# Blatt 07 - Praktische Optimierung - Adrian Lentz, Robert

Lösungen und Erklärungen für Blatt 07.

\newline

Adrian Lentz - Matrikelnummer: 258882

\newline

Robert Schönewald - Matrikelnummer: 188252

## Aufgabe 7.3

In [1]:
```
import numpy as np
import timeit
import matplotlib.pyplot as plt
import scipy
import statsmodels.api as sm
import statsmodels.distributions.empirical_distribution as edf
from scipy.stats import multivariate_normal
```

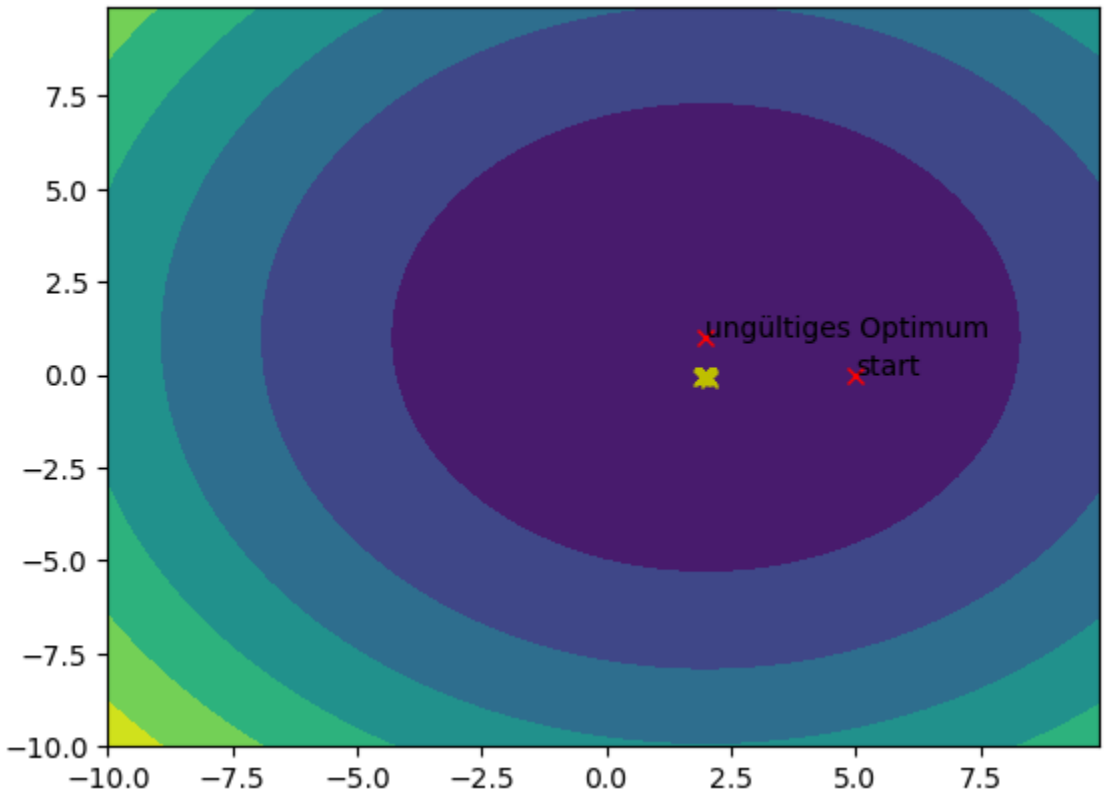
In [2]:
```
#Funktion definieren
def f(x):
    return (x[0]-2)**2 + (x[1]-1)**2
def g1(x):                              #Nebenbedingung 1
    return x[0]**2 - 2*x[0] + x[1]
def g2(x):                              #Nebenbedingung 2
    return -x[0]+x[1]+2
```

In [3]:
```
def seqstraf(f,g1,g2,x0,z1,gamma):
    def T(x):
        return f(x)+z1*(np.max([0,g1(x)]) + np.max([0,g2(x)]))
    #x0=scipy.optimize.minimize(f,x0).x
    while g1(x0)>0 or g2(x0)>0:
        x0=scipy.optimize.minimize(T,x0).x
        z1=gamma*z1
    return x0
```

