Jacob Dennis

Physics 411

2 February 2016

Midterm 1

Code and Results

**LU Decomposition**

*Code*

import math

import numpy as np

A = np.asarray([[2.0, 1.0, 4.0, 1.0],

[3.0, 4.0, -1.0, -1.0],

[1.0, -4.0, 1.0, 5.0],

[2.0, -2.0, 1.0, 3.0]])

def createL(M, iterationNum):

L = []

for i in range(len(M)):

dummyRow = []

for j in range(len(M[i])):

if j == iterationNum:

if i == iterationNum:

dummyRow.append(1.0)

elif i > iterationNum:

dummyRow.append(-M[i][j])

else:

dummyRow.append(0.0)

elif i == j:

dummyRow.append(M[iterationNum][iterationNum])

else:

dummyRow.append(0.0)

L.append(dummyRow)

L = np.asarray(L)

return (np.asarray(L) / M[iterationNum][iterationNum])

LList = []

MList = [A]

finalU = np.identity(len(A))

finalL = np.identity(len(A))

names = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H']

for i in range(len(A)):

num = str(i)

L = createL(MList[i], i)

LList.append(L)

M = np.dot(L, MList[i])

MList.append(M)

print 'Round {}'.format(num)

print '{}:\n'.format(names[i]), MList[i]

print 'L{} \* {}[{}][{}]:\n'.format(num, names[i], num, num), L \* MList[i][i][i]

for k in range(len(LList)):

finalL = np.dot(finalL, np.linalg.inv(LList[k]))

for j in np.flipud(np.arange(len(LList))):

finalU = np.dot(finalU, LList[j])

finalU = np.dot(finalU, A)

print '\nL:\n', finalL

print 'U:\n', finalU

print 'Test: L \* U\n', np.dot(finalL, finalU)

*Results*

Round 0

A:

[[ 2. 1. 4. 1.]

[ 3. 4. -1. -1.]

[ 1. -4. 1. 5.]

[ 2. -2. 1. 3.]]

L0 \* A[0][0]:

[[ 1. 0. 0. 0.]

[-3. 2. 0. 0.]

[-1. 0. 2. 0.]

[-2. 0. 0. 2.]]

Round 1

B:

[[ 1. 0.5 2. 0.5]

[ 0. 2.5 -7. -2.5]

[ 0. -4.5 -1. 4.5]

[ 0. -3. -3. 2. ]]

L1 \* B[1][1]:

[[ 2.5 0. 0. 0. ]

[ 0. 1. 0. 0. ]

[ 0. 4.5 2.5 0. ]

[ 0. 3. 0. 2.5]]

Round 2

C:

[[ 1. 0.5 2. 0.5]

[ 0. 1. -2.8 -1. ]

[ 0. 0. -13.6 0. ]

[ 0. 0. -11.4 -1. ]]

L2 \* C[2][2]:

[[-13.6 0. 0. 0. ]

[ 0. -13.6 0. 0. ]

[ 0. 0. 1. 0. ]

[ 0. 0. 11.4 -13.6]]

Round 3

D:

[[ 1. 0.5 2. 0.5]

[ 0. 1. -2.8 -1. ]

[ 0. 0. 1. 0. ]

[ 0. 0. 0. -1. ]]

L3 \* D[3][3]:

[[-1. 0. 0. 0.]

[ 0. -1. 0. 0.]

[ 0. 0. -1. 0.]

[ 0. 0. 0. 1.]]

L:

[[ 2. 0. 0. 0. ]

[ 3. 2.5 0. 0. ]

[ 1. -4.5 -13.6 0. ]

[ 2. -3. -11.4 -1. ]]

U:

[[ 1.00000000e+00 5.00000000e-01 2.00000000e+00 5.00000000e-01]

[ 0.00000000e+00 1.00000000e+00 -2.80000000e+00 -1.00000000e+00]

[ 1.38777878e-17 5.55111512e-17 1.00000000e+00 0.00000000e+00]

[ 0.00000000e+00 0.00000000e+00 0.00000000e+00 1.00000000e+00]] Test: L \* U

[[ 2. 1. 4. 1.]

[ 3. 4. -1. -1.]

[ 1. -4. 1. 5.]

[ 2. -2. 1. 3.]]

**Accuracy and Calculation**

*Code*

import math

import numpy as np

def f1(x):

return (-3.0 \* x\*\*2.0 + x) / (3.0 \* x\*\*2 + 4.0 \* x + 1.0)

def f2(x):

return (1.0 - x) / (1.0 + x) - 1.0 / (3.0\*x + 1)

def f3(x, y):

return 2.0 \* math.cos((x + y) / 2.0) \* math.sin((x - y) / 2.0)

def f4(x, y):

return math.sin(x) - math.sin(y)

exponents = np.arange(-17.0, 1.0, 1.0)

X = 10.0 \*\* exponents

X64 = np.float64(X)

X32 = np.float32(X)

