GIẢI TÍCH SỐ I

Trịnh Anh Ngọc

1/9/2009

Mục đích của giải tích số: phát triển các phương pháp hiệu quả và chính xác để tính xấp xỉ các đại lượng mà khó hoặc không thể nhận được bằng các phương tiện giải tích.

■ Mục đích

1. Cung cấp kiến thức cơ sở về phương pháp tính.
2. Sinh viên biết áp dụng, thực hiện (bằng máy tính) và phân tích kết quả tính toán.
3. Sinh viên có thể tự đọc các sách, bài báo về phương pháp tính.

■ Nội dung

1. Sai số và số học dấu chấm động
2. Hệ phương trình đại số tuyến tính
3. Nội suy
4. Các phương trình phi tuyến
5. Đạo hàm và tích phân số
6. Phương trình vi phân thường

■ Tài liệu đọc thêm

David Kincaid and Ward Cheney, Numerical Analysis Mathematics of Scientific Computing, Brooks/Cole Publishing Company 1991.

Chương 1

Sai số và số học dấu chấm động

Các loại sai số:

- sai số khi thiết lập mô hình (toán học);

- sai số do phép đo dữ liệu của bài toán;

- sai số do phương pháp tính, gọi là sai số rời rạc hóa (discretization error) hay sai số chặt cụt (truncation error);

- sai số do phép biểu diễn số (bằng một số hữu hạn các bit) và tính toán trong máy tính, gọi là sai số làm tròn (roundoff error).

1.1 Các khái niệm cơ bản

Hai cách đo độ chính xác của một đại lượng xấp xỉ: *sai số tuyệt đối* (absolute error) và *sai số tương đối* (relative error).

Định nghĩa 1.1. Cho $\tilde{x}$ là giá trị xấp xỉ của x. Thì sai số tuyệt đối trong $\tilde{x}$ là

$$\Delta x = x - \tilde{x},$$

và nếu $x \neq 0$ sai số tương đối là

$$\frac{\Delta x}{x} = \frac{x - \tilde{x}}{x}.$$

Sai số tương đối thể hiện độ chính xác tốt hơn, nhưng sai số tuyệt đối lại có ích khi giá trị chính xác gần bằng không.

Bài toán số (numerical problem) là mối quan hệ hàm giữa dữ liệu nhập (input data) - "biến độc lập" trong bài toán - và dữ liệu xuất (output data) - kết quả cần tìm. Dữ liệu nhập và xuất gồm một số hữu hạn các đại lượng thực (hoặc phức) và như vậy được biểu diễn bởi các vectơ có kích thước hữu hạn. Mối quan hệ hàm có thể biểu diễn dưới dạng ẩn hoặc hiển. Thường ta đòi hỏi dữ liệu xuất phải được xác định duy nhất và phụ thuộc liên tục vào dữ liệu nhập.

Thuật toán (algorithm) cho một bài toán số là sự mô tả đầy đủ các phép toán xác định tốt qua đó mỗi vectơ dữ liệu nhập khả dĩ được chuyển thành một vectơ dữ liệu xuất. Các"phép toán" ở đây được hiểu là các phép toán số học và lôgic mà máy tính có thể thực hiện được, hoặc là tham chiếu đến những thuật toán đã biết.

Phương pháp số (numerical method) là một thủ tục để xấp xỉ một bài toán toán học bằng một bài toán số hay để giải một bài toán số (hay ít nữa là dẫn nó về một bài toán đơn giản hơn). Thuật ngữ phương pháp số tổng quát và được dùng rộng rãi hơn thuật toán, nó ít nhấn mạnh vào các chi tiết tính toán.

Thí dụ 1.1. Xác định nghiệm thực lớn nhất của phương trình bậc ba

$$p(z) = a_0 z^3 + a_1 z^2 + a_2 z^1 + a_3 = 0$$

với các hệ số thực a_0, a_1, a_2, a_3, là một bài toán số. Vectơ dữ liệu nhập là (a_0, a_1, a_2, a_3). Dữ liệu xuất là nghiệm x cần tìm. Một thuật toán cho bài toán này có thể được xây dựng dựa vào phương pháp Newton, cùng quy tắc chọn giá trị đầu và điệu kiện dừng thuật toán. Cũng có thể xây dựng thuật toán trên cơ sở công thức (cho nghiệm chính xác) của Cardano. Công thức này chứa căn bậc hai và bậc ba vì vậy ta cần chỉ định thuật toán tính các căn thức này ∘

Một bài toán toán học với dữ liệu nhập x và dữ liệu xuất $y = F(x)$ được gọi là *điều kiện tốt* (well-conditioned) nếu những thay đổi "nhỏ" trong x dẫn đến những thay đổi "nhỏ" trong y. Nếu sự thay đổi trong y lớn, bài toán được gọi là (ở trong) *điều kiện xấu* (ill-condition). Điều kiện tốt hay xấu có thể phụ thuộc vào cách đo sự thay đổi. Về phương diện tính toán, điều kiện của bài toán có liên hệ với tính ổn định (stability) của thuật toán. Một thuật toán là *ổn định* (stable) nếu những thay đổi "nhỏ" trong dữ liệu nhập dẫn đến những thay đổi "nhỏ" trong dữ liệu xuất. Trường hợp ngược lại, ta nói thuật toán *không ổn định* (unstable).

Thí dụ 1.2. Cho hàm $F(x)$ khả vi. Giả sử đối số nhập x có sai số tương đối bằng ϵ, khi đó sai số tuyệt đối trong giá trị xuất của $F(x)$ là

$$F(x) - F(x + \epsilon x) \approx -\epsilon x F'(x).$$

Sai số tương đối là

$$\frac{F(x) - F(x + \epsilon x)}{F(x)} = -\epsilon x \frac{F'(x)}{F(x)}.$$

Trường hợp $F(x) = e^x$, sai số tuyệt đối trong giá trị của hàm mũ gây ra do sai số ϵx trong đối số x được xấp xỉ bởi $-\epsilon x e^x$, và sai số tương đối áng chừng $-\epsilon x$. Khi x lớn, điều kiện của phép (bài toán) đánh giá hàm này đối với sai số tương đối ϵ nhỏ phụ thuộc rất nhiều vào việc chọn cách đo sai số.

Trường hợp $F(x) = \cos(x)$, ở gần $x = \pi/2$, sai số tuyệt đối do sự nhiễu x thành $x + \epsilon x$ xấp xỉ bằng $\epsilon x \sin(x) \approx \epsilon\pi/2$. Sai số tương đối tại $\pi/2$ không xác định. Tuy nhiên, các giá trị chính xác

$$\cos(1.57079) = 0.63268 \times 10^{-5}, \quad \cos(1.57078) = 1.6327 \times 10^{-5}$$

cho thấy một thay đổi rất nhỏ trong đối số gần $\pi/2$ có thể dẫn tới sai số tương đối trong giá trị hàm rất lớn (61%) ∘

Thí dụ 1.3. Tích phân từng phần thường được dùng để thiết lập công thức truy hồi. Thí dụ, xét

$$E_n = \int_0^1 x^n e^{x-1} dx \text{ với } n = 1, 2, \ldots \tag{1.1}$$

Từ (1.1) ta có ngay

$$E_1 > E_2 > \ldots > 0. \tag{1.2}$$

Áp dụng công thức tích phân từng phần, sau một số biến đổi, ta được công thức truy hồi.

$$E_n = 1 - nE_{n-1}. \tag{1.3}$$

Thành phần đầu

$$E_1 = 1 - \int_0^1 e^{x-1} dx = 1/e,$$

dùng Matlab (IEEE-64[1], chính xác đơn) ta tính được:

$$\begin{aligned}
E_1 &= 0.3679 \\
E_2 &= 0.2642 \\
&\dots \\
E_{16} &= 0.0555 \\
E_{17} &= 0.0572\ E_n \text{ không giảm!} \\
E_{18} &= -0.0295\ E_n \text{ không dương!} \\
&\dots \\
E_{20} &= -30.1924\ E_n \text{ không nằm giữa 0 và 1!}
\end{aligned}$$

Đây là một thí dụ về thuật toán không ổn định. Phân tích: Giả sử ta bắt đầu bằng $\hat{E}_1 = E_1 + \delta$, và các tính toán theo sau không có sai số. Thì

$$\begin{aligned}
\hat{E}_2 &= 1 - 2\hat{E}_1 = 1 - 2\hat{E}_1 - 2\delta = E_2 - 2\delta, \\
\hat{E}_3 &= 1 - 3\hat{E}_2 = 1 - 3\hat{E}_2 + 6\delta = E_3 + 6\delta, \\
&\dots \\
\hat{E}_n &= E_n + (-1)^{n-1} n!\delta.
\end{aligned}$$

Một thay đổi nhỏ trong giá trị đầu E_1 "lớn lên" rất nhanh trong E_n sau đó.

Ảnh hưởng này là "xấu" vì các đại lượng E_n giảm khi n tăng.

Một phương pháp thường dùng để cải thiện tính ổn định là viết lại công thức hoặc thay đổi thứ tự tính toán. Giả sử ta biết giá trị xấp xỉ $\hat{E}_N$ của E_N với N nào đó, ta có thể đánh giá tích phân theo công truy hồi ngược:

$$E_{n-1} = \frac{1 - E_n}{n} \quad n = N, N-1, \dots, 2. \tag{1.4}$$

[1]IEEE, viết tắt của Institute of Electrical and Electronics Engineers, là một hiệp hội thế giới các chuyên gia kỹ thuật.

Nghiên cứu sự ổn định của thuật toán giống như trên. Nếu $\hat{E}_N = E_N + \epsilon$ thì

$$\begin{aligned}
\hat{E}_{N-1} &= \frac{1-\hat{E}_N}{N} = \frac{1-E_N}{N} - \frac{\epsilon}{N} = E_{N-1} - \frac{\epsilon}{N} \\
\hat{E}_{N-2} &= E_{N-2} + \frac{\epsilon}{N(N-1)} \\
&\vdots \\
\hat{E}_1 &= E_1 \pm \frac{\epsilon}{N!}.
\end{aligned}$$

Bằng công thức (1.4) ta có thể tính xấp xỉ E_n bắt đầu từ $\hat{E}_N$ với N đủ lớn. Thật vậy, từ bất đẳng thức

$$0 < E_n < \int_0^1 x^n dx = \frac{1}{n+1}$$

ta thấy có thể chọn N để cho sai số tuyệt đối của E_n cần tính không vượt quá giới hạn cho trước ∘

1.2 Biểu diễn số trong máy tính

1.2.1 Số dấu chấm động

Số thực $x \neq 0$ bất kỳ có thể được biểu diễn trong hệ thập phân như sau

$$x = \pm .d_1 d_2 \ldots d_s d_{s+1} \ldots \times 10^e, \tag{1.5}$$

trong đó $d_1 > 0$ và mỗi d_i $(i = 2, 3, \ldots)$ nhận một trong các giá trị $0, \ldots, 9$. Phần $.d_1 d_2 \ldots$ được gọi là phần định trị (mantissa),

$$.d_1 d_2 \ldots = d_1 \times 10^{-1} + d_2 \times 10^{-2} + \ldots$$

Các tính toán trên máy tính được thực hiện trên hệ thống số dấu chấm động (floating point). Đây là hệ thống số dùng một số hữu hạn các con số để xấp xỉ hệ thống số thực (vô hạn). Tất cả các số thuộc hệ thống với s con số dùng cơ số 10 có dạng

$$x = \pm .d_1 d_2 \ldots d_s \times 10^e \tag{1.6}$$

trong đó $m \le e \le M$. Số không là một trường hợp đặc biệt, nó được viết như là

$$0.0\ldots0 \times 10^m \ (?)$$

Thí dụ 1.4. Nếu $s = 1, m = -1, M = 1$, thì tập hợp các số dấu chấm động là

$$\pm 0.1 \times 10^{-1}, \pm 0.2 \times 10^{-1}, \ldots, \pm 0.9 \times 10^{-1},$$
$$\pm 0.1 \times 10^{0}, \pm 0.2 \times 10^{0}, \ldots, \pm 0.9 \times 10^{0},$$
$$\pm 0.1 \times 10^{1}, \pm 0.2 \times 10^{1}, \ldots, \pm 0.9 \times 10^{1}$$

cùng với số không 0.0×10^{-1}. Như vậy có tất cả 55 con số ∘

Hình 1.1: Phân bố số dấu chấm động với $\beta = 4,\ s = 1,\ m = -1,\ M = 1$.

Do tập hợp số dấu chấm động, ký hiệu F, là hữu hạn nên một số dấu chấm động biểu diễn (xấp xỉ) nhiều số thực. Khi số mũ e trong (1.5) lớn hơn M thì x không thể được biểu diễn trong hệ thống dấu chấm động này. Nếu trong quá trình tính toán xuất hiện số với $e > M$ thì ta nói tính toán đã overflow. Các hệ điều hành khi gặp trường hợp này sẽ dừng lại việc tính toán. Trường hợp khi $e < m$, một cách tự nhiên, ta có thể xấp xỉ x bằng không. Khi tính toán gặp số như vậy ta nói tính toán đã underflow. Một số hệ điều hành, trong trường hợp này, gán cho kết quả bằng không rồi tiếp tục, một số khác thì dừng chương trình. Khi sử dụng các ngôn ngữ lập trình hoặc các phần mềm tính toán ta cần lưu ý đến hệ thống số dấu chấm động mà nó sử dụng để tránh những kết quả không đáng có.

Thí dụ 1.5. Một thuật toán đơn giản dùng để giải phương trình $f(x) = 0$ là thuật toán chia đôi khoảng. Trong thuật toán này cần phải xác định xem $f(a)$ và $f(b)$ có trái dấu không, nghĩa là $f(a)f(b) < 0$ đúng hay sai. Tuy nhiên, khi a hay b khá gần nghiệm của phương trình thì tích $f(a)f(b)$ có thể underflow và dấu của nó không xác định! ∘

Thí dụ 1.6. Định thức của ma trận chéo được tính theo công thức

$$\det = a_1 a_2 \cdots a_n.$$

Trong nhiều trường hợp khi dùng công thức này ta gặp underflow hay overflow. Chẳng hạn, với hệ thống dấu chấm động $M = 100$, $a_1 = 10^{50}$, $a_2 = 10^{60}$, $a_3 = 10^{-30}$, tất cả các số a_i còn lại đều thuộc phạm vi biểu diễn

được và $det = 10^{80}$. Nhưng $(a_1 \times a_2) \times a_3$ là overlow trong khi $a_1 \times (a_2 \times a_3)$ lại thực hiện được. Điều này cho thấy các số dấu chấm động không tuân theo luật kết hợp của phép nhân trong hệ thống số thực.

Một vấn đề cũng cơ bản không kém là sự "nhạy cảm" của định thức khi nhân ma trận với một số. Nếu A là ma trận vuông cấp n thì $\det(cA) = c^n \det(A)$, khi n lớn điều này có thể gây ra overflow hay underflow.

Để giải quyết vấn đề ở đây, phần mềm LINPACK mở rộng phạm vi hiệu lực của số mũ. Một cách làm khác là dùng hàm lôgarit và hàm mũ

$$\begin{aligned} \ln|\det| &= \sum_{i=1}^{n} \ln|a_i| \\ \det &= \exp(\ln|\det|). \end{aligned}$$

Nếu điều này dẫn đến overflow thì là vì bản thân kết quả không biểu diễn được trong hệ thống số dấu chấm động ∘

Thí dụ 1.7. Khi tính môđun của số phức $z = x + iy$,

$$|z| = \sqrt{x^2 + y^2},$$

ta gặp trở ngại khi x hay y lớn. Giả sử $|x| \geq |y|$. Nếu x đủ lớn, x^2 sẽ overflow và ta không thể tính được $|z|$ ngay cả trường hợp nó là điểm dấu chấm động có hiệu lực. Nếu tính toán được thực hiện như sau

$$|z| = |x|\sqrt{1 + \left(\frac{y}{x}\right)^2},$$

chướng ngại được vượt qua.

Việc đánh giá chuẩn Euclide của vectơ $v = (v_1, v_2, \ldots, v_n)$,

$$\|v\|_2 = \left(\sum_{i=1}^{n} v_i^2\right)^{0.5},$$

cùng một kiểu tính toán như trên. Một số người lập trình phần mềm toán học đề nghị dùng chuẩn maximum

$$\|v\|_\infty = \max_{1 \leq i \leq n} |v_i|,$$

vì nó tránh được các overflow và underflow. ∘

Nếu một số thực x có số mũ trong phạm vi cho phép thì có hai cách xấp xỉ x thành số dấu chấm động, ký hiệu $\mathrm{fl}(x)$. Nếu $\mathrm{fl}(x)$ là kết quả của việc loại bỏ tất cả các chữ số sau s chữ số đầu trong (1.5) thì $\mathrm{fl}(x)$ được gọi là biểu diễn chặt cụt (chopped, truncated) của x. Nếu cộng thêm $5 \times 10^{-(s+1)}$ vào (1.5) rồi chặt cụt thì ta được một số dấu chấm động gần với x hơn (biểu diễn chặt cụt). Xác định $\mathrm{fl}(x)$ theo cách này gọi là làm tròn số (rounding).

Thí dụ 1.8. Nếu $m = -99$, $M = 99$, $s = 5$ và $\pi = 3.1415926\ldots$ thì số chặt cụt

$$\mathrm{fl}(\pi) = 0.31415 \times 10^1$$

trong khi

$$\mathrm{fl}(\pi) = 0.31416 \times 10^1$$

là số làm tròn ∘

Nếu dùng biểu diễn chặt cụt thì sai số tương đối của $\mathrm{fl}(x)$:

$$\begin{aligned}\left|\frac{x - \mathrm{fl}(x)}{x}\right| &= \frac{0.00\ldots0d_{s+1}d_{s+2}\ldots \times 10^e}{0.d_1d_2\ldots d_sd_{s+1}d_{s+2}\ldots \times 10^e} \\ &\leq \frac{0.00\ldots099\ldots}{0.10\ldots000\ldots} \\ &\leq \frac{0.00\ldots100\ldots}{0.10\ldots000\ldots} = 10^{1-s}.\end{aligned}$$

Trong hệ thập phân, khi phép chặt cụt được dùng, con số 10^{1-s} được định nghĩa là *đơn vị làm tròn* (unit roundoff), ký hiệu u. Nếu dùng phép làm tròn thì

$$\left|\frac{x - \mathrm{fl}(x)}{x}\right| \leq \frac{1}{2}10^{1-s}$$

và $u = 0.5 \times 10^{1-s}$.

Số u là chặn trên của sai số tương đối trong phép biểu diễn dấu chấm động của một số khác không. Nếu biểu diễn

$$\mathrm{fl}(x) = x(1 + \delta)$$

thì $\delta \leq u$.

máy	β	s	m	M	u
VAX	2	24	-128	127	6.0×10^{-08}
VAX	2	48	-128	127	1.4×10^{-17}
CRAY-1	2	56	-16384	16383	3.6×10^{-15}
IBM 3081	16	6	-64	63	9.5×10^{-07}
IBM 3081	16	14	-64	63	2.2×10^{-16}
IEEE					
single	2	24	-125	128	6.0×10^{-08}
double	2	53	-1021	1024	1.1×10^{-16}

Bảng 1.1: Các thí dụ về hệ thống số dấu chấm động.

Thí dụ 1.9. Các chương trình hiện nay tìm nghiệm của phương trình, tính tích phân xác định, giải phương tình vi phân v.v. ..., thường cho phép người sử dụng chỉ định độ chính xác. Rõ ràng là không thể có lời giải chính xác hơn biểu diễn dấu chấm động của nghiệm đúng. Điều này có nghĩa là người sử dụng không nên yêu cầu sai số tương đối nhỏ hơn đơn vị làm tròn u. Nghe có vẻ kỳ quặc, điều này vẫn xảy ra! Lý do là người sử dụng không biết giá trị của u nên đưa ra yêu cầu quá đáng về độ chính xác. Một lý do thường thấy hơn là người sử dụng chỉ định sai số tuyệt đối r. Điều này có nghĩa là số bất kỳ y^* sẽ được chấp nhận như là một xấp xỉ của y nếu

$$|y - y^*| \leq r.$$

Yêu cầu như vậy tương ứng với đòi hỏi sai số tương đối

$$\left| \frac{y - y^*}{y} \right| \leq \frac{r}{|y|}.$$

Khi $|r/y| < u$, nghĩa là, $r < u|y|$, yêu cầu này không chấp nhận được. Nếu nghiệm đúng là cực lớn, một chặn sai số tuyệt đối "vừa phải" thường là không thể trong thực hành. Các chương trình cho phép người dùng chỉ định chặn sai số tuyệt đối cần phải có thể giám sát độ lớn của nghiệm và đưa ra khuyến cáo cho người dùng khi yêu cầu về độ chính xác là không thể. ∘

Các cơ số $\beta = 2$ (nhị phân), $\beta = 16$ (thập lục phân) thường được dùng trong máy tính hơn là cơ số 10. Các trình bày trên có thể thực hiện cho cơ số β bất kỳ.

Định lý 1.1. *Trong hệ thống số dấu chấm động F cơ số β giữ s chữ số với $m < e < M$, ký hiệu $F = F(\beta, s, m, M)$, mọi số thực trong phạm vi của F có thể*

được biểu diễn với sai số tương đối không vượt quá đơn vị làm tròn u,

$$u = \begin{cases} \beta^{1-s}, & \textit{chặt cụt} \\ 0.5\beta^{1-s}, & \textit{làm tròn}. \end{cases} \tag{1.7}$$

1.2.2 Thuật toán chuyển đổi giữa các hệ thống số

Cho a là số nguyên trong hệ thống số với cơ số α. Ta cần xác định biểu diễn của nó trong hệ thống số với cơ số β:

$$a = b_n\beta^n + b_{n-1}\beta^{n-1} + \ldots + b_0, \quad 0 \le b_i < \beta. \tag{1.8}$$

Các phép tính trong (1.8) phải được thực hiện trong hệ thống số với cơ số α và cũng vậy β được biểu diễn trong hệ thống số này. Sự chuyển đổi được thực hiện bằng cách chia liên tiếp của a cho β:

Đặt $q_0 = a$, và

$$q_k = q_{k+1}\beta + b_k, \quad k = 0, 1, \ldots. \tag{1.9}$$

(q_{k+1} là thương còn b_k là dư trong phép chia.)

Nếu a không là số nguyên, ta viết $a = b + c$, trong đó b là phần nguyên (integer part) và

$$c = b_{-1}\beta^{-1} + b_{-2}\beta^{-2} + \ldots \tag{1.10}$$

là phần phân số (fractional part) phải được xác định. Các chữ số này nhận được như phần nguyên (của kết quả) khi nhân liên tiếp c với β:

Đặt $p_{-1} = c$, và

$$p_k \cdot \beta = b_k + p_{k-1}, \quad k = -1, -2, \ldots. \tag{1.11}$$

Vì một phân số hữu hạn trong hệ thống số với cơ số β thường không tương ứng với một phân số *hữu hạn* trong hệ thống số với cơ số β nên cần thiết phải làm tròn.

Khi chuyển đổi bằng tay giữa hệ thập phân và, chẳng hạn, hệ nhị phân tất cả các phép tính được làm trong hệ thập phân ($\alpha = 10$ và $\beta = 2$). Nếu ngược lại, sự chuyển đổi được thực hiện trên một máy nhị phân, các phép tính được làm trong hệ nhị phân ($\alpha = 2$ và $\beta = 10$).

Thí dụ 1.10. Chuyển đổi số thập phân 176.524 thành dạng tam phân (ternary) (cơ số $\beta = 3$). Với phần nguyên ta có $173/3 = 58$ dư 2; $58/3 = 19$ dư 1; $19/3 = 6$ dư 1; $6/3 = 2$ dư 0; $2/3 = 0$ dư 2. Như vậy, $(176)_{10} = (20112)_3$.

Với phần phân số ta tính $.524 \times 3 = 1.572$, $.572 \times 3 = 1.716$, $.716 \times 3 = 2.148$, ... Tiếp tục ta nhận được $(.524)_{10} = (.112010222\ldots)_3$. Số thập phân hữu hạn không tương ứng với phân số hữu hạn trong hệ tam phân! ∘

1.2.3 Số học dấu chấm động

Số chấm động cũng là số thực. Các kết quả khi thực hiện phép $+$, $-$, $\times$, $/$ trong hệ thống số dấu chấm động luôn kèm theo một phép "làm tròn" nào đó, nghĩa là xấp xỉ kết quả nhận được của phép tính tương ứng trong hệ thống số thực. Sau này ta dùng các ký hiệu $\oplus$, $\ominus$, $\otimes$, $\oslash$ để chỉ xấp xỉ dấu chấm động (floating point approximation) của các phép tính $+$, $-$, $\times$, $/$ trong hệ thống số thực. Ta giả sử các thủ tục số học của phần cứng sinh ra các kết quả thỏa

$$\begin{aligned} x \oplus y &= \mathrm{fl}(x + y), \\ x \ominus y &= \mathrm{fl}(x - y), \\ x \otimes y &= \mathrm{fl}(x \times y), \\ x \oslash y &= \mathrm{fl}(x/y) \end{aligned}$$

miễn là kết quả tính nằm trong phạm vi hệ thống số dấu chấm động. Mô hình phần cứng thỏa các điều kiện trên gọi là *mô hình chuẩn*.

Như vậy, trong mô hình chuẩn, với $x, y \in F$, ta có

$$\mathrm{fl}(x \text{ op } y) = (x \text{ op } y)(1 + \delta), \quad |\delta| \le u, \tag{1.12}$$

trong đó u là đơn vị làm tròn và "op" thay cho một trong bốn phép tính $+$, $-$, $\cdot$ và $/$.

Để thực hiện các phép tính trong mô hình số học này bằng tay, với mỗi phép tính $+, -, \times, /$, thực hiện phép toán bằng số học chính xác, chuẩn hóa kết quả, và làm tròn (chặt cụt) nó. Dùng một cách khác, với mỗi phép tính, tính kết quả và chuyển đổi nó thành biểu diễn trong máy trước khi tiếp tục phép tính kế tiếp.

1.3 Các thí dụ tính toán số dấu chấm động

Hệ thống số dấu chấm động có các tính chất như hệ thống số thực nhưng không phải tất cả. Ta sẽ thấy phép nhân và phép chia thỏa mãn các tính

chất của hệ thống số thực tốt hơn phép cộng và phép trừ.

Với $x, y, z \in F$,

$$\begin{aligned} x \otimes y &= xy(1+\delta_1), \\ (x \otimes y) \otimes z &= (xy(1+\delta_1))z(1+\delta_2) \\ &= xyz(1+\delta_1)(1+\delta_2). \end{aligned}$$

Tích

$$(1+\delta_1)(1+\delta_2) = 1+\epsilon,$$

trong đó ϵ là "nhỏ" và có thể đánh giá so với đơn vị làm tròn u

$$(1+\delta_1)(1+\delta_2) = 1+\delta_1+\delta_2+\delta_1\delta_2 \approx 1+\delta_1+\delta_2$$

suy ra

$$\epsilon \approx \delta_1 + \delta_2$$

và chặn trên của ϵ là $2u$. Trước khi tổng quát hóa kết quả này, ta lưu ý rằng có thể xảy ra trường hợp mà

$$x \otimes (y \otimes z) \neq (x \otimes y) \otimes z,$$

ngay cả khi số mũ không vượt quá phạm vi. Tuy nhiên

$$x \otimes (y \otimes z) = xyz(1+\delta_3)(1+\delta_4)$$

suy ra

$$\frac{x \otimes (y \otimes z)}{(x \otimes y) \otimes z} = \frac{(1+\delta_3)(1+\delta_4)}{(1+\delta_1)(1+\delta_2)} = 1+\eta,$$

trong đó η là "nhỏ". Như vậy, luật kết hợp cho phép nhân là đúng một cách xấp xỉ.

Trong trường hợp tổng quát, nếu ta muốn nhân $x_1, x_2, \ldots, x_n$ ta có thể làm bằng thuật toán lặp

$$\begin{aligned} P_1 &= x_1 \\ P_i &= P_{i-1} \otimes x_i, \quad i = 2, 3, \ldots, n. \end{aligned}$$

Tính trong hệ thống số thực ta có

$$P_i = x_1 x_2 \cdots x_i = (1+\delta_1)(1+\delta_2)\cdots(1+\delta_i),$$

trong đó các $|\delta_i| \leq u$. Sai số tương đối của mỗi P_i có thể được chặn nhờ u không khó, nếu dùng xấp xỉ

$$P_i \approx x_1 x_2 \cdots x_i (1 + \delta_1 + \delta_2 + \ldots + \delta_i),$$

thì

$$|\delta_1 + \delta_2 + \ldots + \delta_i| \leq iu.$$

Điều này cho thấy chặn trên sai số tương đối phát triển một cách cộng dồn. Mỗi phép nhân làm gia tăng sai số tương đối một lượng không nhiều hơn đơn vị làm tròn. Phép chia có thể phân tích theo cùng một cách và kết luận cũng tương tự (xem bổ đề 1.2).

Thí dụ 1.11. Hàm gamma, định nghĩa như là

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt,$$

tổng quát hóa hàm giai thừa (factorial function) cho các số nguyên lên các số thực x (và cũng cho số phức x). Từ công thức truy hồi cơ bản

$$\Gamma(x) = (x-1)\Gamma(x-1) \tag{1.13}$$

và sự kiện là $\Gamma(1) = 1$. Một phương pháp chuẩn để xấp xỉ $\Gamma(x)$ với $x \geq 2$ là dùng công thức trên dẫn về việc xấp xỉ $\Gamma(y)$ với $2 \leq y \leq 3$. Điều này được thực hiện bằng cách cho N là một số nguyên sao cho $N \leq x < N + 1$, bằng cách cho $y = x - N + 2$, và rồi chú ý rằng áp dụng liên tiếp (1.13) ta được

$$\Gamma(x) = \Gamma(y)(x - N + 2)(x - N + 3) \cdots (x - 2)(x - 1).$$

Hàm $\Gamma(y)$ có thể xấp xỉ tốt bởi tỉ số $R(y)$ của hai đa thức với $2 \leq y \leq 3$. Suy ra, ta xấp xỉ

$$\Gamma(x) \approx R(y)(x - N + 2) \cdots (x - 1).$$

Nếu x không quá lớn, một chút chính xác bị mất khi các phép nhân này được thực hiện trong số học dấu chấm động. Tuy nhiên không thể tính $\Gamma(x)$ với x lớn bằng cách tiếp cận này bởi vì giá trị của nó phát triển rất nhanh như là hàm của x. Điều này có thể thấy từ công thức Stirling

$$\Gamma(x) \approx \sqrt{2\pi/x} \left(\frac{x}{e}\right)^x.$$

Thí dụ này cho thấy một vấn đề khác: do số học dấu chấm động tự động đối xử với các số có độ lớn khác nhau. Nhưng nhiều hàm đặc biệt của vật lý toán lại phát triển hoặc suy giảm rất nhanh, thường là ra ngoài phạm vi số mũ. Khi điều này xảy ra cần thiết viết lại công thức cho bài toán để nhận được kết quả tốt hơn. Chẳng hạn, thường ta làm việc với hàm đặc biệt $\ln \Gamma(x)$ hơn là với $\Gamma(x)$. ∘

Phép cộng và phép trừ rất ít thỏa mãn trong số học dấu chấm động so với phép nhân và phép chia. Khi các số có độ lớn rất khác nhau được cộng (hay được trừ), một số thông tin có thể bị mất. Thí dụ ta muốn cộng 0.123456×10^{-4} với 0.100000×10^{1} trong số học chặt cụt sáu-chữ số. Trước hết, số mũ được điều chỉnh để trở nên giống nhau và rồi các số được cộng lại

$$\begin{array}{rll} & 0.100000 & \times 10^{1} \\ + & 0.00000123456 & \times 10^{1} \\ \hline & 0.10000123456 & \times 10^{1}. \end{array}$$

Kết quả được chặt cụt thành 0.100001×10^{1}. Chú ý rằng một vài chữ số đã không tham gia vào phép cộng. Thật vậy, nếu $|y| < |x|u$, thì $x \oplus y = x$ và số y chẳng đóng vai trò gì. Sự mất mát thông tin không có nghĩa là đáp số không chính xác; thật ra nó chính xác đến một đơn vị làm tròn. Vấn đề là thông tin bị mất này có thể cần đến cho các tính toán về sau.

Thí dụ 1.12. Ta có công thức xấp xỉ

$$F'(x) \approx \frac{F(x+\delta) - F(x)}{\delta}.$$

Vế phải của công thức này được gọi là tỉ sai phân (difference quotient) của hàm F tại điểm x. Trong nhiều áp dụng công thức này được dùng để xấp xỉ $F'(x)$. Để có xấp xỉ chính xác, δ phải "nhỏ" so với x. Tốt hơn nó không nên quá nhỏ (để có sự chính xác) nếu không ta sẽ có $x \oplus \delta = x$ và giá trị tính của $F'(x)$ sẽ bằng không. Nếu δ đủ lớn để ảnh hưởng đến tổng nhưng vẫn "nhỏ", một vài chữ số của nó (δ) sẽ không ảnh hưởng đến tổng theo nghĩa $x \oplus \delta - x \neq \delta$. Trong tỉ sai phân ta cần chia cho hiệu thực của đối số chứ không phải là δ, như vậy nên định nghĩa

$$\Delta = (x \oplus \delta) \ominus x$$

và xấp xỉ

$$F'(x) \approx \frac{F(x+\Delta) - F(x)}{\Delta}.$$

Hai xấp xỉ của $F'(x)$ ở trên là tương đương về mặt toán học, nhưng về mặt tính toán thì khác. Thí dụ, giả sử $F(x) = x$ và ta xấp xỉ $F'(x)$ tại $x = 1$ bằng cách dùng $\delta = 0.123456 \times 10^{-4}$ trong số học chặt cụt sáu-chữ số. Ta có $1 \oplus \delta = 0.100001 \times 10^{1}$; tương tự, $\Delta = 0.100000 \times 10^{-4}$ chỉ thể hiện những chữ số của δ có ảnh hưởng thực đến tổng. Công thức đầu tiên cho

$$\frac{(1 \oplus \delta) \ominus 1}{\delta} = \frac{0.100000 \times 10^{-4}}{0.123456 \times 10^{-4}} = 0.810000 \times 10^{0}.$$

Công thức thứ hai cho

$$\frac{(1 \oplus \delta) \ominus 1}{\Delta} = \frac{0.100000 \times 10^{-4}}{0.100000 \times 10^{-4}} = 0.100000 \times 10^{1}.$$

Hiển nhiên công thức thứ hai cho xấp xỉ tốt hơn $F'(1) = 1$. ∘

Thí dụ 1.13. Xấp xỉ tích phân xác định

$$\int_a^b f(x)dx$$

có thể được thực hiện bằng cách phân hoạch $[a, b]$ thành những đoạn con $[\alpha, \beta]$ và xấp xỉ tích phân trên mỗi đoạn con này. Giả sử ta dùng công thức

$$\int_\alpha^\beta f(x)dx \approx \frac{\beta - \alpha}{6}\left[f(\alpha) + 4f\left(\frac{\alpha+\beta}{2}\right) + f(\beta)\right].$$

Độ chính xác của công thức này phụ thuộc vào độ dài $|\beta - \alpha|$, độ dài càng nhỏ thì càng chính xác. Tuy nhiên, nếu $|\beta - \alpha| < 2u|\alpha|$, các số dấu chấm động α và $\alpha + (\beta - \alpha)/2$ là giống nhau. Nếu α và β không thể được phân biệt trong độ chính xác (đang dùng), các kết quả tính sẽ không giống các kết quả toán học trong hệ thống số thực. Trong trường hợp này người sử dụng phần mềm phải được cảnh báo là sự chính xác như yêu cầu là không khả thi. ∘

Thí dụ 1.14. Tổng S của chuỗi số

$$\sum_{m=1}^{\infty} a_m$$

là giới hạn của dãy các tổng riêng

$$S_n = \sum_{m=1}^{n} a_m.$$

Một thuật toán tự nhiên đế tính S:

$$\begin{aligned} S_1 &= a_1, \\ S_n &= S_{n-1} \oplus a_n \quad n = 2, 3, \ldots, \end{aligned}$$

tiếp tục cho đến khi tổng riêng không còn thay đổi. Một thí dụ cổ điển về chuỗi phân kỳ là chuỗi điều hòa

$$\sum_{m=1}^{\infty} \frac{1}{m}.$$

Nếu thuật toán trên được áp dụng cho chuỗi điều hòa, giá trị tính toán S_n tăng và $a_n = 1/n$ giảm cho đến khi

$$S_n = S_{n-1} \oplus a_n = S_n$$

và thuật toán dừng! Trong số học dấu chấm động chuỗi phân kỳ này có tổng hữu hạn! Như vậy cần phải có vài phân tích toán học phụ thêm (khi tính tổng của chuỗi số) để có kết quả tin cậy. ∘

Thí dụ 1.15. Hai số có các chữ số đầu giống nhau thì phép trừ giữa chúng sẽ khử đi các chữ số này. Thí dụ, nếu $x = 0.123654 \times 10^{-5}$ và $y = 0.123456 \times 10^{-5}$, thì

$$\begin{array}{rll} & 0.123654 & \times 10^{-5} \\ - & 0.123456 & \times 10^{-5} \\ \hline & 0.000198 & \times 10^{-5} \end{array} = 0.198000 \times 10^{-8}.$$

Điều đáng quan tâm là khi sự khử được thực hiện, phép trừ được thực hiện chính xác, $x \ominus y = x - y$. Nhưng có sự "mất mát" thông tin, được gọi là *sự mất ý nghĩa* (loss of significane). Khi sự khử xảy ra, kết quả $x - y$ là nhỏ hơn x và y về độ lớn, vì vậy các sai số *đã hiện diện* trong x và y là tương đối lớn so với $x - y$. Giả sử x là một xấp xỉ của X và y là một xấp xỉ của Y. Chúng có thể là các giá trị đo hay kết quả của một vài tính toán. Hiệu $x - y$ là một xấp xỉ của $X - Y$ với sai số tương đối thỏa

$$\begin{aligned}\left|\frac{(x-y)-(X-Y)}{X-Y}\right| &= \left|\frac{(x-X)-(y-Y)}{X-Y}\right| \\ &\leq \left|\frac{x-X}{X}\right|\left|\frac{X}{X-Y}\right| + \left|\frac{y-Y}{Y}\right|\left|\frac{Y}{X-Y}\right|.\end{aligned}$$

Nếu x gần y đến độ có sự khử, sai số tương đối có thể lớn vì mẫu số $X - Y$ là nhỏ so với X hay Y. Thí dụ, nếu $X = 0.123654700\ldots \times 10^{-5}$, thì x giống X sai khác một đơn vị làm tròn trong số học sáu-chữ số. Với $Y = y$ giá trị ta tìm là $X - Y = 0.198700\ldots \times 10^{-8}$. Mặc dù hiệu $x - y = 0.198000 \times 10^{-8}$ được thực hiện cách chính xác, $x - y$ và $X - Y$ khác nhau ở số lẻ thứ bốn. Trong thí dụ này, x và y có ít nhất sáu chữ số có nghĩa, nhưng hiệu của chúng chỉ còn ba chữ số có nghĩa. ∘

Nhận xét 1.1. Một nhận xét có giá trị là ta đã dùng sự khử trong thí dụ 1.12 khi tính

$$\Delta = (x \oplus \delta) \ominus x.$$

Vì δ là nhỏ so với x, có sự khử và $\Delta = (x \oplus \delta) - x$. Theo cách này ta nhận được trong Δ các chữ số của δ thực sự ảnh hưởng đến tổng.

Thí dụ 1.16. Công thức tính nghiệm phương trình bậc hai

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

là

$$x_{1,2} = -\frac{b}{2} \pm \sqrt{\left(\frac{b}{2}\right)^2 - c},$$

giả sử $b \geq 0$. Nếu c nhỏ so với b, căn bậc hai có thể viết lại và xấp xỉ bằng cách dùng chuỗi nhị thức

$$\frac{b}{2}\sqrt{1 - \frac{4c}{b^2}} \approx \frac{b}{2}\left(1 - \frac{2c}{b^2} + \ldots\right).$$

Điều này chứng tỏ các nghiệm thực

$$\begin{aligned} x_1 &\approx -b, \\ x_2 &\approx -c/b. \end{aligned}$$

Trong số học có độ chính xác hữu hạn một vài chữ số của c không ảnh hưởng lên tổng $(b/2)^2 - c$. Trường hợp tới hạn là

$$\left(\frac{b}{2}\right)^2 \ominus c = \left(\frac{b}{2}\right)^2.$$

Điều quan trọng để nhận thức đúng là đại lượng này được tính một cách chính xác theo nghĩa tương đối. Tuy nhiên, một vài thông tin bị mất và ta sẽ thấy trong vài trường hợp ta cần đến nó trong tính toán sau này. Một căn bậc hai được tính với một sai số tương đối nhỏ cũng đúng với phép trừ theo sau. Vậy thì nghiệm lớn hơn $x_1 \approx -b$ được tính chính xác bởi công thức ở trên. Trong tính toán nghiệm nhỏ hơn, có sự khử khi số hạng căn bậc hai bị trừ từ $-b/2$. Bản thân phép trừ được thực hiện chính xác, nhưng sai số đã hiện diện trong $(b/2)^2 \ominus c$ trở nên quan trọng theo nghĩa tương đối. Trong trường hợp tới hạn công thức cho kết quả bằng không như một xấp xỉ của x_2. Sắp xếp lại công thức tính có thể tránh được khó khăn này. Dùng công thức $x_1 x_2 = c$, nghiệm x_2 có thể được tính bằng

$$x_2 = c/x_1$$

cho giá trị chính xác hơn. ∘

Thí dụ 1.17. Muốn tính tổng của một chuỗi, điều quan trọng là biết khi nào đã lấy đủ các số hạng từ chuỗi để xấp xỉ giới hạn (tổng của chuỗi) với độ chính xác mong muốn. Chuỗi đan dấu thu hút sự chú ý này. Giả sử $a_0 \geq a_1 \geq \ldots \geq a_n \geq a_{n+1} \geq \ldots \geq 0$. Thì chuỗi đan dấu

$$\sum_{m=0}^{\infty} (-1)^m a_m$$

hội tụ tới giới hạn S và sai số của tổng riêng

$$S_n = \sum_{m=0}^{n} (-1)^m a_m$$

thỏa

$$|S - S_n| \leq a_{n+1}.$$

Xem một trường hợp cụ thể, đánh giá $\sin(x)$ bằng chuỗi Maclaurin của nó

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots.$$

Mặc dầu chuỗi này hội tụ nhanh với x cho trước bất kỳ, có khó khăn số khi $|x|$ lớn. Nếu, chẳng hạn, $x = 10$, a_m là

$$10, \frac{10^3}{6}, \frac{10^5}{120}, \frac{10^7}{5040}, \ldots.$$

Rõ ràng có một vài số hạng thực sự lớn cần phải loại để nhận được kết quả $\sin 10$ có độ lớn không quá 1. Các số hạng a_m là kết quả của một số phép tính ở đây có thể nhận được với sai số tương đối nhỏ. Tuy nhiên, nếu a_m là lớn so với tổng S, một sai số tương đối nhỏ trong a_m sẽ không nhỏ so với S và S sẽ không được tính chính xác.

Người ta đã lập trình đánh giá chuỗi này bằng một cách trực tiếp, cẩn thận tính, cụ thể,

$$-\frac{x^7}{7!} = -\left(\frac{x^5}{5!}\right)\left(\frac{x}{6}\right)\left(\frac{x}{7}\right),$$

để tránh những con số lớn không cần thiết. Bằng cách dùng số học dấu chấm động chính xác đơn tiêu chuẩn IEEE người ta cộng các số hạng cho đến khi các tổng riêng không còn thay đổi. Cách tính này cho giá trị -0.544040 trong khi giá trị chính xác là -0.544021. Do phạm vi số mũ trong chính xác đơn là nhỏ nên ta sẽ gặp trường hợp overflow nếu cố gắng tính với $x = 100!$ Rõ ràng số học dấu chấm động không thoát khỏi tất cả những điều có liên quan về độ lớn.

Các chuỗi thường được dùng như một cách đánh giá giá trị các hàm số. Nếu giá trị của hàm cần tính là nhỏ và nếu một vài số hạng trong chuỗi tương đối lớn, thì phải được loại bỏ và ta phải nghĩ rằng sự không chính xác trong tính toán các số hạng sẽ làm cho giá trị của hàm không chính xác theo một nghĩa tương đối. ∘

Ta đã thấy các thí dụ chứng tỏ tổng của nhiều số phụ thuộc vào thứ tự trong đó chúng được cộng. Như vậy thứ tự nào là "tốt"? Trở lại với thuật

toán trong thí dụ 1.14 để tính tổng $a_1 + a_2 + \ldots + a_N$. Trước hết tổng riêng thứ nhất là

$$\begin{aligned} \mathrm{fl}(S_2) &= a_1 \oplus a_2 = (a_1 + a_2)(1 + \delta_2) \\ &= S_2 + \delta_2 a_1 + \delta_2 a_2, \end{aligned}$$

trong đó $|\delta_2| \leq u$. Tiếp tục,

$$\begin{aligned} \mathrm{fl}(S_3) &= \mathrm{fl}(S_2) \oplus a_3 = (\mathrm{fl}(S_2) + a_3)(1 + \delta_3) \\ &= S_3 + (\delta_2 + \delta_3)a_1 + (\delta_2 + \delta_3)a_2 + \delta_3 a_3 + \delta_2\delta_3 a_1 + \delta_2\delta_3 a_2, \end{aligned}$$

trong đó $|\delta_3| \leq u$. Bỏ qua các số hạng chứa các tích của các thừa số bé,

$$\mathrm{fl}(S_3) = S_3 + (\delta_2 + \delta_3)a_1 + (\delta_2 + \delta_3)a_2 + \delta_3 a_3.$$

Cuối cùng ta tìm được

$$\begin{aligned} \mathrm{fl}(a_1 + \ldots + a_N) \approx\ & (a_1 + \ldots + a_N) \\ & +(\delta_2 + \delta_3 + \ldots + \delta_N)a_1 \\ & +(\delta_2 + \delta_3 + \ldots + \delta_N)a_2 \\ & +(\delta_3 + \delta_4 + \ldots + \delta_N)a_3 \\ & + \ldots + \delta_N a_N. \end{aligned}$$

Theo xấp xỉ này, sai số phát sinh khi a_k được cộng vào S_k có thể tăng lên nhưng ảnh hưởng của nó trong S sẽ không lớn hơn $(N - k + 1)u|a_k|$. Điều này đề nghị rằng để giảm sai số toàn phần, các số hạng nên được cộng theo thứ tự độ lớn tăng.

Sai số xấp xỉ có thể ước lượng bởi

$$|\mathrm{fl}(S_N) - S_N| \subseteq Nu \sum_{n=1}^{N} |a_n|.$$

Ở đây ta dùng ký hiệu $\subseteq$ có nghĩa là "nhỏ hơn hay bằng với một đại lượng mà là một cách xấp xỉ". Sai số tương đối của tổng

$$\frac{|\mathrm{fl}(S_N) - S_N|}{|S_N|} \subseteq Nu \frac{\sum_{n=1}^{N} |a_n|}{|\sum_{n=1}^{N} a_n|}.$$

Trường hợp nguy hiểm là khi $|\sum_{n=1}^{N} a_n| \ll \sum_{n=1}^{N} |a_n|$, lúc đó xảy ra sự khử. Một hệ quả quan trọng là nếu tất cả các số hạng có cùng dấu, tổng sẽ được

tính chính xác theo nghĩa tương đối, miễn là số các số hạng không quá lớn để sự chính xác có hiệu lực.

Với chuỗi hội tụ

$$S = \sum_{m=0}^{\infty} a_m$$

thì $|a_m| \to 0$ khi $m \to \infty$. Trước hết ta lấy tổng theo thứ tự tự nhiên $m = 0, 1, \ldots$, để chọn số số hạng cần thiết N để tính tổng rồi tính S_N theo thứ tự đảo $m = N, N-1, \ldots, 0$.

Thí dụ 1.18. Cho $f(x) = x^2 - 2x + 1$ được đánh giá tại $x = 1.018$ với số học chặt cụt 3-chữ số và $-100 < e < 100$. Đáp số chính xác là $f(1.018) = 0.324 \times 10^{-3}$. Vì các hệ số của f là số nguyên bé, không có sai khi biểu diễn như là số dấu chấm động. Tuy nhiên, với x thì khác, $\mathrm{fl}(x) = 0.101 \times 10^{1}$, phép biểu diễn có sai số. Có nhiều thuật toán để tính $f(x)$:

$$\begin{aligned} f(x) &= [(x^2) - (2x)] + 1, \\ f(x) &= x(x-2) + 1, \\ f(x) &= (x-1)^2. \end{aligned}$$

Các dạng này cho:

$$\begin{aligned} y_1 &= [(x \otimes x) \ominus (2 \otimes x)] \oplus 1 = 0.000 \times 10^{-100}, \\ y_2 &= x \otimes (x \ominus 2) \oplus 1 = 0.100 \times 10^{-2}, \\ y_3 &= (x \ominus 1) \otimes (x \ominus 1) = 0.100 \times 10^{-3}. \end{aligned}$$

Tất cả các kết quả có sai số tương đối lớn. Đó là vì bài toán là điều kiện xấu. ∘

1.4 Ảnh hưởng của sai số làm tròn - sự truyền sai số

Trong tính toán các bài toán khoa học, dữ liệu nhập thường không chính xác, sai số trong dữ liệu nhập được truyền đi trong quá trình tính toán gây ra sai số trong dữ liệu xuất. Ngoài ra sai số làm tròn ở mỗi bước tính cũng được truyền đi và xuất hiện trong kết quả cuối cùng. Ảnh hưởng của sai số làm tròn lên kết quả cuối cùng có thể được đánh giá bằng cách dùng các bổ đề sau.

Bổ đề 1.1. *Trong phép cộng và phép trừ, cái chặn cho sai số tuyệt đối trong kết quả được cho bởi tổng của các chặn cho sai số tuyệt đối của các toán hạng*

$$y = \sum_{i=1}^{n} \pm x_i, \quad |\Delta y| \leq \sum_{i=1}^{n} |\Delta x_i|. \tag{1.14}$$

Để nhận được kết quả tương ứng cho phép nhân và chia, ta bắt đầu bằng nhận định rằng với $y = \ln(x)$ ta có $\Delta(\ln(x)) \approx \Delta x/x$. Phát biểu bằng lời: *sai số tương đối trong một đại lượng xấp xỉ bằng sai số tuyệt đối trong logarit tự nhiên của nó.* Từ nhận xét này ta có:

Bổ đề 1.2. *Trong phép nhân và phép chia, cái chặn xấp xỉ cho sai số tương đối được nhận bằng cách cộng các sai số tương đối của các toán hạng. Tổng quát hơn, cho* $y = x_1^{m_1} x_2^{m_2} \cdots x_n^{m_n}$,

$$\left|\frac{\Delta y}{y}\right| \lessapprox \sum_{i=1}^{n} |m_i| \left|\frac{\Delta x_i}{x_i}\right|. \tag{1.15}$$

Chứng minh. Chứng minh bằng cách lấy đạo hàm $\ln(y) = m_1 \ln(x_1) + m_2 \ln(x_2) + \ldots + m_n \ln(x_n)$ rồi đánh giá nhiễu trong từng số hạng ■

Thí dụ 1.19. Trong phương pháp Newton giải phương trình phi tuyến $f(x) = 0$ một hiệu chỉnh Δx_k (từ nghiệm xấp xỉ ở bước k, x_k) phải được tính như là tỉ số $y = f(x_k)/f'(x_k)$. Giả sử rằng $f(x_k)$ chỉ được biết với độ chính xác tương đối nào đó, ta nên tính $f'(x_k)$ chính xác thế nào để có độ chính xác cao hơn? Vì giới hạn của sai số tương đối trong y bằng tổng của các chặn cho sai số tương đối trong $f(x_k)$ và $f'(x_k)$, nên sẽ không có lợi nếu cố gắng đạt sai số tương đối trong $f'(x_k)$ "rất ít hơn" sai số tương đối trong $f(x_k)$ ∘

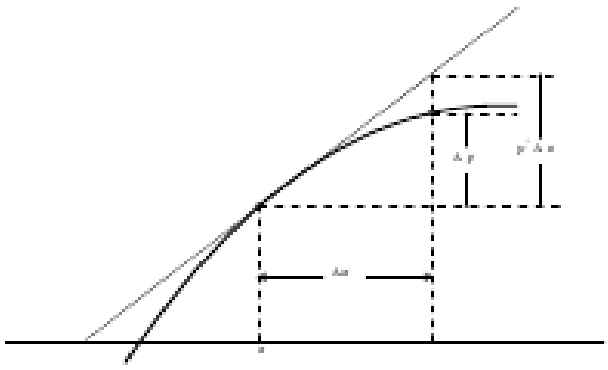
Bây giờ ta nghiên cứu sự truyền sai số trong các biểu thức phi tuyến tổng quát hơn. Giả sử ta cần tính hàm $y = f(x)$ của một biến độc lập x. Sai số trong x truyền tới y như thế nào? Cho $\Delta x = x - \tilde{x}$. Bởi định lý giá trị trung gian,

$$\Delta y = f(x) - f(\tilde{x}) = f'(\xi)\Delta x,$$

trong đó ξ nằm giữa x và $\tilde{x}$. Giả sử $|\Delta x| \leq \epsilon$ thì

$$|\Delta y| \leq \max_{\xi} |f'(\xi)|\epsilon, \quad \xi \in [x - \epsilon, x + \epsilon].$$

Trong thực hành, thường ta thay ξ bằng đánh giá có hiệu lực của x là đủ. Ngay cả nếu cần sự chính xác cao trong giá trị của $f(x)$, hiếm khi cần sự chính xác tương đối cao trong cái chặn sai số hoặc đánh giá sai số. Chỉ cần cẩn thận hơn khi ξ nằm trong lân cận của không điểm của $f'(x)$.



Hình 1.2: Sai số truyền trong hàm $y = f(x)$.

Trường hợp y là hàm ẩn của x xác định bởi phương trình $g(x, y) = 0$. Nếu $\partial g/\partial y \neq 0$ thì

$$|\Delta y| \leq \max_{\xi} \left| \frac{\partial g}{\partial x}(\xi) : \frac{\partial g}{\partial y}(\xi) \right| \epsilon, \quad \xi \in [x - \epsilon, x + \epsilon].$$

Thí dụ 1.20. Trong thí dụ 1.16 bên dưới, giải phương trình bậc hai

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

ta thấy nghiệm tính toán rất "nhạy cảm" với sai số của c. Thật vậy, đạo hàm phương trình $x^2 + bx + c = 0$, trong đó $x = x(c)$ đối với c ta được

$$(2ax + b)\frac{dx}{dc} + 1 = 0 \Rightarrow \frac{dx}{dc} = -\frac{1}{2ax + b}.$$

Lấy $x = x_1$ và dùng các hệ thức liên hệ giữa nghiệm $x_{1,2}$ với các hệ số a, b, c ta có

$$\frac{dx_1}{dc} = -\frac{c/a}{c(2x_1 + b/a)} \Rightarrow \frac{dx_1}{x_1} = -\frac{dc}{c}\frac{x_2}{x_1 - x_2}.$$

Điều này chứng tỏ khi $|x_1 - x_2| \ll |x_2|$ có thể rất nhạy cảm với nhiễu tương đối nhỏ trong c.

Khi $x_1 = x_2 = r$, phương trình có nghiệm kép, phân tích trên không còn hiệu lực. Tuy nhiên, nếu gọi x là nghiệm của phương trình với c bị nhiễu

$$ax^2 + bx + c + a\Delta c = 0,$$

thì (do $b^2 - 4ac = 0$, $r = -b/2a$)

$$(x - r)^2 + \Delta c = 0.$$

Phương trình này có nghiệm $x = r \pm \sqrt{\Delta c}$. ∘

Để phân tích sự truyền sai số trong một hàm nhiều biến $f = f(x_1, x_2, \ldots, x_n)$ ta cần một tổng quát hóa của định lý giá trị trung gian:

Định lý 1.2. *Giả sử hàm lấy giá trị thực f khả vi trong một lân cận của điểm x=$(x_1, x_2, \ldots, x_n)$, và cho $x = x + \Delta x$ là điểm nằm trong lân cận này. Thì tồn tại số θ, $0 \le \theta \le 1$, sao cho*

$$\Delta f = f(x + \Delta x) - f(x) = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}(x + \theta\Delta x)\Delta x_i .$$

Tương tự như trên ta dễ dàng chứng minh:

Định lý 1.3 (Công thức tổng quát cho sự truyền sai số)**.** *Giả sử hàm lấy giá trị thực f khả vi trong một lân cận của điểm x=$(x_1, x_2, \ldots, x_n)$ với sai số Δx_1, Δx_2,.., Δx_n. Thì ước lượng sau là có hiệu lực*

$$\Delta f \approx \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}\Delta x_i, \tag{1.16}$$

trong đó các đạo hàm riêng được đánh giá tại x.

Với sai số cực đại trong $f(x_1, x_2, \ldots, x_n)$ ta có chặn của xấp xỉ

$$|\Delta f| \lessapprox \sum_{i=1}^{n} \left|\frac{\partial f}{\partial x_i}\right| |\Delta x_i|. \tag{1.17}$$

Trong (1.17) $|\partial f/\partial x_i|$ là giá trị lớn nhất của trị tuyệt đối các đạo hàm riêng trong lân cận của điểm đã biết x. Trong hầu hết các trường hợp thực hành chỉ cần tính $\partial f/\partial x_i$ tại x rồi cộng thêm một lượng khoảng 5 đến 10 phần trăm cho an toàn.

Thí dụ 1.21. Tính các chặn sai số cho $f = x_1^2 - x_2$, trong đó $x_1 = 1.03 \pm 0.01$, $x_2 = 0.45 \pm 0.01$. Ta nhận được

$$\left|\frac{\partial f}{\partial x_1}\right| = |2x_1| \leq 2 \times 1.03 < 2.1, \quad \left|\frac{\partial f}{\partial x_2}\right| = |-1| = 1,$$

suy ra $\Delta f \leq 2.1 \times 0.01 + 1 \times 0.01 = 0.031$ hay $f = 1.061 - 0.450 \pm 0.032 = 0.611 \pm 0.032$; chặn sai số đã được nâng lên 0.001 vì sự làm tròn trong khi tính x_1^2. ∘

Trong giải tích số rất hiếm trường hợp yêu cầu cho các chặn sai số để bảo đảm toán học. Thường thì chỉ cần cho một đánh giá về cấp của độ lớn (order of magnitude) của dự báo sai số.

1.5 Số điều kiện

Như đã biết, một bài toán số là điều kiện xấu nếu dữ liệu nhập thay đổi một chút nhưng dữ liệu xuất lại thay đổi rất lớn. Vì vậy việc có một số đo cho sự nhạy cảm của dữ liệu xuất khi dữ liệu nhập thay đổi là rất hữu ích. Trong bài toán tính số giá trị hàm $y = f(x)$ ta có thể lấy $|f'(x)|$ làm số đo độ nhạy cảm của $f(x)$ đối với nhiễu Δx của x. Trong nhiều tài liệu, tỉ số giữa sai số tương đối trong $f(x)$ và x được dùng.

Định nghĩa 1.2. Giả sử $x \neq 0$ và $f(x) \neq 0$, thì số điều kiện κ cho bài toán tính giá trị hàm $y = f(x)$, bởi định nghĩa là

$$\kappa = \lim_{|\Delta x| \to 0} \frac{|f(x + \Delta x) - f(x)|}{|f(x)|} : \frac{|\Delta x|}{|x|} = |x| \frac{|f'(x)|}{|f(x)|}. \tag{1.18}$$

Ta nói bài toán tính $f(x)$ từ x là điều kiện xấu nếu κ "lớn" và điều kiện tốt nếu ngược lại.

Số điều kiện là một thuộc tính của bài toán số và không phụ thuộc vào thuật toán được dùng! Một bài toán điều kiện xấu có khó khăn nội tại khi

giải bằng bất kỳ thuật toán nào. Ngay cả nếu dữ liệu nhập là chính xác sai số làm tròn xuất hiện khi tính toán bằng số học dấu chấm động vẫn có thể gây ra nhiễu rất lớn trong kết quả cuối cùng.

Thí dụ 1.22. Xét hệ phương trình tuyến tính

$$\begin{bmatrix} 1 & \alpha \\ \alpha & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

trong đó α là dữ liệu nhập. Nghiệm chính xác là

$$x = 1/(1-\alpha^2),\ \ y = -\alpha/(1-\alpha^2).$$

Ma trận suy biến khi $\alpha = 1$, và bài toán tính x, y là điều kiện xấu khi $\alpha \approx 1$. Dùng phương trình (1.18) ta thấy số điều kiện khi tính x là

$$\kappa = \alpha x'(\alpha)/x(\alpha) = 2\alpha^2/|1-\alpha^2|.$$

Với $\alpha = 0.9950$ ta có bằng cách dùng phép khử Gauss và tính toán trong hệ thập phân bốn chữ số các giá trị tính được là

$$\bar{y} = -0.995/(1-0.9900) = -99.50,\ \ \bar{x} = 1 + 0.9950 \times 99.50 = 100.0025,$$

thay vì giá trị chính xác $y = -99.7494$, $x = 100.2506$. Chú ý rằng hai chữ số bị mất khi làm tròn trong mẫu số lúc tính y ($\alpha^2 = 0.990025 \to 0.9900$). Số điều kiện $\kappa = 198$ chỉ ra một cách chính xác rằng ta có thể mất đi sự chính xác của hai chữ số thập phân có nghĩa. ∘

Bây giờ ta xét bài toán số nhiều biến P trong đó dữ liệu xuất $y_j = f_j(x)$, $j = 1, 2, \ldots, m$ phụ thuộc vào dữ liệu nhập $x = (x_1, x_2, \ldots, x_n)$. Thì, bởi công thức truyền sai số tổng quát (1.17), ta có đánh giá sai số cực đại

$$|\Delta f_j| \lessapprox \sum_{i=1}^{n} \left|\frac{\partial f_j}{\partial x_i}\right| |\Delta x_i|. \tag{1.19}$$

Điều này cho ta một ma trận các số điều kiện (tương đối)

$$\kappa_{ij} = \left|\frac{\partial f_j}{\partial x_i}\right| \frac{|x_i|}{|y_j|} \quad i = 1, 2, \ldots, n,\ \ j = 1, 2, \ldots, m.$$

Thuận tiện hơn nếu dùng một số duy nhất để đo điều kiện của bài toán. Điều này có thể thực hiện được bằng cách dùng chuẩn

Định nghĩa 1.3. Số điều kiện κ của bài toán P với dữ liệu nhập $(x_1, x_2, \ldots, x_n)$ và dữ liệu xuất $(y_1, y_2, \ldots, y_m)$ là

$$\kappa(P) = \lim_{\epsilon \to 0} \sup \frac{1}{\epsilon} \frac{\|\tilde{y} - y\|}{\|y\|}, \quad \|\tilde{x} - x\| \leq \epsilon \|x\|. \tag{1.20}$$

Chú ý rằng $\kappa(P)$ là một hàm của dữ liệu nhập x và phụ thuộc vào cách chọn chuẩn trong không gian dữ liệu và không gian nghiệm. Với ϵ đủ bé ta có đánh giá

$$\|\tilde{y} - y\| \leq \kappa \epsilon \|y\| + O(\epsilon^2).$$

Kết quả là nghiệm sẽ có, về đại thể, ít hơn $s(= \log_{10} \kappa)$ chữ số thập phân có nghĩa so với dữ liệu nhập.

1.6 Phân tích sai số thuật toán

Cho thuật toán với dữ liệu nhập $(a_1, a_2, \ldots, a_r)$ qua các tính toán giá trị trung gian bằng các phép toán số học cho lời giải $(w_1, w_2, \ldots, w_t)$. Có hai kiểu phân tích sai số làm tròn cơ bản cho một thuật toán như vậy.

Cho đến nay ta đã xét cách *phân tích sai số tiến* (forward error analysis). Phân tích này tìm các chặn sai số trong lời giải $|w_i - \tilde{w}_i|$, $i = 1, 2, \ldots, t$, bằng cách chặn tại mỗi bước tính các sai số có thể xuất hiện và ảnh hưởng của chúng.

Để cụ thể, nhắc lại biểu thức cho sai số của tổng ba số:

$$\mathrm{fl}(S_3) = S_3 + (\delta_2 + \delta_3 + \delta_2\delta_3)x_1 + (\delta_2 + \delta_3 + \delta_2\delta_3)x_2 + \delta_3 x_3.$$

Một phân tích sai số tiến có thể chặn sai số tuyệt đối bởi

$$|\mathrm{fl}(S_3) - S_3| \leq (2u + u^2)(|x_1| + |x_2|) + u|x_3|.$$

Phân tích sai số lùi[2] (backward error analysis) đi tìm một tập các dữ liệu $\tilde{a}_i$ sao cho lời giải tính toán $\tilde{w}_i$ là *lời giải chính xác* nếu các $\tilde{a}_i$ được lấy làm dữ liệu nhập; đồng thời cho các chặn của $|a_i - \tilde{a}_i|$. Có thể có vô số tập hợp dữ liệu như vậy; thỉnh thoảng có đúng một tập và có thể xảy ra, ngay cả với thuật toán rất đơn giản, không tồn tại tập nào cả.

[2]Do J.H. Wilkinson đề xuất vào thập niên 50 của thế kỷ 20.

Ta hãy diễn giải biểu thức cho $\mathrm{fl}(S_3)$ theo cách phân tích này. Ta thấy rằng

$$\mathrm{fl}(S_3) = y_1 + y_2 + y_3,$$

trong đó

$$\begin{aligned} y_1 &= x_1(1 + \delta_2 + \delta_3 + \delta_2\delta_3), \\ y_2 &= x_2(1 + \delta_2 + \delta_3 + \delta_2\delta_3), \\ y_3 &= x_3(1 + \delta_3). \end{aligned}$$

Theo phân tích sai số lùi, tổng tính toán là tổng chính xác của các số hạng y_k mà mỗi số hạng gần với giá trị cho x_k theo nghĩa tương đối.

Định nghĩa 1.4. Một thuật toán ổn định, theo nghĩa phân tích sai số lùi, nếu nó cho nghiệm chính xác của bài toán là $(w_1, w_2, \ldots, w_t)$ với dữ liệu $(\tilde{a}_1, \tilde{a}_2, \ldots, \tilde{a}_r)$ gần với dữ liệu $(a_1, a_2, \ldots, a_r)$ của bài toán gốc.

Về phần hai lời giải (chính xác và tính toán) có gần nhau hay không, đó là vấn đề về điều kiện của bài toán. Hiệu lực của cách nhìn sai số này là tách rời vai trò ổn định của thuật toán và điều kiện của bài toán. Do phép phân tích sai số lùi không tham chiếu đến lời giải chính xác cho dữ liệu gốc, nên nó đặc biệt hấp dẫn khi dữ liệu nhập có độ chính xác hữu hạn, chẳng hạn, khi dữ liệu được đo hoặc tính toán. Có thể xảy ra là một thuật toán ổn định cho nghiệm chính xác một bài toán với dữ liệu mà không thể phân biệt được với dữ liệu cho trước vì sự chính xác hữu hạn của chúng. Thật ra ta không thể yêu cầu nhiều hơn về sơ đồ số trong trường hợp như vậy, nhưng một lần nữa ta phải nhấn mạnh là *gần nghiệm như thế nào, tương ứng với dữ liệu cho trước, phụ thuộc vào điều kiện của bài toán.*

Một thí dụ số sẽ giúp nêu lên vấn đề này. Cho $x_1 = 0.12 \times 10^2$, $x_2 = 0.34 \times 10^1$, $x_3 = -0.15 \times 10^2$, giá trị thực của tổng là $S_3 = 0.40 \times 10^0$. Khi tính toán bằng số học thập phân chặt cụt hai chữ số, $\mathrm{fl}(S_3) = 0.00 \times 10^0$, một kết quả rất không chính xác. Tuy nhiên, với $y_1 = 0.116 \times 10^2$, $y_2 = x_2$ và $y_3 = x_3$, ta có $\mathrm{fl}(S_3) = y_1 + y_2 + y_3$. Kết quả tính là tổng chính xác của các số gần với dữ liệu gốc. Thật vậy, hai số trùng với dữ liệu gốc và số còn lại sai khác một lượng ít hơn đơn vị làm tròn.

1.7 Biểu diễn số dấu chấm động 64-bit IEEE

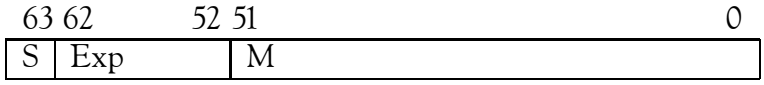
■ Câu chuyện lịch sử

Năm 1991 tên lửa Patriot đã thất bại khi được dùng để đánh chặn tên lửa Scud tấn công vào Saudi Arabi do sự sai lệch trong bài toán quyết định. Tên lửa Scud đã

bắn trúng một doanh trại giết chết 28 lính Mỹ. Nguyên nhân: máy tính dùng để điều khiển tên lửa Patriot dùng thiết kế số học 24-bit từ năm 1970. Thời gian hiệu chỉnh đường đi được thực hiện nhờ đồng hồ hệ thống với đơn vị một phần mười giây nhưng chuyển thành số dấu chấm động 24-bit. Sai số làm tròn trong phép chuyển đổi gây ra một sai số khi hiệu chỉnh. Sau 100 giờ liên tục vận hành thời gian tính toán bằng giây là 359999.6567 thay vì giá trị chính xác là 360000, một sai số 0.3433 giây dẫn đến sai lệch 687 mét! Một phần mềm sau đó đã được cài đặt để hiệu chỉnh sai lệch này.

■ **Số dấu chấm động 64-bit IEEE** (dùng trong Matlab) [14]

Số dấu chấm động 64-bit là một cấu trúc từ (word) gồm bit dấu S (sign bit), vùng mũ Exp (exponent field) và vùng định trị M (mantissa field)



Mỗi vùng này biểu diễn S, E, M của một số f theo cách sau.

• Bit dấu

$$S = b_{63} = \begin{cases} 0 & \text{với các số dương} \\ 1 & \text{với các số âm} \end{cases}$$

• Vùng mũ ($b_{62}b_{61}b_{60}\dots b_{52}$): theo mã "quá 1023"

$$\begin{aligned} E &= Exp - 1023 = \{0, 1, \dots, 2^{11} - 1 = 2047\} - 1023 \\ &= \{-1023, -1022, \dots, +1023, +1024\} \\ &= \begin{cases} -1023 + 1 & \text{khi } |f| < 2^{-1022} \text{ (Exp = (00000000000))} \\ -1022 \sim 1023 & \text{khi } 2^{-1022} \le |f| < 2^{1024} \text{ (vùng chuẩn)} \\ 1024 & \text{khi } \pm\infty \end{cases} \end{aligned}$$

• Vùng định trị ($b_{51}b_{50}\dots b_1b_0$):

Trong vùng không chuẩn ở đó các số nhỏ đến nỗi chúng có thể được biểu diễn chỉ bằng giá trị của bit ẩn (hide bit) 0, số biểu diễn bởi định trị là

$$M = 0.b_{51}b_{50}\dots b_1b_0 = [b_{51}b_{50}\dots b_1b_0] \times 2^{-52} \tag{1.21}$$

Ta có thể nghĩ rằng giá trị của bit ẩn được thêm vào số mũ thay vì vào định trị.

Trong vùng chuẩn, số biểu diễn bởi định trị cùng với giá trị của bit ẩn $b_h = 1$ là

$$\begin{aligned} M &= 1.b_{51}b_{50}\ldots b_1b_0 = 1 + [b_{51}b_{50}\ldots b_1b_0] \times 2^{-52} \\ &= 1 + b_{51} \times 2^{-1} + b_{50} \times 2^{-2} + \ldots + b_1 \times 2^{-51} + b_0 \times 2^{-52} \\ &= \{1, 1 + 2^{-52}, 1 + 2 \times 2^{-52}, \ldots, 1 + (2^{52} - 1) \times 2^{-52}\} \\ &= \{1, 1 + 2^{-52}, 1 + 2 \times 2^{-52}, \ldots, (2 - 2^{-52})\} \\ &= \{1, 1 + \Delta, 1 + 2\Delta, \ldots, 2 - \Delta\} \ (\Delta = 2^{-52}) \end{aligned} \tag{1.22}$$

Với S, E, M, số f được biểu diễn là

$$f = \pm M \cdot 2^E \tag{1.23}$$

Ta phân loại phạm vi của các số phụ thuộc vào giá trị (E) của số mũ và ký hiệu nó như là

$$[2^E, 2^{E+1}) \quad \text{với} \ -1022 \leq E \leq +1023 \tag{1.24}$$

Trong mỗi vùng, đơn vị nhỏ nhất -- nghĩa là giá trị của LSB (least significant bit, bit có nghĩa nhỏ nhất) hay hiệu giữa hai số liên tiếp biểu diễn bởi định trị 52 bit -- là

$$\Delta_E = \Delta \times 2^E = 2^{E-52} \tag{1.25}$$

Cụ thể:

0. 0 (số không)

63 62 52 51 0

S	000...0000	0000 0000 ... 0000 0000

1. Vùng không chuẩn (với giá trị của bit ẩn $b_h = 0$)

$R_{-1023} = [2^{-1074}, 2^{-1023})$ với Exp = 0, E = Exp − 1023 + 1 = −1022

S	000...0000	0000 0000 ... 0000 0001

$(0 + 2^{-52}) \times 2^E = (0 + 2^{-52}) \times 2^{-1022}$

...

S	000...0000	1111 1111 ... 1111 1111

$\{(0 + (2^{52} - 1)2^{-52}) = (1 - 2^{52})\} \times 2^{-1022}$

Giá trị của LSB: $\Delta_{-1023} = \Delta_{-1022} = 2^{-1022-52} = 2^{-1074}$

2. Vùng chuẩn nhỏ nhất (với giá trị của bit ẩn $b_h = 1$)

$R_{-1022} = [2^{-1022}, 2^{-1021})$ với Exp = 1, E = Exp − 1023 = −1022

S	000…0001	0000 0000 … 0000 0000

$(1+0)\times 2^{E} = (1+0)\times 2^{-1022}$

S	000…0001	0000 0000 … 0000 0001

$(1+2^{-52})\times 2^{-1022}$

…

S	000…0001	1111 1111 … 1111 1111

$\{(1+(2^{52}-1)2^{-52}) = (2-2^{-52})\}\times 2^{-1022}$

Giá trị của LSB: $\Delta_{-1022} = 2^{-1022-52} = 2^{-1074}$

3. Vùng chuẩn cơ sở (với giá trị của bit ẩn $b_h = 1$)

$$R_0 = [2^0, 2^1) \text{ với } \text{Exp} = 2^{10}-1,\ \text{E} = \text{Exp} - 1023 = 0$$

S	011…1111	0000 0000 … 0000 0000

$(1+0)\times 2^{E} = (1+0)\times 2^{0} = 1$

S	011…1111	0000 0000 … 0000 0001

$(1+2^{-52})\times 2^{0}$

…

S	011…1111	1111 1111 … 1111 1111

$\{(1+(2^{52}-1)2^{-52}) = (2-2^{-52})\}\times 2^{0}$

Giá trị của LSB: $\Delta_0 = 2^{-52}$

4. Vùng chuẩn lớn nhất (với giá trị của bit ẩn $b_h = 1$)

$$R_{1024} = [2^{1023}, 2^{1024}) \text{ với } \text{Exp} = 2^{11}-2,\ \text{E} = \text{Exp} - 1023 = 1023$$

S	111…1110	0000 0000 … 0000 0000

$(1+0)\times 2^{E} = (1+0)\times 2^{1023}$

S	111…1110	0000 0000 … 0000 0001

$(1+2^{-52})\times 2^{1023}$

…

S	111…1110	1111 1111 … 1111 1111

$\{(1+(2^{52}-1)2^{-52}) = (2-2^{-52})\}\times 2^{1023}$

Giá trị của LSB: $\Delta_{1023} = 2^{1023-52} = 2^{971}$

5. $\pm\infty$ (inf) $\text{Exp} = 2^{11}-1 = 2047,\ \text{E} = \text{Exp} - 1023 = 1024$ (vô nghĩa)

0	111…1111	0000 0000 … 0000 0000

$+\infty \neq (1+0)\times 2^{\mathrm{E}} = (1+0)\times 2^{1024}$

1	111…1111	0000 0000 … 0000 0000

$-\infty \neq -(1+0)\times 2^{\mathrm{E}} = -(1+0)\times 2^{1024}$

S	111…1111	0000 0000 … 0000 0001

không hiệu lực (không dùng)

…

S	111…1111	1111 1111 … 1111 1111

không hiệu lực (không dùng)

Số dương nhỏ nhất và lớn nhất có thể biểu diễn là

$$f_{\min} = (0+2^{-52})\times 2^{-1022} = 2^{-1074} = 4.9407\times 10^{-324}$$
$$f_{\max} = (2-2^{-52})\times 2^{1023} = 1.7977\times 10^{+308}$$

Cơ chế thực hiện phép tính số học trong máy tính. Thí dụ, phép cộng 3 cho 14 được thực hiện như sau.

Đổi thập phân thành nhị phân → Chuẩn hóa → Biểu diễn 64-bit

$$3_{10} = 11_2 = 1.1_2\times 2^1 = \boxed{1}.1_2\times 2^{1024-1023}$$
$$14_{10} = 1110_2 = 1.110_2\times 2^3 = \boxed{1}.110_2\times 2^{1026-1023}$$

Biểu diễn 64-bit

$3_{10} =$	0	1024_{10}	$\boxed{1}.10000\ldots$	$\ldots 0$	
$14_{10} =$	0	1026_{10}	$\boxed{1}.11000\ldots$	$\ldots 0$	
$3_{10} =$	0	1026_{10}	$\boxed{0}.01100\ldots$	$\ldots 0$	
$14_{10} =$	0	1026_{10}	$\boxed{1}.11000\ldots$	$\ldots 0$	
	0	1026_{10}	$10.00100\ldots$	$\ldots 0$	
Chuẩn hóa	0	1027_{10}	$\boxed{1}.00010\ldots$	$\ldots 0$	$= 1.0001_2\times 2^{1027-1023} = 10001_2$
Chuyển đổi					$= 1\times 2^4 + 1\times 2^0 = 17_{10}$

Trong quá trình cộng hai số, một sự "gióng cột" được thực hiện để cho hai số mũ trong biểu diễn 64-bit bằng nhau; và nó loại đi phần lớn hơn 52 bit, điều này làm xuất hiện sai số.

Nhận xét 1.2. Mỗi vùng có đơn vị tối thiểu (giá trị LSB) khác nhau. Điều này hàm ý rằng các số được phân bố đều trong mỗi vùng. Các vùng gần 0 trù mật hơn. Phép biểu diễn số như vậy làm cho lượng sai số tuyệt đối lớn/nhỏ đối với số lớn/nhỏ, giảm khả năng sai số tương đối lớn. •

Câu hỏi và bài tập

1.1. Phân tích điều kiện của thuật toán dùng công thức truy hồi (1.4). Áp dụng tính E_5 với sai số tuyệt đối không quá 10^{-6}.

1.2. Cho hệ thống số dấu chấm động với $\beta = 2$, $s = 3$, $m = -1$, $M = 2$. Hệ này có bao nhiêu số? Biểu diễn các số này trên trục số.

1.3. Viết thuật toán chuyển đổi hệ cơ số 10 sang cơ số β và ngược lại.

1.4. Cho A là ma trận vuông và n là số nguyên dương. Ta cần tính A^n. Để tính $A^{k+1} = AA^k$, $k = 1, \ldots, n-1$, đòi hỏi $n-1$ phép nhân ma trận. Chứng tỏ số các phép nhân có thể giảm bớt còn $2[\log_2 n]$ bằng cách chuyển đổi n thành dạng nhị phân và lũy thừa liên tiếp $A^{2k} = (A^k)^2$, $k = 1, \ldots, [\log_2 n]$.

1.5. Cho a và b là hai số dấu chấm động với $a \le b$. Chứng tỏ rằng bất đẳng thức

$$a \le (a \oplus b) \oslash 2 \le b$$

có thể sai trong biểu diễn theo cơ số 10.

1.6. Cho $\mathbf{u} = (u_1, u_2)$ và $\mathbf{v} = (v_1, v_2)$ là hai vectơ. Góc φ giữa hai vectơ này được cho bởi công thức

$$\cos\varphi = \frac{\mathbf{u} \cdot \mathbf{v}}{|\mathbf{u}||\mathbf{v}|}$$

a) Chứng tỏ tính φ từ các thành phần của $\mathbf{u}$ và $\mathbf{v}$ luôn là bài toán điều kiện tốt.

b) Chứng tỏ công thức trên là không ổn định khi φ nhỏ.

c) Chứng tỏ thuật toán sau là ổn định. Trước hết chuẩn hóa hai vectơ thành $\tilde{\mathbf{u}}$, $\tilde{\mathbf{v}}$, rồi tính $\alpha = \|\tilde{\mathbf{u}} - \tilde{\mathbf{v}}\|_2$ và $\beta = \|\tilde{\mathbf{u}} + \tilde{\mathbf{v}}\|_2$. Bây giờ lấy

$$\varphi = \begin{cases} 2\text{arctg}(\alpha/\beta), & \text{nếu } \alpha \le \beta; \\ \pi - 2\text{arctg}(\alpha/\beta), & \text{nếu } \alpha > \beta. \end{cases}$$

1.7. Thiết lập công thức truy hồi tiến và lùi để tính tích phân

$$I_n = \int_0^1 \frac{x^n dx}{4x + 1}.$$

Phân tích tính ổn định của từng thuật toán.

Chương 2

Hệ phương trình đại số tuyến tính

Một trong các bài toán thường gặp trong tính toán khoa học là giải hệ phương trình đại số tuyến tính

$$\left.\begin{array}{rcrcrcrcl} a_{11}x_1 & + & a_{12}x_2 & + & \ldots & + & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \ldots & + & a_{2n}x_n & = & b_2 \\ & & \vdots & & & & \vdots & & \\ a_{n1}x_1 & + & a_{n2}x_2 & + & \ldots & + & a_{nn}x_n & = & b_n, \end{array}\right\} \tag{2.1}$$

trong đó vế phải b_i, $i = 1, \ldots, n$, và các hệ số a_{ij}, $i, j = 1, \ldots, n$ là các dữ liệu cho trước; $x_1, \ldots, x_n$ là ẩn.

Hệ phương trình (2.1) có thể viết dưới dạng ma trận,

$$\mathbf{Ax} = \mathbf{b}, \tag{2.2}$$

trong đó

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{bmatrix}, \ \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \ \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}. \tag{2.3}$$

Xét trường hợp $n = 1$ trong (2.1),

$$a_{11}x_1 = b_1.$$

Nếu $a_{11} \neq 0$, phương trình có nghiệm duy nhất $x_1 = b_1/a_{11}$. Nếu $a_{11} = 0$ thì có lúc phương trình vô nghiệm ($b_1 \neq 0$), có lúc phương trình có vô số nghiệm ($b_1 = 0$). Điều này cũng đúng với n tổng quát. Có hai loại ma trận, không suy biến (nonsingular) và suy biến (singular). Nếu ma trận $\mathbf{A}$ không suy biến, thì hệ tồn tại duy nhất vectơ nghiệm $\mathbf{x}$ với vế phải cho trước bất kỳ $\mathbf{b}$. Nếu $\mathbf{A}$ suy biến, thì hệ vô nghiệm với một vài $\mathbf{b}$ nhưng vô số nghiệm với các $\mathbf{b}$ khác. Trong chương này ta xét hệ phương trình đại số tuyến tính với các ma trận hệ số không suy biến.

2.1 Phương pháp khử Gauss

Ý tưởng đằng sau phương pháp khử Gauss là dùng các phép biến đổi sơ cấp để khử các ẩn của hệ (2.1). Hệ phương trình tương đương, sau khi biến đổi, có dạng tam giác trên (upper trianglular system), được giải bằng phép thế ngược (back-substitution).

Nếu $a_{11} \neq 0$, thì ở bước đầu tiên ta khử x_1 khỏi $(n-1)$ phương trình cuối bằng cách trừ phương trình thứ i với nhân tử (multiplier)

$$m_{i1} = a_{i1}/a_{11}, \quad i = 2, \ldots, n$$

lần phương trình đầu. Điều này sinh ra một hệ rút gọn gồm $(n-1)$ phương trình với các ẩn $x_2, \ldots, x_n$, trong đó các hệ số mới được cho bởi

$$a_{ij}^{(2)} = a_{ij} - m_{i1}a_{1j}, \quad b_i^{(2)} = b_i - m_{i1}b_1, \quad j = 1, 2, \ldots, n.$$

Nếu $a_{22}^{(2)} \neq 0$, tiếp theo bằng cách tương tự ta khử x_2 từ $(n-2)$ phương trình cuối của hệ phương trình này. Sau $k-1$ bước, $k \leq n$, của phép khử Gauss ma trận A trở thành ma trận có dạng

$$\mathbf{A}^{(k)} = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \ldots & a_{1k}^{(1)} & \ldots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \ldots & a_{2k}^{(2)} & \ldots & a_{2n}^{(2)} \\ \vdots & \vdots & & \vdots & & \vdots \\ 0 & 0 & \ldots & a_{kk}^{(k)} & \ldots & a_{kn}^{(k)} \\ \vdots & \vdots & & \vdots & & \vdots \\ 0 & 0 & \ldots & a_{nk}^{(k)} & \ldots & a_{nn}^{(k)} \end{bmatrix}, \ \mathbf{b}^{(k)} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_k^{(k)} \\ \vdots \\ b_n^{(k)} \end{bmatrix}, \tag{2.4}$$

trong đó ta đã đặt $\mathbf{A}^{(1)} = \mathbf{A}$, $\mathbf{b}^{(1)} = \mathbf{b}$. Các phần tử chéo (diagonal elements) $a_{11}^{(1)}, a_{22}^{(2)}, \ldots$, xuất hiện trong quá trình khử được gọi là các phần tử trụ (pivotal elements).

Ký hiệu $\mathbf{A}_k$ là ma trận con chính của $\mathbf{A}$,

$$\mathbf{A}_k = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1k} \\ a_{21} & a_{22} & \ldots & a_{2k} \\ \vdots & \vdots & & \vdots \\ a_{k1} & a_{k2} & \ldots & a_{kk} \end{bmatrix}.$$

Vì định thức của ma trận không thay đổi dưới phép biến đổi sơ cấp thứ ba nên

$$\det(\mathbf{A}_k) = a_{11}^{(1)} a_{22}^{(2)} \cdots a_{kk}^{(k)}, \quad k = 1, \ldots, n.$$

Các phần tử trụ $a_{ii}^{(i)}$, $i = 1, \ldots, n$, trong phép khử Gauss là khác không nếu và chỉ nếu $\det(\mathbf{A}_k) \neq 0$, $k = 1, \ldots, n$. Trong trường hợp này ta có thể khử cho đến sau bước thứ $(n-1)$, còn lại một phương trình duy nhất

$$a_{nn}^{(n)} x_n = b_n^{(n)} \quad (a_{nn}^{(n)} \neq 0).$$

Các ẩn của phương trình có thể tính bằng công thức truy hồi

$$x_n = b_n^{(n)} / a_{nn}^{(n)}, \quad x_i = \left(b_i^{(i)} - \sum_{k=i+1}^{n} a_{ik}^{(i)} x_k \right) / a_{ii}^{(i)}, \quad i = n-1, \ldots, 1. \quad (2.5)$$

Quá trình này gọi là thế ngược.

Giả sử ở bước thứ k của phép khử Gauss ta có

$$a_{kk}^{(k)} = 0.$$

Nếu $\mathbf{A}$ không suy biến, thì k cột đầu của ma trận $\mathbf{A}$ là độc lập tuyến tính. Điều này cũng đúng với ma trận đã biến đổi. Nghĩa là tồn tại $a_{pk}^{(k)} \neq 0$ $(k < p \leq n)$. Bằng cách hoán vị dòng k và dòng p thì phần tử này có thể

lấy làm phần tử trụ và phép khử được tiếp tục. Tóm lại, *ma trận không suy biến bất kỳ có thể dẫn về dạng tam giác trên bằng phép khử Gauss nếu phép hoán vị dòng được dùng nếu cần.*

Khi hoán vị dòng định thức của ma trận bị đổi dấu, do đó

$$\det(\mathbf{A}) = (-1)^s a_{11}^{(1)} a_{22}^{(2)} \cdots a_{nn}^{(n)}, \tag{2.6}$$

trong đó s là tổng số lần thực hiện phép hoán vị. Ở đây, ta đã thay đổi ký hiệu khi thực hiện phép hoán vị dòng cho phù hợp.

Nếu $\text{rank}(\mathbf{A}) < n$ thì có thể xảy ra ở bước thứ $k-1$ nào đó

$$a_{ik}^{(k)} = 0, \quad i = k, \ldots, n.$$

Nếu toàn bộ các phần tử $a_{ij}^{(k)} = 0$, $i, j = k, \ldots, n$ thì $\text{rank}(\mathbf{A}) = k-1$ và ta dừng lại. Ngược lại, nếu có phần tử khác không, chẳng hạn

$$a_{pq}^{(k)},$$

ta có thể mang nó đến vị trí trụ bằng cách hoán vị dòng k với p, cột k với q (khi cột của ma trận $\mathbf{A}$ bị hoán vị thì ta cũng phải hoán vị các phần tử tương ứng trong vectơ $\mathbf{x}$. Tiến hành theo cách này mọi ma trận $\mathbf{A}$ đều có thể đưa về dạng hình thang (trapezoidal form) trên

$$\mathbf{A}^{(r)} = \left[\begin{array}{ccc|ccc} a_{11}^{(1)} & \ldots & a_{1r}^{(1)} & a_{1,r+1}^{(1)} & \ldots & a_{1n}^{(1)} \\ 0 & \ddots & \vdots & \vdots & & \vdots \\ \vdots & & a_{rr}^{(r)} & a_{r,r+1}^{(r)} & \ldots & a_{rn}^{(r)} \\ \hline 0 & \ldots & 0 & 0 & \ldots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \ldots & 0 & 0 & \ldots & 0 \end{array}\right], \quad \mathbf{b}^{(r)} = \left[\begin{array}{c} b_1^{(1)} \\ \vdots \\ b_r^{(r)} \\ \hline b_{r+1}^{(r)} \\ \vdots \\ b_n^{(r)} \end{array}\right], \tag{2.7}$$

ở bước $r = \text{rank}(\mathbf{A})$.

Từ (2.7) ta đọc được hạng của ma trận $\mathbf{A}$.

Bằng cách dùng phép biến đổi sơ cấp thứ nhất (hoán vị), quá trình khử Gauss có phần tử trụ bằng không chỉ nếu bài toán gốc là suy biến. Phát biểu

này là đúng về phương diện lý thuyết, nhưng sự phân biệt giữa suy biến và không suy biến của các bài toán là "mơ hồ" trong thực hành do ảnh hưởng của việc làm tròn. Trừ phi phần tử trụ chính xác bằng không, còn thì sự hoán vị các phương trình là không cần thiết về phương diện lý thuyết. Tuy nhiên, biến đổi với một phần tử trụ hầu như bằng không sẽ dẫn đến vấn đề về sự chính xác trong số học dấu chấm động. Xem thí dụ dưới đây do Forsythe và Moler đưa ra.

Thí dụ 2.1. Cho hệ phương trình

$$\begin{aligned} 0.000100x_1 + 1.00x_2 &= 1.00, \\ 1.00x_1 + 1.00x_2 &= 2.00. \end{aligned}$$

Bằng cách dùng số học dấu chấm động thập phân làm tròn ba-chữ số, một bước trong quá trình khử x_1 trong phương trình thứ hai không dùng hoán vị

$$\begin{aligned} [1.00 - 10000 \times 1.00]x_2 &= [2.00 - 10000 \times 1.00] \\ -10,000x_2 &= -10,000. \end{aligned}$$

Rõ ràng, $x_2 = 1.00$ và bằng phép thế ngược, $x_1 = 0.00$. Chú ý thông tin chứa trong phương trình thứ hai bị mất ở bước này. Điều này xảy ra vì phần tử trụ nhỏ gây ra một nhân tử lớn và sau đó phép trừ các số có độ lớn rất khác nhau. Nếu dùng hoán vị ta có

$$\begin{aligned} 1.00x_1 + 1.00x_2 &= 2.00, \\ 1.00x_2 &= 1.00 \end{aligned}$$

và $x_1 = 1.00$, $x_2 = 1.00$. Nghiệm chính xác là xấp xỉ $x_1 = 1.00010$, $x_2 = 0.99990$ ∘

Các phần tử trụ nhỏ có thể dẫn đến kết quả không chính xác. Như đã thấy trong thí dụ trên, khi khử biến x_k trong dòng i, một phần tử trụ nhỏ dẫn tới nhân tử $m_{ik} = a_{ik}^{(k)}/a_{kk}^{(k)}$ lớn. Khi tính

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik}a_{kj}^{(k)},$$

có sự mất mát thông tin bất cứ khi nào $m_{ik}a_{kj}^{(k)}$ lớn hơn rất nhiều $a_{ij}^{(k)}$, thông tin mà có thể rất cần đến sau đó. Một nhân tử lớn gây ra hậu quả cũng giống

như phần tử trong ma trận tam giác trên lớn do phép khử. Trong phép giải hệ phương trình tuyến tính bằng phép thế ngược ta tính

$$x_k = \frac{b_k^{(k)} - \sum_{j=k+1}^{n} a_{kj}^{(k)} x_j}{a_{kk}^{(k)}}.$$

Nếu phần tử trụ (mẫu số) là nhỏ và giá trị đúng x_k có độ lớn vừa phải, thì tử số cũng phải nhỏ. Nhưng nếu có các thành phần $a_{ij}^{(k)}$ của ma trận tam giác trên mà lớn, điều này chỉ có thể nếu sự khử xảy ra ở tử số. Các thành phần lớn có thể đã được tính với sai số tương đối vừa phải, nhưng vì các thành phần là lớn dẫn đến sai số tuyệt đối lớn trong tử số sau khi có sự khử. Mẫu số nhỏ khuếch đại sai số này dẫn đến sai số tương đối đáng kể trong x_k.

Có một cách để tránh phần tử trụ nhỏ được gọi là *phép xoay cục bộ* (tạm dịch chữ partial pivoting). Theo cách này, khi khử x_k, ta chọn hệ số lớn nhất (về giá trị tuyệt đối) của x_k trong $n-k+1$ phương trình cuối như là phần tử trụ. Nghĩa là, nếu $|a_{lk}^{(k)}|$ là lớn nhất của các $|a_{jk}^{(k)}|$ với $j = k, k+1, \ldots, n$ ta hoán vị dòng k và l. Bằng cách đánh số lại ta có thể giả sử rằng phần tử trụ $a_{kk}^{(k)}$ có độ lớn lớn nhất. Phép xoay cục bộ cho ta các nhân tử có độ lớn

$$|m_{ik}| = \left| \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} \right| \leq 1.$$

Sự điều khiển độ lớn của các nhân tử làm bớt đi sự tăng lên của các thành phần trong ma trận tam giác trên gây ra do phép khử (Gauss). Cho $a = \max_{i,j} |a_{jk}^{(1)}|$. Bấy giờ

$$a_{ij}^{(2)} = |a_{ij}^{(1)} - m_{i1} a_{ij}^{(1)}| \leq 2a$$

và cứ thế ta có

$$|a_{ij}^{(k)}| \leq 2^{k-1} a.$$

Điều này hàm ý

$$\max_{i,j,k} |a_{ij}^{(k)}| \leq 2^{n-1} \max_{i,j} |a_{ij}| \tag{2.8}$$

khi phép xoay cục bộ được thực hiện. Wilkinson đã chỉ ra rằng dấu bằng trong bất đẳng thức trên có thể xảy ra với các ma trận có dạng

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ -1 & 1 & 0 & 0 & 1 \\ -1 & -1 & 1 & 0 & 1 \\ -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 \end{bmatrix}.$$

Tuy nhiên, thường thì sự gia tăng là vừa phải.

Con số

$$g_n = \max_{i,j,k} |a_{ij}^{(k)}| / \max_{i,j} |a_{ij}| \tag{2.9}$$

thường được dùng làm số đo cho sự gia tăng của các phần tử trong ma trận rút gọn, và được gọi là *tỉ số gia tăng* (growth ratio).

2.2 Thuật toán khử Gauss

Trường hợp "vuông". Cho ma trận $\mathbf{A} = \mathbf{A}^{(1)} \in \mathbb{R}^{n\times n}$ và vectơ $\mathbf{b} = \mathbf{b}^{(1)} \in \mathbb{R}^n$,

```
% Khử Gauss
for k=1:n-1
    hoán vị các dòng sao cho abs(a(k,k))=max(abs(a(k:n,k))
    if abs(a(k,k) == 0, ' suy biến', return
    for i=k+1:n
        t=a(i,k)/a(k,k)
        for j=k+1:n
            a(i,j)=a(i,j)-t*a(k,j)
        end
        b(i)=b(i)-t*b(k)
```

```
    end
end
if abs(a(n,n)) ==0, ' suy biến', return
% Thế ngược
for i=n:-1:1
    x(i)=b(i)
    for j=i+1:n
        x(i)=x(i)-a(i,j)*x(j)
    end
    x(i)=x(i)/a(i,i)
end
```

Nhận xét 2.1. Để đo khối lượng công việc ta đếm số phép toán số học được thực hiện. Giai đoạn khử:

+ Mỗi bước của vòng lặp j gồm 1 phép nhân và 1 phép trừ. Như vậy có (n-k) phép nhân và (n-k) phép trừ.

+ Mỗi bước của vòng lặp i gồm 1 phép chia (không kể phép tính hiệu chỉnh **b**) và các phép tính trong vòng lặp j. Vòng lặp i gồm (n-k) bước. Như vậy có $(n-k)^2$ phép nhân, $(n-k)^2$ phép trừ và (n-k) phép chia.

+ Mỗi bước của vòng lặp k chứa các phép toán của vòng lặp i tương ứng. Vòng lặp k có (n-1) bước. Như vậy số phép tính nhân (=số phép tính trừ) và số phép tính chia lần lượt là

$$(n-1)^2 + (n-2)^2 + \ldots + 1^2 = \frac{n(n-1)(2n-1)}{6},$$

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

Phần hiệu chỉnh **b** đòi hỏi số phép tính nhân (=số phép tính trừ) bằng $n(n-1)/2$.

Giai đoạn thế ngược, số phép tính nhân (=số phép tính trừ) và phép tính chia lần lượt là $n(n-1)/2$ và n. Tóm lại, khối lượng tính toán bằng thuật toán khử Gauss: $n(4n^2+9n-7)/6$. •

Nhận xét 2.2. Khi phải giải nhiều hệ phương trình có cùng ma trận **A** với các vectơ **b** khác nhau, các nhân tử m_{ik} cần được lưu trữ để hiệu chỉnh vectơ **b**. Để ý rằng, khi m_{ik} được tính thì phần tử $a_{ik}^{(k)}$ được đặt bằng không, vì vậy nhân tử m_{ik} có thể được đặt vào vị trí này (không cần thêm vùng nhớ để lưu trữ nó). •

2.3 Phép nhân tử hóa ma trận (matrix factorization)

Trong mục này ta sẽ thấy nếu không dùng phép xoay cục bộ, thuật toán khử chính là *phép nhân tử hóa*, phân tích ma trận $\mathbf{A}$ thành tích $\mathbf{LU}$ của một ma trận tam giác dưới $\mathbf{L} = [l_{ij}]$, trong đó

$$l_{ij} = 0 \quad \text{nếu } i < j,$$

và một ma trận tam giác trên $\mathbf{U} = [u_{ij}]$, trong đó

$$u_{ij} = 0 \quad \text{nếu } i > j.$$

Nhìn lại phép khử được mô tả trong mục trước, ta thấy nếu $a_{11}^{(1)} \neq 0$, ta có thể thay dòng i bằng dòng i trừ với $m_{i1} = a_{i1}^{(1)}/a_{11}^{(1)}$ lần dòng 1. Điều này được thực hiện cho các dòng $2, 3, \ldots, n$. Dễ dàng kiểm chứng, nếu nhân bên trái ma trận $\mathbf{A}$ với ma trận

$$\mathbf{M}_1 = \begin{bmatrix} 1 & & & & \\ -m_{21} & 1 & & & \\ -m_{31} & 0 & 1 & & \\ \vdots & \vdots & \vdots & \ddots & \\ -m_{n1} & 0 & 0 & & 1 \end{bmatrix}$$

ta nhận được cùng một kết quả như khi thực hiện phép khử; nghĩa là

$$\mathbf{M}_1\mathbf{A} = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \ldots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \ldots & a_{2n}^{(2)} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & a_{n2}^{(2)} & a_{n3}^{(2)} & \ldots & a_{nn}^{(2)} \end{bmatrix}.$$

Hơn nữa, nghịch đảo của ma trận $\mathbf{M}_1$ là

$$\mathbf{M}_1^{-1} = \begin{bmatrix} 1 & & & & \\ m_{21} & 1 & & & \\ m_{31} & 0 & 1 & & \\ \vdots & \vdots & \vdots & \ddots & \\ m_{n1} & 0 & 0 & & 1 \end{bmatrix}.$$

Tiếp tục khử các ẩn, sau $n-1$ bước ta được

$$\mathbf{M}_{n-1}\mathbf{M}_{n-2}\cdots\mathbf{M}_1\mathbf{A} == \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \dots & a_{1n}^{(1)} \\ & a_{22}^{(2)} & \dots & a_{21}^{(2)} \\ & & \ddots & \\ & & & a_{nn}^{(n)} \end{bmatrix} = \mathbf{U}.$$

Suy ra,

$$\mathbf{A} = \mathbf{M}_1^{-1}\mathbf{M}_2^{-1}\cdots\mathbf{M}_{n-1}^{-1}\mathbf{U} = \mathbf{L}\mathbf{U},$$

trong đó $\mathbf{L} = \mathbf{M}_1^{-1}\mathbf{M}_2^{-1}\cdots\mathbf{M}_{n-1}^{-1}$. Kiểm trực tiếp ta thấy $\mathbf{L}$ là ma trận tam giác dưới,

$$\mathbf{L} = \begin{bmatrix} 1 & & & & \\ m_{21} & 1 & & & \\ m_{31} & m_{32} & 1 & & \\ \vdots & \vdots & \vdots & \ddots & \\ m_{n1} & m_{n2} & m_{n3} & \dots & 1 \end{bmatrix}.$$

Vì các phần tử trên đường chéo chính của ma trận $\mathbf{L}$ đều bằng 1 nên ta không cần lưu trữ chúng. Các cột của ma trận $\mathbf{L}$ xuất liện lần lượt theo thứ tự của phép khử nên có thể viết chồng lên vị trí tương ứng của ma trận $\mathbf{A}$ mà trong phép khử được đặt bằng không. Ma trận tam giác trên $\mathbf{U}$ chính là kết quả của phép khử Gauss.

Với phép nhân tử hóa ma trận $\mathbf{A}$, hệ phương trình $\mathbf{A}\mathbf{x} = \mathbf{b}$ dẫn về việc giải lần lượt hai hệ phương trình với ma trận hệ số có dạng tam giác,

$$\mathbf{L}\mathbf{y} = \mathbf{b}, \tag{2.10}$$

và

$$\mathbf{U}\mathbf{x} = \mathbf{y}. \tag{2.11}$$

Thật vậy, $\mathbf{b} = \mathbf{L}\mathbf{y} = \mathbf{L}(\mathbf{U}\mathbf{x}) = (\mathbf{L}\mathbf{U})\mathbf{x} = \mathbf{A}\mathbf{x}$.

Hệ tam giác dưới (2.10) được giải bằng thế tiến (forward substitution),

$$\begin{aligned} y_1 &= b_1 \\ y_2 &= b_2 - m_{21} y_1 \\ &\vdots \\ y_n &= b_n - \sum_{j=1}^{n-1} m_{nj} y_j . \end{aligned}$$

Hệ tam giác trên (2.11) được giải bằng phép thế lùi (backward substitution)

$$\begin{aligned} x_n &= y_n / u_{nn} \\ x_{n-1} &= (y_{n-1} - u_{n-1,n} x_n)/u_{n-1,n-1} \\ &\vdots \\ x_1 &= \left(y_1 - \sum_{j=2}^{n} u_{1j} x_j \right) / u_{11} . \end{aligned}$$

Nhận xét 2.3. Khi có sử dụng phép xoay cục bộ, áp dụng phép nhân tử hóa cho ma trận $\mathbf{PA}$ với $\mathbf{P}$ là ma trận được xây dựng từ ma trận đơn vị bằng cách hoán vị các dòng tương ứng. Ta có $\mathbf{PA} = \mathbf{LU}$. •

2.4 Sự chính xác

Có hai nguồn sai số trong nghiệm tính toán $\mathbf{z}$ của hệ phương trình $\mathbf{Ax} = \mathbf{b}$. Thứ nhất, dữ liệu $\mathbf{A}$ và $\mathbf{b}$ có thể không được đo chính xác, và ngay cả nếu có chính xác, thì vẫn có các sai số được phát sinh khi biểu diễn chúng bằng số dấu chấm động. Thứ hai, sai số làm tròn xuất hiện trong quá trình khử và thuật toán thay thế tiến/lùi. Một cách tự nhiên ta cần nghiên cứu sai số

$$\mathbf{e} = \mathbf{x} - \mathbf{z}.$$

Nhưng có một cách tiếp cận khác về vấn đề chính xác. Phân tích sai số lùi xem $\mathbf{z}$ như là nghiệm chính xác của bài toán nhiễu

$$(\mathbf{A} + \Delta\mathbf{A})\mathbf{z} = \mathbf{b} + \Delta\mathbf{b}.$$

Nếu các nhiễu, $\Delta\mathbf{A}$ và $\Delta\mathbf{b}$ là so sánh được với các sai số đo đạc hay làm tròn trong các phần tử của $\mathbf{A}$ và $\mathbf{b}$, thì *có lý* để nói rằng $\mathbf{z}$ gần như là nghiệm tốt như ta có thể hy vọng.

2.4.1 Phân tích sai số lùi

Xét hệ phương trình

$$\begin{aligned} u_{11}x_1 + u_{12}x_2 &= b_1, \\ u_{22}x_2 &= b_2. \end{aligned}$$

Ta bàn về cách một phần tử bé, ở đây là u_{11} và u_{22}, có thể gây nguy hiểm vì các ảnh hưởng trực tiếp của nó lẫn bản thân nó có thể dẫn đến các phần tử lớn trong ma trận tam giác trên, ở đây là u_{12}. Phép thế lùi bằng phép tính số học chính xác cho nghiệm đúng là

$$\begin{aligned} x_2 &= \frac{b_2}{u_{22}}, \\ x_1 &= \frac{b_1 - x_2 u_{12}}{u_{11}}. \end{aligned}$$

Trong số học dấu chấm động,

$$x_2^* = b_2 \oslash u_{22} = \frac{b_2}{u_{22}}(1+\delta_1) = x_2(1+\delta_1).$$

Tính toán thành phần còn lại bao gồm nhiều bước. Trước hết ta tính

$$x_2^* \otimes u_{12} = x_2^* u_{12}(1+\delta_2) = x_2 u_{12}(1+\delta_1)(1+\delta_2),$$

thì

$$b_1 \ominus (x_2^* \otimes u_{12}) = (b_1 - (x_2^* \otimes u_{12}))(1+\delta_3),$$

và cuối cùng

$$\begin{aligned} x_1^* &= (b_1 \ominus (x_2^* \otimes u_{12})) \oslash u_{11} \\ &= \frac{(b_1 \ominus (x_2^* \otimes u_{12}))}{u_{11}}(1+\delta_4) \\ &= \frac{(b_1 - x_2^* u_{12}(1+\delta_2))}{u_{11}}(1+\delta_3)(1+\delta_4). \end{aligned}$$

Trong phân tích sai số lùi, ta biểu diễn nghiệm x_1^*, x_2^* (kết quả tính bằng số học dấu chấm động) như là nghiệm (tính bằng số học chính xác) của một bài toán nhiễu (bài toán gần):

$$\begin{aligned} u_{11}^* x_1^* + u_{12}^* x_2^* &= b_1, \\ u_{22}^* x_2^* &= b_2. \end{aligned}$$

Ở đây không có nhiễu ở vế phải. Phương trình

$$\begin{aligned} x_2^* &= \frac{b_2}{u_{22}^*} \\ &= \frac{b_2}{u_{22}}(1+\delta_1) \end{aligned}$$

sẽ đúng nếu ta định nghĩa

$$u_{22}^* = \frac{u_{22}}{1+\delta_1} \approx u_{22}(1-\delta_1).$$

Tương tự, phương trình

$$\begin{aligned} x_1^* &= \frac{b_1 - x_2^* u_{12}^*}{u_{11}^*} \\ &= \frac{b_1 - x_2^* u_{12}(1+\delta_2)}{u_{11}}(1+\delta_3)(1+\delta_4) \end{aligned}$$

sẽ đúng nếu ta định nghĩa

$$\begin{aligned} u_{12}^* &= u_{12}(1+\delta_2), \\ u_{11}^* &= \frac{u_{11}}{(1+\delta_3)(1+\delta_4)} \approx u_{11}(1-\delta_3-\delta_4). \end{aligned}$$

Với các định nghĩa này ta đã biểu diễn nghiệm tính toán của bài toán cho trước như là nghiệm chính xác của bài toán với ma trận các hệ số bị nhiễu. Nó cho thấy không có hệ số nào của ma trận bị nhiễu nhiều hơn hai đơn vị làm tròn.

Phân tích này nói cho chúng ta rằng thuật toán thế ngược bảo đảm sinh ra một kết quả tốt theo nghĩa nghiệm tính toán là nghiệm chính xác của bài toán gần với bài toán cho. Tuy nhiên, điều đó không đồng nghĩa với phát biểu: *nghiệm tính toán gần với nghiệm thực.*

Phân tích sai số tiến cho phép ước lượng sự khác nhau giữa nghiệm tính toán và nghiệm thực. Giả thiết cơ bản của chúng ta về số học dấu chấm động là một phép toán được thực hiện với một sai số tương đối bị chặn bởi đơn vị làm tròn u, vì vậy ta có

$$\frac{x_2^* - x_2}{x_2} = |\delta_1| \le u.$$

Thay thế các biểu thức đã phát triển trước đây và một tính toán nhỏ chứng tỏ rằng

$$\frac{x_1^* - x_1}{x_1} = \sigma_2 - \frac{x_2 u_{12}}{x_1 u_{11}}\sigma_1(1+\sigma_2),$$

trong đó

$$\begin{aligned} \sigma_1 &= \delta_1 + \delta_2 + \delta_1\delta_2, \\ \sigma_2 &= \delta_3 + \delta_4 + \delta_3\delta_4. \end{aligned}$$

Suy ra

$$\left|\frac{x_1^* - x_1}{x_1}\right| \le (2u+u^2)\left[1 + \left|\frac{x_2 u_{12}}{x_1 u_{11}}\right|(1+2u+u^2)\right].$$

Theo ước lượng này, sai số tương đối nói chung là nhỏ. Một sai số tương đối lớn chỉ có thể xảy ra khi $|x_2 u_{12}| \gg |x_1 u_{11}|$. Nếu nghiệm là sao cho cả

hai thành phần có độ lớn so sánh được, một sai số tương đối lớn chỉ xảy ra khi phần tử trụ u_{11} là nhỏ và/hay phần tử u_{12} trong ma trận tam giác trên là lớn. Các sai số tương đối lớn có khả năng xảy ra nhiều hơn khi $|x_2| \gg |x_1|$. Mẫu số có thể viết lại dưới dạng

$$x_1 u_{11} = b_1 - x_2 u_{12},$$

chứng tỏ rằng sai số tương đối có thể là lớn khi tử số là lớn và mẫu số là nhỏ bởi sự khử (xem thí dụ 1.15, Ch. 1).

2.4.2 Phân tích sự làm tròn

Một cách tự nhiên để đo chất lượng của một nghiệm xấp xỉ $\mathbf{z}$ là thay nó vào phương trình gốc rồi xem nó thỏa phương trình "tốt" như thế nào. Với cách làm này giá trị thặng dư (residual),

$$\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{z},$$

cho biết mức độ chính xác của lời giải. Một nghiệm tốt $\mathbf{z}$ thì có giá trị thặng dư nhỏ. Vì sự khử, nếu ta cần giá trị thặng dư chính xác cho một nghiệm tốt thì phải tính nó bằng số học có độ chính xác cao hơn, mà điều này thì không thể. Giá trị thặng dư cung cấp một nhiễu $\Delta\mathbf{b}$ cho phân tích sai số lùi, cụ thể,

$$\Delta\mathbf{b} := -\mathbf{r}.$$

Giá trị thặng dư $\mathbf{r}$ liên hệ với sai số $\mathbf{e}$ bởi

$$\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{z} = \mathbf{A}\mathbf{x} - \mathbf{A}\mathbf{z} = \mathbf{A}(\mathbf{x} - \mathbf{z}) = \mathbf{A}\mathbf{e}$$

hay $\mathbf{e} = \mathbf{A}^{-1}\mathbf{r}$.

Một giá trị thặng dư nhỏ $\mathbf{r}$, thì $\Delta\mathbf{b}$ nhỏ và theo quan điểm phân tích sai số lùi thì $\mathbf{z}$ là nghiệm tốt ngay cả khi sai số tương ứng $\mathbf{e}$ không nhỏ.

Thí dụ 2.2. Để minh họa sự khác biệt giữa hai quan điểm, xét hệ phương trình

$$\begin{bmatrix} 0.747 & 0.547 \\ 0.623 & 0.457 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0.200 \\ 0.166 \end{bmatrix}. \tag{2.12}$$

Thực hiện phép khử dùng số học thập phân chặt cụt ba-chữ số. Sau bước đầu ta có

$$\begin{bmatrix} 0.747 & 0.547 \\ 0 & 0.001 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0.200 \\ 0.000 \end{bmatrix}.$$

Suy ra

$$\begin{aligned} z_2 &= 0.000, \\ z_1 &= (0.200 - 0.547 z_2)/0.747 = 0.267. \end{aligned}$$

Như vậy nghiệm tính toán là

$$\mathbf{z} = \begin{bmatrix} 0.267 \\ 0.000 \end{bmatrix}$$

Dễ thấy nghiệm chính xác là $x_1 = 1$ và $x_2 = -1$. Vì vậy

$$\mathbf{e} = \mathbf{x} - \mathbf{z} = \begin{bmatrix} 0.733 \\ -1 \end{bmatrix}.$$

Trái lại, giá trị thặng dư (trong số học chính xác) là

$$\begin{aligned} \mathbf{r} &= \mathbf{b} - \mathbf{A}\mathbf{z} \\ &= \begin{bmatrix} 0.200 - [(0.747 \times 0.267) + (0.547 \times 0.000)] \\ 0.166 - [(0.623 \times 0.267) + (0.457 \times 0.000)] \end{bmatrix} \\ &= \begin{bmatrix} 0.000551 \\ -0.000341 \end{bmatrix}. \end{aligned}$$

Điều này chứng tỏ $\mathbf{z}$ là nghiệm chính xác của $\mathbf{A}\mathbf{z} = \mathbf{b} + \Delta\mathbf{b}$, trong đó $b_1 = 0.200$ bị nhiễu thành 0.199449 và $b_2 = 0.166$ bị nhiễu thành 0.166341. Như vậy, $\mathbf{z}$ là nghiệm của bài toán rất gần với bài toán cho, mặc dù nó sai so với nghiệm $\mathbf{x}$ rất đáng kể ∘

Khó khăn cơ bản trong thí dụ 2.2 là ma trận của hệ (2.12) gần suy biến. Thật vậy phương trình đầu, trong phạm vi sai số làm tròn, bằng 1.2 lần phương trình thứ hai. Trong quá trình giải ta thấy z_2 được tính từ hai đại lượng mà bản thân chúng có độ lớn cùng cấp với sai số làm tròn. Thực hiện tính toán với nhiều chữ số hơn ta sẽ thấy z_2 có giá trị hoàn toàn khác. Sai số trong z_2 được truyền đến sai số trong z_1 và nghiệm tính toán là không tốt. Nhưng tại sao giá trị thặng dư lại nhỏ? Bất kể z_2, số z_1 được tính để làm cho giá trị thặng dư của phương trình đầu gần bằng không. Thặng dư của phương trình

thứ hai cũng nhỏ vì hệ thống gần như kỳ dị: phương trình đầu xấp xỉ bằng một bội của phương trình thứ hai.

Để phân tích sai số làm tròn trong quá trình khử Gauss ta dùng cách diễn giải nhân tử hóa. Đơn giản ta xét trường hợp không dùng phép xoay cục bộ. Gọi $\Delta\mathbf{L}$ và $\Delta\mathbf{U}$ là sai số khi tính toán $\mathbf{L}$ và $\mathbf{U}$. Như vậy ma trận $\mathbf{A}$ không bằng $(\mathbf{L}+\Delta\mathbf{L})(\mathbf{U}+\Delta\mathbf{U})$. Đặt $\Delta\mathbf{A}$ là sai biệt, ta có

$$\Delta\mathbf{A} = (\Delta\mathbf{L})\mathbf{U} + \mathbf{L}(\Delta\mathbf{U}) + (\Delta\mathbf{L})(\Delta\mathbf{U}).$$

Ta có thể hy vọng là quá trình tính $\mathbf{L}$ cũng như $\mathbf{U}$ có sai số tương đối nhỏ. Tuy nhiên, biểu thức của $\Delta\mathbf{A}$ chứng tỏ rằng độ lớn của $\mathbf{L}$ và $\mathbf{U}$ đóng vai trò quan trọng trong kết quả nhân tử hóa ma trận $\mathbf{A}$. Phép xoay cục bộ giữ cho các phần tử của $\mathbf{L}$ nhỏ hơn hay bằng 1 về mặt độ lớn. Ta cũng thấy trong (2.8) rằng độ lớn của các phần tử của $\mathbf{U}$, $a_{ij}^{(k)}$, được làm "dịu" đi bằng phép xoay cục bộ. Đặc biệt, chúng không thể vượt quá $2^{n-1}\times\max_{ij}|a_{ij}|$ với $n\times n$-ma trận. Có thể chỉ ra, bằng cách tính sai số của phép nhân tử hóa và phép thế tiến/lùi, nghiệm tính toán $\mathbf{z}$ của phương trình $\mathbf{Ax}=\mathbf{b}$ thỏa

$$(\mathbf{A}+\Delta\mathbf{A})\mathbf{z} = \mathbf{b}, \tag{2.13}$$

trong đó các phần tử của $\Delta\mathbf{A}$ thường là nhỏ. Để chính xác ta cần đưa vào cách đo độ lớn của vectơ và ma trận. Một cách đo độ dài quen thuộc của vectơ là chuẩn Euclide ($p=2$), $(\sum_{i=1}^{n}x_i^2)^{1/2}$. Tuy nhiên trong tài liệu này ta dùng chuẩn maximum ($p=\infty$)

$$\|\mathbf{x}\| = \max_{1\le i\le n}|x_i|. \tag{2.14}$$

Nếu $\mathbf{A}$ là ma trận vuông cấp n và $\mathbf{x}$ là một n-vectơ. Chuẩn của ma trận $\mathbf{A}$ được định bởi

$$\|\mathbf{A}\| = \max_{\mathbf{x}\neq 0}\frac{\|\mathbf{Ax}\|}{\|\mathbf{x}\|}. \tag{2.15}$$

Một cách hình học, điều này nói rằng $\|\mathbf{A}\|$ là sự lệch tương đối cực đại (maximum relative distorsion) mà ma trận $\mathbf{A}$ tạo ra khi nó nhân với vectơ $\mathbf{x}\neq 0$. Do chuẩn này không dễ tính toán, thường ta dùng một chuẩn tương đương với nó:

$$\|\mathbf{A}\| = \max_{1\le i\le n}\left\{\sum_{j=1}^{n}|a_{ij}|\right\}. \tag{2.16}$$

Chú ý, ta có bất đẳng thức quan trọng sau

$$\|\mathbf{Ax}\| \leq \|\mathbf{A}\|\|\mathbf{x}\|. \tag{2.17}$$

Thí dụ 2.3. Cho

$$\mathbf{x} = \begin{bmatrix} -1 \\ 2 \\ 3 \end{bmatrix}.$$

Thì

$$\|\mathbf{x}\| = \max\{|-1|, |2|, |3|\} = 3.$$

Cho

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

Thì

$$\|\mathbf{A}\| = \max\{|1| + |-1|, |2| + |-2| + |3|, |-4| + |1| + |-1|\} = 7 \circ$$

Trở lại vấn đề phân tích sai số làm tròn cho phép khử Gauss. Người ta chứng minh được rằng nghiệm tính toán $\mathbf{z}$ thỏa phương trình (2.13) trong đó

$$\|\Delta\mathbf{A}\| \leq \gamma_n u \|\mathbf{A}\|. \tag{2.18}$$

Như thường lệ, u là đơn vị làm tròn, nhân tử γ_n phụ thuộc n và có thể tăng như 2^{n-1}.

Từ đây có thể kết luận rằng phép khử Gauss luôn cho nghiệm $\mathbf{z}$ là nghiệm chính xác của bài toán gần với bài toán gốc. Vì $\mathbf{Az} - \mathbf{b} = -\Delta\mathbf{Az}$ nên thặng dư thỏa

$$\|\mathbf{r}\| = \|\mathbf{Az} - \mathbf{b}\| \leq \|\Delta\mathbf{A}\|\|\mathbf{z}\| \leq \gamma_n u \|\mathbf{A}\|\|\mathbf{z}\|.$$

Điều này nói rằng kích thước thặng dư hầu như tương đối nhỏ so với kích thước của $\mathbf{A}$ và $\mathbf{z}$. Tuy nhiên, nhắc lại rằng, điều này không ám chỉ rằng sai số thực $\mathbf{e}$ là nhỏ.

Nhận xét 2.4. Để hiểu thêm lý do tại sao phép khử Gauss dẫn đến các nghiệm tính toán với thặng dư nhỏ, xét phân tích $\mathbf{LU}$ của $\mathbf{A}$. Quá trình thế tiến dùng để giải hệ tam giác dưới $\mathbf{Ly} = \mathbf{b}$ tính liên tiếp $y_1, y_2, \ldots, y_n$ để làm cho thặng

dư bằng không. Chẳng hạn, bất chấp sai số trong y_1 và $m_{2,1}$ giá trị y_2 được tính để mà

$$m_{2,1}y_1 + y_2 = b_2,$$

nghĩa là, thặng dư của phương trình này là không (trong số học chính xác) với giá trị này của y_2. Điều giống như vậy xảy ra trong quá trình thế lùi để tính $x_n, x_{n-1}, \ldots, x_1$ thỏa $\mathbf{Ux} = \mathbf{y}$. Vậy, rất tự nhiên về quá trình phản ứng với các sai số trong dữ liệu theo cách như vậy nhận được một thặng dư bé. Điều này không đúng chút nào khi $\mathbf{x}$ được tính bằng $\mathbf{A}^{-1}\mathbf{b}$. Thêm một chút "thao tác" có thể làm phép khử Gauss ổn định theo nghĩa mạnh. •

2.4.3 Ước lượng chuẩn cho sai số

Bây giờ ta xét ảnh hưởng của sai số trong dữ liệu nhập lên sai số $\mathbf{e}$. Cho $\mathbf{x} + \Delta\mathbf{x}$ là nghiệm của

$$(\mathbf{A} + \Delta\mathbf{A})(\mathbf{x} + \Delta\mathbf{x}) = \mathbf{b} + \Delta\mathbf{b}.$$

Trừ cho $\mathbf{Ax} = \mathbf{b}$ ta được $(\mathbf{A} + \Delta\mathbf{A})\Delta\mathbf{x} = -\Delta\mathbf{Ax} + \Delta\mathbf{b}$. Giả sử ma trận $\mathbf{A} + \Delta\mathbf{A}$ không suy biến, ta có thể nhân hai vế với $\mathbf{A}^{-1}$ rồi giải ra $\Delta\mathbf{x}$

$$\Delta\mathbf{x} = (\mathbf{I} + \mathbf{A}^{-1}\Delta\mathbf{A})^{-1}\mathbf{A}^{-1}(-\Delta\mathbf{Ax} + \Delta\mathbf{b}). \tag{2.19}$$

Dùng bất đẳng thức cho chuẩn ta suy ra

$$\|\Delta\mathbf{x}\| \leq \|(\mathbf{I} + \mathbf{A}^{-1}\Delta\mathbf{A})^{-1}\|\|\mathbf{A}^{-1}\|(\|\Delta\mathbf{A}\|\|\mathbf{x}\| + \|\Delta\mathbf{b}\|). \tag{2.20}$$

Bỏ qua các số hạng cấp hai (nhiễu $\Delta\mathbf{A}$ đủ nhỏ) và dùng bất đẳng thức $\|\mathbf{b}\| = \|\mathbf{AX}\| \leq \|\mathbf{A}\|\|\mathbf{x}\|$,

$$\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|} \lessapprox \|\mathbf{A}\|\|\mathbf{A}^{-1}\|\left(\frac{\|\Delta\mathbf{b}\|}{\|\mathbf{b}\|} + \frac{\|\Delta\mathbf{A}\|}{\|\mathbf{A}\|}\right). \tag{2.21}$$

Trong trường hợp đơn giản $\Delta\mathbf{A} = 0$, ta có

$$\|\Delta\mathbf{x}\| \leq \|\mathbf{A}^{-1}\|\|\Delta\mathbf{b}\|, \tag{2.22}$$

$$\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|} \lessapprox \|\mathbf{A}\|\|\mathbf{A}^{-1}\|\frac{\|\Delta\mathbf{b}\|}{\|\mathbf{b}\|}. \tag{2.23}$$

Bất đẳng thức (2.20) tương ứng (2.22) cho sai số còn bất đẳng thức (2.21) tương ứng (2.23) cho sai số theo nghĩa tương đối. Đại lượng $\|\mathbf{A}\|\|\mathbf{A}^{-1}\|$, ký hiệu bởi $\text{cond}(\mathbf{A})$ hay $\kappa(\mathbf{A})$, được gọi là số điều kiện (condition number) của $\mathbf{A}$.

Định lý sau cho ta ý nghĩa của số điều kiện.

n	$\kappa_2(H_n)$	n	$\kappa_2(H_n)$
1	1	7	4.753×10^8
2	19.281	8	1.526×10^{10}
3	5.241×10^2	9	4.932×10^{11}
4	1.551×10^4	10	1.602×10^{13}
5	4.766×10^5	11	5.220×10^{14}
6	1.495×10^7	12	1.678×10^{16}

Bảng 2.1: Số điều kiện của ma trận Hilbert cấp ≤ 12.

Định lý 2.1. *tồn tại ma trận suy biến* $\mathbf{S}$ *sai khác* $\mathbf{A}$ *theo nghĩa tương đối bằng nghịch đảo số điều kiện của* $\mathbf{A}$

$$\min_{\det(\mathbf{S})=0} \frac{\|\mathbf{S}-\mathbf{A}\|}{\|\mathbf{A}\|} = \frac{1}{\text{cond}(\mathbf{A})}.$$

Như vậy, nếu $\mathbf{A}$ có số điều kiện "lớn" thì nó gần với ma trận suy biến.

Thí dụ 2.4. Ma trận Hilbert H_n cấp n là ma trận $n \times n$ với các phần tử

$$H_n(i,j) = h_{ij} = \frac{1}{i+j-1}.$$

Ma trận này là một thí dụ cho ma trận điều kiện xấu.

Bảng 2.1 cho kết quả tính số điều kiện của các ma trận Hilbert cấp ≤ 12 dùng số chính xác kép IEEE (chuẩn $p = 2$). Ta thấy số điều kiện tăng dạng mũ của n. Khi $n > 12$ ma trận H_n cực kỳ xấu ngay cả với số chính xác kép! Theo một kết quả của G. Szegö ta có ước lượng sau

$$\kappa(H_n) \approx \frac{(\sqrt{2}+1)^{4(n+1)}}{2^{15/4}\sqrt{\pi n}} \sim e^{3.5n} \circ$$

Thí dụ 2.5. Cho $\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$. Tìm $\|\mathbf{A}\|$, $\|\mathbf{A}^{-1}\|$, $\text{cond}(\mathbf{A})$.

$$\|\mathbf{A}\| = \max\{|1|+|2|, |3|+|4|\} = 7.$$

Theo công thức tính ma trận nghịch đảo,

$$\mathbf{A}^{-1} = \begin{bmatrix} -2 & 1 \\ \frac{3}{2} & -\frac{1}{2} \end{bmatrix},$$

nên

$$\|\mathbf{A}^{-1}\| = \max\{|-2| + |1|, |3/2| + |-1/2|\} = 3.$$

Vậy, $\text{cond}(\mathbf{A}) = \|\mathbf{A}\|\|\mathbf{A}^{-1}\| = 21$ ∘

Thí dụ 2.6. Ma trận $\mathbf{A} = \begin{bmatrix} 1 & -1 \\ 1 & -1 + 10^{-5} \end{bmatrix}$ "gần" kỳ dị vì

$$\mathbf{A}^{-1} = \begin{bmatrix} 1 - 10^5 & 10^5 \\ -10^5 & 10^5 \end{bmatrix},$$

$\|\mathbf{A}\| = 2$, $\|\mathbf{A}^{-1}\| = 2 \times 10^5$ và $\text{cond}(\mathbf{A}) = 4 \times 10^5$.

Định lý 2.1 khẳng định sự tồn tại một ma trận suy biến sai khác (tương đối) với $\mathbf{A}$ khoảng $1/\text{cond}(\mathbf{A}) = 2.5 \times 10^{-6}$. Mặc dù không hoàn toàn gần $\mathbf{A}$, ma trận đơn giản $\mathbf{S} = \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}$ là kỳ dị và $\dfrac{\|\mathbf{S} - \mathbf{A}\|}{\|\mathbf{A}\|} = 5 \times 10^{-6}$ ∘

Thí dụ 2.7. Giả sử ta giải phương trình $\mathbf{Ax} = \mathbf{b}$ trên một máy với $u = 5 \times 10^{-11}$ và nhận được

$$\mathbf{z} = \begin{bmatrix} 6.23415 \\ 18.6243 \end{bmatrix}, \quad \text{cond}(\mathbf{A}) = 1.0 \times 10^4.$$

Giả sử dữ liệu là chính xác để cho $\|\Delta\mathbf{A}\|/\|\mathbf{A}\| \approx 5 \times 10^{-10}$, từ (2.23), chặn trên của sai số tương đối là

$$\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|} \lessapprox 10^4 \times 5 \times 10^{-10} = 5 \times 10^{-6}.$$

Nếu dữ liệu nhập có sai số, chẳng hạn, $\|\Delta\mathbf{A}\|/\|\mathbf{A}\| \approx 10^{-6}$, $\|\Delta\mathbf{b}\|/\|\mathbf{b}\| \approx 10^{-6}$, thì chặn trên của sai số tương đối là

$$\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|} \lessapprox 10^4 \times 2 \times 10^{-6} = 0.02.$$

Lấy $\|\mathbf{x}\| \approx \|\mathbf{z}\| \approx 18.6$ chặn trên sai số tuyệt đối là 0.37. Vì vậy phân tích này cho

$$x_1 = 6.23 \pm 0.37, \quad x_2 = 18.62 \pm 0.37 \circ$$

2.5 Chương trình

Mục này giới thiệu hai function viết bằng ngôn ngữ lập trình Matlab. Sinh viên tự nghiên cứu và chạy thử cho các bài tập.

2.5.1 Factor

Mục đích: Phân tích ma trận A bằng cách dùng phép khử Gauss và đánh giá số điều kiện của nó. `Factor` được dùng chung với `Solve` để giải A*x=b.
Đối số nhập:

`A` - ma trận `neq` dòng và `cols` cột cần được phân tích.
Đối số xuất:

`A` - chứa ma trận tam giác trên `U` trong phần trên của nó và một phiên bản hoán vị của ma trận tam giác dưới `(I-L)`. Nhân tử hóa thỏa (ma trận hoán vị)*A=L*U.

`flag` - thông báo sự thành công hay thất bại. `flag = 0` chỉ sự thành công. Nếu `flag > 0`, một phần tử trụ bằng không và dừng tính toán.

`pivots` - bản ghi các hoán vị dòng. Đưa vào `pivots(neq)=(-1)^`(số của dòng thay đổi).

Khi `flag > 0`, định thức của `A` bằng 0 và

khi `flag = 0`, `det(A)=pivots(neq) * A(1,1) * ⋯ * A(neq,neq)`.
Đối số xuất tùy chọn:

`Cond` - khi `flag >= 0`, một đánh giá số điều kiện của `A` trong chuẩn vô cùng.

Factor.m

```
function [A,flag,pivots,Cond] = Factor(A)
[neq,cols] = size(A);
flag = 0;
pivots = zeros(neq,1);
pivots(neq) = 1;
if nargout == 4
   % Initialize Cond for A that is numerically singular.
    Cond = realmax;
   % Compute the infinity norm of A before the matrix is
   % overwritten by its factorization.
   Anorm = norm(A,inf);
end
if neq == 1
   if A(1,1) == 0
      flag = 1;
    elseif nargout == 4
        Cond = 1;
    end
    return
end
 % Gaussian elimination with partial pivoting.
for k = 1:neq-1
   % Determine the row m containing the largest element in
   % magnitude to be used as a pivot and its magnitude biggest.
    [biggest,occurred] = max(abs(A(k:neq,k)));
    m = occurred + k - 1;
     % If all possible pivots are zero, A is numerically singular.
    if biggest == 0
        flag = k;
        return
    end
    pivots(k) = m;
    if m ~= k
         % Interchange the current row k with the pivot row m.
        A([m k],k:neq) = A([k m],k:neq);
        pivots(neq) = - pivots(neq);
    end
   % Eliminate subdiagonal entries of column k.
    for i = k+1:neq
        t = A(i,k) / A(k,k);
        A(i,k) = - t;
```

```
        if t ~= 0
            A(i,k+1:neq) = A(i,k+1:neq) - t * A(k,k+1:neq);
        end
    end
end
if A(neq,neq) == 0
    flag = neq;
    return
end
if nargout == 4
   % Estimate the condition number of A by computing the infinity
   % norm of A directly and a lower bound for the norm of A^(-1).
   % A lower bound for the norm of A^(-1) is provided by the ratio
   % norm(y)/norm(d) for any vectors such that A*y = d and d  ~= 0.
   % A "large" ratio is obtained by computing y as one iteration of
   % inverse iteration for the smallest singular value of A, i.e.,
   % by solving for y such that (A'*A)*y = e.  This exploits the
   % fact that an LU decomposition of A can be used to solve the
   % linear system A'*d = e as well as A*y = d.  The entries of e
   % are +1 or -1 with the sign chosen during the computation of d
   % to increase the size of the entry of d and so make a "large"
   % lower bound for the norm of A^(-1) more likely.
   % Solve A'*d = e using the decomposition of A.
      d = zeros(neq,1);
      d(1) = -1 / A(1,1);
      for k = 2:neq
          t = A(1:k-1,k)' * d(1:k-1);
          if t < 0
              ek = -1;
          else
              ek =  1;
          end
          d(k) = -(ek + t) / A(k,k);
       end
      for k = neq-1:-1:1
          d(k) = d(k) + A(k+1:neq,k)'*d(k+1:neq);
          m = pivots(k);
          d([m k]) = d([k m]);
      end
   % Solve A*y = d.
      y = Solve(A,pivots,d);
```

```
   % Compute the infinity norms of the vectors.
      dnorm = norm(d,inf);
      ynorm = norm(y,inf);
      Cond = max(Anorm * ynorm / dnorm, 1);
end
```

2.5.2 Solve

Mục đích: Giải hệ neq phương trình tuyến tính theo `neq` ẩn bằng cách dùng phân tích LU nhận được bằng cách gọi `Factor`.
Đối số nhập:

`A` - output của `Factor`.

`pivots` - output của `Factor`.

`b` - vế phải, vectơ có độ dài `neq`.

Đối số xuất:

`x` - vectơ nghiệm có cùng kích thước như `b`.

Solve.m

```
function x = Solve(A,pivots,b)
neq = length(b);
x = b;
if neq == 1
     x(1) = x(1)/A(1,1);
else
   % Forward elimination.
     for k = 1:neq-1
         m = pivots(k);
          x([m k]) = x([k m]);
          x(k+1:neq) = x(k+1:neq) + A(k+1:neq,k)*x(k);
     end
     % Back substitution.
     x(neq) = x(neq) / A(neq,neq);
     for i = neq-1:-1:1
          x(i) = (x(i) - A(i,i+1:neq)*x(i+1:neq)) / A(i,i);
     end
end
```

2.6 Ma trận có cấu trúc đặc biệt

Hầu hết các bộ giải (solver) hệ phương trình đại số tuyến tính đều dựa trên phép khử Gauss với phép xoay cục bộ. Khi ma trận **A** có tính chất đặc biệt, ta có thể giảm bớt việc lưu trữ và chi phí cho việc giải hệ.

Khi quá trình khử không cần đến phép xoay cục bộ thì việc lưu trữ giảm đi cũng như quá trình tính sẽ nhanh hơn rất nhiều. Có hai loại ma trận, nói chung, không cần đến phép xoay cục bộ. Một ma trận $n \times n$ **A** được gọi là trội trên đường chéo (diagonally dominant), nếu với mỗi cột

$$|A_{jj}| \geq \sum_{i \neq j}^{n} |A_{ij}|.$$

Có thể thấy, với ma trận này ta không cần đến phép xoay cục bộ trong quá trình khử Gauss. Loại ma trận còn lại là ma trận đối xứng, $\mathbf{A} = \mathbf{A}^T$.

2.6.1 Ma trận băng

Nhắc lại rằng trong thuật toán khử Gauss, vòng lặp trong cùng có thể được bỏ đi khi nhân tử $t = 0$. Điều này phản ảnh sự kiện biến (tương ứng) không hiện diện trong phương trình và vì vậy không cần đến phép khử. Khi ma trận **A** "gần" với ma trận tam giác trên, việc kiểm tra nhân tử bằng không có thể tiết kiệm được lượng tính toán. Một loại ma trận cực kỳ quan trọng trong nhiều lãnh vực tính toán là ma trận băng. Ma trận $\mathbf{A} = [a_{ij}]$ được gọi là ma trận băng khi mọi phần tử khác không nằm trong một dải "dọc theo" đường chéo chính. Cụ thể, khi $a_{ij} = 0$ nếu $i - j > m_\ell$ và $j - i > m_u$, ma trận được gọi là có chiều rộng băng dưới m_ℓ, chiều rộng băng trên m_u, và chiều rộng băng $m_\ell + m_u + 1$. Một thí dụ của ma trận với $m_\ell = 2$, $m_u = 1$ là

$$\begin{bmatrix} * & * & 0 & 0 & 0 \\ * & * & * & 0 & 0 \\ * & * & * & * & 0 \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \end{bmatrix}$$

Ở đây $*$ chỉ phần tử có thể khác không. Khi tiến hành phép khử trên ma trận như vậy, tối đa m_ℓ phần tử phải được khử ở mỗi bước. Xem xét các

phần tử trên đường chéo chứng tỏ rằng nhiều phần tử không vẫn giữ bằng không. Thật vậy, phép xoay cục bộ sẽ để lại các số không trong $a_{ij}^{(k)}$ với $j - i > m_\ell + m_u$. Vì với các nhân tử không, ta có thể tăng tốc quá trình tính toán bằng cách nhận diện các phần tử không vẫn còn bằng không. Quan sát quan trọng khác là, bằng cách dùng sơ đồ lưu trữ đặc biệt, không cần lưu trữ các phần tử $a_{ij}^{(k)}$ với $i - j > m_\ell$, hay $j - i > m_\ell + m_u$. Mã cài đặt phép khử Gauss đặc biệt cho các ma trận băng có thể tìm thấy trong LINPACK cũng như LAPACK. Mặc dù có khó khăn trong việc cài đặt **A** theo cách lưu trữ đặc biệt, nhưng sự tiết kiệm có thể rất lớn. Các kết quả số là giống nhau, nhưng việc lưu trữ ở dạng băng xấp xỉ $n(2m_\ell + m_u)$ thay vì n^2. Khối lượng tính toán vào khoảng $nm_\ell(m_\ell + m_u)$ thay vì $n^3/3$, và có sự thuận lợi tương tự trong phép thế tiến và lùi.

Bây giờ ta xét một dạng khác của phép khử Gauss thuận tiện cho cách cài đặt ma trận băng. Giả sử phép phân tích $\mathbf{A} = \mathbf{LU}$ tồn tại. Trước hết chú ý rằng

$$a_{11} = \sum_{m=1}^{n} \ell_{1m} u_{m1} = \ell_{11} u_{11}$$

vì các ma trận là tam giác. Chọn $\ell_{11} \neq 0$ thì

$$u_{11} = a_{11}/\ell_{11}.$$

Với $i > 1$,

$$a_{i1} = \sum_{m=1}^{n} \ell_{im} u_{m1} = \ell_{i1} u_{11},$$

vì vậy

$$\ell_{i1} = a_{i1}/u_{11} \quad \text{với } i = 2, \ldots, n.$$

Cũng vậy, với $j > 1$,

$$a_{1j} = \sum_{m=1}^{n} \ell_{1m} u_{mj} = \ell_{11} u_{1j},$$

vậy,

$$u_{1j} = a_{1j}/\ell_{11} \quad \text{với } j = 2, \ldots, n.$$

Nói chung, mỗi lần ta lập một cột của **L** và một dòng của **U**. Giả sử ta đã tính được các cột $1, \ldots, k-1$ của **L** và các dòng $1, \ldots, k-1$ của **U**. Thì

$$a_{kk} = \sum_{m=1}^{n} \ell_{km} u_{mk} = \ell_{kk} u_{kk} + \sum_{m=1}^{k-1} \ell_{km} u_{mk}.$$

Các số hạng trong tổng cuối cùng là đã biết. Chọn ℓ_{kk}, thì

$$u_{kk} = \left(a_{kk} - \sum_{m=1}^{k-1} \ell_{km} u_{mk} \right) / \ell_{kk}.$$

Với $i > k$,

$$a_{ik} = \sum_{m=1}^{n} \ell_{im} u_{mk} = \ell_{ik} u_{kk} + \sum_{m=1}^{k-1} \ell_{im} u_{mk},$$

vì vậy

$$\ell_{ik} = \left(a_{ik} - \sum_{m=1}^{k-1} \ell_{im} u_{mk} \right) / u_{kk} \quad \text{với } i = k+1, \ldots, n.$$

Với $j > k$,

$$a_{kj} = \sum_{m=1}^{n} \ell_{km} u_{mj} = \ell_{kk} u_{kj} + \sum_{m=1}^{k-1} \ell_{km} u_{mj},$$

vậy,

$$u_{kj} = \left(a_{kj} - \sum_{m=1}^{k-1} \ell_{km} u_{mj} \right) / \ell_{kk} \quad \text{với } j = k+1, \ldots, n.$$

Nếu tất cả các phần tử trên đường chéo chính của **L** lấy bằng 1, thuật toán này là phép khử Gauss không dùng phép xoay cục bộ. Trong bàn luận của chúng ta về phép khử áp dụng cho ma trận băng **A**, ta thấy nhiều công việc và vùng lưu trữ có thể được tiết kiệm. Nếu **A** là ma trận băng với chiều rộng băng dưới m_ℓ và chiều rộng băng trên m_u, thì **L** cũng là ma trận băng với chiều rộng băng dưới m_ℓ và **U** là ma trận băng với chiều rộng băng trên m_u. Nếu ta chọn các phần tử chéo của **L** bằng 1, thì không cần lưu trữ chúng và giống như trường hợp ma trận đầy đủ, nhân tử **L** và **U** có thể được viết chồng lên **A** khi chúng được tính toán.

2.6.2 Ma trận ba đường chéo

Khi $m_\ell = m_u = 1$ ma trận các hệ số được gọi là ma trận ba đường chéo (tridiagonal matrix). Việc giải số phương trình đạo hàm riêng thường dẫn về việc giải các hệ phương trình gồm một số rất lớn các ẩn, có thể lên đến hàng ngàn. Khi ấy việc dùng Factor/Solve là không thích hợp, nhưng với thuật toán có mục đích đặc biệt thì gặp khó khăn này. Giả sử hệ ba đường chéo được viết dưới dạng

$$\begin{bmatrix} a_1 & c_1 & & & & \\ b_2 & a_2 & c_2 & & & 0 \\ & \ddots & \ddots & \ddots & & \\ 0 & & & b_{n-1} & a_{n-1} & c_{n-1} \\ & & & & a_n & c_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix}.$$

Khi không dùng phép xoay cục bộ, khử các b_i ta được

$$\begin{bmatrix} f_1 & c_1 & & & & \\ & f_2 & c_2 & & & 0 \\ & \ddots & \ddots & \ddots & & \\ 0 & & & & f_{n-1} & c_{n-1} \\ & & & & f_n & c_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_{n-1} \\ e_n \end{bmatrix}.$$

Như ta thấy các c_i không thay đổi. Bây giờ thiết lập công thức cho f_i, e_i, trước hết ta thấy $f_1 = a_1$, $e_1 = d_1$. Để khử b_2, không dùng phép xoay cục

bộ, nhân tử $m_2 = b_2/f_1$, suy ra:

$$\begin{aligned} f_2 &= a_2 - m_2 c_1, \\ c_2 &= c_2 - m_2 \cdot 0 = c_2, \\ e_2 &= d_2 - m_2 d_1. \end{aligned}$$

Để hoàn tất việc thiết lập ta dùng quy nạp. Giả sử rằng ta đã thiết lập được f_i và e_i đến dòng k. Thì ta có

$$\begin{array}{cccccc} 0 & f_k & c_k & 0 & | & e_k \\ 0 & b_{k+1} & a_{k+1} & c_{k+1} & | & d_{k+1} \end{array}$$

trên dòng k và $k+1$. Rõ ràng nhân tử là $m_{k+1} = b_{k+1}/f_k$. Phép khử dòng $k+1$ cho:

$$\begin{aligned} f_{k+1} &= a_{k+1} - m_{k+1} c_k, \\ c_{k+1} &= c_{k+1} - m_{k+1} \cdot 0 = c_{k+1}, \\ e_{k+1} &= d_{k+1} - m_{k+1} d_k. \end{aligned}$$

Việc lưu trữ có thể được tổ chức cực kỳ hiệu quả. Một ma trận tổng quát cấp n cần lưu trữ n^2 phần tử, nhưng một ma trận tam giác trên chỉ cần lưu trữ $3n-2$ phần tử khác không. Một sơ đồ tự nhiên là lưu trữ ba dải a_k, b_k và c_k như là ba vectơ chiều dà n. Ta có thể viết m_k lên b_k và f_k lên a_k khi chúng được tính; theo các bước thế tiến và lùi e_k và x_k có thể viết chồng lên d_k để cho chỉ cần thêm một vectơ chiều dài n. Thuật toán trên không dùng phép xoay cục bộ nên kết quả số có thể rất xấu. Với hệ ba đường chéo có một điều kiện đơn giản, thường thỏa mãn trong thức hành, bảo đảm kết quả thu được là tốt. Trước hết chú ý rằng nếu bất kỳ c_k hay b_k triệt tiêu, hệ có thể "bẻ" thành các hệ nhỏ hơn mà cũng là ba đường chéo. Suy ra, ta có thể giả sử c_k và b_k khác không với mọi k. Giả thiết then chốt là

$$\begin{aligned} |a_1| &> |b_2|, \\ |a_k| &\geq |b_{k+1}| + |c_{k+1}|, \quad k = 2, \ldots, n-1, \\ |a_n| &> |c_{n-1}|. \end{aligned}$$

Điều kiện này mạnh hơn điều kiện trội trên đường chéo, cho phép chứng tỏ ma trận không suy biến.

2.6.3 Ma trận đối xứng

Nếu ma trận $\mathbf{A}$ có thể phân tích thành $\mathbf{U}^T\mathbf{U}$ với $\mathbf{U}$ là ma trận tam giác trên, thì $\mathbf{A}$ là ma trận đối xứng, xác định dương. Ngược lại, một ma trận đối xứng,

xác định dương $\mathbf{A}$ có thể phân tích thành tích $\mathbf{U}^T\mathbf{U}$ với $\mathbf{U}$ là ma trận tam giác trên không suy biến. Bằng thủ tục trình bày ở trên ta có thể xác định $\mathbf{U}$. Ta phải có $\mathbf{L}^T = \mathbf{U}$ nên

$$a_{11} = \ell_{11}u_{11} = u_{11}^2,$$

suy ra $u_{11} = \sqrt{a_{11}}$, và như trước

$$u_{1j} = a_{ij}/u_{11} \quad j = 2, \ldots, n.$$

Bây giờ

$$a_{kk} = \sum_{m=1}^{n} \ell_{km}u_{mk} = \sum_{m=1}^{n} u_{mk}^2, \tag{2.24}$$

từ đây suy ra

$$u_{kk} = \left(a_{kk} - \sum_{m=1}^{k-1} u_{mk}^2\right)^{1/2}.$$

Rồi, cũng như trước,

$$u_{kj} = \left(a_{kj} - \sum_{m=1}^{k-1} u_{mk}u_{mj}\right)/u_{kk}, \quad j = k+1, \ldots, n.$$

Từ (2.24) ta thấy

$$a_{kk} \geq u_{mk}^2;$$

suy ra

$$|u_{mk}| \leq \sqrt{a_{kk}}$$

với mọi $m \geq k$, với mọi k. Điều này nói rằng các nhân tử không thể lớn đối với $\mathbf{A}$. Phép phân tích này gọi là phương pháp Cholesky hay phép phân tích căn bậc hai. Nó bảo vệ tốt hơn cấu trúc băng của ma trận.

2.7 Các phương pháp lặp

Trong mục này ta xét hai phương pháp lặp Jacobi và Gauss-Seidel cho phương trình $\mathbf{Ax} = \mathbf{b}$. Viết lại phương trình dưới dạng

$$\mathbf{Mx} = \mathbf{b} + (\mathbf{M} - \mathbf{A})\mathbf{x},$$

trong đó $\mathbf{M}$ là ma trận "gần" $\mathbf{A}$; tính dãy nghiệm xấp xỉ $\mathbf{x}^{(k)}$ bởi

$$\mathbf{M}\mathbf{x}^{(k+1)} = \mathbf{b} + (\mathbf{M} - \mathbf{A})\mathbf{x}^{(k)}.$$

Phép lặp Jacobi có dạng này với $\mathbf{M}$ là ma trận chéo với đường chéo chính là đường chéo chính của ma trận $\mathbf{A}$. Tương tự, phép lặp Gauss-Seidel ứng với trường hợp $\mathbf{M}$ là ma trận tam giác dưới gồm đường chéo chính của $\mathbf{M}$ và các phần tử bên dưới đường chéo chính của nó. Rõ ràng rất dễ giải các phương trình ở dạng này.

Qua giới hạn cả hai vế của phương trình xác định phép lặp, ta thấy nếu xấp xỉ hội tụ thì chúng phải hội tụ về $\mathbf{x}$. Sai số $\mathbf{e}^{(k)} = \mathbf{x} - \mathbf{x}^{(k)}$ thỏa

$$\mathbf{e}^{(k+1)} = \mathbf{M}^{-1}(\mathbf{M} - \mathbf{A})\mathbf{e}^{(k)},$$

suy ra

$$\|\mathbf{e}^{(k+1)}\| \leq \|\mathbf{M}^{-1}(\mathbf{M} - \mathbf{A})\|\|\mathbf{e}^{(k)}\|.$$

Nếu $\rho = \|\mathbf{M}^{-1}(\mathbf{M} - \mathbf{A})\| < 1$, bất đẳng thức này chứng tỏ quá trình lặp hội tụ với $\mathbf{x}^{(0)}$ bất kỳ. Sai số giảm do nhân tử ρ ở mỗi bước lặp, vì vậy nếu $\mathbf{M}$ càng gần $\mathbf{A}$ theo nghĩa ρ càng nhỏ thì quá trình hội tụ càng nhanh. Chú ý, lượng ρ là một loại sai số tương đối của xấp xỉ $\mathbf{A}$ bằng $\mathbf{M}$.

Thường ta tính nghiệm xấp xỉ liên tiếp nhờ lượng hiệu chỉnh $\delta^{(k+1}$, $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \delta^{(k+1)}$, trong đó $\delta^{(k+1}$ xác định bởi

$$\mathbf{M}\delta^{(k+1)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k)} = \mathbf{r}^{(k)}.$$

Ngay cả khi $\mathbf{A}$ là ma trận điều kiện xấu, $\mathbf{M}$ được chọn đủ gần $\mathbf{A}$ thì kết quả vừa thiết lập đảm bảo hội tụ.

Câu hỏi và bài tập

2.1. Viết chương trình, theo thuật toán khử Gauss, giải phương trình $\mathbf{A}\mathbf{x} = \mathbf{b}$.

2.2. Giải hệ

$$\begin{aligned} 0.461x_1 + 0.311x_2 &= 0.150 \\ 0.209x_1 + 0.141x_2 &= 0.068 \end{aligned}$$

dùng số thập phân chặt cụt ba chữ số. So sánh kết quả tìm được với nghiệm chính xác $x_1 = 1, x_2 = -1$.

2.3. Với hệ phương trình trong bài tập 2.2. Cho

$$\mathbf{y} = \begin{bmatrix} 0.999 \\ -1.001 \end{bmatrix}, \quad \mathbf{z} = \begin{bmatrix} 0.463 \\ -0.204 \end{bmatrix}.$$

Trong số học chính xác, tính các thặng dư $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{y}$, $\mathbf{s} = \mathbf{b} - \mathbf{A}\mathbf{z}$. Xấp xỉ tốt hơn có thặng dư nhỏ hơn?

2.4. Cho hệ phương trình tuyến tính

$$\begin{aligned} x_1 + \frac{1}{2}x_2 + \frac{1}{3}x_3 &= 1 \\ \frac{1}{2}x_1 + \frac{1}{3}x_2 + \frac{1}{4}x_3 &= 0 \\ \frac{1}{3}x_1 + \frac{1}{4}x_2 + \frac{1}{5}x_3 &= 0 \end{aligned}$$

a) Giải hệ bằng cách dùng số học chính xác.

b) Viết hệ dưới dạng ma trận dùng biểu diễn thập phân chặt cụt 2 chữ số.

c) Giải hệ ở câu b) không dùng phép xoay cục bộ (dùng số học như ở câu b).

d) Giải hệ ở câu b) dùng phép xoay cục bộ (dùng số học như ở câu b)).

e) Giải hệ ở câu b) dùng số học chính xác.

2.5. Cho hệ

$$\begin{aligned} x_1 + x_2 &= 2 \\ 10x_1 + 10^{18}x_2 &= 10 + 10^{18} \end{aligned}$$

Không dùng số học thập phân hơn 15-chữ số để tính toán câu a) và b).

a) Giải hệ bằng phương pháp khử Gauss với phép xoay cục bộ.

b) Chia mỗi dòng với $|a_{ij}|$ lớn nhất của nó rồi dùng phương pháp khử Gauss với phép xoay cục bộ.

c) Giải hệ bằng tay bằng bất kỳ phương pháp nào và dùng số học chính xác.

d) Dùng số học chính xác tính các thặng dư cho mỗi nghiệm tìm được ở các câu trên. Phương pháp nào có vẻ tốt hơn. Thặng dư có chỉ ra điều này không?

e) Tính cond(**A**).

2.6. Giả sử nghiệm tính toán của hệ phương trình không suy biến là

$$(-10.4631,\ 0.00318429,\ 3.79144,\ -0.000422790)$$

và số điều kiện là 1200.

a) Giả sử dữ liệu chính xác và đơn vị làm tròn $u = 10^{-6}$. Thì sai số tuyệt đối ($\pm$) trong mỗi thành phần nghiệm bằng bao nhiêu?

b) Câu hỏi tương tự với $\|\Delta\mathbf{A}\|/\|\mathbf{A}\| \approx 10^{-5}$, $\|\Delta\mathbf{b}\|/\|\mathbf{b}\| \approx 10^{-5}$.

2.7. Viết thuật toán và chương trình phân tích **LU** cho ma trận vuông.

2.8. Có bao nhiêu phép toán cần thiết để:

a) Thực hiện phân tích **LU** một ma trận vuông.

b) Giải phương trình $\mathbf{Ax} = \mathbf{b}$ khi các ma trận tam giác của phân tích là đã biết.

2.9. Giải hệ

$$\begin{aligned} x_1 + x_2 + x_3 &= 110.00 \\ x_1 + x_2 &= 78.33 \\ x_2 + x_3 &= 58.33 \end{aligned}$$

bằng cách dùng Factor/Solve. So sánh với nghiệm chính xác.

2.10. Cho ma trận

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9.01 \end{bmatrix}.$$

Dùng Factor/Solve tìm ma trận nghịch đảo.

2.11. Xét hệ tuyến tính

$$\begin{bmatrix} 0.217 & 0.732 & 0.414 \\ 0.508 & 0.809 & 0.376 \\ 0.795 & 0.886 & 0.338 \end{bmatrix} \mathbf{x} = \begin{bmatrix} 0.741 \\ 0.613 \\ 0.485 \end{bmatrix}$$

a) Giải tìm $\mathbf{x}$ bằng Factor/Solve.

b) Nếu các phần tử đưa vào của $\mathbf{A}$ và $\mathbf{b}$ có sai số tuyệt đối ± 0.0005, độ tin cậy của $\mathbf{x}$ như thế nào.

c) Tạo nhiễu ± 0.0005 trong các phần tử đưa vào của $\mathbf{A}$ và $\mathbf{b}$ để có $\mathbf{A} + \Delta\mathbf{A}$ và $\mathbf{b} + \Delta\mathbf{b}$. Giải $(\mathbf{A} + \Delta\mathbf{A})(\mathbf{x} + \Delta\mathbf{x}) = \mathbf{b} + \Delta\mathbf{b}$ để có $\mathbf{x} + \Delta\mathbf{x}$. Tính $\|\Delta\mathbf{x}\|/\|\mathbf{x}\|$. Điều nào có tương thích với câu b)? Sự thay đổi tương đối trong mỗi x_i?

2.8 Vấn đề nghiên cứu

Vấn đề 2 - Hệ phương trình có nghiệm không ổn định

Như đã thấy trong nhiều thí dụ số, giá trị thặng dư của nghiệm rất nhỏ nhưng nghiệm lại không gần với nghiệm chính xác. Sai số trong những trường hợp như vậy không phải do sự tích lũy của sai số làm tròn trong quá trình tính toán.

Theo [2], có nhiều cách để nhận ra một hệ phương trình có nghiệm không ổn định[1]:

- Sự thay đổi nhỏ trong các hằng số của hệ cũng dẫn đến sự thay đổi lớn của lời giải;

- Nếu hệ có ma trận các hệ số là $\mathbf{A}$, định nghĩa định thức chuẩn của $\mathbf{A}$ là

$$\text{norm}|\mathbf{A}| := \frac{|\mathbf{A}|}{\alpha_1\alpha_2\cdots\alpha_n},$$

trong đó $\alpha_k = \sum_{j=1}^{n} a_{kj}^2$, $k = 1, 2, \ldots, n$.

Nếu hệ có nghiệm không ổn định thì $\text{norm}|\mathbf{A}|$ nhỏ hơn so với $|\mathbf{A}|$.

Cũng trong tài liệu đã dẫn, tác giả giới thiệu phương pháp hoàn thiện lời giải cho hệ phương trình có nghiệm không ổn định.

Một cách tiếp cận khác, chỉnh hóa bài toán theo phương pháp chính quy hóa Tikhonov, xem [2].

Hãy tìm hiểu các phương pháp vừa nêu (cơ sở lý thuyết, phân tích sai số). Viết thuật toán, chương trình cho các phương pháp. Tính toán trên các

[1]Ngoài cách dùng số điều kiện.

thí dụ số điển hình để nêu nhận xét.

Tư liệu

[1] V.A. Ilyin and E.G. Poznyak, *Linear Algebra*, Mir Publisher, Moscow, 1986.

[2] Đặng Văn Liệt, *Giải tích số*, NXB ĐHQG TP. HCM, 2004.

Chương 3

Nội suy

Trong thực hành ta thường gặp các hàm mà giá trị của nó chỉ biết tại một số điểm (nhờ thí nghiệm) nhưng lại cần thiết phải tính tích phân, đạo hàm, hoặc thậm chí giá trị của hàm tại điểm mà dữ liệu thí nghiệm không cung cấp. Khi đó ta cần xấp xỉ hàm bằng một hàm đã biết mà giá trị tại các điểm đã cho trùng với dữ liệu thí nghiệm. Cũng có thể biểu thức xác định hàm quá phức tạp để có thể thực hiện việc tính toán. Một nguyên lý cơ bản của giải tích số: nếu ta không thể thực hiện một phép tính cơ bản với hàm đang xét, ta xấp xỉ nó bằng một hàm đơn giản hơn mà ta có thể thực hiện được phép tính.

Trong chương này ta xét vấn đề xấp xỉ hàm $f(x)$ bằng một hàm $F(x)$, trùng với $f(x)$ tại các điểm nào đó. Ta nói $F(x)$ nội suy (interpolate) $f(x)$ tại các điểm này. Quá trình xây dựng hàm $F(x)$ được gọi là phép nội suy (interpolation). Có nhiều loại hàm xấp xỉ, việc chọn lựa phụ thuộc vào bản chất của dữ liệu. Có lẽ hàm xấp xỉ đơn giản nhất là đa thức. Như đã biết, mọi hàm liên tục trên một khoảng hữu hạn đều có thể được xấp xỉ tốt bằng một đa thức. Hơn nữa, đa thức và các tỉ số của chúng (phân thức) là các hàm duy nhất có thể được tính trực tiếp nhờ máy tính. Vì lý do này đa thức được dùng không chỉ để nội suy mà còn là cơ sở cho hầu hết các phương pháp của giải tích số. Các spline đa thức, nghĩa là các hàm đa thức từng mảnh, là một công cụ rất hữu ích để xấp xỉ hàm và là đối tượng nghiên cứu chính của chương này.

3.1 Nội suy đa thức

Trong mục này hàm xấp xỉ $F(x)$ là đa thức và ta sẽ dùng ký hiệu gợi nhớ P_N thay cho F.

Nội suy Lagrange. Bài toán nội suy được phát biểu như sau.

Cho trước các cặp (x_j, f_j) với $j = 1, 2, \ldots, N$, trong đó các f_j là giá trị của hàm $f(x)$ (có thể không biết) tại các điểm x_j, $f_j = f(x_j)$. Tìm đa thức $P_N(x)$ sao cho

$$P_N(x_j) = f_j, \quad 1 \le j \le N. \tag{3.1}$$

Các điểm x_j, gọi là các điểm nội suy (interpolation points) hay các nút (nodes).

Một đa thức được xác định bởi các hệ số của nó, ở đây các điều kiện (3.1) cho các phương trình xác định các hệ số của đa thức nội suy. Ta có ngay $P_N(x)$ phải có bậc nhỏ hơn N. Dưới đây ta sẽ dùng ký hiệu $\mathcal{P}_N$ để chỉ tập hợp tất cả các đa thức có bậc nhỏ hơn N (không gian vectơ).

Định lý 3.1. *Cho N điểm phân biệt $\{x_1, x_2, \ldots, x_N\}$ có một và chỉ một đa thức $P_N(x)$ bậc nhỏ hơn N nội suy hàm cho trước $f(x)$ tại các điểm này.*

Chứng minh. Dạng Lagrange của đa thức nội suy:

$$P_N(x) = \sum_{k=1}^{N} f_k L_k(x), \tag{3.2}$$

trong đó các hàm $L_k(x)$ được chọn độc lập đối với $f(x)$. Vì $P_N(x)$ là đa thức bậc nhỏ hơn N với bất kỳ cách chọn $f_1, f_2, \ldots, f_N$ nên mỗi $L_k(x)$ cũng phải là đa thức có bậc nhỏ hơn N. Hơn nữa, để có $P_N(x_j) = f_j$ với $1 \le j \le N$, một lần nữa với cách chọn dữ liệu bất kỳ thì $L_k(x)$ phải thỏa

$$L_k(x_j) = \begin{cases} 0 & \text{nếu} \quad j \ne k \\ 1 & \text{nếu} \quad j = k. \end{cases}$$

Nghĩa là, các không điểm của $L_k(x)$ là các điểm x_j với $j \ne k$, như vậy $L_k(x)$ có dạng

$$L_k(x) = C \prod_{j=1, j\ne k}^{N} (x - x_j)$$

với hằng số C nào đó. Điều kiện $L_k(x_k) = 1$ cho

$$C = 1/\prod_{j=1, j\neq k}^{N} (x_k - x_j);$$

suy ra

$$L_k(x) = \prod_{j=1, j\neq k}^{N} \frac{x - x_j}{x_k - x_j}. \tag{3.3}$$

Để chứng minh $P_N(x)$ duy nhất, gọi $Q_N(x)$ là đa thức khác có bậc nhỏ hơn N thỏa $Q_N(x_j) = f_j$ với $1 \leq j \leq N$. Hiệu $D = P_N(x) - Q_N(x)$ cũng là đa thức có bậc nhỏ hơn N triệu tiêu tại N điểm x_j nên $D \equiv 0$ suy ra $P_N \equiv Q_N$. ■

Thí dụ 3.1. Cho $f(x) = \sin x$. Tìm $P_3(x)$ nội suy $f(x)$ tại ba điểm $\{0, \pi/2, \pi\}$. Các giá trị hàm tương ứng là $\{0, 1, 0\}$, như vậy

$$\begin{aligned} P_3(x) &= 0 \cdot L_1(x) + 1 \cdot L_2(x) + 0 \cdot L_3(x) \\ &= \frac{(x-0)(x-\pi)}{(\frac{\pi}{2}-0)(\frac{\pi}{2}-\pi)} \\ &= -\frac{4}{\pi^2} x(x-\pi). \end{aligned}$$

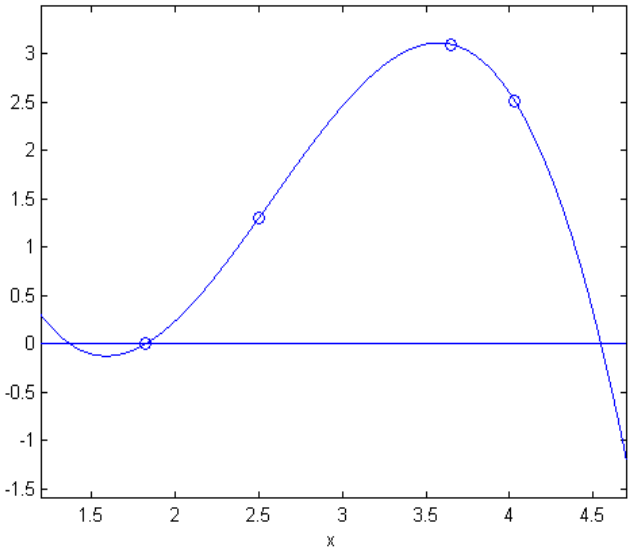
Thí dụ 3.2. Cho bảng dữ liệu

x	1.82	2.50	3.65	4.03
y	0.00	1.30	3.10	2.52

Xây dựng nội suy $P_4(x)$ theo bảng dữ liệu.

Dạng Lagrange của $P_4(x)$ là

$$\begin{aligned} P_4(x) &= 0L_1(x) + 1.30L_2(x) + 3.10L_3(x) + 2.52L_4(x) \\ &= 1.30\frac{(x-1.82)(x-3.65)(x-4.03)}{(0.68)(-1.15)(-1.53)} + 3.10\frac{(x-1.82)(x-2.50)(x-4.03)}{(1.83)(1.15)(-0.38)} \\ &\quad +2.52\frac{(x-1.82)(x-2.50)(x-3.65)}{(2.21)(1.53)(0.38)} \\ &= 1.09(x-1.82)(x-3.65)(x-4.03) - 3.88(x-1.82)(x-2.50)(x-4.03) \\ &\quad +1.96(x-1.82)(x-2.50)(x-3.65). \end{aligned}$$



Hình 3.1: Đồ thị hàm $P_4(x)$ trong thí dụ 3.2.

Nhận xét 3.1. Một cách tiếp cận khác để tính $P_N(x)$ là viết

$$p_N(x) = c_1q_1(x) + c_2q_2(x) + \ldots + c_Nq_N(x), \tag{3.4}$$

trong đó $q_1(x), q_2(x), \ldots, q_N(x)$ là các đa thức lập thành một cơ sở của $\mathcal{P}_N$. Bài toán nội suy dẫn về giải một hệ phương trình tuyến tính

$$c_1q_1(x_i) + c_2q_2(x_i) + \ldots + c_Nq_N(x_i) = f(x_i), \quad i = 1, 2, \ldots, N. \tag{3.5}$$

Đưa vào ma trận các hệ số $\mathbf{M}_q = [q_j(x_i)]_{i,j=1}^m$, các vectơ cột $\mathbf{c} = (c_1, c_2, \ldots, c_N)^T$, $\tilde{\mathbf{f}} = (f(x_1), f(x_2), \ldots, f(x_N))^T$. Hệ phương trình (3.5) có thể viết lại

$$\mathbf{M}_q \mathbf{c} = \tilde{\mathbf{f}};$$

suy ra $\mathbf{c} = \mathbf{M}_q^{-1}\tilde{\mathbf{f}}$. •

Thí dụ 3.3 (Một ứng dụng vào tích phân số)**.** Tìm một công thức để tính các tích phân có dạng

$$I = \int_0^1 x^{-1/2} f(x)dx$$

sao cho công thức này là chính xác khi $f \in \mathcal{P}_N$ và dùng các giá trị $f(x_i)$, $i = 1, 2, \ldots, N$.

Đặt $\mu_j = \int_0^1 x^{-1/2} q_j(x)dx$ và đưa vào vectơ dòng $\mu^T = (\mu_1, \mu_2, \ldots, \mu_N)$. Thì

$$I \approx \int_0^1 x^{-1/2} P_N(x)dx = \sum_{j=1}^N c_j \mu_j = \mu^T \mathbf{c} = \mu^T \mathbf{M}_q^{-1}\tilde{\mathbf{f}}.$$

Sai số trong nội suy đa thức. Bây giờ ta xét chất lượng của sự xấp xỉ. $P_N(x)$ xấp xỉ $f(x)$ tốt như thế nào? Việc tăng số nút nội suy có cải thiện được sự chính xác hay không? Định lý dưới đây giúp trả lời những câu hỏi này.

Định lý 3.2. *Giả sử $f(x)$ có đạo hàm đến cấp N trên khoảng I chứa các điểm nội suy $\{x_j\}_{j=1}^N$. Nếu $P_N(x)$ là đa thức bậc nhỏ hơn N nội suy $f(x)$ trên các dữ liệu này, thì với mỗi $x \in I$ có điểm $\xi_x \in I$ sao cho sai số trong nội suy đa thức là*

$$f(x) - P_N(x) = \frac{1}{N!} f^{(N)}(\xi_x) w_N(x), \tag{3.6}$$

trong đó

$$w_N(x) = \prod_{j=1}^N (x - x_j) \tag{3.7}$$

và

$$\min(x_1, \ldots, x_N, x) < \xi_x < \max(x_1, \ldots, x_N, x).$$

Chứng minh. Rõ ràng đẳng thức (3.6) đúng với $x = x_j$, $1 \le j \le N$. Với x không trùng với các điểm nội suy, định nghĩa hàm mới

$$G(t) = f(t) - P_N(t) - \frac{f(x) - P_N(x)}{w_N(x)} w_N(t).$$

Ta thấy hàm G có đạo hàm đến cấp N trên I và

$$G(x_j) = f_j - f_j - 0\frac{f(x) - P_N(x)}{w_N(x)} = 0, \quad 1 \le j \le N.$$

Cũng vậy, $G(x) = f(x) - P_N(x) - w_N(x)[f(x) - P_N(x)]/w_N(x) = 0$, như vậy G có $N+1$ không điểm phân biệt. Bởi định lý Rolle, G' có ít nhất N không điểm phân biệt trong I. Lập lại chứng minh này, G'' có ít nhất $N-1$ không điểm phân biệt trong I, ..., và $G^{(N)}$ có ít nhất một không điểm trong I. Ký hiệu không điểm này bằng ξ_x, ta thấy

$$0 = G^{(N)}(\xi_x) = f^{(N)}(\xi_x) - P_N^{(N)}(\xi_x) - w_N^{(N)}(\xi_x)\frac{f(x) - P_N(x)}{w_N(x)}.$$

Đa thức P_N có bậc nhỏ hơn N, vì vậy đạo hàm cấp N đồng nhất không. Đa thức $w_N(x)$ có bậc N với số hạng bậc cao nhất là t^N nên đạo hàm cấp N là $N!$. Tóm lại,

$$0 = f^{(N)}(\xi_x) - N!\frac{f(x) - P_N(x)}{w_N(x)},$$

điều này chứng minh định lý. ∎

Nếu $I = [a, b]$ và đặt

$$M_N = \max_{x \in I} |f^{(N)}(x)|,$$

thì ta có hai chặn trên của sai số nội suy:

$$|f(x) - P_N(x)| \leq \frac{M_N}{N!}|w_N(x)| \tag{3.8}$$

$$\leq \frac{M_N(b-a)^N}{N!} \quad \text{với } x \in (a,b). \tag{3.9}$$

Ước lượng sắc hơn của chặn thứ hai là

$$\max_{x\in I}|f(x) - P_N(x)| \leq \frac{M_N}{N!}\max_{x\in I}|w_N(x)|.$$

Thí dụ 3.4. Xét nội suy của hàm $f(x) = \sin x$ tại các điểm nội suy $\{0.0,\ 0.2,\ 0.4,\ 0.6,\ 0.8\}$. Bất đẳng thức (3.8) có thể được dùng để chặn sai số trong xấp xỉ $\sin(0.28)$ bởi $P_5(0.28)$. Vì

$$M_5 = \max_{t\in[0,0.8]}|\sin^{(5)} t| = \max_{t\in[0,0.8]}|\cos t| = 1,$$

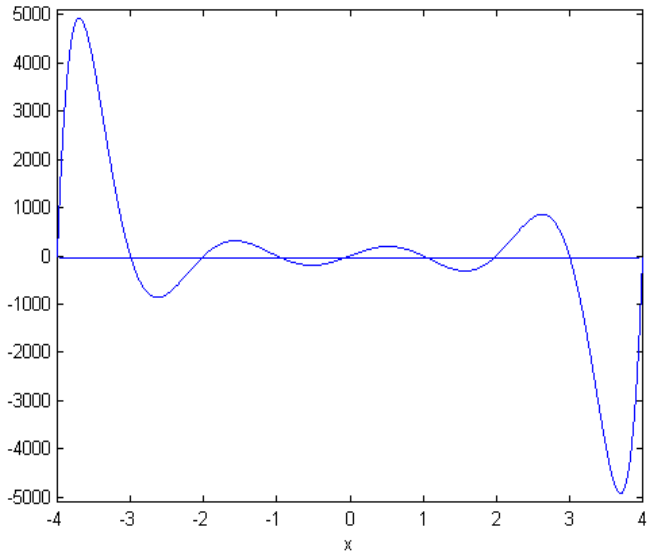
ta có chặn

$$|\sin(0.28) - P_5(0.28)| \leq |0.28(0.28-0.2)(0.28-0.4)(0.28-0.6)(0.28-0.8)|/5! = 3.7\times10^{-6}.$$

Đánh giá thực $P_5(0.28) = 0.2763591$, $\sin(0.28) = 0.2763556$, vậy sai số chính xác là -3.5×10^{-6}.

Định lý 3.2 và các chặn cho ta sự hiểu biết và những hướng dẫn khi thực hiện phép nội suy. Nhân tử $w_N(x)$ trong biểu thức sai số tăng ở gần các điểm cuối của khoảng dữ liệu và tăng rất nhanh khi x ra xa khỏi đoạn $[a,b]$; ở bậc cao hơn điều này càng đúng. Vì điều này chặn (3.8) tăng rất nhanh. Nhưng đẳng thức sắc hơn (3.6) chứng tỏ rằng ảnh hưởng này có được giảm thiểu với f và x cho trước bằng cách lấy nhân tử đạo hàm nhỏ hơn. Xấp xỉ $f(x)$ bằng $P_N(x)$ bên ngoài khoảng I được gọi là phép ngoại suy (extrapolation). Nói chung, rất nguy hiểm khi ngoại suy ở những điểm quá xa khoảng dữ liệu, đặc biệt khi dùng đa thức bậc cao. Mặc khác, $w_N(x)$ tương đối nhỏ khi x ở giữa các điểm nút. Và, tất nhiên, vì tính liên tục, sai số phải nhỏ khi x gần một nút. Hai nhận định này đề nghị, khi có thể, tốt nhất nội suy tại các nút xoay quanh x và càng gần x càng tốt.

Đồ thị của $w_9(x)$ trên $[-4,4]$ (hình 3.2) thể hiện dáng điệu định tính của nhân tử này trong biểu thức sai số. Hàm này phát triển cực kỳ nhanh



Hình 3.2: Đồ thị hàm $w_9(x)$ trên khoảng $[-4, 4]$.

bên ngoài khoảng chứa các nút và lớn ở các đầu mút, nhưng nó có độ lớn vừa phải ở giữa. Hình 3.5 thể hiện một nội suy đa thức bậc cao. Rõ ràng nội suy P_{12} không xấp xỉ tốt $f(x)$ trên toàn khoảng. Tuy nhiên, ở giữa khoảng này sự phù hợp xuất hiện là chấp nhận được. Dáng điệu định tính thấy ở đây với nội suy đa thức bậc cao có thể đoán được từ sự khảo sát nhân tử $w_N(x)$ trong sai số.

Thỉnh thoảng ta có thể đánh giá một hàm tại bất kỳ đâu ta muốn trong một khoảng, nhưng muốn xấp xỉ nó bởi một hàm đơn giản hơn để tính xấp xỉ đạo hàm hay tích phân của nó hay … Thì tự nhiên phải hỏi xem có cách chọn các nút nội suy tốt hay không theo nghĩa làm cho

$$\max_{a \le x \le b} |w_N(x)| \tag{3.10}$$

nhỏ. Câu trả lời là có. Các điểm

$$x_j = \frac{b+a}{2} + \frac{b-a}{2} \cos \frac{(2j-1)\pi}{2N}, \quad j = 1, \ldots, N, \tag{3.11}$$

gọi là các điểm Chebyshev, làm cho (3.10) nhỏ như có thể. Ta sẽ xét chi tiết về xấp xỉ này trong mục tiếp theo.

Nội suy Hermite. Nếu tại các nút x_j ta biết thêm giá trị của đạo hàm cấp

một f'_j thì có thể nội suy hàm $f(x)$ theo giá trị hàm và giá trị đạo hàm tại các nút. Có nhiều cách thực hiện phép nội suy như vậy, ở đây ta chỉ xét nội suy Hermite (Hermite interpolation). Giả sử ta có các giá trị f_j và đạo hàm f'_j tại các nút x_j với $j = 1, \ldots, N$. Với $2N$ giá trị độc lập có thể thấy đa thức nội suy có bậc $2N - 1$. Hơn nữa, có thể chứng minh các đa thức cơ sở $\phi_k(x)$, $\psi_k(x)$ có bậc bé hơn $2N$ thỏa

$$\begin{aligned} \phi_k(x) &= \begin{cases} 0 & \text{nếu } j \neq k \\ 1 & \text{nếu } j = k, \end{cases} \qquad \phi'_k(x_j) = 0 \quad \text{với mọi } j \\ \psi'_k(x) &= \begin{cases} 0 & \text{nếu } j \neq k \\ 1 & \text{nếu } j = k, \end{cases} \qquad \psi_k(x_j) = 0 \quad \text{với mọi } j, \end{aligned}$$

được cho bởi

$$\begin{aligned} \phi_k(x) &= [1 - 2L'_k(x_k)(x - x_k)]L_k^2(x), \\ \psi_k(x) &= (x - x_k)L_k^2(x). \end{aligned} \tag{3.12}$$

Ở đây $L_k(x)$ là các đa thức nội suy cơ sở. Hiển nhiên, đa thức nội suy:

$$P(x) = \sum_{k=1}^{N} f_k \phi_k(x) + \sum_{k=1}^{N} f'_k \psi_k(x)$$

thỏa

$$P(x_j) = f_j, \quad P'(x_j) = f'_j, \quad j = 1, \ldots, N.$$

Ta cũng có một kết quả tương tự như (3.6) cho nội suy Hermite:

$$f(x) - P(x) = \frac{1}{2N!} f^{(2N)}(\xi_x) w_N^2(x). \tag{3.13}$$

Trong phép giải số các phương trình vi phân thường ta dùng đa thức bậc năm (quintics) để nội suy f và f' tại ba điểm. Trong chương này ta dùng

các đa thức bậc ba để nội suy f và f' tại hai điểm. Để dùng đến sau này ta biểu diễn nội suy Hermite bậc ba ở dạng khác. Nếu ta viết

$$H(x) = a + b(x - x_n) + c(x - x_n)^2 + d(x - x_n)^3 \tag{3.14}$$

và đòi hỏi là

$$\begin{aligned} H(x_n) = f_n, \quad H'(x_n) = f'_n, \\ H(x_{n+1}) = f_{n+1}, \quad H'(x_{n+1}) = f'_{n+1} \end{aligned}$$

thì dễ dàng chứng tỏ rằng với $h = x_{n+1} - x_n$

$$\begin{aligned} a &= f_n, && (3.15)\\ b &= f'_n, && (3.16)\\ c &= [3(f_{n+1} - f_n)/h - 2f'_n - f'_{n+1}]/h, && (3.17)\\ d &= [f'_n + f'_{n+1} - 2(f_{n+1} - f_n)/h]/h^2. && (3.18) \end{aligned}$$

3.2 Các chặn sai số

Trong mục này một số kết quả hữu ích được bàn luận và một vài kết quả về sai số liên quan đến xấp xỉ đạo hàm bằng đạo hàm của đa thức nội suy.

Một số đo cách $P_N(x)$ xấp xỉ $f(x)$ trên khoảng $[a, b]$ là sai số tệ nhất

$$\|f - P_N\| = \max_{a \le x \le b} |f(x) - P_N(x)|.$$

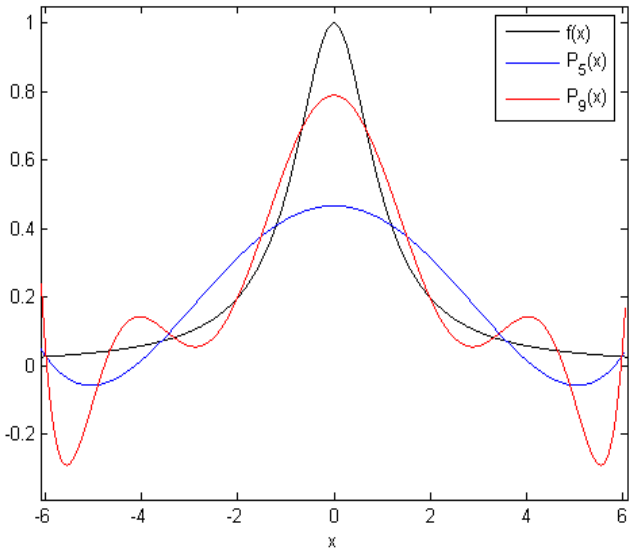
Một định lý cơ bản của Weierstrass phát biểu rằng hàm bất kỳ $f(x)$ liên tục trên khoảng hữu hạn $[a, b]$ có thể xấp xỉ tốt tùy ý bằng một đa thức, nghĩa là, cho trước $\epsilon > 0$, tồn tại đa thức $P(x)$ sao cho $\|f - P\| < \epsilon$. Thật hợp lý khi cho rằng càng nhiều điểm nội suy hơn sẽ cho xấp xỉ tốt hơn. Chặn (3.9) chứng tỏ rằng nếu M_N không tăng nhanh khi $N \to \infty$, nội suy P_N xấp xỉ f tốt tùy ý. Đáng tiếc, điều này không đúng cho mọi hàm liên tục. Một kết quả do Faber chỉ ra rằng với tập các nút cho trước bất kỳ $\{x_1^{(1)}\}$, $\{x_1^{(2)}, x_2^{(2)}\}$, ...trong $[a, b]$, tồn tại một hàm $f(x)$ liên tục trong $[a, b]$ sao cho các nội suy $P_N(x)$ có bậc nhỏ hơn N xác định bởi

$$P_N(x_i^{(N)}) = f(x_i^{(N)}), \quad i = 1, \ldots, N,$$

không có ngay cả $\|f - P_N\|$ bị chặn khi $N \to \infty$. Hàm Runge

$$f(x) = \frac{1}{1 + x^2} \tag{3.19}$$

trên $[-5, 5]$ là một thí dụ cổ điển. Có vẻ hiển nhiên nội suy một hàm trơn như vậy tại càng nhiều điểm nội suy cách đều sẽ dẫn đến hội tụ, nhưng thực tế cho thấy ngay cả với N vừa phải, nội suy hoàn toàn không chấp nhận được.



Hình 3.3: Đồ thị hàm Runge và các đa thức nội suy $P_5(x)$ và $P_9(x)$.

Nếu có thể nội suy tại các nút tốt (3.11), thì hóa ra phép nội suy là cách tốt để xấp xỉ $f(x)$ bằng đa thức bậc thấp nhất có thể. Thực tế là hàm Runge có thể được xấp xỉ một cách hoàn toàn chính xác bởi đa thức nội suy tại các điểm Chebyshev. Trong trường hợp tổng quát, tồn tại một đa thức $P_N^*(x)$ bậc nhỏ hơn N xấp xỉ $f(x)$ tốt nhất trên $[a, b]$ theo nghĩa $\|f - P_N^*\|$ cho giá trị nhỏ nhất của $\|f - P\|$ với mọi đa thức P bậc nhỏ hơn N. Cho $P_N(x)$ nội suy $f(x)$ tại các nút $x_1, \ldots, x_N$ trong $[a, b]$. Với bất kỳ x,

$$f(x) - P_N(x) = f(x) - P_N^*(x) + P_N^*(x) - P_N(x).$$

Bây giờ $P_N^*(x) - P_N(x)$ là một đa thức bậc nhỏ hơn N, vậy

$$P_N^*(x) - P_N(x) = \sum_{k=1}^{N} (P_N^*(x_k) - P_N(x_k)) L_k(x)$$

vì nội suy Lagrange tại N điểm là chính xác với các đa thức như vậy. Dùng sự kiện là $P_N(x_k) = f_k$, ta thấy rằng

$$\begin{aligned} |f(x) - P_N(x)| &\leq |f(x) - P_N^*(x)| + \sum_{k=1}^{N} |P_N^*(x_k) - f_k||L_k(x)| \\ &\leq \|f - P_N^*\| \left(1 + \max_{a \leq x \leq b} \sum_{k=1}^{N} |L_k(x)|\right), \end{aligned}$$

và rồi

$$\|f - P_N\| = \max_{a \leq x \leq b} |f(x) - P_N(x)| \leq \|f - P_N^*\| \left(1 + \max_{a \leq x \leq b} \sum_{k=1}^{N} |L_k(x)|\right),$$

Bất đẳng thức này liên hệ sai số của P_N với sai số của đa thức xấp xỉ tốt nhất P_N^* bởi một thừa số

$$1 + \max_{a \leq x \leq b} \sum_{k=1}^{N} |L_k(x)|,$$

mà được cho trước chỉ nhờ các điểm nội suy. Đặc biệt, nếu các nút nội suy là điểm Chebyshev (3.11) thì (xem [11])

$$1 + \max_{a \leq x \leq b} \sum_{k=1}^{N} |L_k(x)| \leq 1 + \frac{1}{N} \sum_{k=1}^{N} \operatorname{tg} \frac{(2k-1)\pi}{4N}.$$

Điều ngạc nhiên là với bậc N cỡ trung bình thì chặn này là quá nhỏ. Với $N \leq 20$, nó ít hơn 4. Như vậy

$$\|f - P_N^*\| \leq \|f - P_N\| \leq 4\|f - P_N^*\|$$

với mọi $N \leq 20$. Các đa thức nội suy như vậy được xây dựng dễ dàng và là tốt như có thể.

Phép nội suy không đạt hiệu quả như vậy khi các nút nội suy không thể được chọn, và như định lý của Faber chỉ ra, nội suy cấp cao có thể không hoàn toàn thỏa mãn. Thường trong thực hành nội suy đa thức cấp cao thể hiện các dao động có biên độ lớn ngay cả khi dữ liệu lấy từ một hàm trơn.

Xấp xỉ đạo hàm bằng đạo hàm của đa thức nội suy. Nội suy đa thức là công cụ cơ bản trong giải tích số. Như một thí dụ, các đạo hàm của nội suy $P_N(x)$ của $f(x)$ có thể được dùng để xấp xỉ các đạo hàm của $f(x)$. Một chứng minh tương tự như định lý 3.2 (xem [8]) có thể được dùng để chứng tỏ với bất kỳ $r < N$

$$f^{(r)}(x) - P_N^{(r)}(x) = \frac{f^{(N)}(\xi_x)}{(N-r)!} \prod_{k=1}^{N-r} (x - \zeta_k),$$

trong đó các điểm $\{\zeta_k\}$ được biết là phân biệt và thỏa

$$x_k < \zeta_k < x_{k+r}, \quad 1 \leq k \leq N - r.$$

Điểm ζ_k phụ thuộc x và nằm trong cùng khoảng I như ξ_x trong định lý 3.2. Như một hệ quả,

$$|f^{(r)}(x) - P_N^{(r)}(x)| \leq \frac{M_N (x_N - x_1)^{N-r}}{(N-r)!} \tag{3.20}$$

miễn là $x_1 \leq x \leq x_N$. Dạng Lagrange của đa thức nội suy là tiện lợi để thiết lập công thức vi phân số. Để xấp xỉ đạo hàm của $f(x)$ tại điểm z, cho trước các giá trị f_k tại các điểm $\{x_1, \ldots, x_N\}$, ta đơn giản thiết lập nội suy, đạo hàm nó, và đánh giá nó tại z:

$$f^{(r)}(z) \approx P_N^{(r)}(z) = \sum_{k=1}^{N} f_k L_k^{(r)}(z).$$

Bởi vì các hệ số trong biểu thức này chỉ phụ thuộc vào các nút, ta có ở đây một công thức mà có thể dùng cho bất kỳ hàm $f(x)$ nào.

Nhận xét 3.2. Các chặn sai số như (3.20) có thể được thiết lập cho các đa thức Hermite (xem [2]). Dùng ký hiệu như trên, nếu f có đạo hàm cấp bốn với x bất kỳ trong khoảng $[x_n, x_n + h]$, thì với $M_4 = \max_{x_n \le x \le x_n+h} |f^{(4)}(x)|$,

$$|f(x) - H(x)| \le \frac{1}{384} M_4 h^4, \tag{3.21}$$

$$|f'(x) - H'(x)| \le \frac{\sqrt{3}}{216} M_4 h^3, \tag{3.22}$$

$$|f''(x) - H''(x)| \le \frac{1}{12} M_4 h^2, \tag{3.23}$$

$$|f'''(x) - H'''(x)| \le \frac{1}{2} M_4 h^4. \tag{3.24}$$

3.3 Dạng Newton của đa thức nội suy

Cách biểu diễn đa thức nội suy dưới dạng Lagrange (3.2) tuy có tiện lợi vì sự phụ thuộc vào các f_j đơn giản, nhưng cách thức các nút x_j xuất hiện lại không đơn giản chút nào. Đặc biệt, nó không tiện lợi khi chúng ta không biết trước bậc của đa thức xấp xỉ. Vì vậy một dạng khác do Newton đề nghị thường được dùng hơn trong thực hành. Dẫn chứng: hai loại phương pháp số được dùng rộng rãi khi giải bài toán Cauchy cho phương trình vi phân thường là (1) các phương pháp Adams, và (2) công thức sai phân lùi (các phương pháp Gear). Trong các phương pháp này, ở mỗi bước giải (lặp), các thuật toán tìm bậc thích hợp nhất cho đa thức nội suy. Vì thế những thuật toán như vậy dùng dạng Newton của đa thức. Mặc dù phương pháp thiết lập dạng Newton mới nhìn xem ra thật "kinh khủng".

Tỉ sai phân. Một thủ thuật cơ bản của giải tích số là đánh giá sai số về lượng (đánh giá hậu nghiệm) bằng cách so sánh nó với một đại lượng được cho là chính xác hơn. Nếu $P_N(x)$ nội suy tại các nút $\{x_1, \ldots, x_N\}$ và $P_{N+1}(x)$ là nội suy tại các nút $\{x_1, \ldots, x_N, x_{N+1}\}$, thì trong các trường hợp phù hợp đa thức sau xấp xỉ $f(x)$ tốt hơn và $f(x) - P_N(x) \approx P_{N+1}(x) - P_N(x)$. Nếu ta không biết bậc thích hợp, điều này đề nghị một cách tiến hành. Bắt đầu bằng đa thức hằng $P_1(x) = f_1$ nội suy tại x_1. Từ $P_N(x)$ đã tính, tính P_{N+1} và dùng nó để đánh giá sai số của $P_N(x)$. Nếu sai số đánh giá quá lớn, tăng bậc bằng cách nội suy thêm tại nút khác và lặp lại quá trình. Thủ tục này

là cơ sở của dạng Newton của đa thức nội suy.

Với mỗi n, đa thức nội suy $P_n(x)$ được xây dựng như là một "hiệu chỉnh" $P_{n-1}(x)$. Vì $P_{n-1}(x)$ có bậc nhỏ hơn $n-1$ và $P_n(x)$ có bậc tối đa là $n-1$, hiệu của chúng phải là đa thức có bậc tối đa bằng $n-1$:

$$P_n(x) = P_{n-1}(x) + Q_n(x). \tag{3.25}$$

Đa thức $P_n(x)$ nội suy tại $x_1, \ldots, x_{n-1}$ giống như $P_{n-1}(x)$, như vậy với $j = 1, \ldots, n-1$,

$$f_j = P_n(x_j) = P_{n-1}(x_j) + Q_n(x_j) = f_j + Q_n(x_j).$$

Điều này ám chỉ $x_1, \ldots, x_{n-1}$ là các nghiệm của $Q_n(x)$. Vì bậc của nó tối đa bằng $n-1$, $Q_n(x)$ phải có dạng

$$Q_n(x) = c_n(x - x_1)(x - x_2)\cdots(x - x_{n-1})$$

với c_n là hằng số nào đó. Đa thức $P_n(x)$ cũng nội suy tại x_n:

$$f_n = P_n(x_n) = P_{n-1}(x_n) + Q_n(x_n) = P_{n-1}(x_n) + c_n \prod_{j=1}^{n-1}(x_n - x_j).$$

Vì các nút là phân biệt nên không có nhân tử $(x_n - x_j)$ nào bằng không, và

$$c_n = \frac{f_n - P_{n-1}(x_n)}{\prod_{j=1}^{n-1}(x_n - x_j)}. \tag{3.26}$$

Các hệ thức (3.25) và (3.26) cùng với $P_1(x) = f_1$ cho dạng Newton của đa thức nội suy. Các hệ số c_n được gọi là tỉ sai phân cấp $(n-1)$ ($(n-1)$st order divided difference) trên các điểm $x_1, \ldots, x_n$, ký hiệu

$$c_n = f[x_1, \ldots, x_n].$$

Theo ký hiệu này dạng tỉ sai phân Newton là

$$\begin{aligned} P_N(x) &= f[x_1] + f[x_1, x_2](x - x_1) + f[x_1, x_2, x_3](x - x_1)(x - x_2) + \ldots \\ &\quad + f[x_1, \ldots, x_N] \prod_{j=1}^{N-1}(x - x_j). \end{aligned} \tag{3.27}$$

Rõ ràng từ (3.27) ta thấy hệ số dẫn đầu (hệ số của số hạng bậc cao nhất) của $P_N(x)$ là $f[x_1, \ldots, x_N]$. Một số tác giả dùng điều này như là định nghĩa của tỉ sai phân cấp $(N-1)$.

Định lý dưới đây cho mối liên hệ giữa tỉ sai phân cấp n với một cặp các tỉ sai phân cấp $(n-1)$. Liên hệ này dẫn đến một thuật toán tính c_n thuận tiện hơn (3.26).

Định lý 3.3. *Với các nút phân biệt $\{x_j\}$ và $k > i$ bất kỳ,*

$$f[x_i, \ldots, x_{k-1}, x_k] = \frac{f[x_{i+1}, \ldots, x_k] - f[x_i, \ldots, x_{k-1}]}{x_k - x_i} \tag{3.28}$$

và

$$f[x_i] = f_i.$$

Chứng minh. Cho $R_1(x)$ là đa thức bậc nhỏ hơn $k-i$ nội suy $f(x)$ trên $x_{i+1}, \ldots, x_k$ và cho $R_2(x)$ là đa thức bậc nhỏ hơn $k-i$ nội suy $f(x)$ trên $x_i, \ldots, x_{k-1}$. Đa thức

$$S(x) = \frac{(x_k - x)R_2(x) + (x - x_i)R_1(x)}{x_k - x_i} \tag{3.29}$$

có bậc tối đa hơn $R_1(x)$, $R_2(x)$ một bậc. Theo đó, bậc của nó nhỏ hơn $k-i+1$. Với $j = i+1, \ldots, k-1$,

$$S(x_j) = \frac{(x_k - x_j)R_2(x_j) + (x_j - x_i)R_1(x_j)}{x_k - x_i} = \frac{(x_k - x_j)f_j + (x_j - x_i)f_j}{x_k - x_i} = f_j,$$

vì vậy $S(x)$ nội suy $f(x)$ trên $x_{i+1}, \ldots, x_{k-1}$. Hơn nữa, $S(x_i) = f_i$ và $S(x_k) = f_k$. Bởi định lý 3.1, $S(x)$ là đa thức bậc nhỏ hơn $k-i+1$ nội suy $f(x)$ trên tất cả dữ liệu. Kết quả (3.28) biểu diễn một cách đơn giản sự kiện hệ số dẫn đầu của vế trái (3.29) bằng hệ số dẫn đầu của vế phải. ■

Để minh họa cách dùng định lý này, ta xây dựng bảng tỉ sai phân. Giả sử ba dòng và cột của tỉ sai phân đã được tính và được viết dưới dạng ma trận tam giác dưới như sau:

$$\begin{array}{llll} x_1 & f[x_1] & & \\ x_2 & f[x_2] & f[x_1, x_2] & \\ x_3 & f[x_3] & f[x_2, x_3] & f[x_1, x_2, x_3]. \end{array}$$

Để thêm vào dòng mới tương ứng với nút x_4, bắt đầu bằng dữ liệu $f[x_4] = f_4$. Rồi

$$\begin{aligned} f[x_3, x_4] &= \frac{f[x_4] - f[x_3]}{x_4 - x_3} \\ f[x_2, x_3, x_4] &= \frac{f[x_3, x_4] - f[x_2, x_3]}{x_4 - x_2} \\ f[x_1, x_2, x_3, x_4] &= \frac{f[x_2, x_3, x_4] - f[x_1, x_2 x_3]}{x_4 - x_1} \end{aligned}$$

Chú ý cách thức tính toán ở đây

$$\begin{array}{llllllll} x_1 & f[x_1] & & & & & & \\ x_2 & f[x_2] & & f[x_1, x_2] & & & & \\ x_3 & f[x_3] & & f[x_2, x_3] & & f[x_1, x_2, x_3] & & \\ & & \searrow & & \searrow & & \searrow & \\ x_4 & f[x_4] & \to & f[x_3, x_4] & \to & f[x_2, x_3, x_4] & \to & f[x_1, x_2, x_3, x_4]. \end{array}$$

Tổng quát, cột đầu của bảng tỉ sai phân là x_j, thứ hai là f_j, kế tiếp là tỉ sai phân thứ nhất, và vân vân. Bảng tỉ sai phân cung cấp một phương sách tiện lợi để xây dựng các tỉ sai phân cần thiết: các hệ số của đa thức nội suy là các đại lượng dọc theo đường chéo.

Thí dụ 3.5. Với các dữ liệu từ thí dụ 3.2, trước hết lập bảng

x_1	f_j	$f[\ ,\ ]$	$f[\ ,\ ,\ ]$	$f[\ ,\ ,\ ,\ ]$
1.82	0.00			
2.50	1.30	$\frac{1.30-1.00}{2.50-1.82} = 1.91$		
3.65	3.10	$\frac{3.10-1.30}{3.65-2.50} = 1.56$	$\frac{1.56-1.91}{3.65-1.82} = -0.19$	
4.03	2.52	$\frac{2.52-3.10}{4.03-3.65} = -1.53$	$\frac{-1.53-1.56}{4.03-2.50} = -2.02$	$\frac{-2.02+0.19}{4.03-1.82} = -.83$

Rồi theo (3.27),

$$P_4(x) = 0.0+1.91(x-1.82)-0.19(x-1.82)(x-2.50)-0.83(x-1.82)(x-2.50)(x-3.65).$$

Để tính toán hiệu quả ta nên đánh giá ở dạng xếp lồng vào nhau

$$P_4(x) = (x - 1.82)\{1.91 + (x - 2.50)[-0.19 - 0.83(x - 3.65)]\}. \circ$$

Thuật toán tìm dạng Newton. Thuật toán gồm hai phần. Trước hết tính các tỉ sai phân cần cho các hệ số của $P_N(x)$. Không cần thiết phải lưu trữ toàn bộ bảng vì ta có thể dùng vectơ `c` để lưu trữ mục nhập (entry) trong dòng hiện hành `j` miễn là mỗi lần ta tính một đường chéo (và thực hiện các phép tính theo thứ tự chính xác):

```
c(N)=f(N);
for j=N-1:-1:1
    c(j)=f(j);
    for k=j+1:N
        c(k)=(c(k)-c(k-1))/(x(k)-x(j));
    end
end
```

Để dễ hiểu ta xem "nội dung" vectơ `c` ở mỗi bước tính `j`, trường hợp `N=4`, và so sánh với bảng tỉ sai phân ở trên

j	vectơ `c`			
	1	2	3	4
4				$f[x_4]$
3			$f[x_3]$	$f[x_3, x_4]$
2		$f[x_2]$	$f[x_2, x_3]$	$f[x_2, x_3, x_4]$
1	$f[x_1]$	$f[x_1, x_2]$	$f[x_1, x_2, x_3]$	$f[x_1, x_2, x_3, x_4]$

Ta thấy, ở bước cuối cùng (`j=1`), vectơ `c` chứa các hệ số của đa thức nội suy.

Ngay khi các hệ số này được tính xong, phần thứ hai của thuật toán là đánh giá $P_N(x)$ tại x cho trước:

```
P(N)=c(N);
for k=N-1:-1:1
    P(N)=P(N)*(x-x(k))+c(k);
end
```

Liên hệ giữa tỉ sai phân và các đạo hàm của $f(x)$. Áp dụng định lý 3.2 cho $P_{n-1}(x)$ với $x = x_n$ ta có

$$f(x_n) - P_{n-1}(x_n) = \frac{f^{(n-1)}(\xi_n)}{(n-1)!} \prod_{j=1}^{n-1} (x_n - x_j),$$

trong đó

$$\min(x_1, \ldots, x_n) < \xi_n < \max(x_1, \ldots, x_n).$$

Tuy nhiên, ta cũng có

$$f(x_n) - P_{n-1}(x_n) = P_n(x_n) - P_{n-1}(x_n) = c_n \prod_{j=1}^{n-1}(x_n - x_j).$$

Cân bằng hai biểu thức chứng tỏ rằng

$$f[x_1, \ldots, x_n] = \frac{f^{(n-1)}(\xi_n)}{(n-1)!} \tag{3.30}$$

với điểm ξ_n nằm trong miền dữ liệu $x_1, \ldots, x_n$.

Nhận xét 3.3. 1) Với cách tính hiệu quả tỉ sai phân trình bày trên và (3.30) ta có một cách xấp xỉ đạo hàm của hàm $f(x)$ mà chỉ biết giá trị của nó tại các điểm nào đó.

2) Đẳng thức (3.30) cho ta hiểu biết tốt hơn về việc đánh giá sai số mà ta thường dùng. Theo định lý 3.2,

$$f(x) - P_N(x) = \frac{1}{N!} f^{(N)}(\xi) \prod_{j=1}^{N}(x - x_j).$$

Ta vừa thấy rằng

$$P_{N+1}(x) - P_N(x) = f[x_1, \ldots, x_{N+1}] \prod_{j=1}^{N}(x - x_j) = \frac{1}{N!} f^{(N)}(\eta) \prod_{j=1}^{N}(x - x_j).$$

So sánh hai biểu thức này cho thấy nếu $f^{(N)}$ không thay đổi nhiều trên miền dữ liệu, sai số của $P_N(x)$ có thể được đánh giá bằng cách so sánh nó với $P_{N+1}(x)$.

3) Dạng (3.27) liên hệ mật thiết với chuỗi Taylor của $f(x)$ đối với điểm x_1:

$$f(x_1) + \frac{f^{(1)}(x_1)}{1!}(x - x_1) + \frac{f^{(2)}(x_1)}{2!}(x - x_1)^2 + \ldots + \frac{f^{(N-1)}(x_1)}{(N-1)!}(x - x_1)^{N-1} + \ldots$$

Như một hệ quả của (3.30), dạng Newton của đa thức nội suy $P_N(x)$ trở thành đa thức Taylor bậc $N-1$ khi các nút $x_2, \ldots, x_n$ tiến tới x_1.

3.4 Định giá sự chính xác

Làm thế nào biết được ta có xấp xỉ tốt? Ta đã thấy hai khả năng. Một là dùng (3.6), nghĩa là, $f(x) - P_N(x) = f^{(N)}(\xi_x) w_N(x)/N!$. Vì $w_N(x)$ là một đa thức nên dễ dàng đánh giá tại x bất kỳ. Tuy nhiên, nhân tử đạo hàm là vấn đề vì ta không biết ξ_x và thậm chí không biết cả $f^{(N)}$. Khả năng khác là so sánh kết quả nội suy trên một tập các nút với kết kết quả nội suy có bậc cao hơn nhờ nội suy trên cùng một tập với một nút bổ sung. Một biến thể khác là so sánh với kết quả có cùng bậc nhận được trên một tập nút nội suy khác. Thường cách tiếp cận tốt nhất là giữ lại một số nút và đánh giá sai số chính xác $f(x) - P_N(x)$ tại các nút này. Một đánh giá thực tế có thể đòi hỏi nhiều dữ liệu được giữ lại, và khó mà rõ được dùng nút nào để nội suy và nút nào giữ lại để so sánh. Thông thường ta có một vài ý tưởng về dáng điệu của hàm đang xét. Một đồ thị của dữ liệu và nội suy là sự trợ giúp quan trọng trong quyết định xem phép nội suy có mô phỏng dáng điệu này một cách thích đáng không.

Thí dụ minh họa dưới đây cho thấy dùng các đa thức nội suy bậc cao (nhiều điểm nút), nói chung, không phải là ý tưởng tốt.

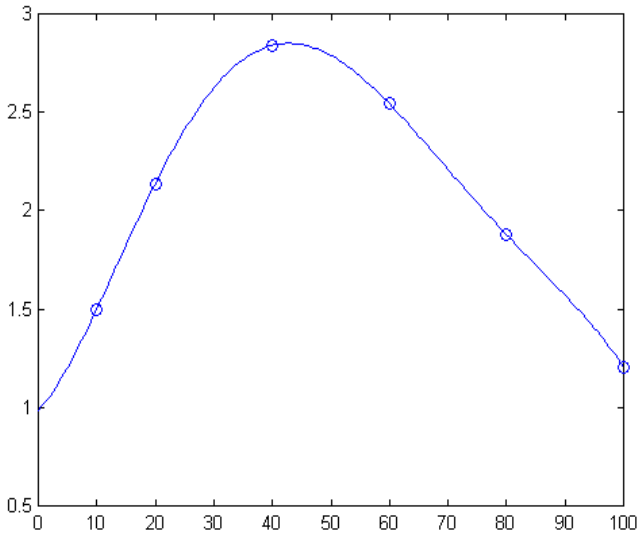
Thí dụ 3.6. Bảng dưới đây cho độ nhớt tương đối V của ethanol như là hàm phần trăm của trọng lượng chất tan w

w	5	10	15	20	30	40
$V(w)$	1.226	1.498	1.882	2.138	2.622	2.840
	50	60	70	80	90	100
	2.807	2.542	2.210	1.877	1.539	1.201

Để xem $P(w)$ tốt hay xấu như thế nào, một vài dữ liệu được giữ lại. Đặc biệt, ta xác định $P_6(w)$ như là đa thức nội suy tại các nút $\{10, 20, 40, 60, 80, 100\}$. Sai số của phép nội suy này được định giá bằng cách tính nó tại các nút còn lại ở đó ta biết giá trị của hàm:

w	5	15	30	50	70	90
$P_6(w)$	1.201	1.824	2.624	2.787	2.210	1.569
$V(w) - P_6(w)$	0.025	−0.002	0.038	0.020	0.000	−0.030

Đây có lẽ là kết quả tốt (hình 3.4).

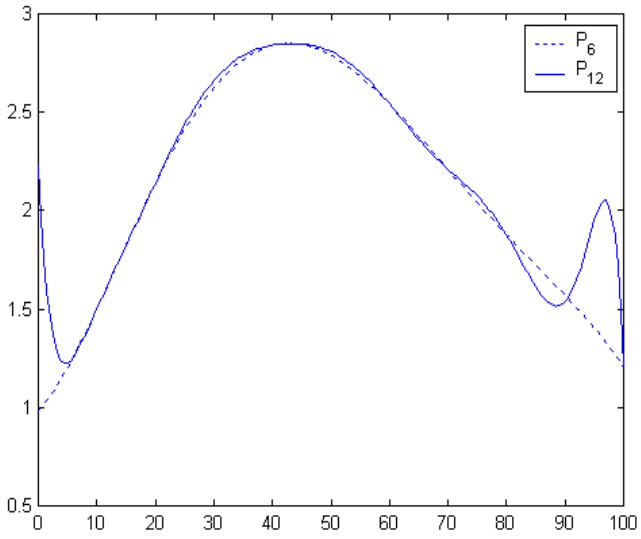


Hình 3.4: Đồ thị hàm $P_6(x)$.

Nếu dùng tất cả 12 điểm nội suy thì kết quả không tốt (hình 3.5).

3.5 Nội suy spline

Công thức biểu diễn sai số của định lý 3.2 đề nghị nâng bậc đa thức nội suy sẽ làm cho xấp xỉ chính xác hơn. Đáng tiếc, các nhân tố khác thường làm cho điều này không thực hiện được. Sai số phụ thuộc mạnh vào độ dài của khoảng chứa các nút. Nếu ta có thể bằng cách nào đó làm giảm độ dài này, thì định lý chỉ ra rằng ta sẽ có một xấp xỉ tốt hơn. Ý tưởng cơ bản của mục này là xấp xỉ $f(x)$ bởi hàm đa thức từng mảnh, nội suy kiểu này gọi là spline. Cụ thể hơn, hàm $f(x)$ được xấp xỉ trên $[x_1, x_N]$. Khoảng $[x_1, x_N]$ được phân hoạch thành các khoảng con $[x_n, x_{n+1}]$, trong đó $x_1 < x_2 < \ldots < x_N$. Một spline là một đa thức trên từng khoảng $[x_n, x_{n+1}]$, các điểm $\{x_i\}$ được gọi là điểm cắt (breakpoint) hay điểm gút (knot). Một vấn đề then chốt là tại các gút hàm xấp xỉ trơn như thế nào, và điều này chi phối bậc của spline.



Hình 3.5: Đồ thị hàm $P_6(x)$ và $P_{12}(x)$.

3.5.1 Spline gián đoạn và spline liên tục

Các spline đơn giản nhất là những hàm xuất phát từ phép nội suy một cách độc lập trên mỗi khoảng con $[x_n, x_{n+1}]$. Chặn (3.9) có thể được dùng cho các khoảng con. Chẳng hạn, giả sử rằng bốn nút bất kỳ được chọn trong mỗi khoảng con $[x_n, x_{n+1}]$. Cho nội suy spline $S(x)$ gồm các đa thức nội suy bậc ba trên các khoảng con. Nếu $h = \max(x_{n+1} - x_n)$ và

$$M_4 = \max_{x_1 \le x \le x_N} |f^{(4)}(x)|,$$

thì

$$|f(x) - S(x)| \le \frac{M_4}{4!} h^4 \quad \text{khi} \quad x_1 \le x \le x_N.$$

Khi $h \to 0$, một xấp xỉ tốt nhận được trên toàn khoảng. Hiển nhiên thủ thuật cố định bậc và xấp xỉ hàm trên từng mảnh có triển vọng hơn cách xấp xỉ hàm trên toàn khoảng nhờ gia tăng bậc đa thức.

Nói chung đa thức trên $[x_n, x_{n+1}]$ không trùng tại x_n với đa thức trên $[x_{n-1}, x_n]$, vì vậy spline này nói chung bất liên tục tại các điểm gút. Khi xấp xỉ một hàm liên tục $f(x)$ điều này không chấp nhận được. Dễ dàng sửa đổi cấu trúc này để nhận được một spline liên tục. Tất cả điều cần làm là bao

gồm các điểm mút của mỗi khoảng con vào các điểm ở đó $f(x)$ được nội suy. Đa thức trên $[x_n, x_{n+1}]$ sẽ có giá trị $f(x_n)$ tại x_n và cũng vậy với đa thức trên $[x_{n-1}, x_n]$.

Chỉ dữ liệu từ $[x_{n-1}, x_n]$ được dùng khi xây dựng spline trên khoảng con này, vì vậy sai số chỉ phụ thuộc vào dáng điệu của $f(x)$ trên khoảng con này. Điều này sẽ không đúng với các spline được đề cập đến sau này. Trong một số hoàn cảnh spline phải được xây dựng trước khi tất cả dữ liệu có hiệu lực và đặc điểm này của phép xây dựng spline là cốt yếu.

Spline liên tục đơn giản nhất là tuyến tính từng mảnh, nghĩa là, đồ thị của hàm $S(x)$ là đường gấp khúc. Nếu $S(x)$ được yêu cầu nội suy $f(x)$ tại các gút, thì trên $[x_n, x_{n+1}]$ với $1 \le n \le N-1$ dạng Lagrange là

$$S(x) = f_n \frac{x - x_{n+1}}{x_n - x_{n+1}} + f_{n+1} \frac{x - x_n}{x_{n+1} - x_n},$$

hay có thể viết lại là

$$S(x) = f_n + \frac{f_{n+1} - f_n}{x_{n+1} - x_n}(x - x_n). \tag{3.31}$$

Thí dụ 3.7. Với dữ liệu (5, 1.226) (30, 2.662) (60, 2.542) (100, 1.201) ta có hàm spline tuyến tính là

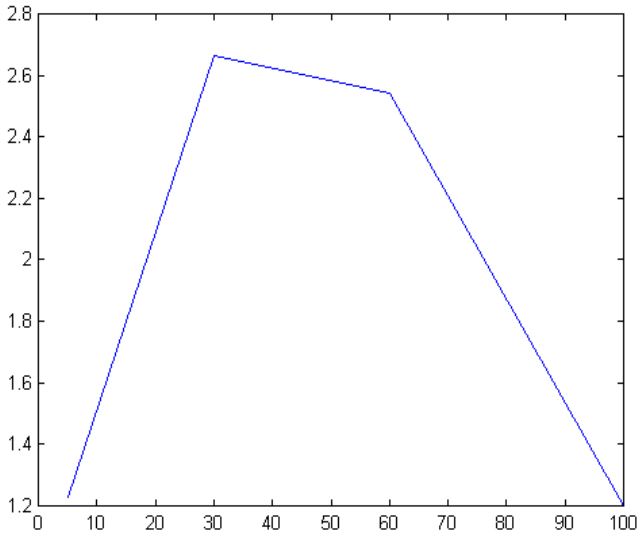
$$S(x) = \begin{cases} 1.266 + 0.05744(x-5), & 5 \le x \le 30 \\ 2.662 - 0.00400(x-30), & 30 \le x \le 60 \\ 2.542 - 0.03352(x-60), & 60 \le x \le 100 \end{cases} \circ$$

Spline nội suy tuyến tính (3.31) rất dễ đánh giá một khi khoảng con cụ thể được xác định. Tất cả các chương trình con đánh giá phải chứa một thuật toán tìm đúng khoảng con. Thường việc này chiếm nhiều thời gian hơn việc đánh giá đa thức. Với nội suy tuyến tính, một chặn sai số là

$$|f(x) - S(x)| \le \frac{1}{8} M_2 h^2 \quad \text{khi} \quad x_1 \le x \le x_N, \tag{3.32}$$

trong đó $M_2 = \max_{x_1 \le x \le x_N} |f''(x)|$. Sự hội tụ được bảo đảm khi $h \to 0$ nếu $|f''|$ bị chặn. Một chứng minh tương tự dùng (3.20) cho

$$|f'(x) - S'(x)| \le M_2 h, \quad x_n \le x \le x_{n+1}, \ 1 \le n < N-1. \tag{3.33}$$



Hình 3.6: Đồ thị hàm spline tuyến tính, thí dụ 3.7.

Vậy, $S'(x)$ có thể dùng để đánh giá $f'(x)$ mà kết quả tốt hơn khi $h \to 0$.

Các spline liên tục được dùng trong giải số bằng phần tử hữu hạn bài toán biên cho các phương trình vi phân cấp hai. Bậc cao hơn cho xấp xỉ chính xác hơn, nhưng bậc cao hơn lại có thể có những dao động không mong đợi. Điều này không thành vấn đề khi dùng spline cho các phần tử hữu hạn, nhưng nó phải được tránh khi dùng spline để biểu diễn dữ liệu. Với mục đích biểu diễn dữ liệu một chọn lựa tốt là dùng đa thức bậc ba.

Sai số của spline bậc ba liên tục xây dựng bởi nội suy cách độc lập trên mỗi khoảng con có thể được phân tích bằng cách dùng các biểu thức sai số thiết lập cho đa thức nội suy. Trên mỗi khoảng con

$$|f^{(k)}(x) - P_4^{(k)}(x)| \leq C_k h^{4-k}$$

với $k = 0, \ldots, 3$ và các hằng số C_k thích hợp. Kết quả tương tự có thể được thiết lập cho tất cả các spline bậc ba ta đề cập đến. Khi $k = 1$ bất đẳng thức trên ám chỉ rằng, với h đủ nhỏ, trên mỗi khoảng con $P_4'(x) \approx f'(x)$, nghĩa là $P_4'(x)$ có cùng dấu với $f'(x)$ miễn là $f'(x) \neq 0$. Nói cách khác, ngoại trừ gần cực trị của $f(x)$, với h nhỏ spline là đơn điệu tăng (giảm) giống như $f(x)$. Cùng một chứng minh áp dụng cho đạo hàm cấp hai, dẫn tới kết luận ngoại trừ gần điểm uốn của $f(x)$, với h nhỏ spline là lồi (lõm) giống như $f(x)$. Ta kết luận với h đủ nhỏ, spline sẽ tái hiện hình dạng của hàm mà nó nội suy. Điều này đúng cho tất cả các spline bậc ba ta đề cập đến. Đây là lý do giải

thích tại sao nội suy spline thỏa mãn nhiều hơn nội suy bằng đa thức bậc cao. Nhưng điều gì xảy ra nếu h không nhỏ? Khi dữ liệu là thưa? Cần phải đặt các điều kiện trên spline để bảo vệ hình dạng của hàm.

3.5.2 Đạo hàm cấp một liên tục

Nếu ta có dữ liệu về đạo hàm, dễ dàng mở rộng cách tiếp cận của tiểu mục trước để nhận được một nội suy với đạo hàm liên tục. Chẳng hạn, ta có thể nội suy $f(x_n)$, $f'(x_n)$, $f(x_{n+1})$, $f'(x_{n+1})$ bằng đa thức nội suy Hermite bậc ba trên $[x_n, x_{n+1}]$. Làm điều này trên mỗi khoảng con ta được một spline $H(x)$ với đạo hàm cấp một liên tục. Mỗi khoảng con được đối xử cách độc lập, vì vậy các chặn (3.21)-(3.24) đúng và chứng tỏ rằng xấp xỉ nhận được là tốt. Liên quan đến phương trình vi phân ta thường phải xấp xỉ hàm $y(x)$ và đạo hàm của nó tại các điểm x_n, $x_n + h/2$, $x_n + h$. Bằng cách lập nội suy Hermite bậc năm cho các dữ liệu này, một spline với đạo hàm liên tục được thiết lập xấp xỉ $y(x)$ và $y'(x)$ với mọi x.

Bây giờ chúng ta hãy biểu diễn dữ liệu khi chỉ có các giá trị $f(x_i)$ là được biết và không có nhiều những giá trị như vậy. Như đã biết spline $H(x)$ có đồ thị đẹp mắt nếu nó có đạo hàm liên tục và nếu nó giữ tính đơn điệu của hàm cho. Vấn đề là phải tránh không cho những dao động xuất hiện trong dữ liệu. Thoạt nghĩ thì các spline tuyến tính bảo vệ tính đơn điệu. Vấn đề là đồ thị của chúng có thể có những "điểm góc". Bằng cách nâng lên bậc ba và giữ đạo hàm cấp một liên tục, ta tránh được các điểm góc. Một nội suy "bảo vệ hình dạng" như vậy có thể xây dựng theo các đường của nội suy Hermite bậc ba. Các đa thức bậc ba trên $[x_{n-1}, x_n]$ và $[x_n, x_{n+1}]$ cả hai nội suy f_n tại x_n. Nếu đạo hàm cấp một phải liên tục thì các đạo hàm cấp một của hai đa thức bậc ba đang xét phải có cùng giá trị tại x_n, nhưng bây giờ giá trị của đạo hàm cấp một là tham số chưa biết mà ta phải chọn để có được tính đơn điệu.

Như trong (3.14) đa thức bậc ba được viết dưới dạng

$$H(x) = a_n + b_n(x - x_n) + c_n(x - x_n)^2 + d_n(x - x_n)^3$$

khi $x_n \le x \le x_{n+1}$, $1 \le n \le N - 1$. Chú ý rằng tham số b_n chính là độ dốc của $H(x)$ tại điểm x_n. Tiến hành như trong thiết lập (3.15)-(3.18) với ký hiệu $h_n = x_{n+1} - x_n$ và $\Delta_n = (f_{n+1} - f_n)/h_n$ ta được

$$\begin{aligned} a_n &= f_n, \\ c_n &= (3\Delta_n - 2b_n - b_{n+1})/h_n, \\ d_n &= (b_n + b_{n+1} - 2\Delta_n)/h_n^2. \end{aligned} \tag{3.34}$$

Các phương trình này là kết quả phép giải ba điều kiện nội suy $H(x_n) = f_n$, $H(x_{n+1}) = f_{n+1}$, và $H'(x_{n+1}) = b_{n+1}$ cho ba ẩn a_n, c_n, và d_n.

Đại lượng Δ_n là độ dốc của đường thẳng đi qua (x_n, f_n) và (x_{n+1}, f_{n+1}). Nếu $\Delta_n = 0$. Có vẻ hợp lý để ép $H(x)$ là hằng trên $[x_n, x_{n+1}]$, nghĩa là, cho các độ dốc $b_n = b_{n+1} = 0$. Nếu $\Delta_n \neq 0$, ta định nghĩa $\alpha_n = b_n/\Delta_n$ và $\beta_n = b_{n+1}/\Delta_n$. Để giữ tính đơn điệu cần thiết là dấu của độ dốc của $H(x)$ tại x_n và x_{n+1} giống như dấu của Δ_n tại các điểm đó. Một cách toán học điều này là $\alpha_n \geq 0$, $\beta_n \geq 0$.

Một điều kiện đủ trên α và β để giữ tính đơn điệu được phát hiện bởi Ferguson và Miller [4]. Điều này cũng được phát hiện độc lập bởi Fritsch và Carson [6]. Chứng minh điều kiện trên bao gồm việc nghiên cứu $H'(x)$ như là một hàm của α_n và β_n. Một điều kiện đơn giản đảm bảo tính đơn điệu được giữ là $\alpha_n, \beta_n \in [0, 3]$. Có nhiều công thức cho α_n và β_n thỏa hạn chế này. Kết quả hay dùng là [5]

$$b_n = \frac{\Delta_{n-1}\Delta_n}{r_n\Delta_n + (1 - r_n)\Delta_{n-1}} \tag{3.35}$$

với

$$r_n = \frac{h_{n-1} + 2h_n}{3(h_{n-1} + h_n)} \tag{3.36}$$

khi $n = 2, 3, \ldots, N - 1$. Nếu $\Delta_{n-1}\Delta_n < 0$, thì các độ dốc đổi dấu tại x_n. Trong trường hợp như vậy có lẽ ta không nên đặt bất kỳ đòi hỏi nào lên độ dốc của $H(x)$ tại x_n. Một số người đề nghị dặt $b_n = 0$ khi điều này xảy ra. Một số khác dùng (3.35) miễn là không có phép chia cho không. Việc chọn lựa mò mẫm có thể làm mất sự bảo toàn hình dạng của spline bậc ba gần những miền ở đó $\Delta_{n-1}\Delta_n < 0$. Tại các điểm cuối quy tắc đơn giản nhất là dùng $b_1 = \Delta_1$ và $b_N = \Delta_{N-1}$. Một chọn lựa tốt hơn là dùng độ dốc cuối của nội suy bậc hai của ba điểm dữ liệu gần nhất (giả sử nó thỏa ràng buộc trên α và β); các khả năng khác được cho trong [5]. Với (3.35) và lựa chọn đơn giản cho b_1 và b_N dễ dàng chứng tỏ rằng các điều kiện đủ trên α_n và β_n được thỏa. Thật vậy, tại các điểm cuối $\alpha_1 = 1$ và $\beta_{N-1} = 1$, mà chắc chắn thuộc $[0, 3]$. Khi $n = 2, 3, \ldots, N - 1$, rõ ràng $\frac{1}{3} \leq r_n \leq \frac{2}{3}$ vì vậy

$$\alpha_n = \frac{\Delta_{n-1}}{[r_n\Delta_n + (1 - r_n)\Delta_{n-1}]} \leq \frac{1}{1 - r_n} \leq 3$$

và

$$\beta_{n-1} = \frac{\Delta_n}{[r_n\Delta_n + (1-r_n)\Delta_{n-1}]} \le \frac{1}{r_n} \le 3$$

như đòi hỏi.

Thuật toán cho $H(x)$ rất đơn giản. Tính b_1 bằng bất kỳ công thức nào; với $n = 2, 3, \ldots, N-1$ lấy $b_n = 0$ nếu $\Delta_{n-1}\Delta_n \le 0$, nếu khác tính b_n từ (3.35), (3.36). Tính b_N. Các giá trị c_n và d_n có thể được tính từ (3.34) khi $n = 1, \ldots, N-1$.

3.5.3 Đạo hàm cấp hai liên tục

Để xây dựng spline bậc ba trơn, ta viết

$$S(x) = a_n + b_n(x - x_n) + c_n(x - x_n)^2 + d_n(x - x_n)^3 \tag{3.37}$$

trên mỗi $[x_n, x_{n+1}]$, $1 \le n \le N-1$. Có $4(N-1)$ tham số tự do phải được xác định. Điều kiện nội suy đòi hỏi rằng khi $1 \le n \le N-1$

$$S(x_n^+) = f_n \quad \text{và} \quad S(x_{n+1}^-) = f_{n+1} \tag{3.38}$$

cho $2(N-1)$ điều kiện. Còn lại $2(N-1)$ bậc tự do mà có thể được dùng để làm $S(x)$ trơn trên toàn bộ $[x_1, x_N]$. Chú ý rằng (3.38) bảo đảm là S liên tục trên $[x_1, x_N]$. Khi S' liên tục tại các gút trong,

$$S'(x_n^-) = S'(x_n^+), \quad 2 \le n \le N-1. \tag{3.39}$$

Điều này cho $N-2$ điều kiện, vì vậy còn lại N bậc tự do. Khi S'' liên tục tại các gút trong,

$$S''(x_n^-) = S''(x_n^+), \quad 2 \le n \le N-1. \tag{3.40}$$

cho $N-2$ điều kiện khác. Chính xác còn lại 2 bậc tự do. Điều này không đủ để có S''' liên tục. Có nhiều khả năng cho hai ràng buộc bổ sung, còn được gọi là *điều kiện cuối*,

Loại (1). $S'(x_1) = f'(x_1)$, $S'(x_N) = f'(x_N)$.

Loại (2). $S''(x_1) = S''(x_N) = 0$.

Loại (3). $S''(x_1) = f'''(x_1)$, $S'''(x_N) = f'''(x_N)$.

Loại (4). $S''(x_1) = S''(x_N)$, $S''(x_1) = S''(x_N)$.

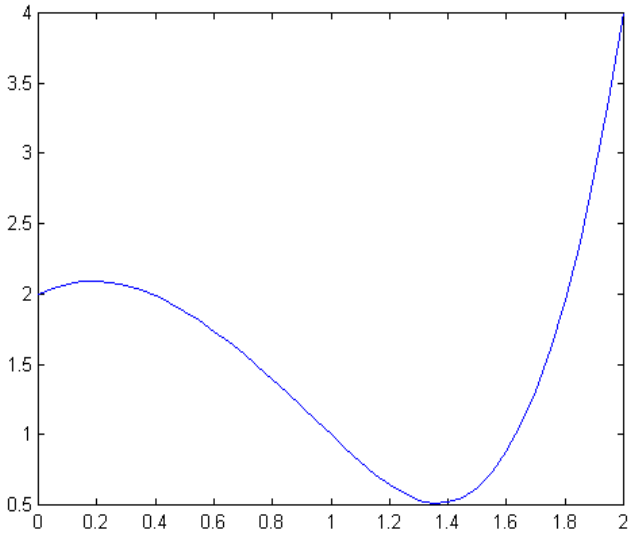
Điều kiện (2) dẫn tới *spline vật lý*. Các spline vật lý làm thẳng nhiều nhất có

thể qua các điểm cuối, vì vậy nó trở thành đường thẳng với đạo hàm cấp hai bằng không. Các điều kiện (1) và (3) hữu dụng chỉ nếu có thêm thông tin về $f(x)$. Tuy nhiên, độ dốc chính xác hay độ cong cần đến ở đây thường được thay thế bằng xấp xỉ nội suy trong thực hành. Điều kiện (4) thích hợp khi $f(x)$ là tuần hoàn với chu kỳ $x_N - x_1$.

Thí dụ 3.8. Cho

$$S(x) = \begin{cases} 2 + x - 3x^2 + x^3, & 0 \le x \le 1 \\ 1 - 2(x-1) + 5(x-1)^3, & 1 \le x \le 2. \end{cases}$$

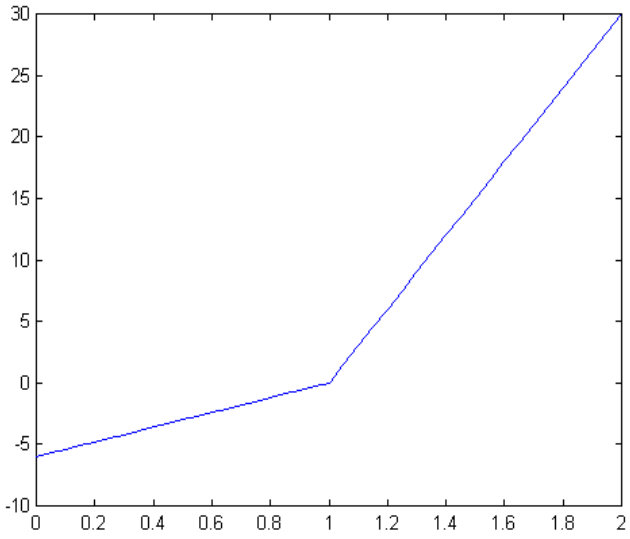
Dễ dàng kiểm tra rằng S thuộc lớp hàm $C^2[0,2]$, và thỏa các điều kiện nội suy $S(0) = 2$, $S(1) = 1$, $S(2) = 4$ và các điều kiện cuối $S'(0) = 1$, $S'(2) = 13$. Đồ thị của S và S'' được cho trên hình 3.7 và 3.8. Chú ý rằng đồ thị của S rất trơn, trong khi đó S'' có điểm góc tại gút $x = 1$. ∘



Hình 3.7: Đồ thị hàm $S(x)$ thí dụ 3.8.

Trở lại sự đặc trưng hóa $S(x)$ ở trên, ta có $4(N-1)$ điều kiện lên $4(N-1)$ ẩn cho bởi (3.37). Phương pháp ma trận có thể dùng ở đây, nhưng trước hết ta làm một số biến đổi. Trên mỗi khoảng $[x_n, x_{n+1}]$

$$S'(x) = b_n + 2c_n(x - x_n) + 3d_n(x - x_n)^2 \tag{3.41}$$
$$S''(x) = 2c_n + 6d_n(x - x_n) \tag{3.42}$$



Hình 3.8: Đồ thị hàm $S''(x)$ thí dụ 3.8.

Điều kiện nội suy cho, từ (3.37),

$$a_n = f_n, \quad 1 \le n \le N-1, \tag{3.43}$$

và cũng vậy $f_{n+1} = a_n + b_n h_n + c_n h_n^2 + d_n h_n^3$ mà có thể viết như sau

$$b_n = (f_{n+1} - f_n)/h_n - c_n h_n - d_n h_n^2, \quad 1 \le n \le N-1. \tag{3.44}$$

Điều này khử đi một nửa số ẩn. Điều kiện liên tục (3.40) trên $S''(x)$ nói rằng $2c_n = 2c_{n-1} + 6d_{n-1}h_{n-1}$. Áp đặt thêm $c_N = S''(x_N)/2$ (được gợi ý từ (3.42), ta được:

$$d_n = \frac{c_{n+1} - c_n}{3h_n}, \quad 1 \le n \le N-1. \tag{3.45}$$

Chỉ còn lại các công thức cho các $c_1, \ldots, c_N$. Chúng được cho bởi hai điều kiện cuối và tính liên tục của $S'(x)$. Từ (3.39) và (3.41) ta có ngay $b_n = b_{n-1} + 2c_{n-1}h_{n-1} + 3d_{n-1}h_{n-1}^2$ khi $2 \le n \le N-1$. Thay vào (3.44) và (3.45) cho:

$$\frac{f_{n+1} - f_n}{h_n} - c_n h_n - \frac{1}{3}h_n(c_{n+1} - c_n) \;=\; \frac{f_n - f_{n-1}}{h_{n-1}} + c_{n-1}h_{n-1} + \frac{2}{3}h_{n-1}(c_n - c_{n-1}),$$

và sắp xếp lại ta được

$$h_{n-1}c_{n-1} + 2(h_{n-1} + h_n)c_n + h_n c_{n+1} = 3\left(\frac{f_{n+1} - f_n}{h_n} - \frac{f_n - f_{n-1}}{h_{n-1}}\right) \quad (3.46)$$

với $2 \le n \le N-1$.

Chỉ loại thứ nhất của các điều kiện cuối (độ dốc cho trước) được đề cập đến ở đây. Từ (3.37), (3.44), và (3.45),

$$f'(x_1) = S'(x_1) = b_1 = \frac{f_2 - f_1}{h_1} - c_1 h_1 - d_1 h_1^2 = \frac{f_2 - f_1}{h_1} - c_1 h_1 - \frac{1}{3}h_1(c_2 - c_1),$$

vậy

$$2h_1 c_1 + h_1 c_2 = 3\left(\frac{f_2 - f_1}{h_1} - f'(x_1)\right). \quad (3.47)$$

Tương tự, $f'(x_N) = S'(x_N)$ dẫn đến

$$h_{N-1}c_{N-1} + 2h_{N-1}c_N = 3\left(f'(x_N) - \frac{f_N - f_{N-1}}{h_{N-1}}\right). \quad (3.48)$$

Các phương trình (3.45)-(3.48) cho N phương trình theo N ẩn $c_1, \ldots, c_N$. Ma trận các hệ số có cấu trúc rất đặc biệt

$$\begin{bmatrix} 2h_1 & h_1 & & & & & \\ h_1 & 2(h_1+h_2) & h_2 & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & & h_{N-2} & 2(h_{N-2}+h_{N-1}) & h_{N-1} & \\ & & & & h_{N-1} & 2h_{N-1} & \end{bmatrix}$$

Ma trận như vậy được gọi là ma trận ba đường chéo. Hệ phương trình với ma trận các hệ số có dạng ba đường chéo có nghiệm duy nhất với mọi vế phải và nghiệm có thể được tìm chính xác bằng phép khử Gauss mà không cần hoán vị dòng.

Vế hai:

$$\begin{bmatrix} 3\left(\frac{f_2-f_1}{h_1} - f'(x_1)\right) \\ 3\left(\frac{f_3-f_2}{h_2} - \frac{f_2-f_1}{h_1}\right) \\ \vdots \\ 3\left(\frac{f_N-f_{N-1}}{h_{N-1}} - \frac{f_{N-1}-f_{N-2}}{h_{N-2}}\right) \\ 3\left(f'(x_N) - \frac{f_N-f_{N-1}}{h_{N-1}}\right) \end{bmatrix}.$$

Với các $c_1, c_2, \ldots, c_N$ tìm được, tính các hệ số b_k, d_k ($k = 1, \ldots, N-1$) theo các công thức tương ứng.

Tóm lại, ta đã chứng minh được định lý sau.

Định lý 3.4. *Cho trước các gút $x_1 < x_2 < \ldots < x_N$ và $f_n = f(x_n)$, $1 \le n \le N$, tồn tại một và chỉ một hàm $S(x)$ thỏa các điều kiện sau*

1. *$S(x)$ là đa thức bậc ba trong mỗi $[x_n, x_{n+1}]$, $1 \le n \le N-1$.*
2. *$S(x)$ thuộc lớp $C^2[x_1, x_N]$.*
3. *$S(x_n) = f_n$, $1 \le n \le N$.*
4. *$S'(x_1) = f'(x_1)$, $S'(x_N) = f'(x_N)$.*

Với sự chọn lựa này của điều kiện cuối, $S(x)$ được gọi là *spline bậc ba đầy đủ* (complete cubic spline). Ma trận hệ số có cùng cấu trúc cho các điều kiện loại (2) và (3) và các kết quả tương tự là đúng với chúng. Với sự chọn lựa (2), $S(x)$ được gọi là *spline bậc ba tự nhiên* (natural cubic spline). Với điều kiện cuối loại (4) ma trận hệ số có dạng khác nhưng kết quả tương tự là đúng và spline có thể được tính một cách tiện lợi.

Nhận xét 3.4. Trong thiết lập spline bậc ba đầy đủ ở trên ta đã dùng

$$c_N = S''(x_N)/2.$$

Thật ra, có thể xây dựng spline bậc ba như ở đây mà không cần áp đặt này. Khi đó, công thức (3.45) chỉ có hiệu lực với $1 \le n \le N-2$. Như vậy, sau đó ta phải tìm các công thức cho d_{N-1} và các $c_1, \ldots, c_{N-1}$. Việc thiết lập theo cách này xem như bài tập.

Function `spline_3` dưới đây được viết dựa trên các phân tích ở trên (spline bậc ba đầy đủ)[1]. Mã của function này gọi `trisolve`, đây là function giải hệ phương trình đại số tuyến tính 3 đường chéo.

[1]Trong Matlab, function `spline` cũng được viết với cùng thuật toán nêu ra ở đây.

spline_3.m

```
function s=spline_3(t,y,d1,dn)
% SPLINE_3 tra ve mang cac he so cua da thuc bac 3 tren cac khoang con
% cu phap: s = spline_3(t,y)
% input:
%       t: vector chua cac nut noi suy
%       y: vector chua cac gia tri ham noi suy
%       d1: gia tri dao ham tai diem dau
%       dn: gia tri dao ham tai diem cuoi
% output:
%       s: mang chua cac he so cua da thuc bac 3 tren cac khoang con
%          theo thu tu luy thua lui
N=length(t);
s=zeros(N-1,4);
f=zeros(N,1);
k=1:N-1;
h=t(k+1)-t(k);
dy=(y(k+1)-y(k))./h(k);
% an=fn (cot 4)
s(:,4)=y(1:N-1);
% ma tran cac he so va vecto xac dinh cac cn (cot 2)
% lline, dline, uline la ba duong cheo
dline(1)=2*h(1);
uline(1)=h(1);
f(1)=3*(dy(1)-d1);
for i=2:N-1
    lline(i-1)=h(i-1);
    dline(i)=2*(h(i-1)+h(i));
    uline(i)=h(i);
    f(i)=3*(dy(i)-dy(i-1));
end
lline(N)=h(N-1);
dline(N)=2*h(N-1);
f(N)=3*(dn-dy(N-1));
c=trisolve(lline,dline,uline,f);
s(:,2)=transpose(c(1:N-1));
% xac dinh dn (cot 1)
for i=1:N-1
    s(i,1)=(c(i+1)-c(i))/h(i)/3;
end
% xac dinh bn (cot 3)
```

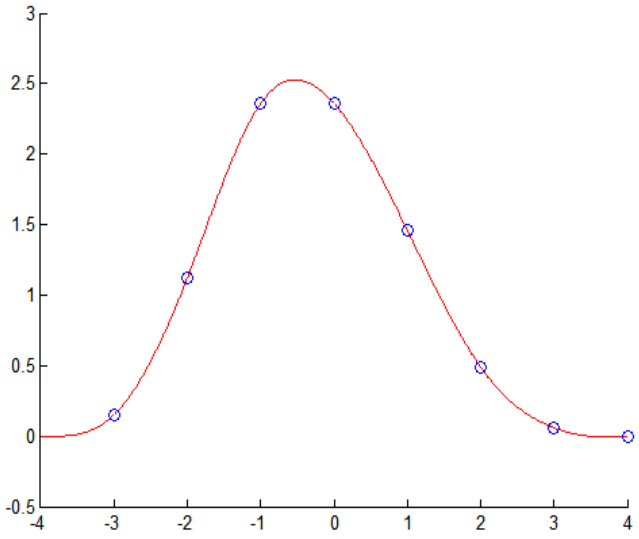
```
for i=1:N-1
    s(i,3)=dy(i)-s(i,2)*h(i)-s(i,1)*h(i)^2;
end
```

trisolve.m

```
function [b]= trisolve(lline,dline,uline,b)
% TRISOLVE giai he ba duong cheo
% cu phap = trisolve(lline,dline,uline,b)
% input:
%            lline - duong cheo duoi
%            dline - duong cheo chinh
%            uline - duong cheo tren
%            b - ve phai
% output:    b - nghiem
N=length(dline);
% khu
for i=1:N-1
    lline(i)=lline(i)/dline(i);
    dline(i+1)=dline(i+1)-lline(i)*uline(i);
end
% giai Ly = b bang phep the tien
for i=2:N
    b(i)=b(i)-lline(i-1)*b(i-1);
end
% giai Ux = y bang phep the lui
b(N)=b(N)/dline(N);
for i=N-1:-1:1
    b(i)=(b(i)-uline(i)*b(i+1))/dline(i);
end
```

Chương trình viết bằng Matlab dưới đây gọi function `spline_3` tính spline bậc ba đầy đủ với số liệu cho tại các nút: `x, y`.

```
clear all
x = -4:4;
y = [0 .15 1.12 2.36 2.36 1.46 .49 .06 0];
cs=spline_3(x,y,0,0);
hold on
for i=1:length(x)-1
    xx=linspace(x(i),x(i+1));
    yy=polyval(cs(i,:),xx-x(i));
    plot(xx,yy,'r-');
end
```



Hình 3.9: Xấp xỉ bằng spline bậc ba đầy đủ.

```
plot(x,y,'o');
```

Chú giải

⋆ **linspace** - phát sinh vectơ các điểm cách đều. `linspace(x1, x2)` sinh ra vectơ dòng chứa 100 điểm cách đều giữa `x1` và `x2`. `linspace(x1, x2, N)` sinh ra `N` điểm. Nếu $N < 2$, `linspace` trả về `x2`.

⋆ **polyval** - đánh giá đa thức.

`y = polyval(p,x)` trả về giá trị của đa thức `p` được đánh giá tại `x`. `p` là vectơ dòng mà các phần tử là các hệ số của đa thức theo thứ tự lũy thừa lùi. Nếu `x` là là ma trận hay vectơ, đa thức được đánh giá tại tất cả các điểm trong `x`.

⋆ **plot** - vẽ đường nối. Nếu `x,y` là vectơ có độ dài `N+1`, `plot(x,y)` sẽ vẽ đường "gấp khúc" gồm các đoạn thẳng nối `(x(i),y(i)` với `(x(i+1),y(i+1)`, `i=1,...,N`. `plot` có các tùy chọn liên quan đến loại đường nối, màu sắc, ký hiệu. Thí dụ, `plot(x,y,'r+:'` vẽ đường chấm chấm màu đỏ với dấu + tại mỗi điểm dữ liệu (xem thêm trợ giúp của Matlab, dùng lệnh `help plot`.).

⋆ **hold** - giữ đồ thị hiện hành. `hold on` giữ đồ thị hiện hành và tất cả các tính chất của trục tọa độ. `hold off` trả về mốt mặc định mỗi khi lệnh `plot` xóa các đồ thị trước đó và đặt lại tất cả các tính chất của trục trước khi

vẽ đồ thị mới.

Câu hỏi và bài tập

3.1. Bậc của đa thức nội suy Lagrange luôn luôn bằng $N-1$? Nếu không, hãy minh họa bằng một thí dụ.

3.2. Giả sử $f(x)$ là đa thức bậc nhỏ hơn hay bằng $N-1$. Chứng minh rằng nếu $P_N(x)$ nội suy $f(x)$ tại N điểm phân biệt, thì $P_N(x) \equiv f(x)$. Cho một thí dụ ($N \geq 3$) và tính toán trực tiếp để kiểm tra.

3.3. Cho bảng dữ liệu

x	1	2
$f(x)$	2	4

Xây dựng đa thức nội suy Lagrange $P_2(x)$. Tìm một đa thức $Q(x)$ bậc hai cũng nội suy các dữ liệu này. Điều này có mâu thuẩn với tính duy nhất của đa thức nội suy không? Giải thích.

3.4. Một phương pháp khác để tính $P_N(x)$ là viết

$$P_N(x) = c_1 + c_2 x + \ldots + c_N x^{N-1}.$$

Các điều kiện nội suy, $P_N(x_j) = f_j$ với $1 \leq j \leq N$, cho hệ gồm N phương trình đại số tuyến tính theo N ẩn $c_1, \ldots, c_N$. Chú ý, ma trận các hệ số có thể rất "xấu". Viết thuật toán và chương trình tính.

3.5. Có hai cách tính giá trị của $P_N(x) = c_1 + c_2 x + \ldots + c_N x^{N-1}$.

(a) Thuật toán 1

P:=c_1
`for` i=2:N
  `begin`
    P:=P+$c_i * x^{i-1}$
  `end` i

Có bao nhiêu phép nhân được thực hiện trong thuật toán này?

(b) Thuật toán 2 (dùng dạng xếp lồng vào nhau của $P_N(x)$)

P:=c_N
`for` i=N-1:-1:1

```
begin
        P:=P*x + c_i
end i
```

So sánh số phép nhân của hai thuật toán.

3.6. Đạo hàm của $f(x)$ có thể được đánh giá nhờ đạo hàm tương ứng của $P_N(x)$ với cách chọn của N và $\{x_n\}_{n=1}^N$. Cách tiếp cận thông thường là

$$f^{(N-1)}(x) \approx P_N^{(N-1)}(x).$$

Vì $P_N(x)$ có bậc $N-1$ nên $P_N^{(N-1)}(x)$ phải là hàm hằng.

(a) Chứng minh

$$P_N^{(N-1)}(x) = (N-1)! \sum_{k=1}^{N} \frac{f_k}{\prod_{j \neq k}(x_k - x_j)}.$$

(b) Khi $N = 2$ xấp xỉ của $f'(x)$ là gì?

3.7. Thiết lập các phương trình (3.15)-(3.18).

3.8. Kiểm tra bằng cách dùng nội suy đa thức tại $N = 2m+1$ điểm cách đều $x_j = -5 + 5(j-1)/m$ cho xấp xỉ xấu hàm Runge trên $[-5, 5]$.

a) Tính giá trị cực đại của $|f(x) - P_{2m+1}(x)|$ trên một tập hợp nhiều giá trị x (không là điểm nội suy) trong $[-5, 5]$ với $m = 7$, $m = 10$, và $m = 13$. Sai số tăng hay giảm khi m lớn hơn.

b) Lập lại câu a) nhưng lần này chỉ tính trên $[-1, 1]$. Dùng cùng $\{x_j\}$ và cùng ba giá trị m như câu a). Lần này thì điều gì xảy ra khi N tăng?

3.9. Kiểm tra bằng các dùng nội suy đa thức tại các điểm Chebyshev (3.11) cho xấp xỉ tốt hàm Runge. Như trong bài tập trên, tính giá trị cực đại của $|f(x) - P_N(x)|$ trên một tập hợp nhiều giá trị x (không là điểm nội suy) trong $[-5, 5]$ với $N = 15$, $N = 21$, và $N = 27$. Dáng điệu của sai số khi N lớn hơn?

3.10. Cho bảng dữ liệu

x	-1	0	1	2
f	2	2	2	5

tính $P_4(x)$

a) ở dạng Lagrange (3.2),

b) dùng phương pháp ma trận trong bài tập 3.4,

c) ở dạng tỉ sai phân (3.27).

3.11. Cho bảng dữ liệu

x	-2	-1	0	1	2
f	-4	1	1	2	10

tính $P_5(x)$ với các trường hợp như bài tập 3.10.

3.12. Tính bảng tỉ sai phân và $P_3(x)$ cho dữ liệu của thí dụ 3.1. Kiểm lại rằng đa thức này giống đa thức ở dạng Lagrange.

3.13. Viết chương trình tính đa thức nội suy ở dạng tỉ sai phân Newton.

3.14. Số phép tính để đánh giá các hệ số trong dạng tỉ sai phân của đa thức nội suy là bao nhiêu? Số phép tính để đánh giá giá trị $P_N(x)$ là bao nhiêu? So sánh với dạng Lagrange.

Chương 4

Nghiệm phương trình phi tuyến

Tìm nghiệm của hệ phương trình phi tuyến

$$f(x) = 0 \tag{4.1}$$

là công việc thường gặp trong tính toán. Hầu hết chương này được dành cho trường hợp $f(x)$ là hàm thực liên tục theo một biến thực x vì nó quan trọng và có thể được bàn luận một cách sơ cấp. Trường hợp tổng quát n phương trình n ẩn số thì khó hơn cả về lý thuyết lẫn thực hành. Tuy vậy lý thuyết cũng được trình bày ở đây, một vài phương pháp đơn giản được bàn luận cách vắn tắt ở cuối chương.

4.1 Nhập môn

Nghiệm của (4.1), hay không điểm của $f(x)$, là số α sao cho $f(\alpha) = 0$. Một nghiệm được mô tả đầy đủ hơn bởi bội m của nó. Điều này có nghĩa là với x đủ gần α, $f(x)$ có thể được biểu diễn dưới dạng

$$f(x) = (x - \alpha)^m g(x) \tag{4.2}$$

trong đó $g(x)$ là hàm thực liên tục gần α và $g(\alpha) \neq 0$. Nếu $m = 1$, nghiệm được gọi là đơn (simple), nếu khác được gọi là bội (multiple). Định nghĩa cơ bản thừa nhận m là hữu tỉ. Chẳng hạn, với hàm

$$f(x) = x\sqrt{x-1},$$

phương trình (4.1) có $\alpha = 1$ là nghiệm bội $1/2$ (và $\alpha = 0$ là nghiệm đơn). Tuy nhiên, nếu $f(x)$ đủ trơn, thì m phải là nguyên dương. Thật vậy, nếu $f(x)$ có

đạo hàm đến cấp m liên tục trên một khoảng chứa α và

$$\begin{cases} f(\alpha) = 0, \\ f'(\alpha) = f''(\alpha) = \ldots = f^{(m-1)}(\alpha) = 0, \\ f^{(m)}(\alpha) \neq 0, \end{cases} \tag{4.3}$$

thì α là một nghiệm bội m. Điều này được thấy bằng cách khai triển $f(x)$ thành chuỗi Taylor trong lân cận α

$$\begin{aligned} f(x) &= f(\alpha) + (x-\alpha) f'(\alpha) + \frac{(x-\alpha)^2}{2} f''(\alpha) + \ldots \\ &\quad + \frac{(x-\alpha)^{m-1}}{(m-1)!} f^{(m-1)}(\alpha) + \frac{(x-\alpha)^m}{m!} f^{(m)}(\xi_x), \end{aligned}$$

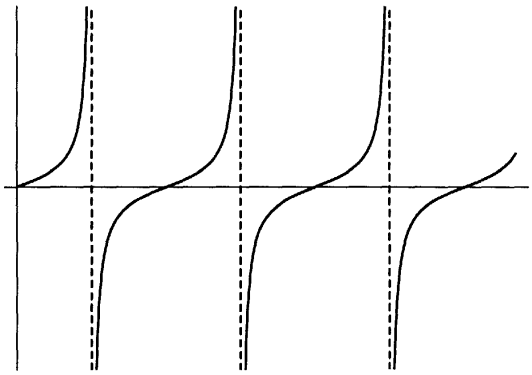
trong đó ξ_x nằm giữa x và α. Dùng (4.3), phương trình trên thành

$$f(x) = \frac{(x-\alpha)^m}{m!} f^{(m)}(\xi_x). \tag{4.4}$$

Nếu ta lấy $g(x) = f^{(m)}(\xi_x)/m!$, thì $g(\alpha) = f^{(m)}(\alpha)/m! \neq 0$. Ta sẽ luôn giả sử rằng $f(x)$ đủ trơn trong lân cận α để có thể dùng (4.4) thay vì dùng định nghĩa cơ bản (4.2) và đặc biệt, nghiệm là bội nguyên.

Theo định nghĩa của nghiệm α, đồ thị của $f(x)$ tiếp xúc với trục x tại α (hình 4.1). Với nghiệm bội m, hàm $f^{(m)}(x)$ không đổi dấu trong lân cận của α vì nó liên tục và $f^{(m)}(\alpha) \neq 0$. Nhận xét này và hệ thức (4.4) chứng tỏ rằng nếu m chẵn, $f(x)$ tiếp xúc với trục x tại α nhưng không đi qua điểm đó và nếu m lẻ, $f(x)$ cắt trục x tại α.

Khảo sát hàm số và vẽ phác đồ thị của nó là cách thường dùng để định vị các nghiệm và xác định bội của chúng. Xét một trường hợp đặc biệt của họ các bài toán dạng $0 = f(x) = F(x) - \gamma$ với tham số $\gamma > 0$, cho $F(x) = x\exp(-x)$ và một giá trị đại diện của γ là 0.07. Với họ này, khi $x \to -\infty$, $f(x) \to -\infty$ và khi $x \to +\infty$, $f(x) \to -\gamma$. Từ đạo hàm cấp một, $f'(x) = (1-x)\exp(-x)$, có thể thấy rằng f tăng ngặt khi $x < 1$ và giảm ngặt khi $x > 1$. Tại cực trị $f(1) = e^{-1} - \gamma$ là dương với $\gamma = 0.07$. Cũng vậy, $f(0) = -\gamma$ là âm. Các sự kiện này và tính liên tục của hàm f báo cho ta biết khi $\gamma = 0,07$, có đúng hai nghiệm đơn, một nằm trong $(0, 1)$ và một thì



Hình 4.1:

lớn hơn 1. Nói chung, với nghiệm của $f(x)$ là bội, $f'(x)$ phải triệt tiêu tại nghiệm. Vì vậy, bất cứ đâu mà hàm số tăng ngặt hay giảm ngặt, thì nghiệm nếu có phải là nghiệm đơn. Với họ các hàm số, sự kiện $f'(x) = 0$ chỉ tại $x = 1$ có nghĩa là đây là vị trí duy nhất ở đó hàm số có thể có nghiệm bội. Dễ thấy rằng phương trình có một nghiệm bội chỉ khi $\gamma = e^{-1}$ và nghiệm là bội 2 (nghiệm kép).

Một nghiệm xấp xỉ z làm cho giá trị tính toán $f(z) = 0$ thì không hiếm, đặc biệt khi nó xấp xỉ một nghiệm bội α. Sau hết, mục đích là phải tìm z làm cho $f(z)$ triệt tiêu. Khi nghiệm là bội m, $f(z) \approx (z-\alpha)^m g(\alpha)$. Vài con số giúp chúng ta hiểu điều này. Với một nghiệm số bội cao như $m = 10$, một xấp xỉ chính xác nhất giống như $z = \alpha + 10^{-4}$ dẫn đến $f(z) = 10^{-40} g(\alpha)$. Thì nếu $|g(\alpha)| \leq 1$, hàm $f(z)$ underflow trong số học chính xác đơn IEEE.

Như ta sẽ thấy, các phương pháp chuẩn không có hiệu quả đối với nghiệm bội như chúng là đối với nghiệm đơn. Để hiểu sự thực hiện của chương trình (mã) đặt cơ sở trên các phương pháp này, cần thiết hiểu rõ rằng các nghiệm gần nhau có vẻ giống như nghiệm bội. Giả sử $f(x)$ có hai nghiệm đơn $\alpha_1 \neq \alpha_2$. Định nghĩa cơ bản và một chút chứng minh chứng tỏ rằng $f(x) = (x-\alpha_1)(x-\alpha_2)G(x)$ với $G(x)$ không triệt tiêu tại cả hai nghiệm. Biểu thức này có thể được viết

$$f(x) = (x-\alpha_1)[(x-\alpha_1) + (\alpha_1 - \alpha_2)]G(x).$$

Khi x xa các nghiệm theo nghĩa $|x-\alpha_1| \gg |\alpha_2 - \alpha_1|$, cặp nghiệm đơn "có vẻ" giống như cặp nghiệm kép vì

$$f(x) \approx (x-\alpha_1)^2 G(x).$$

Một khái niệm từ lý thuyết biến phức liên hệ với nghiệm bội m là cực

điểm (pole) bội m. Nếu ta có thể viết

$$F(x) = (x-\alpha)^{-m} G(x),$$

trong đó $G(\alpha) \neq 0$, thì ta nói rằng α là một cực điểm của $F(x)$ bội m. Dễ thấy nếu α là nghiệm của $f(x)$ bội m, thì nó là một cực điểm của $F(x) = 1/f(x)$ cùng số bội, và ngược lại. Một thí dụ quen thuộc là $\tan(x) = \sin(x)/\cos(x)$, vẽ trên hình 4.1. Hàm này có nghiệm nơi $\sin(x)$ triệt tiêu và cực điểm nơi $\cos(x)$ triệt tiêu. Các hàm đổi dấu tại cực điểm có bội lẻ.

Một khó khăn trong tính toán nghiệm của $f(x) = 0$ là quyết định khi nào một xấp xỉ z là đủ tốt. Thặng dư $f(z)$ có vẻ là cách hiển nhiên để khẳng định chất lượng của một nghiệm xấp xỉ. MATHCAD làm điều này một cách chính xác. Nó chấp nhận z như một nghiệm khi $|f(z)| < \text{TOL}$, với $\text{TOL} = 10^{-3}$ là giá trị mặc nhiên. Điều phiền nhiễu với kiểm tra thặng dư là không có thang đo hiển nhiên. Các nghiệm bội thể hiện khó khăn vì hàm gần như phẳng trong một khoảng lân cận của nghiệm. Vấn đề không chỉ liên hệ tới điều kiện của nghiệm, nhưng còn liên hệ tới cách chúng ta đặt phương trình.

Khi thiết lập bài toán, ta chọn một tỉ lệ. Điều này có thể đơn thuần chỉ là chọn hệ đơn vị, nhưng thường ta dùng sự kiện rằng không điểm bất kỳ của $f(x)$ là không điểm của $f(x)g(x)$. Việc đưa vào một tỉ lệ $g(x)$ có thể tạo một khác biệt hoàn toàn. Chẳng hạn, hai bài toán $\sin(x) = 0$ và $F(x) = 10^{-38}\sin(x) = 0$ là tương đương về mặt toán học, nhưng phương trình thứ hai được lấy tỉ lệ xấu vì sự hình thành $F(z)$ mà với ngay cả một nghiệm xấp xỉ tốt vừa phải z sẽ gây ra underflow trong số học chính xác đơn IEEE. Thường ta lấy tỉ lệ các bài toán mà không suy nghĩ chút nào về vấn đề này, nhưng một tỉ lệ tốt có thể hoàn toàn hữu ích. Đó là lời khuyên rất hữu ích khi đối xử với các kỳ dị thực hay biểu kiến. Hàm $f(x) = \sin(x)/x$ có dáng điệu hoàn toàn tốt tại $x = 0$ (nó là giải tích), nhưng nó có kỳ dị biểu kiến ở đó và cần một chút cẩn thận khi đánh giá nó. Điều này có thể được phá vỡ bằng cách tính nghiệm của hàm được lấy tỉ lệ $F(x) = xf(x)$. Cần giữ trong trí rằng như với thí dụ này, $F(x)$ có tất cả các nghiệm của $f(x)$, nhưng nó có thể lấy thêm nghiệm ngoại lai từ $g(x)$. Một thí dụ đáng kể hơn được cung cấp bởi phương trình

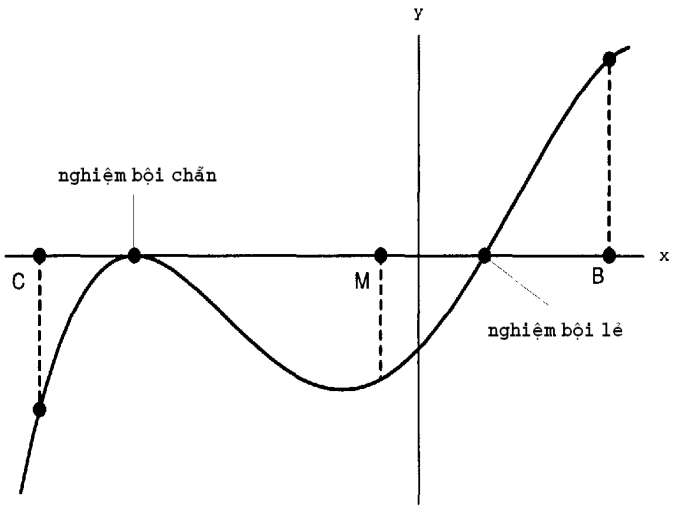
$$f(x) = \frac{1}{180} - \left(\frac{1-\cos(\pi/10)}{\cos(\pi/10)-\cos(x)}\right)\frac{\sin(x)}{x}$$

Hàm này có một cực điểm đơn tại tất cả những điểm mà ở đó $\cos(x) = \cos(\pi/10)$ và một kỳ dị biểu kiến tại $x = 0$. Lấy tỉ lệ hàm này với $g(x) = x(\cos(\pi/10) - \cos(x))$ làm cho việc tính nghiệm trực tiếp hơn.

Thỉnh thoảng một độ đo tỉ lệ tự nhiên được cung cấp bởi hệ số trong phương trình. Một thí dụ cho điều này là họ các bài toán $f(x) = F(x) - \gamma$, với $\gamma > 0$. Giống như khi giải phương trình tuyến tính, thặng dư $r = f(z) = F(z) - \gamma$ có thể được dùng trong phân tích sai số lùi. Hiển nhiên z là nghiệm chính xác của bài toán $0 = F(x) - \gamma'$, trong đó $\gamma' = \gamma + r$. Nếu $|r|$ là nhỏ so với $|\gamma|$, thì z là nghiệm chính xác của một bài toán gần với bài toán cho. Với những bài toán như vậy chúng ta có một cách hợp lý để chỉ định thặng dư phải nhỏ như thế nào.

4.2 Phương pháp chia đôi

Nếu hàm liên tục $f(x)$ có dấu đối nhau tại các điểm $x = B$ và $x = C$, thì nó có ít nhất một không điểm trong khoảng giữa B và C. Phương pháp chia đôi (hay tìm kiếm nhị phân) dựa trên sự kiện này. Nếu $f(B)f(C) < 0$, hàm $f(x)$ được đánh giá tại điểm giữa $M = (B + C)/2$ của khoảng. Nếu $f(M) = 0$, một không điểm được tìm thấy. Nếu khác, $f(B)f(M) < 0$ hoặc



Hình 4.2:

$f(M)f(C) < 0$. Trong trường hợp đầu có ít nhất một không điểm ở giữa M và B, như trong hình 4.2, và trong trường hợp thứ hai có ít nhất một nghiệm ở giữa C và M. Trường hợp này một khoảng chứa nghiệm được tìm thấy có chiều dài bằng nửa chiều dài khoảng ban đầu. Thủ tục được lặp lại cho đến khi định vị được nghiệm với độ chính xác mong muốn.

Thuật toán chia đôi

```
until abs(B - C)  đủ nhỏ hay  f(M) = 0
      M = (B + C)/2
      if  f(B)*f(M) < 0  then
          C=M
      else
          B:=M
end until
```

Thí dụ 4.1. Khi $f(x) = x^2 - 2$, phương trình (4.1) có nghiệm đơn $\alpha = \sqrt{2}$. Với $B = 0$, $C = 6$, thủ tục chia đôi

B	C	$\lvert\alpha - M\rvert$
0.0	6.0	0.16×10^{1}
0.0	3.0	0.86×10^{-1}
0.0	1.5	0.66
0.75	1.5	0.29
1.125	1.5	0.10
1.3125	1.5	0.80×10^{-2}
1.40625	1.5	0.39×10^{-1}
1.40625	1.453125	0.15×10^{-1}

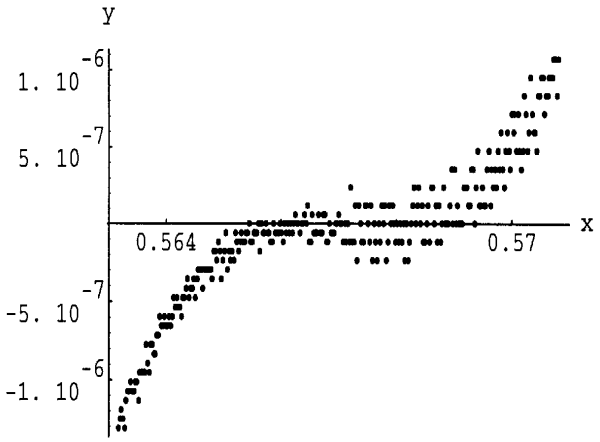
Chú ý dáng điệu thất thường của sai số, mặc dù chiều dài $|B - C|$ giảm một nửa ở mỗi bước ∘

Một nghiên cứu sâu hơn về phương pháp chia đôi cho thấy một số điểm quan trọng để hiểu các phương pháp tìm không điểm, những điểm mà ta cần biết khi phát triển một thuật toán nhằm có được cái tốt nhất từ nhiều phương pháp.

Một khoảng $[B, C]$ với $f(B)f(C) < 0$ được gọi là khoảng giữa hai điểm trên dưới (bracket). Đồ thị của hàm số $f(x)$ cho ta nhiều thông tin hơn là nhận xét "$f(x)$ có một nghiệm trong khoảng". Các không điểm bội chẵn giữa B và C không gây ra sự đổi dấu còn các không điểm bội lẻ thì cho. Nếu có một số chẵn các không điểm bội lẻ giữa B và C, sự đổi dấu sẽ bị loại và f sẽ có cùng dấu tại các điểm cuối. Vậy nếu, $f(B)f(C) < 0$, phải có một số lẻ các không điểm bội lẻ và có thể có vài không điểm bội chẵn giữa B và C. Nếu ta đồng ý đếm số không điểm theo bội của chúng (i.e., một không điểm bội m được đếm như là m không điểm), thì ta thấy có một số lẻ các không điểm giữa B và C.

Một cài đặt cẩn thận thuật toán chia đôi đưa vào một số vấn đề đã được đề cập đến trong chương 1. Trong thuật toán trên, có: (1) kiểm tra cho các giá trị chính xác bằng không; (2) kiểm tra cho sự đổi dấu không được lập trình như kiểm tra $f(B)f(C) < 0$ vì khả năng underflow của tích; và (3) điểm giữa nên được tính như là $M = B + (B - C)/2$ vì nó dễ để tính và chính xác hơn $M = (B + C)/2$.

Chúng ta thường cố gắng tìm một nghiệm xấp xỉ z để cho $f(z)$ là nhỏ đến mức có thể. Trong nỗ lực này, độ dài từ (word) hữu hạn phải được tính đến và vì vậy phải chi tiết thủ tục đánh giá f. Cuối cùng ngay cả dấu của giá trị tính toán có thể không chính xác. Điều này là do độ chính xác hữu hạn của phép tính dấu chấm động. Hình 4.3 chỉ độ lớn thất thường và dấu



Hình 4.3: Kết quả đánh giá dấu chấm động biểu thức $f(x) = x^2e^{3x} - 3x^2e^{2x} + 3xe^x - 1$ bằng nhiều cách.

của các giá trị hàm khi giá trị là quá nhỏ đến nỗi bản chất rời rạc của hệ thống số dấu chấm động trở nên quan trọng.

Nếu giá trị tính toán của hàm có dấu sai vì đối số rất gần nghiệm, có thể xảy ra rằng khoảng giữa hai điểm trên dưới đã chọn trong phép chia đôi không chứa nghiệm. Ngay cả như vậy, các xấp xỉ tính toán sau đó sẽ ở trong lân cận của nghiệm. Người ta thường nói rằng *mã thuật toán chia đôi sẽ sinh ra một khoảng có độ dài chỉ định chứa nghiệm vì* $f(B)f(C) < 0$. Điều này là hời hợt. Nên được lượng hóa bằng cách nói rằng điều này là thực, hay nghiệm đã được tìm thấy chính xác như độ chính xác cho phép. Diễn ngữ "chính xác như độ chính xác cho phép" ở đây có nghĩa là $f(z)$ tính toán triệt tiêu, hay một trong các giá trị tính toán $f(B)$, $f(C)$ có dấu sai.

Một giả thiết cơ bản của phương pháp chia đôi là $f(x)$ liên tục. Không có gì ngạc nhiên phương pháp có thể thất bại khi $f(x)$ không liên tục. Vì mã

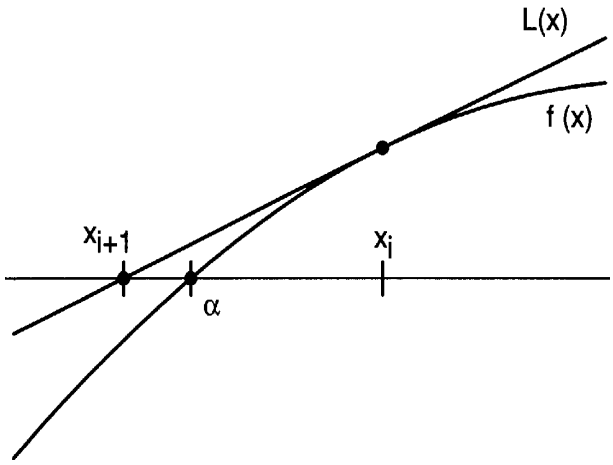
phép chia đôi không chú ý tới các giá trị của hàm, nó không thể nói sự khác nhau giữa cực điểm bội lẻ và nghiệm bội lẻ (trừ phi nó "nỗ lực" đánh giá $f(x)$ một cách chính xác tại điểm cực và nơi có overflow). Như vậy, chẳng hạn, nếu mã phép chia đôi được cho trước hàm $\tan(x)$ và được yêu cầu tìm nghiệm trong $[5, 7]$, nó sẽ không gặp khó khăn. Nhưng nếu được yêu cầu tìm nghiệm trong $[4, 7]$, nó sẽ không nhận ra có nghiệm trong khoảng này vì sự đổi dấu do cực điểm đơn loại bỏ sự đổi dấu do nghiệm đơn. Và, tệ nhất là, nếu được yêu cầu tìm nghiệm trong $[4, 5]$, nó sẽ định vị cực điểm hoặc gây ra overflow. Chúng ta thấy ở đây lý do khác để định tỷ lệ (scaling): loại bỏ các cực điểm lẻ bằng cách định tỷ lệ loại sự đổi dấu mà có thể làm cho phép

chia đôi xác định cực điểm lẻ thay vì không điểm. Ở đây điều này được thực hiện bằng cách $F(x) = \cos(x)\tan(x) = \sin(x)$. Vì lẽ khả năng xảy ra việc xác định cực điểm bội lẻ, nên cẩn trọng khi dùng mã chia đôi để kiểm tra thặng dư $f(z)$ của một nghiệm z được đưa ra - sẽ rất lúng túng để khẳng định z dẫn đến giá trị rất nhỏ của $f(z)$ khi thực sự nó dẫn đến một giá trị rất lớn!

Mã chia đôi có thể hội tụ tới cực điểm vì nó không dùng giá trị $f(M)$, mà chỉ dấu của nó. Vì điều này tốc độ hội tụ của nó là như nhau dù nghiệm là đơn hay không và dù hàm số là trơn hay không. Các phương pháp khác sự hội tụ là nhanh hơn nhiều khi nghiệm là đơn và hàm là trơn, nhưng chúng làm việc không tốt nếu những điều kiện này không được thỏa.

Phép chia đôi có một số ưu điểm. Miễn là một khoảng giữa hai điểm trên dưới ban đầu có thể được tìm thấy, nó sẽ hội tụ bất chấp khoảng ban đầu chứa nghiệm lớn đến đâu đi nữa. Dễ lựa chọn khi nào xấp xỉ là đủ tốt. Nó hội tụ khá nhanh và tốc độ hội tụ độc lập với bội của nghiệm và tính trơn của hàm. Phương pháp đối xử tốt với độ chính xác hữu hạn.

Phép chia đôi cũng có một số hạn chế. Nếu có một số lẻ không điểm giữa B và C, nó sẽ không nhận ra có bất kỳ không điểm nào vì không có sự đổi dấu. Đặc biệt, nó không thể tìm không điểm bội chẵn ngoại trừ do sự cố. Nó có thể bị đánh lừa bởi các cực điểm. Một bất lợi chính là với các không điểm đơn, mà thường xảy ra như vậy, có các phương pháp hội tụ nhanh hơn nhiều. Không có cách đáng tin để tìm một nghiệm riêng cũng như không có cách tìm tất cả các nghiệm. Đây là sự khó chịu với tất cả các phương pháp, nhưng một số phương pháp tính nghiệm tốt hơn khi nghiệm gần giá trị đề nghị.



Hình 4.4: Phương pháp Newton.

4.3 Phương pháp Newton - phương pháp cát tuyến

Phương pháp chia đôi không tổng quát hóa cho hàm biến phức cũng như không dùng cho các hàm nhiều biến. Bây giờ chúng ta tiếp tục với hai phương pháp tốt hơn phép chia đôi về, dù không phải là tất cả, vài khía cạnh. Cả hai phương pháp xấp xỉ $f(x)$ bằng đường thẳng $L(x)$ và rồi xấp xỉ nghiệm của phương trình $f(x) = 0$ bởi nghiệm của $L(x) = 0$.

Phương pháp Newton lấy tiếp tuyến $L(x)$ của $f(x)$ tại xấp xỉ x_i trước và xấp xỉ tiếp theo (lặp) x_{i+1} là nghiệm của $L(x) = 0$. Một cách tương đương, xấp xỉ $f(x)$ bởi số hạng tuyến tính của khai triển Taylor tại x_i,

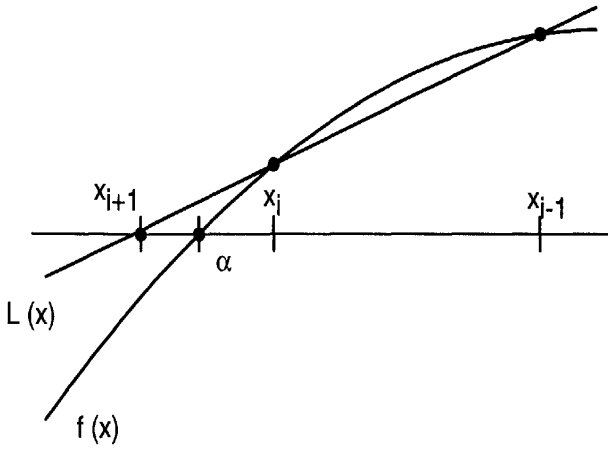
$$f(x) \approx f(x_i) + f'(x_i)(x - x_i),$$

phương trình xấp xỉ cho nghiệm (giả sử $f'(x_i) \neq 0$).

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}. \tag{4.5}$$

Khi việc tính $f'(x_i)$ gặp bất tiện hoặc quá quá phức tạp ta có thể dùng tỉ sai phân để xấp xỉ nó,

$$f'(x_i) \approx \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}.$$



Hình 4.5: Phương pháp cát tuyến.

Nếu $f(x_i) \neq f(x_{i-1})$ thì (4.5) thành

$$x_{i+1} = x_i - f(x_i)\frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})}. \tag{4.6}$$

Về mặt hình học cách làm này tương đương với việc xấp xỉ $f(x)$ bằng cát tuyến đi qua hai điểm lặp trước đó, $(x_{i-1}, f(x_{i-1}))$ và $(x_i, f(x_i))$. Vì lẽ này phương pháp được gọi là cát tuyến. Chú ý, phương pháp cát tuyến cần đến hai giá trị lặp trước đó trong khi phương pháp Newton chỉ cần một.

Thí dụ 4.2. Cho $f(x) = x^2 - 2$. Giải phương trình $f(x) = 0$ bằng hai phương pháp Newton và cát tuyến. Lấy $x_1 = 2, x_2 = 3$.

Kết quả tính bằng phương pháp Newton, lấy $x_1 = 2$:

i	x_i	$\lvert\alpha - x_i\rvert$
1	2.0	0.5858
2	1.5	0.0858
3	1.41666666666667	0.0025
4	1.41421568627451	2.1239×10^{-6}
5	1.41421356237469	1.5947×10^{-12}

Kết quả tính bằng phương pháp cát tuyến, lấy $x_1 = 2$, $x_2 = 3$:

i	x_i	$\lvert\alpha - x_i\rvert$
1	2.0	0.5858
2	3.0	1.5858
3	1.6000	0.1858
4	1.4783	0.0640
5	1.4181	0.0039
6	1.4143	1.1666×10^{-7}
7	1.4142	3.5254×10^{-12}

Qua thí dụ này ta thấy phương pháp Newton và cát tuyến nhanh hơn phương pháp chia đôi rất nhiều (so sánh với kết quả ở thí dụ 4.1) ∘

Trong trường hợp phương trình $f(x) = 0$ có nghiệm đơn α thì phương pháp Newton và phương pháp cát tuyến hội tụ nhanh hơn phương pháp chia đôi trước đây. Trước hết, ta xét phương pháp Newton. Từ (4.5) ta có

$$\alpha - x_{i+1} = \alpha - x_i + \frac{f(x_i)}{f'(x_i)}.$$

Khi x_i khá gần α ta có nhờ công thức Taylor:

$$\begin{aligned} f(x_i) &\approx f(\alpha) + f'(\alpha)(x_i - \alpha) + \frac{(x_i - \alpha)^2}{2} f''(\alpha), \\ f'(x_i) &\approx f'(\alpha) + (x_i - \alpha) f''(\alpha). \end{aligned}$$

Vì α là nghiệm đơn nên $f(\alpha) = 0$, $f'(\alpha) \neq 0$, và như vậy,

$$\begin{aligned} \alpha - x_{i+1} &\approx \alpha - x_i + \frac{f'(\alpha)(x_i - \alpha) + \frac{(x_i-\alpha)^2}{2} f''(\alpha)}{f'(\alpha) + (x_i - \alpha) f''(\alpha)} \\ \alpha - x_{i+1} &\approx -(x_i - \alpha)^2 \frac{f''(\alpha)}{2f'(\alpha)}. \end{aligned}$$

Nếu x_i gần nghiệm đơn thì sai số trong x_{i+1} cỡ một hằng số nhân với bình phương sai số trong x_i. Vì lý do này sự hội tụ của dãy lặp được gọi là *hội tụ bậc hai* (quadratic convergence).

Tương tự với phương pháp cát tuyến (4.6), ta có

$$\alpha - x_{i+1} \approx -(x_i - \alpha)(x_{i-1} - \alpha)\frac{f''(\alpha)}{2f'(\alpha)}.$$

Phương pháp này hội tụ không nhanh bằng phương pháp Newton. Nhưng nó nhanh hơn phương pháp chia đôi nhiều.

Định nghĩa 4.1. Cho dãy $\{x_n\}$ hội tụ về α. Bởi định nghĩa, *cấp của sự hội tụ* là số thực p lớn nhất sao cho giới hạn

$$\lim_{n\to\infty} \frac{|x_{n+1} - \alpha|}{|x_n - \alpha|^p} = \gamma \neq 0. \tag{4.7}$$

Khi đó, ta nói phương pháp hội tụ với tốc độ p theo hằng số γ.

Như vậy, với phương pháp Newton,

$$\lim_{x_i\to\alpha} \frac{|x_{i+1} - \alpha|}{|x_i - \alpha|^2} = C \neq 0;$$

nghĩa là $p = 2$. Đối với phương pháp cát tuyến, với $p = (1 + \sqrt{5})/2 \approx 1.618$, có thể chứng minh

$$\lim_{x_i\to\alpha} \frac{|x_{i+1} - \alpha|}{|x_i - \alpha|^p} = c \neq 0.$$

Định lý 4.1. *Phương pháp cát tuyến (4.6) với các giá trị lặp ban đầu x_0, x_1, hội tụ tới không điểm đơn α của $f(x)$ nếu x_0, x_1 nằm trong đoạn đóng đủ bé chứa α trên đó $f'(x)$ và $f''(x)$ tồn tại liên tục và $f'(x)$ không triệt tiêu.*

Chứng minh. Trước hết ta thiết lập biểu thức liên hệ các giá trị hàm tại ba bước lặp liên tiếp x_{i-1}, x_i, x_{i+1}. Gọi $L(x)$ là đa thức bậc nhất nội suy $f(x)$ trên tập $\{x_{i-1}, x_i\}$. Giá trị lặp x_{i+1} là không điểm của $L(x)$. Theo công thức sai số của phép nội suy, ta có

$$f(x_{i+1}) - L(x_{i+1}) = (x_{i+1} - x_{i-1})(x_{i+1} - x_i)\frac{f''(\xi)}{2}$$

hay, vì $L(x_{i+1}) = 0$

$$f(x_{i+1}) = (x_{i+1} - x_{i-1})(x_{i+1} - x_i)\frac{f''(\xi)}{2} \tag{4.8}$$

với ξ thích hợp. Mặt khác, từ (4.6) ta có hai hệ thức:

$$x_{i+1} - x_i = -\frac{(x_i - x_{i-1})f(x_i)}{f(x_i) - f(x_{i-1})}, \tag{4.9}$$

$$x_{i+1} - x_{i-1} = -\frac{(x_i - x_{i-1})f(x_{i-1})}{f(x_i) - f(x_{i-1})}. \tag{4.10}$$

Hệ thức thứ ba nhận được nhờ định lý giá trị trung gian cho đạo hàm:

$$\frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} = f'(\eta), \tag{4.11}$$

trong đó η nằm giữa x_{i-1} và x_i, chưa biết. Tổ hợp các phương trình (4.8) - (4.11) ta thu được

$$f(x_{i+1}) = f(x_i)f(x_{i-1})\frac{f''(\xi)}{2[f'(\eta)]^2}.$$

Dưới các giả thiết của định lý, trên khoảng thích hợp, ta có

$$|f''(x)| < M_2, \quad 0 \le m_1 \le |f'(x)| \le m_2 \tag{4.12}$$

và vì nghiệm cần tìm là nghiệm đơn, nên các chặn ở đây và biểu thức của $f(x_{i+1})$ ở trên cho

$$|f(x_{i+1})| \le |f(x_i)||f(x_{i-1})|\frac{M_2}{2m_1^2}.$$

Nếu đặt

$$\epsilon_i = |f(x_i)|\frac{M_2}{2m_1^2}$$

thì bất đẳng thức trên dẫn đến

$$\epsilon_{i+1} \leq \epsilon_{i-1}\epsilon_i.$$

Để ý rằng, theo định lý giá trị trung gian cho đạo hàm, ta có:

$$\begin{aligned} |f(x_0)| &= |f(\alpha) + (x_0 - \alpha)f'(\eta_1)| \leq |x_0 - \alpha| m_2, \\ |f(x_1)| &= |f(\alpha) + (x_1 - \alpha)f'(\eta_2)| \leq |x_1 - \alpha| m_2. \end{aligned}$$

Điều này ám chỉ

$$\epsilon = \max\{\epsilon_0, \epsilon_1\} < 1,$$

nếu x_0, x_1 đủ gần α. Từ đây, dễ dàng suy ra

$$\begin{aligned} \epsilon_2 &\leq \epsilon^2, \\ \epsilon_3 &\leq \epsilon^2\epsilon = \epsilon^3, \\ \epsilon_4 &\leq \epsilon^3\epsilon^2 = \epsilon^5, \\ &\vdots \\ \epsilon_i &\leq \epsilon^{\delta_i}, \end{aligned}$$

trong đó

$$\delta_i = \frac{1}{\sqrt{5}} \left[\left(\frac{1+\sqrt{5}}{2} \right)^{i+1} - \left(\frac{1-\sqrt{5}}{2} \right)^{i+1} \right]. \tag{4.13}$$

Vì

$$\left| \frac{1-\sqrt{5}}{2} \right| < 1 < \frac{1+\sqrt{5}}{2}$$

nên khi i lớn

$$\delta_i \approx \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2} \right)^{i+1}.$$

Như vậy, $\delta_i \to \infty$, và vì $0 < \epsilon < 1$ nên $\epsilon_i \to 0$.

Từ cách đặt của ϵ_i và vì

$$|f(x_i)| = |f(x_i) - f(\alpha)| = |x_i - \alpha||f'(v)| \geq |x_i - \alpha| m_1,$$

ta thấy $x_i \to \alpha$. ■

Các phương pháp lặp hội tụ với tốc độ $r > 1$ được gọi là *hội tụ siêu tyến tính* (superlinearly convergent). Như đã thấy, phương pháp Newton và phương pháp cát tuyến là hội tụ siêu tuyến tính khi được dùng để tính nghiệm đơn. Đáng tiếc, nó không còn như vậy khi tính nghiệm bội. Thật vậy, xét trường hợp phương pháp Newton, nếu x_i gần nghiệm bội m ($m > 1$), α, thì

$$\begin{aligned} f(x) &\approx \frac{(x-\alpha)^m}{m!} f^{(m)}(\alpha), \\ f'(x) &\approx \frac{(x-\alpha)^{m-1}}{(m-1)!} f^{(m)}(\alpha). \end{aligned}$$

Điều này ám chỉ

$$x_{i+1} - \alpha = x_i - \alpha - \frac{f(x_i)}{f'(x_i)} \approx \frac{m-1}{m}(x_i - \alpha).$$

Biểu thức này chứng tỏ rằng, với một nghiệm bội m, phương pháp Newton chỉ hội tụ tuyến tính với hằng số $(m-1)/m$.

Thí dụ 4.3. Phương trình $x^{20} - 1 = 0$ có nghiệm đơn $\alpha = 1$. Dùng phương pháp Newton với $x_1 = 1/2$. Nhận xét gì về quá trình tính.

Nếu ta lấy $x_1 = 1/2$ thì từ (4.5)

$$x_2 = \frac{1}{2} - \frac{(1/2)^{20} - 1}{20(1/2)^{19}} = 26214.875.$$

Vì tiếp tuyến hầu như nằm ngang, một giá trị lặp ban đầu tốt (gần với nghiệm chính xác) nhưng lại cho giá trị lặp tiếp theo rất xấu (rất xa nghiệm chính xác)!

Cũng vậy, nếu $x_i \gg 1$ thì

$$x_{i+1} = x_i - \frac{x_i^{20} - 1}{20x_i^{19}} \approx x_i - \frac{x_i^{20}}{20x_i^{19}} = \frac{19}{20} x_i.$$

So sánh sự xấp xỉ nghiệm

$$\frac{x_{i+1}-1}{x_i-1} \approx \frac{x_{i+1}}{x_i} \approx \frac{19}{20}$$

ta thấy xấp xỉ là bậc 1, quá trình lặp có tốc độ rất chậm, chậm hơn cả phương pháp chia đôi. Tại sao 1 là nghiệm đơn của phương trình $x^{20}-1=0$ mà lại xảy ra như vậy? Thật ra, trong mặt phẳng phức, nghiệm của phương trình đang xét là căn bậc 20 của đơn vị, gồm 20 điểm phân bố đều trên vòng tròn đơn vị. Nhưng ở khoảng cách xa ≈ 26000 từ vị trí x_2 thì các nghiệm này hầu như trùng nhau, nghĩa là 1 là nghiệm bội 20.

Phương pháp Newton chỉ hội tụ bậc hai khi nghiệm là đơn. Ngay cả với nghiệm đơn, thí dụ này chỉ ra sự hội tụ bậc hai chỉ xảy ra khi các giá trị lặp đủ "gần" nghiệm. Nhưng "quá" gần nghiệm thì số học có độ chính xác hữu hạn lại ảnh hưởng đến tốc độ hội tụ ∘

Bây giờ ta xét ứng xử của phương pháp Newton và phương pháp cát tuyến khi tính toán với độ chính xác hữu hạn. Trong lân cận của nghiệm, giá trị tính toán của $f(x)$ thay đổi thất thường về độ lớn và dấu. Rất thường xảy ra là giá trị tính toán của $f(x)$ không có chữ số nào giống giá trị thực. Với một nghiệm đơn $|f'(\alpha)|$ là khác không, và nếu nghiệm không là điều kiện xấu thì $|f'(\alpha)|$ không nhỏ. Như một hệ quả, giá trị tính toán của đạo hàm cấp một thông thường có vài chữ số đúng. Từ đó, sự hiệu chỉnh tới x_i được tính bởi phương pháp Newton là rất nhỏ với độ chính xác hữu hạn giá trị lặp kế tiếp vẫn ở gần nghiệm ngay cả nếu nó di chuyển ra ngoài (khoảng chứa nghiệm) do $f(x_i)$ có dấu sai. Điều này giống như phương pháp chia đôi và được gọi là "ổn định với độ chính xác hữu hạn". Phương pháp cát tuyến thì ứng xử hoàn toàn khác. Sự hiệu chỉnh tới giá trị lặp hiện hành,

$$\frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})}$$

có những giá trị không lường trước được do độ chính xác hữu hạn. Rõ ràng giá trị lặp kế tiếp có thể nằm xa ngoài khoảng của độ chính xác hữu hạn.

Có một cách nhìn khác về phương pháp cát tuyến làm sáng tỏ bản chất của nó. Một cách tiếp cận tìm nghiệm α của $f(x)$ là dùng nội suy đa thức $P(x)$ theo nhiều giá trị $y_i = f(x_i)$ của $f(x)$ và rồi xấp xỉ α bởi nghiệm của đa thức này. Phương pháp cát tuyến là trường hợp nội suy tuyến tính. Các nội suy cấp cao hơn cho xấp xỉ $f(x)$ chính xác hơn, nói chung, dẫn đến sơ đồ

với tốc độ hội tụ cao hơn. Sơ đồ dựa trên nội suy bậc hai được gọi là *phương pháp Muller*. Nó có phức tạp hơn phương pháp cát tuyến (vì phải tìm nghiệm của phương trình bậc hai), và hội tụ nhanh hơn một chút. Với tất cả các phương pháp dựa trên nội suy đa thức có bậc lớn hơn 1, ta cần tìm nghiệm của đa thức khi tính giá trị lặp kế tiếp x_{i+1}. Để có sự hội tụ, nghiệm gần x_i nhất nên được chọn. Một khác biệt quan trọng giữa phương pháp Muller và phương pháp cát tuyến là đa thức bậc hai có thể có nghiệm phức. Ngay cả giá trị lặp x_i là thực và $f(x)$ là hàm thực, phương pháp Muller vẫn có thể sinh ra giá trị lặp phức. Đây là một khuyết điểm của phương pháp Muller.

4.4 Điểm bất động và phương pháp lặp

Phương pháp Newton là một thí dụ của các thủ tục trong đó một dãy điểm được tính từ công thức dạng

$$x_{n+1} = F(x_n) \quad (n \geq 1). \tag{4.14}$$

Thuật toán định bởi một phương trình như vậy được gọi là *phép lặp hàm* (functional iteration). Trong phương pháp Newton, hàm F được cho bởi

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

Trong phương pháp được biết với tên gọi *phương pháp Steffensen* (Steffensen's method), ta có

$$F(x) = x - \frac{[f(x)]^2}{f(x + f(x)) - f(x)}.$$

Có nhiều cách chọn F dưới dạng

$$F(x) = x - g(x),$$

nhưng dãy các điểm $\{x_n\}$ có thể không hội tụ. Nếu F liên tục và $x_n \to \alpha$ khi $n \to \infty$ thì $F(\alpha) = \alpha$, và ta gọi s là *điểm bất động* (fixed point) của hàm F. Một ánh xạ (hay hàm) F được gọi là *co* nếu tồn tại một số thực $\lambda < 1$ sao cho

$$|F(x) - F(y)| \leq \lambda |x - y| \tag{4.15}$$

với mọi x, y nằm trong miền xác định của F.

Định lý 4.2 (Định lý ánh xạ co)**.** *Cho F là ánh xạ co của tập đó $C \subset \mathbb{R}$ vào C. Thì F có duy nhất một điểm bất động. Hơn nữa, điểm bất động này là giới hạn của mọi dãy nhận được từ phương trình (4.14) với bất kỳ điểm khởi đầu $x_0 \in C$.*

Chứng minh. Dùng tính chất co và phương trình (4.14) ta có thể viết

$$|x_n - x_{n-1}| = |F(x_{n-1}) - F(x_{n-2})| \leq \lambda |x_{n-1} - x_{n-2}|.$$

Lặp lại chứng minh này ta được:

$$|x_n - x_{n-1}| \leq \lambda |x_{n-1} - x_{n-2}| \leq \lambda^2 |x_{n-2} - x_{n-3}| \leq \cdots \leq \lambda^{n-1} |x_1 - x_0|.$$

Vì x_n có thể viết dưới dạng

$$x_n = (x_n - x_{n-1}) + (x_{n-1} - x_{n-2}) + \cdots + (x_1 - x_0) + x_0 = x_0 + \sum_{i=1}^{n-1} (x_i - x_{i-1}).$$

Để chứng minh dãy $\{x_n\}$ hội tụ ta chỉ cần chứng minh chuỗi

$$\sum_{n=1}^{\infty} (x_n - x_{n-1})$$

hội tụ.

Do $|x_n - x_{n-1}| \leq \lambda^{n-1} |x_1 - x_0|$ với mọi n, và chuỗi $\sum \lambda^{n-1}$ hội tụ nên chuỗi

$$\sum_{n=1}^{\infty} |x_n - x_{n-1}|$$

hội tụ tuyệt đối (theo tiêu chuẩn so sánh).

Gọi α là giới hạn của dãy, ta có ngay $F(\alpha) = \alpha$ (chú ý ánh xạ co là hàm liên tục).

Tính duy nhất của điểm bất động được suy ra trực tiếp từ (4.15). Giả sử F có hai điểm bất động x và y, thì

$$|x - y| = |F(x) - F(y)| \leq \lambda |x - y|$$

Vì $\lambda < 1$ nên $x = y$. ■

Bây giờ ta phân tích sai số trong phương pháp lặp. Giả sử F có điểm bất động, α, và dãy $\{x_n\}$ xác định bởi công thức $x_{n+1} = F(x_n)$. Đặt

$$\epsilon_n = x_n - \alpha.$$

Nếu F' tồn tại và liên tục, thì theo định lý giá trị trung gian

$$x_{n+1} - \alpha = F(x_n) - F(\alpha) = F'(\xi_n)(x_n - \alpha)$$

hay

$$\epsilon_{n+1} = F'(\xi_n)\epsilon_n.$$

trong đó ξ_n là điểm nằm giữa x_n và α. Điều kiện $F'(x) < 1$ với mọi x bảo đảm sai số giảm khi n tăng. Nếu ϵ_n là nhỏ, thì ξ_n nằm gần α, và $F'(\xi_n) \approx F'(\alpha)$. Ta hy vọng tốc độ hội tụ lớn nếu $F'(\alpha)$ là nhỏ. Một trường hợp lý tưởng là $F'(\alpha) = 0$. Trong trường hợp đó, ta cần đến số hạng cộng thêm trong chuỗi taylor.

Giả sử tồn tại số nguyên q sao cho

$$F^{(k)}(\alpha) = 0 \quad \text{khi } 1 \le k < q \quad \text{nhưng } F^{(q)}(\alpha) \neq 0.$$

Từ công thức khai triển Taylor của $F(x_n)$ tại α, ta có

$$\begin{aligned}
\epsilon_{n+1} &= F(x_n) - F(\alpha) \\
&= F(\alpha + \epsilon_n) - F(\alpha) \\
&= \left[F(\alpha) + \epsilon_n F'(\alpha) + \frac{\epsilon_n^2}{2} F''(\alpha) + \cdots \frac{\epsilon_n^{q-1}}{(q-1)!} F^{(q-1)}(\alpha) + \frac{\epsilon_n^q}{q!} F^{(q)}(\xi_n) \right] - F(\alpha)
\end{aligned}$$

và vì vậy

$$e_{n+1} = \frac{\epsilon_n^q}{q!} F^{(q)}(\xi_n). \tag{4.16}$$

Do $x_n \to \alpha$ nên

$$\lim_{n\to\infty} \frac{|\epsilon_{n+1}|}{|\epsilon_n|^q} = \frac{1}{q!} |F^{(q)}(\alpha)|; \tag{4.17}$$

nghĩa là x_n hội tụ cấp q về α.

4.5 Tiêu chuẩn dừng phép lặp

Các thuật toán trình bày ở trên có thể cài đặt dưới dạng chương trình con, tìm nghiệm xấp xỉ phương trình $f(x) = 0$. Với mục đích giới thiệu cách cài đặt thuật toán lặp, ở đây, ta trình bày một chương trình dạng function cho phương pháp cát tuyến. Nhưng trước hết, để bảo đảm thuật toán dừng, ta phải chọn tiêu chuẩn dừng cho phép giải lặp. Thường dãy lặp được dừng theo một trong ba tiêu chuẩn sau:

(1) $|f(x_n)| < \epsilon_1$;

(2) $|x_{n+1} - x_n| < \epsilon_2$;

(3) $\dfrac{|x_{n+1} - x_n|}{|x_{n+1}|} < \epsilon_3$.

Ở đây, các ϵ_i là độ chính xác cho trước. Trong thực hành, người ta thường dùng tiêu chuẩn (1) hoặc (2); trường hợp nghiệm quá lớn hoặc quá nhỏ so với 1, người ta dùng tiêu chuẩn (3). Cũng cần nhắc lại rằng, khi dùng tiêu chuẩn (1) - thặng dư bé hơn ϵ_1 - cần lưu ý đến cách chọn tỉ lệ khi thiết lập phương trình.

Trong function secantm, giải phương trình $f(x) = 0$, dưới đây cho phép người dùng chọn tiêu chuẩn dừng cho phép lặp thông qua biến opt trong danh sách đối số. Ngoài ra, function secantm còn cho phép in giá trị lặp trung gian nếu được yêu cầu thông qua biến trace.

```
function [x,flag]=secantm(f,x1,x2,N,EPS,opt,trace,varargin)
% tim nghiem phuong trinh f(x)=0 bang pp cat tuyen
% update: 11/11/09
% input:
% f - dieu khien ham cua ham f(x)
% x1, x2 - hai gia tri ban dau
% N - so lan lap toi da
% EPS - sai so toi da
% opt - tieu chuan dung; opt=1 - tc 1, opt=2 - tc 2, opt=3 - tc 3; mac dinh=1
% trace - in day lap; trace=0 - khong in, ttrac~=0 in; mac dinh=0
% output:
% x - nghiem xap xi
% flag - co; flag=0 - khong giai duoc, flag~=0 so buoc lap thuc hien
flag=0;
```

```
xold1=x1;
xold2=x2;
if nargin<7, trace=0; end
if nargin<6, opt=1; trace=0; end
if nargin<5, EPS=10^-12; opt=1; trace=0; end
if nargin<4, N=100; EPS=10^-12; opt=1; trace=0; end
if opt~=1&opt~=2&opt~=3, opt=1; end
if N<1, N=100; end
for k=1:N
    x=xold2-f(xold2)*(xold2-xold1)/(f(xold2)-f(xold1));
    switch opt
        case 1,
            if abs(f(x))<EPS
                flag=k
                break;
            end
        case 2,
            if abs(x-xold2)<EPS
                flag=k;
                break;
            end
        case 3,
            if abs(x-xold2)/abs(x)<EPS
                flag=k;
                break;
            end
    end
    if trace~=0, disp(x); end
    xold1=xold2;
    xold2=x;
end
```

Chú giải

⋆ **cell array** - mảng (ma trận) tế bào. Đây là kiểu dữ liệu cho phép gọi tên và thao tác với một nhóm dữ liệu có nhiều kích cỡ và nhiều kiểu.

⋆ **varargin** - biến chiều dài danh sách đối số nhập. Cho phép số lượng đối số bất kỳ cho một function. Biến varargin là một mảng tế bào chứa các đối số tùy chọn cho một function. varargin phải được phát biểu như là đối số nhập cuối cùng và tụ tập tất cả các đối số nhập từ điểm đó trở đi. Bằng cách dùng biến này function secantm có thể được gọi với số đối số ít hơn 6.

⋆ **nargin** - số đối số nhập của function. Bên trong thân của function, do người dùng định nghĩa, nargin trả về số đối số nhập được dùng để gọi function.

⋆ function secantm gọi hàm $f(x)$ khi tính toán. Để truyền hàm $f(x)$ bằng đối số ta dùng biến điều khiển hàm.

Cú pháp:

`dieukhien = @ten_ham`

`dieukhien = @(ds_doiso)ham_nac_danh`

Mô tả:

`dieukhien = @ten_ham` trả về một điều khiển tới hàm Matlab được chỉ định.

Một điều khiển hàm (function handle) là một giá trị Matlab cung cấp một phương tiện gọi hàm cách gián tiếp. Ta có thể truyền các điều khiển hàm khi gọi các hàm khác. Ta cũng có thể lưu trữ các điều khiển hàm trong các cấu trúc dữ liệu để dùng sau này. Điều khiển hàm là một trong các kiểu dữ liệu chuẩn của Matlab (`function_handle`).

`dieukhien = @(ds_doiso)ham_nac_danh` xây dựng một hàm nặc danh và trả về một điều khiển tới hàm đó. Thân của hàm, bên phải dấu ngoặc đơn, là một lệnh hay phát biểu Matlab. `ds_doiso` là danh sách các đối số nhập, cách nhau bằng dấu phẩy. Thi hành hàm bằng cách gọi nó nhờ điều khiển hàm, `dieukhien`. Thí dụ, lệnh dưới đây tạo một hàm nặc danh tìm căn bậc hai của một số. Khi ta gọi hàm này, Matlab gán giá trị ta truyền vào biến `x`, và rồi dùng `x` trong phương trình `x.^2`:

```
sqr = @(x) x.^2;
```

Tác tử `@` xây dựng một điều khiển hàm cho hàm này, và gán điều khiển tới biến xuất `sqr`. Để thi hành hàm `sqr` định nghĩa ở trên, gõ

```
a = sqr(5)
a =
   25
```

⋆ Để việc gọi function secantm được mềm dẻo ta dùng biến varargin như chỉ ra ở trên. Khi đó, các đối số không được nhập sẽ gán các giá trị mặc định. Điều này được thực hiện nhờ cấu trúc chuyển (rẽ nhánh): **switch** - chuyển (rẽ nhánh) giữa nhiều trường hợp dựa trên biểu thức.

Dạng tổng quát cũa lệnh switch:

```
switch bieuthuc
        case th1,
             (các) lệnh
        case th1, th2, th3,...
             (các) lệnh
```

```
        otherwise,
            (các) lệnh
end
```

⋆ Matlab cung cấp function **fzero** tìm nghiệm phương trình $f(x) = 0$.

Cú pháp thường dùng: `x = fzero(fun,x0,options)`, trong đó:

fun là điều khiển hàm;

x0 là giá trị lặp ban đầu;

options là các tùy chọn được tạo bằng function optimset của Matlab.

⋆ **optimset** - tạo hay biên tập các tùy chọn. Với function fzero thường ta chỉ cần hiển thị giá trị lặp trung gian nên cú pháp được dùng:

`options = optimset('display',giatri)`

trong đó giatri là một trong các giá trị sau 'off' - không xuất, 'iter' - xuất các kết quả trung gian, 'final' - chỉ xuất kết quả cuối cùng, 'notify' - chỉ xuất nếu quá trình không hội tụ ▷

Bây giờ ta gọi function secantm giải phương trình $x^2 - 2 = 0$ (thí dụ 4.1).

```
>> clear all
>> f=@(x) x^2-2;
>> [x,flag]=secantm(f,2,3,100,10^-12,1,1)
```

Kết quả trả về

```
    1.6000
    1.4783
    1.4181
    1.4143
    1.4142
    1.4142
x =
    1.4142
flag =
     7
```

Dùng function fzero, trước hết ta đặt tùy chọn rồi gọi function

```
>> clear all
>> options = optimset('Display','iter');
>> x=fzero(f,2,options)
```

Kết quả trả về

```
Search for an interval around 2 containing a sign change:
 Func-count    a          f(a)             b          f(b)        Procedure
     1              2             2             2             2   initial interval
     3        1.94343       1.77693       2.05657       2.22947   search
     5           1.92        1.6864          2.08        2.3264   search
     7        1.88686       1.56025       2.11314       2.46535   search
     9           1.84        1.3856          2.16        2.6656   search
    11        1.77373        1.1461       2.22627        2.9563   search
    13           1.68        0.8224          2.32        3.3824   search
    15        1.54745      0.394607       2.45255       4.01499   search
    16           1.36       -0.1504       2.45255       4.01499   search

Search for a zero in the interval [1.36, 2.45255]:
 Func-count    x          f(x)             Procedure
    16           1.36       -0.1504        initial
    17        1.39945    -0.0415434        interpolation
    18        1.41435   0.000384399        interpolation
    19        1.41421 -2.01697e-006        interpolation
    20        1.41421 -9.69058e-011        interpolation
    21        1.41421  4.44089e-016        interpolation
    22        1.41421  4.44089e-016        interpolation

Zero found in the interval [1.36, 2.45255]
x =
    1.4142
```

4.6 Hệ phương trình phi tuyến

Một bài toán thường xuất hiện trong toán học tính toán là tìm một vài hoặc tất cả các nghiệm của một hệ gồm n phương trình phi tuyến với n ẩn. Những bài toán như vậy tổng quát và khó hơn nhiều so với bài toán một phương trình một ẩn số. Có thể thấy ngay phương pháp chia đôi không áp dụng được (mở rộng được) cho trường hợp này. Phương pháp cát tuyến có thể tổng quát hóa cho trường hợp này nhưng cách làm không hiển nhiên vì sự phức tạp hình học khi số chiều lớn. Với phương pháp Newton thì khác, sự tổng quát hóa ra trường hợp n phương trình n ẩn số rất tự nhiên và rất ...đẹp. Để đơn giản việc trình bày, xét hệ gồm hai phương trình theo hai ẩn:

$$\begin{aligned} f(x,y) &= 0 \\ g(x,y) &= 0 \end{aligned} \tag{4.18}$$

Viết dưới dạng vectơ

$$\mathbf{h}(\mathbf{w}) = \mathbf{0},$$

trong đó

$$\mathbf{w} = \begin{bmatrix} x \\ y \end{bmatrix}, \quad \mathbf{h} = \begin{bmatrix} f \\ g \end{bmatrix}.$$

Tương tự như tròng trường hợp 1-chiều (phương pháp Newton), khai triển hàm f, g nhờ công thức Taylor chỉ giữ lại các số hạng bậc nhất

$$\begin{aligned} f(x_0, y_0) + \frac{\partial f}{\partial x}(x_0, y_0)(x - x_0) + \frac{\partial f}{\partial y}(x_0, y_0)(y - y_0) &= 0 \\ g(x_0, y_0) + \frac{\partial g}{\partial x}(x_0, y_0)(x - x_0) + \frac{\partial g}{\partial y}(x_0, y_0)(y - y_0) &= 0 \end{aligned}$$

Xấp xỉ kế tiếp $(x_1, y_1) = (x_1 + \Delta x_1, y_1 + \Delta y_1)$ có thể tìm bằng cách giải hệ phương trình

$$\begin{aligned} \frac{\partial f}{\partial x}(x_0, y_0)\Delta x_1 + \frac{\partial f}{\partial y}(x_0, y_0)\Delta y_1 &= -f(x_0, y_0) \\ \frac{\partial g}{\partial x}(x_0, y_0)\Delta x_1 + \frac{\partial g}{\partial y}(x_0, y_0)\Delta y_1 &= -g(x_0, y_0) \end{aligned}$$

xác định $(\Delta x_1, \Delta y_1)$.

Nếu ký hiệu

$$\mathbf{J}(\mathbf{w}_k) = \begin{bmatrix} \frac{\partial f}{\partial x}(x_k, y_k) & \frac{\partial f}{\partial y}(x_k, y_k) \\ \frac{\partial g}{\partial x}(x_k, y_k) & \frac{\partial g}{\partial y}(x_k, y_k) \end{bmatrix}$$

là *ma trận jacobi* của hệ phương trình (4.18). Thì phương pháp Newton cho hệ hai phương trình hai ẩn là

$$\mathbf{J}(\mathbf{w}_k)\Delta\mathbf{w}_k = -\mathbf{h}(\mathbf{w}_k). \tag{4.19}$$

(x_0, y_0)	Nghiệm (x, y)	Số lần lặp
$(1.2, 2.5)$	$(1.3364, 1.7542)$	4
$(-2.0, 2.5)$	$(-0.9013, -2.0866)$	9
$(-1.2, -2.5)$	$(-0.9013, -2.0866)$	4
$(2.0, -2.5)$	$(-3.0016, 0.1481)$	19

Bảng 4.1: Kết quả số thí dụ 4.4

Thí dụ 4.4. Giải hệ phương trình

$$\begin{aligned} x^2 + xy^3 - 9 &= 0 \\ 3x^2y - y^3 - 4 &= 0 \end{aligned} \tag{4.20}$$

Vì

$$\frac{\partial f}{\partial x} = 2x + y^3, \qquad \frac{\partial f}{\partial y} = 3xy^2,$$

$$\frac{\partial g}{\partial x} = 6xy, \qquad \frac{\partial g}{\partial y} = 3x^2 - 3y^2,$$

hệ được giải tại mỗi bước lặp là

$$\begin{bmatrix} 2x_k + y_k^3 & 3x_k y_k^2 \\ 6x_k y_k & 3x_k^2 - 3y_k^2 \end{bmatrix} \begin{bmatrix} \Delta x_k \\ \Delta y_k \end{bmatrix} = - \begin{bmatrix} x_k^2 + x_k y_k^3 - 9 \\ 3x_k^2 y_k - y_k^3 - 4 \end{bmatrix}.$$

Bảng 4.4 cho kết quả số với các điểm khởi đầu (x_0, y_0) khác nhau. Trong tất cả các trường hợp phép lặp dừng khi

$$\max \left\{ \|\mathbf{h}\|, \frac{\|\Delta \mathbf{w}\|}{\|\mathbf{w}\|} \right\} \leq 10^{-6}.$$

Kết quả tính toán chứng tỏ rằng hệ có ít nhất ba nghiệm, mỗi nghiệm được tìm thấy phụ thuộc vào điểm khởi đầu (x_0, y_0). ∘

Cũng như phương pháp Newton cho hàm một biến, có thể chứng tỏ rằng nếu $\mathbf{h}$ hai lần khả vi gần nghiệm $\mathbf{a}$ của $\mathbf{h}(\mathbf{w}) = \mathbf{0}$, nếu ma trận Jacobi tại $\mathbf{a}$, $\mathbf{J}(\mathbf{a})$, không suy biến, và nếu $\mathbf{w}_0$ đủ gần $\mathbf{a}$, thì phương pháp Newton sẽ hội tụ về $\mathbf{a}$ và sự hội tụ là cấp hai.

Một khó khăn trong thực hành là tìm điểm khởi đầu đủ gần để quá trình lặp dẫn tới nghiệm cần tìm. Sự hiểu biết về bài toán và sự mở rộng nghiệm có thể rất hữu ích ở đây. Một cách tiếp cận tổng quát là liên hệ việc tìm nghiệm $\mathbf{w}$ của $\mathbf{h}(\mathbf{w}) = \mathbf{0}$ với sự cực tiểu hóa thặng dư $R(\mathbf{w}) = \|\mathbf{h}(\mathbf{w})\|^2 = [f(\mathbf{w})]^2 + [g(\mathbf{w})]^2$. Rõ ràng hàm này có cực tiểu là 0 tại mọi nghiệm của $\mathbf{h}(\mathbf{w}) = \mathbf{0}$. Ý tưởng là chú ý đến sự biến thiên $\Delta\mathbf{w}_k$ tính từ phương pháp Newton như việc cho hướng theo đó ta tìm một giá trị λ sao cho phép quả lặp

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \lambda\Delta\mathbf{w}_k$$

cho giá trị nhỏ của thặng dư:

$$R(\mathbf{w}_{k+1}) < R(\mathbf{w}_k).$$

Điều này luôn luôn thực hiện được vì cho đến khi nhận được nghiệm,

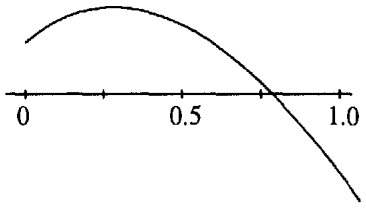
$$\left[\frac{\partial}{\partial\lambda}R(\mathbf{w}_k + \lambda\Delta\mathbf{w}_k)\right]_{\lambda=0} = -2R(\mathbf{w}_k) < 0.$$

Có nhiều chi tiết thực hành phải được vạch ra. Chẳng hạn, không nhất thiết, hay ngay cả mong muốn, tìm giá trị của λ cực tiểu hóa thặng dư. Các phương pháp thuộc loại này được gọi là các phương pháp Newton chùng (damped Newton methods). Một cài đặt cẩn thận phương pháp này sẽ phát sinh dãy lặp hội tụ khi mà phương pháp Newton thất bại. Trong trường hợp cả hai phương pháp đều hội tu thì hiệu quả của chúng là như nhau.

Câu hỏi và bài tập

4.1. Thặng dư của một nghiệm đưa ra r của $F(x)$ là $F(r)$. Ta thường thấy phát biểu: thặng dư là "nhỏ" nên nghiệm phải "tốt". Điều này có xác thực không? Vai trò của việc. Việc định tỉ lệ đóng vai trò gì?

4.2. Phân biệt nghiệm đơn và nghiệm bội bằng đồ thị như thế nào? Giải thích bằng đồ thị cách xác định tốt các nghiệm. So sánh với bài tập 4.1.



Hình 4.6: Bài tập 4.3.

4.3. Đánh giá bằng hình học nghiệm của hàm $F(x)$ mà đồ thị được cho bên dưới.

(a) Với khoảng chứa nghiệm ban đầu $[0.0, 1.0]$ ba khoảng chứa nghiệm kế tiếp là những khoảng nào?

(b) Nếu $x_1 = 0.0$ và $x_2 = 1.0$, đánh dấu trên đồ thị vị trí xấp xỉ của x_3 bằng cách dùng một bước của phương pháp cát tuyến.

(c) Nếu $x_1 = 0.5$, đánh dấu trên đồ thị vị trí xấp xỉ của x_2 và x_3 bằng cách dùng hai bước của phương pháp Newton.

4.4. Đa thức $f(x) = x^3 - 2x - 5$ có một nghiệm trong $[2, 3]$.

(a) Chứng tỏ rằng $[2, 3]$ là khoảng chứa nghiệm của $f(x)$.

(b) Áp dụng bốn bước của phương pháp chia đôi để giảm khoảng chứa nghiệm xuống còn 1/16.

(c) Tính x_3 và x_4 bằng phương pháp cát tuyến bắt đầu với $x_1 = 3$ và $x_2 = 2$.

(d) Tính x_2, x_3, và x_4 dùng phương pháp Newton với $x_1 = 2$.

4.5. Để tìm nơi $\sin x = x/2$ với $x > 0$,

(a) Tìm một khoảng thích hợp chứa nghiệm của một hàm phù hợp $f(x)$.

(b) Áp dụng bốn bước của phương pháp chia đôi để giảm khoảng chứa nghiệm xuống còn 1/16.

(c) Tính x_3 và x_4 bằng phương pháp cát tuyến bắt đầu với x_1 và x_2 lấy bằng các giá trị của khoảng.

(d) Tính $x2$, x_3, và x_4 dùng phương pháp Newton với x_1 là điểm giữa của khoảng.

4.6. Có bao nhiêu phép đánh giá hàm trong phương pháp chia đôi?

4.7. Trong chứng minh hội tụ của phương pháp cát tuyến, phát biểu rằng, nếu $\epsilon = \max\{\epsilon_0, \epsilon_1\} < 1$ thì bất đẳng thức

$$\epsilon_{i+1} \leq \epsilon_i \cdot \epsilon_{i-1}$$

ám chỉ

$$\epsilon_i \leq \epsilon^{\delta_i},$$

và

$$\delta_i = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^{i+1} - \left(\frac{1-\sqrt{5}}{2}\right)^{i+1}\right].$$

Hãy thiết lập điều này.

4.8. Viết mã Matlab cho phương pháp Newton.

4.9. Hàm đặc biệt

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt,$$

gọi là *hàm sai* (error function), được dùng trong lý thuyết xác suất và nhiều lãnh vực của khoa học và kỹ thuật. Vì hàm dưới dấu tích phân là dương với mọi t, hàm tăng ngặt và vì vậy có hàm ngược $x = \text{erf}^{-1}(y)$. Hàm sai ngược là một hàm quan trọng do khả năng áp dụng của nó và có thể đánh giá với y cho trước bằng cách giải phương trình $\text{erf}(x) = y$. Function `erfinv` của Matlab cho hàm sai ngược. Hãy dùng phương pháp Newton để tìm hàm sai ngược, so sánh với kết quả tính nhờ `erfinv`.

4.10. Tìm cấp hội tụ của các dãy sau:

a) $x_n = \sqrt{1/n}$; b) $x_n = \sqrt[n]{n}$; c) $x_n = \sqrt{1+1/n}$; $x_{n+1} = \text{arctg}x_n$.

4.11. Chứng tỏ hàm $F(x) = 4x(1-x)$ ánh xạ đoạn $[0,1]$ vào chính nó và không co. Chứng tỏ rằng nó có điểm bất động. Tại sao điều này không mâu thuẩn với định lý ánh xạ co?

4.12. Chứng tỏ hàm $F(x) = 2 + x - \text{arctg}x$ có tính chất $|F'(x)| < 1$ nhưng nó không có điểm bất động. Giải thích lý do điều này không mâu thuẩn với định lý ánh xạ co.

4.13. Chứng tỏ phương pháp tính $\sqrt{R}$ dưới đây hội tụ cấp ba:

$$x_{n+1} = \frac{x_n(x_n^2+3R)}{3x_n^2+R}.$$

4.14. Dùng phương pháp lặp Newton tìm một nghiệm (với sai số tuyệt đối tối đa là 10^{-4}) gần điểm $(0.5, 1.0, 0.0)$ của hệ phi tuyến

$$\begin{aligned} 2x^2 - x + y^2 - z &= 0, \\ 32x^2 - y^2 - 20z &= 0, \\ y^2 - 14xz &= 0. \end{aligned}$$

4.15. Tìm ba tham số α, β và γ trong mô hình

$$f(x) = \alpha e^{\beta x} + \gamma,$$

bằng phép nội suy ba điểm $(1, 10), (2, 12)$ và $(3, 18)$. Dùng phép lặp Newton tìm ba tham số với ba chữ số có nghĩa.

Chương 5

Tích phân số

Xấp xỉ $\int_a^b f(x)dx$ bằng số được gọi là tích phân số hay cầu phương. Hầu hết chương này liên quan đến khoảng hữu hạn $[a, b]$, nhưng có một vài bàn luận về tích phân với a và/hay b là vô hạn. Thỉnh thoảng đưa vào hàm trọng lượng $w(x) > 0$ là hữu ích và như vậy xấp xỉ tích phân dạng $\int_a^b f(x)w(x)dx$. Có nhiều lý do nghiên cứu tích phân số. Nguyên hàm (antiderivative) của f có thể không biết hay không là hàm sơ cấp. Tích phân có thể không có hiệu lực vì hàm f được xác định bởi các giá trị trong bảng hay bởi một chương trình con. Hay, các tích phân xác định phải được xấp xỉ như thành phần của sơ đồ tính toán phức tạp hơn, chẳng hạn như giải các phương trình vi phân bằng phần tử hữu hạn nhờ các phương pháp biến phân hay Galerkin.

Một nguyên lý cơ bản trong giải tích số là nếu ta không thể làm điều ta muốn với một hàm $f(x)$ cho trước, ta xấp xỉ nó bằng một hàm mà với nó ta có thể thực hiện được. Thường hàm xấp xỉ là một đa thức nội suy. Bằng cách dùng nguyên lý này ta sẽ thiết lập một vài quy tắc cầu phương và nghiên cứu sai số của chúng. Khi xấp xỉ hàm ta thấy rằng đa thức nội suy từng mảnh tiện lợi hơn đa thức nội suy, ở đây điều này cũng đúng. Cách nội suy đa thức từng mảnh là tự nhiên cho cầu phương vì dùng hàm như vậy chung qui là bẻ khoảng lấy tích phân thành các mảnh và xấp xỉ bằng đa thức trên mỗi mảnh ấy. Ý tưởng then chốt trong cầu phương là phải tính đến dáng điệu của $f(x)$ khi chia tách khoảng. Phép cầu phương "thích ứng" này được mô tả trong mục 5.2 và mã được bàn luận trong mục tiếp theo. Phép cầu phương thích ứng là vấn đề chính của chương, nhưng vài chú ý được cho cho tích phân của bảng dữ liệu và cho tích phân của các hàm hai biến. Chú ý đặc biệt được dành cho các bài toán chuẩn bị cho lời giải có hiệu quả của chúng bằng các mã của loại phát triển ở đây.

5.1 Các quy tắc cầu phương cơ bản

Để xấp xỉ

$$\int_a^b f(x)w(x)dx \tag{5.1}$$

giả sử đã biết giá trị của f tại N điểm phân biệt $x_1, x_2, \ldots, x_N$. Gọi $P_N(x)$ là đa thức nội suy f tại các điểm này. Dạng Lagrange của $P_N(x)$ dễ dàng dẫn đến xấp xỉ

$$\int_a^b f(x)w(x)dx \approx \int_a^b P_N(x)w(x)dx = \int_a^b \sum_{i=1}^{N} f(x_i)L_i(x)w(x)dx =$$

$$\sum_{i=1}^{N} f(x_i)\int_a^b L_i(x)w(x)dx = \sum_{i=1}^{N} A_i f(x_i). \tag{5.2}$$

Ở đây giả thiết các trọng lượng A_i tồn tại. Điều này tương đương với sự tồn tại các tích phân

$$\int_a^b x^j w(x)dx \quad \text{với } j = 0, 1, ..., N-1.$$

Trong trường hợp $w(x) = 1$, a và b hữu hạn, thì giả thiết này là đúng. Tuy nhiên, nếu khoảng là vô hạn (e.g., $\int_0^\infty f(x)dx$), tiếp cận trên thất bại vì không có x^j nào có tích phân trên khoảng này.

Khó khăn cơ bản của việc tiếp cận, trong trường hợp $\int_0^\infty f(x)dx$, là nó được đặt cơ sở trên sự xấp xỉ $f(x)$ bởi đa thức, mà các đa thức không có tích phân hữu hạn trên khoảng vô hạn. Vì tích phân $f(x)$ tồn tại, nó phải dần tới không thật nhanh khi $x \to \infty$. Một lời khuyên hữu ích là phải cô lập dáng điệu "khác đa thức" vào hàm trọng lượng. Chẳng hạn, nếu ta đưa vào hàm trọng lượng $w(x) = e^{-x}$ và định nghĩa $F(x) = f(x)e^x$, tích phân có thể viết lại như là $\int_0^\infty F(x)e^{-x}dx$. Không phức tạp lắm để nhận được công thức cho các tích phân dạng $\int_0^\infty F(x)e^{-x}dx$ vì các tích phân $\int_0^\infty x^j e^{-x}dx$ tồn tại với mọi j. Lời khuyên này có cho một xấp xỉ tốt $\int_0^\infty f(x)dx$ hay không là vấn đề $F(x)$ có ứng xử giống một đa thức hơn $f(x)$ hay không.

Tích phân trên khoảng vô hạn là loại bài toán chứa nhiều khó khăn. Tích phân với hàm dưới dấu tích phân có kỳ dị cũng chứa đựng khó khăn vì chúng ứng xử không giống đa thức. Thông thường việc dùng hàm trọng lượng là cách tốt để đối xử với những bài toán như vậy. Chẳng hạn, trong lời giải các bài toán thế vị phẳng bằng phương pháp phần tử biên, cần xấp xỉ các tích phân thuộc dạng

$$\int_0^1 F(x) \ln x dx$$

(và sau đó giải hệ các phương trình tuyến tính để có được nghiệm số cho phương trình tích phân của lý thuyết thế vị). Hàm $\ln x$ có thể được xem như hàm trọng lượng vì nó không dương trên khoảng $(0, 1)$ và các tích phân

$$\int_0^1 x^j \ln(x) dx$$

tồn tại với mọi j (hàm trọng lượng $w(x)$ trong (5.1) có thể lấy là $-\ln x$). Tương tự cách làm trong thí dụ lấy tích phân trên khoảng vô hạn, nếu ta muốn tính $\int_0^1 f(x)dx$ và $f(x)$ ứng xử giống $\ln x$ khi $x \to 0$, ta có thể đưa vào $\ln x$ như hàm trọng lượng và viết $F(x) = f(x)/\ln(x)$. "Ứng xử giống" khi $x \to 0$ có nghĩa là

$$\lim_{x \to 0} \frac{f(x)}{\ln(x)} = c.$$

Từ đây về sau điều này sẽ được viết là $f(x) \sim c \ln(x)$. Vì $F(x)$ có một giới hạn hữu hạn tại $x = 0$, nó được xấp xỉ bởi đa thức tốt hơn $f(x)$, mà là vô hạn ở đó.

Một công thức dạng

$$\sum_{i=1}^{N} A_i f(x_i) \tag{5.3}$$

để xấp xỉ (5.1) được gọi là công thức hay quy tắc cầu phương. Sơ đồ để phát sinh các quy tắc vừa mô tả dẫn đến các quy tắc cầu phương nội suy. Một quy

tắc như vậy sẽ tích phân chính xác đa thức bất kỳ có bậc nhỏ hơn N. Đó là vì nếu $f(x)$ là đa thức bậc nhỏ hơn N, thì bởi tính duy nhất của đa thức nội suy, $P_N(x) \equiv f(x)$, và quy tắc được xây dựng để tích phân $P_N(x)$ là chính xác.

Định nghĩa 5.1. Sai số tuyệt đối của công thức cầu phương dạng (5.3) là lượng

$$E(f) = \int_a^b f(x)w(x)dx - \sum_{i=1}^{N} A_i f(x_i). \tag{5.4}$$

Ta nói công thức cầu phương dạng (5.3) có *bậc chính xác* (degree of precision) $d \geq 0$ nếu:

(i) $E(x^j) = 0$, $j = 0, 1, \ldots, d$, và

(ii) $E(x^{d+1}) \neq 0$.

Sau này ta sẽ thấy, một sự chọn lựa đúng đắn các điểm nội suy x_i khi xây dựng (5.2) dẫn đến công thức với bậc chính xác lớn hơn $N - 1$. Thường thì các x_i thuộc $[a, b]$, nhưng thực ra không nhất thiết phải như vậy. Chẳng hạn, công thức Adams cho lời giải phương trình vi phân dựa trên quy tắc cầu phương dùng các điểm nút, ngoại trừ hai điểm cuối a và b, nằm bên ngoài khoảng này. Điều này cũng đúng với phương pháp tích phân theo bảng dữ liệu sẽ được đề cập đến sau này.

Định lý dưới đây phát triển một vài chặn trên sai số của công thức với bậc chính xác d. Nó được phát biểu bằng cách dùng ký hiệu $\|f\|$ cho maximum trên $[a, b]$ của $|f(x)|$. Cũng vậy, như trong chương 3, M_q được dùng để chỉ $\|f^{(q)}\|$.

Định lý 5.1. *Nếu công thức cầu phương (5.2) có bậc chính xác d, thì với bất kỳ đa thức $p(x)$ bậc $q \leq d$,*

$$|E(f)| \leq \|f - p\| \left(\int_a^b w(x)dx + \sum_{i=1}^{N} |A_i| \right). \tag{5.5}$$

Nếu mọi $A_i > 0$, thì

$$|E(f)| \leq 2\|f - p\| \int_a^b w(x)dx. \tag{5.6}$$

Chứng minh. Với $p(x)$ là đa thức bất kỳ bậc $q \le d$,

$$\begin{aligned} |E(f)| &\le \left| \int_a^b p(x)w(x)dx + \int_a^b (f(x)-p(x))w(x)dx \right. \\ &\qquad \left. - \sum_{i=1}^N A_i p(x_i) - \sum_{i=1}^N A_i (f(x_i)-p(x_i)) \right| \\ &\le |E(p)| + \int_a^b |f(x)-p(x)|w(x)dx + \sum_{i=1}^N |A_i||f(x_i)-p(x_i)| \\ &\le \|f-p\| \left(\int_a^b w(x)dx + \sum_{i=1}^N |A_i| \right), \end{aligned}$$

trong đó ta đã dùng $E(p)=0$. Đây là (5.5). Khi mọi $A_i > 0$ thì dấu trị tuyệt đối trong (5.5) có thể bỏ. Vì công thức cầu phương chính xác khi $f(x) \equiv 1$ nên

$$\sum_{i=1}^N A_i \cdot 1 = \int_a^b w(x) \cdot 1 dx,$$

và ta có (5.6). ■

Hệ qủa 1. *Nếu $f(x)$ có $d+1$ đạo hàm liên tục trên $[a,b]$, thì*

$$|E(f)| \le \left(\frac{b-a}{2}\right)^{d+1} \frac{M_{d+1}}{(d+1)!} \left(\int_a^b w(x)dx + \sum_{i=1}^N |A_i| \right). \tag{5.7}$$

Nếu mọi $A_i > 0$, thì

$$|E(f)| \le \left(\frac{b-a}{2}\right)^{d+1} \frac{M_{d+1}}{(d+1)!} 2 \int_a^b w(x)dx. \tag{5.8}$$

Chứng minh. Vì chặn của định lý 5.1 đúng với mọi đa thức $p(x)$ có bậc $q \le d$, ta có thể dùng $p(x)$ là đa thức Taylor khai triển $f(x)$ tại $x_0 = (a+b)/2$, $n = q$:

$$p(x) = f(x_0) + \frac{x - x_0}{1} f'(x_0) + \cdots + \frac{(x - x_0)^q}{q!} f^{(q)}(x_0)$$

và

$$R_{q+1}(x) = \frac{(x - x_0)^{q+1}}{(q+1)!} f^{(q+1)}(z)$$

với z nằm giữa x_0 và x. Điều này ám chỉ

$$\|f - p\| = \max_{a \le x \le b} \left| \frac{(x - x_0)^{q+1}}{(q+1)!} f^{(q+1)}(z) \right| \le \left(\frac{b-a}{2} \right)^{q+1} \frac{M_{q+1}}{(q+1)!}. \qquad (5.9)$$

Thay điều này với $q = d$ vào (5.5) hay (5.6) ta nhận được (5.7) hay (5.8). ■

Nhận xét 5.1. Khi nghiên cứu nội suy đa thức, ta đã biết các nội suy bậc cao có khả năng dao động và gây ra những thay đổi không phù hợp. Tình hình bây giờ thì khác vì nó là xấp xỉ diện tích bên dưới đường cong và dường như các dao động sẽ bị trung bình hóa. Điều này quan trọng đối với trường hợp đặc biệt của công thức khi mà chặn sai số của định lý 5.1 là đúng, và tất cả $A_i > 0$. Đáng tiếc, các công thức cầu phương nội suy dựa trên $\{x_i\}$ cách đều trong $[a, b]$, gọi là các công thức cầu phương *Newton - Cotes*, có một vài A_i lấy giá trị âm ngay cả với các bậc chính xác vừa phải. Kết quả của các công thức này có thể không hội tụ tới giá trị của tích phân khi bậc gia tăng.

Nhận xét 5.2. Trong các chặn (5.5), (5.6) ta có thể lấy đa thức $p(x)$ bất kỳ với bậc $q \le d$. Với a, b hữu hạn, tồn tại đa thức $p^*(x)$ với bậc $\le d$ gần f nhất theo nghĩa

$$\|f - p^*\| = \min_{\deg p \le q} \|f - p\|.$$

Khi đó,

$$|E(f)| \le \|f - p^*\| \left(\int_a^b w(x)dx + \sum_{i=1}^{N} |A_i| \right). \qquad (5.10)$$

Nếu mọi $A_i > 0$, thì

$$|E(f)| \leq 2\|f - p^*\| \int_a^b w(x)dx. \tag{5.11}$$

Các bất đẳng thức (5.10), (5.11) cho chặn sai số dựa trên xấp xỉ tốt nhất của $f(x)$; chúng hữu ích khi hàm lấy tích phân không đủ trơn.

Một phân tích sai số chi tiết hơn chứng tỏ rằng sai số $E(f)$ có thể được biểu diễn như là

$$E(f) = c\left(\frac{b-a}{2}\right)^{d+2} f^{(d+1)}(\xi) \tag{5.12}$$

với $c \in \mathbb{R}$ và $\xi \in (a, b)$. Nếu một công thức cầu phương có bậc chính xác d, thì

$$E(x^j) = 0, \quad j = 0, 1, \ldots, d \tag{5.13}$$
$$E(x^{d+1}) \neq 0. \tag{5.14}$$

Nếu ta giả sử rằng sai số có dạng (5.12), dễ dàng tìm c từ

$$E(x^{d+1}) = c\left(\frac{b-a}{2}\right)^{d+2} (d+1)!. \tag{5.15}$$

Các phương trình (5.13), (5.14) cung cấp cách khác để sinh ra các quy tắc cầu phương. Cách tiếp cận được biết như là *phương pháp hệ số bất định*. Trong cách tiếp cận này các hệ số A_i được xem như các ẩn được tìm bằng sự thỏa mãn hệ phương trình tuyến tính (5.13 với d lớn có thể. Trước khi cho các thí dụ, ta chú ý rằng, nên áp dụng phương pháp hệ số bất định cho khoảng chuẩn $[-1, 1]$ và rồi biến đổi thành khoảng tổng quát $[a, b]$ bằng một phép đổi biến đơn giản. Nếu ta có

$$\int_{-1}^{1} f(x)dx = \sum_{i=1}^{N} A_i f(x_i) + cf^{(d+1)}(\xi),$$

đặt

$$t = \frac{b-a}{2}x + \frac{a+b}{2}.$$

Thì $dt = (b-a)dx/2$ và

$$\begin{aligned}\int_a^b f(t)dt &= \frac{b-a}{2}\int_{-1}^{1} f\left(\frac{b-a}{2}(x+1)+a\right)dx \\ &= \frac{b-a}{2}\sum_{i=1}^{N} A_i f\left(\frac{b-a}{2}x_i + \frac{a+b}{2}\right) + \frac{b-a}{2}E(f).\end{aligned}$$

Vì

$$\frac{d}{dx} = \frac{dt}{dx}\frac{d}{dt} = \frac{b-a}{2}\frac{d}{dt} \Rightarrow \frac{d^{d+1}}{dx^{d+1}} = \left(\frac{b-a}{2}\right)^{d+1}\frac{d^{d+1}}{dt^{d+1}},$$

nên phép đổi biến cho

$$\int_a^b f(t)dt = \sum_{i=1}^{N}\left(\frac{b-a}{2}A_i\right) f\left(\frac{b-a}{2}x_i + \frac{a+b}{2}\right) + \left(\frac{b-a}{2}\right)^{d+2} f^{(d+1)}(\xi).$$

Thí dụ 5.1. Tìm công thức cầu phương dạng

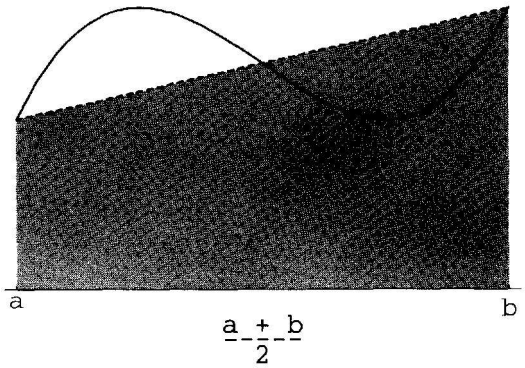
$$\int_{-1}^{1} f(x)dx = A_1 f(-1) + A_2 f(1) + E(f).$$

Theo phương pháp hệ số bất định

$$\begin{aligned} f(x) = 1 &\Rightarrow 2 = A_1 + A_2, \\ f(x) = x &\Rightarrow 0 = -A_1 + A_2. \end{aligned}$$

Suy ra: $A_1 = A_2 = 1$. Ta cũng thấy rằng, bằng cách xây dựng, $d \geq 1$. Thì $f(x) = x^2$ nhận được

$$\frac{2}{3} = A_1 + A_2 + E(x^2) \Rightarrow E(x^2) = -\frac{4}{3}.$$



Hình 5.1: Quy tắc hình thang.

Vì $E(x^2) \neq 0$ điều này nói rằng $d = 1$ và $c = E(x^2)/2! = -2/3$, nghĩa là

$$\int_{-1}^{1} f(x)dx = f(-1) + f(1) - \frac{2}{3} f''(\xi)$$

với $\xi \in (-1, 1)$.

Với khoảng $[a, b]$ tổng quát, áp dụng công thức đổi biến ta có (công thức trong dấu [ ])

Quy tắc hình thang (trapezoid rule)

$$\int_a^b f(x)dx = \frac{b-a}{2}(f(a) + f(b)) - \frac{(b-a)^3}{12} f''(\xi), \tag{5.16}$$

trong đó $\xi \in (a, b)$. ∘

Thí dụ 5.2. Tìm công thức chính xác nhất dạng

$$\int_{-1}^{1} f(x)dx = A_1 f(-1) + A_2 f(0) + A_3 f(1) + E(f).$$

Theo phương pháp hệ số bất định

$$\begin{aligned} f(x) = 1 &\Rightarrow 2 = A_1 + A_2 + A_3, \\ f(x) = x &\Rightarrow 0 = -A_1 + A_3, \\ f(x) = x^2 &\Rightarrow 2/3 = A_1 + A_3, \end{aligned}$$

Suy ra: $A_1 = A_3 = 1/3$, $A_2 = 4/3$.

Để tìm bậc chính xác ta kiểm $E(f) \neq 0$ với $f(x)$ là đa thức bậc cao hơn 2. Nếu $f(x) = x^3$ ta được

$$0 = -A_1 + A_3 + E(x^3) \Rightarrow E(x^3) = 0;$$

nghĩa là, quy tắc có bậc chính xác lớn hơn 2. Nếu lấy $f(x) = x^4$ ta được

$$\frac{2}{5} = A_1 + A_3 + E(x^4) \Rightarrow E(x^4) = -\frac{4}{15} \Rightarrow c = -\frac{1}{90}.$$

Như vậy, bậc chính xác $d = 3$, và

$$\int_{-1}^{1} f(x)dx = \frac{1}{3}f(-1) + \frac{4}{3}f(0) + \frac{1}{3}f(1) - \frac{2}{9}f^{(4)}(\xi)$$

với $\xi \in (-1, 1)$. Cũng dùng công thức đổi biến ta suy ra công thức tổng quát.

Quy tắc Simpson

$$\int_a^b f(x)dx = \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right] - \frac{(b-a)^5}{2880}f^{(4)}(\xi), \quad (5.17)$$

trong đó $\xi \in (a, b)$. ∘

Cả hai công thức thuộc lớp *công thức Newton-Cotes* vì các nút cách đều trong $[a, b]$. Thủ tục thiết lập công thức bao gồm việc chọn trước các nút x_i và rồi giải hệ phương trình tuyến tính xác định các trọng lượng A_i. Nhưng nếu các x_i chưa biết (được phép chọn)? Số ẩn cần tìm sẽ gấp đôi, $2N$, ở cách sắp xếp của ta, có thể hy vọng tìm các công thức với bậc chính xác cao hơn nhiều, như sẽ thấy dưới đây, có công thức với bậc chính xác $2N - 1$ mà dùng chỉ N giá trị của f. Đáng tiếc, hệ phương trình cho A_i và x_i là phi tuyến. Không hiển nhiên hệ đó có nghiệm thực, mà nếu có, làm thế nào để nhận được chúng. Gauss đã giải quyết vấn đề một cách "thanh lịch" với N tổng quát, ngay cả với các hàm trọng lượng tổng quát hơn và các khoảng là vô hạn. Kết quả được biết như là *công thức cầu phương Gauss*. Một số trường hợp đặc biệt có thể chỉ ra theo cách sơ cấp.

Thí dụ 5.3. Cho $N = 1$ công thức Gauss có dạng

$$\int_{-1}^{1} f(x)dx = A_1 f(x_1) + E(f).$$

Bằng phương pháp hệ số bất định

$$\begin{aligned} f(x) = 1 &\Rightarrow 2 = A_1, \\ f(x) = 0 &\Rightarrow 0 = A_1 x_1, \end{aligned}$$

suy ra $A_1 = 2$ và $x_1 = 0$. Để xác định sai số, ta thử

$$f(x) = x^2 \Rightarrow \frac{2}{3} = 2 \times 0 + E(x^2),$$

và thấy $d = 1$, $c = 1/3$, và

$$\int_{-1}^{1} f(x)dx = 2f(0) + \frac{1}{3} f''(\xi).$$

Trên $[a, b]$ công thức này trở thành

$$\int_a^b f(x)dx = \left[(b-a) f\left(\frac{a+b}{2}\right)\right] + \frac{(b-a)^3}{24} f''(\xi). \tag{5.18}$$

Công thức này được biết như là quy tắc điểm giữa ∘

Thí dụ 5.4. Cho $N = 3$ công thức Gauss có dạng

$$\int_{-1}^{1} f(x)dx = A_1 f(x_1) + A_2 f(x_2) + A_3 f(x_3) + E(f).$$

Do tính đối xứng của khoảng $[-1, 1]$, có thể cho rằng $A_1 = A_3$, $x_2 = 0$, và $x_1 = -x_3$, nghĩa là

$$\int_{-1}^{1} f(x)dx = A_1 f(x_1) + A_2 f(0) + A_1 f(-x_1) + E(f).$$

Bằng phương pháp hệ số bất định,

$$\begin{aligned}
f(x) = 1 &\Rightarrow 2 = 2A_1 + A_2,\\
f(x) = x &\Rightarrow 0 = A_1x_1 + A_1(-x_1) \quad \text{(tự động thỏa)},\\
f(x) = x^2 &\Rightarrow 2/3 = 2A_1x_1^2,\\
f(x) = x^3 &\Rightarrow 0 = A_1x_1^3 + A_1(-x_1^3) \quad \text{(tự động thỏa)},\\
f(x) = x^4 &\Rightarrow 2/5 = 2A_1x_1^5.
\end{aligned}$$

Giải hệ này, ta được: $A_1 = 5/9$, $A_2 = 8/9$, $x_1 = -\sqrt{3/5} = -x_3$.

Để tìm sai số, thử

$$f(x) = x^5 \Rightarrow 0 = A_1x_1^5 + A_1(-x_1^5) + E(x^5) \Rightarrow E(x^5) = 0!$$

Như vậy, bậc chính xác lớn hơn 4. Tiếp tục với $f(x) = x^6$, ta có:

$$2/7 = 2A_1x_1^6 + E(x^6) = \frac{6}{25} + E(x^6),$$

suy ra: $d = 5$, $c = 1/15750$, và

$$\int_{-1}^{1} f(x)dx = \frac{1}{9}\left[5f\left(-\sqrt{\frac{3}{5}}\right) + 8f(0) + 5f\left(\sqrt{\frac{3}{5}}\right)\right] + \frac{1}{15750}f^{(6)}(\xi).$$

Trên $[a, b]$, ký hiệu $x_0 = (a + b)/2$, $h = (b - a)/2$, kết quả là *công thức cầu phương Gauss 3-điểm*

$$\int_a^b f(x)dx = \frac{h}{9}\left[5f\left(x_0 - h\sqrt{\frac{3}{5}}\right) + 8f(x_0) + 5f\left(x_0 + h\sqrt{\frac{3}{5}}\right)\right] + \frac{h^7}{15750}f^{(6)}(\xi) \quad (5.19)$$

∘

Với N lớn phương pháp hệ số bất định để thiết lập quy tắc cầu phương Gauss là không thực tế. Bên cạnh đó, vấn đề tồn tại công thức và bậc chính xác tốt nhất có thể là câu hỏi còn để mở trong cách tiếp cận này. Gauss đã dùng lý thuyết các đa thức trực giao để trả lời. Lời giải của Gauss không được trình bày ở đây, nhưng ta có thể xem bậc chính xác của công thức cao như

thế nào. Với các điều kiện phù hợp trên $w(x)$ và $[a,b]$, ta biết rằng tồn tại một dãy các đa thức $\theta_{N+1}(x)$, $N = 0, 1, \ldots$ sao cho $\theta_{N+1}(x)$ là bậc N và

$$\int_a^b x^j \theta_{N+1}(x) w(x) dx = 0 \quad \text{khi} \quad j < N. \tag{5.20}$$

Khi $w(x) = 1$, $a = -1$, $b = 1$, các đa thức này là *đa thức Legendre*. Ta cũng biết rằng N nghiệm phân biệt của $\theta_{N+1}(x)$ là thực và nằm trong khoảng (a,b). Giả sử rằng công thức cầu phương nội suy (5.2) dựa trên cơ sở nội suy tại các nghiệm của $\theta_{N+1}(x)$. Nếu $f(x)$ là đa thức bậc $2N-1$, nó có thể được viết

$$f(x) = q(x)\theta_{N+1}(x) + r(x),$$

trong đó đa thức thương $q(x)$ và đa thức dư $r(x)$ có bậc tối đa $N-1$. Thì

$$\int_a^b f(x)w(x)dx = \int_a^b q(x)\theta_{N+1}(x)w(x)dx + \int_a^b r(x)w(x)dx = \int_a^b r(x)w(x)dx,$$

trong đó số hạng đầu triệt tiêu do (5.20). Với cách chọn bất kỳ các nút x_i, công thức (5.2) tích phân đa thức bậc N cách chính xác, vậy

$$\int_a^b r(x)w(x)dx = \sum_{i=1}^N A_i r(x_i). \tag{5.21}$$

Công thức áp dụng cho $f(x)$ có dạng

$$\sum_{i=1}^N A_i f(x_i) = \sum_{i=1}^N A_i q(x_i)\theta_{N+1}(x_i) + \sum_{i=1}^N A_i r(x_i).$$

Bây giờ ta dùng sự kiện x_i là các nghiệm của $\theta_{N+1}(x)$ dễ thấy

$$\int_a^b f(x)w(x)dx = \sum_{i=1}^N A_i f(x_i) = \sum_{i=1}^N A_i r(x_i) = \int_a^b r(x)w(x)dx.$$

Dấu bằng cuối cùng là do (5.21). Vì đa thức bất kỳ $f(x)$ có bậc $2N-1$ được tích phân chính xác, nên công thức này có bậc chính xác ít nhất là $2N-1$.

Có nhiều cách thuận tiện về phương diện tính toán để thiết lập các công thức cầu phương Gauss, và các công thức có thể tìm thấy trong các sách chuyên khảo. Các công thức Gauss có giá trị vì chúng cung cấp bậc chính xác cao nhất với số các giá trị $f(x)$. Một sự kiện quan trọng về công thức Gauss là tất cả các A_i đều dương. Như đã bàn trong phần chặn sai số, điều này có nghĩa là ta có thể dùng công thức với bậc chính xác cao, ngay cả khi hàm dưới dấu tích phân không trơn. Công thức Gauss kết hợp chặt chẽ với các hàm trọng lượng là công cụ đặc biệt quan trọng khi đối xử với các tích phân mà hàm dưới dấu tích phân có kỳ dị hoặc các khoảng lấy tích phân vô hạn. Dù có hàm trọng lượng hay không, tất cả các nút của công thức Gauss đều nằm trong khoảng mở (a,b) (không dùng đến $f(a)$ và $f(b)$). Điều này rất hữu ích khi đối xử với tích phân mà hàm dưới dấu tích phân kỳ dị.

5.2 Quy tắc cầu phương đa hợp

Cho đến nay ta chỉ xét các thủ tục dựa trên xấp xỉ hàm $f(x)$ trên toàn bộ khoảng $[a,b]$. Cũng như nội suy đa thức, sai số phụ thuộc mạnh vào chiều dài của khoảng. Điều này đề nghị ta phân hoạch khoảng và xấp xỉ hàm bằng hàm đa thức từng mảnh. Cách tiếp cận đơn giản nhất là chia khoảng thành những khoảng con chỉ định trước. Nếu ta phân hoạch $[a,b]$ thành $a = x_1 < x_2 < \ldots < x_{n+1} = b$, thì

$$\int_a^b f(x)dx = \sum_{i=1}^{n} \int_{x_i}^{x_{i+1}} f(x)dx,$$

ta có thể áp dụng các quy tắc cầu phương chuẩn cho n tích phân ở vế phải. Kết quả được biết như là *quy tắc đa hợp* (composite rule) hay *quy tắc ghép* (compound rule). Người ta thường dùng phân hoạch đều khoảng $[a,b]$ và dùng cùng công thức cầu phương trên mỗi khoảng con, nhưng điều này là không nhất thiết.

Thí dụ 5.5 (Quy tắc hình thang đa hợp)**.** Quy tắc hình thang đa hợp xấp xỉ $I = \int_a^b f(x)dx$ bằng cách phân hoạch $[a,b]$ thành n khoảng con độ dài $h = (b-a)/n$ và áp dụng công thức cầu phương hình thang cho mỗi khoảng.

Với định nghĩa $x_i = a + ih$.

$$I \approx T_n = \frac{h}{2}[f(x_0) + f(x_1)] + \frac{h}{2}[f(x_1) + f(x_2)] + \ldots + \frac{h}{2}[f(x_n) + f(x_{n+1})],$$

thu gọn, ta được:

$$T_n = h\left[\frac{1}{2}f(x_0) + f(x_1) + f(x_2) + \ldots + f(x_n) + \frac{1}{2}f(x_{n+1})\right] \circ$$

Theo *công thức tổng Euler-Maclaurin*, nếu $f^{(2v)}(x)$ liên tục trên $[a, b]$, thì tồn tại $\xi \in (a, b)$ sao cho

$$I = T_n + \underbrace{\sum_{k=1}^{v-1} \frac{-h^{2k}}{(2k)!} B_{2k}[f^{(2k-1)}(b) - f^{(2k-1)}(a)] - \frac{nh^{2v+1}}{(2k)!} B_{2v} f^{(2v)}(\xi)}_{\text{sai số}}.$$

Các hệ số B_{2k} xuất hiện ở đây được biết như là các *số Bernoulli*. Vài số hạng đầu của khai triển sai số là

$$I = T_n - \frac{h^2}{12}[f'(b) - f'(a)] + \frac{h^4}{720}[f^{(3)}(b) - f^{(3)}(a)] - \cdots$$

Quy tắc hình thang áp dụng cho một khoảng độ dài h có sai số dần về không như h^3. Khi $n = (b-a)/h$ từ được tổ hợp, sai số của xấp xỉ tích phân dần về không như h^2. Tuy nhiên, nếu xảy ra $f'(b) = f'(a)$, công thức sẽ chính xác hơn bình thường. Nếu thêm vào các đạo hàm khác tại các điểm cuối của khoảng tích phân bằng nhau, thì công thức còn chính xác hơn nữa. Khi tích phân hàm tuần hoàn trên một bội của chu kỳ, tất cả đạo hàm tại các điểm cuối của khoảng lấy tích phân là bằng nhau và công thức này cực kỳ chính xác. Thực ra, nếu hàm tuần hoàn là giải tích, nó có đạo hàm mọi cấp, thì $T_n \to I$ nhanh hơn bất kỳ lũy thừa nào của h! Mặc dù khá đặc biệt, nhưng điều này là cực kỳ quan trọng trong giải tích Fourier.

Sai số của T_n có thể được đánh giá bằng cách so sánh nó với kết quả chính xác hơn, T_{2n}, nhận được bằng cách chia đôi mỗi khoảng con. Một cách

thuận tiện để đánh giá công thức là

$$T_{2n} = \frac{h}{2}\left[\frac{1}{2}f(x_0) + f(x_{1/2} + f(x_1) + \ldots + f(x_{n-1}) + f(x_{n-1/2}) + \frac{1}{2}f(x_n)\right] = \frac{1}{2}(T_n + M_n),$$

trong đó

$$M_n = h\sum_{k=1}^{n} f(a + (k - 1/2)h).$$

Chú ý rằng tất cả các đánh giá của f thực hiện trong T_n đều được dùng lại trong T_{2n}.

Có một cách khai thác khai triển sai số của quy tắc hình thang đa hợp do Romberg tìm ra rất phổ biến cho hàm dưới dấu tích phân tổng quát. Ý tưởng là tổ hợp T_n và T_{2n} để nhận được kết quả có bậc chính xác cao hơn. Từ các khai triển sai số

$$\begin{aligned} I &= T_n - \frac{h^2}{12}[f'(b) - f'(a)] + \frac{h^4}{720}[f^{(3)}(b) - f^{(3)}(a)] - \cdots \\ &= T_{2n} - \frac{(h/2)^2}{12}[f'(b) - f'(a)] + \frac{(h/2)^4}{720}[f^{(3)}(b) - f^{(3)}(a)] - \cdots \end{aligned}$$

ta suy ra

$$I = \frac{2^2 T_{2n} - T_n}{2^2 - 1} + \left(\frac{2^2 - 2^4}{2^2 - 1}\right)\frac{(h/2)^4}{720}[f^{(3)}(b) - f^{(3)}(a)] - \cdots$$

Công thức

$$I \approx \frac{2^2 T_{2n} - T_n}{2^2 - 1}$$

chính xác cao hơn mỗi công thức thành phần. Thủ tục tổ hợp trình bày trên được gọi là *phép ngoại suy Romberg*.

Tích phân Romberg thường rất hiệu quả. Nó thích ứng bậc của phương pháp với bài toán. Tuy nhiên, kết quả phụ thuộc vào tính trơn của hàm dưới

dấu tích phân. Cũng vậy, nó đánh giá $f(x)$ tại các điểm cuối của khoảng, mà điều này đôi lúc gây bất tiện. Nếu có kỳ dị tại điểm cuối của khoảng hoặc quá trình không hội tụ thì nên dùng quy tắc điểm giữa cho khoảng chứa điểm cuối này và nên chia nhỏ khoảng này thành 2 hay 3 khoảng con.

5.3 Cầu phương thích ứng

Mục này trình bày ý tưởng cơ bản - cầu phương thích ứng - cho các chương trình tính xấp xỉ tích phân đạt độ chính xác theo yêu cầu người dùng. Điều này được thực hiện bằng cách chia nhỏ khoảng $[a, b]$ thành các khoảng con và áp dụng công thức cầu phương cơ bản cho mỗi khoảng. Khoảng được phân hoạch theo cách thích ứng với dáng điệu của hàm $f(x)$, dựa trên kết quả việc đánh giá sai số. Nếu sai số không chấp nhận được, việc chia nhỏ khoảng con sẽ được tiếp tục. Như ta đã thấy, ngay cả với công thức cầu phương có bậc chính xác vừa phải, sự giảm thiểu độ dài của khoảng, về cơ bản, làm tăng sự chính xác của xấp xỉ. Tiến hành theo cách này, công thức được áp dụng trên toàn bộ các khoảng con trong đó $f(x)$ được xấp xỉ tốt hơn. Ý tưởng được trình bày ở đây là cơ sở các chương trình con của một số thư viện chương trình như QUADPACK, NAG và IMSL; thậm chí trong các môi trường tính toán như Matlab.

Trong các chương trình "cầu phương thích ứng", khi mã nhận được các dung sai[1] (tolerance) của sai số tuyệt đối `abs_err` và sai số tương đối `rel_err`, nó sẽ cố gắng tính giá trị `ans` sao cho

$$|I - \texttt{ans}| \leq \max\{\texttt{abs\_err}, \texttt{rel\_err} \times |I|\}. \tag{5.22}$$

Khi cài đặt (5.22) ta dùng đánh giá sai số

$$\texttt{err} \approx I - \texttt{ans}$$

và thay I bằng `ans` trong (5.22)

$$|\texttt{err}| \leq \max\{\texttt{abs\_err}, \texttt{rel\_err} \times |\texttt{ans}|\}. \tag{5.23}$$

Lưu ý, ta không biết giá trị đúng của I.

Không thể có được xấp xỉ tích phân chính xác hơn giá trị đúng của nó được làm tròn, vì vậy lấy $\texttt{rel\_err} < u$ (u là đơn vị làm tròn) là vô nghĩa.

[1]Thuật ngữ dung sai dùng ở đây được hiểu theo nghĩa kỹ thuật, nó là độ lớn chấp nhận được của sai số.

Thông thường người ta lấy rel_err $> 10u$. Cũng vậy, abs_err > 0 để đối xử với trường hợp $I = 0$.

Phương pháp thích ứng dùng trong mã chia khoảng $[a, b]$ thành các khoảng con $[\alpha, \beta]$, trên đó quy tắc cầu phương cơ bản được dùng, đạt độ chính xác yêu cầu. Để quyết định xem kết quả tính toán đã đủ chính xác chưa ta phải đánh giá sai số của công thức. Điều này được thực hiện dựa trên nguyên lý cơ bản của giải tích số, ấy là đánh giá sai số của kết quả bằng cách so sánh nó với kết quả chính xác hơn. Gọi Q là kết quả tính toán và $\hat{Q}$ là kết quả chính xác hơn, ta dùng đánh giá sai số:

$$\texttt{err} = \hat{Q} - Q,$$

nghĩa là, thay I bằng $\hat{Q}$. Thí dụ, Q được tính bằng quy tắc hình thang (theo $f(\alpha)$ và $f(\beta)$), $\hat{Q}$ được tính bằng quy tắc Simpson (theo $f(\alpha)$, $f(\beta)$ và $f((\alpha + \beta)/2)$), được biết là chính xác hơn. Cũng có thể lấy $\hat{Q}$ như là kết quả tính bằng cùng phương pháp (hình thang) nhưng khoảng $[\alpha, \beta]$ được chia đôi. Một lưu ý khi thiết kế chương trình là yêu cầu về tính kinh tế. Nên chọn phương pháp tính $\hat{Q}$ sao cho nó thừa hưởng các đánh giá $f(x)$ đã có khi tính Q. Như trong thí dụ vừa nêu, để tính $\hat{Q}$ ta chỉ cần đánh giá thêm $f((\alpha + \beta)/2)$, các đánh giá $f(\alpha)$, $f(\beta)$ được thừa hưởng từ quá trình tính Q.

5.4 Các chương trình con

Trong mục này ta làm quen với một số chương trình con dạng function tính tích phân số. function smpsns là chương trình con tính tích phân bằng phương pháp Simpson (**simpson**'s method) với N đoạn. Hàm $f(x)$ có thể có kỳ dị tại a và/hay b nhưng tích phân hội tụ. Thí dụ $|f(x)| \to \infty$ khi $x \to a$, cận dưới sẽ được "dời lên" $a(1 + \texttt{EPS})$ hoặc $a + \texttt{EPS}$ (để tránh trường hợp $a \times \texttt{EPS}$ underflow). Cũng để tránh trường hợp $f(x_i)$ overflow, ta đặt $f(x_i)$ bằng $\pm$realmax nếu $f(x_i)$ bằng $\pm$inf.

```
function INTf=smpsns(f,a,b,N,varargin)
% tich phan cua f(x) tren [a,b] bang quy tac Simpson voi N doan
EPS=1e-12;
if nargin<4, N=100; end
if abs(b-a)<1e-12|N<=0, INTf=0; return; end
if mod(N,2)~=0, N=N+1; end % lam cho N chan
fa=feval(f,a,varargin{:});
if isnan(fa)|abs(fa)==inf, a=a+max(abs(a)*EPS,EPS); end
fb=feval(f,b,varargin{:});
```

```
if isnan(fb)|abs(fb)==inf, b=b-max(abs(b)*EPS,EPS); end
h=(b-a)/N;  x=a+[0:N]*h; % cac nut
kodd=2:2:N; keven=3:2:N-1; % tap cac chi so le/chan
fx=feval(f,x,varargin{:});
fx(find(fx==inf))=realmax; fx(find(fx==-inf))=-realmax;
INTf= h/3*(fx(1)+fx(N+1)+4*sum(fx(kodd))+2*sum(fx(keven)));
```

Chú giải

⋆ **isnan** đúng khi "Not-a-Number". `isnan(x)` trả về 1 khi `x` là NaN, 0 khi ngược lại.

⋆ **feval** - đánh giá hàm chỉ định. `feval(f,x1,...,xn)` đánh giá hàm, chỉ định bởi tên hàm `f`, tại `x1,...,xn`.

⋆ **realmax** - số dấu chấm động dương lớn nhất.

⋆ **find** - tìm của các phần tử khác không. `find(x)` trả về các chỉ số tương ứng với các phần tử khác không của mảng `x`. Lưu ý `x` có thể là biểu thức logic.

Hai function `asmpsn` chương trình tính tích phân bằng phương pháp Simpson thích ứng (**a**dapted **s**i**mpso**n method). Với function này ta có thể tính tích phân với sai số tối đa do đối số `tol` chỉ định. Đầu tiên, hàm $f(x)$ được tính bằng phương pháp Simpson 1 đoạn (bằng cách gọi function `smpsns`). Kết quả trả về được hiệu chỉnh dần. Sau khi tính toán, kết quả trả về bao gồm `INTf` - tích phân của hàm $f(x)$, `points` - vectơ chứa các điểm nút và `err` - sai số.

```
function  [INTf,points,err]=asmpsn(f,a,b,tol,varargin)
% ap dung quy tac Simpson thich ung
INTf=smpsns(f,a,b,1,varargin:);
points=[a (a+b)/2 b];
err=10;
notdone=true;
while notdone
    for k=1:length(points)-1
        SUBINTf(k)=smpsns(f,points(k),points(k+1),1,varargin:);
    end
    INTfnew=sum(SUBINTf);
    err=abs(INTfnew-INTf);
    INTf= INTfnew;
    if isnan(err)|err<tol|tol<eps
        notdone=false;
    else
        for k=1:length(points)-1
            points=[points,points(1),(points(1)+points(2))/2];
```

```
            points(1)=[];
        end
        points=[points,points(1)];
        points(1)=[];
    end
end
```

Bây giờ ta dùng các function trên để tính tích phân $\int_0^1 x\sin(10x)dx$. Dùng phương pháp Simpson 5 đoạn:

```
>> f = @(x) x.*sin(10.*x);
>> smpsns(f,a,b,5)
```

Kết quả trả về

```
ans =
    0.0851
```

Dùng phương pháp Simpson 20 đoạn:

```
>> f = @(x) x.*sin(10.*x);
>> smpsns(f,a,b,20)
```

Kết quả trả về

```
ans =
    0.0785
```

Nếu dùng phương pháp Simpson thích ứng với `tol`= 10^{-6}:

```
>> [INTf,points,err]=asmpsn(f,0,1,10^-6);
>> INTf
INTf =
    0.0785
>> length(points)
ans =
    65
```

5.5 Một số vấn đề thực hành

Các công thức cầu phương xấp xỉ tích phân dùng một số hữu hạn các điểm. Nếu giá trị hàm tại các điểm này không biểu diễn tốt hàm thì kết quả tính

toán có thể không chính xác mặc dù sai số đánh giá là chấp nhận được. Nguyên nhân là tích phân xấp xỉ và sai số đánh giá dựa trên giả thiết hàm dưới dấu tích phân là trơn giữa các điểm.

Chương trình cầu phương thích ứng cho kết quả tốt nếu nhận dạng được dáng điệu của hàm dưới dấu tích phân. Nhưng nếu $f(x)$ có điểm nhọn hoặc dao động nhiều trong khoảng lấy tích phân thì kết quả không lường trước được. Đôi khi ta phải cắt khoảng lấy tính phân thành những khoảng phù hợp để các điểm đánh giá (phát sinh trong mã cầu phương thích ứng) nằm trong những vùng có "vấn đề". Xem thí dụ sau.

Họ các tích phân

$$I_n = \int_0^{\pi} \sin^{2n} x dx.$$

có thể tính bằng công thức truy hồi

$$I_n = \frac{2n-1}{2n} I_{n-1}, \quad I_0 = \pi.$$

Khi n lớn hàm dưới dấu tích phân có một định nhọn tạo điểm giữa của khoảng.

Dùng function `asmpsn` với $n = 200$, `tol`$= 10^{-6}$, kết quả trả về

```
INTf = 0.12525310615320509044501307016617
err = 7.8051e-011
```

với 129 điểm đánh giá.

Nếu so với giá trị đúng: `0.12525310615320497863722064113372`, ta thấy kết quả cho rất chính xác. Mã `asmpsn` không gặp khó khăn vì đỉnh nhọn là điểm đánh giá. Nhưng nếu khoảng lấy tích phân được tách thành $[0, 2.6]$ và $[2.6, \pi]$. Khi đó đỉnh nhọn sẽ không rơi vào các điểm đánh giá. Tích phân trên hai đoạn này, dùng `asmpsn`, rồi cộng lại, ta được kết quả là `1.5417e-007`, tổng cộng chỉ có 6 điểm đánh giá. Kết quả sai biệt rất nhiều so với giá trị đúng!

Như vậy, với một chút khảo sát toán học hàm dưới dấu tích phân, ta có thể tránh được sai sót khi áp dụng các mã cầu phương. Điều này chỉ có thể làm được nếu ta biết thuật toán của chương trình.

Hàm dưới dấu tích phân dao động

Nếu hàm dưới dấu tích phân $f(x)$ là tuần hoàn chu kỳ p, i.e., $f(x+p) =$

$f(x)$ với mọi x, và $b-a$ là một bội của chu kỳ, $a-b=np$, thì

$$\int_a^b f(x)dx = n\int_0^p f(x)dx.$$

Khi đó, chỉ cần áp dụng quy tắc cầu phương cho tích phân trong một chu kỳ.

Đối với hàm dao động không tuần hoàn vấn đề có khó khăn hơn. Nói chung, khoảng lấy tích phân nên được phân thành nhiều khoảng con sao cho mỗi khoảng con chỉ chứa vài dao động. Thí dụ, tích phân

$$\int_0^\pi \frac{\sin(20x)}{1+x^2}dx,$$

có thể viết lại:

$$\sum_{j=1}^{2} 0 \int_{(j-1)\pi/20}^{j\pi/20} \frac{\sin(20x)}{1+x^2}dx,$$

khi đó việc áp dụng mã cầu phương thích ứng cho kết quả tốt.

Tích phân với cận vô hạn

Mã cầu phương thích ứng không thể áp dụng trực tiếp để tính tích phân với vận vô hạn. Một cách để áp dụng nó là dùng định nghĩa

$$\int_a^\infty f(x)dx = \lim_{b\to\infty}\int_a^b f(x)dx.$$

Ý tưởng là xác định một chặn giải tích cho phần dư $|\int_b^\infty f(x)dx|$. Nhờ nó cận b được chọn đủ lớn để cho $\int_a^b f(x)dx$ xấp xỉ $\int_a^\infty f(x)dx$ với độ chính xác yêu cầu. Không thành vấn đề nếu b lớn hơn giá trị cần thiết, vì vậy một chặn "thô" cho phần dư là đủ.

Một cách khác là đổi biến thích hợp để có tích phân với cận hữu hạn. Chẳng hạn, để đánh giá tích phân $\int_1^\infty e^{-x}\sin x dx$, dùng biến mới $s=1/x$, tích phân thành

$$\int_0^1 e^{1/s}\sin(1/s)/s ds$$

trên khoảng hữu hạn [0, 1]. Nói chung, điều này sinh ra khó khăn khác, tích phân có kỳ dị ở điểm cuối. Trong trường hợp đặc biệt ở đây, $\lim_{s\to 0^+} e^{1/s}\sin(1/s)/s = 0$, vì vậy hàm dưới dấu tích phân liên tục tại $s = 0$, và mã cầu phương thích ứng có thể áp dụng được.

5.6 Tích phân của bảng dữ liệu

Bài toán được bàn đến ở đây là xấp xỉ $\int_a^b f(x)dx$ mà chỉ được cho (x_n, y_n) với $1 < n < N$, trong đó $y_n = f(x_n)$. Các chương trình con cầu phương thích ứng không thể dùng được vì chúng tự động chọn các điểm ở đó $f(x)$ được đánh giá mà các điểm này có thể không nằm trong dữ liệu $\{x_i\}$ được cho của hàm. Cách tiếp cận cơ bản: xấp xỉ $f(x)$ bằng đa thức từng mảnh $F(x)$, rồi tích phân hàm này cách chính xác.

Vì spline bậc ba cho xấp xỉ tốt nên cách chọn tự nhiên hàm $F(x)$ là spline bậc ba. Để đơn giản, giả sử $a = x_1$ và $b = x_N$. Dùng ký hiệu của chương 3 cho spline,

$$\begin{aligned}\int_a^b S(x)dx &= \sum_{n=1}^{N-1}\int_{x_n}^{x_{n+1}} S(x)dx \\ &= \sum_{n=1}^{N-1}\left(a_n h_n + b_n\frac{h_n^2}{2} + c_n\frac{h_n^3}{3} + d_n\frac{h_n^4}{4}\right).\end{aligned}$$

Thay các biểu thức của a_n, b_n và d_n theo dữ liệu (f_n) và c_n, ta được

$$\begin{aligned}\int_a^b S(x)dx &= \sum_{n=1}^{N-1}\left\{f_n h_n + \left[\frac{f_{n+1}-f_n}{h_n} - \frac{2}{3}c_n h_n - \frac{1}{3}c_{n+1}h_n\right]\right. \\ &\qquad \left. + \frac{h_n^2}{2} + c_n\frac{h_n^3}{3} + \frac{c_{n+1}-c_n}{3h_n}\frac{h_n^4}{4}\right\} \\ &= \sum_{n=1}^{N-1}\left[\frac{h}{2}(f_n + f_{n+1}) - \frac{h_n^3}{12}(c_n + c_{n+1})\right]. \qquad (5.24)\end{aligned}$$

Một sơ đồ được dùng rộng rãi đặt cơ sở trên nội suy bậc hai địa phương. Để xấp xỉ $f(x)$ bằng đa thức bậc hai trên $[x_n, x_{n+1}]$ cần phải nội suy hàm tại ba điểm. Ta có thể nội suy tại x_{n-1}, x_n, x_{n+1} hoặc x_n, x_{n+1}, x_{n+2}. Không có lý do gì để khẳng định cách nào cho kết quả tốt hơn, vì vậy một cách phù hợp là tính bằng cả hai cách rồi lấy trung bình. Kết quả là một công thức đối xứng làm trơn hóa sai số có trong bảng dữ liệu. Tất nhiên, ở các khoảng chứa điểm dầu, cuối ($n = 1$, $n = N - 1$), chỉ dùng một trong hai phép nội suy.

5.7 Tích phân bội

Tích phân xác định theo hai hay nhiều biến hơn, nói chung, khó xấp xỉ hơn nhiều, chủ yếu do hình học của miền lấy tích phân. Mục này chỉ đưa ra một số bàn luận cho trường hợp hai biến, đặc biệt, những vấn đề có liên quan đến phương pháp phần tử hữu hạn.

Tích phân trên hình chữ nhật,

$$I(f) = \int_{a_1}^{b_1} \int_{a_2}^{b_2} f(x, y) dx dy,$$

có thể thực hiện dễ dàng nhờ công thức cho trường hợp một biến, bằng tính tích phân lặp. Đầu tiên xấp xỉ

$$I(f) \approx \sum_{i=1}^{N_1} A_i \int_{a_2}^{b_2} f(x_i, y) dy$$

với quy tắc cầu phương dùng N_1 điểm $\{x_i\}$, và rồi

$$I(f) \approx \sum_{i=1}^{N_1} A_i \left(\sum_{j=1}^{N_2} B_j f(x_i, y_j) \right),$$

dùng quy tắc N_2 điểm $\{y_j\}$. Cách làm này có thể tổng quát hóa cho trường hợp

$$I(f) = \int_{a_1}^{b_1} \int_{x_1(y)}^{x_2(y)} f(x, y) dy dx.$$

Bậc chính xác bây giờ tham chiếu đến các đa thức theo hai biến, vì vậy một công thức, chẳng hạn, có bậc chính xác là 2 thì phải tích phân chính xác tất cả các đa thức có dạng

$$a_{0,0} + a_{l,0}x + a_{0,l}y + a_{2,0}x^2 2 + a_{l,l}xy + a_{0,2}y^2$$

trên miền đang xét. Giống như trong trường hợp một biến, ta có thể thiết lập công thức cầu phương bằng cách nội suy $f(x, y)$ và tích phân hàm nội suy. Điều này hoàn toàn thực hiện được trên hình vuông hay tam giác như chỉ ra ở trên. Sơ đồ tính cho hình chữ nhật dựa trên tích phân lặp có hiệu quả khi công thức cho trường hợp một biến là quy tắc Gauss. Nhưng chúng không nhất thiết là cái tốt nhất. Cũng như trường hợp một biến, có thể thiết lập công thức các đánh giá $f(x, y)$ là tối thiểu với bậc chính xác cho trước. Tuy nhiên, công thức hiệu quả nhất có thể không "hấp dẫn" trong thực hành. Cách tiếp cận dựa trên nội suy rất tiện lợi khi phép nội suy được thực hiện tại các điểm được quan tâm vì lý do khác, như trong phương pháp phần tử hữu hạn. Cách tiếp cận bằng tích phân lặp có thể rất thuận tiện vì tính đơn giản và tổng quát của nó.

Trong một chiều biến đổi khoảng hữu hạn bất ký $[a, b]$ thành khoảng chuẩn $[-1, 1]$ là tầm thường. Trong trường hợp hai chiều thì vấn đề trở nên quan trọng và khó hơn nhiều. Giả sử ta cần tính tích phân trên miền R tổng quát. Khi đó, R phải được phân nhỏ thành các mảnh có thể biến đổi thành hình vuông hay tam giác chuẩn. Việc rời rạc hóa miền R theo cách này là phần quan trọng trong mã phần tử hữu hạn. Nếu miền R được phân hoạch thành các tam giác (với cạnh thẳng) thì phép biến đổi là affine rất đơn giản. Một tích phân trên tam giác tổng quát T trở thành tích phân trên tam giác chuẩn T^* (trong phần tử hữu hạn gọi là *tam giác tham chiếu*),

$$\int\int_T f(x, y)dxdy = \int_{T^*} f(x^*, y^*)|D^*|dx^*dy^*$$

Ở đây D^* là định thức Jacobi của phép biến đổi.

Trên đây chỉ là những bàn luận về ý tưởng cơ bản, áp dụng công thức một biến cho trường hợp nhiều biến. Có nhiều vấn đề nảy sinh liên quan đến phép phân hoạch miền, các phép biến đổi mỗi mảnh tổng quát về miền chuẩn. Lãnh vực tích phân hàm nhiều biến cho đến nay vẫn còn đang nghiên cứu.

Câu hỏi và bài tập

5.1. Dùng phương pháp hệ số bất định để thiết lập quy tắc Newton 3/8

$$\int_{-1}^{1} f(x)dx = A_1 f(-1) + A_2 f\left(-\frac{1}{3}\right) + A_3 f\left(\frac{1}{3}\right) + A_4 f(1) + cf^{(d+1)}(\xi).$$

Tính A_1, A_2, A_3, A_4, d và c.

5.2. Dùng phương pháp hệ số bất định để tìm công thức cầu phương Gauss 2-điểm với sai số liên kết. Bắt đầu bằng

$$\int_{-1}^{1} f(x)dx = A_1 f(-x_1) + A_2 f(x_1) + E(f)$$

và tính A_1 và x_1. Giả sử $E(f) = cf^{(d+1)}(\xi)$, tìm d và c. Công thức trong trường hợp tổng quát, khoảng $[a, b]$.

5.3. Cài đặt quy tắc cầu phương hình thang đa hợp và áp dụng nó để tính

$$\int_0^{\pi} \frac{dx}{4 + \sin(20x)}.$$

Tất nhiên bạn phải chọn h đủ nhỏ và được lấy trong mỗi chu kỳ. Xấp xỉ tích phân với một số cách chọn h dần về 0. Theo lý thuyết thì T_n hội tụ rất nhanh. Ở đây bạn thấy gì?

Chương 6

Phương trình vi phân thường

6.1 Cơ sở lý thuyết

Cho hàm $f(x, y)$ liên tục (theo biến x) trong đoạn $[a, b]$ với mọi y. Phương trình vi phân cấp một tổng quát có dạng

$$y'(x) = f(x, y(x)) \tag{6.1}$$

với mọi $x \in (a, b)$. Trong chương này ta xét bài toán tìm nghiệm $y(x)$, là hàm của x có đạo hàm liên tục khi $x \in (a, b)$, thỏa phương trình (6.1) và giá trị của nó tại điểm đầu của khoảng:

$$y(a) = A \tag{6.2}$$

Phương trình (6.2) được gọi là điều kiện đầu, và tổ hợp (6.1) và (6.2) được gọi là *bài toán giá trị đầu* hay *bài toán Cauchy* cho phương trình vi phân.

Một điều kiện đơn giản bảo đảm sự tồn tại và duy nhất nghiệm có thể được thiết lập nhờ cách $f(x, y)$ phụ thuộc y.

Hàm $f(x, y)$ thỏa điều kiện Lipschitz theo y nếu với mọi x trong khoảng $[a, b]$ và với mọi u, v,

$$|f(x, u) - f(x, v)| \leq L|u - v| \tag{6.3}$$

với L là hằng số, sau này được gọi là hằng số Lipschitz. Trường hợp f có đạo hàm riêng liên tục theo biến thứ hai,

$$|f(x, u) - f(x, v)| = \left|\frac{\partial f}{\partial y}(x, w)\right| |u - v|$$

với w ở giữa u và v, và nếu $\partial f/\partial y$ bị chặn với mọi đối số, thì f thỏa điều kiện Lipschitz và hằng số L bất kỳ sao cho

$$\left|\frac{\partial f}{\partial y}(x,w)\right| \le L$$

với mọi x trong $[a,b]$ và với mọi w là một hằng Lipschitz. Nếu đạo hàm riêng không bị chặn, có thể chỉ ra rằng bất đẳng thức (6.3) không thể đúng với mọi u, v và với mọi x trong $[a,b]$, vậy f không thỏa điều kiện Lipschitz.

Thí dụ 6.1. Hàm $f(x,y) = x^2\cos^2 y + y\sin^2 x$, xác định với $|x| \le 1$ và mọi y, là Lipschitz với hằng số $L = 3$. Để thấy điều này, đạo hàm đối với y cho

$$\frac{\partial f}{\partial y} = -2x^2\cos y \sin y + \sin^2 x,$$

và như vậy với mọi x, $|x| \le 1$, ta có

$$\left|\frac{\partial f}{\partial y}\right| \le 2\times 1\times 1 + 1 = 3 \circ$$

Thí dụ 6.2. Hàm $f(x,y) = \sqrt{|y|}$ không thỏa điều kiện Lipschitz vì nó có đạo hàm riêng liên tục với $y > 0$, không bị chặn khi $y \to 0$:

$$\frac{\partial f}{\partial y} = \frac{1}{2\sqrt{y}} \circ$$

Một trường hợp quan trọng của (6.1) là phương trình vi phân tuyến tính, $f(x,y) = g(x)y + h(x)$. Hàm $f(x,y)$ liên tục theo (x,y) tương đương với $g(x)$ và $h(x)$ liên tục theo x. Vì

$$\frac{\partial f}{\partial y} = g(x)$$

và vì hàm liên tục $g(x)$ bị chặn trên khoảng hữu hạn $[a,b]$ bất kỳ, nên phương trình tuyến tính thỏa điều kiện Lipschitz trong hầu hết các trường hợp thực hành.

Thí dụ 6.3. Tích phân Dawson là hàm

$$y(x) = e^{-x^2} \int_0^x e^{t^2} dt.$$

Có thể kiểm tra rằng tích phân trên là nghiệm của bài toán giá trị đầu cho phương trình vi phân tuyến tính

$$\begin{aligned} y' &= 1 - 2xy, \\ y(0) &= 0. \end{aligned}$$

Trên khoảng $[0, b]$ với bất kỳ $b \neq 0$, hàm $f(x, y) = 1 - 2xy$ liên tục và Lipschitz với hằng số Lipschitz $L = 2|b|$ ∘

Các điều kiện đủ để phương trình vi phân tồn tại và duy nhất nghiệm có thể được phát biểu một cách hình thức:

Định lý 6.1. *Cho $f(x, y)$ liên tục với mọi x trong khoảng $[a, b]$ và mọi y, và thỏa (6.3). Thì với bất kỳ số A, bài toán giá trị đầu $y' = f(x, y)$, $y(a) = A$ có nghiệm duy nhất $y(x)$ xác định với mọi x thuộc khoảng $[a, b]$.*

Cho đến nay ta đã nói về một phương trình vi phân với một ẩn $y(x)$. Một hệ phương trình vi phân cấp một với m ẩn là

$$\begin{aligned} Y_1' &= F_1(x, Y_1, Y_2, \ldots, Y_m) \\ Y_2' &= F_2(x, Y_1, Y_2, \ldots, Y_m) \\ &\vdots \\ Y_m' &= F_m(x, Y_1, Y_2, \ldots, Y_m) \end{aligned} \tag{6.4}$$

Cùng với các phương trình (6.4) có các điều kiện đầu

$$\begin{aligned} Y_1(a) &= A_1 \\ Y_2(a) &= A_2 \\ &\vdots \\ Y_m(a) &= A_m \end{aligned} \tag{6.5}$$

Nếu đặt

$$\mathbf{Y}(x) = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_m \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_m \end{bmatrix}, \quad \mathbf{F}(x, \mathbf{Y}) = \begin{bmatrix} F_1(x, \mathbf{Y}) \\ F_2(x, \mathbf{Y}) \\ \vdots \\ F_m(x, \mathbf{Y}) \end{bmatrix} \tag{6.6}$$

thì (6.4) và (6.5) trở thành

$$\mathbf{Y}' = \mathbf{F}(x, \mathbf{Y}), \tag{6.7}$$

$$\mathbf{Y}(a) = \mathbf{A}. \tag{6.8}$$

Một lần nữa ta xem tổ hợp của (6.4) và (6.5) như là bài toán giá trị đầu. Bằng cách dùng ký hiệu vectơ làm cho trường hợp m ẩn trông giống như trường hợp một ẩn. Một trong các khía cạnh may mắn của lý thuyết bài toán giá trị đầu là lý thuyết cho hệ phương trình vi phân cấp một cốt yếu giống như trường hợp một ẩn. Các chứng minh cho hệ chính là đưa vào các vectơ và chuẩn của chúng ở đâu có các vô hướng và các giá trị tuyệt đối trong chứng minh cho một ẩn. Với hàm vectơ $\mathbf{F}(x, \mathbf{Y})$ thỏa điều kiện Lipschitz, điều kiện đủ là mỗi $F_i(x, Y_1, Y_2, \ldots, Y_m)$ thỏa điều kiện Lipschitz đối với mỗi Y_j; nghĩa là, tồn tại các hằng số L_{ij} sao cho

$$|F_i(x, Y_1, \ldots, Y_{j-1}, u, Y_{j+1}, \ldots, Y_m) - F_i(x, Y_1, \ldots, Y_{j-1}, v, Y_{j+1}, \ldots, Y_m)| \leq L_{ij}|u - v|$$

với mọi i, j. Với điều này, định lý tương tự định lý 6.1 cho trường hợp m ẩn đúng. Vì lý thuyết các phương pháp số cho hệ các phương trình về cốt yếu cũng giống như với một phương trình, nên ta tự hạn chế chỉ đối xử chi tiết với trường hợp một phương trình và phát biểu kết quả tương tự cho hệ.

Hầu hết chương trình máy tính đòi hỏi bài toán phải được cho dưới dạng chuẩn (6.4) và (6.5), nhưng các phương trình xuất hiện trong nhiều dạng khác nhau. Chẳng hạn, phương trình cấp hai, nghĩa là các phương trình dạng

$$y'' = g(x, y, y'),$$

thường gặp trong các tài liệu về hệ động lực. Định nghĩa về nghiệm là mở rộng hiển nhiên của trường hợp cấp một và điều kiện đầu thích hợp là $y(a) = A_1, y'(a) = A_2$. Đây là phương trình vi phân cấp hai cho một đại lượng chưa biết, $y(x)$. Một bài toán tương đương ở dạng chuẩn (6.4) có thể được tìm bằng cách đưa vào hai đại lượng chưa biết và tìm hai phương trình vi phân cấp một được thỏa bởi chúng. Một trong hai ẩn mới phải cho chúng

ta ẩn gốc, vậy ta lấy $Y_1(x) = y(x)$. Ta lấy ẩn còn lại là đạo hàm của ẩn gốc, $Y_2(x) = y'(x)$. Đạo hàm các ẩn mới, ta thu được

$$\begin{aligned} Y_1' &= y'(x) = Y_2(x), \\ Y_2' &= y''(x) = g(x, y(x), y'(x)) = g(x, Y_1(x), Y_2(x)). \end{aligned}$$

Bằng cách này ta đi đến hệ hai phương trình vi phân cấp một theo hai ẩn:

$$\begin{aligned} Y_1' &= Y_2, \\ Y_2' &= g(x, Y_1, Y_2). \end{aligned}$$

Đây là dạng chuẩn và lý thuyết có thể áp dụng cho nó để kết luận sự tồn tại nghiệm duy nhất $Y_1(x)$ và $Y_2(x)$ thỏa điều kiện đầu

$$\begin{aligned} Y_1(a) &= A_1, \\ Y_2(a) &= A_2. \end{aligned}$$

Nghiệm của bài toán gốc nhận được từ $y(x) = Y_1(x)$. Để kiểm điều này, trước hết chú ý rằng một phương trình phát biểu rằng $y'(x) = Y_1'(x) = Y_2(x)$, và phương trình còn lại phát biểu rằng

$$y''(x) = Y_2'(x) = g(x, Y_1(x), Y_2(x)) = g(x, y(x), y'(x)).$$

Tương tự, có thể thấy rằng các điều kiện đầu được thỏa.

Phương trình vi phân cấp m tổng quát một ẩn,

$$\begin{aligned} y^{(m)} &= g(x, y, y', \ldots, y^{(m-1)}), \\ y(a) &= A_1, y'(a) = A_2, \ldots, y^{(m-1)}(a) = A_m \end{aligned}$$

có thể được đặt thành dạng chuẩn theo m ẩn $Y_1(x) = y(x), Y_2(x) = y'(x), \ldots, Y_m(x) = y^{(m-1)}(x)$ và

$$\begin{aligned} F_1(x, Y_1, Y_2, \ldots, Y_m) &= Y_2 \\ F_2(x, Y_1, Y_2, \ldots, Y_m) &= Y_3 \\ &\vdots \\ F_{m-1}(x, Y_1, Y_2, \ldots, Y_m) &= Y_m \\ F_m(x, Y_1, Y_2, \ldots, Y_m) &= g(x, Y_1, Y_2, \ldots, Y_m). \end{aligned}$$

Thí dụ 6.4. Để chuyển bài toán giá trị đầu

$$y'' + (y^2 - 1)y' + y = 0, \ y(0) = 1, \ y'(0) = 4$$

thành hệ phương trình vi phân cấp một, đặt

$$Y_1(x) = y(x),\ Y_2(x) = y'(x).$$

Thì

$$\begin{aligned} Y_1' &= y' = Y_2 \\ Y_2' &= y'' = -(Y_1^2 - 1)Y_2 - Y_1 \end{aligned}$$

và

$$Y_1(0) = 1,\ Y_2(0) = 4.$$

Bài toán này có thể đặt thành dạng (6.4) bằng cách định nghĩa

$$\mathbf{Y} = \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix},\ \mathbf{A} = \begin{bmatrix} 1 \\ 4 \end{bmatrix},\ \mathbf{F}(x, \mathbf{Y}) = \begin{bmatrix} Y_2 \\ -(Y_1^2 - 1)Y_2 - Y_1 \end{bmatrix} \circ$$

6.2 Một sơ đồ số đơn giản

Xét bài toán giá trị đầu (6.1) và (6.2),

$$\begin{aligned} y' &= f(x, y) \\ y(a) &= A, \end{aligned}$$

trên khoảng $[a, b]$. Các phương pháp số ta xét sinh ra một bảng các giá trị xấp xỉ cho $y(x)$. Tạm thời ta giả sử rằng các điểm nhập vào cách đều theo biến không gian x. Nghĩa là, ta chọn một số nguyên N và với $h = (b-a)/N$, ta xây dựng xấp xỉ tại các điểm $x_n = a + nh$ với $n = 0, 1, \ldots, N$. Ký hiệu $y(x_n)$ được dùng cho nghiệm của (6.1) và (6.2) được đánh giá tại $x = x_n$, còn y_n được dùng cho một xấp xỉ của $y(x_n)$.

Phương trình vi phân không có "ký ức". Nếu ta biết giá trị $y(x_n)$, Định lý 6.1 áp dụng cho bài toán

$$\begin{aligned} u' &= f(x, u) \\ u(x_n) &= y(x_n) \end{aligned}$$

nói rằng nghiệm của bài toán giá trị đầu này trên khoảng $[x_n, b]$ chính là $y(x)$. [Sau hết, $y(x)$ là nghiệm và định lý nói rằng chỉ có một nghiệm.] Nghĩa là, các giá trị của $y(x)$ với x ở trước x_n, không ảnh hưởng trực tiếp đến nghiệm của phương trình vi phân với x ở sau x_n. Một vài phương pháp số

cho phương trình vi phân có ký ức và một vài phương pháp thì không. Lớp các phương pháp được biết như là *phương pháp một bước* (one-step method) không có ký ức - cho trước y_n, có một công thức cho giá trị y_{n+1} phụ thuộc vào x_n, y_n, f và h. Bắt đầu với giá trị ban đầu hiển nhiên $y_0 = A$, phương pháp một bước sinh ra một bảng giá trị $y(x)$ bằng cách thực hiện lập lại một bước theo x với độ dài h để sinh ra dãy liên tiếp $y_1, y_2, \ldots$

Thí dụ đơn giản nhất của phương pháp một bước là *phương pháp Euler*. Ta nghiên cứu nó vì các chi tiết không làm mờ đi ý tưởng và trường hợp tổng quát là rất giống. Khai triển Taylor $y(x)$ quanh $x = x_n$, cho

$$y(x_{n+1}) = y(x_n) + hy'(x_n) + \frac{h^2}{2}y''(\xi_n)$$

với $x_n < \xi_n < x_{n+1}$, miễn là $y(x) \in C^2[a, b]$. Dùng sự kiện $y(x)$ thỏa (6.1), phương trình trên thành

$$y(x_{n+1}) = y(x_n) + hf(x_n, y(x_n)) + \frac{h^2}{2}y''(\xi_n). \tag{6.9}$$

Với h nhỏ,

$$y(x_{n+l}) \approx y(x_n) + hf(x_n, y(x_n)).$$

Hệ thức này đề nghị

Phương pháp Euler

$$\begin{aligned} y_0 &= A \\ y_{n+1} &= y_n + hf(x_n, y_n), \quad n = 0, 1, \ldots, N-1. \end{aligned} \tag{6.10}$$

Thí dụ 6.5. Lập bảng tích phân Dawson trên $[0, 0.5]$ dùng sơ đồ Euler với $h = 0.1$. Nhắc lại, từ thí dụ 6.3 rằng tích phân Dawson là nghiệm của bài toán giá trị đầu

$$\begin{aligned} y' &= 1 - 2xy \\ y(0) &= 0. \end{aligned}$$

Lấy $y_0 = 0$, ta thấy rằng

$$y_1 = 0 + 0,1 \times (1 - 2 \times 0 \times 0) = 0.1;$$

tương tự,

$$y_2 = 0.1 + 0,1 \times (1 - 2 \times 0.1 \times 0.1) = 0.198.$$

Tiếp tục theo lối này, ta nhận được bảng kết quả sau. Giá trị chính xác của tích phân được lấy từ [1].

x_n	y_n	$y(x_n)$
0.0	0.00000	0.00000
0.1	0.10000	0.09934
0.2	0.19800	0.19475
0.3	0.29008	0.28263
0.4	0.37268	0.35994
0.5	0.44287	0.42444

∘

Để nghiên cứu sự hội tụ của phương pháp Euler, ta liên hệ sai số tại x_{n+1} với sai số tại x_n. Trừ (6.10) với (6.9) cho

$$y(x_{n+1}) - y_{n+1} = y(x_n) - y_n + h[f(x_n, y(x_n)) - f(x_n, y_n)] + \frac{h^2}{2}y''(\xi_n).$$

Ký hiệu sai số tại x_n bởi $E_n = y(x_n) - y_n$, điều kiện Lipschitz trên f và phương trình này đưa đến

$$|E_{n+1}| < |E_n| + hL|y(x_n) - y_n| + \frac{h^2}{2}|y''(\xi_n)|.$$

Với điều kiện

$$M_2 = \max_{a \le x \le b} |y''(x)|$$

ta được

$$|E_{n+1}| < |E_n|(1 + hL) + \frac{h^2}{2}M_2, \quad n = 0, 1, \ldots, N-1. \tag{6.11}$$

Ở đây số hạng $h^2M_2/2$ chặn sai số trong bước hiện hành và số hạng còn lại chặn sai số truyền từ các bước trước.

Để chứng minh sự hội tụ, ta chặn sai số có thể xuất hiện khi ta bước từ $x_0 = a$ tới $x_N = b$ và rồi chứng tỏ rằng nó dần tới không khi h dần tới không. Công việc đầu tiên là xét xem bất đẳng thức (6.11) cho phép sai số phát triển nhanh như thế nào. Để làm điều này ta thiết lập một kết quả tổng quát mà sau này sẽ dùng đến.

Bổ đề 6.1. *Giả sử tồn tại các số thực $\delta > 0$ và $M > 0$ sao cho dãy $d_0, d_1, \ldots$ thỏa*

$$d_{n+1} \leq (1+\delta)d_n + M, \quad n = 0, 1, \ldots,$$

thì

$$d_n \leq (1+\delta)^n d_0 + M[1 + (1+\delta) + (1+\delta)^2 + \ldots + (1+\delta)^{n-1}]. \tag{6.12}$$

Chứng minh. Để chứng minh điều này ta dùng quy nạp. Dễ thấy, bất đẳng thức (6.12) đúng với $n = 1$. Giả sử bất đẳng thức (6.12) đúng với trường hợp $n = k$. Thì

$$\begin{aligned} d_{k+1} &\leq (1+\delta)d_k + M \\ &\leq (1+\delta)^{k+1} d_0 + M[1 + (1+\delta) + \ldots + (1+\delta)^k], \end{aligned}$$

nghĩa là, bất đẳng thức đúng với $n = k+1$ và chứng minh hoàn tất. ■

Bổ đề 6.2. *Giả sử có các số $\delta > 0$ và $M > 0$ sao cho dãy $d_0, d_1, \ldots$ thỏa*

$$d_{k+1} \leq (1+\delta)d_k + M, \quad k = 0, 1, \ldots$$

Thì với $n > 0$ bất kỳ,

$$d_n \leq e^{n\delta} d_0 + M\frac{e^{n\delta} - 1}{\delta}. \tag{6.13}$$

Chứng minh. Áp dụng công thức tính tổng cấp số nhân với công bội $x = 1+\delta$, ta thấy vế phải của (6.12) có thể viết dưới dạng

$$(1+\delta)^n d_0 + M\frac{(1+\delta)^n - 1}{\delta}. \tag{6.14}$$

Khai triển hàm mũ ở lân cận không, với $\delta > 0$, cho

$$e^\delta = 1 + \delta + \frac{\delta^2}{2}e^\eta, \quad 0 < \eta < \delta.$$

Suy ra

$$1 + \delta \leq e^\delta$$

và

$$(1+\delta)^n \le e^{n\delta}.$$

Điều này cho thấy (6.14) bị chặn bởi

$$e^{n\delta} d_0 + M \frac{e^{n\delta} - 1}{\delta},$$

và (6.13) được chứng minh. ■

Bây giờ trở lại với phương pháp Euler, ta áp dụng Bổ đề 6.2 cho (6.11) và đi đến

$$|E_n| \le e^{nhL}|E_0| + \frac{hM_2}{2L}(e^{nhL} - 1).$$

Tuy nhiên, $nh = x_n - a$ và $E_0 = y_0 - A = 0$, vì vậy

$$|y(x_n) - y_n| \le \frac{hM_2}{2L}(e^{L(x_n - a)} - 1). \tag{6.15}$$

Dùng $x_n - a < b - a$, điều này dẫn đến

$$\max_{0 \le n \le N} |y(x_n) - y_n| \le \frac{hM_2}{2L}(e^{L(b-a)} - 1). \tag{6.16}$$

Ta thấy rằng sai số của phương pháp Euler bị chặn bởi một hằng số lần h. Khi giá trị của hằng số không quan trọng, các biểu thức như vậy được viết là $0(h)$.

Nói chung, ta đã cố tình lờ đi ảnh hưởng của số học chính xác hữu hạn. Tuy nhiên, nếu nghiệm khó xấp xỉ chính xác tại x_n, kích thước bước có thể phải nhỏ đến nỗi độ chính xác cần được xét. Để ý rằng, từ chương trình con, ta không nhận được $f(x_n, y_n)$ mà được $f(x_n, y_n) + \epsilon_n$. Tương tự, trong tính toán $y_{n+1} = y_n + h[f(x_n, y_n) + \epsilon_n]$ thêm sai số ρ_n được tạo ra. Thì kết quả dãy tính toán sinh ra là

$$y_{n+1} = y_n + hf(x_n, y_n) + h\epsilon_n + \rho_n.$$

Ta hãy giả sử rằng $|\rho_n| \le \rho$ và $|\epsilon_n| \le \epsilon$ với mọi $h \le h_0$. Thì phân tích trên có thể được hiệu chỉnh để nhận được

$$\max_{0 \le n \le N} |y(x_n) - y_n| \le \frac{e^{L(b-a)} - 1}{L} \left(\frac{hM_2}{2} + \epsilon + \frac{\rho}{h} \right).$$

Theo chặn này, các ảnh hưởng làm tròn là xấu khi ta giảm kích thước bước nhằm thu được nghiệm chính xác hơn. Rõ ràng có một độ chính xác cực đại phụ thuộc vào bài toán, phương pháp số, và số học mà máy tính sử dụng. Các ảnh hưởng thì phức tạp hơn điều mà chặn này cho thấy, nhưng một cách định tính chặn là chính xác. Dễ dàng chứng tỏ bằng thực nghiệm số rằng khi h giảm, nghiệm số thoạt đầu chính xác hơn, tiến tới một giá trị tốt, rồi thì sau đó sự giảm thiểu chính xác gia tăng.

Phép phân tích sự hội tụ vừa trình bày là cách truyền thống. Cái khó là đây không phải là cách mà các chương trình hiện nay làm việc. Thực ra từ kích thước bước chỉ định h, chương trình tự động chọn một kích thước bước mà sẽ sinh ra một nghiệm với độ chính xác chỉ định. Một mô hình hợp lý cho kích thước bước được chọn (trong các chương trình như vậy) là tại x_n chương trình chọn một bước $h_n = \Theta(x_n)H$, trong đó $\Theta(x)$ là một hàm liên tục từng khúc thỏa $0 < \theta \leq \Theta(x) \leq 1$ với $a \leq x \leq b$. Với mô hình này ta dễ dàng sửa đổi chứng minh hội tụ vừa cho để tính đến sự thay đổi kích thước bước. Kết quả là khi kích thước bước cực đại H dần tới không, $\max_{0 \leq n \leq N} |y(x_n) - y_n|$ là $0(H)$. Người ta chỉ định trước một dung sai τ. Trong bước độ dài h từ x_n, phương pháp Euler tạo ra một sai số xấp xỉ bằng $h^2|y''(x_n)|/2$. Kích thước bước lớn nhất h_n có thể được dùng mà vẫn giữ sai số nhỏ hơn τ là

$$h_n \approx \sqrt{\frac{2\tau}{|y''(x_n)|}}.$$

Khi $y''(x_n)$ gần bằng không, ta cần đến các quy tắc đặc biệt trong chương trình. Giả sử $y''(x)$ không triệt tiêu trong $[a, b]$. Nếu

$$\zeta = \min_{[a,b]} |y''(x)| > 0$$

và

$$H = \sqrt{\frac{2\tau}{\zeta}}$$

thì

$$h_n \approx \sqrt{\frac{\zeta}{|y''(x_n)|}} H = \Theta(x_n)H$$

xác định $\Theta(x)$. Chú ý rằng $H = 0(\tau^{1/2})$ để $\max|y(x_n) - y_n|$ là $0(\tau^{1/2})$ cho phương pháp Euler với sự chọn lựa tự động kích thước bước.

6.3 Các phương pháp một bước

Bây giờ ta xét các phương pháp một bước và đặt các giả thiết của chúng dựa theo phương pháp Euler. Công thức tổng quát có dạng

$$\begin{aligned} y_0 &= A, \\ y_{n+1} &= y_n + h\Phi(x_n, y_n, f, h), \quad n = 0, 1, \ldots \end{aligned} \tag{6.17}$$

Phương pháp không có ký ức, nên Φ chỉ phụ thuộc vào các đối số x_n, y_n, f, h. Thông thường f và h được bỏ đi trong ký hiệu. Giả sử Φ liên tục theo x và y. Phương pháp Euler lấy $\Phi(x, y) = f(x, y)$ và điều kiện Lipschitz được dùng là cốt yếu. Vậy, với công thức tổng quát ta giả sử rằng

$$|\Phi(x, u) - \Phi(x, v)| \leq L_\Phi |u - v| \tag{6.18}$$

khi $a \leq x \leq b$, với mọi $0 < h \leq h_0$ với h_0 nào đó, hàm liên tục bất kỳ f thỏa điều kiện Lipschitz, và với mọi u, v.

Khi bàn luận về phương pháp Euler ta đã dùng, như là điểm bắt đầu, sự kiện nghiệm $y(x)$ hầu như thỏa công thức (6.10) để xác định xấp xỉ số. Cái tương tự ở đây là

$$y(x_{n+1}) = y(x_n) + h\Phi(x_n, y(x_n)) + h\mu_n, \tag{6.19}$$

với μ_n "nhỏ". Chính xác hơn, nếu với mọi x_n trong $[a, b]$ và mọi $h \leq h_0$, có các hằng số C và p sao cho

$$|\mu_n| \leq Ch^p, \tag{6.20}$$

thì ta nói rằng phương pháp thuộc cấp p cho phương trình (6.1). Đại lượng μ_n được gọi là *sai số chặt cụt địa phương* (local truncation error).

Định lý 6.2. *Giả sử bài toán giá trị đầu*

$$\begin{aligned} y' &= f(x, y), \\ y(a) &= A \end{aligned}$$

trên khoảng hữu hạn $[a, b]$ được giải bằng phương pháp một bước (6.17) và giả sử rằng các giả thiết của định lý 6.1 được thỏa. Nếu $\Phi(x, y)$ thỏa (6.18) và nếu phương pháp là cấp $p > 1$ cho $y(x)$, thì với bất kỳ $x_n = a + nh \in [a, b]$

$$|y(x_n) - y_n| \leq \frac{Ch^p}{L_\Phi}(e^{L_\Phi(x_n - a)} - 1). \tag{6.21}$$

Chứng minh. Như trước, đặt $E_n = y(x_n) - y_n$ và trừ (6.17) cho (6.19) ta được

$$E_{n+1} = E_n + h[\Phi(x_n, y(x_n)) - \Phi(x_n, y_n)] + h\mu_n$$

Dùng điều kiện Lipschitz (6.18) và giả thiết phương pháp là cấp p, ta thấy rằng

$$|E_{n+1}| = (1 + hL_\Phi)|E_n| + Ch^p.$$

Bây giờ định lý là kết quả của Bổ đề 6.2 và sự kiện $E_0 = 0$. ■

Cũng như bàn luận của phương pháp Euler, kết quả của định lý cho sự hội tụ $0(h^p)$. Điều này giải thích việc ta gọi phương pháp là cấp p cho $y(x)$. Thuật ngữ "phương pháp thuộc cấp p" được dùng để mô tả một phương pháp mà thuộc cấp p nếu f là đủ trơn. Cấp của sự hội tụ là thấp hơn khi f không trơn như vậy.

Như đã giải thích trong mối liên hệ với phương pháp Euler, đoạn mã chọn tự động kích thước bước để giữ cho sai số luôn nhỏ hơn một dung sai τ. Đồng thời chúng cố gắng dùng một bước đủ lớn. Một mô hình hợp lý của thuật toán tìm kích thước bước như vậy dẫn đến một kích thước bước h_n tại x_n cho bởi

$$h_n = \Theta(x_n)H$$

với một hàm liên tục từng khúc $\Theta(x)$ với $0 < \theta \leq \Theta(x) \leq 1$ trên $[a, b]$. Với kích thước bước được chỉ định theo cách này, chứng minh sự hội tụ có thể thay đổi dễ dàng để kết luận rằng sai số là $0(H^p) = 0(\tau^{1/p})$.

Công việc quan trọng nhất còn lại bây giờ là phải tìm các hàm Φ không "đắt tiền" khi đánh giá và thuộc cấp p với f trơn. Từ công thức (6.19) ta cần $\mu_n = 0(h^p)$. Khai triển Taylor của $y(x)$ chứng tỏ rằng

$$y(x_{n+1}) = y(x_n) + h\left[y'(x_n) + \ldots + \frac{h^{p-1}}{(p)!}y^{(p)}(x_n)\right] + \frac{h^{p+1}}{(p+1)!}y^{(p+1)}(\xi_n)$$

nếu $y(x) \in C^{p+1}[a, b]$. Vậy, ta tìm xem Φ, nếu phương pháp là cấp p, thì nó phải có

$$\Phi(x, y(x)) = y'(x) + \frac{h}{2!}y''(x) + \ldots + \frac{h^{p-1}}{(p)!}y^{(p)}(x) + \zeta(x),$$

với $\zeta(x) = 0(h^p)$. Vì $y(x)$ là nghiệm của phương trình vi phân $y'(x) = f(x, y(x)$ các đạo hàm của y có thể được biểu diễn nhờ đạo hàm toàn phần

của f. Dùng ký hiệu $f^{(m)}(x, y(x))$ để chỉ đạo hàm toàn phần cấp m của f và chỉ số dưới để chỉ đạo hàm riêng, hệ thức là

$$y^{(m)} = f^{(m)}(x, y(x)),$$

trong đó

$$\begin{aligned} f^{(1)} &= = f_x(x, y(x)) + f_y(x, y(x)) f(x, y(x)), \\ f^{(m)} &= = f_x^{(m-1)}(x, y(x)) + f_y^{(m-1)}(x, y(x)) f(x, y(x)), \quad m = 2, 3, \ldots \end{aligned}$$

Biểu thức cho $\Phi(x, y)$ trở thành

$$\Phi(x, y) = f(x, y) + \frac{h}{2!} f^{(1)}(x, y) + \ldots + \frac{h^{p-1}}{p!} f^{(p-1)}(x, y) + 0(h^p). \quad (6.22)$$

Một chọn lựa hiển nhiên cho Φ là hàm $T(x, y)$

$$T(x, y) = f(x, y) + \frac{h}{2!} f^{(1)}(x, y) + \ldots + \frac{h^{p-1}}{p!} f^{(p-1)}(x, y),$$

cung cấp một họ các phương pháp một bước, gọi là các *phương pháp chuỗi taylor* (Taylor series methods). Phương pháp Euler là trường hợp $p = 1$. Một khi có thể tính được các đạo hàm thì các phương pháp này rất có hiệu quả.

Các *phương pháp Runge-Kutta* dùng tổ hợp tuyến tính nhiều đánh giá của $f(x, y)$ để xấp xỉ $y(x)$. Trường hợp đơn giản nhất là phương pháp Euler chỉ dùng một đánh giá. Bây giờ ta thiết lập một thủ tục dùng hai đánh giá $f(x_n, y_n)$ và $f(x_n + p_1 h, y_n + p_2 h f(x_n, y_n))$, trong đó p_1 và p_2 là các tham số. Thì với Φ ta dùng tổ hợp tuyến tính $R(x, y)$:

$$R(x_n, y_n) = a_1 f(x_n, y_n) + a_2 f(x_n + p_1 h, y_n + p_2 h f(x_n, y_n)).$$

Trong biểu thức này ta tự do chọn các giá trị hữu dụng cho p_1, p_2, a_1, và a_2. Mục đích là chọn các tham số để cho biểu diễn (6.20) đúng với giá trị của p càng lớn càng tốt. Để thực hiện điều này ta khai triển tất cả các lượng trong chuỗi Taylor theo h và đồng nhất các hệ số của lũy thừa. Để đơn giản cách ký hiệu, các đối số được viết ra nếu chúng khác (x_n, y_n). Ta tiến hành như

sau.

$$\begin{aligned}
R &= a_1 f + a_1 f(x_n + p_1 h, y_n + p_2 hf) \\
&= a_1 f + a_2 \Big[f(x_n, y_n + p_2 hf) + p_1 h f_x(x_n, y_n + p_2 hf) \\
&\quad + \frac{p_1^2 h^2}{2} f_{xx}(x_n, y_n + p_2 hf) + 0(h^3) \Big] \\
&= a_1 f + a_2 \Big[f + p_2 hf f_y + \frac{p_2^2 h^2}{2} f^2 f_{yy} + 0(h^3) \\
&\quad + p_1 h f_x + p_1 p_2 h^2 f f_{xy} + 0(h^3) + \frac{p_1^2 h^2}{2} f_{xx} + 0(h^3) \Big] \\
&= (a_1 + a_2) f + a_2 h (p_2 f f_y + p_1 f_x) \\
&\quad + \frac{a_2 h^2}{2} (p_2^2 f^2 f_{yy} + 2 p_1 p_2 f f_{xy} + p_1^2 f_{xx}) + 0(h^3).
\end{aligned}$$

Bây giờ ta muốn chọn các tham số để cho

$$R = f + \frac{h}{2} f^{(1)} + \frac{h^2}{6} f^{(2)} + 0(h^3),$$

hay viết tường minh là

$$R = f + \frac{h}{2}(f f_y + f_x) + \frac{h^2}{6}(f^2 f_{yy} + 2 f f_{xy} + f_{xx} + f_x f_y + f f_y^2) + 0(h^3).$$

Cân bằng các hệ số lũy thừa của h cùng bậc, ta được

$$\begin{aligned}
a_1 + a_2 &= 1, \\
a_2 p_2 &= 1/2, \\
a_2 p_1 &= 1/2.
\end{aligned}$$

Lấy $a_2 = \alpha$ thì với giá trị bất kỳ của tham số α,

$$a_2 = \alpha, \quad a_1 = 1 - \alpha$$

cho công thức phù hợp với đẳng thức đầu. Hơn nữa, nếu đòi hỏi $\alpha \neq 0$, chọn

$$p_1 = p_2 = \frac{1}{2\alpha},$$

cho công thức phù hợp với hai đẳng thức cuối. Tóm lại,

$$R(x,y) = (1-\alpha) f(x,y) + \alpha f\left(x + \frac{h}{2\alpha}, y + \frac{h}{2\alpha} f(x,y)\right)$$

cho một họ các phương pháp một bước cấp 2 khi $\alpha \neq 0$ và f đủ trơn.

Một vài thành viên của họ công thức này có tên. Phương pháp Euler có $\alpha = 0$ và cấp $p = 1$. Phương pháp Heun (còn gọi là phương pháp Euler cải tiến) là trường hợp $\alpha = 1/2$, và phương pháp Euler điểm giữa (midpoint Euler method) hay phương pháp Euler hiệu chỉnh (modified Euler method) là trường hợp $\alpha = 1$. Để thấy khả năng áp dụng các công thức này ta cần biết điều kiện cần để định lý hội tụ có hiệu lực. Tính liên tục của R hiển nhiên được suy ra từ tính liên tục của f. Điều kiện Lipschitz trên R cũng rút ra từ f.

$$\begin{aligned}
|R(x,u) - R(x,v)| &= \left|(1-\alpha)[f(x,u) - f(x,v)] + \alpha\left[f\left(x + \frac{h}{2\alpha}, u + \frac{h}{2\alpha} f(x,u)\right)\right.\right. \\
&\qquad \left.\left. - f\left(x + \frac{h}{2\alpha}, v + \frac{h}{2\alpha} f(x,v)\right)\right]\right| \\
&\leq (1-\alpha)L|u-v| + |\alpha|L\left|u - v + \frac{h}{2\alpha}[f(x,u) - f(x,v)]\right| \\
&\leq (1-\alpha)L|u-v| + |\alpha|L|u-v| + \frac{h}{2}L^2|u-v| \\
&\leq \left[(1-\alpha) + |\alpha| + \frac{h}{2}\right] L|u-v|
\end{aligned}$$

với mọi $0 < h < h_0$, và ta có thể lấy hằng số Lipschitz cho R là

$$\left[(1-\alpha) + |\alpha| + \frac{h}{2}\right] L.$$

Vì vậy, nếu phương trình vi phân thỏa các điều kiện của định lý 6.1, và nếu hàm f có đạo hàm đến cấp 2 liên tục [như vậy nghiệm $y(x) \in C^3[a,b]$], thành viên bất kỳ của họ với $\alpha \neq 0$ hội tụ cấp 2.

Các thủ tục cấp cao bao gồm nhiều thay thế hơn có thể được thiết lập theo cùng một cách, mặc dù một cách tự nhiên các khai triển trở nên rất dài dòng và tẻ nhạt. Như xảy ra, thủ tục thuộc cấp p cần p đánh giá ở mỗi bước khi $p = 1, 2, 3, 4$ nhưng không như vậy khi $p = 5$. Vì lý do này, các công thức cấp bốn với kích thước bước hằng thường được dùng để tích phân số phương trình vi phân. Giống như trong trường hợp cấp hai, có một họ các thủ tục cấp bốn phụ thuộc nhiều tham số. Cách chọn cổ điển các tham số dẫn đến thuật toán

$$y_0 = A,$$

và khi $n = 0, 1, \ldots$

$$\begin{aligned} k_0 &= f(x_n, y_n), \\ k_1 &= f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_0\right), \\ k_2 &= f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right), \\ k_3 &= f(x_n + h, y_n + hk_2), \\ y_{n+1} &= y_n + \frac{h}{6}(k_0 + 2k_1 + 2k_2 + k_3). \end{aligned}$$

Đối với hệ phương trình vi phân cấp 1,

$$\begin{aligned} \mathbf{Y}' &= \mathbf{F}(x, \mathbf{Y}), \\ \mathbf{Y}(a) &= \mathbf{A}, \end{aligned}$$

một cách tự nhiên

$$\mathbf{Y}_0 = \mathbf{A},$$

và khi $n = 0, 1, \ldots$

thuật toán Runge-Kutta cổ điển là

$$\begin{aligned}
\mathbf{K}_0 &= \mathbf{F}(x_n, \mathbf{Y}_n), \\
\mathbf{K}_1 &= \mathbf{F}\left(x_n + \frac{h}{2}, \mathbf{Y}_n + \frac{h}{2}\mathbf{K}_0\right), \\
\mathbf{K}_2 &= \mathbf{F}\left(x_n + \frac{h}{2}, \mathbf{Y}_n + \frac{h}{2}\mathbf{K}_1\right), \\
\mathbf{K}_3 &= \mathbf{F}(x_n + h, \mathbf{Y}_n + h\mathbf{K}_2), \\
\mathbf{Y}_{n+1} &= \mathbf{Y}_n + \frac{h}{6}(\mathbf{K}_0 + 2\mathbf{K}_1 + 2\mathbf{K}_2 + \mathbf{K}_3).
\end{aligned}$$

Một thủ tục cấp bốn khác, hoàn toàn tương tự

$$\begin{aligned}
\mathbf{K}_0 &= \mathbf{F}(x_n, \mathbf{Y}_n), && (6.23) \\
\mathbf{K}_1 &= \mathbf{F}\left(x_n + \frac{h}{2}, \mathbf{Y}_n + \frac{h}{2}\mathbf{K}_0\right), \\
\mathbf{K}_2 &= \mathbf{F}\left(x_n + \frac{h}{2}, \mathbf{Y}_n + \frac{h}{4}\mathbf{K}_0 + \frac{h}{4}\mathbf{K}_1\right), \\
\mathbf{K}_3 &= \mathbf{F}(x_n + h, \mathbf{Y}_n - h\mathbf{K}_1 + 2h\mathbf{K}_2), && (6.24) \\
\mathbf{Y}_{n+1} &= \mathbf{Y}_n + \frac{h}{6}(\mathbf{K}_0 + 4\mathbf{K}_2 + \mathbf{K}_3).
\end{aligned}$$

6.4 Sai số địa phương và toàn cục

Các mã hiện nay cho bài toán giá trị đầu không dùng kích thước bước cố định. Sai số ở mỗi bước được đánh giá và h được điều chỉnh lại để nhận được xấp xỉ đủ chính xác. Có một nhầm lẫn đáng tiếc từ nhiều người dùng mã với đánh giá sai số về cái được đo và liên hệ của nó với sai số thực.

Hàm $y(x)$ ký hiệu nghiệm duy nhất của bài toán

$$\begin{aligned}
y' &= f(x, y), \\
y(a) &= A.
\end{aligned}$$

Sai số thực hay toàn tục tại x_{x+1} là

$$y(x_{n+1}) - y_{n+1}.$$

Nhưng tiếc là có khó khăn và tốn kém để đánh giá đại lượng này, vì trong bước tính x_{n+1} thủ tục số chỉ cung cấp x_n, y_n để đánh giá f. Nghiệm địa phương tại x_n là nghiệm $u(x)$

$$\begin{aligned} u' &= f(x,u), \\ u(x_n) &= y_n. \end{aligned}$$

Sai số địa phương là

$$u(x_{n+1}) - y_{n+1}.$$

Đây là sai số do xấp xỉ nghiệm phương trình vi phân gốc tại (x_n, y_n) bằng một bước. Sai số này được minh họa trên hình 6.1. Việc đòi hỏi thủ tục số giữ cho sai số này nhỏ là hợp lý. Sai số này ảnh hưởng lên sai số toàn cục phụ thuộc vào bản thân phương trình vi phân. Sau hết,

$$y(x_{n+1}) - y_{n+1} = [y(x_{n+1}) - u(x_{n+1})] + [u(x_{n+1}) - y_{n+1}]. \tag{6.25}$$

Đại lượng

$$y(x_{n+1}) - u(x_{n+1})$$

là số đo sự ổn định của phương trình vi phân vì nó là hậu quả (tại x_{n+1}) của sự sai biệt ban đầu $y(x_n) - y_n$ tại x_n. Nếu đại lượng này gia tăng ngày càng lớn, thì bài toán được đặt xấu hay điều kiện xấu hay không ổn định.

Thí dụ 6.6. Xét

$$y' = \alpha y$$

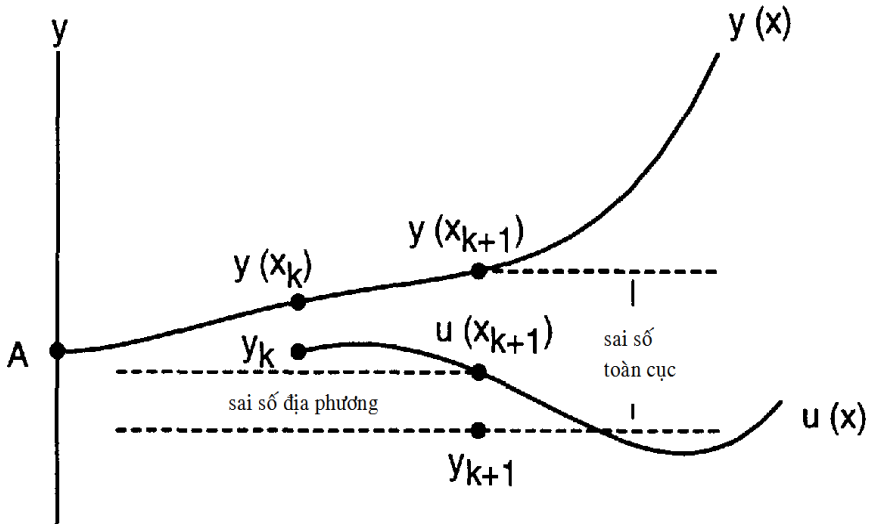
với α là hằng số. Ta có sau một số tính toán:

$$\begin{aligned} y(x) &= y(x_n)e^{\alpha(x-x_n)}, \\ u(x) &= y_n e^{\alpha(x-x_n)}; \end{aligned}$$

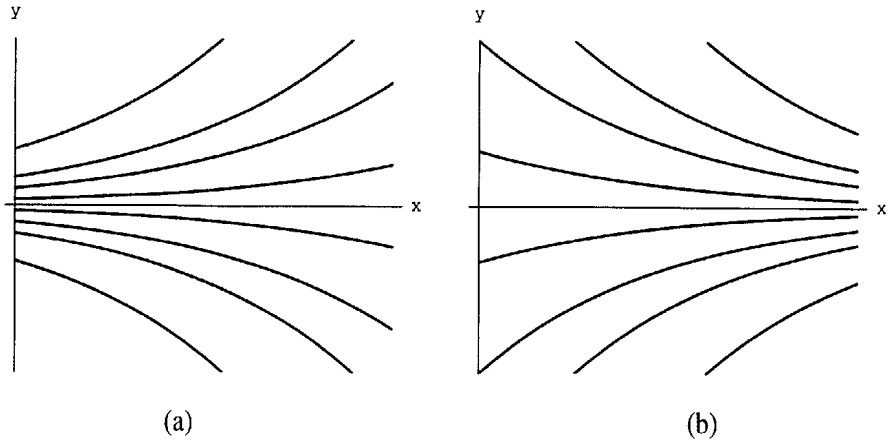
hơn nữa,

$$y(x_{n+1}) - u(x_{n+1}) = [y(x_n) - y_n]e^{\alpha h}. \tag{6.26}$$

Nếu $\alpha > 0$, các đường cong nghiệm trải rộng ra (Hình 6.2a), càng nhiều khi α lớn. Từ biểu thức (6.26) rõ ràng sai số địa phương nhỏ tại mỗi bước không cho sai số toàn cục nhỏ.Mặt khác, nếu $\alpha < 0$, các đường cong tụ vào nhau (Hình 6.2b) và (6.26) chứng tỏ rằng sự điều khiển sai số địa phương sẽ điều khiển sai số toàn cục. Với các hàm $f(x, y)$ tổng quát điều kiện Lipschitz không thể tiên đoán dáng điệu này, vì với thí dụ này hằng số Lipschitz là $|\alpha|$ trong cả hai trường hợp ∘



Hình 6.1: Sai số địa phương và sai số toàn cục.



Hình 6.2: Các đường cong nghiệm với: (a) $y' = 2y$; (b) $y' = -2y$.

Sai số địa phương liên hệ với sai số chặt cụt địa phương. Thật vậy, nó đúng bằng h lần sai số chặt cụt địa phương, μ, với nghiệm địa phương $u(x)$:

$$\begin{aligned} \text{sai số địa phương} &= u(x_{n+1}) - y_{n+1} \\ &= (y_n) + h\Phi(x_n, y_n) + h\mu_n) - y_{n+1} \\ &= h\mu_n. \end{aligned}$$

Chẳng hạn, khi y_n là nghiệm của $y' = f(x, y)$, ta đã thấy phương pháp Euler có

$$y(x_{n+1}) = y(x_n) + hf(x_n, y(x_n)) + \frac{h^2}{2}[f(x_n, y(x_n))f_y(x_n, y(x_n)) + f_x(x_n, y(x_n))] = 0(h^3).$$

Áp dụng cho $u(x)$, ta có

$$\text{sai số địa phương} = \frac{h^2}{2}(ff_y + f_x) + 0(h^3).$$

Tương tự với công thức Rung-Kutta cấp 2 ($\alpha \neq 0$), ta có

$$u(x_{n+1}) = y_n + h\left[f + \frac{h}{2}f^{(1)} + \frac{h^2}{6}f^{(2)}\right] + 0(h^4)$$

thì xấp xỉ số thỏa

$$\hat{y}_{n+1} = y_n + h\left[f + \frac{h}{2}(ff_y + f_x) + \frac{h^2}{8\alpha}(f^2 f_{yy} + 2ff_{xy} + f_{xx})\right] + 0(h^4).$$

Điều này dẫn đến

$$\text{sai số địa phương} = h\hat{\mu}_n = h^3\left(\frac{1}{6} - \frac{1}{8\alpha}\right)(f^2 f_{yy} + 2ff_{xy} + f_{xx}) + \frac{h^3}{6}(f_x f_y + ff_y^2) + 0(h^4).$$

Các biểu thức này đề nghị một cách đánh giá sai số địa phương. Giả sử ta tính y_{n+1} bằng phương pháp Euler và ta cũng tính một xấp xỉ nghiệm $\hat{y}_{n+1}$

bằng một trong các công thức Runge-Kutta cấp 2. Biểu thức trên chứng tỏ rằng

$$\hat{y}_{n+1} - y_{n+1} = \frac{h^2}{2}(ff_y + f_x) + 0(h^3) = h\mu_n + 0(h^3).$$

Nghĩa là, sự khác nhau giữa hai giá trị cho đánh giá sai số bằng công thức cấp thấp hơn. Điều này giống nguyên lý dùng trong chương 5 để đánh giá các sai số cầu phương. Nói chung, giả sử rằng thêm vào giá trị

$$y_{n+1} = y_n + h\Phi(x_n, y_n)$$

với sai số chặt cụt $\mu_n = 0(h^p)$, ta tính xấp xỉ khác

$$\hat{y}_{n+1} = y_n + h\hat{\Phi}(x_n, y_n)$$

với sai số chặt cụt $\hat{\mu}_n = 0(h^q)$ có cấp cao hơn, $q > p$. Thì bởi định nghĩa

$$u(x_{n+1}) = y_n + h\Phi(x_n, y_n) + h\mu_n = y_{n+1} + h\mu_n$$

và, tương tự,

$$u(x_{n+1}) = \hat{y}_{n+1} + h\hat{\mu}_n,$$

mà, bằng cách trừ nhau, chứng tỏ rằng

$$\hat{y}_{n+1} - y_{n+1} = h\mu_n - h\hat{\mu}_n = h\mu_n + 0(h^{q+1}).$$

Vì $h\hat{\mu}_n$ dần về không nhanh hớn $h\mu_n$, ta có thể đánh giá sai số địa phương bởi

$$\text{sai số địa phương} = h\mu_n \approx \hat{y}_{n+1} - y_{n+1}.$$

Ta muốn xấp xỉ nghiệm địa phương $u(x_{n+1})$. Vì sự kiện ta có một đánh giá sai số trong y_{n+1} tốt, tại sao không cố gắng cải thiện nó bằng cách loại bỏ sai số? Quá trình này, gọi là *ngoại suy địa phương* (local extrapolation), ở đây tương đương cách hình thức với việc đề xuất phép tích phân bằng xấp xỉ cấp cao hơn $\hat{y}_n$ bởi vì

$$u(x_{n+1} = y_{n+1} + h\mu_n \approx y_{n+1} + (\hat{y}_{n+1} - y_{n+1}) = \hat{y}_{n+1}.$$

Điều này bảo cho chúng ta rằng ngoại suy địa phương sẽ nâng cấp hiệu quả của cặp từ p lên q. Như vậy ta có thể nghĩ về điều đang xảy ra trong hai cách. Công thức cấp p đang được dùng với kết quả của nó được cải thiện nhờ ngoại suy địa phương. Công thức còn lại, cấp q đang được dùng với kích thước bước được chọn cách dè dặt nhờ đòi hỏi rằng bước được lấy với công

thức cấp thấp hơn p. Bởi vì ngoại suy địa phương gia tăng sự chính xác mà không gia tăng sự tốn kém, tất cả các mã sản xuất hiện nay dựa trên các phương pháp Runge-Kutta hiển đều dùng nó.

Công thức Runge-Kutta cấp 4 đòi hỏi (ít nhất) bốn đánh giá của $\mathbf{F}$ ở mỗi bước và một công thức cùng loại cấp 5 đòi hỏi ít nhất sáu. Đúng như cầu phương Gauss-Kronrod, thủ thuật có hiệu quả là phải thiết lập công thức như một cặp trong đó các đánh giá hàm được dùng trong cả hai công thức. R. England đã công bố một cặp công thức như vậy trong [3]. Để tiến từ x_n đến $x_n + h$, ông lấy bước độ dài $h/2$ với (6.24) để có kết quả cấp 4 $\mathbf{Y}_{n+1/2} \approx \mathbf{Y}(x_n + h/2)$ và rồi bước khác độ dài $h/2$ để có kết quả cấp 4 $\mathbf{Y}_{n+1} \approx \mathbf{Y}(x_n + h)$. Bằng cách thực hiện hai bước một nửa, ông ta có đủ các đánh giá hàm có hiệu lực mà với chỉ thêm một đánh giá, ông ta có thể lập một xấp xỉ cấp 5 $\hat{\mathbf{Y}}_{n+1}$ cho $\mathbf{Y}_{n+1}$. Bằng cách này, thêm một đánh giá hàm được thực hiện ở mỗi hai bước một nửa để có đánh giá sai số. Một đánh giá sai số được dùng để điều khiển sai số địa phương và như vậy cho sự tin cậy nào đó vào nghiệm tính toán. Nó cũng cho phép mã chọn lựa kích thước bước lớn nhất mà kết quả vẫn qua được sự kiểm tra sai số. Ngoại trừ các trường hợp không thông thường, sự thích ứng kích thước bước cho nghiệm theo cách này gia tăng tính hiệu quả của phép tích phân rất nhiều. Nó tương ứng với các sơ đồ cầu phương thích ứng của chương 5.

Công thức của England là như sau.

$$
\begin{aligned}
\mathbf{K}_0 &= \mathbf{F}(x_n, \mathbf{Y}_n),\\
\mathbf{K}_1 &= \mathbf{F}\left(x_n + \frac{h}{4}, \mathbf{Y}_n + \frac{h}{4}\mathbf{K}_0\right),\\
\mathbf{K}_2 &= \mathbf{F}\left(x_n + \frac{h}{4}, \mathbf{Y}_n + \frac{h}{8}(\mathbf{K}_0 + \mathbf{K}_1)\right),\\
\mathbf{K}_3 &= \mathbf{F}\left(x_n + \frac{h}{2}, \mathbf{Y}_n - \frac{h}{2}\mathbf{K}_1 + h\mathbf{K}_2\right),\\
\mathbf{Y}_{n+1/2} &= \mathbf{Y}_n + \frac{h}{12}(\mathbf{K}_0 + 4\mathbf{K}_2 + \mathbf{K}_3);\\
\mathbf{K}_4 &= \mathbf{F}\left(x_n + \frac{h}{2}, \mathbf{Y}_{n+1/2}\right),\\
\mathbf{K}_5 &= \mathbf{F}\left(x_n + \frac{3h}{4}, \mathbf{Y}_{n+1/2} + \frac{h}{4}\mathbf{K}_4\right),\\
\mathbf{K}_6 &= \mathbf{F}\left(x_n + \frac{3h}{4}, \mathbf{Y}_{n+1/2} + \frac{h}{8}(\mathbf{K}_4 + \mathbf{K}_5)\right),\\
\mathbf{K}_7 &= \mathbf{F}\left(x_n + h, \mathbf{Y}_{n+1/2} - \frac{h}{2}\mathbf{K}_5 + h\mathbf{K}_6\right),\\
\mathbf{Y}_{n+1} &= \mathbf{Y}_{n+1/2} + \frac{h}{12}(\mathbf{K}_4 + 4\mathbf{K}_6 + \mathbf{K}_7);\\
\mathbf{K}_8 &= \mathbf{F}\left(x_n + h, \mathbf{Y}_n + \frac{h}{12}(-\mathbf{K}_0 - 96\mathbf{K}_1 + 92\mathbf{K}_2 - 121\mathbf{K}_3 + 144\mathbf{K}_4 + \mathbf{K}_5 - 12\mathbf{K}_6\right),\\
\hat{\mathbf{Y}}_{n+1} &= \mathbf{Y}_n + \frac{h}{180}(14\mathbf{K}_0 + 64\mathbf{K}_2 + \mathbf{K}_3 - 8\mathbf{K}_4 + 64\mathbf{K}_6 + 15\mathbf{K}_7 - \mathbf{K}_8).
\end{aligned}
$$

Mặt không thuận lợi của các thuật toán giải bài toán giá trị đầu là chúng sinh ra một bảng các giá trị xấp xỉ trong khi nghiệm toán học $y(x)$ là

một hàm liên tục. Có thể xấp xỉ nghiệm cho mọi x bằng nội suy.

Câu hỏi và bài tập

6.1. Như một thí dụ không duy nhất nghiệm, kiểm tra rằng với hằng số c bất kỳ, $0 \le c \le b$, hàm $y(x)$ xác định bởi

$$y(x) = \begin{cases} 0, & \text{nếu } 0 \le x \le c \\ \frac{1}{4}(x-c)^2, & \text{nếu } c < x \le b \end{cases}$$

là một nghiệm của bài toán giá trị đầu

$$\begin{aligned} y' &= \sqrt{|y|} \\ y(0) &= 0. \end{aligned}$$

6.2. Xét bài toán

$$\begin{aligned} y' &= \sqrt{|1-y^2|} \\ y(0) &= 1. \end{aligned}$$

Kiểm tra rằng

(a) $y(x) = 1$ là nghiệm trên khoảng bất kỳ chứa $x = 0$,

(b) $y(x) = \cosh x$ là nghiệm trên $[0, b]$ với bất kỳ $b > 0$, và

(c) $y(x) = \cos x$ là nghiệm trên khoảng thích hợp.

Cái gì là khoảng lớn nhất chứa $x = 0$ trên đó $\cos x$ là nghiệm?

6.3. Dùng phương pháp Euler cho các bài toán sau bằng cách dùng kích thước bước cố định $h = 1.0$, và rồi $h = 0.5$. Trong mỗi trường hợp tính sai số tại $x = 1.0$.

(a) $y' = -y/(x+1)$ với $y(0) = 1$, vậy $y(x) = 1/(x+1)$.

(b) $y' = -y^3/2$ với $y(0) = 1$, vậy $y(x) = l/\sqrt{1+x}$.

6.4. Áp dụng phương pháp Euler để đánh giá nghiệm của bài toán giá trị đầu trong bài tập 6.3b. Dùng $h = 1/40$ và $h = 1/80$. Tính sai số tại $x = 0.5$ và $x = 1.0$ để thấy nếu chúng được chia đôi một cách thô như h là. Đánh giá xem h cần nhỏ bao nhiêu để sai số tuyệt đối nhỏ hơn 10^{-6} về độ lớn.

Hướng dẫn & Đáp số bài tập

Các bài tập trong tài liệu này nhằm giúp sinh viên tự kiểm tra kiến thức, hoặc bổ sung các điểm không được trình bày trong bài giảng. Sinh viên nên cố gắng tự giải các bài tập. Chỉ nên tham khảo lời giải có ở đây sau khi đã giải được (so sánh tìm cách giải tốt hơn); hoặc sau khi đã cố gắng nhiều lần nhưng không thành công.

Cuối cùng là một số chứng minh các kết quả (không được chứng minh đầy đủ) trong tài liệu.

Bài tập chương 1

1.1 Gọi N là chỉ số bắt đầu của thuật toán dùng công thức truy hồi lùi (1.4), ϵ là sai số tuyệt đối của $\hat{E}_N$, $\hat{E}_N - E_N = \epsilon$. Ta có:

$$\Delta E_{N-i} = \hat{E}_{N-i} - E_{N-i} = \frac{\epsilon}{N(N-1)\cdots(N-i+1)}.$$

Mặt khác, từ bất đẳng thức

$$0 < E_N < \frac{1}{N+1},$$

nếu lấy $E_{11} = 0$ thì $\epsilon < 10^{-1}$ và

$$\Delta E_5 = \frac{\epsilon}{11\cdot 10\cdot 9\cdot 8\cdot 7\cdot 6} < \frac{10^{-1}}{332640} < 10^{-6}.$$

Chú ý, trong phân tích này ta giả thiết các phép tính thực hiện trong thuật toán là chính xác!

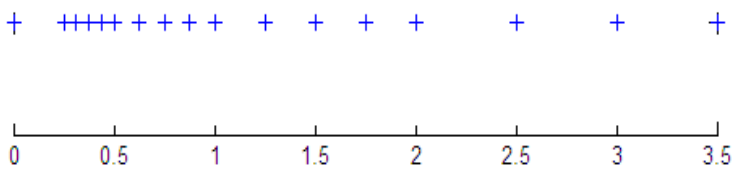
1.2 Hệ thống số dấu chấm động với $\beta = 2$, $s = 3$, $m = -1$, $M = 2$:

$$\pm 0.100 \times 2^{-1},\ \pm 0.101 \times 2^{-1},\ \pm 0.110 \times 2^{-1},\ \pm 0.111 \times 2^{-1}$$
$$\pm 0.100 \times 2^{0},\ \pm 0.101 \times 2^{0},\ \pm 0.110 \times 2^{0},\ \pm 0.111 \times 2^{0}$$
$$\pm 0.100 \times 2^{1},\ \pm 0.101 \times 2^{1},\ \pm 0.110 \times 2^{1},\ \pm 0.111 \times 2^{1}$$
$$\pm 0.100 \times 2^{2},\ \pm 0.101 \times 2^{2},\ \pm 0.110 \times 2^{2},\ \pm 0.111 \times 2^{2}$$

và 0.000×2^{-1}. Như vậy, hệ gồm 33 số.

Để biểu diễn trên trục số ta chuyển đổi chúng sang hệ thập phân:

	$\times 2^{-1}$	$\times 2^{0}$	$\times 2^{1}$	$\times 2^{2}$
0.100	0.25	0.5	1.0	2.0
0.101	0.3125	0.625	1.25	2.5
0.110	0.375	0.75	1.5	3.0
0.111	0.4375	0.875	1.75	3.5



Hình 6.3: Biểu diễn các số dương trên trục số.

Chú ý, từ biểu diễn trong hệ thập phân ta có thể kiểm tra lại đơn vị làm tròn là $u = 2^{1-3} = 0.25$.

1.3 Thuật toán chuyển đổi hệ cơ số 10 sang cơ số β

1. `read` a (hệ thập phân), β, MAX số chữ số tối đa (phần phân số hệ cơ số β)
2. Phân tích $a = b + c$, trong đó b là phần nguyên, c là phần phân số
3. Chuyển đổi phần nguyên b

   3.1 $q_{\text{old}} = b$, $k = 0$

3.2 `while` $q_{\text{old}} \sim= 0$
Thực hiện phép chia cho β: $q_{\text{old}} = q_{\text{new}} \cdot \beta + r$ (r là dư số)
$q_{\text{old}} = q_{\text{new}}$, $k = k + 1$, $B(k) = r$
`end`

4. Chuyển đổi phần phân số c

4.1 $p_{\text{old}} = c$, $k = 0$

4.2 `while` $p_{\text{old}} \sim= 0$ & $k <$ MAX
Thực hiện phép nhân cho β: $q_{\text{old}} \cdot \beta = p_{\text{new}} + s$ (s là phần nguyên)
$p_{\text{old}} = p_{\text{new}}$, $k = k + 1$, $C(k) = s$
`end`

5. Xuất kết quả

Chương trình viết bằng Matlab

chdoi.m

```
% function chuyen doi he co so 10 sang he co so beta
function s=chdoi(a,beta,MAX)
b=floor(a) % phan nguyen
c=a-b % phan phan so
% chuyen doi phan nguyen
k=0;
while b~=0
    k=k+1;
    B(k)=rem(b,beta);
    b=floor(b/beta);
end
% chuyen doi phan phan so
k=0;
while (c~=0)&(k<MAX)
    k=k+1;
    C(k)=floor(c*beta);
    c=c*beta-C(k);
end
% xuat ket qua
s='';
for k=1:length(B)
    s=strcat(s,int2str(B(length(B)-k+1)));
end
s=strcat(s,'.');
for k=1:length(C)
    s=strcat(s,int2str(C(k)));
```

```
end
```

Thuật toán chuyển đổi ngược lại là tương tự, nhưng các phép tính được thực hiện trong hệ thống số cơ số β, chương trình vì thế sẽ khác (dành cho sv).

1.4 Số nguyên dương n có biểu diễn trong hệ nhị phân là

$$n = (a_m \dots a_1 a_0)_2, \quad a_0, a_1, \dots, a_{m-1} \in \{0, 1\}, a_m = 1.$$

Vì $n = 2^m + a_{m-1}2^{m-1} + \dots + a_1 2^1 + a_0 \geq 2^m$ nên $m \leq [\log_2 n]$.

Mặt khác,

$$A^n = A^{2^m + a_{m-1}2^{m-1} + \dots + a_1 2^1 + a_0} = \underbrace{(A^{2^m})(A^{2^{m-1}})^{a_{m-1}} \cdots (A^2)^{a_1}(A^1)^{a_0}}_{m \text{ phép nhân ma trận}}.$$

Các ma trận trong dấu ngoặc được tính từ các tích:

$$A^2 = AA \to A^{2^2} = A^2 A^2 \to A^{2^3} = A^{2^2} A^{2^2} \to \dots \to A^{2^m} = A^{2^{m-1}} A^{2^{m-1}}.$$

Tất cả có m phép nhân ma trận. Tóm lại, để tính được A^n, ta cần nhiều lắm là $2m \leq 2[\log_2 n]$ phép nhân ma trận.

1.5 Trong hệ thống số dấu chấm động $\beta = 10$, $s = 2$, $m = -1$, $M = 2$, xét hai số

$$\begin{aligned} a &= 0.77 \times 10^{-1}, \\ b &= 0.79 \times 10^{-1}. \end{aligned}$$

Vì $a + b = 1.56 \times 10^{-1} = 0.156 \times 10^0$ nên $a \oplus b = 0.16 \times 10^0$ (làm tròn).
Vì $(a \oplus b)/2 = 0.08 \times 10^0 = 0.80 \times 10^{-1}$ nên $(a \oplus b) \oslash 2 = 0.80 \times 10^{-1} > b$.

1.6 a) Công thức tính φ (hàm ẩn), theo các thành phần của **u** và **v**, có thể viết cách hình thức là $\varphi = \varphi(u_1, u_2, v_1, v_2)$. Bằng cách lấy đạo hàm $\cos\varphi$ theo u_1, ta suy ra:

$$\frac{\partial \varphi}{\partial u_1} = -\frac{u_2(v_1 u_2 - u_1 v_2)}{\sin\varphi(u_1^2 + u_2^2)\sqrt{v_1^2 + v_2^2}}.$$

Ta lại có:

$$|\sin\varphi| = \sqrt{1-\cos^2\varphi} = \frac{|u_1v_2 - u_2v_1|}{\sqrt{u_1^2+u_2^2}\sqrt{v_1^2+v_2^2}},$$

nên

$$\left|\frac{\partial\varphi}{\partial u_1}\right| = \frac{|u_2|}{\sqrt{u_1^2+u_2^2}} \le 1.$$

Ta cũng có kết quả tương tự với u_2, v_1, v_2 (do tính đối xứng).

Từ công thức (1.17) ta có đánh giá

$$|\Delta\varphi| \le |\Delta u_1| + |\Delta u_2| + |\Delta v_1| + |\Delta v_2|.$$

Như vậy, tính φ từ các thành phần của **u** và **v** luôn là bài toán điều kiện tốt.

b) Số điều kiện (đối với u_1) của thuật toán tính theo công thức cho:

$$\kappa = \left|\frac{\partial\varphi}{\partial u_1}\right| \left|\frac{u_1}{\varphi}\right|$$

khi φ nhỏ là rất lớn, do đó thuật toán tính theo công thức này là không ổn định!

c) Dành cho sv.

1.7 *HD.* Công thức truy hồi tiến:

$$I_n = \frac{1}{4n} - \frac{I_{n-1}}{4} \quad \text{(ổn định)}.$$

Công thức truy hồi lùi:

$$I_{n-1} = \frac{1}{n} - 4I_n \quad \text{(không ổn định)}.$$

Bài tập chương 2

2.1

```
gauss_eli.m
function [x,flag]=gauss_eli(a,b)
% ham tra ve nghiem cua phuong trinh dstt a*x=b
% cu phap: [x,flag]=gauss_eli(a,b)
% flag=0 thanh cong; flag>0 he phuong trinh suy bien, dung
flag=0;
n=length(b);
% khu Gauss
for k=1:n-1
    [v,p]=max(abs(a(k:n,k)));
    tam=a(p+k-1,:);
    a(p+k-1,:)=a(k,:);
    a(k,:)=tam;
    if a(k,k)== 0
        flag=1;
        return;
    end
    for i=k+1:n
        t=a(i,k)/a(k,k);
        for j=1:n %j=k+1:n
            a(i,j)=a(i,j)-t*a(k,j);
        end
        b(i)=b(i)-t*b(k);
    end
end
if a(n,n)==0
    flag=1;
    return
end
% The nguoc
for i=n:-1:1
    x(i)=b(i);
    for j=i+1:n
        x(i)=x(i)-a(i,j)*x(j);
    end
    x(i)=x(i)/a(i,i);
end
if flag==1
    disp('suy bien');
end
```

2.2 Khử phương trình hai, nhân tử là $a(2,1) \oslash a(1,1) = 0.453$, ta được

$$\begin{aligned} 0.461x_1 + 0.311x_2 &= 0.150 \\ 0.001x_2 &= 0.001 \end{aligned}$$

Thế ngược: $x_2 = 1$, $x_1 = -0.349$. Sai số lớn!

2.3 Kết quả tính các thặng dư:

$$\mathbf{r} = \begin{bmatrix} 0.772 \times 10^{-3} \\ 0.35 \times 10^{-3} \end{bmatrix}, \quad \mathbf{s} = \begin{bmatrix} 0.1 \times 10^{-5} \\ -0.3 \times 10^{-5} \end{bmatrix}.$$

Xấp xỉ tốt hơn nhưng thặng dư không nhỏ hơn. Như vậy, việc xét thặng dư để kiểm tra độ chính xác của nghiệm là chưa đủ để kết luận.

2.4 a) Khử Gauss ma trận các hệ số nới rộng, cuối cùng ta thu được:

$$\left[\begin{array}{ccc|c} 1 & 1/2 & 1/3 & 1 \\ 0 & 1/12 & 1/12 & -1/2 \\ 0 & 0 & 1/180 & 1/6 \end{array}\right]$$

Thế ngược $x_3 = 30$, $x_2 = -36$, $x_1 = 9$ (nghiệm chính xác).

b) Nếu dùng biểu diễn thập phân chặt cụt 2-chữ số thì ma trận nới rộng của hệ là

$$\tilde{\mathbf{A}} \left[\begin{array}{ccc|c} 1 & 0.5 & 0.33 & 1 \\ 0.5 & 0.33 & 0.25 & 0 \\ 0.33 & 0.25 & 0.2 & 0 \end{array}\right]$$

c) Khử Gauss (không dùng phép xoay cục bộ)

$$\tilde{\mathbf{A}} \sim \left[\begin{array}{ccc|c} 1 & 0.5 & 0.33 & 1 \\ 0 & 0.08 & 0.09 & -0.5 \\ 0 & 0.09 & 0.1 & -0.33 \end{array}\right] \sim \left[\begin{array}{ccc|c} 1 & 0.5 & 0.33 & 1 \\ 0 & 0.08 & 0.09 & -0.5 \\ 0 & 0 & 0.001 & 0.22 \end{array}\right].$$

Thế ngược: $x_3 = 0.22 \times 10^3$, $x_2 = -0.25 \times 10^3$, $x_1 = 0.45 \times 10^2$.

d) Khử Gauss (dùng phép xoay cục bộ). Sau phép khử cột 1 (ở câu c)), hoán vị dòng 2 và 3:

$$\tilde{\mathbf{A}} \sim \left[\begin{array}{cccc} 1 & 0.5 & 0.33 & 1 \\ 0 & 0.09 & 0.1 & -0.33 \\ 0 & 0.08 & 0.09 & -0.5 \end{array}\right] \sim \left[\begin{array}{cccc} 1 & 0.5 & 0.33 & 1 \\ 0 & 0.09 & 0.10 & -0.33 \\ 0 & 0 & 0.2 \times 10^{-2} & -0.21 \end{array}\right].$$

Thế ngược: $x_3 = -0.10 \times 10^3$, $x_2 = 0,10 \times 10^3$, $x_1 = -0.16 \times 10^2$.

e) Nghiệm chính xác của b): $x_1 = 500/9$, $x_2 = -2500/9$, $x_3 = 2300/9$.

Sai số tính toán trong các trường c), d) và e)

	Δx_1	Δx_2	Δx_3
c) - a)	36	−214	190
d) - a)	−25	136	−130
e) - a)	46.5556	−241.7778	225.5556

2.5 Với số học thập phân ít hơn hay bằng 15-chữ số thì $10 \oplus 10^{18} = 10^{18}$! Ở đây ta dùng số học thập phân làm tròn 15-chữ số.

a) Dùng phép xoay cục bộ:

$$\begin{aligned} 10x_1 + 10^{18}x_2 &= 10^{18} \\ x_1 + x_2 &= 2 \end{aligned}$$

Khử Gauss:

$$\begin{aligned} 10x_1 + 10^{18}x_2 &= 10^{18} \\ -0.1 \times 10^{18}x_2 &= -0.1 \times 10^{18} \end{aligned}$$

Nghiệm $x_1 = 0,\ x_2 = 1$.

b) Chia mỗi dòng với $|a_{ij}|$ lớn nhất của nó, ta được hệ:

$$\begin{aligned} x_1 + x_2 &= 2 \\ (0.1 \times 10^{-16})x_1 + x_2 &= 1 \end{aligned}$$

Khử Gauss:

$$\begin{aligned} x_1 + x_2 &= 2 \\ (0.1 \times 10^{-16})x_1 + x_2 &= 1 \\ x_2 &= 1 \end{aligned}$$

Nghiệm: $x_1 = 1,\ x_2 = 1$.

c) ĐS. $x_1 = x_2 = 1$.

d) Thặng dư cho nghiệm tìm được ở các câu a), b), c):

$$\mathbf{r}_a = \begin{bmatrix} 1 \\ 10 \end{bmatrix}, \quad \mathbf{r}_b = \mathbf{r}_c = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Như vậy, nếu tính toán số dấu chấm động thì phương pháp ở câu b) tốt hơn phương pháp ở câu a). Trong trường hợp đang xét, thặng dư chỉ ra điều này.

e) dành cho sv.

2.9 Dùng Factor/Solve (Matlab)

```
>> clear all
>> A=[1 1 1; 1 1 0; 0 1 1]
A =
     1     1     1
     1     1     0
     0     1     1
>> b=[110;78.33;58.33]
b =
  110.0000
   78.3300
   58.3300
>> [A,flag,pivots,Cond] = Factor(A)
A =
     1     1     1
    -1     1     1
     0     0    -1
flag =
     0
pivots =
     1
```

```
     3
    -1
Cond =
     7
>> x = Solve(A,pivots,b)
x =
   51.6700
   26.6600
   31.6700
```

Nghiệm chính xác

```
x =
   5167/100
   1333/50
   3167/100
```

Kết quả giải bằng Factor/Solve cho kết quả chính xác!

2.9 Do $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$ nên ta có thể tìm các cột của ma trận $\mathbf{A}^{-1}$ bằng cách giải $\mathbf{A}\mathbf{x} = \mathbf{I}_i$, trong đó $\mathbf{I}_i$ là cột thứ i của ma trận đơn vị.

```
>> clear all
>> A=[1 2 3; 4 5 6; 7 8 9.01];
>> I1=[1;0;0];
>> I2=[0;1;0];
>> I3=[0;0;1];
>> [A,flag,pivots,Cond] = Factor(A)
A =
    7.0000    8.0000    9.0100
   -0.5714    0.8571    1.7129
   -0.1429   -0.5000   -0.0050
flag =
     0
pivots =
     3
     3
     1
Cond =
```

```
  1.4370e+004
>> B(:,1)=Solve(A,pivots,I1);
>> B(:,2)=Solve(A,pivots,I2);
>> B(:,3)=Solve(A,pivots,I3)
B =
   98.3333 -199.3333  100.0000
 -198.6667  399.6667 -200.0000
  100.0000 -200.0000  100.0000
```

Số điều kiện `Cond=1.4370e+004` quá lớn. Ma trận là điều kiện xấu. Kiểm

```
>> A*B
ans =
         0         0    1.0000
  -55.1905  113.9048  -57.2857
   84.7857 -170.3571   85.2143
```

`ans` không gần với ma trận đơn vị!

2.10 a)

```
>> clear all
>> A=[0.217 0.732 0.414; 0.508 0.809 0.376; 0.795 0.886 0.338];
>> b=[0.741; 0.613; 0.485];
>> [A,flag,pivots,Cond] = Factor(A)
A =
    0.7950    0.8860    0.3380
   -0.6390    0.4902    0.3217
   -0.2730   -0.4955    0.0006
flag =
     0
pivots =
     3
     3
     1
Cond =
  4.9409e+003
>> x = Solve(A,pivots,b)
```

```
x =
    0.0000
   -0.4160
    2.5254
```

b) Chuẩn của $\mathbf{A}$ và $\mathbf{b}$:

```
>> normA=norm(A,inf)
norm =
    2.0190
>> normb=norm(b,inf)
normb =
    0.7410
```

Như vậy,

$$\frac{\|\Delta\mathbf{A}\|}{\|\mathbf{A}\|} = \frac{3 \times 0.0005}{2.0190} = 7.4294 \times 10^{-4}, \quad \frac{\|\Delta\mathbf{b}\|}{\|\mathbf{b}\|} = \frac{0.0005}{0.7410} = 6.7476 \times 10^{-4}.$$

Theo bất đẳng thức (2.21), ta có

$$\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|} \lessapprox \texttt{Cond}(7.4294 \times 10^{-4} + 6.7476 \times 10^{-4}) = 7.0047 \approx 700\ \%.$$

Kết quả không đáng tin cậy.

c) dành cho sv.

Bài tập chương 3

3.3 Dùng Matlab

```
>> clear all
>> xn=[1 2];
>> fn=[2 4];
```

```
>> syms x
>> L1=(x-xn(2))/(xn(1)-xn(2))
L1 =
2 - x
>> L2=(x-xn(1))/(xn(2)-xn(1))
L2 =
x - 1
>> P2=fn(1)*L1+fn(2)*L2
P2 =
2*x
```

Có thể chọn $Q(x) = P_2(x) + (x-1)(x-2)$. Điều này không mâu thuẩn với tính duy nhất của đa thức nội suy. Vì theo chứng minh định lý 3.1 sự duy nhất hiểu theo nghĩa các đa thức bậc $\leq N-1$.

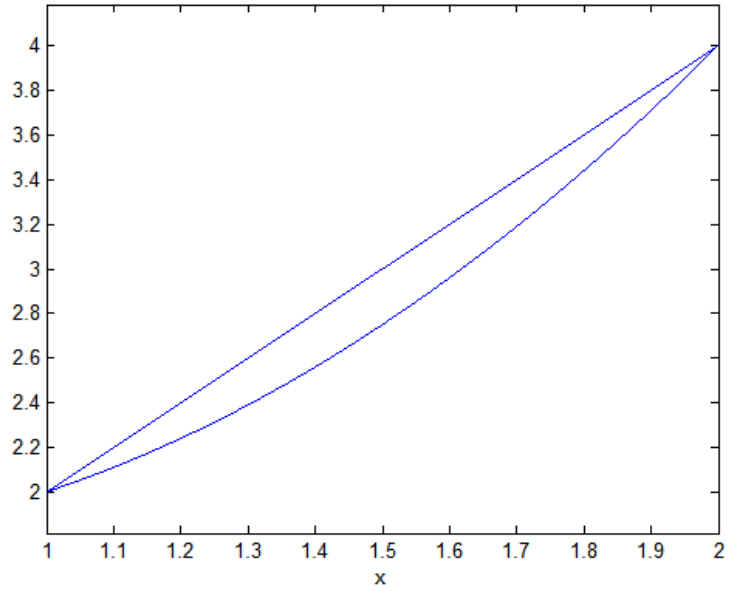
```
>> Q=simplify(P2+(x-xn(1))*(x-xn(2)))
Q =
x^2 - x + 2
>> ezplot(P2,[1 2])
>> hold on
>> ezplot(Q,[1 2])
>> hold off
```

3.4 Phương trình

$$P_N(x) = c_1 + c_2 x + \ldots + c_N x^{N-1}$$

có thể viết dưới dạng ma trận:

$$P_N(x) = [1 \; x \; x^2 \; \ldots \; x^{N-1}] \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{N-1} \end{bmatrix}.$$



Hình 6.4: Đồ thị hàm $P_2(x)$, $Q(x)$, bài tập 3.3.

Như vậy,

$$\underbrace{P_N(x_j) = [1 \; x_j \; x_j^2 \; \ldots \; x_j^{N-1}] \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{N-1} \end{bmatrix} = f(x_j),}_{\text{phương trình thứ } j} \quad j = 1, \ldots, N.$$

Hệ phương trình xác định các hệ số $c_1, \ldots, c_{N-1}$:

$$\underbrace{\begin{bmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^{N-1} \\ 1 & x_2 & x_2^2 & \ldots & x_2^{N-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_N & x_N^2 & \ldots & x_N^{N-1} \end{bmatrix}}_{\mathbf{M}} \underbrace{\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{N-1} \end{bmatrix}}_{\mathbf{c}} = \underbrace{\begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_N) \end{bmatrix}}_{\mathbf{f}}.$$

Thuật toán

1. Nhập dữ liệu: $xn = [x(1), x(2), \ldots, x(N)]$; $fn = [x(1), x(2), \ldots, x(N)]^T$.

2. Lập ma trận **M**

```
for i=1:N
    M(i,1)=1;
    for j=2:N
        M(i,j)=M(i,j-1)*xn(i);
    end
end
```

3. Giải phương trình $\mathbf{Mc} = \mathbf{f}$.

4. Xuất kết quả.

Kết quả của thuật toán là các hệ số $c_1, \ldots, c_{N-1}$. Các hệ số này hoàn toàn xác định một đa thức.

Chứng minh một số công thức

Chứng minh công thức (3.12), chương 3

$$\begin{aligned} \phi_k(x) &= [1 - 2L_k'(x_k)(x - x_k)]L_k^2(x), \\ \psi_k(x) &= (x - x_k)L_k^2(x). \end{aligned}$$

Từ các điều kiện của $\phi_k(x)$ và $\psi_k(x)$,

$$\phi_k(x) = \begin{cases} 0 & \text{nếu } j \neq k \\ 1 & \text{nếu } j = k, \end{cases} \qquad \phi_k'(x_j) = 0 \quad \text{với mọi } j$$

$$\psi_k'(x) = \begin{cases} 0 & \text{nếu } j \neq k \\ 1 & \text{nếu } j = k, \end{cases} \qquad \psi_k(x_j) = 0 \quad \text{với mọi } j,$$

ta nhận thấy, với mọi j ($j \neq k$), x_j là nghiệm của $\phi_k(x)$, $\phi_k'(x)$, $\psi_k(x)$, $\psi_k'(x)$. Do đó, ϕ_k và $\psi_k(x)$ đều chứa các nhân tử $(x - x_j)^2$; ngoài ra, $\psi_k(x_k) = 0$.

Như vậy, $\phi_k(x)$, $\psi_k(x)$ có dạng:

$$\phi_k(x) = A(x) \prod_{j=1, j\neq k}^{N} (x - x_j)^2,$$

$$\psi_k(x) = B(x)(x - x_k) \prod_{j=1, j\neq k}^{N} (x - x_j)^2.$$

Vì $\deg \phi_k, \deg \psi_k \leq 2N - 1$ nên $\deg A \leq 1$, $\deg B = 0$. Ta viết lại biểu thức của $\phi_k(x)$, $\psi_k(x)$:

$$\phi_k(x) = [c(x - x_k) + d] \prod_{j=1, j\neq k}^{N} (x - x_j)^2,$$

$$\psi_k(x) = e(x - x_k) \prod_{j=1, j\neq k}^{N} (x - x_j)^2,$$

trong đó c, d, e là hằng số, được xác định nhờ các đặc trưng còn lại của $\phi_k(x)$, $\psi_k(x)$.

1) $\phi_k(x_k) = 1$ suy ra

$$d = 1/ \prod_{j=1, j\neq k}^{N} (x_k - x_j)^2.$$

2) $\phi_k'(x_k) = 0$ suy ra (dùng kết quả trên):

$$c \prod_{j=1, j\neq k}^{N} (x_k - x_j)^2 + 2d \prod_{j=1, j\neq k}^{N} (x_k - x_j) \times \left[\frac{d}{dx} \left(\prod_{j=1, j\neq k}^{N} (x - x_j) \right) \right]_{x=x_k} = 0$$

$$c + \frac{2}{\prod_{j=1, j\neq k}^{N} (x_k - x_j)^2} \left[\frac{d}{dx} \left(\prod_{j=1, j\neq k}^{N} \frac{x - x_j}{x_k - x_j} \right) \right]_{x=x_k} = 0$$

$$c = -\frac{2L_k'(x_k)}{\prod_{j=1, j\neq k}^{N} (x_k - x_j)^2}.$$

Thay c và d vào biểu thức của $\phi_k(x)$ ta được kết quả.

3) $\psi_k'(x_k) = 1$ suy ra

$$e \prod_{j=1, j\neq k}^{N} (x_k - x_j)^2 = 1 \Rightarrow e = \frac{1}{\prod_{j=1, j\neq k}^{N} (x_k - x_j)^2}.$$

Thay e vào biểu thức của $\psi_k(x)$ ta được kết quả cần tìm (CMX).

Đề thi giữa kỳ

Năm 2008

Bài toán

Cho đường cong C với các điểm nút đã biết (x_i, y_i), $i = 1, 2, \ldots, N$. Bài toán: Xấp xỉ đường cong C.

Giải pháp

Dùng biểu diễn tham số của đường cong $(x(s), y(s))$ và xấp xỉ các hàm tọa độ $x(s)$, $y(s)$ một cách độc lập. Tham số s có thể chọn bất kỳ, nhưng

nên lấy s là độ dài cung. Sau khi đã chọn các nút s_i, $i = 1, 2, \ldots, N$, ta có thể nội suy $x(s)$ bằng spline $S_x(s)$; tương tự với $y(s)$, ta có nội suy $S_y(s)$. Bây giờ ta có đường cong $(S_x(s), S_y(s))$ xấp xỉ đường cong $(x(s), y(s))$.

Câu hỏi

1) Để bảo vệ tính liên tục của độ cong ta nên dùng spline bậc ba trơn. Vì sao ?

2) Trong trường hợp dữ liệu "thưa thớt" ta phải dùng spline như thế nào để có được đường cong theo yêu cầu ?

3) Trình bày thuật toán và viết chương trình bằng Matlab.

4) Áp dụng cho đường cong có tập các dữ liệu sau:
$(2.5, -2.5), (3.5, -0.5), (5, 2), (7.5, 4), (9.5, 4.5), (11.8, 3.5), (13, 0.5), (11.5, -2), (9, -3),$
$(6, -3.3), (2.5, -2.5), (0, 0), (-1.5, 2), (-3, 5), (-3.5, 9), (-2, 11), (0, 11.5), (2, 11), (3.5, 9), (3, 5),$
$(1.5, 2), (0, 0), (-2.5, -2.5), (-6, -3.3), (-9, -3), (-11.5, -2), (-13, 0.5), (-11.8, 3.5),$
$(-9.5, 4.5), (-7.5, 4), (-5, 2), (-3.5, -0.5), (-2.5, -2.5).$

Vẽ đường cong xấp xỉ.

HD. Công thức tính độ dài cung xấp xỉ: $s_1 = 0$,

$$s_{i+1} = s_i + \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}.$$

Đán án

Câu 1)-3) xem bài giảng. Để giải bài toán ta cần: (1) thuật toán xây dựng spline bậc ba; (2) thuật toán giải hệ ba đường chéo; (3) chương trình áp dụng spline bậc ba cho bài toán xấp xỉ đường cong theo tập các dữ liệu rời rạc.

4) Các hàm và chương trình áp dụng

trisolve.m

```
function [b]= trisolve(lline,dline,uline,b)
% TRISOLVE giai he ba duong cheo
% cu phap = trisolve(lline,dline,uline,b)
% input:
%            lline - duong cheo duoi
%            dline - duong cheo chinh
%            uline - duong cheo tren
%            b - ve phai
% output: b - nghiem
N=length(dline);
% khu
for i=1:N-1
```

```
    lline(i)=lline(i)/dline(i);
    dline(i+1)=dline(i+1)-lline(i)*uline(i);
end
% giai Ly = b bang phep the tien
for i=2:N
    b(i)=b(i)-lline(i-1)*b(i-1);
end
% giai Ux = y bang phep the lui
b(N)=b(N)/dline(N);
for i=N-1:-1:1
    b(i)=(b(i)-uline(i)*b(i+1))/dline(i);
end
```

spline_3.m

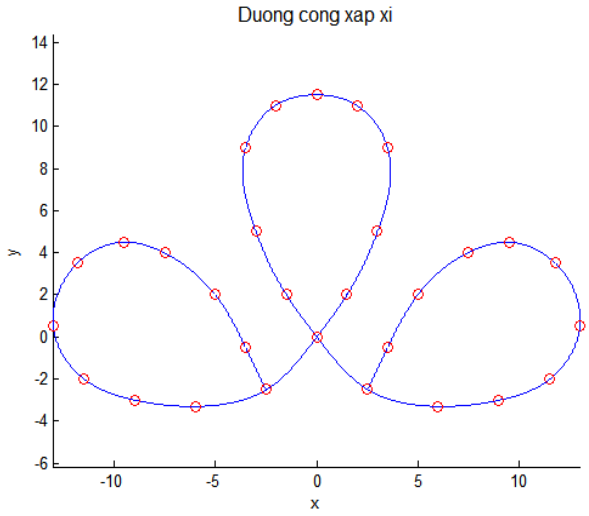
```
function s=spline_3(t,y)
% SPLINE_3 tra ve mang cac he so cua da thuc bac 3 tren cac khoang con
% cu phap: s = spline_3(t,y)
% input:
%       t: vector chua cac nut noi suy
%       y: vector chua cac gia tri ham noi suy
% output:
%       s: mang chua cac he so cua da thuc bac 3 tren cac khoang con
N=length(t);
s=zeros(N-1,4);
f=zeros(N-1,1);
k=1:N-1;
h=t(k+1)-t(k);
dy=(y(k+1)-y(k))./h(k);
% an=fn (cot 1)
s(:,1)=y(1:N-1);
% ma tran cac he so va vecto xac dinh cac cn (cot 3)
% lline, dline, uline la ba duong cheo
dline(1)=2*h(1);
uline(1)=h(1);
f(1)=0;
for i=2:N-2
    lline(i-1)=h(i-1);
    dline(i)=2*(h(i-1)+h(i));
    uline(i)=h(i);
    f(i)=3*(dy(i)-dy(i-1));
end
lline(N-2)=2*h(N-2);
```

```
dline(N-1)=3*h(N-1)+4*h(N-2);
f(N-1)=6*(dy(N-1)-dy(N-2));
s(:,3)=transpose(trisolve(lline,dline,uline,f));
% xac dinh dn (cot 4)
for i=1:N-2
    s(i,4)=(s(i+1,3)-s(i,3))/h(i)/3;
end
s(N-1,4)=(dy(N-1)-dy(N-2)-s(N-1,3)*(h(N-1)+2*h(N-2)/3) ...
    -s(N-2,3)*h(N-2)/3)/h(N-1)^2;
% xac dinh bn (cot 2)
for i=1:N-1
    s(i,2)=dy(i)-s(i,3)*h(i)-s(i,4)*h(i)^2;
end
```

dapan08.m

```
% chuong trinh dapan08.m
clear all
M=20; % so diem ve tren moi khoang con
% nhap du lieu (toa do cac diem)
A=[2.5 3.5 5 7.5 9.5 11.8 13 11.5 9 6 2.5 0 -1.5 -3 -3.5 -2 0 2 3.5 ...
    3 1.5 0 -2.5 -6 -9 -11.5 -13 -11.8 -9.5 -7.5 -5 -3.5 -2.5];
B=[-2.5 -0.5 2 4 4.5 3.5 0.5 -2 -3 -3.3 -2.5 0 2 5 9 11 11.5 11 9 5 ...
    2 0 -2.5 -3.3 -3 -2 0.5 3.5  4.5 4 2 -0.5 -2.5];
N=length(A);
% tham so duong cong (do dai cung)
t(1) = 0;
for i=1:N-1
    t(i+1)=t(i)+sqrt((A(i+1)-A(i))^2+(B(i+1)-B(i))^2);
end
% xap xi spline bac ba hoanh do va tung do duong cong
SX=spline_3(t,A);
SY=spline_3(t,B);
% xuat ket qua (ve duong cong xap xi)
figure(1)
hold on
for i=1:N-1
    u=linspace(t(i),t(i+1),M);
    s1=u-t(i);
    x=SX(i,1)+SX(i,2)*s1+SX(i,3)*s1.^2+SX(i,4)*s1.^3;
    y=SY(i,1)+SY(i,2)*s1+SY(i,3)*s1.^2+SY(i,4)*s1.^3;
    plot(A(i),B(i),'ro');
    plot(x,y);
```

```
end
title(['Duong cong xap xi'],'FontSize',12)
plot (A(N),B(N),'ro');
hold off
```



Hình 6.5: Đường cong xấp xỉ bằng spline bậc ba.

Chú thích. `linspace(x1,x2,N)` phát sinh `N` điểm ở giữa `x1` và `x2`. Khi `N<2`, `linspace` trả về `x2`. Lưu ý, `x1`, `x2` phải thuộc lớp float (double, single).

Năm 2009

Bài toán

Vận tốc $w(x, y)$ của dòng chảy dừng của chất lỏng nhớt trong đường ống tiết diện vuông $\Omega = (-1, 1) \times (-1, 1)$ là nghiệm của phương trình

$$1 + \frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} = 0 \quad \text{trong } \Omega, \tag{a}$$

thỏa điều kiện biên không trượt

$$w = 0 \quad \text{trên } |x| = 1 \text{ và } |y| = 1. \tag{b}$$

Một trong các phương pháp số giải bài toán này là phương pháp sai phân hữu hạn. Theo cách tiếp cận này, $w(x, y)$ được xấp xỉ chỉ trên *lưới*, gồm

các điểm nằm trong và trên biên Ω) là giao điểm của các đường tọa độ trong hệ tọa độ Descartes chọn trước. Chẳng hạn, nếu cho N là số nguyên dương và $h = 1/N$ là *bước lưới* ta có lưới gồm các điểm (ih, jh) với $i, j = -N, -N+1, \ldots, N-1, N$. Ký hiệu $w_{ij} = w(ih, jh)$.

Công thức sai phân

Cho hàm $u(x)$ và h là bước lưới trên trục x. Sai phân tiến (tương ứng, lùi) cấp một của hàm u tại x với bước h, bởi định nghĩa, là

$$\Delta u(x) = u(x+h) - u(x) \text{ (tương ứng, } \nabla u(x) = u(x) - u(x-h)).$$

Từ công thức khai triển Taylor ta có thể xấp xỉ đạo hàm cấp một của u tại x bằng công thức sai phân lùi

$$\frac{du}{dx}(x) \approx \frac{\nabla u(x)}{h} = \frac{u(x) - u(x-h)}{h}.$$

Để tính xấp xỉ đạo hàm cấp hai, dùng sai phân tiến

$$\frac{d^2u}{dx^2} \approx \frac{1}{h}\Delta\left(\frac{\nabla u(x)}{h}\right) = \frac{u(x+h) - 2u(x) + u(x-h)}{h^2}. \qquad (c)$$

Câu hỏi

1) Dùng công thức (c) xấp xỉ phương trình (a) và điều kiện (b) của bài toán biên. Thiết lập hệ phương trình đại số tuyến tính xác định các giá trị nút w_{ij}.

2) Viết thuật toán giải bài toán biên.

3) Viết chương trình giải số bài toán (nộp sau 01 ngày).

Đáp án

1) Điều kiện biên cho ($8N$ phương trình):

$$w_{ij} = 0 \quad \text{khi } i = \pm N, j = \pm N. \qquad (1)$$

Dùng công thức Taylor, ta có:

$$\frac{w_{i+1,j} - 2w_{ij} + w_{i-1,j}}{h^2} = \frac{\partial^2 w}{\partial x^2}(ih, jh) + 0(h^2).$$

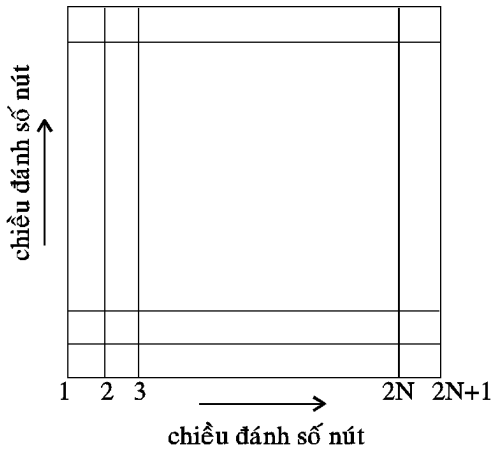
Như vậy, phương trình được xấp xỉ:

$$w_{i,j-1} + w_{i+1,j} - 4w_{i,j} + w_{i-1,j} + w_{i,j+1} = -h^2, \quad -N < i, j < N. \qquad (2)$$

Số phương trình là $(2N-1)^2$. Tổng cộng có $(2N+1)^2$ phương trình.

Để thiết lập hệ phương trình đại số tuyến tính ta cần đánh số các nút (i, j). Quy tắc đánh số từ trái sang phải, từ dưới lên trên, hình 6.6. Theo quy tắc này, thứ tự của nút (i, j) là $k = (2N+1)(j+N) + i + N + 1$; nghĩa là

$$(i, j) \leftrightarrow k = (2N+1)(j+N) + i + N + 1. \qquad (3)$$



Hình 6.6: Quy tắc đánh số nút.

Ký hiệu $q_k = w_{ij}$, $k = 1, 2, \ldots, 2N+1$. Từ (3), ta có:

$$\begin{aligned}
(i, j-1) &\rightarrow (2N+1)(j-1+N) + i + N + 1,\\
(i+1, j) &\rightarrow (2N+1)(j+N) + i + N + 2,\\
(i, j) &\rightarrow (2N+1)(j+N) + i + N + 1,\\
(i-1, j) &\rightarrow (2N+1)(j+N) + i + N,\\
(i, j+1) &\rightarrow (2N+1)(j+1+N) + i + N + 1.
\end{aligned}$$

Nếu đặt $k = (2N+1)(j+N) + i + N + 1$ (tương ứng với nút (i, j)) thì

$$\begin{aligned}
(i, j-1) &\rightarrow k - (2N+1),\\
(i+1, j) &\rightarrow k + 1,\\
(i-1, j) &\rightarrow k - 1,\\
(i, j+1) &\rightarrow k + (2N+1).
\end{aligned}$$

các phương trình (2) được viết lại (có sắp xếp):

$$q_{k-(2N+1)}+q_{k-1}-4q_k+q_{k+1}+q_{k+(2N+1)} = -h^2, \quad 2N+3 \le k \le 4N^2-4N+1,$$

và k không là các nút nằm trên biên. Ở các nút này, phương trình (1) cho: $q_k = 0$.

Tóm lại, ta có hệ phương trình đại số tuyến tính

$$\mathbf{Aq} = \mathbf{B},$$

trong đó $\mathbf{A} \in \mathrm{Mat}_{(2N+1)^2}(\mathbb{R})$, $\mathbf{B}, \mathbf{q} \in \mathrm{Mat}_{(2N+1)^2\times 1}(\mathbb{R})$,

$$\mathbf{q} = [q_1, q_2, \ldots, q_{(2N+1)^2}]^T.$$

2) Thuật toán

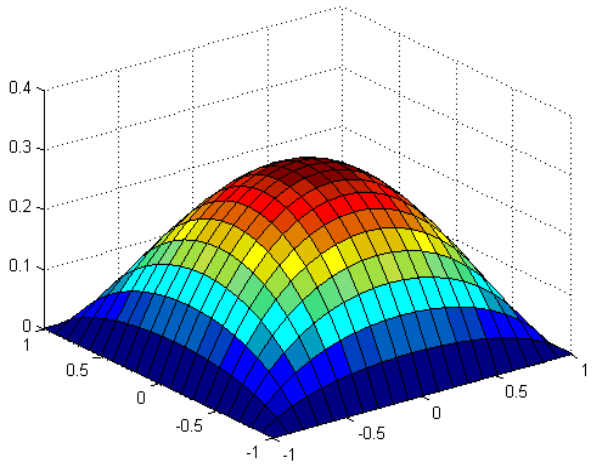
```
% nhập dữ liệu
read  N
h=1/N
% khởi tạo ma trận A, vectơ B
A=zeros((2*N+1)^2)
B=zeros((2*N+1)^2,1)
% tính ma trận A, vectơ B
for i=-N:N
     for j=-N:N
            k=(2*N+1)*(j+N)+i+N+1;
            if abs(i)==N or abs(j)==N
                 A(k,k)=1;
                 B(k)=0;
            else
                 A(k,k-(2*N+1))=1;
                 A(k,k-1)=1;
                 A(k,k)=-4;
                 A(k,k+1)=1;
                 A(k,k+(2*N+1))=1;
                 B(k)=-h^2;
            end
     end
end
% giải phương trình đại số tuyến tính
q=inv(A)*B;
% xuất kết quả
```

3) Chương trình

dapan09.m

```
% chuong trinh dapan09.m giai de thi giua ky 2009
% Trinh Anh Ngoc
% 24/10/2009
clear all
% nhap du lieu
N=10;
h=1/N;
% khoi tao ma tran A, vecto B
A=zeros((2*N+1)^2);
B=zeros((2*N+1)^2,1);
% tnh ma tran A, vecto B
for i=-N:N
     for j=-N:N
            k=(2*N+1)*(j+N)+i+N+1;
            if abs(i)==N|abs(j)==N
                 A(k,k)=1;
                 B(k)=0;
            else
                 A(k,k-(2*N+1))=1;
                 A(k,k-1)=1;
                 A(k,k)=-4;
                 A(k,k+1)=1;
                 A(k,k+(2*N+1))=1;
                 B(k)=-h^2;
            end
     end
end
% giai phuong trinh dai so tuyen tinh
q=inv(A)*B;
% xuat ket qua (ve do thi)
x=-1:h:1;
y=-1:h:1;
for j=-N:N
    for i=-N:N
        k=(2*N+1)*(i+N)+j+N+1;
        W(j+N+1,i+N+1)=q(k);
    end
end
```

```
surf(x,y,W);
```



Hình 6.7: Đồ thị hàm $w(x, y)$.

Chú thích:

1. `surf(x,y,z)` vẽ mặt tham số. Nếu `x` và `y` là vectơ, `length(x) = n` và `length(Y) = m`, trong đó `[m,n] = size(z)`. Thì các đỉnh của mặt là bộ ba `(x(j), y(i), z(i,j))`.

2. *Phương pháp giải lặp*

   Viết lại phương trình

   $$w_{i,j} = (h^2 + w_{i,j-1} + w_{i+1,j} + w_{i-1,j} + w_{i,j+1})/4.$$

   Phương pháp lặp Jacobi, xấp xỉ liên tiếp nghiệm bài toán bằng cách tính

   $$w_{i,j}^{(k+1)} = (h^2 + w_{i,j-1}^{(k)} + w_{i+1,j}^{(k)} + w_{i-1,j}^{(k)} + w_{i,j+1}^{(k)})/4.$$

   với mọi i, j. Đây là phương pháp lặp rất đơn giản và không "tốn kém", chỉ cần lưu trữ $w_{i,j}$ hiện hành và $w_{i,j}$ tiếp sau.

Còn tiếp

Tài liệu tham khảo

[1] M. Abramowitz and I. Stegun, eds., *Handbook of Mathematical Functions,* M. Dover, Mineola, N.Y., 1964.

[2] Birkhoff G. and Priver A., Hermite interpolation errors for derivatives, *J. Math. and Physics,* 46(1967), pp. 440-447.

[3] R. England, Error estimates for Runge-Kutta type solutions to systems of ordinary differential equations, *Computer Journal,* 12 (1969), pp. 166-170.

[4] Ferguson J. and Miller K., Characterization of shape in a class of third degree algebraic curves, TRW Report 5322-3-5, 1969.

[5] Fritsch F. and Butland J., A method for constructing locac monotone piecewise cubic interpolants, *SIAM J. Sci. Stat. Comp.,* 5(1984), pp. 300-304.

[6] Fritsch F. and Carlson R., Monotone piecewise cubic interpolation, *SIAM J. Numer Anal.,* 17(1980), pp. 238-246.

[7] *Handbook of Chemistry and Physics*, 63rd ed., CRC Press, Cheveland, 1982-1983.

[8] Isaacson E. and Keller H., *Analysis of numerical Methods,* Dover, Mineola, N.Y., 1994.

[9] Đặng Văn Liệt, *Giải tích số*, NXB ĐHQG TP. HCM, 2004.

[10] *Getting Started with MATLAB*, MathWorks, Inc., 1998.

[11] Powell M.J.D., On the maximum errors of polynomial approximation defined by interpolation and by least squares criteria, *Comp. J.,* 9(1967), pp. 404-407.

[12] *Symbolic Math Toolbox User's Guide*, MathWorks, Inc., 1998.

[13] L. F. Shampine, R. C. Allen, Jr., S. Pruess, *Fundamentals of numerical computing*, John Wiley & Sons, Inc., 1997.

[14] Won Y. Yang, Wenwu Cao, Tae S. Chung, John Morris, *Applied numerical methods using MATLAB,* John Wiley & Sons, Inc., Hoboken, New Jersey, 2005.

Mục lục