In [18]:
```
ergebnisse=[]
zlist={0.1 , 0.2 , 0.3 , 0.4 , 0.5 , 0.6 , 0.7 , 0.8 , 0.9 , 1}
gammalist={1.1 , 1.2 , 1.3 , 1.4 , 1.5 , 1.6 , 1.7 , 1.8 , 1.9 , 2}
for gamma in gammalist:
    for z in zlist:
        x0=seqstraf(f,g1,g2,[0,5],z,gamma)
        ergebnisse.append([x0,f(x0),gamma,z])
```

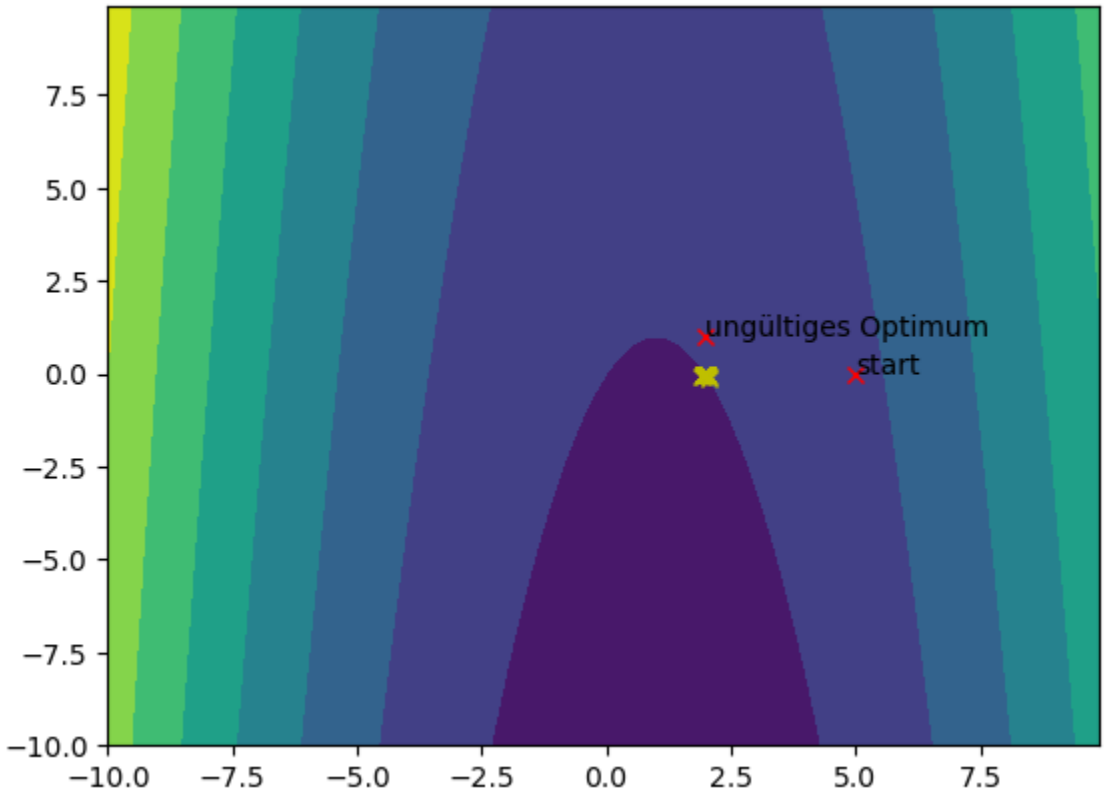
In [19]:
```
x, y = np.mgrid[-10:10:.1, -10:10:.1]
x0=scipy.optimize.minimize(f,[0,5]).x
pos = [x, y]
fig2 = plt.figure()
ax2 = fig2.add_subplot(111)
ax2.contourf(x, y, f(pos))
ax2.text(5,0,"start")
ax2.plot(5,0,"rx")
ax2.text(x0[0],x0[1],"ungültiges Optimum")
ax2.plot(x0[0],x0[1],"rx")
for x in ergebnisse:
```

```
    ax2.plot(x[0][0],x[0][1],"yx")
```

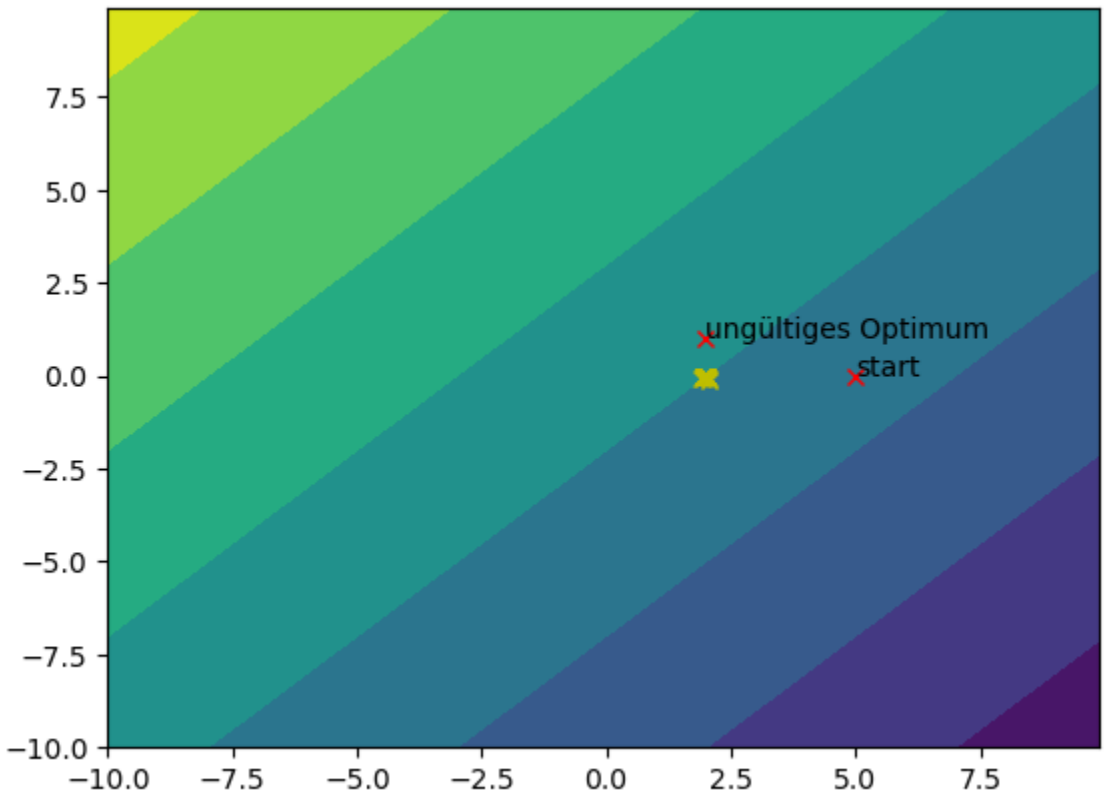



In [20]:

```
x, y = np.mgrid[-10:10:.1, -10:10:.1]
pos = [x, y]
fig2 = plt.figure()
ax2 = fig2.add_subplot(111)
ax2.contourf(x, y, g1(pos))
ax2.text(5,0,"start")
ax2.plot(5,0,"rx")
ax2.text(x0[0],x0[1],"ungültiges Optimum")
ax2.plot(x0[0],x0[1],"rx")
for x in ergebnisse:
    ax2.plot(x[0][0],x[0][1],"yx")
```




In [21]:
```
x, y = np.mgrid[-10:10:.1, -10:10:.1]
pos = [x, y]
fig2 = plt.figure()
ax2 = fig2.add_subplot(111)
ax2.contourf(x, y, g2(pos))
ax2.text(5,0,"start")
ax2.plot(5,0,"rx")
ax2.text(x0[0],x0[1],"ungültiges Optimum")
ax2.plot(x0[0],x0[1],"rx")
for x in ergebnisse:
    ax2.plot(x[0][0],x[0][1],"yx")
```



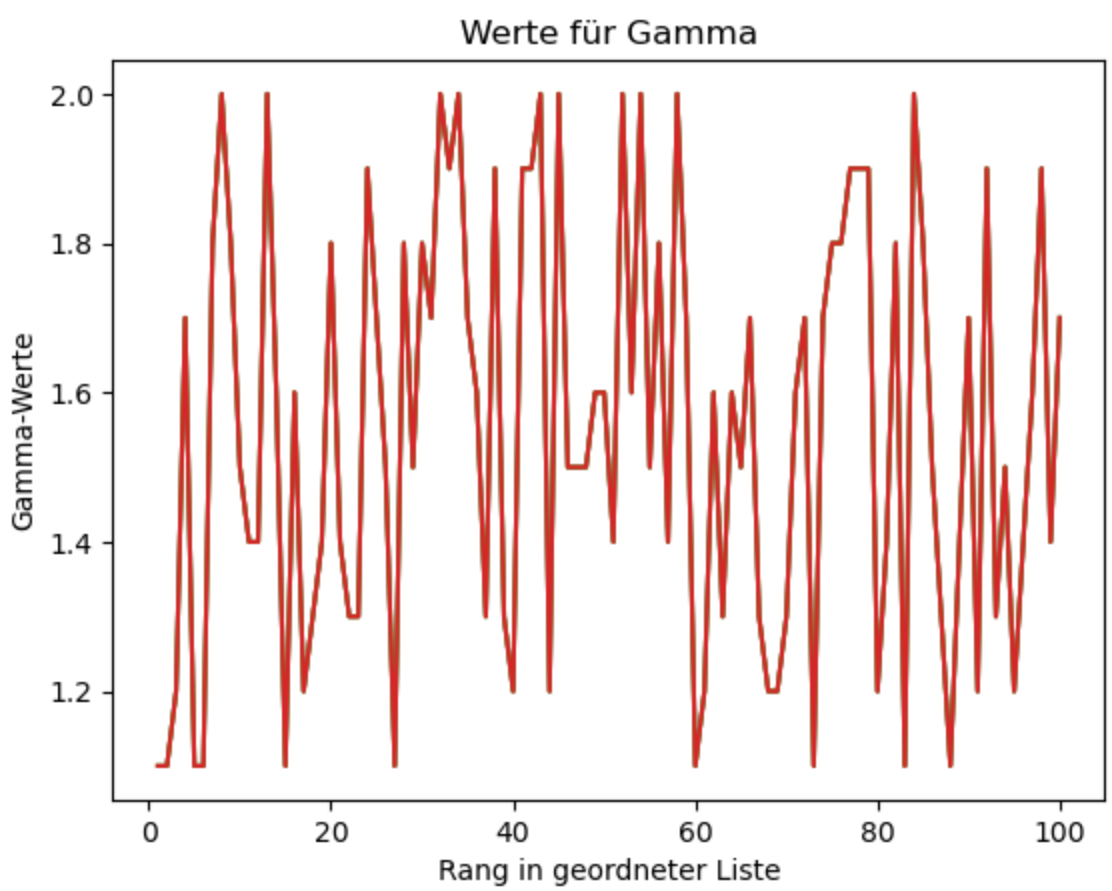
In [24]:
```
ergebnisse.sort(key=lambda x: x[1])
ergebnisse
```

