Homework 11

Juan Camilo Velasquez and Joaquin Rodriguez

Assignment 11: In the present assignment, we used Cython to create a wrapper module for Lapack's DSYSV. DSYSV is an mkl implementation that computes the solution to the System of linear equations with a real symmetric matrix A and multiple right-hand sides B. Where A is the coefficient matrix, for this assignment, a Python function "mkl_solver_symm" was created, as shown in Figure 1.

```
def mkl_solver_symm(double[:,::1] A, double[:,::1] B, algorithm=None):
    """
    Function that uses MKL's LAPACK DESV routine to solve a general system
    of equations. This uses a LU factorization approach.

    The system solved has the form: AX = B

    INPUTS:
    - A: double array (n x n) with the coefficient matrix info.
    - B: double array (n x nrhs) with the right hand sides of the system.

    Note: This function overwrites the values in A and B.

    A is overwritten with the values of the LU decomposition.
    B is overwriten with the values of the solution.

    """
    cdef int64_t lda, ldb, n, nrhs, matrix_layout
    cdef int64_t[:] ipiv_memview, i
    cdef char uplo
    cdef double[:] work
    cdef int64_t lwork

    matrix_layout = 101 # Row major

    uplo = 'L'

    lda = A.shape[1]
    ldb = B.shape[1]

    n = A.shape[0]
    nrhs = B.shape[1]

    lwork = 1
    work=np.zeros(lwork)

    # Use numpy to create the memory for the ipiv input.
    ipiv_memview = np.zeros(A.shape[0], dtype=np.int64)

    # Call LAPACK function imported from C library.

    if algorithm=='dsysv_work':
        LAPACKE_dsysv_work( matrix_layout, uplo, n, nrhs,
                            &A[0,0], lda, &ipiv_memview[0],
                            &B[0,0], ldb, &work[0], lwork )
    elif algorithm=='dsysv':
        LAPACKE_dsysv( matrix_layout, uplo, n, nrhs,
                       &A[0,0], lda, &ipiv_memview[0],
                       &B[0,0], ldb )
    else:
        LAPACKE_dsysv( matrix_layout, uplo, n, nrhs,
                       &A[0,0], lda, &ipiv_memview[0],
                       &B[0,0], ldb )
```

Figure 1 Code modifications (highlighted)

The code was used to solve two linear systems of equations, one with a coefficient matrix of size (5X5) and three right-hand sides and one with a coefficient matrix of size (10000 X 10000) and a single right-hand side. The run times were compared for the DSYSV and the DGESV algorithms. During the development of the codes, it was found that the DSYSV algorithm did not provide additional performance gains. Therefore, the implementation of DSYSV_WORK, an implementation that has access to a more efficient BLAS library implementation, was also included as part of the Cython module.

The runtimes for the three algorithms and the Scipy version of DSYSV and DGESV for both systems of equations are shown in Table 1:

Algorithm	**Run Time (5X5 System)**	**Run Time (10000X10000 System)**
LAPACK DGESV	0.013054s	0.914589 s
LAPACK DSYSV	0.053441 s	2.145465 s
LAPACK DSYSV_WORK	0.004576 s	0.439012 s
SCIPY DGESV	NA	1.111397 s
SCIPY DSYSV	NA	0.690050 s

Table 1 Run time comparison for the different algorithms

Conclusions: As seen in the results, the DSYSV algorithm does not improve the runtimes by itself compared to the DGESV. These results are counter-intuitive because symmetric algorithms usually exploit the characteristics of the coefficient matrix to reduce the number of operations required to solve the System. However, when the DSYSV_WORK algorithm is used, it, in fact, reduces the total runtime to approximately half of the original execution time. The reasons behind those differences might be related to the use of different levels of the BLAS library in both algorithms.