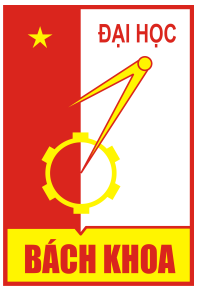
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC

———*———



BÁO CÁO BÀI TẬP LỚN

HỌC PHẦN: KỸ THUẬT LẬP TRÌNH

Bài 3

Giảng viên: TS. Nguyễn Thị Thanh Huyền

Mã Học phần: MI3310
Mã Lớp học: 116444

Sinh viên thực hiện: Nhóm 10

Nguyễn Trần Thức MSSV: 20185483 Mã Lớp: MI2
Đoàn Quốc Vĩnh MSSV: 20185483 Mã Lớp: MI1

HÀ NỘI, 2020

Mục lục

Danh sách hình vẽ

Chương 1

Các bước thực hiện phát triển chương trình

Bài 3 - Thiết kế và viết chương trình

Giải gần đúng hệ phương trình đại số tuyến tính $Ax = b$ bằng **phương pháp lặp đơn và lặp Seidel**:

1) Nhập A, b theo khuân dạng ma trận

2) Kiểm tra tính chéo trội

3) Tính chuẩn của ma trận, kiểm tra sự hội tụ

4) Tính nghiệm gần đúng số lần lặp k cho trước, đánh giá sai số

5) Tính nghiệm gần đúng với sai số e cho trước

6) Tính nghiệm gần đúng $X^{(k)}$ thỏa mãn: $\left\|X^{(k)} - X^{(k-1)}\right\| \leq e$ cho trước

- Mọi kết quả hiển thị với số chữ số thập phân nhập từ bàn phím.
- In cả kết quả trung gian ra màn hình và tệp văn bản.
- Chương trình có chức năng hiển thị kết quả từ tệp văn bản[a]
- Điều khiển chương trình bằng menu.

[a]Theo ý hiểu của nhóm thì yêu cầu này có nghĩa: Ngoài ma trận nhập từ bàn phím, thì chương trình còn có tính năng nhập ma trận từ đọc file dữ liệu

Hình 1.1: Yêu cầu bài toán

1.1 Xác định bài toán

1.1.1 Dữ liệu đầu vào

Đầu vào của bài toán là hệ phương trình tuyến tính n ẩn, n phương trình. Khi biểu diễn trên ngôn ngữ lập trình ta đưa dữ liệu vào dưới dạng *ma trận hệ số* và *ma trận*

vế phải. Ngoài ra, Input còn có số lần lặp *k*, sai số *epsilon*, số *e* , số chữ số thập phân nhập từ bàn phím, và *file data* chứa các phần tử của ma trận.

Về kiểu dữ liệu thì số ẩn, số k, số chữ số thập phân *kiểu nguyên*, sai số và số e kiểu *số thực*, còn ma trận có thể tạo kiểu dữ liệu ma trận hoặc đơn giản hơn là dùng *mảng 2 chiều* để biểu diễn.

Về menu điều khiển, nhóm dùng các phím bấm sau để điều khiển chương trình:

- Hai phím *ARROW UP* và *ARROW DOWN* để chuyển giữa các lựa chọn.
- *ENTER* để chọn lựa chọn đó.

1.1.2 Dữ liệu đầu ra

1. Tính chéo trội: chéo trội hoặc không chéo trội.
2. Chuẩn của ma trận: Kiểu số thực.
3. Sự hội tụ: Hội tụ tới X^* hoặc không hội tụ tới X^*.
4. Nghiệm gần đúng: Kiểu mảng 2 chiều, in ra màn hình và tệp văn bản(cả kết quả chung gian).
5. Sai số: Kiểu số thực.

