

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**ĐHQG TPHCM**

**Báo Cáo Đồ Án 1**

**Instructors: Nguyễn Đình Thúc**

**Nguyễn Văn Quang Huy**

**Nguyễn Trọng Hiến**

**Võ Nam Thục Đoan**

**Thông tin Cá Nhân:**

1. Tôn Thất Bách MSSV:20127442

Lưu ý: các đoạn chương trình cung cấp trong báo cáo có thể cần có hàm phụ để chạy được, xem chi tiết hơn ở mã nguồn gửi kèm báo cáo.

Mục lục

[I. Nội dung thực hiện 3](#_Toc104805157)

[**1.** **Phép khử Gauss** 3](#_Toc104805158)

[**2.** **Phương pháp Backward Subtitution (Phương pháp thế ngược)** 5](#_Toc104805159)

[**i.** **Trường hợp hệ phương trình có nghiệm duy nhất** 5](#_Toc104805160)

[**ii.** **Trường hợp hệ phương trình có vô số nghiệm** 7](#_Toc104805161)

[II. Tham khảo 9](#_Toc104805162)

# I. Nội dung thực hiện

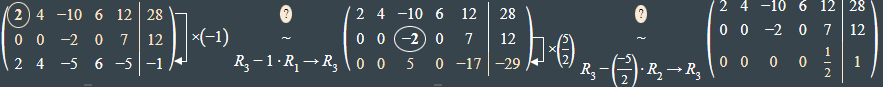
1. **Phép khử Gauss**
   * Mô tả thuật toán: phép khử Gauss dùng các thao tác cơ bản trên hàng để đưa một ma trận thành **dạng bậc thang**, từ đó có thể được dùng để tìm nghiệm của một hệ phương trình tuyến tính… Theo yêu cầu đồ án, tập trung sử dụng thuật toán Gauss để tìm nghiệm của **hệ phương trình tuyến tính**
   * Ý tưởng thực hiện:

Giả sử cần tìm nghiệm của hệ phương trình tuyến tính:

* + - Ảnh có chứa văn bản, đồng hồ đo, thiết bị

      Mô tả được tạo tự độngBiểu diễn phương trình dưới dạng ma trận:
    - Ảnh có chứa văn bản, thiết bị đo

      Mô tả được tạo tự độngBước 1: Kiểm tra từng cột trong ma trận, nếu phát hiện khác vector 0 thì đổi chỗ 2 hàng ( nếu cần thiết), nếu cột hiện tại bằng vector 0 thì kiểm tra sang cột khác
    - Bước 2: Thực hiện các thao tác trên hàng để đưa về dạng bậc thang



* + **Cài đặt Phép khử Gauss sử dụng ngôn ngữ python**

def gaussElimination(matrix,m,n):

    p=0

    for i in range(m):

        if all(matrix[i][j]==0 for j in range(n-1)) and matrix[i][n-1]!=0:

            break

        if matrix[i][p]==0:

            check=0

            for j in range (i,m): #Kiem tra tung cot de swap neu can thiet

                if matrix[j][p]!=0:

                    matrix[i], matrix[j] = matrix[j], matrix[i]

                    check=1

                    break

            if check == 0: #neu cot bang vector 0 thi doi sang cot ke tiep

                p+=1

                if p>n-1:

                    p-=1

        if matrix[i][p]!=0:

            temp = matrix[i][p]+0.0

            for j in range (n):

                if matrix[i][j]!=0:

                    matrix[i][j]/=temp

            for k in range (i+1,m,1): #kiem dong` khac 0 tinh tu cot pivot

                if matrix[k][p]!=0:

                    multi= matrix[k][p]+0.0 # gia tri can` nhan^ len

                    for j in range (p,n,1):

                        matrix[k][j]-=(multi\*matrix[i][j])+0.0

#matrix[i][j] la dong` cua? pivot

            p+=1

    return matrix

Kết quả thực thi với ma trận giả sử(đã kiểm tra với nguồn [**matrixcalc.org**](https://matrixcalc.org/)**)**:

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản, bàn phím, thiết bị điện tử

Mô tả được tạo tự độngẢnh có chứa văn bản

Mô tả được tạo tự độngKết quả với một số ma trận khác:

1. **Phương pháp Backward Subtitution (Phương pháp thế ngược)**

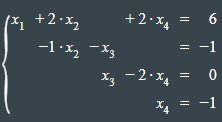
* **Mô tả:** là phương pháp khá thông dụng để tìm nghiệm của hệ phương trình tuyến tính dựa trên ma trận dạng bậc thang của nó

**Ý tưởng thực hiện:**

1. **Trường hợp hệ phương trình có nghiệm duy nhất**

* Ảnh có chứa văn bản, đóng

  Mô tả được tạo tự độngGiả sử ta có ma trận dạng bậc thang sau:
* Ảnh có chứa văn bản

  Mô tả được tạo tự độngBiểu diễn dưới dạng hệ phương trình tuyến tính:

**Nguồn:** [**matrixcalc.org**](https://matrixcalc.org/)

**Ý tưởng thực hiện:**

**Các hệ số ở cột cuối cùng trong ma trận là kết quả sau dấu ‘=’**

Bước 1: Ta duyệt từ trái sang phải các phần tử trong ma trận bắt đầu từ dòng cuối cùng, phần tử khác 0 đầu tiên là biến số cần tìm.

