# Math 645 Problem Set 1

Tucker DiNapoli

*<2016-09-28 Wed>*

1. §1.1 #33,34

| | | | |
|---|---|---|---|
| | 20 | 20 | |
| 10 | T1 | T2 | 40 |
| 10 | T4 | T3 | 40 |
| | 30 | 30 | |

4T1 = 30 + T2 + T4 -> 4T1 - T2 - 0T3 - T4 = 30
4T2 = 60 + T1 + T3 -> -T1 + 4T2 - T3 - 0T4 = 60
4T3 = 70 + T2 + T4 -> 0T1 - T2 + 4T3 - T4 = 70
4T4 = 40 + T3 + T1 -> -T1 -0T2 - T3 + 4T4 = 40

Written out as an augmented matrix Ab this is:

| | | | | |
|---|---|---|---|---|
| 4 | -1 | 0 | -1 | 30 |
| -1 | 4 | -1 | 0 | 60 |
| 0 | -1 | 4 | -1 | 70 |
| -1 | 0 | -1 | 4 | 40 |

After performing row reduction we get:

| | | | | |
|---|---|---|---|---|
| -1 | 0 | -1 | 4 | 40 |
| 0 | -1 | 4 | -1 | 70 |
| 0 | 0 | 16 | -8 | 300 |
| 0 | 0 | 0 | 12 | 270 |

And solving gives:
x = [20 , 27.5, 30, 22.5] The following code performs row reduction and solves using back substution:

```
import numpy as np;
from numpy import array;
import scipy as sci;
import scipy.linalg as la;
import matplotlib.pyplot as plt;
import operator as op;
from functools import reduce;
lmap = lambda f, *lists: list(map(f,*lists))
#The above imports/defination are assumed in the rest of the code fragments
```

```
A = array([[4,-1,0,-1,30],[-1,4,-1,0,60],
           [0,-1,4,-1,70],[-1,0,-1,4,40]],dtype=float)
A[[3,0]] = A[[0,3]]
A[[2,1]] = A[[1,2]]
A[2]-=A[0]
A[2]+=4*A[1]
A[3] += 4*A[0]
A[3] -= A[1]
A[3] += 0.5*A[2]
print(A)
b = A[:,4]
x = np.zeros(4)
for i in range(n-1,-1,-1):
    sum = reduce(op.add,map(lambda j: C[i,j]*x[j],range(i+1,n)),0)
    x[i] = (b[i]-sum)/A[i,i]
print(x)
```

2. (a) The fact that matrix multiplication is not communicative can be shown simply by doing out the multiplication of two matrices A and B as AB and BA which shows that they are not equal.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| x | y | * | a | b | = | ax+cy | bx+dy |
| z | w | | c | d | | az+cw | bz+dw |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| a | b | * | x | y | = | ax+bz | ay+bw |
| c | d | | z | w | | cx+dz | cy+dw |

   (b) See a

   (c) We can see this by comparing the equations for each element in AB and BA. ∃A,B,i,j such that: (AB[i,j] = sum(k=0…n,A[i,k]*B[k,j]) ≠ (BA[i,j] = sum(k=0…n,B[i,k]*A[k,j])

3. §1.2 #33,34 The following code generates the plots for these two problems

```
import numpy as np;
# 1.2 33,34
def find_interpolating_polynomial(points):
    degree = len(points)
    x,y = zip(*points)
    A = np.array(lmap(lambda x: lmap(lambda t: x**t, range(degree)), x))
    b = np.array(y)
    x = la.solve(A,b)
    return (A,b,x)
def plot_interpolating_polynomial(points, coefficents, filename):
    x = coefficents
    t = np.linspace(0,10,100)
    f = np.vectorize(lambda t: reduce(op.add,
                                      lmap(lambda i: x[i]*t**i,
                                           range(len(points)))))
    plt.plot(t,f(t));
```

```
    plt.plot(*list(zip(*points)), '.');
    plt.savefig(filename);
    plt.clf()
def problem_33():
    points = ((1,12),(2,15),(3,16))
    (A,b,x) = find_interpolating_polynomial(points)
    plot_interpolating_polynomial(points, x, "problem_33.png")
def problem_34():
    points = ((0,0),(2,2.9),(4,14.8),(6,39.6),(8,74.3),(10,119))
    (A,b,x) = find_interpolating_polynomial(points)
    plot_interpolating_polynomial(points, x, "problem_34.png")
problem_33()
problem_34()
```

