NUMERICAL COMPUTING ASSIGNMENT 3

20K-1873

SHAYAN HASSAN

**Q.1 (DIFFRENTIAL EQUATIONS EULER METHOD & 4RK)**

**CODE:**

def f(x, y):

return x + y

# RK-4 method

def rk4(x0, y0, xn, n):

# Calculating step size

h = (xn - x0) / n

print('\n--------SOLUTION--------')

print('-------------------------')

print('x0\ty0\tyn')

print('-------------------------')

for i in range(n):

k1 = h \* (f(x0, y0))

k2 = h \* (f((x0 + h / 2), (y0 + k1 / 2)))

k3 = h \* (f((x0 + h / 2), (y0 + k2 / 2)))

k4 = h \* (f((x0 + h), (y0 + k3)))

k = (k1 + 2 \* k2 + 2 \* k3 + k4) / 6

yn = y0 + k

print('%.4f\t%.4f\t%.4f' % (x0, y0, yn))

print('-------------------------')

y0 = yn

x0 = x0 + h

print('\nAt x=%.4f, y=%.4f' % (xn, yn))

# Inputs

print('Enter initial conditions:')

x0 = float(input('x0 = '))

y0 = float(input('y0 = '))

print('Enter calculation point: ')

xn = float(input('xn = '))

print('Enter number of steps:')

step = int(input('Number of steps = '))

rk4(x0, y0, xn, step)

**OUTPUT:**

--------SOLUTION--------

-------------------------

x0 y0 yn

-------------------------

0.0000 1.0000 11.0000

-------------------------

2.0000 11.0000 93.0000

-------------------------

At x=4.0000, y=93.0000

**Q.2(LU DECOMPOSITION AND LDLT)**

**CODE:**

import pprint

def mult\_matrix(M, N):

tuple\_N = zip(\*N)

# Nested list comprehension to calculate matrix multiplication

return [[sum(el\_m \* el\_n for el\_m, el\_n in zip(row\_m, col\_n)) for col\_n in tuple\_N] for row\_m in M]

def pivot\_matrix(M):

"""Returns the pivoting matrix for M, used in Doolittle's method."""

m = len(M)

# Create an identity matrix, with floating point values

id\_mat = [[float(i ==j) for i in xrange(m)] for j in xrange(m)]

# Rearrange the identity matrix such that the largest element of

# each column of M is placed on the diagonal of of M

for j in xrange(m):

row = max(xrange(j, m), key=lambda i: abs(M[i][j]))

if j != row:

# Swap the rows

id\_mat[j], id\_mat[row] = id\_mat[row], id\_mat[j]

return id\_mat

def lu\_decomposition(A):

n = len(A)

L = [[0.0] \* n for i in xrange(n)]

U = [[0.0] \* n for i in xrange(n)]

P = pivot\_matrix(A)

PA = mult\_matrix(P, A)

for j in xrange(n):

L[j][j] = 1.0

# LaTeX: u\_{ij} = a\_{ij} - \sum\_{k=1}^{i-1} u\_{kj} l\_{ik}

for i in xrange(j+1):

s1 = sum(U[k][j] \* L[i][k] for k in xrange(i))

U[i][j] = PA[i][j] - s1

# LaTeX: l\_{ij} = \frac{1}{u\_{jj}} (a\_{ij} - \sum\_{k=1}^{j-1} u\_{kj} l\_{ik} )

for i in xrange(j, n):

s2 = sum(U[k][j] \* L[i][k] for k in xrange(j))

L[i][j] = (PA[i][j] - s2) / U[j][j]

return (P, L, U)

A = [[7, 3, -1, 2], [3, 8, 1, -4], [-1, 1, 4, -1], [2, -4, -1, 6]]

P, L, U = lu\_decomposition(A)

print "A:"

pprint.pprint(A)

print "P:"

pprint.pprint(P)

print "L:"

pprint.pprint(L)

print "U:"

pprint.pprint(U)

**OUTPUT:**

A:

[[7, 3, -1, 2], [3, 8, 1, -4], [-1, 1, 4, -1], [2, -4, -1, 6]]

P:

[[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]]

L:

[[1.0, 0.0, 0.0, 0.0],

[0.42857142857142855, 1.0, 0.0, 0.0],

[-0.14285714285714285, 0.2127659574468085, 1.0, 0.0],

[0.2857142857142857, -0.7234042553191489, 0.0898203592814371, 1.0]]

U:

[[7.0, 3.0, -1.0, 2.0],

[0.0, 6.714285714285714, 1.4285714285714286, -4.857142857142857],

[0.0, 0.0, 3.5531914893617023, 0.31914893617021267],

[0.0, 0.0, 0.0, 1.88622754491018]]

**Q.3(GUASS SEIDAL AND JACOBI METHOD)**

**CODE:**

f1 = lambda x,y,z: (17-y+2\*z)/20

f2 = lambda x,y,z: (-18-3\*x+z)/20

f3 = lambda x,y,z: (25-2\*x+3\*y)/20

# Initial setup

x0 = 0

y0 = 0

z0 = 0

count = 1

# Reading tolerable error

e = float(input('Enter tolerable error: '))

# Implementation of Gauss Seidel Iteration

print('\nCount\tx\ty\tz\n')

condition = True

while condition:

x1 = f1(x0,y0,z0)

y1 = f2(x1,y0,z0)

z1 = f3(x1,y1,z0)

print('%d\t%0.4f\t%0.4f\t%0.4f\n' %(count, x1,y1,z1))

e1 = abs(x0-x1);

e2 = abs(y0-y1);

e3 = abs(z0-z1);

count += 1

x0 = x1

y0 = y1

z0 = z1

condition = e1>e and e2>e and e3>e

print('\nSolution: x=%0.3f, y=%0.3f and z = %0.3f\n'% (x1,y1,z1))

**OUTPUT:**

Enter tolerable error: 0.0001

Count x y z

1 0.8500 -1.0275 1.0109

2 1.0025 -0.9998 0.9998

3 1.0000 -1.0000 1.0000

4 1.0000 -1.0000 1.0000

Solution: x=1.000, y=-1.000 and z = 1.000