Scientific Programming
(Wissenschaftliches Programmieren)

Exercise 9

1. Raising exceptions

- Change the `gaussian_eliminate()` and the `lu_decompose()` functions, so that they signalize linear dependency by raising an exception (e.g. `ValueError`) with a proper message.
- Modify the tests with the linearly dependent systems to check whether the correct exception is raised.
- Commit your changes.

2. API-documentation

- Create a docs/ subfolder in your project and set up a Sphinx-documentation, which automatically extracts the API-documentation from your sources.
- Check that the API-documentation is created correctly.
- Add the relevant files to your repository (and the irrelevant files/directories to the `.gitignore` file).
- Commit your changes

3. *LU-decomposition

- Implement the function `lu_decompose()` for LU decomposition with partial pivoting. The function should produce the LU-factorized matrix (as one matrix, see the form below) along with a permutation vector representing the row-permutations. If the matrix is linearly dependent, the routine should raise a `ValueError` exception.
- Implement the function `forward_substitute()` for forward substitution. It should take the LU-decomposed matrix LU (as returned by the LU-decomposition routine) and a vector b as arguments, solve the equation $Ly = b$ and return the solution vector y.
- Implement the function `backward_substitute()` for backward substitution. It should take the LU-decomposed matrix LU (as returned by the LU-decomposition routine) and a vector y as arguments, solve the equation $Ux = y$ and return the solution vector x.
- Add unit tests for all three newly created functions. Make sure to test `lu_decompose()` also for a linearly dependent system.
- Commit your changes to the repository.
- Revise the `gaussian_eliminate()` function so that it solves the linear system of equation by calling these new functions using appropriate arguments. Rename the routine to `solve()` (as it does not use the Gaussian elimination any more). Make sure, it passes all previously stored unit tests.
- Commit your changes to the repository.

- **Note:** Given the lower triangle matrix L and the upper triangle matrix U, the LU-decomposed matrix LU has the form as shown below:
- $\mathrm{L} = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix} \qquad \mathrm{U} = \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix} \qquad \mathrm{LU} = \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ l_{21} & u_{22} & u_{23} \\ l_{31} & l_{32} & u_{33} \end{pmatrix}$

4. *Cholesky-decomposition

- Implement the function `cholesky_decompose()` for Cholesky-decomposition of symmetric, positive definite real matrices. The function should take a symmetric, positive definite matrix A as argument and return the lower triangle matrix L, so that

$$\mathrm{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \mathrm{L}\ \mathrm{L}^{\mathrm{T}} \qquad \text{with} \qquad \mathrm{L} = \begin{pmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{pmatrix}$$

- Write unit tests for the new function.
- Create the routine `solve_posdef()` which solves a positive definite linear system of equations using the Cholesky-decomposition and the forward and backward substitution routines.
- Write unit tests for the new function.
- Commit the changes.