

Blatt 1

Aufgabe 1a), b) und c)

Es werden die zwei Funktionen a)

$$f(x) = \left(x^3 + \frac{1}{3}\right) - \left(x^3 - \frac{1}{3}\right)$$

und b)

$$g(x) = \frac{\left(3 + \frac{x^3}{3}\right) - \left(3 - \frac{x^3}{3}\right)}{x^3}$$

empirisch untersucht. Dabei soll festgestellt werden, für welche Bereiche von x das numerische Ergebnis vom algebraischen um nicht mehr als 1% abweicht und in welchen Bereichen das Ergebnis gleich null ist. Anschließend gibt es eine graphische Auswertung.

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:

```
def f(x): #Erzeugen der ersten Funktion
    return (x**3 + 1/3) - (x**3 - 1/3)
```

In [3]:

```
def g(x): #Erzeugen der zweiten Funktion
    return ((3+(x**3)/3) - (3 - (x**3)/3))/x**3
```

In [4]:

```
def interval(a): #Algorithmus zur Bestimmung und Ausgabe der Intervalle für Zahlen aus dem Bereich der natürlichen Zahlen
    if len(a) != 0:
        new = np.arange(0, len(a), 1)
        index1 = (a[new-1]+1 != a[new]) #Durch diese Bedingung werden alle Startwerte eines Intervalls bestimmt
        index2= np.append(index1[1:], index1[0])#Durch die Indexverschiebung um 1 nach links, wird das Intervall der Endwerte bestimmt
        a1= a[index1]
        a2= a[index2]
        for i in range(len(a1)):
            if (a1[i]==a2[i]): #Wenn der Start und Endwert derselbe ist, reicht es, wenn man ihn einmal printed, die Liste ist auch so schon zu lang
                print("(", a1[i], ")", "; ", sep="", end="")
            else:
                print("(", a1[i], ",", a2[i], ")", "; ", sep="", end="")
        print("")
    else: print("Leeres Array.")
```

In [5]:

```
def makeplot(func, fig, c): #Funktion, Nummer der figure und Potenz des Defini-
tionsbereichs kann eingegeben werden
    x= np.linspace(1, 10**c, 10**c) #Definitionsereich wird definiert
    y= func(x) #y-Werte werden definiert
    y0= 2/3* np.ones(len(x)) #"exakter Wert" 2/3
    x1= x[(y==2/3)] #numerische Werte mit "keiner" Abweichung
    x2= x[(y>((2/3)+0.01)) | (y<(2/3 - 0.01))] #numerische Abweichung unter ei-
nem Prozent
    x3= x[(y<((2/3)+0.01)) & (y>(2/3 - 0.01)) & (y!=2/3)] #numerische Abweichu-
ng über einem Prozent
    x4= x[y==0] #numerischer Wert gleich Null
    plt.figure(fig, figsize=(8,6))
    plt.xscale("log") #Logarithmierte Skala
    plt.plot(x, y0, "b", label="Tatsächlicher Wert")
    plt.plot(x1, func(x1), "k.", label="Numerische Abweichung von 0")
    plt.plot(x2, func(x2), "g.", label="Abweichung größer als 1%", linestyle="
None")
    plt.plot(x3, func(x3), "r.", label="Abweichung weniger als 1%", linestyle=
"None")
    plt.plot(x4, func(x4), "y.", label="Numerischer Wert ist 0", linestyle="No
ne")
    plt.xlabel(r'$x$')
    plt.ylabel(r'$y$')
    plt.legend()
    #Ausgabe der Werte mittels der Interval-Funktion
    print("Tatsächlicher Wert für Formel", fig,": ", end="")
    interval(x1)
    print("\n\n")
    print("Abweichung größer als 1% für Formel", fig,": ", end="")
    interval(x2)
    print("\n\n")
    print("Abweichung weniger als 1% für Formel", fig, ": ", end="")
    interval(x3)
    print("\n\n")
    print("Numerischer Wert gleich Null für Formel", fig, ": ", end="")
    interval(x4)
    print("\n\n")
```

In [6]:

```
makeplot(f, 1, 6) #Ausführung für Funktion f
makeplot(g, 2, 3) #Ausführung für Funktion g, Potenz kleiner gewählt, da die I-
ntervalle sonst die ganze Datei überschwemmen.
```

Tatsächlicher Wert für Formel 1 : Leeres Array.

Abweichung größer als 1% für Formel 1 : (41286.0,1000000.0);

Abweichung weniger als 1% für Formel 1 : (1.0,41285.0);

Numerischer Wert gleich Null für Formel 1 : (165141.0,1000000.0);

Tatsächlicher Wert für Formel 2 : (3.0,10.0); (12.0); (14.0,16.0); (18.0); (20.0,21.0); (23.0,25.0); (27.0,28.0); (30.0); (32.0,33.0); (35.0,37.0); (39.0,40.0); (42.0,43.0); (45.0,48.0); (50.0,51.0); (53.0,54.0); (56.0,57.0); (60.0,61.0); (63.0,64.0); (66.0,67.0); (69.0,70.0); (72.0,75.0); (77.0,78.0); (80.0,81.0); (83.0,84.0); (86.0,87.0); (89.0,90.0); (92.0,94.0); (96.0,97.0); (99.0,100.0); (102.0,103.0); (105.0,106.0); (108.0,109.0); (111.0,112.0); (114.0,115.0); (117.0); (119.0,120.0); (122.0,123.0); (125.0,126.0); (128.0,129.0); (131.0,132.0); (134.0,135.0); (137.0,138.0); (140.0,141.0); (143.0,144.0); (146.0,148.0); (150.0,151.0); (153.0,154.0); (156.0,157.0); (159.0,160.0); (162.0,163.0); (165.0,166.0); (168.0,169.0); (171.0,172.0); (174.0,175.0); (177.0,178.0); (180.0,181.0); (183.0,186.0); (188.0,189.0); (191.0,192.0); (194.0,195.0); (197.0,198.0); (200.0,201.0); (203.0,204.0); (206.0,207.0); (209.0,210.0); (212.0,213.0); (215.0,216.0); (218.0,219.0); (221.0,222.0); (224.0,225.0); (227.0,228.0); (230.0,231.0); (234.0,235.0); (237.0,238.0); (240.0,241.0); (243.0,244.0); (246.0,247.0); (249.0,250.0); (252.0,253.0); (255.0,256.0); (258.0,259.0); (261.0,262.0); (264.0,265.0); (267.0,268.0); (270.0,271.0); (273.0,274.0); (276.0,277.0); (279.0,280.0); (282.0,283.0); (285.0,286.0); (288.0,289.0); (291.0,292.0); (294.0); (296.0,297.0); (299.0,300.0); (302.0,303.0); (305.0,306.0); (308.0,309.0); (311.0,312.0); (314.0,315.0); (317.0,318.0); (320.0,321.0); (323.0,324.0); (326.0,327.0); (329.0,330.0); (332.0,333.0); (335.0,336.0); (338.0,339.0); (341.0,342.0); (344.0,345.0); (347.0,348.0); (350.0,351.0); (353.0,354.0); (356.0,357.0); (359.0,360.0); (362.0,363.0); (365.0,366.0); (368.0,370.0); (372.0,373.0); (375.0,376.0); (378.0,379.0); (381.0,382.0); (384.0,385.0); (387.0,388.0); (390.0,391.0); (393.0,394.0); (396.0,397.0); (399.0,400.0); (402.0,403.0); (405.0,406.0); (408.0,409.0); (411.0,412.0); (414.0,415.0); (417.0,418.0); (420.0,421.0); (423.0,424.0); (426.0,427.0); (429.0,430.0); (432.0,433.0); (435.0,436.0); (438.0,439.0); (441.0,442.0); (444.0,445.0); (447.0,448.0); (450.0,451.0); (453.0,454.0); (456.0,457.0); (459.0,460.0); (462.0,463.0); (465.0); (467.0,468.0); (470.0,471.0); (473.0,474.0); (476.0,477.0); (479.0,480.0); (482.0,483.0); (485.0,486.0); (488.0,489.0); (491.0,492.0); (494.0,495.0); (497.0,498.0); (500.0,501.0); (503.0,504.0); (506.0,507.0); (509.0,510.0); (512.0,513.0); (515.0,516.0); (518.0,519.0); (521.0,522.0); (524.0,525.0); (527.0,528.0); (530.0,531.0); (533.0,534.0); (536.0,537.0); (539.0,540.0); (542.0,543.0); (545.0,546.0); (548.0,549.0); (551.0,552.0); (554.0,555.0); (557.0,558.0); (560.0,561.0); (563.0,564.0); (566.0,567.0); (569.0,570.0); (572.0,573.0); (575.0,576.0); (578.0,579.0); (581.0,582.0); (584.0,585.0); (588.0,589.0); (591.0,592.0); (594.0,595.0); (597.0,598.0); (600.0,601.0); (603.0,604.0); (606.0,607.0); (609.0,610.0); (612.0,613.0); (615.0,616.0); (618.0,619.0); (621.0,622.0); (624.0,625.0); (627.0,628.0); (630.0,631.0); (633.0,634.0); (636.0,637.0); (639.0,640.0); (642.0,643.0); (645.0,646.0); (648.0,649.0); (651.0,652.0); (654.0,655.0); (657.0,658.0); (660.0,661.0); (663.0,664.0); (666.0,667.0); (669.0,670.0); (672.0,673.0); (675.0,676.0); (678.0,679.0); (681.0,682.0); (684.0,685.0); (687.0,688.0); (690.0,691.0); (693.0,694.0); (696.0,697.0); (699.0,700.0); (702.0

