Übungsblatt 13

Y. Wang, M. Ladecký, L. Pastewka

2026-01-21

Abgabe: 28. Januar 2025, 14:00 Uhr über ILIAS als Jupyter Notebook.

In dieser Übung simulieren Sie ein gravitatives N-Körper-System und vergleichen numerische Integrationsverfahren.

Analytische Aufgaben

Aufgabe A12 (5 Punkte):

Potentielle Energie: Die Gravitationspotentialenergie zwischen zwei starren Körpern ist gegeben durch:

$$V_{ij} = -G\frac{m_i m_j}{\|\vec{r}_i - \vec{r}_j\|}$$

wobei G die Gravitationskonstante ist, m_i und m_j die Massen sind, und $\vec{r}_i = (x_i, y_i)$ sowie $\vec{r}_j = (x_j, y_j)$ die Ortsvektoren der beiden Körper i und j sind. Die Norm $\|\vec{r}_i - \vec{r}_j\| = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$ bezeichnet den euklidischen Abstand zwischen den Körpern i und j.

Für ein System mit N starren Körpern ist die gesamte potentielle Energie die Summe aller paarweisen Wechselwirkungen:

$$V_{\text{ges}} = \frac{1}{2}\sum_{i=1}^{N}\sum_{\substack{j=1\\ j\neq i}}^{N}\left(-G\frac{m_i m_j}{\|\vec{r}_i - \vec{r}_j\|}\right)$$

wobei der Faktor $\frac{1}{2}$ die Doppelzählung jedes Paares vermeidet.

Kinetische Energie:

$$T = \sum_{i=1}^{N}\frac{1}{2}m_i\|\dot{\vec{r}}_i\|^2$$

Hinweis: $\|\dot{\vec{r}}_i\| = \sqrt{\dot{x_i}^2 + \dot{y_i}^2}$

Aufgabe:

1. Leiten Sie die Bewegungsgleichungen des Systems mit Lagrange-Methoden für drei Körper her. Sie können die Summenschreibweise verwenden oder die sechs Bewegungsgleichungen (für die Variablen $x_1, y_1, x_2, y_2, x_3, y_3$) explizit aufschreiben.

Programmieraufgaben

Aufgabe P12 (5+5+5 Punkte):

Wir simulieren die berühmte Achter-Dreikörperbahn mit $G = 1.0$ und den folgenden Anfangsbedingungen:

Körper	**Masse** M	**Position** $\vec{r}$	**Geschwindigkeit** $\vec{v}$
1	1.0	(-0.97, 0.24)	(0.47, 0.43)
2	1.0	(0.97, -0.24)	(0.47, 0.43)
3	1.0	(0, 0)	(-0.94, -0.86)

1. Implementieren Sie die Funktion `dgl` zur Verwendung mit `scipy.solve_ivp`. Die Vorlage ist unten angegeben. Verwenden Sie `np.linalg.norm(r_j-r_i)` um den euklidischen Abstand zwischen den Körpern i und j zu berechnen.
2. Lösen Sie das System sowohl mit `scipy.solve_ivp` als auch mit `velocity_verlet`. Verwenden Sie `dt=0.01` und `nb_steps=3000` für `velocity_verlet`. Verwenden Sie passende `t_span` und `t_eval` für `solve_ivp`. Plotten Sie die Trajektorien für beide Methoden und vergleichen Sie diese. Der `velocity_verlet`-Integrator ist unten angegeben. Beachten Sie, dass er die Funktion `dgl` zur Berechnung der Beschleunigungen verwendet.
3. Plotten Sie die Gesamtenergie als Funktion der Zeit für beide Methoden (`solve_ivp` und `velocity_verlet`) im selben Diagramm. Vergleichen Sie die maximale Energiedrift der beiden Methoden.

```
def dgl(t, y, masses):
    """
    Berechnet die Ableitung des Zustandsvektors.

    Parameter
    ---------
    t : float
        Aktuelle Zeit (ungenutzt, aber von solve_ivp benötigt)
    y : Array der Form (12,)
```

```
        Zustandsvektor: [r1, r2, r3, v1, v2, v3] abgeflacht
        (Positionen der 3 Körper, dann Geschwindigkeiten der 3 Körper)
    masses : Array der Form (3,)
        Massen der drei Körper

    Rückgabe
    --------
    dy_dt : Array der Form (12,)
        Ableitung: [v1, v2, v3, a1, a2, a3] abgeflacht
        (zuerst Geschwindigkeiten, dann Beschleunigungen)
    """
```

```
def get_accelerations(positions, velocities, masses):
    """
    Packt Positionen und Geschwindigkeiten in einen Zustandsvektor und
    extrahiert den Beschleunigungsteil aus der DGL.

    Parameter
    ---------
    positions : Array der Form (N, 2)
        Positionen der N Körper
    velocities : Array der Form (N, 2)
        Geschwindigkeiten der N Körper
    masses : Array der Form (N,)
        Massen der drei Körper

    Rückgabe
    --------
    acceleration : Array der Form (N, 2)
        Beschleunigungen der N Körper
    """
    y = np.concatenate([positions.flatten(), velocities.flatten()])
    dy_dt = dgl(0, y, masses)
    N = np.size(masses)
    return dy_dt[N*2:].reshape(N, 2)

def velocity_verlet(positions, velocities, masses, dt, nb_steps):
    """
    Integriert die Bewegungsgleichungen mit dem Geschwindigkeits-Verlet-Algorithmus.

    Parameter
    ---------
```

```
    positions : Array der Form (N, 2)
        Anfangspositionen der N Körper
    velocities : Array der Form (N, 2)
        Anfangsgeschwindigkeiten der N Körper
    masses : Array der Form (N,)
        Massen der drei Körper
    dt : float
        Zeitschritt
    nb_steps : int
        Anzahl der Integrationsschritte

    Rückgabe
    --------
    position_trajectory : Array der Form (1+nb_steps, N, 2)
        Positionsverlauf im Format (Zeit, Koordinaten, Körper)
    velocity_trajectory : Array der Form (1+nb_steps, N, 2)
        Velocitiesverlauf im Format (Zeit, Koordinaten, Körper)
    """
    N = np.size(masses)
    position_trajectory = np.zeros((1+nb_steps, N, 2))
    velocity_trajectory = np.zeros((1+nb_steps, N, 2))

    position_trajectory[0] = positions
    velocity_trajectory[0] = velocities

    pos = positions.copy()
    vel = velocities.copy()
    acc = get_accelerations(pos, vel, masses)
    for i_step in range(nb_steps):
        pos = pos + vel * dt + 0.5 * acc * dt**2
        acc_new = get_accelerations(pos, vel, masses)
        vel = vel + 0.5 * (acc + acc_new) * dt
        acc = acc_new

        position_trajectory[i_step+1] = pos
        velocity_trajectory[i_step+1] = vel

    return position_trajectory, velocity_trajectory
```