Out[24]:
```
[[array([ 1.99999999e+00, -1.00736977e-08]), 1.0000000201473953, 1.1, 0.4],
 [array([ 1.99999996e+00, -3.53383355e-08]), 1.0000000706766738, 1.1, 0.1],
 [array([ 1.99999976e+00, -2.41409214e-07]), 1.0000004828185438, 1.2, 0.5],
 [array([ 1.99999962e+00, -3.84122652e-07]), 1.0000007682455936, 1.7, 0.2],
 [array([ 1.99999907e+00, -9.28535402e-07]), 1.0000018570725273, 1.1, 0.8],
 [array([ 1.99999872e+00, -1.28502855e-06]), 1.0000025700604092, 1.1, 0.2],
 [array([ 1.99999698e+00, -3.02871753e-06]), 1.0000060574533594, 1.8, 0.8],
 [array([ 1.99999636e+00, -3.64221030e-06]), 1.0000072844471204, 2, 0.9],
 [array([ 1.99999526e+00, -4.74391587e-06]), 1.000009487876683, 1.8, 0.6],
 [array([ 1.99999497e+00, -5.02847625e-06]), 1.0000100570030543, 1.5, 0.2],
 [array([ 1.99999350e+00, -6.50951331e-06]), 1.0000130191113117, 1.4, 0.4],
 [array([ 1.99999146e+00, -8.54345130e-06]), 1.000017087048573, 1.4, 0.2],
 [array([ 1.99998436e+00, -1.56463261e-05]), 1.0000312931415014, 2, 0.8],
 [array([ 1.99997684e+00, -2.31650045e-05]), 1.0000463310819652, 1.6, 0.6],
 [array([ 1.99996570e+00, -3.42989659e-05]), 1.0000686002844985, 1.1, 0.5],
 [array([ 1.99995991e+00, -4.00970440e-05]), 1.0000801973030669, 1.6, 0.5],
 [array([ 1.99994354e+00, -5.64593722e-05]), 1.0001129251196044, 1.2, 0.3],
 [array([ 1.99984823e+00, -1.51770105e-04]), 1.0003035862772136, 1.3, 0.4],
 [array([ 1.99984003e+00, -1.59969083e-04]), 1.0003199893461445, 1.4, 0.5],
 [array([ 2.00007483e+00, -1.84862761e-04]), 1.0003697652974, 1.8, 0.2],
 [array([ 1.99981432e+00, -1.85688803e-04]), 1.000371446563529, 1.4, 0.9],
 [array([ 2.00011633e+00, -2.32681704e-04]), 1.0004654310810377, 1.3, 0.8],
 [array([ 1.99974436e+00, -2.55640072e-04]), 1.0005114108472868, 1.3, 0.9],
 [array([ 2.00012730e+00, -2.73494689e-04]), 1.0005470803814422, 1.9, 0.8],
 [array([ 1.99970795e+00, -2.92055423e-04]), 1.0005842814338575, 1.7, 1],
 [array([ 1.99965271e+00, -3.47299973e-04]), 1.0006948411754277, 1.5, 0.3],
 [array([ 1.99957519e+00, -4.24813979e-04]), 1.0008499888909717, 1.1, 0.6],
 [array([ 1.99952677e+00, -4.73233292e-04]), 1.0009469144813523, 1.8, 0.5],
 [array([ 1.99952323e+00, -4.76771306e-04]), 1.0009539972339734, 1.5, 0.9],
 [array([ 1.9994403e+00, -5.5970469e-04]), 1.0011200359113197, 1.8, 1],
 [array([ 1.99933163e+00, -6.68380950e-04]), 1.0013376553573374, 1.7, 0.3],
 [array([ 1.99934786e+00, -6.98348817e-04]), 1.0013976106174336, 2, 0.1],
 [array([ 1.99927241e+00, -7.27593415e-04]), 1.0014562456052665, 1.9, 0.1],
 [array([ 2.00032508e+00, -7.57551324e-04]), 1.0015157822113425, 2, 0.6],
 [array([ 2.00034246e+00, -7.90918506e-04]), 1.0015825798456612, 1.7, 0.8],
 [array([ 1.99911368e+00, -8.86325043e-04]), 1.0017742212288157, 1.6, 0.7],
 [array([ 1.99905472e+00, -9.45286693e-04]), 1.0018923605171632, 1.3, 0.3],
 [array([ 1.99902642e+00, -9.73578911e-04]), 1.0019490535291435, 1.9, 0.9],
 [array([ 1.99900201e+00, -9.97988825e-04]), 1.00199796960921, 1.3, 0.7],
 [array([ 1.99891558e+00, -1.10573058e-03]), 1.0022138597765546, 1.2, 0.7],
 [array([ 1.99869902e+00, -1.30098399e-03]), 1.0026053530824994, 1.9, 0.3],
 [array([ 1.99835250e+00, -1.64750437e-03]), 1.0033004372722496, 1.9, 0.2],
 [array([ 1.99829988e+00, -1.70012203e-03]), 1.0034060248943397, 2, 0.3],
 [array([ 1.99818217e+00, -1.81782886e-03]), 1.0036422667167528, 1.2, 0.4],
 [array([ 2.00075457, -0.00200946]), 1.0040235254253203, 2, 0.4],
 [array([ 1.99790913, -0.00209088]), 1.0041904961189503, 1.5, 0.1],
 [array([ 1.99759995, -0.00240005]), 1.0048116172967079, 1.5, 0.4],
 [array([ 1.99733753, -0.00266248]), 1.0053391284328148, 1.5, 1],
 [array([ 1.99728065, -0.00271935]), 1.0054534953833016, 1.6, 0.2],
 [array([ 2.00162745, -0.00355957]), 1.0071344687414054, 1.6, 0.3],
 [array([ 1.99632781, -0.00367219]), 1.0073713585500466, 1.4, 0.1],
 [array([ 2.00190564, -0.00397859]), 1.0079766371520735, 2, 1],
 [array([ 1.99578649, -0.00421352]), 1.008462543178847, 1.6, 0.8],
 [array([ 1.99570777, -0.00429223]), 1.0086213019868981, 2, 0.7],
 [array([ 1.99541377, -0.00458623]), 1.0092145353697735, 1.5, 0.5],
 [array([ 1.99528485, -0.00471516]), 1.0094747762401322, 1.8, 0.4],
 [array([ 1.99446015, -0.00553985]), 1.011141087490789, 1.4, 0.7],
 [array([ 1.99427586, -0.00572414]), 1.0115138093157443, 2, 0.2],
 [array([ 1.99353949, -0.00646051]), 1.0130044944011949, 1.7, 0.5],
 [array([ 1.99328011, -0.0067199 ]), 1.0135301043148242, 1.1, 0.7],
 [array([ 1.99272193, -0.00727807]), 1.0146620818171828, 1.2, 0.6],
 [array([ 1.99255945, -0.00744056]), 1.014991837852075, 1.6, 1],
```

