ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Đề tài: PHƯƠNG PHÁP GIẢI GẦN ĐÚNG PHƯƠNG TRÌNH PHI TUYẾN

CHUYÊN ĐỀ ĐẠI SỐ

Đại học Quốc gia Hà Nội
Đại học Khoa học Tự nhiên
Khoa Toán-Cơ-Tin học

Giảng viên: TS. Trần Hiếu

Học viên: Trần Thế Khải - 23007943
Nguyễn Thị Đông - 21007930
Vũ Thảo - 2300xxx

Mục lục

Giới thiệu

1 Nghiệm của phương trình một ẩn

Trong nhiều lĩnh vực khoa học và kỹ thuật, việc giải phương trình phi tuyến đóng vai trò quan trọng. Một ví dụ tiêu biểu là mô hình tăng trưởng dân số, trong đó tốc độ tăng trưởng tỉ lệ thuận với quy mô hiện tại của quần thể. Khi bổ sung thêm yếu tố nhập cư với tốc độ hằng, ta thu được phương trình vi phân phức tạp hơn. Xét trường hợp một cộng đồng có dân số ban đầu $N(0) = 1{,}000{,}000$, sau một năm có thêm 435,000 người nhập cư và tổng dân số đạt $N(1) = 1{,}564{,}000$. Để xác định tỉ lệ sinh λ, ta cần giải phương trình

$$1{,}564{,}000 \;=\; 1{,}000{,}000e^{\lambda} + \frac{435{,}000}{\lambda}\big(e^{\lambda} - 1\big).$$

Rõ ràng phương trình này không thể giải chính xác bằng các phương pháp đại số thông thường. Trong những tình huống như vậy, các kỹ thuật số trở thành công cụ thiết yếu, cho phép tìm nghiệm xấp xỉ với độ chính xác tùy ý.

Mục tiêu của Chương 2 là trình bày và phân tích một số phương pháp số cơ bản để giải các phương trình một biến, bao gồm *phương pháp chia đôi*, *lặp điểm cố định*, *Newton–Raphson* và *Secant*. Những phương pháp này không chỉ minh họa cách tiếp cận bài toán nghiệm phi tuyến, mà còn cho thấy sức mạnh của tính toán số trong việc giải quyết các phương trình mà giải tích không xử lý được.

1.1 Phương pháp chia đôi (The Bisection Method)

Một trong những kỹ thuật cơ bản nhất để giải phương trình phi tuyến $f(x) = 0$ là *phương pháp chia đôi* (Bisection Method). Ý tưởng dựa trên *Định lý Giá trị Trung gian*: nếu f liên tục trên khoảng $[a, b]$ và $f(a)f(b) < 0$, thì tồn tại ít nhất một nghiệm $p \in (a, b)$ sao cho $f(p) = 0$.

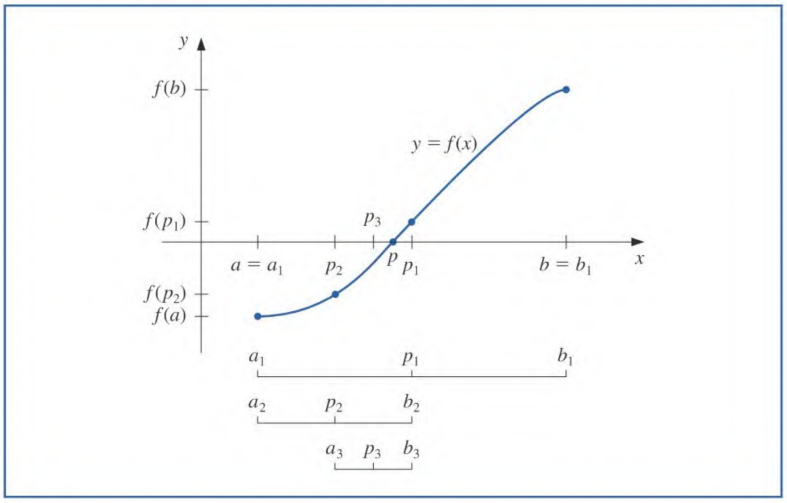
Phương pháp tiến hành như sau: bắt đầu với khoảng $[a, b]$, ta tính trung điểm

$$p = \frac{a + b}{2}.$$

Nếu $f(p) = 0$, nghiệm đã được tìm thấy. Ngược lại, tùy thuộc vào dấu của $f(p)$, ta chọn nửa khoảng chứa nghiệm:

$$[a, p] \quad \text{nếu } f(a)f(p) < 0, \qquad \text{hoặc} \qquad [p, b] \quad \text{nếu } f(p)f(b) < 0.$$

Quy trình này được lặp lại cho đến khi khoảng có độ dài nhỏ hơn sai số cho phép TOL.



Hình 1.1: Minh họa phương pháp chia đôi (Figure 2.1).

Thuật toán (Bisection Method)

(1) Chọn a, b sao cho $f(a)f(b) < 0$, cùng số lần lặp tối đa N và sai số TOL.

(2) Với $i = 1, 2, \ldots, N$:

  (a) Tính $p = \frac{a+b}{2}$.

  (b) Nếu $|f(p)| < \text{TOL}$ hoặc $\frac{b-a}{2} < \text{TOL}$, kết thúc và nhận p là nghiệm gần đúng.

  (c) Nếu $f(a)f(p) > 0$, đặt $a = p$; ngược lại, đặt $b = p$.

Nếu sau N bước mà chưa đạt điều kiện dừng, thuật toán được xem là thất bại.

Ví dụ 1: Phương pháp chia đôi

Chứng minh rằng phương trình

$$f(x) = x^3 + 4x^2 - 10 = 0$$

có nghiệm trong khoảng $[1, 2]$, và sử dụng phương pháp chia đôi để tìm nghiệm xấp xỉ với độ chính xác ít nhất 10^{-4}.

Lời giải. Vì $f(1) = -5 < 0$ và $f(2) = 14 > 0$, nên theo Định lý Giá trị Trung gian, tồn tại ít nhất một nghiệm trong $[1, 2]$.

Ở lần lặp đầu tiên, $p_1 = 1.5$ và $f(1.5) = 2.375 > 0$, do đó nghiệm thuộc khoảng $[1, 1.5]$. Tiếp tục chia đôi, $p_2 = 1.25$ với $f(1.25) = -1.7969 < 0$, nên nghiệm nằm trong $[1.25, 1.5]$. Lặp lại quá trình này, ta thu được các giá trị trong Bảng 1.1.

Bảng 1.1: Các bước của phương pháp chia đôi cho $f(x) = x^3 + 4x^2 - 10$

n	a_n	b_n	p_n	$f(p_n)$
1	1.0000	2.0000	1.5000	2.3750
2	1.0000	1.5000	1.2500	-1.7969
3	1.2500	1.5000	1.3750	0.1621
4	1.2500	1.3750	1.3125	-0.8484
5	1.3125	1.3750	1.3438	-0.3510
6	1.3438	1.3750	1.3594	-0.0964
7	1.3594	1.3750	1.3672	0.0324
8	1.3594	1.3672	1.3633	-0.0322
9	1.3633	1.3672	1.3652	0.000072
10	1.3633	1.3652	1.3643	-0.01605
11	1.3643	1.3652	1.3647	-0.00799
12	1.3647	1.3652	1.3650	-0.00396
13	1.3650	1.3652	1.3651	-0.00194

Sau 13 lần lặp, ta được nghiệm xấp xỉ

$$p_{13} = 1.365112305,$$

với sai số

$$|p - p_{13}| < |p_{14} - p_{13}| = |1.365234375 - 1.365112305| = 0.00012207,$$

nên đảm bảo chính xác ít nhất 10^{-4}. Giá trị đúng của nghiệm (chính xác đến 9 chữ số thập phân) là

$$p = 1.365230013.$$

Định lý

Giả sử f là hàm liên tục trên đoạn $[a, b]$ và thỏa $f(a)f(b) < 0$. Khi áp dụng phương pháp chia đôi, ta thu được một dãy $\{p_n\}_{n=1}^{\infty}$ hội tụ về nghiệm p của $f(x) = 0$. Hơn nữa, sai số tại bước lặp thứ n được ước lượng bởi

$$|p_n - p| < \frac{b-a}{2^n}, \quad n \geq 1.$$

Hệ quả. Dãy $\{p_n\}$ hội tụ về p với tốc độ $O(2^{-n})$, tức là

$$p_n = p + O(2^{-n}).$$

Ví dụ 2: Ước lượng số lần lặp

Xác định số lần lặp cần thiết để giải phương trình

$$f(x) = x^3 + 4x^2 - 10 = 0$$

với độ chính xác 10^{-3}, sử dụng $a_1 = 1$ và $b_1 = 2$.

Lời giải. Theo Định lý, sai số sau N lần lặp thỏa

$$|p_N - p| \leq \frac{b-a}{2^N}.$$

Với $a = 1, b = 2$, ta có

$$|p_N - p| \leq \frac{1}{2^N}.$$

Để đạt độ chính xác 10^{-3}, cần

$$\frac{1}{2^N} \leq 10^{-3} \quad \Rightarrow \quad N \geq \frac{3}{\log_{10} 2} \approx 9.96.$$

Vậy cần ít nhất $N = 10$ lần lặp để đảm bảo nghiệm gần đúng nằm trong sai số 10^{-3}.

1.2 Phương pháp lặp điểm cố định (Fixed-Point Iteration)

Định nghĩa Một số p được gọi là *điểm cố định* của một hàm g nếu

$$g(p) = p.$$

Khái niệm điểm cố định xuất hiện trong nhiều lĩnh vực toán học và là một công cụ quan trọng cho các nhà kinh tế khi chứng minh các kết quả liên quan đến cân bằng. Thuật ngữ "điểm cố định" được giới thiệu bởi nhà toán học người Hà Lan **L. E. J. Brouwer (1882–1966)** vào đầu những năm 1900.

Trong phần này, chúng ta xét bài toán tìm điểm cố định của một hàm và mối quan hệ của nó với bài toán tìm nghiệm của phương trình phi tuyến. Hai bài toán này là tương đương theo nghĩa sau:

- Cho một bài toán tìm nghiệm $f(p) = 0$, có thể xác định một hàm g sao cho p là điểm cố định, ví dụ:

  $$g(x) = x - f(x), \quad \text{hoặc} \quad g(x) = x + \lambda f(x),$$

  với λ thích hợp.

- Ngược lại, nếu g có một điểm cố định tại p, thì hàm

  $$f(x) = x - g(x)$$

  có một nghiệm tại p.

Mặc dù thường thì chúng ta quan tâm đến việc giải phương trình $f(x) = 0$, nhưng việc đưa về dạng điểm cố định thường thuận lợi hơn cho việc phân tích, và trong nhiều trường hợp, sự lựa chọn thích hợp của hàm g có thể đem lại những thuật toán tìm nghiệm hiệu quả hơn.

Để gần đúng một điểm cố định của một hàm g, ta bắt đầu với một giá trị gần đúng ban đầu p_0 và xác định một dãy $\{p_n\}$ theo công thức lặp

$$p_n = g(p_{n-1}), \quad n \geq 1.$$

Nếu dãy $\{p_n\}$ hội tụ và g liên tục, thì

$$p = \lim_{n\to\infty} p_n = g(p).$$

Do đó p là một nghiệm của phương trình điểm cố định $x = g(x)$. Phương pháp này được gọi là *lặp điểm cố định (fixed-point iteration)* hoặc *lặp hàm (functional iteration)*.

Thuật toán – Lặp điểm cố định

INPUT: giá trị gần đúng ban đầu p_0, sai số cho phép `TOL`, số lần lặp tối đa N_0.

OUTPUT: nghiệm xấp xỉ p, hoặc thông báo thất bại.

(1) Đặt $i = 1$.

(2) Trong khi $i \leq N_0$, thực hiện:

  (a) Tính $p = g(p_0)$.

  (b) Nếu $|p - p_0| < \texttt{TOL}$, thì xuất p và **STOP**.

  (c) Tăng $i = i + 1$.

  (d) Đặt $p_0 = p$.

(3) Nếu chưa hội tụ sau N_0 bước, xuất thông báo: "Phương pháp thất bại sau N_0 lần lặp" và **STOP**.

Minh hoạ

Xét phương trình

$$x^3 + 4x^2 - 10 = 0,$$

phương trình này có một nghiệm duy nhất trong khoảng $[1, 2]$.

Có nhiều cách biến đổi phương trình này sang dạng điểm cố định $x = g(x)$ bằng các phép biến đổi đại số đơn giản. Ví dụ:

$$
\begin{aligned}
g_1(x) &= x - (x^3 + 4x^2 - 10),\\
g_2(x) &= \sqrt{\frac{10 - x^3}{4}},\\
g_3(x) &= \tfrac{1}{2}\sqrt{10 - x^3}\,,\\
g_4(x) &= \sqrt{\frac{10}{x + 4}},\\
g_5(x) &= x - \frac{x^3 + 4x^2 - 10}{3x^2 + 8x}.
\end{aligned}
$$

Với giá trị khởi đầu $p_0 = 1.5$, áp dụng phương pháp lặp điểm cố định cho từng hàm $g_i(x)$, ta thu được các dãy giá trị khác nhau. Kết quả được trình bày trong Bảng 1.2.

Bảng 1.2: Kết quả lặp điểm cố định cho $f(x) = x^3 + 4x^2 - 10 = 0$ với các lựa chọn $g(x)$ khác nhau, bắt đầu từ $p_0 = 1.5$

n	(a)	(b)	(c)	(d)	(e)
0	1.5	1.5	1.5	1.5	1.5
1	-0.875	0.8165	1.286953768	1.348399725	1.373333333
2	6.732	2.9969	1.402540804	1.367376372	1.365262015
3	-469.7	$(-8.65)^{1/2}$	1.345458374	1.364957015	1.365230014
4	1.03×10^8		1.375170253	1.365264748	1.365230013
5			1.360094193	1.365225594	
6			1.367846968	1.365230576	
7			1.363887004	1.365229942	
8			1.365916734	1.365230022	
9			1.364878217	1.365230012	
10			1.365410062	1.365230014	
15			1.365223680	1.365230013	
20			1.365230236	1.365230013	
25			1.365230006	1.365230013	
30			1.365230013	1.365230013	

Giải thích. Bảng 1.2 minh họa rõ ràng rằng cách chọn hàm $g(x)$ ảnh hưởng trực tiếp đến sự hội tụ của phương pháp lặp điểm cố định:

- **(a)** Dãy lặp nhanh chóng phân kỳ. Các giá trị trở nên cực lớn (bùng nổ) sau vài bước.
- **(b)** Sau hai bước thì giá trị $p_2 = 2.9969$, dẫn đến $p_3 = \sqrt{-8.65}$ là vô nghĩa (không xác định trên $\mathbb{R}$). Do đó quá trình lặp dừng lại.

- **(c)** Dãy lặp hội tụ đến nghiệm đúng $p \approx 1.365230$, nhưng tốc độ hội tụ chậm; cần khoảng 30 bước mới đạt chính xác 10^{-9}.
- **(d)** Dãy lặp hội tụ nhanh chóng về nghiệm đúng $p \approx 1.365230$, với sai số giảm nhanh chỉ sau vài bước.
- **(e)** Đây là dạng công thức Newton cho phương trình $f(x) = 0$. Dãy lặp hội tụ cực nhanh (hội tụ bậc hai) và đạt chính xác 10^{-9} chỉ sau 5 bước.

Kết quả này cho thấy rằng mặc dù có nhiều cách viết lại phương trình $f(x) = 0$ thành dạng $x = g(x)$, nhưng chỉ một số lựa chọn g mang lại sự hội tụ hiệu quả. Tiêu chuẩn quan trọng để đảm bảo hội tụ là điều kiện $|g'(p)| < 1$ trong một lân cận của nghiệm p.

Định lý (Định lý điểm cố định)

Giả sử $g \in C[a, b]$ và $g(x) \in [a, b]$ với mọi $x \in [a, b]$. Giả sử thêm rằng g' tồn tại trên (a, b) và tồn tại một hằng số $0 < k < 1$ sao cho

$$|g'(x)| \leq k, \quad \forall x \in (a, b).$$

Khi đó, với mọi giá trị khởi đầu $p_0 \in [a, b]$, dãy được xác định bởi

$$p_n = g(p_{n-1}), \quad n \geq 1,$$

hội tụ về điểm cố định duy nhất $p \in [a, b]$.

—

Hệ quả

Nếu g thỏa mãn các giả thiết của Định lý điểm cố định, thì ta có các cận sai số khi dùng p_n để xấp xỉ p như sau:

$$|p_n - p| \leq k^n \max\{\, p_0 - a,\ b - p_0 \,\}, \tag{2.5}$$

và

$$|p_n - p| \leq \frac{k^n}{1 - k} |p_1 - p_0|, \quad \text{với mọi } n \geq 1. \tag{2.6}$$

Minh hoạ

Xét lại các dạng lặp điểm cố định trong Bảng 1.2 dựa trên Định lý điểm cố định và Hệ quả:

(a) Với $g_1(x) = x - (x^3 + 4x^2 - 10)$, ta có $g_1(1) = 6$ và $g_1(2) = -12$, nên g_1 không ánh xạ $[1, 2]$ vào chính nó. Ngoài ra, $g_1'(x) = 1 - 3x^2 - 8x$, và $|g_1'(x)| > 1$ với mọi $x \in [1, 2]$. Do đó, không có lý do để mong đợi sự hội tụ, và thực tế dãy lặp phân kỳ.

(b) Với $g_2(x) = \sqrt{\frac{10}{x} - 4x}$, dễ thấy g_2 không ánh xạ $[1, 2]$ vào $[1, 2]$, và dãy $\{p_n\}$ không xác định khi bắt đầu từ $p_0 = 1.5$. Ngoài ra, $|g_2'(p)| \approx 3.4 > 1$, nên điều kiện hội tụ của Định lý điểm cố địnhkhông thỏa mãn. Vì vậy, không thể mong đợi hội tụ.

(c) Với $g_3(x) = \frac{1}{2}\sqrt{10 - x^3}$, ta có $g_3'(x) = -\frac{3}{4}x^2(10 - x^3)^{-1/2} < 0$ trên $[1, 2]$, tức là g_3 giảm nghiêm ngặt. Tuy nhiên, $|g_3'(2)| \approx 2.12 > 1$, nên điều kiện của Định lý điểm cố định không thỏa trên toàn khoảng $[1, 2]$. Nhưng nếu thu hẹp miền lại còn $[1, 1.5]$, thì g_3 ánh xạ $[1, 1.5]$ vào chính nó và $|g_3'(x)| \leq 0.66 < 1$, do đó hội tụ được đảm bảo (như đã quan sát trong Bảng 1.2).

(d) Với $g_4(x) = \sqrt{\frac{10}{x+4}}$, ta có $|g_4'(x)| < 0.15$ với mọi $x \in [1, 2]$. Theo Hệ quả điểm cố định, phương pháp này phải hội tụ rất nhanh, và thực tế cho thấy đúng như vậy.

(e) Với

$$g_5(x) = x - \frac{x^3 + 4x^2 - 10}{3x^2 + 8x},$$

ta thu được công thức Newton áp dụng cho phương trình $f(x) = 0$. Đây là lý do giải thích tại sao cột (e) trong Bảng 1.2 cho thấy sự hội tụ cực kỳ nhanh (hội tụ bậc hai).

1.3 Phương pháp Newton và các mở rộng

Isaac Newton (1643–1727) là một trong những nhà khoa học vĩ đại nhất mọi thời đại. Vào cuối thế kỷ 17, các công trình của ông đã chạm đến hầu hết các lĩnh vực toán học. Phương pháp mang tên ông được giới thiệu nhằm tìm nghiệm của phương trình

$$y^3 - 2y - 5 = 0.$$

Mặc dù Newton minh hoạ phương pháp chủ yếu cho đa thức, ông đã nhận ra khả năng ứng dụng rộng hơn của nó.

Phương pháp Newton (Newton–Raphson) là một trong những kỹ thuật số mạnh mẽ và nổi tiếng để tìm nghiệm của phương trình phi tuyến. Có nhiều cách trình bày phương pháp: (i) trực quan bằng hình học/đồ thị như trong giải tích; (ii) như một

lược đồ có tốc độ hội tụ nhanh hơn so với một số lặp hàm khác; và (iii) dựa trên khai triển Taylor, vừa dẫn ra công thức, vừa cho phép ước lượng sai số xấp xỉ. Dưới đây là cách suy ra công thức bằng khai triển Taylor.

Giả sử $f \in C^2[a,b]$. Cho $p_0 \in [a,b]$ là một xấp xỉ ban đầu của nghiệm p sao cho $f'(p_0) \neq 0$ và $|p - p_0|$ "nhỏ". Khai triển Taylor bậc nhất của f tại $x = p_0$, đánh giá ở $x = p$, cho

$$f(p) \;=\; f(p_0) + (p - p_0)\, f'(p_0) + \tfrac{1}{2} f''(\xi)\, (p - p_0)^2,$$

với một ξ nằm giữa p và p_0. Vì $f(p) = 0$, ta được

$$0 \;=\; f(p_0) + (p - p_0)\, f'(p_0) + \tfrac{1}{2} f''(\xi)\, (p - p_0)^2.$$

Nếu $|p - p_0|$ đủ nhỏ, bỏ qua hạng bậc hai cho xấp xỉ tuyến tính

$$0 \;\approx\; f(p_0) + (p - p_0)\, f'(p_0),$$

từ đó

$$p \;\approx\; p_0 - \frac{f(p_0)}{f'(p_0)}.$$

Suy ra công thức truy hồi Newton

$$p_n \;=\; p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})}, \qquad n \geq 1. \tag{2.7}$$

Thuật toán – Phương pháp Newton

INPUT: giá trị gần đúng ban đầu p_0, sai số cho phép `TOL`, số lần lặp tối đa N_0.

OUTPUT: nghiệm gần đúng p, hoặc thông báo thất bại.

(1) Đặt $i = 1$.

(2) Trong khi $i \leq N_0$, thực hiện:

  (a) Tính
  $$p = p_0 - \frac{f(p_0)}{f'(p_0)}.$$

  (b) Nếu $|p - p_0| < \texttt{TOL}$, thì xuất p và **STOP**.

  (c) Đặt $i = i + 1$, $p_0 = p$.

(3) Nếu chưa hội tụ sau N_0 bước, xuất thông báo: "Phương pháp thất bại sau N_0 lần lặp." và **STOP**.

Điều kiện dừng và dạng lặp của phương pháp Newton

Các bất đẳng thức dừng được dùng trong phương pháp chia đôi cũng có thể áp dụng cho phương pháp Newton. Cụ thể, chọn một ngưỡng sai số $\varepsilon > 0$ và xây dựng dãy $\{p_n\}$ cho đến khi đạt một trong các điều kiện:

$$|p_N - p_{N-1}| < \varepsilon \tag{2.8}$$

$$\frac{|p_N - p_{N-1}|}{|p_N|} < \varepsilon, \quad p_N \neq 0 \tag{2.9}$$

hoặc

$$|f(p_N)| < \varepsilon. \tag{2.10}$$

Một dạng của (2.8) thường được sử dụng ở bước 4 trong Thuật toán 2.3. Lưu ý rằng không bất đẳng thức nào trong (2.8)–(2.10) cung cấp thông tin chính xác về sai số thực sự $|p_N - p|$.

Phương pháp Newton cũng có thể được xem như một dạng lặp điểm cố định, với công thức

$$p_n = g(p_{n-1}) = p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})}, \quad n \geq 1. \tag{2.11}$$

Đây chính là dạng lặp đã mang lại sự hội tụ rất nhanh được quan sát ở cột (e) trong Bảng **??**. Tuy nhiên, phương pháp không thể tiếp tục nếu $f'(p_{n-1}) = 0$ tại một bước nào đó. Trên thực tế, phương pháp Newton hiệu quả nhất khi f' không tiến gần 0 trong một lân cận của nghiệm p.

Ví dụ 1

Xét hàm $f(x) = \cos x - x$. Hãy xấp xỉ nghiệm của f bằng:

(a) phương pháp lặp điểm cố định;

(b) phương pháp Newton.

Lời giải.

(a) Nghiệm của bài toán tìm nghiệm cũng chính là nghiệm của bài toán điểm cố định $x = \cos x$. Đồ thị trong Hình **??** cho thấy có đúng một điểm cố định p trong khoảng $[0, \frac{\pi}{2}]$.

Với giá trị khởi đầu $p_0 = \pi/4$, ta thu được kết quả ở Bảng 1.3.

Bảng 1.3: Lặp điểm cố định cho $f(x) = \cos x - x$, $p_0 = \pi/4$

n	p_n
0	0.7853981635
1	0.7071067810
2	0.7602445972
3	0.7246674808
4	0.7487198858
5	0.7325608446
6	0.7434642113
7	0.7361282565

Từ bảng trên, có thể kết luận nghiệm $p \approx 0.74$.

(b) Để áp dụng phương pháp Newton, ta có

$$f'(x) = -\sin x - 1.$$

Với $p_0 = \pi/4$, lần lượt tính được:

$$p_1 = p_0 - \frac{\cos(\pi/4) - \pi/4}{-\sin(\pi/4) - 1} \approx 0.7395361337,$$

$$p_2 = p_1 - \frac{\cos(p_1) - p_1}{-\sin(p_1) - 1} \approx 0.7390851781,$$

$$p_3 = p_2 - \frac{\cos(p_2) - p_2}{-\sin(p_2) - 1} \approx 0.7390851332.$$

Kết quả được trình bày trong Bảng 1.4.

Bảng 1.4: Phương pháp Newton cho $f(x) = \cos x - x$, $p_0 = \pi/4$

n	p_n
0	0.7853981635
1	0.7395361337
2	0.7390851781
3	0.7390851332
4	0.7390851332

Chỉ sau ba bước lặp, nghiệm gần đúng đã đạt $p \approx 0.7390851332$, chính xác hơn nhiều so với bảy bước lặp điểm cố định.

Hội tụ khi dùng phương pháp Newton

Ví dụ 1 đã cho thấy phương pháp Newton có thể tạo ra xấp xỉ cực kỳ chính xác chỉ với rất ít bước lặp. Trong ví dụ đó, chỉ một lần lặp Newton đã cho độ chính xác tốt hơn bảy lần lặp bằng phương pháp điểm cố định. Bây giờ, ta sẽ xem xét kỹ hơn để hiểu tại sao Newton lại hiệu quả như vậy.

Khai triển Taylor ở đầu mục 2.3 đã chỉ ra tầm quan trọng của việc chọn gần đúng ban đầu chính xác. Giả định then chốt là số hạng bậc hai $(p - p_0)^2$ nhỏ hơn nhiều so với $|p - p_0|$, nên có thể bỏ qua. Điều này rõ ràng không đúng trừ khi p_0 đủ gần nghiệm p. Nếu p_0 không đủ gần nghiệm thực sự, thì khó có lý do để tin rằng Newton sẽ hội tụ về nghiệm. Tuy nhiên, trong một số trường hợp, ngay cả những xấp xỉ ban đầu kém cũng có thể dẫn đến hội tụ (xem thêm các Bài tập 15 và 16).

Định lý sau đây làm sáng tỏ sự hội tụ của phương pháp Newton và nhấn mạnh tầm quan trọng của việc chọn giá trị khởi đầu p_0.

Định lý (Hội tụ của phương pháp Newton)

Giả sử $f \in C^2[a, b]$. Nếu $p \in (a, b)$ thỏa mãn $f(p) = 0$ và $f'(p) \neq 0$, thì tồn tại một $\delta > 0$ sao cho phương pháp Newton sinh ra dãy $\{p_n\}_{n=1}^{\infty}$ hội tụ về p với mọi giá trị khởi đầu $p_0 \in [p - \delta,\, p + \delta]$.

Phác thảo chứng minh. Xem phương pháp Newton như một lặp điểm cố định:

$$p_n = g(p_{n-1}), \quad n \geq 1,$$

trong đó

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

Ta cần tìm một khoảng $[p - \delta,\, p + \delta]$ sao cho g ánh xạ khoảng này vào chính nó và tồn tại hằng số $k < 1$ sao cho $|g'(x)| \leq k$ trên đó.

Vì $f'(p) \neq 0$ và $f \in C^2$, nên g khả vi và liên tục trong một lân cận của p. Hơn nữa,

$$g'(p) = \frac{f(p)f''(p)}{[f'(p)]^2} = 0,$$

vì $f(p) = 0$. Do đó, tồn tại $\delta > 0$ sao cho $|g'(x)| < k < 1$ với mọi $x \in [p - \delta,\, p + \delta]$.

Theo Định lý điểm cố định, dãy $\{p_n\}$ sẽ hội tụ đến điểm cố định p.

The Secant Method

Phương pháp Newton là một kỹ thuật rất mạnh, nhưng có một nhược điểm lớn: cần biết giá trị của đạo hàm $f'(x)$ tại mỗi bước lặp. Trong nhiều trường hợp, việc tính đạo hàm còn khó khăn và tốn nhiều phép tính hơn so với tính $f(x)$.

Để khắc phục vấn đề này, ta xét một biến thể: thay vì sử dụng $f'(p_{n-1})$, ta xấp xỉ nó bằng công thức sai phân

$$f'(p_{n-1}) \approx \frac{f(p_{n-1}) - f(p_{n-2})}{p_{n-1} - p_{n-2}}.$$

Thay vào công thức Newton, ta thu được

$$p_n = p_{n-1} - f(p_{n-1}) \cdot \frac{p_{n-1} - p_{n-2}}{f(p_{n-1}) - f(p_{n-2})}, \qquad n > 1. \tag{2.12}$$

Kỹ thuật này được gọi là *phương pháp Secant* và được mô tả trong Thuật toán 2.4.

Phương pháp bắt đầu với hai giá trị khởi đầu p_0 và p_1. Nghiệm gần đúng p_2 chính là giao điểm với trục hoành của đường secant đi qua hai điểm $(p_0, f(p_0))$ và $(p_1, f(p_1))$. Sau đó, p_3 được tính bằng giao điểm của secant nối $(p_1, f(p_1))$ và $(p_2, f(p_2))$, và tiếp tục như vậy.

Điểm quan trọng là, sau khi tính $f(p_0)$ và $f(p_1)$, mỗi bước của phương pháp Secant chỉ cần thêm một phép tính giá trị hàm số. Trong khi đó, mỗi bước của Newton yêu cầu cả f và f'. Vì vậy, trong nhiều trường hợp phương pháp Secant là một lựa chọn kinh tế hơn.

—

Thuật toán – Phương pháp Secant

INPUT: hai giá trị khởi đầu p_0, p_1, sai số cho phép `TOL`, số lần lặp tối đa N_0.

OUTPUT: nghiệm gần đúng p, hoặc thông báo thất bại.

(1) Đặt $i = 2$.

(2) Trong khi $i \le N_0$, thực hiện:

  (a) Tính

$$p = p_{i-1} - f(p_{i-1}) \cdot \frac{p_{i-1} - p_{i-2}}{f(p_{i-1}) - f(p_{i-2})}.$$

(b) Nếu $|p - p_{i-1}| < \texttt{TOL}$, thì xuất p và **STOP**.

(c) Đặt $i = i + 1$, $p_{i-2} = p_{i-1}$, $p_{i-1} = p$.

(3) Nếu chưa hội tụ sau N_0 bước, xuất thông báo: "Phương pháp thất bại sau N_0 lần lặp." và **STOP**.

Phương pháp dây cung

Phương pháp Secant có thể hội tụ nhanh hơn phương pháp Newton trong một số trường hợp, nhưng nó không đảm bảo giữ nghiệm xấp xỉ trong khoảng ban đầu $[a, b]$. Nếu $f(a)$ và $f(b)$ trái dấu, phương pháp dây cung (*method of false position*, hay *Regula Falsi*) vừa tận dụng ý tưởng secant vừa giữ được tính chất "bracketing" như phương pháp chia đôi.

Ý tưởng cơ bản là: bắt đầu với a_0 và b_0 sao cho $f(a_0)f(b_0) < 0$. Xác định p_0 là giao điểm trục hoành của đường secant qua $(a_0, f(a_0))$ và $(b_0, f(b_0))$, tức là

$$p_0 = b_0 - f(b_0)\,\frac{b_0 - a_0}{f(b_0) - f(a_0)}.$$

Nếu $f(p_0) = 0$ thì p_0 chính là nghiệm. Nếu không, ta chọn cặp khoảng mới $[a_1, b_1]$ sao cho $f(a_1)f(b_1) < 0$, trong đó một đầu mút là p_0 và đầu mút còn lại được giữ từ bước trước. Tiếp tục quá trình để thu được dãy $\{p_n\}$.

Thuật toán – Phương pháp dây cung

INPUT: a, b sao cho $f(a)f(b) < 0$; sai số cho phép $\texttt{TOL}$; số lần lặp tối đa N_0.

OUTPUT: nghiệm gần đúng p, hoặc thông báo thất bại.

(1) Đặt $i = 1$.

(2) Trong khi $i \le N_0$, thực hiện:

(a) Tính
$$p = b - f(b)\,\frac{b - a}{f(b) - f(a)}.$$

(b) Nếu $|f(p)| < \texttt{TOL}$, thì xuất p và **STOP**.

(c) Nếu $f(a)f(p) < 0$, đặt $b = p$; ngược lại, đặt $a = p$.

(d) Đặt $i = i + 1$.

(3) Nếu chưa hội tụ sau N_0 bước, xuất thông báo: "Phương pháp thất bại sau N_0 lần lặp." và **STOP**.

1.4 Phân tích sai số cho các phương pháp lặp (Error Analysis for Iterative Methods)

Trong phần này, ta khảo sát *bậc hội tụ* của các sơ đồ lặp hàm, từ đó rút ra phương pháp Newton như một công cụ để đạt hội tụ nhanh. Đồng thời, ta cũng xem xét các cách cải thiện tốc độ hội tụ của Newton trong một số trường hợp đặc biệt.

Trước hết, cần một khái niệm để đo tốc độ hội tụ của một dãy.

Định nghĩa 2.7 (Bậc hội tụ)

Giả sử $\{p_n\}$ là một dãy hội tụ về p, với $p_n \neq p$ với mọi n. Nếu tồn tại các hằng số dương λ và α sao cho

$$\lim_{n\to\infty} \frac{|p_{n+1} - p|}{|p_n - p|^\alpha} = \lambda,$$

thì ta nói dãy $\{p_n\}$ hội tụ về p với **bậc** α, và λ được gọi là *hằng số sai số tiệm cận*.

Một kỹ thuật lặp dạng $p_n = g(p_{n-1})$ được gọi là *hội tụ bậc* α nếu dãy $\{p_n\}$ hội tụ về nghiệm $p = g(p)$ với bậc α.

Nói chung, bậc hội tụ càng cao thì dãy càng nhanh tiến tới nghiệm. Hằng số λ cũng ảnh hưởng đến tốc độ, nhưng yếu tố quyết định là bậc α. Hai trường hợp thường gặp:

(1) Nếu $\alpha = 1$ và $\lambda < 1$, dãy hội tụ tuyến tính (*linear convergence*).

(2) Nếu $\alpha = 2$, dãy hội tụ bậc hai (*quadratic convergence*).

Ví dụ minh hoạ tiếp theo sẽ so sánh một dãy hội tụ tuyến tính với một dãy hội tụ bậc hai, cho thấy lý do tại sao nên tìm các phương pháp có bậc hội tụ cao.

Minh hoạ

Giả sử $\{p_n\}$ là dãy hội tụ tuyến tính về 0 với hằng số sai số tiệm cận 0.5, tức là

$$\lim_{n\to\infty} \frac{|p_{n+1}|}{|p_n|} = 0.5.$$

Đồng thời, xét một dãy $\{q_n\}$ hội tụ bậc hai về 0 với cùng hằng số sai số tiệm cận 0.5, tức là

$$\lim_{n\to\infty} \frac{|q_{n+1}|}{|q_n|^2} = 0.5.$$

Với giả thiết $|p_0| = |q_0| = 1$, Bảng 1.5 so sánh tốc độ hội tụ của hai dãy.

Bảng 1.5: So sánh tốc độ hội tụ tuyến tính và bậc hai

n	Tuyến tính: $(0.5)^n$	Bậc hai: $(0.5)^{2^n-1}$
1	5.0000×10^{-1}	5.0000×10^{-1}
2	2.5000×10^{-1}	1.2500×10^{-1}
3	1.2500×10^{-1}	7.8125×10^{-3}
4	6.2500×10^{-2}	3.0518×10^{-5}
5	3.1250×10^{-2}	4.6566×10^{-10}
6	1.5625×10^{-2}	1.0842×10^{-19}
7	7.8125×10^{-3}	5.8775×10^{-39}

Ta thấy dãy hội tụ bậc hai đạt độ chính xác 10^{-38} chỉ sau 7 bước, trong khi dãy hội tụ tuyến tính cần ít nhất 126 bước để đạt cùng mức chính xác.

Định lý 2.8

Giả sử $g \in C[a, b]$ và g ánh xạ $[a, b]$ vào chính nó. Giả sử thêm rằng g' liên tục trên (a, b) và tồn tại hằng số $k < 1$ sao cho

$$|g'(x)| \leq k, \quad \forall x \in (a, b).$$

Nếu $g'(p) \neq 0$, thì với mọi giá trị khởi đầu $p_0 \neq p$ trong $[a, b]$, dãy

$$p_n = g(p_{n-1}), \quad n > 1,$$

hội tụ tuyến tính về nghiệm duy nhất $p \in [a, b]$, với hằng số sai số tiệm cận $|g'(p)|$.

Định lý 2.9

Giả sử p là nghiệm của phương trình $x = g(x)$. Nếu $g'(p) = 0$ và g'' liên tục với $|g''(x)| < M$ trên một khoảng mở I chứa p, thì tồn tại $\delta > 0$ sao cho, với mọi $p_0 \in [p - \delta, p + \delta]$, dãy

$$p_n = g(p_{n-1}), \quad n > 1,$$

hội tụ ít nhất bậc hai đến p. Hơn nữa, với n đủ lớn,

$$|p_{n+1} - p| \leq \tfrac{M}{2} |p_n - p|^2.$$

Multiple Roots

Trong các phần trước, ta đã giả định rằng $f'(p) \neq 0$, với p là nghiệm của phương trình $f(x) = 0$. Tuy nhiên, trong thực tế, khi $f'(p) = 0$ đồng thời với $f(p) = 0$, các phương pháp Newton và Secant thường gặp khó khăn. Để xem xét chi tiết hơn vấn đề này, ta đưa ra định nghĩa sau.

Định nghĩa 2.10 (Nghiệm bội)

Một nghiệm p của phương trình $f(x) = 0$ được gọi là *nghiệm bội (zero of multiplicity)* bậc m của f nếu, với mọi $x \neq p$, ta có thể viết

$$f(x) = (x - p)^m q(x),$$

trong đó

$$\lim_{x \to p} q(x) \neq 0.$$

Với đa thức, p là nghiệm bội bậc m của f nếu $f(x) = (x - p)^m q(x)$ và $q(p) \neq 0$.

Về bản chất, $q(x)$ biểu diễn phần của hàm $f(x)$ không đóng góp vào nghiệm tại $x = p$. Kết quả tiếp theo sẽ giúp xác định các *nghiệm đơn (simple zeros)*, tức là những nghiệm có bội bằng 1.

Định lý 2.11

Giả sử $f \in C^1[a, b]$. Hàm f có *nghiệm đơn (simple zero)* tại $p \in (a, b)$ khi và chỉ khi

$$f(p) = 0 \quad \text{và} \quad f'(p) \neq 0.$$

Định lý 2.12

Giả sử $f \in C^m[a, b]$. Hàm f có nghiệm bội m tại $p \in (a, b)$ khi và chỉ khi

$$0 = f(p) = f'(p) = f''(p) = \cdots = f^{(m-1)}(p), \quad \text{nhưng} \quad f^{(m)}(p) \neq 0.$$

Kết quả trong Định lý 2.12 cho thấy rằng nếu tồn tại một lân cận của p mà tại đó phương pháp Newton được áp dụng, thì nó sẽ hội tụ bậc hai về p đối với mọi giá trị khởi đầu p_0 đủ gần p, miễn là p là nghiệm đơn. Ví dụ tiếp theo sẽ minh hoạ rằng hội tụ bậc hai có thể không xảy ra khi nghiệm không phải là nghiệm đơn.

Ví dụ 1 – Ảnh hưởng của nghiệm bội đến phương pháp Newton

Xét hàm

$$f(x) = e^x - x - 1.$$

(a) Chứng minh rằng f có nghiệm bội 2 tại $x = 0$.

(b) Áp dụng phương pháp Newton với $p_0 = 1$ để khảo sát tốc độ hội tụ.

Lời giải.

(a) Ta có

$$f'(x) = e^x - 1, \qquad f''(x) = e^x.$$

Tại $x = 0$, ta nhận được

$$f(0) = 0, \quad f'(0) = 0, \quad f''(0) = 1 \neq 0.$$

Do đó, $x = 0$ là nghiệm bội 2 của $f(x)$.

(b) Áp dụng công thức Newton

$$p_n = p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})} = p_{n-1} - \frac{e^{p_{n-1}} - p_{n-1} - 1}{e^{p_{n-1}} - 1}.$$

Với $p_0 = 1$, ta thu được:

$$\begin{aligned}
p_1 &= 1 - \frac{e-2}{e-1} \approx 0.5819767, \\
p_2 &= 0.5819767 - \frac{e^{0.5819767} - 0.5819767 - 1}{e^{0.5819767} - 1} \approx 0.3190623, \\
p_3 &\approx 0.1729500, \\
p_4 &\approx 0.0906560, \\
p_5 &\approx 0.0460287.
\end{aligned}$$

Bảng 1.6: Các giá trị lặp của phương pháp Newton cho $f(x) = e^x - x - 1$

n	p_n
0	1.0000000
1	0.5819767
2	0.3190623
3	0.1729500
4	0.0906560
5	0.0460287

Ta thấy dãy $\{p_n\}$ hội tụ đến nghiệm $p = 0$, nhưng tốc độ hội tụ chỉ là *tuyến tính*, không phải bậc hai như trường hợp nghiệm đơn. Đây chính là hiện tượng chung của phương pháp Newton khi áp dụng cho các nghiệm bội.

1.5 Tăng tốc độ hội tụ(Accelerating Convergence)

Định lý 2.8 cho thấy rằng hội tụ bậc hai là một tính chất hiếm khi đạt được. Phần này giới thiệu một kỹ thuật gọi là *phương pháp Aitken's* Δ^2, dùng để *tăng tốc quá*

trình hội tụ tuyến tính của một dãy, bất kể dãy đó bắt nguồn từ đâu hay áp dụng trong trường hợp nào.

Nhà toán học Alexander Aitken (1895–1967) đã đề xuất kỹ thuật này năm 1926 để tăng tốc hội tụ của các chuỗi trong nghiên cứu về phương trình đại số. Quá trình này tương tự với phương pháp đã được nhà toán học Nhật Bản Takakazu Seki Kowa (1642–1708) sử dụng từ thế kỷ 17.

—

***Phương pháp Aitken's* Δ^2**

Giả sử $\{p_n\}_{n=0}^{\infty}$ là dãy hội tụ tuyến tính đến giới hạn p. Để xây dựng một dãy hội tụ nhanh hơn, ta giả định rằng các sai số $(p_n - p)$, $(p_{n+1} - p)$ và $(p_{n+2} - p)$ có cùng dấu và n đủ lớn sao cho

$$\frac{p_{n+1} - p}{p_n - p} = \frac{p_{n+2} - p}{p_{n+1} - p}.$$

Giải phương trình này theo p thu được:

$$p = p_n - \frac{(p_{n+1} - p_n)^2}{p_{n+2} - 2p_{n+1} + p_n}.$$

Công thức trên định nghĩa *một dãy mới* hội tụ nhanh hơn đến p so với dãy gốc $\{p_n\}$.

Ví dụ 1

Xét dãy $p_n = \cos(1/n)$, hội tụ tuyến tính đến $p = 1$. Dùng phương pháp Aitken's Δ^2 để tạo ra năm phần tử đầu tiên của dãy hội tụ nhanh hơn.

Lời giải. Để tính một phần tử $p_n^{(A2)}$, cần ba phần tử liên tiếp p_n, p_{n+1}, p_{n+2} của dãy gốc. Do đó, để xác định $p_5^{(A2)}$, cần bảy phần tử đầu tiên của $\{p_n\}$. Bảng dưới đây cho thấy dãy $\{p_n^{(A2)}\}$ hội tụ đến $p = 1$ nhanh hơn rõ rệt so với $\{p_n\}$.

Bảng 1.7: So sánh tốc độ hội tụ của dãy gốc và dãy Aitken's Δ^2

n	$p_n = \cos(1/n)$	$p_n^{(A2)}$
1	0.540302	–
2	0.877583	–
3	0.940007	0.998947
4	0.968912	1.000020
5	0.983347	1.000000
6	0.991698	1.000000
7	0.996043	1.000000

Định nghĩa 2.13 (Sai phân tiến)

Với dãy $\{p_n\}_{n=0}^{\infty}$, *sai phân tiến (forward difference)* được định nghĩa là

$$\Delta p_n = p_{n+1} - p_n, \quad n \geq 0.$$

Sai phân bậc cao hơn được định nghĩa đệ quy:

$$\Delta^k p_n = \Delta(\Delta^{k-1} p_n), \quad k \geq 2.$$

Do đó,

$$\Delta^2 p_n = p_{n+2} - 2p_{n+1} + p_n.$$

Khi đó, công thức Aitken có thể được viết gọn dưới dạng:

$$p_n^{(A2)} = p_n - \frac{(\Delta p_n)^2}{\Delta^2 p_n}.$$

Định lý 2.14

Giả sử $\{p_n\}$ hội tụ tuyến tính đến giới hạn p và

$$\lim_{n\to\infty} \frac{p_{n+1} - p}{p_n - p} = \lambda, \quad |\lambda| < 1.$$

Khi đó, dãy Aitken $\{p_n^{(A2)}\}$ hội tụ đến p *nhanh hơn* dãy ban đầu, theo nghĩa:

$$\lim_{n\to\infty} \frac{p_n^{(A2)} - p}{p_n - p} = 0.$$

Phương pháp Steffensen

Bằng cách áp dụng một biến thể của phương pháp Aitken's Δ^2 cho một dãy hội tụ tuyến tính thu được từ quá trình lặp điểm cố định, ta có thể tăng tốc độ hội tụ lên bậc hai. Thủ tục này được gọi là *phương pháp Steffensen*, và chỉ khác một chút so với việc áp dụng trực tiếp công thức Aitken's Δ^2 cho dãy lặp điểm cố định hội tụ tuyến tính.

Phương pháp Aitken's Δ^2 xây dựng các phần tử theo thứ tự:

$$p_0, \quad p_1 = g(p_0), \quad p_2 = g(p_1), \quad \hat{p}_0 = [\Delta^2](p_0),$$

trong đó ký hiệu $[\Delta^2]$ biểu thị công thức (2.15) được sử dụng, tức là

$$\hat{p}_0 = p_0 - \frac{(p_1 - p_0)^2}{p_2 - 2p_1 + p_0}.$$

Tiếp tục quá trình, ta có

$$p_3 = g(p_2), \quad \hat{p}_1 = [\Delta^2](p_1), \text{ v.v.}$$

Phương pháp Steffensen xây dựng các phần tử đầu tiên p_0, p_1, p_2 và $\hat{p}_0$ giống như phương pháp Aitken, nhưng sau đó sử dụng $\hat{p}_0$ làm giá trị khởi đầu mới thay vì p_2.

Nói cách khác, ta có dãy được tạo thành:

$$p_0^{(0)}, \quad p_1^{(0)} = g(p_0^{(0)}), \quad p_2^{(0)} = g(p_1^{(0)}), \quad p_0^{(1)} = [\Delta^2](p_0^{(0)}), \quad p_1^{(1)} = g(p_0^{(1)}), \ldots$$

Cứ mỗi ba phần tử, một phần tử mới của dãy Steffensen được sinh ra bằng công thức Aitken's Δ^2, trong khi các phần tử còn lại được tính bằng phép lặp điểm cố định thông thường.

Thuật toán – Phương pháp Steffensen

INPUT: Giá trị khởi đầu p_0, sai số cho phép `TOL`, và số lần lặp tối đa N_0.

OUTPUT: Nghiệm gần đúng p, hoặc thông báo thất bại.

(1) Đặt $i = 1$.

(2) Trong khi $i \leq N_0$, thực hiện các bước 3–6:

  (a) Tính $p_1 = g(p_0)$;

  (b) Tính $p_2 = g(p_1)$;

  (c) Tính
$$p = p_0 - \frac{(p_1 - p_0)^2}{p_2 - 2p_1 + p_0}.$$

  (d) Nếu $|p - p_0| < \texttt{TOL}$, thì xuất p và **STOP** (thủ tục thành công).

(3) Đặt $i = i + 1$.

(4) Gán $p_0 = p$ (cập nhật giá trị khởi đầu).

(5) Nếu $i > N_0$, xuất thông báo: "Phương pháp thất bại sau N_0 lần lặp." và **STOP**.

Ghi chú: Nếu mẫu số $(p_2 - 2p_1 + p_0)$ bằng 0, thì công thức Aitken's Δ^2 không xác định. Trong trường hợp đó, thuật toán sẽ dừng và lấy p_2 làm xấp xỉ tốt nhất có thể.

Minh hoạ

Để giải phương trình

$$x^3 + 4x^2 - 10 = 0$$

bằng *phương pháp Steffensen*, ta viết lại

$$x^3 + 4x^2 = 10,$$

suy ra công thức lặp điểm cố định:

$$g(x) = \sqrt{\frac{10}{x+4}}.$$

Phương pháp lặp điểm cố định này đã được xét trong Bảng 2.2, cột (d) của Mục 2.2. Áp dụng quy trình Steffensen với $p_0 = 1.5$, ta thu được các giá trị trong Bảng 1.8. Giá trị $p_0^{(2)} = 1.365230013$ chính xác đến chữ số thập phân thứ 9.

Trong ví dụ này, phương pháp Steffensen cho kết quả chính xác tương đương với phương pháp Newton, *nhưng không cần tính đạo hàm*. Kết quả tương tự có thể quan sát ở phần minh hoạ cuối của Mục 2.2.

Bảng 1.8: Các bước lặp của phương pháp Steffensen cho $f(x) = x^3 + 4x^2 - 10$

k	$p_0^{(k)}$	$p_1^{(k)} = g(p_0^{(k)})$	$p_2^{(k)} = g(p_1^{(k)})$
0	1.5	1.348399725	1.367376372
1	1.365265224	1.365225534	1.365230583
2	1.365230013	–	–

Định lý 2.15 (Hội tụ bậc hai của phương pháp Steffensen)

Giả sử $x = g(x)$ có nghiệm p với $g'(p) \neq 0$. Nếu tồn tại $\delta > 0$ sao cho $g \in C^3[p-\delta, p+\delta]$, thì phương pháp Steffensen hội tụ *bậc hai* về p cho mọi giá trị khởi đầu $p_0 \in [p-\delta, p+\delta]$.

1.6 Nghiệm của Đa thức và Phương pháp Muller

1.6.1 Giới thiệu về Nghiệm của Đa thức

Một đa thức bậc n có dạng tổng quát:

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

trong đó a_i là các hằng số (có thể là số thực hoặc phức) được gọi là *hệ số*, và $a_n \neq 0$. Bài toán tìm các giá trị x sao cho $P(x) = 0$ là một trong những bài toán

nền tảng và quan trọng nhất trong toán học. Các giá trị x này được gọi là **nghiệm** (root/solution) của đa thức $P(x)$ hoặc nghiệm của phương trình $P(x) = 0$.

Định lý 2.16 (Định lý Cơ bản của Đại số). *Nếu $P(x)$ là một đa thức bậc $n \geq 1$ với các hệ số thực hoặc phức, thì phương trình $P(x) = 0$ có ít nhất một nghiệm (nghiệm này có thể là số phức).*

Giải thích: Định lý này đảm bảo rằng mọi đa thức (không phải hằng số - bậc nhất trở lên) đều có ít nhất một nghiệm. Đây này là nền tảng cho việc tìm nghiệm, vì nó khẳng định sự tồn tại của nghiệm mà chúng ta đang tìm. Mặc dù tên gọi là "Định lý Cơ bản của Đại số", tuy nhiên việc chứng minh định lý này lại thường đòi hỏi các công cụ của giải tích phức. (được chứng minh lần đầu bởi Gauss vào năm 1799)

Ví dụ 1. Xác định tất cả các nghiệm của đa thức $P(x) = x^3 - 5x^2 + 17x - 13$.

Giải.

Ta dễ dàng kiểm tra (nhẩm nghiệm nguyên) rằng $P(1) = 1 - 5 + 17 - 13 = 0$. Do đó, $x = 1$ là một nghiệm của P và $(x - 1)$ là một nhân tử của đa thức. Thực hiện phép chia đa thức $P(x)$ cho $(x - 1)$, ta được:

$$P(x) = (x - 1)(x^2 - 4x + 13)$$

Để xác định các nghiệm của $x^2 - 4x + 13$, ta sử dụng công thức nghiệm bậc hai:

$$\frac{-(-4) \pm \sqrt{(-4)^2 - 4(1)(13)}}{2(1)} = \frac{4 \pm \sqrt{16 - 52}}{2} = \frac{4 \pm \sqrt{-36}}{2} = 2 \pm 3i.$$

Do đó, đa thức bậc ba $P(x)$ có ba nghiệm là: $x_1 = 1$, $x_2 = 2 - 3i$, và $x_3 = 2 + 3i$.

Hệ quả 2.17. *Nếu $P(x)$ là một đa thức bậc $n \geq 1$ với các hệ số thực hoặc phức, t hì tồn tại các hằng số duy nhất $x_1, x_2, \ldots, x_k$ (có thể là số phức) và các số nguyên dương duy nhất $m_1, m_2, \ldots, m_k$ (gọi là **bội** của nghiệm) sao cho $\sum_{i=1}^{k} m_i = n$ và*

$$P(x) = a_n(x - x_1)^{m_1}(x - x_2)^{m_2} \cdots (x - x_k)^{m_k}$$

.

Giải thích: Một đa thức bậc n có **chính xác** n **nghiệm** trong tập số phức, *nếu ta đếm cả số lần lặp lại của nghiệm* (tức là đếm theo bội).

Hệ quả 2.18. *Giả sử $P(x)$ và $Q(x)$ là các đa thức có bậc không quá n. Nếu $x_1, x_2, ..., x_k$ (với $k > n$) là các số phân biệt thỏa mãn $P(x_i) = Q(x_i)$ với $i = 1, 2, ..., k$, thì $P(x) = Q(x)$ với mọi giá trị của x.*

Kết quả này cho thấy rằng để chứng minh hai đa thức có bậc nhỏ hơn hoặc bằng n là giống nhau, ta chỉ cần chứng minh rằng chúng trùng nhau tại $n+1$ giá trị phân biệt.

1.6.2 Phương pháp Horner

Để sử dụng phương pháp Newton nhằm tìm nghiệm xấp xỉ của một đa thức $P(x)$, chúng ta cần tính $P(x)$ và $P'(x)$ tại những giá trị được cho trước. Phương pháp Horner là một thuật toán cực kỳ hiệu quả để thực hiện việc này.

Ý tưởng cốt lõi của Horner là viết lại đa thức dưới dạng "lồng nhau" (nested form) để tối ưu hóa phép tính:

$$P(x) = (\dots((a_n x + a_{n-1})x + a_{n-2})x + \cdots + a_1)x + a_0$$

Cách tính này chỉ yêu cầu n phép nhân và n phép cộng. Phương pháp này còn được gọi là **phép chia tổng hợp (synthetic division)**. Kỹ thuật này đã được biết đến ở Trung Quốc từ rất lâu trước khi Horner (1819) và Ruffini (trước đó) công bố.

Định lý 2.19 (Phương pháp Horner). *Cho đa thức $P(x) = a_n x^n + \cdots + a_0$. Đặt $b_n = a_n$ và*

$$b_k = a_k + b_{k+1}x_0 \quad \textit{cho} \quad k = n-1, n-2, \dots, 0.$$

Khi đó, $b_0 = P(x_0)$. Hơn nữa, nếu ta đặt $Q(x) = b_n x^{n-1} + b_{n-1}x^{n-2} + \cdots + b_1$, thì:

$$P(x) = (x - x_0)Q(x) + b_0$$

Điều này có nghĩa là $Q(x)$ là thương và b_0 là số dư khi chia $P(x)$ cho $(x - x_0)$.

Chứng minh: Theo định nghĩa của $Q(x)$, ta có

$$\begin{aligned}(x - x_0)Q(x) + b_0 &= (x - x_0)(b_n x^{n-1} + \cdots + b_2 x + b_1) + b_0 \\ &= (b_n x^n + b_{n-1}x^{n-1} + \cdots + b_2 x^2 + b_1 x) - (b_n x_0 x^{n-1} + \cdots + b_2 x_0 x + b_1 x_0) + b_0 \\ &= b_n x^n + (b_{n-1} - b_n x_0)x^{n-1} + \cdots + (b_1 - b_2 x_0)x + (b_0 - b_1 x_0).\end{aligned}$$

Theo giả thiết, $b_n = a_n$ và $b_k - b_{k+1}x_0 = a_k$ (do $b_k = a_k + b_{k+1}x_0$). Do đó, $(x - x_0)Q(x) + b_0 = P(x)$ và $b_0 = P(x_0)$. □

Ví dụ 2. Sử dụng phương pháp Horner để tính $P(x) = 2x^4 - 3x^2 + 3x - 4$ tại $x_0 = -2$.

Giải. Khi tính toán bằng tay, ta lập một lược đồ (bảng) chia tổng hợp như sau. Chú ý các hệ số của x^3 bằng 0.

	Hệ số của x^4 $a_4 = 2$	Hệ số của x^3 $a_3 = 0$	Hệ số của x^2 $a_2 = -3$	Hệ số của x $a_1 = 3$	Hạng tử hằng số $a_0 = -4$
$x_0 = -2$		$b_4x_0 = -4$	$b_3x_0 = 8$	$b_2x_0 = -10$	$b_1x_0 = 14$
	$b_4 = 2$	$b_3 = -4$	$b_2 = 5$	$b_1 = -7$	$\mathbf{b_0 = 10}$

Kết quả: $P(-2) = b_0 = 10$. Đa thức thương là $Q(x) = 2x^3 - 4x^2 + 5x - 7$, và ta có thể viết: $P(x) = (x + 2)(2x^3 - 4x^2 + 5x - 7) + 10$.

Ứng dụng Tính Đạo hàm trong Phương pháp Newton

Một ưu điểmcủa phương pháp Horner là nó giúp tính đạo hàm $P'(x_0)$ một cách rất hiệu quả. Từ $P(x) = (x - x_0)Q(x) + b_0$, ta lấy đạo hàm hai vế theo x:

$$P'(x) = Q(x) + (x - x_0)Q'(x)$$

Thay $x = x_0$ vào, ta được:

$$P'(x_0) = Q(x_0) \quad (2.16)$$

Như vậy, **giá trị đạo hàm $P'(x_0)$ chính bằng giá trị của đa thức thương $Q(x)$ tại điểm x_0.**

Ví dụ 3. Tìm nghiệm xấp xỉ cho $P(x) = 2x^4 - 3x^2 + 3x - 4$ bằng phương pháp Newton, bắt đầu với $x_0 = -2$ và dùng phép chia tổng hợp (Horner) để tính $P(x_n)$ và $P'(x_n)$.

Giải. Với $x_0 = -2$, từ Ví dụ 2 ta đã có $P(-2) = 10$. Sử dụng (2.16), ta có $P'(-2) = Q(-2)$, với $Q(x) = 2x^3 - 4x^2 + 5x - 7$. Ta tính $Q(-2)$ bằng lược đồ Horner:

	2	-4	5	-7
$x_0 = -2$		-4	16	-42
	2	-8	21	$\mathbf{-49} = Q(-2) = P'(-2)$

Bước lặp Newton đầu tiên:

$$x_1 = x_0 - \frac{P(x_0)}{P'(x_0)} = -2 - \frac{10}{-49} \approx -1.796$$

Để tính x_2, ta lặp lại quy trình tại $x_1 = -1.796$:

$x_1 = -1.796$	2	0	-3	3	-4
		-3.592	6.451	-6.197	5.742
	2	-3.592	3.451	-3.197	$\mathbf{1.742} = P(x_1)$
		-3.592	12.902	-29.368	
	2	-7.184	16.353	$\mathbf{-32.565} = P'(x_1)$	

Bước lặp Newton thứ hai:

$$x_2 = x_1 - \frac{P(x_1)}{P'(x_1)} = -1.796 - \frac{1.742}{-32.565} \approx -1.7425$$

Tiếp tục tương tự, ta có $x_3 \approx -1.73897$. Nghiệm chính xác đến 5 chữ số thập phân là -1.73896.

Thuật toán Horner

Algorithm 1 Phương pháp Horner

INPUT: Bậc n; các hệ số $a_0, a_1, \ldots, a_n$; điểm x_0.
OUTPUT: $y = P(x_0)$ và $z = P'(x_0)$.

Bước 1:
    Đặt $y \leftarrow a_n$ ▷ Tính b_n cho P
    Đặt $z \leftarrow a_n$ ▷ Tính b_{n-1} cho Q
Bước 2: ▷ Vòng lặp chính
for $j \leftarrow n-1$ **downto** 1 **do**
    Đặt $y \leftarrow x_0 y + a_j$ ▷ Tính b_j cho P
    Đặt $z \leftarrow x_0 z + y$ ▷ Tính b_{j-1} cho Q
Bước 3: ▷ Tính b_0 cho P
    Đặt $y \leftarrow x_0 y + a_0$
Bước 4: ▷ Trả về kết quả
    return (y, z)

1.6.3 Phép Giảm bậc (Deflation)

Khi $\hat{x}_1$ là nghiệm xấp xỉ của $P(x)$, ta có $P(x) \approx (x - \hat{x}_1)Q_1(x)$. Các nghiệm còn lại của $P(x)$ xấp xỉ bằng nghiệm của $Q_1(x)$ (đa thức bậc thấp hơn). Quá trình thay thế $P(x)$ bằng $Q_1(x)$ để tìm nghiệm tiếp theo gọi là **phép giảm bậc (deflation)**.

Vấn đề Sai số Tích lũy: Phép giảm bậc nhạy cảm với sai số. Sai số nhỏ trong $\hat{x}_1$ và sai số làm tròn khi chia sẽ ảnh hưởng đến hệ số của $Q_1(x)$, làm cho các nghiệm tìm được sau này (từ các đa thức $Q_k(x)$) ngày càng kém chính xác.

Giải pháp: Một chiến lược hiệu quả là sử dụng phép giảm bậc chỉ để tìm các **giá trị khởi tạo** $\hat{x}_2, \ldots, \hat{x}_n$. Sau đó, sử dụng các giá trị này làm điểm bắt đầu cho phương pháp Newton (hoặc Muller) áp dụng trên **đa thức gốc** $P(x)$. Quy trình này giúp "tinh chỉnh" lại các nghiệm và loại bỏ phần lớn sai số tích lũy từ quá trình giảm bậc.

1.6.4 Phương pháp Muller

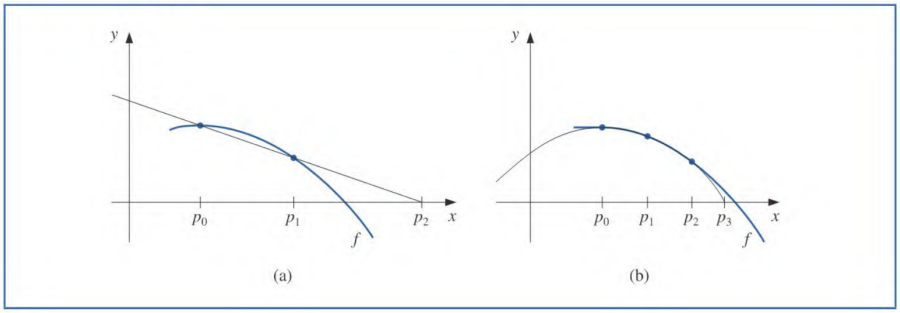
Hạn chế của các Phương pháp (và Nghiệm phức)

Một vấn đề quan trọng là đa thức với hệ số thực hoàn toàn có thể có nghiệm phức. Các phương pháp như Newton, Cát tuyến (Secant), hay Dây cung (False Position), nếu được khởi tạo với các giá trị thực, sẽ chỉ tạo ra các xấp xỉ thực trong suốt quá trình lặp. Chúng không thể "tự nhiên" tìm ra được nghiệm phức.

Định lý 2.20. *Nếu $P(x)$ là đa thức có hệ số thực và $z = a + bi$ (với $b \neq 0$) là một nghiệm phức có bội m, thì số phức liên hợp của nó $\bar{z} = a - bi$ cũng là một nghiệm của $P(x)$ với cùng bội m. Nhân tử bậc hai $(x - z)(x - \bar{z}) = x^2 - 2ax + a^2 + b^2$ (có hệ số thực) là một nhân tử của $P(x)$.*

Ý tưởng: Xấp xỉ bằng Parabol

Phương pháp Muller là một giải pháp cho vấn đề tìm nghiệm phức mà không cần khởi tạo số phức, dựa trên ý tưởng thay thế đường thẳng (trong phương pháp Cát tuyến) bằng một đường cong bậc hai (parabol).



Hình 1.2: So sánh phương pháp dây cung và phương pháp Muller

So sánh ý tưởng:

- **Phương pháp Cát tuyến (Secant):** Dùng đường thẳng qua 2 điểm $(p_0, f(p_0))$, $(p_1, f(p_1))$. Giao điểm với trục hoành là p_2.
- **Phương pháp Muller:** Dùng parabol qua 3 điểm $(p_0, f(p_0))$, $(p_1, f(p_1))$, $(p_2, f(p_2))$. Tìm các giao điểm của parabol với trục hoành. Chọn giao điểm gần p_2 nhất làm p_3.

Xây dựng Công thức Lặp

Ta viết parabol dưới dạng $P_{parabol}(x) = a(x - p_2)^2 + b(x - p_2) + c$. Việc parabol đi qua 3 điểm $(p_i, f(p_i))$ cho phép ta xác định các hệ số a, b, c:

- $f(p_2) = a(0)^2 + b(0) + c \implies c = f(p_2)$.
- $f(p_0) = a(p_0 - p_2)^2 + b(p_0 - p_2) + c$.
- $f(p_1) = a(p_1 - p_2)^2 + b(p_1 - p_2) + c$.

Giải hệ 2 phương trình (2.17) và (2.18) (sau khi thay $c = f(p_2)$) ta tìm được a và b.

Sau khi có a, b, c, ta cần tìm nghiệm của $P_{parabol}(x) = 0$. Để ổn định số học (tránh trừ các số gần bằng nhau), ta sử dụng công thức nghiệm thay thế:

$$x - p_2 = \frac{-2c}{b \pm \sqrt{b^2 - 4ac}}$$

Phương pháp Muller chọn nghiệm p_3 là nghiệm gần p_2 nhất. Điều này đạt được bằng cách chọn dấu $\pm$ sao cho mẫu số có trị tuyệt đối lớn nhất, tức là chọn dấu của $\sqrt{b^2 - 4ac}$ trùng với dấu của b:

$$p_3 = p_2 - \frac{2c}{b + \text{sgn}(b)\sqrt{b^2 - 4ac}}$$

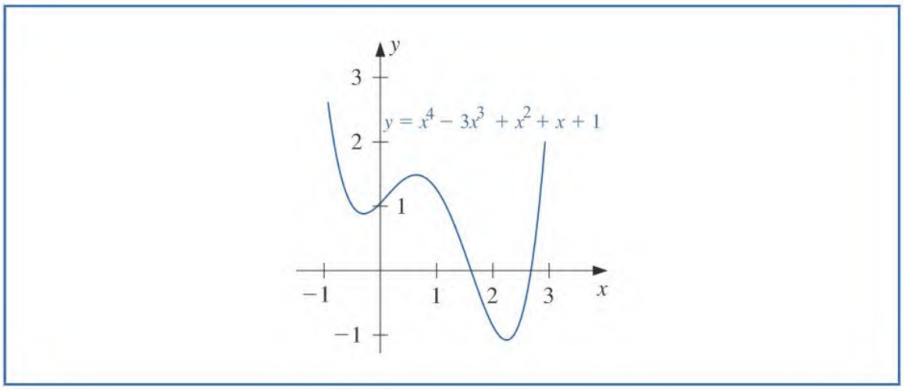
Nếu $b^2 - 4ac < 0$, $\sqrt{b^2 - 4ac}$ là số ảo, và p_3 sẽ là một số phức. Đây chính là cách phương pháp Muller "tự động" chuyển sang tìm nghiệm phức.

Thuật toán Muller

Sau khi tính được p_3, ta bỏ điểm "cũ nhất" là p_0, và lặp lại quy trình với 3 điểm mới (p_1, p_2, p_3) để tìm p_4. Thuật toán 2.8 mô tả chi tiết phương pháp này.

Ví dụ Minh họa

Xét đa thức $f(x) = x^4 - 3x^3 + x^2 + x + 1$. Sử dụng Thuật toán 2.8 với $\texttt{TOL} = 10^{-5}$ để xấp xỉ các nghiệm.



Hình 1.3: Đồ thị hàm số f(x)

Algorithm 2 Phương pháp Muller

Để tìm nghiệm của $f(x) = 0$ cho trước ba xấp xỉ p_0, p_1, p_2:

INPUT: p_0, p_1, p_2; sai số TOL; số lần lặp tối đa N_0.
OUTPUT: Nghiệm xấp xỉ p hoặc thông báo không tìm được nghiệm.

Bước 1: ▷ Khởi tạo các giá trị ban đầu
  Đặt $h_1 \leftarrow p_1 - p_0$
  Đặt $h_2 \leftarrow p_2 - p_1$
  Đặt $\delta_1 \leftarrow (f(p_1) - f(p_0))/h_1$
  Đặt $\delta_2 \leftarrow (f(p_2) - f(p_1))/h_2$
  Đặt $d \leftarrow (\delta_2 - \delta_1)/(h_2 + h_1)$
  Đặt $i \leftarrow 3$
Bước 2: ▷ Bắt đầu vòng lặp
while $i \leq N_0$ **do**
  Bước 3:
    Đặt $b \leftarrow \delta_2 + h_2 d$
    Đặt $D \leftarrow (b^2 - 4f(p_2)d)^{1/2}$ ▷ Có thể cần số phức
  Bước 4: ▷ Chọn mẫu số E để tránh sai số triệt tiêu
  if $|b - D| < |b + D|$ **then**
    Đặt $E \leftarrow b + D$
  else
    Đặt $E \leftarrow b - D$
  Bước 5: ▷ Tính h và nghiệm xấp xỉ p
    Đặt $h \leftarrow -2f(p_2)/E$
    Đặt $p \leftarrow p_2 + h$
  Bước 6: ▷ Kiểm tra điều kiện dừng
  if $|h| <$ TOL **then**
    return (p) ▷ Thủ tục thành công
  Bước 7: ▷ Cập nhật các giá trị cho lần lặp tiếp theo
    Đặt $p_0 \leftarrow p_1$
    Đặt $p_1 \leftarrow p_2$
    Đặt $p_2 \leftarrow p$
    Đặt $h_1 \leftarrow p_1 - p_0$
    Đặt $h_2 \leftarrow p_2 - p_1$
    Đặt $\delta_1 \leftarrow (f(p_1) - f(p_0))/h_1$
    Đặt $\delta_2 \leftarrow (f(p_2) - f(p_1))/h_2$
    Đặt $d \leftarrow (\delta_2 - \delta_1)/(h_2 + h_1)$
    Đặt $i \leftarrow i + 1$
Bước 8: ▷ Thất bại sau N_0 lần lặp
  return ('Phương pháp thất bại sau N_0 lần lặp.')

- **Tìm nghiệm phức:** Bắt đầu với $p_0 = 0.5$, $p_1 = -0.5$, $p_2 = 0$.

Bảng 1.9: Phương pháp Muller tìm nghiệm phức

i	p_i	$f(p_i)$
3	$-0.100000 + 0.888819i$	$-0.011200 + 3.014875i$
4	$-0.492146 + 0.447031i$	$-0.169120 - 0.736733i$
5	$-0.352226 + 0.484132i$	$-0.178600 + 0.018187i$
6	$-0.340229 + 0.443036i$	$0.011977 - 0.010556i$
7	$-0.339095 + 0.446656i$	$-0.001055 + 0.000387i$
8	$-0.339093 + 0.446630i$	$0.000000 + 0.000000i$
9	$-0.339093 + 0.446630i$	$0.000000 + 0.000000i$

- **Tìm nghiệm thực:** Sử dụng các bộ điểm khởi tạo khác nhau.

Bảng 1.10: Phương pháp Muller tìm nghiệm thực

	$p_0 = 0.5, p_1 = 1.0, p_2 = 1.5$		$p_0 = 1.5, p_1 = 2.0, p_2 = 2.5$	
i	p_i	$f(p_i)$	p_i	$f(p_i)$
3	1.40637	−0.04851	2.24733	−0.24507
4	1.38878	0.00174	2.28652	−0.01446
5	1.38939	0.00000	2.28878	−0.00012
6	1.38939	0.00000	2.28880	0.00000
7			2.28879	0.00000

Phương pháp Muller có thể hội tụ về nghiệm với nhiều lựa chọn điểm bắt đầu khác nhau. Tuy nhiên, phương pháp này vẫn có thể không tìm được nghiệm, ví dụ nếu $f(p_i) = f(p_{i+1}) = f(p_{i+2}) \neq 0$ (parabol trở thành đường thẳng song song trục hoành).

1.7 Các thư viện số học và Tổng kết Chương

1.7.1 Tổng quan về các Phần mềm số

Một chương trình số hiệu quả, khi nhận đầu vào là một hàm f và một sai số ϵ (tolerance) cho trước, cần phải đưa ra được một hoặc nhiều nghiệm xấp xỉ của $f(x) = 0$. Mỗi nghiệm này phải có sai số tuyệt đối hoặc tương đối nằm trong ϵ, và kết quả phải được tạo ra trong một khoảng thời gian hợp lý.

Nếu chương trình không thể hoàn thành nhiệm vụ, nó ít nhất phải cung cấp các giải trình có ý nghĩa về nguyên nhân thất bại và đưa ra chỉ dẫn về cách khắc phục.

Chúng tôi giới thiệu một số phần mềm (thư viện) hiệu quả, chủ yếu dựa trên các nguyên tắc và phương pháp đã trình bày bên trên.

- **Thư viện IMSL:** Cung cấp các chương trình con thực thi **phương pháp Muller** kết hợp với **giảm bậc đa thức (deflation)**. Gói này cũng bao gồm một chương trình con của R. P. Brent, sử dụng kết hợp **nội suy tuyến tính**, **nội suy bậc hai nghịch đảo** (tương tự Muller), và **phương pháp Chia đôi**. Phương pháp **Laguerre** cũng được dùng để tìm nghiệm của đa thức thực, và phương pháp **Jenkins-Traub** được dùng cho đa thức thực và phức.

- **Thư viện NAG:** Cung cấp một chương trình con kết hợp **Chia đôi**, **nội suy tuyến tính**, và **ngoại suy (extrapolation)** để xấp xỉ một nghiệm thực trên một khoảng cho trước. NAG cũng cung cấp các chương trình con để xấp xỉ *tất cả* các nghiệm của đa thức thực hoặc phức, cả hai đều sử dụng **phương pháp Laguerre cải tiến**.

- **Thư viện Netlib:** Chứa một chương trình con của T.J. Dekker, kết hợp **Chia đôi** và **phương pháp Cát tuyến (Secant method)** để xấp xỉ một nghiệm thực trong một khoảng. Nó yêu cầu một khoảng chứa nghiệm và trả về một khoảng có độ rộng nằm trong sai số ϵ cho phép. Một chương trình con khác sử dụng kết hợp Chia đôi, nội suy, và ngoại suy.

1.7.2 Tổng kết Chương

Trong chương này, chúng ta đã xem xét bài toán giải phương trình $f(x) = 0$ cho một hàm f liên tục. Tất cả các phương pháp đều bắt đầu với các xấp xỉ ban đầu và tạo ra một dãy (nếu thành công) hội tụ về một nghiệm của phương trình.

- **Phương pháp Chia đôi** và **Phương pháp Dây cung (False Position):** Đây là các phương pháp "khoảng chứa nghiệm" (bracketing). Chúng đảm bảo hội tụ nếu có một khoảng $[a, b]$ mà $f(a)$ và $f(b)$ trái dấu. Tuy nhiên, tốc độ hội tụ của chúng có thể rất chậm.

- **Phương pháp Cát tuyến (Secant)** và **Phương pháp Newton:** Thường cho tốc độ hội tụ nhanh hơn. Tuy nhiên, chúng yêu cầu các xấp xỉ ban đầu tốt (hai điểm cho Cát tuyến, và một điểm cho Newton).

- **Chiến lược kết hợp:** Các phương pháp "khoảng chứa nghiệm" (như Chia đôi) có thể được sử dụng làm "phương pháp khởi động" (starter methods) để cung cấp các xấp xỉ ban đầu đủ tốt cho các phương pháp nhanh hơn (như Newton hoặc Cát tuyến).

- **Phương pháp Muller:**

  - Cung cấp tốc độ hội tụ nhanh (bậc hội tụ $\alpha \approx 1.84$) mà không yêu cầu xấp xỉ ban đầu đặc biệt tốt.
  - Hiệu quả không bằng Newton (bậc $\alpha = 2$), nhưng tốt hơn Cát tuyến (bậc $\alpha \approx 1.62$).
  - Có ưu điểm lớn là khả năng xấp xỉ các **nghiệm phức**, ngay cả khi các xấp xỉ ban đầu là số thực.

- **Giảm bậc Đa thức (Deflation):**
  - Thường được sử dụng với phương pháp Newton hoặc Muller khi tìm nghiệm của đa thức.
  - **Lưu ý quan trọng:** Sau khi tìm được một nghiệm xấp xỉ x_1 từ đa thức *đã giảm bậc*, nghiệm này nên được sử dụng làm xấp xỉ ban đầu để chạy lại phương pháp (Newton hoặc Muller) trên đa thức *gốc*. Quy trình này đảm bảo nghiệm tìm được là nghiệm của phương trình gốc, chứ không phải của phương trình đã giảm bậc (vốn có thể bị tích lũy sai số).

- **Các phương pháp khác:**
  - **Phương pháp Laguerre:** Có tốc độ hội tụ bậc ba (cubic) và cũng xấp xỉ được nghiệm phức.
  - **Phương pháp Cauchy:** Tương tự như Muller, nhưng tránh được trường hợp thất bại của Muller khi $f(x_i) = f(x_{i+1}) = f(x_{i+2})$ tại một vài bước lặp.

2 Các Kỹ thuật Lặp trong Đại số Ma trận (Matrix Algebra)

2.1 Giới thiệu một số khái niệm cơ bản

Để giải hệ phương trình tuyến tính $Ax = b$, người ta có thể sử dụng các *phương pháp trực tiếp* (như phép khử Gauss). Tuy nhiên, trong nhiều bài toán thực tế, đặc biệt là khi giải các phương trình vi phân, chúng ta gặp các hệ phương trình tuyến tính có kích thước cực lớn và ma trận A là **ma trận thưa** (sparse matrix), tức là có tỉ lệ các phần tử bằng 0 rất cao.

Trong những trường hợp này, các phương pháp trực tiếp trở nên kém hiệu quả và tốn kém bộ nhớ. Thay vào đó, chúng ta sử dụng các **phương pháp lặp** (iterative methods). Các phương pháp này bắt đầu từ một nghiệm xấp xỉ ban đầu $x^{(0)}$ và xây dựng một dãy các nghiệm xấp xỉ $x^{(1)}, x^{(2)}, \ldots$ sao cho dãy này hội tụ về nghiệm đúng x.

2.2 Các khái niệm nền tảng

Để phân tích các phương pháp lặp, chúng ta cần các công cụ toán học cơ bản để đo lường "khoảng cách" giữa các vector và phân tích sự hội tụ.

2.2.1 Chuẩn Vector và Ma trận

Để đo lường "khoảng cách" giữa các vector nghiệm xấp xỉ và kiểm tra sự hội tụ, ta cần một cách để đo "kích thước" của vector. Khái niệm này được gọi là **chuẩn (norm)**.

Chuẩn Vector

Một **chuẩn vector** trên $\mathbb{R}^n$ là một hàm $||\cdot||$, gán cho mỗi vector x một số thực không âm, thỏa mãn các tính chất sau với mọi $x, y \in \mathbb{R}^n$ và mọi $\alpha \in \mathbb{R}$:

- $||x|| \geq 0$
- $||x|| = 0$ khi và chỉ khi $x = 0$
- $||\alpha x|| = |\alpha|||x||$
- $||x + y|| \leq ||x|| + ||y||$ (Bất đẳng thức tam giác)

Hai chuẩn vector thông dụng nhất trong $\mathbb{R}^n$ là:

- **Chuẩn l_2 (chuẩn Euclid):** $\|x\|_2 = (\sum_{i=1}^{n} x_i^2)^{1/2}$. Đây chính là độ dài hình học thông thường của vector.
- **Chuẩn l_∞ (chuẩn vô cùng/chuẩn max):** $\|x\|_\infty = \max_{1 \le i \le n} |x_i|$.

Một dãy vector $\{x^{(k)}\}_{k=1}^{\infty}$ được gọi là **hội tụ** về x (theo một chuẩn cho trước) nếu $\lim_{k \to \infty} \|x^{(k)} - x\| = 0$. Trên $\mathbb{R}^n$, sự hội tụ theo mọi chuẩn là tương đương.

Chuẩn Ma trận

Một **chuẩn ma trận** thỏa mãn 5 tính chất (bốn tính chất như chuẩn vector, và thêm tính chất $||AB|| \le ||A||||B||$). Một **chuẩn ma trận tự nhiên (natural norm)** được sinh ra (induced) bởi một chuẩn vector:

$$||A|| = \max_{||x||=1} ||Ax||$$

Chuẩn l_∞ của ma trận $A = (a_{ij})$ có công thức tính rất đơn giản:

$$||A||_\infty = \max_{1 \le i \le n} \sum_{j=1}^{n} |a_{ij}| \quad \text{(Tổng hàng lớn nhất)}$$

2.2.2 Trị riêng và Bán kính phổ

Các phương pháp lặp mà chúng ta sẽ thảo luận đều biến đổi hệ $Ax = b$ về dạng phương trình điểm bất động $x = Tx + c$. Sự hội tụ của phương pháp phụ thuộc hoàn toàn vào các tính chất của **ma trận lặp** T.

- **Trị riêng (Eigenvalue)** λ và **vector riêng (Eigenvector)** $x \neq 0$ của ma trận T thỏa mãn phương trình $Tx = \lambda x$.
- **Bán kính phổ (Spectral Radius)** của T, ký hiệu là $\rho(T)$, là trị riêng có giá trị tuyệt đối lớn nhất: $\rho(T) = \max |\lambda|$.

Một kết quả nền tảng là bán kính phổ luôn nhỏ hơn hoặc bằng bất kỳ chuẩn ma trận tự nhiên nào: $\rho(A) \le ||A||$.

2.3 Các phương pháp lặp Jacobi và Gauss-Seidel

Các kỹ thuật lặp hiếm khi được sử dụng để giải các hệ tuyến tính có kích thước nhỏ. Tuy nhiên, đối với các hệ có kích thước lớn với tỉ lệ phần tử bằng 0 cao (ma trận thưa), các kỹ thuật này lại hiệu quả cả về mặt lưu trữ máy tính lẫn tính toán.

2.3.1 Phương pháp lặp Jacobi

Phương pháp lặp Jacobi thu được bằng cách giải phương trình thứ i trong $Ax = b$ theo x_i (với điều kiện $a_{ii} \neq 0$):

$$x_i = \sum_{\substack{j=1 \\ j \neq i}}^{n} \left(-\frac{a_{ij}x_j}{a_{ii}} \right) + \frac{b_i}{a_{ii}}, \quad \text{với } i = 1, 2, \ldots, n.$$

Với mỗi $k \geq 1$, ta sinh ra các thành phần $x_i^{(k)}$ của $x^{(k)}$ từ các thành phần của $x^{(k-1)}$ theo công thức:

$$x_i^{(k)} = \frac{1}{a_{ii}} \left[\sum_{\substack{j=1 \\ j \neq i}}^{n} (-a_{ij}x_j^{(k-1)}) + b_i \right], \quad i = 1, 2, \ldots, n.$$

Ví dụ 1. Xét hệ phương trình tuyến tính $Ax = b$ sau:

$$\begin{aligned} 10x_1 - x_2 + 2x_3 &= 6, \\ -x_1 + 11x_2 - x_3 + 3x_4 &= 25, \\ 2x_1 - x_2 + 10x_3 - x_4 &= -11, \\ 3x_2 - x_3 + 8x_4 &= 15. \end{aligned}$$

Hệ này có nghiệm duy nhất là $x = (1, 2, -1, 1)^t$. Sử dụng kỹ thuật lặp Jacobi để tìm nghiệm xấp xỉ, bắt đầu với $x^{(0)} = (0, 0, 0, 0)^t$ cho đến khi $\frac{||x^{(k)} - x^{(k-1)}||_\infty}{||x^{(k)}||_\infty} < 10^{-3}$.

Giải.

Ta giải phương trình E_i cho x_i để có:

$$\begin{aligned} x_1 &= \quad + \frac{1}{10}x_2 - \frac{1}{5}x_3 \quad + \frac{3}{5}, \\ x_2 &= \frac{1}{11}x_1 \quad + \frac{1}{11}x_3 - \frac{3}{11}x_4 + \frac{25}{11}, \\ x_3 &= -\frac{1}{5}x_1 + \frac{1}{10}x_2 \quad + \frac{1}{10}x_4 - \frac{11}{10}, \\ x_4 &= \quad - \frac{3}{8}x_2 + \frac{1}{8}x_3 \quad + \frac{15}{8}. \end{aligned}$$

Từ $x^{(0)} = (0, 0, 0, 0)^t$, ta có $x^{(1)} = (0.6000, 2.2727, -1.1000, 1.8750)^t$. Các lần lặp tiếp theo được trình bày trong bảng dưới đây.

Bảng 2.1: Kết quả lặp Jacobi cho Ví dụ 1 (Lần lặp 0-5)

k	0	1	2	3	4	5
$x_1^{(k)}$	0.0000	0.6000	1.0473	0.9326	1.0152	0.9890
$x_2^{(k)}$	0.0000	2.2727	1.7159	2.0533	1.9537	2.0114
$x_3^{(k)}$	0.0000	-1.1000	-0.8052	-1.0493	-0.9681	-1.0103
$x_4^{(k)}$	0.0000	1.8750	0.8852	1.1309	0.9739	1.0214

Bảng 2.2: Kết quả lặp Jacobi cho Ví dụ 1 (Lần lặp 6-10)

k	6	7	8	9	10
$x_1^{(k)}$	1.0032	0.9981	1.0006	0.9997	1.0001
$x_2^{(k)}$	1.9922	2.0023	1.9987	2.0004	1.9998
$x_3^{(k)}$	-0.9945	-1.0020	-0.9990	-1.0004	-0.9998
$x_4^{(k)}$	0.9944	1.0036	0.9989	1.0006	0.9998

Ta dừng sau 10 lần lặp vì điều kiện dừng được thỏa mãn. Nghiệm xấp xỉ là $x^{(10)} = (1.0001, 1.9998, -0.9998, 0.9998)^t$.

Phương pháp lặp $x^{(k)} = Tx^{(k-1)} + c$ có thể được viết dưới dạng ma trận. Ta tách ma trận A thành: $A = D - L - U$ trong đó D là ma trận đường chéo, $-L$ là phần tam giác dưới (strictly lower-triangular), và $-U$ là phần tam giác trên (strictly upper-triangular) của A. Hệ $Ax = b$ trở thành $(D - L - U)x = b$, hay $Dx = (L + U)x + b$. Nếu D^{-1} tồn tại (tức là $a_{ii} \neq 0$), ta có:

$$x = D^{-1}(L + U)x + D^{-1}b$$

Đây chính là dạng ma trận của phương pháp Jacobi:

$$x^{(k)} = T_j x^{(k-1)} + c_j, \quad \text{với } T_j = D^{-1}(L + U) \text{ và } c_j = D^{-1}b$$

Algorithm 3 Phương pháp lặp Jacobi

INPUT: Số phương trình n; ma trận $A = (a_{ij})$; vector $b = (b_i)$; vector $XO = x^{(0)}$; sai số TOL; số lặp tối đa N.
OUTPUT: Nghiệm xấp xỉ $x = (x_i)$ hoặc thông báo thất bại.
Đặt $k \leftarrow 1$.
while $k \leq N$ **do**
    for $i \leftarrow 1, \ldots, n$ **do**
        Đặt $x_i \leftarrow \frac{1}{a_{ii}} \left[-\sum_{\substack{j=1 \\ j \neq i}}^{n} (a_{ij} XO_j) + b_i \right]$.
    if $\|x - XO\| <$ TOL **then**
        return $(x_1, \ldots, x_n)$ ▷ Thành công
    Đặt $k \leftarrow k + 1$.
    Cho $i \leftarrow 1, \ldots, n$: đặt $XO_i \leftarrow x_i$.
return ('Vượt quá số lần lặp tối đa') ▷ Thất bại

2.3.2 Phương pháp lặp Gauss-Seidel

Một cải tiến của phương pháp Jacobi là: khi tính $x_i^{(k)}$, ta nhận thấy các giá trị $x_1^{(k)}, \ldots, x_{i-1}^{(k)}$ vừa được tính ở cùng bước lặp k. Các giá trị "mới" này được kỳ vọng là tốt hơn các giá trị "cũ" $x_1^{(k-1)}, \ldots, x_{i-1}^{(k-1)}$. Phương pháp Gauss-Seidel sử dụng ngay các giá trị mới nhất vừa tính được:

$$x_i^{(k)} = \frac{1}{a_{ii}} \left[-\sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k-1)} + b_i \right], \quad i = 1, \ldots, n.$$

Ví dụ 3. Sử dụng phương pháp Gauss-Seidel cho hệ phương trình trong Ví dụ 1, bắt đầu với $x^{(0)} = (0, 0, 0, 0)^t$ và $\texttt{TOL} = 10^{-3}$.

Giải. Sử dụng các phương trình đã biến đổi từ Ví dụ 1, nhưng cập nhật ngay lập tức:

$$\begin{aligned}
x_1^{(k)} &= \quad + \frac{1}{10} x_2^{(k-1)} - \frac{1}{5} x_3^{(k-1)} \quad + \frac{3}{5}, \\
x_2^{(k)} &= \frac{1}{11} x_1^{(k)} \quad + \frac{1}{11} x_3^{(k-1)} - \frac{3}{11} x_4^{(k-1)} + \frac{25}{11}, \\
x_3^{(k)} &= -\frac{1}{5} x_1^{(k)} + \frac{1}{10} x_2^{(k)} \quad + \frac{1}{10} x_4^{(k-1)} - \frac{11}{10}, \\
x_4^{(k)} &= \quad - \frac{3}{8} x_2^{(k)} + \frac{1}{8} x_3^{(k)} \quad + \frac{15}{8}.
\end{aligned}$$

Với $x^{(0)} = (0, 0, 0, 0)^t$, ta có $x^{(1)}$:

$$\begin{aligned}
x_1^{(1)} &= \quad + \frac{1}{10}(0) - \frac{1}{5}(0) \quad + \frac{3}{5} = 0.6000, \\
x_2^{(1)} &= \frac{1}{11}(0.6000) \quad + \frac{1}{11}(0) - \frac{3}{11}(0) + \frac{25}{11} \approx 2.3272, \\
x_3^{(1)} &= -\frac{1}{5}(0.6000) + \frac{1}{10}(2.3272) \quad + \frac{1}{10}(0) - \frac{11}{10} \approx -0.9873, \\
x_4^{(1)} &= \quad - \frac{3}{8}(2.3272) + \frac{1}{8}(-0.9873) \quad + \frac{15}{8} \approx 0.8789.
\end{aligned}$$

Các lần lặp tiếp theo được trình bày trong Bảng dưới.

Bảng 2.3: Kết quả lặp Gauss-Seidel cho Ví dụ

k	0	1	2	3	4	5
$x_1^{(k)}$	0.0000	0.6000	1.0300	1.0065	1.0009	1.0001
$x_2^{(k)}$	0.0000	2.3272	2.0370	2.0036	2.0003	2.0000
$x_3^{(k)}$	0.0000	-0.9873	-1.0140	-1.0025	-1.0003	-1.0000
$x_4^{(k)}$	0.0000	0.8789	0.9844	0.9983	0.9999	1.0000

Ta dừng sau 5 lần lặp vì điều kiện dừng được thỏa mãn. Lưu ý rằng phương pháp Gauss-Seidel chỉ cần 5 lần lặp, trong khi phương pháp Jacobi cần 10 lần lặp để đạt cùng độ chính xác.

Dạng ma trận của Gauss-Seidel là:

$$(D - L)x^{(k)} = Ux^{(k-1)} + b$$

$$x^{(k)} = (D - L)^{-1}Ux^{(k-1)} + (D - L)^{-1}b$$

Hay $x^{(k)} = T_g x^{(k-1)} + c_g$, với $T_g = (D - L)^{-1}U$ và $c_g = (D - L)^{-1}b$.

Algorithm 4 Phương pháp lặp Gauss-Seidel (Algorithm 7.2)

INPUT: Số n; ma trận $A = (a_{ij})$; vector $b = (b_i)$; vector $XO = x^{(0)}$; sai số TOL; số lặp tối đa N.
OUTPUT: Nghiệm xấp xỉ $x = (x_i)$ hoặc thông báo thất bại.
Đặt $k \leftarrow 1$.
while $k \leq N$ **do**
    for $i \leftarrow 1, \ldots, n$ **do**
        Đặt $x_i \leftarrow \frac{1}{a_{ii}} \left[-\sum_{j=1}^{i-1} a_{ij} x_j - \sum_{j=i+1}^{n} a_{ij} XO_j + b_i \right]$.
    if $\|x - XO\| <$ TOL **then**
        return $(x_1, \ldots, x_n)$ ▷ Thành công
    Đặt $k \leftarrow k + 1$.
    Cho $i \leftarrow 1, \ldots, n$: đặt $XO_i \leftarrow x_i$.
return ('Vượt quá số lần lặp tối đa') ▷ Thất bại

2.3.3 Phân tích Hội tụ

Kết quả lý thuyết quan trọng nhất để xác định sự hội tụ của phương pháp lặp $x^{(k)} = Tx^{(k-1)} + c$ được cho bởi các định lý sau:

Định lý 7.17. *Các mệnh đề sau là tương đương:*

(i) A *là một ma trận hội tụ (tức là* $\lim_{k\to\infty}(A^k)_{ij} = 0$ *với mọi* i, j*).*

(ii) $\lim_{n\to\infty} ||A^n|| = 0$ *với một chuẩn tự nhiên nào đó.*

(iii) $\lim_{n\to\infty} ||A^n|| = 0$ *với mọi chuẩn tự nhiên.*

(iv) $\rho(A) < 1$ *(Bán kính phổ của* A *nhỏ hơn 1).*

(v) $\lim_{n\to\infty} A^n x = 0$*, với mọi vector* x*.*

Định lý 7.19. *Với mọi* $x^{(0)} \in \mathbb{R}^n$*, dãy* $\{x^{(k)}\}_{k=0}^{\infty}$ *xác định bởi*

$$x^{(k)} = Tx^{(k-1)} + c, \quad \text{với } k \geq 1,$$

hội tụ đến nghiệm duy nhất của $x = Tx + c$ ***khi và chỉ khi*** $\rho(T) < 1$.

Kết quả này là cho ta 1 nhận xét quan trọng: phương pháp lặp hội tụ khi và chỉ khi bán kính phổ của ma trận lặp T (dù là T_j hay T_g) nhỏ hơn 1. Tốc độ hội tụ phụ thuộc vào $\rho(T)$. Ta muốn $\rho(T)$ càng nhỏ càng tốt.

Hệ quả 7.20. *Nếu* $\|T\| < 1$ *đối với một chuẩn ma trận tự nhiên bất kỳ, thì dãy* $x^{(k)} = Tx^{(k-1)} + c$ *hội tụ đến nghiệm duy nhất* $x = Tx + c$*, và ta có các cận sai số sau:*

(i) $\|x - x^{(k)}\| \leq \|T\|^k \|x^{(0)} - x\|$;

(ii) $\|x - x^{(k)}\| \leq \frac{\|T\|^k}{1-\|T\|} \|x^{(1)} - x^{(0)}\|$.

Định lý 7.21. *Nếu A là ma trận **chéo trội nghiêm ngặt** (strictly diagonally dominant), thì với mọi lựa chọn $x^{(0)}$, cả hai phương pháp Jacobi và Gauss-Seidel đều hội tụ đến nghiệm duy nhất của $Ax = b$.*

Định lý 7.22 (Stein-Rosenberg). *Nếu $a_{ij} \leq 0$ với mọi $i \neq j$ và $a_{ii} > 0$ với mọi i, thì chỉ một trong các trường hợp sau xảy ra:*

(i) $0 \leq \rho(T_g) < \rho(T_j) < 1$ *(Cả hai hội tụ, Gauss-Seidel nhanh hơn)*

(ii) $1 < \rho(T_j) < \rho(T_g)$ *(Cả hai phân kỳ, Gauss-Seidel phân kỳ nhanh hơn)*

(iii) $\rho(T_j) = \rho(T_g) = 0$

(iv) $\rho(T_j) = \rho(T_g) = 1$

2.4 Kỹ thuật Tăng tốc (Relaxation) cho Hệ Tuyến tính

Tốc độ hội tụ của phương pháp lặp phụ thuộc vào $\rho(T)$. Ta muốn chọn phương pháp có $\rho(T)$ nhỏ nhất.

2.4.1 Vector Dư và Phương pháp SOR

Ta định nghĩa **vector dư (residual vector)** cho một nghiệm xấp xỉ $\tilde{x}$ là $r = b - A\tilde{x}$. Mục tiêu ở đây là làm cho r hội tụ về $\mathbf{0}$.

Hãy xem lại phương pháp Gauss-Seidel. Khi ta chuẩn bị tính $x_i^{(k)}$, ta đang có vector xấp xỉ $x_i^{(k)} = (x_1^{(k)}, \ldots, x_{i-1}^{(k)}, x_i^{(k-1)}, \ldots, x_n^{(k-1)})^t$.

Thành phần thứ i của vector dư tại bước này là:

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

Sắp xếp lại, ta có:

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

Công thức Gauss-Seidel chính là đặt vế phải bằng $a_{ii}x_i^{(k)}$. Do đó:

$$a_{ii}x_i^{(k-1)} + r_{ii}^{(k)} = a_{ii}x_i^{(k)} \implies x_i^{(k)} = x_i^{(k-1)} + \frac{r_{ii}^{(k)}}{a_{ii}}$$

Nói cách khác, ở mỗi bước sẽ điều chỉnh x_i một lượng bằng $r_{ii}^{(k)}/a_{ii}$ để làm cho thành phần thứ i của vector dư tiếp theo bằng 0.

Các phương pháp **Relaxation** (điều hòa) tổng quát hóa bước này bằng cách đưa vào một tham số ω:

$$x_i^{(k)} = x_i^{(k-1)} + \omega\frac{r_{ii}^{(k)}}{a_{ii}}$$

Hoặc, ở dạng tính toán:

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

- $0 < \omega < 1$: gọi là **under-relaxation** (giảm tốc), dùng để ổn định các hệ phân kỳ.
- $\omega > 1$: gọi là **Successive Over-Relaxation (SOR)** (tăng tốc), dùng để tăng tốc các hệ hội tụ.
- $\omega = 1$: chính là phương pháp Gauss-Seidel.

Dạng ma trận của SOR là $x^{(k)} = T_\omega x^{(k-1)} + c_\omega$, với:

$$T_\omega = (D - \omega L)^{-1}[(1-\omega)D + \omega U] \quad \text{và} \quad c_\omega = \omega(D - \omega L)^{-1}b$$

Ví dụ 1 (Mục 7.4). Hệ $4x_1 + 3x_2 = 24$, $3x_1 + 4x_2 - x_3 = 30$, $-x_2 + 4x_3 = -24$ có nghiệm $(3, 4, -5)^t$. So sánh Gauss-Seidel ($\omega = 1$) và SOR ($\omega = 1.25$) với $x^{(0)} = (1, 1, 1)^t$.

Giải. Phương trình lặp Gauss-Seidel ($\omega = 1$):

$$\begin{aligned} x_1^{(k)} &= -0.75x_2^{(k-1)} + 6 \\ x_2^{(k)} &= -0.75x_1^{(k)} + 0.25x_3^{(k-1)} + 7.5 \\ x_3^{(k)} &= \qquad\quad + 0.25x_2^{(k)} - 6 \end{aligned}$$

Phương trình lặp SOR ($\omega = 1.25$):

$$\begin{aligned} x_1^{(k)} &= -0.25x_1^{(k-1)} - 0.9375x_2^{(k-1)} + 7.5 \\ x_2^{(k)} &= -0.9375x_1^{(k)} - 0.25x_2^{(k-1)} + 0.3125x_3^{(k-1)} + 9.375 \\ x_3^{(k)} &= \qquad\quad + 0.3125x_2^{(k)} - 0.25x_3^{(k-1)} - 7.5 \end{aligned}$$

Kết quả 7 lần lặp đầu tiên được cho trong các bảng dưới.

Bảng 2.4: Gauss-Seidel ($\omega = 1$)

k	0	1	2	3	4	5	6	7
$x_1^{(k)}$	1.0	5.2500	3.8125	3.1406	3.0879	3.0549	3.0343	3.0215
$x_2^{(k)}$	1.0	3.8125	3.8828	3.9268	3.9542	3.9714	3.9821	3.9888
$x_3^{(k)}$	1.0	-5.0469	-5.0293	-5.0183	-5.0114	-5.0072	-5.0045	-5.0028

Bảng 2.5: SOR ($\omega = 1.25$)

k	0	1	2	3	4	5	6	7
$x_1^{(k)}$	1.0	6.3125	3.5195	2.6223	3.1333	2.9571	3.0037	2.9963
$x_2^{(k)}$	1.0	3.5195	3.9585	4.0103	4.0075	4.0029	4.0009	4.0003
$x_3^{(k)}$	1.0	-6.6501	-4.6004	-5.0967	-4.9735	-5.0057	-4.9983	-5.0003

Để đạt độ chính xác 10^{-7}, Gauss-Seidel cần 34 lần lặp, trong khi SOR (với $\omega = 1.25$) chỉ cần 14 lần lặp.

Như vậy, rõ ràng việc chọn ω tối ưu là rất quan trọng. Ta có các định lý sau đây về việc lựa chọn ω

Định lý 7.24 (Kahan). *Nếu $a_{ii} \neq 0$ với mọi i, thì $\rho(T_\omega) \geq |\omega - 1|$. Điều này suy ra rằng phương pháp SOR* ***chỉ có thể hội tụ nếu*** $0 < \omega < 2$.

Định lý 7.25 (Ostrowski-Reich). *Nếu A là ma trận* ***xác định dương*** *(positive definite) và $0 < \omega < 2$, thì phương pháp SOR hội tụ với mọi vector ban đầu $x^{(0)}$.*

Định lý 7.26. *Nếu A là ma trận* ***xác định dương và tam tuyến (tridiagonal)****, thì $\rho(T_g) = [\rho(T_j)]^2 < 1$, và giá trị ω tối ưu cho SOR là:*

$$\omega = \frac{2}{1 + \sqrt{1 - [\rho(T_j)]^2}}$$

Với lựa chọn này, $\rho(T_\omega) = \omega - 1$.

Algorithm 5 Phương pháp SOR

INPUT: Số n; ma trận $A = (a_{ij})$; vector $b = (b_i)$; vector $XO = x^{(0)}$; tham số ω; sai số TOL; số lặp tối đa N.
OUTPUT: Nghiệm xấp xỉ $x = (x_i)$ hoặc thông báo thất bại.
Đặt $k \leftarrow 1$.
while $k \leq N$ **do**
  for $i \leftarrow 1, \ldots, n$ **do**
    Đặt $x_i \leftarrow (1-\omega)XO_i + \frac{\omega}{a_{ii}}\left[b_i - \sum_{j=1}^{i-1} a_{ij}x_j - \sum_{j=i+1}^{n} a_{ij}XO_j\right]$.
  if $\|x - XO\| <$ TOL **then**
    return $(x_1, \ldots, x_n)$ ▷ Thành công
  Đặt $k \leftarrow k + 1$.
  Cho $i \leftarrow 1, \ldots, n$: đặt $XO_i \leftarrow x_i$.
return ('Vượt quá số lần lặp tối đa') ▷ Thất bại

2.5 Cận sai số (Error Bound) và Tinh chỉnh Lặp (Iterative Refinement)

2.5.1 Số điều kiện và Nghịch lý Vector Dư

Một cách trực quan để kiểm tra nghiệm xấp xỉ $\tilde{x}$ của $Ax = b$ là tính vector phần dư $r = b - A\tilde{x}$. Ta thường nghĩ rằng nếu $||r||$ nhỏ, thì sai số $||x - \tilde{x}||$ cũng nhỏ. Tuy nhiên, điều này có thể sai hoàn toàn. Hãy xem xét ví dụ dưới đây:

Ví dụ 1 Hệ tuyến tính

$$\begin{pmatrix} 1 & 2 \\ 1.0001 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 3 \\ 3.0001 \end{pmatrix}$$

có nghiệm đúng $x = (1, 1)^t$.

Xét một xấp xỉ rất tồi: $\tilde{x} = (3, -0.0001)^t$.

Sai số thực sự là $||x - \tilde{x}||_\infty = \max\{|1-3|, |1-(-0.0001)|\} = 2$.

Tuy nhiên, vector phần dư lại rất nhỏ:

$$r = b - A\tilde{x} = \begin{pmatrix} 3 \\ 3.0001 \end{pmatrix} - \begin{pmatrix} 1 & 2 \\ 1.0001 & 2 \end{pmatrix} \begin{pmatrix} 3 \\ -0.0001 \end{pmatrix} = \begin{pmatrix} 0.0002 \\ 0 \end{pmatrix}$$

Vậy $||r||_\infty = 0.0002$. Sai số rất lớn, nhưng vector dư lại rất nhỏ!

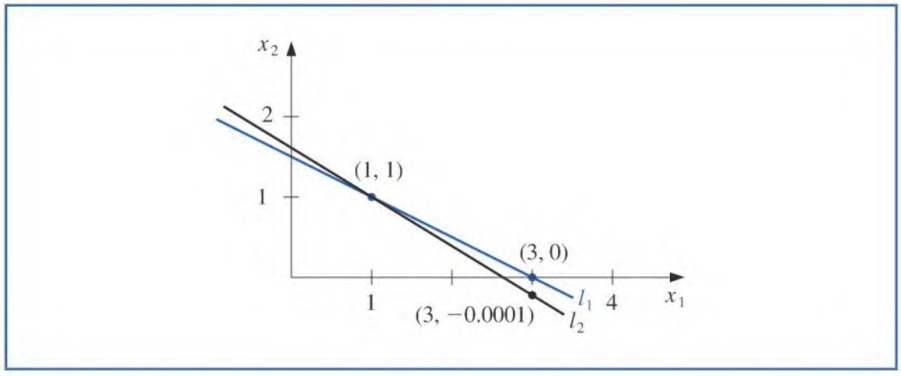
Nguyên nhân là do nghiệm của hệ là giao điểm của hai đường thẳng:

- $l_1 : x_1 + 2x_2 = 3 \implies x_2 = -0.5x_1 + 1.5$

- $l_2 : 1.0001x_1 + 2x_2 = 3.0001 \implies x_2 = -0.50005x_1 + 1.50005$

Hai đường thẳng này có hệ số góc (-0.5 và -0.50005) gần như y hệt nhau, tức là chúng **gần như song song**.

Điểm $\tilde{x} = (3, -0.0001)$ nằm rất gần (hoặc trên) đường l_2. Vì l_1 và l_2 gần như trùng nhau, điểm này cũng nằm rất gần l_1. Điều này lý giải tại sao vector dư r (khoảng cách đến vế phải b) rất nhỏ, mặc dù $\tilde{x}$ ở rất xa giao điểm thực sự là $(1, 1)$.



Hình 2.1: Đồ thị 2 đường thẳng trong ví dụ 1

Hiện tượng này xảy ra ở các hệ **bị điều kiện hóa xấu (ill-conditioned)**. Ta định nghĩa **số điều kiện (condition number)** của ma trận A (so với một chuẩn) là:

$$K(A) = \|A\| \cdot \|A^{-1}\|$$

Một ma trận là "well-conditioned" (điều kiện tốt) nếu $K(A)$ gần 1, và "ill-conditioned" (điều kiện xấu) nếu $K(A)$ rất lớn. Với ma trận trong Ví dụ 1, $K_\infty(A) = 60002$, một giá trị rất lớn, khẳng định ma trận có điều kiện rất xấu.

Chúng ta có định lý sau đây về sai số của nghiệm:

Định lý 7.27. *Giả sử $\tilde{x}$ là nghiệm xấp xỉ của $Ax = b$ và r là vector dư. Khi đó, với mọi chuẩn tự nhiên:*

$$||x - \tilde{x}|| \leq ||A^{-1}|| \cdot ||r||$$

và nếu $x \neq 0, b \neq 0$:

$$\frac{||x - \tilde{x}||}{||x||} \leq K(A)\frac{||r||}{||b||}$$

Bất đẳng thức thứ hai giải thích mọi thứ: **Sai số tương đối của nghiệm** bị chặn bởi **Số điều kiện (Conditional Number)** nhân với **vector dư tương đối**. Nếu $K(A)$ lớn, sai số nghiệm có thể lớn ngay cả khi vector dư nhỏ.

2.5.2 Kỹ thuật Tinh chỉnh Lặp (Iterative Refinement)

Ta có thể ước lượng sai số $e = x - \tilde{x}$ bằng cách giải hệ $Ay = r$, vì $A(x - \tilde{x}) = Ax - A\tilde{x} = b - A\tilde{x} = r$. Kỹ thuật Tinh chỉnh Lặp (hay Cải thiện Lặp) dựa trên ý tưởng này để cải thiện nghiệm đã tính (thường bằng phép khử Gauss).

Algorithm 6 Tinh chỉnh Lặp (Iterative Refinement)

INPUT: Số n; ma trận A; vector b; số lặp tối đa N; sai số `TOL`; số chữ số có nghĩa t.
OUTPUT: Nghiệm xấp xỉ xx và ước lượng số điều kiện $COND$.
Bước 0: Giải hệ $Ax = b$ bằng phép khử Gauss (với t chữ số), thu được $x = (x_1, \ldots, x_n)^t$. Lưu lại các phép biến đổi (phân rã LU và các phép hoán vị).
Đặt $k \leftarrow 1$.
while $k \leq N$ **do**
    Tính vector dư $r = b - Ax$. (Quan trọng: **Phải thực hiện bằng độ chính xác cao hơn**, ví dụ double precision, $2t$ chữ số).
    Giải hệ tuyến tính $Ay = r$ để tìm $y = (y_1, \ldots, y_n)^t$. (Sử dụng lại phép phân rã LU từ Bước 0).
    Cập nhật nghiệm: $xx \leftarrow x + y$.
    if $k = 1$ **then**
        Ước lượng $COND \leftarrow \frac{||y||_\infty}{||xx||_\infty} 10^t$.
    if $||x - xx||_\infty <$ `TOL` (hoặc $||y||_\infty <$ `TOL`) **then**
        return (xx, COND)
    Đặt $k \leftarrow k + 1$.
    Đặt $x \leftarrow xx$.
return ('Vượt quá số lần lặp', COND)

Kỹ thuật này cực kỳ hiệu quả. Nếu $K(A) \approx 10^q$, sau k lần tinh chỉnh, nghiệm sẽ có khoảng $k(t - q)$ chữ số đúng.

2.6 Tổng kết chương

Trong chương này, chúng ta đã nghiên cứu các kỹ thuật lặp để xấp xỉ nghiệm của các hệ tuyến tính. Chúng ta bắt đầu với phương pháp Jacobi và phương pháp Gauss-Seidel để giới thiệu về các phương pháp lặp. Cả hai phương pháp đều yêu cầu một xấp xỉ ban đầu $x^{(0)}$ tùy ý và tạo ra một dãy các vector $x^{(i+1)}$ bằng cách sử dụng một phương trình có dạng:

$$x^{(i+1)} = Tx^{(i)} + c$$

Một điểm mấu chốt đã được lưu ý là phương pháp sẽ hội tụ **khi và chỉ khi** bán kính phổ của ma trận lặp $\rho(T) < 1$, và bán kính phổ càng nhỏ thì tốc độ hội tụ càng nhanh.

Việc phân tích các vector dư (residual vectors) của kỹ thuật Gauss-Seidel đã dẫn đến phương pháp lặp **SOR (Successive Over-Relaxation)**, vốn sử dụng một tham số ω để tăng tốc độ hội tụ.

Các phương pháp lặp này và các biến thể của chúng được sử dụng rộng rãi trong việc giải các hệ phương trình tuyến tính phát sinh từ việc giải số các bài toán giá trị biên và phương trình vi phân riêng phần. Các hệ thống này thường rất lớn (có thể lên đến 10,000 phương trình) và **thưa** (sparse), với các phần tử khác không nằm ở các vị trí có thể dự đoán được. Các phương pháp lặp cũng hữu ích cho các hệ thưa lớn khác và dễ dàng được điều chỉnh để sử dụng hiệu quả trên các máy tính song song.

Phụ lục