# MIDTERM 1 PROGRAMMING ASSIGNMENT

LIAM DUNGAN

## $\mathbf{A}_1$:

For matrix $A_1$, method (a) proved far more accurate. Method (a) shows a relative error of roughly 2.8% while method (b) shows an error between 20-40%. However, the results did improve when tested using a higher number of randomly generated vectors $y_i$. The results of 5 different test cases are shown below.

```
         cond(A): 12.7741935484
cond(A) method a:       12.4160902151    error:  2.80333417461 %
cond(A) method b:       9.58165040344    error:  24.9921306801 %
nwk-148-162:desktop liamdungan$ python midterm.py

         cond(A): 12.7741935484
cond(A) method a:       12.4160902151    error:  2.80333417461 %
cond(A) method b:       9.89428982989    error:  22.5447008266 %
nwk-148-162:desktop liamdungan$ python midterm.py

         cond(A): 12.7741935484
cond(A) method a:       12.4160902151    error:  2.80333417461 %
cond(A) method b:       5.78666485836    error:  54.7003508563 %
nwk-148-162:desktop liamdungan$ python midterm.py

         cond(A): 12.7741935484
cond(A) method a:       12.4160902151    error:  2.80333417461 %
cond(A) method b:       8.31826002711    error:  34.8823078686 %
nwk-148-162:desktop liamdungan$ python midterm.py

         cond(A): 12.7741935484
cond(A) method a:       12.4160902151    error:  2.80333417461 %
cond(A) method b:       8.09923584338    error:  36.5968911251 %
nwk-148-162:desktop liamdungan$ python midterm.py
```

## $\mathbf{A}_2$:

Similarly, for matrix $A_2$, method (a) was more accurate. This method shows a relative error of roughly 2.0% while method (b) shows a relative error between 20-40%. The results of 5 different test cases are shown below.

---

*Date*: Nov. 5, 2017.

```
        cond(A): 4016285.00021
cond(A) method a:       3936928.52587    error:  1.97586760753 %
cond(A) method b:       2735196.19297    error:  31.8973580603 %
nwk-148-162:desktop liamdungan$ python midterm.py

        cond(A): 4016285.00021
cond(A) method a:       3936928.52587    error:  1.97586760753 %
cond(A) method b:       2835171.03207    error:  29.4081213879 %
nwk-148-162:desktop liamdungan$ python midterm.py

        cond(A): 4016285.00021
cond(A) method a:       3936928.52587    error:  1.97586760753 %
cond(A) method b:       3052935.59527    error:  23.9860817867 %
nwk-148-162:desktop liamdungan$ python midterm.py

        cond(A): 4016285.00021
cond(A) method a:       3936928.52587    error:  1.97586760753 %
cond(A) method b:       2345283.10366    error:  41.6056603668 %
nwk-148-162:desktop liamdungan$ python midterm.py

        cond(A): 4016285.00021
cond(A) method a:       3936928.52587    error:  1.97586760753 %
cond(A) method b:       3085835.0787     error:  23.1669296741 %
nwk-148-162:desktop liamdungan$
```

**Code:**

```
#Liam Dungan
#Nov. 5, 2017
#CS 323 - midterm 1 programming assignment

import numpy as np
from scipy.linalg import lu

#Custom forward substitution for lower triangular system
def Forward_Substitution(A,b,x):
    m=A.shape[0]
    n=A.shape[1]
    if(m!=n):
        print 'Invalid: Matrix is not square'
        return
    for j in range(0,n):
        if A[j,j] == 0:
            print 'Invalid: Matrix is singular'
            return          # matrix is singular

        x[j] = b[j]/A[j,j]
        for i in range(j+1,n):
            if( abs(-1. - A[i,j]*x[j]) >= abs(b[i] - A[i,j]*x[j]) ):
                b[i]=-1.
            b[i] = b[i] - A[i,j]*x[j]


#this routine calculates and prints the condition number
def getCond(A1):
#------METHOD-A--------

    #LU Decomposition
    P,L,U = lu(A1,permute_l = False)

    #transpose the triangular matrices
    Lt = np.transpose(L)
    Ut = np.transpose(U)

```

```
    #Solve for Y using method A (forward Sub)
    c=np.array([1.,1.,1.])
    v=np.zeros(3)
    Forward_Substitution(Ut,c,v)
    y_a = np.linalg.solve(Lt,v)

    #Find Ratio ||A^-1||
    z_a = np.linalg.solve(A1,y_a)
    ratio_a = np.linalg.norm(z_a)/np.linalg.norm(y_a)


#------METHOD-B--------

    #5 ranodmly generated Ys
    Y = np.array([np.random.rand(3,1),
        np.random.rand(3,1),
        np.random.rand(3,1),
        np.random.rand(3,1),
        np.random.rand(3,1)])
    ratio_b = ( np.linalg.norm(np.linalg.solve(A1,Y[0]) ) / np.linalg.norm(Y[0]) )

    for i in range(5):
        z_norm = np.linalg.norm( np.linalg.solve(A1,Y[i]) )
        y_norm = np.linalg.norm(Y[i])
        if( (z_norm/y_norm) > ratio_b):
            ratio_b = (z_norm/y_norm)

#------Results--------

    #Find l1 norm of matrix A
    n=A1.shape[1]
    normA = 0
    for i in range(0,n):
        vectorNorm = np.linalg.norm(A1[:,i],1)
        if(vectorNorm > normA):
            normA = vectorNorm

    #Actual conition number for comparison
    actualCond = np.linalg.cond(A1,1)
```

```
77      print "\n \t cond(A):", actualCond
78
79      #Estimate condition number
80      cond_methodA = normA * ratio_a
81      cond_methodB = normA * ratio_b
82
83      errorA = (abs(cond_methodA - actualCond)/actualCond )*100
84      errorB = (abs(cond_methodB - actualCond)/actualCond )*100
85      print "cond(A) method a:\t",cond_methodA,"\t error: ", errorA,"%"
86      print "cond(A) method b:\t",cond_methodB,"\t error: ", errorB,"%"
87
88
89
90
91  def main():
92      #test matrices
93      #A1 = np.array([[-10.,7.,0.],[5.,-1.,5.],[-3.,2.,6.]])
94      A2 = np.array([[92.,66.,25.],[-73.,78.,24.],[-80.,37.,10.]])
95      getCond(A2)
96
97
98  if __name__ == "__main__":
99      main()
100
```