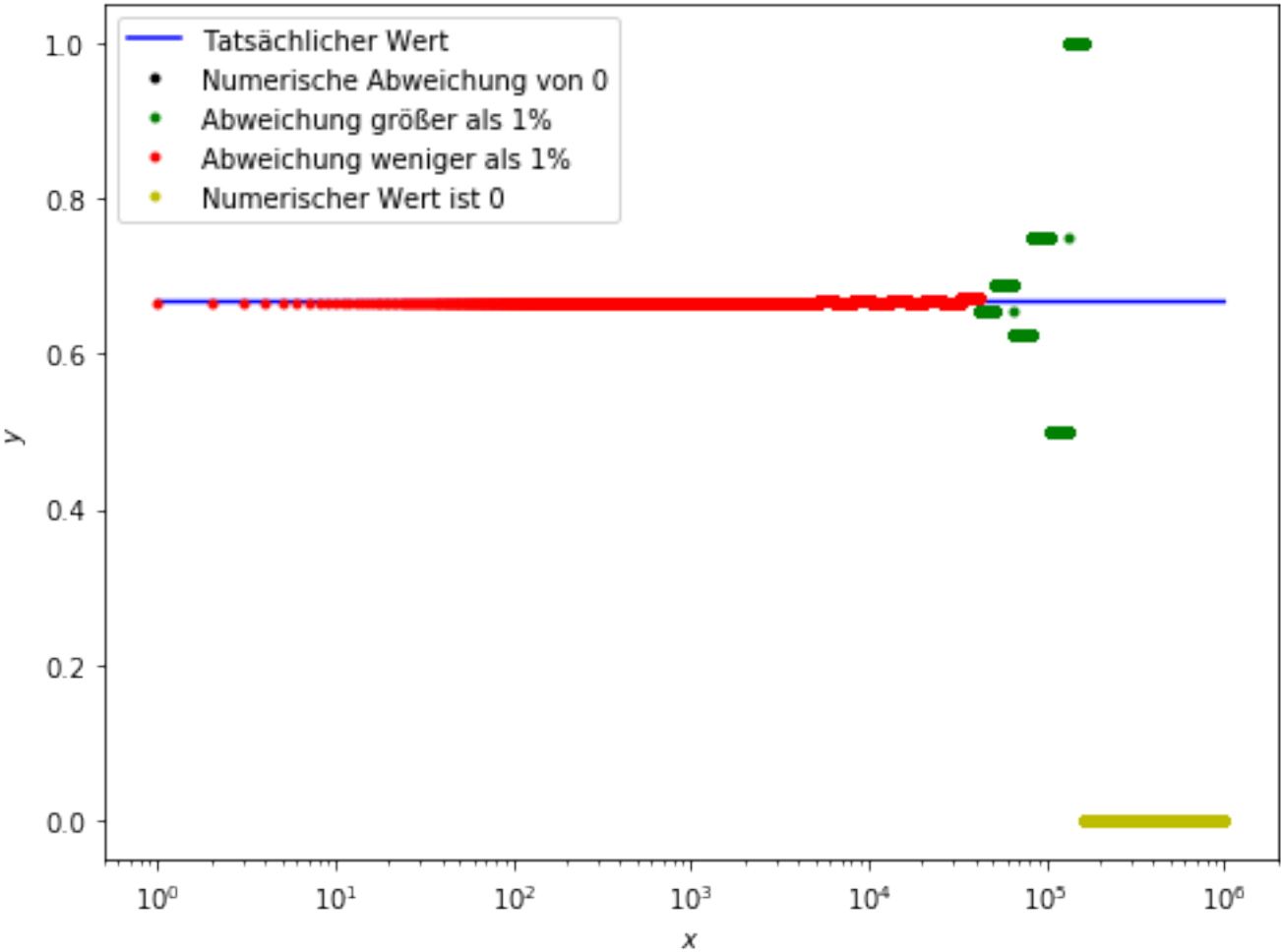
,703.0); (705.0,706.0); (708.0,709.0); (711.0,712.0); (714.0,715.0); (717.0,718.0); (720.0,721.0); (723.0,724.0); (726.0,727.0); (729.0,730.0); (732.0,733.0); (735.0,736.0); (738.0); (740.0,741.0); (743.0,744.0); (746.0,747.0); (749.0,750.0); (752.0,753.0); (755.0,756.0); (758.0,759.0); (761.0,762.0); (764.0,765.0); (767.0,768.0); (770.0,771.0); (773.0,774.0); (776.0,777.0); (779.0,780.0); (782.0,783.0); (785.0,786.0); (788.0,789.0); (791.0,792.0); (794.0,795.0); (797.0,798.0); (800.0,801.0); (803.0,804.0); (806.0,807.0); (809.0,810.0); (812.0,813.0); (815.0,816.0); (818.0,819.0); (821.0,822.0); (824.0,825.0); (827.0,828.0); (830.0,831.0); (833.0,834.0); (836.0,837.0); (839.0,840.0); (842.0,843.0); (845.0,846.0); (848.0,849.0); (851.0,852.0); (854.0,855.0); (857.0,858.0); (860.0,861.0); (863.0,864.0); (866.0,867.0); (869.0,870.0); (872.0,873.0); (875.0,876.0); (878.0,879.0); (881.0,882.0); (884.0,885.0); (887.0,888.0); (890.0,891.0); (893.0,894.0); (896.0,897.0); (899.0,900.0); (902.0,903.0); (905.0,906.0); (908.0,909.0); (911.0,912.0); (914.0,915.0); (917.0,918.0); (920.0,921.0); (923.0,924.0); (926.0,927.0); (929.0,931.0); (933.0,934.0); (936.0,937.0); (939.0,940.0); (942.0,943.0); (945.0,946.0); (948.0,949.0); (951.0,952.0); (954.0,955.0); (957.0,958.0); (960.0,961.0); (963.0,964.0); (966.0,967.0); (969.0,970.0); (972.0,973.0); (975.0,976.0); (978.0,979.0); (981.0,982.0); (984.0,985.0); (987.0,988.0); (990.0,991.0); (993.0,994.0); (996.0,997.0); (999.0,1000.0);

