Assignment 11 Report

High Performance Computing

Pratik Mitra

James Smith

1. Create an interface to the Fortran searchutils.f90 using f2py.

Evaluate the CPU time of the Fortran search algorithm calls using a large array of values against numpy’s searchsorted and numpy

• Create a document with a table showing the performance of the different implementations

|  |  |
| --- | --- |
| Search Type | CPU Time (seconds) |
| Linear (Fortran) | 0.010589122772216797 |
| Binary (Fortran) | 0.010595560073852539 |
| Numpy (SortedSearch) | 0.010623693466186523 |
| Numpy (Native) | 0.0198056697845459 |

2. Create an interface to the lapack DSYSV function.

• Write a small document with the difference in CPU times of both solvers.

|  |  |
| --- | --- |
| Solver Type | CPU Time (seconds) |
| linalg\_symm.py | 2.200225115 |
| linalg\_solve.py | 0.799071312 |

The linalg\_symm.py takes significantly longer to solve than linalg\_solve.py

Changes made in module.pyx

cdef extern from "mkl\_lapacke.h" nogil:

int LAPACKE\_dsysv(int matrix\_layout, char uplo, int64\_t n, int64\_t nrhs,

double\* a, int64\_t lda, int64\_t\* ipiv,

double\* b, int64\_t ldb)

And

# Define the mkl\_solver\_symm function for symmetric matrices

def mkl\_solver\_symm(double[:, ::1] A, double[:, ::1] B):

"""

Solve a system of equations for symmetric matrices using MKL's DSYSV routine.

INPUTS:

- A: double array (n x n) with the symmetric coefficient matrix.

- 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 factorization.

B is overwritten with the values of the solution.

Returns:

- B: double array (n x nrhs) containing the solution to the system.

"""

cdef int64\_t lda, ldb, n, nrhs, matrix\_layout

cdef int64\_t[:] ipiv\_memview, i

cdef char uplo

matrix\_layout = 101 # Row major

uplo = b'L' # Lower triangular part of A (symmetric)

lda = A.shape[1]

ldb = B.shape[1]

n = A.shape[0]

nrhs = B.shape[1]

# Use numpy to create the memory for the ipiv input

ipiv\_memview = np.zeros(A.shape[0], dtype=np.int64)

# Call LAPACK function for symmetric matrices

LAPACKE\_dsysv(matrix\_layout, uplo, n, nrhs,

&A[0, 0], lda, &ipiv\_memview[0],

&B[0, 0], ldb)

return np.asarray(B)

With the above subroutine, we can execute the Assignment-4 “Ku=f” matrix and both execute the same results.