CS211 HW2 Hugo Wan twan012 862180666

Github Link: https://github.com/UCR-HPC/cs211-hw2-solving-large-linear-system-private-HugoWan0504

Q1 A = LU

A =	1	2	3	First pivot is A [1,1] = 1
	4	13	18	For row 2, the multiplier is A [2,1] / A[1,1] = 4 / 1 = 4
	7	54	78	Row 2 = Row 2 – 4 x Row 1

= [4 13 18] – 4 x [1 2 3]

= [4 13 18] – [4 8 12] = [0 5 6]

For row 3, the multiplier is A [3, 1] / A [1, 1] = 7 / 1 = 7

Row 3 = Row 3 – 7 x Row 1 = [7 54 78] – 7 x [1 2 3]

= [7 54 78] – [7 14 21] = [0 40 57]

New A =	1	2	3	Second pivot is A [2, 2] = 5
	0	5	6	For row 3, the multiplier is A [3, 2] / A [2, 2] = 40 / 8 = 5
	0	40	57	Update New A == Upper Triangular Matrix U

Row 3 = Row 3 – 8 x Row 2

= [0 40 57] – 8 x [0 5 6]

= [0 40 57] – [0 40 48] = [0 0 9]

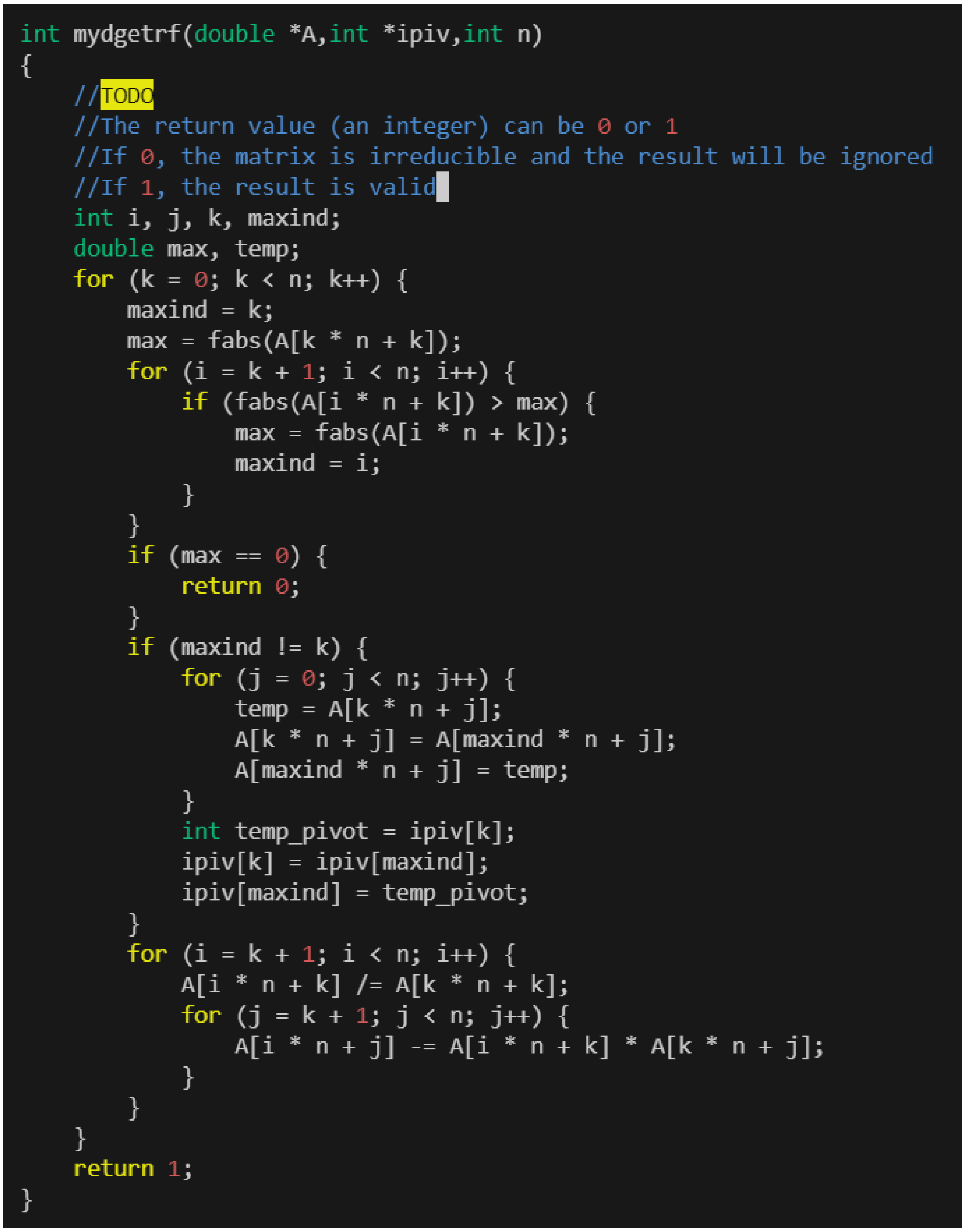
U =	1	2	3
	0	5	6
	0	0	9

L =	1	0	0	1st pivot =	1	0	0	2nd pivot =	1	0	0
	0	1	0		4	1	0	L =	4	1	0
	0	0	1		7	0	1		7	8	1

Result:	**A =**	**1**	**2**	**3**		**L =**	**1**	**0**	**0**		**U =**	**1**	**2**	**3**
A = LU		**4**	**13**	**18**	**=**		**4**	**1**	**0**	**x**		**0**	**5**	**6**
		7	**54**	**78**			**7**	**8**	**1**			**0**	**0**	**9**

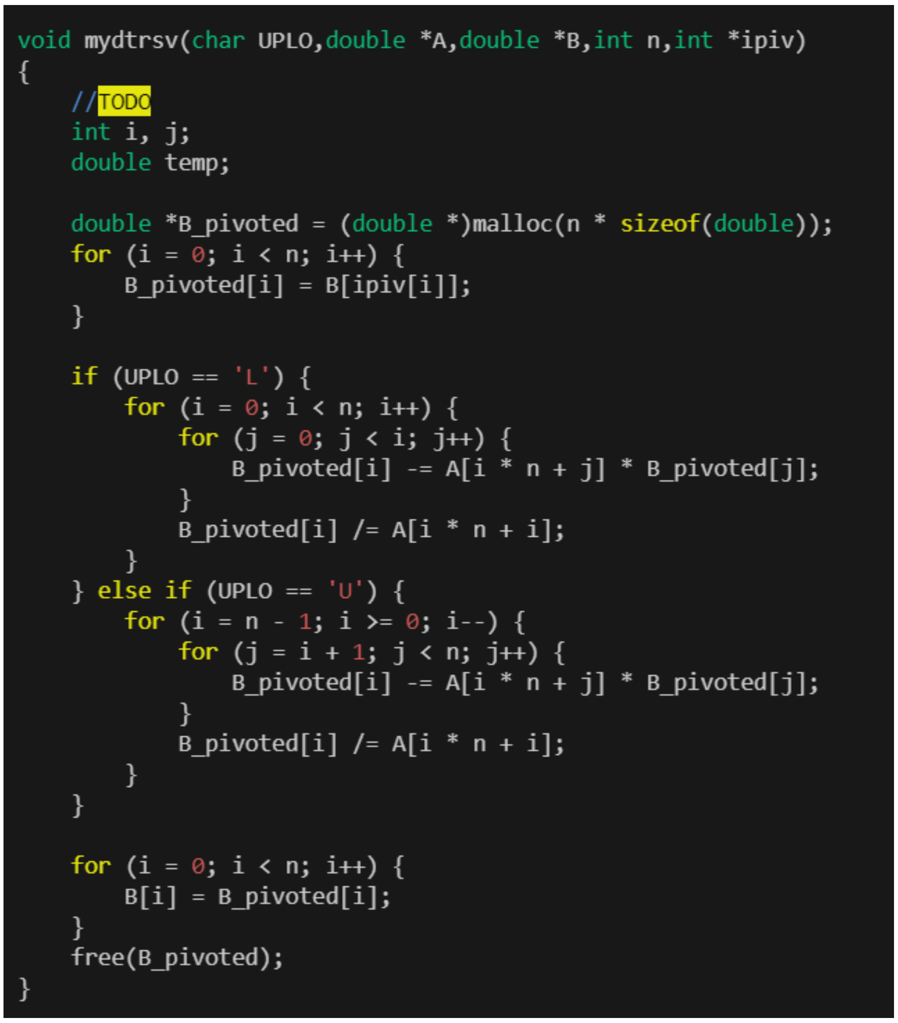
Q2

Function mydgetrf:



