Mục lục

Bài 1

Thực hiện một số phép tính trên LC-3 ISA

1.1 Giới thiệu

– Với mục tiêu giúp cho sinh viên nắm vững ngôn ngữ LC3 bài tập này nhằm giúp cho sinh viên làm quen với một số lệnh của ngôn ngữ LC3.

– Trong các chương trình ứng dụng về máy tính số học, chúng ta thấy người sử dụng thường thao tác các con số với các phép tính cơ bản như cộng, trừ, nhân, chia,… Để bước đầu làm quen với kiến trúc tập lệnh LC-3 (LC3- ISA) trong môn Hệ thống máy tính và ngôn ngữ C, bài tập này yêu cầu sinh viên xây dựng một chương trình được viết dưới dạng mã nhị phân mô phỏng một máy tính thực hiện các phép tính cộng, trừ, dịch trái bit và phép luận lý OR hai giá trị đã được lưu trữ sẵn trên bộ nhớ của máy tính bằng tập lệnh LC-3.

– Chương trình máy tính này gồm có bốn chức năng. Các giá trị đầu vào của chương trình được cung cấp sẵn trên bộ nhớ của máy tính gồm có: giá trị 1 (GiaTri1**)**, giá trị 2 (GiaTri2**)** và phép tính (PhepTinh**)** như được thấy ở bảng 1, với PhepTinh cho biết phép tính nào sẽ được thực hiện đối với hai giá trị: GiaTri1 và GiaTri2. Kết quả sau khi thực hiện phép tính được lưu trữ trở lại bộ nhớ của máy tính tại ô nhớ có địa chỉ x2FF3.

Nhập/Xuất	Vị trí ô nhớ
PhepTinh	x2FF0
GiaTri1	x2FF1
GiaTri2	x2FF2
KetQua	x2FF3

Bảng 1: Vị trí ô nhớ của thao tác nhập và xuất

Phép tính	PhepTinh	Minh họa
Phép cộng	x0000	GiaTri1 + GiaTri2
Phép trừ	x0001	GiaTri1 – GiaTri2
Dịch trái	x0002	Dịch trái (GiaTri1) GiaTri2 bit
OR	x0003	GiaTri1 OR GiaTri2

Bảng 2: Các phép tính

Ví dụ 1:	
PhepTinh	x0000
GiaTri1	x0133
GiaTri2	x0124
KetQua	x0257

Ví dụ 2:	
PhepTinh	x0001
GiaTri1	x0133
GiaTri2	x0124
KetQua	x000F

Ví dụ 3:	
PhepTinh	x0002
GiaTri1	x0133
GiaTri2	x0003
KetQua	x0998

Ví dụ 4:	
PhepTinh	x0003
GiaTri1	x0133
GiaTri2	x0124
KetQua	x0137

1.2 Chuẩn bị về ngôn ngữ LC-3

Để thực hiện tốt bài tập này, sinh viên cần xem lại tập lệnh LC-3, bao gồm các nhóm lệnh sau:

- Các nhóm lệnh thi hành: ADD, AND, NOT.
- Các nhóm lệnh di chuyển dữ liệu giữa bộ nhớ và thanh ghi: LD, LDI, LDR, ST, STI, STR, LEA…
- Các nhóm lệnh liên quan đến thao tác điều khiển thứ tự thực hiện lệnh: BR, JMP, JSR, …

1.3 Các yêu cầu khi thiết kế chương trình

– Chương trình phải bắt đầu tại ô nhớ có địa chỉ 0x3000. Dòng đầu tiên trong tập tin .bin xác định địa chỉ bắt đầu của chương trình (x3000) ở dạng nhị phân.

– Chương trình phải được viết bằng ngôn ngữ máy LC-3 (mỗi lệnh được viết ở dạng nhị phân).

– Chương trình cần phải được chú thích rõ ràng. Chú thích được bắt đầu bằng dấu ‘;’. Đối với từng đoạn lệnh và mỗi dòng lệnh nên chú thích nêu rõ mục đích và cách thức giải quyết vấn đề.

– Chương trình kết thúc bằng lệnh HALT (mã thập lục phân xF025)

– Dùng phím ‘space’ để ngăn cách các phần trong mã nhị phân.

– Không cần xử lý các trường hợp tràn số đối với phép cộng hoặc phép trừ.

– GiaTri2 đối với phép dịch trái là một số có giá trị từ 0 đến 15, xác định số lượng bit cần dịch trái đối với GiaTri1. Các bit thêm vào bên phải GiaTri1 khi dịch trái luôn là 0.

– PhepTinh chỉ chọn 1 trong 4 giá trị: x0000, x0001, x0002 hay x0003, ngoài ra không có giá trị nào khác.

Gợi ý: Trước tiên chương trình nên nạp các giá trị cần tính và phép tính từ các ô nhớ vào các thanh ghi. Sinh viên lập trình đối với từng phép tính cộng, trừ, dịch trái và OR, cuối cùng sử dụng nhóm lệnh điều khiển để điều khiển cho phù hợp yêu cầu tính toán.

Bài 2

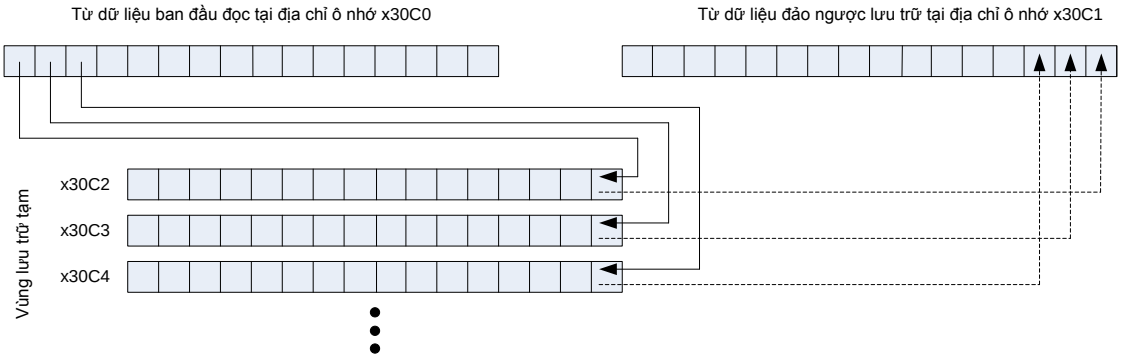
Thao tác với các bit trong LC3

2.1 Giới thiệu

– Thao tác với các bit riêng lẻ bên trong một tập dữ liệu bit lớn là một công việc rất thường gặp trong kỹ thuật số hoá dữ liệu, đặc biệt khi muốn chuyển đổi một đối tượng đã có thành đối tượng mới có nhiều ưu điểm hơn đối tượng cũ như kích thước, mức độ bảo mật thông tin nhưng vẫn giữ được độ tin cậy cao. Những việc này thường xảy ra trong các thao tác mã hóa, giải mã hay đơn giản chỉ là đáp ứng một yêu cầu cụ thể nào đó của vấn đề.

– Bài tập này yêu cầu viết một chương trình sử dụng tập lệnh LC-3 đảo các bit trong một từ (16-bit) được cung cấp sẵn trong bộ nhớ của máy tính sau đó lưu kết quả thực hiện được vào một vị trí khác trên bộ nhớ. Chương trình có thể dùng các ô nhớ còn trống trên bộ nhớ sử dụng như vùng nhớ tạm để làm đơn giản vấn đề.

– Hình bên dưới minh họa cách thức giải quyết vấn đề. Đầu tiên chúng ta nên chia từ (16 bit) dữ liệu được cung cấp tại ô nhớ x30C0 thành các bit riêng lẻ (chúng ta cần tìm cách xác định giá trị của mỗi bit mà không dùng quá nhiều lệnh). Mỗi bit nên được lưu tại một trong các ô nhớ tạm như được thấy trong hình vẽ, với bit đầu tiên/ bit dấu (mở rộng thành 16 bit bằng cách thêm các bit 0 ở đầu) được lưu tại ô nhớ x30C2, bit thứ hai ở ô nhớ x30C3….Sau khi lắp đầy vùng nhớ tạm theo cách này, chúng ta có thể lấy các bit theo thứ tự ngược lại thành một phiên bản bit đảo ngược của từ ban đầu. Cách thu thập các bit dùng như cách kiểm tra chúng. Cuối cùng lưu kết quả tại ô nhớ x30C1 và thực hiện lệnh TRAP để kết thúc chương trình.



Địa chỉ ô nhớ	Giá trị ví dụ	Mô tả
x30C0 (giá trị được cung cấp)	x5237	Từ dữ liệu cần chuyển đổi
x30C1 (tạo ra bởi chương trình)	xEC4A	Từ sau khi đảo ngược bit
x30C2 (tạo ra bởi chương trình)	x0000	Bit đầu tiên (MSB) của từ
x30C3 (tạo ra bởi chương trình)	x0001	Bit thứ hai của từ
x30C4 (tạo ra bởi chương trình)	x0000	Bit thứ ba của từ
… (tạo ra bởi chương trình)	…	Các bit tiếp theo
x30D1 (tạo ra bởi chương trình)	x0001	Bit cuối cùng (LSB) của từ

2.2 Chuẩn bị về ngôn ngữ LC-3

Sinh viên cần xem lại các nhóm lệnh sau trong ngôn ngữ LC3

- Các nhóm lệnh thi hành: ADD, AND, NOT.
- Các nhóm lệnh di chuyển dữ liệu giữa bộ nhớ và thanh ghi: LD, LDI, LDR, ST, STI, STR, LEA…
- Các nhóm lệnh liên quan đến thao tác điều khiển thứ tự thực hiện lệnh: BR, JMP, JSR, …

2.3 Các yêu cầu khi thiết kế chương trình

– Chương trình phải bắt đầu tại ô nhớ có địa chỉ x3000.

– Lệnh cuối cùng trong chương trình phải là HALT (TRAP x25).

– Chương trình phải thực hiện từ dữ liệu cần chuyển đổi tại ô nhớ có địa chỉ x30C0. Từ sau khi chuyển đổi phải được đặt tại ô nhớ có địa chỉ x30C1.

– Chương trình phải lắp đầy các ô nhớ tạm có địa chỉ từ x30C2 đến x30D1 cho biết các bit của từ dữ liệu ban đầu. Mỗi bit của từ dữ liệu ban đầu phải được đặt tại vị trí thấp nhất của ô nhớ hiện thời và thêm các bit 0 ở các vị trí còn lại để lắp đầy ô nhớ.

– Chương trình chỉ được phép thay đổi trên các ô nhớ có địa chỉ từ x30C1 đến x30D1.

– Chương trình nên được chú thích rõ ràng. Mỗi đoạn lệnh và mỗi dòng lệnh nên có chú thích cho biết mục đích và hướng giải quyết vấn đề.

Bài 3

GIẢI PHƯƠNG TRÌNH

3.1 Lý thuyết

3.1.1 Giải phương trình bậc hai tổng quát

Xét bài toán

$$ax^2 + bx + c = 0$$

trong đó: a,b,c là 3 hệ số của phương trình, x là nghiệm cần tìm.

– Khi $a = 0$ thì phương trình suy biến về bậc nhất, có dạng

$$bx + c = 0$$

+ Nếu $b = 0$ thì phương trình tùy thuộc vào c, nếu c là bằng 0 thì phương trình là vô số nghiệm, còn nếu c khác 0 thì phương trình vô nghiệm.

+ Nếu $b \neq 0$ thì phương trình sẽ có nghiệm là $x = -c/b$

– Khi $a \neq 0$ thì phương trình bậc hai sẽ có nghiệm được tính qua:

$$delta = b^2 - 4ac$$

Như vậy, có thể có ba trường hợp xảy ra cho delta:

+ Nếu delta > 0 phương trình có hai nghiệm phân biệt:

$$x_{1,2} = \frac{-b \pm \sqrt{delta}}{2a}$$

+ Nếu delta = 0 phương trình có nghiệm kép

$$x_{1,2} = -b/2a$$

+ Còn nếu delta < 0 thì phương trình có hai nghiệm phức:

$$x_1 = \frac{-b + i\sqrt{-delta}}{2a}$$

$$x_2 = \frac{-b - i\sqrt{-delta}}{2a}$$

Như vậy x_1 và x_2 là hai số phức có phần thực và ảo là :

$$\mathrm{Re}(x_1) = -b/2a$$

$$\mathrm{Im}(x_1) = -\sqrt{-delta}/2a$$

và

$$\mathrm{Re}(x_2) = -b/2a$$

$$\mathrm{Im}(x_2) = -\sqrt{-delta}/2a$$

3.1.2 Giải gần đúng các phương trình đại số hoặc siêu việt

3.1.2.1 Đặt bài toán và các bước tiến hành

Xét phương trình

$$f(x) = 0 \qquad (3.1)$$

trong đó: f(x) là hàm cho trước của biến x, ta cần tìm giá trị gần đúng của các nghiệm thực của phương trình (3.1).

Quá trình giải gần đúng phương trình (3.1) thường được chia làm 2 bước:

- Bước sơ bộ tìm nghiệm thô
- Bước kiện toàn nghiệm hay bước đi tìm nghiệm gần đúng của phương trình

1- Bước giải sơ bộ

Bước này có 3 nhiệm vụ là: vây nghiệm, tách nhiệm và thu hẹp khoảng chứa nghiệm.

– Vây nghiệm là tìm xem các nghiệm của phương trình nằm trên những khoảng nào của trục Ox.

– Tách nghiệm là xem trong các khoảng chứa nghiệm để sao cho "trong mỗi khoảng chỉ có đúng một nghiệm".

– Thu hẹp khoảng chứa nghiệm là làm sao cho khoảng chứa nghiệm càng nhỏ càng tốt.

Sau bước giải sơ bộ, ta có được khoảng chứa nghiệm "đủ nhỏ". Các bước sơ bộ thường được làm bằng tay mà không cần dùng đến máy tính.

2- Bước kiện toàn nghiệm

Ở bước này ta sẽ tìm các nghiệm gần đúng theo yêu cầu đặt ra. Các yêu cầu thường là

- Tìm xấp xỉ thứ n_o nào đó với n_o cho trước.
- Tìm xấp xỉ của nghiệm với sai số tuyệt đối, tương đối được định trước.

Bước kiện toàn nghiệm sẽ được thực hiện bằng chương trình trên máy điện toán. Ở đây ta gặp khái niệm xấp xỉ, ta có thể hiểu rõ hơn về nó như sau:

Xấp xỉ, hội tụ

Xét bài toán

$$y = Bx$$

Giả sử y^* là nghiệm đúng của bài toán mà ta chưa biết. Bằng một phương pháp nào đó, người ta lấy y_1 thay cho y^*, khi đó y_1 gọi là xấp xỉ thứ nhất của nghiệm và viết

$$y_1 \approx y^*$$

Cũng bằng phương pháp tương tự, chúng ta xây dựng được một dãy các xấp xỉ $y_1, y_2, y_3, ..., y_n$

Ký hiệu $\{y_n\}_1^\infty$ và viết $y_n \approx y^*$

Nếu $\lim_{n\to\infty} y_n = y^*$, ta nói dãy xấp xỉ mà ta xây dựng được ở trên $\{y_n\}_1^\infty$ hội tụ tới nghiệm đúng y^*.

Với mọi epsilon > 0 tồn tại ít nhất N(ε) sao cho:

Với mọi $n > N(\varepsilon) \to |y_n - y^*| < \varepsilon$

Hiển nhiên trong trường hợp $\{y_n\}_1^\infty$ hội tụ tới y^* ta có

$$y^n - y^* = 0(h^k)$$

trong đó $0(h^k)$ với k > 0 là một số rất bé (vô cùng bé) cấp h^k, $0(h^k) \to 0$ khi $n \to \infty$

Giả sử lại có dãy xấp xỉ $\{Z_n\}_1^\infty$ cũng hội tụ về y^* và như vậy

$$Z_n - y^* = 0(h^S), S > 0$$

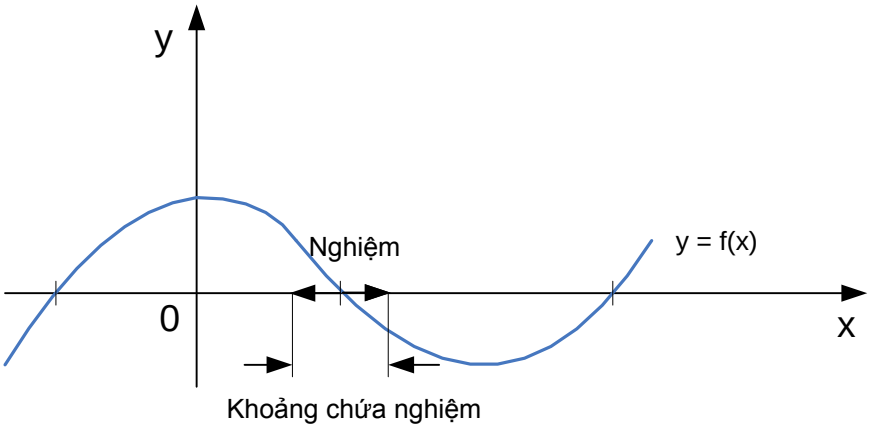
Nếu s > k, ta nói dãy xấp xỉ có tốc độ hội tụ tới nghiệm đúng nhanh hơn dãy xấp xỉ $\{y_n\}_1^\infty$

Việc xây dựng dãy xấp xỉ có tốc độ hội tụ nhanh là việc rất được chú trọng trong phương pháp tính, nó ảnh hưởng trực tiếp đến độ chính xác của bài toán đặt ra.

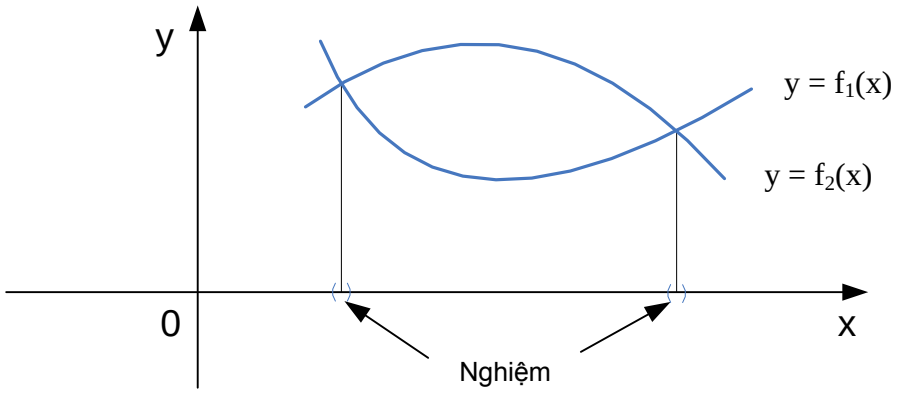
3.1.2.2 Bước giải sơ bộ bài toán *(tìm nghiệm thô)*

1- Phương pháp đồ thị

Vẽ đồ thị hàm $y = f(x)$ trên hệ trục tọa độ Decac Oxy. Căn cứ vào giao điểm của đồ thị với trục hoành Ox ta xác định được số nghiệm và các khoảng chứa nghiệm.



Đối với trường hợp hàm $f(x)$ khá phức tạp, khó vẽ đồ thị, ta tìm cách đưa phương trình $f(x) = 0$ về dạng: $f_1(x) = f_2(x)$ trong đó $f_1(x), f_2(x)$ có thể vẽ đồ thị dễ dàng. Căn cứ vào giao điểm của hai đồ thị trên hệ tọa độ ta xác định số nghiệm và các khoảng chứa nghiệm.



Phương pháp đồ thị có ưu điểm là có hình ảnh trực quan song không chính xác.

2- Phương pháp giải tích

Chúng ta xem hai định lý sau đây là hiển nhiên

Định lý 1: nếu hàm số f(x) liên tục trên đoạn [a, b], f(a).f(b) < 0 thì phương trình f(x) có ít nhất một nghiệm trong (a, b).

Định lý 2: nếu các giả thiết của định lý 1 được thỏa và f′(x) không đổi dấu trên (a, b) thì phương trình f(x) = 0 có nghiệm duy nhất trong (a, b).

Phương pháp giải tích thường được dùng để kiểm tra kết quả của phương pháp đồ thị.

Ví dụ 3.1. Xét phương trình $f(x) = x^4 - 4x + 1$, tìm nghiệm f(x) = 0.

Ta thấy,

$f(0) = 1 > 0$

$f(1) = 1 - 4 + 1 < 0$

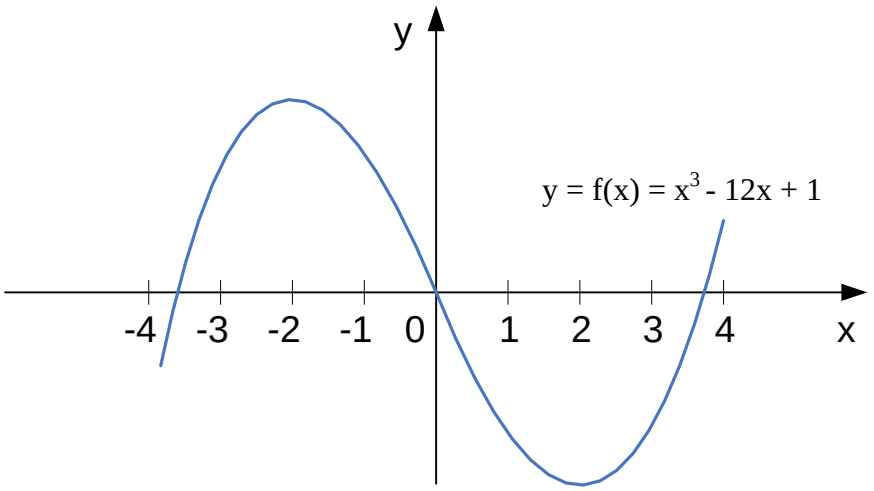
$f(2) = 16 - 8 + 1 > 0$

Vậy phương trình sẽ có nghiệm trong khoảng (0, 1) và (1, 2)

Ví dụ 3.2. Xét phương trình $f(x) = x^3 - 12x + 1$. Ta hãy tìm nghiệm của nó (f(x) = 0). Ta có

$f'(x) = 3x^2 - 12 = 0 \rightarrow x = \pm 2$

Vậy f(x) có hai cực trị tại x = –2 và x = +2



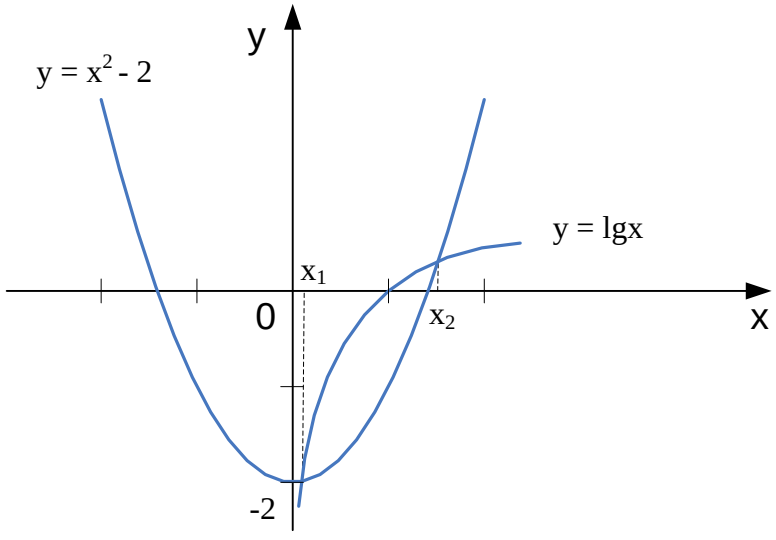
f(2) = 8 – 24 + 1 < 0
f(–2) = –8 + 24 + 1 > 0
f(–3) > 0 và f(–4) < 0, f(–3).f(–4) < 0 → có nghiệm trong (–4, –3)
f(0) > 0 và f(1) < 0 → có nghiệm trong (0, 1)
f(3) < 0 và f(4) > 0 → có nghiệm trong (3, 4)

Ví dụ 3.3. Thực hiện bước giải sơ bộ đối với phương trình

$$x^2 - 2 - \lg x = 0$$

Giải: đưa phương trình về dạng $x^2 - 2 = \lg x$

Vẽ đồ thị của hàm $y = x^2 - 2$, $y = \lg x$ trên cùng một hệ tọa độ



Nhìn vào đồ thị, thấy phương trình có hai nghiệm x_1, x_2. Cụ thể x_1 thuộc $\left(\frac{1}{100}, \frac{1}{10}\right)$ và x_2 thuộc (1, 2)

Ta có thể kiểm tra kết quả của phương pháp đồ thị này lại bằng phương pháp giải tích.

Gọi $f(x) = x^2 - 2 - \lg x$

$f(x)$ liên tục trên R^+ $(0 \to \infty)$

$$f\left(\frac{1}{100}\right) = \left(\frac{1}{100}\right)^2 - 2 - \lg\frac{1}{100} = \left(\frac{1}{100}\right)^2 - 2 - (-2) = \left(\frac{1}{100}\right)^2 > 0$$

$$f\left(\frac{1}{10}\right) = \left(\frac{1}{10}\right)^2 - 2 - \lg\frac{1}{10} = \frac{1}{100} - 2 - (-1) = \frac{1}{100} - 1 < 0$$

$f(1) = -1 < 0$

$f(2) = 2 - \lg 2 > 0$

Vậy, kết quả thu được là đúng.

