Übung 02b: `numpy + scipy`
Numerisch rechnen mit Python

Ziel der Übung ist das Kennenlernen der Pakete `numpy` und `scipy` anhand der Simulation und Identifikation dynamischer Systeme.

Teil 1: Simulation mittels `odeint`

In der vorangegangenen Übung wurden die Bewegungsgleichungen eines mechanischen Systems mit zwei Freiheitsgraden hergeleitet. Dabei handelt es sich um zwei gekoppelte nichtlineare Differentialgleichungen zweiter Ordnung. Eine analytische Lösung dieser Gleichungen ist nicht möglich.
Mit numerischen Integrationsverfahren kann allerdings für gegebenen Anfangswerte näherungsweise eine Lösung (d.h. der zeitliche Verlauf aller Bewegungsgrößen) bestimmt werden.

Aufgaben

Hinweise:

- Bearbeiten Sie die vorgegebene Datei `simulation.py`
- Stellen Sie sicher, dass sie den Inhalt des Notebooks Simulation dynamischer Systeme.ipynb zur Kenntnis genommen und verstanden haben.

1. Importieren Sie die Funktionen zur numerischen Berechnung der Beschleunigungen $\ddot{x}$ und $\ddot{\varphi}$ aus dem Modul `lagrange_lsg` ($\hat{=}$ Ergebnis der Übung zu Kurs02a).

2. Schreiben Sie eine Funktion `rhs(z,t)`, welche die Ableitung $\dot{\mathbf{z}}$ des Zustands berechnet. Gehen Sie dabei von folgender Zustands-Definition aus:

   $$\mathbf{z} = (z_1, z_2, z_3, z_4)^T = (x, \varphi, \dot{x}, \dot{\varphi})^T$$

   Hinweis: Die rechte Seite der DGL hängt nicht von der Zeit ab.

3. Erstellen Sie einen Array für die Simulationszeit und legen Sie sinnvolle Anfangsbedingungen fest (z.B. $x(0) = 0$, $\varphi(0) = \frac{\pi}{2}$, $\dot{x}(0) = 0$, $\dot{\varphi}(0) = 0$).

4. Benutzen Sie die Funktion `odeint` aus dem Modul `scipy.integrate` um (näherungsweise) den Verlauf der vier Zustandsgrößen zu bestimmen.

5. Stellen Sie $x(t)$ und $\varphi(t)$ unter Nutzung des vorhandenen Codes grafisch dar.

Teil 2: Parameteridentifikation mit `fmin`

Jetzt wird angenommen die Parameter m_2 und l seien unbekannt, aber es existieren Messwerte der Bewegung. Mittels `fmin` sollen diejenigen Werte für m_2 und l gefunden werden, mit denen die gemessenen Werte am besten per Simulation reproduzierbar sind.
Hinweis: Bearbeiten Sie diese Aufgaben in der Datei `identifikation.py` bzw. `lagrange_lsg.py`!

1. Erstellen Sie in dem Modul `lagrange_lsg` zwei neue Funktionen `xdd_fnc2`, `phidd_fnc2`, welche zusätzlich zu den bisherigen Argumenten (vier Zustandskomponenten und die Kraft) noch die beiden Parameter m_2 und l als Argumente empfangen und importieren sie diese Funktion in `identifikation.py`.

2. Laden Sie mittels `np.load(...)` oder `np.loadtxt` die (fikitiven) Messwerte aus der entsprechenden Datei.
   Hinweis: Aus Praktikabilitätsgründen handelt es sich ebenfalls um Simulationsdaten. Gleichen Sortierung wie in Teil 1, Simulationsdauer 10s.

3. Erstellen sie eine Funktion `min_target(p)`, die einen Parameter-Array als Argument erwartet. Sie können davon ausgehen, dass der Array p zwei Elemente hat. Erstellen Sie lokale Variablen `m2`, `l` und weisen Sie diesen den Inhalt von p zu.

4. Definieren sie *in der Funktion* `min_target` die Funktion `rhs(z, t)`, analog zu Teil 1, allerdings unter Beachtung der in der übergeordneten Funktion festgelegten Parameter-Werte für m_2 und l.
   Hinweis: Verschachtelte Namensräume ausnutzen, siehe Kurs01b.

5. Führen Sie bei jedem Aufruf von `min_target` eine Simulation mit den entsprechenden Parameterwerten für m_2 und l durch. Wählen Sie die Anfangswerte so, dass sie zu den Messdaten passen.

6. Berechnen Sie ein quadratisches Maß für den Positionsfehler des Wagens (simulierte Werte minus Messwerte) und geben Sie dieses Maß als Ergebnis der Funktion `min_target` zurück.

7. Benutzen Sie `scipy.optimize.fmin`, um die optimalen Werte für m_2 und l zu finden. Lassen Sie ggf. Status-Informationen bzw. Zwischenergebnisse in `min_target` ausgeben. Nutzen Sie z. B. die Startwerte `p0 = [0.5, 0.7]`.