Gram-Schmidt Orthonormalization

Name: Sanghyeon Kim

Student ID: 20181605

Date: 10/02/2020

The program takes two row vectors, $\mathbf{v}_1$ and $\mathbf{v}_2$, as input. The Gram-Schmidt method iteratively constructs orthogonal vectors by projecting each vector onto a vector space spanned by orthogonal vectors and subtracting the projection from that vector. Because there are only two vectors in three-dimensional space, the first vector $\mathbf{v}_1$ is set as the first orthogonal vector $\mathbf{u}_1$. Then, the projection of the second vector $\mathbf{v}_2$ onto the first vector $\mathbf{u}_1$ is found. The projection can be represented as a scalar multiple of the first vector. The scalar multiple is declared as lambda and it is found by using the relationship between the inner product and orthogonality. The dot product is used as the inner product and lambda is used to find the vector $\mathbf{u}_2$. After $\mathbf{u}_1$ and $\mathbf{u}_2$ have been found, each orthogonal vector is normalized to create a set of orthonormal vectors $\mathbf{c}_1$ and $\mathbf{c}_2$.

There are three functions defined in the program, which are of the following:

- int main(): Produces and prints the orthonormalized vectors $\mathbf{c}_1$ and $\mathbf{c}_2$ in each line.
- void normalize(float V[], int size): Takes a vector(array) address and its size and normalizes the vector to length 1.
- float dot_product(float v1[], float v2[], int size): Takes address of two vectors(arrays) of the same size and their size. Then the dot product of the two vectors is returned as float.

To calculate the norm(length) of a vector, the square root of its inner product must be found. Thus, the program depends on <math.h> header that contains the function <double sqrt(double)>. When using GCC on GNU/Linux system, the <math.h> must be explicitly linked with -lm.

```c
/*

Name:   gram_schmidt.c

Author: Sanghyeon Kim(Student ID: 20181605)

Date:   10/02/2020

-----------------------------------------------------------------

Input:  Takes two row vectors in three-dimensional space
        as input. Each row vector is entered elementwise.

Process:Following the Gram-Schmidt orthonormalization, vector
        v_1 is set as u_1 and orthogonal vector u_2 is found
        using v_2 and u_1. Then, u_1 and u_2 are normalized
        to length 1 into orthonormal vectors c_1 and c_2.

Output: Prints two orthonormalized row vectors c_1 and c_2.

*/

#include <stdio.h>
#include <math.h>

void normalize(float V[], int size);
float dot_product(float v1[], float v2[], int size);

int main(){
    float V[2][3] = {{0}};
    float lambda;

    printf("<Input two row vectors(v_1, v_2) in three-
dimensional space>\n\n");

    for(int row = 0; row < 2; row++){
        printf("Enter the elements of v_%d: ", row+1);
        for(int col = 0; col < 3; ++col){
            scanf("%f", *(V+row)+col);
        }
    }

    // Find lambda(the coordinate) of the projection of v_2 onto u_1(v_1)
    lambda = dot_product(V[1], V[0], 3)/dot_product(V[0], V[0], 3);

    // The second orthogonal vector u_2 replaces v_2
    for(int col = 0; col < 3; ++col){
        V[1][col] = V[1][col] - lambda*V[0][col];
```

```
    }

    printf("\n");

    // Print the orthonormalized vectors c_1 and c_2
    for(int row = 0; row < 2; ++row){
        normalize(V[row], 3); // Normalizes each of the orthogonal vectors
        printf("c_%d: (", row+1);
        for(int col = 0; col < 3; ++col){
            printf("%f, ", V[row][col]);
        }
        printf("\b\b)\n");
    }

    return 0;
}

// Normalizes the vector V to length 1.
void normalize(float V[], int size){
    float sum, root;

    sum = dot_product(V, V, size);

    root = sqrt(sum);

    for(int i = 0; i < size; ++i){
        V[i] /= root;
    }

    return;
}

/*
    Performs a dot product of two vectors v1, v2
    and returns the result
*/
float dot_product(float v1[], float v2[], int size){
    float result = 0;

    for(int i = 0; i < size; ++i){
        result += v1[i]*v2[i];
    }

    return result;
}
```