CGL
Blatt 7

Universität Stuttgart

Author 1: Laurens Viehoff
Author 2: Felix Roth

Abstract

Inhaltsverzeichnis

1 Aufgabe 1

Hier der Nachweis für die Nature einbindung von Quelle [**biamonte_quantum_2017**].

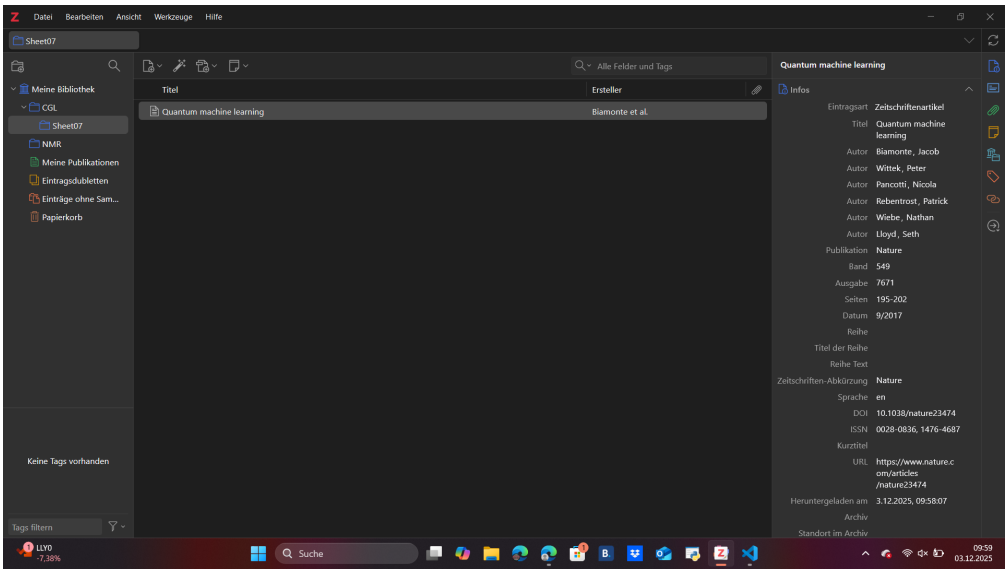

Abb. 1: Hier des Quantum machine learning paper.

Hier dann noch der Nachweis der eigenen Erstellung einer Quelle vom Buch [**demtroder_experimentalphysik_2003**].

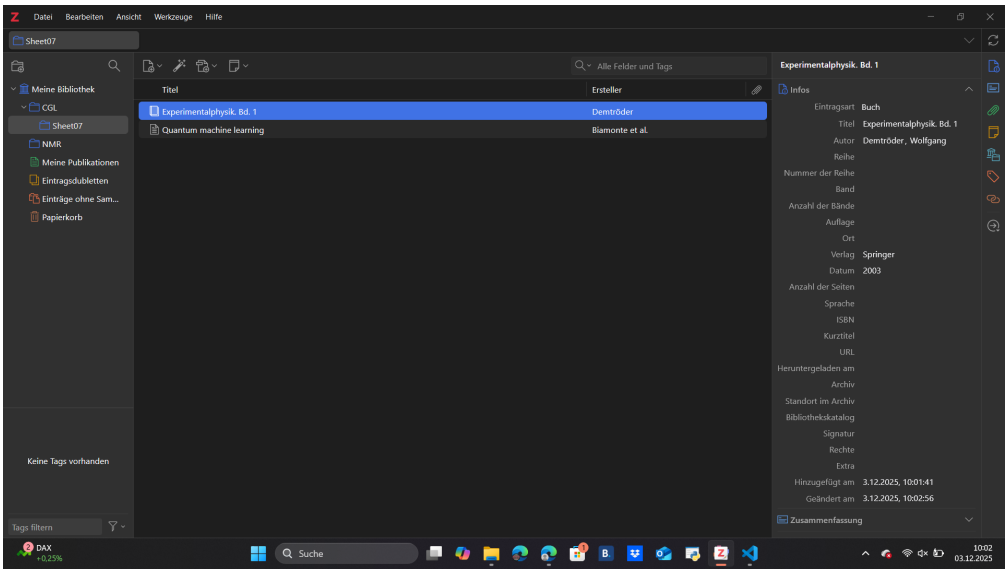

Abb. 2: Hier das Demtröder Buch.

