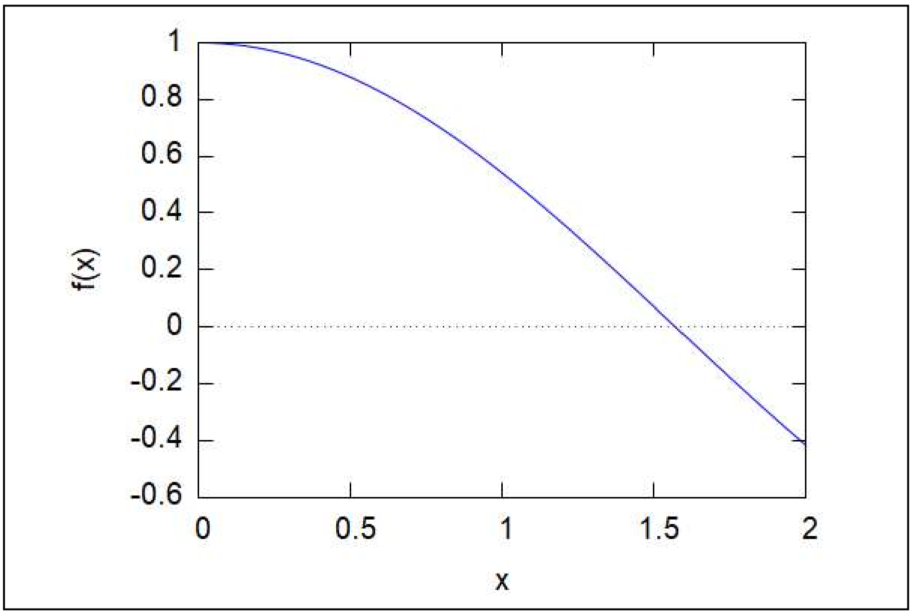
1 Find the real root of the function f(x)=cos(x) by using regula falsi method.

```
kill(all)$
f(x):=cos(x);
x0=x0:0;
x1=x1:2;
if(float(f(x0)·f(x1)>0)) then
print("change values")
else
for i:1 thru 10 do
(x2:float(((x0·f(x1))-(x1·f(x0)))/(f(x1)-f(x0))),
    if (f(x2)·f(x1)<0) then x0:x2
     else (x1:x2),print(i,"iteration gives",x2))$
print("the root is",x2)$
```

```
(%o1) f(x):=cos(x)
(%o2) x0=0
(%o3) x1=2
1 iteration gives 1.412282927437392
2 iteration gives 1.57390632372288
3 iteration gives 1.570783521943903
4 iteration gives 1.570796326815453
5 iteration gives 1.570796326794897
6 iteration gives 1.570796326794897
7 iteration gives 1.570796326794897
8 iteration gives 1.570796326794897
9 iteration gives 1.570796326794897
10 iteration gives 1.570796326794897
the root is 1.570796326794897
```

```
wxplot2d(cos(x),[x,0,2],[ylabel,"f(x)"]);
```

(%t6)



(%o6)

2 Find a real root of f(x)= x³-2x-5=0 using the regula falsi method upto four iteration.

```
kill(all)$
f(x):=x^(3)-2·x-5;
x0=x0:2;
x1=x1:3;
n:4$
if(float(f(x0)·f(x1)>0)) then
print("change values")
else
for i:1 thru n do
(x2:float(((x0·f(x1))-(x1·f(x0)))/(f(x1)-f(x0))),
    if (f(x2)·f(x1)<0) then x0:x2
     else (x1:x2),print("iteration",i,"gives",x2))$
print("after iteration",n,"the root is",x2)$
```

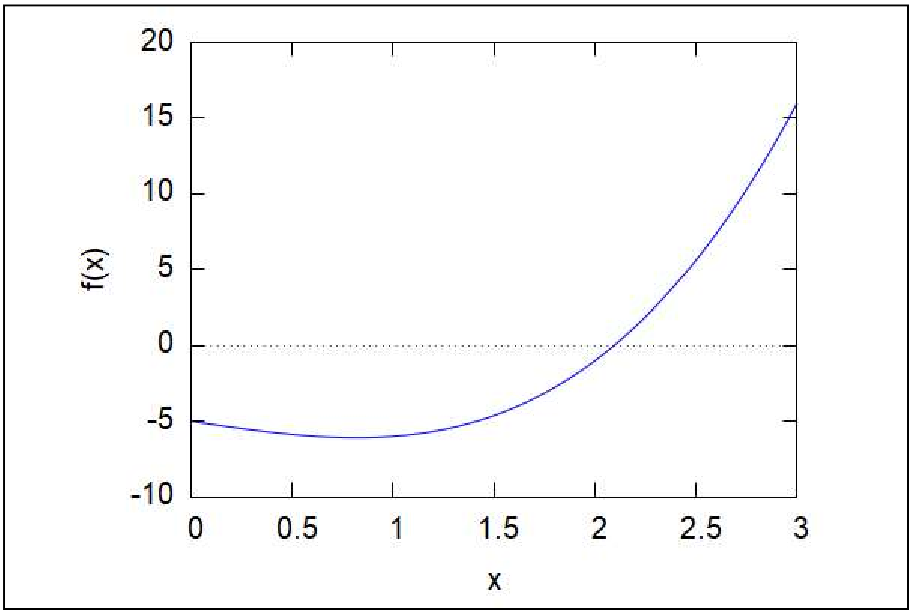
(%o1) $f(x):=x^3-2x-5$

(%o2) $x0=2$

(%o3) $x1=3$

```
iteration 1 gives 2.058823529411764
iteration 2 gives 2.081263659845022
iteration 3 gives 2.089639210090847
iteration 4 gives 2.092739574318006
after iteration 4 the root is 2.092739574318006
```

```
wxplot2d(f(x),[x,0,3],[ylabel,"f(x)"]);
```

(%t7) 

(%o7)

3 Find a real root of f(x)= x^10 - 1 using the regula falsi method upto five iteration.

```
kill(all)$
f(x):= x^10 - 1;
x0=x0:0;
x1=x1:1.3;
n:5$
if(float(f(x0)·f(x1)>0)) then
print("change values")
else
for i:1 thru n do
(x2:float(((x0·f(x1))-(x1·f(x0)))/(f(x1)-f(x0))),
    if (f(x2)·f(x1)<0) then x0:x2
     else (x1:x2),print("iteration",i,"gives",x2))$
print("after iteration",n,"the root is",x2)$
```

(%o1) $f(x):=x^{10}-1$

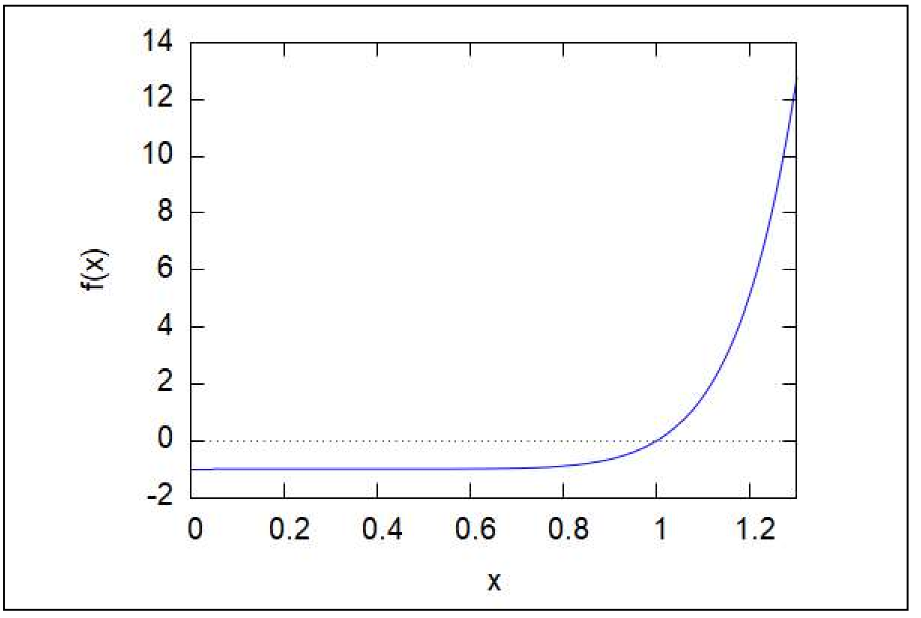
(%o2) $x0=0$

(%o3) $x1=1.3$

iteration 1 gives 0.09429959537232274
iteration 2 gives 0.1817588725190794
iteration 3 gives 0.2628740125203043
iteration 4 gives 0.3381051033222695
iteration 5 gives 0.4078779165927524
after iteration 5 the root is 0.4078779165927524

```
wxplot2d(f(x),[x,0,1.3],[ylabel,"f(x)"]);
```

(%t7)



(%o7)

4 Determine the real root for eq f(x) = x3− 6x2 + 11x − 6.1 using the regula-falsi method.

```
kill(all)$
f(x):= x^3- 6·x^2 + 11·x - 6.1;
x0=x0:2.5;
x1=x1:3.5;
n:6$
if(float(f(x0)·f(x1)>0)) then
print("change values")
else
for i:1 thru n do
(x2:float(((x0·f(x1))-(x1·f(x0)))/(f(x1)-f(x0))),
    if (f(x2)·f(x1)<0) then x0:x2
     else (x1:x2),print("iteration",i,"gives",x2))$
print("after iteration",n,"the root is",x2)$
```

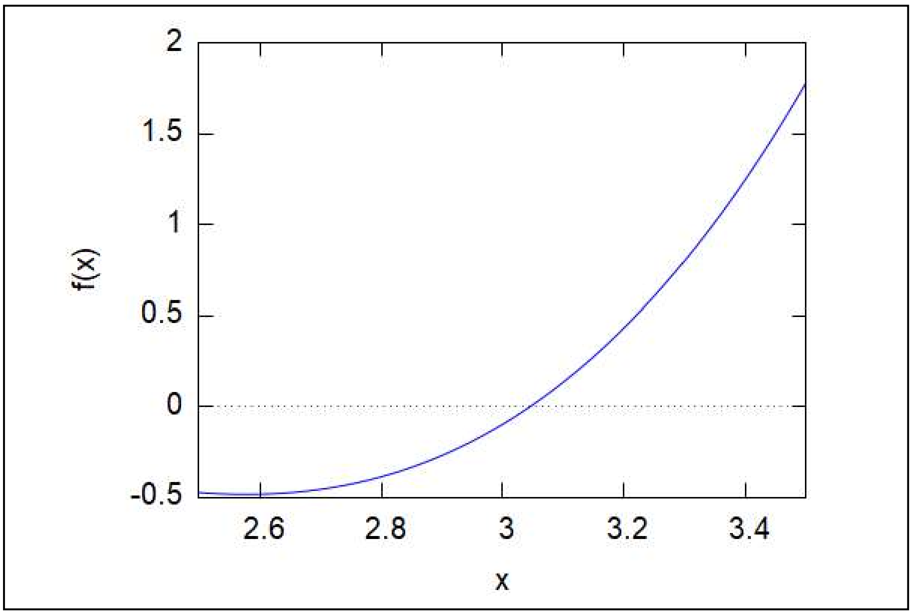
(%o1) $f(x):=x^3-6x^2+11x-6.1$

(%o2) $x0=2.5$

(%o3) $x1=3.5$

```
iteration 1 gives 2.711111111111111
iteration 2 gives 2.871090503477539
iteration 3 gives 2.96462521199529
iteration 4 gives 3.01067414289048
iteration 5 gives 3.031349848568632
iteration 6 gives 3.040239684107106
after iteration 6 the root is 3.040239684107106
```

```
wxplot2d(f(x),[x,2.5,3.5],[ylabel,"f(x)"]);
```

(%t7) 

(%o7)

Practical 3:
Newton-Raphson Method

Submitted by - Anshul Verma
(19/78065)
BSc (Hons) Computer Science

1 *Determine the positive root of f(x) = x^3-3*x+1 using the Newton-Raphson method and an initial guess of x = 0.3.*

```
kill(all)$
f(x):=x^3-3·x+1;
define(df(x),diff(f(x),x));
'x0=x0:0.3;
for i:1 thru 6 do (
    if(equal(f(x0),0))
       then return()
    else
    float(x1:(x0-f(x0)/df(x0))),
    x0:x1,
print("iteration",i,"root",float(x1)))$;
print("Root is: ",float(x1))$
wxplot2d(f(x),[x,-2,3],[y,-10,15]);
```

(%o1) $f(x):=x^3-3x+1$

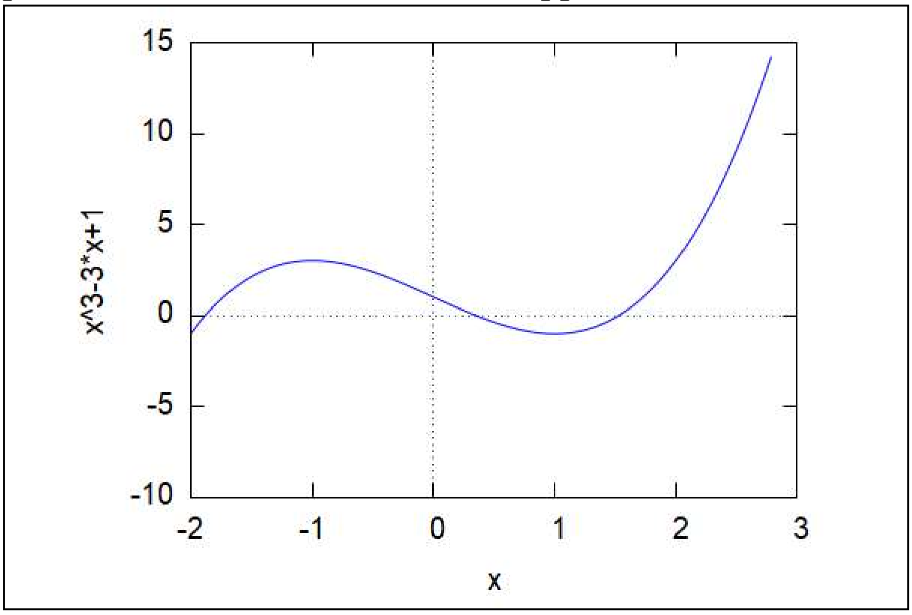
(%o2) $df(x):=3x^2-3$

(%o3) $x0=0.3$

```
iteration 1 root 0.3465201465201466
iteration 2 root 0.3472961178879339
iteration 3 root 0.3472963553338384
iteration 4 root 0.3472963553338607
iteration 5 root 0.3472963553338606
iteration 6 root 0.3472963553338607
Root is:  0.3472963553338607
```

plot2d: some values were clipped.

(%t6)



(%o6)

The above method can not be done by directly putting the value of diff(f(x),x) because the value of diff(f(x0),x) will be zero since its constant. therefore we have defined df(x) in above question.

```
kill(all)$
f(x):=x^3-3·x+1;
'x0=x0:0.3;
for i:1 thru 6 do (
    if(equal(diff(f(x0),x),0))
       then return()
    else
    float(x1:(x0-f(x0)/diff(f(x0),x)),
    x0:x1,
print("iteration",i,"root",float(x1))))$;
print("Root is: ",float(x1))$
```

(%o1) $f(x) := x^3 - 3x + 1$

(%o2) $x0 = 0.3$

Root is: x1

2 Determine the positive root of f(x) = x^10-1 using the Newton-Raphson method and an initial guess of x = 0.5.

```
kill(all)$
f(x):=x^10-1;
define(df(x),diff(f(x),x));
'x0=x0:0.5;
for i:1 thru 50 do (
    if(equal(f(x0),0))
       then return()
    else
    float(x1:(x0-f(x0)/df(x0))),
    x0:x1,
print("iteration",i,"root",float(x1)))$;
print("Root is: ",float(x1))$
wxplot2d(f(x),[x,0,2],[y,-10,15]);
```

(%o1) $f(x):=x^{10}-1$

(%o2) $df(x):=10\,x^{9}$

(%o3) $x0=0.5$

iteration 1 root 51.65
iteration 2 root 46.485
iteration 3 root 41.8365
iteration 4 root 37.65285
iteration 5 root 33.887565
iteration 6 root 30.4988085
iteration 7 root 27.44892765000001
iteration 8 root 24.70403488500002
iteration 9 root 22.23363139650005
iteration 10 root 20.01026825685012
iteration 11 root 18.0092414311653
iteration 12 root 16.20831728804927
iteration 13 root 14.58748555924564
iteration 14 root 13.12873700332442
iteration 15 root 11.81586330300061
iteration 16 root 10.63427697272282
iteration 17 root 9.570849275508033
iteration 18 root 8.613764348105635
iteration 19 root 7.752387913678128
iteration 20 root 6.977149123299052
iteration 21 root 6.279434213521248
iteration 22 root 5.651490798756543
iteration 23 root 5.086341735884172
iteration 24 root 4.57770760618419 8
iteration 25 root 4.119936958849513
iteration 26 root 3.707943555369608
iteration 27 root 3.337149954580646
iteration 28 root 3.003436907255125
iteration 29 root 2.703098244970869
iteration 30 root 2.432801399542296
iteration 31 root 2.189554759223996
iteration 32 root 1.970685739811615
iteration 33 root 1.773840237097568
iteration 34 root 1.597031347969508
iteration 35 root 1.438807931427029
iteration 36 root 1.298711342726572
iteration 37 root 1.178354715620168

3 Determine the lowest positive root of

f (x) = 7 sin(x)e−x − 1:
(a) Graphically.
(b) Using the Newton-Raphson method (three iterations, xi = 0.3).

