University of Toronto at Scarborough
CSCD37—Analysis of Numerical Algorithms for Computational Mathematics, Winter 2024

Assignment #1: QR factorization / LSPs

Due: February 12, 2024 at 11:45 p.m.
This assignment is worth 13% of your final grade.

Warning: Your electronic submission on *MarkUs* affirms that this assignment is your own work and no one else's, and is in accordance with the University of Toronto Code of Behaviour on Academic Matters, the Code of Student Conduct, and the guidelines for avoiding plagiarism in CSCD37.

This assignment is due by 11:45 p.m. February 12. If you haven't finished by then, you may hand in your assignment late with a penalty as specified in the course information sheet.

[15] 1. In CSCC37 we saw how to compute the $PA = LU$ factorization of $A \in \mathbb{R}^{n\times n}$. The algorithm involved pre-multiplying A with $n-1$ permutation matrices $\mathcal{P}_i$ interleaved with $n-1$ Gauss transforms $\mathcal{L}_i$, where $\mathcal{P}_i, \mathcal{L}_i \in \mathbb{R}^{n\times n}, i = 1, \ldots, n-1$. This transformation triangularizes A, producing the upper-triangular matrix $U \in \mathbb{R}^{n\times n}$. The unit-lower triangular matrix $L \in \mathbb{R}^{n\times n}$ is later computed by a sequence of algebraic manipulations on $\mathcal{P}_i, \mathcal{L}_i, i = 1, \ldots, n-1$.

(a) Show how $A \in \mathbb{R}^{m\times n}, m > n$ can be triangularized in a similar fashion using permutation matrices and Gauss transforms. Clearly specify the dimension and purpose of each $\mathcal{P}_i$ and $\mathcal{L}_i$ required, and the number of $\mathcal{P}_i$ and $\mathcal{L}_i$ required.
(**Note:** You *do not* need to show how to compute the final L.)

(b) Given that the $PA = LU$ factorization of $A \in \mathbb{R}^{m\times n}, m > n$ exists, **either** show how to use the factorization to solve the least squares problem $\min_x \|Ax - b\|_2$, **or** argue why the $PA = LU$ factorization **cannot** be used.

(c) Given that the $PA = LU$ factorization of $A \in \mathbb{R}^{m\times n}, m > n$ exists, **either** show how to use the factorization to solve the null space problem, **or** argue why the $PA = LU$ factorization **cannot** be used. (Recall that in the null space problem, $A \in \mathbb{R}^{m\times n}, m > n$ arises by transposing the original coefficient matrix in the underdetermined system.)

[10] 2. In lecture we proved that the Householder reflection

$$\mathcal{Q} = I - 2\frac{vv^T}{v^Tv}$$

is both orthogonal and symmetric for any $v \neq 0$, and we *claimed* that if we choose

$$v = x - \|x\|_2 e_1 \tag{1}$$

we have

$$Qx = \left[I - 2\frac{vv^T}{v^T v}\right] x = \|x\|_2 e_1 \qquad (2)$$

where e_1 is the first column of the identity matrix. We then used (2) to compute the QR-factorization of $A \in \mathbb{R}^{m\times n}, m > n,$ by embedding a sequence of Householder reflections in the identity matrix.

Prove that if we choose v as in (1), (2) holds.

[10] 3. A sequence of QR factorizations can be used to solve a general *Almost Block Diagonal* (ABD) linear system arising in the numerical solution of Boundary Value Problems (BVPs) for Ordinary Differential Equations (ODEs).

Consider the following $5n \times 5n$ ABD linear system:

$$\begin{bmatrix} \mathcal{B}_a & & & & \mathcal{B}_b \\ \mathcal{S}_0 & \mathcal{T}_0 & & & \\ & \mathcal{S}_1 & \mathcal{T}_1 & & \\ & & \mathcal{S}_2 & \mathcal{T}_2 & \\ & & & \mathcal{S}_3 & \mathcal{T}_3 \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} g \\ \phi_0 \\ \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix} \qquad (3)$$

where $\mathcal{B}_a, \mathcal{B}_b, \mathcal{S}_0, \mathcal{T}_0, \mathcal{S}_1, \mathcal{T}_1, \mathcal{S}_2, \mathcal{T}_2, \mathcal{S}_3, \mathcal{T}_3 \in \mathbb{R}^{n\times n}$ are given, $g, \phi_0, \phi_1, \phi_2, \phi_3 \in \mathbb{R}^n$ are given, and we must solve for the unknowns $y_0, y_1, y_2, y_3, y_4 \in \mathbb{R}^n$.

(a) Show how a sequence of $2n \times n$ QR factorizations can be used to reduce (3) to a $2n \times 2n$ dense linear system involving only the unknowns y_0 and y_4. You may assume the QR factorizations exist; there are no numbers here.

(b) Show how y_1, y_2 and y_3 can be recovered, assuming y_0 and y_4 have been computed as in (a).

Hint: Your first QR factorization is performed on the submatrix $\begin{bmatrix} \mathcal{T}_0 \\ \mathcal{S}_1 \end{bmatrix}$. Also, you may find Chapter 2, §2.2.1 and §2.2.3 of *http://www.cs.toronto.edu/pub/reports/na/pancer-06-phd.pdf* useful when answering this question.

[15] 4. Consider the overdetermined linear system $Ax = b$ where

$$A = \begin{bmatrix} 2 & 2 \\ 1 & 1 \\ 2 & 2 \end{bmatrix}, \quad b = \begin{bmatrix} 3 \\ 6 \\ 9 \end{bmatrix}.$$

(a) Compute the QR-factorization of A using Householder reflections. Show *all* intermediate calculations. For example, the first reflection is given by

$$Q_1 = I - 2\frac{vv^T}{v^T v}, \quad v = a_1 - \|a_1\|_2 e_1,$$

where a_1 is the first column of A. Verify that $Q = Q_1^T Q_2^T$ is orthogonal.

(b) Use the QR-factorization computed in (a) to solve the least squares problem $\min_x \|Ax - b\|_2$. **Hint**: This problem is *rank-deficient*—there are infinitely many solutions. For full marks, you must compute the *minimum-norm* solution.

[25] 5. Write a *MatLab* function that computes a polynomial approximation to a user-defined function. Your function should be invoked as:

```
function [c,flag] = approx(f,a,b,n,m)
```

where

- `f` is the user-defined function to be approximated (see MatLab's `feval` for an explanation of how to pass function names as parameters),
- `[a,b]` is the interval on which `f` is to be approximated,
- `n` is the degree of the approximating polynomial,
- `m+1` is the number of interpolation conditions to be used in setting up the overdetermined linear system (see below),
- `c` is a vector containing the `n+1` coefficients of the approximating polynomial, and
- `flag` indicates whether the approximating polynomial is unique—`flag` $= 0 \Rightarrow$ the polynomial is unique; `flag` $\neq 0 \Rightarrow$ the polynomial is not unique and the vector `c` is not computed (see below).

The objective of your function is to find the coefficients c_k of an n-th degree polynomial

$$p(x) = \sum_{k=0}^{n} c_k x^k$$

such that $p(x) \approx f(x)$ on the interval $[a,b]$. We'll use the least-squares approach to determine the $p(x)$ that "fits best". Let $t_j = a + (b-a)j/m, j = 0,1,\ldots,m$. First set up the overdetermined linear system $Vc = g, V \in \mathbb{R}^{(m+1)\times(n+1)}, g \in \mathbb{R}^{m+1}, c \in \mathbb{R}^{n+1}$, representing the $m+1$ interpolation conditions $p(t_j) = f(t_j), j = 0,1,\ldots,m$. Next compute $c^* = \min_c \|Vc - g\|_2$ using the QR-based method described in lecture. Use MatLab's QR function to obtain the QR-factorization of V.

As shown in lecture, the uniqueness of c^* depends on the rank of V (or equivalently on the rank of R in the QR-factorization of V.) Use MatLab's `rank` function to determine matrix rank. If the solution is not unique due to near rank-deficiency in the Vandermonde matrix, **do not** attempt to compute c^*. Simply set `flag` $\neq 0$ and return. Not handling the nearly rank-deficient case simplifies the code considerably. However, you need to understand the nearly rank-deficient case as it may appear on the term test or final exam. Note that there cannot be exact rank-deficiency in this problem. Why not?

Note: *Do not* use the normal equations method for this question.

Test your function thoroughly on full-rank problems. Select a reasonable subset of your tests to submit for marking. Also, in order to give the marker a common problem to look at for all assignments, you must submit a test approximating $\cos(x), x \in [0,\pi]$ with a 4-th degree polynomial using 21 interpolation points ($m = 20$). For this test, submit

- a table listing $p(x_i), \cos(x_i)$, and the relative error $|p(x_i) - cos(x_i)|/|cos(x_i)|$, for sample points $x_i = (0.01)i\pi; i = 0,1,\ldots,100; i \neq 50$, and
- a plot of $p(x)$ and $\cos(x), x \in [0,\pi]$ on a single graph (use MatLab's `plot` function).

[total: 75 marks]