CS 211: High Performance Computing Project 2

High Performance Sequential Codes for Solving Large Linear Systems

Due at 11:59 PM on Oct 20th, 2017

Part 1 (50 points)

Attached is a Matlab program to solve the general linear system ***Ax=b*** and verify the solution with the Matlab build-in solver. Write a C/C++ /Fortran program to solve the linear system through the following two approaches and verify that the two approaches give the same solution:

(1). Call the function **dgetrf()** in LAPACK (http://www.netlib.org/lapack/) to perform the LU factorization of the coefficient matrix ***A*** and then call the function **dtrsm()** in LAPACK to perform the forward substitution first and then call it again to perform the backward substitution;

(2). Call the **mydgetrf()** and **mydtrsm()** implemented by yourself to perform the LU factorization, forward substitution, and backward substitution.

Your C/C++ /Fortran functions **mydgetrf()** and **mydtrsm()** must follow the same algorithm in the Matlab code. Do NOT perform any code optimization or use any compiler optimization flag. Test your codes with random matrices of size 1000, 2000, 3000, 4000, 5000 on TARDIS. Compare the performance (i.e., Gflops) of the two approaches.

Part 2 (50 points)

Implement the blocked GEPP algorithm in the lecture. Solve the linear system through your blocked version code and optimize your to achieve as high performance as possible using any other techniques available to you and compare your performance with your un-optimized version.

Note that, in syllabus, we emphasize for ALL homework assignments: “Please make sure that your programs are properly documented and indented. Provide instructions on how to run your programs, give example runs, and analyze your results.” Please test your code on TARDIS and submit all source codes and a detailed report in PDF format into blackboard.

```
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)
```