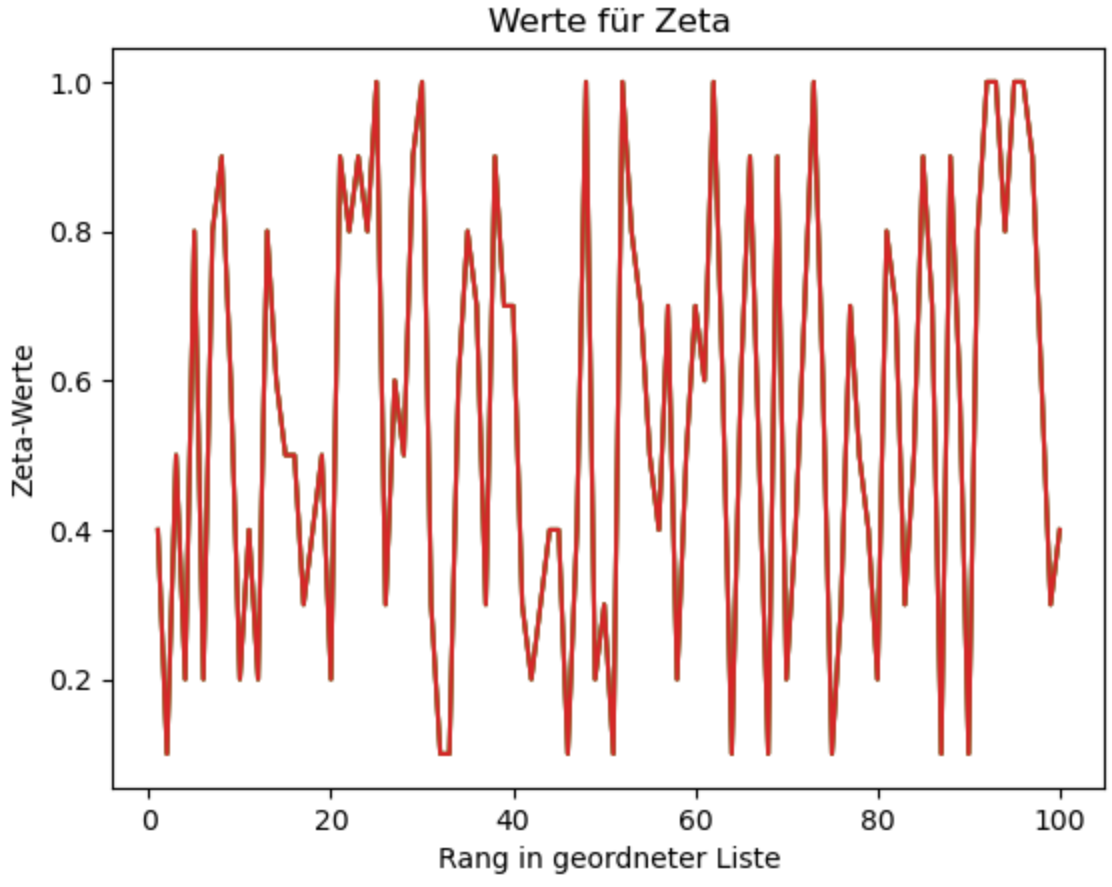
```
[array([ 2.00343391, -0.00774604]), 1.0155638675249807, 1.3, 0.6],
[array([ 1.99157162, -0.00842838]), 1.0169988407667816, 1.6, 0.1],
[array([ 1.99146205, -0.00853795]), 1.01722170324762, 1.5, 0.6],
[array([ 2.00430242, -0.00876925]), 1.0176339081877754, 1.7, 0.9],
[array([ 2.00473863, -0.00969202]), 1.0195004295196177, 1.3, 0.5],
[array([ 1.98879994, -0.01120007]), 1.0226510141474243, 1.2, 0.1],
[array([ 1.98829247, -0.01170753]), 1.0236891921291182, 1.2, 0.9],
[array([ 1.98815581, -0.01184419]), 1.0239689597481512, 1.3, 0.2],
[array([ 1.98777061, -0.01222939]), 1.0247578954119887, 1.6, 0.4],
[array([ 1.9900245 , -0.01298536]), 1.0262388540671448, 1.7, 0.7],
[array([ 2.00631548, -0.01337918]), 1.026977246491478, 1.1, 1],
[array([ 1.9866098, -0.0133902]), 1.0271389943835936, 1.7, 0.6],
[array([ 1.98533711, -0.01466289]), 1.0297557788018885, 1.8, 0.1],
[array([ 1.98292512, -0.01707488]), 1.034732872338242, 1.8, 0.3],
[array([ 1.98113675, -0.01886325]), 1.038438153406618, 1.9, 0.7],
[array([ 1.97601109, -0.02398891]), 1.0491287616575402, 1.9, 0.5],
[array([ 1.97515733, -0.02484267]), 1.0509196501458056, 1.9, 0.4],
[array([ 1.974469, -0.025531]), 1.0523656591631645, 1.2, 0.2],
[array([ 1.97305721, -0.02694279]), 1.05533740390435, 1.4, 0.8],
[array([ 1.9722964, -0.0277036]), 1.05694218731795, 1.8, 0.7],
[array([ 1.97090109, -0.02909892]), 1.059891326622797, 1.1, 0.3],
[array([ 2.01444692, -0.02990208]), 1.0609069988372954, 2, 0.5],
[array([ 1.96872655, -0.03127345]), 1.0645029660418543, 1.8, 0.9],
[array([ 1.96566845, -0.03433155]), 1.0710204129472536, 1.5, 0.7],
[array([ 1.96281771, -0.03718229]), 1.0771296233201737, 1.3, 0.1],
[array([ 1.96275763, -0.03724237]), 1.0772587387062518, 1.1, 0.9],