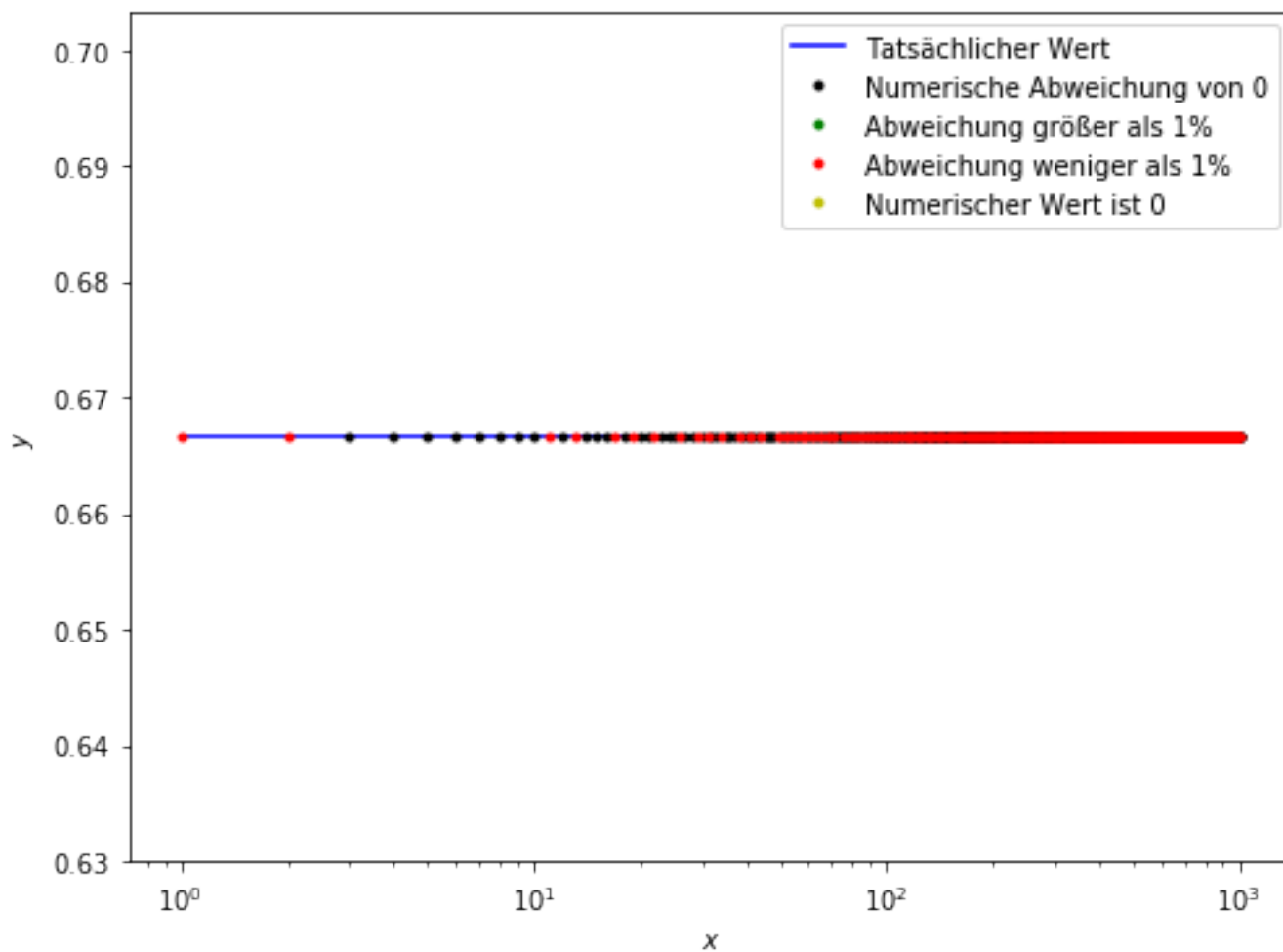
Abweichung größer als 1% für Formel 2 : Leeres Array.

Abweichung weniger als 1% für Formel 2 : (1.0,2.0); (11.0); (13.0); (17.0); (19.0); (22.0); (26.0); (29.0); (31.0); (34.0); (38.0); (41.0); (44.0); (49.0); (52.0); (55.0); (58.0,59.0); (62.0); (65.0); (68.0); (71.0); (76.0); (79.0); (82.0); (85.0); (88.0); (91.0); (95.0); (98.0); (101.0); (104.0); (107.0); (110.0); (113.0); (116.0); (118.0); (121.0); (124.0); (127.0); (130.0); (133.0); (136.0); (139.0); (142.0); (145.0); (149.0); (152.0); (155.0); (158.0); (161.0); (164.0); (167.0); (170.0); (173.0); (176.0); (179.0); (182.0); (187.0); (190.0); (193.0); (196.0); (199.0); (202.0); (205.0); (208.0); (211.0); (214.0); (217.0); (220.0); (223.0); (226.0); (229.0); (232.0,233.0); (236.0); (239.0); (242.0); (245.0); (248.0); (251.0); (254.0); (257.0); (260.0); (263.0); (266.0); (269.0); (272.0); (275.0); (278.0); (281.0); (284.0); (287.0); (290.0); (293.0); (295.0); (298.0); (301.0); (304.0); (307.0); (310.0); (313.0); (316.0); (319.0); (322.0); (325.0); (328.0); (331.0); (334.0); (337.0); (340.0); (343.0); (346.0); (349.0); (352.0); (355.0); (358.0); (361.0); (364.0); (367.0); (371.0); (374.0); (377.0); (380.0); (383.0); (386.0); (389.0); (392.0); (395.0); (398.0); (401.0); (404.0); (407.0); (410.0); (413.0); (416.0); (419.0); (422.0); (425.0); (428.0); (431.0); (434.0); (437.0); (440.0); (443.0); (446.0); (449.0); (452.0); (455.0); (458.0); (461.0); (464.0); (466.0); (469.0); (472.0); (475.0); (478.0); (481.0); (484.0); (487.0); (490.0); (493.0); (496.0); (499.0); (502.0); (505.0); (508.0); (511.0); (514.0); (517.0); (520.0); (523.0); (526.0); (529.0); (532.0); (535.0); (538.0); (541.0); (544.0); (547.0); (550.0); (553.0); (556.0); (559.0); (562.0); (565.0); (568.0); (571.0); (574.0); (577.0); (580.0); (583.0); (586.0,587.0); (590.0); (593.0); (596.0); (599.0)

); (602.0); (605.0); (608.0); (611.0); (614.0); (617.0); (620.0);
(623.0); (626.0); (629.0); (632.0); (635.0); (638.0); (641.0); (64
4.0); (647.0); (650.0); (653.0); (656.0); (659.0); (662.0); (665.0
); (668.0); (671.0); (674.0); (677.0); (680.0); (683.0); (686.0);
(689.0); (692.0); (695.0); (698.0); (701.0); (704.0); (707.0); (71
0.0); (713.0); (716.0); (719.0); (722.0); (725.0); (728.0); (731.0
); (734.0); (737.0); (739.0); (742.0); (745.0); (748.0); (751.0);
(754.0); (757.0); (760.0); (763.0); (766.0); (769.0); (772.0); (77
5.0); (778.0); (781.0); (784.0); (787.0); (790.0); (793.0); (796.0
); (799.0); (802.0); (805.0); (808.0); (811.0); (814.0); (817.0);
(820.0); (823.0); (826.0); (829.0); (832.0); (835.0); (838.0); (84
1.0); (844.0); (847.0); (850.0); (853.0); (856.0); (859.0); (862.0
); (865.0); (868.0); (871.0); (874.0); (877.0); (880.0); (883.0);
(886.0); (889.0); (892.0); (895.0); (898.0); (901.0); (904.0); (90
7.0); (910.0); (913.0); (916.0); (919.0); (922.0); (925.0); (928.0
); (932.0); (935.0); (938.0); (941.0); (944.0); (947.0); (950.0);
(953.0); (956.0); (959.0); (962.0); (965.0); (968.0); (971.0); (97
4.0); (977.0); (980.0); (983.0); (986.0); (989.0); (992.0); (995.0
); (998.0);

Numerischer Wert gleich Null für Formel 2 : Leeres Array.





