

TRƯỜNG ĐẠI HỌC KINH TẾ-KỸ THUẬT CÔNG NGHIỆP

CHU BÌNH MINH

TÀI LIỆU HỌC TẬP

PHƯƠNG PHÁP TÍNH

LƯU HÀNH NỘI BỘ

MỤC LỤC

Mục lục	i
Danh sách bảng	v
Danh sách hình vẽ	vii
MỞ ĐẦU	1
Chương 1. SAI SỐ	5
1.1 Giới thiệu	6
1.2 Sai số và cách viết xấp xỉ	6
1.2.1 Sai số tuyệt đối và sai số tương đối	6
1.2.2 Sai số quy tròn	8
1.2.3 Cách viết xấp xỉ	9
1.3 Các quy tắc tính sai số	10
1.3.1 Sai số tính toán	10
1.3.2 Sai số phương pháp	12
Chương 2. TÍNH GẦN ĐÚNG NGHIỆM THỰC CỦA PHƯƠNG TRÌNH	15
2.1 Giới thiệu	16
2.2 Một số phương pháp khoảng	17
2.2.1 Phương pháp tìm kiếm gia tăng	18
2.2.2 Phương pháp chia đôi	19
2.2.3 Phương pháp dây cung	24
2.3 Các phương pháp mở	27
2.3.1 Phương pháp lặp	28
2.3.2 Phương pháp Newton-Raphson	32
2.4 0-diểm của đa thức	39
2.4.1 Giới thiệu	39
2.4.2 Tính giá trị đa thức	40
2.4.3 Giảm bậc đa thức	42
2.4.4 Phương pháp Laguerre	44
2.5 Hàm Matlab giải phương trình	49

Chương 3. TÍNH GẦN ĐÚNG NGHIỆM CỦA HỆ PHƯƠNG TRÌNH ĐẠI SỐ TUYẾN TÍNH	53
3.1 Giới thiệu	54
3.2 Phương pháp khử Gauss	59
3.2.1 Giới thiệu	59
3.2.2 Thuật toán khử Gauss	61
3.3 Phương pháp phân rã LU	69
3.3.1 Giới thiệu	69
3.3.2 Phương pháp phân rã Doolittle	70
3.3.3 Phương pháp phân rã Cholesky	76
3.4 Ma trận dải và ma trận đối xứng	81
3.4.1 Ma trận ba đường chéo	82
3.4.2 Ma trận đối xứng	86
3.4.3 Ma trận dải đối xứng	89
3.5 Phần tử xoay	95
3.5.1 Giới thiệu	95
3.5.2 Ma trận chéo trội	96
3.5.3 Phương pháp khử Gauss kết hợp đổi hàng xoay	97
3.6 Phương pháp lặp đơn	106
3.6.1 Giới thiệu	106
3.6.2 Phương pháp lặp Jacobi	107
3.6.3 Phương pháp lặp Seidel	110
3.6.4 Phương pháp Gradient liên hợp	117
3.7 Một số hàm Matlab tính toán trong đại số tuyến tính	124
Chương 4. NỘI SUY VÀ PHƯƠNG PHÁP BÌNH PHƯƠNG NHỎ NHẤT	133
4.1 Giới thiệu	134
4.2 Nội suy đa thức	135
4.2.1 Đa thức nội suy Lagrange	135
4.2.2 Đa thức nội suy Newton	137
4.2.3 Phương pháp nội suy Neville	143
4.3 Phương pháp bình phương nhỏ nhất	147
4.3.1 Mở đầu	147

4.3.2	Phương trình hồi quy tuyến tính	149
4.3.3	Xấp xỉ dạng tuyến tính	151
4.3.4	Xấp xỉ dạng đa thức	151
4.3.5	Dữ liệu có trọng số	157
4.4	HÀM MATLAB XẤP XỈ HÀM SỐ	161
Chương 5. TÍNH GẦN ĐÚNG ĐẠO HÀM VÀ TÍCH PHÂN		167
5.1	Giới thiệu	168
5.2	Tính gần đúng đạo hàm	168
5.2.1	Mở đầu	168
5.2.2	Xấp xỉ sai phân hữu hạn	169
5.2.3	Phép ngoại suy Richardson	175
5.2.4	Xấp xỉ đạo hàm bằng nội suy	176
5.3	Tính gần đúng tích phân xác định	180
5.3.1	Mở đầu	180
5.3.2	Các công thức Newton-Cotes	181
5.3.3	Tích phân Romberg	192
5.3.4	Công thức cầu phương Gaussian	197
5.4	Tích phân bội	214
5.4.1	Mở đầu	214
5.4.2	Phương pháp cầu phương Gauss-Legendre trên phần tử tứ giác	215
5.4.3	Phương pháp cầu phương trên một phần tử tam giác	222
5.5	HÀM MATLAB TÍNH ĐẠO HÀM-TÍCH PHÂN	227
Chương 6. TÍNH GẦN ĐÚNG NGHIỆM CỦA PHƯƠNG TRÌNH VI PHÂN		233
6.1	Giới thiệu	234
6.2	Phương pháp chuỗi Taylor	235
6.3	Các phương pháp Runge-Kutta	241
6.3.1	Phương pháp Euler	241
6.3.2	Phương pháp Runge-Kutta bậc-hai	243
6.3.3	Phương pháp Runge-Kutta bậc-bốn	246
6.4	Tính ổn định và tính cương	253
6.4.1	Tính ổn định của phương pháp Euler	254
6.4.2	Tính cương của bài toán	255

6.5	Phương pháp Runge-Kutta thích ứng	257
6.6	Phương pháp Bulirsch-Stoer	265
6.6.1	Phương pháp trung điểm	265
6.6.2	Ngoại suy Richardson	266
6.6.3	Thuật toán Bulirsch-Stoer	271
6.7	Một số hàm Matlab giải phương trình vi phân	274
PHỤ LỤC A: Danh sách các hàm Matlab được lập trình trong sách		280
CHỈ MỤC		283
TÀI LIỆU THAM KHẢO		285

DANH SÁCH BẢNG

2.1	Bảng kết quả tìm nghiệm của phương trình $x^3 - x - 1 = 0$ trên đoạn $[1, 2]$ bằng phương pháp lặp đơn	28
2.2	Bảng kết quả tìm nghiệm của phương trình $x^3 - x - 1 = 0$ trên đoạn $[1, 2]$ bằng phương pháp lặp đơn	32
2.3	Bảng kết quả tìm nghiệm của phương trình $x^3 - x - 1 = 0$ trên đoạn $[1, 2]$ bằng phương pháp Newton	37
3.1	Bảng liệt kê một số phương pháp trực tiếp giải hệ phương trình tuyến tính.	58
3.2	Bảng liệt kê một số phương pháp phân rã LU phổ biến.	69
3.3	Bảng kết quả sử dụng phương pháp lặp Jacobi để giải hệ phương trình tuyến tính $Ax = b$ sau 7 bước lặp.	110
4.1	Bảng tính toán hệ số cho đa thức nội suy Newton với $n = 5$	140
4.2	Bảng tính đa thức nội suy Neville với $n = 4$	145
4.3	Bảng xấp xỉ bình phương nhỏ nhất với trường hợp dữ liệu có trọng số. .	159
5.1	Bảng các hệ số của xấp xỉ sai phân trung tâm với độ chính xác $\mathcal{O}(h^2)$. .	170
5.2	Bảng các hệ số của xấp xỉ sai phân tiến với độ chính xác $\mathcal{O}(h)$	171
5.3	Bảng các hệ số của xấp xỉ sai phân hữu hạn lùi với độ chính xác $\mathcal{O}(h)$. .	172
5.4	Bảng các hệ số của xấp xỉ sai phân hữu hạn tiến với độ chính xác $\mathcal{O}(h^2)$. .	173
5.5	Bảng các hệ số của xấp xỉ sai phân hữu hạn lùi với độ chính xác $\mathcal{O}(h^2)$. .	173
5.6	Tính e^{-x} tại $x = 1$ bằng công thức sai phân hữu hạn trung tâm. . . .	174
5.7	Bảng liệt kê một số tập các đa thức trực giao	199
5.8	Bảng xác định một số tập các đa thức trực giao dưới dạng đệ quy . . .	200
5.9	Bảng tính phương pháp cầu phương Gauss-Legendre trong (5.35) . . .	203
5.10	Bảng tính phương pháp cầu phương Gauss-Laguerre trong (5.43) . . .	205
5.11	Bảng tính phương pháp cầu phương Gauss-Hermit trong (5.45) . . .	206
5.12	Bảng tính phương pháp cầu phương Gauss với tính kỳ dị logarit trong (5.47)	207
5.13	Các điểm tích phân, các trọng số và bậc tương ứng trong phương pháp cầu phương trên phần tử tam giác	224

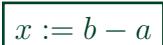
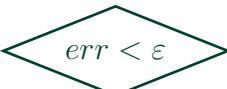
DANH SÁCH HÌNH VẼ

2.1 Sơ đồ khối của phương pháp chia đôi	20
2.2 Minh họa quá trình xấp xỉ nghiệm cho phương trình $x^3 - x - 1 = 0$ bằng phương pháp chia đôi.	23
2.3 Minh họa phương pháp dây cung khi $f'(x) > 0$	25
2.4 Minh họa phương pháp dây cung khi $f'(x) < 0$	26
2.5 Sơ đồ khối của phương pháp lặp dây cung	27
2.6 Đồ thị minh họa nghiệm của phương trình $f(x) = 0$ và $x = \varphi(x)$. .	29
2.7 Minh họa phương pháp lặp xấp xỉ nghiệm phương trình.	29
2.8 Sơ đồ khối của phương pháp lặp xấp xỉ nghiệm thực của phương trình	30
2.9 Sơ đồ khối của phương pháp lặp Newton-Raphson xấp xỉ nghiệm thực của phương trình	34
2.10 Minh họa phương pháp lặp Newton	36
2.11 Sơ đồ khối tính giá trị của đa thức và giá trị đạo hàm bậc một bậc hai của đa thức	42
2.12 Sơ đồ khối của phương pháp lặp Laguerre xấp xỉ 0-điểm đa thức . . .	45
3.1 Sơ đồ khối các thủ tục giải hệ phương trình tuyến tính dưới dạng tam giác	64
3.2 Sơ đồ khối giải hệ phương trình tuyến tính bằng phương pháp khử Gauss	65
3.3 Sơ đồ khối giải hệ phương trình tuyến tính bằng phương pháp phân rã Doolittle (LU)	72
3.4 Sơ đồ khối giải hệ phương trình tuyến tính bằng phương pháp phân rã Cholesky	79
3.5 Sơ đồ khối phân rã ma trận ba đường chéo bằng phương pháp phân rã LU	84
3.6 Sơ đồ khối giải hệ phương trình tuyến tính ba đường chéo bằng phương pháp phân rã LU	85
3.7 Sơ đồ khối phân rã ma trận năm đường chéo đối xứng bằng phương pháp phân rã LU	90
3.8 Sơ đồ khối giải hệ phương trình tuyến tính năm đường chéo đối xứng bằng phương pháp phân rã LU	93

3.9	Sơ đồ khối giải hệ phương trình tuyến tính bằng phương pháp khử Gauss kết hợp đổi hàng xoay	100
3.10	Sơ đồ khối phân rã ma trận hệ số bằng phương pháp LU kết hợp đổi hàng xoay	102
3.11	Sơ đồ khối giải hệ phương trình tuyến tính bằng phương pháp phân rã LU kết hợp đổi hàng xoay	103
3.12	Sơ đồ khối của phương pháp Gauss-Seidel giải hệ phương trình tuyến tính	113
3.13	Sơ đồ khối của phương pháp gradient liên hợp giải hệ phương trình tuyến tính	118
4.1	Đồ thị minh họa hai phương pháp xấp xỉ dữ liệu: phương pháp nội suy và xấp xỉ đường cong	134
4.2	Hình minh họa về các hàm cơ bản bậc 2 để tính đa thức nội suy Lagrange.	136
4.3	Sơ đồ khối tính đa thức nội suy Lagrange qua n điểm dữ liệu cho trước	137
4.4	Sơ đồ khối tính đa thức nội suy Newton qua n điểm dữ liệu cho trước	138
4.5	Sơ đồ khối tính hệ số của đa thức nội suy Newton qua n điểm dữ liệu cho trước	140
4.6	Sơ đồ khối tính đa thức nội suy Neville qua n điểm dữ liệu cho trước .	144
4.7	Đồ thị minh họa đa thức nội suy bậc cao bị dao động khi xấp xỉ nhiều điểm dữ liệu	146
4.8	Đồ thị minh họa phép nội suy có thể không theo xu hướng dữ liệu .	146
4.9	Sơ đồ khối tính hệ số của đa thức xấp xỉ bởi n điểm dữ liệu cho trước bằng phương pháp bình phương nhỏ nhất	153
4.10	Sơ đồ khối tính độ lệch chuẩn của phương pháp bình phương nhỏ nhất khi xấp xỉ n điểm dữ liệu cho trước bởi đa thức bậc m	154
5.1	Đa thức xấp xỉ hàm $f(x)$	181
5.2	Đồ thị minh họa công thức hình thang	182
5.3	Đồ thị minh họa công thức hình thang kết hợp	183
5.4	Sơ đồ khối của phương pháp hình thang kết hợp tính tích phân xác định	184
5.5	Sơ đồ khối của phương pháp hình thang đệ quy tính tích phân xác định	187
5.6	Đồ thị minh họa quá trình tích phân xác định bằng công thức Simpson	189
5.7	Sơ đồ khối tính tích phân xác định bằng phương pháp Simpson 1/3 .	190

5.8	Sơ đồ khôi tính tích phân xác định bằng phương pháp tích phân Romberg	194
5.9	Sơ đồ khôi tính tích phân xác định bằng phương pháp cầu phương Gaussian sử dụng đa thức trực giao Legendre	208
5.10	Đồ thị minh họa mô hình phần tử hữu hạn của miền lấp tích phân kép tùy ý	214
5.11	Ánh xạ một hình tứ giác vào hình chữ nhật chuẩn	215
5.12	Sơ đồ khôi tính tích phân bội trên miền tứ giác bằng phương pháp cầu phương Gaussian	217
5.13	Hình minh họa miền lấp tích phân bội trên hình tam giác	222
5.14	Các điểm tích phân trên phần tử tam giác	223
6.1	Sơ đồ khôi giải phương trình vi phân bằng phương pháp Taylor bậc m	236
6.2	Đồ thị minh họa quá trình xấp xỉ nghiệm phương trình vi phân bằng công thức Euler	242
6.3	Sơ đồ khôi giải phương trình vi phân bằng phương pháp Euler	242
6.4	Đồ thị minh họa quá trình xấp xỉ nghiệm của phương trình vi phân bằng công thức Euler cải tiến	244
6.5	Sơ đồ khôi giải phương trình vi phân bằng phương pháp Euler cải tiến	245
6.6	Sơ đồ khôi giải phương trình vi phân bằng phương pháp Runge - Kutta bậc-bốn	247
6.7	Đồ thị biểu diễn phương pháp trung điểm	265
6.8	Lưới sử dụng trong phương pháp trung điểm	266
6.9	Sơ đồ khôi giải phương trình vi phân bằng công thức trung điểm	267
6.10	Sơ đồ khôi giải phương trình vi phân bằng công thức trung điểm kết hợp ngoại suy Richardson	268
6.11	Sơ đồ khôi giải phương trình vi phân bằng thuật toán Bulirsch-Stoer .	271

DANH MỤC CÁC KÝ HIỆU VÀ CHỮ VIẾT TẮT

KÝ HIỆU	Ý NGHĨA
\mathbb{N}	Tập các số tự nhiên
\mathbb{Z}	Tập các số nguyên
\mathbb{R}	Tập các số thực
\mathbb{C}	Tập các số phức
A, B, C, \dots	Các ma trận
A^T	Ma trận chuyển vị của A
I	Ma trận đơn vị
$\ A\ $	Chuẩn của ma trận A
$\text{cond}(A)$	Số điều kiện của ma trận A
x, y, \dots	Các vectơ
i	Đơn vị ảo
x_k	Thành phần thứ k của vectơ x
$x^{(k)}$	Giá trị xấp xỉ của bước thứ k của vectơ x
$x \cdot y$	Tích vô hướng của các vectơ x, y
 BEGIN-END	Bắt đầu, kết thúc một thuật toán trong sơ đồ khôi
 Input/Print	Nhập số liệu/ in kết quả trong sơ đồ khôi
 $x := b - a$	Tính toán trong sơ đồ khôi
 $i = 1, 2, \dots$	Chu trình trong sơ đồ khôi
 $err < \varepsilon$	Kiểm tra điều kiện trong sơ đồ khôi

MỞ ĐẦU

PHƯƠNG PHÁP TÍNH LÀ GÌ VÀ TẠI SAO BẠN NÊN NGHIÊN CỨU PHƯƠNG PHÁP TÍNH?

Phương pháp tính là các kỹ thuật mà theo đó các bài toán được xây dựng sao cho chúng có thể được giải bằng các phép toán số học và logic. Các phép tính này được thực hiện một cách chính xác và hiệu quả bởi các máy tính kỹ thuật số nên các kỹ thuật trong phương pháp tính rất phù hợp cho việc tính toán bằng máy tính. Do vậy, phương pháp tính đôi khi được gọi là *toán học máy tính*.

Trước khi máy tính ra đời, việc thực hiện tính toán một số lượng lớn các phép toán số học và logic tốn nhiều thời gian và nhảm chán nên phương pháp tính chưa được quan tâm. Tuy nhiên, với sự ra đời của máy tính kỹ thuật số ngày càng nhanh và phổ biến, vai trò của phương pháp tính trong việc giải quyết các vấn đề kỹ thuật và khoa học đã trở nên quan trọng. Do vậy, phương pháp tính đã trở thành một phần nền tảng giáo dục cơ bản cho mỗi kỹ sư và mỗi nhà khoa học. Bên cạnh đó, một số lý do sau giải thích cho việc bạn cần nghiên cứu phương pháp tính.

1. Phương pháp tính giúp mở rộng các bài toán mà bạn có thể giải quyết. Phương pháp tính trình bày một số thuật toán có khả năng xử lý các hệ phương trình có kích thước lớn, phương trình phi tuyến và phức tạp mà những phương trình này không phải là hiếm gặp trong kỹ thuật và thường thì không thể giải bằng các phương pháp giải tích. Như vậy, phương pháp tính giúp nâng cao kỹ năng giải quyết vấn đề của bạn.
2. Phương pháp tính cho bạn sự hiểu biết sâu sắc về một số chương trình thương mại. Trong cuộc sống, rất nhiều chương trình thương mại được xây dựng dựa trên nền tảng là các thuật toán trong phương pháp tính. Nếu bạn nắm vững các thuật toán này, bạn sẽ có sự hiểu biết sâu sắc và áp dụng các chương trình này một cách phù hợp và hiệu quả.
3. Phương pháp tính giúp giải quyết hiệu quả nhiều bài toán. Nhiều bài toán không thể được tiếp cận bằng cách sử dụng các chương trình thương mại. Nếu bạn là người có kiến thức về phương pháp tính và thành thạo lập trình máy tính, bạn có thể thiết kế các chương trình của riêng mình để giải quyết vấn đề mà không phải mua các phần mềm đắt tiền hoặc thuê người khác lập trình giúp.

4. Phương pháp tính là một phương pháp hiệu quả để học sử dụng máy tính. Bởi vì các thuật toán trong phương pháp tính được thiết kế rõ ràng để thực hiện trên máy tính nên nó rất tốt cho việc minh họa quá trình hoạt động của máy tính. Khi bạn thực hiện lập trình thành công một thuật toán trong phương pháp tính trên máy tính, bạn có thể áp dụng chúng để giải quyết nhiều bài toán khó tương tự trong thực tế. Từ đó, bạn sẽ hiểu được về cách mà máy tính hoạt động và các có thể tận dụng các thế mạnh mà máy tính mang lại.
5. Phương pháp tính cung cấp một phương pháp để bạn củng cố kiến thức toán học. Bởi vì các thuật toán trong phương pháp tính được xây dựng trên cơ sở thay thế các phép toán phức tạp bằng các phép toán cơ bản. Quá trình này đòi hỏi biết được đặc điểm của các phép tính này, từ đó, bạn có thể củng cố nền tảng kiến thức toán học của mình.

TẠI SAO BẠN NÊN CHỌN CUỐN SÁCH NÀY MÀ KHÔNG PHẢI LÀ CUỐN SÁCH KHÁC?

Phương pháp tính đã được đưa vào giảng tại các trường đại học khoa học và đại học kỹ thuật ở Việt Nam từ khá sớm. Do yêu cầu về mức độ áp dụng và nền tảng toán học của sinh viên các trường đại học là khác nhau nên có rất nhiều sách về phương pháp tính để phù hợp với người học trong đó có một số sách phổ biến nhất là [1, 2, 3].

Tài liệu mà bạn đang cầm trên tay có thể nói là “*trung bình*” của [1, 2, 3]. Nghĩa là nếu bạn là người có nền tảng toán học vừa phải, cần tìm hiểu giải số các bài toán bằng cách sử dụng các chương trình MATLAB hoặc dựa vào sơ đồ khối để rèn luyện kỹ năng lập trình thì đây chính là cuốn sách bạn cần.

Trong tài liệu này, chúng tôi bỏ qua một số thuật ngữ chính xác, phức tạp của toán học, cố gắng mô tả các phương pháp một cách đơn giản để phù hợp với đối tượng là bạn đọc với nền tảng toán học vừa phải. Sau mỗi phương pháp số, chúng tôi thực hiện một số ví dụ được tính toán bằng tay một cách chi tiết để giúp bạn đọc có thể hiểu được nội dung phương pháp. Các phương pháp trong tài liệu được minh họa bằng sơ đồ khối giúp cho bạn đọc có thể sử dụng để lập trình bằng ngôn ngữ lập trình yêu thích của mình.

Theo chúng tôi, ngôn ngữ lập trình MATLAB có thể nói là một công cụ rất quan trọng đối với mỗi kỹ sư trong việc xử lý các bài toán chuyên môn. Do vậy chúng tôi chọn MATLAB làm ngôn ngữ lập trình để tính toán bằng máy tính cho các phương pháp số. Bạn đọc sẽ được tiếp cận một hệ thống các chương trình MATLAB (M-file) để tính toán cho tất cả các phương pháp số trong tài liệu này. Chúng tôi hy vọng

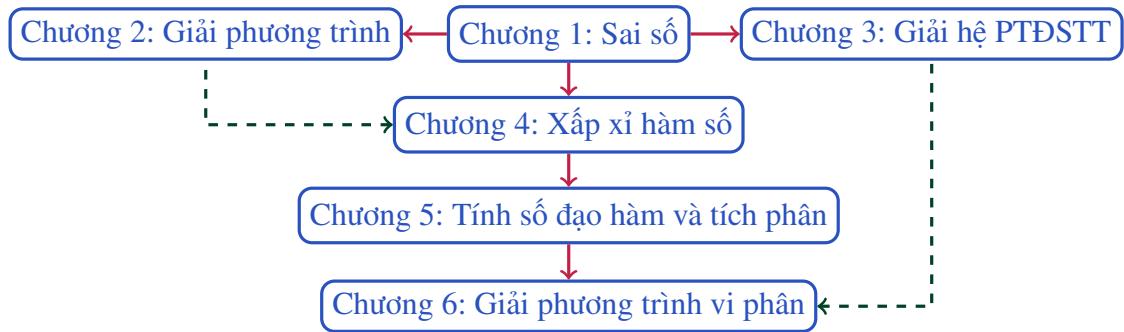
vệc này sẽ giúp củng cố và nâng cao phần nào đó kỹ năng sử dụng MATLAB để giải quyết vấn đề chuyên môn cho bạn đọc.

NỘI DUNG CỦA TÀI LIỆU NÀY LÀ GÌ VÀ BẠN NÊN HỌC NHƯ THẾ NÀO?

Nội dung của tài liệu này được tham khảo chủ yếu từ [2, 4, 5, 6]. Một số sơ đồ khái của các phương pháp được tham khảo từ [2], nội dung và các bài tập được tham khảo từ [4, 5, 6]. Tuy nhiên, để các bạn có thể tham khảo nhiều tài liệu phương pháp tính khác mà không phải hệ thống lại nội dung kiến thức, tài liệu này được chúng tôi sắp xếp nội dung theo kiểu “truyền thống” như trong tài liệu [1]. Tức là chương đầu tiên sẽ giới thiệu về sai số và cách viết sai số, năm nội dung cơ bản sẽ lần lượt được trình bày trong năm chương còn lại. Cụ thể như sau:

- Chương 2 giới thiệu về một số phương pháp xấp xỉ nghiệm thực của phương trình $f(x) = 0$.
- Chương 3 đề cập đến các phương pháp giải hệ phương trình đại số tuyến tính. Các phương pháp này được chia thành hai nhóm: các phương pháp tìm nghiệm chính xác (phương pháp Gauss, phân rã LU, ...) và các phương pháp lặp.
- Chương 4 trình bày một số phương pháp xấp xỉ hàm số trong nhóm hai phương pháp là phương pháp nội suy và phương pháp bình phương nhỏ nhất.
- Chương 5 tính số đạo hàm và tích phân. Bên cạnh các phương pháp cơ bản để xấp xỉ đạo hàm và tích phân, chúng tôi giới thiệu kỹ thuật ngoại suy để nâng độ chính xác cho một số phương pháp.
- Chương 6 đề cập đến một số phương pháp giải phương trình vi phân thường.

Do mỗi chương của cuốn sách này giải quyết trọn vẹn một vấn đề nên việc đầu tiên khi bắt đầu học một chương là bạn cần nắm được về nội dung của bài toán tổng quát mà chương này cần giải quyết. Mỗi liên quan giữa các chương trong sách được biểu diễn bằng sơ đồ bên dưới, bạn nên tham khảo để có thể chọn đọc các chương tùy theo nhu cầu. Trước tiên bạn nên nắm được nội dung Chương 1, sau đó bạn có thể chọn đọc Chương 2 hoặc Chương 3 hoặc Chương 4, 5, 6 tùy theo yêu cầu nội dung cần thiết cho công việc của bạn.



Sau đó, với mỗi bài, bạn cố gắng ghi nhớ được ý tưởng và ý nghĩa của mỗi phương pháp. Thực hiện tính toán lại các ví dụ minh họa để hiểu hơn về phương pháp. Cách tốt nhất là bạn nên copy đoạn code MATLAB trong bài chạy số để hiểu một cách trực quan. Cuối cùng, dựa vào sơ đồ khái của phương pháp, bạn hãy lập trình chương trình giải bài toán bằng ngôn ngữ yêu thích của mình. Cuối mỗi chương bạn nên ôn lại các phương pháp bằng cách thực hiện một số bài tập mà chúng tôi đã chuẩn bị, trong đó, những bài tập có ký hiệu là những bài tập cần thực hiện bằng lập trình.

CHƯƠNG 1

SAI SỐ

1.1	Giới thiệu	6
1.2	Sai số và cách viết xấp xỉ	6
1.2.1	Sai số tuyệt đối và sai số tương đối	6
1.2.2	Sai số quy tròn	8
1.2.3	Cách viết xấp xỉ	9
1.3	Các quy tắc tính sai số	10
1.3.1	Sai số tính toán	10
1.3.2	Sai số phương pháp	12

1.1 GIỚI THIỆU

Trong tính toán, chúng ta luôn phải làm việc với các giá trị gần đúng của các đại lượng, vậy nên vấn đề đầu tiên cần nghiên cứu là sai số. Trong tính toán ta thường gặp phải 4 loại sai số sau:

a, Sai số giả thiết - Do mô hình hoá, lý tưởng hoá các bài toán thực tế. Sai số này không loại trừ được.

b, Sai số phương pháp - Các bài toán thường rất phức tạp không thể giải đúng được mà phải sử dụng các phương pháp gần đúng. Sai số này sẽ được nghiên cứu cho từng trường hợp cụ thể.

c, Sai số các số liệu - Các số liệu thường được thu bằng thực nghiệm do đó có sai số là hiển nhiên và không thể loại bỏ.

d, Sai số tính toán - Các số vốn đã có sai số, còn thêm sai số khi tính toán thu gọn số gọi là sai số tính toán.

1.2 SAI SỐ VÀ CÁCH VIẾT XẤP XỈ

Sai số tính toán sinh ra từ việc sử dụng các xấp xỉ để biểu diễn các phép toán và số lượng chính xác. Đối với sai số này, mỗi quan hệ giữa giá trị chính xác hoặc giá trị đúng với giá trị xấp xỉ có thể được xác định bởi

$$\text{Giá trị chính xác} = \text{Giá trị xấp xỉ} + \text{Sai số} \quad (1.1)$$

Từ (1.1) ta thấy rằng sai số tính toán chính là khoảng cách giữa giá trị chính xác và giá trị xấp xỉ, tức là

$$\Delta = \text{Giá trị chính xác} - \text{Giá trị xấp xỉ}, \quad (1.2)$$

với Δ là giá trị chính xác của sai số.

1.2.1 SAI SỐ TUYỆT ĐỐI VÀ SAI SỐ TƯƠNG ĐỐI

ĐỊNH NGHĨA 1.2.1 (Sai số tuyệt đối). Nếu đại lượng A được xấp xỉ bởi đại lượng gần đúng a thì ta ký hiệu $A \approx a$. Khi đó $\Delta = |A - a|$ được gọi là *sai số tuyệt đối* của a hoặc là sai số thực sự của a . Tuy nhiên, vì thông thường ta không biết đại lượng A nên sai số tuyệt đối Δ cũng không biết. Do đó, số dương Δ_a được gọn là *sai số tuyệt*

đối giới hạn của a nếu

$$|A - a| \leq \Delta_a. \quad (1.3)$$

Khi đó, ta có thể viết A dưới dạng

$$A = a \pm \Delta_a \quad (1.4)$$

Ta thấy rằng, nếu Δ_a là sai số tuyệt đối giới hạn thì các giá trị lớn hơn Δ_a cũng là sai số tuyệt đối giới hạn. Do đó, trong các phép xấp xỉ, người ta cố gắng tìm Δ_a càng nhỏ thì càng tốt. Trong thực tế, khi Δ_a đủ nhỏ thì ta có thể coi như là sai số tuyệt đối của a .

VÍ DỤ 1.2.1 (Xấp xỉ cho π). Xét phép xấp xỉ $a = 3.14$ cho giá trị $A = \pi$. Ta thấy rằng sai số tuyệt đối của phép xấp xỉ này là $\Delta = |A - a| = |\pi - 3.14|$ không thể biết được chính xác vì π là số vô tỉ. Tuy nhiên, ta thấy rằng $3.14 < \pi < 3.15$ nên $|\pi - 3.14| < 0.01$. Do đó, ta có thể lấy sai số tuyệt đối giới hạn $\Delta_a = 0.01$. Hiển nhiên $|\pi - 3.14| < 0.02$ nên 0.02 cũng là một sai số tuyệt đối giới hạn nhưng do $0.01 < 0.02$ nên ta chọn $\Delta_a = 0.01$.

Sai số tuyệt đối cho ta biết độ lớn của một phép xấp xỉ nhưng chưa thể hiện được chất lượng của phép xấp xỉ này. Chẳng hạn đo thể tích của lon Pepsi 330 ml bằng phương pháp 1 ta có xấp xỉ là 299 ± 2 ml và đo thể tích của lon Pepsi 500 ml bằng phương pháp 2 ta có xấp xỉ là 499 ± 2 ml. Rõ ràng ta thấy phương pháp 2 chính xác hơn phương pháp 1 dù cả hai phương pháp đều cho ta cùng sai số tuyệt đối. Do vậy, chất lượng của một xấp xỉ được xác định bằng khái niệm sau.

ĐỊNH NGHĨA 1.2.2. Xét phép xấp xỉ $A = a \pm \Delta_a$. Khi đó, sai số tương đối giới hạn của a là số thực δ_a xác định bởi

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

Từ công thức (1.4) và (1.5), mối liên hệ giữa số đúng A và sai số tương đối δ_a được biểu diễn dưới dạng sau.

$$A = a(1 \pm \delta_a). \quad (1.6)$$

Do ta coi Δ_a là sai số tuyệt đối nên trong thực tế, ta gọi δ_a là sai số tương đối của a .

VÍ DỤ 1.2.2. Xét xấp xỉ trong Ví dụ 1.2.1, $\pi = 3.14 \pm 0.01$. Sai số tương đối của phép xấp xỉ là

$$\delta_a = \frac{0.01}{3.14} = 0.003185 = 0.3185\%.$$

Khi đó, π cũng có thể biểu diễn dưới dạng $\pi = 3.14(1 \pm 0.003185)$.

1.2.2 SAI SỐ QUY TRÒN

Ta biết rằng, một số thập phân a luôn được biểu diễn dưới dạng dãy liên tiếp các chữ số tự nhiên $0, 1, \dots, 9$ (có thể có dấu “,” để phân cách giữa phần nguyên và phần thập phân) dưới dạng

$$a = \pm a_n a_{n-1} \dots a_0, a_{-1} \dots a_{-s} \quad (1.7)$$

$$\pm (a_n \cdot 10^n + a_{n-1} \cdot 10^{n-1} + \dots + a_0 \cdot 10^0 + a_{-1} \cdot 10^{-1} + \dots + a_{-s} \cdot 10^{-s}), \quad (1.8)$$

trong đó $n, s \in \mathbb{N}$, $a_k \in \{0, 1, \dots, 9\}$, $k = n, n-1, \dots, -s$.

Ta thấy rằng nếu $s < \infty$ thì a là số hữu tỉ, nếu $s = \infty$ thì a là số thập phân vô hạn. Tuy nhiên, khi s lớn thì việc biểu diễn a chính xác theo công thức (1.7) rất bất tiện và đôi khi không cần thiết vì ta có thể làm tròn số a bằng cách ngắt bỏ một số chữ số cuối của dãy số đi để nhận được số \hat{a} (ngắn hơn a), gọi là số quy tròn của a . Việc quy tròn số được thực hiện theo quy tắc sau đây.

QUY TẮC 1.2.1 (Quy tròn số). Giả sử số a được biểu diễn dưới dạng

$$a = \pm a_n \dots a_{k+1} a_k a_{k-1} \dots a_{-s}.$$

Nếu ta muốn quy tròn số a bằng cách bỏ đi các chữ số từ chữ số thứ a_{k-1} đến cuối dãy để được số quy tròn $\hat{a} = \pm a_n \dots a_{k+1} \hat{a}_k$ thì chữ số \hat{a}_k được xác định theo quy tắc sau.

(i) Nếu $a_{k-1} < 5$ thì $\hat{a}_k = a_k$;

(ii) Nếu $a_{k-1} > 5$ thì $\hat{a}_k = a_k + 1$;

(iii) Nếu $a_{k-1} = 5$ thì $\hat{a}_k = a_k$ khi a_k là số chẵn và $\hat{a}_k = a_k + 1$ khi a_k là số lẻ.

Việc quy tròn theo quy tắc trên là một phép xấp xỉ của \hat{a} cho a với sai số $|a - \hat{a}|$ không quá $0.5 \cdot 10^k$ (sai số quy tròn không quá nửa đơn vị của hàng giữ lại sau cùng). Do vậy, nếu a là xấp xỉ của A với sai số tuyệt đối Δ_a thì \hat{a} là xấp xỉ của A với sai số tuyệt đối $\Delta_{\hat{a}}$ không quá $\Delta_a + 0.5 \cdot 10^k$ vì

$$\Delta_{\hat{a}} = |A - \hat{a}| = |A - a + a - \hat{a}| \leq |A - a| + |a - \hat{a}| \leq \Delta_a + 0.5 \cdot 10^k. \quad (1.9)$$

Ví dụ sau đây sẽ minh họa việc quy tròn số theo quy tắc trên.

VÍ DỤ 1.2.3. Gọi $a = 1.414213562$ là một xấp xỉ của $A = \sqrt{2}$ với sai số tuyệt đối $\Delta_a = 0.0000000005$. Ta có thể xấp xỉ số a bởi quy tắc quy tròn như sau:

$$\begin{aligned} a &= 1.414213562 \approx 1.41421356 \\ &\approx 1.414214 \\ &\approx 1.4142 \\ &\approx 1.41 \\ &\approx 1.4. \end{aligned}$$

Khi quy tròn a để được số $\hat{a} = 1.4$, ta có sai số do quy tròn gây ra là 0.014213562 . Nếu dùng \hat{a} để xấp xỉ cho $A = \sqrt{2}$ thì sai số tuyệt đối $\Delta_{\hat{a}}$ nhỏ hơn 0.0142135625 .

1.2.3 CÁCH VIẾT XẤP XỈ

Cho số a là xấp xỉ của A với sai số tuyệt đối Δ_a . Công thức (1.4) và (1.6) cho ta hai cách viết cho việc xấp xỉ một số. Tuy nhiên, khi sử dụng cách viết này đôi khi sẽ rất bất tiện và khó khăn trong việc tính toán. Hơn nữa, khi a được biểu diễn bởi dãy quá nhiều chữ số ta có thể quy tròn cho dễ tính toán và việc quy tròn này làm tăng sai số tính toán. Vấn đề đặt ra là ta nên quy tròn đến bao nhiêu chữ số để sai số quy tròn này phù hợp với sai số tuyệt đối đã biết. Vì vậy, để đơn giản, số xấp xỉ làm tròn được viết dựa trên quy tắc sau: *mọi chữ số có nghĩa là chữ số đáng tin*, tức là ta loại bỏ các chữ số có nghĩa nhưng không đáng tin trong khi quy tròn số. Để minh họa quy tắc này, trước tiên ta định nghĩa các khái niệm *chữ số có nghĩa* và *chữ số đáng tin* trong định nghĩa sau.

ĐỊNH NGHĨA 1.2.3. Cho a là xấp xỉ của A với sai số tuyệt đối Δ_a . Giả sử a được biểu diễn dưới dạng thập phân trong công thức (1.7). Khi đó ta có:

- (i) Những chữ số được gọi là *có nghĩa* trong (1.7) là những chữ số tính từ chữ số khác không đầu tiên kể từ trái sang phải;
- (ii) Chữ số $a_k, k = n, n - 1, \dots, -s$ trong (1.7) được gọi là *đáng tin* nếu thoả mãn

$$\Delta_a \leq 0.5 \cdot 10^k. \quad (1.10)$$

Ngược lại, số a_k gọi là số *đáng nghi*.

Ta lưu ý rằng nếu a_k là chữ số có nghĩa thì mọi số bên phải của a_k trong dãy số đều là số có nghĩa. Nếu a_k là số đáng tin thì mọi số bên trái của a_k trong dãy số đều

là số đáng tin, nếu a_k là số đáng nghi thì mọi số bên phải của a_k trong dãy số đều là số đáng nghi. Ví dụ sau đây sẽ minh họa cho các khái niệm trên.

VÍ DỤ 1.2.4. Gọi $a = 0.0201500$ là xấp xỉ cho số A với sai số tuyệt đối $\Delta_a = 0.00001$. Khi đó, các chữ số có nghĩa của a là: 2; 0; 1; 5; 0; 0. Số a có thể viết dưới dạng

$$a = 2 \cdot 10^{-2} + 0 \cdot 10^{-3} + 1 \cdot 10^{-4} + 5 \cdot 10^{-5} + 0 \cdot 10^{-6} + 0 \cdot 10^{-7}.$$

Vì $0.5 \cdot 10^{-5} < \Delta_a = 0.00001 < 0.5 \cdot 10^{-4}$ nên số 1 là số đáng tin và số 5 là số đáng nghi. Vậy các chữ số đáng tin của a là: 0; 0; 2, 0, 1, các chữ số đáng nghi của a là: 5; 0; 0.

Vậy theo quy tắc viết số xấp xỉ, a có thể quy tròn thành $\hat{a} = 0.0202$. Khi đó, sai số tuyệt đối của \hat{a} luôn nhỏ hơn 0.0001 (1 đơn vị hàng cuối cùng).

Kể từ đây trong tài liệu này, nếu không có chú thích gì thì ta hiểu mọi số thập phân đều được viết dựa trên nguyên tắc: mọi chữ số có nghĩa đều là chữ số đáng tin.

1.3 CÁC QUY TẮC TÍNH SAI SỐ

Diện tích bề mặt trái đất của chúng ta được xấp xỉ là 510100000 km^2 . Để thực hiện tính toán này, trước tiên người ta giả thiết rằng trái đất là một khối cầu với bán kính $R = 6317 \text{ km}$ và tính toán bằng cách sử dụng công thức

$$S = 4\pi R^2.$$

Tuy nhiên ta thấy quá trình tính toán này sẽ dẫn đến sai số. Trước tiên, sai số xảy ra khi ta coi Trái đất là khối cầu. Đây là sai số giả thuyết và sai số này không loại trừ được. Tiếp theo là xác định đường kính Trái đất bằng các phương pháp đo sẽ dẫn đến sai số, sai số này phụ thuộc vào phương tiện và các phương pháp đo. Đây gọi là sai số số liệu và sai số này cũng không thể loại bỏ. Cuối cùng, do quy tròn số (chẳng hạn quy tròn số π) nên việc tính toán sẽ xuất hiện sai số, đây được gọi là sai số tính toán. Mục sau đây sẽ giới thiệu một số công thức cơ bản cho sai số tính toán.

1.3.1 SAI SỐ TÍNH TOÁN

Giả sử rằng ta cần tính đại lượng \hat{y} theo công thức

$$\hat{y} = f(\hat{x}_1, \hat{x}_2, \dots, \hat{x}_n), \quad (1.11)$$

với $\hat{\mathbf{x}} = (\hat{x}_1, \hat{x}_2, \dots, \hat{x}_n)$ là số đúng chưa biết. Giả sử \hat{x}_i được xấp xỉ bởi x_i với sai số tuyệt đối Δ_{x_i} , $i = 1, \dots, n$. Nếu đặt $y = f(\mathbf{x})$ thì ta có

$$|y - \hat{y}| = |f(\mathbf{x}) - f(\hat{\mathbf{x}})| = \sum_{i=1}^n |f'_{x_i}(\bar{\mathbf{x}})| |x_i - \hat{x}_i|,$$

trong đó $f'_{x_i}(\bar{\mathbf{x}})$ là đạo hàm riêng của hàm f theo biến thứ i lấy tại $\bar{\mathbf{x}}$, với $\bar{\mathbf{x}}$ nằm trên đoạn $\overline{\mathbf{x}\hat{\mathbf{x}}}$. Nếu ta xấp xỉ $f'_{x_i}(\bar{\mathbf{x}}) = f'_{x_i}(x_1, x_2, \dots, x_n)$ và thay $|x_i - \hat{x}_i|$ bởi Δ_{x_i} vào công thức trên thì ta sẽ có công thức tính sai số tuyệt đối của y như sau

$$\Delta_y = \sum_{i=1}^n |f'_{x_i}(x_1, x_2, \dots, x_n)| \Delta_{x_i}. \quad (1.12)$$

Do đó, sai số tương đối của y được tính như sau

$$\delta_y = \sum_{i=1}^n \left| \frac{f'_{x_i}(x_1, x_2, \dots, x_n)}{f(x_1, x_2, \dots, x_n)} \right| \Delta_{x_i}. \quad (1.13)$$

Ta sẽ áp dụng các công thức (1.12) và (1.13) để tính sai số tuyệt đối và sai số tương đối cho một số hàm đơn giản trong các ví dụ sau.

VÍ DỤ 1.3.1 (Sai số của tổng). Xét hàm số $f(x_1, x_2) = x_1 + x_2$. Do $f'_{x_1} = f'_{x_2} = 1$ nên theo công thức (1.12), sai số tuyệt đối của y là

$$\Delta_y = \Delta_{x_1} + \Delta_{x_2}$$

và theo công thức (1.13), sai số tương đối của y là

$$\delta_y = \frac{\Delta_{x_1} + \Delta_{x_2}}{|x_1 + x_2|}.$$

VÍ DỤ 1.3.2 (Sai số của tích). Xét hàm số $f(x_1, x_2) = x_1 x_2$. Do $f'_{x_1}(x_1, x_2) = x_2$, $f'_{x_2}(x_1, x_2) = x_1$ nên theo công thức (1.12), sai số tuyệt đối của y là

$$\Delta_y = |x_2| \Delta_{x_1} + |x_1| \Delta_{x_2}$$

và theo công thức (1.13), sai số tương đối của y là

$$\delta_y = \frac{|x_2| \Delta_{x_1} + |x_1| \Delta_{x_2}}{|x_1 x_2|} = \frac{\Delta_{x_1}}{|x_1|} + \frac{\Delta_{x_2}}{|x_2|}.$$

Suy ra

$$\delta_y = \delta_{x_1} + \delta_{x_2}.$$

VÍ DỤ 1.3.3 (Diện tích mặt cầu). Tính sai số tuyệt đối và sai số tương đối của diện tích Trái đất (giả thiết Trái đất hình cầu) theo công thức

$$V = 4\pi R^2$$

nếu cho $R = 6317 \pm 10$ km và $\pi = 3.14159 \pm 0.000005$.

Do $V'_\pi = 4R^2$ và $V'_R = 8\pi R$ nên theo (1.12) và (1.13) ta có công thức tính sai số tương đối và sai số tuyệt đối của V là

$$\Delta_V = |4R^2| \Delta_\pi + |8\pi R| \Delta_R,$$

$$\delta_V = \frac{|4R^2| \Delta_\pi + |8\pi R| \Delta_R}{4\pi R^2} = \delta_\pi + 2\delta_R.$$

Thay $R = 6317$, $\Delta_R = 10$, $\pi = 3.14159$, $\Delta_\pi = 0.000005$ vào các công thức trên ta có

$$\Delta_V = 1588432.01218 \text{ km}^2 \text{ và } \delta_V = 0.003168.$$

1.3.2 SAI SỐ PHƯƠNG PHÁP

Khi thay thế việc tính toán một bài toán phức tạp bằng tính toán một bài toán đơn giản sẽ gây nên sai số, sai số này được gọi là *sai số phương pháp*. Việc xấp xỉ và tính toán sai số phương pháp phụ thuộc vào các bài toán cụ thể mà sẽ được đề cập ở các chương sau. Trong mục này, chúng ta sẽ minh họa cho việc tính toán sai số phương pháp bằng ví dụ sau.

VÍ DỤ 1.3.4. Tính giá trị của biểu thức sau với sai số không vượt quá 0.005

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

Ta thấy rằng quá trình tính toán sẽ được thực hiện bởi hai bước. Bước một là ta phải xấp xỉ biểu thức vô hạn A bởi một biểu thức hữu hạn A_n dưới dạng

$$A_n = \frac{1}{1^3} - \frac{1}{2^3} + \frac{1}{3^3} + \cdots + (-1)^{n-1} \frac{1}{n^3}.$$

Việc xấp xỉ này tạo ra sai số tuyệt đối của A_n là Δ_{A_n} . Bước hai là ta cần tính toán cụ thể cho A_n . Tuy nhiên, việc tính toán này cần phải làm tròn số trong các phân số nên ta chỉ nhận được giá trị a , là giá trị xấp xỉ của A_n , với sai số tuyệt đối Δ_a . Vì vậy, sai số tuyệt đối của quá trình tính toán này là $\Delta_{A_n} + \Delta_a$.

Do yêu cầu sai số không vượt quá 0.005 nên

$$|A - A_n| = \left| \frac{1}{(n+1)^3} - \frac{1}{(n+2)^3} + \cdots \right| < \frac{1}{(n+1)^3} < 0.005.$$

Ta thấy $n = 6$ là số tự nhiên nhỏ nhất thoả mãn biểu thức trên nên ta lấy $\Delta_{A_6} = 0.003$. Vậy ta cần tính biểu thức

$$A_6 = \frac{1}{1^3} - \frac{1}{2^3} + \frac{1}{3^3} - \frac{1}{4^3} + \frac{1}{5^3} - \frac{1}{6^3}.$$

Vì sai số cả quá trình $\Delta_{A_n} + \Delta_a \leq 0.005$ nên $\Delta_a \leq 0.002$. Ta lại biết rằng sai số của việc xấp xỉ cho A_6 được gây ra bởi 6 lần quy tròn số nên để $\Delta_A \leq 0.002$ thì mỗi lần quy tròn số này sai số phải nhỏ hơn $0.002/6 \approx 0.0003$, tức là mỗi phân số sẽ được làm tròn đến số thập phân thứ 3 sau dấu phẩy. Cụ thể ta có

$$\frac{1}{1^3} = 1; \frac{1}{2^3} = 0.125; \frac{1}{3^3} \approx 0.037; \frac{1}{4^3} \approx 0.016; \frac{1}{5^3} = 0.008; \frac{1}{6^3} \approx 0.005.$$

Do đó ta có

$$A \approx a = 1 - 0.125 + 0.037 - 0.016 + 0.008 - 0.005 = 0.899$$

với sai số không vượt quá 0.005.

TÓM TẮT CHƯƠNG 1

Nội dung chính của Chương 1 trình bày những vấn đề sau:

- Khái niệm về sai số tuyệt đối và sai số tương đối được trình bày trong Định nghĩa 1.2.1 và Định nghĩa 1.2.2. Khái niệm về số có nghĩa, số đáng tin được giới thiệu trong Định nghĩa 1.2.3.
- Làm tròn số thập phân theo Quy tắc 1.2.1 và biểu diễn số xấp xỉ dưới dạng các công thức (1.4), (1.6) hoặc bằng quy tắc trong Mục 1.2.3.
- Công thức tính sai số tuyệt đối và sai số tương đối cho hàm f được cho trong các công thức (1.12), (1.13) và áp dụng các công thức này cho một số trường hợp cơ bản.

BÀI TẬP CHƯƠNG 1

Bài tập 1.1. Hãy tính sai số tương đối của các số xấp xỉ của các góc sau khi biết rằng sai số tuyệt đối trong các phép đo là $1''$.

a. $a = 21^\circ 39' 3''$.

b. $b = 1^\circ 10''$.

Bài tập 1.2. Hãy xác định sai số tuyệt đối của các phép xấp xỉ sau

a. $a = 13226; \delta_a = 0,1\%$.

b. $b = 2,32; \delta_b = 0,7\%$.

Bài tập 1.3. Hãy xác định các chữ số đáng tin trong phép xấp xỉ sau

a. $a = 0,3941; \Delta_a = 0,25 \cdot 10^{-2}$.

b. $b = 38,2543; \Delta_b = 0,27 \cdot 10^{-2}$.

Bài tập 1.4. Hãy xác định các chữ số đáng tin trong phép xấp xỉ sau

a. $a = 1,8921; \delta_a = 0,1 \cdot 10^{-2}$.

b. $b = 22,351; \delta_b = 0,1$.

Bài tập 1.5. Hãy quy tròn các số dưới đây với 3 chữ số đáng tin và xác định sai số tuyệt đối, sai số tương đối của chúng.

a. 2,1514

b. 0,16152

c. 0,01204

d. -0,0015281

Bài tập 1.6. Hãy xác định giá trị của hàm số dưới đây cùng với sai số tuyệt đối và sai số tương đối ứng với giá trị của các đối số đã cho với mọi chữ số có nghĩa đều đáng tin.

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

b. $u = \frac{x+y^1}{z}$ với $x = 3,28, y = 0,932, z = 1,12$.

Bài tập 1.7. Tính tổng S sau đây với 3 chữ số lẻ thập phân đáng tin

$$S = \frac{1}{11} + \frac{1}{12} + \frac{1}{13} + \frac{1}{14} + \frac{1}{15} + \frac{1}{16} + \frac{1}{17}.$$

Bài tập 1.8. Tính số e theo công thức sau đây với sai số tuyệt đối không quá 10^{-4}

$$S = 1 + \frac{1}{1} + \frac{1}{2} + \cdots + \frac{1}{n} + \cdots$$

Bài tập 1.9. Tính $\sqrt{102} - \sqrt{101}$ chính xác tới 4 chữ số phần thập phân.

CHƯƠNG 2

TÍNH GẦN ĐÚNG NGHIỆM THỰC CỦA PHƯƠNG TRÌNH

2.1	Giới thiệu	16
2.2	Một số phương pháp khoảng	17
2.2.1	Phương pháp tìm kiếm gia tăng	18
2.2.2	Phương pháp chia đôi	19
2.2.3	Phương pháp dây cung	24
2.3	Các phương pháp mở	27
2.3.1	Phương pháp lặp	28
2.3.2	Phương pháp Newton-Raphson	32
2.4	0-diểm của đa thức	39
2.4.1	Giới thiệu	39
2.4.2	Tính giá trị đa thức	40
2.4.3	Giảm bậc đa thức	42
2.4.4	Phương pháp Laguerre	44
2.5	Hàm Matlab giải phương trình	49

2.1 GIỚI THIỆU

Khi học tại trường phổ thông, bạn đã học sử dụng công thức

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \quad (\text{PT2.1})$$

để giải phương trình

$$f(x) = ax^2 + bx + c = 0. \quad (\text{PT2.2})$$

Các giá trị tính từ (PT2.1) được gọi là “*nghiệm*” của phương trình (PT2.2). Chúng biểu diễn các giá trị của x thoả mãn (PT2.2), tức là làm $f(x)$ bằng không. Vì lý do này, đôi khi nghiệm còn được gọi là *0-diểm* của hàm số $y = f(x)$.

Dù công thức (PT2.1) rất thuận tiện để giải phương trình (PT2.2) nhưng trong thực tế thì có rất nhiều phương trình mà nghiệm không dễ dàng xác định như vậy. Trước khi máy tính kỹ thuật số ra đời, chỉ có một số ít phương trình có công thức tìm được nghiệm chính xác, một trong số ít này là trường hợp nghiệm được tìm bằng phương pháp trực tiếp như với (PT2.1). Mặc dù có một số phương trình có thể được giải trực tiếp nhưng có rất nhiều phương trình khác không thể giải được. Trong những trường hợp như vậy, sự thay thế duy nhất là tìm các kỹ thuật xấp xỉ nghiệm cho các phương trình này.

Về mặt hình học, nghiệm của phương trình $f(x) = 0$ chính là hoành độ giao điểm của đồ thị hàm số $y = f(x)$ với trục hoành. Do đó, một phương pháp trực quan để tìm nghiệm xấp xỉ là vẽ đồ thị hàm số $y = f(x)$ và tìm giá trị x xấp xỉ giao điểm của đồ thị này với trục hoành. Mặc dù các phương pháp đồ thị rất hữu ích để tìm được các xấp xỉ thô cho nghiệm, nhưng chúng bị hạn chế do thiếu độ chính xác. Một cách tiếp cận đơn giản khác là sử dụng *thử-sai*. Kỹ thuật này bắt đầu bằng cách đoán một giá trị x để xấp xỉ cho nghiệm và tính xem $f(x)$ có bằng không hay không. Nếu không (hầu như luôn luôn như vậy), một dự đoán khác của x được đưa ra và $f(x)$ một lần nữa được đánh giá để xác định xem giá trị mới có cung cấp ước tính gốc tốt hơn không. Quá trình được lặp lại cho đến khi thu được một kết quả x mà $f(x)$ gần bằng không với sai lệch chấp nhận được.

Vì các phương pháp trên phụ thuộc vào “may-rủi” nên rõ ràng là không hiệu quả và không thể đáp ứng các yêu cầu trong kỹ thuật và trong khoa học. Các phương pháp số hiện đại cũng sử dụng chiến lược trên nhưng được cải tiến sao cho giá trị của những lần thử sau của x càng gần nghiệm hơn. Với các phương pháp hiện đại này và

sự ra đời của máy tính kỹ thuật số, việc tìm nghiệm xấp xỉ của các phương trình ngày càng đơn giản và hiệu quả.

Tương tự như phương pháp thử-sai, cách tiếp cận mới cũng yêu cầu có một dự đoán ban đầu xấp xỉ cho nghiệm, sau đó các xấp xỉ nghiệm mới được tính toán bằng các bước lặp xuất phát từ dự đoán ban đầu này. Dựa vào các dự đoán ban đầu, có thể phân các phương pháp xấp xỉ nghiệm của phương trình thành hai tập:

- Các phương pháp khoảng: Đúng như tên gọi, các phương pháp này bắt đầu bằng các dự đoán ban đầu là một khoảng $[a, b]$ mà nghiệm của phương trình nằm trong khoảng này.
- Các phương pháp mở: Đây là các phương pháp bắt đầu với một hoặc nhiều dự đoán ban đầu nhưng những dự đoán này không yêu cầu trên một khoảng.

Đối với các bài toán giả định đúng¹, các phương pháp khoảng luôn có thể tính toán được một cách ổn định nhưng lại hội tụ rất chậm (tức là nghiệm xấp xỉ được tìm sau rất nhiều bước lặp). Ngược lại, các phương pháp mở không phải lúc nào cũng tính toán được nhưng các phương pháp này thường hội tụ rất nhanh.

Với cả hai lớp phương pháp, các dự đoán ban đầu luôn luôn được yêu cầu. Các dự đoán ban đầu này thường thì được xác định từ điều kiện vật lý của bài toán thực tế. Tuy nhiên, trong nhiều trường hợp, các dự đoán ban đầu có thể không tốt. Trong những trường hợp như vậy, các phương pháp khoảng sẽ được áp dụng để giải “sơ bộ” tìm dự đoán ban đầu tốt sau đó sẽ áp dụng các phương pháp mở để tận dụng tốc độ hội tụ của các phương pháp này.

2.2 MỘT SỐ PHƯƠNG PHÁP KHOẢNG

Giả sử phương trình $f(x) = 0$ có nghiệm trên $[a, b]$. Mục đích của các phương pháp khoảng là tìm đoạn con của đoạn $[a, b]$ chứa nghiệm của phương trình. Vì các phương pháp khoảng không yêu cầu hàm $f(x)$ có các tính chất đặc biệt nên hiệu quả tính toán không cao. Do đó, các phương pháp khoảng thường dùng để giải sơ bộ để tìm khoảng chứa nghiệm đủ nhỏ mà trên đó ta có thể xét các tính chất đặc biệt của hàm $f(x)$. Sau đó, các phương pháp có tốc độ hội tụ nhanh hơn sẽ tiếp tục quá trình tìm nghiệm của phương trình.

¹well-posed problems

2.2.1 PHƯƠNG PHÁP TÌM KIẾM GIA TĂNG

Ý tưởng của phương pháp tìm kiếm gia tăng rất đơn giản: Ta bắt đầu xét với đoạn con $[x_1, x_2] = [a, a + \Delta_x]$, nếu $f(x_1)$ và $f(x_2)$ trái dấu thì có ít nhất một nghiệm trong $[x_1, x_2]$. Nếu ngược lại, ta xét đoạn con tiếp theo bằng cách gán $x_1 := x_2; x_2 := x_1 + \Delta_x$. Quá trình tính toán sẽ kết thúc khi hoặc ta tìm được đoạn $[x_1, x_2]$ mà $f(x_1)$ và $f(x_2)$ trái dấu, hoặc đoạn con đang xét vượt ra ngoài đoạn $[a, b]$.

Có một vài vấn đề cần lưu ý khi thực hiện tính toán bằng phương pháp tìm kiếm gia tăng:

- Nếu có 2 nghiệm trên đoạn $[x_1, x_2]$ thì $f(x_1), f(x_2)$ sẽ cùng dấu nên đoạn này sẽ bị bỏ qua. Do đó, phương pháp có thể sẽ bỏ qua các đoạn mà chứa nhiều nghiệm quá gần nhau.
- Không xác định được nghiệm kép;
- Nếu hàm $f(x)$ không liên tục thì phương pháp có thể không chính xác.

rootsearch

Hàm `rootsearch` tìm các khoảng chứa 0-điểm của $f(x)$ trên khoảng (a, b) . Quá trình tìm bắt đầu từ a và mỗi bước tăng lên dx đến b . Với mỗi 0-điểm được xác định, hàm `rootsearch` trả về khoảng (x_1, x_2) chứa 0-điểm này. Nếu không có nghiệm nào được xác định thì thông báo $x_1 = x_2 = \text{NaN}$ được trả về².

```

function [x1,x2] = rootsearch(func,a,b,dx)
% Incremental search for a root of f(x) .
% USAGE: [x1,x2] = rootsearch(func,a,d,dx)
% INPUT:
% func = handle of function that returns f(x) .
% a,b =limits of search.
% dx = search increment.
% OUTPUT:
% x1,x2 = bounds on the smallest root in (a,b);
% set to NaN if no root was detected
x1 = a; f1 = feval(func,x1);
x2=a+dx;f2=feval(func,x2);
while f1*f2 > 0.0
    if x1 >= b

```

²Trong MATLAB, NaN là ký hiệu của “not a number”

```

x1 = NaN; x2 = NaN; return
end
x1=x2; f1=f2;
x2 = x1 + dx; f2 = feval(func,x2);
end

```

VÍ DỤ 2.2.1. Sử dụng số gia tìm kiếm $\Delta x = 0.2$, tìm khoảng chứa 0-điểm dương nhỏ nhất của $f(x) = x^3 - 10x^2 + 5$.

GIẢI. Tính giá trị hàm $f(x)$ tại các điểm bắt đầu từ $x = 0$ theo các khoảng $\Delta x = 0.2$ ta có kết quả

x	0.0	0.2	0.4	0.6	0.8
$f(x)$	5.000	4.608	3.464	1.616	-0.888

Từ việc đổi dấu của hàm $f(x)$, ta kết luận rằng nghiệm dương nhỏ nhất của phương trình $f(x) = 0$ nằm trong khoảng từ $x = 0.6$ đến $x = 0.8$.

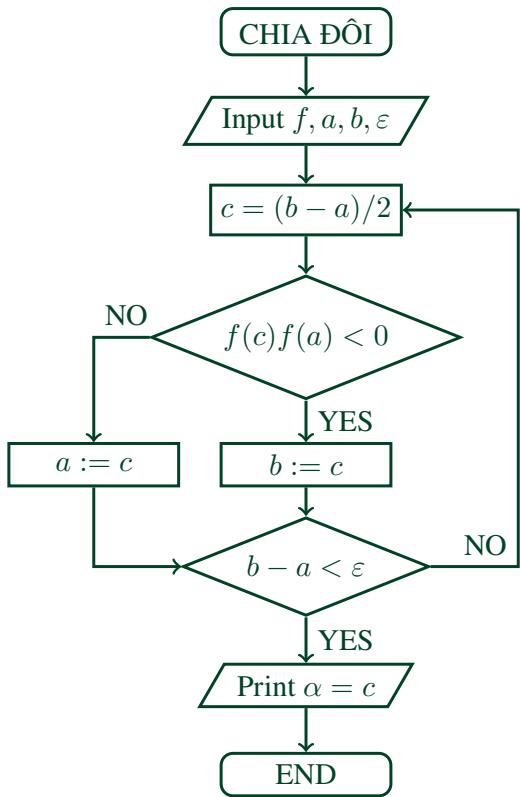
□

2.2.2 PHƯƠNG PHÁP CHIA ĐÔI

Sau khi đã tìm được khoảng (x_1, x_2) chứa nghiệm của phương trình $f(x) = 0$, có một số phương pháp có thể làm “mịn” khoảng này. *Phương pháp chia đôi* thực hiện điều này bằng cách liên tiếp giảm một nửa khoảng chứa nghiệm cho đến khi nó trở nên đủ nhỏ. Phương pháp chia đôi không phải là phương pháp nhanh nhất xấp xỉ nghiệm nhưng nó là đáng tin cậy nhất. Khi đã biết một khoảng có chứa nghiệm thì phương pháp chia đôi sẽ luôn tìm được khoảng con của khoảng này chứa nghiệm.

Phương pháp chia đôi sử dụng nguyên tắc tương tự như phương pháp tìm kiếm gia tăng. Nếu $f(x_1) \cdot f(x_2) < 0$ thì luôn có ít nhất một nghiệm trong khoảng (x_1, x_2) . Để giảm một nửa khoảng chứa nghiệm ta tính $f(x_3)$, với $x_3 = (x_1 + x_2)/2$ là trung điểm của khoảng. Nếu $f(x_2) \cdot f(x_3) < 0$ thì nghiệm phải nằm trong khoảng (x_3, x_2) và do đó ta thay thế x_1 ban đầu bởi x_3 . Ngược lại, nghiệm nằm trong khoảng (x_1, x_3) nên ta thay thế x_2 bởi x_3 . Trong cả hai trường hợp, khoảng mới (x_1, x_2) có chiều dài bằng một nửa khoảng gốc. Quá trình chia đôi được lặp lại cho đến khi khoảng chứa nghiệm được rút gọn có độ dài không lớn hơn độ dài cho phép ε , tức là

$$|x_2 - x_1| \leq \varepsilon.$$



**Hình 2.1: Sơ đồ khối
của phương pháp
chia đôi**

Phương pháp chia đôi được mô tả bằng sơ đồ khối cho trong Hình 2.1.

Ta có thể dễ dàng tìm được số bước lặp cần thiết để phương pháp chia đôi đạt độ dài cho phép ε . Độ dài khoảng gốc Δx trở thành $\Delta x/2$ sau lần chia đôi thứ nhất, thành $\Delta x/2^2$ sau lần chia đôi thứ hai và sau n lần chia đôi thì độ dài khoảng chứa nghiệm là $\Delta x/2^n$. Đặt $\Delta x/2^n = \varepsilon$ và giải theo n ta có

$$n = \frac{\ln(|\Delta x|/\varepsilon)}{\ln 2}. \quad (2.1)$$

bisect.

Hàm sau đây sử dụng phương pháp chia đôi để tìm nghiệm của phương trình $f(x) = 0$ trên khoảng (x_1, x_2) . Với độ dài cho phép tol , số lần chia đôi n được tính toán theo công thức (2.1). Tham số `filter` kiểm soát lọc các điểm suy biến. Bằng cách đặt `filter = 1`, chúng ta kiểm tra định kỳ xem giá trị của $f(x)$ có giảm sau mỗi nửa khoảng hay không. Nếu không, `root` có thể không phải là nghiệm mà là điểm suy biến. Khi đó `root = NaN` được trả về. Vì trường hợp này ít khi gặp nên ta mặc định là `filter = 0`.

```

function root = bisect(func,x1,x2,filter,tol)
% Finds a bracketed zero of f(x) by bisection.
% USAGE: root = bisect(func,x1,x2,filter,tol)
% INPUT:

```

```
% func = handle of function that returns f(x).
% x1,x2 = limits on interval containing the root.
% filter=singularity filter:0=off (default),1=on.
% tol = error tolerance (default is 1.0e4*eps).
% OUTPUT:
% root = zero of f(x), or NaN if singularity suspected.

if nargin < 5; tol = 1.0e4*eps; end
if nargin < 4; filter = 0; end
f1 = feval(func,x1);
if f1 == 0.0; root = x1; return; end
f2 = feval(func,x2);
if f2 == 0.0; root = x2; return; end
if f1*f2 > 0;
    error('Root is not bracketed in (x1,x2)')
end
n = ceil(log(abs(x2 - x1)/tol)/log(2.0));
for i=1:n
    x3 = 0.5*(x1 + x2);
    f3 = feval(func,x3);
    if(filter == 1) & (abs(f3) > abs(f1))...
        & (abs(f3) > abs(f2))
        root = NaN; return
    end
    if f3 == 0.0
        root = x3; return
    end
    if f2*f3 < 0.0
        x1=x3; f1=f3;
    else
        x2=x3; f2=f3;
    end
end
root=(x1 + x2)/2;
```

Một số ví dụ được trình bày sau đây sẽ minh họa cho phương pháp chia đôi. Các bước tính toán của phương pháp sẽ được minh họa chi tiết trong Ví dụ 2.2.2 và việc sử dụng chương trình MATLAB để giải số sẽ được minh họa trong Ví dụ 2.2.3.

VÍ DỤ 2.2.2. Sử dụng phương pháp chia đôi, hãy xấp xỉ nghiệm của phương trình $f(x) = x^3 - x - 1 = 0$ trên đoạn $[1, 2]$ với sai số không vượt quá 0.2. Nếu muốn xấp xỉ nghiệm với sai số tuyệt đối không vượt quá 0.001 thì ta cần thực hiện bao nhiêu lần lặp.

GIẢI. Do $f(1)f(2) = (-1)(5) = -5 < 0$ nên phương trình $x^3 - x - 1 = 0$ tồn tại nghiệm trên đoạn này. Áp dụng phương pháp chia đôi với $x_1 = 1, x_2 = 2, \varepsilon = 0, 2$ ta có:

Vòng lặp 1:

1. $x_3 = \frac{x_1+x_2}{2} = \frac{1+2}{2} = 1.5.$
2. $f(x_3)f(x_1) = f(1.5)f(1) < 0$ nên $x_2 := 1.5.$
3. $|x_2 - x_1| = |1.5 - 1| = 0.5 > \varepsilon = 0.2$ nên chuyển sang vòng lặp sau.

Vòng lặp 2:

1. $x_3 = \frac{x_1+x_2}{2} = \frac{1+1.5}{2} = 1.25.$
2. $f(x_3)f(x_1) = f(1.25)f(1) > 0$ nên $x_1 := 1.25.$
3. $|x_2 - x_1| = |1.5 - 1.25| = 0.25 > \varepsilon = 0.2$ nên chuyển sang vòng lặp sau.

Vòng lặp 3:

1. $x_3 = \frac{x_1+x_2}{2} = \frac{1.25+1.5}{2} = 1.375.$
2. $f(x_3)f(x_1) = f(1.375)f(1.25) < 0$ nên $x_2 := 1.375.$
3. $|x_2 - x_1| = |1.375 - 1.25| = 0.125 < \varepsilon = 0.2$ nên kết thúc lặp.

Nghiệm xấp xỉ của phương trình là $\alpha := x_3 = 0.1375$ với sai số không vượt quá 0, 2.

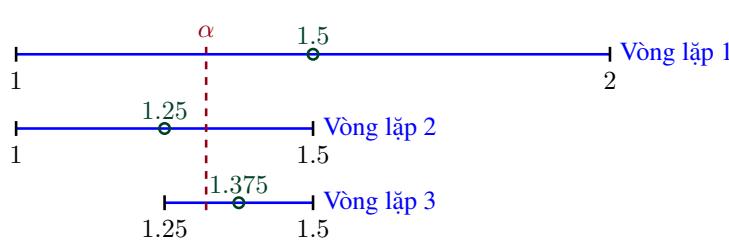
Gọi n là số lần lặp nhỏ nhất để sai số tuyệt đối không quá 0.001. Khi đó, n thoả mãn

$$\frac{1}{2^n} \leq 0.001 \Rightarrow n \geq 10.$$

Vậy ta cần thực hiện ít nhất 10 lần lặp để được nghiệm xấp xỉ với sai số tuyệt đối không quá 0.001.

Quá trình lặp để xấp xỉ nghiệm của phương trình được minh họa trong Hình 2.2.

□



Hình 2.2: Minh họa quá trình xấp xỉ nghiệm cho phương trình $x^3 - x - 1 = 0$ bằng phương pháp chia đôi.

Ví dụ 2.2.2 đã minh họa chi tiết ba bước tính toán của phương pháp chia đôi, sau đây ta sẽ trình bày một ví dụ áp dụng các hàm MATLAB đã biết để tìm 0-điểm của hàm suy biến.

VÍ DỤ 2.2.3. Tìm tất cả 0-điểm của $f(x) = x - \tan x$ trên khoảng $(0, 20)$ bằng phương pháp chia đôi. Sử dụng các hàm `rootsearch` và `bisect`.

GIẢI. Chú ý rằng $\tan x$ suy biến và đổi dấu tại các điểm $x = \pi/2, 3\pi/2, \dots$. Để cho hàm `bisect` không bỏ sót các nghiệm, chúng ta đặt `filter = 1`. Các nghiệm có thể gần các điểm kỳ dị nên để giảm việc bỏ sót các nghiệm, chúng ta sử dụng Δx nhỏ cho mỗi bước tìm kiếm. Ta thực hiện chương trình sau

```
>> % Example 2.2.3 (root finding with bisection)
a=0.0;b=20.0;dx=0.01;
nroots = 0;
while 1
    [x1,x2] = rootsearch(@fex2_2_3,a,b,dx);
    if isnan(x1)
        break
    else
        a=x2;
        x = bisect(@fex2_2_3,x1,x2,1);
        if ~isnan(x)
            nroots = nroots + 1;
            root(nroots) = x;
        end
    end
end
root
```

Nhắc lại rằng trong MATLAB, ký hiệu `@` được đặt trước tên hàm gọi một hàm được nhập vào bằng tay. Tham số đầu vào `@fex2_2_3` trong `rootsearch` là gọi hàm `fex2_2_3` được nhập vào như chương trình sau

```

function y=fex2_2_3(x)
% Function used in Example2.2.3
y=x-tan(x);

```

Chương trình cho kết quả là

root =

0	4.4934	7.7253	10.9041	14.0662	17.2208
---	--------	--------	---------	---------	---------

□

Từ các ví dụ trên ta thấy rằng phương pháp phân đôi có ưu điểm là đơn giản và có thể áp dụng tìm nghiệm xấp xỉ cho hầu hết các phương trình thực. Tuy nhiên, nhược điểm của phương pháp này là không sử dụng các đặc điểm của hàm số $f(x)$ nên hiệu quả không cao.

2.2.3 PHƯƠNG PHÁP DÂY CUNG

Ý tưởng của phương pháp *dây cung*³ là: Do nghiệm của phương trình $f(x) = 0$ là giao của cung AB với trực hoành nên ý tưởng của phương pháp dây cung là thay cung AB bởi đoạn thẳng (dây cung) đi qua A, B . Khi đó nghiệm của phương trình được xấp xỉ bởi giao của đoạn thẳng AB với trực hoành. Thực hiện quá trình này liên tiếp ta sẽ được dãy nghiệm xấp xỉ $\{x_n\}$.

Trong mục này và trong các mục tiếp theo của chương này, ta luôn giả thiết điều kiện sau được thoả mãn:

(g1) Phương trình $f(x) = 0$ có nghiệm duy nhất $\alpha \in [a, b]$.

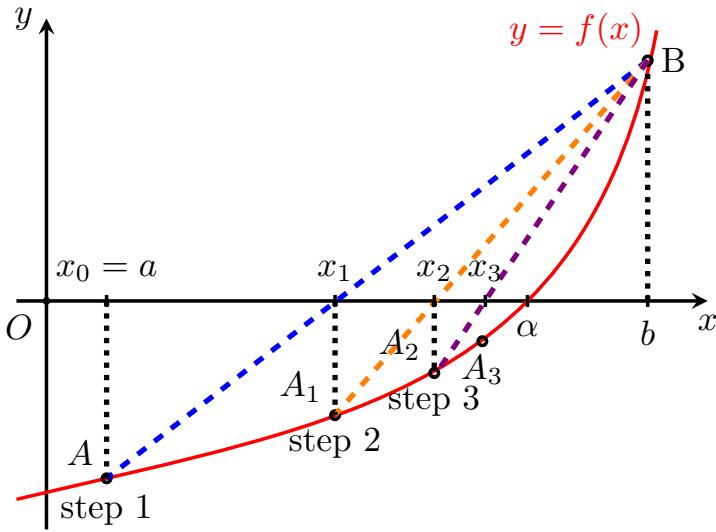
(g2) $f'(x), f''(x)$ không đổi dấu trên đoạn $[a, b]$.

Không mất tổng quát, ta có thể giả sử $f''(x) > 0$ trên đoạn $[a, b]$ vì nếu ngược lại thì ta xét phương trình $g(x) = 0$ với $g(x) = -f(x)$. Để xây dựng dây xấp xỉ, ta xét trường hợp $f'(x) > 0$ và thực hiện các bước:

Bước 1 Xác định x_1 . Lập phương trình đường thẳng qua A, B

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

³còn được gọi là phương pháp nội suy tuyến tính



Hình 2.3: Minh họa phương pháp dây cung khi $f'(x) > 0$

Cho $y = 0$ ta có hoành độ giao điểm x_1 của đường thẳng qua A, B với trục hoành là

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

Bước lặp 2 Xác định x_2 . Gọi $A_1 = (x_1, f(x_1))$. Phương trình đường thẳng qua A_1, B được xác định bởi

$$\frac{y - f(x_1)}{f(b) - f(x_1)} = \frac{x - x_1}{b - x_1}.$$

Cho $y = 0$ ta có hoành độ giao điểm x_2 của đường thẳng qua A_1, B với trục hoành là

$$x_2 = x_1 - \frac{f(x_1)}{f(b) - f(x_1)}(b - x_1).$$

⋮

Bước lặp n Xác định x_n . Giả sử sau $n - 1$ bước ta có nghiệm xấp xỉ x_{n-1} . Đặt $A_{n-1} = (x_{n-1}, f(x_{n-1}))$. Lập phương trình đường thẳng qua A_{n-1}, B

$$\frac{y - f(x_{n-1})}{f(b) - f(x_{n-1})} = \frac{x - x_{n-1}}{b - x_{n-1}}.$$

Cho $y = 0$ ta có hoành độ giao điểm x_n của đường thẳng qua A_{n-1}, B với trục hoành là

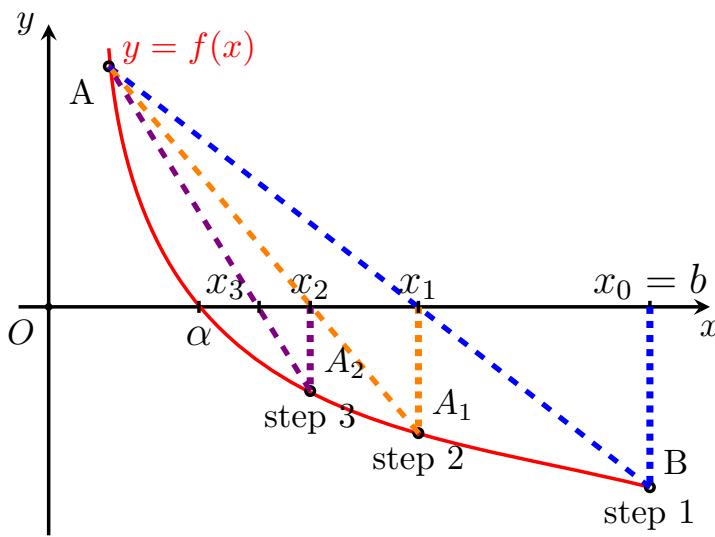
$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f(b) - f(x_{n-1})}(b - x_{n-1}). \quad (2.2)$$

Phương pháp tính x_n theo công thức (2.2) được gọi là *phương pháp dây cung*. Ta thấy rằng nếu tại **Bước lặp 2** thay vì lập phương trình đường thẳng qua A_1, B mà ta lại lập đường thẳng qua A_1, A thì hoành độ giao điểm x_2 sẽ không gần với nghiệm đúng α . Trong trường hợp đang xét, dây các điểm A, A_1, A_2, \dots luôn nằm bên dưới

trục hoành, điểm B nằm bên trên trục hoành nên hoành độ giao điểm x_n của các đoạn thẳng này sẽ hội tụ về nghiệm đúng α . Như vậy, để phương pháp dây cung hội tụ thì việc chọn điểm cố định B là rất quan trọng. Ta thấy trong trường hợp đang xét thì $f(b)f''(b) > 0$ nên ta chọn điểm B là điểm cố định.

Tương tự như vậy, công thức lặp dây cung trong trường hợp $f'(x) < 0$ trên đoạn $[a, b]$ được minh họa như trong Hình 2.4 được cho trong công thức sau

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f(x_{n-1}) - f(a)}(x_{n-1} - a), x_0 = b. \quad (2.3)$$



Hình 2.4: Minh họa phương pháp dây cung khi $f'(x) < 0$

Phương pháp lặp dây cung được minh họa bằng sơ đồ khôi trong Hình 2.5.

Định lý sau đây cho ta đánh giá về sai số và tốc độ hội tụ của phương pháp dây cung⁴.

ĐỊNH LÝ 2.2.1. *Giả sử phương trình $f(x) = 0$ có duy nhất nghiệm $\alpha \in [a, b]$. Nếu $f'(x), f''(x)$ không đổi dấu trên đoạn $[a, b]$ thì phương pháp lặp dây cung xác định bởi công thức (2.2) hoặc công thức (2.3) hội tụ. Hơn nữa, nếu $m \leq |f'(x)| \leq M, \forall x \in [a, b]$ thì ta có các công thức đánh giá sai số sau*

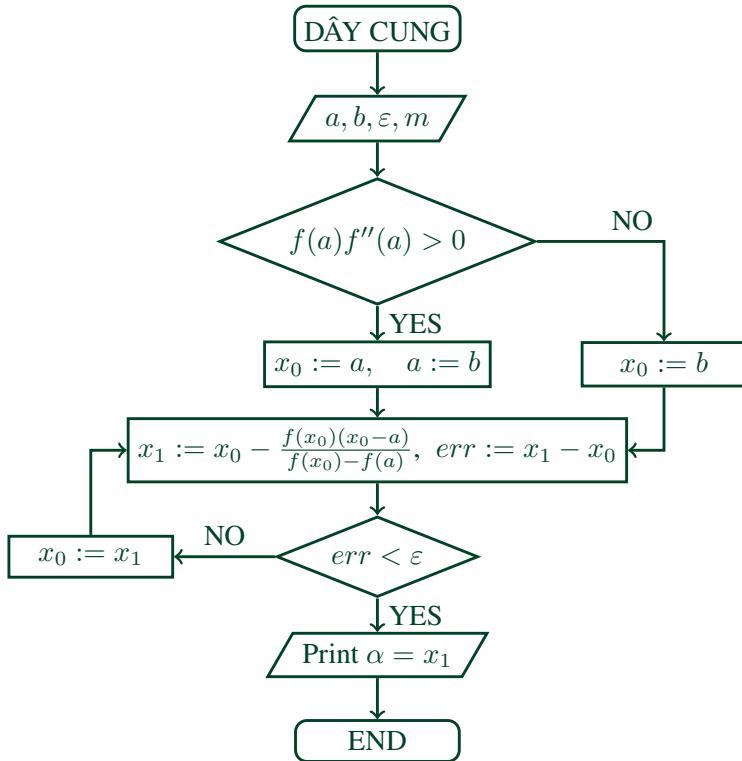
$$|x_n - \alpha| \leq \frac{|f(x_n)|}{m}, \quad (2.4)$$

$$|x_n - \alpha| \leq \frac{|M - n|}{m} |x_n - x_{n-1}|. \quad (2.5)$$

VÍ DỤ 2.2.4. Sử dụng phương pháp dây cung để xấp xỉ nghiệm của phương trình

$$f(x) = x^3 - x - 1 = 0.$$

⁴định lý được chứng minh chi tiết trong [2]



**Hình 2.5: Sơ đồ khôi
của phương pháp
lắp dây cung**

GIẢI. Ta đã biết phương trình có nghiệm trên $[1, 2]$. Việc tiếp theo là ta kiểm tra các điều kiện của Định lý 2.2.1. Ta có

$$2 \leq f'(x) = 3x^2 - 1 \leq 11, f''(x) = 6x \geq 6 > 0, \forall x \in [1, 2]$$

Do $f(2)f''(2) = 60 > 0$ nên chọn $x_0 = 1$ và thực hiện tính xấp xỉ theo công thức lắp (2.2) ta có

$$x_n = x_{n-1} - \frac{x_{n-1}^3 - x_{n-1} - 1}{5 - (x_{n-1}^3 - x_{n-1} - 1)}(2 - x_{n-1}),$$

với cận trên sai số được xác định

$$\Delta_n = \frac{11 - 2}{2} |x_n - x_{n-1}|$$

Kết quả tính toán được cho trong Bảng 2.1.

□

2.3 CÁC PHƯƠNG PHÁP MỞ

Với các phương pháp khoảng đã giới thiệu ở mục trước, nghiệm của phương trình được xác định trên một khoảng được xác định bằng một cận dưới và một cận trên. Áp

Bước lặp	Nghiệm xấp xỉ x_n	Sai số Δ
1	1.16666666666667	0.75
2	1.25311203319502	0.389004149377593
3	1.29343740191868	0.181464159256482
4	11.31128102148723	0.0802962880584793
5	1.31898850356646	0.034683669356528

Bảng 2.1: Bảng kết quả tìm nghiệm của phương trình $x^3 - x - 1 = 0$ trên đoạn $[1, 2]$ bằng phương pháp lặp đơn

dụng lặp đi lặp lại các phương pháp này luôn dẫn đến các khoảng ước lượng gần hơn với giá trị chính xác của nghiệm. Do vậy, các phương pháp khoảng đã trình bày được gọi là hội tụ.

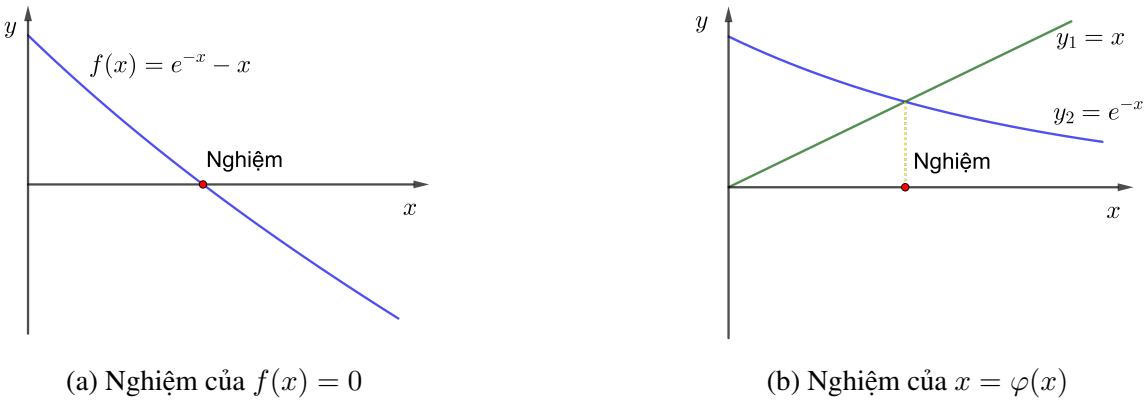
Trái lại, các phương pháp mở sẽ được trình bày trong mục này chỉ yêu cầu một hoặc hai giá trị bắt đầu mà không cần yêu cầu khoảng chứa nghiệm. Do đó, đôi khi chúng phân kỳ hoặc di chuyển ra xa giá trị chính xác của nghiệm khi tính toán. Tuy nhiên, khi mà các phương pháp mở hội tụ thì chúng thường thực hiện nhanh hơn nhiều so với các phương pháp khoảng. Chúng ta sẽ bắt đầu thảo luận về các phương pháp mở bằng một phương pháp điển hình là phương pháp lặp.

2.3.1 PHƯƠNG PHÁP LẶP

Một trong những phương pháp cơ bản trong tính toán khoa học là *phương pháp lặp*. Lý do phương pháp được gọi như vậy vì quá trình tính toán được thực hiện bằng cách lặp liên tiếp các giá trị cho đến khi tìm được giá trị chấp nhận được. Trong mục này chúng ta sẽ mô tả phương pháp lặp đơn để tìm nghiệm xấp xỉ của phương trình thực một biến.

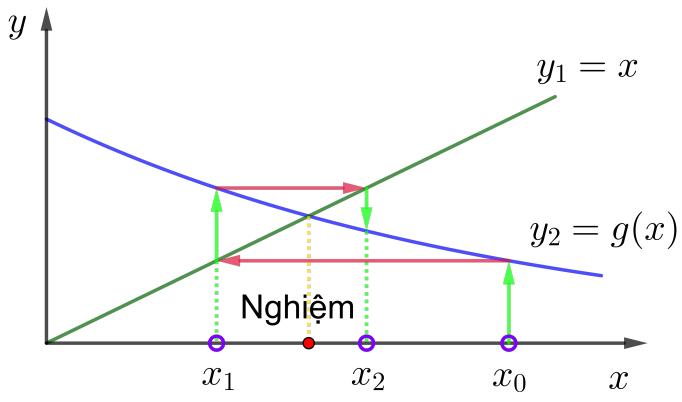
Giả sử α là nghiệm thực phương trình $f(x) = 0$. Nếu minh họa bằng đồ thị thì α chính là hoành độ giao điểm của đường đồ thị hàm $f(x)$ với trục x như Hình 2.6a. Ta biến đổi tương đương, đưa phương trình $f(x) = 0$ về dạng $x = \varphi(x)$. Khi đó, α chính là hoành độ giao điểm của hai đường $y = \varphi(x)$ và $y = x$ như Hình 2.6b. Chọn một giá trị ban đầu x_0 , ta tìm nghiệm xấp xỉ cho phương trình bằng cách thực hiện theo công thức truy hồi sau.

$$x_n = \varphi(x_{n-1}). \quad (2.6)$$



Hình 2.6: Đồ thị minh họa nghiệm của phương trình $f(x) = 0$ và $x = \varphi(x)$

Hàm φ được gọi là *hàm lặp*. Phương pháp lặp được minh họa như Hình 2.7.



Hình 2.7: Minh họa phương pháp lặp xấp xỉ nghiệm phương trình.

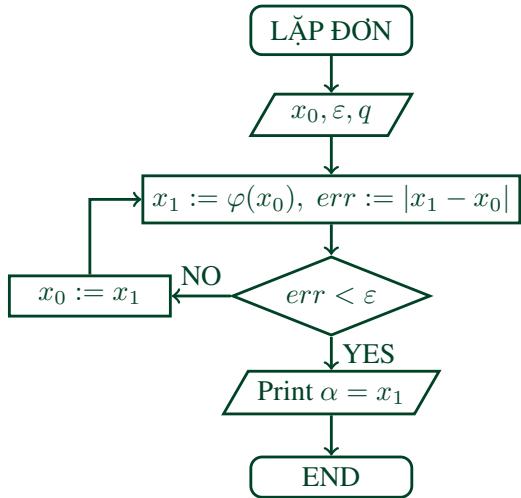
Quá trình tính toán được thực hiện lần lượt từ $x_1, x_2, \dots, x_{n-1}, x_n$ thông qua hàm φ và sẽ dừng lại khi sai lệch giữa hai lần lặp nhỏ hơn sai số cho phép ε cho trước. Sơ đồ khái của phương pháp lặp được minh họa bằng Hình 2.8.

Do đó, ta mong muốn tìm được dãy các xấp xỉ $\{x_n\}_1^\infty$ sao cho khi n càng lớn thì x_n càng gần nghiệm đúng α . Một dãy có tính chất như vậy được gọi là *hội tụ*.

ĐỊNH NGHĨA 2.3.1 (Tính hội tụ). Dãy $\{x_n\}$ được xác định bằng công thức (2.6) gọi là *hội tụ* nếu thoả mãn điều kiện

$$\lim_{n \rightarrow \infty} x_n = \alpha. \quad (2.7)$$

Ta thấy rằng việc hàm lặp φ phụ thuộc vào từng trường hợp cụ thể của hàm $f(x)$ và không phải hàm nào cũng cho dãy lặp hội tụ. Trong một vài trường hợp đơn giản,



Hình 2.8: Sơ đồ khôi của phương pháp lắp xấp xỉ nghiệm thực của phương trình

ta có thể đưa phương trình $f(x) = 0$ về một trong các dạng sau

$$x = x - \tau f(x) \quad (2.8)$$

$$x = x - \mu(x)f(x), \quad (2.9)$$

trong đó $\tau \neq 0$, $\mu(x)$ là hàm khả vi và bị chặn trên $[a, b]$. Định lý sau đây cho ta điều kiện để chọn một hàm lắp $\varphi(x)$ sao cho dãy $\{x_n\}$ xác định trong công thức (2.6) hội tụ⁵.

ĐỊNH LÝ 2.3.1. Xét phương pháp lắp $x_n = \varphi(x_{n-1})$, nếu ta có các điều kiện sau

(i) Phương trình $f(x) = 0$ tồn tại duy nhất nghiệm trong đoạn $[a, b]$;

(ii) $x_n = \varphi(x_{n-1}) \in (a, b)$, $n = 1, 2, \dots$;

(iii) $|\varphi'(x)| \leq q < 1$, $\forall x \in [a, b]$, với q là hằng số,

thì phương pháp lắp $x_n = \varphi(x_{n-1})$ hội tụ. Hơn nữa, sai số của phương pháp được đánh giá bởi các công thức sau

$$|\alpha - x_n| \leq \frac{q}{1-q} |x_n - x_{n-1}|. \quad (2.10)$$

$$|\alpha - x_n| \leq \frac{q^n}{1-q} |x_1 - x_0|. \quad (2.11)$$

Khi áp dụng phương pháp lắp đơn xấp xỉ nghiệm thực cho một phương trình, cần lưu ý một số tính chất sau.

⁵định lý được chứng minh chi tiết trong [2]

NHẬN XÉT 2.3.1. (i) Công thức đánh giá sai số (2.11) dùng làm ước lượng tiên nghiệm, tức là không cần tính toán ta cũng có thể ước lượng số lần lặp để tìm được nghiệm gần đúng x_n với sai số không quá một giá trị ε cho trước. Thật vậy, nếu muốn $|x_n - \alpha| \leq \varepsilon$ ta cần

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

nên suy ta

$$n \geq \frac{\ln \frac{(1-q)\varepsilon}{|x_1 - x_0|}}{\ln q}.$$

Công thức đánh giá sai số (2.10) rất thuận lợi cho quá trình tính toán vì nó cho ta ước lượng hậu nghiệm, tức là nếu muốn $|x_n - \alpha| \leq \varepsilon$ ta cần sai số giữa hai lần lặp liên tiếp thoả mãn

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

- (ii) Phương pháp lặp đơn có thể tự sửa sai. Tức là nếu lần lặp thứ k mắc sai số thì ta có thể coi như xấp xỉ ban đầu mới.
- (iii) Nếu hệ số q gần 1 thì phép lặp hội tụ chậm.

Sau đây ta sẽ thực hiện một số ví dụ minh họa cho phương pháp lặp đơn. Đầu tiên, ta sẽ tìm xấp xỉ nghiệm cho phương trình đã cho trong Ví dụ 2.2.2 bằng phương pháp lặp đơn để so sánh kết quả của phương pháp lặp đơn với phương pháp chia đôi.

VÍ DỤ 2.3.1. Sử dụng phương pháp lặp đơn để xấp xỉ nghiệm của phương trình

$$f(x) = x^3 - x - 1 = 0.$$

GIẢI. Ta đã biết phương trình có khoảng phân ly nghiệm là $[1, 2]$. Việc tiếp theo là ta cần tìm hàm lặp $\varphi(x)$ thoả mãn Định lý 2.3.1 để tìm được dãy lặp hội tụ. Ta thấy rằng nếu biến đổi phương trình về dạng $x = x^3 - 1$ thì hàm lặp $\varphi(x) = x^3 - 1$. Tuy nhiên, vì $\varphi'(x) = 3x^2 \geq 3 \forall x \in [1, 2]$ nên $\varphi(x)$ không thoả mãn mục (iii) của Định lý 2.3.1.

Ta biến đổi phương trình đã cho về dạng $x = \sqrt[3]{x+1}$, ta có hàm lặp $\varphi(x) = \sqrt[3]{x+1}$.

Do

$$\varphi'(x) = \frac{1}{3} \sqrt[3]{(x+1)^2} < \frac{1}{3} < 1, \forall x \in [1, 2]$$

nên $\varphi(x) = \sqrt[3]{x+1}$ thoả mãn Định lý 2.3.1. Lấy $x_0 = 1 \in [1, 2]$ và sử dụng hàm lặp φ ta có dãy lặp sau

$$x_n = \sqrt[3]{x_{n-1} + 1}, n = 1, 2, \dots$$

với sai số được đánh giá

$$|\alpha - x_n| \leq \frac{\left(\frac{1}{3}\right)^n}{1 - \frac{1}{3}} |x_1 - x_0| = \left(\frac{1}{3}\right)^{n-1}$$

hoặc ta đánh giá theo công thức

$$|\alpha - x_n| \leq \frac{\frac{1}{3}}{1 - \frac{1}{3}} |x_n - x_{n-1}| = \frac{1}{2} |x_n - x_{n-1}|.$$

Kết quả tính toán được cho trong Bảng 2.2.

Bước lặp	Nghiệm xấp xỉ c	Sai số Δ_c
1	1,259921050	0,130000
2	1,312293837	0,027000
3	1,322353819	0,005000
4	1,324068754	0,000960
5	1,324632625	0,000182

Bảng 2.2: Bảng kết quả tìm nghiệm của phương trình $x^3 - x - 1 = 0$ trên đoạn $[1, 2]$ bằng phương pháp lặp đơn

□

2.3.2 PHƯƠNG PHÁP NEWTON-RAPHSON

Phương pháp Newton-Raphson (còn được biết đến với tên gọi là phương pháp tiếp tuyến) là một phương pháp lặp tốt nhất thường được sử dụng để xấp xỉ nghiệm của phương trình vì lý do: đơn giản và hội tụ nhanh. Có hai cách để xây dựng công thức lặp cho phương pháp Newton-Raphson là sử dụng khai triển Taylor và sử dụng đồ thị. Trước tiên ta sẽ sử dụng khai triển Taylor để xây dựng công thức xấp xỉ, sau đó, cách tiếp cận bằng đồ thị để xây dựng công thức sẽ được giới thiệu trong mục Nhận xét 2.3.2.

Giả sử phương trình $f(x) = 0$ có nghiệm α trên $[a, b]$ và $f'(x) \neq 0, \forall x \in (a, b)$. Khai triển Taylor hàm $f(x)$ tại điểm x_0 ta có

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + \frac{(x - x_0)^2}{2}f''(c), \quad (2.12)$$

với $x, c \in [a, b]$. Bỏ qua số hạng cuối cùng của (2.12) ta có xấp xỉ

$$f(x) \approx f(x_0) + (x - x_0)f'(x_0). \quad (2.13)$$

Như vậy, ta đã xấp xỉ hàm $f(x)$ bởi một hàm tuyến tính đơn giản $f(x_0) + (x - x_0)f'(x_0)$ nên nghiệm của phương trình $f(x) = 0$ sẽ được xấp xỉ bởi quá trình lặp bởi phương trình tuyến tính $f(x_0) + (x - x_0)f'(x_0)$, cụ thể như sau.

Bước lặp 1

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

Bước lặp 2

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

⋮

Bước lặp n

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}. \quad (2.14)$$

Công thức lặp (2.14) được gọi là công thức Newton-Raphson.

Sai số và tính hội tụ của phương pháp Newton-Raphson được xác định trong định lý sau đây⁶.

ĐỊNH LÝ 2.3.2. *Giả sử phương trình $f(x) = 0$ có nghiệm α trong khoảng phân ly nghiệm $[a, b]$, hàm số $f(x)$ có đạo hàm $f'(x), f''(x)$ liên tục và không đổi dấu trên (a, b) . Nếu ta chọn $x_0 \in [a, b]$ sao cho $f(x_0)f''(x_0) > 0$ thì x_n xác định bởi (2.14) sẽ hội tụ về α khi $n \rightarrow \infty$, tức là*

$$\lim_{n \rightarrow \infty} x_n = \alpha.$$

Hơn nữa, nếu $|f'(x)| \geq m, |f''(x)| \leq M, \forall x \in [a, b]$ thì ta có công thức đánh giá sai số sau đây

$$|x_n - \alpha| \leq \frac{|f(x_n)|}{m}, \quad (2.15)$$

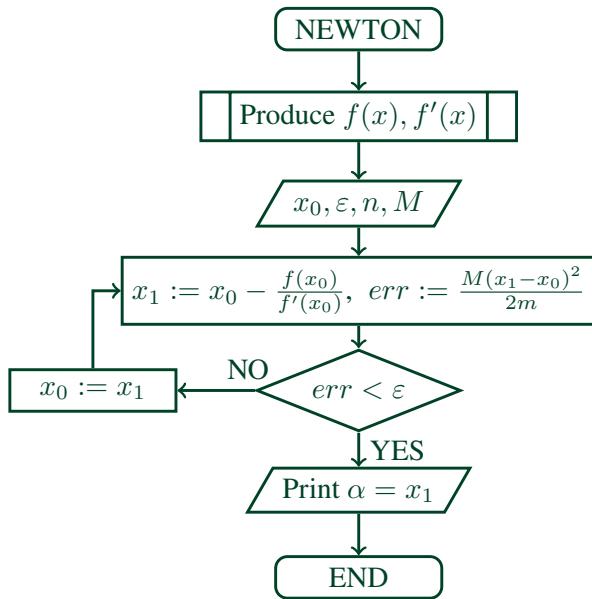
$$|x_n - \alpha| \leq \frac{M}{2m} |x_n - x_{n-1}|^2. \quad (2.16)$$

Từ định lý trên ta có $|x_n - \alpha| = \mathcal{O}(|x_n - x_{n-1}|^2)$, tức là x_n hội tụ về α có bậc 2. Khi n lớn ($|x_n - x_{n-1}|$ khá bé) thì x_n rất gần α . Phương pháp Newton được mô tả bởi sơ đồ khối trong Hình 2.9.

 [newtonRaphson](#).

Phiên bản “an toàn” sau đây của phương pháp Newton-Raphson giả định rằng nghiệm cần tìm nằm trong khoảng ban đầu (a, b) . Trung điểm của khoảng này được dùng để làm giá trị xấp xỉ đầu tiên. Các khoảng được cập nhật sau mỗi lần lặp. Nếu có một phép lặp Newton-Raphson không nằm trong khoảng, nó sẽ bị bỏ qua và được thay thế

⁶định lý được chứng minh chi tiết trong [2]



Hình 2.9: Sơ đồ khôi của phương pháp lặp Newton-Raphson xấp xỉ nghiệm thực của phương trình

bằng phương pháp chia đôi. Vì newtonRaphson sử dụng hàm $f(x)$ cũng như đạo hàm của nó, nên các hàm này (ký hiệu là func và dfunc) phải được cung cấp từ người dùng.

```

function root = newtonRaphson(func,dfunc,a,b,tol)
% Newton-Raphson method combined with bisection for
% finding a root of f(x)=0.
% USAGE: root = newtonRaphson(func,dfunc,a,b,tol)
% INPUT:
% func = handle of function that returns f(x).
% dfunc = handle of function that returns f'(x).
% a,b = brackets (limits) of the root.
% tol = error tolerance (default is 1.0e6*eps).
% OUTPUT:
% root = zero of f(x) (root = NaN if no convergence).
if nargin < 5; tol = 1.0e6*eps; end
fa = feval(func,a); fb = feval(func,b);
if fa == 0; root = a; return; end
if fb == 0; root = b; return; end
if fa*fb > 0.0
    error('Root is not bracketed in (a,b)')
end
x=(a+b)/2.0;
for i=1:30

```

```

fx = feval(func,x);
if abs(fx) < tol; root = x; return; end
% Tighten brackets on the root
if fa*fx< 0.0;b=x;
else;a=x;
end
% Try Newton--Raphson step
dfx = feval(dfunc,x);
if abs(dfx)==0;dx=b-a;
else; dx = -fx=dfx;
end
x=x+dx;
% If x not in bracket, use bisection
if (b-x)*(x-a)<0.0
dx = (b - a)/2.0;
x=a+dx;
end
% Check for convergence
if abs(dx) < tol*max(b,1.0)
root = x; return
end
end
root = NaN

```

Ta cũng có thể xây dựng công thức lặp Newton-Raphson trong (2.14) bằng hình học trong nhận xét sau.

NHẬN XÉT 2.3.2. Ý tưởng của cách tiếp cận bằng hình học là: Tại mỗi điểm lặp, ta xấp xỉ đường cong $f(x)$ bởi đường thẳng $y = a_0 + a_1x$ tiếp tuyến với $f(x)$ tại điểm này⁷. Khi đó, nghiệm của phương trình $f(x) = 0$ sẽ được xấp xỉ bởi nghiệm phương trình $a_0 + a_1x = 0$. Cụ thể, với giá trị ban đầu x_0 , phương pháp được thực hiện như sau:

Bước lặp 1: Ta viết phương trình tiếp tuyến với đồ thị hàm $f(x)$ tại điểm $(x_0, f(x_0))$ dưới dạng

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

⁷chính vì vậy nên phương pháp Newton-Raphson còn được gọi là phương pháp tiếp tuyến

Tiếp tuyến này giao với trục hoành tại điểm x_1 thoả mãn điều kiện

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

Bước lặp 2: Ta viết phương trình tiếp tuyến với đồ thị hàm $f(x)$ tại điểm $(x_1, f(x_1))$ tương tự như Bước 1. Tiếp tuyến này giao với trục hoành tại điểm x_2 thoả mãn điều kiện

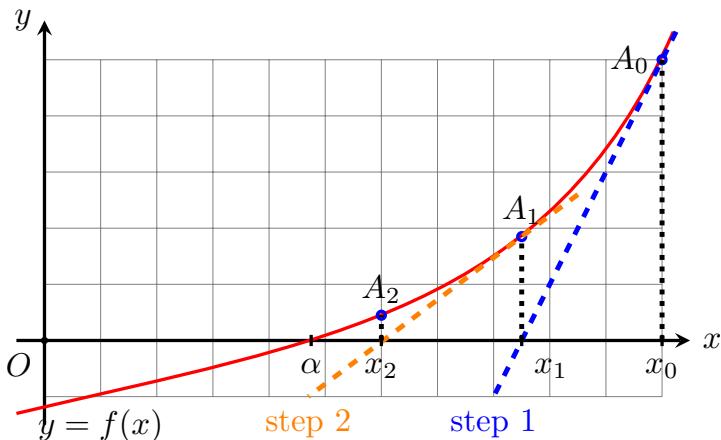
$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}.$$

⋮

Bước lặp n : Ta xây dựng được công thức xấp xỉ nghiệm sau

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

Các bước lặp trên được minh họa bằng Hình 2.10



Hình 2.10: Minh họa phương pháp lặp Newton

VÍ DỤ 2.3.2. Một nghiệm của phương trình $f(x) = x^3 - 10x^2 + 5 = 0$ nằm gần $x = 0.7$.

Sử dụng phương pháp Newton-Raphson để tìm nghiệm này.

GIẢI. Đạo hàm của hàm số là $f'(x) = 3x^2 - 20x$ nên công thức lặp Newton-Raphson là

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})} = x_{n-1} - \frac{x_{n-1}^3 - 10x_{n-1}^2 + 5}{3x_{n-1}^2 - 20x_{n-1}} = \frac{2x_{n-1}^3 - 10x_{n-1}^2 - 5}{x_{n-1}[3x_{n-1} - 20]}$$

Lấy $x_0 = 0.7$ và thực hiện lặp hai lần ta có

$$x_1 = \frac{2(0.7)^3 - 10(0.7)^2 - 5}{0.7[3(0.7) - 20]} = 0.73536$$

$$x_2 = \frac{2(0.73536)^3 - 10(0.73536)^2 - 5}{0.73536[3(0.73536) - 20]} = 0.73460$$

□

VÍ DỤ 2.3.3. Sử dụng phương pháp Newton-Raphson để xấp xỉ nghiệm của phương trình

$$f(x) = x^3 - x - 1 = 0.$$

GIẢI. Ta đã biết phương trình có khoảng phân ly nghiệm là $[1, 2]$. Việc tiếp theo là ta kiểm tra các điều kiện của Định lý 2.3.2. Ta có

$$f'(x) = 3x^2 - 1 \geq 2 > 0, f''(x) = 6x \geq 6 > 0, \forall x \in [1, 2].$$

Do $f(x) = 5 > 0$ nên chọn $x_0 = 2$ và thực hiện tính xấp xỉ theo công thức lặp

$$x_n = x_{n-1} - \frac{x_{n-1}^3 - x_{n-1} - 1}{3x_{n-1} - 1},$$

với cận trên sai số được xác định

$$\Delta_n = \frac{12}{2 \times 2} |x_n - x_{n-1}|^2.$$

Kết quả tính toán được cho trong Bảng 2.3.

Bước lặp	Nghiệm xấp xỉ x_n	Sai số Δ
1	1.545454545455	0.619834710743802000
2	1.35961491591518	0.103609103721981000
3	1.32580134500585	0.003430072732922690
4	1.32471904941713	3.51409122408669E-06
5	1.32471795724586	3.57851423362127E-12

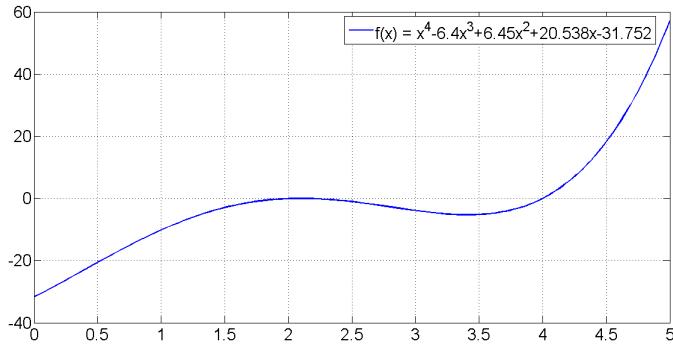
Bảng 2.3: Bảng kết quả tìm nghiệm của phương trình $x^3 - x - 1 = 0$ trên đoạn $[1, 2]$ bằng phương pháp Newton

□

VÍ DỤ 2.3.4. Tìm 0-điểm dương nhỏ nhất của

$$f(x) = x^4 - 6.4x^3 + 6.45x^2 - 20.538x - 31.752$$

GIẢI.



Quan sát đồ thị của hàm số, ta nghi ngờ rằng nghiệm dương nhỏ nhất là nghiệm kép nằm gần $x = 2$. Các phương pháp khoảng như phương pháp tìm kiếm gia tăng, phương pháp chia đôi không thực hiện được vì hàm số không đổi dấu khi qua nghiệm này. Phương pháp Newton-Raphson cũng gặp tình trạng tương tự. Tuy nhiên, chúng ta có thể điều chỉnh hàm newtonRaphson để có thể áp dụng được cho trường hợp này. Hàm sau đây tính toán và in ra số lước lặp và nghiệm xấp xỉ của bài toán.

```

function [root,numIter] =
newton_simple(func,dfunc,x,tol)
% Simple version of Newton-Raphson method used in
Example 2.3.4.
if nargin < 5; tol = 1.0e6*eps; end
for i=1:30
    dx = -feval(func,x)/feval(dfunc,x);
    x=x+dx;
    if abs(dx) < tol
        root = x; numIter = i; return
    end
end
root = NaN

```

Với hàm số $f(x)$ và đạo hàm của nó được tính bởi các M-file

```

function y=fex2_3_4 (x)
% Function used in Example2.2.4
y=(x.^4)-6.4*(x.^3)+6.45*(x.^2)+20.538*(x)-31.752;

function y=dfex2_3_4 (x)
% Function used in Example 4.7.
y = 4.0*x^3 - 19.2*x^2 + 12.9*x + 20.538;

```

Kết quả là

```
>> [root, numIter] =
    newton_simple(@fex2_3_4, @dfex2_3_4, 2.0)
root =
2.1000
numIter =
27
```

Ta thấy rằng gần nghiệm bội thì sự hội tụ của phương pháp Newton-Raphson là tuyến tính, không phải bậc hai. Điều này giải thích số phép lặp lớn. Tốc độ hội tụ của phương pháp Newton-Raphson cho các nghiệm bội có thể được cải thiện bằng cách thay thế công thức (2.14) bởi

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

với m là bội số của nghiệm. Sau khi thay thực hiện thay đổi này trong chương trình, ta thu được kết quả sau 5 bước lặp.

□

2.4 0-ĐIỂM CỦA ĐA THỨC

2.4.1 GIỚI THIỆU

Một đa thức bậc n có dạng

$$P_n(x) = a_1x^n + a_2x^{n-1} + \cdots + a_nx + a_{n+1}, \quad (2.17)$$

với các hệ số a_i có thể là số thực hoặc số phức. Quá trình tính toán trong tài liệu này thực hiện với các đa thức với hệ số thực nhưng các thuật toán này cũng có thể làm việc với các đa thức có hệ số phức.

Phương trình đa thức $P_n(x) = 0$ có chính xác n nghiệm thực hoặc phức. Các nghiệm phức luôn thu được dưới dạng cặp số phức liên hợp $(x_r + ix_i, x_r - ix_i)$ với x_r, x_i tương ứng là phần thực và phần ảo. Với các hệ số thực, các nghiệm thực có thể được đánh giá theo *quy tắc Descartes*:

- Số nghiệm thực dương bằng với số lần đổi dấu của các hệ số của biểu thức $P_n(x)$ hoặc nhỏ hơn một số chẵn.

- Số nghiệm thực âm bằng với số lần đổi dấu của các hệ số của biểu thức $P_n(-x)$ hoặc nhỏ hơn một số chẵn.

Chẳng hạn, xét đa thức $P_3(x) = x^3 - 2x^2 - 8x + 27$. Vì các hệ số có hai lần đổi dấu nên $P_3(x) = 0$ có hai hoặc không có nghiệm dương. $P_3(-x) = -x^3 - 2x^2 + 8x + 27$ có hệ số đổi dấu một lần nên $P_3(x) = 0$ có một nghiệm âm.

0-điểm thực của một đa thức hệ số thực luôn có có thể được ước lượng bằng một trong các phương pháp đã giới thiệu trong các mục trước. Nhưng nếu các nghiệm phức cần được tính toán thì tốt nhất là sử dụng một phương pháp chuyên về đa thức. Mục này sẽ trình bày một phương pháp đáng tin cậy và đơn giản để thực hiện của Laguerre. Trước khi tiến hành phương pháp Laguerre, chúng ta phải phát triển hai công cụ số cần thiết trong bất kỳ phương pháp tính 0-điểm đa thức. Công cụ đầu tiên là một thuật toán hiệu quả để tính giá trị một đa thức và các đạo hàm của nó. Thuật toán thứ hai chúng ta cần là *giảm bậc* của một đa thức, tức là dùng để chia $P_n(x)$ cho $x - r$, trong đó r là nghiệm của $P_n(x) = 0$.

2.4.2 TÍNH GIÁ TRỊ ĐA THỨC

Ta tính toán giá trị đa thức trong (2.17) từ trái sang phải bằng thuật toán sau (giả sử các hệ số của đa thức được lưu trữ trong mảng a)

```
p=0.0
for i=1:n+1
    p=p+a(i)*x^(n-i+1)
end
```

Vì x^k được tính như là $x \times x \times \cdots \times x$ ($k - 1$ phép nhân), ta suy ra số lượng phép nhân trong thuật toán này là

$$1 + 2 + \cdots + n - 1 = \frac{1}{2}n(n - 1).$$

Nếu n lớn, số lượng phép nhân có thể giảm nếu ta tính đa thức từ phải sang trái. Ví dụ, với

$$P_4(x) = a_1x^4 + a_2x^3 + a_3x^2 + a_4x + a_5$$

có thể viết lại dưới dạng

$$P_4(x) = a_5 + x\{a_4 + x[a_3 + x(a_2 + a_1x)]\}.$$

Ta thấy dãy các tính toán hiệu quả cho việc tính giá trị đa thức là

$$\begin{aligned}P_0(x) &= a_1 \\P_1(x) &= a_2 + xP_0(x) \\P_2(x) &= a_3 + xP_1(x) \\P_3(x) &= a_4 + xP_2(x) \\P_4(x) &= a_5 + xP_3(x).\end{aligned}$$

Với một đa thức bậc n , quá trình tính toán có thể tóm tắt dạng

$$\begin{aligned}P_0(x) &= a_1 \\P_i(x) &= a_{i+1} + xP_{i-1}(x), \quad i = 1, 2, \dots, n\end{aligned}\tag{2.18}$$

dẫn đến ta có thuật toán

```
p = a(1);  
for i=1:n  
    p=p*x+a(i+1)  
end
```

Thuật toán này chỉ tồn n phép nhân nên khi $n > 3$ thì nó hiệu quả hơn thuật toán trước rất nhiều. Tuy nhiên, vấn đề tiết kiệm số phép tính không phải lý do chính để thuật toán này được chọn. Ta biết rằng kết quả của mỗi phép nhân sẽ được làm tròn, do đó, những thuật toán có số phép tính ít nhất thì dẫn đến sẽ hạn chế sai số làm tròn tích lũy tốt nhất.

Một số thuật toán tìm nghiệm, bao gồm cả thuật toán Laguerre, cũng yêu cầu tính giá trị đạo hàm cấp một và đạo hàm cấp hai của $P_n(x)$. Từ (2.18), ta thu được các thuật toán tính đạo hàm

$$P'_0(x) = 0, \quad P'_i(x) = P_{i-1}(x) + xP'_{i-1}(x), \quad i = 1, 2, \dots, n.\tag{2.19a}$$

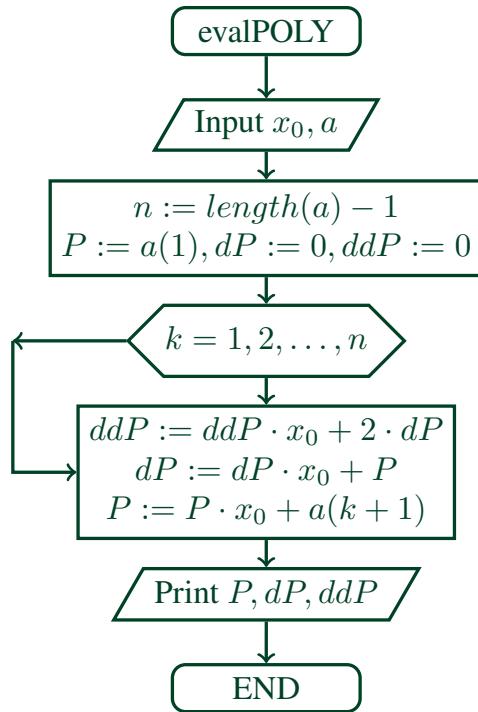
$$P''_0(x) = 0, \quad P''_i(x) = 2P'_{i-1}(x) + xP''_{i-1}(x), \quad i = 1, 2, \dots, n.\tag{2.19b}$$

Sơ đồ khôi của thuật toán tính giá trị của đa thức bằng công thức (2.18) và các đạo hàm của nó bằng công thức (2.19) được minh họa bằng Hình 2.11.

 evalpoly

Chương trình sau tính giá trị của đa thức và các đạo hàm của nó.

```
function [p, dp, ddp] = evalpoly(a, x)
```



Hình 2.11: Sơ đồ khôi tính giá trị của đa thức và giá trị đạo hàm bậc một bậc hai của đa thức

```

% Evaluates the polynomial
% p=a(1)*x^n+a(2)*x^(n-1)+...+a(n+1)
% and its first two derivatives dp and ddp.
% USAGE: [p,dp,ddp] = evalpoly(a,x)
n = length(a) - 1;
p=a(1);dp=0.0;ddp=0.0;
for i=1:n
    ddp = ddp*x + 2.0*dp;
    dp=dp*x+p;
    p=p*x+a(i+1);
end
  
```

2.4.3 GIẢM BẬC ĐA THỨC

Sau khi nghiệm r của $P_n(x) = 0$ được tính, ta mong muốn phân tích đa thức dưới dạng

$$P_n(x) = (x - r)P_{n-1}(x). \quad (2.20)$$

Quá trình này, được biết đến là quá trình *giảm cấp* hoặc *synthetic division*, cần xác định các hệ số của $P_{n-1}(x)$. Vì các 0-điểm còn lại của $P_n(x)$ cũng là các 0-điểm của $P_{n-1}(x)$ nên việc tìm nghiệm có thể áp dụng cho $P_{n-1}(x)$ thay vì áp dụng cho $P_n(x)$.

Do đó, giảm cấp sẽ làm quá trình tìm nghiệm dễ dàng hơn sau mỗi nghiệm được tìm thấy vì bậc của đa thức giảm sau khi tìm được một nghiệm. Hơn nữa, bằng cách khử các nghiệm đã được tìm thấy, việc phải tính lại các nghiệm này sẽ bị loại bỏ.

Nếu ta đặt

$$P_{n-1}(x) = b_1x^{n-1} + b_2x^{n-2} + \cdots + b_{n-1}x + b_n$$

thì (2.20) trở thành

$$a_1x^n + a_2x^{n-1} + \cdots + a_nx + a_{n+1} = (x - r)(b_1x^{n-1} + b_2x^{n-2} + \cdots + b_{n-1}x + b_n).$$

Đồng nhất hệ số của bội của x ở hai vế, ta có

$$b_1 = a_1, \quad b_2 = a_2 + rb_1, \quad \cdots, b_n = a_n + rb_{n-1}. \quad (2.21)$$

Điều này dẫn đến thuật toán *Horner giảm cấp* sau

```
b(1) = a(1);
for i=2:n
    b(i) = a(i) + r*b(i-1);
end
```

VÍ DỤ 2.4.1. Một 0-điểm của đa thức $P_4(x) = 3x^4 - 10x^3 - 48x^2 - 2x + 12$ là $x = 6$. Hãy giảm cấp đa thức bằng thuật toán Horner, tức là tìm $P_3(x)$ sao cho $P_4(x) = (x - 6)P_3(x)$.

GIẢI. Với $r = 6$ và $n = 4$, (2.21) trở thành

$$\begin{aligned} b_1 &= a_1 = 3 \\ b_2 &= a_2 + 6b_1 = -10 + 6(3) = 8 \\ b_3 &= a_3 + 6b_2 = -48 + 6(8) = 0 \\ b_4 &= a_4 + 6b_3 = -2 + 6(0) = -2. \end{aligned}$$

Do đó

$$P_3(x) = 3x^3 + 8x^2 - 2.$$

□

2.4.4 PHƯƠNG PHÁP LAGUERRE

Công thức Laguerre không dễ dàng suy ra được từ dạng tổng quát của đa thức $P_n(x)$. Tuy nhiên, nó có thể dễ dàng suy ra nếu ta xét trường hợp đặc biệt khi đa thức có 0-điểm tại r và có $(n - 1)$ 0-điểm tại q . Khi đó, đa thức có thể được viết dưới dạng

$$P_n(x) = (x - r)(x - q)^{n-1} \quad (a)$$

Bài toán của chúng ta trở thành: cho đa thức (a) dưới dạng

$$P_n(x) = a_1x^n + a_2x^{n-1} + \cdots + a_nx + a_{n+1},$$

tìm 0-điểm r của đa thức (chú ý rằng q chưa biết).???

Đạo hàm hai vế (a) theo x ta có

$$\begin{aligned} P'_n(x) &= (x - q)^{n-1} + (n - 1)(x - r)(x - q)^{n-2} \\ &= P_n(x) \left(\frac{1}{x - r} + \frac{n - 1}{x - q} \right). \end{aligned}$$

Do đó

$$\frac{P'_n(x)}{P_n(x)} = \frac{1}{x - r} + \frac{n - 1}{x - q}. \quad (b)$$

Đạo hàm hai vế ta có

$$\frac{P''_n(x)}{P_n(x)} - \left[\frac{P'_n(x)}{P_n(x)} \right]^2 = -\frac{1}{(x - r)^2} - \frac{n - 1}{(x - q)^2}. \quad (c)$$

Để thuận lợi ta ký hiệu

$$G(x) = \frac{P'_n(x)}{P_n(x)}, \quad H(x) = G^2(x) - \frac{P''_n(x)}{P_n(x)} \quad (2.22)$$

thì (b) và (c) sẽ trở thành

$$G(x) = \frac{1}{x - r} + \frac{n - 1}{x - q}. \quad (2.23a)$$

$$H(x) = \frac{1}{(x - r)^2} + \frac{n - 1}{(x - q)^2}. \quad (2.23b)$$

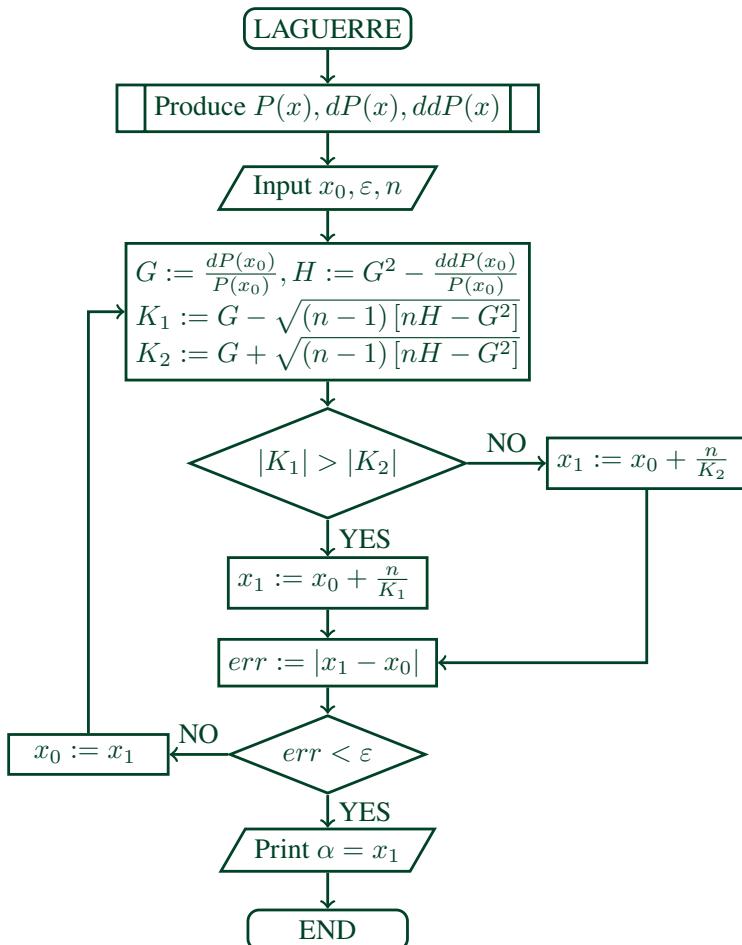
Nếu giải (2.23a) theo $x - r$ rồi thay vào (2.23b), ta sẽ thu được phương trình bậc hai theo $x - r$. Nghiệm của phương trình là *công thức Laguerre*

$$x - r = \frac{n}{G(x) \pm \sqrt{(n - 1)[nH(x) - G^2(x)]}}. \quad (2.24)$$

Quá trình tìm 0-điểm của một đa thức tổng quát bằng công thức Laguerre được mô tả bằng sơ đồ khối như trong Hình 2.12 và được thực hiện cụ thể như sau:

1. Cho x là một xấp xỉ nghiệm của $P_n(x) = 0$.
2. Tính $P_n(x), P'_n(x)$ và $P''_n(x)$ bằng các công thức (2.18) và (2.19).
3. Tính $G(s), H(x)$ bằng công thức (2.22).
4. Xác định nghiệm xấp xỉ mới r từ (2.24). Chú ý chọn dấu sao cho mẫu số có trị tuyệt đối lớn (điều này cải thiện tốc độ hội tụ).
5. Đặt $x := r$ và lặp lại các bước 2-5 đến khi $|P_n(x)| < \varepsilon$ hoặc $|x - r| < \varepsilon$, với ε là sai số cho phép.

Một đặc tính tuyệt vời của phương pháp Laguerre là hội tụ đến một nghiệm từ bất kỳ giá trị bắt đầu nào của x (ngoại trừ một số rất ít trường hợp đặc biệt).



Hình 2.12: Sơ đồ khôi của phương pháp lặp Laguerre xấp xỉ 0-diểm đa thức

polyroots

Hàm polyroots tính tất cả các nghiệm của $P_n(x) = 0$ với đa thức $P_n(x)$ có các hệ số được lưu trữ trong mảng $a = [a_1, a_2, \dots, a_{n+1}]$. Sau khi nghiệm đầu tiên được tính toán bằng hàm con laguerre, đa thức được giảm cấp bằng cách sử dụng hàm

`deflpoly` và 0-điểm tiếp theo được tính bằng cách áp dụng hàm `laguerre` cho đa thức giảm cấp. Quá trình này được lặp lại cho đến khi n nghiệm được tìm thấy. Nếu một nghiệm được tính toán có phần ảo quá nhỏ thì có thể nó bị gây ra do lỗi làm tròn, khi đó `polyroots` sẽ thay thế phần ảo rất nhỏ này bằng không.

```

function root = polyroots(a,tol)
% Returns all the roots of the polynomial
% a(1)*x^n + a(2)*x^(n-1) + ... + a(n+1).
% USAGE: root = polyroots(a,tol).
% tol = error tolerance (default is 1.0e4*eps).
if nargin == 1; tol = 1.0e-6; end
n = length(a) - 1;
root = zeros(n,1);
for i=1:n
    x = laguerre(a,tol);
    if abs(imag(x))<tol;x=real(x);end
    root(i) = x;
    a = deflpoly(a,x);
end

function x=laguerre(a,tol)
% Returns a root of the polynomial
% a(1)*x^n + a(2)*x^(n-1) + ... + a(n+1).
x = randn; % Start with random number
n = length(a) - 1;
for i=1:30
    [p,dp,ddp] = evalpoly(a,x);
    if abs(p) < tol; return; end
    g=dp/p;h=g*g-ddp/p;
    f = sqrt((n - 1)*(n*h - g*g));
    if abs(g + f) >= abs(g - f); dx = n/(g + f);
    else; dx = n/(g - f); end
    x=x-dx;
    if abs(dx) < tol; return; end
end
error('Too many iterations in laguerre')

function b=deflpoly(a,r)

```

```
% Horner's deflation:
% a(1)*x^n + a(2)*x^(n-1) + ... + a(n+1)
% = (x - r) [b(1)*x^(n-1) + b(2)*x^(n-2) + ... + b(n)] .
n = length(a) - 1;
b = zeros(n,1);
b(1) = a(1);
for i=2:n; b(i)=a(i)+r*b(i-1); end
```

Vì các nghiệm được tính với độ chính xác hữu hạn, mỗi lần giảm cấp tích luỹ một lượng nhỏ sai số vào các hệ số của đa thức giảm cấp. Sai số làm tròn tích luỹ tăng cùng với bậc của đa thức và có thể trở nên rất tệ nếu đa thức là điều kiện xấu. Do đó, các kết quả nên được xem xét thận trọng khi xử lý các đa thức bậc cao.

Sai số gây ra bởi giảm cấp có thể được thu gọn lại bằng cách tính toán lại từng nghiệm mà sử dụng đa thức gốc, chưa giảm cấp. Các nghiệm thu được trước đây kết hợp với giảm cấp được sử dụng làm giá trị bắt đầu.

VÍ DỤ 2.4.2. Một nghiệm của đa thức $P_3(x) = x^3 - 4.0x^2 - 4.48x + 26.1$ xấp xỉ bằng $x = 3 - i$. Hãy tìm một ước lượng của nghiệm bằng cách lặp công thức Laguerre một lần.

GIẢI. Sử dụng ước lượng đã cho như giá trị ban đầu, ta tính

$$x = 3 - i, \quad x^2 = 8 - 6i, \quad x^3 = 18 - 26i.$$

Thay các giá trị này vào $P_3(x)$ và các đạo hàm của nó ta có

$$\begin{aligned} P_3(x) &= x^3 - 4.0x^2 - 4.48x + 26.1 \\ &= (18 - 6i) - 4.0(8 - 6i) - 4.48(3 - i) + 26.1 = -1.34 + 2.48i \\ P'_3(x) &= 3x^2 - 8.0x - 4.48 \\ &= 3(8 - 6i) - 8.0(3 - i) - 4.48 = -4.48 - 10.0i \\ P''_3(x) &= 6x - 8.0 = 6(3 - i) - 8.0 = 10.0 - 6.0i. \end{aligned}$$

Công thức (2.22) thu được là

$$\begin{aligned} G(x) &= \frac{P'_3(x)}{P_3(x)} = \frac{-4.48 - 10.0i}{-1.34 + 2.48i} = -2.36557 + 3.08462i \\ H(x) &= G^2(x) - \frac{P''_3(x)}{P_3(x)} = (-2.36557 + 3.08462i)^2 - \frac{10.0 - 6.0i}{-1.34 + 2.48i} \\ &= 0.35995 - 12.488452i. \end{aligned}$$

Biểu thức căn bậc hai ở mẫu số trong công thức (2.24) trở thành

$$\begin{aligned} F(x) &= \sqrt{(n-1)[nH(x) - G^2(x)]} \\ &= \sqrt{2[3(0.35995 - 12.488452i) - (-2.36557 + 3.08462i)^2]} \\ &= \sqrt{5.67822 - 45.71946i} = 5.08670 - 4.49402i. \end{aligned}$$

Dấu trong (2.24) được chọn bằng cách tính độ lớn của mẫu số:

$$\begin{aligned} |G(x) + F(x)| &= |(-2.36557 + 3.08462i) + (5.08670 - 4.49402i)| \\ &= |2.72113 - 1.40940i| = 3.06448 \\ |G(x) - F(x)| &= |(-2.36557 + 3.08462i) - (5.08670 - 4.49402i)| \\ &= |-7.45227 + 7.57864i| = 10.62884. \end{aligned}$$

Sử dụng dấu trừ trong (2.24), ta thu được nghiệm xấp xỉ là

$$\begin{aligned} r = x - \frac{n}{G(x) - F(x)} &= (3-i) - \frac{3}{-7.45227 + 7.57864i} \\ &= 3.19790 - 0.79875i. \end{aligned}$$

Nhờ giá trị khởi đầu tốt, giá trị xấp xỉ đã rất gần với nghiệm chính xác là $3.20 - 0.80i$.

□

VÍ DỤ 2.4.3. Sử dụng `polyroots` tìm tất cả các nghiệm của $x^4 - 5x^3 - 9x^2 + 155x - 250 = 0$.

GIẢI. Thực hiện lệnh

```
>> polyroots([1 -5 -9 155 -250])
```

ta có kết quả là

```
ans =
```

```
2.0000
-5.0000
4.0000 + 3.0000i
4.0000 - 3.0000i
```

Phương trình có hai nghiệm thực ($x = 2$ và $x = -5$) và một cặp nghiệm phức liên hợp ($x = 4 \pm 3i$).

□

2.5 HÀM MATLAB GIẢI PHƯƠNG TRÌNH

Một số hàm MATLAB thông dụng dùng để xấp xỉ nghiệm của phương trình sẽ được giới thiệu sau đây.

- $x = \text{fzero}(@\text{func}, x_0)$ trả về 0-diểm của hàm func gần x_0 nhất.
- $x = \text{fzero}(@\text{func}, [a b])$ có thể được sử dụng khi nghiệm nằm trong $[a b]$.
- $x = \text{roots}(a)$ trả về tất cả các 0-diểm của đa thức có hệ số a.

TÓM TẮT CHƯƠNG 2

Nội dung chính của Chương 2 là trình bày một số phương pháp xấp xỉ nghiệm của phương trình. Các phương pháp xấp xỉ nghiệm được chia thành hai lớp là các phương pháp khoảng và các phương pháp mở. Lớp các phương pháp khoảng, yêu cầu khoảng chứa nghiệm, bao gồm:

1. Phương pháp tìm kiếm gia tăng
2. Phương pháp chia đôi
3. Phương pháp dây cung

Các phương pháp khoảng thường có ưu điểm là ổn định nhưng tốc độ hội tụ chậm. Các phương pháp mở có thể khắc phục điều này nhưng yêu cầu một số điều kiện chặt chẽ hơn. Hai phương pháp mở đã được giới thiệu là:

1. Phương pháp lặp
2. Phương pháp Newton-Raphson

Ta có thể sử dụng các phương pháp mở và các phương pháp khoảng để tìm 0-diểm của đa thức. Tuy nhiên, các phương pháp này không tìm được nghiệm phức. Do đó, dựa vào đặc điểm của đa thức, phần cuối Chương 2 đã giới thiệu phương pháp lặp Laguerre để tìm 0-diểm của đa thức một cách hiệu quả.

BÀI TẬP CHƯƠNG 2

Bài tập 2.1. Sử dụng 3 bước lặp của phương pháp chia đôi tìm nghiệm trong khoảng $[0, 2]$ của phương trình $x^3 - 2 = 0$.

Bài tập 2.2. Với mỗi hàm số và khoảng tương ứng sau đây, hãy sử dụng phương pháp chia đôi để xấp xỉ 0-điểm với độ chính xác không quá 0, 1.

- a. $f(x) = x - e^{-x^2}$, $[a, b] = [0, 1]$.
- b. $f(x) = \ln x + x$, $[a, b] = [1/10, 1]$.
- c. $f(x) = x^3 - 3$, $[a, b] = [0, 3]$.
- d. $f(x) = x^6 - x - 1$, $[a, b] = [0, 2]$.
- e. $f(x) = 3 - 2^x$, $[a, b] = [0, 2]$.

Bài tập 2.3. Sử dụng phương pháp chia đôi tìm nghiệm dương nhỏ nhất (thực) của phương trình $x^3 - 3.23x^2 - 5.54x + 9.84 = 0$.

Bài tập 2.4. Một nghiệm của phương trình $\tan x - \tanh x = 0$ nằm trong $(7.0, 7.4)$. Hãy tìm nghiệm này bằng phương pháp chia đôi với độ chính xác ba chữ số thập phân.

Bài tập 2.5. Sử dụng phương pháp Newton-Raphson tìm nghiệm dương khác không nhỏ nhất nằm trong khoảng $(4, 5)$ của phương trình $\cosh x \cos x - 1 = 0$.

Bài tập 2.6. Sử dụng 3 bước lặp phương pháp Newton-Raphson xấp xỉ nghiệm các phương trình sau đây.

- a. $x^6 - x - 1 = 0$, $x_0 = 1$.
- b. $x + \tan x = 0$, $x_0 = 3$.
- c. $1 - 2xe^{-x/2} = 0$, $x_0 = 0$.
- d. $x^2 - \sin x = 0$, $x_0 = 1/2$.
- e. $5 - x^{-1} = 0$, $x_0 = 1/2$.
- f. $x^3 - 2x - 5 = 0$, $x_0 = 2$.
- g. $x^2 - \sin x = 0$, $x_0 = 1/2$.
- h. $x^3 - 2 = 0$, $x_0 = 1$.

- i. $x - e^{-x} = 0, x_0 = 1.$
- j. $2 - x^{-1} \ln x = 0, x_0 = 1/3.$

Bài tập 2.7. Sử dụng phương pháp Newton-Raphson tìm nghiệm nằm trong khoảng $(-2, 2)$ của phương trình $\sin x + 3 \cos x - 2 = 0$.

Bài tập 2.8. Hãy xây dựng công thức lặp đơn để tìm một nghiệm cho phương trình sau.

- a. $x + \ln x = 2.$
- b. $5x^2 - x - 1.$
- c. $x^2 - 10 \ln x - 3 = 0.$
- d. $(x - 1)^2 = \frac{1}{2}e^x.$

Bài tập 2.9. Hãy viết chương trình sử dụng phương pháp chia đôi để xấp xỉ 0-điểm của các hàm trên khoảng tương ứng sau đây với sai số không quá 10^{-6} .

- a. $f(x) = x^3 - 2, [a, b] = [0, 2].$
- b. $f(x) = e^x - 2, [a, b] = [0, 1].$
- c. $f(x) = x - e^x, [a, b] = [0, 1].$
- d. $f(x) = x^6 - x - 1, [a, b] = [0, 2].$
- e. $f(x) = x^3 - 2x - 5, [a, b] = [0, 3].$
- f. $f(x) = 1 - 2xe^{-x/2}, [a, b] = [0, 2].$
- g. $f(x) = 5 - x^{-1}, [a, b] = [0.1, 0.25].$
- h. $f(x) = x^2 - \sin x, [a, b] = [0.1, \pi].$

Bài tập 2.10. Cho trước một 0-điểm $x = r$ của đa thức $P_n(x)$. Hãy kiểm tra lại r thực sự là một 0-điểm và sau đó giảm cấp đa thức $P_n(x)$, tức là tìm $P_{n-1}(x)$ sao cho $P_n(x) = (x - r)P_{n-1}(x)$.

- a. $P_3(x) = 3x^3 + 7x^2 - 36x + 20, r = -5.$
- b. $P_4(x) = x^4 - 3x^2 + 3x - 1, r = 1.$

c. $P_5(x) = x^5 - 30x^4 + 361x^3 - 2178x^2 + 6588x - 7992$, $r = 6$.

d. $P_4(x) = x^4 - 5x^3 - 2x^2 - 20x - 24$, $r = 2i$.

e. $P_3(x) = 3x^3 - 19x^2 + 45x - 13$, $r = 3 - 2i$.

Bài tập 2.11. Xác định tất cả các nghiệm của $x^4 + 0.9x^3 - 2.3x^2 + 3.6x - 25.2 = 0$.

Bài tập 2.12. Tính tất cả các nghiệm dương của $x^4 + 2x^3 - 7x^2 + 3 = 0$.

Bài tập 2.13. Tính tất cả các nghiệm dương của $\sin x - 0.1x = 0$.

Bài tập 2.14. Tìm tất cả các 0-điểm của đa thức $P_n(x)$.

a. $P_4(x) = x^4 + 2.1x^3 - 2.52x^2 + 2.1x - 3.52$.

b. $P_5(x) = x^5 - 156x^4 - 5x^5 + 780x^2 + 4x - 624$.

c. $P_6(x) = x^6 + 4x^5 - 8x^4 - 34x^3 + 57x^2 + 130x - 150$.

d. $P_7(x) = 8x^7 + 28x^6 + 34x^5 - 13x^4 - 124x^3 + 19x^2 + 220x - 100$.

e. $P_8(x) = x^8 - 7x^7 + 7x^6 + 25x^5 + 24x^4 - 98x^3 - 472x^2 + 440x + 800$.

CHƯƠNG 3

TÍNH GẦN ĐÚNG NGHIỆM CỦA HỆ PHƯƠNG TRÌNH ĐẠI SỐ TUYẾN TÍNH

3.1	Giới thiệu	54
3.2	Phương pháp khử Gauss	59
3.2.1	Giới thiệu	59
3.2.2	Thuật toán khử Gauss	61
3.3	Phương pháp phân rã LU	69
3.3.1	Giới thiệu	69
3.3.2	Phương pháp phân rã Doolittle	70
3.3.3	Phương pháp phân rã Cholesky	76
3.4	Ma trận dải và ma trận đối xứng	81
3.4.1	Ma trận ba đường chéo	82
3.4.2	Ma trận đối xứng	86
3.4.3	Ma trận dải đối xứng	89
3.5	Phần tử xoay	95
3.5.1	Giới thiệu	95
3.5.2	Ma trận chéo trội	96
3.5.3	Phương pháp khử Gauss kết hợp đổi hàng xoay	97
3.6	Phương pháp lặp đơn	106
3.6.1	Giới thiệu	106
3.6.2	Phương pháp lặp Jacobi	107
3.6.3	Phương pháp lặp Seidel	110
3.6.4	Phương pháp Gradient liên hợp	117
3.7	Một số hàm Matlab tính toán trong đại số tuyến tính	124

3.1 GIỚI THIỆU

Trong chương này chúng ta xét các phương pháp tìm nghiệm của hệ n phương trình đại số tuyến tính, n ẩn (đôi khi gọi vẫn tắt là hệ phương trình tuyến tính). Đây là một chương dài và là một trong những chương quan trọng nhất của quyển sách này. Lý do cho việc này là hầu hết các tính toán số sẽ không thể thực hiện được nếu không giải hệ phương trình đại số tuyến tính. Hơn nữa, các hệ phương trình tuyến tính phát sinh từ việc giải các bài toán vật lý trong thực tế thường có kích thước rất lớn nên gây tốn kém cho việc lưu trữ và tính toán. Việc giảm lưu trữ và thời gian tính toán các hệ phương trình tuyến tính này thường được dựa vào một số tính chất của hệ số của ma trận, chẳng hạn như tính thưa (hầu hết các phần tử của ma trận là số không). Do vậy, có một số thuật toán tìm nghiệm của hệ tuyến tính có kích thước lớn được xây dựng, mỗi thuật toán này thường dựa vào các tính chất của hệ số của ma trận để tính toán (tính đối xứng, tính thưa, ...).

Vì có rất nhiều các thuật toán tìm nghiệm hệ phương trình tuyến tính nên chúng ta không thể thảo luận hết các thuật toán trong mục này. Cách tốt nhất là chúng ta sẽ giới thiệu một số thuật toán cơ bản, trong đó có bổ sung một số thuật toán hữu ích cho việc tính toán liên quan đến các ma trận thưa.

Một hệ phương trình tuyến tính có dạng

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases} \quad (3.1)$$

với a_{ij} là các hệ số và b_i là hệ số tự do đã biết, x_i là các biến. Hệ phương trình (3.1) có thể biểu diễn dưới dạng ma trận

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \quad (3.2)$$

hoặc đơn giản là

$$\mathbf{Ax} = \mathbf{b}. \quad (3.3)$$

Một cách ký hiệu hệ phương trình tuyến tính đặc biệt hữu ích trong tính toán gọi là *ma trận hệ số mở rộng*, là cách ký hiệu nhận được bằng cách ghép vectơ b với ma trận hệ số A dạng

$$\left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{array} \right] \quad (3.4)$$

Hệ n phương trình tuyến tính n ẩn có nghiệm duy nhất khi ma trận hệ số không suy biến, tức là $|A| \neq 0$. Khi đó, các hàng hoặc các cột của ma trận hệ số *độc lập tuyến tính*, tức là không có hàng (cột) nào có thể biểu diễn dưới dạng tổ hợp tuyến tính của các hàng (cột) còn lại.

Trong trường hợp ma trận hệ số *suy biến* thì hệ tuyến tính có thể có vô số nghiệm hoặc vô nghiệm phụ thuộc vào vectơ hệ số b .

Một câu hỏi tự nhiên đặt ra là: việc gì sẽ xảy ra khi ma trận hệ số *hầu như suy biến*, tức là $|A|$ rất nhỏ? Để xác định xem khi nào định thức của ma trận hệ số “nhỏ” chúng ta cần một tham chiếu mà có thể đo được định thức của ma trận. Tham chiếu đó được gọi là *chuẩn* của ma trận, được ký hiệu bởi $\|A\|$. Chúng ta có thể nói định thức của ma trận A nhỏ nếu

$$|A| \ll \|A\|.$$

Có nhiều chuẩn ma trận được đưa ra, trong quyển sách này chúng ta sử dụng hai loại chuẩn ma trận sau.

$$\|A\| = \sqrt{\sum_{i=1}^n \sum_{j=1}^n |a_{ij}|^2} \quad (3.5)$$

$$\|A\| = \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}| \quad (3.6)$$

Một đại lượng đo tình trạng suy biến của một ma trận là *số điều kiện ma trận*, được xác định bởi

$$\text{cond}(A) = \|A\| \|A^{-1}\|. \quad (3.7)$$

Nếu $\text{cond}(A) = 1$ thì ma trận A là ma trận điều kiện tốt (well-conditioned). Số điều kiện ma trận tăng theo mức độ điều kiện xấu (ill-conditioned) của ma trận, và nhận giá trị vô cùng khi ma trận suy biến. Lưu ý rằng số điều kiện của ma trận không duy nhất vì phụ thuộc vào chuẩn của ma trận. Không may là việc tính toán số điều

kiện của ma trận cỡ lớn rất tốn kém nên trong hầu hết các trường hợp, số điều kiện của ma trận được ước lượng bằng cách so sánh định thức với độ lớn của các phần tử của ma trận.

Nếu một hệ phương trình điều kiện xấu thì một sự thay đổi nhỏ của hệ số của ma trận sẽ dẫn đến sự thay đổi lớn của nghiệm. Ví dụ sau đây minh họa cho nhận xét này.

VÍ DỤ 3.1.1. Xét hệ phương trình

$$\begin{cases} 2x + y &= 3 \\ 2x + 1.001y &= 0 \end{cases}$$

GIẢI. Ta thấy hệ phương trình trên có nghiệm $x = 1501.5; y = -3000$. Do $|A| = 2(1.001) - 2(1) = 0.002$ rất nhỏ so với các hệ số của ma trận A nên đây là hệ điều kiện xấu. Nếu thay phương trình thứ hai trong hệ bởi phương trình $2x + 1.002y = 0$ thì ta có nghiệm $x = 751.5; y = -1500$. Vậy với sự thay đổi rất nhỏ hệ số của y (0,1%) thì dẫn đến sự thay đổi rất lớn của nghiệm (100%).

□

Nghiệm số của hệ phương trình điều kiện xấu không đáng tin cậy. Lý do vì trong quá trình tính toán, các hệ số của ma trận luôn phải làm tròn số, quá trình tích luỹ thay đổi nhỏ này sẽ dẫn đến sai số rất lớn của nghiệm. Trong trường hợp bị nghi ngờ là hệ điều kiện xấu, định thức của ma trận hệ số được tính toán sao cho bậc của điều kiện xấu có thể được ước lượng. Việc này được thực hiện trong lúc hoặc sau khi giải với chỉ một nỗ lực tính toán nhỏ.

Hệ phương trình đại số tuyến tính xuất hiện trong rất nhiều lĩnh vực của kỹ thuật như chất rắn đàn hồi, nhiệt lượng, thẩm thấu của chất lỏng, trường điện từ và mạch điện, Nhưng ứng dụng rõ ràng nhất trong kỹ thuật là việc phân tích hệ động lực tuyến tính (bất kỳ hệ có phản hồi nào trong kỹ thuật có thể được xem như là hệ tuyến tính).

Nếu một mô hình có cấu trúc rời rạc, chẳng hạn một mạch điện, thì việc phân tích mô hình này sẽ dẫn đến trực tiếp một hệ phương trình tuyến tính. Nếu mô hình có cấu trúc liên tục thì sẽ được mô tả bằng phương trình vi phân hoặc hệ phương trình đại số. Tuy nhiên, việc tính toán chỉ được thực hiện khi mô hình có cấu trúc rời rạc nên việc cần thiết đầu tiên là xấp xỉ các phương trình vi phân này để được hệ phương trình tuyến tính. Quá trình xấp xỉ này được gọi là quá trình *rời rạc hóa*.

Tóm lại, khi phân tích một số mô hình kỹ thuật thì luôn dẫn đến một hệ phương trình tuyến tính dạng $Ax = b$, với b là đầu vào và x là phản hồi của hệ. Ma trận hệ

số A , với các tính chất phụ thuộc vào tính chất của mô hình gốc, độc lập với đầu vào b . Nói cách khác, nếu đầu vào thay đổi, thì hệ phương trình được giải với các giá trị khác nhau của b nhưng ma trận A không thay đổi. Do đó, các thuật toán để giải hệ phương trình luôn được thiết kế để giải hệ với bất kỳ vectơ đầu vào nào với số lượng tính toán tối thiểu.

Có hai nhóm phương pháp để giải hệ phương trình tuyến tính, đó là nhóm các phương pháp giải trực tiếp và nhóm các phương pháp lặp. Tính chất chung của nhóm các phương pháp trực tiếp là biến đổi hệ gốc về *hệ tương đương* (là hệ có cùng nghiệm) có thể giải dễ dàng. Các phương pháp biến đổi được sử dụng gọi là *phép biến đổi tuyến tính*, là các phép biến đổi có thể làm thay đổi giá trị định thức hoặc hệ số của ma trận nhưng không làm thay đổi nghiệm của hệ. Các phép biến đổi bao gồm:

- Đổi vị trí hai phương trình (làm thay đổi dấu của $|A|$);
- Nhân một phương trình với một hằng số khác không (Nhân $|A|$ với một hằng số);
- Lấy một phương trình trừ đi bội số (khác không) một phương trình khác ($|A|$ không đổi).

Các phương pháp lặp hoặc các phương pháp gián tiếp, bắt đầu bằng một dự đoán của x , và sau đó lặp đi lặp lại quá trình tính toán các xấp xỉ khác của x cho đến khi dừng theo một tiêu chí nào đó. Các phương pháp lặp hầu hết không hiệu quả bằng các phương pháp trực tiếp do yêu cầu số lượng phép lặp lớn. Tuy nhiên, chúng có những lợi thế đáng kể khi tính toán cho hệ có ma trận hệ số có kích thước lớn và có tính chất thưa (hầu hết các phần tử bằng không).

Bảng sau đây liệt kê một số phương pháp trực tiếp phổ biến mà sử dụng các phép biến đổi tuyến tính biến phương trình tuyến tính từ dạng ban đầu về dạng sau cùng để giải. Trong Bảng 3.1, U là ma trận tam giác trên, L là ma trận tam giác dưới, I là ma trận đơn vị. Một ma trận được gọi là ma trận *tam giác* nếu các phần tử nằm một phía của đường chéo chính bằng không. Chẳng hạn, ma trận tam giác trên cỡ 3×3 có dạng

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

Phương pháp	Dạng ban đầu	Dạng cuối cùng
Khử Gauss	$\mathbf{A}\mathbf{x} = \mathbf{b}$	$\mathbf{U}\mathbf{x} = \mathbf{c}$
Phân rã LU	$\mathbf{A}\mathbf{x} = \mathbf{b}$	$\mathbf{L}\mathbf{U}\mathbf{x} = \mathbf{b}$
Khử Gauss-Jordal	$\mathbf{A}\mathbf{x} = \mathbf{b}$	$\mathbf{I}\mathbf{x} = \mathbf{c}$

Bảng 3.1: Bảng liệt kê một số phương pháp trực tiếp giải hệ phương trình tuyến tính.

và ma trận tam giác dưới cỡ 3×3 có dạng

$$\begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{23} & l_{33} \end{bmatrix}$$

Ma trận tam giác đóng vai trò quan trọng trong đại số tuyến tính vì có nhiều cách để tính toán đơn giản. Ví dụ xét hệ phương trình $\mathbf{L}\mathbf{x} = \mathbf{c}$, hoặc

$$\begin{cases} l_{11}x_1 &= c_1 \\ l_{21}x_1 + l_{22}x_2 &= c_2 \\ l_{31}x_1 + l_{32}x_2 + l_{33}x_3 &= c_3 \\ &\vdots \end{cases}$$

Ta giải hệ phương trình bằng cách xét lần lượt từng phương trình, quá trình tính toán dễ ràng bắt đầu từ phương trình thứ nhất, sau đó mỗi phương trình chỉ chứa một biến chưa biết. Nghiệm của hệ phương trình thu được bằng quá trình sau

$$\begin{aligned} x_1 &= c_1/l_{11} \\ x_2 &= (c_2 - l_{21}x_1)/l_{22} \\ x_3 &= (c_3 - l_{31}x_1 - l_{32}x_2)/l_{33} \\ &\vdots \end{aligned}$$

Quá trình tìm nghiệm trên gọi là *thay thế thuận*. Bằng cách tương tự, nghiệm của hệ phương trình $\mathbf{U}\mathbf{x} = \mathbf{c}$ thu được bằng phép khử Gauss có thể được giải dễ ràng bằng *thay thế ngược* với việc bắt đầu bằng phương trình cuối cùng rồi thay thế ngược lên.

Hệ phương trình $\mathbf{L}\mathbf{U}\mathbf{x} = \mathbf{b}$, là dạng cuối cùng của phương pháp phân rã LU có thể giải nhanh chóng bằng cách thay thế bởi hai hệ phương trình tương đương: $\mathbf{Ly} = \mathbf{b}$ và

$Ux = y$. Nghiệm y của hệ $Ly = b$ có thể tìm được bằng quá trình thay thế thuận và sau đó hệ $Ux = y$ có thể tìm được x bằng quá trình thay thế ngược.

Hệ phương trình $Ix = c$, là dạng cuối của phương pháp khử Gauss-Jordan, tương đương với $x = c$ nên c chính là nghiệm của hệ phương trình.

VÍ DỤ 3.1.2. Giải hệ phương trình $Ax = b$, với

$$A = \begin{bmatrix} 8 & -6 & 2 \\ -4 & 11 & -7 \\ 4 & -7 & 6 \end{bmatrix}, b = \begin{bmatrix} 28 \\ -40 \\ 33 \end{bmatrix}$$

nếu biết phân rã LU của ma trận hệ số là

$$A = LU = \begin{bmatrix} 2 & 0 & 0 \\ -1 & 2 & 0 \\ 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} 4 & -3 & 1 \\ 0 & 4 & -3 \\ 0 & 0 & 2 \end{bmatrix}.$$

GIẢI. Trước tiên ta giải hệ phương trình $Ly = b$ bằng cách thay thế thuận

$$\begin{cases} 2y_1 &= 28 \\ -y_1 + 2y_2 &= -40 \\ y_1 - y_2 + y_3 &= 33 \end{cases} \Rightarrow \begin{cases} y_1 &= 28/2 = 14 \\ y_2 &= (-40 - y_1)/2 = -13 \\ y_3 &= 33 - y_1 + y_2 = 6 \end{cases}$$

Sau đó, nghiệm x thu được từ hệ $Ux = y$ bằng cách thay thế ngược

$$\begin{cases} 2x_3 &= y_3 \\ 4x_2 - 3x_3 &= y_2 \\ 4x_1 - 3x_2 + x_3 &= y_1 \end{cases} \Rightarrow \begin{cases} x_3 &= y_3/2 = 6/3 = 2 \\ x_2 &= (y_2 + 3x_3)/4 = -1 \\ x_1 &= (y_1 + 3x_2 - x_3)/4 = 2 \end{cases}.$$

Do đó, nghiệm của hệ phương trình là $x = [2 \quad -1 \quad 2]^T$.

□

3.2 PHƯƠNG PHÁP KHỬ GAUSS

3.2.1 GIỚI THIỆU

Phương pháp khử Gauss là một trong những phương pháp quan trọng nhất để giải hệ phương trình tuyến tính, nó bao gồm hai phần: *pha khử* và *pha giải*. Như đã đề cập

trong Bảng 3.1, pha khử là sử dụng các phép biến đổi tuyến tính để đưa hệ phương trình về dạng $Ux = c$. Sau đó, hệ phương trình được giải bằng cách thay thế ngược. Để minh họa cho phương pháp khử Gauss, ta xét ví dụ sau.

VÍ DỤ 3.2.1. Giải hệ phương trình sau bằng phương pháp khử Gauss

$$\begin{cases} 4x_1 - 2x_2 + x_3 &= 11 \\ -2x_1 + 4x_2 - 2x_3 &= -16 \\ x_1 - 2x_2 + 4x_3 &= 17 \end{cases} \quad (3.8a)$$

$$\begin{cases} 4x_1 - 2x_2 + x_3 &= 11 \\ -2x_1 + 4x_2 - 2x_3 &= -16 \\ x_1 - 2x_2 + 4x_3 &= 17 \end{cases} \quad (3.8b)$$

$$\begin{cases} 4x_1 - 2x_2 + x_3 &= 11 \\ -2x_1 + 4x_2 - 2x_3 &= -16 \\ x_1 - 2x_2 + 4x_3 &= 17 \end{cases} \quad (3.8c)$$

GIẢI. Pha khử: Pha khử sử dụng duy nhất phép biến đổi tuyến tính là lấy một phương trình trừ đi bội số (khác không) một phương trình khác, tức là

$$Pt.(i) := Pt.(i) - \lambda \times Pt.(j), \lambda \neq 0, \quad (3.9)$$

với $Pt.(j)$ được gọi là *phương trình xoay*.

Chúng ta bắt đầu khử bằng cách lấy phương trình (3.8a) là phương trình xoay và chọn hệ số λ sao cho x_1 ở phương trình (3.8b) và phương trình (3.8c) bị khử, tức là

$$Pt.(3.8b) := Pt.(3.8b) - (-0.5) \times Pt.(3.8a)$$

$$Pt.(3.8c) := Pt.(3.8c) - (0.25) \times Pt.(3.8a)$$

Sau khi biến đổi, hệ phương trình trở thành

$$\begin{cases} 4x_1 - 2x_2 + x_3 &= 11 \\ 3x_2 - 1.5x_3 &= -10.5 \\ -1.5x_2 + 3.75x_3 &= 14.25 \end{cases} \quad (3.8a)$$

$$\begin{cases} 4x_1 - 2x_2 + x_3 &= 11 \\ 3x_2 - 1.5x_3 &= -10.5 \\ -1.5x_2 + 3.75x_3 &= 14.25 \end{cases} \quad (3.8b)$$

$$\begin{cases} 4x_1 - 2x_2 + x_3 &= 11 \\ 3x_2 - 1.5x_3 &= -10.5 \\ 3x_3 &= 9 \end{cases} \quad (3.8c)$$

Tiếp theo, ta sử dụng phương trình (3.8b) là phương trình xoay để khử x_2 trong phương trình (3.8c)

$$Pt.(3.8c) := Pt.(3.8c) - (-0.5) \times Pt.(3.8b)$$

ta thu được hệ phương trình

$$\begin{cases} 4x_1 - 2x_2 + x_3 &= 11 \\ 3x_2 - 1.5x_3 &= -10.5 \\ 3x_3 &= 9 \end{cases} \quad (3.8a)$$

$$\begin{cases} 4x_1 - 2x_2 + x_3 &= 11 \\ 3x_2 - 1.5x_3 &= -10.5 \\ 3x_3 &= 9 \end{cases} \quad (3.8b)$$

$$\begin{cases} 4x_1 - 2x_2 + x_3 &= 11 \\ 3x_2 - 1.5x_3 &= -10.5 \\ 3x_3 &= 9 \end{cases} \quad (3.8c)$$

Đến đây đã hoàn thành pha khử, hệ phương trình ban đầu đã được thay thế bằng hệ phương trình tương đương có thể giải dễ ràng bằng cách thay thế ngược.

Như đã nói ở trên, ma trận hệ số mở rộng có thể là công cụ thích hợp cho quá trình tính toán. Hệ phương trình ban đầu có thể biểu diễn dưới dạng

$$\left[\begin{array}{ccc|c} 4 & -2 & 1 & 11 \\ -2 & 4 & -2 & -16 \\ 1 & -2 & 4 & 17 \end{array} \right]$$

và các hệ phương trình nhận được sau các bước khử thứ nhất và bước khử thứ hai của phương pháp khử Gauss được biểu diễn tương ứng dưới dạng

$$\left[\begin{array}{ccc|c} 4 & -2 & 1 & 11 \\ 0 & 3 & -1.5 & -10.5 \\ 0 & -1.5 & 3.75 & 14.25 \end{array} \right],$$

$$\left[\begin{array}{ccc|c} 4 & -2 & 1 & 11 \\ 0 & 3 & -1.5 & -10.5 \\ 0 & 0 & 3 & 19 \end{array} \right].$$

Ta thấy rằng quá trình biến đổi hệ phương trình là việc lấy một hàng trừ đi bội số khác không của hàng khác nên không làm thay đổi định thức của ma trận nên định thức của ma trận A và định thức của ma trận đường chéo U bằng nhau. Do đó ta có công thức tính định thức

$$|\mathbf{A}| = |\mathbf{U}| = u_{11} \times u_{22} \times \cdots \times u_{nn}. \quad (3.10)$$

Pha giải (pha thay thế ngược): Các biến có thể được tính toán dễ dàng bằng cách thay tính toán và thay thế ngược từ phương trình cuối. Giải lần lượt phương trình (3.8c), (3.8b) và (3.8a) ta có

$$\begin{aligned} x_3 &= 9/3 = 3, \\ x_2 &= (-10.5 + 1.5x_3)/3 = -2, \\ x_1 &= (11 + 2x_2 - x_3)/4 = 1. \end{aligned}$$

□

3.2.2 THUẬT TOÁN KHỬ GAUSS

Tương tự như ví dụ trên, thuật toán khử Gauss cho trường hợp tổng quát được thực hiện qua hai pha, cụ thể như sau.

Pha khử. Ta xét hệ phương trình tại một bước tuỳ ý trong pha khử. Giả sử k hàng đầu của ma trận \mathbf{A} đã được chuyển về dạng ma trận tam giác trên. Do đó, ta chọn phương trình thứ k làm phương trình xoay để khử biến x_k ở các phương trình dưới. Hệ phương trình tại bước này được minh họa bởi ma trận hệ số mở rộng sau. Lưu ý rằng các hệ số của ma trận \mathbf{A} và hệ số của vectơ hằng \mathbf{b} không phải là hệ số lúc đầu do đã bị biến đổi trong các bước khử trước.

$$\left[\begin{array}{ccccccccc|c} a_{11} & a_{12} & a_{13} & \cdots & a_{1k} & \cdots & a_{1j} & \cdots & a_{1n} & b_1 \\ 0 & a_{22} & a_{23} & \cdots & a_{2k} & \cdots & a_{2j} & \cdots & a_{2n} & b_2 \\ 0 & 0 & a_{33} & \cdots & a_{3k} & \cdots & a_{3j} & \cdots & a_{3n} & b_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{kk} & \cdots & a_{kj} & \cdots & a_{kn} & b_k \\ \hline \vdots & \vdots & \vdots & & \vdots & & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{ik} & \cdots & a_{ij} & \cdots & a_{in} & b_i \\ \vdots & \vdots & \vdots & & \vdots & & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{nk} & \cdots & a_{nj} & \cdots & a_{nn} & b_n \end{array} \right] \begin{matrix} \leftarrow \text{hàng xoay} \\ \leftarrow \text{hàng cần được biến đổi} \end{matrix}$$

Cho hàng i là hàng phía dưới hàng xoay ($i > k$) là hàng cần được biến đổi, có nghĩa là ta cần biến đổi sao cho $a_{ik} = 0$. Để làm điều này, ta lấy hàng i trừ đi $\lambda = a_{ik}/a_{kk}$ lần hàng xoay (hàng k). Khi đó, các phần tử của hàng i được tính toán như sau

$$a_{ij} := a_{ij} - \lambda a_{kj}, j = k, k+1, \dots, n, \quad (3.11a)$$

$$b_i := b_i - \lambda b_k. \quad (3.11b)$$

Để biến đổi ma trận hệ số \mathbf{A} thành ma trận tam giác trên thì k và i trong công thức (3.11) phải lần lượt nhận các giá trị $k = 1, 2, \dots, n-1$ (chọn hàng xoay), $i = k+1, k+2, \dots, n$ (chọn hàng cần được biến đổi). Thuật toán cho pha khử được mô tả như sau:

```

for k=1:n-1
    for i= k+1:n
        if A(i,k) ~= 0
            lambda = A(i,k)/A(k,k);
            A(i,k+1:n) = A(i,k+1:n) - lambda*A(k,k+1:n);
            b(i)= b(i) - lambda*b(k);
        end
    end
end

```

```
    end
end
```

Để tránh những phép toán không cần thiết, khi thực hiện thuật toán trên để minh họa cho (3.11), cần một số lưu ý sau:

- Nếu $a_{ik} = 0$ thì phép biến đổi với dòng i được bỏ qua.
- Chỉ số j trong (3.11a) thường bắt đầu bằng $k + 1$ chứ không phải k . Do đó, a_{ik} không được thay thế bằng không mà vẫn giữ giá trị gốc. Vì pha giải không cần sử dụng đến các phần tử ở vị trí phía dưới đường chéo chính nên việc này không ảnh hưởng đến thuật toán.

Pha giải (thay thế ngược): Sau pha khử, ma trận hệ số mở rộng của hệ phương trình có dạng

$$[\mathbf{A}|\mathbf{b}] = \left[\begin{array}{ccccc|c} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} & b_1 \\ 0 & a_{22} & a_{23} & \cdots & a_{2n} & b_2 \\ 0 & 0 & a_{23} & \cdots & a_{2n} & b_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{nn} & b_n \end{array} \right].$$

Giải phương trình cuối, $a_{nn}x_n = b_n$, ta có

$$x_n = \frac{b_n}{a_{nn}}. \quad (3.12)$$

Bằng cách thay thế ngược, giả sử ta đã tính được $x_n, x_{n-1}, \dots, x_{k+1}$, ta sẽ tính x_k từ phương trình thứ k

$$a_{kk}x_k + a_{k,k+1}x_{k+1} + \cdots + a_{nn}x_n = b_k$$

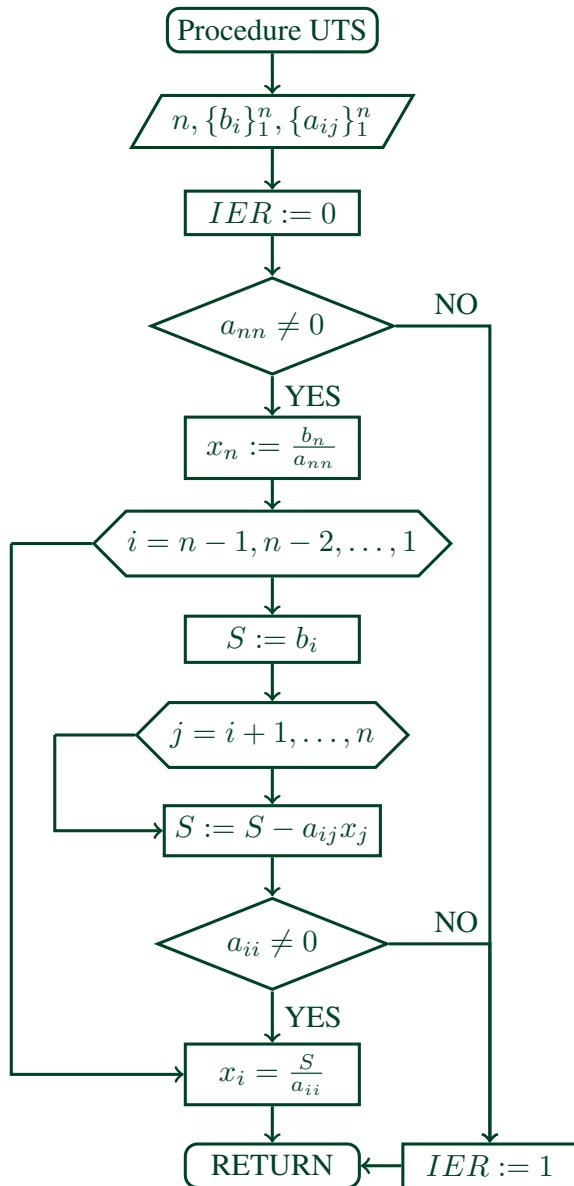
ta có

$$x_k = \left(b_k - \sum_{j=k+1}^n a_{kj}x_j \right) \frac{1}{a_{kk}}, k = n-1, n-2, \dots, 1. \quad (3.13)$$

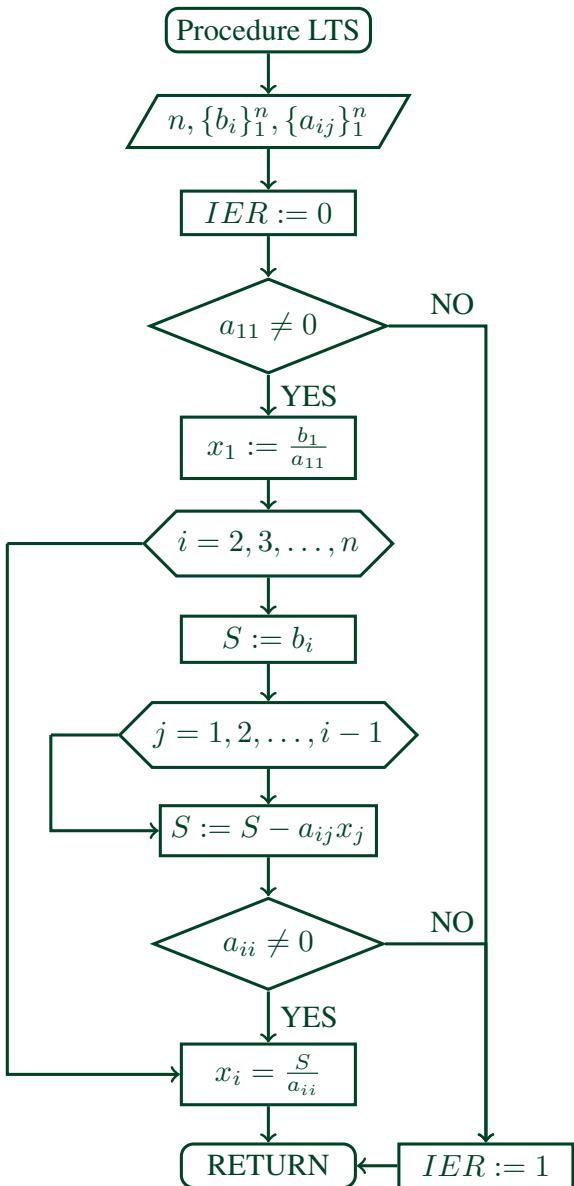
Thuật toán tương ứng tính toán theo công thức trên là

```
for k=n:-1:1
    b(k) = (b(k) - A(k, k+1:n) * b(k+1:n)) / A(k, k);
end
```

Quá trình tính toán của phương pháp khử Gauss để giải hệ phương trình tuyến tính được minh họa bằng sơ đồ khối trong Hình 3.2, trong đó pha thay thế ngược được minh họa bằng sơ đồ khối cho trong Hình 3.1a.



(a) Giải hệ tam giác trên



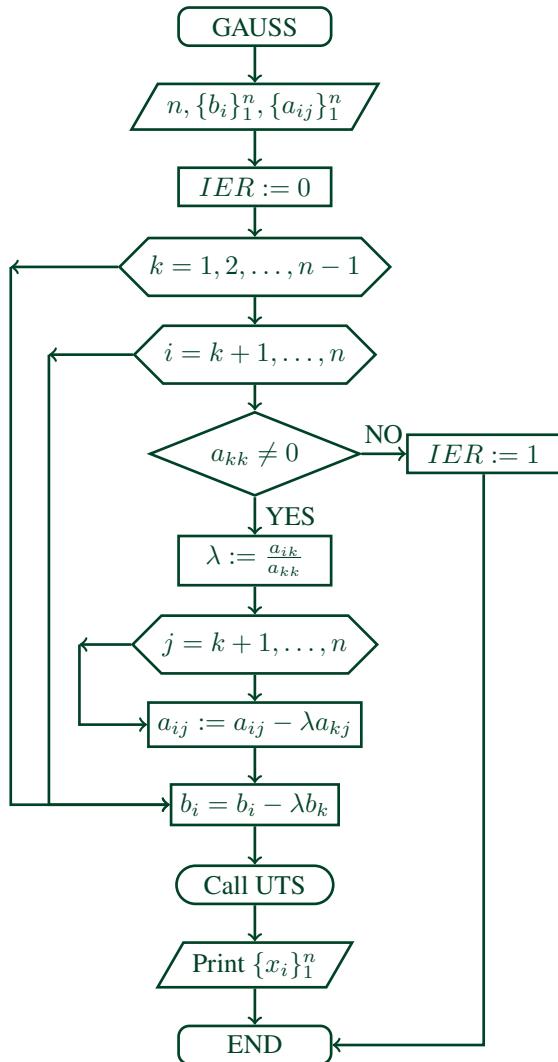
(b) Giải hệ tam giác dưới

Hình 3.1: Sơ đồ khối các thủ tục giải hệ phương trình tuyến tính dưới dạng tam giác

 gauss.

Hàm gauss kết hợp pha khử và pha giải. Trong khi thay thế ngược, b được ghi đè bởi vectơ nghiệm x, do đó b chứa nghiệm khi chương trình dừng.

```
function [x, det] = gauss(A, b)
```



Hình 3.2: Sơ đồ khối giải hệ phương trình tuyến tính bằng phương pháp khử Gauss

```

% Solves A*x=b by Gausse limination and computes det (A) .
% USAGE: [x,det] = gauss(A,b)
if size(b,2) > 1; b = b'; end % b must be column vector
n = length(b);
for k=1:n-1 %Elimination phase
    for i= k+1:n
        if A(i,k) ~= 0
            lambda = A(i,k)/A(k,k);
            A(i,k+1:n) = A(i,k+1:n) - lambda*A(k,k+1:n);
            b(i)= b(i) - lambda*b(k);
        end
    end
end
if nargout == 2; det = prod(diag(A)); end
for k=n:-1:1 %Back substitution phase

```

```

b(k) = (b(k) - A(k, k+1:n) * b(k+1:n)) / A(k, k);
end
x=b;

```

NHẬN XÉT 3.2.1. (i) Phương pháp phử Gauss chỉ thực hiện được khi các phần tử xoay tại mỗi bước khử khác không ($a_{kk} \neq 0$).

(ii) Trong trường hợp phương trình tuyến tính cần giải có dạng $\mathbf{AX} = \mathbf{B}$ với

$$\mathbf{X} = [\mathbf{x}^{(1)} \quad \mathbf{x}^{(2)} \quad \dots \quad \mathbf{x}^{(m)}], \quad \mathbf{B} = [\mathbf{b}^{(1)} \quad \mathbf{b}^{(2)} \quad \dots \quad \mathbf{b}^{(m)}].$$

tương ứng là các ma trận nghiệm và ma trận hằng số cỡ $n \times m$. Ta có thể giải hệ phương trình tuyến tính trên bằng cách áp dụng pha khử để đưa ma trận hệ số về dạng ma trận tạm giác, sau đó sẽ thực hiện pha thay thế ngược để tìm vectơ nghiệm $\mathbf{x}^{(j)}$ ứng với vectơ hằng $\mathbf{b}^{(j)}, j = 1, 2, \dots, m$.

Để minh họa nhận xét trên, ta xét ví dụ sau đây.

VÍ DỤ 3.2.2. Áp dụng phương pháp khử Gauss để giải hệ phương trình $\mathbf{AX} = \mathbf{B}$, với

$$\mathbf{A} = \begin{bmatrix} 6 & -4 & 1 \\ -4 & 6 & -4 \\ 1 & -4 & 6 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} -14 & 22 \\ 36 & -18 \\ 6 & 7 \end{bmatrix}.$$

GIẢI. Ma trận hệ số mở rộng có dạng

$$[\mathbf{A}|\mathbf{B}] = \left[\begin{array}{ccc|cc} 6 & -4 & 1 & -14 & 22 \\ -4 & 6 & 4 & 36 & -18 \\ 1 & -4 & 6 & 6 & 7 \end{array} \right].$$

Pha khử được thực hiện qua hai bước khử sau

$$\text{hàng 2 := hàng 2} + (2/3) \times \text{hàng 1}$$

$$\text{hàng 3 := hàng 3} - (1/6) \times \text{hàng 1}$$

$$\left[\begin{array}{ccc|cc} 6 & -4 & 1 & -14 & 22 \\ 0 & 10/3 & -10/3 & 80/3 & -10/3 \\ 0 & -10/3 & 35/6 & 25/3 & 10/3 \end{array} \right]$$

và

$$\text{hàng 3} := \text{hàng 3} + \text{hàng 2}$$

$$\left[\begin{array}{ccc|cc} 6 & -4 & 1 & -14 & 22 \\ 0 & 10/3 & -10/3 & 80/3 & -10/3 \\ 0 & 0 & 5/2 & 35 & 0 \end{array} \right].$$

Trong pha giải, trước tiên ta tính $\mathbf{x}^{(1)}$ bằng cách thay thế ngược

$$\begin{aligned} x_{31} &= \frac{35}{5/2} = 14, \\ x_{21} &= \frac{80/3 + 10/3x_{31}}{10/3} = 22, \\ x_{11} &= \frac{-14 + 4x_{21} - x_{31}}{6} = 10. \end{aligned}$$

Do đó, vectơ thứ nhất của nghiệm là $\mathbf{x}^{(1)} = [10 \quad 22 \quad 14]^T$.

Vectơ thứ hai của nghiệm được tính bằng cách thay thế ngược tương tự như sau

$$\begin{aligned} x_{32} &= \frac{0}{5/2} = 0, \\ x_{22} &= \frac{-10/3 + 10/3x_{31}}{10/3} = -1, \\ x_{12} &= \frac{22 + 4x_{21} - x_{31}}{6} = 3, \end{aligned}$$

nên $\mathbf{x}^{(2)} = [3 \quad -1 \quad 0]^T$. Suy ra nghiệm của hệ là

$$\mathbf{X} = [\mathbf{x}^{(1)} \quad \mathbf{x}^{(2)}] = \begin{bmatrix} 10 & 3 \\ 22 & -1 \\ 14 & 0 \end{bmatrix}.$$

□

VÍ DỤ 3.2.3. Ma trận Vandermode cỡ $n \times n$ là ma trận $\mathbf{A} = [a_{ij}]$ được xác định bởi

$$a_{ij} = v_i^{n-j}, \quad i = 1, 2, \dots, n, \quad j = 1, 2, \dots, n$$

với \mathbf{v} là một vectơ. Trong MATLAB, một ma trận Vandermode được tạo ra bằng hàm `vander(v)`. Sử dụng hàm `gauss` tính nghiệm của hệ phương trình $\mathbf{Ax} = \mathbf{b}$ với \mathbf{A} là ma trận Vandermode cỡ 6×6 được tạo ra từ vectơ

$$\mathbf{v} = [1.0 \quad 1.2 \quad 1.4 \quad 1.6 \quad 1.8 \quad 2.0]$$

và

$$\mathbf{v} = [0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1].$$

Ước lượng độ chính xác của nghiệm (Các ma trận Vandermode dẫn đến hệ điều kiện xấu).

GIẢI. Chúng ta sử dụng chương trình sau đây. Sau khi xây dựng \mathbf{A}, \mathbf{b} , chúng ta định dạng 14 chữ số thập phân để tính nghiệm (format long). Kết quả là

```
>> % Example 3.2.3 (Gauss elimination)
A = vander(1:0.2:2);
b=[0 1 0 1 0 1]';
format long
[x, det] = gauss(A, b)

x =
1.0e+004 *
0.04166666667010
-0.312500000002465
0.925000000006972
-1.350000000009722
0.97093333340017
-0.275100000001813

det =
-1.132462079991823e-006
```

Vì định thức rất nhỏ so với các phần tử của \mathbf{A} . Ta nghi ngờ có thể xảy ra lỗi làm tròn. Do nghiệm chính xác là

$$\mathbf{x} = [1250/3 \quad -3125 \quad 9250 \quad -13500 \quad 29128/3 \quad -2751]^T$$

nên trong trường hợp này, nghiệm số chính xác đến chín chữ số thập phân.

Một cách khác để đo độ chính xác là tính \mathbf{Ax} rồi so sánh với \mathbf{b} :

```
>> A*x
```

```

ans =
0
1.00000000000910
-0.00000000000909
0.99999999997272
-0.000000000024556
1.00000000000910

```

Kết quả đúng như xác nhận kết luận bên trên.

□

3.3 PHƯƠNG PHÁP PHÂN RÃ LU

3.3.1 GIỚI THIỆU

Ta luôn có thể phân rã một ma trận vuông A bất kỳ là tích của một ma trận tam giác dưới L với một ma trận tam giác trên U :

$$A = LU. \quad (3.14)$$

Quá trình tính toán các ma trận L và U gọi là quá trình phân rã LU. Có rất nhiều phương pháp phân rã LU cho ma trận A , bảng sau đây liệt kê ba phương pháp phân rã LU phổ biến.

Phương pháp	Đặc điểm
Phân rã Doolittle	$l_{ii} = 1, i = 1, 2, \dots, n$
Phân rã Crout	$u_{ii} = 1, i = 1, 2, \dots, n$
Phân rã Cholesky	$L = U^T$

Bảng 3.2: Bảng liệt kê một số phương pháp phân rã LU phổ biến.

Sau khi ma trận A được phân rã, việc giải hệ phương trình $Ax = b$ được thực hiện dễ dàng tương tự như VÍ DỤ 3.1.2. Chúng ta viết hệ phương trình dưới dạng $LUX = b$. Đặt $y = UX$ ta có

$$Ly = b.$$

Bằng cách thay thế thuận ta sẽ tìm được y . Từ

$$\mathbf{U}\mathbf{x} = \mathbf{y}$$

ta sẽ tính được x bằng cách thay thế ngược.

Phương pháp phân rã LU hiệu quả hơn phương pháp khử Gauss ở điểm là ma trận \mathbf{A} chỉ cần phân rã một lần và sau đó ta có thể giải hệ phương trình với mọi vectơ hằng số \mathbf{b} tùy ý. Khối lượng của việc tính toán khi thay thế này rất nhỏ vì việc tính toán trong quá trình thay thế thuận, nghịch tồn rất ít thời gian so với việc phân rã.

3.3.2 PHƯƠNG PHÁP PHÂN RÃ DOOLITTLE

Pha phân rã. Phân rã Doolittle có mối liên hệ chặt chẽ với phương pháp khử Gauss. Để minh họa cho mối liên hệ này, ta xét ma trận \mathbf{A} có cỡ 3×3 và giả sử tồn tại các ma trận tam giác

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix} \quad \mathbf{U} = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

thoả mãn $\mathbf{A} = \mathbf{LU}$. Thực hiện tính toán về phải, ta có

$$\mathbf{A} = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ u_{11}l_{21} & u_{12}l_{21} + u_{22} & u_{13}l_{21} + u_{23} \\ u_{11}l_{31} & u_{12}l_{31} + u_{22}l_{32} & u_{13}l_{31} + u_{23}l_{32} + u_{33} \end{bmatrix} \quad (3.15)$$

Ta áp dụng pha khử trong phương pháp khử Gauss cho (3.15). Bước khử đầu tiên, ta chọn hàng thứ nhất làm hàng xoay và thực hiện các phép biến đổi

$$\text{hàng } 2 := \text{hàng } 2 - l_{21} \times \text{hàng } 1 \text{ (khử } a_{21})$$

$$\text{hàng } 3 := \text{hàng } 3 - l_{31} \times \text{hàng } 1 \text{ (khử } a_{31}).$$

Kết quả thu được là

$$\mathbf{A}' = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & u_{22}l_{32} & u_{23}l_{32} + u_{33} \end{bmatrix}.$$

Tiếp theo, ta chọn hàng thứ hai làm hàng xoay và thực hiện biến đổi

$$\text{hàng } 3 := \text{hàng } 3 - l_{32} \times \text{hàng } 2 \text{ (khử } a_{32})$$

ta được kết quả

$$\mathbf{A}'' = \mathbf{U} = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}.$$

Từ minh họa trên, ta thấy phân rã Doolittle có hai đặc điểm quan trọng sau đây:

- Ma trận \mathbf{U} là ma trận tam giác trên được tính toán từ pha khử Gauss;
- Các phần tử của ma trận \mathbf{L} (trừ các phần tử trên đường chéo chính) là bội số của hàng xoay trong quá trình khử Gauss. l_{ij} là bội số để khử a_{ij} .

Để tiện trong quá trình tính toán, các bội số sẽ được lưu l_{ij} sẽ được thay thế vị trí phần tử bị khử a_{ij} . Các phần tử trên đường chéo chính của \mathbf{L} không cần lưu trữ vì chúng bằng một. Dạng cuối cùng cả ma trận hệ số là dạng kết hợp của \mathbf{L} và \mathbf{U} như sau

$$[\mathbf{L} \setminus \mathbf{U}] = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ l_{21} & u_{22} & u_{23} \\ l_{31} & l_{32} & u_{33} \end{bmatrix}. \quad (3.16)$$

Phương pháp phân rã Doolittle để giải hệ phương trình tuyến tính được minh họa bằng sơ đồ khối trong Hình 3.3.

Thuật toán cho phân rã Doolittle được thực hiện tương tự như quá trình khử Gauss trong gauss ngoại trừ việc mỗi bội số λ bây giờ được lưu trữ trong phần phía tam giác dưới của ma trận \mathbf{A} .

 LUdec.

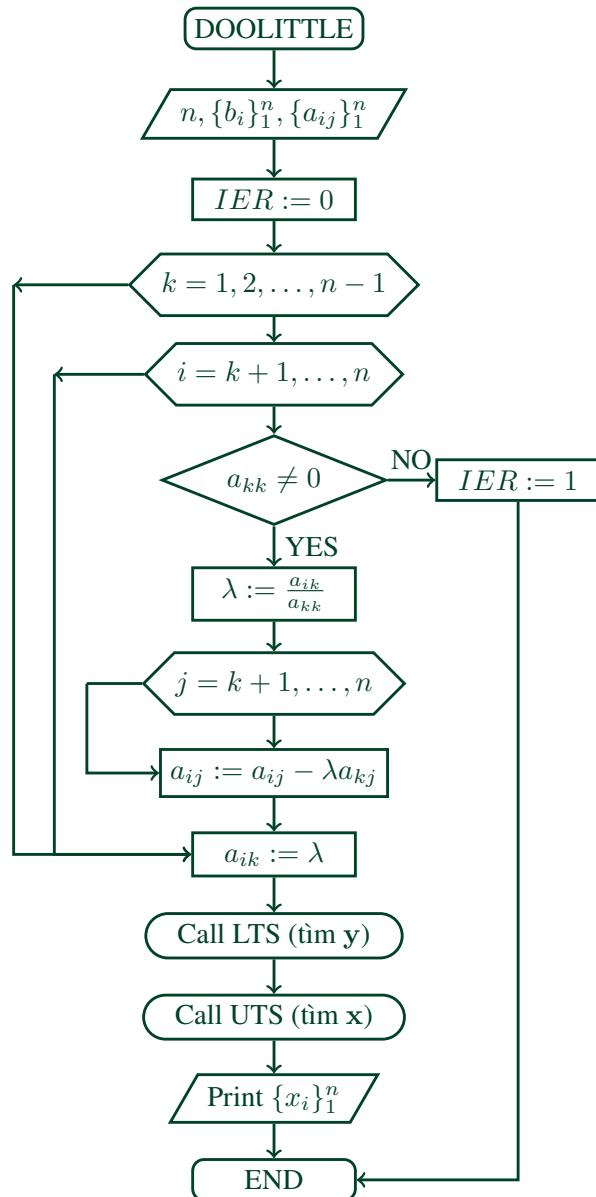
Trong phiên bản phân rã LU này, ma trận hệ số \mathbf{A} bị triệt tiêu và được ghi đè bằng các phân rã dạng $[\mathbf{L} \setminus \mathbf{U}]$.

```

function A=LUdec(A)
% LU decomposition of matrix A; returns A=[L\U].
% USAGE:A=LUdec(A)

n = size(A,1);
for k=1:n-1
    for i=k+1:n
        if A(i,k) ~= 0.0
            lambda = A(i,k)/A(k,k);
            A(i,k+1:n) = A(i,k+1:n) - lambda*A(k,k+1:n);
        end
    end
end

```



Hình 3.3: Sơ đồ khối giải hệ phương trình tuyến tính bằng phương pháp phân rã Doolittle (LU)

```

A(i,k) = lambda;
end
end
end
  
```

Pha giải.. Bây giờ ta xét quá trình giải $Lx = b$ bằng cách thay thế thuận. Dạng vô hướng của hệ phương trình là (nhắc lại $l_{ii} = 1$)

$$\begin{aligned}
 y_1 &= b_1 \\
 L_{21}y_1 + y_2 &= b_2 \\
 &\vdots \\
 L_{k1}y_1 + L_{k2}y_2 + \cdots + L_{k,k-1}y_{k-1} + y_k &= b_k \\
 &\vdots
 \end{aligned}$$

Giải phương trình thứ k theo y_k ta có

$$y_k = b_k - \sum_{j=1}^{k-1} L_{kj}y_j, \quad k = 2, 3, \dots, n. \quad (3.17)$$

Đặt y ghi đè lên b ta thu được thuật toán thay thế thuận sau

```

for k=2:n
    y(k) = b(k) - A(k,1:k-1)*y(1:k-1);
end

```

Pha thay thế ngược để giải $\mathbf{U}\mathbf{x} = \mathbf{y}$ được thực hiện tương tự như phương pháp khử Gauss.

 LUsol.

Chương trình này thực hiện pha giải (thay thế thuận và thay thế ngược). Giả thiết rằng ma trận hệ số ban đầu đã được phân rã dưới dạng $\mathbf{A} = [\mathbf{L} \setminus \mathbf{U}]$. Trong quá trình thay thế thuận \mathbf{b} sẽ được ghi đè bởi \mathbf{y} và trong quá trình thay thế ngược thì \mathbf{y} được ghi đè bởi \mathbf{x} .

```

function x=LUsol(A,b)
% Solves L\U\b = x, where A contains both L and U;
% that is, A has the form[L\U].
% USAGE:x=LUsol(A,b)
if size(b,2) > 1; b = b'; end
n = length(b);
for k=2:n
    b(k) = b(k) - A(k,1:k-1)*b(1:k-1);
end
for k=n:-1:1
    b(k) = (b(k) - A(k,k+1:n)*b(k+1:n))/A(k,k);
end
x=b;

```

VÍ DỤ 3.3.1. Sử dụng phương pháp phân rã Doolittle để giải hệ phương trình $\mathbf{Ax} = \mathbf{b}$, với

$$\mathbf{A} = \begin{bmatrix} 1 & 4 & 1 \\ 1 & 6 & -1 \\ 2 & -1 & 2 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 7 \\ 13 \\ 5 \end{bmatrix}.$$

GIẢI. Ta phân rã ma trận A bởi phép khử Gauss. Chọn hàng thứ nhất làm hàng xoay và thực hiện các phép biến đổi

$$\text{hàng 2} := \text{hàng 2} - 1 \times \text{hàng 1} \text{ (khử } a_{21})$$

$$\text{hàng 3} := \text{hàng 3} - 2 \times \text{hàng 1} \text{ (khử } a_{31}).$$

Lưu các bội số $l_{21} = 1, l_{31} = 2$ vào vị trí a_{21}, a_{31} ta có

$$\mathbf{A}' = \begin{bmatrix} 1 & 4 & 1 \\ 1 & 2 & -2 \\ 2 & -9 & 0 \end{bmatrix}.$$

Chọn hàng thứ hai làm hàng xoay và thực hiện phép biến đổi (chỉ áp dụng cho hai cột sau)

$$\text{hàng 3} := \text{hàng 3} - (-4.5) \times \text{hàng 2} \text{ (khử } a_{32}).$$

Lưu các bội số $l_{32} = -4.5$ vào vị trí a_{32} ta có

$$\mathbf{A}'' = [\mathbf{L} \setminus \mathbf{U}] = \begin{bmatrix} 1 & 4 & 1 \\ 1 & 2 & -2 \\ 2 & -4.5 & -9 \end{bmatrix}.$$

Phân rã LU được thực hiện xong với

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 2 & -4.5 & 1 \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} 1 & 4 & 1 \\ 0 & 2 & -2 \\ 0 & 0 & -9 \end{bmatrix}.$$

Tiếp theo ta giải phương trình $\mathbf{Ly} = \mathbf{b}$ bằng phép thế thuận. Ma trận hệ số mở rộng của phương trình này là

$$[\mathbf{L} | \mathbf{b}] = \left[\begin{array}{ccc|c} 1 & 0 & 0 & 7 \\ 1 & 1 & 0 & 13 \\ 2 & -4.5 & 1 & 5 \end{array} \right].$$

Nghiệm của hệ được tính bởi

$$y_1 = 7,$$

$$y_2 = 13 - y_1 = 6,$$

$$y_3 = 5 - 2y_1 + 4, 5y_2 = 18.$$

Cuối cùng nghiệm của hệ $\mathbf{U}\mathbf{x} = \mathbf{y}$ hoặc

$$[\mathbf{U}|\mathbf{y}] = \left[\begin{array}{ccc|c} 1 & 4 & 1 & 7 \\ 0 & 2 & -2 & 6 \\ 0 & 0 & -9 & 18 \end{array} \right].$$

được tính bằng cách thế ngược. Ta có

$$x_3 = \frac{18}{-9} = -2,$$

$$x_2 = \frac{6 + 2x_3}{2} = 1,$$

$$x_1 = 7 - 4x_2 - x_3 = 5.$$

□

VÍ DỤ 3.3.2. Giải $\mathbf{A}\mathbf{X} = \mathbf{B}$ bằng phương pháp phân rã Doolittle và tính $|\mathbf{A}|$, với

$$\mathbf{A} = \begin{bmatrix} 3 & -1 & 4 \\ -2 & 0 & 5 \\ 7 & 2 & -2 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 6 & -4 \\ 3 & 2 \\ 7 & -5 \end{bmatrix}.$$

GIẢI. Trong chương trình sau, ma trận hệ số \mathbf{A} trước tiên được phân rã bằng hàm LUdec. Sau đó, hàm LUsol sẽ được sử dụng để tính nghiệm với mỗi vectơ một lần.

```
% Example 3.3.2 (Doolittle's decomposition)
A=[3-14;-205;72-2];
B=[6-4;32;7-5];
A = LUdec(A);
det = prod(diag(A))
for i=1:size(B,2)
    X(:,i) = LUsol(A,B(:,i));
end
X
```

Kết quả là

```

>> det =
-77
X=
1.0000 -1.0000
1.0000 1.0000
1.0000 0.0000

```

□

3.3.3 PHƯƠNG PHÁP PHÂN RÃ CHOLESKY

Phân rã Cholesky $\mathbf{A} = \mathbf{LL}^T$ có hai hạn chế sau

- Do tích ma trận \mathbf{LL}^T là đối xứng nên phân rã Cholesky yêu cầu ma trận \mathbf{A} là ma trận đối xứng.
- Quá trình phân rã yêu cầu tính căn bậc hai của tổ hợp tuyến tính các phần tử của ma trận \mathbf{A} . Để tránh được việc lấy căn bậc hai của số âm thì ma trận \mathbf{A} là ma trận xác định dương.

Mặc dù số lượng các phép biến đổi của phương pháp phân rã Cholesky cũng tương đương với các phương pháp khác nhưng phương pháp phân rã Cholesky không phải là một phương pháp phổ biến để giải hệ phương trình tuyến tính vì những hạn chế nêu ở trên. Chúng ta nghiên cứu phương pháp phân rã Cholesky vì nó có ứng dụng ở rất nhiều lĩnh vực khác (chẳng hạn việc biến đổi của các bài toán giá trị riêng).

Chúng ta xét phân rã Cholesky

$$\mathbf{A} = \mathbf{LL}^T \quad (3.18)$$

của ma trận cỡ 3×3

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} l_{11} & l_{12} & l_{13} \\ 0 & l_{22} & l_{23} \\ 0 & 0 & l_{33} \end{bmatrix}.$$

Thực hiện nhân hai ma trận ở về phải ta có

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} l_{11}^2 & l_{11}l_{21} & l_{11}l_{31} \\ l_{11}l_{21} & l_{21}^2 + l_{22}^2 & l_{21}l_{31} + l_{22}l_{32} \\ l_{11}l_{31} & l_{21}l_{31} + l_{22}l_{32} & l_{31}^2 + l_{32}^2 + l_{33}^2 \end{bmatrix}. \quad (3.19)$$

Ta thấy về phải là ma trận đối xứng nên (3.19) tương đương với sáu phương trình với sáu ẩn là các phần tử của L . Ta có thể chọn thứ tự giải các phương trình này để mỗi phương trình chỉ có một nghiệm chưa biết.

Ta xét các phần tử ở tam giác dưới của (3.19) (ta cũng có thể xét các phần tử ở vị trí tam giác trên). Bằng cách cho các phần tử tương ứng ở các cột thứ nhất bằng nhau, ta có thể tính l_{11}, l_{21} và l_{31} như sau

$$\begin{aligned} a_{11} &= l_{11}^2 & l_{11} &= \sqrt{a_{11}} \\ a_{21} &= l_{11}l_{21} \Rightarrow l_{21} &= a_{21}/l_{11} \\ a_{31} &= l_{11}l_{31} & l_{31} &= a_{31}/l_{11} \end{aligned}$$

Ở cột thứ hai, bắt đầu từ hàng thứ hai ta sẽ có l_{22}, l_{32}

$$\begin{aligned} a_{22} &= l_{21}^2 + l_{22}^2 \Rightarrow l_{22} &= \sqrt{a_{22} - l_{21}^2} \\ a_{32} &= l_{21}l_{31} + l_{22}l_{32} \quad l_{21} &= (a_{32} - l_{21}l_{31})/l_{22} \end{aligned}.$$

Cuối cùng, từ cột thứ ba, hàng thứ ba ta có l_{33}

$$a_{33} = l_{31}^2 + l_{32}^2 + l_{33}^2 \Rightarrow l_{33} = \sqrt{a_{33} - l_{31}^2 - l_{32}^2}.$$

Ta có thể tổng quát hoá cho trường hợp ma trận cỡ $n \times n$. Ta thấy các phần tử của ở vị trí tam giác dưới của ma trận LL^T có dạng

$$(LL^T)_{ij} = l_{i1}l_{j1} + l_{i2}l_{j2} + \dots + l_{ij}l_{jj} = \sum_{k=1}^j l_{ik}l_{jk}, i \geq j.$$

Do (3.19) nên các phần tử của A thỏa mãn

$$a_{ij} = \sum_{k=1}^j l_{ik}l_{jk}, i = j, j+1, \dots, n; j = 1, 2, \dots, n. \quad (3.20)$$

Các chỉ số thể hiện ta tính toán các phần tử ở vị trí tam giác dưới của ma trận A . Bắt đầu với cột thứ nhất ($j = 1$), từ công thức (3.20), ta có

$$l_{11} = \sqrt{a_{11}}, \quad l_{i1} = a_{i1}/l_{11}, i = 2, 3, \dots, n. \quad (3.21)$$

Quá trình tính toán tiếp tục với các cột khác. Giả sử ta đã thực hiện tính toán được các cột $1, 2, \dots, j-1$, tức là đã tính được $l_{ik}, i \geq k, k = 1, 2, \dots, j-1$, ta xét cột thứ j để tính các phần tử $l_{ij}, i \geq j$. Ta viết công thức (3.20) dưới dạng

$$a_{ij} = \sum_{k=1}^{j-1} l_{ik}l_{jk} + l_{ij}l_{jj}.$$

Với $i = j$ (phần tử trên đường chéo), ta có

$$l_{jj} = \sqrt{a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2}. \quad (3.22)$$

Với các phần tử còn lại ta tính theo công thức

$$l_{ij} = \left(a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk} \right) / l_{jj}, \quad j = 2, 3, \dots, n-1, i = j+1, \dots, n. \quad (3.23)$$

Phương pháp phân rã Cholesky được minh họa bằng sơ đồ khối cho trong hình sau.

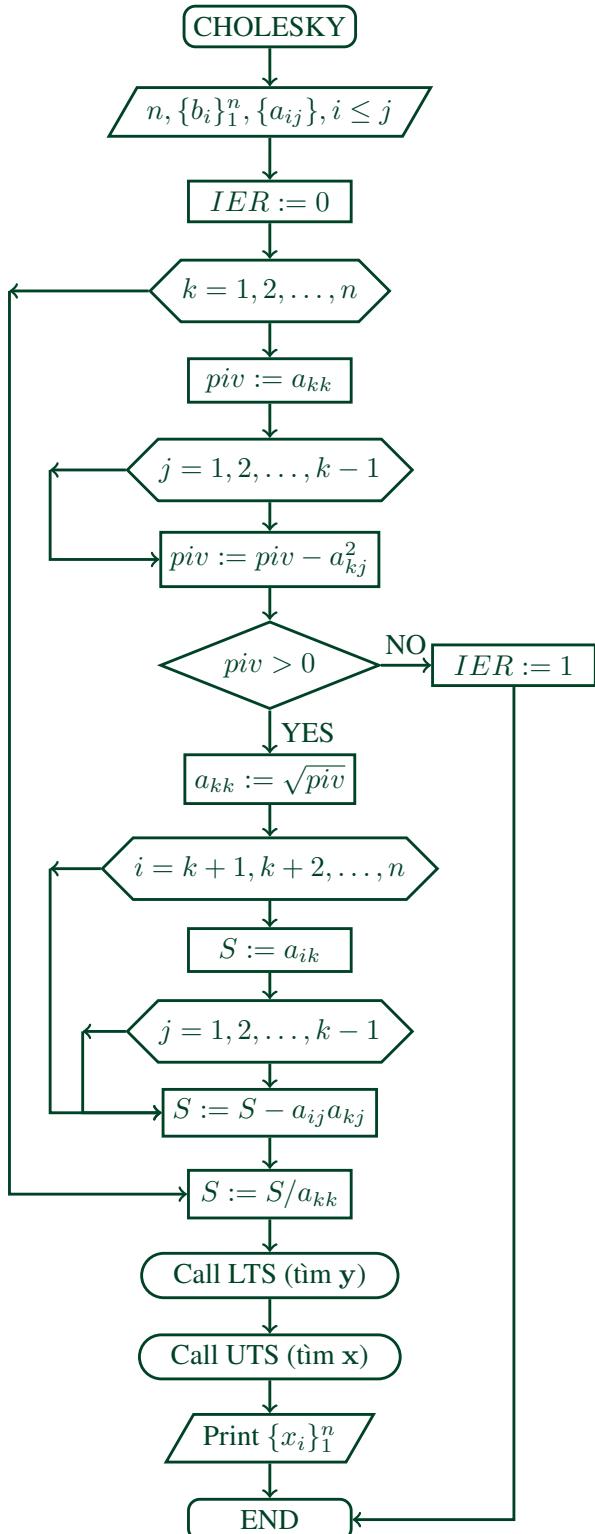
choleski.

Ta thấy a_{ij} trong công thức (3.22) và (3.23) chỉ dùng để tính l_{ij} . Vì khi l_{ij} được tính thì a_{ij} không cần nữa nên ta có thể lưu các phần tử của L vào vị trí tam giác dưới của ma trận A. các phần tử nằm trên đường chéo chính của ma trận A được giữ nguyên. Khi kết thúc phân rã, L được trích ra từ MATLAB bằng lệnh `tril(A)`. Nếu trong quá trình phân rã mà gặp một số âm l_{jj}^2 thì sẽ có một thông báo lỗi được in và kết thúc chương trình.

```

function L=choleski (A)
% Computes L in Choleski's decomposition A=LL'.
% USAGE:L=choleski(A)
n = size(A,1);
for j=1:n
    temp = A(j,j) - dot(A(j,1:j-1),A(j,1:j-1));
    if temp < 0.0
        error('Matrix is not positive definite')
    end
    A(j,j) = sqrt(temp);
    for i=j+1:n
        A(i,j)=(A(i,j) - dot(A(i,1:j-1),A(j,1:j-1)))/A(j,j);
    end
end
L = tril(A)

```



**Hình 3.4: Sơ đồ khối
giải hệ phương trình
tuyến tính bằng
phương pháp phân
rã Cholesky**

VÍ DỤ 3.3.3. Phân rã Cholesky cho ma trận

$$\mathbf{A} = \begin{bmatrix} 4 & -2 & 2 \\ -2 & 2 & -4 \\ 2 & -4 & 11 \end{bmatrix}.$$

GIẢI. Ma trận \mathbf{A} là ma trận đối xứng, nếu ma trận \mathbf{A} xác định dương thì ta có thể phân rã Cholesky. Tuy nhiên, việc kiểm tra tính xác định dương không cần thiết vì nếu không phân rã được thì ma trận không xác định dương.

Thay ma trận \mathbf{A} vào phương trình (3.19), ta có

$$\begin{bmatrix} 4 & -2 & 2 \\ -2 & 2 & -4 \\ 2 & -4 & 11 \end{bmatrix} = \begin{bmatrix} l_{11}^2 & l_{11}l_{21} & l_{11}l_{31} \\ l_{11}l_{21} & l_{21}^2 + l_{22}^2 & l_{21}l_{31} + l_{22}l_{32} \\ l_{11}l_{31} & l_{21}l_{31} + l_{22}l_{32} & l_{31}^2 + l_{32}^2 + l_{33}^2 \end{bmatrix}.$$

Giải các phương trình tương ứng với vị trí tam giác dưới ta có

$$l_{11} = \sqrt{4} = 2$$

$$l_{21} = -2/l_{11} = -1$$

$$l_{31} = 2/l_{11} = 1$$

$$l_{22} = \sqrt{2 - l_{21}^2} = 1$$

$$l_{32} = \frac{-4 - l_{21}l_{31}}{l_{22}} = -3$$

$$l_{33} = \sqrt{11 - l_{31}^2 - l_{32}^2}.$$

Nên suy ra

$$\mathbf{L} = \begin{bmatrix} 2 & 0 & 0 \\ -1 & 1 & 0 \\ 1 & -3 & 1 \end{bmatrix}.$$

□

VÍ DỤ 3.3.4. Kiểm tra chương trình Choleski bằng cách phân rã

$$\mathbf{A} = \begin{bmatrix} 1.44 & -0.36 & 5.52 & 0.00 \\ -0.36 & 10.33 & -7.78 & 0.00 \\ 5.52 & -7.78 & 28.40 & 9.00 \\ 0.00 & 0.00 & 9.00 & 61.00 \end{bmatrix}$$

GIẢI.

```
>> % Example 3.3.4 (Choleski decomposition
A = [1.44 -0.36 5.52 0.00;
      -0.36 10.33 -7.78 0.00;
```

```

5.52 -7.78 28.40 9.00;
0.00 0.00 9.00 61.00];
L = choleski(A)
Check = L*L' % Verify the result

L =
1.2000e+000      0      0      0
-3.0000e-001 3.2000e+000      0      0
4.6000e+000 -2.0000e+000 1.8000e+000 0
0      0 5.0000e+000 6.0000e+000

L =
1.2000e+000      0      0      0
-3.0000e-001 3.2000e+000      0      0
4.6000e+000 -2.0000e+000 1.8000e+000 0
0      0 5.0000e+000 6.0000e+000

Check =
1.4400e+000 -3.6000e-001 5.5200e+000 0
-3.6000e-001 1.0330e+001 -7.7800e+000 0
5.5200e+000 -7.7800e+000 2.8400e+001 9.0000e+000
0      0 9.0000e+000 6.1000e+001

```

□

3.4 MA TRẬN DẢI VÀ MA TRẬN ĐỐI XỨNG

Các bài toán kỹ thuật thường dẫn đến ma trận hệ số có tính chất thưa, có nghĩa là hầu hết các phần tử bằng không. Nếu các phần tử khác không của một ma trận co cụm lại thành dải đường chéo thì ma trận này được gọi là *ma trận dải*. Chẳng hạn ta có ma

trận dải sau

$$\mathbf{A} = \begin{bmatrix} X & X & 0 & 0 & 0 \\ X & X & X & 0 & 0 \\ 0 & X & X & X & 0 \\ 0 & 0 & X & X & X \\ 0 & 0 & 0 & X & X \end{bmatrix}$$

với X là các phần tử khác không của ma trận tạo thành một dải đường chéo. Các phần tử nằm ngoài dải này bằng không. Ta thấy hầu như mỗi hàng của ma trận có ba phần tử khác không, các phần tử này tạo thành một dải gồm ba đường chéo nên ma trận \mathbf{A} còn được gọi là ma trận *ba đường chéo*.

Nếu một ma trận dải được phân rã dưới dạng $\mathbf{A} = \mathbf{LU}$ thì cả \mathbf{L} và \mathbf{U} đều có cấu trúc tương tự như \mathbf{A} . Chẳng hạn, nếu ma trận \mathbf{A} phía trên được phân rã \mathbf{LU} thì ta có

$$\mathbf{L} = \begin{bmatrix} X & 0 & 0 & 0 & 0 \\ X & X & 0 & 0 & 0 \\ 0 & X & X & 0 & 0 \\ 0 & 0 & X & X & 0 \\ 0 & 0 & 0 & X & X \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} X & X & 0 & 0 & 0 \\ 0 & X & X & 0 & 0 \\ 0 & 0 & X & X & 0 \\ 0 & 0 & 0 & X & X \\ 0 & 0 & 0 & 0 & X \end{bmatrix}.$$

Trong việc giải hệ phương trình tuyến tính, cấu trúc dải của ma trận hệ số có thể được khai thác để tiết kiệm bộ nhớ và thời gian tính toán. Nếu ma trận hệ số có tính đối xứng thì việc lưu trữ mà tính toán sẽ tiết kiệm hơn. Một số phương pháp áp dụng cho ma trận hệ số dải hoặc ma trận hệ số dải có tính đối xứng sẽ được trình bày sau đây.

3.4.1 MA TRẬN BA ĐƯỜNG CHÉO

Tìm nghiệm hệ phương trình $\mathbf{Ax} = \mathbf{b}$ bằng phương pháp phân rã Doolittle, với \mathbf{A} là ma trận ba đường chéo cỡ $n \times n$

$$\mathbf{A} = \begin{bmatrix} d_1 & e_1 & 0 & 0 & \cdots & 0 \\ c_1 & d_2 & e_2 & 0 & \cdots & 0 \\ 0 & c_2 & d_3 & e_3 & \cdots & 0 \\ 0 & 0 & c_3 & d_4 & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & c_{n-1} & d_n \end{bmatrix}.$$

Như ký hiệu trong ma trận, ta lưu các phần tử khác không của A trong ba vectơ

$$\mathbf{c} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{bmatrix}, \quad \mathbf{d} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix}, \quad \mathbf{e} = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_{n-1} \end{bmatrix}.$$

Kết quả của việc lưu trữ như trên là tiết kiệm đáng kể. Chẳng hạn, với ma trận ba đường chéo cỡ 100×100 có 10000 phần tử nhưng ta chỉ cần lưu trữ trong $99 + 100 + 99 = 298$ ô nhớ. Tiết kiệm được 33 lần.

Trước tiên ta thực hiện phân rã LU cho ma trận A. Giả sử ta cần khử c_{k-1} ở hàng k. Ta chọn hàng $(k-1)$ làm hàng xoay và thực hiện phép biến đổi

$$\text{hàng } k := \text{hàng } k - \frac{c_{k-1}}{d_{k-1}} \times \text{hàng } (k-1), k = 2, 3, \dots, n.$$

Suy ra

$$d_k := d_k - \frac{c_{k-1}}{d_{k-1}} e_{k-1}, \quad (3.24)$$

với phần tử e_k không thay đổi. Để biểu diễn ma trận A dưới dạng $[L \setminus U]$, ta lưu bội số $\lambda = c_{k-1}/d_{k-1}$ tại vị trí của c_{k-1} .

$$c_{k-1} := \frac{c_{k-1}}{d_{k-1}}. \quad (3.25)$$

Quá trình sử dụng phương pháp phân rã Doolittle ma trận ba đường chéo A về dạng $[L \setminus U]$ như trên được minh họa bằng sơ đồ khối trong Hình 3.5.

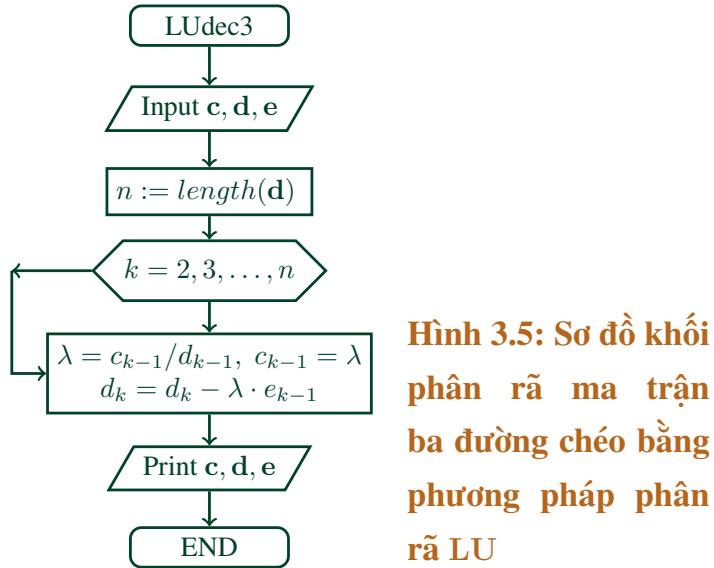
LUdec3.

Hàm LUdec3 chứa code của quá trình phân rã ma trận ba đường chéo. Các vectơ gốc c và d được phân rã và được ghi đè bởi các vectơ của ma trận phân rã.

```

function [c,d,e] = LUdec3(c,d,e)
    % LU decomposition of tridiagonal matrix A=[c\d\c].
    % USAGE: [c,d,e] = LUdec3(c,d,e)
    n = length(d);
    for k=2:n
        lambda = c(k-1)/d(k-1);
        d(k) = d(k) - lambda*e(k-1);
        c(k-1) = lambda;
    end

```



Hình 3.5: Sơ đồ khối phân rã ma trận ba đường chéo bằng phương pháp phân rã LU

Tiếp theo ta thực hiện pha giải, tức là trước tiên ta giải hệ phương trình $Ly = b$ rồi sau đó giải hệ $Ux = y$. Hệ $Ly = b$ có thể được biểu diễn dưới dạng ma trận hệ số mở rộng

$$[\mathbf{L}|\mathbf{b}] = \left[\begin{array}{cccccc|c} 1 & 0 & 0 & 0 & \dots & 0 & b_1 \\ c_1 & 1 & 0 & 0 & \dots & 0 & b_2 \\ 0 & c_2 & 1 & 0 & \dots & 0 & b_3 \\ 0 & 0 & c_3 & 1 & \dots & 0 & b_4 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & c_{n-1} & 1 & b_n \end{array} \right]$$

Chú ý rằng hệ số c_i không phải là hệ số ban đầu của ma trận mà mà bội số của quá trình phân rã tính bởi công thức (3.25). Sử dụng cách thay thế thuận ta có

$$y_1 = b_1;$$

$$y_k = b_k - c_{k-1} \times y_{k-1}, k = 2, 3, \dots, n.$$

Ma trận hệ số mở rộng của hệ phương trình $Ux = y$ có dạng

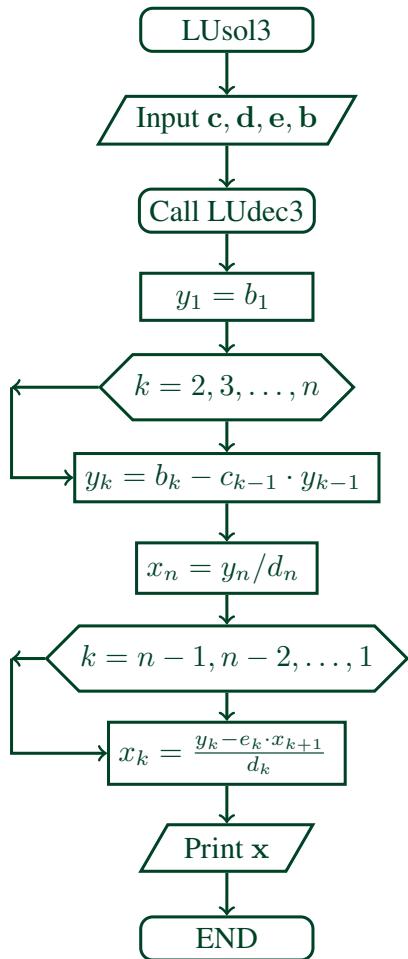
$$[\mathbf{U}|y] = \left[\begin{array}{cccccc|c} d_1 & e_1 & 0 & \cdots & 0 & 0 & y_1 \\ 0 & d_2 & e_2 & \cdots & 0 & 0 & y_2 \\ 0 & 0 & d_1 & \cdots & 0 & 0 & y_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & d_{n-1} & e_{n-1} & y_{n-1} \\ 0 & 0 & 0 & \cdots & 0 & d_n & y_n \end{array} \right].$$

Chú ý các hệ số d_i không phải hệ số ban đầu mà được tính bởi công thức (3.24), các hệ số e_i là hệ số ban đầu. Sử dụng cách thay thế ngược ta có

$$x_n = \frac{y_n}{d_n};$$

$$x_k = \frac{y_k - e_{k-1}x_{k+1}}{d_k}, k = n-1, n-2, \dots, 1.$$

Pha giải của phương pháp phân rã Doolittle giải hệ phương trình tuyến tính ba đường chéo được minh họa trong Hình 3.6.



Hình 3.6: Sơ đồ khôi giải hệ phương trình tuyến tính ba đường chéo bằng phương pháp phân rã LU

LUsol3.

Hàm sau đây tính toán nghiệm của hệ ba đường chéo. Để tiết kiệm ô nhớ thì vectơ y được ghi đè lên vectơ hằng b khi giải Ly = b. Tương tự vectơ x sẽ ghi đè lên y khi giải Ux = y.

```

function x=LUsol3(c,d,e,b)
% Solves A*x=b where A=[c\d\|e] is the LU
% decomposition of the original tridiagonal A.
% USAGE:x=LUsol3(c,d,e,b)
  
```

```

n = length(d);
for k=2:n %Forward substitution
    b(k) = b(k) - c(k-1)*b(k-1);
end
b(n) = b(n)/d(n); % Back substitution
for k=n-1:-1:1
    b(k) = (b(k) - e(k)*b(k+1))/d(k);
end
x=b;

```

3.4.2 MA TRẬN ĐỐI XỨNG

Ma trận đối xứng là ma trận mà các phần tử nằm ở vị trí đối xứng với nhau qua đường chéo chính có giá trị bằng nhau. Cũng như ma trận dải, ma trận đối xứng thường phát sinh từ những bài toán kỹ thuật. Do đó, ta cần phải nghiên cứu các tính chất đặc biệt của nó để xây dựng thuật toán để tính toán một cách hiệu quả.

Nếu ma trận A là một ma trận đối xứng thì có thể được phân rã LU dưới dạng

$$A = LU = LDL^T \quad (3.26)$$

với ma trận D là ma trận đường chéo. Một ví dụ là phân rã Cholesky đã thảo luận ở mục trước (trong trường hợp này $D = I$). Với phân rã Doolittle, ta có

$$U = DL^T = \begin{bmatrix} D_1 & 0 & 0 & \cdots & 0 \\ 0 & D_2 & 0 & \cdots & 0 \\ 0 & 0 & D_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & D_n \end{bmatrix} \begin{bmatrix} 1 & l_{21} & l_{31} & \cdots & l_{n1} \\ 0 & 1 & l_{32} & \cdots & l_{n2} \\ 0 & 0 & 1 & \cdots & l_{n3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

nên

$$U = \begin{bmatrix} D_1 & D_1l_{21} & D_1l_{31} & \cdots & D_1l_{n1} \\ 0 & D_2 & D_2l_{32} & \cdots & D_2l_{n2} \\ 0 & 0 & D_3 & \cdots & D_3l_{n3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & D_n \end{bmatrix}.$$

Ta thấy rằng trong quá trình phân rã một ma trận đối xứng thì ta có thể xác định các ma trận D và L từ ma trận U . Do đó, để tiết kiệm bộ nhớ lưu trữ và tiện tính toán thì

một ma trận đối xứng chỉ cần lưu dưới dạng

$$\mathbf{U}^* = \begin{bmatrix} D_1 & l_{21} & l_{31} & \cdots & l_{n1} \\ 0 & D_2 & l_{32} & \cdots & l_{n2} \\ 0 & 0 & D_3 & \cdots & l_{n3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & D_n \end{bmatrix}. \quad (3.27)$$

Với ma trận \mathbf{U} được xác định bởi công thức $u_{ij} = D_i l_{ji}$. Điều này dẫn đến sơ đồ tính toán hiệu quả hơn cho pha giải. Do đó, chúng ta sẽ áp dụng cho ma trận dải có tính đối xứng trong mục sau.

VÍ DỤ 3.4.1. Bằng cách sử dụng phương pháp khử Gauss, ma trận đối xứng \mathbf{A} được chuyển về dạng ma trận tam giác trên

$$\mathbf{U} = \begin{bmatrix} 4 & -2 & 1 & 0 \\ 0 & 3 & -3/2 & 1 \\ 0 & 0 & 3 & -3/2 \\ 0 & 0 & 0 & 35/12 \end{bmatrix}.$$

Hãy xác định ma trận \mathbf{A} .

GIẢI. Trước tiên ta tìm ma trận \mathbf{L} trong phân rã $\mathbf{A} = \mathbf{LU}$. Lấy các hàng của ma trận \mathbf{U} chia cho phần tử nằm trên đường chéo của hàng đó ta có

$$\mathbf{L}^T = \begin{bmatrix} 1 & -1/2 & 1/4 & 0 \\ 0 & 1 & -1/2 & 1/3 \\ 0 & 0 & 1 & -1/2 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Do đó, ma trận $\mathbf{A} = \mathbf{LU}$ được xác định bởi

$$\begin{aligned} \mathbf{A} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1/2 & 1 & 0 & 0 \\ 1/4 & -1/2 & 1 & 0 \\ 0 & 1/3 & -1/2 & 1 \end{bmatrix} \begin{bmatrix} 4 & -2 & 1 & 0 \\ 0 & 3 & -3/2 & 1 \\ 0 & 0 & 3 & -3/2 \\ 0 & 0 & 0 & 35/12 \end{bmatrix} \\ &= \begin{bmatrix} 4 & -2 & 1 & 0 \\ -2 & 4 & -2 & 1 \\ 1 & -2 & 4 & -2 \\ 0 & 1 & -2 & 4 \end{bmatrix}. \end{aligned}$$

□

VÍ DỤ 3.4.2. Xác định ma trận L và ma trận D trong phép phân rã Doolittle ma trận $A = LDL^T$, với ma trận đối xứng

$$A = \begin{bmatrix} 3 & -3 & 3 \\ -3 & 5 & 1 \\ 3 & 1 & 10 \end{bmatrix}.$$

GIẢI. Sử dụng phép khử Gauss, chúng ta lưu các bội số trong quá trình khử lên vị trí tam giác trên của ma trận A dưới dạng ma trận U^* trong (3.27).

Trước tiên, ta khử các phần tử a_{21}, a_{31} bằng cách thực hiện các phép biến đổi

$$\text{hàng } 2 := \text{hàng } 2 - (-1) \times \text{hàng } 1$$

$$\text{hàng } 3 := \text{hàng } 3 - (1) \times \text{hàng } 1.$$

Lưu các bội số (-1 và 1) vào vị trí của a_{12}, a_{13} ta có

$$A' = \begin{bmatrix} 3 & -1 & 1 \\ 0 & 2 & 4 \\ 0 & 4 & 7 \end{bmatrix}.$$

Tiếp theo thực hiện phép biến đổi

$$\text{hàng } 3 := \text{hàng } 3 - 2 \times \text{hàng } 2.$$

và lưu bội số (2) vào vị trí a_{23} ta có

$$A'' = [\mathbf{0} \setminus \mathbf{D} \setminus \mathbf{L}] = \begin{bmatrix} 3 & -1 & 1 \\ 0 & 2 & 2 \\ 0 & 0 & -1 \end{bmatrix}.$$

Do đó

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 1 & 2 & 1 \end{bmatrix}, \quad \mathbf{D} = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & -1 \end{bmatrix}.$$

□

3.4.3 MA TRẬN DẢI ĐỐI XỨNG

Trong mục này chúng ta sẽ thảo luận về phương pháp giải hệ phương trình tuyến tính mà ma trận hệ số là ma trận năm đường chéo đối xứng. Ma trận hệ số năm đường chéo xuất hiện trong quá trình sai phân hoá để giải phương trình vi phân bậc bốn. Thông thường nó là ma trận đối xứng cỡ $n \times n$ có dạng

$$\mathbf{A} = \begin{bmatrix} d_1 & e_1 & f_1 & 0 & 0 & 0 & \cdots & 0 \\ e_1 & d_2 & e_2 & f_2 & 0 & 0 & \cdots & 0 \\ f_1 & e_2 & d_3 & e_3 & f_3 & 0 & \cdots & 0 \\ 0 & f_2 & e_3 & d_4 & e_4 & f_4 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & f_{n-4} & e_{n-3} & d_{n-2} & e_{n-2} & f_{n-2} \\ 0 & 0 & \cdots & 0 & f_{n-3} & e_{n-2} & d_{n-1} & e_{n-1} \\ 0 & 0 & 0 & \cdots & 0 & f_{n-2} & e_{n-1} & d_n \end{bmatrix}. \quad (3.28)$$

Để không lãng phí bộ nhớ, ta lưu trữ các phần tử khác không của ma trận trong ba vecto

$$\mathbf{d} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-2} \\ d_{n-1} \\ d_n \end{bmatrix}, \quad \mathbf{e} = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_{n-2} \\ e_{n-1} \end{bmatrix}, \quad \mathbf{f} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_{n-2} \end{bmatrix}.$$

Tiếp theo, chúng ta giải hệ phương trình $\mathbf{Ax} = \mathbf{b}$ bằng phương pháp phân rã Doolittle. Đầu tiên, chúng ta chuyển ma trận A về dạng ma trận tam giác trên bằng phương pháp khử Gauss. Giả sử ta đã có k hàng đầu tiên có dạng tam giác trên, ta chọn hàn k làm hàng xoay, khi đó ta có

$$\mathbf{A} = \left[\begin{array}{c|ccc|cccc} \ddots & \vdots \\ \cdots & 0 & d_k & e_k & f_k & 0 & 0 & 0 & \cdots \\ \cdots & 0 & e_k & d_{k+1} & e_{k+1} & f_{k+1} & 0 & 0 & \cdots \\ \cdots & 0 & f_k & e_{k+1} & d_{k+2} & e_{k+2} & f_{k+2} & 0 & \cdots \\ \cdots & 0 & 0 & f_{k+1} & e_{k+2} & d_{k+3} & e_{k+3} & f_{k+3} & \cdots \\ \vdots & \ddots \end{array} \right] \leftarrow \text{hàng xoay}$$

Các phần tử e_k, d_k phía dưới hàng xoay được khử bởi các phép biến đổi

$$\begin{aligned}\text{hàng } (k+1) &:= \text{hàng } (k+1) - (e_k/d_k) \times \text{hàng } (k) \\ \text{hàng } (k+2) &:= \text{hàng } (k+2) - (f_k/d_k) \times \text{hàng } (k)\end{aligned}$$

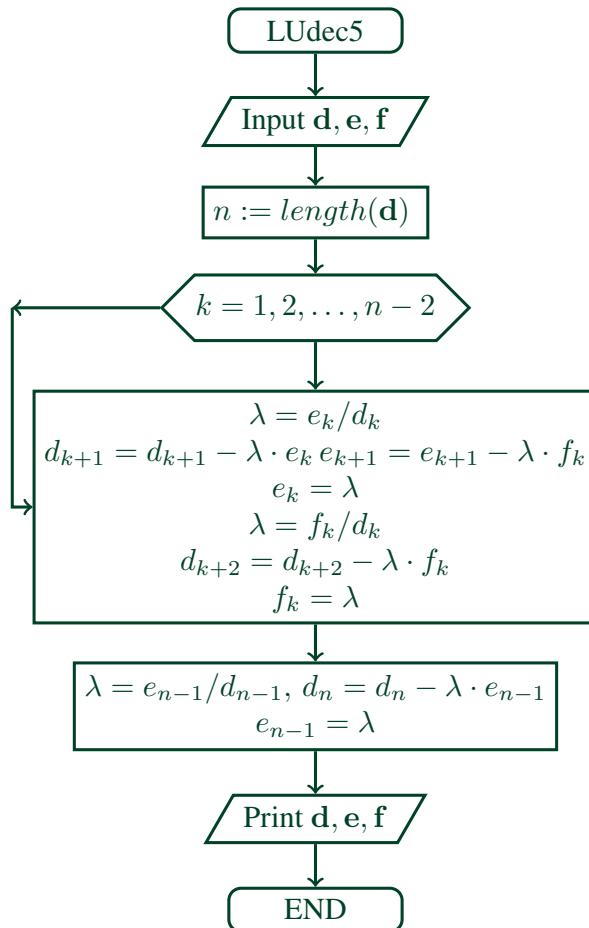
Phép biến đổi trên chỉ thay đổi các phần tử sau

$$\begin{aligned}d_{k+1} &:= d_{k+1} - (e_k/d_k)e_k \\ e_{k+1} &:= e_{k+1} - (e_k/d_k)f_k \\ d_{k+2} &:= d_{k+2} - (f_k/d_k)f_k\end{aligned}\tag{3.29}$$

Các bội số của phép khử được lưu ở các vị trí tương ứng ở tam giác trên của ma trận theo công thức

$$e_k := e_k/d_k, \quad f_k := f_k/d_k.\tag{3.30}$$

Quá trình phân rã ma trận năm đường chéo đối xứng bằng phương pháp phân rã Doolittle được minh họa trong Hình 3.7.



Hình 3.7: Sơ đồ khôi phân rã ma trận năm đường chéo đối xứng bằng phương pháp phân rã LU

Hàm LUdec5 phân rã một ma trận năm đường chéo đối xứng A lưu trữ dưới dạng $A = [f \backslash e \backslash d \backslash e \backslash f]$. Các vectơ d, e, f được tính toán và lưu lại vào ma trận được phân rã.

```

function [d,e,f] = LUdec5(d,e,f)
% LU decomposition of pentadiagonal matrix A=[f\e\d\e\f].
% USAGE: [d,e,f] = LUdec5(d,e,f)
n = length(d);
for k=1:n-2
lambda = e(k)/d(k);
d(k+1) = d(k+1) - lambda*e(k);
e(k+1) = e(k+1) - lambda*f(k);
e(k) = lambda;
lambda = f(k)/d(k);
d(k+2) = d(k+2) - lambda*f(k);
f(k) = lambda;
end
lambda = e(n-1)/d(n-1);
d(n) = d(n) - lambda*e(n-1);
e(n-1) = lambda;

```

Sau pha phân rã, ma trận A được biểu diễn dưới dạng

$$\mathbf{U}^* = \begin{bmatrix} d_1 & e_1 & f_1 & 0 & \cdots & 0 \\ 0 & d_2 & e_2 & f_2 & \cdots & 0 \\ 0 & 0 & d_3 & e_3 & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & d_{n-1} & e_{n-1} \\ 0 & 0 & \cdots & 0 & 0 & d_n \end{bmatrix}.$$

Để thực hiện pha giải, trước tiên ta xét hệ phương trình $\mathbf{Ly} = \mathbf{b}$ có ma trận hệ số mở rộng là

$$[\mathbf{L}|\mathbf{b}] = \left[\begin{array}{cccccc|c} 1 & 0 & 0 & 0 & \cdots & 0 & b_1 \\ e_1 & 1 & 0 & 0 & \cdots & 0 & b_2 \\ f_1 & e_2 & 1 & 0 & \cdots & 0 & b_3 \\ 0 & f_2 & e_3 & 1 & \cdots & 0 & b_4 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & f_{n-2} & e_{n-1} & 1 & b_n \end{array} \right].$$

Giải hệ này bằng cách thay thế thuận, ta có

$$\begin{aligned} y_1 &= b_1 \\ y_2 &= b_2 - e_1 y_1 \\ &\vdots \\ y_k &= b_k - f_{k-2} y_{k-2} - e_{k-1} y_{k-1}, k = 3, 4, \dots, n. \end{aligned} \tag{3.31}$$

Nghiệm của phương trình được thay vào hệ phương trình $\mathbf{U}\mathbf{x} = \mathbf{y}$ với ma trận hệ số mở rộng có dạng

$$[\mathbf{U}|\mathbf{y}] = \left[\begin{array}{ccccccc|c} d_1 & d_1 e_1 & d_1 f_1 & 0 & \cdots & 0 & y_1 \\ 0 & d_2 & d_2 e_2 & d_2 f_2 & \cdots & 0 & y_2 \\ 0 & 0 & d_3 & d_3 e_3 & \ddots & 0 & y_3 \\ \vdots & \vdots & & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & d_{n-1} & d_{n-1} e_{n-1} & y_{n-1} \\ 0 & 0 & \cdots & 0 & 0 & d_n & y_n \end{array} \right].$$

Nghiệm của hệ trên thu được bằng cách thay thế ngược như sau

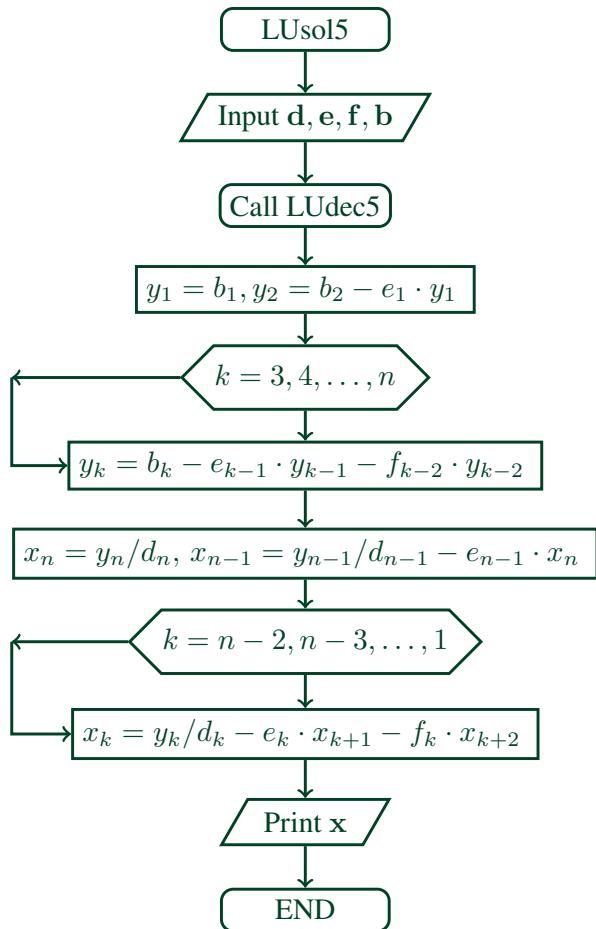
$$\begin{aligned} x_n &= xy_n/d_n \\ x_{n-1} &= y_{n-1}/d_{n-1} - e_{n-1}x_n \\ &\vdots \\ x_k &= y_k/d_k - e_k x_{k+1} - f_k x_{k+2}, k = n-2, n-3, \dots, 1. \end{aligned} \tag{3.32}$$

Dựa vào các công thức (3.31) và (3.32), pha giải của phương pháp phân rã Doolittle giải hệ phương trình tuyến tính năm đường chéo đối xứng được minh họa trong Hình 3.8.

 LUsol5.

Hàm LUsol5 thực hiện pha giải trong việc giải hệ phương trình tuyến tính năm đường chéo đối xứng. Trong hàm LUsol5, vectơ y viết chèn lên vectơ hằng b trong khi thay thế thuận và vectơ x thay thế vectơ y trong khi thay thế ngược.

```
function x=LUsol5(d,e,f,b)
% Solves A*x=b where A=[f\e\d\e\f] is the LU
% decomposition of the original pentadiagonal A.
% USAGE:x=LUsol5(d,e,f,b)
n = length(d);
b(2) = b(2) - e(1)*b(1); % Forward substitution
```



**Hình 3.8: Sơ đồ
khối giải hệ phương
trình tuyến tính
năm đường chéo đối
xứng bằng phương
pháp phân rã LU**

```

for k=3:n
    b(k) = b(k) - e(k-1)*b(k-1) - f(k-2)*b(k-2);
end
b(n) = b(n)/d(n); % Back substitution
b(n-1) = b(n-1)/d(n-1) - e(n-1)*b(n);
for k=n-2:-1:1
    b(k) = b(k)/d(k) - e(k)*b(k+1) - f(k)*b(k+2);
end
x=b;

```

VÍ DỤ 3.4.3. Giải phương trình tuyến tính 10 ẩn $\mathbf{Ax} = \mathbf{b}$, với

$$\mathbf{A} = \begin{bmatrix} 6 & -4 & 1 & 0 & 0 & \cdots \\ -4 & 6 & -4 & 1 & 0 & \cdots \\ 1 & -4 & 6 & -4 & 1 & \cdots \\ \ddots & \ddots & \ddots & \ddots & \ddots & \cdots \\ \cdots & 0 & 1 & -4 & 6 & -4 \\ \cdots & 0 & 0 & 1 & -4 & 7 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 3 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 4 \end{bmatrix}.$$

GIẢI. Vì ma trận \mathbf{A} là ma trận ngũ giác đối xứng nên sử dụng các hàm LUdec5 và LUsol5 để tính toán

```
>> % Example 3.4.3 (Solution of pentadiagonal eqs.)
n=10;
d = 6*ones(n,1); d(n) = 7;
e = -4*ones(n-1,1);
f = ones(n-2,1);
b=zeros(n,1); b(1)=3; b(n)=4;
[d,e,f] = LUdec5(d,e,f);
x = LUsol5(d,e,f,b)
```

Sau khi chạy chương trình ta có kết quả

```
>>x=
2.3872
4.1955
5.4586
6.2105
6.4850
6.3158
5.7368
4.7820
3.4850
1.8797
```

3.5 PHẦN TỬ XOAY

3.5.1 GIỚI THIỆU

Đôi khi thứ tự các phương trình trong hệ phương trình được áp dụng cho thuật toán ảnh hưởng đáng kể đến kết quả. Chẳng hạn, ta xét hệ phương trình

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

Ma trận hệ số mở rộng của hệ phương trình là

$$[\mathbf{A}|\mathbf{b}] = \left[\begin{array}{ccc|c} 2 & -1 & 0 & 1 \\ -1 & 1 & -1 & 0 \\ 0 & -1 & 1 & 0 \end{array} \right]. \quad (\text{a})$$

Ta có thể sử dụng phương pháp khử Gauss hoặc phương pháp phân rã LU để giải về phái của phương trình (a) và thu được nghiệm của hệ là $x_1 = x_2 = x_3 = 1$. Bây giờ, giả sử ta đổi vị trí của phương trình thứ nhất cho phương trình thứ ba của hệ, khi đó ma trận hệ số mở rộng của hệ là

$$[\mathbf{A}|\mathbf{b}] = \left[\begin{array}{ccc|c} 0 & -1 & 1 & 0 \\ -1 & 1 & -1 & 0 \\ 2 & -1 & 0 & 1 \end{array} \right]. \quad (\text{b})$$

Ta thấy rằng dù hệ vẫn có nghiệm $x_1 = x_2 = x_3 = 1$ nhưng ta không thể sử dụng phương pháp khử Gauss hoặc phương pháp phân rã LU để biến đổi về phái của phương trình (b) vì phần tử xoay bằng không (phần tử a_{11}).

Từ ví dụ trên ta thấy rằng đôi khi, việc thay đổi lại thứ tự các phương trình của hệ trong pha khử là cần thiết. Ta cần thay đổi lại thứ tự các phương trình trong hệ sao cho phần tử xoay khác không. Tuy nhiên, nếu khi thay đổi thứ tự các phương trình mà ta chọn được một phần tử xoay quá bé thì sẽ làm thay đổi tính chất của hệ. Nhận xét này được chỉ ra bằng hệ phương trình cho dưới dạng ma trận hệ số mở rộng sau

$$[\mathbf{A}|\mathbf{b}] = \left[\begin{array}{ccc|c} \varepsilon & -1 & 1 & 0 \\ -1 & 1 & -1 & 0 \\ 2 & -1 & 0 & 1 \end{array} \right]. \quad (\text{c})$$

Hệ phương trình (c) tương tự như hệ phương trình (b) ngoại trừ một số nhỏ ε được thay thế cho 0 tại vị trí a_{11} . Ta thấy nếu $\varepsilon \rightarrow 0$ thì hệ phương trình (b) và (c) trùng nhau. Sau bước đầu tiên của phương pháp khử Gauss, ma trận hệ số mở rộng của hệ có dạng

$$[\mathbf{A}'|\mathbf{b}'] = \left[\begin{array}{ccc|c} \varepsilon & -1 & 1 & 0 \\ 0 & 2 - 1/\varepsilon & -1 + 1/\varepsilon & 0 \\ 0 & -1 + 2/\varepsilon & -2/\varepsilon & 1 \end{array} \right]. \quad (\text{d})$$

Bởi vì máy tính làm việc với các số có độ dài cố định nên các số cần phải làm tròn. Nếu ε rất nhỏ thì $1/\varepsilon$ rất lớn và ta có thể làm tròn $a + 1/\varepsilon$ thành $1/\varepsilon$. Do đó, với hệ số rất nhỏ ε , hệ phương trình (d) có dạng

$$[\mathbf{A}'|\mathbf{b}'] = \left[\begin{array}{ccc|c} \varepsilon & -1 & 1 & 0 \\ 0 & -1/\varepsilon & 1/\varepsilon & 0 \\ 0 & 2/\varepsilon & -2/\varepsilon & 1 \end{array} \right].$$

Từ hàng thứ hai và hàng thứ ba ta thấy hệ vô nghiệm. Vấn đề này sẽ không xảy ra nếu ta đổi vị trí phương trình thứ nhất với phương trình thứ hai hoặc đổi vị trí phương trình thứ nhất với phương trình thứ ba của hệ trước khi thực hiện khử.

Ví dụ cuối cùng minh họa một trường hợp cực đoan với ε quá nhỏ thì lỗi làm tròn đã dẫn đến sai lầm hoàn toàn của hệ. Tuy các lỗi làm tròn vẫn có thể khiến cho nghiệm không đáng tin cậy nhưng nếu ta làm cho ε lớn hơn thì sai số trong việc làm tròn không còn bị bùng nổ như trước. Một lần nữa, vấn đề khó khăn này được giải quyết bằng cách đổi thứ tự cho hàng xoay.

3.5.2 MA TRẬN CHÉO TRỘI

Ma trận A có cỡ $n \times n$ được gọi là *ma trận chéo trội* nếu với mỗi hàng thì phần tử nằm trên đường chéo lớn hơn tổng các phần tử còn lại trên hàng (chúng ta nói về giá trị tuyệt đối). Do vậy, ma trận chéo trội là ma trận thoả mãn điều kiện

$$|a_{ij}| > \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}|, \quad i = 1, 2, \dots, n. \quad (3.33)$$

Ví dụ, ma trận

$$\begin{bmatrix} -2 & 4 & -1 \\ 1 & -1 & 3 \\ 4 & -2 & 1 \end{bmatrix}$$

không phải là ma trận chéo trội nhưng nếu ta đổi vị trí các hàng thì ta được ma trận

$$\begin{bmatrix} 4 & -2 & 1 \\ -2 & 4 & -1 \\ 1 & -1 & 3 \end{bmatrix}$$

là ma trận chéo trội.

Có thể chỉ ra rằng nếu ma trận hệ số A của hệ phương trình $Ax = b$ là ma trận chéo trội thì việc đổi thứ tự các phương trình không đem lại lợi ích gì vì nếu ma trận hệ số là ma trận chéo trội thì đây cũng là thứ tự tối ưu của hệ phương trình. Do đó, với một hệ bất kỳ, việc sắp xếp lại các phương trình phải sao cho ma trận hệ số càng gần với ma trận chéo trội càng tốt. Đây là vấn đề mà ta sẽ thảo luận tiếp theo.

3.5.3 PHƯƠNG PHÁP KHỦ GAUSS KẾT HỢP ĐỔI HÀNG XOAY

Xét việc giải hệ phương trình bằng phương pháp khủ Gauss. Nhắc lại ở phần trước là việc thay đổi thứ tự các phương trình để chọn hàng xoay dựa vào ưu thế của ma trận chéo trội, tức là ta chọn hàng xoay sao cho phần tử xoay lớn nhất có thể so với các phần tử còn lại trong hàng. Việc so sánh này dễ ràng được thực hiện nhờ việc thiết lập dãy s gồm các phần tử thỏa mãn

$$s_i = \max_j |a_{ij}|, \quad i = 1, 2, \dots, n. \quad (3.34)$$

Phần tử s_i được gọi là *hệ số tỉ lệ* của hàng i , chứa giá trị tuyệt đối lớn nhất của phần tử thuộc hàng i của ma trận A. vectơ s có thể thu được bằng đoạn code

```
for i=1:n
    s(i) = max(abs(A(i,1:n)))
end
```

Kích cỡ tương đối của phần tử a_{ij} (tương đối so với phần tử lớn nhất của hàng i) được xác định bởi tỉ lệ

$$r_{ij} = \frac{|a_{ij}|}{s_i}. \quad (3.35)$$

Giả sử rằng pha khủ đang thực hiện đến giai đoạn mà hàng thứ k sẽ được chọn làm

hàng xoay. Ma trận hệ số mở rộng lúc này có dạng

$$\left[\begin{array}{cccccc|c} a_{11} & a_{12} & a_{13} & a_{14} & \cdots & a_{1n} & b_1 \\ 0 & a_{22} & a_{23} & a_{24} & \cdots & a_{2n} & b_2 \\ 0 & 0 & a_{33} & a_{34} & \cdots & a_{3n} & b_3 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & a_{kk} & \cdots & a_{kn} & b_k \\ \vdots & \cdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & \cdots & 0 & a_{nk} & \cdots & a_{nn} & b_n \end{array} \right] \leftarrow$$

Chúng ta không tự động chọn a_{kk} làm phần tử xoay mà tìm trong các hàng phía dưới a_{kk} để tìm phần tử xoay tốt hơn. Lựa chọn tốt nhất là phần tử a_{pk} , là phần tử có kích cỡ tương đối lớn nhất, tức là ta chọn p sao cho

$$r_{pk} = \max_{j \geq k} r_{jk}.$$

Sau khi đã chọn được phần tử xoay tốt nhất, ta thực hiện đổi vị trí hàng k cho hàng p và tiếp tục thực hiện pha khử. Chú ý rằng quá trình đổi vị trí hai hàng ta cũng phải đổi vị trí các phần tử tương ứng trong s . Thuật toán thực hiện các công việc đó là

```

for k=1:n-1
    % Find element with largest relative size
    % and the corresponding row number p
    [Amax,p] = max(abs(A(k:n,k))./s(k:n));
    p=p+k-1;
    % If this element is very small, matrix is singular
    if Amax < eps
        error('Matrix is singular')
    end
    % Interchange rows k and p if needed
    if p~=k
        b = swapRows(b,k,p);
        s = swapRows(s,k,p);
        A = swapRows(A,k,p);
    end
    % Elimination pass
end

```

Quá trình đổi vị trí hàng i với hàng j của ma trận hoặc vectơ v được thực hiện bởi chương trình `swapRows(v,i,j)` sau đây

swapRows.

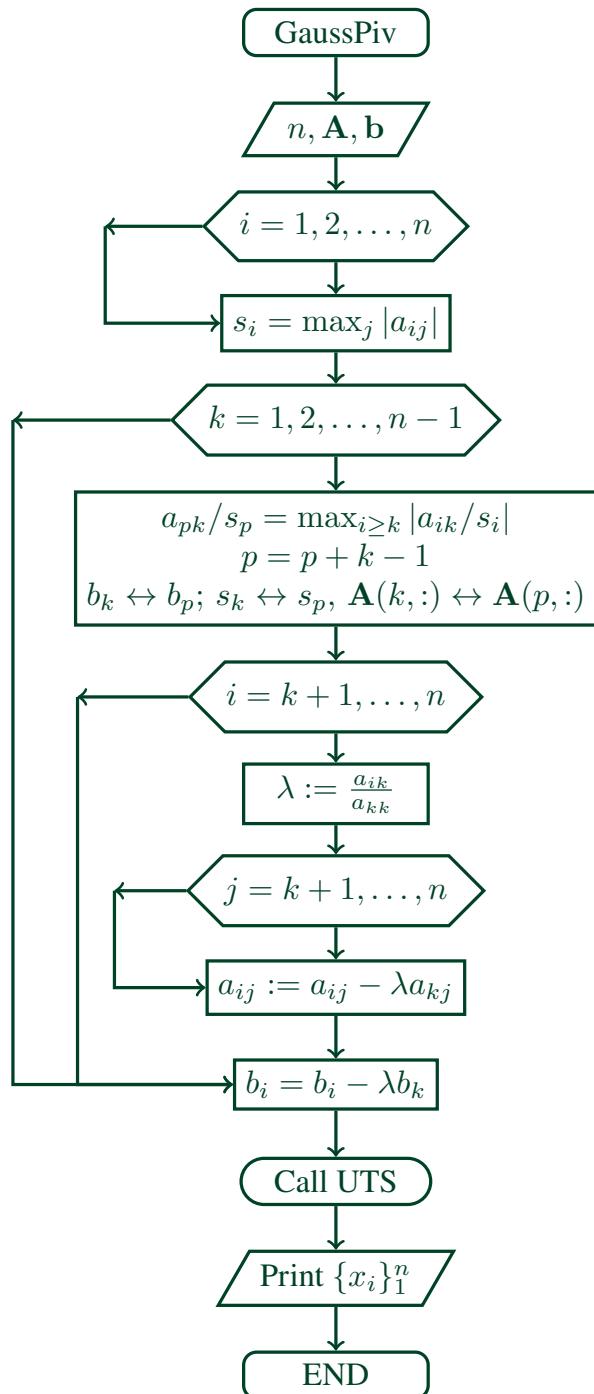
```
function v=swapRows(v,i,j)
    % Swap rows i and j of vector or matrix v.
    % USAGE:v=swapRows(v,i,j)
    temp = v(i,:);
    v(i,:) = v(j,:);
    v(j,:) = temp;
```

Phương pháp khử Gauss kết hợp đổi hàng xoay được thực hiện bằng cách điều chỉnh phương pháp khử Gauss. Tại bước khử thứ k , thay vì chọn phần tử k làm hàng xoay như phương pháp khử Gauss, phương pháp khử Gauss kết hợp đổi hàng xoay sẽ chọn hàng p , là hàng có phần tử có kích thước tương đối lớn nhất, làm hàng xoay. Việc làm này sẽ giúp cho ma trận hệ số có dạng gần với dạng tối ưu là dạng ma trận chéo trội. Quá trình sử dụng phương pháp Gauss kết hợp đổi hàng xoay giải hệ phương trình tuyến tính được minh họa trong Hình 3.9

gaussPiv.

Hàm gaussPiv sau đây thực hiện phương pháp khử Gauss với việc chọn phần tử xoay. Ngoài việc chọn phần tử xoay ra thì các bước đều thực hiện tương tự như hàm gauss ở trang 64.

```
function x=gaussPiv(A,b)
    % Solves A*x=b by Gausse limination with row pivoting.
    % USAGE:x=gaussPiv(A,b)
    if size(b,2) > 1; b = b'; end
    n=length(b);s=zeros(n,1);
    %-----Set up scale factor array-----
    for i=1:n;s(i)=max(abs(A(i,1:n)));end
    %-----Exchange rows if necessary-----
    for k=1:n-1
        [Amax,p] = max(abs(A(k:n,k))./s(k:n));
        p=p+k-1;
        if Amax < eps; error('Matrix is singular'); end
        if p~=k
            b = swapRows(b,k,p);
            s = swapRows(s,k,p);
            A = swapRows(A,k,p);
        end
    end
```



Hình 3.9: Sơ đồ khối giải hệ phương trình tuyến tính bằng phương pháp khử Gauss kết hợp đổi hàng xoay

```

end
%-----Elimination pass-----
for i=k+1:n
    if A(i,k) ~= 0
        lambda = A(i,k)/A(k,k);
        A(i,k+1:n) = A(i,k+1:n) - lambda*A(k,k+1:n);
        b(i) = b(i) - lambda*b(k);
    end

```

```

    end
end
%-----Back substitution phase-----
for k=n:-1:1
    b(k) = (b(k) - A(k,k+1:n)*b(k+1:n)) / A(k,k);
end
x=b;

```

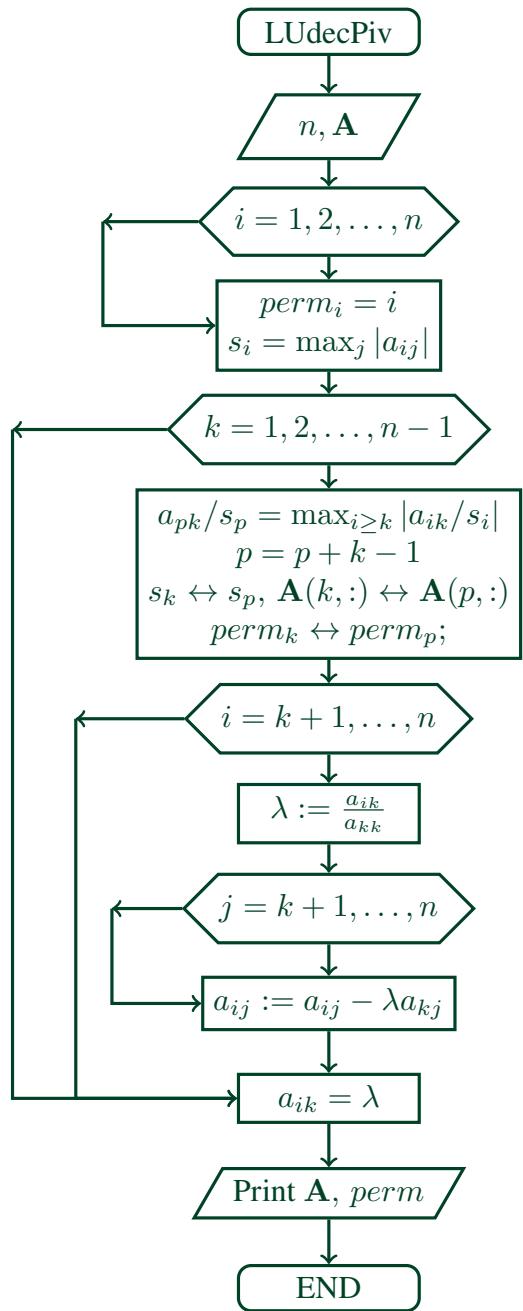
Phương pháp khử Gauss có thể được thay đổi để áp dụng cho phương pháp phân rã Doolittle. Điều quan trọng là luôn làm cho bản ghi thay đổi tương ứng trong pha khử. Trong hàm LUdecPiv, bản ghi được lưu dưới dạng dãy trong biến perm với giá trị ban đầu là $[1, 2, \dots, n]^T$. Khi hai hàng nào đó bị thay đổi sẽ kéo theo thay đổi hai phần tử tương ứng của bản ghi. Do đó, perm chỉ ra các hàng ban đầu đã được thay đổi như thế nào. Thông tin này sẽ được sử dụng trong hàm LUSolPiv để xếp lại thứ tự vectơ hàng b theo thứ tự tương ứng trong quá trình giải hệ phương trình bằng cách thay thế thuận và thay thế nghịch.

 LUdecPiv.

```

function [A,perm] = LUdecPiv(A)
% LU decomposition of matrix A; returns A=[L\U]
% and the row permutation vector 'perm'.
% USAGE: [A,perm] = LUdecPiv(A)
n = size(A,1); s = zeros(n,1);
perm = (1:n)';
%-----Set up scale factor array-----
for i=1:n;s(i)=max(abs(A(i,1:n)));end
%-----Exchange rows if necessary-----
for k=1:n-1
    [Amax,p] = max(abs(A(k:n,k))./s(k:n));
    p=p+k-1;
    if Amax < eps
        error('Matrix is singular')
    end
    if p~=k
        s = swapRows(s,k,p);
        A = swapRows(A,k,p);
        perm = swapRows(perm,k,p);
    end
end

```



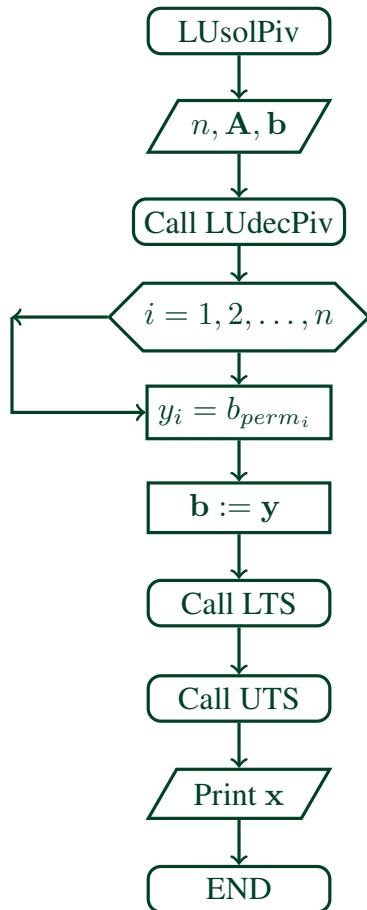
Hình 3.10: Sơ đồ khôi phục rã ma trận hệ số bằng phương pháp LU kết hợp đổi hàng xoay

```

end
%-----Elimination pass-----
for i=k+1:n
    if A(i,k) ~= 0
        lambda = A(i,k)/A(k,k);
        A(i,k+1:n) = A(i,k+1:n) - lambda*A(k,k+1:n);
        A(i,k) = lambda;
    end
end

```

end



**Hình 3.11: Sơ đồ
khối giải hệ phương
trình tuyến tính
bằng phương pháp
phân rã LU kết hợp
đổi hàng xoay**

File: LUsolPiv.m

```

function x=LUsolPiv(A,b,perm)
% Solves L*U*b=x, where A contains row-wise
% permutation of L and U in the form A=[L\U].
% vector 'perm' holds the row permutation data.
% USAGE: x = LUsolPiv(A,b,perm)
%-----Rearrange b, store it in x-----
if size(b) > 1; b = b'; end
n = size(A,1);
x=b;
for i=1:n;x(i)=b(perm(i));end
%-----Forward and back substitution-----
for k=2:n
    x(k) = x(k) - A(k,1:k-1)*x(1:k-1);
end

```

```

end
for k=n:-1:1
x(k) = (x(k) - A(k,k+1:n)*x(k+1:n)) / A(k,k);
end

```

Việc xếp lại các phương trình trong hệ phương trình có hai hạn chế. Thứ nhất là làm tăng khối lượng tính toán. Thứ hai là phá vỡ cấu trúc đối xứng hoặc cấu trúc dải của ma trận hệ số ban đầu. Vấn đề hạn chế thứ hai là rất nghiêm trọng vì trong kỹ thuật, các ma trận hệ số phát sinh đều có cấu trúc đối xứng hoặc cấu trúc dải, một tính chất có thể lợi dụng để giải hệ phương trình tuyến tính. May mắn là hầu hết các ma trận này đều có tính chất chéo trội nên việc áp dụng xếp lại thứ tự các phương trình là không cần thiết.

Vấn đề đặt ra là khi nào chọn việc xếp lại thứ tự các phương trình. Kinh nghiệm thực tế chỉ ra rằng việc xếp lại các phương trình sẽ phản tác dụng khi ma trận hệ số có dạng dải. Trường hợp ma trận xác định dương hoặc ma trận dải cũng không nên áp dụng việc thay đổi thứ tự. Và chúng ta cũng nên nhớ rằng việc xếp lại thứ tự các phương trình không phải cách duy nhất để hạn chế sai số làm tròn, chẳng hạn ta có thể làm tròn các số đến độ chính xác gấp đôi.

Ta chú ý rằng những vấn đề trên chỉ là những vấn đề phát sinh từ những bài toán kỹ thuật. Trong trường hợp tổng quát, việc xếp lại thứ tự các phương trình để chọn phần tử xoay tốt nhất là cần thiết.

VÍ DỤ 3.5.1. Hãy sử dụng phương pháp Gauss kết hợp xếp lại các phương trình để giải hệ phương trình $\mathbf{Ax} = \mathbf{b}$, với

$$\mathbf{A} = \begin{bmatrix} 2 & -2 & 6 \\ -2 & 4 & 3 \\ -1 & 8 & 4 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 16 \\ 0 \\ -1 \end{bmatrix}.$$

GIẢI. Ma trận hệ số mở rộng và thang hệ số của hệ phương trình là

$$[\mathbf{A}|\mathbf{b}] = \left[\begin{array}{ccc|c} 2 & -2 & 6 & 16 \\ -2 & 4 & 3 & 0 \\ -1 & 8 & 4 & -1 \end{array} \right], \quad \mathbf{s} = \begin{bmatrix} 6 \\ 4 \\ 8 \end{bmatrix}.$$

Chú ý rằng \mathbf{s} chứa trị tuyệt đối lớn nhất của các phần tử mỗi hàng của ma trận \mathbf{A} . Tại bước này, tất cả các phần tử ở cột thứ nhất của ma trận \mathbf{A} có thể làm phần tử xoay. Để

xác định phần tử xoay tốt nhất, ta tính kích thước tương đối của các phần tử ở cột thứ nhất

$$\begin{bmatrix} r_{11} \\ r_{21} \\ r_{31} \end{bmatrix} = \begin{bmatrix} |a_{11}|/s_1 \\ |a_{21}|/s_2 \\ |a_{31}|/s_3 \end{bmatrix} = \begin{bmatrix} 2/6 \\ 2/4 \\ 1/8 \end{bmatrix}.$$

Phần tử r_{21} lớn nhất nên ta chọn a_{21} làm phần tử xoay. Do đó, ta đổi hàng 1 và hàng 2 của ma trận hệ số mở rộng và thang hệ số của hệ phương trình

$$[\mathbf{A}|\mathbf{b}] = \left[\begin{array}{ccc|c} -2 & 4 & 3 & 0 \\ 2 & -2 & 6 & 16 \\ -1 & 8 & 4 & -1 \end{array} \right] \leftarrow , \quad \mathbf{s} = \begin{bmatrix} 4 \\ 6 \\ 8 \end{bmatrix}.$$

Thực hiện khử cho cột thứ nhất ta có

$$[\mathbf{A}'|\mathbf{b}'] = \left[\begin{array}{ccc|c} -2 & 4 & 3 & 0 \\ 0 & 2 & 9 & 16 \\ 0 & 6 & 5/2 & -1 \end{array} \right], \quad \mathbf{s} = \begin{bmatrix} 4 \\ 6 \\ 8 \end{bmatrix}.$$

Để chọn một trong hai phần tử a_{22} hoặc a_{32} làm phần tử xoay, ta tính

$$\begin{bmatrix} * \\ r_{22} \\ r_{32} \end{bmatrix} = \begin{bmatrix} * \\ |a_{22}|/s_2 \\ |a_{32}|/s_3 \end{bmatrix} = \begin{bmatrix} * \\ 1/3 \\ 3/4 \end{bmatrix}.$$

Chú ý rằng r_{12} không liên quan vì hàng 1 đã được chọn làm hàng xoay. Do vậy ta loại r_{12} ra. Vì r_{32} lớn hơn r_{22} nên hàng thứ ba chọn làm hàng xoay sẽ tốt hơn. Thực hiện đổi vị trí hàng 2 và hàng 3 ta có

$$[\mathbf{A}'|\mathbf{b}'] = \left[\begin{array}{ccc|c} -2 & 4 & 3 & 0 \\ 0 & 6 & 5/2 & -1 \\ 0 & 2 & 9 & 16 \end{array} \right] \leftarrow , \quad \mathbf{s} = \begin{bmatrix} 4 \\ 8 \\ 6 \end{bmatrix}.$$

Thực hiện khử cho a_{32} ta có

$$[\mathbf{A}''|\mathbf{b}''] = [\mathbf{U}|\mathbf{c}] \left[\begin{array}{ccc|c} -2 & 4 & 3 & 0 \\ 0 & 6 & 5/2 & -1 \\ 0 & 0 & 49/6 & 49/3 \end{array} \right].$$

Đến đây là hoàn thành pha khử. Lưu ý rằng ma trận U chính là kết quả của phép phân rã LU của ma trận hoán vị của A (với thứ tự các hành thay đổi theo thứ tự chọn hàng xoay)

$$\begin{bmatrix} -2 & 4 & 3 \\ -1 & 8 & 4 \\ 2 & -2 & 6 \end{bmatrix}.$$

Thực hiện giải hệ phương trình $Ux = c$ bằng cách thay thế ngược, ta có nghiệm của hệ $x^T = [1 \ -1 \ 2]$.

□

3.6 PHƯƠNG PHÁP LẶP ĐƠN

3.6.1 GIỚI THIỆU

Trong các mục trước, chúng ta đã thảo luận các phương pháp trực tiếp giải hệ phương trình tuyến tính. Đặc điểm chung của các phương pháp này là tìm nghiệm của hệ sau hữu hạn các phép biến đổi. Nếu các bước tính toán đều chính xác thì sẽ cho ta nghiệm đúng của hệ phương trình.

Phương pháp lặp, hoặc *phương pháp gián tiếp*, bắt đầu với giá trị phỏng đoán ban đầu của x , sau đó quá trình tính toán sẽ được lặp cho đến khi tìm được giá trị xấp xỉ của x với sai số cho phép. Để giải hệ phương trình $Ax = b$, ta biến đổi hệ về dạng

$$x = Hx + g,$$

với H là ma trận $cỡ n \times n$ và g là vectơ hằng số. Chọn giá trị phỏng đoán $x^{(0)}$ ban đầu (sau này gọi tắt là *giá trị ban đầu*), ta tính các xấp xỉ nghiệm của hệ theo công thức lặp

$$x^{(m)} = Hx^{(m-1)} + g, \quad m = 1, 2, \dots \quad (3.36)$$

Vì phải yêu cầu số lần lặp lớn nên thường thì phương pháp gián tiếp sẽ chậm hơn các phương pháp trực tiếp. Tuy nhiên, phương pháp lặp lại có những ưu điểm cho một số bài toán nhất định.

- Nó có thể lưu trữ các phần tử khác không của ma trận hệ số. Điều này dẫn đến có thể giải quyết các ma trận thừa nhưng không có dạng dải. Trong nhiều bài toán, việc lưu trữ tất cả các phần tử của ma trận là không cần thiết.

2. Quá trình lặp có thể tự sửa lỗi, có nghĩa là các lỗi làm tròn (hoặc có thể là các lỗi số học) ở vòng lặp này sẽ được sửa ở vòng lặp tiếp theo.

Phương pháp lặp đơn trong công thức (3.36) gọi là hội tụ nếu $\|\mathbf{x}^{(m)} - \alpha\| \rightarrow 0$ khi $m \rightarrow \infty$, với α là nghiệm của hệ phương trình tuyến tính. Một vấn đề nghiêm trọng của các phương pháp lặp không phải lúc nào cũng hội tụ về nghiệm của hệ. Giá trị phỏng đoán ban đầu không có vai trò trong việc xác định sự hội tụ của một phương pháp lặp. Nếu một phương pháp lặp hội tụ với một vectơ thì nó cũng hội tụ với bất kỳ giá trị ban đầu nào. Giá trị ban đầu chỉ ảnh hưởng đến số lần lặp. Điều kiện để phương pháp lặp đơn theo công thức (3.36) hội tụ được phát biểu trong định lý sau đây.

ĐỊNH LÝ 3.6.1. *Nếu $\|\mathbf{H}\| < 1$ thì dãy lặp xác định bởi công thức (3.36) hội tụ về nghiệm α với giá trị ban đầu $\mathbf{x}^{(0)} \in \mathbb{R}^n$ tùy ý. Hơn nữa, ta có các công thức đánh giá sai số*

$$\|\mathbf{x}^{(m)} - \alpha\| \leq \frac{\|\mathbf{H}\|}{1 - \|\mathbf{H}\|} \|\mathbf{x}^{(m)} - \mathbf{x}^{(m-1)}\|, \quad (3.37)$$

$$\|\mathbf{x}^{(m)} - \alpha\| \leq \frac{\|\mathbf{H}\|^m}{1 - \|\mathbf{H}\|} \|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\|. \quad (3.38)$$

3.6.2 PHƯƠNG PHÁP LẶP JACOBI

Cho hệ phương trình $\mathbf{Ax} = \mathbf{b}$ dưới dạng

$$\sum_{j=1}^n a_{ij}x_j = b_i, \quad i = 1, 2, \dots, n.$$

Khai triển theo biến x_i ta có

$$a_{ii}x_i + \sum_{\substack{j=1 \\ j \neq i}}^n a_{ij}x_j = b_i, \quad i = 1, 2, \dots, n.$$

Nếu $a_{ii} \neq 0$, suy ra

$$x_i = \sum_{\substack{j=1 \\ j \neq i}}^n \left(-\frac{a_{ij}}{a_{ii}} \right) x_j + \frac{b_i}{a_{ii}}, \quad i = 1, 2, \dots, n.$$

Khi đó, phương trình $\mathbf{Ax} = \mathbf{b}$ tương đương với phương trình

$$\mathbf{x} = \mathbf{Hx} + \mathbf{g}, \quad (3.39)$$

với

$$\mathbf{H} = \begin{bmatrix} 0 & -a_{12}/a_{11} & -a_{13}/a_{11} & \cdots & -a_{1n}/a_{11} \\ -a_{21}/a_{22} & 0 & -a_{23}/a_{22} & \cdots & -a_{2n}/a_{22} \\ -a_{31}/a_{33} & -a_{32}/a_{33} & 0 & \cdots & -a_{3n}/a_{33} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -a_{n1}/a_{nn} & -a_{n2}/a_{nn} & -a_{n3}/a_{nn} & \cdots & 0 \end{bmatrix}, \quad \mathbf{g} = \begin{bmatrix} b_1/a_{11} \\ b_2/a_{22} \\ b_3/a_{33} \\ \vdots \\ b_n/a_{nn} \end{bmatrix}.$$

Với vectơ $\mathbf{x}^{(0)}$ là giá trị ban đầu tuỳ ý, ta có phương pháp *lặp Jacobi* theo công thức sau

$$\mathbf{x}^{(m)} = \mathbf{H}\mathbf{x}^{(m-1)} + \mathbf{g}, \quad m = 1, 2, \dots \quad (3.40)$$

VÍ DỤ 3.6.1. Với giá trị ban đầu $\mathbf{x}^{(0)} = [0, 0, 0]^T$, hãy giải hệ phương trình $\mathbf{Ax} = \mathbf{b}$ bằng phương pháp lặp Jacobi trong đó

$$\mathbf{A} = \begin{bmatrix} 10 & 2 & 1 \\ 1 & 10 & 2 \\ 1 & 1 & 10 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 10 \\ 12 \\ 8 \end{bmatrix}.$$

GIẢI. Bằng các phương pháp trực tiếp, ta dễ thấy nghiệm của hệ phương trình là $\alpha = [704/955, 956/955, 598/955]^T$. Để xây dựng công thức lặp Jacobi, trước tiên ta biểu diễn x_i từ phương trình thứ i , ta có hệ phương trình tương đương

$$\begin{cases} x_1 = -0.2x_2 - 0.1x_1 + 1 \\ x_2 = -0.1x_1 - 0.2x_2 + 1.2 \\ x_3 = -0.1x_1 - 0.1x_2 + 0.8 \end{cases}$$

Ta có công thức lặp Jacobi như sau

$$\begin{cases} x_1^{(m)} = -0.2x_2^{(m-1)} - 0.1x_1^{(m-1)} + 1 \\ x_2^{(m)} = -0.1x_1^{(m-1)} - 0.2x_2^{(m-1)} + 1.2 \\ x_3^{(m)} = -0.1x_1^{(m-1)} - 0.1x_2^{(m-1)} + 0.8 \end{cases}, \quad m = 1, 2, \dots$$

Lấy giá trị ban đầu $x^{(0)} = [0 \ 0 \ 0]^T$, tính toán cụ thể các giá trị xấp xỉ nghiệm theo các bước lặp.

Bước 1. Tính $\mathbf{x}^{(1)}$.

$$\begin{cases} x_1^{(1)} = -0.2x_2^{(0)} - 0.1x_1^{(0)} + 1 = 1 \\ x_2^{(1)} = -0.1x_1^{(0)} - 0.2x_2^{(0)} + 1.2 = 1.2 \\ x_3^{(1)} = -0.1x_1^{(0)} - 0.1x_2^{(0)} + 0.8 = 0.8 \end{cases}$$

nên $\mathbf{x}^{(1)} = [1 \ 1.2 \ 0.8]^T$.

Bước lặp 2. Tính $\mathbf{x}^{(2)}$.

$$\begin{cases} x_1^{(2)} = -0.2x_2^{(1)} - 0.1x_1^{(1)} + 1 = 0.68 \\ x_2^{(2)} = -0.1x_1^{(1)} - 0.2x_2^{(1)} + 1.2 = 0.94 \\ x_3^{(2)} = -0.1x_1^{(1)} - 0.1x_2^{(1)} + 0.8 = 0.58 \end{cases}$$

nên $\mathbf{x}^{(2)} = [0.68 \ 0.94 \ 0.58]^T$.

Bước lặp 3. Tính $\mathbf{x}^{(3)}$.

$$\begin{cases} x_1^{(3)} = -0.2x_2^{(2)} - 0.1x_1^{(2)} + 1 = 0.754 \\ x_2^{(3)} = -0.1x_1^{(2)} - 0.2x_2^{(2)} + 1.2 = 1.016 \\ x_3^{(3)} = -0.1x_1^{(2)} - 0.1x_2^{(2)} + 0.8 = 0.638 \end{cases}$$

nên $\mathbf{x}^{(3)} = [0.754 \ 1.016 \ 0.638]^T$.

⋮

Do ma trận hệ số \mathbf{H}

$$\mathbf{H} = \begin{bmatrix} 0 & -0.2 & -0.1 \\ -0.1 & 0 & -0.2 \\ -0.1 & -0.1 & 0 \end{bmatrix}$$

có chuẩn là $\|\mathbf{H}\| = 0.3 < 1$ nên phép lặp Jacobi hội tụ với sai số được tính bởi công thức (3.37) và (3.38) tương ứng như sau

$$\begin{aligned} \|\mathbf{x}^{(m)} - \boldsymbol{\alpha}\| &\leq \frac{0.3}{1 - 0.3} \|\mathbf{x}^{(m)} - \mathbf{x}^{(m-1)}\|, \\ \|\mathbf{x}^{(m)} - \boldsymbol{\alpha}\| &\leq \frac{0.3^m}{1 - 0.3} \|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\|. \end{aligned}$$

Thực hiện tính toán nghiệm xấp xỉ và các sai số của phương pháp lặp trong 7 bước lặp, ta có kết quả tính toán được cho trong bảng sau.

Ta có $\|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\| = 3$ nên nếu muốn tìm nghiệm xấp xỉ \mathbf{x}^* với độ chính xác $\varepsilon = 10^{-5}$ thì số lần lặp m tối thiểu được xác định bởi

$$\frac{0.3^m}{1 - 0.3} \times 3 < 10^{-5}.$$

Suy ra $m \geq 11$.

□

Bước lặp	$x_1^{(m)}$	$x_2^{(m)}$	$x_3^{(m)}$	Sai số (3.37)	Sai số (3.38)	Sai số đúng
1	1.00000	1.20000	0.80000	1.28571	1.28571	0.63560
2	0.68000	0.94000	0.58000	0.34286	0.38571	0.16440
3	0.75400	1.01600	0.63800	0.08914	0.11571	0.04360
4	0.73300	0.99700	0.62300	0.02357	0.03471	0.01140
5	0.73830	1.00210	0.62700	0.00617	0.01041	0.00300
6	0.73688	1.00077	0.62596	0.00162	0.00312	0.00079
7	0.73725	1.00112	0.62624	0.00043	0.00094	0.00021

Bảng 3.3: Bảng kết quả sử dụng phương pháp lặp Jacobi để giải hệ phương trình tuyến tính $\mathbf{Ax} = \mathbf{b}$ sau 7 bước lặp.

3.6.3 PHƯƠNG PHÁP LẶP SEIDEL

Phương pháp lặp Seidel là một cải tiến của phương pháp Jacobi với ý tưởng *thông tin càng được sử dụng càng sớm thì càng tốt*. Tức là với phương pháp Jacobi thì biến ở vòng lặp trước được áp dụng để tính cho biến ở vòng lặp sau nhưng với phương pháp Seidel thì có thể tận dụng các biến vừa tính ở vòng lặp hiện tại thay thế cho các biến tương ứng ở vòng lặp trước. Chẳng hạn, các bước lặp trong Ví dụ 3.6.1 sẽ được thay đổi như sau.

Bước lặp 1. Tính $\mathbf{x}^{(1)}$.

$$\begin{cases} x_1^{(1)} = -0.2x_2^{(0)} - 0.1x_1^{(0)} + 1 = 1.00 \\ x_2^{(1)} = -0.1x_1^{(1)} - 0.2x_2^{(0)} + 1.2 = 1.10 \\ x_3^{(1)} = -0.1x_1^{(1)} - 0.1x_2^{(1)} + 0.8 = 0.59 \end{cases}$$

Thay đổi: $x_1^{(1)}$ được tính ở phương trình thứ nhất sẽ được đưa luôn vào phương trình thứ hai để tính $x_2^{(1)}$. Tiếp theo, $x_1^{(1)}, x_2^{(1)}$ được tính ở phương trình thứ nhất và phương trình thứ hai sẽ được đưa vào phương trình thứ ba để tính $x_3^{(1)}$. Ta có $\mathbf{x}^{(1)} = [1.00 \ 1.10 \ 0.59]$.

Bước lặp 2. Tính $\mathbf{x}^{(2)}$.

$$\begin{cases} x_1^{(2)} = -0.2x_2^{(1)} - 0.1x_1^{(1)} + 1 = 0.72100 \\ x_2^{(2)} = -0.1x_1^{(2)} - 0.2x_2^{(1)} + 1.2 = 1.00990 \\ x_3^{(2)} = -0.1x_1^{(2)} - 0.1x_2^{(2)} + 0.8 = 0.62691 \end{cases}$$

Thay đổi: $x_1^{(2)}$ được tính ở phương trình thứ nhất sẽ được đưa luôn vào phương trình thứ hai để tính $x_2^{(2)}$. Tiếp theo, $x_1^{(2)}, x_2^{(2)}$ được tính ở phương trình thứ nhất và phương trình thứ hai sẽ được đưa vào phương trình thứ ba để tính $x_3^{(2)}$. Ta có $\mathbf{x}^{(2)} = [0.72100 \ 1.00990 \ 0.62691]$.

⋮

Bước lặp m . Tính $\mathbf{x}^{(m)}$.

$$\begin{cases} x_1^{(m)} = -0.2x_2^{(m-1)} - 0.1x_1^{(m-1)} + 1 \\ x_2^{(m)} = -0.1x_1^{(m)} - 0.2x_2^{(m-1)} + 1.2 \\ x_3^{(m)} = -0.1x_1^{(m)} - 0.1x_2^{(m)} + 0.8 \end{cases}, \quad m = 1, 2, \dots$$

Nếu ma trận hệ số \mathbf{H} của phép lặp Jacobi được biểu diễn dưới dạng

$$\mathbf{H} = \begin{bmatrix} 0 & -0.2 & -0.1 \\ -0.1 & 0 & -0.2 \\ -0.1 & -0.1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ -0.1 & 0 & 0 \\ -0.1 & -0.1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & -0.2 & -0.1 \\ 0 & 0 & -0.2 \\ 0 & 0 & 0 \end{bmatrix}$$

thì các phần tử phía dưới đường chéo chính của ma trận \mathbf{H} sẽ được kết hợp với các biến vừa tính của vòng lặp m còn các phần tử phía trên đường chéo chính kết hợp với các biến được tính trong vòng lặp $m - 1$ (vòng lặp trước).

Từ ví dụ trên, ta có thể đi đến xây dựng công thức lặp Seidel như sau. Để giải hệ phương trình $\mathbf{A}\mathbf{x} = \mathbf{b}$, ta biến đổi tương đương về hệ $\mathbf{x} = \mathbf{H}\mathbf{x} + \mathbf{g}$ sau đó, với giá trị ban đầu $\mathbf{x}^{(0)}$ cho trước, tính nghiệm xấp xỉ theo công thức lặp

$$\mathbf{x}^{(m)} = \mathbf{H}_L \mathbf{x}^{(m)} + \mathbf{H}_U \mathbf{x}^{(m-1)} + \mathbf{g}, \quad m = 1, 2, \dots, \quad (3.41)$$

với $\mathbf{H}_L, \mathbf{H}_U$ là các ma trận tam giác tương ứng các phần tử ở vị trí tam giác trên và vị trí tam giác dưới của ma trận \mathbf{H} có dạng

$$\mathbf{H}_L = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ h_{21} & 0 & 0 & \cdots & 0 \\ h_{31} & h_{32} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h_{n1} & h_{n2} & h_{n3} & \cdots & 0 \end{bmatrix}, \quad \mathbf{H}_U = \begin{bmatrix} 0 & h_{12} & h_{13} & \cdots & h_{1n} \\ 0 & 0 & h_{23} & \cdots & h_{2n} \\ 0 & 0 & 0 & \cdots & h_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}.$$

Một số tính chất của phương pháp lặp Seidel được cho trong nhận xét sau đây.

NHẬN XÉT 3.6.1. 1. Phương pháp Seidel hội tụ khi $\|\mathbf{H}\| < 1$.

2. Phương pháp Seidel tiết kiệm bộ nhớ hơn phương pháp Jacobi vì thành phần vừa được tính được sử dụng ngay để tính thành phần tiếp theo.

3. Phương pháp Seidel cũng là phương pháp lặp đơn với công thức lặp

$$\mathbf{x}^{(m)} = (\mathbf{I} - \mathbf{H}_L)^{-1} \mathbf{H}_U \mathbf{x}^{(m-1)} + (\mathbf{I} - \mathbf{H}_L)^{-1} \mathbf{g}, \quad m = 1, 2, \dots \quad (3.42)$$

4. Trong trường hợp ma trận A là ma trận chéo trội, phương pháp được gọi là phương pháp Gauss-Seidel.

Nội dung của phương pháp lặp Gauss-Seidel để giải hệ phương trình được minh họa bằng sơ đồ khối cho trong Hình 3.12.

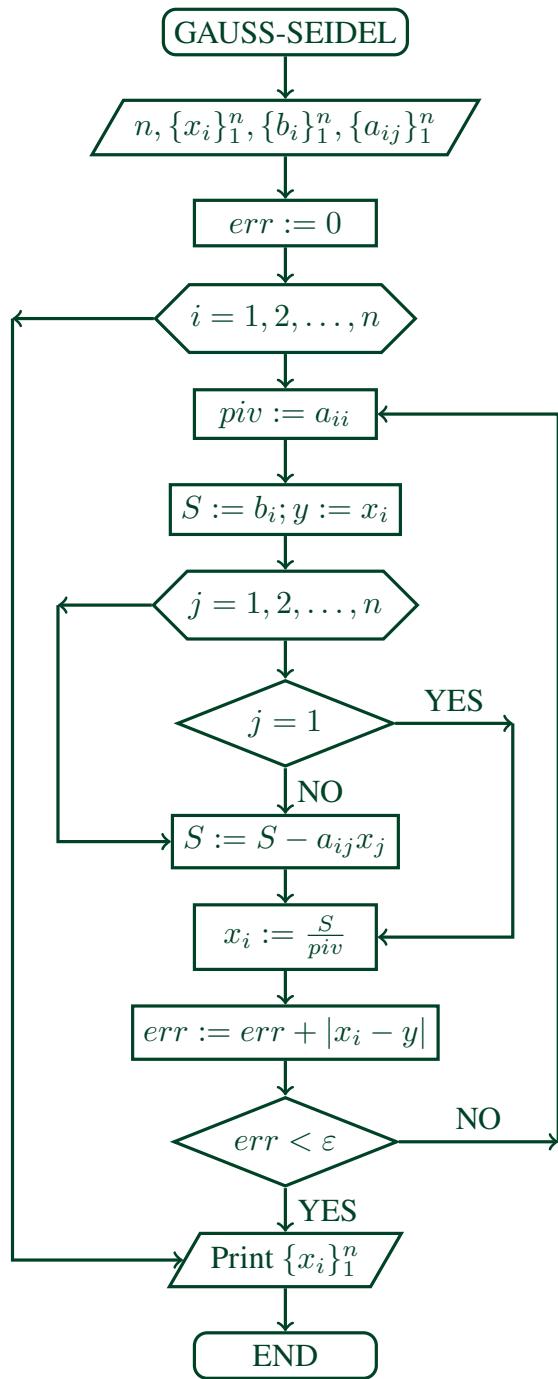
Dựa vào sơ đồ khối trong Hình 3.12 việc giải hệ phương trình tuyến tính theo phương pháp lặp Gauss-Seidel được lập trình bằng ngôn ngữ lập trình MATLAB như sau.

gaussSeidel.

```

function [x, numIter, omega] =
    gaussSeidel(func, x, maxIter, epsilon)
% Solves Ax=b by Gauss-Seidel method with relaxation.
% USAGE: [x, numIter, omega] =
    gaussSeidel(func, x, maxIter, epsilon)
% INPUT:
% func = handle of function that returns improved x using
% the iterative formulas in Eq. (2.35).
% x = starting solution vector
% maxIter = allowable number of iterations (default is 500)
% epsilon = error tolerance (default is 1.0e-9)
% OUTPUT:
% x = solution vector
% numIter = number of iterations carried out
% omega = computed relaxation factor
if nargin < 4; epsilon = 1.0e-9; end
if nargin < 3; maxIter = 500; end
k=10;p=1;omega=1;
for numIter = 1:maxIter
    xOld = x;
    x = feval(func, x, omega);
    if abs(x-xOld) < epsilon
        break;
    end
end
numIter = numIter+1;

```



Hình 3.12: Sơ đồ khôi của phương pháp Gauss-Seidel giải hệ phương trình tuyến tính

```

dx = sqrt(dot(x - xOld, x - xOld));
if dx < epsilon; return; end
if numIter == k; dx1 = dx; end
if numIter==k+p
    omega = 2 / (1 + sqrt(1 - (dx/dx1)^(1/p)));
end
error('Too many iterations')
    
```

VÍ DỤ 3.6.2. Giải hệ phương trình

$$\begin{bmatrix} 4 & -1 & 1 \\ -1 & 4 & -2 \\ 1 & -2 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 12 \\ -1 \\ 5 \end{bmatrix}$$

bằng phương pháp Gauss-Seidel.

GIẢI. Với dữ liệu đã cho, công thức lặp trong (3.41) trở thành

$$\begin{aligned} x_1^{(m)} &= \frac{1}{4} (12 + x_2^{(m-1)} - x_3^{(m-1)}), \\ x_2^{(m)} &= \frac{1}{4} (-1 + x_1^{(m)} + 2x_3^{(m-1)}), \\ x_3^{(m)} &= \frac{1}{4} (5 - x_1^{(m)} + 2x_2^{(m)}). \end{aligned}$$

Chọn các giá trị bắt đầu $x_1^{(0)} = x_2^{(0)} = x_3^{(0)} = 0$ ta có bước lặp thứ nhất

$$\begin{aligned} x_1^{(1)} &= \frac{1}{4} (12 + 0 - 0) = 3, \\ x_2^{(1)} &= \frac{1}{4} (-1 + 3 + 2(0)) = 0.5, \\ x_3^{(1)} &= \frac{1}{4} (5 - 3 + 2(0.5)) = 0.75. \end{aligned}$$

Bước lặp thứ hai là

$$\begin{aligned} x_1^{(2)} &= \frac{1}{4} (12 + 0.5 - 0.75) = 2.9375, \\ x_2^{(2)} &= \frac{1}{4} (-1 + 2.9375 + 2(0.75)) = 0.85938, \\ x_3^{(2)} &= \frac{1}{4} (5 - 2.9375 + 2(0.85938)) = 0.94531. \end{aligned}$$

và bước lặp thứ ba có kết quả là

$$\begin{aligned} x_1^{(3)} &= \frac{1}{4} (12 + 0.85938 - 0.94531) = 2.97852, \\ x_2^{(3)} &= \frac{1}{4} (-1 + 2.97852 + 2(0.94531)) = 0.96729, \\ x_3^{(3)} &= \frac{1}{4} (5 - 2.97852 + 2(0.96729)) = 0.98902. \end{aligned}$$

Tiếp tục thực hiện thêm năm bước lặp, kết quả hội tụ về nghiệm chính xác là $x_1 = 3, x_2 = 1, x_3 = 1$ với năm chữ số thập phân.

□

VÍ DỤ 3.6.3. Viết chương trình giải hệ n ẩn¹ sau đây bằng phương pháp Gauss-Seidel (chương trình có thể làm việc với mọi giá trị n).

$$\begin{bmatrix} 2 & -1 & 0 & 0 & \dots & 0 & 0 & 0 & 1 \\ -1 & 2 & -1 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & \dots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-2} \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

Chạy chương trình với $n = 20$. Nghiệm chính xác có thể được chỉ ra là $x_i = -n/4 + i/2, i = 1, 2, \dots, n$.

GIẢI. Trong trường hợp này, công thức lặp trong (3.41) là

$$\begin{aligned} x_1 &= \omega(x_2 - x_n)/2 + (1 - \omega)x_1 \\ x_i &= \omega(x_{i-1} - x_{i+1})/2 + (1 - \omega)x_i, \quad i = 2, 3, \dots, n-1 \\ x_n &= \omega(1 - x_1 + x_{n-1})/2 + (1 - \omega)x_n, \end{aligned} \tag{a}$$

được tính toán bởi chương trình sau đây

```
function x=fex3_6_3(x, omega)
% Iteration formula Eq. (3.41) for Example 3.6.3.
n = length(x);
x(1) = omega*(x(2) - x(n))/2 + (1-omega)*x(1);
for i=2:n-1
    x(i) = omega*(x(i-1) + x(i+1))/2 + (1-omega)*x(i);
end
x(n) = omega * (1 - x(1) + x(n-1))/2 + (1-omega)*x(n);
```

Nghiệm có thể thu được từ một lệnh như sau (chú ý rằng vectơ $x = 0$ là vectơ ban đầu)

>> [x, numIter, omega] = gaussSeidel(@fex3_6_3, zeros(20, 1))

ta có kết quả là

$x =$

¹Những phương trình dạng này được gọi là *chu kỳ ba đường chéo*. Nó xuất hiện trong công thức sai phân hữu hạn của phương trình vi phân bậc hai với bài toán biên tuần hoàn.

```

-4.5000e+000
-4.0000e+000
-3.5000e+000
-3.0000e+000
-2.5000e+000
-2.0000e+000
-1.5000e+000
-1.0000e+000
-5.0000e-001
2.1405e-009
5.0000e-001
1.0000e+000
1.5000e+000
2.0000e+000
2.5000e+000
3.0000e+000
3.5000e+000
4.0000e+000
4.5000e+000
5.0000e+000

```

numIter =

259

omega =

1.7055e+000

Ta thấy rằng tốc độ hội tụ rất chậm bởi vì ma trận hệ số A không phải là ma trận chéo trội. Nếu ta thay thế các phần tử trên đường chéo bởi 4, ma trận A là ma trận chéo trội và tốc độ hội tụ sẽ cải thiện (22 vòng lặp).

□

3.6.4 PHƯƠNG PHÁP GRADIENT LIÊN HỢP

Xét bài toán, tìm vectơ \mathbf{x} cực tiểu hóa hàm vô hướng

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T \mathbf{A}\mathbf{x} - \frac{1}{2}\mathbf{b}^T \mathbf{x}, \quad (3.43)$$

với \mathbf{A} là ma trận *đối xứng, xác định dương*. Bởi vì $f(\mathbf{x})$ bị cực tiểu khi $\nabla f = \mathbf{A}\mathbf{x} - \mathbf{b}$ bằng không nên bài toán cực tiểu hóa tương đương với giải

$$\mathbf{A}\mathbf{x} = \mathbf{b}. \quad (3.44)$$

Phương pháp Gradient thực hiện bài toán cực tiểu hóa bằng phương pháp lặp với vectơ ban đầu $\mathbf{x}^{(0)}$. Vòng lặp thứ k được thực hiện tính toán theo công thức

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{s}^{(k)}. \quad (3.45)$$

Chiều dài bước α_k được chọn sao cho $\mathbf{x}^{(k+1)}$ làm cực tiểu hóa hàm $f(\mathbf{x}^{(k+1)})$ theo *hướng tìm kiếm* $\mathbf{s}^{(k)}$. Có nghĩa là $\mathbf{x}^{(k+1)}$ phải thoả mãn phương trình (3.44):

$$\mathbf{A}(\mathbf{x}^{(k)} + \alpha_k \mathbf{s}^{(k)}) = \mathbf{b}. \quad (\text{a})$$

Nếu *thặng dư* thứ k được tính bởi

$$\mathbf{r}^{(k)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k)} \quad (3.46)$$

thì phương trình (a) trở thành $\alpha_k \mathbf{A}\mathbf{s}^{(k)} = \mathbf{r}^{(k)}$. Nhân bên trái hai vế với $(\mathbf{s}^{(k)})^T$ và tính α_k ta có

$$\alpha_k = \frac{(\mathbf{s}^{(k)})^T \mathbf{r}^{(k)}}{(\mathbf{s}^{(k)})^T \mathbf{A}\mathbf{s}^{(k)}}. \quad (3.47)$$

Chúng ta vẫn còn vấn đề là xác định hướng tìm kiếm $\mathbf{s}^{(k)}$. Trực giác nói với ta nên chọn $\mathbf{s}^{(k)} = -\nabla f = \mathbf{r}^{(k)}$ vì nó là hướng thay đổi không âm lớn nhất trong $f(x)$. Kết quả của quá trình này được biết đến với tên gọi *phương pháp dốc nhất*. Nó không phải là phương pháp phổ biến vì tốc độ hội tụ chậm. Phương pháp Gradient liên hợp hiệu quả hơn khi sử dụng hướng tìm kiếm

$$\mathbf{s}^{(k+1)} = \mathbf{r}^{(k+1)} + \beta_k \mathbf{s}^{(k)}. \quad (3.48)$$

Hằng số β_k được chọn sao cho hai hướng tìm kiếm liên tiếp là liên hợp với nhau, tức là $(\mathbf{s}^{(k+1)})^T \mathbf{A}\mathbf{s}^{(k)} = 0$. Thay $\mathbf{s}^{(k+1)}$ từ (3.48) ta có $[(\mathbf{r}^{(k+1)})^T + \beta_k (\mathbf{s}^{(k)})^T] \mathbf{A}\mathbf{s}^{(k)} = 0$, do đó

$$\beta_k = -\frac{(\mathbf{r}^{(k+1)})^T \mathbf{A}\mathbf{s}^{(k)}}{(\mathbf{s}^{(k)})^T \mathbf{A}\mathbf{s}^{(k)}}. \quad (3.49)$$

Do đó, ta có thể phác họa thuật toán Gradient liên hợp như sau

- Chọn $\mathbf{x}^{(0)}$ (ta có thể chọn tuỳ ý nhưng nếu gần nghiệm thì số lần lặp sẽ ít đi).
- $\mathbf{r}^{(0)} := \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$.
- $\mathbf{s}^{(0)} := \mathbf{r}^{(0)}$ (ở bước đầu tiên, ta chọn hướng tìm kiếm là hướng dốc nhất).
- thực hiện với $k = 0, 1, 2, \dots$

$$\alpha_k = \frac{(\mathbf{s}^{(k)})^T \mathbf{r}^{(k)}}{(\mathbf{s}^{(k)})^T \mathbf{A}\mathbf{s}^{(k)}}.$$

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{s}^{(k)}.$$

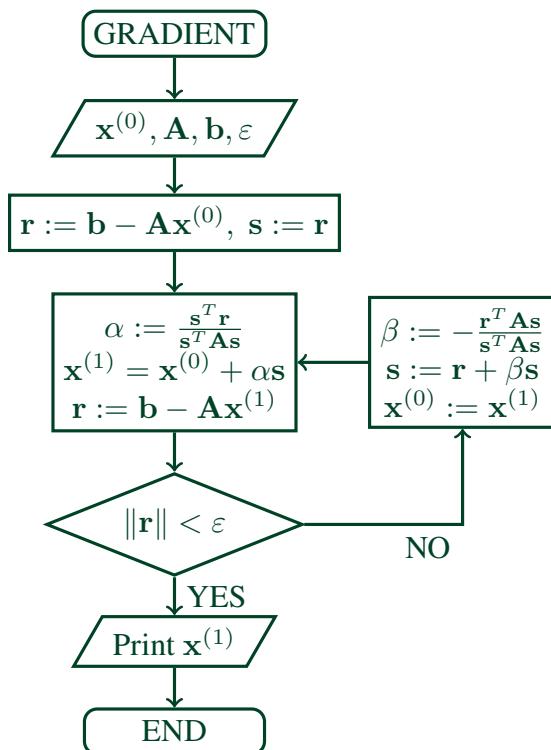
$$\mathbf{r}^{(k+1)} := \mathbf{b} - \mathbf{A}\mathbf{x}^{(k+1)}.$$

Nếu $|\mathbf{r}^{(k+1)}| \leq \varepsilon$ thì dừng (tiêu chuẩn hội tụ; ε là sai số cho phép).

$$\beta_k = -\frac{(\mathbf{r}^{(k+1)})^T \mathbf{A}\mathbf{s}^{(k)}}{(\mathbf{s}^{(k)})^T \mathbf{A}\mathbf{s}^{(k)}}.$$

$$\mathbf{s}^{(k+1)} := \mathbf{r}^{(k+1)} + \beta_k \mathbf{s}^{(k)}.$$

Quá trình thực hiện phương pháp Gradient liên hợp để giải hệ phương trình tuyến tính được minh họa trong Hình 3.13.



Hình 3.13: Sơ đồ khôi của phương pháp gradient liên hợp giải hệ phương trình tuyến tính

Có thể chỉ ra rằng các vectơ thặng dư $\mathbf{r}^{(1)}, \mathbf{r}^{(2)}, \mathbf{r}^{(3)}, \dots$ tạo ra từ thuật toán có tính trực giao, nghĩa là $\mathbf{r}^{(i)} \cdot \mathbf{r}^{(j)} = 0$. Bây giờ ta giả sử rằng đã thực hiện đủ số vòng lặp

phải tính để được tập n vectơ thặng dư. vectơ thặng dư cho lần lặp kế tiếp phải là vectơ không ($\mathbf{r}^{(k+1)} = \mathbf{0}$). Điều này chỉ ra nghiệm đã được tính toán. Nó cũng cho ta thấy phương pháp Gradient liên hợp không chỉ là phương pháp lặp mà nó còn cho ta nghiệm chính xác sau n vòng lặp. Tuy nhiên, trong thực tế, sự hội tụ cần ít vòng lặp hơn nhiều so với n .

Phương pháp Gradient liên hợp không cạnh tranh được với các phương pháp trực tiếp để tìm nghiệm trong một tập nhỏ các nghiệm. Nó thể hiện được sức mạnh khi giải các hệ thưa có kích thước lớn (là hệ mà đa số các phần tử của ma trận hệ số bằng không). Một chú ý quan trọng là ma trận A trong thuật toán chủ yếu liên quan đến các phép nhân với một vectơ, tức là dạng Av , với v là một vectơ ($x^{(k+1)}$ hoặc $s^{(k)}$). Nếu ma trận A thưa, chúng ta có thể viết một chương trình con hiệu quả cho phép nhân và thực hiện nó trong thuật toán Gradient liên hợp.

conjGrad.

Hàm `conjGrad` sau đây thực hiện thuật toán Gradient liên hợp. Số vòng lặp tối đa là n . Chú ý rằng hàm `conjGrad` gọi hàm `Av(v)`, là hàm trả kết quả là Av . Hàm này được nhập từ người sử dụng (xem VÍ DỤ 3.6.5) Chúng ta bắt buộc phải nhập vào vectơ ban đầu x và vectơ hằng b .

```

function [x,numIter] = conjGrad(func,x,b,epsilon)
% Solves Ax=b by conjugate gradient method.
% USAGE: [x,numIter] = conjGrad(func,x,b,epsilon)
% INPUT:
% func = handle of function that returns the vector A*v
% x = starting solution vector
% b = constant vector in A*x = b
% epsilon = error tolerance (default = 1.0e-9)
% OUTPUT:
% x = solution vector
% numIter = number of iterations carried out
if nargin == 3; epsilon = 1.0e-9; end
n = length(b);
r=b-feval(func,x);s=r;
for numIter = 1:n
    u = feval(func,s);
    alpha = dot(s,r)/dot(s,u);
    x=x+alpha*s;
    r=b-feval(func,x);
end

```

```

if sqrt(dot(r,r)) < epsilon
    return
else
    beta = -dot(r,u)/dot(s,u);
    s=r+beta*s;
end
end
error('Too many iterations')

```

VÍ DỤ 3.6.4. Giải bài toán trong VÍ DỤ 3.6.2 bằng phương pháp Gradient liên hợp.
GIẢI. Phương pháp gradient liên hợp hội tụ sau ba bước lặp. Chọn vectơ bắt đầu

$$\mathbf{x}^{(0)} = [0 \quad 0 \quad 0]^T$$

và tính toán theo các bước sau đây

$$\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{Ax}^{(0)} = \begin{bmatrix} 12 \\ -1 \\ 5 \end{bmatrix} - \begin{bmatrix} 4 & -1 & 1 \\ -1 & 4 & -2 \\ 1 & -2 & 4 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 12 \\ -1 \\ 5 \end{bmatrix}$$

$$\mathbf{s}^{(0)} = \mathbf{r}^{(0)} = \begin{bmatrix} 12 \\ -1 \\ 5 \end{bmatrix}$$

$$\mathbf{As}^{(0)} = \begin{bmatrix} 4 & -1 & 1 \\ -1 & 4 & -2 \\ 1 & -2 & 4 \end{bmatrix} \begin{bmatrix} 12 \\ -1 \\ 5 \end{bmatrix} = \begin{bmatrix} 54 \\ -26 \\ 34 \end{bmatrix}$$

$$\alpha_0 = \frac{(\mathbf{s}^{(0)})^T \mathbf{r}^{(0)}}{(\mathbf{s}^{(0)})^T \mathbf{As}^{(0)}} = \frac{12^2 + (-1)^2 + 5^2}{12(54) + (-1)(-26) + 5(34)} = 0.20142$$

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \alpha_0 \mathbf{x}^{(0)} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + 0.20142 \begin{bmatrix} 12 \\ -1 \\ 5 \end{bmatrix} = \begin{bmatrix} 2.41704 \\ -0.20142 \\ 1.00710 \end{bmatrix}$$

$$\mathbf{r}^{(1)} = \mathbf{b} - \mathbf{Ax}^{(1)} = \begin{bmatrix} 12 \\ -1 \\ 5 \end{bmatrix} - \begin{bmatrix} 4 & -1 & 1 \\ -1 & 4 & -2 \\ 1 & -2 & 4 \end{bmatrix} \begin{bmatrix} 2.41704 \\ -0.20142 \\ 1.00710 \end{bmatrix} = \begin{bmatrix} 1.12332 \\ 4.23692 \\ -1.84828 \end{bmatrix}$$

$$\beta_0 = -\frac{(\mathbf{r}^{(1)})^T \mathbf{A} \mathbf{s}^{(1)}}{(\mathbf{s}^{(1)})^T \mathbf{A} \mathbf{s}^{(1)}} = -\frac{1.12332(54) + 4.23692(-26) + (-1.84828)(34)}{12(54) + (-1)(-26) + 5(34)} = 0.133107$$

$$\mathbf{s}^{(1)} = \mathbf{r}^{(1)} + \beta_0 \mathbf{s}^{(0)} = \begin{bmatrix} 1.12332 \\ 4.23692 \\ -1.84828 \end{bmatrix} + 0.133107 \begin{bmatrix} 12 \\ -1 \\ 5 \end{bmatrix} = \begin{bmatrix} 2.72076 \\ 4.10380 \\ -1.18268 \end{bmatrix}$$

$$\mathbf{A} \mathbf{s}^{(1)} = \begin{bmatrix} 4 & -1 & 1 \\ -1 & 4 & -2 \\ 1 & -2 & 4 \end{bmatrix} \begin{bmatrix} 2.72076 \\ 4.10380 \\ -1.18268 \end{bmatrix} = \begin{bmatrix} 5.59656 \\ 16.05980 \\ -10.21760 \end{bmatrix}$$

$$\begin{aligned} \alpha_1 &= \frac{(\mathbf{s}^{(1)})^T \mathbf{r}^{(1)}}{(\mathbf{s}^{(1)})^T \mathbf{A} \mathbf{s}^{(1)}} \\ &= \frac{2.72076(1.12332) + 4.10380(4.23692) + (-1.18268)(-1.84828)}{2.72076(5.59656) + 4.10380(16.05980) + (-1.18268)(-10.21760)} \\ &= 0.24276 \end{aligned}$$

$$\mathbf{x}^{(2)} = \mathbf{x}^{(1)} + \alpha_1 \mathbf{s}^{(1)} = \begin{bmatrix} 2.41704 \\ -0.20142 \\ 1.00710 \end{bmatrix} + 0.24276 \begin{bmatrix} 2.72076 \\ 4.10380 \\ -1.18268 \end{bmatrix} = \begin{bmatrix} 3.07753 \\ 0.79482 \\ 0.71999 \end{bmatrix}$$

$$\mathbf{r}^{(2)} = \mathbf{b} - \mathbf{A} \mathbf{x}^{(2)} = \begin{bmatrix} 12 \\ -1 \\ 5 \end{bmatrix} - \begin{bmatrix} 4 & -1 & 1 \\ -1 & 4 & -2 \\ 1 & -2 & 4 \end{bmatrix} \begin{bmatrix} 3.07753 \\ 0.79482 \\ 0.71999 \end{bmatrix} = \begin{bmatrix} -0.23529 \\ 0.33823 \\ 0.63215 \end{bmatrix}$$

$$\begin{aligned} \beta_1 &= -\frac{(\mathbf{r}^{(2)})^T \mathbf{A} \mathbf{s}^{(1)}}{(\mathbf{s}^{(1)})^T \mathbf{A} \mathbf{s}^{(1)}} \\ &= -\frac{(-0.23529)(5.59656) + 0.33823(16.05980) + 0.63215(-10.21760)}{2.72076(5.59656) + 4.10380(16.05980) + (-1.18268)(-10.21760)} \\ &= 0.0251452 \end{aligned}$$

$$\mathbf{s}^{(2)} = \mathbf{r}^{(2)} + \beta_1 \mathbf{s}^{(1)} = \begin{bmatrix} -0.23529 \\ 0.33823 \\ 0.63215 \end{bmatrix} + 0.0251452 \begin{bmatrix} 2.72076 \\ 4.10380 \\ -1.18268 \end{bmatrix} = \begin{bmatrix} -0.166876 \\ 0.441421 \\ 0.602411 \end{bmatrix}$$

$$\mathbf{As}^{(2)} = \begin{bmatrix} 4 & -1 & 1 \\ -1 & 4 & -2 \\ 1 & -2 & 4 \end{bmatrix} \begin{bmatrix} -0.166876 \\ 0.441421 \\ 0.602411 \end{bmatrix} = \begin{bmatrix} -0.506514 \\ 0.727738 \\ 1.359930 \end{bmatrix}$$

$$\begin{aligned} \alpha_2 &= \frac{(\mathbf{s}^{(2)})^T \mathbf{r}^{(2)}}{(\mathbf{s}^{(2)})^T \mathbf{A} \mathbf{s}^{(2)}} \\ &= \frac{(-0.23529)(-0.166876) + 0.33823(0.441421) + 0.63215(0.602411)}{(-0.166876)(-0.506514) + 0.441421(0.727738) + 0.602411(1.359930)} \\ &= 0.46480 \end{aligned}$$

$$\mathbf{x}^{(3)} = \mathbf{x}^{(2)} + \alpha_2 \mathbf{s}^{(2)} = \begin{bmatrix} 3.07753 \\ 0.79482 \\ 0.71999 \end{bmatrix} + 0.46480 \begin{bmatrix} -0.166876 \\ 0.441421 \\ 0.602411 \end{bmatrix} = \begin{bmatrix} 2.99997 \\ 0.99999 \\ 0.99999 \end{bmatrix}.$$

Nghiệm $\mathbf{x}^{(3)}$ chính xác đến năm chữ số thập phân. Sự sai khác nhỏ này bị gây ra do việc làm tròn trong quá trình tính toán.

□

VÍ DỤ 3.6.5. Giải bài toán trong VÍ DỤ 3.6.3 bằng phương pháp Gradient liên hợp, cũng sử dụng $n = 20$.

GIẢI. Với ma trận \mathbf{A} đã cho, các thành phần của vectơ \mathbf{Av} là

$$\begin{aligned} (\mathbf{Av})_1 &= 2v_1 - v_2 + v_n \\ (\mathbf{Av})_i &= -v_{i-1} + 2v_i - v_{i+1}, \quad i = 2, 3, \dots, n-1 \\ (\mathbf{Av})_n &= -v_{n-1} + 2v_n + v_1 \end{aligned}$$

và được tính bởi chương trình sau

```
function Av = fex3_6_5(v)
% Computes the product A*v in Example 3.6.5
n = length(v);
Av = zeros(n,1);
Av(1) = 2*v(1) - v(2) + v(n);
Av(2:n-1) = -v(1:n-2) + 2*v(2:n-1) - v(3:n);
Av(n) = -v(n-1) + 2*v(n) + v(1);
```

Chương trình sau đây sử dụng hàm conjGrad. Trong chương trình, vectơ nghiệm \mathbf{x} được tìm với giá trị ban đầu là vectơ không.

```
>> % Example 3.6.5 (Conjugate gradient method)
n=20;
x = zeros(n,1);
b = zeros(n,1); b(n) = 1;
[x, numIter] = conjGrad(@fex3_6_5, x, b)
```

Kết quả sau khi chạy chương trình là

```
x =
-4.5000e+000
-4.0000e+000
-3.5000e+000
-3.0000e+000
-2.5000e+000
-2.0000e+000
-1.5000e+000
-1.0000e+000
-5.0000e-001
0
5.0000e-001
1.0000e+000
1.5000e+000
2.0000e+000
2.5000e+000
3.0000e+000
3.5000e+000
4.0000e+000
4.5000e+000
5.0000e+000
```

```
numIter =
10
```

□

3.7 MỘT SỐ HÀM MATLAB TÍNH TOÁN TRONG ĐẠI SỐ TUYẾN TÍNH

Một số hàm Matlab thông dụng dùng để tính toán trong đại số tuyến tính sẽ được giới thiệu sau đây.

- $x = A \setminus b$ tính nghiệm x của hệ phương trình $Ax = b$ bằng phương pháp khử Gauss. Nếu ma trận A có số hàng nhiều hơn số cột thì nghiệm có bình phương tối thiểu nhỏ nhất được tính.
- $[L, U] = lu(A)$ phân rã Doolittle $A = LU$, trong đó U là ma trận tam giác trên, L có chứa một hoán vị hàng của ma trận tam giác dưới.
- $[M, U, P] = lu(A)$ trả về ma trận tam giác trên U , ma trận tam giác dưới M và ma trận trực giao P thoả mãn $M = P^* L$. Chú ý rằng $P^* A = M^* U$.
- $L = chol(A)$ cho ma trận đường chéo L trong phân rã Cholesky $A = LL^T$.
- $B = inv(A)$ cho ma trận B là ma trận ngược của ma trận A .
- $n = norm(A, 1)$ cho n là chuẩn của ma trận A được tính theo công thức $n = \max_j \sum_i a_{ij}$ (tổng lớn nhất các phần tử trong các cột của ma trận A).
- $c = cond(A)$ trả về c là số điều kiện của ma trận A .

Matlab không hỗ trợ các ma trận dải một cách rõ ràng. Tuy nhiên, các ma trận dải có thể được xử lý như các ma trận thưa, mảng được MATLAB hỗ trợ rất rộng rãi. Một ma trận dải dưới dạng thưa có thể được tạo ra bởi lệnh sau.

$A = spdiags(B, d, n, n)$ tạo ra ma trận dải A có cỡ $n \times n$ từ các cột của ma trận B . Các đường chéo của ma trận A được xác định bởi vectơ d . Giả sử để tạo ra ma trận thưa A

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

ta thực hiện

```
>>c=ones(5,1);
>>A=spdiags([-c 2*c -c], [-1 0 1], 5, 5)
```

Với một ma trận thưa, MATLAB chỉ lưu trữ các phần tử khác không bằng cách lưu trữ vị trí và giá trị của mỗi phần tử này. Hầu hết các hàm liệt kê phía trên đều làm việc với ma trận thưa. Chẳng hạn $[L, U] = \text{lu}(A)$ sẽ cho các ma trận L và U dưới dạng thưa nếu ma trận A thưa. Có một số hàm cho ma trận thưa trong MATLAB như sau

$A = \text{full}(S)$	chuyển ma trận dải S sang ma trận đầy (full) A .
$S = \text{sparse}(A)$	chuyển ma trận đầy A sang ma trận dải S .
$x = \text{lsqr}(A, b)$	tính nghiệm x của hệ phương trình tuyến tính $Ax = b$ bằng phương pháp Gradient liên hợp.
$\text{spy}(S)$	vẽ đồ thị các phần tử khác không của ma trận S .

TÓM TẮT CHƯƠNG 3

Một số lớn các bài toán giải số được quy về việc tìm nghiệm của hệ phương trình đại số tuyến tính nên việc tìm kiếm các phương pháp giải hệ phương trình tuyến tính hiệu quả là rất cần thiết. Các phương pháp giải hệ phương trình tuyến tính được chia làm hai nhóm: nhóm các phương pháp trực tiếp, là các phương pháp nếu quá trình tính toán không có sai số thì sẽ cho ta nghiệm chính xác, và nhóm các phương pháp lặp (gián tiếp).

Có một số phương pháp giải trực tiếp nổi tiếng như phương pháp Cramer, phương pháp ma trận nghịch đảo, phương pháp khử Gauss và phương pháp phân rã LU. Do yêu cầu số lượng các phép tính lớn nên phương pháp Cramer và phương pháp ma trận nghịch đảo thường không được quan tâm trong giải số. Chương này trình bày chi tiết các phương pháp khử Gauss và phương pháp phân rã LU cũng như việc áp dụng đổi hàng xoay để cải tiến cho các phương pháp này. Các phương pháp trực tiếp đã giới thiệu bao gồm:

- Thuật toán khử Gauss;
- Thuật toán phân rã Doolittle;
- Thuật toán phân rã Cholesky;
- Thuật toán khử Gauss kết hợp đổi hàng xoay;
- Thuật toán phân rã LU kết hợp đổi hàng xoay.

Hệ phương trình có ma trận hệ số dải và đối xứng thường hay gấp trong các bài toán kỹ thuật. Việc lưu trữ và giải các hệ này cần được thiết kế riêng để tận dụng được những đặc điểm của hệ. Bằng cách cải tiến các phương pháp phân rã LU, các thuật toán được xây dựng để giải các hệ này gồm:

- Thuật toán phân rã LU giải hệ phương trình tuyến tính ba đường chéo;
- Thuật toán phân rã LU giải hệ phương trình tuyến tính năm đường chéo đối xứng.

Tuy tốc độ hội tụ chậm nhưng các phương pháp lặp đặc biệt hiệu quả khi áp dụng giải các hệ có tính chất thưa. Chương này giới thiệu ba phương pháp lặp nổi tiếng là:

- Phương pháp lặp Jacobi;
- Phương pháp lặp Gauss-Seidel;
- Phương pháp Gradient liên hợp.

BÀI TẬP CHƯƠNG 3

Bài tập 3.1. Giải hệ phương trình sau bằng phương pháp khử Gauss tính đến 3 chữ số thập phân

a.

$$\begin{cases} 2,75x_1 + 1,78x_2 + 1,11x_3 = 13,62 \\ 3,28x_1 + 0,71x_2 + 1,15x_3 = 17,98 \\ 1,15x_1 + 2,70x_2 + 3,58x_3 = 39,72 \end{cases}$$

b.

$$\begin{cases} 3,2x_1 - 1,5x_2 + 0,5x_3 = 0,9 \\ 1,6x_1 + 2,5x_2 + x_3 = 1,55 \\ x_1 + 4,1x_2 - 1,5x_3 = 2,08 \end{cases}$$

c.

$$\begin{cases} 1,5x_1 - 0,2x_2 + 0,1x_3 = 0,4 \\ -0,1x_1 + 1,5x_2 - 0,1x_3 = 0,8 \\ -0,3x_1 + 0,2x_2 - 0,5x_3 = 0,2 \end{cases}$$

Bài tập 3.2. Giải hệ phương trình $\mathbf{Ax} = \mathbf{b}$ bằng phương pháp khử Gauss với phép tính lũy thừa 3 chữ số thập phân

a.

$$\mathbf{A} = \begin{bmatrix} 1,5 & 1,4 & 1,3 & 1,2 \\ 1,4 & 1,6 & 1,2 & 1,3 \\ 1,3 & 1,2 & 1,7 & 1,4 \\ 1,2 & 1,3 & 1,4 & 1,5 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 5,80 \\ 5,90 \\ 6,06 \\ 5,81 \end{bmatrix}$$

b.

$$\mathbf{A} = \begin{bmatrix} 4,1 & 1,0 & 1,1 & 1,2 \\ 1,0 & 5,1 & 1,2 & 1,1 \\ 1,1 & 1,2 & 4,1 & 1,0 \\ 1,2 & 1,1 & 1,0 & 5,1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 2,3 \\ 2,3 \\ 5,1 \\ 6,1 \end{bmatrix}$$

Bài tập 3.3. Cho trước phân rã LU của $\mathbf{A} = \mathbf{LU}$. Hãy xác định \mathbf{A} và tính $|\mathbf{A}|$.

a.

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 5/3 & 1 \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} 1 & 2 & 4 \\ 0 & 3 & 21 \\ 0 & 0 & 0 \end{bmatrix}$$

b.

$$\mathbf{L} = \begin{bmatrix} 2 & 0 & 0 \\ 11 & 1 & 0 \\ 1 & -3 & 1 \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} 2 & -1 & 1 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{bmatrix}$$

Bài tập 3.4. Sử dụng kết quả phân rã LU

$$\mathbf{A} = \mathbf{LU} = \begin{bmatrix} 1 & 0 & 0 \\ 3/2 & 1 & 0 \\ 1/2 & 11/13 & 1 \end{bmatrix} \begin{bmatrix} 2 & -3 & -1 \\ 0 & 13/2 & -7/2 \\ 0 & 0 & 32/13 \end{bmatrix}$$

để giải phương trình $\mathbf{Ax} = \mathbf{b}$, với $\mathbf{b} = [1 \ -1 \ 2]$.

Bài tập 3.5. Giải phương trình $\mathbf{A} = \mathbf{x} = \mathbf{b}$ bằng phương pháp khử Gauss, với

a.

$$\mathbf{A} = \begin{bmatrix} 2 & -3 & -1 \\ 3 & 2 & -5 \\ 2 & 4 & -1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 3 \\ -9 \\ -5 \end{bmatrix}$$

b.

$$\mathbf{A} = \begin{bmatrix} 2 & 0 & -1 & 0 \\ 0 & 1 & 2 & 0 \\ -1 & 2 & 0 & 1 \\ 0 & 0 & 1 & -2 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

c.

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 2 & 1 & 2 \\ 0 & 1 & 0 & 2 & -1 \\ 1 & 2 & 0 & -2 & 0 \\ 0 & 0 & 0 & -1 & 1 \\ 0 & 1 & -1 & 1 & -1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1 \\ 1 \\ -4 \\ -2 \\ -1 \end{bmatrix}$$

Bài tập 3.6. Tìm ma trận L và U sao cho

$$\mathbf{LU} = \mathbf{A} = \begin{bmatrix} 4 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 4 \end{bmatrix}$$

bằng cách sử dụng

a. phương pháp phân rã Doolittle;

b. phương pháp phân rã Cholesky.

Bài tập 3.7. Giải phương trình $\mathbf{A} = \mathbf{x} = \mathbf{b}$ bằng phương pháp phân rã Doolittle, với

a.

$$\mathbf{A} = \begin{bmatrix} -3 & 6 & -4 \\ 9 & -8 & 24 \\ -12 & 24 & -26 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} -3 \\ 65 \\ -42 \end{bmatrix}$$

b.

$$\mathbf{A} = \begin{bmatrix} 2,34 & -4,10 & 1,78 \\ -1,98 & 3,47 & -2,22 \\ 2,36 & -15,17 & 6,18 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 0,02 \\ -0,73 \\ -6,63 \end{bmatrix}$$

c.

$$\mathbf{A} = \begin{bmatrix} 4 & -3 & 6 \\ 8 & -3 & 10 \\ -4 & 12 & 10 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

Bài tập 3.8. Giải phương trình $\mathbf{Ax} = \mathbf{b}$ bằng phương pháp phân rã Cholesky, với

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1 \\ 3/2 \\ 3 \end{bmatrix}$$

Bài tập 3.9. Tính ma trận L bằng phương pháp phân rã Cholesky ma trận đường chéo

$$\mathbf{A} = \begin{bmatrix} \alpha_1 & 0 & 0 & \cdots \\ 0 & \alpha_2 & 0 & \cdots \\ 0 & 0 & \alpha_3 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Bài tập 3.10. Sửa lại hàm gauss.m để giải hệ phương trình tuyến tính với vế phải gồm m vectơ. Kiểm tra chương trình bằng cách giải hệ $\mathbf{AX} = \mathbf{B}$ với

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Bài tập 3.11. Viết một function cho pha giải của phương pháp phân rã Cholesky. Kiểm tra function vừa viết bằng cách giải hệ $\mathbf{Ax} = \mathbf{b}$ với

$$\mathbf{A} = \begin{bmatrix} 4 & -2 & 2 \\ -2 & 2 & -4 \\ 2 & -4 & 11 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} 6 \\ -10 \\ 27 \end{bmatrix}$$

Bài tập 3.12. Lập trình tính hệ số của đa thức $y = a_0 + a_1x + a_2x^2 + a_3x^3$ biết đồ thị của đa thức này đi qua các điểm $(0, 10), (1, 35), (3, 31)$ và $(4, 2)$.

Bài tập 3.13. Lập trình tìm đa thức bậc 4 biết đồ thị của đa thức này đi qua các điểm $(0, -1), (1, 1), (3, 3), (5, 2)$ và $(6, -2)$.

Bài tập 3.14. Xác định ma trận L, D là kết quả của phương pháp phân rã Doolittle ma trận

$$\mathbf{A} = \begin{bmatrix} 2 & -2 & 0 & 0 & 0 \\ -2 & 5 & -6 & 0 & 0 \\ 0 & -6 & 16 & 12 & 0 \\ 0 & 0 & 12 & 39 & -6 \\ 0 & 0 & 0 & -6 & 14 \end{bmatrix}$$

Bài tập 3.15. Giải hệ phương trình ba đường chéo $\mathbf{Ax} = \mathbf{b}$ bằng phương pháp phân rã Doolittle, với

$$\mathbf{A} = \begin{bmatrix} 6 & 2 & 0 & 0 & 0 \\ -1 & 7 & 2 & 0 & 0 \\ 0 & -2 & 8 & 2 & 0 \\ 0 & 0 & 3 & 7 & -2 \\ 0 & 0 & 0 & 3 & 5 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 2 \\ -3 \\ 4 \\ -3 \\ 1 \end{bmatrix}$$

Bài tập 3.16. Sử dụng phương pháp khử Gauss kết hợp phần tử trội giải hệ phương trình $\mathbf{Ax} = \mathbf{b}$, với

a.

$$\mathbf{A} = \begin{bmatrix} 4 & -2 & 1 \\ -2 & 1 & -1 \\ -2 & 3 & 6 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 2 \\ -1 \\ 0 \end{bmatrix}$$

b.

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & -1 & 2 & -1 \\ -1 & 2 & -1 & 0 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Bài tập 3.17. Lập trình giải hệ phương trình ba đường chéo đối xứng sau

$$\begin{cases} 4x_1 - x_2 &= 9 \\ -x_{i-1} + 4x_i - x_{i+1} &= 5, \quad i = 2, 3, \dots, n-1 \\ -x_{n-1} + 4x_n &= 5 \end{cases}$$

Bài tập 3.18. Sử dụng phương pháp lặp Gauss-Seidel giải hệ phương trình $\mathbf{Ax} = \mathbf{b}$, với

a.

$$\mathbf{A} = \begin{bmatrix} -2 & 5 & 9 \\ 7 & 1 & 1 \\ -3 & 7 & -1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1 \\ 6 \\ -26 \end{bmatrix}$$

b.

$$\mathbf{A} = \begin{bmatrix} 12 & -2 & 3 & 1 \\ -2 & 15 & 6 & -3 \\ 1 & 6 & 20 & -4 \\ 0 & -3 & 2 & 9 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ 20 \\ 0 \end{bmatrix}$$

Bài tập 3.19. Cho

$$\mathbf{A} = \begin{bmatrix} 42 & -1 & 0 & 0 \\ -1 & 4 & -1 & 0 \\ 0 & -1 & 4 & -1 \\ -1 & 0 & -1 & 4 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} -4 \\ 2 \\ 4 \\ 10 \end{bmatrix}$$

- a. Chứng minh rằng nghiệm của hệ phương trình $\mathbf{Ax} = \mathbf{b}$ là $\mathbf{x} = (0 \ 1 \ 2 \ 3)^T$.
- b. Sử dụng 3 vòng lặp của phương pháp lặp Jacobi để xấp xỉ nghiệm của hệ phương trình $\mathbf{Ax} = \mathbf{b}$ với $\mathbf{x}^{(0)} = (0 \ 0 \ 0 \ 0)^T$.

Bài tập 3.20. Tính nghiệm xấp xỉ của hệ phương trình sau bằng phương pháp lặp đơn sau ba vòng lặp. Tính sai số của nghiệm xấp xỉ.

$$\begin{cases} 1,02x_1 - 0,05x_2 - 0,10x_3 = 0,795 \\ -0,11x_1 + 1,03x_2 - 0,05x_3 = 0,849 \\ -0,11x_1 - 0,12x_2 + 1,04x_3 = 1,398 \end{cases}$$

Bài tập 3.21. Tính nghiệm xấp xỉ của hệ phương trình $\mathbf{Ax} = \mathbf{b}$ với các ma trận hệ số, giá trị ban đầu và sai số cho bởi

a.

$$\mathbf{A} = \begin{bmatrix} 1,02 & -0,25 & -0,30 \\ -0,41 & 1,13 & -0,15 \\ -0,25 & -0,14 & 1,21 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 0,515 \\ 1,555 \\ 2,780 \end{bmatrix}, \quad \mathbf{x}^{(0)} = \begin{bmatrix} 2,0 \\ 2,5 \\ 3,0 \end{bmatrix}, \quad \varepsilon = 10^{-3}.$$

b.

$$\mathbf{A} = \begin{bmatrix} 10,9 & 1,2 & 2,1 & 0,9 \\ 1,2 & 11,2 & 1,5 & 2,5 \\ 2,1 & 1,5 & 9,8 & 1,3 \\ 0,9 & 2,5 & 1,3 & 12,1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} -7,0 \\ 5,3 \\ 17,3 \\ 24,6 \end{bmatrix}, \quad \mathbf{x}^{(0)} = \begin{bmatrix} -1 \\ 0 \\ 1 \\ 2 \end{bmatrix}, \quad \varepsilon = 10^{-3}.$$

CHƯƠNG 4

NỘI SUY VÀ PHƯƠNG PHÁP BÌNH PHƯƠNG NHỎ NHẤT

4.1	Giới thiệu	134
4.2	Nội suy đa thức	135
4.2.1	Đa thức nội suy Lagrange	135
4.2.2	Đa thức nội suy Newton	137
4.2.3	Phương pháp nội suy Neville	143
4.3	Phương pháp bình phương nhỏ nhất	147
4.3.1	Mở đầu	147
4.3.2	Phương trình hồi quy tuyến tính	149
4.3.3	Xấp xỉ dạng tuyến tính	151
4.3.4	Xấp xỉ dạng đa thức	151
4.3.5	Dữ liệu có trọng số	157
4.4	HÀM MATLAB XẤP XỈ HÀM SỐ	161

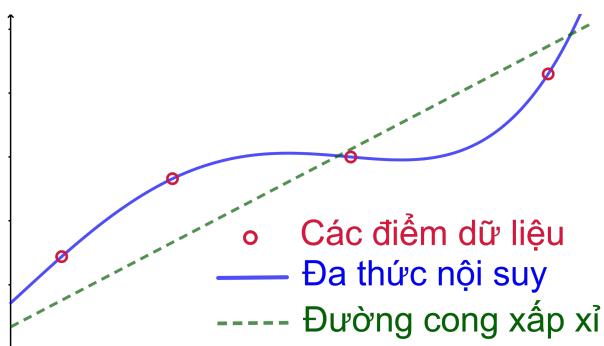
4.1 GIỚI THIỆU

Dữ liệu thường được cho dưới dạng các giá trị rời rạc đọc theo một hướng nào đó. Tuy nhiên, nhiều bài toán thực tế lại yêu cầu ước lượng giá trị tại các điểm nằm giữa các giá trị rời rạc này. Chương này sẽ trình bày các kỹ thuật để xấp xỉ đường cong với các dữ liệu này để có một ước lượng trung gian. Bên cạnh đó, nhiều khi việc xấp xỉ các bài toán trong thực tế dẫn đến một hàm số phức tạp mà việc tính toán khó khăn, ta có thể xấp xỉ một hàm số phức tạp này bằng một hàm số đơn giản hơn. Để làm điều này, ta tính toán giá trị hàm số tại một số điểm rời rạc đọc theo khoảng giá trị mà ta quan tâm. Sau đó, một hàm đơn giản hơn được xấp xỉ dựa vào các giá trị dữ liệu rời rạc này. Các dữ liệu thu được thường được cho dưới dạng bảng:

x_1	x_2	x_3	\dots	x_n
y_1	y_2	y_3	\dots	y_n

Dựa vào độ chính xác và phân tán của dữ liệu, có hai hướng tiếp cận để giải quyết các vấn đề nêu trên. Đầu tiên, khi dữ liệu đã biết có độ chính xác cao, cách tiếp cận là sử dụng một hoặc một dãy các đường cong đi qua các điểm dữ liệu này. Phương pháp nổi tiếng để ước lượng đường cong đi qua các điểm là *phương pháp nội suy*. Trong phạm vi cuốn sách này, phương pháp nội suy đa thức sẽ được trình bày trong Mục 4.2.

Tiếp theo, khi dữ liệu có nhiều lớn hoặc dữ liệu có xu hướng không phân tán thì chiến lược đưa ra là sử dụng một đường cong duy nhất biểu thị xu hướng chung của dữ liệu. Bởi vì có nhiều nên bất kỳ điểm dữ liệu riêng lẻ nào cũng có thể không chính xác, do đó việc cố gắng tìm đường cong đi qua điểm này là không cần thiết. Thay vào đó, đường cong được thiết kế theo hình dạng của nhóm các điểm dữ liệu. Cách tiếp cận này được gọi là *phương pháp bình phương nhỏ nhất* sẽ được trình bày trong Mục 4.3. Hình 4.1 minh họa phương pháp nội suy và phương pháp bình phương nhỏ nhất.



Hình 4.1: Đồ thị minh họa hai phương pháp xấp xỉ dữ liệu: phương pháp nội suy và xấp xỉ đường cong

4.2 NỘI SUY ĐA THỨC

Nội suy đa thức là trường hợp phổ biến nhất của phương pháp nội suy. Với n điểm dữ liệu cho trước, luôn tồn tại duy nhất đa thức bậc $n - 1$ đi qua các điểm dữ liệu này. Chẳng hạn, với hai điểm dữ liệu thì ta luôn xác định được một đường thẳng (đa thức bậc một) qua hai điểm này; với ba điểm dữ liệu thì tồn tại một parabola (đa thức bậc hai) đi qua ba điểm này. Mục này sẽ giới thiệu một số cách xây dựng đa thức nội suy đi qua n điểm dữ liệu.

4.2.1 ĐA THỨC NỘI SUY LAGRANGE

Có nhiều cách xây dựng đa thức nội suy bậc $n - 1$ qua n điểm dữ liệu, một trong những cách này là *công thức Lagrange*

$$P_{n-1}(x) = \sum_{i=1}^n y_i l_i(x), \quad (4.1)$$

với

$$\begin{aligned} l_i(x) &= \frac{x - x_1}{x_i - x_1} \cdot \frac{x - x_2}{x_i - x_2} \cdots \frac{x - x_{i-1}}{x_i - x_{i-1}} \cdot \frac{x - x_{i+1}}{x_i - x_{i+1}} \cdots \frac{x - x_n}{x_i - x_n} \\ &= \prod_{\substack{j=1 \\ j \neq i}}^n \frac{x - x_j}{x_i - x_j} \end{aligned} \quad (4.2)$$

được gọi là *các hàm cơ bản*.

Chẳng hạn, nếu $n = 2$, đa thức nội suy là đường thẳng $P_1(x) = y_1 l_1(x) + y_2 l_2(x)$, với

$$l_1(x) = \frac{x - x_2}{x_1 - x_2}, \quad l_2(x) = \frac{x - x_1}{x_2 - x_1}.$$

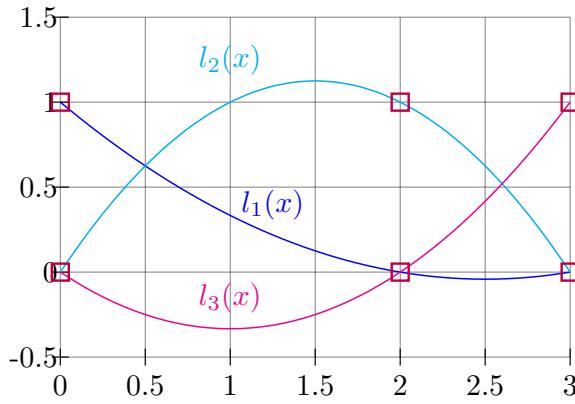
Với $n = 3$, đa thức nội suy là parabolic $P_2 = y_1 l_1(x) + y_2 l_2(x) + y_3 l_3(x)$, với

$$\begin{aligned} l_1(x) &= \frac{(x - x_2)(x - x_3)}{(x_1 - x_2)(x_1 - x_3)}; \\ l_2(x) &= \frac{(x - x_1)(x - x_3)}{(x_2 - x_1)(x_2 - x_3)}; \\ l_3(x) &= \frac{(x - x_1)(x - x_2)}{(x_3 - x_1)(x_3 - x_2)}. \end{aligned}$$

Các hàm cơ bản là đa thức bậc $n - 1$ và có tính chất

$$l_i(x_j) = \begin{cases} 0 & \text{khi } i \neq j \\ 1 & \text{khi } i = j \end{cases} = \delta_{ij}, \quad (4.3)$$

với δ_{ij} là hàm *delta Kronecker*. Tính chất này được minh họa trong Hình 4.2 cho trường hợp ba điểm nội suy ($n = 3$) với $x_1 = 0, x_2 = 2$ và $x_3 = 3$.



Hình 4.2: Hình minh họa về các hàm cơ bản bậc 2 để tính đa thức nội suy Lagrange.

Để chỉ ra đa thức nội suy đi qua các điểm dữ liệu, ta thay $x = x_j$ vào (4.1). Từ (4.3) ta có

$$P_{n-1}(x_j) = \sum_{i=1}^n y_i l_i(x_j) = \sum_{i=1}^n y_i \delta_{ij} = y_i.$$

Ta có thể chỉ ra hàm sai số của đa thức nội suy là

$$f(x) - P_{n-1}(x) = \frac{(x - x_1)(x - x_2) \dots (x - x_n)}{n!} f^{(n)}(\xi) \quad (4.4)$$

với ξ là một hằng số nằm trong khoảng (x_1, x_n) . Từ công thức của hàm sai số, lưu ý rằng nếu giá trị xấp xỉ x càng xa khoảng dữ liệu thì sai số càng lớn.

Quá trình tính đa thức nội suy Lagrange từ n điểm dữ liệu được minh họa trong Hình 4.3.

VÍ DỤ 4.2.1. Cho các điểm dữ liệu

x	0	2	3
y	7	11	28

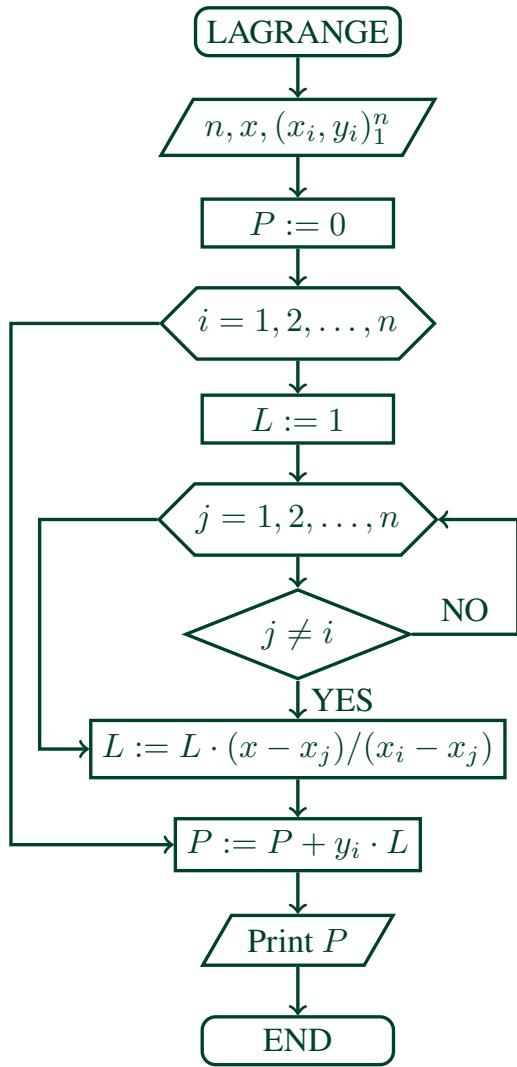
sử dụng phương pháp Lagrange xác định y tại $x = 1$.

GIẢI.

$$\begin{aligned} l_1(1) &= \frac{(x - x_2)(x - x_3)}{(x_1 - x_2)(x_1 - x_3)} = \frac{(1 - 2)(1 - 3)}{(0 - 2)(0 - 3)} = \frac{1}{3} \\ l_2(1) &= \frac{(x - x_1)(x - x_3)}{(x_2 - x_1)(x_2 - x_3)} = \frac{(1 - 0)(1 - 3)}{(2 - 0)(2 - 3)} = 1 \\ l_3(1) &= \frac{(x - x_1)(x - x_2)}{(x_3 - x_1)(x_3 - x_2)} = \frac{(1 - 0)(1 - 2)}{(3 - 0)(3 - 2)} = -\frac{1}{3} \end{aligned}$$

$$y(1) = y_1 l_1(1) + y_2 l_2(1) + y_3 l_3(1) = \frac{7}{3} + 11 - \frac{28}{3} = 4.$$

□



Hình 4.3: Sơ đồ khôi tính đa thức nội suy Lagrange qua n điểm dữ liệu cho trước

4.2.2 ĐA THỨC NỘI SUY NEWTON

Mặc dù phương pháp nội suy Lagrange có ưu điểm là đơn giản nhưng khi thêm mốc nội suy thì đa thức nội suy phải tính lại từ đầu. Một phương pháp cải tiến xây dựng được đa thức nội suy khắc phục được nhược điểm này gọi là *đa thức nội suy Newton*, có công thức như sau

$$P_{n-1}(x) = a_1 + (x - x_1)a_2 + (x - x_1)(x - x_2)a_3 + \dots + (x - x_1)(x - x_2)\dots(x - x_{n-1})a_n.$$

Chẳng hạn, với bốn điểm dữ liệu $n = 4$, ta có đa thức nội suy bậc 3 là

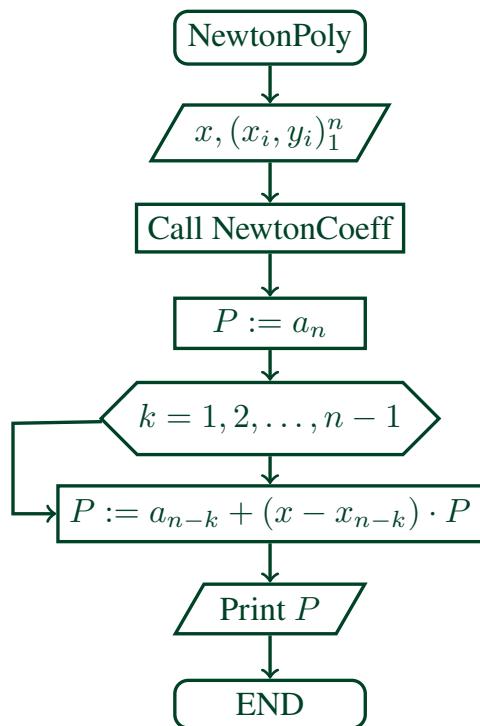
$$\begin{aligned}
 P_3(x) &= a_1 + (x - x_1)a_2 + (x - x_1)(x - x_2)a_3 + (x - x_1)(x - x_2)(x - x_3)a_4 \\
 &= a_1 + (x - x_1) \{a_2 + (x - x_2)[a_3 + (x - x_3)a_4]\}.
 \end{aligned}$$

Ta có thể tính $P_3(x)$ bằng cách thay thế ngược

$$\begin{aligned}P_0(x) &= a_4 \\P_1(x) &= a_3 + (x - x_3)P_0(x) \\P_2(x) &= a_2 + (x - x_2)P_1(x) \\P_3(x) &= a_1 + (x - x_1)P_2(x).\end{aligned}$$

Với n tùy ý, ta có

$$P_0(x) = a_n, \quad P_k(x) = a_{n-k} + (x - x_{n-k})P_{k-1}(x), \quad k = 1, 2, \dots, n-1. \quad (4.5)$$



**Hình 4.4: Sơ đồ khôi
tính đa thức nội suy
Newton qua n điểm
dữ liệu cho trước**

newtonPoly.

Chương trình MATLAB sau tính toán đa thức nội suy Newton trong đó, các mốc nội suy được cho trong vectơ $xData$ và hệ số a_i của đa thức cho trong vectơ a (sẽ được tính toán bằng hàm `newtonCoeff`).

```

function p=newtonPoly(a,xData,x)
% Returns value of Newton's polynomial at x.
% USAGE:p=newtonPoly(a,xData,x)
% a = coefficient array of the polynomial;
% must be computed first by newtonCoeff.
% xData = x-coordinates of data points.
  
```

```

n = length(xData);
p = a(n);
for k=1:n-1;
    p = a(n-k) + (x - xData(n-k)) * p;
end

```

Tính các hệ số của đa thức nội suy. Các hệ số của đa thức nội suy $P_{n-1}(x)$ được tính bằng cách cho đa thức nội suy đi qua các điểm dữ liệu, tức là $y_i = P_{n-1}(x_i)$, $i = 1, 2, \dots, n$. Từ đó ta suy ra

$$\begin{aligned}
y_1 &= a_1 \\
y_2 &= a_1 + (x_2 - x_1)a_2 \\
y_3 &= a_1 + (x_3 - x_1)a_2 + (x_3 - x_1)(x_3 - x_2)a_3 \\
&\vdots \\
y_n &= a_1 + (x_3 - x_1)a_2 + \dots + (x_n - x_1)(x_n - x_2)\dots(x_n - x_{n-1})a_n
\end{aligned} \quad . \quad (a)$$

Áp dụng công thức *tỉ sai phân*

$$\begin{aligned}
\nabla y_i &= \frac{y_i - y_1}{x_i - x_1}, \quad i = 2, 3, \dots, n \\
\nabla^2 y_i &= \frac{\nabla y_i - \nabla y_2}{x_i - x_2}, \quad i = 3, 4, \dots, n \\
\nabla^3 y_i &= \frac{\nabla^2 y_i - \nabla^2 y_3}{x_i - x_3}, \quad i = 4, 5, \dots, n \\
&\vdots \\
\nabla^{n-1} y_n &= \frac{\nabla^{n-2} y_n - \nabla^{n-2} y_{n-1}}{x_n - x_{n-1}}
\end{aligned} \quad (4.6)$$

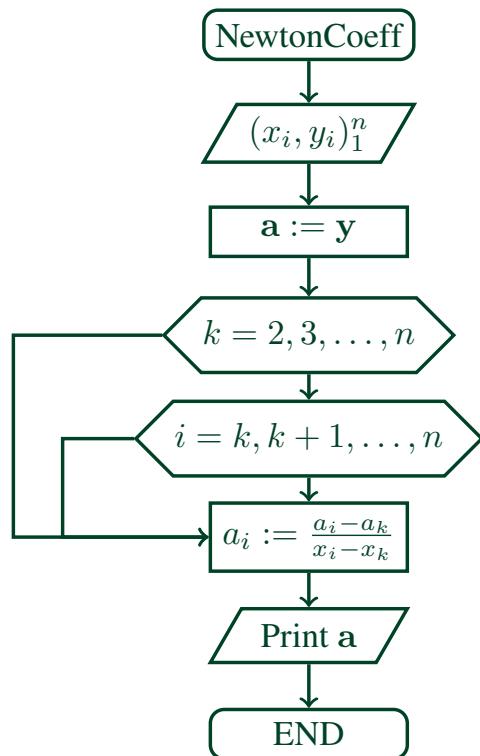
nghiệm của hệ phương trình (a) là

$$a_1 = y_1, \quad a_2 = \nabla y_2, \quad a_3 = \nabla^2 y_3, \dots, a_n = \nabla^{n-1} y_n. \quad (4.7)$$

Để thuận lợi cho việc tính toán thủ công các hệ số, ta thực hiện theo Bảng 4.1 (với $n = 5$).

Bộ số $(y_1, \nabla y_2, \nabla^2 y_3, \nabla^3 y_4, \nabla^4 y_5)$ trong bảng là các hệ số của đa thức nội suy Newton. Nếu thứ tự các điểm dữ liệu thay đổi thì các hệ số này sẽ thay đổi nhưng đa thức nội suy bậc $n - 1$ thu được là duy nhất. Việc tính toán hệ số a của đa thức nội suy Newton sẽ được thực hiện bởi hàm MATLAB sau, quá trình tính toán tốt nhất trong trường hợp a là mảng một chiều.

 newtonCoeff.



Hình 4.5: Sơ đồ khôi tính hệ số của đa thức nội suy Newton qua n điểm dữ liệu cho trước

x_1	y_1				
x_2	y_2	∇y_2			
x_3	y_3	∇y_3	$\nabla^2 y_3$		
x_4	y_4	∇y_4	$\nabla^2 y_4$	$\nabla^3 y_4$	
x_5	y_5	∇y_5	$\nabla^2 y_5$	$\nabla^3 y_5$	$\nabla^4 y_5$

Bảng 4.1: Bảng tính toán hệ số cho đa thức nội suy Newton với $n = 5$.

```

function a=newtonCoeff(xData,yData)
  % Returns coefficients of Newton's polynomial.
  % USAGE:a=newtonCoeff(xData,yData)
  % xData = x-coordinates of data points.
  % yData = y-coordinates of data points.
  n = length(xData);
  a = yData;
  for k=2:n
    a(k:n) = (a(k:n) - a(k-1))./(xData(k:n) - xData(k-1));
  end

```

Ban đầu, a nhận các giá trị y của dữ liệu, là cột thứ hai trong Bảng 4.1. Sau mỗi vòng lặp, các cột tương ứng trong Bảng 4.1 lần lượt được tính và gán các phần tử tương ứng của a. Do đó, a chính là các phần tử đường chéo của Bảng 4.1.

VÍ DỤ 4.2.2. Các điểm dữ liệu

x	-2	1	4	-1	3	-4
y	-1	2	59	4	24	-53

nằm trên một đa thức. Xác định bậc của đa thức này bằng cách xây dựng bảng sai phân tương tự Bảng 4.1.

GIẢI. Ta có

i	x_i	y_i	∇y_i	$\nabla^2 y_i$	$\nabla^3 y_i$	$\nabla^4 y_i$	$\nabla^5 y_i$
1	-2	-1					
2	1	2	1				
3	4	59	10	3			
4	-1	4	5	-2	1		
5	3	24	5	2	1	0	
6	-4	-53	26	-5	1	0	0

Sau đây là một số tính toán minh họa cho bảng trên.

$$\nabla y_3 = \frac{y_3 - y_1}{x_3 - x_1} = \frac{59 - (-1)}{4 - (-2)} = 10$$

$$\nabla^2 y_3 = \frac{\nabla y_3 - \nabla y_2}{x_3 - x_2} = \frac{59 - 1}{4 - 1} = 3$$

$$\nabla^3 y_6 = \frac{\nabla^2 y_6 - \nabla^2 y_3}{x_6 - x_3} = \frac{-5 - 3}{-4 - 4} = 1$$

từ Bảng trên ta thấy hệ số khác không sau cùng của đa thức nội suy Newton là $\nabla^3 y_3$ nên đa thức này là đa thức bậc ba.

□

VÍ DỤ 4.2.3. Các điểm dữ liệu trong bảng nằm trên đồ thị của hàm $f(x) = 4.8 \cos(\pi x / 20)$.

Sử dụng phương pháp Newton nội suy tại các điểm $x = 0; 0.5; 1.0; \dots; 8.0$ và so sánh với giá trị chính xác cho bởi $y = f(x)$.

x	0.15	2.30	3.15	4.85	6.25	7.95
y	4.79867	4.49013	4.2243	3.47313	2.66674	1.51909

GIẢI. Ta sử dụng các hàm newtonCoeff và newtonPoly để tính giá trị cho đa thức nội suy tại các điểm đã cho. Trước tiên, các hệ số nội suy được tính bằng hàm newtonCoeff, sau đó, ta lặp lại hàm newtonPoly tại mỗi điểm để tính giá trị đa thức nội suy tại điểm này đồng thời in ra giá trị số và giá trị chính xác. Chương trình được code như sau.

```
>> % Example 4.2.3 (Newton's interpolation)
xData = [0.15; 2.3; 3.15; 4.85; 6.25; 7.95];
yData = [4.79867; 4.49013; 4.22430; 3.47313; ...
2.66674; 1.51909];
a = newtonCoeff(xData,yData);
' x yInterp yExact'
for x=0:0.5:8
y = newtonPoly(a,xData,x);
yExact = 4.8*cos(pi*x/20);
fprintf('%10.5f',x,y,yExact)
fprintf('\n')
end
```

Kết quả nhận được là

```
ans =
x yInterp yExact
0.00000 4.80003 4.80000
0.50000 4.78518 4.78520
1.00000 4.74088 4.74090
1.50000 4.66736 4.66738
2.00000 4.56507 4.56507
2.50000 4.43462 4.43462
3.00000 4.27683 4.27683
3.50000 4.09267 4.09267
4.00000 3.88327 3.88328
4.50000 3.64994 3.64995
5.00000 3.39411 3.39411
5.50000 3.11735 3.11735
6.00000 2.82137 2.82137
```

6.50000	2.50799	2.50799
7.00000	2.17915	2.17915
7.50000	1.83687	1.83688
8.00000	1.48329	1.48328

□

4.2.3 PHƯƠNG PHÁP NỘI SUY NEVILLE

Để tính giá trị của đa thức nội suy bằng phương pháp nội suy Newton, phải thực hiện hai bước: trước tiên tính các hệ số và sau đó thay giá trị cần tính vào đa thức nội suy. Phương pháp này hiệu quả khi cần tính nhiều giá trị của đa thức nội suy. Nếu chỉ cần tính giá trị của đa thức nội suy tại một điểm x thì phương pháp Neville sau đây sẽ mang lại hiệu quả hơn.

Gọi $P_k[x_i, x_{i+1}, \dots, x_{i+k}]$ là đa thức bậc k đi qua $k + 1$ điểm dữ liệu (x_i, y_i) , $(x_{i+1}, y_{i+1}), \dots, (x_{i+k}, y_{i+k})$. Với một điểm dữ liệu ta có

$$P_0[x_i] = y_i. \quad (4.8)$$

Nội suy dựa vào hai điểm dữ liệu là

$$P_1[x_i, x_{i+1}](x) = \frac{(x - x_{i+1})P_0[x_i] + (x_i - x)P_0[x_{i+1}]}{x_i - x_{i+1}}.$$

Ta dễ thấy $P_1[x_i, x_{i+1}](x)$ đi qua hai điểm dữ liệu $(x_i, y_i), (x_{i+1}, y_{i+1})$.

Nội suy ba điểm là

$$P_2[x_i, x_{i+1}, x_{i+2}](x) = \frac{(x - x_{i+2})P_1[x_i, x_{i+1}](x) + (x_i - x)P_1[x_{i+1}, x_{i+2}](x)}{x_i - x_{i+2}}.$$

Để chỉ ra đa thức này đi qua ba điểm dữ liệu, trước tiên ta thay $x = x_i$, ta có

$$P_2[x_i, x_{i+1}, x_{i+2}](x_i) = P_1[x_i, x_{i+1}](x_i) = y_i.$$

Tương tự, $x = x_{i+2}$ ta có

$$P_2[x_i, x_{i+1}, x_{i+2}](x_{i+2}) = P_1[x_{i+1}, x_{i+2}](x_{i+2}) = y_{i+2}.$$

Cuối cùng, thay $x = x_{i+1}$, vì

$$P_1[x_i, x_{i+1}](x_{i+1}) = P_1[x_{i+1}, x_{i+2}](x_{i+1}) = y_{i+1}$$

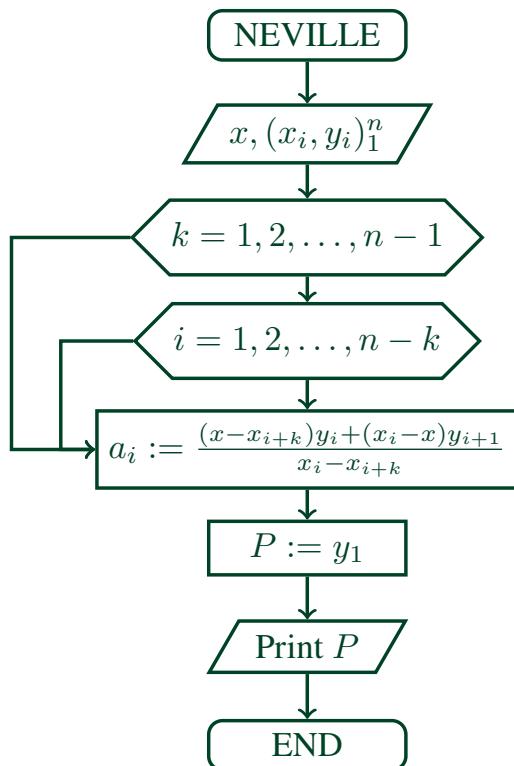
nên

$$P_2[x_i, x_{i+1}, x_{i+2}](x_{i+1}) = \frac{(x - x_{i+2})y_{i+1} + (x_i - x)y_{i+1}}{x_i - x_{i+2}} = y_{i+1}.$$

Tương tự, ta có công thức đệ quy tổng quát cho phương pháp Neville

$$\begin{aligned} P_k[x_i, x_{i+1}, \dots, x_{i+k}](x) \\ = \frac{(x - x_{i+k})P_{k-1}[x_i, x_{i+1}, \dots, x_{i+k-1}](x) + (x_i - x)P_{k-1}[x_{i+1}, x_{i+2}, \dots, x_{i+k}](x)}{x_i - x_{i+k}}. \end{aligned} \quad (4.9)$$

Với mỗi giá trị x cho trước, quá trình tính toán được thực hiện như trong bảng sau (với bốn điểm dữ liệu).



**Hình 4.6: Sơ đồ khối
tính đa thức nội suy
Neville qua n điểm
dữ liệu cho trước**

nevilles.

Chương trình sau đây làm việc với mảng một chiều y mà giá trị ban đầu là giá trị y của dữ liệu (cho trong cột thứ hai của Bảng 4.2). Mỗi vòng lặp sẽ tính một cột tương ứng và ghi đè lên các phần tử của y . Đến vòng lặp cuối cùng, y chứa các phần tử nằm trên đường chéo của Bảng 4.2. Giá trị của đa thức nội suy (tại điểm x) bằng giá trị y_1 , phần tử đầu tiên của y .

```
function yInterp = neville(xData, yData, x)
```

	$k = 0$	$k = 1$	$k = 2$	$k = 3$
x_1	$P_0[x_1] = y_1$	$P_1[x_1, x_2]$	$P_2[x_1, x_2, x_3]$	$P_3[x_1, x_2, x_3, x_4]$
x_2	$P_0[x_2] = y_2$	$P_1[x_2, x_3]$	$P_2[x_2, x_3, x_4]$	
x_3	$P_0[x_3] = y_3$	$P_1[x_3, x_4]$		
x_4	$P_0[x_4] = y_4$			

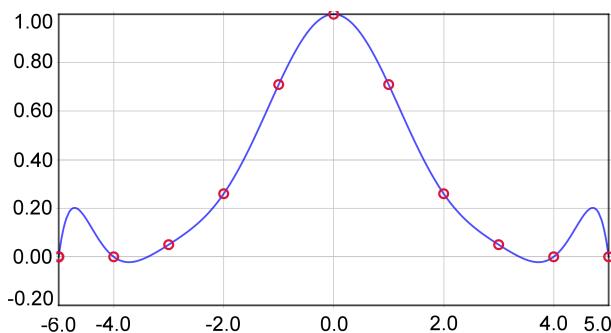
Bảng 4.2: Bảng tính đa thức nội suy Neville với $n = 4$.

```
% Neville's polynomial interpolation;
% returns the value of the interpolant at x.
% USAGE: yInterp = neville(xData,yData,x)
% xData = x-coordinates of data points.
% yData = y-coordinates of data points.
n = length(xData);
y = yData;
for k=1:n-1
    y(1:n-k) = ((x - xData(k+1:n)).*y(1:n-k) ...
        + (xData(1:n-k) - x).*y(2:n-k+1)) ...
        ./ (xData(1:n-k) - xData(k+1:n));
end
yInterp = y(1);
```

Hạn chế của đa thức nội suy.

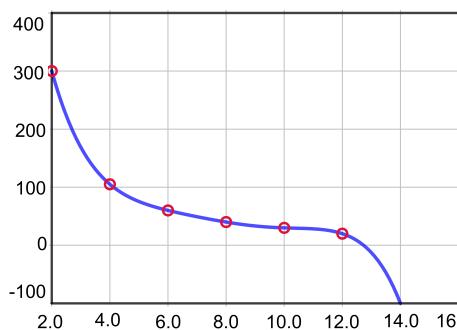
Một hạn chế lớn của các phương pháp nội suy là đa thức nội suy sẽ có bậc lớn khi nhiều điểm dữ liệu. Điều này dẫn đến cần nhiều bộ nhớ để lưu trữ cũng như quá trình tính toán liên quan đến đa thức nội suy sẽ cần nhiều phép tính. Hơn nữa, số bậc lớn của đa thức nội suy có thể khiến cho việc xấp xỉ hàm số tại lân cận các mốc nội suy không còn được chính xác nữa. Ta xét trường hợp minh họa trong Hình 4.7. Có 11 điểm dữ liệu cách đều nhau được biểu thị bằng các vòng tròn. Đường liền nét là đa thức nội suy bậc mười đi qua 11 điểm dữ liệu. Như đã thấy trong hình, đa thức có xu hướng dao động quá mức giữa hai điểm dữ liệu phía trước và hai điểm phía sau. Nếu xấp xỉ hàm số tại điểm nằm giữa hai điểm dữ liệu này sẽ gây nên sai số rất lớn. Một cách làm tốt hơn là: sử dụng một đa thức nội suy bậc 5 đi qua 6 điểm dữ liệu trước và một đa thức nội suy bậc 5 đi qua 6 điểm dữ liệu sau. Sau đó ghép hai đa thức này lại ta sẽ có một hàm có bậc 5, nhỏ hơn bậc của đa thức nội suy (bậc 10). Hàm nhận được bằng phương pháp này gọi là *hàm ghép trơn* (Spline).

Như vậy, khi số điểm dữ liệu quá lớn thì ta nên chia các điểm dữ liệu này thành các khoảng với một số ít điểm (thường thì không nên quá 6 điểm/khoảng), sử dụng phương pháp nội suy để tìm các đa thức nội suy trên các khoảng này, sau đó ghép các đa thức này lại để được hàm ghép trơn. Khi đó, ta sẽ được một đa thức có bậc không quá lớn đi qua tất cả các điểm dữ liệu.



Hình 4.7: Đồ thị minh họa đa thức nội suy bậc cao bị dao động khi xấp xỉ nhiều điểm dữ liệu

Phép ngoại suy đa thức (lấy giá trị xấp xỉ nằm ngoài phạm vi các điểm dữ liệu) là nguy hiểm. Ví dụ, xem xét Hình 4.8. Có sáu điểm dữ liệu, được hiển thị dưới dạng vòng tròn. Đa thức nội suy bậc 6 được biểu diễn bằng đường liền nét. Việc nội suy sẽ tốt nếu ta lấy giá trị cần xấp xỉ trong phạm vi các điểm dữ liệu, nhưng giá trị của y sẽ không còn chính xác nếu khi $x > 12$.



Hình 4.8: Đồ thị minh họa phép nội suy có thể không theo xu hướng dữ liệu

Nếu không thể tránh việc ngoại suy, cần phải thực hiện những bước sau đây để hạn chế sai số.

- Vẽ dữ liệu và xác minh các giá trị ngoại suy có nghĩa.
- Sử dụng đa thức bậc thấp cho các điểm dữ liệu gần nhau.
- Làm việc với một biểu đồ $\log x$ với $\log y$ thường mượt hơn nhiều so với đường cong $x - y$ và do đó an toàn hơn để ngoại suy.

VÍ DỤ 4.2.4. Cho các điểm dữ liệu

x	4.0	4.9	3.8	3.7
y	-0.06604	-0.02724	0.01282	0.05383

xác định nghiệm của $y(x) = 0$ bằng phương pháp Neville.

GIẢI. Đây là ví dụ về nội suy ngược với x, y đổi vai trò cho nhau. Thay vì tính y tại giá trị x cho trước, chúng ta tính x tại giá trị y cho trước (trường hợp này $y = 0$). Dựa theo Bảng 4.2, ta có

i	y_i	$P_0[] = x_i$	$P_1[,]$	$P_2[,,]$	$P_3[,,,]$
1	-0.06604	4.0	3.8298	3.8316	3.8317
2	-0.02724	3.9	3.8320	3.8318	
3	0.01282	3.8	3.8313		
4	0.05383	3.7			

Hai tính toán sau minh họa các tính toán trong bảng trên.

$$\begin{aligned} P_1[y_1, y_2] &= \frac{(y - y_2)P_0[y_1] + (y_1 - y)P_0[y_2]}{y_1 - y_2} \\ &= \frac{(0 + 0.02724)4.0 + (-0.06604 - 0)3.9}{-0.06604 + 0.02724} = 3.8298. \\ P_2[y_2, y_3, y_4] &= \frac{(y - y_4)P_1[y_2, y_3] + (y_2 - y)P_1[y_3, y_4]}{y_2 - y_4} \\ &= \frac{(0 - 0.05383)3.8320 + (-0.02724 - 0)3.8313}{-0.02724 - 0.05383} = 3.8318. \end{aligned}$$

Tất cả các P_s trong bảng trên đều được tính toán với $y = 0$ nên P_s là nghiệm của đa thức nội suy đi qua các điểm tương ứng. Chẳng hạn $P_1[y_1, y_2]$ là nghiệm của đa thức bậc một đi qua hai điểm đầu tiên, $P_2[y_2, y_3, y_4]$ là nghiệm của đa thức bậc hai đi qua ba điểm cuối cùng. Do đó, nghiệm của đa thức bậc ba đi qua bốn điểm dữ liệu là $x = P_3[y_1, y_2, y_3, y_4] = 3.8317$.

□

4.3 PHƯƠNG PHÁP BÌNH PHƯƠNG NHỎ NHẤT

4.3.1 MỞ ĐẦU

Nếu dữ liệu thu được từ thực nghiệm, chúng thường chứa một lượng nhiễu ngẫu nhiên đáng kể do lỗi đo lường. Nhiệm vụ của việc xấp xỉ đường cong là tìm đường

cong trơn phù hợp với điểm dữ liệu theo nghĩa trung bình. Đường cong này phải có dạng đơn giản (ví dụ: đa thức bậc thấp), để không tái tạo nhiễu.

Cho

$$f(x) = f(x; a_1, \dots, a_m)$$

là hàm dùng để xấp xỉ n điểm dữ liệu $(x_i, y_i), i = 1, 2, \dots, n$. Ký hiệu hàm số trên có nghĩa là đây là hàm theo biến x mà chứa các tham số $a_j, j = 1, 2, \dots, m$, với $m < n$. Dạng của $f(x)$ được xác định trước, thường dựa vào giả thuyết thực nghiệm mà từ đó thu được các dữ liệu. Vấn đề còn lại là xác định các tham số. Ví dụ nếu dữ liệu biểu diễn khoảng cách y_i của con lắc theo thời gian t thì hàm xấp xỉ có dạng $f(t) = a_1 t e^{-a_2 t}$. Do vậy, việc tìm hàm xấp xỉ được thực hiện qua hai bước: trước tiên tìm dạng hàm số và sau đó xấp xỉ một cách tốt nhất các tham số bằng cách dựa vào các dữ liệu.

Điều này đặt ra câu hỏi: “Xấp xỉ tốt nhất” nghĩa là gì? Nếu các nhiễu chỉ hạn chế trên các biến y thì phương pháp phổ biến là phương pháp *bình phương nhỏ nhất*, đây là phương pháp cực tiểu hàm số

$$S(a_1, \dots, a_m) = \sum_{i=1}^n [y_i - f(x_i)]^2 \quad (4.10)$$

phụ thuộc vào a_j . Do đó, các giá trị tối ưu của các tham số được tính bằng cách giải hệ phương trình

$$\frac{\partial S}{\partial a_k} = 0, k = 1, 2, \dots, m. \quad (4.11)$$

Các đại lượng $r_i = y_i - f(x_i)$ trong (4.10) được gọi là phần dư, chúng biểu diễn sự khác biệt giữa các điểm dữ liệu và hàm xấp xỉ tại x_i . Hàm S cần cực tiểu là tổng của bình phương các phần dư. Phương trình (4.11) cho trường hợp tổng quát, nếu phi tuyến với các a_j thì bài toán rất khó giải.

Nếu hàm xấp xỉ $f(x)$ được chọn dưới dạng tổ hợp tuyến tính của các hàm đặc biệt $f_j(x)$:

$$f(x) = a_1 f_1(x) + a_2 f_2(x) + \dots + a_m f_m(x)$$

thì hệ phương trình (4.11) là hệ phương trình tuyến tính. Hàm xấp xỉ là đa thức khi $f_1(x) = 1, f_2(x) = x, f_3(x) = x^2, \dots$

Độ phân tán của dữ liệu để xấp xỉ hàm số được đo bởi đại lượng gọi là *độ lệch chuẩn* và được xác định bởi

$$\sigma = \sqrt{\frac{S}{n-m}}. \quad (4.12)$$

Chú ý rằng khi $m = n$ ta có nội suy đa thức. Trong trường hợp này, cả tử số và mẫu số của công thức (4.12) đều bằng không nên σ không có nghĩa.

4.3.2 PHƯƠNG TRÌNH HỒI QUY TUYẾN TÍNH

Xấp xỉ dữ liệu bởi đường thẳng

$$f(x) = a + bx \quad (4.13)$$

còn được biết đến với tên gọi *hồi quy tuyến tính*. Trong trường hợp này, hàm cần cực tiểu là

$$S(a, b) = \sum_{i=1}^n [y_i - a - bx_i]^2.$$

Hệ phương trình (4.11) trở thành

$$\begin{aligned} \frac{\partial S}{\partial a} &= \sum_{i=1}^n -2(y_i - a - bx_i) = 2 \left(-\sum_{i=1}^n y_i + na + b \sum_{i=1}^n x_i \right) = 0 \\ \frac{\partial S}{\partial b} &= \sum_{i=1}^n -2(y_i - a - bx_i)x_i = 2 \left(-\sum_{i=1}^n x_i y_i + na \sum_{i=1}^n x_i + b \sum_{i=1}^n x_i^2 \right) = 0. \end{aligned}$$

Chia hai vế các phương trình cho $2n$ ta có

$$a + \bar{x}b = \bar{y}, \quad a\bar{x} + \left(\frac{1}{n} \sum_{i=1}^n x_i^2 \right) b = \frac{1}{n} \sum_{i=1}^n x_i y_i,$$

với

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i, \quad \bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$$

là giá trị trung bình của dữ liệu x và y . Giải các phương trình cho ta các tham số là

$$a = \frac{\bar{y} \sum x_i^2 - \bar{x} \sum x_i y_i}{\sum x_i^2 - n\bar{x}^2}, \quad b = \frac{\sum x_i y_i - n\bar{x}\bar{y}}{\sum x_i^2 - n\bar{x}^2}. \quad (4.14)$$

Các biểu thức này dễ bị lỗi làm tròn nên các tham số nên được tính theo công thức

$$b = \frac{\sum y_i(x_i - \bar{x})}{\sum x_i(x_i - \bar{x})}, \quad a = \bar{y} - \bar{x}b. \quad (4.15)$$

tương đương với công thức (4.14) mà sai số làm tròn giảm hơn.

VÍ DỤ 4.3.1. Xấp xỉ cho dữ liệu dưới đây bằng đường thẳng và tính độ lệch chuẩn.

x	0.0	1.0	2.0	2.5	3.0
y	2.9	3.7	4.1	4.4	5.0

GIẢI. Trung bình của dữ liệu là

$$\bar{x} = \frac{1}{5} \sum x_i = \frac{0.0 + 1.0 + 2.0 + 2.5 + 3.0}{5} = 1.7$$

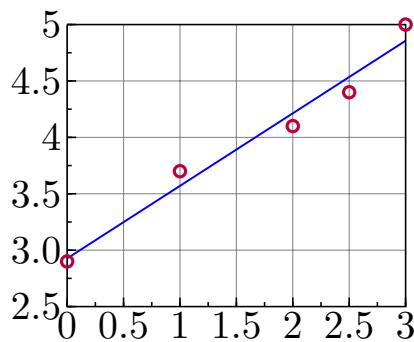
$$\bar{y} = \frac{1}{5} \sum y_i = \frac{2.9 + 3.7 + 4.1 + 4.4 + 5.0}{5} = 4.02.$$

Từ (4.15) ta có

$$b = \frac{\sum y_i(x_i - \bar{x})}{\sum x_i(x_i - \bar{x})} = 0.6431$$

$$a = \bar{y} - \bar{x}b = 2.927.$$

Do đó, đường hồi quy là $f(x) = 2.927 + 0.6431x$ và được mô tả bằng hình sau.



Chúng ta bắt đầu đánh giá độ lệch chuẩn bằng cách tính các phần dư

y	2.900	3.700	4.100	4.400	5.000
$f(x)$	2.927	3.570	4.213	4.535	4.856
$y - f(x)$	-0.027	0.130	-0.113	-0.135	0.144

Tổng bình phương của các phần dư là

$$S = \sum [y_i - f(x_i)]^2$$

$$= (-0.027)^2 + (0.130)^2 + (-0.113)^2 + (-0.135)^2 + (0.144)^2 = 0.06936$$

nên độ lệch chuẩn trong (4.12) trở thành

$$\sigma = \sqrt{\frac{S}{n-m}} = \sqrt{\frac{0.06936}{5-2}} = 0.1520.$$

□

4.3.3 XÂP XỈ DẠNG TUYẾN TÍNH

Xét phương pháp xấp xỉ bình phương nhỏ nhất dạng tuyến tính

$$f(x) = a_1 f_1(x) + a_2 f_2(x) + \cdots + a_m f_m(x) = \sum_{j=1}^m a_j f_j(x), \quad (4.16)$$

với $f_i(x)$ là hàm số xác định trước của x , được gọi là các *hàm cơ sở*. Thay vào (4.10) ta có

$$S = \sum_{i=1}^n \left[y_i - \sum_{j=1}^m a_j f_j(x_i) \right]^2. \quad (a)$$

Do đó, hệ phương trình (4.11) là

$$\frac{\partial S}{\partial a_k} = -2 \left\{ \sum_{i=1}^n \left[y_i - \sum_{j=1}^m a_j f_j(x_i) \right] f_k(x_i) \right\} = 0, k = 1, 2, \dots, m.$$

Bỏ -2 và chuyển về ta có

$$\sum_{j=1}^m \left[\sum_{i=1}^n f_j(x_i) f_k(x_i) \right] a_j = \sum_{i=1}^n f_k(x_i) y_i, k = 1, 2, \dots, m.$$

Dạng ma trận của hệ phương trình này là

$$\mathbf{A}\mathbf{a} = \mathbf{b}, \quad (4.17)$$

với

$$a_{kj} = \sum_{i=1}^n f_j(x_i) f_k(x_i), \quad b_k = \sum_{i=1}^n f_k(x_i) y_i. \quad (4.18)$$

Hệ phương trình (4.17) được gọi là *hệ phương trình chuẩn* của phương pháp xấp xỉ bình phương nhỏ nhất và được giải bằng các phương pháp đã thảo luận trong Chương 3. Chú ý rằng ma trận hệ số có tính đối xứng, tức là $a_{kj} = a_{jk}$.

4.3.4 XÂP XỈ DẠNG ĐA THỨC

Dạng tuyến tính thường được dùng để xấp xỉ là dạng đa thức. Nếu đa thức xấp xỉ có bậc là $m - 1$, ta có $f(x) = \sum_{j=1}^m a_j x^{j-1}$. Các hàm cơ sở là

$$f_j(x) = x^{j-1}, j = 1, 2, \dots, m. \quad (4.19)$$

Khi đó, hệ số của hệ phương trình chuẩn là

$$a_{kj} = \sum_{i=1}^n x_i^{j+k-2}, \quad b_k = \sum_{i=1}^n x_i^{k-1} y_i$$

hoặc

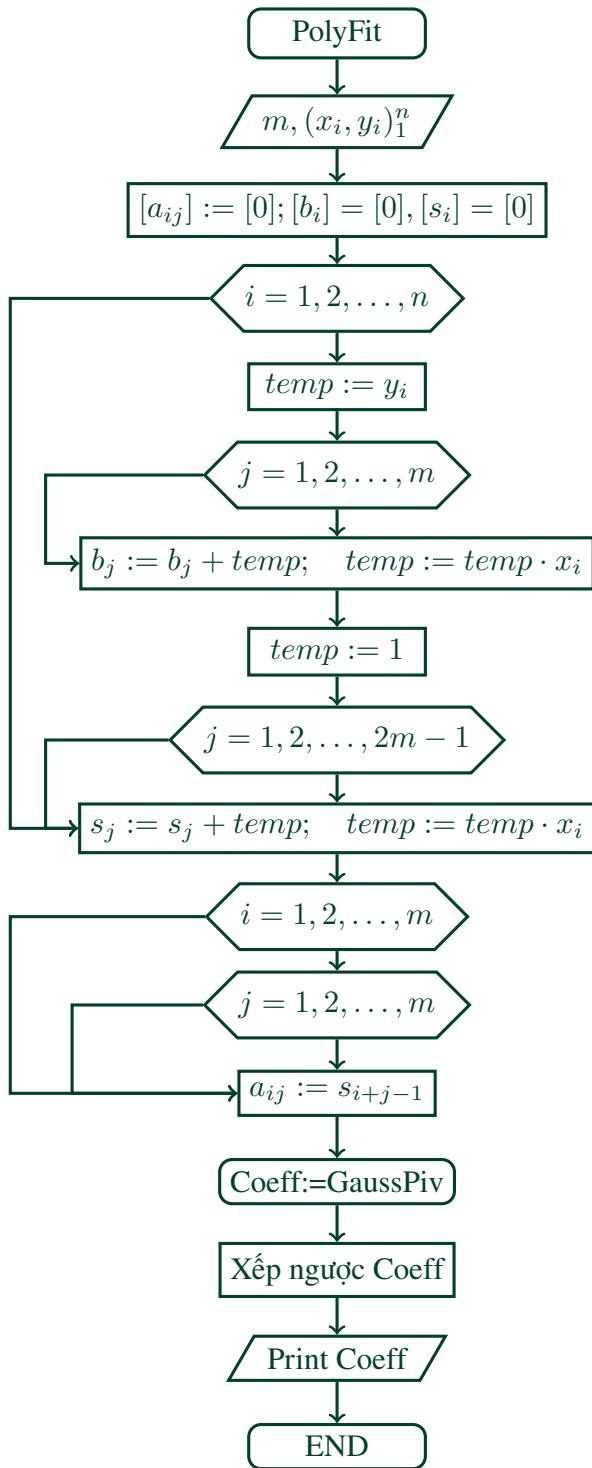
$$\mathbf{A} = \begin{bmatrix} n & \sum x_i & \sum x_i^2 & \cdots & \sum x_i^{m-1} \\ \sum x_i & \sum x_i^2 & \sum x_i^3 & \cdots & \sum x_i^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sum x_i^{m-1} & \sum x_i^m & \sum x_i^{m+1} & \cdots & \sum x_i^{2m-2} \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \sum y_i \\ \sum x_i y_i \\ \vdots \\ \sum x_i^{m-1} y_i \end{bmatrix}, \quad (4.20)$$

với \sum thay thế cho $\sum_{i=1}^n$. Hệ phương trình chuẩn trở thành điều kiện xâu khi m tăng. May mắn là điều này ít gây hậu quả vì trong thực tế ta chỉ sử dụng các đa thức bậc thấp để xấp xỉ bằng phương pháp bình phương nhỏ nhất. Các đa thức bậc cao không được khuyến nghị, vì chúng có xu hướng tái tạo nhiều vốn có trong dữ liệu.

polynFit.

Hàm `polynFit` dùng để tính toán các hệ số của đa thức bậc $m - 1$ khi xấp xỉ n điểm dữ liệu bằng phương pháp bình phương nhỏ nhất. Để thuận lợi cho tính toán, các đại lượng $n, \sum x_i, \sum x_i^2, \dots, \sum x_i^{2m-2}$ của ma trận hệ số \mathbf{A} trong công thức (4.20) trước tiên được lưu trữ trong vectơ `s` rồi sau đó thay thế vào `A`. Hệ phương trình được giải bằng phương pháp khử Gauss kết hợp hàng xoay và cho kết quả là vectơ hệ số `coeff`. Do các phần tử của `coeff` không được xếp theo thứ tự thông thường (hệ số bậc cao hơn của x được xếp trước), nên `coeff` sẽ được “lật lại” sau khi tính xong.

```
function coeff = polynFit(xData, yData, m)
    % Returns the coefficients of the polynomial
    % a(1)*x^(m-1) + a(2)*x^(m-2) + ... + a(m)
    % that fits the data points in the least squares sense.
    % USAGE: coeff = polynFit(xData, yData, m)
    % xData = x-coordinates of data points.
    % yData = y-coordinates of data points.
A=zeros(m); b=zeros(m, 1); s=zeros(2*m-1, 1);
for i=1:length(xData)
    temp = yData(i);
    for j=1:m
        b(j) = b(j) + temp;
        temp = temp*xData(i);
    end
    temp = 1;
    for j=1:2*m-1
        s(j) = s(j) + temp;
        temp = temp*xData(i);
    end
end
```



Hình 4.9: Sơ đồ khôi tính hệ số của đa thức xấp xỉ bởi n điểm dữ liệu cho trước bằng phương pháp bình phương nhỏ nhất

```

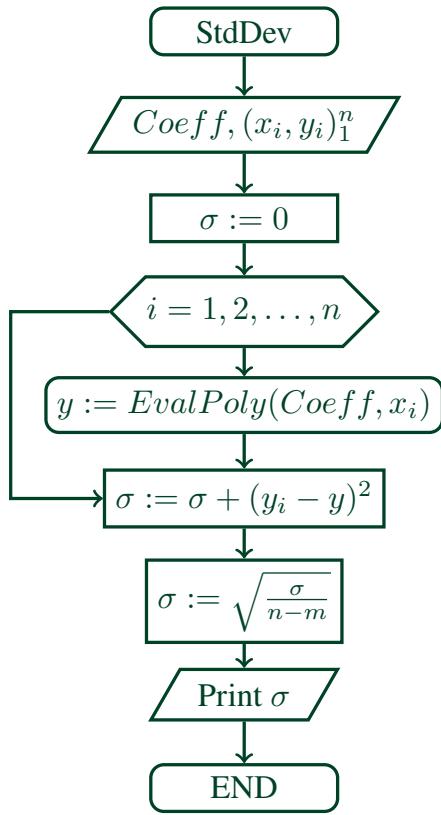
end
end
for i=1:m
    for j=1:m
        A(i, j) = s(i+j-1);
    end

```

```

end
% Rearrange coefficients so that coefficient
% of x^(m-1) is first
coeff = flipdim(gaussPiv(A,b),1);

```



Hình 4.10: Sơ đồ
khởi tính độ lệch
chuẩn của phương
pháp bình phương
nhỏ nhất khi xấp xỉ
n điểm dữ liệu cho
trước bởi đa thức
bậc m

stdDev.

Sau khi hệ số của đa thức xấp xỉ được tính, độ lệch chuẩn σ được tính bởi hàm stdDev. Giá trị của đa thức trong stdDev được tính bởi hàm polyEval trong công thức (2.18).

```

function sigma = stdDev(coeff,xData,yData)
    % Returns the standard deviation between data
    % points and the polynomial
    % a(1)*x^(m-1) + a(2)*x^(m-2) + ... + a(m)
    % USAGE: sigma = stdDev(coeff,xData,yData)
    % coeff = coefficients of the polynomial.
    % xData = x-coordinates of data points.
    % yData = y-coordinates of data points.
    m=length(coeff);n=length(xData);
    sigma = 0;

```

```

for i=1:n
    y = polyEval(coeff,xData(i));
    sigma = sigma + (yData(i) - y)^2;
end
sigma =sqrt(sigma/(n-m));

```

polyEval.

```

function y=polyEval(coeff,x)
% Returns the value of the polynomial at x.
m = length(coeff);
y = coeff(1);
for j=1:m-1
    y = y*x + coeff(j+1);
end

```

VÍ DỤ 4.3.2. Viết một chương trình xấp xỉ một đa thức bậc k tuỳ ý cho các điểm dữ liệu sau. Sử dụng chương trình để xác định k sao cho xấp xỉ dữ liệu tốt nhất bằng phương pháp bình phương nhỏ nhất.

x	-0.04	0.93	1.95	2.90	3.83	5.00
y	-8.66	-6.44	-4.35	-3.27	-0.88	0.87
x	5.98	7.05	8.21	9.08	10.09	
y	3.31	4.63	6.19	7.04	8.85	

GIẢI. Chương trình sau đây tính toán bằng cách nhập k từ bàn phím. Quá trình tính toán sẽ chấm dứt khi nhấn “return”.

```

>> % Example 4.3.2 (Polynomial curve fitting)
xData = [-0.04,0.93,1.95,2.90,3.83,5.0,...  

5.98,7.05,8.21,9.08,10.09]';  

yData = [-8.66,-6.44,-4.36,-3.27,-0.88,0.87,...  

3.31,4.63,6.19,7.4,8.85]';  

format short e  

while 1  

k = input('degree of polynomial = ');  

if isempty(k) % Loop is terminated  

fprintf('Done') % by pressing "return"

```

```

break
end
coeff = polynFit(xData,yData,k+1)
sigma = stdDev(coeff,xData,yData)
fprintf('\n')
end

```

Kết quả là

Degree of polynomial = 1

coeff =

1.7286e+000

-7.9453e+000

sigma =

5.1128e-001

degree of polynomial = 2

coeff =

-4.1971e-002

2.1512e+000

-8.5701e+000

sigma =

3.1099e-001

degree of polynomial = 3

coeff =

-2.9852e-003

2.8845e-003

1.9810e+000

-8.4660e+000

sigma =

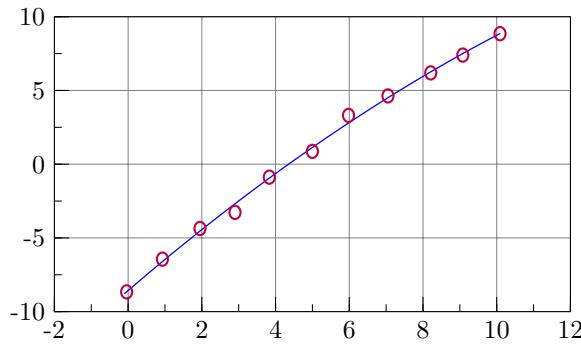
3.1948e-001

degree of polynomial =

Done

Bởi vì tam thức bậc hai $f(x) = -0.041971x^2 + 2.1512x - 8.5701$ có độ lệch chuẩn nhỏ

nhất nên nó là xấp xỉ tốt nhất cho dữ liệu. Lưu ý rằng độ lệch chuẩn không phải là độ đo không thể phạm sai lầm. Cách tốt nhất là ta vẽ các điểm dữ liệu và đồ thị hàm $f(x)$ trên cùng một hệ trục toạ độ để kiểm tra. Hình dưới đây vẽ các điểm dữ liệu đã cho và hàm bậc hai cho thấy xấp xỉ này là hợp lý.



□

4.3.5 DỮ LIỆU CÓ TRỌNG SỐ

Trong một số trường hợp, niềm tin vào độ chính xác của dữ liệu thay đổi theo từng điểm. Chẳng hạn, công cụ thực hiện các phép đo có thể nhạy hơn trong một phạm vi dữ liệu nhất định. Đôi khi dữ liệu đại diện cho kết quả của một số thực nghiệm và mỗi thực nghiệm này lại được thực hiện dưới các tiêu chuẩn khác nhau. Trong các trường hợp này, chúng ta có thể gán hệ số tin cậy hoặc trọng số cho từng điểm dữ liệu và cực tiểu hóa tổng bình phương của phần dư có trọng số $r_i = W_i[y_i - f_i(x_i)]$, với W_i là các trọng số. Do đó, hàm cần được cực tiểu hoá là

$$S(a_1, \dots, a_m) = \sum_{i=1}^n W_i^2 [y_i - f(x_i)]^2. \quad (4.21)$$

Quy trình này buộc hàm xấp xỉ $f(x)$ gần hơn với các điểm dữ liệu có trọng số cao hơn.
Hồi quy tuyến tính có trọng số.

Nếu hàm xấp xỉ là đường thẳng $f(x) = a + bx$ thì (4.21) trở thành

$$S(a, b) = \sum_{i=1}^n W_i^2 [y_i - a - bx_i]^2. \quad (4.22)$$

Điều kiện để cực tiểu hàm S là

$$\frac{\partial S}{\partial a} = -2 \sum_{i=1}^n W_i^2 [y_i - a - bx_i],$$

$$\frac{\partial S}{\partial b} = -2 \sum_{i=1}^n W_i^2 [y_i - a - bx_i] x_i$$

hoặc

$$a \sum_{i=1}^n W_i^2 + b \sum_{i=1}^n W_i^2 x_i = \sum_{i=1}^n W_i^2 y_i \quad (4.23)$$

$$a \sum_{i=1}^n W_i^2 x_i + b \sum_{i=1}^n W_i^2 x_i^2 = \sum_{i=1}^n W_i^2 x_i y_i. \quad (4.24)$$

Chia hai vế (4.23) cho $\sum W_i^2$ và đặt các trung bình trọng số

$$\hat{x} = \frac{\sum W_i^2 x_i}{\sum W_i^2}, \quad \hat{y} = \frac{\sum W_i^2 y_i}{\sum W_i^2} \quad (4.25)$$

ta có

$$a = \hat{y} - b\hat{x}. \quad (4.26)$$

Thay (4.26) vào (4.24) và tính toán ta thu được

$$b = \frac{\sum_{i=1}^n W_i^2 y_i (x_i - \hat{x})}{\sum_{i=1}^n W_i^2 x_i (x_i - \hat{x})}. \quad (4.27)$$

Chú ý rằng các công thức (4.26)-(4.27) tương tự như (4.15) trong trường hợp không có trọng số.

Xấp xỉ bởi hàm mũ.

Một ứng dụng đặc biệt của hồi quy tuyến tính có trọng số là việc xấp xỉ các hàm số mũ cho dữ liệu. Xét một ví dụ hàm xấp xỉ dạng

$$f(x) = ae^{bx}.$$

Nếu giải bình thường, phương pháp bình phương nhỏ nhất sẽ dẫn chúng ta đến hệ phương trình phi tuyến đối với a, b . Nhưng nếu ta xấp xỉ $\ln y$ thay vì y thì bài toán sẽ được chuyển về dạng hồi quy tuyến tính: tìm hàm xấp xỉ

$$F(x) = \ln f(x) = \ln a + bx$$

với các điểm dữ liệu $(x_i, \ln y_i), i = 1, 2, \dots, n$. Cái giá của sự đơn giản này là: thay vì áp dụng phương pháp bình phương nhỏ nhất cho dữ liệu thì ta phải áp dụng phương pháp bình phương nhỏ nhất cho logarit của dữ liệu. Các phần dư của xấp xỉ logarit là

$$R_i = \ln y_i - F(x_i) = \ln y_i - \ln a - bx_i \quad (4.28)$$

trong khi phần dư được sử dụng trong xấp xỉ dữ liệu gốc là

$$r_i = y_i - f(x_i) = y_i - ae^{bx_i}. \quad (4.29)$$

Sự khác biệt này có thể được loại bỏ phần lớn bằng trọng số từ xấp xỉ logarit. Từ (4.29) ta có $\ln(r_i - y_i) = \ln ae^{bx_i} = \ln a + bx_i$ nên (4.28) được viết lại dưới dạng

$$R_i = \ln y_i - \ln(r_i - y_i) = \ln\left(1 - \frac{r_i}{y_i}\right).$$

Nếu phần dư r_i đủ nhỏ ($r_i \ll y_i$) ta có thể xấp xỉ $\ln(1 - r_i/y_i) \approx r_i/y_i$, do đó

$$R_i \approx \frac{r_i}{y_i}.$$

Bây giờ ta có thể thấy rằng, bằng cách cực tiểu hóa $\sum R_i^2$, chúng ta đã tìm cờ giới thiệu trọng số $1/y_i$. Quá trình này có thể bị phủ nhận nếu chúng ta áp dụng các trọng số y_i khi xấp xỉ hàm $F(x)$ cho các dữ liệu $(\ln y_i, x_i)$ bằng cách cực tiểu hóa

$$S = \sum_{i=1}^n y_i^2 R_i^2. \quad (4.30)$$

Các ví dụ hữu ích khác khi lấy trọng số $W_i = y_i$ được cho trong bảng sau.

$f(x)$	$F(x)$	Dữ liệu được xấp xỉ cho $F(x)$
axe^{bx}	$\ln[f(x)/x] = \ln a + bx$	$[x_i, \ln(y_i/x_i)]$
ax^b	$\ln f(x) = \ln a + b \ln x$	$(\ln x_i, \ln y_i)$

Bảng 4.3: Bảng xấp xỉ bình phương nhỏ nhất với trường hợp dữ liệu có trọng số.

VÍ DỤ 4.3.3. Xác định tham số a, b sao cho hàm $f(x) = ae^{bx}$ là hàm xấp xỉ dữ liệu dưới đây bằng phương pháp bình phương nhỏ nhất

x	1.2	2.8	4.3	5.4	6.8	7.9
y	7.5	16.1	38.9	67.0	146.6	266.2

theo hai cách: (1) xỉ $\ln y_i$ và (2) xỉ $\ln y_i$ với các trọng số $W_i = y_i$. Tính độ lệch chuẩn cho mỗi trường hợp.

GIẢI.

Trường hợp (1). Bài toán trở thành tìm hàm xấp xỉ $\ln ae^{bx} = \ln a + bx$ theo dữ liệu

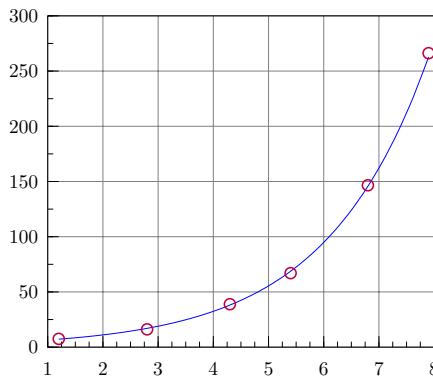
x	1.2	2.8	4.3	5.4	6.8	7.9
$z = \ln y$	2.015	2.779	3.661	4.205	4.988	5.584

Ta tìm đường hồi quy tuyến tính với tham số cần tìm là $A = \ln a$ và b . Thực hiện tính toán tương tự như Ví dụ 4.3.1 ta được

$$\bar{x} = \frac{1}{6} \sum x_i = 4.733 \quad \bar{z} = \frac{1}{6} \sum z_i = 3.872$$

$$b = \frac{\sum z_i(x_i - \bar{x})}{\sum x_i(x_i - \bar{x})} = \frac{16.716}{31.153} = 0.5366 \quad A = \bar{z} - \bar{x}b = 1.3323.$$

Do đó, $a = e^A = 3.790$ và hàm xấp xỉ trở thành $f(x) = 3.790e^{0.5366x}$. Hàm $f(x)$ và các điểm dữ liệu được vẽ trong hình sau.



Tiếp theo ta tính độ lệch chuẩn:

y	7.5	16.1	38.9	67.0	146.6	266.2
$f(x)$	7.21	17.02	38.07	68.69	145.60	262.72
$y - f(x)$	0.29	-0.92	0.83	-1.69	1.00	3.48

$$S = \sum [y_i - f(x_i)]^2 = 17.59$$

$$\sigma = \sqrt{\frac{S}{n-m}} = 2.10.$$

Như đã trình bày phía trên, thay vì xấp xỉ y_i , chúng ta xấp xỉ $\ln y_i$ để được lời giải xấp xỉ của bài toán đã nêu.

Trường hợp (2). Chúng ta tiếp tục xấp xỉ hàm $\ln ae^{bx} = \ln a + bx$ nhưng các trọng số $W_i = y_i$ được sử dụng. Từ (4.25), các trung bình trọng số được tính (nhắc lại là ta xấp xỉ $z = \ln y$)

$$\hat{x} = \frac{\sum y_i^2 x_i}{\sum y_i^2} = \frac{737.5 \times 10^3}{98.67 \times 10^3} = 7.474,$$

$$\hat{z} = \frac{\sum y_i^2 z_i}{\sum y_i^2} = \frac{528.2 \times 10^3}{98.67 \times 10^3} = 5.353$$

và từ (4.27) ta có tham số

$$b = \frac{\sum_{i=1}^n y_i^2 z_i(x_i - \hat{x})}{\sum_{i=1}^n y_i^2 x_i(x_i - \hat{x})} == \frac{35.39 \times 10^3}{65.05 \times 10^3} = 0.5440,$$

$$\ln a = \hat{z} - b\hat{x} = 5.353 - 0.5440(7.474) = 1.287.$$

Vì

$$a = e^{\ln a} = e^{1.287} = 3.622$$

nên hàm xấp xỉ cần tìm là $f(x) = 3.622e^{0.5440x}$. Ta thấy kết quả này trùng với kết quả thu được phần (1).

Các phần dư và độ lệch chuẩn được tính như sau:

y	7.50	16.10	38.90	67.00	146.60	266.20
$f(x)$	6.96	16.61	37.56	68.33	146.33	266.20
$y - f(x)$	0.54	-0.51	1.34	-1.33	0.267	0.00

$$S = \sum [y_i - f(x_i)]^2 = 41.68$$

$$\sigma = \sqrt{\frac{S}{n-m}} = 1.023.$$

Ta quan sát thấy rằng các phần dư và độ lệch chuẩn này nhỏ hơn phần (1) nên xấp xỉ tốt hơn. Ta thấy rằng nếu xấp xỉ trực tiếp y_i (xấp xỉ liên quan đến tìm nghiệm hàm siêu việt) ta sẽ thu được kết quả là $f(x) = 3.164e^{0.5442x}$ với độ lệch chuẩn là $\sigma = 1.022$, nhỏ hơn một chút so với phần (2).

□

4.4 MỘT SỐ HÀM MATLAB TÍNH TOÁN XẤP XỈ HÀM SỐ

Một số hàm MATLAB thông dụng dùng để tính toán trong nội suy và xấp xỉ hàm số sẽ được giới thiệu sau đây.

`y = interp1(xData, yData, x, method)` trả giá trị của đa thức nội suy y tại điểm x xác định theo `method`: `method = 'linear'` là sử dụng phương pháp nội suy tuyến tính giữa các điểm liền kề (đây là phương pháp mặc định). Nếu x là một mảng thì y sẽ được tính theo tất cả phần tử của x .

`a = polyfit (xData, yData, m)` trả các hệ số a của đa thức bậc m xấp xỉ các điểm dữ liệu bằng phương pháp bình phương nhỏ nhất.

`y = polyval (a, x)` tính giá trị của đa thức có các hệ số a tại điểm x . Nếu x là một mảng thì y sẽ được tính theo tất cả phần tử của x .

`s = std (x)` trả về độ lệch chuẩn của các phần tử của mảng x . Nếu x là một ma trận thì y sẽ được tính theo từng cột của x .

`y = mean (x)` tính giá trị trung bình của các phần tử của mảng x . Nếu x là một ma trận thì y sẽ được tính theo từng cột của x .

TÓM TẮT CHƯƠNG 4

Nội dung chính của Chương 4 trình bày các phương pháp xấp xỉ dữ liệu bởi một hàm số. Dựa vào đặc điểm của dữ liệu mà có hai nhóm phương pháp xấp xỉ. Nếu các dữ liệu đã biết có độ chính xác cao hoặc cần xấp xỉ một hàm số phức tạp bởi một đa thức đơn giản, chúng ta sử dụng các đa thức nội suy. Các phương pháp xây dựng đa thức nội suy đã trình bày gồm có:

1. Đa thức nội suy Lagrange;
2. Đa thức nội suy Newton;
3. Đa thức nội suy Neville.

Nếu dữ liệu có nhiều lớn hoặc phân tán thì phương pháp hữu hiệu để xấp xỉ là phương pháp bình phương nhỏ nhất. Đây là phương pháp xấp xỉ các điểm dữ liệu bằng một đường (thường là có bậc thấp) theo hình xu hướng của nhóm các điểm dữ liệu. Chương này giới thiệu một số dạng xấp xỉ sau:

1. Xấp xỉ bởi phương trình hồi quy tuyến tính;
2. Xấp xỉ dạng tuyến tính;
3. Xấp xỉ dạng đa thức;
4. Xấp xỉ cho dữ liệu có trọng số.

BÀI TẬP CHƯƠNG 4

Bài tập 4.1. Cho các điểm dữ liệu

x	-1.2	0.3	1.2
y	-5.76	-5.61	-3.69

xác định y tại $x = 0$ bằng cách sử dụng

- a. phương pháp Neville
- b. phương pháp Lagrange

Bài tập 4.2. Tìm điểm không của $y(x)$ từ dữ liệu

x	0.0	0.5	1.0	1.5	2.0	2.5	3.0
y	1.8421	2.4694	2.4921	1.9047	0.8509	-0.4112	-1.5727

Sử dụng nội suy Lagrange với (a) ba điểm; và (b) bốn điểm dữ liệu lân cận. Gọi ý.
Sau khi hoàn thành phần (a), phần (b) có thể tính toán với khối lượng nhỏ.

Bài tập 4.3. Hàm $f(x)$ biểu diễn dữ liệu trong Bài tập 4.2 lớn nhất tại $x = 0.7679$.
Tính giá trị lớn nhất này bởi nội suy đa thức Neville trên bốn điểm dữ liệu lân cận.

Bài tập 4.4. Sử dụng phương pháp Neville tính y tại $x = \pi/4$ từ các điểm dữ liệu

x	0.0	0.5	1.0	1.5	2.0
y	-1.00	1.75	4.00	5.75	7.00

Bài tập 4.5. Cho các điểm dữ liệu

x	0.0	0.5	1.0	1.5	2.0
y	-0.7854	0.6529	1.7390	2.2071	1.9425

tìm y tại $x = \pi/4$ và $\pi/2$. Sử dụng phương pháp mà bạn cho là thuận tiện nhất.

Bài tập 4.6. Các điểm

x	-2	1	4	- - 1	3	-4
y	-1	2	59		4	24

nằm trên một đa thức. Sử dụng bảng sai phân của phương pháp Newton để xác định bậc của đa thức.

Bài tập 4.7. Sử dụng phương pháp Newton tìm đa thức có bậc nhỏ nhất xấp xỉ các điểm sau:

x	-3	2	-1	3	1
y	0	5	-4	12	0

Bài tập 4.8. Sử dụng phương pháp Neville tìm đa thức có bậc hai đi qua các điểm sau:

x	-1	1	3
y	1.225	0.905	0.652

Bài tập 4.9. Viết phương trình tính đa thức nội suy bằng phương pháp Neville. Chương trình phải có khả năng tính giá trị của đa thức tại những điểm cho trước. Kiểm tra chương trình bằng cách xác định y tại các điểm $x = 1.1; 1.2$ và 1.3 từ dữ liệu sau:

x	-2.0	-0.1	-1.5	0.5
y	2.2796	1.0025	1.6467	1.0635
x	-0.6	2.2	1.0	1.8
y	1.0920	2.6291	1.2661	1.9896

(Đáp án: $y = 1.3262; 1.3938; 1.4693$)

Bài tập 4.10. Xác định y tại $x = 0.46$ từ dữ liệu

x	0	0.0204	0.1055	0.241	0.582	0.712	0.981
y	0.285	1.04	1.79	2.63	4.39	4.99	5.27

Lưu ý. Các bài sau đây yêu cầu vẽ các điểm dữ liệu và hàm xấp xỉ.

Bài tập 4.11. Sử dụng hồi quy tuyến tính để tìm đường xấp xỉ dữ liệu

x	-1.0	-0.5	0	0.5	1.0
y	-1.00	-0.55	0.00	0.45	1.00

và xác định độ lệch chuẩn.

Bài tập 4.12. Ba thử nghiệm độ bền kéo được thực hiện trên một thanh nhôm. Trong mỗi thử nghiệm, biến dạng được đo ở số số lực kéo. Kết quả là

Lực kéo (MPa)	34.5	69.0	103.5	138.0
Biến dạng (Test 1)	0.46	0.95	1.48	1.93
Biến dạng (Test 2)	0.34	1.02	1.51	2.09
Biến dạng (Test 3)	0.73	1.10	1.62	2.12

với đơn vị của biến dạng là mm/m. Sử dụng hồi quy tuyến tính để ước tính mô đun đàn hồi của thanh (mô đun đàn hồi = ứng suất / biến dạng).

Bài tập 4.13. Xấp xỉ đường thẳng cho dữ liệu dưới đây và tính độ lệch chuẩn.

x	0	0.5	1	1.5	2	2.5
y	3.076	2.810	2.588	2.297	1.981	1.912
x	3	3.5	4	4.5	5	
y	1.653	1.478	1.399	1.018	0.794	

Bài tập 4.14. Bảng sau đây thể hiện khối lượng M và mức tiêu thụ nhiên liệu ϕ của các xe ô tô xả suất năm 1999 của Honda và Ford. Xấp xỉ đường thẳng $\phi = a + bM$ cho dữ liệu và tính độ lệch chuẩn.

Model	M (kg)	ϕ (km/liter)
Contour	1310	10.2
Crown Victoria	1810	8.1
Escort	1175	11.9
Expedition	2360	5.5
Explorer	1960	6.8
F-150	2020	6.8
Ranger	1755	7.7
Taurus	1595	8.9
Accord	1470	9.8
CR-V	1430	10.2
Civic	1110	13.2
Passport	1785	7.7

Bài tập 4.15. Mật độ tương đối của không khí ρ được đo ở nhiều độ cao h . Kết quả là

h (km)	0	1.525	3.050	4.575	6.10	7.625	9.150
ρ (km)	1	0.8617	0.7385	0.6292	0.5328	0.4481	0.3741

Sử dụng xấp xỉ bình phương nhỏ nhất bậc ba để xác định mật độ tương đối của không khí tại $h = 10.5$ km.

Bài tập 4.16. Xấp xỉ dữ liệu sau đây bằng đường thẳng và đa thức bậc hai.

x	1	2.5	3.5	4.0	1.1	1.8	2.2	3.7
y	6.008	15.722	27.130	33.772	5.257	9.549	11.098	28.828

Xấp xỉ nào tốt hơn?

Bài tập 4.17. Bảng sau hiển thị hiệu suất nhiệt của một số động cơ hơi nước sớm. Xác định đa thức xấp xỉ dữ liệu phù hợp nhất và sử dụng nó để dự đoán hiệu suất nhiệt trong năm 2000.

Năm	Hiệu suất %	Động cơ
1718	0.5	Newcomen
1767	0.8	Smeaton
1774	1.4	Smeaton
1775	2.7	Watt
1792	4.5	Watt
1816	7.5	Woolf compound
1828	12.0	ImprovedCornish
1834	17.0	ImprovedCornish
1878	17.2	Corliss compound
1906	23.0	Triple expansion

Bài tập 4.18. Bảng sau cho thấy sự biến thiên của độ dẫn nhiệt tương đối k của một loại hoá chất với nhiệt độ T . Tìm phương trình bậc hai phù hợp với dữ liệu theo nghĩa bình phương nhỏ nhất.

T ($^{\circ}$ C)	79	190	357	524	690
k	1.00	0.932	0.839	0.759	0.693

Bài tập 4.19. Xác định tham số a, b cho hàm $f(x) = a \sin(\pi x/2) + b \cos(\pi x/2)$ xấp xỉ dữ liệu sau bằng phương pháp bình phương nhỏ nhất.

x	-0.5	-0.19	0.02	0.20	0.35	0.50
y	-3.558	-2.874	-1.995	-1.040	-0.068	0.677

Bài tập 4.20. Xác định tham số a, b cho hàm $f(x) = ae^{bx}$ xấp xỉ dữ liệu sau bằng phương pháp bình phương nhỏ nhất.

x	0.5	1.0	1.5	2.0	2.5
y	0.49	1.60	3.36	6.44	10.16

Bài tập 4.21. Xấp xỉ hàm $f(x) = ae^{bx}$ cho dữ liệu sau và tính độ lệch chuẩn.

x	0.5	1.0	1.5	2.0	2.5
y	0.541	0.398	0.232	0.106	0.052

CHƯƠNG 5

TÍNH GẦN ĐÚNG ĐẠO HÀM VÀ TÍCH PHÂN

5.1	Giới thiệu	168
5.2	Tính gần đúng đạo hàm	168
5.2.1	Mở đầu	168
5.2.2	Xấp xỉ sai phân hữu hạn	169
5.2.3	Phép ngoại suy Richardson	175
5.2.4	Xấp xỉ đạo hàm bằng nội suy	176
5.3	Tính gần đúng tích phân xác định	180
5.3.1	Mở đầu	180
5.3.2	Các công thức Newton-Cotes	181
5.3.3	Tích phân Romberg	192
5.3.4	Công thức cầu phương Gaussian	197
5.4	Tích phân bội	214
5.4.1	Mở đầu	214
5.4.2	Phương pháp cầu phương Gauss-Legendre trên phần tử tứ giác	215
5.4.3	Phương pháp cầu phương trên một phần tử tam giác	222
5.5	HÀM MATLAB TÍNH ĐẠO HÀM-TÍCH PHÂN	227

5.1 GIỚI THIỆU

Trong trường phổ thông hoặc trong những năm đầu của đại học, bạn đã được giới thiệu về phép tính vi phân và tích phân. Tại những nơi đó, bạn đã được học kỹ thuật để thu được đạo và tích phân dạng giải tích hay chính xác.

Về mặt toán học, *đạo hàm* biểu diễn tỉ lệ thay đổi giữa một biến phụ thuộc với một biến độc lập. Chẳng hạn, nếu bạn được cho một hàm $y(t)$ là hàm xác định vị trí của một đối tượng theo biến thời gian t thì đạo hàm là một cách để xác định vận tốc của nó dưới dạng:

$$v(t) = \frac{d}{dt}y(t).$$

Tích phân là quá trình ngược của vi phân. Giống như vi phân sử dụng sự khác biệt để định lượng một quá trình tức thời còn tích phân tổng hợp thông tin tức thời để đưa ra kết quả tổng hợp trong một khoảng. Do đó, nếu chúng ta đã biết vận tốc của một đối tượng là một hàm theo thời gian thì tích phân có thể được sử dụng để xác định khoảng cách di chuyển:

$$y(t) = \int_0^t v(t)dt.$$

Do mối quan hệ chặt chẽ giữa vi phân và tích phân nên chúng tôi dành chương này để nói về hai vấn đề này. Vì cả hai quá trình đều có sử dụng cùng một số công thức toán học nên việc này giúp chúng tôi không phải nhắc lại những công thức này và giúp cho người đọc dễ dàng hệ thống kiến thức. Mục đầu tiên của chương này sẽ dành để giới thiệu một số phương pháp tính số vi phân. Việc tính chính xác tích phân không phải khi nào cũng có thể thực hiện được nên việc tính số tích phân rất quan trọng. Do đó, chúng tôi dành phần lớn nội dung của chương vào giải quyết vấn đề này. Cuối cùng, cũng như các chương trước, chúng tôi giới thiệu một số hàm MATLAB thông dụng trong việc tích vi phân và tích phân.

5.2 TÍNH GẦN ĐÚNG ĐẠO HÀM

5.2.1 MỞ ĐẦU

Phép tính số vi phân thường đi với bài toán sau: Cho trước hàm $y = f(x)$, tìm một trong các đạo hàm của nó tại điểm $x = x_k$. Thuật ngữ *cho trước* có nghĩa là hoặc chúng ta biết được thuật toán để tính hàm số, hoặc chúng ta biết tập các điểm dữ liệu rời rạc $(x_i, y_i), i = 1, 2, \dots, n$. Trong cả hai trường hợp, chúng ta có quyền truy cập vào

một số hữu hạn các cặp dữ liệu (x, y) từ đó để tính đạo hàm. Nếu bạn nghi ngờ rằng phép tính số vi phân liên quan đến phép nội suy thì bạn đã đúng. Một trong những phương pháp để tìm đạo hàm là xấp xỉ nó bởi một đa thức và sau đó lấy đạo hàm. Một công cụ hiệu quả tương đương là sử dụng khai triển chuỗi Taylor của $f(x)$ tại điểm x_k . Phương pháp này có lợi ích là cung cấp cho ta thông tin liên quan đến sai số của phép xấp xỉ.

Phép tính số vi phân không phải là quá trình tính toán có độ chính xác cao. Nó chịu ảnh hưởng từ các sai số làm tròn và các sai số gắn với quá trình nội suy. Vì lý do này, một đạo hàm của hàm không bao giờ có thể được tính toán với độ chính xác như chính hàm đó.

5.2.2 XẤP XỈ SAI PHÂN HỮU HẠN

Phương pháp xấp xỉ sai phân hữu hạn để tính đạo hàm của hàm $f(x)$ dựa vào các công thức khai triển chuỗi Taylor tiến và lùi của hàm $f(x)$ với biến x như sau

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2!}f''(x) + \frac{h^3}{3!}f'''(x) + \frac{h^4}{4!}f^{(4)}(x) + \dots \quad (a)$$

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2!}f''(x) - \frac{h^3}{3!}f'''(x) + \frac{h^4}{4!}f^{(4)}(x) + \dots \quad (b)$$

$$f(x+2h) = f(x) + 2hf'(x) + \frac{(2h)^2}{2!}f''(x) + \frac{(2h)^3}{3!}f'''(x) + \frac{(2h)^4}{4!}f^{(4)}(x) + \dots \quad (c)$$

$$f(x-2h) = f(x) - 2hf'(x) + \frac{(2h)^2}{2!}f''(x) - \frac{(2h)^3}{3!}f'''(x) + \frac{(2h)^4}{4!}f^{(4)}(x) + \dots \quad (d)$$

Thực hiện tính toán ta có

$$f(x+h) + f(x-h) = 2f(x) + h^2f''(x) + \frac{h^4}{12}f^{(4)}(x) + \dots \quad (e)$$

$$f(x+h) - f(x-h) = 2hf'(x) + \frac{h^3}{3}f'''(x) + \dots \quad (f)$$

$$f(x+2h) + f(x-2h) = 2f(x) + 4h^2f''(x) + \frac{4(h)^4}{3}f^{(4)}(x) + \dots \quad (g)$$

$$f(x+2h) - f(x-2h) = 4hf'(x) + \frac{8(h)^3}{3}f'''(x) + \dots \quad (h)$$

Lưu ý rằng các tổng chỉ chứa các đạo hàm chẵn, trong khi các hiệu chỉ giữ lại các đạo hàm lẻ. Nhờ việc kết hợp các phương trình (a)-(h), chúng ta có thể tìm được các công thức xấp xỉ các đạo hàm khác nhau của $f(x)$. Số lượng phương trình kết hợp và số lượng biểu thức được giữ trong mỗi phương trình phụ thuộc vào bậc của đạo hàm và mức độ chính xác mong muốn.

A. XẤP XỈ SAI PHÂN TRUNG TÂM BẬC MỘT

Giải phương trình (f) cho $f(x)$ là

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{h^2}{6} f'''(x) - \dots$$

Chỉ giữ lại biểu thức thứ nhất bên vế phải ta có công thức

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + \mathcal{O}(h^2) \quad (5.1)$$

được gọi là *xấp xỉ sai phân trung tâm bậc một* cho $f'(x)$. Ký hiệu $\mathcal{O}(h^2)$ nhắc rằng chúng ta đã bỏ đi sai số như h^2 .

Từ phương trình (e) ta thu được

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + \frac{h^2}{12} f^{(4)}(x) + \dots$$

hoặc

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + \mathcal{O}(h^2). \quad (5.2)$$

Sử dụng xấp xỉ sai phân trung tâm từ các phương trình (a)-(h) cho các đạo hàm khác được thực hiện tương tự. Ví dụ, khử $f'(x)$ từ các phương trình (f) và (h) ta có

$$f'''(x) = \frac{f(x+2h) - 2f(x+h) + 2f(x-h) - f(x-2h)}{2h^3} + \mathcal{O}(h^2). \quad (5.3)$$

Biểu thức

$$f^{(4)}(x) = \frac{f(x+2h) - 4f(x+h) + 6f(x) - 4f(x-h) + f(x-2h)}{h^4} + \mathcal{O}(h^2) \quad (5.4)$$

nhận được bằng cách khử $f''(x)$ ở các phương trình (e) và (g). Bảng 5.1 tổng hợp các kết quả trên.

	$f(x-2h)$	$f(x-h)$	$f(x)$	$f(x+h)$	$f(x+2h)$
$2hf'(x)$		-1	0	1	
$h^2f''(x)$		1	-2	1	
$2h^3f'''(x)$	-1	2	0	-2	1
$h^4f^{(4)}(x)$	1	-4	6	-4	1

Bảng 5.1: Bảng các hệ số của xấp xỉ sai phân trung tâm với độ chính xác $\mathcal{O}(h^2)$.

B. XẤP XỈ SAI PHÂN MỘT PHÍA BẬC MỘT

Xấp xỉ sai phân trung tâm bậc một không phải lúc nào cũng có thể sử dụng được. Chẳng hạn, xét tình huống mà hàm số được cho trước tại n điểm rác x_1, x_2, \dots, x_n .

Vì sai phân trung tâm sử dụng các giá trị của hàm số tại hai phía của x nên sai phân trung tâm sẽ không thể tính được tại x_1 và x_n . Do vậy, cần thiết phương pháp tính sai phân hữu hạn mà có thể tính đạo hàm mà chỉ yêu cầu giá trị tại một phía của x . Các biểu thức này được gọi là *xấp xỉ sai phân hữu hạn tiến* và *lùi*.

Sai phân hữu hạn một phía có thể thu được từ các phương trình (a)-(h). Giải phương trình (a) theo $f'(x)$ ta có

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{h}{2!} f''(x) - \frac{h^2}{3!} f'''(x) - \frac{h^3}{4!} f^{(4)}(x) - \dots$$

Chỉ giữ lại biểu thức đầu tiên bên về phải ta có *xấp xỉ sai phân tiến bậc một*

$$f'(x) = \frac{f(x+h) - f(x)}{h} + \mathcal{O}(h). \quad (5.5)$$

Tương tự, từ phương trình (b) ta có *xấp xỉ sai phân lùi bậc một*

$$f'(x) = \frac{f(x) - f(x-h)}{h} + \mathcal{O}(h). \quad (5.6)$$

Chú ý rằng các xấp xỉ này có sai số là $\mathcal{O}(h)$ nên không tốt bằng các xấp xỉ trung tâm có sai số $\mathcal{O}(h^2)$.

Bằng cách tương tự, chúng ta có thể xấp xỉ cho các đạo hàm bậc cao. Chẳng hạn, từ phương trình (a) và (c) ta thu được

$$f''(x) = \frac{f(x+2h) - 2f(x+h) + f(x)}{h^2} + \mathcal{O}(h). \quad (5.7)$$

Các đạo hàm bậc ba và bậc bốn được tính toán tương tự. Các kết quả được chỉ ra trong Bảng 5.2 và Bảng 5.3.

	$f(x)$	$f(x+h)$	$f(x+2h)$	$f(x+3h)$	$f(x+4h)$
$hf'(x)$	-1	1			
$h^2 f''(x)$	1	-2	1		
$h^3 f'''(x)$	-1	3	-3	1	
$h^4 f^{(4)}(x)$	1	-4	6	-4	1

Bảng 5.2: Bảng các hệ số của xấp xỉ sai phân tiến với độ chính xác $\mathcal{O}(h)$.

C. XẤP XỈ SAI PHÂN HỮU HẠN MỘT PHÍA BẬC HAI

Những xấp xỉ sai phân hữu hạn một phía với sai số $\mathcal{O}(h)$ không phổ biến vì lý do sai số lớn. Phổ biến trong thực hành là sử dụng biểu thức $\mathcal{O}(h^2)$. Để thu được công

	$f(x - 4h)$	$f(x - 3h)$	$f(x - 2h)$	$f(x - h)$	$f(x)$
$hf'(x)$				-1	1
$h^2 f''(x)$			1	-2	1
$h^3 f'''(x)$		-1	3	-3	1
$h^4 f^{(4)}(x)$	1	-4	6	-4	1

Bảng 5.3: Bảng các hệ số của xấp xỉ sai phân hữu hạn lùi với độ chính xác $\mathcal{O}(h)$.

thức sai phân một phía với sai số này, chúng ta cần giữ lại nhiều hơn các biểu thức trong khai triển Taylor. Chúng ta sẽ minh họa bằng việc xác định biểu thức cho $f'(x)$. Từ phương trình (a) và phương trình (c) ta có

$$\begin{aligned}f(x+h) &= f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f'''(x) + \frac{h^4}{24}f^{(4)}(x) + \dots \\f(x+2h) &= f(x) + 2hf'(x) + 2h^2f''(x) + \frac{4h^3}{3}f'''(x) + \frac{2h^4}{3}f^{(4)}(x) + \dots\end{aligned}$$

Khử $f''(x)$ bằng cách nhân phương trình thứ nhất với -4 rồi cộng với phương trình thứ hai. Ta được kết quả

$$f(x+2h) - 4f(x+h) = -3f(x) - 2hf'(x) + \frac{2h^2}{3}f'''(x) + \dots$$

Do đó

$$f'(x) = \frac{-f(x+2h) + 4f(x+h) - 3f(x)}{2h} + \frac{h^2}{3}f'''(x) + \dots$$

hoặc

$$f'(x) = \frac{-f(x+2h) + 4f(x+h) - 3f(x)}{2h} + \mathcal{O}(h^2). \quad (5.8)$$

Phương trình (5.8) được gọi là *xấp xỉ sai phân hữu hạn tiến bậc hai*.

Để sử dụng xấp xỉ sai phân hữu hạn cho đạo hàm cấp cao hơn thì ta cần bổ sung các chuỗi Taylor. Chẳng hạn, để xấp xỉ sai phân hữu hạn cho $f''(x)$ ta cần sử dụng các chuỗi cho $f(x+h), f(x+2h)$ và $f(x+3h)$; xấp xỉ cho $f'''(x)$ liên quan đến các chuỗi Taylor cho $f(x+h), f(x+2h), f(x+3h)$ và $f(x+4h)$, v.v. Như đã thấy, việc tính toán cho các đạo hàm bậc cao hơn trở nên rất nhầm chán. Các kết quả của hai trường hợp xấp xỉ sai phân hữu hạn tiến và lùi bậc hai được tóm tắt trong Bảng 5.4 và Bảng 5.5

D. SAI SỐ TRONG XẤP XỈ SAI PHÂN HỮU HẠN

Ta để ý thấy rằng trong các biểu thức sai phân hữu hạn thì tổng các hệ số bằng không. Ảnh hưởng của sai số làm tròn có thể trở nên nghiêm trọng. Nếu giá trị của h nhỏ thì $f(x), f(x \pm h), f(x \pm 2h), \dots$ xấp xỉ bằng nhau. Khi chúng nhân với các hệ số

	$f(x)$	$f(x + h)$	$f(x + 2h)$	$f(x + 3h)$	$f(x + 4h)$	$f(x + 5h)$
$2hf'(x)$	-3	4	-1			
$h^2f''(x)$	2	-5	4	-1		
$2h^3f'''(x)$	-5	18	-24	14	-3	
$h^4f^{(4)}(x)$	3	-14	26	-24	11	-2

Bảng 5.4: Bảng các hệ số của xấp xỉ sai phân hữu hạn tiến với độ chính xác $\mathcal{O}(h^2)$.

	$f(x - 5h)$	$f(x - 4h)$	$f(x - 3h)$	$f(x - 2h)$	$f(x - h)$	$f(x)$
$2hf'(x)$				1	-4	3
$h^2f''(x)$			-1	4	-5	2
$2h^3f'''(x)$		3	-14	24	-18	5
$h^4f^{(4)}(x)$	-2	11	-24	26	-14	3

Bảng 5.5: Bảng các hệ số của xấp xỉ sai phân hữu hạn lùi với độ chính xác $\mathcal{O}(h^2)$.

của công thức sai phân hữu hạn và cộng dồn vào thì một vài hệ số quan trọng có thể bị mất. Mặt khác, chúng ta không thể để h quá lớn bởi sẽ làm sai số phương pháp bùng nổ. Tình huống không may này không có cách khắc phục, nhưng chúng ta có thể hạn chế bằng cách thực hiện các biện pháp phòng ngừa sau đây:

- Sử dụng số học chính xác cao (tính toán với nhiều chữ số thập phân).
- Sử dụng công thức sai phân hữu hạn với độ chính xác ít nhất $\mathcal{O}(h^2)$.

Để minh họa cho các sai số, chúng ta tính đạo hàm bậc hai của hàm $f(x) = e^{-x}$ tại $x = 1$ bằng công thức sai phân trung tâm (5.2)

$$f''(x) = \frac{f(x + h) - 2f(x) + f(x - h)}{h^2}$$

Thực hiện tính toán với độ chính xác đến sáu và tám chữ số với các giá trị khác nhau của h . Kết quả được cho trong Bảng 5.6 và được so sánh với giá trị chính xác $f''(1) = e^{-1} = 0.36787944$.

Trong các tính toán sáu chữ số, giá trị tối ưu h là 0,08, mang lại kết quả chính xác cho ba chữ số có nghĩa. Do đó ba chữ số có nghĩa bị mất do sự kết hợp của sai số phương pháp và sai số làm tròn. Kết quả tốt nhất thu được với tính toán tám chữ số là

h	6 chữ số chính xác	8 chữ số chính xác
0.64	0.380610	0.38060911
0.32	0.371035	0.37102939
0.16	0.368711	0.36866484
0.08	0.368281	0.36807656
0.04	0.36875	0.36783125
0.02	0.37	0.3679
0.01	0.38	0.3679
0.005	0.40	0.3676
0.0025	0.48	0.3680
0.00125	11.28	0.3712

Bảng 5.6: Tính e^{-x} tại $x = 1$ bằng công thức sai phân hữu hạn trung tâm.

chính xác đến bốn chữ số có nghĩa. Bởi vì độ chính xác mở rộng tám chữ số làm giảm sai số làm tròn nên giá trị tính toán cho kết quả chính xác hơn so với tính toán sáu chữ số (khoảng 0.02).

VÍ DỤ 5.2.1. Cho các điểm dữ liệu cách đều

x	0	0.1	0.2	0.3	0.4
y	0.0000	0.0819	0.1341	0.1646	0.1797

Tính $f'(x)$, $f''(x)$ tại $x = 0$ và $x = 0.2$ bằng cách sử dụng xấp xỉ sai phân hữu hạn với sai số $\mathcal{O}(h^2)$.

GIẢI. Từ công thức sai phân tiên trong Bảng 5.4 ta có

$$f'(0) = \frac{-3f(0) + 4f(0.1) - f(0.2)}{2(0.1)} = \frac{-3(0) + 4(0.0819) - 0.1341}{0.2} = 0.967.$$

$$\begin{aligned} f''(0) &= \frac{2f(0) - 5f(0.1) + 4f(0.2) - f(0.3)}{0.1^2} \\ &= \frac{2(0) - 5(0.0819) + 4(0.1341) - 0.1646}{0.1^2} = -3.77. \end{aligned}$$

Từ công thức sai phân trung tâm trong Bảng 5.1 ta có

$$f'(0.2) = \frac{-f(0.1) + f(0.3)}{2(0.1)} = \frac{-0.0819 + 0.1646}{0.2} = 0.4135.$$

$$\begin{aligned} f''(0.2) &= \frac{f(0.1) - 2f(0.2) + f(0.3)}{0.1^2} \\ &= \frac{0.0819 - 2(0.1341) + 0.1646}{0.1^2} = -2.17. \end{aligned}$$

□

5.2.3 PHÉP NGOẠI SUY RICHARDSON

Phép ngoại suy Richardson là một phương pháp đơn giản để tăng tính chính xác của một số quá trình giải số, bao gồm cả các xấp xỉ sai phân hữu hạn (chúng ta cũng sẽ sử dụng ngoại suy Richardson sau này khi tính số tích phân).

Giả sử rằng chúng ta có một phương pháp xấp xỉ để tính toán cho đại lượng G . Hơn nữa, giả sử rằng kết quả phụ thuộc vào tham số h . Ký hiệu lượng xấp xỉ bởi $g(h)$, ta có $G = g(h) + E(h)$, với $E(h)$ biểu diễn sai số. Phép ngoại suy Richardson bỏ qua các dạng khác của sai số, giả sử rằng nó có dạng $E(h) = ch^p$, c và p là các hằng số. Chúng ta tính $E(h)$ với một số giá trị của h , chẳng hạn $h = h_1$. Khi đó, ta có

$$G = g(h_1) + ch_1^p. \quad (\text{i})$$

Lặp lại quá trình tính toán với $h = h_2$ ta có

$$G = g(h_2) + ch_2^p. \quad (\text{j})$$

Khử c và giải theo G từ (i) và (j), công thức

$$G = \frac{(h_1/h_2)^p g(h_2) - g(h_1)}{(h_1/h_2)^p - 1} \quad (5.9\text{a})$$

được gọi là *công thức ngoại suy Richardson*. Trong thực hành tính toán, ta thay $h_2 = h_1/2$ để được công thức

$$G = \frac{2^p g(h_1/2) - g(h_1)}{2^p - 1}. \quad (5.9\text{b})$$

Chúng ta minh họa cho phép ngoại suy Richardson bằng cách áp dụng cho xấp xỉ sai phân hữu hạn cho $f(x) = e^{-x}$ tại $x = 1$. Thực hiện tính toán với sáu chữ số chính xác và sử dụng kết quả trong Bảng 5.6. Do phép ngoại suy chỉ làm tăng tính chính xác với sai số phương pháp (truncation error) nên chúng ta giới hạn h nhận các giá trị với sai số làm tròn nhỏ. Chọn $h_1 = 0.64$ và đặt $g(h)$ là xấp xỉ của $f'(1)$ thu được tại h . Từ Bảng 5.6 ta có

$$g(h_1) = 0.380610 \quad g(h_1/2) = 0.371035.$$

Sai số phương pháp trong phép xấp xỉ sai phân trung tâm là $E(h) = \mathcal{O}(h^2) = c_1 h^2 + c_2 h^4 + c_3 h^6 + \dots$. Do đó, chúng ta có thể khử thành phần bậc một nếu ta thay $p = 2$ và $h_1 = 0.64$ vào phương trình (5.9b). Kết quả là

$$G = \frac{2^2 g(0.32) - g(0.64)}{2^2 - 1} = \frac{4(0.371035) - 0.380610}{3} = 0.367843.$$

Đây là xấp xỉ của $(e^{-x})''$ với sai số $\mathcal{O}(h^4)$. Chú ý rằng ta cũng được kết quả tốt hơn nên thực hiện tương với tám chữ số.

VÍ DỤ 5.2.2. Hãy sử dụng dữ liệu trong VÍ DỰ 5.2.1 để tính $f'(0)$ chính xác nhất mà bạn có thể.

GIẢI. Một giải pháp để tăng độ chính xác là áp dụng ngoại suy Richardson cho xấp xỉ sai phân hữu hạn. Chúng ta bắt đầu với hai xấp xỉ sai phân tiến cho $f'(0)$: một áp dụng với $h = 0.2$ và một với $h = 0.1$. Dựa vào công thức cho $\mathcal{O}(h^2)$ trong Bảng 5.4 ta có

$$\begin{aligned} f'(0.2) &= \frac{-3f(0) + 4f(0.2) - f(0.4)}{2(0.2)} = \frac{-3(0) + 4(0.1341) - 0.1797}{0.4} = 0.8918 \\ f'(0.1) &= \frac{-3f(0) + 4f(0.1) - f(0.2)}{2(0.1)} = \frac{-3(0) + 4(0.0819) - 0.1341}{0.2} = 0.9675. \end{aligned}$$

với g ký hiệu xấp xỉ sai phân hữu hạn của $f'(0)$. Nhắc lại là sai số của cả hai xấp xỉ có dạng $E(h) = c_1 h^2 + c_2 h^4 + c_3 h^6 + \dots$, chúng ta sẽ sử dụng ngoại suy Richardson để khử biểu thức sai số trội. Với $p = 2$ ta thu được từ phương trình (5.9)

$$f'(0) \approx G = \frac{2^2 g(0.1) - g(0.2)}{2^2 - 1} = \frac{4(0.9675) - 0.8918}{3} = 0.9927$$

là xấp xỉ sai phân hữu hạn với sai số $\mathcal{O}(h^4)$.

□

5.2.4 XẤP XỈ ĐẠO HÀM BẰNG NỘI SUY

Nếu $f(x)$ cho dưới dạng tập các điểm dữ liệu rời rạc thì nội suy có thể cho ta một phương pháp hiệu quả để xấp xỉ đạo hàm của nó. Ý tưởng là xấp xỉ đạo hàm của $f(x)$ bằng đạo hàm của hàm nội suy. Phương pháp này đặc biệt hữu ích nếu các điểm dữ liệu không cách đều, khi đó các phương pháp xấp xỉ sai phân hữu hạn không thể áp dụng được.

Ta xấp xỉ đa thức bậc $n - 1$

$$P_{n-1}(x) = a_1 x^{n-1} + a_2 x^{n-2} + \dots + a_n \quad (\text{a})$$

đi qua n điểm dữ liệu và sau đó tính đạo hàm của $P_{n-1}(x)$ tại điểm x . Như đã chỉ ra trong Chương 4, thông thường nên giới hạn bậc của đa thức dưới sáu để tránh sự dao động quá lớn của hàm nội suy gây nên sai số. Vì các sai số này được khuếch đại với mỗi sai phân nên ảnh hưởng của chúng có thể rất nguy hiểm. Vì yêu cầu giới hạn này, phép nội suy thường áp dụng trong lân cận địa phương vài điểm dữ liệu lân cận gần nhất.

Với các điểm dữ liệu cách đều thì nội suy đa thức và xấp xỉ sai phân hữu hạn cho kết quả đồng nhất. Trong thực tế thì công thức sai phân hữu hạn tương đương với nội suy đa thức.

Một vài phương pháp nội suy đa thức đã được giới thiệu trong Chương 4 nhưng thật không may là chúng không phù hợp cho việc tính đạo hàm. Phương pháp chúng ta cần là một phương pháp xác định các hệ số a_1, a_2, \dots, a_n của đa thức trong công thức (a). Chỉ có một phương pháp xấp xỉ như vậy được thảo luận trong Chương 4 là phương pháp bình phương nhỏ nhất. Mặc dù mục đích của phương pháp này là làm tròn dữ liệu nhưng ta có thể thực hiện phép nội suy bằng cách lấy $m = n$ trong công thức (4.19). Nếu dữ liệu có nhiều thì phương pháp xấp xỉ bình phương nhỏ nhất nên được sử dụng với $m < n$. Sau khi các hệ số của đa thức nội suy được xác định thì ta có thể tính hai đạo hàm đầu tiên bằng hàm `evalpoly` trong Chương 4.

VÍ DỤ 5.2.3. Cho dữ liệu

x	1.5	1.9	2.1	2.4	2.6	3.1
y	1.0628	1.3961	1.5432	1.7349	1.8423	2.0397

tính $f'(2)$ và $f''(2)$ bằng cách sử dụng đa thức nội suy tại ba điểm lân cận gần nhất.

GIẢI. Gọi đa thức nội suy đi qua ba điểm 1.9; 2.1 và 2.4 là $P_2(x) = a_1 + a_2x + a_3x^2$.

Hệ phương trình chuẩn theo công thức (4.20) của xấp xỉ bình phương nhỏ nhất là

$$\begin{bmatrix} 3 & \sum x_i & \sum x_i^2 \\ \sum x_i & \sum x_i^2 & \sum x_i^3 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum x_i y_i \\ \sum x_i^2 y_i \end{bmatrix}.$$

Sau khi thay dữ liệu, ta có

$$\begin{bmatrix} 3 & 6.4 & 13.78 \\ 6.4 & 13.78 & 29.944 \\ 13.78 & 29.944 & 65.6578 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 4.6742 \\ 10.0571 \\ 21.8385 \end{bmatrix}$$

với $\mathbf{a} = [-0.7714, 1.5075, -0.1930]$. Vậy, đa thức nội suy và các đạo hàm của nó là

$$\begin{aligned}P_2(x) &= -0.1930x^2 + 1.5075x - 0.7714 \\P'_2(x) &= -0.3860x + 1.5075 \\P''_2(x) &= -0.3860\end{aligned}$$

nên ta có

$$\begin{aligned}f'(2) &\approx P'_2(2) = -0.3860(2) + 1.5075 = 0.7355 \\f''(2) &\approx P''_2(2) = -0.3860.\end{aligned}$$

□

VÍ DỤ 5.2.4. Xác định $f'(0)$ và $f'(1)$ từ dữ liệu có nhiều sau

x	0	0.2	0.4	0.6
$f(x)$	1.9934	2.1465	2.2129	2.1790
x	0.8	1.0	1.2	1.4
$f(x)$	2.0683	1.9448	1.7655	1.5891

GIẢI. Chúng ta sử dụng chương trình trong VÍ DỰ 4.3.2 để tìm đa thức xấp xỉ tốt nhất (bằng phương pháp bình phương nhỏ nhất) cho dữ liệu. Kết quả là

degree of polynomial = 2

coeff =

-7.0240e-001

6.4704e-001

2.0262e+000

sigma =

3.6097e-002

degree of polynomial = 3

coeff =

4.0521e-001

-1.5533e+000

1.0928e+000

1.9921e+000

sigma =

8.2604e-003

degree of polynomial = 4

coeff =

-1.5329e-002

4.4813e-001

-1.5906e+000

1.1028e+000

1.9919e+000

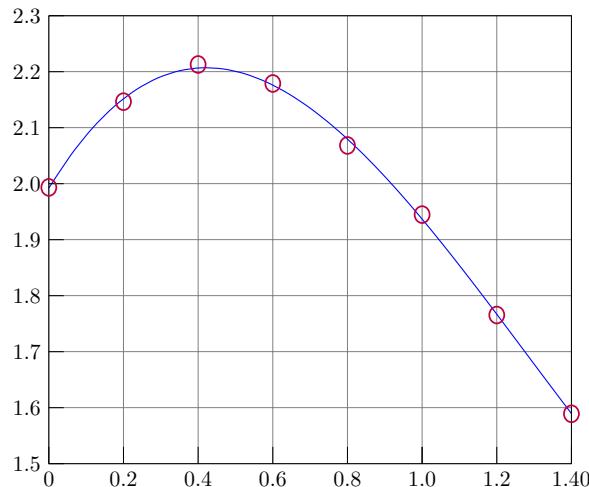
sigma =

9.5193e-003

degree of polynomial =

Done

Dựa vào độ lệch chuẩn, đường bậc ba dường như là xấp xỉ tốt nhất. Trước khi khẳng định, chúng ta vẽ các điểm dữ liệu và đường nội suy trên một hệ trục tọa độ như hình dưới. Ta thấy xấp xỉ thỏa mãn.



Xấp xỉ $f(x)$ bởi đa thức nội suy ta có

$$f(x) \approx a_1x^3 + a_2x^2 + a_3x + a_4$$

nên

$$f'(x) \approx 3a_1x^2 + 2a_2x + a_3.$$

Do đó

$$f'(0) \approx a_3 = 1.093,$$

$$f'(1) \approx 3a_1 + 2a_2 + a_3 = 3(0.405) + 2(-1.553) + 1.093 = -0.798.$$

□

5.3 TÍNH GẦN ĐÚNG TÍCH PHÂN XÁC ĐỊNH

5.3.1 MỞ ĐẦU

Tính số tích phân, còn được gọi là *luật cầu phương*, là quá trình tính số chính xác hơn tính số vi phân. Luật cầu phương xấp xỉ tích phân

$$\int_a^b f(x)dx$$

bởi tổng

$$I = \sum_{i=1}^n A_i f(x_i)$$

với các *nút* x_i và các *trọng số* A_i phụ thuộc vào từng phương pháp cụ thể sử dụng cho luật cầu phương. Tất cả các phương pháp cầu phương đều được bắt nguồn từ việc nội suy đa thức hàm dưới dấu tích phân. Do đó, việc xấp xỉ tốt nhất là nếu $f(x)$ là một đa thức.

Các phương pháp tính số tích phân được chia ra thành hai nhóm: Các công thức Newton-Cotes và các công thức cầu phương Gaussian. Các công thức Newton-Cotes được đặc trưng bởi các nút cách đều và bao gồm cả các phương pháp nổi tiếng như phương pháp hình thang, phương pháp Simpson. Chúng rất hữu ích khi hàm $f(x)$ đã được tính toán tại các khoảng bằng nhau vì có thể tính toán với khối lượng phép tính ít. Vì các công thức Newton-Cotes dựa vào nội suy nội bộ nên chúng là các hàm ghép tron đa thức.

Trong các phương pháp cầu phương Gaussian, các nút được chọn sao cho độ chính xác là tốt nhất. Bởi vì phương pháp cầu phương Gaussian yêu cầu ít hơn các giá trị của hàm dưới dấu tích phân cho mỗi mức độ chính xác cho trước nên nó rất thuận lợi trong trường hợp tính các giá trị của hàm $f(x)$ phức tạp. Một ưu điểm khác của phương pháp cầu phương Gaussian là có khả năng tính khi hàm dưới dấu tích phân có

điểm kỳ dị, chẳng hạn cho phép đánh giá biểu thức

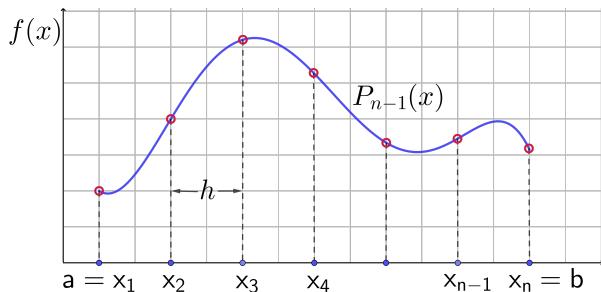
$$\int_0^1 \frac{g(x)}{\sqrt{1-x^2}} dx$$

với điều kiện hàm $g(x)$ đủ tốt.

5.3.2 CÁC CÔNG THỨC NEWTON-COTES

Xét tích phân xác định

$$I = \int_a^b f(x) dx. \quad (5.10)$$



**Hình 5.1: Đa thức
xấp xỉ hàm $f(x)$**

Ta chia miền lấy tích phân (a, b) thành $n - 1$ đoạn bằng nhau với chiều dài $h = (b-a)/(n-1)$ như Hình 5.1 và ký hiệu toạ độ trên trục hoành bởi các nút x_1, x_2, \dots, x_n . Tiếp theo, ta xấp xỉ hàm $f(x)$ bởi đa thức bậc $n - 1$ đi qua tất cả các nút. Công thức nội suy Lagrange trong (4.1) là

$$P_{n-1}(x) = \sum_{i=1}^n y_i l_i(x),$$

với $l_i(x)$ là các hàm cơ bản được xác định trong (4.2). Do đó, một xấp xỉ của tích phân trong (5.10) là

$$I = \int_a^b P_{n-1}(x) dx = \sum_{i=1}^n \left[f(x_i) \int_a^b l_i(x) dx \right] = \sum_{i=1}^n A_i f(x_i), \quad (5.11a)$$

với

$$A_i = \int_a^b l_i(x) dx, \quad i = 1, 2, \dots, n. \quad (5.11b)$$

Các công thức (5.11) được gọi là *công thức Newton-Cotes*. Một số ví dụ cở điểm của những công thức này là *công thức hình thang* ($n=2$), *công thức Simpson 1/3* ($n=3$) và

công thức Simpsom 3/8 ($n=4$). Quan trọng nhất trong các công thức này là công thức hình thang. Nó có thể kết hợp với phép ngoại suy Richardson để được một thuật toán hiệu quả hơn được biết đến với tên gọi là *tích phân Romberg*.

A. CÔNG THỨC HÌNH THANG

Nếu $n = 2$, ta có $l_1 = (x - x_2)/(x_1 - x_2) = -(x - b)/h$. Do đó

$$A_1 = -\frac{1}{h} \int_a^b (x - b)dx = \frac{1}{2h}(b - a)^2 = \frac{h}{2}.$$

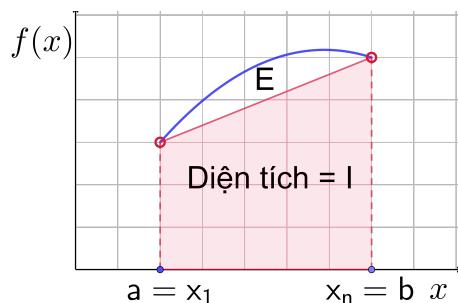
Tương tự, $l_2 = (x - x_1)/(x_2 - x_1) = (x - a)/h$ nên

$$A_2 = \frac{1}{h} \int_a^b (x - a)dx = \frac{1}{2h}(b - a)^2 = \frac{h}{2}.$$

Thay vào phương trình (5.11a) ta có công thức

$$I = [f(a) + f(b)] \frac{h}{2} \quad (5.12)$$

là công thức được biết đến với tên gọi *công thức hình thang*. Về ý nghĩa hình học, công thức hình thang là việc xấp xỉ hình thang cong bởi một hình thang có cùng các đỉnh như minh họa trong Hình 5.2.



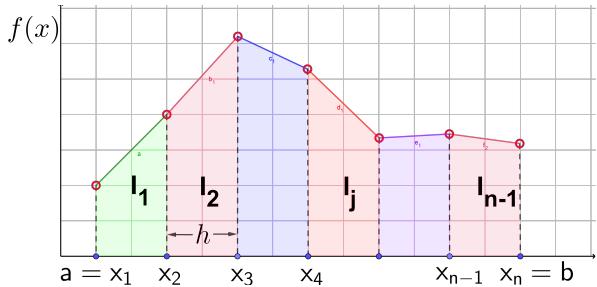
**Hình 5.2: Đồ thị
minh họa công thức
hình thang**

Sai số của công thức hình thang

$$E = \int_a^b f(x)dx - I$$

là phần diện tích giới hạn bởi đường cong $f(x)$ và đường thẳng nội suy được minh họa trong Hình 5.2. Từ (4.4) ta có

$$\begin{aligned} E &= \frac{1}{2!} \int_a^b (x - x_1)(x - x_2)f''(\xi)dx = \frac{1}{2}f''(\xi) \int_a^b (x - a)(x - b)dx \\ &= -\frac{1}{12}(b - a)^3 f''(\xi) = -\frac{h^3}{12} f''(\xi). \end{aligned} \quad (5.13)$$



**Hình 5.3: Đồ thị
minh họa công thức
hình thang kết hợp**

B. CÔNG THỨC HÌNH THANG KẾT HỢP

Trong thực hành, công thức hình thang được áp dụng cho các hàm ghép trơn. Chia miền lấy tích phân (a, b) thành $n - 1$ đoạn có độ rộng h như hình 5.3. Hàm dưới dấu tích phân $f(x)$ được xấp xỉ bởi đường thẳng ở mỗi đoạn. Từ công thức hình thang, ta thu được xấp xỉ của đoạn thứ i

$$I_i = [f(x_i) + f(x_{i+1})] \frac{h}{2}.$$

Do đó, tổng diện tích, xấp xỉ cho $\int_a^b f(x)dx$, là

$$I = \sum_{i=1}^{n-1} I_i = [f(x_1) + 2f(x_2) + 2f(x_3) + \cdots + 2f(x_{n-1}) + f(x_n)] \frac{h}{2} \quad (5.14)$$

được gọi là *công thức hình thang kết hợp*. Quá trình tính tích phân xác định bằng công thức hình thang kết hợp được minh họa bằng sơ đồ khối trong Hình 5.4.

Từ (5.13), sai số phương pháp của mỗi đoạn là

$$E_i = -\frac{h^3}{12} f''(\xi_i)$$

với ξ_i nằm trong khoảng (x_i, x_{i+1}) . Do đó, sai số của (5.14) là

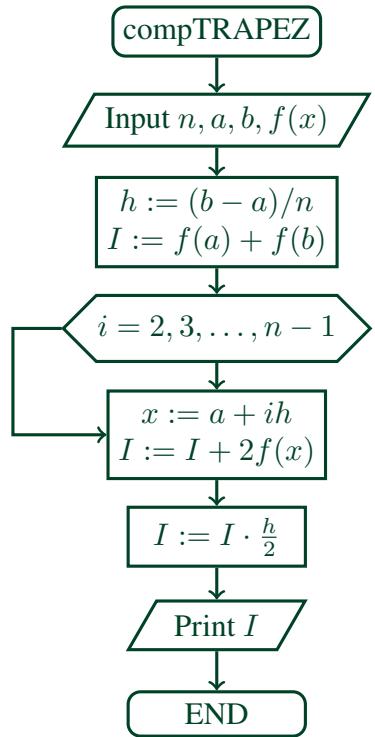
$$E = \sum_{i=1}^{n-1} E_i = -\frac{h^3}{12} \sum_{i=1}^{n-1} f''(\xi_i). \quad (a)$$

Nhưng

$$\sum_{i=1}^{n-1} f''(\xi_i) = (n-1) \bar{f}''$$

với \bar{f}'' là trung bình số học của các đạo hàm cấp hai. Nếu $f''(x)$ liên tục thì tồn tại ξ trong (a, b) mà tại đó ta có $f''(\xi) = \bar{f}''$. Ta có thể viết lại biểu thức trên dưới dạng

$$\sum_{i=1}^{n-1} f''(\xi_i) = (n-1) f''(\xi) = \frac{b-a}{h} f''(\xi).$$



Hình 5.4: Sơ đồ khối của phương pháp hình thang kết hợp tính tích phân xác định

Do đó, phương trình (a) trở thành

$$E = -\frac{(b-a)h^2}{12}f''(\xi). \quad (5.15)$$

Sẽ là sai lầm nếu từ (5.15) ta kết luận rằng $E = ch^2$ (c là hằng số) bởi vì $f''(\xi)$ không hoàn toàn độc lập với h . Một phân tích kỹ hơn¹ đã chỉ ra rằng nếu $f(x)$ và các đạo hàm của nó hữu hạn trên (a, b) thì

$$E = c_1h^2 + c_2h^4 + c_3h^6 + \dots \quad (5.16)$$

VÍ DỤ 5.3.1. Xấp xỉ tích phân $\int_0^\pi \sin(x)dx$ bằng công thức hình thang kết hợp sử dụng: (1) 8 đoạn và (2) 16 đoạn.

GIẢI.

Trường hợp (1). Với 8 đoạn ta có 9 nút với khoảng cách là $h = \pi/8$. Hoành độ của các nút là $x_i = (i-1)\pi/8$, $i = 1, 2, \dots, 9$. Từ (5.14) ta có

$$I = \left[\sin 0 + 2 \sum_{i=2}^9 \sin \frac{i\pi}{8} + \sin \pi \right] \frac{\pi}{16} = 1.97423.$$

Từ (5.15) ta có sai số

$$E = -\frac{(b-a)h^2}{12}f''(\xi) = -\frac{(\pi-0)(\pi/8)^2}{12} \sin \xi = \frac{\pi^3}{768} \sin \xi,$$

¹xem chứng minh trong [2]

với $0 < \xi < \pi$. Do ta không biết giá trị của ξ nên ta không thể tính E nhưng ta có thể ước lượng cận cho nó

$$E_{min} = \frac{\pi^3}{768} \sin 0 = 0; \quad E_{max} = \frac{\pi^3}{768} \sin \frac{\pi}{2} = 0.04037.$$

Do đó $I + E_{min} < \int_0^\pi \sin(x)dx < I + E_{max}$ hay

$$1.97423 < \int_0^\pi \sin(x)dx < 2.01460.$$

Đĩ nhiên, ta có thể tính kết quả chính xác của tích phân này là 2.

Trường hợp (2). Các nút mới được tạo ra bằng cách lấy điểm giữa của các nút cũ. Chúng có hoành độ là

$$x_j = \frac{\pi}{16} + (j-1)\frac{\pi}{8} = (2j-1)\frac{\pi}{16}, j = 1, 2, \dots, 8.$$

Sử dụng công thức hình thang đệ quy trong (5.18b), ta có

$$I = \frac{1.97423}{2} + \frac{\pi}{16} \sum_{j=1}^8 \sin \frac{(2j-1)\pi}{16} = 1.99358$$

và các cận của sai số là (sai số giảm 4 lần khi h giảm một nửa) $E_{min} = 0, E_{max} = 0.04037/4 = 0.01009$. Vậy

$$1.99358 < \int_0^\pi \sin(x)dx < 2.00367.$$

□

Ta thấy rằng nếu sử dụng công thức (5.15) để đánh giá sai số cho công thức hình thang kết hợp trong (5.14) liên quan đến việc ước lượng giá trị của đạo hàm bậc hai của $f(x)$. Việc này phải tính toán bằng tay không phải lúc nào cũng thực hiện dễ dàng. Thuật toán sau cải tiến phương pháp hình thang kết hợp để quá trình tính toán sẽ dừng khi sai lệch giữa hai lần lặp liên tiếp nhỏ hơn sai số cho phép.

C. CÔNG THỨC HÌNH THANG ĐỆ QUY

Cho I_k là giá trị tích phân khi sử dụng công thức hình thang kết hợp trên 2^{k-1} đoạn. Chú ý rằng nếu k tăng một đơn vị thì số đoạn tăng gấp đôi. Sử dụng ký hiệu

$$H = b - a$$

ta thu được từ (5.14) các kết quả sau với $k = 1, 2, \text{ và } 3$

$k = 1$ (1 đoạn):

$$I_1 = [f(a) + f(b)] \frac{H}{2} \quad (5.17)$$

$k = 2$ (4 đoạn):

$$I_2 = \left[f(a) + 2f\left(a + \frac{H}{2}\right) + f(b) \right] \frac{H}{4} = \frac{1}{2}I_1 + f\left(a + \frac{H}{2}\right) \frac{H}{2}$$

$k = 3$ (8 đoạn):

$$\begin{aligned} I_3 &= \left[f(a) + 2f\left(a + \frac{H}{4}\right) + 2f\left(a + \frac{H}{2}\right) + 2f\left(a + \frac{3H}{4}\right) + f(b) \right] \frac{H}{8} \\ &= \frac{1}{2}I_2 + \left[f\left(a + \frac{H}{4}\right) + f\left(a + \frac{3H}{4}\right) \right] \frac{H}{4}. \end{aligned}$$

Ta thấy rằng với $k > 1$ tùy ý ta có công thức

$$I_k = \frac{1}{2}I_{k-1} + \frac{H}{2^{k-2}} \sum_{i=1}^{2^{k-1}} f\left[a + \frac{(2i-1)H}{2^{k-1}}\right], k = 2, 3, \dots \quad (5.18a)$$

được gọi là *công thức hình thang đệ quy*. Quan sát thấy rằng tổng trên chỉ chứa các nút mới được tạo ra khi số lượng nút đã được nhân đôi. Do đó, việc tính toán dãy $I_1, I_2, I_3, \dots, I_k$ từ (5.17) và (5.18) có cùng số lượng phép tính bằng với cách tính trực tiếp trong công thức (5.14). Ưu điểm của công thức hình thang đệ quy là nó cho phép ta quan sát được sự hội tụ và kết thúc quá trình tính toán khi mà sai lệch giữa hai lần liên tiếp I_{k-1} và I_k đủ nhỏ. Từ (5.18a) ta có

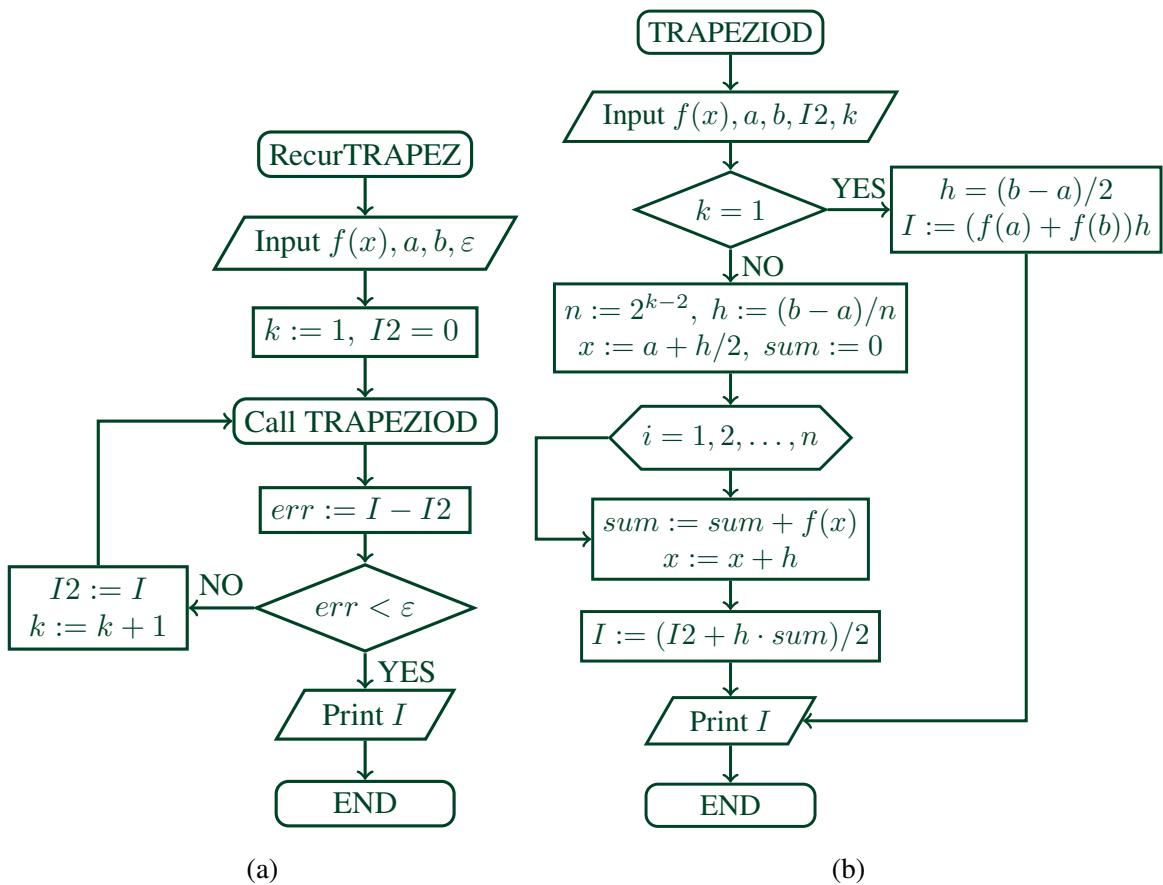
$$I(h) = \frac{1}{2}I(2h) + h \sum f(x_{new}), \quad (5.18b)$$

với $h = H/(n-1)$ là chiều rộng của mỗi đoạn. Quá trình tính toán của phương pháp hình thang đệ quy được minh họa bằng sơ đồ khối trong Hình 5.5a

 trapezoid

Hàm trapezoid tính $I(h)$ với $I(2h)$ cho trước từ các phương trình (5.17) và (5.18). Chúng ta có thể tính $\int_a^b f(x)dx$ bằng cách gọi trapezoid lặp với $k = 1, 2, \dots$ đến khi đạt độ chính xác cần thiết.

```
function Ih = trapezoid(func, a, b, I2h, k)
% Recursive trapezoidal rule.
% USAGE: Ih = trapezoid(func, a, b, I2h, k)
% func = handle of function being integrated.
% a, b = limits of integration.
```



Hình 5.5: Sơ đồ khôi của phương pháp hình thang để quy tính tích phân xác định

```

% I2h = integral with 2^(k-1) panels.
% Ih = integral with 2^k panels.

if k==1
    fa = feval(func,a); fb = feval(func,b);
    Ih=(fa+fb)*(b-a)/2.0;
else
    n = 2^(k - 2); % Number of new points
    h=(b-a)/n; % Spacing of new points
    x=a+h/2.0; % Coord.of 1st new point
    sum=0.0;
    for i=1:n
        fx = feval(func,x);
        sum=sum+fx;
        x=x+h;
    end
    Ih = (I2h + h*sum)/2.0;
end
    
```

end

VÍ DỤ 5.3.2. Sử dụng công thức hình thang để quy ước lượng $\int_0^\pi \sqrt{x} \cos x dx$ với 6 số thập phân. Có bao nhiêu nút cần thiết để đạt được kết quả này?

GIẢI. Chương trình sau đây thực hiện lặp hàm trapezoid tối đa 20 lần để tính tích phân đã cho đến khi sai số giữa hai lần lặp liên tiếp nhỏ hơn $1.0e-6$. Ngoài việc in ra giá trị tích phân, nó còn biểu diễn số các nút cần thiết trong tính toán.

```
>> % Example 5.3.2 (Recursive trapezoidal rule)
format long % Display extra precision
I2h=0;
for k=1:20
Ih = trapezoid(@fex5_3_2,0,pi,I2h,k);
if (k>1 & abs(Ih-I2h)<1.0e-6)
Integral = Ih
No_of_func_evaluations = 2^(k-1) + 1
return
end
I2h=Ih;
end
error('Too many iterations')

Integral =
-0.894831664853286

No_of_func_evaluations =
32769
??? Too many iterations
```

Với M-file tính hàm dưới dấu tích phân là

```
function y=fex5_3_2(x)
% Function used in Example 5.3.2
y = sqrt(x)*cos(x);
```

Làm tròn đến sáu chữ số thập phân ta có $\int_0^\pi \sqrt{x} \cos x dx = -0.894$. Số lượng nút trong tính toán này là rất lớn, 32769 nút. Tốc độ hội tụ chậm là do đạo hàm của hàm $f(x)$ bị suy biến tại $x = 0$. Do đó, sai số không được biểu diễn như (5.17),

$E = c_1 h^2 + c_2 h^4 + c_3 h^6 + \dots$, mà không thể tính trước được. Vấn đề khó khăn này thường được giải quyết bằng cách đổi biến. Trong trường hợp này, ta đặt $t = \sqrt{x}$, khi đó $dt = dx/(2\sqrt{x}) = dx/(2t)$ hoặc $dx = 2tdt$. Do đó

$$\int_0^\pi \sqrt{x} \cos x dx = \int_0^{\sqrt{\pi}} 2t^2 \cos t^2 dt.$$

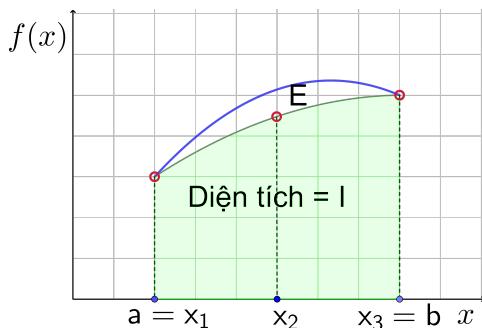
Thực hiện tính toán cho tích phân về phải, ta đạt được kết quả với độ chính xác mong muốn mà chỉ cần có 4097 nút.

□

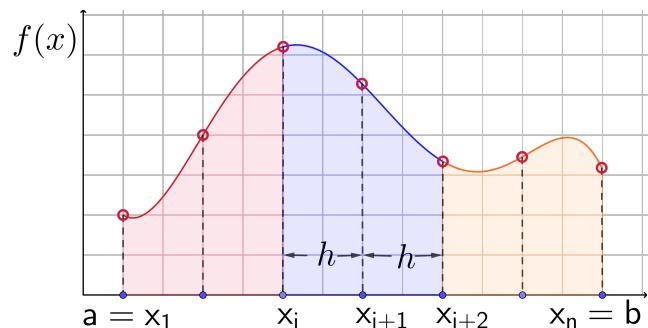
D. CÔNG THỨC SIMPSON

Công thức Simpson 1/3 có thể nhận được từ công thức Newton-Cotes khi lấy $n = 3$; đó là kết quả của việc sử dụng parabol nội suy qua ba điểm lân cận như minh họa trong Hình 5.6a. Diện tích phía dưới parabol, dùng để xấp xỉ tích phân $\int_a^b f(x)dx$, được tính bởi

$$I = \left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b) \right] \frac{b-a}{3}. \quad (\text{a})$$



(a) Công thức Simpson 1/3



(b) Công thức Simpson 1/3 kết hợp

Hình 5.6: Đồ thị minh họa quá trình tích phân xác định bằng công thức Simpson

Để thu được công thức Simpson 1/3 kết hợp, khoảng lıyla tích phân (a, b) được chia thành $n - 1$ đoạn (n lẻ) với độ rộng mỗi đoạn là $h = (b - a)/(n - 1)$, xem minh họa trong Hình 5.6b. Áp dụng (a) cho hai đoạn liền kề liên tiếp ta có

$$\int_{x_i}^{x_{i+2}} f(x)dx \approx [f(x_i) + 4f(x_{i+1}) + f(x_{i+2})] \frac{h}{3}. \quad (\text{b})$$

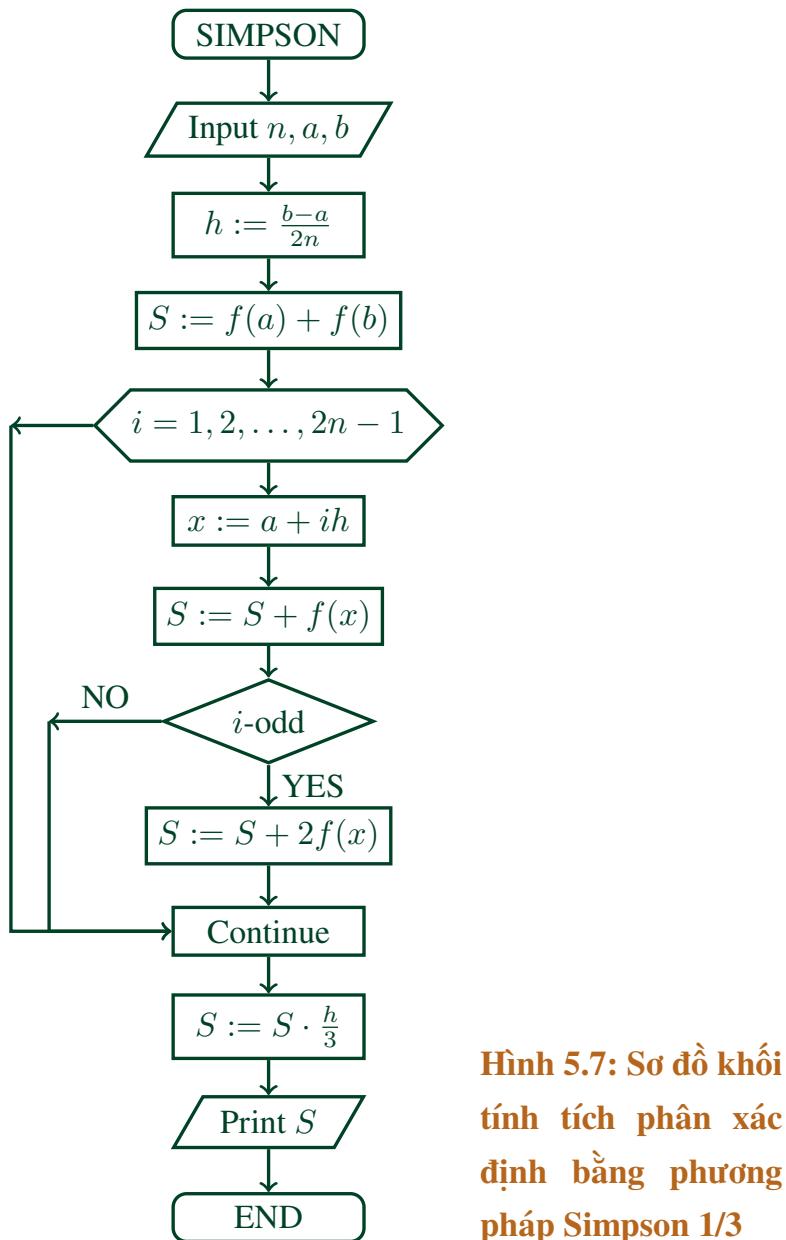
Thay (b) vào

$$\int_a^b f(x)dx = \int_{x_1}^{x_n} f(x)dx = \sum_{i=1,3,\dots}^{n-2} \left[\int_{x_i}^{x_{i+2}} f(x)dx \right]$$

thu được

$$\int_a^b f(x)dx \approx I = [f(x_1) + 4f(x_2) + 2f(x_3) + 4f(x_4) + \cdots + \\ + 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)] \frac{h}{3}. \quad (5.19)$$

Công thức Simpson 1/3 có lẽ là công thức nổi tiếng nhất để tính số tích phân. Quá trình tính toán của công thức Simpson 1/3 được minh họa bởi sơ đồ khôi trong Hình 5.7.



**Hình 5.7: Sơ đồ khôi
tính tích phân xác
định bằng phương
pháp Simpson 1/3**

Sai số của phương pháp Simpson kết hợp là

$$E = \frac{(b-a)h^4}{180} f^{(4)}(\xi). \quad (5.20)$$

Từ đây ta thấy công thức (5.19) là công thức tính đúng nếu hàm $f(x)$ là đa thức có bậc nhỏ hơn hoặc bằng ba.

Công thức Simpson 1/3 yêu cầu số đoạn là chẵn. Nếu điều kiện này không thoả mãn, chúng ta có thể tính tích phân trên ba đoạn đầu tiên bằng công thức Simpson 3/8

$$I = [f(x_1) + 3f(x_2) + 3f(x_3) + f(x_4)] \frac{3h}{8} \quad (5.21)$$

và sử dụng công thức Simpson 1/3 cho các đoạn còn lại. Sai số của (5.21) có bậc như sai số của (5.19).

VÍ DỤ 5.3.3. Xây dựng công thức Simpson 1/3 từ công thức Newton-Cotes.

GIẢI. Xem lại Hình 5.6a ta thấy công thức Simpson sử dụng ba điểm là $x_1 = a$, $x_2 = (a + b)/2$ và $x_3 = b$. Khoảng cách giữa các điểm là $h = (b - a)/2$. Các hàm cơ bản Lagrange đi qua ba điểm này là

$$l_1(x) = \frac{(x - x_2)(x - x_3)}{(x_1 - x_2)(x_1 - x_3)}; l_2(x) = \frac{(x - x_1)(x - x_3)}{(x_2 - x_1)(x_2 - x_3)}; l_3(x) = \frac{(x - x_1)(x - x_2)}{(x_3 - x_1)(x_3 - x_2)}.$$

Từ (5.11b), các trọng số được tính bởi

$$A_1 = \int_a^b l_1(x) dx = \frac{h}{3}; A_2 = \int_a^b l_2(x) dx = \frac{4h}{3}; A_3 = \int_a^b l_3(x) dx = \frac{h}{3}.$$

Từ (5.11a) ta có công thức

$$I = \sum_{i=1}^3 A_i f(x_i) = \left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b) \right] \frac{h}{3}$$

là công thức Simpson 1/3.

□

VÍ DỤ 5.3.4. Ước lượng $\int_0^{2.5} f(x) dx$ từ dữ liệu

x	0.0	0.5	1.0	1.5	2.0	2.5
$f(x)$	1.5000	2.0000	2.0000	1.6364	1.2500	0.9565

GIẢI. Chúng ta sẽ sử dụng công thức Simpson vì có độ chính xác cao hơn công thức hình thang. Vì có 5 đoạn nên ta sẽ áp dụng công thức Simpson 3/8 cho 3 đoạn đầu tiên và sử dụng công thức Simpson 1/3 cho hai đoạn sau:

$$\begin{aligned} I &= [f(0) + 3f(0.5) + 3f(1.0) + f(1.5)] \frac{3(0.5)}{8} \\ &\quad + [f(1.5) + 4f(2.0) + f(2.5)] \frac{0.5}{3} \\ &= 2.8381 + 1.2655 = 4.1036. \end{aligned}$$

□

5.3.3 TÍCH PHÂN ROMBERG

Công thức tính tích phân Romberg là sự kết hợp của công thức hình thang đệ quy với phép ngoại suy Richardson. Trước tiên, ta ký hiệu

$$R_{i,1} = I_i,$$

với I_i là giá trị xấp xỉ tích phân $\int_a^b f(x)dx$ tính bằng công thức đệ quy sử dụng 2^{i-1} đoạn. Nhắc lại rằng sai số của xấp xỉ này là $E = c_1 h^2 + c_2 h^4 + c_3 h^6 + \dots$, với

$$h = \frac{b-a}{2^{i-1}}$$

là chiều rộng của mỗi đoạn.

Tích phân Romberg bắt đầu bằng việc tính $R_{1,1} = I_1$ (1 đoạn) và $R_{2,1} = I_2$ (2 đoạn) bằng công thức hình thang. Biểu thức sai số thứ nhất $c_1 h^2$ được khử bởi phép ngoại suy Richardson. Sử dụng $p = 2$ (số mũ trong biểu thức sai số) trong (5.9b) và ký hiệu kết quả là $R_{2,2}$ ta có

$$R_{2,2} = \frac{2^2 R_{2,1} - R_{1,1}}{2^2 - 1} = \frac{4}{3} R_{2,1} - \frac{1}{3} R_{1,1}. \quad (\text{a})$$

Để thuận tiện thì kết quả được lưu dưới dạng mảng

$$\begin{bmatrix} R_{1,1} \\ R_{2,1} & R_{2,2} \end{bmatrix}.$$

Bước tiếp theo tính $R_{3,1} = I_3$ (4 đoạn) và lặp ngoại suy Richardson với $R_{2,1}$ và $R_{3,1}$, sau đó lưu trực kết quả ra $R_{3,2}$:

$$R_{3,2} = \frac{4}{3} R_{3,1} - \frac{1}{3} R_{2,1}. \quad (\text{b})$$

Các phần tử của mảng \mathbb{R} đã tính toán là

$$\begin{bmatrix} R_{1,1} \\ R_{2,1} & R_{2,2} \\ R_{3,1} & R_{3,2} \end{bmatrix}.$$

Hai phần tử ở cột hai đều có sai số dạng $c_2 h^4$ nên ta có thể sử dụng phép ngoại suy Richardson để khử. Sử dụng $p = 4$ trong (5.9b), ta có

$$R_{3,3} = \frac{2^4 R_{3,2} - R_{2,2}}{2^4 - 1} = \frac{16}{15} R_{3,2} - \frac{1}{15} R_{2,2}. \quad (\text{c})$$

Kết quả này có sai số là $\mathcal{O}(h^6)$. Mảng bây giờ được mở rộng thành

$$\begin{bmatrix} R_{1,1} \\ R_{2,1} & R_{2,2} \\ R_{3,1} & R_{3,2} & R_{3,3} \end{bmatrix}.$$

Sau khi thực hiện một vòng tính toán tương tự thì ta có

$$\begin{bmatrix} R_{1,1} \\ R_{2,1} & R_{2,2} \\ R_{3,1} & R_{3,2} & R_{3,3} \\ R_{4,1} & R_{4,2} & R_{4,3} & R_{4,4} \end{bmatrix}$$

với sai số $R_{4,4}$ là $\mathcal{O}(h^8)$. Chú ý rằng xấp xỉ chính xác nhất của tích phân là phần tử cuối cùng trên đường chéo của mảng. Quá trình tiếp tục cho đến khi sự khác biệt giữa hai phần tử cuối cùng trên đường chéo đủ nhỏ. Công thức ngoại suy tổng quát được sử dụng là

$$R_{i,j} = \frac{4^{j-1}R_{i,j-1} - R_{i-1,j-1}}{4^{j-1} - 1}, i > 1, j = 2, 3, \dots, i. \quad (5.22a)$$

Một biểu diễn trực quan của (5.22a) là

$$\begin{array}{c} \boxed{R_{i-1,j-1}} \\ \searrow \\ \alpha \\ \searrow \\ \boxed{R_{i,j-1}} \rightarrow \beta \rightarrow \boxed{R_{i,j}} \end{array} \quad (5.22b)$$

với α, β là các hằng số phụ thuộc vào j được tính toán như sau

j	2	3	4	5	6	
α	-1/3	-1/15	-1/63	-1/255	-1/1023	
β	4/3	16/15	64/63	256/255	1024/1023	

(5.22c)

Mảng trên thuận lợi cho việc thực hiện tính toán bằng tay nhưng máy tính có thể thực hiện thuật toán Romberg trong một mảng một chiều r . Sau khi ngoại suy lần đầu, xem (a), $R_{1,1}$ sẽ không cần sử dụng tiếp, nên nó có thể được thay thế bởi $R_{2,2}$. Kết quả là ta có mảng

$$\begin{bmatrix} r_1 = R_{2,2} \\ r_2 = R_{2,1} \end{bmatrix}.$$

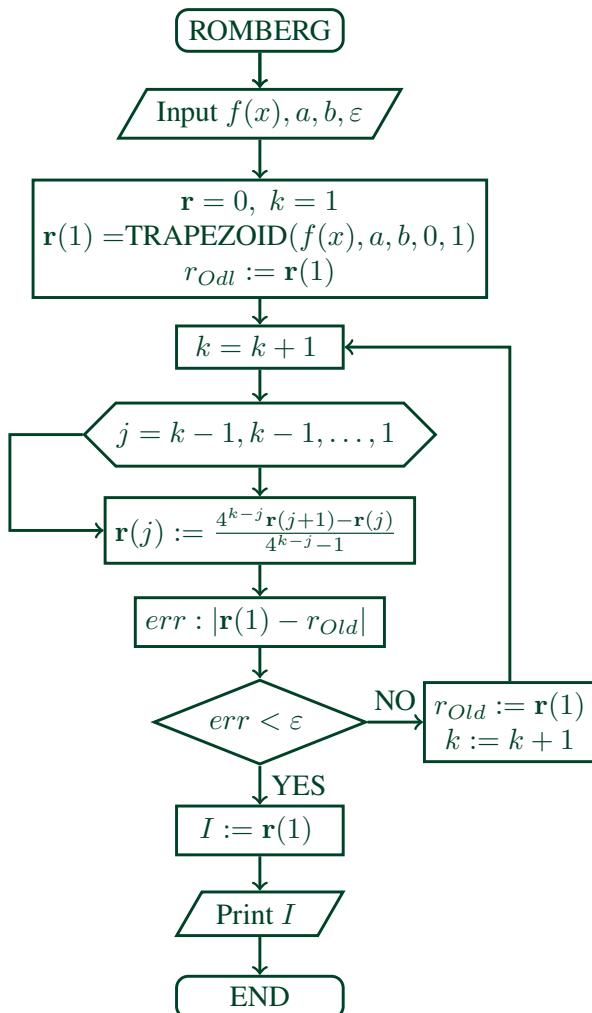
Ở vòng ngoại suy thứ hai, tính toán bởi (b) và (c), $R_{3,2}$ được ghi đè lên $R_{2,1}$ và $R_{3,3}$ thay thế $R_{2,2}$, do đó ta có

$$\begin{bmatrix} r_1 = R_{3,3} \\ r_2 = R_{3,2} \\ r_3 = R_{3,1} \end{bmatrix}.$$

Thực hiện tính toán các vòng khác tương tự. Ta thấy rằng phần tử r_1 luôn chứa kết quả hiện tại tốt nhất. Công thức ngoại suy cho vòng thứ k là

$$r_j = \frac{4^{k-j}r_{j+1} - r_j}{4^{k-j} - 1}, j = k-1, k-2, \dots, 1. \quad (5.23)$$

Quá trình tính tích phân xác định bằng phương pháp tích phân Romberg được minh họa bằng sơ đồ khối cho trong Hình 5.8



**Hình 5.8: Sơ đồ khối
tính tích phân xác
định bằng phương
pháp tích phân
Romberg**

romberg.

Thuật toán cho tích phân Romberg được thực hiện bằng hàm romberg. Nó trả kết

quả của tích phân và yêu cầu số của ước lượng hàm số. Ngoại suy Richardson được thực hiện bởi hàm con richardson.

```

function [I,numEval] = romberg(func,a,b,tol,kMax)
    % Romberg integration.

    % USAGE: [ I, numEval ] = romberg( func, a, b, tol, kMax )
    % INPUT:
    % func = handle of function being integrated.
    % a,b = limits of integration.
    % tol = error tolerance (default is 1.0e-8).
    % kMax = limit on the number of panel doublings
    % (default is 20).

    % OUTPUT:
    % I = value of the integral.
    % numEval = number of function evaluations.

    if nargin < 5; kMax = 20; end
    if nargin < 4; tol = 1.0e-8; end
    r = zeros(kMax);
    r(1) = trapezoid(func,a,b,0,1);
    rOld = r(1);
    for k=2:kMax
        r(k) = trapezoid(func,a,b,r(k-1),k);
        r = richardson(r,k);
        if abs(r(1) - rOld) < tol
            numEval=2^(k-1)+1; I=r(1);
            return
        end
        rOld = r(1);
    end
    error('Failed to converge')

function r=richardson(r,k)
    % Richardson's extrapolation in Eq. (5.23).
    for j=k-1:-1:1
        c = 4^(k-j); r(j) = (c*r(j+1) - r(j))/(c-1);
    end

```

VÍ DỤ 5.3.5. Hãy chỉ ra $R_{k,2}$ trong tích phân Romberg chính là công thức Simpson $1/3$ trong (5.19) với 2^{k-1} đoạn.

GIẢI. Nhắc lại rằng trong tích phân Romberg thì $R_{k,1} = I_k$ là ký hiệu xấp xỉ tích phân thu được bằng phương pháp hình thang kết hợp với 2^{k-1} đoạn. Ký hiệu hoành độ của các nút là x_1, x_2, \dots, x_n , ta có công thức hình thang kết hợp trong (5.14) là

$$R_{k,1} = I_k = \left[f(x_1) + 2 \sum_{i=2}^{n-1} f(x_i) + f(x_n) \right] \frac{h}{2}.$$

Khi chúng ta giảm một nửa số đoạn (mỗi đoạn có chiều rộng $2h$) thì chỉ có hoành độ của những nút lẻ mới tham gia vào công thức hình thang kết hợp. Khi đó

$$R_{k-1,1} = I_{k-1} = \left[f(x_1) + 2 \sum_{i=3,5,\dots}^{n-2} f(x_i) + f(x_n) \right] h.$$

Áp dụng phép ngoại suy Richardson ta thu được công thức

$$R_{k,2} = \frac{4}{3} R_{k,1} - \frac{1}{3} R_{k-1,1} = \left[\frac{1}{3} f(x_1) + \frac{4}{3} \sum_{i=2,4,\dots}^{n-1} f(x_i) + \frac{2}{3} \sum_{i=3,5,\dots}^{n-2} f(x_i) + \frac{1}{3} f(x_n) \right] h.$$

chính là công thức Simpson trong (5.19).

□

VÍ DỤ 5.3.6. Sử dụng tích phân Romberg ước lượng $\int_0^\pi f(x)dx$, với $f(x) = \sin x$. Thực hiện với bốn chữ số thập phân.

GIẢI. Từ công thức hình thang đệ quy trong (5.18b) ta có

$$R_{1,1} = I(\pi) = \frac{\pi}{2} [f(0) + f(\pi)] = 0$$

$$R_{2,1} = I(\pi/2) = \frac{1}{2} I(\pi) + \frac{\pi}{2} f(\pi/2) = 1.5708$$

$$R_{3,1} = I(\pi/4) = \frac{1}{2} I(\pi/2) + \frac{\pi}{4} [f(\pi/4) + f(3\pi/4)] = 1.8961$$

$$R_{4,1} = I(\pi/8) = \frac{1}{2} I(\pi/4) + \frac{\pi}{8} [f(\pi/8) + f(3\pi/8) + f(5\pi/8) + f(7\pi/8)] = 1.9724.$$

Sử dụng (5.22) ta có

$$\begin{bmatrix} R_{1,1} \\ R_{2,1} & R_{2,2} \\ R_{3,1} & R_{3,2} & R_{3,3} \\ R_{4,1} & R_{4,2} & R_{4,3} & R_{4,4} \end{bmatrix} = \begin{bmatrix} 0 \\ 1.5708 & 2.0944 \\ 1.8961 & 2.0046 & 0.9986 \\ 1.9742 & 2.0003 & 2.0000 & 2.0000 \end{bmatrix}.$$

Có vẻ như đây là quá trình hội tụ nên $\int_0^\pi \sin x dx = R_{4,4} = 2$, đây cũng là kết quả chính xác của tích phân này.

□

VÍ DỤ 5.3.7. Sử dụng tích phân Romberg, ước lượng $\int_0^{\sqrt{\pi}} 2x^2 \cos x^2 dx$ và so sánh với kết quả trong Ví dụ 5.3.2.

GIẢI. Sử dụng hàm romberg để tính số ta có

```
>> format long
>> [Integral,numEval] = romberg(@fex5_3_7,0,sqrt(pi))

Integral =
-0.894831469484157

numEval =
129
```

với M-file xác định hàm dưới dấu tích phân là

```
function y=fex5_3_7(x)
% Function used in Example 5.3.7
y = 2*(x^2)*cos(x^2);
```

Rõ ràng tích phân Romberg hiệu quả hơn đáng kể so với công thức hình thang. Nó yêu cầu 129 ước lượng hàm số so với 4097 ước lượng của công thức hình thang trong Ví dụ 5.3.2.

□

5.3.4 CÔNG THỨC CẦU PHƯƠNG GAUSSIAN

A. CÔNG THỨC CẦU PHƯƠNG GAUSSIAN

Chúng ta thấy rằng công thức Newton-Cotes xấp xỉ $\int_a^b f(x)dx$ tốt nhất nếu $f(x)$ là một hàm trơn, chẳng hạn như một đa thức. Điều này cũng đúng với phương pháp cầu phương Gaussian. Tuy nhiên, công thức cầu phương Gaussian tính toán tốt hơn khi xấp xỉ cho tích phân dạng

$$\int_a^b w(x)f(x)dx \quad (5.24)$$

với $w(x)$, được gọi là *hàm trọng*, có thể chứa một số điểm kỳ dị, miễn là nó khả tích. Chẳng hạn như tích phân $\int_0^1 (1+x^2) \ln x dx$. Một số trường hợp tích phân có cận vô

cùng, chẳng hạn $\int_0^\infty e^{-x} \sin x dx$, thì công thức cầu phương Gaussian cũng có thể tính toán được.

Công thức cầu phương Gaussian cũng có dạng tương tự như công thức tích phân Newton-Cotes:

$$I = \sum_{i=1}^n A_i f(x_i), \quad (5.25)$$

với I biểu diễn xấp xỉ của tích phân trong (5.24). Sự khác biệt nằm ở chỗ cách xác định các trọng số A_i và các hoành độ nút x_i . Trong tích phân Newton-Cotes, các nút được cách đều và nằm trong (a, b) , tức là vị trí của chúng đã được xác định trước. Trong cầu phương Gaussian, các nút và các trọng số được chọn sao cho công thức (5.25) là chính xác nếu $f(x)$ là đa thức bậc $2n - 1$ hoặc nhỏ hơn, tức là

$$\int_a^b w(x) P_m(x) dx = \sum_{i=1}^n A_i P_m(x_i), \quad m \leq 2n - 1. \quad (5.26)$$

Một cách để xác định các trọng số và hoành độ các nút là thay $P_1(x) = 1, P_2(x) = x, \dots, P_{2n-1}(x) = x^{2n-1}$ vào (5.26) và giải hệ $2n$ phương trình

$$\int_a^b w(x) x^j dx = \sum_{i=1}^n A_i x_i^j, \quad j = 0, 1, 2, \dots, 2n - 1$$

để tìm nghiệm A_i và $x_i, i = 1, 2, \dots, n$.

Xét ví dụ minh họa với $w(x) = e^{-x}, a = 0, b = \infty$ và $n = 2$. Bốn phương trình xác định x_1, x_2, A_1 và A_2 là

$$\begin{aligned} \int_0^\infty e^{-x} dx &= A_1 + A_2 \\ \int_0^\infty e^{-x} x dx &= A_1 x_1 + A_2 x_2 \\ \int_0^\infty e^{-x} x^2 dx &= A_1 x_1^2 + A_2 x_2^2 \\ \int_0^\infty e^{-x} x^3 dx &= A_1 x_1^3 + A_2 x_2^3. \end{aligned}$$

Sau khi tính các tích phân, ta có

$$\begin{aligned} x_1 &= 2 - \sqrt{2}, & A_1 &= \frac{\sqrt{2} + 1}{2\sqrt{2}} \\ x_2 &= 2 + \sqrt{2}, & A_2 &= \frac{\sqrt{2} - 1}{2\sqrt{2}} \end{aligned}$$

do đó công thức cầu phương là

$$\int_0^\infty e^{-x} f(x) dx \approx \frac{1}{2\sqrt{2}} [(\sqrt{2} + 1) f(2 - \sqrt{2}) + (\sqrt{2} - 1) f(2 + \sqrt{2})].$$

Do tính phi tuyến của các phương trình nên việc xấp xỉ này sẽ không tốt khi n lớn. Các phương pháp thực tế để tìm x_i và A_i yêu cầu một số kiến thức về các đa thức trực giao và mối liên hệ của chúng với cầu phương Gaussian. Tuy nhiên, có một số công thức tích phân Gaussian cổ điển với các nút (abscissas) và các trọng số được tính với độ chính xác cao và được lập bảng. Những công thức này có thể sử dụng được mà không cần biết lý thuyết về chúng vì chỉ có một yêu cầu của tích phân Gaussian là các giá trị của x_i và A_i . Nếu bạn không có ý định tìm hiểu kỹ về việc tìm x_i và A_i thì bạn có thể bỏ qua hai chủ đề tiếp theo.

B*. CÁC ĐA THỨC TRỰC GIAO

Các đa thức trực giao được sử dụng trong một số lĩnh vực của toán học và giải tích số. Chúng được nghiên cứu rất nhiều và một số tính chất của chúng rất nổi tiếng. Những điều sẽ trình bày sau đây chỉ là một tóm tắt nhỏ của một chủ đề lớn.

Các đa thức $\varphi_n(x)$, $n = 1, 2, \dots$ (n là bậc của đa thức) được gọi là dạng một *tập trực giao* trong khoảng (a, b) với hàm trọng $w(x)$ nếu

$$\int_a^b w(x)\varphi_m(x)\varphi_n(x)dx = 0, m \neq n. \quad (5.27)$$

Tập này được xác định bằng cách chọn hàm trọng và các cận lấy tích phân. Do đó, mỗi tập các đa thức trực giao được liên hệ với $w(x)$, a và b nhất định. Một số các đa thức trực giao cổ điển, lấy tên của các nhà toán học nổi tiếng, được liệt kê trong Bảng 5.7. Cột cuối cùng trong bảng là chuẩn hóa được sử dụng.

Tên	Ký hiệu	a	b	$w(x)$	$\int_a^b w(x)[\varphi_n(x)]^2 dx$
Legendre	$p_n(x)$	-1	1	1	$2/(2n+1)$
Chebyshev	$T_n(x)$	-1	1	$(1-x^2)^{-1/2}$	$\pi/2 \quad (n > 0)$
Laguerre	$L_n(x)$	0	∞	e^{-x}	1
Hermite	$H_n(x)$	$-\infty$	∞	e^{-x^2}	$\sqrt{\pi}2^n n!$

Bảng 5.7: Bảng liệt kê một số tập các đa thức trực giao

Các đa thức trực giao tuân theo mối liên hệ đệ quy dạng

$$a_n \varphi_{n+1}(x) = (b_n + c_n x) \varphi_n(x) - d_n \varphi_{n-1}(x). \quad (5.28)$$

Nếu hai đa thức đầu tiên của tập được xác định thì các đa thức còn lại của tập sẽ được tính toán bằng công thức (5.28). Hệ số của công thức đê quy cùng với $\varphi_0(x)$ và $\varphi_1(x)$ của một số tập đa thức trực giao được cho trong Bảng 5.8.

Tên	$\varphi_0(x)$	$\varphi_1(x)$	a_n	b_n	c_n	d_n
Legendre	1	x	$n + 1$	0	$2n + 1$	n
Chebyshev	1	x	1	0	2	1
Laguerre	1	$1 - x$	$n + 1$	$2n + 1$	-1	n
Hermite	1	$2x$	1	0	2	2

Bảng 5.8: Bảng xác định một số tập các đa thức trực giao dưới dạng đê quy

Các công thức trực giao cổ điển cũng thu được từ các công thức

$$\begin{aligned} p_n(x) &= \frac{(-1)^n}{2^n n!} \frac{d^n}{dx^n} [(1-x^2)^n] \\ T_n(x) &= \cos(n \cos^{-1} x), n > 0 \\ L_n(x) &= \frac{e^x}{n!} \frac{d^n}{dx^n} (x^n e^{-x}) \\ H_n(x) &= (-1)^n e^{x^2} \frac{d^n}{dx^n} (e^{-x^2}) \end{aligned} \tag{5.29}$$

và các đạo hàm có thể được xác định từ công thức

$$\begin{aligned} (1-x^2)p'_n(x) &= n[-xp_n(x) + p_{n-1}(x)] \\ (1-x^2)T'_n(x) &= n[-xT_n(x) + nT_{n-1}(x)] \\ xL'_n(x) &= n[L_n(x) - L_{n-1}(x)] \\ H'_n(x) &= 2nH_{n-1}(x). \end{aligned} \tag{5.30}$$

Một số tính chất khác của các đa thức nội suy liên quan đến sự chính xác của công thức cầu phương Gaussian là:

- $\varphi_n(x)$ có n nghiệm thực rời rạc trong khoảng (a,b) .
- Nghiệm của $\varphi_n(x)$ nằm giữa 0 và nghiệm của $\varphi_{n+1}(x)$.
- Mọi đa thức $P_n(x)$ bất kỳ có bậc n có thể biểu diễn dưới dạng

$$P_n(x) = \sum_{i=0}^n c_i \varphi_i(x). \tag{5.31}$$

□ Từ (5.31) và tính chất trực giao trong (5.27) ta có

$$\int_a^b w(x) P_n(x) \varphi_{m+n}(x) dx = 0, \quad m \geq 0. \quad (5.32)$$

C*. XÁC ĐỊNH CÁC MỐC VÀ TRỌNG SỐ

ĐỊNH LÝ 5.3.1. *Hoành độ các nút x_1, x_2, \dots, x_n là nghiệm của đa thức $\varphi_n(x)$ mà thuộc tập trực giao được xác định trong (5.27)*

Chứng minh. Chúng ta bắt đầu bằng việc đặt $f(x) = P_{2n-1}(x)$ là đa thức bậc $2n - 1$. Vì tích phân Gaussian với n nút là công thức chính xác với đa thức này, ta có

$$\int_a^b w(x) P_{2n-1}(x) dx = \sum_{i=1}^n A_i P_{2n-1}(x_i). \quad (a)$$

Một đa thức bậc $2n - 1$ luôn có thể viết dưới dạng

$$P_{2n-1}(x) = Q_{n-1}(x) + R_{n-1}(x) \varphi_n(x) \quad (b)$$

với $Q_{n-1}(x), R_{n-1}(x)$ và $\varphi_n(x)$ là các đa thức có bậc tương ứng với chỉ số lớn. Do đó

$$\int_a^b w(x) P_{2n-1}(x) dx = \int_a^b w(x) Q_{n-1}(x) dx + \int_a^b w(x) R_{n-1}(x) \varphi_n(x) dx.$$

Nhưng theo (5.32), tích phân thứ hai bên vế phải bị triệt tiêu nên

$$\int_a^b w(x) P_{2n-1}(x) dx = \int_a^b w(x) Q_{n-1}(x) dx. \quad (c)$$

Bởi vì một đa thức bậc $n - 1$ luôn có thể xác định duy nhất n điểm nên ta có thể tìm A_i sao cho

$$\int_a^b w(x) Q_{n-1}(x) dx = \sum_{i=1}^n A_i Q_{n-1}(x_i). \quad (d)$$

Để đạt được phương trình (a), ta phải chọn sao cho hoành độ x_i của nghiệm của $\varphi_n(x) = 0$. Theo (b) ta có

$$P_{2n-1}(x_i) = Q_{n-1}(x_i), \quad i = 1, 2, \dots \quad (e)$$

cùng với (c) và (d) dẫn đến

$$\int_a^b w(x) P_{2n-1}(x) dx = \int_a^b w(x) Q_{n-1}(x) dx = \sum_{i=1}^n A_i P_{2n-1}(x_i).$$

Đây là điều phải chứng minh. □

ĐỊNH LÝ 5.3.2.

$$A_i = \int_a^b w(x) P_n(x) l_i(x) dx, \quad i = 1, 2, \dots, \quad (5.33)$$

với $l_i(x)$ là các hàm cơ bản Lagrange được xác định dựa vào các nút tại x_1, x_2, \dots, x_n . Các hàm này được cho trong (4.2).

Chứng minh. Áp dụng công thức nội suy (4.1) cho $Q_{n-1}(x)$ thu được

$$Q_{n-1}(x) = \sum_{i=1}^n Q_{n-1}(x_i) l_i(x).$$

Thay $Q_{n-1}(x)$ vào (d) ta có

$$\sum_{i=1}^n \left[Q_{n-1}(x_i) \int_a^b w(x) P_n(x) l_i(x) dx \right] = \sum_{i=1}^n A_i Q_{n-1}(x_i)$$

hoặc

$$\sum_{i=1}^n Q_{n-1}(x_i) \left[A_i - \int_a^b w(x) P_n(x) l_i(x) dx \right] = 0.$$

Phương trình này thoả mãn với Q_{n-1} tuỳ ý nếu

$$A_i - \int_a^b w(x) P_n(x) l_i(x) dx = 0, \quad i = 1, 2, \dots, n.$$

Đây chính là điều cần chứng minh. □

Không khó khăn để tìm nghiệm $x_i, i = 1, 2, \dots, n$ của đa thức $\varphi_n(x)$ thuộc tập các đa thức trực giao bằng một trong các phương pháp đã thảo luận trong Chương 2. Khi các nghiệm này đã biết, ta có thể tính $A_i, i = 1, 2, \dots, n$ từ công thức (5.33), một số công thức trọng số được cho sau đây (bỏ qua chứng minh)

$$\begin{aligned} \text{Gauss-Legendre} \quad A_i &= \frac{2}{(1-x_i^2)[p'_n(x_i)]^2} \\ \text{Gauss-Laguerre} \quad A_i &= \frac{1}{x_i [L'_n(x_i)]^2} \\ \text{Gauss-Hermite} \quad A_i &= \frac{2^n n! \sqrt{\pi}}{[H'_n(x_i)]^2} \end{aligned} \quad (5.34)$$

D. CÁC MỐC VÀ CÁC TRỌNG SỐ CHO CẦU PHƯƠNG GAUSSIAN

Chúng ta liệt kê ở mục này một số công thức cầu phương Gaussian cổ điển. Các sau bảng thể hiện các mốc và trọng số được tính toán với $n = 2$ đến 6 và làm tròn đến

sáu chữ số thập phân. Các bảng này có thể phù hợp cho tính toán bằng tay nhưng nếu lập trình thì bạn có thể tính toán với nhiều mốc hơn.

Sai số phương pháp (truncation error) của phương pháp cầu phương Gaussian

$$E = \int_a^b w(x)f(x)dx - \sum_{i=1}^n A_i f(x_i)$$

có dạng $E = K(n)f^{(2n)}(c)$, với $a < c < b$ (giá trị của c chưa biết, ta chỉ biết nó bị chặn). Biểu thức $K(n)$ phụ thuộc vào từng phương pháp cầu phương được sử dụng. Nếu đạo hàm các cấp của $f(x)$ có thể được tính toán thì công thức này rất hữu ích trong việc ước lượng cận sai số.

Phương pháp cầu phương Gauss-Legendre.

$$\int_{-1}^1 f(\xi)d\xi \approx \sum_{i=1}^n A_i f(\xi_i). \quad (5.35)$$

$\pm \xi_i$	A_i	$\pm \xi_i$	A_i
$n = 2$		$n = 5$	
0.577350	1.000000	0.000000	0.568889
$n = 3$		$n = 6$	
0.000000	0.888889	0.906180	0.236927
0.774597	0.555556	$n = 4$	
$n = 4$		0.238619	0.467914
0.339981	0.652145	0.661209	0.306762
0.816136	0.347855	0.932470	0.171324

Bảng 5.9: Bảng tính phương pháp cầu phương Gauss-Legendre trong (5.35)

Đây là công thức cầu phương Gaussian được sử dụng nhiều. Các mốc có tính đối xứng qua $\xi = 0$ và các trọng số ứng với các mốc đối xứng bằng nhau. Chẳng hạn với $n = 2$ thì ta có $\xi_1 = -\xi_2$ và $A_1 = A_2$. Sai số của công thức (5.35) là

$$E = \frac{2^{2n+1}(n!)^4}{(n2+1)[(2n)!]^3} f^{(2n)}(c), \quad -1 < c < 1. \quad (5.36)$$

Để áp dụng công thức cầu phương Gauss-Legendre cho tích phân $\int_a^b f(x)dx$ thì trước tiên ta phải đưa miền lối tích phân (a, b) về khoảng $(-1, 1)$. Ta có thể thực hiện

điều này bằng phép đổi biến sau

$$x = \frac{b+a}{2} + \frac{b-a}{2}\xi. \quad (5.37)$$

Do $dx = \frac{b-a}{2}d\xi$ nên công thức cầu phương trở thành

$$\int_a^b f(x)dx \approx \frac{b-a}{2} \sum_{i=1}^n A_i f(x_i) \quad (5.38)$$

với x_i được tính từ (5.37). Sai số của phương pháp là

$$E = \frac{(b-a)^{2n-1}(n!)^4}{(2n+1)[(2n)!]^3} f^{(2n)}(c), \quad a < c < b. \quad (5.39)$$

Công thức cầu phương Gauss-Chebysev.

$$\int_{-1}^1 (1-x^2)^{-1/2} f(x)dx \approx \frac{\pi}{n} \sum_{i=1}^n f(x_i). \quad (5.40)$$

Chú ý rằng các trọng số đều là hằng số $A_i = \pi/n$ và các mốc có tính đối xứng qua $x = 0$ và được cho bởi

$$x_i = \cos \frac{(2i-1)\pi}{2n}. \quad (5.41)$$

Sai số là

$$E = \frac{2\pi}{2^{2n}(2n)!} f^{(2n)}(c), \quad -1 < c < 1. \quad (5.42)$$

Công thức cầu phương Gauss-Laguerre.

$$\int_0^\infty e^{-x} f(x)dx \approx \sum_{i=1}^n A_i f(x_i). \quad (5.43)$$

Các mốc và các trọng số của công thức cầu phương Gauss-Laguerre với n nhận các giá trị từ 2 đến 6 được cho trong Bảng 5.10. Trong bảng này, một số giá trị của A_i được nhân với 10^k , với k là giá trị nằm trong ngoặc đơn.

x_i	A_i	x_i	A_i		
$n = 2$			$n = 5$		
0.585786	0.853554	0.263560	0.521756		
3.414214	0.146447	1.413403	0.398667		
$n = 3$			$(-1)0.759424$		
0.415775	0.711093	7.085810	$(-2)0.361175$		
2.294280	0.278517	12.640801	$(-4)0.233670$		
6.289945	$(-1)0.103892$	$n = 6$			
$n = 4$			0.222847		
0.322548	0.603154	1.188932	0.417000		
1.745761	0.357418	2.992736	0.113373		
4.536620	$(-1)0.388791$	5.775144	$(-1)0.103992$		
9.395071	$(-3)0.539295$	9.837467	$(-3)0.261017$		
		15.982874	$(-6)0.898548$		

Bảng 5.10: Bảng tính phương pháp cầu phương Gauss-Laguerre trong (5.43)

Sai số của (5.43) được cho bởi công thức

$$E = \frac{(n!)^2}{(2n)!} f^{(2n)}(c), \quad 0 < c < \infty. \quad (5.44)$$

Phương pháp cầu phương Gauss-Hermit.

$$\int_{-\infty}^{\infty} e^{-x^2} f(x) dx \approx \sum_{i=1}^n A_i f(x_i). \quad (5.45)$$

Các mốc của công thức cầu phương Gauss-Hermit có tính đối xứng qua $x = 0$ và các trọng số ứng với các mốc đối xứng bằng nhau. Bảng 5.11 liệt kê các mốc và các trọng số của phương pháp với n nhận các giá trị từ 2 đến 6. Trong bảng này, một số giá trị của A_i được nhân với 10^k , với k là giá trị nằm trong ngoặc đơn.

Sai số của (5.45) được cho bởi công thức

$$E = \frac{\sqrt{\pi} n!}{2^2 (2n)!} f^{(2n)}(c), \quad 0 < c < \infty. \quad (5.46)$$

Công thức cầu phương Gauss với tính kỳ dị logarit.

x_i	A_i	x_i	A_i
	$n = 2$		$n = 5$
0.707107	0.886227	0.000000	0.945308
	$n = 3$	0.958572	0.393619
0.000000	1.181636	2.020183	0.393169
1.224745	0.295409		$n = 6$
	$n = 4$	0.436077	0.724629
0.524648	0.804914	1.335849	0.157067
1.650680	(-1)0.813128	2.350605	(-2)0.453001

Bảng 5.11: Bảng tính phương pháp cầu phương Gauss-Hermit trong (5.45)

$$\int_0^\infty f(x) \ln x dx \approx - \sum_{i=1}^n A_i f(x_i). \quad (5.47)$$

Các mốc và các trọng số của công thức cầu phương Gauss-Laguerre với n nhận các giá trị từ 2 đến 6 được cho trong Bảng 5.12. Trong bảng này, một số giá trị của x_i, A_i được nhân với 10^k , với k là giá trị nằm trong ngoặc đơn.

Sai số của (5.47) được cho bởi công thức

$$E = \frac{k(n)}{(2n)!} f^{(2n)}(c), \quad 0 < c < 1. \quad (5.48)$$

với $k(2) = 0.00285, k(3) = 0.00017, k(4) = 0.00001$.

 gaussNodes.

Hàm gaussNodes tính các mốc x_i và các trọng số A_i tương ứng bằng cách sử dụng phương pháp cầu phương Gauss-Legendre. Nó cũng chỉ ra rằng giá trị xấp xỉ của các mốc là

$$x_i = \cos \frac{\pi(i - 0.25)}{n + 0.5}.$$

Sử dụng các xấp xỉ này làm các giá trị bắt đầu, chúng ta tính các mốc bằng cách tìm nghiệm không âm của đa thức Legendre bằng phương pháp Newton-Raphson (các nghiệm âm thu được bằng cách lấy đối xứng). Chú ý rằng hàm gaussNodes gọi lên hàm con legendre, đây là hàm trả về $p_n(x)$ và đạo hàm của nó.

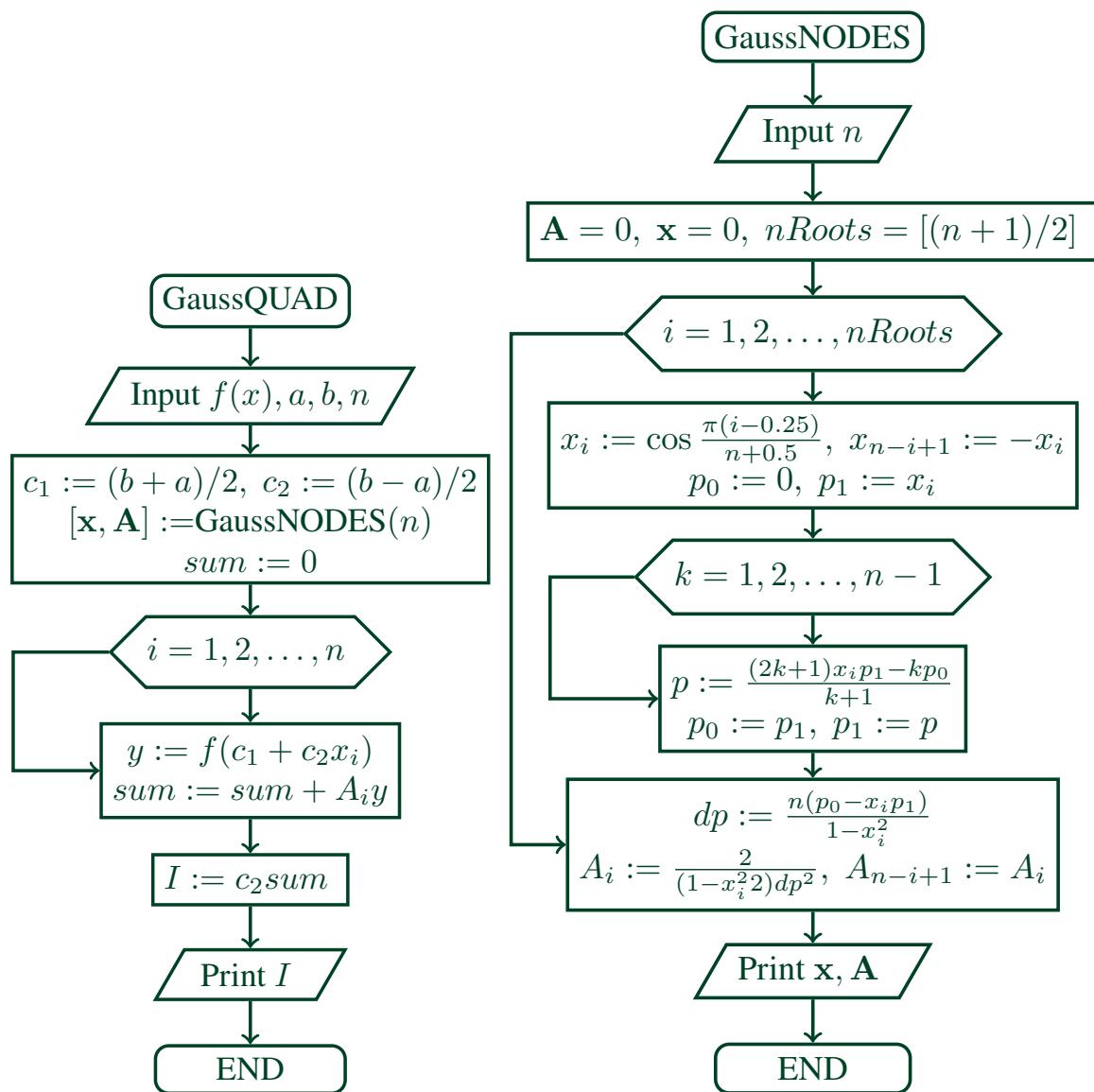
```
function [x, A] = gaussNodes(n, tol)
```

x_i	A_i	\mathbf{x}_i	A_i
$n = 2$		$n = 5$	
0.112009	0.718539	(-1)0.219345	0.297893
0.602277	0.281461	0.173977	0.349776
$n = 3$		$n = 6$	
(-1)0.638907	0.513405	0.677314	(-1)0.989305
0.368997	0.391980	0.894771	(-1)0.189116
0.766884	(-1)0.946154	$n = 4$	
$n = 4$		(-1)0.216344	0.238764
(-1)0.414485	0.383464	0.129583	0.308287
0.245275	0.386875	0.314020	0.245317
0.556165	0.190435	0.538657	0.142009
0.848982	(-1)0.392255	0.756916	(-1)0.554546
		0.922669	(-1)0.101690

Bảng 5.12: Bảng tính phương pháp cầu phương Gauss với tính kỳ dị logarit trong (5.47)

```
% Computes nodal abscissas x and weights A of
% Gauss-Legendre n-point quadrature.
% USAGE: [x,A] = gaussNodes(n,epsilon,maxIter)
% tol = error tolerance (default is 1.0e4*eps).

if nargin < 2; tol = 1.0e4*eps; end
A=zeros(n,1);x=zeros(n,1);
nRoots = fix(n + 1)/2; % Number of non-neg. roots
for i=1:nRoots
    t = cos(pi*(i - 0.25)/(n + 0.5)); % Approx. roots
    for j=i:30
        [p,dp] = legendre(t,n); % Newton's
        dt=-p/dp;t=t+dt; %rootfinding
        if abs(dt) < tol % method
            x(i) = t; x(n-i+1) = -t;
            A(i) = 2/(1-t^2)/dp^2; % Eq. (5.25)
            A(n-i+1) = A(i);
            break
        end
    end
end
```



(a) Tính tích phân bằng phương pháp Gaussian

(b) Tính các nodes bằng đa thức Legendre

Hình 5.9: Sơ đồ khôi tính tích phân xác định bằng phương pháp cầu phương Gaussian sử dụng đa thức trực giao Legendre

```

    end
    end
end

```

Hàm con legendre được lập trình trong M-file sau đây.

```

function [p, dp] = legendre(t, n)
    % Evaluates Legendre polynomial p of degree n

```

```
% and its derivative p at x=t.

p0=1.0;p1=t;
for k=1:n-1
    p = ((2*k + 1)*t*p1 - k*p0) / (k + 1); % Eq. (5.19)
    p0=p1;p1=p;
end
dp=n*(p0-t*p1) / (1-t^2); %Eq. (5.21)
```

gaussQuad.

Hàm gaussQuad ước lượng $\int_a^b f(x)dx$ bằng phương pháp cầu phương Legendre với n mốc. Hàm $f(x)$ được cung cấp từ người sử dụng. Các mốc và trọng số được tính bằng hàm gaussNodes.

```
function I=gaussQuad(func,a,b,n)
    % Gauss-Legendre quadrature.
    % USAGE:I=gaussQuad(func,a,b,n)
    % INPUT:
    % func=handle of function to be integrated.
    % a,b = integration limits.
    % n = order of integration.
    % OUTPUT:
    % I=integral

    c1=(b+a)/2;c2=(b-a)/2; % Mapping constants
    [x,A] = gaussNodes(n); % Nodal abscissas & weights
    sum=0;
    for i=1:length(x)
        y = feval(func,c1 + c2*x(i)); % Function at node i
        sum = sum + A(i)*y;
    end
    I = c2*sum;
```

VÍ DỤ 5.3.8. Ước lượng $\int_{-1}^1 (1-x^2)^{3/2}dx$ chính xác nhất có thể bằng tích phân Gaussian.

GIẢI. Vì hàm dưới dấu tích phân trơn và không có điểm kỳ dị nên ta có thể sử dụng phương pháp cầu phương Gauss-Legendre. Tuy nhiên, tích phân chính xác có thể thu

được bằng công thức Gauss-Chebysev. Ta viết lại

$$\int_{-1}^1 (1-x^2)^{3/2} dx = \int_{-1}^1 \frac{(1-x^2)^2 \sqrt{1-x^2}}{d} x.$$

Tử số $f(x) = (1-x^2)^2$ là đa thức bậc 4 nên phương pháp cầu phương Gaussian-Chebysev có thể tính chính xác với 3 nút.

Hoành độ của các nút này thu được từ (5.41). Thay $n = 3$ ta có

$$x_i = \cos \frac{(2i-1)\pi}{2(3)}, \quad i = 1, 2, 3.$$

Do đó

$$\begin{aligned} x_1 &= \cos \frac{\pi}{6} = \frac{\sqrt{3}}{2}, \\ x_2 &= \cos \frac{\pi}{2} = 0, \\ x_3 &= \cos \frac{5\pi}{6} = -\frac{\sqrt{3}}{2} \end{aligned}$$

và từ (5.40) thu được

$$\begin{aligned} \int_{-1}^1 (1-x^2)^{3/2} dx &= \frac{\pi}{3} \sum_{i=1}^3 (1-x_i^2)^2 \\ &= \frac{\pi}{3} \left[\left(1-\frac{3}{4}\right)^2 + (1-0)^2 + \left(1-\frac{3}{4}\right)^2 \right] \frac{3\pi}{8}. \end{aligned}$$

□

VÍ DỤ 5.3.9. Sử dụng tích phân Gaussian ước lượng $\int_0^{0.5} \cos \pi x \ln x dx$.

GIẢI. Chúng ta tách tích phân thành hai phần

$$\int_0^{0.5} \cos \pi x \ln x dx = \int_0^1 \cos \pi x \ln x dx - \int_{0.5}^1 \cos \pi x \ln x dx$$

Tích phân thứ nhất bên về phải có một điểm kỳ dị tại $x = 0$ nên có thể tính toán bằng phương pháp cầu phương Gaussian đặc biệt trong (5.47). Chọn $n = 4$ ta có

$$\int_0^1 \cos \pi x \ln x dx \approx - \sum_{i=1}^4 A_i \cos \pi x_i$$

với x_i, A_i được cho trong Bảng 5.12. Tổng được ước lượng trong bảng sau

x_i	$\cos \pi x_i$	A_i	$A_i \cos \pi x_i$
0.041448	0.991534	0.383464	0.380218
0.245275	0.717525	0.386875	0.227592
0.556165	-0.175533	0.190435	-0.033428
0.848982	-0.889550	0.039225	-0.034892
			$\sum = 0.589490$

Do đó

$$\int_0^1 \cos \pi x \ln x dx \approx -0.589490.$$

Tích phân thứ hai không có điểm kỳ dị nên có thể được ước lượng bằng phương pháp cầu phương Gauss-Legendre. Tiếp tục chọn $n = 4$ ta có

$$\int_{0.5}^1 \cos \pi x \ln x dx \approx 0.25 \sum_{i=1}^4 A_i \cos \pi x_i \ln x_i$$

với hoành độ các nút là (xem (5.37))

$$x_i = \frac{1+0.5}{2} + \frac{1-0.5}{2}\xi_i = 0.75 + 0.25\xi_i.$$

Từ các giá trị ξ_i và A_i ở Bảng 5.9, ta có tính toán sau

ξ_i	x_i	$\cos \pi x_i \ln x_i$	A_i	$A_i \cos \pi x_i \ln x_i$
-0.861136	0.534716	0.068141	0.347855	0.023703
-0.339981	0.665005	0.202133	0.652145	0.131820
0.339981	0.834995	0.156638	0.652145	0.102151
0.861136	0.965284	0.035123	0.347855	0.012218
				$\sum = 0.269892$

nên

$$\int_{0.5}^1 \cos \pi x \ln x dx \approx 0.25(0.269892) = 0.067473.$$

Do đó

$$\int_0^{0.5} \cos \pi x \ln x dx \approx -0.589490 - 0.067473 = -0.656963.$$

□

VÍ DỤ 5.3.10. Ước lượng chính xác nhất có thể tích phân

$$F = \int_0^\infty \frac{x+3}{\sqrt{x}} e^{-x} dx.$$

GIẢI. Dạng tích phân này không phù hợp với bất kỳ dạng cầu phương Gaussian nào mà ta đã liệt kê phía trên. Tuy nhiên, thực hiện đổi biến

$$x = t^2, \quad dx = 2tdt$$

ta có

$$F = 2 \int_0^{\infty} (t^2 + 3)e^{-t^2} dt = \int_{-\infty}^{\infty} (t^2 + 3)e^{-t^2} dt$$

nên tích phân này có thể tính chính xác bằng công thức Gauss-Hermit sử dụng 2 nút ($n = 2$). Do đó

$$\begin{aligned} F &= A_1(t_1^2 + 3) + A_2(t_2^2 + 3) \\ &= 0.886227 [(0.707107)^2 + 3] + 0.886227 [(-0.707107)^2 + 3] \\ &= 6.20359. \end{aligned}$$

□

VÍ DỤ 5.3.11. Xác định xem có bao nhiêu nút được yêu cầu để ước lượng $\int_0^{\pi} \left(\frac{\sin x}{x}\right)^2 dx$ bằng phương pháp cầu phương Gauss-Legendre chính xác tới sáu số thập phân. Giá trị chính xác làm tròn đến sáu số thập phân của tích phân này là 1.41815.

GIẢI. Hàm dưới dấu tích phân là hàm trơn nên nó phù hợp cho tích phân Gauss-Legendre. Hàm số không xác định tại $x = 0$ nhưng nó không gây khó khăn cho phương pháp cầu phương vì hàm số không bị triệt tiêu tại điểm này. Chúng ta sử dụng chương trình sau đây tính toán phương pháp cầu phương với số nút là 2, 3, ... bằng cách lặp hàm gaussQuad cho đến khi đạt độ chính xác mong muốn.

```
>> % Example 5.3.11 (Gauss-Legendre quadrature)
a=0;b=pi;Iexact=1.41815;
for n=2:12
    I = gaussQuad(@fex6_11,a,b,n);
    if abs(I - Iexact) < 0.00001
        I
        n
        break
    end
end
```

M-file tính hàm dưới dấu tích phân là

```
function y=fex5_3_11(x)
% Function used in Example 5.3.11
y = (sin(x)/x)^2;
```

Chương trình cho kết quả sau

I=

1.41815026780139

n=

5

□

VÍ DỤ 5.3.12. Ước lượng số $\int_{1.5}^3 f(x)dx$, với $f(x)$ được biểu diễn bởi dữ liệu không cách đều

x	1.2	1.7	2.0	2.4	2.9	3.3
$f(x)$	-0.36236	0.12884	0.41615	0.73739	0.97096	0.98748

Biết rằng các điểm dữ liệu nằm trên đường cong $f(x) = -\cos x$, ước lượng độ chính xác của nghiệm.

GIẢI. Chúng ta xấp xỉ $f(x)$ bởi $P_5(x)$ đi qua tất cả các điểm dữ liệu. Khi đó, ta có ước lượng $\int_{1.5}^3 f(x)dx \approx \int_{1.5}^3 P_5(x)dx$ bằng phương pháp Gauss-Legendre. Vì đa thức có bậc năm nên chỉ yêu cầu ba nút trong công thức cầu phương.

Từ (5.37) và Bảng 5.9 ta thu được hoành độ của các nút

$$\begin{aligned}x_1 &= \frac{3+1.5}{2} + \frac{3-1.5}{2}(-0.774597) = 1.6691 \\x_2 &= \frac{3+1.5}{2} = 2.25 \\x_3 &= \frac{3+1.5}{2} + \frac{3-1.5}{2}(0.774597) = 2.8309.\end{aligned}$$

Chúng ta tính các giá trị của đa thức nội suy $P_5(x)$ tại các nút. Điều này có thể thực hiện được bằng cách sử dụng hàm newtonPoly hoặc hàm neville. Kết quả là

$$P_5(x_1) = 0.09808, \quad P_5(x_2) = 0.62816, \quad P_5(x_3) = 0.95216.$$

Sử dụng cầu phương Gauss-Legendre

$$\int_{1.5}^3 P_5(x)dx = \frac{3-1.5}{2} \sum_{i=1}^3 A_i P_5(x_i)$$

ta có

$$\begin{aligned}I &= 0.75 [0.555556(0.09808) + 0.888889(0.62816) + 0.555556(0.95216)] \\&= 0.85637\end{aligned}$$

So sánh với $-\int_{1.5}^3 \cos x dx = 0.85638$ thì ta thấy rằng sự khác nhau chỉ là sai số làm tròn.

□

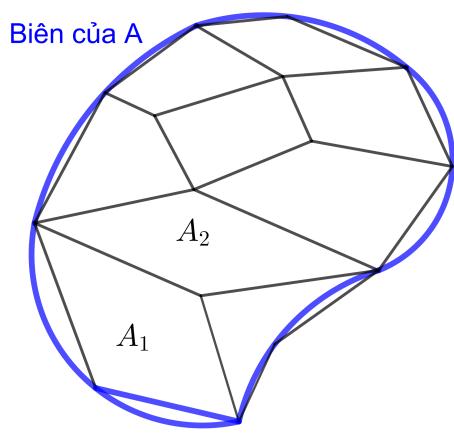
5.4 TÍCH PHÂN BỘI

5.4.1 MỞ ĐẦU

Tích phân bội, chẳng hạn như tích phân kép $\iint_A f(x, y) dx dy$, cũng có thể được tính toán bởi phương pháp cầu phương. Việc tính toán có thể thực hiện trực tiếp nếu miền lấp tích phân có dạng hình học phẳng đơn giản như hình tam giác, hình tứ giác. Do phức tạp trong việc xác định cận lấp tích phân cho x và y , phương pháp cầu phương không phải là phương pháp thực hành tốt khi miền lấp tích phân có hình dạng bất thường. Tuy nhiên, miền có hình dạng bất thường A luôn có thể xấp xỉ là một tập hợp các miền nhỏ có hình tam giác hoặc tứ giác A_1, A_2, \dots được gọi là *phần tử hữu hạn*. Minh họa như Hình 5.10. Tích phân trên miền A có thể được xấp xỉ bằng tổng các tích phân trên các phần tử hữu hạn

$$\iint_A f(x, y) dx dy \approx \sum_i \iint_{A_i} f(x, y) dx dy.$$

Các tích phân bội trên các khối trong không gian ba chiều được tính toán theo cách



Hình 5.10: Đồ thị minh họa mô hình phần tử hữu hạn của miền lấp tích phân kép tuỳ ý

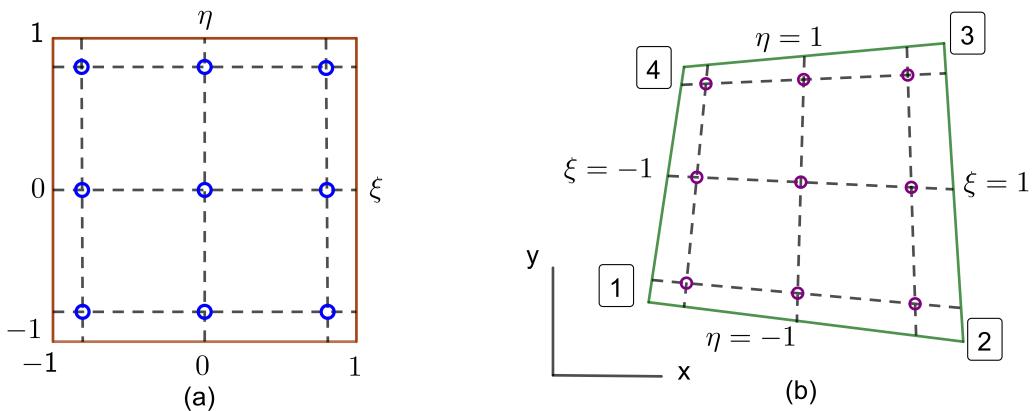
tương tự bằng cách sử dụng các khối tứ diện hoặc lăng trụ là các phần tử hữu hạn. Trong nội dung của cuốn sách này, chúng ta sẽ thực hiện tính toán tích phân kép trên miền tứ giác và miền tam giác bằng phương pháp cầu phương Gauss-Legendre.

5.4.2 PHƯƠNG PHÁP CẦU PHƯƠNG GAUSS-LEGENDRE TRÊN PHẦN TỬ TỨ GIÁC

Xét tích phân kép

$$I = \int_{-1}^1 \int_{-1}^1 f(\xi, \eta) d\xi d\eta$$

trên miền hình chữ nhật được cho trong Hình 5.11a. Thực hiện tính tích phân trên mỗi cạnh bằng phương pháp cầu phương Gauss-Legendre sử dụng n nút cho mỗi trục toạ độ ta có



Hình 5.11: Ánh xạ một hình tứ giác vào hình chữ nhật chuẩn

$$I = \int_{-1}^1 \sum_{i=1}^n A_i f(\xi_i, \eta) d\eta = \sum_{j=1}^n A_j \left[\sum_{i=1}^n A_i f(\xi_i, \eta_j) \right]$$

hay

$$I = \sum_{i=1}^n \sum_{j=1}^n A_i A_j f(\xi_i, \eta_j). \quad (5.49)$$

Số nút n trên mỗi trục toạ độ được gọi là *bậc tích phân*. Hình 5.11 mô tả bậc tích phân bằng ba. Bởi vì các cận của tích phân có dạng cận “chuẩn” $(-1, 1)$ của phương pháp cầu phương Gauss-Legendre nên toạ độ các điểm tích phân được liệt kê như trong Bảng 5.9.

Trong trường hợp áp dụng phương pháp cầu phương cho phần tử tứ giác trong Hình 5.11b, chúng ta phải ánh xạ tứ giác này sang dạng hình chữ nhật “chuẩn”. Ánh xạ chúng ta nói đến là phép biến đổi toạ độ $x = x(\xi, \eta), y = y(\xi, \eta)$ kết quả là mỗi

điểm của hình tứ giác tương ứng với một điểm trong hình chữ nhật. Phép biến đổi toạ độ được biểu diễn dạng

$$x(\xi, \eta) = \sum_{k=1}^4 N_k(\xi, \eta) x_k, \quad y(\xi, \eta) = \sum_1^4 N_k(\xi, \eta) y_k \quad (5.50)$$

với (x_k, y_k) là toạ độ góc thứ k của hình tứ giác và

$$\begin{aligned} N_1(\xi, \eta) &= \frac{1}{4}(1 - \xi)(1 - \eta) \\ N_2(\xi, \eta) &= \frac{1}{4}(1 + \xi)(1 - \eta) \\ N_3(\xi, \eta) &= \frac{1}{4}(1 + \xi)(1 + \eta) \\ N_4(\xi, \eta) &= \frac{1}{4}(1 - \xi)(1 + \eta) \end{aligned} \quad (5.51)$$

Hàm $N_k(\xi, \eta)$, được biết đến là *hàm hình dạng*, là song tuyến tính (tức là tuyến tính theo mỗi biến). Do đó, các đường thẳng vẫn được giữ qua ánh xạ. Đặc biệt, lưu ý rằng các cạnh của tứ giác được ánh xạ thành các đường thẳng $\xi = \pm 1$ và $\eta = \pm 1$.

Bởi vì ánh xạ làm biến dạng miền lấy tích phân nên vi phân $dA = dx dy$ của phần tử tứ giác không bằng vi phân $d\xi d\eta$ của hình chữ nhật. Có thể chỉ ra mối qua hệ giữa các miền này là

$$dx dy = |\mathbf{J}(\xi, \eta)| d\xi d\eta, \quad (5.52)$$

với

$$\mathbf{J}(\xi, \eta) = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \quad (5.53a)$$

được biết đến với tên gọi là *ma trận Jacobian* của ánh xạ. Thay (5.50) vào (5.51) và lấy vi phân, ta được các phần tử của ma trận Jacobian là

$$\begin{aligned} J_{11} &= \frac{1}{4} [-(1 - \eta)x_1 + (1 - \eta)x_2 + (1 + \eta)x_3 - (1 + \eta)x_4] \\ J_{12} &= \frac{1}{4} [-(1 - \eta)y_1 + (1 - \eta)y_2 + (1 + \eta)y_3 - (1 + \eta)y_4] \\ J_{21} &= \frac{1}{4} [-(1 - \xi)x_1 - (1 + \xi)x_2 + (1 + \xi)x_3 + (1 - \xi)x_4] \\ J_{22} &= \frac{1}{4} [-(1 - \xi)y_1 - (1 + \xi)y_2 + (1 + \xi)y_3 + (1 - \xi)y_4]. \end{aligned} \quad (5.53b)$$

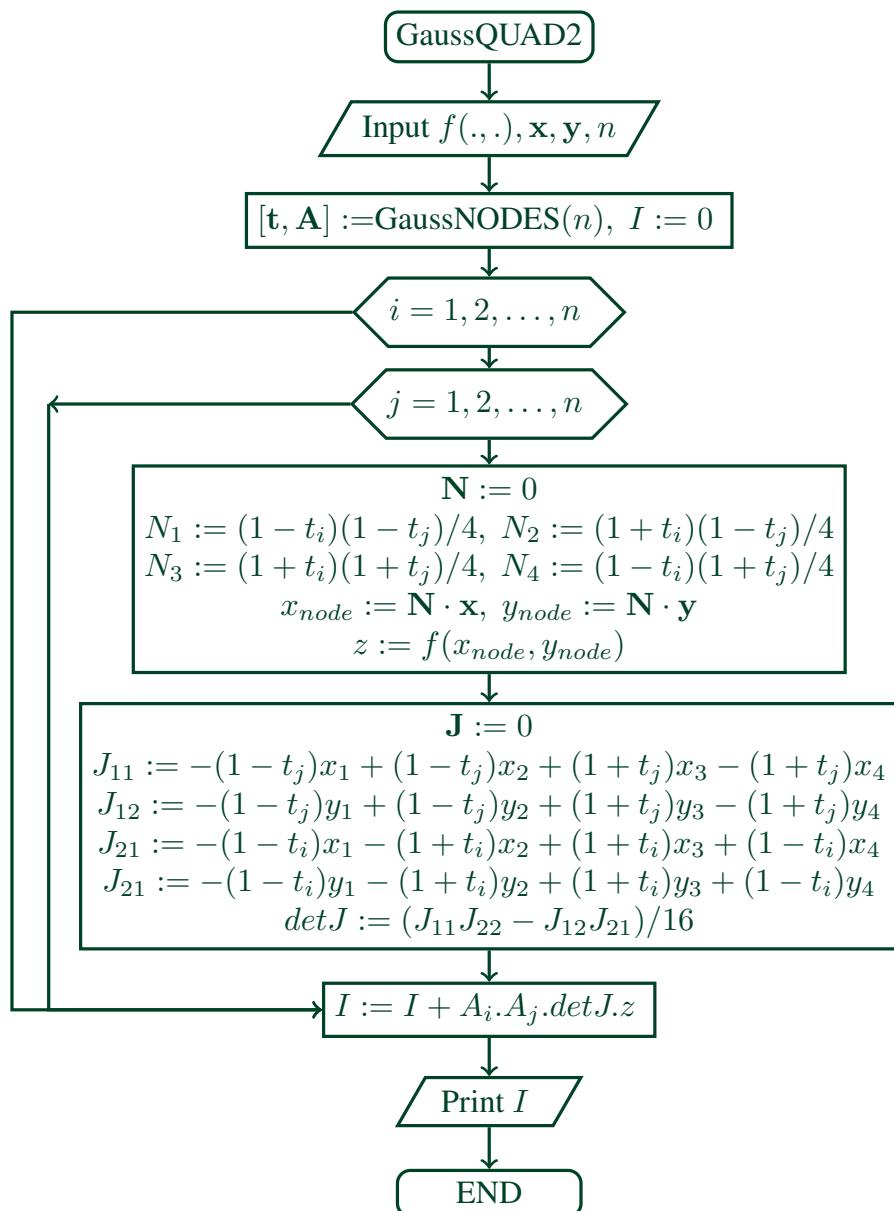
Bây giờ ta có thể viết

$$\iint_A f(x, y) dx dy = \int_{-1}^1 \int_{-1}^1 f[x(\xi, \eta), y(\xi, \eta)] |\mathbf{J}(\xi, \eta)| d\xi d\eta. \quad (5.54)$$

Vì vế phải lấy tích phân trên miền hình chữ nhật “chuẩn” nên có thể tính toán bằng công thức (5.49). Thay hàm $f(\xi, \eta)$ trong công thức (5.49) bởi hàm dưới dấu tích phân trong công thức (5.54) ta có công thức cho phương pháp cầu phương Gauss-Legendre cho miền tứ giác

$$I = \sum_{i=1}^n \sum_{j=1}^n A_i A_j f[x(\xi_i, \eta_j), y(\xi_i, \eta_j)] |\mathbf{J}(\xi_i, \eta_j)|. \quad (5.55)$$

Các toạ độ ξ, η của các điểm tích phân và các trọng số có thể thu được từ Bảng 5.9.



Hình 5.12: Sơ đồ khối tích phân bội trên miền tứ giác bằng phương pháp cầu phương Gaussian

gaussQuad2.

Hàm gaussQuad2 tính $\iint_A f(x, y) dx dy$ trên miền lấp tích phân là phần tử tứ giác bằng phương pháp cầu phương với n bậc tích phân. Tứ giác được các định bằng các mảng x và y mà chứa toạ độ của bốn góc xếp theo thứ tự ngược chiều kim đồng hồ. Định thức của ma trận Jacobian thu được bằng cách gọi hàm $\det J$; ánh xạ được thực hiện bởi map . Các trọng số và các giá trị ξ, η của các điểm tích phân được tính toán bằng hàm gaussNodes đã đề cập ở mục trước.

```

function I=gaussQuad2(func,x,y,n)
% Gauss-Legendre quadrature over a quadrilateral.
% USAGE:I=gaussQuad2(func,x,y,n)
% INPUT:
% func=handle of function to be integrated.
% x = [x1;x2;x3;x4] = x-coordinates of corners.
% y = [y1;y2;y3;y4] = y-coordinates of corners.
% n = order of integration
% OUTPUT:
% I = integral

[t,A]=gaussNodes(n);I=0;
for i=1:n
    for j=1:n
        [xNode,yNode] = map(x,y,t(i),t(j));
        z = feval(func,xNode,yNode);
        detJ = jac(x,y,t(i),t(j));
        I=I+A(i)*A(j)*detJ*z;
    end
end

function detJ = jac(x,y,s,t)
% Computes determinant of Jacobian matrix.
J = zeros(2);
J(1,1)=-(1-t)*x(1)+(1-t)*x(2)...
+(1+t)*x(3)-(1+t)*x(4);
J(1,2)=-(1-t)*y(1)+(1-t)*y(2)...
+(1+t)*y(3)-(1+t)*y(4);
J(2,1)=-(1-s)*x(1)-(1+s)*x(2)...
+(1+s)*x(3)+(1-s)*x(4);

```

```

J(2, 2) = - (1-s) * y(1) - (1+s) * y(2) ...
+ (1+s) * y(3) + (1-s) * y(4);
detJ = (J(1, 1) * J(2, 2) - J(1, 2) * J(2, 1)) / 16;

```

```

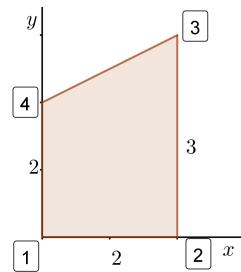
function [xNode, yNode] = map(x, y, s, t)
% Computes x and y-coordinates of nodes.
N = zeros(4, 1);
N(1) = (1-s) * (1-t) / 4;
N(2) = (1+s) * (1-t) / 4;
N(3) = (1+s) * (1+t) / 4;
N(4) = (1-s) * (1+t) / 4;
xNode = dot(N, x); yNode = dot(N, y);

```

VÍ DỤ 5.4.1. Ước lượng tích phân

$$I = \iint_A f(x, y) dx dy$$

trên tứ giác như trong hình



GIẢI. Toạ độ các góc của hình tứ giác là

$$\mathbf{x}^T = [0 \quad 2 \quad 2 \quad 0], \quad \mathbf{y}^T = [0 \quad 0 \quad 3 \quad 2].$$

Ánh xạ là

$$\begin{aligned}
 x(\xi, \eta) &= \sum_{k=1}^4 N_k(\xi, \eta) x_k \\
 &= 0 + \frac{(1+\xi)(1-\eta)}{4}(2) + \frac{(1+\xi)(1+\eta)}{4}(2) + 0 \\
 &= 1 + \xi \\
 y(\xi, \eta) &= \sum_{k=1}^4 N_k(\xi, \eta) y_k \\
 &= 0 + 0 + \frac{(1+\xi)(1+\eta)}{4}(3) + \frac{(1-\xi)(1+\eta)}{4}(2) \\
 &= \frac{(5+\xi)(1+\eta)}{4}
 \end{aligned}$$

nên thu được ma trận Jacobian

$$\mathbf{J}(\xi, \eta) = \begin{bmatrix} \frac{\partial x}{\xi} & \frac{\partial y}{\xi} \\ \frac{\partial x}{\eta} & \frac{\partial y}{\eta} \end{bmatrix} = \begin{bmatrix} 1 & \frac{1+\eta}{4} \\ 0 & \frac{5+\eta}{4} \end{bmatrix}.$$

Hệ số dẫn điện tích là

$$|\mathbf{J}(\xi, \eta)| = \frac{5+\xi}{4}.$$

Bây giờ ta có thể ánh xạ tích phân từ hình tứ giác vào hình chữ nhật chuẩn. Từ (5.54)

$$\begin{aligned}
 I &= \int_{-1}^1 \int_{-1}^1 \left[(1+\xi)^2 + \frac{(5+\xi)(1+\eta)}{4} \right] \frac{5+\xi}{4} d\xi d\eta \\
 &= \int_{-1}^1 \int_{-1}^1 \left(\frac{45}{16} + \frac{21}{8}\xi + \frac{29}{16}\xi^2 + \frac{1}{4}\xi^3 + \frac{25}{16}\eta + \frac{5}{8}\xi\eta + \frac{1}{16}\xi^2\eta \right) d\xi d\eta.
 \end{aligned}$$

Chú ý rằng các luỹ thừa lẻ của ξ và η bị triệt tiêu trên miền lấy tích phân đối xứng nên ta có

$$I = \int_{-1}^1 \int_{-1}^1 \left(\frac{45}{16} + \frac{29}{16}\xi^2 \right) d\xi d\eta = \frac{41}{3}.$$

□

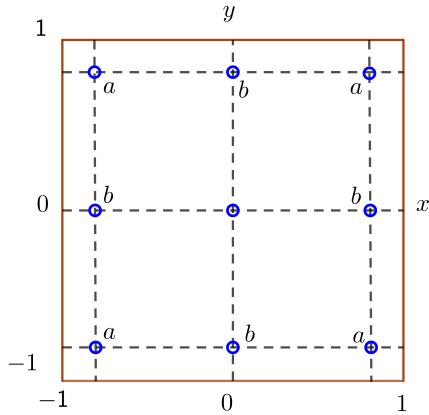
VÍ DỤ 5.4.2. Ước lượng tích phân

$$I = \int_{-1}^1 \int_{-1}^1 \cos \frac{\pi x}{2} \cos \frac{\pi y}{2} dx dy$$

bằng phương pháp cầu phương Gauss-Legendre bậc ba.

GIẢI. Sử dụng công thức cầu phương trong (5.49) ta có

$$I = \sum_{i=1}^3 \sum_{j=1}^3 A_i A_j \cos \frac{\pi x_i}{2} \cos \frac{\pi y_j}{2}.$$



Các điểm tích phân được chỉ ra như trong hình, toạ độ của chúng và các điểm trọng số tương ứng được liệt kê trong Bảng 5.9. Chú ý rằng trong tích phân này thì các điểm tích phân và các trọng số đối xứng qua các trục toạ độ. Do đó, các điểm gán nhãn a đóng góp bằng nhau cho I và các điểm gán nhãn b cũng tương tự. Do đó

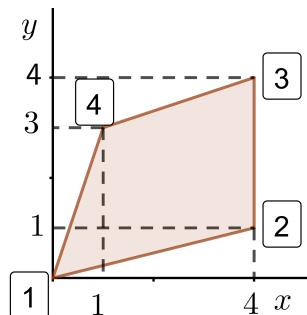
$$\begin{aligned}
 I &= 4(0.555556)^2 \cos^2 \frac{\pi(0.774597)}{2} \\
 &\quad + 4(0.555556)(0.888889) \cos \frac{\pi(0.774597)}{2} \cos \frac{\pi(0)}{2} \\
 &\quad + 4(0.888889)^2 \cos^2 \frac{\pi(0)}{2} \\
 &= 1.623391.
 \end{aligned}$$

Giá trị chính xác của tích phân là $16/\pi^2 \approx 1.621139$.

□

VÍ DỤ 5.4.3. Sử dụng hàm gaussQuad2 ước lượng cho $I = \iint_A f(x, y) dx dy$ trên tứ giác như trong hình, với

$$f(x, y) = (x - 2)^2(y - 2)^2.$$



Lưu ý sử dụng số nút tối thiểu để tính chính xác.

GIẢI. Yêu cầu bậc tích phân được xác định bằng hàm dưới dấu tích phân trong công

thức (5.54)

$$I = \int_{-1}^1 \int_{-1}^1 f[x(\xi, \eta), y(\xi, \eta)] |\mathbf{J}(\xi, \eta)| d\xi d\eta. \quad (\text{a})$$

Chú ý rằng $|\mathbf{J}(\xi, \eta)|$ được xác định trong (5.53b) là trùng phương. Vì $f(x, y)$ cũng là hàm trùng phương nên hàm dưới dấu tích phân (a) là hàm bậc 4 theo cả biến ξ và biến η . Do đó, bậc của tích phân bằng 3 ($n = 3$) là đủ để tính chính xác. Ta sử dụng hàm MATLAB sau để tính tích phân.

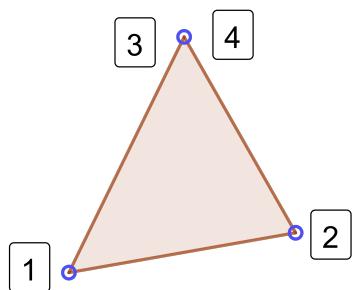
```
>> I=gaussQuad2 (@fex5_4_3, [0; 4; 4; 1], [0; 1; 4; 3], 3)
I=
11.3778
>>
```

M-file trả giá trị hàm dưới dấu tích phân là

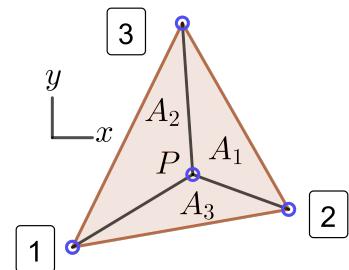
```
function z=fex5_4_3(x,y)
% Function used in Example 5.4.3.
z=((x-2)*(y-2))^2;
```

□

5.4.3 PHƯƠNG PHÁP CẦU PHƯƠNG TRÊN MỘT PHẦN TỬ TAM GIÁC



(a) Tam giác là tứ giác với hai góc trùng nhau



(b) Phần tử tam giác trong công thức cầu phương

Hình 5.13: Hình minh họa miền lấp tích phân bội trên hình tam giác

Một hình tam giác có thể coi như một hình tứ giác suy biến với hai góc trùng nhau như Hình 5.13a. Do đó, công thức tính tích phân trên một miền tứ giác có thể được sử

dụng cho phần tử tam giác. Tuy nhiên, sẽ thuận lợi hơn khi ta phát triển một phương pháp tính tích phân trên miền tam giác.

Xét phần tử tam giác như trong Hình 5.13b. Kẻ các đoạn thẳng nối điểm P với các góc tam giác để tạo ra các tam giác nhỏ với diện tích là A_1, A_2 và A_3 . *Toạ độ diện tích* của P được xác định bởi

$$\alpha_i = \frac{A_i}{A}, \quad i = 1, 2, 3, \quad (5.56)$$

với A là diện tích của phần tử. Vì $A_1 + A_2 + A_3 = A$ nên

$$\alpha_1 + \alpha_2 + \alpha_3 = 1. \quad (5.57)$$

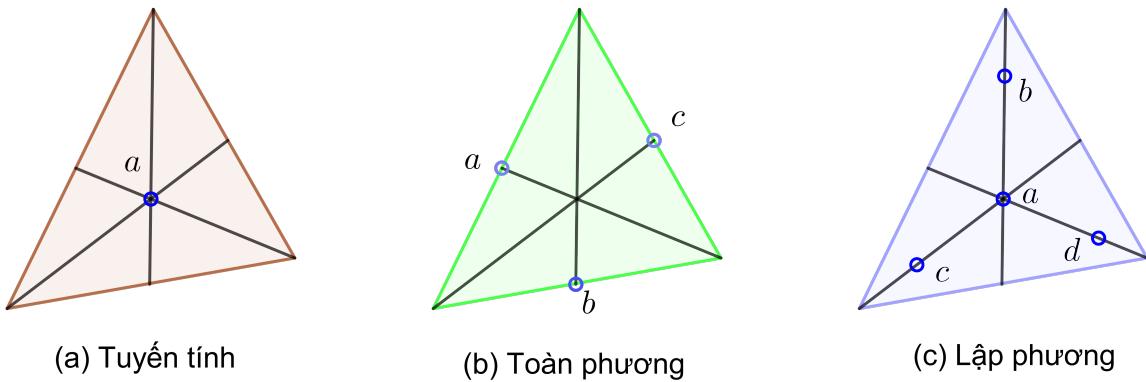
Chú ý rằng α_i nhận các giá trị từ 0 (khi P nằm trên cạnh đối diện đỉnh i) đến 1 (khi P là góc i).

Công thức đơn giản để tính A khi biết toạ độ (x_i, y_i) của các góc là

$$A = \frac{1}{2} \begin{vmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{vmatrix}. \quad (5.58)$$

Toạ độ diện tích được ánh xạ vào trong hệ toạ độ Catersian là

$$x(\alpha_1, \alpha_2, \alpha_3) = \sum_{i=1}^3 \alpha_i x_i, \quad y(\alpha_1, \alpha_2, \alpha_3) = \sum_{i=1}^3 \alpha_i y_i. \quad (5.59)$$



Hình 5.14: Các điểm tích phân trên phần tử tam giác

Công thức tích phân trên phần tử tam giác là

$$\iint_A f[x(\boldsymbol{\alpha}), y(\boldsymbol{\alpha})] dA = A \sum_k W_k f[x(\boldsymbol{\alpha}_k), y(\boldsymbol{\alpha}_k)] \quad (5.60)$$

với α_k là toạ độ diện tích của điểm lấy tích phân thứ k và W_k là các trọng số. Vị trí của các điểm tích phân được chỉ ra trong Hình 5.14, và các giá trị α với các trọng số W_k tương ứng đƣợng liệt kê trong Bảng 5.13. Công thức cầu phƣơng (5.60) là chính xác nếu $f(x, y)$ là đa thức có bậc tương ứng.

Bậc của $f(x, y)$	Điểm	α_k	W_k
(a) Tuyến tính	a	$1/3, 1/3, 1/3$	1
(b) Toàn phƣơng	a	$1/2, 0, 1/2$	$1/3$
	b	$1/2, 1/2, 0$	$1/3$
	c	$0, 1/2, 1/2$	$1/3$
(c) Lập phƣơng	a	$1/3, 1/3, 1/3$	$-27/48$
	b	$1/5, 1/5, 3/5$	$25/48$
	c	$3/5, 1/5, 1/5$	$25/48$
	d	$1/5, 3/5, 1/5$	$25/48$

Bảng 5.13: Các điểm tích phân, các trọng số và bậc tương ứng trong phƣơng pháp cầu phƣơng trên phần tử tam giác

↙ triangleQuad.

Hàm triangleQuad tính $\iint_A f(x, y)dx dy$ bằng công thức lập phƣơng trên miền tam giác, trường hợp (c) trong Hình 5.14. Tam giác được xác định bởi toạ độ các góc cho trong mảng x và y với các đỉnh được liệt kê theo chiều ngược chiều kim đồng hồ.

```
function I=triangleQuad(func,x,y)
% Cubic quadrature over a triangle.
% USAGE:I=triangleQuad(func,x,y)
% INPUT:
% func=handle of function tobe integrated.
% x = [x1;x2;x3] x-coordinates of corners.
% y = [y1;y2;y3] y-coordinates of corners.
% OUTPUT:
% I = integral

alpha = [1/3 1/3 1/3; 1/5 1/5 3/5; ...
          3/5 1/5 1/5; 1/5 3/5 1/5];
W = [-27/48; 25/48; 25/48; 25/48];
xNode = alpha*x; yNode = alpha*y;
```

```

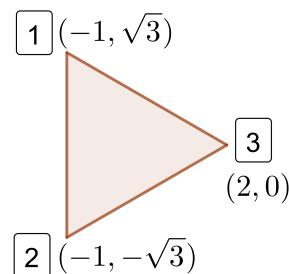
A = (x(2)*y(3) - x(3)*y(2)...
    - x(1)*y(3) + x(3)*y(1)...
    + x(1)*y(2) - x(2)*y(1))/2;
sum=0;
for i=1:4
    z = feval(func,xNode(i),yNode(i));
    sum = sum + W(i)*z;
end
I = A*sum

```

VÍ DỤ 5.4.4. Tính $I = \iint_A f(x, y) dx dy$ trên tam giác đều như trong hình với

$$f(x, y) = \frac{1}{2}(x^2 + y^2) - \frac{1}{6}(x^3 - 3xy^2) - \frac{2}{3}$$

Sử dụng công thức cầu phương để tính trên (1) một hình tứ giác (2) một hình tam giác.



GIẢI. (1) **sử dụng công thức cầu phương trên hình tứ giác.** Ta coi hình tam giác là hình tứ giác với đỉnh 3 và đỉnh 4 trùng nhau. Toạ độ các đỉnh theo chiều ngược chiều kim đồng hồ được xác định $x = [-1, -1, 2, 2]$ và $y = [\sqrt{3}, -\sqrt{3}, 0, 0]$. Để xác định số điểm tích phân yêu cầu nhỏ nhất để tính chính xác ta phải nghiên cứu hàm dưới dấu tích phân trong công thức (5.54) là $f[x(\xi, \eta), y(\xi, \eta)] |\mathbf{J}(\xi, \eta)|$. Vì $|\mathbf{J}(\xi, \eta)|$ là hàm trùng phương và $f(x, y)$ là hàm lập phương theo x nên hàm dưới dấu tích phân là đa thức bậc 5 theo x . Do đó ta cần 4 điểm tích phân để tính chính xác. Lệnh thực hiện tính toán tương tự như trong Ví dụ 5.4.3.

```

>> I=gaussQuad2 (@fex5_4, [-1;-1;2;2], ...
[sqrt(3);-sqrt(3);0;0], 3)
I=
-1.5588

```

Hàm trả về hàm dưới dấu tích phân $z = f(x, y)$ là

```

function z=fex5_4_4(x,y)
% Function used in Example 5.4.4
z = (x^2 + y^2)/2 - (x^3 - 3*x*y^2)/6 - 2/3;

```

(2) sử dụng công thức cầu phương trên hình tam giác. Lệnh sau đây thực hiện phuowng pháp cầu phương trên phần tử tam giác

```

>>I=triangleQuad(@fex5_4_4, [-1; -1; 2], [sqrt(3); -sqrt(3); 0])
I=
-1.5588

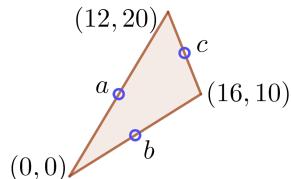
```

Vì hàm dưới dấu tích phân là hàm lập phương nên tích phân là chính xác.

□

VÍ DỤ 5.4.5. Toạ độ các góc của tam giác là $(0,0)$, $(16,10)$ và $(12,20)$. Tính $\iint_A f(x,y)dx dy$ trên miền tam giác trên.

GIẢI. Bởi vì hàm $f(x,y)$ là toàn phương nên phương pháp cầu phương trên ba điểm



tích phân như trong trên là chính xác. Chú ý rằng các điểm tích phân là trung điểm của các cạnh nên toạ độ của chúng là $(6,10)$, $(8,5)$ và $(14,15)$. Diện tích của hình tam giác thu được từ (5.58) là

$$A = \frac{1}{2} \begin{vmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{vmatrix} = \frac{1}{2} \begin{vmatrix} 1 & 1 & 1 \\ 0 & 16 & 12 \\ 0 & 10 & 20 \end{vmatrix} = 100$$

Từ (5.60) ta có

$$\begin{aligned}
I &= A \sum_{k=a}^c W_k f[x_k, y_k] \\
&= 100 \left[\frac{1}{3} f(6, 10) + \frac{1}{3} f(8, 5) + \frac{1}{3} f(14, 15) \right] \\
&= \frac{100}{3} [(6^2 - 10^2) + (8^2 - 5^2) + (14^2 - 15^2)] = 1800.
\end{aligned}$$

□

5.5 MỘT SỐ HÀM MATLAB TÍNH ĐẠO HÀM VÀ TÍCH PHÂN HÀM SỐ

Một số hàm Matlab thông dụng dùng để tính toán đạo hàm và tích phân hàm số sẽ được giới thiệu sau đây.

`d = diff(y)` trả giá trị các sai phân $d(i) = y(i+1) - y(i)$. Chú ý rằng `length(d) = length(y) - 1`.

`dn = diff(y, n)` trả các sai phân thứ n ; tức là $d2(i) = d(i+1) - d(i)$, $d3(i) = d2(i+1) - d2(i)$, v.v. Chú ý `length(dn) = length(y) - n`.

`d = gradient(y, h)` trả về xấp xỉ sai phân hữu hạn tại của dy/dx tại mỗi điểm với h là khoảng cách giữa các điểm.

`d2 = del2(y, h)` trả về xấp xỉ sai phân hữu hạn tại của $(d^2y/d^2x)/4$ tại mỗi điểm với h là khoảng cách giữa các điểm.

`I = quad(func, a, b, tol)` sử dụng công thức Simpson tính tích phân $I = \int_a^b f(x)dx$ với sai số cho phép `tol` (mặc định là $1.0e-6$). Để tăng quá trình thực hiện, các phép tính trên vectơ của hàm `func` sử dụng các toán tử mảng `.*`, `./` và `.^`.

`I = dblquad(func, xMin, xMax, yMin, yMax, tol)` sử dụng hàm `quad` tính tích phân trên hình chữ nhật

$$I = \int_{yMin}^{yMax} \int_{xMin}^{xMax} f(x, y) dx dy.$$

TÓM TẮT CHƯƠNG 5

Nội dung chính của Chương 5 trình bày về tính số vi phân và tính số đạo hàm. Việc tính số đạo hàm được trình bày trong Mục 5.2 với hai nhóm phương pháp có bản là phương pháp sai phân hữu hạn (trong Mục 5.2.2) và phương pháp sử dụng đa thức nội suy (trong Mục 5.2.4). Bên cạnh đó, ngoại suy Richardson cũng được giới thiệu (trong Mục 5.2.3) để nâng độ chính xác của các phương pháp xấp xỉ đạo hàm.

Phần còn lại của chương này dành để giới thiệu về cách xấp xỉ tích phân xác định

và tích phân bội. Mục 5.3 giới thiệu các phương pháp tính số tích phân gồm nhóm các phương pháp Newton-Cotes và nhóm các công thức cầu phương Gaussian. Một số phương pháp Newton-Contes được giới thiệu trong Mục 5.3.2 gồm:

1. Công thức hình thang
2. Công thức simpsom 1/3
3. Công thức simpsom 3/8

Một số phương pháp cầu phương Gaussian được giới thiệu trong Mục 5.3.4 gồm:

1. Phương pháp cầu phương Gauss-Legendre
2. Phương pháp cầu phương Gauss-Chebysev
3. Phương pháp cầu phương Gauss-Laguerre
4. Phương pháp cầu phương Gauss-Hermit
5. Phương pháp cầu phương Gauss với tính kỳ dị logarit

Bên cạnh đó, phương pháp ngoại suy Richardson cũng được áp dụng cho các công thức Newton-Cotes để nâng độ chính xác. Tài liệu này trình bày một trường hợp như vậy và gọi là tích phân Romberg (trong Mục 5.3.3).

Mục 5.4 trình bày việc áp dụng công thức cầu phương vài tích tích phân bội với hai phương pháp được giới thiệu là Công thức cầu phương Gauss trên miền tứ giác (trong Mục 5.4.2) và Công thức cầu phương Gauss trên miền tam giác (trong Mục 5.4.3).

BÀI TẬP CHƯƠNG 5

Bài tập tính số đạo hàm

Bài tập 5.1. Sử dụng xấp xỉ sai phân hữu hạn với sai số $\mathcal{O}(h^2)$ tính $f'(2.36)$ và $f''(2.36)$ từ dữ liệu

x	2.36	2.37	2.38	2.39
y	0.85866	0.86289	0.86710	0.87129

Bài tập 5.2. Ước lượng $f'(1)$ và $f''(1)$ từ dữ liệu

x	0.97	1.00	1.05
y	0.85040	0.84147	0.82162

Bài tập 5.3. Cho dữ liệu

x	0.84	0.92	1.00	1.08	1.16
y	0.431711	0.398519	0.367879	0.339569	0.313486

Tính $f''(1)$ chính xác nhất mà bạn có thể.

Bài tập 5.4. Tính $f'(0.2)$ chính xác nhất mà bạn có thể từ dữ liệu

x	0	0.1	0.2	0.3	0.4
y	0.000000	0.078348	0.138910	0.192916	0.244981

Bài tập 5.5. Sử dụng nội suy đa thức tính f' , f'' tại $x = 0$ bằng cách sử dụng dữ liệu

x	-2.2	-0.3	0.8	1.9
$f(x)$	15.180	10.962	1.920	-2.040

Bài tập tính số tích phân

Bài tập 5.6. Sử dụng công thức hình thang để quy ước lượng $\int_0^{\pi/4} \ln(1 + \tan x) dx$. Giải thích kết quả.

Bài tập 5.7. Ước lượng $\int_{-1}^1 \cos(2 \cos^{-1}(x)) dx$ bằng công thức Simpson 1/3 sử dụng 2,4 và 6 đoạn. Giải thích kết quả.

Bài tập 5.8. Xấp xỉ $\int_0^\infty (1 + x^4)^{-1} dx$ bằng phương pháp hình thang sử dụng 5 đoạn và so sánh với kết quả chính xác là 0.23475. *Gợi ý:* Thực hiện đổi biến $x^3 = 1/t$.

Bài tập 5.9. Ước lượng $\int_0^2 (x^5 + 3x^3 - 2) dx$ bằng tích phân Romberg.

Bài tập 5.10. Ước lượng $\int_0^\pi f(x) dx$ chính xác nhất có thể biết $f(x)$ được xác định bởi dữ liệu

x	0	$\pi/4$	$\pi/2$	$3\pi/4$	π
$f(x)$	1.0000	0.3431	0.2500	0.3431	1.0000

Bài tập 5.11. Ước lượng

$$\int_0^1 \frac{\sin x}{\sqrt{x}} dx$$

bằng tích phân Romberg. *Gợi ý:* Đổi biến để khử tính không xác định tại $x = 0$.

Bài tập 5.12. Sử dụng chương trình máy tính ước lượng

$$\int_0^{\pi/4} \frac{dx}{\sqrt{\sin x}}$$

bằng tích phân Romberg. *Gợi ý:* Sử dụng đổi biến $\sin x = t^2$.

Bài tập cầu phương Gaussian

Bài tập 5.13. Ước lượng

$$\int_0^\pi \frac{\ln x}{x^2 - 2x + 2} dx$$

bằng phương pháp cầu phương Gauss-Legendre sử dụng

a. 2 nút

b. 4 nút

Bài tập 5.14. Sử dụng phương pháp cầu phương Gauss-Legendre để ước lượng $\int_0^\infty (1 - x^2)^2 e^{-x} dx$.

Bài tập 5.15. Sử dụng phương pháp cầu phương Gauss-Chebysev với 6 nút để ước lượng

$$\int_0^{\pi/2} \frac{dx}{\sqrt{\sin x}}.$$

So sánh kết quả với giá trị chính xác của tích phân là 2.62206. *Gợi ý:* thay $\sin x = t^2$.

Bài tập 5.16. Có bao nhiêu nút được yêu cầu khi sử dụng phương pháp cầu phương Gauss-Laguerre để ước lượng $\int_0^\infty e^{-x} \sin x dx$ tới 6 chữ số thập phân.

Bài tập 5.17. Ước lượng tích phân sau chính xác nhất có thể

$$\int_0^1 \frac{2x + 1}{\sqrt{x(1-x)}} dx.$$

Gợi ý: thay $x = (1 + t)/2$.

Bài tập 5.18. Ước lượng tích phân

$$\int_0^\infty \frac{xdx}{e^x + 1}$$

tới 6 ssô thập phân. *Gợi ý:* thay $e^x = t$.

Bài tập 5.19. Phương trình của ellipse là $x^2/a^2 + y^2/b^2 = 1$. Viết chương trình tính chiều dài

$$S = 2 \int_{-a}^a \sqrt{1 + (dy/dx)^2} dx$$

của chu vi tới 4 chữ số thập phân với a, b cho trước. Kiểm tra chương trình với $a = 1$ và $b = 2$.

Bài tập 5.20. Hàm sai số, là một hàm quan trọng trong thống kê, được định nghĩa là

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^\infty e^{-t^2} dt.$$

Viết chương trình theo phương pháp cầu phương Gauss-Legendre tính giá trị của hàm erf , với x cho trước, chính xác đến 6 chữ số thập phân. Chú ý rằng $\operatorname{erf}(x) = 1.000000$ (chính xác đến 6 chữ số thập phân) khi $x > 5$. Kiểm tra chương trình biết rằng $\operatorname{erf}(1.0) = 0.842701$.

Bài tập tính số tích phân bội

Bài tập 5.21. Sử dụng phương pháp cầu phương Gauss-Legendre tính

$$I = \int_{-1}^1 \int_{-1}^1 (1 - x^2)(1 - y^2) dx dy$$

Bài tập 5.22. Ước lượng tích phân sau bằng phương pháp cầu phương Gauss-Legendre

$$I = \int_{y=2}^2 \int_{x=0}^3 x^2 y^2 dx dy$$

Bài tập 5.23. Tính giá trị xấp xỉ của

$$I = \int_{-1}^1 \int_{-1}^1 e^{-(x^2+y^2)} dx dy$$

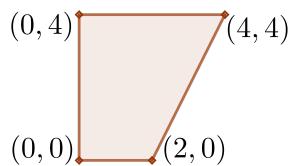
bằng phương pháp cầu phương Gauss-Legendre với số bậc tích phân là (2) hai và (b) ba. (Giá trị đúng của tích phân là 2.230985).

Bài tập 5.24. Sử dụng phương pháp cầu phương Gauss-Legendre bốn bậc để xấp xỉ giá trị của

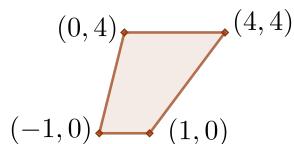
$$I = \int_{-1}^1 \int_{-1}^1 \cos \frac{\pi(x-y)}{2} dx dy.$$

(Giá trị đúng của tích phân là 1.621139).

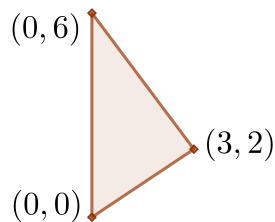
Bài tập 5.25. Ánh xạ tích phân $\iint_A xy dx dy$ từ miền tứ giác về hình chữ nhật chuẩn sau đó phân tích và tính toán.



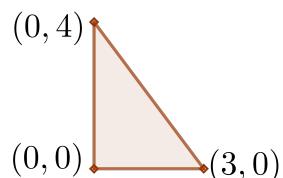
Bài tập 5.26. Ánh xạ tích phân $\iint_A x dx dy$ từ miền tứ giác về hình chữ nhật chuẩn sau đó phân tích và tính toán.



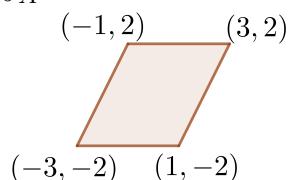
Bài tập 5.27. Sử dụng công thức cầu phương tính $\iint_A x^2 dx dy$ trên miền tam giác như hình vẽ.



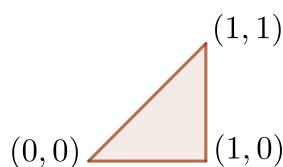
Bài tập 5.28. Ước lượng $\iint_A (3-x)y dx dy$ trên miền như hình vẽ.



Bài tập 5.29. Ước lượng $\iint_A xy(2-x^2)(2-xy) dx dy$ trên miền như hình vẽ.



Bài tập 5.30. Ước lượng $\iint_A (1-x)(y-x) y dx dy$ trên miền tam giác như hình vẽ.



CHƯƠNG 6

TÍNH GẦN ĐÚNG NGHIỆM CỦA PHƯƠNG TRÌNH VI PHÂN

6.1	Giới thiệu	234
6.2	Phương pháp chuỗi Taylor	235
6.3	Các phương pháp Runge-Kutta	241
6.3.1	Phương pháp Euler	241
6.3.2	Phương pháp Runge-Kutta bậc-hai	243
6.3.3	Phương pháp Runge-Kutta bậc-bốn	246
6.4	Tính ổn định và tính cương	253
6.4.1	Tính ổn định của phương pháp Euler	254
6.4.2	Tính cương của bài toán	255
6.5	Phương pháp Runge-Kutta thích ứng	257
6.6	Phương pháp Bulirsch-Stoer	265
6.6.1	Phương pháp trung điểm	265
6.6.2	Ngoại suy Richardson	266
6.6.3	Thuật toán Bulirsch-Stoer	271
6.7	Một số hàm Matlab giải phương trình vi phân	274
PHỤ LỤC A: Danh sách các hàm Matlab được lập trình trong sách		280
CHỈ MỤC		283
TÀI LIỆU THAM KHẢO		285

6.1 GIỚI THIỆU

Dạng tổng quát của phương trình vi phân cấp một là

$$y' = f(x, y) \quad (6.1a)$$

với $y' = dy/dx$ và $f(x, y)$ là hàm số cho trước. Nghiệm của phương trình này có chứa một hằng số tuỳ ý (hằng số của tích phân). Để tìm hằng số này, chúng ta phải biết được một điểm trên đường cong nghiệm, có nghĩa là y phải được xác định tại một số giá trị của x , chẳng hạn tại $x = a$. Ta viết điều kiện phụ trợ này là

$$y(a) = \alpha. \quad (6.1b)$$

Một phương trình vi phân bậc n

$$y^{(n)} = f(x, y, y', \dots, y^{(n-1)}) \quad (6.2)$$

luôn có thể biến đổi về hệ n phương trình vi phân bậc một. Nếu sử dụng ký hiệu

$$y_1 = y, \quad y_2 = y', \quad y_3 = y'', \quad y_n = y^{(n-1)} \quad (6.3)$$

thì hệ phương trình vi phân bậc một tương đương là

$$y'_1 = y_2, \quad y'_2 = y_3, \quad y'_3 = y_4, \dots, \quad y'_n = f(x, y_1, y_2, \dots, y_n). \quad (6.4a)$$

Trong trường hợp biết được n điều kiện phụ trợ của nghiệm cần tìm. Nếu các điều kiện này tại một giá trị nào đó của x thì bài toán đang xét được gọi là *bài toán giá trị ban đầu*. Khi đó, các điều kiện phụ trợ, gọi là *các điều kiện ban đầu*, có dạng

$$y_1(a) = \alpha_1, \quad y_2(a) = \alpha_2, \quad y_3(a) = \alpha_3, \quad \dots, \quad y_n(a) = \alpha_n. \quad (6.4b)$$

Nếu y_i được xác định tại một số giá trị khác nhau của x thì bài toán được gọi là *bài toán giá trị biên*.

Lấy ví dụ

$$y'' = -y, \quad y(0) = 1, \quad y'(0) = 0$$

là một bài toán giá trị ban đầu vì cả hai điều kiện phụ trợ cho nghiệm đều được xác định tại $x = 0$. Mặt khác

$$y'' = -y, \quad y(0) = 1, \quad y(\pi) = 0$$

là một bài toán giá trị biên vì cả hai điều kiện được xác định tại các giá trị khác nhau của x .

Trong chương này chúng ta chỉ xét bài toán giá trị ban đầu. Để ngắn gọn, chúng ta sẽ ký hiệu hệ phương trình vi phân bôc một dưới dạng ma trận. Chẳng hạn, hệ phương trình (6.4) được viết là

$$\mathbf{y}' = \mathbf{F}(x, \mathbf{y}), \quad \mathbf{y}(a) = \boldsymbol{\alpha} \quad (6.5a)$$

với

$$\mathbf{F}(x, \mathbf{y}) = \begin{bmatrix} y_2 \\ y_3 \\ \vdots \\ y_n \\ f(x, \mathbf{y}) \end{bmatrix}. \quad (6.5b)$$

Một nghiệm số của hệ phương trình vi phân thường được cho dưới dạng một bảng liệt kê các giá trị rời rạc của y theo x .

6.2 PHƯƠNG PHÁP CHUỖI TAYLOR

Phương pháp chuỗi Taylor là phương pháp đơn giản và có khả năng cho độ chính xác cao. Ý tưởng của nó dựa trên việc chặt bỏ các biểu thức phía sau trong khai triển Taylor của hàm số y theo biến x

$$\mathbf{y}(x + h) \approx \mathbf{y}(x) + \mathbf{y}'(x)h + \frac{1}{2!}\mathbf{y}''(x)h^2 + \frac{1}{3!}\mathbf{y}'''(x)h^3 + \cdots + \frac{1}{m!}\mathbf{y}^{(m)}(x)h^m. \quad (6.6)$$

Bởi vì công thức (6.6) dự đoán y tại $x + h$ từ thông tin của biến tại x nên nó cũng là công thức tính số tích phân. Biểu thức cuối cùng được giữ lại trong chuỗi xác định *bậc của tích phân*. Với chuỗi trong (6.6) có bậc tích phân là m .

Sai số phương pháp là sai số chặt (truncation error) được xác định bởi công thức

$$E = \frac{1}{(m+1)!}\mathbf{y}^{(m+1)}(\xi)h^{m+1}, \quad x < \xi < x + h.$$

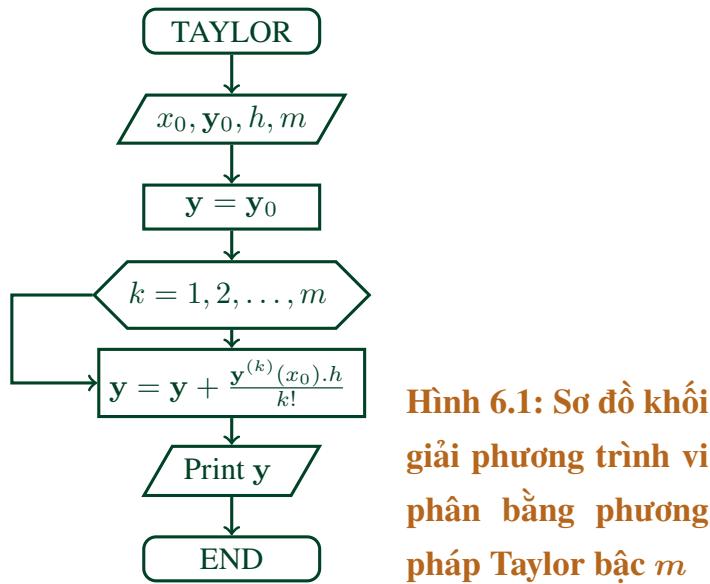
Sử dụng xấp xỉ sai phân hữu hạn

$$\mathbf{y}^{(m+1)} \approx \frac{\mathbf{y}^{(m)}(x+h) - \mathbf{y}^{(m)}(x)}{h}$$

ta thu được dạng dễ sử dụng hơn

$$E \approx \frac{h^m}{(m+1)!} \left[\mathbf{y}^{(m)}(x+h) - \mathbf{y}^{(m)}(x) \right] \quad (6.7)$$

cũng là công thức mà có thể kết hợp trong thuật toán để theo dõi sai số của từng bước. Quá trình xấp xỉ nghiệm của phương trình vi phân bằng phương pháp Taylor được minh họa bằng sơ đồ khối trong Hình 6.1.



**Hình 6.1: Sơ đồ khối
giải phương trình vi
phân bằng phương
pháp Taylor bậc m**

taylor

Hàm `taylor` thực hiện phương pháp chuỗi Taylor với bậc tích phân bằng bốn. Nó có thể xử lý tuỳ ý số lượng các phương trình vi phân bậc một $y'_i = f_i(x, y_1, y_2, \dots, y_n), i = 1, 2, \dots, n$. Người dùng được yêu cầu cung cấp hàm `deriv` trả về mảng $4 \times n$

$$\mathbf{d} = \begin{bmatrix} (y')^T \\ (y'')^T \\ (y''')^T \\ (y^{(4)})^T \end{bmatrix} = \begin{bmatrix} y'_1 & y'_2 & \dots, y'_n \\ y''_1 & y''_2 & \dots, y''_n \\ y'''_1 & y'''_2 & \dots, y'''_n \\ y^{(4)}_1 & y^{(4)}_2 & \dots, y^{(4)}_n \end{bmatrix}$$

Hàm `taylor` trả về các mảng `xSol` và `ySol` chứa các giá trị của x, y tại khoảng h .

```

function [xSol,ySol] = taylor(deriv,x,y,xStop,h)
% 4th-order Taylor series method of integration.
% USAGE: [xSol,ySol] = taylor(deriv,x,y,xStop,h)
% INPUT:
% deriv = handle of function that returns the matrix

```

```
% d = [dy/dx d^2y/dx^2 d^3y/dx^3 d^4y/dx^4].
% x,y = initial values; y must be a row vector.
% xStop = terminal value of x
% h = increment of x used in integration (h > 0).
% OUTPUT:
% xSol = x-values at which solution is computed.
% ySol = values of y corresponding to the x-values.

if size(y,1)>1;y=y';end % y must be row vector
xSol = zeros(2,1); ySol = zeros(2,length(y));
xSol(1) = x; ySol(1,:) = y;
k=1;
while x < xStop
    h = min(h,xStop - x);
    d = feval(deriv,x,y); % Derivatives of [y]
    hh=1;
    for j=1:4 %Build Taylor series
        hh = hh*h/j; % hh = h^j/j!
        y=y+d(j,:)*hh;
    end
    x=x+h; k=k+1;
    xSol(k) = x; ySol(k,:) = y; % Store current soln.
end
```

printSol

Hàm printSol thực hiện in kết quả xSol và ySol ra dạng bảng. Số lượng dữ liệu được điều khiển bởi tần số in ra freq. Ví dụ, nếu freq = 5 thì in ra bước thứ 0, 5, 10, Nếu freq = 0 thì chỉ có các giá trị ban đầu và giá trị cuối cùng sẽ được biểu diễn.

```
function printSol(xSol,ySol,freq)
    % Prints xSol and ySol arrays in tabular format.
    % USAGE: printSol(xSol,ySol,freq)
    % freq = printout frequency (prints every freq-th
    % line of xSol and ySol).

    [m,n] = size(ySol);
    if freq == 0; freq = m; end
```

```

head = ' x';
for i=1:n
    head = strcat(head, ' y', num2str(i));
end
fprintf(head); fprintf('\n')
for i=1:freq:m
    fprintf('%14.4e', xSol(i), ySol(i,:)); fprintf('\n')
end
if i ~= m; fprintf('%14.4e', xSol(m), ySol(m,:)); end

```

VÍ DỤ 6.2.1. Cho

$$y' + 4y = x^2, \quad y(0) = 1$$

xác định $y(0.2)$ bằng phương pháp chuỗi Taylor với số bậc tích phân là bốn và sử dụng một bước tích phân. Từ đó, tính sai số ước lượng theo công thức (6.7) và so sánh nó với sai số thật. Biết nghiệm giải tích của phương trình vi phân là

$$y = \frac{31}{32}e^{-4x} + \frac{1}{4}x^2 - \frac{1}{8}x + \frac{1}{32}.$$

GIẢI. Khai triển Taylor đến biểu thức thứ tư ta có

$$y(h) = y(0) + y'(0)h + \frac{1}{2!}y''(0)h^2 + \frac{1}{3!}y'''(0)h^3 + \frac{1}{4!}y^{(4)}(0)h^4. \quad (\text{a})$$

Đạo hàm phương trình vi phân ta thu được

$$\begin{aligned} y' &= -4y + x^2 \\ y'' &= -4y' + 2x = 16y - 4x^2 + 2x \\ y''' &= 16y' - 8x + 2 = -64y + 16x^2 - 8x + 2 \\ y^{(4)} &= -64y' + 32x - 8 = 256y - 64x^2 + 32x - 8. \end{aligned}$$

Do vậy

$$\begin{aligned} y'(0) &= -4(1) = -4 \\ y''(0) &= 1(1) = 16 \\ y'''(0) &= -64(1) + 2 = -62 \\ y^{(4)}(0) &= 256(1) - 8 = 248. \end{aligned}$$

Với $h = 0.2$ thì phương trình (a) trở thành

$$\begin{aligned} y(0.2) &= 1 + (-4)(0.2) + \frac{1}{2!}16(0.2)^2 + \frac{1}{3!}(-62)(0.2)^3 + \frac{1}{4!}248(0.2)^4 \\ &= 0.4539. \end{aligned}$$

Dựa theo (6.7), sai số chặt là là

$$E = \frac{h^4}{5!} \left[y^{(4)}(0.2) - y^{(4)}(0) \right]$$

với

$$\begin{aligned} y^{(4)}(0) &= 248 \\ y^{(4)}(0.2) &= 256(0.4539) - 64(0.2)^2 + 32(0.2) - 8 = 112.04. \end{aligned}$$

Do đó

$$E = \frac{(0.2)^4}{5!} [112.04 - 248] = -0.0018.$$

Kết quả chính xác từ nghiệm giải tích

$$y(0.2) = \frac{31}{32}e^{-4(0.2)} + \frac{1}{4}(0.2)^2 - \frac{1}{8}(0.2) + \frac{1}{32}0.4515$$

nên sai số thật là $0.4515 - 0.4539 = -0.0024$.

□

VÍ DỤ 6.2.2. Giải phương trình vi phân

$$y'' = -0.1y' - x, \quad y(0) = 0, y'(0) = 1$$

từ $x = 0$ đến 2 bằng phương pháp chuỗi Taylor với số bậc tích phân bằng bốn và sử dụng $h = 0.25$.

GIẢI. Với $y_1 = y$ và $y_2 = y'$ thì bài toán đã cho tương đương với hệ phương trình vi phân bậc một với điều kiện ban đầu là

$$\mathbf{y}' = \begin{bmatrix} y'_1 \\ y'_2 \end{bmatrix} = \begin{bmatrix} y_2 \\ -0.1y_2 - x \end{bmatrix}, \quad \mathbf{y}(0) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Lần lượt tính đạo hàm hai vế và thay thế vào ta có

$$\mathbf{y}'' = \begin{bmatrix} y'_2 \\ -0.1y'_2 - 1 \end{bmatrix} = \begin{bmatrix} -0.1y_2 - x \\ 0.11y_2 + 0.1x - 1 \end{bmatrix}$$

$$\mathbf{y}''' = \begin{bmatrix} -0.1y'_2 - 1 \\ 0.11y'_2 + 0.1 \end{bmatrix} = \begin{bmatrix} 0.01y_2 + 0.1x - 1 \\ -0.001y_2 - 0.01x + 0.1 \end{bmatrix}$$

$$\mathbf{y}^{(4)} = \begin{bmatrix} 0.01y'_2 + 0.1 \\ -0.001y'_2 - 0.01 \end{bmatrix} = \begin{bmatrix} -0.001y_2 - 0.01x + 0.1 \\ 0.0001y_2 + 0.001x - 0.01 \end{bmatrix}.$$

Do đó mảng đạo hàm yêu cầu bởi hàm taylor là

$$\mathbf{d} = \begin{bmatrix} y_2 & -0.1y_2 - x \\ -0.01y_2 - x & 0.001y_2 + 0.1x - 1 \\ 0.001y_2 + 0.1x - 1 & -0.0001y_2 - 0.01x + 0.1 \\ -0.0001y_2 - 0.01x + 0.1 & 0.0001y_2 + 0.001x - 0.01 \end{bmatrix}$$

và được tính toán bởi

```
function d=fex6_2_2(x,y)
% Derivatives used in Example 6.2.2
d = zeros(4,2);
d(1,1) = y(2);
d(1,2) = -0.1*y(2) - x;
d(2,1) = d(1,2);
d(2,2) = 0.01*y(2) + 0.1*x - 1;
d(3,1) = d(2,2);
d(3,2) = -0.001*y(2) - 0.01*x + 0.1;
d(4,1) = d(3,2);
d(4,2) = 0.0001*y(2) + 0.001*x - 0.01;
```

Thực hiện chạy chương trình và ta có kết quả

```
>> [x,y] = taylor(@fex6_2_2, 0, [0 1], 2, 0.25);
>> printSol(x,y,1)
```

x	y1	y2
0.0000e+000	0.0000e+000	1.0000e+000
2.5000e-001	2.4431e-001	9.4432e-001
5.0000e-001	4.6713e-001	8.2829e-001
7.5000e-001	6.5355e-001	6.5339e-001
1.0000e+000	7.8904e-001	4.2110e-001
1.2500e+000	8.5943e-001	1.3281e-001
1.5000e+000	8.5090e-001	-2.1009e-001

1.7500e+000 7.4995e-001 -6.0625e-001
 2.0000e+000 5.4345e-001 -1.0543e+000

Nghiệm giải tích của bài toán là

$$y = 100x - 5x^2 + 990(e^{-0.1x} - 1).$$

Từ đó ta thu được $y(2) = 0.54345$ và $y'(2) = -1.0543$, đây là kết quả trùng với kết quả số.

□

Điểm hạn chế của phương pháp chuỗi Taylor là yêu cầu tính đạo hàm đến cấp bốn của y . Những biểu thức này có thể trở nên rất dài và do đó rất dễ xảy ra nhầm lẫn khi tính toán. Hơn nữa, việc lập trình trở nên phức tạp vì phải tính từng đạo hàm.

6.3 CÁC PHƯƠNG PHÁP RUNGE-KUTTA

6.3.1 PHƯƠNG PHÁP EULER

Mục đích của các phương pháp Runge-Kutta là loại bỏ việc lặp lại quá trình tính đạo hàm các cấp bằng tay của phương trình vi phân. Vì công thức tích phân chuỗi Taylor bậc-một

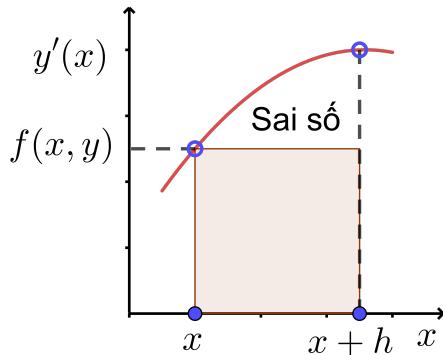
$$\mathbf{y}(x+h) = \mathbf{y}(x) + \mathbf{y}'(x)h = \mathbf{y}(x) + \mathbf{F}(x, \mathbf{y})h \quad (6.8)$$

không liên quan đến việc tính các đạo hàm của y nên có thể coi như công thức này là phương pháp Runge-Kutta bậc-một. Phương pháp Runge-Kutta bậc-một còn được biết đến với tên gọi nổi tiếng khác là *phương pháp Euler*. Do sai số chát quá lớn (vì giữ lại quá ít biểu thức) nên phương pháp này hiếm khi được sử dụng trong thực tế.

Công thức Euler được minh họa bằng đồ thị trong Hình 6.2. Để đơn giản, chúng ta giả thiết rằng chỉ có một biến phụ thuộc y , khi đó phương trình vi phân là $y' = f(x, y)$. Sự thay đổi trong nghiệm y giữa x và $x + h$ là

$$y(x+h) - y(x) = \int_x^{x+h} y' dx = \int_x^{x+h} f(x, y) dx$$

đây là diện tích của hình thang cong trong Hình 6.2. Công thức Euler thực chất là việc xấp xỉ diện tích hình thang cong bởi hình chữ nhật nên phần diện tích giữa hai hình



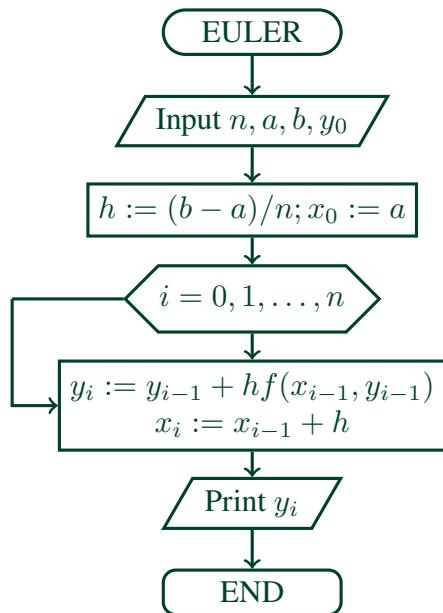
Hình 6.2: Đồ thị minh họa quá trình xấp xỉ nghiệm phương trình vi phân bằng công thức Euler

này là sai số chặt. Rõ ràng sai số chặt trong trường hợp này rất lớn, nó tỉ lệ thuận với độ dốc của đồ thị, tức là tỉ lệ với y'' .

Ta có thể sử dụng công thức (6.8) để xấp xỉ nghiệm của phương trình $y' = f(x, y)$, $y(a) = y_0$ trên đoạn $[a, b]$. Trước tiên ta chia đoạn $[a, b]$ thành n đoạn bởi các điểm $x_i = a + ih$, với $h = (b - a)/n$ và tại mỗi khoảng này thì thực hiện công thức lặp

$$y_i = y_{i-1} + h f(x_{i-1}, y_{i-1}), \quad i = 1, 2, \dots, n.$$

Quá trình xấp xỉ nghiệm của phương trình vi phân trên đoạn $[a, b]$ được mô tả bằng sơ đồ khối cho trong Hình 6.3



Hình 6.3: Sơ đồ khối giải phương trình vi phân bằng phương pháp Euler

6.3.2 PHƯƠNG PHÁP RUNGE-KUTTA BẬC-HAI

Để đạt được phương pháp Runge-Kutta bậc-hai, chúng ta giả sử rằng công thức tích phân có dạng

$$\mathbf{y}(x+h) = \mathbf{y}(x) + c_0 \mathbf{F}(x, \mathbf{y})h + c_1 \mathbf{F}[x+ph, \mathbf{y}+qh\mathbf{F}(x, \mathbf{y})]h \quad (\text{a})$$

và cố gắng tìm các tham số c_0, c_1, p và q bằng cách so sánh (a) với chuỗi Taylor

$$\begin{aligned} \mathbf{y}(x+h) &= \mathbf{y}(x) + \mathbf{y}'(x)h + \frac{1}{2}\mathbf{y}''(x)h^2 + \mathcal{O}(h^3) \\ &= \mathbf{y}(x) + \mathbf{F}(x, \mathbf{y})h + \frac{1}{2!}\mathbf{F}'(x, \mathbf{y})h^2 + \mathcal{O}(h^3). \end{aligned} \quad (\text{b})$$

Chú ý rằng

$$\mathbf{F}'(x, \mathbf{y}) = \frac{\partial \mathbf{F}}{\partial x} + \sum_{i=1}^n \frac{\partial \mathbf{F}}{\partial y_i} y'_i = \frac{\partial \mathbf{F}}{\partial x} + \sum_{i=1}^n \frac{\partial \mathbf{F}}{\partial y_i} F_i(x, \mathbf{y})$$

với n là số phương trình vi phân cấp một thành phần của \mathbf{F} . Ta có thể viết lại (b) dưới dạng

$$\mathbf{y}(x+h) = \mathbf{y}(x) + \mathbf{F}(x, \mathbf{y})h + \frac{1}{2} \left(\frac{\partial \mathbf{F}}{\partial x} + \sum_{i=1}^n \frac{\partial \mathbf{F}}{\partial y_i} F_i(x, \mathbf{y}) \right) h^2 + \mathcal{O}(h^3). \quad (\text{c})$$

Quay trở lại công thức (a), ta có thể áp dụng công thức khai triển Taylor nhiều biến cho biểu thức cuối

$$\mathbf{F}[x+ph, \mathbf{y}+qh\mathbf{F}(x, \mathbf{y})] = \mathbf{F}(x, \mathbf{y}) + \frac{\partial \mathbf{F}}{\partial x}ph + qh \sum_{i=1}^n \frac{\partial \mathbf{F}}{\partial y_i} F_i(x, \mathbf{y}) + \mathcal{O}(h^2)$$

nên công thức (a) trở thành

$$\mathbf{y}(x+h) = \mathbf{y}(x) + (c_0 + c_1)\mathbf{F}(x, \mathbf{y})h + c_1 \left[\frac{\partial \mathbf{F}}{\partial x}p + q \sum_{i=1}^n \frac{\partial \mathbf{F}}{\partial y_i} F_i(x, \mathbf{y}) \right] h^2 + \mathcal{O}(h^3). \quad (\text{d})$$

So sánh (c) và (d) ta thấy chúng đồng nhất khi

$$c_0 + c_1 = 1, \quad c_1 p = \frac{1}{2}, \quad c_1 q = \frac{1}{2}. \quad (\text{e})$$

Bởi vì (e) biểu diễn ba phương trình của bốn tham số chưa biết nên ta có thể gán một giá trị tuỳ ý cho một tham số. Ta liệt kê sau đây một số lựa chọn phổ biến với tên gọi tương ứng:

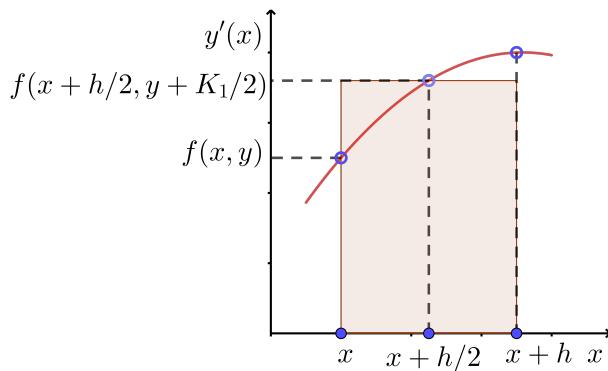
$c_0 = 0$	$c_1 = 1$	$p = 1/2$	$q = 1/2$	Phương pháp Euler cải tiến
$c_0 = 1/2$	$c_1 = 1/2$	$p = 1$	$q = 1$	Phương pháp Heun
$c_0 = 1/3$	$c_1 = 2/3$	$p = 3/4$	$q = 3/4$	Phương pháp Ralston

Tất cả các công thức này được phân loại là công thức Runge-Kutta bậc-hai và có ưu thế như nhau. Để thu được *phương pháp Euler cải tiến*, ta thay các tham số tương ứng vào (a)

$$\mathbf{y} = \mathbf{y}(x) + \mathbf{F} \left[x + \frac{h}{2}, \mathbf{y} + \frac{h}{2} \mathbf{F}(x, \mathbf{y}) \right] h. \quad (\text{f})$$

Công thức tích phân này có thể được tính toán thuận lợi bằng cách biểu diễn dưới dạng dãy

$$\begin{aligned} \mathbf{K}_1 &= h\mathbf{F}(x, \mathbf{y}) \\ \mathbf{K}_2 &= h\mathbf{F} \left(x + \frac{h}{2}, \mathbf{y} + \frac{1}{2}\mathbf{K}_1 \right) \\ \mathbf{y}(x+h) &= \mathbf{y}(x) + \mathbf{K}_2. \end{aligned} \quad (6.9)$$



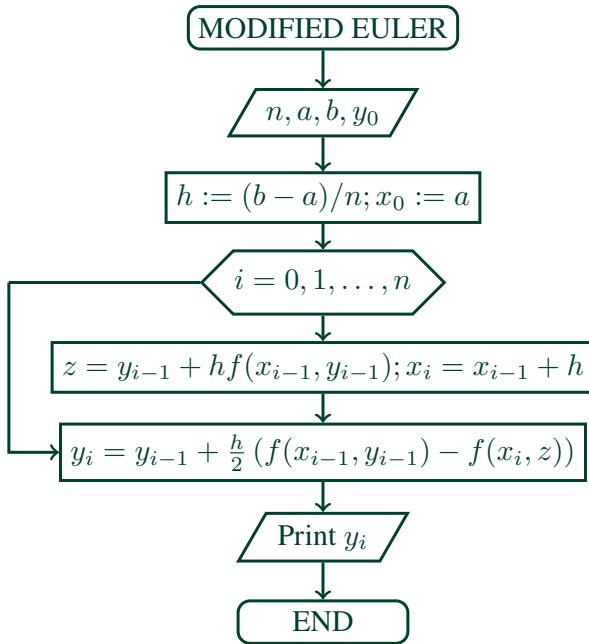
Hình 6.4: Đồ thị minh họa quá trình xấp xỉ nghiệm của phương trình vi phân bằng công thức Euler cải tiến

Hình 6.4 biểu diễn hình học cho công thức Euler cải tiến khi xấp xỉ nghiệm của phương trình vi phân cấp một $y' = f(x, y)$. Bước đầu của (6.9) thu được một ước lượng của y tại điểm giữa của khoảng bằng công thức Euler: $y(x+h/2) = h(x) + f(x, y)h/2 = y(x) + K_1/2$. Sau đó, phương trình thứ hai xấp xỉ diện tích hình thang cong bởi diện tích hình chữ nhật đã tô màu. Sai số của phương pháp tỉ lệ với y'' .

Quá trình áp dụng công thức Euler cải tiến xấp xỉ nghiệm cho phương trình vi phân trên đoạn $[a, b]$ được minh họa bằng sơ đồ khôi trong Hình 6.5

VÍ DỤ 6.3.1. Sử dụng công thức Runge-Kutta bậc-hai tính

$$y' = \sin y, \quad y(0) = 1$$



**Hình 6.5: Sơ đồ khối
giải phương trình vi
phân bằng phương
pháp Euler cải tiến**

từ $x = 0$ đến 5 với các bước là $h = 0.1$. Thực hiện tính toán với bốn chữ số thập phân.

GIẢI. Trong bài toán này ta có

$$f(x, y) = \sin y$$

nên công thức tích phân trong (6.9) là

$$K_1 = hf(x, y) = 0.1 \sin y$$

$$K_2 = hf\left(x + \frac{h}{2}, y + \frac{1}{2}K_1\right) = 0.1 \sin\left(y + \frac{1}{2}K_1\right)$$

$$y(x + h) = y(x) + K_2.$$

Chú ý rằng $y(0) = 1$ nên chúng ta thực hiện quá trình tính liên tiếp như sau

$$K_1 = 0.1 \sin 1.0000 = 0.0841$$

$$K_2 = 0.1 \sin\left(1.0000 + \frac{0.0841}{2}\right) = 0.0863$$

$$y(0.1) = 1.0000 + 0.0863 = 1.0863.$$

$$K_1 = 0.1 \sin 1.0863 = 0.0885$$

$$K_2 = 0.1 \sin\left(1.0863 + \frac{0.0885}{2}\right) = 0.0905$$

$$y(0.2) = 1.0863 + 0.0905 = 1.0905.$$

Tiếp tục thực hiện tương tự, kết quả của quá trình tính toán được tóm tắt trong bảng sau.

x	y	K_1	K_2
0.0	1.0000	0.0841	0.0863
0.1	1.0863	0.0885	0.0905
0.2	1.1768	0.0923	0.0940
0.3	1.2708	0.0955	0.0968
0.4	1.3676	0.0979	0.0988
0.5	1.4664		

Vì nghiệm chính xác có thể được chỉ ra là

$$x(y) = \ln \tan \frac{y}{2} + 0.604582$$

nên $x(1.4664) = 0.5000$. Do đó, tại điểm này thì nghiệm số chính xác với bốn chữ số thập phân. Tuy nhiên, không chắc là độ chính xác này sẽ được duy trì nếu chúng ta tiếp tục tích phân. Do sai số (do chặt và làm tròn) có xu hướng tích lũy, nên khoảng tích phân dài hơn đòi hỏi các công thức tích phân tốt hơn và các số liệu có ý nghĩa hơn trong các tính toán.

□

Các phương pháp Runge-Kutta bậc hai hiếm khi được sử dụng trong tính toán. Các lập trình viên thích sử dụng các công thức Runge-Kutta bậc bốn hơn vì các công thức bậc bốn được thiết kế để đạt độ chính xác cho trước với số lượng tính toán ít. Mục sau đây sẽ giới thiệu một trong các công thức bậc bốn này.

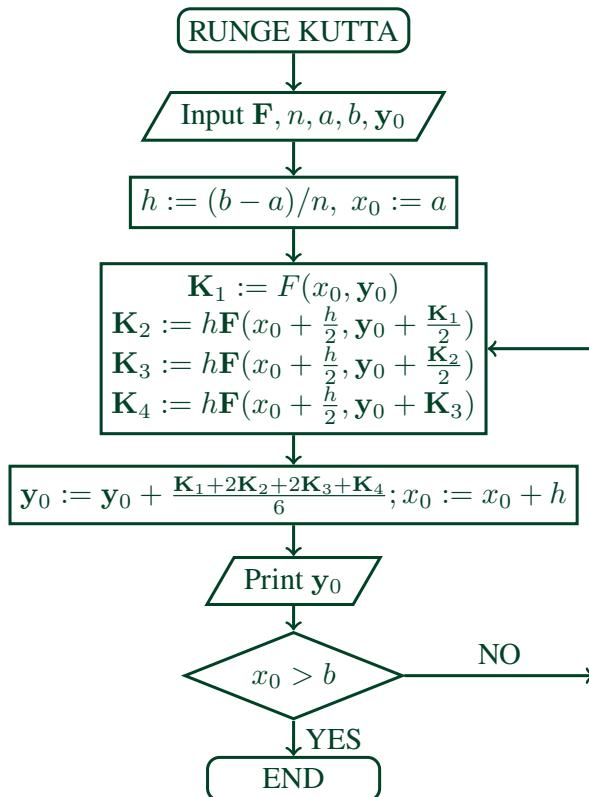
6.3.3 PHƯƠNG PHÁP RUNGE-KUTTA BẬC-BỐN

Phương pháp Runge-Kutta bậc-bốn thu được từ chuỗi Taylor tương tự như phương pháp Runge-Kutta bậc-hai. Quá trình tính toán để xây dựng công thức bậc bốn dài và mang nặng kiến thức toán học nên ta không giới thiệu trong cuốn sách này. Dạng cuối cùng của công thức tích phân Runge-Kutta bậc-bốn cũng phụ thuộc vào việc chọn các tham số nên không có duy nhất một công thức Runge-Kutta bậc-bốn. Phiên bản phổ biến và đơn giản nhất, được gọi là *phương pháp Runge-Kutta bậc-bốn*, được biểu diễn

dưới dạng dãy như sau

$$\begin{aligned}
 \mathbf{K}_1 &= h\mathbf{F}(x, \mathbf{y}) \\
 \mathbf{K}_2 &= h\mathbf{F}\left(x + \frac{h}{2}, \mathbf{y} + \frac{\mathbf{K}_1}{2}\right) \\
 \mathbf{K}_3 &= h\mathbf{F}\left(x + \frac{h}{2}, \mathbf{y} + \frac{\mathbf{K}_2}{2}\right) \\
 \mathbf{K}_4 &= h\mathbf{F}(x + h, \mathbf{y} + \mathbf{K}_3) \\
 \mathbf{y}(x + h) &= \mathbf{y}(x) + \frac{1}{6}(\mathbf{K}_1 + 2\mathbf{K}_2 + 2\mathbf{K}_3 + \mathbf{K}_4).
 \end{aligned} \tag{6.10}$$

Để xấp xỉ nghiệm của phương trình $y' = f(x, y)$, $y(a) = y_0$ trên đoạn $[a, b]$ bằng công thức (6.10), trước tiên ta chia đoạn $[a, b]$ thành các đoạn nhỏ có độ dài bằng $h = (b - a)/n$. Sau đó, thực hiện lặp công thức (6.10) tại mỗi đoạn này. Quá trình tính toán này được minh họa bằng sơ đồ khối trong Hình 6.6



Hình 6.6: Sơ đồ khối giải phương trình vi phân bằng phương pháp Runge - Kutta bậc-bốn

runKut4

Hàm runKut4 thực hiện phương pháp Runge-Kutta bậc-bốn. Hàm đEqs để xác định hệ phương trình vi phân bậc một $y' = F(x, y)$ được nhập từ người sử dụng.

```

function [xSol,ySol] = runKut4(dEqs,x,y,xStop,h)
% 4th-order Runge--Kutta integration.
  
```

```

% USAGE: [xSol,ySol] = runKut4(dEqs,x,y,xStop,h)
% INPUT:
% dEqs = handle of function that specifies the
% 1st-order differential equations
% F(x,y) = [dy1/dx dy2/dx dy3/dx ...].
% x,y = initial values; y must be row vector.
% xStop = terminal value of x.
% h = increment of x used in integration.
% OUTPUT:
% xSol = x-values at which solution is computed.
% ySol = values of y corresponding to the x-values.
if size(y,1)>1;y=y';end %ymustberowvector
xSol = zeros(2,1); ySol = zeros(2,length(y));
xSol(1) = x; ySol(1,:) = y;
i=1;
while x<xStop
    i=i+1;
    h = min(h,xStop - x);
    K1 = h*feval(dEqs,x,y);
    K2 = h*feval(dEqs,x + h/2,y + K1/2);
    K3 = h*feval(dEqs,x + h/2,y + K2/2);
    K4 = h*feval(dEqs,x+h,y + K3);
    y=y+(K1+2*K2+2*K3+K4)/6;
    x=x+h;
    xSol(i) = x; ySol(i,:) = y; % Store current soln.
end

```

VÍ DỤ 6.3.2. Giải

$$y'' = -0.1y' - x, \quad y(0) = 0, y'(0) = 1$$

từ $x = 0$ tới 2 với $h = 0.25$ bằng phương pháp Runge-Kutta bậc-bốn. (Bài toán này đã được giải bằng phương pháp chuỗi Taylor trong VÍ DỤ 6.2.2).

GIẢI. Đặt $y_1 = y, y_2 = y'$, ta viết phương trình đã cho dưới dạng hệ phương trình vi phân bậc một

$$\mathbf{F}(x, \mathbf{y}) = \mathbf{y}' = \begin{bmatrix} y'_1 \\ y'_2 \end{bmatrix} = \begin{bmatrix} y_2 \\ -0.1y_2 - x \end{bmatrix}$$

và được lập trình bằng chương trình sau

```

function F=fex6_3_2(x,y)
    % Differential. eqs. used in Example 6.3.2
    F = zeros(1,2);
    F(1) = y(2); F(2) = -0.1*y(2) - x;

```

So sánh hàm `fex6_3_2` ở ví dụ này với hàm `fex6_2_2` trong VÍ DỤ 6.2.2 ta thấy phương pháp Runge-Kutta đơn giản hơn rất nhiều vì không phải nhập mảng các đạo hàm. Thực hiện các lệnh sau và ta có kết quả

```

>> [x,y] = runKut4(@fex6_3_2,0,[0 1],2,0.25);
>> printSol(x,y,1)
      x          y1          y2
0.0000e+000 0.0000e+000 1.0000e+000
2.5000e-001 2.4431e-001 9.4432e-001
5.0000e-001 4.6713e-001 8.2829e-001
7.5000e-001 6.5355e-001 6.5339e-001
1.0000e+000 7.8904e-001 4.2110e-001
1.2500e+000 8.5943e-001 1.3281e-001
1.5000e+000 8.5090e-001 -2.1009e-001
1.7500e+000 7.4995e-001 -6.0625e-001
2.0000e+000 5.4345e-001 -1.0543e+000

```

Các kết quả này giống như kết quả thu được bằng phương pháp chuỗi Taylor trong VÍ DỤ 6.2.2. Điều này đã được dự báo vì cả hai phương pháp đều có cùng độ chính xác.

□

VÍ DỤ 6.3.3. Sử dụng phương pháp Runge-Kutta bốn bước tính

$$y' = 3y - 4e^{-x}, \quad y(0) = 1.$$

từ $x = 0$ đến 10 với $h = 0.1$. So sánh kết quả này với nghiệm giải tích $y = e^{-x}$.

GIẢI. Hàm số xác định phương trình vi phân là

```

function F=fex6_3_3(x,y)
    % Differential eq. used in Example 6.3.3.
    F=3*y-4*exp(-x);

```

Nghiệm là (tần số dòng in là 20)

```

>> [x,y] = runKut4(@fex6_3_3,0,1,10,0.1);
>> printSol(x,y,20)
    x          y1
0.0000e+000 1.0000e+000
2.0000e+000 1.3250e-001
4.0000e+000 -1.1237e+000
6.0000e+000 -4.6056e+002
8.0000e+000 -1.8575e+005
1.0000e+001 -7.4912e+007

```

Điều này chỉ ra có phần nào đó bị sai bởi vì theo nghiệm giải tích thì y giảm dần về không khi x tăng nhưng kết quả số lại chỉ ra điều ngược lại: sau giá trị ban đầu giản, độ lớn của y tăng đột ngột. Lời giải thích được tìm thấy bằng cách nghiên cứu kỹ hơn nghiệm giải tích của bài toán. Nghiệm tổng quát của phương trình vi phân đã cho là

$$y = Ce^{3x} + e^{-x}.$$

Vì điều kiện ban đầu $y(0) = 1$ nên $C = 0$. Do đó, nghiệm của bài toán là $y = e^{-x}$.

Nguyên nhân của vấn đề trong nghiệm số nằm ở biểu thức Ce^{3x} . Giả sử rằng điều kiện ban đầu có chứa một sai số ε nhỏ, do đó ta có $y(0) = 1 + \varepsilon$. Điều này biến đổi nghiệm giải tích thành

$$y = \varepsilon e^{3x} + e^{-x}.$$

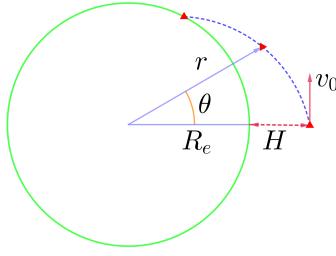
Bây giờ chúng ta thấy rằng biểu thức có chứa ε trở thành trội khi x tăng. Do các sai số vốn có trong nghiệm số có tác động tương tự như những thay đổi nhỏ trong điều kiện ban đầu, nên chúng ta có thể kết luận rằng nghiệm số của chúng ta là nạn nhân của sự mất ổn định số do độ nhạy của nghiệm với điều kiện ban đầu. Bài học ở đây là: không được luôn tin tưởng vào sự đúng đắn của nghiệm số.

□

VÍ DỤ 6.3.4. Một tàu vũ trụ được phóng ở độ cao $H = 772$ km so với mực nước biển với tốc độ $v_0 = 6700$ m/s theo hướng hiển thị. Các phương trình vi phân mô tả chuyển động của tàu vũ trụ là

$$\ddot{r} = r\dot{\theta}^2 - \frac{GM_e}{r^2}, \quad \ddot{\theta} = -\frac{2\dot{r}\dot{\theta}}{r},$$

với r và θ là các toạ độ cực của tàu vũ trụ. Các hằng số trong phương trình chuyển động là



$$G = 6.672 \times 10^{-11} m^3 kg^{-1} s^{-2} = \text{hằng số hấp dẫn vũ trụ}$$

$$M_e = 5.9742 \times 10^{24} kg = \text{trọng lượng trái đất}$$

$$R_e = 6378.14 km = \text{bán kính của trái đất ở mực nước biển}$$

(1) Xác định hệ phương trình vi phân bậc một và điều kiện ban đầu dạng $\dot{\mathbf{y}} = \mathbf{F}(x, \mathbf{y})$, $\mathbf{y}(0) = \mathbf{b}$. (2) Sử dụng phương pháp Runge-Kutta bậc-bốn để tính tích phân hệ phương trình từ lúc phóng đến lúc tàu vũ trụ chạm đất. Xác định θ tại vị trí va chạm.

GIẢI. Phần (1). Ta có

$$GM_e = (6.672 \times 10^{-11}) (5.99742 \times 10^{24}) = 3.9860 \times 10^{14} m^2 s^{-2}.$$

Đặt

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} r \\ \dot{r} \\ \theta \\ \dot{\theta} \end{bmatrix}$$

hệ phương trình vi phân bậc một tương đương là

$$\dot{\mathbf{y}} = \begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \\ \dot{y}_3 \\ \dot{y}_4 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_0 y_3^2 - 3.9860 \times 10^{14} / y_0^2 \\ y_3 \\ -2y_1 y_3 / y_0 \end{bmatrix}$$

với các điều kiện ban đầu là

$$r(0) = R_e + H = R_e = (6378.14 + 772) \times 10^3 = 7.15014 \times 10^6 m$$

$$\dot{r}(0) = 0$$

$$\theta(0) = 0$$

$$\dot{\theta}(0) = v_0 / r(0) = (6700) / (7.15014 \times 10^6) = 0.937045 \times 10^{-3} rad/s.$$

Do đó

$$\mathbf{y}(0) = \begin{bmatrix} 7.15014 \times 10^6 \\ 0 \\ 0 \\ 0.937045 \times 10^{-3} \end{bmatrix}.$$

Phần (2). Hàm số trả lại phương trình vi phân là

```
function F=fex6_3_4(x,y)
% Differential eqs. used in Example 6.3.4.
F = zeros(1,4);
F(1) = y(2);
F(2) = y(1)*y(4)^2 - 3.9860e14/y(1)^2;
F(3) = y(4);
F(4) = -2*y(2)*y(4)/y(1);
```

Sử dụng chương trình dùng cho giải số được cho sau đây ta được kết quả. Chú ý rằng biến độc lập t được ký hiệu bằng x .

```
>> % Example 6.3.4 (Runge-Kutta integration)
x=0;y=[7.15014e6 0 0 0.937045e-3];
xStop = 1200;h=50;freq=2;
[xSol,ySol] = runKut4(@fex6_3_4,x,y,xStop,h);
printSol(xSol,ySol,freq)
x y1 y2 y3 y4
0.0000e+000 7.1501e+006 0.0000e+000 0.0000e+000 9.3704e-004
1.0000e+002 7.1426e+006 -1.5173e+002 9.3771e-002 9.3904e-004
2.0000e+002 7.1198e+006 -3.0276e+002 1.8794e-001 9.4504e-004
3.0000e+002 7.0820e+006 -4.5236e+002 2.8292e-001 9.5515e-004
4.0000e+002 7.0294e+006 -5.9973e+002 3.7911e-001 9.6951e-004
5.0000e+002 6.9622e+006 -7.4393e+002 4.7697e-001 9.8832e-004
6.0000e+002 6.8808e+006 -8.8389e+002 5.7693e-001 1.0118e-003
7.0000e+002 6.7856e+006 -1.0183e+003 6.7950e-001 1.0404e-003
8.0000e+002 6.6773e+006 -1.1456e+003 7.8520e-001 1.0744e-003
9.0000e+002 6.5568e+006 -1.2639e+003 8.9459e-001 1.1143e-003
1.0000e+003 6.4250e+006 -1.3708e+003 1.0083e+000 1.1605e-003
1.1000e+003 6.2831e+006 -1.4634e+003 1.1269e+000 1.2135e-003
1.2000e+003 6.1329e+006 -1.5384e+003 1.2512e+000 1.2737e-003
```

Tàu vũ trụ chạm đất khi r bằng $R_e = 6.37814 \times 10^6$ m. Điều này xảy ra trong khoảng 1000 đến 1100 s. Giá trị chính xác hơn của t có thể đạt được bằng đa thức nội suy. Đặt thời điểm va chạm là $1000 + \Delta t$, ta có thể viết

$$r(1000 + \Delta t) = R_e.$$

Khai triển r bằng một chuỗi Taylor hai bước ta có

$$\begin{aligned} r(1000) + \dot{r}(1000)\Delta t &= R_e \\ 6.4250 \times 10^6 + (-1.3708 \times 10^3)\Delta t &= 6378.14 \times 10^3 \end{aligned}$$

suy ra

$$\Delta t = 34.184 \text{ s.}$$

Do đó, thời điểm va chạm là 1034.2 s.

Toạ độ θ của vị trí va chạm có thể ước lượng bằng cách tương tự. Tiếp tục sử dụng hai biểu thức trong chuỗi Taylor ta có

$$\begin{aligned} \theta(1000 + \Delta t) &= \theta(1000) + \dot{\theta}(1000)\Delta t \\ &= 1.0083 + (1.1605 \times 10^{-3})(34.184) \\ &= 1.0480 \text{ rad} = 60.0^\circ. \end{aligned}$$

□

Hạn chế chính của phương pháp này là khi h cố định thì nó có thể không đáp ứng được tính chính xác mong muốn vì những thay đổi của hàm $f(x, y)$ trên miền này. Do đó, chúng ta phải đoán kích thước bước tích phân h hoặc xác định nó bằng thử - sai. Ngược lại, các *phương pháp thích ứng* được cho là có thể đánh giá sai số trong mỗi bước tích phân và điều chỉnh giá trị của h cho phù hợp (nhưng với chi phí tính toán cao hơn). Một phương pháp thích ứng như vậy sẽ được giới thiệu trong Mục 6.5.

6.4 TÍNH ỔN ĐỊNH VÀ TÍNH CƯỜNG

Một cách không chặt chẽ, một phương pháp tích phân số được cho là ổn định nếu sai số gây ra bởi quá trình tích luỹ các sai số cục bộ không bùng nổ; tức là sai số toàn cục vẫn bị chặn. Nếu một phương pháp không ổn định, sai số toàn cục sẽ tăng theo cấp số nhân, cuối cùng gây ra tràn số. Sự ổn định không có tác dụng gì với độ chính xác; Trên thực tế, một phương pháp không chính xác có thể rất ổn định.

Tính ổn định được xác định bởi ba yếu tố: phương trình vi phân, phương pháp giải và giá trị của số gia h . Thật không may là không dễ xác định trước tính ổn định của một phương trình vi phân trừ trường hợp đây là phương trình vi phân tuyến tính.

6.4.1 TÍNH ỔN ĐỊNH CỦA PHƯƠNG PHÁP EULER

Để minh họa cho tính ổn định khi giải số một phương trình vi phân, ta xét bài toán

$$y' = -\lambda y, \quad y(0) = \beta \quad (6.11)$$

với λ là một hằng số dương. Nghiệm chính xác của bài toán là

$$y(x) = \beta e^{-\lambda x}.$$

Bây giờ chúng ta sẽ nghiên cứu điều gì sẽ xảy ra khi ta cố gắng giải số (6.11) bằng phương pháp Euler

$$y(x+h) = y(x) + hy'(x). \quad (6.12)$$

Thay $y'(x) = -\lambda y(x)$ ta có

$$y(x+h) = (1 - \lambda h)y(x).$$

Nếu $|1 - \lambda h| > 1$ thì phương pháp không ổn định vì $|y|$ tăng trong mọi bước tích phân. Do đó, phương pháp Euler là ổn định nếu $|1 - \lambda h| \leq 1$, hoặc

$$h \leq 2/\lambda. \quad (6.13)$$

Kết quả này có thể mở rộng cho hệ n phương trình vi phân dưới dạng

$$\mathbf{y}' = -\Lambda \mathbf{y} \quad (6.14)$$

với Λ là ma trận hằng số với các giá trị riêng dương $\lambda_i, i = 1, 2, \dots, n$. Ta có thể chỉ ra rằng phương pháp Euler ổn cho công thức tích phân là ổn định nếu chỉ khi

$$h < 2/\lambda_{max} \quad (6.15)$$

với λ_{max} là giá trị riêng lớn nhất của Λ .

6.4.2 TÍNH CƯƠNG CỦA BÀI TOÁN

Một bài toán giá trị ban đầu được gọi là *cương* nếu một số biểu thức trong vecto nghiệm $\mathbf{y}(x)$ thay đổi theo biến x nhanh hơn nhiều so với một số biểu thức khác. Tính cương có thể dễ dàng dự đoán cho hệ phương trình vi phân $\mathbf{y}' = -\Lambda \mathbf{y}$ với ma trận hằng số Λ . Nghiệm của hệ phương trình này là $\mathbf{y}(x) = \sum_i C_i \mathbf{v}_i \exp(-\lambda_i x)$, với λ_i là các giá trị riêng của Λ và \mathbf{v}_i là vecto riêng tương ứng. Rõ ràng bài toán là cương nếu có sự chênh lệch lớn về độ lớn của các giá trị riêng.

Tính tích phân số của phương trình cương yêu cầu phương pháp đặc biệt. Kích thước mỗi bước h cần cho tính ổn định được xác định bằng giá trị riêng lớn nhất λ_{max} , thậm chí nếu biểu thức $\exp(-\lambda_{max}x)$ trong nghiệm phân rã rất nhanh và trở nên không có ý nghĩa khi chúng ta di chuyển xa gốc toạ độ.

Lấy ví dụ, xét phương trình vi phân

$$y'' + 1001y' + 1000y = 0. \quad (6.16)$$

Đặt $y_1 = y, y_2 = y'$, hệ phương trình bậc một tương đương là

$$\mathbf{y}' = \begin{bmatrix} y_2 \\ -1000y_1 - 1001y_2 \end{bmatrix}.$$

Trong trường hợp này

$$\Lambda = \begin{bmatrix} 0 & -1 \\ 1000 & 1001 \end{bmatrix}.$$

Các giá trị riêng của Λ là nghiệm của

$$|\Lambda - \lambda I| = \begin{vmatrix} -\lambda & -1 \\ 1000 & 1001 - \lambda \end{vmatrix} = 0.$$

Khai triển định thức ta có

$$-\lambda(1001 - \lambda) + 1000 = 0$$

với các nghiệm $\lambda_1 = 1, \lambda_2 = 1000$. Do đó, hệ phương trình đã cho là cương. Theo (6.15) thì cần $h < 2/\lambda_2 = 0.002$ để phương pháp Euler ổn định. Phương pháp Runge-Kutta có khoảng giới hạn trên các bước tương tự.

Khi bài toán là rất cương, các phương pháp thông thường của nghiệm, chẳng hạn như công thức Runge-Kutta, trở thành không thực tế bởi vì kích thước mỗi bước h

phải rất nhỏ để đảm bảo yêu cầu của tính ổn định. Những bài toán này được giải quyết tốt nhất bằng những phương pháp được thiết kế đặc biệt dành riêng cho các phương trình cương. Việc giải quyết các bài toán cương nằm ngoài phạm vi của quyển sách vì nó yêu cầu nhiều tính chất phức tạp khác của tính ổn định. Tuy nhiên, việc tăng bậc của tính ổn định cũng phải trả giá theo nguyên tắc là: muốn cải thiện tính ổn định thì phải giảm bậc của tích phân và do đó sẽ làm tăng sai số phương pháp.

VÍ DỤ 6.4.1. (1) chỉ ra rằng bài toán

$$y'' = -\frac{19}{4}y - 10y', \quad y(0) = -9, \quad y'(0) = 0$$

là cương vừa phải và ước lượng h_{max} , giá trị lớn nhất của h để phương pháp Runge-Kutta là ổn định. (2) Xác định lại ước lượng bằng cách tính $y(10)$ với $h \approx h_{max}/2$ và $h \approx 2h_{max}$.

GIẢI. **Phần (1).** Đặt $y_1 = y, y_2 = y'$, hệ phương trình vi phân bậc một tương đương là

$$\mathbf{y}' = \begin{bmatrix} y_2 \\ -\frac{19}{4}y_1 - 10y_2 \end{bmatrix} = -\boldsymbol{\Lambda} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix},$$

với

$$\boldsymbol{\Lambda} = \begin{bmatrix} 0 & -1 \\ \frac{19}{4} & 10 \end{bmatrix}.$$

Các giá trị riêng của $\boldsymbol{\Lambda}$ được cho bởi

$$|\boldsymbol{\Lambda} - \lambda \mathbf{I}| = \begin{vmatrix} -\lambda & -1 \\ \frac{19}{4} & 10 - \lambda \end{vmatrix}$$

nên thu được $\lambda_1 = 1/2$ và $\lambda_2 = 19/2$. Bởi vì λ_2 lớn hơn λ_1 một chút nên bài toán là cương vừa phải.

Để phương pháp là ổn định thì từ (6.15) ta có

$$h_{max} = \frac{2}{\lambda_{max}} = \frac{2}{19/2} = 0.2153.$$

Phần (2). Sử dụng phương pháp Runge-Kutta với $h = 0.1$ để giải bài toán, ta được kết quả sau (ta sử dụng freq = 0 trong printSol để in các giá trị đầu và giá trị cuối)

```
>> x           y1           y2
0.0000e+000 -9.0000e+000 0.0000e+000
1.0000e+001 -6.4011e-002 3.2005e-002
```

Nghiệm giải tích của hệ là

$$y(x) = -\frac{19}{2}e^{-x/2} + \frac{1}{2}e^{-19x/2}$$

ta có $y(10) = -0.064011$ nên ta đồng ý với kết quả số thu được. Với $h = 0.5$ thì phương pháp không ổn định nên ta có kết quả như dự đoán là

```
>> x y1 y2
0.0000e+000 -9.0000e+000 0.0000e+000
1.0000e+001 2.7030e+020 -2.5678e+021
```

□

6.5 PHƯƠNG PHÁP RUNGE-KUTTA THÍCH ỨNG

Việc xác định chiều dài của mỗi bước h thích hợp có thể trở thành một chủ đề đau đầu trong tính số tích phân. Nếu h quá lớn thì có thể sai số chặt không thể chấp nhận; nếu h quá nhỏ thì ta lãng phí tài nguyên tính toán. Hơn nữa, chiều dài mỗi bước cố định có thể không phù hợp với toàn bộ khoảng lấy tích phân. Chẳng hạn, nếu đường cong nghiệm bắt đầu với sự thay đổi lớn rồi sau đó trơn ổn định thì ta cần bắt đầu bằng h rất nhỏ rồi tăng dần lên trên miền trơn. Những phương pháp như vậy được gọi là *phương pháp thích ứng*. Những phương pháp này ước lượng sai số chặt tại mỗi bước tích phân rồi sau đó điều chỉnh chiều dài của bước để sai số trong giới hạn cho phép.

Các phương pháp Runge-Kutta thích hợp còn được gọi là *các công thức tích phân nhúng*. Các công thức này gồm một cặp: một công thức có bậc tích phân là m và công thức còn lại có bậc tích phân là $m + 1$. Ý tưởng là sử dụng cả hai công thức để tìm nghiệm từ x đến $x + h$. Ký hiệu kết quả là $\mathbf{y}_m(x + h)$ và $\mathbf{y}_{m+1}(x + h)$ chúng ta có thể ước lượng sai số chặt trong công thức bậc m là

$$\mathbf{E}(h) = \mathbf{y}_{m+1}(x + h) - \mathbf{y}_m(x + h). \quad (6.17)$$

Điều mà tạo nên tính hấp dẫn của các công thức nhúng là chúng chia sẻ những điểm mà $\mathbf{F}(x, \mathbf{y})$ được ước lượng. Điều này có nghĩa là một khi $\mathbf{y}_m(x + h)$ đã được tính thì một sự nỗ lực bổ sung tương đối nhỏ được yêu cầu để tính $\mathbf{y}_{m+1}(x + h)$.

Các công thức nhúng Runge-Kutta trình bày trong cuốn sách này có bậc năm và bậc bốn, các công thức này ban đầu được đưa ra bởi Fehlberg nên còn được gọi là *các công thức Runge-Kutta-Fehlberg*:

$$\mathbf{K}_1 = h\mathbf{F}(x, yy) \quad (6.18)$$

$$\mathbf{K}_i = h\mathbf{F} \left(x + A_i h, \mathbf{y} + \sum_{j=1}^{i-2} B_{ij} \mathbf{K}_j \right), i = 2, 3, \dots, 6 \quad (6.19)$$

$$\mathbf{y}_5(x+h) = \mathbf{y}(x) + \sum_{i=1}^6 C_i \mathbf{K}_i \quad \text{công thức bậc 5} \quad (6.19a)$$

$$\mathbf{y}_4(x+h) = \mathbf{y}(x) + \sum_{i=1}^6 D_i \mathbf{K}_i \quad \text{công thức bậc 4} \quad (6.19b)$$

Các hệ số xuất hiện trong các công thức trên là không duy nhất. Bảng cho dưới đây là các hệ số được đề xuất bởi Cash và Karp¹, những người đã cải tiến từ các hệ số gốc của Fehlberg.

i	A_i	B_{ij}					C_i	D_i
1	—	—	—	—	—	—	$\frac{37}{378}$	$\frac{2825}{27648}$
2	$\frac{1}{5}$	$\frac{1}{5}$	—	—	—	—	0	0
3	$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$	—	—	—	$\frac{250}{621}$	$\frac{18575}{48384}$
4	$\frac{3}{5}$	$\frac{3}{10}$	$-\frac{9}{10}$	$\frac{6}{5}$	—	—	$\frac{125}{594}$	$\frac{13525}{55296}$
5	1	$-\frac{11}{54}$	$\frac{5}{2}$	$-\frac{770}{27}$	$\frac{35}{27}$	—	0	$\frac{277}{14336}$
6	$\frac{7}{8}$	$\frac{1631}{55296}$	$\frac{175}{512}$	$\frac{575}{13824}$	$\frac{44275}{110592}$	$\frac{253}{4096}$	$\frac{512}{1771}$	$\frac{1}{4}$

Bảng 6.1: Các hệ số Cash-Karp của các công thức Runge-Kutta-Fehlberg

Lời giải được nâng cao với công thức bậc năm trong (6.19a). Công thức bậc bốn chỉ được sử dụng ẩn trong việc ước lượng sai số chật

$$\mathbf{E}(h) = \mathbf{y}_5(x+h) - \mathbf{y}_4(x+h) = \sum_{i=1}^6 (C_i - D_i) \mathbf{K}_i. \quad (6.20)$$

Vì (6.20) thực sự áp dụng cho công thức bậc bốn, nó có xu hướng đánh giá cao sai số trong công thức bậc năm.

Chú ý rằng $\mathbf{E}(h)$ là một vectơ, thành phần $E_i(h)$ của nó biểu diễn sai số cho biến phụ thuộc y_i . Điều này dẫn đến câu hỏi: Cái gì đo sai số $e(h)$ mà chúng ta mong điều khiển? Không có một lựa chọn riêng lẻ nào tốt cho mọi bài toán. Nếu chúng ta muốn

¹J.R. Cash and A.H. Carp, ACM Transactions on Mathematical Software 16, 201–222 (1990).

điều khiển nhiều nhất các thành phần của $E(h)$ thì độ đo sai số có thể là

$$e(h) = \max_i |E_i(h)|. \quad (6.21)$$

Chúng ta cũng có thể điều khiển tổng độ đo của các sai số, chẳng hạn như căn bậc hai của trung bình bình phương sai số được xác định bởi:

$$\bar{E}(h) = \sqrt{\frac{1}{n} \sum_{i=1}^n E_i^2(h)} \quad (6.22)$$

với n là số lượng phương trình vi phân bậc một. Khi đó, ta sử dụng

$$e(h) = \bar{E}(h) \quad (6.23)$$

cho độ đo sai số. Vì căn bậc hai của trung bình bình phương sai số có thể dễ dàng tính bằng tay nên ta bỏ qua trong chương trình của chúng ta.

Việc điều khiển sai số được thiết kế bằng cách điều chỉnh số gia h sao cho sai số ở bước tiếp theo e xấp xỉ bằng độ chính xác cho phép ε . Chú ý rằng sai số chật trong công thức bậc bốn là $\mathcal{O}(h^5)$ nên ta có thể suy ra

$$\frac{e(h_1)}{e(h_2)} \approx \left(\frac{h_1}{h_2} \right)^5. \quad (a)$$

Chúng ta giả sử rằng đã thực hiện một tích phân với bậc h_1 và sai số tương ứng là $e(h_1)$. Kích thước bước h_2 mà chúng ta có thể sử dụng thu được từ phương trình (a) bằng cách đặt $e(h_2) = \varepsilon$:

$$h_2 = h_1 \left[\frac{\varepsilon}{e(h_1)} \right]^{1/5}. \quad (b)$$

Nếu $h_2 \geq h_1$ thì chúng ta có thể lặp với bậc tích phân h_2 nhưng vì sai số tương ứng với h_1 thấp hơn cho phép nhưng đó có thể là một sự lãng phí của một kết quả tốt. Vậy nên chúng ta chấp nhận bước hiện tại h_1 và cố gắng với h_2 ở bước tiếp theo. Mặt khác, nếu $h_2 < h_1$ thì sai số $e(h_1)$ cao hơn cho phép nên ta loại bỏ bước hiện tại và lặp lại với h_2 . Vì (b) chỉ là một công thức xấp xỉ nên để chắc hơn ta nên kết hợp với một biên độ an toàn nhỏ. Trong chương trình của chúng ta sẽ sử dụng công thức

$$h_2 = 0.9h_1 \left[\frac{\varepsilon}{e(h_1)} \right]^{1/5}. \quad (6.24)$$

Nhắc lại rằng $e(h)$ áp dụng cho một bước tích phân đơn nên nó chỉ là độ đo của sai số chật địa phương. Sau tất cả, sai số toàn cục quan trọng là so sánh luỹ của các

sai số địa phương. Nên đặt ε ở mức nào để đạt được sai số toàn cục không lớn hơn ε_{global} ? Vì $e(h)$ là một ước lượng của sai số thực nên ta luôn có thể đặt $\varepsilon = \varepsilon_{global}$. Nếu số lượng các bước tích phân lớn thì ta nên giảm ε cho phù hợp.

Sau cùng, có lý do nào để sử dụng phương pháp không thích nghi không? Câu trả lời thường là không, tuy nhiên, có một vài trường hợp đặc biệt mà phương pháp thích nghi bị phá vỡ. Chẳng hạn, phương pháp thích nghi tổng quát không thực hiện được khi $F(x, y)$ chứa hàm số không liên tục. Bởi vì sai số có tính không ổn định tại những điểm không liên tục nên chương trình có thể bị kẹt trong vòng lặp vô hạn khi tìm giá trị của h . Chúng ta cũng sẽ sử dụng các phương pháp không thích nghi nếu đầu ra có chứa các giá trị cách đều của x .

runKut5

Phương pháp Runge-Kutta thích nghi được thực hiện bởi hàm `runKut5` sau đây. Tham số đầu vào h là giá trị được sử dụng làm số gia cho bước tích phân thứ nhất.

```

function [xSol,ySol] = runKut5(dEqs,x,y,xStop,h,eTol)
% 5th-order Runge-Kutta integration.
% USAGE: [xSol,ySol] = runKut5(dEqs,x,y,xStop,h,eTol)
% INPUT:
% dEqs = handle of function that specifyies the
% 1st-order differential equations
% F(x,y) = [dy1/dx dy2/dx dy3/dx ...].
% x,y = initial values; y must be row vector.
% xStop = terminal value of x.
% h = trial value of increment of x.
% eTol = per-step error tolerance (default = 1.0e-6).
% OUTPUT:
% xSol = x-values at which solution is computed.
% ySol = values of y corresponding to the x-values.

if size(y,1)>1;y=y';end % y must be row vector
if nargin < 6; eTol = 1.0e-6; end
n = length(y);
A= [ 0 1/5 3/10 3/5 1 7/8];
B= [ 0 0 0 0 0
      1/5 0 0 0 0
      3/40 9/40 0 0 0
      3/10 -9/10 6/5 0 0
      ]
```

```

-11/54 5/2 -70/27 35/27      0
1631/55296 175/512 575/13824 44275/110592 253/4096];
C = [37/378 0 250/621 125/594 0 512/1771];
D = [2825/27648 0 18575/48384 13525/55296 277/14336 1/4];
% Initialize solution
xSol = zeros(2,1); ySol = zeros(2,n);
xSol(1) = x; ySol(1,:) = y;
stopper=0;k=1;
for p=2:5000
    % Compute K's from Eq. (6.18)
    K = zeros(6,n);
    K(1,:) = h*feval(dEqs,x,y);
    for i=2:6
        BK = zeros(1,n);
        for j=1:i-1
            BK = BK + B(i,j)*K(j,:);
        end
        K(i,:)=h*feval(dEqs,x+A(i)*h,y+BK);
    end
    % Compute change in y and per-step error from
    % Eqs.(6.19) & (6.20)
    dy=zeros(1,n);E=zeros(1,n);
    for i=1:6
        dy = dy + C(i)*K(i,:);
        E=E+(C(i)-D(i))*K(i,:);
    end
    e = sqrt(sum(E.*E)/n);
    % If error within tolerance, accept results and
    % check for termination
    if e<=eTol
        y=y+dy;x=x+h;
        k=k+1;
        xSol(k) = x; ySol(k,:) = y;
        if stopper == 1;
            break
        end
    end
    % Size of next integration step from Eq. (6.24)

```

```

if e~= 0; hNext = 0.9*h*(eTol/e)^0.2;
else; hNext=h;
end

% Check if next step is the last one (works
% with positive and negative h)
if(h>0)==(x+hNext>=xStop)
    hNext = xStop - x; stopper = 1;
end

h=hNext;
end

```

VÍ DỤ 6.5.1. Lực kéo khí động lực tác dụng lên một vật thể nào đó rơi tự do có thể được xấp xỉ bằng

$$F_D = av^2 e^{-by}$$

với

$$\begin{aligned} v &= \text{vận tốc của vật thể (m/s)} \\ y &= \text{độ cao của vật thể (m)} \\ a &= 7.45 \text{ kg/m} \\ b &= 10.53 \times 10^{-5} m^{-1}. \end{aligned}$$

Biểu thức mũ giải thích sự thay đổi của mật độ không khí với độ cao. Phương trình vi phân mô tả cú rơi là

$$m\ddot{y} = -mg + F_D$$

với $g = 9.80665 \text{ m/s}^2$ và $m = 114 \text{ kg}$ là khối lượng của vật thể. Nếu vật thể được thả ở độ cao 9 km, hãy xác định độ cao của nó sau khi rơi 10 s bằng phương pháp Runge-Kutta thích ứng.

GIẢI. Phương trình vi phân và các điều kiện ban đầu là

$$\begin{aligned} \ddot{y} &= -g + \frac{a}{m}\dot{y}^2 e^{-by} \\ &= -9.80655 + \frac{7.45}{114}\dot{y}^2 e^{-10.53 \times 10^{-5}y} \end{aligned}$$

$$y(0) = 9000 \text{ m}, \quad \dot{y}(0) = 0.$$

Đặt $y_1 = y, y_2 = \dot{y}$ ta thu được hệ phương trình vi phân bậc một tương đương và các điều kiện ban đầu là

$$\begin{aligned}\dot{\mathbf{y}} &= \begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \end{bmatrix} = \begin{bmatrix} y_2 \\ -980665 + (65.351 \times 10^{-3})y_2^2 e^{-10.53 \times 10^{-5}y_1} \end{bmatrix} \\ \mathbf{y}(0) &= \begin{bmatrix} 9000 \text{ m} \\ 0 \end{bmatrix}.\end{aligned}$$

Hàm mô tả các phương trình vi phân là

```
function F=fex6_5_1(x,y)
    % Diff. eqs. used in Example 6.5.1
    F = zeros(1,2);
    F(1) = y(2);
    F(2) = -9.80665...
        + 65.351e-3 * y(2)^2 * exp(-10.53e-5 * y(1));
```

Các lệnh thực hiện tích phân và mô tả được cho bên dưới. Trong runKut5, chúng ta xác định sai số cho phép trong một bước bằng 10^{-2} . Để xét độ lớn của y , ta giữ năm số thập phân trong nghiệm. Thực hiện các lệnh

```
>> [x,y] = runKut5(@fex6_5_1,0,[9000 0],10,0.5,1.0e-2);
>> printSol(x,y,1)
```

ta có kết quả là

```
>> x           y1           y2
0.0000e+000 9.0000e+003 0.0000e+000
5.0000e-001 8.9988e+003 -4.8043e+000
1.9246e+000 8.9841e+003 -1.4632e+001
3.2080e+000 8.9627e+003 -1.8111e+001
4.5031e+000 8.9384e+003 -1.9195e+001
5.9732e+000 8.9099e+003 -1.9501e+001
7.7786e+000 8.8746e+003 -1.9549e+001
1.0000e+001 8.8312e+003 -1.9519e+001
```

Bước tích phân đầu tiên được thực hiện với giá trị thử nghiệm $h = 0.5\text{s}$. Rõ ràng sai số trong mức chấp nhận nên bước này được chấp nhận. Kích thước các bước kế tiếp được xác định từ (6.24) lớn hơn đáng kể.

□

Quan sát kết quả đầu ra ta thấy rằng tại thời điểm $t = 10s$, vật thể đang di chuyển với vận tốc $v = -\dot{y} = 19.52 \text{ m/s}$ và ở độ cao $y = 8831 \text{ m}$.

VÍ DỤ 6.5.2. Tích phân bài toán cương vừa phải

$$y'' = -\frac{14}{4}y - 10y', \quad y(0) = -9, y'(0) = 0$$

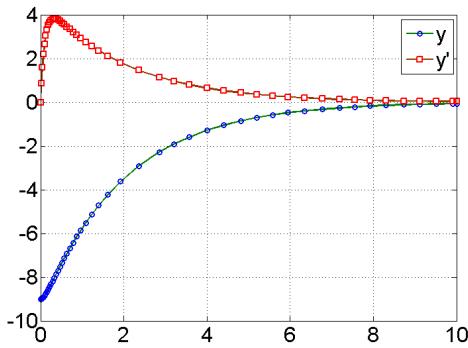
từ $x = 0$ đến 10 bằng phương pháp Runge-Kutta thích nghi và vẽ kết quả. (Bài toán này cũng xuất hiện trong VÍ DỤ 6.4.1).

GIẢI. Vì chúng ta giải bằng một phương pháp tích nghi nên không cần lo lắng về khoảng ổn định của h như trong VÍ DỤ 6.4.1. Miễn là chúng ta xác định mức độ cho phép sai số của mỗi bước hợp lý, thuật toán sẽ tìm các kích thước bước phù hợp cho mỗi bước. Sau đây là các lệnh và kết quả thực hiện

```
>> [x, y] = runKut5(@fex7_7, 0, [-9 0], 10, 0.1);
>> printSol(x, y, 4)
```

x	y1	y2
0.0000e+000	-9.0000e+000	0.0000e+000
9.8941e-002	-8.8461e+000	2.6651e+000
2.1932e-001	-8.4511e+000	3.6653e+000
3.7058e-001	-7.8784e+000	3.8061e+000
5.7229e-001	-7.1338e+000	3.5473e+000
8.6922e-001	-6.1513e+000	3.0745e+000
1.4009e+000	-4.7153e+000	2.3577e+000
2.8558e+000	-2.2783e+000	1.1391e+000
4.3990e+000	-1.0531e+000	5.2656e-001
5.9545e+000	-4.8385e-001	2.4193e-001
7.5596e+000	-2.1685e-001	1.0843e-001
9.1159e+000	-9.9591e-002	4.9794e-002
1.0000e+001	-6.4010e-002	3.2005e-002

Kết quả phù hợp với nghiệm giải tích.



Đồ thị của y' và y chỉ ra mọi bước tích phân thứ tự. Chú ý rằng khi gần $x = 0$ thì y' thay đổi nhanh nên mật độ các điểm cao. Khi đường cong y' trơn hơn thì khoảng cách giữa các điểm tăng.

□

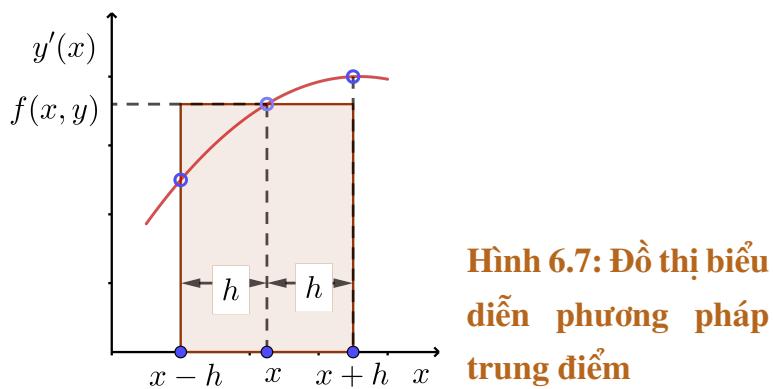
6.6 PHƯƠNG PHÁP BULIRSCH-STOER

6.6.1 PHƯƠNG PHÁP TRUNG ĐIỂM

Công thức trung điểm của tích phân số của $y' = F(x, y)$ là

$$y(x+h) = y(x-h) + 2hF[x, y(x)]. \quad (6.25)$$

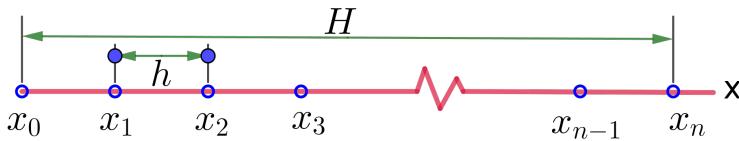
Đây là công thức bậc hai giống như công thức Euler. Chúng ta thảo luận về công thức này vì nó là cơ sở của *phương pháp Bulirsch-Stoer*, là phương pháp mà kỹ thuật chọn trong bài toán đáp ứng được yêu cầu độ chính xác cao.



Hình 6.7 minh họa công thức trung điểm cho một phương trình vi phân đơn lẻ $y' = f(x, y)$. Sự thay đổi của y trên hai ô là

$$y(x+h) - y(x-h) = \int_{x-h}^{x+h} y'(x)dx$$

cũng là diện tích bên dưới đường cong y' . Công thức trung điểm xấp xỉ diện tích hình thang cong này bởi diện tích của hình chữ nhật được tô màu (bằng $2hf(x, y)$).



Hình 6.8: Lưới sử dụng trong phương pháp trung điểm

Bây giờ ta xét lời giải nâng cao của $y' = F(x, y)$ từ $x = x_0$ đến $x_0 + H$ bằng phương pháp trung điểm. Ra chia khoảng lấy tích phân thành n đoạn với chiều dài mỗi đoạn là $h = H/n$, như trong Hình 6.8, và thực hiện tính

$$\begin{aligned} y_1 &= y_0 + hF_0 \\ y_2 &= y_0 + 2hF_1 \\ y_3 &= y_1 + 2hF_2 \\ &\vdots \\ y_n &= y_{n-2} + 2hF_{n-1} \end{aligned} \tag{6.26}$$

Ở đây ta sử dụng ký hiệu $y_i = y(x_i)$, $F_i = F(x_i, y_i)$. Phương trình đầu tiên trong (6.26) sử dụng công thức Euler để “gioi hạt” cho công thức trung điểm, các phương trình còn lại là công thức trung điểm. Kết quả cuối cùng thu được bằng cách lấy trung bình của y_n trong (6.26) với xấp xỉ của $y_n \approx y_{n-1} + hF_n$ từ công thức Euler

$$y(x_0 + H) = \frac{1}{2} [y_n + (y_{n-1} + hF_n)]. \tag{6.27}$$

Quá trình xấp xỉ nghiệm của phương trình vi phân bằng công thức trung điểm 6.26-6.27 được minh họa bằng sơ đồ khối cho trong Hình 6.9.

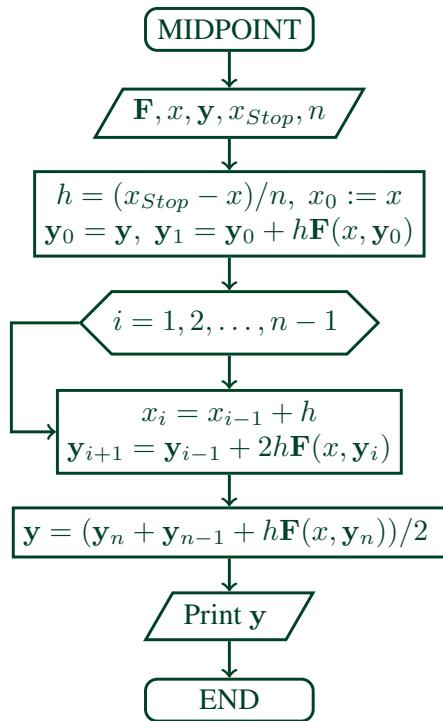
6.6.2 NGOẠI SUY RICHARDSON

Ta có thể thấy sai số trong (6.27) là

$$E = c_1 h^2 + c_2 h^4 + c_3 h^6 + \dots$$

Công thức này chính là một ưu điểm của phương pháp trung điểm vì chúng ta có thể khử các biểu thức đầu tiên trong công thức sai số trên đến độ chính xác mong muốn bằng ngoại suy Richardson. Chẳng hạn, chúng ta có thể tính $y(x_0 + H)$ với giá trị h và sau đó lặp lại với $h/2$. Ký hiệu kết quả tương ứng của hai quá trình này là $g(h)$ và $g(h/2)$. Theo công thức ngoại suy Richardson trong (5.9) thì ta có kết quả

$$y_{better}(x_0 + H) = \frac{4g(h/2) - g(h)}{3}$$



Hình 6.9: Sơ đồ khôi giải phương trình vi phân bằng công thức trung điểm

với độ chính xác bậc bốn. Tiếp tục lặp lại quá trình với $h/4$ thì bằng phương pháp ngoại suy Richardson ta sẽ thu được độ chính xác bậc sáu,v.v. Quá trình kết hợp công thức trung điểm và ngoại suy Richardson để giải phương trình vi phân được minh họa bằng sơ đồ khôi cho trong Hình 6.10.

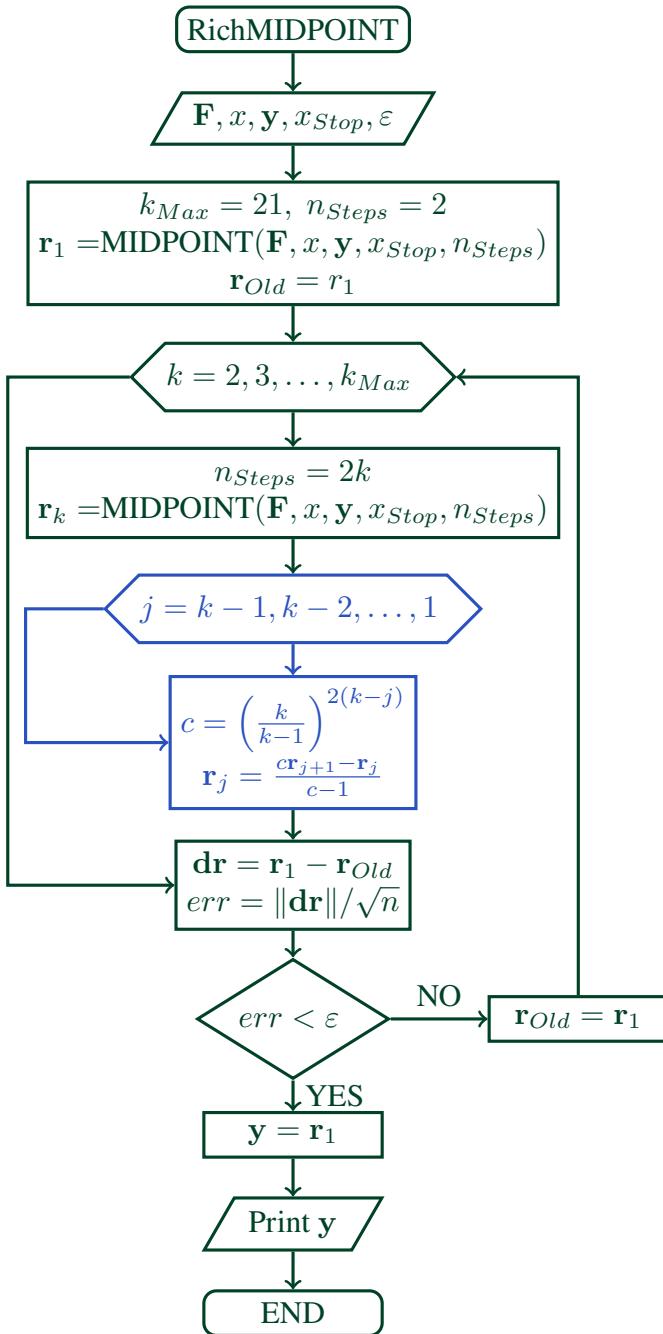
midpoint

Hàm midpoint sau đây kết hợp công thức trung điểm với ngoại suy Richardson. Trước tiên áp dụng công thức trung điểm cho hai quá trình tích phân. Số lượng các bước tăng gấp đôi trong lần tích phân kế tiếp, sau đó áp dụng ngoại suy Richardson. Quá trình sẽ dừng lại khi sai số giữa hai nghiệm liên tiếp (theo nghĩa căn bậc hai của trung bình bình phương sai số) nhỏ hơn độ chính xác cho phép.

```

function y=midpoint (dEqs,x,y,xStop,tol)
% Modified midpoint method for intergration of y' = F(x,y) .
% USAGE:y=midpoint (dEqs,xStart,yStart,xStop,tol)
% INPUT:
% dEqs = handle of function that returns the first-order
% differential equations F(x,y) = [dy1/dx,dy2/dx,...].
% x, y = initial values; y must be a row vector.
% xStop = terminal value of x.
% tol = per-step error tolerance (default = 1.0e-6).
% OUTPUT:

```



Hình 6.10: Sơ đồ khôi giải phương trình vi phân bằng công thức trung điểm kết hợp ngoại suy Richardson

%y=y(xStop) .

```

if size(y,1)>1;y=y';end % y must be row vector
if nargin <5; tol = 1.0e-6; end
kMax = 51;
n = length(y);
r = zeros(kMax,n); % Storage for Richardson extrapolation.
% Start with two integration steps.
nSteps = 2;
    
```

```

r(1,1:n) = mid(dEqs,x,y,xStop,nSteps);
rOld = r(1,1:n);
for k=2:kMax
    % Double the number of steps & refine results by
    % Richardson extrapolation.
    nSteps = 2*k;
    r(k,1:n) = mid(dEqs,x,y,xStop,nSteps);
    r = richardson(r,k,n);
    % Check for convergence.
    dr = r(1,1:n) - rOld;
    e = sqrt(dot(dr,dr)/n);
    if e<tol;y=r(1,1:n);return;end
    rOld = r(1,1:n);
end
error('Midpoint method did not converge')

function r=richardson(r,k,n)
    % Richardson extrapolation.
    for j=k-1:-1:1
        c =(k/(k-1))^(2*(k-j));
        r(j,1:n) =(c*r(j+1,1:n) - r(j,1:n))/(c - 1.0);
    end
    return

function y=mid(dEqs,x,y,xStop,nSteps)
    % Midpoint formulas.
    h = (xStop - x)/nSteps;
    y0=y;
    y1 = y0 + h*feval(dEqs,x,y0);
    for i=1:nSteps-1
        x=x+h;
        y2 = y0 + 2.0*h*feval(dEqs,x,y1);
        y0=y1;
        y1=y2;
    end
    y = 0.5*(y1 + y0 + h*feval(dEqs,x,y2));

```

VÍ DỤ 6.6.1. Tính nghiệm của bài toán giá trị ban đầu

$$y' = \sin y, \quad y(0) = 1$$

tại $x = 0.5$ bằng phương pháp trung điểm sử dụng $n = 2$ và $n = 4$, sau đó sử dụng ngoại suy Richardson (bài toán này đã được giải bằng phương pháp Runge-Kutta bậc hai trong VÍ DỤ 6.3.1).

GIẢI. Với $n = 2$ thì chiều dài bước là $h = 0.25$. Từ các công thức trung điểm trong (6.26) và (6.27) ta có

$$y_1 = y_0 + hf_0 = 1 + 0.25 \sin 1.0 = 1.210368$$

$$y_2 = y_0 + 2hf_1 = 1 + 2(0.25) \sin 1.210368 = 1.467873$$

$$\begin{aligned} y_h(0.5) &= \frac{1}{2}(y_2 + y_1 + hf_2) \\ &= \frac{1}{2}(1.467873 + 1.210368 + 0.25 \sin 1.467873) \\ &= 1.463459. \end{aligned}$$

Sử dụng $n = 4$ thì ta có $h = 0.125$ và các công thức trung điểm trở thành

$$y_1 = y_0 + hf_0 = 1 + 0.125 \sin 1.0 = 1.105184$$

$$y_2 = y_0 + 2hf_1 = 1 + 2(0.125) \sin 1.105184 = 1.223387$$

$$y_3 = y_1 + 2hf_2 = 1.105184 + 2(0.25) \sin 1.223387 = 1.340248$$

$$y_4 = y_2 + 2hf_3 = 1.223387 + 2(0.125) \sin 1.340248 = 1.466772$$

$$\begin{aligned} y_{h/2}(0.5) &= \frac{1}{2}(y_4 + y_3 + hf_4) \\ &= \frac{1}{2}(1.466772 + 1.340248 + 0.125 \sin 1.466772) \\ &= 1.465672. \end{aligned}$$

Kết quả ngoại suy Richardson trong

$$y(0.5) = \frac{4(1.465672) - 1.463459}{3} = 1.466410$$

là kết quả tốt so với nghiệm đúng $y(0.5) = 1.466404$.

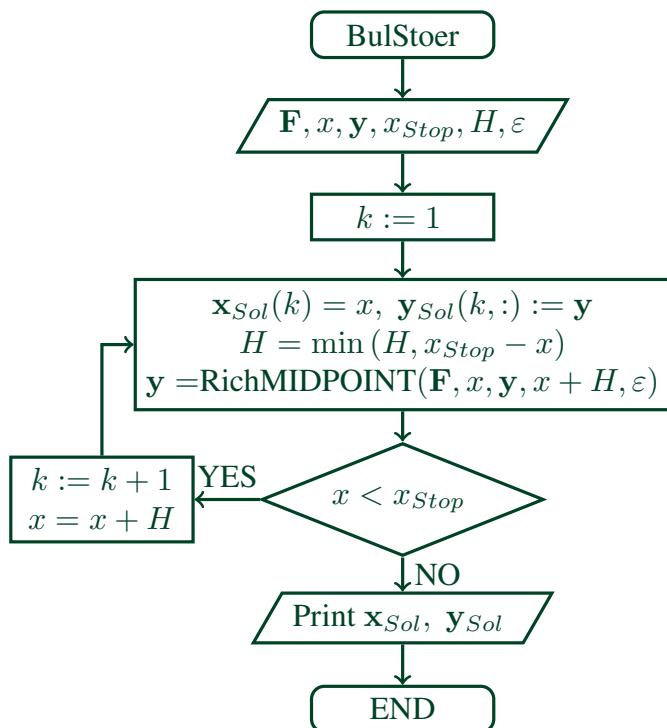
□

Lưu ý rằng các y_i trong (6.26) có thể được xem như là các biến trung gian bởi vì không giống như $y(x_0 + H)$, chúng không được làm mịn nhờ ngoại suy Richardson. Điểm hạn chế này được khắc phục bằng thuật toán Bulirsch-Stoer sẽ được trình bày trong mục sau đây.

6.6.3 THUẬT TOÁN BULIRSCH-STOER

Khi được sử dụng riêng, hàm midpoint có một thiếu sót lớn: nghiệm tại các điểm giữa giá trị ban đầu và giá trị cuối của x không thể được tinh chỉnh bằng phép ngoại suy Richardson, do đó, y chỉ có thể sử dụng được ở điểm cuối cùng. Sự thiếu sót này được khắc phục trong phương pháp Bulirsch-Stoer. Ý tưởng cơ bản đằng sau phương pháp này rất đơn giản: áp dụng phương pháp trung điểm theo kiểu từng phần. Đó là, chia khoảng lấy tích phân thành nhiều đoạn con có độ dài H , sau đó, sử dụng phương pháp trung điểm với phép ngoại suy Richardson để thực hiện tích phân trong đoạn con này. Giá trị của H có thể rất lớn nhưng không ảnh hưởng vì độ chính xác của kết quả được xác định chủ yếu bởi độ dài bước h trong phương pháp trung điểm.

Thuật toán Bulirsch-Stoer ban đầu² là một quy trình phức tạp kết hợp nhiều tinh chỉnh không có trong thuật toán của chúng tôi. Tuy nhiên, hàm bulStoer chúng tôi đưa ra dưới đây vẫn giữ nguyên các ý tưởng cơ bản của Bulirsch và Stoer. Hình 6.11 minh họa thuật toán Bulirsch-Stoer giải phương trình vi phân.



Hình 6.11: Sơ đồ khôi giải phương trình vi phân bằng thuật toán Bulirsch-Stoer

Những ưu điểm tương đối của các phương pháp Runge-Kutta thích ứng và Bulirsch-Stoer là gì? Phương pháp Runge-Kutta thích ứng vững hơn (robust) nên có khả năng tính toán hiệu quả hơn khi thực hiện tính toán với các hàm không trơn và các bài toán cương. Trong hầu hết các ứng dụng không yêu cầu độ chính xác cao, nó cũng có

²Stoer,J., and Bulirsch, R., Introduction to Numerical Analysis, Springer, 1980.

xu hướng hiệu quả hơn. Tuy nhiên, phương pháp Runge-Kutta thích ứng không được lựa chọn khi yêu cầu nghiệm có độ chính xác cao liên quan đến các hàm trơn, trong trường hợp này, thuật toán Bulirsch-Stoer thường được lựa chọn.

bulStoer.

Hàm này có chứa các thuật toán đơn giản cho phương pháp Bulirsch-Stoer.

```

function [xSol,ySol] = bulStoer(dEqs,x,y,xStop,H,tol)
% Simplified Bulirsch-Stoer method for integration of y' =
F(x,y)
% USAGE: [xSol,ySol] = bulStoer(dEqs,x,y,xStop,H,tol)
% INPUT:
% dEqs = handle of function that returns the first-order
% differential equations F(x,y) = [dy1/dx,dy2/dx,...].
% x, y = initial values; y must be a row vector.
% xStop = terminal value of x.
% H = increment of x at which solution is stored.
% tol = per-step error tolerance (default = 1.0e-6).
% OUTPUT:
% xSol, ySol = solution at increments H.

if size(y,1)>1;y=y';end % y must be row vector
if nargin < 6; tol = 1.0e-6; end
n = length(y);
xSol = zeros(2,1); ySol = zeros(2,n);
xSol(1) = x; ySol(1,:) = y;
k=1;
while x<xStop
    k=k+1;
    H = min(H,xStop - x);
    y = midpoint(dEqs,x,y,x + H,tol);
    x=x+H;
    xSol(k) = x; ySol(k,:) = y;
end

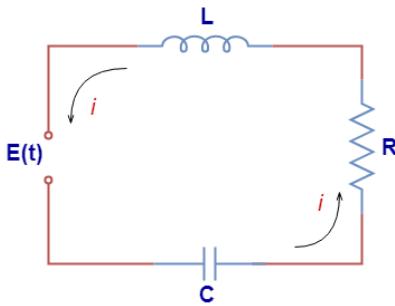
```

VÍ DỤ 6.6.2. Các phương trình vi phân điều khiển dòng điện vòng i và điện tích q

trên tụ điện của mạch điện là

$$L \frac{di}{dt} + Ri + \frac{q}{C} = E(t), \quad \frac{dq}{dt} = i.$$

Nếu điện áp E bất ngờ tăng từ không đến $9V$ thì hãy vẽ kết quả dòng điện vòng trong mươi giây đầu tiên. Với $R = 10\Omega$, $L = 2H$ và $C = 0.45F$.



GIẢI. Đặt

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} q \\ i \end{bmatrix}$$

và thay dữ liệu đã cho, các phương trình vi phân trở thành

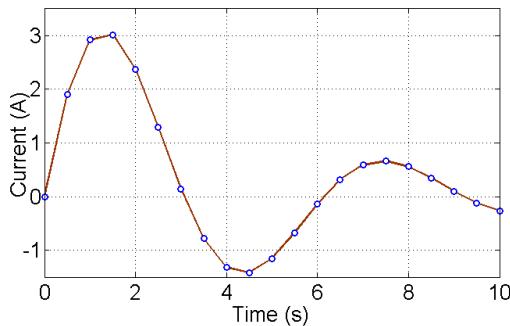
$$\dot{\mathbf{y}} = \begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \end{bmatrix} = \begin{bmatrix} y_2 \\ (-Ry_2 - y_1/C + E)/L \end{bmatrix}.$$

Các điều kiện ban đầu là

$$\mathbf{y}(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Chúng ta giải bài toán bằng hàm bulStoer với số gia $H = 0.5$ s. Chương trình sau đây được sử dụng để vẽ kết quả của dòng điện vòng trong mươi giây đầu bằng Matlab.

```
% Example 6.6.2 (Bulirsch-Stoer integration)
[xSol,ySol] = bulStoer(@fex6_6_2_11,0,[0 0],10,0.5);
plot(xSol,ySol(:,2),'k:o')
grid on
xlabel('Time (s)')
ylabel('Current (A)')
```



6.7 MỘT SỐ HÀM MATLAB GIẢI PHƯƠNG TRÌNH VI PHÂN

Một số hàm Matlab thông dụng dùng để giải phương trình vi phân sẽ được giới thiệu sau đây.

[xSol, ySol] = ode23(dEqs, [xStart, xStop], yStart) Phương pháp Runge-Kutta thích nghi bậc thấp (có thể là bậc ba). Hàm text phải trả các phương trình vi phân dưới dạng một vectơ cột (nhắc lại rằng hàm runKut4 và hàm runKut5 yêu cầu các vectơ hàng). Khoảng lấy tích phân là từ xStart đến xStop với điều kiện ban đầu yStart (cũng là một vectơ cột).

[xSol, ySol] = ode45(dEqs, [xStart, xStop], yStart) tương tự như ode23 nhưng sử dụng phương pháp Runge-Kutta bậc cao hơn (có thể bậc năm).

Hai phương pháp này, cũng như tất cả các phương pháp được mô tả trong cuốn sách này, thuộc về nhóm được gọi là *phương pháp một bước*. Tên gọi này bắt nguồn từ thực tế là chỉ sử dụng thông tin tại một điểm duy nhất trên đường cong nghiệm là đủ để tính điểm tiếp theo. Ngoài ra còn có các phương pháp đa bước sử dụng một số điểm trên đường cong để ngoại suy nghiệm ở bước tiếp theo. Những phương pháp này đã từng phổ biến nhưng đã mất đi một phần hắp dẫn trong vài năm qua. Phương pháp nhiều bước có hai vấn đề làm phức tạp việc thực hiện của chúng:

- Các phương pháp đa bước không tự bắt đầu, nó phải sử dụng các phương pháp một bước để tính nghiệm tại những điểm đầu tiên.

- Các công thức tích phân của các phương pháp đa bước sử dụng các bước cách đều nhau, điều này gây khó khăn cho việc thay đổi kích thước bước.

Cả hai vấn đề này có thể vượt qua, nhưng giá là độ phức tạp của thuật toán tăng lên với độ điều chỉnh của phương pháp. Lợi ích của các phương pháp đa bước là rất ít, hiếm khi chúng vượt qua được các phương pháp một bước. MATLAB cung cấp một phương pháp đa bước chung là:

`[xSol, ySol] = ode113(dEqs, [xStart, xStop], yStart)` sử dụng phương pháp Adams–Bashforth–Moulton.

MATLAB cũng cung cấp một số hàm số để giải các bài toán cương. Đó là `ode15s` (đây là phương pháp ta nén thử trước tiên khi gấp bài toán cương), `ode23s`, `ode23t` và `ode23tb`.

TÓM TẮT CHƯƠNG 6

Nội dung chính của Chương 6 trình bày các phương pháp xấp xỉ nghiệm của phương trình vi phân thường. Các phương pháp xấp xỉ này được chia làm hai nhóm là các phương pháp giải tích và phương pháp Runge–Kutta. Phương pháp giải tích dựa vào việc giữ lại các biểu thức đầu tiên trong khai triển Taylor. Phương pháp này có ưu điểm là có thể tính toán với độ chính xác cao nhưng đi đôi với việc này là phải thực hiện tính toán bằng tay các đạo hàm các đạo hàm cấp cao của hàm số.

Các phương pháp Runge–Kutta khắc phục được nhược điểm của phương pháp Taylor là không cần tính đạo hàm các cấp. Có rất nhiều phương pháp Runge–Kutta với độ chính xác khác nhau. Trong chương này chỉ giới thiệu ba phương pháp sau:

1. Phương pháp Euler.
2. Phương pháp Runge–Kutta bậc hai.
3. Phương pháp Runge–Kutta bậc bốn.

Do các phương pháp Runge–Kutta chỉ làm việc tốt khi các mốc cách đều. Điều này dẫn các phương pháp này làm việc không hiệu quả khi phải giải bài toán có tính cương. Các phương pháp Runge–Kutta được cải tiến thành các phương pháp Runge–Kutta thích ứng, đây là các phương pháp đưa ra khoảng cách lặp phù hợp cho mỗi bước, do vậy thích hợp để giải các bài toán cương hoặc hàm số ở về phải không trơn.

Phần cuối của chương này giới thiệu một thuật toán có tốc độ hội tụ cao là thuật toán Bulirsch-Stoer, đây là thuật toán được xây dựng dựa trên việc áp dụng công thức ngoại suy Richardson cho các công thức trung điểm. Bằng cách lặp lại công thức trung điểm ở các mốc khác nhau để khử sai số, chúng ta sẽ được công thức xấp xỉ với độ chính xác mong muốn.

BÀI TẬP CHƯƠNG 6

Bài tập về phương pháp chuỗi Taylor và phương pháp Runge-Kutta

Bài tập 6.1. Cho

$$y' + 4y = x^2, \quad y(0) = 1$$

tính $y(0.1)$ bằng cách sử dụng mmot bước của phương pháp chuỗi Taylor với bậc là (a) hai và (b) bốn. So sánh kết quả với nghiệm giải tích

$$y(x) = \frac{31}{32}e^{-4x} + \frac{1}{4}x^2 - \frac{1}{8}x + \frac{1}{32}.$$

Bài tập 6.2. Giải bài toán trong Bài tập 6.1 với một bước của phương pháp Runge-Kutta có bậc là (2) và (b) bốn.

Bài tập 6.3. Tích phân

$$t' = \sin y, \quad y(0) == 1$$

từ $x = 0$ tới 0.5 với phương pháp chuỗi Taylor bậc bốn sử dụng $h = 0.1$. So sánh kết quả với Ví dụ 6.3.1.

Bài tập 6.4. Chuyển các phương trình vi phân sau đây về hệ phương trình vi phân bậc một dạng $y' = F(x, y)$:

- a. $\ln y' + y = \sin x$
- b. $y''y - xy' - 2y^2 = 0$
- c. $y^{(4)} - 4y''\sqrt{1 - y^2} = 0$
- d. $(y'')^2 = |32y'x - y^2|$

Bài tập 6.5. Chuyển các cặp phương trình vi phân dưới đây về hệ phương trình vi phân bậc một dạng $\dot{y} = F(t, y)$:

- a. $\ddot{y} = x - 2y$ $\ddot{x} = y - x$
- b. $\ddot{y} = -y \left((dot{y}^2 + \dot{x}^2)^{1/4} \right)$ $\ddot{x} = -x \left(\dot{y}^2 + \dot{x}^2 \right)^{1/4} - 32$
- c. $\ddot{y}^2 + t \sin y = 4\dot{x}$ $x\ddot{x} + t \cos y y = 4\dot{y}$

Bài tập 6.6. Phương trình vi phân cho chuyển động của con lắc đơn là

$$\frac{d^2\theta}{dt^2} = -\frac{g}{L} \sin \theta$$

với

$$\begin{aligned}\theta &= \text{góc dịch chuyển so với trục dọc} \\ g &= \text{gia tốc trọng trường} \\ L &= \text{chiều dài của con lắc.}\end{aligned}$$

Với phép đổi biến $\tau = t\sqrt{g/L}$ phương trình trở thành

$$\frac{d^2\theta}{d\tau^2} = -\sin \theta.$$

Sử dụng tích phân số xác định chu kỳ tuần hoàn của con lắc nếu biên độ $\theta_0 = 1$ rad. Chú ý rằng với biên độ nhỏ ($\sin \theta \approx \theta$) chu kỳ là $2\pi\sqrt{L/g}$.

Bài tập 6.7. Một người nhảy dù khói lượng m trong một cú rơi tự do thẳng đứng trải qua lực kéo khí động học $F_D = c_D y^2$, trong đó y được đo xuống từ đầu cú rơi. Phương trình vi phân mô tả cú rơi là

$$\ddot{y} = g - \frac{c_D}{m} \dot{y}^2.$$

Xác định thời gian rơi 500 m. Sử dụng $g = 9.80665 \text{ m/s}^2$, $c_D = 0.2028 \text{ kg/m}$ và $m = 80 \text{ kg}$.

Bài tập về phương pháp Runge-Kutta thích nghi và phương pháp Bulirsch-Stoer

Bài tập 6.8. Tìm nghiệm giải tích của bài toán

$$y'' + y' - 380y = 0, \quad y(0) = 1, \quad y'(0) = -20.$$

Liệu có khó khăn khi giải số bài toán này?

Bài tập 6.9. Xét bài toán

$$y' = x - 10y, \quad y(0) = 10.$$

- a. Xác nhận rằng nghiệm giải tích là $y(x) = 0.1x - 0.01 + 10.01e^{-10x}$.

- b. Xác định chiều dài bước h mà ta sử dụng để giải số bài toán bằng phương pháp Runge-Kutta (không thích nghi).

Bài tập 6.10. Tích phân bài toán giá trị ban đầu trong Bài tập 6.9 từ $x = 0$ đến 5 bằng phương pháp Runge-Kutta sử dụng (a) $h = 0.1$; (b) $h = 0.25$; và (c) $h = 0.5$. Bình luận về các kết quả.

Bài tập 6.11. Tích phân bài toán giá trị ban đầu trong Bài tập 6.9 từ $x = 0$ đến 10 bằng phương pháp Runge-Kutta thích nghi.

Bài tập 6.12. Tính nghiệm số của phương trình vi phân

$$y'' = 16.81y$$

từ $x = 0$ đến 2 bằng phương pháp Runge-Kutta thích nghi. Sử dụng các điều kiện ban đầu

a. $y(0) = 1.0, y'(0) = -4.1$

b. $y(0) = 1.0, y'(0) = -4.11$

Giải thích sự khác nhau lớn giữa hai nghiệm (Gợi ý: Tìm nghiệm giải tích)

Bài tập 6.13. Tích phân

$$y'' + y' - y^2 = 0, \quad y(0) = 1, y'(0) = 0$$

từ $x = 0$ đến 3.5. Điều tra xem sự gia tăng đột ngột của y gần cận trên là có thật hay là bị gây ra bởi sự mất ổn định. Gợi ý: thử nghiệm với các giá trị khác nhau của h .

Bài tập 6.14. Giải bài toán cương (xem phương trình (6.16))

$$y'' + 1001y' + 1000y = 0, \quad y(0) = 1, y'(0) = 0$$

từ $x = 0$ đến 0.2 bằng phương pháp Runge-Kutta thích nghi và vẽ đồ thị y' và x .

Bài tập 6.15. Giải

$$y'' + 2y' + 3y = 0, \quad y(0) = 1, y'(0) = \sqrt{2}$$

bằng phương pháp Runge-Kutta thích nghi từ $x = 0$ đến 5 (nghiệm giải tích là $y = e^{-x} \sin \sqrt{2}x$).

Bài tập 6.16. Sử dụng phương pháp Runge-Kutta thích ứng giải phương trình vi phân

$$y'' = 2yy'$$

từ $x = 0$ đến 10 với điều kiện ban đầu $y(0) = 1, y'(0) = -1$. Vẽ y' và x .

Bài tập 6.17. Sử dụng phương pháp Runge-Kutta thích ứng tính tích phân

$$y' = \left(\frac{9}{9} - y \right) x, \quad y(0) = 5$$

từ $x = 0$ đến 5 và vẽ y' và x .

Bài tập 6.18. Sử dụng phương pháp Bulirsch–Stoer tính tích phân

$$x^2 y'' + xy' + y = 0, \quad y(1) = 0, y'(1) = -2$$

từ $x = 1$ đến 20 và vẽ y' và x .

PHỤ LỤC A: DANH SÁCH CÁC HÀM MATLAB ĐƯỢC LẬP TRÌNH TRONG SÁCH

Thứ tự	Tên hàm	Trang	Nội dung
1	rootsearch	18	Tìm các khoảng chứa nghiệm của phương trình bằng phương pháp tìm kiếm gia tăng.
2	bisect	20	Xấp xỉ nghiệm của phương trình bằng phương pháp chia đôi.
3	newtonRaphson	33	Xấp xỉ nghiệm của phương trình bằng phương pháp Newton-Raphson.
4	evalpoly	41	Tính giá trị của đa thức và đạo hàm cấp một, đạo hàm cấp hai của đa thức.
5	polyroots	45	Tìm các 0-điểm của đa thức bằng phương pháp lặp Laguerre.
6	gauss	64	Giải hệ phương trình đại số tuyến tính bằng phương pháp khử Gauss.
7	LUdec	71	Phân rã Doolittle cho ma trận A.
8	LUsol	73	Thực hiện pha giải hệ phương trình đại số tuyến tính khi ma trận A đã được phân rã Doolittle.
9	choleski	78	Phân rã Choleski cho ma trận đối xứng A.
10	LUdec3	83	Phân rã LU cho ma trận 3 đường chéo.
11	LUsol3	85	Giải hệ phương trình đại số tuyến tính khi ma trận ba đường chéo A đã được phân rã LU bằng LUdec3.
12	LUdec5	90	Phân rã LU một ma trận năm đường chéo.
13	LUsol5	92	Giải hệ phương trình đại số tuyến tính khi ma trận năm đường chéo A đã được phân rã LU bằng LUdec5.
14	swapRows	99	Đổi vị trí hàng i với hàng j của ma trận hoặc vectơ v.
15	gaussPiv	99	Thực hiện phương pháp khử Gauss với việc chọn phần tử xoay.

Thứ tự	Tên hàm	Trang	Nội dung
16	LUdecPiv	101	Hàm LUdecPiv.
17	LUsolPiv	103	Hàm LUsolPiv.
18	gaussSeidel	112	Giải hệ phương trình tuyến tính theo phương pháp lặp Gauss-Seidel.
19	conjGrad	119	Giải hệ phương trình đại số tuyến tính bằng thuật toán Gradient liên hợp.
20	newtonCoeff	139	Tính toán hệ số a của đa thức nội suy Newton.
21	newtonPoly	138	Tính toán đa thức nội suy Newton khi các hệ số đã được tính bởi newtonCoeff.
22	neville	144	Tính đa thức nội suy bằng phương pháp Neville
23	polynFit	152	Tính toán các hệ số của đa thức bậc $m - 1$ khi xấp xỉ n điểm dữ liệu bằng phương pháp bình phương nhỏ nhất.
24	stdDev	154	Sau khi hệ số của đa thức xấp xỉ được tính, độ lệch chuẩn σ được tính bởi hàm stdDev. Giá trị của đa thức trong stdDev được tính bởi hàm polyEval trong công thức (4.10) tài liệu tham khảo.
25	polyEval	155	Tính gias trị của đa thức tại một điểm
26	trapezoid	186	Tính tích phân xác định bằng công thức hình thang.
27	romberg	194	Tính tích phân xác định bằng công thức tích phân Romberg.
28	gaussNodes	206	Hàm gaussNodes tính các mốc x_i và các trọng số A_i tương ứng bằng các sử dụng phương pháp cầu phương Gauss-Legendre.
29	legendre	208	Hàm con legendre được lập trình trong M-file sau đây.
30	gaussQuad	209	Tính tích phân xác định bằng phương pháp cầu phương Legendre với n mốc. Các mốc và trọng số được tính bằng hàm gaussNodes.
31	gaussQuad2	218	Tính tích phân bội trên phần tử tứ giác bằng phương pháp cầu phương với n bậc tích phân.

Thứ tự	Tên hàm	Trang	Nội dung
32	triangleQuad	224	Tính tích phân bội bằng công thức lập phương trên miền tam giác.
33	taylor	236	Hàm taylor thực hiện phương pháp chuỗi Taylor với bậc tích phân bằng bốn. Nó có thể xử lý tùy ý số lượng các phương trình vi phân bậc một $y'_i = f_i(x, y_1, y_2, \dots, y_n), i = 1, 2, \dots, n$. Người dùng được yêu cầu cung cấp hàm deriv trả về mảng $4 \times n$
34	printSol	237	Hàm printSol thực hiện in kết quả xSol và ySol ra dạng bảng.
35	runKut4	247	Thực hiện phương pháp Runge-Kutta bốn bước giải phương trình vi phân thường.
36	runKut5	260	Thực hiện phương pháp Runge-Kutta thích nghi giải phương trình vi phân thường.
37	midpoint	267	Giai phương trình vi phân thường bằng cách kết hợp công thức trung điểm với ngoại suy Richardson.
38	bulStoer	272	Hàm này có chứa các thuật toán đơn giản cho phương pháp Bulirsch-Stoer.

CHỈ MỤC

- Điều kiện ổn định của PP. Euler, 258
độc lập tuyến tính, 57
đa thức nội suy Newwton, 140
- cách viết xấp xỉ, 9
công thức hình thang đệ quy, 188
công thức hình thang kết hợp, 185
công thức Lagrange, 137
công thức Newton-Cotes, 183
công thức Simpson 1/3, 192
công thức Simpson 3/8, 193
cầu phương Gauss với tính kỳ dị, 208
cầu phương Gauss-Chebysev, 206
cầu phương Gauss-Hermit, 207
cầu phương Gauss-Laguerre, 206
cầu phương Gauss-Legendre, 205
cầu phương Gauss-Legendre cho tứ giác, 219
cầu phương Gauss-Legendre cho tam giác, 225
chữ số đáng tin, 9
chữ số có nghĩa, 9
chuẩn ma trận, 57
- hồi quy có trọng số, 160
hồi quy tuyến tính, 151
hàm cơ bản, 137
hàm hình dạng, 218
hệ phương trình chuẩn, 153
Hệ phương trình vi phân cấp một, 239
hệ số đa thức trực giao, 204
- ma trận ba đường chéo, 85
ma trận chéo trội, 98
- ma trận dải, 84
ma trận hệ số mở rộng, 57
ma trận Jacobian, 218
- nội suy Neville, 146
ngoại suy Richardson, 177
- phân rã LU, 71
phương pháp lặp, 30
phương trình vi phân cấp, 238
PP. Bulirsch-Stoer, 275
PP. chuỗi Taylor, 239
PP. Euler, 245
PP. Euler cải tiến, 248
PP. lặp đơn giải HPTTT, 108
PP. lặp Jacobi giải HPTTT, 110
PP. lặp Seidel giải HPTTT, 112
PP. Runge-Kutta bốn bước, 251
PP. Runge-Kutta thích ứng, 261
PP. Trung điểm, 270
- quy tròn số, 8
- số điều kiện ma trận, 57
sai số chặt, 239
sai số phương pháp lặp đơn, 32
sai số tương đối, 7
sai số tuyệt đối, 6
- tích phân Romberg, 195
tính ổn định của PTVP, 258
tính cương của PTVP, 259
tính hội tụ, 31
- xấp xỉ mũ, 160
xấp xỉ sai phân hữu hạn tiến thứ hai, 174

- xấp xỉ sai phân lùi thứ nhất, 173
- xấp xỉ sai phân tiến thứ nhất, 173
- xấp xỉ sai phân trung tâm thứ nhất, 172

TÀI LIỆU THAM KHẢO

- [1] Tạ Văn Đĩnh (2001), “*Phương Pháp Tính*”, Nhà xuất bản Giáo Dục.
- [2] Phạm Kỳ Anh (2000), “*Giải Tích Số*”, Nhà xuất bản Đại Học Quốc Gia Hà Nội.
- [3] Trần Văn Trần (2007), “*Phương Pháp Số Thực Hành*”, T1,T2, Nhà xuất bản Đại Học Quốc Gia Hà Nội.
- [4] Epperson, J.F., (2013), “*An Introduction to Numerical Methods and Analysis*”, John Wiley and Sons, New Jersey.
- [5] Chapra,S.C., (2017), “*Applied Numerical Methods with Matlab for Engineers and Scientist*”, McGraw Hill, New York.
- [6] Kiusalaas, J., (2005), ‘*Numerical Methods in Engineering with MATLAB*’, Cambridge University Press.