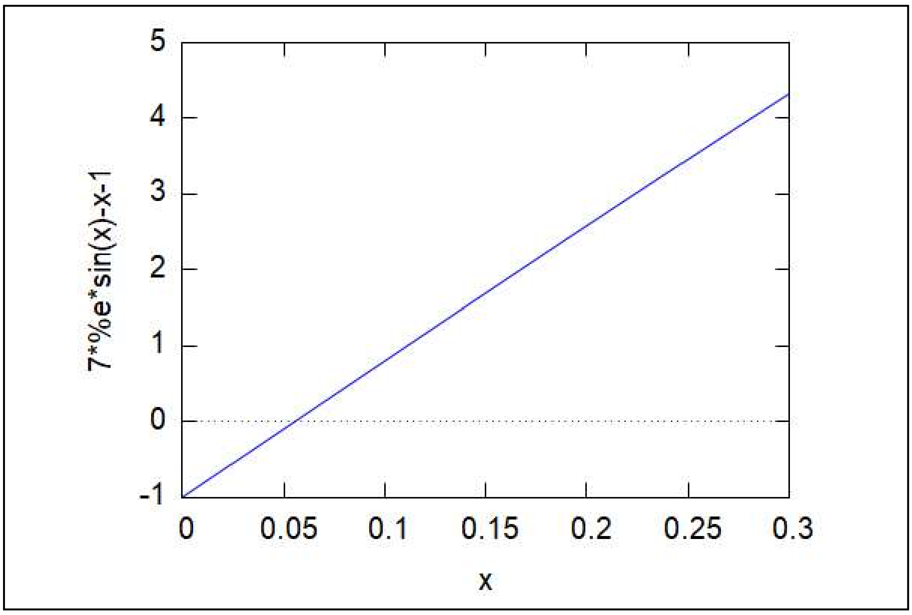
```
kill(all)$
keepfloat:true$
f(x):= 7·sin(x)·(%e)-x - 1;
define(df(x),diff(f(x),x));
'x0=x0:0.3;
for i:1 thru 3 do (
    if(equal(f(x0),0))
       then return()
    else
    float(x1:(x0-f(x0)/df(x0))),
    x0:x1,
print("iteration",i,"root",float(x1)))$;
print("Root is: ",float(x1))$
wxplot2d(f(x),[x,0,0.3]);
```

(%o2) $f(x):=7\sin(x)\,\%e-x-1$

(%o3) $df(x):=7\,\%e\cos(x)-1$

(%o4) $x0=0.3$

iteration 1 root 0.04833385276589969
iteration 2 root 0.05549804539250931
iteration 3 root 0.05549942090726669
Root is: 0.05549942090726669

(%t7) 

(%o7)

4 Use the Newton-Raphson method to determine a root of $f(x) = -0.9x2 + 1.7x + 2.5$ using $x0 = 5$.

```
→ kill(all)$
  keepfloat:true$
  f(x):= -0.9·x^2 + 1.7·x + 2.5;
  define(df(x),diff(f(x),x));
  'x0=x0:5;
  for i:1 thru 10 do (
      if(equal(f(x0),0))
         then return()
      else
      float(x1:(x0-f(x0)/df(x0))),
      x0:x1,
  print("iteration",i,"root",float(x1)))$;
  print("Root is: ",float(x1))$
  wxplot2d(f(x),[x,0,5],[y,-10,15]);
```

(%o2) $f(x):=(-0.9)\,x^2+1.7\,x+2.5$

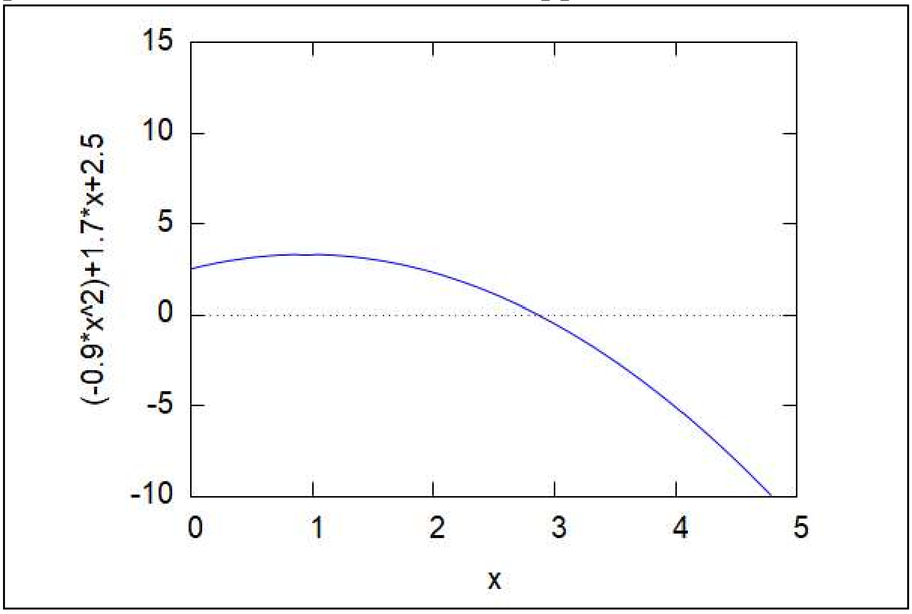
(%o3) $df(x):=1.7-1.8\,x$

(%o4) $x0=5$

```
iteration 1 root 3.424657534246575
iteration 2 root 2.924356996641545
iteration 3 root 2.861146975661041
iteration 4 root 2.860104689054935
iteration 5 root 2.860104405507428
iteration 6 root 2.860104405507407
iteration 7 root 2.860104405507407
iteration 8 root 2.860104405507407
iteration 9 root 2.860104405507407
iteration 10 root 2.860104405507407
Root is: 2.860104405507407
```

plot2d: some values were clipped.

(%t7)



(%o7)

5 *Use the Newton-Raphson method and to determine a root of f (x) = x5 −16.05x4 +88.75x3 −192.0375x2 +116.35x +31.6875 using an initial guess of x = 0.5825*

```
kill(all)$
keepfloat:true$
f(x):= x^5 -16.05·x^4 +88.75·x^3 -192.0375·x^2 +116.35·x +31.6875;
define(df(x),diff(f(x),x));
'x0=x0:0.5825;
for i:1 thru 21 do (
    if(equal(f(x0),0))
       then return()
    else
    float(x1:(x0-f(x0)/df(x0))),
    x0:x1,
print("iteration",i,"root",float(x1)))$;
print("Root is: ",float(x1))$
wxplot2d(f(x),[x,6,7],[y,-5,5]);
```

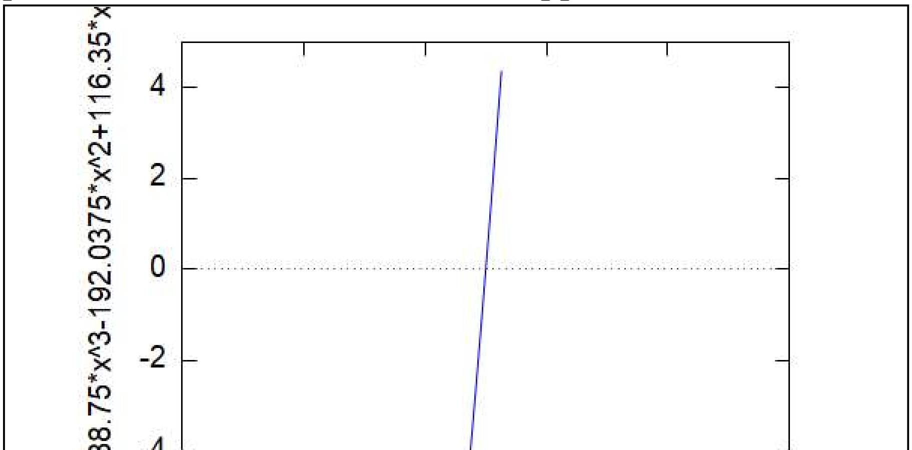
(%o2) $f(x):=x^5-16.05x^4+88.75x^3+(-192.0375)x^2+116.35x+31.6875$

(%o3) $df(x):=5x^4-64.2x^3+266.25x^2-384.075x+116.35$

(%o4) $x0=0.5825$

```
iteration 1 root 2.300097735514282
iteration 2 root 90.07505700858952
iteration 3 root 72.71519620965479
iteration 4 root 58.83059124381164
iteration 5 root 47.72701020805061
iteration 6 root 38.8492720172654
iteration 7 root 31.75348765970685
iteration 8 root 26.08486672492231
iteration 9 root 21.55998037635415
iteration 10 root 17.95259568994002
iteration 11 root 15.08237674302753
iteration 12 root 12.80589504579965
iteration 13 root 11.00951528647989
iteration 14 root 9.603832150966982
iteration 15 root 8.519442093848085
iteration 16 root 7.703943262344701
iteration 17 root 7.12005663602736
iteration 18 root 6.743746231180463
iteration 19 root 6.554961873071568
iteration 20 root 6.503642907818461
iteration 21 root 6.500017452342051
Root is:  6.500017452342051
```

plot2d: some values were clipped.

(%t7) 

Problem 4(a): Gauss Elimination Method

Submitted by - Anshul Verma
(19/78065)
BSc (Hons) Computer Science

1 *Use Gauss elimination to solve*
$2x - y = -5$,
$4x - y = 3$

```
→   kill(all)$ /* kill all variables and clear memory */
    keepfloat:true$ /* keep float values as it is */
    'A = A:matrix( /* function to create a matrix */
            [2, -1],
            [4, -1])$
    'B = B:matrix(
            [-5], [3])$
    'X = X:matrix(
            ['x], ['y])$
    /* Self-explainatory print statements */
    print("Let",'A=A,",",'B=B,",",'X=X)$
    print("")$
    print("Now, the augmented matrix will be,")$
    print("")$
    /* Creating augmented matrix by joining B to A */
    /* Since, B is a column matrix, addcol adds it to end */
    'Aug = Aug:addcol(A,B);
    print("")$
    print("I. FORWARD ELIMINATION")$
    n:length(A[1])$
    /* -------- Forward Elimination -------- */
    /* Moving from one pivot row to the next */
    for k:1 thru n-1 do(
        /* Moving below the pivot row */
        for i:k+1 thru n do(
            factor: Aug[i,k]/Aug[k,k],
            print(""),
            print("=> R",i,"= R",i,"- (",'Aug[i,k]/'Aug[k,k],")*","R",k),
            /* Applying Ri -> Ri - (Augik/Augkk)*Rk */
            Aug[i]: Aug[i]-factor·Aug[k],
            print(Aug)
        )
    )$
    print("")$
    print("Therefore, the augmented matrix")$
    print("reduced to upper triangular form will be,")$
    print("")$
    Aug;
    /* Printing reduced system of eqs */
    print("")$
    print("Now, the system of equations will be,")$
    load("eigen")$ /* to use innerproduct function */
    /* innerproduct returns dot product */
    /* submatrix returns a new matrix from matrix Aug
    with mentioned rows and columns deleted. */
    AA:innerproduct(submatrix(Aug,n+1),X)$
    BB:col(Aug,n+1)$ /* col returns specified column */
    for i:1 thru n do(
        print(AA[i,1]=BB[i,1])
    )$
    print("")$
    print("II. BACKWARD SUSTITUTION")$
    print("")$
    /* -------- Backward Substitution -------- */
    X1: zeromatrix(n,1)$ /* creates Zero matrix of dimention nx1 */
```

2 *Use Gauss elimination to solve*

$3x - 0.1y - 0.2z = 7.85$,
$0.1x + 7y - 0.3z = -19.3$,
$0.3x - 0.2y + 10z = 71.4$

```
➔   kill(all)$ /* kill all variables and clear memory */
    keepfloat:true$ /* keep float values as it is */
    'A = A:matrix( /* function to create a matrix */
            [3.0, -0.1, -0.2],
            [0.1, 7.0, -0.3],
            [0.3, -0.2, 10.0])$
    'B = B:matrix(
            [7.85], [-19.3], [71.4])$
    'X = X:matrix(
            ['x], ['y], ['z])$
    /* Self-explainatory print statements */
    print("Let",'A=A,",",'B=B,",",'X=X)$
    print("")$
    print("Now, the augmented matrix will be,")$
    print("")$
    /* Creating augmented matrix by joining B to A */
    /* Since, B is a column matrix, addcol adds it to end */
    'Aug = Aug:addcol(A,B);
    print("")$
    print("I. FORWARD ELIMINATION")$
    n:length(A[1])$
    /* -------- Forward Elimination -------- */
    /* Moving from one pivot row to the next */
    for k:1 thru n-1 do(
        /* Moving below the pivot row */
        for i:k+1 thru n do(
            factor: Aug[i,k]/Aug[k,k],
            print(""),
            print("=> R",i,"= R",i,"- (",'Aug[i,k]/'Aug[k,k],")*","R",k),
            /* Applying Ri -> Ri - (Augik/Augkk)*Rk */
            Aug[i]: Aug[i]-factor·Aug[k],
            print(Aug)
        )
    )$
    print("")$
    print("Therefore, the augmented matrix")$
    print("reduced to upper triangular form will be,")$
    print("")$
    Aug;
    /* Printing reduced system of eqs */
    print("")$
    print("Now, the system of equations will be,")$
    load("eigen")$ /* to use innerproduct function */
    /* innerproduct returns dot product */
    /* submatrix returns a new matrix from matrix Aug
    with mentioned rows and columns deleted. */
    AA:innerproduct(submatrix(Aug,n+1),X)$
    BB:col(Aug,n+1)$ /* col returns specified column */
    for i:1 thru n do(
        print(AA[i,1]=BB[i,1])
    )$
    print("")$
    print("II. BACKWARD SUSTITUTION")$
    print("")$
    /* -------- Backward Substitution -------- */
```

3 *Solve the system of equations*

$2y + 3z = 8$,

$4x + 6y + 7z = -3$,

$2x - 3y + 6z = 5$,

using Gauss elimination with partial pivoting.

```
kill(all)$
keepfloat:true$
'A = A:matrix(
        [0, 2, 3],
        [4, 6, 7],
        [2, -3, 6])$
'B = B:matrix(
        [8], [-3], [5])$
'X = X:matrix(
        ['x], ['y], ['z])$
print("Let",'A=A,",",'B=B,",",'X=X)$
print("")$
print("Now, the augmented matrix will be,")$
print("")$
'Aug = Aug:addcol(A,B);
print("")$
print("I. FORWARD ELIMINATION")$
n:length(A[1])$
/* -------- Forward Elimination -------- */
for k:1 thru n-1 do(

    /* Partial Pivoting */
    /* determine the largest element in the column */
    /* and store the row number to max_i */
    max_i: k,
    for i:k thru n do(
        if abs(Aug[i,k]) > abs(Aug[max_i,k]) then
            max_i: i
    ),

    if max_i#k then( /* if row number is not k */
        /* switch rows */
        [Aug[k],Aug[max_i]]:[Aug[max_i],Aug[k]],
        print(""),
        print("=> R",k,"< -- >","R",max_i),
        print(Aug)
    ),

    for i:k+1 thru n do(
        factor: Aug[i,k]/Aug[k,k],
        print(""),
        print("=> R",i,"= R",i,"- (",'Aug[i,k]/'Aug[k,k],")*","R",k),
        /* Applying Ri -> Ri - (Augik/Augkk)*Rk */
        Aug[i]: Aug[i]-factor·Aug[k],
        print(Aug)
    )
)$
print("")$
print("Therefore, the augmented matrix")$
print("reduced to upper triangular form will be,")$
print("")$
Aug;
print("")$
print("Now, the system of equations will be,")$
load("eigen")$
```

4 Given the equations

$2x1 - 6x2 - x3 = -38$
$-3x1 - x2 + 7x3 = -34$
$-8x1 + x2 - 2x3 = -20$
Solve by Gauss elimination with partial pivoting.

```
kill(all)$
keepfloat:true$
'A = A:matrix(
        [2, -6, -1],
        [-3, -1, 7],
        [-8, 1, 2])$
'B = B:matrix(
        [-38], [-34], [-20])$
'X = X:matrix(
        ['x1], ['x2], ['x3])$
print("Let",'A=A,",",'B=B,",",'X=X)$
print("")$
print("Now, the augmented matrix will be,")$
print("")$
'Aug = Aug:addcol(A,B);
print("")$
print("I. FORWARD ELIMINATION")$
n:length(A[1])$
/* -------- Forward Elimination -------- */
for k:1 thru n-1 do(

    /* Partial Pivoting */
    /* determine the largest element in the column */
    /* and store the row number to max_i */
    max_i: k,
    for i:k thru n do(
        if abs(Aug[i,k]) > abs(Aug[max_i,k]) then
            max_i: i
    ),

    if max_i#k then( /* if row number is not k */
        /* switch rows */
        [Aug[k],Aug[max_i]]:[Aug[max_i],Aug[k]],
        print(""),
        print("=> R",k,"< -- >","R",max_i),
        print(Aug)
    ),

    for i:k+1 thru n do(
        factor: Aug[i,k]/Aug[k,k],
        print(""),
        print("=> R",i,"= R",i,"- (",'Aug[i,k]/'Aug[k,k],")*","R",k),
        /* Applying Ri -> Ri - (Augik/Augkk)*Rk */
        Aug[i]: Aug[i]-factor·Aug[k],
        print(Aug)
    )
)$
print("")$
print("Therefore, the augmented matrix")$
print("reduced to upper triangular form will be,")$
print("")$
Aug;
print("")$
print("Now, the system of equations will be,")$
load("eigen")$
```

5 *Given the system of equations*

2x2 + 5x3 = 1

2x1 + x2 + x3 = 1

3x1 + x2 = 2

Use Gauss elimination with partial pivoting to solve.

```
kill(all)$
keepfloat:true$
'A = A:matrix(
        [0, 2, 5],
        [2, 1, 1],
        [3, 1, 0])$
'B = B:matrix(
        [1], [1], [2])$
'X = X:matrix(
        ['x1], ['x2], ['x3])$
print("Let",'A=A,",",'B=B,",",'X=X)$
print("")$
print("Now, the augmented matrix will be,")$
print("")$
'Aug = Aug:addcol(A,B);
print("")$
print("I. FORWARD ELIMINATION")$
n:length(A[1])$
/* -------- Forward Elimination -------- */
for k:1 thru n-1 do(

    /* Partial Pivoting */
    /* determine the largest element in the column */
    /* and store the row number to max_i */
    max_i: k,
    for i:k thru n do(
        if abs(Aug[i,k]) > abs(Aug[max_i,k]) then
            max_i: i
    ),

    if max_i#k then( /* if row number is not k */
        /* switch rows */
        [Aug[k],Aug[max_i]]:[Aug[max_i],Aug[k]],
        print(""),
        print("=> R",k,"< -- >","R",max_i),
        print(Aug)
    ),

    for i:k+1 thru n do(
        factor: Aug[i,k]/Aug[k,k],
        print(""),
        print("=> R",i,"= R",i,"- (",'Aug[i,k]/'Aug[k,k],")*","R",k),
        /* Applying Ri -> Ri - (Augik/Augkk)*Rk */
        Aug[i]: Aug[i]-factor·Aug[k],
        print(Aug)
    )
)$
print("")$
print("Therefore, the augmented matrix")$
print("reduced to upper triangular form will be,")$
print("")$
Aug;
print("")$
print("Now, the system of equations will be,")$
load("eigen")$
```

6 Given the system of equations
-2x2 - x3 + 8x4 + 9x5 = -38
-9x1 - x2 + 2x3 + 6x4 + 7x5 = -34
-x1 + 5x2 + 8x3 + 4x4 + 8x5 = -20
3x2 - 9x3 + 3x4 - 6x5 = -29
8x1 + 17x2 +8x3 +5x5 = -21
Use Gauss elimination with partial pivoting to solve.

```
→   kill(all)$
    keepfloat:true$
    'A = A:matrix(
            [0, -2, -1, 8, 9],
            [-9, -1, 2, 6, 7],
            [-1, 5, 8, 4, 8],
            [0, 3, -9, 3, -6],
            [8, 17, 8, 0, 5])$
    'B = B:matrix(
            [-38], [-34], [-20], [-29], [-21])$
    'X = X:matrix(
            ['x1], ['x2], ['x3], ['x4], ['x5])$
    print("Let",'A=A,",",'B=B,",",'X=X)$
    print("")$
    print("Now, the augmented matrix will be,")$
    print("")$
    'Aug = Aug:addcol(A,B);
    print("")$
    print("I. FORWARD ELIMINATION")$
    n:length(A[1])$
    /* -------- Forward Elimination -------- */
    for k:1 thru n-1 do(

        /* Partial Pivoting */
        /* determine the largest element in the column */
        /* and store the row number to max_i */
        max_i: k,
        for i:k thru n do(
            if abs(Aug[i,k]) > abs(Aug[max_i,k]) then
                max_i: i
        ),

        if max_i#k then( /* if row number is not k */
            /* switch rows */
            [Aug[k],Aug[max_i]]:[Aug[max_i],Aug[k]],
            print(""),
            print("=> R",k,"< -- >","R",max_i),
            print(Aug)
        ),

        for i:k+1 thru n do(
            factor: Aug[i,k]/Aug[k,k],
            print(""),
            print("=> R",i,"= R",i,"- (",'Aug[i,k]/'Aug[k,k],")*","R",k),
            /* Applying Ri -> Ri - (Augik/Augkk)*Rk */
            Aug[i]: Aug[i]-factor·Aug[k],
            print(Aug)
        )
    )$
    print("")$
    print("Therefore, the augmented matrix")$
    print("reduced to upper triangular form will be,")$
    print("")$
    Aug;
    print("")$
```