Zu 1)

Somit ist zu erkennen, dass die erste Funktion für keinen Wert dem exakten Wert $\frac{2}{3}$ entspricht, für das Intervall zwischen 1 und 41286 bei unter einem Prozent Abweichung liegt und ab diesem Wert über einem Prozent bis zum Ende des Definitionsbereichs. Ab dem Wert 165141 bis zum Ende liegt der Wert über einem Prozent und ist null.

Die zweite Funktion ist deutlich stabiler. Bei ihr hat kein Wert eine Abweichung über einem Prozent. Die Werte unter einem Prozent springen für verschiedene Intervalle zwischen exakt $\frac{2}{3}$ und unter einem Prozent hin und her, wie man an der langen Liste der Intervalle erkennen kann. Der Wert Null ergibt sich nie und allgemein weicht kein Wert im untersuchten Intervall über einen Prozent ab.

Aufgabe 2a)

Es ist ein Term des differentiellen Wirkungsquerschnitts für die Reaktion $e^- e^+ \rightarrow \gamma\gamma$ gegeben und es soll bestimmt werden, ob dieser numerisch stabil ist. Außerdem soll der Bereich von θ bestimmt werden, in dem die Gleichung für $E_e = 50 \text{ GeV}$ numerisch instabil ist.

Der Ausdruck ist numerisch nicht stabil, da es, wenn θ Werte nahe π oder Werte nahe eines Vielfachen von π annimmt, im Nenner zu einer Subtraktion zweier fast gleich großer Zahlen kommt, was immer mit großen Rundungsfehlern behaftet ist.

b)

Das Stabilitätsproblem soll durch eine geeignete analytische Umformung gelöst werden.

Mithilfe der angegebenen Umformungen kann man den Term umschreiben zu:

$$\frac{d\sigma}{d\Omega} = \frac{\alpha^2}{s} \left(\frac{2 + \sin^2(\theta)}{\sin^2(\theta) + \frac{1}{\gamma} \cos^2(\theta)} \right)$$

c)

Es soll gezeigt werden, dass die Stabilitätsprobleme behoben wurden, indem die Gleichungen in den kritischen Intervallen graphisch dargestellt werden.

In [7]:

```
import scipy.constants as const
```

In [8]:

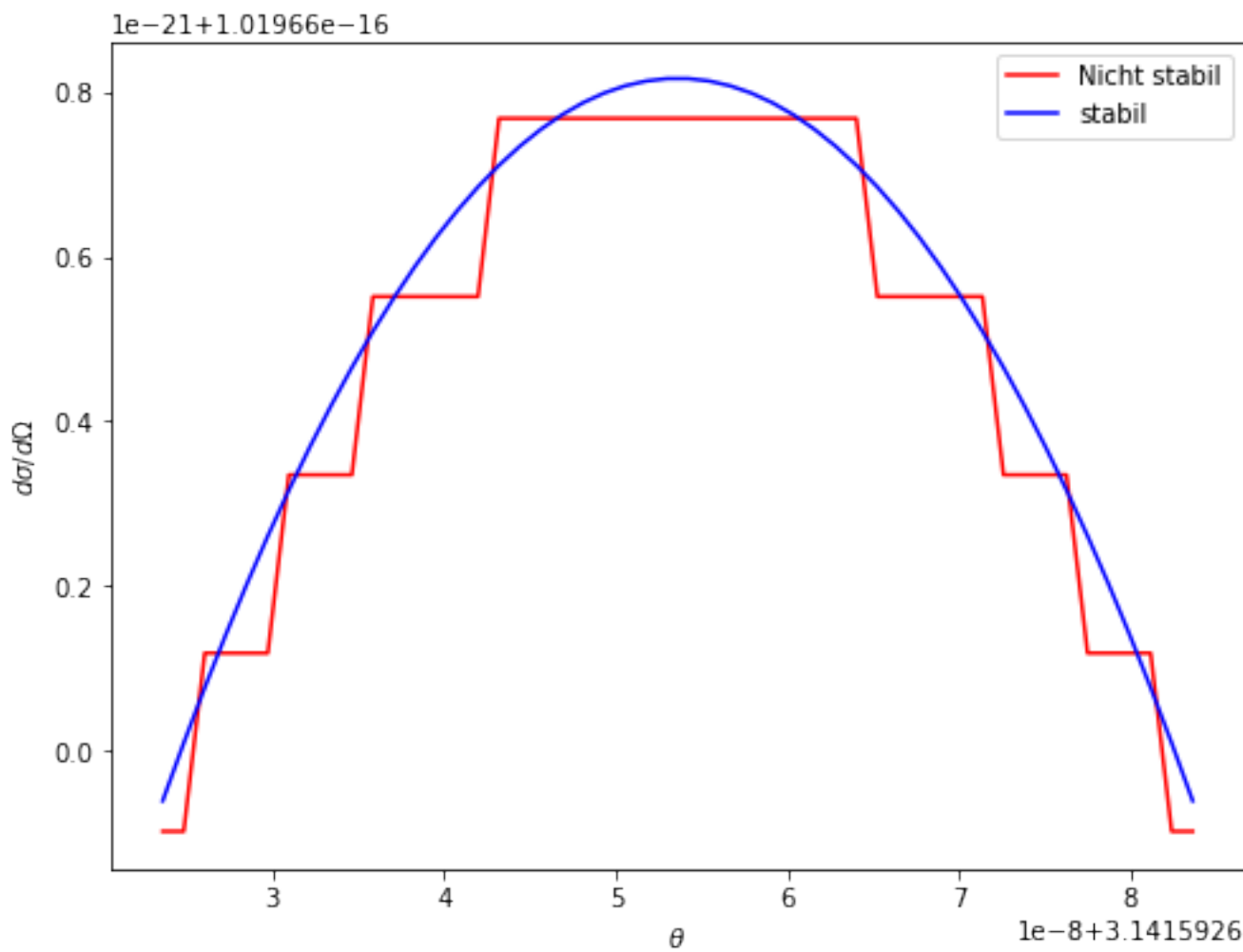
```
#Definition der Konstanten in der Funktion
Ee = 50*1e9
me = 511*1e3
s = (2*Ee)**2
gamma = Ee/me
beta = np.sqrt(1-gamma**(-2))
alpha= const.alpha

def wirkung(theta): #instabile Funktion
    return alpha**2 /s * ((2 + np.sin(theta)**2)/(1-beta**2 *np.cos(theta)**2)
)

def verbessert(theta): #umgeformte, stabile Funktion
    return alpha**2 /s * ((2 + np.sin(theta)**2)/(np.sin(theta)**2+1/(gamma**2
)*np.cos(theta)**2))

x=np.linspace(np.pi-3e-8, np.pi+3e-8) #Definitionsbereich in kleinem Bereich u
m pi gewählt

plt.figure(3, figsize=(8,6))
plt.plot(x, wirkung(x), 'r-', label="Nicht stabil")
plt.plot(x, verbessert(x), 'b-', label="stabil")
plt.xlabel(r'$\theta$')
plt.ylabel(r'$d\sigma/d\Omega$')
plt.legend()
None
```