3- Phương pháp thu hẹp khoảng chứa nghiệm

Giả sử x* là nghiệm duy nhất của phương trình f(x) = 0 trên (a, b), nghĩa là ta tìm được x* trong khoảng (a, b) sau khi qua hai bước vây nghiệm và tách nghiệm.

Gọi $c = \frac{a+b}{2}$ (điểm giữa đoạn [a, b])

Theo định lý 1, nếu

- f(c).f(a) < 0 thì x* thuộc (a, c)
- f(c).f(a) > 0 thì x* thuộc (c, b)
- f(c).f(a) = 0 thì x* = c

Phương pháp trên gọi là phương pháp chia đôi khoảng. Mỗi lần áp dụng phương pháp này, khoảng chứa nghiệm "bé đi" một nữa. Như vậy, ta có thể làm cho khoảng chứa nghiệm "đủ nhỏ".

Phương pháp thu hẹp khoảng chứa nghiệm này nằm trong bước giải sơ bộ, Như ta đã nói, các công việc ở bước sơ bộ thường làm bằng tay (thủ công) và như vậy phương pháp thu hẹp này cũng được giải bằng tay. Tuy nhiên, vào lúc máy vi tính "bùng nổ" như hiện nay, ta vẫn nên làm bước này ở trên máy tính.

3.1.2.3 Bước kiện toàn nghiệm

Ở bước này nghiệm gần đúng của phương trình sẽ được tìm bằng chương trình trên máy tính. Có hai phương pháp được nêu trong chương trình học.

- Phương pháp dây cung
- Phương pháp tiếp tuyến

Mỗi phương pháp sẽ đưa đến chương trình và giải thuật khác nhau và sẽ được trình bày chi tiết trong mục Giải thuật và minh họa bằng C (nếu có).

3.2 Chuẩn bị về ngôn ngữ C

Để thiết kế tốt chương trình các kiến thức về ngôn ngữ C cần nắm:

- Cấu trúc tổng quát một chương trình C, nơi đặt hàm, khai báo biến, …
- Cách nhập dữ liệu có có kiểm tra: hàm scanf, ...

- Xuất dữ liệu theo dạng mong muốn hàm printf, ...
- Các cấu trúc dữ liệu cần thiết
- Một số giải thuật cơ bản về kiểm tra số liệu nhập có thỏa yêu cầu của bài toán, một số giải thuật xuất dữ liệu theo dạng phù hợp bài toán.
- Cách sử dụng các lệnh điều khiển và vòng lặp if – else, do – while, …
- Một số hàm thư viện chuẩn sqrt, …
- Cách thiết kế hàm và cách khai báo đối số hàm, các trả trị về từ hàm.
- Trong một số trường hợp nếu cần có thể biểu diễn hàm đồ thị để minh họa, do đó việc hiển thị trong chế độ graph cũng cần được nắm chính xác, một số hàm, biến, hằng,… được dùng trong chế độ graph cần được xem lại, cách sử dụng và khai báo.

3.3 Giải thuật

3.3.1 Giải thuật giải phương trình bậc hai tổng quát

Chương trình bậc hai tổng quát sẽ có giải thuật như sau:

- Nếu a = 0: phương trình suy biến: $bx + c = 0$
  + Nếu b = 0: phương trình thành c = 0
    - Nếu c = 0: phương trình vô số nghiệm
    - Nếu c ≠ 0: phương trình vô nghiệm
  + Nếu b ≠ 0: phương trình có nghiệm $-c/b$
- Nếu a ≠ 0: $delta = b^2 - 4ac$
  + Nếu delta > 0: phương trình có hai nghiệm phân biệt $x_{1,2}$

$$x_{1,2} = \frac{-b \pm \sqrt{delta}}{2a}$$

  + Nếu delta = 0: phương trình có nghiệm kép

$$x_1 = x_2 = -b/2a$$

  + Nếu delta < 0: phương trình có hai nghiệm phức

$$x_{1,2} = \frac{-b \pm i\sqrt{-delta}}{2a}$$

3.3.2 Giải thuật các phương pháp tìm nghiệm

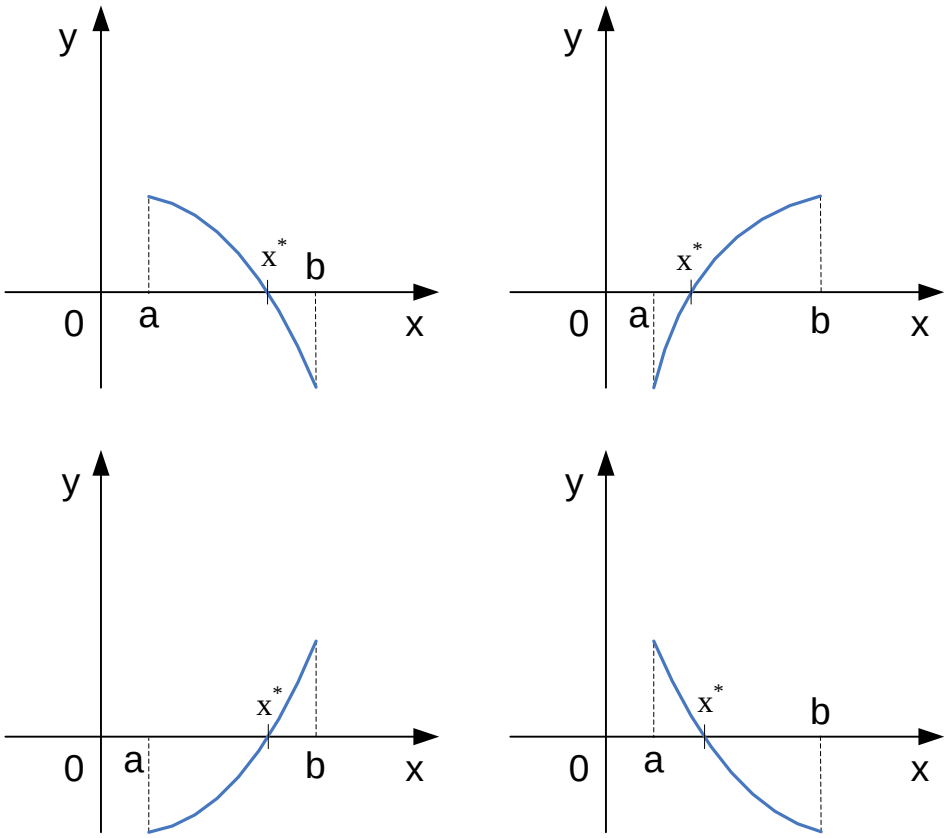
3.3.2.1 Phương pháp dây cung

Định nghĩa

Xét phương trình f(x) = 0, và x* là nghiệm được tách trong (a, b). Giả sử f, f′, f″ là các hàm liên tục trên (a, b); f′, f″ không đổi dấu trên (a, b). Điểm α thuộc (a, b) được gọi là thỏa mãn điều kiện Fourier nếu:

f(α).f″(α) > 0 thường chọn α = a hoặc α = b

Minh họa: Các điểm có đánh dấu 0 trên đồ thị có hoành độ thỏa mãn điều kiện Fourier



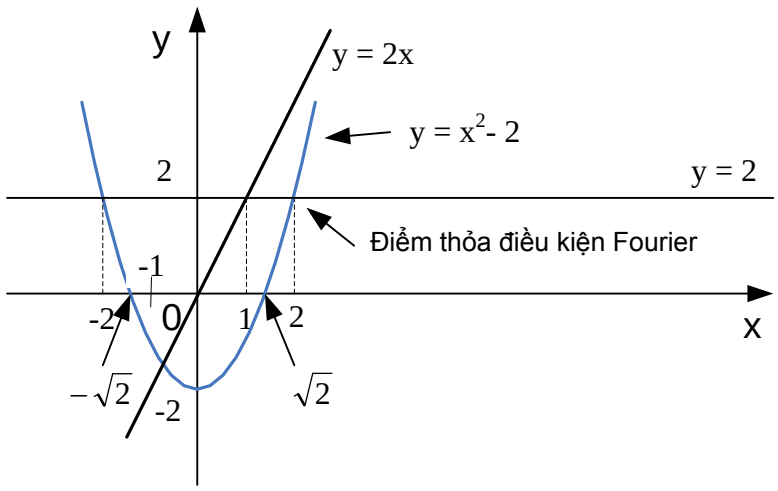
Ví dụ 3.4. Hàm số $y = x^2 - 2$

Ta có

$y = 0 \rightarrow x = \pm\sqrt{2}$

$y' = 2x$

$y'' = 2$

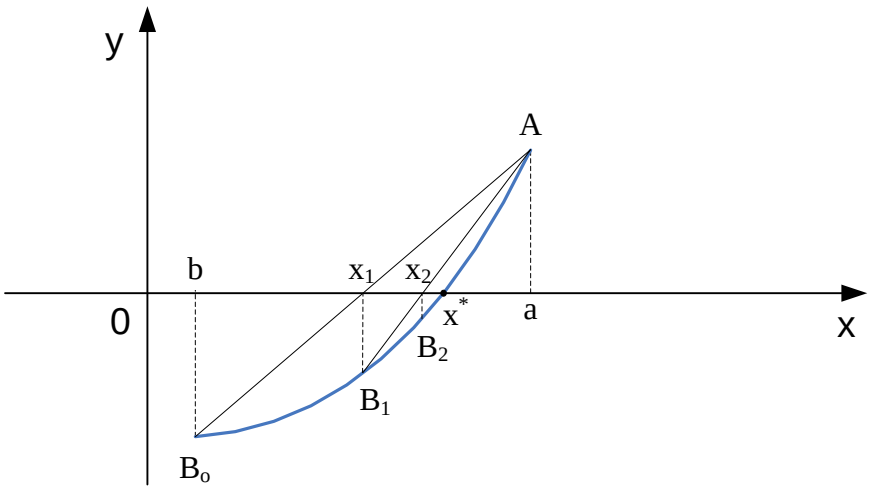


Nội dung của phương pháp dây cung là xuất phát từ điểm A thỏa mãn điều kiện Fourier có hoành độ thỏa mãn điều kiện Fourier lập các dây cung liên tiếp: AB_o, AB_1, AB_2, …, AB_n

Lập phương trình đường thẳng qua hai điểm.

$$A(a, f(a), B_{n-1}(x_{n-1}, f(x_{n-1}))$$

$$\frac{y - f(a)}{f(x_{n-1}) - f(a)} = \frac{x - a}{x_{n-1} - a} \qquad (3.2)$$



Tìm giao điểm x_n của đường thẳng (3.2) với trục hoành (cho y = 0)

$$x_n = a - \frac{f(a)}{f(x_{n-1}) - f(a)}(x_{n-1} - a) \qquad (3.3)$$

với $x_0 = b, n = 1,2,...$

Đó là công thức lập để xây dựng dãy xấp xỉ. Trong công thức thì a thỏa điều kiện Fourier $b = x_0$ là đầu nút còn lại.

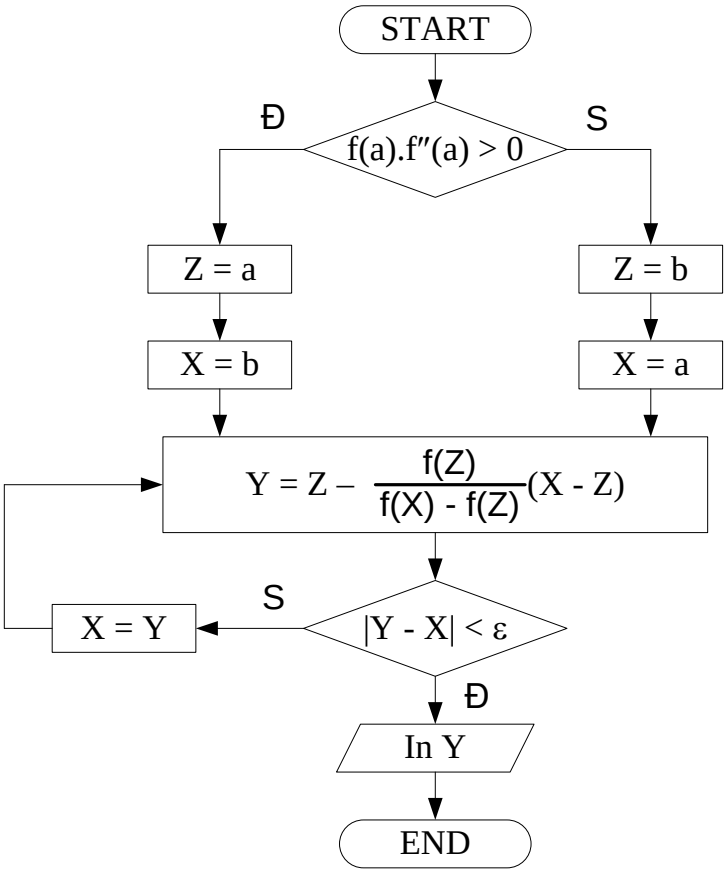
Quy đồng vế phải của (3.3) ta nhận ra rằng phương pháp dây cung chính là phương pháp lặp với:

$$\varphi(x) = \frac{a(f) - xf(a)}{f(x) - f(a)}$$

Người ta đã chứng minh được rằng nếu các giả thiết đã được nêu thỏa mãn thì dãy xấp xỉ $\{x_n\}_1^{\infty}$ sẽ hội tụ về x* và sai số của dãy xấp xỉ thứ n cho bởi công thức:

$$|x_n - x^*| \sim \left|\frac{f(x_n)}{f'(x_n)}\right|$$

Kết luận: phương pháp dây cung chính là một dạng của phương pháp lặp nhưng có tốc độ hội tụ nhanh hơn phương pháp lặp "thuần túy".



Lưu đồ của phương pháp dây cung với tham số ε cho trước

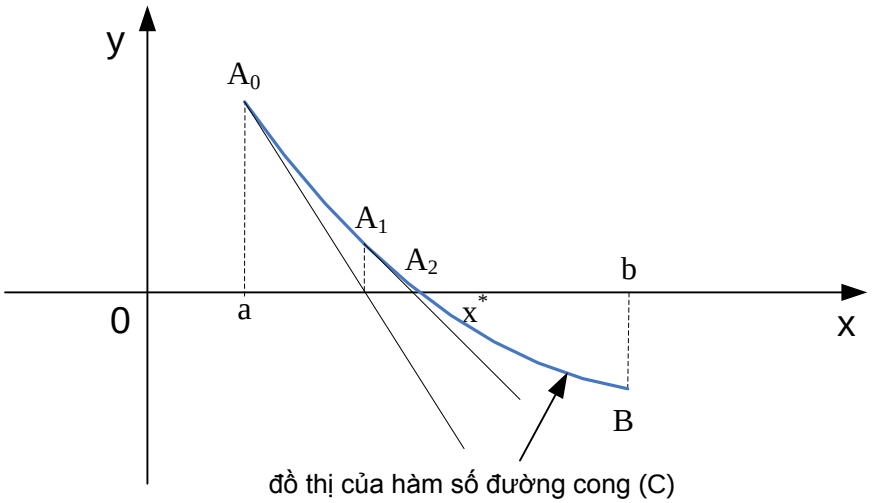
3.3.2.2 Phương pháp tiếp tuyến

Định nghĩa

Cũng với giả thiết như phương pháp dây cung, xuất phát từ một điểm A_o có hoành độ thỏa mãn điều kiện Fourier vẽ tiếp tuyến với đồ thị.

Giao điểm của tiếp tuyến với trục hoành cho ta xấp xỉ của nghiệm.

Quá trình trên tiếp tục với các tiếp tuyến tại A_1, A_2, …, A_n như hình vẽ.



Phương trình tiếp tuyến với đường cong (C) tại điểm $A_{n-1}(x_{n-1}, f(x_{n-1}))$ là

$$y - f(x_{n-1}) = f'(x_{n-1})(x - x_{n-1})$$

Từ đó ta tìm được x_n, tức là giao điểm của tiếp tuyến trên với trục hoành (cho y = 0).

$$x = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})} \qquad (3.4)$$

với n = 1, 2, … và x = a thỏa điều kiện Fourier

Người ta đã chứng minh rằng nếu thỏa mãn các giả thiết đã nêu thì dãy xấp xỉ (3.4) sẽ hội tụ về nghiệm đúng x^*.

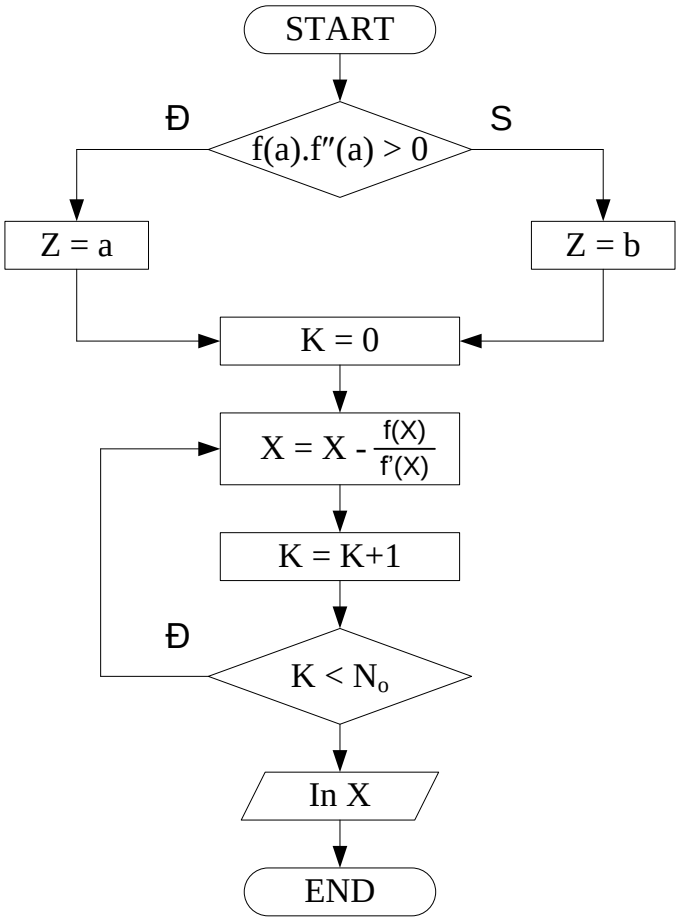
Ghi chú:

+ Phương pháp tiếp tuyến là trường hợp đặc biệt của phương pháp lặp với.

$$\varphi(x) = x - \frac{f(x)}{f'(x)}$$

+ Phương pháp tiếp tuyến có tốc độ hội tụ nhanh hơn phương pháp dây cung (*và người ta cũng chứng minh nó có* ***tốc độ hội tụ bậc hai***)

+ Các giả thiết của phương pháp dây cung và tiếp tuyến tuy nhiều nhưng dễ được thỏa mãn khi khoảng tách nghiệm (a, b) đủ nhỏ.



Lưu đồ của phương pháp tiếp tuyến với số lần lặp N_o cho trước

Ví dụ 3.5. Giải phương trình $2x-\cos x-1=0$ bằng phương pháp tiếp tuyến.

Cho điểm $A(x_o, y_o)$ thỏa điều kiện Fourier với $x_o = 0.9$ (*giả sử* x_o, y_o) thỏa điều kiện Fourier.

Từ công thức

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Ta có: $x_1 = x_0 - \dfrac{2x_0 - \cos x_0 - 1}{2 + \sin x_0} = 0.8359... = x_1$

$$x_2 = x_1 - \frac{2x_1 - \cos x_1 - 1}{2 + \sin x_1} = ... = x_2$$

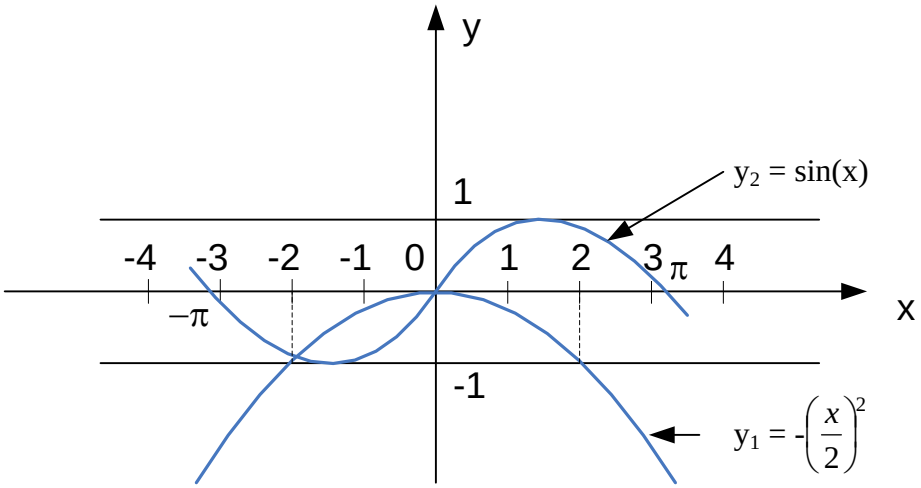
$$\vdots$$

$$x_{n+1} = x_n - \frac{2x_n - \cos x_n - 1}{2 + \sin x_n}$$

Chọn số lần lặp N_o hoặc sai số ε làm điều kiện thoát thì lúc đó ta có

$$x_{n+1} \sim x^*$$

Ví dụ 3.6. Giải phương trình $x^2 + 4\sin x = 0 \rightarrow f(x) = 0$



Ta đưa phương trình trên về dạng $\sin x = -\left(\frac{x}{2}\right)^2$

Để tìm nghiệm thô, ta vẽ đồ thị của hàm số này, giao điểm của chúng cho ta được nghiệm thô.

Trên đồ thị, ta thấy nghiệm thuộc (–2, –1,5) = (a, b) khoảng chứa nghiệm thô.

Xét điểm (–2) có thỏa điều kiện Fourier hay không

f(–2) = $(-2)^2 + 4\sin(-2) = 4 - 4.(K) = 4 - 4K > 0$ (K = sin(2) < 1)

f′(–2) = $(2x + 4\cos x)' \rightarrow$ điểm (–2) thỏa điều kiện Fourier

Ta xây dựng dãy xấp xỉ,

$$x_n = x_{n-1} - \frac{x_{n-1}^2 + 4\sin x_{n-1}}{2x_{n-1} + 4\cos x_{n-1}}$$

với $x_o = -2$ và n = 1, 2, 3 …

$$x_1 = x_0 - \frac{x_0^2 + 4\sin x_0}{2x_0 + 4\cos x_0} = -1{,}93595\ldots$$

$$x_2 = x_1 - \frac{x_1^2 + 4\sin x_1}{2x_1 + 4\cos x_1}$$

3.4 Thiết kế chương trình

3.4.1 Thiết kết chương trình giải phương trình bậc hai tổng quát

Thiết kế chương trình giải phương trình bậc hai tổng quát bằng hai cách:

1- Chỉ dùng chương trình chính để nhập, giải và in ra màn hình (có thể biểu diễn trị trên trục tọa độ bằng hình cụ thể) các nghiệm.

2- Dùng chương trình chính để gọi các hàm nhập, giải và in ra màn hình các nghiệm.

Chú ý: khi giải theo cách 2, hàm giải phương trình bậc hai phải có trị trả về cho biết nghiệm của phương trình đang trong trường hợp nào (nghiệm kép, nghiệm phức, vô nghiệm, …).

Dùng các lệnh xuất nhập và các vòng điều khiển của ngôn ngữ C để đối thoại giữa người và máy về:

- Nhập tham số.
- Cần giải thêm một phương trình bậc hai nữa hay không ?

3.4.2 Thiết kết chương trình giải các phương pháp tìm nghiệm

Hãy thiết kế hai chương trình sau:

1- Tìm nghiệm thô của phương trình ở ví dụ 3.5: $2x - \cos x - 1 = 0$ bằng phương pháp đồ thị. Điểm $x_o = 0,8$ có thỏa mãn điều kiện Fourier hay không nếu giải bài toán này theo phương pháp dây cung ? (*hướng dẫn: đưa phương trình về dạng* $x = \varphi(x)$)

2- Vẽ lưu đồ của giải thuật giải phương trình phi tuyến. Yêu cầu: chương trình phải có tính tổng quát cho mọi dữ liệu nhập vào, nếu có dữ liệu sai thì phải có việc kiểm tra và báo lỗi.

Bài 4

TÍNH GẦN ĐÚNG TÍCH PHÂN XÁC ĐỊNH BẰNG PHƯƠNG PHÁP HÌNH THANG VÀ PHƯƠNG PHÁP PARABOL

4.1 Lý thuyết

4.1.1 Công thức Newton – Leibnitz

Định lý: nếu hàm $f(x)$ liên tục trên đoạn $[a, b]$ và $F(x)$ là một nguyên hàm của nó trong đoạn đó thì

$$\int_a^b f(x)dx = F(b) - F(a) = \left[F(x)\right]_a^b$$

Đẳng thức trên được gọi là công thức Newton – Leibnitz

4.1.2 Các mệnh đề cơ bản

– Nếu f và g khả tích trên đoạn $[a, b]$ thì hàm tổng $(f + g)$ cũng khả tích trên đoạn $[a, b]$.