1.2 Cơ sở học phần *Giải tích số*: giải gần đúng hệ phương trình tuyến tính

Trong học phần đại số tuyến tính mà nhóm đã học thì hệ Gramer được dùng phổ biến để giải hệ phương trình tuyến tính. Nhưng nhược điểm của phương pháp này khi cài đặt trên máy tính điện tử là số lượng phép tính sơ cấp khá lớn, do đó phương pháp này là không khả thi. Vì vậy có phương pháp khác hiệu quả hơn để tính gần đúng nghiệm của hệ phương trình tuyến tính là phương pháp lặp đơn và phiên bản tối ưu hơn của lặp đơn là lặp Seidel. Vì đây không phải là học phần giải tích số nên nhóm sẽ tóm gọn lại công thức liên quan đến bài toán.

1.2.1 Ma trận chéo trội, chuẩn của ma trận

A là ma trận chéo trội thì:

$$| a_{ii} | > \sum_{1 \leq i \neq j \leq n} | a_{ij} | \tag{1.1}$$

Chuẩn của ma trận B theo hàng:

$$\left\|B\right\|_{(\infty)} = \max_i \sum_{j=1}^{n} |\, b_{ij} \,| \qquad (\left\|B\right\|_{(\infty)} \geq 0) \tag{1.2}$$

1.2.2 Điều kiện hội tụ tới nghiệm đúng

Để hệ phương trình $Ax = b \Leftrightarrow x = \alpha x + \beta$ hội tụ tới nghiệm X^* thì:

- A là ma trận chéo trội
- $\left\|\alpha\right\|_{(\infty)} < 1$

1.2.3 Phương pháp lặp đơn

Ta xét hệ phương trình khi biến đổi về dạng: $x = \alpha x + \beta$ thỏa mãn (1.2.2)

Chọn xấp xỉ đầu $X^{(0)} = \beta$ được nghiệm gần đúng $X^{(k)}$ tính bởi công thức lặp:

$$X^{(k)} = \alpha X^{(k-1)} + \beta \tag{1.3}$$

$$x_i^{(k)} = \sum_{j=1}^{i-1} \alpha_{ij}.x_j^{(k)} + \sum_{j=i}^{n} \alpha_{ij}.x_j^{(k-1)} + \beta_i \tag{1.4}$$

$$(i = \overline{1,n}, \qquad k = 1,2,3...)$$

Công thức *lặp đơn* (1.3), và công thức *lặp Seidel* (1.4). Sở dĩ công thức *lặp Seidel* được viết ở mục này (1.2.3) là vì phương pháp lặp đơn là tổng quát, 2 phương pháp con của lặp đơn là *lặp Jacobi* (chi tiết hơn về cách biến đổi về dạng $x = \alpha x + \beta$) và *lặp Seidel* (cải tiến về công thức lặp).

1.2.4 Sai số phương pháp

$$\left\|X^{(k)} - X^*\right\|_{(\infty)} \leq \frac{\left\|\alpha\right\|_{(\infty)}}{1 - \left\|\alpha\right\|_{(\infty)}} \left\|X^{(k)} - X^{(k-1)}\right\|_{(\infty)} \tag{1.5}$$

Hai phương pháp có điểm chung là cùng công thức sai số (1.5)

1.3 Ý tưởng giải quyết bài toán

1.3.1 Thiết kế bằng phương pháp tinh chỉnh dần từng bước

Phương pháp này sẽ đề cập chi tiết trong chương 2 của bài *báo cáo* này.

1.3.2 Cài đặt chương trình theo thuật toán

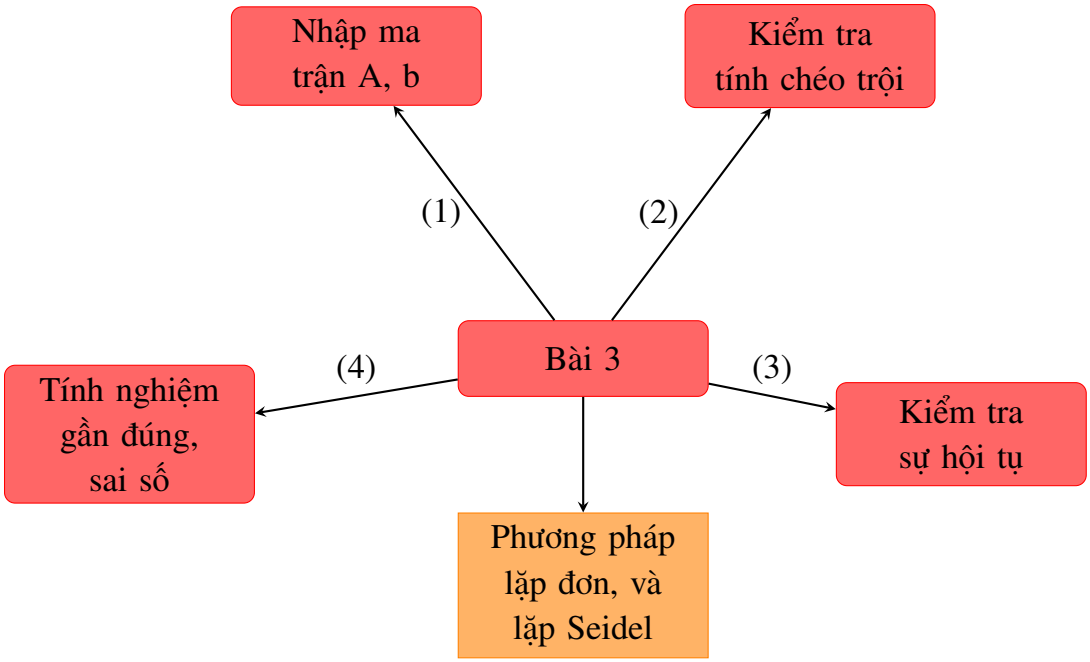
1. Viết các module. Kiểm nghiệm và sửa lỗi từng module thật kỹ trước rồi ghép và chương trình chính. Nếu có các module liên quan đến nhau, giao tiếp với nhau(một cách dễ hiểu là trong hàm này, có lệnh gọi đến hàm khác) thì viết chúng trong cùng 1 chương trình để kiểm nghiệm tính đúng đắn của các module đó. Nếu những module có thể tách thành các module khác nhỏ hơn thì viết code cho các module đó trước.

2. Viết chương trình chính, chương trình chính dùng các lệnh, và gọi lại các module theo 1 tính logic để hoàn thành bài toán. Lúc này việc sửa lỗi sẽ dựa trên chương trình chính và kết quả khi chạy chương trình.
   Ngược lại, nếu ta viết chương trình chính trước, thì sẽ khó có thể hình dung được kết quả trả về và tính logic của chương trình chính bởi vì lúc này chương trình không chạy được vì thiếu các module (Nhóm phản biện lại quan điểm viết chương trình chính trước, module sau trong Slide của giảng viên).

Chương 2

Phương pháp tinh chỉnh từng bước

2.1 Module hóa bài toán - thiết kế kiểu top-down



Hình 2.1: Các module chính của bài toán

Bài toán được chia thành 4 module và mặc định có 2 module xử lý chính của bài toán là phương pháp lặp đơn, lặp Seidel.

Module (1) được chia làm 2 module nhỏ hơn là nhập ma trận từ bàn phím và từ tệp văn bản.

Module (2) Giữ nguyên do không phân tách thành module nhỏ hơn.

Module (3) để giải được module này và các module phía sau nữa thì phải tạo ra thêm 2 module nữa là tính chuẩn của ma trận và biến đổi ma trận.

Module (4) không phân tách thành module nhỏ hơn nhưng do yêu cầu bài toán,

nhóm tách thành 3 module con là nghiệm gần đúng với số lần lặp k cho trước, đánh giá sai số, với sai số cho trước, thỏa mãn điều kiện $\left\|X^{(k)} - X^{(k-1)}\right\| \leq e$ cho trước.

Ngoài ra, ta cần viết thêm các module theo yêu cầu thêm của bài toán:

- In số chữ thập phân nhập từ bàn phím.
- Điều khiển chương tình bằng menu (điều khiển này được viết trong chương trình chính)
- In kết quả ra tệp văn bản (Nhóm thống nhất ghép chung vào module in kết quả vì nó thuận tiện và không đáng phải tách ra module riêng).

2.2 Chi tiết hóa dần Module

Việc module hóa bài toán ở mục (2.1) đã tạo ra khoảng trên 10 module nhỏ, việc chi tiết hóa từng module ở mục này sẽ đi từ mô tả input, output, và giả code của từng module đến module hoàn thiện.

2.2.1 Module xử lý chính

Input:

- Mảng 2 chiều a: chứa ma trận T, c sau khi biến đổi ($x = Tx + c$)
- Mảng 2 chiều result: trống
- Số nghiệm của hệ phương trình: n
- Số lần lặp: k

Output: Không trả về giá trị, kết quả xử lý truyền vào mảng 2 chiều result và được in ra bởi hàm in kết quả

Mô tả:

- gán cột đầu tiên của ma trận kết quả bằng ma trận c (xấp xỉ đầu $X^{(0)} = c$)
- duyệt theo cột của ma trận kết quả tới số k
- nhân ma trận T với ma trận kết quả này rồi cộng với ma trận c và trả về ma trận kết quả đứng sau nó (tính theo công thức lặp 1.3).
- Ma trận kết quả có số hàng là số nghiệm, số cột là số lần lặp k.

Hình 2.2 là module hoàn thiện của phương pháp lặp đơn.

```
void iterative_method (double a[][MAX], double result[][MAX], int n,
    int k)
{
    for (int i = 1; i <= n; i++)
        result[i][0] = a[i][n+1];
    for (int h = 1; h <= k; h++) {
        for (int i = 1; i <= n; i++) {
            result[i][h] = 0;
            for (int j = 1; j <= n; j++)
                result[i][h] += (a[i][j] * result[j][h-1]);
            result[i][h] += a[i][n+1];
        }
    }
}
```

Hình 2.2: Phương pháp lặp đơn

Input:

- Mảng 2 chiều a: chứa ma trận T, c sau khi biến đổi ($x = Tx + c$)
- Mảng 2 chiều result: trống
- Số nghiệm của hệ phương trình: n
- Số lần lặp: k

Output: Không trả về giá trị, kết quả xử lý truyền vào mảng 2 chiều result và được in ra bởi hàm in kết quả.

Mô tả:

- gán cột đầu tiên của ma trận kết quả bằng ma trận c (xấp xỉ đầu $X^{(0)} = c$)
- Các nghiệm x_1 (hàng 1 của ma trận kết quả) được tính như phương pháp lặp phía trên
- Duyệt dòng thứ 2 tới n của ma trận kết quả có hai vòng for bên trong để tính 2 cái tổng $\sum$ của công thức lặp (1.4)
- Ma trận kết quả có số hàng là số nghiệm, số cột là số lần lặp k.

Hình 2.3 là module hoàn thiện của phương pháp lặp Seidel

```
void seidel_iterative_method (double a[][MAX], double result[][MAX],
    int n, int k)
{
    for (int i = 1; i <= n; i++)
        result[i][0] = a[i][n+1];
    for (int h = 1; h <= k; h++) {
        result[1][h] = result[1][0];
        for (int j = 1; j <= n; j++)
            result[1][h] += a[1][j] * result[j][h-1];
        for (int i = 2; i <= n; i++) {
            result[i][h] = result[i][0];
            for (int j = 1; j <= i-1; j++)
                result[i][h] += a[i][j] * result[j][h];
            for (int t = i; t <= n; t++)
                result[i][h] += a[i][t] * result[t][h-1];
        }
    }
}
```