d)

Die Konditionszahl soll berechnet werden und es soll erklärt werden, wie diese von θ abhängt.

Die Konditionszahl ergibt sich nach einigen Umformungen zu

$$K = \left| \frac{f'(\theta)}{f(\theta)} \right| \theta = \left| \frac{2 \sin(\theta) \cos(\theta) (3m_e^2 - 2E_e^2)}{(\sin(\theta)^2 + 2)(m_e^2 \cos(\theta)^2 + E_e^2 \sin(\theta)^2)} \right| \theta.$$

Für Werte um $\theta = \frac{n}{2}\pi$, $\forall n \in \mathbb{N}$ ist die Konditionierung gut, da dann entweder der cos- oder der sin-Teil null werden.

e)

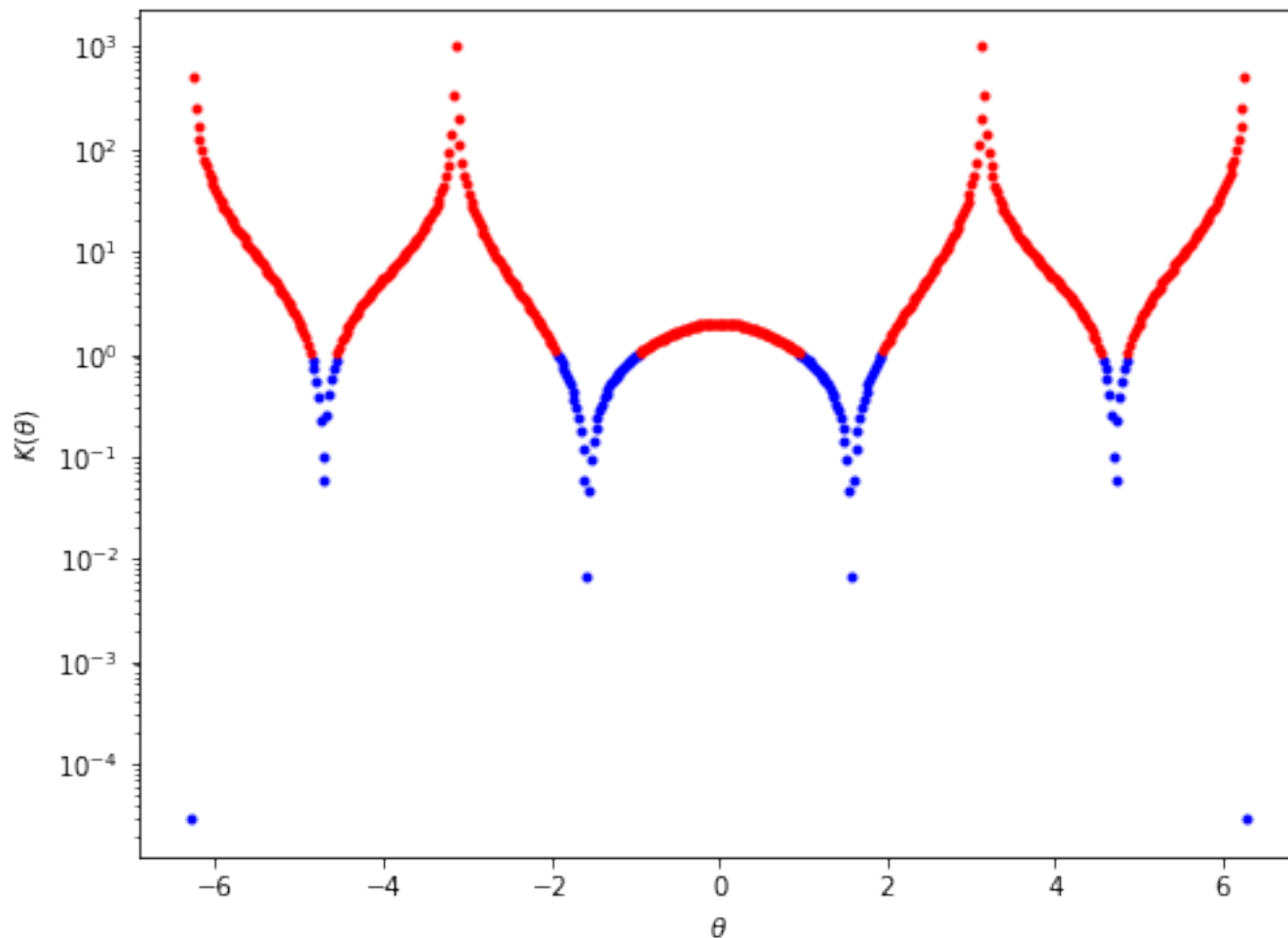
Der Verlauf der Konditionszahl soll als Funktion von θ im Intervall $(0 \leq \theta \leq \pi)$ graphisch dargestellt werden. Außerdem soll erklärt werden, in welchem Bereich das Problem gut und in welchem schlecht konditioniert ist.