X16 = np.float16(X)

print 'Function 1: f(x) = (1-x)/(1+x) - 1/(3x+1)'

print ' x | error'

for i in range(len(X)):

print '10e{} |'.format(exponents[i]), f1(X64[i]) - f2(X64[i])

print ' x | float64 | float32 | float16'

for i in range(len(X)):

print '10e{} |{} |{} |{} '.format(exponents[i], f2(X64[i]), f2(X32[i]), f2(X16[i]))

Y = X + 1.5

Y64 = np.float64(Y)

Y32 = np.float32(Y)

Y16 = np.float16(Y)

print '\nFunction 2: f(x, y) = sin(x) - sin(y), x = 1.5'

print ' (y-x) | error'

for i in range(len(Y)):

print '10e{} |'.format(exponents[i]), f3(np.float64(1.5), Y64[i]) - f4(np.float64(1.5), Y64[i])

print ' (x-y) | float64 | float32 | float16'

for i in range(len(Y)):

print '10e{} |{} |{} |{} '.format(exponents[i], f4(np.float64(1.5), Y64[i]),

f4(np.float32(1.5), Y32[i]), f4(np.float16(1.5), Y16[i]))

*Results*

Function 1: f(x) = (1-x)/(1+x) - 1/(3x+1)

x | error

10e-17.0 |1e-17

10e-16.0 |-1.10223024625e-17

10e-15.0 |7.99277837352e-19

10e-14.0 |7.99277837289e-18

10e-13.0 |-3.10945187966e-17

10e-12.0 |2.21217131215e-17

10e-11.0 |-8.28103710574e-19

10e-10.0 |-8.34403710051e-18

10e-9.0 |2.12819313084e-17

10e-8.0 |-8.41137515223e-17

10e-7.0 |1.07733432347e-16

10e-6.0 |6.34368385187e-17

10e-5.0 |1.15975751138e-17

10e-4.0 |1.26174027823e-17

10e-3.0 |7.09068220806e-17

10e-2.0 |3.12250225676e-17

10e-1.0 |-6.93889390391e-18

10e0.0 |0.0

x | float64 | float32 | float16

10e-17.0 |0.0 |0.0 |0.0

10e-16.0 |1.11022302463e-16 |1.11022302463e-16 |0.0

10e-15.0 |9.99200722163e-16 |9.99200722163e-16 |0.0

10e-14.0 |9.99200722163e-15 |9.99200722163e-15 |0.0

10e-13.0 |1.00031094519e-13 |1.00031094519e-13 |0.0

10e-12.0 |9.9997787828e-13 |9.9997787828e-13 |0.0

10e-11.0 |1.00000008274e-11 |1.00000008274e-11 |0.0

10e-10.0 |1.00000008274e-10 |1.00000008274e-10 |0.0

10e-9.0 |9.99999971718e-10 |9.99999749673e-10 |0.0

10e-8.0 |9.99999938411e-09 |9.99999927309e-09 |0.0

10e-7.0 |9.99999298923e-08 |9.99999311135e-08 |1.19209190075e-07

10e-6.0 |9.99992999962e-07 |9.99992997408e-07 |1.01327177404e-06

10e-5.0 |9.99930002499e-06 |9.99929977241e-06 |1.00128784448e-05

10e-4.0 |9.99300249921e-05 |9.99300224693e-05 |9.99465957041e-05

10e-3.0 |0.00099302492124 |0.000993024968076 |0.000993423647207

10e-2.0 |0.00932423339421 |0.00932423320038 |0.00932608589521

10e-1.0 |0.048951048951 |0.0489510491332 |0.0489480626547

10e0.0 |-0.25 |-0.25 |-0.25

Function 2: f(x, y) = sin(x) - sin(y), x = 1.5

(y-x) | error

10e-17.0 |0.0

10e-16.0 |0.0

10e-15.0 |3.24882324735e-17

10e-14.0 |-4.06728151262e-17

10e-13.0 |3.73610585927e-17

10e-12.0 |-2.2283577003e-17

10e-11.0 |-4.89861666901e-17

10e-10.0 |-4.57679683232e-17

10e-9.0 |-1.31416005523e-17

10e-8.0 |-2.37014766829e-17

10e-7.0 |-1.42552278398e-17

10e-6.0 |2.517711468e-17

10e-5.0 |-6.31182482514e-17

10e-4.0 |3.11081320209e-17

10e-3.0 |-4.89110705063e-17

10e-2.0 |-8.67361737988e-18

10e-1.0 |-6.67868538251e-17

10e0.0 |-5.55111512313e-17

(x-y) | float64 | float32 | float16

10e-17.0 |0.0 |0.0 |0.0

10e-16.0 |0.0 |0.0 |0.0

10e-15.0 |-1.11022302463e-16 |0.0 |0.0

10e-14.0 |-6.66133814775e-16 |0.0 |0.0

10e-13.0 |-7.1054273576e-15 |0.0 |0.0

10e-12.0 |-7.07212066686e-14 |0.0 |0.0

10e-11.0 |-7.07323088989e-13 |0.0 |0.0

10e-10.0 |-7.0736749791e-12 |0.0 |0.0

10e-9.0 |-7.07371938802e-11 |0.0 |0.0

10e-8.0 |-7.07371938802e-10 |0.0 |0.0

10e-7.0 |-7.07371516917e-09 |-8.43252445648e-09 |0.0

10e-6.0 |-7.07367029396e-08 |-6.74597988581e-08 |0.0

