MATH 543 - HW 3

Parham Khodadi

February 19, 2026

1 Problem statement

Implement the reduced QR-factorization by Classical Gram-Schmidt.

1.1 Write a function.

Given $A \in \mathbb{C}^{m\times n}$ the function computes $Q \in \mathbb{C}^{m\times n}$, and $R \in \mathbb{C}^{n\times n}$.

1.2 Validate your function.

Test that:

- (i) $||A - QR|| \approx 0$
- (ii) Q is unitary, meaning $Q^TQ = I$
- (iii) R is upper triangular

Show 3 test cases for (3×3), (5×5), and (251×251) matrices.

1.3 Compare the result.

Look closer the the results for the (3×3) and (5×5) cases and compare with the built-in ("library") version of the QR-factorization; comment on the similarities/differences.

1.4 Can you find a non-zero matrix where your QR-factorization breaks?

2 Answer and Output

3 Code and Output.

My MATLAB code is found in Section 5.1. Its output is as follows:

```
------ Test size: 3 x 3 ------
||A - Q*R||_F      = 0.0000e+00
||Q'*Q - I||_F     = 1.0969e-15
||tril(R,-1)||_F   = 0.0000e+00
--- MATLAB qr(A,0) ---
||A - Qm*Rm||_F                        = 4.338e-16
Sign-insensitive ||abs(Q)-abs(Qm)||_F  = 7.156e-16
Sign-insensitive ||abs(R)-abs(Rm)||_F  = 6.866e-16

------ Test size: 5 x 5 ------
||A - Q*R||_F      = 1.5701e-16
```

```
||Q'*Q - I||_F     = 6.8922e-15
||tril(R,-1)||_F   = 0.0000e+00
--- MATLAB qr(A,0) ---
||A - Qm*Rm||_F                         = 1.070e-15
Sign-insensitive ||abs(Q)-abs(Qm)||_F  = 4.666e-15
Sign-insensitive ||abs(R)-abs(Rm)||_F  = 1.727e-15

------ Test size: 251 x 251 ------
||A - Q*R||_F      = 7.4948e-14
||Q'*Q - I||_F     = 1.3194e-11
||tril(R,-1)||_F   = 0.0000e+00

------ Part 4: CGS break example (Hilbert matrix) ------
||A - Q*R||_F  = 1.300e-16
||Q'*Q - I||_F = 4.515e+00
```

3.1 Why I wrote the function the way I did.

I simply wrote the Classical Gram-Schmidt algorithm in MATLAB exactly as written on slide 23 of Lecture Notes #6. The only difference was that I had to account for the possibility of dividing by zero if $R_{kk} = 0$ by manually setting those points in Q_{ik} to 0.

3.2 Result Analysis.

3.2.1 Part 2.

As you can see in the output, the validity of test cases was done using the Frobenius norm. I chose to do this because a simple google search will tell you that if you stack a matrix into one long vector, the Frobenius norm is exactly the Euclidean norm of that vector:

$$||E||_F = \sqrt{\sum_{i,j} |e_{ij}|^2}$$

For each test criteria:

- (i) $||A - QR|| \approx 0$ became progressively worse as the size n increased.
- (ii) Orthogonality $Q^TQ = I$ also worsened with increasing size n, although still very small and close to zero.
- (iii) R's upper triangular status consistently remained true across all three cases.

3.2.2 Part 3.

When comparing test cases (3×3) and (5×5) in CGS and in the built-in MATLAB-library algorithm, it appears that CGS very slightly outperforms. However, the difference is increasibly subtle, on the order of 10^{-15} and 10^{-16}, meaning that the difference is neglibile.

3.2.3 Part 4.

As you can see, I used a Hilbert matrix to push Classical Gram-Schmidt to its limit. I asked ChatGPT, "What is a type of matrix in the MATLAB library that would quickly make Classical Gram-Schmidt lose orthogonality." You can see its response in Figure 1. ChatGPT was correct, as a (12×12) Hilbert matrix quickly lost orthogonality.

4 Conclusion

For random matrices, Classical Gram-Schmidt produces a small residual $||A-QR||$ and nearly orthonormal Q. For an ill-conditioned matrix, such as a Hilbert matrix, Classical Gram-Schmidt suffers loss of orthogonality due to large $||Q^TQ - I||$, illustrating the known numerical instability of Classical Gram-Schmidt due to roundoff errors.

5 Appendix

5.1 MATLAB code

Listing 1: My MATLAB Code

```
%% MATH 543 - Homework 3
% By Parham Khodadi
clear; clc; close all;

%% Classical  G r a m Schmidt  Function
% Based on page 23 of Notes #6
function [Q,R] = qr_cgs(A)
    [m,n] = size(A);
    Q = zeros(m,n,class(A));
    R = zeros(n,n,class(A));

    for k = 1:n
        v = A(:,k);

        for i = 1:k-1
            R(i,k) = Q(:,i)' * A(:,k);
            v = v - R(i,k) * Q(:,i);
        end

        R(k,k) = norm(v,2);

        % Avoid dividing by zero when setting Q(:,k) to v/R(k,k)
        if R(k,k) == 0
            Q(:,k) = zeros(m,1,class(A));
        else
            Q(:,k) = v / R(k,k);
        end
    end
end

%% Validation

sizes = [3,5,251];

for s = sizes
    fprintf('\n------ Test size: %d x %d ------\n', s,s);
    A = rand(s);
    [Q, R] = qr_cgs(A);

    % (i) A - Q*R = 0
    err_factor = norm(A - Q*R, 'fro');
```

```
42
43      fprintf('||A - Q*R||_F      = %.4e\n', err_factor);
44
45      % (ii) Q'*Q = I
46      err_orth = norm(Q'*Q - eye(s), 'fro');
47      fprintf('||Q''*Q - I||_F     = %.4e\n', err_orth);
48
49      % (iii) strictly-lower part = 0
50      err_upper = norm(tril(R,-1), 'fro');
51      fprintf('||tril(R,-1)||_F   = %.4e\n', err_upper);
52
53      % Part 3
54      if s <= 5
55          fprintf('--- MATLAB qr(A,0) ---\n');
56
57          [Qm,Rm] = qr(A,0);
58
59          err_factor_matlab = norm(A - Qm*Rm, 'fro');
60          fprintf('||A - Qm*Rm||_F                          = %.3e\n',
                err_factor_matlab);
61
62          err_Q_diff = norm(abs(Q) - abs(Qm), 'fro');
63          fprintf('Sign-insensitive ||abs(Q)-abs(Qm)||_F  = %.3e\n',
                err_Q_diff);
64
65          err_R_diff = norm(abs(R) - abs(Rm), 'fro');
66          fprintf('Sign-insensitive ||abs(R)-abs(Rm)||_F  = %.3e\n',
                err_R_diff);
67      end
68  end
69
70
71  %% Find a non-zero matrix where CGS breaks
72  fprintf('\n------ Part 4: CGS break example (Hilbert matrix) ------\n');
73
74  n = 12;
75  A = hilb(n);
76
77  [Q,R] = qr_cgs(A);
78  fprintf('||A - Q*R||_F  = %.3e\n', norm(A - Q*R,'fro'));
79  fprintf('||Q''*Q - I||_F = %.3e\n', norm(Q'*Q - eye(n),'fro'));
```

5.2 ChatGPT Question

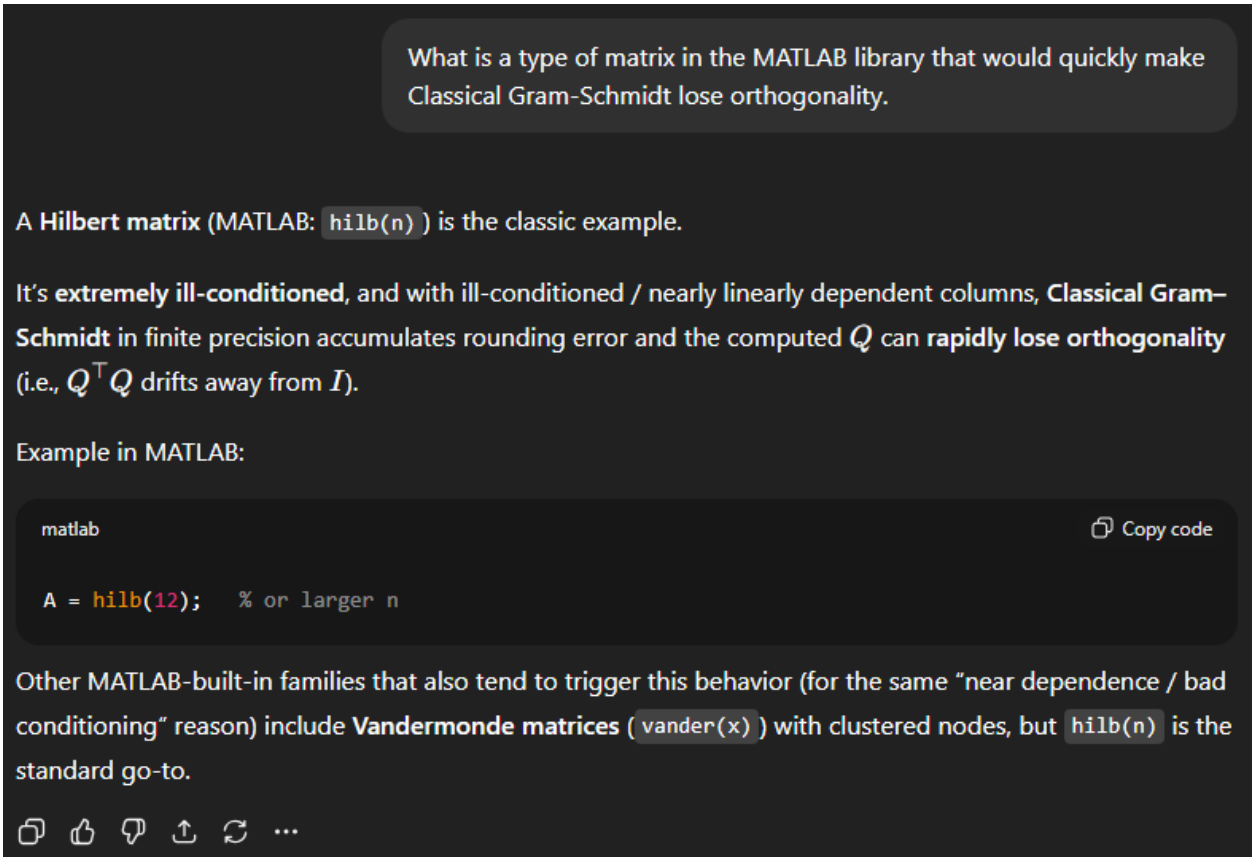

Figure 1: Asked ChatGPT: "What is a type of matrix in the MATLAB library that would quickly make Classical Gram-Schmidt lose orthogonality."