Übung 4b: Verschachtelte Funktionen, Funktionale Programmierung, Exceptions, Imports

Gegeben sind vier Dateien mit numerischen Daten. Aus diesen Daten soll jeweils eine quadratische Matrix (genauer: ein 2d-numpy-array) erzeugt werden. Diese Matrix, A, definiert ein lineares dynamisches System (vgl. Kurs 2b):

$$\dot{x} = Ax \quad \text{mit} \quad x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \quad \text{und} \quad A \in \mathbb{R}^{n \times n} \tag{1}$$

welches für zufällige Anfangswerte simuliert werden soll. Die Dimension n des Zustandsvektors ergibt sich jeweils aus der Anzahl der Einträge in der Datei. Der zeitliche Verlauf der ersten Zustandskomponente x_1 ist grafisch darzustellen.
Nutzen Sie die in der Datei `simulation.py` vorgegebenen Code-Schnipsel.
Spyder-Tipp: Blockweise Kommentarzeichen entfernen: `Edit→Comment/Uncomment`

Aufgaben

1. Nutzen Sie `np.loadtxt` um den Inhalt von `data1.txt` in einen Array zu laden.

2. Laden Sie in einer Schleife den Inhalt aller `data*.txt`-Dateien und geben sie die entsprechenden arrays aus.
   Welcher Fehler tritt dabei auf? Ergänzen Sie die vorgegebene try-except-else Struktur um den Fehler abzufangen. Nach Ausgabe einer Fehlermeldung, soll mit der nächsten Datei fortgefahren werden.

3. Aus den arrays soll nun die für die Simulation benötigte rhs-Funktion (rhs: „right hand side“ von Gleichung (1) erzeugt werden. Nutzen Sie dafür die Funktionen `create_rhs_from_1darr(...)` und `rhs_factory(...)`. Letztere muss derart angepasst werden, dass sie ein Funktionsobjekt zurückgibt. Verschieben Sie dazu die Funktion `rhs(...)` **in** die `rhs_factory(...)`.

   Spyder-Tipp: `ALT` +↑ und `ALT` +↓ dienen zum Verschieben von einer oder mehreren Zeilen.

4. Stellen Sie zu Beginn der Funktion `rhs_factory(...)` mittels `assert` sicher, dass die übergebene Matrix quadratisch ist. (Hinweis: `shape`-Attribut von `array`)

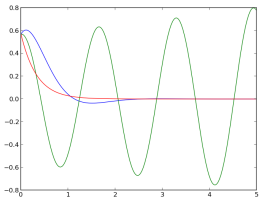
5. Erzeugen Sie vor der Rückgabe des `rhs`-Funktionsobjekts ein Attribut für die Anzahl der Zustandskomponenten.
   Hintergrund: Es treten sowohl 2×2- als auch 3×3-Matrizen auf, d.h. Systeme mit Zustandsdimension 2 bzw. 3. Die jeweilige Anzahl wird für einen Anfangszustand der richtigen Größe benötigt.

6. Machen Sie sich klar, woher die `rhs`-Funktionsobjekte ihre Daten (Matrix A) beziehen (Namensräume).

7. Führen Sie mit Hilfe der Funktion `simulation(...)` die Simulation zunächst für das erste `rhs`-Objekt durch.

8. Wenden Sie mittels `list(map(...))` die Funktion `simulation(...)` auf alle Elemente ihrer `rhs`-Liste an.

   Erwünschtes Ergebnis:

9. Schränken Sie mittels `filter` und einer geeigneten lambda-Funktion die Simulation auf Systeme der Zustandsdimension 2 ein. Nutzen Sie dazu das in Aufgabe 5 erzeugte Attribut.

10. Lagern Sie die Funktionen `create_rhs_from_1darr(...)` und `rhs_factory` in ein neues Modul namens `data_tools` aus. Ergänzen Sie die erforderlichen `import`-Anweisungen.
11. Ersetzen Sie die Aufrufe von `map` und `filter` durch List-Comprehension (siehe Folie 3).
12. (Zusatz): Bestimmen sie für jede A-Matrix die Eigenwerte (`np.linalg.eig(A)`) und beschriften Sie die Kurven mittels `plt.text(x, y, txt)`