Practical 4(b): Gauss-Jordan Elimination Method

Submitted by - Anshul Verma
(19/78065)
BSc (Hons) Computer Science

1 *Solve the Pair of Linear Equation using Gauss Jordan Method:*

$$x + 2y + 6z = 22$$
$$3x + 4y + z = 26$$
$$6x - y - z = 19$$

```
kill(all)$
keepfloat:true$
A:matrix(                    /*...Coefficient Matrix...*/
        [1.0, 2.0, 6.0],
        [3.0, 4.0, 1.0],
        [6.0, -1.0, -1.0])$
B:matrix(                    /*...Constants Matrix...*/
        [22.0], [26.0], [19.0])$
X:matrix(                    /*...Variables Matrix...*/
        [x], [y], [z])$
print("Now, the augmented matrix will be,")$
Aug:addcol(A,B);             /*...Creating Augmented Matrix...*/
print(" ");
print("Now, the Echelon Form is,")$
S : echelon(Aug);            /*..Calculates Echolen Form of Matrix..*/
print(" ");
print("R2 -> R2 - ",float(S[2][3])," * R3")$
S[2] : S[2] - S[2][3].S[3]$
S;
print(" ");
print("R1 -> R1 - ",float(S[1][3])," * R3")$
S[1] : S[1] - S[1][3].S[3]$
S;
print(" ");
print("R1 -> R1 - ",float(S[1][2])," * R2")$
S[1] : S[1] - S[1][2].S[2]$
S;
print(" ");
print("The Solution Matrix: ")$
X=col(S,4);
```

Now, the augmented matrix will be,

(%o6) $\begin{pmatrix} 1.0 & 2.0 & 6.0 & 22.0 \\ 3.0 & 4.0 & 1.0 & 26.0 \\ 6.0 & -1.0 & -1.0 & 19.0 \end{pmatrix}$

(%o7)

Now, the Echelon Form is,

(%o9) $\begin{pmatrix} 1 & 2.0 & 6.0 & 22.0 \\ 0 & 1 & 8.5 & 20.0 \\ 0 & 0 & 1 & 2.0 \end{pmatrix}$

(%o10)

$R2 \rightarrow R2 - \quad 8.5 \quad * R3$

(%o13) $\begin{pmatrix} 1 & 2.0 & 6.0 & 22.0 \\ 0 & 1 & 0.0 & 3.0 \\ 0 & 0 & 1 & 2.0 \end{pmatrix}$

(%o14)

$R1 \rightarrow R1 - \quad 6.0 \quad * R3$

$\begin{pmatrix} 1 & 2.0 & 0.0 & 10.0 \end{pmatrix}$

2 ***Show that the following system of equations have infinite number of solutions :***

$$2x + y - 3z = 0$$
$$5x + 8y + z = 14$$
$$4x + 13y + 11z = 28$$

```
kill(all)$
keepfloat:true$
A:matrix(                    /*...Coefficient Matrix...*/
        [2.0, 1.0, -3.0],
        [5.0, 8.0, 1.0],
        [4.0, 13.0, 11.0])$
B:matrix(                    /*...Constants Matrix...*/
        [0.0], [14.0], [28.0])$
X:matrix(                    /*...Variables Matrix...*/
        [x], [y], [z])$
print("Now, the augmented matrix will be,")$
Aug:addcol(A,B);             /*...Creating Augmented Matrix...*/
print(" ");
print("Now, the Echelon Form is,")$
S : echelon(Aug);            /*..Calculates Echolen Form of Matrix..*/
print(" ");
print("R2 -> R2 - ",float(S[2][3])," * R3")$
S[2] : S[2] - S[2][3].S[3]$
S;
print(" ");
print("R1 -> R1 - ",float(S[1][3])," * R3")$
S[1] : S[1] - S[1][3].S[3]$
S;
print(" ");
print("R1 -> R1 - ",float(S[1][2])," * R2")$
S[1] : S[1] - S[1][2].S[2]$
S;
/*..The last row after solving the matrix consists of all zeors
this shows that the given set of equations has infinite number of
solutions...*/
```

Now, the augmented matrix will be,

(%o6) $$\begin{pmatrix} 2.0 & 1.0 & -3.0 & 0.0 \\ 5.0 & 8.0 & 1.0 & 14.0 \\ 4.0 & 13.0 & 11.0 & 28.0 \end{pmatrix}$$

(%o7)

Now, the Echelon Form is,

(%o9) $$\begin{pmatrix} 1 & 0.5 & -1.5 & 0 \\ 0 & 1 & 1.545454545454545 & 2.545454545454545 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

(%o10)

$R2 \rightarrow R2 - \quad 1.545454545454545 \quad * \ R3$

(%o13) $$\begin{pmatrix} 1 & 0.5 & -1.5 & 0 \\ 0 & 1 & 1.545454545454545 & 2.545454545454545 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

(%o14)

$R1 \rightarrow R1 - \quad -1.5 \quad * \ R3$

$$\begin{pmatrix} 1 & 0.5 & -1.5 & 0 \end{pmatrix}$$

3 Using the Gauss-Jordan method, find the inverse of
([1 1 1]
[4 3 -1]
[3 5 3])

```
kill(all)$
keepfloat:true$
A:matrix(                        /*...Given Matrix...*/
        [1.0, 1.0, 1.0],
        [4.0, 3.0, -1.0],
        [3.0, 5.0, 3.0])$
B:matrix(                        /*...Identity Matrix...*/
        [1.0,0.0,0.0],
        [0.0,1.0,0.0],
        [0.0,0.0,1.0])$
print("Now, the augmented matrix will be,")$
Aug:addcol(A,B);                 /*...Creating Augmented Matrix...*/
print("")$
print("The Echelon Form is :")$
S : echelon(Aug);                /*..Calculates Echolen Form of Matrix..*/
print(" ")$
/*..Operations so as to form reduced row echelon form..*/
print("R2 -> R2 - ",float(S[2][3])," * R3")$
S[2] : S[2] - S[2][3].S[3]$
S;
print(" ")$
print("R1 -> R1 - ",float(S[1][3])," * R3")$
S[1] : S[1] - S[1][3].S[3]$
S;
print(" ")$
print("R1 -> R1 - ",float(S[1][2])," * R2")$
S[1] : S[1] - S[1][2].S[2]$
S;
print(" ")$
print("The Inverse of the Given Matrix is: ")$
Inv: submatrix(S,1,2,3);
```

Now, the augmented matrix will be,

(%o5) $$\begin{pmatrix} 1.0 & 1.0 & 1.0 & 1.0 & 0.0 & 0.0 \\ 4.0 & 3.0 & -1.0 & 0.0 & 1.0 & 0.0 \\ 3.0 & 5.0 & 3.0 & 0.0 & 0.0 & 1.0 \end{pmatrix}$$

The Echelon Form is :

(%o8) $$\begin{pmatrix} 1 & 1.0 & 1.0 & 1.0 & 0 & 0 \\ 0 & 1 & 0 & -1.5 & 0 & 0.5 \\ 0 & 0 & 1 & 1.1 & -0.2 & -0.1 \end{pmatrix}$$

$R2 \to R2 - \quad 0.0 \quad * \ R3$

(%o12) $$\begin{pmatrix} 1 & 1.0 & 1.0 & 1.0 & 0 & 0 \\ 0 & 1 & 0 & -1.5 & 0 & 0.5 \\ 0 & 0 & 1 & 1.1 & -0.2 & -0.1 \end{pmatrix}$$

$R1 \to R1 - \quad 1.0 \quad * \ R3$

(%o16) $$\begin{pmatrix} 1 & 1.0 & 0.0 & -0.1000000000000001 & 0.2 & 0.1 \\ 0 & 1 & 0 & -1.5 & 0 & 0.5 \\ 0 & 0 & 1 & 1.1 & -0.2 & -0.1 \end{pmatrix}$$

4 *Solve the system of equations*

$2y + 3z = 8,$

$4x + 6y + 7z = -3,$

$2x - 3y + 6z = 5,$

using Gauss Jordan method.

```
kill(all)$
keepfloat:true$
A:matrix(                    /*...Coefficient Matrix...*/
        [0, 2.0, 3.0],
        [4.0, 6.0, 7.0],
        [2.0, -3.0, 6.0])$
B:matrix(                    /*...Constants Matrix...*/
        [8.0], [-3.0], [5.0])$
X:matrix(                    /*...Variables Matrix...*/
        [x], [y], [z])$
print("Now, the augmented matrix will be,")$
Aug:addcol(A,B);             /*...Creating Augmented Matrix...*/
print(" ");
print("Now, the Echelon Form is,")$
S : echelon(Aug);            /*..Calculates Echolen Form of Matrix..*/
print(" ");
print("R2 -> R2 - ",float(S[2][3])," * R3")$
S[2] : S[2] - S[2][3].S[3]$
S;
print(" ");
print("R1 -> R1 - ",float(S[1][3])," * R3")$
S[1] : S[1] - S[1][3].S[3]$
S;
print(" ");
print("R1 -> R1 - ",float(S[1][2])," * R2")$
S[1] : S[1] - S[1][2].S[2]$
S;
print(" ");
print("The Solution Matrix: ")$
X=col(S,4);
```

Now, the augmented matrix will be,

(%o6) $\begin{pmatrix} 0 & 2.0 & 3.0 & 8.0 \\ 4.0 & 6.0 & 7.0 & -3.0 \\ 2.0 & -3.0 & 6.0 & 5.0 \end{pmatrix}$

(%o7)

Now, the Echelon Form is,

(%o9) $\begin{pmatrix} 1 & 1.5 & 1.75 & -0.75 \\ 0 & 1 & 1.5 & 4.0 \\ 0 & 0 & 1 & 2.652173913043478 \end{pmatrix}$

(%o10)

$R2 \rightarrow R2 - \ 1.5 \ * R3$

(%o13) $\begin{pmatrix} 1 & 1.5 & 1.75 & -0.75 \\ 0 & 1 & 0.0 & 0.02173913043478315 \\ 0 & 0 & 1 & 2.652173913043478 \end{pmatrix}$

(%o14)

$R1 \rightarrow R1 - \ 1.75 \ * R3$

$\begin{pmatrix} 1 & 1.5 & 0.0 & -5.391304347826087 \end{pmatrix}$

5 Using the Gauss-Jordan method, find the inverse of

([4 8 6 3]
[6 3 -1 9]
[7 5 3 3]
[-3 6 1 -1])

```
kill(all)$
keepfloat:true$
A:matrix(                    /*...Given Matrix...*/
        [4.0, 8.0, 6.0, 3.0],
        [6.0, 3.0, -1.0, 9.0],
        [7.0, 5.0, 3.0, 3.0],
        [-3.0, 6.0, 1.0, -1.0])$
B:matrix(                    /*...Identity Matrix...*/
        [1.0,0.0,0.0,0.0],
        [0.0,1.0,0.0,0.0],
        [0.0,0.0,1.0,0.0],
        [0.0,0.0,0.0,1.0])$
print("Now, the augmented matrix will be,")$
Aug:addcol(A,B);             /*...Creating Augmented Matrix...*/
print("")$
print("The Echelon Form is :")$
S : echelon(Aug);            /*..Calculates Echolen Form of Matrix..*/
print(" ")$
/*..Operations so as to form reduced row echelon form..*/
print("R2 -> R2 - ",float(S[2][3])," * R3")$
S[2] : S[2] - S[2][3].S[3]$
S;
print(" ")$
print("R1 -> R1 - ",float(S[1][3])," * R3")$
S[1] : S[1] - S[1][3].S[3]$
S;
print(" ")$
print("R1 -> R1 - ",float(S[1][2])," * R2")$
S[1] : S[1] - S[1][2].S[2]$
S;
print(" ")$
print("The Inverse of the Given Matrix is: ")$
Inv: submatrix(S,1,2,3);
```

Now, the augmented matrix will be,

(%o5)
$$\begin{pmatrix} 4.0 & 8.0 & 6.0 & 3.0 & 1.0 & 0.0 & 0.0 & 0.0 \\ 6.0 & 3.0 & -1.0 & 9.0 & 0.0 & 1.0 & 0.0 & 0.0 \\ 7.0 & 5.0 & 3.0 & 3.0 & 0.0 & 0.0 & 1.0 & 0.0 \\ -3.0 & 6.0 & 1.0 & -1.0 & 0.0 & 0.0 & 0.0 & 1.0 \end{pmatrix}$$

The Echelon Form is :

(%o8)
$$\begin{pmatrix} 1 & 2.0 & 1.5 & 0.75 & 0.25 & 0 & 0 \\ 0 & 1 & 1.111111111111111 & -0.5 & 0.1666666666666667 & -0.1111111111111111 & 0 \\ 0 & 0 & 1 & -2.7 & -0.1 & -0.4 & 0.4 \\ 0 & 0 & 0 & 1 & 0.1462829736211031 & 0.1294964028776978 & -0.225419664 \end{pmatrix}$$

*R2 → R2 - 1.111111111111111 * R3*

(%o12)
$$\begin{pmatrix} 1 & 2.0 & 1.5 & 0.75 & 0.25 & 0 & 0 \\ 0 & 1 & 0.0 & 2.5 & 0.2777777777777778 & 0.3333333333333334 & -0.4444444444444445 \end{pmatrix}$$

Practical 5(a): To solve system of linear equations using Gauss-Jacobi method

Submitted by - Anshul Verma
(19/78065)
BSc (Hons) Computer Science

1 *Q: Solve the following system of linear equations using Gauss-Jacobi method:*
5x1+x2+2x3=10
3x1-9x2-4x3=14
x1+2x2-7x3=-33

Solution: Method 1:

```
kill(all)$
x10=x10:0.0;
x20=x20:0.0;
x30=x30:0.0;
print("itr","    ","","","","x1","   ","   ","    ","    ","x2","
","  ","  ","  ","x3")$
for i:1 thru 14 do(
x1:(10-x20-2.0·x30)/5,
x2:(-14+3·x10-4·x30)/9,
x3:(33+x10+2·x20)/7,
print(i," "," "," ",x1,"","","",x2,"","","",x3),
x10:x1,
x20:x2,
x30:x3)$
print("x1=",x1)$
print("x2=",x2)$
print("x3=",x3)$
```

(%o1) $x10=0.0$

(%o2) $x20=0.0$

(%o3) $x30=0.0$

```
itr        x1                    x2               x3
1        2.0      -1.555555555555556
4.714285714285714
2        0.4253968253968253      -2.984126984126984
4.555555555555555
3        0.7746031746031747      -3.438447971781305
3.922448979591837
4        1.118710002519526      -3.040665154950869
3.842529604434366
5        1.071121189216427      -2.890443156686542
4.005339956088256
6        0.9759526489020061      -2.97866625074486
4.041462125120478
7        0.979148400100781      -3.026443394863988
4.002660021058898
8        1.004224670549239      -3.008132764881472
3.989465944338972
9        1.005840175240706      -2.993909973967575
3.998279877255186
10        0.9994700438914409      -2.997288775922069
 4.002574318186508
11        0.9984280279098104      -3.001320793452412
 4.000698927435329
12        0.9999845877163509      -3.000834625112432
 3.999398063000712
13        1.000407699822202      -2.999737609872644
3.999759333927355
14        1.000043788403587      -2.999757137360313
4.000133211439559
x1= 1.000043788403587
x2= -2.999757137360313
x3= 4.000133211439559
```

Method 2:

```
kill(all)$'n=n:3;
'a=a:matrix([5,1,2],[3,-9,-4],[1,2,-7]);
'x=x:matrix([0],[0],[0]);'b=b:matrix([10],[14],[-33]);
print("itr","    ","","","","x1","   ","   ","    ","    ","x2","  ","
for k:1 thru 14 do(
for i:1 thru n do(
y[i]:float((b[i]-sum(a[i,j]·x[j],j,1,i-1)-sum(a[i,j]·x[j],j,i+1,n))/a[
for i:1 thru n do (
x[i]:y[i]),print(k,"","","",x[1],"","","",x[2],"","","",x[3]))$
for p:1 thru n do print('x[p]=x[p])$;
```

(%o1) $n=3$

(%o2) $a=\begin{pmatrix}5 & 1 & 2\\ 3 & -9 & -4\\ 1 & 2 & -7\end{pmatrix}$

(%o3) $x=\begin{pmatrix}0\\ 0\\ 0\end{pmatrix}$

(%o4) $b=\begin{pmatrix}10\\ 14\\ -33\end{pmatrix}$

itr x1 x2 x3
1 [2.0] [-1.555555555555556] [
4.714285714285714]
2 [0.4253968253968253] [-2.984126984126984]
[4.555555555555555]
3 [0.7746031746031747] [-3.438447971781305]
[3.922448979591836]
4 [1.118710002519526] [-3.040665154950869]
[3.842529604434366]
5 [1.071121189216428] [-2.890443156686542]
[4.005339956088256]
6 [0.9759526489020062] [-2.97866625074486]
[4.041462125120478]
7 [0.979148400100781] [-3.026443394863988]
[4.002660021058897]
8 [1.004224670549239] [-3.008132764881472]
[3.989465944338972]
9 [1.005840175240706] [-2.993909973967575]
[3.998279877255185]
10 [0.9994700438914411] [-2.997288775922069]
[4.002574318186507]
11 [0.9984280279098109] [-3.001320793452412]
[4.000698927435329]
12 [0.9999845877163506] [-3.000834625112431]
[3.999398063000712]
13 [1.000407699822201] [-2.999737609872644]
[3.999759333927355]
14 [1.000043788403587] [-2.999757137360313]
[4.000133211439558]

2 *Solve the system of equations*
4x1 + x2 + x3 = 2
x1 + 5x2 + 2x3 = – 6
x1 + 2x2 + 3x3 = – 4
using the Jacobi iteration method.

```
kill(all)$'n=n:3;
'a=a:matrix([4,1,1],[1,5,2],[1,2,3]);
'x=x:matrix([0],[0],[0]);'b=b:matrix([2],[-6],[-4]);
print("itr","    ","","","","x1","   ","   ","    ","    ","x2","  ","
for k:1 thru 14 do(
for i:1 thru n do(
y[i]:float((b[i]-sum(a[i,j]·x[j],j,1,i-1)-sum(a[i,j]·x[j],j,i+1,n))/a[
for i:1 thru n do (
x[i]:y[i]),print(k,"","","",x[1],"","","",x[2],"","","",x[3]))$
for p:1 thru n do print('x[p]=x[p])$;
```

(%o1) $n=3$

(%o2) $a=\begin{pmatrix}4 & 1 & 1\\1 & 5 & 2\\1 & 2 & 3\end{pmatrix}$

(%o3) $x=\begin{pmatrix}0\\0\\0\end{pmatrix}$

(%o4) $b=\begin{pmatrix}2\\-6\\-4\end{pmatrix}$

itr x1 x2 x3
1 [0.5] [-1.2] [-1.333333333333333]
2 [1.133333333333333] [-0.7666666666666667]
[-0.7]
3 [0.8666666666666667] [-1.146666666666667]
[-1.2]
4 [1.086666666666666] [-0.8933333333333335]
[-0.8577777777777779]
5 [0.9377777777777778] [-1.074222222222222]
[-1.1]
6 [1.043555555555555] [-0.9475555555555557]
[-0.9297777777777778]
7 [0.9693333333333334] [-1.0368] [-
1.049481481481481]
8 [1.02157037037037] [-0.9740740740740741]
[-0.9652444444444443]
9 [0.9848296296296296] [-1.018216296296296]
[-1.024474074074074]
10 [1.010672592592593] [-0.9871762962962966]
[-0.9827901234567789]
11 [0.9924938271604938] [-1.009014913580247]
[-1.012106666666666]
12 [1.005280395061728] [-0.9936560987654321]
[-0.9914879999999999]
13 [0.9962860246913581] [-1.004460879012346]
[-1.005989399176955]
14 [1.002612569547325] [-0.9968614452674898]
[-0.9957880888888889]

