6. If `A` is symmetric (or Hermitian) and has real positive diagonal elements, a Cholesky factorization is tried. It uses CHOLMOD in the sparse case (either an up-looking algorithm much like `cs_chol` if `A` is very sparse or a supernodal algorithm otherwise). The matrix is first permuted via an approximate minimum degree ordering algorithm (see `cs_amd`). This algorithm uses most of the theory and algorithms presented in Chapters 2 through 4 and Chapter 7. In the full case, it uses LAPACK Cholesky factorization routines. This step is terminated early if the matrix is found not to be positive definite.

7. If `A` is a full Hessenberg matrix, it is reduced to an upper triangular matrix and the BLAS backsolve is used.

8. If `A` is square and sparse, UMFPACK is used to perform an LU factorization, `L*U=P*(R\A)*Q`, where `R` is a diagonal matrix that scales the rows of `A` (GPLU can be optionally used instead; it can be faster for very sparse matrices). UMFPACK selects one of three ordering strategies: COLAMD, AMD, or a cheap diagonal matching followed by AMD, depending on the matrix properties (how symmetric its nonzero pattern is and how many entries are on the diagonal). It uses its default pivot tolerances (0.001 for diagonal entries and 0.1 for off-diagonal entries). These tolerances lead to a much sparser factorization than partial pivoting but may result in a numerically unstable factorization. If $\min_i |u_{ii}| / \max_j |u_{jj}|$ is less than machine epsilon (about 2×10^{-16}), the matrix is factorized again using partial pivoting (a tolerance of 1.0). UMFPACK is a multifrontal method, based on the column elimination tree and a symbolic QR analysis. Iterative refinement with sparse backward