– c là một hằng, nếu f khả tích trên đoạn $[a, b]$ thì (c.f) cũng khả tích trên đoạn ấy.

$$\int_a^b (cf) = c\int_a^b f$$

– Nếu $\int_a^b f$ tồn tại và $a \le c \le d \le b$ thì $\int_c^d f$ tồn tại

– Cho f, g khả tích trên đoạn $[a, b]$. Nếu $f \le g$ trên $[a, b]$ thì $\int_a^b f \le \int_a^b g$

– Nếu $m \le f(x) \le M$ với mọi x trên $[a, b]$ và nếu $\int_a^b f(x)$ tồn tại thì ta có bất đẳng thức

$$m \le \frac{1}{b-a}\int_a^b f \le M$$

– Nếu f khả tích trên đoạn $[a, b]$ thì trị tuyệt đối $|f|$ cũng khả tích và ta có

$$\left|\int_a^b f\right| \le \int_a^b |f|$$

– Nếu f, g khả tích trên đoạn $[a, b]$ thì hàm tích $f.g$ cũng khả tích trên $[a, b]$
Ta có bất đẳng thức sau (*gọi là bất đẳng thức Schwartz*)

$$\left|\int_a^b f.g\right|^2 \le \int_a^b f^2 . \int_a^b g^2$$

4.2 Chuẩn bị về ngôn ngữ C

Sinh viên cần phải xem lại các vấn đề sau đây của ngôn ngữ C:

– Các hàm xuất nhập printf, scanf

– Các thao tác gán, thay đổi trị cho biến.

– Cách tạo hàm, đối số hàm, trị trả về từ hàm, prototype, …
– Cấu trúc tổng quát một chương trình C, nơi đặt hàm khai báo biến, …
– Cách sử dụng các lệnh điều khiển và vòng lặp.
– Các cấu trúc dữ liệu cần thiết.
– Một số giải thuật cơ bản về kiểm tra số liệu nhập có thỏa yêu cầu của bài toán, một số giải thuật xuất dữ liệu theo dạng phù hợp bài toán.
– Trong một số trường hợp nếu cần có thể biểu diễn hàm đồ thị để minh họa, do đó việc hiển thị trong chế độ graph cũng cần được nắm chính xác, một số hàm, biến, hằng, … được dùng trong chế độ graph cần được xem lại, cách sử dụng và khai báo.

4.3 Giải thuật chương trình

4.3.1 Giải thuật phương pháp hình thang

Cho hàm số $f(x)$ khả tích trên đoạn $[a, b]$. Nếu ta tìm được nguyên hàm $F(x)$ của nó thì công thức Newton – Leibnitz có được,

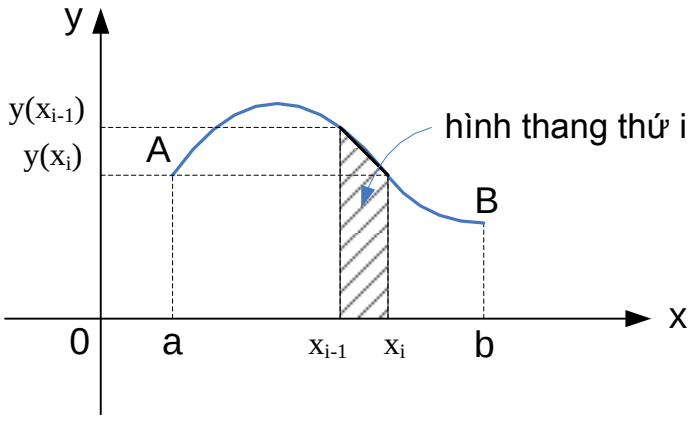
$$\int_a^b f(x)dx = F(b) - F(a)$$

Song trong rất nhiều trường hợp chúng ta không thể tìm được nguyên hàm $F(x)$ hoặc tìm được nhưng rất khó khăn, phức tạp. Vì vậy, trong một chừng mực nào đó chúng ta vẫn có thể tính được tích phân trên một cách gần đúng để tránh khỏi khó khăn trên.

Trong phần này trình bày một số phương pháp tính gần đúng tích phân xác định theo *phương pháp hình thang*.

Dựa vào ý nghĩa hình học của tích phân xác định ta thay diện tích hình thang cong aABb bằng tổng n diện tích hình thang thường.

Xét hình thang thứ i, có bề rộng là $h = \frac{b-a}{n}$



Diện tích của nó là $S_i = \frac{f(x_{i-1}) + f(x_i)}{2} h = \frac{y_{i-1} + y_i}{2} h$

Khi đó, tổng n diện tích hình thang thường là

$$I_n = \sum_{i=1}^{n} S_i = \frac{h}{2}\left[(y_o + y_n) + 2(y_1 + y_2 + ... + y_{n-1})\right]$$

$$\int_a^b f(x)dx \approx \frac{h}{2}\left[(y_o + y_n) + 2(y_1 + y_2 + ... + y_{n-1})\right]$$

Từ đó, công thức gần đúng (*diện tích hình thang cong*)

$$\int_a^b f(x)dx \approx \frac{h}{2}\left[(y_o + y_n) + 2(y_1 + y_2 + ... + y_{n-1})\right]$$

trong đó: $h = \frac{b-a}{n}$; $y_i = f(x_i)$; $x_i = a + ih (i = 0,1,2,...,n)$

$y_o = f(x_o) = f(a)$; $y_n = f(x_n) = f(b)$

Phép xấp xỉ trên càng chính xác nếu n càng lớn (h càng nhỏ) tức là khi chia càng nhiều hình thang thì sai số của phép xấp xỉ trên càng nhỏ.

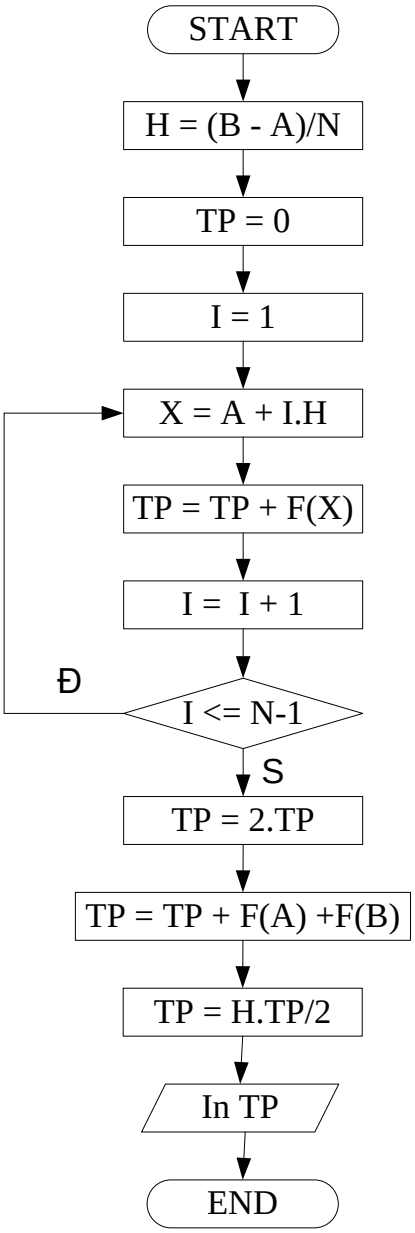
Sai số của xấp xỉ thứ n được cho bởi công thức sau

$$\left|\int_a^b f(x)dx - I_n\right| \le \frac{b-a}{12}.h^2.M$$

với: $M = \max|f(x)|$, x thuộc $[a, b]$.

Thực chất của phương pháp hình thang là thay thế cung của đường cong y = $f(x)$ trên mỗi đoạn nhỏ bởi một đoạn thẳng. Vì vậy, phương pháp này còn được gọi là phương pháp nội suy tuyến tính.

Để thiết lập công thức hình thang ta đã sử dụng ý nghĩa hình học của tích phân ($f(x) > 0$) song công thức đó cũng đúng với $f(x)$ bất kỳ.



Lưu đồ của phương pháp hình thang

Ví dụ 4.1 Tính gần đúng tích phân xác định như sau

$$\int_0^1 \frac{1}{1+x}dx$$

Ta chia đoạn [0, 1] thành 10 phần nhỏ đều nhau

$$h = \frac{1-0}{10} = 0{,}1$$

$a_o = 0$, $a_1 = 0{,}1$, …, $a_{n-1} = 0{,}9$, $a_n = 1$, $n = 10$

áp dụng công thức

$$\int_a^b f(x)dx \approx \frac{h}{2}\left[(y_o + y_n) + 2(y_1 + y_2 + ... + y_{n-1})\right]$$

Ta được,

$$\int_0^1 \frac{1}{1+x}dx \approx \left[\frac{0,1}{2}\left(\frac{1}{1}+\frac{1}{2}\right)+2\left(\frac{1}{1+0,1}+\frac{1}{1+0,2}+\frac{1}{1+0,3}+\frac{1}{1+0,4}+\frac{1}{1+0,5}+\frac{1}{1+0,6}+\frac{1}{1+0,7}+\frac{1}{1+0,8}+\frac{1}{1+0,9}\right)\right]$$

$$= 0,1\left[\frac{3}{4}+\left(\frac{1}{1+0,1}+\frac{1}{1+0,2}+\frac{1}{1+0,3}+...+\frac{1}{1+0,9}\right)\right]$$

$$= 0,1\left[0,75+\left(\frac{1}{1,1}+\frac{1}{1,2}+\frac{1}{1,3}+...+\frac{1}{1,9}\right)\right]$$

$$\rightarrow \int_0^1 \frac{1}{1+x}dx \approx 0,075+\frac{0,1}{1,1}+\frac{0,1}{1,2}+\frac{0,1}{1,3}+...+\frac{0,1}{1,9}$$

Kiểm tra lại bằng phương pháp giải tích

Phương pháp giải tích cho ta kết quả

$$\int_0^1 \frac{1}{1+x}dx \approx F(b)-F(a)=\ln(1+1)-\ln(1+0)=\ln 2$$

Kết luận: Ta nhận thấy phép gần đúng của chúng ta càng chính xác (tiến về ln2) nếu ta càng tăng số phần chia nhỏ đoạn [0, 1].

4.3.2 Giải thuật phương pháp parabol

Cho hàm số $f(x)$ khả tích trên đoạn $[a, b]$. Nếu ta tìm được nguyên hàm $F(x)$ của nó thì công thức Newton – Leibnitz có được

$$\int_a^b f(x)dx = F(b)-F(a)$$

Song trong rất nhiều trường hợp chúng ta không thể tìm được nguyên hàm $F(x)$ hoặc tìm được nhưng rất khó khăn, phức tạp. Vì vậy, trong một chừng mực nào đó chúng ta vẫn có thể tính được tích phân trên một cách gần đúng để tránh khỏi khó khăn trên.

Trong phần này trình bày một phương pháp tính gần đúng tích phân xác định: *phương pháp parabol.*

Phương pháp parabol này khác với phương pháp hình thang là ta chia đoạn [a, b] thành 2.n phần đều nhau và trên hai phần nhỏ, ta thay cung y = $f(x)$ bằng cung parabol.

Trên đoạn $[x_{2i-2}, x_{2i}]$ thay thế cung y = $f(x)$ bởi cung parabol qua ba điểm.

