Chapter 13 - Numerical solution of ordinary differential equations

Edit examples of this chapter

Example No. 13_01 Pg No. 414

In [1]:
```
from __future__ import division
from math import sqrt
#Taylor method

def f(x,y):
    F = x**2 + y**2
    return F
def d2y(x,y):
    D2Y = 2*x + 2*y*f(x,y)
    return D2Y
def d3y(x,y):
    D3Y = 2 + 2*y*d2y(x,y) + 2*f(x,y)**2
    return D3Y
def y(x):
    Y = 1 + f(0,1)*x + d2y(0,1)*x**2/2 + d3y(0,1)*sqrt(x)
    return Y
print ' y(0.25) = ',y(0.25)
print ' y(0.5) = ',y(0.5)
```

```
 y(0.25) =  5.3125
 y(0.5) =  7.40685424949
```

Example No. 13_02 Pg No. 415

In [1]:
```
from __future__ import division
from scipy.misc import factorial
#Recursive Taylor Method

def f(x,y):
    F = x**2 + y**2
    return F
def d2y(x,y):
    D2Y = 2*x + 2*y*f(x,y)
    return D2Y
def d3y(x,y):
    D3Y = 2 + 2*y*d2y(x,y) + 2*f(x,y)**2
    return D3Y
def d4y(x,y):
    D4Y = 6*f(x,y)*d2y(x,y) + 2*y*d3y(x,y)
    return D4Y
h = 0.2 #
def y(x,y):
    Y = y + f(x,y)*h + d2y(x,y)*h**2/2 + d3y(x,y)*h**3/6 + d4y(x,y)*h**4/factorial(4)
    return Y
x0 = 0#
y0 = 0 #
y_=[]
for i in range(1,3):
    y_.append(y(x0,y0))
    print ' Iteration-%d\n\n dy(0) = %f\n d2y(0) = %f\n d3y(0) = %f\n d4y(0) = %f\n '%(i,f(x0,y0),d2y(x
0,y0),d3y(x0,y0),d4y(x0,y0))
    x0 = x0 + i*h
    y0 = y_[i-1]
    print 'y(0) = %f\n\n'%(y_[i-1])
```

```
 Iteration-1

 dy(0) = 0.000000
 d2y(0) = 0.000000
 d3y(0) = 2.000000
 d4y(0) = 0.000000

y(0) = 0.002667


 Iteration-2

 dy(0) = 0.040007
 d2y(0) = 0.400213
 d3y(0) = 2.005336
 d4y(0) = 0.106763

y(0) = 0.021353
```

Example No. 13_03 Pg No. 417

In [2]:
```
from __future__ import division
from math import exp
#Picard's Method

#y'(x) = x**2 + y**2,y(0) = 0
#y(1) = y0 + integral(x**2 + y0**2 x0 x)
```

```
#y(1) = y0 + integral(x**2 + y0**2,x0,x)
#y(1) = x**3/3
#y(2) = 0 + integral(xY2 + y1**2,x0,x)
#     = integral(x**2 + x**6/9,0,x) = x**3/3 + x**7/63
#therefore y(x) = x**3 /3 + x**7/63
def y(x):
    Y = x**3/3 + x**7/63
    return Y
print 'for dy(x) = x**2 + y**2 the results are :'
print 'y(0.1) = ',y(0.1)
print 'y(0.2) = ',y(0.2)
print 'y(1) = ',y(1)

#y'(x) = x*e**y, y(0) = 0
#y0 = 0 , x0 = 0
#Y(1) = 0 + integral(x*e**0,0,x) = x**2/2
#y(2) = 0 + integral( x*e**( x**2/2 ) ,0,x) = e**(x**2/2)-1
#therefore y(x) = e**(x**2/2) - 1
def y(x):
    Y = exp(x**2/2) - 1
    return Y
print 'for dy(x) = x*e**y the results are '
print 'y(0.1) = ',y(0.1)
print 'y(0.2) = ',y(0.2)
print 'y(1) = ',y(1)
```

```
for dy(x) = x**2 + y**2 the results are :
y(0.1) =  0.000333334920635
y(0.2) =  0.00266686984127
y(1) =  0.349206349206
for dy(x) = x*e**y the results are
y(0.1) =  0.0050125208594
y(0.2) =  0.0202013400268
y(1) =  0.6487212707
```

