Klassische Mechanik (SS 2018) – Blatt 4

Aufgabe 10: Zentralpotential (6 Punkte)
Wir betrachten verschiedene Arten von Zentralpotentialen und deren Eigenschaften.

(a) Ein Massepunkt der Masse m bewege sich unter dem Einfluss einer Zentralkraft mit dem Potential

$$V(r) = \frac{c}{r^2}, \quad c \neq 0 \text{ ist konstant.} \tag{1}$$

Geben Sie die beiden Erhaltungsgrößen des Systems an. Berechnen Sie den größten und kleinsten Abstand des Massepunkts zum Kraftzentrum. (2 Punkte)

(b) Für welche Konstanten c und n sind im Kraftfeld

$$\boldsymbol{F}(\boldsymbol{r}) = \frac{c}{r^n}\hat{\boldsymbol{r}} \tag{2}$$

stabile Kreisbahnen möglich? (2 Punkte)

(c) Beweisen Sie, dass für die Bewegung eines Teilchens der Masse m im Zentralpotential $V(r) = \frac{\alpha}{r}$ (α ist eine Konstante) der Runge-Lenz-Vektor

$$\boldsymbol{A} = \frac{\alpha}{|\alpha|}\left(\frac{\boldsymbol{p}\times\boldsymbol{L}}{m\alpha} + \frac{\boldsymbol{r}}{r}\right) \tag{3}$$

eine Erhaltungsgröße ist. (2 Punkte)

Aufgabe 11: Das explizite Euler-Verfahren (14 Punkte)
In dieser Aufgabe wollen wir uns mit dem numerischen Lösen von Differentialgleichungen auseinandersetzen. Es ist hierbei zweckmäßig mit dem *Euler-Verfahren* zu beginnen. Das Ausgangsproblem sei eine Differentialgleichung oder ein Differentialgleichungssystem der Form

$$\dot{\boldsymbol{y}}(t) = \boldsymbol{F}(t,\boldsymbol{y}(t)), \quad \boldsymbol{y}(0) = \boldsymbol{y}_0. \tag{4}$$

Es handelt sich also um DGLs erster Ordnung. Deren Lösung ist bekannterweise durch einfaches Integrieren zu erhalten

$$\boldsymbol{y}(t) = \boldsymbol{y}_0 + \int_{t_0}^{t} \boldsymbol{F}(s,\boldsymbol{y}(s))\,\mathrm{d}s. \tag{5}$$

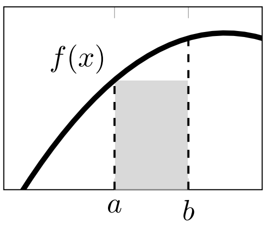
Da es im Allgemeine nicht möglich ist jedes Integral analytisch zu lösen, diskretisieren wir zunächst die Zeitachse, d.h. $t \to t_k$ und $k \in \mathbb{N}_0$, sodass die obige Gleichung auf die Form

$$\boldsymbol{y}(t_{k+1}) = \boldsymbol{y}(t_k) + \int_{t_k}^{t_{k+1}} \boldsymbol{F}(s,\boldsymbol{y}(s))\,\mathrm{d}s. \tag{6}$$

umgeschrieben werden kann.
Um das Integral zu approximieren, verwenden wir eine Boxregel. Angenommen wir wollen die Funktion $f(x)$ im Intervall $[a,b]$ integrieren, so besagt diese Regel

$$\int_a^b f(x)\,\mathrm{d}x \approx (b-a)f(a). \tag{7}$$

Im wesentlichen wird hierbei die Fläche unter einer Funktion durch ein Rechteck genähert (vgl. nebenstehende Abbildung).



Übertragen auf unser Problem, erhalten wir

$$\boldsymbol{y}(t_{k+1}) \approx \boldsymbol{y}(t_k) + (t_{k+1} - t_k)\boldsymbol{F}(t_k, \boldsymbol{y}(t_k)). \tag{8}$$

Einführen der Schrittweite $h = t_{k+1} - t_k$ führt letztlich auf das eigentliche Euler-Verfahren:

$$\boldsymbol{y}(t_{k+1}) = \boldsymbol{y}(t_k) + h\boldsymbol{F}(t_k, \boldsymbol{y}(t_k)), \tag{9}$$

$$t_k = t_0 + h \cdot k, \quad \text{und } k \in \mathbb{N}_0. \tag{10}$$

Im Folgenden betrachten wir nun ein simples Federpendel mit der DGL $\ddot{x} + \omega_0^2 x = 0$, das wir nun numerisch lösen wollen. Es gilt $\omega_0^2 = \frac{D}{m}$ mit $m = 0.5\,\mathrm{kg}$, sowie die Anfangsbedingungen $\dot{x}(0) = 0$ und $x(0) = -1$.

(a) Schreiben Sie die DGL zweiter Ordnung in ein DGL-System erster Ordnung um, indem Sie $y_1 = x$ und $y_2 = \dot{x}$ setzen. Damit sollten Sie in der Lage sein, die DGL in der Form $\dot{\boldsymbol{y}} = \boldsymbol{F}(\boldsymbol{y})$ darzustellen. (2 Punkte)

(b) Implementieren Sie das Euler-Verfahren in einer von ihnen bevorzugten Programmier-, oder Skriptsprache (bspw. Python, C++, Fortran, Matlab, Mathematica, C, ...).
Hinweis: Beachten Sie den Hinweis zum Programmieren im grauen Kasten! (6 Punkte)

(c) Lösen Sie die DGL aus (a) nun explizit mit ihrem Programm für insgesamt $t_{\text{ges}} = 100\,\mathrm{s}$. Für die Kreisfrequenz wählen Sie $\omega_0 = 0.628\,31\,\mathrm{s}^{-1}$. Für den Zeitschritt wollen wir die Fälle $h = 0.1, 0.01, 0.001$ untersuchen. Plotten Sie für jeden der Fälle $x(t)$, $\dot{x}(t)$, $E_{\text{Ges}}(t)$, sowie die Phasenraumdarstellung $p(x)$. Was stellen Sie fest? (6 Punkte)

Hinweis zum Programmieren:
Im Folgenden finden Sie einen Pseudocode des expliziten Euler-Verfahren. Übertragen Sie diesen in dieser oder ähnlicher Weise in eine Programmiersprache ihrer Wahl. Beachten Sie, dass Sie jedoch noch das Feld $\boldsymbol{F}$ als Funktion implementieren müssen.

```
function EULER_METHOD(x0, v0, h, t_ges, F)
    t = [], x = [], v = []                 ▷ initialize empty lists/vectors
    for k in range(t_ges/h) do
        t ← k * h                          ▷ save time in t list
        if k = 0 then
            x ← x0                         ▷ save x0 in x list
            v ← v0                         ▷ save v0 in v list
        else
            f = F(x_{k-1},v_{k-1},t_{k-1})
            x ← x_{k-1} + h * f[0]         ▷ calculate new x and save in list
            v ← v_{k-1} + h * f[1]         ▷ calculate new v and save in list
        end if
    end for
    return t, x, v
end function
```

Hierbei handelt es sich nur um einen Vorschlag, wie das Euler-Verfahren implementiert werden kann. Sicherlich gibt es auch noch effizientere Varianten.