(x_{2i-2}, y_{2i-2}); (x_{2i-1}, y_{2i-1}); (x_{2i}, y_{2i})

Sau đó tính diện tích hình thang cong nhỏ này theo công thức

$$S_i = \frac{h}{3}(y_{2i-2}+4y_{4i-1}+y_{2i})\ ;\ (\text{i= 1, 2,…, n})$$

và vì vậy: $I_n = \sum_{i=1}^{n} S_i$

Cuối cùng ta có:

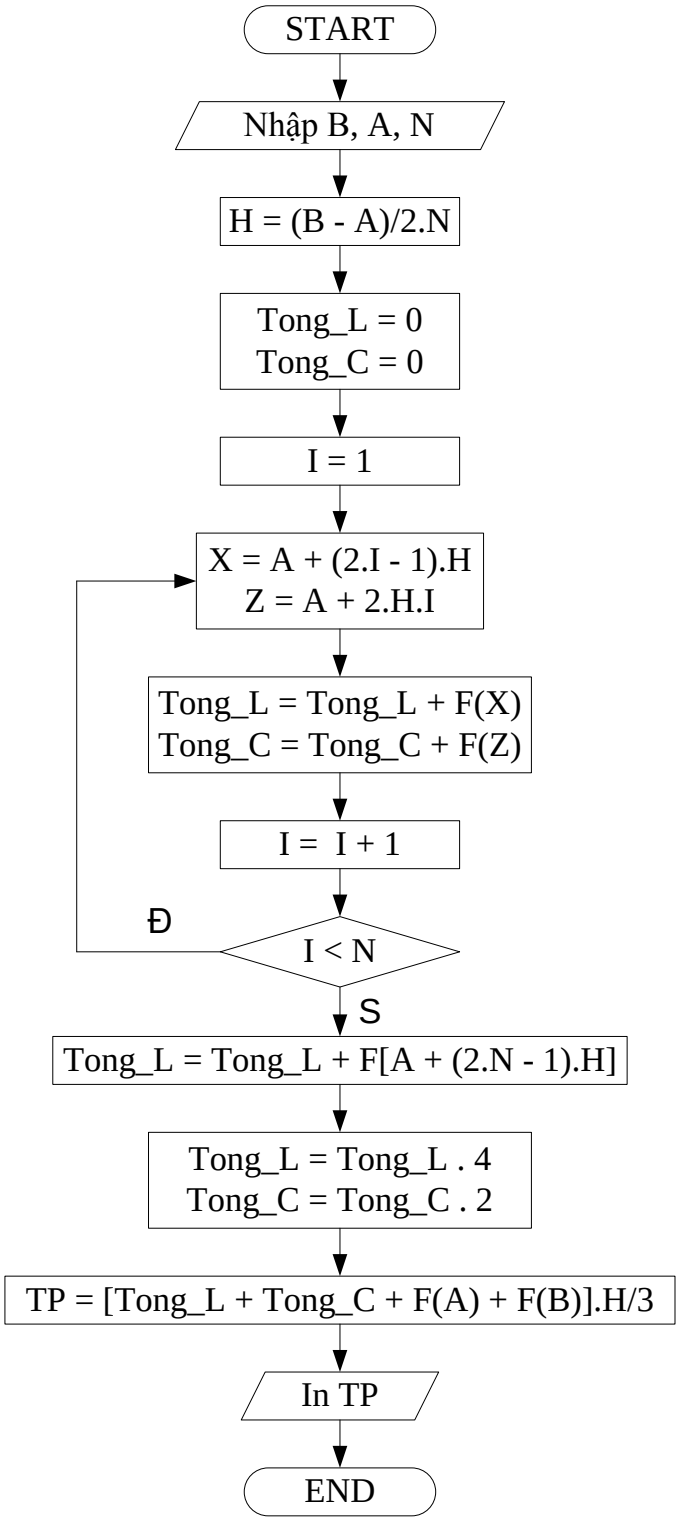
$$\int_a^b f(x)dx \approx \frac{h}{3}\left[y_o+4(y_1+y_3+...+y_{2n-1})+2(y_2+y_4+...+y_{2n-2})+y_{2n}\right]$$

trong đó: $y_o = a$; $y_{2n} = b$; $h = \dfrac{b-a}{2n}$; $x_i = a + ih (i = 0,1,...,2n)$

Người ta cũng cho sai số của xấp xỉ thứ n theo công thức

$$\left| \int_a^b f(x)dx - I_n \right| \le \frac{b-a}{180}.h^4.N$$

trong đó: $N = \max \left| f^{(4)}(x) \right|$; x thuộc $[a, b]$



Lưu đồ của phương pháp Parabol

Kết luận:

Phương pháp parabol cho độ chính xác cao hơn phương pháp hình thang.

Ví dụ 4.2. Tính gần đúng tích phân xác định

$$\int_0^1 \frac{1}{1+x^2}dx$$

Ta chia [0, 1] thành 4 đoạn nhỏ

$a_o = 0$, $a_1 = 0{,}25$, $a_2 = 0{,}5$, $a_3 = 0{,}75$, $a_4 = 1$

$$f(x) = \frac{1}{1+x^2}\begin{bmatrix} f(0)=1 \\ f(0{,}25)=0{,}94118 \\ f(x)=f(0{,}5)=0{,}8 \\ f(0{,}75)=0{,}64 \\ f(1)=1/2 \end{bmatrix}$$

áp dụng công thức parabol ta được

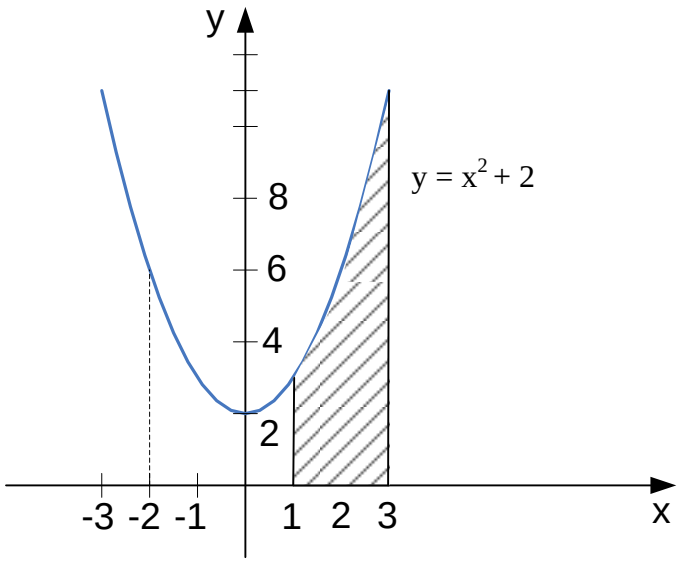
$$\int_0^1 f(x)dx \approx \frac{0{,}25}{3}[1+0{,}5+4(0{,}94118+0{,}64)+2(0{,}8)] \approx 0{,}7854$$

4.4 Thiết kế chương trình

Cho hàm số sau

$$y = f(x) = x^2 + 2$$

có đồ thị là



Hãy tính tích phân $\int_1^3 f(x)$ bằng phương pháp hình thang và parabol.

Yêu cầu kiểm tra thông số nhập.

Nhận xét hai phương pháp hình thang và parabol.

Bài 5

GIẢI HỆ PHƯƠNG TRÌNH TUYẾN TÍNH

Bài thí nghiệm này chủ yếu cung cấp kiến thức lý thuyết cần thiết và ngắn ngọn, cũng như giải thuật của phương pháp giải hệ phương trình tuyến tính nhằm làm cho sinh viên hiểu thấu đáo hơn về ngôn ngữ C và kiến thức vững về phương pháp tính để giải hệ phương trình tuyến tính tổng quát.

5.1 Lý thuyết

Giới thiệu phương trình đại số tuyến tính

Hệ phương trình đại số tuyến tính là hệ có dạng

$$\begin{cases} a_{11}.x_1 + a_{12}.x_2 + \ldots + a_{1n}.x_n = b_1 \\ a_{21}.x_1 + a_{22}.x_2 + \ldots + a_{2n}.x_n = b_2 \\ \vdots \\ a_{n1}.x_1 + a_{n2}.x_2 + \ldots + a_{nn}.x_n = b_n \end{cases} \qquad (5.1)$$

Nếu gọi ma trận A, B, X là

$$A = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{pmatrix}$$

$$X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

$$B = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

thì hệ có thể được viết lại,

$$AX = B \text{ với } \det A = \neq 0 \qquad (*)$$

hay

$$\begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \qquad (5.2)$$

Trong (5.2), nếu lấy từng phần tử của ma trận A $(a_{11}, a_{12}, a_{13}, ..., a_{nn})$ nhân với từng phần tử của ma trận X $(x_1, x_2, ..., x_n)$ ta có được vế trái của biểu thức (5.1). Như vậy hệ phương trình tuyến tính (5.1) thường được viết dưới dạng (5.2) hay (*) hơn là (5.1).

– Định thức (determinant) của ma trận A (matrix) là trị

$$\text{Det A} = \begin{vmatrix} a_{11} & a_{12} & ... & a_{1n} \\ ... & ... & ... & ... \\ ... & ... & ... & ... \\ a_{n1} & a_{n2} & ... & a_{nn} \end{vmatrix}$$

Ví dụ 5.1. Ta có ma trận A

$$\text{A} = \begin{vmatrix} 1 & 2 \\ 2 & 1 \end{vmatrix}$$ (A gọi là ma trận cấp n = 2)

thì: $$\det \text{A} = \begin{vmatrix} 1 & 2 \\ 2 & 1 \end{vmatrix} = 1 \text{ x } 1 - 2 \text{ x } 2 = -3$$

Nếu ma trận A là

$$\text{A} = \begin{vmatrix} 1 & 0 & 1 \\ 2 & 1 & 2 \\ 4 & 2 & 0 \end{vmatrix}$$ thì A là ma trận cấp 3

$$\det \text{A} = \begin{vmatrix} 1 & 0 & 1 \\ 2 & 1 & 2 \\ 4 & 2 & 0 \end{vmatrix} = 1 \text{ x } \begin{vmatrix} 1 & 2 \\ 2 & 0 \end{vmatrix} - 0 \text{ x } \begin{vmatrix} 2 & 2 \\ 4 & 0 \end{vmatrix} + 1 \text{ x } \begin{vmatrix} 2 & 1 \\ 4 & 2 \end{vmatrix}$$

$$= 1 \text{ x } (1 \text{ x } 0 - 2 \text{ x } 2) - 0 \text{ x } (2 \text{ x } 0 - 2 \text{ x } 4) + 1 \text{ x } (2 \text{ x } 2 - 1 \text{ x } 4)$$

$$= -4 + 0 + 0 = -4$$

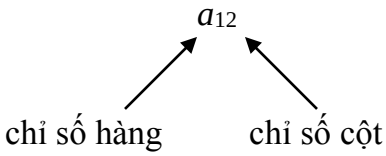
Chú ý: ma trận tổng quát m x n (*m hàng, n cột*) là ma trận có dạng

$$\text{hàng } 2 \rightarrow \begin{vmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & ... & ... & ... \\ ... & ... & ... & ... \\ a_{m1} & a_{m2} & ... & a_{mn} \end{vmatrix}$$

$$\uparrow$$

cột

phần tử a_{12} sẽ ở hàng 1, cột 2



5.2 Chuẩn bị về ngôn ngữ C

Sinh viên cần phải xem lại các vấn đề sau đây của ngôn ngữ C:

- Các hàm xuất nhập printf, scanf
- Các thao tác gán, thay đổi trị cho biến.
- Cách tạo hàm, đối số hàm, trị trả về từ hàm, prototype, …
- Cấu trúc tổng quát một chương trình C, nơi đặt hàm khai báo biến, …
- Cách sử dụng các lệnh điều khiển và vòng lặp.
- Các cấu trúc dữ liệu cần thiết
- Một số giải thuật cơ bản về kiểm tra số liệu nhập có thỏa yêu cầu của bài toán, một số giải thuật xuất dữ liệu theo dạng phù hợp bài toán.
- Trong một số trường hợp nếu cần có thể biểu diễn hàm đồ thị để minh họa, do đó việc hiển thị trong chế độ graph cũng cần được nắm chính xác, một số hàm, biến, hằng, … được dùng trong chế độ graph cần được xem lại, cách sử dụng và khai báo.

5.3 Giải thuật và minh họa

5.1.1 Ma trận đường chéo

Ma trận A có dạng đường chéo

$$\begin{pmatrix} a_{11} & 0 & ... & 0 \\ 0 & a_{22} & ... & 0 \\ ... & ... & ... & ... \\ 0 & 0 & ... & a_{nn} \end{pmatrix}$$

tức các phần tử $a_{ii}(i=\overline{1,n}): a_{ii} \neq 0$

Khi đó nghiệm của hệ là

$$x_i = \frac{b_i}{a_{ii}}(i=\overline{1,n})$$

Ví dụ 5.2. Có hệ phương trình

$$\begin{cases} 3x_1 = 6 \\ 2x_2 = 8 \\ 4x_3 = 24 \end{cases}$$

Ta có thể viết lại hệ dưới dạng

$$\begin{cases} 3x_1 + 0x_2 + 0x_3 = 6 \\ 0x_1 + 2x_2 + 0x_3 = 8 \\ 0x_1 + 0x_2 + 4x_3 = 24 \end{cases}$$

hay dưới dạng ma trận

$$\underset{A}{\begin{pmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{pmatrix}} \underset{X}{\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}} = \underset{B}{\begin{pmatrix} 6 \\ 8 \\ 24 \end{pmatrix}}$$

Như vậy ma trận A có dạng đường chéo, do đó nghiệm của hệ là

$$\begin{cases} x_1 = \dfrac{b_1}{a_{11}} = \dfrac{6}{3} = 2 \\ x_2 = \dfrac{b_2}{a_{22}} = \dfrac{8}{2} = 4 \\ x_3 = \dfrac{b_3}{a_{33}} = \dfrac{24}{4} = 6 \end{cases}$$

5.1.2 Ma trận tam giác trên

Ma trận A có dạng tam giác trên

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ 0 & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & a_{nn} \end{pmatrix}$$

tức các phần tử

$a_{ij} = 0$ với i > j

$a_{ii} \neq 0$ (hay có thể viết $a_{ij} \neq 0$ với i = j)

a_{ij} bất kỳ với i < j

Nếu viết theo dạng (5.1), ta có:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1 \\ 0x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2 \\ \dots \\ 0x_1 + 0x_2 + \dots + a_{nn}x_n = b_n \end{cases}$$

hay viết gọn lại ta có,

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1 (h.1) \\ a_{22}x_2 + \dots + a_{2n}x_n = b_2 (h.2) \\ \dots \\ \dots a_{nn}x_n = b_n (h.n) \end{cases}$$

Từ hàng n (h.n) ta có,

$$x_n = \frac{b_n}{a_{nn}} \qquad (5.3)$$

Hàng h.(n-1) có dạng

$$a_{(n-1),(n-1)}x_{n-1} + a_{(n-1),n}x_n = b_{n-1}$$

$$\Rightarrow x_{n-1} = \frac{1}{a_{(n-1),(n-1)}}(b_{n-1} - a_{n-1,n}x_n)$$

Hàng h.(n-2) có dạng

$$a_{(n-2),(n-2)}x_{n-2} + a_{(n-2),(n-1)}x_{n-1} + a_{n-2,n}x_n = b_{n-2}$$

$$\Rightarrow x_{n-2} = \frac{1}{a_{(n-2),(n-2)}}\left[b_{\text{n-2}} \ (a_{(\text{n-2}),(\text{n-1})}x_{\text{n-1}} + a_{\text{n-2,n}}x_{\text{n}})\right]$$

Cứ như vậy tới hàng h.1 ta có

$$\Rightarrow x_1 = \frac{1}{a_{11}}\left[b_1 - (a_{12}x_2 + a_{13}x_3 + \ldots + a_{1\text{n}}x_{\text{n}})\right]$$

Tổng quát, xét hàng j ta có nghiệm,

$$x_j = \frac{1}{a_{jj}}\left[b_{\text{j}} - (a_{\text{j,j+1}}x_{j+1} + a_{\text{j,j+2}}x_{\text{j+2}} + \ldots + a_{\text{jn}}x_{\text{n}})\right]$$

Nếu ký hiệu

$$a_{\text{j,j+1}}x_{\text{j+1}} + a_{\text{j,j+2}}x_{\text{j+2}} + \ldots + a_{\text{jn}}x_{\text{n}} = \sum_{i=j+1}^{n} a_{ji}x_i$$

thì $$x_j = \frac{1}{a_{jj}}\left[b_j - \sum_{i=j+1}^{n} a_{ji}x_i\right] \qquad (5.4)$$

(5.4) là công thức tổng quát tính ra nghiệm của hệ.

Ý nghĩa của công thức (5.4)

Công thức (5.4) trình bày cách tính phần tử nghiệm thứ j từ các nghiệm j+1, j+2,…, n

Ví dụ 5.3. Tính nghiệm của hệ

$$\begin{cases} 2x_1 - 3x_2 + x_3 = 7 & (1) \\ 5x_2 + 3x_3 = 1 & (2) \\ -2x_3 = -4 & (3) \end{cases}$$

từ (3), ta có nghiệm

$$x_3 = \frac{-4}{-2} = 2$$

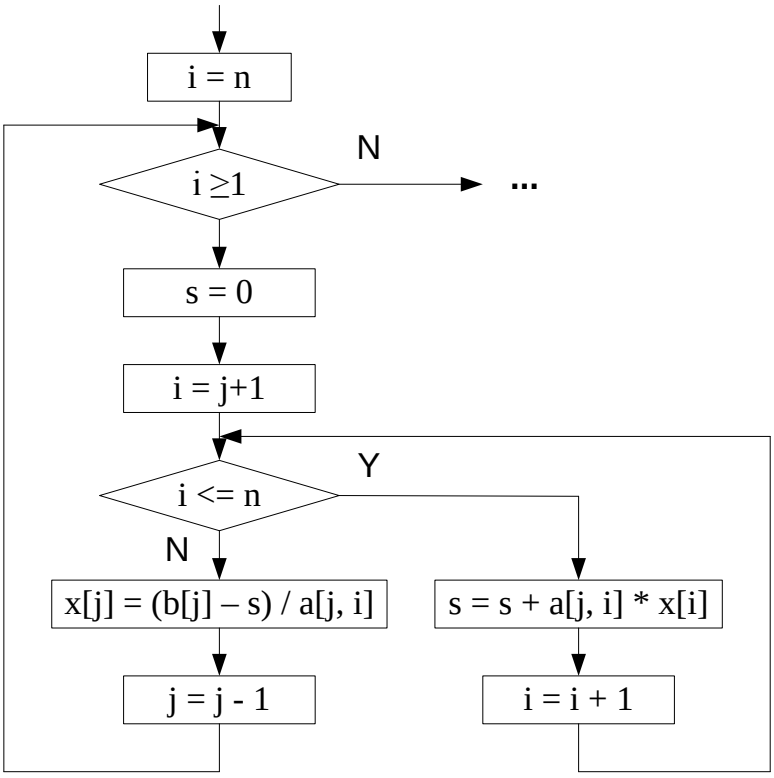
từ (2), ta có nghiệm

$$x_2 = \frac{1}{5}(1 - 3x_3)$$
$$= \frac{1}{5}(1 - 3.2) = -1$$

từ (1), ta có

$$x_1 = \frac{1}{2}\left[7 - (-3x_2 + x_3)\right]$$
$$= \frac{1}{2}\left[7 - (-3.(-1) + 2)\right]$$
$$= \frac{1}{2}\left[7 - (5)\right] = 1$$

Giải thuật theo họa đồ cách tính ra n nghiệm theo công thức (5.4)



Đoạn chương trình C minh họa cho giải thuật trên

```
for(j = n; j >= 1; j-- )
{
      s = 0;
      i = j + 1;
      while(i <= n)
      {
            s += a[j][i] * x[i];
            i++;
      }
      x[j] = (b[j] – s) / a[j][j];
}
```

Chú ý: đoạn chương trình trên chỉ là một đoạn cắt trong chương trình giải hệ phương trình tuyến tính có dạng tam giác trên với các biên.

n: là chiều hay số hàng, cột của ma trận A, B, X

s: là biến trung gian đã được khai báo

i, j: là hai biến đếm dùng trong việc đếm lặp

a, b, x: là các ma trận, các mảng hai chiều và một chiều đã được khai báo, biểu diễn các ma trận tương ứng sau

a <–> A

b <–> B

x <–> X

5.1.3 Ma trận tam giác dưới

Ma trận A có dạng tam giác dưới khi A là

$$A = \begin{pmatrix} a_{11} & 0 & ... & 0 \\ a_{21} & a_{22} & ... & a_{2n} \\ ... & ... & ... & ... \\ a_{n1} & ... & ... & a_{nn} \end{pmatrix}$$

Tức các phần tử,

$a_{ij} = 0$ với j > i

$a_{ii} \neq 0$ (hay có thể viết $a_{ij} \neq 0$ với i = j)

a_{ij} bất kỳ với i > j

Nếu viết theo dạng (5.1), ta có:

$$\begin{cases} a_{11}x_1 + 0x_2 + \ldots + 0x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + 0x_n = b_2 \\ \ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n \end{cases}$$

hay viết gọn lại ta có,

$$\begin{cases} a_{11}x_1 = b_1 (h.1) \\ a_{21}x_1 + a_{22}x_2 = b_2 (h.2) \\ \ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n (h.n) \end{cases}$$

Từ hàng (h.1) ta có,

$$x_1 = \frac{b_1}{a_{11}}$$

Hàng 2 có dạng

$$a_{21}x_1 + a_{22}x_2 = b_2$$

$$\Rightarrow x_2 = \frac{1}{a_{22}}(b_2 - a_{21}x_1)$$

Hàng 3 có dạng

$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3$$

$$\Rightarrow x_3 = \frac{1}{a_{33}}\left[b_3 - (a_{n1}x_1 + a_{n2}x_2)\right]$$

Cứ như vậy tới hàng n, ta có

$$\Rightarrow x_n = \frac{1}{a_{nn}}\left[b_n - (a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{n,n-1}x_{n-1})\right]$$

Tổng quát xét hàng j ta có nghiệm,

$$\Rightarrow x_j = \frac{1}{a_{jj}}\left[b_j - (a_{j1}x_1 + a_{j2}x_2 + \ldots + a_{j,j-1}x_{j-1})\right]$$

Nếu ký hiệu

$$a_{j1}x_1 + a_{j2}x_2 + \ldots + a_{j,j-1}x_{j-1} = \sum_{i=1}^{j-1} a_{ji}x_i$$

thì $$x_j = \frac{1}{a_{jj}}\left[b_j - \sum_{i=1}^{j-1} a_{ji}x_i\right] \qquad (5.5)$$

(5.5) là công thức tổng quát tính ra nghiệm của hệ

Ý nghĩa của công thức (5.5)

Công thức (5.5) trình bày cách tính phần tử nghiệm thứ j từ các nghiệm đã có trước đó $x_1, x_2, \ldots, x_{j-1}$

Ví dụ 5.4. Tính nghiệm của hệ.

$$\begin{cases} 2x_1 = -2 & (1) \\ -x_1 + x_2 = 3 & (2) \\ x_1 + x_2 - x_3 = 0 & (3) \end{cases}$$

Từ (1), ta có nghiệm

$$x_1 = \frac{-2}{2} = -1$$

Từ (2), ta có

$$x_2 = \frac{1}{1}[3 + x_1]$$

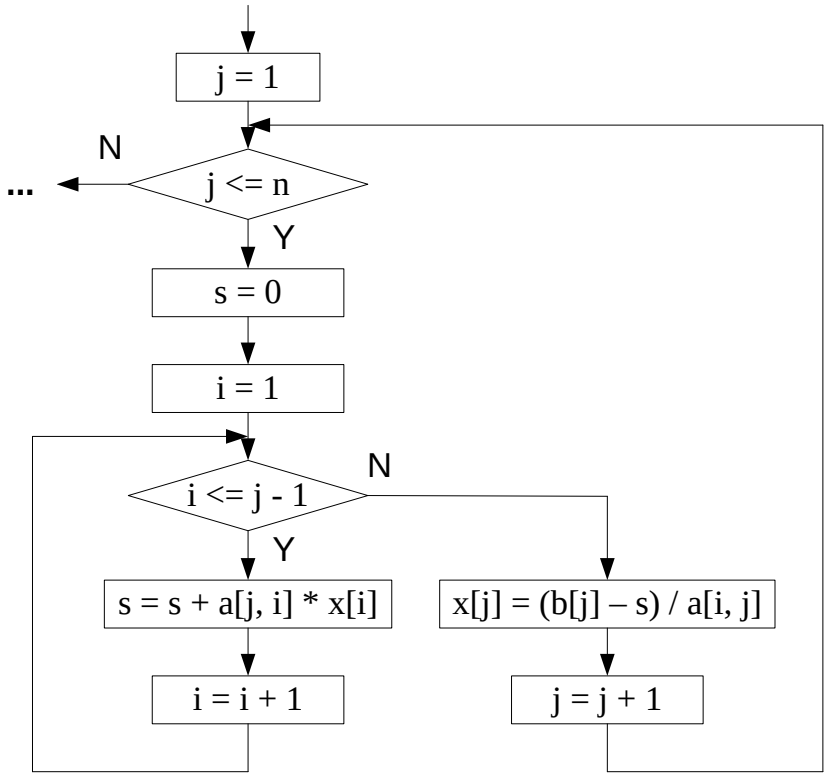
$$= (3 + x_1) = (3 - 1) = 2$$

Từ (3), ta có

$$x_3 = x_1 + x_2$$

$$= -1 + 2 = 1$$

Giải thuật theo lưu đồ cách tính ra n nghiệm theo công thức (5.5)



Đoạn chương trình C minh họa cho giải thuật trên

```
for(j=1; j <= n; j++)
{
        s = 0;
        i = 1;
        while(i <= j–1)
        {
                s += a[j][i] * x[i];
                i++;
        }
        x[j] = (b[j] – s) / a[j][j];
}
```

Chú ý:

Đoạn chương trình trên chỉ là một đoạn cắt trong chương trình giải hệ phương trình tuyến tính có dạng tam giác dưới các biến

n: là chiều hay số hàng, cột của ma trận A, B, X

s: là biến trung gian đã được khai báo

i, j: là hai biến đếm dùng để kiểm soát vòng lặp

a, b, x: là các ma trận, các mảng hai chiều và một chiều đã được khai báo biểu diễn các ma trận tương ứng như sau

a <–> A

b <–> B

x <–> X

5.1.4 Ma trận bất kỳ

Cho hệ AX = B với A là ma trận cấp n x n bất kỳ

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n \end{cases}$$

Giả thiết với mọi $a_{ii} \neq 0$, với mọi i = 1, 2, …, n

Đối với hệ trên ta sẽ dùng phương pháp khử Gauss để giải hệ này.