Bước 2: Tiếp theo ta duyệt sang phải bắt đầu từ phần tử khác 0 đầu tiên lần lượt lấy kết quả cuối cùng (S) trừ đi (hệ số\*kết quả của biến ở vị trí tương ứng)

Bước 3: cập nhật kết quả tương ứng vào mảng Ans

Bước 4: lặp lại cho đến khi duyệt hết tất cả các hàng

* **Cài đặt bằng ngôn ngữ python**

def subtitution(a,m,n):

    x=[0]\*(n-1)

    for i in range(m-1,-1,-1):

        for j in range(n):

            if a[i][j]!=0:

                temp=a[i][n-1]

                for k in range(j+1,n-1):

                    temp-=a[i][k]\*x[k]

#a[i][k] la he so tren ma tran

                temp/=a[i][j]

                x[i]=temp

                break

    return x

* Ảnh có chứa văn bản

  Mô tả được tạo tự độngKết quả với ma trận giả sử (đã kiểm tra với nguồn [**matrixcalc.org**](https://matrixcalc.org/)**)**:

1. **Trường hợp hệ phương trình có vô số nghiệm**

* Một hệ phương trình tuyến tính A có vô số nghiệm khi số dòng khác 0 nhỏ hơn số biến số của nó ở dạng ma trận bậc thang hay Rank(A) < k với k là số lượng ẩn số

Cài đặt điều kiện với Python:

def InfiniteSol(matrix,m,n): #Kiểm tra phương trình có vô số nghiệm hay không

    check = False

    dem =0

    for i in range(m):

        for j in range(n):

            if matrix[i][j]!=0:

                dem+=1

                break

    if dem<len(matrix[0])-1:

#Phương trình vô số nghiệm khi số hàng khác 0 nhỏ hơn số lượng biến số trong hệ

        check=True

    return check

Ý tưởng thực hiện Back Subtitution:

* Thực hiện tương tự như với hệ phương trình có nghiệm duy nhất ở trên
* Chỉ khác ở bước cập nhật lại kết quả S thêm 1 bước kiểm tra nếu kết quả của biến ở vị trí tương ứng không tồn tại thì đặt nó là một biến tự do và tiếp tục thực hiện

**Để tiện hơn trong việc biểu diễn kết quả cuối cùng trong đồ án có sử dụng thêm thuật toán Jordan để tối ưu hóa ma trận bậc thang Gauss nhưng không đề cập vì mục tiêu Đồ Án là thuật toán Gauss (xem chi tiết cài đặt thuật toán Jordan ở mã nguồn gửi kèm)**

**Cài đặt Back Subtitution cho hệ vô số nghiệm với Python:**

def InfiniteSolSubtitution(a,m,n):

    x=['none']\*(n-1)

    for i in range(m-1,-1,-1):

        for j in range(n):

            if a[i][j]!=0:

                temp=a[i][n-1]

                for k in range (j+1,n-1):

                    temp = str(temp)

                    if(a[i][k]==0): continue

                    if x[k]=='none':

                        x[k]='a'+str(k)

                        temp=temp+'-'+str(a[i][k])+str(x[k])

                    else: temp=temp+'-'+str(a[i][k])+str(x[k])

                if isinstance(temp,str):

                    temp='('+temp+')'

                else: temp='('+str(temp)+')'

                x[j]=temp

                x[j]=x[j].replace('--','+')

                x[j]=x[j].replace('0.0','')

                break

    return x

Ảnh có chứa văn bản, màn hình, ảnh chụp màn hình

Mô tả được tạo tự độngẢnh có chứa văn bản

Mô tả được tạo tự động**Kết quả với một số ma trận(đã kiếm tra với nguồn** [**matrixcalc.org**](https://matrixcalc.org/)**) :**

**Ngoài ra còn cài đặt điều kiện xác định hệ phương trình vô nghiệm:**

def Nosolution(matrix,m,n):

    check = False

    for i in range(m):

        if all(matrix[i][j]==0 for j in range(n-1)) and matrix[i][n-1]!=0:

#hệ vô nghiệm khi có toàn bộ trước dấu '=' là không và phía sau dấu '=' khác 0

            check = True

            break

    return check

**Ví dụ:(đã kiểm tra từ nguồn** [**matrixcalc.org**](https://matrixcalc.org/)

Ảnh có chứa văn bản

Mô tả được tạo tự động

# II. Tham khảo

<https://vi.wikipedia.org/wiki/H%E1%BB%87_ph%C6%B0%C6%A1ng_tr%C3%ACnh_tuy%E1%BA%BFn_t%C3%ADnh>

<https://en.wikipedia.org/wiki/Gaussian_elimination>