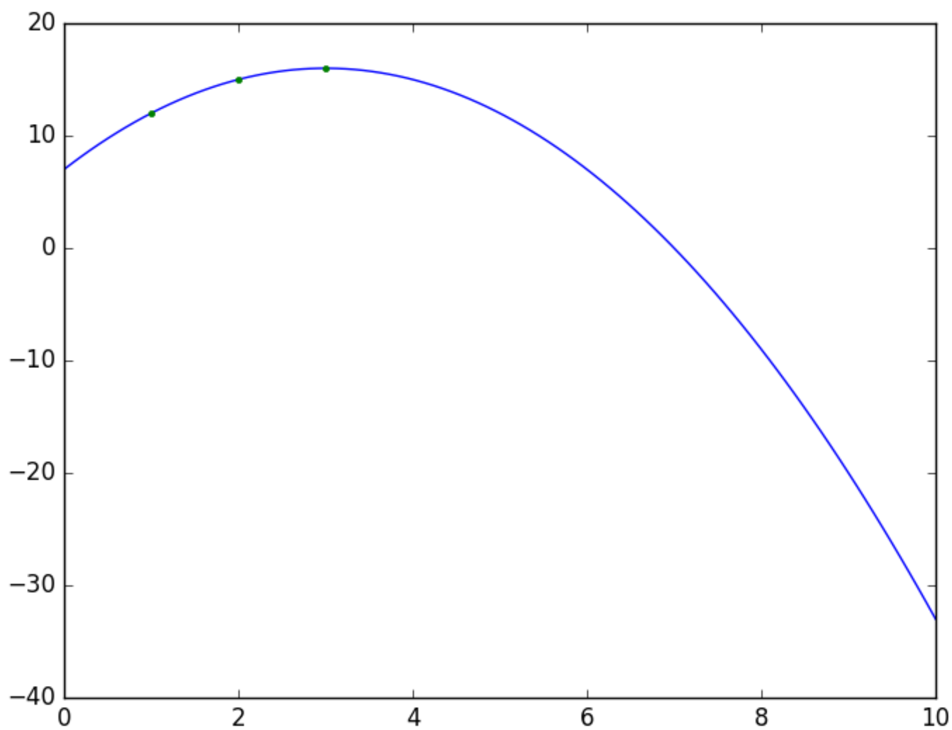


Figure 1: Problem 33 plot

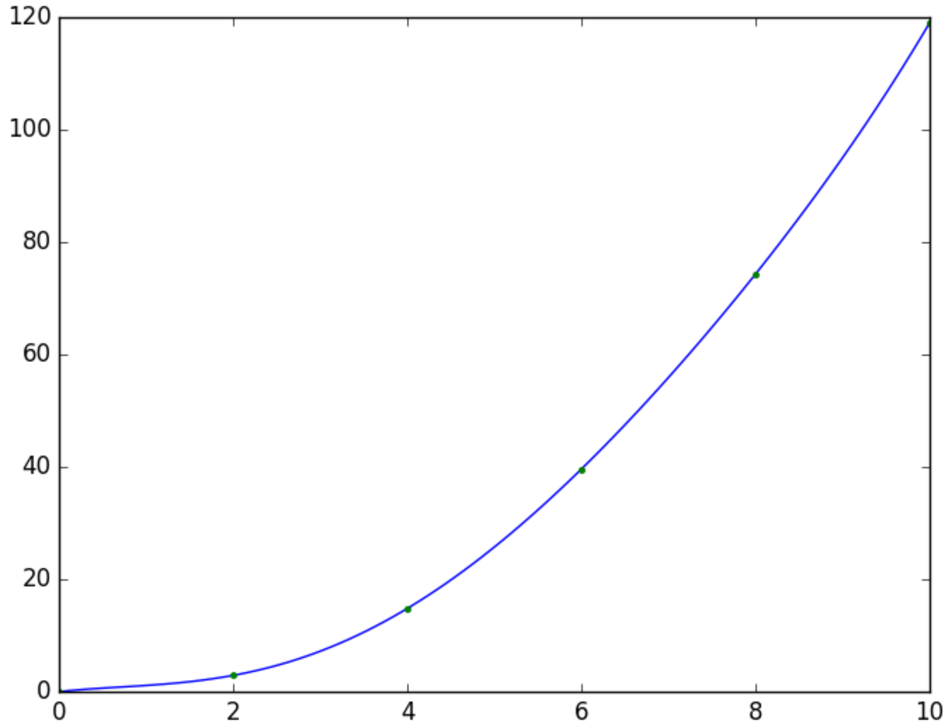


Figure 2: Problem 34 plot

1. §1.3 #28 let x = million btu,
   y = grams sulfur dioxide, and
   z = grams solid particle pollutants
   A: f(t) = t*(27.6x + 3100y + 250z)
   B: f(t) = t*(30.2x + 6400y + 360z)