$x_1=[1.002612569547325]$

3 The following system of equations is designed to determine concentrations (the c's in g/m3) in a series of coupled reactors as a function of the amount of mass input to each reactor (the right-hand sides in g/day):

$$15c_1 - 3c_2 - c_3 = 4000$$
$$-3c_1 + 18c_2 - 6c_3 = 1500$$
$$-4c_1 - c_2 + 12c_3 = 2400$$

Determine the solution.

```
→ kill(all)$'n=n:3;
'a=a:matrix([15,-3,-1],[-3,18,-6],[-4,-1,12]);
'C=x:matrix([0],[0],[0]);'b=b:matrix([4000],[1500],[2400]);
print("itr","","","","","solution")$
print("itr","    ","","","","c1","   ","   ","    ","    ","c2","  ","
for k:1 thru 14 do(
for i:1 thru n do(
y[i]:float((b[i]-sum(a[i,j]·x[j],j,1,i-1)-sum(a[i,j]·x[j],j,i+1,n))/a[
for i:1 thru n do (
x[i]:y[i]),print(k,"","","",x[1],"","","",x[2],"","","",x[3]))$
for p:1 thru n do print('c[p]=x[p])$;
```

(%o1) $n=3$

(%o2) $a=\begin{pmatrix}15 & -3 & -1\\ -3 & 18 & -6\\ -4 & -1 & 12\end{pmatrix}$

(%o3) $C=\begin{pmatrix}0\\ 0\\ 0\end{pmatrix}$

(%o4) $b=\begin{pmatrix}4000\\ 1500\\ 2400\end{pmatrix}$

```
itr        solution
itr            c1                     c2                   c3
1       [266.6666666666667]      [83.33333333333333]
[200.0]
2       [296.6666666666667]      [194.4444444444444]
[295.8333333333333]
3       [325.2777777777777]      [231.3888888888889]
[315.0925925925926]
4       [333.9506172839506]      [242.5771604938271]
[327.7083333333333]
5       [337.0293209876543]      [248.2278806584362]
[331.5316358024691]
6       [338.4143518518518]      [250.0154320987654]
[333.0287637174211]
7       [338.8716706675812]      [250.745313214449]       [
333.639403292181]
8       [339.0583561957018]      [251.0250795419905]
[333.8526663237311]
9       [339.1285269966468]      [251.1272814738606]
[333.9382086937331]
10       [339.1546702076877]      [251.1674907306855]
 [333.9701157883707]
11       [339.1648391986952]      [251.1824836307381]
 [333.982180963453]
12       [339.1686421237112]      [251.1882001876002]
 [333.9868200354599]
13       [339.1700947065507]      [251.1903803657718]
 [333.9885640568704]
```

4 Mass balances can be written for each reservoir, and the following set of simultaneous linear algebraic equations results:

[[13.422 0 0 0],
[−13.422 12.252 0 0],
[0 −12.252 12.377 0],
[0 0 −12.377 11.797]]
×
[[c1],[c2],[c3],[c4]]
=
[[750.5],[300],[102],[30]]

where the right-hand-side vector consists of the loadings of chloride to each of the four lakes and c
1, c2, c3, and c4 = the resulting chloride concentrations for Lakes Powell, Mead, Mohave, and Havasu, respectively.
Solve for the concentrations in each of the four lakes.

```
kill(all)$'n=n:4;
'a=a:matrix([13.422,0,0,0],[-13.422,12.252,0,0],[0,-12.252,12.377,0],
    [0,0,-12.377,11.797]);
'C=x:matrix([0],[0],[0],[0]);'b=b:matrix([750.5],[300],[102],[30]);
print("itr","","","","","solution")$
print("itr","    ","","","","c1","   ","   ","    ","    ","c2","  ","
for k:1 thru 14 do(
for i:1 thru n do(
y[i]:float((b[i]-sum(a[i,j]·x[j],j,1,i-1)-sum(a[i,j]·x[j],j,i+1,n))/a[
for i:1 thru n do (
x[i]:y[i]),print(k,"","","",x[1],"","","",x[2],"","","",x[3]))$
for p:1 thru n do print('c[p]=x[p])$;
```

(%o1) $n=4$

(%o2) $a=\begin{pmatrix} 13.422 & 0 & 0 & 0 \\ -13.422 & 12.252 & 0 & 0 \\ 0 & -12.252 & 12.377 & 0 \\ 0 & 0 & -12.377 & 11.797 \end{pmatrix}$

(%o3) $C=\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$

(%o4) $b=\begin{pmatrix} 750.5 \\ 300 \\ 102 \\ 30 \end{pmatrix}$

```
itr      solution
itr          c1                       c2                  c3
1      [55.91566085531218]      [24.48579823702253]
[8.24109234871132]
2      [55.91566085531218]      [85.74110349330721]
[32.47959925668579]
3      [55.91566085531218]      [85.74110349330721]
[93.11626403813526]
4      [55.91566085531218]      [85.74110349330721]
[93.11626403813526]
5      [55.91566085531218]      [85.74110349330721]
[93.11626403813526]
6      [55.91566085531218]      [85.74110349330721]
[93.11626403813526]
7      [55.91566085531218]      [85.74110349330721]
[93.11626403813526]
8      [55.91566085531218]      [85.74110349330721]
[93.11626403813526]
9      [55.91566085531218]      [85.74110349330721]
[93.11626403813526]
10      [55.91566085531218]      [85.74110349330721]
 [93.11626403813526]
11      [55.91566085531218]      [85.74110349330721]
```

5 Solve the system of equations

$x + 3y + 52z = 173.61$,
$x - 27y + 2z = 71.31$,
$41x - 2y + 3z = 65$
using the Jacobi iteration method.

```
kill(all)$'n=n:3;
'a=a:matrix([1,3,52],[1,-27,2],[41,-2,3]);
'x=x:matrix([0],[0],[0]);'b=b:matrix([173.61],[71.31],[65]);
print("itr","    ","","","","x1","   ","   ","    ","    ","x2","  ","
for k:1 thru 14 do(
for i:1 thru n do(
y[i]:float((b[i]-sum(a[i,j]·x[j],j,1,i-1)-sum(a[i,j]·x[j],j,i+1,n))/a[
for i:1 thru n do (
x[i]:y[i]),print(k,"","","",x[1],"","","",x[2],"","","",x[3]))$
for p:1 thru n do print('x[p]=x[p])$;
```

(%o1) $n=3$

(%o2) $a=\begin{pmatrix}1 & 3 & 52\\ 1 & -27 & 2\\ 41 & -2 & 3\end{pmatrix}$

(%o3) $x=\begin{pmatrix}0\\ 0\\ 0\end{pmatrix}$

(%o4) $b=\begin{pmatrix}173.61\\ 71.31\\ 65\end{pmatrix}$

itr x1 x2 x3
1 [173.61] [-2.641111111111111] [21.66666666666667]
2 [-945.1333333333333] [5.393827160493828] [-2352.764074074074]
3 [122501.1603703704] [-211.9248696844993] [12942.08477366255]
4 [-672179.0236213992] [5493.111848803535] [-1674302.141641518]
5 [$8.704740563981251\ 10^{7}$] [-148920.5413668309] [9190130.397391658]
6 [$-4.774398454302657\ 10^{8}$] [3904725.745355401] [$-1.189747135771682\ 10^{9}$]
7 [$6.185513705650141\ 10^{10}$] [$-1.058123773438382\ 10^{8}$] [$6.527614393043869\ 10^{9}$]
8 [$-3.391185111326397\ 10^{11}$] [$2.774457991528858\ 10^{9}$] [$-8.454240813354152\ 10^{11}$]
9 [$4.395372885564061\ 10^{13}$] [$-7.518395088425112\ 10^{10}$] [$4.636469290828761\ 10^{12}$]
10 [$-2.408708512702692\ 10^{14}$] [$1.971358053230623\ 10^{12}$] [$-6.007510836609895\ 10^{14}$]
11 [$3.123314227621193\ 10^{16}$] [$-5.342122291082665\ 10^{13}$] [$3.293215872729189\ 10^{15}$]

Practical 5(b): To solve system of linear equations using Gauss-Seidel method

Submitted by - Anshul Verma
(19/78065)
BSc (Hons) Computer Science

1 *Q. Solve the following system of linear equations using Gauss-Seidel method:*
5x1+x2+2x3=10
3x1-9x2-4x3=14
x1+2x2-7x3=-33

Solution: Method 1:

```
kill(all)$
x1:0.0;
x2:0.0;
x3:0.0;
print("itr","","","","","","","","solution")$
for i:1 thru 10 do(
x1:(10-x2-2.0·x3)/5,
x2:(-14+3·x1-4·x3)/9,
x3:(33+x1+2·x2)/7,
print(i,"","","","x1=",x1," x2=",x2," x3=",x3))$
print("x1=",x1)$
print("x2=",x2)$
print("x3=",x3)$
```

```
(%o1) 0.0
(%o2) 0.0
(%o3) 0.0
itr               solution
1       x1= 2.0   x2= -0.8888888888888888   x3=
4.746031746031746
2       x1= 0.2793650793650794   x2= -3.571781305114639
 x3= 3.7336860670194
3       x1= 1.220881834215168   x2= -2.8080109739369   x3=
 4.086408555191624
4       x1= 0.9270387727107305   x2= -3.062724211403812
 x3= 3.971655764271873
5       x1= 1.023882536572013   x2= -2.979441716374606
 x3= 4.0092855862604
6       x1= 0.9921741087707613   x2= -3.00673555763659
 x3= 3.996957570499654
7       x1= 1.002564083327456   x2= -2.997793114668472
 x3= 4.000996836284359
8       x1= 0.9991598884199508   x2= -3.000723075541954
 x3= 3.999673391048006
9       x1= 1.000275258689188   x2= -2.999763087569384
 x3= 4.000107011935774
10       x1= 0.9999098127395672   x2= -3.000077623280488
 x3= 3.999964938025513
x1= 0.9999098127395672
x2= -3.000077623280488
x3= 3.999964938025513
```

Method 2:

```
kill(all)$
'n=n:3;
'a=a:matrix([5,1,2],[3,-9,-4],[1,2,-7]);
'x=x:matrix([0],[0],[0]);'b=b:matrix([10],[14],[-33]);
print("itr","","","","","solution")$
for k:1 thru 10 do(
for i:1 thru n do (
 x[i]:float((b[i]-sum(a[i,j].x[j],j,1,i-1)-sum(a[i,j].x[j],j,i+1,n))/a
print(k,"","","", 'x[1]=x[1],'x[2]=x[2],'x[3]=x[3]))$
for p:1 thru n do print('x[p]=x[p])$
```

(%o1) $n=3$

(%o2) $a=\begin{pmatrix} 5 & 1 & 2 \\ 3 & -9 & -4 \\ 1 & 2 & -7 \end{pmatrix}$

(%o3) $x=\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$

(%o4) $b=\begin{pmatrix} 10 \\ 14 \\ -33 \end{pmatrix}$

itr solution

1 $x_1=[2.0]$ $x_2=[-0.8888888888888888]$ $x_3=[4.746031746031746]$

2 $x_1=[0.2793650793650794]$ $x_2=[-3.571781305114638]$ $x_3=[3.7336860670194]$

3 $x_1=[1.220881834215167]$ $x_2=[-2.808010973936899]$ $x_3=[4.086408555191624]$

4 $x_1=[0.9270387727107304]$ $x_2=[-3.062724211403812]$ $x_3=[3.971655764271873]$

5 $x_1=[1.023882536572013]$ $x_2=[-2.979441716374606]$ $x_3=[4.0092855862604]$

6 $x_1=[0.9921741087707613]$ $x_2=[-3.00673555763659]$ $x_3=[3.996957570499654]$

7 $x_1=[1.002564083327456]$ $x_2=[-2.997793114668472]$ $x_3=[4.000996836284359]$

8 $x_1=[0.9991598884199508]$ $x_2=[-3.000723075541953]$ $x_3=[3.999673391048006]$

9 $x_1=[1.000275258689188]$ $x_2=[-2.999763087569384]$ $x_3=[4.000107011935774]$

10 $x_1=[0.9999098127395674]$ $x_2=[-3.000077623280488]$ $x_3=[3.999964938025513]$

$x_1=[0.9999098127395674]$

$x_2=[-3.000077623280488]$

$x_3=[3.999964938025513]$

2 Q. Solve the following system of linear equations using Gauss-Seidel method:

$$10x1 + 2x2 - x3 = 27$$

$$-3x1 - 6x2 + 2x3 = -61.5$$

$$x1 + x2 + 5x3 = -21.5$$

```
kill(all)$
'n=n:3;
'a=a:matrix([10,2,-1],[-3,-6,2],[1,1,5]);
'x=x:matrix([0],[0],[0]);'b=b:matrix([27],[-61.5],[-21.5]);
print("itr","","","","","solution")$
for k:1 thru 10 do(
for i:1 thru n do (
 x[i]:float((b[i]-sum(a[i,j].x[j],j,1,i-1)-sum(a[i,j].x[j],j,i+1,n))/a
print(k,"","","", 'x[1]=x[1],'x[2]=x[2],'x[3]=x[3]))$
for p:1 thru n do print('x[p]=x[p])$
```

(%o1) $n=3$

(%o2) $a=\begin{pmatrix}10 & 2 & -1\\ -3 & -6 & 2\\ 1 & 1 & 5\end{pmatrix}$

(%o3) $x=\begin{pmatrix}0\\ 0\\ 0\end{pmatrix}$

(%o4) $b=\begin{pmatrix}27\\ -61.5\\ -21.5\end{pmatrix}$

itr solution

1 $x_1=[2.7]$ $x_2=[8.899999999999999]$ $x_3=[-6.619999999999999]$

2 $x_1=[0.2580000000000006]$ $x_2=[7.914333333333333]$ $x_3=[-5.934466666666667]$

3 $x_1=[0.5236866666666664]$ $x_2=[8.010001111111111]$ $x_3=[-6.006737555555556]$

4 $x_1=[0.4973260222222223]$ $x_2=[7.999091137037036]$ $x_3=[-5.999283431851852]$

5 $x_1=[0.5002534294074075]$ $x_2=[8.000112141345678]$ $x_3=[-6.000073114150617]$

6 $x_1=[0.4999702603158028]$ $x_2=[7.999990498458559]$ $x_3=[-5.999992151754873]$

7 $x_1=[0.5000026851328009]$ $x_2=[8.000001273515307]$ $x_3=[-6.000000791729622]$

8 $x_1=[0.4999996661239763]$ $x_2=[7.999999903028137]$ $x_3=[-5.999999913830423]$

9 $x_1=[0.5000000280113301]$ $x_2=[8.000000014717527]$ $x_3=[-6.000000008545772]$

10 $x_1=[0.4999999962019175]$ $x_2=[7.99999999905045]$ $x_3=[-5.999999999050473]$

$x_1=[0.4999999962019175]$

$x_2=[7.99999999905045]$

$x_3=[-5.999999999050473]$

3 *Q. Solve the following system of linear equations using Gauss-Seidel method:*

$$-8x1 + x2 - 2x3 = -20$$

$$2x1 - 6x2 - x3 = -38$$

$$-3x1 - x2 + 7x3 = -34$$

```
kill(all)$
'n=n:3;
'a=a:matrix([-8,1,-2],[2,-6,-1],[-3,-1,7]);
'x=x:matrix([0],[0],[0]);'b=b:matrix([-20],[-38],[-34]);
print("itr","","","","","solution")$
for k:1 thru 10 do(
for i:1 thru n do (
 x[i]:float((b[i]-sum(a[i,j].x[j],j,1,i-1)-sum(a[i,j].x[j],j,i+1,n))/a
print(k,"","","", 'x[1]=x[1],'x[2]=x[2],'x[3]=x[3]))$
for p:1 thru n do print('x[p]=x[p])$
```

(%o1) $n=3$

(%o2) $a=\begin{pmatrix} -8 & 1 & -2 \\ 2 & -6 & -1 \\ -3 & -1 & 7 \end{pmatrix}$

(%o3) $x=\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$

(%o4) $b=\begin{pmatrix} -20 \\ -38 \\ -34 \end{pmatrix}$

itr solution

1 $x_1=[2.5]$ $x_2=[7.166666666666666]$ $x_3=[-2.761904761904762]$

2 $x_1=[4.086309523809524]$ $x_2=[8.155753968253968]$ $x_3=[-1.940759637188209]$

3 $x_1=[4.004659155328798]$ $x_2=[7.9916796579743]$ $x_3=[-1.999191839434186]$

4 $x_1=[3.998757917105334]$ $x_2=[7.999451278940809]$ $x_3=[-2.000610709963312]$

5 $x_1=[4.000084087358429]$ $x_2=[8.000129814113361]$ $x_3=[-1.999945417687336]$

6 $x_1=[4.000002581186004]$ $x_2=[7.999991763343223]$ $x_3=[-2.000000704442681]$

7 $x_1=[3.999998988028573]$ $x_2=[7.999996744166637]$ $x_3=[-2.000000480213949]$

8 $x_1=[4.000000079355567]$ $x_2=[8.000000106487512]$ $x_3=[-1.999999950777969]$

9 $x_1=[4.000000001005431]$ $x_2=[7.999999992131472]$ $x_3=[-2.000000000693177]$

10 $x_1=[3.999999999189728]$ $x_2=[7.999999999845439]$ $x_3=[-2.00000000036934]$

$x_1=[3.999999999189728]$

$x_2=[7.999999999845439]$

$x_3=[-2.00000000036934]$

4 The following system of equations is designed to determine concentrations (the c's in g/m3) in a series of coupled reactors as a function of the amount of mass input to each reactor (the right-hand sides in g/day):

$$15c1 - 3c2 - c3 = 4000$$
$$-3c1 + 18c2 - 6c3 = 1500$$
$$-4c1 - c2 + 12c3 = 2400$$

Determine the solution.

```
kill(all)$
'n=n:3;
'a=a:matrix([15,-3,-1],[-3,18,-6],[-4,-1,12]);
'C=x:matrix([0],[0],[0]);'b=b:matrix([4000],[1500],[2400]);
print("itr","","","","","solution")$
for k:1 thru 10 do(
for i:1 thru n do (
 x[i]:float((b[i]-sum(a[i,j].x[j],j,1,i-1)-sum(a[i,j].x[j],j,i+1,n))/a
print(k,"","","", 'c[1]=x[1],'c[2]=x[2],'c[3]=x[3]))$
for p:1 thru n do print('c[p]=x[p])$
```

(%o1) $n=3$

(%o2) $a=\begin{pmatrix} 15 & -3 & -1 \\ -3 & 18 & -6 \\ -4 & -1 & 12 \end{pmatrix}$

