CS211 HW2: Solving Large Linear Systems

Due: 10/22/2024 23:59

Before Start:

1. Click the following link to accept your assignment on GitHub Classroom:

   https://classroom.github.com/a/viOw4eaV

   Clone the template code to your workspace on the **hpc-001** cluster (refer to the attached guide: **CS211 HPC Cluster.pdf** for instructions on how to log in). The core functions for this assignment are implemented in **my.c, my_block.c.** Your task is to complete the code in these files to ensure the project works correctly. Specifically, you need to implement LU factorization by completing the **mydgetrf**() function in **my.c**, and handle forward and backward substitution by finishing the **mydtrsm**() function in **my.c**. For bonus points, you can optimize your code with blocking techniques. The corresponding functions are in **my_block.c**. When testing different block sizes, you can modify the **pad.txt** file to avoid potential corner cases.

2. Project compilation: run the following commands step by step
   - **CURDIR=$(pwd)**
   - **export LD_LIBRARY_PATH=$CURDIR/extern:/act/opt/intel/composer_xe_2013.3.163/mkl/lib/intel64:$LD_LIBRARY_PATH**
   - **make**

3. Function test:
   - **./main {function name} {n}** to test on the header node
   - **srun main {function name} {n}** to test on the compute nodes
   - **starter.py** can be used to compile your project and test your codes. You can add more test cases within this file to automatically run your functions on either the header node or compute nodes

4. We will use **main.c** to measure running time and performance. Please do not modify the **main.c** file when submitting your code.

5. We will use **hpc-001** to test your performance during grading. You can test and debug your code on other computers, but ensure that it runs on **hpc-001** before submission.

Submission:

Following the instructions in the questions below, complete the corresponding code or theoretical analysis and answer the questions. Collect experiment data and compare the different optimization strategies when analyzing your results. You need to submit a **PDF** report for this assignment, and your code must be submitted on GitHub Classroom (push your final code to your repository in GitHub Classroom). To better align your report with your code, please upload a copy of your report to your GitHub repository as well.

Report requirements: Do not include the problem description in your report; the report should only contain answers to the questions, the corresponding experimental data, data analysis, and the theoretical reasoning process.

Report naming format: **cs211_hw2_firstname_lastname.pdf**. Please include the link to your GitHub repository, your name, and your SID on the first page of your report.

Q1 (10 points)

Perform a non-blocked LU factorization $\boldsymbol{A} = \boldsymbol{LU}$ for the following matrix $\boldsymbol{A}$ without pivoting:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 13 & 18 \\ 7 & 54 & 78 \end{bmatrix}$$

Your calculation process should show the intermediate state of $\boldsymbol{A}$ after each step of elimination.

Hint:

1. The non-zero elements of matrices L and U are positive integers, each no larger than 10.

Q2 (80 points)

Attached is a MATLAB program designed to solve the general linear system $\boldsymbol{Ax} = \boldsymbol{B}$ and verify the solution with the MATLAB build-in solver. Please review this implementation and incorporate its approach into your own program.

- Your task: complete **mydgetrf**() and **mydtrsv**() functions in **my.c** to perform the LU factorization, forward substitution, and backward substitution.

```
function mylu(n)

A=randn(n,n); b=randn(n,1); Abk=A; pvt = 1:n;
%Factorize A. Your task: transform this part to mydgetrf().
for i = 1 : n-1,
  % pivoting %
  maxind=i; max=abs(A(i,i));
  for t=i+1:n,
    if ( abs(A(t,i))>max )
      maxind=t; max=abs(A(t,i));
    end
  end
  if (max==0)
    disp ( 'LUfactoration failed: coefficient matrix is singular' ); return;
  else
    if (maxind ~= i )
      %save pivoting infomation
      temps=pvt(i);
      pvt(i)=pvt(maxind);
      pvt(maxind)=temps;
      %swap rows
      tempv=A(i,:);
      A(i,:)=A(maxind,:);
      A(maxind,:)=tempv;
    end
  end
  %factorization
  for j = i+1 : n,
    A(j,i) = A(j,i)/A(i,i);
    for k = i+1 : n,
      A(j,k) = A(j,k) - A(j,i) * A(i,k);
    end
  end
end
%verify my factorization with Matlab for small matrix by printing results on screen. May skip in your code
myfactorization=A
mypivoting=pvt
[Matlab_L, Matlab_U, Matlab_P]=lu(Abk)

%forward substitution. Your task: transform this part to mydtrsm().
y(1) = b(pvt(1));
for i = 2 : n,
  y(i) = b(pvt(i)) - sum ( y(1:i-1) .* A(i, 1:i-1) );
end
%  back substitution. Your task: transform this part to mydtrsm().
x(n) = y(n) / A(n, n);
for i = n-1 : -1 : 1,
  x(i) = ( y(i) - sum ( x(i+1:n) .* A(i, i+1:n) ) ) / A(i,i);
end
%Matlab solve. Your task: call dgetrf() to factorize and dtrsm() twice (back and forward substit.) to solve.
xx= Abk\b;
%verify my solution with matlab. Your task: verify your solution with the solution from LAPACK.
Solution_Difference_from_Matlab=norm(x'-xx)
```