Hình 2.3: Phương pháp lặp Seidel

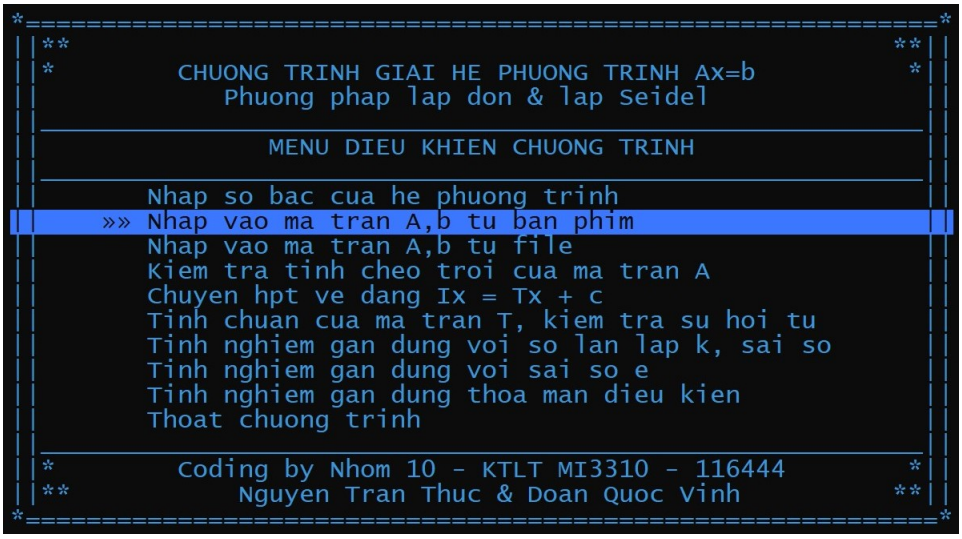
Chương 3

Đánh giá kết quả thu được

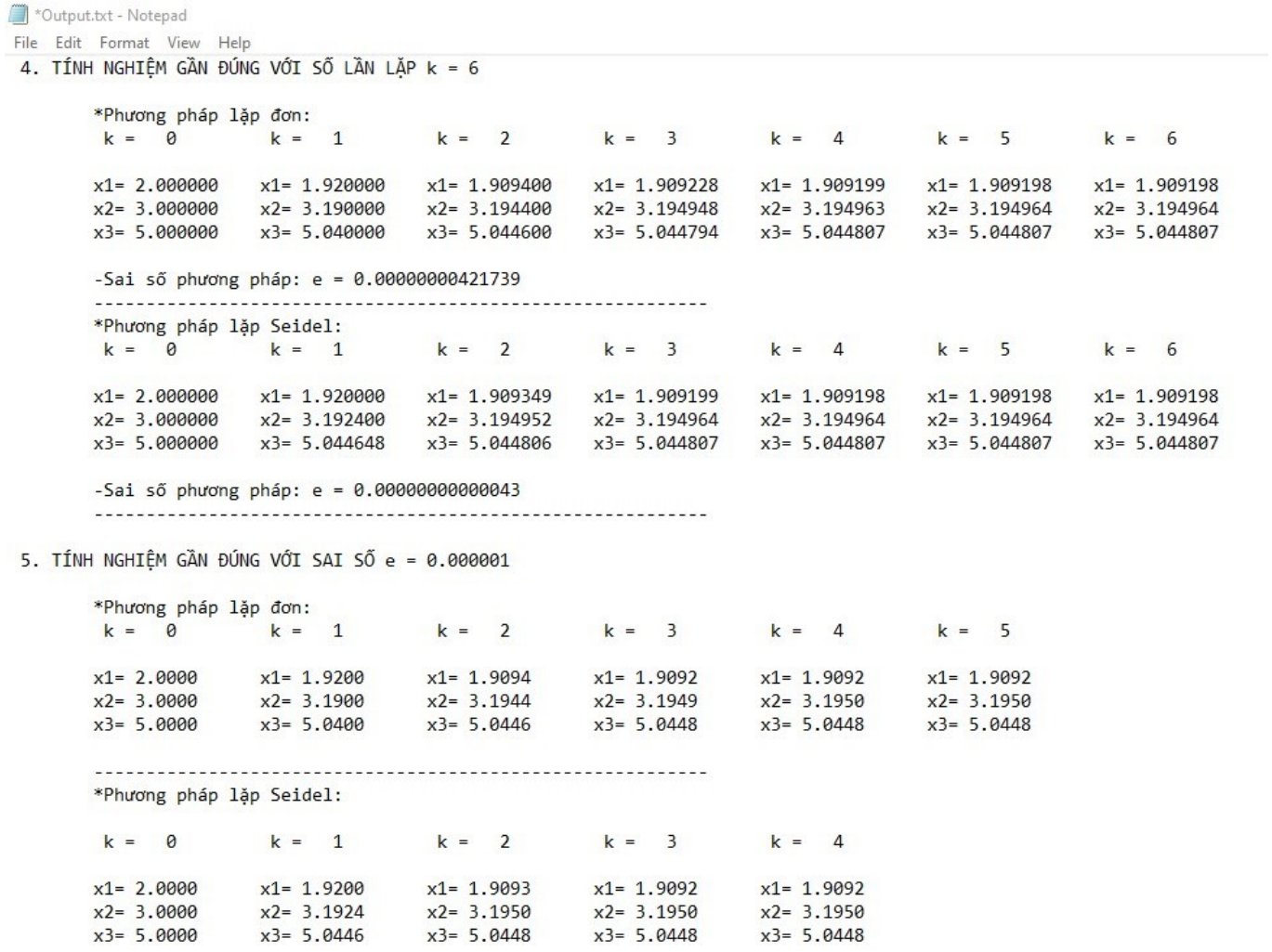
3.1 Tính đúng đắn của kết quả

3.2 Đánh giá về phong cách lập trình

3.3 Hình ảnh kết quả thu được



Hình 3.1: Menu điều khiển chương trình



*Output.txt - Notepad

File Edit Format View Help

```
4. TÍNH NGHIỆM GẦN ĐÚNG VỚI SỐ LẦN LẶP k = 6

      *Phương pháp lặp đơn:
       k =   0        k =   1        k =   2        k =   3        k =   4        k =   5        k =   6

      x1= 2.000000    x1= 1.920000    x1= 1.909400    x1= 1.909228    x1= 1.909199    x1= 1.909198    x1= 1.909198
      x2= 3.000000    x2= 3.190000    x2= 3.194400    x2= 3.194948    x2= 3.194963    x2= 3.194964    x2= 3.194964
      x3= 5.000000    x3= 5.040000    x3= 5.044600    x3= 5.044794    x3= 5.044807    x3= 5.044807    x3= 5.044807

      -Sai số phương pháp: e = 0.00000000421739
      ------------------------------------------------------------
      *Phương pháp lặp Seidel:
       k =   0        k =   1        k =   2        k =   3        k =   4        k =   5        k =   6

      x1= 2.000000    x1= 1.920000    x1= 1.909349    x1= 1.909199    x1= 1.909198    x1= 1.909198    x1= 1.909198
