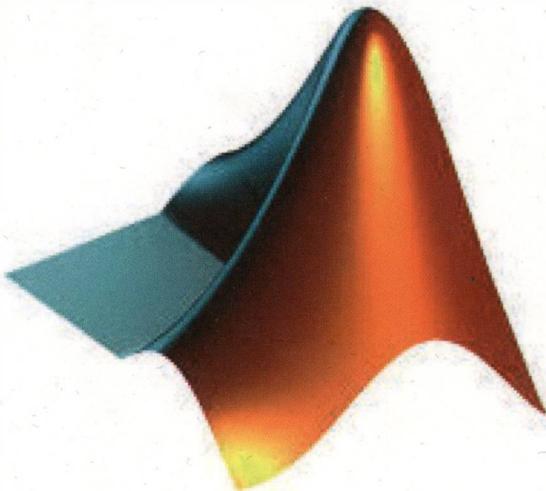


LÊ TRỌNG VINH, TRẦN MINH TOÀN

Giáo trình Phương pháp tính và Matlab

Lý thuyết, bài tập và chương trình minh họa
(Dùng cho sinh viên khối các trường Khoa học Công nghệ)



NHÀ XUẤT BẢN BÁCH KHOA – HÀ NỘI

LÊ TRỌNG VINH, TRẦN MINH TOÀN



**Giáo trình
PHƯƠNG PHÁP TÍNH VÀ MATLAB
Lý thuyết, bài tập và chương trình minh họa**

**(Dùng cho sinh viên khối các trường
Khoa học Công nghệ)**

NHÀ XUẤT BẢN BÁCH KHOA - HÀ NỘI

Bản quyền thuộc về trường Đại học Bách Khoa Hà Nội.

Mọi hình thức xuất bản, sao chép mà không có sự cho phép bằng văn bản của trường là vi phạm pháp luật.

Mã số: 1264 – 2013/CXB/01 – 51/BKHN

Biên mục trên xuất bản phẩm của Thư viện Quốc gia Việt Nam

Lê Trọng Vinh

Giáo trình phương pháp tính và matlab : Lý thuyết, bài tập và chương trình minh họa : Dùng cho sinh viên khối các trường Khoa học Công nghệ / Lê Trọng Vinh, Trần Minh Toàn. - H. : Bách khoa Hà Nội, 2013. - 228tr. ; 24cm

Thư mục: tr. 223

ISBN 9786049115578

1. Phương pháp tính
 2. Phần mềm Matlab
 3. Giáo trình
- 515.0285 - dc14

BKF0033p-CIP

LỜI NÓI ĐẦU

PHƯƠNG pháp tính còn được gọi là giải tích số hay toán học tính toán là môn khoa học nghiên cứu cách giải gần đúng, chủ yếu là giải số các phương trình, các bài toán xấp xỉ hàm số và các bài toán tối ưu. Ngay từ đầu, toán học sinh ra do yêu cầu giải quyết các vấn đề thực tế như tính diện tích một mảnh đất; đo chiều cao của các vật có độ cao lớn; tìm quỹ đạo của sao chổi, đường đi của tàu buôn trên biển,... Như vậy, có thể nói toán học ban đầu xuất hiện chính là toán học tính toán.

Từ những năm cuối của thế kỷ XX, phương pháp tính được phát triển mạnh mẽ cùng với sự bùng nổ của tin học. Đặc biệt, với sự xuất hiện của các siêu máy tính thì khả năng song song hóa các quá trình tính toán ngày càng được mở rộng.

Các nhiệm vụ chính của phương pháp tính là:

1. Giải gần đúng các loại phương trình đại số hay siêu việt; giải hệ phương trình; tìm trị riêng, vector riêng của ma trận; giải gần đúng phương trình vi phân,...
2. Xấp xỉ hàm số: Thay hàm có dạng phức tạp hay hàm cho dưới dạng bảng số bởi hàm có dạng đơn giản hơn để dễ tính toán.

Trong khi đó, tin học có nhiệm vụ cài đặt và khai thác thực hiện quá trình tính để cho kết quả mong muốn. Song việc tăng tốc độ tính toán (khi khối lượng tính toán lớn) đối với máy gấp nhiều khăn về kỹ thuật và đòi hỏi chi phí lớn. Do đó, cần thiết phải cải tiến thuật toán để

có thể giải các bài toán cỡ lớn. Điều đó có nghĩa là toán học tính toán và tin học có mối quan hệ qua lại đặc biệt quan trọng.

Hiện nay, đã có nhiều giáo trình khác nhau, giới thiệu các cách sử dụng các loại ngôn ngữ lập trình khác nhau để khai thác tính toán như ngôn ngữ C, Maple,... Trong giáo trình này chúng tôi giới thiệu ngôn ngữ lập trình bậc cao MATLAB chuyên được sử dụng cho các tính toán kỹ thuật. Đối với hầu hết các vấn đề, chúng tôi giới thiệu thuật toán và kèm theo chương trình MATLAB (đã được chạy thử một cách cẩn thận) để độc giả kiểm nghiệm và có thể dễ dàng sử dụng để giải quyết các vấn đề cần nghiên cứu.

Giáo trình bao gồm 6 chương, trong đó phần các thuật toán (phương pháp tính) do PGS. TS. Lê Trọng Vinh biên soạn, phần lập trình MATLAB do ThS. Trần Minh Toàn biên soạn.

Chúng tôi xin được bày tỏ lòng biết ơn sâu sắc đến TS. Hà Thị Ngọc Yên và TS. Nguyễn Thanh Huyền đã đọc bản thảo và có những đóng góp vô cùng quý báu. Do giáo trình được biên soạn lần đầu nên không thể tránh khỏi những thiếu sót. Chúng tôi rất mong sự lượng thứ và góp ý của bạn đọc. Mọi ý kiến xin chuyển về địa chỉ: Viện Toán ứng dụng và Tin học, Đại học Bách Khoa Hà Nội, số 1 Đại Cồ Việt, Hà Nội.

Các tác giả

MỤC LỤC

Lời nói đầu	iii
Mục lục	1
Chương 1 . Sai số	7
1 Số xấp xỉ, sai số	7
1.1 Sai số tuyệt đối	7
1.2 Sai số tương đối	8
1.3 Chữ số có nghĩa	8
1.4 Chữ số tin tưởng, chữ số khả nghi trong một số . . .	8
1.5 Sai số quy tròn và quy tròn số	9
1.6 Cách viết số xấp xỉ	9
2 Các phép tính về sai số	9
2.1 Các phép tính	9
2.2 Công thức tổng quát về sai số	11
2.3 Bài toán ngược về sai số	12
2.4 Sai số phương pháp, sai số tính toán và sự ổn định .	14
Chương 2 . MATLAB cơ bản	17
1 Khởi động MATLAB	18
2 Biểu thức MATLAB: biến, số, toán tử, hàm	19
2.1 Biến	19
2.2 Số	20
2.3 Toán tử	20
2.4 Hàm	21

3	Các dạng dữ liệu cơ bản trong MATLAB	22
3.1	Vector	22
3.2	Đa thức	27
3.3	Ma trận	30
4	Vẽ đồ thị trong MATLAB	37
4.1	Đồ thị 2D	37
4.2	Đồ thị 3D	41
5	Lập trình với MATLAB	43
5.1	Thủ tục (Script)	44
5.2	Hàm "m-files" (Function)	45
5.3	Nhập, xuất dữ liệu	48
5.4	Điều khiển luồng	55
5.5	Vector hóa (Vectorization)	61
5.6	Tính giá trị hàm một cách gián tiếp	66
5.7	Chú thích	66
5.8	Gỡ lỗi	67
5.9	Phân tích một chương trình sử dụng "The Profiler"	68
6	Bài tập	69
Chương 3 . Giải gần đúng phương trình		79
1	Tìm khoảng phân ly nghiệm	80
1.1	Khoảng phân ly nghiệm	80
1.2	Phương pháp hình học tìm khoảng phân ly nghiệm	80
2	Phương pháp lặp đơn	82
2.1	Nội dung phương pháp	82
2.2	Sự hội tụ của phương pháp	82
2.3	Sai số	84
2.4	Chương trình MATLAB	85
3	Phương pháp Newton (phương pháp tiếp tuyến)	87
3.1	Nội dung phương pháp	87
3.2	Sự hội tụ của phương pháp Newton	88
3.3	Sai số	89
3.4	Chương trình MATLAB	90
4	Phương pháp dây cung	92

4.1	Nội dung phương pháp	92
4.2	Sự hội tụ của phương pháp dây cung	93
4.3	Sai số	94
4.4	Chương trình MATLAB	94
5	Phương trình đa thức bậc n	96
5.1	Miền chứa nghiệm của đa thức	96
5.2	Sơ đồ Horner tính giá trị của đa thức	97
5.3	Chương trình MATLAB	97
6	Giải gần đúng hệ phương trình phi tuyến	98
6.1	Phương pháp Newton	99
6.2	Phương pháp lặp	100
6.3	Chương trình MATLAB	101
7	Bài tập	102

Chương 4 . Phương pháp số trong đại số tuyến tính 105

1	Mở đầu về hệ đại số tuyến tính	105
1.1	Đặt vấn đề	105
1.2	Phương pháp Cramer	106
2	Phương pháp giải đúng và chương trình MATLAB	107
2.1	Phương pháp Gauss	107
2.2	Chương trình MATLAB	111
2.3	Phương pháp Gauss-Jordan (tru tối đại)	112
2.4	Phương pháp Cholesky (khai căn)	116
2.5	Công thức truy đuổi giải hệ có ma trận dạng ba đường chéo	121
2.6	Chương trình MATLAB	122
3	Phương pháp lặp đơn	123
3.1	Chuẩn của ma trận và sự hội tụ của dây ma trận . .	123
3.2	Phương pháp lặp đơn (lặp cổ điển)	125
4	Sự không ổn định của hệ đại số tuyến tính	128
5	Ma trận nghịch đảo và chương trình MATLAB	130
5.1	Bài toán	130
5.2	Phương pháp tìm A^{-1} trong đại số tuyến tính	131
5.3	Phương pháp Gauss-Jordan	131
5.4	Trường hợp ma trận đối xứng. Phương pháp Cholesky	134

6	Tri riêng, vector riêng của ma trận và chương trình MATLAB	137
6.1	Khái niệm về tri riêng và vector riêng	137
6.2	Tri riêng và vector riêng của các ma trận đồng dạng	137
6.3	Phương pháp Danhilepski	138
6.4	Phương pháp Krulôp A.N.	145
7	Tìm gần đúng tri riêng và vector riêng	147
7.1	Nội dung phương pháp tìm tri riêng trôi	147
7.2	Trường hợp ma trận đối xứng, xác định dương	152
8	Bài tập	157
Chương 5 . Phép nội suy và xấp xỉ hàm		161
1	Khái niệm về nội suy	161
1.1	Bài toán	161
1.2	Sự duy nhất của đa thức nội suy	163
2	Đa thức nội suy Lagrange	163
2.1	Đa thức nội suy Lagrange	163
2.2	Sai số của đa thức nội suy Lagrange	164
2.3	Chương trình MATLAB	165
3	Đa thức nội suy Newton có mốc cách đều	166
3.1	Khái niệm về sai phân	166
3.2	Đa thức nội suy Newton tiến có mốc nội suy cách đều	168
3.3	Đa thức nội suy Newton lùi có mốc nội suy cách đều	169
3.4	Sai số	170
3.5	Chương trình MATLAB	172
4	Phép nội suy ngược	173
4.1	Sử dụng đa thức nội suy Lagrange	173
4.2	Trường hợp các mốc nội suy cách đều	174
4.3	Chương trình MATLAB	174
5	Phương pháp bình phương tối thiểu...	175
5.1	Khái niệm về sai số trung bình phương	175
5.2	Bài toán	176
5.3	Xây dựng phương pháp tính	177
5.4	Sai số của phương pháp	179
5.5	Trường hợp $\{\varphi_k(x)\}_{k=0,m}$ là hệ trực chuẩn	180

5.6	Trường hợp hệ cơ bản là hệ đại số	181
5.7	Trường hợp hệ cơ bản là hệ lượng giác	182
5.8	Chương trình MATLAB	185
6	Tìm hàm thực nghiệm theo phương pháp bình phương tối thiểu	185
6.1	Hàm thực nghiệm dạng $y = ae^{bx}$, ($a > 0$)	186
6.2	Hàm thực nghiệm dạng $y = ax^b$, ($a > 0$, $x > 0$) . .	187
6.3	Chương trình MATLAB	189
7	Bài tập	189
Chương 6 . Đạo hàm, tích phân và phương trình vi phân . . .		193
1	Tính đạo hàm	193
1.1	Tính đạo hàm nhờ đa thức nội suy Lagrange	193
1.2	Trường hợp các mốc nội suy cách đều nhau	194
1.3	Trường hợp $n = 2$	194
1.4	Trường hợp $n = 3$	195
2	Tính gần đúng tích phân xác định	197
2.1	Mở đầu	197
2.2	Công thức hình thang	197
2.3	Công thức Simpson	202
2.4	Chương trình MATLAB	205
3	Giải gần đúng phương trình vi phân (bài toán Cauchy) . .	205
3.1	Mở đầu	205
3.2	Bài toán Cauchy	206
3.3	Phương pháp giải bài toán Cauchy	206
3.4	Chương trình MATLAB	216
4	Giải phương trình vi phân cấp cao và hệ phương trình vi phân	217
4.1	Bài toán	217
4.2	Công thức Euler	218
4.3	Công thức dạng Runge-Kutta	218
4.4	Chương trình MATLAB	219
5	Bài tập	220
Tài liệu tham khảo		223

1 CHƯƠNG

SAI SỐ

§1. SỐ XẤP XỈ, SAI SỐ

TEONG tính toán, ta phải làm việc với các giá trị gần đúng của một đại lượng, vì vậy luôn có sai số. Việc hiểu rõ sai số đó là rất cần thiết. Giả sử A là đại lượng cần nghiên cứu, nhưng thực tế ta chỉ thu được đại lượng gần đúng với nó, ký hiệu là a . Khi đó, ta nói a là số xấp xỉ của A và ký hiệu $a \approx A$.

1.1 Sai số tuyệt đối

Đại lượng $\Delta = |a - A|$ được gọi là sai số tuyệt đối của số xấp xỉ a . Do A nói chung không biết nên không thể biết được Δ . Vì vậy, người ta đưa ra cận trên của Δ và gọi là sai số tuyệt đối giới hạn cho phép của a .

Định nghĩa 1.1. Sai số tuyệt đối giới hạn cho phép của số gần đúng a là số $\Delta_a > 0$ thỏa mãn

$$|a - A| \leq \Delta_a \text{ hay } a - \Delta_a \leq A \leq a + \Delta_a. \quad (1.1)$$

Tất nhiên, ta luôn mong tìm được Δ_a càng nhỏ càng tốt. Khi đó, bất đẳng thức (1.1) cũng có thể được thay thế bởi đẳng thức

$$|a - A| = \Delta_a \text{ hay } A = a \pm \Delta_a.$$

1.2 Sai số tương đối

Để mô tả chất lượng của việc xấp xỉ, người ta gọi $\delta_a = \frac{\Delta_a}{|a|}$. ($a \neq 0$) là sai số tương đối của величина a . Sai số tương đối δ_a thường được tính theo tỷ lệ phần trăm (%), nghĩa là

$$\delta_a = \frac{\Delta_a}{|a|} \times 100\%.$$

1.3 Chữ số có nghĩa

Để thống nhất việc biểu diễn các số thập phân trong giáo trình, ta đưa ra quy ước sau.

Quy ước: Với dạng số thập phân $a = \pm \star \star \star a_1 a_0 a_{-1} \star \star \star$, từ đây về sau ta hiểu dấu "chấm" chính là dấu "phẩy" để phân biệt phần nguyên và phần thập phân của số đó. Ví dụ khi ta viết 2.5 chính là 2,5 như cách viết tay thông thường.

Trong tính toán cũng như trong máy tính thường sử dụng số trong dạng chữ số thập phân do 10 ký tự 0, 1, 2, ..., 9 tạo nên. Trong một số có nhiều chữ số, người ta quy ước *những chữ số có nghĩa* trong số đó là những chữ số kể từ chữ số khác "0" đầu tiên tính từ trái qua phải.

Ví dụ 1.1. Số 0.003042 có 4 chữ số có nghĩa là 3, 0, 4, 2.

Xét một ví dụ khác, số $3600 = 360 \times 10^2$ chỉ có 3 chữ số có nghĩa. Mặt khác, nếu viết $3600 = 36 \times 10^3$ thì chỉ có 2 chữ số có nghĩa. Như vậy, thừa số 10^n trong số đó không được tính vào chữ số có nghĩa. Cùng với số trên, để có 5 chữ số có nghĩa thì ta viết $3600 = 36000 \times 10^{-1}$.

1.4 Chữ số tin tưởng, chữ số khả nghi trong một số

Định nghĩa 1.2. Giả sử số $a = \pm a_n \dots a_2 a_1 a_0, a_{-1} a_{-2} \dots a_{-\alpha} \dots$ có sai số tuyệt đối giới hạn là Δ_a . Khi đó, nếu $0 \leq \Delta_a \leq 0.5 \times 10^{-n}$ thì ta nói chữ số a_s là *chữ số tin tưởng*; ngược lại ($\Delta_a > 0.5 \times 10^{-n}$) thì chữ số a_s được gọi là *chữ số khả nghi*.

Từ định nghĩa trên ta dễ dàng suy ra, nếu chữ số a_s là chữ số tin tưởng thì mọi chữ số đứng trước nó đều là chữ số tin tưởng, nếu a_s là chữ số khả

nghi thì mọi chữ số đứng sau nó đều là chữ số khả nghi.

Ví dụ 1.2. Giả sử số $a = 2.5785030$ có sai số tuyệt đối là $\Delta_a = 0.0043$.

Do $\Delta_a = 0.43 \times 10^{-2} < 0.5 \times 10^{-2}$ nên chữ số thứ hai sau dấu chấm là chữ số tin tưởng. Vậy số a có 3 chữ số tin tưởng là 2, 5, 7 còn các chữ số 8, 5, 0, 3, 0 là các chữ số khả nghi.

1.5 Sai số quy tròn và quy tròn số

Trong tính toán, ta tính được số a có nhiều chữ số, thường phải ngắt bớt và chỉ giữ lại một số chữ số trong số đó và thu được số \tilde{a} để tính tiếp. Việc thay a bởi \tilde{a} được gọi là sự quy tròn số và đại lượng $\theta_a = |a - \tilde{a}|$ được gọi là sai số quy tròn.

Quy ước: Nếu chữ số đầu tiên (kể từ trái sang phải) trong các số bị bỏ đi của số a lớn hơn hoặc bằng 5 thì chữ số cuối cùng được giữ lại liền kề trước nó được tăng lên một đơn vị, ngược lại thì giữ nguyên.

Ví dụ 1.3. 1. Hãy quy tròn số $a = 237.542457$ để được số \tilde{a} có 2 số lẻ thập phân. Theo quy ước trên thì $\tilde{a} = 237.54$.

2. Cho số $b = 237865$. Hãy quy tròn số b để thu được số \tilde{b} chỉ gồm 4 chữ số. Theo quy ước trên ta có $\tilde{b} = 2379 \times 10^2$.

1.6 Cách viết số xấp xỉ

Giả sử A là số cần tính, ta chỉ tính được a và Δ_a là sai số tuyệt đối giới hạn cho phép. Ta viết $A = a \pm \Delta_a$ để biểu diễn kết quả tính toán. Còn cách viết $A \approx a$ (xấp xỉ) là cách viết chỉ gồm các chữ số có nghĩa (thường là số tin tưởng như trong bảng tính, bảng logarit, bảng giá trị các hàm số lượng giác,...).

§2. CÁC PHÉP TÍNH VỀ SAI SỐ

2.1 Các phép tính

Giả sử đại lượng f có sai số tuyệt đối giới hạn là Δ_f và sai số tương đối là δ_f . Ta đã biết $\Delta_f = |\Delta f|$ (Δf là số gia của đại lượng f). Khi đó:

1. Nếu $u = x + y + z$ thì $\Delta_u = \Delta_x + \Delta_y + \Delta_z$, ($x, y, z > 0$).

2. Nếu $u = x - y$ thì $\delta_u = \frac{\Delta_x + \Delta_y}{|x - y|}$, ($x, y > 0$).

3. Nếu $u = xyz$ thì $\delta_u = \delta_x + \delta_y + \delta_z$, ($x, y, z > 0$).

4. Nếu $u = \frac{x}{y}$ thì $\delta_u = \delta_x + \delta_y$, ($x, y > 0$).

Chứng minh. Ta sẽ chứng minh cho trường hợp 2 và 3 còn các trường hợp 1 và 4 được chứng minh tương tự.

2. Với $u = x - y \implies \Delta u = \Delta x - \Delta y \implies |\Delta u| \leq |\Delta x| + |\Delta y|$

hay $\Delta_u \leq \Delta_x + \Delta_y \implies \frac{\Delta_u}{|u|} \leq \frac{\Delta_x + \Delta_y}{|x - y|}$.

Vậy $\delta_u = \frac{\Delta_u}{|u|} = \frac{\Delta_x + \Delta_y}{|x - y|}$.

3. Ta có

$$\begin{aligned} u = xyz &\implies \ln u = \ln x + \ln y + \ln z \\ &\implies \Delta(\ln u) = \Delta(\ln x) + \Delta(\ln y) + \Delta(\ln z). \end{aligned}$$

Mặt khác, $\Delta(\ln u) \approx d(\ln u) = \frac{du}{u} \approx \frac{\Delta u}{u}$. Tương tự ta cũng có

$$\Delta(\ln x) \approx \frac{\Delta x}{x}; \quad \Delta(\ln y) \approx \frac{\Delta y}{y}; \quad \Delta(\ln z) \approx \frac{\Delta z}{z}.$$

Khi đó

$$\begin{aligned} \frac{\Delta u}{u} &\approx \frac{\Delta x}{x} + \frac{\Delta y}{y} + \frac{\Delta z}{z} \\ \implies \left| \frac{\Delta u}{u} \right| &\approx \left| \frac{\Delta x}{x} \right| + \left| \frac{\Delta y}{y} \right| + \left| \frac{\Delta z}{z} \right| \end{aligned}$$

hay

$$\delta_u = \frac{\Delta_u}{|u|} = \frac{\Delta_x}{|x|} + \frac{\Delta_y}{|y|} + \frac{\Delta_z}{|z|} = \delta_x + \delta_y + \delta_z.$$

Chú ý 1.1. Trong công thức hiệu, nếu $|x - y|$ quá bé thì sai số sẽ lớn. Vì vậy, trong tính toán người ta tìm cách tránh phép trừ các số khá gần nhau khi tính sai số tương đối của hiệu hai số.

2.2 Công thức tổng quát về sai số

Giả sử $u = f(x_1, x_2, \dots, x_n)$ với f là hàm khả vi liên tục thì

$$\Delta_u = \sum_{i=1}^n \left| \frac{\partial f}{\partial x_i} \right| \Delta_{x_i}, \quad (1.2)$$

và nếu $f > 0$ thì ta có:

$$\delta_u = \sum_{i=1}^n \left| \frac{\partial \ln f}{\partial x_i} \right| \Delta_{x_i}. \quad (1.3)$$

Chứng minh. Bằng cách sử dụng công thức số gia hàm số

$$\Delta u \approx du = \sum_{i=1}^n \frac{\partial f}{\partial x_i} \Delta_{x_i},$$

ta có

$$|\Delta u| \leq \sum_{i=1}^n \left| \frac{\partial f}{\partial x_i} \right| |\Delta_{x_i}|$$

hay

$$\Delta_u = \sum_{i=1}^n \left| \frac{\partial f}{\partial x_i} \right| \Delta_{x_i}.$$

Còn

$$\begin{aligned} \delta_u &= \frac{\Delta_u}{|u|} = \frac{1}{|f|} \sum_{i=1}^n \left| \frac{\partial f}{\partial x_i} \right| \Delta_{x_i} \\ &= \sum_{i=1}^n \left| \frac{1}{f} \frac{\partial f}{\partial x_i} \right| \Delta_{x_i} = \sum_{i=1}^n \left| \frac{\partial \ln f}{\partial x_i} \right| \Delta_{x_i}. \end{aligned}$$

Đây chính là điều phải chứng minh. ■

Ví dụ 1.4. Tìm sai số tuyệt đối và sai số tương đối giới hạn của thể tích hình cầu $V = \frac{1}{6}\pi d^3$, nếu đường kính $d = 3.7 \pm 0.05$ (cm) và $\pi = 3.14 \pm 0.0016$.

Giải: Xem d và π là các đối số của hàm V , ta có:

$$\begin{aligned} \frac{\partial V}{\partial \pi} &= \frac{1}{6} d^3 = \frac{1}{6} (3.7)^3 = 8.44; \\ \frac{\partial V}{\partial d} &= \frac{1}{2} \pi d^2 = \frac{1}{2} (3.14)(3.7)^2 = 21.5. \end{aligned}$$

Theo công thức (1.2) ta được sai số tuyệt đối giới hạn

$$\begin{aligned}\Delta_V &= \left| \frac{\partial V}{\partial \pi} \right| \Delta_\pi + \left| \frac{\partial V}{\partial d} \right| \Delta_d = 8.44 \times 0.0016 + 21.5 \times 0.05 \\ &= 1.088 \approx 1.1(cm^3).\end{aligned}$$

Vậy $V = \frac{1}{6}\pi d^3 = \frac{1}{6}(3.14)(3.7)^3 \pm 1.1 = 27.4 \pm 1.1(cm^3)$.

Sai số tương đối $\delta_V = \frac{\Delta_V}{|V|} = \frac{1.088}{27.4} \approx 0.04 = 4\%$.

2.3 Bài toán ngược về sai số

Trong phần trước ta đã tính sai số của hàm số tùy thuộc vào sai số của đối số. Nay giờ ta xét trường hợp cho sai số của hàm thì giới hạn sai số của đối số cần cho phép là bao nhiêu?

Bài toán: Cho hàm số $u = f(x_1, x_2, \dots, x_n)$. Hỏi sai số tuyệt đối (hay tương đối) giới hạn của mỗi đối số là bao nhiêu để sai số tuyệt đối (hay tương đối) giới hạn của hàm số không vượt quá một số dương cho trước (số dương này được gọi là sai số cho phép)?

Để giải quyết vấn đề đã đặt ra, có hai cách thường được sử dụng.

1. Cách giải quyết theo nguyên tắc sai số của các đối số ngang bằng nhau.

Theo công thức (1.2) ta có:

$$\Delta_u = \sum_{i=1}^n \left| \frac{\partial f}{\partial x_i} \right| \Delta_{x_i}.$$

Ta xem ảnh hưởng của mọi thành phần như nhau, tức là

$$\begin{aligned}\left| \frac{\partial f}{\partial x_1} \right| \Delta_{x_1} &= \left| \frac{\partial f}{\partial x_2} \right| \Delta_{x_2} = \dots = \left| \frac{\partial f}{\partial x_n} \right| \Delta_{x_n} \\ \implies \Delta_u &= \sum_{i=1}^n \left| \frac{\partial f}{\partial x_i} \right| \Delta_{x_i} = n \left| \frac{\partial f}{\partial x_i} \right| \Delta_{x_i}.\end{aligned}$$

Từ đó ta có công thức sai số tuyệt đối giới hạn cho phép của mỗi thành phần:

$$\Delta_{x_i} = \frac{\Delta_u}{n \left| \frac{\partial f}{\partial x_i} \right|} \quad i = 1, 2, \dots, n. \quad (1.4)$$

Ví dụ 1.5. Cho hình trụ đứng, bán kính $R = 2(m)$; chiều cao $H = 3(m)$. Hỏi sai số tuyệt đối của R và H là bao nhiêu để thể tích $V = \pi R^2 H$ của hình trụ đạt sai số tuyệt đối giới hạn là $0.1(m^3)$.

Giải: Ta có

$$V = \pi R^2 H, \quad \Delta_V = 0.1(m^3), \quad R = 2(m), \quad H = 3(m)$$

và số $\pi = 3.14$ thì khi đó:

$$\begin{aligned}\frac{\partial V}{\partial \pi} &= R^2 H = 2^2 \times 3 = 12; \\ \frac{\partial V}{\partial H} &= \pi R^2 = 3.14 \times 2^2 = 12.6; \\ \frac{\partial V}{\partial R} &= 2\pi RH = 2 \times 3.14 \times 2 \times 3 = 37.7.\end{aligned}$$

Ở đây $n = 3$, theo nguyên tắc ngang bằng nhau ta thu được:

$$\begin{aligned}\Delta_\pi &= \frac{0.1}{3 \times 12} < 0.003; \\ \Delta_R &= \frac{0.1}{3 \times 12.6} < 0.001; \\ \Delta_H &= \frac{0.1}{3 \times 37.7} < 0.003\end{aligned}$$

là các sai số của mỗi thành phần để V có sai số tuyệt đối giới hạn không vượt quá $0.1(m^3)$.

2. Phương pháp điều chỉnh

Trong thực tế, khi sai số của thành phần nào đó tính theo nguyên tắc ngang bằng nhau là không phù hợp (không thực hiện được) ta phải điều chỉnh thành phần đó. Chẳng hạn đối với bài toán sau:

Ví dụ 1.6. Hỏi phải đo bán kính $R = 30.5(cm)$ có sai số tối đa là bao nhiêu và số π lấy là bao nhiêu để được hình tròn có diện tích đạt sai số tương đối không quá 1%?

Giải: Ta có diện tích hình tròn $S = \pi R^2$, nên $\ln S = \ln \pi + 2 \ln R$, suy ra sai số tương đối

$$\delta_S = \frac{\Delta_S}{S} = \frac{1}{\pi} \Delta_\pi + \frac{2}{R} \Delta_R = 1\% = 0.01.$$

Theo nguyên tắc ngang bằng nhau (có hai thành phần) thì

$$\frac{\Delta_\pi}{\pi} = 2 \frac{\Delta_R}{R} = \frac{0.01}{2} = 0.005.$$

Từ đó suy ra

- $\Delta_\pi = \pi \times 0.005 \leq 0.0016$ với $\pi = 3.14$;
- $\Delta_R = \frac{R}{2} \times 0.005 \leq 0.07(cm)$ với $R = 30.5(cm)$.

Rõ ràng đối với R khó thực hiện được vì Δ_R quá nhỏ, do đó cần phải tăng Δ_R lên. Để tổng cả hai thành phần là 0.01 nên khi Δ_R tăng thì Δ_π phải giảm, nghĩa là cần phải lấy thêm chữ số thập phân trong số π .

Chẳng hạn, lấy $\pi = 3.142$ thì $\frac{\Delta_\pi}{\pi} = \frac{3.142 - 3.1416}{3.142} = 0.00013$. Khi đó $2 \frac{\Delta_R}{R} = 0.01 - 0.00013 = 0.00983$, suy ra

$$\Delta_R = \frac{0.00983}{2} \times 30.5 < 0.15(cm).$$

Ta thấy yêu cầu của độ chính xác này có thể thực hiện được.

Qua ví dụ trên ta thấy, việc điều chỉnh là tùy thuộc công việc, không theo nguyên tắc chung nên chúng tôi chỉ nêu ví dụ để cùng suy ngẫm khi thực hiện công việc mà thực tế đặt ra.

2.4 Sai số phương pháp, sai số tính toán và sự ổn định

2.4.1. Sai số phương pháp và sai số tính toán

Giả sử cần tính đại lượng A mà không có khả năng tính đúng. Bằng cách nào đó ta tính gần đúng được con số là a , thì $\Delta_a = |A - a|$ là sai số phương pháp. Trong quá trình tính toán ta cũng không thu được số a mà chỉ thu được số \tilde{a} do phải quy tròn số. Ta gọi $\theta_a = |\tilde{a} - a|$ là sai số tính toán. Số gần đúng cuối cùng thu được là $\tilde{a} \approx A$, khi đó sai số tuyệt đối giới hạn của A (bao gồm sai số phương pháp và sai số tính toán) sẽ là

$$\Delta_{\tilde{a}} = |\tilde{a} - A| \leq |\tilde{a} - a| + |a - A| = \theta_a + \Delta_a.$$

Như vậy, nếu chọn phương pháp tính đúng thì sai số phương pháp $\Delta_a = 0$, số gần đúng thu được chỉ là do quy tròn số trong quá trình tính toán.

Ví dụ 1.7. Tính $A = (\sqrt{2} - 1)^{10}$.

Giải: Ta đã biết (theo khai triển nhị thức Newton):

$$(\sqrt{2} - 1)^{10} = A = 3363 - 2378\sqrt{2}.$$

Do $\sqrt{2} = 1.414213563 \star \star$, kết quả tính theo bên trái, bên phải của A (đều là các phương pháp tính đúng) được cho trong bảng sau:

$\sqrt{2}$	Về trái của A	Về phải của A
1.4	0.0001048576	33.8
1.41	0.00013422659	10.2
1.414	0.00147912	0.508
1.41421	0.00014866399	0.00862
1.414213563	0.00014867678	0.0001472

Bảng 1.1

Nhìn vào bảng 1.1 ta thấy, cùng là phương pháp tính đúng nhưng hai bên trái, phải của A kết quả sai khác nhau quá nhiều, nguyên nhân là do sự quy tròn số sinh ra. Chọn cách tính này hay tính khác phải luôn luôn kiểm nghiệm theo thực tế vì $(\sqrt{2} - 1)^{10}$ không thể lớn hơn 1!

Ví dụ 1.8. Hãy tính đại lượng

$$S = \sum_{n=1}^{\infty} (-1)^{n-1} \frac{1}{n^2} = \frac{1}{1^2} - \frac{1}{2^2} + \frac{1}{3^2} - \cdots + (-1)^{n-1} \frac{1}{n^2} + \cdots$$

sao cho sai số tuyệt đối không vượt quá 4×10^{-3} .

Giải: Ta có $|S - S_n| = \left| \frac{1}{(n+1)^2} - \frac{1}{(n+2)^2} + \cdots \right|$ chính là sai số của phương pháp. Theo lý thuyết chuỗi đan dauen thì $|S - S_n| \leq \frac{1}{(n+1)^2}$.

Sai số cuối cùng là tổng của sai số phương pháp và sai số tính toán (do quy tròn số). Vì vậy, trước hết cần xác định số số hạng (n) cụ thể sao cho sai số phương pháp $|S - S_n|$ bé hơn nhiều so với 4×10^{-3} , chẳng hạn $|S - S_n| < 3 \times 10^{-3}$, phần còn lại là 10^{-3} dành cho sai số tính toán. Vậy n sẽ được chọn sao cho

$$|S - S_n| < \frac{1}{(n+1)^2} < 3 \times 10^{-3} \implies n = 6.$$

Khi đó

$$\begin{aligned}
 S_6 &= \frac{1}{1^2} - \frac{1}{2^2} + \frac{1}{3^2} - \frac{1}{4^2} + \frac{1}{5^2} - \frac{1}{6^2} \\
 &= 1.000 - 0.125 + (0.0037 \pm 10^{-4}) - (0.0016 \pm 4 \times 10^{-4}) + 0.008 \\
 &\quad - (0.005 \pm 4 \times 10^{-4}) \\
 &= 0.899 \pm 9 \times 10^{-4} = \tilde{S}_6 \pm 9 \times 10^{-4}.
 \end{aligned}$$

Vậy nếu xem $S \approx \tilde{S}_6 = 0.899$ thì ta có đánh giá

$$\begin{aligned}
 |S - \tilde{S}_6| &= |S - 0.899| \leq |S - S_6| + |S_6 - \tilde{S}_6| < 3 \times 10^{-3} + 9 \times 10^{-4} = \\
 &= (3 + 0.9) \times 10^{-3} < 4 \times 10^{-3}.
 \end{aligned}$$

Điều này đảm bảo yêu cầu đặt ra ban đầu.

2.4.2. Sự ổn định của quá trình tính

Để tính một đại lượng thường phải lặp đi lặp lại nhiều lần. Quá trình tính gọi là ổn định nếu sai số tính toán (quy tròn số) tích lũy lai không tăng ra vô hạn. Ngược lại quá trình tính đó gọi là không ổn định.

Thông thường người ta xét sai số ở một bước tính. Giả sử sau một số bước tính ta được con số, xem số đó là đúng, ta tính một số bước nữa thấy sai số tăng không đáng kể thì quá trình tính là ổn định, ngược lại là không ổn định. Nói chung, đây là một vấn đề khó, cần được nghiên cứu tiếp.

CHƯƠNG 2

MATLAB CƠ BẢN

THÔNG thường, để giải một bài toán thực tế bằng máy tính, ta phải lần lượt thực hiện một số bước sau:

1. Từ các số liệu thu được trong quá trình thực nghiệm, đo đạc,..., xây dựng mô hình toán học cho bài toán thực tế đặt ra.
2. Phân tích mô hình, tính tương thích của mô hình toán học với hiện tượng thực tế (sự tồn tại nghiệm, tính duy nhất nghiệm,...).
3. Rời rạc hóa mô hình để đưa việc tìm lời giải trên miền liên tục của đối số thành bài toán có số ẩn hữu hạn trên miền của đối số rời rạc.
4. Xây dựng thuật toán. Trong bước này cần chú ý đến độ phức tạp của thuật toán, tính hội tụ, sự ổn định của phương pháp giải bài toán.
5. Cài đặt thuật toán trên máy.

Để thuận tiện cho việc giải số các bài toán, trong giáo trình này chúng tôi giới thiệu ngôn ngữ lập trình được sử dụng rộng rãi trong môi trường khoa học và công nghệ - ngôn ngữ lập trình tính toán MATLAB. Do khuôn khổ có hạn, trong phạm vi giáo trình này chúng tôi chỉ giới thiệu những phần cơ bản nhất về MATLAB, đặc biệt là ứng dụng để xây dựng các chương trình minh họa cho các thuật toán trong phần phương pháp tính.

Để tìm hiểu thêm về MATLAB cũng như ứng dụng nó vào các lĩnh vực khác nhau, bạn đọc có thể tìm hiểu thêm tại các tài liệu tham khảo [8, 9].

§1. KHỞI ĐỘNG MATLAB

MATLAB (viết tắt của cụm từ tiếng Anh "MATrix LABoratory") được thiết kế bởi công ty MathWorks, là một ngôn ngữ lập trình bậc cao, chuyên sử dụng cho các tính toán kỹ thuật, đặc biệt là các bài toán có dạng ma trận hoặc vector. MATLAB tích hợp các phép tính toán, đồ họa và lập trình trong một môi trường thân thiện, cho phép thể hiện các bài toán cũng như nghiêm của chúng dưới dạng các ký hiệu toán học quen thuộc.

Sau khi cài đặt, cách đơn giản nhất để khởi động MATLAB là kích đúp vào biểu tượng của nó trên màn hình Desktop. Giao diện cơ bản của MATLAB phiên bản 7.10.0 (R2010a) (xem hình 2.1) bao gồm các phần sau:

- **Current Folder:** thư mục hiện thời. Khi muốn thực thi một tập tin ".m", người dùng phải chắc chắn rằng tập tin đó được chứa trong thư mục này.
- **Command Window:** Cửa sổ dòng lệnh thao tác trực tiếp. Ở chế độ này, sau khi gõ câu lệnh và thực thi (nhấn Enter), kết quả sẽ được hiển thị ngay trong cửa sổ lệnh. Nếu không muốn in kết quả ta thêm dấu ";" vào cuối câu lệnh đó.
- **Editor:** Soạn thảo chương trình do người dùng lập ra.
- **Workspace:** Chứa danh sách các biến đã được khai báo và sử dụng trong chương trình. Tại đây, ta có thể xem được tên, giá trị, kích thước và có thể sửa giá trị của biến.
- **Command History:** Chứa tất cả các câu lệnh đã được thực thi trong Command Window. Người dùng có thể kích đúp vào các câu lệnh bất kỳ để thực thi lại hoặc tạo các "m-files".



Hình 2.1: Giao diện MATLAB

§2. BIỂU THỨC MATLAB: BIẾN, SỐ, TOÁN TỬ, HÀM

2.1 Biến

Một điều thú vị trong MATLAB khác với các ngôn ngữ lập trình quen thuộc (Pascal, C,...) đó là MATLAB không yêu cầu phải khai báo trước tên biến cũng như kích thước của nó. Trong MATLAB, một biến được khai báo và khởi tạo thông qua lệnh gán, ví dụ:

```

>> num = 98
num =
98
>> pi = 3.1415926535897931
pi =
3.1416
>> msg = 'Hello World'
msg =
Hello World

```

Trong MATLAB tên biến có thể bao gồm các ký tự chữ, số và ký hiệu gạch dưới “_” nhưng **phải bắt đầu bằng ký tự chữ** và có độ dài tùy thích. Tuy nhiên, MATLAB chỉ sử dụng N ký tự đầu tiên được tính bằng lệnh

```
>> N = namelengthmax
N =
63
```

MATLAB phân biệt chữ hoa và chữ thường. Do đó A và a là các biến khác nhau. Khi MATLAB gặp một tên biến mới, nó sẽ tự động tạo ra và lưu biến đó trong bộ nhớ. Nếu biến đó đã tồn tại, MATLAB sẽ thay đổi giá trị và nếu cần, cấp phát bộ nhớ mới.

2.2 Số

MATLAB sử dụng ký hiệu số thập phân theo qui ước với số chữ số tùy chọn và các dấu +, - cho các số. Ký hiệu khoa học sử dụng chữ cái e cho lũy thừa của 10. Số phức sử dụng các chữ i hoặc j cho đơn vị ảo. Một số ví dụ:

3	-99	0.0001
9.6397238	1.60210e-20	6.02252e23
1i	-3.14159j	3e5i

Tất cả các số được lưu trữ bên trong bằng cách sử dụng long format theo chuẩn dấu chấm động IEEE (Institute of Electrical and Electronics Engineers). Các số dưới dạng dấu chấm động có độ chính xác hữu hạn với 16 chữ số thập phân có nghĩa và nằm trong khoảng $(10^{-308}, 10^{+308})$.

2.3 Toán tử

Các biểu thức MATLAB sử dụng các toán tử quen thuộc được liệt kê trong bảng dưới đây theo thứ tự ưu tiên (từ dưới lên trên).

Phép toán	Ý nghĩa
+	Cộng
-	Trừ
*	Nhân
/	Chia thông thường
\	Chia trái
^	Lũy thừa
'	Chuyển vị, chuyển vị liên hợp phức
()	Xác định thứ tự ưu tiên của các phép toán

2.4 Hàm

MATLAB cung cấp một số lượng rất phong phú các hàm toán học sơ cấp (elementary mathematical functions), ví dụ abs, sqrt, exp, sin,... Để hiển thị danh sách các hàm toán học sơ cấp, nhập vào lệnh

```
>> help elfun
```

MATLAB cũng đồng thời cung cấp rất nhiều các hàm toán học nâng cao (advanced mathematical functions), ví dụ các hàm Bessel và gamma. Hầu hết các hàm này chấp nhận đối số phức. Để liệt kê danh sách các hàm toán học nâng cao và hàm xử lý ma trận ta dùng các lệnh

```
>> help specfun
```

```
>> help elmat
```

Một số các hàm như sqrt, sin được cài đặt sẵn (built-in functions). Các hàm này là một phần của nhân MATLAB nên chúng rất hiệu quả, nhưng ta không biết được các tính toán chi tiết trong đó. Các hàm khác, ví dụ bessel được lập trình dưới dạng "m-files". Có một số sự khác biệt giữa các hàm được cài đặt sẵn và các hàm khác. Đó là, với các hàm "built-in", ta không thể xem mã, còn đối với các hàm khác ta có thể xem mã và thậm chí sửa đổi nếu muốn. Kiểm chứng điều này bằng các lệnh

```
>> type sqrt
```

```
>> type bessel
```

Nhiều hàm đặc biệt trong MATLAB cho ta giá trị của các hằng số hữu ích, được liệt kê trong bảng dưới đây:

pi	3.14159265...
i, j	Đơn vị ảo, $\sqrt{-1}$
eps	Sai số tương đối dạng dấu chấm động, $\varepsilon = 2.2204e - 016$
realmin	Số thực nhỏ nhất, $2.2251e - 308$
realmax	Số thực lớn nhất, $1.7977e + 308$
Inf	∞
NaN	Not-a-number

Trong đó,

- inf được tạo bởi phép chia một số khác 0 cho 0, hoặc việc tính giá trị của một biểu thức toán học đúng đắn mà bị tràn bộ nhớ, tức là vượt quá realmax.
- NaN được tạo ra khi tính giá trị của một biểu thức dạng vô định $\frac{0}{0}$ hoặc $\infty - \infty$.

Ta có thể gán cho các hằng giá trị mới, ví dụ:

```
>> eps = 1.e-6
```

và sử dụng giá trị này cho một dãy tính toán. Giá trị ban đầu của nó sẽ được phục hồi với lệnh

```
>> clear eps;
```

§3. CÁC DẠNG DỮ LIỆU CƠ BẢN TRONG MATLAB

3.1 Vector

3.1.1. Khởi tạo vector

Vector là một ma trận có một hàng hoặc một cột. Để khởi tạo vector hàng chứa các giá trị rời rạc, các phần tử trong vector phải nằm trong cặp ngoặc vuông [] và được ngăn cách bởi dấu phẩy "," hoặc khoảng trắng.

Ví dụ:

```
>> arr1 = [1 2 3]
arr1 =
    1     2     3
>> arr2 = [0, -5]
arr2 =
    0    -5
>> arr3 = [arr1 arr2]
arr3 =
    1     2     3     0    -5
```

Để khởi tạo vector hàng chứa các giá trị liên tiếp hoặc cách nhau một giá trị nhất định (bước nhảy), MATLAB sử dụng toán tử ":" và khi đó, giá trị đầu và cuối của vector không cần thiết phải đặt trong dấu ngoặc vuông []. Ví dụ:

```
>> arr1 = 1:5
arr1 =
    1     2     3     4     5
>> arr2 = [1:0.5:2]
arr2 =
    1.0000    1.5000    2.0000
>> arr3 = 10:-1:6
arr3 =
    10     9     8     7     6
>> 0: pi/4: pi
ans =
    0     0.7854    1.5708    2.3562    3.1416
```

Ngược lại, để tạo vector cột, ta cần chuyển vị vector hàng bằng cách dùng toán tử "'" hoặc dùng dấu ";" để ngăn cách các phần tử. Ví dụ:

```
>> col_arr=[1:3]'
col_arr =
    1
```

```

2
3
>> col_arr=[1;2;3]
col_arr =
1
2
3

```

Để tạo vector rỗng (không chứa phần tử nào) ta khai báo như sau:

```

>> emp_arr = []
emp_arr =
[]

```

3.1.2. Chỉ số

Giá trị của một phần tử tại một vị trí bất kỳ trong vector được truy xuất thông qua chỉ số. Trong MATLAB, chỉ số **luôn bắt đầu từ 1** và có thể là một giá trị đơn hoặc một mảng. Sau đây là qui tắc truy cập đến các phần tử của mảng x thông qua chỉ số.

- Trích phần tử thứ i: $x(i)$
- Trích nhiều phần tử: $x([danh sách các vị trí])$

```

>> arr = 10:-1:3
arr =
10      9      8      7      6      5      4      3
>> arr(5)
ans =
6
>> arr([4,5,8])
ans =
7      6      3

```

- Để xóa các phần tử trong vector, ta gán chúng với vector rỗng:

```
>> arr([2 5]) = []
arr =
    10      8      7      5      4      3
```

Biểu thức logic cho phép ta truy xuất một cách linh hoạt đến các thành phần của một vector hay ma trận. Ví dụ:

```
>> x = [-1 0 2 3 5 6 7 4 9];
>> x>0
ans =
    0      0      1      1      1      1      1      1      1
>> x(x>0)
ans
    2      3      5      6      7      4      9
>> x(x>2 & x<=5)
ans =
    3      5      4
>> x>2
ans =
    0      0      0      1      1      1      1      1      1
```

Biểu thức logic thường được sử dụng cùng với các hàm any, all và find, được mô tả chi tiết dưới đây.

- any: Kiểm tra xem có tồn tại một phần tử của vector thỏa mãn điều kiện nào đó không. Nếu có thì trả về 1, ngược lại là 0. Ví dụ:

```
>> x=[-1 2 3];
>> any(x>0)
ans =
    1
```

- all: Kiểm tra xem tất cả các phần tử của vector thỏa mãn điều kiện nào đó không. Ví dụ:

```
>> all(x<0)
```

```
ans =
```

```
0
```

- **find:** trả về các chỉ số của một vector thỏa mãn một điều kiện nào đó. Ví dụ:

```
>> A=[4 3 6 5 7 9];
>> find(isprime(A))
% trả về các vị trí có giá trị là một số nguyên tố
ans =
2      4      5
```

3.1.3. Các phép toán cơ bản trên vector

Giả sử a, b là hai vector có cùng kích thước. Khi đó ta có các phép toán cơ bản như sau:

```
a.*b; % nhân từng từ
a.^b % trả về vector dạng ( $a_1^{b_1}, \dots, a_n^{b_n}$ )
a.^n; % lũy thừa từng từ
a.\b; % chia trái từng từ
a./b; % chia phải từng từ
a & b; % phép hội từng từ
a | b; % phép tuyển từng từ
~a; % phủ định
sort(a); sort(a,'descend');
% sắp xếp mảng theo thứ tự tăng, giảm dần
arrayfun(@fn,a);
% tính giá trị hàm fn tại từng thành phần của a
isequal(a,b); % Đúng nếu  $a==b$ 
ismember(a,b);
% đúng khi mọi phần tử của a đều là phần tử của b
intersect(a,b); % phép giao 2 tập hợp
union(a,b); % phép hợp 2 tập hợp
setdiff(a,b); % hiệu 2 tập hợp
```

```

setxor(a,b);
% các phần tử không thuộc phần chung của a và b

```

3.2 Đa thức

Một đa thức bao giờ cũng có dạng $f(x) = a_nx^n + a_{n-1}x^{n-1} + \dots + a_1x + a_0$, trong đó nếu $a_n \neq 0$ thì n là bậc của đa thức và a_n, a_{n-1}, \dots, a_0 là các hệ số. Trong MATLAB, một đa thức được biểu diễn bởi một vector hàng có các thành phần là các hệ số theo thứ tự bậc giảm dần, kể cả hệ số 0. Việc biểu diễn này có ưu thế là rất gọn nhẹ và cũng thuận tiện cho việc thực hiện các phép toán đối với đa thức sau này. Ví dụ:

- $p = [8 \ 5]$ biểu diễn đa thức $8x + 5$;
- $h = [6 \ 0 \ -150]$ biểu diễn đa thức $6x^2 - 150$.

Sau đây, chúng ta xét một số phép toán cơ bản về đa thức trong MATLAB.

1. Tính giá trị của đa thức

Để tính giá trị của một đa thức tại điểm x ta sử dụng hàm

`polyval(p, x);`

trong đó,

- p là vector biểu diễn hệ số đa thức;
- x là một số, biến hoặc biểu thức.

Ví dụ:

```

>> p=[5 6 -7 3];
>> x=2;
>> y=polyval(p,x)
y =

```

2. Tìm nghiệm của đa thức

Nhắc lại rằng nghiệm của đa thức là các giá trị của biến sao cho giá trị của đa thức tại đó bằng 0. MATLAB có thể tìm các nghiệm của một đa thức bằng lệnh

```
r=roots(p);
```

trong đó,

- p là vector biểu diễn đa thức;
- r là vector cột chứa các nghiệm của đa thức.

3. Tìm đa thức khi biết trước các nghiệm

Cho trước các nghiệm của một đa thức, MATLAB có thể tính các hệ số của đa thức đó bằng lệnh

```
p=poly(r);
```

trong đó,

- r là vector hàng hoặc cột chứa các nghiệm của đa thức;
- p là vector hàng chứa các hệ số của đa thức.

4. Cộng đa thức

Để cộng, trừ hai đa thức trong MATLAB thì các vector hệ số cần phải cùng kích cỡ, do đó vector có độ dài ngắn hơn phải được thêm các phần tử 0 vào phía bên trái. Ví dụ, để cộng hai đa thức $f_1(x) = 3x^6 + 15x^5 - 10x^3 - 3x^2 + 15x - 40$ và $f_2(x) = 3x^3 - 2x - 6$ ta thực hiện như sau:

```
>> p1=[3 15 0 -10 -3 15 -40];
>> p2=[0 0 0 3 0 -2 -6];
>> p=p1+p2
p =
    3      15      0     -7     -3     13     -46
```

Đa thức kết quả $f(x) = 3x^6 + 15x^5 - 7x^3 - 3x^2 + 13x - 46$.

5. Nhân đa thức

Cú pháp:

`c=conv(a, b);`

trong đó,

- a và b là các vector hệ số của các đa thức
- c là vector hệ số của tích

6. Chia đa thức

Cú pháp:

`[q, r]=deconv(u, v);`

trong đó,

- u là vector hệ số của các đa thức bị chia;
- v là vector hệ số của các đa thức chia;
- q là vector hệ số của thương;
- r là vector hệ số của phần dư.

7. Dao hàm của đa thức

MATLAB có thể tính đạo hàm của đa thức bởi lệnh

`k=polyder(p);`

trong đó,

- p là vector hệ số của đa thức;
- k là vector hệ số của đạo hàm.

Ví dụ:

```
>> p=[3 -2 4];
>> k=polyder(p)
k =
```

6 -2

8. Nguyên hàm của đa thức

MATLAB có thể tính nguyên hàm của đa thức bởi lệnh

```
g=polyint (h, k);
```

trong đó,

- h là vector hệ số của đa thức;
- g là vector hệ số của nguyên hàm;
- k là hằng số tích phân, mặc định là 0.

3.3 Ma trận

3.3.1. Nhập ma trận

Khi nhập ma trận trong môi trường dòng lệnh, ta phải tuân theo các qui định sau:

- Ngăn cách các phần tử trên một hàng của ma trận bởi dấu "," hay khoảng trắng;
- Dùng dấu ";" để kết thúc một hàng;
- Bao các phần tử của ma trận bởi cặp dấu [].

Ví dụ:

```
>> A = [ 16 3 2 13 ; 5 10 11 8 ; 9 6 7 12 ; 4 15 14 1]
A =
    16      3      2      13
      5     10     11      8
      9      6      7     12
      4     15     14      1
```

Khi muốn nhập một ma trận có kích thước lớn thì cách nhập trực tiếp từ cửa sổ lệnh là không phù hợp, ta thường sử dụng hàm `load` hoặc tạo một "m-file". Hàm `load` đọc một file văn bản chứa các dữ liệu số. File văn bản phải được tổ chức như là một bảng chữ nhật của các số, cách nhau bởi

các khoảng trắng, mỗi hàng trên một dòng và số phần tử trên mỗi hàng là như nhau. Ví dụ, ta tạo file matrix.dat có nội dung sau:

```
16.0 3.0 2.0 13.0
5.0 10.0 11.0 8.0
9.0 6.0 7.0 12.0
4.0 15.0 14.0 1.0
```

Khi đó, lệnh

```
>> load matrix.dat
```

sẽ đọc file và tạo ra một biến matrix chứa các phần tử như trên.

Ta có thể tạo ra các ma trận bằng cách sử dụng các file văn bản chứa mã MATLAB ("m-files"). Sử dụng trình soạn thảo **Matlab Editor** hoặc một trình soạn thảo bất kỳ tạo ra một file chứa các lệnh giống như dùng trong môi trường dòng lệnh MATLAB, sau đó lưu file này dưới dạng ".m". Ví dụ, tạo ra một file chứa 5 dòng sau:

```
A = [ ...
16.0 3.0 2.0 13.0
5.0 10.0 11.0 8.0
9.0 6.0 7.0 12.0
4.0 15.0 14.0 1.0 ];
```

Lưu file trên dưới tên matrix.m. Khi đó lệnh

```
>> matrix
```

sẽ đọc file và tạo ra một biến A có các phần tử như trên.

Ngoài ra, MATLAB cũng cung cấp các hàm để tạo các ma trận cơ bản, được liệt kê trong bảng sau:

<code>zeros</code>	Tất cả các phần tử bằng 0
<code>ones</code>	Tất cả các phần tử bằng 1
<code>rand</code>	Các phần tử có phân bố đều trên $[0, 1]$
<code>randn</code>	Các phần tử có phân bố chuẩn
<code>magic(n)</code>	Tạo ra ma trận cấp n gồm các số nguyên từ 1 đến n^2 với tổng các hàng bằng tổng các cột $n \geq 3$
<code>pascal(n)</code>	Tạo ra ma trận có các phần tử lấy từ tam giác Pascal
<code>eye(n)</code>	Tạo ma trận đơn vị cấp n

Sau đây là một số ví dụ:

```
>> Z=zeros(2,4)
Z =
    0     0     0     0
    0     0     0     0

>> F=5*ones(3)
F =
    5     5     5
    5     5     5
    5     5     5

>> R=randn(4)
R =
    0.5377    0.3188    3.5784    0.7254
    1.8339   -1.3077    2.7694   -0.0631
   -2.2588   -0.4336   -1.3499    0.7147
    0.8622    0.3426    3.0349   -0.2050
```

3.3.2. Chỉ số

Sau đây là quy tắc truy cập đến các phần tử của ma trận thông qua chỉ số.

- Phần tử ở hàng i , cột j của ma trận ($cỡ m \times n$) A là $A(i, j)$.
- Ta cũng có thể tham chiếu tới phần tử của mảng nhờ một chỉ số. Ví dụ phần tử $A(i, j)$ ứng với $A(k)$ với $k = i + (j - 1)m$ (duyệt theo cột),

từ trên xuống dưới, từ trái qua phải). Để chuyển từ chỉ số ma trận sang chỉ số mảng một chiều dùng lệnh

```
>> k=sub2ind(size(A), i, j);
```

Ngược lại, để chuyển từ chỉ số mảng một chiều sang chỉ số ma trận, ta dùng hàm `ind2sub` như sau:

```
>> [i, j]=ind2sub(size(A), k);
```

- Trong MATLAB, chỉ số cuối cùng của hàng hay cột của ma trận hoặc vector có thể thay thế bởi `end`. Ví dụ:

```
>> x=[1 2 3; 4 5 6];
>> y=x(1:end, 1:end-1)
y =
    1      2
    4      5
```

Để xác định kích thước của một ma trận ta có thể dùng lệnh `length` (trả về kích thước lớn nhất) hay lệnh `size` (số hàng và cột). Ví dụ:

```
>> c = [1 2 3 4; 5 6 7 8];
>> length(c)
ans =
    4
>> [m, n] = size(c)
m =
    2
n =
    4
```

Các lệnh tính kích thước của ma trận được liệt kê dưới bảng sau:

whos	Hiển thị các biến trong không gian làm việc cùng kích cỡ tương ứng
s = size(A)	Trả về là vector hàng s, s(1) - số hàng và s(2) - số cột
[r, c] = size(A)	Trả về hai số r, c ứng với số hàng và số cột
r = size(A, 1)	Trả về số hàng của A
c = size(A, 2)	Trả về số cột của A
n = length(A)	Trả về max(size(A)) khi A khác []

Các biểu thức chỉ số có thể tham chiếu đến một phần của ma trận. Ví dụ, viết $A(1:k, j)$ là tham chiếu đến k phần tử đầu tiên của cột j của ma trận. Ngoài ra toán tử ":" tham chiếu tới **tất cả các phần tử** trong một hàng hay một cột. Ví dụ:

```
>> A(:, 3) % Trả về cột thứ 3 của ma trận
ans =
2
11
7
14
```

và:

```
>> A(3, :) % Trả về hàng thứ 3 của ma trận
ans =
9     6     7     12
```

Viết $B = A(:, [1 3 2 4])$ sẽ tạo ra ma trận B bằng cách đổi thứ tự các cột từ $[1 2 3 4]$ thành $[1 3 2 4]$

```
>> B=A(:, [1 3 2 4])
B =
16      2      3      13
      5     11     10      8
      9      7      6     12
      4     14     15      1
```

3.3.3. Phép ghép nối các ma trận

Trong MATLAB ta có thể ghép nối các ma trận nhỏ để tạo thành các ma trận lớn hơn một cách rất đơn giản. Cụ thể, với hai ma trận A, B, ta có thể có các cách ghép nối như sau:

- $C=[A \ B]$ (với điều kiện các ma trận A, B phải có cùng số hàng);
- $D=[A; \ B]$ (với điều kiện các ma trận A, B phải có cùng số cột).

Ví dụ:

```
>> A=ones(3)
A =
    1     1     1
    1     1     1
    1     1     1
>> B=[A A+3; A+4 A+6]
B =
    1     1     1     4     4     4
    1     1     1     4     4     4
    1     1     1     4     4     4
    5     5     5     7     7     7
    5     5     5     7     7     7
    5     5     5     7     7     7
```

Ta có thể xóa hàng và cột của ma trận bằng cách gán cho chúng giá trị `[]`. Ví dụ:

```
>> A=[1 2 3; 4 5 6; 7 8 9]
A =
    1     2     3
    4     5     6
    7     8     9
>> X=A;
```

Để xóa cột thứ 2 của x , dùng lệnh:

```
>> X(:, 2) = []
```

$X =$

1	3
4	6
7	9

3.3.4. Một số lệnh xử lý ma trận

Cộng	$X = A + B$
Trừ	$X = A - B$
Nhân ma trận	$X = A * B$
Nhân các phần tử tương ứng với nhau	$X = A.*B$
Chia	$X = A/B$, khi đó $X*A = B$
Chia trái	$X = A\B$, khi đó $A*X = B$
Chia các phần tử tương ứng cho nhau	$X = A./B$
Lũy thừa	$X = A^2$
Lũy thừa từng từ	$X = A.^2$
Chuyển vị (liên hợp đối với ma trận phức)	$X = A'$
Chuyển vị (không liên hợp)	$X = A.'$
Nghịch đảo	$X = \text{inv}(A)$
Dịnh thức	$d = \det(A)$
Hệ đại số tuyến tính $Ax = b$	Nghiệm $x = A\b$
Phân tích Cholesky	$R = \text{chol}(A)$
Phân tích LU	$[L, U] = \text{lu}(A)$
Phân tích QR	$[Q, R] = \text{qr}(A)$
Giá trị riêng, vector riêng	$\text{eig}(A)$, $[d, r] = \text{eig}(A)$
Quay ma trận	$B = \text{rot90}(A)$
Đảo ma trận từ trái sang phải	$C = \text{fliplr}(A)$
Đảo ma trận từ trên xuống dưới	$D = \text{flipud}(A)$
Định dạng lại ma trận A với số hàng mới m và số cột mới n	$\text{reshape}(A, m, n)$

Lấy các phần tử trên đường chéo chính và lưu thành một vector	<code>diag(A)</code>
Chọn đường chéo tùy theo giá trị của k	<code>diag(A, k)</code>
$k = 0$	chọn đường chéo chính;
$k > 0$	chọn đường chéo thứ k phía trên đường chéo chính;
$k < 0$	chọn đường chéo thứ k phía dưới đường chéo chính;

§4. VẼ ĐỒ THỊ TRONG MATLAB

4.1 Đồ thị 2D

Lệnh cơ bản để vẽ đồ thị một hàm số trong không gian hai chiều:

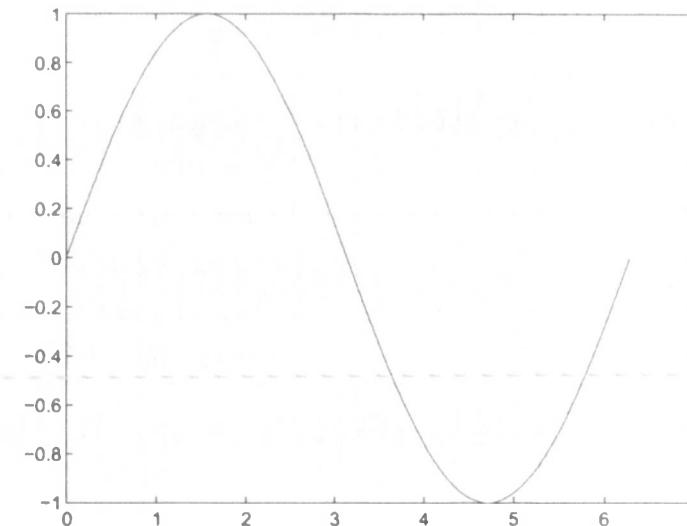
```
>> plot(x, f(x))
```

trong đó, x là vector chứa miền xác định của hàm có biểu thức là $f(x)$.

Ví dụ 2.1. Vẽ đồ thị hàm số $y = \sin(x)$ với x biến thiên trong khoảng $[0, 2\pi]$:

```
x = 0:pi/100: 2*pi;
y = sin(x);
plot(x, y);
```

Ta có thể chú thích thêm cho đồ thi bằng cách dùng các lệnh trong bảng dưới đây.

Hình 2.2: Đồ thị hàm $y = \sin(x)$

```

text(x, y, '...')      % Đặt chú thích lên đồ thi tại tọa độ (x, y)
gtext('...')            % Đặt chú thích lên đồ thi,
title('...')            % vị trí được xác định bởi click chuột
legend('...','...',...) % Tiêu đề của đồ thi
xlabel('...')           % Thêm chú giải cho đồ thi
ylabel('...')           % Ghi nhãn cho trục Ox
\bf                      % Ghi nhãn cho trục Oy
\it                      % Font in đậm
\rm                      % Font in nghiêng
\rm                      % Font chữ thường
hold on/off             % Bật/tắt chế độ cho phép vẽ nhiều đồ thi
                        % trong cùng một hệ trục tọa độ

```

<code>text(x, y, '...')</code>	Đặt chú thích lên đồ thi tại tọa độ (x, y)
<code>gtext('...')</code>	Đặt chú thích lên đồ thi, vị trí được xác định bởi click chuột
<code>title('...')</code>	Tiêu đề của đồ thi
<code>legend('...','...',...)</code>	Thêm chú giải cho đồ thi
<code>xlabel('...')</code>	Ghi nhãn cho trục O_x
<code>ylabel('...')</code>	Ghi nhãn cho trục O_y
<code>\bf</code>	Font in đậm
<code>\it</code>	Font in nghiêng
<code>\rm</code>	Font chữ thường
<code>hold on/off</code>	Bật/tắt chế độ cho phép vẽ nhiều đồ thi trong cùng một hệ trục tọa độ

Ta có thể chọn các tùy chỉnh nét vẽ, dấu và màu sắc bằng lệnh

```
>> plot(x,y,'color_style_marker');
```

Trong đó:

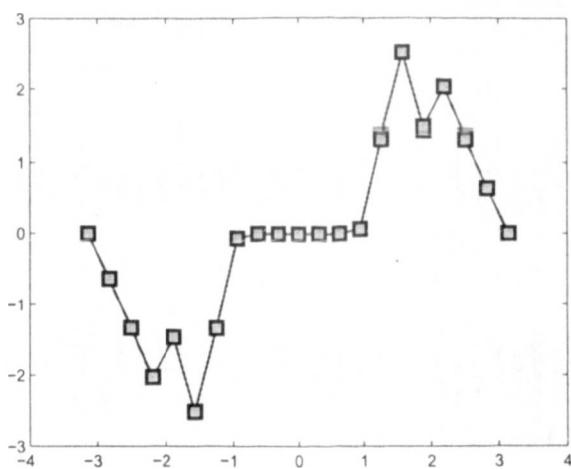
- Màu sắc (color): 'c' -cyan, 'm' -tím (magenta), 'y' -vàng (yellow), 'r' -đỏ (red), 'g' -xanh lá cây (green), 'b' -xanh nước biển (blue), 'w' -trắng (white) và 'k' -đen (black).
- Nét vẽ (style): '-' : nét liền, '--' : nét đứt, ':' : chấm chấm, '-.' : gạch chấm.
- Dấu (marker): '+', 'o', '*' và 'x'; 's': □, 'd': ♦, '^': ▲, 'v': ▼, '>': ▶, '<': ◀, 'p': ★, 'h': ngôi sao 6 cạnh.

Các tùy chỉnh màu sắc và độ rộng của nét vẽ

LineWidth	Độ rộng của nét vẽ, tính bằng pt
MarkerEdgecolor	Màu sắc của đường viền dấu
MarkerFacecolor	Màu bên trong dấu
Markersize	Độ lớn của dấu, tính bằng pt

Ví dụ 2.2. $x = -\pi : \pi / 10 : \pi$;

```
y = tan(sin(x)) - sin(tan(x));
plot(x,y,'-rs','LineWidth',2,'MarkerEdgecolor','k',...
'MarkerFacecolor','g', 'Markersize',10)
```



Hình 2.3:

Để xác định tọa độ, tùy chỉnh các kiểu tọa độ, ta có thể dùng các lệnh sau:

```
axis([xmin xmax ymin ymax])
xlim([xmin xmax])
ylim([ymin ymax])
axis on/off/auto
axis normal/square/equal/tight
axis ij/xy
grid on/off
```

Để vẽ nhiều đồ thị trong cùng một cửa sổ, ta có thể dùng lệnh:

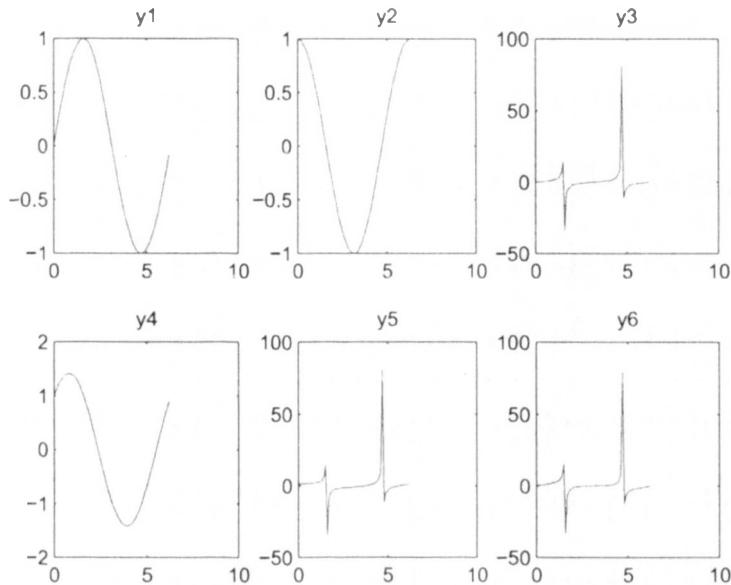
```
>> subplot(m, n, p);
```

Lệnh trên sẽ tạo ra một ma trận m hàng, n cột chứa $m \times n$ đồ thị, p là vị trí của từng đồ thị, thứ tự từ trên xuống dưới.

Ví dụ 2.3. Vẽ 6 đồ thị trong cùng một cửa sổ.

```
>> t=0:0.1:2*pi;
>> y1=sin(t); y2=cos(t); y3=tan(t);
>> y4=y1+y2; y5=y2+y3; y6=y1+y3;
>> subplot(2,3,1)
>> plot(t,y1)
>> title('y1')
>> subplot(2,3,2)
>> plot(t,y2)
>> title('y2')
>> subplot(2,3,3)
>> plot(t,y3)
>> title('y3')
>> subplot(2,3,4)
>> plot(t,y4)
>> title('y4')
>> subplot(2,3,5)
```

```
>> plot(t,y5)
>> title('y5')
>> subplot(2,3,6)
>> plot(t,y6)
>> title('y6')
```



Hình 2.4:

4.2 Đồ thị 3D

Lệnh cơ bản

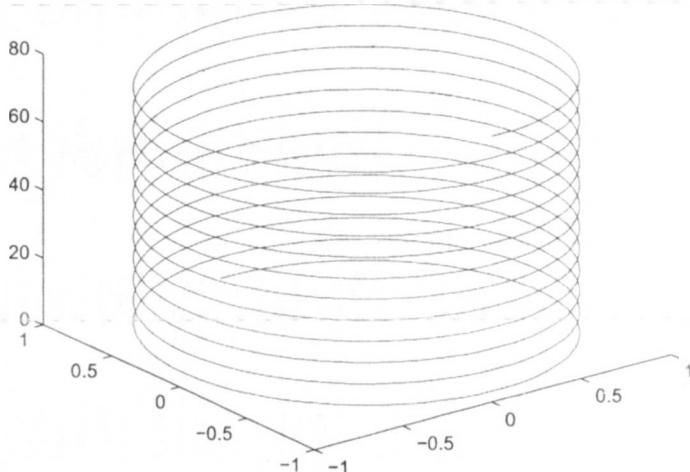
```
>> plot3(x, y, z)
```

Trong plot3, ta cần xác định các vector (x, y, z). Để vẽ mặt $z = f(x, y)$ ta thường sử dụng lệnh

```
>> meshgrid(x, y)
```

Ví dụ 2.4. (hình 2.5)

```
>> t = 0:0.02*pi:25*pi;
>> x = sin(t); y = cos(t);
>> z = t;
>> plot3(x,y,z);
```



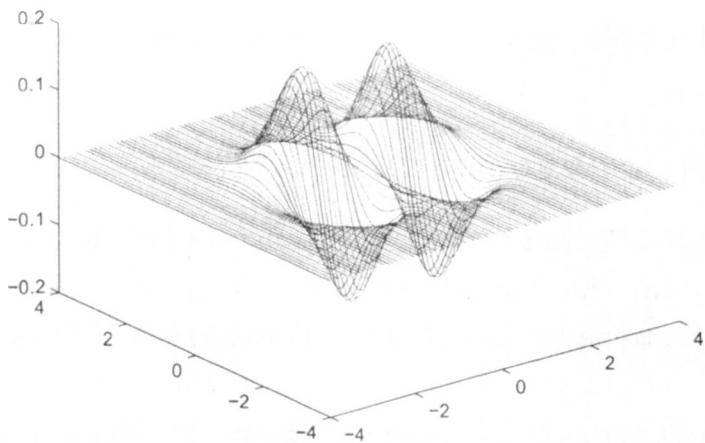
Hình 2.5:

Ví dụ 2.5. Vẽ mặt $z(x, y) = x^2ye^{-x^2-y^2}$ với $-4 \leq x \leq 4$; $-4 \leq y \leq 4$ (hình 2.6).

```
[x, y] = meshgrid([-4:0.1:4]);
z = x.*x.*y.*exp(-x.^2-y.^2);
plot3(x, y, z)
```

Bạn đọc có thể tham khảo một số lệnh khác (đọc help!).

- contour / contourf / contour3
- mesh / meshc / meshz



Hình 2.6:

- `surf / surf`
- `waterfall`
- `bar3 / bar3h`
- `pie3 / fill3`
- `comet3 / scatter3 / stem3`

Để in và xuất đồ thị dưới các định dạng khác nhau, cách thông dụng nhất là dùng **Plotting Tools**.

§5. LẬP TRÌNH VỚI MATLAB

Cũng giống như các ngôn ngữ lập trình khác, một thủ tục chuẩn của việc sử dụng lập trình MATLAB để giải quyết một bài toán kỹ thuật bao gồm các bước:

1. Phân tích bài toán và xác định sơ đồ giải (trên giấy);

2. Phác thảo các công thức tính toán (trên giấy);
3. Viết chương trình MATLAB ("m-files") sử dụng MATLAB Editor/Debugger;
4. Kiểm nghiệm và sửa lỗi;
5. Khai thác chương trình để giải bài toán.

Các chương trình MATLAB được chứa trong các "m-files" (phải có đuôi ".m"), là các file văn bản thông thường, không phải file nhị phân và phải được đặt trong đường dẫn hiện thời trong cửa sổ Command Window. Có hai loại chương trình MATLAB là thủ tục (script) và hàm (function). MATLAB quản lý đường dẫn trong của nó. Một chương trình có thể tồn tại và không có lỗi nhưng có thể vẫn không chạy nếu MATLAB không tìm thấy nó. Ta có thể thay đổi đường dẫn bằng cách dùng các lệnh path, addpath và rmpath.

5.1 Thủ tục (Script)

Các thủ tục không thực sự là các chương trình bởi vì chúng không có các dữ liệu input/output. Về bản chất, thủ tục bao gồm một khối các câu lệnh được thực hiện một cách tuần tự khi ta gọi tên thủ tục đó. Một điều cần lưu ý khi sử dụng thủ tục, đó là các biến được dùng trong thủ tục là một phần của không gian làm việc. Các thủ tục trong MATLAB luôn tạo ra các tác dụng phụ, lý do là bởi tất cả các biến được tạo ra trong thủ tục sẽ được thêm vào không gian làm việc mà không đưa ra khuyến cáo nào cả. Điều này sẽ có ảnh hưởng đáng kể bởi vì các biến đã tồn tại trong không gian làm việc có thể bị viết chồng lên.

Ví dụ 2.6. Xét thủ tục easyplot:

```
% Load
D=load('xy.dat'); % D is a matrix with two columns
x=D(:,1); % x is the first column,
y=D(:,2); % y is second one
plot(x,y) % Generate the plot and label it
```

```
xlabel('x axis')
ylabel('y axis')
title('Plot of generic x-y data set')
```

Thủ tục `easyplot` khi được thực thi sẽ tác động lên không gian làm việc bằng cách tạo ra ba biến `D`, `x` và `y`. Cụ thể:

```
>> clear
>> who
(Không có biến nào trong Workspace trước khi chạy thủ tục.)
>> easyplot
>> who
Your variables are:
D  x  y
```

Trong MATLAB thì các hàm có rất nhiều tiện ích so với các thủ tục, do đó các chuyên gia đều khuyến cáo rằng luôn luôn sử dụng hàm thay cho thủ tục.

5.2 Hàm "m-files" (Function)

Trong MATLAB thì bắt buộc tên hàm phải trùng với tên của file có đuôi ".m". Hàm là một chương trình sử dụng các tham số đầu vào/ra để kết hợp chúng với các hàm khác và các lệnh trong Command Window. Các biến được sử dụng trong hàm là các biến địa phương (local variables), chỉ tồn tại khi hàm đang thực thi. Các biến địa phương này được phân biệt với các biến trùng tên trong không gian làm việc hoặc của các hàm khác. Các dữ liệu đầu vào của hàm cho phép cùng một thủ tục tính toán (cùng thuật toán) có thể được áp dụng với các dữ liệu khác nhau. Do đó, các hàm "m-files" có thể dùng lại nhiều lần. Trong MATLAB thì các hàm có thể gọi các hàm khác đồng thời các thủ tục riêng có thể gói vào trong một hàm. Các tiếp cận này cho phép phát triển lời giải cấu trúc của các bài toán phức tạp.

Về cú pháp, dòng đầu tiên của hàm "m-file" bao giờ cũng có dạng:

```
function [outArgs]=funName(inArgs)
```

trong đó `outArgs` là danh sách các biến đầu ra, được đặt trong `[]`, cách nhau bởi dấu `,`. Nếu chỉ có một tham số đầu ra thì dấu `[]` là tùy chọn. Hàm mà không có `outArgs` vẫn là hợp lệ. Trong khi đó, danh sách các biến đầu vào `inArgs` được đặt trong `()`, cách nhau bởi dấu `,`. Hàm mà không có `inArgs` vẫn là hợp lệ. Có thể kiểm tra tính hợp lệ của tên hàm bằng cách dùng lệnh

```
>> isvarname functionName
```

Xét ví dụ về các hàm:

`twosum.m: two inputs, no output`

```
function twosum(x,y)
% twosum Add two matrices and print the result
x+y
```

`threesum.m: three inputs, one output`

```
function s=threesum(x,y,z)
% threesum Add three matrices and return the result
s=x+y+z;
```

`addmult.m: two inputs, two outputs`

```
function [s,p]=addmult(x,y)
% addmult Compute sum and product of two matrices
s=x+y;
p=x*y;
```

Bây giờ, ta xét cách thực thi hàm `twosum`:

```
>> twosum(2,2)
ans =
4
>> x=[1 2]; y=[3 4];
>> twosum(x,y)
ans =
```

```

>> A = [1 2; 3 4]; B = [5 6; 7 8];
>> twosum(A, B);
ans =
      6      8
     10     12

>> clear
>> x = 4; y = -2;
>> twosum(1, 2)
ans =
      3

>> x+y
ans =
      2

>> disp([x y])
      4      -2

>> who
Your variables are:
ans  x  y

```

Trong ví dụ trên, các biến x và y được khai báo trong không gian làm việc khác với các biến x , y được xác định trong hàm `twosum`. Các biến x , y trong `twosum` là các biến địa phương trong hàm này.

Có thể tóm tắt về các tham số Input và Output và việc quản lý chúng trong hàm như sau:

- Các giá trị được kết hợp thông qua các dữ liệu input và output.
- Các biến được định nghĩa trong một hàm là biến địa phương. Các hàm khác và môi trường cửa sổ lệnh sẽ không "nhìn" được chúng.
- Khi gọi hàm để thực thi thì số lượng các biến trả về nên trùng với số lượng các biến output trong hàm.
- Mỗi hàm có các biến nội tại bao gồm `nargin` (number of input arguments) và `nargout` (number of output arguments);

- Sử dụng giá trị nargin trong phần đầu của hàm để xác định có bao nhiêu biến đầu vào sẽ được sử dụng.
- Sử dụng giá trị nargout trong phần cuối của hàm để xác định số biến đầu ra mong muốn.

Lợi ích của công việc trên là cho phép một chương trình đơn có thể thực hiện nhiều công việc liên quan, đồng thời cho phép các hàm đặt các giá trị mặc định của một số biến đầu vào, do đó làm đơn giản việc sử dụng hàm trong một số trường hợp.

Ví dụ 2.7. Xét hàm plot:

Câu lệnh gọi hàm	nargin	nargout
plot(x, y)	2	0
plot(x, y, 's')	3	0
plot(x, y, 's--')	3	0
plot(x1, y1, 's', x2, y2, 'o')	6	0
h=plot(x, y)	2	1

Các giá trị của nargin và nargout được xác định khi hàm plot được gọi ra.

5.3 Nhập, xuất dữ liệu

Tiếp theo, chúng ta sẽ xét cách nhập, xuất dữ liệu trong MATLAB. Để nhập dữ liệu từ bàn phím ta có thể dùng hàm input, ví dụ:

```
>> x=input('Enter your age:') % nhập một số
>> s=input('Enter your name:','s') % nhập một xâu
```

Tuy nhiên cách nhập các dữ liệu như là tham số đầu vào của các hàm được ưa dùng hơn.

Các hàm xuất dữ liệu thông dụng trong MATLAB bao gồm hàm disp và fprintf. Trong khi hàm disp thường được sử dụng để xuất dữ liệu trong cửa sổ lệnh và chỉ dùng cho các kết quả đơn giản thì hàm fprintf sử dụng cho các dữ liệu định dạng trước. Hơn nữa, khi muốn ghi dữ liệu vào file thì bắt buộc phải dùng hàm fprintf.

5.3.1. Hàm disp

Cú pháp hàm disp như sau:

```
>> disp(outMatrix)
```

trong đó, outMatrix có thể là ma trận số hoặc xâu.

Ví dụ 2.8. >> disp(5)

5

```
>> x = 1:3; disp(x)
```

1 2 3

```
>> y = 3-x; disp([x; y])
```

1 2 3

2 1 0

```
>> disp([x y])
```

1 2 3 2 1 0

```
>> disp([x' y])
```

??? Error using ==> horzcat

CAT arguments dimensions are not consistent.

Ví dụ 2.9. >> disp('Hello World!')

Hello World!

```
>> s='Have a nice day'; disp(s)
```

Have a nice day

```
>> t='You are using Matlab 7.10.0';
```

```
>> disp([s;t])
```

??? Error using ==> vertcat

CAT arguments dimensions are not consistent.

```
>> disp(char(s,t))
```

Have a nice day

```
You are using Matlab 7.10.0
```

Chú ý 2.1. Trong ví dụ trên, lệnh `disp([s;t])` xuất hiện lỗi bởi vì s có ít ký tự hơn t. Hàm `char` tạo một ma trận xâu bằng cách đặt mỗi input trên một dòng riêng và chèn thêm các khoảng trắng nếu cần.

```
>> S=char(s,t);
>> length(s), length(t), length(S(1,:))
ans =
15
ans =
27
ans =
27
```

Để tạo ra dữ liệu đầu ra được gán nhãn của một giá trị số, ta thường kết hợp hàm `disp` với hàm `num2str`. Cụ thể, lệnh

```
stringValue=num2str(numericValue)
```

sẽ chuyển `numericValue` thành một xâu biểu diễn giá trị số đó. Xét các ví dụ:

Ví dụ 2.10. `>> num2str(pi)`

```
ans =
3.1416
```

và

Ví dụ 2.11. `>> A=eye(3)`

```
A =
1     0     0
0     1     0
0     0     1
```

```
>> S=num2str(A)
```

```
S =
1 0 0
0 1 0
0 0 1
```

Mặc dù A và S có vẻ chứa cùng các giá trị, chúng không tương đương. A là một ma trận số còn S là ma trận xâu. Do đó, các phép toán ma trận đối với A và S là không thực hiện được. Ví dụ:

```
>> A-S
??? Error using ==> minus
Matrix dimensions must agree.
```

Tiếp theo, ta xét ví dụ về việc sử dụng num2str kết hợp với disp:

```
>> x=sqrt(2);
>> outString=['x=',num2str(x)];
>> disp(outString)
x=1.4142
```

hoặc

```
>> disp(['x=',num2str(x)])
x=1.4142
```

Chú ý 2.2. Cấu trúc

```
disp(['x=',num2str(x)])
```

chỉ làm việc khi x là một ma trận hàng còn với ma trận cột thì không. Ví dụ:

```
>> y=1:4;
>> z=y';
>> disp(['z=',num2str(z)])
??? Error using ==> horzcat
CAT arguments dimensions are not consistent.
```

Thay vào đó, ta có thể sử dụng hai lệnh disp để hiển thị cột của các vector hay ma trận, ví dụ:

```
>> disp('z='); disp(z)
z=
1
```

2
3
4

hoặc đơn giản là nhập vào tên của biến mà không có dấu ";" cuối dòng

```
>> z
z =
1
2
3
4
```

Ta có thể dùng hàm `format` điều chỉnh độ chính xác của dữ liệu in ra, ví dụ:

```
>> format short
>> disp(pi)
3.1416
>> format long
>> disp(pi)
3.141592653589793
```

Ngoài ra, thông số thứ hai của hàm `num2str` cũng có thể dùng với mục đích trên:

```
>> disp(['pi=', num2str(pi, 2)])
pi=3.1
>> disp(['pi=', num2str(pi, 4)])
pi=3.142
>> disp(['pi=', num2str(pi, 8)])
pi=3.1415927
```

5.3.2. Hàm `fprintf`

Quay trở lại hàm `fprintf`, cú pháp của hàm này như sau:

```
fprintf(outFormat, outVariables);
fprintf(fileHandle, outFormat, outVariables);
```

Hàm `fprintf` sử dụng `outFormat` để chuyển `outVariables` thành các xâu được in ra. Trong dạng đầu tiên, kết quả sẽ hiển thị trong cửa sổ lệnh. Trong dạng thứ hai, kết quả sẽ được lưu vào file được tham chiếu bởi `fileHandle`. Ví dụ:

```
>> x=3;
>> fprintf('Square root of %g is %8.6f\n',x,sqrt(x))
Square root of 3 is 1.732051
```

Thành phần `outFormat` định rõ cách các `outVariables` được chuyển thành và hiển thi. Xâu `outFormat` có thể chứa bất kỳ một ký tự nào. Nó cũng phải chứa một mã chuyển đổi cho mỗi `outVariables`. Các mã chuyển đổi cơ bản được cho dưới bảng sau:

Mã	Dạng
%s	Dạng xâu
%d	Dạng số nguyên
%f	Dạng dấu chấm động
%e	Dạng dấu chấm động trong ký hiệu khoa học
%g	Dạng gọn nhất của %f hoặc %e
\n	Chèn một dòng mới sau xâu kết quả
\t	Chèn một tab sau xâu kết quả

Ta có thể điều chỉnh độ rộng và độ chính xác của kết quả bằng các cú pháp:

```
% wd
% w.pf
% w.pe
```

trong đó `w` là số ký tự trong độ rộng của kết quả cuối cùng và `p` là số chữ số sau dấu `.` sẽ được hiển thị. Một số ví dụ

Giá trị	%8.4f	%12.3e	%10g	%8d
2	2.0000	2.000e+00	2	2
$\sqrt{2}$	1.4142	1.414e+00	1.41421	1.414214e+00
$\sqrt{2e-11}$	0.0000	4.472e-06	4.47214e-06	4.472136e-06
$\sqrt{2e11}$	447213.5955	4.472e+05	447214	4.472136e+05

Có thể dùng `fprintf` để in vector hoặc ma trận dưới dạng ngắn gọn. Điều này có thể dẫn tới các kết quả không như mong muốn. Ví dụ

```
>> x=1:4; y=sqrt(x);
>> fprintf('%9.4f\n',y)
1.0000
1.4142
1.7321
2.0000
```

Ở đây, định dạng `%9.4f` được sử dụng lại cho mỗi thành phần của `y`. Điều này có thể sẽ không cho kết quả như mong muốn:

```
>> fprintf('y=%9.4f\n',y)
y= 1.0000
y= 1.4142
y= 1.7321
y= 2.0000
```

Hàm `fprintf` duyệt các `outVariables` theo các cột. Điều này cũng có thể dẫn đến các kết quả không như mong muốn:

```
>> A=[1 2 3; 4 5 6; 7 8 9]
```

A =

1	2	3
4	5	6
7	8	9

```
>> fprintf(' %8.2f %8.2f % 8.2f \n', A )
    1.00      4.00      7.00
    2.00      5.00      8.00
    3.00      6.00      9.00
```

Để ghi dữ liệu ra file cần phải tạo ra một fileHandle với lệnh fopen. Khi đó, tất cả tác dụng của các định dạng cũng như vector hóa (xem mục 5.5) đều có thể được áp dụng.

Ví dụ 2.12. Lưu các thành phần của một vector vào một file.

```
x=1:10;
fout=fopen('out.dat','wt');
fprintf(fout,'    k    x(k)\n');
for k=1:length(x)
    fprintf(fout,'%4d    % 5.2f\n',k,x(k));
end
fclose(fout)
```

5.4 Điều khiển luồng

Để có thể thực thi một thuật toán, một ngôn ngữ lập trình cần có các cấu trúc điều khiển bao gồm:

- Cấu trúc tuần tự (Sequential structure);
- Cấu trúc lặp (Looping or Iteration structure);
- Cấu trúc điều kiện: rẽ nhánh (Branching structure).

5.4.1. So sánh (Comparison)

Khi sử dụng các cấu trúc điều kiện, ta thường gặp các phép so sánh. Sự so sánh được thể hiện qua các toán tử quan hệ (Relational Operators). Các toán tử này được dùng để kiểm tra hai giá trị bằng nhau, nhỏ hơn, lớn hơn, cụ thể:

Toán tử	Ý nghĩa
<	<
<=	\leq
>	>
>=	\geq
==	=
~=	\neq

Khi áp dụng các toán tử quan hệ thì kết quả sẽ là một giá trị logic, tức là *True* hoặc *False*. Trong MATLAB, các giá trị khác 0, bao gồm cả một xâu khác rỗng là tương đương với *True*. Chỉ có giá trị 0 là tương đương với *False*.

Chú ý 2.3. Trong các toán tử quan hệ $<=$, $>=$ và $~=$ thì ký hiệu " $=$ " phải đứng sau. Điều này có nghĩa $=<$, $=>$ và $=~$ là không hợp lệ.

Trong phép so sánh ta đặc biệt quan tâm đến hai loại toán tử là toán tử quan hệ và toán tử logic. Kết quả của một phép toán quan hệ là *True* (1) hoặc *False* (0), ví dụ:

```

>> a=3; b=5;
>> aIsSmaller=a<b
aIsSmaller =
    1
>> bIsSmaller=b<a
bIsSmaller =
    0
>> x=1:5; y=5:-1:1;
>> z=x>y
z =
    0      0      0      1      1

```

Các toán tử logic được sử dụng để kết hợp các biểu thức logic (với "and" và "or") hoặc thay đổi giá trị logic với "not".

Toán tử	Ý nghĩa
&&	and
	or
~	not

```
Ví dụ 2.13. >> a=3; b=5;
>> aIsSmaller=a<b; bIsSmaller=b<a;
>> bothTrue=aIsSmaller && bIsSmaller
bothTrue =
0
>> eitherTrue=aIsSmaller || bIsSmaller
eitherTrue =
1
>> ~eitherTrue
ans =
0
```

Ta có thể tóm tắt về các phép toán so sánh như sau:

- Các toán tử quan hệ liên quan đến các phép so sánh của hai giá trị.
- Kết quả của một phép toán quan hệ là một giá trị logic (True (1) hoặc False (0)).
- Các toán tử logic kết hợp (hoặc phủ định) các giá trị logic tạo ra các giá trị logic mới.
- Luôn có nhiều hơn một cách thể hiện cùng một phép so sánh.

Từ đó có một lời khuyên đối với người dùng, đó là để bắt đầu, nên tập trung vào các so sánh đơn giản.

5.4.2. Cấu trúc điều kiện hoặc rẽ nhánh

Dựa vào kết quả của một phép so sánh, hoặc của phép kiểm tra logic, các khối mã chương trình đã chọn sẽ được thực thi hoặc bỏ qua. Các cấu trúc điều kiện bao gồm: if, if...else, if...elseif và cấu trúc switch. Có 3 dạng của cấu trúc if là:

1. if
2. if...else
3. if...elseif

Cú pháp của cấu trúc if như sau:

```
if expression
    block of statements
end
```

Nguyên tắc hoạt động của cấu trúc trên có thể hiểu một cách đơn giản, đó là khối block of statements chỉ được thực thi nếu expression nhận giá trị *True*.

Ví dụ 2.14. if $a < 0$

```
    disp('a is negative');
end
```

Nếu muốn viết các câu lệnh trên trên cùng một dòng thì sau if expression cần có dấu "", ví dụ:

```
if a<0, disp('a is negative'); end
```

Cấu trúc if...else:

```
if x<0
    error('x is negative; sqrt(x) is imaginary');
else
    r=sqrt(x);
end
```

và cấu trúc if...elseif:

```
if x>0
    disp('x is positive');
elseif x<0
    disp('x is negative');
else
    disp('x is exactly zero');
end
```

Câu lệnh switch rất hữu dụng khi tập giá trị của các biến kiểm tra là rời rạc (có thể là số nguyên hay xâu ký tự). Cú pháp của câu lệnh này như sau:

```
switch    expression
    case value1
        block of statements
    case value2
        block of statements
    ...
    otherwise
        block of statements
end
```

Ví dụ 2.15. color=input('Enter your favorite color: ','s');
% color is a string
switch color
 case 'red'
 disp('Your color is red');
 case 'blue'
 disp('Your color is blue');
 case 'green'
 disp('Your color is green');
 otherwise
 disp('Your color is not red, blue or green');
end

5.4.3. Câu trúc lặp for

Cú pháp:

```
for index=expression
    block of statements
end
```

Ví dụ 2.16. Tính tổng các thành phần của một vector

```
x=1:5; % create a row vector
sumx=0; % initialize the sum
for k=1:length(x)
    sumx=sumx+x(k);
end
```

Ví dụ 2.17. Vòng lặp for với chỉ số tăng theo mức 2 đơn vị:

```
for k=1:2:n
    block of statements
end
```

Ví dụ 2.18. Vòng lặp for với chỉ số giảm dần:

```
for k=n:-1:1
    block of statements
end
```

Ví dụ 2.19. Vòng lặp for với chỉ số không phải là số nguyên:

```
for x=0:pi/15:pi
    fprintf('%8.2f %8.5f\n', x, sin(x));
end
```

Chú ý 2.4. Trong ví dụ trên, x là một đại lượng vô hướng trong vòng lặp. Mỗi lần lặp, x được gán với một trong các cột của 0:pi/15:pi.

5.4.4. Cấu trúc lặp while

Cú pháp:

```
while expression
    block of statements
end
```

Mô tả:

- Khối lệnh block of statements được thực thi nếu điều kiện expression vẫn là True.
- Để tránh tình trạng lặp vô hạn, nên đặt giới hạn trên cho số lần lặp.

5.4.5. Các câu lệnh break và return

- Các câu lệnh break và return là các cách khác nhau để thoát khỏi một cấu trúc lặp. Cả hai lệnh này đều có thể dùng cho cấu trúc for và while.
- break được sử dụng để thoát khỏi phạm vi của vòng lặp hiện thời for hoặc while, chương trình sẽ tiếp tục sau đó.
- return được dùng để thoát khỏi một hàm hiện thời. Điều này sẽ ảnh hưởng đến việc thoát khỏi một vòng lặp. Bất kỳ một câu lệnh nào tiếp theo vòng lặp trong hàm đều bị bỏ qua.

5.5 Vector hóa (Vectorization)

Vector hóa là việc sử dụng các phép toán vector để xử lý toàn bộ các phần tử của một vector hay ma trận. Thật ra các biểu thức vector hóa là tương đương với phép lặp trên các phần tử của ma trận hay vector. Tuy nhiên, biểu thức vector hóa sẽ ngắn gọn và thực thi nhanh hơn các biểu thức lặp thông thường. Một số chú ý khi sử dụng phép toán vector hóa:

- Sử dụng các phép toán vector thay cho vòng lặp khi có thể;
- Tiết kiệm bộ nhớ cho các vector hay ma trận;
- Sử dụng việc đánh chỉ mục vector hóa và các hàm logic ;
- Mã không sử dụng vector hóa gọi là mã vô hướng (scalar code) bởi vì các phép toán được thực hiện trên các phần tử vô hướng của vector hay ma trận thay vì toàn bộ.

Chương trình tuy chậm mà chính xác còn hơn chương trình nhanh mà không chính xác. Do đó, có một lời khuyên với người dùng là nên bắt đầu với các mã vô hướng, sau đó vector hóa nếu cần. Tiếp theo ta xét một ví dụ dùng vector hóa khi thay thế vòng lặp bởi các phép toán vector.

Ví dụ 2.20. Xét đoạn mã vô hướng:

```
x=...
for k=1:length(x)
    y(k)=sin(x(k));
end
```

Mã vector hóa tương đương:

```
x=...
y=sin(x);
```

Khi sử dụng phép toán vector hóa thì việc tiền cấp phát bộ nhớ là rất cần thiết. Nhắc lại là trong MATLAB các biến cũng như số chiều của chúng là không cần khai báo trước, MATLAB sẽ tự động điều chỉnh thông qua giá trị của các biến này. Ví dụ, vòng lặp sau sẽ tăng chiều của s sau mỗi lần lặp:

```
y=[4 -1 9 0];
for j=1:length(y)
    if y(j)>0
        s(j)=sqrt(y(j));
    else
        s(j)=0;
    end
end
```

Để tiền cấp phát cho s trước khi gán các giá trị cho các thành phần, ta làm như sau:

```
y=[4 -1 9 0];
s=zeros(size(y));
```

```

for j=1:length(y)
    if y(j)>0
        s(j)=sqrt(y(j));
    end
end

```

Việc vector hóa mã hoàn toàn đòi hỏi sử dụng việc đánh *chi số mảng* (array indexing) và đánh *chi số logic* (logical indexing). Xét ví dụ đánh chi số mảng:

```

>> x=sqrt(0:4:20)
x =
    0      2.0000      2.8284      3.4641      4.0000      4.4721
>> i=[1 2 5];
>> y=x(i)
y =
    0      2      4

```

Biểu thức $y=x(i)$ tương đương với đoạn mã vô hướng:

```

k=0;
for i=[1 2 5]
    k=k+1;
    y(k)=x(i);
end

```

Để dùng chỉ số logic ta làm như sau:

```

>> x=sqrt(0:4:20)
x =
    0      2.0000      2.8284      3.4641      4.0000      4.4721
>> j=find(rem(x,2)==0)
j =
    1      2      5
>> z=x(j)
z =
    0      2      4

```

Ví dụ 2.21. Vector hóa mā vô hướng.

Xét đoạn mã:

```
y=.
s=zeros(size(y));
for j=1:length(y)
    if y(j)>0
        s(j)=sqrt(y(j));
    end
end
```

Thực ra, có thể thay thế toàn bộ vòng lặp bằng cách sử dụng đánh chỉ số logic hoặc đánh chỉ số mảng, cụ thể:

```
y=.
s=zeros(size(y));
i=find(y>0);
s(i)=sqrt(y(i));
```

hoặc gọn hơn:

```
y=.
s=zeros(size(y));
s(y>0)=sqrt(y(y>0));
```

Tiếp theo, ta xét ví dụ sử dụng vector hóa trong các phép sao chép.

Xét ví dụ việc sao chép toàn bộ các cột (hàng) của ma trận A:

Mā vô hướng

```
[m,n]=size(A);
for i=1:m
    B(i,1)=A(i,1);
end
```

Mā vector hóa

```
B(:,1)=A(:,1);
```

Để sao chép và chuyển vị các ma trận con, ta thực hiện như sau:

Mã vô hướng

```
for j=2:3
    B(1, j)=A(j, 3);
end
```

Mã vector hóa

```
B(1, 2:3)=A(2:3, 3);
```

Một số ví dụ khác

1. Để xóa các thành phần không phải là số (NaN) hoặc giá trị vô cùng (inf) của một mảng x, ta có thể dùng đoạn mã sử dụng đánh chỉ số mảng:

```
i=find(isnan(x) | isinf(x)); % Find bad elements
x(i)=[ ]; % and delete them
```

hay một cách khác:

```
i=find(~isnan(x) & ~isinf(x)); % Find good elements
x=x(i); % Keep those elements
```

Ta có thể thay đổi các đoạn mã trên bằng cách sử dụng chỉ số logic như sau:

```
x(isnan(x) | isinf(x))=[]; % Delete bad elements
```

hoặc

```
x=x(~isnan(x) & ~isinf(x)); % Keep good elements
```

2. *Hàm từng khúc (Piecewise functions)*

Hàm sinc được định nghĩa bởi $\text{sinc}(x) = \begin{cases} \sin(x)/x, & x \neq 0 \\ 1, & x = 0. \end{cases}$

So sánh đoạn mã sử dụng lệnh find:

```
function y=sinc(x)
y=ones(size(x)); % Set y to all ones, sinc(0)=1;
i=find(x~=0); % Find nonzero x values
y(i)=sin(x(i))./x(i); % Compute sinc when x ~=0
end
```

và một cách viết thú vị khác:

```
y=(sin(x)+(x==0))./(x+(x==0));
```

5.6 Tính giá trị hàm một cách gián tiếp

Khi lập trình MATLAB, đặc biệt là trong phương pháp tính, ta thường phải tính giá trị của một hàm số tại một điểm nào đó. Hàm `feval` thường được sử dụng trong trường hợp này vì nó rất thuận lợi. Thứ nhất, nó cho phép các chương trình đã được viết xử lý một hàm $f(x)$ bất kỳ. Hàm này cũng đồng thời giúp ta có thể chia nhỏ một thuật toán phức tạp bằng cách sử dụng các đoạn mã riêng.

Ví dụ 2.22. `function s=fsum(fun,a,b,n)`

```
x=linspace(a,b,n);
y=feval(fun,x);
s=sum(y);
end
```

```
>> fsum('sin',0,pi,5)
```

```
ans =
```

```
2.4142
```

```
>> fsum('cos',0,pi,5)
```

```
ans =
```

```
0
```

5.7 Chú thích

- Cú pháp: % Matlab comment line.
- Các chú thích đặc biệt: Các khối comment liền nhau trong hàm m-file ngay sau dòng đầu tiên chính là phần help của m-file đó. Phần này sẽ được hiển thị khi ta nhập lệnh:

```
>> help filename
```

→ Khi viết một hàm m-file, có gắng thêm các chú thích để mô tả mục đích của hàm, yêu cầu về các biến input và định dạng của các biến output.

5.8 Gỡ lỗi

MATLAB hỗ trợ một trình gỡ lỗi tương tác, bao gồm các câu lệnh sau:

- Các lệnh type và dbtype hiển thị toàn bộ nội dung của một "m-file";
- Lệnh error hiển thị một lời nhắn trên màn hình và dừng hẳn chương trình;
- Hàm warning hiển thị một lời nhắn lên màn hình tuy nhiên không dừng chương trình;
- Các lệnh pause hoặc keyboard có thể dùng để tạm dừng chương trình. Để thoát khỏi chế độ gỡ lỗi (debug-mode) và tiếp tục chạy chương trình, ta có thể dùng một trong các lệnh return, dbcont, dbquit.

Ví dụ sử dụng lệnh keyboard

```
function r = quadroot(a,b,c)
% quadroot Roots of quadratic equation and demo of keyboard
% command
% Synopsis: r = quadroot(a,b,c)
%
% Input: a,b,c = coefficients of a*x^2 + b*x + c = 0
%
% Output: r=column vector containing real or complex roots
d = b^2 - 4*a*c;
if d<0
    fprintf('Warning in function QUADROOT:\n');
    fprintf('\tNegative discriminant\n\t...');
    Type "return" to continue\n';
end
```

```

    keyboard;
end
q = -0.5*( b + sign(b)*sqrt(b^2 - 4*a*c) );
r = [q/a; c/q]; % store roots in a column vector

```

5.9 Phân tích một chương trình sử dụng "The Profiler"

MATLAB phiên bản 5.0 hoặc mới hơn cung cấp một công cụ gọi là "profiler" hỗ trợ việc xác định các đoạn tắc nghẽn (bottlenecks) trong chương trình. Xét chương trình

```

function result=example1(Count)
for k=1:Count
    result(k)=sin(k/50);
    if result(k) < -0.9
        result(k)=gammaln(k);
    end
end
end

```

Để phân tích chương trình, trước hết dùng các lệnh sau để khởi động "profiler" và xóa tất cả các dữ liệu cũ:

```

>> profile on
>> profile clear

```

Bây giờ, chạy thử chương trình:

```
>> example1(50000);
```

Sau đó, nhập vào lệnh

```
>> profile report;
```

Profiler tạo một thông báo dạng HTML về chương trình và khởi tạo một cửa sổ trình duyệt. Tùy theo từng hệ thống máy tính mà các kết quả có thể hiển thị ở các dạng khác nhau.

Trong các chương tiếp theo, ta sẽ nêu các thuật toán thường được sử dụng trong kỹ thuật và các chương trình minh họa cụ thể.



Hình 2.7:

§6. BÀI TẬP

Bài tập 2.1. Tính bằng tay các biểu thức sau rồi thử lại bằng MATLAB.

1. $10/2\backslash 5 - 3 + 2 * 4$
2. $3^2/4$
3. $3^2 \cdot 2^2$
4. $2+\text{round}(6/9 + 3*2)/2-3$
5. $2+\text{floor}(6/11)/2-3$
6. $2+\text{ceil}(-6/9)-3$
7. $\text{fix}(-4/9)+\text{fix}(3*(5/6))$

Bài tập 2.2. Cho $a = 36, b = 15$. Tính bằng tay các biểu thức sau rồi thử lại bằng MATLAB.

1. $\text{mod}(a, b)$
2. $\text{rem}(a, b)$
3. $\text{gcd}(a, b)$
4. $\text{lcm}(a, b)$

Thử lại với các cặp giá trị (a, b) khác.

Bài tập 2.3. Dự đoán kết quả những phép toán sau, giải thích và kiểm tra lại bằng MATLAB.

1. $1 \& -1$ 4. $0 \leq 0.2 \leq 0.4$ 2. $13 \& (-6)$ 5. $5 > 4 > 3$ 3. $0 < -20$ 6. $2 > 3 \& 1$

Bài tập 2.4. Nhập vào vector $x = [3 1 5 7 9 2 6]$, dự đoán kết quả các dòng lệnh sau và thử lại bằng MATLAB.

1. $x(3)$ 5. $x(6:-2:1)$ 2. $x(1:7)$ 6. $x([1 6 2 1 1])$ 3. $x(1:end)$ 4. $x(1:end-1)$ 7. $\text{sum}(x), \text{min}(x), \text{max}(x)$.

Bài tập 2.5. Cho $x = [1 5 2 8 9 0 1]$ và $y = [5 2 2 6 0 0 2]$, giải thích kết quả các dòng lệnh sau:

1. $x>y$ 6. $x \mid y$ 2. $y<x$ 7. $x \& y$ 3. $x==y$ 8. $x \& (-y)$ 4. $x<=y$ 9. $(x>y) \mid (y<x)$ 5. $y>=x$ 10. $(x>y) \& (y<x)$

Bài tập 2.6. Cho hai vector $a = [1 0 2]$ và $b = [0 2 2]$, xác định giá trị các biểu thức sau, giải thích, sau đó kiểm tra lại bằng MATLAB.

1. $a=b$ 5. $a \mid (a)$ 2. $a<b$ 6. $b \& (b)$ 3. $a<b<a$ 7. $a=b==a$ (xác định giá trị cuối của a)4. $a<b<b$

Bài tập 2.7. Cho $x = 1:10$ và $y = [3 1 5 6 8 2 9 4 7 0]$, dự đoán kết quả các dòng lệnh sau, giải thích và thử lại bằng MATLAB.

1. $(x > 3) \& (x < 8)$

4. $x((x < 2) \mid (x \geq 8))$

2. $x(x > 5)$

5. $y((x < 2) \mid (x \geq 8))$

3. $y(x \leq 4)$

6. $x(y < 0)$

Bài tập 2.8. Cho $x = [1 \ 4 \ 8]$, $y = [2 \ 1 \ 5]$, và
 $A = [2 \ 7 \ 9 \ 7 ; 3 \ 1 \ 5 \ 6 ; 8 \ 1 \ 2 \ 5]$. Xét xem dòng lệnh nào hợp
lẽ, dự đoán kết quả, giải thích và thử lại bằng MATLAB.

1. $[x; y']$

5. $A(:)$

2. $[x; y]$

6. $[A; A(end, :)]$

3. $A(:, [1 \ 4])$

7. $A(1:3, :)$

4. $A([2 \ 3], [3 \ 1])$

8. $[A; A(1:2, :)]$

Bài tập 2.9. Cho $x = [2 \ -5 \ 1 \ 6]$. Viết các lệnh thực hiện:

1. Cộng thêm 16 vào tất cả các phần tử.

2. Cộng thêm 3 vào tất cả các phần tử ở vị trí lẻ.

3. Lấy căn bậc hai của tất cả các phần tử có giá trị dương.

4. Tính bình phương tất cả các phần tử.

Bài tập 2.10. Tao vector $x=\text{randperm}(35)$ (tao ngẫu nhiên 1 hoán vị từ 1 đến 35). Viết các lệnh để tính giá trị hàm sau sử dụng chỉ số logic:

$$y(x) = \begin{cases} 2, & \text{nếu } x < 6 \\ x - 4, & \text{nếu } 6 \leq x < 20 \\ 36 - x, & \text{nếu } 20 \leq x \leq 35 \end{cases}$$

Bài tập 2.11. Cho $x = [3, 15, 9, 12, -1, 0, -12, 9, 6, 1]$.

1. Viết lệnh thực hiện chuyển các giá trị dương thành giá trị 0.

2. Chuyển các bội số của 3 thành số 3.

3. Nhân các giá trị chẵn cho x .
4. Gán cho vector y các giá trị lớn hơn 10 của x .
5. Chuyển các giá trị nhỏ hơn trung bình cộng thành giá trị 0.

Bài tập 2.12. Cho x, y là các vector cột

$x = [3 \ 2 \ 6 \ 8]', y = [4 \ 1 \ 3 \ 5]'$. Viết các lệnh thực hiện:

1. Lấy tổng các phần tử của x thêm vào từng phần tử của y .
2. Lũy thừa mỗi phần tử của x với số mũ là các phần tử của y .
3. Chia các phần tử của tương ứng của y và x .
4. Nhân các phần tử tương ứng của x và y , gán kết quả cho vector z .
5. Tính tổng các phần tử của z , gán kết quả cho w .
6. Tính $x.^*y-w$.
7. Tính tích vô hướng của x và y .

Bài tập 2.13. Cho $x = [1 \ 4 \ 8]$, $y = [2 \ 1 \ 5]$

và $A = [3 \ 1 \ 6 ; 5 \ 2 \ 7]$. Xét xem dòng lệnh nào hợp lệ, dự đoán kết quả, giải thích rồi thử lại bằng MATLAB.

- | | |
|-------------|--------------------|
| 1. $x + y$ | 4. $A = [x' \ y']$ |
| 2. $x + A$ | |
| 3. $x' + y$ | 5. $A = 3$ |

Bài tập 2.14. Cho $A = [2 \ 7 \ 9 \ 7 ; 3 \ 1 \ 5 \ 6 ; 8 \ 1 \ 2 \ 5]$, dự đoán kết quả các lệnh, giải thích rồi thử lại bằng MATLAB.

- | | |
|--|-----------------------|
| 1. A' | 3. $\text{sum}(A')$ |
| 2. $\text{sum}(A)$ | 4. $\text{sum}(A, 2)$ |
| 5. $[[A; \ \text{sum}(A)] \ [\text{sum}(A, 2); \ \text{sum}(A(:))]]$ | |

Bài tập 2.15. Tao ma trận A cỡ 4×4 có giá trị nguyên ngẫu nhiên nằm trong khoảng $[-10, 10]$, sau đó:

1. Cộng thêm 10 vào các phần tử ở dòng 1 và dòng 2, gán kết quả cho B.
2. Cộng thêm 10 vào các phần tử ở cột 1 và cột 4, gán kết quả cho C.
3. Tính nghịch đảo của mọi phần tử khác 0, gán kết quả cho D.
4. Lấy căn bậc hai mọi phần tử dương, gán kết quả cho E.

Bài tập 2.16. Cho ma trận $A = [2 \ 4 \ 1 \ ; \ 6 \ 7 \ 2 \ ; \ 3 \ 5 \ 9]$, viết các lệnh MATLAB thực hiện:

1. Gán cho vector x dòng thứ nhất của A.
2. Gán cho y hai dòng còn lại của A.
3. Tìm phần tử lớn nhất và nhỏ nhất của ma trận A.
4. Tính tổng tất cả các phần tử của A.

Bài tập 2.17. Nhập vào 2 ma trận

$$A = \begin{bmatrix} 2 & 1 & 0 \\ -2 & 5 & -1 \\ 3 & 4 & 9 \end{bmatrix}; \quad B = \begin{bmatrix} 3 & 1 & 2 \\ -1 & 3 & -2 \\ 3 & 4 & 5 \end{bmatrix}$$

Viết các lệnh thực hiện:

1. Tìm ma trận X sao cho $X^*B = A$.
2. Tìm ma trận X sao cho $X^*A = B$.
3. Xóa cột thứ hai của ma trận A.
4. Thêm cột thứ nhất của ma trận B vào sau cột cuối của ma trận A.

Bài tập 2.18. 1. Vẽ đồ thị của hàm số $y = x + \sin x$ theo dõi x, trong đó $x = -\pi : 0.1 : \pi$. Sau đó thêm tiêu đề cho đồ thị và gán nhãn cho các trục tọa độ.

2. Vẽ đồ thị của hàm số $y = x^2 \sin x$ theo dõi x , trong đó $x = -\pi : 0.1 : \pi$. Sau đó thêm tiêu đề cho đồ thị và gán nhãn cho các trục tọa độ.

Bài tập 2.19. Nhập vào các lệnh sau để định nghĩa và vẽ đồ thị hàm $g(x) = \exp(x)$:

```
syms x
g = exp(x)
ezplot(g)
```

Sau đó chỉnh lại miền xác định bằng cách nhập: `ezplot(g, [-2, 2])`

Bài tập 2.20. Vẽ đồ thị các hàm $y = \sin(x)$, $y = \sin(x + \pi/2)$, $y = \sin(x + \pi/3)$ trên cùng một hệ trục tọa độ với vector `x=linspace(0, 2*pi)`.

Bài tập 2.21. 1. Nhập vào lệnh sau và xem kết quả:

```
syms x y;
ezmesh(sin(x)*cos(y), [0, 10, 0, 10])
```

2. Kích vào **Tools**, sau đó kích vào **Rotate 3D**. Trò vào đồ thị, dùng chuột trái giữ và quay ảnh theo ý muốn.
3. Sử dụng lệnh `ezmesh` vẽ đồ thị hàm $f(x, y) = x^2 - y^2$ với $x \in [-2, 2]$, $y \in [-2, 2]$.
4. Vẽ đồ thị hàm $f(x, y) = \sin(x^5) * \cos y$ sử dụng miền mặc định.

Bài tập 2.22. 1. Nhập vào các lệnh sau

```
[X, Y] = meshgrid(-1:.2:1);
Z = X.^2 - Y.^2;
contour(Z)
```

Nhận xét về nhãn của các trục tọa độ. Thử lại bằng lệnh:
`contour(X, Y, Z)`

2. Thử các lệnh sau và cho biết kết quả:

```
contourf(X, Y, Z)
contour(X, Y, Z, 10)
contour(X, Y, Z, 20)
contourf(X, Y, Z, 20)
```

Bài tập 2.23. 1. Tạo một thủ tục MATLAB với tên 'test1.m' thực hiện các công việc sau:

- (a) Xóa tất cả các biến trong bộ nhớ;
- (b) Tao một vector hàng $x=1:0.1:10$;
- (c) Tính $y = \sin(x) - \cos(x)$;
- (d) Vẽ đồ thị của y theo đối x .

2. Viết thêm các chú thích và lưu chương trình của bạn trong thư mục làm việc. Sau đó:

- (a) Kiểm tra và gỡ lỗi nếu có.
- (b) Hiển thi các chú thích bằng lệnh

```
>> help test1
```

Bài tập 2.24. 1. Tạo một hàm MATLAB tên 'test2.m' với biến đầu vào (input) x và biến đầu ra y , tính toán và hiển thị lên màn hình.

- (a) Tính $y = \sin(x) - \cos(x)$;
- (b) Vẽ đồ thị của y theo đối x .

2. Viết thêm các chú thích và lưu chương trình của bạn trong thư mục làm việc. Sau đó:

- (a) Kiểm tra chương trình với $x=1:0.1:10$, gỡ lỗi nếu có;
- (b) Hiển thi các chú thích trong chương trình bởi lệnh .

```
>> help test2
```

3. Giải thích sự khác nhau giữa thủ tục và hàm.

Bài tập 2.25. Tính gần đúng số π sử dụng kết quả

$$\frac{\pi^2}{6} = \sum_{n=1}^{\infty} \frac{1}{n^2}$$

Muốn sai số là $1e-12$ thì cần ít nhất bao nhiêu số hạng?

Bài tập 2.26. Có một thuật toán khác để tính gần đúng số π như sau:

1. Đặt $a = 1; b = 1/\sqrt{2}; t = 1/4$; và $x = 1$;

2. Lặp lại các phép tính sau:

```
y = a;
a = (a + b)/2;
b = sqrt(b*y);
t = t - x*(y - a)^2;
x = 2*x;
```

cho đến khi $|a - b| < \varepsilon$ với sai số ε cho trước.

3. Từ các giá trị a, b, t , ước lượng giá trị π bởi

```
Pi_est = ((a + b)^2)/(4*t);
```

Hãy viết chương trình MATLAB thể hiện thuật toán trên. Cần bao nhiêu bước lặp để sai số là $1e-8, 1e-12$? So sánh với thuật toán trong bài tập 2.25.

Bài tập 2.27. Viết các chương trình tính $n!, (2n-1)!!$, $(2n)!!$.

Bài tập 2.28. Dãy Fibonacci là dãy số được xác định như sau:

$$F_1 = F_2 = 1; \quad F_n = F_{n-1} + F_{n-2}, \quad n \geq 3.$$

Viết các chương trình MATLAB:

1. Tính 10 số hạng đầu tiên của dãy Fibonacci.

2. Với 50 số hạng đầu tiên của dãy, tính tỷ số

$$\frac{F_n}{F_{n-1}}.$$

Người ta chứng minh được rằng tỷ số này dần tới tỷ lệ vàng $\rho = \frac{1 + \sqrt{5}}{2}$. Kiểm định điều này qua kết quả của bạn.

Bài tập 2.29. Đa thức Legendre $P_n(x)$ được định nghĩa theo công thức truy hồi sau:

$$(n+1)P_{n+1}(x) - (2n+1)xP_n(x) + nP_{n-1}(x) = 0,$$

với $P_0(x) = 1$, $P_1(x) = x$ và $P_2(x) = \frac{3x^2 - 1}{2}$. Lập chương trình tính đa thức Legendre bậc n , kết quả lưu dưới dạng vector hệ số.

Bài tập 2.30. Đa thức Chebyshev $T_n(x)$ được định nghĩa theo công thức truy hồi sau:

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x),$$

với $T_0(x) = 1$, $T_1(x) = x$. Lập chương trình tính đa thức Chebyshev bậc n , kết quả lưu dưới dạng vector hệ số.

Bài tập 2.31. Viết chương trình tính tiền điện cho một gia đình, biết rằng nếu số Kwh sử dụng (n):

- $n < 100$ thì giá 1 Kwh là 500đ;
- $100 \leq n < 200$ thì giá 1 Kwh là 700đ;
- $200 \leq n < 500$ thì giá 1 Kwh là 1000đ;
- $n \geq 500$ thì giá 1 Kwh là 1200đ.

Đồng thời:

- Nếu $n < 0$ thì in ra: "không hợp lệ";
- Nếu $n \geq 500$ thì sau dòng in tổng số tiền ghi kèm thêm câu: "Hạn chế sử dụng điện!"

Bài tập 2.32. Viết chương trình giải và biện luận phương trình bậc hai $ax^2 + bx + c = 0$. Áp dụng để giải các phương trình $x^2 - 3x + 2 = 0$ và $x^2 + 2x - 1 = 0$. Kiểm tra lại kết quả bằng cách tính bằng tay hoặc sử dụng hàm roots của MATLAB.

Bài tập 2.33. Viết chương trình với input là một số nguyên dương n và output là 2 số S_1, S_2 , trong đó $S_1 = \sum_{k=1}^n k$, $S_2 = \sum_{k=1}^n \frac{1}{k^3}$. Áp dụng tính $\sum_{k=1}^{2012} \frac{1}{k^3}$, sử dụng format long để in kết quả.

Bài tập 2.34. Viết chương trình tính gần đúng căn bậc hai \sqrt{a} với sai số ε cho trước theo công thức lặp Newton.

- Chọn $x_1 = \frac{a}{2}$;
- Sử dụng công thức lặp: $x_{n+1} = \frac{1}{2}(x_n + \frac{a}{x_n})$;
- Điều kiện dừng: $\left| \frac{x_{n+1} - x_n}{x_n} \right| < \varepsilon$.

Bài tập 2.35. Viết chương trình tính gần đúng căn bậc ba $\sqrt[3]{a}$ với sai số ε cho trước theo công thức lặp Newton.

- Chọn $x_1 = \frac{a}{3}$;
- Sử dụng công thức lặp: $x_{n+1} = \frac{1}{3}(\frac{a}{x_n^2} + 2x_n)$;
- Điều kiện dừng: $\left| \frac{x_{n+1} - x_n}{x_n} \right| < \varepsilon$.

Bài tập 2.36. Viết chương trình MATLAB tính tổng $\sum_{i=1}^n \frac{1}{i^3}$ bằng cách sử dụng mā vô hướng và mā vector hóa. So sánh thời gian chạy của hai cách trên bằng cách sử dụng lệnh profile.

CHƯƠNG 3

GIẢI GẦN ĐÚNG PHƯƠNG TRÌNH

T RONG kỹ thuật, ta thường gặp bài toán: Tìm nghiệm thực của phương trình $f(x) = 0$ (đại số hoặc siêu việt). Nói chung, nghiệm của nó khó có thể tìm được trong dạng nghiệm đúng. Ngay cả trong trường hợp nghiệm đúng được tìm trong dạng số thập phân vô hạn (vô tỷ hay hữu tỷ tuần hoàn) thì trong máy tính cũng gặp sự quy tròn số. Khi đó ta cũng chỉ có được nghiệm gần đúng mà thôi.

Để tìm nghiệm gần đúng (thực) của phương trình $f(x) = 0$, ta thường phải tiến hành theo các bước sau:

1. Khẳng định phương trình có nghiệm trong một khoảng nào đó;
2. Chọn xấp xỉ đầu x_0 , xem đó là nghiệm gần đúng đầu tiên thuộc khoảng đang xét;
3. Điều chỉnh dần x_0 sao cho càng gần tới nghiệm đúng càng tốt, nghĩa là xây dựng thuật toán;
4. Đánh giá sai số của nghiệm gần đúng tìm được so với nghiệm đúng.

Để hoàn thành các công việc đề ra ta lần lượt xét các vấn đề sau.

§1. TÌM KHOẢNG PHÂN LY NGHIỆM

1.1 Khoảng phân ly nghiệm

Xét phương trình

$$f(x) = 0. \quad (3.1)$$

Định nghĩa 3.1. Giả sử α là một nghiệm của phương trình (3.1). Ta nói (a, b) là *khoảng phân ly nghiệm* α của phương trình (3.1) nếu trong khoảng đó phương trình (3.1) chỉ có một nghiệm thực α duy nhất.

Để tìm khoảng phân ly nghiệm, ta dựa vào các kết quả đã có trong giải tích:

Định lý 3.1. *Giả sử $f(x)$ là hàm số liên tục trên tập $D \subset \mathbb{R}$. Khi đó $(a, b) \subset D$ là một khoảng phân ly nghiệm của phương trình (3.1) nếu $f(a).f(b) < 0$ và $f(x)$ đơn điệu trên $[a, b]$.*

Định lý 3.2. *Với giả thiết của định lý 3.1 thì (a, b) là khoảng phân ly nghiệm α của phương trình (3.1) nếu $f(a).f(b) < 0$ và tồn tại $f'(x)$ đồng thời $f'(x)$ giữ nguyên một dấu $\forall x \in [a, b]$.*

Từ định lý ta suy ra, nếu (a, b) là khoảng phân ly nghiệm α của phương trình (3.1) thì mọi khoảng $(c, d) \subset (a, b)$, sao cho $f(c).f(d) < 0$ cũng là khoảng phân ly nghiệm của phương trình trên (thu hẹp khoảng chứa nghiệm). Khi đã có khoảng phân ly nghiệm thì tùy theo từng phương pháp có thể chọn điểm $x_0 \in [a, b]$ làm xấp xỉ đầu.

1.2 Phương pháp hình học tìm khoảng phân ly nghiệm

Giả thiết $f(x)$ là hàm số liên tục trên tập $D \subset \mathbb{R}$. Để tìm khoảng phân ly nghiệm, ta có thể tiến hành như sau:

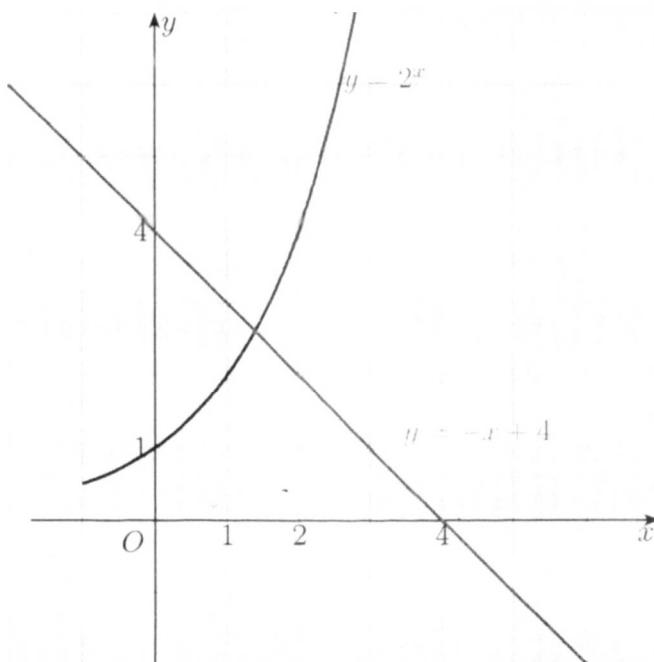
- Khảo sát, vẽ đồ thi hàm số $y = f(x)$. Giao điểm của đồ thi với trục hoành là điểm $x_0 \in D$. Nếu $f(x_0) = 0$ thì $x_0 = \alpha$ là nghiệm của phương trình. Nếu $f(x_0) \neq 0$ thì khoảng phân ly nghiệm α của phương trình (3.1) được chọn là lân cận về hai phía của x_0 , nghĩa là khoảng (a, b) sao cho $a < x_0 < b$, nhưng phải kiểm tra lại điều kiện của định lý 3.1 hoặc 3.2 (để khẳng định sự duy nhất của nghiệm).
- Trường hợp $y = f(x)$ khó vẽ, ta viết lại phương trình (3.1) dưới dạng

$$h(x) = g(x).$$

Khảo sát vẽ đồ thi các hàm $y = h(x)$ và $y = g(x)$ trên cùng một hệ trục tọa độ, hoành độ giao điểm của hai đồ thi là x_0 .

Nếu $h(x_0) = g(x_0)$ thì $x_0 = \alpha$ là nghiệm của phương trình. Nếu $h(x_0) \neq g(x_0)$ thì lân cận về hai phía của x_0 sẽ là khoảng phân ly nghiệm nếu thỏa mãn điều kiện của định lý 3.1 hoặc 3.2.

Ví dụ 3.1. Dùng phương pháp hình học tìm khoảng phân ly nghiệm của phương trình $2^x + x - 4 = 0$.



Hình 3.1

Giải: Viết lại phương trình đã cho dưới dạng

$$2^x = -x + 4.$$

Áp dụng phương pháp hình học ta thu được phương trình đã cho có một nghiệm trong khoảng $(1, 2)$ (xem hình 3.1). Để dàng kiểm tra khoảng phân ly nghiệm $(1, 2)$ thỏa mãn điều kiện của định lý 3.2.

§2. PHƯƠNG PHÁP LẮP ĐƠN

2.1 Nội dung phương pháp

Giả sử (a, b) là khoảng phân ly nghiệm α của phương trình (3.1), tức là $f(\alpha) = 0$, $\alpha \in (a, b)$.

Viết lại phương trình (3.1) trong dạng

$$x = g(x), \quad (3.2)$$

trong đó $g(x)$ là hàm số liên tục trên $[a, b]$.

Chọn x_0 là điểm bất kỳ thuộc $[a, b]$ và tính dãy lặp theo công thức

$$x_n = g(x_{n-1}), \quad n = 1, 2, \dots \quad (3.3)$$

Công thức (3.3) được gọi là quá trình lặp, n được gọi là thứ của phép lặp.

2.2 SỰ HỘI TỤ CỦA PHƯƠNG PHÁP

Định nghĩa 3.2. Quá trình lặp (3.3) được gọi là hội tụ, nếu dãy $\{x_n\}$ tính theo (3.3) hội tụ, tức là tồn tại $\lim_{n \rightarrow \infty} x_n = \bar{x}$.

Định lý 3.3. Nếu quá trình lặp (3.3) hội tụ, nghĩa là $\lim_{n \rightarrow \infty} x_n = \bar{x}$ thì $\bar{x} = \alpha$ là nghiệm đúng của phương trình (3.1).

Chứng minh. Do $g(x)$ là hàm liên tục nên ta có

$$\bar{x} = \lim_{n \rightarrow \infty} x_n = \lim_{n \rightarrow \infty} g(x_{n-1}) = g\left(\lim_{n \rightarrow \infty} x_{n-1}\right) = g(\bar{x}).$$

Vậy $\bar{x} = \alpha$ là nghiệm đúng của phương trình (3.1). ■

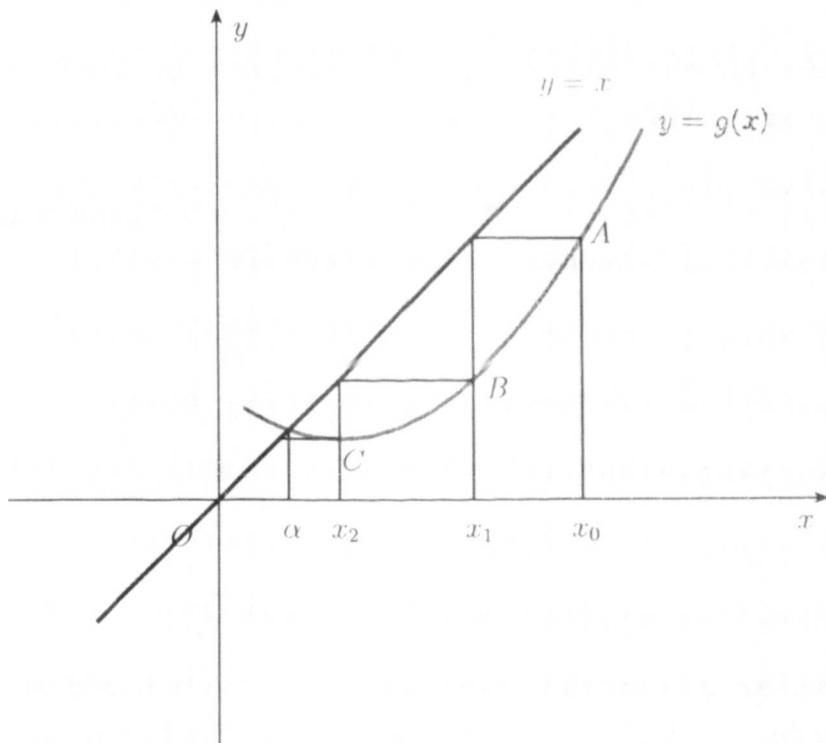
Vấn đề đặt ra là điều kiện nào để quá trình lặp (3.3) hội tụ. Ta công nhận định lý sau (chi tiết có thể xem trong tài liệu tham khảo).

Định lý 3.4. *Giả sử $g(x)$ là hàm liên tục và có đạo hàm $g'(x)$ liên tục $\forall x \in [a, b]$. Giả thiết tồn tại số $q > 0$ sao cho*

$$|g'(x)| \leq q < 1 \quad (3.4)$$

và $x_n = g(x_{n-1}) \in [a, b]$, $\forall n = 1, 2, \dots$ với $x_0 \in [a, b]$ bất kỳ là xấp xỉ ban đầu. Khi đó, quá trình lặp (3.3) hội tụ.

Minh họa hình học của phương pháp lặp đơn được mô tả như trong hình dưới đây.



Hình 3.2

Với điều kiện của định lý 3.4 và theo định lý 3.3 thì quá trình lặp (3.3) hội tụ tới nghiệm đúng α của phương trình (3.1). Khi tính toán thì quá trình lặp không thể kéo dài ra vô hạn mà phải dừng ở bước thứ n ; ta xem $x_n \approx \alpha$, tức là có sai số.

2.3 Sai số

Giả sử α là nghiệm đúng của phương trình (3.1) và x_n là nghiệm gần đúng tính theo công thức (3.3). Khi đó sai số tuyệt đối của nghiệm $|x_n - \alpha|$ được đánh giá theo các công thức

$$|x_n - \alpha| \leq \frac{q^n}{1-q} |x_1 - x_0| \quad (3.5)$$

hoặc

$$|x_n - \alpha| \leq \frac{q}{1-q} |x_n - x_{n-1}|. \quad (3.6)$$

(Phản chứng minh chi tiết các bất đẳng thức trên có thể xem trong tài liệu tham khảo.)

Chú ý 3.1. Công thức (3.5) thuận lợi trong việc tìm nghiệm gần đúng x_n đạt sai số $\varepsilon > 0$ cho trước. Cụ thể, từ đánh giá

$$|x_n - \alpha| \leq \frac{q^n}{1-q} |x_1 - x_0| < \varepsilon$$

ta suy ra cần chọn số n nhỏ nhất thỏa mãn $n > \log_q \frac{\varepsilon(1-q)}{|x_1 - x_0|}$ hay

$$n = [\log_q \frac{\varepsilon(1-q)}{|x_1 - x_0|}] + 1.$$

Trong khi đó, công thức (3.6) thuận lợi khi tính trên máy. Muốn nghiệm gần đúng x_n đạt sai số ε , ta chỉ cần kiểm tra điều kiện dừng:

$$|x_n - x_{n-1}| < \delta = \frac{\varepsilon(1-q)}{q}.$$

Tóm tắt thuật toán:

Giả sử (a, b) là khoảng phân ly nghiệm α của phương trình $f(x) = 0$.

- *Bước 1:* Viết $f(x) = 0$ trong dạng $x = g(x)$;
- *Bước 2:* Kiểm tra điều kiện $|g'(x)| \leq q < 1$;
- *Bước 3:* Chọn xấp xỉ đầu $x_0 \in [a, b]$ và tính

$$x_n = g(x_{n-1}), \quad n = 1, 2, \dots$$

Sau k lần lặp, được $x_k \approx \alpha$ là nghiệm gần đúng của phương trình.

Sai số được đánh giá theo các công thức (3.5) hoặc (3.6).

2.4 Chương trình MATLAB

```

function [x, k]=lapdon(g, q, x0, epsilon, maxit)
% Hàm lapdon minh họa thuật toán
% lặp đơn giải phương trình x=g(x)
% Input: _ g là hàm về phải
%         _ q là cận trên của đạo hàm g' (x) (0<q<1)
%         _ x0 là xấp xỉ ban đầu
%         _ epsilon là sai số tuyệt đối
%         _ maxit là số phép lặp tối đa
% Output: _ k là số lần lặp cần thiết
%           _ x là nghiệm gần đúng thứ k
if nargin<5
    maxit=1e+3;
end
if nargin<4
    epsilon=1e-5;
end
delta=epsilon*(1-q)/q;
k=1; x=feval(g,x0);
while (abs(x-x0)>=delta) && (k<maxit)
    x0=x;
    x=feval(g,x0);
    k=k+1;
end

```

```

x=feval(g,x0);
k=k+1;
end
end

```

Chú ý 3.2. Trong chương trình trên thì hàm g được sử dụng như là đầu vào của hàm `lapdon`. Thông thường, có hai cách khai báo hàm g để có thể chạy thử chương trình `lapdon` như sau:

1. Tạo file "g.m", khi gọi hàm `lapdon` thì hàm này nhất thiết phải nhập dưới dạng xâu 'g'.
2. Trong cửa sổ lệnh nhập vào

```
>> g=@(x) biểu thức của g,
```

và khi gọi chương trình `lapdon` thì chỉ cần nhập g dưới dạng tên nguyên bản, không đặt trong dấu nháy đơn như trường hợp trên.

Ví dụ 3.2. Giả sử ta muốn chạy chương trình `lapdon` để tìm nghiệm gần đúng của phương trình $r = g(x)$, trong đó $g(x) = 1 + x - \frac{x^2}{4}$ trên đoạn $[1, 3]$. Để thấy $(1, 3)$ là khoảng phân ly nghiệm $\alpha = 2$ của phương trình trên, đồng thời ta có đánh giá

$$|g'(x)| = \left|1 - \frac{x}{2}\right| < \frac{1}{2} =: q < 1, \forall x \in (1, 3).$$

Ta sẽ lần lượt thực hiện theo hai cách đã nêu trong chú ý 3.2:

1. Tao hàm g.m với nội dung:

```

function temp=g(x)
temp=1+x-x.^2/4;
end

```

sau đó gọi thủ tục `lapdon` dưới dạng

```
>> [x,k]=lapdon('g',q,x0,1e-5,100)
```

2. Nhập các lệnh sau:

```
>> q=@(x) 1+x-x^2/4;
>> [x,k]=lapdon(q,q,x0,1e-5,100)
```

Bạn đọc hãy tự thực hiện theo cả hai cách trên và kiểm tra kết quả.

§3. PHƯƠNG PHÁP NEWTON (PHƯƠNG PHÁP TIẾP TUYẾN)

3.1 Nội dung phương pháp

Giả thiết (a, b) là khoảng phân ly nghiệm α của phương trình (3.1), đồng thời $f(x)$ là hàm liên tục và có đạo hàm liên tục trên $[a, b]$. Như vậy AB là cung trơn với $A(a, f(a))$ và $B(b, f(b))$.

Thay cung AB bởi đường tiếp tuyến tại A hoặc tại B (hình 3.3). Hoành độ giao điểm của tiếp tuyến của đồ thị hàm $f(x)$ tại A (hoặc B) với trục hoành Ox được gọi là nghiệm gần đúng thứ nhất x_1 của phương trình (3.1) nếu $x_1 \in (a, b)$. Khi đó ta có điểm $M_1(x_1, f(x_1))$. Lặp lại quá trình trên đối với $M_1 \dots$

Bây giờ ta xây dựng thuật toán. Lập phương trình tiếp tuyến tại A hoặc B , chẳng hạn tại $B(b, f(b))$.

Đặt $x_0 = b$ làm xấp xỉ thì $B(x_0, f(x_0))$ và phương trình tiếp tuyến tại B :

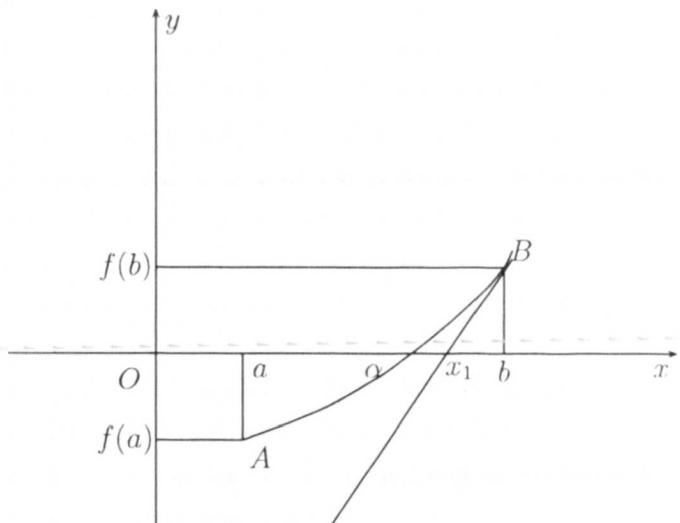
$$y = f(x_0) + f'(x_0)(x - x_0).$$

Cho $y = 0$ (tìm hoành độ giao điểm của tiếp tuyến trên với trục hoành) ta được

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}.$$

Nếu $x_1 \in (a, b)$ ta có điểm $M_1(x_1, f(x_1))$. Lặp lại quá trình trên với điểm M_1 ta được

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}; \dots$$



Hình 3.3

Tổng quát ta có

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}, \quad n = 1, 2, \dots \quad (3.7)$$

Công thức (3.7) được gọi là quá trình lặp Newton (hay tiếp tuyến).

3.2 Sự hội tụ của phương pháp Newton

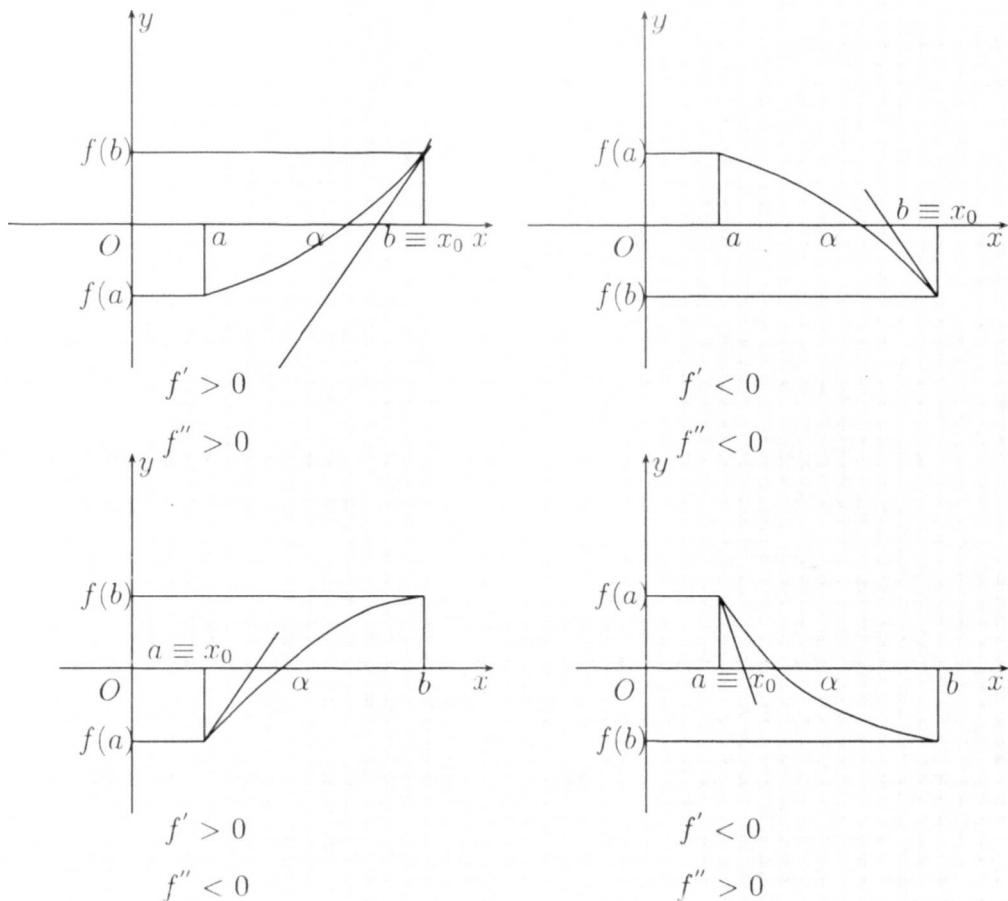
Định lý 3.5. Giả sử (a, b) là khoảng phân ly nghiệm α của phương trình (3.1), $f(x)$ là hàm số liên tục và có đạo hàm liên tục đến cấp hai; đồng thời $f'(x), f''(x)$ giữ nguyên dấu $\forall x \in [a, b]$. Xấp xỉ ban đầu x_0 được chọn là a (hoặc b) sao cho

$$f(x_0) \cdot f''(x) > 0 \quad \forall x \in [a, b]. \quad (3.8)$$

Khi đó dãy $\{x_n\}$ tính theo công thức (3.7) sẽ hội tụ đơn điệu tới nghiệm đúng α của phương trình (3.1).

Cụ thể hơn, nếu $f'(x), f''(x)$ cùng dấu thì dãy $\{x_n\}$ đơn điệu giảm tới nghiệm đúng α , còn nếu $f'(x), f''(x)$ khác dấu thì dãy $\{x_n\}$ đơn điệu tăng tới nghiệm đúng α của phương trình (3.1).

Định lý này có thể chứng minh bằng giải tích, ở đây ta chỉ mô tả hình học (hình 3.4).



Hình 3.4

3.3 Sai số

Xem $x_n \approx \alpha$ thì $|x_n - \alpha|$ là sai số tuyệt đối của nghiệm gần đúng x_n của phương trình (3.1).

Với các điều kiện của định lý 3.5 thì ta có đánh giá

$$|x_n - \alpha| \leq \frac{|f(x_n)|}{m_1}, \quad (3.9)$$

trong đó $0 < m_1 \leq |f'(x)| \quad \forall x \in [a, b]$, hoặc

$$|x_n - \alpha| \leq \frac{M_2}{2m_1} |x_n - x_{n-1}|^2, \quad (3.10)$$

với $M_2 \geq |f''(x)| \quad \forall x \in [a, b]$.

Chứng minh chi tiết các đánh giá (3.9) và (3.10) có thể xem trong tài liệu tham khảo.

Từ (3.10), ta thấy, để $|x_n - \alpha| < \varepsilon$ thì cần

$$|x_n - x_{n-1}| < \sqrt{\frac{2m_1\varepsilon}{M_2}} = \delta.$$

- Chú ý 3.3.**
1. Bất đẳng thức (3.9) đúng với mọi phương pháp lặp, chỉ cần xem x_n là nghiệm gần đúng của phương trình (3.1).
 2. Bất đẳng thức (3.10) cho thấy phương pháp Newton hội tụ tới nghiệm đúng cấp hai, nhanh hơn so với phương pháp lặp đơn.

Tóm tắt thuật toán:

Giả sử (a, b) là khoảng phân ly nghiệm α của phương trình $f(x) = 0$.

- *Bước 1:* Tính $f'(x)$; $f''(x)$ và xét dấu của chúng.
- *Bước 2:* Chọn $x_0 = a$ hoặc $x_0 = b$ sao cho thỏa mãn điều kiện

$$f(x_0) \cdot f''(x) > 0 \quad \forall x \in [a, b].$$

- *Bước 3:* Từ xấp xỉ đầu x_0 , tính

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}, \quad n = 1, 2, \dots$$

Sau k lần lặp ta thu được $x_k \approx \alpha$ là nghiệm gần đúng của phương trình. Sai số $|x_k - \alpha|$ được đánh giá theo công thức (3.10).

3.4 Chương trình MATLAB

Chương trình MATLAB minh họa việc dùng thuật toán Newton (tiếp tuyến) tìm nghiệm xấp xỉ của phương trình $f(x) = 0$ trong khoảng phân ly nghiệm (a, b) .

```
function [x,n]=newton(fun,dfun,m1,M2,x0,err,maxl)
% Hàm newton tìm nghiệm gần đúng trên đoạn [a,b]
```

% với sai số tuyệt đối err bằng phương pháp Newton.

% Input:

```
%           _ fun là hàm về trái
%           _ dfun là đạo hàm của hàm fun
%           _ x0 là xấp xỉ ban đầu
%           _ m1 là cận dưới của đạo hàm cấp một
%           _ M2 là cận trên của đạo hàm cấp hai
%           _ err là sai số tuyệt đối
%           _ max1 là số phép lặp tối đa
```

% Output:

```
%           _ x là nghiệm xấp xỉ
%           _ k là số phép lặp cần thiết
```

if nargin<7

max1=1e+3;

end

if nargin<6

err=1e-5;

end

delta=sqrt(2*m1*err/M2);

n=1; x=x0-feval(fun,x0)/feval(dfun,x0);

while abs(x-x0)>=delta && n<max1

x0=x;

x=x0-feval(fun,x0)/feval(dfun,x0);

n=n+1;

end

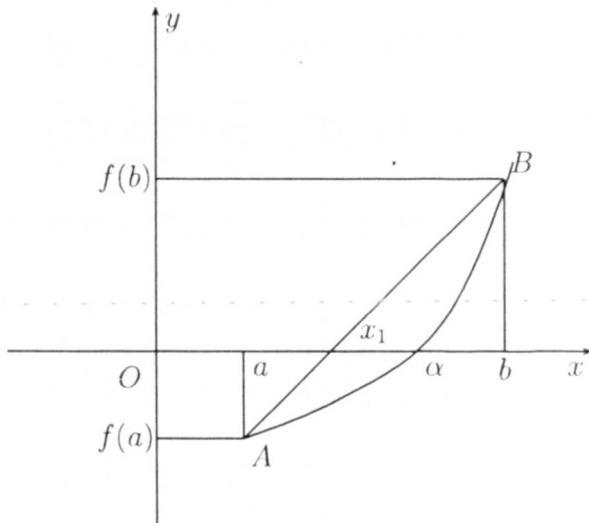
end

Bạn đọc hãy tự áp dụng để tính đến kết quả đối với bài toán: Tìm nghiệm gần đúng của phương trình $f(x) = x^4 - 3x^2 + 75x - 10000 = 0$ trong khoảng $(-11; -10)$. Chọn điểm xuất phát $x_0 = -11$, tìm nghiệm x_n sao cho sai số tuyệt đối không vượt quá 10^{-5} .

§4. PHƯƠNG PHÁP DÂY CUNG

4.1 Nội dung phương pháp

Giả sử (a, b) là khoảng phân ly nghiệm α của phương trình (3.1); đồng thời $f(x)$ là hàm số liên tục và có đạo hàm liên tục trên đoạn $[a, b]$. Trên mặt phẳng tọa độ ta có hai điểm $A(a, f(a))$ và $B(b, f(b))$ (hình 3.5).



Hình 3.5

Dây cung qua A, B cắt trục hoành tại điểm có hoành độ là x_1 . Xem x_1 là nghiệm gần đúng của phương trình. Phương trình đường thẳng qua A và B :

$$y = \frac{f(b) - f(a)}{b - a}(x - a) + f(a).$$

Cho $y = 0$ ta suy ra

$$x_1 = a - \frac{b - a}{f(b) - f(a)} f(a). \quad (3.11)$$

Chọn $x_0 = a$ (hoặc $x_0 = b$) thì $d = b$ (hoặc $d = a$). Thay vào (3.11) ta có

$$x_1 = x_0 - \frac{d - x_0}{f(d) - f(x_0)} \cdot f(x_0) \quad (3.12)$$

Từ (3.12) ta có điểm $M_1(x_1, f(x_1))$ thuộc cung AB . Lặp lại quá trình trên đối với đoạn $[x_1, d]; \dots$ ta được quá trình lặp theo phương pháp dây cung

như sau:

$$x_n = x_{n-1} - \frac{d - x_{n-1}}{f(d) - f(x_{n-1})} \cdot f(x_{n-1}), \quad n = 1, 2, \dots \quad (3.13)$$

4.2 SỰ HỘI TỤ CỦA PHƯƠNG PHÁP DÂY CUNG

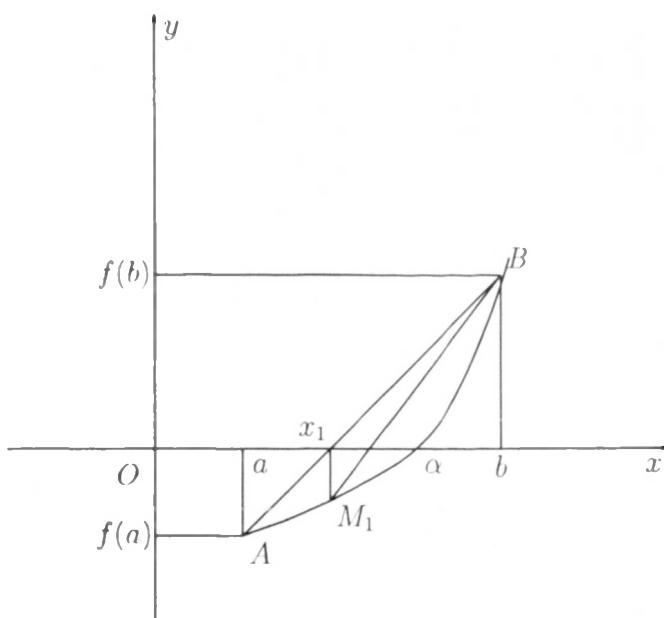
Định lý 3.6. *Giả sử (a, b) là khoảng phân ly nghiệm α của phương trình (3.1). Hàm số $f(x)$ liên tục và có đạo hàm $f'(x)$, $f''(x)$; đồng thời các đạo hàm đó giữ nguyên dấu trên đoạn $[a, b]$. Xấp xỉ ban đầu x_0 được chọn sao cho*

$$\begin{cases} f(x_0) \cdot f''(x) < 0; \\ f(d) \cdot f(x_0) < 0. \end{cases} \quad (3.14)$$

Khi đó dãy $\{x_n\}$ tính theo công thức (3.13) sẽ đơn điệu hội tụ tới α khi $n \rightarrow \infty$.

Cụ thể: nếu $f'(x)$, $f''(x)$ cùng dấu thì dãy $\{x_n\}$ đơn điệu tăng tới α khi $n \rightarrow \infty$ và ngược lại, nếu $f'(x)$, $f''(x)$ khác dấu thì dãy $\{x_n\}$ đơn điệu giảm tới α khi $n \rightarrow \infty$.

Định lý này có thể chứng minh bằng giải tích (độc giả tự chứng minh hoặc xem tài liệu tham khảo). Ở đây ta chỉ mô tả hình vẽ cho một trường hợp $f' > 0$, $f''(x) > 0$ (xem hình vẽ 3.6).



Hình 3.6

4.3 Sai số

Quá trình lặp (3.13) không thể kéo dài ra vô hạn, dừng ở bước thứ lặp thứ n ta được $x_n \approx \alpha$. Sai số được đánh giá theo công thức (có thể xem chứng minh trong tài liệu tham khảo):

$$|x_n - \alpha| \leq \frac{M_1 - m_1}{m_1} |x_n - x_{n-1}|, \quad (3.15)$$

trong đó,

$$0 < m_1 \leq |f'(x)| \leq M_1, \quad \forall x \in [a, b].$$

Từ (3.15) ta thấy, để $|x_n - \alpha| < \varepsilon$ thì cần

$$|x_n - x_{n-1}| < \frac{m_1 \varepsilon}{M_1 - m_1}.$$

Tóm tắt thuật toán:

Giả sử (a, b) là khoảng phân ly nghiệm α của phương trình $f(x) = 0$.

- **Bước 1:** Tính $f'(x)$, $f''(x)$ và xét dấu của chúng.
- **Bước 2:** Kiểm tra điều kiện

$$\begin{cases} f(x_0).f''(x) < 0 \\ f(d).f(x_0) < 0 \end{cases}$$

để chọn xấp xỉ ban đầu x_0 là a (hay b) thì $d = b$ (hay a).

- **Bước 3:** Từ xấp xỉ đầu x_0 , tính

$$x_n = x_{n-1} - \frac{d - x_{n-1}}{f(d) - f(x_{n-1})}.f(x_{n-1}), \quad n = 1, 2, \dots$$

Sau k bước lặp ta được $x_k \approx \alpha$ là nghiệm gần đúng cần tìm. Sai số $|x_k - \alpha|$ được đánh giá theo công thức (3.15).

4.4 Chương trình MATLAB

```
function [x,n]=secant(fun,m1,M1,x0,x1,err,max1)
% Hàm secant tìm nghiệm gần đúng trên đoạn [a,b]
% với sai số tuyệt đối err bằng phương pháp dây cung.
```

```

% Input:
%   _ fun is the object function
%   _ x0, x1 are two starting points
%       (initial approximation to a zero)
%   _ m1, M1 are lower bound and upper bound
%       of the derivative f' (x)
%   _ err is the tolerance
%   _ max1 is the maximum of number of iterations

% Output:
%   _ x is the zero
%   _ n is the number of iterations

if nargin<7
    max1=1e+3;
end

if nargin<6
    err=1e-5;
end

n=0;
delta=m1*err/(M1-m1);
while abs(x1-x0)>=delta && n<max1
    x2=x0;
    x0=x1;
    x1=x1+(x1-x2)/(feval(fun,x2)/feval(fun,x1)-1);
    n=n+1;
end
x=x1;
end

```

Chú ý 3.4. Qua hai phương pháp Newton và dây cung đã xét trong các tiết §3 và §4, ta thấy, trong cùng điều kiện của hàm $f(x)$ là hàm số liên tục và có đạo hàm liên tục đến cấp hai trên khoảng phân ly nghiệm α của phương trình (3.1) thì:

- Nếu f' , f'' cùng dấu thì tính theo phương pháp Newton sẽ thu được dây đơn điệu giảm tới nghiệm đúng α còn theo phương pháp dây cung sẽ được dây đơn điệu tăng tới α .
- Nếu f' , f'' trái dấu sẽ cho ta kết quả ngược lại.

Vì vậy, ta có thể tính đồng thời theo cả hai phương pháp trong cùng điều kiện (gọi là sự phối hợp của các phương pháp lặp). Nghiệm gần đúng sau k phép lặp ta được \bar{x}_k là nghiệm gần đúng theo phương pháp Newton, còn \tilde{x}_k là nghiệm gần đúng theo phương pháp dây cung. Khi đó, nghiệm gần đúng x_k có thể lấy $x_k = \frac{\bar{x}_k + \tilde{x}_k}{2}$ sẽ đạt kết quả tốt hơn.

§5. PHƯƠNG TRÌNH ĐA THỨC BẬC n

Xét phương trình đa thức bậc n :

$$p(x) = 0. \quad (3.16)$$

trong đó, $p(x) = a_0x^n + a_1x^{n-1} + \dots + a_{n-1}x + a_n$ với $a_0 \neq 0$, $a_i \in \mathbb{R}$, $i = \overline{0, n}$.

5.1 Miền chứa nghiệm của đa thức

Ta đã biết phương trình đa thức dạng (3.16) có đủ n nghiệm cả thực lẫn phức (đối với nghiệm phức luôn là cặp nghiệm phức liên hợp). Tập hợp các nghiệm đó chỉ có trong miền được cho trong định lý sau:

Định lý 3.7. *Đặt $A = \max \{|a_1|, \dots, |a_n|\} = \max_{i=1,n} |a_i|$ thì các nghiệm (thực hoặc phức) của phương trình (3.16) thỏa mãn*

$$|x| < 1 + \frac{A}{|a_0|}. \quad (3.17)$$

Nghĩa là các nghiệm của phương trình nằm trong hình tròn tâm $O(0, 0)$, bán kính $R = 1 + \frac{A}{|a_0|}$ trong mặt phẳng phức.

Từ định lý trên ta thấy nghiệm thực x của phương trình (3.16) luôn thỏa mãn $-R < x < R$.

5.2 Sơ đồ Horner tính giá trị của đa thức

Ta có

$$\begin{aligned}
 p(x) &= a_0x^n + a_1x^{n-1} + \cdots + a_{n-1}x + a_n \\
 &= (a_0x + a_1)x^{n-1} + a_2x^{n-1} + \cdots \\
 &= ((a_0x + a_1)x + a_2)x^{n-1} + a_3x^{n-3} + \cdots \\
 &= (\cdots + ((a_0x + a_1)x + a_2)x + \cdots + a_{n-1})x + a_n.
 \end{aligned} \tag{3.18}$$

Do đó, khi thay $x = c$ ta tính dần

$$\begin{aligned}
 b_0 &= a_0; \\
 b_1 &= b_0c + a_1; \\
 b_2 &= b_1c + a_2; \\
 &\quad \cdots \quad \cdots \\
 b_n &= b_{n-1}c + a_n.
 \end{aligned} \tag{3.19}$$

Từ (3.18) ta suy ra $p(c) = b_n$ là giá trị của đa thức tại $x = c$. Quá trình tính $p(c)$ theo (3.19) gọi là sơ đồ Horner, được mô tả trên bảng tính như sau:

Hệ số đa thức	a_0	a_1	a_2	\dots	a_{n-1}	a_n
	b_0c	b_1c	\dots	$b_{n-2}c$	$b_{n-1}c$	
c	b_0	b_1	b_2	\dots	b_{n-1}	$b_n = p(c)$

Dễ dàng chỉ ra rằng các hệ số $a_0 = b_0, b_1, \dots, b_{n-1}$ cũng là hệ số của đa thức thương $q(x) = b_0x^{n-1} + b_1x^{n-2} + \cdots + b_{n-2}x + b_{n-1}$ khi chia $p(x)$ cho nhị thức $x - c$. Vậy khi tìm nghiệm gần đúng thực của phương trình đa thức (3.16), ta tìm khoảng chứa nghiệm thực $-R < x < R$. Nhờ sơ đồ Horner để phân ly nghiệm và áp dụng các phương pháp đã xét trong §2, 3, 4.

5.3 Chương trình MATLAB

Chương trình MATLAB minh họa việc tính giá trị của một đa thức tại một điểm bằng cách áp dụng thuật toán Horner.

```

function v=horner1(a,c)
% Input: a là vector hệ số của đa thức;
%         c là giá trị của biến
% Ouput: v là giá trị của đa thức tại điểm x=c.
m=length(a);
v=a(1);
for i=2:m
    v=v*c+a(i);
end
end

```

Thử nghiệm chương trình trên, tính giá trị của đa thức

$p(x) = -x^7 + 4x^5 - 2x^4 + 5x^2 + 7$ có vector biểu diễn hệ số tương ứng $a = [-1 \ 0 \ 4 \ -2 \ 0 \ 5 \ 0 \ 7]$, ta thu được giá trị $p(2) = -5$. Chú ý rằng ta hoàn toàn có thể thực hiện công việc trên bằng cách sử dụng lệnh MATLAB như sau:

```

>> a=[-1 0 4 -2 0 5 0 7];
>> v=polyval(a,2)
v =
-5

```

§6. GIẢI GẦN ĐÚNG HỆ PHƯƠNG TRÌNH PHI TUYẾN

Trong tiết này ta mở rộng phương pháp Newton (tiếp tuyến) và phương pháp lặp để tìm nghiệm gần đúng của hệ phương trình phi tuyến. Để thuận tiện cho việc trình bày, ta xét hệ hai phương trình hai ẩn

$$\begin{cases} f(x, y) = 0 \\ g(x, y) = 0, \end{cases} \quad (3.20)$$

trong đó f, g là các hàm (phi tuyến) phụ thuộc vào hai biến x, y . Thuật toán và kết quả có thể dễ dàng mở rộng cho hệ n phương trình n ẩn.

6.1 Phương pháp Newton

Bằng cách nào đó (chẳng hạn phương pháp đồ thị), ta chọn được điểm xấp xỉ ban đầu $M_0(x_0, y_0)$. Xuất phát từ điểm đó ta tìm $M_1(x_1, y_1)$ gần với nghiệm hơn. Đặt

$$x_1 = x_0 + h_0, \quad y_1 = y_0 + k_0$$

và giả thiết $f(x, y), g(x, y)$ là những hàm số khả vi đến cấp cần thiết nào đó. Áp dụng khai triển Taylor các hàm $f(x, y)$ và $g(x, y)$ tại điểm M_0 ta có

$$\begin{aligned} f(x_0 + h_0, y_0 + k_0) &= f(x_0, y_0) + h_0 \frac{\partial f}{\partial x}|_{M_0} + k_0 \frac{\partial f}{\partial y}|_{M_0} + \dots \\ g(x_0 + h_0, y_0 + k_0) &= g(x_0, y_0) + h_0 \frac{\partial g}{\partial x}|_{M_0} + k_0 \frac{\partial g}{\partial y}|_{M_0} + \dots \end{aligned} \quad (3.21)$$

Với h_0, k_0 đủ bé, bỏ qua các số hạng từ h_0^2, k_0^2 trở đi trong (3.21) ta được xấp xỉ; thay dấu " \approx " bởi dấu " $=$ " ta thu được hệ phương trình để xác định h_0, k_0 :

$$\begin{aligned} f'_x(M_0)h_0 + f'_y(M_0)k_0 &= -f(M_0); \\ g'_x(M_0)h_0 + g'_y(M_0)k_0 &= -g(M_0). \end{aligned} \quad (3.22)$$

Trong hệ (3.22), nếu định thức Jacobi

$$J_0 = J(M_0) = \begin{vmatrix} f'_x(M_0) & f'_y(M_0) \\ g'_x(M_0) & g'_y(M_0) \end{vmatrix} \neq 0$$

thì hệ trên tồn tại duy nhất nghiệm tính bởi công thức

$$\begin{aligned} h_0 &= -\frac{1}{J(M_0)} \begin{vmatrix} f(M_0) & f'_y(M_0) \\ g(M_0) & g'_y(M_0) \end{vmatrix}; \\ k_0 &= -\frac{1}{J(M_0)} \begin{vmatrix} f'_x(M_0) & f(M_0) \\ g'_x(M_0) & g(M_0) \end{vmatrix}. \end{aligned} \quad (3.23)$$

đồng thời ta được điểm gần đúng thứ nhất tốt hơn gần đúng $M_0(x_0, y_0)$ ban đầu

$$x_1 = x_0 + h_0, \quad y_1 = y_0 + k_0.$$

Nếu điểm $M_1(x_1, y_1)$ chưa đạt mong muốn, ta thực hiện lặp lại quá trình trên đối với điểm M_1 . Tổng quát ta có công thức lặp

$$\begin{aligned} x_{n+1} &= x_n + h_n; \\ y_{n+1} &= y_n + k_n, \end{aligned} \quad (3.24)$$

trong đó, h_n, k_n được xác định theo công thức

$$\begin{aligned} h_n &= -\frac{1}{J(M_n)} \begin{vmatrix} f(M_n) & f'_y(M_n) \\ g(M_n) & g'_y(M_n) \end{vmatrix}; \\ k_n &= -\frac{1}{J(M_n)} \begin{vmatrix} f'_x(M_n) & f(M_n) \\ g'_x(M_n) & g(M_n) \end{vmatrix}, \quad n = 0, 1, \dots \end{aligned} \quad (3.25)$$

Người ta đã chứng minh được rằng, nếu xấp xỉ đầu $M_0(x_0, y_0)$ gần đúng với nghiệm của hệ (3.20) thì quá trình lặp theo công thức (3.24), (3.25) sẽ hội tụ tới nghiệm đúng (x^*, y^*) của hệ (3.20) khi $n \rightarrow \infty$ (có thể xem trong tài liệu tham khảo).

6.2 Phương pháp lặp

Viết lại hệ (3.20) dưới dạng

$$\begin{cases} x &= F(x, y) \\ y &= G(x, y). \end{cases} \quad (3.26)$$

Chọn $M_0(x_0, y_0)$ làm xấp xỉ ban đầu và tính

$$\begin{cases} x_1 &= F(x_0, y_0) \\ y_1 &= G(x_0, y_0). \end{cases}$$

Tổng quát

$$\begin{cases} x_n &= F(x_{n-1}, y_{n-1}) \\ y_n &= G(x_{n-1}, y_{n-1}), \end{cases} \quad n = 1, 2, \dots \quad (3.27)$$

Điều kiện để quá trình lặp (3.27) hội tụ, nghĩa là dãy $M_n(x_n, y_n)$ hội tụ tới điểm $M^*(x^*, y^*)$ là nghiệm đúng của hệ (3.26) khi $n \rightarrow \infty$ được cho bởi định lý sau

Định lý 3.8. *Giả sử $M^*(x^*, y^*)$ là nghiệm đúng của hệ (3.26) nằm trong hình chữ nhật*

$$\mathcal{D} = \{a \leq x \leq b, c \leq y \leq d\}.$$

Giả thiết $M_0(x_0, y_0) \in \mathcal{D}$ bất kỳ là xấp xỉ ban đầu, ngoài ra các hàm số $F(x, y), G(x, y)$ liên tục, có các đạo hàm riêng liên tục và thỏa mãn các điều kiện

$$\begin{aligned} \left| \frac{\partial F}{\partial x} \right| + \left| \frac{\partial G}{\partial x} \right| &\leq q, \\ \left| \frac{\partial F}{\partial y} \right| + \left| \frac{\partial G}{\partial y} \right| &\leq q, \quad 0 < q < 1. \end{aligned} \quad (3.28)$$

Khi đó, quá trình lặp (3.27) hội tụ tới nghiệm đúng $M^*(x^*, y^*)$ của hệ (3.26) khi $n \rightarrow \infty$.

Nếu quá trình lặp trên dừng ở bước thứ n , ta có $x_n \approx x^*$, $y_n \approx y^*$ và sai số được đánh giá theo biểu thức

$$|x_n - x^*| + |y_n - y^*| \leq q^n \{ |b - a| + |d - c| \}. \quad (3.29)$$

Chứng minh định lý 3.8 và đánh giá chi tiết có thể xem trong tài liệu tham khảo.

6.3 Chương trình MATLAB

Xét ví dụ tìm nghiệm gần đúng của hệ bằng phương pháp Newton:

$$\begin{cases} x + 3 \lg x - y^2 = 0 \\ 2x^2 - xy - 5x + 1 = 0. \end{cases}$$

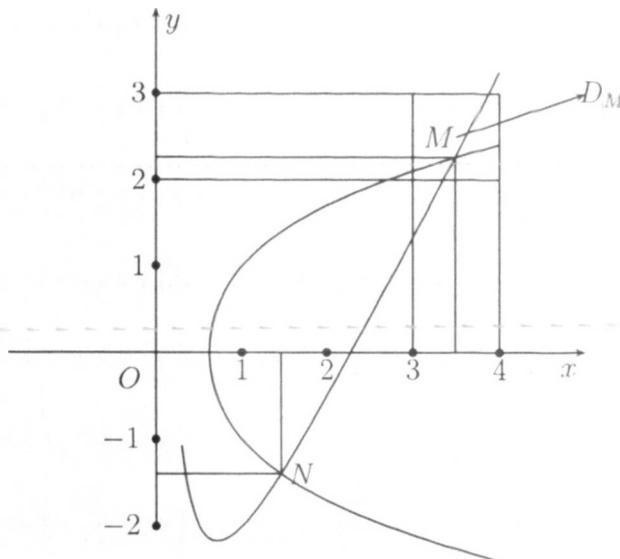
Xấp xỉ đầu có thể được chọn sơ bộ theo phương pháp hình học như sau:
Khảo sát và vẽ đồ thị các hàm

$$\begin{aligned} y^2 &= x + 3 \lg x, \quad (x > 0); \\ y &= \frac{2x^2 - 5x + 1}{x}. \end{aligned}$$

trên cùng một hệ trục tọa độ (hình 3.7), ta thu được hai giao điểm N và M với các lân cận tương ứng

$$D_N = \{1 \leq x \leq 2; -2 \leq y \leq -1\};$$

$$D_M = \{3 \leq x \leq 4; 2 \leq y \leq 3\}.$$



Hình 3.7

Ở đây, ta tìm nghiệm trong lân cận của M với $x_0 = 3.4$; $y_0 = 2.2$.

Thủ tục tính hàm F :

```
function F=F(X)
F=zeros(2,1);
F(1)=X(1)+3*log10(X(1))-X(2)*X(2);
F(2)=2*X(1)*X(1)-X(1)*X(2)-5*X(1)+1;
end
```

và thủ tục tính ma trận Jacobi JF :

```
function JF=JF(X)
x=X(1); y=X(2);
JF=[1+3/(x*log(10)), -2*y; 4*x-y-5, -x];
end
```

§7. BÀI TẬP

Các bài tập sau đây đều sử dụng phương pháp tính theo công thức và lập trình tính theo chương trình MATLAB để kiểm tra lại.

Bài tập 3.1. Tìm khoảng phân ly nghiệm của các phương trình sau:

1. $f(x) = x^3 - 6x + 2 = 0;$
2. $f(x) = 2^x - 5x - 3 = 0;$
3. $f(x) = \lg(10 + x) - x - 6 = 0;$
4. $f(x) = \lg x + x^2 - 6 = 0.$

Bài tập 3.2. Bằng phương pháp lặp đơn, phương pháp Newton, và phương pháp dây cung tìm nghiệm gần đúng của các phương trình sau sao cho đạt sai số tuyệt đối < 0.01 .

- | | |
|----------------------------------|----------------------------|
| 1. $x^3 - 12x - 5 = 0;$ | 5. $\sin x + \cos x = 4x;$ |
| 2. $x^3 - 2x^2 - 4x + 7 = 0;$ | 6. $x = \sin 3x;$ |
| 3. $x^2 - 10 \lg x = 2;$ | 7. $x^4 + 3x = 20;$ |
| 4. $(x - 1)^2 = \frac{1}{2}e^x;$ | 8. $x^4 - 3x + 1 = 0.$ |

Bài tập 3.3. Cho phương trình

$$\sqrt{1.65}x^3 + 1.12x^2 + 14.35x - 12.75 = 0.$$

Chứng tỏ rằng khoảng $(0.5; 1.5)$ là một khoảng phân ly nghiệm của phương trình trên. Kiểm tra sự hội tụ của các phương pháp lặp; phương pháp Newton (tiếp tuyến). Tính đến phép lặp thứ ba x_3 bằng các phương pháp trên. Đánh giá sai số của x_3 theo từng phương pháp.

Bài tập 3.4. Cho đa thức

$$P_n(x) = a_0x^n + a_1x^{n-1} + \cdots + a_{n-1}x + a_n, \quad (a_0 \neq 0).$$

Tìm công thức tính đạo hàm $P^{(k)}(x)$, $k = 0, 1, 2, \dots$ theo sơ đồ Horner. Áp dụng tính

$$P_7(x) = x^7 - 6x^5 + 7x^4 - 8x^3 + 6x + 7$$

tại điểm $x = -3$.

Bài tập 3.5. Tìm nghiệm dương của hệ phương trình sau:

$$\begin{cases} x^2 + y^2 = 1 \\ x^3 - y = 0 \end{cases}$$

- bằng các phương pháp lặp và phương pháp Newton (khi tính toán lấy 7 chữ số sau dấu phẩy).

CHƯƠNG 4

PHƯƠNG PHÁP SỐ TRONG ĐẠI SỐ TUYẾN TÍNH

§1. MỞ ĐẦU VỀ HỆ ĐẠI SỐ TUYẾN TÍNH

1.1 Đặt vấn đề

T RONG thực tế, nhiều mô hình toán học dẫn đến việc tìm nghiệm của hệ đại số tuyến tính:

$$\left\{ \begin{array}{l} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = a_{1,n+1} \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = a_{2,n+1} \\ \dots \dots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = a_{m,n+1} \end{array} \right. \quad (4.1)$$

trong đó $a_{ij} \in \mathbb{R}$ $\forall i, j$ là các số thực đã cho (hoặc tìm được trong mô hình toán học hoặc có thể quan sát được).

Nghiệm của hệ (4.1) là một bộ số thực (hoặc phức) có thứ tự sao cho khi thay vào (4.1) được đồng nhất thức.

Hệ (4.1) có thể viết dưới dạng ma trận:

$$Ax = b, \quad (4.2)$$

trong đó,

- $A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \in \mathbb{R}^{m \times n}$ là ma trận hệ số;
- $b = (a_{1,n+1}, a_{2,n+1}, \dots, a_{m,n+1})^t \in \mathbb{R}^m$ là vectơ (cột) về phải;
- $x = (x_1, x_2, \dots, x_n)^t \in \mathbb{R}^n$ là vectơ nghiệm cần tìm.

Trường hợp $m = n$ (số phương trình bằng số ẩn) thì A là ma trận vuông cấp n . Trong giáo trình này ta chỉ đề cập đến trường hợp $m = n$ còn trường hợp $m \neq n$ nếu có ta sẽ xét cụ thể.

1.2 Phương pháp Cramer

Xét hệ phương trình đại số tuyến tính:

$$Ax = b, \quad (4.3)$$

trong đó A là ma trận vuông cấp n . Ta đã biết, điều kiện cần và đủ để hệ (4.3) tồn tại duy nhất nghiệm đó là A là ma trận không suy biến ($\det A \neq 0$). Khi đó, nghiệm duy nhất $x^* \in \mathbb{R}^n$ được tính bởi công thức

$$x^* = A^{-1}b, \quad (4.4)$$

hoặc theo công thức Cramer (đã xét trong giáo trình đại số tuyến tính):

$$x_i^* = \frac{\Delta_i}{\Delta}, \quad i = \overline{1, n}; \quad (4.5)$$

trong đó $\Delta = \det A$ và Δ_i là định thức của ma trận nhân được từ A bằng cách thay cột thứ i bởi cột b .

Như vậy, muốn giải hệ phương trình (4.3) bằng phương pháp Cramer, ta phải tính $n + 1$ định thức cấp n . Mỗi định thức cấp n tính theo khai triển Laplace cần $n! - 1$ phép cộng (trừ) và $n!(n - 1)$ phép nhân. Vậy với hệ n phương trình n ẩn cần $(n + 1)(n! - 1) = (n + 1)! - (n + 1)$ phép cộng và $n!(n^2 - 1)$ phép nhân, ngoài ra còn n phép chia để tìm nghiệm theo công

thức (4.5). Sau đây ta nêu một vài số liệu để thấy rằng khi n tăng thì số phép tính tăng rất nhanh tới mức không có khả năng tính toán, ngay cả khi sử dụng những máy tính hiện đại nhất hiện nay (bảng 4.1).

n	Số phép nhân	Số phép cộng
4	360	115
5	2880	716
10	359,251,810	39,916,791

Bảng 4.1

Để khắc phục khối lượng tính toán, ta sẽ xét một vài phương pháp giải đúng và giải gần đúng sau.

§2. PHƯƠNG PHÁP GIẢI ĐÚNG VÀ CHƯƠNG TRÌNH MATLAB

Các phương pháp sau cho phép tìm nghiệm đúng. Nghiệm gần đúng thu được là do sự quy tròn số sinh ra trong quá trình tính toán.

2.1 Phương pháp Gauss

Nội dung của phương pháp là loại trừ ẩn, nhằm giảm khối lượng tính toán. Phương pháp Gauss được thực hiện qua hai quá trình thuận và nghịch để tìm nghiệm của hệ đại số tuyến tính.

Để cách trình bày được đơn giản, ta xét hệ bốn phương trình bốn ẩn

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 = a_{15} \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 = a_{25} \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 = a_{35} \\ a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 = a_{45} \end{cases} \quad (4.6)$$

Ta luôn có thể giả thiết $a_{11} \neq 0$. Trường hợp $a_{11} = 0$, hệ (4.6) át phải có ít nhất một phần tử $a_{i1} \neq 0$, ($i = 2, 3, 4$). Chẳng hạn $a_{31} \neq 0$, đổi chỗ phương trình thứ ba cho phương trình đầu và đánh số lại ta được $a_{11} \neq 0$.

2.1.1 Quá trình thuận

Đưa hệ (4.6) về dạng có ma trận hệ số dạng tam giác trên.

- Bước 1: Giữ nguyên phương trình đầu của hệ (4.6). Từ phương trình thứ hai trở đi, các hệ số, kể cả vé phải được biến đổi theo công thức:

$$a_{ij}^{(1)} = a_{ij} - \frac{a_{i1}a_{1j}}{a_{11}}, \quad i = 2, 3, 4; \quad j = 2, 3, 4, 5 \quad (4.7)$$

còn

$$a_{i1}^{(1)} = a_{i1} - \frac{a_{i1}a_{11}}{a_{11}} = 0, \quad i = 2, 3, 4;$$

nghĩa là, ẩn x_1 của hệ (4.6) được loại trừ ra khỏi các phương trình thứ hai, ba và tư. Kết thúc bước một ta thu được hệ

$$\left\{ \begin{array}{l} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 = a_{15} \\ a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 + a_{24}^{(1)}x_4 = a_{25}^{(1)} \\ a_{32}^{(1)}x_2 + a_{33}^{(1)}x_3 + a_{34}^{(1)}x_4 = a_{35}^{(1)} \\ a_{42}^{(1)}x_2 + a_{43}^{(1)}x_3 + a_{44}^{(1)}x_4 = a_{45}^{(1)} \end{array} \right. \quad (4.8)$$

- Bước 2: Loại trừ ẩn x_2 ra khỏi hệ (4.8) đối với phương trình thứ ba và tư.

Với giả thiết $a_{22}^{(1)} \neq 0$, các hệ số của hai phương trình sau được biến đổi theo công thức:

$$\begin{aligned} a_{i2}^{(2)} &= 0, \quad i = 3, 4; \\ a_{ij}^{(2)} &= a_{ij}^{(1)} - \frac{a_{i2}^{(1)} \cdot a_{2j}^{(1)}}{a_{22}^{(1)}}, \quad i = 3, 4; \quad j = 3, 4, 5. \end{aligned} \quad (4.9)$$

Kết thúc bước 2 ta được hệ

$$\left\{ \begin{array}{l} a_{11}x_1 + a_{13}x_3 + a_{14}x_4 = a_{15} \\ a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 + a_{24}^{(1)}x_4 = a_{25}^{(1)} \\ a_{33}^{(2)}x_3 + a_{34}^{(2)}x_4 = a_{35}^{(2)} \\ a_{43}^{(2)}x_3 + a_{44}^{(2)}x_4 = a_{45}^{(2)} \end{array} \right. \quad (4.10)$$

- Bước 3: Loại trừ ẩn x_3 ra khỏi phương trình thứ tư của hệ (4.10). Giả thiết $a_{33}^{(2)} \neq 0$, các hệ số của phương trình thứ tư được biến đổi như sau:

$$\begin{aligned} a_{43}^{(3)} &= 0; \\ a_{4j}^{(3)} &= a_{4j}^{(2)} - \frac{a_{43}^{(2)} \cdot a_{3j}^{(2)}}{a_{33}^{(2)}}, \quad j = 4, 5. \end{aligned} \quad (4.11)$$

Kết thúc bước 3, ta được hệ

$$\left\{ \begin{array}{lcl} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 & = a_{15} \\ a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 + a_{24}^{(1)}x_4 & = a_{25}^{(1)} \\ a_{33}^{(2)}x_3 + a_{34}^{(2)}x_4 & = a_{35}^{(2)} \\ a_{44}^{(3)}x_4 & = a_{45}^{(3)} \end{array} \right. \quad (4.12)$$

Hệ (4.12) là hệ có ma trận hệ số dạng tam giác trên. Quá trình thuận kết thúc.

2.1.2. Quá trình nghịch

Từ hệ (4.12), giải từ dưới lên ta được nghiệm của hệ (4.6) và được tính theo công thức:

$$x_i = \frac{1}{a_{ii}^{(i-1)}} \left[a_{i,5}^{(i-1)} - \sum_{j=i+1}^4 a_{ij}^{(i-1)} x_j \right], \quad i = 4, 3, 2, 1. \quad (4.13)$$

2.1.3. Thuật toán (Sơ đồ tính toán)

Quá trình thuận chuyển từ hệ (4.6) về dạng (4.12) thực chất là biến đổi hệ số (kể cả về phải). Ta có thể mô tả trên sơ đồ tính toán như sau:

	<i>A</i>	<i>B</i>	Các công thức tính
$\bar{A}_4 = A B$	$a_{11} \ a_{12} \ a_{13} \ a_{14}$ $a_{21} \ a_{22} \ a_{23} \ a_{24}$ $a_{31} \ a_{32} \ a_{33} \ a_{34}$ $a_{41} \ a_{42} \ a_{43} \ a_{44}$	a_{15} a_{25} a_{35} a_{45}	(a)
\bar{A}_3	$a_{22}^{(1)} \ a_{23}^{(1)} \ a_{24}^{(1)}$ $a_{32}^{(1)} \ a_{33}^{(1)} \ a_{34}^{(1)}$ $a_{42}^{(1)} \ a_{43}^{(1)} \ a_{44}^{(1)}$	$a_{25}^{(1)}$ $a_{35}^{(1)}$ $a_{45}^{(1)}$	(b) $a_{ij}^{(1)} = a_{ij} - \frac{a_{i1}a_{1j}}{a_{11}}$ $i = 2, 3, 4; j = 2, 3, 4, 5$
\bar{A}_2	$a_{33}^{(2)} \ a_{34}^{(2)}$ $a_{43}^{(2)} \ a_{44}^{(2)}$	$a_{35}^{(2)}$ $a_{45}^{(2)}$	(c) $a_{ij}^{(2)} = a_{ij}^{(1)} - \frac{a_{i2}^{(1)}a_{2j}^{(1)}}{a_{22}^{(1)}}$ $i = 3, 4; j = 3, 4, 5$
\bar{A}_1	$a_{44}^{(3)}$	$a_{45}^{(3)}$ (d)	$a_{4j}^{(3)} = a_{4j}^{(2)} - \frac{a_{43}^{(2)}a_{3j}^{(2)}}{a_{33}^{(2)}}, j = 4, 5$

Các hàng a, b, c, d tạo nên hệ dạng (4.12). Quá trình thuận kết thúc.

Quá trình nghịch: Nghiệm $x = (x_1, x_2, x_3, x_4)^t$ của hệ (4.6) được tính như sau:

- Từ (d) suy ra $x_4 = \frac{a_{45}^{(3)}}{a_{44}^{(3)}}$;
- Từ (c) suy ra $x_3 = \frac{1}{a_{33}^{(2)}} [a_{35}^{(2)} - a_{34}^{(2)}x_4]$;
- Từ (b) suy ra $x_2 = \frac{1}{a_{22}^{(1)}} [a_{25}^{(1)} - a_{23}^{(1)}x_3 - a_{24}^{(1)}x_4]$;
- Từ (a) suy ra $x_1 = \frac{1}{a_{11}} [a_{15} - a_{12}x_2 - a_{13}x_3 - a_{14}x_4]$.

Chú ý 4.1. 1. Với hệ bốn phương trình, bốn ẩn, quá trình thuận được tiến hành qua ba bước. Hoàn toàn có thể mở rộng đối với hệ n phương trình, n ẩn (khi đó cần $n - 1$ bước).

2. Thuật toán Gauss có thể áp dụng được cả đối với hệ vô định hoặc vô nghiệm. Do đó, không cần phải tính trước $\det A$.

2.2 Chương trình MATLAB

Chương trình MATLAB minh họa thuật toán giải hệ đại số tuyến tính $Ax = b$ bằng phương pháp khử Gauss.

```
function x = gauss(A,b)
n = size(A,1);
b = b(:); % Đảm bảo vector b luôn ở dạng cột
nb = n+1; Ab = [A b]; % ma tran mo rong
% Qua trinh thuan
for i = 1:n-1
    for j = i+1:n
        Ab(j,i:nb) = Ab(j,i:nb)-Ab(j,i)*Ab(i,i:nb)/Ab(i,i);
    end
end
% Qua trinh nghich
x=zeros(n,1);
x(n) = Ab(n,nb)/Ab(n,n);
for i = n-1:-1:1
    x(i) = (Ab(i,nb) - Ab(i,i+1:n)*x(i+1:n))/Ab(i,i);
end
end
```

Ví dụ 4.1. Áp dụng chương trình trên, giải hệ phương trình

$$\begin{cases} 7.9x_1 + 5.6x_2 + 5.7x_3 - 7.2x_4 = 6.68 \\ 8.5x_1 - 4.8x_2 + 0.8x_3 + 3.5x_4 = 9.95 \\ 4.3x_1 + 4.2x_2 - 3.2x_3 + 9.3x_4 = 8.6 \\ 3.2x_1 - 1.4x_2 - 8.9x_3 + 3.3x_4 = 1.0 \end{cases}$$

Nghiệm tính theo chương trình MATLAB là:

$$x_1 = 0.96710; \quad x_2 = 0.12480; \quad x_3 = 0.42630; \quad x_4 = 0.56790.$$

Từ thuật toán Gauss ta thấy, quá trình thuận luôn phải giả thiết

$$a_{ii}^{(i-1)} \neq 0, \quad i = 1, 2, \dots$$

Nếu gặp $a_{ii}^{(i-1)} \approx 0$ thì theo công thức (4.13) sẽ gặp phép chia cho số " ≈ 0 " đó. Khi đó sai số sẽ lớn, đó chính là điều bất lợi của phương pháp Gauss. Để khắc phục điều đó, ta xét phương pháp dưới đây, gọi là phương pháp trù tối đai (Gauss-Jordan).

2.3 Phương pháp Gauss-Jordan (trù tối đai)

Nội dung của phương pháp là loại trù ẩn.

Xét hệ phương trình (4.3) (trường hợp $m = n$). Do phương pháp là loại trù ẩn nên thực chất là biến đổi hệ số của hệ phương trình để được hệ tương đương sao cho trong một phương trình của hệ mới chỉ còn lại một ẩn số x_k . Sự biến đổi của hệ số được mô tả trên bảng số của ma trận mở rộng

$$[\mathcal{A}|b] = \overline{\mathcal{A}}^{(0)} = \left[\begin{array}{cccccc|c} a_{11} & \dots & a_{1j} & \dots & a_{1q} & a_{1n} & a_{1,n+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{i1} & \dots & \boxed{a_{ij}} & \dots & a_{iq} & a_{in} & a_{i,n+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{p1} & \dots & a_{pj} & \dots & \boxed{a_{pq}} & a_{pn} & a_{p,n+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1} & \dots & a_{nj} & \dots & a_{nq} & a_{nn} & a_{n,n+1} \end{array} \right]$$

Trong ma trận mở rộng $[\mathcal{A}|b]$, hàng thứ i ứng với phương trình thứ i của hệ (4.3), nghĩa là $\sum_{j=1}^n a_{ij}x_j = a_{i,n+1}$, $i = \overline{1, n}$; cột thứ j ứng với ẩn x_j ($j = \overline{1, n}$) và cột về phải $a_{i,n+1}$, $i = \overline{1, n}$.

Ta nói a_{pq} là phần tử trôi của ma trận \mathcal{A} nếu

$$|a_{pq}| = \max_{1 \leq i, j \leq n} |a_{ij}|.$$

Phần tử trôi đó được gọi là phần tử giải, hàng thứ p được gọi là hàng giải và cột thứ q gọi là cột giải. Quá trình loại ẩn được tiến hành theo các bước sau:

- *Bước 1:* Giả sử a_{pq} là phần tử giải của ma trận $\overline{\mathcal{A}}^{(0)}$. Đặt

$$\overline{\mathcal{A}}^{(1)} = \left[a_{ij}^{(1)} \right], \quad i = \overline{1, n}; \quad j = \overline{1, n+1},$$

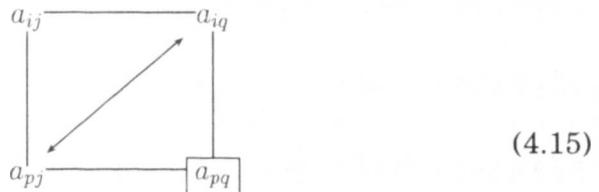
trong đó $a_{pj}^{(1)} = a_{pj}$, $j = \overline{1, n+1}$, nghĩa là giữ nguyên hàng giải thứ p . Các phần tử khác của $\bar{A}^{(1)}$ được biến đổi theo công thức:

$$a_{ij}^{(1)} = a_{ij} - \frac{a_{pq}a_{iq}}{a_{pq}}, \quad i = \overline{1, n}, \quad i \neq p, \quad j = \overline{1, n+1}. \quad (4.14)$$

Từ (4.14), ta có với $j = q$ thì

$$a_{iq}^{(1)} = a_{iq} - \frac{a_{pq}a_{iq}}{a_{pq}} = 0, \quad i = \overline{1, n}, \quad i \neq p,$$

nghĩa là các phần tử ở cột giải đều bằng 0 trừ $a_{pq} = a_{pq}^{(1)}$ của hàng giải. Ta có thể mô tả cách tính các phần tử $a_{ij}^{(1)}$ của ma trận $\bar{A}^{(1)}$ trừ hàng giải và cột giải (kể cả cột về phải) theo sơ đồ hình chữ nhật:



Từ đó, ta thu được hệ mới có ma trận mở rộng

$$\bar{A}^{(1)} = \left[\begin{array}{ccccccc|c} a_{11}^{(1)} & \dots & a_{1j}^{(1)} & \dots & 0 & \dots & a_{1n}^{(1)} & a_{1,n+1}^{(1)} \\ \vdots & \vdots \\ a_{i1}^{(1)} & \dots & a_{ij}^{(1)} & \dots & 0 & \dots & a_{in}^{(1)} & a_{i,n+1}^{(1)} \\ \vdots & \vdots \\ a_{p1}^{(1)} & \dots & a_{pj}^{(1)} & \dots & a_{pq}^{(1)} & \dots & a_{pn}^{(1)} & a_{p,n+1}^{(1)} \\ \vdots & \vdots \\ a_{n1}^{(1)} & \dots & a_{nj}^{(1)} & \dots & 0 & \dots & a_{nn}^{(1)} & a_{n,n+1}^{(1)} \end{array} \right]$$

- *Bước 2:*

Trong $\bar{A}^{(1)}$, chọn phần tử giải là $a_{rs}^{(1)}$ với điều kiện $r \neq p$, $s \neq q$. Giữ nguyên hàng thứ r là hàng giải và đặt $a_{rs}^{(2)} = a_{rs}^{(1)}$, $s = \overline{1, n+1}$. Các phần tử trên cột giải s đều bằng 0 trừ phần tử giải, các phần tử khác

được tính theo quy tắc hình chữ nhật (4.15). Để dàng thấy rằng, các phần tử bằng "0" ở trong cột giải q ở bước trước vẫn bằng "0". Khi đó, ta thu được $\bar{A}^{(2)}$.

- Quá trình trên được lặp lại, sau n bước ta thu được hệ phương trình mà mỗi phương trình chỉ còn một ẩn số. Từ đó ta suy ra nghiệm của hệ (4.3).

Về khái lượng tính toán, đối với hệ n phương trình n ẩn, người ta dựa vào các công thức tính để đếm các phép nhân, chia. Số lượng các phép tính cụ thể như sau:

- Nhân và chia: $N = \frac{n}{3} (n^2 + 3n - 1)$;
- Cộng và trừ: $C = \frac{n}{6} (2n^2 + 3n - 5)$.

Ta xét một vài trường hợp để so sánh với phương pháp Cramer

n	N	C
3	17	11
4	36	26
5	65	50
10	430	375

Bảng 4.2

Chú ý 4.2. 1. Nếu chọn các phần tử giải chính là các phần tử trên đường chéo chính $a_{ii}^{(i-1)}$ thì phương pháp trên được gọi là phương pháp Gauss. Vì vậy, phương pháp Gauss-Jordan tránh được trường hợp $a_{ii}^{(i-1)} \approx 0$ trong phép chia của công thức (4.14).

2. Trong công thức (4.14), ta gấp phép chia cho phần tử giải a_{pq} , vì vậy có thể chọn phần tử giải là số "1" để tránh được phép chia (nếu có).
3. Phương pháp Gauss-Jordan áp dụng được kể cả với hệ vô định, vô nghiệm và hệ có số phương trình khác số ẩn.

Chương trình MATLAB

```

function A=gauss_jordan(A)
% gauss-jordan giải hệ Ax=b bằng phương pháp Gauss-Jordan
% Input: A là ma trận mở rộng của hệ
% Output: A là ma trận mở rộng sau khi đã biến đổi
% cột cuối cùng của A chính là nghiệm x của hệ ban đầu.
s=size(A);
n=min(s);
for i=1:n
    %
    [t,r]=max(abs(A(i:end,i)));
    t=A(r+i-1,:);
    A(r+i-1,:)=A(i,:);
    A(i,:)=t;
    % Elimination
    A(i,[1:i-1 i+1:end])= A(i,[1:i-1 i+1:end])/A(i,i);
    A(i,i)=1;
    s=A(i,i+1:end);
    for j=[1:i-1 i+1:n]
        A(j,i+1:end)= A(j,i+1:end)-A(j,i).*s;
    end
    A([1:i-1 i+1:end],i)=0;
end

```

Ví dụ 4.2. Giải hệ phương trình

$$\begin{cases} x_1 + x_2 - 3x_3 + 2x_4 = 6 \\ x_1 - 2x_2 - x_4 = -6 \\ x_2 + x_3 + 3x_4 = 16 \\ 2x_1 - 3x_2 + 2x_3 = 6 \end{cases}$$

Sử dụng chương trình MATLAB, ta thu được kết quả nghiệm của hệ là $x_1 = 8; x_2 = 6; x_3 = 4; x_4 = 2$.

2.4 Phương pháp Cholesky (khai căn)

Nội dung phương pháp là dựa vào việc giải hệ $Ax = b$ về việc giải liên tiếp hai hệ có ma trận dạng tam giác trên, dưới. Để việc tính toán được thuận lợi, Cholesky xét lại hệ phương trình (4.3), trong đó $A = [a_{ij}]_n$ là ma trận đối xứng, nghĩa là $a_{ij} = a_{ji}$, $\forall i, j = \overline{1, n}$. Ta đã biết, nếu $\det A \neq 0$ thì A luôn phân tích được thành tích của một ma trận tam giác trên và một ma trận tam giác dưới:

$$A = P \cdot Q. \quad (4.16)$$

Đặc biệt, nếu A là ma trận đối xứng thì $A = Q^t Q$, trong đó $Q^t = P$ là ma trận chuyển vị của

$$Q = \begin{bmatrix} q_{11} & q_{12} & \dots & q_{1n} \\ 0 & q_{22} & \dots & q_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & q_{nn} \end{bmatrix}.$$

Từ $Ax = b$ ta suy ra:

$$Q^t Q x = b. \quad (4.17)$$

Đặt:

$$Qx = y, \quad (4.18)$$

thay vào (4.17) ta thu được:

$$Q^t y = b. \quad (4.19)$$

Giải (4.19) ta có y , sau đó thay vào (4.18) và giải hệ này ta có x là nghiệm của hệ (4.3). Thuật toán của phương pháp được tiến hành qua các bước sau:

2.3.1. Phân tích $A = Q^t Q$

Ta có:

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} = \begin{bmatrix} q_{11} & 0 & \dots & 0 \\ q_{21} & q_{22} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ q_{n1} & q_{n2} & \dots & q_{nn} \end{bmatrix} \begin{bmatrix} q_{11} & q_{12} & \dots & q_{1n} \\ 0 & q_{22} & \dots & q_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & q_{nn} \end{bmatrix}$$

$$\Rightarrow a_{ij} = \sum_{k=1}^n q_{ki}q_{kj}, \quad \forall i \leq j$$

hay

$$\begin{aligned} q_{11} &= \sqrt{a_{11}}, \quad q_{1j} = \frac{a_{1j}}{q_{11}}, \quad j = \overline{2, n}; \\ q_{ii} &= \left(a_{ii} - \sum_{k=1}^{i-1} q_{ki}^2 \right)^{1/2}, \quad i = \overline{2, n}; \\ q_{ij} &= \frac{a_{ij} - \sum_{k=1}^{i-1} q_{ki}q_{kj}}{q_{ii}}, \quad (i < j); \quad q_{ij} = 0 \quad (i > j). \end{aligned} \tag{4.20}$$

Công thức (4.20) luôn được thực hiện vì

$$\det A = \det Q^t \cdot \det Q = \prod_{i=1}^n q_{ii} \cdot \prod_{i=1}^n q_{ii} = \prod_{i=1}^n q_{ii}^2 \neq 0,$$

nên $q_{ii} \neq 0, \quad i = \overline{1, n}$.

2.3.2. Giải hệ dạng (4.19)

Từ $Q^t y = b$ ta có

$$\begin{bmatrix} q_{11} & 0 & \dots & 0 \\ q_{21} & q_{22} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ q_{n1} & q_{n2} & \dots & q_{nn} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

hay

$$\begin{aligned} q_{11}y_1 &= b_1 \\ q_{12}y_1 + q_{22}y_2 &= b_2 \\ &\vdots \\ q_{1n}y_1 + q_{2n}y_2 + \dots + q_{nn}y_n &= b_n. \end{aligned}$$

Từ đó, ta suy ra:

$$\begin{aligned} y_1 &= \frac{b_1}{q_{11}}; \\ y_i &= \frac{b_i - \sum_{k=1}^{i-1} q_{ik} y_k}{q_{ii}}, \quad (i > 1). \end{aligned} \quad (4.21)$$

2.3.3. Giải hệ dạng (4.18)

Từ $Qx = y$ ta có

$$\left[\begin{array}{cccc} q_{11} & q_{12} & \dots & q_{1n} \\ 0 & q_{22} & \dots & q_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & q_{nn} \end{array} \right] \left[\begin{array}{c} x_1 \\ x_2 \\ \vdots \\ x_n \end{array} \right] = \left[\begin{array}{c} y_1 \\ y_2 \\ \vdots \\ y_n \end{array} \right]$$

Từ đó suy ra:

$$\begin{aligned} x_n &= \frac{y_n}{q_{nn}}; \\ x_i &= \frac{y_i - \sum_{k=i+1}^n q_{ik} x_k}{q_{ii}}, \quad (i < n). \end{aligned} \quad (4.22)$$

Thuật toán được thực hiện cụ thể như sau:

- Bước 1:* Từ hệ phương trình $Ax = b$ với A là ma trận đối xứng, ta phân tích $A = Q^t Q$, trong đó các phần tử q_{ij} được tính theo công thức (4.20).
- Bước 2:* Giải hệ $Q^t y = b$ với vector $y = (y_1, y_2, \dots, y_n)^t$ được tính theo công thức (4.21).
- Bước 3:* Giải hệ $Qx = y$, ta thu được vector nghiệm x được tìm theo công thức (4.22).

Chú ý 4.3. Trong quá trình tính chỉ lấy dấu $+$ và có thể gặp số phức (khi khai căn số âm) nhưng không ảnh hưởng đến kết quả.

Ví dụ 4.3. Giải hệ phương trình $Ax = b$, trong đó:

$$A = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 5 & 1 \\ 1 & 1 & 3 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

Thực hiện theo các bước của thuật toán ta thu được:

- *Bước 1:* Phân tích $A = Q^t Q$ theo công thức (4.20) ta thu được:

$$Q = \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ 0 & q_{22} & q_{23} \\ 0 & 0 & q_{33} \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

- *Bước 2:* Giải hệ $Q^t y = b$ theo công thức (4.21) với

$$Q^t = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

ta thu được $y = (1, -1, -1)^t$.

- *Bước 3:* Giải $Qx = y$ theo công thức (4.22) ta được $x = (6, -2, -1)^t$ là nghiệm của hệ ban đầu.

2.3.4. Chương trình MATLAB

```
function Q=chodec(A)
% Find the lower triangular matrix Q in Cholesky
% decomposition A=Q^tQ
% Input: A is (often) positive definite and symmetric
% Output: Q
n=size(A,1);
Q=zeros(n,n);
for i=1:n
    for j=i:n
        if (j==1)
            s=A(i,i);
        else
            s=A(i,j)-Q(1:i-1,i)'*Q(1:i-1,j);
        end
        if j>i
            Q(i,j)=s/Q(i,i);
        end
    end
end
```

```

    else
        Q(i,i)=sqrt(s);
    end
end

function y=uptriag(Q,b)
% This function solves the upper triangular system Q^t y = b
% Input: upper triangular matrix Q; vector b
% Output: solution y of above system
n=size(Q,1);
y=zeros(n,1);
y(1)=b(1)/Q(1,1);
for i=2:n
    s=b(i)-Q(1:i-1,i)'*y(1:i-1);
    y(i)=s/Q(i,i);
end
end

function x=lowtriag(Q,y)
% This function solves the lower triangular system Qx = y
% Input: lower triangular matrix Q; vector y
% Output: solution x of above system
n=size(Q,1);
x=zeros(n,1);
x(n)=y(n)/Q(n,n);
for i=n-1:-1:1
    s=y(i)-Q(i,i+1:n)*x(i+1:n);
    x(i)=s/Q(i,i);
end
end

```

2.5 Công thức truy đuổi giải hệ có ma trận dạng ba đường chéo

Xét hệ phương trình dạng:

$$\begin{cases} x_0 - m_1 x_1 = n_1 \\ a_i x_{i-1} - c_i x_i + b_i x_{i+1} = -f_i, \quad i = \overline{1, N-1} \\ -m_2 x_{N-1} + x_N = n_2. \end{cases} \quad (4.23)$$

trong đó:

$$\begin{aligned} a_i > 0, \quad b_i > 0, \quad c_i \geq a_i + b_i, \quad i = \overline{1, N-1} \\ 0 \leq m_1 \leq 1; \quad 0 \leq m_2 \leq 1; \quad m_1 + m_2 < 2. \end{aligned} \quad (4.24)$$

2.4.1. Công thức truy đuổi từ phải

Tìm nghiệm của hệ (4.23) trong dạng

$$x_i = \alpha_{i+1} x_{i+1} + \beta_{i+1}, \quad i = N-1, N-2, \dots, 1, 0, \quad (4.25)$$

trong đó α_i, β_i là các hệ số cần xác định.

Bằng cách thay (4.25) vào (4.23) và sử dụng các điều kiện (4.24), ta được công thức tính như dưới đây.

Đặt $\alpha_1 = m_1; \beta_1 = n_1$ và tính

$$\alpha_{i+1} = \frac{b_i}{c_i - a_i \alpha_i}; \quad \beta_{i+1} = \frac{a_i \beta_i + f_i}{c_i - a_i \alpha_i}, \quad i = \overline{1, N-1}, \quad (4.26)$$

$$\begin{aligned} x_N &= \frac{n_2 + m_2 + \beta_N}{1 - m_2 \alpha_N}; \\ x_i &= \alpha_{i+1} x_{i+1} + \beta_{i+1}, \quad i = N-1, N-2, \dots, 1, 0. \end{aligned} \quad (4.27)$$

Tóm lại, xuất phát từ α_1, β_1 tính $\alpha_{i+1}, \beta_{i+1}$ theo công thức (4.26). Sau đó, tính x_i ($i = N, N-1, \dots, 1, 0$) theo công thức (4.27). Quá trình tính theo các công thức (4.26), (4.27) gọi là công thức truy đuổi phải (vì các nghiệm x_i được tính theo thứ tự từ phải qua trái).

2.4.2. Công thức truy đuổi từ trái

Tìm nghiệm của hệ (4.23) trong dạng:

$$x_{i+1} = \xi_{i+1}x_i + \eta_{i+1}. \quad (4.28)$$

Khi đó ta có công thức tính:

$$\begin{aligned}\xi_N &= m_2; \quad \eta_N = n_2; \\ \xi_i &= \frac{a_i}{c_i - \xi_{i+1}b_i}; \quad \beta_i = \frac{b_i\eta_{i+1} + f_i}{c_i - \xi_{i+1}b_i}, \quad i = N-1, N-2, \dots, 1, 0; \\ x_0 &= \frac{n_1 + m_1n_1}{1 - m_1\xi_1}; \\ x_{i+1} &= \xi_{i+1}x_i + \eta_{i+1}, \quad i = 0, 1, \dots, N-1.\end{aligned} \quad (4.29)$$

Quá trình tìm nghiệm xuất phát từ x_0 , gọi là công thức truy đuổi trái (tính dần từ trái sang phải).

2.6 Chương trình MATLAB

```
function x=forward_pursuing(n,m,A,B,C,F)
% Solves a three diagonals system by using the
% forward pursuing method
% Input: n,m in R^2; A,B,C,F in R^{N-1}
% Output: x in R^{N+1}
x=zeros(N+1,1);
alpha = zeros(N+1,1);
alpha(1) = m(1);
beta = zeros(N+1,1);
beta(1) = n(1);
for i = 1:N
    alpha(i+1) = B(i)/(C(i)-A(i)*alpha(i));
    beta(i+1) = (A(i)*beta(i)+F(i))/(C(i)-A(i)*alpha(i));
end
x(N+1) = (n(2)+m(2)*beta(N+1))/(1-m(2)*alpha(N+1));
for i = N:-1:1
    x(i) = alpha(i+1)*x(i+1)+beta(i+1);
```

```
end
end
```

Ví dụ 4.4. Áp dụng chương trình MATLAB giải hệ

$$\begin{cases} -2x_0 + x_1 = -1 \\ x_{i-1} - 3x_i + x_{i+1} = -1, \quad i = \overline{1, 9} \\ x_9 - 2x_{10} = -1 \end{cases}$$

ta thu được nghiệm $x_i = 1, \quad i = \overline{1, 10}$.

Chú ý 4.4. Ta có thể áp dụng đồng thời cả hai công thức truy đuổi phải, trái rồi lấy nghiệm là trung bình cộng của chúng để đạt được độ chính xác cao hơn (do sai số quy tròn).

Các phương pháp giải hệ đại số tuyến tính trong §2 là các phương pháp giải đúng. Song kết quả vẫn có thể là gần đúng vì gặp sai số tính toán sinh ra do sự quy tròn số trong các phép tính.

§3. PHƯƠNG PHÁP LẮP ĐƠN

3.1 Chuẩn của ma trận và sự hội tụ của dãy ma trận

Định nghĩa 4.1. Chuẩn của ma trận $A = [a_{ij}]_{m \times n} \in \mathbb{R}^{m \times n}$ ký hiệu bởi $\|A\|$ là số thực không âm thỏa mãn các điều kiện sau:

1. $\|A\| \geq 0$, dấu " $=$ " xảy ra khi và chỉ khi $A = 0$ (ma trận không);
2. $\|kA\| = |k| \|A\|$, k – const;
3. $\|A + B\| \leq \|A\| + \|B\|$.

Một số chuẩn ma trận thường được sử dụng:

$$\begin{aligned} \|A\|_{(\infty)} &= \max_{1 \leq i \leq m} \sum_{j=1}^n |a_{ij}| \quad (\text{theo hàng}); \\ \|A\|_{(1)} &= \max_{1 \leq j \leq n} \sum_{i=1}^m |a_{ij}| \quad (\text{theo cột}); \\ \|A\|_{(2)} &= \left[\sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^2 \right]^{1/2} \quad (\text{chuẩn Euclide}). \end{aligned} \tag{4.31}$$

Dễ dàng kiểm tra ba dạng chuẩn trên thỏa mãn các điều kiện về chuẩn, đồng thời

$$\begin{aligned}\|A \cdot B\| &\leq \|A\| \cdot \|B\|, \\ \|A^k\| &\leq \|A\|^k.\end{aligned}$$

Xét trường hợp khi $n = 1$, ta có vector $A_{m \times 1} = x = (x_1, x_2, \dots, x_m)^t \in \mathbb{R}^m$ và ta cũng có các chuẩn của vector x như sau:

$$\begin{aligned}\|x\|_{(\infty)} &= \max_{1 \leq i \leq m} |x_i| \quad (\text{chuẩn cực đại}) \\ \|x\|_{(1)} &= \sum_{i=1}^m |x_i| \quad (\text{chuẩn tuyệt đối}) \\ \|x\|_{(2)} &= \left[\sum_{i=1}^m |x_i|^2 \right]^{1/2} \quad (\text{chuẩn Euclide})\end{aligned}\tag{4.32}$$

đồng thời ta cũng có:

$$\|Ax\|_{(p)} \leq \|A\|_{(p)} \cdot \|x\|_{(p)}, \quad p = \infty, 1, 2.$$

Định nghĩa 4.2. Dãy ma trận $A^{(k)} = \left\{ [a_{ij}^{(k)}] \right\}_{k=1,2,\dots}$ được gọi là hội tụ tới ma trận $A = [a_{ij}]_{m \times n}$ và ký hiệu là $\lim_{k \rightarrow \infty} A^{(k)} = A$, nếu

$$\lim_{k \rightarrow \infty} a_{ij}^{(k)} = a_{ij} \quad \forall i, j \quad (\text{hội tụ theo dãy phần tử})$$

hoặc

$$\|A^{(k)} - A\|_{(p)} \rightarrow 0 \text{ khi } k \rightarrow \infty \quad (p = \infty, 1, 2) \quad (\text{hội tụ theo chuẩn}).$$

Từ đây về sau ta sử dụng sự hội tụ của dãy ma trận $A^{(k)}$ tới A theo nghĩa hội tụ theo chuẩn (hội tụ theo chuẩn hay theo phần tử là tương đương).
Trường hợp $n = 1$ ta có vector cột

$$A_{m \times 1} = x = (x_1, x_2, \dots, x_m)^t$$

và cũng có khái niệm về sự hội tụ của dãy vector như dưới đây.

Định nghĩa 4.3. Dãy vector $x^{(k)} = (x_1^{(k)}, x_2^{(k)}, \dots, x_m^{(k)})^t$ được gọi là hội tụ tới vector $x = (x_1, x_2, \dots, x_m)^t$ khi $k \rightarrow \infty$ và ký hiệu $\lim_{k \rightarrow \infty} x^{(k)} = x$ nếu

$$\lim_{k \rightarrow \infty} x_i^{(k)} = x_i \quad \forall i = 1, m$$

hoặc

$$\|x^{(k)} - x\|_{(p)} \rightarrow 0 \text{ khi } k \rightarrow \infty, \quad p = \infty, 1, 2.$$

3.2 Phương pháp lặp đơn (lặp cổ điển)

3.2.1. Nội dung phương pháp

Xét hệ (4.3). Viết lại hệ trên dưới dạng:

$$x = \alpha x + \beta, \quad (4.33)$$

trong đó α là ma trận vuông cấp n . Hệ dạng (4.3) luôn có thể viết được dưới dạng (4.33), chẳng hạn:

$$Ax + x = bx + x \implies x = (I + A)x - b,$$

trong đó I là ma trận đơn vị và $\alpha = I + A$; $\beta = -b$.

Chọn xấp xỉ đầu bất kỳ $x^{(0)} = (x_1^{(0)}, x_2^{(0)}, \dots, x_n^{(0)})^t$. Quá trình lặp được tính theo công thức:

$$x^{(k)} = \alpha x^{(k-1)} + \beta, \quad k = 1, 2, \dots \quad (4.34)$$

Vấn đề đặt ra là với điều kiện nào thì quá trình lặp (4.34) hội tụ tới nghiêm đúng x^* của hệ (4.3) hoặc (4.33). Ta công nhận kết quả sau (xem chứng minh chi tiết trong tài liệu tham khảo).

3.2.2. Điều kiện hội tụ của quá trình lặp

Định lý 4.1. Nếu một chuẩn nào đó của ma trận α thỏa mãn điều kiện

$$\|\alpha\|_{(p)} \leq q \quad (p = \infty, 1, 2), \quad (4.35)$$

trong đó $0 < q < 1$, thì hệ (4.33) có nghiêm duy nhất x^* và dãy vector $x^{(k)}$ được tính theo công thức lặp (4.34) sẽ hội tụ tới x^* theo chuẩn tương ứng, tức là

$$\|x^{(k)} - x^*\|_{(p)} \rightarrow 0 \text{ khi } k \rightarrow \infty \quad (p = \infty, 1, 2).$$

3.2.3. Đánh giá sai số

Khi tính toán thì quá trình lặp (4.34) không thể kéo dài ra vô hạn mà phải dừng ở bước thứ k nào đó ta được $x^{(k)} \approx x^*$. Sai số được đánh giá theo công thức

$$\|x^{(k)} - x^*\|_{(p)} \leq \frac{q}{1-q} \|x^{(k)} - x^{(k-1)}\|_{(p)}, \quad p = \infty, 1, 2 \quad (4.36)$$

hoặc

$$\|x^{(k)} - x^*\|_{(p)} \leq \frac{q^k}{1-q} \|x^{(1)} - x^{(0)}\|_{(p)}, \quad p = \infty, 1, 2. \quad (4.37)$$

Để sai số giữa nghiệm đúng x^* và nghiệm gần đúng $x^{(k)}$ nhỏ hơn $\varepsilon > 0$ cho trước thì từ (4.36) ta suy ra

$$\|x^{(k)} - x^{(k-1)}\|_{(p)} < \frac{(1-q)}{q} \varepsilon = \delta. \quad (4.38)$$

Như vậy, khi cài đặt trên máy, ta so sánh độ lệch giữa hai phần tử lặp liên tiếp, nếu thỏa mãn (4.38) thì dừng máy. Còn theo (4.37) ta sẽ có số phép lặp cần thiết phải thực hiện để đạt được sai số nhỏ hơn ε . Số phép lặp k cần thỏa mãn:

$$q^k \leq \frac{(1-q)\varepsilon}{\|x^{(1)} - x^{(0)}\|_{(p)}},$$

hay

$$k \geq \log_q \frac{(1-q)\varepsilon}{\|x^{(1)} - x^{(0)}\|_{(p)}} = J.$$

Khi đó chọn $k = [J] + 1$.

Ví dụ 4.5. Bằng phương pháp lặp đơn tìm nghiệm gần đúng của hệ

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

sao cho đạt sai số tuyệt đối không vượt quá 0.02.

Giải:

Viết lại hệ trên trong dạng $x = \alpha x + \beta$, trong đó:

$$\alpha = \begin{bmatrix} 0 & -0.06 & 0.02 \\ -0.03 & 0 & 0.05 \\ -0.01 & 0.02 & 0 \end{bmatrix}; \quad \beta = \begin{bmatrix} 2 \\ 3 \\ 5 \end{bmatrix}$$

Ta có:

$$\|\alpha\|_{\infty} = \max \{0.08; -0.08; -0.03\} = 0.08 < 1.$$

Chọn xấp xỉ đầu $x^{(0)} = (0, 0, 0)^t$, suy ra $x^{(1)} = (2, 3, 5)^t$.

Số phép lặp cần thiết:

$$k \geq \log_{0.08} \frac{(1 - 0.08) \times 0.02}{\|x^{(1)} - x^{(0)}\|_{\infty}} = \log_{0.08} 0.00368 = 2.2190.$$

Vậy chọn $k = [2.2190] + 1 = 3$.

Áp dụng công thức lặp ta thu được:

$$\begin{aligned} x^{(2)} &= (1.92; 3.19; 5.05)^t; \\ x^{(3)} &= (1.9094; 3.1944; 5.0446)^t. \end{aligned}$$

Vậy nếu lấy $x^* \approx x^{(3)}$ thì ta có sai số $\|x^* - x^{(3)}\|_{\infty} < 0.02$.

3.2.4. Chương trình MATLAB

```
function [X,numites]=jacobi(A,B,X0,tol,max1)
% Input: _ A is an nxn strictly diagonally dominant matrix
%         _ B is an nx1 matrix
%         _ X0 is an nx1 matrix: the initial guess
%         _ tol is the tolerance for X
%         _ max1 is the maximum of number of iteration
% Output: _ X is the jacobi approximation to the solution
%         _ numites is number of iterations
if nargin<5
    max1=100;
end
if nargin<4
    tol=1e-5;
end
n=length(B);
X=X0; numites=0;
for k=1:max1
    for i=1:n
```

```

X(i)=(B(i)-A(i,[1:i-1 i+1:n]))...
    *X0([1:i-1 i+1:n]))/A(i,i);
end
err=abs(norm(X-X0)); relerr=err/(norm(X)+eps);
X0=X; numites=numites+1;
if (err<tol)|| (relerr<tol)
    break
end
end
-----
```

§4. SỰ KHÔNG ỔN ĐỊNH CỦA HỆ ĐẠI SỐ TUYẾN TÍNH

Trong hệ đại số tuyến tính, nếu trong quá trình khảo sát thực tế cho ta các giá trị đúng của các hệ số của hệ (các phần tử của ma trận về trái A và cột về phải b) thì ta có thể áp dụng được các phương pháp giải đúng đã xét trong §2. Tuy nhiên do trong thực tế các số liệu thu được nói chung chỉ là gần đúng trong dạng số thập phân, quy tròn nên chúng ta chỉ được hệ $\tilde{A}x = \tilde{b}$. Tùy thuộc sự quy tròn mà sinh ra sự khác biệt đáng kể đối với nghiệm tìm được, có trường hợp dẫn đến sự mâu thuẫn đối với thực tiễn. Sau đây ta nêu một số ví dụ để thấy rõ điều này.

Ví dụ 4.6. Tìm nghiệm của hệ phương trình

$$\begin{cases} 10x_1 + 7x_2 + 8x_3 + 7x_4 = 32.01 \\ 7x_1 + 5x_2 + 6x_3 + 5x_4 = 22.99 \\ 8x_1 + 6x_2 + 10x_3 + 9x_4 = 32.99 \\ 7x_1 + 5x_2 + 9x_3 + 10x_4 = 31.01 \end{cases}$$

Khi giải hệ trên bằng phương pháp Gauss-Jordan ta thu được:

$$x_1 = 1.50; \quad x_2 = 0.15; \quad x_3 = 1.19; \quad x_4 = 0.89.$$

Bây giờ ta thay đổi về phải của hệ theo thứ tự là:

$$32.1; \quad 22.9; \quad 32.9; \quad 31.1.$$

Cũng giải bằng phương pháp trên ta được nghiệm:

$$x_1 = 6; \quad x_2 = -7.2; \quad x_3 = 2.9; \quad x_4 = -0.1.$$

Còn nếu về phải của hệ là 32; 23; 33; 31 thì nghiệm sẽ là $x_1 = x_2 = x_3 = x_4 = 1$.

Như vậy, về phải chỉ thay đổi chút ít đã làm cho nghiệm thay đổi đáng kể.

Ví dụ 4.7. Xét hệ phương trình:

$$\begin{cases} 2x_1 + x_2 &= 2 \\ 2x_1 + 1.01x_2 &= 2.01. \end{cases}$$

Giải hệ ta được $x_1 = 0.5; x_2 = 1$. Trong hệ trên, thay đổi một chút về hệ số trong phương trình thứ hai, ta thu được hệ:

$$\begin{cases} 2x_1 + x_2 &= 2 \\ 2.01x_1 + x_2 &= 2.05 \end{cases}$$

thì nghiệm sẽ là $x_1 = 5; x_2 = -8$. Nghiệm này khác khá xa với nghiệm của hệ ban đầu.

Ví dụ 4.8. Xét hệ phương trình

$$\begin{cases} 2x_1 + x_2 &= 2 \\ 8x_1 + 3.03x_2 &= 7.03 \end{cases}$$

Giải hệ ta được $x_1 = 0.5; x_2 = 1$. Thay đổi hệ trên bởi

$$\begin{cases} 2x_1 + x_2 &= 2 \\ 8x_1 + 3x_2 &= 7 \end{cases}$$

và giải ta được $x_1 = 0.5; x_2 = 1$. Hai nghiệm của hai hệ là trùng nhau.

Như vậy, vấn đề đặt ra là, với điều kiện nào thì hệ đại số tuyến tính $Ax = b$ được thay bởi $\tilde{A}x = \tilde{b}$ mà nghiệm của hệ thay đổi không đáng kể. Hệ có tính chất như vậy được gọi là hệ ổn định tính, ngược lại hệ được gọi là không ổn định tính. Người ta đã chứng minh được rằng, sự ổn định của hệ đại số tuyến tính được đặc trưng bởi số điều kiện của ma trận A cho bởi công thức:

$$\text{cond}(A) = \|A\| \cdot \|A^{-1}\|.$$

Nói chung, $\text{cond}(A) \gg 1$. Khi $\text{cond}(A)$ càng gần với "1" thì hệ đại số tuyến tính càng ổn định tính.

Trong ba ví dụ trên thì các hệ trong các ví dụ 4.6, 4.7 không ổn định tính còn ở ví dụ 4.8 hệ là ổn định tính. Vấn đề ổn định tính là vấn đề khó, ở đây chỉ đưa ra để bạn đọc suy ngẫm. Trong thực tế thấy rằng: nếu các hệ số của hệ quan sát được là giá trị đúng thì ta có thể sử dụng phương pháp giải đúng. Còn nếu hệ số của hệ đó có được do quy tròn số thì nên áp dụng phương pháp lặp đơn.

§5. MA TRẬN NGHỊCH ĐẢO VÀ CHƯƠNG TRÌNH MATLAB

Khi giải hệ đại số tuyến tính (4.3), nếu $\det A \neq 0$ thì tồn tại A^{-1} và $r = A^{-1}b$ là nghiệm của hệ. Trong phần này ta trình bày một số phương pháp tìm ma trận nghịch đảo A^{-1} .

5.1 Bài toán

Cho ma trận vuông $A = [a_{ij}]_n$, tìm ma trận $B = [x_{ij}]_n$ sao cho

$$A \cdot B = B \cdot A = E, \quad (4.39)$$

trong đó E là ma trận đơn vị cùng cấp n .

Ma trận B thỏa mãn (4.39) được gọi là ma trận nghịch đảo của ma trận A , ký hiệu $B = A^{-1}$. Điều kiện cần và đủ để tồn tại A^{-1} là: ma trận vuông A không suy biến ($\det A \neq 0$).

5.2 Phương pháp tìm A^{-1} trong đại số tuyến tính

Công thức tìm A^{-1} được cho như sau:

$$A^{-1} = \frac{1}{\det A} \mathcal{A}^t, \quad (\det A \neq 0), \quad (4.40)$$

trong đó \mathcal{A}^t là ma trận chuyển vị của ma trận $\mathcal{A} = [A_{ij}]_n$ với A_{ij} là phần phu đại số của phần tử a_{ij} của ma trận A , tức là $A_{ij} = (-1)^{i+j} M_{ij}$, trong đó M_{ij} là định thức cấp $n - 1$ có được từ ma trận A khi bớt hàng i và cột j , $\forall i, j = \overline{1, n}$.

Ta thấy, khối lượng tính toán theo công thức (4.40) là quá lớn, cần khắc phục.

5.3 Phương pháp Gauss-Jordan

Giả sử $B = [x_{ij}]_n$. Từ (4.39) ta có:

$$\begin{bmatrix} a_{11} & \dots & a_{1j} & \dots & a_{1n} \\ a_{21} & \dots & a_{2j} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{i1} & \dots & a_{ij} & \dots & a_{in} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1} & \dots & a_{nj} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_{11} & \dots & x_{1j} & \dots & x_{1n} \\ x_{21} & \dots & x_{2j} & \dots & x_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{i1} & \dots & x_{ij} & \dots & x_{in} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{n1} & \dots & x_{nj} & \dots & x_{nn} \end{bmatrix} = \begin{bmatrix} 1 & \dots & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & \dots & 1 \end{bmatrix} \quad (4.41)$$

Đặt

$$X_j = (x_{1j}, x_{2j}, \dots, x_{nj})^t, \quad j = \overline{1, n}$$

là các vector cột của ma trận B và

$$E_j = (0, \dots, 0, \underbrace{1}_{(j)}, 0, \dots, 0)^t, \quad j = \overline{1, n}$$

là các cột của ma trận đơn vị E . Từ (4.41) ta được n hệ phương trình dạng ma trận:

$$AX_j = E_j, \quad j = \overline{1, n}. \quad (4.42)$$

Üng với mỗi $j = \overline{1, n}$, (4.42) là hệ đại số tuyến tính có cùng ma trận hệ số là A . Do giả thiết $\det A \neq 0$ nên n hệ của (4.42) đều tồn tại duy nhất

nghiệm là X_j ($j = \overline{1, n}$), có thể tìm được bằng phương pháp Gauss-Jordan đã xét trong §2. Lại do n hệ của (4.42) có cùng ma trận hệ số A nên ta giải đồng thời n hệ bằng cách lập ma trận mở rộng $\bar{A} = A|E$ và tiến hành các bước theo phương pháp Gauss-Jordan.

Chú ý 4.5. Quá trình giải bao gồm việc giải đồng thời n hệ phương trình, mỗi hệ có n ẩn (có cùng ma trận hệ số là A). Để được nghiệm là ma trận A^{-1} thì nên chọn phần tử giải qua mỗi bước là phần tử $a_{ii}^{(i-1)}$ trên đường chéo chính. Hơn thế nữa ta lại chọn $a_{ii}^{(i-1)} = 1$ (bằng cách chia dòng thứ i đó cho $a_{ii}^{(i-1)}$). Chẳng hạn ngay bước một, phần tử a_{11} của ma trận A có hai khả năng xảy ra:

1. $a_{11} = 0$ thì đổi chỗ hàng một cho hàng i nào đó có $a_{i1} \neq 0$ ($i > 1$) và đánh số lại ta được $a_{11} \neq 0$.
2. $a_{11} \neq 0$. Chia tất cả hàng đầu cho a_{11} và đánh số lại ta được $a_{11} = 1$. Ở các bước sau thì các phần tử $a_{ii}^{(i-1)}$ cũng được tiến hành tương tự.

Ví dụ 4.9. Tìm ma trận nghịch đảo của ma trận sau theo phương pháp Gauss-Jordan:

$$A = \begin{bmatrix} 50 & 107 & 36 \\ 25 & 54 & 20 \\ 31 & 66 & 21 \end{bmatrix}$$

Giải: Quá trình tính được mô tả trong bảng sau:

Bước	A	E	Chú thích
\bar{A}	50 107 36 25 54 20 31 66 21	1 0 0 0 1 0 0 0 1	
$\bar{A}^{(1)}$	1 2.14 0.72 0 0.50 2 0 -0.34 -1.32	0.02 0 0 -0.5 1 0 -0.62 0 1	Chia hàng đầu cho 50 Các phần tử tính theo sơ đồ (4.15)
$\bar{A}^{(2)}$	1 0 -7.84 0 1 4 0 0 0.04	2.16 -4.28 0 -1 2 0 -0.96 0.68 1	Tính theo sơ đồ (4.15) Chia hàng thứ hai của $\bar{A}^{(1)}$ cho 0.50 Tính theo sơ đồ (4.15)
$\bar{A}^{(3)}$	1 0 0 0 1 0 0 0 1	-186 129 196 95 -66 -100 -24 17 25	Tính theo sơ đồ (4.15) Chia hàng thứ ba của $\bar{A}^{(2)}$ cho 0.04

$$\text{Vậy } A^{-1} = \begin{bmatrix} -186 & 129 & 196 \\ 95 & -66 & -100 \\ -24 & 17 & 25 \end{bmatrix}$$

Chương trình MATLAB

```

function X=GJInv(A)
% GJInv computes the inverse matrix of a square matrix A
% by using Gauss-Jordan method.
% Input: A is a square matrix
% Output: X is the inverse of the matrix A
n=size(A,1);
E=eye(n);
X=zeros(size(A));
for j=1:n
    X(:,j)=gauss_jordan([A E(:,j)]);
end
end

```

5.4 Trường hợp ma trận đối xứng. Phương pháp Cholesky

Giả sử $A = [a_{ij}]_n$ là ma trận đối xứng ($a_{ij} = a_{ji} \forall i \neq j$). Phân tích ma trận A thành tích hai ma trận tam giác trên Q và tam giác dưới P (xem §2). Do A đối xứng nên ta có phân tích $A = PQ = Q^t Q$ theo công thức (4.20), từ đó ta suy ra

$$A^{-1} = (Q^t Q)^{-1} = Q^{-1}(Q^t)^{-1} = Q^{-1}(Q^{-1})^t. \quad (4.43)$$

Vậy, bài toán đặt ra là từ ma trận

$$Q = \begin{bmatrix} q_{11} & q_{12} & \dots & q_{1n} \\ 0 & q_{22} & \dots & q_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & q_{nn} \end{bmatrix},$$

hãy tìm ma trận nghịch đảo Q^{-1} .

Giả sử

$$Q^{-1} = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \dots & \alpha_{1n} \\ \alpha_{21} & \alpha_{22} & \dots & \alpha_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{n1} & \alpha_{n2} & \dots & \alpha_{nn} \end{bmatrix}.$$

Từ đồng nhất thức $Q \cdot Q^{-1} = E$, ta có:

$$\begin{bmatrix} q_{11} & q_{12} & \dots & q_{1n} \\ 0 & q_{22} & \dots & q_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & q_{nn} \end{bmatrix} \begin{bmatrix} \alpha_{11} & \alpha_{12} & \dots & \alpha_{1n} \\ \alpha_{21} & \alpha_{22} & \dots & \alpha_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{n1} & \alpha_{n2} & \dots & \alpha_{nn} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix} \quad (4.44)$$

Sử dụng tính chất bằng nhau của hai ma trận, từ (4.44) ta suy ra công thức tính Q^{-1} như sau:

$$\begin{aligned}\alpha_{ii} &= \frac{1}{q_{ii}}, \quad i = n, n-1, \dots, 2, 1; \\ \alpha_{ij} &= 0 \quad \forall i > j; \\ \alpha_{n-1,n} &= -\frac{1}{q_{n-1,n-1}} q_{n-1,n} \alpha_{nn}; \\ \alpha_{n-2,k} &= -\frac{1}{q_{n-2,n-2}} \sum_{j=n-1}^n q_{n-2,j} \alpha_{jk}, \quad k = n-1, n, \quad (j \leq k); \\ \alpha_{2,k} &= -\frac{1}{q_{22}} \sum_{j=3}^n q_{2j} \alpha_{jk}, \quad k = 3, 4, \dots, n \quad (j \leq k); \\ \alpha_{1,k} &= -\frac{1}{q_{11}} \sum_{j=2}^n q_{1j} \alpha_{jk}, \quad k = 2, 3, \dots, n \quad (j \leq k).\end{aligned}\tag{4.45}$$

Như vậy, Q^{-1} là ma trận tam giác trên. Các phần tử của đường chéo chính được tính theo công thức $\alpha_{ii} = \frac{1}{q_{ii}}$, $i = \overline{1, n}$, còn các phần tử phía trên đường chéo chính được tính theo (4.45) từ dưới lên, ta có Q^{-1} và theo (4.43) ta có A^{-1} .

Ví dụ 4.10. Tìm ma trận nghịch đảo của ma trận

$$A = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 5 & 1 \\ 1 & 1 & 3 \end{bmatrix}.$$

Giải:

Từ ví dụ 4.2 ta đã có phân tích $A = Q^t Q$ với

$$Q = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}.$$

Tìm Q^{-1} theo công thức (4.44) ta thu được:

$$Q^{-1} = \begin{bmatrix} 1 & -2 & -3 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

Khi đó:

$$A^{-1} = Q^{-1}(Q^{-1})^t = \begin{bmatrix} 14 & -5 & -3 \\ -5 & 2 & 1 \\ -3 & 1 & 1 \end{bmatrix}$$

Chương trình MATLAB

```

function B=chodecinv(A)
% Find the inverse of a matrix
% using the Cholesky decomposition
% Input: A is (often) positive definite and symmetric
% Output: B is the inverse matrix of A

n = size(A,1);
Q = chodec(A); % Find the lower matrix Q such that A=Q^tQ
P = zeros(n,n); % P is the inverse matrix of Q

for i=1:n
    P(i,i)=1/Q(i,i);
end
for i=n-1:-1:1
    for j=i+1:n
        P(i,j)=-Q(i,i+1:n)*P(i+1:n,j)/Q(i,i);
    end
end
B=P*P';

```

§6. TRỊ RIÊNG, VECTOR RIÊNG CỦA MA TRẬN VÀ CHƯƠNG TRÌNH MATLAB

6.1 Khái niệm về trị riêng và vector riêng

Cho ma trận $A = [a_{ij}]_n$ vuông cấp n . Hãy tìm vector $X \in \mathbb{R}^n \setminus \{\theta\}$ và số $\lambda \in \mathbb{C}$ thỏa mãn điều kiện

$$AX = \lambda X, \quad (\lambda - \text{tham số}). \quad (4.46)$$

Số λ thỏa mãn (4.46) được gọi là trị riêng, còn vector $X \neq \theta$ gọi là vector riêng của ma trận A ứng với trị riêng λ . Ý nghĩa của bài toán trên là vector $X \neq \theta$, nói chung các vector AX và X không tỷ lệ với nhau (không đồng phuong với nhau), nhưng nếu có số λ thỏa mãn (4.46) thì chúng tỷ lệ với nhau theo hệ số tỷ lệ là λ . Trong giáo trình đại số ta đã biết, từ phương trình (4.46) ta có:

$$(A - \lambda E) X = 0, \quad (4.47)$$

trong đó E là ma trận đơn vị cùng cấp với ma trận A . Để tồn tại $X \neq \theta$ thỏa mãn (4.47) thì điều kiện là:

$$\det(A - \lambda E) = |A - \lambda E| = 0 \quad (4.48)$$

Đa thức $P(\lambda) = |A - \lambda E|$ gọi là đa thức đặc trưng. Giải phương trình đa thức (4.48) ta được các giá trị riêng của ma trận A . Ứng với mỗi λ , giải (4.47) ta được vector riêng tương ứng. Tuy nhiên, số lượng phép tính để tìm đa thức đặc trưng là rất lớn khi n tăng. Để giảm khối lượng tính toán, ta xét một số phương pháp sẽ được trình bày sau đây.

6.2 Trị riêng và vector riêng của các ma trận đồng dạng

Cho hai ma trận vuông cùng cấp A và B .

Định nghĩa 4.4. Ta nói hai ma trận A và B đồng dạng với nhau và ký hiệu là $A \sim B$, nếu tồn tại ma trận T không suy biến ($\det T \neq 0$) sao cho

$$B = T^{-1}AT.$$

Từ giáo trình đại số tuyến tính, ta đã biết:

1. Nếu $X \neq 0$ là một vector riêng của ma trận A ứng với trị riêng λ thì mọi vector dạng cX ($c = \text{const}$) cũng là vector riêng ứng với giá trị riêng λ đó.
2. Nếu $A \sim B$ thì $B \sim A$.
3. Nếu $A \sim B; B \sim C$ thì $A \sim C$.
4. Nếu $A \sim B$ thì A và B có cùng giá trị riêng và do đó chúng có cùng đa thức đặc trưng.

Như vậy, để tìm giá trị riêng của ma trận A , ta tìm ma trận đồng dạng với nó mà đa thức đặc trưng có thể tìm được một cách dễ dàng.

6.3 Phương pháp Danhilepski

Nội dung của phương pháp này là biến đổi ma trận về dạng đồng dạng với ma trận P có đa thức đặc trưng dễ tìm hơn.

6.3.1. Đa thức đặc trưng của ma trận dạng "Phờ-rô-bơ-niuyt" (dạng P)

Ma trận "Phờ-rô-bơ-niuyt" là ma trận có dạng:

$$P = \begin{bmatrix} p_1 & p_2 & p_3 & \dots & p_{n-1} & p_n \\ 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & \dots & 1 & 0 \end{bmatrix}$$

Sử dụng khai triển theo cột và quy nạp ta thu được đa thức đặc trưng của ma trận trên:

$$\det(P - \lambda E) = \det \begin{vmatrix} p_1 - \lambda & p_2 & p_3 & \dots & p_{n-1} & p_n \\ 1 & -\lambda & 0 & \dots & 0 & 0 \\ 0 & 1 & -\lambda & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -\lambda & 0 \\ 0 & 0 & 0 & \dots & 1 & -\lambda \end{vmatrix} = (-1)^n(\lambda^n - p_1\lambda^{n-1} - p_2\lambda^{n-2} - \dots - p_{n-1}\lambda - p_n). \quad (4.49)$$

6.3.2. Quá trình biến đổi ma trận A về dạng P

- *Bước 1.*

Đặt $A_n^{(1)} = A_n = [a_{ij}^{(1)}] = [a_{ij}]$, $\forall i, j = \overline{1, n}$. Giả thiết $a_{n,n-1}^{(1)} \neq 0$. Chọn

$$M_1 = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1}^{(1)} & a_{n2}^{(1)} & \dots & a_{n,n-1}^{(1)} & a_{nn}^{(1)} \\ 0 & 0 & \dots & 0 & 1 \end{bmatrix}$$

là ma trận thu được từ ma trận đơn vị E bằng cách thay hàng thứ $n - 1$ bởi hàng thứ n của ma trận $A_n^{(1)}$. Do $\det M_1 = a_{n-1,n}^{(1)} \neq 0$ nên tồn tại M_1^{-1} và

$$M_1^{-1} = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ -\frac{a_{n1}^{(1)}}{a_{n,n-1}^{(1)}} & -\frac{a_{n2}^{(1)}}{a_{n,n-1}^{(1)}} & \dots & \frac{1}{a_{n,n-1}^{(1)}} & -\frac{a_{nn}^{(1)}}{a_{n,n-1}^{(1)}} \\ 0 & 0 & \dots & 0 & 1 \end{bmatrix}$$

Tính $A_n^{(2)} = M_1 A_n^{(1)} M_1^{-1}$ ta được:

$$A_n^{(2)} = \begin{bmatrix} a_{11}^{(2)} & a_{12}^{(2)} & \dots & a_{1,n-2}^{(2)} & a_{1,n-1}^{(2)} & a_{1,n}^{(2)} \\ a_{21}^{(2)} & a_{22}^{(2)} & \dots & a_{2,n-2}^{(2)} & a_{2,n-1}^{(2)} & a_{2,n}^{(2)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n-1,1}^{(2)} & a_{n-1,2}^{(2)} & \dots & a_{n-1,n-2}^{(2)} & a_{n-1,n-1}^{(2)} & a_{n-1,n}^{(2)} \\ 0 & 0 & \dots & 0 & 1 & 0 \end{bmatrix} \quad (4.50)$$

- Bước 2. Giả thiết $a_{n-1,n-2}^{(2)} \neq 0$, chọn

$$M_2 = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n-1,1}^{(2)} & a_{n-1,2}^{(2)} & \dots & a_{n-1,n-2}^{(2)} & a_{n-1,n-1}^{(2)} & a_{n-1,n}^{(2)} \\ 0 & 0 & \dots & 0 & 1 & 0 \\ 0 & 0 & \dots & 0 & 0 & 1 \end{bmatrix}$$

là ma trận thu được từ ma trận đơn vị E bằng cách thay hàng thứ $n-2$ bởi hàng thứ $n-1$ của ma trận $A_n^{(2)}$. Do $\det M_2 = a_{n-1,n-2}^{(2)} \neq 0$ nên tồn tại M_2^{-1} và

$$M_2^{-1} = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -\frac{a_{n-1,1}^{(2)}}{a_{n-1,n-2}^{(2)}} & -\frac{a_{n-1,2}^{(2)}}{a_{n-1,n-2}^{(2)}} & \dots & \frac{1}{a_{n-1,n-2}^{(2)}} & -\frac{a_{n-1,n-1}^{(2)}}{a_{n-1,n-2}^{(2)}} & -\frac{a_{n-1,n}^{(2)}}{a_{n-1,n-2}^{(2)}} \\ 0 & 0 & \dots & 0 & 1 & 0 \\ 0 & 0 & \dots & 0 & 0 & 1 \end{bmatrix}$$

Tiếp theo, tính $A_n^{(3)} = M_2 A_n^{(2)} M_2^{-1}$ ta được:

$$A_n^{(3)} = \begin{bmatrix} a_{11}^{(3)} & a_{12}^{(3)} & \dots & a_{1,n-2}^{(3)} & a_{1,n-1}^{(3)} & a_{1,n}^{(3)} \\ a_{21}^{(3)} & a_{22}^{(3)} & \dots & a_{2,n-2}^{(3)} & a_{2,n-1}^{(3)} & a_{2,n}^{(3)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n-2,1}^{(3)} & a_{n-2,2}^{(3)} & \dots & a_{n-2,n-2}^{(3)} & a_{n-2,n-1}^{(3)} & a_{n-2,n}^{(3)} \\ 0 & 0 & \dots & 1 & 0 & 0 \\ 0 & 0 & \dots & 0 & 1 & 0 \end{bmatrix} \quad (4.51)$$

- Quá trình trên được thực hiện và sau $n - 1$ bước ta thu được ma trận dạng P :

$$\begin{aligned} P &= A_n^{(n)} = M_{n-1}A_n^{(n-1)}M_{n-1}^{-1} = M_{n-1}M_{n-2}A_n^{(n-2)}M_{n-2}^{-1}M_{n-1}^{-1} = \\ &= (M_{n-1}M_{n-2} \dots M_2M_1)A_n^{(1)}(M_1^{-1}M_2^{-1} \dots M_{n-2}^{-1}M_{n-1}^{-1}). \end{aligned} \quad (4.52)$$

Đặt

$$M_{n-1}M_{n-2} \dots M_2M_1 = M,$$

thì ta có:

$$M^{-1} = (M_{n-1}M_{n-2} \dots M_2M_1)^{-1} = M_1^{-1}M_2^{-1} \dots M_{n-2}^{-1}M_{n-1}^{-1}.$$

Thay vào (4.52) ta thu được:

$$P = A_n^{(n)} = MA_n^{(1)}M^{-1} = MA_nM^{-1}, \quad (4.53)$$

tức là $P \sim A_n$.

Số lượng phép tính để biến đổi từ A_n đến P có $N = n^3 - n^2$ phép tính nhân và chia.

Trong các bước tính trên, ta luôn giả thiết $a_{i,i-1}^{n-i+1} \neq 0$, nếu $a_{i,i-1}^{n-i+1} = 0$ thì quá trình trên không thực hiện được. Chẳng hạn sau một bước, ta được $A_n^{(2)} \sim A_n^{(1)}$ và có $a_{n-1,n-2}^{(2)} = 0$. Trường hợp này có hai khả năng xảy ra.

- Mọi phần tử $a_{n-1,k}^{(2)} = 0 \forall k < n-2$ (mọi phần tử của hàng $n-1$, đứng trước $a_{n-1,n-2}^{(2)}$ đều bằng "0"). Khi đó đa thức đặc trưng của $A_n^{(2)}$ sẽ là:

$$\begin{aligned} \det(A_n^{(2)} - \lambda E) &= \\ &= \begin{vmatrix} a_{11}^{(2)} - \lambda & a_{12}^{(2)} & \dots & a_{1,n-2}^{(2)} & a_{1,n-1}^{(2)} & a_{1,n}^{(2)} \\ a_{21}^{(2)} & a_{22}^{(2)} - \lambda & \dots & a_{2,n-2}^{(2)} & a_{2,n-1}^{(2)} & a_{2,n}^{(2)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n-2,1}^{(2)} & a_{n-2,2}^{(2)} & \dots & a_{n-2,n-2}^{(2)} - \lambda & a_{n-2,n-1}^{(2)} & a_{n-2,n}^{(2)} \\ 0 & 0 & \dots & 0 & a_{n-1,n-1}^{(2)} - \lambda & a_{n-1,n}^{(2)} \\ 0 & 0 & \dots & 0 & 1 & -\lambda \end{vmatrix} \\ &= [-\lambda(a_{n-1,n-1}^{(2)} - \lambda) - a_{n-1,n}^{(2)}] \det(A_{n-2}^{(2)} - \lambda E), \end{aligned} \quad (4.54)$$

trong đó $A_{n-2}^{(2)} = [a_{ij}^{(2)}] \forall i, j = \overline{1, n-2}$ và E là ma trận đơn vị cấp $n-2$, nghĩa là chỉ cần tìm đa thức đặc trưng của ma trận cấp $n-2$.

2. Tồn tại phần tử $a_{n-1,k}^{(2)} \neq 0$, $k < n - 2$. Gọi C là ma trận thu được từ ma trận đơn vị cấp n bằng cách đổi chỗ cột thứ k và cột thứ $n - 2$ cho nhau:

$$C = \begin{bmatrix} 1 & \dots & 0 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & \dots & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 1 & \dots & 0 & 0 & 0 \\ 0 & \dots & 0 & \dots & 0 & 1 & 0 \\ 0 & \dots & 0 & \dots & 0 & 0 & 1 \end{bmatrix}$$

Để thấy $\det C = \pm 1 \Rightarrow C^{-1} = C$. Tính $\overline{A}_n^{(2)} = CA_n^{(2)}C$ thì $A_n^{(2)}C$ chính là kết quả của việc đổi chỗ hai cột thứ k và thứ $n - 2$ cho nhau. Còn $C^{-1}A_n^{(2)}C = CA_n^{(2)}C$ là phép đổi hàng thứ k và thứ $n - 2$ cho nhau. Với $\overline{A}_n^{(2)}$ có phần tử $\bar{a}_{n-1,n-2}^{(2)} \neq 0$. Quá trình tiếp tục như trên ta được đa thức đặc trưng dạng (4.49). Giải phương trình đa thức (xem chương 2) ta thu được các trị riêng λ .

6.3.3. Vector riêng của ma trận A_n ứng với trị riêng λ

Giả sử Y là vector riêng ứng với trị riêng λ của ma trận P , tức là $PY = \lambda Y$. Từ (4.53) ta có $P \sim A_n$, do đó λ cũng là trị riêng của ma trận A_n , tức là

$$MA_nM^{-1}Y = \lambda Y \Rightarrow A_nM^{-1}Y = \lambda M^{-1}Y$$

Đặt $X = M^{-1}Y = (M_{n-1}M_{n-2}\dots M_2M_1)^{-1}Y$ hay

$$X = (M_1^{-1}M_2^{-1}\dots M_{n-1}^{-1})Y \quad (4.55)$$

ta được:

$$A_nX = \lambda X.$$

Vậy X xác định theo (4.55) là vector riêng ứng với trị riêng λ của ma trận A_n .

Ta tìm vector riêng Y ứng với trị riêng λ của ma trận P . Từ $PY = \lambda Y$ với

$Y = (y_1, y_2, \dots, y_n)^t$ ta có:

$$\begin{bmatrix} p_1 & p_2 & \cdots & p_{n-1} & p_n \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \lambda \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

hay

$$\begin{aligned} p_1 y_1 + p_2 y_2 + \cdots + p_n y_n &= \lambda y_1 \\ y_1 &= \lambda y_2 \\ y_2 &= \lambda y_3 \\ &\vdots \\ y_{n-2} &= \lambda y_{n-1} \\ y_{n-1} &= \lambda y_n. \end{aligned} \tag{4.56}$$

Do các vector riêng sai khác nhau hằng số nhân nên ta có thể chọn $y_n = 1$. Từ (4.56) ta có:

$$Y = (\lambda^{n-1}, \lambda^{n-2}, \dots, \lambda, 1)^t.$$

Theo (4.55) ta thu được vector riêng X ứng với trị riêng λ của ma trận A .

Ví dụ 4.11. Tìm trị riêng và vector riêng của ma trận $A = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 3 & 1 \\ 0 & 1 & 2 \end{bmatrix}$.

Giải:

- *Bước 1:* Đặt $A = A^{(1)}$. Do $a_{32}^{(1)} = a_{32} = 1 \neq 0$ nên chọn

$$M_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \implies M_1^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}.$$

Từ đó ta có:

$$A^{(2)} = M_1 A^{(1)} M_1^{-1} = \begin{bmatrix} 2 & 1 & -2 \\ 1 & 5 & -5 \\ 0 & 1 & 0 \end{bmatrix}.$$

- Bước 2: Do $a_{12}^{(2)} = 1 \neq 0$ nên ta chọn

$$M_2 = \begin{bmatrix} 1 & 5 & -5 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \implies M_2^{-1} = \begin{bmatrix} 1 & -5 & 5 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Từ đó ta có:

$$A^{(3)} = M_2 A^{(2)} M_2^{-1} = \begin{bmatrix} 7 & -14 & 8 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

Vậy

$$\begin{aligned} P_3(\lambda) &= (-1)^3 (\lambda^3 - 7\lambda^2 + 14\lambda - 8) = 0 \\ \implies \lambda_1 &= 4; \quad \lambda_2 = 2; \quad \text{và} \quad \lambda_3 = 1. \end{aligned}$$

- Üng với giá trị riêng $\lambda_1 = 4$, theo công thức (4.55) ta có vector riêng tương ứng:

$$X_1 = (M_1^{-1} M_2^{-1}) Y_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -5 & 5 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4^2 \\ 4 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}.$$

- Với $\lambda_2 = 2$ thì

$$X_2 = (M_1^{-1} M_2^{-1}) Y_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -5 & 5 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}.$$

- Với $\lambda_3 = 1$ thì

$$X_3 = (M_1^{-1} M_2^{-1}) Y_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -5 & 5 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}.$$

Chương trình MATLAB

Bạn đọc hãy dựa theo ví dụ để lập chương trình MatLab cho phần này.

6.4 Phương pháp Krulôp A.N.

6.4.1. Nội dung phương pháp

Dựa vào đồng nhất thức Hamilton - Kelly. Giả sử ma trận A có đa thức đặc trưng

$$P(\lambda) = \det(A - \lambda E) = (-1)^n [\lambda^n + b_1 \lambda^{n-1} + \cdots + b_{n-1} \lambda + b_n], \quad (4.57)$$

trong đó b_i ($i = \overline{1, n}$) là các hệ số cần tìm.

Ta đã biết (4.57) là đa thức đặc trưng của ma trận A thì ma trận A đồng thời thỏa mãn phương trình ma trận:

$$A^n + b_1 A^{n-1} + \cdots + b_{n-1} A + b_n E = 0 \quad (4.58)$$

trong đó E là ma trận đơn vị cùng cấp với A . Phương trình (4.58) được gọi là đồng nhất thức Hamilton - Kelly.

Chọn vector bất kỳ $Y^{(0)} = (y_1^{(0)}, y_2^{(0)}, \dots, y_n^{(0)})^t$ và nhân (về bên phải) hai vế của (4.58) với $Y^{(0)}$ ta có:

$$A^n Y^{(0)} + b_1 A^{n-1} Y^{(0)} + \cdots + b_{n-1} A Y^{(0)} + b_n Y^{(0)} = 0. \quad (4.59)$$

Đặt

$$A^k Y^{(0)} = Y^{(k)}, \quad k = \overline{1, n}, \quad (4.60)$$

thay vào (4.59) ta thu được:

$$Y^{(n)} + b_1 Y^{(n-1)} + \cdots + b_{n-1} Y^{(1)} + b_n Y^{(0)} = 0$$

hay:

$$b_1 Y^{(n-1)} + \cdots + b_{n-1} Y^{(1)} + b_n Y^{(0)} = -Y^{(n)}. \quad (4.61)$$

Đây là hệ đại số tuyến tính, n phương trình, n ẩn b_i ($i = \overline{1, n}$) có dạng ma trận:

$$\begin{bmatrix} y_1^{(n-1)} & y_1^{(n-2)} & \cdots & y_1^{(1)} & y_1^{(0)} \\ y_2^{(n-1)} & y_2^{(n-2)} & \cdots & y_2^{(1)} & y_2^{(0)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ y_n^{(n-1)} & y_n^{(n-2)} & \cdots & y_n^{(1)} & y_n^{(0)} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = - \begin{bmatrix} y_1^{(n)} \\ y_2^{(n)} \\ \vdots \\ y_n^{(n)} \end{bmatrix}. \quad (4.62)$$

Giải hệ trên ta thu được b_1, b_2, \dots, b_n là hệ số của đa thức đặc trưng của ma trận A (dạng (4.57)).

6.4.2. Thuật toán

Chọn vector $Y^{(0)} \neq \theta$ bất kỳ và tính dãy $\{Y^{(n)}\}$ theo công thức (4.60):

$$Y^{(1)} = AY^{(0)}, \quad Y^{(2)} = AY^{(1)} = A^2Y^{(0)}, \dots$$

$$Y^{(n)} = AY^{(n-1)} = \dots = A^nY^{(0)}.$$

Quá trình tính được mô tả trên bảng tính như sau (bảng 4.3):

	Y^0	$Y^{(1)}$	$Y^{(2)}$	\dots	$Y^{(n)}$
A	$Y^{(0)}$	$AY^{(0)}$	$A^2Y^{(0)}$	\dots	$A^nY^{(0)}$

Bảng 4.3

Từ đó ta có hệ (4.61), giải hệ này ta thu được b_i , $i = \overline{1, n}$.

Chú ý 4.6. Khi giải hệ (4.61), nếu hệ có nghiệm duy nhất thì chúng chính là hệ số của đa thức đặc trưng của ma trận A . Trong trường hợp hệ không có nghiệm duy nhất (do $Y^{(0)}$ được chọn bất kỳ), ta chọn lại $Y^{(0)}$ sao cho hệ trên có nghiệm duy nhất.

6.4.3. Vector riêng ứng với trị riêng λ

Giả sử giải phương trình đa thức (4.57) ta được các trị riêng λ_i ($i = \overline{1, n}$) nào đó. Khi đó, vector riêng $X^{(i)}$ ứng với trị riêng λ_i được tìm theo công thức:

$$X^{(i)} = Y^{(n-1)} + q_{1i}Y^{(n-2)} + q_{2i}Y^{(n-3)} + \dots + q_{n-1,i}Y^{(0)}, \quad (i = \overline{1, n}) \quad (4.63)$$

trong đó q_{ji} , $j = \overline{1, n-1}$, $i = \overline{1, n}$ chính là hệ số của đa thức thương khi chia $P(\lambda)$ cho $(\lambda - \lambda_i)$, ($i = \overline{1, n}$), có thể dễ dàng tìm được theo sơ đồ Horner.

Chú ý 4.7. Vector riêng $X^{(i)}$ được tính theo công thức (4.63) chỉ áp dụng khi trị riêng λ_i là số thực. Ở đây, ta không xét trường hợp λ_i là số phức.

Ví dụ 4.12. Tìm đa thức đặc trưng của ma trận $A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 2 & 3 \\ 3 & 2 & 1 & 2 \\ 4 & 3 & 2 & 1 \end{bmatrix}$.

Giải:

Theo bảng 4.3 ta có:

A	Y^0	$Y^{(1)} = AY^{(0)}$	$Y^{(2)} = AY^{(1)}$	$Y^{(3)} = AY^{(2)}$	$Y^{(4)} = AY^{(3)}$
1 2 3 4	1	1	30	208	2108
2 1 2 3	0	2	22	178	1704
3 2 1 2	0	3	18	192	1656
4 3 2 1	0	4	20	242	1992

Từ bảng trên ta được hệ phương trình

$$\begin{bmatrix} 208 & 30 & 1 & 1 \\ 178 & 22 & 2 & 0 \\ 192 & 18 & 3 & 0 \\ 242 & 20 & 4 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} = - \begin{bmatrix} 2108 \\ 1704 \\ 1656 \\ 1992 \end{bmatrix}$$

Giải hệ trên ta thu được $b_1 = -4$, $b_2 = -40$, $b_3 = -56$, $b_4 = -20$ và từ đó ta có đa thức đặc trưng

$$p(\lambda) = \det(A - \lambda E) = \lambda^4 - 4\lambda^3 - 40\lambda^2 - 56\lambda - 20.$$

Chương trình MATLAB

Bạn đọc hãy dựa theo ví dụ để lập chương trình MatLab cho phần này.

§7. TÌM GẦN ĐÚNG TRỊ RIENG VÀ VECTOR RIENG

Hai phương pháp đã nêu trong §6 là các phương pháp tìm đúng. Song chúng chỉ là gần đúng do sự quy tròn số sinh ra (sai số tính toán). Trong tiết này, ta nêu một phương pháp gần đúng để tìm trị riêng trội (thực đơn hoặc bội) mà không cần tìm đa thức đặc trưng.

7.1 Nội dung phương pháp tìm trị riêng trội

Giả sử ma trận A có đủ n trị riêng thực đơn, bội, hoặc phức (có cặp phức liên hợp) và có đủ n vector riêng $\{X_k\}$, $k = \overline{1, n}$ độc lập tuyến tính. Chọn vector Y là tổ hợp tuyến tính bất kỳ của chúng:

$$Y = \sum_{k=1}^n c_k X_k, \quad c_1 \neq 0. \quad (4.64)$$

Do $A X_k = \lambda_k X_k$ nên

$$\begin{aligned} AY &= \sum_{k=1}^n c_k A X_k = \sum_{k=1}^n c_k \lambda_k X_k; \\ A^2 Y &= A(AY) = \sum_{k=1}^n c_k \lambda_k A X_k = \sum_{k=1}^n c_k \lambda_k^2 X_k; \\ \dots & \\ A^m Y &= A(A^{m-1} Y) = \sum_{k=1}^n c_k \lambda_k^m X_k; \\ \dots & \end{aligned} \quad (4.65)$$

Ta đi tìm trị riêng thực trội, tức là trị riêng có trị tuyệt đối lớn nhất, có thể xảy ra các trường hợp như dưới đây.

7.1.1. Trường hợp trị riêng thực trội bội một

Giả sử trị riêng của ma trận A thỏa mãn điều kiện

$$|\lambda_1| \geq |\lambda_2| \geq |\lambda_3| \geq \dots \geq |\lambda_n|. \quad (4.66)$$

Từ hệ (4.65) (với $c_1 \neq 0$) ta có:

$$A^m Y = c_1 \lambda_1^m X_1 + \sum_{k=2}^n c_k \lambda_k^m X_k = \lambda_1^m \left[c_1 X_1 + \sum_{k=2}^n c_k \left(\frac{\lambda_k}{\lambda_1} \right)^m X_k \right].$$

Do giả thiết (4.66), khi $m \rightarrow \infty$ thì $\left(\frac{\lambda_k}{\lambda_1} \right)^m \rightarrow 0$ ($k = 2, n$) và khi đó ta có:

$$A^m Y \rightarrow \lambda_1^m c_1 X_1 \quad (m \rightarrow \infty),$$

hay khi m đủ lớn thì $A^m Y \approx \lambda_1^m c_1 X_1$, đồng thời

$$\begin{aligned} A^{m+1} Y &\approx \lambda_1^{m+1} c_1 X_1 \\ &= \lambda_1 (\lambda_1^m c_1 X_1) \approx \lambda_1 A^m Y. \end{aligned} \quad (4.67)$$

Từ đó ta có:

$$\lambda_1 \approx \frac{(A^{m+1} Y)_j}{(A^m Y)_j}, \quad j = \overline{1, n}. \quad (4.68)$$

Từ (4.67) ta lại có:

$$A(A^m Y) \approx \lambda_1 A^m Y.$$

Điều này chứng tỏ λ_1 là trị riêng của ma trận A , vector riêng tương ứng là $X_1 = A^m Y$ hoặc $X_1 = A^{m+1} Y$.

Từ (4.68) ta thấy, với vector Y bất kỳ có $c_1 \neq 0$, ta tính $AY, A^2Y, \dots, A^mY, A^{m+1}Y$. Khi hai phép tính liên tiếp có xu hướng tỷ lệ với nhau thì m được gọi là đủ lớn.

7.1.2. Trường hợp trị riêng thực trội bởi r

Trong trường hợp này, giả sử

$$\lambda_1 = \lambda_2 = \dots = \lambda_r;$$

$$|\lambda_r| > |\lambda_{r+1}| \geq \dots \geq |\lambda_n|. \quad (4.69)$$

Từ (4.65) ta có:

$$\begin{aligned} A^m Y &= \lambda_1^m \sum_{k=1}^r c_k X_k + \sum_{j=r+1}^n c_j \lambda_j^m X_j \\ &= \lambda_1^m \left[\sum_{k=1}^r c_k X_k + \sum_{j=r+1}^n c_j \left(\frac{\lambda_j}{\lambda_1} \right)^m X_j \right], \quad c_k \neq 0, \quad k = \overline{1, n}. \end{aligned}$$

Bằng cách lập luân như trên, với m đủ lớn thì

$$A^{m+1} Y \approx \lambda_1 A^m Y$$

hay $A(A^m Y) \approx \lambda_1 (A^m Y)$. Từ đó suy ra

$$\lambda_1 \approx \frac{(A^{m+1} Y)_j}{(A^m Y)_j}, \quad \forall j = \overline{1, n}. \quad (4.70)$$

Như vậy trong quá trình tính $A^m Y$ và $A^{m+1} Y$, nếu thấy tỷ số (4.70) xấp xỉ bằng nhau, nghĩa là m đủ lớn và λ_1 xác định theo công thức (4.70) trùng với (4.68). Điều này có nghĩa là tỷ số (4.68) hoặc (4.70) chỉ xác định được λ_1 thực trội mà không biết được λ_1 đơn hay bội.

Vector riêng ứng với trị riêng λ_1 , như trên ta có thể lấy là $X_1 \approx A^m Y$ hoặc $X_1 \approx A^{m+1} Y$; nhưng nó chỉ là một vector riêng trong số r vector riêng độc lập tuyến tính ứng với trị riêng λ_1 .

7.1.3. Trường hợp trị riêng trội (thực đơn) đối dấu nhau

Xét trường hợp

$$\lambda_1 = -\lambda_2; \quad (4.71)$$

$$|\lambda_1| = |\lambda_2| > |\lambda_3| \geq \dots \geq |\lambda_n|. \quad (4.72)$$

Trong trường hợp này ta có:

$$\begin{aligned}
 AY &= \lambda_1(c_1X_1 - c_2X_2) + \sum_{j=3}^n c_j\lambda_j X_j \quad (c_1, c_2 \neq 0); \\
 A^2Y &= A(AY) = \lambda_1(c_1AX_1 - c_2AX_2) + \sum_{j=3}^n c_j\lambda_j AX_j \\
 &= \lambda_1^2(c_1X_1 - c_2X_2) + \sum_{j=3}^n c_j\lambda_j^2 X_j; \\
 A^{2k-1}Y &= \lambda_1^{2k-1}(c_1X_1 - c_2X_2) + \sum_{j=3}^n c_j\lambda_j^{2k-1} X_j; \\
 A^{2k}Y &= \lambda_1^{2k}(c_1X_1 + c_2X_2) + \sum_{j=3}^n c_j\lambda_j^{2k} X_j.
 \end{aligned} \tag{4.73}$$

Lập luận tương tự như trên, khi k đủ lớn ta thu được

$$\begin{aligned}
 A^{2k-1}Y &\approx \lambda_1^{2k-1}(c_1X_1 - c_2X_2); \\
 A^{2k}Y &\approx \lambda_1^{2k}(c_1X_1 + c_2X_2).
 \end{aligned}$$

Từ đó ta có

$$\begin{aligned}
 A^{2k+2}Y &\approx \lambda_1^{2k+2}(c_1X_1 + c_2X_2) = \lambda_1^2\lambda_1^{2k}(c_1X_1 + c_2X_2) \\
 &\approx \lambda_1^2 A^{2k}Y.
 \end{aligned} \tag{4.74}$$

Từ (4.74) ta suy ra

$$\lambda_1^2 \approx \frac{(A^{2k+2}Y)_j}{(A^{2k}Y)_j}, \quad j = \overline{1, n}. \tag{4.75}$$

Trong công thức (4.75) cũng có thể lấy:

$$\lambda_1^2 \approx \frac{(A^{2k+1}Y)_j}{(A^{2k-1}Y)_j}, \quad j = \overline{1, n}. \tag{4.76}$$

Như vậy, trong quá trình tính, với hai lũy thừa liên tiếp nhau, các tỷ số nói chung không có xu hướng tỷ lệ với nhau mà hai lũy thừa cùng chẵn (hoặc cùng lẻ) tỷ lệ với nhau thì ta xác định được λ_1^2 theo công thức (4.75) hoặc (4.76). Từ đó ta thu được $\pm\lambda_1$.

Các vector riêng X_1, X_2 ứng với trị riêng $\pm\lambda_1$ được xác định như sau:

Từ các xấp xỉ

$$\begin{aligned}
 A^{2k-1}Y &\approx \lambda_1^{2k-1}(C_1X_1 - C_2X_2); \\
 A^{2k}Y &\approx \lambda_1^{2k}(C_1X_1 + C_2X_2),
 \end{aligned}$$

suy ra

$$A^{2k}Y + \lambda_1 A^{2k-1}Y \approx \lambda_1^{2k} 2C_1 X_1. \quad (4.77)$$

Mặt khác, ta lại có

$$A(A^{2k}Y + \lambda_1 A^{2k-1}Y) \approx \lambda_1^{2k} 2C_1 A X_1,$$

hay

$$\begin{aligned} A(A^{2k}Y + \lambda_1 A^{2k-1}Y) &\approx \lambda_1^{2k} 2C_1 A X_1 = \lambda_1(\lambda_1^{2k} 2C_1 X_1) \\ &\approx \lambda_1(A^{2k}Y + \lambda_1 A^{2k-1}Y). \end{aligned} \quad (4.78)$$

Trong (4.78) chọn

$$X_1 \approx A^{2k}Y + \lambda_1 A^{2k-1}Y \quad (4.79)$$

thì ta có $A X_1 \approx \lambda_1 X_1$; nghĩa là X_1 tính theo công thức (4.79) là vector riêng ứng với trị riêng λ_1 .

Hoàn toàn tương tự ta cũng có vector riêng X_2 ứng với trị riêng $\lambda_2 = -\lambda_1$ được tính bởi công thức:

$$X_2 \approx A^{2k}Y - \lambda_1 A^{2k-1}Y. \quad (4.80)$$

Chú ý 4.8. 1. Trong quá trình biến đổi, nếu các tỷ số dang (4.68), (4.70), (4.75) không xảy ra thì chắc chắn trị riêng trội có cặp nghiệm phức liên hợp. Ta không xét trường hợp này ở đây.

2. Sau khi tìm được trị riêng trội, trị riêng trội kế tiếp có thể tìm được bằng phương pháp xuống thang, ta cũng không xét ở đây.

Các vấn đề của chú ý trên có thể xem trong [2, 5].

Ví dụ 4.13. Tìm trị riêng trội của ma trận $A = \begin{bmatrix} 2 & 3 & 2 \\ 4 & 3 & 5 \\ 3 & 2 & 9 \end{bmatrix}$.

Giải:

Chọn $Y = (1, 1, 1)^t$ và lập bảng tính:

A	Y	AY	A^2Y	A^3Y	A^4Y	A^5Y	A^6Y
2 3 2	1	7	78	900	10589	125128	1480345
4 3 5	1	12	134	1569	18512	218927	2590563
3 2 9	1	14	171	2041	24207	286654	3393124

Ta thấy

$$\frac{(A^6Y)_j}{(A^5Y)_j} = \begin{cases} \frac{1480345}{125128} \approx 11.8306, & j = 1 \\ \frac{2590563}{218927} \approx 11.8330, & j = 2 \\ \frac{3393124}{286654} \approx 11.8370, & j = 3. \end{cases}$$

Vậy chọn trị riêng trội $\lambda_1 \approx 11.83$; vector riêng chọn là A^6Y . Do các vector riêng khác nhau hằng số nên có thể chọn

$$X_1 \approx (1; 1.750; 2.991)^t.$$

(thu được từ vector A^6Y sau khi đã chia tất cả các thành phần cho 1480345).

Chương trình MATLAB

Bạn đọc hãy dựa theo ví dụ để lập chương trình MatLab cho phần này.

7.2 Trường hợp ma trận đối xứng, xác định dương

Xét ma trận đối xứng $A = [a_{ij}]$ ($a_{ij} = a_{ji} \forall i, j$).

Ma trận A được gọi là xác định dương nếu với mọi bộ (x_1, x_2, \dots, x_n) không đồng thời bằng "0" ta đều có

$$\sum_{i,j} a_{ij}x_i x_j > 0.$$

Ta đã biết nếu A là ma trận đối xứng, xác định dương thì mọi trị riêng đều là thực, phân biệt và dương. Nghĩa là, hai vector riêng $X = (x_1, x_2, \dots, x_n)$ và $Y = (y_1, y_2, \dots, y_n)$ ứng với hai trị riêng khác nhau thì trực giao với nhau, tức là:

$$\langle X, Y \rangle = \sum_{k=1}^n x_k y_k = 0, \quad (X \neq Y).$$

Điều kiện cần và đủ để A là ma trận xác định dương là:

$$\begin{aligned}\Delta_1 &= a_{11} > 0; \quad \Delta_2 = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} > 0; \\ &\dots \quad \dots \quad \dots \\ \Delta_k &= \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1k} \\ a_{21} & a_{22} & \dots & a_{2k} \\ \vdots & \vdots & \vdots & \vdots \\ a_{k1} & a_{k2} & \dots & a_{kk} \end{vmatrix} > 0, \quad \forall k = \overline{1, n}.\end{aligned}$$

Đối với ma trận đối xứng, xác định dương ta có thể tìm được tất cả các trị riêng và vector riêng dựa vào tính chất trực giao của các vector riêng.

Thuật toán: Xuất phát từ phương trình

$$AX = \lambda X \quad (4.81)$$

hay

$$\sum_{j=1}^n a_{ij}x_j = \lambda x_i, \quad i = \overline{1, n}. \quad (4.82)$$

Ta thấy hệ (4.82) gồm n phương trình, $n+1$ ẩn là x_i , $i = \overline{1, n}$ và λ . Nhưng do các vector riêng ứng với trị riêng nào đó sai khác nhau một hằng số nhân, nên trong các ẩn x_1, x_2, \dots, x_n , ta có thể chọn một ẩn nào đó bằng "1". Khi đó, hệ còn lại đù n phương trình và n ẩn. Quá trình giải (4.82) được tiến hành qua các bước:

- *Bước 1: Chọn $x_n = 1$, hệ (4.82) được viết lại dưới dạng:*

$$\begin{aligned}x_i &= \frac{1}{\lambda} \left[\sum_{j=1}^{n-1} a_{ij}x_j + a_{i,n} \right], \quad i = \overline{1, n-1} \\ \lambda &= \sum_{j=1}^{n-1} a_{ij}x_j + a_{n,n}.\end{aligned} \quad (4.83)$$

Hệ (4.83) tuyến tính đối với x_i ($i = \overline{1, n-1}$), phi tuyến theo λ . Ta giải hệ này bằng phương pháp lặp.

Chọn xấp xỉ đầu bất kỳ

$$X^{(0)} = \left(x_1^{(0)}, x_2^{(0)}, \dots, x_{n-1}^{(0)}, 1 \right)^t,$$

sau đó tính

$$\lambda^{(0)} = \sum_{j=1}^{n-1} a_{nj} x_j^{(0)} + a_{nn}$$

ta được xấp xỉ đầu của $\lambda^{(0)}$.

Quá trình lặp thực hiện theo công thức:

$$\begin{aligned} x_i^{(m)} &= \frac{1}{\lambda^{(m-1)}} \left[\sum_{j=1}^{n-1} a_{ij} x_j^{(m-1)} + a_{i,n} \right], \quad i = \overline{1, n-1} \\ \lambda^{(m)} &= \sum_{j=1}^{n-1} a_{nj} x_j^{(m-1)} + a_{n,n}, \quad m = 1, 2, \dots \end{aligned} \quad (4.84)$$

Nếu quá trình dừng ở bước thứ m thì ta được:

$$\lambda_1 \approx \lambda^{(m)}, \quad X = (x_1^{(m)}, x_2^{(m)}, \dots, x_{n-1}^{(m)}, 1)^t \approx (x_1, x_2, \dots, x_{n-1}, 1)^t.$$

Bước 2: Tìm trị riêng và vector riêng thứ hai là λ_2 và

$Y = (y_1, y_2, \dots, y_n)^t$. Do X và Y trực giao với nhau (vì $\lambda_1 \neq \lambda_2$), nên

$$\langle X, Y \rangle = \sum_{j=1}^n x_j y_j = 0 \quad (x_n = 1)$$

hay

$$y_n = - \sum_{j=1}^{n-1} x_j y_j, \quad (4.85)$$

nghĩa là, nếu tìm được y_j , $j = \overline{1, n-1}$ thì có y_n .

Lặp lại quá trình trên, chọn $y_{n-1} = 1$ ta thu được:

$$y_n = - \sum_{j=1}^{n-2} x_j y_j - x_{n-1}. \quad (4.86)$$

Thay vào hệ (4.83) ta được hệ $n-1$ phương trình $n-1$ ẩn là y_1, y_2, \dots, y_{n-2} và λ .

Chọn xấp xỉ đầu $y_1^{(0)}, y_2^{(0)}, \dots, y_{n-2}^{(0)}, 1$, lặp lại bước 1 ta được $y_1^{(m)}, y_2^{(m)}, \dots, y_{n-2}^{(m)}, 1$ và y_n tính theo (4.85).

Sau n bước ta tìm được tất cả các trị riêng và vector riêng tương ứng.

Ví dụ 4.14. Tìm trị riêng và vector riêng gán đúng của ma trận

$$A = \begin{bmatrix} 4 & 2 & 2 \\ 2 & 5 & 1 \\ 2 & 1 & 6 \end{bmatrix}.$$

Giải:

Ma trận A đã cho là ma trận đối xứng, xác định dương (bạn đọc hãy tự kiểm tra). Từ $AX = \lambda X$ ta có

$$\begin{cases} 4x_1 + 2x_2 + 2x_3 = \lambda x_1 \\ 2x_1 + 5x_2 + x_3 = \lambda x_2 \\ 2x_1 + x_2 + 6x_3 = \lambda x_3 \end{cases} \quad (\text{a})$$

- *Bước 1:* Chọn $x_3 = 1$, từ (a) ta suy ra

$$\begin{cases} x_1 = \frac{1}{\lambda}(4x_1 + 2x_2 + 2) \\ x_2 = \frac{1}{\lambda}(2x_1 + 5x_2 + 1) \\ \lambda = 2x_1 + x_2 + 6 \end{cases} \quad (\text{b})$$

Chọn xấp xỉ đầu $x_1^{(0)} = 1$, $x_2^{(0)} = 1$. Thay vào (b) ta thu được $\lambda^{(0)} = 9$ và $x_1^{(1)} = 0.89$, $x_2^{(1)} = 0.89$, $x_3^{(1)} = 1$. Cứ thế tiếp tục ta thu được kết quả tính như trong bảng sau:

m	$x_1^{(m)}$	$x_2^{(m)}$	$\lambda^{(m)}$
0	1	1	9
1	0.89	0.89	8.67
2	0.85	0.83	8.53
3	0.83	0.80	8.46
4	0.81	0.78	8.40
5	0.8505	0.770	8.38
6	0.806	0.771	8.383
7	0.807	0.771	8.385
8	0.8074	0.7715	8.3863
9	0.8076	0.7717	8.3869
10	0.8076	0.7719	8.3871
11	0.8077	0.7720	8.3874

Ta thấy với hai phép lặp liên tiếp 10 và 11 thì các thành phần tương ứng khá gần nhau nên có thể chọn $\lambda_1 \approx 8.3874$ thì $X = (0.8077; 0.7720; 1)^t$.

- *Bước 2:* Tìm λ_2 và vector riêng tương ứng.

Ta gọi vector riêng tương ứng với trị riêng λ_2 là $Y = (y_1, y_2, y_3)^t$ trong đó $y_2 = 1$ còn y_3 được xác định từ điều kiện

$$\begin{aligned} \langle X, Y \rangle &= x_1 y_1 + x_2 y_2 + y_3 = 0 \\ \Rightarrow y_3 &= -0.8077 y_1 - 0.7720. \end{aligned} \quad (\text{c})$$

Từ hệ (a) ta thay x_i bởi y_i , với $y_2 = 1$, y_3 theo (c) ta thu được hệ

$$\begin{cases} y_1 &= \frac{1}{\lambda} (2.3864 y_1 + 0.4560) \\ \lambda_1 &= 1.1923 y_1 + 4.2280 \end{cases} \quad (\text{d})$$

Chọn xấp xỉ đầu $y_1^{(0)} = 1 \Rightarrow \lambda^{(0)} = 5.42$; $y_1^{(1)} = 0.52$. Kết quả tính chi tiết được cho trong bảng dưới đây:

m	$y_1^{(m)}$	$\lambda^{(m)}$
0	1	5.42
1	0.52	4.85
2	0.35	4.64
3	0.28	4.56
4	0.25	4.53
5	0.23	4.50
6	0.223	4.494
7	0.220	4.490
8	0.218	4.488
9	0.2174	4.487
10	0.2171	4.486
11	0.2170	4.4867

Chọn $\lambda_2 \approx 4.4867$; $y_1 \approx 0.2170$. Thay y_1 vào (c) ta có $y_3 \approx -0.9473$. Vậy $\lambda_2 \approx 4.4867$ và $Y = (0.2170; 1; -0.9473)^t$.

- Bước 3: Tìm λ_3 và vector riêng tương ứng $Z = (z_1, z_2, z_3)^t$. Ta có

$$\begin{cases} \langle Z, X \rangle \approx 0.8077z_1 + 0.7720z_2 + z_3 = 0; \\ \langle Z, Y \rangle \approx 0.2170z_1 + z_2 - 0.9473z_3 = 0. \end{cases}$$

Chọn $z_1 = 1$ và giải hệ này ta được $z_2 \approx -0.5673$; $z_3 \approx -0.3698$.

Từ hệ (a) thay X bởi Z và z_1, z_2 đồng thời chọn $z_1 = 1$ ta thu được $\lambda \approx 2.1260$ và $Z = (1; -0.5673; -0.3698)^t$.

Tóm lại, ta có các trị riêng và vector riêng tương ứng:

$$\begin{aligned} \lambda_1 &\approx 8.3871, \quad X \approx (0.8077; 0.7720; 1)^t; \\ \lambda_2 &\approx 4.4867, \quad Y \approx (0.2170; 1; -0.9473)^t; \\ \lambda_3 &\approx 2.1260, \quad Z \approx (1; -0.5673; -0.3698)^t. \end{aligned}$$

Chương trình MATLAB

Bạn đọc hãy dựa theo ví dụ để lập chương trình MatLab cho phần này.

§8. BÀI TẬP

Bài tập 4.1. Giải hệ phương trình $Ax = b$ bằng phương pháp Gauss-Jordan, trong đó:

$$1. A = \begin{bmatrix} 5 & 3 & 1 & 2 \\ 4 & 7 & 2 & 1 \\ 1 & 3 & 9 & 2 \\ 2 & 1 & 0 & 8 \end{bmatrix}, \quad b = \begin{bmatrix} 6 \\ 6 \\ 10 \\ 2 \end{bmatrix}; \quad 2. A = \begin{bmatrix} 7 & 2 & 0 & 1 \\ 1 & 0 & 3 & 2 \\ 2 & 1 & 0 & 3 \\ 3 & 0 & 2 & 7 \end{bmatrix}, \quad b = \begin{bmatrix} 3 \\ 2 \\ 4 \\ 7 \end{bmatrix};$$

$$3. A = \begin{bmatrix} 3.5 & 1.1 & 0.1 & -0.1 \\ 1.0 & -3.1 & 1.0 & 0.2 \\ 0.1 & 1.0 & 5.7 & -0.2 \\ -0.1 & 0.2 & -0.2 & 2.1 \end{bmatrix}, \quad b = \begin{bmatrix} 0.2 \\ 0.8 \\ 5.9 \\ -2.3 \end{bmatrix};$$

$$4. A = \begin{bmatrix} 1.7 & 0.2 & 0.1 & 0.3 \\ 0.2 & 1.9 & -0.1 & 0.2 \\ 0.1 & -0.1 & 1.3 & -0.4 \\ 0.3 & 0.2 & -0.4 & 4.1 \end{bmatrix}, \quad b = \begin{bmatrix} 1.6 \\ 0.3 \\ -1.2 \\ 0.7 \end{bmatrix};$$

$$5. A = \begin{bmatrix} 6.1 & 1.7 & 1.0 & 9.1 \\ 1.7 & 1.6 & 1.1 & 8.1 \\ 1.0 & 1.1 & 1.0 & 6.1 \\ 9.1 & 8.1 & 0.8 & 7.1 \end{bmatrix}, \quad b = \begin{bmatrix} 18.27 \\ 12.88 \\ 9.51 \\ 26.07 \end{bmatrix};$$

$$6. A = \begin{bmatrix} 3.2 & 7.7 & 0.6 & 9.1 \\ 1.7 & 1.6 & 1.1 & 8.1 \\ 1.0 & 1.1 & 1.0 & 6.1 \\ 9.1 & 8.1 & 0.8 & 7.1 \end{bmatrix}, \quad b = \begin{bmatrix} 8.26 \\ 1.89 \\ 1.15 \\ 2.40 \end{bmatrix}.$$

Quá trình tính theo chương trình MATLAB lấy tới 7 số lẻ sau dấu phẩy (nếu có).

Bài tập 4.2. Giải hệ phương trình bằng phương pháp Gauss-Jordan (tính theo thuật toán kẻ bảng và sử dụng máy tính tay).

$$1. \begin{cases} x_1 + x_2 - 2x_3 + x_4 = 1 \\ x_1 - 3x_2 + x_3 + x_4 = 0 \\ 4x_1 - x_2 - x_3 - x_4 = 1 \\ 4x_1 + 3x_2 - 4x_3 - x_4 = 2 \end{cases}; \quad 2. \begin{cases} 6x - 5y + 7z + 8t = 3 \\ 3x + 11y + 2z + 4t = 6 \\ 3x + 2y + 3z + 4t = 1 \\ x + y + z = 0 \end{cases}.$$

Bài tập 4.3. Giải hệ phương trình sau bằng phương pháp Cholesky (khai căn).

$$1. A = \begin{bmatrix} 5 & 3 & 2 & 1 \\ 3 & 6 & 1 & 2 \\ 2 & 1 & 5 & 1 \\ 1 & 2 & 1 & 6 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 4 \\ 7 \\ 2 \end{bmatrix}; \quad 2. A = \begin{bmatrix} 7 & 4 & 3 & 1 \\ 4 & 8 & 2 & 5 \\ 3 & 2 & 7 & 4 \\ 1 & 5 & 4 & 11 \end{bmatrix}, \quad b = \begin{bmatrix} 7 \\ 10 \\ 9 \\ 9 \end{bmatrix};$$

$$3. A = \begin{bmatrix} 4.9 & 1.0 & 0.1 & 1.1 \\ 1.0 & 6.4 & 1.2 & 0.2 \\ 0.1 & 1.2 & 3.6 & 1.1 \\ 1.1 & 0.2 & 1.1 & 6.4 \end{bmatrix}, \quad b = \begin{bmatrix} 5.0 \\ 2.2 \\ 3.7 \\ 2.2 \end{bmatrix};$$

$$4. A = \begin{bmatrix} 7.7 & 1.2 & 1.1 & 2.1 \\ 1.2 & 8.7 & 1.3 & 1.1 \\ 1.1 & 1.3 & 8.3 & 1.0 \\ 2.1 & 1.1 & 1.0 & 4.9 \end{bmatrix}, \quad b = \begin{bmatrix} 9.8 \\ 2.3 \\ 2.1 \\ 7.0 \end{bmatrix}.$$

Tính trực tiếp theo công thức và theo chương trình MATLAB.

Bài tập 4.4. Giải các hệ phương trình sau bằng phương pháp lặp đơn theo cách áp dụng chương trình MATLAB sao cho đạt sai số $\varepsilon \leq 10^{-4}$.

$$\begin{array}{ll} 1. \begin{cases} -8x_1 + x_2 + x_3 = 1 \\ x_1 - 5x_2 + x_3 = 16; \\ x_1 + x_2 - 4x_3 = 7 \end{cases} & 3. \begin{cases} 2x_1 + 3x_2 - 4x_3 + x_4 = 3 \\ x_1 - 2x_2 - 5x_3 + x_4 = 2 \\ 5x_1 - 3x_2 + x_3 - 4x_4 = 1 \\ 10x_1 + 2x_2 - x_3 + 2x_4 = -4 \end{cases} \\ 2. \begin{cases} 1.02x - 0.25y - 0.30z = 0.515 \\ -0.41x + 1.13y - 0.15z = 1.555; \\ -0.25x - 0.14y + 1.21z = 2.780 \end{cases} & \end{array}$$

Bài tập 4.5. Tìm ma trận nghịch đảo của các ma trận theo phương pháp Gauss-Jordan bằng cách biến đổi trực tiếp theo thuật toán tính toán và cách dùng chương trình MATLAB đã viết.

$$\begin{array}{ll} 1. A = \begin{bmatrix} 3 & 2 & 2 \\ 1 & 3 & 1 \\ 5 & 3 & 4 \end{bmatrix}; & 3. A = \begin{bmatrix} 13 & 14 & 6 & 4 \\ 8 & -1 & 13 & 9 \\ 6 & 7 & 3 & 2 \\ 9 & 5 & 16 & 11 \end{bmatrix}. \\ 2. A = \begin{bmatrix} 1 & 2 & 6 \\ 2 & 5 & 15 \\ 6 & 15 & 46 \end{bmatrix}; & \end{array}$$

Bài tập 4.6. Tìm ma trận nghịch đảo của ma trận

$$A = \begin{bmatrix} 1 & 2 & 6 \\ 2 & 5 & 15 \\ 6 & 15 & 46 \end{bmatrix}$$

sử dụng phương pháp Cholesky bằng cách biến đổi trực tiếp theo thuật toán tính toán và cách dùng chương trình MATLAB đã viết.

Bài tập 4.7. Xây dựng chương trình MATLAB tìm đa thức đặc trưng của ma trận theo phương pháp Danhilepski. Áp dụng tính với các ma trận sau:

$$1. \ A = \begin{bmatrix} 6.1 & 1.0 & 5.1 \\ 1.0 & 0.5 & 6.1 \\ 5.1 & 6.1 & 7.2 \end{bmatrix}; \quad 2. \ A = \begin{bmatrix} 5.1 & 1.1 & 1.0 \\ 1.1 & 6.1 & 1.1 \\ 1.0 & 1.1 & 5.1 \end{bmatrix}.$$

Từ đó tìm trị riêng và vector riêng tương ứng.

Bài tập 4.8. Tìm trị riêng và vector riêng của các ma trận bằng phương pháp Krulôp A. N.

$$1. \ A = \begin{bmatrix} 5.7 & 1.2 & 2.0 \\ 1.2 & 2.7 & 1.0 \\ 2.0 & 1.2 & 7.2 \end{bmatrix}; \quad 2. \ A = \begin{bmatrix} 6.1 & 1.7 & 1.0 \\ 1.7 & 1.6 & 1.1 \\ 1.0 & 1.1 & 5.1 \end{bmatrix}.$$

Xây dựng chương trình MATLAB minh họa thuật toán trên.

Bài tập 4.9. Tìm trị riêng trội và vector riêng tương ứng của các ma trận sau:

$$1. \ A = \begin{bmatrix} 2.1 & 1.9 & 1.8 & 1.7 \\ 1.9 & 2.0 & 1.7 & 1.8 \\ 1.8 & 1.7 & 2.1 & 1.9 \\ 1.7 & 1.8 & 1.9 & 2.0 \end{bmatrix}; \quad 2. \ A = \begin{bmatrix} 5.1 & 4.1 & 3.1 & 2.1 \\ 4.1 & 6.1 & 2.1 & 5.1 \\ 3.1 & 2.1 & 5.1 & 4.1 \\ 2.1 & 5.1 & 4.1 & 6.1 \end{bmatrix}.$$

Kết quả tính lấy 5 chữ số sau dấu phẩy.

Bài tập 4.10. Tìm trị riêng và vector riêng của ma trận A theo phương pháp trực giao.

$$1. \ A = \begin{bmatrix} 5 & 3 & 2 \\ 3 & 6 & 1 \\ 2 & 1 & 5 \end{bmatrix}; \quad 2. \ A = \begin{bmatrix} 4.9 & 0.1 & 1.1 \\ 0.1 & 3.6 & 1.0 \\ 1.1 & 1.0 & 4.9 \end{bmatrix}$$

Xây dựng chương trình MATLAB minh họa phương pháp trên.

CHƯƠNG 5

PHÉP NỘI SUY VÀ XẤP XÌ HÀM

TRONG thực tế, ta thường gặp dạng hàm số $y = f(x)$ mà không biết biểu thức giải tích cụ thể của nó. Thông thường, bằng đặc, thực nghiệm ta chỉ thu được trong dạng một bảng số, nghĩa là biết giá trị y_i tại các điểm x_i ($i = \overline{0, n}$) tương ứng thuộc đoạn $[a, b]$ nào đó. Vấn đề là cần tính giá trị \bar{y} tại điểm $\bar{x} \neq x_i$, $i = \overline{0, n}$ mà không thể thu được bằng các phương pháp thực nghiệm như trên. Cũng có trường hợp biết quy luật biến đổi $y = f(x)$, nhưng biểu thức của nó quá phức tạp thì giá trị $\bar{y} = f(\bar{x})$ cũng khó mà tìm được. Vì vậy, người ta tìm cách thay hàm $f(x)$ bởi hàm $F(x)$ đơn giản hơn, dễ tính $F(\bar{x})$ sao cho sự sai khác giữa $f(\bar{x})$ và $F(\bar{x})$ càng bé càng tốt. Cách thay $f(x)$ bởi $F(x)$ như vậy gọi là xấp xỉ hàm. Có nhiều cách xấp xỉ khác nhau, ở đây ta chỉ để cập đến một phương pháp xấp xỉ đơn giản nhất là: Xấp xỉ $f(x)$ bởi hàm trong dạng đa thức, vì khi tính giá trị của đa thức tại điểm \bar{x} đã có sơ đồ Horner (dễ tính toán).

§1. KHÁI NIỆM VỀ NỘI SUY

1.1 Bài toán

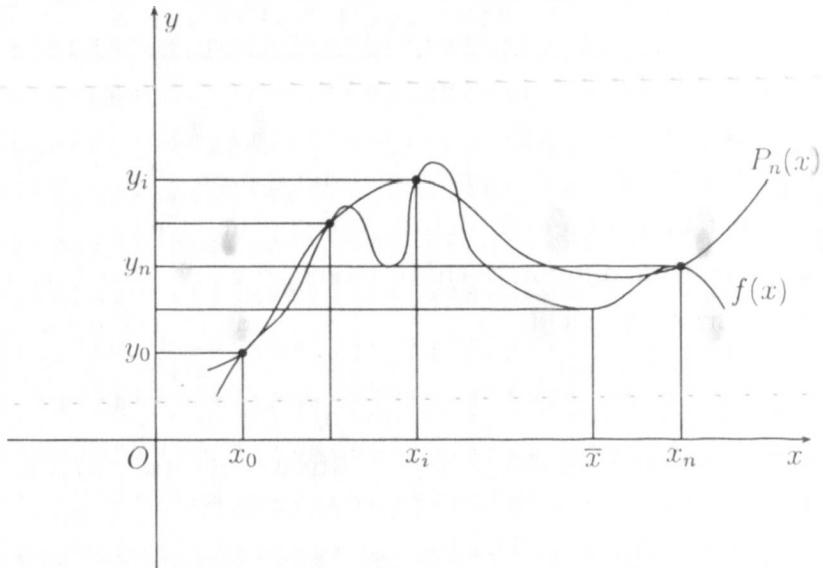
Giả sử có hàm số $y = f(x)$ xác định trên đoạn $[a, b]$. Bằng cách nào đó ta thu được bảng số

$$y_i = f(x_i), \quad i = \overline{0, n}; \quad x_i \in [a, b]. \quad (5.1)$$

Từ bảng số (5.1), hãy xây dựng đa thức $P_n(x)$ bậc $\leq n$ sao cho

$$P_n(x_i) = y_i, \quad i = \overline{0, n}. \quad (5.2)$$

Đa thức $P_n(x)$ sinh ra từ bảng số (5.1), thỏa mãn điều kiện (5.2) gọi là đa thức nội suy (hay công thức nội suy) và ta có $P_n(x) \approx f(x)$ (xấp xỉ) trên đoạn $[a, b]$. Các điểm $x_i \in [a, b]$ ($i = \overline{0, n}$) gọi là các mốc nội suy (hình 5.1).



Hình 5.1

Giá trị \bar{y} tại điểm $\bar{x} \in (a, b)$; $\bar{x} \neq x_i$, ($i = \overline{0, n}$), tức là

$$\bar{y} = P_n(\bar{x}) \approx f(\bar{x})$$

gọi là giá trị nội suy, còn với $\bar{x} \notin [a, b]$ thì ta gọi

$$\bar{y} = P_n(\bar{x}) \approx f(\bar{x})$$

là giá trị ngoại suy.

Hiệu số

$$R_n(\bar{x}) = f(\bar{x}) - P_n(\bar{x}) \quad (5.3)$$

gọi là sai số của phép nội suy tại điểm \bar{x} .

1.2 Sự duy nhất của đa thức nội suy

Định lý 5.1. *Đa thức $P_n(x)$ bậc $\leq n$ sinh ra từ bảng số (5.1), thỏa mãn điều kiện (5.2) là duy nhất.*

Chứng minh. Giả sử có hai đa thức $P_n(x)$, $Q_n(x)$ với bậc $\leq n$ sinh ra từ bảng số (5.1), thỏa mãn điều kiện (5.2).

Xét

$$H(x) = P_n(x) - Q_n(x)$$

là đa thức bậc $\leq n$. Mặt khác ta có

$$H(x_i) = P_n(x_i) - Q_n(x_i) = y_i - y_i = 0, \quad \forall i = \overline{0, n},$$

nghĩa là $H(x)$ là đa thức bậc $\leq n$ có $n+1$ nghiệm x_i ($i = \overline{0, n}$). Điều này chỉ xảy ra khi $H(x) \equiv 0$, tức là ta có

$$P_n(x) \equiv Q_n(x). \quad \blacksquare$$

Sau đây ta nêu một số công thức nội suy thường được sử dụng trong thực tế.

§2. ĐA THỨC NỘI SUY LAGRANGE

2.1 Đa thức nội suy Lagrange

Giả sử đa thức $P_n(x)$ bậc n sinh ra từ bảng số

$$y_i = f(x_i), \quad i = \overline{0, n} \quad (5.4)$$

với các mốc nội suy

$$a \leq x_0 < x_1 < \cdots < x_n \leq b$$

thỏa mãn điều kiện

$$P_n(x_i) = y_i \quad i = \overline{0, n}. \quad (5.5)$$

Lagrange dựa vào các tính chất của đa thức như: tổng hai đa thức bậc n là đa thức bậc $\leq n$; đa thức bậc n nhân với hằng số $a \neq 0$ cũng là đa thức

bậc n . Vì vậy, Lagrange lập đa thức cơ sở bậc n :

$$L_j(x) = \frac{(x - x_0)(x - x_1) \dots (x - x_{j-1})(x - x_{j+1}) \dots (x - x_n)}{(x_j - x_0)(x_j - x_1) \dots (x_j - x_{j-1})(x_j - x_{j+1}) \dots (x_j - x_n)}, \quad j = \overline{0, n}. \quad (5.6)$$

Đa thức cơ sở (5.6) có tính chất

$$L_j(x_i) = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases}, \quad i, j = \overline{0, n}. \quad (5.7)$$

Chọn

$$P_n(x) = \sum_{j=0}^n L_j(x) y_j. \quad (5.8)$$

Đa thức (5.8) là đa thức bậc n và do (5.7) ta dễ dàng suy ra

$$P_n(x_i) = \sum_{j=0}^n L_j(x_i) y_j = y_i, \quad (i = \overline{0, n}).$$

Vậy $P_n(x)$ là đa thức nội suy sinh ra từ bảng số (5.4), thỏa mãn điều kiện (5.5) và $P_n(x) \approx f(x)$ (xấp xỉ) được xác định theo (5.8) gọi là đa thức nội suy Lagrange.

2.2 Sai số của đa thức nội suy Lagrange

Với x cố định, $x \neq x_i$ ($i = \overline{0, n}$) thì

$$R_n(x) = f(x) - P_n(x) \quad (5.9)$$

được gọi là sai số tại điểm x .

Người ta đã chứng minh được rằng, nếu hàm số $f(x)$ xác định, liên tục và có đạo hàm liên tục đến cấp $n + 1$, đồng thời nếu

$$|f^{(n+1)}(x)| \leq M \quad \forall x \in [a, b] \ni x_i |_{i=\overline{0, n}}$$

thì

$$R_n(x) = f(x) - P_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \omega_{n+1}(x), \quad (5.10)$$

trong đó $\xi \in [a, b]$, $\omega_{n+1}(x) = \prod_{i=0}^n (x - x_i)$. Từ (5.10) ta suy ra

$$|R_n(x)| \leq \frac{M}{(n+1)!} |\omega_{n+1}(x)|. \quad (5.11)$$

2.3 Chương trình MATLAB

```

function v=lagrangeinterp(x,y,u)
% lagrangeinterp computes
% the Lagrange interpolation polynomial

%Input: _ x, y are vectors of the same length
% that define the interpolating points.
%       _ u is the vector of points where
%           the function is to be evaluated.

% Output: v is the same length as u
if nargin<3
    u=sym('x');
end
n=length(x);
v=zeros(size(u));
for i=1:n
    w=ones(size(u));
    for j=[1:i-1 i+1:n]
        w=(u-x(j))./(x(i)-x(j)).*w;
    end
    v=v+w*y(i);
end
end

```

Ví dụ 5.1. Tìm đa thức nội suy Lagrange của hàm số được cho theo bảng số sau:

<i>i</i>	0	1	2	3	4
<i>x_i</i>	1	2	3	4	7
<i>y_i</i>	17	17.5	76	210.5	1970

Áp dụng chương trình MATLAB trên ta thu được

$$y = f(x) \approx p_4(x) = 2x^4 - 17x^3 + 81x^2 - 153.5x + 104.5$$

§3. ĐA THỨC NỘI SUY NEWTON CÓ MỐC CÁCH ĐỀU

Xét bài toán ((5.1), (5.2)) trong §1, trong đó các mốc nội suy cách đều

$$a \equiv x_0 < x_1 < \cdots < x_n \equiv b$$

và $h = x_i - x_{i-1} = \frac{b-a}{n}$, ($i = \overline{1, n}$) gọi là bước từ mốc x_{i-1} đến mốc x_i .

3.1 Khái niệm về sai phân

3.1.1. Sai phân tiến

Ta gọi

$$\Delta y_i = y_{i+1} - y_i \quad (i = \overline{0, n-1})$$

là sai phân tiến cấp một.

Sai phân tiến của sai phân tiến cấp một gọi là sai phân tiến cấp hai và được ký hiệu bởi

$$\Delta^2 y_i = \Delta(\Delta y_i) = \Delta y_{i+1} - \Delta y_i = y_{i+2} - 2y_{i+1} + y_i, \quad (i = \overline{0, n-2}).$$

Tổng quát, sai phân tiến của sai phân tiến cấp $n-1$ được gọi là sai phân tiến cấp n , ký hiệu

$$\Delta^n y_0 = \Delta(\Delta^{n-1} y_0) = \Delta^{n-1} y_1 - \Delta^{n-1} y_0. \quad (5.12)$$

3.1.2. Sai phân lùi

Ta gọi

$$\nabla y_i = y_i - y_{i-1} \quad i = \overline{1, n}$$

là sai phân lùi cấp một.

Sai phân lùi của sai phân lùi cấp $n-1$ gọi là sai phân lùi cấp n và được tính theo công thức

$$\nabla^n y_1 = \nabla(\nabla^{n-1} y_1) = \nabla^{n-1} y_1 - \nabla^{n-1} y_0. \quad (5.13)$$

Từ khái niệm sai phân tiến, lùi ta suy ra mối liên hệ giữa chúng:

$$\Delta^k y_i = \nabla^k y_{i+k}. \quad (5.14)$$

Quá trình tính sai phân tiến (5.12) được mô tả trong bảng sau:

\vdots	\vdots			
x_{i-2}	y_{i-2}	Δy_{i-2}		
x_{i-1}	y_{i-1}		$\Delta^2 y_{i-2}$	
		Δy_{i-1}		$\Delta^3 y_{i-2}$
x_i	y_i		$\Delta^2 y_{i-1}$	$\Delta^4 y_{i-2}$
		Δy_i		$\Delta^3 y_{i-1}$
x_{i+1}	y_{i+1}		$\Delta^2 y_i$	
		Δy_{i+1}		
x_{i+2}	y_{i+2}			
\vdots	\vdots			

Theo liên hệ (5.14) thì bảng sai phân tiến cũng là bảng sai phân lùi, hàng cuối cùng chính là $\nabla y_n, \nabla^2 y_n, \dots, \nabla^n y_n$.

Ta xem $x \equiv x_i$ là mốc x_i nào đó thì mốc $x_{i+1} = x + h$ và xem Δ là toán tử tác động lên hàm f tại mốc x , nghĩa là:

$$\Delta f(x) = f(x + h) - f(x)$$

$$If(x) = \Delta^0 f(x) = f(x) \text{ là toán tử đơn vị hay đồng nhất.}$$

Các tính chất sau đây phát biểu đối với sai phân tiến và cũng đúng với sai phân lùi.

Tính chất 1. Toán tử Δ là toán tử tuyến tính, nghĩa là $\forall \alpha, \beta \in \mathbb{R}: \forall f, g$ ta có

$$\Delta(\alpha f + \beta g)(x) = \alpha \Delta f(x) + \beta \Delta g(x),$$

đồng thời

$$\Delta^m(\Delta^k f)(x) = \Delta^{m+k} f(x).$$

Tính chất 2. Nếu $f(x) \equiv c$ với $c - \text{const}$ thì $\Delta f = 0 \implies \Delta^m f = 0 \quad \forall m = 1, 2, \dots$

Nếu $f(x) = x^n$ thì $\Delta^n(x^n) = n!h^n$ và $\Delta^m(x^n) = 0 \quad \forall m > n$.

Các tính chất 1,2 có thể dễ dàng suy ra từ định nghĩa về sai phân.

Tính chất 3. Giá trị của hàm số $f(x)$ tại mốc $x + mh$ được biểu diễn thông qua sai phân các cấp của nó theo công thức

$$f(x + mh) = \sum_{k=0}^m C_m^k \Delta^k f(x), \quad (5.15)$$

trong đó $C_m^k = \frac{m!}{k!(m-k)!}$.

Chứng minh. Từ $\Delta f(x) = f(x+h) - f(x)$ ta suy ra

$$\begin{aligned} f(x+h) &= \Delta f(x) + f(x) = (\Delta + I)f(x); \\ f(x+2h) &= f(x+h+h) = (\Delta + I)f(x+h) = (\Delta + I)^2 f(x). \end{aligned}$$

Áp dụng công thức truy hồi ta thu được

$$f(x+mh) = (\Delta + I)^m f(x).$$

Khai triển $(\Delta + I)^m$ theo công thức nhị thức Newton ta thu được (5.15). ■

3.2 Đa thức nội suy Newton tiến có mốc nội suy cách đều

Xuất phát từ mốc x_0 (đầu bảng), đặt $x = x_0 + ht$, từ đó suy ra

$$\begin{aligned} x - x_0 &= ht \\ x - x_1 &= h(t-1) \\ &\vdots \\ x - x_k &= h(t-k). \end{aligned}$$

Xây dựng đa thức nội suy trong dạng

$$\begin{aligned} p_n(x) &= p_n(x_0 + ht) \\ &= P_n(t) = a_0 + a_1 t + a_2 t(t-1) + \cdots + a_n t(t-1) \dots (t-n+1), \end{aligned} \quad (5.16)$$

trong đó a_i ($i = \overline{0, n}$) sẽ được xác định sao cho

$$p_n(x_i) = P_n(t_i) = y_i, \quad i = \overline{0, n}. \quad (5.17)$$

Ta thấy

- khi $x = x_0$ thì từ $x - x_0 = ht$ suy ra $t = 0$;
- khi $x = x_1$ thì $x_1 - x_0 = h = ht$ suy ra $t = 1$;
- ...
- khi $x_k - x_0 = kh = ht$ suy ra $t = k$.

Từ (5.16) và điều kiện (5.17) ta có

$$p_n(x_0) = P_n(0) = a_0 = y_0$$

$$p_n(x_1) = P_n(1) = a_0 + a_1 \cdot 1 = y_1 \Rightarrow a_1 = y_1 - a_0 = y_1 - y_0 = \Delta y_0;$$

$$p_n(x_2) = P_n(2) = a_0 + a_1 \cdot 2 + a_2 \cdot 2 = y_0 + 2(y_1 - y_0) + 2a_2 = y_2$$

hay

$$a_2 = \frac{y_0 - 2y_1 + y_2}{2} = \frac{1}{2!} \Delta^2 y_0.$$

Truy hồi ta được

$$a_k = \frac{1}{k!} \Delta^k y_0, \quad k = 1, 2, \dots, n. \quad (5.18)$$

Thay vào (5.16) ta thu được

$$\begin{aligned} p_n(x) &= p_n(x_0 + h) \\ &= P_n(t) = y_0 + \frac{t}{1!} \Delta y_0 + \frac{t(t-1)}{2!} \Delta^2 y_0 + \dots + \frac{t(t-1)\cdots(t-n+1)}{n!} \Delta^n y_0, \end{aligned} \quad (5.19)$$

trong đó $x = x_0 + ht$.

Công thức (5.19) sinh ra từ bảng số (5.1) thỏa mãn điều kiện (5.2), xuất phát từ mốc x_0 tiến dần lên nên gọi là đa thức nội suy Newton tiến có mốc cách đều. Nhìn vào bảng sai phân, ta thấy công thức (5.19) sử dụng hàng đầu của bảng sai phân.

3.3 Đa thức nội suy Newton lùi có mốc nội suy cách đều

Xuất phát từ mốc x_n cuối bảng. Đặt $x = x_n + ht$ thì

- khi $x = x_n$ ta có $t = 0$;

- khi $x = x_{n-1}$ ta có $x_{n-1} - x_n = -h = ht$ suy ra $t = -1$;
- ...
- khi $x = x_k$ thì $x_k - x_n = -hk = ht$ suy ra $t = -k$.

Hoàn toàn tương tự như trên ta được

$$\begin{aligned} p_n(x) &= p_n(x_n + ht) = P_n(t) = y_n + \frac{t}{1!} \nabla y_n + \frac{t(t+1)}{2!} \nabla^2 y_n + \dots \\ &\quad + \frac{t(t+1)(t+2)\cdots(t+n-1)}{n!} \nabla^n y_n. \end{aligned} \quad (5.20)$$

Do mỗi liên hệ (5.14) nên ta có

$$\begin{aligned} p_n(x) &= p_n(x_n + ht) = P_n(t) \\ &= y_n + \frac{t}{1!} \Delta y_{n-1} + \frac{t(t+1)}{2!} \Delta^2 y_{n-2} + \dots + \frac{t(t+1)(t+2)\cdots(t+n-1)}{n!} \Delta^n y_0. \end{aligned} \quad (5.21)$$

Công thức (5.21) được gọi là đa thức nội suy Newton lùi xuất phát từ nút x_n lùi dần lại. Nhìn vào bảng sai phân, ta thấy công thức (5.21) sử dụng hàng cuối cùng của bảng. Như vậy hai công thức tiến (5.19) và lùi (5.21) sử dụng cùng một bảng sai phân.

3.4 Sai số

Theo công thức (5.10) ta có sai số

$$R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \omega_{n+1}(x). \quad (5.22)$$

Nếu sử dụng công thức nội suy Newton tiến (5.19) với $x = x_0 + ht$ thì

$$\omega_{n+1}(x) = \prod_{x=0}^n (x - x_i) = h^{n+1} \prod_{k=0}^n (t - k).$$

Lại do $f^{(n+1)}(x) = \lim_{h \rightarrow 0} \frac{\Delta^{n+1} y_0}{h^{n+1}}$, nên ta có thể xem

$$f^{(n+1)}(\xi) \approx \frac{\Delta^{n+1} y_0}{h^{n+1}}.$$

Vậy

$$R_n(x) \approx \frac{\Delta^{n+1} y_0}{(n+1)!} \prod_{k=0}^n (t - k) \quad (5.23)$$

là công thức đánh giá sai số.

Chú ý 5.1. Các công thức nội suy Newton tiến (5.19), lùi (5.21) sử dụng theo hai cạnh của tam giác cân trong bảng sai phân. Do đó, khi cần tính giá trị \bar{y} tại điểm \bar{x} gần x_0 , ta sử dụng công thức tiến, còn khi \bar{x} gần x_n ta sử dụng công thức lùi.

Chú ý 5.2. Do đặc điểm trên nên khi cần tính \bar{y} tại \bar{x} gần giữa bảng thì việc sử dụng các công thức tiến, lùi đều được, nhưng bất lợi do một loạt các giá trị sai phân các cấp bị bỏ qua. Vì vậy, trong trường hợp này ta nên sử dụng công thức nội suy Lagrange. Cũng có những công thức khác sử dụng khá tốt như các công thức nội suy trung tâm mà ở đây ta không xét, có thể xem trong tài liệu tham khảo.

Chú ý 5.3. Nếu cần tìm giá trị gần đúng \bar{y} tại điểm $\bar{x} \notin [x_0, x_n]$ (ngoài bảng), trong trường hợp này

- nếu $\bar{x} < x_0$ ta sử dụng công thức tiến (5.19);
- nếu $\bar{x} > x_n$ ta sử dụng công thức lùi (5.21).

Giá trị thu được gọi là giá trị ngoại suy.

Ví dụ 5.2. Giá trị của tích phân xác suất

$$\Phi(x) = \frac{\sqrt{2}}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$

được cho trong bảng sau:

x	1.4	1.5	1.6	1.7	1.8	1.9	2.0
Φ	0.9523	0.9661	0.9763	0.9838	0.9891	0.9928	0.9953

Hãy tìm giá trị gần đúng của $\Phi(1.43)$.

Giải:

Lập bảng sai phân từ hàm số đã cho (mỗi cột số nguyên đều được nhân với 10^{-4}):

x	Φ	$\Delta\Phi$	$\Delta^2\Phi$	$\Delta^3\Phi$	$\Delta^4\Phi$	$\Delta^5\Phi$	$\Delta^6\Phi$
1.4	0.9523
...	...	138
1.5	0.9661	...	-36
...	...	102	...	9
1.6	0.9763	...	-27	...	-4
...	...	75	...	5	...	5	...
1.7	0.9838	...	-22	...	1	...	-8
...	...	53	...	6	...	-3	...
1.8	0.9891	...	-16	...	-2
...	...	37	...	4
1.9	0.9928	...	-12
...	...	25
2.0	0.9953

Do điểm $\bar{x} = 1.43$ gần đầu bảng nên ta sử dụng công thức nội suy Newton tiến (5.19), với $\bar{t} = \frac{\bar{x} - x_0}{h} = 0.3$; $h = 0.1$. Kết quả thu được

$$\Phi(1.43) \approx 0.95686.$$

3.5 Chương trình MATLAB

```

function [v,d]=newtoninterp(x,y,u)
% This function illustrates the Newton interpolation formula

% Input: x, y are vectors of the same length that define
% the interpolating points.

% u is the vector of points where the function is to be
% evaluated.

% Output: v is the same length as u
if nargin<3
    u=sym('x');
end

```

```

n=length(y);
if length(x) ~= n
    error('x and y are not compatible!!!!')
end
% Compute the divided-difference table
d=y(:);

for j=2:n
    for i=n:-1:j
        d(i) = (d(i)-d(i-1))/(x(i)-x(i-j+1));
    end
end
% Computes vector v by using backward interpolation formul
v=d(n)*ones(size(u));
for i=n-1:-1:1
    v = v.* (u-x(i)) + d(i);
end
end

```

§4. PHÉP NỘI SUY NGƯỢC

Trong các tiết trước ta xét bài toán: Tìm giá trị gần đúng \bar{y} của hàm tại điểm $\bar{x} \neq x_i$ ($i = \overline{0, n}$) không có trong bảng số (5.1). Nay giờ ta xét bài toán ngược nghĩa là có giá trị của hàm số là \bar{y} , hãy tìm $\bar{x} \neq x_i$ ($i = \overline{0, n}$) sao cho $\bar{y} \approx f(\bar{x})$. Ở đây ta xét hai hướng giải quyết cơ bản, lần lượt sử dụng công thức nội suy Lagrange và Newton.

4.1 Sử dụng đa thức nội suy Lagrange

Từ bảng số đã cho trong dạng $y = f(x)$, có \bar{y} , hãy tìm \bar{x} tương ứng, nghĩa là tìm $\bar{x} = \varphi(\bar{y})$ là hàm ngược của hàm $y = f(x)$. Như vậy, ta xem y_i , $i = \overline{0, n}$ là giá trị của dồi số, nói chung các giá trị y_i không cách đều

nhau. Bằng đa thức nội suy Lagrange, ta xây dựng được đa thức nội suy

$$x = \varphi(y) \approx Q_n(y) = \sum_{j=0}^n L_j(y) \cdot x_j. \quad (5.24)$$

Từ đó, do đã biết \bar{y} ta có xấp xỉ

$$\bar{x} \approx Q_n(\bar{y}).$$

4.2 Trường hợp các mốc nội suy cách đều

Trong trường hợp này ta có thể sử dụng phương pháp lặp như sau: Giả sử cần tìm \bar{x} , biết \bar{y} ở gần đầu bảng sai phân. Sử dụng đa thức nội suy Newton tiến

$$y \approx p_n(x) = p_n(x_0 + th) = P_n(t) = y_0 + t\Delta y_0 + \varphi(t).$$

trong đó

$$\varphi(t) = \frac{t(t-1)}{2!} \Delta^2 y_0 + \dots,$$

ta suy ra

$$t = \frac{y - y_0}{\Delta y_0} - \frac{1}{\Delta y_0} \varphi(t) = \psi(t), \quad (5.25)$$

trong đó $t = \frac{x - x_0}{h}$.

Do cần tìm \bar{x} ứng với \bar{y} đã cho, ta tìm $\bar{t} = \frac{\bar{x} - x_0}{h}$. Từ phương trình (5.25) thực hiện quá trình lặp với xấp xỉ đầu $t_0 = \frac{\bar{y} - y_0}{\Delta y_0}$ và công thức lặp

$$t_m = \psi(t_{m-1}), \quad m = 1, 2, \dots \quad (5.26)$$

Sau một số bước lặp, ta xem $t_m \approx \bar{t}$, từ đó suy ra $\bar{x} \approx h\bar{t} + x_0$ là giá trị gần đúng cần tìm.

4.3 Chương trình MATLAB

Bạn đọc hãy dựa theo thuật toán để lập chương trình
MatLab cho phần này.

§5. PHƯƠNG PHÁP BÌNH PHƯƠNG TỐI THIỂU HAY XẤP XỈ TRUNG BÌNH PHƯƠNG

Trong các tiết trước ta đã xấp xỉ hàm $f(x)$ bằng đa thức nội suy, xuất phát từ bảng số $y_i = f(x_i)$; $i = \overline{1, n}$ thỏa mãn điều kiện $p_{n-1}(x_i) = y_i$ ($i = \overline{1, n}$). Các giá trị y_i được xem là các giá trị đúng của hàm $f(x)$ tại các mốc x_i ($i = \overline{1, n}$). Nhưng thực tế trong nhiều trường hợp các giá trị y_i đó không có mà chỉ có các giá trị gần đúng, nghĩa là $y_i = f(x_i) + \varepsilon_i$ ($i = \overline{1, n}$). Trong trường hợp này, việc xấp xỉ hàm $f(x)$ bằng đa thức nội suy như đã xét ở trên không còn phù hợp nữa, ta cần xấp xỉ hàm $f(x)$ trong dạng khác thiết thực hơn.

Sau đây ta sẽ đưa ra dạng xấp xỉ khác gọi là xấp xỉ trung bình phương (hay phương pháp bình phương tối thiểu). Sai số tính theo phương pháp này được dàn đều trên đoạn $[a, b]$.

5.1 Khái niệm về sai số trung bình phương

Định nghĩa 5.1. Xét hai hàm số $f(x)$ và $\varphi(x)$. Đại lượng

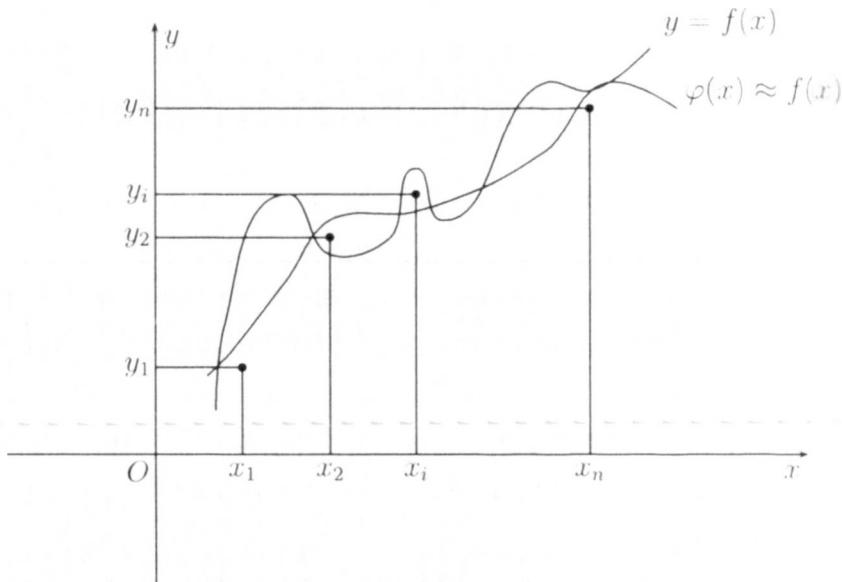
$$\sigma_m = \left(\frac{1}{n} \sum_{i=1}^n [f(x_i) - \varphi(x_i)]^2 \right)^{1/2} \quad (5.27)$$

được gọi là *sai số (hay độ lệch) trung bình phương* của hai hàm $f(x)$ và $\varphi(x)$ trên tập hợp điểm x_i ($i = \overline{1, n}$).

Từ định nghĩa ta thấy, độ lệch σ_m càng nhỏ càng tốt, nghĩa là $\varphi(x)$ càng gần với $f(x)$. Từ đó ta nói hàm $\varphi(x) \approx f(x)$ (xấp xỉ) trên đoạn tập điểm $[x_1, x_2, \dots, x_n] \subset [a, b]$ theo nghĩa trung bình phương (hình 5.2).

Công thức (5.27) gồm tổng các số hạng $|f(x_i) - \varphi(x_i)|^2$ ($i = \overline{1, n}$) là *độ lệch giữa f và φ tại mốc x_i* , đồng thời ta lại có $f(x_i) = y_i + \varepsilon_i$ ($i = \overline{1, n}$). Do đó, thay $f(x_i) \approx y_i$ ta được

$$\sigma_m = \left(\frac{1}{n} \sum_{i=1}^n [y_i - \varphi(x_i)]^2 \right)^{1/2}. \quad (5.28)$$



Hình 5.2

Công thức (5.28) cho thấy sai số trung bình phương σ_m bao gồm cả sai số ε_i sinh ra trong quá trình quan sát tại các mốc x_i .

5.2 Bài toán

Xét $\varphi(x) = F(x, a_0, a_1, \dots, a_m) \in J$ là tập hợp các hàm phụ thuộc vào các tham số a_0, a_1, \dots, a_m . Hãy tìm bộ tham số đó sao cho

$$\sigma_m = \left(\frac{1}{n} \sum_{i=1}^n [y_i - F(x_i, a_0, a_1, \dots, a_m)]^2 \right)^{1/2} \text{đạt min.} \quad (5.29)$$

trong đó $y_i \approx f(x_i)$, $i = \overline{1, n}$.

Nhưng để đạt được (5.29), ta chỉ cần tìm bộ số a_i ($i = \overline{0, m}$) sao cho

$$\phi(a_0, a_1, \dots, a_m) = \sum_{i=1}^n [y_i - F(x_i, a_0, \dots, a_m)]^2 \text{đạt min.} \quad (5.30)$$

Vậy bài toán được đặt lại như sau: Từ bảng số $y_i \approx f(x_i)$ ($i = \overline{1, n}$). Hãy tìm bộ số a_i ($i = \overline{0, m}$) thỏa mãn (5.30).

Giả sử đã tìm được bộ số là \bar{a}_i ($i = \overline{0, m}$) sao cho

$$\varphi(x) = F(x, \bar{a}_0, \bar{a}_1, \dots, \bar{a}_m) \in J.$$

Khi đó, ta nói $\varphi(x) \approx f(x)$ tốt nhất theo nghĩa trung bình phương và phương pháp tìm hàm $\varphi(x)$ như vậy gọi là phương pháp bình phương tối thiểu (cực tiểu). Để đơn giản ta xét $F(x, \bar{a}_0, \bar{a}_1, \dots, \bar{a}_m) \in J$ có dạng tuyến tính đối với các tham số a_0, a_1, \dots, a_m .

5.3 Xây dựng phương pháp tính

Định nghĩa 5.2. (đa thức suy rộng) Cho hệ hàm

$$\{\varphi_k(x)\} \quad (k = \overline{0, m}) \quad (5.31)$$

là các hàm đối với x . Ta gọi

$$P_m(x) = \sum_{k=0}^m a_k \varphi_k(x) \quad (5.32)$$

là đa thức suy rộng của hệ hàm (5.31), trong đó a_k ($k = \overline{0, m}$) là hệ số, còn hệ (5.31) được gọi là hệ cơ bản.

Từ định nghĩa ta thấy, nếu cho hệ $\varphi_k(x) = x^k$ ($k = \overline{0, m}$) thì hệ đó là hệ đại số và $P_m(x)$ là đa thức bậc m . Còn nếu hệ (5.31) là hệ lượng giác $1; \cos x; \sin x; \cos 2x; \sin 2x; \dots$ thì $P_m(x)$ là đa thức lượng giác. Bài toán được đặt lại như sau:

Tìm bảng số

$$y_i \approx f(x_i) \quad i = \overline{1, n}; \quad (5.33)$$

Hãy tìm hàm

$$\varphi(x) = P_m(x) = \sum_{k=0}^m a_k \varphi_k(x) \quad (5.34)$$

sao cho

$$\phi(a_0, a_1, \dots, a_m) = \sum_{i=1}^n \left[y_i - \sum_{k=0}^m a_k \varphi_k(x_i) \right]^2 \text{ đạt min} \quad (5.35)$$

trên $[a, b]$ theo phân hoạch

$$\Delta = \{a \equiv x_1 < x_2 < \dots < x_n \equiv b\}.$$

Xem $\phi(a_0, a_1, \dots, a_m)$ là hàm số của $m+1$ biến độc lập a_k ($k = \overline{0, m}$). Theo lý thuyết của hàm số nhiều biến, cực trị của hàm (5.35) đạt được khi

$$\frac{\partial \phi}{\partial a_k} = 0, \quad k = \overline{0, m} \quad (5.36)$$

hay

$$\left\{ \sum_{i=1}^n \left[y_i - \sum_{k=0}^m a_k \varphi_k(x_i) \right] \right\} \varphi_0(x_i) = 0$$

$$\left\{ \sum_{i=1}^n \left[y_i - \sum_{k=0}^m a_k \varphi_k(x_i) \right] \right\} \varphi_k(x_i) = 0$$

$$\left\{ \sum_{i=1}^n \left[y_i - \sum_{k=0}^m a_k \varphi_k(x_i) \right] \right\} \varphi_m(x_i) = 0$$

hay

$$\begin{aligned} \sum_{i=1}^n \varphi_0(x_i) \varphi_k(x_i) a_0 + \sum_{i=1}^n \varphi_1(x_i) \varphi_k(x_i) a_1 + \cdots + \sum_{i=1}^n \varphi_m(x_i) \varphi_k(x_i) a_m \\ = \sum_{i=1}^n y_i \varphi_k(x_i), \quad k = \overline{0, m}. \end{aligned} \quad (5.37)$$

Ký hiệu

$$\varphi_r = (\varphi_r(x_1), \varphi_r(x_2), \dots, \varphi_r(x_n))^t$$

là vector n chiều, có thành phần thứ i là $\varphi_r(x_i)$ ($i = \overline{1, n}$) và vector

$$y = (y_1, y_2, \dots, y_n)^t$$

cũng là vector n chiều. Ta định nghĩa tích vô hướng

$$\begin{aligned} [\varphi_r, \varphi_s] &= \sum_{i=1}^n \varphi_r(x_i) \varphi_s(x_i); \\ [y, \varphi_r] &= \sum_{i=1}^n y_i \varphi_r(x_i). \end{aligned}$$

Khi đó, hệ (5.37) được viết lại dưới dạng

$$\left\{ \begin{array}{l} [\varphi_0, \varphi_0] a_0 + [\varphi_1, \varphi_0] a_1 + \cdots + [\varphi_m, \varphi_0] a_m = [y, \varphi_0] \\ [\varphi_0, \varphi_1] a_0 + [\varphi_1, \varphi_1] a_1 + \cdots + [\varphi_m, \varphi_1] a_m = [y, \varphi_1] \\ \cdots \cdots \\ [\varphi_0, \varphi_m] a_0 + [\varphi_1, \varphi_m] a_1 + \cdots + [\varphi_m, \varphi_m] a_m = [y, \varphi_m] \end{array} \right. \quad (5.38)$$

Hệ (5.38) là hệ đại số tuyến tính $m+1$ phương trình, $m+1$ ẩn là a_0, a_1, \dots, a_m và được gọi là hệ phương trình chuẩn.

Định thức của hệ (5.38) xác định bởi

$$G[\varphi_0, \varphi_1, \dots, \varphi_m] = \begin{vmatrix} [\varphi_0, \varphi_0] & [\varphi_1, \varphi_0] & \dots & [\varphi_m, \varphi_0] \\ [\varphi_0, \varphi_1] & [\varphi_1, \varphi_1] & \dots & [\varphi_m, \varphi_1] \\ \vdots & \vdots & \vdots & \vdots \\ [\varphi_m, \varphi_1] & [\varphi_m, \varphi_m] & \dots & [\varphi_m, \varphi_m] \end{vmatrix} \quad (5.39)$$

được gọi là định thức Gram của hệ vector $\varphi_0, \varphi_1, \dots, \varphi_m$. Từ (5.39) ta thấy ma trận Gram đối xứng. Nếu định thức Gram khác không thì hệ (5.38) tồn tại duy nhất nghiệm.

Người ta chứng minh được rằng: nếu hệ vector (hàm) $\{\varphi_k\}_{k=0, m}$ độc lập tuyến tính trên phân hoạch

$$\Delta = [a \equiv x_1 < x_2 < \dots < x_n \equiv b]$$

thì định thức Gram khác không và do đó hệ (5.38) tồn tại duy nhất nghiệm, đồng thời hàm số $\phi(a_0, a_1, \dots, a_m)$ đạt cực tiểu tại điểm $(\bar{a}_1, \bar{a}_2, \dots, \bar{a}_m) \in \mathbb{R}^{m+1}$ là nghiệm của hệ (5.38) và ta có hàm

$$\bar{\varphi}(x) = \sum_{j=0}^m \bar{a}_j \varphi_j(x) \quad (5.40)$$

xấp xỉ hàm $f(x)$ (tức là $f(x) \approx \bar{\varphi}(x)$) tốt nhất theo nghĩa trung bình phương.

5.4 Sai số của phương pháp

Ta gọi

$$\bar{\sigma}_m = \left(\frac{1}{n} \sum_{i=1}^n [y_i - \bar{\varphi}(x_i)]^2 \right)^{1/2} \quad (5.41)$$

là sai số của phương pháp xấp xỉ trung bình phương.

Xét

$$\sum_{i=1}^n [y_i - \bar{\varphi}(x_i)]^2 = [y - \bar{\varphi}, y - \bar{\varphi}] = [y - \bar{\varphi}, y] - [y - \bar{\varphi}, \bar{\varphi}] . \quad (5.42)$$

Do \bar{a}_k là nghiệm của hệ (5.38) nên ta có

$$\begin{aligned} [y - \bar{\varphi}, \bar{\varphi}] &= [y, \bar{\varphi}] - [\bar{\varphi}, \bar{\varphi}] = \left[y, \sum_{j=0}^m \bar{a}_j \varphi_j \right] - \left[\sum_{j=0}^m \bar{a}_j \varphi_j, \sum_{k=0}^m \bar{a}_k \varphi_k \right] \\ &= \sum_{j=0}^m \bar{a}_j [y, \varphi_j] - \sum_{j=0}^m \bar{a}_j \left[\varphi_j, \sum_{k=0}^m \bar{a}_k \varphi_k \right] \\ &= \sum_{j=0}^m \bar{a}_j \left\{ [y, \varphi_j] - \sum_{k=0}^m \bar{a}_k [\varphi_j, \varphi_k] \right\} = 0. \end{aligned}$$

Vậy từ (5.42) ta có

$$\sum_{i=1}^n [y_i - \bar{\varphi}(x_i)]^2 = [y - \bar{\varphi}, y] = [y, y] - [\bar{\varphi}, y].$$

Từ đó, theo (5.41) ta có công thức đánh giá sai số:

$$\begin{aligned} \bar{\sigma}_m &= \left(\frac{1}{n} \sum_{i=1}^n [y_i - \bar{\varphi}(x_i)]^2 \right)^{1/2} = \left(\frac{1}{n} \{ [y, y] - [\bar{\varphi}, y] \} \right)^{1/2} \\ &= \left(\frac{1}{n} \left\{ [y, y] - \sum_{k=0}^m \bar{a}_k [\varphi_k, y] \right\} \right)^{1/2}, \end{aligned} \quad (5.43)$$

trong đó \bar{a}_k ($k = \overline{0, m}$) là nghiệm của hệ phương trình chuẩn (5.38) còn $\{\varphi_k(x)\}_{k=\overline{0, m}}$ là hệ hàm độc lập tuyến tính trên phân hoạch $\Delta = \{a \equiv x_1 < x_2 < \dots < x_n \equiv b\}$ của đoạn $[a, b]$.

5.5 Trường hợp $\{\varphi_k(x)\}_{k=\overline{0, m}}$ là hệ trực chuẩn

Định nghĩa 5.3. (hệ trực giao) Hệ hàm $\{\varphi_j(x)\}_{j=\overline{0, m}}$ được gọi là hệ trực giao trên phân hoạch Δ của đoạn $[a, b]$ nếu

$$[\varphi_r, \varphi_s] = \sum_{i=1}^n \varphi_r(x_i) \varphi_s(x_i) \begin{cases} = 0, & \text{nếu } r \neq s \\ \neq 0, & \text{nếu } r = s. \end{cases}$$

Số

$$\|\varphi_r\| = ([\varphi_r, \varphi_r])^{1/2}$$

gọi là chuẩn của φ_r trên phân hoạch Δ .

Hệ hàm $\{\varphi_j(x)\}_{j=\overline{0, m}}$ gọi là hệ trực chuẩn nếu nó là hệ trực giao và các vector đều có chuẩn bằng "1", tức là $\|\varphi_r\| = 1$, ($r = \overline{0, m}$).

Vậy nếu chọn $\{\varphi_r\}_{r=0,\overline{m}}$ là hệ trực giao thì từ hệ phương trình chuẩn (5.38) ta có ngay

$$\bar{a}_k = \frac{[y, \varphi_k]}{[\varphi_k, \varphi_k]}, \quad k = \overline{0, m}. \quad (5.44)$$

Thay \bar{a}_k ($k = \overline{0, m}$) vào (5.43) ta được

$$\bar{\sigma}_m = \left(\frac{1}{n} \left\{ \|y\|^2 - \sum_{k=0}^m \frac{[y, \varphi_k]^2}{\|\varphi_k\|^2} \right\} \right)^{1/2}. \quad (5.45)$$

Từ (5.45) ta thấy khi m tăng thì $\bar{\sigma}_m$ giảm. Vì vậy, khi tính $\bar{\sigma}_m$ có thể tính dần từng bước từ $m = 0, 1, 2, \dots$. Giả sử tới bước thứ $k = m$ ta thu được $\bar{\sigma}_k < \varepsilon$ (theo mong muốn) thì dừng quá trình tính. Còn nếu $\bar{\sigma}_k \geq \varepsilon$ thì ta tính tiếp bước thứ $m = k + 1$. Khi tính đến bước thứ $m = s$ sao cho $\bar{\sigma}_s < \varepsilon$ ($s > k$), ta dừng quá trình tính.

5.6 Trường hợp hệ cơ bản là hệ đại số

Xét hệ

$$\varphi_j(x) = x^j, \quad j = \overline{0, m} \quad (m \leq n - 1).$$

Bài toán trở thành tìm a_j , ($j = \overline{0, m}$) sao cho

$$\varphi(x) = P_m(x) = \sum_{j=0}^m a_j x^j \quad (5.46)$$

chính là đa thức bậc m (vì hệ $\{1, x, x^2, \dots, x^n\}$ là độc lập tuyến tính). Các hệ số a_j được xác định từ nghiệm của hệ phương trình chuẩn (5.38).

Trong trường hợp này ta có

$$\begin{aligned} [y, \varphi_s] &= \sum_{i=1}^n y_i \varphi_s(x_i) = \sum_{i=1}^n y_i x_i^s \\ [\varphi_r, \varphi_s] &= \sum_{i=1}^n \varphi_r(x_i) \varphi_s(x_i) = \sum_{i=1}^n x_i^{r+s}. \end{aligned}$$

Khi đó, hệ phương trình chuẩn trở thành

$$\left\{ \begin{array}{lcl} na_0 + \sum_{i=1}^n x_i a_1 + \sum_{i=1}^n x_i^2 a_2 + \cdots + \sum_{i=1}^n x_i^m a_m & & = \sum_{i=1}^n y_i \\ \sum_{i=1}^n x_i a_0 + \sum_{i=1}^n x_i^2 a_1 + \sum_{i=1}^n x_i^3 a_2 + \cdots + \sum_{i=1}^n x_i^{m+1} a_m & & = \sum_{i=1}^n y_i x_i \\ \vdots & & \\ \sum_{i=1}^n x_i^m a_0 + \sum_{i=1}^n x_i^{m+1} a_1 + \sum_{i=1}^n x_i^{m+2} a_2 + \cdots + \sum_{i=1}^n x_i^{2m} a_m & & = \sum_{i=1}^n y_i x_i^m \end{array} \right. \quad (5.47)$$

Giải hệ (5.47) ta được $\bar{a}_k | k = \overline{0, m}$. Từ đó suy ra $f(x) \approx \varphi(x) = P_m(x)$ trong đó

$$P_m(x) = \sum_{k=0}^m \bar{a}_k x^k. \quad (5.48)$$

là đa thức bậc m .

5.7 Trường hợp hệ cơ bản là hệ lượng giác

Trong thực tế, nhiều khi ta gặp hàm $f(x)$ có tính chất tuần hoàn, hàm xấp xỉ cũng cần mang tính chất đó, nghĩa là tìm $\varphi(x) \approx f(x)$ và $\varphi(x)$ có dạng tuần hoàn. Trong trường hợp này, ta chọn hàm xấp xỉ là đa thức lượng giác cấp k :

$$T_k(x) = \alpha_0 + \sum_{r=1}^k (\alpha_r \cos rx + \beta_r \sin rx) \quad (5.49)$$

trong đó $\alpha_0, \alpha_r, \beta_r$ ($r = \overline{1, k}$) là các tham số cần xác định.

Rõ ràng $T_k(x)$ có dạng của đa thức suy rộng $m = 2k$ với hệ cơ bản là

$$\varphi_0 = 1; \quad \varphi_1(x) = \cos x; \quad \varphi_2(x) = \sin x, \dots$$

$$\varphi_{2r-1}(x) = \cos rx; \quad \varphi_{2r}(x) = \sin rx, \dots$$

$$\varphi_{2k-1}(x) = \cos kx; \quad \varphi_{2k}(x) = \sin kx.$$

Dễ thấy hệ trên là hệ độc lập tuyến tính. Các tích vô hướng trong hệ phương trình chuẩn (5.38) sẽ là

$$[\varphi_0, \varphi_0] = \sum_{i=1}^n 1^2 = n; \quad [y, \varphi_0] = \sum_{i=1}^n y_i;$$

$$[\varphi_0, \varphi_j] = \begin{cases} \sum_{i=1}^n \cos px_i, & j = 2p - 1 \\ \sum_{i=1}^n \sin px_i, & j = 2p \end{cases}, \quad p = \overline{1, k}; \quad j = \overline{1, 2k}$$

$$[\varphi_\ell, \varphi_j] = \begin{cases} \sum_{i=1}^n \cos rx_i \cos px_i, & \ell = 2r - 1; \quad j = 2p - 1 \\ \sum_{i=1}^n \sin rx_i \cos px_i, & \ell = 2r; \quad j = 2p - 1 \\ \sum_{i=1}^n \sin rx_i \sin px_i, & \ell = 2r; \quad j = 2p \end{cases}, \quad j = \overline{1, 2k}; \quad \ell = \overline{1, 2k}$$

$$[y, \varphi_j] = \begin{cases} \sum_{i=1}^n y_i \cos px_i, & j = 2p - 1 \\ \sum_{i=1}^n y_i \sin px_i, & j = 2p. \end{cases}$$

Từ đó ta có hệ phương trình chuẩn

$$\begin{aligned}
 n\alpha_0 + \sum_{r=1}^k \left\{ \alpha_r \sum_{i=1}^n \cos rx_i + \beta_r \sum_{i=1}^n \sin rx_i \right\} &= \sum_{i=1}^n y_i \\
 \alpha_0 \sum_{i=1}^n \cos px_i + \sum_{r=1}^k \left\{ \alpha_r \sum_{i=1}^n \cos rx_i \cos px_i + \beta_r \sum_{i=1}^n \sin rx_i \cos px_i \right\} &= \sum_{i=1}^n y_i \cos px_i \\
 \alpha_0 \sum_{i=1}^n \sin px_i + \sum_{r=1}^k \left\{ \alpha_r \sum_{i=1}^n \cos rx_i \sin px_i + \beta_r \sum_{i=1}^n \sin rx_i \sin px_i \right\} &= \sum_{i=1}^n y_i \sin px_i \\
 p = 1, 2, \dots, k.
 \end{aligned} \tag{5.50}$$

Hệ (5.50) gồm $2k+1$ phương trình, $2k+1$ ẩn là $\alpha_0, \alpha_r, \beta_r, r = \overline{1, k}$. Giải hệ trên ta thu được $\bar{\alpha}_0, \bar{\alpha}_r, \bar{\beta}_r$ và do đó ta có

$$f(x) \approx \bar{T}_k(x) = \bar{\alpha}_0 + \sum_{r=1}^k \bar{\alpha}_r \cos rx + \bar{\beta}_r \sin rx \tag{5.51}$$

Chú ý 5.4. Nếu tập hợp các mốc $x_i \in (0, 2\pi]$ cách đều nhau, khoảng cách $h = \frac{2\pi}{n}$, tức là $x_i = ih, i = \overline{1, n}$. Trong trường hợp này ta có

$$\left\{
 \begin{array}{l}
 \sum_{i=1}^n \cos rx_i = \sum_{i=1}^n \sin rx_i = 0 \\
 \sum_{i=1}^n \cos rx_i \sin px_i = 0, \quad r = \overline{1, k}, \quad p = \overline{1, k} \\
 \sum_{i=1}^n \cos rx_i \cos px_i = \sum_{i=1}^n \sin rx_i \sin px_i = 0, \quad p \neq r; \quad p, r = \overline{1, k} \\
 \sum_{i=1}^n \cos^2 rx_i = \sum_{i=1}^n \sin^2 rx_i = \frac{n}{2}, \quad r = \overline{1, k}.
 \end{array}
 \right. \tag{5.52}$$

Khi đó hệ (5.50) trở nên đơn giản hơn (chỉ còn các phần tử trên đường chéo chính và về phải của hệ).

Ví dụ 5.3. Cho bảng số $y_i \approx f(x_i)$:

x	0.56	0.84	1.14	2.44	3.16
y	-0.80	-0.97	-0.089	1.07	3.66

Hãy tìm hàm xấp xỉ theo nghĩa trung bình phương có dạng đa thức bậc hai.

Giải:

Ta tìm hàm xấp xỉ có dạng $p(x) = a_0 + a_1x + a_2x^2$. Theo hệ phương trình chuẩn (5.47) ta có:

$$\begin{cases} 5a_0 + \sum_{i=1}^5 x_i a_1 + \sum_{i=1}^5 x_i^2 a_2 &= \sum_{i=1}^5 y_i \\ \sum_{i=1}^5 x_i a_0 + \sum_{i=1}^5 x_i^2 a_1 + \sum_{i=1}^5 x_i^3 a_2 &= \sum_{i=1}^5 x_i y_i \\ \sum_{i=1}^5 x_i^2 a_0 + \sum_{i=1}^5 x_i^3 a_1 + \sum_{i=1}^5 x_i^4 a_2 &= \sum_{i=1}^5 x_i^2 y_i \end{cases}$$

Lập bảng tính các hệ số

	x_i	y_i	x_i^2	x_i^3	x_i^4	$x_i y_i$	$x_i^2 y_i$
	0.56	-0.80	0.31	0.18	0.09	-0.45	-0.25
	0.84	-0.97	0.71	0.59	0.50	-0.81	-0.68
	1.14	-0.98	1.30	1.48	1.69	-1.12	-1.27
	2.44	1.07	5.95	14.53	35.45	2.61	6.37
	3.16	3.66	9.99	31.55	99.71	11.57	36.55
Σ	8.14	1.98	18.260	48.335	137.456	11.791	40.710

Từ đó ta được hệ

$$\begin{cases} 5a_0 + 8.14a_1 + 18.260a_2 &= 1.98 \\ 8.14a_0 + 18.260a_1 + 48.335a_2 &= 11.791 \\ 18.260a_0 + 48.335a_1 + 137.456a_2 &= 40.710 \end{cases}$$

Giải hệ trên ta thu được $a_0 = 0$; $a_1 = -2$; $a_2 = 1$, từ đó

$$f(x) \approx p(x) = x^2 - 2x.$$

Sai số được đánh giá theo công thức (5.43):

$$\bar{\sigma}_2 = \left(\frac{1}{5} \sum_{i=0}^2 [y_i - p(x_i)]^2 \right)^{1/2} = \left(\frac{1}{5} \times 0.00004 \right)^{1/2} = 0.0063.$$

Ví dụ 5.4. Cho bảng số

x	30°	60°	90°	120°	150°	180°	210°	240°	270°
y	2.611	3.102	2.912	2.105	0.612	-1.321	-1.906	-2.412	-2.802

300°	330°	360°
-2.703	-1.610	1.500

Hãy tìm đa thức lượng giác cấp hai theo phương pháp bình phương tối thiểu.

Giải:

Đa thức lượng giác cấp hai có dạng

$$T_2(x) = \alpha_0 + \alpha_1 \cos x + \alpha_2 \cos 2x + \beta_1 \sin x + \beta_2 \sin 2x.$$

Các mốc $x_i \in (0, 2\pi]$ cách đều nhau, có bước $h = \frac{2\pi}{12} = 30^\circ$.

Từ hệ phương trình chuẩn (5.50) và công thức (5.52) ta có hệ

$$\begin{cases} 12\alpha_0 &= \sum_{i=1}^{12} y_i \\ \alpha_1 \sum_{i=1}^{12} \cos^2 x_i &= \sum_{i=1}^{12} y_i \cos x_i \\ \alpha_2 \sum_{i=1}^{12} \cos^2 2x_i &= \sum_{i=1}^{12} y_i \cos 2x_i \\ \beta_1 \sum_{i=1}^{12} \sin^2 x_i &= \sum_{i=1}^{12} y_i \sin x_i \\ \beta_2 \sum_{i=1}^{12} \sin^2 2x_i &= \sum_{i=1}^{12} y_i \sin 2x_i \end{cases}$$

Lập bảng tính toán các hệ số, sau đó giải hệ trên ta thu được đa thức lượng giác

$$T_2(x) = 0.007 + 0.860 \cos x + 3.004 \sin x - 0.021 \cos 2x + 0.432 \sin 2x.$$

5.8 Chương trình MATLAB

Bạn đọc hãy dựa theo ví dụ để lập chương trình MatLab cho phần này.

§6. TÌM HÀM THỰC NGHIỆM THEO PHƯƠNG PHÁP BÌNH PHƯƠNG TỐI THIỂU

Phương pháp bình phương tối thiểu thường được dùng để tìm hàm thực nghiệm.

Giả sử cần tìm quan hệ hàm số $y = f(x)$ theo quy luật đã được rút ra từ

quá trình thực tiễn của công việc. Bằng thí nghiệm, do đặc ta thu được bảng số

$$y_i \approx f(x_i), \quad i = \overline{1, n}.$$

Từ bảng số đó qui về việc đi tìm các tham số của các dạng hàm đơn giản (đã biết), gọi là các hàm số thực nghiệm. Các dạng hàm thực nghiệm thường gặp:

1. $y = a + bx;$
2. $y = a + bx + cx^2;$
3. $y = a + bx + cx^2 + dx^3;$
4. $y = a + b \cos x + c \sin x;$
5. $y = ae^{bx} \quad (a > 0);$
6. $y = ax^b \quad (a > 0);$

Trong ba dạng 1, 2 và 3 thì hệ cơ bản chính là hệ đại số đã xét trong mục 5.6. Với dạng 4, hệ cơ bản là hệ lượng giác đã được xét trong mục 5.7. Các dạng 5, 6 có dạng phi tuyến đối với các tham số a, b cần tìm.

6.1 Hàm thực nghiệm dạng $y = ae^{bx}, \quad (a > 0)$

Vấn đề đặt ra: Từ bảng số $y_i \approx f(x_i), \quad i = \overline{1, n}$; hãy tìm hàm thực nghiệm trong dạng $y = ae^{bx}$.

Do hàm số có dạng phi tuyến đối với các tham số a và b nên cần thiết tìm cách biến đổi để đưa về dạng tuyến tính.

Từ

$$y = ae^{bx}. \quad (5.53)$$

lấy lôgarít hai vế ta được

$$\lg y = \lg a + bx \lg e. \quad (5.54)$$

Đặt

$$Y = \lg y; \quad A = \lg a; \quad B = b \lg e; \quad X = x.$$

Từ (5.54) ta suy ra

$$Y = A + BX \quad (5.55)$$

là dạng tuyến tính đối với A và B và bài toán được đặt lại như sau:

Tìm A và B , xuất phát từ bảng số

$$Y_i \approx \lg y_i = \lg(f(x_i)) = \lg f(X_i) = F(X_i), \text{ với } X_i = x_i, \quad i = \overline{1, n}. \quad (5.56)$$

Cách tìm A và B bằng phương pháp bình phương tối thiểu chính là đang 1 đã xét trong mục 5.6. Từ đó suy ra

$$a = 10^A; \quad b = \frac{B}{\lg e}, \quad (\lg e \approx 0.43429).$$

6.2 Hàm thực nghiệm dạng $y = ax^b$. ($a > 0, x > 0$)

Hoàn toàn tương tự như trên, lôgarít hai vế ta thu được

$$\lg y = \lg a + b \lg x.$$

Bằng cách đổi biến

$$Y = \lg y; \quad A = \lg a; \quad B = b; \quad X = \lg x,$$

ta thu được bảng số $Y_i = \lg y_i, \quad X_i = \lg x_i$ hay

$$Y_i \approx F(X_i), \quad i = \overline{1, n}. \quad (5.57)$$

Từ (5.57), tìm hàm thực nghiệm dưới dạng:

$$Y = A + BX. \quad (5.58)$$

Bằng phương pháp bình phương tối thiểu ta thu được A, B và từ đó suy ra a, b .

Chú ý 5.5. Cả hai trường hợp trên đều dẫn đến dạng tuyến tính, cho nên khi tìm A, B xuất phát từ bảng số X và Y , nghĩa là các điểm $M_i(x_i, \lg y_i)$ được phân bố gần như nằm trên một đường thẳng thì quan hệ giữa x và y có dạng $y = ae^{bx}$. Tương tự, nếu các điểm $(\lg x_i, \lg y_i)$ có phân bố gần như trên đường thẳng thì quan hệ giữa y và x có dạng $y = ax^b$.

Chú ý 5.6. Các trường hợp tìm dạng hàm xấp xỉ trong dạng đa thức với hệ cơ sở đại số hay lượng giác hoặc các dạng phi tuyến có khả năng tuyến tính hóa được là do các định luật vật lý, hóa học gợi ra. Trường hợp không thể biết gì về quan hệ giữa y và x , khi đó ta phải làm thí nghiệm nhiều lần để loại bỏ các kết quả vô lý, sau đó biểu diễn các cặp số (x_i, y_i) thành những điểm M_i trong mặt phẳng tọa độ Oxy , ta suy ra dạng hợp lý nhất của quan hệ hàm số giữa x và y .

Để minh họa cho các chú ý trên, ta xét ví dụ sau.

Ví dụ 5.5. Cho bảng số

t	14.5	30.0	64.5	74.5	86.7	94.5	98.9
k	0	0.004	0.018	0.029	0.051	0.073	0.090

có được từ mối liên hệ của suất dẫn điện k của thủy tinh phu thuộc vào nhiệt độ t (tính theo $^{\circ}\text{C}$). Hãy tìm công thức thực nghiệm của hàm số $k = f(t)$.

Giải:

Ta đã biết quy luật suất dẫn điện k của thủy tinh không thể biểu diễn trong dạng đa thức hay lượng giác. Các điểm $M_i(t_i, \lg k_i)$, $i = \overline{1, 7}$, trừ điểm đầu tiên, nói chung nằm trên một đường thẳng. Cụ thể ta có bảng số sau:

t	14.5	30.0	64.5	74.5	86.7	94.5	98.9
$\lg k$	-	-2.3979	-1.7447	-1.5376	-1.2924	-1.1367	-1.0458

Vậy quan hệ có dạng $k = ae^{bt}$ hay

$$\lg k = \lg a + bt \lg e.$$

Đặt $Y = \lg k$, $A = \lg a$, $B = b \lg e$ ta được

$$Y = A + Bt.$$

Giải hệ phương trình chuẩn theo (5.47) ta thu được $A = 0.00335$; $B = 0.01965$, từ đó suy ra

$$\begin{aligned} a &= 10^{a_0} = 10^{0.00335} = 1.00774, \\ b &= \frac{B}{\lg e} = \frac{0.01965}{0.43429} = 0.04525. \end{aligned}$$

Tóm lại ta có:

$$k = 1.00774 e^{0.04525t}$$

6.3 Chương trình MATLAB

Bạn đọc hãy dựa theo ví dụ để lập chương trình MatLab cho phần này.

§7. BÀI TẬP

Bài tập 5.1. Thành lập đa thức nội suy Lagrange từ bảng số sau:

x	2	4	6	8	10
$y = f(x)$	0	3	5	4	1

Bài tập 5.2. Tìm đa thức nội suy bậc hai của hàm $y = 3^x$ trên $[-1, 1]$, từ đó suy ra giá trị gần đúng của $\sqrt{3}$.

Bài tập 5.3. Tính gần đúng giá trị $f(323.5)$ bằng đa thức nội suy Lagrange từ bảng số sau:

x	321.0	322.8	324.2	325.0
$y = f(x)$	2.50651	2.50893	2.51081	2.51188

Bài tập 5.4. Xây dựng đa thức nội suy Lagrange trong trường hợp các mốc nội suy cách đều.

Bài tập 5.5. Áp dụng kết quả bài 5.4, tính giá trị hàm $y = \cos x$ tại điểm $x = 5.437$ từ bảng số sau

x	5.0	5.1	5.2	5.3
$y = \cos x$	0.283662185	0.377977743	0.468516671	0.554374336
5.4	5.5	5.6	5.7	
0.63469287	0.70866977	0.77556587	0.83471278	

Bài tập 5.6. Khi tính $\sqrt{115}$ bằng đa thức nội suy Lagrange đối với hàm $y = \sqrt{x}$ tại các mốc $x_0 = 100$; $x_1 = 121$; $x_2 = 144$ thì đạt sai số là bao nhiêu?

Bài tập 5.7. Cho giá trị của hàm số

$$y = \arctan \frac{3x - x^3}{1 - 3x^2} - 3 \arctan x + \frac{x^2}{4} (2 \ln x - 3)$$

trong dạng bảng số sau:

x	58	58.34	58.68	59.02	59.36	59.7
y	4303.52	4364.11	4425.17	4486.69	4548.69	4611.16

Xây dựng công thức nội suy Newton tiến và tính gần đúng giá trị của y tại $x = 58.17$.

Bài tập 5.8. Từ bảng số

x	6.3	6.72	7.14	7.56	7.98	8.4
y	21.4259	23.377	25.3622	27.3831	29.438	31.5253

Hãy xây dựng đa thức nội suy để tính giá trị gần đúng của hàm tại điểm $x = 6.51$.

Bài tập 5.9. Hãy xây dựng đa thức nội suy xấp xỉ hàm $y = e^x$ trên đoạn $[3.5; 3.7]$ được cho từ bảng số sau:

x	3.50	3.55	3.60	3.65	3.70
$y = e^x$	33.115	34.813	36.598	38.475	40.447

Bài tập 5.10. Giá trị của tích phân xác suất

$$y = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$

được cho trong bảng sau

x	0.46	0.47	0.48	0.49
y	0.4816555	0.4937452	0.5027498	0.5116683

Hãy tìm x sao cho $y = \frac{1}{2}$ (nội suy ngược) bằng đa thức nội suy Lagrange.

Bài tập 5.11. Cho bảng giá trị của tích phân xác suất (trong bài tập 5.10)

x	0.45	0.46	0.47	0.48	0.49	0.50
y	0.475818	0.4846555	0.4937452	0.5027498	0.5116683	0.52049999

Tìm x sao cho $y = \frac{1}{2}$ bằng công thức lặp.

Bài tập 5.12. Sử dụng các giá trị của hàm $y = \lg x$ cho bởi

x	20	25	30
$y = \lg x$	1.3010	1.3979	1.4771

tìm x sao cho $y = 1.35$ theo công thức nội suy Lagrange và công thức lặp.

Bài tập 5.13. Cho bảng giá trị của hàm Betxen $J_0(x)$:

x	2.4	2.5	2.6
y	0.0025	-0.0484	-0.0968

Tìm nghiệm của phương trình $J_0(x) = 0$ trong khoảng $(2.4; 2.6)$ có độ chính xác đến 10^{-3} .

Bài tập 5.14. Tìm hàm thực nghiệm dạng

$$F(x) = a_0x^2 + a_1x + a_2$$

bằng phương pháp bình phương tối thiểu từ các bảng số sau

1.	x	7	8	9	10	11	12	13
	y	3.1	4.9	5.3	5.8	6.1	6.1	5.9

2.	x	0.78	1.56	2.34	3.12	3.81
	y	2.50	1.20	1.12	2.25	4.28

3.	x	-3	-2	-1	0	1	2	3
	y	-0.71	4.9	5.3	5.8	6.1	6.1	5.9

Bài tập 5.15. Tìm hàm thực nghiệm dạng $S = ae^{bx}$ bằng phương pháp bình phương tối thiểu từ bảng số

x	0	2	4	6	8	10	12
S	1280	635	324	162	76	43	19

Bài tập 5.16. Tìm hàm thực nghiệm dạng $S = ax^b$ từ bảng số

x	1	2	3	4	5
y	7.1	15.2	48.1	96.3	150.1

CHƯƠNG 6

ĐẠO HÀM, TÍCH PHÂN VÀ PHƯƠNG TRÌNH VI PHÂN

§1. TÍNH ĐẠO HÀM

NHÌỀU bài toán kỹ thuật đòi hỏi phải tìm đạo hàm của hàm số tại điểm nào đó. Trường hợp biết $y = f(x)$ nhưng $f(x)$ có dạng là biểu thức khá phức tạp thì việc tính $y'(\bar{x}) = f'(\bar{x})$ cũng gặp nhiều khó khăn. Cũng có trường hợp ta chỉ có bảng số $y_i = f(x_i)$, $i = \overline{0, n}$ (chẳng hạn, cần tìm vận tốc, gia tốc của chuyển động $s = s(t)$ nào đó, từ bảng số $s_i = s(t_i)$, $i = \overline{0, n}$). Khi đó việc tính chính xác $y'(\bar{x})$ là vô nghĩa. Để khắc phục khó khăn trên, người ta thường sử dụng đa thức nội suy.

1.1 Tính đạo hàm nhờ đa thức nội suy Lagrange

Từ bảng số

$$y_j = f(x_j), \quad j = \overline{0, n}, \quad (6.1)$$

theo công thức nội suy Lagrange trong tiết 2, chương 5 ta có

$$f(x) \approx P_n(x) = \sum_{j=0}^n L_j(x)y_j, \quad (6.2)$$

trong đó

$$L_j(x) = \frac{(x - x_0)(x - x_1) \dots (x - x_{j-1})(x - x_{j+1}) \dots (x - x_n)}{(x_j - x_0)(x_j - x_1) \dots (x_j - x_{j-1})(x_j - x_{j+1}) \dots (x_j - x_n)}, \quad j = \overline{0, n}. \quad (6.3)$$

Từ đó ta suy ra

$$f'(x) \approx P'_n(x) = \sum_{j=0}^n L'_j(x)y_j. \quad (6.4)$$

Sai số

$$R'_n(x) = f'(x) - P'_n(x). \quad (6.5)$$

Từ (6.4) ta cũng có thể tính các đạo hàm cấp cao $f''(x), \dots$

1.2 Trường hợp các mốc nội suy cách đều nhau

Theo công thức nội suy Newton tiên (5.19) ta có

$$\begin{aligned} f(x) &\approx P_n(x) = P_n(x_0 + ht) = \mathcal{P}(t) \\ &= y_0 + \frac{t}{1!}\Delta y_0 + \frac{t(t-1)}{2!}\Delta^2 y_0 + \dots + \frac{t(t-1)\cdots(t-n+1)}{n!}\Delta^n y_0, \end{aligned} \quad (6.6)$$

trong đó

$$h = x_{i+1} - x_i \quad \forall i = \overline{0, n-1}; \quad x = x_0 + ht.$$

Vậy

$$\begin{aligned} f'(x) &\approx P'_n(x) = \mathcal{P}'_t \cdot t'_x = \frac{1}{h}\mathcal{P}'(t) \\ &= \frac{1}{h}[\Delta y_0 + \frac{2t-1}{2}\Delta^2 y_0 + \dots]. \end{aligned} \quad (6.7)$$

Sau đây, ta xét một số trường hợp riêng.

1.3 Trường hợp $n = 2$

Trong trường hợp này ta có 3 mốc nội suy x_0, x_1, x_2 và các giá trị hàm tương ứng là $y_i = f(x_i)$, $i = 0, 1, 2$.

Từ (6.7) ta có

$$\begin{aligned} f'(x) &\approx P'_2(x) = \frac{1}{h}[\Delta y_0 + \frac{2t-1}{2}\Delta^2 y_0] \\ &= \frac{1}{h}[(y_1 - y_0) - (t - \frac{1}{2})(y_0 - 2y_1 + y_2)], \end{aligned} \quad (6.8)$$

trong đó $t = \frac{x - x_0}{h}$.

Đặc biệt, khi cần tính đạo hàm tại các mốc nội suy x_0, x_1, x_2 :

1. Khi $x = x_0 \Rightarrow t = 0$, thay vào (6.8) ta được

$$f'(x_0) \approx \frac{1}{2h} (-3y_0 + 4y_1 - y_2). \quad (6.9)$$

2. Khi $x = x_1 \Rightarrow t = 1$ thì

$$f'(x_1) \approx \frac{1}{2h} (-y_0 + y_2). \quad (6.10)$$

3. Khi $x = x_2 \Rightarrow t = 2$ và ta có

$$f'(x_2) \approx \frac{1}{2h} (y_0 - 4y_1 + 3y_2). \quad (6.11)$$

Từ công thức khai triển Taylor, các công thức (6.9) - (6.11) đạt sai số cấp $O(h^2)$.

1.4 Trường hợp $n = 3$

Ta có 4 mốc nội suy là x_0, x_1, x_2, x_3 . Theo (6.7) ta được

$$f'(x) \approx P'_3(x) = \frac{1}{h} [\Delta y_0 + \frac{2t-1}{2} \Delta^2 y_0 + \frac{3t^2-6t+2}{6} \Delta^3 y_0] \quad (6.12)$$

với $t = \frac{x - x_0}{h}$.

1. Khi $x = x_0 \Rightarrow t = 0$ và ta có

$$f'(x_0) \approx P'_3(x_0) = \frac{1}{6h} (-11y_0 + 18y_1 - 9y_2 + 2y_3) \quad (6.13)$$

2. Khi $x = x_1 \Rightarrow t = 1$, ta có

$$f'(x_1) \approx P'_3(x_1) = \frac{1}{6h} (-2y_0 - 3y_1 + 6y_2 - y_3) \quad (6.14)$$

3. Khi $x = x_2 \Rightarrow t = 2$, ta có

$$f'(x_2) \approx P'_3(x_2) = \frac{1}{6h} (y_0 - 6y_1 + 3y_2 + 2y_3) \quad (6.15)$$

4. Khi $x = x_3 \Rightarrow t = 3$, ta có

$$f'(x_3) \approx P'_3(x_3) = \frac{1}{6h} (-2y_0 + 9y_1 - 18y_2 + 11y_3) \quad (6.16)$$

Trong trường hợp này sai số đạt cấp $O(h^3)$.

Chú ý 6.1. Nếu lấy số mốc nội suy là $5, 6, \dots$, theo công thức (6.7) ta có công thức tính đạo hàm đạt sai số cấp cao hơn, song công thức sẽ phức tạp hơn. Tùy theo công việc cần thiết mà chọn công thức thích hợp.

Chú ý 6.2. Công thức (6.7) xuất phát từ công thức nội suy Newton tiến, vì vậy nó thuận lợi khi tính đạo hàm tại điểm $\bar{x} \geq x_0$, gần x_0 (ở đầu bảng sai phân).

Chú ý 6.3. Nếu xuất phát từ công thức nội suy Newton lùi (5.21), ta cũng có công thức tính đạo hàm tương ứng. Các công thức đó sẽ thích hợp cho việc tính đạo hàm tại $\bar{x} \leq x_n$ và gần với x_n (cuối bảng sai phân).

Chú ý 6.4. Khi cần tính đạo hàm tại điểm \bar{x} gần x_i (giữa bảng sai phân), ta xét hai khả năng. Với $\bar{x} \geq x_i$, ta sử dụng công thức (6.7) và xem x_i là x_0 . Còn nếu $\bar{x} \leq x_i$, ta có thể sử dụng công thức tính theo chú ý 6.3 và xem x_i là x_n .

Ví dụ 6.1. Tính gần đúng $f'(55)$ của hàm $f(x) = \lg x$ cho từ bảng số

x	50	55	60	65
$f(x) = \lg x$	1.6990	1.7401	1.7782	1.8129

Giải

Ở đây có bốn mốc nội suy là $x_0 = 50 \rightarrow x_4 = 65$; $h = 5$; $n = 3$ và ta cần tính $f'(x_1)$ với $x_1 = 55$. Theo công thức (6.14) ta có

$$f'(55) \approx \frac{1}{30} (-2 \times 1.6990 - 3 \times 1.7401 + 6 \times 1.7782 - 1.8129) = 0.007933.$$

Bằng cách tính trực tiếp ta được $f'(x) = (\lg x)' = \frac{1}{x \ln 10}$, suy ra

$$f'(55) = 0.18182 \times \frac{1}{\ln 10} = 0.007896.$$

Hai kết quả trên trùng nhau tới ba số lẻ sau dấu ":".

§2. TÍNH GẦN ĐÚNG TÍCH PHÂN XÁC ĐỊNH

2.1 Mở đầu

Giả sử $f(x)$ là hàm số liên tục trên đoạn $[a, b]$ có nguyên hàm là $F(x)$, khi đó giá trị của tích phân xác định trên đoạn $[a, b]$ được tính theo công thức Newton-Leibniz:

$$I = \int_a^b f(x)dx = F(x) \Big|_a^b = F(b) - F(a). \quad (6.17)$$

Tuy nhiên trong nhiều trường hợp có $F(x)$ là nguyên hàm của hàm $f(x)$ không phải là hàm số sơ cấp hoặc $F(x)$ có biểu thức quá phức tạp. Khi đó giá trị của tích phân I chỉ tìm được là số gần đúng. Trong trường hợp hàm số cho trong dạng bảng số thì khái niệm nguyên hàm không còn ý nghĩa gì nữa.

Bài toán tìm giá trị của tích phân xác định trong toán học tính toán nghĩa là giá trị I đó được tìm thông qua giá trị của hàm số $f(x)$ dưới dấu tích phân tại một số điểm thuộc đoạn $[a, b]$. Phương pháp đơn giản nhất để giải quyết bài toán trên là dựa vào ý nghĩa hình học của tích phân xác định và từ bảng số $y_i = f(x_i)$, $i = \overline{0, n}$, ta xây dựng đa thức nội suy $\varphi(x)$.

Từ đó ta có

$$I = \int_a^b f(x)dx \approx \int_a^b \varphi(x)dx. \quad (6.18)$$

Sau đây ta xét một số dạng công thức mà tính toán khá thuận lợi trong thực tế.

2.2 Công thức hình thang

2.2.1. Trường hợp đoạn $[a, b]$ gồm hai mốc nội suy

Giả sử $x_0 = a$, $x_1 = b$; $h = b - a$, khi đó

$$y_0 = f(x_0) = f(a); \quad y_1 = f(x_1) = f(b).$$

Áp dụng công thức nội suy Lagrange ta có

$$\begin{aligned} f(x) &\approx P_1(x) = L_0(x)y_0 + L_1(x)y_1 \\ &= \frac{x - x_1}{x_0 - x_1}y_0 + \frac{x - x_0}{x_1 - x_0}y_1. \end{aligned}$$

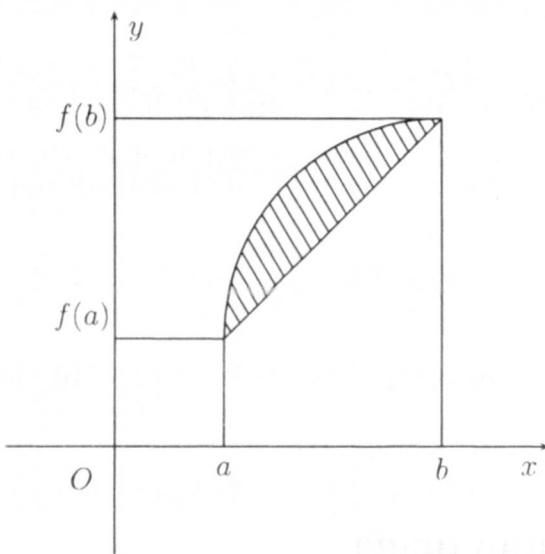
Từ đó suy ra

$$\begin{aligned} I &= \int_a^b f(x)dx \approx \int_{x_0}^{x_1} [\frac{x - x_1}{x_0 - x_1}y_0 + \frac{x - x_0}{x_1 - x_0}y_1]dx \\ &= -\frac{y_0}{2(x_0 - x_1)}(x_0 - x_1)^2 + \frac{y_1}{2(x_1 - x_0)}(x_1 - x_0)^2 \\ &= \frac{1}{2}(x_1 - x_0)(y_0 + y_1), \end{aligned}$$

hay

$$I = \int_a^b f(x)dx \approx \frac{b-a}{2}[f(a) + f(b)]. \quad (6.19)$$

Về mặt hình học, công thức (6.19) chính là công thức hình thang (hình 6.1).



Hình 6.1

2.2.2. Sai số

Gọi

$$I^* = \frac{b-a}{2}[f(a) + f(b)]$$

thì sai số

$$R = I - I^* = \int_a^b f(x)dx - \frac{b-a}{2} [f(a) + f(b)].$$

Người ta đã chứng minh được rằng, nếu $f(x)$ là hàm số liên tục và có đạo hàm liên tục đến cấp hai trên đoạn $[a, b]$ đồng thời $M_2 = \max_{x \in [a, b]} |f''(x)|$ thì

$$I = I^* + \frac{f''(c)}{12} (b-a)^3, \quad c \in (a, b).$$

Khi đó ta có

$$|R| = |I - I^*| \leq \frac{M_2}{12} (b-a)^3. \quad (6.20)$$

2.2.3. Công thức hình thang tổng quát

Từ công thức (6.20) ta thấy, nếu $h = b-a$ lớn thì sai số sẽ lớn, nên để được sai số bé theo mong muốn thì h phải bé. Để khắc phục điều đó, người ta dựa vào tính chất khả tổng của tích phân xác định, chia đoạn $[a, b]$ thành n phần đều nhau có khoảng cách là $h = \frac{b-a}{n}$ bởi các điểm chia

$$a \equiv x_0 < x_1 < \cdots < x_n \equiv b, \quad h = x_{i+1} - x_i, \quad i = \overline{0, n-1}.$$

Vậy

$$\begin{aligned} I &= \int_a^b f(x)dx = \int_{x_0}^{x_n} f(x)dx \\ &= \int_{x_0}^{x_1} f(x)dx + \int_{x_1}^{x_2} f(x)dx + \cdots + \int_{x_{n-1}}^{x_n} f(x)dx \end{aligned} \quad (6.21)$$

Áp dụng công thức (6.19) cho n tích phân trong vế phải của (6.21) ta thu được

$$I = \int_a^b f(x)dx \approx \frac{h}{2} [f(x_0) + f(x_1)] + \frac{h}{2} [f(x_1) + f(x_2)] + \cdots + \frac{h}{2} [f(x_{n-1}) + f(x_n)]$$

hay

$$I \approx I^* = \frac{h}{2} [(y_0 + y_n) + 2(y_1 + y_2 + \cdots + y_{n-1})], \quad (6.22)$$

trong đó $h = \frac{b-a}{n}$; $y_i = f(x_i)$, $i = \overline{0, n}$.

Công thức (6.22) gọi là công thức hình thang tổng quát và ta có công thức đánh giá sai số

$$|R| = |I - I^*| \leq \frac{M_2}{12} (b-a) \cdot h^2, \quad (6.23)$$

trong đó

$$M_2 = \max_{x \in [a, b]} |f''(x)|; \quad h = \frac{b-a}{n}.$$

Từ bất đẳng thức (6.23) ta thấy: Để $|I - I^*| < \varepsilon$, chỉ cần chia đoạn $[a, b]$ thành n đoạn sao cho thỏa mãn bất đẳng thức

$$\frac{M_2}{12} (b-a)h^2 = \frac{M_2}{12} \frac{(b-a)^3}{n^2} < \varepsilon,$$

từ đó suy ra $n = \left[\left(\frac{M_2(b-a)^3}{12\varepsilon} \right)^{1/2} \right] + 1$ (với ký hiệu $[\alpha]$ là phần nguyên của số thực α).

2.2.4. Thuật toán

Từ tích phân $I = \int_a^b f(x)dx$ đã cho ta xác định n , tính $h = \frac{b-a}{n}$ và các điểm chia

$$a \equiv x_0 < x_1 < \dots < x_n \equiv b, \quad h = x_{i+1} - x_i, \quad i = \overline{0, n-1}.$$

Tiếp đó, tính

$$y_i = f(x_i), \quad i = \overline{0, n} \quad (6.24)$$

thì

$$I = \int_a^b f(x)dx \approx I^* = \frac{h}{2} [(y_0 + y_n) + 2(y_1 + y_2 + \dots + y_{n-1})]. \quad (6.25)$$

Sai số được đánh giá theo công thức (6.23).

Ví dụ 6.2. Tính theo công thức (6.25) tích phân $I = \int_0^1 \frac{dx}{1+x}$ sao cho đạt sai số < 0.002 .

Giải:

Để $|I - I^*| \leq \frac{(b-a)^2}{12} \cdot \frac{M_2}{n^2} < 0.002$ thì cần chọn $n = \left\lceil \sqrt{\frac{M_2(b-a)^3}{12 \times 0.002}} \right\rceil + 1$,

trong đó $b-a = 1$; $M_2 = \max_{x \in [0,1]} |f''(x)| = \max_{x \in [0,1]} \left| \frac{2}{(1+x)^3} \right| \leq 2$.

Vậy $n = 10 \Rightarrow h = 0.1$.

Tính $f(x_i) = \frac{1}{1+x_i}$ với $x_i = ih$, $i = \overline{0, 10}$, sau đó thay vào công thức (6.25) ta thu được

$$I = \int_0^1 \frac{dx}{1+x} = 0.69377 \pm 0.002.$$

2.2.5. Chương trình MATLAB

```
function I = trapezoid(fun,a,b,npanel)
% trapezoid Tinh gan dung tich phan
% theo cong thuc hinh thang
% Cu phap: I = trapezoid(fun,a,b,n)
%
% Input: fun      = (string) Ten ham
%         a, b    = Cac can cua tich phan xac dinh
%         npanel = so doan chia
%
% Output: I = gia tri gan dung cua tich phan
n=npanel+1; % So diem chia
h = (b-a)/(n-1); % Buoc
x = a:h:b;
f = zeros(size(x));
for i=1:n
    f(i)=feval(fun,x(i));
end
I = h * ( 0.5*f(1) + sum(f(2:n-1)) + 0.5*f(n) );
```

2.3 Công thức Simpson

2.3.1. Trường hợp 3 mốc nội suy

Xét trường hợp đoạn $[a, b]$ gồm 3 mốc nội suy là

$$x_0 = a, \quad x_1 = \frac{a+b}{2}, \quad x_2 = b, \quad h = \frac{b-a}{2}.$$

Tại các mốc nội suy x_i , $i = 0, 1, 2$ ta có bảng số

$$y_0 = f(x_0), \quad y_1 = f(x_1), \quad y_2 = f(x_2).$$

Từ bảng số trên, áp dụng công thức nội suy Newton tiến với $x = x_0 + ht$ ta thu được

$$\begin{aligned} f(x) &\approx p_2(x) = p_2(x_0 + ht) = P_2(t) \\ &= y_0 + \frac{\Delta y_0}{1!}t + \frac{\Delta^2 y_0}{2!}t(t-1). \end{aligned} \quad (6.26)$$

Từ $x = x_0 + ht$ ta có

$$\begin{aligned} I &= \int_a^b f(x)dx = \int_{x_0}^{x_2} f(x)dx \approx \int_0^2 [y_0 + \Delta y_0 t + \frac{\Delta^2 y_0}{2}t(t-1)]hdt \\ &= h[y_0 t + \Delta y_0 \frac{t^2}{2} + \frac{\Delta^2 y_0}{2}(\frac{t^3}{3} - \frac{t^2}{2})] \Big|_0^2 \\ &= h[2y_0 + 2\Delta y_0 + \frac{\Delta^2 y_0}{2}(\frac{8}{3} - 2)]. \end{aligned} \quad (6.27)$$

Mặt khác, do

$$\Delta y_0 = y_1 - y_0, \quad \Delta^2 y_0 = y_0 - 2y_1 + y_2$$

nên thay vào (6.27) ta thu được

$$I = \int_a^b f(x)dx = \int_{x_0}^{x_2} f(x)dx \approx \frac{h}{3} (y_0 + 4y_1 + y_2) =: I^*. \quad (6.28)$$

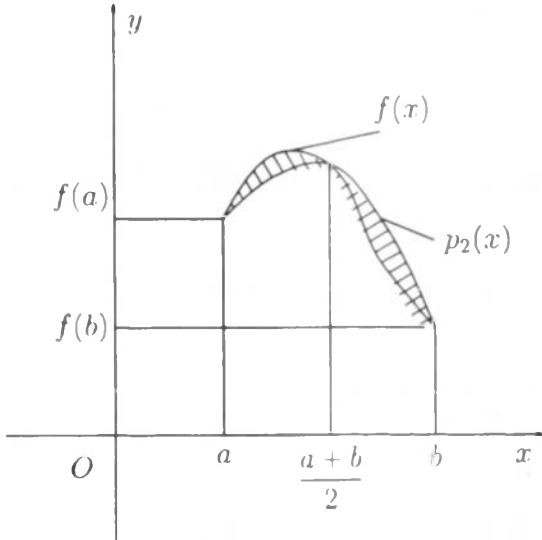
Công thức (6.28) được gọi là công thức Simpson (hình 6.2).

Sai số

$$|R| = |I - I^*| = \left| \int_a^b f(x)dx - \frac{h}{3} (y_0 + 4y_1 + y_2) \right|$$

Với giả thiết hàm số $f(x)$ là liên tục, có đạo hàm liên tục đến cấp 4 và $|f^{(4)}(x)| \leq M_4 \quad \forall x \in [a, b]$ thì sai số được đánh giá theo công thức

$$|I - I^*| \leq \frac{M_4}{90} h^5, \quad h = \frac{b-a}{2}. \quad (6.29)$$



Hình 6.2

2.3.2. Công thức Simpson tổng quát

Để sai số đạt độ chính xác cao, tương tự như đối với công thức hình thang, ta cũng chia đoạn $[a, b]$ thành $2n$ đoạn bằng nhau (vì mỗi đoạn cần 3 mốc nội suy) bởi các điểm chia

$$a \equiv x_0 < x_1 < x_2 < \cdots < x_{2n-2} < x_{2n-1} < x_{2n} \equiv b, \quad h = \frac{b-a}{2n}.$$

Tại các điểm chia trên ta có bảng số

$$y_i = f(x_i), \quad i = \overline{0, 2n}. \quad (6.30)$$

Khi đó ta có

$$\begin{aligned} I &= \int_a^b f(x) dx = \int_{x_0}^{x_{2n}} f(x) dx = \int_{x_0}^{x_2} f(x) dx + \int_{x_2}^{x_4} f(x) dx + \cdots + \int_{x_{2n-2}}^{x_{2n}} f(x) dx \\ &\approx \frac{h}{3} [(y_0 + 4y_1 + y_2) + (y_2 + 4y_3 + y_4) + \cdots + (y_{2n-2} + 4y_{2n-1} + y_{2n})] \end{aligned}$$

hay

$$I \approx I^* = \frac{h}{3} [(y_0 + y_{2n}) + 2(y_2 + y_4 + \dots + y_{2n-2}) + 4(y_1 + y_3 + \dots + y_{2n-1})]. \quad (6.31)$$

Công thức trên được gọi là công thức Simpson tổng quát. Công thức đánh giá sai số

$$|R| = |I - I^*| \leq \frac{M_4}{180}(b-a)h^4, \quad (6.32)$$

trong đó

$$M_4 = \max_{x \in [a,b]} |f^{(4)}(x)|, \quad h = \frac{b-a}{2n},$$

Từ công thức (6.32) ta thấy, để sai số tuyệt đối $|I - I^*| < \varepsilon$ thì số mốc cần chọn

$$n = \left\lceil \sqrt[4]{\frac{16 \times 180 \times \varepsilon}{M_4(b-a)^5}} \right\rceil + 1$$

và ta có $2n+1$ mốc là x_0, x_1, \dots, x_{2n} .

Ví dụ 6.3. Áp dụng tính gần đúng $I = \int_0^1 \frac{dx}{1+x^2}$ theo công thức Simpson tổng quát với (6.31) $n = 5$.

Giải:

Do $n = 5$ nên suy ra $h = \frac{1}{2.5} = 0.1$. Tiếp theo, tính

$$y_i = f(x_i) = \frac{1}{1+x_i^2}, \quad x_i = ih, \quad i = \overline{0, 10}.$$

Vậy

$$\begin{aligned} I^* &= \frac{h}{3} [(y_0 + y_{10}) + 2(y_2 + y_4 + y_6 + y_8) + 4(y_1 + y_3 + y_5 + y_7)] \\ &= 0.7854648. \end{aligned}$$

Bằng cách tính trực tiếp ta thu được

$$I = \int_0^1 \frac{dx}{1+x^2} = \arctan x \Big|_0^1 = \frac{\pi}{4} = 0.78539816.$$

Sai số đạt được là 0.0085%.

Từ công thức hình thang và công thức Simpson ta thấy, nếu lấy nhiều mốc nôi suy hơn (công thức Newton-Cotes) thì sẽ đạt được sai số cấp cao hơn, song công thức sẽ trở nên phức tạp hơn (xem [5]). Vì vậy, trong thực tế người ta thường sử dụng hai loại công thức tổng quát là (6.22) và (6.32).

2.4 Chương trình MATLAB

```

function I = simpson(fun,a,b,npanel)
% simpson Tinh gan dung tich phan theo cong thuc Simpson
% Cu phap: I = simpson(fun,a,b,npanel)
% Input: fun      = (string) Ten ham
%         a, b      = Cac can cua tich phan xac dinh
%         2*npanel = So doan chia
% Output: I = Gia tri gan dung cua tich phan

n = 2*npanel + 1;      % So diem chia
h = (b-a)/(n-1);       % buoc
x = a:h:b;
f = zeros(size(x));
for i=1:n
    f(i)=feval(fun,x(i));
end
I = (h/3)*(f(1)+4*sum(f(2:2:n-1))+2*sum(f(3:2:n-2))+f(n));

```

§3. GIẢI GẦN ĐÚNG PHƯƠNG TRÌNH VI PHÂN (BÀI TOÁN CAUCHY)

3.1 Mở đầu

Trong toán ứng dụng, ta thường gặp rất nhiều bài toán trong dạng phương trình vi phân và cần phải tìm nghiệm của phương trình đó. Ta đã biết trong giáo trình phương trình vi phân, chỉ tìm được nghiệm giải

tích của lớp phương trình rất hẹp; nói chung không có khả năng. Vì vậy buộc phải tìm nghiệm của chúng trong dạng gần đúng.

Trong phần này ta giới thiệu một số thuật toán tìm nghiệm gần đúng thường được dùng trong kỹ thuật đối với bài toán Cauchy, còn đối với dạng phương trình đạo hàm riêng sẽ được nghiên cứu trong giáo trình khác.

3.2 Bài toán Cauchy

Bài toán: Tìm hàm $y = y(x)$ là nghiệm của phương trình

$$y^{(n)} = F(x, y, y', \dots, y^{(n-1)}), \quad x_0 \leq x \leq X \quad (6.33)$$

thỏa mãn điều kiện

$$y(x_0) = y_0, \quad y'(x_0) = y_1; \dots; \quad y^{(n-1)}(x_0) = y_{n-1}. \quad (6.34)$$

trong đó $y_0, y_1, \dots, y_{n-1} \in \mathbb{R}$ và $x_0 \in \mathbb{R}$ là các số đã cho.

Bài toán (6.33), (6.34) gọi là bài toán Cauchy cấp n (cấp cao nhất của đạo hàm có mặt trong phương trình). Điều kiện (6.34) được gọi là điều kiện ban đầu (hay sơ kiện); đồng thời giả thiết hàm F phải thỏa mãn các điều kiện cần thiết để bài toán Cauchy tồn tại duy nhất nghiệm.

3.3 Phương pháp giải bài toán Cauchy

3.3.1. Công thức Euler

Xét bài toán Cauchy cấp một:

Tìm hàm $y = y(x)$ là nghiệm của bài toán sau

$$\begin{aligned} y' &= f(x, y), \quad x_0 \leq x \leq X, \\ y(x_0) &= y_0 \end{aligned} \quad (6.35)$$

Thuật toán: Giả sử biết nghiệm $y = y(x)$ tại điểm $x \in [x_0, X]$, ta tìm nghiệm y tại điểm $x + h \in [x_0, X]$ là $y = y(x + h)$ với $h > 0$ được gọi là bước để có được nghiệm tại điểm $x + h$.

Tích phân hai vế của phương trình (6.35) trên đoạn $[x, x+h]$ ta có

$$y(x+h) = y(x) + \int_x^{x+h} f(t, y(t)) dt = y(x) + \int_x^{x+h} y'(t) dt.$$

Đổi biến $t = x+z$ ta thu được

$$y(x+h) = y(x) + \int_0^h y'(x+z) dz. \quad (6.36)$$

Áp dụng khai triển Taylor ta có

$$y'(x+z) = y'(x) + \frac{y''(x)}{1!} z + O(z^2),$$

từ đó suy ra

$$\begin{aligned} \int_0^h y'(x+z) dz &= \int_0^h [y'(x) + y''(x)z + O(z^2)] dz \\ &= y'(x).h + O(h^2) = hf(x, y) + O(h^2). \end{aligned}$$

Thay vào (6.36) ta thu được

$$y(x+h) = y(x) + hf(x, y) + O(h^2). \quad (6.37)$$

Trong (6.37) bỏ qua $O(h^2)$ ta được xấp xỉ, sau đó thay dấu xấp xỉ bởi dấu " $=$ " và cũng gọi y là $y(x)$ ta thu được

$$y(x+h) = y(x) + hf(x, y(x)). \quad (6.38)$$

Công thức (6.38) được gọi là công thức Euler. Quá trình tìm nghiệm theo công thức này được tiến hành như sau:

Chia đoạn $[x_0, X]$ thành N đoạn bởi các điểm chia

$$x_0, x_1 = x_0 + h, x_2 = x_1 + h, \dots, x_{i+1} = x_i + h, \dots$$

Áp dụng công thức (6.38) ta được

$$\begin{aligned} y(x_0) &= y_0 \quad (\text{đã cho theo điều kiện đầu của bài toán}) \\ y(x_1) &= y_1 = y_0 + hf(x_0, y_0) \\ y(x_2) &= y_2 = y_1 + hf(x_1, y_1) \\ &\vdots \\ y(x_{k+1}) &= y_{k+1} = y_k + hf(x_k, y_k) \\ &\vdots \end{aligned} \quad (6.39)$$

Sai số một bước từ điểm x đến $x + h$ là $O(h^2)$.

Ví dụ 6.4. Tính nghiệm của bài toán Cauchy

$$\begin{cases} y' = y - \frac{2x}{y} \\ y(0) = 1 \end{cases}$$

bằng phương pháp Euler lấy với $h = 0.2$ trên đoạn $[0, 1]$.

Giải:

Theo công thức (6.39) ta có

$$\begin{aligned} y_0 &= 1, \\ y_1 &= y(0.2) = y_0 + h(y_0 - \frac{2x_0}{y_0}) = 1.2000, \\ y_2 &= y(0.4) = y_1 + h(y_1 - \frac{2x_1}{y_1}) = 1.3733, \\ y_3 &= y(0.6) = 1.5294, \\ y_4 &= y(0.8) = 1.6786, \\ y_5 &= y(1) = 1.8237. \end{aligned}$$

Nghiệm đúng $\bar{y} = \sqrt{2x + 1}$ và ta có các giá trị tương ứng

$$\begin{aligned} \bar{y}_0 &= 1, \\ \bar{y}_1 &= 1.1832, \\ \bar{y}_2 &= 1.3416, \\ \bar{y}_3 &= 1.4832, \\ \bar{y}_4 &= 1.6124, \\ \bar{y}_5 &= 1.7320. \end{aligned}$$

So sánh giữa nghiệm gần đúng y_5 với nghiệm đúng \bar{y}_5 :

$$|y_5 - \bar{y}_5| = 0.0917 \approx 9\%.$$

Sai số quá lớn là do quá trình tích lũy sai số: từ y_1 gấp sai số, từ đó tính tiếp y_2, \dots

Chương trình MATLAB

```

function [x,y]=euler(fun,x0,xn,y0,h)
% This function illustrates the euler's method for the
% iIVP: dy/dx=fun(x,y); x0<=x<=xn; y(x0)=y0;

% Synopsis: _ [x, y]=euler(fun, x0, xn, y0)
%           _ [x, y]=euler(fun, x0, xn, y0, h)

% Input: _ fun is the function in string type
%         _ [x0, xn] is the interval
%         _ y0 is the initial value
%         _ h is the step size

% Output: Two vector x and y
if nargin<5
    h=0.01;
end
x=(x0:h:xn)';
n=length(x);
y=y0*ones(n,1);
for i=2:n
    y(i)=y(i-1)+h*feval(fun,x(i-1),y(i-1));
end
end

```

3.3.2. Công thức Euler cải tiến

Mục đích của công thức này là để đạt được sai số một bước cấp cao hơn. Trong tích phân về phải của công thức (6.36), thay bởi công thức hình thang (6.19) ta được

$$\begin{aligned}
 \int_0^h y'(x+z)dz &= \frac{h}{2} [y'(x+h) + y'(x)] + O(h^3) \\
 &= \frac{h}{2} [f(x+h, y(x+h)) + f(x, y(x))] + O(h^3).
 \end{aligned}$$

Thay vào (6.35) ta có

$$y(x+h) = y(x) + \frac{h}{2} [f(x+h, y(x+h)) + f(x, y(x))] + O(h^3).$$

Bỏ qua $O(h^3)$, ta được xấp xỉ. Sau đó thay xấp xỉ bởi dấu bằng, ta thu được

$$y(x+h) = y(x) + \frac{h}{2} [f(x+h, y(x+h)) + f(x, y(x))]. \quad (6.40)$$

Công thức (6.40) được gọi là công thức Euler cải tiến, đạt sai số một bước là $O(h^3)$.

Theo công thức (6.40) ta thấy, nếu biết $y = y(x)$ khi tính $y = y(x+h)$ thì phải giải phương trình phi tuyến, có thể áp dụng công thức lặp. Để thuận lợi cho việc tính toán, ta tìm cách biến đổi (6.40) sao cho sai số vẫn đạt cấp $O(h^3)$.

Thay $y(x+h)$ trong vế phải của (6.40) bởi đẳng thức (6.37)

$$y(x+h) = y(x) + hf(x, y(x)) + O(h^2)$$

hay

$$y(x+h) - y^* = O(h^2). \quad (6.41)$$

trong đó

$$y^* = y(x) + hf(x, y(x)).$$

Mặt khác, áp dụng định lý Lagrange ta có

$$f(x+h, y(x+h)) - f(x+h, y^*) = f'_y(x+h, \bar{y})(y(x+h) - y^*).$$

Kết hợp điều này với (6.41) ta suy ra

$$f(x+h, y(x+h)) - f(x+h, y^*) = O(h^2)$$

với giả thiết $f'_y(x+h, \bar{y})$ là đại lượng giới hạn. Vậy ta có

$$f(x+h, y(x+h)) = f(x+h, y^*) + O(h^2).$$

Thay vào (6.40) ta thu được

$$\begin{aligned} y(x+h) &= y(x) + \frac{h}{2} [f(x, y(x)) + f(x+h, y^*)], \\ y^* &= y(x) + hf(x, y(x)). \end{aligned} \quad (6.42)$$

Công thức này đạt sai số một bước là $O(h^3)$. Công thức lặp cụ thể của (6.42) như sau:

$$\begin{aligned} y(x_0) &= y_0, \\ y_1^* &= y_0 + hf(x_0, y_0), \\ y(x_1) &= y(x_0 + h) \approx y_1 = y_0 + \frac{h}{2}(f(x_0, y_0) + f(x_0 + h, y_1^*)) \\ &\vdots \end{aligned}$$

Tổng quát

$$\left\{ \begin{array}{l} y_i^* = y_i + hf(x_i, y_i), \quad i = 0, 1, 2, \dots \\ y(x_i + h) = y_{i+1} = y_i + \frac{h}{2}[f(x_i, y_i) + f(x_i + h, y_i^*)]. \end{array} \right. \quad (6.43)$$

Ví dụ 6.5. Tìm nghiệm gần của bài toán Cauchy

$$\begin{cases} y' = \frac{xy}{2} \\ y(0) = 1 \end{cases}$$

theo công thức Euler cải tiến lấy $h = 0.1$ tính trên đoạn $(0; 0.5)$.

Giải:

Nghiệm chính xác $\bar{y} = e^{\frac{x^2}{4}}$. Kết quả tính toán được ghi trong bảng sau:

i	x_i	y_i	\bar{y}_i
0	0	1	1
1	0.1	1.0025	1.0025
2	0.2	1.0100	1.0100
3	0.3	1.0227	1.0227
4	0.4	1.0408	1.0408
5	0.5	1.0645	1.0645

Nhận thấy các kết quả trùng nhau tới 4 số lẻ sau dấu "".

Chương trình MATLAB

```

function [x,y]=modified_euler(fun,x0,xn,y0,h)
% This function illustrates the modified euler method for
% the IVP: dy/dx=fun(x,y); y(x0)=y0;

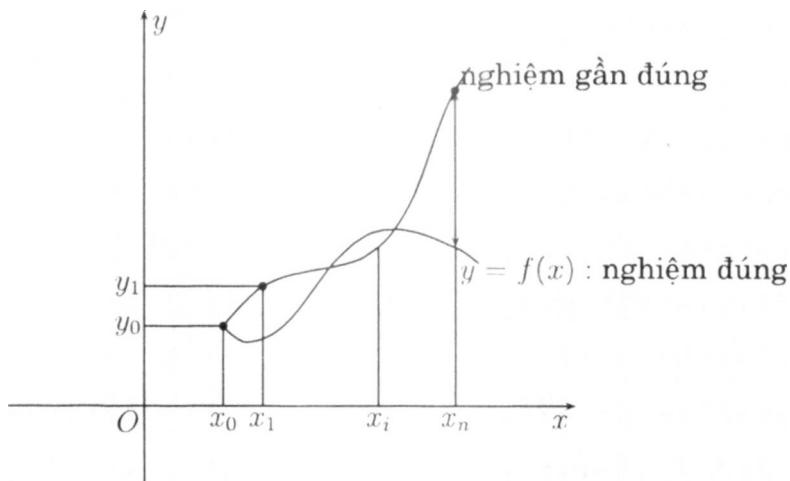
if nargin<4
    h=0.01;
end

x=(x0:h:xn)';
n=length(x);
y=y0*ones(n,1);

for i=2:n
    z=y(i-1)+h*feval(fun,x(i-1),y(i-1));
    y(i)=y(i-1)+...
        h*(feval(fun,x(i-1),y(i-1))+feval(fun,x(i-1)+h,z))/2;
end
plot(x,y);
end

```

Chú ý 6.5. Công thức Euler (6.39) và Euler cải tiến (6.43) đều có ưu điểm là tính toán thuận lợi, dễ lập trình, song có nhược điểm chính là sai số một bước của chúng lần lượt là $O(h^2)$ và $O(h^3)$. Giả sử từ $y(x_0) = y_0$ là đúng thì việc tính $y_1 = y(x_0 + h)$ đã gấp sai số tương ứng, không còn là đúng nữa. Dựa trên gần đúng này để đi tìm gần đúng tiếp theo dẫn đến quá trình tính tới $y_i = y(x_0 + ih)$ với x_i xa với x_0 sẽ có thể dẫn đến hiện tượng "sai một ly đi một dặm" (hình 6.3). Vì vậy các công thức trên chỉ thích hợp với các điểm x_i gần với x_0 .



Hình 6.3

3.3.3. Công thức dạng Runge-Kutta

Để đạt được độ chính xác cao hơn mà công thức tính cũng không quá phức tạp, Runge-Kutta đã đưa ra công thức tìm nghiệm số của bài toán Cauchy cấp một (6.35) như sau: Biết $y(x)$, tìm $y(x + h)$ theo công thức

$$y(x + h) = y(x) + \sum_{r=1}^q p_r k_r(h), \quad (6.44)$$

trong đó

$$\begin{aligned} k_1(h) &= hf(x, y(x)) \\ k_2(h) &= hf(x + \alpha_2 h, y(x) + \beta_{21} k_1(h)) \\ k_3(h) &= hf(x + \alpha_3 h, y(x) + \beta_{31} k_1(h) + \beta_{32} k_2(h)) \\ &\vdots \\ k_q(h) &= hf(x + \alpha_q, \sum_{j=1}^{q-1} \beta_{qj} k_j(h)). \end{aligned} \quad (6.45)$$

Các hệ số

$$p_r, r = \overline{1, q}; \quad \alpha_j, j = \overline{2, q}; \quad \beta_{pj}, p = \overline{2, q}, j = \overline{1, q-1}$$

được xác định sao cho (6.44) thỏa mãn, đồng thời khai triển Taylor

$$y(x+h) = y(x) + hy'(x) + \frac{h^2}{2!}y''(x) + \cdots + \frac{h^m}{m!}y^{(m)}(x) + \frac{h^{m+1}}{(m+1)!}y^{(m+1)}(c) \quad (6.46)$$

trùng nhau tới cấp cao nhất của h mà ta mong muốn. Các chỉ số $q = 1, 2, 3, \dots$ gọi là bậc của công thức dạng Runge-Kutta.

Áp dụng công thức khai triển Taylor theo lũy thừa của h và so sánh (6.44), (6.46), ta suy ra được các công thức sau, thường được sử dụng trong kỹ thuật:

1. Khi $q = 1$ (bậc một) ta có

$$y(x + h) = y(x) + hf(x, y(x)) \quad (6.47)$$

chính là công thức Euler, sai số một bước là $O(h^2)$.

2. Khi $q = 2$ (bậc hai).

Khai triển Taylor $k_1(h), k_2(h)$ theo lũy thừa của h , thay vào (6.44), đồng nhất với (6.46) theo lũy thừa của h tới h^2 , ta thu được hệ ba phương trình để xác định bốn hệ số $p_1, p_2, \alpha_2, \beta_{21}$ như sau:

$$\begin{cases} p_1 + p_2 &= 1 \\ \alpha_2 p_2 &= \frac{1}{2} \\ \beta_{21} p_2 &= \frac{1}{2}. \end{cases}$$

Do hệ trên có số ẩn nhiều hơn số phương trình nên ta chọn một ẩn tự do và giải các ẩn phụ thuộc và ta có công thức dạng Runge-Kutta bậc hai với sai số một bước là $O(h^3)$. Chẳng hạn:

(a) Chọn $p_1 = 0$ thì $p_2 = 1$ và suy ra $\alpha_2 = \beta_{21} = \frac{1}{2}$ ta thu được công thức

$$\begin{aligned} k_1(h) &= hf(x, y(x)) \\ k_2(h) &= hf\left(x + \frac{h}{2}, y(x) + \frac{k_1(h)}{2}\right) \\ y(x + h) &= y(x) + k_2(h) \end{aligned} \quad (6.48)$$

(b) Chọn $p_1 = p_2 = \frac{1}{2}$ thì $\alpha_2 = \beta_{21} = 1$, ta có

$$\begin{aligned} k_1(h) &= hf(x, y(x)) \\ k_2(h) &= hf\left(x + h, y(x) + k_1(h)\right) \\ y(x + h) &= y(x) + \frac{1}{2}(k_1(h) + k_2(h)) \end{aligned} \quad (6.49)$$

Trong công thức (6.49) ta thấy, nếu đặt

$$y^* = y(x) + k_1(h) = y(x) + hf(x, y(x))$$

thì ta có

$$y(x+h) = y(x) + \frac{h}{2} [f(x, y(x)) + f(x+h, y^*)]$$

chính là công thức Euler cải tiến.

Có thể chọn p_r , α_j , β_{pj} thích hợp, ta thu được dạng công thức khác, nên người ta nói: ta có họ công thức dạng Runge-Kutta bậc hai. Sai số một bước là $O(h^3)$.

3. Khi $q = 3$ (bậc 3).

Hoàn toàn tương tự như trên, ta có họ công thức dạng Runge-Kutta bậc ba (R-K3). Ở đây ta chỉ nêu một công thức thường được sử dụng, đạt sai số một bước là $O(h^4)$:

$$k_1(h) = hf(x, y(x)) \quad (6.50)$$

$$k_2(h) = hf\left(x + \frac{h}{2}, y(x) + \frac{k_1}{2}\right) \quad (6.50)$$

$$k_3(h) = hf\left(x + h, y(x) - k_1 + 2k_2\right) \quad (6.51)$$

$$y(x+h) = y(x) + \frac{1}{6}(k_1 + 4k_2 + k_3)$$

4. Tương tự khi $q = 4$ (bậc 4) ta có công thức dạng R-K4:

$$k_1(h) = hf(x, y(x)) \quad (6.52)$$

$$k_2(h) = hf\left(x + \frac{h}{2}, y(x) + \frac{k_1}{2}\right)$$

$$k_3(h) = hf\left(x + \frac{h}{2}, y(x) + \frac{k_2}{2}\right) \quad (6.52)$$

$$k_4(h) = hf(x+h, y(x) + k_3)$$

$$y(x+h) = y(x) + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

đạt sai số một bước là $O(h^5)$.

Lấy $q = 5, 6, \dots$ ta thu được các công thức dạng R-K đạt độ chính xác cao hơn song việc tính toán là quá phức tạp nên công thức (6.52) thường được sử dụng nhiều trong kỹ thuật.

3.4 Chương trình MATLAB

```

function [x,y]=RK4(fun,x0,xn,y0,h)
% This funtion illustrates the Runge - Kutta method for
% the initial value problem: dy/dx=fun(x,y); y(x0)=y0;

if nargin<4
    h=0.01;
end

x=(x0:h:xn)';
n=length(x);
y=y0*ones(n,1);

for i=2:n
    k1=h*feval(fun,x(i-1),y(i-1));
    k2=h*feval(fun,x(i-1)+h/2,y(i-1)+k1/2);
    k3=h*feval(fun,x(i-1)+h/2,y(i-1)+k2/2);
    k4=h*feval(fun,x(i-1)+h,y(i-1)+k3);
    y(i)=y(i-1)+(k1+2*k2+2*k3+k4)/6;
end
plot(x,y);
end

```

Ví dụ 6.6. Áp dụng chương trình MATLAB giải bài toán sau bằng phương pháp R-K4

$$\begin{cases} y' = x + y \\ y(0) = 1 \end{cases}$$

trên đoạn $[0; 0.5]$ lấy với $h = 0.1$.

Nghiệm đúng $\bar{y} = 2e^x - x - 1$. Kết quả tính được cho trong bảng sau:

i	x_i	y_i	\bar{y}_i
0	0	1	1
1	0.1	1.1103	1.1103
2	0.2	1.2427	1.2428
3	0.3	1.3996	1.3997
4	0.4	1.5836	1.5836
5	0.5	1.7974	1.79744

§4. GIẢI PHƯƠNG TRÌNH VI PHÂN CẤP CAO VÀ HỆ PHƯƠNG TRÌNH VI PHÂN

Để việc trình bày đơn giản, trong tiết này ta chỉ đề cập đến bài toán Cauchy đối với hệ phương trình vi phân cấp một và phương trình vi phân cấp hai.

4.1 Bài toán

Tìm hàm số $y = y(x)$ và $z = z(x)$ là nghiệm của bài toán Cauchy sau:

$$\begin{aligned} y' &= f(x, y, z), \quad y(x_0) = \alpha \\ z' &= g(x, y, z), \quad z(x_0) = \beta \\ x &\in [x_0, x_0 + X] \end{aligned} \tag{6.53}$$

hoặc tìm $y = y(x)$ là nghiệm của bài toán Cauchy cấp hai

$$\begin{aligned} y'' &= f(x, y, y'), \quad x \in [x_0, x_0 + X] \\ y(x_0) &= \alpha; \quad y'(x_0) = \beta. \end{aligned} \tag{6.54}$$

Ta thấy bài toán (6.54) luôn có thể đưa về dạng (6.53). Thật vậy, đặt

$$y' = z(x) \Rightarrow y'' = z'(x).$$

Kết hợp điều này với (6.54) ta thu được

$$\begin{aligned} y' &= z(x), \quad y(x_0) = \alpha \\ z' &= f(x, y, z), \quad z(x_0) = \beta \end{aligned} \tag{6.55}$$

chính là bài toán dạng (6.53).

Do vậy, trong tiết này ta chỉ nêu một số công thức tìm nghiệm của bài toán Cauchy dạng (6.53).

4.2 Công thức Euler

Giả sử có $y = y(x)$, $z = z(x)$, $x \in [x_0, x_0 + X]$, ta đi tìm $y(x+h)$, $z(x+h)$.
Tương tự trường hợp một phương trình, ta có các công thức

$$\begin{aligned} y(x+h) &= y(x) + hf(x, y(x), z(x)); \\ z(x+h) &= z(x) + hf(x, y(x), z(x)). \end{aligned}$$

Sai số một bước là $O(h^2)$. Để lập công thức lặp cho trường hợp này, ta chia đoạn $[x_0, x_0 + X]$ thành n đoạn bằng nhau với bước là $h = \frac{X}{n}$ bởi các điểm chia

$$x_{i+1} = x_0 + ih, \quad i = 0, 1, 2, \dots$$

Xuất phát từ x_0 ta có $y_0 = \alpha$, $z_0 = \beta$, công thức lặp:

$$\begin{aligned} y_{i+1} &= y_i + hf(x_i, y_i, z_i); \\ z_{i+1} &= z_i + hf(x_i, y_i, z_i), \quad i = 0, 1, 2, \dots \end{aligned} \tag{6.56}$$

4.3 Công thức dạng Runge-Kutta

Hoàn toàn tương tự như trên, ta có công thức bậc bốn (R-K4) như sau:
Xuất phát từ

$$y(x_0) = y_0 = \alpha; \quad z(x_0) = z_0 = \beta.$$

Giả sử ở bước thứ i ta có y_i , z_i . Khi đó

$$y_{i+1} = y_i + \frac{1}{6} \Delta y_i; \quad z_{i+1} = z_i + \frac{1}{6} \Delta z_i, \tag{6.57}$$

trong đó

$$\begin{aligned} \Delta y_i &= k_1 + 2k_2 + 2k_3 + k_4 \\ \Delta z_i &= l_1 + 2l_2 + 2l_3 + l_4 \end{aligned} \tag{6.58}$$

và

$$\begin{aligned}
 k_1 &= hf(x_{i-1}, y_{i-1}, z_{i-1}); & l_1 &= hz(x_{i-1}, y_{i-1}, z_{i-1}); \\
 k_2 &= hf\left(x_{i-1} + \frac{h}{2}, y_{i-1} + \frac{k_1}{2}, z_{i-1} + \frac{l_1}{2}\right); & l_2 &= hz\left(x_{i-1} + \frac{h}{2}, y_{i-1} + \frac{k_1}{2}, z_{i-1} + \frac{l_1}{2}\right); \\
 k_3 &= hf\left(x_{i-1} + \frac{h}{2}, y_{i-1} + \frac{k_2}{2}, z_{i-1} + \frac{l_2}{2}\right); & l_3 &= hz\left(x_{i-1} + \frac{h}{2}, y_{i-1} + \frac{k_2}{2}, z_{i-1} + \frac{l_2}{2}\right); \\
 k_4 &= hf(x_{i-1} + h, y_{i-1} + k_3, z_{i-1} + l_3); & l_4 &= hz(x_{i-1} + h, y_{i-1} + k_3, z_{i-1} + l_3).
 \end{aligned} \tag{6.59}$$

Ta đã biết, công thức này đạt sai số một bước là $O(h^5)$.

4.4 Chương trình MATLAB

```

function [x,y,z]=RK4_Sys(fun1,fun2,x0,xn,y0,z0,h)
% This function illustrates the Runge - Kutta method for the
% initial value problem:
% dy/dx=fun1(x,y,z); y(x0)=y0; dz/dx=fun2(x,y,z); z(x0)=z0

if nargin<7
    h=0.01;
end

x=(x0:h:xn)';
n=length(x);
y=y0*ones(n,1);
z=z0*ones(n,1);

for i=2:n
    k1=h*feval(fun1,x(i-1),y(i-1),z(i-1));
    l1=h*feval(fun2,x(i-1),y(i-1),z(i-1));

    k2=h*feval(fun1,x(i-1)+h/2,y(i-1)+k1/2,z(i-1)+l1/2);
    l2=h*feval(fun2,x(i-1)+h/2,y(i-1)+k1/2,z(i-1)+l1/2);

```

```

k3=h*feval(fun1,x(i-1)+h/2,y(i-1)+k2/2,z(i-1)+l2/2);
l3=h*feval(fun2,x(i-1)+h/2,y(i-1)+k2/2,z(i-1)+l2/2);

k4=h*feval(fun1,x(i-1)+h,y(i-1)+k3,z(i-1)+l3);
l4=h*feval(fun2,x(i-1)+h,y(i-1)+k3,z(i-1)+l3);

y(i)=y(i-1)+(k1+2*k2+2*k3+k4)/6;
z(i)=z(i-1)+(l1+2*l2+2*l3+l4)/6;

end
end

```

Áp dụng tìm nghiệm của bài toán sau

$$\begin{cases} y' = z, & y(0) = 0 \\ z' = -xz - y, & z(0) = 1 \end{cases}$$

trên đoạn $[0; 0.5]$ với $h = 0.1$.

Áp dụng chương trình MATLAB ta thu được kết quả trong bảng sau

i	x_i	y_i	z_i
0	0	0	1
1	0.1	0.1	0.9908
2	0.2	0.1979	0.9661
3	0.3	0.2933	0.9321
4	0.4	0.3834	0.8892
5	0.5	0.4699	0.8379

§5. BÀI TẬP

Bài tập 6.1. Cho 5 mốc nội suy là x_0, x_1, x_2, x_3, x_4 . Hãy lập công thức tính đạo hàm của hàm $y = f(x)$ tại 5 mốc đã cho.

Bài tập 6.2. Tính gần đúng đạo hàm của hàm $y = \log x$ tại điểm $x = 1$ từ bảng số sau:

x	0.98	1.00	1.02
$y = \log x$	0.7739332	0.7651977	0.7563321

Bài tập 6.3. Tính gần đúng tích phân xác định

$$I = \int_1^2 \sqrt{x} dx$$

bằng công thức hình thang tổng quát với $n = 10$. Đánh giá sai số.

Bài tập 6.4. Tính

$$I = \int_0^1 \sqrt{1+x^2} dx$$

bằng công thức Simpson tổng quát sao cho đạt sai số 0.001.

Bài tập 6.5. Cho $I = \int_{4.2}^{5.4} \frac{3.5x^2 + 5.1x - 4.8}{x - 0.4} dx$.

1. Tính gần đúng tích phân I theo công thức hình thang tổng quát với bước $h = 0.1$;
2. Nếu sử dụng công thức Simpson tổng quát thì phải chia đoạn $[4.2; 5.4]$ thành bao nhiêu đoạn nhỏ bằng nhau để đạt được sai số $< 10^{-3}$?

Bài tập 6.6. Tìm nghiệm gần đúng của bài toán Cauchy

$y'(x) = y - x$; $y(0) = 1$ trên đoạn $[0; 0.5]$ theo công thức Euler lấy $h = 0.1$.

Bài tập 6.7. Tìm nghiệm gần đúng của các bài toán Cauchy sau:

1. $y' = \frac{(x+y)(1-xy)}{x+2y}$; $y(0) = 1$;

2. $y' = y - \frac{2x}{y}$, $y(0) = 1$;

3. $y' = y^2 + \frac{y}{x}$, $y(2) = 4$;

4. $y' = x^2 + y^2$, $y(0) = -1$;

theo các công thức Euler, Euler cải tiến và công thức R-K4 trên đoạn $[0; 0.5]$ với $h = 0.1$.

Bài tập 6.8. Tìm nghiệm gần đúng của các bài toán sau trên đoạn $[0; 0.5]$ lấy với $h = 0.1$ bằng công thức R-K4.

1. $\begin{cases} y' = x + z^2, & y(0) = 1 \\ z' = xy, & z(0) = -1 \end{cases}$

2. $\begin{cases} y' = y + z, & y(0) = 1 \\ z' = -y + z, & z(0) = -1 \end{cases}$

TÀI LIỆU THAM KHẢO

- [1] Phạm Kỳ Anh, Phan Văn Hạp, Hoàng Đức Nguyên, Lê Đình Thịnh, *Cơ sở phương pháp tính tập I, II*, NXB Đại học Quốc gia Hà Nội, 1996.
- [2] Ta Văn Đĩnh, Lê Trọng Vinh, *Phương pháp tính*, NXB Đại học và Trung học chuyên nghiệp, 1983.
- [3] Phan Văn Hạp, Lê Đình Thịnh, *Phương pháp tính và các thuật toán*, NXB Giáo dục, 2001.
- [4] Phan Văn Hạp và các tác giả khác, *Phương pháp tính tập I, II*, NXB Đại học và Trung học chuyên nghiệp, 1970 - 1971.
- [5] Lê Trọng Vinh, *Giáo trình Giải tích số*, NXB Khoa học và Kỹ thuật, 2007.
- [6] Dương Thuỷ Vỹ, *Giáo trình Phương pháp tính*, NXB Khoa học và Kỹ thuật, 2011.
- [7] Granville Sewell, *The numerical solution of ordinary and partial differential equations*, Welley interscience, 2005.
- [8] Hanselman L. and Littlefield B., *Mastering MatLab 6*, Prentice Hall, 2001.
- [9] Higham D. J., *MatLab guide*, SIAM, 2000.
- [10] John. H. Mathews, Kurtis D. Fink, *Numerical Methods Using MatLab*, Pearson Prentice Hall, 2004.
- [11] Jaacson E., Keller H., *Analysis of numerical methods*, J. Wiley, 1968.

GIÁO TRÌNH

PHƯƠNG PHÁP TÍNH VÀ MATLAB

NHÀ XUẤT BẢN BÁCH KHOA – HÀ NỘI
Ngõ 17 Tạ Quang Bửu – Hai Bà Trưng – Hà Nội
ĐT: 04. 38684569; Fax: 04. 38684570
www.nxbbk.hust.edu.vn

Chịu trách nhiệm xuất bản:

TS. PHÙNG LAN HƯƠNG

Phản biện: TS. HÀ THỊ NGỌC YÊN
TS. NGUYỄN THANH HUYỀN
Biên tập: ĐÔ THANH THỦY
Sửa bản in: TRẦN MINH TOÀN
Trình bày: HOÀNG HẢI YÊN

In 500 cuốn khổ 16 × 24cm tại Xưởng thực hành kỹ thuật in DHBK Hà Nội, số 0:05, ngõ 40 Tạ Quang Bửu, Hà Nội.

Số đăng ký KHXB: 1264 - 2013/CXB/Q1 - 51/BKHN; ISBN: 97860411557878, do Cục Xuất bản cấp ngày 11/9/2013.

Số QĐXB: 228/QĐ - DHBK - BKHN ngày 24/9/2013.

In xong và nộp lưu chiểu quý IV năm 2013.