

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO BÀI THỰC HÀNH LAB 2

Giảng viên hướng dẫn: Trần Hà Sơn

Họ và tên: Trần Đình Nhật Trí

Mã số sinh viên: 21120576

I. Thông tin cá nhân:

Họ và tên	MSSV
Trần Đình Nhật Trí	21120576

II. Ý tưởng thực hiện:

1. Inner Product:

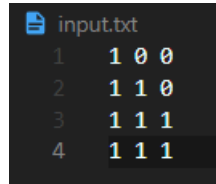
- Trước tiên, xét xem số hàng của vector $v1$ có bằng số cột của vector $v2$ không. Nếu không có thì hàm báo lỗi và thoát chương trình, ngược lại đi vào vòng lặp với độ dài là số hàng của vector $v1$ hoặc số cột của $v2$. Cộng dồn các tích của các cặp phần tử $v1$ và $v2$ lại với nhau sau đó trả về kết quả là tích vô hướng của hai vector.

2. QR Factorization:

- Sử dụng quá trình gram-schmidt để trực chuẩn hóa một ma trận thành ma trận orthogonal (Q), sau đó ta sử dụng phép biến đổi ma trận orthogonal để tạo thành ma trận tam giác trên (R). Sau khi thực hiện gram-schmidt để tạo ra Q , chuyển vị ma trận Q để nhân với ma trận đã cho để tính R .

III. Cách nhập ma trận:

Từ file input.txt, nhập vào một ma trận tùy ý như sau:



```
input.txt
1 1 0 0
2 1 1 0
3 1 1 1
4 1 1 1
```

Hình 1. File đầu vào để đọc ma trận

IV. Mô tả các hàm:

1. `innerproduct(v1, v2)`:

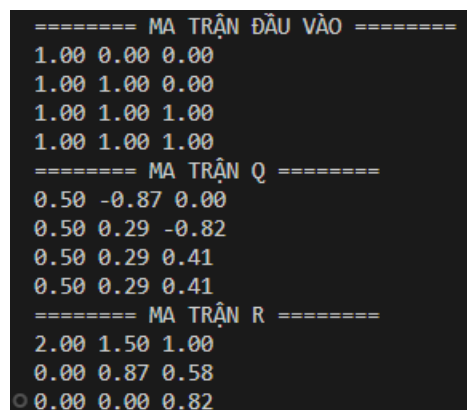
- Hàm xử lý tính toán tích vô hướng của hai vector, với đối số truyền vào là hai vector cần tính.

2. `gramSchmidtProcess(A)`:

- Hàm tính quá trình gram-schmidt, hỗ trợ cho việc tính toán phân rã QR.

3. `QR_factorization(A)`:

- Hàm phân rã QR, được sử dụng để tìm hai ma trận trực giao (Q) và ma trận tam giác trên (R), với đối số truyền vào là ma trận A cho trước.



```
===== MA TRẬN ĐẦU VÀO =====
1.00 0.00 0.00
1.00 1.00 0.00
1.00 1.00 1.00
1.00 1.00 1.00
===== MA TRẬN Q =====
0.50 -0.87 0.00
0.50 0.29 -0.82
0.50 0.29 0.41
0.50 0.29 0.41
===== MA TRẬN R =====
2.00 1.50 1.00
0.00 0.87 0.58
0.00 0.00 0.82
```

Hình 2. Hai ma trận Q và R tính được từ ma trận hình 1.

4. Các hàm phụ trợ:

- **multiplyScalarMatrix(scalar, A)**: hàm giúp để nhân hệ số với ma trận.
- **getSize(v)**: hàm được sử dụng để tính độ lớn của vector.
- **NhanMaTran(A, B)**: hàm giúp nhân hai ma trận với nhau.
- **Transpose(A)**: hàm tạo ma trận chuyển vị từ ma trận đầu vào.
- **TruMaTran(A, B)**: hàm giúp trừ hai ma trận.
- **guassElimination(A)**: hàm tìm hạng của ma trận A.
- **isLinearDependent(A)**: hàm kiểm tra ma trận có phụ thuộc tuyến tính không.
- **isZeroCols** và **isZeroRows**: hàm xét các hàng 0 và cột 0 để kiểm tra tính phụ thuộc.

---- HẾT ----