10e-5.0 |-7.07322141857e-07 |-7.08282640316e-07 |0.0

10e-4.0 |-7.06873268008e-06 |-7.06990483301e-06 |0.0

10e-3.0 |-7.02384424264e-05 |-7.0241701363e-05 |-6.86036448868e-05

10e-2.0 |-0.000657485893494 |-0.000657485314052 |-0.000643218115762

10e-1.0 |-0.00207861643745 |-0.00207861574128 |-0.00208994623915

10e0.0 |0.3990228425 |0.3990228425 |0.3990228425

**Finite Difference**

*Code*

import math

import numpy as np

def SecondOrderf4(x, h, f):

return 0.0

def FourthOrderf4(x, h, f):

return (-f(x - 2.0 \* h) + 4.0 \* f(x - h) - 6.0 \* f(x) + 4.0 \* f(x + h) - f(x + 2.0 \* h)) / (5.0 \* h\*\*2.0)

hList = np.array([0.1, 0.01, 0.001, 0.0001])

for h in hList:

print '\nh =',h

print 'Second Order:', SecondOrderf4(1.0, h, math.sin)

print 'Fourth Order:', FourthOrderf4(1.0, h, math.sin)

*Results*

h = 0.1

Second Order: 0.0

Fourth Order: -0.00168013916905

h = 0.01

Second Order: 0.0

Fourth Order: -1.68291394154e-05

h = 0.001

Second Order: 0.0

Fourth Order: -1.68309810533e-07

h = 0.0001

Second Order: 0.0

Fourth Order: 1.11022302463e-08

**Errors on Integrals**

*Code*

import math

import numpy as np

import matplotlib.pyplot as plt

outputFile4 = open('Midterm 1 problem 4 results.txt', 'w')

def inverse(x):

return 1.0 / x

def T(f, a, b, h):

N = (b - a) / h

krange = np.arange(1.0, N - 1.0, 2.0)

Tsum = 0.0

for k in krange:

Tsum += f(a + k \* h)

return (Tsum \* (2.0 / 3.0))

def S(f, a, b, h):

N = (b - a) / h

krange = np.arange(2.0, N - 2.0, 2.0)

Ssum = 0.0

for k in krange:

Ssum += f(a + k \* h)

return (1.0 / 3.0 \* (f(a) + f(b) + 2.0 \* Ssum))

iRange = np.arange(0, 23)

SList = [S(inverse, 1.0, 2.0, 1.0)]

TList = [T(inverse, 1.0, 2.0, 1.0)]

IList = [1.0 \* (SList[0] + 2.0 \* TList[0])]

hList = [1.0]

hListLog = [math.log10(abs(hList[0]))]

errorList = [1.0]

errorListLog = [math.log10(abs(errorList[0]))]

outputFile4.write('Generation number | Error | Integral\n')

for i in iRange[1:]:

hList.append(hList[i - 1] / 2.0)

SList.append(SList[i - 1] + TList[i - 1])

TList.append(T(inverse, 1.0, 2.0, hList[i]))

IList.append(hList[i] \* (SList[i] + 2.0 \* TList[i]))

errorList.append(1.0/15.0 \* (IList[i] - IList[i - 1]))

errorListLog.append(math.log10(abs(errorList[i])))

hListLog.append(math.log10(abs(hList[i])))

outputFile4.write('{} |{} |{}\n'.format(i, errorList[i], IList[i]))

plt.clf()

plt.plot(errorListLog, hListLog)

plt.title('log(Error) vs. log(stepsize) for the Simpson rule \nof the function 1/x')

plt.xlabel('log error')

plt.ylabel('log stepsize')

plt.savefig('Midterm 1 graph.png')

outputFile4.close()

*Results*

Generation number | Error | Integral

1 |-0.0166666666667 |0.25

2 |0.00944444444444 |0.391666666667

3 |0.00729495812829 |0.501091038591

4 |0.00501575584871 |0.576327376322

5 |0.00319989084848 |0.624325739049

6 |0.0019468683586 |0.653528764428

7 |0.00114700806651 |0.670733885426

8 |0.00066030494503 |0.680638459601

9 |0.000373554670331 |0.686241779656

10 |0.000208478651589 |0.68936895943

11 |0.000115090011183 |0.691095309598

12 |6.29703516816e-05 |0.692039864873

13 |3.41978493105e-05 |0.692552832613

14 |1.8455261443e-05 |0.692829661534

15 |9.9057991222e-06 |0.692978248521

16 |5.29198376193e-06 |0.693057628277

17 |2.81553398197e-06 |0.693099861287

18 |1.49253804058e-06 |0.693122249358

19 |7.88654545566e-07 |0.693134079176

20 |4.15520037015e-07 |0.693140311977

21 |2.1835639948e-07 |0.693143587323

22 |1.14476389013e-07 |0.693145304468

(see attached graph)