Math 4353/5393 – Numerical Linear Algebra Spring 2024
Program 4 Due Wednesday, May 1, 2024

1. Write the subroutine houseqr to compute the QR factorization. houseqr.m should have as its first line:

   function [A,u1] = houseqr(A)

   It should implement the Householder QR factorization without explicitly forming Q. The $\tilde{u}_k$ vectors should overwrite A in the strictly lower triangle, and the $\tilde{u}_k(1)$ scalars should be stored in $u1(k)$. R will be stored in the upper triangle of A.

2. Write the subroutine houseqtact to compute $Q^t b$, given the "decapitated" $\tilde{u}_1, \ldots, \tilde{u}_n$ stored in the strictly lower triangle of Qu and their "heads" in $u1$ (so $u1 = (\tilde{u}_1(1), \tilde{u}_2(1), \ldots, \tilde{u}_n(1))^T$. houseqact.m should have as its first line:

   function y = houseqtact(Qu,u1,b)

   It should implement the algorithm we discussed in class/handout to find $Q^T b$.

3. Hand in and email me the output, prog4run.txt, of the test routine NLAProg4test.m and your houseqr.m and houseqtact.m

   Notes:

   (a) For efficiency, we do not form Q explicitly when doing the Householder QR factorization, we just save the $\tilde{u}_i$ over the columns of A, but below R_1. This is why we need the HOUSEQTACT routine.

   (b) Please notice how this *admittedly opaque* storage scheme results in remarkably little data movement and efficient memory use.

   (c) Remember to document your code. This means using comment lines to describe all input and output variables, and to describe what the code is doing when it is not obvious to the uninitiated.

   (d) Don't divide by 0.

   (e) You might be interested to see how this method compares to your mgs and cgs code. If you have the interest, try running a test to see which, if either, of xqr or xmgs or xcgs is consistently closer to xtrue.