   (a) 27.6*x1 + 30*x2

   (b) A(x1) + B(x2), or more explicitly:
   A(x1) + B(x2) = [27.6*x1,3100*x1,250*x1] + [30.2*x2,6400*x2,360*x2]

   (c)

   | | | | | | | | | | |
   |---|---|---|---|---|---|---|---|---|---|
   | 27.6 | 3100 | 250 | * | t1 | t2 | = | 162 | 23610 | 1623 |
   | 30.2 | 6400 | 360 | | | | | | | |

   I'm not really sure how to solve this as a vector equation, so I'm just going to solve it by hand.

   t1 = (162-30.2t2)/27.6 ~~ 5.869 - 1.094t2
   1623 = 360t2 + 360*(5.869 - 1.094t2)
   1623 = 633.55t2 + 1467.39
   t2 = 1.8
   t1 = 3.827

So 3.827 tons of A and 1.8 tons of B, with some rounding error.

2. I did this problem using the following code, the function $\text{do}_{\text{problem5}}$ does the actual work and prints out the results, the output is shown below.

```
def do_problem_5(datafile):
    print_arr = lambda x,y: \
                print("{} =\n{}".format(y,
                                        np.array2string(x,precision = 6,
                                                        suppress_small = True,
                                                         separator=',')))
    np.set_printoptions(precision=6)
    A = loadtxt(datafile)
    (n,m) = A.shape
    (LU,p) = lup_decomp(A)
    (LU_control,p_control) = la.lu_factor(A)
    ## Check that my LU is equal to the actual LU, with a small
    ## tolerence for floating point rouding errors
    assert(np.allclose(LU,LU_control));

    L = np.tril(LU)
    U = np.triu(LU)
    P = np.zeros((n,n))
    for i in range(n):
        L[i,i] = 1
        P[i,p[i]] = 1
    print("Problem 5:")
    print("LUP decomposition")
    print_arr(L,"L")
    print_arr(U,"U")
    print_arr(P,"P")

    print("Solving Ax = b for various values of b")

    b1 = array([2,3,-1,5,7],dtype=float)
    x1 = lup_solve(LU,p,b1)
    x1_control = la.lu_solve((LU_control,p_control),b1)
    assert(np.allclose(x1,x1_control));
    print_arr(b1,"b1")
    print_arr(x1,"x1")

    b2 = array([15,29,8,4,-49],dtype=float)
    x2 = lup_solve(LU,p,b2)
    x2_control = la.lu_solve((LU_control,p_control),b2)
    assert(np.allclose(x2,x2_control));
    print_arr(b2,"b2")
```

```
    print_arr(x2,"x2")

    b3 = array([8,-11,3,-8,-32],dtype=float)
    x3 = lup_solve(LU,p,b3)
    x3_control = la.lu_solve((LU_control,p_control),b3 )
    assert(np.allclose(x3,x3_control));
    print_arr(b3,"b3")
    print_arr(x3,"x3")
```

```
Problem 5:
LUP decomposition
L =
[[ 1.      , 0.      , 0.      , 0.      , 0.      ],
 [-0.      , 1.      , 0.      , 0.      , 0.      ],
 [-0.333333, 0.166667, 1.      , 0.      , 0.      ],
 [ 0.666667,-0.833333,-0.5     , 1.      , 0.      ],
 [-0.333333, 0.166667, 0.25    , 0.5     , 1.      ]]
U =
[[ -3.      ,  6.      ,-14.      ,-36.      , -2.      ],
 [  0.      , -6.      ,  4.      ,  6.      ,  0.      ],
 [  0.      ,  0.      , -1.333333, -5.      , -2.666667],
 [  0.      ,  0.      ,  0.      ,  0.5     ,  2.      ],
 [  0.      ,  0.      ,  0.      ,  0.      , -1.      ]]
P =
[[ 0., 0., 0., 0., 1.],
 [ 0., 0., 0., 1., 0.],
 [ 0., 0., 1., 0., 0.],
 [ 0., 1., 0., 0., 0.],
 [ 1., 0., 0., 0., 0.]]
Solving Ax = b for various values of b
b1 =
[ 2., 3.,-1., 5., 7.]
x1 =
[ 19.      ,-18.666667,-47.      , 13.5     , -2.      ]
b2 =
[ 15., 29.,  8.,  4.,-49.]
x2 =
[ 175.      , -37.666667, -57.      ,   1.      ,  30.      ]
b3 =
[  8.,-11.,  3., -8.,-32.]
x3 =
[ 0., 3., 1., 1.,-0.]
```