Wang_T_1

July 12, 2018

1 Tianqi Wang

1.1 Homework 1

```
In [56]: # Import related packages
         import numpy as np
         import pandas as pd
```

1.1.1 Question A

```
In [ ]: # Check the linalg module
        np.linalg??
```

```
In [50]: # Generate Coefficient Matrix A
         A=np.array(np.arange(1,101))
         A=A.reshape(10,10)
         print(A)
```

```
[[  1   2   3   4   5   6   7   8   9  10]
 [ 11  12  13  14  15  16  17  18  19  20]
 [ 21  22  23  24  25  26  27  28  29  30]
 [ 31  32  33  34  35  36  37  38  39  40]
 [ 41  42  43  44  45  46  47  48  49  50]
 [ 51  52  53  54  55  56  57  58  59  60]
 [ 61  62  63  64  65  66  67  68  69  70]
 [ 71  72  73  74  75  76  77  78  79  80]
 [ 81  82  83  84  85  86  87  88  89  90]
 [ 91  92  93  94  95  96  97  98  99 100]]
```

```
In [38]: # Check linalge.solve module
         np.linalg.solve?
```

```
In [51]: # Generate v
         v=np.array(np.arange(1,11))
         print(v)
```

```
[ 1  2  3  4  5  6  7  8  9 10]
```

In [52]:
```
# Generate b
b=np.array(np.ones(10))
print(b)
```

```
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

In [53]:
```
# Compute Av
print(A*v)
```

```
[[   1    4    9   16   25   36   49   64   81  100]
 [  11   24   39   56   75   96  119  144  171  200]
 [  21   44   69   96  125  156  189  224  261  300]
 [  31   64   99  136  175  216  259  304  351  400]
 [  41   84  129  176  225  276  329  384  441  500]
 [  51  104  159  216  275  336  399  464  531  600]
 [  61  124  189  256  325  396  469  544  621  700]
 [  71  144  219  296  375  456  539  624  711  800]
 [  81  164  249  336  425  516  609  704  801  900]
 [  91  184  279  376  475  576  679  784  891 1000]]
```

In [54]:
```
# Solve Ax=v
print(np.linalg.solve(A,v))
```

```
---------------------------------------------------------------------------

LinAlgError                               Traceback (most recent call last)

<ipython-input-54-714a3f0fe25b> in <module>()
  1 # Solve Ax=v
----> 2 print(np.linalg.solve(A,v))


~/anaconda3/lib/python3.6/site-packages/numpy/linalg/linalg.py in solve(a, b)
388     signature = 'DD->D' if isComplexType(t) else 'dd->d'
389     extobj = get_linalg_error_extobj(_raise_linalgerror_singular)
--> 390     r = gufunc(a, b, signature=signature, extobj=extobj)
391
392     return wrap(r.astype(result_t, copy=False))


~/anaconda3/lib/python3.6/site-packages/numpy/linalg/linalg.py in _raise_linalgerror_sin
 87
 88 def _raise_linalgerror_singular(err, flag):
---> 89     raise LinAlgError("Singular matrix")
 90
 91 def _raise_linalgerror_nonposdef(err, flag):
```

```
LinAlgError: Singular matrix
```

```
In [55]: # Solve Ax=b
         print(np.linalg.solve(A,b))
```

```
---------------------------------------------------------------------------
LinAlgError                               Traceback (most recent call last)

<ipython-input-55-c2df154952d6> in <module>()
  1 # Solve Ax=b
----> 2 print(np.linalg.solve(A,b))

~/anaconda3/lib/python3.6/site-packages/numpy/linalg/linalg.py in solve(a, b)
388     signature = 'DD->D' if isComplexType(t) else 'dd->d'
389     extobj = get_linalg_error_extobj(_raise_linalgerror_singular)
--> 390     r = gufunc(a, b, signature=signature, extobj=extobj)
391
392     return wrap(r.astype(result_t, copy=False))

~/anaconda3/lib/python3.6/site-packages/numpy/linalg/linalg.py in _raise_linalgerror_sin
 87
 88 def _raise_linalgerror_singular(err, flag):
---> 89     raise LinAlgError("Singular matrix")
 90
 91 def _raise_linalgerror_nonposdef(err, flag):

LinAlgError: Singular matrix
```

1.1.2 Question B

```
In [57]: # Check numpy.random package
         np.random?
```

```
In [59]: # Check numpy.random.normal module
         np.random.normal?
```

```
In [62]: # Generate "shake factor" R
         R=np.array(np.random.normal(0,10,100))
         R=R.reshape(10,10)
         print(R)
```