2 Aufgabe 2

```
import random
import math
```

```
import matplotlib.pyplot as plt

def generate_random_point_in_square(length):
    """
    Generate a random point in a square of a given side length centered
        around the origin with the sides aligned with x- and y-axis.
    """
    return tuple([length*random.uniform(-0.5, 0.5), length*random.uniform
        (-0.5, 0.5)])

def distance(p1, p2):
    """
    Measures the Euclidean distance between the points p1 and p2 in 2D.
    """

    return math.sqrt((p2[0]-p1[0])**2+(p2[1]-p1[1])**2)


def point_in_circle(p, center=[0,0], radius=None):
    """
    Checks if the provided point p lies in a circle of a given radius
        around the provided center.
    """
    length = distance(center, p)

    if length <= radius:

        print("liegt im Bereich")
        return True
    else:
        print("leigt nicht im Bereich")
        return False
i= 1
full_list_of_points = []

while i < 1000:

    full_list_of_points.append(generate_random_point_in_square(2.0))
    i = i+1

filtered_list_of_points = []

while i <= len(full_list_of_points):

    if point_in_circle(full_list_of_points[i]) == True:
        filtered_list_of_points.append(full_list_of_points[i])
        i = i+1
    else:
```

```
        i =i+1

print(len(filtered_list_of_points)/(len(full_list_of_points)-len(
    filtered_list_of_points)))
# TODO: Generate separate lists of the x- and y-values of
    full_list_of_points and filtered_list_of_points
x_values_full =
y_values_full =

x_values_filtered =
y_values_filtered =

plt.scatter(x_values_full, y_values_full)
plt.scatter(x_values_filtered, y_values_filtered)
plt.show()
```

3 Aufgabe 3

```
    import matplotlib.pyplot as plt

dateiname = "sheet07\messdaten.txt"
x_werte = []
y_werte = []

# 1. Datei ffnen und Schleife ber Zeilen
with open(dateiname, 'r') as datei:
    print(f"Verarbeite Daten aus '{dateiname}'...")

    for zeilennummer, zeile in enumerate(datei, 1):

        # 2. .strip() verwenden
        bereinigte_zeile = zeile.strip()

        # 3. Filtern von leeren Zeilen und Kommentaren
        if not bereinigte_zeile or bereinigte_zeile.startswith('#'):
            # Ignoriere auch Zeilen, die mit '#' (Kommentar) beginnen
            continue

        # 4. Sonst Zeile ausgeben und Werte extrahieren
        print(f"Verarbeite: '{bereinigte_zeile}'")

        #Teile die bereinigte_zeile in Einzelteile auf
        teile = bereinigte_zeile.split()

        print(teile)

        if len(teile) == 2:
```

```
            # Wandel die Teile in Gleitkommazahlen um
            x = float(teile[0])
            y = float(teile[1])

            # Speicher die Werte in den Listen x_werte und y_werte
        x_werte.append(x)
        y_werte.append(y)


# 5. Am Ende alle Tupel x-y plotten
if x_werte:
    print(f"Erfolgreich {len(x_werte)} Datenpunkte gesammelt. Erstelle
        Plot.")

    plt.figure()
    # Verwende Sie plt.scatter() um x_werte gegen y_werte zu plotten
    plt.scatter(x_werte, y_werte, label = "yeet")

    plt.title('Messdaten aus Datei')
    plt.xlabel('X-Koordinate')
    plt.ylabel('Y-Koordinate')
    plt.grid(True)
    plt.savefig('messdaten_plot.pdf')
    plt.show()
else:
print("keine g ltigen Daten zum Plotten gefunden.")
```