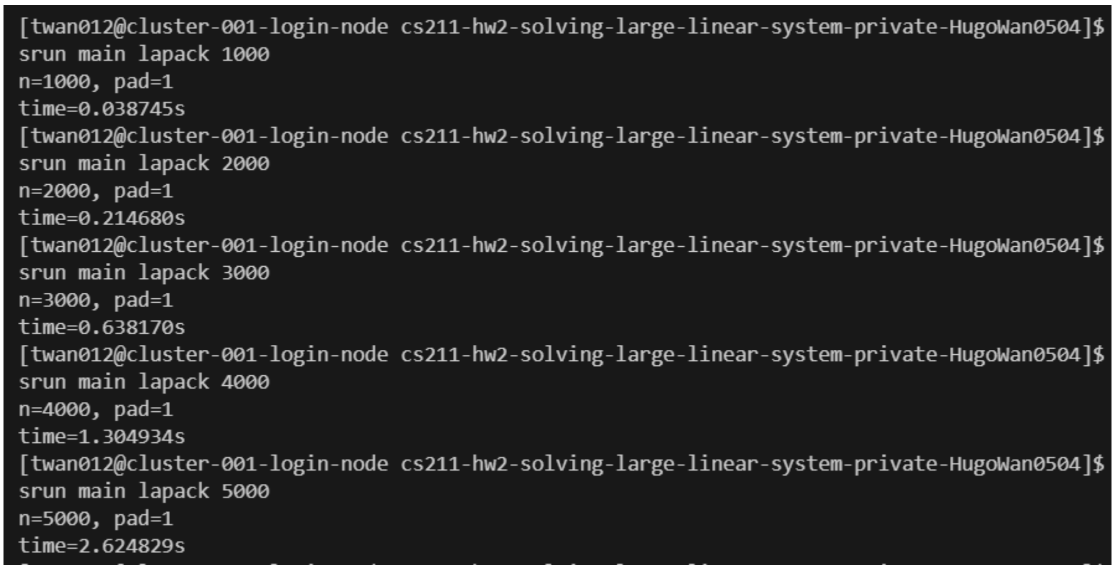
```c
int mydgetrf(double *A,int *ipiv,int n)
{
    //TODO
    //The return value (an integer) can be 0 or 1
    //If 0, the matrix is irreducible and the result will be ignored
    //If 1, the result is valid
    int i, j, k, maxind;
    double max, temp;
    for (k = 0; k < n; k++) {
        maxind = k;
        max = fabs(A[k * n + k]);
        for (i = k + 1; i < n; i++) {
            if (fabs(A[i * n + k]) > max) {
                max = fabs(A[i * n + k]);
                maxind = i;
            }
        }
        if (max == 0) {
            return 0;
        }
        if (maxind != k) {
            for (j = 0; j < n; j++) {
                temp = A[k * n + j];
                A[k * n + j] = A[maxind * n + j];
                A[maxind * n + j] = temp;
            }
            int temp_pivot = ipiv[k];
            ipiv[k] = ipiv[maxind];
            ipiv[maxind] = temp_pivot;
        }
        for (i = k + 1; i < n; i++) {
            A[i * n + k] /= A[k * n + k];
            for (j = k + 1; j < n; j++) {
                A[i * n + j] -= A[i * n + k] * A[k * n + j];
            }
        }
    }
    return 1;
}
```

Function mydtrsv:



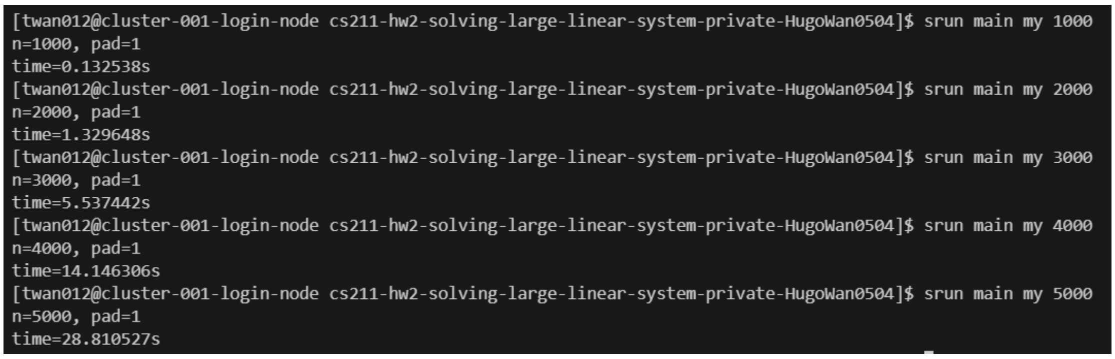
```c
void mydtrsv(char UPLO,double *A,double *B,int n,int *ipiv)
{
    //TODO
    int i, j;
    double temp;

    double *B_pivoted = (double *)malloc(n * sizeof(double));
    for (i = 0; i < n; i++) {
        B_pivoted[i] = B[ipiv[i]];
    }

    if (UPLO == 'L') {
        for (i = 0; i < n; i++) {
            for (j = 0; j < i; j++) {
                B_pivoted[i] -= A[i * n + j] * B_pivoted[j];
            }
            B_pivoted[i] /= A[i * n + i];
        }
    } else if (UPLO == 'U') {
        for (i = n - 1; i >= 0; i--) {
            for (j = i + 1; j < n; j++) {
                B_pivoted[i] -= A[i * n + j] * B_pivoted[j];
            }
            B_pivoted[i] /= A[i * n + i];
        }
    }

    for (i = 0; i < n; i++) {
        B[i] = B_pivoted[i];
    }
    free(B_pivoted);
}
```

Test outputs for srun lapack 1000 to 5000:



```
[twan012@cluster-001-login-node cs211-hw2-solving-large-linear-system-private-HugoWan0504]$
srun main lapack 1000
n=1000, pad=1
time=0.038745s
[twan012@cluster-001-login-node cs211-hw2-solving-large-linear-system-private-HugoWan0504]$
srun main lapack 2000
n=2000, pad=1
time=0.214680s
[twan012@cluster-001-login-node cs211-hw2-solving-large-linear-system-private-HugoWan0504]$
srun main lapack 3000
n=3000, pad=1
time=0.638170s
[twan012@cluster-001-login-node cs211-hw2-solving-large-linear-system-private-HugoWan0504]$
srun main lapack 4000
n=4000, pad=1
time=1.304934s
[twan012@cluster-001-login-node cs211-hw2-solving-large-linear-system-private-HugoWan0504]$
srun main lapack 5000
n=5000, pad=1
time=2.624829s
```

Test outputs for srun my 1000 to 5000 (without showing the errors):



```
[twan012@cluster-001-login-node cs211-hw2-solving-large-linear-system-private-HugoWan0504]$ srun main my 1000
n=1000, pad=1
time=0.132538s
[twan012@cluster-001-login-node cs211-hw2-solving-large-linear-system-private-HugoWan0504]$ srun main my 2000
n=2000, pad=1
time=1.329648s
[twan012@cluster-001-login-node cs211-hw2-solving-large-linear-system-private-HugoWan0504]$ srun main my 3000
n=3000, pad=1
time=5.537442s
[twan012@cluster-001-login-node cs211-hw2-solving-large-linear-system-private-HugoWan0504]$ srun main my 4000
n=4000, pad=1
time=14.146306s
[twan012@cluster-001-login-node cs211-hw2-solving-large-linear-system-private-HugoWan0504]$ srun main my 5000
n=5000, pad=1
time=28.810527s
```

Organized Table:

N	LAPACK Time (s)	MY Time (s)
1000	0.038745	0.132538
2000	0.214680	1.329648
3000	0.638170	5.537442
4000	1.304934	14.146306
5000	2.624829	28.810527

Calculations:

N	LAPACK Gflops	MY Gflops
1000	17.206521	5.030004
2000	24.843177	4.011087
3000	28.205651	3.250598
4000	32.696417	3.016100
5000	31.748100	2.892461

Compare the performance:

Higher Gflops indicate better performance. As the matrix size increases, the performance of LAPACK gradually improves, while the performance of MY decreases. Overall, LAPACK's performance is at least three times better than MY's algorithm.

Q3

a. Starting matrix: $A = \begin{array}{cccc} 1 & 2 & 3 & 4 \\ 2 & 9 & 12 & 15 \\ 3 & 26 & 41 & 40 \\ 5 & 40 & 107 & 135 \end{array}$

b. B = 2 → 2x2 block, and A = LU.

c. Step 1: Factorize the first block is A (0:1, 0:1).

d. $L = \begin{array}{cc} 1 & 0 \\ L21 & 1 \end{array}$ $U = \begin{array}{cc} U11 & U12 \\ 0 & U22 \end{array}$ U11 = A (0,0) = 1, L21 = A (1,0) / U11 = 2/1 = 2 $L = \begin{array}{cc} 1 & 0 \\ 2 & 1 \end{array}$

U12 = A (0,1) = 2 → $U = \begin{array}{cc} 1 & 2 \\ 0 & U22 \end{array}$ → U22 = A (1,1) – L21 x U22 = 9 – 2 x 2 = 9 – 4 = 5 → $U = \begin{array}{cc} 1 & 2 \\ 0 & 5 \end{array}$

e. A (0:1, 0:1) = L x U =

The decomposition: $\begin{array}{cc} 1 & 0 \\ 2 & 1 \end{array}$ x $\begin{array}{cc} 1 & 2 \\ 0 & 5 \end{array}$ = $\begin{array}{cc} 1 & 2 \\ 2 & 9 \end{array}$

f. Update the matrix: $A = \begin{array}{cccc} 1 & 2 & 3 & 4 \\ 2 & 5 & 12 & 15 \\ 3 & 26 & 41 & 49 \\ 5 & 40 & 107 & 135 \end{array}$ Update U22 9 → 5

a. Step 2: Update the Block Column Below (Row 2, 3 in Column 0, 1):

L31 = A (2,0)/U11 = 3/1 = 3 → A (2,0) = 3 U12 = A (0,1) = 2

L41 = A (3,0)/U11 = 5/1 = 5 → A (3,0) = 5

b. Update A (2,1): A (2,1) – L31 x U12 = 26 – 3 x 2 = 26 – 6 = 20

Update A (3,1): A (3,1) – L41 x U12 = 40 – 5 x 2 = 40 – 10 = 30

c. Now, the matrix becomes $A = \begin{array}{cccc} 1 & 2 & 3 & 4 \\ 2 & 5 & 12 & 15 \\ 3 & 20 & 41 & 49 \\ 5 & 30 & 107 & 135 \end{array}$

a. Step 3: Update the Block to the right (Column 2, 3 in Row 0, 1):

b. Update A (1,2): A (1,2) – A (1,0)/U11 x A (0,2) = 12 – 2 x 3 = 12 – 6 = 6

Update A (1,3): A (1,3) – A (1,0)/U11 x A (0,3) = 15 – 2 x 4 = 15 – 8 = 7

c. Matrix update:

$$A = \begin{matrix} 1 & 2 & 3 & 4 \\ 2 & 5 & 6 & 7 \\ 3 & 20 & 41 & 49 \\ 5 & 30 & 107 & 135 \end{matrix}$$

a. Step 4: Update the Bottom-Right Block (Row 2, 3 in Column 2, 3):

b. $L = \begin{matrix} 1 & 0 \\ L43 & 1 \end{matrix}$ $U = \begin{matrix} U33 & U34 \\ 0 & U44 \end{matrix}$ U33 = A (2,2) = 41