```
[[  4.00609149  -2.59635585   9.18160947  -1.89900667  -6.6130337
    3.08153922   5.61944874 -17.55300314 -10.66343588   7.11785298]
 [ -2.60786806   4.66707601   7.62373632   1.69709429  -0.84000761
  -17.24871642 -11.04258779  -3.43734063  -1.82363213   1.24975022]
 [ 20.56424203  11.90288829  -0.0530191  -10.40481846  -8.86149229
   -6.52637544  -7.17166045  -8.30664131  25.50854555  -5.45919632]
 [ 13.70773458  -8.14566443   5.85731177  -3.57354484   0.23081293
   -2.09457247  -4.33255473   3.55085853  -9.61348733   0.0320571 ]
 [  7.64142947   3.7705118    3.10374816  11.58623991  -9.88539621
   14.59489216  -0.93421316   1.60593661   1.1250553  -15.51472923]
 [  7.01252349   9.82021824   2.44212413   1.8484397    4.92607591
   -7.0841132    7.65226571   8.52246221  -3.13201329  -8.64637734]
 [  3.96492921  13.63572654   4.41731675  -2.74202944 -19.55056735
   -8.67281697   6.48928316   5.59584535  -2.32423256   9.90131888]
 [ -6.50058423   9.13053576  13.48060563   5.83553297  10.83572225
   13.2510233   17.25236866  -3.81106451  -8.11977435   7.78628336]
 [  5.28010583 -18.61041344   3.51274168  -7.91331213   0.59884203
   -9.04838058   5.26583951   5.15077056  -0.66655377   3.11036194]
 [  9.48466217 -20.10735012   1.70869345  -6.14259788  -1.59336848
   -8.6888861    0.969155   -10.51024661  13.85662956   9.44628298]]
```

In [64]:
```
# Combine A and R
AR=A+R
print(AR)
```

```
[[  5.00609149  -0.59635585  12.18160947   2.10099333  -1.6130337
    9.08153922  12.61944874  -9.55300314  -1.66343588  17.11785298]
 [  8.39213194  16.66707601  20.62373632  15.69709429  14.15999239
   -1.24871642   5.95741221  14.56265937  17.17636787  21.24975022]
 [ 41.56424203  33.90288829  22.9469809   13.59518154  16.13850771
   19.47362456  19.82833955  19.69335869  54.50854555  24.54080368]
 [ 44.70773458  23.85433557  38.85731177  30.42645516  35.23081293
   33.90542753  32.66744527  41.55085853  29.38651267  40.0320571 ]
 [ 48.64142947  45.7705118   46.10374816  55.58623991  35.11460379
   60.59489216  46.06578684  49.60593661  50.1250553   34.48527077]
 [ 58.01252349  61.82021824  55.44212413  55.8484397   59.92607591
   48.9158868   64.65226571  66.52246221  55.86798671  51.35362266]
 [ 64.96492921  75.63572654  67.41731675  61.25797056  45.44943265
   57.32718303  73.48928316  73.59584535  66.67576744  79.90131888]
 [ 64.49941577  81.13053576  86.48060563  79.83553297  85.83572225
   89.2510233   94.25236866  74.18893549  70.88022565  87.78628336]
 [ 86.28010583  63.38958656  86.51274168  76.08668787  85.59884203
   76.95161942  92.26583951  93.15077056  88.33344623  93.11036194]
 [100.48466217  71.89264988  94.70869345  87.85740212  93.40663152
   87.3111139   97.969155    87.48975339 112.85662956 109.44628298]]
```

In [65]:
```
# Solve previous questions using new matrix AR
```

```
        print(AR*v)
        print(np.linalg.solve(AR,v))
        print(np.linalg.solve(AR,b))
```

```
[[   5.00609149   -1.19271171   36.5448284     8.4039733    -8.0651685
    54.48923532   88.33614119  -76.42402511  -14.97092292  171.17852983]
 [   8.39213194   33.33415202   61.87120895   62.78837715   70.79996196
    -7.49229854   41.70188547  116.50127496  154.58731085  212.49750222]
 [  41.56424203   67.80577657   68.84094269   54.38072618   80.69253854
   116.84174737  138.79837682  157.54686951  490.57690994  245.40803682]
 [  44.70773458   47.70867114  116.57193531  121.70582065  176.15406463
   203.4325652   228.6721169   332.40686821  264.47861404  400.32057104]
 [  48.64142947   91.5410236   138.31124448  222.34495964  175.57301895
   363.56935295  322.46050789  396.84749286  451.12549767  344.85270772]
 [  58.01252349  123.64043647  166.3263724   223.39375881  299.63037954
   293.49532078  452.56585995  532.17969771  502.81188038  513.53622658]
 [  64.96492921  151.27145307  202.25195025  245.03188224  227.24716324
   343.9630982   514.42498212  588.76676279  600.08190693  799.01318878]
 [  64.49941577  162.26107151  259.44181689  319.3421319   429.17861125
   535.50613982  659.76658059  593.51148393  637.92203089  877.86283364]
 [  86.28010583  126.77917311  259.53822504  304.34675147  427.99421014
   461.70971654  645.86087654  745.2061645   795.00101609  931.10361943]
 [ 100.48466217  143.78529977  284.12608036  351.42960849  467.0331576
   523.86668341  685.784085    699.91802711 1015.70966602 1094.46282978]]
[ 0.02783796 -0.00789099  0.09182013  0.01935992 -0.01363405 -0.02425736
  0.01829894 -0.00102032  0.01029008 -0.01919096]
[ 0.03728427  0.00485838  0.14454765 -0.00505227 -0.02486605 -0.03741092
  0.00392012 -0.03384067 -0.01406743 -0.06019859]
```

Now we can verify after the "shake", previous questions could be computed.

1.1.3 Question C