Example No. 13_04 Pg No. 417

In [3]:

```
from __future__ import division

#Euler's Method

def dy(x):
    DY = 3*x**2 + 1
    return DY
x0 = 1
y = [0,2] #
#h = 0.5
h = 0.5
print 'for h = %f\n'%(h)
for i in range(2,4):
    y.append(y[i-1] + h*dy(x0+(i-2)*h))
    print 'y(%f) = %f\n'%(x0+(i-1)*h,y[i])

#h = 0.25
h = 0.25
print '\nfor h = %f\n'%(h)
y = [0,2] #
for i in range(2,6):
    y.append(y[i-1] + h*dy(x0+(i-2)*h))
    print 'y(%f) = %f\n'%(x0+(i-1)*h,y[i])
```

```
for h = 0.500000

y(1.500000) = 4.000000

y(2.000000) = 7.875000


for h = 0.250000

y(1.250000) = 3.000000

y(1.500000) = 4.421875

y(1.750000) = 6.359375

y(2.000000) = 8.906250
```

Example No. 13_05 Pg No. 422

In [1]:

```
from numpy import nditer
from __future__ import division
#Error estimation in Euler's Method

def dy(x):
    DY = 3*x**2 + 1
    return DY
def d2y(x):
    D2Y = 6*x
    return D2Y
def d3y(x):
    D3Y = 6
    return D3Y
def exacty(x):
    Exacty = x**3 + x
    return Exacty
x0 = 1
y = 2
```

```
h = 0.5
X=[];ESTY=[];TRUEY=[];ET=[];GERR=[];
for i in range(2,4):
    x = x0 + (i-1)*h
    y = y + h*dy(x0+(i-2)*h)
    print '\n Step %d \n x(%d) = %f\n y(%f) = %f\n'%(i-1,i-1,x,x,y)
    Et = d2y(x0+(i-2)*h)*h**2/2 +  d3y(x0+(i-2)*h)*h**3/6
    print '\n Et(%d) = %f\n'%(i-1,Et)
    truey = exacty(x0+(i-1)*h)
    gerr = truey - y
    X.append(x)
    ESTY.append(y)
    TRUEY.append(truey)
    ET.append(Et)
    GERR.append(gerr)

#table = [x y(2:3) truey Et gerr]
print 'x      Est y   true y    Et      Globalerr'
for a,b,c,d,e in nditer([X,ESTY,TRUEY,ET,GERR]):
    print a,'\t',b,'\t',c,'\t',d,'\t',e
```

```
 Step 1
 x(1) = 1.500000
 y(1.500000) = 4.000000


 Et(1) = 0.875000


 Step 2
 x(2) = 2.000000
 y(2.000000) = 7.875000


 Et(2) = 1.250000

x      Est y   true y    Et      Globalerr
1.5     4.0     4.875   0.875   0.875
2.0     7.875   10.0    1.25    2.125
```

