num_06_h03_a2c

April 12, 2022

[3]:
```python
# Aufgabe 2 c
import numpy


def zerlegung(A):
    lu = A.copy()
    p = []
    for zeile in range(lu.shape[0] - 1):
        # Diagonale != 0?
        if lu[zeile][zeile] == 0:
            for f in range(zeile, lu.shape[0]):
                if lu[f][zeile] != 0:
                    # Zeilen tauschen
                    tmp = numpy.copy(lu[zeile])
                    lu[zeile] = lu[f]
                    lu[f] = tmp
                    p.append(f + 1)
                    break
        else:
            p.append(zeile + 1)

        for y in range(zeile + 1, lu.shape[0]):
            div = int(lu[y][zeile] / lu[zeile][zeile])
            if div != 0:
                lu[y][zeile] = div
                for x in range(zeile + 1, lu.shape[1]):
                    lu[y][x] = lu[y][x] - (lu[zeile][x] * div)

    print(f"LU =\n{lu}")
    print(f"p = {p}")
    return numpy.array(lu), numpy.array(p)


def permutation(p, u):
    c = numpy.copy(u)
    for x in range(len(p)):
        tmp = numpy.copy(c[x])
```

```
        c[x] = c[p[x] - 1]
        c[p[x] - 1] = tmp
    print(f"Pu = {c}")
    return numpy.array(c)


def vorwaerts(LU, u):
    # L*y=u
    y = []
    for x in range(LU.shape[0]):
        res = u[x]
        for r in range(len(y)):
            if x == r:
                res -= y[r]
            else:
                res -= (LU[x][r] * y[r])
        y.append(res)
    print(f"y = {y}")
    return numpy.array(y)


def rueckwaerts(LU, y):
    # U*z=y
    z = []
    for x in range(LU.shape[0] - 1, -1, -1):
        res = y[x]
        if len(z) == 0:
            res /= LU[x][x]
        else:
            for r in range(0, len(z)):
                res -= (LU[x][LU.shape[1] - r - 1] * z[len(z) - r - 1])
            res /= LU[x][x]
        z.insert(0, res)
    print(f"z = {z}")
    return numpy.array(z)


def teil1u3(A, u):
    LU, p = zerlegung(A)
    c = permutation(p, u)
    z = rueckwaerts(LU, vorwaerts(LU, c))
    return z


def teil2(v, z):
    x = 1 + numpy.dot(numpy.transpose(v), z)
    if x != 0:
```

```
        print("Ad ist regulär")
    else:
        print("Ad ist nicht regulär")

    a = 1 / x
    print(f"a = {a}")
    return a


def teil4(zd, a, v, z):
    xd = zd - numpy.dot(a, numpy.dot(numpy.transpose(v), zd)) * z
    print(f"xd = {xd}")
    return numpy.array(xd)


A = numpy.array([[0, 0, 0, 1], [2, 1, 2, 0], [4, 4, 0, 0], [2, 3, 1, 0]])
u = numpy.array([0, 1, 2, 3])
v = numpy.array([0, 0, 0, 1])
Ad = numpy.array([[0, 0, 0, 1], [2, 1, 2, 1], [4, 4, 0, 2], [2, 3, 1, 3]])
bd = numpy.array([3, 5, 4, 5])

if __name__ == '__main__':
    print("\nBerechne z = A^-1*u als Lösung von A*z = u über die LU-Zerlegung␣
 ↪von A (LU*z = P*u,\n" +
          "also L*y = P*u, U*z = y):")
    z = teil1u3(A, u)
    print("\nÜberprüfe ob 1 + v^T*z = 0 (also ob Â überhaupt regulär ist) und␣
 ↪bestimme   = 1/(1 + v^T*z):")
    a = teil2(v, z)
    print("\nberechne ẑ = A*-1 b̂ als Lösung von Aẑ = b̂ über die LU-Zerlegung␣
 ↪von A (LU*ẑ = P*b̂,\n" +
          "also L*ŷ = P*b̂, U*ẑ = ŷ):")
    zd = teil1u3(Ad, bd)
    print("\nbestimme die Lösung x̂ über x̂ = ẑ -  *(v^T*ẑ)*z:")
    xd = teil4(zd, a, v, z)
```

```
Berechne z = A^-1*u als Lösung von A*z = u über die LU-Zerlegung von A (LU*z =
P*u,
also L*y = P*u, U*z = y):
LU =
[[ 2  1  2  0]
 [ 2  2 -4  0]
 [ 1  1  3  0]
 [ 0  0  0  1]]
p = [2, 3, 4]
Pu = [1 2 3 0]
```

```
y = [1, 0, 2, 0]
z = [-0.8333333333333333, 1.3333333333333333, 0.6666666666666666, 0.0]

Überprüfe ob 1 + v^T*z = 0 (also ob Ã überhaupt regulär ist) und bestimme  =
1/(1 + v^T*z):
Ad ist regulär
a = 1.0

berechne ẑ = A*-1 b̃ als Lösung von Aẑ = b̃ über die LU-Zerlegung von A (LU*ẑ =
P*b̃,
also L*ŷ = P*b̃, U*ẑ = ŷ):
LU =
[[ 2  1  2  1]
 [ 2  2 -4  0]
 [ 1  1  3  2]
 [ 0  0  0  1]]
p = [2, 3, 4]
Pu = [5 4 5 3]
y = [5, -6, 6, 3]
z = [2.5, -3.0, 0.0, 3.0]

bestimme die Lösung x̂ über x̂ = ẑ -  *(v^T*ẑ)*z:
xd = [ 5. -7. -2.  3.]
```

[ ]: