Exercise 08: Nested Functions, Functional Programming, Exceptions, Imports

Given are four files with numerical data. From each of these data files a square matrix (i.e. a 2d-numpy-array) is to be generated. This matrix, A, defines a linear dynamic system (cf. course 3):

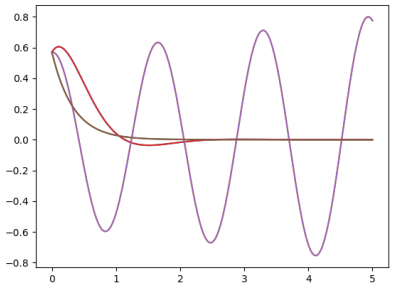
$$\dot{x} = Ax \quad \text{with} \quad x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \quad \text{and} \quad A \in \mathbb{R}^{n \times n} \tag{1}$$

which is to be simulated for random initial values. The dimension n of the state vector results in each case from the number of entries in the file. The time course of the first state component x_1 is to be represented graphically.
Use the code snippets given in the `simulation.py` file.
Spyder Tip 1: Remove comment characters block by block: `Edit→Comment/Uncomment`
Spyder Tip 2: Use `ALT` +↑ and `ALT` +↓ for moving one or more lines. (If this does not work, see *Tools* → *Preferences*) → *Keyboard Shortcuts* and search for `move line up` and `move line down`.

Tasks

1. Use `np.loadtxt` to load the contents of `data1.txt` into an into an array.
2. Loop load the contents of all `data*.txt` files and output the corresponding arrays. What error occurs during this process? Complete the given `try` - `except` - `else` -structure to catch the error. After outputting an error message, continue with the next file.
3. From the arrays now the rhs-function needed for the simulation (rhs: „right hand side“ of equation (1)) is to be generated. Use the functions `create_rhs_from_1darr(..)` and `rhs_factory(...)`. The latter must be adapted in such a way that it returns a function object. To do this, move the `rhs(...)` **inside** the function to the `rhs_factory(...)`.
4. At the beginning of the `rhs_factory(...)` function, use `assert` to make sure that the matrix is square. (Note: `shape` attribute of `array`).
5. Before returning the `rhs` function object create an attribute `rhs.state_dimension` for the number of state components.
   Background: in the dataset bot cases, 2×2 and 3×3 matrices, occur, corresponding to systems with state dimension $n = 2$ and $n = 3$ respectively. The respective number is required for an initial state of the right size.
6. Make sure you understand where the `rhs` function objects get their data (matrix A) from (namespaces).
7. Using the `simulation(...)` function, run the simulation first for the first `rhs` object.
8. Use `list(map(...))` to apply the `simulation(...)` function to all elements of your `rhs` list.

   Desired result:



9. Use `filter` and a suitable lambda function to restrict the simulation to systems of the state dimension 2. Use the attribute generated in Task 5 for this purpose.

10. Store the functions `create_rhs_from_1darr(...)` and `rhs_factory` to a new module named `data_tools`. Add the required `import` statements to `simulation.py` (which is your main module).
11. Replace the calls to `map` and `filter` with list comprehension (see slide 3).
12. (addition): For each A matrix, determine the eigenvalues (`np.linalg.eig(A)`) and add them as label to the curves using `plt.text(x, y, txt)`