Nội dung phương pháp khử Gauss là biến hệ ban đầu về hệ có ma trận A có dạng tam giác trên tương đương, nhờ ba phép biến đổi tương đương như sau.

- Nhân một phương trình nào đó với một hằng số khác zero
- Cộng một phương trình vào một phương trình khác sau khi đã nhân nó với một số khác zero
- Đổi chỗ hai phương trình

Các bước của phương pháp khử Gauss

<u>Bước 1</u>:

Giả sử $a_{11} \neq 0$, ta giữ lại dòng 1 gọi là dòng quay (*hay dòng trụ*), và khử x_1 từ dòng 2 đến dòng n. Muốn vậy ta chỉ cần lần lượt nhân dòng trụ 1 với hệ số $m_{i1} = \frac{a_{i1}}{a_{11}}$, rồi lấy dòng i trừ với dòng 1 (*sau khi đã nhân nó với m_{i1}*)

Ta có công thức chung cho các dòng từ 2 đến n như sau,

*dòng$^{(1)}$ i = dòng i – m_{i1} * dòng 1*

với *i = 2, 3, ...n*

(1) chỉ số bước 1

Ta có hệ mới thu được sau bước 1 là,

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ 0 + a_{22}^{(1)}x_2 + \ldots + a_{2n}^{(1)}x_n = b_2^{(1)} \\ \ldots \\ 0 + a_{n2}^{(1)}x_2 + \ldots + a_{nn}^{(1)}x_n = b_n^{(1)} \end{cases}$$

trong đó: $a_{ij}^{(1)} = a_{ij} - m_{i1} * a_{1j}$

$$b_i^{(1)} = b_i - m_{i1} * b_1 (i, j = 2,3,..., n)$$

$$m_{i1} = \frac{a_{i1}}{a_{11}}$$

<u>Bước 2:</u>

Giả sử $a_{22}^{(1)} \neq 0$, ta giữ lại dòng 2 làm dòng quay thứ 2 và khử x_2 từ dòng 3 đến dòng n. Tương tự như bước 1, ta có công thức thực hiện đó là,

$dòng^{(2)}\ i\ =\ dòng\ ^{(1)}i - m_{i2}*dòng^{(1)}2$

với $i = 3, 4, \ldots, n$

và hệ mới sẽ là

$$\begin{cases} a_{11}x_{11} + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ 0 + a_{22}^{(1)}x_2 + \ldots + a_{2n}^{(1)}x_n = b_2^{(1)} \\ 0 + 0 + a_{33}^{(2)}x_3 + \ldots + a_{3n}^{(2)}x_n = b_3^{(2)} \\ \ldots \\ 0 + 0 + a_{n3}^{(2)}x_3 + \ldots + a_{nn}^{(2)}x_n = b_n^{(2)} \end{cases}$$

Trong đó: $a_{ij}^{(2)} = a_{ij}^{(1)} - m_{i2} * a_{2j}^{(1)}$

$$b_i^{(2)} = b_i^{(1)} - m_{i2} * b_2^{(1)} (i, j = 3,4,..., n)$$

$$m_{i2} = \frac{a_{i2}^{(1)}}{a_{22}^{(1)}}$$

Tổng quát, công thức thực hiện ở bước K là

$dòng^{(K)}\ i\ =\ dòng\ ^{(K-1)}i - m_{iK}*dòng^{(K-1)}K$

với $i = K+1, K+2, \ldots, n$

trong đó: $a_{ij}^{(K)} = a_{ij}^{(K-1)} - m_{iK} * a_{Kj}^{(K-1)}$

$$b_i^{(K)} = b_i^{(K-1)} - m_{iK} * b_K^{(K-1)} (i, j = K+1, K+2,..., n)$$

$$m_{iK} = \frac{a_{iK}^{(K-1)}}{a_{KK}^{(K-1)}}$$

Qua (n-1) bước (K = 1, 2, …, n-1) ta có hệ cuối cùng có dạng tam giác là

$$\begin{cases} a_{11}x_{11} + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ 0 + a_{22}^{(1)}x_2 + \ldots + a_{2n}^{(1)}x_n = b_2^{(1)} \\ 0 + 0 + a_{33}^{(2)}x_3 + \ldots + a_{3n}^{(2)}x_n = b_3^{(2)} \\ \ldots \\ 0 + 0 + 0 + \ldots + a_{nn}^{(n-1)}x_n = b_n^{(n-1)} \end{cases}$$

Các phần tử $a_{11}, a_{22}^{(1)}, \ldots, a_{(n-1)(n-1)}^{(n-2)}$ được gọi là phần tử quay hay phần tử trụ

Ví dụ 5.5. Giải hệ phương trình

$$\begin{cases} x_1 + x_2 + x_3 = 6 & (1) \\ 2x_1 - x_3 = -1 & (2) \\ x_1 - x_2 - x_3 = -4 & (3) \end{cases}$$

Ma trận A và B viết gọn lại như sau

$$\left(\begin{array}{ccc|c} 1 & 1 & 1 & 6 \\ 2 & 0 & -1 & -1 \\ 1 & -1 & -1 & -4 \end{array}\right) \begin{matrix} \\ (1) \\ (2) \\ (3) \end{matrix}$$

A B

Bước 1:

Ta nhận thấy $a_{11} = 1$ (a_{11} đã khác 0) nên

hàng $^{(1)}$(2) = hàng (2) – $\frac{a_{21}}{a_{11}}$ hàng (1)

=hàng (2) – $\frac{2}{1}$ hàng (1)

Như vậy: hàng (2) : 2 0 –1 –1

$\frac{2}{1}$ hàng (1): 2 2 2 12

$\Rightarrow$ hàng $^{(1)}$(2): 0 –2 –3 –13

hàng $^{(1)}$(3) = hàng (3) – $\frac{a_{31}}{a_{11}}$ hàng (1)

=hàng (3) – $\frac{1}{1}$ hàng (1)

Như vậy: hàng (3): 1 –1 –1 –4

$\frac{1}{1}$ hàng (1): 1 1 1 6

$\Rightarrow$ hàng $^{(1)}$(3): 0 –2 –2 –10

Sau khi thực hiện bước 1, ta có ma trận A, B như sau

$$\left(\begin{array}{ccc|c} 1 & 1 & 1 & 6 \\ 0 & -2 & -3 & -13 \\ 0 & -2 & -2 & -10 \end{array}\right)$$

Bước 2:

Ta nhận thấy $a_{22} = -2$ ($a_{22} \neq 0$) nên

hàng $^{(2)}$(3) = hàng $^{(1)}$(3) – $\frac{a_{32}}{a_{22}}$ hàng $^{(1)}$(2)

= hàng $^{(1)}$(3) – $\frac{-2}{-2}$ hàng $^{(1)}$(2)

= hàng $^{(1)}$(3) – hàng $^{(1)}$(2)

Như vậy: hàng $^{(1)}$(3): 0 –2 –2 –10

hàng $^{(1)}$(2): 0 –2 –2 –13

hàng $^{(2)}$(3): 0 0 1 3

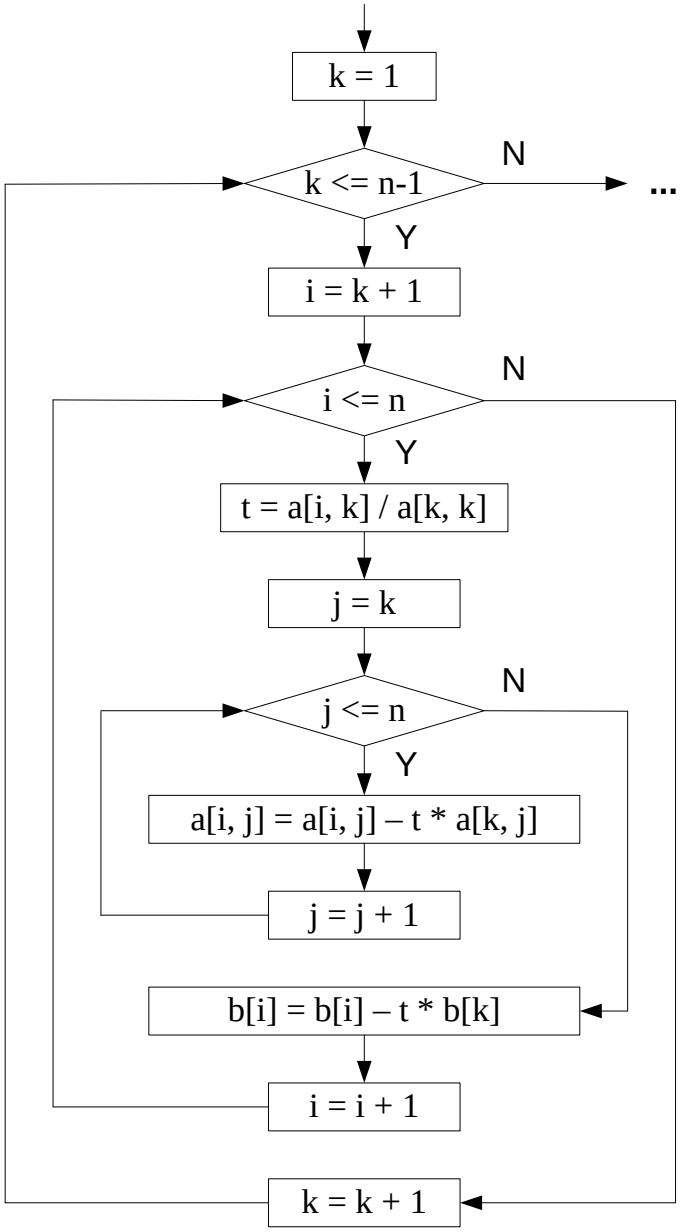
Sau khi thực hiện bước 2, ta có ma trận A, B như sau

$$\left(\begin{array}{ccc|c} 1 & 1 & 1 & 6 \\ 0 & -2 & -3 & -13 \\ 0 & 0 & 1 & -3 \end{array}\right)$$

Ma trận A có dạng tam giác trên, áp dụng cách giải đã nêu ở mục 2, ta có nghiệm

$$\begin{cases} x_1 = 1 \\ x_2 = 2 \\ x_3 = 3 \end{cases}$$

Như vậy vấn đề đặt ra là từ ma trận A, B có sẵn, ta phải có giải thuật theo phép biến đổi Guass đưa nó về ma trận tam giác trên.



Lưu đồ của giải thuật

Đoạn chương trình C minh họa giải thuật trên

```
for(k = 1; k <= n –1; k++)
      for(i = k+1; i <= n; i++)
      {
```

```
            t = a[i][k] / a[k][k];
            for(j = k; j <= n; j++)
                  a[i][j] -= t * a[k][j];
            b[i] -= t * b[k];
      }
```

Chú ý:

Đoạn chương trình trên chỉ là một đoạn cắt trong chương trình giải hệ phương trình tổng quát với các biến theo quy định.

n: là chiều hay số hàng, cột của ma trận A, B, X

k, i, j: là ba biến kiểm soát vòng lặp

a, b: là hai mảng hai và một chiều đã khai báo, biểu diễn các ma trận A và B của hệ phương trình.

Khi thiết kế chương trình giải hệ phương trình tuyến tính tổng quát thì chương trình giải phải có ba phần (thủ tục hoặc hàm) cơ bản:

1- Hàm kiểm tra các phần tử a_{ii}, các a_{ii} phải thỏa $a_{ii} \neq 0$, nếu có hàng không thỏa điều kiện này thì phải hoán đổi hàng, cộng hàng … theo nguyên tắc của ma trận để đạt được $a_{ii} \neq 0$.

2- Hàm biến đổi ma trận A (mảng a) thành ma trận tam giác trên.

3- Hàm giải ra nghiệm của hệ phương trình tuyến tính với A (mảng a) là ma trận của tam giác trên.

5.4 Thiết kế chương trình

Hãy thiết kế chương trình giải hệ phương trình tuyến tính tổng quát, có kiểm tra các hệ số nhập vào cho các ma trận A và B, giải thuật đã nêu ở phần 5.3.