(%o3) $C=\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$

(%o4) $b=\begin{pmatrix} 4000 \\ 1500 \\ 2400 \end{pmatrix}$

itr solution

1 $c_1=[266.6666666666667]$ $c_2=[127.7777777777778]$ $c_3=[299.537037037037]$

2 $c_1=[312.1913580246913]$ $c_2=[235.2109053497942]$ $c_3=[323.66469478738]$

3 $c_1=[335.2864940557841]$ $c_2=[247.1026472717573]$ $c_3=[332.3540519579078]$

4 $c_1=[338.244132918212]$ $c_2=[250.4920394723379]$ $c_3=[333.6223809287654]$

5 $c_1=[339.0065666230519]$ $c_2=[251.0418880800971]$ $c_3=[333.9223462143587]$

6 $c_1=[339.13653403031]$ $c_2=[251.1635377431712]$ $c_3=[333.9758061553676]$

7 $c_1=[339.1644279589921]$ $c_2=[251.1860067116212]$ $c_3=[333.9869765456324]$

8 $c_1=[339.1696664453664]$ $c_2=[251.1906032561052]$ $c_3=[333.9891057531309]$

9 $c_1=[339.1707277014297]$ $c_2=[251.1914898679486]$ $c_3=[333.9895333894723]$

10 $c_1=[339.1709335328879]$ $c_2=[251.1916667186387]$ $c_3=[333.9896167375158]$

$c_1=[339.1709335328879]$

$c_2=[251.1916667186387]$

$c_3=[333.9896167375158]$

5 Mass balances can be written for each reservoir, and the following set of simultaneous linear algebraic equations results:

[[13.422 0 0 0],
[−13.422 12.252 0 0],
[0 −12.252 12.377 0],
[0 0 −12.377 11.797]]
×
[[c1],[c2],[c3],[c4]]
=
[[750.5],[300],[102],[30]]

where the right-hand-side vector consists of the loadings of chloride to each of the four lakes and c
1, c2, c3, and c4 = the resulting chloride concentrations for Lakes Powell, Mead, Mohave, and Havasu, respectively.
Solve for the concentrations in each of the four lakes.

```
kill(all)$
'n=n:4;
'a=a:matrix([13.422,0,0,0],[-13.422,12.252,0,0],[0,-12.252,12.377,0],
    [0,0,-12.377,11.797]);
'C=x:matrix([0],[0],[0],[0]);'b=b:matrix([750.5],[300],[102],[30]);
print("itr","","","","","solution")$
for k:1 thru 10 do(
for i:1 thru n do (
 x[i]:float((b[i]-sum(a[i,j].x[j],j,1,i-1)-sum(a[i,j].x[j],j,i+1,n))/a
print(k,"","","", 'c[1]=x[1],'c[2]=x[2],'c[3]=x[3]))$
for p:1 thru n do print('c[p]=x[p])$
```

(%o1) $n=4$

(%o2) $a=\begin{pmatrix} 13.422 & 0 & 0 & 0 \\ -13.422 & 12.252 & 0 & 0 \\ 0 & -12.252 & 12.377 & 0 \\ 0 & 0 & -12.377 & 11.797 \end{pmatrix}$

(%o3) $C=\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$

(%o4) $b=\begin{pmatrix} 750.5 \\ 300 \\ 102 \\ 30 \end{pmatrix}$

itr solution

1 $c_1=[55.91566085531218]$ $c_2=[85.74110349330721]$ $c_3=[93.11626403813526]$

2 $c_1=[55.91566085531218]$ $c_2=[85.74110349330721]$ $c_3=[93.11626403813526]$

3 $c_1=[55.91566085531218]$ $c_2=[85.74110349330721]$ $c_3=[93.11626403813526]$

4 $c_1=[55.91566085531218]$ $c_2=[85.74110349330721]$ $c_3=[93.11626403813526]$

5 $c_1=[55.91566085531218]$ $c_2=[85.74110349330721]$ $c_3=[93.11626403813526]$

6 $c_1=[55.91566085531218]$ $c_2=[85.74110349330721]$ $c_3=[93.11626403813526]$

7 $c_1=[55.91566085531218]$ $c_2=[85.74110349330721]$ $c_3=[93.11626403813526]$

8 $c_1=[55.91566085531218]$ $c_2=[85.74110349330721]$ $c_3=[93.11626403813526]$

9 $c_1=[55.91566085531218]$ $c_2=[85.74110349330721]$ $c_3=[93.11626403813526]$

10 $c_1=[55.91566085531218]$ $c_2=[85.74110349330721]$ $c_3=[93.11626403813526]$

$c_1=[55.91566085531218]$

Practical 6(a): Lagrange Interpolation

Submitted by - Anshul Verma
(19/78065)
BSc (Hons) Computer Science

Using Iterations

1 *Construct the Lagrange Interpolation Polynomial for the data.*

x	-1	1	4	7
f(x)	-2	0	63	342

Hence, interpolate at x = 5.

```
kill(all)$
p = p: [
    [-1, -2],
    [1, 0],
    [4, 63],
    [7, 342]
];
n: length(p)$
Y: 0$
for i: 1 thru n do (
    l_i: 1,
    for j: 1 thru n do (
        if notequal(i, j) then
            l_i: l_i · (x - p[j][1]) / (p[i][1] - p[j][1])
    ),
    Y: Y + l_i · p[i][2],
    print("iteration", i, "=>", Y, "=>", expand(Y))
)$
'f(x) = f: expand(Y);
print("f(5) =", ev(f, x = 5))$
wxplot2d([f, [discrete, map(first, p), map(second, p)]], [x, -2, 10],
```

(%o1) $p=[[-1,-2],[1,0],[4,63],[7,342]]$

iteration 1 => $\frac{(x-7)(x-4)(x-1)}{40}$ => $\frac{x^3}{40}-\frac{3x^2}{10}+\frac{39x}{40}-\frac{7}{10}$

iteration 2 => $\frac{(x-7)(x-4)(x-1)}{40}$ => $\frac{x^3}{40}-\frac{3x^2}{10}+\frac{39x}{40}-\frac{7}{10}$

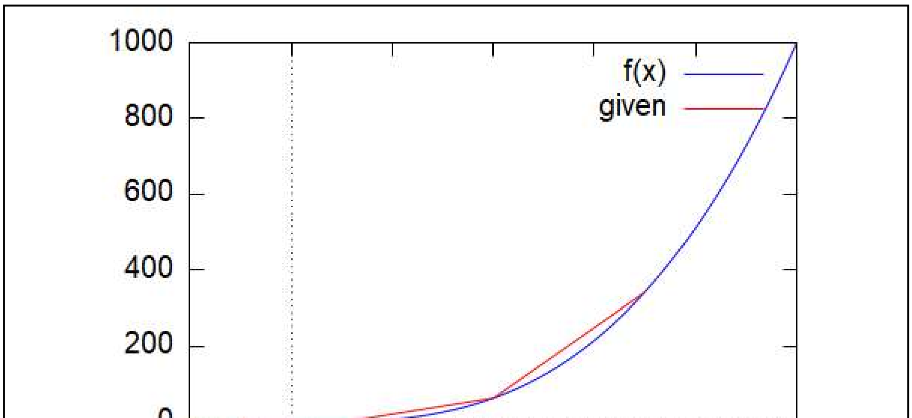
iteration 3 => $\frac{(x-7)(x-4)(x-1)}{40}-\frac{7(x-7)(x-1)(x+1)}{5}$ => $-\frac{11x^3}{8}+\frac{19x^2}{2}+\frac{19x}{8}-\frac{21}{2}$

iteration 4 => $\frac{19(x-4)(x-1)(x+1)}{8}-\frac{7(x-7)(x-1)(x+1)}{5}+\frac{(x-7)(x-4)(x-1)}{40}$ => x^3-1

(%o5) $f(x)=x^3-1$

f(5) = 124

(%t7)



2 ***The following values of a function f(x) = sin(x) + cos(x) are given***

| x | 10deg | 20deg | 30deg |
|--------------------------------|
| f(x) | 1.1585 | 1.2817 | 1.3660 |

Construct the quadratic Lagrange Interpolating Polynomial that fits the data.

Hence, find f(π / 12).

```
kill(all)$
p = p: [
    [10 · %pi / 180, 1.1585],
    [20 · %pi / 180, 1.2817],
    [30 · %pi / 180, 1.3660]
];
n: length(p)$
Y: 0$
for i: 1 thru n do (
    l_i: 1,
    for j: 1 thru n do (
        if notequal(i, j) then
            l_i: l_i · (x − p[j][1]) / (p[i][1] − p[j][1])
    ),
    Y: Y + l_i · p[i][2],
    print("iteration", i, "=>", Y, "=>", expand(Y))
)$
'f(x) = f: expand(Y);
print("f(π / 12) =", ev(f, x = %pi / 12))$
wxplot2d([f, [discrete, map(first, p), map(second, p)]], [x, 0.15, 0.6
```

(%o1) $p=[[\frac{\%pi}{18},1.1585],[\frac{\%pi}{9},1.2817],[\frac{\%pi}{6},1.366]]$

iteration 1 => $\frac{187.677\left(x-\frac{\%pi}{6}\right)\left(x-\frac{\%pi}{9}\right)}{\%pi^2}$ => $\frac{187.677\,x^2}{\%pi^2}-\frac{52.13250000000001\,x}{\%pi}+3.4755$

iteration 2 => $\frac{187.677\left(x-\frac{\%pi}{6}\right)\left(x-\frac{\%pi}{9}\right)}{\%pi^2}-\frac{415.2708\left(x-\frac{\%pi}{6}\right)\left(x-\frac{\%pi}{18}\right)}{\%pi^2}$ => $-\frac{227.5938\,x^2}{\%pi^2}+\frac{40.14989999999999\,x}{\%pi}-0.3695999999999997$

iteration 3 => $\frac{221.292\left(x-\frac{\%pi}{9}\right)\left(x-\frac{\%pi}{18}\right)}{\%pi^2}-\frac{415.2708\left(x-\frac{\%pi}{6}\right)\left(x-\frac{\%pi}{18}\right)}{\%pi^2}+\frac{187.677\left(x-\frac{\%pi}{6}\right)\left(x-\frac{\%pi}{9}\right)}{\%pi^2}$ => $-\frac{6.301799999999957\,x^2}{\%pi^2}+\frac{3.267899999999983\,x}{\%pi}+0.9964000000000004$

(%o5) $f(x)=-\frac{6.301799999999957\,x^2}{\%pi^2}+\frac{3.267899999999983\,x}{\%pi}+0.9964000000000004$

f(π / 12) = 1.224962499999999

3 Find the value of y at x = 0 given some set of values (-2, 5), (1, 7), (3, 11), (7, 34).

```
kill(all)$
p = p: [
    [-2, 5],
    [1, 7],
    [3, 11],
    [7, 34]
];
n: length(p)$
Y: 0$
for i: 1 thru n do (
    l_i: 1,
    for j: 1 thru n do (
        if notequal(i, j) then
            l_i: l_i · (x − p[j][1]) / (p[i][1] − p[j][1])
    ),
    Y: Y + l_i · p[i][2],
    print("iteration", i, "=>", Y, "=>", expand(Y))
)$
'f(x) = f: expand(Y);
print("f(0) =", ev(f, x = 0))$
wxplot2d([f, [discrete, map(first, p), map(second, p)]], [x, -2, 7], [
```

(%o1) $p=[[-2,5],[1,7],[3,11],[7,34]]$

iteration 1 => $-\frac{(x-7)(x-3)(x-1)}{27}$ => $-\frac{x^3}{27}+\frac{11x^2}{27}-\frac{31x}{27}+\frac{7}{9}$

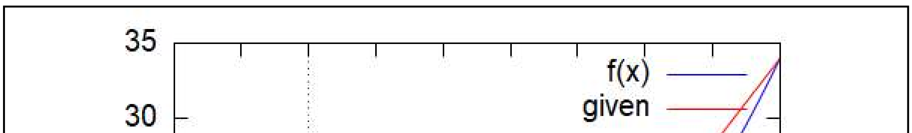
iteration 2 => $\frac{7(x-7)(x-3)(x+2)}{36}-\frac{(x-7)(x-3)(x-1)}{27}$ => $\frac{17x^3}{108}-\frac{31x^2}{27}-\frac{103x}{108}+\frac{161}{18}$

iteration 3 => $-\frac{11(x-7)(x-1)(x+2)}{40}+\frac{7(x-7)(x-3)(x+2)}{36}-\frac{(x-7)(x-3)(x-1)}{27}$ => $-\frac{127x^3}{1080}+\frac{271x^2}{540}+\frac{1643x}{1080}+\frac{917}{180}$

iteration 4 => $\frac{17(x-3)(x-1)(x+2)}{108}-\frac{11(x-7)(x-1)(x+2)}{40}+\frac{7(x-7)(x-3)(x+2)}{36}-\frac{(x-7)(x-3)(x-1)}{27}$ => $\frac{43x^3}{1080}+\frac{101x^2}{540}+\frac{793x}{1080}+\frac{1087}{180}$

(%o5) $f(x)=\frac{43x^3}{1080}+\frac{101x^2}{540}+\frac{793x}{1080}+\frac{1087}{180}$

$f(0) = \frac{1087}{180}$



Using interpol Package

4 Construct the Lagrange Interpolation Polynomial for the data.

x	-1	1	4	7
f(x)	-2	0	63	342

Hence, interpolate at x = 5.

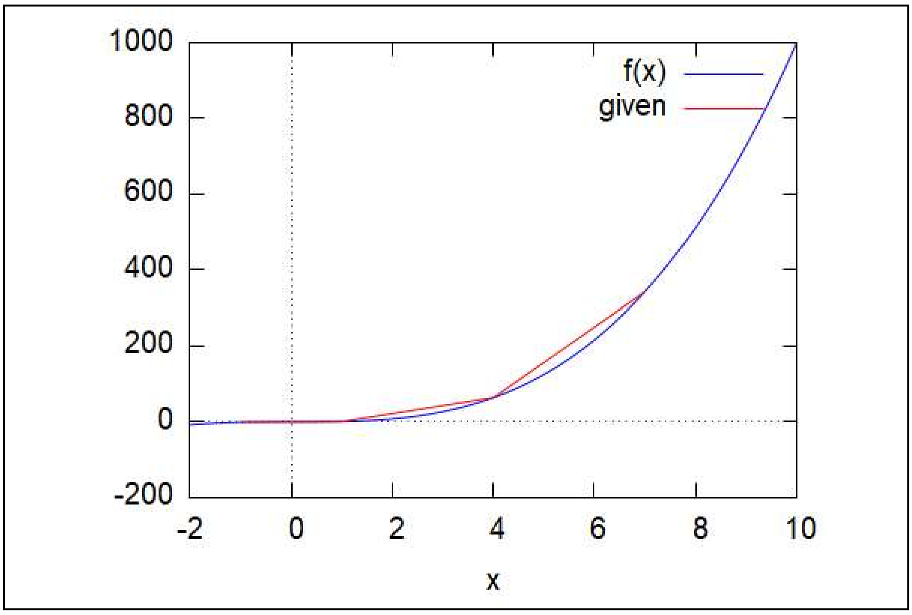
```
kill(all)$
load(interpol)$
p = p: [
    [-1, -2],
    [1, 0],
    [4, 63],
    [7, 342]
];
'f(x) = f: lagrange(p);
'f(x) = f: expand(f);
print("f(5) =", ev(f, x = 5))$
wxplot2d([f, [discrete, map(first, p), map(second, p)]], [x, -2, 10],
```

(%o2) $p=[[-1,-2],[1,0],[4,63],[7,342]]$

(%o3) $f(x)=\frac{19(x-4)(x-1)(x+1)}{8}-\frac{7(x-7)(x-1)(x+1)}{5}+\frac{(x-7)(x-4)(x-1)}{40}$

(%o4) $f(x)=x^3-1$

f(5) = 124

(%t6) 

(%o6)

5 ***The following values of a function f(x) = sin(x) + cos(x) are given***

| x | 10deg | 20deg | 30deg |
|--------------------------------|
| f(x) | 1.1585 | 1.2817 | 1.3660 |

Construct the quadratic Lagrange Interpolating Polynomial that fits the data.

Hence, find f(π / 12).

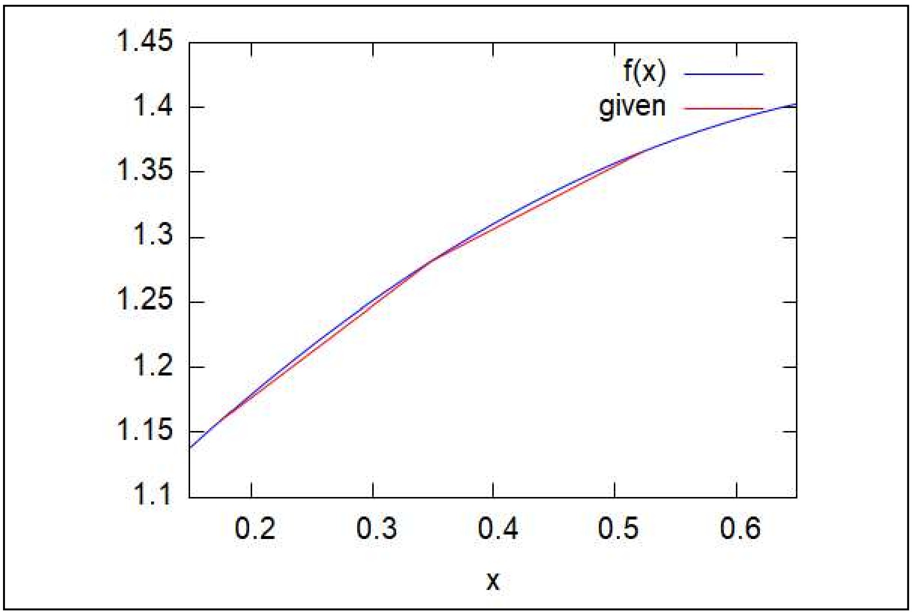
```
kill(all)$
load(interpol)$
p = p: [
    [10 · %pi / 180, 1.1585],
    [20 · %pi / 180, 1.2817],
    [30 · %pi / 180, 1.3660]
];
'f(x) = f: lagrange(p);
'f(x) = f: expand(f);
print("f(π / 12) =", ev(f, x = %pi / 12))$
wxplot2d([f, [discrete, map(first, p), map(second, p)]], [x, 0.15, 0.6
```

(%o2) $p=[[\frac{\%pi}{18},1.1585],[\frac{\%pi}{9},1.2817],[\frac{\%pi}{6},1.366]]$

(%o3) $$f(x)=\frac{221.292\left(x-\frac{\%pi}{9}\right)\left(x-\frac{\%pi}{18}\right)}{\%pi^2}-\frac{415.2708\left(x-\frac{\%pi}{6}\right)\left(x-\frac{\%pi}{18}\right)}{\%pi^2}+\frac{187.677\left(x-\frac{\%pi}{6}\right)\left(x-\frac{\%pi}{9}\right)}{\%pi^2}$$

(%o4) $$f(x)=-\frac{6.301799999999957\,x^2}{\%pi^2}+\frac{3.267899999999983\,x}{\%pi}+0.9964000000000004$$

f(π / 12) = 1.224962499999999

(%t6) 

(%o6)