U34 = A (2,3) = 49

c. L43 = A (3,2)/U33 = 107/41 = 2.61 A (3,2) = 2.61

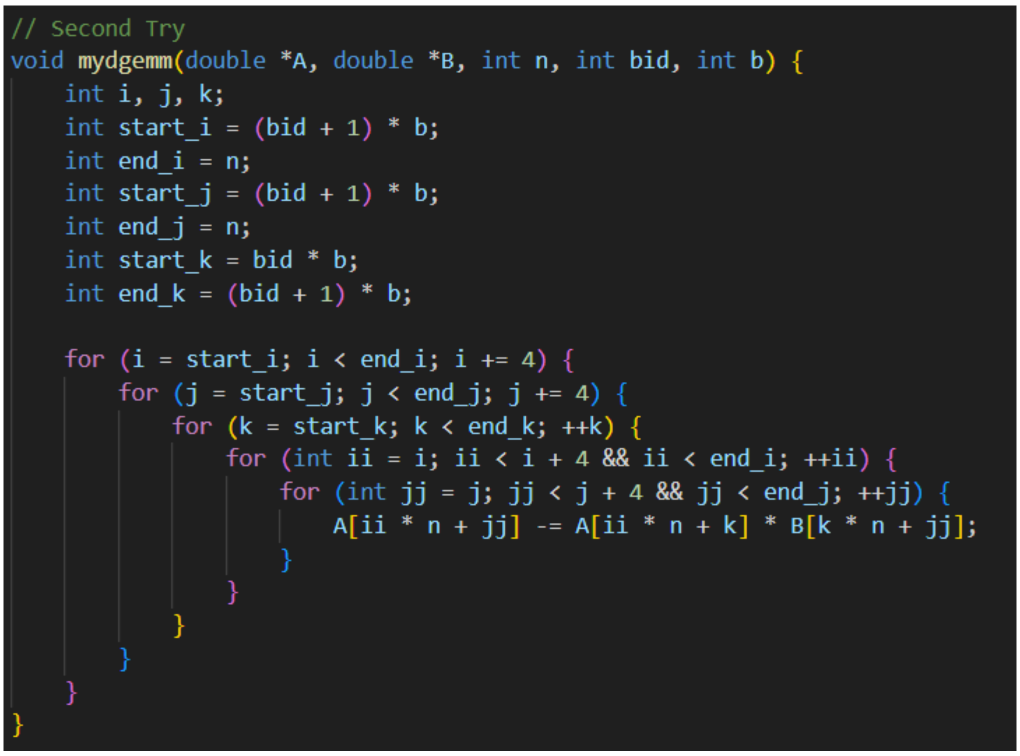
d. U44 = A (3,3) – L43 x U34 = 135 – 2.61 x 49 = 6.11 A (3,3) = 6.11

e. Final matrix is:

$$\mathbf{A =} \begin{matrix} \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} \\ \mathbf{2} & \mathbf{5} & \mathbf{6} & \mathbf{7} \\ \mathbf{3} & \mathbf{20} & \mathbf{41} & \mathbf{49} \\ \mathbf{5} & \mathbf{30} & \mathbf{2.61} & \mathbf{6.11} \end{matrix}$$

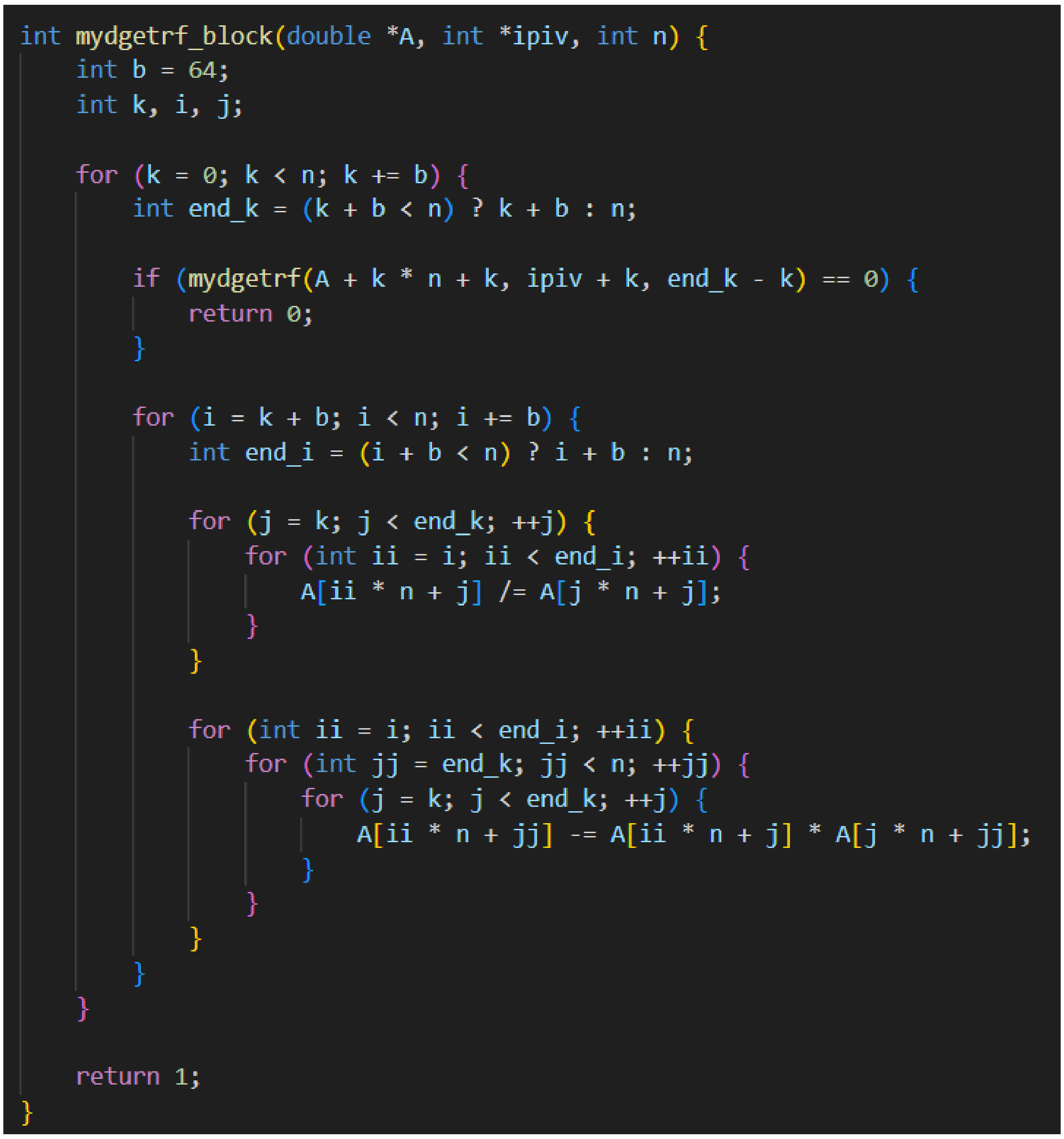
Q4 Best Performance Program:

Function mydgemm:



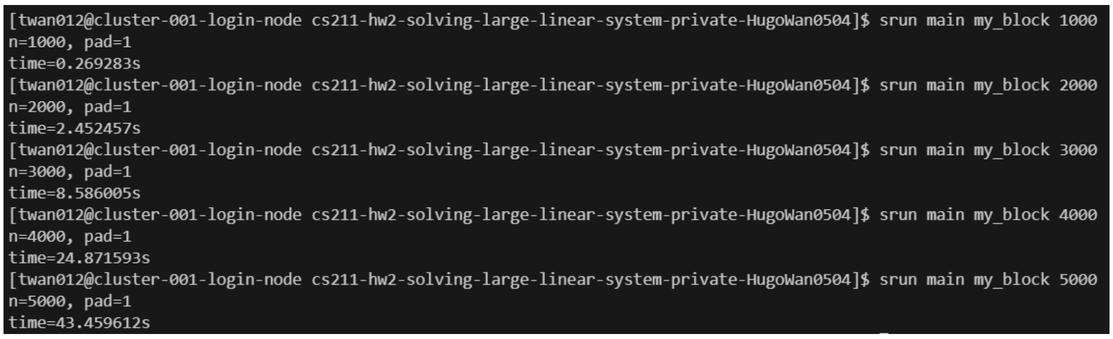
```c
// Second Try
void mydgemm(double *A, double *B, int n, int bid, int b) {
    int i, j, k;
    int start_i = (bid + 1) * b;
    int end_i = n;
    int start_j = (bid + 1) * b;
    int end_j = n;
    int start_k = bid * b;
    int end_k = (bid + 1) * b;

    for (i = start_i; i < end_i; i += 4) {
        for (j = start_j; j < end_j; j += 4) {
            for (k = start_k; k < end_k; ++k) {
                for (int ii = i; ii < i + 4 && ii < end_i; ++ii) {
                    for (int jj = j; jj < j + 4 && jj < end_j; ++jj) {
                        A[ii * n + jj] -= A[ii * n + k] * B[k * n + jj];
                    }
                }
            }
        }
    }
}
```

Function mydegtrf_block:



```c
int mydgetrf_block(double *A, int *ipiv, int n) {
    int b = 64;
    int k, i, j;

    for (k = 0; k < n; k += b) {
        int end_k = (k + b < n) ? k + b : n;

        if (mydgetrf(A + k * n + k, ipiv + k, end_k - k) == 0) {
            return 0;
        }

        for (i = k + b; i < n; i += b) {
            int end_i = (i + b < n) ? i + b : n;

            for (j = k; j < end_k; ++j) {
                for (int ii = i; ii < end_i; ++ii) {
                    A[ii * n + j] /= A[j * n + j];
                }
            }

            for (int ii = i; ii < end_i; ++ii) {
                for (int jj = end_k; jj < n; ++jj) {
                    for (j = k; j < end_k; ++j) {
                        A[ii * n + jj] -= A[ii * n + j] * A[j * n + jj];
                    }
                }
            }
        }
    }

    return 1;
}
```

