Scientific Programming
(Wissenschaftliches Programmieren)

Exercise 10

Preparation

- The exercises 1A and 1B should be preferably solved by two developers, each of them making one part of the exercise in **separate** repository. Once each developer finished the work, the repositories should be synchronized / merged as shown in the lecture. Please note, that the two repositories must have a common ancestor (e.g. before starting the exercise, one developer should clone the repository of the other and work in that).
- In case, the exercises 1A and 1B are both solved by the same developer, the developments should happen in two different branches, which should be then merged to main accordingly.

1A. Optimized solver algorith (Developer 1)

- Write a small standalone “profiler” script `profile_solvers`, which creates a random linear system of equations (coefficient matrix and right-hand-side vector) of a given size using the `numpy.random.random()` routine and and calls the Gaussian-elimination routine for that matrix.
- Run the profiler script with different matrix sizes, up to a size, which is large enough that the calculation takes ca. 5-10 seconds. How do the execution times scale with the dimension of the equations (linear, quadratic, cubic, etc.)? You can measure the execution time of a python script by prepending the `time` command when starting the scipt (e.g. `time python profile_solvers.py` or `time python3 profile_solvers.py`).
- Replace the Gaussian-elimination algorithm in the `linsolve` script by the `numpy.linalg.solve()` routine. How does it affect the execution speed for the same matrix sizes?
- Rename the gaussian_eliminate() routine into solve() (as it does not use the Gaussian elimination algorithm any more…).
- Make sure, all tests are still passed.
- Commit your changes (add also the profile script to the repository).

1B. Linear solver as a program (Developer 2)

- Create a standalone Python program `linsolve`, which reads a linear system of equations from a file `linsolve.in`, solves the linear system of equation and writes the result into a file `linsolve.out`.
- The script should import the solver module (solvers.py) and use the routine therein to solve the linear system of equation.
- The script should do its file I/O using the following three routines from a new module linsolverio.py:
  - read_input(): reads the input from a given file and returns the matrix *A* and the right hand side vector *b*.
  - write_result(): writes the result vector *x* into a given file

    - write_lindep_error(): writes an error message about linear dependency into a given file

- The input file should have following format:

  ```
  Matrix_A_1st_line
  Matrix_A_2nd_line
  …
  RHS_of equation_as_one_line
  ```

  Example:

  ```
  2.0  4.0  4.0
  1.0  2.0 -1.0
  5.0  4.0  2.0
  1.0  2.0  4.0
  ```

- The output file should contain the solution vector in one row, or the string “`ERROR::LINDEP: Linearly dependent equations`” if the linear system of equation is linearly dependent.

  Example output when the linear system of equation has a solution:

  ```
  0.666666666666667 0.416666666666667 -0.5
  ```

  Example output when the linear system of equation is linearly dependent

  ```
  ERROR::LINDEP: Linearly dependent equations
  ```

- Make sure, the `linsolve` script exists with a well formatted error message gracefully (and not with an unhandled exception), if the input file is not found.

- You migh find the `numpy.loadtxt()` and the `numpy.savetxt()` routines helpful when implementing the functions for reading the input and writing the result..

- Commit your changes to the repository.

2. Synchronization of concurrent developments

- Merge the concurrent developments in one repository.

- Check, whether the unit tests and the standalone script with file I/O work as expected. If any problems arise, fix them and commit the changes to the repository.

- Synchronize the two repositories, so that the main branch of both repositories contain the same project status (a working linsolve program reading its input from a file, writing its results to a file and using the optimized Numpy routine for solving the linear system of equations).