Multi-Step Problem

Table of Contents

Setup

```
format long;
```

Part (a):

```
% In this part of the problem we will define the 5x5 Hilbert Matrix, we can
% do this by hand or by using hilb(n). In our case this is hilb(5).

A = hilb(5);

% We will also use the provided code to apply the classical Gram-Schmidt
% process to the above matrix. We will output the Q matrix to the Command
% Window.

% Run the classical Gram-Schmidt process:
[n,k] = size(A);
Q = zeros(n,k);

for j = 1:k
    v_j = A(:,j);
    for p = 1:j-1
        v_j = v_j - (transpose(Q(:,p))*A(:,j))*Q(:,p);
    end
    if norm(v_j,2)==0
        error('Matrix A is not full rank.');
    end
    Q(:,j) = v_j/norm(v_j,2);
end

% Output the Q matrix to the Command Window:
Q
```

```
Q =

  Columns 1 through 3
```

```
   0.826584298073692  -0.533354625741032   0.175305353859225
   0.413292149036846   0.374053540533383  -0.717262398141420
   0.275528099357897   0.462945978156888  -0.057664733696256
   0.206646074518423   0.443319111628251   0.352625606033429
   0.165316859614738   0.405913757574913   0.571955108009368

  Columns 4 through 5

  -0.039102073788536   0.005504737805264
   0.403345206656191  -0.110094734308526
  -0.678964622494018   0.495426233019006
  -0.206153706721816  -0.770662945733810
   0.576447189861588   0.385331440514156
```

Part (b):

```
% In this part, we define a notion of error for the classical Gram-Schmidt
% process as the maximum entry in absolute value of the matrix Q^T Q - I_n.
% We will compute and display this error in the command window. Based on
% this value we will find out roughly how many decimal places we can trust
% the Q obtained in a.

% Compute the matrix Q^T Q - I_5:
Z = (transpose(Q) * Q) - eye(5);

% Compute and display the maximum entry in absolute value of the matrix
% Q^T Q - I_n. To do this we will compute the maximum aboslute value entry
% in each row, then the maximum in the resulting row vector.
max(max(abs(Z)))

% Based on this value, we can trust the Q obtained in (a) by up to roughly 7
% decimal places.
```

```
ans =

     6.635477955985181e-08
```

Part (c):

```
% In this part, we will apply the modified Gram-Schmidt algorithm to the
% Hilbert matrix. We will then display the resulting Q matrix to the
% command window.

% Run the modified Gram-Schmidt process:
[n,k] = size(A);
for j = 1:k
    if norm(A(:,j),2)==0
        error('Matrix A is not full rank.');
    end
    A(:,j) = A(:,j)/norm(A(:,j),2);
```

```
    for p = j+1:k
        A(:,p) = A(:,p) - (transpose(A(:,j))*A(:,p))*A(:,j);
    end
end
Q = A;

% Output the Q matrix to the Command Window:
Q

% As can be seen from the following output, the Q matrix from the modified
% Gram-Schmidt is very similar to that of the classical Gram-Schmidt, with
% only minor differences farther down the numbers.


Q =

  Columns 1 through 3

   0.826584298073692  -0.533354625741032   0.175305353859193
   0.413292149036846   0.374053540533383  -0.717262398141397
   0.275528099357897   0.462945978156888  -0.057664733696227
   0.206646074518423   0.443319111628251   0.352625606033457
   0.165316859614738   0.405913757574913   0.571955108009393

  Columns 4 through 5

  -0.039102073792843   0.005504735413709
   0.403345206675991  -0.110094708423228
  -0.678964622492973   0.495426187907808
  -0.206153706732331  -0.770662958963769
   0.576447189844912   0.385331479484320
```

Part (d):

```
% In this part, we will apply the notion of error defined in part (b) for
% the above matrix. With this error value, we will again be able to say how
% may decimal places we can trust the Q obtained in part c.

% Compute the matrix Q^T Q - I_n:
Z = (transpose(Q) * Q) - eye(5);

% Compute and display the maximum entry in absolute value of the matrix
% Q^T Q - I_n.
max(max(abs(Z)))

% Based on this value, we can trust the Q obtained in (a) by roughly
% 11 decimal places.


ans =

     6.200845392712040e-12
```

Part (e):

```
% In this part, we will compute the condition number of the matrix A using
% cond(A) in order to see "how dependent" the columns of A are. We will
% then use this condition number to find approximately how many significant
% digits we will lose performing the modified Gram-Schmidt on A.

% Redefine the matrix A to ensure we are using the correct matrix in
% computations:
A = hilb(5);

% Compute and display the condition number of A:
cond_num = cond(A);
cond_num

% Compute and display the significant digits lost:
k = log10(cond_num);
k

% Based on computing the rule-of-thumb, we expect to lose approzimately 6
% significant digits (rounded) when perfroming Gram-Schmidt on the 5x5
% Hilbert matrix. Notice that the modified Gram-Schmidt algorithm lost
% approximately 5 digits, this is very close to the number we got using the
% rule of thumb calculation above which was approximately 5.68.
```

```
cond_num =

     4.766072502422425e+05


k =

   5.678160644632865
```

Published with MATLAB® R2023a