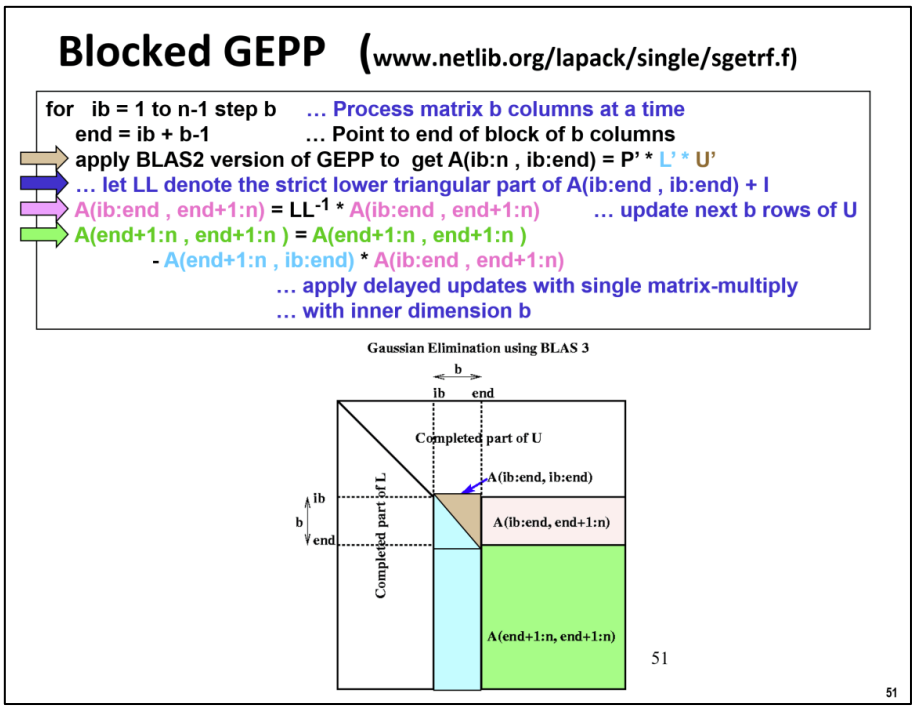
Hint:

1. The **lapack_f()** in **lapack.c** is utilized to verify the correctness of your implementation. The **lapack_f()** call the **LAPACKE_dgetrf()** in LAPACK (http://www.netlib.org/lapack) to perform the LU factorization on the coefficient matrix A and then call the function **cblas_dtrsv()** to perform the forward substitution first and then call it again to perform the backward substitution. The verification process is implemented in **main.c**. To test the LAPACK implementation, please run **./main lapack {n}**

2. Your C functions **mydgetrf()** and **mydtrsv()** should follow the same algorithm as the MATLAB code, although the implementation may differ. Test your implementation and LAPACK version with matrix size 1000, 2000, 3000, 4000, 5000. Compare the performance (i.e., Gflops) of the two approaches.

Q3 (10 points)

The following slide shows the blocked GEPP algorithm.



Perform this algorithm on the following matrix $\boldsymbol{A}$ with block size $\boldsymbol{b} = \mathbf{2}$ and without pivoting to achieve the LU factorization result $\boldsymbol{A} = \boldsymbol{LU}$:

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 9 & 12 & 15 \\ 3 & 26 & 41 & 49 \\ 5 & 40 & 107 & 135 \end{bmatrix}$$

Your calculation process should show the intermediate state of $\boldsymbol{A}$ after each step (line) of the blocked **GEPP** algorithm.

Hint:

1. The non-zero elements of matrices L and U are positive integers, each no larger than 10.

Q4 (20 points, Bonus Problem)

Implement the blocked GEPP algorithm **mydgetrf_block**.

- Solve the linear system through your blocked version code using your own matrix multiplication. Please complete **mydgemm()** and **mydgetrf_block()**.
- Optimize your code to achieve as high performance as possible using any other techniques available to you (such as changing block size). Compare your performance with your un-optimized version in Q2.