[array([ 2.01865091, -0.03873169]), 1.0793113816244284, 1.4, 0.6],
[array([ 1.96152569, -0.03847432]), 1.0799091779804593, 1.7, 0.1],
[array([ 2.0182824 , -0.03923416]), 1.0803418801201785, 1.2, 0.8],
[array([ 2.01943519, -0.03946535]), 1.0808659407556234, 1.9, 1],
[array([ 1.95463728, -0.04536273]), 1.0948410153689667, 1.3, 1],
[array([ 2.02295921, -0.04741065]), 1.0975961963733938, 1.5, 0.8],
[array([ 1.95239933, -0.04760068]), 1.0997330028613428, 1.2, 1],
[array([ 1.95203546, -0.04796455]), 1.1005302860297437, 1.4, 1],
[array([ 1.92327504, -0.07672496]), 1.1652233702298158, 1.6, 0.9],
[array([ 2.04821798, -0.10051533]), 1.2134589724499516, 1.9, 0.6],
[array([ 1.90162566, -0.09837434]), 1.2161036988397553, 1.4, 0.3],
[array([ 2.06838372, -0.14404263]), 1.3135098785734094, 1.7, 0.4]]
```

Man erkennt an der oben sortierten Liste, dass der beste Wert von f der erreicht wird 1.0000000201473953 ist mit gamma=1.1 und zeta=0.4, während die schlechteste Konfiguration gamma=1.7 und zeta=0.4 ist mit einem Wert von 1.3135098785734094. Insbesondere fällt auf, dass beide einen zeta-Wert von 0.4 haben. Zusätzlich sind beinahe alle Werte nah aneinander, wie im Diagramm sichtbar.

In [35]:
```
plt.xlabel("Rang in geordneter Liste")
plt.ylabel("Gamma-Werte")
plt.title("Werte für Gamma")
for i in range(len(ergebnisse[0])):
    plt.plot(list(range(1,101)),[pt[2] for pt in ergebnisse])
plt.show()
```




In [34]:
```
plt.xlabel("Rang in geordneter Liste")
plt.ylabel("Zeta-Werte")
plt.title("Werte für Zeta")
for i in range(len(ergebnisse[0])):
    plt.plot(list(range(1,101)),[pt[3] for pt in ergebnisse])
plt.show()
```



Hier sieht man noch einmal deutlicher, dass die Parameter kaum eine Auswirkung auf die Qualität der Berechnung haben.

In [ ]: