Blatt 7

Aufgabe 1: Zelluläre Automaten: `Regel 30`

Schreiben Sie ein Programm, welches den eindimensionalen zellulären Automaten `Regel 30` simuliert. Der Automat besteht aus einem unendlich langen, eindimensionalen Gitter aus Zellen. Diese Zellen können den Zustand 0 (tot, weiß) oder 1 (lebendig, schwarz) annehmen. Am Anfang wird die Konfiguration der Zellen festgelegt, z. B. eine einzelne schwarze Zelle. In jedem folgenden Zeitschritt wird auf die einzelnen Zellen eine Regel angewandt:

Aktueller Zustand	111	110	101	100	011	010	001	000
Neuer Zustand	0	0	0	1	1	1	1	0

Gehen Sie davon aus, dass das System periodisch ist, d. h. der linke Nachbar der ersten Zelle ist die letzte Zelle und der rechte Nachbar der letzten Zelle ist die erste Zelle.

Aufgabe 2: Zelluläre Automaten: `Regel 90`

Schreiben Sie ein Programm, welches den eindimensionalen zellulären Automaten `Regel 90` simuliert. Der Automat besteht aus einem unendlich langen, eindimensionalen Gitter aus Zellen. Diese Zellen können den Zustand 0 (tot, weiß) oder 1 (lebendig, schwarz) annehmen. Am Anfang wird die Konfiguration der Zellen festgelegt, z. B. eine einzelne tote Zelle. In jedem folgenden Zeitschritt wird auf die einzelnen Zellen eine Regel angewandt:

Aktueller Zustand	111	110	101	100	011	010	001	000
Neuer Zustand	0	1	0	1	1	0	1	0

Gehen Sie davon aus, dass das System periodisch ist, d. h. der linke Nachbar der ersten Zelle ist die letzte Zelle und der rechte Nachbar der letzten Zelle ist die erste Zelle.

Aufgabe 3: Zelluläre Automaten: `Conway's Game of Life`

Implementieren Sie den zellulären Automaten "Conway's Game of Life". Jede Zelle kann zwei Zustände haben, lebendig (1) oder tot (0). Die Regeln für die Entwicklung dieses zweidimensionalen Automaten basieren auf der Betrachtung der Nachbarn einer jeden Zelle. In jedem Zeitschritt werden folgende Regeln berücksichtigt:

- Eine lebende Zelle lebt im nächsten Schritt weiter, sofern sie zwei oder drei lebendige Nachbarn hat.
- Eine lebende Zelle stirbt, wenn sie weniger als zwei oder mehr als drei lebendige Nachbarn hat.
- Eine tote Zelle wird lebendig, sofern sie genau drei lebendige Nachbarn hat.

- Eine tote Zelle bleibt tot, wenn sie weniger oder mehr als drei lebendige Nachbarn hat.

Diese Regeln werden simultan für das gesamte Spielfeld ausgeführt, nicht Zelle für Zelle.

Hinweise:

Als Anfangsbedingungen bieten sich zum einen ein zufälliges Feld aus lebenden und toten Zellen an. Dieses lässt sich in Python z.B. erzeugen mit

```
a = np.random.randint(0,2,size=(50,50)).
```

Ein Gleiter ist durch die folgende Anfangsbedingung gegeben:

```
glider = [[1, 0, 0],
          [0, 1, 1],
          [1, 1, 0]]
a = np.zeros((20, 20))
a[:3, :3] = glider
```

Weitere Anfangsbedingungen finden sich auf der Homepage.

Die Berechnung der Anzahl lebendiger Nachbarn kann schnell und einfach als eine 2D Faltung berechnet werden, wobei als Faltungs-Kern eine 3x3 Matrix aus Einsen mit einer Null in der Mitte verwendet wird. Die Faltung kann mit der Funktion `convolve2d` aus dem `scipy`-Paket berechnet werden:

```
filter = np.ones((3,3),dtype=int)
filter[1,1] = 0
neighbors = signal.convolve2d(a, filter ,boundary="wrap",mode="same")
```

Aufgabe 4: Logistische Abbildung

Betrachten Sie die logistische Abbildung

$$x_{n+1} = f(x_n) := r\, x_n\, (1 - x_n), \quad r \in [0,\, 4]\,.$$

- Bestimme die Fixpunkte der Gleichung und untersuche deren Stabilität.
- Simulieren Sie das System für unterschiedliche Werte von r. Interpretieren Sie das Ergebniss.
- Berechnen Sie ein Feigenbaum-Bifurkationsdiagramm: Plotten Sie (nach der Transienten) die Werte von x_n über dem entsprechenden r Wert.