Example No. 13_06 Pg No. 427

In [1]:

```
from numpy import nditer,sqrt
from __future__ import division

#Heun's Method

def f(x,y):
    F = 2*y/x
    return F
def exacty(x):
    Exacty = 2*sqrt(x)
    return Exacty
x = [0,1] #
y = [0,2] #
h = 0.25 #
#Euler's Method
print 'EULERS METHOD'
for i in range(2,6):
    x.append(x[i-1] + h )
    y.append(y[i-1] + h*f(x[i-1],y[i-1]))
    print 'y(%f) = %f \n '%(x[i],y[i])

eulery = y
#Heun's Method
print 'HEUNS METHOD'
ye=[0,0]
y = [0,2] #
for i in range(2,6):
    m1 = f(x[i-1],y[i-1]) #
    ye.append(y[i-1] + h*f(x[(i-1)],y[(i-1)]))
    m2 = f(x[(i)],ye[(i)])
    y.append(y[(i-1)] + h*(m1 + m2)/2)
    print '\nIteration %d \n m1 = %f\n ye(%f) = %f \n m2 = %f \n y(%f) = %f \n'%(i-1,m1,x[(i)],ye[(i)],
m2,x[(i)],y[(i)])


truey = exacty(x)
print 'x     Eulers   Heuns   \t\tAnalytical'
for a,b,c,d in nditer([x,eulery,y,truey]):
    print a,'\t',b,'\t%.6f'%c,'\t',d
```

```
EULERS METHOD
y(1.250000) = 3.000000

y(1.500000) = 4.200000

y(1.750000) = 5.600000

y(2.000000) = 7.200000

HEUNS METHOD

Iteration 1
 m1 = 4.000000
 ye(1.250000) = 3.000000
 m2 = 4.800000
 y(1.250000) = 3.100000
```

```
Iteration 2
 m1 = 4.960000
 ye(1.500000) = 4.340000
 m2 = 5.786667
 y(1.500000) = 4.443333


Iteration 3
 m1 = 5.924444
 ye(1.750000) = 5.924444
 m2 = 6.770794
 y(1.750000) = 6.030238


Iteration 4
 m1 = 6.891701
 ye(2.000000) = 7.753163
 m2 = 7.753163
 y(2.000000) = 7.860846

x     Eulers   Heuns           Analytical
0.0     0.0     0.000000        0.0
1.0     2.0     2.000000        2.0
1.25    3.0     3.100000        2.2360679775
1.5     4.2     4.443333        2.44948974278
1.75    5.6     6.030238        2.64575131106
2.0     7.2     7.860846        2.82842712475
```

Example No. 13_07 Pg NO. 433

In [1]:
```
from __future__ import division

#Polygon Method

def f(x,y):
    F = 2*y/x
    return F
x=[1]
y=[2]
h = 0.25 #
for i in range(1,3):
    x.append(x[(i-1)] + h )
    y.append(y[(i-1)] + h*f(  x[(i-1)]+ h/2  ,  y[(i-1)] + h*f( x[(i-1)]  ,  y[(i-1)] )/2  ))
    print 'y(%f) = %f \n '%(x[(i)],y[(i)])
```

```
y(1.250000) = 3.111111

y(1.500000) = 4.468687
```

Example No. 13_08 Pg No. 439

In [1]:
```
from __future__ import division

#Classical Runge Kutta Method

def f(x,y):
    F = x**2 + y**2
    return F
h = 0.2
x=[0]
y=[0]

for i in range(0,2):
    m1 = f(  x[(i)]  ,  y[(i)]  ) #
    m2 = f(  x[i] + h/2  ,  y[(i)] + m1*h/2  ) #
    m3 = f(  x[(i)] + h/2  ,  y[(i)] + m2*h/2  ) #
    m4 = f(  x[(i)] + h  ,  y[(i)] + m3*h  ) #
    x.append(x[(i)] + h )
    y.append(y[(i)] +  (m1 + 2*m2 + 2*m3 + m4)*h/6 )

    print '\nIteration - %d\n m1 = %f\n m2 = %f \n m3 = %f \n m4 = %f \n y(%f) = %f \n'%(i+1,m1,m2,m3,m
4,x[(i+1)],y[(i+1)])
```

```
Iteration - 1
 m1 = 0.000000
 m2 = 0.010000
 m3 = 0.010001
 m4 = 0.040004
 y(0.200000) = 0.002667


Iteration - 2
 m1 = 0.040007
 m2 = 0.090044
 m3 = 0.090136
 m4 = 0.160428
 y(0.400000) = 0.021360
```

Example No. 13_09 Pg No. 444

In [1]:
```
from __future__ import division

#Optimum step size
```

```
x = 0.8 #
h1 = 0.05 #
y1 = 5.8410870 #
h2 = 0.025 #
y2 = 5.8479637 #

#d = 4
h = ((h1**4 - h2**4)*10**(-4)/(2*(y2 - y1)))**(1/4)
print 'for four decimal places'
print 'h = ',h

#d = 6
h = ((h1**4 - h2**4)*10**(-6)/(2*(y2 - y1)))**(1/4)
print 'for six decimal places'
print 'h = ' ,h
print 'Note-We can use h = 0.01 for four decimal places and h = 0.004 for six decimal places'
```

```
for four decimal places
h =  0.0143668085653
for six decimal places
h =  0.00454318377739
Note-We can use h = 0.01 for four decimal places and h = 0.004 for six decimal places
```

