Aufgabe 13 (Quadraturformel für das D2Q9 Schema)
Im Folgenden leiten wir die Quadraturformel für das D2Q9 Schema her, ausgehend von der Gauß-Hermite Quadratur in einer Dimension.

(a) Laut Gauß-Hermite Quadraturformel der Ordnung 3 gilt

$$\int_{-\infty}^{\infty} h(z) \frac{\mathrm{e}^{-z^2}}{\sqrt{\pi}} \mathrm{d}z = \sum_{i=-1}^{1} \tilde{w}_i h(z_i)$$

für Polynome h bis zu Grad 5, mit Stützstellen

$$z_{-1} = -\sqrt{3/2}, \quad z_0 = 0, \quad z_1 = \sqrt{3/2}$$

sowie Gewichten

$$\tilde{w}_{-1} = \frac{1}{6}, \quad \tilde{w}_0 = \frac{2}{3}, \quad \tilde{w}_1 = \frac{1}{6}.$$

(Dies kann als bekannt vorausgesetzt werden.) Leiten Sie hieraus durch Substitution die folgende modifizierte Variante mit Parameter $\beta > 0$ her:

$$\int_{-\infty}^{\infty} h(x) \left(\frac{\beta}{2\pi}\right)^{1/2} \mathrm{e}^{-\frac{1}{2}\beta x^2} \,\mathrm{d}x = \sum_{i=-1}^{1} \tilde{w}_i h(x_i) \tag{1}$$

mit Gewichten $\tilde{w}_i$ wie zuvor und Stützstellen

$$x_{-1} = -\sqrt{3/\beta}, \quad x_0 = 0, \quad x_1 = \sqrt{3/\beta}. \tag{2}$$

[2 Punkte]

(b) Begründen Sie mittels Gl. (1), dass

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_1(x) h_2(y) \left(\frac{\beta}{2\pi}\right) \mathrm{e}^{-\frac{1}{2}\beta(x^2+y^2)} \,\mathrm{d}x\,\mathrm{d}y = \sum_{i,j=-1}^{1} \tilde{w}_i \tilde{w}_j h_1(x_i) h_2(x_j) \tag{3}$$

für Polynome h_1 und h_2 bis Grad 5 und den Stützstellen in Gl. (2) gilt. Wieso darf auf beiden Seiten $h_1(x)h_2(y)$ durch $h(x, y)$ für ein Polynom h in zwei Variablen x und y bis zu Grad 5 ersetzt werden? [2 Punkte]

(c) Berechnen Sie nun ausgehend von Gl. (3) die Stützstellen $\vec{x}_i \in \mathbb{R}^2$ und Gewichte w_i des D2Q9 Schemas, so dass

$$\int_{\mathbb{R}^2} h(\vec{x}) \left(\frac{\beta}{2\pi}\right) \mathrm{e}^{-\frac{1}{2}\beta|\vec{x}|^2} \,\mathrm{d}^2x = \sum_{i=0}^{8} w_i h(\vec{x}_i)$$

für Polynome h bis zu Grad 5 (wobei $h(\vec{x}) \equiv h(x_1, x_2)$ komponentenweise). [2 Punkte]

Aufgabe 14 (Lattice-Boltzmann Implementierung in 2D)

(a) Auf der Webseite zur Vorlesung[1] findet sich die Vorlage `lbm2D.py` einer Python-Implementierung des LBM-Algorithmus (D2Q9 Schema). Machen Sie sich mit der Implementierung vertraut, und vervollständigen Sie den Code an den vier (mit `#### ... ####`) angegebenen Stellen. [4 Punkte]

(b) Lassen Sie nun das (ebenfalls auf der Vorlesungsseite erhältliche) Python-Programm `lbm_demo_periodic.py` laufen. Welche Auswirkungen beobachten Sie, wenn Sie `tau = 5` setzen? [1 Punkt]
Hinweis: Eine Angabe von Source-Code oder Grafiken ist in den Teilaufgaben (b) und (c) nicht gefordert.