In [9]:

```
def K(theta): #Konditionszahl als Funktion von theta
    return np.abs((2*np.sin(theta)* np.cos(theta)*(3*me**2 -2*Ee**2)) / ((np.sin(theta)**2 +2)*(me**2 *np.cos(theta)**2+ Ee**2 * np.sin(theta)**2))*theta)
```

In [10]:

```
theta= np.linspace(-2*np.pi, 2*np.pi, 500) #Definitionsbereich recht klein gew
ählt, damit es nicht so viele Werte in der Ausgabe sind
y= K(theta)
plt.figure(1, figsize=(8,6))
plt.plot(theta[y<1], y[y<1], "b.", label="Gute Konditionierung, Fehlerdämpfung
")
plt.plot(theta[y>1], y[y>1], "r.", label="Schlechte Konditionierung")
plt.xlabel(r'$\theta$')
plt.ylabel(r'$K(\theta)$')
#plt.legend(loc="best")
plt.yscale("log") #logarithmierte y-Skala zur besseren Darstellung der Werte
```



In [11]:

```
print("Gute Konditionierung für die Werte:", end="")# Ausgabe der gut konditio
nierten x-Werte
print(theta[y<1])
print("\n\n")
print("Schlechte Konditionierung für die Werte:", end="") #Ausgabe der schlech
t konditionierten x-Werte
print(theta[y>1])
print("\n\n")
```

```
Gute Konditionierung für die Werte:[-6.28318531 -4.84774818 -4.822
56508 -4.79738197 -4.77219886 -4.74701575
-4.72183265 -4.69664954 -4.67146643 -4.64628332 -4.62110022 -4.59
591711
-4.570734 -1.92650772 -1.90132461 -1.8761415 -1.8509584 -1.82
577529
-1.80059218 -1.77540907 -1.75022597 -1.72504286 -1.69985975 -1.67
467664
```

-1.64949354	-1.62431043	-1.59912732	-1.57394422	-1.54876111	-1.523578
-1.49839489	-1.47321179	-1.44802868	-1.42284557	-1.39766246	-1.37247936
-1.34729625	-1.32211314	-1.29693003	-1.27174693	-1.24656382	-1.22138071
-1.1961976	-1.1710145	-1.14583139	-1.12064828	-1.09546517	-1.07028207
-1.04509896	-1.01991585	-0.99473274	-0.96954964	0.96954964	0.99473274
1.01991585	1.04509896	1.07028207	1.09546517	1.12064828	1.14583139
1.1710145	1.1961976	1.22138071	1.24656382	1.27174693	1.29693003
1.32211314	1.34729625	1.37247936	1.39766246	1.42284557	1.44802868
1.47321179	1.49839489	1.523578	1.54876111	1.57394422	1.59912732
1.62431043	1.64949354	1.67467664	1.69985975	1.72504286	1.75022597
1.77540907	1.80059218	1.82577529	1.8509584	1.8761415	1.90132461
1.92650772	4.570734	4.59591711	4.62110022	4.64628332	4.67146643
4.69664954	4.72183265	4.74701575	4.77219886	4.79738197	4.82256508
4.84774818	6.28318531]				

Schlechte Konditionierung für die Werte:[-6.2580022 -6.23281909 -6.20763598 -6.18245288 -6.15726977 -6.13208666 -6.10690356 -6.08172045 -6.05653734 -6.03135423 -6.00617113 -5.98098802 -5.95580491 -5.9306218 -5.9054387 -5.88025559 -5.85507248 -5.82988937 -5.80470627 -5.77952316 -5.75434005 -5.72915694 -5.70397384 -5.67879073 -5.65360762 -5.62842451 -5.60324141 -5.5780583 -5.55287519 -5.52769208 -5.50250898 -5.47732587 -5.45214276 -5.42695965 -5.40177655 -5.37659344 -5.35141033 -5.32622722 -5.30104412 -5.27586101 -5.2506779 -5.22549479 -5.20031169 -5.17512858 -5.14994547 -5.12476236 -5.09957926 -5.07439615 -5.04921304 -5.02402993 -4.99884683 -4.97366372 -4.94848061 -4.92329751 -4.8981144 -4.87293129 -4.84774818 -4.82256508 -4.79738197 -4.77219886 -4.74701575 -4.72183265 -4.69664954 -4.67146643 -4.64628332 -4.62110022 -4.59591711 -4.570734 -4.54555089 -4.52036779 -4.49518468 -4.47000157 -4.44481846 -4.41963536 -4.39445225 -4.36926914 -4.34408603 -4.31890293 -4.29371982 -4.26853671 -4.2433536 -4.2181705 -4.19298739 -4.16780428 -4.14262117 -4.11743807 -4.09225496 -4.06707185 -4.04188874 -4.01670564 -3.99152253 -3.96633942 -3.94115631 -3.91597321 -3.8907901 -3.86