6 Find the value of y at x = 0 given some set of values (-2, 5), (1, 7), (3, 11), (7, 34)

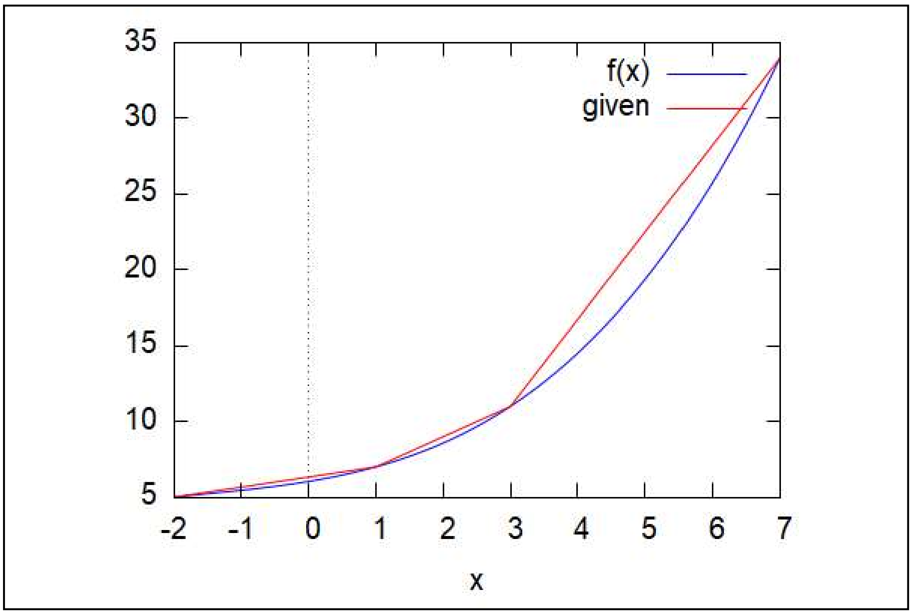
```
kill(all)$
load(interpol)$
p = p: [
     [-2, 5],
     [1, 7],
     [3, 11],
     [7, 34]
];
'f(x) = f: lagrange(p);
'f(x) = f: expand(f);
print("f(0) =", ev(f, x = 0))$
wxplot2d([f, [discrete, map(first, p), map(second, p)]], [x, -2, 7], [
```

(%o2) $p=[[-2,5],[1,7],[3,11],[7,34]]$

(%o3) $$f(x)=\frac{17(x-3)(x-1)(x+2)}{108}-\frac{11(x-7)(x-1)(x+2)}{40}+\frac{7(x-7)(x-3)(x+2)}{36}-\frac{(x-7)(x-3)(x-1)}{27}$$

(%o4) $$f(x)=\frac{43x^3}{1080}+\frac{101x^2}{540}+\frac{793x}{1080}+\frac{1087}{180}$$

$$f(0)\ =\ \frac{1087}{180}$$

(%t6) 

(%o6)

Practical 6(b): Newton's Interpolation

Submitted by - Anshul Verma
(19/78065)
BSc (Hons) Computer Science

1 Construct the Newton's interpolation polynomial for the following given data:

x	*0*	*1*	*2*	*3*
f(x)	*1*	*2*	*5*	*7*

Hence, interpolate at x = 3/2.

```
→   kill(all)$
    x: [0, 1, 2, 3];
    y: zeromatrix(4, 4)$
    y[1][1] : 1$
    y[2][1] : 2$
    y[3][1] : 5$
    y[4][1] : 7$
    'y=y;
    n: length(x)$
    for i: 2 thru n do (
        for j: 1 thru n - i + 1 do (
            y[j][i]: (y[j + 1][i - 1] - y[j][i-1]) / (x[j + i - 1] - x[j])
        )
    )$
    x_t: 1$
    f: y[1][1]$
    for j: 1 thru n - 1 do (
        x_t: x_t · ('x - x[j]), /*Calculating the (x-x0)),(x-x0)*(x-x1),..
                                       ...,till x_n-1*/
    f: f + y[1][j + 1] · x_t,  /*In each iteration,we are adding a
                                 term cumulatively to this*/

    print("iteration", j, "=>",f,"=>", expand(f))
    )$
    'b = y;                    /*Gives the divided difference table*/
    'f('x) = expand(f);
    print("f(3 / 2) =", ev(f, x = 3 / 2))$
    wxplot2d([f, [discrete, x, args(map(first, y))]],
        ['x,-1, 5], [legend, "f(x)", "given"]);
```

(%o1) $[0,1,2,3]$

(%o7) $y=\begin{pmatrix} 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 \\ 7 & 0 & 0 & 0 \end{pmatrix}$

iteration 1 => $x+1$ => $x+1$

iteration 2 => $(x-1)\,x+x+1$ => x^2+1

iteration 3 => $-\frac{(x-2)\,(x-1)\,x}{2}+(x-1)\,x+x+1$ => $-\frac{x^3}{2}+\frac{5\,x^2}{2}-x+1$

(%o13) $b=\begin{pmatrix} 1 & 1 & 1 & -\frac{1}{2} \\ 2 & 3 & -\frac{1}{2} & 0 \\ 5 & 2 & 0 & 0 \\ 7 & 0 & 0 & 0 \end{pmatrix}$

(%o14) $f(x)=-\frac{x^3}{2}+\frac{5\,x^2}{2}-x+1$

f(3 / 2) = 55

2 Construct the Newton's interpolation polynomial for the function f(x)=e^x+1 for the following given data:

x	1.0	1.5	2.0	2.5
f(x)	3.7183	5.4817	5.3891	13.1825

Hence, interpolate at x = 2.5.

```
kill(all)$
```

```
→   x: [1.0,1.5,2.0,2.5 ];
    y: zeromatrix(4, 4)$
    y[1][1] : 3.7183$
    y[2][1] : 5.4817$
    y[3][1] : 5.3891$
    y[4][1] : 13.1825$
    'y=y;
    n: length(x)$
    for i: 2 thru n do (
        for j: 1 thru n - i + 1 do (
            y[j][i]: (y[j + 1][i - 1] - y[j][i-1]) / (x[j + i - 1] - x[j])
        )
    )$
    x_t: 1$
    f: y[1][1]$
    for j: 1 thru n-1  do (
        x_t: x_t · ('x - x[j]),  /*Calculating the (x-x0)),(x-x0)*(x-x1),.
                                               ...,till x_n-1*/

    f: f + y[1][j + 1] · x_t,     /*In each iteration,we are adding a
                                    term cumulatively to this*/

    print("iteration", j, "=>",f,"=>", expand(f))
    )$
    'b = y;                        /*Gives the divided difference table*/
    'f('x) = expand(f);
    print("f((2.5)) =", ev(f, x = 2.5))$
```

(%o1) $[1.0,1.5,2.0,2.5]$

(%o7) $y=\begin{pmatrix} 3.7183 & 0 & 0 & 0 \\ 5.4817 & 0 & 0 & 0 \\ 5.3891 & 0 & 0 & 0 \\ 13.1825 & 0 & 0 & 0 \end{pmatrix}$

iteration 1 => $3.5268\,(x-1.0)+3.7183$ => $3.5268\,x+0.1915000000000004$

iteration 2 => $-3.712\,(x-1.5)\,(x-1.0)+3.5268\,(x-1.0)+3.7183$ => $-3.712\,x^2+12.8068\,x-5.376499999999999$

iteration 3 => $12.98933333333333\,(x-2.0)\,(x-1.5)\,(x-1.0)-3.712\,(x-1.5)\,(x-1.0)+3.5268\,(x-1.0)+3.7183$ => $12.98933333333333\,x^3-62.164\,x^2+97.23746666666666\,x-44.3445$

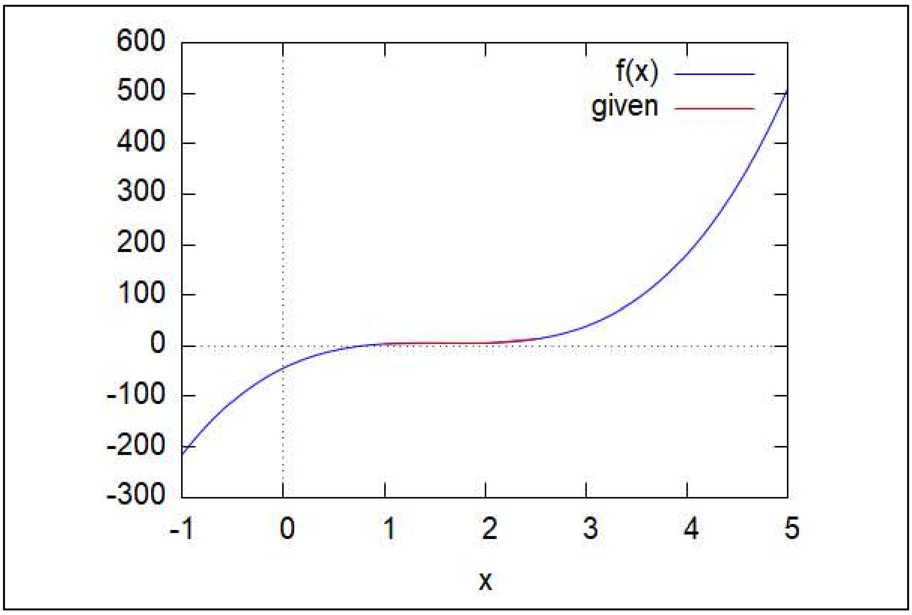
(%o13) $b=\begin{pmatrix} 3.7183 & 3.5268 & -3.712 & 12.98933333333333 \\ 5.4817 & -0.1852 & 15.772 & 0 \\ 5.3891 & 15.5868 & 0 & 0 \\ 13.1825 & 0 & 0 & 0 \end{pmatrix}$

(%o14) $f(x)=12.98933333333333\,x^3-62.164\,x^2+97.23746666666666\,x-44.3445$

f((2.5)) = 13.1825

```
wxplot2d([f, [discrete, x, args(map(first, y))]],
    ['x,-1, 5], [legend, "f(x)", "given"]);
```

(%t16)



(%o16)

3 *Determine the newton form of the interpolating polynomial for the following data sets. Then use that polynomial to find f(1.5). (x,y)=((-1,5),(0,1),(1,1),(2,11)).*

```
kill(all)$
x = x: [-1, 0, 1, 2];
y: zeromatrix(4, 4)$
y[1][1] : 5$
y[2][1] : 1$
y[3][1] : 1$
y[4][1] : 11$
'y = y;
n: length(x)$
for i: 2 thru n do (
    for j: 1 thru n - i + 1 do (
        y[j][i]: (y[j + 1][i - 1] - y[j][i - 1]) / (x[j + i - 1] - x[j
    )
)$
x_t: 1$
f: y[1][1]$
for j: 1 thru n - 1 do (
    x_t: x_t · ('x - x[j]),
    f: f + y[1][j + 1] · x_t,
    print("iteration", j, "=>", expand(f))
)$
'b = y;
'f('x) = expand(f);
print("f(3 / 2) =", ev(f, x = 3 / 2))$
wxplot2d([f, [discrete, x, args(map(first, y))]],
    ['x,-1, 5], [legend, "f(x)", "given"]);
```

(%o1) $x=[-1,0,1,2]$

(%o7) $y=\begin{pmatrix}5&0&0&0\\1&0&0&0\\1&0&0&0\\11&0&0&0\end{pmatrix}$

iteration 1 => $1-4x$

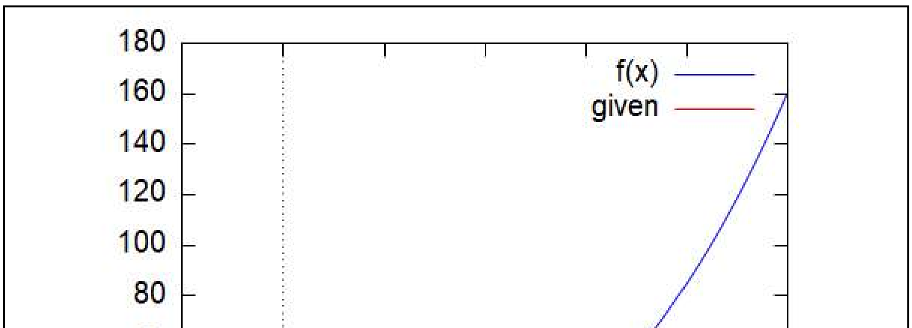
iteration 2 => $2x^2-2x+1$

iteration 3 => x^3+2x^2-3x+1

(%o13) $b=\begin{pmatrix}5&-4&2&1\\1&0&5&0\\1&10&0&0\\11&0&0&0\end{pmatrix}$

(%o14) $f(x)=x^3+2x^2-3x+1$

$f(3\ /\ 2)\ =\ \frac{35}{8}$

(%t16) 

4 Find the value of y at x = 0 given some set of values (-2, 5), (1, 7), (3, 11), (7, 34).

```
kill(all)$
x = x: [-2, 1, 3, 7];
y: zeromatrix(4, 4)$
y[1][1] : 5$
y[2][1] : 7$
y[3][1] : 11$
y[4][1] : 34$
'y = y;
n: length(x)$
for i: 2 thru n do (
    for j: 1 thru n - i + 1 do (
        y[j][i]: (y[j + 1][i - 1] - y[j][i - 1]) / (x[j + i - 1] - x[j
    )
)$
x_t: 1$
f: y[1][1]$
for j: 1 thru n - 1 do (
    x_t: x_t · ('x - x[j]),
    f: f + y[1][j + 1] · x_t,
    print("iteration", j, "=>", expand(f))
)$
'b = y;
'f('x) = expand(f);
print("f(0) =", ev(f, x = 0))$
wxplot2d([f, [discrete, x, args(map(first, y))]],
    ['x,-2, 7], [legend, "f(x)", "given"]);
```

(%o1) $x=[-2,1,3,7]$

(%o7) $y=\begin{pmatrix} 5 & 0 & 0 & 0 \\ 7 & 0 & 0 & 0 \\ 11 & 0 & 0 & 0 \\ 34 & 0 & 0 & 0 \end{pmatrix}$

iteration 1 => $\frac{2x}{3}+\frac{19}{3}$

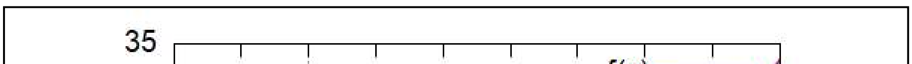
iteration 2 => $\frac{4x^2}{15}+\frac{14x}{15}+\frac{29}{5}$

iteration 3 => $\frac{43x^3}{1080}+\frac{101x^2}{540}+\frac{793x}{1080}+\frac{1087}{180}$

(%o13) $b=\begin{pmatrix} 5 & \frac{2}{3} & \frac{4}{15} & \frac{43}{1080} \\ 7 & 2 & \frac{5}{8} & 0 \\ 11 & \frac{23}{4} & 0 & 0 \\ 34 & 0 & 0 & 0 \end{pmatrix}$

(%o14) $f(x)=\frac{43x^3}{1080}+\frac{101x^2}{540}+\frac{793x}{1080}+\frac{1087}{180}$

f(0) = $\frac{1087}{180}$



5 ***The following supply schedule gives the quantities supplied (S) in hundreds of a product at prices (P) in rupees:***

P,S
80,25
90,30
100,42
110,50
120,68

Interpolate the quantity of the product supplied at the price rs 85.

```
kill(all)$
x = x: [80, 90, 100, 110, 120];
y: zeromatrix(5, 5)$
y[1][1] : 25$
y[2][1] : 30$
y[3][1] : 42$
y[4][1] : 50$
y[5][1] : 68$
'y = y;
n: length(x)$
for i: 2 thru n do (
    for j: 1 thru n - i + 1 do (
        y[j][i]: (y[j + 1][i - 1] - y[j][i - 1]) / (x[j + i - 1] - x[j
    )
)$
x_t: 1$
f: y[1][1]$
for j: 1 thru n - 1 do (
    x_t: x_t · ('x - x[j]),
    f: f + y[1][j + 1] · x_t,
    print("iteration", j, "=>", expand(f))
)$
'b = y;
'f('x) = expand(f);
print("f(85) =", ev(f, x = 85))$
wxplot2d([f, [discrete, x, args(map(first, y))]],
    ['x,80, 120], [legend, "f(x)", "given"]);
```

(%o1) $x=[80,90,100,110,120]$

(%o8) $y=\begin{pmatrix}25 & 0 & 0 & 0 & 0\\ 30 & 0 & 0 & 0 & 0\\ 42 & 0 & 0 & 0 & 0\\ 50 & 0 & 0 & 0 & 0\\ 68 & 0 & 0 & 0 & 0\end{pmatrix}$

iteration 1 => $\frac{x}{2}-15$

iteration 2 => $\frac{7\,x^2}{200}-\frac{109\,x}{20}+237$

iteration 3 => $-\frac{11\,x^3}{6000}+\frac{53\,x^2}{100}-\frac{2989\,x}{60}+1557$

iteration 4 => $\frac{x^4}{9600}-\frac{497\,x^3}{12000}+\frac{14747\,x^2}{2400}-\frac{48253\,x}{120}+9807$

(%o14) $b=\begin{pmatrix}25 & \frac{1}{2} & \frac{7}{200} & -\frac{11}{6000} & \frac{1}{9600}\\ 30 & \frac{6}{5} & -\frac{1}{50} & \frac{7}{3000} & 0\\ 42 & \frac{4}{5} & \frac{1}{20} & 0 & 0\\ 50 & \frac{9}{5} & 0 & 0 & 0\end{pmatrix}$

Practical 7(a): Trapezoidal Rule

Submitted by - Anshul Verma
(19/78065)
BSc (Hons) Computer Science

1 Q.Approximate the integral of f(x) =1/(1+x^2) on the interval [0,1] using the trapezoidal rule.

- Method 1

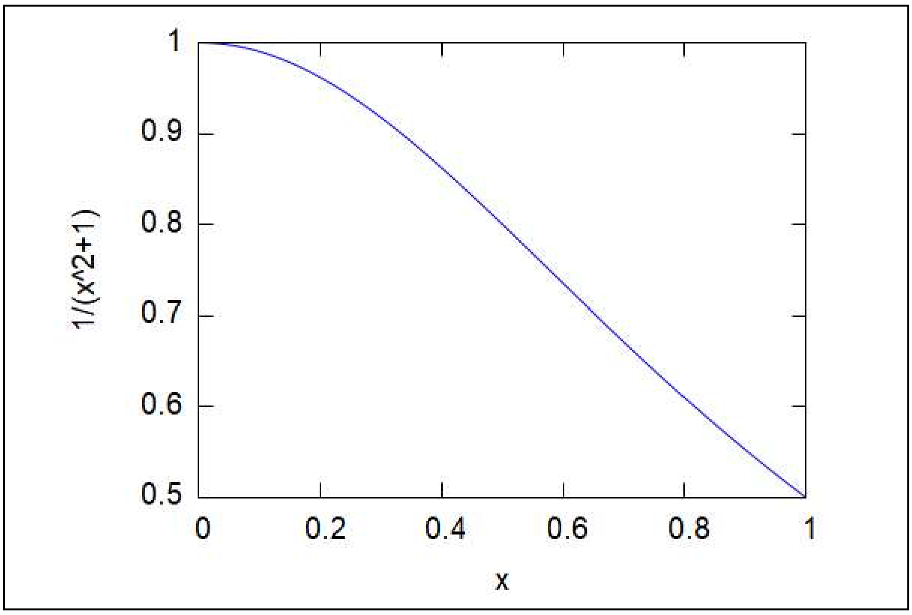
```
kill(all);
f(x):=1/(1+x^2);
a=a:0;
b=b:1;
wxplot2d(f(x),[x,a,b]);
n=n:6;
h=h:(b-a)/n;
for i:0 thru n do
(
    x[i]:a+((i)·h),
    y[i]:float(f(x[i])),
    print('x[i]=x[i],"        ",'y[i]=y[i])
)$
sum=sum:0$
for i:1 thru n-1 do
(
    sum:float(sum + (2·y[i]))
)$
print("Integral of ",f(x)," from 0 to 1 =",( float((h/2·( y[0] + sum +
```

