COMPSCI 527 — Homework 2

Due on September 24, 2015

Work on this assignment either alone or in pairs. You may work with different partners on different assignments, but you can only have up to one partner for any one assignment. You may not talk about this assignment with others until all of you have handed in their work. See Mechanics→Homework on the class web page for details on the homework policy.

Hand in your work as explained in the instructions in homework 1 (of course, change `hw1` to `hw2`).

1. Let

$$A = \begin{bmatrix} \mathbf{a}_1 & \dots & \mathbf{a}_n \end{bmatrix}$$

be an $m \times n$ matrix. The Gram-Schmidt procedure transforms A into an orthogonal $m \times r$ matrix

$$Q = \begin{bmatrix} \mathbf{q}_1 & \dots & \mathbf{q}_r \end{bmatrix}$$

whose range is the same space as the range of A. This procedure can be summarized by the following algorithm given in the class notes:

$r = 0$
for $j = 1$ to n
  $\mathbf{a}'_j = \mathbf{a}_j - \sum_{i=1}^{r} (\mathbf{q}_i^T \mathbf{a}_j) \mathbf{q}_i$
  if $\|\mathbf{a}'_j\| \neq \mathbf{0}$
    $r = r + 1$
    $\mathbf{q}_r = \frac{\mathbf{a}'_j}{\|\mathbf{a}'_j\|}$
  end
end

The convention is made here that a sum $\sum_{i=a}^{b}$ where $a > b$ evaluates to zero.

(a) Apply Gram-Schmidt to find an *exact* orthogonal basis

$$Q = \begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 & \mathbf{q}_3 \end{bmatrix}$$

for the space spanned by the columns of the following matrix:

$$A_0 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}.$$

"Exact" here means that numerical values should not be approximated. For instance, $1/\sqrt{2}$ should stay that, or perhaps be written as $\sqrt{2}/2$ (your choice), but it should not be approximated with, say, 0.7071. So you cannot use MATLAB for this problem, but rather do the calculations by hand. Show your calculations, and simplify expressions as much as possible.

(b) Check, either exactly by hand or approximately by numerical calculation in MATLAB, that the matrix Q you found is orthogonal. Show either your calculations or your code and its output.

(c) When applying Gram-Schmidt to an arbitrary matrix A, what is the meaning of the final value of r?

(d) Can you use Gram-Schmidt to compute the rank of an arbitrary matrix A, assuming perfect arithmetic (no numerical errors)? Briefly describe how to do this, or explain why this is not possible.

(e) How can you use Gram-Schmidt to determine whether the following linear system admits a solution? Explain briefly but clearly, and assume perfect arithmetic. You need *not* (yet) give a method to find a solution if one exists.

$$A\mathbf{x} = \mathbf{b}$$

(f) Assume that the n columns of A are linearly independent, so that the test in the `if` statement in Gram-Schmidt always succeeds. Then, the Gram-Schmidt procedure simplifies as follows:

for $j = 1$ to n
  $\mathbf{a}'_j = \mathbf{a}_j - \sum_{i=1}^{j-1} (\mathbf{q}_i^T \mathbf{a}_j) \mathbf{q}_i$
  $\mathbf{q}_j = \frac{\mathbf{a}'_j}{\|\mathbf{a}'_j\|}$
end

and we can write the two assignments in the loop as a single expression as follows

$$r_{jj} \mathbf{q}_j = \mathbf{a}_j - \sum_{i=1}^{j-1} r_{ij} \mathbf{q}_i \ ,$$

that is,

$$\mathbf{a}_j = \sum_{i=1}^{j} r_{ij} \mathbf{q}_i \ . \tag{1}$$

Write expressions for r_{jj} and r_{ij} for $i \neq j$ in terms of $\mathbf{a}'_j$, $\mathbf{q}_i$, and $\mathbf{a}_j$.

(g) If you collect the numbers r_{ij} (regardless of whether $i = j$ or not) from the previous question into a matrix R so that r_{ij} is in row i, column j, then R has a special structure that can be described by its *fill pattern*. The fill pattern of a matrix R is a matrix of the same size as R that has an asterisk where R might have a nonzero value. The fill pattern of R has nothing in entries where R is always zero. For instance, the fill pattern of the 4×4 identity matrix is

$$\begin{bmatrix} * & & & \\ & * & & \\ & & * & \\ & & & * \end{bmatrix} .$$

Show the fill pattern for a 4×4 matrix R resulting from the computations in your answer to the previous question.

(h) Equation 1 can be written in matrix form as follows:

$$A = QR \ .$$

If the columns of A are linearly independent, Q is $m \times n$ and R is $n \times n$. However, in general, the columns of A may be linearly dependent. To understand what happens to Q and R then, let us consider an example. Suppose that in a 5×4 matrix A the columns $\mathbf{a}_1$, $\mathbf{a}_2$, $\mathbf{a}_4$ are linearly independent but $\mathbf{a}_3$ depends linearly on $\mathbf{a}_1$ and $\mathbf{a}_2$. Then the `if` statement in Gram-Schmidt fails for $j = 3$. Describe how the two matrices Q and R change as a consequence. Specifically, show the columns of Q and the fill pattern of R, and explain your reasoning by showing what happens in the third and fourth iteration of the `for` loop. Remember that Q must remain orthogonal and that it must be possible to multiply Q with R.

(i) What are the sizes of Q and R if A has m rows, n columns, and rank r?

(j) Write a MATLAB function

```
function [Q, R] = gs(A)
```

that returns the matrices Q and R obtained by applying Gram-Schmidt to the matrix A. Make sure that the resulting matrices `Q` and `R` are neither bigger nor smaller than they need to be.

Here and elsewhere in this assignment, use the following test to check whether some number `z` is nonzero:

```
if abs(z) > sqrt(eps)
```

and remove the `abs()` if you know that `z` is nonnegative. The expression `sqrt(eps)` results in a small number that is large relative to `eps`, the smallest strictly positive number that can be represented with a variable of type `double` (you can check that `sqrt(eps)` $\approx 1.5 \times 10^{-8}$).

It is OK to use `for` loops.

IMPORTANT: Writing A as the product QR is called the QR *decomposition* in the literature, and MATLAB has a function `qr` to compute it (although the matrices that result from the MATLAB version are in general different from what you obtain in this assignment). You should *not* use the MATLAB `qr` function, nor any of the functions MATLAB provides to solve linear systems, compute the rank of a matrix, compute any other matrix decomposition, perform back-substitution, and the like. In other words, you should write your code from scratch by following your answers to the previous questions, and using low-level MATLAB constructs only. This holds for all coding questions in this assignment.

(k) For convenience in what follows, let us generalize the Gram-Schmidt algorithm slightly. Specifically, assume that you are given an initial QR decomposition of some matrix $B = [\mathbf{b}_1, \ldots, \mathbf{b}_{n_0}] = QR$ that has m rows and n_0 columns, but then you are given an additional set of n vectors in the $m \times n$ matrix $A = [\mathbf{a}_1, \ldots, \mathbf{a}_n]$. The task is to grow the input matrices Q and R so that the new matrices are a QR decomposition of the $m \times (n_0 + n)$ matrix $[B|A] = [\mathbf{b}_1, \ldots, \mathbf{b}_{n_0}, \mathbf{a}_1, \ldots, \mathbf{a}_n]$.

Write a new MATLAB function

```
function [Q, R] = ggs(A, Q, R)
```

(`ggs` stands for "generalized Gram-Schmidt") that does this. In particular, if you call

```
[Q, R] = ggs(A);
```

or

```
[Q, R] = ggs(A, [], []);
```

you obtain the same result as

```
[Q, R] = gs(A);
```

Of course, `ggs` is a relatively minor modification of `gs`, but requires some care with matrix indices. A `ggs.m` template file provided with this assignment shows you how to handle a variable number of input arguments, and does some size checks for you.

REMARK: Typically, since `ggs` subsumes `gs`, we would just implement `ggs` (and perhaps call it `gs`), but developing the code in the stages given in this assignment makes it simpler to think about the various aspects in isolation.

2. NOTE: *The questions that follow depend on a correct implementation of `ggs`. To prevent excessive penalties in case you did not get your `ggs` right, the function `mggs` is provided with this assignment. This function implements (inefficiently) the functionality of `ggs` using the* MATLAB *functions `qr` and `rank`, neither of which you are allowed to use for this assignment. Results may differ by signs, since simultaneously switching signs for a column of Q and a row of R leaves the product QR unaltered. You get 80 percent credit for each question in this problem where your answer uses `mggs` instead of your own `ggs`. Either way, state clearly whether you used your `ggs` or the provided `mggs`.*

Equation 1 can be written in matrix form as follows:

$$A = QR$$

so if we have a linear system

$$A\mathbf{x} = \mathbf{b}\ , \tag{2}$$

we can rewrite it as

$$QR\mathbf{x} = \mathbf{b}$$

or

$$R\mathbf{x} = \mathbf{c}\ . \tag{3}$$

(a) Write an expression for $\mathbf{c}$ that does not contain $\mathbf{x}$.

(b) What property of Q makes $\mathbf{c}$ very easy to compute given Q and $\mathbf{b}$?

(c) Explain clearly and in detail why the system in equation 3 is easier to solve than that in equation 2. The clearest answer is in the form of a pseudo-code algorithm to solve the system. Assume that the columns of A are linearly independent.

(d) How does your method for solving equation 3 change if the columns of A are linearly dependent? Interpret "solving" to mean "find a solution" rather than "find all solutions." Explain clearly, preferably in the form of an algorithm.

(e) If you have the matrices Q and R of the QR decomposition of a matrix A, how can you use `ggs` efficiently to find a matrix L whose columns are an orthonormal basis for the left null space of A? Explain your reasoning and show a code snippet. [Hint: extend A with additional columns to make sure that the span of the extended matrix is all of $\mathbb{R}^m$.]

(f) Put everything together by writing a MATLAB function

```
function [x, N, W, L, Q, R] = solve(A, b)
```

that uses `ggs` efficiently to return the solution $\mathbf{x}$ of an arbitrary linear system of the form shown in equation 2, as well as the following matrices (some of which are possibly empty):

`N`: A matrix whose columns are an orthonormal basis for the **n**ull space of `A`.

`W`: A matrix whose columns are an orthonormal basis for the ro**w** space of `A`.

`L`: A matrix whose columns are an orthonormal basis for the **l**eft null space of `A`.

`Q`: A matrix whose columns are an orthonormal basis for the range of `A`.

`R`: A matrix such that the QR decomposition of `A` is `QR`.

If the system has no solution, the output variable `x` should be set to the empty matrix `[]`.

(g) How can you tell from the output from `solve` if the system in equation 3 admits infinitely many solutions? If it does, define the set of solutions precisely in terms of the outputs from `solve`. What kind of space is this set?

(h) How can you check that N and W are bases for mutually orthogonal spaces, and that so are L and Q?

(i) Run the function `tests` provided with this assignment as follows:

```
tests(@solve)
```

This will pass a handle for your function `solve` to the `tests` function. If all works well, this function will create a file `tests.tex` with the results of several tests on various matrices. Make sure that the resulting file `tests.tex` is in the same directory as your `.tex` file, and include it with the instruction

```
\input{tests}
```

in your `.tex` file in the appropriate place.

If your code has no mistakes, when you look at your `.pdf` file you will see a fresh page with six test results, each starting with a line like the following (with appropriate values of n and r):

> Test case n: A has rank r. Product checks passed. Dimension checks passed.

If execution of your function `solve` fails on a particular test case, MATLAB prints a warning message during execution of `tests` and the results for that test case are omitted from the output file. If some of the matrices have inconsistent dimensions or some of their products are not what is expected, the results are placed into `tests.tex`, but the line above will say "**FAILED**" instead of "passed" as appropriate. If any of these problems occur, debug your `solve` function and try again.

IMPORTANT: Do not change anything in the file `tests.m`. We rely on what is done in that file to check that your results are correct, so if you modify that file your results may appear different from what we expect.