```
In [209]: # Define the function BackSub
          def BackSub(A,b):
              if ((len(A.shape)!=2) # A must be a matrix
              or (A.shape[0]!=A.shape[1]) # A must be a square matrix
              or ((A == np.triu(A)).sum()!=np.square(A.shape[0])) # A must be a upper triangle s
              or (np.linalg.det(A)==0) # A's diagonals are all nonzero
              or (len(b.shape)!=1) # b must be a vector
              or (b.shape[0]!=A.shape[0])) :# dimension of b must equal to a
                  raise TypeError('Inputs do not satisfy requirements')
              else:
                  for i in range(len(b)-1,-1,-1): # Start the backwards elimination from the las
                      b[i]=b[i]/A[i,i]
                      A[i,i]=1
                      for j in range(i-1,-1,-1): # In each column, do the elimination from the d
```

```
                    b[j]=b[j]-A[j,i]*b[i]
                    A[j,i]=0
        return b
```

(a) Test function on problem 3

```
In [214]: test1=np.array([[1,-3,4],
                          [0,1,2],
                          [0,0,1]])

In [215]: b=np.array([7,2,5])

In [216]: BackSub(test1,b)

Out[216]: array([-37,  -8,   5])

In [218]: np.linalg.solve(test,b)

Out[218]: array([-37.,  -8.,   5.])
```

(b) Test function on a 3 Œ 3 matrix that is not upper triangular

```
In [220]: test2=np.array([[1,-3,4],
                          [0,1,2],
                          [3,2,1]])
          b=np.array([7,2,5])
          BackSub(test2,b)


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

        TypeError                                 Traceback (most recent call last)

        <ipython-input-220-9189978b9114> in <module>()
          3               [3,2,1]])
          4 b=np.array([7,2,5])
    ----> 5 BackSub(test2,b)


        <ipython-input-209-536d4bf7c5ff> in BackSub(A, b)
          7     or (len(b.shape)!=1) # b must be a vector
          8     or (b.shape[0]!=A.shape[0])) :# dimension of b must equal to a
    ----> 9         raise TypeError('Inputs do not satisfy requirements')
         10     else:
         11         for i in range(len(b)-1,-1,-1):


        TypeError: Inputs do not satisfy requirements
```

(c) Test function on a 3 Œ 2 matrix

```
In [221]: test3=np.array([[1,-3],
                          [0,1],
                          [0,2]])
          b=np.array([7,2,5])
          BackSub(test3,b)
```

```
        ---------------------------------------------------------------------------

        TypeError                                 Traceback (most recent call last)

        <ipython-input-221-c4f808e976bb> in <module>()
          3               [0,2]])
          4 b=np.array([7,2,5])
    ----> 5 BackSub(test3,b)


        <ipython-input-209-536d4bf7c5ff> in BackSub(A, b)
          7     or (len(b.shape)!=1) # b must be a vector
          8     or (b.shape[0]!=A.shape[0])) :# dimension of b must equal to a
    ----> 9         raise TypeError('Inputs do not satisfy requirements')
         10     else:
         11         for i in range(len(b)-1,-1,-1):


        TypeError: Inputs do not satisfy requirements
```

(d) Test function on Problem 5

```
In [222]: test4=np.array([[1,-3,7],
                          [0,1,4],
                          [0,0,0]])
          b=np.array([1,0,5])
          BackSub(test4,b)
```

```
        ---------------------------------------------------------------------------

        TypeError                                 Traceback (most recent call last)

        <ipython-input-222-87f4ce849b14> in <module>()
          3               [0,0,0]])
          4 b=np.array([1,0,5])
    ----> 5 BackSub(test4,b)
```

```
    <ipython-input-209-536d4bf7c5ff> in BackSub(A, b)
      7     or (len(b.shape)!=1) # b must be a vector
      8     or (b.shape[0]!=A.shape[0])) :# dimension of b must equal to a
----> 9         raise TypeError('Inputs do not satisfy requirements')
     10     else:
     11         for i in range(len(b)-1,-1,-1):


    TypeError: Inputs do not satisfy requirements
```