Aufgabe 2

(a) Lösen Sie folgende Gleichungen:

i. cos(x) = x <=> f(x) = x – cos(x)

*Sekantenverfahren:*

def secant(f, a, b, N):  
 if f(a) \* f(b) >= 0:  
 print("Secant method fails.")  
 return None  
 a\_n = a  
 b\_n = b  
 for n in range(1, N + 1):  
 m\_n = a\_n - f(a\_n) \* (b\_n - a\_n) / (f(b\_n) - f(a\_n))  
 f\_m\_n = f(m\_n)  
 if f(a\_n) \* f\_m\_n < 0:  
 a\_n = a\_n  
 b\_n = m\_n  
 elif f(b\_n) \* f\_m\_n < 0:  
 a\_n = m\_n  
 b\_n = b\_n  
 elif f\_m\_n == 0:  
 print("Found exact solution.")  
 return m\_n  
 else:  
 print("Secant method fails.")  
 return None  
 return a\_n - f(a\_n) \* (b\_n - a\_n) / (f(b\_n) - f(a\_n))

result1 = secant(f,-5,5,50)

***Ergebnis:*** 1 Nullstelle bei x = 0.7390851332151607

ii. 2 sin(x) = x <=> j(x) = x – 2 \* sin(x); dj(x) = 1 – 2 \* cos(x)

*Newton-Verfahren:*

def newton(f, Df, x0, epsilon, max\_iter):  
  
 xn = x0  
 for n in range(0, max\_iter):  
 fxn = f(xn)  
 if abs(fxn) < epsilon:  
 print('Found solution after', n, 'iterations.')  
 return xn  
 Dfxn = Df(xn)  
 if Dfxn == 0:  
 print('Zero derivative. No solution found.')  
 return None  
 xn = xn - fxn / Dfxn  
 print('Exceeded maximum iterations. No solution found.')  
 return None

result1 = newton(j,dj,-2,1e-6,500)  
result2 = newton(j,dj,0,1e-6,500)

result3 = newton(j,dj,1,1e-6,500)

***Ergebnis****:* 3 Nullstellen bei x1 = -1.8954942672087132; x2 = 0; x3 = 1.8954945666276892

iii. x = 2 – exp(-x) <=> q(x) = (2 – exp(-x)) – x

*Bisektionsverfahren:*

def bisection(f, a, b, N):  
 if f(a) \* f(b) >= 0:  
 print("Bisection method fails.")  
 return None  
 a\_n = a  
 b\_n = b  
 for n in range(1, N + 1):  
 m\_n = (a\_n + b\_n) / 2  
 f\_m\_n = f(m\_n)  
 if f(a\_n) \* f\_m\_n < 0:  
 a\_n = a\_n  
 b\_n = m\_n  
 elif f(b\_n) \* f\_m\_n < 0:  
 a\_n = m\_n  
 b\_n = b\_n  
 elif f\_m\_n == 0:  
 print("Found exact solution.")  
 return m\_n  
 else:  
 print("Bisection method fails.")  
 return None  
 return (a\_n + b\_n) / 2

result1 = bisection(q,-2,-1,300)  
result3 = bisection(q,-1,3,300)

***Ergebnis:*** 2 Nullstellen bei x1 = -1.1461932206205825; x2 = 1.8414056604369606

iv. x = exp(1 – x^2) <=> p(x) = exp(1 – x^2) – x

result1 = bisection(p,-5,5,100)

***Ergebnis:*** 1 Nullstelle bei x = 1

(b) In der Molekularfeldtheorie des Ferromagnetismus wird die Magnetisierung

m aus der Gleichung m = tanh(m/T) bestimmt. Bestimmen Sie die Lösung m (m != 0) als Funktion der Temperatur T mit einer Genauigkeit von 10^-6.

*Fixpunktiteration:*

def fixpt(f, x, epsilon=1.0E-6, N=500):  
 y = f(x)  
 n = 0  
 while abs(y - x) >= epsilon and n < N:  
 x = f(x)  
 n += 1  
 y = f(x)  
 if n >= N:  
 return "No fixed point for given start value"  
 return y

Um die Magnetisierung m als eine Funktion der Temperatur T abzubilden, muss die Fixpunktiteration mit verschiedenen Werten für T (zwischen 0 und 1) durchgeführt werden.

def f(x):  
 return np.tanh(x / t)  
  
  
fig, ax = plt.subplots()  
  
t = 0.01  
m\_values = []  
t\_values = []  
for i in range(0, 99):  
 t\_values.append(t)  
 m = fixpt(f, 0.1)  
 m\_values.append(m)  
 t = t + 0.01  
  
ax.plot(m\_values, t\_values)  
ax.set\_xlabel(r"$m$", fontsize=18)  
ax.set\_ylabel(r"$T$", fontsize=18)  
plt.show()

Hierfür wird für T der Startwert 0.01 gewählt. In einer Schleife wird ein Array t\_values schrittweise um 0.01 erhöht. Dabei wird zu jedem T ein korrespondierender Wert m durch Fixpunktiteration errechnet und im Array m\_values gespeichert. Die Iteration erfolgt dabei immer mit dem Startwert 0.1. Zuletzt werden die Werte der Temperatur auf die Werte der Magnetisierung geplottet und ergeben folgende Lösung:

