HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BÀI GIẢNG

PHƯƠNG PHÁP SỐ

Biên soạn : Ths. PHAN THỊ HÀ
Ts. PHAN ĐĂNG CẦU

Lưu hành nội bộ

HÀ NỘI - 2006

GIỚI THIỆU MÔN HỌC

I. GIỚI THIỆU CHUNG

Phương pháp số là một lĩnh vực của toán học chuyên nghiên cứu các phương pháp giải các bài toán (chủ yếu là gần đúng) bằng cách dựa trên những ***dữ liệu số*** cụ thể và cho ***kết quả cũng dưới dạng số***. Nói gọn hơn, phương pháp số như bản thân tên gọi của nó, có nghĩa là phương pháp *giải các bài toán bằng những con số cụ thể.*

Ngày nay phần lớn các công việc tính toán đều được thực hiện trên máy tính. Tuy vậy thực tế chứng tỏ rằng, việc áp dụng các thuật toán và phương pháp tính toán khác nhau có thể cho tốc độ tính toán và độ chính xác rất khác nhau. Lấy ví dụ đơn giản như tính định thức của ma trận chẳng hạn, nếu tính trực tiếp theo định nghĩa thì việc tính định thức của một ma trận vuông cấp 25 cũng mất hàng triệu năm (ngay cả với máy tính hiện đại nhất hiện nay); trong khi đó nếu sử dụng phương pháp khử Gauss thì kết quả nhận được gần như tức thời.

Như vậy, phương pháp số là công cụ không thể thiếu trong các công việc cần thực hiện nhiều tính toán với tốc độ tính toán nhanh và độ chính xác cao như vật lý, điện tử viễn thông, ... và dĩ nhiên là tất cả các ngành và mọt lĩnh vực đều cần đến là công nghệ thông tin.

Phương pháp số được nghiên cứu từ rất lâu và cho đến nay những thành tựu đạt được là một khối lượng kiến thức đồ sộ được in trong nhiều tài liệu sách, báo... Tuy nhiên, môn học "Phương pháp số" chỉ nhằm cung cấp những kiến thức căn bản nhất về phương pháp số. Với lượng kiến thức này sinh viên có thể áp dụng vào giải quyết những bài toán thông thường trong thực tế và có khả năng tự tìm hiểu để nâng cao kiến thức cho mình khi gặp các vấn đề phức tạp hơn.

II. MỤC ĐÍCH

Môn học phương pháp số cung cấp cho sinh viên kiến thức căn bản nhất về một số phương pháp giải gần đúng trên dữ liệu số .

Tạo cơ sở để học tốt và nghiên cứu các nghành khoa học kỹ thuật nói chung và Công nghệ thông tin nói riêng.

Góp phần rèn luyện phương pháp suy luận khoa học, tư duy logic, phương pháp nghiên cứu thực nghiệm

Góp phần xây dựng thế giới quan khoa học và tác phong khoa học cần thiết cho người kỹ sư tương lai.

III. PHẠM VI NGHIÊN CỨU

Nghiên cứu một số phương pháp cơ bản nhất của phương pháp số, được ứng dụng nhiều trong thực tế như các phương pháp số trong đại số tuyến tính, bài toán nội suy, tìm nghiệm gần đúng các phương trình phi tuyến, tính gần đúng đạo hàm và tích phân, giải gần đúng một số dạng của phương trình vi phân...

Tìm hiểu các lĩnh vực ứng dụng của các phương pháp trong thực tế.

Nghiên cứu cách cài đặt các thuật toán trên máy tính.

IV. PHƯƠNG PHÁP NGHIÊN CỨU:

Để học tốt môn học này, sinh viên cần lưu ý những vấn đề sau:

1. Kiến thức cần trước:

- Sinh viên phải có kiến thức cơ bản về toán học cao cấp.

- Thành thạo ít nhất một ngôn ngữ lập trình. Đặc biệt trong cuốn sách này đã sử dụng ngôn ngữ lập trình C để mô tả thuật toán, vì vậy sinh viên phải nắm được ngôn ngữ lập trình C.

2. Thu thập đầy đủ các tài liệu:

Giáo trình Phương pháp số. Phan Đăng Cầu, Phan Thị Hà, Học viện Công nghệ BCVT, 2002.

Nếu cần sinh viên nên tham khảo thêm:

- Giải tích số. Phạm Kỳ Anh, nhà xuất bản đại học Quốc Gia Hà Nội, 1966.
- Phương pháp tính. Tạ Văn Đỉnh, Nhà xuất bản Giáo dục - 1995.
- Phương Pháp tính. Dương Thuỳ Vỹ, Nhà xuất bản Khoa học và Kỹ thuật, 2001.

3. Đặt ra mục tiêu, thời hạn cho bản thân:

Đặt ra các mục tiêu tạm thời và thời hạn cho bản thân và cố gắng thực hiện chúng

Xây dựng mục tiêu trong chương trình nghiên cứu.

4 Nghiên cứu và nắm những kiến thức cốt lõi:

Sinh viên nên đọc qua sách hướng dẫn học tập trước khi nghiên cứu bài giảng môn học và các tài liệu tham khảo khác.

5. Tham gia đầy đủ các buổi hướng dẫn học tập:

Thông qua các buổi hướng dẫn học tập, giảng viên sẽ giúp sinh viên nắm được nội dung tổng thể của môn học và giải đáp thắc mắc, đồng thời sinh viên cũng có thể trao đổi, thảo luận với những sinh viên khác về nội dung bài học.

6. Chủ động liên hệ với bạn học và giảng viên:

Cách đơn giản nhất là tham dự các diễn dàn học tập trên mạng Internet, qua đó có thể trao đổi trực tiếp các vấn đề vướng mắc với giảng viên hoặc các bạn học khác đang online.

7. Tự ghi chép lại những ý chính:

Việc ghi chép lại những ý chính là một hoạt động tái hiện kiến thức, kinh nghiệm cho thấy nó giúp ích rất nhiều cho việc hình thành thói quen tự học và tư duy nghiên cứu.

8. Học đi đôi với hành

Học lý thuyết đến đâu thực hành làm bài tập ngay đến đó để hiểu và nắm chắc lý thuyết. Nói chung cuối mỗi chương, sinh viên cần tự trả lời các câu hỏi, bài tập. Hãy cố gắng vạch ra những ý trả lời chính, từng bước phát triển thành câu trả lời hoàn thiện.

Liên hệ với các môn học khác và các vấn đề thực tế có liên quan để hiểu sâu hơn ý nghĩa của các phương pháp.

Cài đặt các thuật toán bằng nhiều cách khác nhau, có sử dụng đồ họa để làm nổi bật các đặc trưng và kết quả của các thuật toán. Dùng đồ thị so sánh các phương pháp khác nhau cùng giải quyết một bài toán, phân tích những điểm yếu điểm mạnh của các thuật toán. Khi cài đặt thuật toán nếu có gì vướng mắc thì sinh viên có thể tham khảo thêm phần code của toàn bộ chương trình tương ứng đã được viết bằng ngôn ngữ lập trình C trong tài liệu: "Phương pháp số. Phan Đăng Cầu, Phan Thị Hà, Học viện Công nghệ BCVT, 2002".

CHƯƠNG 1
SỐ XẤP XỈ VÀ SAI SỐ

MỤC ĐÍCH, YÊU CẦU

Sau khi nghiên cứu chương 1, yêu cầu sinh viên:

1. Hiểu được Phương Pháp Số là gì, vai trò và tầm quan trọng của Phương pháp số.
2. Hiểu được sai số tuyệt đối và sai số tương đối.
3. Nắm được cách viết số xấp xỉ.
4. Nắm được các qui tắc tính sai số.
5. Hiểu và biết cách đánh giá sai số tính toán và sai số phương pháp .

1.1. TỔNG QUAN VỀ PHƯƠNG PHÁP SỐ

1.1.1. Phương pháp số là gì?

Phương pháp số (numerical method) hay đôi khi còn được gọi là Phương pháp tính (Computational method), Toán học tính toán (Computational mathematics) hoặc Giải tích số (Numerical analysis) là một lĩnh vực của toán học chuyên nghiên cứu các phương pháp giải gần đúng các bài toán bằng cách dựa trên những ***dữ liệu số*** cụ thể và cho ***kết quả cũng dưới dạng số***. Nói gọn hơn, phương pháp số như bản thân tên gọi của nó, có nghĩa là phương pháp giải các bài toán bằng những con số cụ thể.

Trong phương pháp số chúng ta thường quan tâm đến hai vấn đề:

- Phương pháp để giải bài toán.
- Mối liên hệ giữa lời giải số gần đúng và lời giải đúng, hay vấn đề ***sai số*** của lời giải.

1.1.2. Những dạng sai số thường gặp

Khi thực hiện một bài toán bằng phương pháp số ta thường gặp những loại sai số sau đây:

- *Sai số trong việc mô hình hóa* bài toán
- *Sai số phương pháp*
- *Sai số của số liệu*
- *Sai số tính toán*

Những sai số trên đây tổng hợp lại nhiều khi dẫn đến những lời giải quá cách xa so với lời giải đúng và vì vậy không thể dùng được. Chính vì vậy việc tìm ra những thuật toán hữu hiệu để giải các bài toán thực tế là điều rất cần thiết.

1.2. SAI SỐ TUYỆT ĐỐI VÀ SAI SỐ TƯƠNG ĐỐI

1.2.1. Sai số tuyệt đối

Trong tính gần đúng ta làm việc với các giá trị gần đúng của các đại lượng. Cho nên vấn đề đầu tiên cần nghiên cứu là vần đề sai số.Xét đại lượng đúng A và đại lượng gần đúng của nó là a. Ta nói a xấp xỉ A và viết $a \approx A$.

Trị tuyệt đối $\Delta_a = |a-A|$ (1.1)

được gọi là sai số tuyệt đối của a (khi dùng a để xấp xỉ A).

Trong thực tế ta không biết được số đúng A, do đó nói chung sai số tuyệt đối không tính được. Vì vậy ta tìm cách ước lượng sai số tuyệt đối của a bằng số $E_a>0$ sao cho

$$|a - A| \le E_a \quad (1.2)$$

Số dương E_a được gọi là *sai số tuyệt đối giới hạn* của a. Rõ ràng nếu E_a là sai số tuyệt đối giới hạn của a thì mọi $E > E_a$ đều là sai số tuyệt đối giới hạn của a. Nếu sai số tuyệt đối giới hạn quá lớn so với sai số tuyệt đối thì nó không còn có ý nghĩa về phương diện sai số nữa. Trong những điều kiện cụ thể người ta cố gắng chọn E_a là số dương bé nhất có thể được thoã mãn (1.1). Nếu E_a là sai số tuyệt đối giới hạn của a khi xấp xỉ A thì ta quy ước viết:

$$A = a \pm E_a \quad (1.3)$$

với ý nghĩa của (1.1), tức là

$$a - E_a \le A \le a + E_a \quad (1.4)$$

1.2.2. Sai số tương đối

Gọi Δ_a là sai số tuyệt đối của a khi dùng a để xấp xỉ A, khi đó đại lượng

$$\delta_a = \frac{\Delta_a}{|a|} \quad (1.5)$$

được gọi là ***sai số tương đối*** của a. Tuy nhiên một lần nữa ta thấy rằng A thường không biết, vì vậy người ta định nghĩa đại lượng

$$\varepsilon_a = \frac{E_a}{|a|} \quad (1.6)$$

là sai số tương đối giới hạn của a. Từ đây ta có

$$E_a = |a| \varepsilon_a \quad (1.7)$$

Từ đây người ta thường viết

$$A = a(1 \pm \varepsilon_a) \quad (1.8)$$

Vì trong thực tế chúng ta chỉ có thể thao tác với các sai số giới hạn, do đó người ta thường gọi một cách đơn giản E_a là sai số tuyệt đối, ε_a là sai số tương đối. Đôi khi người ta biểu diễn sai số tương đối dưới dạng %. Ví dụ với a =10, E_a = 0.05, khi đó ta có ε_a = 0.05/10 = 0.5 %.

1.2.3. Chú thích:

Sai số tuyệt đối không nói lên đầy đủ "chất lượng" của một số xấp xỉ, "chất lượng" ấy còn được phản ánh qua sai số tương đối.

1.3. CÁCH VIẾT SỐ XẤP XỈ

1.3.1. Chữ số có nghĩa

Một số viết dưới dạng thập phân có thể gồm nhiều chữ số, nhưng ta chỉ kể các chữ số từ chữ số khác không đầu tiên tính từ trái đến chữ số cuối cùng khác không phía bên phải là các ***chữ số có nghĩa***. Chẳng hạn số 2.740 có 3 chữ số có nghĩa, số 0.02078 có 4 chữ số có nghĩa.

1.3.2. Chữ số đáng tin

Mọi số thập phân đều có dạng

$$a = \pm \overline{\alpha_n \alpha_{n-1} ... \alpha_1 \alpha_0 . \alpha_{-1} \alpha_{-2} ... \alpha_{-m}} = \pm \Sigma \ \alpha_s 10^s$$

Trong đó α_s là những số nguyên từ 0 đến 9. Giả sử a là xấp xỉ của số A với sai số tuyệt đối là Δ_a. Nếu $\Delta_a \le 0.5*10^s$ thì ta nói rằng chữ số α_s là *đáng tin* (và như vậy các chữ số có nghĩa bên trái α_s đều là đáng tin). Nếu $\Delta_a > 0.5*10^s$ thì ta nói rằng chữ số α_s là *đáng nghi* (và như vậy các chữ số bên phải α_s đều là đáng nghi).

Ví dụ. Số xấp xỉ a = 4.67329

với $\Delta_a = 0.004726$. Ta có $|\Delta_a| \le 0.5 *10^{-2}$ do đó các chữ số đáng tin là: 4,6,7; các chữ số đáng ngờ là 3,2, 9.

với $\Delta_a = 0.005726$. Ta có $|\Delta_a| \le 0.5 *10^{-1}$ (nhưng $|\Delta_a| > 0.5 *10^{-2}$) do đó các chữ số đáng tin là: 4,6; các chữ số đáng ngờ là 7, 3, 2, 9.

1.3.3. Cách viết số xấp xỉ

a. Kèm theo sai số

Cách thứ nhất là viết kèm theo sai số như công thức (1.3) $A = a \pm E_a$

b. Mọi chữ số có nghĩa đều đáng tin

Cách thứ hai là viết theo quy ước: mọi chữ số có nghĩa đều đáng tin; có nghĩa là sai số tuyệt đối giới hạn không lớn hơn một nửa đơn vị ở hàng cuối cùng.

1.3.4. Sai số quy tròn

Trong tính toán với các con số ta thường làm tròn các số theo quy ước sau: nếu chữ số bỏ đi đầu tiên ≥ 5 thì thêm vào chữ số giữ lại cuối cùng một đơn vị, còn nếu chữ số bỏ đi đầu tiên < 5 thì để nguyên chữ số giữ lại cuối cùng.

Giả sử a là xấp xỉ của A với sai số tuyệt đối giới hạn là E . Giả sử ta quy tròn a thành a' với sai số quy tròn tuyệt đối giới hạn là θ, tức là:

$$|a' - a| \le \theta.$$

Ta có

$$|a' - A| = |a' - a + a - A| \le |a' - a| + |a - A| \le \theta + E$$

Vậy có thể lấy θ +E làm sai số tuyệt đối giới hạn của a'. Như vậy việc quy tròn làm tăng sai số tuyệt đối giới hạn.

1.4. CÁC QUY TẮC TÍNH SAI SỐ

1.4.1. Mở đầu

Ta xét bài toán tổng quát hơn như sau:

Xét hàm số u của 2 biến số x và y:

$u = f(x,y)$

Giả sử x là xấp xỉ của giá trị đúng X, y là xấp xỉ của giá trị đúng Y và ta coi u là xấp xỉ của giá trị đúng $U = f(X,Y)$.

Cho biết sai số về x và y, hãy lập công thức tính sai số về u.

Cho biến x ta sẽ ký hiệu $\Delta x = x - X$ là số gia của x, còn dx là vi phân của x.

Theo định nghĩa về sai số tuyệt đối, ta có $|\Delta x| \le \Delta_x$

Theo công thức vi phân của hàm nhiều biến ta có:

$$du = \frac{\partial u}{\partial x} dx + \frac{\partial u}{\partial y} dy$$

Từ đây

$$\Delta u \approx \frac{\partial u}{\partial x} \Delta x + \frac{\partial u}{\partial y} \Delta y$$

Suy ra

$$\Delta_u = \left|\frac{\partial u}{\partial x}\right| \Delta_x + \left|\frac{\partial u}{\partial y}\right| \Delta_y \qquad (1.9)$$

1.4.2. Sai số của tổng

Cho $u = x + y$

Ta có

$$\frac{\partial u}{\partial x} = \frac{\partial u}{\partial y} = 1$$

Từ (1.9) suy ra

$$\Delta_u = \Delta_x + \Delta_y \qquad (1.10)$$

Ta có quy tắc sau:

Sai số tuyệt đối giới hạn của một tổng bằng tổng các sai số tuyệt đối giới hạn của các số hạng.

Ghi chú. Xét trường hợp $u = x - y$ và x, y cùng dấu. Lúc đó ta có

$$\delta_u = \Delta_u/|u| = (\Delta_x + \Delta_y)/|x-y|$$

Ta thấy rằng nếu $|x - y|$ rất bé thì sai số tương đối giới hạn rất lớn. Do đó trong tính toán người ta tìm cách tránh trừ những số gần nhau.

1.4.3. Sai số của tích

Cho $u = xy$

Ta có

$$\frac{\partial u}{\partial x} = y,\ \frac{\partial u}{\partial y} = x$$

Từ (1.9) suy ra

$$\Delta_u = |y|\Delta_x + |x|\Delta_y$$

Do đó $\delta_u = \Delta_u/|u| = \Delta_x/|x| + \Delta_y/|y| = \delta_x + \delta_y$

Vậy

$$\delta_u = \delta_x + \delta_y \qquad (1.11)$$

Ta có quy tắc sau:

Sai số tương đối giới hạn của một tích bằng tổng các sai số tương đối giới hạn của các số hạng của tích.

Xét trường hợp đặc biệt $u = x^n$ ta có

$$\delta_x n = n\,\delta_x \qquad (1.12)$$

1.4.4. Sai số của thương

Cho $u = x/y$

Ta có

$$\frac{\partial u}{\partial x} = \frac{1}{y},\ \frac{\partial u}{\partial y} = -\frac{x}{y^2}$$

Từ (1.9) suy ra

$$\Delta_u = |\frac{1}{y}|\Delta_x + |\frac{x}{y^2}|\Delta_y$$

Ta có

$$\Delta_u/|u| = \Delta_u . |\frac{y}{x}| = |\frac{y}{x}|(|\frac{1}{y}|\Delta_x + |\frac{x}{y^2}|\Delta_y) = |\frac{1}{x}|\Delta_x + |\frac{1}{y}|\Delta_y =$$

Suy ra:

$$\delta_{xy} = \delta_x + \delta_y \qquad (1.13)$$

Ta có quy tắc sau:

Sai số tương đối giới hạn của một thương bằng tổng các sai số tương đối giới hạn của các số hạng của thương.

1.4.5. Sai số của hàm bất kỳ

Cho $u = f(x_1, x_2, ..., x_n)$

Theo công thức vi phân của hàm nhiều biến ta có:

$$du = \frac{\partial u}{\partial x_1}dx_1 + \frac{\partial u}{\partial x_2}dx_2 + ... + \frac{\partial u}{\partial x_n}dx_n$$

Từ đây ta có

$$\Delta u \approx \frac{\partial u}{\partial x_1}\Delta x_1 + \frac{\partial u}{\partial x_2}\Delta x_2 + ... + \frac{\partial u}{\partial x_n}\Delta x_n$$

Suy ra

$$\Delta_u = |\frac{\partial u}{\partial x_1}| \Delta_{x_1} + |\frac{\partial u}{\partial x_2}| \Delta_{x_2} + ... + |\frac{\partial u}{\partial x_n}| \Delta_{x_n} \quad (1.14)$$

Ví dụ. Tính sai số tuyệt đối giới hạn và sai số tương đối giới hạn của thể tích hình cầu:

$V = (1/6)\pi d^3$

nếu cho đường kính $d = 3.7 \pm 0.05$ cm và $\pi = 3.14 \pm 0.0016$.

Giải.

Xem π và d là đối số của hàm V, áp dụng (1.12) và (1.13) ta có

$\delta_V = \delta_\pi + 3\delta_d$ (Hệ số 1/6 không ảnh hương đến sai số tương đối)

$\delta_\pi = 0.0016/3.14 = 0.0005$

$\delta_d = 0.05/3.7 = 0.0135$

Suy ra $\delta_V = 0.0005 + 3 * 0.0135 = 0.04$

Mặt khác $V = (1/6)\pi d^3 = 26.5$ cm^3

Ta có $\Delta_V = |V|*\delta_V = 26.5*0.04 = 1.06 \approx 1.1$ cm^3

$V = 26.5 \pm 1.1$ cm^3

1.5. SAI SỐ TÍNH TOÁN VÀ SAI SỐ PHƯƠNG PHÁP

Như chúng tôi đã nhắc đến ở trên, khi giải một bài toán phức tạp ta phải thay bài toán đó bằng bài toán đơn giản hơn để có thể tính toán bằng tay hoặc bằng máy. Phương pháp thay bài toán phức tạp bằng một phương pháp đơn giản tính được như vậy gọi là ***phương pháp gần đúng***. Sai số do phương pháp gần đúng tạo ra gọi là sai số phương pháp. Mặc dầu bài toán đã ở dạng đơn giản, có thể tính toán được bằng tay hoặc trên máy tính, nhưng trong quá trình tính toán ta thường xuyên phải làm tròn các kết quả trung gian. Sai số tạo ra bởi tất cả những lần quy tròn như vậy được gọi là ***sai số tính toán***. Trong thực tế việc đánh giá các loại sai số, nhất là sai số tính toán nhiều khi là bài toán rất khó thực hiện. Để hiểu rõ hơn bản chất của sai số phương pháp và sai số tính toán ta xét ví dụ sau:

Ta biết rằng với số x bất kỳ ta có

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + ... + \frac{x^n}{n!} + ...$$

Công thức này có thể dùng để tính giá trị e^x. Tuy nhiên đây là tổng vô hạn, nên trong thực tế ta chỉ tính được tổng $S_n = 1 + \frac{x}{1!} + \frac{x^2}{2!} + ... + \frac{x^n}{n!}$, nghĩa là chúng ta đã dùng phương pháp gần đúng. Khi tính tổng S_n ta lại thường xuyên phải làm tròn, do đó ta lại gặp sai số khi tính toán S_n. Việc đưa ra một đánh giá về sai số tổng hợp của cả hai loại sai số trên là bài toán rất phức tạp.

1.6. SỰ ỔN ĐỊNH CỦA MỘT QUÁ TRÌNH TÍNH TOÁN

Xét một quá trình tính toán về lý thuyết có vô hạn bước để tính ra một đại lượng nào đó. Ta nói rằng quá trình tính là *ổn định nếu sai số tính toán tức là sai số quy tròn tích lũy lại không tăng vô hạn*. Nếu sai số đó tăng vô hạn thì ta nói quá trình tính là không ổn định.

Rõ ràng nếu quá trình tính không ổn định thì không có hy vọng tính được đại lượng cần tính với sai số nhỏ hơn sai số cho phép.

Để kiểm tra tính ổn định của một quá trình tính toán thường người ta giả sử sai số chỉ xảy ra tại một bước, các bước sau đó coi như không có sai số khác phát sinh. Nếu cuối cùng sai số tính toán không tăng vô hạn thì coi như quá trình tính là ổn định.

1.7. MỘT VÀI ĐIỀU VỀ MỐI QUAN HỆ GIỮA THỰC TẾ VÀ MÔ HÌNH

Theo những điều vừa nói trên đây thì chúng ta luôn hiểu thực tế là tuyệt đối đúng, sai số chỉ xảy ra khi ta muốn mô hình hóa thực tế và tiến hành tính toán mô hình đó. Thực vậy, chúng ta có cảm giác rằng giới tự nhiên đang hoạt động một cách chính xác: hệ mặt trời đã có khoảng 5 tỷ năm tuổi, nhưng sự vận hành của nó có vẻ vẫn hoàn hảo: hàng ngày mặt trời mọc, mặt trời lặn đều theo quy luật. Cứ sau 365 ngày + 1/4 ngày thì quả đất quay đủ một vòng quanh mặt trời và hầu hết các vùng trên trái đất đều trải qua bốn mùa. Chúng ta có thể hình dung rằng chỉ cần mỗi năm sự vận hành của các hành tinh sai lệch đi chút ít thì trong hàng tỷ năm sai số tích lũy có thể sẽ gây nên những biến cố khôn lường! Tuy nhiên theo các nhà thiên văn thì sự vận hành của các hành tinh không tuyệt đối hoàn hảo như ta tưởng. Xét vị trí của mặt trời và trái đất chẳng hạn, theo lý thuyết thì nếu ngày hôm nay mặt trời đứng ở vị trí giữa bầu trời tính từ đông sang tây thì sau 24 giờ nữa nó cũng ở vị trí giữa bầu trời (tất nhiên là có thể chếch về phía nam nếu ta đang ở Việt nam). Nhưng trong thực tế không phải như vậy. Các nhà thiên văn đã không thể xây dựng được múi giờ một cách chính xác và nhất quán nếu dựa vào vị trí của mặt trời. Nói cụ thể hơn, nếu dựa vào vị trí mặt trời của năm nay làm múi giờ cho các vùng trên trái đất thì năm sau thời gian đó không còn thích hợp cho quỹ đạo của mặt trời nữa, mà có khác đi chút ít. Chính vì sự "đỏng đảnh" của mặt trời như vậy nên các nhà thiên văn đã đưa ra khái niệm mặt trời trung bình và thời gian trung bình. So với mặt trời trung bình và thời gian trung bình thì hàng năm mặt trời thật đi lệch trong khoảng thời gian từ -14,3 đến +16,3 phút. Tuy nhiên sở dĩ các sai số này không tích lũy từ năm này sang năm khác là vì các sai số giao động quanh vị trí trung bình và triệt tiêu lẫn nhau theo thời gian.

Nghĩa là, không chỉ mô hình của chúng ta, mà ngay cả giới tự nhiên cũng có những sai số. Tuy nhiên các sai số trong giới tự nhiên đều có quy luật và thường triệt tiêu lẫn nhau, do đó không làm ảnh hưởng đến sự vận hành của các vật thể.

BÀI TẬP

Bài 1. Khi đo 1 số góc ta được các giá trị sau:

a= 21^{o}37’3”; b=1^{o}10’

Hãy xác định sai số tương đối của các số xấp xỉ đó biết rằng sai số tuyệt đối trong phép đo là 1”.

Bài 2. Hãy xác định sai số tuyệt đối của các số xấp xỉ sau đây cho biết sai số tương đối của chúng:

a) a= 13267 ; δ_a=0,1%

b) b=2,32; δ_b=0,7%

Bài 3. Hãy xác định số các chữ số đáng tin trong các số a,b với sai số như sau:

a) a= 0,3941; $\Delta_a=0,25.10^{-2}$

b) b=38,2543; $\Delta_a= 0,27.10^{-2}$

Bài 4. Hãy xác định số những chữ số đáng tin trong các số a với sai số tương đối như sau:

a) a=1,8921; $\delta_a=0,1.10^{-2}$

b) a=22,351; δ_a=0,1

Bài 5. Hãy qui tròn các số dưới đây(xem là đúng) với 3 chữ số có nghĩa đáng tin và xác định sai số tuyệt đối Δ và sai số tương đối δ của chúng:

a) a= 2,514; b) 0,16152

c) 0,01204; d) –0,0015281

Bài 6. Hãy xác định giá trị của các hàm số dưới đây cùng với sai số tuyệt đối và sai số tương đối ứng với những giá trị của các đối số cho với mọi chữ số có nghĩa đều đáng tin.

a) $u=\ln(x+y^2)$; x=0,97; y=1,132

b) $u=(x+y)^2z$; x=3,28; y=0,932; z=1,132

CHƯƠNG 2
CÁC PHƯƠNG PHÁP SỐ TRONG ĐẠI SỐ TUYẾN TÍNH

MỤC ĐÍCH, YÊU CẦU:

Sau khi nghiên cứu chương 1, yêu cầu sinh viên:

1. Hiểu và nắm được các phương pháp tìm nghiệm đúng, nghiệm xấp xỉ của hệ phương trình tuyến tính.

2. Biết cách ứng dụng các phương pháp trên vào việc tính định thức của ma trận, tìm ma trận nghịch đảo, giải quyết các bài toán thực tế.

3. Biết cách đánh giá sai số của từng phương pháp

2.1. MA TRẬN VÀ ĐỊNH THỨC

2.1.1. Ma trận

Cho ma trận chữ nhật A cấp m x n:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ . & . & \dots & . \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

ở đây a_{ij} là các số thực. Ma trận này có m hàng và n cột. Khi m = n ta có ma trận cấp nxn và *được gọi tắt là ma trận vuông cấp n*.

Ma trận vuông cấp n mà mọi phần tử nằm ngoài đường chéo chính bằng 0, tức là $a_{ij} = a_{ji} = 0$ với $i \neq j$, được gọi là ***ma trận đường chéo***. Nếu ma trận đường chéo có $a_{ii} = 1$ thì ta gọi A là ma trận đơn vị và ta thường ký hiệu là E hoặc I.

Ma trận vuông A được gọi là ***ma trận tam giác trên***, nếu A có dạng

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

Tương tự, ma trận vuông A được gọi là ***ma trận tam giác dưới***, nếu A có dạng:

$$A = \begin{bmatrix} a_{11} & 0 & ... & 0 \\ a_{21} & a_{22} & ... & 0 \\ . & . & ... & . \\ a_{n1} & a_{n2} & ... & a_{nn} \end{bmatrix}$$

Ma trận chữ nhật A^T cấp n x m được gọi là ***ma trận chuyển vị*** của ma trận A cấp m x n nếu:

$$A^T = \begin{bmatrix} a_{11} & a_{21} & ... & a_{m1} \\ a_{12} & a_{22} & ... & a_{m2} \\ . & . & ... & . \\ a_{1n} & a_{2n} & ... & a_{mn} \end{bmatrix}$$

2.1.2. Định thức của ma trận

Trước khi đưa ra định nghĩa định thức của ma trận, chúng tôi giới thiệu khái niệm hoán vị chẵn, hoán vị lẻ của một tập hợp n số nguyên {1, 2, ... , n}.

Cho $\alpha = (i_1, i_2,..., i_n)$ là một hoán vị của tập {1,2,...,n}. Ta xét tất cả các cặp (i_k, i_h), trong đó $k < h$. Nếu $i_k > i_h$ thì ta gọi cặp (i_k, i_h) là cặp ngược, tức là các giá trị i_k, i_h được sắp xếp ngược với k,h. Nếu trong α số cặp ngược là chẵn thì ta gọi α là *hoán vị chẵn*, ngược lại thì ta gọi α là *hoán vị lẻ*.

Với mỗi ma trận vuông A cấp n:

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

tồn tại một số thực được gọi là ***định thức*** của ma trận A, ký hiệu là det A, được xác định bởi công thức:

$$\det A = \sum_{\alpha} s(i_1, i_2,..., i_n)\ a_{1i_1} a_{2i_2} ... a_{ni_n} \qquad (2.0)$$

với $\alpha = (i_1, i_2,..., i_n)$ chạy trong tập tất cả các hoán vị của tập {1,2,...,n}, và

$$s(i_1, i_2,..., i_n) = \begin{cases} 1 \text{ nếu } \alpha \text{ là hoán vị chẵn} \\ -1 \text{ nếu } \alpha \text{ là hoán vị lẻ} \end{cases}$$

Định thức của ma trận còn được ký hiệu là

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

Với mỗi ma trận chữ nhật A cấp m x n bất kỳ ta có thể tính định thức của tất cả các ma trận con vuông cấp k, với $k \leq \min(m, n)$. Nếu tồn tại một số r sao cho có một ma trận con cấp r có định thức khác 0, còn mọi ma trận con vuông cấp lớn hơn r đều bằng 0 thì ta nói rằng r là ***hạng của ma trận*** A.

Các phép biến đổi sơ cấp sau đây không làm biến đổi hạng của ma trận:

- Đổi chỗ 2 hàng hoặc 2 cột bất kỳ.
- Nhân một hàng hay một cột bất kỳ với một số khác không.
- Cộng các thành phần tương ứng của 2 hàng hoặc hai cột bất kỳ.

Các phép biến đổi sơ cấp sẽ được sử dụng để tính định thức của ma trận và tìm nghiệm của hệ phương trình tuyến tính.

Ma trận E được gọi là ma trận đơn vị cấp n nếu E là ma trận vuông cấp n và E có dạng

$$E = \begin{bmatrix} 1 & 0 & ... & 0 \\ 0 & 1 & ... & 0 \\ . & . & ... & . \\ 0 & 0 & ... & 1 \end{bmatrix}$$

2.1.3. Các phương pháp tính định thức

a. Tính định thức dựa trực tiếp vào định nghĩa

Ta có thể dùng (2.0) để tính định thức của một ma trận trên máy tính. Tuy nhiên cách tính này đòi hỏi khoảng c*n! phép tính. Đây là con số khổng lồ với n không lớn lắm. Ví dụ với máy tính hiện đại nhất hiện nay cũng cần hàng triệu năm để tính định thức của ma trận cấp n = 25.

b. Tính định thức dựa vào công thức khai triển theo hàng

Cho A là ma trận vuông cấp n và a_{ij} là một phần tử bất kỳ của nó. Định thức của ma trận con cấp n-1 sau khi "xóa" hàng thứ i và cột thứ j đi và không thay đổi vị trí các thành phần còn lại, được gọi là *minor* của phần tử a_{ij} , và được ký hiệu là M_{ij}. Giá trị $A_{ij} = (-1)^{i+j} M_{ij}$ được gọi là phần bù đại số của phần tử a_{ij}. Ta có các công thức sau để tính định thức ma trận vuông cấp n thông qua việc tính định thức của các ma trận con cấp bé hơn:

Khai triển định thức theo hàng thứ i:

$$\det A = \sum_{j=1}^{n} a_{ij} A_{ij}$$

Khai triển định thức theo cột thứ j:

$$\det A = \sum_{i=1}^{n} a_{ij} A_{ij}$$

Áp dụng các công thức trên đây ta có thể dùng thuật toán đệ quy sau đây để tính định thức của ma trận vuông cấp n :

Nếu $n = 1$: $A_{11} = 1$; $\det A = a_{11} A_{11}$

$n > 1$: $\det A = \sum_{j=1}^{n} a_{1j} A_{1j}$

Tuy nhiên, cũng như cách tính trực tiếp, cách tính này cần khoảng c*n! phép tính, và như vậy không thể thực hiện được trên máy tính hiện đại nhất hiện nay dù chỉ với n không lớn lắm. Rõ ràng việc phân tính thuật toán giúp chúng ta đánh giá được thời gian tính toán trên máy tính và nếu thời gian đó là quá lớn thì chúng ta khỏi phải tốn công vô ích viết chương trình và chạy thử.

c. Tính định thức bằng cách chuyển ma trận về dạng tam giác trên

Ta sẽ biến đổi để đưa ma trận A về dạng ma trận tam giác trên

$$B = \begin{bmatrix} b_{11} & b_{12} & ... & b_{1n} \\ 0 & b_{22} & ... & B_{2n} \\ . & . & ... & . \\ 0 & 0 & ... & b_{mn} \end{bmatrix}$$

Vậy $\det A = \det B = b_{11} b_{22} ... b_{nn}$

2.1.4. Ma trận nghịch đảo

Ma trận nghịch đảo của một ma trận vuông A cấp n là ma trận được ký hiệu là A^{-1}, thoả mãn điều kiện

$$A^{-1}A = A A^{-1} = E$$

Trong đó E là ma trận đơn vị. Có thể chứng minh rằng để thỏa mãn điều kiện trên thì bắt buộc A^{-1} phải là ma trận vuông, và ma trận đảo nếu tồn tại là duy nhất.

Điều kiện tồn tại của ma trận nghịch đảo: Ma trận vuông A cấp n có ma trận nghịch đảo ***khi và chỉ khi*** $\det A \neq 0$.

Cách tính ma trận nghịch đảo:

Gọi A_{ij} là phần bù đại số của phần tử a_{ij} , khi đó ta có:

$$A^{-1} = \frac{1}{\det A} \begin{bmatrix} A_{11} & A_{21} & ... & A_{n1} \\ A_{12} & A_{22} & ... & A_{n2} \\ . & . & ... & . \\ A_{1n} & A_{2n} & ... & A_{nn} \end{bmatrix}$$

Tuy nhiên công thức này chỉ có ý nghĩa lý thuyết, không thể áp dụng để tính trực tiếp ma trận đảo trên máy tính được vì số phép tính đòi hỏi quá lớn.