(%o0) done

(%o1) $f(x) := \frac{1}{1+x^2}$

(%o2) $a=0$

(%o3) $b=1$

(%t4) 

(%o4)

(%o5) $n=6$

(%o6) $h=\frac{1}{6}$

$x_0=0$ $y_0=1.0$

$x_1=\frac{1}{6}$ $y_1=0.972972972972973$

$x_2=\frac{1}{3}$ $y_2=0.9$

$x_3=\frac{1}{2}$ $y_3=0.8$

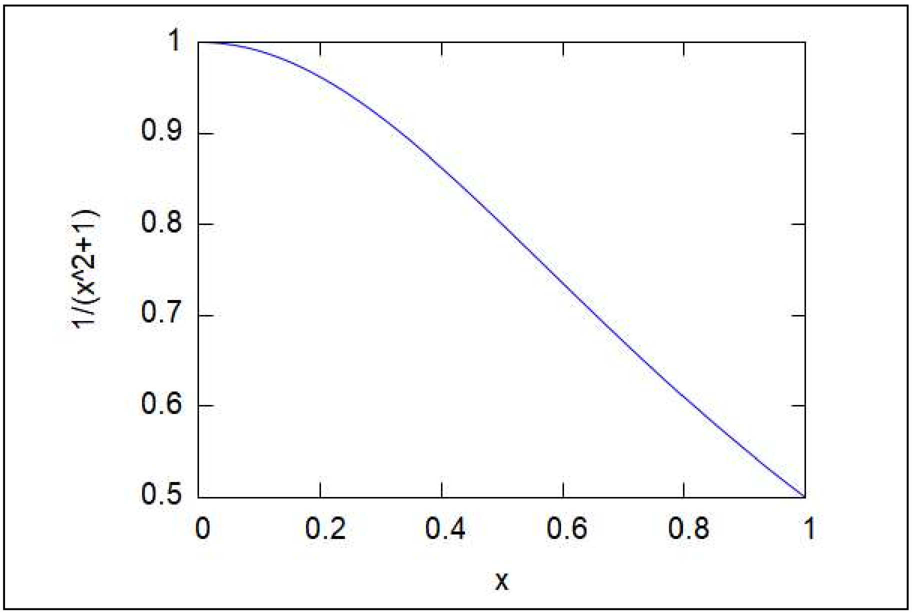
$x_4=\frac{2}{3}$ $y_4=0.6923076923076923$

- Method 2

```
kill(all)$
f(x):=1/(1+x^2)$
b=b:1$
a=a:0$
print('integrate(f(x),x,a,b),"dx")$
wxplot2d(f(x),[x,a,b]);
n:6$
h:(b-a)/n$
int:0$
print('a=a,'b=b,'n=n,'h=h)$
for i:0 thru n-1 do
(
    x[i]:a+(i·h),
    x[i+1]:a+((i+1)·h),
    y[i]:f(x[i]),
    y[i+1]:f(x[i+1]),
    inti:float(h/2·(y[i]+y[i+1])),
    int:float(int+inti),
    print(""),
    print("Value of ",'integrate(f(x),x,x[i],x[i+1]),"dx = ",inti)
)$
print("Thus,",'integrate(f(x),x,a,b),"dx =",int)$
```

$$\int_{0}^{1} \frac{1}{x^2+1}\, dx$$

(%t5)



(%o5)

$a=0 \quad b=1 \quad n=6 \quad h=\frac{1}{6}$

Value of $\int_{0}^{\frac{1}{6}} \frac{1}{x^2+1}\, dx =$ 0.1644144144144144

Value of $\int_{1}^{\frac{1}{3}} \frac{1}{x^2+1}\, dx =$ 0.1560810810810811

2 *Q. Find the approximation value of I= integrate(1/(1+x), x, 0, 1); using the trapezoidal rule with 8 equal subintervals.Using the exact solution , find the absolute error.*

Sol :- Given that,

```
→ print("I= ",'integrate(1/(1+x), x, 0, 1),"dx ; where y = f(x) = ",'1/(
```

$$I= \int_{0}^{1} \frac{1}{x+1} dx \; ; where\ y = f(x) = \frac{1}{x+1}$$

$$, a = 0 , b = 1 , n = 8$$

```
kill(all)$
f(x):=1/(1+x);
a=a:0; /*lower limit*/
b=b:1; /*upper limit*/
n=n:8;
h=h:(b-a)/n;
print("Now")$
print("                 ","x","                         ","y=f(x)")$
print("")$
for i:0 thru n do
(
    x[i]:a+((i)·h),
    y[i]:float(f(x[i])),
    print("             ",'x[i] =x[i],"                   ",'y[i] =y[
)$
sum:0$
for i:1 thru n-1 do
(
    sum:float(sum + (2·y[i]))
)$
I[1]:float((h/2·( y[0] + sum + y[n]) ))$
print("Thus approximation value of integration")$
'I[1]= 'integrate(1/(1+x), x, 0, 1);
print('I[1]=I[1] ) $
print("")$
print("Exact value of the integration is  ")$
I[0]= 'integrate(1/(1+x), x, 0, 1);
I[0]=I[0]: integrate(1/(1+x), x, 0, 1);
'I[0]=I[0]:float(integrate(1/(1+x), x, 0, 1));
print("")$
print("Now Absolute error is ") $
print('I[1]-'I[0],"=",I[1],"-",I[0],"=",I[1]-I[0])$
print("")$
print(" Plot of f(x)")$
wxplot2d(f(x),[x,a,b]);
```

(%o1) $f(x):=\frac{1}{1+x}$

(%o2) $a=0$

(%o3) $b=1$

(%o4) $n=8$

(%o5) $h=\frac{1}{8}$

Now

x y=f(x)

$x_0=0$ $y_0=1.0$

$x_1=\frac{1}{8}$ $y_1=$
0.8888888888888888

$x_2=\frac{1}{4}$ $y_2=0.8$

$x_3=\frac{3}{8}$ $y_3=$
0.7272727272727273

3 Using the trapezium rule, evaluate I = integrate(1/(1+x^2), x, -1, 1) taking 8 intervals.

```
print("I= ",'integrate(1/(1+x^2), x, -1, 1),"dx ; where y = f(x) = ",'
```

$$I= \int_{-1}^{1}\frac{1}{x^2+1}\,dx \; ; \text{ where } y = f(x) = \frac{1}{x^2+1}$$

$, a = -1 , b = 1 , n = 8$

```
kill(all)$
f(x):=1/(1+x^2);
a=a:-1; /*lower limit*/
b=b:1; /*upper limit*/
n=n:8;
h=h:(b-a)/n;
print("Now")$
print("                    ","x","                            ","y=f(x)")$
print("")$
for i:0 thru n do
(
    x[i]:a+((i)·h),
    y[i]:float(f(x[i])),
    print("                ",'x[i] =x[i],"                    ",'y[i] =y[
)$
sum:0$
for i:1 thru n-1 do
(
    sum:float(sum + (2·y[i]))
)$
I[1]:float((h/2·( y[0] + sum + y[n]) ))$
print("Thus approximation value of integration")$
'I[1]= 'integrate(1/(1+x^2), x, -1, 1);
print('I[1]=I[1] ) $
print("")$
print("Exact value of the integration is  ")$
I[0]= 'integrate(1/(1+x^2), x, -1, 1);
I[0]=I[0]: integrate(1/(1+x^2), x, -1, 1);
'I[0]=I[0]:float(integrate(1/(1+x^2), x, -1, 1));
print("")$
print("Now Absolute error is ") $
print('I[1]-'I[0],"=",I[1],"-",I[0],"=",I[1]-I[0])$
print("")$
print(" Plot of f(x)")$
wxplot2d(f(x),[x,a,b]);
```

(%o1) $f(x):=\frac{1}{1+x^2}$

(%o2) $a=-1$

(%o3) $b=1$

(%o4) $n=8$

(%o5) $h=\frac{1}{4}$

Now

x $y=f(x)$

$x_0=-1$ $y_0=0.5$

$x_1=-\frac{3}{4}$ $y_1=0.64$

$x_2=-\frac{1}{2}$ $y_2=0.8$

$x_3=-\frac{1}{4}$ $y_3=$

0.9411764705882353

4 Using the trapezium rule, evaluate I= integrate(sinx, x, 1, 6) with h = 0.5.

```
print("I= ",'integrate(sin(x), x, 1, 6),"dx ; where y = f(x) = ",'sin(
```

$$I= \int_{1}^{6} \sin(x) \; dx \; ; where\ y = f(x) = \sin(x)$$

$, a = 1 , b = 6 , h = 0.5$

```
kill(all)$
f(x):=sin(x);
a=a:1; /*lower limit*/
b=b:6; /*upper limit*/
h=h:0.5;
n=n:(b-a)/h;
print("Now")$
print("                    ","x","                              ","y=f(x)")$
print("")$
for i:0 thru n do
(
    x[i]:a+((i)·h),
    y[i]:float(f(x[i])),
    print("             ",'x[i] =x[i],"                    ",'y[i] =y[
)$
sum:0$
for i:1 thru n-1 do
(
    sum:float(sum + (2·y[i]))
)$
I[1]:(h/2·( y[0] + sum + y[n]))$
print("Thus approximation value of integration")$
'I[1]= 'integrate(sin(x), x, 1, 6);
print('I[1]=I[1] ) $
print("")$
print("Exact value of the integration is  ")$
I[0]= 'integrate(sin(x), x, 1, 6);
I[0]=I[0]: integrate(sin(x), x, 1, 6);
'I[0]=I[0]:float(integrate(sin(x), x, 1, 6));
print("")$
print("Now Absolute error is ") $
print('I[1]-'I[0],"=",I[1],"-",I[0],"=",I[1]-I[0])$
print("")$
print(" Plot of f(x)")$
wxplot2d(f(x),[x,a,b]);
```

(%o1) $f(x):=\sin(x)$

(%o2) $a=1$

(%o3) $b=6$

(%o4) $h=0.5$

(%o5) $n=10.0$

Now

x $y=f(x)$

$x_0=1$ $y_0=$
0.8414709848078965

$x_1=1.5$ $y_1=$
0.9974949866040544

$x_2=2.0$ $y_2=$
0.9092974268256817

$x_3=2.5$ $y_3=$
0.5984721441039564

$x_4=3.0$ $y_4=$
0.1411200080598672

5 Approximate the area under the curve y = 2^x between x = −1 and x = 3 using the Trapezoidal Rule with n = 4 subintervals.

```
print("I= ",'integrate(2^x, x, -1, 3),"dx ; where y = f(x) = ",'2^x,",
```

$$I= \int_{-1}^{3} 2^{x}\, dx \; ; \text{ where } y = f(x) = 2^{x}$$

$, a = -1 , b = 3 , n = 4$

```
kill(all)$
f(x):=2^x;
a=a:-1; /*lower limit*/
b=b:3; /*upper limit*/
n=n:4;
h=h:(b-a)/n;
print("Now")$
print("                    ","x","                            ","y=f(x)")$
print("")$
for i:0 thru n do
(
    x[i]:a+((i)·h),
    y[i]:float(f(x[i])),
    print("                    ",'x[i] =x[i],"                    ",'y[i] =y[
)$
sum:0$
for i:1 thru n-1 do
(
    sum:float(sum + (2·y[i]))
)$
I[1]:(h/2·( y[0] + sum + y[n]) )$
print("Thus approximation value of integration")$
'I[1]= 'integrate(2^x, x, -1, 3);
print('I[1]=I[1] ) $
print("")$
print("Exact value of the integration is  ")$
I[0]= 'integrate(2^x, x, -1, 3);
I[0]=I[0]: integrate(2^x, x, -1, 3);
'I[0]=I[0]:float(integrate(2^x, x, -1, 3));
print("")$
print("Now Absolute error is ") $
print('I[1]-'I[0],"=",I[1],"-",I[0],"=",I[1]-I[0])$
print("")$
print(" Plot of f(x)")$
wxplot2d(f(x),[x,a,b]);
```

(%o1) $f(x):=2^x$

(%o2) $a=-1$

(%o3) $b=3$

(%o4) $n=4$

(%o5) $h=1$

Now

x	$y=f(x)$
$x_0=-1$	$y_0=0.5$
$x_1=0$	$y_1=1.0$
$x_2=1$	$y_2=2.0$
$x_3=2$	$y_3=4.0$
$x_4=3$	$y_4=8.0$

Thus approximation value of integration

(%o14) $I_1=\int_{-1}^{3} 2^x$

Practical 7(b): Simpson's Integration Rule

Submitted by - Anshul Verma
(19/78065)
BSc (Hons) Computer Science

1 Q: Evaluate I = integrate(1/(5+3x), x, 1, 2) using the Simpson’s 1/3 rule with 4 and 8 subintervals. Compare with the exact solution and find absolute errors in the solutions.

Solution: Given =>

```
print("I= ",'integrate(1/(5+3·x), x, 1, 2),"dx ; where y = f(x) = ",'1
```

$$I= \int_{1}^{2} \frac{1}{3\,x+5}\, dx \text{ ; where } y = f(x) = \frac{1}{3\,x+5}$$

$$, a = 1 , b = 2 , n = 8$$

```
kill(all)$
f(x):= 1/(5+3·x);
a = a: 1;          /* Lower Limit */
b = b: 2;          /* Upper Limit */
n = n: 8;          /* Subintervals */
h = h: (b-a)/n;    /* Step Size */
m: zeromatrix(n+1, 2)$
print("                 ", "x", "                                  ", "y=f(x)")$
print("                 ------------------------------------------------------
for i:0 thru n do
(
    m[i+1][1]: a+((i)·h),
    m[i+1][2]: float(f(m[i+1][1])),
    print("             ",'x[i] = m[i+1][1],"                    ",'y[
)$
print("                 ------------------------------------------------------
sum2: 0.0$
sum4: 0.0$
for i:1 thru n-1 do
(
    if (equal(mod(i, 2), 0))
        then(sum2: float(sum2 + m[i+1][2]))
    else (sum4: float(sum4 + m[i+1][2]))
)$
I[0]: float((h/3·( m[1][2] + 2·sum2 + 4·sum4 + m[n+1][2])))$
print("");
print("Thus, the approximate value of the integration is:")$
'I[0] = 'integrate(1/(5+3·x), x, 1, 2);
print('I[0] = I[0])$
print("");
print("Exact value of the integration is:")$
I[1] = 'integrate(1/(5+3·x), x, 1, 2);
I[1] = I[1]: integrate(1/(5+3·x), x, 1, 2);
'I[1] = I[1]: float(integrate(1/(5+3·x), x, 1, 2));
print("");
print("Now, the absolute error in the solution:")$
print('I[0]-'I[1], "=", I[0], "-", I[1], "=", (I[0]-I[1]))$
print("");
print(" Plot")$
wxplot2d([f(x), [discrete, args(map(first, m)), args(map(second, m))]]
          [legend, "f(x)", "Given intervals"], [xlabel, "value of x"],
```

(%o1) $f(x):=\frac{1}{5+3x}$

(%o2) $a=1$

(%o3) $b=2$

(%o4) $n=8$

(%o5) $h=\frac{1}{8}$

x $y=f(x)$

$x_0=1$ $y_0=0.125$

$x_1=\frac{9}{8}$ $y_1=$

2 *Q: FIND SOLUTION OF AN EQUATION integrate(1/(1+x²),x,0,6) USING "simpson's 3/8 rule".*

Solution :

```
kill(all)$
f(x):= 1/(1+x²);
a = a: 0;
b = b: 6;
n = n: 6;
h = h: (b-a)/n;
xy: zeromatrix(n+1,2)$
sum_multipleof3: 0.0$
sum_remaining: 0.0$
print("");
print("  x", "             :             ", "      y=f(x)")$
for i:0 thru n do
(
    xy[i+1][1]: a+((i)·h),
    xy[i+1][2]: float(f(xy[i+1][1])),
    print(x[i] = xy[i+1][1],"           :             ",y[i] =xy[i+1][2])
)$
for i:1 thru n-1 do
(
    if (equal(mod(i, 3), 0))
        then(sum_multipleof3: float(sum_multipleof3 + xy[i+1][2]))
    else (sum_remaining: float(sum_remaining + xy[i+1][2]))
)$
sol1=sol1: float(3·h/8·(( xy[1][2]+xy[n+1][2]) + 2·sum_multipleof3 + 3
print("");
print("SOLUTION using simpson's 3/8 rule:",sol1)$
print("");
print("-------FINDING ERROR-------")$
print("");
sol2=sol2:romberg(f(x),x,a,b)$
print("SOLUTION using integration:",sol2)$
print("ERROR:",sol2,"-",sol1,"= ",sol2-sol1)$
print("");
print("-------GRAPH-------")$
print("");
wxplot2d(f(x),[x,a,b])$
```

(%o1) $f(x):=\frac{1}{1+x^2}$

(%o2) $a=0$

(%o3) $b=6$

(%o4) $n=6$

(%o5) $h=1$

(%o9)

x	:	y=f(x)
$x_0=0$	:	$y_0=1.0$
$x_1=1$	:	$y_1=0.5$
$x_2=2$	:	$y_2=0.2$
$x_3=3$	:	$y_3=0.1$
$x_4=4$	:	$y_4=0.05882352941176471$
$x_5=5$	:	$y_5=0.03846153846153846$
$x_6=6$	:	$y_6=0.02702702702702703$

3 Q: Find the solution of an equation ∫(e^sinx) dx on interval [0,Π/2] using Simpson's 3/8 rule.

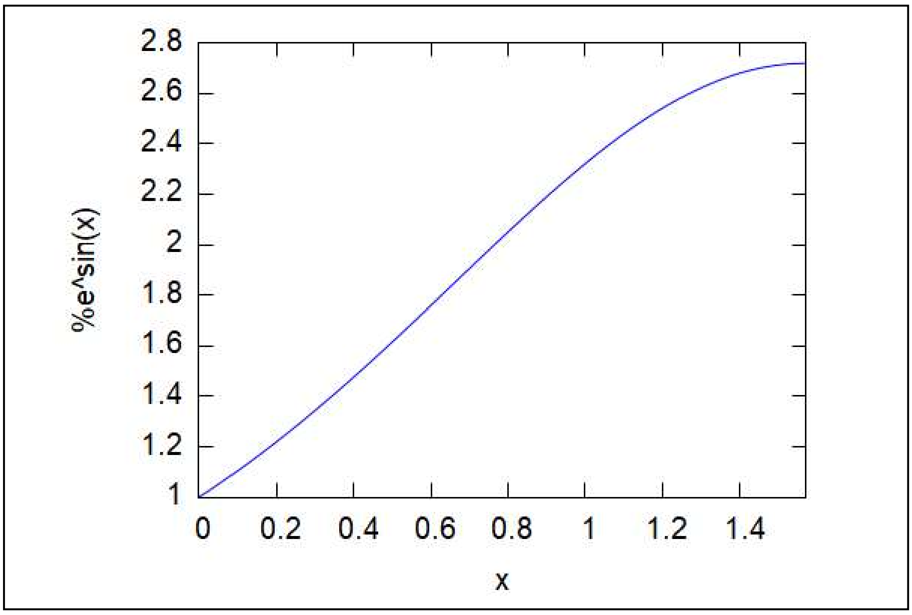
Solution :

```
kill(all)$
f(x):=%e^sin(x);
a=a:0;
b=b:%pi/2;
wxplot2d(f(x),[x,a,b]);
n=n:3;
h=h:(b-a)/n;
print("       ","x","      ","y=f(x)")$
print("")$
for i:0 thru n do
(
    x[i]:a+((i)·h),
    y[i]:float(f(x[i])),
    print("       ",'x[i]=x[i],"      ",'y[i]=y[i])
)$
sum1=sum1:0$
sum2=sum2:0$
for i:1 thru n-1 do
(
    if (equal(mod(i, 3), 0))
        then( sum1:float(sum1 + y[i]))
    else(sum2:float(sum2 + y[i]))
)$
formula=formula:(float((3·h)/8·(y[0] + 2·sum1 + 3·sum2 + y[n])))$
print("∫",f(x)," on interval [0,Π/2] = ",formula)$
print("");
print("CHECKING ERROR")$
print("");
sol=sol:romberg(f(x), x, 0, %pi/2)$
print('integrate(f(x),x,a,b),"dx = ",sol)$
print("Error = ",float(sol-formula))$
```