      x2= 3.000000    x2= 3.192400    x2= 3.194952    x2= 3.194964    x2= 3.194964    x2= 3.194964    x2= 3.194964
      x3= 5.000000    x3= 5.044648    x3= 5.044806    x3= 5.044807    x3= 5.044807    x3= 5.044807    x3= 5.044807

      -Sai số phương pháp: e = 0.00000000000043
      ------------------------------------------------------------

5. TÍNH NGHIỆM GẦN ĐÚNG VỚI SAI SỐ e = 0.000001

      *Phương pháp lặp đơn:
       k =   0        k =   1        k =   2        k =   3        k =   4        k =   5

      x1= 2.0000      x1= 1.9200      x1= 1.9094      x1= 1.9092      x1= 1.9092      x1= 1.9092
      x2= 3.0000      x2= 3.1900      x2= 3.1944      x2= 3.1949      x2= 3.1950      x2= 3.1950
      x3= 5.0000      x3= 5.0400      x3= 5.0446      x3= 5.0448      x3= 5.0448      x3= 5.0448

      ------------------------------------------------------------
      *Phương pháp lặp Seidel:

       k =   0        k =   1        k =   2        k =   3        k =   4

      x1= 2.0000      x1= 1.9200      x1= 1.9093      x1= 1.9092      x1= 1.9092
      x2= 3.0000      x2= 3.1924      x2= 3.1950      x2= 3.1950      x2= 3.1950
      x3= 5.0000      x3= 5.0446      x3= 5.0448      x3= 5.0448      x3= 5.0448
```

Hình 3.2: Kết quả in ra tệp văn bản