Trong phần sau ta sẽ áp dụng phương pháp khử Gauss-Jordan để tính ma trận nghịch đảo với số phép tính nhỏ hơn nhiều (khoảng n^3)

2.2. HỆ PHƯƠNG TRÌNH ĐẠI SỐ TUYẾN TÍNH

Xét một hệ phương trình gồm n phương trình tuyến tính với n ẩn số $x_1, x_2,...,x_n$ như sau:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n \end{cases} \quad (2.1)$$

Hệ phương trình này có thể viết dưới dạng ma trận $Ax = b$, trong đó

$$A = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ . & . & \ldots & . \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{bmatrix}, x = \begin{pmatrix} x_1 \\ x_2 \\ . \\ x_n \end{pmatrix}, b = \begin{pmatrix} b_1 \\ b_2 \\ . \\ b_n \end{pmatrix}$$

Nếu $\det A \neq 0$ thì nghiệm của hệ (2.1) có thể tính theo công thức $x = A^{-1}b$. Áp dụng công thức tính ma trận đảo ta có thể biến đổi và dẫn đến lời giải được diễn tả bằng định lý Cramer như sau:

Định lý Cramer. Gọi A_j là ma trận nhận được từ ma trận A bằng cách thay cột thứ j bằng cột b, khi đó hệ (2.1) có nghiệm duy nhất và x_j được tính bởi công thức

$$x_j = \frac{\det A_j}{\det A}$$

Tuy nhiên trong thực hành người ta không dùng công thức này để tính nghiệm vì số phép tính quá lớn. Người ta dùng những phương pháp hữu hiệu hơn mà chúng tôi sẽ giới thiệu sau đây.

2.2.1. Phương pháp trực tiếp giải hệ phương trình tuyến tính

Giả sử ta giải hệ phương trình(2.1)

a. Phương pháp khử Gauss

Phương pháp khử Gauss dùng cách khử dần các ẩn để đưa hệ phương trình đã cho về một dạng tam giác trên rồi giải hệ tam giác này từ giới lên trên, không phải tính một định thức nào

Phương pháp này được thực hiện qua các bước sau:

Quá trình xuôi:

- Bước 0: Dùng phương trình đầu tiên để khử x_1 trong n-1 phương trình còn lại. Giả sử $a_{11}\neq 0$. (Để cho công thức đơn giản , trước khi khử ta có thể chia phương trình thứ nhất cho a_{11}).

  Cụ thể để khử x_1 ở hàng thứ k(k=2,3,…n) ta phải tính lại các hệ số a_{kj} ở hàng thứ k (j=1,2,..n+1) như sau: $a_{kj}=a_{kj}-a_{1j}*a_{k1}/a_{11}$

  . . .

- **Bước 1:** Dùng phương trình thứ 2 để khử x_2 trong n-2 phương trình còn lại phía sau. Giả sử $a_{22}\neq 0$. (Để cho công thức đơn giản, trước khi khử ta có thể chia phương trình thứ hai cho a_{22}).

  Cụ thể để khử x_2 ở hàng thứ k (k=3,4,…n) ta phải tính lại các hệ số a_{kj} ở hàng thứ k (j=2,..n+1) như sau: $a_{kj}=a_{kj}-a_{2j}*a_{k2}/a_{22}$

  …….

- **Bước i:** Dùng phương trình i để khử x_i trong các phương trình thứ i+1,i+2, ..., n. Giả sử $a_{ii}\neq 0$. Để cho công thức đơn giản, trước khi khử ta có thể chia phương trình thứ i cho a_{ii}).

  Cụ thể để khử x_i ở hàng thứ k (k=i+1,…n) ta phải tính lại các hệ số a_{kj} ở hàng thứ k (j=i,..n+1) như sau: $a_{kj}=a_{kj}-a_{ij}*a_{ki}/a_{ii}$

- **Bước n-1:** Dùng phương trình thứ n-1 để khử x_{n-1} trong phương trình thứ n.Giả sử $a_{n-1\ n-1}\neq 0$. (Để cho công thức đơn giản, trước khi khử ta có thể chia phương trình thứ n-1 cho $a_{n-1\ n-1}$)

  Cụ thể để khử x_{n-1} ở hàng thứ n ta phải tính lại các hệ số a_{nj} ở hàng thứ n (j=n-1,n,n+1) như sau: $a_{nj}=a_{nj}-a_{n-1j}*a_{n-1i}/a_{n-1n-1}$

Kết thúc quá trình khử.

Chú ý:

Trong quá trình giải xuôi ta giả thiết $a_{11}\neq 0$, $a_{22}\neq 0$,$a_{33}\neq 0$,...,$a_{n-1\ n-1}\neq 0$. Nếu 1 trong các hệ số đó bằng không thì quá trình không tiếp tục được. Lúc đó ta phải thay đổi cách tính.

Giả sử khi khử x_1 ta gặp $a_{11}=0$ thì ta nhìn các hệ số a_{21}, a_{31} ...a_{n1} của x_1 ở các phương trình phía dưới, nếu có hệ số nào khác không ta có thể lấy nó thay cho vai trò của a_{11} bằng cách hoán vị hai phương trình. Nếu tất cả các hệ số số a_{11}, a_{21}, a_{31} ...,a_{n1} đều bằng không thì hệ đã cho suy biến. Vậy tốt nhất là trước khi khử x_1 ta chọn trong các hệ số a_{11}, a_{21}, a_{31} ...,a_{n1} hệ số có giá trị tuyệt đối lớn nhất làm trụ thứ nhất**(gọi là trụ tối đại thứ nhất)** rồi hoán vị hàng thứ nhất cho hàng có giá trị tuyệt đối lớn nhất). Tức là ta chọn hàng r sao cho:

$|a_{r1}| = \max\{|a_{k1}| / k=1,2,...,n\}$ Sau đó ta đổi hàng r cho hàng 1.

Tương tự trong các bước khử x_2,... x_{n-1} , trước khi khử ta cũng **tìm trụ tối đại**:

$|a_{ri}| = \max\{|a_{ki}| / k=i,i+1,...,n\}$ (với i=2,3,…,n-1)

Sau đó ta đổi hàng r cho hàng i.

Sau khi thực hiện xong quá trình giải xuôi hệ phương trình (2.1) có dạng:

Dạng1: Tại các bước (bước i) ta không chia cho hệ số a_{ii}

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

hoặc: Dạng 2: Tại các bước (bước i) ta chia cho hệ số a_{ii}:

$$\begin{cases} x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ \quad x_2 + \ldots + a_{2n}x_n = b_2 \\ \quad \ldots\ldots\ldots\ldots \\ \quad\quad\quad x_n = b_n \end{cases}$$

Xuất phát từ phương trình thứ n ta lần lượt tính được các giá trị x_i bằng các công thức của quá trình giải ngược sau:

Quá trình giải ngược

$x_n = b_n/a_{nn}$ hoặc ($x_n=b_n$)

. . .

$$x_i = (b_i - (\sum_{j=i+1}^{n} a_{ij}x_j))/a_{ii}) \text{ hoặc } (b_i - (\sum_{j=i+1}^{n} a_{ij}x_j)), \quad i = n-1, n-2, \ldots, 1$$

Để việc viết chương trình được đơn giản, khi cài đặt trên máy tính ta dùng một mảng a thay cho cả ma trận a và vec tơ b. Tức là

$$\begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} & a_{1,(n+1)} \\ a_{21} & a_{22} & \ldots & a_{2n} & a_{2,(n+1)} \\ . & . & \ldots & . & . \\ a_{n1} & a_{n2} & \ldots & a_{nn} & a_{n,(n+1)} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} & b_1 \\ a_{21} & a_{22} & \ldots & a_{2n} & b_2 \\ . & . & \ldots & . & . \\ a_{n1} & a_{n2} & \ldots & a_{nn} & b_n \end{bmatrix}$$

Ta áp dụng các phép biến đổi sơ cấp như vừa trình bày để biến đổi ma trận chữ nhật cấp nx(n+1) trên đây về dạng

$$\begin{bmatrix} 1 & a'_{12} & \ldots & a'_{1n} & a'_{1,(n+1)} \\ 0 & 1 & \ldots & a'_{2n} & a'_{2,(n+1)} \\ . & . & \ldots & . & . \\ 0 & 0 & \ldots & 1 & a'_{n,(n+1)} \end{bmatrix}$$

Ví dụ:Giải hệ phương trình sau bằng phương pháp khử Gauss:

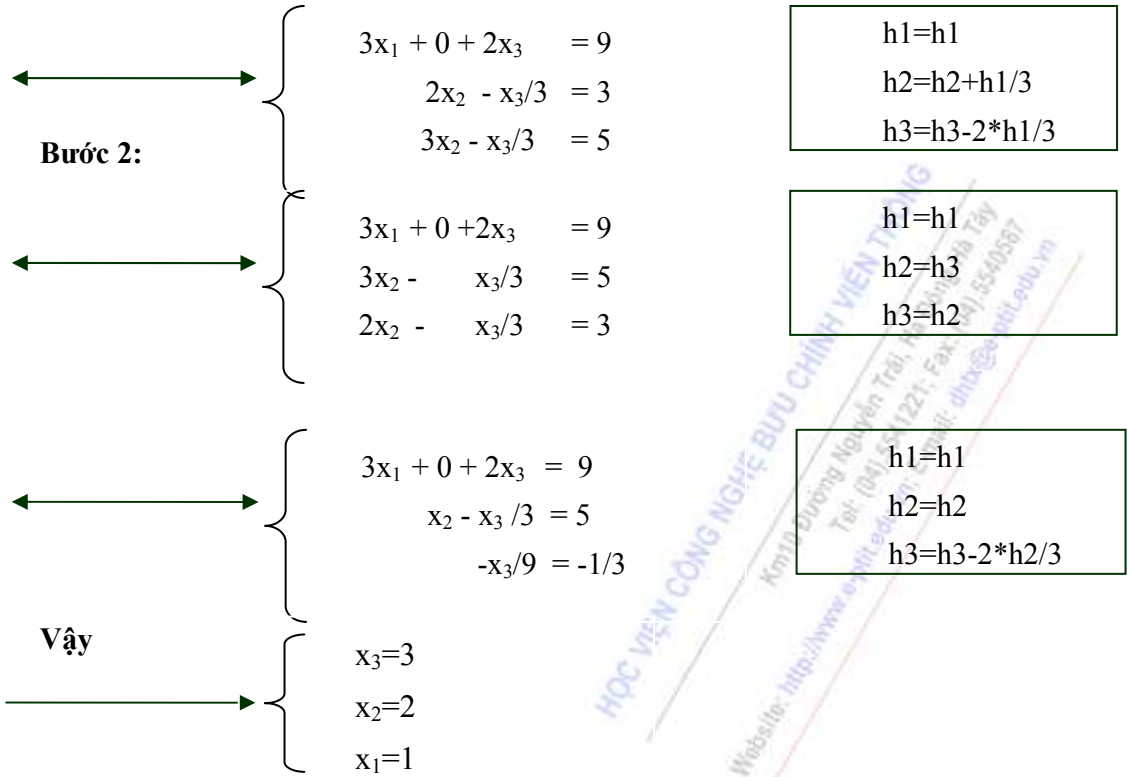
$$\begin{cases} 2x_1 + 3x_2 + x_3 = 11 \\ -x_1 + 2x_2 - x_3 = 0 \\ 3x_1 + 2x_3 = 9 \end{cases}$$

Bước1: Hệ phương trình trên tương đương với:

$\longleftrightarrow$

$$\begin{cases} 3x_1 + 2x_3 = 9 \\ -x_1 + 2x_2 - x_3 = 0 \\ 2x_1 + 3x_2 + x_3 = 11 \end{cases}$$

h1=h3
h2=h2
h3=h1



Chương trình minh họa.

Sau đây là đoạn chương trình chính thể hiện (mô tả) thuật toán khử Gauss.

```
/*Giai he phuong trinh tuyen tinh dung khu Gauss, ma tran vuong n,
  cac phan tu cot thu n+1 la vecto b*/
/*Dua ma tran a ve dang tam giac tren Giai he phuong trinh tuyen tinh.
  Tra ve gia tri true neu co nghiem */
int khugauss(kmatran a,double *x,int n)
 {
  int i,j,k,h;double tmp,p;kmatran aa;
  int n1=n+1;
  for(i=1;i<=n;i++)
  for(j=1;j<=n1;j++) aa[i][j]=a[i][j];
  for(i=1;i<=n;i++) //Vong lap cac buoc khu
   {//Tim hang co phan tu dau lon nhat
    h=i;
    for(k=i+1;k<=n;k++)
     if(fabs(a[k][i])>fabs(a[h][i]) {h=k;}
    if(a[h][i])==0) {cout<<"Ma tran suy bien";delay(1000);return false;}
```

```
 if(h!=i) //Doi hang i va hang h vi a[h][i] > a[i][i]
  {int j;double tmp;
   for(j=i;j<=n1;j++)
    {tmp=a[i][j];a[i][j]=a[h][j];a[h][j]=tmp;}
  }
  //chuyen he so a[i][i] = 1
  tmp=a[i][i];
  for(j=i;j<=n1;j++) a[i][j] = a[i][j]/tmp;
  //Bat tinh lai cac hang
  for(k=i+1;k<=n;k++)
   {p=a[k][i];
    /*Vi ta biet a[k][i] =0 sau bien doi,
  chi tinh tu a[k][i+1]*/
    for(j=i+1;j<=n1;j++) a[k][j]=a[k][j] - p*a[i][j];
   }
 }
x[n]=a[n][n+1];
for(i=n-1;i>=1;i--)
 {double xx=0;
  for(j=i+1;j<=n;j++) xx=xx+a[i][j]*x[j];
  x[i]=a[i][n+1]-xx;//b[i]-xx
 }
//Dat cac gia tri phi duoi duong cheo chinh bang 0(phan nay khong can)
for(i=2;i<=n;i++)
for(j=1;j<i;j++) a[i][j]=0;

//Thu lai
kvecto bb;
for(i=1;i<=n;i++)
 {bb[i]=aa[i][1]*x[1];
  for(j=2;j<=n;j++) bb[i]+=aa[i][j]*x[j];
 }
//Dua ket qua vao tep ketqua
return true;
}
```

b. Phương pháp khử Gauss-Jordan

Phương pháp khử Gauss-Jordan dùng cách khử dần các ẩn để đưa hệ phương trình đã cho về một dạng ma trận đường chéo rồi giải hệ phương trình này, không phải tính một định thức nào

Phương pháp này được thực hiện qua các bước sau:

- **Bước 1:** Dùng phương trình đầu tiên để khử x_1 trong n-1 phương trình còn lại, cách làm tương tự như phương pháp khử để tính định thức... (Để cho công thức đơn giản, trước khi khử ta có thể chia phương trình thứ nhất cho a_{11}).

  Cụ thể để khử x_1 ở hàng thứ k(k=2,3,…n) ta phải tính lại các hệ số a_{kj} ở hàng thứ k (j=1,2,..n+1) như sau: $a_{kj}=a_{kj}-a_{1j}*a_{k1}/a_{11}$

  . . .

- **Bước i:** Dùng phương trình i để khử x_i trong các phương trình thứ 1,2, i-1,i+1,i+2,...,n.. (Để cho công thức đơn giản , trước khi khử ta có thể chia phương trình thứ i cho a_{ii})

  Cụ thể để khử x_i ở hàng thứ k (k=1,2, i-1,i+1,i+2,...,n.) ta phải tính lại các hệ số a_{kj} ở hàng thứ k (j=i,..n+1) như sau: $a_{kj}=a_{kj}-a_{ij}*a_{ki}/a_{ii}$

  . . .

- **Bước n:** Dùng phương trình thứ n để khử x_n trong phương trình thứ 1,2, ..., n-1.. (Để cho công thức đơn giản, trước khi khử ta có thể chia phương trình thứ n cho a_{nn})

  Cụ thể để khử x_n ở hàng thứ k(k=1,2, ..,n-1.) ta phải tính lại các hệ số a_{kj} ở hàng thứ k (j=n,n+1) như sau: $a_{kj}=a_{kj}-a_{nj}*a_{kn}/a_{nn}$

Tương tự phép khử Gauss tại mỗi bước, trước khi khử ta phải chọn trụ tối đại. Cụ thể tại bước i ta luôn chọn hàng có phần tử a_{ri} có giá trị tuyệt đối lớn nhất rồi đổi cho hàng thứ i cho hàng thứ r.

Hệ phương trình sau khi khử có dạng:

$$\begin{cases} a_{11}x_1 & & & = b_1 \\ & a_{22}x_2 & & = b_2 \\ & \cdots\cdots\cdots & & \\ & & a_{nn}x_n & = b_n \end{cases}$$

Hoặc (Nếu tại các bước (bước i) ta chia cho hệ số a_{ii}):

$$\begin{cases} x_1 & & & = b_1 \\ & x_2 & & = b_2 \\ & \cdots\cdots\cdots & & \\ & & x_n & = b_n \end{cases}$$

Tức là ta đã có các nghiệm mà không cần phải tính toán thêm.

Cũng như trong phương pháp khử Gauss, khi cài đặt trên máy tính ta dùng một mảng a thay cho cả ma trận A và vec tơ b. Tức là

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} & a_{1,(n+1)} \\ a_{21} & a_{22} & \dots & a_{2n} & a_{2,(n+1)} \\ . & . & \dots & . & . \\ a_{n1} & a_{n2} & \dots & a_{nn} & a_{n,(n+1)} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ . & . & \dots & . & . \\ a_{n1} & a_{n2} & \dots & a_{nn} & b_n \end{bmatrix}$$

Ta áp dụng các phép biến đổi sơ cấp như vừa trình bày để biến đổi ma trận chữ nhật cấp n x (n+1) trên đây về dạng

$$\begin{bmatrix} 1 & 0 & \dots & 0 & a'_{1,(n+1)} \\ 0 & 1 & \dots & 0 & a'_{2,(n+1)} \\ . & . & \dots & . & . \\ 0 & 0 & \dots & 1 & a'_{n,(n+1)} \end{bmatrix}$$

Vậy ta có $x_i = a'_{i,(n+1)}$

Ví dụ:Giải hệ phương trình sau bằng phương pháp khử Gauss-Jordan:

$$\begin{cases} 2x_1 + 3x_2 + x_3 = 11 \\ -x_1 + 2x_2 - x_3 = 0 \\ 3x_1 + 2x_3 = 9 \end{cases}$$

Bước1: Hệ phương trình trên tương đương với:

$\longleftrightarrow$

$$\begin{cases} 3x_1 + 2x_3 = 9 \\ -x_1 + 2x_2 - x_3 = 0 \\ 2x_1 + 3x_2 + x_3 = 11 \end{cases}$$

h1=h3
h2=h2
h3=h1

Bước 1:

$\longleftrightarrow$

$$\begin{cases} 3x_1 + 0 + 2x_3 = 9 \\ 2x_2 - x_3/3 = 3 \\ 3x_2 - x_3/3 = 5 \end{cases}$$

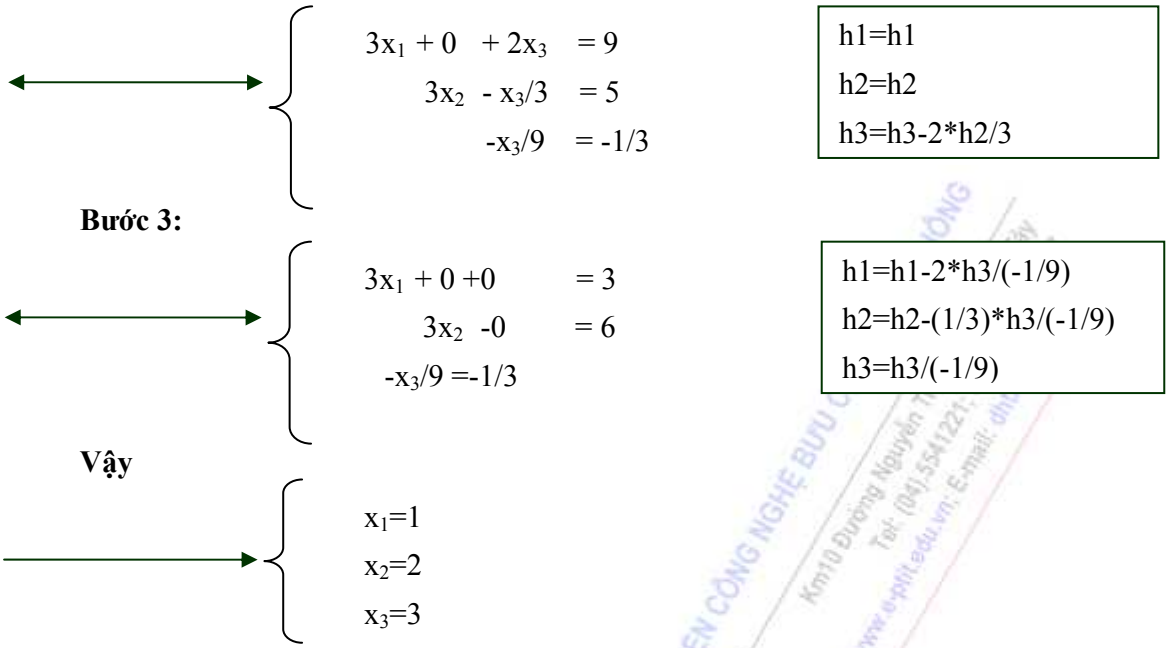
h1=h1
h2=h2+h1/3
h3=h3-2*h1/3

Bước 2:

$\longleftrightarrow$

$$\begin{cases} 3x_1 + 0 + 2x_3 = 9 \\ 3x_2 - x_3/3 = 5 \\ 2x_2 - x_3/3 = 3 \end{cases}$$

h1=h1
h2=h3
h3=h2



Chương trình minh họa.

Sau đây là đoạn chương trình chính thể hiện (mô tả) thuật toán khử Gauss-Jordan.

```
int gjordan(kmatran a,double *x,int n)
 {int i,j,k,h;double tmp,p;kmatran aa;
  int n1=n+1;
  for(i=1;i<=n;i++)
  for(j=1;j<=n1;j++) aa[i][j]=a[i][j];
  for(i=1;i<=n;i++) //Vong lap cac buoc khu
   {//Tim hang co phan tu dau lon nhat
    h=i;
    for(k=i+1;k<=n;k++)
     if(fabs(a[k][i])>fabs(a[h][i]) {h=k;}
    if(a[h][i]==0) {cout<<"Ma tran suy bien";delay(1000);return false;}
    if(h!=i) //Doi hang i va hang h vi a[h][i] > a[i][i]
     {int j;double tmp;
      for(j=i;j<=n1;j++)
       {tmp=a[i][j];a[i][j]=a[h][j];a[h][j]=tmp;}
     }
    //chuyen he so a[i][i] = 1
```

```
 tmp=a[i][i];
 for(j=i;j<=n1;j++) a[i][j] = a[i][j]/tmp;
 //Bat tinh lai cac hang
 for(k=1;k<=n;k++)
  {if(k==i) continue;
   p=a[k][i];
   /*Vi ta biet a[k][i] =0 sau bien doi,
chi tinh tu a[k][i+1]*/
   for(j=i+1;j<=n1;j++) a[k][j]=a[k][j] - p*a[i][j];
  }
}
for(i=1;i<=n;i++) x[i]=a[i][n+1];

 /*Dat cac gia tri khong o tren duong cheo chinh bang 0
   (phan nay khong can)*/
for(i=1;i<=n;i++)
for(j=1;j<=n;j++) {if(i!=j) a[i][j]=0;}

//Thu lai
kvecto bb;
for(i=1;i<=n;i++)
 {bb[i]=aa[i][1]*x[1];
  for(j=2;j<=n;j++) bb[i]+=aa[i][j]*x[j];
 }
//Dua ket qua vao tep ketqua
return true;
```

2.2.2. Áp dụng phương pháp khử Gauss-Jordan để tính ma trận nghịch đảo

Để giải hệ n phương trình n ẩn Ax = b, trong phương pháp khử Gauss-Jordan ta đã dùng các phép biến đổi sơ cấp để đưa phương trình này về dạng

Ex = b'

Vì Ex = x, do đó ta có x=b'. Nếu B là một ma trận chữ nhật cấp n x k tùy ý, ta có thể áp dụng phương pháp khử Gauss-Jordan để giải đồng thời k hệ n phương trình n ẩn:

AX = B (2.2)

trong đó

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1k} \\ x_{21} & x_{22} & \dots & x_{2k} \\ . & . & \dots & . \\ x_{n1} & x_{n2} & \dots & x_{nk} \end{bmatrix}$$

$$B = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1k} \\ b_{21} & b_{22} & \dots & b_{2k} \\ . & . & \dots & . \\ b_{n1} & b_{n2} & \dots & b_{nk} \end{bmatrix}$$

Ta viết ma trận B bên phải ma trận A như sau:

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} & b_{11} & b_{12} & \dots & b_{1k} \\ a_{21} & a_{21} & \dots & a_{2n} & b_{21} & b_{22} & \dots & b_{2k} \\ . & . & \dots & . & . & . & \dots & . \\ a_{n1} & a_{n2} & \dots & a_{nn} & b_{n1} & b_{n2} & \dots & b_{nk} \end{bmatrix}$$

Nếu ma trận A không suy biến, ta có thể áp dụng các phép biến đổi sơ cấp để đưa ma trận này về dạng:

$$\begin{bmatrix} 1 & 0 & \dots & 0 & b'_{11} & b'_{12} & \dots & b'_{1k} \\ 0 & 1 & \dots & 0 & b'_{21} & b'_{22} & \dots & b'_{2k} \\ . & . & \dots & . & . & . & \dots & . \\ 0 & 0 & \dots & 1 & b'_{n1} & b'_{n2} & \dots & b'_{nk} \end{bmatrix}$$

Khi đó ta có

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1k} \\ x_{21} & x_{22} & \dots & x_{2k} \\ . & . & \dots & . \\ x_{n1} & x_{n2} & \dots & x_{nk} \end{bmatrix} = \begin{bmatrix} b'_{11} & b'_{12} & \dots & b'_{1k} \\ b'_{21} & b'_{22} & \dots & b'_{2k} \\ . & . & \dots & . \\ b'_{n1} & b'_{n2} & \dots & b'_{nk} \end{bmatrix};$$

Đặt $B' = \begin{bmatrix} b'_{11} & b'_{12} & \dots & b'_{1k} \\ b'_{21} & b'_{22} & \dots & b'_{2k} \\ . & . & \dots & . \\ b'_{n1} & b'_{n2} & \dots & b'_{nk} \end{bmatrix}$

Xét trường hợp đặc biệt B = E, ta có ma trận B' chính là ma trận nghịch đảo của ma trận A. Thật vậy, nếu X là nghiệm của (2.2) thì

$X = A^{-1}B$

Nếu B = E thì $X = A^{-1}$. Do đó việc tìm ma trận nghịch đảo của ma trận A tương đương với việc giải phương trình

$AX = E$

Ta có thể tóm tắt các bước cần thực hiện để tính ma trận đảo như sau:

- Viết thêm ma trận đơn vị E bên cạnh ma trận A

$$\begin{bmatrix} a_{11} & a_{12} & ... & a_{1n} & 1 & 0 & ... & 0 \\ a_{21} & a_{21} & ... & a_{2n} & 0 & 1 & ... & 0 \\ . & . & ... & . & . & . & ... & . \\ a_{n1} & a_{n2} & ... & a_{nn} & 0 & 0 & ... & 1 \end{bmatrix} \qquad (2.3)$$

- Áp dụng phép biến đổi sơ cấp lên các hàng của ma trận (2.3) cho đến khi ma trận có dạng

$$\begin{bmatrix} 1 & 0 & ... & 0 & c_{11} & c_{12} & ... & c_{1n} \\ 0 & 1 & ... & 0 & c_{21} & c_{22} & ... & c_{2n} \\ . & . & ... & . & . & . & ... & . \\ 0 & 0 & ... & 1 & c_{n1} & c_{n2} & ... & c_{nn} \end{bmatrix}$$

Khi đó ta có

$$A^{-1} = \begin{bmatrix} c_{11} & c_{12} & ... & c_{1n} \\ c_{21} & c_{22} & ... & c_{2n} \\ . & . & ... & . \\ c_{n1} & c_{n2} & ... & c_{nn} \end{bmatrix}$$

Chú ý. Trong quá trình biến đổi ta *có thể đổi các hàng* của ma trận. Điều này không ảnh hưởng đến kết quả thu được: Ma trận C vẫn là ma trận nghịch đảo của ma trận A ban đầu. Lý do là vì để tìm ma trận nghịch đảo ta chỉ cần xác định ma trận nghiệm. Ma trận nghiệm không bị thay đổi nếu ta đổi chỗ các hàng.

Chương trình minh họa.

Sau đây là đoạn chương trình chính thể hiện (mô tả) thuật toán tìm ma trận nghịch đảo

```
int daomtran(kmatran a,kmatran &ad,int n)
 {int i,j,k,h;double tmp,p;
  int n2=n*2;
  kmatran aa;
  for(i=1;i<=n;i++)
   for(j=1;j<=n;j++) aa[i][j]=a[i][j];
  //Them phan sau cua ma tran a de co dang ma tran don vi
  for(i=1;i<=n;i++)
   for(j=1;j<=n;j++) a[i][n+j]=0;//Cho phan ma tran vuong phia sau bang 0
  for(i=1;i<=n;i++) a[i][n+i]=1; //Duong cheo chinh phan phia sau bang 1
  //Vong lap cac buoc khu
  for(i=1;i<=n;i++)
   {//Tim hang co phan tu dau lon nhat
    h=i;
```

```
 for(k=i+1;k<=n;k++)
  if(fabs(a[k][i])> fabs(a[h][i])) h=k;
 if(a[h][i])==0) {cout<<"Ma tran suy bien";delay(1000);return false;}
 if(h!=i) //Doi hang i va hang h vi a[h][i] > a[i][i]
  {int j;double tmp;
   for(j=i;j<=n2;j++)
    {tmp=a[i][j];a[i][j]=a[h][j];a[h][j]=tmp;}
  }
 //chuyen he so a[i][i] = 1
 tmp=a[i][i];
 for(j=i;j<=n2;j++) a[i][j] = a[i][j]/tmp;
 //Bat tinh lai cac hang
 for(k=1;k<=n;k++)
  {if(k==i) continue;
   p=a[k][i];
   /*Vi ta biet a[k][i] =0 sau bien doi,
chi tinh tu a[k][i+1]*/
   for(j=i+1;j<=n2;j++) a[k][j]=a[k][j] - p*a[i][j];
  }
}
/*Dat cac gia tri khong o tren duong cheo chinh bang 0
 (phan nay khong can)*/
for(i=1;i<=n;i++)
 for(j=1;j<=n;j++) {if(i!=j) a[i][j]=0;}

//Ma tran dao la phan phia sau cua mang a
for(i=1;i<=n;i++)
 for(j=1;j<=n;j++) ad[i][j]=a[i][n+j];

 //Thu lai A x AD = C
kmatran c;
for(i=1;i<=n;i++)
 for(j=1;j<=n;j++)
  {c[i][j]=aa[i][1]*ad[1][j];
  for(k=2;k<=n;k++) c[i][j]+=aa[i][k]*ad[k][j];
  }
  return true;
}
```

2.2.3. Sự không ổn định của hệ phương trình đại số tuyến tính

a. Chuẩn của ma trận và vec tơ

Chuẩn của ma trận chữ nhật cấp m x n $A = (a_{ij})$ là một số thực không âm được ký hiệu là ||A|| thỏa mãn các điều kiện sau

(1) $||A|| \geq 0$ (với $||A|| = 0 \Leftrightarrow A = 0$)

(2) $||\alpha A|| = |\alpha| \, ||A||$, α là số thực bất kỳ.

(3) $||A + B|| \leq ||A|| + ||B||$

(4) $||A.B|| = ||A||.||B||$

Người ta thường dùng ba chuẩn sau:

Chuẩn cột: $$||A||_1 = \max_j \sum_{i=1}^{m} |a_{ij}|$$

Chuẩn Ơclit: $$||A||_2 = (\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2)^{1/2}$$

Chuẩn hàng: $$||A||_\infty = \max_i \sum_{j=1}^{n} |a_{ij}|$$

Ví dụ. Cho

$$A = \begin{bmatrix} 5 & -2 & 1 \\ 1 & 4 & 3 \\ 2 & -1 & 7 \end{bmatrix}$$

Ta tính được các chuẩn của A theo định nghĩa trên như sau:

$||A||_1 = \max(5+1+2, 2+4+1, 1+3+7) = \max(8, 7, 11) = 11$

$||A||_2 = (5^2 + 2^2 + 1 + 1 + 4^2 + 3^2 + 2^2 + 1 + 7^2)^{1/2} = 110^{1/2} = 10.5$

$||A||_\infty = \max(5+2+1, 1+4+3, 2+1+7) = \max(8, 8, 10) = 10$

Vec tơ là ma trận chỉ có một cột, do đó đối với vec tơ

$$x = \begin{pmatrix} x_1 \\ x_2 \\ . \\ x_n \end{pmatrix}$$

ta có 3 chuẩn sau

$$||x||_1 = \sum_{i=1}^{n} |x_i|$$

$$||x||_2 = (\sum_{i=1}^{n} x_i^2)^{1/2}$$

$$||x||_\infty = \max_i |x_i|$$

Ví dụ. Cho

$$x = \begin{pmatrix} 2 \\ -3 \\ 4 \\ 1 \\ 4 \end{pmatrix}$$

Ta có

$||x||_1 = 2 +3 +4 +1 +4 = 14$

$||x||_2 = (2^2+ 3^2+ 4^2+ 1 +4^2)^{1/2} = \sqrt{46}$

$||x||_\infty = \max(2,3,4,1,4) = 4$

Trong các phần tiếp theo chúng ta sẽ ký hiệu đơn giản là $||A||$ hoặc $||x||$ để chỉ chuẩn của ma trận và vec tơ. Nếu không có gì giải thích thêm thì cách ký hiệu này được hiểu là một trong ba chuẩn trên đây.

b. Sự không ổn định của hệ phương trình đại số tuyến tính

Trên đây ta đã tìm hiểu các phương pháp giải hệ phương trình đại số tuyến tính một cách trực tiếp. Nếu như mọi tính toán của ta là chính xác thì các phương pháp trên cho kết quả hoàn toàn chính xác. Tuy nhiên trong thực tế khi tính toán ta phải thường xuyên làm tròn các số, nghĩa là ta thường chỉ tính toán trên các số gần đúng mà thôi. Liệu cách làm tròn trong tính toán có làm ảnh hưởng nhiều đến kết quả cuối cùng không? Ví dụ sau đây cho thấy rằng có những hệ phương trình đại số tuyến tính rất "nhạy cảm" với sai số, nghĩa là sai số nhỏ khi tính toán có thể ảnh hưởng nghiêm trọng đến kết quả cuối cùng. Nói một cách hình tượng thì ta gặp tình huống "sai một li đi một dặm". Những hệ thống phương trình kiểu này được gọi là hệ phương trình không ổn định.

Ví dụ . Ta xét hệ phương trình sau:

$$2x_1 + x_2 = 2$$

$$2x_1 + 1.01x_2 = 2.01$$

Hệ này có nghiệm $x_1 = 0.5$, $x_2 = 1$.

Tuy nhiên hệ phương trình sau đây nhận được với chút ít thay đổi hệ số trong hệ trên

$$2x_1 + x_2 = 2$$

$$2.01x_1 + 1x_2 = 2.05$$

lại có nghiệm $x_1 = 5$, $x_2 = -8$, khác xa so với nghiệm trên đây.

2.2.4. Phương pháp lặp giải hệ phương trình tuyến tính

Các phương pháp trực tiếp giải hệ phương trình tuyến tính nói chung cần khoảng cn^3 phép tính, trong đó c là một hằng số và người ta ước lượng $c \approx 2/3$. Phương pháp khử Gauss như chúng ta vừa tìm hiểu chẳng hạn, là một phương pháp đúng, nghĩa là nếu các phép tính sơ cấp được thực hiện *đúng hoàn toàn* thì cuối cùng ta được *nghiệm đúng* của hệ. Tuy nhiên trong thực tế ta phải luôn luôn làm tròn khi thực hiện các phép tính, và như ta đã thấy ở trên, sai số tổng hợp đôi khi có thể sẽ khá lớn. Và chúng ta gặp một nghịch lý: về lý thuyết phương pháp cho kết quả chính xác

100%, nhưng khi thực hiện để áp dụng thực tế thì đôi khi kết quả lại khác xa so với kết quả lý thuyết. Vì những lý do trên đây, người ta đã tìm kiếm những phương pháp gần đúng để giải các bài toán, tức là ngay từ đầu người ta chấp nhận kết quả xấp xỉ, hay sự xấp xỉ đã nằm ngay trong mô hình. Khi thực hiện tính toán cụ thể chúng ta lại gặp sai số một lần nữa. Như vậy trong các phương pháp gần đúng thì sai số sẽ là tổng hợp của sai số mô hình và sai số tính toán. Một điều đáng ngạc nhiên là trong nhiều trường hợp phương pháp gần đúng lại cho kết quả tốt hơn phương pháp đúng. Thực ra điều này cũng không có gì khó hiểu, vì trong thực tế chúng ta cũng rất hay gặp những trường hợp một lần sai còn nặng nề trầm trọng hơn 2 lần hay thậm chí một số lần sai cộng lại.

a. Các bước chung trong phương pháp lặp

Giả sử ta cần giải phương trình F(x) = 0, trong đó F(x) là một hàm trên không gian định chuẩn nào đó và 0 được hiểu là phần tử 0 của không gian này. Ví dụ nếu không gian định chuẩn là R^n thì 0 là vectơ $(0,0,...,0)^T$. Ta biến đổi phương trình này về dạng tương đương x = G(x). Ta có thể phát biểu định lý sau:

Định lý. Giả sử y = G(x) là một hàm liên tục trên không gian định chuẩn nào đó và phép lặp $x_n=G(x_{n-1})$ n=1,2,... hội tụ tới x^* với xuất phát ban đầu x_0. Khi đó x^* là nghiệm của phương trình x = G(x), tức là ta có $x^* = G(x^*)$.

Chứng minh. Từ $x_n=G(x_{n-1})$, với lưu ý là hàm G(x) liên tục, ta có

$$\lim_{n->+\infty} x_n = \lim_{n->+\infty} G(x_{n-1}) = G(\lim_{n->+\infty} x_{n-1}) \Rightarrow x^* = G(x^*)$$

b. Phương pháp lặp đơn

Trở lại bài toán giải hệ phương trình tuyến tính

$$Ax = b \qquad (2.4)$$

Ta đưa (2.4) về dạng

$$x = Cx + d \qquad (2.5)$$

Trong đó ma trận C và vec tơ d được xây dựng từ A và b.

Để thực hiện phép lặp ta chọn một vec tơ ban đầu $x^{(0)}$, sau đó tính các $x^{(i)}$, i =1,2,... theo công thức lặp sau:

$$\begin{aligned} x^{(1)} &= Cx^{(0)} + d \\ x^{(2)} &= Cx^{(1)} + d \\ &\dots \\ x^{(k)} &= Cx^{(k-1)} + d \\ &\dots \end{aligned} \qquad (2.6)$$

Véc tơ $x^{(k)}$ được gọi là vec tơ lặp thứ k.

Ta có định lý sau:

Định lý. (Sự hội tụ của phương pháp)

a. Nếu phép lặp (2.6) hội tụ, tức là tồn tại x^* sao cho $x^* = \lim_{k->+\infty} x^{(k)}$

thì khi đó x^* là nghiệm của (2.5) (và như vậy cũng là nghiệm của (2.4))

b. Nếu ||C|| < 1 với một chuẩn nào đó, thì (2.6) hội tụ và sai số giữa nghiệm gần đúng $x^{(k)}$ (nghiệm gần đúng tại bước lặp thứ k) và nghiệm đúng x^* có thể đánh giá bằng các công thức sau:

$$||x^{(k)} - x^*|| \le \frac{||C||}{1-||C||} ||x^{(k)} - x^{(k-1)}|| \qquad (2.7)$$

hoặc

$$||x^{(k)} - x^*|| \le \frac{||C||^k}{1-||C||} ||x^{(1)} - x^{(0)}|| \qquad (2.8)$$

Nói chung theo phương pháp lặp đơn, điều kiện để phép lặp được hội tụ thì ||C|| < 1. Tuy nhiên trong thực tế thì ta chỉ có ma trận A. Một câu hỏi đặt ra là ma trận A phải thỏa mãn điều kiện gì để ta có thể đưa (2.4) về dạng (2.5) và áp dụng phương pháp lặp đơn?

Để phương pháp lặp hội tụ thì thường ma trận A phải thỏa mãn tính chéo trội của ma trận vuông.

Định nghĩa: (Tính chéo trội của một ma trận vuông): Ma trận A với các thành phần a_{ij} được gọi là ***có tính chéo trội***, nếu giá trị tuyệt đối của các phần tử nằm trên đường chéo chính lớn hơn tổng các giá trị tuyệt đối của các phần còn lại nằm cùng hàng, tức là

$$| a_{ii} | > \sum_{j-1, j\neq i}^{n} | a_{ij} |, \; i = 1, 2, \ldots, n. \qquad (2.9)$$

Sau đây sẽ giới thiệu 2 phương pháp lặp đơn Jacobi và Gaus-Seidel

c. Phương pháp lặp Jacobi

Với giả thiết ma trận A có tính chéo trội, khi đó các hệ số $a_{ii} \neq 0$, i = 1,2,...,n do đó ta có thể chia phương trình thứ i của hệ (2.1) cho a_{ii} và nhận được hệ tương tương

$$x_1 + \frac{a_{12}}{a_{11}} x_2 + \frac{a_{13}}{a_{11}} x_3 + \ldots + \frac{a_{1n}}{a_{11}} x_n = \frac{b_1}{a_{11}}$$

$$\frac{a_{21}}{a_{22}} x_1 + x_2 + \frac{a_{23}}{a_{22}} x_3 + \ldots + \frac{a_{2n}}{a_{22}} x_n = \frac{b_2}{a_{22}}$$

$$\ldots$$

$$\frac{a_{i1}}{a_{ii}} x_1 + \frac{a_{i2}}{a_{ii}} x_2 + \ldots + \frac{a_{i,i-1}}{a_{ii}} x_{i-1} + x_i + \frac{a_{i,i+1}}{a_{ii}} x_{i+1} \ldots + \frac{a_{in}}{a_{ii}} x_n = \frac{b_i}{a_{ii}}$$

$$\ldots$$

$$\frac{a_{n1}}{a_{nn}} x_1 + \frac{a_{n2}}{a_{nn}} x_2 + \ldots + \frac{a_{n,n-1}}{a_{nn}} x_{n-1} + x_n = \frac{b_n}{a_{nn}}$$

Từ đây ta có

$$x_1 = -\left(0.x_1 + \frac{a_{12}}{a_{11}} x_2 + \frac{a_{13}}{a_{11}} x_3 + \ldots + \frac{a_{1n}}{a_{11}} x_n\right) + \frac{b_1}{a_{11}}$$

$$x_2 = -\left(\frac{a_{21}}{a_{22}} x_1 + 0.x_2 + \frac{a_{23}}{a_{22}} x_3 + \ldots + \frac{a_{2n}}{a_{22}} x_n\right) + \frac{b_2}{a_{22}}$$

$$\ldots$$

$$x_i = -\left(\frac{a_{i1}}{a_{ii}}x_1 + \frac{a_{i2}}{a_{ii}}x_2 + \ldots + \frac{a_{i,i-1}}{a_{ii}}x_{i-1} + 0.x_i + \frac{a_{i,i+1}}{a_{ii}}x_{i+1} \ldots + \frac{a_{in}}{a_{ii}}x_n\right) + \frac{b_i}{a_{ii}}$$

. . .

$$x_n = -\left(\frac{a_{n1}}{a_{nn}}x_1 + \frac{a_{n2}}{a_{nn}}x_2 + \ldots + \frac{a_{n,n-1}}{a_{nn}}x_{n-1} + 0.x_n\right) + \frac{b_n}{a_{nn}}$$

Khi đó ma trận C, vectơ d là:

$$C = -\begin{bmatrix} 0 & \frac{a_{12}}{a_{11}} & \ldots & \frac{a_{1n}}{a_{11}} \\ \frac{a_{21}}{a_{22}} & 0 & \ldots & \frac{a_{2n}}{a_{22}} \\ . & . & \ldots & . \\ \frac{a_{n1}}{a_{nn}} & \frac{a_{n2}}{a_{nn}} & \ldots & 0 \end{bmatrix}, \quad d = \begin{pmatrix} \frac{b_1}{a_{11}} \\ \frac{b_2}{a_{22}} \\ . \\ \frac{b_n}{a_{nn}} \end{pmatrix}$$

(Đến đây ta đã đưa hệ (2.4) về dạng (2.5) và dễ thấy rằng ma trận C thỏa mãn điều kiện lặp đơn, tức là $\|C\|_\infty < 1$.).

Vậy đến đây ta tiếp tục áp dụng phương pháp lặp (2.6) để tính nghiệm ở các bước lặp như sau:

Với vec tơ $x^{(0)}$ cho trước bất kỳ, ví dụ $x^{(0)} = \theta$ (vec tơ 0) ta có thể tính các vec tơ $x^{(k)}$ tại bước lặp k bằng công thức $x^{(k)} = C\,x^{(k-1)} + d$, k =1, 2, ...

Cụ thể hơn, nếu $x^{(k)} = (x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)})$ thì ta có

$$\begin{pmatrix} x_1^{(k)} \\ x_2^{(k)} \\ . \\ x_n^{(k)} \end{pmatrix} = -\begin{bmatrix} 0 & \frac{a_{12}}{a_{11}} & \ldots & \frac{a_{1n}}{a_{11}} \\ \frac{a_{21}}{a_{22}} & 0 & \ldots & \frac{a_{2n}}{a_{22}} \\ . & . & \ldots & . \\ \frac{a_{n1}}{a_{nn}} & \frac{a_{n2}}{a_{nn}} & \ldots & 0 \end{bmatrix} \begin{pmatrix} x_1^{(k-1)} \\ x_2^{(k-1)} \\ . \\ x_n^{(k-1)} \end{pmatrix} + \begin{pmatrix} \frac{b_1}{a_{11}} \\ \frac{b_2}{a_{22}} \\ . \\ \frac{b_n}{a_{nn}} \end{pmatrix}$$

Với từng thành phần $x_i^{(k)}$ ta có

$$x_i^{(k)} = -\sum_{j-1, j\neq i}^{n} \frac{a_{ij}}{a_{ii}} x_j^{(k-1)} + \frac{b_i}{a_{ii}} = \frac{1}{a_{ii}}\left(b_i - \sum_{j-1, j\neq i}^{n} a_{ij}\, x_j^{(k-1)}\right) \qquad (2.10)$$

i = 1, 2, . . ., n, k = 1,2,...

Điều kiện hội tụ, đánh gái sai số của phương pháp lặp Jacobi cũng giống với phương pháp lặp đơn.

Ví dụ. Dùng phương pháp lặp Jacobi tìm nghiệm gần đúng của hệ phương trình:

$$\begin{cases} 4x_1 + 0.24x_2 - 0.08x_3 = 8 \\ 0.09x_1 + 3x_2 - 0.15x_3 = 9 \\ 0.04x_1 - 0.08x_2 + 4x_3 = 20 \end{cases}$$

Giải. (1).Có thể thấy rằng ma trận các hệ số của hệ phương trình trên đây thỏa mãn tính chéo trội, do đó ta có thể biến đổi hệ này để áp dụng phương pháp lặp Jacobi. Chia hai vế phương trình đầu tiên cho 4, hai vế phương trình thứ hai cho 3 và hai vế của phương trình thứ ba cho 4 rồi biến đổi thích hợp ta nhận được:

$$x_1 = 2 - 0.06x_2 + 0.02x_3$$
$$x_2 = 3 - 0.03x_1 + 0.05x_3$$
$$x_3 = 5 - 0.01x_1 + 0.02x_2$$

hay

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{bmatrix} 0 & -0.06 & 0.02 \\ -0.03 & 0 & 0.05 \\ -0.01 & 0.02 & 0 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} 2 \\ 3 \\ 5 \end{pmatrix} = Cx + d$$

$||C||_\infty = \max(0 + 0.06 + 0.02, 0.03 + 0 + 0.05, 0.01 + 0.02 + 0) =$

$= \max(0.08, 0.08, 0.03) = 0.08 < 1$

(2) Chọn $x^{(0)} = (2,3,5)^T$, rồi tính $x^{(1)}, x^{(2)}, \ldots$ theo công thức (2.10) với lưu ý $a_{ii} = 1$ ta được bảng kết quả sau:

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$
0	2	3	5
1	1.92	3.19	5.04
2	1.9094	3.1944	5.0446
3	1.909228	3.194948	5.044794

(3) Xem $x^{(3)}$ là nghiệm gần đúng cần tìm, ta có thể đánh giá sai $x^{(3)}$ với nghiệm đúng x^* theo (2.10) như sau: $||x^{(3)} - x^*|| \le \dfrac{||C||}{1 - ||C||} ||x^{(3)} - x^{(2)}||$

$$||x^{(3)} - x^{(2)}||_\infty = \max_i |x_i^{(3)} - x_i^{(2)}| = \max(0.000172, 0.000548, 0.000194) = 0.000548$$

Như vậy

$$||x^{(3)} - x^*||_\infty \le \frac{0.08}{1 - 0.08} 0.000548 = 0.0000476 \approx 0.00005$$

d. Phương pháp lặp Gauss - Seidel

Với giả thiết ma trận A có tính chéo trội. Từ công thức (2.10) ta thấy rằng phần tử thứ i của vec tơ nghiệm tại bước k được tính qua các phần tử ở các vị trí khác i trong bước k-1. Phương pháp Gauss-Seidel cải tiến phương pháp Jacobi bằng cách dùng ngay những kết quả vừa tính được cho các thành phần của nghiệm tại bước k để tính các thành phần khác của bước k, chỉ có những thành phần nào chưa được tính thì mới lấy ở bước k-1. Cụ thể hơn ta có tại các bước:

(1) Giá trị $x_1^{(1)}$ được tính qua các giá trị $x_2^{(0)}, x_3^{(0)}, \ldots x_n^{(0)}$

Giá trị $x_2^{(1)}$ được tính qua các giá trị $x_1^{(1)}, x_3^{(0)}, \ldots x_n^{(0)}$

Giá trị $x_3^{(1)}$ được tính qua các giá trị $x_1^{(1)}, x_2^{(1)}, x_4^{(0)}, \ldots x_n^{(0)}$

. . .

(h) Giá trị $x_1^{(h)}$ được tính qua các giá trị $x_2^{(h-1)}, x_3^{(h-1)}, \ldots x_n^{(h-1)}$

Giá trị $x_2^{(h)}$ được tính qua các giá trị $x_1^{(h)}, x_3^{(h-1)}, \ldots x_n^{(h-1)}$

Giá trị $x_3^{(h)}$ được tính qua các giá trị $x_1^{(h)}, x_2^{(h)}, x_4^{(h-1)}, \ldots x_n^{(h-1)}$

. . .

Với vec tơ $x^{(0)}$ cho trước bất kỳ, ví dụ $x^{(0)} = \theta$ (vec tơ 0) ta có thể tính các vec tơ $x^{(k)}$ tại bước lặp k bằng công thức

$$x_i^{(k)} = \frac{1}{a_{ii}}(b_i - (\sum_{j=1}^{i-1} a_{ij}x_j^{(k)} + \sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)})) \qquad (2.11)$$

$i = 1, 2, \ldots, n, k = 1,2,...$

Trong công thức (2.11) chúng ta có thể không dùng chỉ số trên để chỉ ra rằng chúng ta chỉ dùng một mảng là vec tơ có n thành phần để lưu trữ nghiệm. Giá trị nào vừa được tính toán thì được lưu trữ ngay vào vị trí cũ và được dùng ngay trong công thức tính các giá trị khác.

$$x_i = \frac{1}{a_{ii}}(b_i - \sum_{j=1, j\neq i}^{n} a_{ij}\, x_j)$$

$i = 1, 2, \ldots, n, k = 1,2,...$

Sự hội tụ của phương pháp Gause-Seidel

Điều kiện hội tụ của phương pháp lặp Gause- Seidel cũng giống với phương pháp lặp đơn. Như ta sẽ thấy trong ví dụ trong phần sau, phương pháp Gause- Seidel nói chung hội tụ nhanh hơn phương pháp lặp đơn.

Ta có thể sử dụng các công thức sau để đánh giá sai số của phương pháp lặp Gause-Seidel:

Gọi x^* là nghiệm đúng của hệ phương trình và gọi

$$p_i = \sum_{j=1}^{i-1} |c_{ij}|,\ q_i = \sum_{j=i}^{n} |c_{ij}|\ ,\ \mu = \max_i \frac{q_i}{1-p_i}$$

Khi đó ta có

$$\|x^{(k)} - x^*\| \le \frac{\mu}{1-\mu}\|x^{(k)} - x^{(k-1)}\| \qquad (2.12)$$

hoặc

$$\|x^{(k)} - x^*\| \le \frac{\mu^k}{1-\mu}\|x^{(1)} - x^{(0)}\| \qquad (2.13)$$

Ví dụ. Dùng phương pháp lặp ***Gause-Seidel*** tìm nghiệm gần đúng của hệ phương trình:

$$\begin{cases} 4x_1 + 0.24x_2 - 0.08x_3 = 8 \\ 0.09x_1 + 3x_2 - 0.15x_3 = 9 \\ 0.04x_1 - 0.08x_2 + 4x_3 = 20 \end{cases}$$

Giải. (1).Có thể thấy rằng ma trận các hệ số của hệ phương trình trên đây thỏa mãn tính chéo trội, do đó ta có thể biến đổi hệ này để áp dụng phương pháp lặp Jacobi. Chia hai vế phương trình đầu tiên cho 4, hai vế phương trình thứ hai cho 3 và hai vế của phương trình thứ ba cho 4 rồi biến đổi thích hợp ta nhận được:

$$\begin{cases} x_1 = 2 - 0.06x_2 + 0.02x_3 \\ x_2 = 3 - 0.03x_1 + 0.05x_3 \\ x_3 = 5 - 0.01x_1 + 0.02x_2 \end{cases}$$

hay

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{bmatrix} 0 & -0.06 & 0.02 \\ -0.03 & 0 & 0.05 \\ -0.01 & 0.02 & 0 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} 2 \\ 3 \\ 5 \end{pmatrix} = Cx + d$$

$\|C\|_\infty$ = max(0 + 0.06 + 0.02, 0.03 + 0 + 0.05, 0.01 + 0.02 + 0) =

= max(0.08,0.08,0.03) = 0.08 <1

(2) Chọn $x^{(0)} = (2,3,5)^T$, rồi tính $x^{(1)}$, $x^{(2)}$,... theo công thức (2.11) với lưu ý a_{ii} =1 ta được bảng kết quả sau:

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$
0	2	3	5
1	1.92	3.1924	5.044648
2	1.9093489	3.194952	5.0448056
3	1.909199	3.1949643	5.0448073

Xem $x^{(3)}$ là nghiệm gần đúng cần tìm, ta có thể đánh giá sai số phạm phải của $x^{(3)}$ theo(2.12): $\|x^{(k)} - x^*\| \le \frac{\mu}{1-\mu}\|x^{(k)} - x^{(k-1)}\|$

Trong đó:

$\|x^{(3)} - x^{(2)}\|_\infty = \max_i |x_i^{(3)} - x_i^{(2)}|$ = max(0.0001499,0.000123,0.0000017) = 0.0001499

$$\mu = \max_i \frac{q_i}{1-p_i} = \max(0.08,0.0515463,0) = 0.08$$

Như vậy

$$\|x^{(3)} - x^*\|_\infty \le \frac{\mu}{1-\mu}\|x^{(k)} - x^{(k-1)}\| \le \frac{0.08}{1-0.08}\ 0.00001499 \approx 0.000013$$

Thuật toán Jacobi cũng tương tự như thuật toán Gauss-Seidel, nhưng thuật toán Gauss - Seidel có tốc độ hội tụ nhanh hơn.

Chương trình minh họa.

Sau đây là đoạn chương trình chính thể hiện (mô tả) thuật toán *lặp Gauss - Seidel*

```
/*Giai he phuong trinh tuyen tinh dung lap Gauss-Seidel, ma tran vuong n,
  cac phan tu cot thu n+1 la vecto b*/
//================================================
double kcach(double *x,double *y,int n)
 {double tmp=0;
  for(int i=1;i<=n;i++) tmp=tmp+fabs(x[i]-y[i]);
  return tmp;
 }
//================================================
int cheotroi(kmatran a, int n)
 {double tmp;int i,j;
  for(i=1;i<=n;i++)
   {tmp=0;
    for(j=1;j<=n;j++) {if(j!=i) tmp=tmp+fabs(a[i][j]);}
    if(fabs(a[i][i])<=tmp) return false;
   }
  return true;
 }
//================================================
/*Giai he phuong trinh tuyen tinh bang phep lap Gauss-Seidel.
  Tra ve true neu co nghiem */
int gseidel(kmatran aa,double *x,int n)
{int h,i,j,k;double tmp;kvecto z;kmatran a;
 int n1=n+1;
 for(i=1;i<=n;i++)
 for(j=1;j<=n1;j++) a[i][j]=aa[i][j];
 if(!cheotroi(a,n))
  {cout<<endl<<"Khong phai cheo troi";delay(1000);return false;}
 for(i=1;i<=n;i++)  //chuyen ve dang he so a[i][i] == 1
  {tmp=a[i][i];
   for(j=1;j<=n1;j++) a[i][j] = a[i][j]/tmp;
  }
 //Vong lap cac buoc khu
```

```
for(i=1;i<=n;i++) {x[i]=0;z[i]=0;}
k=1;
while(true)
 {for(i=1;i<=n;i++)
  {tmp=0;
   for(j=1;j<=n;j++) if(j!=i) tmp+=a[i][j]*x[j];
   x[i] = a[i][n+1]-tmp;
  }
  k++;
  if(kcach(x,z,n)<epsi) break;
  if(k>kmax)
   {cout<<endl<<"Phep lap chua hoi tu";delay(1000);return(false);}
  //Gan z = x va chuan bi sang vong lap tinh x
  for(i=1;i<=n;i++) z[i]=x[i];
 }
//Thu lai
kvecto bb;
for(i=1;i<=n;i++)
 {bb[i]=aa[i][1]*x[1];
  for(j=2;j<=n;j++) bb[i]+=aa[i][j]*x[j];
 }
//Dua ket qua vao tep ketqua
return true;
)
```

2.3. BÀI TẬP

Bài 1. Tính và kiểm tra bằng chương trình định thức của ma trận

$$A = \begin{bmatrix} 1 & 3 & 1 \\ 2 & 4 & 6 \\ 7 & 6 & 11 \end{bmatrix}$$

Bài 2. Tìm và kiểm tra bằng chương trình nghịch đảo của ma trận

$$A = \begin{bmatrix} 2 & 3 & 1 \\ -1 & 2 & -1 \\ 3 & 0 & 2 \end{bmatrix}$$

Bài 3. Tìm nghiệm hệ phương trình

$$\begin{cases} 2x_1 + 3x_2 + x_3 = 11 \\ -x_1 + 2x_2 - x_3 = 0 \\ 3x_1 + 2x_3 = 9 \end{cases}$$

Bằng phương pháp khử Gauss và Jordan. Kiểm tra bằng chương trình.

Bài 4. Giải bằng các phương pháp khử Gauss, khử Gauss-Jordan, phương lặp Jacobi và lặp Gauss-Seidel (nếu thỏa mãn điều kiện) hệ phương trình sau:

$$A = \begin{bmatrix} 17 & 65 & -13 & 50 \\ 12 & 16 & 37 & 28 \\ 56 & 23 & 11 & -19. \\ 3 & -5 & 47 & 10 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 84 \\ 25 \\ 36 \\ 18 \end{pmatrix}$$

Kiểm tra trên máy tính và thông báo về khả năng giải được hay không các phương pháp trên.

Bài 5. Giải bằng các phương pháp lặp hệ phương trình sau:

$$\begin{cases} 10x_1 + 2x_2 + x_3 = 9 \\ 2x_1 + 20x_2 - 2x_3 = -44 \\ -2x_1 + 3x_2 + 10x_3 = 22 \end{cases}$$

TÓM TẮT NỘI DUNG CHƯƠNG 2

Trong chương này sinh viên cần nắm vững ít nhất là các vấn đề sau:

1. Phương pháp trực tiếp giải hệ phương trình tuyến tính

a.Phương pháp khử Gauss

Phương pháp khử Gauss dùng cách khử dần các ẩn để đưa hệ phương trình đã cho về một dạng tam giác trên rồi giải hệ tam giác này từ giới lên trên, không phải tính một định thức nào.

b. Phương pháp khử Gauss-Jordan

Phương pháp khử Gauss-Jordan dùng cách khử dần các ẩn để đưa hệ phương trình đã cho về một dạng ma trận đường chéo rồi giải hệ phương trình này, không phải tính một định thức nào.

2. Phương pháp lặp giải hệ phương trình tuyến tính

a. Phương pháp lặp đơn

- Giả sử phải tìm nghiệm gần đúng của hệ phương trình tuyến tính (2.1) có dạng Ax=b. Đối với phương pháp lặp đơn, nói chung chúng ta phải đưa hệ (2.1) về dạng x=Cx+d.Trong đó ma trận C và vec tơ d được xây dựng từ A và b. Ma trận phải thoả mãn điều kiện ||C||<1.

Để thực hiện phép lặp ta chọn một vec tơ ban đầu $x^{(0)}$, sau đó tính các $x^{(i)}$, i =1,2,... theo công thức lặp sau: $x^{(i)} = Cx^{(i-1)} + d$ cho tới khi nào thảo mãn điều kiện dừng.

- Sai số của phương pháp:

$$||x^{(k)} - x^{*}|| \le \frac{||C||}{1-||C||} ||x^{(k)} - x^{(k-1)}||$$

hoặc

$$||x^{(k)} - x^{*}|| \le \frac{||C||^{k}}{1-||C||} ||x^{(1)} - x^{(0)}||$$

b. Phương pháp lặp Jacobi

- Giả thiết ma trận A có tính chéo trội. Phương pháp lặp Jacobi sẽ có các bước lặp như :

Với vec tơ $x^{(0)}$ cho trước bất kỳ, ví dụ $x^{(0)} = \theta$ (vec tơ 0) ta có thể tính các vec tơ $x^{(k)}$ tại bước lặp k bằng công thức $x^{(k)} = C\,x^{(k-1)} + d$, k =1, 2, ...

Cụ thể hơn, nếu $x^{(k)} = (x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)})$ thì ta có

$$\begin{pmatrix} x_1^{(k)} \\ x_2^{(k)} \\ . \\ x_n^{(k)} \end{pmatrix} = - \begin{bmatrix} 0 & \frac{a_{12}}{a_{11}} & \ldots & \frac{a_{1n}}{a_{11}} \\ \frac{a_{21}}{a_{22}} & 0 & \ldots & \frac{a_{2n}}{a_{22}} \\ . & . & \ldots & . \\ \frac{a_{n1}}{a_{nn}} & \frac{a_{n2}}{a_{nn}} & \ldots & 0 \end{bmatrix} \begin{pmatrix} x_1^{(k-1)} \\ x_2^{(k-1)} \\ . \\ x_n^{(k-1)} \end{pmatrix} + \begin{pmatrix} \frac{b_1}{a_{11}} \\ \frac{b_2}{a_{22}} \\ . \\ \frac{b_n}{a_{nn}} \end{pmatrix}$$

Với từng thành phần $x_i^{(k)}$ ta có

$$x_i^{(k)} = - \sum_{j-1, j\neq i}^{n} \frac{a_{ij}}{a_{ii}} x_j^{(k-1)} + \frac{b_i}{a_{ii}} = \frac{1}{a_{ii}} (b_i - \sum_{j-1, j\neq i}^{n} a_{ij} x_j^{(k-1)})$$

$i = 1, 2, \ldots, n, k = 1,2,...$

- Điều kiện hội tụ, đánh gái sai số của phương pháp lặp Jacobi cũng giống với phương pháp lặp đơn.

c.Phương pháp lặp Gauss – Seidel

- Giả thiết ma trận A có tính chéo trội. Phương pháp lặp ***Gauss - Seidel*** sẽ có các bước lặp như sau:

Với vec tơ $x^{(0)}$ cho trước bất kỳ, ví dụ $x^{(0)} = \theta$ (vec tơ 0) ta có thể tính các vec tơ $x^{(k)}$ tại bước lặp k bằng công thức :

$$x_i^{(k)} = \frac{1}{a_{ii}} (b_i - (\sum_{j-1}^{i-1} a_{ij} x_j^{(k)} + \sum_{j-i+1}^{n} a_{ij} x_j^{(k-1)}))$$

$i = 1, 2, \ldots, n, k = 1,2,...$

- Đánh giá sai số:

$$p_i = \sum_{j=1}^{i-1} |c_{ij}|, \; q_i = \sum_{j=i}^{n} |c_{ij}| \;, \; \mu = \max_i \frac{q_i}{1 - p_i}$$

Khi đó ta có:

$$||x^{(k)} - x^*|| \leq \frac{\mu}{1-\mu} ||x^{(k)} - x^{(k-1)}||$$

hoặc

$$||x^{(k)} - x^*|| \leq \frac{\mu^k}{1-\mu} ||x^{(1)} - x^{(0)}||$$

CHƯƠNG 3
PHÉP NỘI SUY VÀ HỒI QUY

MỤC ĐÍCH, YÊU CẦU

Sau khi học xong chương 3, yêu cầu sinh viên:

1. Hiểu được thế nào là bài toán nội suy và hồi quy.

2. Nắm được các phương pháp nội suy đa thức, biết cách tìm các đa thức nội suy theo các phương pháp đó.

3. Biết được khớp đường cong - Nội suy Spline là gì?

4. Nắm và giải được các bài toán bằng phương pháp bình phương tối thiểu

5. Biết cách đánh giá sai số của từng phương pháp.

3.1. MỞ ĐẦU

Thông thường trong một số lĩnh vực như kinh tế chẳng hạn, các đại lượng khảo sát thường không được cho dưới dạng hàm liên tục, mà là bảng các giá trị rời rạc. Các phương pháp giải tích toán học thường tính toán với các hàm cho bởi các công thức, do đó không thể áp dụng trực tiếp để nghiên cứu các hàm cho dưới dạng rời rạc như thế này. Cũng có khi ta biết rằng đại lượng y là một hàm của đại lượng x, tức là $y = f(x)$, nhưng ta không biết biểu thức hàm f(x) mà chỉ biết một số giá trị y_i tương ứng với các giá trị của x tại các điểm x_i như trong bảng sau:

x	x_0	x_1	x_2	. . .	x_{n-1}	x_n
y	y_0	y_1	y_2	. . .	y_{n-1}	y_n

Thông thường thì $x_0 < x_1 < x_2 < \ldots < x_n$ và các điểm này có thể phân bố cách đều hoặc không. Mặc dầu ta chỉ biết các giá trị của y tại các điểm mốc x_i, nhưng trong nhiều trường hợp ta cần tính toán với các giá trị y tại các vị trí khác của x. Một câu hỏi đặt ra là: cho một điểm x không thuộc các điểm x_i cho ở trên, làm thế nào chúng ta có thể tính được giá trị y tương ứng với nó, sao cho chúng ta có thể tận dụng tối đa các thông tin đã có?

Bài toán nội suy là bài toán tìm giá trị gần đúng của y tại các điểm nằm giữa các giá trị x không có trong bảng trên. Nếu cần tìm các giá trị gần đúng của y tại các điểm x nằm ngoài khoảng $[x_0,x_n]$ thì bài toán được gọi là bài toán ngoại suy. Một bộ n+1 cặp các giá trị đã biết của x và y: (x_0,y_0), (x_1,y_1), . . . ,(x_n,y_n) được gọi là một mẫu quan sát, còn $x_0, x_1, \ldots, x_n$ được gọi là các điểm quan sát và $y_0, y_1, \ldots, y_n$ là các kết quả quan sát.

Vì bài toán của chúng ta không chỉ giải quyết với một giá trị x cụ thể, mà là cả một miềm giá trị nào đó của x. Do đó câu hỏi trên cũng tương đương với vấn đề sau: hãy tìm một hàm g(x) sao cho miền giá trị của nó chứa các điểm $(x_0, x_1, ..., x_n)$ và hàm này xấp xỉ tốt nhất tập số liệu đã có là các cặp (x_0,y_0), (x_1,y_1), ..., (x_n,y_n) theo một nghĩa nào đó. Chúng ta thấy ngay là tập số liệu là hữu hạn, còn tập các giá trị cần ước lượng là vô hạn, nên sẽ có vô số hàm g(x) nếu chúng ta không đưa ra một số ràng buộc nào đó về g(x). Điều đầu tiên chúng ta quan tâm là nên chọn dạng hàm g(x) như thế nào.

Một cách tự nhiên, ta có thể đặt điều kiện về hàm g(x) như sau:

- $g(x_i)$ $i = 0,1,2,...,n$ gần các điểm y_i nhất theo một nghĩa nào đó.
- g(x) là duy nhất theo một số điều kiện nào đó.
- Hàm g(x) liên tục, không có điểm gấp khúc và ít thay đổi trong từng đoạn $[x_i,x_{i+1}]$.

Các định lý về xấp xỉ sau đây của Weierstrass sẽ cho chúng ta gợi ý về dạng hàm của g(x).

Định lý Weierstrass 1 về xấp xỉ hàm.

Cho f (x) là một hàm thực liên tục xác định trên khoảng [a,b]. Khi đó với mọi $\varepsilon>0$ tồn tại một đa thức p(x) bậc m với các hệ số thực sao cho với mọi giá trị $x\in[a,b]$ ta có $|f(x) - p(x)|<\varepsilon$.

Định lý Weierstrass 2 về xấp xỉ hàm.

Cho f (x) là một hàm thực liên tục xác định trên khoảng $[-\pi,\pi]$ và $f(-\pi) = f(\pi)$. Khi đó với mọi $\varepsilon>0$ tồn tại một đa thức lượng giác

$$q_m(x) = \frac{a_0}{2} + \sum_{j=1}^{m} [a_j \cos(jx) + b_j \sin(jx)]$$

với các hệ số thực sao cho với mọi giá trị $x\in[-\pi,\pi]$ ta có $|f(x) - q(x)|<\varepsilon$.

Từ các định lý trên đây ta thấy rằng chọn ***đa thức*** là thích hợp cho dạng hàm g(x). Đa thức là hàm quen thuộc và ta đã biết nhiều tính chất của nó.

Người ta thường dùng các phương pháp xấp xỉ sau để xác định đa thức p(x):

1. Nếu ta biết rằng các cặp giá trị (x_0,y_0), (x_1,y_1), ..., (x_n,y_n) là thể hiện của một hàm f(x) nào đó, tức là ta biết rằng $y=f(x)$ và như vậy tại các điểm x_i, $i=0,1,...,n$ $y_i = f(x_i)$. Trong trường hợp này ta đòi hỏi đa thức p(x) phải đi qua các điểm (x_i,y_i), $i=0,1,...,n$.

   Bài toán nội suy bây giờ có thể phát biểu cụ thể hơn như sau:

   Cho một mẫu quan sát gồm n+1 cặp các giá trị đã biết của x và y : (x_0,y_0), (x_1,y_1), . . . ,(x_n,y_n) . Hãy xây dựng một đa thức bậc $m \le n$

   $$p_m(x) = a_0 + a_1x^1 + . . . a_{m-1}x^{m-1} + a_mx^m \qquad (3.1)$$

   sao cho $p_m(x_i) = y_i$, $i = 0, 1,..., n$ (3.2)

   Người ta gọi bài toán trên đây là bài toán ***nội suy đa thức***, và đa thức $p_m(x)$ được gọi là ***đa thức nội suy***.

   Trong một số ứng dụng vật lý ta gặp các hiện tượng có tính chất tuần hoàn. Khi đó đa thức lượng giác tỏ ra thích hợp hơn trong bài toán nội suy. Và trong bài toán trên đa thức $p_m(x)$ được thay bằng đa thức lượng giác

$$q_m(x) = \frac{a_0}{2} + \sum_{j=1}^{m} [a_j \cos(jx) + b_j \sin(jx)]$$

2. Nội suy trong trường hợp số đo không hoàn toàn chính xác:

Trong thực tế các giá trị y_i tại các điểm quan sát lại thường chỉ là các giá trị gần đúng của các giá trị thật. Nói cách khác thực ra ta chỉ có $y_i \approx f(x_i)$ mà thôi. Trong trường hợp này nếu ta áp đặt điều kiện về đa thức nội suy phải thỏa mãn $p_m(x_i) = y_i$ thì không hợp lý. Thay vì tìm một đa thức thỏa mãn điều kiện này, ta tìm đa thức $p_m(x) = a_0 + a_1x^1 + \ldots a_{m-1}x^{m-1} + a_mx^m$, tức là xác định các hệ số $a_0, a_1, \ldots, a_m$ sao cho tổng bình phương sai số là bé nhất, tức là

$$e = \sum_{i=0}^{n} (y_i - \sum_{j=0}^{m} a_j x_i^j)^2$$

là bé nhất. Phương pháp nội suy theo tiêu chuẩn này được gọi là *phương pháp bình phương bé nhất* hay là *phương pháp bình phương cực tiểu*.

Ngoài hai phương pháp thông dụng trên, người ta còn dùng phương pháp xấp xỉ Csebisev dựa trên tiêu chuẩn:

$$\max_{0 \le i \le n} |y_i - p(x_i)|$$

cực tiểu.

3.2. NỘI SUY ĐA THỨC

3.2.1. Sự duy nhất của đa thức nội suy

Ta có định lý sau đây:

Định lý. Có duy nhất một đa thức có bậc không quá n và đi qua n+1 điểm cho trước $(x_0,y_0), (x_1,y_1), \ldots, (x_n,y_n)$.

Chứng minh. Ta xét đa thức có dạng (3.1) trên đây và thỏa mãn (3.2). Kết hợp (3.1) và (3.2) ta có

$$\begin{pmatrix} y_0 \\ y_1 \\ . \\ y_n \end{pmatrix} = \begin{bmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ 1 & x_1 & x_1^2 & \ldots & x_1^n \\ . & . & . & \ldots & . \\ 1 & x_n & x_n^2 & x_n^2 & x_n^n \end{bmatrix} \begin{pmatrix} a_0 \\ a_1 \\ . \\ a_n \end{pmatrix} \qquad (3.3)$$

Hay có thể biểu diễn gọn hơn dưới dạng ma trận

$$Y = V\,a$$

Trong đó

$$V = \begin{bmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ 1 & x_1 & x_1^2 & \ldots & x_1^n \\ . & . & . & \ldots & . \\ 1 & x_n & x_n^2 & x_n^2 & x_n^n \end{bmatrix}$$

chính là ma trận Vandermon, ta có

$$\det V = \prod_{0 \le i < j \le n} (x_j - x_i)$$

Vì ta đã giả thiết các điểm x_i và x_j khác nhau, do đó ma trận này khác 0 nên hệ phương trình (3.3) có nghiệm duy nhất cho các a_i, và như vậy đa thức $p_n(x)$ được xác định duy nhất. (Nếu khi giải phương trình (3.3) mà ta nhận được $a_n \neq 0$ thì đa thức này có bậc là n).

3.2.2. Tính giá trị đa thức bằng phương pháp Horner

Trong bài toán nội suy đa thức ta sẽ phải thường xuyên tính giá trị của đa thức $p_m(x)= a_0 + a_1x^1 + \ldots a_{m-1}x^{m-1} + a_mx^m$ tại điểm x. Nếu tính trực tiếp ta phải thực hiện khá nhiều phép tính, và khi tính các giá trị mũ của x có thể ta gặp các giá trị lớn, cho dù trong thực tế các thành phần của đa thức triệt tiêu lẫn nhau và giá trị của đa thức không lớn. Horner đưa ra cách tính sau loại trừ được các nhược điểm trên.

Ta viết lại đa thức $p_m(x)$ dưới một dạng khác:

$$p_m(x) = a_mx^m + a_{m-1}x^{m-1} + \ldots + a_1x^1 + a_0 = (\ldots((a_mx + a_{m-1})x + a_{m-2})x+\ldots+a_1)x+a_0$$

Từ đây ta có cách tính $p_m(x)$ trên máy tính như sau:

Đặt $P_m = a_m$

$P_{m-1} = P_mx+a_{m-1}$

...

$P_i = P_{i-1}x + a_i$

Khi tính toán ta không cần lưu trữ tất c các giá trị của Pi, mà chỉ cần lưu trữ các giá trị của Pi trong một vị trí bộ nhớ. Thuật toán trên trở thành:

Đặt $P = a_m$

Cho i chạy từ m-1 đến 0, tức là i=m-1,m-2,...,0

Đặt $P = Px + a_i$

P cuối cùng tính được chính là giá trị của đa thức tại x.

3.2.3. Sai số của đa thức nội suy

Định lý Rolle: Cho f(x) là hàm số thực liên tục trên khoảng đóng [a,b] và khả vi trên khoảng mở (a,b) và f(a) = f(b). Khi đó tồn tại điểm $\xi \in (a,b)$ sao cho $f'(\xi) = 0$.

Định lý: Giả sử hàm f(x) có đạo hàm liên tục đến cấp n+1 trên đoạn [a,b] và $p_m(x)$ là đa thức nội suy, tức là:$p_m(x_i) = f(x_i) = y_i$, i = 0, 1,..., n. Với các mốc nội suy là $a = x_0 < x_1 <\ldots < x_n = b$.

Đặt $\omega_{n+1}(x) = \prod_{i=0}^{n} (x - x_i)$ và $R(x) = f(x) - p_m(x)$

Khi đó với $\forall x \in [a,b]$ tồn tại $\eta \in [a,b]$ (phụ thuộc vào x) sao cho

$$R(x) = \frac{f^{(n+1)}(\eta)}{(n+1)!}\omega_{n+1}(x) \qquad (3.4)$$

Hệ quả. Gọi $M = \sup_{a \le x \le b} |f^{(n+1)}(x)|$ khi đó ta có

$$|R(x)| \le |f(x) - p_m(x)| \le \frac{M}{(n+1)!} |\omega_{n+1}(x)| \qquad (3.5)$$

đây là công thức đánh giá sai số của đa thức nội suy.

Ta có thể áp dụng hệ quả (3.5) để đánh giá sai số đa thức nội suy.

Ví dụ. Cho bảng giá trị của hàm số y = sinx

x	0	$\frac{\pi}{4}$	$\frac{\pi}{2}$
y	0	0.707	1

Hãy đánh giá sai số khi dùng đa thức nội suy để tính gần đúng $\sin\frac{\pi}{3}$.

Giải. Bài ra không đặt vấn đề tính xấp xỉ **sin** $\frac{\pi}{3}$ mà chỉ yêu cầu tính sai số.

Ta có n = 2 và như vậy $M = \sup_{a \le x \le b} |\sin^{(n+1)}(x)| = 1$, do đó

$$|R_2(\frac{\pi}{3})| \le \frac{1}{3!}\frac{\pi}{3}\frac{\pi}{12}\frac{\pi}{6} = 0.024$$

Sau đây ta sẽ xét một số phương pháp tìm đa thức nội suy dựa vào các điểm mốc cách đều và không cách đều.

3.2.4. Phương pháp nội suy Lagrange

Giả sử ta có các điểm quan sát $x_0, x_1, \ldots x_n$ với khoảng chia đều hoặc không đều và một dãy các giá trị quan sát $y_0, y_1, \ldots y_n$.

Ý tưởng đơn giản đầu tiên là tìm một đa thức nội suy có bậc n (chính xác hơn là có bậc không quá n) sao cho trong đó các cặp (x_i,y_i) i = 0,1, ..., n có vai trò bình đẳng. Thí dụ ta tìm $p_n(x)$ có dạng:

$$p_n(x) = H_0(x) + H_1(x) + \ldots + H_n(x)$$

Các hàm $H_i(x)$ đều có bậc không quá n và $H_i(x_i) = y_i$, $H_{ki}(x_j) = 0$ khi $j \ne i$. Để $H_i(x_j) = 0$ khi $j \ne i$ thì $H_i(x)$ có dạng:

$$H_i(x) = K(x)(x-x_0)(x-x_1)\ldots(x-x_{i-1})(x-x_{i+1})\ldots(x-x_n)$$

Từ điều kiện $H_i(x_i) = y_i$ ta có

$$K(x)(x-x_0)(x-x_1)\ldots(x-x_{i-1})(x-x_{i+1})\ldots(x-x_n) = y_i$$

Suy ra

$$K(x) = y_i \frac{(x-x_1)(x-x_2)\ldots(x-x_{i-1})(x-x_{i+1})\ldots(x-x_n)}{(x_i-x_1)(x_i-x_2)\ldots(x_i-x_{i-1})(x_i-x_{i+1})\ldots(x_i-x_n)}$$

Nếu ta ký hiệu $L_i(x) = \dfrac{(x - x_1)(x - x_2)\ldots(x - x_{i-1})(x - x_{i+1})\ldots(x - x_n)}{(x_i - x_1)(x_i - x_2)\ldots(x_i - x_{i-1})(x_i - x_{i+1})\ldots(x_i - x_n)}$

Ta nhận thấy đa thức $L_i(x)$ có tính chất

$$L_i(x_j) = \begin{cases} 1 & j = i \\ 0 & j \neq i \end{cases}$$

và đa thức $p_n(x)$ có dạng

$$p_n(x) = y_0L_0(x) + y_1L_1(x) + \ldots + y_nL_n(x) \qquad (3.6)$$

Như vậy ta có

$$L_0(x) = \frac{(x - x_1)(x - x_2)\ldots(x - x_n)}{(x_0 - x_1)(x_0 - x_2)\ldots(x_0 - x_n)}$$

$$L_1(x) = \frac{(x - x_0)(x - x_2)\ldots(x - x_n)}{(x_1 - x_0)(x_1 - x_2)\ldots(x_1 - x_n)}$$

. . .

$$L_i(x) = \frac{(x - x_1)(x - x_2)\ldots(x - x_{i-1})(x - x_{i+1})\ldots(x - x_n)}{(x_i - x_1)(x_i - x_2)\ldots(x_i - x_{i-1})(x_i - x_{i+1})\ldots(x_i - x_n)}$$

. . .

$$L_n(x) = \frac{(x - x_0)(x - x_1)\ldots(x - x_{n-1})}{(x_n - x_0)(x_n - x_2)\ldots(x_n - x_{n-1})}$$

Đa thức nôi suy được xây dựng theo cách trên đây được gọi là đa thức nội suy Lagrange.

Ví dụ1 :Với hàm số y=sin(x/2) tại các nút giá trị sau:

I	x_i	y_i
0	0	0.000
1	1.5	0.682
2	2	0.841

Hãy xác định đa thức nội suy Lagrange đi qua các điểm trên? Hãy tính giá trị gần đúng của hàm số tại điểm x=1? Hãy đánh giá sai số lý thuyết tại x=1

Theo phần lý thuyết trên đa thức nội suy Lagrange đi qua các điểm (x_i,y_i) được xác định như sau

$$P(x)=\sum_{i=0}^{n} y_iL_i$$

Với $L_i=(\prod_{\substack{j=0 \\ j\neq i}}^{n} (x-x_j))/(\prod_{\substack{j=0 \\ j\neq i}}^{n} (x_i-x_j))$

Ta có:

$$L0(x)=(x-1.5)(x-2)/3$$

$$L1(x)=-4/3x(x-2)$$

$$L2(x)=x(x-1.5)$$

Vậy $P(x)= y_0L_{0(x)}+y_1L_{1(x)}+y_2L_{2(x)}=0- 0.682*4/3x(x-2) + 0.841*x(x-1.5)$

Vậy $P(1)=0- 0.682*4/3(1-2) + 0.841(1-1.5)$

$$P(1)=0.4888$$

* Đánh giá sai số lý thuyết:

$$R(x)=(f(3)(\eta)/3!)*x(x-1.5)(x-2).$$

$f(x)=\sin(x/2)$. vËy $f^{(3)}(x)=(-1/2^{(3)})\cos(x/2)$

Ta có $|f^{(3)}(x)|\le 1/8$

Vậy $R(1)\le(f^{(3)}(x)/3!)*(1-1.5)(1-2)\approx 0.01042$.

Ví dụ 2.Với hàm số y=sin(x/3) tại các nút giá trị sau:

I	x_i	y_i
0	0	0.000
1	1.5	0.479
2	2	0.618

Hãy xác định đa thức nội suy Lagrange đi qua các điểm trên? Hãy tính giá trị gần đúng của hàm số tại điểm x=1? Hãy đánh giá sai số lý thuyết tại x=1

Theo phần lý thuyết trên đa thức nội suy Lagrange đi qua các điểm (x_i,y_i) được xác định như sau

$$P(x)=\sum_{i=0}^{n} y_iL_i$$

Với $$L_i=\left(\prod_{\substack{j=0\\ j\neq i}}^{n} (x-x_j)\right)/\left(\prod_{\substack{j=0\\ j\neq i}}^{n} (x_i-x_j)\right)$$

Ta có:

$L_0(x)=(x-1.5)(x-2)/3$

$L1(x)=-4/3x(x-2)$

$L2(x)=x(x-1.5)$

Vậy $P(x)= y_0L_{0(x)}+y_1L_{1(x)}+y_2L_{2(x)}=0- 0.479*4/3x(x-2) + 0.618*x(x-1.5)$

Vậy $P(1)=0- 0.479*4/3(1-2) + 0.618(1-1.5)$

$P(1)\approx 0.3297$

* Đánh giá sai số lý thuyết:

$R(x)=(f(3)(\eta)/3!)*x(x-1.5)(x-2).$

$f(x)=\sin(x/3)$. vậy $f^{(3)}(x)=(-1/3^{(3)})\cos(x/3)$

Ta có $|f^{(3)}(x)|\leq 1/27$

Vậy $R(1)\leq(|f^{(3)}(x)|/3!)*(1-1.5)(1-2)\approx((1/27)/6)*0.5=0.00386$

3.2.5. Sai phân

Cách xây dựng đa thức nội suy Lagrange khá đơn giản về mặt ý tưởng. Tuy nhiên nhược điểm của nó là mỗi lần bổ sung thêm một số điểm quan sát mới ta lại phải tính lại từ đầu. Người ta tìm cách xây dựng một đa thức nội suy sao cho khi bổ sung các điểm quan sát thì ta không phải tính lại phần đa thức đã có. Thí dụ từ các điểm quan sát (x_0,y_0), (x_1,y_1),..., (x_k,y_k) ta tính được đa thức $p_k(x)$. Khi bổ sung thêm các điểm (x_{k+1},y_{k+1}),..., (x_n,y_n) thì đa thức nội suy tương ứng với mẫu quan sát (x_0,y_0),..., (x_n,y_n) sẽ có dạng $p_n(x) = p_k(x) + u(x)$.

Để thực hiện và trình bày điều này một cách rõ ràng, sáng sủa, trước hết ta cần đến khái niệm sai phân như sau:

a. Định nghĩa:

Cho f(x) là hàm của x và $h = \Delta x$ là một hằng số không đổi biểu thị cho khoảng thay đổi trên biến x và được gọi là số gia của x. Khi đó số gia tương ứng trên f(x):

$$\Delta f(x) = f(x+\Delta x) - f(x) \qquad (3.7)$$

được gọi là ***sai phân tiến*** cấp một tại điểm x của f(x) tương ứng với h. Gia số được tính bởi

$$\nabla f(x) = f(x) - f(x-\Delta x) \qquad (3.8)$$

được gọi là ***sai phân lùi*** cấp một tại điểm x của f(x) tương ứng với h.

Vì sai phân tiến g(x) của một hàm lại là một hàm của x do đó ta lại có thể định nghĩa sai phân tiến của g(x). Khi đó ta gọi sai phân tiến cấp một của g(x) là sai phân tiến cấp 2 của f(x), và cứ như vậy ta có thể định nghĩa sai phân tiến cấp k của một hàm f(x).

Với sai phân lùi ta cũng có lập luận và định nghĩa tương tự.

b. Sai phân tiến

Giả sử các điểm $x_0, x_1, ... x_n$ thoả mãn điều kiện

$$x_{i+1} - x_i = h$$

$$y_i = f(x_i), \ i = 0, 1, ... \qquad (3.9)$$

Ta có thể thấy rằng sai phân tiến

$$\Delta^2 y_i = \Delta(\Delta y_i) = \Delta y_{i+1} - \Delta y_i = y_{i+2} - y_{i+1} - (y_{i+1} - y_i) = y_{i+2} - 2y_{i+1} + y_i$$

Tổng quát ta có thể chứng minh rằng

$$\Delta^k y_i = \sum_{j=0}^{k} (-1)^j C_k^j y_{i+(k-j)} \qquad (3.10)$$

Bảng các sai phân tiến

x	**y**	**Δy**	**Δ^2y**	**Δ^3y**	**Δ^4y**
x_0	y_0	Δy_0	$\Delta^2 y_0$	$\Delta^3 y_0$	$\Delta^4 y_0$
x_1	y_1	Δy_1	$\Delta^2 y_1$	$\Delta^3 y_1$	$\Delta^4 y_1$
x_2	y_2	Δy_2	$\Delta^2 y_2$	$\Delta^3 y_2$	$\Delta^4 y_2$
x_3	y_3	Δy_3	$\Delta^2 y_3$	$\Delta^3 y_3$	$\Delta^4 y_3$
x_4	y_4	Δy_4	$\Delta^2 y_4$	$\Delta^3 y_4$	$\Delta^4 y_4$
.	.	.	.	.	.
					$\Delta^4 y_{n-5}$
				$\Delta^3 y_{n-4}$	$\Delta^4 y_{n-4}$
			$\Delta^2 y_{n-3}$	$\Delta^3 y_{n-3}$	
.	.	Δy_{n-2}	$\Delta^2 y_{n-2}$		
x_{n-1}	y_{n-1}	Δy_{n-1}			
x_n	y_n				

c. Sai phân lùi

Với sai phân lùi ta có

$$\nabla^2 y_i = \nabla (\nabla y_i) = \nabla y_i - \nabla y_{i-1} = y_i - y_{i-1} - (y_{i-1} - y_{i-2}) = y_i - 2y_{i-1} + y_{i-2}$$

Tổng quát ta có thể chứng minh rằng

$$\nabla^k y_i = \sum_{j=0}^{k} (-1)^j C_k^j y_{i-j} \qquad (3.11)$$

Bảng các sai phân lùi:

x	y	∇y	$\nabla^2 y$	$\nabla^3 y$	$\nabla^4 y$
x_0	y_0				
x_1	y_1	∇y_1			
.	.	∇y_2	$\nabla^2 y_2$		
			$\nabla^2 y_3$	$\nabla^3 y_3$	
				$\nabla^3 y_4$	$\nabla^4 y_4$
					$\nabla^4 y_5$
.	.	.	.	.	.
x_{n-4}	y_{n-4}	∇y_{n-4}	$\nabla^2 y_{n-4}$	$\nabla^3 y_{n-4}$	$\nabla^4 y_{n-4}$
x_{n-3}	y_{n-3}	∇y_{n-3}	$\nabla^2 y_{n-3}$	$\nabla^3 y_{n-3}$	$\nabla^4 y_{n-3}$
x_{n-2}	y_{n-2}	∇y_{n-2}	$\nabla^2 y_{n-2}$	$\nabla^3 y_{n-2}$	$\nabla^4 y_{n-2}$
x_{n-1}	y_{n-1}	∇y_{n-1}	$\nabla^2 y_{n-1}$	$\nabla^3 y_{n-1}$	$\nabla^4 y_{n-1}$
x_n	y_n	∇y_n	$\nabla^2 y_n$	$\nabla^3 y_n$	$\nabla^4 y_n$

3.2.6. Phương pháp sai phân Newton

a. Ý tưởng của phương pháp

Ta sẽ tìm một đa thức nội suy có bậc n (chính xác hơn là có bậc không quá n) sao cho trong đó các cặp (x_i,y_i) i = 0,1, ..., n, sao cho mỗi lần bổ sung thêm số liệu (thí dụ về cuối của mẫu quan sát) thì ta vẫn tận dụng được đa thức nội suy đã tính trước đó. Chính xác hơn, ta sẽ tìm đa thức nội suy $p_n(x)$ có dạng

$$p_n(x) = D_0(x) + D_1(x) + . . . + D_n(x)$$

sao cho đa thức $p_i(x) = D_0(x) + D_1(x) + . . . + D_i(x)$ là đa thức nội suy của mẫu:

$$(x_0,y_0), (x_1,y_1), . . ., (x_i,y_i). \qquad (3.12)$$

Từ những điều kiện trên đây ta thấy $D_i(x)$ là đa thức bậc cao nhất là i, và các hệ số của nó chỉ phụ thuộc vào mẫu con $(x_0,y_0), (x_1,y_1), . . ., (x_i,y_i)$.

Vì $D_0(x)$ là đa thức nội suy đi qua một điểm duy nhất (x_0,y_0), do đó đây là đa thức bậc 0, $D_0(x)=a_0$, trong đó a_0 là hằng số và ta có thể suy ra là $a_0 = y_0$.

Với i không âm, do $p_i(x)$ là đa thức nội suy của mẫu $(x_0,y_0), (x_1,y_1), . . ., (x_i,y_i)$, ta có:

$$p_i(x_j) = D_0(x_j) + D_1(x_j) + . . . + D_i(x_j) = y_j \quad j = 0,1, ...,i$$

Nhưng $p_n(x)$ là đa thức nội suy của mẫu $(x_0,y_0), (x_1,y_1), . . ., (x_n,y_n)$ nên ta cũng có

$$p_n(x_j) = D_0(x_j) + D_1(x_j) + . . . + D_i(x_j) + D_{i+1}(x_j) +... + D_n(x_j) = y_j \quad j = 0,1, ...,i$$

Kết hợp hai đẳng thức trên ta suy ra:

$$D_k(x_j) = 0, \text{ với } k = i+1, i+2,..., n; j = 0,1,2,...,i$$

Hay $x_0, x_1,..., x_i$ là nghiệm của các đa thức $D_k(x)$, k = i+1, i+2,..., n. Nghĩa là các đa thức $D_k(x)$ có dạng

$$D_k(x) = R(x)(x-x_0)(x-x_1)...(x-x_i),\ k = i+1,i+2,...,n$$

Với k = i+1 thì đa thức $D_k(x)$ có bậc không cao hơn i+1, do đó nó có dạng

$$D_{i+1}(x) = a_{i+1}(x-x_0)(x-x_1)...(x-x_i)$$

trong đó a_{i+1} là hằng số phụ thuộc vào mẫu $(x_0,y_0), (x_1,y_1), . . ., (x_{i+1},y_{i+1})$.

Như vậy đa thức nội suy được xây dựng thỏa mãn (3.12) có dạng

$$p_n(x) = a_0 + a_1(x-x_0) + a_2(x-x_0)(x-x_1) + . . . + a_i(x-x_0)(x-x_1)... (x-x_{i-1})+ . . .$$
$$+ a_n(x-x_0) (x-x_1)... (x-x_{n-1}) \qquad (3.13)$$

Trong đó a_i là hằng số phụ thuộc vào mẫu quan sát $(x_0,y_0), (x_1,y_1), . . ., (x_i,y_i)$.

b. Phương pháp sai phân tiến Newton với khoảng chia đều

Giả sử các điểm $x_0, x_1, ... x_n$ thoả mãn điều kiện

$$xi+1 - xi = h$$

$$y_i = f(x_i),\ i = 0, 1, ...$$

Ta giả thiết rằng:

$$\Delta^k y_i \neq 0,\ k =1,2, ...,m;\ m \leq n$$

$$\Delta^k y_i = 0,\ k > m$$

Khi đó ta có thể chọn đa thức nội suy có bậc m $p_m(x)$ theo phương pháp Newton tiến như sau:

$$p_m(x) = a_0 + a_1(x-x_0) + a_2(x-x_0)(x-x_1) + . . . + a_i(x-x_0)(x-x_1)... (x-x_{i-1})+ . . .$$
$$+ a_m(x-x_0) (x-x_1)... (x-x_{m-1}) \qquad (3.14)$$

Và ta có thể dùng đa thức này để nội suy các giá trị trong khoảng $[x_0,x_m]$.

Xác định các hệ số a_i

Thay lần lượt các giá trị $x = x_i$, i =0,1,2,...,m vào (3.14) ta được

$$p_m(x_0) = y_0 = a_0$$

Vậy $a_0 = y_0$

$$p_m(x_1) = y_1 = a_0 + a_1(x_1 - x_0) = y_0 + a_1h$$

Vậy $a_1 = \dfrac{y_1 - y_0}{h} = \dfrac{\Delta y_0}{h}$

$$p_m(x_2) = y_2 = a_0 + a_1(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1) = y_0 + a_1h =$$
$$= y_0 + 2(y_1 - y_0) + a_2\, 2h^2$$

Do đó $a_2\, 2h^2 = y_2 - 2y_1 + y_0 = \Delta^2 y_0$

Vậy $a_2 = \dfrac{\Delta^2 y_0}{2h^2}$

Tương tự với trường hợp tổng quát ta có

$$a_i = \frac{\Delta^i y_0}{i!h^i}\ ,\ i \leq m$$

Thay các a_i vào (3.14) ta có

$$p_m(x) = y_0 + \frac{\Delta y_0}{h}(x-x_0) + \frac{\Delta^2 y_0}{2h^2}(x-x_0)(x-x_1) + \ldots + \frac{\Delta^i y_0}{i!h^i}(x-x_0)(x-x_1)\ldots$$

$$+ (x-x_{i-1}) + \ldots + \frac{\Delta^m y_0}{m!h^m}(x-x_0)(x-x_1)\ldots(x-x_{m-1}) \qquad (3.15)$$

Ta có thể biểu diễn (3.15) dưới một dạng khác bằng phép biến đổi

$$t = \frac{x - x_0}{h} \quad \text{-> } x = x_0 + th$$

$$p_m(x) = y_0 + \frac{\Delta y_0}{h}(x-x_0) + \frac{\Delta^2 y_0}{2h^2}(x-x_0)(x-x_1) + \ldots$$

$$+ \frac{\Delta^i y_0}{i!h^i}(x-x_0)(x-x_1)\ldots(x-x_{i-1}) + \ldots + \frac{\Delta^m y_0}{m!h^m}(x-x_0)(x-x_1)\ldots(x-x_{m-1})$$

$$p_m(x) = p_m(x_0 + th) = y_0 + (\Delta y_0)t + \frac{\Delta^2 y_0}{2}t(t-1) + \ldots$$

$$+ \frac{\Delta^i y_0}{i!}t(t-1)\ldots(t-i+1) + \ldots + \frac{\Delta^m y_0}{m!}t(t-1)\ldots(t-m+1) \quad (3.15b)$$

Ví dụ. Cho bảng sai phân sau:

x	y	Δy	Δ^2y	Δ^3y
2	23	70	96	48
4	93	166	144	48
6	259	310	192	48
8	569	502	240	48
10	1071	742	288	
12	1813	1030		
14	2843			

Ta thấy các sai phân bậc nhỏ hơn 4 khác không nhưng sai phân bậc bốn đều bằng không, do đó chúng ta chỉ xây dựng được đa thức bậc cao nhất là 3. Chọn x_0=4, x_1=6, x_2 = 8, ta được đa thức bậc ba là

$$p_3(x) = 93 + 83(x-4) + 18(x-4)(x-6) + (x-4)(x-6)(x-8)$$

Muốn tính giá trị của hàm f(x) tại các điểm x thuộc khoảng [4,8] ta chỉ cần thay giá trị x vào đa thức vừa lập được và tính giá trị của đa thức. Chẳng hạn với x = 4.2 ta có:

$$p_3(4.2) = 93 + 83(4.2-4) + 18(4.2-4)(4.2-6) + (4.2-4)(4.2-6)(4.2-8) = 104.48$$

Nhận xét về phương pháp sai phân tiến Newton:

Công thức nội suy Newton tiến thường được dùng để nội suy các giá trị của hàm f(x) ở vùng đầu bảng. Để nội suy các giá trị ở cuối bảng người ta thường dùng phương pháp sai phân lùi.

c. Phương pháp Newton với khoảng chia không đều

Trong thực tế các điểm $x_0, x_1, ... x_n$ có thể không cách đều. Lúc này khoảng cách $x_{i+1} - x_i = h_i$ không phải là hằng số. Trong trường hợp này ta cũng xây dựng được đa thức nội suy Newton có dạng như (3.14) như trường hợp cách đều, nhưng cách tính toán các hệ số có khác. Ta đưa ra các ký hiệu sau, có thể xem là dạng tổng quát hóa của sai phân tiến.

Với số số nguyên $p \geq 0$ bất kỳ ta định nghĩa ***tỷ sai phân cấp 1*** là đại lượng

$$y[x_{p+1},x_p] = \frac{y_{p+1} - y_p}{x_{p+1} - x_p}$$

Ví dụ: $y[x_1,x_0] = \frac{y_1 - y_0}{x_1 - x_0}$

Tỷ sai phân cấp 2

$$y[x_{p+2},x_{p+1},x_p] = \frac{y[x_{p+2},x_{p+1}] - y[x_{p+1},x_p]}{x_{p+2} - x_p}$$

Ví dụ: $y[x_2,x_1,x_0] = \frac{y[x_2,x_1] - y[x_1,x_0]}{x_2 - x_0}$

. . .

Tỷ sai phân cấp k:

$$y[x_{p+k},..., x_{p+1}, x_p] = \frac{y[x_{p+k},...,x_{p+2},x_{p+1}] - y[x_{p+k-1},...,x_{p+1},x_p]}{x_{p+k} - x_p}$$

Ví dụ: $y[x_k,..., x_1, x_0] = \frac{y[x_k,...,x_2,x_1] - y[x_{k-1},...,x_1,x_0]}{x_k - x_0}$

Bây giờ ta xét đa thức nội suy

$$p_m(x) = a_0 + a_1(x-x_0) + a_2(x-x_0)(x-x_1) + . . . + a_i(x-x_0)(x-x_1)... (x-x_{i-1}) + . . .$$
$$+ a_m(x-x_0) (x-x_1)... (x-x_{m-1}) \qquad (3.16)$$

Thay lần lượt cácgiá trị $x = x_i$, i=0,1,...,n vào (3.16)

Ta có:

$a_0 = y_0$

$$y_1 - y_0 = a_1(x_1 - x_0) \Rightarrow a_1 = \frac{y_1 - y_0}{x_1 - x_0} = y[x_1,x_0].$$

$$y_2 - y_0 = a_1(x_2-x_0)+ a_2(x_2-x_0)(x_2-x_1) = \frac{y_1 - y_0}{x_1 - x_0}(x_2 - x_0) + a_2(x_2-x_0)(x_2-x_1)$$

$$a_2(x_2-x_0)(x_2-x_1) = (y_2 - y_0) - \frac{y_1 - y_0}{x_1 - x_0}(x_2 - x_0)$$

$$a_2 = \frac{y_2 - y_0}{(x_2 - x_0)(x_2 - x_1)} - \frac{y_1 - y_0}{(x_1 - x_0)(x_2 - x_1)} =$$

$$= \frac{(y_2 - y_1 + y_1 - y_0)(x_1 - x_0) - (y_1 - y_0)(x_2 - x_0)}{(x_2 - x_0)(x_1 - x_0)(x_2 - x_1)} \qquad (3.17)$$

Xét tử số ta có

$(y_2 - y_1 + y_1 - y_0)(x_1 - x_0) - (y_1 - y_0)(x_2 - x_1 + x_1 - x_0) =$

$(y_2 - y_1)(x_1 - x_0) + (y_1 - y_0)(x_1 - x_0) - (y_1 - y_0)(x_2 - x_1) - (y_1 - y_0)(x_1 - x_0) =$

$= (y_2 - y_1)(x_1 - x_0) - (y_1 - y_0)(x_2 - x_1)$

Thay vào (3.17) và giản ước ta có

$$a_2 = \frac{y[x_2,x_1] - y[x_1,x_0]}{(x_2 - x_0)} = y[x_2, x_1, x_0]$$

Tổng quát ta có thể chứng minh rằng

$a_k = y[x_k,..., x_1, x_0]$

Tương tự như phương pháp sai phân chia đều, ta có bảng tính tỷ sai phân (t.s.p) Newton tổng quát như sau:

X	y	t.s.p bậc 1	t.s.p bậc 2	t.s.p bậc 3	t.s.p bậc 4
X_0	y_0	$y[x_1,x_0]$	$y[x_2,x_1,x_0]$	$y[x_3,x_2,x_1,x_0]$	$y[x_4,x_3,x_2,x_1,x_0]$
X_1	y_1	$y[x_2,x_1]$	$y[x_3,x_2,x_1]$	$y[x_4,x_3,x_2,x_1]$	
X_2	y_2	$y[x_3,x_2]$	$y[x_4,x_3,x_2]$		
X_3	y_3	$y[x_4,x_3]$			
X_4	y_4				

Ví dụ. Cho bảng giá trị (x_i,y_i) (i=0,2...,4)tương ứng như sau

x	-4	-1	0	2	5
y	1245	33	5	9	1335

Hãy thiết lập đa thức nội suy Newton qua các điểm trên?

Vậy ta có bảng tỷ sai phân như sau:

x	y	t.s.p bậc 1	t.s.p bậc 2	t.s.p bậc 3	t.s.p bậc 4
-4	1245	-404	94	-14	3
-1	33	-28	10	13	
0	5	2	88		
2	9	442			
5	1335				

Chọn $x_0 = -4$ ta được đa thức nội suy

$$p_4(x) = 1245 - 404(x+4) + 94(x+4)(x+1) - 14(x+4)(x+1)x + 3(x+4)x(x-2) =$$
$$= 5 - 14x + 6x^2 - 5x^3 + 3x^4$$

d. Cài đặt phương pháp sai phân Newton tổng quát

Ta có thể thấy rằng thuật toán trong trường hợp khoảng chia không cách đều hoàn toàn có thể áp dụng cho trường hợp cách đều. Do vậy ta chỉ cài đặt cho trường hợp tổng quát.

Ta sẽ lưu trữ các điểm quan sát và các kết quả quan sát trong các vectơ x,y; các hệ số a_i trong vectơ a.

Để tính toán và lưu trữ các tỷ sai phân ta sẽ viết lại bảng tỷ sai phân dưới dạng sau

x	y	s.p bậc 1	s.p bậc 2	s.p bậc 3	s.p bậc 4
x_0	y_0	$y[x_1,x_0]$	$y[x_2,x_1,x_0]$	$y[x_3,x_2,x_1,x_0]$	$y[x_4,x_3,x_2,x_1,x_0]$
x_1	y_1	$y[x_2,x_1]$	$y[x_3,x_2,x_1]$	$y[x_4,x_3,x_2,x_1]$	
x_2	y_2	$y[x_3,x_2]$	$y[x_4,x_3,x_2]$		
x_3	y_3	$y[x_4,x_3]$			
x_4	y_4				

Các tỷ sai phân bậc k được tính bởi công thức sau:

$$y[x_{i+k},x_{i+k-1},...,x_i] = \frac{y[x_{i+k},x_{i+k-1},...,x_{i+1}] - y[x_{i+k-1},...,x_{i+1},x_i]}{(x_{i+k} - x_i)} \quad (3.18)$$

Ta có nhận xét sau:

Ta đánh số các sai phân bậc k căn cứ vào vị trí xuất phát i của nó và lưu trữ trong vectơ sp[i], i =0,1,2,...,n-1. Tỷ sai phân bậc k có vị trí xuất phát là i sẽ được lưu trữ trong phần tử sp[i] như bảng sau:

x	y	sp[i]	s.p bậc 1	s.p bậc 3	s.p bậc 4
x_0	y_0	sp[0]	$y[x_1,x_0]$	$y[x_3,x_2,x_1,x_0]$	$y[x_4,x_3,x_2,x_1,x_0]$
x_1	y_1	sp[1]	$y[x_2,x_1]$	$y[x_4,x_3,x_2,x_1]$	
x_2	y_2	sp[2]	$y[x_3,x_2]$		
x_3	y_3	sp[3]	$y[x_4,x_3]$		
x_4	y_4				

Khi k thay đổi thì ta có cảm giác như cần một mảng 2 chiều để lưu trữ các tỷ sai phân vì ở đây ta có 2 chỉ số là i và k. Tuy nhiên mục đích chúng ta không phải là tính các tỷ sai phân mà là tính các hệ số a_i, do đó ta sẽ thấy rằng chỉ cần lưu trữ sai phân trong một mảng vectơ là đủ. Các bước được tiến hành như sau:

- **Bước 0:** Đặt a[0] = y[0]

- **Bước 1:** Ta tính các sai phân bậc nhất và lưu vào vectơ sp, $y[x_{i+1},x_i]$ được lưu trữ vào sp[i], i = 0,1,2, ..., n-1

  Đặt a[1] = sp[0]

- **Bước 2:** Ta tính các sai phân bậc hai $y[x_{i+2},x_i]$ bắt đầu từ i=0,1,...,n-2 và lưu trữ vào sp[i], i = 0,1,2,..., n-2. Để tính $y[x_{0+2},x_0]$ chẳng hạn, ta đặt

$$sp[0] = \frac{y[x_2,x_1] - y[x_1,x_0]}{x_2 - x_0} = \frac{sp[1] - sp[0]}{x_2 - x_0}$$

  Tương tự ta có

$$sp[1] = \frac{y[x_3,x_2] - y[x_2,x_1]}{x_3 - x_1} = \frac{sp[2] - sp[1]}{x_3 - x_1}$$

  . . .

  Như vậy ta thấy khi tính lại sp[0] ta cần đến sp[0] và sp[1], khi tính sp[1] ta cần đến sp[1] và sp[2], ... như vậy quá trình tính toán sp[i] chỉ cần đến các phần tử từ vị trí i trở về sau mà không cần đến các vị trí trước i. Như vậy sau khi tính tỷ sai phân bậc 2 thứ i ta có thể dùng ngay vị trí i để lưu trữ không sợ rằng vị trí này bị những tính toán tiếp theo làm thay đổi, vì khi tính tỷ sai phân thứ i+1 ta chỉ cần giá trị sp[i+1] và sp[i+2].

  Đặt a[2] = sp[0].

  ……..

- **Bước k:** Ta tính các tỷ sai phân bậc k $y[x_{i+k},x_i]$ bắt đầu từ i=0,1,...,n-k và lưu trữ vào sp[i], i = 0,1,2, ..., n -k. Để tính $y[x_{0+k},x_0]$ chẳng hạn, ta đặt

$$sp[0] = \frac{y[x_k,x_1] - y[x_{k-1},x_0]}{x_k - x_0} = \frac{sp[1] - sp[0]}{x_k - x_0}$$

  Thực hiện tương tự với i =1,2,.., n-k.

  Đặt a[k] = sp[0]

  . . .

- **Bước n:** Ở bước cuối cùng ta tính

$$sp[0] = \frac{sp[1] - sp[0]}{x_n - x_0}$$

  Đặt a[n] = sp[0]

(Đoạn chương trình mô tả phương pháp được thể hiện ở phần sau)

3.2.7. Phép nội suy ngược

Trong các phần trước ta xét bài toán cho giá trị hàm y = f(x) tại các điểm quan sát $x_0, x_1, ... x_n$ và cần xác định giá trị y = f(x) tại những điểm x không có trong các điểm quan sát. Bây giờ ta xét bài toán ngược lại: vẫn là các giả thiết trên, tức là cho bảng các giá trị y_i của hàm y = f(x) tại các điểm x_i, i=0,1,...,n. Cho biết giá trị y', ta hãy tính x' tương ứng. Bài toán này được gọi là bài toán nội suy ngược. Một trong những ứng dụng của nội suy ngược là tìm nghiệm xấp xỉ của phương trình f(x)=0.

Cách giải quyết bằng đa thức nội suy Lagrange.

Ta xem x = φ(y) là hàm ngược của hàm y = f(x) và xem các giá trị $y_0, y_1,..., y_n$ là các mốc nội suy (nói chung các mốc y_i không đều). Từ đó từ đó biết y' ta sẽ tính được x' = φ(y').

Chương trình cài đặt các đa thức nội suy

Sau đây là đoạn chương trình chính thể hiện (mô tả) phương pháp Newton và phương pháp Lagrange:

```
/*Tra ve gia tri da thuc noi suy tai x; anew[i] la cac he so da thuc noi suy Newton, xqs[i] la cac diem quan sat*/
double polinew(double x)
 {int i;double mx,s;
  s=anew[0];
  mx=1;
  for(i=0;i<nqs;i++)
   {mx*=(x-xqs[i]);
    s+=anew[i+1]*mx;
   }
  return s;
 }
//=================================================
/*Tra ve gia tri da thuc noi suy tai x; alag[i] la cac he so cua
  da thuc noi suy Lagrange , xqs[i] la cac diem quan sat*/
double polilag(double x)
 {int i,j;double mx,s;
  s=0;
  for(i=0;i<=nqs;i++)
   {mx=1;
    for(j=0;j<=nqs;j++)
     if(j!=i) mx*=(x-xqs[j]);
    s+=alag[i]*mx;
   }
  return s;
 }
//=================================================
/*Noi suy Newton voi khoang chia khong can deu */
void nsnewton(double *a)
```

```
{ int h,i,j,k;double tmp;kvecto sp;
  a[0]=yqs[0];
  for(i=0;i<=nqs-1;i++) //Tinh sai phan bac 1;
  sp[i]=(yqs[i+1]-yqs[i])/(xqs[i+1]-xqs[i]);
  a[1]=sp[0];
  for(k=2;k<=nqs;k++)//Tinh cac sai phan bac k va cac he so a[i]
  { for(i=0;i<=nqs-k;i++)
    sp[i]=(sp[i+1]-sp[i])/(xqs[i+k]-xqs[i]);
    a[k]=sp[0];
  }
}
//=====================================================
/*Noi suy Lagrange voi khoang chia khong can deu
  Tinh cac he so*/
  a[i] = 1/((x[i]-x[0])(x[i]-x[1])...(x[i]-x[i-1])(x[i]-x[i+1])...x[m])*/
void nslagrange(double *a)
{ int h,i,j,k;double tmp;
  for(i=0;i<=nqs;i++) //Tinh cac he so a[i];
  { tmp=1;
    for(j=0;j<=nqs;j++) if(j!=i) tmp=tmp*(xqs[i]-xqs[j]);
    a[i]=yqs[i]/tmp;
  }
}
```

3.3. KHỚP ĐƯỜNG CONG - NỘI SUY SPLINE

Trong phần trước ta đã xét bài toán nội suy dùng đa thức và như ta đã thấy, các đa thức nội suy thường có bậc là n, trong đó n+1 là số điểm quan sát. Ta có thể nội suy bằng đa thức bậc m nhỏ hơn n, nhưng như vậy thì ta cũng chỉ dùng đến mẫu quan sát dựa trên m+1 điểm là (x_0,y_0), (x_1,y_1), . . . ,(x_m,y_m) và như thế chỉ nội suy được giá trị của hàm tại các điểm $x \in [x_0,x_m]$. Điều này tỏ ra không được phù hợp với thực tế cho lắm. Thật vậy, giả sử trong thực tế hàm f(x) là một đa thức bậc 3 nhưng vì ta không biết điều này nên phải dùng đa thức nội suy. Theo một cách tự nhiên, ta nghĩ rằng nếu có càng nhiều thông tin thì ta càng giải quyết bài toán tốt hơn. Nghĩa là nếu có càng nhiều điểm quan sát thì kết quả của chúng ta càng gần với thực tế hơn. Tuy nhiên nếu dùng đa thức nội suy như kiểu chúng ta vừa khảo sát thì không có được như điều chúng ta mong đợi. Mặc dầu dạng thật của đa thức là bậc 3, nhưng nếu dùng 5 điểm quan sát thì ta phải tính các hệ số đa thức bậc 4, 10 điểm thì ta phải tính toán với đa thức bậc 9,...nghĩa là càng dùng nhiều điểm thì ta càng đi xa thực tế hơn. Phép nội suy đa thức còn có một nhược điểm nữa là số lượng

phép tính cần thực hiện phụ thuộc rất nhiều vào cỡ của mẫu quan sát. Trong kỹ thuật truyền thông chẳng hạn, việc chuyển đổi một tín hiệu số có hàng ngàn điểm quan sát sang dạng tương tự là vấn đề thường gặp. Thế nhưng chỉ cần nội suy đa thức cho 101 điểm quan sát ta đã phải dùng đến đa thức bậc 100, và việc dùng đa thức bậc 100 để tính toán cho các điểm còn lại là một việc tiêu tốn tài nguyên máy một cách quá lãng phí. Vì vậy có thể nói rằng phép nội suy đa thức chỉ có ý nghĩa lý thuyết mà thôi, trong thực tế hầu như người ta không dùng đến.

Để tìm kiếm một cách nội suy gần với thực tế hơn, chúng ta hãy bắt đầu bằng một thao tác đơn giản mà chúng ta hay thực hiện hồi còn học phổ thông. Khi vẽ một đồ thị hàm số nào đó, đầu tiên ta vẽ các điểm rời rạc, và vẽ được càng nhiều điểm càng tốt. Sau đó ta dùng bút nối các điểm đó với nhau, nhưng ta không nối bằng thước kẻ, mà nối bằng bút và sự quan sát bằng mắt sao cho các đoạn nối các điểm thành một đường mịn, không bị gãy khúc.

Những người chuyên vẽ sơ đồ thiết kế dùng một thiết bị cơ học gọi là spline để vẽ các đường cong đẹp, có thẩm mỹ: người vẽ xác định tập hợp các điểm (nút) rồi bẻ cong một giải plastic hay thanh gỗ linh hoạt (spline) quanh chúng và lấy vết chúng để tạo thành một đường cong. Nội suy spline về mặt toán học tương đương với tiến trình này và cho ra cùng một kết quả.

3.4. PHƯƠNG PHÁP BÌNH PHƯƠNG CỰC TIỂU

Trong phương pháp nội suy đa thức đã xét, ta đòi hỏi đa thức phải đi qua các điểm quan sát. Điều này đôi khi là điều kiện quá chặt trong thực tế. Sau đây ta sẽ xác định tham số của một hàm f(x) có dạng đã biết, sao cho các giá trị $f(x_i)$ xấp xỉ tốt nhất các giá trị y_i theo một tiêu chuẩn gọi là bình phương cực tiểu. Trong bài toán ước lượng bình phương cực tiểu ta phải giả thiết là dạng hàm f(x) đã biết và ta chỉ cần ước lượng các tham số. Nói chung đối với dạng bài toán này thì ta không thể đặt ra yêu cầu là đồ thị hàm số y = f(x) phải đi qua các điểm quan sát, mà chỉ có thể đặt ra yêu cầu là đồ thị gần các điểm quan sát nhất trong tập hợp các dạng hàm đã cho.

3.4.1. Trường hợp hàm nội suy là đa thức hay y phụ thuộc các tham số một cách tuyến tính

Bây giờ ta xét một trường hợp hay được áp dụng nhất là hàm f(x) có dạng đa thức bậc m, tức là:

$$f(x) = a_0 + a_1x^1 + \ldots a_{m-1}x^{m-1} + a_mx^m$$

Vì giá trị $f(x_i)$ nói chung khác giá trị y_i, giả sử sai số là e_i, ta có

$$y_i = a_0 + a_1x_i^1 + \ldots a_{m-1}x_i^{m-1} + a_m x_i^m + e_i$$

$i=0,1,2,\ldots,n$

Như vậy ta có:

$$e_i = y_i - \sum_{j=0}^{m} a_j x_i^j$$

Và tổng bình phương các sai số bằng

$$S = \sum_{i=0}^{n} e_i^2 = \sum_{i=0}^{n} (y_i - \sum_{j=0}^{m} a_j x_i^j)^2$$

Để S đạt giá trị nhỏ nhất thì điều kiện sau phải thỏa mãn

$\partial S / \partial a_k = 0$, với $k=0,1,...,m$

Thực hiện phép lấy đạo hàm riêng từng thành phần của tổng theo a_k ta có

$$\partial (y_i - \sum_{j=0}^{m} a_j x_i^j)^2 / \partial a_k = 2(y_i - \sum_{j=0}^{m} a_j x_i^j)(- x_i^k) = 2(-y_i x_i^k + \sum_{j=0}^{m} a_j x_i^{j+k})$$

Như vậy

$$\partial S / \partial a_k = 2\sum_{i=0}^{n} (-y_i x_i^k + \sum_{j=0}^{m} a_j x_i^{j+k}) = 0, \quad k=0,1,2,...,m$$

Từ đây

$$\sum_{j=0}^{m} a_j \sum_{i=0}^{n} x_i^{j+k} = \sum_{i=0}^{n} y_i x_i^k, \quad k = 0,1,2,...,m$$

Với $k = 0,1,2,..,m$

$$(n+1)a_0 + a_1\sum_{i=0}^{n} x_i + a_2\sum_{i=0}^{n} x_i^2 + \ldots + a_m\sum_{i=0}^{n} x_i^m = \sum_{i=0}^{n} y_i$$

$$a_0\sum_{i=0}^{n} x_i + a_1\sum_{i=0}^{n} x_i^2 + a_2\sum_{i=0}^{n} x_i^3 + \ldots + a_m\sum_{i=0}^{n} x_i^{m+1} = \sum_{i=0}^{n} y_i x_i$$

$$a_0\sum_{i=0}^{n} x_i^2 + a_1\sum_{i=0}^{n} x_i^3 + a_2\sum_{i=0}^{n} x_i^4 + \ldots + a_m\sum_{i=0}^{n} x_i^{m+2} = \sum_{i=0}^{n} y_i x_i^2$$

. . .

$$a_0\sum_{i=0}^{n} x_i^m + a_1\sum_{i=0}^{n} x_i^{m+1} + a_2\sum_{i=0}^{n} x_i^{m+2} + \ldots + a_m\sum_{i=0}^{n} x_i^{m+m} = \sum_{i=0}^{n} y_i x_i^m$$

Đặt

$$C = \begin{bmatrix} \sum_{i=0}^{n} x_i^0 & \sum_{i=0}^{n} x_i & \sum_{i=0}^{n} x_i^2 & \sum_{i=0}^{n} x_i^3 & \ldots & \sum_{i=0}^{n} x_i^m \\ \sum_{i=0}^{n} x_i & \sum_{i=0}^{n} x_i^2 & \sum_{i=0}^{n} x_i^3 & \sum_{i=0}^{n} x_i^4 & \ldots & \sum_{i=0}^{n} x_i^{m+1} \\ \sum_{i=0}^{n} x_i^2 & \sum_{i=0}^{n} x_i^3 & \sum_{i=0}^{n} x_i^4 & \sum_{i=0}^{n} x_i^5 & \ldots & \sum_{i=0}^{n} x_i^{m+1} \\ \ldots & & & & & \\ \sum_{i=0}^{n} x_i^m & \sum_{i=0}^{n} x_i^{m+1} & \sum_{i=0}^{n} x_i^{m+2} & \sum_{i=0}^{n} x_i^{m+3} & \ldots & \sum_{i=0}^{n} x_i^{2m} \end{bmatrix}$$

$$d = (\sum_{i=0}^{n} y_i x_i^0, \sum_{i=0}^{n} y_i x_i^1, \ldots, \sum_{i=0}^{n} y_i x_i^m)$$

Ta có hệ phương trình

$$C\,a = d$$

Tuy nhiên, để tiện cho việc tính toán, ta có nhận xét sau đây:

Đặt $y = (y_0, y_1, \ldots, y_n)^T$, $e = (e_0, e_1, \ldots, e_n)^T$, $a = (a_0, a_1, \ldots, a_n)^T$

$$F = \begin{bmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^m \\ 1 & x_1 & x_1^2 & \ldots & x_1^m \\ . & . & & \ldots & . \\ 1 & x_n & x_n^2 & \ldots & x_n^m \end{bmatrix}$$

Hệ phương trình $C\,a = d$ có thể viết dưới dạng sau:

$$F^T F\,a = F^T y$$

Ta có thể áp dụng phương pháp Gauss-Jordan để giải hệ phương trình này.

Ta có thể thấy rằng nếu $m \le n$ thì định thức của ma trận C khác không và vì vậy đa thức nội suy bình phương cực tiểu được xác định duy nhất.

Thật vậy, viết chi tiết ma trận C ta có

$$C = \begin{bmatrix} 1+1+\ldots+1 & x_0+x_1+\ldots+x_n & \ldots & x_0^m+x_1^m+\ldots+x_n^m \\ x_0+x_1+\ldots+x_n & x_0^2+x_1^2+\ldots+x_n^2 & \ldots & x_0^{m+1}+x_1^{m+1}+\ldots+x_n^{m+1} \\ . & . & \ldots & . \\ x_0^m+x_1^m+\ldots+x_n^m & x_0^{m+1}+x_1^{m+1}+\ldots+x_n^{m+1} & \ldots & x_0^{2m}+x_1^{2m}+\ldots+x_n^{2m} \end{bmatrix}$$

Bằng cách tách ra các cột ta thu được $(n+1)^{m+1}$ ma trận con $C_1, C_2, \ldots$, mỗi cột của ma trận con chỉ phụ thuộc các số $1, x_0, x_1, \ldots, x_n$. Sau khi đã tách được như vậy, bằng cách đặt thừa số chung ra ngoài ta lại thu được các ma trận mà mỗi ma trận có $m+1$ cột, các cột này được lấy từ tổ hợp của $(n+1)$ các cột có dạng

$$\begin{pmatrix} 1 \\ x_0 \\ x_0^2 \\ . \\ x_0^m \end{pmatrix}, \begin{pmatrix} 1 \\ x_1 \\ x_1^2 \\ . \\ x_1^m \end{pmatrix}, \ldots, \begin{pmatrix} 1 \\ x_n \\ x_n^2 \\ . \\ x_n^m \end{pmatrix}$$

Dễ thấy rằng:

- Nếu $m>n$ thì các ma trận con luôn có 2 cột nào đó trùng nhau nên định thức bằng 0 và do đó $\det C = \Sigma \det C_i = 0$.
- Nếu $n \ge m$: ma trận C được tách thành hai loại ma trận:

Loại 1: Các ma trận có 2 cột nào đó cùng phụ thuộc một thành phần trong $x_0,x_1,...,x_n$ nên có định thức bằng 0.

Loại 2: Các ma trận có m+1 cột khác nhau lấy từ (n+1) cột có dạng trên. Các ma trận này có dạng Vandermond nên khác không. Do đó det C ≠ 0.

3.4.2. Trường hợp y phụ thuộc các tham số một cách phi tuyến

a. $y = ae^{bx}$, a>0 (3.19)

Lấy logarit hai vế, ta có

$$\ln y = \ln a + bx$$

Đặt Y = lny, A = lna, B = B, X = x (3.18) trở thành

$$Y = A + BX \quad (3.20)$$

Như vậy bằng cách lấy logarit hai vế, ta đã đưa quan hệ phi tuyến (3.19) thành dạng tuyến tính (3.20). Ta tính được A và B, từ đây tính được a, b.

b. $y = ax^b$, a>0 (3.21)

Lấy logarit hai vế, ta có

$$\ln y = \ln a + b\ln x$$

Đặt Y = lny, A = lnA, B = b, X = lnx (3.21) trở thành

$$Y = A + BX \quad (3.22)$$

Từ đây ta tính được A và B, và suy ra a, b.

Chương trình cài đặt các đa thức nội suy

Sau đây là đoạn chương trình chính thể hiện (mô tả) thuật toán hồi qui bằng bình phương cực tiểu

```
/*Hoi quy dung da thuc uoc luong theo phuong phap binh phuong cuc tieu*/
/*Cho truoc bac m, xac dinh cac he so da thuc thuc nghiem , tra ve tong binh phuong cac sai so*/
double regresspoli(double *a,int m)
 {int i,j,k;
  kmatran aa;
  double **f,**ft;
  f = new double* [nqs+1];
  for(I=0;i<=nqs;i++) f[i] = new double [m+1];
  ft = new double* [m+1];
  for(I=0;i<=m;i++) ft[i] = new double [nqs+1];
 /*Tinh ma tran
   f la ma tran co kieu nhu Vandermon nhung co n hang m cot,
   ft la ma tran chuyen vi cua f. Nhu vay ft x f se la ma tran aa cua
   he phuong trinh tuyen tinh
 */
```

```
for(i=0;i<=nqs;i++)
 {f[i][0]=1;
  for(j=1;j<=m;j++)
   f[i][j]=f[i][j-1]*xqs[i];
 }
//Tinh ma tran chuyen vi
for(i=0;i<=m;i++)
 for(j=0;j<=nqs;j++) ft[i][j]=f[j][i];
/*Tinh ma tran vuong aa cap m, chi can tinh tinh nua tren roi gan cho
  nua duoi, vi ma tran aa la doi xung
*/
for(i=0;i<=m;i++)
 for(j=i;j<=m;j++)
  {aa[i][j]=ft[i][0]*f[0][j];
   for(k=1;k<=nqs;k++) aa[i][j]+=ft[i][k]*f[k][j];
  }
//Gan nua duoi
for(i=0;i<=m;i++)
 for(j=0;j<i;j++) aa[i][j]=aa[j][i];
//Tinh ve phai cua he phuong trinh
for(i=0;i<=m;i++)
 {aa[i][m+1]=ft[i][0]*yqs[0];
  for(k=1;k<=nqs;k++) aa[i][m+1]+=ft[i][k]*yqs[k];
 }
for(i=0;i<=nqs;i++) delete [] f[i];
for(i=0;i<=m;i++) delete [] ft[i];
gjordan(aa,a,m);
//Tinh tong binh phuong cac sai so
double ss,fa,xx;
ss=0;
for(i=0;i<=nqs;i++)
 {fa=1;xx=1;
  for(j=1;j<=m;j++)
    {xx=xx*xqs[i];fa+=a[j]*xx;}
 ss+=(yqs[i]-fa)*(yqs[i]-fa);
 }
return ss;
}
```

3.5. BÀI TẬP

Bài 1. Cho hàm số $y = 2^x$ với các giá trị tại $x_i = 3,50 + 0,05i$, i=0,1,2,3,4 tuần tự là 33,115;34,813; 36,598; 38,475; 40,477. Hãy lập đa thức nội suy Newton tiến xuất phát từ nút 3,50.

Bài 2. Tích phân xác suất

$$\phi(x) = (2/\text{sqrt}(\pi))\int_0^x \exp(-t^2)\, dt$$

Không tính được bằng nguyên hàm. Người ta tính gần đúng nó tại $x_i = 1 + 0,1i$, i=0,1,2,..,10 được các giá trị xấp xỉ tuần tự là 0,8427; 0,8802; 0,9103; 0,9523; 0,9661; 0,9763; 0,9838; 0,9891; 0,9928; 0,9953. Hãy tính $\phi(1,43)$

Bài 3. Cho hàm số $y = e^x$ tại $x = 0,65 + 0,1i$ i=0,1,...,5 tuần tự là 1,91554; 2,11700; 2,33965; 2,58571; 2,85765; 3,15819. Hãy tính ln2.

Bài 4. Biết rằng đại lượng y là một tam thức bậc hai của x và tại x =0,78; 1,56; 2,34; 3,12; 3,81 các giá trị của y tuần tự là 2,5; 1,20; 1,12; 4,28. Hãy lập công thức y biểu diễn qua x.

Bài 5. Hãy đánh giá sai số nhận được khi xấp xỉ hàm số y = sinx bằng đa thức nội suy Lagrange bậc 5 $L_5(x)$, biết rằng đa thức này trùng với hàm số đã cho tại các giá trị x bằng: 0^0, 5^0, 10^0, 15^0, 20^0, 25^0.

Xác định gía trị của sai số khi $x = 12^0 30'$.

Bài 6. Cho bảng các giá trị:

x	2	4	6	8	10	12
	7 .32	8 .24	9 .20	10 .19	11 .01	12 .05

Hãy áp dụng phương pháp bình phương cực tiểu xác định các đa thức xấp xỉ có các dạng: $y = a + bx$, $y = a+bx +cx^2$, $y = ax^b$

và so sánh các sai số để kết luận dạng nào thích hợp nhất.

Bài 7. Thử lại hoặc viết mới các chương trình cài đặt các thuật toán rồi chạy thử để kiểm tra các kết quả trên đây.

TÓM TẮT NỘI DUNG CHƯƠNG 3

Trong chương này chúng ta cần chú ý nhất là các vấn đề sau:

1. Sai số của đa thức nội suy

Với $M = \sup_{a \le x \le b} |f^{(n+1)}(x)|$ khi đó ta có: $|R(x)| \le |f(x) - p_m(x)| \le \frac{M}{(n+1)!} |\omega_{n+1}(x)|$

đây là công thức đánh giá sai số của đa thức nội suy.

2. Phương pháp nội suy Lagrange

Đa thức $p_n(x)$ có dạng như sau, được gọi là đa thức nội suy Lagrange:

$$p_n(x) = y_0L_0(x) + y_1L_1(x) + \ldots + y_nL_n(x) \qquad (3.6)$$

Với:

$$L_0(x) = \frac{(x - x_1)(x - x_2)\ldots(x - x_n)}{(x_0 - x_1)(x_0 - x_2)\ldots(x_0 - x_n)}$$

$$L_1(x) = \frac{(x - x_0)(x - x_2)\ldots(x - x_n)}{(x_1 - x_0)(x_1 - x_2)\ldots(x_1 - x_n)}$$

. . .

$$L_i(x) = \frac{(x - x_1)(x - x_2)\ldots(x - x_{i-1})(x - x_{i+1})\ldots(x - x_n)}{(x_i - x_1)(x_i - x_2)\ldots(x_i - x_{i-1})(x_i - x_{i+1})\ldots(x_i - x_n)}$$

. . .

$$L_n(x) = \frac{(x - x_0)(x - x_1)\ldots(x - x_{n-1})}{(x_n - x_0)(x_n - x_2)\ldots(x_n - x_{n-1})}$$

3. Phương pháp sai phân Newton

Phương pháp sai phân tiến Newton với khoảng chia đều

Theo phương pháp Newton tiến thì đa thức nội suy có bậc m là $p_m(x)$ với khoảng chia đều như sau:

$$p_m(x) = y_0 + \frac{\Delta y_0}{h}(x-x_0) + \frac{\Delta^2 y_0}{2h^2}(x-x_0)(x-x_1) + \ldots + \frac{\Delta^i y_0}{i!h^i}(x-x_0)(x-x_1)\ldots(x-x_{i-1}) + \ldots + \frac{\Delta^m y_0}{m!h^m}(x-x_0)(x-x_1)\ldots(x-x_{m-1})$$

Hoặc có thể biểu diễn công thức trên dưới một dạng khác bằng phép biến đổi $t = \frac{x - x_0}{h}$

-> $x = x_0 + th$:

$$p_m(x) = p_m(x_0 + th) = y_0 + (\Delta y_0)t + \frac{\Delta^2 y_0}{2} t(t-1) + \ldots$$

$$+ \frac{\Delta^i y_0}{i!} t(t-1)\ldots(t-i+1) + \ldots + \frac{\Delta^m y_0}{m!} t(t-1)\ldots(t-m+1)$$

Phương pháp sai phân tiến Newton với khoảng chia không đều

Theo phương pháp Newton tiến thì đa thức nội suy có bậc m là $p_m(x)$ với khoảng chia không đều như sau:

$$p_m(x) = a_0 + a_1(x-x_0) + a_2(x-x_0)(x-x_1) + \ldots + a_i(x-x_0)(x-x_1)\ldots(x-x_{i-1}) + \ldots$$
$$+ a_m(x-x_0)(x-x_1)\ldots(x-x_{m-1}) \quad (3.16)$$

Trong đó:

$a_0 = y_0$

$a_1 = \frac{y_1 - y_0}{x_1 - x_0} = y[x_1,x_0].$

$a_2 = \frac{y[x_2,x_1] - y[x_1,x_0]}{(x_2 - x_0)} = y[x_2, x_1, x_0]$

.....

$a_m = y[x_m,..., x_1, x_0]$

CHƯƠNG 4
TÍNH GẦN ĐÚNG NGHIỆM CỦA PHƯƠNG TRÌNH PHI TUYẾN

MỤC ĐÍCH, YÊU CẦU

Sau khi học xong chương 4, yêu cầu sinh viên:

1. Hiểu được thế nào là nghiệm và khoảng phân ly nghiệm
2. Nắm được một số phương pháp lặp để tìm nghiệm gần đúng của phương trình phi tuyến.
3. Biết vận dụng các phương pháp trên vào các bài toán thực tế.

4.1. NGHIỆM VÀ KHOẢNG PHÂN LY NGHIỆM

4.1.1. Nghiệm của phương trình một ẩn

Xét phương trình một ẩn

$$f(x) = 0 \qquad (4.1)$$

trong đó f(x) là một hàm số cho trước của đối số x.

Giá trị x_0 được gọi là nghiệm của (4.1) nếu

$$f(x_0) = 0$$

Nghiệm của phương trình (4.1) có thể là số thực hoặc số phức, nhưng ở đây ta chỉ khảo sát các nghiệm thực.

4.1.2. Sự tồn tại nghiệm của phương trình

Định lý. Nếu hàm số f(x) liên tục trên đoạn [a,b] và f(a) và f(b) trái dấu, tức là

$$f(a)f(b)<0 \qquad (4.2)$$

Thì phương trình (4.1) có ít nhất một nghiệm trong khoảng [a,b].

4.1.3. Khoảng phân ly nghiệm

Định nghĩa. Khoảng [a,b] được gọi là khoảng phân ly nghiệm của phương trình (4.1) nếu nó chứa ***một và chỉ một nghiệm*** của phương trình đó.

Định lý. Nếu hàm số f(x) ***liên tục, đơn điệu*** trên đoạn [a,b] và ***f(a)f(b)<0*** thì đoạn [a,b] là một khoảng phân ly nghiệm của phương trình (4.1).

Ví dụ. Xét phương trình

$$f(x) = x^2 - 2 = 0$$

Ta thấy hàm số f(x) liên tục, và f'(x) = 2x.

Ta xét đoạn [1,2]. Ta có f(1) = -1; f(2) = 2. Vậy f(1)f(2)<0. Hàm số f(x) liên tục và đơn điệu vì f'(x) = 2x >0 trên đoạn [1,2]. Vậy đoạn [1,2] là khoảng phân ly nghiệm của phương trình trên.

Tuy nhiên ví dụ sau đây chứng tỏ rằng điều kiện liên tục, đơn điệu chỉ là điều kiện đủ. Hàm số không đơn điệu trong một khoảng nào đó vẫn có thể chỉ có một nghiệm duy nhất.

Ví dụ. Xét phương trình

$$f(x) = x^3 - x - 1 = 0 \qquad (4.3)$$

Ta sẽ chứng tỏ rằng phương trình này có nghiệm thực và xác định khoảng phân ly nghiệm.

Ta thấy hàm số f(x) liên tục, và $f'(x) = 3x^2 - 1 = 0$ tại $x = \pm\frac{1}{\sqrt{3}}$.

Ta có bảng biến thiên sau:

x	$-\infty$		$-\frac{1}{\sqrt{3}}$		$\frac{1}{\sqrt{3}}$		$+\infty$
f'(x)		+	0	-	0	+	
f(x)	$-\infty$	↗	M	↘	m	↗	$+\infty$

trong đó $f(M) = f(-\frac{1}{\sqrt{3}}) = -\frac{1}{3\sqrt{3}} + \frac{1}{\sqrt{3}} - 1 = \frac{2}{3\sqrt{3}} - 1 < 0$

Vậy đồ thị chỉ cắt trục hoành tại một điểm duy nhất và do đó phương trình (4.3) chỉ có một nghiệm duy nhất (Mặc dù trên đoạn $[-\frac{1}{\sqrt{3}}; 2]$ hàm số không đơn điệu)

Ngoài ra theo bảng biến thiên ta có: hàm số f(x) liên tục,đơn điệu trên đoạn [1;2] và

$f(1) = 1^3 - 1 - 1 = -1 < 0$

$f(2) = 2^3 - 2 - 1 = 5 > 0$

Tức là f(1)*f(2)<0

Vậy khoảng [1,2] chính là khoảng phân ly nghiệm.

4.1.4. Về vấn đề đánh giá sai số nghiệm xấp xỉ

Cũng như các phương pháp gần đúng nói chung, khi tìm nghiệm gần đúng của phương trình siêu việt, ta thường thiết lập cả một dãy $x_0, x_1,...,x_n,...$ sao cho $x_n \to \alpha$ khi $n \to \infty$, trong đó α là nghiệm đúng của phương trình (4.1). Do giả thiết liên tục của hàm f(x) ta có

$$\lim_{n\to+\infty} f(x_n) = f(\alpha) = 0$$

Điều này có nghĩa là khi x_n khá gần α thì $f(x_n)$ khá gần $f(\alpha)$ và có thể xem $f(x_n) \approx 0$, hay x_n thực sự có thể xem là xấp xỉ của nghiệm.

Người ta thường cho trước số $\varepsilon>0$ đủ nhỏ và nếu

$$|x_n-\alpha| \le \varepsilon \qquad (4.4)$$

thì chọn x_n làm nghiệm xấp xỉ và dừng quá trình tính toán. Một câu hỏi đặt ra là với cách chọn như vậy thì $f(x_n)$ đã có thể thực sự xem là xấp xỉ của $f(\alpha)$ không, có bảo đảm rằng $|f(x_n)-f(\alpha)| = |f(x_n)|$ là khá gần 0 không? Cũng có lúc ta chỉ quan tâm là x_n xấp xỉ α tốt như thế nào thôi, như trong ví dụ áp dụng tính $\sqrt{2}$ mà ta sẽ xét đến chẳng hạn, khi đó ta không cần quan tâm đến câu hỏi này lắm. Nhưng cũng có những trường hợp ta lại quan tâm là $f(x_n)$ có thể coi là gần 0 không, (Ví dụ để có thể bỏ qua trong quá trình tính toán) thì lúc này sự xấp xỉ của x_n so với α chưa đủ, mà ta còn phải xét cả giá trị $|f(x_n)|$ nữa. Chính vì lý do này mà trong các chương trình tính toán chúng tôi đưa thêm điều kiện dừng về $f(x_n)$. Quá trình tính toán sẽ dừng nếu điều kiện (4.4) và

$$|f(x_n)| < \delta \qquad (4.5)$$

thỏa mãn.

Để bạn đọc hiểu rõ hơn những điều chúng tôi vừa trình bày, chúng ta xét ví dụ sau đây:

Ví dụ.

Ta xét 2 hàm sau đây:

$$f(x) = (3363 - 2378\sqrt{2}) - (x-1)^{10}$$

$$g(x) = (3363 - 2378x) - (\sqrt{2}-1)^{10}$$

và xét 2 phương trình

$$f(x) = 0 \qquad (4.6)$$

$$g(x) = 0 \qquad (4.7)$$

Ta có thể thấy rằng

$$f(\sqrt{2}) = g(\sqrt{2}) = 0$$

Vì không tính được $\alpha = \sqrt{2}$ nên chúng ta sẽ dùng kết quả độ chính xác gấp đôi do máy tính thực hiện bằng hàm sqrt(2). Giá trị này vào khoảng 1.41421356237309551454746218. Ta định nghĩa dãy x_n như sau

$x_0 = 1$ $\quad | x_0 - \alpha | < 0.5e+01$

$x_1 = 1.4$ $\quad | x_1 - \alpha | < 0.5e-01$

$x_2 = 1.41$ $\quad | x_2 - \alpha | < 0.5e-02$

$x_3 = 1.414$ $\quad | x_3 - \alpha | < 0.5e-03$

$x_4 = 1.4142$ $\quad | x_4 - \alpha | < 0.5e-04$

. . .

Rõ ràng dãy x_n hội tụ đến α.

Ta có bảng sau đây biểu diễn mối liên hệ giữa dãy x_n , sai số trên dãy x_n là ε_n, hàm $f(x_n)$ và hàm $g(x_n)$.

n	x_n	ε_n	$f(x_n)$	$g(x_n)$
0	1	0.5	0.0001486768	985
1	1.4	0.5e-01	0.0000438191	34
2	1.41	0.5e-02	0.0000144501	10
3	1.414	0.5e-03	0.0000007647	0.5
4	1.4142	0.5e-04	0.0000000488	0.03
5	1.41421	0.5e-05	0.0000000128	0.008
6	1.414213	0.5e-06	0.0000000002	0.001
7	1.4142135	0.5e-07	0.0000000002	0.0001

Rõ ràng nếu chỉ dựa vào độ lệch $| x_n - \alpha |$, hay trong thực tế là $| x_n - x_{n-1} |$ để chọn nghiệm xấp xỉ, ta sẽ có những kết quả rất khác biệt giữa 2 phương trình (4.6) và (4.7). Với phương trình (4.6) thì tất cả các giá trị x_n trong bảng trên đều có thể xem là nghiệm gần đúng, và như vậy $\varepsilon = 0.5$ cũng là đủ. Ngược lại nếu ta nói rằng giá trị $x_n = 1.41$ là nghiệm xấp xỉ của phương trình (4.7) với độ chính xác 0.5% thì rõ ràng không ổn, vì giá trị $g(1.41) = 10$ còn là giá trị quá lớn so với 0. Thậm chí khi ε đạt độ chính xác 0.0005 thì $g(1.4142) = 0.03$ vẫn còn lớn so với 0, nên nếu ta nói rằng 1.4142 là nghiệm xấp xỉ của (4.7) thì cũng không ổn.

Nếu bây giờ ta thay điều kiện (4.4) bằng điều kiện (4.5) thì ta không còn gặp điều phiền toái trên đây nữa. Với phương trình (4.6) ta chỉ cần chọn n rất bé là đủ, còn với phương trình (4.7) thì ta phải chọn n lớn hơn nếu muốn đạt được độ chính xác như mong muốn. Trong thực hành việc thử điều khiện (4.5) được thực hiện rất dễ dàng. Vì vậy chúng tôi nghĩ rằng trong những bài toán tìm nghiệm xấp xỉ ta nên thêm một cột $f(x_n)$, chúng ta sẽ thấy được tốc độ hội tụ đến 0 của $f(x_n)$ và vì vậy sẽ dừng bước tính toán ở thời điểm thích hợp hơn.

4.2. MỘT SỐ PHƯƠNG PHÁP LẶP GIẢI PHƯƠNG TRÌNH

4.2.1. Mở đầu

Ý tưởng chung của phương pháp lặp tìm nghiệm của phương trình (4.1) là xây dựng một dãy các số $x_0, x_1, ..., x_n, ...$ với x_0 là giá trị xuất phát sao cho

$$\lim_{n->+\infty} x_n = \alpha$$

Như vậy với n khá lớn, ta có thể xem x_n là xấp xỉ của nghiệm α.

Ta có thể đưa ra một đánh giá về sai số tổng quát cho hầu hết các phép lặp như sau:

Định lý. Với hàm f(x) liên tục và khả vi trên đoạn [a,b], ngoài ra

$$\exists\ m_1 \text{ sao cho } 0 < m_1 \le |f'(x)| \text{ với } \forall\ x \in [a,b] \qquad (4.8)$$

khi đó ta có đánh giá

$$|x_n - \alpha| \le \frac{|f(x_n)|}{m_1} \qquad (4.9)$$

4.2.2. Phương pháp chia đôi (bisection)

a. Mô tả phương pháp

Giả sử f(x) liên tục trên [a,b] và f(a), f(b) trái dấu (đoạn [a,b] không cần phải là khoảng phân ly). Như vậy trong khoảng này phải có một nghiệm α. Ta sẽ tìm nghiệm này bằng cách chia đôi khoảng [a,b], chọn khoảng con chứa nghiệm, rồi chia đôi tiếp khoảng con chứa nghiệm này cho đến khi tìm thấy nghiệm hoặc khoảng con đã đủ nhỏ để bảo đảm rằng mọi giá trị trong khoảng đó đều có thể xem là xấp xỉ nghiệm. Cụ thể trước hết ta đặt $a_0 = a$, $b_0 = b$ và cho trước một giá trị $\varepsilon > 0$ đủ nhỏ để dùng làm điều kiện xấp xỉ nghiệm và dừng quá trình tính toán. Sau đó ta thực hiện các bước sau:

- **Bước 0:** Đặt $x_0 = \dfrac{a_0 + b_0}{2}$

Vì $f(a_0)f(b_0)<0$, do đó một trong 2 trường hợp sau xảy ra:

a. $f(x_0) = 0$. Ta có x_0 là nghiệm và ***kết thúc***.

b. $f(x_0) \neq 0$. Nếu $f(a)f(x_0)<0$ thì nghiệm sẽ ở trong khoảng $[a, x_0]$ do đó ta đặt

$a_1 = a_0$, $b_1 = x_0$

Nếu $f(x_0)f(b)<0$ thì nghiệm sẽ ở trong khoảng $[x_0,b]$ do đó ta đặt

$a_1 = x_0$, $b_1 = b$

Vì nghiệm $\alpha \in [a_1,b_1]$, ta có $|x_0 - \alpha| \leq |b_1 - a_1| = \dfrac{b-a}{2}$

Chuyển sang bước 1.

- **Bước 1:** Đặt $x_1 = \dfrac{a_1 + b_1}{2}$

Vì $f(a_1)f(b_1)<0$, do đó một trong 2 trường hợp sau xảy ra:

a. $f(x_1) = 0$. Ta có nghiệm là x_1 và ***kết thúc***.

b. $f(x_1) \neq 0$.

Nếu $f(a_1)f(x_1)<0$ thì nghiệm sẽ ở trong khoảng $[a_1, x_1]$ do đó ta đặt

$a_2 = a_1$, $b_2 = x_1$

Nếu $f(x_1)f(b_1)<0$ thì nghiệm sẽ ở trong khoảng $[x_1,b_1]$ do đó ta đặt

$a_2 = x_1$, $b_2 = b_1$

Vì nghiệm $\alpha \in [a_2,b_2]$, ta có $|x_1 - \alpha| \leq |b_2 - a_2| = \dfrac{b-a}{2^2}$

Chuyển sang bước 2.

. . .

- **Bước n:** Đặt $x_n = \dfrac{a_n + b_n}{2}$

Vì $f(a_n)f(b_n)<0$, do đó một trong 2 trường hợp sau xảy ra:

a. $f(x_n) = 0$. Ta có nghiệm là x_n và ***kết thúc***.

b. $f(x_n) \neq 0$.

Nếu $f(a_n)f(x_n)<0$ thì nghiệm sẽ ở trong khoảng $[a_n, x_n]$ do đó ta đặt

$a_{n+1} = a_n$, $b_{n+1} = x_n$

Nếu $f(x_n)f(b_n)<0$ thì nghiệm sẽ ở trong khoảng $[x_n,b_n]$ do đó ta đặt

$a_{n+1} = x_n$, $b_{n+1} = b_n$

Vì nghiệm $\alpha \in [a_{n+1},b_{n+1}]$, ta có $|x_n - \alpha| \leq |b_{n+1}-a_{n+1}| = \dfrac{b-a}{2^{n+1}}$

Ta kiểm tra xem nếu $\dfrac{b-a}{2^{n+1}} \leq \varepsilon$ thì ***kết thúc***, nếu không thì

Chuyển sang bước n+1.

b. Sự hội tụ của phương pháp và sai số

Dãy $a_0,a_1,...,a_n$ đơn điệu tăng và bị chặn bởi b, bãy $b_0,b_1,...,b_n$ đơn điệu giảm và bị chặn bởi a mặt khác dãy b_n-a_n dương và giảm dần đến 0, như vậy khi n -> ∞ ta có

$$\lim_{n->+\infty} a_n = \lim_{n->+\infty} b_n = \lim_{n->+\infty} x_n = \alpha$$

Do tính liên tục của hàm f và $f(a_n)f(b_n)<0$ ta có

$$\lim_{n->+\infty} f(a_n)f(b_n) = |f(\alpha)|^2 \leq 0$$

Suy ra $f(\alpha) = 0$, hay α chính là nghiệm của phương trình (4.1).

Tóm lại xuất phát từ $a_0 = a$ và $b_0 = b$, cho n = 0,1,2,... nếu ta lấy nghiệm gần đúng là $x_n = a_n$ hoặc $x_n = b_n$ thì sai số là

$$|x_n - \alpha| \leq \frac{b-a}{2^n} \qquad (4.10)$$

Còn nếu ta lấy nghiệm gần đúng là $\dfrac{a_n + b_n}{2}$ thì sai số là

$$|x_n - \alpha| \leq \frac{b-a}{2^{n+1}} \qquad (4.11)$$

(4.9) chính là đánh giá sai số của nghiệm xấp xỉ tính bằng thuật toán đã trình bày ở trên đây.

c. Ví dụ

Ta xét phương trình $f(x) = \sin(x) - x^2\cos(x) = 0$.

Phương trình này có nghiệm đúng là x = 0.

Ta thấy nếu a= -0.5, b = 2 thì f(a) = -0.6988, f(b) = 2.5739, tức là trái dấu. Vậy ta có thể áp dụng phương pháp chia đôi. Đặt ε = 1.0e-03 ta có sau 8 bước lặp ta nhận được nghiệm là 0.0005.

d. Nhận xét về thuật toán chia đôi

Ưu điểm của phương pháp chia đôi là đơn giản. Nhược điểm là tốc độ hội tụ chậm, không tận dụng được tính chất của hàm số f(x). Dù hàm số có dạng gì thì chúng ta cũng chỉ chia đôi, xét

giá trị của hàm tại các điểm chia rồi quyết định chọn đoạn nào để chia tiếp. Nếu khoảng [a,b] ban đầu lớn thì phải khá nhiều bước mới đạt được độ chính xác cần thiết.

e. Cài đặt chương trình cho thuật toán chia đôi

Phương pháp chia đôi đòi hỏi phải biết hàm f(x) và 2 giá trị a và b là 2 điểm cận dưới và cận trên của khoảng quan sát. Trong chương trình chúng tôi dùng chương trình con chiadoi (double (*f)(double),double &x) để thực hiện thuật toán chia đôi. Ở đây chúng tôi dùng con trỏ tới hàm là f, như vậy chúng ta không phụ thuộc vào một dạng hàm cụ thể. Sau này ta có thể gọi thủ tục chia đôi cho một hàm do ta định nghĩa hoặc có sẵn, ví dụ chiadoi(sin,xx), chiadoi(g,x),... Thủ tục chiadoi được viết dưới dạng hàm. Khi chạy thủ tục này ta sẽ phải nhập giá trị a,b. Chương trình sẽ kiểm tra điều kiện f(a)f(b)<0; nếu không thỏa mãn hoặc số bước lặp đã vượt quá số bước lặp tối đa thì hàm trả về giá trị false. Trong trường hợp thành công thì hàm trả về giá trị true và giá trị x là nghiệm xấp xỉ.

Sau đây là đoạn chương trình chính thể hiện thuật toán:

```
/*Phuong phap tim nghiem bang cach chia doi lien tiep khoang [a,b]*/
int chiadoi(double (*f)(double),double a,double b,double &x,
 double &errx,double &erry,int &buoclap)
/*Phuong phap chia doi de tim nghiem f(x)=0 trong khoang trai dau [a,b].
  Neu co nghiem thi tra ve gia tri true, neu khong thi tra ve gia tri false.
  x la nghiem, errx, erry sai so tren x va tren y, buoclap la so buoc lap
  da thuc hien*/
{clrscr();
 if(f(a)==0) {x=a;errx=erry=0;buoclap=0;return true;}
 if(f(b)==0) {x=b;errx=erry=0;buoclap=0;return true;}
 if(f(a)*f(b)>0)
  {cout<<endl<<"f(a) va f(b) khong trai dau";delay(1000);return false;}
   int k=1;double c,aa,bb;aa=a;bb=b;
   kvecto xc;
   xc[0]=a;xc[1]=b;
 while(true)
 {c = (a+b)/2;
  if(f(a)*f(c)<0) b=c;else a=c;
  xc[++k]=c;
  if(b-a<epsix && fabs(f(c))<epsiy) break;//f(c) = 0
  if(k>kmax)
   {cout<<endl<<"Lap chua hoi tu sau  "<<kmax<<"  buoc";
    delay(1000);return false;
   }
  }
```

```
 x=c;
errx=b-a;erry=fabs(f(c));buoclap=k;
return true;
}
```

4.2.3. Phương pháp dây cung

a. Mô tả phương pháp

Điều kiện để thực hiện phương pháp dây cung cũng giống như phương pháp chia đôi, là f(x) liên tục trên [a,b] và f(a), f(b) trái dấu.

Về nguyên tắc phương pháp dây cung cũng giống như phương pháp chia đôi, nghĩa là dựa vào hai điểm $a_0 = a$, $b_0 = b$ ban đầu ta sẽ chọn tiếp các điểm x_n nằm trong khoảng $[a_n,b_n]$, sao cho khoảng chọn luôn luôn chứa nghiệm đúng của phương trình. Ở phương pháp chia đôi, x_n được chọn là điểm giữa của khoảng $[a_n ,b_n]$ và khoảng $[a_{n+1} ,b_{n+1}]$ tiếp theo sẽ là khoảng chứa nghiệm trong 2 khoảng con $[a_n ,x_n]$ hoặc $[x_n,b_n]$. Như vậy khoảng $[a_n,b_n]$ sẽ nhỏ dần tới 0, cho đến lúc ta có thể xem tất cả các điểm nằm trong khoảng là xấp xỉ của nghiệm.

Còn ở phương pháp dây cung giá trị x_n được chọn tiếp theo lại là giao điểm của dây cung nối 2 điểm biểu diễn đồ thị tại 2 đầu khoảng con với trục hoành.

Ta lưu ý là trong phương pháp chia đôi độ dài khoảng con $[a_n,b_n]$ tiến tới 0, nhưng trong phương pháp dây cung nói chung điều này không đúng. Có thể một trong 2 giá trị a hoặc b sẽ giữ nguyên. Giá trị này sẽ luôn đóng vai trò là a_n hoặc b_n . Ví dụ nếu b giữ nguyên và luôn luôn đóng vai trò b_n thì a_n sẽ thay đổi và chính là x_{n-1}, n=1,2,...Dãy $x_0,x_1,...,x_n,...$ là dãy đơn điệu tăng hoặc giảm và hội tụ đến nghiệm đúng. Không như ở phương pháp chia đôi, điều kiện dừng ở đây không còn là độ dài khoảng $[a_n,b_n]$, mà là độ dài khoảng $[x_n, x_{n-1}]$. Ta sẽ dừng thuật toán và xem x_n là nghiệm xấp xỉ nếu $|x_n - x_{n-1}| \le \varepsilon$ hoặc (và) $|f(x_n)| \le \delta$.

Ta biết rằng phương trình đường thẳng đi qua 2 điểm (a,f(a)) và (b,f(b) là

$$\frac{x-a}{b-a} = \frac{y-f(a)}{f(b)-f(a)}$$

Điểm cắt trục hoàng có tọa độ (c,0) do đó ta có

$$\frac{c-a}{b-a} = \frac{-f(a)}{f(b)-f(a)}$$

Từ đây

$$c = a - \frac{f(a)[b-a]}{f(b)-f(a)} = \frac{af(b)-bf(a)}{f(b)-f(a)} \qquad (4.12)$$

Trước hết ta đặt $a_0 = a$, $b_0 = b$ và cho trước một giá trị $\varepsilon > 0$ và $\delta > 0$ đủ nhỏ để dùng làm điều kiện xấp xỉ nghiệm và dừng quá trình tính toán. Ta cũng cho trước một số kmax để làm số bước lặp tối đa. Sau kmax bước mà thuật toán chưa kết thúc thì ta thông báo số bước đã quá lớn, chưa nhận được kết quả và kết thúc.

Sau đó ta thực hiện các bước sau:

- **Bước 0:** Đặt $x_0 = \dfrac{a_0 f(b_0) - b_0 f(a_0)}{f(b_0) - f(a_0)}$

Vì $f(a_0)f(b_0)<0$, do đó một trong 2 trường hợp sau xảy ra:

a. $|f(x_0)| \le \delta$. Ta có x_0 là nghiệm xấp xỉ và ***kết thúc***.

b. $f(x_0) \ne 0$.

Nếu $f(a)f(x_0)<0$ thì nghiệm sẽ ở trong khoảng $[a, x_0]$ do đó ta đặt

$a_1 = a_0$, $b_1 = x_0$

Nếu $f(x_0)f(b)<0$ thì nghiệm sẽ ở trong khoảng $[x_0,b]$ do đó ta đặt

$a_1 = x_0$, $b_1 = b$

Chuyển sang bước 1.

- **Bước 1:** Đặt $x_1 = \dfrac{a_1 f(b_1) - b_1 f(a_1)}{f(b_1) - f(a_1)}$

Vì $f(a_1)f(b_1)<0$, do đó một trong 2 trường hợp sau xảy ra:

a. Nếu $|x_1 - x_0| \le \varepsilon$ và $|f(x_1)| \le \delta$. Ta có nghiệm là x_1 xấp xỉ và ***kết thúc***.

b. $f(x_1) \ne 0$.

Nếu $f(a_1)f(x_1)<0$ thì nghiệm sẽ ở trong khoảng $[a_1, x_1]$ do đó ta đặt

$a_2 = a_1$, $b_2 = x_1$

Nếu $f(x_1)f(b_1)<0$ thì nghiệm sẽ ở trong khoảng $[x_1,b_1]$ do đó ta đặt

$a_2 = x_1$, $b_2 = b_1$

Chuyển sang bước 2.

. . .

- **Bước n:** Đặt $x_n = \dfrac{a_n f(b_n) - b_n f(a_n)}{f(b_n) - f(a_n)}$

Vì $f(a_n)f(b_n)<0$, do đó một trong 2 trường hợp sau xảy ra:

a. Nếu $|x_n - x_{n-1}| \le \varepsilon$ và $|f(x_n)| \le \delta$ Ta có nghiệm xấp xỉ là x_n và ***kết thúc***.

b. $f(x_n) \ne 0$.

Nếu $f(a_n)f(x_n)<0$ thì nghiệm sẽ ở trong khoảng $[a_n, x_n]$ do đó ta đặt

$a_{n+1} = a_n$, $b_{n+1} = x_n$

Nếu $f(x_n)f(b_n)<0$ thì nghiệm sẽ ở trong khoảng $[x_n,b_n]$ do đó ta đặt

$a_{n+1} = x_n$, $b_{n+1} = b_n$

Nếu $n > kmax$ thì thông báo và ***kết thúc.***

Chuyển sang bước n+1.

b. Sự hội tụ của phương pháp và đánh giá sai số

Về ý tưởng thì phương pháp dây cung rất đơn giản và dễ hiểu. Tuy nhiên để khảo sát một cách chặt chẽ thì ta phải phân ra một số điều kiện hàm lồi hoặc lõm trong khoảng [a,b]. Ở đây chúng ta sẽ không đi quá vào các chi tiết này.

Ta viết lại công thức (4.10)

$$c = a - \frac{f(a)[b-a]}{f(b)-f(a)} = a - \frac{f(a)[a-b]}{f(a)-f(b)}$$

vì khi tính c thì a và b đều bình đẳng nên

$$c = b - \frac{f(b)[b-a]}{f(b)-f(a)}$$

Bây giờ để đơn giản ta chỉ xét trường hợp hàm f(x) lồi (f''(x)<0) hoặc lõm (f''(x)>0) trên đoạn [a,b]. Trong trường hợp này một trong 2 điểm a hoặc b sẽ được cố định. Nếu ta gọi giá trị cố định này là d và giá trị còn lại là x_0 (tức là nếu d = a thì x_0 = b, nếu d = b thì x_0 = a) thì các giá trị x_n được tính theo công thức:

$$x_n = x_{n-1} - \frac{f(x_{n-1})[x_{n-1}-d]}{f(x_{n-1})-f(d)} \qquad (4.13)$$

n = 0,1,2,...

d và x_0 được chọn cụ thể trong các trường hợp như sau:

(1) Nếu f(x) là hàm lồi trên [a,b], tức là f''(x) < 0, ta chọn d cùng dấu với f''(x). Trong trường hợp (a) thì d =a, x_0 = b. Trong trường hợp (b) thì d =b và x_0=a.

(2) Nếu f(x) là hàm lõm trên [a,b], tức là f''(x) > 0, ta cũng chọn d cùng dấu với f''(x). Trong trường hợp (c) thì d =a, x_0 = b. Trong trường hợp (d) thì d =b và x_0=a.

Vậy ta luôn luôn chọn d cùng dấu với f''(x).

Có thể chứng minh rằng dãy x_n đơn điệu tăng hoặc giảm và bị chặn, do đó hội tụ đến giá trị α. Từ (4.11) ta có:

$$\alpha = \alpha - \frac{f(\alpha)[\alpha-d]}{f(\alpha)-f(d)} \qquad (4.14)$$

Từ đây suy ra f(α) =0 hay α chính là nghiệm của phương trình (4.1).

Ngoài công thức đánh giá sai số chung cho phương pháp lặp (4.9), nếu thêm điều kiện về f'(x) ta có thể đánh giá sai số của nghiệm gần đúng x_n thông qua 2 gần đúng liên tiếp x_n và x_{n-1}.

Định lý.

Giả sử f'(x) liên tục và không đổi dấu trên [a,b] và thỏa mãn:

$$\exists\ m_1, M_1 \text{ sao cho } 0 < m_1 \le |f'(x)| \le M_1 < \infty \text{ với } \forall\ x \in [a,b] \qquad (4.15)$$

khi đó ta có :

$$|x_n - \alpha| \le \frac{M_1 - m_1}{m_1} |x_n - x_{n-1}| \qquad (4.16)$$

c. Ví dụ

Ta xét phương trình $f(x) = \sin(x) - x^2\cos(x) = 0$.

Phương trình này có nghiệm đúng là x = 0.

Ta thấy nếu a= -0.5, b = 2 thì f(a) = -0.6988, f(b) = 2.5739, tức là trái dấu. Vậy ta có thể áp dụng phương pháp dây cung. Đặt ε = 1.0e-03 ta có sau 3 bước lặp ta nhận được nghiệm là 0.0008.

Vậy trong trường hợp này ta thấy phương pháp dây cung hội tụ nhanh hơn phương pháp chia đôi. Điều này cũng dễ hiểu, vì trong phương pháp chia đôi ta không hề tận dụng tính đặc thù của hàm f(x). Nghĩa là cho dù hàm f(x) kiểu gì thì ta cũng chỉ chia đôi khoảng [a,b]. Còn phương pháp dây cung thì chúng ta đã tận dụng thông tin về giá trị của hàm tại 2 đầu của đoạn chứa nghiệm.

d. Nhận xét thuật toán dây cung

Thuật toán dây cung là một trong những phương pháp được sử dụng rộng rãi để tính gần đúng nghiệm thực của phương trình siêu việt. Ưu điểm của nó là thuật toán đơn giản. Nhược điểm là tuy có nhanh hơn thuật toán chia đôi nhưng vẫn còn hội tụ chậm, chỉ hội tụ tuyến tính.

e. Cài đặt chương trình cho thuật toán dây cung

Sau đây là đoạn chương trình chính thể hiện (mô tả) thuật toán dây cung

```
/*Phuong phap tim nghiem bang phuong phap day cung tren khoang [a,b]*/
/*Phuong phap day cung de tim nghiem f(x)=0 trong khoang trai dau [a,b].
  Neu co nghiem thi tra ve gia tri true, neu khong thi tra ve gia tri false.
  x la nghiem, errx, erry sai so tren x va tren y, buoclap la so buoc lap
  da thuc hien*/
int daycung(double (*f)(double),double a,double b,double &x,
 double &errx,double &erry,int &buoclap)
{clrscr();
 if(f(a)==0) {x=a;errx=erry=0;buoclap=0;return true;}
 if(f(b)==0) {x=b;errx=erry=0;buoclap=0;return true;}
 if(f(a)*f(b)>0)
  {cout<<endl<<"f(a) va f(b) khong trai dau";delay(1000);return false;}
 int k=1;double c,cp,aa,bb;aa=a;bb=b;//cp=c previouse
 kvecto xa,xb;
 xa[0]=a;xb[0]=b;
 c = (a*f(b)-b*f(a))/(f(b)-f(a));
 if(f(a)*f(c)<0) b=c;else a=c;
 xa[1]=a;xb[1]=b;
 while(true)
  {cp=c;
   c = (a*f(b)-b*f(a))/(f(b)-f(a));
   if(f(a)*f(c)<0) b=c;else a=c;
   k++;
   xa[k]=a;xb[k]=b;
```

```
    if(fabs(c-cp)<epsix && fabs(f(c))<epsiy) break;
    if(k>kmax)
     {cout<<endl<<"Lap chua hoi tu sau  "<<kmax<<"  buoc";
      delay(1000);return false;
     }
    }
   x=c;
   errx=fabs(c-cp);erry=fabs(f(c));buoclap=k;
  }
```

4.2.4. Phương pháp lặp đơn

a. Mô tả phương pháp

Giả sử phương trình (4.1) có nghiệm trong khoảng [a,b] và ta biến đổi được về dạng tương đương

$$x = \varphi(x) \qquad (4.17)$$

Tiếp theo ta chọn một giá trị $x_0 \in [a,b]$ làm giá trị xấp xỉ ban đầu rồi tính dần các nghiệm xấp xỉ x_n theo quy tắc

$$x_n = \varphi(x_{n-1}),\ n=1,2,... \qquad (4.18)$$

Phương pháp này gọi là phương pháp lặp và hàm φ gọi là hàm lặp.

b. Điều kiện hội tụ của phương pháp và đánh giá sai số

Điều kiện hội tụ của phương pháp lặp:

Định lý. Giả sử phương pháp lặp (4.17) và (4.18) thỏa mãn các điều kiện sau:

(1) [a,b] là khoảng phân ly nghiệm α của phương trình (4.17)

(2) Mọi x_n tính theo (4.18) đều $\in [a,b]$.

(3) Hàm $\varphi(x)$ có đạo hàm $\varphi'(x)$ và thỏa mãn: $|\varphi'(x)| \le q < 1,\ x \in (a,b)$

Khi đó phương pháp lặp (4.18) hội tụ, tức là: $x_n \to \alpha$ khi $n \to \infty$

Chứng minh.

Chúng ta sẽ dùng đến công thức Lagrange như sau:

Cho hàm số g(x) liên tục trên [a,b] và có đạo hàm trong (a,b). Khi đó tồn tại $c \in (a,b)$ sao cho $g(b)-g(a) = g'(c)(b-a)$.

Bây giờ ta chứng minh định lý trên.

Vì α là nghiệm của (4.4) ta có: $\alpha = \varphi(\alpha)$

theo công thức lặp thì $x_n = \varphi(x_{n-1})$

Trừ 2 đẳng thức này vế với vế ta có

$$\alpha - x_n = \varphi(\alpha) - \varphi(x_{n-1}) \qquad (4.19)$$

Áp dụng công thức Lagrange vào vế phải của (4.19) ta có

$$\alpha - x_n = \varphi'(c)(\alpha - x_{n-1})$$

Từ đây suy ra

$$| \alpha - x_n | = | \varphi'(c) | |\alpha - x_{n-1}| \leq q |\alpha - x_{n-1}| \quad (4.20)$$

Vì bất đẳng thức này đúng với mọi n, do đó cho n giảm dần đến 1 ta có

$$| \alpha - x_n | = | \varphi'(c) | |\alpha - x_{n-1}| \leq q^n |\alpha - x_0| \to 0 \text{ khi } n \to \infty. \quad (4.21)$$

Vậy ta có $| \alpha - x_n | \to 0$ khi $n \to \infty$, là điều cần chứng minh.

Đánh giá sai số phương pháp lặp:

Ta có thể dùng công thức (4.21) để đánh giá sai số của phương pháp lặp đơn. Tuy nhiên trong thực tế giá trị α lại chưa biết nên ta phải thay khoảng $[\alpha,x_0]$ bằng khoảng [a,b], và như vậy

$$| \alpha - x_n | \leq q^n | b - a | \quad (4.22)$$

Tuy nhiên công thức này phụ thuộc vào a, b và không sát với thực tế, nhất là khi khoảng [a,b] lớn.

Sau đây chúng ta sẽ đưa ra một cách đánh giá khác, sát với thực tế hơn.

Từ (4.20) ta có

$$| \alpha - x_n | \leq q |\alpha - x_{n-1}| = q | \alpha - x_n + x_n - x_{n-1}| \leq q | \alpha - x_n | + q | x_n - x_{n-1}|$$

Vậy

$$(1-q) | \alpha - x_n | \leq q | x_n - x_{n-1}|$$

Hay

$$| \alpha - x_n | \leq \frac{q}{1-q} | x_n - x_{n-1}| \quad (4.23)$$

Mặt khác vì

$$| \varphi'(x) | \leq q < 1, x \in (a,b)$$

Do đó

$$| x_n - x_{n-1} | = | \varphi(x_{n-1}) - \varphi(x_{n-2})| = | \varphi'(c) | | x_{n-1} - x_{n-2}| \leq q | x_{n-1} - x_{n-2}|$$

Từ đây suy ra

$$| x_n - x_{n-1} | \leq q^{n-1} | x_1 - x_0|$$

Tức là $| x_n - x_{n-1} |$ hội tụ đến 0.

Kết hợp với (4.22) ta có

$$| \alpha - x_n | \leq \frac{q}{1-q} | x_n - x_{n-1}| \leq q^n \frac{1}{1-q} | x_1 - x_0| \quad (4.24)$$

Áp dụng (4.24) ta có thể đánh giá sai số $| x_n - \alpha |$ qua các giá trị lặp vừa tính. Về sau này khi xét điều kiện để dừng quá trình lặp người ta thường đánh giá qua $|x_n - x_{n-1}|$. Nếu sai số cho phép là ε thì quá trình lặp sẽ dừng nếu $|x_n - x_{n-1}| \leq \varepsilon$.

Ví dụ.

Ta xét phương trình:

$$f(x) = x^3 - x - 1 = 0 \quad (4.25)$$

Theo ví dụ ở phần (4.1.3) hàm số này luôn đơn điệu, liên tục trên [1;2] và f(1) = -1, f(2) =5. Vậy khoảng [1,2] chính là khoảng phân ly nghiệm. Ta sẽ dùng phương pháp lặp để tính gần đúng nghiệm α của nó. Ta sẽ biến đổi (4.25) về dạng (4.17). Ví dụ ta có thể viết

$$\varphi(x) = x^3 - 1$$

Nhưng lúc đó

$$\varphi'(x) = 3x^2 \geq 3 \text{ tại mọi } x \in [1,2]$$

Với cách chọn như vậy phương pháp lặp không có hy vọng hội tụ. Điều này có thể thấy rõ trong bảng sau:

n	$x_0 = 1$ $x_n = \varphi(x_{n-1})$
0	1
1	0
2	-1
3	-2
4	-9
5	-730

Bây giờ ta viết (4.25) dưới dạng sau

$$x^3 = x + 1$$

$$x = (x+1)^{1/3}$$

$$\varphi(x) = (x+1)^{1/3}$$

Lúc đó

$$\varphi'(x) = (1/3)(x+1)^{-2/3} = (1/3)(1/\sqrt[3]{(x+1)^2}) \leq 1/3 \text{ tại mọi } x \in [1,2]$$

Ngoài ra $\varphi'(x) > 0$, như vậy hàm $\varphi(x)$ thỏa mãn định lý trên đây do đó ta có thể thực hiện quá trình lặp với giá trị bắt đầu là $x_0 = 1$ chẳng hạn.Ta có bảng giá trị như sau:

n	$x_0 = 1$ $x_n = \varphi(x_{n-1})$
0	1
1	1.2599
2	1.3122
3	1.3223
4	1.3242
5	1.3246

Ta có thể lấy nghiệm xấp xỉ là 1.3246

4.2.5. Phương pháp Newton-Rapson hay còn gọi là phương pháp tiếp tuyến

a. Mô tả phương pháp

Điều kiện để thực hiện phương pháp Newton có phần chặt hơn 2 phương pháp ta vừa khảo sát trên đây. Ta giả thiết rằng hàm f(x) trái dấu tại 2 vị trí a và b, đồng thời tồn tại đạo hàm cấp một $f'(x) \neq 0$ trong khoảng [a,b], và đạo hàm cấp 2 tại $x \in (a,b)$.

Ý chủ đạo của phương pháp Newton là thay phương trình (4.1) phi tuyến đối với x bằng phương trình gần đúng, tuyến tính đối với x.

Trước hết ta nhắc lại định lý về khai triển Taylo của một hàm như sau:

Định lý. Cho hàm số f(x) xác định và có đạo hàm đến cấp n+1 tại x_0 và lân cận của x_0. Giả sử h là một giá trị sao cho $x_0 + h$ cũng thuộc lân cận này. Ta có công thức sau đây được gọi là khai triển Taylor bậc n của f(x) tại x_0:

$$f(x_0 + h) = f(x_0) + \frac{h}{1!}f'(x_0) + \frac{h^2}{2!}f''(x_0) + \ldots + \frac{h^n}{n!}f^{(n)}(x_0) + \frac{h^{n+1}}{(n+1)!}f^{(n+1)}(c)$$

Trong đó $c \in (x_0, x_0+h)$

Dựa vào khai triển Taylo, ta sẽ xác định một hàm $\varphi(x)$ và tìm nghiệm của (4.1) bằng phép lặp:

$$x_{n+1} = \varphi(x_n)$$

Giả sử x là nghiệm đúng của (4.1), còn x_n là nghiệm xấp xỉ tại bước lặp thứ n. Ta đặt $x = x_n + \Delta x_n$. Theo khai triển Taylo ta có

$$f(x) = f(x_n + \Delta x_n) = f(x_n) + \Delta x_n f'(x_n) + \frac{\Delta x_n^{\,2}}{2!}f''(c) = 0$$

Nếu Δx_n đủ nhỏ thì ta có công thức gần đúng:

$$f(x_n) + \Delta x_n f'(x_n) \approx f(x) = 0$$

Từ đây

$$\Delta x_n \approx -\frac{f(x_n)}{f'(x_n)}$$

Vì $\Delta x_n = x - x_n$

Do đó

$$x \approx x_n - \frac{f(x_n)}{f'(x_n)}$$

Và ta suy ra công thức lặp cho phép lặp Newton:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \qquad (4.25)$$

Về ý nghĩa hình học x_{n+1} chính là giao điểm của tiếp tuyến đường cong y = f(x) tại điểm $(x_n, f(x_n))$ với trục hoành. Do đó phương pháp này còn được gọi là phương pháp tiếp tuyến.

Từ điểm $(x_n,f(x_n))$ ta vẽ tiếp tuyến của đồ thị $y = f(x)$. Phương trình đồ thị này là

$$y = f(x_n) + f'(x_n)(x-x_n).$$

Giả sử đường tiếp tuyến này cắt trục hoành tại x_{n+1}, ta có

$$0 = f(x_n) + f'(x_n)(x_{n+1}-x_n)$$

Từ đây suy ra

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

Công thức này cũng chính là công thức (4.25) ở trên.

b. Điều kiện hội tụ của phương pháp Newton và đánh giá sai số

Định lý. Điều kiện đủ để phương pháp tiếp tuyến hội tụ:

Giả sử những điều kiện sau đây thỏa mãn:

$f(a)f(b) < 0$, tức là giá trị hàm $f(x)$ trái dấu tại hai đầu đoạn $[a,b]$.

Hàm $f(x)$ có đạo hàm bậc nhất và bậc 2 $f'(x)$ và $f''(x)$, với $f(x)$ và $f'(x)$ liên tục trên $[a,b]$, ***f' và f'' không đổi dấu*** trong (a,b) (tức là hàm $f(x)$ đơn điệu, lồi hoặc lõm trong đoạn $[a,b]$).

Xấp xỉ đầu x_0 được chọn $\in [a,b]$, sao cho ***$f(x_0)$ cùng dấu với $f''(x)$***, tức là $f(x_0)f''(x) > 0$ (hàm lồi thì chọn phía giá trị hàm âm, hàm lõm thì chọn phía giá trị dương).

Khi đó dãy x_n được định nghĩa bởi (4.25) sẽ hội tụ tới α.

Đánh giá sai số của nghiệm gần đúng.

Ngoài công thức đanh giá sai số (4.9), nếu thêm điều kiện về $f''(x)$ ta có thể đánh giá sai số của nghiệm gần đúng x_n thông qua 2 gần đúng liên tiếp x_n và x_{n-1}.

Định lý.

Giả sử $f'(x)$ liên tục và không đổi dấu trên $[a,b]$ và thỏa mãn:

$\exists\ m_1, M_2$ dương sao cho $m_1 \le |f'(x)|$; $f''(x) \le M_2$ với $\forall\ x \in [a,b]$ (4.28)

khi đó ta có

$$|x_n - \alpha| \le \frac{M_2}{2m_1}|x_n - x_{n-1}|^2 \qquad (4.29)$$

Chứng minh.

Dùng công thức khai triển Taylor cho $f(x_n)$ tại x_{n-1} ta có

$$f(x_n) = f(x_{n-1}) + \frac{x_n - x_{n-1}}{1!}f'(x_{n-1}) + \frac{(x_n - x_{n-1})^2}{2!}f''(c) \qquad (4.30)$$

trong đó $c \in (x_{n-1},x_n)$

Theo (4.25)

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

Từ đây

$$f(x_{n-1}) + (x_n - x_{n-1})f'(x_{n-1}) = 0$$

Thay vào (4.30) ta có

$$f(x_n) = \frac{(x_n - x_{n-1})^2}{2!} f''(c)$$

Như vậy theo (4.27) và (4.28)

$$|x_n - \alpha| \le \frac{|f(x_n)|}{m_1} = \frac{(x_n - x_{n-1})^2}{2m_1} f''(c) \le \frac{M_2}{2m_1} |x_n - x_{n-1}|^2 \qquad (4.31)$$

Là điều cần chứng minh.

c. Ví dụ về phương pháp Newton

Tính $\sqrt{2}$ bằng cách giải phương trình sau:

$$f(x) = x^2 - 2 = 0 \qquad (4.32)$$

Giải:

Ta thấy f(1) = -1, f(2) = 2, như vậy điều kiện 1) thỏa mãn.

f'(x) = 2x > 2 với mọi $x \in [1,2]$

f''(x) = 2 > 1 với mọi $x \in [1,2]$, vậy điều kiện 2) thỏa mãn

Vì f(2) = 2, nên ta chọn x_0 =2, như vậy thì f(2)f''(x) = 2.2 = 4 >0 và điều kiện 3) thỏa mãn.

Vậy ta có thể áp dụng phương pháp lặp Newton để tính nghiệm xấp xỉ của phương trình (4.32). Ta có bảng sau

n	$x_0 = 2$ $x_{n+1} = x_n - f(x)/f'(x_n)$
0	2
1	1.5
2	1.417
3	1.41421

Ta có thể lấy nghiệm xấp xỉ là 1.41421. Ta biết rằng $\sqrt{2}$ = 1.414213562, như vậy phương pháp lặp Newton hội tụ rất nhanh.

d. Nhận xét về phương pháp Newton

Nhờ việc sử dụng đạo hàm của hàm số f(x) nên nói chung phương pháp Newton hội tụ nhanh hơn phương pháp chia đôi và phương pháp dây cung. Tuy nhiên việc kiểm tra điều kiện để áp dụng phương pháp Newton phức tạp hơn. Những điều kiện để phương pháp Newton hội tụ là quan trọng và cần thiết phải kiểm tra khi áp dụng phương pháp này. Ví dụ tương ứng với hình vẽ sau đây chỉ ra rằng có trường hợp nếu áp dụng các phương pháp chia đôi hoặc dây cung thì quá trình lặp sẽ hội tụ, còn nếu ta áp dụng phương pháp Newton nhưng chọn điểm xuất phát không thích hợp thì không đạt được kết quả như mong muốn.

e. Chương trình minh họa phương pháp Newton (tiếp tuyến)

Sau đây là đoạn chính của chương trình thể hiện (mô tả) phương pháp newton :

```
/*Phuong phap tim nghiem bang phuong Newton tren khoang [a,b]*/
/*Phuong phap Newton de tim nghiem f(x)=0 trong khoang trai dau [a,b].
  Neu co nghiem thi tra ve gia tri true, neu khong thi tra ve gia tri false.
  x la nghiem, errx, erry sai so tren x va tren y, buoclap la so buoc lap
  da thuc hien*/
int ttuyen(double (*f)(double),double (*f1)(double),double a,double b,
double &x,double &errx,double &erry,int &buoclap)
{clrscr();
 if(f(a)==0) {x=a;errx=erry=0;buoclap=0;return true;}
 if(f(b)==0) {x=b;errx=erry=0;buoclap=0;return true;}
 if(f(a)*f(b)>0)
  {cout<<endl<<"f(a) va f(b) khong trai dau";delay(1000);return false;}
 int k=0;
 kvecto xb;double xp,xn;//previouse x;
 xb[0]=b;
 xn=b;
 while(true)
  {if(f1(xn)<epsiy1)
    {cout<<endl<<"Dao ham qua be";delay(1000);return false;}
  xp=xn;
  xn = xn-f(xn)/f1(xn);
  k++;
  xb[k]=xn;
  if(fabs(xn-xp)<epsix && fabs(f(xn))<epsiy) break;
  if(k>kmax)
   {cout<<endl<<"Lap chua hoi tu sau  "<<kmax<<"  buoc";
    delay(1000);return false;
   }
  }
 errx=fabs(xn-xp);erry=fabs(f(xn));buoclap=k;
 x=xn;
}
```

4.3. BÀI TẬP

Bài 1. Hãy mô tả phương pháp chia đôi để tìm nghiệm gần đúng của phương trình phi tuyến.

Bài 2. Hãy mô tập hương pháp dây cung để tìm nghiệm gần đúng của phương trình phi tuyến.

Bài 3. Hãy mô tả phương pháp tiếp tuyến để tìm nghiệm gần đúng của phương trình phi tuyến.

Bài 4. Hãy mô tả phương pháp lặp đơn để tìm nghiệm gần đúng của phương trình phi tuyến.

Bài 5. Giải gần đúng phương trình:

$$x-1/2\sin x = 0,25$$

bằng phương pháp lặp với 4 lần lặp.

Bài 6. Dùng phương pháp chia đôI tính nghiệm gần đúng của phương trình: x^3-x-1 qua 4 bước lặp. Đánh giá sai số $|x_4-\alpha|$

Bài 7. Dùng phương pháp dây cung tính nghiệm gần đúng của phương trình: x3-x-1 qua 4 bước lặp. Đánh giá sai số $|x4-\alpha|$

Bài 8. Dùng phương pháp chia đôi tính gần đúng $\sqrt{5}$ qua 4 bước lặp. Đánh giá sai số $|x4-\sqrt{5}|$

Bài 9. Dùng phương pháp lặp hãy tính gần đúng nghiệm dương lớn nhất của phương trình:

$x^3 - x - 1000 = 0$

với sai số tuyệt đối không lớn hơn 10^{-5}.

Bài 10. Thử lại hoặc viết mới các chương trình cài đặt các thuật toán rồi chạy thử để kiểm tra các kết quả trên đây.

TÓM TẮT NỘI DUNG CHƯƠNG 4

Trong chương này chúng ta cần chú ý nhất là các vấn đề sau:

1. Khoảng phân ly nghiệm

Định nghĩa. Khoảng [a,b] được gọi là khoảng phân ly nghiệm của phương trình (4.1) nếu nó chứa ***một và chỉ một nghiệm*** của phương trình đó.

Định lý. Nếu hàm số f(x) ***liên tục, đơn điệu*** trên đoạn [a,b] và ***f(a)f(b)<0*** thì đoạn [a,b] là một khoảng phân ly nghiệm của phương trình (4.1)

2.Phương pháp chia đôi (bisection):

- Phương pháp:

+ Xác định khoảng phân ly nghiệm [a,b]

+Tính giá trị của nghiệm gần đúng ở bước lặp thứ i trên khoảng phân ly nghiệm $[a_i,b_i]$ (i=0,1,2...) theo công thức $x_i= (a_i+b_i)/2$ với $a_0=a;b_0=b$. Sau đó là xác định khoảng phân ly mới cho nghiệm ở bước mới thứ i+1 là $[a_{i+1},b_{i+1}]$. Cứ tiếp tục phép lặp như thế cho đến khi nào thoả mãn điều kiện dừng của phương pháp.

- Đánh giá sai số:

Giả sử ở bước lặp cuối cùng là bước thức n (i=n) ta đã xác định được nghiệm gần đúng x_n. Khi đó sai số được đánh giá như sau:$|x_n - \alpha| \le \frac{|a-b|}{2^{n+1}}$. (Trong đó α là nghiệm đúng của phương trình (4.1)

3.Phương pháp dây cung

- Phương pháp:

+ Xác định khoảng phân ly nghiệm [a,b]

+Tính giá trị của nghiệm gần đúng ở bước lặp thứ i trên khoảng phân ly nghiệm $[a_i,b_i]$ (i=0,1,2...) theo công thức $x_i= \frac{a_i f(b_i) - b_i f(a_i)}{f(bi) - f(a_i)}$ với $a_0=a;b_0=b$. Sau đó là xác định khoảng phân ly mới cho nghiệm ở bước mới thứ i+1 là $[a_{i+1},b_{i+1}]$. Cứ tiếp tục phép lặp như thế cho đến khi nào thoả mãn điều kiện dừng của phương pháp.

- Đánh giá sai số:

Giả sử ở bước lặp cuối cùng là bước thức n (i=n) ta đã xác định được nghiệm gần đúng x_n. Khi đó sai số được đánh giá như sau: $|x_n - \alpha| \le \frac{|f(x_n)|}{m_1}$. (Trong đó α là nghiệm đúng của phương trình (4.1), và $0 < m_1 \le |f'(x)|$ với $\forall x \in [a,b]$). Hoặc ta có đánh giá sai số thức 2 là:$|x_n - \alpha| \le \frac{M_1 - m_1}{m_1} |x_n - x_{n-1}|$.

4 Phương pháp lặp đơn

- Phương pháp:

+Xác định khoảng phân ly nghiệm [a,b] của (4.1), phương trình (4.1) chuyển về dạng x = φ(x), với điều kiện :|φ'(x)| ≤q<1 trên (a,b).Tiếp theo ta chọn một giá trị $x_0 \in [a,b]$ làm giá trị xấp xỉ ban đầu rồi tính dần các nghiệm xấp xỉ x_i theo quy tắc:

$x_i = \varphi(x_{i-1})$, i=1,2,.. cho tới khi nào thảo mãn điều kiện dừng của phương pháp.

- Đánh giá sai số:

Giả sử ở bước lặp cuối cùng là bước thức n (i=n) ta đã xác định được nghiệm gần đúng x_n. Khi đó sai số được đánh giá như sau: $|x_n - \alpha| \le q^n | b - a |$ hoặc $|x_n - \alpha| \le q^n \frac{1}{1-q} | x_1 - x_0|$.

5 Phương pháp tiếp tuyến

- Phương pháp:

+ Xác định khoảng phân ly nghiệm [a,b] của phương trình (4.1).

+Tính giá trị của nghiệm gần đúng ở bước lặp thứ i theo công thức

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}.$$

Xấp xỉ đầu x_0 được chọn ∈ [a,b], sao cho ***f(x₀) cùng dấu với f''(x)***, tức là $f(x_0)f''(x) > 0$ (hàm lồi thì chọn phía giá trị hàm âm, hàm lõm thì chọn phía giá trị dương).

- Đánh giá sai số :

Giả sử ở bước lặp cuối cùng là bước thức n (i=n) ta đã xác định được nghiệm gần đúng x_n. Khi đó sai số được đánh giá như sau: $|x_n - \alpha| \le \frac{|f(x_n)|}{m_1}$ hoặc

$$|x_n - \alpha| \le \frac{M_2}{2m_1} |x_n - x_{n-1}|^2$$

CHƯƠNG 5
TÍNH GẦN ĐÚNG ĐẠO HÀM VÀ TÍCH PHÂN XÁC ĐỊNH

MỤC ĐÍCH, YÊU CẦU

Sau khi học xong chương 5, yêu cầu sinh viên:

1. Hiểu và nắm được thế nào là bài toán tính gần đúng đạo hàm và tích phân xác định

2. Nắm được các phương pháp tính gần đúng đạo hàm, qua đó biết cách tính giá trị gần đúng đạo hàm cho một hàm bất kỳ.

3. Nắm được các phương pháp tính gần đúng tích phân xác định, qua đó biết cách tính giá trị gần đúng tích phấn xác định của một hàm bất kỳ

4. Biết cách áp dụng các phương pháp tính gần đúng trên vào việc giải các bài toán ngoài thực tế.

5. Biết cách đánh giá sai số của từng phương pháp.

5.1 TÍNH ĐẠO HÀM

Người ta thường dùng một số phương pháp để tính gần đúng đạo hàm của hàm f(x) tại x trong đó hai phương pháp sau đây thường được dùng nhất:

5.1.1. Áp dụng đa thức nội suy

Giả sử người ta phải tính xấp xỉ đạo hàm của hàm số f(x) trên đoạn (a,b). Trước hết người ta thay hàm f(x) bằng đa thức nội suy p(x), sau đó lấy đạo hàm p'(x) và coi là xấp xỉ của đạo hàm f'(x).

Ví dụ.

Giả sử ta xác định được đa thức nội suy là:

$$p_3(x) = 8x^3 - 29x + 5$$

Khi đó đạo hàm:

$p_3'(x) = 24x^2 - 29$ được xem là xấp xỉ của f'(x).

5.1.2. Áp dụng công thức Taylor

Theo công thức Taylor ta có

$$f(x+h) = f(x) + \frac{h}{1!}f'(x) + \frac{h^2}{2!}f''(c)$$

$$c = x + \theta h, \quad 0 < \theta < 1$$

Khi $|h|$ khá bé thì có thể bỏ qua số hạng h^2

$$f(x+h) - f(x) \approx hf'(x)$$

Vậy ta có: $f'(x) \approx \dfrac{f(x+h) - f(x)}{h}$

Đây cũng chính là định nghĩa của đạo hàm. Vậy cách dùng khai triển Taylor cũng chính là cách dùng định nghĩa đạo hàm.

Chương trình minh họa

Sau đây là đoạn chương trình chính thể hiện (mô tả) phương pháp tính gần đúng đạo hàm bằng phương pháp nội suy

```
/*Noi suy dung da thuc Vandermon roi tinh dao ham*/
/*Tra ve gia tri da thuc noi suy tai x; avan[i] la cac he so cua da
  thuc giai truc tiep tu ma tran Vandermon, xqs[I] la
  cac diem quan sat*/
double poli(double x) //Tinh da thuc bang phuong phap Horner
 {int i;double s;
  s=avan[nqs];
  for(i=nqs-1;i>=0;i--)  s= s*x+avan[i];
  return s;
 }
//================================================
/*Tra ve dao ham gia tri da thuc noi suy tai x; avan[i] la cac he so cua da
  thuc giai truc tiep tu ma tran Vandermon, xqs[i] la
  cac diem quan sat*/
double poli1(double x) //Tinh da thuc bang phuong phap Horner
 {int i;double s;
  s=nqs*avan[nqs];
  for(i=nqs-1;i>0;i--)  s= s*x+i*avan[i];
  return s;
 }
//================================================
/*Noi suy bang cach giai truc tiep he phuong trinh tuyen tinh voi
  ma tran Vandermon */
void nsvandermon(double *a)
 {int i,j,k,n1;kmatran aa;kvecto b;
  //Tinh ma tran Vandermon
  for(i=0;i<=nqs;i++)
```

```
 {aa[i][0]=1;
  for(j=1;j<=nqs;j++)
   aa[i][j]=aa[i][j-1]*xqs[i];
 }
for(i=0;i<=nqs;i++) aa[i][nqs+1]=yqs[i];
 gjordan(aa,a,nqs);
}
```

5.2. TÍNH GẦN ĐÚNG TÍCH PHÂN XÁC ĐỊNH

5.2.1. Mô tả bài toán

Xét tích phân xác định của một hàm số f(x) trong khoảng [a,b]

$$I = \int_a^b f(x)dx \qquad (5.1)$$

Nếu hàm f(x) liên tục trên [a,b] và có nguyên hàm F(x), thì I có thể tính một cách đơn giản thông qua công thức Newton-Leibniz:

$$I = \int_a^b f(x)dx = F(b) - F(a) \qquad (5.2)$$

Tuy nhiên trong thực tế thì chúng ta thường gặp trường hợp hàm f(x) không có nguyên hàm hoặc nguyên hàm quá phức tạp không thể xác định được. Trong những trường hợp này người ta phải tính gần đúng (5.1). Có nhiều cách để tính gần đúng tích phân, ví dụ có thể dùng ngay định nghĩa của tích phân

$$I = \lim_{n->+\infty} \sum_{i=0}^{n-1} f(x_i)\Delta x_i \qquad (5.3)$$

Tuy nhiên tổng Darboux hội tụ rất chậm, do đó để đạt được độ chính xác cao đòi hỏi một khối lượng tính toán rất lớn. Do đó trong thực tế người ta hầu như không dùng (5.3) để tính xấp xỉ tích phân.

Sau đây là một số phương pháp tính gần đúng tích phân hay được dùng. Ý tưởng cơ bản của phương pháp này là chia nhỏ khoảng [a,b] cần lấy tích phân, sau đó trên mỗi khoảng nhỏ này ta xấp xỉ hàm số bằng một đa thức. Với các đa thức ta có thể dùng nguyên hàm của chúng để tính tích phân, sau đó ta cộng các tích phân thành phần để được xấp xỉ của tích phân toàn thể.

5.2.2. Công thức hình thang

a. Mô tả phương pháp

Ta chia đoạn [a,b] thành n đoạn con bằng nhau:

$a = x_0 < x_1 < ... < x_n = b$

$x_i = a + ih,\ h = \dfrac{b-a}{n}$

$i = 0,1,2,... ,n$

Thay diện tích hình thang cong bằng diện tích hình thang thẳng ta được

$$\int_{x1}^{x2} f(x)dx \approx \; h\frac{y_1 + y_2}{2} \tag{5.4}$$

Thực chất của (5.4) là ta đã thay hàm f(x) bằng hàm nội suy

$$p(x) = y_0 + \frac{\Delta y_0}{h}(x\text{-}x_0) = y_0 + \Delta y_0 \frac{x - x_0}{h} \tag{5.5}$$

Đặt $t = \frac{x - x_0}{h}$, hay $x = x_0 + th$ ta có $dx = hdt$

$$\int_{x1}^{x2} p(x)dx = \int_0^1 (y_0 + t\Delta y_0)hdt = h\,(ty_0 + \frac{t^2}{2}\Delta y_0)\Big|_{t=0}^{t=1} =$$

$$= h(y_0 + \frac{1}{2}\Delta y_0) = h\,\frac{y_0 + y_1}{2}$$

Như vậy

$$I = \int_a^b f(x)dx \approx I^* = \frac{b-a}{2n}(y_0 + 2y_1 + \ldots + 2y_{n-1} + y_n) =$$

$$= h(\frac{y_0 + y_n}{2} + y_1 + \ldots + y_{n-1}) \tag{5.6}$$

b. Đánh giá sai số

Định lý.

Giả sử hàm số $y = f(x)$ có đạo hàm cấp 2 liên tục và

$$|f''(x)| \le M_2 \,,\; x \in [a,b] \tag{5.7}$$

khi đó ta có đánh giá

$$|I - I^*| \le \frac{M_2}{12} h^2(b-a) \tag{5.8}$$

c. Ví dụ

Hãy tính gần đúng tích phân

$$I = \int_0^1 (1/(1+x^2))dx$$

Ta đã biết giá trị đúng của tích phân này là $\pi/4$. Như vậy $I \approx 0.78539816$

Ta sẽ tính gần đúng I bằng công thức hình thang rồi so sánh kết quả.

Chia đoạn [0,1] thành n = 10, đoạn con bằng nhau, tức là h = 0.1; ta tính ra bảng sau:

n	x	y
0	0.0	1.0000000
1	0.1	0.9900990
2	0.2	0.9615385
3	0.3	0.9174312
4	0.4	0.8620690
5	0.5	0.8000000
6	0.6	0.7352941
7	0.7	0.6711409
8	0.8	0.6097561
9	0.9	0.5524862
10	1.0	0.5000000

Áp dụng công thức hình thang ta được:

$$I= h(\frac{y_0 + y10}{2} + y_1 + \ldots + y_9)$$

Thay giá trị từ bảng trên vào ta có:

$I \approx 0.7849815$ với sai số tương đối là 0.054%.

d. Chương trình minh họa

Thuật toán được thực hiện trong chương trình có khác chút ít so với thuật toán đã trình bày ở trên. Xuất phát từ n=1, h=b-a, ta sẽ tăng n lên gấp đôi tại mỗi bước tính toán. Quá trình tính toán sẽ dừng lại nếu sự khác biệt của tích phân xấp xỉ ở bước hiện tại so với bước trước đó nhỏ hơn một số epsilon cho trước. Ta sẽ phân tổng tích phân thành 3 tổng s0,s1 và s2. Tổng s0 = (f(a)+f(b))/2; mỗi lần tăng n lên gấp đôi thì ta chỉ cần tính lại tổng s2 ở các vị trí 1,3,5,...,n-1. Tổng s1 là tổng của các giá trị hàm tại các điểm không phải là đầu mút. Sau khi tính lại s2, ta tính lại s1 bằng phép gán

s1=s1+s2

Và tổng xấp xỉ của tích phân là

$I_n = h(s0+s1)$

Sau đây là đoạn chính của chương trình thể hiện (mô tả) thuật toán

```
/*Phuong phap tinh xap xi tich phan bang phuong phap hinh thang
 tren khoang [a,b]*/
/*Phuong phap hinh thang tinh tich phan xac dinh trong khoang [a,b].
  Bien gttp  la gia tri xap xi cua tich phan tinh duoc.
  Tra ve gia tri true neu da dat duoc do chinh xac*/
int hinhthang(double (*f)(double),double a,double b,double &gttp,
```

```
double &err,int &khoangchia)
{clrscr();
double s0,s1,s2,h,tp,tp1;int k,nkc,i;
kvecto x;
nkc=1;
h=b-a;
s0=(f(a)+f(b))/2;
s1=0;
tp=(s0+s1)*h;
do
 {tp1=tp;
  nkc=nkc*2;
  h=h/2;
  s2=0;//Bat dau tinh tong tai cac diem moi
  for(i=1;i<nkc;i+=2) s2+=f(a+i*h);
  s1=s1+s2;
  tp=h*(s0+s1);
  if(nkc>nmax)
   {cout<<endl<<"Tich phan chua hoi tu voi  "<<nmax<<"  khoang chia";
    delay(1000);return false;
   }
  }
 while(fabs(tp-tp1)>epsi);
 err=fabs(tp-tp1);khoangchia=nkc;gttp=tp;
 return(true);
}
```

5.2.3. Công thức parabol (Simpson)

a. Mô tả phương pháp

Ta chia đoạn [a,b] thành 2n đoạn con bằng nhau

$a = x_0 < x_1 < ... < x_{2n} = b$

$x_i = a + ih,\ h = \dfrac{b-a}{2n}$

$y_i = f(x_i)$

$i = 0,1,2,\ldots,2n$

Để tính tích phân (5.1) coi khoảng nối 3 điểm liên tiếp nhau là 1 đoạn (như vậy qua 2n+1 điểm ta có n đoạn), đoạn thứ i(i =0,1,...,n) gồm các điểm x_{2i}, x_{2i+1}, x_{2i+2}, và trong mỗi đoạn con ta dùng đa thức nội suy bậc 2 $p_2(x)$.

Giả sử các điểm của một đoạn con là x_0,x_1,x_2 và các giá trị f(x) tương ứng là y_0, y_1, y_2 , ta có:

$$\int_{x0}^{x2} f(x)dx \approx \int_{x0}^{x2} p_2(x)dx \qquad (5.9)$$

trong đó

$$p_2(x) = y_0 + \frac{\Delta y_0}{h}(x-x_0) + \frac{\Delta^2 y_0}{2h^2}(x-x_0)(x-x_1)$$

$$= y_0 + \Delta y_0 \frac{x-x_0}{h} + \Delta^2 y_0 \frac{(x-x_0)(x-x_1)}{2h^2} \qquad (5.10)$$

Đặt $t = \frac{x-x_0}{h}$, hay $x = x_0 + th$ ta có $dx = hdt$,

nếu $x = x_0$ thì $t = 0$, $x=x_2$ thì $t=2$.

Như vậy

$$\int_{x0}^{x2} f(x)dx \approx \int_0^2 p_2(x)dx = h\int_0^2 (y_0+t\Delta y_0 + \frac{t(t-1)}{2}\Delta^2 y_0)dt =$$

$$= h(ty_0+\frac{1}{2}t^2\Delta y_0 + \frac{1}{2}(t^3/3 - t^2/2)\Delta^2 y_0) \Big|_{t=0}^{t=2} =$$

$$= h(2y_0+2\Delta y_0 + \frac{1}{2}(8/3 - 4/2)\Delta^2 y_0) = \frac{h}{3}(y_0 + 4y_1 + y_2)$$

Tính tích phân xấp xỉ cho từng đoạn $[x_0,x_2]$, $[x_2,x_4]$, ... ,$[x_{2n-2},x_{2n}]$ và cộng lại ta có

$$\int_a^b f(x)dx \approx I^* = \frac{h}{3}[(y_0 +y_{2n}) + 4(y_1 +y_3 + \ldots + y_{2n-1}) + 2(y_2 +y_4 + \ldots + y_{2n-2})] \qquad (5.11)$$

$$h= \frac{(b-a)}{2n}$$

Công thức (5.11) được gọi là công thức Simpson.

b.Đánh giá sai số

Định lý.

Giả sử hàm số y = f(x) có đạo hàm cấp 4 liên tục và

$$| f^{(4)}(x) | \le M_4 , x \in [a,b] \qquad (5.12)$$

khi đó ta có đánh giá

$$| I - I^* | \le \frac{M_4}{180} h^4(b-a) \qquad (5.13)$$

c. Ví dụ

Hãy tính gần đúng tích phân

$$I = \int_0^1 (1/(1+x^2))dx$$

Ta đã biết giá trị đúng của tích phân này là $\pi/4$. Như vậy $I \approx 0.78539816$

Ta sẽ tính gần đúng I bằng công thức Simson rồi so sánh kết quả.

Chia đoạn [0,1] thành 2n = 4 đoạn con bằng nhau, với h=0.25, ta tính ra bảng sau:

i	x_i	$y_i=f(x_i)$
0	0	1
1	0.25	0.941176
2	0.5	0.8
3	0.75	0.64
4	1	0.5

Theo công thức Simpson ta có

$I = (h/3)*(y_0 + y_4 + 4y_1 + 4y_3 + 2y_2)$. Thay các giá trị ở bảng trên vào ta có

$= (0.25/3)*(1 + 3.76471 + 1.6 + 2.56000 + 0.5) \approx 0.785399$

d. Chương trình minh họa

Sau đây là đoạn chương trình chính thể hiện (mô tả) thuật toán:

```
//SIMSON.CPP
/*Phuong phap tinh xap xi tich phan bang phuong phap Simson
 (pp cau phuong) tren khoang [a,b]
  Bien gttp la gia tri xap xi cua tich phan tinh duoc.
  Tra ve gia tri true neu da dat duoc do chinh xac*/
int simson(double (*f)(double),double a,double b,double &gttp,
 double &err,int &khoangchia)
{clrscr();
 double s0,s1,s2,h,tp,tp1;int k,nkc,i;
 kvecto x;
 nkc=1;
 h=b-a;
 h=b-a;
 s0=f(a)+f(b);
 s1=0;
 s2=0;
```

```
tp=(s0+2*s1+4*s2)*h/3;
do
 {tp1=tp;
  s1=s1+s2;
  nkc=nkc*2;
  h=h/2;
  s2=0;//Bat dau tinh tong tai cac diem moi
  for(i=1;i<nkc;i+=2) s2=s2+f(a+i*h);
  tp=(s0+2*s1+4*s2)*h/3;
  if(nkc>nmax)
   {cout<<endl<<"Tich phan chua hoi tu voi  "<<nmax<<" khoang chia";
    delay(1000);return false;
   }
 }
while(fabs(tp-tp1)>epsi);
err=fabs(tp-tp1);khoangchia=nkc;gttp=tp;
return true;
}
```

5.3. BÀI TẬP

Bài 1. Cho hàm số y = logx với số các giá trị tại x = 50; 55; 60; 65 tuần tự là 1,6990; 1,7404; 1,7782; 1,8129. Hãy tính đạo hàm của y tại x = 50 và so sánh với kết quả trực tiếp.

Bài 2. Cho tích phân

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

Hãy chia đoạn [0,1] thành n = 10 đoạn con bằng nhau rồi tính gần đúng I và cho đánh giá sai số bằng:

a) Công thức hình thang

b) Công thức Simson

Bài 3. Cho tích phân

$$I = \int_0^1 \frac{\sin x}{x} dx$$

a) Hỏi phải chia đoạn [0,1] thành mấy (n = ?) đoạn con bằng nhau để khi tính gần đúng I bằng công thức hình thang bảo đảm được sai số tuyệt đối < 3.10^{-4}

b) Với n ấy khi tính theo công thức Simson thì sai số là bao nhiêu?

c) Hãy tính I với n đã chọn ở trên bằng công thức hình thang và công thức Simson đến 6 chữ số lẻ thập phân.

Bài 4. Thử lại hoặc viết mới các chương trình cài đặt các thuật toán rồi chạy thử để kiểm tra các kết quả trên đây.

TÓM TẮT NỘI DUNG CHƯƠNG 5

Trong chương này chúng ta cần chú ý nhất là các vấn đề sau:

1. Tính gần đúng đạo hàm

Người ta thường dùng một số phương pháp để tính gần đúng đạo hàm của hàm f(x) tại x trong đó hai phương pháp sau đây thường được dùng nhất: Áp dụng đa thức nội suy và áp dụng công thức Taylor.

2. Tính gần đúng tích phân

- Công thức hình thang:

  + Công thức:

$$I = \int_a^b f(x)dx \approx I^* = = h(\frac{y_0 + y_n}{2} + y_1 + \ldots + y_{n-1})$$

với $h= \frac{(b-a)}{n}$

  + Đánh giá sai số

  Giả sử hàm số $y = f(x)$ có đạo hàm cấp 2 liên tục và

  $| f''(x) | \le M_2 , x \in [a,b]$

  khi đó ta có đánh giá

$$| I - I^*| \le \frac{M_2}{12} h^2(b-a)$$

- Công thức parabol (Simpson)

  + Công thức:

$$I=\int_a^b f(x)dx \approx I^* = \frac{h}{3}[(y_0 + y_{2n}) + 4(y_1 + y_3 + \ldots + y_{2n-1}) + 2(y_2 + y_4 + \ldots + y_{2n2})]$$

  với $h= \frac{(b-a)}{2n}$

  + Đánh giá sai số

  Giả sử hàm số $y = f(x)$ có đạo hàm cấp 4 liên tục và

  $| f^{(4)}(x) | \le M_4 , x \in [a,b]$

  khi đó ta có đánh giá

$$| I - I^* | \le \frac{M_4}{180} h^4(b-a)$$

CHƯƠNG 6
GIẢI GẦN ĐÚNG PHƯƠNG TRÌNH VI PHÂN

MỤC ĐÍCH, YÊU CẦU

Sau khi học xong chương 3, yêu cầu sinh viên:

1. Hiểu được vai trò và tầm quan trọng của bài toán giải gần đúng phương trình vi phân.
2. Nắm được các phương pháp tìm nghiệm gần đúng của phương trình vi phân.
3. Biết cách áp dụng các phương pháp trên vào việc giải quyết các bài toán thực tế.
4. Biết cách đánh giá sai số của từng phương pháp.

6.1. MỞ ĐẦU

Nhiều bài toán khoa học kỹ thuật dẫn về việc tìm nghiệm phương trình vi phân thỏa mãn một số điều kiện nào đó. Những phương trình vi phân mô tả những hệ cơ học, lý học, hóa học, sinh học nói chung rất phức tạp, không hy vọng tìm lời giải đúng.

Trong chương này ta nghiên cứu bài toán đơn giản nhất của phương trình vi phân là ***bài toán Cauchy đối với phương trình vi phân cấp 1*** như sau:

Hãy tìm hàm y=y(x) thỏa mãn:

$$y'(x) = f(x,y) \quad x \in [a,b], \; x_0 = a \qquad (6.1)$$

$$y(x_0) = y_0 \qquad (6.1b)$$

Điều kiện (6.1b) được gọi là điều kiện ban đầu hay điều kiện Cauchy.

Tương tự, bài toán Cauchy đối với phương trình vi phân cấp n được mô tả như sau:

Hãy tìm hàm y=y(x) thỏa mãn:

$$y^{(n)} = f(x,y,y',y^{(2)},...,y^{(n-1)}) \qquad (6.2)$$

$$y(x_0) = \alpha_0,\; y'(x_0) = \alpha_1,\; y^{(2)}(x_0) = \alpha_2,\; ...,\; y^{(n-1)}(x_0) = \alpha_{n-1}$$

trong đó f() là một hàm đã biết của n+1 đối số $x,y,y',y^{(2)},...,y^{(n-1)}$; x_0, b, α_0, α_1 ..., α_{n-1} là những số cho trước.

(6.1) còn được mở rộng cho hệ thống các phương trình vi phân cấp một với bài toán Cauchy chư sau:

$$\begin{aligned} y_1' &= f_1(x,y_1, y_2,..., y_n) \\ y_2' &= f_2(x,y_1, y_2,..., y_n) \\ &... \\ y_n' &= f_n(x,y_1, y_2,..., y_n) \end{aligned} \qquad (6.3)$$

$y_1(x_0) = \alpha_1, y_2(x_0) = \alpha_2, \ldots, y_n(x_0) = \alpha_n$

$x \in [a,b], x_0 = a$

Nếu đặt

$\vec{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_n]^T$

$\vec{y} = [y_1, y_2, \ldots, y_n]^T$

$\vec{y} = [y'_1, y'_2, \ldots, y'_n]^T$

$\vec{f} = [f_1, f_2, \ldots, f_n]^T$

Bài toán (6.3) có thể viết gọn hơn dưới dạng vectơ như sau:

$\vec{y} = \vec{f}(x, \vec{y}), \quad x \in [a,b], \quad x_0 = a$

$\vec{y}(x_0) = \vec{\alpha}$

Ghi chú. Phương trình vi phân cấp n có thể đưa về hệ các phương trình vi phân cấp một bằng phép biến đổi

$y_1 = y, y_2 = y', \ldots, y_i = y^{(i-1)}, \ldots, y_n = y^{(n-1)}$

Nói chung có hai nhóm phương pháp để giải các phương trình vi phân thường:

Phương pháp tìm nghiệm chính xác: bằng cách dựa vào cách tính tích phân trực tiếp, xác định được dạng tổng quát của nghiệm rồi dựa vào điều kiện ban đầu để xác định nghiệm riêng cần tìm.

Phương pháp tìm nghiệm gần đúng xuất phát từ điều kiện ban đầu. Phương pháp này có thể áp dụng cho một lớp phương trình vi phân rộng hơn rất nhiều so với phương pháp trực tiếp, do đó được dùng nhiều trong thực tế.

Trong phần tiếp theo ta sẽ tập trung vào nhóm phương pháp thứ hai.

6.2. PHƯƠNG PHÁP EULER

Trở lại bài toán

$$y'(x) = f(x,y) \quad x \in [a,b], \quad x_0 = a \qquad (6.4)$$

$y(x_0) = y_0$

Cách giải gần đúng (6.4) là tìm các giá trị gần đúng y_i của giá trị đúng $y(x_i)$ tại các điểm x_i, i = 0,1,2,... n, trong đó

$a = x_0 < x_1 < \ldots < x_n = b$

$x_i = x_0 + ih, \; i=0,1,\ldots,n-1$

$$h = \frac{b-a}{n}$$

Ta đã biết $y_0 = \alpha_0$, ta sẽ lần lượt xác định y_1 tại x_1, rồi y_2 tại x_2, và nói chung từ giá trị gần đúng y_i tại x_i ta sẽ tính y_{i+1} tại x_{i+1}.

Phương pháp Euler cũng như một vài phương pháp sẽ được trình bày sẽ dựa vào giả thiết sau đây (cho dù giả thiết này nói chung không thể kiểm tra được)

Giả thiết rằng bài toán (6.4) có nghiệm duy nhất y = y(x), $x \in [a,b]$, $a = x_0$, và nghiệm y(x) đủ trơn, nghĩa là nó có đạo hàm đến cấp đủ cao. (6.5)

Ta khai triển Taylo nghiệm y(x) của (6.4) tại x_i

$$y(x) = y(x_i) + \frac{x - x_i}{1!} y'(x_i) + \frac{(x - x_i)^2}{2!} y''(c_i), \quad c_i \in (x_i, x) \tag{6.5}$$

Thay $x = x_{i+1} = x_i + h$, $y'(x_i) = f(x_i, y(x_i))$ vào đẳng thức trên ta có

$$y(x_{i+1}) = y(x_i) + h\, f(x_i, y(x_i)) + \frac{h^2}{2} y''(c_i), \quad c_i \in (x_i, x_{i+1}) \tag{6.6}$$

Bỏ qua số hạng cuối cùng bên phải, đồng thời thay các giá trị đúng $y(x_{i+1})$, $y(x_i)$, $f(x, y(x_i))$ bằng các giá trị xấp xỉ y_{i+1}, y_i, $f(x, y_i)$ vào (6.6) ta có

$$y_{i+1} = y_i + h\, f(x_i, y_i) \tag{6.7}$$

Với giá trị $y_0 = y(x_0) = \alpha_0$ ban đầu (như vậy y_0 là giá trị đúng của $y(x_0)$), ta có thể tính tiếp các giá trị y_i, i =1,2, ..., n.

Công thức (6.7) được gọi là công thức Euler. Công thức này cho ta cách tính y_{i+1} khi đã biết y_i mà không phải giải một phương trình nào. Vì vậy phương pháp Euler là một phương pháp hiện.

Sai số địa phương của phương pháp Euler là

$$R_i(h) = y(x_i) - y_i = \frac{h^2}{2!} y''(c_{i-1}) = O(h^2) \tag{6.8}$$

Người ta chứng minh được rằng: sai số của phương pháp Euler tại điểm x_i là

$$| y_i - y(x_i) | \le Mh \tag{6.9}$$

trong đó M là hằng số không phụ thuộc h.

Điều này chứng tỏ rằng khi $h \to 0$ thì $y_i \to y(x_i)$ tại mọi điểm x_i cố định. Tuy nhiên việc xác định giá trị M rất phức tạp. Vì vậy trong thực hành người ta thường làm như sau: Quá trình tính được chia làm nhiều bước, khoảng cách giữa các điểm x_i và x_{i+1} ở bước sau sẽ là một nửa khoảng cách của bước trước, tức là $h_{k+1} = h_k/2$. Nếu gọi lần đầu tiên với $h = \frac{b-a}{n}$ ta gọi là lần thứ 0 với n+1 điểm chia thì tại lần thứ k+1 sẽ có $n.2^{k+1} + 1$ điểm chia. Trong số điểm chia này thì có $n.2^k + 1$ điểm chia trùng với lần thứ k. Các điểm trùng đó là 0,2,4,..., $n.2^{k+1}$. Tại các điểm chia này ta có ở lần thứ k giá trị xấp xỉ của y_i là $y_i^{(k)}$, còn ở lần thứ k+1 thì giá trị xấp xỉ giá trị y tại vị trí này lại là $y_{2i}^{(k+1)}$. Ta xét đại lượng sau

$$\text{maxdiff} = \max_i | y_{2i}^{(k+1)} - y_i^{(k)} |, \; i = 0,1,2,\ldots, n.2^k$$

Và sẽ dừng quá trình tính toán nếu maxdiff nhỏ hơn giá trị ε khá nhỏ cho trước.

Ta có thể mô tả thuật toán Euler để cài đặt trên máy tính theo các bước sau:

a.Thuật toán cho một lần chia khoảng duy nhất.

Nhập a, b , y_0 và n.

Đặt $h = \frac{b-a}{n}$, $x_0 = a$.

Với i =1,2,..., n tính

$y_i = y_{i-1} + hf(x_{i-1},y_{i-1})$

$x_i = x_{i-1} + h$

b.Thuật toán cho nhiều lần chia khoảng.

- **Bước 0:** Nhập a, b , y_0 , kmax và $\varepsilon>0$.

Đặt $h_0 = \dfrac{b-a}{n}$, $x_0^{(0)} = a$, $y_0^{(0)} = y_0$

Tính $y_i^{(0)} = y_{i-1}^{(0)} + hf(x_{i-1}^{(0)}, y_{i-1}^{(0)}), x_i^{(0)} = x_{i-1}^{(0)} + h_0$, i = 1,2,...,n

- **Bước 1:**

Đặt $h_1 = \dfrac{h_0}{2}$, $x_0^{(1)} = a$, $y_0^{(1)} = y_0$

Tính $y_i^{(1)} = y_{i-1}^{(1)} + hf(x_{i-1}^{(1)}, y_{i-1}^{(1)}), x_i^{(1)} = x_{i-1}^{(1)} + h_1$, i = 1,2,...,n.2

$d_1 = \max\limits_i | y_{2i}^{(1)} - y_i^{(0)} |$, i=0,1,2,...,n

Nếu $d_1 < \varepsilon$ thì dừng thuật toán và lấy mẫu (x_0,y_0), (x_1,y_1), ..., (x_{2n},y_{2n}) làm nghiệm xấp xỉ.

Nếu 2 điều trên đây không xẩy ra thì chuyển qua bước (3).

...

- **Bước k:**

Đặt $h_k = \dfrac{h_{k-1}}{2}$, $x_0^{(k)} = a$, $y_0^{(k)} = y_0$

Tính $y_i^{(k)} = y_{i-1}^{(k)} + hf(x_{i-1}^{(k)}, y_{i-1}^{(k)}), x_i^{(k)} = x_{i-1}^{(k)} + h_k$, i = 1,2,...,n.2^k

$d_k = \max\limits_i | y_{2i}^{(k)} - y_i^{(k-1)} |$, i=0,1,2,...,n.$2^{k-1}$

Nếu $d_k < \varepsilon$ thì dừng thuật toán và lấy mẫu (x_0,y_0), (x_1,y_1), ..., (x_N,y_N), trong đó $N = n.2^k$ làm nghiệm xấp xỉ.

Nếu $k \geq kmax$ thì thông báo phép lặp chưa hội tụ và cũng dừng thuật toán.

Nếu 2 điều trên đây không xẩy ra thì chuyển qua bước (k+1).

c.Chương trình cài đặt thuật toán Euler:

```
//EULER.CPP
/*Phuong phap Euler giai gan dung phuong trinh vi phan y'=f(x,y)*/
#include "zheader.cpp"
#define kmax 17
const double epsi=1.0E-02;
double g(double,double);//a=0,b=1
double yg(double);//La nghiem dung y=y(x) cua phuong trinh y'=g(x,y)
```

```
int euler(double (*f)(double,double),double (*yf)(double));
/*Phuong phap Euler giai phuong trinh vi phan tren [a,b].
  y'(x)=f(x,y), y(x0)=y0.
  Neu buoc lap vuot qua kmax thi tra ve false
  Ham yf la nghiem dung de so sanh*/
//=================================================
double g(double x,double y)
 {return x*y/2;
 }
//=================================================
double yg(double x)
 {return exp(x*x/4);
 }
//=================================================
//Ham de tim y(x).
int euler(double (*f)(double,double),double (*yf)(double))
 {clrscr();
  double a,b,h,maxdiff,x[kmax],y[kmax],y1[kmax];int n,m,i;
  cout<<endl<<"Can duoi: ";cin>>a;
  cout<<"Can tren: ";cin>>b;
  cout<<"Dieu kien ban dau: y(a) = ";cin>>y[0];
  n=1;
  h=b-a;
  x[0]=a;
  y[1]=y[0]+h*f(x[0],y[0]);
  do
   {for(i=0;i<=n;i++) y1[i]=y[i];//Gan y1=y de bat dau tinh y moi
    m=n;
    n=n*2;
    h=h/2;
    for(i=0;i<n;i++)
     {x[i+1]=x[i]+h;
      y[i+1]=y[i]+h*f(x[i],y[i]);
     }
    maxdiff=0;
```

```
 for(i=0;i<=m;i++)
  maxdiff=maxdiff<fabs(y[2*i]-y1[i])?fabs(y[2*i]-y1[i]):maxdiff;
 if(maxdiff>epsi&&2*n>kmax-1)
  {cout<<endl<<"Chua hoi tu voi  "<<n<<"  khoang chia";
   delay(1000);break;
  }
 }
while(maxdiff>epsi);

cout<<endl<<" i    x[i]     y[i]       y(x[i])";
for(i=0;i<=n;i++)
 {cout<<endl<<setw(4)<<i;
  cout<<setw(10)<<setprecision(2)<<x[i];
  cout<<setw(10)<<setprecision(4)<<y[i];
  cout<<setw(10)<<setprecision(4)<<yf(x[i]);
 }
cout<<endl<<endl<<"So khoang chia :";
cout<<setw(10)<<n;
cout<<endl<<"Sai so |y[i]-y(x[i])| cuc dai:";
cout<<setw(10)<<setprecision(4)<<maxdiff;
//================Ghi vao tep EULER.DAT
FILE *fp;
fp=fopen("EULER.DAT","wt");
fprintf(fp," i    x[i]     y[i]       y(x[i])");
for(i=0;i<=n;i++)
 {fprintf(fp,"\n%4d",i);
  fprintf(fp,"%10.2f",x[i]);
  fprintf(fp,"%10.4f",y[i]);
  fprintf(fp,"%10.4f",yf(x[i]));
 }
fprintf(fp,"\n\nSo khoang chia : %4d",n);
fprintf(fp,"\nSai so |y[i]-y(x[i])| cuc dai: %10.4f",maxdiff);
fclose(fp);
getch();
clrscr();
```

```
  if(n>kmax) return false;
   else  return true;
 }
//================================================
void main()
 {char mchon;
  while(true)
   {clrscr();
   gotoxy(20,2);cout<<"PHUONG PHAP EULER";
   gotoxy(2,2);
   cout<<endl<<" 1. Giai phuong trinh vi phan ";
   cout<<endl<<" z. Ket thuc";
   cout<<endl<<" Nhan phim  1 -> z de chon";
   mchon=toupper(getch());
   if(mchon=='Z') break;
   switch(mchon)
    {case '1':clrscr();
              euler(g,yg);
              break;
    }
   gotoxy(2,22);
   cout<<"Nhan phim bat ky de tiep tuc";
   getch();
   }
}
```

Nhận xét.

Ưu điểm của phương pháp Euler là tính toán đơn giản, nhưng nhược điểm là độ chính xác thấp. Khi ta giảm bước h thì độ chính xác tăng lên và sai số tích lũy cũng tăng lên. Để giảm sai số tích luỹ, trong chương trình tính toán trên máy tính ta nên dùng các đại lượng có độ chính xác gấp đôi.

Phương pháp Eurler có thể áp dụng cho hệ thống phương trình vi phân cấp một (6.3).

Sử dụng các ký hiệu như trong (6.3) ta có công thức Euler giải bài toán Cauchy có dạng sau:

$$\vec{y}_0 = \vec{y}(x_0) = \alpha$$

$$\vec{y}_{i+1} = \vec{y}_i + h\,\vec{f}(x_i,\vec{y}_i),\ i=0,1,2,...,n-1 \qquad (6.10)$$

Đối với hệ thống hai phương trình vi phân cấp một:

$$y' = f_1(x,y,z)$$

$$z' = f_2(x,y,z)$$

với điều kiện ban đầu:

$$y(x_0) = \alpha_1 \ , \ z(x_0) = \alpha_2$$

Công thức (6.8) có dạng sau:

$$y_0 = y(x_0) = \alpha_1 \ , \ z_0 = z(x_0) = \alpha_2$$

$$y_{i+1} = y_i + hf_1(x_i, y_i, z_i)$$

$$z_{i+1} = z_i + hf_2(x_i, y_i, z_i) \quad i=0,1,...,n-1 \qquad (6.11)$$

Ví dụ.

Cho hệ thống phương trình vi phân cấp một:

$$y' = (z-y)x$$

$$z' = (z+y)x$$

Với điều kiện ban đầu: $y(0)=z(0)=1$.

Hãy tìm nghiệm gần đúng bằng phương pháp Euler trên khoảng [0,0.6] với bước h=0.1.

Giải:

Ta có $x_i = 0.1i$, i=0,1,2,...,6.

Xuất phát từ $y(0)=z(0)=1$, áp dụng (6.11) ta nhận được kết quả tính toán sau:

i	x_i	y_i	$f_1(x_i, y_i, z_i)$	$hf_1(x_i, y_i, z_i)$	z_i	$f_2(x_i, y_i, z_i)$	$hf_2(x_i, y_i, z_i)$
0	0	1.0000	0	0	1.0000	0	0
1	0.1	1.0000	0	0	1.0000	0.2000	0.0200
2	0.2	1.0000	0.0040	0.0004	1.0200	0.4040	0.0404
3	0.3	1.0004	0.0180	0.0018	1.0604	0.6182	0.0618
4	0.4	1.0022	0.0480	0.0048	1.1222	0.8498	0.0850
5	0.5	1.0070	0.1001	0.0100	1.2072	1.1071	0.1107
6	0.6	1.0170			1.3179		

6.3.PHƯƠNG PHÁP EULER CẢI TIẾN

Để tăng độ chính xác của phương pháp Euler người ta làm như sau:

Theo công thức Newton-Lepnitz, ta có

$$y(x2) - y(x1) = \int_{x1}^{x2} y'(x)dx$$

Tính gần đúng tích phân xác định ở vế phải bằng công thức hình thang ta có:

$$\int_{x1}^{x2} y'(x)dx \approx \frac{h}{2} [y'(x1)+y'(x2)] = \frac{h}{2} [f(x1,y(x1)) + f(x2,y(x2)], \text{ trong đó } h=x2-x1$$

Thay x1 bằng x_i, x2 bằng x_{i+1}, ta có

$$y_{i+1} = y_i + \frac{h}{2}\,[f(x_i,y_i) + f(x_{i+1},y_{i+1})]\ ,\ i=0,1,...,n-1 \qquad (6.12)$$

với $y_0 = \alpha$.

Người ta đã chứng minh được rằng sai số của (6.12) tại điểm x_i là:

$$| y_i - y(x_i) | \le Mh^2$$

trong đó M là hằng số không phụ thuộc h.

Vậy công thức (6.12) chính xác hơn công thức Euler. Tuy nhiên nó có nhược điểm là y_{i+1} xuất hiện cả ở vế phải. Như vậy khi đã biết y_i ta vẫn còn phải giải một phương trình đại số phi tuyến đối với y_{i+1} (nếu f(x,y) phi tuyến đối với y). Vì vậy đây là một phương pháp ẩn. Người ta đã cải tiến phương pháp (6.12) bằng cách phối hợp (6.12) với phương pháp Euler như sau:

$y_0 = y(x_0)$ đã biết

Với i =1, 2,... n ta tính

$$z = y_{i-1} + hf(x_{i-1},y_{i-1})$$

$$y_i = y_{i-1} + \frac{h}{2}\,[f(x_{i-1},y_{i-1}) + f(x_i,z)] \qquad (6.13)$$

i=0,1,...,n-1, m=1,2,...

Công thức (6.13) được gọi là công thức Euler cải tiến. Như vậy trong (6.13) đầu tiên người ta dùng công thức Euler để ước lượng giá trị của y_i (được ký hiệu là z) và dùng z để tính

$$y_i = y_{i-1} + \frac{h}{2}\,[f(x_{i-1},y_{i-1}) + f(x_i,z)]$$

Ta có thể mô tả thuật toán ***Euler cải tiến*** để cài đặt trên máy tính theo các bước sau:

a.Thuật toán cho một lần chia khoảng duy nhất.

Nhập a, b , y_0 và n.

Đặt $h = \frac{b-a}{n}$, $x_0 = a$.

Với i =1,2,..., n tính

$$z = y_{i-1} + hf(x_{i-1},y_{i-1})$$

$$y_i = y_{i-1} + \frac{h}{2}[f(x_{i-1},y_{i-1}) + f(x_i,z)]$$

$$x_i = x_{i-1} + h$$

b.Thuật toán cho nhiều lần chia khoảng.

- **Bước 0:** Nhập a, b , y_0 , kmax và $\varepsilon>0$.

Đặt $h_0 = \frac{b-a}{n}$, $x_0^{(0)} = a$, $y_0^{(0)} = y_0$

Với i = 1, 2,... tính

$$x_i^{(0)} = x_{i-1}^{(0)} + h_0$$

$$z = y_{i-1}^{(0)} + h_0 f(x_{i-1}^{(0)}, y_{i-1}^{(0)})$$

$$y_i^{(0)} = y_{i-1}^{(0)} + \frac{h_0}{2}[f(x_{i-1}^{(0)}, y_{i-1}^{(0)}) + f(x_i^{(0)}, z)]$$

- **Bước 1:** Đặt $h_1 = \frac{h_0}{2}$, $x_0^{(1)} = a$, $y_0^{(1)} = y_0$

Với i = 1, 2,... tính

$$x_i^{(1)} = x_{i-1}^{(1)} + h_1$$

$$z = y_{i-1}^{(1)} + h_1 f(x_{i-1}^{(1)}, y_{i-1}^{(1)})$$

$$y_i^{(1)} = y_{i-1}^{(1)} + \frac{h_1}{2}[f(x_{i-1}^{(1)}, y_{i-1}^{(1)}) + f(x_i^{(1)}, z)]$$

$$d_1 = \max_i | y_{2i}^{(1)} - y_i^{(0)} |, i=0,1,2,...,n$$

Nếu $d_1 < \varepsilon$ thì dừng thuật toán và lấy mẫu (x_0,y_0), (x_1,y_1), ..., (x_{2n},y_{2n}) làm nghiệm xấp xỉ.

Nếu 2 điều trên đây không xẩy ra thì chuyển qua bước (3).

...

- **Bước k:** Đặt $h_k = \frac{h_{k-1}}{2}$, $x_0^{(k)} = a$, $y_0^{(k)} = y_0$

Với i = 1, 2,... tính

$$x_i^{(k)} = x_{i-1}^{(k)} + h_k$$

$$z = y_{i-1}^{(k)} + h_k f(x_{i-1}^{(k)}, y_{i-1}^{(k)})$$

$$y_i^{(k)} = y_{i-1}^{(k)} + \frac{h_k}{2}[f(x_{i-1}^{(k)}, y_{i-1}^{(k)}) + f(x_i^{(k)}, z)]$$

$$d_k = \max_i | y_{2i}^{(k)} - y_i^{(k-1)} |, i=0,1,2,...,n.2^{k-1}$$

Nếu $d_k < \varepsilon$ thì dừng thuật toán và lấy mẫu (x_0,y_0), (x_1,y_1), ..., (x_N,y_N), trong đó $N = n.2^k$ làm nghiệm xấp xỉ.

Nếu $k \geq k_{max}$ thì thông báo phép lặp chưa hội tụ và cũng dừng thuật toán.

Nếu 2 điều trên đây không xẩy ra thì chuyển qua bước (k+1).

6.4. PHƯƠNG PHÁP EULER-CAUCHY

Thực chất phương pháp Euler-Cauchy cũng là phương pháp Euler cải tiến. Tuy nhiên sự khác biệt là khi tính y_i ta dùng công thức (6.13) nhiều lần để đạt được độ chính xác cao hơn. Cụ thể từ $h = \frac{b-a}{n}$, $x_0 = a$ ta tính các $x_i = x_{i-1} + h$.

Từ điều kiện ban đầu $y_0 = y(x_0)$, cho trước $\delta>0$, với i = 1, 2, ... ta thực hiện thuật toán sau:

- **Bước 1:**

(a) $u = y_0 + hf(x_0,y_0)$

(b) $v = y_0 + \frac{h}{2}[f(x_0,y_0) + f(x_0,u)]$ (6.14)

Nếu $|v-u|< \delta$ thì ta chọn $y_1 = v$, ngược lại ta đặt u = v và tính lại (b).

Người ta chứng minh được rằng với h đủ bé thì quá trình lặp (6.14) hội tụ. Vì vậy nếu sau ba bốn lần lặp mà vẫn không đạt được sự trùng nhau đến mức đòi hỏi của các gần đúng liên tiếp thì cần giảm bước h và làm lại từ đầu.

. . .

- **Bước i:**

(a) $u = y_{i-1} + hf(x_{i-1},y_{i-1})$

(b) $v = y_{i-1} + \frac{h}{2}[f(x_{i-1},y_{i-1}) + f(x_i,u)]$ (6.15)

Nếu $|v-u|< \delta$ thì ta chọn $y_i = v$, ngược lại ta đặt u = v và tính lại (b).

Người ta chứng minh được rằng với h đủ bé thì quá trình lặp (6.15) hội tụ. Vì vậy nếu sau ba bốn lần lặp mà vẫn không đạt được sự trùng nhau đến mức đòi hỏi của các gần đúng liên tiếp thì cần giảm bước h và làm lại từ đầu.

Ta có thể mô tả thuật toán ***Euler - Cauchy*** để cài đặt trên máy tính theo các bước sau:

a. Thuật toán cho một lần chia khoảng duy nhất.

Nhập a, b , y_0, δ và n.

Đặt $h = \frac{b-a}{n}$, $x_0 = a$ ta tính các $x_i = x_{i-1} + h$.

Từ điều kiện ban đầu $y_0 = y(x_0)$, với i = 1, 2, ... ta thực hiện thuật toán như (6.14) và (6.15).

b. Thuật toán cho nhiều lần chia khoảng.

- **Bước 0:** Nhập a, b , y_0 , δ, n, kmax và ε.

Đặt $h_0 = \frac{b-a}{n}$, $x_0^{(0)} = a$, $y_0^{(0)} = y_0$

Với i = 1, 2,... tính các $y_i^{(0)}$ theo (6.14) và (6.15).

- **Bước 1:** Đặt $h_1 = \frac{h_0}{2}$, $x_0^{(1)} = a$, $y_0^{(1)} = y_0$

Với i = 1, 2,... tính các $y_i^{(1)}$ theo (6.14) và (6.15).

Tính $d_1 = \max_i | y_{2i}^{(1)} - y_i^{(0)} |$, i=0,1,2,...,n

Nếu $d_1 < \varepsilon$ thì dừng thuật toán và lấy mẫu $(x_0,y_0), (x_1,y_1), ..., (x_{2n},y_{2n})$ làm nghiệm xấp xỉ.

Nếu 2 điều trên đây không xẩy ra thì chuyển qua bước (3).

...

- **Bước k:** Đặt $h_k = \frac{h_{k-1}}{2}$, $x_0^{(k)} = a$, $y_0^{(k)} = y_0$

Với i = 1, 2,... tính các $y_i^{(k)}$ theo (6.14) và (6.15).

Tính $d_k = \max_i | y_{2i}^{(k)} - y_i^{(k-1)} |$, i=0,1,2,...,$n.2^{k-1}$

Nếu $d_k < \varepsilon$ thì dừng thuật toán và lấy mẫu $(x_0,y_0), (x_1,y_1), ..., (x_N,y_N)$, trong đó $N = n.2^k$ làm nghiệm xấp xỉ.

Nếu k ≥ kmax thì thông báo phép lặp chưa hội tụ và cũng dừng thuật toán.

Nếu 2 điều trên đây không xẩy ra thì chuyển qua bước (k+1).

6.5. PHƯƠNG PHÁP RUNGE - KUTTA

Phương pháp Runge - Kutta là phương pháp rất hiệu quả: nó vừa có độ chính xác cao lại vừa là phương pháp hiện. Để thành lập những công thức Runge-Kutta có độ chính xác cao hơn công thức Euler, người ta dùng khai triển Taylo nghiệm y(x) tại x_i với nhiều số hạng hơn. Xây dựng công thức Runge-Kutta trong trường hợp tổng quát khá phức tạp, ở đây ta chỉ xét trường hợp đơn giản nhất.

Trở lại xét bài toán (6.1), ta xét khai triển Taylor của nghiệm đúng y(x) :

$$y(x) = y(x_i) + \frac{x-x_i}{1!}y'(x_i) + \frac{(x-x_i)^2}{2!}y''(x_i) + \frac{(x-x_i)^3}{3!}y'''(c_i) , \ c_i \in (x_i,x)$$

Thay $x = x_{i+1} = x_i + h$, ta có

$$y(x_{i+1}) = y(x_i) + hy'(x_i) + \frac{h^2}{2}y''(x_i) + \frac{h^3}{6}y'''(c_i), \ c \in (x_i,x) \qquad (6.16)$$

Trong đó

$$y'(x_i) = f(x_i,y(x_i))$$

$$y''(x_i) = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial y}\frac{dy}{dx} \ | \ x=x_i$$

$$= f_x'(x_i,y_i) + f_y'(x_i,y(x_i))y'(x_i)$$

Thay vào (6.13) ta có

$$y_{i+1} = y_i + hy'_i + \frac{h^2}{2}[f_x'(x_i,y_i) + f_y'(x_i,y_i)y'_i] + O(h^3) \qquad (6.17)$$

Để tránh tính trực tiếp $f_x'(x_i,y_i)$ và $f_y'(x_i,y_i)$, Runge và Kutta đã làm như sau:

Đặt

$$y_{i+1} = y_i + r_1k_1^{(i)} + r_2k_2^{(i)} \qquad (6.18)$$

Trong đó

$$k_1^{(i)} = hf(x_i, y_i)$$

$$k_2^{(i)} = hf(x_i + \alpha h,\ y_i + \beta k_1^{(i)}) \qquad (6.19)$$

và chọn α, β, r_1, r_2 sao cho khai triển theo lũy thừa của h của y_{i+1} xác định bởi (6.18) trùng nhau đến 3 số hạng đầu của vế phải công thức (6.17).

Dùng công thức Taylor của hàm hai biến, ta có:

$$f(x_i + \alpha h,\ y_i + \beta k_1^{(i)}) = f(x_i,y_i) + \alpha h f_x'(x_i,y_i) + \beta k_1^{(i)} f_y'(x_i,y_i) + O(h^2) =$$
$$= y_i' + \alpha h f_x'(x_i,y_i) + \beta k_1^{(i)} f_y'(x_i,y_i) + O(h^2)$$

Từ đây ta có

$$k_1^{(i)} = hf(x_i, y_i) = hy_i'$$

$$k_2^{(i)} = hf(x_i + \alpha h,\ y_i + \beta k_1^{(i)}) =$$

$$= h y_i' + \alpha h^2 f_x'(x_i,y_i) + \beta h^2 y_i' f_y'(x_i,y_i) + O(h^3)$$

Do đó (6.18) có thể viết dưới dạng

$$y_{i+1} = y_i + r_1 h y_i' + r_2[h y_i' + \alpha h^2 f_x'(x_i,y_i) + \beta h^2 y_i' f_y'(x_i,y_i)] + O(h^3) =$$
$$= y_i + r_1 h y_i' + r_2 h y_i' + \alpha r_2 h^2 f_x'(x_i,y_i) + \beta r_2 h^2 y_i' f_y'(x_i,y_i) + O(h^3) =$$
$$= y_i + (r_1 + r_2) h y_i' + h^2 (\alpha r_2 f_x'(x_i,y_i) + \beta r_2 f_y'(x_i,y_i) y_i'] + O(h^3) \quad (6.20)$$

So sánh các hệ số lũy thừa của h trong (6.17) và (6.20) ta có

$$r_1 + r_2 = 1$$

$$\alpha r_2 = \beta r_2 = 1/2$$

Đây là một hệ thống 3 phương trình, 4 ẩn số nên là một hệ vô định. Ta xét một vài họ nghiệm đơn giản

(1) $r_1 = 0$, $r_2 = 1$, $\alpha = \beta = 1/2$. Khi đó (6.18) và (6.19) có dạng

$y_0 = y(x_0)$ đã biết

$$k_1^{(i)} = hf(x_i, y_i)$$

$$k_2^{(i)} = hf(x_i + h/2,\ y_i + k_1^{(i)}/2)$$

$$y_{i+1} = y_i + k_2^{(i)} \quad i=0,1,...,n-1 \qquad (6.21)$$

(2) $r_1 = r_2 = 1/2$, $\alpha = \beta = 1$. Khi đó (6.18) và (6.19) có dạng

$y_0 = y(x_0)$ đã biết

$$k_1^{(i)} = hf(x_i, y_i)$$

$$k_2^{(i)} = hf(x_i + h,\ y_i + k_1^{(i)})$$

$$y_{i+1} = y_i + (1/2)(k_1^{(i)} + k_2^{(i)}) \quad i=0,1,...,n-1 \qquad (6.22)$$

Khi thành lập các công thức (6.18) và (6.19) trên đây ta bỏ qua số hạng $O(h^3)$ trong khai triển Taylor. Ta có thể chứng minh được rằng sai số tại điểm x_i thỏa mãn:

$|y_i - y(x_i)| \le Mh^2$, trong đó M là hằng số dương không phụ thuộc h.

Vậy các phương pháp Runge-Kutta trên đây có độ chính xác cấp hai.

Hoàn toàn tương tự, nếu trong khai triển Taylor của $y(x_{i+1})$ tại x_i ta bỏ qua số hạng $o(h^4)$ thì sẽ nhận được công thức Runge-Kutta có độ chính xác cấp ba, nghĩa là

$|y_i - y(x_i)| \le Mh^3$, trong đó M là hằng số dương không phụ thuộc h.

$y_0 = y(x_0)$ đã biết

$k_1^{(i)} = hf(x_i, y_i)$

$k_2^{(i)} = hf(x_i + h/2,\ y_i + k_1^{(i)}/2)$

$k_3^{(i)} = hf(x_i + h,\ y_i - k_1^{(i)} + 2k_2^{(i)})$

$y_{i+1} = y_i + (1/6)(k_1^{(i)} + 4k_2^{(i)} + k_3^{(i)})$ i=0,1,...,n-1 (6.23)

Nếu bỏ qua số hạng $O(h^5)$ thì ta nhận được công thức Runge-Kutta có độ chính xác cấp 4:

$y_0 = y(x_0)$ đã biết

$k_1^{(i)} = hf(x_i, y_i)$

$k_2^{(i)} = hf(x_i + h/2,\ y_i + k_1^{(i)}/2)$

$k_3^{(i)} = hf(x_i + h/2,\ y_i + k_2^{(i)}/2)$

$k_4^{(i)} = hf(x_i + h/2,\ y_i + k_3^{(i)})$

$y_{i+1} = y_i + (1/6)(k_1^{(i)} + 2k_2^{(i)} + 2k_3^{(i)} + k_4^{(i)})$ i=0,1,...,n-1 (6.24)

Trong các công thức Runge-Kutta nêu trên người ta thường dùng công thức (6.24) vì nó có độ chính xác cao mà lại không quá phức tạp. Trong thực tế việc xác định hằng số M trong đánh giá sai số của phương pháp Runge-Kutta khá phức tạp, do đó người ta thường xác định sai số bằng cách "tính 2 lần" như sau:

Lần đầu tính bằng công thức (6.24) với bước h, nhận được $y_n^{(h)}$ là giá trị gần đúng của y(b) . Sau đó ta lại tính với bước h/2 nhận được $y_{2n}^{(\frac{h}{2})}$ là giá trị gần đúng của y(b) và sai số được xác định bởi:

$$| y_{2n}^{(\frac{h}{2})} - y(b) | \approx (1/15) | y_{2n}^{(\frac{h}{2})} - y_n^{(h)} | \qquad (6.25)$$

Ví dụ.

Cho bài toán Cauchy như sau:

$y' = x + y,\ y(0) = 1$

Hãy tìm nghiệm gần đúng bằng phương pháp Runge-Kutta (6.24) trên [0,0.5] với bước h=0.1

Giải: Ta có $x_i = 0.1i$; i = 0,1,2,3,4,5

$y_0 = 1$

$k_1^{(0)} = 0.1(0+1) = 0.1$

$k_2^{(0)} = 0.1(0+0.05) + (1+0.05)] = 0.11$

$k_3^{(0)} = 0.1(0+0.05) + (1 + 0.055)] = 0.1105$

$k_4^{(0)} = 0.1(0+0.1) + (1 + 0.1105)] = 0.12105$

Từ đó

$$y_1 = 1 + \frac{1}{6}(0.1 + 2*0.11 + 2*0.1105 + 0.12105) = 1.1103$$

Tương tự ta có thể tính y_2, y_3,y_4 và y_5.

Ta có thể thấy rằng $y(0.5) \approx y_5 = 1.7974$

Nghiệm đúng của bài toán Cauchy đã cho là $y=2e^x - x - 1$

từ đó:

$$y(0.5) = 2\, e^{0.5} - 0.5 - 1 = 1.79744$$

Như vậy kết quả nhận được dúng đến 4 số lẻ thập phân.

Các phương pháp Runge - Kutta nêu trên đều có thể áp dụng cho hệ phương trình vi phân cấp một.

Bạn đọc có thể tự viết chương trình theo công thức (6.24) và thử in ra các kết quả trên đây.

6.5. BÀI TẬP

Bài 1. Giải phương trình sau bằng phương pháp Euler

$y' = \frac{xy}{2}$; $x \in [0,1]$, $y(0) = 1$; $h = 0,1$

Bài 2. Giải phương trình sau bằng phương pháp Euler

$y' = x^2 + y^2$; $x \in [0,1]$, $y(0) = 1$; $h = 0,2$

Bài 3. Giải phương trình sau bằng phương pháp Runge-Kutta:

$y' = y - 2x/y$; $x \in [0,1]$, $y(0) = 1$; $h = 0,2$

Bài 4. Giải bài toán sau bằng phương pháp Euler cải tiến và so sánh kết quả với nghiệm đúng:

$y' = x^2 + y^2$; $x \in [0,1]$, $y(0) = 1$; $h = 0,2$

Bài 5. Thử lại hoặc viết mới các chương trình cài đặt các thuật toán Euler rồi chạy thử để kiểm tra các kết quả trên đây.

TÓM TẮT NỘI DUNG CHƯƠNG 6

Trong chương này chúng ta cần chú ý nhất là các vấn đề sau:

1. Bài toán Cauchy

Tìm hàm y=y(x) thỏa mãn:

$y'(x) = f(x,y)$ $x \in [a,b]$, $x_0 = a$ (6.1)

$y(x_0) = y_0$ (6.1b)

Điều kiện (6.1b) được gọi là điều kiện ban đầu hay điều kiện Cauchy.

2. Một số phương pháp tìm nghiệm gần đúng của bài toán Cauchy

a.Phương pháp EULER

b.Phương pháp EULER cải tiến

c. Phương pháp EULER-CAUCHY

d.Phương pháp RUNGE-KUTTA

HƯỚNG DẪN TRẢ LỜI

CHƯƠNG 1

1. Khi đo 1 số góc ta được các giá trị sau:

   a= $21^o37'3"$; b=$1^o10'$

   Sai số tương đối của các số xấp xỉ đó là

   $\delta_a = 0,13.10^{-4}$; $\delta_b = 0,28.10^{-3}$

2. Sai số tuyệt đối của các số xấp là :

   $\Delta_a = 0,13.10^{2}$; $\Delta_b = 0,16.10^{-1}$

3. Số các chữ số đáng tin trong các số a,b là:

   a) 2 ; b) 4

4. Số những chữ số đáng tin trong các số a là:

   a) 3 ; b) 1

   Các số qui tròn với 3 chữ số có nghĩa đáng tin, sai số tuyệt đối Δ và sai số tương đối δ của chúng là:

   a) 2,15 $\Delta = -0,14.10^{-2}$; $\delta = 0,65.10^{-3}$

   b) 0,162 $\Delta = 0,48.10^{-3}$; $\delta = 0,3.10^{-2}$

   c) 0,0120 $\Delta = 0,4.10^{-4}$; $\delta = 0,33.10^{-2}$

   d) –0,00153 $\Delta = 0,19.10^{-5}$; $\delta = 125.10^{-2}6.$

5. Giá trị của các hàm số, sai số tuyệt đối và sai số tương :

   a) u = 0,81 $\Delta_u = 0,27.10^{-2}$; $\delta_u = 0,33.10^{-2}$

   b) u = 3,665 $\Delta_u = 0,7.10^{-2}$; $\delta_u = 0,20.10^{-2}$

CHƯƠNG 2

1. Tính và kiểm tra bằng chương trình định thức của ma trận

   + Dựa vào thuật toán Gauss đã được mô tả bằng ngôn ngữ lập trình C để hoàn thiện chương trình để tính định thức.

   +Det A=52

2. Tìm và kiểm tra bằng chương trình nghịch đảo của ma trận

   + Ma trận nghịch đảo là:

$$A^{-1} = \begin{bmatrix} -4 & 6 & 5 \\ 1 & -1 & -1 \\ 6 & -9 & -7 \end{bmatrix}$$

+ Dựa vào thuật toán khử Gauss-Jordan đã được mô tả bằng ngôn ngữ lập trình C để hoàn thiện chương trình tìm ma trận nghịch đảo.

3. Nghiệm của hệ phương trình là:

   $x_1=1;x_2=2;x_3=3$

4. Giải bằng các phương pháp khử Gauss, khử Gauss-Jordan, phương lặp Jacobi và lặp Gauss-Seidel (nếu thỏa mãn điều kiện) :

   + Phương pháp Gauss x=(-2,09;3,977;1,451;-2,401)

   + Phương pháp Gauss-Jordan x=(-2,057;3,976;1,451;-2,401)

   +Hệ phương trình đã cho không thoả mãn tính chất đường chéo trội

   + Phần viết chương trình: Xem phần đoạn chương trình chính thể hiện thuật toán ***Gauss - Seidel*** để hoàn thiện chương trình với dữ liệu cụ thể trong bài toán.

5. Giải bằng các phương pháp lặp hệ phương trình sau:

   +Phương pháp lặp jacobi qua 3 bước lặp; $x^{(3)}\approx(1,01;-2,008;3,045)$

   +Phương pháp lặ Gauss Seidel qua 3 bước lặ; $x^{(3)}\approx(1;-2;3)$

CHƯƠNG 3

1. Đa thức nội suy Newton tiến xuất phát từ nút 3,50 là:

   $p_3^{(x)}|_{x=} 3,50 + 0,05t$

   $= 33,115 + 1,698\,t + 0,0435\,t(t-1) + 0,00083\,t(t-1)(t-2)$

2. Tích phân xác xuất

   $\Phi(1,43) \approx 0,95686$

3. Cho hàm số $y = e^x$ tại $x = 0,65 + 0,1i$ i=0,1,...,5 tuần tự là 1,91554; 2,11700; 2,33965; 2,58571; 2,85765; 3,15819. Tính ln2.

   Ln 2 = 0,693148

4. Công thức y biểu diễn qua x là:

   $y = 5,045 - 4,043x + 1,009x^2$

5. Xác định gía trị của sai số khi $x = 12^0 30'$.

   $R(12^0 30') \approx 9,094.10^{-10}$

6. Các đa thức xấp xỉ có dạng tương ứng là:

   Y=6,373+0,471x

   $Y=6,36+0,468x+0,0001786x^2$

   $Y=2,15x^{1,31}$

CHƯƠNG 4

(Gợi ý)

5. Tìm 1 khoảng phân ly nghiệm bất bỳ thoả mãn phương trình. Ví dụ[0,Π/2]
   - Tìm hàm $\Psi(x)=1/2\sin x+0,25$
   - Xét đạo hàm $\Psi'(x)$. Nếu $\Psi'(x)<1$ thì thực hiện quá trình lặp theo thuật toán $x_n=\Psi(x_{n-1})$, với $x_0=0$ qua 4 lần lặp ta sẽ tìm được $x_1,x_2,x_3,x_4\ldots$
6. Tìm 1 khoảng phân ly nghiệm bất bỳ thoả mãn phương trình. Ví dụ[1,2]

Áp dụng phương pháp chia đôi ta có bảng giá trị $x_n=(a_n+b_n)/2$ và các khoảng phân ly mới $[a_n,b_n]$ tương ứng qua các bước lặp sau:

N	a_n	b_n	$X_n=(a_n+b_n)/2$	$f(x_n)$
0	1	2	1.5	0.875
1	1	1.5	1.25	-0.29
2	1.25	1.5	1.375	0.22
3	1.25	1.375	1.3125	-0.05
4	1.3125	1.375	1.34375	0.08

$\Rightarrow x_4 \approx 1.34375$

sai số: $|x_4-\alpha| \le |(b_0-a_0)|/2^5=1/2^5=0.03125$.

7. Tìm 1 khoảng phân ly nghiệm bất kỳ thoả mãn phương trình. Ví dụ[1,2]

$f'(x)=3x^2-1 \Rightarrow |f'(x)| \in [2,11] = [m,M]$

áp dụng phương pháp dây cung ta có bảng giá trị

$x_n=(a_nf(b_n)+b_nf(a_n)/(f(b_n)-f(a_n))$ và các khoảng phân ly mới $[a_n,b_n]$ tương ứng qua các bước lặp sau:

n	a_n	b_n	$x_n=a_nf(b_n)-b_nf(a_n)/f(b_n)-f(a_n)$	$f(x_n)$
0	1	2	1.1667	-0.57
1	1.1667	2	1.25313	-0.285
2	1.25313	2	1.29345	-0.1295
3	1.29395	2	1.31129	-0.06
4	1.31129	2	1.31889	-0.02

$\Rightarrow x_4 \approx 1.31899$

sai số

$|x_4-\alpha| \le ((M-m)/m)*|x_4-x_3|$

$|x_4-\alpha| \le ((11-2)/2)*|1.31899-1.31129| = 0.03465.$

8. Để tính $\sqrt{5}$ bằng phương pháp chia đôi thì ta phải đưa về giải gần đúng phương trình $f(x)=x^2-5=0$ (1) trên khoảng [2,3]

áp dụng phương pháp chia đôi ta có bảng giá trị $x_n=(a_n+b_n)/2$ và các khoảng phân ly mới $[a_n,b_n]$ tương ứng qua các bước lặp sau

N	a_n	b_n	$x_n=(a_n+b_n)/2$	$f(x_n)$
0	2	3	2.5	1.25
1	2	2.5	2.25	0.06
2	2.25	2.5	2.375	0.64
3	2.25	2.375	2.3125	0.34
4	2.25	2.3125	2.28125	0.204

$\Rightarrow x \approx x_4 = 2.28125$

$|x_4-\sqrt{5}| \le (b_0-a_0)/2^5 = 1/32 = 0.03125$

CHƯƠNG 5

1. Cho hàm số y = logx với số các giá trị tại x = 50; 55; 60; 65 tuần tự là 1,6990; 1,7404;1,7782; 1,8129. Hãy tính đạo hàm của y tại x = 50 và so sánh với kết quả trực tiếp:

   Dùng nội suy : $y'(50) = 0{,}0087$

   Tính trực tiếp : $y'(50) = 0{,}43429/x|_{x=50} = 0{,}0087$

2. Cho tích phân

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

   a)

   b) $I = 0{,}69315 \pm 0{,}00002$

3. Cho tích phân

$$I = \int_0^1 \frac{\sin x}{x} dx$$

a) n> = 10

b)Với n = 10 thì sai số < $1{,}2.10^{-7}$

c)Theo công thức hình thang : I = 0,9458

Theo công thức Simson ta có : I = 0,94608

CHƯƠNG 6

1. Giải phương trình sau bằng phương pháp Euler

$y' = \frac{xy}{2}$; $x \in [0,1]$, $y(0) = 1$; $h = 0{,}1$

Áp dụng công thức Euler (6.7) ta có bảng giá trị sau:

I	x_i	y_i
0	0	1
1	0,1	1
2	0,2	1,005
3	0,3	1,0105
4	0,4	1,030275
5	0,5	1,05088
6	0,6	1,07715
7	0,7	1,109468
8	0,8	1,148299
9	0,9	1,194231
10	1	1,2479715

2. Giải phương trình sau bằng phương pháp Euler

$y' = x^2 + y^2$; $x \in [0,1]$, $y(0) = 1$; $h = 0{,}2$

Áp dụng công thức Euler (6.7) ta có bảng giá trị sau:

i	x_i	y_i
0	0	1
1	0,2	1,2
2	0,4	1,496
3	0,6	1,9756
4	0,8	2,8282
5	1	4,5559

3. Giải phương trình sau bằng phương pháp Runge-Kutta:

   y' = y-2x/y ; x∈ [0,1], y(0) =1; h=0,2

   + Sử dụng công thức Runge-Kutta có độ chính xác cấp 4 ta có bảng giá trị:

i	x_i	y_i
0	0,0	1
1	0,2	1,1832292
2	0,4	1,3416668
3	0,6	1,4832847
4	0,8	1,6124665
5	1	1,7320713

4. Giải bài toán sau bằng phương pháp Euler cải tiến và so sánh kết quả với nghiệm đúng:

   $y' = y - \frac{2x}{y}$; x∈ [0,1], y(0) =1; h=0,2.

   Gợi ý: Áp dụng công thức Euler cải tiến (6.13) để tìm nghiệm $x_i=x_{i-1}+h$ với $x_0=0$, sau đó mới so sánh với nghiệm đúng tại các điểm y(0,2), y(0,4), y(0,6), y(0,8), y(1)

TÀI LIỆU THAM KHẢO

1. Dương Thùy Vỹ, Phương pháp tính, Nhà xuất bản Khoa học và Kỹ thuật, 2001
2. Đinh Văn Phong, Phương pháp số trong cơ học, Nhà xuất bản Khoa học và Kỹ thuật, 1999
3. Lê Trọng Vinh, Giải tích số, Nhà xuất bản Khoa học và Kỹ thuật, 2000.
4. Phạm Kỳ Anh , Giải tích số, Nhà xuất bản Đại học Quốc gia Hà Nội, 1966.
5. Phạm Phú Triêm - Nguyễn Bường, Giải tích số, thuật toán, chương trình Pascal, Nhà xuất bản Đại học Quốc gia Hà Nội, 2000
6. Szidarovszky Ferenc, Nhập môn phương pháp số (tiếng Hung), Kozgazdasági és Jogi Konyvkiado, Budapest 1974.
7. Tạ Văn Đỉnh, Phương pháp tính, Nhà xuất bản Giáo dục – 1995
8. Trần Văn Minh, Phương pháp số và chương trình bằng Turbo Pascal, Nhà xuất bản Khoa học và Kỹ thuật, 1998.
9. Phan Đăng Cầu-Phan Thị Hà, Phương pháp số, Học viện CNBCVT,2002

MỤC LỤC