(%o1) $f(x):=\%e^{\sin(x)}$

(%o2) $a=0$

(%o3) $b=\frac{\%pi}{2}$

(%t4)


(%o4)

(%o5) $n=3$

%pi

4 Use Simpson's 1/3 Rule to integrate f (x) = 0.2 + 25x − 200x2 + 675x3 − 900x4 + 400x5 from a = 0 to b = 0.8.

```
kill(all)$
f(x):= 0.2 + 25·x - 200·x^2 + 675·x^3 - 900·x^4 + 400·x^5;
a = a: 0;          /* Lower Limit */
b = b: 0.8;          /* Upper Limit */
n = n: 6;          /* Subintervals */
h = h: (b-a)/n;    /* Step Size */
m: zeromatrix(n+1, 2)$
print("                 ", "x", "                                  ", "y=f(x)")$
print("                 -----------------------------------------------------
for i:0 thru n do
(
    m[i+1][1]: a+((i)·h),
    m[i+1][2]: float(f(m[i+1][1])),
    print("             ",'x[i] = m[i+1][1],"                        ",'y[
)$
print("                 -----------------------------------------------------
sum2: 0.0$
sum4: 0.0$
for i:1 thru n-1 do
(
    if (equal(mod(i, 2), 0))
        then(sum2: float(sum2 + m[i+1][2]))
    else (sum4: float(sum4 + m[i+1][2]))
)$
I[0]: float((h/3·( m[1][2] + 2·sum2 + 4·sum4 + m[n+1][2])))$
print("");
print("Thus, the approximate value of the integration is:")$
'I[0] = 'integrate(0.2 + 25·x - 200·x^2 + 675·x^3 - 900·x^4 + 400·x^5,
print('I[0] = I[0])$
print("");
print("Exact value of the integration is:")$
I[1] = 'integrate(0.2 + 25·x - 200·x^2 + 675·x^3 - 900·x^4 + 400·x^5,
I[1] = I[1]: integrate(0.2 + 25·x - 200·x^2 + 675·x^3 - 900·x^4 + 400·
'I[1] = I[1]: float(integrate(0.2 + 25·x - 200·x^2 + 675·x^3 - 900·x^4
print("");
print("Now, the absolute error in the solution:")$
print('I[0]-'I[1], "=", I[0], "-", I[1], "=", (I[0]-I[1]))$
print("");
print(" Plot")$
wxplot2d([f(x), [discrete, args(map(first, m)), args(map(second, m))]]
          [legend, "f(x)", "Given intervals"], [xlabel, "value of x"],
```

(%o1) $f(x):=0.2+25x+(-200)x^2+675x^3+(-900)x^4+400x^5$

(%o2) $a=0$

(%o3) $b=0.8$

(%o4) $n=6$

(%o5) $h=0.1333333333333333$

x $y=f(x)$

$x_0=0$ $y_0=0.2$

$x_1=0.1333333333333333$

$y_1=1.310189300411523$

$x_2=0.2666666666666667$

5 *Use Simpson's 3/8 rule to integrate f (x) = 0.2 + 25x − 200x2 + 675x3 − 900x4 + 400x5 from a = 0 to b = 0.8.*

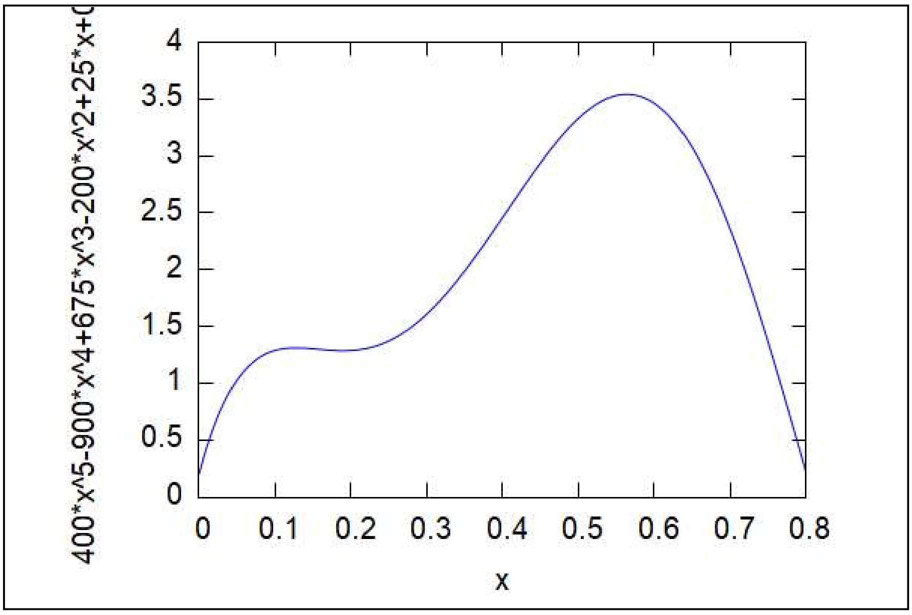
```
kill(all)$
f(x):= 0.2 + 25·x - 200·x^2 + 675·x^3 - 900·x^4 + 400·x^5;
a=a:0;
b=b:0.8;
wxplot2d(f(x),[x,a,b]);
n=n:4;
h=h:(b-a)/n;
print("       ","x","      ","y=f(x)")$
print("")$
for i:0 thru n do
(
    x[i]:a+((i)·h),
    y[i]:float(f(x[i])),
    print("       ",'x[i]=x[i],"     ",'y[i]=y[i])
)$
sum1=sum1:0$
sum2=sum2:0$
for i:1 thru n-1 do
(
    if (equal(mod(i, 3), 0))
        then( sum1:float(sum1 + y[i]))
    else(sum2:float(sum2 + y[i]))
)$
formula=formula:(float((3·h)/8·(y[0] + 2·sum1 + 3·sum2 + y[n])))$
print("∫",f(x)," on interval [0,Π/2] = ",formula)$
print("");
print("CHECKING ERROR")$
print("");
sol=sol:romberg(f(x), x, 0, 0.8)$
print('integrate(f(x),x,a,b),"dx = ",sol)$
print("Error = ",float(sol-formula))$
```

(%o1) $f(x):=0.2+25x+(-200)x^2+675x^3+(-900)x^4+400x^5$

(%o2) $a=0$

(%o3) $b=0.8$

(%t4)



(%o4)

(%o5) $n=4$

(%o6) $h=0.2$

x   y=f(x)

Practical 8: Euler's Method

Submitted by - Anshul Verma
(19/78065)
BSc (Hons) Computer Science

1 *Q: Find an approximation to y(0.4), for the initial value problem*

$$y' = x^2 + y^2, y(0) = 1$$

using the Euler method with h = 0.1 and h = 0.2.

Given: $f(x, y)=x^2+y^2$, x0=0, y0=1 and xn=0.4
To find: y(0.4)=?
Finding n: For h=0.1:-
x1=x0+h=0+0.1=0.1
x2=x0+2h=0+0.2=0.2
x3=x3+3h=0+0.3=0.3
x4=x4+4h=0+0.4=0.4
=>n=4
For h=0.2
x1=x0+h=0+0.2=0.2
x2=x2+2h=0+0.4=0.4
=>n=2

```
kill(all)$
/*x0 and y0: given initial poitns
n=number of approximations to find
h=step size {(xn-x0)/n}   */
euler_method(x0, y0, n, h):=
block([],
    f(x,y):=x^2+y^2,
    array(yn, flonum, 4),
    print("                          ","xi-1","       ","yi-1","

    for i:1 thru n do(
        slope:f(x0,y0), /*calculates f(x0, y0), f(x1, y1), f(x2, y2),
        yn[i]:y0+h·slope,  /*euler formula yn = yn-1 + h*f(xn-1, yn-1)
        print("For approximation i=",i,":-"),
        print("                    ", x0,"        ",y0,"
        y0:yn[i], /*y0=y1, y0=y2, y0=y3, and so on... */
        x0:x0+h), /* x0=x0+h=x1, x0=x1+h=x2, x0=x2+h=x3, and so on...

    /*end of for loop */
    my_points:makelist([k, yn[k]], k, 1, n),
    print("The approximation at n:",n ,"=",yn[n]),
    return (my_points));

pts1:euler_method(0, 1, 4, 0.1);
pts2:euler_method(0, 2, 0.2);
wxplot2d([[discrete, pts1], [discrete, pts2]],[x, 1, 5], [y, 0, 8], [l
```

(%o1) euler_method(x0,y0,n,h):=block

```
                            xi-1          yi-1
f(xi-1, yi-1)          yi
For approximation i= 1 :-
                            0           1            1
         1.1
For approximation i= 2 :-
                            0.1           1.1
1.22          1.222
For approximation i= 3 :-
                            0.2           1.222
 1.533284000000001            1.3753284
For approximation i= 4 :-
                            0.3           1.3753284
          1.981528207846561          1.573481220784656
The approximation at n: 4 = 1.573481220784656
```

(%o2) [[1,1.1],[2,1.222],[3,1.3753284],[4,1.573481220784656]]

```
                            xi-1          yi-1
f(xi-1, yi-1)          yi
For approximation i= 1 :-
                            0           1            1
         1.2
For approximation i= 2 :-
                            0.2           1.2
1.48          1.496
The approximation at n: 2 = 1.496
```

(%o3) [[1,1.2],[2,1.496]]

2 *Q: Consider the IVP dy/dx=(x^2)+y with y(0)=1. Find the approximated value of 0.4 with step size 0.1.*

Given:- f(x, y)=x^2+y
Initial conditions: x0=0, y0=1 and h=0.1
To find:- y(0.4)=?
Finding n:-
x1=x0+h=0+0.1=0.1
x2=x0+2h=0+0.2=0.2
x3=x0+3h=0+0.3=0.3
x4=x0+4h=0+0.4=0.4
=>n=4

```
kill(all)$
f(x,y):=x^2+y;
x0:0;
y0:1;
xn:0.4;
n:4;
h:0.1;
array(yn, flonum, 4);
print("                    ","x0","                    ","y0","
for i:1 thru n do(
    slope:f(x0,y0),
    yn[i]:y0+h·slope,
    print("For approximation",i,":-"),
    print("                    ", x0,"                    ",y0,"
    y0:yn[i],
    x0:x0+h);
my_points:makelist([j, yn[j]], j, 1, n);
print("The approximation y(0.4)=", yn[4]);
wxplot2d([discrete, my_points], [x, 0, 7], [y, -5, 15], [legend, "h=0.
```

(%o1) $f(x,y):=x^2+y$

(%o2) 0

(%o3) 1

(%o4) 0.4

(%o5) 4

(%o6) 0.1

(%o7) yn

x0 y0

f(x0, y0) yn

(%o8) yn

For approximation 1 :-

0 1

1 1.1

For approximation 2 :-

0.1 1.1

1.11 1.211

For approximation 3 :-

0.2 1.211

1.251 1.3361

For approximation 4 :-

0.3 1.3361

1.4261 1.47871

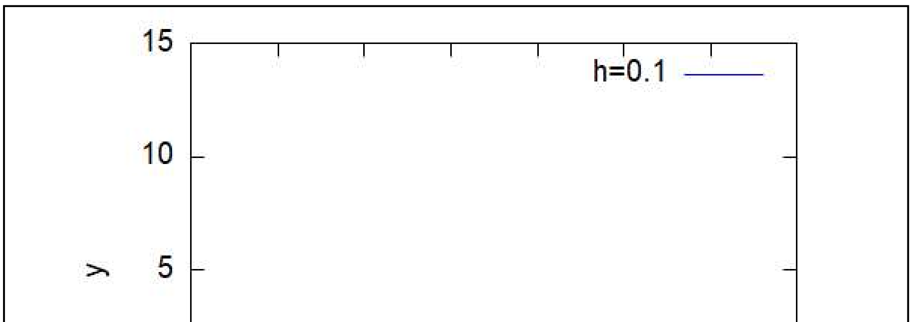
(%o9) done

(%o10) [[1,1.1],[2,1.211],[3,1.3361],[4,1.47871]]

The approximation y(0.4)= 1.47871

(%o11) 1.47871

(%t12)



3 For the IVP
*y' + 2*y = 2 − (%e)^−4*t, y(0) = 1*
Use Euler's Method with a step size of h = 0.1
to find approximate values of the solution at
t = 0.1, 0.2, 0.3, 0.4, and 0.5.

```
kill(all)$
f(t,y):= (2 - (%e)^-4·t) - 2·y;
'x0=x0:0;
'y0=y0:1;
'n=n:6;
'h=h:0.1;
array(yn, flonum, 6)$
print("                        ","x0","                     ","y0","
for i:1 thru n do(
    slope:f(x0,y0),
    yn[i]:y0+h·slope,
    print("For approximation",i,":-"),
    print("                        ", float(x0),"               ",flo
    y0:yn[i],
    x0:x0+h)$
my_points:makelist([j, yn[j]], j, 1, n)$
wxplot2d([discrete, my_points], [x, 0, 7], [legend, "h=0.1"]);
```

(%o1) $f(t,y):=2-\%e^{-4}t-2y$

(%o2) $x0=0$

(%o3) $y0=1$

(%o4) $n=6$

(%o5) $h=0.1$

x0 y0

f(x0, y0) yn

(%o7) yn

For approximation 1 :-

0.0 1.0

0.0 1.0

For approximation 2 :-

0.1 1.0

-0.001831563888873418

0.9998168436111127

For approximation 3 :-

0.2

0.9998168436111127 -

0.003296814999972178 0.9994871621111154

For approximation 4 :-

0.3

0.9994871621111154 -

0.004469015888851124 0.9990402605222303

For approximation 5 :-

0.4

0.9990402605222303 -

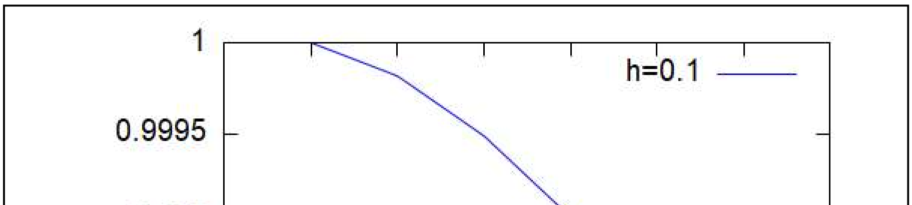
0.005406776599954217 0.9984995828622348

For approximation 6 :-

0.5

0.9984995828622348 -

0.006156985168836761 0.9978838843453511



4 For the IVP

y'− y =
−12*(%e)^(t/2)*sin(5*t)+5*(%e)^(t/2)*cos(5*t),
y(0) = 0.
Use Euler’s Method to find the approximation to the solution at
t = 0.1, t = 0.15, t = 0.20, t = 0.25.
Use h = 0.05 for the approximations.

```
kill(all)$
f(t,y):= -12·(%e)^(t/2)·sin(5·t)+5·(%e)^(t/2)·cos(5·t) + y;
'x0=x0:0;
'y0=y0:0;
'n=n:6;
'h=h:0.05;
array(yn, flonum, 6)$
print("                         ","x0","                       ","y0","
for i:1 thru n do(
    slope:f(x0,y0),
    yn[i]:y0+h·slope,
    print("For approximation",i,":-"),
    print("                         ", float(x0),"                       ",flo
    y0:yn[i],
    x0:x0+h)$
my_points:makelist([j, yn[j]], j, 1, n)$
wxplot2d([discrete, my_points], [x, 0, 7], [legend, "h=0.05"]);
```

(%o1) $f(t,y):=(-12)\,\%e^{t/2}\sin(5\,t)+5\,\%e^{t/2}\cos(5\,t)+y$

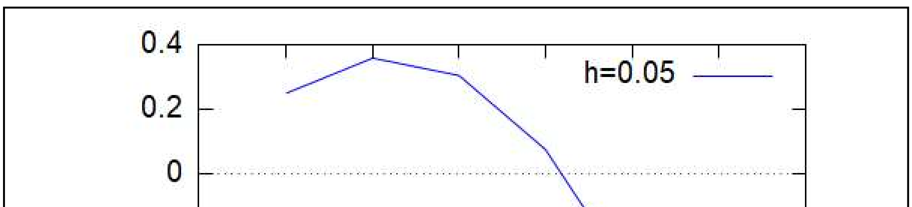
(%o2) $x0=0$

(%o3) $y0=0$

(%o4) $n=6$

(%o5) $h=0.05$

```
                         x0                    y0
            f(x0, y0)                yn
(%o7) yn
For approximation 1 :-
                          0.0                   0.0
            5.0                   0.25
For approximation 2 :-
                          0.05                  0.25
            2.173198538604065
0.3586599269302032
For approximation 3 :-
                          0.1
0.3586599269302032                   -1.076528702264212
            0.3048334918169926
For approximation 4 :-
                          0.15
0.3048334918169926                   -4.568518725187689
            0.07640755555760814
For approximation 5 :-
                          0.2
0.07640755555760814                  -8.097591597138933
            -0.3284720242993385
For approximation 6 :-
                          0.25                  -
0.3284720242993385                   -11.44602222935538
            -0.9007731357671077
```



5 *Use Euler’s method to find the approximation to the solution for y = 4e0.8t − 0.5y at t = 2 with a step size of 1. The initial condition at t = 0 is y = 2.*

```
kill(all)$
f(t,y):= 4·(%e)^(0.8·t) - 0.5·y;
x0:0;
y0:2;
n:4;
h:1;
array(yn, flonum, 4)$
print("                        ","x0","                    ","y0","
for i:1 thru n do(
    slope:f(x0,y0),
    yn[i]:y0+h·slope,
    print("For approximation",i,":-"),
    print("                        ", float(x0),"              ",flo
    y0:yn[i],
    x0:x0+h)$
my_points:makelist([j, yn[j]], j, 1, n)$
print("The approximation y(2)=", yn[2])$
wxplot2d([discrete, my_points], [x, 0, 7], [legend, "h=1"]);
```

(%o1) $f(t,y):=4\,\%e^{0.8\,t}-0.5\,y$

(%o2) 0

(%o3) 2

(%o4) 4

(%o5) 1

```
                        x0                    y0
              f(x0, y0)              yn
For approximation 1 :-
                         0.0                   2.0
              3.0                    5.0
For approximation 2 :-
                         1.0                   5.0
              6.402163713969871
11.40216371396987
For approximation 3 :-
                         2.0
11.40216371396987                14.11104784059552
              25.51321155456539
For approximation 4 :-
                         3.0
25.51321155456539                31.33609974528372
              56.84931129984912
The approximation y(2)= 11.40216371396987
```

(%t11)