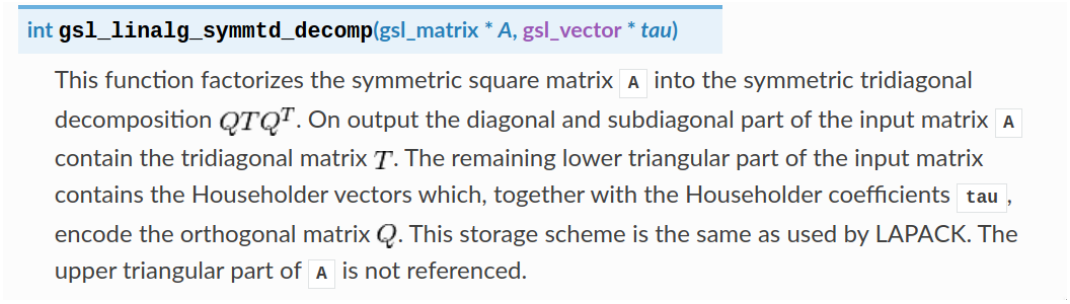
Zadanie 4 – sprowadzenie macierzy do postaci trójdiagonalnej i znalezienie jej wartości własnych.

Rozwiązanie tego zadania podzieliłam na dwie części – sprowadzenie do postaci trójdiagonalnej napisałam w języku C++ (z użyciem biblioteki GSL), natomiast znalezienie wartości własnych w języku Python (z użyciem bibliotek SciPy i numpy).

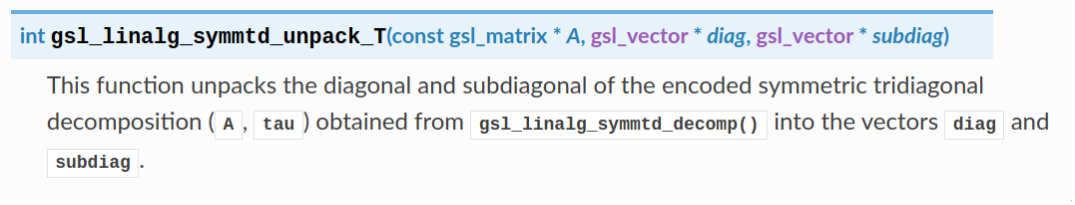
Faktoryzuję macierz A, używając funkcji gsl_linalg_symmtd_decomp, zgodnie z dokumentacją GSL.



int **gsl_linalg_symmtd_decomp**(gsl_matrix * *A*, gsl_vector * *tau*)

This function factorizes the symmetric square matrix `A` into the symmetric tridiagonal decomposition QTQ^T. On output the diagonal and subdiagonal part of the input matrix `A` contain the tridiagonal matrix T. The remaining lower triangular part of the input matrix contains the Householder vectors which, together with the Householder coefficients `tau`, encode the orthogonal matrix Q. This storage scheme is the same as used by LAPACK. The upper triangular part of `A` is not referenced.

Następnie "rozpakowuję" diagonalę i poddiagonalę do odpowiednich wektorów, używając funkcji gsl_linalg_symmtd_unpack_T



int **gsl_linalg_symmtd_unpack_T**(const gsl_matrix * *A*, gsl_vector * *diag*, gsl_vector * *subdiag*)

This function unpacks the diagonal and subdiagonal of the encoded symmetric tridiagonal decomposition (`A` , `tau`) obtained from `gsl_linalg_symmtd_decomp()` into the vectors `diag` and `subdiag` .

Diagonala:
[1.58333 -0.0125957 2.36902 0.060241 1.90646 1.09354]

Poddiagonala:
[-2.396467 0.9347593 -2.078863 1.177896e-15 -0.2911902]

Następnie w Pythonie tworzę dwie tablice w zawierające odpowiednio diagonalę i poddiagonalę. Wywołuję funkcję eigvalsh_tridiagonal, która zgodnie z dokumentacją SciPy znajduje wartości własne symetrycznej macierzy trójdiagonalnej.

Eigenvalues:
[-2.00000069 -1.0000003 0.99999854 2.00000146 2.99999778 3.99999851]