Example No. 13_10 Pg NO. 446

In [1]:

```
from __future__ import division

#Milne-Simpson Predictor-Corrector method

def f(x,y):
    F = 2*y/x
    return F
x0 = 1 #
y0 = 2 #
h = 0.25 #
#Assuming y1 ,y2 and y3(required for milne-simpson formula) are estimated using Fourth- order Runge kut
ta method
x1 = x0 + h
y1 = 3.13 #
x2 = x1 + h
y2 = 4.5 #
x3 = x2 + h
y3 = 6.13 #
#Milne Predictor formula
yp4 = y0 + 4*h*(2*f(x1,y1) - f(x2,y2) + 2*f(x3,y3))/3
x4 = x3 + h
fp4 = f(x4,yp4) #
print 'yp4 = ',yp4
print 'fp4 = ',fp4,
#Simpson Corrector formula
yc4 = y2 + h*( f(x2,y2) + 4*f(x3,y3) + fp4)/3
f4 = f(x4,yc4)
print 'yc4 = ',yc4
print 'f4 = ',f4

yc4 = y2 + h*( f(x2,y2) + 4*f(x3,y3) + f4)/3
print 'yc4 = ',yc4
print 'Note- the exact solution is y(2) = 8'
```

```
yp4 =  8.00914285714
fp4 =  8.00914285714 yc4 =  8.00266666667
f4 =  8.00266666667
yc4 =  8.00212698413
Note- the exact solution is y(2) = 8
```

Example No. 13_11 Pg NO. 446

In [1]:

```
from __future__ import division

#Adams-Bashforth-Moulton Method

def f(x,y):
    F = 2*y/x
    return F
x0 = 1 #
y0 = 2 #
h = 0.25 #
x1 = x0 + h
y1 = 3.13 #
x2 = x1 + h
y2 = 4.5 #
x3 = x2 + h
y3 = 6.13 #
#Adams Predictor formula
yp4 = y3 + h*(55*f(x3,y3) - 59*f(x2,y2) + 37*f(x1,y1) - 9*f(x0,y0))/24
x4 = x3 + h
fp4 = f(x4,yp4)
print 'Adams Predictor formula'
print 'yp4 = ',yp4
print 'fp4 = ',fp4
#Adams Corrector formula
yc4 = y3 + h*( f(x1,y1) - 5*f(x2,y2) + 19*f(x3,y3) + 9*fp4 )/24
f4 = f(x4,yc4)
print '\nAdams Corrector formula'
print 'yc4 = ',yc4
print 'f4 = ',f4

yc4 = y3 + h*( f(x1,y1) - 5*f(x2,y2) + 19*f(x3,y3) + 9*f4 )/24
```

```
print '\nrefined-yc4 = ',yc4
```

```
Adams Predictor formula
yp4 =  8.01135714286
fp4 =  8.01135714286

Adams Corrector formula
yc4 =  8.00727901786
f4 =  8.00727901786

refined-yc4 =  8.00689669364
```

Example No. 13_12 Pg No. 453

In [1]:
```
from __future__ import division

#Milne-Simpson Method using modifier

def f(y):
    F = -y**2
    return F
x = [ 1 , 1.2 , 1.4 , 1.6 ]
y = [ 1 , 0.8333333 , 0.7142857 , 0.625 ]
h = 0.2 #

for i in range(0,2):
    yp = y[(i)] + 4*h*( 2*f( y[(i+1)] ) - f( y[(i+2)] ) + 2*f( y[(i+3)] ) )/3
    fp = f(yp) #
    yc = y[( i+2)] + h*(f( y[(i+2)] ) + 4*f( y[(i+3)] ) + fp )/3
    Etc = -(yc - yp)/29
    y.append(yc + Etc)
    print '\n y%dp = %f\n f%dp = %f \n y%dc = %f \n Modifier Etc = %f \n Modified y%dc = %f \n'%(i+4,y
p,i+4,fp,i+4,yc,Etc,i+4,y[(i+4)])
exactanswer = 0.5 #
err = exactanswer - y[5] #
print 'error = ',err
```

```
 y4p = 0.557351
 f4p = -0.310640
 y4c = 0.555396
 Modifier Etc = 0.000067
 Modified y4c = 0.555464


 y5p = 0.500837
 f5p = -0.250837
 y5c = 0.499959
 Modifier Etc = 0.000030
 Modified y5c = 0.499989

error =  1.11332736336e-05
```

