Auftrag 1

- Einführende Fragen:

1. Wie lautet Q, R so, dass $A = Q \cdot R$ gilt?
2. Welche Dimension hat Q, R?
3. Welche Dimension hat das reduzierte Problem?
4. Warum reichen die reduzierten Matrizen Q, R?

1. Die Matrix Q ist orthogonal und R ist eine reguläre obere rechte Dreiecksmatrix.
   Es gilt: $A\vec{x} = \vec{b}$
   $\Rightarrow R\vec{x} = Q^T\vec{b}$

2. Q, R weisen für eine Matrix $A \in \mathbb{R}^{m \times n}$ folgende Dimensionen auf:
   $Q \in \mathbb{R}^{m \times m}$
   $R \in \mathbb{R}^{n \times n}$

3. Das reduzierte Problem hat die Dimensionen:
   $Q_{red} \in \mathbb{R}^{m \times n}$
   $R_{red} \in \mathbb{R}^{n \times n}$

4. Es entstehen bei der QR-Zerlegung $m-n$ Nullzeilen in der R Matrix. Diese haben keinen Einfluss auf $QR = A$ und können deshalb weggelassen werden.

- Gemäss Praktikumsbeschrieb wird die Householder-Transformation implementiert. Mit dem $\underbrace{\text{Kroneckerprodukt}}$ wird die $\underbrace{\text{Householder-Transformation}}$ definiert.

$$w \cdot w^T = (w_i\, w_j)_{i,j=1\ldots m} \in \mathbb{R}^{m \times m}$$

$$H(w) = \mathrm{id} - 2\,\frac{\boxed{w \cdot w^T}}{\langle w, w \rangle}$$

Implementation in Python

```python
def Kronecker(w):
    res = np.zeros([len(w),len(w)],dtype=float)
    for i in range(len(w)):
        res[:,i] = w * w[i]
    return(res)
```

```python
def HouseholderTransformation(w):
    H = np.eye(len(w)) - (Kronecker(w) / (0.5 * np.dot(w,w)))
    return H
```

Spaltenweise wird nun die Householder Transformation auf die gesamte Matrix A angewandt. Die Hyperebene w ist gemäss Beschrieb definiert als:

$$w = y \pm ||y||_2 e_1 \xrightarrow{\text{Implementation in Python}}$$

```
w = y.T + mysign(y[0]) * np.linalg.norm(y) * e(len(y))
```

Mit folgender for-Schleife wird die Transformation durchgeführt:

```
# Initialisieren von Hilfsmatrizen
Anew = A.copy()
onesmax = np.eye(m)
Q = np.eye(m)

# Householder-Transformation für alle Spalten durchführen
for k in range(n):
    y = Anew[k:,k]
    w = y.T + mysign(y[0]) * np.linalg.norm(y) * e(len(y))
    Qk = HouseholderTransformation(w)
    Q = Q@(onesmax + np.pad(Qk,[(k,0),(k,0)]) - np.pad(np.eye(m-k),[(k,0),(k,0)]))
    Anew[k:,k:] = Qk@Anew[k:,k:]
```

Nach durchlaufen der for-Schleife hat man die Matrix Q, R entspricht Anew. Wie in den einführenden Fragen bereits erklärt, können R und Q reduziert werden:

```
# Überflüssigen Teil aus der R und Q Matrix wegschneiden
R = Anew[0:n]
Q = Q[0:m,0:n]
```

Die Zerlegung wird verifiziert, um sicherzustellen dass $Q \cdot R = A \Leftrightarrow Q \cdot R - A = 0$

```
In [91]: # Q*R = A verifizieren, Resultat der Subtraktion Q*R-A muss 0 sein.
         print(np.round(Q@R - A,4))

[[ 0.  0.  0.  0.  0.]
 [ 0. -0. -0. -0.  0.]
 [ 0.  0.  0. -0. -0.]
 [ 0.  0. -0.  0.  0.]
 [ 0.  0.  0.  0. -0.]
 [ 0.  0. -0.  0.  0.]
 [ 0.  0.  0.  0. -0.]
 [ 0. -0. -0.  0. -0.]
 [-0. -0.  0. -0. -0.]
 [ 0. -0. -0.  0. -0.]]
```

✓ $Q \cdot R = A$