Programmieraufgabe IV (6 + 10 + 4 = 20 Punkte)

Abgabe in der Woche 17. bis 21. Dezember

In [368]:
```
import numpy as np
import matplotlib.pyplot as plt
import scipy.linalg as la
```

Teilaufgabe a)

Schreiben Sie eine Funktion, die eine gegebene Matrix A (numpy-array) mittels Householder-Transformationen in eine obere Hessenbergmatrix überführt.

In [549]:
```
def intohessenberg(A):

    ## to be completed ##

    return A
```

Teilaufgabe b)

Implementieren Sie das in der Vorlesung behandelte QR-Verfahren zur Eigenwertbestimmung einer oberen Hessenbergmatrix A (numpy-array). Input der Funktion soll die Matrix A, eine maximale Anzahl von Iterationen $itmax$ und eine Toleranz tol sein, die in ein (sinnvolles) Abbruchkriterium eingeht.

Ausgegeben werden soll ein array mit den Näherungen für die Eigenwerte, geordnet nach Größe. Außerdem soll eine Liste der Eigenwertnäherungen aller Iterationen ausgegeben werden.

Beachten Sie, dass hier die **in der Vorlesung besprochene praktische Umsetzung des QR-Verfahrens zur Eigenwertbestimmung** implementiert werden soll, d.h. ein bloßes Ausrechnen der QR-Zerlegung $A = QR$ und anschließende Zuweisung $A \leftarrow RQ$, explizites Aufstellen und Ausmultiplizieren von Givens-Rotations-Matrizen etc. soll hier *nicht* verwendet werden!

In [550]:
```
def qr_eigenvalues(A, itmax, tol):

    ## to be completed ##

    return eigvals, eigvalslist
```

Testen Sie Ihre Implementierung, indem Sie Ihre Ergebnisse für für eine zufällige symmetrische $10x10$-Matrix und die übliche Matrix $A = \mathrm{tridiag}(-1, 4, 1) \in \mathbb{R}^{5\times 5}$ mit der in numpy vorimplementierten Eigenwertberechnung (wählen Sie eine geeignete Routine) vergleichen.

In [ ]:
```

```

Teilaufgabe c)

Erstellen Sie einen Plot, aus dem für jeden Eigenwert der Matrix die Konvergenzgeschwindigkeit der jeweiligen Eigenwertnäherungen des QR-Verfahrens hervorgeht. Als die "richtigen" Eigenwerte können Sie beispielsweise die mit einer numpy-Routine berechneten Eigenwerte oder die letzte Iterierte Ihres Verfahrens verwenden.

Test-Matrix soll $B = B(a, b, c, d)$ wie unten angegeben sein.

Variieren Sie die Parameter a, b, c, d und interpretieren Sie Ihre Beobachtungen.

In [551]:
```
a = 1
b = 2
c = 3
d = 4

T = 0.5*np.array([[1,1,1,1],[1,1,-1,-1],[-1,1,-1,1],[-1,1,1,-1]])
B = np.diag([a,b,c,d])
B = np.matmul(T.T, np.matmul(B,T))

## to be completed ##
```