High Performance Computing
Homework 11

Sara Restrepo Velasquez and Dai Nam Nguyen
April 28th, 2024

For this assignment, the **mkl_solver** function (using **dgesv**), **mkl_solver_symm** function (using **dsysv**) and **mkl_solver_symm_work** function (using **dsysv_work**) were implemented in the Cython module file, **module.pyx**. This module was compiled with the **make** command which was used in the **main_program.py** script to solve the system AX = B by factorizing A and overwriting B with the solution X:

- **A** is a symmetric matrix of size N×N. Each diagonal element is K_{ii}=2. Its last diagonal coefficient is K_{NN}=1. Each element next to a diagonal coefficient is −1. The matrix **A** is zero elsewhere.
- **B** is a vector of size N. Each element in **B** is zero except its last element f_N=1/N
- Where **X** is a vector of size N.

dgesv can solve a general system of linear equations, while **dsysv** and **dsysv_work** are specialized solvers for symmetric matrices. As shown in Table 1 (from results.txt), it may be observed that the **mkl_solver_symm** function (using **dsysv**) is more effective with small size matrices, while the **mkl_solver** function (using **dgesv**) is more effective for matrices with a larger size. It is likely that **LAPACKE_dsysv** ‘s slowdown may be related to using BLAS level 3 operations, from a specific version of BLAS.

By using **LAPACKE_dsysv_work** (in **mkl_solver_symm_work**) and using a work array with dimension lwork = 1 , BLAS is forced to use level 2 operations. It can be noticed that **LAPACKE_dsysv_work** is more effective for all matrix sizes tested, as shown in Table 1. This may be due to lower computational complexity for level 2 operations.

Table 1: CPU time for solving using dgesv and dsysv and dsysv_work.

N matrix size	**T1 [s] dgesv CPU time**	**T2 [s] dsysv CPU time**	**T3 [s] dsysv_work CPU time**	**Ratio T1 / T2**	**Ratio T1 / T3**
10	0.018554	0.000324	0.00012	57.27	154.62
100	0.007759	0.002733	0.000184	2.84	42.17
1000	0.040149	0.020583	0.004284	1.95	9.37
5000	0.160039	0.439469	0.113413	0.36	1.41
10000	0.642329	2.107753	0.497343	0.30	1.29
50000	59.823701	79.249212	9.695454	0.75	6.17