Abgabe - Übungsblatt [8]

Angewandte Mathematik: Numerik

[Felix Lehmann] [Markus Menke]

11. Januar 2021

Aufgabe 1

Here comes your text ...

Aufgabe 2

```
import numpy as np
import scipy.linalg as la

def Pmatrix(A):
    m = len(A)

    id_mat = np.identity(m, dtype=float)

    for j in range(m):
        row = max(range(j, m), key=lambda i: abs(A[i,j]))
        if j != row:
            id_mat[[j, row],:] = id_mat[[row, j],:]
            A[[j, row],:] = A[[row, j],:]

    return id_mat

def LUP(A):
    """Computes and returns an LU decomposition with
       pivoting. The return value
       is a tuple (L,U,P) and fulfills L*U=P*A (* is
          matrix-multiplication)."""
    n = len(A)
    U = np.zeros_like(A)
    L = np.zeros_like(A)
    P = Pmatrix(A)
    PA = P@A

    for j in range(n):
        L[j,j] = 1.0
```

```
        for i in range(j+1):
            s1 = sum(U[k,j] * L[i,k] for k in range(i))
            U[i,j] = PA[i,j] - s1

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

    return (L, U, P)

def ForwardSubstitution(L,b):
    """Solves the linear system of equations L*x=b
       assuming that L is a left lower
       triangular matrix. It returns x as column vector.
          """
    x = np.zeros_like(b)
    x[0] = b[0] / L[0,0]
    for i in range(1,len(b)):
        x[i] = b[i] - sum([L[i,j]*x[j] for j in range(1,i
           )])
    return x

def BackSubstitution(U,b):
    """Solves the linear system of equations U*x=b
       assuming that U is a right upper
       triangular matrix. It returns x as column vector.
          """
    x = np.zeros_like(b)
    n = len(x)
    x[n-1] = b[n-1] / U[n-1,n-1]

    for i in range(0,n-2):
        x[i] = (b[i] - sum([U[i,j]*x[j] for j in range(i
           +1,n-1)])) / U[i,i]
    return x

def SolveLinearSystemLUP(A,b):
    """Given a square array A and a matching vector b
       this function solves the
       linear system of equations A*x=b using a pivoted
          LU decomposition and returns
       x."""
    L,U,_ = LUP(A)
    y = ForwardSubstitution(L,b)
    x = BackSubstitution(U,y)
    return x

def LeastSquares(A,b):
    """Given a matrix A and a vector b this function
       solves the least squares
```

```
        problem of minimizing |A*x-b| and returns the
            optimal x."""


if(__name__=="__main__"):
    # A test matrix where LU fails but LUP works fine
    A=np.array([[1,2, 6],
                [4,8,-1],
                [2,3, 5]],dtype=np.double)
    b=np.array([1,2,3],dtype=np.double)
    # Test the LUP-decomposition
    L,U,P=LUP(A)
    print("L")
    print(L)
    print("U")
    print(U)
    print("P")
    print(P)
    print("Zero (LUP sanity check): "+str(np.linalg.norm(
        np.dot(L,U)-np.dot(P,A))))
    # Test the method for solving a system of linear
        equations
    print("Zero (SolveLinearSystemLUP sanity check): "+
        str(np.linalg.norm(np.dot(A,SolveLinearSystemLUP(A
        ,b))-b)))
    # Test the method for solving linear least squares
    A=np.random.rand(6,4)
    b=np.random.rand(6)
    print("Zero (LeastSquares sanity check): "+str(np.
        linalg.norm(LeastSquares(A,b).flat-np.linalg.lstsq
        (A,b)[0])))
```

Aufgabe 3

```
from math import sqrt
import numpy as np

def cholesky(A):
    n = len(A)
    L = np.zeros_like(A)

    for i in range(n):
        for k in range(i+1):
            tmp_sum = sum(L[i,j] * L[k,j] for j in range(
                k))

            if (i == k):
                L[i,k] = sqrt(A[i,i] - tmp_sum)
```

```
            else:
                L[i,k] = (1.0 / L[k,k] * (A[i,k] -
                    tmp_sum))
    return L

A = np.array( [[6, 3, 4, 8], [3, 6, 5, 1], [4, 5, 10, 7],
    [8, 1, 7, 25]],dtype=np.double)
L = cholesky(A)
R = L.transpose()
print("A:")
print(A)

print("L:")
print(L)

print("R:")
print(R)

print("R* R")
print(R.transpose()@R)
```