Michigan State University

CMSE 823 – Numerical Linear Algebra **Spring 2020**
Homework 4
Alexander Harnisch

Hey He!

I choose to use Python for the coding assignments. I'm sure you're familiar with how to execute Python scripts. For this assignment everything should be very self explanatory. Just run the *qr.py* script and it should generate the following output:

```
Case 1:
-------
Check for correctness:
Classical Gram-Schmidt: ||QR - A|| = 1.4936523181711914e-15
Modified Gram-Scmidt: ||QR - A|| = 1.1801832636420706e-15
Householder Transform based: ||QR - A|| = 2.5414166642833352e-15
Numpy: ||QR - A|| = 2.739800093299453e-15

Check orthogonality
Classical Gram-Schmidt: ||Q*Q - I|| = 4.292074685283655e-15
Modified Gram-Scmidt: ||Q*Q - I|| = 1.0481175698739523e-15
Householder Transform based: ||Q*Q - I|| = 9.940313666921538e-16
Numpy: ||Q*Q - I|| = 3.4625011052432235e-16

Case 2:
-------
Check for correctness:
Classical Gram-Schmidt: ||QR - A|| = 0.0
Modified Gram-Scmidt: ||QR - A|| = 0.0
Householder Transform based: ||QR - A|| = 2.7194799110210365e-16
Numpy: ||QR - A|| = 2.7194799110210365e-16

Check orthogonality
Classical Gram-Schmidt: ||Q*Q - I|| = 3.2547231622285317e-11
Modified Gram-Scmidt: ||Q*Q - I|| = 3.2547231622285317e-11
Householder Transform based: ||Q*Q - I|| = 0.0
Numpy: ||Q*Q - I|| = 2.3551386880256624e-16
```

There is not a lot to observe except for the very obvious: All methods seem to work correctly and generate the expected result. Even though they are not directly comparable for the Householder method, because it is the only one that constructs a full QR decomposition while the others only construct reduced decompositions.

As expected we see in both cases that the householder transform method (and NumPy's built in one, which I believe uses householder most of the time) is more accurate (the norms are closer to zero by one order of magnitude).

In the first case the modified Gram-Schmidt implementation seems to be slightly better than the Classical, but the difference is not significant. However, with matrices that small and only two cases it is very hard to make any statistically significant observations.

Maybe one more thing to observe: Since the Gram-Schmidt based implementations do not touch the lower triangular part of R, they stay exactly zero. With the householder method they are numerical zeros which are some small number very close to zero due to rounding errors. It would be legitimate to set these values to exactly zero, since we know from theory that they are indeed zero. However, I have not implemented that here.

Additional comment: I did not print the actual results here, just looked at them during debugging in the console. The assignment says "compute the results", not sure if you want us to explicitly state Q and R here, I don't see the point. Please let me know if that's what you want!

Also, more of a question than a comment: In the book Algorithm 10.1 states the expression:

$$v_k(v_k^* A_{k:m,k:n}) \tag{1}$$

I figured out what it means, but technically this is not a mathematically correct expression is it? $v_k^* A_{k:m,k:n}$ is a column vector and so is v_k. This is not a valid outer or inner product. The expected result here is a matrix where v_k is multiplied with each entry of the vector on the right to produce a matrix with the products in each column, am I right? At least that's how I have implemented it and it gives the correct result.