- Test outputs when b = 64:



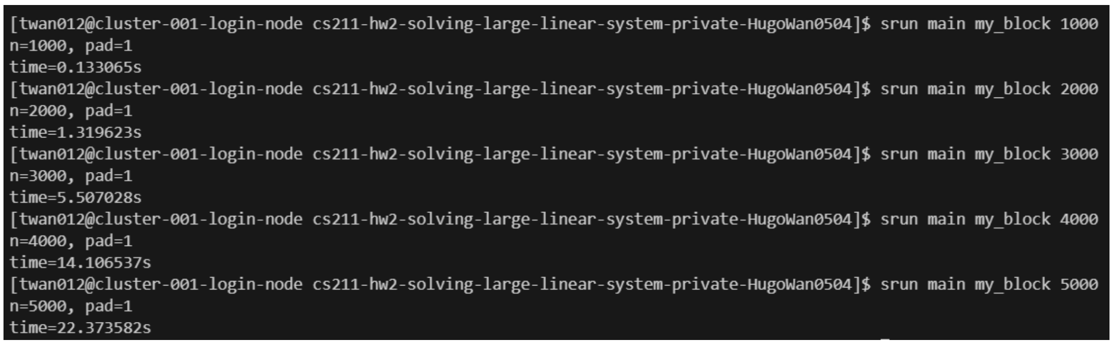
```
[twan012@cluster-001-login-node cs211-hw2-solving-large-linear-system-private-HugoWan0504]$ srun main my_block 1000
n=1000, pad=1
time=0.269283s
[twan012@cluster-001-login-node cs211-hw2-solving-large-linear-system-private-HugoWan0504]$ srun main my_block 2000
n=2000, pad=1
time=2.452457s
[twan012@cluster-001-login-node cs211-hw2-solving-large-linear-system-private-HugoWan0504]$ srun main my_block 3000
n=3000, pad=1
time=8.586005s
[twan012@cluster-001-login-node cs211-hw2-solving-large-linear-system-private-HugoWan0504]$ srun main my_block 4000
n=4000, pad=1
time=24.871593s
[twan012@cluster-001-login-node cs211-hw2-solving-large-linear-system-private-HugoWan0504]$ srun main my_block 5000
n=5000, pad=1
time=43.459612s
```

Organized Table when b = 64:

N	MY_BLOCK Time (s)	MY_BLOCK Gflops	LAPACK Gflops	MY Gflops
1000	0.269283	2.475710	17.206521	5.030004
2000	2.452457	2.174690	24.843177	4.011087
3000	8.586005	2.096435	28.205651	3.250598
4000	24.871593	1.715478	32.696417	3.016100
5000	43.459612	1.917489	31.748100	2.892461

Based on the table, it seems that the Gflops performance of my_block.c improved, but it's still significantly lower than my.c at all values of N, particularly at larger sizes.

- Test outputs when b = 4096:



```
[twan012@cluster-001-login-node cs211-hw2-solving-large-linear-system-private-HugoWan0504]$ srun main my_block 1000
n=1000, pad=1
time=0.133065s
[twan012@cluster-001-login-node cs211-hw2-solving-large-linear-system-private-HugoWan0504]$ srun main my_block 2000
n=2000, pad=1
time=1.319623s
[twan012@cluster-001-login-node cs211-hw2-solving-large-linear-system-private-HugoWan0504]$ srun main my_block 3000
n=3000, pad=1
time=5.507028s
[twan012@cluster-001-login-node cs211-hw2-solving-large-linear-system-private-HugoWan0504]$ srun main my_block 4000
n=4000, pad=1
time=14.106537s
[twan012@cluster-001-login-node cs211-hw2-solving-large-linear-system-private-HugoWan0504]$ srun main my_block 5000
n=5000, pad=1
time=22.373582s
```

Organized Table when b = 4096:

N	MY_BLOCK Time (s)	MY_BLOCK Gflops	LAPACK Gflops	MY Gflops
1000	0.133065	5.010083	17.206521	5.030004
2000	1.319623	4.041558	24.843177	4.011087
3000	5.507028	3.268551	28.205651	3.250598
4000	14.106537	3.024602	32.696417	3.016100
5000	22.373582	3.724631	31.748100	2.892461

Finally, my_block.c Gflops is on par with my.c Gflops when b = 4096. At N = 5000, my_block.c Gflops is significantly better than my.c's.