Numerical Methods for CSE

The Gram-Schmidt Orthogonalisation

(Explanation of Exercise 1.2)

Giuseppe Accaputo
g@accaputo.ch

September 28, 2016

In this document I will guide you through the implementation of the Gram-Schmidt orthogonalisation of an ordered finite set $\{\mathbf{a}^1, \ldots, \mathbf{a}^k\}$, $k \in \mathbb{N}$, $\mathbf{a}^l \in \mathbb{K}^n$ as explained in the solution code for exercise 1.2. The implementations shown in this document are based on the Gram-Schmidt algorithm shown in (1.5.1) of [1].

Notation

Throughout this document (and my material for the exercise sessions in general) I will be using the following notation in mathematical formulas:

$\mathbf{x}$:	Column vector (small letter, bold)
$\mathbf{x}^T$:	Row vector (small letter, bold, transposed)
$\mathbf{A}$:	Matrix (large letter, bold)

Version 1: Implementation Using Two `for`-Loops

The algorithm presented in (1.5.1) of [1] can be directly implemented using two nested `for`-loops; the implementation may look as follows:

```
for (int j = 1; j < A.cols(); ++j)
{
    for (int l = 0; l < j; ++l)
    {
        Q.col(j) -= A.col(j).dot(Q.col(l)) * Q.col(l
          ↪ );
```

```
    }
    ...
}
```

Version 2: Implementation Using Only One `for`-Loop

In this section I am going to explain the implementation and the mathematical idea behind the solution provided in `/NumCSE/Assignments/Codes` ↪ `/MatVec/GramSchmidt/solutions_nolabels/gramschmidt.cpp`.

In the solution code one can see that the inner-most `for`-loop has been replaced by a matrix-vector multiplication:

```
for (int j = 1; j < A.cols(); ++j)
{
    Q.col(j) -= Q.leftCols(j) * (Q.leftCols(j).
        ↪ transpose() * A.col(j));
}
```

Assuming that the vector indices in mathematical formulas (not in the code) start at 1, i.e., $\mathbf{q}^1$ being the first vector, then the equation shown in the above code listing can be written as

$$\mathbf{q}^j = \mathbf{q}^j - \mathbf{Q}_{j-1}\,(\mathbf{Q}_{j-1}^T\,\mathbf{a}^j) \quad , \tag{1}$$

where $\mathbf{Q}_{j-1}$ is the matrix containing the first $j-1$ columns of $\mathbf{Q}$, i.e.,

$$\mathbf{Q}_{j-1} = (\mathbf{q}^1, \cdots, \mathbf{q}^{j-1}) \quad . \tag{2}$$

First Confusing Fact: The Indices

The first thing that may be confusing in Eq. 1 is the fact that in the code both the `Q.col(j)` and `Q.leftCols(j)` functions use the same index j, but in the equation shown in Eq. 1 $\mathbf{q}^j$ and $\mathbf{Q}_{j-1}$ use different values for the indices, namely j and $j-1$ respectively.

The reason for this is the following: since the columns of the matrix $\mathbf{Q} \in \mathbb{R}^{n\times n}$ are numbered from $0, \ldots, n-1$ in the code (not in the mathematical formula), i.e., `Q.col(0)` is the first column (in mathematical notation this would be the vector $\mathbf{q}^1$), by starting with $j = 1$ in the `for`-loop we are accessing the *second* column of $\mathbf{Q}$ (since `Q.col(0)` is the first column). Further, since `Q.leftCols(j)` selects the first j columns of the matrix $\mathbf{Q}$ (see the Eigen documentation), for $j = 1$ we are actually selecting only the first column of $\mathbf{Q}$. Since in each step j of the Gram-Schmidt algorithm the vector $\mathbf{q}^j$ is constructed from the previous $j-1$ vectors, one can see why we are working with $\mathbf{Q}_{j-1} = (\mathbf{q}^1, \ldots, \mathbf{q}^{j-1})$.

Second Confusing Fact: The Equation in General

The second confusing thing in Eq. 1 may be the equation in general.

Why can we replace the inner-most loop by introducing the matrix-vector multiplication shown in Eq. 1?

Working with Projections

The projection operator is defined as [2]

$$\text{proj}_{\mathbf{u}} = \frac{\langle \mathbf{v}, \mathbf{u} \rangle}{\langle \mathbf{u}, \mathbf{u} \rangle}\mathbf{u} = \frac{\mathbf{v} \cdot \mathbf{u}}{\mathbf{u} \cdot \mathbf{u}}\mathbf{u} = \frac{\mathbf{v}^T\mathbf{u}}{\mathbf{u}^T\mathbf{u}}\mathbf{u} \tag{3}$$

and it projects the vector $\mathbf{v}$ orthogonally onto the line spanned by vector $\mathbf{u}$.

If $\mathbf{u}$ is normalized, i.e. $\langle \mathbf{u}, \mathbf{u} \rangle = 1$, then the projection operator simplifies to

$$\text{proj}_{\mathbf{u}} = \mathbf{v} \cdot \mathbf{u}\mathbf{u} = (\mathbf{v}^T\mathbf{u})\mathbf{u} \tag{4}$$

The main part of the Gram-Schmidt algorithm in (1.5.1) of [1] is given by

$$
\begin{array}{l}
\texttt{for} \quad j = 2, 3, \ldots, n \quad \text{do} \\
\quad \mathbf{q}^j := \mathbf{a}^j \\
\quad \texttt{for} \quad l = 1, 2, \ldots, j-1 \quad \text{do} \\
\quad\quad \mathbf{q}^j \leftarrow \mathbf{q}^j - \mathbf{a}^j \cdot \mathbf{q}^j\, \mathbf{q}^j \\
\quad \texttt{if} \quad (\mathbf{q}^j = \mathbf{0}) \quad \text{then STOP} \\
\quad \texttt{else} \quad \mathbf{q}^j \leftarrow \mathbf{q}^j / \|\mathbf{q}^j\|_2
\end{array}
$$

If we omit the inner-most `for`-loop and write the equation for $\mathbf{q}^j$ in an explicit way, we get

$$\mathbf{q}^j \leftarrow \mathbf{q}^j - \left(\mathbf{a}^j \cdot \mathbf{q}^1\, \mathbf{q}^1 + \mathbf{a}^j \cdot \mathbf{q}^2\, \mathbf{q}^2 + \cdots + \mathbf{a}^j \cdot \mathbf{q}^{j-1}\, \mathbf{q}^{j-1}\right) \tag{5}$$

Since the resulting vector $\mathbf{q}^j$ gets normalized in a final step, i.e. $\mathbf{q}^j / \|\mathbf{q}^j\|$, we actually have an orthonormalization algorithm. Thus, by using the fact that all previous $\mathbf{q}_1, \ldots, \mathbf{q}_{j-1}$ are orthonormalized (orthogonalized and normalized) in Eq. 5 and by using the definition of the projection in Eq. 4, the calculation of $\mathbf{q}^j$ in Eq. 5 can be rewritten as

$$\mathbf{q}^j \leftarrow \mathbf{q}^j - \sum_{l=1}^{j-1} \text{proj}_{\mathbf{q}^l}(\mathbf{a}^j) \tag{6}$$

Corollar 7.5 from [3] defines the orthogonal projection $\mathbf{P_Q}$ to be the projection onto the column-space of the matrix $\mathbf{Q} = (\mathbf{q}_1, \ldots, \mathbf{q}_n)$ with orthonormal columns and it is given by

$$\mathbf{P_Q} :\equiv \mathbf{Q}\mathbf{Q}^T. \tag{7}$$

This corollar also defines that

$$\mathbf{P_Q}\mathbf{y} = \mathbf{Q}\mathbf{Q}^T\mathbf{y} = \sum_{j=1}^{n} \mathbf{q}_j \mathbf{q}_j \cdot \mathbf{y} \tag{8}$$

Using the definition of the projection operator from Eq. 4 we can now see that

$$\mathbf{P_Q}\mathbf{y} = \sum_{j=1}^{n} \text{proj}_{\mathbf{q}_j} \mathbf{y} \tag{9}$$

Thus, using the definition of the orthogonal projection $\mathbf{P_Q}$ from Eq. 7 in the calculation of $\mathbf{q}^j$ shown in Eq. 6, we can rewrite Eq. 6 as follows:

$$\begin{aligned} \mathbf{q}^j &\leftarrow \mathbf{q}^j - \sum_{l=1}^{j-1} \text{proj}_{\mathbf{q}^l}(\mathbf{a}^j) \\ &\Longleftrightarrow \\ \mathbf{q}^j &\leftarrow \mathbf{q}^j - \underbrace{\mathbf{Q}_{j-1}\mathbf{Q}_{j-1}^T}_{=:\mathbf{P}_{\mathbf{Q}_{j-1}}} \mathbf{a}_j \quad , \end{aligned} \tag{10}$$

where $\mathbf{Q}_{j-1}$ is defined in Eq. 2.

The final version of the Gram-Schmidt algorithm — which is also implemented in the solution code — is then given by

$$\begin{aligned} &\texttt{for} \quad j = 2, 3, \ldots, n \quad \text{do} \\ &\quad \mathbf{q}^j := \mathbf{a}^j \\ &\quad \mathbf{q}^j \leftarrow \mathbf{q}^j - \mathbf{Q}_{j-1}\mathbf{Q}_{j-1}^T \mathbf{a}_j \\ &\quad \texttt{if} \quad (\mathbf{q}^j = \mathbf{0}) \quad \text{then STOP} \\ &\quad \texttt{else} \quad \mathbf{q}^j \leftarrow \mathbf{q}^j / \|\mathbf{q}^j\|_2 \end{aligned}$$

References

[1] R. Hiptmair, “Numerical methods for computational science and engineering.” https://www.sam.math.ethz.ch/~grsam/HS16/NumCSE/NumCSE16.pdf, 2016.

[2] E. W. Weisstein, “Projection. From MathWorld—A Wolfram Web Resource.” http://mathworld.wolfram.com/Projection.html.

[3] M. H. Gutknecht, “Lineare algebra.” http://www.sam.math.ethz.ch/~mhg/unt/LA/HS07/LAS07.pdf, 2007.