Sprawozdanie lab2 Agnieszka Pierzcha?a

February 28, 2018

In [1]:
```
import numpy as np
import time

#ZADANIE 1

def gauss_jordan(a):

    n = a.shape[0]
    x= np.zeros(n)
    #print("Array before pivoting")
    #print(a)

    for i in range(n):
        for k  in range(i+1,n):
            if(a[i][i]<a[k][i]):
                for j in range(n+1):
                    tmp = a[i][j];
                    a[i][j]=a[k][j];
                    a[k][j]=tmp;

    #print("Array after pivoting")
    #print(a)

    for i in range(n-1):
        for k in range(i+1,n):
            t = a[k][i]/a[i][i];
            for j in range(n+1):
                a[k][j] = a[k][j] - t*a[i][j];

    #print("Array after gaussian elimination")
    #print(a)



    for i in range(n-1,0,-1):
        for k in range(i-1,-1,-1):
            t = a[k][i]/a[i][i];
```

```
                for j in range(n+1):
                    a[k][j] = a[k][j] - t*a[i][j];

        #print("Array after gaussian elimination up")
        #print(a)

        for s in range(n):
            for j in range(n,-1,-1):
                a[s][j] = a[s][j]/a[s][s];
            x[s] = a[s][n]

        #print(x)
        return x
```

```
In [2]: a= np.array([[0.,2.,1.,-8.],[1.,-2.,-3.,0.],[-1.,1.,2.,3.]]);
        z=gauss_jordan(a)
```

```
In [3]: print(z)
```

```
[-4. -5.  2.]
```

```
In [9]: n=500
        b= np.random.rand(n,n+1);
```

```
In [10]: #measure time
         start = time.time()
         x=gauss_jordan(b)
         end = time.time()
         print('My solver time:',end - start)


         b=np.random.rand(n,n);
         y=np.random.rand(n,1);
         start = time.time()
         x=np.linalg.solve(b,y)
         end = time.time()
         print('Library solver time',end - start)
```

```
My solver time: 227.91824388504028
Library solver time 0.016000986099243164
```

```
In [11]: #ZAD 2
         def scale_matrix(matrix):
             scale_coefficient = matrix.max()
             return matrix/ scale_coefficient, scale_coefficient

         def create_pivot_matrix(a):
```

```
        n = a.shape[0]
        x = np.identity(n)

        for i in range(n):
            maxrow = a[i][i];
            row = i;
            for j in range(i,n):
                if(a[j][i]>maxrow):
                    maxrow = a[j][i];
                    row = j;
            if i != row :
                for k in range(n):
                    tmp = x[i][k];
                    x[i][k]=x[row][k];
                    x[row][k]=tmp;

        return x

    def lu_decomposition(a, scaled= True ):
        n=a.shape[0]
        if(scaled):
            a, coefficient = scale_matrix(a)

        s=(n,n)
        L=np.zeros(s)
        U=np.zeros(s)
        P=create_pivot_matrix(a)
        PA = np.dot(P,a)

        for j in range(n):
            L[j][j] = 1.
            for i in range(j+1):
                s1 = 0.
                for k in range(i):
                    s1+=U[k][j] * L[i][k]
                U[i][j] = PA[i][j] - s1;

            for i in range(j,n):
                s2 = 0.
                for k in range(j):
                    s2+= U[k][j] * L[i][k]
                L[i][j] = (PA[i][j] - s2) / U[j][j]



        return (L,U,P)
```

In [12]:
```
d= np.array([[1.,3.,5.],[2.,4.,7.],[1.,1.,0.]]);
```

```
        e= np.array([[11.,9.,24.,2.],[1.,5.,2.,6.],[3.,17.,18.,1.],[2.,5.,7.,1.]]);
```

In [15]:
```
(a,b) = scale_matrix(d)
print(b)
print(a)

(l,u,p)=lu_decomposition(d,True)
print('printing l')
print(l)
print('printing u')
print(u)
print('printing p')
print(p)
```

```
7.0
[[ 0.14285714  0.42857143  0.71428571]
 [ 0.28571429  0.57142857  1.        ]
 [ 0.14285714  0.14285714  0.        ]]
printing l
[[ 1.   0.   0. ]
 [ 0.5  1.   0. ]
 [ 0.5 -1.   1. ]]
printing u
[[ 0.28571429  0.57142857  1.        ]
 [ 0.          0.14285714  0.21428571]
 [ 0.          0.         -0.28571429]]
printing p
[[ 0.  1.  0.]
 [ 1.  0.  0.]
 [ 0.  0.  1.]]
```