560699	-3.84042389	-3.81524078	-3.79005767	-3.76487456	-3.73969146	-3.71
450835	-3.68932524	-3.66414213	-3.63895903	-3.61377592	-3.58859281	-3.56
34097	-3.5382266	-3.51304349	-3.48786038	-3.46267727	-3.43749417	-3.41
231106	-3.38712795	-3.36194484	-3.33676174	-3.31157863	-3.28639552	-3.26
121241	-3.23602931	-3.2108462	-3.18566309	-3.16047998	-3.13529688	-3.11
011377	-3.08493066	-3.05974755	-3.03456445	-3.00938134	-2.98419823	-2.95
901512	-2.93383202	-2.90864891	-2.8834658	-2.85828269	-2.83309959	-2.80
791648	-2.78273337	-2.75755027	-2.73236716	-2.70718405	-2.68200094	-2.65
681784	-2.63163473	-2.60645162	-2.58126851	-2.55608541	-2.5309023	-2.50
571919	-2.48053608	-2.45535298	-2.43016987	-2.40498676	-2.37980365	-2.35
462055	-2.32943744	-2.30425433	-2.27907122	-2.25388812	-2.22870501	-2.20
35219	-2.17833879	-2.15315569	-2.12797258	-2.10278947	-2.07760636	-2.05
242326	-2.02724015	-2.00205704	-1.97687393	-1.95169083	-0.94436653	-0.91
918342	-0.89400031	-0.86881721	-0.8436341	-0.81845099	-0.79326788	-0.76
808478	-0.74290167	-0.71771856	-0.69253545	-0.66735235	-0.64216924	-0.61
698613	-0.59180302	-0.56661992	-0.54143681	-0.5162537	-0.4910706	-0.46
588749	-0.44070438	-0.41552127	-0.39033817	-0.36515506	-0.33997195	-0.31
478884	-0.28960574	-0.26442263	-0.23923952	-0.21405641	-0.18887331	-0.16
36902	-0.13850709	-0.11332398	-0.08814088	-0.06295777	-0.03777466	-0.01
259155	0.01259155	0.03777466	0.06295777	0.08814088	0.11332398	0.13
850709	0.1636902	0.18887331	0.21405641	0.23923952	0.26442263	0.28
960574	0.31478884	0.33997195	0.36515506	0.39033817	0.41552127	0.44
070438	0.46588749	0.4910706	0.5162537	0.54143681	0.56661992	0.59
180302	0.61698613	0.64216924	0.66735235	0.69253545	0.71771856	0.74
290167	0.76808478	0.79326788	0.81845099	0.8436341	0.86881721	0.89
400031	0.91918342	0.94436653	1.95169083	1.97687393	2.00205704	2.02
724015	2.05242326	2.07760636	2.10278947	2.12797258	2.15315569	2.17
833879	2.2035219	2.22870501	2.25388812	2.27907122	2.30425433	2.32
943744						

2.35462055 053608	2.37980365	2.40498676	2.43016987	2.45535298	2.48
2.50571919 163473	2.5309023	2.55608541	2.58126851	2.60645162	2.63
2.65681784 273337	2.68200094	2.70718405	2.73236716	2.75755027	2.78
2.80791648 383202	2.83309959	2.85828269	2.8834658	2.90864891	2.93
2.95901512 493066	2.98419823	3.00938134	3.03456445	3.05974755	3.08
3.11011377 602931	3.13529688	3.16047998	3.18566309	3.2108462	3.23
3.26121241 712795	3.28639552	3.31157863	3.33676174	3.36194484	3.38
3.41231106 82266	3.43749417	3.46267727	3.48786038	3.51304349	3.53
3.5634097 932524	3.58859281	3.61377592	3.63895903	3.66414213	3.68
3.71450835 042389	3.73969146	3.76487456	3.79005767	3.81524078	3.84
3.86560699 152253	3.8907901	3.91597321	3.94115631	3.96633942	3.99
4.01670564 262117	4.04188874	4.06707185	4.09225496	4.11743807	4.14
4.16780428 371982	4.19298739	4.2181705	4.2433536	4.26853671	4.29
4.31890293 481846	4.34408603	4.36926914	4.39445225	4.41963536	4.44
4.47000157 81144	4.49518468	4.52036779	4.54555089	4.87293129	4.89
4.92329751 921304	4.94848061	4.97366372	4.99884683	5.02402993	5.04
5.07439615 031169	5.09957926	5.12476236	5.14994547	5.17512858	5.20
5.22549479 141033	5.2506779	5.27586101	5.30104412	5.32622722	5.35
5.37659344 250898	5.40177655	5.42695965	5.45214276	5.47732587	5.50
5.52769208 360762	5.55287519	5.5780583	5.60324141	5.62842451	5.65
5.67879073 470627	5.70397384	5.72915694	5.75434005	5.77952316	5.80
5.82988937 580491	5.85507248	5.88025559	5.9054387	5.9306218	5.95
5.98098802 690356	6.00617113	6.03135423	6.05653734	6.08172045	6.10
6.13208666 80022]	6.15726977	6.18245288	6.20763598	6.23281909	6.25

Es wurde eine logarithmische y -Achse gewählt, da mit dieser der Bereich gut konditionierte Bereich besser hervorkommt. Die Randwerte sowie zum Beispiel das Intervall zwischen ca. 0.97 und ca. 1.93, also um $(2n + 1)\frac{\pi}{2} \forall n \in \mathbb{N}$ sind gut konditioniert, da K dort unter dem Wert 1 liegt. In den Intervallen dazwischen liegen die Werte teilweise sogar sehr weit über 1 und sind somit schlecht konditioniert.

In []:

In []:

In []:

In []: