Practicum 2: Orthogonality and orthonormal basis

Efstratios Gavves, Leo Dorst and Fokko van de Bult

November 12, 2010

1 The Gram-Schmidt algorithm

The sections 5.1 and 5.2 of the book describe the concept of orthogonality, and orthonormal bases(ONB). The orthornomal bases have a number of appealing properties that makes them more convenient to work with than other bases. Therefore we would like to have an algorithm that takes an arbitrary basis of a linear subspace as input and transforms it into and orthonormal basis. The Gram-Schmid algorithm does exactly this and is explained in details in section 5.2. The assignment for the next week is to implement the Gram-Schmid algorithm. As you can see in section 5.2, there are figures that explain the process graphically. Often it is convenient to plot the results, preferrably in 3D. That way you can not only better grasp the concept, but also visually inspect your implementation and check whether there is something wrong.

1.1 1-D case

We start with 1-dimensional case and we can then further extend the problem to more dimensions. Let's say that we have a vector $\vec{v}$. We now want to find the ONB for the space that is spanned by $\vec{v}$.

Exercise 1 *How can you generate an ONB for the subspace $V = span(\vec{v})$? This ONB will be composed of 1 vector apparently, which will be also an orthonormal one. Write a function called* `normalize`*, which takes as input the vector $\vec{v}$ and gives as output a normalized vector, which forms the orthonormal basis for V. For the implementation part, it is better not to use any for-loops. For loops are rather slow in Matlab. Instead, Matlab has many matrix operations that do a similar thing much more efficiently(remember operators $.*$ and so on).(1pt)*

The template code for this function should be: **Template code**

```
function onb = normalize(v)

% code
```

You could verify the validity of your code using the exercises 5.2.1-5.2.14.

1.2 2-D case

Now, we will add one more vector. Given the basis vector that we found in Exercise 1, we add on more vector v_2. So now we have the basis w_1, v_2, where $w_1 = normalize(v_1)$. Apparently this is not an ONB basis yet and we want to change that, so as w_1 remaings the same and v_2 is replaced by w_2 perpendicular to w_1. The final basis should span the same subspace with the original one, that is $span(w_1, v_2) = span(w_1, w_2)$. As the book states, in order to do show, you first have to find the vector $v_2^{\perp}$ and normalizes it.

Exercise 2 *Write a function called* `onb2`*. This function takes as input w_1 and v_2. The first vector w_1 is already the ONB for the subspace spanned by itself. The function should compute the* *second vector w_2 of the ONB of the 2-D space, such as w_2 is derived by v_2, is perpendicular to w_1 and normalized to unit length. The output of the function should be a matrix. The matrix will contain in its columns the vectors that form the ONB of our space.(1pt)*

For the normalization, you could also use the function that you have already built in the first exercise(if it's built properly of course).

1.3 Many-D case

We want now to generalize this procedure, so as to be able to create an ONB from arbitrarily many vectors, that is given the vectors $\{v_1, ..., v_k\}$ we want to find the ONB for the the space $V = span(v_1, ..., v_k)$. Following the paradigm from exercise 1 and 2 and also the theorem 5.2.1 of the book, we have to perform an iterative procedure. First, we find the ONB for the vector v_1 as if it was the only vector that we have. That is we have to normalize it to unit length. Then, we take vector v_2, we find its projection $v_2^\perp$ to the axis perpendicular to v_1. We then normalize $v_2^\perp$, so now we have $\{w_1, w_2\}$. In the third iteration we take the vector v_3 and we again find its projection $v_3^\perp$, such as $v_3^\perp$ is perpendicular to w_1, w_2. We then normalize $v_3^\perp$. Next, we do the same for as many vectors(that is k) as we have.

Exercise 3 *Write the function* `onb`*. The input for this function should be a matrix V of vectors, whose size is not a constant number. The function should recognize the size of the spanned space and construct the ONB for that space. The output of the function should be again a matrix W of the same size as V, which contains the ONB for the spanned space.(2pt)*

You can easily find the size of the spanned space, since it is the same as the number of the vectors in V. In Matlab you can do that by using the

```
size(V)
```

function.

Hint: For a more efficient implementation(which is important for many dimensions), you might want to use the projection operator (Theorem 5.3.10 in the book). If you are motivated enough, you are advised to look a bit into it. If you find it, you will unlock some secrets of the space $span(Linear Algebra + Matlab)$:-).

1.4 A little piece of advice

If you perform the orthonormalization in a 1, 2, 3-D, you can actually plot the results using the function `vector`. You can use different colors to draw the different vectors. For example, you can plot a red vector by typing

```
vector(v, 'r')
```

If you want to find more about the colors that you can use, just type

```
help plot
```

Be careful when plotting! Matlab scales the axis so as the data to fit and therefore the vectors might be distorted. In order to make the axis equal, type

```
axis equal
```

2 What is required?

Please send me an email with the m-files containing the functions you have implemented. Test the commands using the exercises 5.2:1, 3, 7, 13. Report the results of these runs on the exercises, plotting also the results where possible. You can work in couples, putting both names in the final report.