Example No. 13_13 Pg No. 455

In [1]:
```
from __future__ import division

#System of differetial Equations

def f1(x,y1,y2):
    F1 = x + y1 + y2
    return F1
def f2(x,y1,y2):
    F2 = 1 + y1 + y2
    return F2

x0 = 0 #
y10 = 1 #
y20 = -1 #
h = 0.1 #
m1= [f1( x0 , y10 , y20 )]
m1.append(f2( x0 , y10 , y20 ))
m2=[f1( x0+h , y10 + h*m1[0] , y20 + h*m1[1] )]
m2.append(f2( x0+h , y10 + h*m1[0] , y20 + h*m1[1] ))
m= [(m1[0] + m2[0])/2 ]
m.append((m1[1] + m2[1])/2)

y1_0_1 = y10 + h*m[0]
y2_0_1 = y20 + h*m[1]

print 'm1(1) = %f\n m1(2) = %f\n m2(1) = %f\n m2(2) = %f\n m(1) = %f\n m(2) = %f\n y1(0.1) = %f\n y2(0.
1) = %f\n'%(m1[0],m1[1],m2[0],m2[1],m[0],m[1],y1_0_1,y2_0_1)
```

```
m1(1) = 0.000000
 m1(2) = 1.000000
 m2(1) = 0.200000
 m2(2) = 1.100000
 m(1) = 0.100000
 m(2) = 1.050000
 y1(0.1) = 1.010000
 y2(0.1) = -0.895000
```

Example No. 13_14 Pg No. 457

In [1]:
```
from __future__ import division
#Higher Order Differential Equations
```

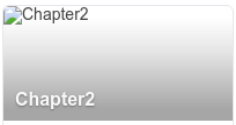
```
x0 = 0
y10 = 0
y20 = 1
h = 0.2
m1 = [y20] #
m1.append(6*x0 + 3*y10 - 2*y20)
m2= [y20 + h*m1[1]]
m2.append(6*(x0+h) + 3*(y10 + h*m1[0]) - 2*(y20 + h*m1[1]) )
m = [(m1[0] + m2[0])/2 ]
m.append((m1[1] + m2[1])/2)

y1_0_2 = y10 + h*m[0]
y2_0_2 = y20 + h*m[1]

print 'm1(1) = %f\n m1(2) = %f\n m2(1) = %f\n m2(2) = %f\n m(1) = %f\n m(2) = %f\n y1(0.1) = %f\n y2(0.
1) = %f\n'%(m1[0],m1[1],m2[0],m2[1],m[0],m[1],y1_0_2,y2_0_2)
```

```
m1(1) = 1.000000
 m1(2) = -2.000000
 m2(1) = 0.600000
 m2(2) = 0.600000
 m(1) = 0.800000
 m(2) = -0.700000
 y1(0.1) = 0.160000
 y2(0.1) = 0.860000
```

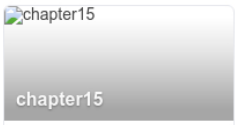

ALSO ON **PYTHON TBC**



Chapter2

5 years ago · 1 comment

Exa2.19 : : Page 95 (2011) #find The power released by the gm …



chapter15

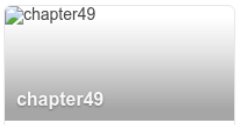
5 years ago · 1 comment

import math import scipy.integrate #Given d = 10.0 fd_ratio = 0.5 f = 3 * …

CHAPT1

5 years ago · 1 comment

#Variable declaration M = 15000.0 + 10.0 #Water evaporated(kg) C = …



chapter49

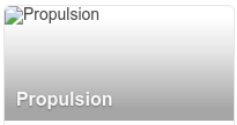
3 years ago · 2 comments

'''A hall 30 m long and 12 m wide is to be illuminated and the illumination required …

ch12

4 years ago · 1 comment

Variables F = 100. ; # Overall feed/basis - [mole] F_H2 = 0.673 ; # Mole …



Propulsion

5 years ago · 1 comment

-*- coding: utf8 -*- from __future__ import division #Example: 9.1 '''Steam …

4 yea

from divis Decl

0 Comments python TBC Disqus' Privacy Policy Login

Recommend Tweet Share Sort by Best

Start the discussion…