(c) Experimentieren Sie schließlich mit dem Python-Programm `lbm_demo_noslip.py` und erklären Sie kurz, warum jetzt die durchschnittliche Gesamtgeschwindigkeit $\vec{u}_{\mathsf{avr}}(t) = \frac{1}{\sum_{\vec{x}} \rho(\vec{x},t)} \sum_{\vec{x}} \rho(\vec{x}, t)\vec{u}(\vec{x}, t)$ nicht mehr zeitlich konstant bleibt. [1 Punkt]

[1] https://tu-dresden.de/mn/math/wir/mendl/studium/courses/mosim_2017_wise

Aufgabe 15 (Mandelbrot-Menge)
Die Mandelbrot-Menge $\mathbb{M}$ besteht aus allen komplexen Zahlen c, für die die Folge $z_0, z_1, z_2, \ldots$ definiert durch

$$z_0 = 0, \qquad z_{k+1} = z_k^2 + c, \quad k = 0, 1, \ldots$$

beschränkt bleibt. Es stellt sich heraus, dass die Folge genau dann unbeschränkt wächst, falls $|z_k| \geq 2$ für ein k eintritt, da dann die nachfolgenden $z_{k+1}, z_{k+2}, \ldots$ im Wesentlichen quadriert werden. Dies liefert bereits eine Vorlage für einen Algorithmus:

```
c = ...
z = 0; k = 0
while abs(z)<2:
    z = z**2 + c
    k += 1
```

Es gilt also $c \in \mathbb{M}$ genau dann, wenn die Schleife nie abbricht. In der Praxis definiert man eine maximale Suchtiefe `depth` und nimmt $c \in \mathbb{M}$ an, falls die Schleife nach `depth` Iterationen noch nicht abgebrochen ist.

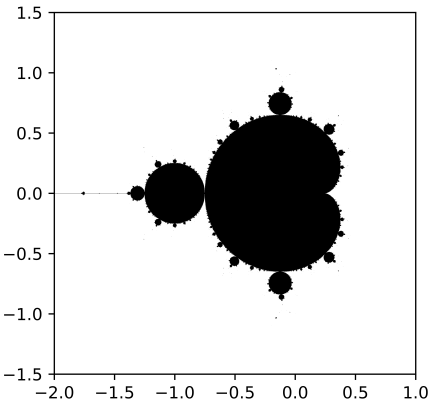
Im Folgenden initialisieren wir c mittels `numpy.meshgrid` gleich mit einem Gitter von Werten und führen den Algorithmus simultan für jeden Eintrag aus. In Python wird die imaginäre Einheit mit `1j` bezeichnet. Das folgende Programm generiert (nach etwas Laufzeit) die nebenstehende Abbildung von $\mathbb{M}$:

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

xpt = np.linspace(-2, 1, 3001)
ypt = np.linspace(-1.5, 1.5, 3001)
xgrid, ygrid = np.meshgrid(xpt, ypt)
c = xgrid + 1j*ygrid

z = np.zeros(c.shape)
n = np.zeros(c.shape, dtype='int')

depth = 512
for k in range(0, depth):
    z = z**2 + c
    n[abs(z) < 2] = k + 1
```




```
plt.imshow(n==depth, cmap='Greys', extent=[xpt[0], xpt[-1], ypt[0], ypt[-1]])
```

(a) Beschreiben Sie die Funktionsweise dieses Python-Programms. Was ist die Interpretation der Matrixeinträge in `n` nach Durchlaufen der for-Schleife? [3 Punkte]
Hinweis: Das Programm operiert elementweise auf den Einträgen der Matrizen `c`, `z` und `n`. Sie können zum Ausprobieren die Anzahl der Gitterpunkte von 3001 auf z.B. 301 reduzieren, um die Laufzeit zu verringern.

(b) Ästhetisch anspruchsvolle Muster ergeben sich durch Visualisierung der Randbereiche von $\mathbb{M}$. Hierzu reicht folgende Modifikation des Programms aus:

```
depth = 512 + 32 - 1
...
plt.imshow(n % 32, cmap='hot_r', extent=[xpt[0], xpt[-1], ypt[0], ypt[-1]])
```

Lassen Sie das modifizierte Programm laufen, und beschreiben Sie kurz den Effekt der Modifikation. [1 Punkt]
Hinweis: Interessant ist auch das Durchspielen anderer "colormaps".

(c) Der Formenreichtum des Randgebiets setzt sich beim Hineinzoomen beliebig fort. Visualisieren Sie einen frei wählbaren Ausschnitt des Randgebiets, indem Sie den Gitterbereich von `c` verändern. [2 Punkte]