Programmieraufgabe III (20 Punkte)

Abgabe in der Woche 13.-17. Mai

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

Teilaufgabe a: 10 Punkte

In dieser Aufgabe wollen wir für einige einfache Beispiele das *Quadratische Penality-Verfahren* untersuchen, wie es auf Übungsblatt 5 beschrieben wird.

Implementieren Sie dazu das quadratische Penalty-Verfahren:

- Übergeben werden sollen Funktionen, die f, g und h sowie ∇f bzw. die Jacobi-Matrizen J_g und J_h berechnen
- Übergeben werden soll auch eine maximale Anzahl von Iterationen $itmax$ sowie der Vergrößerungsfaktor für den Penalty-Parameter und die Toleranz für die Verletzung der Nebenbedingungen
- Das unrestringierte Optimierungsproblem in jeder Iteration soll mit Hilfe des Gradientenverfahrens mit Armijo-Linesearch gelöst werden. Sie dürfen hierfür z.B. ihren Code aus früheren Aufgaben verwenden. Wählen Sie passende Parameter für das Gradientenverfahren (Startwerte, Abbruchkriterium, Armijo-Parameter...).
- Ausgegeben werden soll die letzte Iterierte, eine Liste mit Größe der Verletzung der Nebenbedingungen, der letzte Penalty-Parameter und die Norm des Gradienten des unrestringierten Problems in der letzten Iterierten

In [ ]:
```
```

Teilaufgabe b: 10 Punkte

Testen Sie Ihre Implementierung an den folgenden drei Testbeispielen. Die Funktionen für die Auswertung von f, g, h bzw. ihren Ableitungen sind bereits implementiert.

- Für Beispiele i und iii variieren Sie jeweils die Wahl von z, um zu testen, ob das Penalty-Verfahren immer sinnvolle Ergebnisse liefert. Wählen Sie jeweils z, sodass eine, zwei oder keine Ungleichungsnebenbedingungen aktiv sind. Wählen Sie dazu einen anfänglichen Penalty-Parameter von 1 und einen Vergrößerungsfaktor von 2.
- Variieren Sie für alle drei Beispiele jeweils den Vergrößerungsfaktor für den Penalty-Parameter. Was können Sie hinsichtlich der Abnahme der Nebenbedingunsverletzung beobachten?
- Was beobachten Sie, wenn Sie einen sehr großen anfänglichen Penalty-Parameter wählen? Was beobachten Sie, wenn Sie große Vergrößerungsfaktoren wählen?

Testbeispiel i)

$$\min_{x\in\mathbb{R}^2} f(x) := \frac{1}{2}\|x-z\|_2^2$$

unter der Nebenbedingung

$$g(x) := \begin{pmatrix} -x_1 \\ -x_2 \end{pmatrix} \leq 0$$

sowie $z \in \mathbb{R}^2$.

Testbeispiel ii)

$$\min_{x\in\mathbb{R}^n} f(x) := -x_1$$

unter der Nebenbedingung

$$h(x) = \begin{pmatrix} x_1 + x_2 + \ldots + x_n \\ x_1^3 + x_2^3 + \ldots + x_n^3 - 1 \end{pmatrix} = 0.$$

Testbeispiel iii)

$$\min_{x\in\mathbb{R}^2} f(x) := \frac{1}{2}\|x-z\|_2^2$$

unter der Nebenbedingung

$$g(x) := \begin{pmatrix} -x_1 \\ -x_2 \end{pmatrix} \leq 0, \qquad h(x) := x_1 x_2 = 0.$$

sowie $z \in \mathbb{R}^2$.

In [10]:
```
## Testbeispiel i:

z = np.array([-1,1])
def f_i(x):
    return 0.5*np.linalg.norm(x-z)**2
def gradf_i(x):
    return x-z
def g_i(x):
    return np.array([-x[0], -x[1]])
def Jg_i(x):
    return np.array([[-1, 0], [0, -1]])
def h_i(x):
    return 0
def Jh_i(x):
    return np.array([0,0])
```

In [ ]:

In [ ]:

In [3]:
```
# Testbeispiel ii:

n = 4
def f_iii(x):
    return -x[0]
def gradf_iii(x):
    g = np.zeros(n)
    g[0]=-1
    return g
def h_iii(x):
    return np.array([np.sum(x), np.sum([xx**3 for xx in x])-1])
def Jh_iii(x):
    return np.vstack([np.ones(n), [3*xx**2 for xx in x]])
def g_iii(x):
    return np.array([0])
def Jg_iii(x):
    return np.array([0])
```

In [ ]:

In [5]:
```
# Testbeispiel iii:

z = np.array([-1,-1])
def f_iii(x):
    return 0.5*np.linalg.norm(x-z)**2
def gradf_iii(x):
    return x-z
def g_iii(x):
    return np.array([-x[0], -x[1]])
def Jg_iii(x):
    return np.array([[-1, 0], [0, -1]])
def h_iii(x):
    return np.array([x[0]*x[1]])
def Jh_iii(x):
    return np.array([[x[1],x[0]]])
```

In [ ]:

In [ ]: