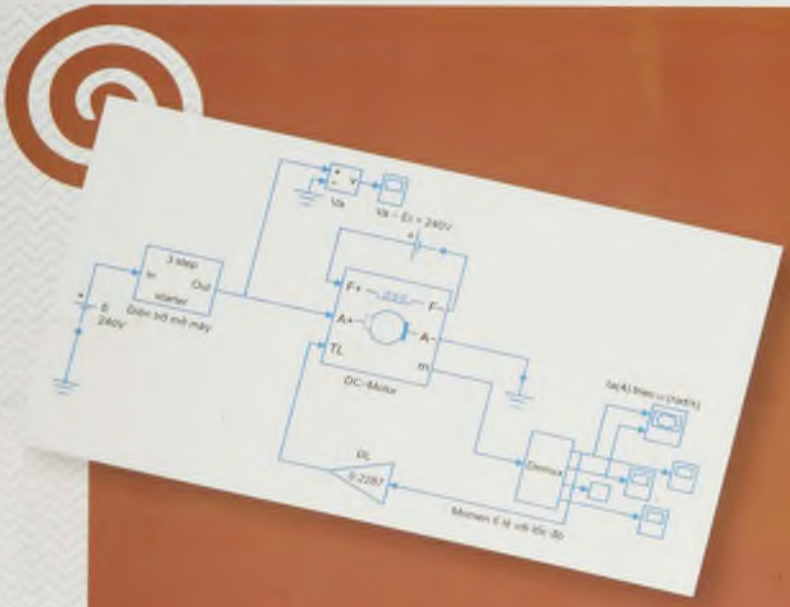
NG (Chủ biên) - NGUYỄN TRỌNG DŨNG

GIÁO TRÌNH

PHƯƠNG PHÁP TÍNH VÀ TIN HỌC CHUYÊN NGÀNH



NHÀ XUẤT BẢN ĐẠI HỌC SƯ PHẠM

NGUYỄN CHÍNH CƯƠNG (Chủ biên)
NGUYỄN TRỌNG DŨNG

Giáo trình

PHƯƠNG PHÁP TÍNH VÀ TIN HỌC CHUYÊN NGÀNH

NHÀ XUẤT BẢN ĐẠI HỌC SƯ PHẠM

MỤC LỤC

Trang

LỜI MỞ ĐẦU

Trong quá trình nghiên cứu, học tập các môn học chuyên ngành Vật lí, chúng ta thường gặp phải một số khó khăn khi tính toán như: Tính một tích phân không ở dạng thông thường, giải một phương trình vi phân nhiều ẩn, biến đổi một biểu thức vật lí có dạng toán học phức tạp, nghiên cứu một hệ vật lí giả định... Việc này rất tốn công sức và không phải lúc nào cũng có thể thu được những kết quả tường minh. Vì vậy, đòi hỏi chúng ta phải sử dụng phương pháp tính số, mô phỏng,... từ đó có thể tìm lại các kết quả ở dạng gần đúng với sai số nằm trong giới hạn cho phép. Để làm được điều đó thì ngoài những kiến thức chuyên ngành tốt đòi hỏi chúng ta phải có kiến thức về phương pháp tính số và các lĩnh vực khoa học tính toán mô phỏng.

Cùng với sự phát triển nhanh chóng của ngành Công nghệ thông tin, việc sử dụng máy tính và các phần mềm chuyên dụng đã và đang là xu thế tất yếu của các ngành khoa học tính toán nói chung và ngành Vật lí học nói riêng. Thực tế chỉ ra rằng, việc sử dụng máy tính điện tử trong nghiên cứu, học tập vật lí đã đem lại những thành tựu vô cùng quan trọng. Điều này được thể hiện ở việc ứng dụng máy tính trong rất nhiều chuyên ngành, nhiều lĩnh vực khác nhau của vật lí như: Vật lí lí thuyết, Thiên văn, Vật lí chất rắn... nhằm hỗ trợ cho việc tính toán và đo đạc chính xác, xây dựng các mô hình lí thuyết, mô phỏng, dự đoán các hiện tượng và hỗ trợ thí nghiệm... Việc sử dụng máy tính nhằm hỗ trợ quá trình nghiên cứu đã trở nên phổ biến đối với công tác nghiên cứu khoa học trong nước và trên thế giới.

Từ mong muốn đó, giáo trình "***Phương pháp tính và tin học chuyên ngành***" bước đầu trang bị cho học viên cao học ngành Vật lí phương pháp tính gần đúng, tính số và ngôn ngữ lập trình Mathlab, Fortran... nhằm hỗ trợ mô phỏng các hệ vật lí. Đây là những công cụ được nhiều nhà khoa học, học viên, sinh viên một số trường đại học tiên tiến sử dụng.

Phương pháp tính số và mô phỏng là một lĩnh vực rộng và mới mẻ nên có nhiều vấn đề cần quan tâm, do thời gian còn hạn chế nên trong giáo trình khó tránh khỏi những thiếu sót. Rất mong được quý bạn đọc góp ý để chúng tôi chỉnh sửa và hoàn thiện giúp cho các học viên cao học và các sinh viên có được một tài liệu học tập hữu ích.

CÁC TÁC GIẢ

PHẦN I

CÁC PHƯƠNG PHÁP TÍNH VÀ MÔ PHỎNG

Chương 1
CÁC PHƯƠNG PHÁP TÍNH VÀ MÔ PHỎNG

§1. SAI SỐ

Sai số thường xuyên xảy ra khi chúng ta chuyển bài toán phức tạp về dạng đơn giản hơn. Mặt khác, sai số còn phản ánh kết quả tính toán có sát với các giá trị thực hay không.

1.1. Sai số tuyệt đối và sai số tương đối

Sai số tuyệt đối: Xét đại lượng đúng A có giá trị gần đúng là a. Lúc đó ta nói "a xấp xỉ A" và viết $a \approx A$. Trị tuyệt đối $|a - A|$ gọi là sai số tuyệt đối của a. Do không biết số đúng A nên không tính được sai số tuyệt đối của a, vì vậy ta tìm cách ước lượng sai số đó bằng số dương Δ_a nào đó lớn hơn hoặc bằng $|a - A|$

$$|a - A| \le \Delta_a \tag{1.1}$$

Số dương Δ_a này gọi là sai số tuyệt đối giới hạn của a.

• Nếu Δ_a là sai số tuyệt đối giới hạn của a thì mọi số $\Delta' > \Delta_a$ đều là sai số tuyệt đối giới hạn của a. Vì vậy trong điều kiện cụ thể người ta chọn Δ_a là số dương nhỏ nhất có thể được thoả mãn (1.1).

• Nếu số xấp xỉ a của A có sai số tuyệt đối giới hạn là Δ_a thì quy ước viết:

$$A = a \pm \Delta_a \tag{1.2}$$

Với nghĩa của (1.1) tức là:

$$a - \Delta_a \le A \le a + \Delta_a \tag{1.3}$$

Sai số tương đối: Tỉ số $\dfrac{|a - A|}{|a|} \cong \dfrac{|a - A|}{|A|}$ gọi là sai số tương đối của a so với A.

Ta gọi tỉ số:

$$\delta_a = \frac{\Delta_a}{a} \tag{1.4}$$

là tỉ số tương đối giới hạn của a. Từ đó ta có :

$$\Delta_a = |a|.\delta_a \tag{1.5}$$

Do (1.5) nên (1.2) cũng có thể viết:

$$A = a.(1 \pm \delta_a) \tag{1.6}$$

* ***Lưu ý***: Sai số tuyệt đối không nói lên chất lượng của một số xấp xỉ, "chất lượng" ấy được phản ánh qua sai số tương đối. Cho nên khi muốn so sánh sự chính xác của các phép đo cùng loại thì ta sử dụng sai số tương đối.

1.2. Cách viết số xấp xỉ

Chữ số có nghĩa: Một số thập phân có thể gồm nhiều chữ số nhưng ta chỉ tính các chữ số khác không đầu tiên từ trái sang phải là chữ số có nghĩa.

Chữ số đáng tin: Mọi số thập phân đều có dạng:

$$a = \pm \sum \alpha_s .10^s \tag{1.7}$$

trong đó: α_s là những số từ 0 đến 9. Giả sử a là giá trị xấp xỉ của A với sai số tuyệt đối giới hạn Δ_a. Nếu $\Delta_a \le 0.5.10^s$ thì nói α_s là chữ số đáng tin, nếu $\Delta_a > 0.5.10^s$ thì nói α_s là chữ số đáng nghi.

- Nếu α_s là đáng tin thì tất cả những chữ số có nghĩa đứng bên trái nó cũng là đáng tin.
- Nếu α_s là đáng nghi thì tất cả những chữ số có nghĩa ở bên phải nó cũng là đáng nghi.

Cách viết số xấp xỉ:

Cách 1: Viết kèm theo sai số như công thức (1.2) hoặc (1.6).

Cách 2: Viết theo quy ước "Mọi chữ số có nghĩa đều đáng tin". Có nghĩa là sai số tuyệt đối giới hạn không lớn hơn một nửa đơn vị của hàng cuối cùng.

1.3. Sai số quy tròn

Hiện tượng quy tròn và sai số quy tròn: Trong tính toán khi gặp một số có quá nhiều chữ số đáng nghi người ta bỏ đi một vài chữ số cuối cho gọn, việc làm đó gọi là quy tròn số. Mỗi khi quy tròn một số người ta tạo ra một sai số mới gọi là sai số quy tròn, nó bằng hiệu giữa số quy tròn và số chưa quy tròn.

Quy tắc: Quy tròn sao cho sai số quy tròn tuyệt đối không lớn hơn một nửa đơn vị ở hàng được giữ lại cuối cùng, tức là 5 đơn vị ở hàng bỏ đi đầu tiên. Cụ thể là: Nếu chữ số bỏ đi đầu tiên không nhỏ hơn 5 thì thêm vào chữ số giữ lại cuối cùng một đơn vị, còn nếu chữ số bỏ đi đầu tiên nhỏ hơn 5 thì để nguyên chữ số giữ lại cuối cùng.

Sai số đã quy tròn:

$$\Delta'_a = \Delta_a + \theta_a. \tag{1.8}$$

Trong đó: Δ_a là sai số tuyệt đối giới hạn

θ_a. là sai số quy tròn tuyệt đối giới hạn

1.4. Sai số tính toán và sai số phương pháp

Khi giải gần đúng một bài toán phức tạp ta phải thay bài toán đã cho bằng một bài toán đơn giản hơn có thể giải được thông qua việc thực hiện các phép tính thông thường bằng tay hoặc trên máy tính. Phương pháp thay bài toán phức tạp bằng bài toán đơn giản như thế gọi là phương pháp gần đúng. Sai số do phương pháp gần đúng tạo ra gọi là ***sai số phương pháp***.

Khi giải các bài toán đơn giản, và thực hiện các phép tính thông thường ta luôn phải quy tròn các kết quả trung gian. Sai số tạo ra bởi tất cả các lần quy tròn như vậy gọi là ***sai số tính toán***.

* Sai số cuối cùng là tổng hợp của hai loại sai số phương pháp và sai số tính toán.

1.5. Các quy tắc tính sai số

Xét hàm số u của hai biến số x và y:

$$u = f(x,y) \tag{1.9}$$

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

Kí hiệu: Δx, Δy, Δu chỉ các số gia của x, y, u.

dx, dy, du chỉ các vi phân của x, y, u.

Δ_x, Δ_y, Δ_u là sai số tuyệt đối của x, y, u. Ta luôn có:

$$|\Delta x| \le \Delta_x; \quad |\Delta y| \le \Delta_y \tag{1.10}$$

Ta cần phải tìm Δ_u để ta có: $|\Delta u| \le \Delta_u$

Sai số của tổng u = x + y

$$\Delta u = \Delta x + \Delta y \tag{1.11}$$

ta suy ra: $|\Delta u| \le |\Delta x| + |\Delta y|$. Do đó ta có: $|\Delta u| \le \Delta_x + \Delta_y$, nếu ta chọn:

$$\Delta_{x+y} = \Delta_x + \Delta_y \tag{1.12}$$

để có $|\Delta u \le \Delta_u|$

Ta có quy tắc: "Sai số tuyệt đối giới hạn của một tổng bằng tổng các sai số tuyệt đối của các số hạng".

Sai số của tích u = x.y

$$\Delta_u \approx du = ydx + xdy \approx y\Delta x + x\Delta y$$

Suy ra: $|\Delta u| \le |y|.|\Delta x| + |x|.|\Delta y| \le |y|.\Delta_x + |x|.\Delta_y$.

Ta thu được: $\Delta_u = |y|.\Delta_x + |x|.\Delta_y$

Do đó: $\delta_u = \dfrac{\Delta_u}{|u|} = \left|\dfrac{|y|.\Delta_x + |x|\Delta_y}{|xy|}\right| = \dfrac{\Delta_x}{|x|} + \dfrac{\Delta_y}{|y|}$

Tức là ta có:

$$\delta_{xy} = \delta_x + \delta_y \tag{1.13}$$

* Quy tắc: "Sai số tương đối của một tích bằng tổng sai số tương đối của các thừa số của tích".

* Từ đó ta có:

$$\delta_{(x^n)} = n.\delta_x \quad (n \text{ nguyên dương}) \tag{1.14}$$

Sai số của thương $u = \dfrac{x}{y}$, $y \neq 0$

Tương tự ta có quy tắc: "Sai số tương đối của một thương bằng tổng các sai số tương đối của tử số và mẫu số".

$$\delta_{x/y} = \delta_x + \delta_y \tag{1.15}$$

Công thức tổng quát: Cho $u = f(x_1, \ldots, x_n)$ ta có sai số tuyệt đối:

$$\Delta_u = \sum_{i=1}^{n} \left|\frac{\partial f}{\partial x_i}\right| .\Delta_{x_i} \tag{1.16}$$

và từ đó suy ra sai số tương đối δ_u theo định nghĩa.

1.6. Sự ổn định của quá trình tính

Xét một quá trình tính vô hạn để tính ra một đại lượng nào đó.

* Ta nói quá trình tính là ổn định nếu sai số tính toán, tức là sai số quy tròn tích luỹ lại không tăng vô hạn.

* Nếu sai số đó tăng vô hạn thì ta nói quá trình tính là không ổn định.

Rõ ràng nếu quá trình tính không ổn định thì khó có hi vọng tính được đại lượng cần tính với sai số nhỏ hơn sai số cho phép. Cho nên trong tính toán, phức tạp nhất là các quá trình tính không ổn định. Để kiểm tra tính ổn định của quá trình tính thường người ta giả sử sai số chỉ xảy ra tại một bước, sau đó các phép tính đều làm đúng không có sai số. Nếu cuối cùng sai số tính toán không tăng vô hạn thì xem như quá trình tính là ổn định.

BÀI TẬP

1.1. Hãy xác định số các chữ số đáng tin trong các số sau, biết sai số:

a. $x = 2{,}36175$ với $\Delta_x = 0{,}34.10^{-2}$

b. $y = 42{,}5464$ với $\Delta_y = 0{,}3.10^{-1}$

c. $z = 153{,}128$ với $\delta_z = 0{,}06$.

1.2. Tính số e^2 với sai số tuyệt đối không quá 10^{-3}.

1.3. Tính giá trị của hàm số $y = \cos(x + 2)$ tại $x = 0{,}57$ với sai số tương đối không quá 0,1%.

§2. PHƯƠNG PHÁP GIẢI GẦN ĐÚNG PHƯƠNG TRÌNH VÀ HỆ PHƯƠNG TRÌNH

2.1. Nghiệm và khoảng phân li nghiệm

Nghiệm thực của phương trình một ẩn:

Xét phương trình một ẩn:

$$f(x) = 0 \tag{1.17}$$

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

Nghiệm thực của phương trình (1.17) là số thực α thoả mãn (1.17), tức là khi thay α vào x ở vế trái ta được:

$$f(\alpha) = 0 \tag{1.18}$$

Ý nghĩa hình học của nghiệm

Ta vẽ đồ thị của hàm số:

$$y = f(x) \tag{1.19}$$

Trong hệ toạ độ vuông góc Oxy (H.1.1). Giả sử đồ thị cắt trục hoành tại điểm M thì điểm M này có toạ độ $y = 0$ và hoành độ $x = \alpha$. Thay chúng vào (1.19) ta được:

$$0 = f(\alpha) \qquad (1.20)$$

Vậy hoành độ α của giao điểm M chính là một nghiệm của (1.17). Ta cũng có thể thay phương trình (1.17) bằng phương trình tương đương:

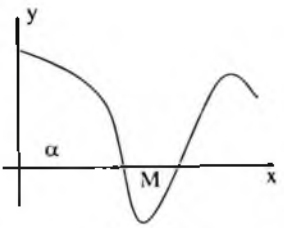
$$h(x) = g(x) \qquad (1.21)$$

rồi vẽ đồ thị của hai hàm số (H.1.2):

$$y = g(x) \text{ và } y = h(x) \quad (1.22)$$

Giả sử hai đồ thị ấy cắt nhau tại điểm M có hoành độ $x = \alpha$ thì ta có:

$$g(\alpha) = h(\alpha) \qquad (1.23)$$



Hình 1.1

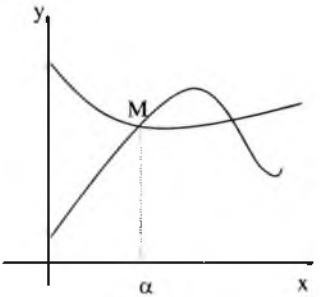
Vậy hoành độ của giao điểm của hai đồ thị (1.22) chính là một nghiệm của (1.21), tức là của (1.17).

Sự tồn tại nghiệm thực của phương trình (1.17).

Định lí: Nếu có hai số thực a và b (a < b) sao cho f(a) và f(b) trái dấu nhau, tức là:

$$f(a).f(b) < 0 \qquad (1.24)$$

đồng thời f(x) liên tục trên [a, b] thì ở trong khoảng [a, b] có ít nhất một nghiệm thực của phương trình (1.17).



Hình 1.2

Khoảng phân li nghiệm: Khoảng [a, b] nào đó gọi là khoảng phân li nghiệm của phương trình f(x) = 0 nếu nó chứa một và chỉ một nghiệm của phương trình đó.

* *Định lí 1:* Nếu [a, b] là một khoảng trong đó hàm số f(x) liên tục và đơn điệu, đồng thời f(a) và f(b) trái dấu thì [a, b] là một khoảng phân li nghiệm của phương trình f(x) = 0.

* *Định lí 2:* Nếu [a, b] là một khoảng trong đó hàm f(x) liên tục, đạo hàm f'(x) không đổi dấu và f(a).f(b) trái dấu thì [a, b] là một khoảng phân li nghiệm của phương trình f(x) = 0.

2.2. Phương pháp chia đôi

Nội dung của phương pháp: Giả sử [a, b] là khoảng phân li nghiệm của phương trình f(x) = 0. Ta chia đôi khoảng [a, b]

– Nếu $f\left(\frac{a+b}{2}\right)=0$ thì $\alpha=\frac{a+b}{2}$ là nghiệm của phương trình.

– Nếu $f\left(\frac{a+b}{2}\right)\neq 0$, ta chọn một trong hai khoảng $[a, \frac{a+b}{2}]$ và $[\frac{a+b}{2}, b]$ mà tại hai mút của khoảng hàm số f(x) có dấu khác nhau làm khoảng phân li nghiệm mới. Gọi khoảng này là $[a_1, b_1]$, nó có độ dài bằng nửa khoảng [a, b]:

$$b_1 - a_1 = \frac{1}{2}(b-a)$$

Lại chia đôi khoảng $[a_1, b_1]$ và tiếp tục làm như trên đến lần thứ n ta được:

$$b_n - a_n = \frac{1}{2^n}(b-a) \qquad (1.25)$$

Sự hội tụ của phương pháp: Nếu ta thực hiện vô hạn lần phương pháp chia đôi với khoảng [a, b] thì hoặc tại một lần nào đó điểm giữa của khoảng này là nghiệm đúng của phương trình $f(x) = 0$ hoặc ta nhận được một dãy vô hạn các khoảng chồng lên nhau và thu nhỏ dần $[a_1, b_1]$, $[a_2, b_2]$,..., $[a_n, b_n]$,..., sao cho:

$$f(a).f(b) < 0$$

và $b_n - a_n = \frac{1}{2^n}(b-a) \qquad (n = 1, 2, ...)$

Vì các nút trái $a_1, a_2, ..., a_n, ...$ tạo nên dãy đơn điệu không giảm và bị chặn trên bởi số b, còn các nút phải $b_1, b_2, ..., b_n, ...$ tạo nên dãy đơn điệu không tăng và bị chặn dưới bởi số a, nên khi $n \to +\infty$ từ (1.25) ta nhận được:

$$\lim_{n\to+\infty} a_n = \lim_{n\to+\infty} b_n = \alpha$$

Cho $n \to +\infty$ trong bất đẳng thức (1.24), do sự liên tục của hàm số f(x) ta có:

$$|f(\alpha)|^2 \leq 0$$

Vậy $f(\alpha) = 0$ và α là nghiệm của phương trình.

Đánh giá sai số của nghiệm gần đúng: Trong thực hành ta không thể thực hiện phương pháp chia đôi vô hạn lần để nhận được nghiệm đúng của phương trình mà chỉ có thể áp dụng n lần phương pháp chia đôi, với n là một số nguyên dương hữu hạn. Dừng lại ở lần thứ n ta có :

$$a_n \leq \alpha \leq b_n \text{ và } b_n - a_n = \frac{1}{2^n}(b-a)$$

Vậy có thể lấy nghiệm gần đúng là:

a) a_n, khi đó sai số của nghiệm gần đúng là: $|a_n - \alpha| \le b_n - a_n = \frac{1}{2^n}(b-a)$

b) b_n, khi đó sai số của nghiệm gần đúng là: $|b_n - \alpha| \le b_n - a_n = \frac{1}{2^n}(b-a)$

c) $\frac{a_n + b_n}{2}$, khi đó sai số của nghiệm gần đúng là:

$$\left|\frac{a_n + b_n}{2} - \alpha\right| \le \frac{1}{2}(b_n - a_n) = \frac{1}{2^{n+1}}(b-a)$$

Phương pháp này có ưu điểm là đơn giản, dễ lập chương trình chạy máy tính nhưng lại có nhược điểm là tốc độ hội tụ chậm.

2.3. Phương pháp lặp

Mô tả phương pháp: Giả sử phương trình f(x) = 0 có nghiệm thực α phân li trong khoảng [a, b]. Trước hết ta chuyển phương trình f(x) = 0 về dạng:

$$x = \varphi(x) \qquad (1.26)$$

Sau đó ta chọn một số x_o nào đó $\in [a, b]$ làm xấp xỉ đầu rồi tính dần dãy số x_n theo quy tắc:

$$x_n = \varphi(x_{n-1}) \qquad n = 1, 2,... \qquad (1.27)$$

Quá trình tính này lặp đi lặp lại nên phương pháp ở đây là phương pháp lặp và φ(x) là hàm lặp.

Sự hội tụ của phương pháp lặp: Nếu dãy $x_n \to \alpha$ khi $n \to +\infty$ thì ta nói phương pháp lặp (1.26), (1.27) hội tụ.

Định lí 3: Xét phương pháp lặp (1.26), (1.27) giả sử:

1) [a, b] là khoảng phân li nghiệm α của phương trình f(x) = 0

2) Mọi x_n tính theo (1.26), (1.27) đều $\in [a, b]$

3) Hàm φ(x) có đạo hàm thoả mãn

$$|\varphi'(x)| \le q < 1, \quad a < x < b \qquad (1.28)$$

trong đó q là một hằng số. Thế thì phương pháp lặp (1.26), (1.27) hội tụ.

* Nếu $\varphi'(x) > 0$ ta có thể chọn $x_0 \in [a, b]$ một cách bất kì, nếu $\varphi'(x) < 0$ thì phải chọn x_0 theo quy tắc sau:

$x_0 = a$ khi $a < \alpha < c = \dfrac{a+b}{2}$

$x_0 = b$ khi $\dfrac{a+b}{2} = c < \alpha < b$ (1.29)

Muốn biết α thuộc nửa khoảng nào ta chỉ việc tính $f(c)$ rồi so sánh dấu của nó với dấu của $f(\alpha)$

Đánh giá sai số

+ Công thức đánh giá sai số thứ nhất

$$|\alpha - x_n| \le \frac{q}{1-q}|x_n - x_{n-1}| \quad (0 \le q < 1) \tag{1.30}$$

+ Công thức đánh giá sai số thứ hai

Định lí 4: Xét phương trình $f(x) = 0$ có nghiệm $x \in [a, b]$ và $\bar{x}$ được xem là giá trị gần đúng của x. Lúc đó ta có:

$$|x - \bar{x}| \le \frac{|f(\bar{x})|}{m} \tag{1.31}$$

trong đó m là một số dương thoả mãn: $|f'(x)| \ge m > 0$.

BÀI TẬP

1.4. Giải gần đúng phương trình: $\sin x + x^2 = 1$ bằng phương pháp chia đôi với kết quả có hai chữ số đáng tin.

1.5. Giải gần đúng phương trình: $\cos x - x = -0{,}15$ bằng phương pháp chia đôi với kết quả có ba chữ số đáng tin.

§3. PHƯƠNG PHÁP NỘI SUY HÀM SỐ

3.1. Đa thức nội suy

Vấn đề nội suy: Trong thực tế nhiều khi phải phục hồi một hàm số $f(x)$ tại mọi giá trị của x trên đoạn $[a, b]$ mà chỉ biết một số hữu hạn giá trị của hàm số tại một số hữu hạn các điểm rời rạc của đoạn đó. Các giá trị đó được cung cấp qua thực nghiệm hay tính toán. Vậy nảy sinh một vấn đề toán học sau: Trên đoạn $[a, b]$ cho một lưới các điểm chia x_i, $i = 0, 1, ..., n$: $a \le x_0, x_1, ..., x_n \le b$ và tại các nút x_i cho giá trị của hàm số $y = f(x)$ là:

$$y_i = f(x_i), \quad i = 0, 1, ..., n \tag{1.32}$$

Ta viết thành bảng sau:

x	x_0	x_1	.....	x_n
y	y_0	y_1	.....	y_n

Hãy xây dựng một đa thức bậc n: $p_n(x) = a_0x^n + a_1.x^{n-1} + ... + a_{n-1}.x + a_n$

Sao cho $p_n(x_i) = y_i$, i = 0, 1,.... n. Đa thức $p_n(x)$ gọi là đa thức nội suy của hàm f(x). Ta chọn đa thức nội suy hàm f(x) vì đa thức là loại hàm đơn giản, luôn có đạo hàm và nguyên hàm, việc tính giá trị cũng đơn giản. Ta có:

$$p_n(x) = (...(a_0.x + a_1)x + a_2)...) + a_n$$

Sơ đồ Hoocne tính giá trị $p_n(c)$:

$$\begin{cases} b_0 = a_0, \\ b_1 = b_0.c + a_1, \\ b_2 = b_1.c + a_2, \\ ..... \\ b_n = b_{n-1}.c + a_n = p_n(c) \end{cases} \tag{1.33}$$

Sự duy nhất của đa thức nội suy:

Định lí 5: Đa thức nội suy $p_n(x)$ của hàm số f(x) định nghĩa như trên nếu có thì chỉ có một mà thôi.

* Như vậy ta thấy đa thức nội suy có thể xây dựng nhiều cách, nhưng vì nó có tính duy nhất nên tất cả các dạng của nó đều có thể quy về nhau được.

3.2. Đa thức nội suy Lagrange

Thành lập đa thức nội suy Lagrange:

$$l_i(x) = \frac{(x - x_0)....(x - x_{i-1})(x - x_{i+1})...(x - x_n)}{(x_i - x_0)...(x_i - x_{i-1})(x_i - x_{i+1})...(x_i - x_n)}$$

Rõ ràng $l_i(x)$ là đa thức bậc n và ta có :

$$l_i(x_j) = \begin{cases} 1 & \text{khi } j = i \\ 0 & \text{khi } j \neq i \end{cases} \tag{1.34}$$

Ta gọi $l_i(xj)$ là đa thức Lagrange cơ bản. Bây giờ xét biểu thức:

$$p_n(x) = \sum_{i}^{n} y_i l_i(x) \tag{1.35}$$

Ta thấy $p_n(x)$ vừa là một đa thức bậc n, vừa thoả mãn: $p_n(x_i) = y_i$. Ta gọi nó là đa thức nội suy Lagrange.

Nội suy bậc nhất: Với n = 1 đa thức nội suy sẽ là:

$$\begin{cases} p_1(x) = y_o l_o(x) + y_1 l_1(x) \\ l_0(x) = \dfrac{x - x_1}{x_o - x_1}, l_1(x) = \dfrac{x - x_o}{x_1 - x_o} \end{cases} \quad (1.36)$$

Do đó: $p_1(x) = y_o.\dfrac{x - x_1}{x_o - x_1} + y_1.\dfrac{x - x_o}{x_1 - x_o}$

Đa thức $p_1(x)$ là bậc nhất với x có dạng: $Ax + B$

Nội suy bậc hai: Với n = 2 đa thức nội suy có dạng là:

$$p_2(x) = y_o l_o(x) + y_1 l_1(x) + y_2 l_2(x) \quad (1.37)$$

Trong đó : $l_o(x) = \dfrac{(x - x_1).(x - x_2)}{(x_o - x_1).(x_o - x_2)}$,

$$l_1(x) = \frac{(x - x_o).(x - x_2)}{(x_1 - x_o).(x_1 - x_2)},$$

$$l_2(x) = \frac{(x - x_o).(x - x_1)}{(x_2 - x_o).(x_2 - x_1)}.$$

Đa thức $p_2(x)$ là một đa thức bậc hai đối với x có dạng $Ax^2 + Bx + C$.

Sai số nội suy:

Định lí 6: Nếu hàm f(x) liên tục trên [a, b] và có trong (a, b) đạo hàm liên tục đến cấp n + 1 thì sai số nội suy có biểu thức :

$$r_n(x) = f^{(n+1)}(c).\frac{\Pi(x)}{(n+1)!}, c \in [a, b] \quad (1.38)$$

trong đó:

$$\Pi(x) = (x - x_o).(x - x_1)....(x - x_n) \quad (1.39)$$

3.3. Đa thức Niutơn

Đa thức nội suy Niutơn tổng quát:

+ Xây dựng đa thức nội suy Niutơn xuất phát từ nút x_o.

Giả sử hàm $y = f(x)$ có bảng giá trị:

x	x_0	x_1	.....	x_n
y	y_o	y_1	.....	y_n

Tỉ hiệu cấp một của y tại x_i, x_j là: $y[x_i, x_j] = \dfrac{y_i - y_j}{x_i - x_j}$

Tỉ hiệu cấp hai của y tại x_i, x_j, x_k là: $y[x_i, x_j, x_k] = \dfrac{(y[x_i, x_j] - y[x_j, x_k])}{(x_i - x_k)}$ ...

Các tỉ hiệu có tính chất đối xứng: $y[x_i, x_j] = y[x_j, x_i]$

và $y[x_i, x_j, x_k] = y[x_k, x_j, x_i]$

Với $y(x) = p_n(x)$ là một đa thức bậc n thì tỉ hiệu cấp một tại x, x_o là :

$$p_o[x, x_o] = \frac{p_n(x) - p_n(x_o)}{(x - x_o)}$$

là một đa thức bậc n – 1. Tỉ hiệu cấp hai tại x, x_o, x_1 là:

$$p_n[x, x_o, x_1] = \frac{p_n[x, x_o] - p_n[x_o, x_1]}{(x - x_1)}$$

là một đa thức bậc n – 2. Tiếp tục xây dựng tới cấp n + 1 thì:

$$p_n[x, x_o, x_1, ..., x_n] = 0$$

Từ định nghĩa tỉ hiệu ta suy ra :

$$p_n(x) = p_n(x_o) + (x - x_0).p_n[x, x_o]$$

$$p_n[x, x_o] = p_n[x_o, x_1] + (x - x_1).p_n[x, x_o, x_1]$$

$$p_n[x, x_o, x_1] = p_n[x_o, x_1, x_2] + (x - x_2).p_n[x, x_o, x_1, x_2]$$

.....

$$p_n[x, x_o, x_1, ..., x_{n-1}] = p_n[x_o, x_1, x_2, ..., x_n] + (x - x_n).p_n[x, x_o, x_1, x_2, ..., x_n]$$

Vì $p_n[x, x_o, x_1, ..., x_n] = 0$, nên từ đó ta có:

$$p_n(x) = p_n(x_o) + (x - x_o).p_n[x_o, x_1] + (x - x_o).(x - x_1).p_n[x, x_o, x_1, x_2] + ... + (x - x_o).(x - x_1)....(x - x_{n-1})p_n[x, x_o, ..., x_n] \quad (1.40)$$

Nếu $P_n(x)$ là đa thức nội suy của hàm $y = f(x)$ thì: $P_n(x_i) = f(x_i)$, $i = 0, 1, ..., n$. Và ta có:

$$p_n(x) = y(x_o) + (x - x_o).y[x_o, x_1] + (x - x_o).(x - x_1).y[x, x_o, x_1, x_2] + ... + (x - x_o).(x - x_1)....(x - x_{n-1})y[x, x_o, ..., x_n] \quad (1.41)$$

Đa thức này gọi là *đa thức Niutơn tiến* xuất phát từ nút x_0 của hàm $y = f(x)$.

+ *Xây dựng đa thức nội suy Niutơn xuất phát từ nút x_n:* Tương tự cách xây dựng như trên ta có:

$$\begin{aligned} p_n(x) &= y(x_n) + (x - x_n).y[x_n, x_{n-1}] \\ &+ (x - x_n).(x - x_{n-1}).y[x, x_n, x_{n-1}, x_{n-2}] + \ldots \\ &+ (x - x_n).(x - x_{n-1}) \ldots (x - x_1) y[x, x_n, \ldots, x_0] \end{aligned} \quad (1.42)$$

Đây là *đa thức Niutơn lùi* xuất phát từ nút x_n của hàm $y = f(x)$

Trường hợp các nút cách đều

$x_i = x_0 + i.h, \quad i = 0, \ldots, n.$

Khi đó h gọi là bước nội suy.

+ Khái niệm sai phân tiến:

Sai phân tiến cấp một tại i: $\Delta y_i = y_{i+1} - y_i$

Sai phân tiến cấp hai tại i: $\Delta^2 y_i = \Delta(\Delta y_i) = y_{i+2} - 2.y_{i+1} + y_i$

.....

Sai phân tiến cấp n tại i là: $\Delta^n y_i = \Delta(\Delta^{n-1} y_i)$

Khi đó ta có : $y[x_0, x_1] = \dfrac{\Delta y_0}{h}$

$$y[x_0, x_1, x_2] = \frac{\Delta^2 y_0}{2.h^2}$$

.....

$$y[x_0, x_1, \ldots, x_n] = \frac{\Delta^n y_0}{n! h^n}$$

Bây giờ đặt $x = x_0 + h.t$ trong đa thức Niutơn tiến (1.41) ta được :

$$p_n(x)\big|_{x = x_0 + h.t} = y_0 + t.\Delta y_0 + \frac{t(t-1)}{2!}\Delta^2 y_0 + \ldots + \frac{t.(t-1)\ldots(t-n+1)}{n!}\Delta^n y_0 \quad (1.43)$$

gọi là đa thức Niutơn tiến xuất phát từ x_0 trong trường hợp các nút cách đều.

+ *Khái niệm sai phân lùi:* Hoàn toàn tương tự như phân sai tiến ta có:

$$\Delta y_i = y_i - y_{i-1}$$

.....

$$\Delta^n y_i = \Delta(\Delta^{n-1} y_i)$$

Ta có đa thức nội suy Niutơn lùi xuất phát từ x_n trong trường hợp nút cách đều:

$$p_n(x)\big|_{x=x_0+ht} = y_0 + t.\Delta y_0 + \frac{t(t-1)}{2!}\Delta^2 y_0 + ... + \frac{t.(t-1)...(t-n+1)}{n!}\Delta^n y_0 \quad (1.44)$$

BÀI TẬP

1.6. Tính tích phân: $\int_0^1 \sin(x^2).dx$ bằng cách lập đa thức nội suy gồm 6 điểm cách đều tính từ điểm x = 0 của hàm $\sin(x^2)$ rồi tính tích phân.

1.7. Một vật khối lượng $m = 10^{-2}$ (kg) được gắn vào con lắc lò xo có độ cứng k = 200 N/m. Đầu kia của con lắc được gắn chặt vào tường, hệ dao động dọc theo trục Ox trên một mặt phẳng nhẵn nằm ngang với hệ số ma sát $\eta = 0,05$. Hệ tọa độ được chọn sao cho vật m khi ở trạng thái cân bằng nằm trùng với gốc tọa độ. Tại thời điểm ban đầu người ta cung cấp một động năng cho vật, sau đó đo đạc thực nghiệm xác định vị trí của vật có kết quả theo bảng sau:

t (s)	0	0,3	0,7	0,9	1,2
x (cm)	0	0,7	1,95	1,8	0,3

Xác định vị trí, vận tốc, gia tốc của vật tại thời điểm t = 1s.

§4. PHƯƠNG PHÁP TÍNH GẦN ĐÚNG ĐẠO HÀM TÍNH TÍCH PHÂN - TÍNH VI PHÂN

4.1. Tính gần đúng đạo hàm

Áp dụng đa thức nội suy: Để tính gần đúng đạo hàm của hàm f(x) tại x tức là f'(x) ta có thể thay hàm f(x) bằng đa thức nội suy p(x) rồi tính đạo hàm của đa thức nội suy: p'(x), lấy p'(x) là giá trị gần đúng của f'(x).

Cách tính này chỉ có ý nghĩa khi các $|x_{i+1} - x_i|$ nhỏ vì nếu không thì sai số có thể rất lớn.

Áp dụng công thức Taylor: Theo công thức Taylor ta có:

$$f(x+h) = f(x) + h.f'(x) + \frac{h^2}{2!}f''(c) \qquad (c = x + \theta.h,\ 0 < \theta < 1)$$

Khi $|h|$ bé thì số hạng cuối ở vế phải rất bé, ta có thể bỏ qua và có:

$$f(x+h) - f(x) \approx h.f'(x) \tag{1.45}$$

Vậy ta có: $f'(x) \approx \dfrac{f(x+h) - f(x)}{h}$ (1.46)

4.2. Tính gần đúng tích phân xác định

Xét tích phân xác định: $I = \int_a^b f(x)dx$

Nếu f(x) liên tục trên [a, b] và có nguyên hàm là F(x) thì công thức Niutơn – Lépnit cho:

$$\int_a^b f(x)dx = F(b) - F(a)$$

Nhưng nếu không tìm được nguyên hàm của f(x) ở dạng sơ cấp hoặc nguyên hàm đó quá phức tạp thì tích phân I phải tính gần đúng. Sau đây ta sẽ trình bày hai công thức tính gần đúng tích phân I dựa trên tư tưởng thay hàm f(x) bằng một đa thức nội suy.

Công thức hình thang: Chia [a, b] thành n đoạn cong bằng nhau bởi các điểm chia x_i: $a = x_0 < x_1 < ... < x_{n-1} < x_n = b$ ($x_i = a + i.h$, $h = \dfrac{(b-a)}{n}$, i = 0, 1,..., n).

Đặt $y_i = f(x_i)$ ta có:

$$\int_a^b f(x)dx = \int_{x_0}^{x_1} fdx + \int_{x_1}^{x_2} fdx + ... + \int_{n-1}^{n} fdx \tag{1.47}$$

Để tính tích phân ở vế phải ta thay hàm f(x) bằng một đa thức nội suy bậc nhất $p_1(x)$. Với tích phân thứ nhất ta có:

$$\int_{x_0}^{x_1} f(x)dx = \int_{x_0}^{x_1} p_1(x)dx$$

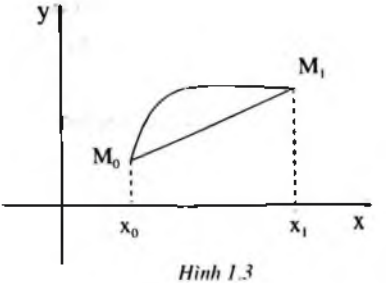
Đổi biến $x = x_0 + h.t$, ứng với x_0 là t = 0, ứng với x_1 là t = 1 nên có:

$$\int_{x_0}^{x_1} p_1(x)dx = h.\int_0^1 (y_0 + t.\Delta y_0)dt = h.\left(y_0 t + \frac{t^2}{2}\Delta y_0\right)\Big|_{t=0}^{t=1} = h.\left[y_0 + \frac{\Delta y_0}{2}\right]$$

$$= h.\frac{y_0 + y_1}{2}$$

Vậy có:

$$\int_{x_0}^{x_1} f(x)dx \approx \int_{x_0}^{x_1} p_1(x)dx = h.\frac{y_0 + y_1}{2}$$



Hình 1.3

Về mặt hình học có điều đó có nghĩa: thay diện tích hình thang cong $x_0M_0M_1x_1$ bởi diện tích hình thang thường $x_0M_0M_1x_1$

Đối với tích phân thứ i + 1 ta có:

$$\int_{x_i}^{x_{i+1}} f(x)dx \approx h.\frac{y_i + y_{i+1}}{2}$$

Vậy công thức (1.47) cho:

$$\int_a^b f(x)dx = \frac{h}{2}.\left[(y_0 + y_1) + (y_1 + y_2) + ... + (y_{n-1} + y_n)\right]$$

$$= h.\left[\frac{(y_0 + y_n)}{2} + y_1 + y_2 + ... + y_{n-1}\right], \qquad (1.48)$$

trong đó: $h = \frac{b-a}{n}$

Công thức này gọi là công thức hình thang.

Đánh giá sai số: Người ta chứng minh được:

$$|I - I_T| \le \frac{M}{12}.h^2(b-a)$$
$$M = \max|f''(x)|, a \le x \le b \qquad (1.49)$$

Công thức Simxơn: Ta chia [a, b] thành 2n đoạn cong bằng nhau bởi các điểm chia x_i:

$$a = x_0 < x_1 < ... < x_{2n} = b$$

$$x_i = a + i.h,\ h = \frac{b-a}{2n},\ i = 0, 1, ..., 2n.$$

Giả sử $y_i = f(x_i)$. Ta có:

$$\int_a^b f(x)dx = \int_{x_0}^{x_2} f dx + \int_{x_2}^{x_4} f dx + ... + \int_{x_{2n-2}}^{x_{2n}} f dx \qquad (1.50)$$

Để tính mỗi tích phân ở vế phải ta thay f(x) bằng đa thức nội suy bậc hai. Với tích phân thứ nhất ta có: $\int_{x_0}^{x_2} f(x)dx = \int_{x_0}^{x_2} p_1(x)dx$

Đổi biến $x = x_0 + h.t$ thì $dx = h.dt$, ứng với x_0 là $t = 0$, ứng với x_2 là $t = 2$. Do đó:

$$\int_{x_0}^{x_2} p_2(x)dx = h.\int_0^2 (y_0 + t\Delta y_0 + \frac{t(t-1)}{2}\Delta^2 y_0)dt = \frac{h}{3}(y_0 + 4y_1 + y_2)$$

Vậy ta có:

$$\int_{x_0}^{x_2} f(x)dx = \frac{h}{3}(y_0 + 4y_1 + y_2)$$

Đối với tích phân sau cũng có cách tính tương tự:

$$\int_{x_{2i}}^{x_{2i+2}} f(x)dx = \frac{h}{3}(y_{2i} + 4y_{2i+1} + y_{2i+2})$$

Vậy ta cũng được:

$$I = \frac{h}{3}.[(y_0 + y_{2n}) + 4(y_1 + y_3 + ... + y_{2n-1}) + 2(y_2 + y_4 + ... + y_{2n-2})], \quad (1.51)$$

với $h = \frac{b-a}{2n}$

Công thức này gọi là *công thức Simxơn*.

Đánh giá sai số: Người ta chứng minh được:

$$|I - I_S| \le \frac{M}{180}.h^4(b-a)$$
$$M = \max|f^{(4)}(x)|, a \le x \le b \quad (1.52)$$

4.3. Phương pháp giải phương trình vi phân

Bài toán Côsi đối với phương trình vi phân cấp một: Cho khoảng $[x_0, X]$. Tìm hàm số $y = y(x)$ xác định trên $[x_0, X]$ và thoả mãn:

$$y' = f(x,y). \quad x_0 \le x \le X \quad (1.53)$$

$$y(x_0) = \eta \quad (1.54)$$

trong đó $f(x, y)$ là một hàm số đã biết của hai đối số x, y còn η là một số thực cho trước.

Phương pháp chuỗi Taylor: Ta tìm nghiệm y(x) khai triển thành chuỗi Taylor tại $x = x_0$:

$$y(x)=y(x_0)+\frac{y'(x_0)}{1!}.(x-x_0)+\frac{y''(x_0)}{2!}(x-x_0)^2+...+\frac{y^{(k)}}{k!}(x-x_0)^k+... \quad (1.55)$$

Tính các đạo hàm $y^{(k)}(x_0)$ của y tại x_0 ta có:

$$y'(x_0) = f(x_0, y(x_0)) \quad (1.56)$$

$$y'(x) = (y')' = (f(x, y(x)))' = \frac{\partial f}{\partial x}(x, y(x)) + \frac{\partial f}{\partial y}(x, y(x)).y'(x) \quad (1.57)$$

Thay $x = x_0$ và $y(x_0) = \eta$ ta được:

$$y''(x_0) = (y')' = (f(x_0, \eta))' = \frac{\partial f}{\partial x}(x_0, \eta) + \frac{\partial f}{\partial y}(x_0, \eta).f(x_0, \eta)$$

Một cách tương tự, để tính $y'''(x_0)$: trước hết ta phải lấy đạo hàm (1.57) sau đó thay $x = x_0$ và cứ thế tiếp tục.

Với x khá gần x_0 thì chuỗi Taylor hội tụ về nghiệm của bài toán và độ chính xác càng cao khi n càng lớn.

Phương pháp Ôle: Chia đoạn $[x_0, X]$ thành n đoạn nhỏ bằng nhau bởi các điểm x_i

$$x_i = x_0 + i.h, \ i = 0, 1, ..., n$$

$$h = \frac{X - x_0}{n} \quad (1.58)$$

Phương pháp Ôle cho phép tìm cách tính gần đúng giá trị của y(x) chỉ tại các nút x_i mà thôi, chứ không phải tại mọi $x \in [x_0, X]$.

* Gọi y(x) là nghiệm của bài toán Côsi và $y(x_i)$ là giá trị của $y(x_i)$ tại x_i

u_i là giá trị gần đúng của $y(x_i)$ mà ta muốn tính.

Sau đây ta xây dựng công thức tính u_i.

Giả sử đã biết u_i tại nút x_i và muốn tính u_{i+1} tại nút x_{i+1}. Khai triển Taylor:

$$y(x)=y(x_i)+\frac{y'(x_i)}{1!}.(x-x_i)+\frac{y''(c_i)}{2!}(x-x_i)^2 \qquad c_i = x_i + \theta(x - x_i), \ 0 < \theta < 1$$

Thay $x = x_{i+1} = x_i + h$ và $y'(x_i) = f(x_i, y(x_i))$, ta được:

$$y(x_{i+1}) = y(x_i) + h.\frac{y'(x_i)}{1!} + h^2.\frac{y''(c_i)}{2!} \quad (1.59)$$

Khi h bé, số hạng ở cuối vế phải có thể xem là bé, không đáng kể ta bỏ qua và thay $y(x_i)$ bằng u_i ta được công thức:

$$u_{i+1} = u_i + h_i.f(x_i, u_i) \quad (1.60)$$

Công thức này cho phép tính u_{i+1} khi biết u_i và điều kiện Côsi đặt $u_0 = \eta$.

Sự hội tụ của phương pháp Ôle: Ta gọi $e_i = u_i - y(x_i)$ là sai số phương pháp Ôle. Nếu tại x_i xác định, $e_i \to 0$ khi $h \to 0$, tức là $u_i \to y(x_i)$ khi $h \to 0$ thì ta nói phương pháp Ôle hội tụ.

4.4. Phương pháp giải hệ phương trình vi phân

Hệ phương trình: Cho khoảng $[x_0, X]$. Tìm hai hàm số $y = y(x)$ và $z = z(x)$ xác định trên đoạn $[x_0, X]$ và thoả mãn

$$y' = f(x,y,z),\ z' = g(x,y,z) \quad (1.61)$$

$$y(x_0) = \eta_1,\ z(x_0) = \eta_2 \quad (1.62)$$

trong đó: η_1, η_2 là hai số thực cho trước.

Phương pháp chuỗi Taylor: Giống như đối với phương trình ta cũng có thể mở rộng áp dụng cho bài toán hệ. Với phương pháp Taylor ta viết:

$$y(x) = y(x_0) + \frac{y'(x_0)}{1!}.(x-x_0) + \frac{y''(x_0)}{2!}(x-x_0)^2 + ... + \frac{y^{(k)}}{k!}(x-x_0)^k + ...$$

$$z(x) = z(x_0) + \frac{z'(x_0)}{1!}.(x-x_0) + \frac{z''(x_0)}{2!}(x-x_0)^2 + ... + \frac{z^{(k)}}{k!}(x-x_0)^k + ...$$

Với: $y(x_0) = \eta_1,\ z(x_0) = \eta_2$

$y'(x_0) = f(x_0, y(x_0), z(x_0)) = f(x_0, \eta_1, \eta_2)$

$z'(x_0) = g(x_0, y(x_0), z(x_0)) = g(x_0, \eta_1, \eta_2)$

Phương pháp Ôle: Trước hết ta chia đoạn $[x_0, X]$ thành n đoạn con, giả thiết bằng nhau cho đơn giản, bởi các điểm:

$$x_i = x_0 + i.h,\quad i = 0, 1, ..., n$$

$$h = \frac{1}{n}$$

Phương pháp Ôle viết: $u_{i+1} = u_i + h_i.f(x_i, u_i, v_i)$

$$v_{i+1} = v_i + h_i g(x_i, u_i, v_i)$$

$$u_0 = \eta_1, v_0 = \eta_2$$

Như vậy biết u_i, v_i tính được ngay u_{i+1}, v_{i+1}.

BÀI TẬP

1.8. Cho tích phân

$$I = \int_0^1 \frac{dx}{2+\cos x}$$

Bằng cách chia đoạn [0, 1] thành 10 đoạn bằng nhau, hãy tính gần đúng I theo

a. Đa thức nội suy.

b. Công thức hình thang.

c. Công thức Simxơn.

1.9. Tính tích phân

$$I = \int_0^1 e^{-2x^2} dx$$

bằng cách khai triển hàm dưới dấu tích phân thành chuỗi luỹ thừa.

§5. MỘT SỐ PHƯƠNG PHÁP GẦN ĐÚNG TRONG VẬT LÍ LƯỢNG TỬ

Khi nghiên cứu hệ lượng tử có tương tác thì Hamiltonian của hệ lượng tử bao gồm số hạng không tương tác và số hạng tương tác. Các bài toán có liên quan đến số hạng tương tác hầu hết không thể giải được một cách chính xác. Các số hạng tương tác này gọi là các nhiễu loạn gây bởi trường ngoài hay do tương tác của chính hệ lượng tử gây ra.

5.1. Phương pháp nhiễu loạn đối với trạng thái dừng không suy biến

Giả sử Hamiltonian của hệ lượng tử có thể viết dưới dạng:

$$\hat{H} = \hat{H}_0 + \lambda \hat{H}_1 \qquad (1.63)$$

trong đó: + $\hat{H}_0$ là Hamiltonian của hệ ở trạng thái không có nhiễu loạn, với trị riêng $E_n^{(0)}$ và vectơ riêng $|\Phi_n\rangle$ không suy biến đã được biết chính xác:

$$\hat{H}_0|\Phi_n\rangle = E_n^{(0)}|\Phi_n\rangle \tag{1.64}$$

+ $\hat{H}_1$ là thành phần nhiễu loạn, λ là thông số nhiễu loạn.

Kí hiệu các vectơ riêng của $\hat{H}$ là $|\Psi_n\rangle$ ứng với trị riêng E_n, ta có:

$$(\hat{H}_0 + \lambda\hat{H}_1)|\Psi_n\rangle = E_n|\Psi_n\rangle \tag{1.65}$$

Khai triển $|\Psi_n\rangle$ theo các vectơ trạng thái riêng $|\Phi_n\rangle$ và giả thiết có sự tương ứng giữa các trạng thái $|\Psi_n\rangle$ và $|\Phi_n\rangle$. Nghĩa là khi $\lambda \to 0$ ta có sự tương ứng sau:

$$|\Psi_n\rangle \to |\Phi_n\rangle \text{ và } E_n \to E_n^{(0)}$$

$$|\Psi_n\rangle = |\Phi_n\rangle + \sum_{k\neq n} c_{nk}(\lambda)|\Phi_k\rangle \tag{1.66}$$

Từ điều kiện $\lambda \to 0$ ta có $|\Psi_n\rangle \to |\Phi_n\rangle$ dẫn đến $c_{nk}(0) = 0$. Các hệ số khai triển c_{nk} và trị riêng E_n cũng được khai triển chuỗi theo λ.

$$c_{nk}(\lambda) = \lambda.c_{nk}^{(1)} + \lambda^2.c_{nk}^{(2)} + \lambda^3.c_{nk}^{(3)} + ...$$

$$E_n = E_n^{(0)} + \lambda.E_n^{(1)} + \lambda^2.E_n^{(2)} + \lambda^3.E_n^{(3)} + ... \tag{1.67}$$

Thay các khai triển vào phương trình Schrodinger ta được:

$$(\hat{H}_0 + \lambda.\hat{H}_1)\left[|\Phi_n\rangle + \sum_{k\neq n}\lambda.c_{kn}^{(1)} + \sum_{k\neq n}\lambda^2.c_{kn}^{(2)} + ...\right]$$

$$= \left(E_n^{(0)} + \lambda.E_n^{(1)} + \lambda^2.E_n^{(2)}\right)\left[|\Phi_n\rangle + \sum_{k\neq n}\lambda.c_{kn}^{(1)} + \sum_{k\neq n}\lambda^2.c_{kn}^{(2)} + ...\right] \tag{1.68}$$

Đây là phương trình thoả mãn với λ bất kì, các hệ số mỗi bậc của λ ở cả hai vế của phương trình phải bằng nhau. Trong gần đúng bậc một, chỉ giữ lại các số hạng tỉ lệ với λ, phương trình (1.68) có dạng:

$$\hat{H}_0\sum_{k\neq n}c_{kn}^{(1)}|\Phi_k\rangle + \hat{H}_1|\Phi_n\rangle = E_n^{(0)}\sum_{k\neq n}c_{kn}^{(1)}|\Phi_k\rangle + E_n^{(1)}|\Phi_n\rangle \tag{1.69}$$

Vì: $\hat{H}_0|\Phi_k\rangle = E_k^{(0)}|\Phi_k\rangle$

nên phương trình (1.69) được viết lại:

$$E_n^{(1)}|\Phi_n\rangle = \hat{H}_1|\Phi_n\rangle + \sum\left(E_k^{(0)} - E_n^{(0)}\right)c_{nk}^{(1)}|\Phi_k\rangle \tag{1.70}$$

Nhân hai vế của (1.70) với vectơ bra $\langle\Phi_n|$ và sử dụng điều kiện trực giao của các vectơ trạng thái không nhiễu loạn $\langle\Phi_n|\Phi_k\rangle = \delta_{nk}$ ta tìm được phần bổ chính bậc một của năng lượng:

$$E_n^{(1)} = \langle\Phi_n|\hat{H}_1|\Phi_n\rangle \tag{1.71}$$

Tương tự, nhân trái hai vế của phương trình (1.70) với $\langle\Phi_m|$ $(m \neq n)$ và sử dụng điều kiện trực giao của vectơ trạng thái không nhiễu loạn ta được:

$$\langle\Phi_m|\hat{H}_1|\Phi_n\rangle + (E_m^{(0)} - E_n^{(0)}).c_{nm}^{(1)} = 0$$

Ta suy ra hệ số khai triển trong gần đúng bậc một:

$$c_{nk}^{(1)} = \frac{\langle\Phi_k|\hat{H}_1|\Phi_n\rangle}{E_n^{(0)} - E_k^{(0)}} \tag{1.72}$$

do đó trạng thái trong gần đúng bậc một có dạng:

$$|\Psi_n\rangle = |\Phi_n\rangle + \sum_{k\neq n}\frac{\langle\Phi_k|\hat{H}_1|\Phi_n\rangle}{E_n^{(0)} - E_k^{(0)}}|\Phi_k\rangle \tag{1.73}$$

Bỏ qua các số hạng chứa luỹ thừa của λ bậc ba trở lên, ta được:

$$\hat{H}_0\sum_{k\neq n}c_{nk}^{(2)}|\Phi_k\rangle + \hat{H}_1\sum c_{nk}^{(1)}|\Phi_k\rangle = E_n^{(0)}\sum c_{nk}^{(2)}|\Phi_k\rangle + E_n^{(1)}\sum c_{nk}^{(1)}|\Phi_k\rangle + E_n^{(2)}|\Phi_n\rangle \tag{1.74}$$

Nhân trái vô hướng hai vế của (1.74) với bra $\langle\Phi_n|$ ta thu được biểu thức phần bổ chính bậc hai của năng lượng:

$$E_n^{(2)} = \sum\langle\Phi_n|\hat{H}_1|\Phi_k\rangle c_{nk}^{(1)}$$

Thay $c_{nk}^{(1)} = \dfrac{\langle\Phi_m|\hat{H}_1|\Phi_n\rangle}{E_n^{(0)} - E_m^{(0)}}$ vào ta được:

$$E_n^{(2)} = \sum_{k\neq n}\frac{\left|\langle\Phi_k|\hat{H}_1|\Phi_n\rangle\right|^2}{E_n^{(0)} - E_k^{(0)}} \tag{1.75}$$

Để cho phép gần đúng có nghĩa thì số hạng bổ chính phải nhỏ, ta phải có:

$$\left|c_{nk}^{(1)}\right| = \left|\frac{\langle\Phi_k|\hat{H}_1|\Phi_n\rangle}{E_n^{(0)} - E_k^{(0)}}\right| << 1 \tag{1.76}$$

5.2. Nhiễu loạn suy biến

Xét toán tử Hamiltơn $\hat{H}_0$ không nhiễu loạn bị suy biến bậc s

$$\hat{H}_0|\Phi_{nk}\rangle = E_n^{(0)}|\Phi_{nk}\rangle,\ k = 1, 2,...., s. \tag{1.77}$$

ứng với một giá trị năng lượng $E_n^{(0)}$, trạng thái của hệ không nhiễu loạn được mô tả bởi các vectơ trạng thái trực giao:

$$|\Phi_{n1}\rangle, |\Phi_{n2}\rangle,....., |\Phi_{ns}\rangle$$

Giả sử Hamiltonian nhiễu loạn $\hat{H}$ của hệ có dạng: $\hat{H} = \hat{H}_0 + \lambda\hat{H}_1$

và các vectơ riêng của $\hat{H}$ được xác định từ phương trình Schrodinger.

$$\hat{H}|\Psi_n\rangle = (\hat{H}_0 + \lambda\hat{H}_1)|\Psi_n\rangle = E_n|\Psi_n\rangle \tag{1.78}$$

Khai triển $|\Psi_n\rangle$ theo các vectơ riêng của toán tử $\hat{H}_0$

$$|\Psi_n\rangle = \sum_{k=1}^{s} c_k|\Phi_{nk}\rangle \tag{1.79}$$

và thay (1.79) vào (1.78) ta được:

$$\sum_{k=1}^{s} c_k\left(\hat{H}_0|\Phi_{nk}\rangle + \lambda.\hat{H}_1|\Phi_{nk}\rangle\right) = E_n\sum_{k=1}^{s} c_k|\Phi_{nk}\rangle \tag{1.80}$$

Nhân hai vế của phương trình (1.80) với bra $\langle\Phi_{nk}|$

$$\sum_{k=1}^{s} c_k E_n^{(0)}\langle\Phi_{n'}|\Phi_{nk}\rangle + \sum c_k\langle\Phi_{n'}|\lambda.\hat{H}_1|\Phi_{nk}\rangle = E_n\sum_{k=1}^{s} c_k\langle\Phi_{n'}|\Phi_{nk}\rangle \tag{1.81}$$

Và kí hiệu yếu tố ma trận $H_{jk} = \langle\Phi_{n'}|\lambda.\hat{H}_1|\Phi_{nk}\rangle$, đồng thời đặt $\varepsilon_n = E_n - E_n^{(0)}$ khi đó phương trình (1.81) biến đổi thành:

$$\sum_{k=1}^{s}(H_{jk} - \varepsilon_n.\delta_{jk})c_k = 0 \tag{1.82}$$

Phương trình này gọi là phương trình thế kỉ.

Muốn cho hệ phương trình trên có nghiệm c_k khác không thì định thức của hệ phải bằng không.

ta phải có:
$$\begin{vmatrix} (H_{11}-\varepsilon_n) & H_{12} & H_{13}\ldots & H_{1s} \\ H_{21} & (H_{22}-\varepsilon_n) & H_{23}\ldots & H_{2s} \\ \ldots\ldots\ldots\ldots & & & \\ H_{s1} & H_{s2} & H_{s3}\ldots & (H_{ss}-\varepsilon_n) \end{vmatrix} = 0 \qquad (1.83)$$

Đây là phương trình đại số bậc s đối với ε_n. Giải phương trình này ta được s nghiệm nói chung khác nhau là: $\varepsilon_n = \varepsilon_{n1}, \varepsilon_{n2}, \ldots, \varepsilon_{ns}$

Như vậy nhiễu loạn một mức suy biến $E_n^{(0)}$ sẽ tách thành một dãy các mức gần nhau: $E_n = E_n^{(0)} + \varepsilon_{nj}$, j = 1, 2,..., s.

Thay một giá trị ε_{nj} vào hệ phương trình (1.82) ta tìm được một giá trị của c_k. Thành thử vectơ trạng thái của hệ nhiễu loạn bây giờ có dạng:

$$|\Psi_{nj}\rangle = \sum_{k=1}^{s} c_k(\varepsilon_{nj})|\Phi_{nk}\rangle \qquad (1.84)$$

5.3. Nhiễu loạn phụ thuộc vào thời gian

Trong trường hợp nhiễu loạn tác dụng lên hệ lượng tử phụ thuộc vào thời gian. Hamiltonian phụ thuộc vào thời gian của hệ có dạng:

$$\hat{H}(t) = \hat{H}_0 + \hat{H}_1(t) \qquad (1.85)$$

Năng lượng của hệ nhiễu loạn nói chung không bảo toàn và hệ không có trạng thái dừng. Tuy nhiên để áp dụng phương pháp nhiễu loạn chúng ta sẽ xác định các vectơ trạng thái của hệ nhiễu loạn theo các trạng thái dừng của hệ không nhiễu loạn $|\Phi_n\rangle$.

Trước hết, ta khai triển các vectơ riêng $|\Psi(t)\rangle$ của Hamiltonian toàn phần theo các trạng thái dừng riêng $|\Phi_n\rangle$ của toán tử $\hat{H}_0$

$$|\Psi(t)\rangle = \sum_n c_n(t)|\Phi_n\rangle \qquad (1.86)$$

Khi $\hat{H}_1 = 0$ ta có:

$$c_n(t) = c_n(0).\exp\left(-\frac{iE_nt}{\hbar}\right) \qquad (1.87)$$

trong đó E_n là năng lượng của hệ ở trạng thái dừng $|\Phi_n\rangle$

$$\hat{H}_0|\Phi_n\rangle = E_n|\Phi_n\rangle$$

Do đó để thuận lợi hơn ta viết lại hệ số $c_n(t)$ khi hệ có nhiễu loạn:

$$|\Psi(t)\rangle = \sum_n c_n(t).\exp\left(-\frac{iE_n t}{\hbar}\right)|\Phi_n\rangle \tag{1.88}$$

Thay khai triển vào phương trình Schrodinger: $i\hbar\frac{d}{dt}|\Psi(t)\rangle = (\hat{H}_0 + \hat{H}_1)|\Psi(t)\rangle$ ta được:

$$\sum\left[i\hbar\frac{dc_n(t)}{dt} + E_n c_n(t).\right]e^{-\frac{iE_n t}{\hbar}}|\Phi_n\rangle = \sum\left[E_n + \hat{H}_1(t)\right]c_n(t).e^{-\frac{iE_n t}{\hbar}}|\Phi_n\rangle \tag{1.89}$$

Nhân vô hướng (1.89) với bra $\langle\Phi_m|$ và sử dụng tính chất trực chuẩn của bracket $\langle\Phi_m|\Phi_n\rangle = \delta_{mn}$, ta được:

$$i\hbar\frac{dc_n(t)}{dt} = \sum_n c_n(t).e^{\frac{i(E_m - E_n)t}{\hbar}}\langle\Phi_m|\hat{H}_1|\Phi_n\rangle \tag{1.90}$$

Giả sử khi $t \le 0$ hệ ở trạng thái không nhiễu loạn nào đó $|\Phi_i\rangle$, khi đó:

$$c_n(0) = c_n^{(0)} = \delta_{ni} \tag{1.91}$$

Bắt đầu từ thời điểm $t = 0$ hệ chịu tác dụng của nhiễu loạn nhỏ $\hat{H}_1$ và ở thời điểm $t > 0$ các hệ số $c_n(t)$ có thể khai triển dưới dạng chuỗi nhiễu loạn sau:

$$c_n = c_n^{(0)} + c_n^{(1)} + c_n^{(2)} + ... \tag{1.92}$$

Thay (1.92), (1.93) vào (1.90) ta nhận được phương trình đối với $c_n(t)$ trong gần đúng bậc một ($m \neq i$).

$$i\hbar\frac{dc_m^1(t)}{dt} = \exp\left[\frac{i.(E_m - E_i).t}{\hbar}\right]\langle\Phi_m|\hat{H}_1|\Phi_i\rangle$$

nghiệm của phương trình này có dạng:

$$c_m^{(1)} = -\frac{i}{\hbar}\int_0^t dt' e^{\frac{i.(E_m - E_i)t'}{\hbar}}\langle\Phi_m|\hat{H}_1(t')|\Phi_i\rangle \tag{1.93}$$

do đó trong gần đúng bậc một, ta có:

$$c_m(t) = \delta_{mi} - \frac{i}{\hbar}\int_0^t dt' e^{\frac{i.(E_m - E_i)t'}{\hbar}}\langle\Phi_m|\hat{H}_1(t')|\Phi_i\rangle \tag{1.94}$$

với điều kiện: $\left|c_n^1(t)\right| << 1$.

5.4. Phương pháp biến phân

Trong cơ học lượng tử, ngoài lí thuyết nhiễu loạn, người ta còn sử dụng phương pháp gần đúng khác gọi là phương pháp biến phân. Phương pháp biến phân xuất phát từ nhận xét đơn giản rằng năng lượng trung bình của một hệ luôn lớn hơn hoặc bằng năng lượng trạng thái cơ bản của hệ lượng tử. Việc tính năng lượng trạng thái cơ bản dẫn đến việc chọn các hàm thử chứa một số thông số chưa biết nào đó. Sau đó tìm cực tiểu của năng lượng trung bình cho phép ta xác định được thông số, nghĩa là xác định được năng lượng trạng thái cơ bản của hệ.

Khai triển vectơ trạng thái của hệ lượng tử $|\Psi\rangle$ theo các vectơ riêng $|u_n\rangle$ của toán tử Hamiltonian $\hat{H}$:

$$|\Psi\rangle = \sum_n a_n |u_n\rangle \tag{1.95}$$

Trị trung bình của năng lượng của hệ ở trạng thái đã cho có dạng:

$$\langle \hat{H} \rangle = \frac{\langle \Psi | \hat{H} | \Psi \rangle}{\langle \Psi | \Psi \rangle} = \frac{\sum_n |a_n|^2 . E_n}{\sum_n |a_n|^2} \tag{1.96}$$

Gọi E_0 là năng lượng trạng thái cơ bản của hệ lượng tử, từ (1.96) ta có:

$$\langle \hat{H} \rangle \geq \frac{E_0 \sum_n |a_n|^2}{\sum_n |a_n|^2} = E_0 \tag{1.97}$$

Chọn các vectơ trạng thái $|\Psi\rangle$ là một hàm của thông số chưa biết nào đó $\lambda_1, \lambda_2, \ldots$ sao cho gần trùng với vectơ trạng thái cơ bản của hệ.

$$\Psi = \Psi(\lambda_1, \lambda_2, \ldots) \tag{1.98}$$

và thực hiện cực tiểu hoá năng lượng trung bình:

$$\frac{\partial \langle \hat{H} \rangle}{\partial \lambda_i} = 0 \tag{1.99}$$

cho phép xác định các thông số λ_i. Bằng cách đó ta sẽ tính được giá trị trung bình năng lượng $E = \dfrac{\langle \Psi(\lambda_{01}, \lambda_{02}, \ldots) | \hat{H} | \Psi(\lambda_{01}, \lambda_{02}, \ldots) \rangle}{\langle \Psi(\lambda_{01}, \lambda_{02}, \ldots) | \Psi(\lambda_{01}, \lambda_{02}, \ldots) \rangle}$ gần với giá trị năng lượng cơ bản của hệ.

Phương pháp tính năng lượng ở trạng thái cơ bản của hệ lượng tử nói trên phụ thuộc vào việc chọn hàm thử. Ngoài ra ta cũng có thể tính năng lượng của trạng thái kích thích thứ nhất E_1 hoặc thứ hai E_2,...

Thực vậy, nếu kí hiệu $|\Psi_0\rangle$ là vectơ trạng thái cơ bản của hệ thì việc tính E_1 đòi hỏi bài toán biến phân:

$$E_1 = \min\langle\Psi_1|\hat{H}|\Psi_1\rangle \tag{1.100}$$

với điều kiện bổ sung $\langle\Psi_1|\Psi_1\rangle = 1$ và $\langle\Psi_1|\Psi_0\rangle = 0$.

Tương tự, việc tính năng lượng ở mức kích thích E_2 dẫn đến giải bài toán:

$$E_2 = \min\langle\Psi_2|\hat{H}|\Psi_2\rangle \tag{1.101}$$

với điều kiện bổ sung $\langle\Psi_2|\Psi_2\rangle = 1$ và $\langle\Psi_2|\Psi_1\rangle = \langle\Psi_2|\Psi_0\rangle = 0$

5.5. Phương pháp trường tự hợp Hartee – Fock

Toán tử Hamilton của nguyên tử nhiều electron hệ toạ độ gắn với hạt nhân có dạng:

$$\hat{H} = \sum_i \hat{H}_i + \frac{1}{2}\sum_{i,k} V_{ki} \qquad i, k = 1, 2, ..., Z \tag{1.102}$$

trong đó $\hat{H}$ là toán tử Hamilton của electron thứ i trong trường hạt nhân điện tích Ze, $V_{ki} = \frac{e^2}{r_{ki}}$ là toán tử tương tác của hai electron, dấu phẩy ở tổng thứ hai kí hiệu tổng chỉ lấy các giá trị k và i với $k \neq i$.

Để thuận tiện, ta dùng phương pháp biến phân để tính năng lượng của trạng thái cơ bản của nguyên tử. Khi đó hàm sóng của nguyên tử được xác định từ đẳng thức.

$$\delta J = \delta \int \psi^* H \psi dV = 0 \tag{1.103}$$

Với điều kiện $\int \psi^* \psi dV = 1$

Ta chọn hàm sóng của nguyên tử là tích của hàm sóng của các electron riêng biệt:

$$\psi(\vec{r}_1, \vec{r}_2 ... \vec{r}_2) = \varphi(\vec{r}_1)\varphi(\vec{r}_2)... \varphi(\vec{r}_z) \tag{1.104}$$

Điều đó ứng với giả thiết là các electron trong nguyên tử độc lập với nhau. Ở đây chưa kể đến tính đối xứng đối với việc hoán vị các cặp hạt. Nếu thay (1.104) vào biểu thức của J và chú ý là $\hat{H}_i$ chỉ có tác dụng lên toạ độ của electron thứ i, còn V_{ki} chỉ có tác dụng lên toạ độ của các electron thứ k và thứ i, ta có:

$$J = \sum_i \int \varphi_i^* \hat{H}_i \varphi_i dV_i + \frac{1}{2} \sum_{k \neq i} \int \varphi_i^* \varphi_k^* V_{ki} \varphi_i \varphi_k dV_k dV_i \qquad (1.105)$$

Thực hiện biến phân theo φ_i^* ta có:

$$\delta J = \sum_i \int \delta\varphi_i^* \hat{H}_i \varphi_i dV_i + \sum_{k \neq i} \int \delta\varphi_i^* \delta_k^* V_{ki} \varphi_i \varphi_k dV_k dV_i = 0 \qquad (1.106)$$

trong đó biến phân theo $\delta\psi_i^*$ thoả mãn điều kiện

$$\int \delta\varphi_i^* \varphi_i dV_i = 0 \qquad (1.107)$$

Nhân đẳng thức này với thừa số bất định Lagrange ε_i và cộng vào (1.106), ta có:

$$\delta J = \sum_i \int \delta\varphi_i^* \{H_i + \sum_{k \neq i} \int \varphi_k^* V_{ki} \varphi_k dV_k - \varepsilon_i\} \varphi_i dV_i \qquad (1.108)$$

Biến phân $\delta\varphi_i^*$ là độc lập nên đẳng thức này sẽ được thoả mãn chỉ với điều kiện

$$\left[H_i + \sum_{k \neq i} \varphi_k^* V_{ki} \varphi_k dV_k - \varepsilon_i \right] \varphi_i = 0 \quad i = 1, 2, ..., Z \qquad (1.109)$$

Đây là một hệ phương trình vi phân tuyến tính đối với các hàm chưa biết $\varphi_1, \varphi_2, ..., \varphi_z$. Hệ phương trình này lần đầu tiên được đưa ra bởi Hartee dựa trên khái niệm trường trung bình và sau đó Fock thiết lập bằng phương pháp gần đúng liên tiếp. Đầu tiên, ta chọn hàm sóng trong phép gần đúng tại không là hàm sóng của nguyên tử có một electron. Đó là hàm sóng của nguyên tử Hiđrô và các ion tương tự He^+, Li^{++}... Nhờ hàm φ_k^o, ta tính được tổng.

$$v_i^{(o)}(\vec{r}_i) = \sum_{k \neq i} \int \varphi_k^{*o} V_{ki} \varphi_k^o dV_k$$

Tổng này là giá trị trung bình của tương tác của electron thứ i với tất cả các electron còn lại ở trong trạng thái được mô tả bằng các hàm φ_k^o. Nếu thay giá trị của tổng này vào (1.109), nhận được hệ phương trình để xác định các hàm φ_i^1 trong phép gần đúng bậc nhất:

$$\left(H_i + v_i^o - \varepsilon_i^o\right) \varphi_i^1 = 0.$$

Sau khi giải hệ phương trình này, tính được thế năng mới.

$$v_i^{(1)}(\vec{r}_i) = \sum_{k \neq i} \int \varphi_k^* V_{ki} \varphi_k^1 dV_k$$

Nhờ thế năng này, tìm được hàm $\varphi_k^{(2)}$ gần đúng bậc 2

$$\left[H_i + v_i^1 - \varepsilon_i^1\right]\varphi_i^2 = 0$$

Nếu quá trình này hội tụ, thì có thể tiếp tục cho đến khi nhận được thế năng

$$v_i(\vec{r}_i) = \sum_{k \neq i} \int \varphi_k^* V_{ki} \varphi_k dV_k \qquad (1.110)$$

và hệ phương trình

$$\left[H_i + v_i(\vec{r}) - \varepsilon_i\right]\varphi_i(\vec{r}_i) = 0 \qquad (1.111)$$

Nhờ hệ phương trình này, ta tính được thế năng. Thế năng nhận được như vậy gọi là trường tự hợp Hartee.

Như vậy, với phương pháp trường tự hợp, bài toán nhiều electron rút về bài toán một electron. Mỗi electron trong nguyên tử được xét như chuyển động trong trường tự hợp tạo bởi hạt nhân cùng với tất cả các electron còn lại.

Fock đã mở rộng phương pháp của Hartee bằng cách kể đến tính đồng nhất của electron. Theo phương pháp của Fock, hàm thử được xây dựng nhờ các hàm sóng của electron riêng biệt phụ thuộc cả vào các biến số không gian lẫn các biến số spin. Đặt ξ_i là tập

$$\psi = \frac{1}{\sqrt{z!}} \begin{vmatrix} \psi_1(\xi_1)\psi_1(\xi_2).....\psi_1(\xi_z) \\ \psi_2(\xi_1)\psi_2(\xi_2).....\psi_2(\xi_z) \\ ..... \\ \psi_z(\xi_1)\psi_z(\xi_2).....\psi_z(\xi_z) \end{vmatrix} \qquad (1.112)$$

Fock đã nhận được phương trình đối với hàm sóng $\psi_i(\xi)$. Phương trình này có dạng phức tạp hơn nhiều so với phương trình nhận được bằng phương pháp Hartee nhưng kết quả chính xác hơn nhiều.

Ví dụ trường hợp nguyên tử Hêli: Tính đối xứng của hàm tọa độ phụ thuộc vào spin toàn phần của hệ. Trong trường hợp nguyên tử Hêli, ta có hai trường hợp:

+ Trạng thái para ứng với spin toàn phần của hệ bằng không.

+ Trạng thái octo ứng với spin tổng cộng bằng một.

Giả sử toán tử Hamiltonian có dạng:

$$\hat{H} = \hat{H}_1^0 + \hat{H}_2^0 + V_{12} \qquad (1.113)$$

Trong đó $\hat{H}_i^0$ chỉ tác dụng lên toạ độ của hạt thứ i.

– Đầu tiên ta tìm phương trình xác định có trạng thái para của hệ: Trạng thái spin tổng cộng bằng không và hàm sóng tọa độ là đối xứng. Ta chọn hàm thử:

$$\psi = \frac{1}{\sqrt{2}}\left[\varphi_a(1)\varphi_b(2) + \varphi_a(2)\varphi_b(1)\right] \tag{1.114}$$

Thay vào biểu thức của J ta có:

$$\begin{aligned} J &= \int \psi^* \hat{H} \psi . dV_1 . dV_2 \\ &= \int \varphi_a^* \hat{H}_1^0 . \varphi_a dV_1 + \int \varphi_b^* . \hat{H}_2^0 . \varphi_b dV_2 + \int \varphi_a^*(1)\varphi_b^*(2) V_{12} \varphi_b(2)\varphi_a(1) dV_1 dV_2 \\ &\quad + \int \varphi_a^*(1).\varphi_b^*(2).V_{12}\varphi_b(1).\varphi_a(2) dV_1 dV_2 \end{aligned} \tag{1.115}$$

Tính biến phân δJ theo hàm φ_a^* và φ_b^* với điều kiện:

$$\int \delta\varphi_i^* \varphi_k dV = \delta_{ik} \; ; \; i, k = a, b.$$

Ta nhận được biểu thức: $\delta(J - E_a \int \varphi_a^* \varphi_a dV - E_b \int \varphi_b^* \varphi_b dV) = 0$

Từ đó ta tìm được hệ hai phương trình:

$$\begin{aligned} (\hat{H}^0 + \nu_{bb} - E_a)\varphi_a + \nu_{ba}\varphi_b &= 0 \\ (\hat{H}^0 + \nu_{aa} - E_b)\varphi_b + \nu_{ab}\varphi_a &= 0 \end{aligned} \tag{1.116}$$

trong đó: $\nu_{bb} = \int \varphi_b^*(1) V_{12} \varphi_b(1) dV_1$ là tích phân tính đến tương tác Coulomb của electron ở trong trạng thái φ_b với electron ở trong trạng thái φ_a. Còn tích phân: $\nu_{ba} = \int \varphi_b^*(1) V_{12} \varphi_a(1) dV_1$ là tích phân trao đổi tính đến tương quan trong chuyển động của electron, gây ra do việc đối xứng hoá các hàm toạ độ.

– Trong trạng thái octo: Trạng thái spin tổng cộng bằng 1 và hàm sóng toạ độ là phản đối xứng.

$$\psi = \frac{1}{\sqrt{2}}\left[\varphi_a(1)\varphi_b(2) - \varphi_a(2)\varphi_b(1)\right] \tag{1.117}$$

Tương tự, hệ phương trình Fock có dạng:

$$\begin{aligned} (\hat{H}^0 + \nu_{bb} - E_a)\varphi_a - \nu_{ba}\varphi_b &= 0 \\ (\hat{H}^0 + \nu_{aa} - E_b)\varphi_b - \nu_{ab}\varphi_a &= 0 \end{aligned} \tag{1.118}$$

Hệ phương trình (1.116) khác (1.118) bởi dấu của tích phân trao đổi. Nếu không kể đến tính đối xứng của các hàm sóng thì tích phân trao đổi sẽ biến mất và hệ hai phương trình này trùng nhau, ta lại nhận được phương trình Hartee ít chính xác, trong đó các mức năng lượng của các trạng thái para và octo là như nhau.

§6. PHƯƠNG PHÁP TÍNH SỐ VÀ MÔ PHỎNG

6.1. Phương pháp tính số

Trên cơ sở các lí thuyết tính và các phương pháp tính gần đúng, người ta xây dựng nên phương pháp tính số nhằm giải quyết những bài toán phức tạp (khối lượng tính toán cồng kềnh, không có kết quả giải tích tường minh...). Đặc điểm cơ bản của phương pháp tính số là triển khai các điều kiện đầu bài thành bảng số liệu và kết quả thu được sau khi giải bài toán là một số hoặc một bảng dữ liệu số, từ đó dễ dàng vẽ đồ thị, so sánh đánh giá. Các phương pháp giải gần đúng phương trình – hệ phương trình (phương pháp chia đôi, phương pháp lặp, vẽ đồ thị), giải gần đúng phương trình vi phân – hệ phương trình vi phân, tính gần đúng tích phân (tính theo đa thức nội suy) đều là các phương pháp tính số.

Máy tính tốc độ cao ra đời cho phép ta thực hiện được khối lượng tính toán lớn, các khoảng chia của biến số cũng cho phép thu hẹp hơn giúp cho kết quả bài toán được giải theo phương pháp tính số có độ chính xác rất cao. Hiện nay phương pháp tính số trở thành một trong những phương pháp tính toán chính xác nhất trong các phương pháp tính hiện đại.

Để đáp ứng nhu cầu tính toán, giải quyết các bài toán ngày càng phức tạp trong phương pháp tính số như hàm phức, mô phỏng các hiện tượng vật lí, tính gần đúng... thì công cụ tính toán cần có tốc độ xử lí cao, dung lượng bộ nhớ lớn cùng với các phần mềm tiện dụng và hiện đại. Trong số các phần mềm chuyên dụng để tính số và mô phỏng phải kể đến một số ngôn ngữ lập trình mạnh như: Mathematica, Maple, Fortran, Matlab... Về cơ bản, các phần mềm đều có những tính năng quan trọng trong tính số như tính giải tích, tính gần đúng, mô phỏng... nhưng giữa chúng cũng có những khác biệt và khi giải các bài toán cụ thể chúng ta cũng nên chọn sử dụng kết hợp các phần mềm phù hợp để mang lại hiệu quả cao hơn.

*** Mathematica:** Phần mềm này có ưu điểm là giao diện đẹp, khả năng tính toán rất mạnh và các phép tính có thể sử dụng kí hiệu toán học thay thế cho các lệnh nên rất thuận tiện nhưng lập trình trên các phiên bản khác nhau nhiều khi

không dùng chung file nguồn được (như từ phiên bản mới có thể chuyển về phiên bản cũ được nhưng chuyển ngược lại thì khó khăn). Nhược điểm của phương pháp này là các đồ thị ba chiều (3D) trong Mathematica phải chỉnh các góc nhìn bằng lệnh rất thủ công và rắc rối.

* **Maple:** Có giao diện thân thiện, dễ tiếp cận, ngôn ngữ đơn giản, các hình vẽ 3D có thể điều chỉnh góc nhìn bằng chuột nên rất đẹp và dễ chọn lựa. Tuy nhiên khả năng kết nối cơ sở dữ liệu trong Maple còn nhiều hạn chế như kết nối giữa các phần mềm khá phức tạp và các phép tính hầu hết phải sử dụng lệnh bằng chữ.

* **Fortran:** Có giao diện thân thiện, dễ tiếp cận, ngôn ngữ đơn giản, khả năng kết nối cơ sở dữ liệu giữa các phần mềm mạnh, rất thuận lợi trong tính toán các bài toán phức tạp và phải lặp nhiều vòng. Tốc độ lập trình trên Fortran chậm do các lệnh vào bằng chữ rất thủ công.

* **Matlab:** Có giao diện thân thiện, dễ tiếp cận, ngôn ngữ đơn giản, khả năng kết nối cơ sở dữ liệu giữa các phần mềm mạnh. Hình vẽ trong Matlab cũng rất đẹp nhưng các lệnh vẫn vào bằng chữ.

Các phần mềm Mathematica và Maple đã được chúng tôi giới thiệu trong **giáo trình tin học ứng dụng** (phương pháp tính số dùng trong Vật lí lí thuyết). Trong giáo trình này chúng tôi sẽ giới thiệu tiếp hai ngôn ngữ Matlab và Fortran để mô phỏng các hệ vật lí và giải số các bài toán này.

6.2. Phương pháp Monte-Carlo và phương pháp động lực học phân tử

Phương pháp Monte-Carlo là phương pháp mô hình hóa thống kê trên máy tính điện tử đối với hệ có nhiều bậc tự do. Nội dung chủ yếu của phương pháp là sử dụng các tham số ngẫu nhiên để vẽ các phân bố thống kê. Khi nghiên cứu các hệ phức tạp thì các phân bố thống kê thường không thu được dạng "rõ ràng" nhưng lợi thế của phương pháp Monte-Carlo là có thể tính trực tiếp các tích phân thống kê mà không phải sử dụng thêm các giả thuyết. Người ta thường sử dụng phương pháp Monte-Carlo để mô phỏng và giải số các bài toán phức tạp và giải lặp nhiều vòng như các hệ bán dẫn có kích thước nano, mạng tinh thể phi điều hòa mạnh... Kết quả thu được theo phương pháp này là thông tin về hệ mẫu do chúng ta giả định, thông tin sẽ chính xác hơn nếu khử được các sai số thống kê và số vòng lặp đủ lớn.

Song song với phương pháp Monte-Carlo người ta còn sử dụng phương pháp động lực học phân tử để giải trực tiếp các phương trình chuyển động của hạt trong tinh thể. Các phương này cũng đòi hỏi máy tính có cấu hình mạnh và có phần mềm chuyên dụng phù hợp.

Hiện nay mô phỏng các quá trình vật lí đã trở thành một công cụ nghiên cứu quan trọng và được ứng dụng rộng rãi trong vật lí học và khoa học vật liệu. Các phương pháp mô phỏng để giải quyết các bài toán mạng tinh thể, chế tạo vật liệu mới, nghiên cứu các hệ nano, nghiên cứu các quá trình phản ứng hạt nhân, nghiên cứu trao đổi chất và các đặc tính sinh – lí của các loại tế bào... đã thu được những kết quả quan trọng đối với khoa học đương đại. Các phương pháp mô phỏng này đều dựa trên nền tảng của phương pháp tính số nói chung và các phương pháp tính số cụ thể cho mỗi lĩnh vực khoa học như: phương pháp Monte-Carlo, phương pháp động lực học phân tử...

Trong khoa học vật liệu, việc mô phỏng giúp định hướng chế tạo vật liệu mô hình và xác định các tính chất vật lí của chúng. Người ta có thể mô hình hóa các môi trường liên tục (vật liệu được coi như một mô hình liên tục và giải các phương trình đạo hàm riêng bằng phương pháp hữu hạn hoặc sai phân hữu hạn), mô hình hóa quy mô nguyên tử (vật liệu được xem như tập hợp các nguyên tử riêng biệt có quy luật vận động riêng), kết hợp cả hai loại mô hình hóa nói trên.

6.3. Phương pháp thống kê momen

Phương pháp thống kê momen được xây dựng từ phương pháp thống kê lượng tử. Giả sử có n biến số ngẫu nhiên $q_1, q_2, ..., q_n$ tuân theo quy luật thống kê và được mô tả bởi hàm phân bố $f(q_1, q_2, ..., q_n)$ thỏa mãn điều kiện chuẩn. Khi đó momen cấp m được định nghĩa là:

$$\left\langle q_1^m \right\rangle = \int ... \int q_1^m . f(q_1, q_2, ..., q_n) . dq_1 dq_2 ... dq_n \qquad (1.119)$$

còn momen trung tâm cấp m được định nghĩa là:

$$\left\langle (q_1 - \langle q_1 \rangle)^m \right\rangle = \int ... \int (q_1 - \langle q_1 \rangle)^m . f(q_1, q_2, ..., q_n) . dq_1 dq_2 ... dq_n \qquad (1.120)$$

Như vậy, đại lượng trung bình thống kê $\langle q \rangle$ chính là momen cấp một còn momen trung tâm cấp hai $\langle (q_1 - \langle q_1 \rangle)^2 \rangle$ chính là phương sai. Từ (1.119) và (1.120) ta thấy khi biết hàm phân bố $f(q_1, q_2, ..., q_n)$ thì sẽ xác định được các momen.

Đối với các hệ lượng tử được mô tả bởi các toán tử thống kê $\hat{\rho}$, các momen được định nghĩa như sau:

$$\left\langle \hat{q}^m \right\rangle = Tr(\hat{q}^m \hat{\rho}), \qquad (1.121)$$

$$\left\langle (\hat{q} - \langle \hat{q} \rangle)^m \right\rangle = Tr((\hat{q} - \langle \hat{q} \rangle)^m \hat{\rho}). \qquad (1.122)$$

Toán tử $\hat{\rho}$ tuân theo phương trình Liouville lượng tử:

$$i\hbar\frac{\partial\hat{\rho}}{\partial t}=\left[\hat{H},\hat{\rho}\right] \tag{1.123}$$

trong đó dấu [...] ở vế phải là dấu móc Poisson lượng tử.

Như vậy, nếu biết toán tử thống kê $\hat{\rho}$ thì có thể tính được momen. Mặt khác, giữa các momen có mối quan hệ với nhau, momen cấp cao có thể biểu diễn qua momen cấp thấp hơn. Các hệ thức liên hệ giữa các momen đóng vai trò quan trọng trong việc nghiên cứu các tính chất nhiệt động của tinh thể phi tuyến nên phương pháp momen có thể coi là một trong những phương pháp hữu dụng để nghiên cứu bài toán này.

PHẦN II

NGÔN NGỮ LẬP TRÌNH MATLAB VÀ FORTRAN

Chương 2
GIỚI THIỆU VỀ MATLAB

§1. MỞ ĐẦU

Việc sử dụng ngôn ngữ lập trình Matlab mô phỏng một số hệ vật lí là rất hữu dụng, vì đây là ngôn ngữ có khả năng ứng dụng rất lớn và linh hoạt trong quá trình thiết kế. Để tiếp cận được ngôn ngữ lập trình này trước hết chúng ta cần nắm được khả năng ứng dụng, vai trò của ngôn ngữ lập trình trong ngành khoa học kĩ thuật nói chung và ngành Vật lí học nói riêng.

1.1. Matlab – ngôn ngữ của tính toán kĩ thuật

MATLAB là một ngôn ngữ bậc cao có môi trường tương tác cho phép tiến hành các nhiệm vụ tính toán có cường độ lớn, nhanh hơn so với các ngôn ngữ lập trình khác như C, C++ và Fortran.

MATLAB viết tắt từ "Matrix Laboratory". Ban đầu Matlab được thiết kế bởi Cleve Moler vào những năm 1970, và được sử dụng như một công cụ dạy học. Từ đó đến nay nó đã được phát triển thành một bộ phần mềm thương mại rất thành công.

Hiện nay MATLAB R14 là một bộ phần mềm dùng để tính toán trong các ngành khoa học kĩ thuật và toán học ứng dụng. Matlab là một ngôn ngữ lập trình mạnh, có giao diện đồ họa đẹp.

Matlab là một thương hiệu đã được thương mại hóa của tập đoàn MathWorks, Massachusetts, USA (hiện là nhà cung cấp hàng đầu thế giới cho các phần mềm tính toán kĩ thuật và thiết kế dựa trên mô hình).

1.2. Khả năng và những ứng dụng của Matlab

Một trong những tính năng tuyệt vời nhất của Matlab nhìn từ góc độ những nhà khoa học tính toán là thư viện dựng sẵn rất phong phú, có các chu trình tính toán và các công cụ hiển thị đồ họa.

Matlab cho phép người sử dụng tiến hành rất nhiều các nhiệm vụ thông thường liên quan tới việc giải quyết các vấn đề một cách số học. Nó cho phép người sử dụng dành nhiều thời gian hơn cho việc suy nghĩ, khuyến khích làm thí nghiệm.

Các tính toán mạnh có thể được thực hiện chỉ với một hoặc hai câu lệnh. Dễ dàng xây dựng riêng những hàm toán học cho những ứng dụng đặc biệt.

Matlab cung cấp giao diện đồ họa đẹp, các hình từ Matlab có thể chèn vào LATEX và các tài liệu Word.

Lập trình theo nghĩa thông thường là nhập vào máy những câu lệnh rõ ràng, theo một thứ tự nhất định sao cho khi máy thực hiện theo đúng thứ tự đó thì sẽ cho kết quả mong muốn. Một khái niệm tương tự như vậy thường thấy trong các ngôn ngữ lập trình khác như: ngôn ngữ C, Pascal...

Khi khởi đầu với MatLab ta hãy hiểu theo nghĩa rộng hơn: Lập trình còn có các bước biểu diễn bài toán dưới dạng các hàm và máy tính qua việc thực hiện các hàm này cho ta kết quả. Phương pháp này có mức độ trừu tượng cao hơn so với các câu lệnh chỉ dẫn đơn thuần.

Tuy vậy, các ngôn ngữ lập trình biên dịch như C hay Fortran cho phép tính toán nhanh và tốc độ cũng là yêu cầu quan trọng trong các chương trình tính lớn. Do đó cách kết hợp thông minh là điều cần thiết phần lõi tính toán được viết bằng ngôn ngữ biên dịch, các thao tác nhập xuất, xử lí, hiển thị số liệu được viết bởi ngôn ngữ MatLab.

§2. CÀI ĐẶT VÀ KHỞI ĐỘNG MATLAB 7.0

2.1. Cài đặt Matlab 7.0

Yêu cầu về cấu hình máy tính:

+ Bộ vi xử lí Pentium hoặc Pentium Pro.

+ Windows 95 hoặc NT (WinXP home, XPprofessional).

+ Bộ điều phối đồ họa 8 bit và card màn hình tối thiểu 256 màu.

+ Dung lượng ổ cứng 25Mb cho tới hơn 1Gb (tùy thuộc vào cấu hình đĩa cứng, phân vùng đĩa, số hợp phần của Matlab được cài đặt), và tới 2.1Gb nếu cài đặt Matlab cùng với Simulink.

+ Bộ nhớ động (RAM) tối thiểu 16Mb (nên có bộ nhớ tối thiểu 128Mb).

+ Các khuyến nghị khác: Bộ nhớ bổ sung, card đồ họa bổ sung, card âm thanh, máy in, MS-Word 7.0 hoặc hơn, trình biên dịch C, Borlean, Microsoft (xây dựng file MEX), trình duyệt internet (để chạy Matlab Helpdesk online).

Quá trình cài đặt Matlab 7.0 cho WindowsXP (bộ gồm 2 đĩa CD):

+ Đưa đĩa CD vào ổ đọc. Nếu chương trình SETUP không tự động chạy thì nhấn đúp vào biểu tượng setup.exe để bắt đầu quá trình cài đặt.

+ Accept (chấp nhận) những thỏa thuận về bản quyền. Sau đó click Next.

+ Nếu cài theo kiểu mặc định (hay còn gọi là Typical setup - kiểu phổ biến), Matlab trên máy tính sẽ có các hợp phần cơ bản nhất để làm việc theo các hướng dẫn trong tài liệu này. Theo các hướng dẫn trên màn hình. Cho đĩa CD 2 vào khi được yêu cầu.

+ Nếu cài đặt theo kiểu tùy chọn cá nhân (Manual setup) thì nhấn vào các hộp thành phần dấu 'v' nếu muốn có tùy chọn đó. Nhấn tiếp nếu không có ý định (có thể thêm vào sau này nếu muốn).

+ Trên màn hình hiển thị 'C:\MATLAB7' là thư mục mặc định của quá trình cài đặt. Nếu muốn cài đặt vào địa chỉ khác, hoặc đổi tên thư mục, thì ta lựa chọn 'Browse'.

+ Chi tiết hướng dẫn cài đặt xin xem file 'install–guide.pdf' trong đĩa CD1.

2.2. Khởi động Matlab (Hệ điều hành Windows)

Từ hệ điều hành Windows, khởi động Matlab đơn giản bằng cách nháy đúp vào biểu tượng MATLAB trên màn hình, hoặc bằng cách chọn MATLAB từ Menu Start. Quá trình khởi động đưa người dùng đến cửa sổ lệnh, nơi các dòng lệnh được biểu thị bằng dấu ">>–". Đây là dấu hiệu cho thấy Matlab đang chờ một (câu) lệnh. Khi hoạt động trong chế độ máy tính, tất cả các lệnh của Matlab được nhập vào dòng lệnh từ bàn phím. Matlab có thể được sử dụng theo nhiều chế độ và nhiều cách khác nhau:

+ Như một máy tính cá nhân cao cấp trong chế độ máy tính cá nhân.

+ Như một ngôn ngữ lập trình bậc cao.

+ Như một chu trình con gọi từ chương trình C.

§3. QUẢN LÍ KHÔNG GIAN LÀM VIỆC CỦA MATLAB

Về cơ bản, không gian làm việc của Matlab gồm các phần sau:

+ Cửa sổ trợ giúp (Help window)

+ Nút Start

+ Cửa sổ nhập lệnh (Command window)

+ Cửa sổ không gian làm việc (Workspace window)

+ Cửa sổ quá trình lệnh (Command History window - lịch sử)

+ Cửa sổ biên tập mảng, vectơ, ma trận (Array editor window)

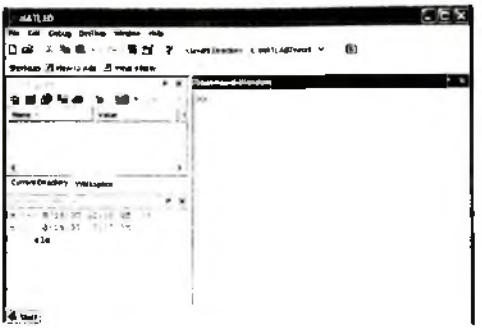
+ Cửa sổ địa chỉ thư mục hiện thời (Current directory window)

– Nút 'x' ở góc trên bên phải mỗi cửa sổ dùng để đóng chúng. Hiển thị lại cửa sổ bằng cách tích vào tên cửa sổ tương ứng trong menu Desktop.

– Nút mũi tên cong bên cạnh nút 'x' dùng để tách các cửa sổ làm việc trong cửa sổ chính Matlab thành cửa sổ con độc lập. Ấn nút này một lần nữa sẽ nhập một cửa sổ độc lập về cửa sổ chính của MATLAB.

– Nút Start: Ở góc dưới bên trái của màn hình Matlab, cho phép chạy các ứng dụng mẫu (demos), các công cụ và cửa sổ chưa hiển thị khi khởi động Matlab. Bằng cách gõ lệnh 'demo' có thể xuất hiện một tập hợp những file trình diễn giá trị cao.

Ví dụ: Thử chạy Start → Matlab → Demos để chạy một ứng dụng mẫu trong cửa sổ Demo(s).



Hình 2.1. Giao diện của Matlab 7.0

Ghi chú:

+ Các diễn giải và câu (mệnh đề) của Matlab được đánh giá khi gõ vào 'cửa sổ lệnh', và các kết quả tính toán cũng được thể hiện tại đây. Không giống như Fortran và các ngôn ngữ tính toán cần biên dịch khác, Matlab là một môi trường tương tác – đưa ra một câu lệnh, thì Matlab cố gắng thực thi câu lệnh đó ngay lập tức trước khi đòi hỏi một lệnh tiếp theo.

+ Cấu trúc có dạng:

>> biến = diễn giải hoặc đơn giản là >> diễn giải

– Các diễn giải thường được soạn bằng các toán tử, các hàm, tên các biến và được hiển thị trên màn hình sau khi ấn Enter.

Các câu lệnh có dạng 'tên biến = diễn giải' thì diễn giải đó sẽ được gán cho biến để sử dụng sau này. Khi 'tên biến' và dấu '=' được bỏ đi thì kết quả của diễn giải sẽ được tự động gán cho biến có tên 'ans' (câu trả lời) và hiển thị trên màn hình.

Một câu (lệnh) thông thường sẽ kết thúc ở cuối dòng. Tuy nhiên có thể tiếp tục một câu bằng ba dấu chấm '...' ở cuối dòng. Có thể đặt một vài câu lệnh trên cùng một hàng, ngăn cách bởi dấu phẩy ',' hoặc chấm phẩy ';'.

Nếu một câu lệnh kết thúc bằng dấu chấm phẩy ở cuối câu thì kết quả của lệnh đó sẽ không được hiển thị, tuy nhiên yêu cầu tính vẫn được thực hiện (phép tính hay phép gán vẫn được thực hiện, kết quả có trong workspace). Điều này là thiết yếu trong việc ẩn đi các kết quả trung gian không mong muốn.

Có thể xóa trắng toàn bộ cửa sổ lệnh bằng lệnh >> clc (clear command window) hoặc vào menu Edit → Clear Command Window. Khi thực hiện lệnh này, toàn bộ giá trị của các biến hiện có không thay đổi hay mất đi.

Cửa sổ không gian làm việc (workspace):

Các biến và dữ liệu khi nhập vào hoặc sau khi tính toán sẽ được Matlab lưu trong phần "không gian làm việc". Tất cả các biến, ngoại trừ những biến cục bộ thuộc về một M-file, sẽ được thể hiện trong không gian làm việc.

– Lệnh 'who' hoặc 'whos' liệt kê các biến hiện có trong không gian làm việc.

Để biết giá trị hiện tại của một biến, ta đánh vào tên biến tại dấu nhắc của cửa sổ lệnh và Enter.

Để xóa một hàm hoặc biến khỏi không gian làm việc, ta sử dụng lệnh 'clear':

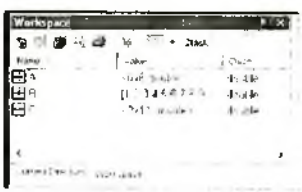
>> clear tên–biến

Bản thân lệnh 'clear' sẽ xóa tất cả các biến hiện có (tương đương với 'clear all')

– Cửa sổ biên tập mảng (ma trận nói chung): Khi đã có một mảng, ta có thể chỉnh sửa, biên tập lại bằng Array Editor. Công cụ này làm việc như một bảng tính (spreadsheet) cho ma trận. Cũng có thể biên tập lại ma trận M bằng cách gõ lệnh:

>> openvar ('C')

– Cửa sổ địa chỉ thư mục hiện thời: Thư mục hiện thời là nơi chương trình Matlab sẽ tìm các M-file và các file không gian làm việc (.mat files) mà ta đã Load và Save.



Hình 2.2. Cửa sổ không gian làm việc

§4. MỘT SỐ LƯU Ý KHI LÀM VIỆC VỚI MATLAB

4.1. Lưu và phục hồi dữ liệu

– Để nhớ các biến, Matlab có thể ghi và gọi lại dữ liệu từ file trong máy tính. Mục Save Workplace as... trong bảng chọn File sẽ mở hộp hội thoại để ghi tất cả các biến hiện tại.

– Tương tự, mục Load Workplace trong bảng chọn File sẽ mở hộp hội thoại để gọi lại tất cả các biến đã ghi lại từ không gian làm việc trước.

Ghi chú: Việc Load không làm mất các biến hiện có trong không gian làm việc hiện tại. Khi gọi lại các biến trùng tên với các biến trong không gian làm việc của Matlab, sẽ làm thay đổi giá trị của các biến theo giá trị của các biến gọi ra từ file.

Ngoài các bảng chọn, Matlab còn cung cấp hai lệnh Save, Load nó thực hiện một cách mềm dẻo hơn. Lệnh save cho phép ghi một hoặc nhiều hơn một biến tùy theo sự lựa chọn.

Ví dụ:

>> save – lưu tất cả các biến trong Matlab theo kiểu nhị phân trong file matlab.mat

>> save dulieu – lưu tất cả các biến trong Matlab theo kiểu nhị phân trong file dulieu.mat

>> save dulieu A B C D – ASCII lưu các biến A, B, C, D theo dạng mã ASCII trong file dulieu.mat.

– Lưu lại một phiên làm việc (session)

Khi làm việc với Matlab, cần lưu tất cả các thông số đầu vào và đầu ra của phiên làm việc để thuận lợi cho việc in ấn sau này. Lệnh 'diary' sẽ lưu tất cả những thông số đầu vào và đầu ra ở giữa hai lệnh 'diary' và 'diary off'.

Ví dụ:

>> diary('diary–file–name')

>> ..... (các câu lệnh của bạn ở đây)

>> diary off

4.2. Sử dụng Help (Trợ giúp)

Trợ giúp thông tin về các lệnh của Matlab có thể được tìm thấy theo nhiều cách:

+ Từ dòng lệnh bằng cách gõ lệnh 'help chủ đề'.

+ Từ cửa sổ màn hình vào Menu Help.

+ Từ helpdesk của Matlab lưu trữ trên đĩa hoặc CD-rom.

+ Từ dòng lệnh gõ lệnh 'help' và Enter.

Kết quả: Matlab cho một bản tóm tắt về hệ thống trợ giúp

HELP topics: (Các chủ đề trợ giúp)

matlab/general - Các lệnh với mục đích tổng quát.

matlab/ops - (operators) Các toán tử và các kí tự đặc biệt...

matlab/lang - (language) Ngôn ngữ lập trình...

matlab/elmat - (elementary) Ma trận căn bản...

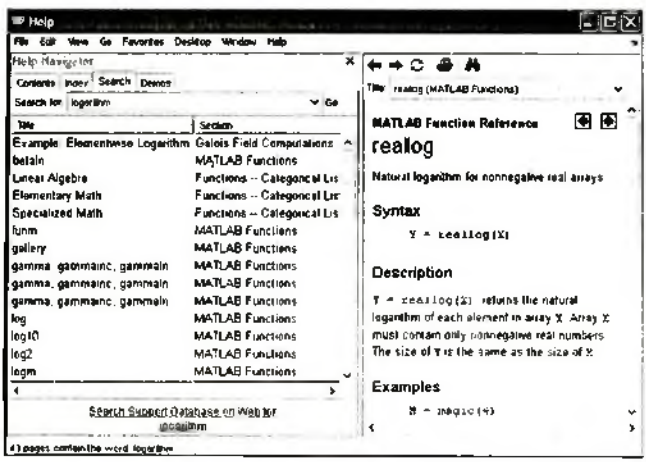
matlab/elfun - (elementary functions) Các hàm toán căn bản.

matlab/specfun - (specialized functions) Các hàm toán đặc biệt.

– Thông thường cửa sổ text không đủ lớn để chứa tất cả thông tin từ một lệnh Matlab. Vì vậy có thể sử dụng chức năng 'more on' để xem thông tin theo từng trang màn hình, sau đó duyệt từng trang một bằng cách nhấn phím bất kì.

– Đánh 'more off' vào cửa sổ lệnh sẽ đưa Matlab trở về cửa sổ thông thường, không duyệt từng trang.

Khi không nhớ chính xác tên của một lệnh Matlab, có thể sử dụng lệnh 'lookfor' (tìm kiếm) như một sự trợ giúp.



Hình 2.3. Giao diện cửa sổ Help của Matlab 7.0

4.3. History

Trong giao diện của Matlab, cửa sổ 'Command History' nằm ở góc phần tư bên dưới, phía trái.

Trong cửa sổ này, các lệnh đã sử dụng trong các lần khởi động Matlab gần đây đều được lưu lại. Mỗi lần khởi động Matlab, toàn bộ các lệnh sử dụng trong lần đó sẽ được lưu lại dưới dạng một nhóm có thể đóng mở bằng nút biểu tượng '+' (expand) hoặc '–' (collapse) ở đầu dòng (dòng ghi mốc thời gian giữa hai dấu chú thích '%'):

Ví dụ: (+) %-- 8/14/07 3:21 PM --%

Để gọi lại lệnh từ cửa sổ 'Command History', bạn tìm đến lệnh đó bằng các thanh cuốn, rồi nháy đúp vào tên lệnh.

Để gọi lại các lệnh bạn đã sử dụng từ dấu nhắc của cửa sổ lệnh, Matlab dùng các phím mũi tên (←↑→↓) trên bàn phím.

Chương 3
NGÔN NGỮ LẬP TRÌNH TÍNH TOÁN TRÊN MATLAB

§1. CÁC PHÉP TOÁN CƠ BẢN

Trước hết chúng ta cần nắm bắt được cách sử dụng các phép toán với các biến, các phép toán có thể từ những phép toán số học thông thường (Phép cộng, trừ...) đến các phép toán phức tạp như: Giải phương trình vi phân... còn các biến có thể bằng số hoặc chữ.

1.1. Các phép toán số học thông dụng: Cộng (+), trừ (−), nhân (*), chia (/)

Ví dụ 1: Để tính A=2+3/4*5

B=2^5 – 3*A

A+B

Bạn gõ các lệnh vào sau dấu nhắc lệnh: >>−

```
>> A = 2 + 3/4*5
A =
5.7500
>> B = 2^5 – 3*A
B =
14.7500
>> A + B
ans =
20.5000
```

Chú ý: – Khi không có toán tử gán, Matlab trả kết quả của phép tính gần nhất vào biến 'ans= ...'

– Thứ tự ưu tiên tính toán: Các đại lượng trong ngoặc đơn, lũy thừa, (* /) làm việc từ trái qua phải, (+ −) làm việc từ trái qua phải.

Lệnh gán:

– Gán trực tiếp các giá trị cho các biến

– Ngoài ra: Có thể dùng biến trả để gán

Ví dụ 2:

```
>> 2^3-3
ans =
```

5

>> ans*6

ans =

30

Quy tắc đặt tên biến:

– Tên biến hợp lệ được cấu tạo bởi các chữ và số, bắt đầu bằng chữ.

– Chiều dài tên biến: Độ dài tùy ý

Chú ý: Tên biến không phụ thuộc vào chữ hoa - chữ thường

Tránh đặt tên biến trùng với tên các hàm chuẩn, hoặc các từ khóa của Matlab như: 'break' 'case' 'catch' 'continue' 'else' 'elseif' 'end' 'for' 'function' 'global' 'if' 'otherwise' 'persistent' 'return' 'switch' 'try' 'while'...

1.2. Các hàm số cơ bản

Lũy thừa: ^, giai thừa: x!

Logarit: ln(x), log[a](b), exp(x)

Các hàm lượng giác: sin(x), cos(x), tan(x), cot(x),...

Một số hàm khác: abs(x) - |x|, sqrt(x) - căn bậc 2 của x

1.3. Các phép toán quan hệ

1.3.1. Các toán tử so sánh

A < B % A nhỏ hơn B

A <= B % A nhỏ hơn hoặc bằng B

A > B % A lớn hơn B

A >= B % A lớn hơn hoặc bằng B

A == B % A bằng B

A ~= B % A không bằng B

– Các toán tử quan hệ thực hiện sự so sánh từng phần tử với phần tử giữa hai mảng. Nó cho kết quả là một mảng logic có cùng kích cỡ với các phần tử của mảng là đúng (1) nếu quan hệ đó là đúng, và phần tử của mảng là sai (0) nếu không đúng.

1.3.2. Các phép toán logic

– Matlab biểu diễn đúng và sai bởi các số nguyên tố 1 và 0: đúng = 1, sai = 0 (true=1, false=0).

Ví dụ 1: Trong quá trình tính toán, biến x (x là một đại lượng vô hướng) nhận một giá trị bất kì, chúng ta có thể tiến hành các phép kiểm tra logic cho nó:

x == 2 xem x có bằng 2 không?

x ~= 2 xem x có khác 2 không?

x > 2 xem x có lớn hơn 2 không?

x < 2 xem x có nhỏ hơn 2 không?

x >= 2 xem x có lớn hơn hoặc bằng 2 không?

x <= 2 xem x có nhỏ hơn hoặc bằng 2 không?

1.3.3. Các toán tử logic chính trong Matlab và ý nghĩa của chúng

&	và (and)
|	hoặc (or)
~	không (not)

1.3.4. Các kiểu dữ liệu - Định dạng kết quả

1.3.4.1. Các kiểu dữ liệu

Matlab sử dụng 15 kiểu (loại) dữ liệu chính. Mỗi một kiểu dữ liệu đều ở dạng của một ma trận hoặc mảng. Các mảng hoặc ma trận có kích cỡ tối thiểu là 0 – nhân – 0 và có thể phát triển tới mảng n - chiều với kích cỡ tùy ý.

a. Các kiểu dữ liệu số và số phức

– Integer: ví dụ như -5 hay 9888.

– Double precision reals: Trong Matlab, tất cả các số thực được lưu với độ chính xác double, không giống các ngôn ngữ lập trình khác như C hay Fortran khi chỉ có một loại riêng biệt float hay real cho các số thực với độ chính xác single.

– Số phức: Được nhập vào dưới dạng 3+2*i hoặc 3+2*sqrt(–1).

– Chuỗi: Là một mảng tập hợp của các kí tự, được nhập vào dưới dạng 'abc' hoặc 'vi du day la mot chuoi'.

b. Các kí tự, chuỗi và văn bản

Khả năng xử lí văn bản trong tính toán rất hữu ích cho việc nhập (xuất) kết quả từ (tới) màn hình vào file lưu trên đĩa. Để có thể quản lí văn bản, một loại dữ liệu 'character' được đưa vào Matlab. Một mảng đơn giản của văn bản là một chuỗi (vectơ) hay một mảng các kí tự.

Các chuỗi có thể được cộng với nhau bằng cách sử dụng các toán tử thao tác trong mảng.

Chú ý: Số các kí tự ở hai dòng phải bằng nhau, nếu không việc thực thi câu lệnh sẽ dẫn tới lỗi:

Dấu ba chấm '...' thể hiện câu lệnh còn tiếp tục ở dòng sau.

c. Chuyển đổi giữa chuỗi và số

Đôi khi chúng ta cần chuyển một chuỗi thành một số tương ứng hoặc ngược lại.

Các công việc chuyển đổi này được thực hiện bởi lệnh:

– 'str2num': Chuyển một chuỗi thành số tương ứng

– 'num2str': Chuyển một số thực thành chuỗi tương ứng

– 'int2str': Chuyển một chuỗi thành số tương ứng

d. Các hằng số

Matlab định nghĩa sẵn nhiều hàm số hữu ích, bao gồm:

pi = 3.141592654...

i và j đều là phần ảo của số phức, = sqrt(-1)

inf, 'infinity' là 'vô cùng'

NaN, 'not-a-number' là 'không phải là một số'

ans luôn được gán cho kết quả của lệnh tính trước đó

e. Các hàm dựng sẵn

Cũng như ngôn ngữ bậc cao khác, Matlab thực thi các 'function' (hàm) nhiều hơn 'procedure' (chương trình con). Các hàm này bao gồm căn bậc hai (sqrt), lũy thừa (exp), logarit (log, log10, log2), giá trị tuyệt đối (abs) và các hàm lượng giác (sin, cos, tan, cotan,..)

Ví dụ 1:

```
>> sin(45)
ans =
0.8509
```

Chú ý rằng tất cả các tính toán của Matlab đều có lỗi làm tròn.

Danh mục các hàm dựng sẵn phổ biến:

+ Các hàm lượng giác:

+ sin - hàm sin,

sind - sin của góc tính theo độ,

sinh - sin hypecbolic,

asin - arcsin hay hàm nghịch đảo của hàm sin,

asind - hàm nghịch đảo của hàm sin, kết quả theo độ,

asinh - hàm nghịch đảo của hàm sin hyperbolic.

+ cos - hàm cos,
cosd - cos của góc tính theo độ,
cosh - cos hyperbolic,
acos - hàm nghịch đảo của hàm cos,
acosd - hàm nghịch đảo của hàm cos, kết quả theo độ,
acosh - hàm nghịch đảo của hàm cos hyperbolic,
+ tan - hàm tang,
tand - tang của góc tính theo độ,
tanh - tang hyperbolic
atan - hàm nghịch đảo của hàm tang,
atand - hàm nghịch đảo của hàm tang, kết quả theo độ,
atan2 - hàm nghịch đảo của hàm tang 4 góc phần tư,
atanh - hàm nghịch đảo của hàm tang hyperbolic,
+ cot - hàm côtang,
cotd - côtang của góc tính theo độ,
coth - côtang hyperbolic,
acot - hàm nghịch đảo của hàm côtang,
acotd - hàm nghịch đảo của hàm côtang, kết quả theo độ,
acoth - hàm nghịch đảo của hàm côtang hyperbolic.
+ Các hàm sơ cấp:
exp - hàm mũ,
expm1 - tính chính xác exp(x)−1,
log - logarit cơ số tự nhiên,
log1p - tính chính xác log(1+x),
log10 - logarit cơ số 10,
reallog - loga cơ số tự nhiên của số thực,
realsqrt - căn bậc hai của một số >= 0,
sqrt - căn bậc hai,
nthroot - nghiệm thực bậc n của các số thực.
+ Các hàm liên quan đến số phức:
abs - giá trị tuyệt đối,
angle - góc pha,

+ complex - xây dựng dữ liệu về số phức từ các phần thực và ảo,

conj - liên hợp của phức,

imag - phần ảo của phức,

real - phần thực của phức,

isreal - hàm logic, trả về giá trị 'true' với mảng số thực,

cplxpair - sắp xếp các số về các cặp liên hợp phức.

+ Các hàm làm tròn và phần dư:

fix - làm tròn về phía 0,

floor - làm tròn về phía âm vô cùng,

ceil - làm tròn về phía dương vô cùng,

round - làm tròn về phía số nguyên gần nhất,

mod - mô đun (lấy phần dư của phép chia),

rem - lấy phần dư của phép chia (tương tự mod),

sign - hàm lấy dấu của một biến, trả về +1, 0, −1 (+, 0, −).

f. Kết hợp nhiều lệnh trên một dòng

Dấu phẩy (,) và dấu chấm phẩy (;) là những kí tự có ý nghĩa đặc biệt trong Matlab như:

+ Toán tử phẩy (,) được dùng để nhóm nhiều lệnh trên một dòng.

Ví dụ 1:

>> x=3.5, y=-5.0, x^ 3 − y

Khi không muốn theo dõi kết quả các tính toán trung gian hay ẩn đi một câu lệnh, ta dùng dấu chấm phẩy (;).

Ví dụ 2:

>> x=3.5; y=-5.0; x^ 3 − y

ans =

47.8750

Trong ví dụ trên kết quả của hai lệnh gán đầu tiên đã được ẩn đi.

1.3.4.2. Định dạng kết quả

– Sử dụng lệnh 'format' cùng các định dạng. Lệnh này chỉ làm thay đổi kết quả được hiển thị trên màn hình, không làm thay đổi độ chính xác của số hoặc phép tính.

– Để thực hiện lệnh, từ dấu nhắc cửa sổ lệnh gõ một trong các lệnh sau:

format short : dấu phẩy thập phân cố định, 5 chữ số.

format long : dấu phẩy cố định, 15 chữ số,

format short e : kí hiệu khoa học, 5 chữ số,

format long e : kí hiệu khoa học, 15 chữ số,

format short g : dấu phẩy cố định hoặc di động, 5 chữ số,

format long g : dấu phẩy cố định hoặc di động, 15 chữ số,

format hex : format dạng Hexa (hệ 16),

format '+' : dương (+), âm (−) và kí tự trắng (blank) ứng với 0,

format bank : Dollars và cents,

format rat : tỉ lệ xấp xỉ integer.

format short là dạng mặc định, khi được gọi lên, một dạng format sẽ có hiệu lực tới khi nó được thay đổi.

1.4. Giới hạn

Các lệnh giới hạn gồm có:

– limit(f): giới hạn của hàm f,

– limit(f,x,a) hoặc limit(f,a): giới hạn của hàm f khi x dần tới a,

– limit(f,x,a,'left'): giới hạn trái của hàm f khi x dần tới a,

– limit(f,x,a,'right'): giới hạn phải của hàm f khi x dần tới a.

1.5. Giải phương trình

Sử dụng lệnh solve(f)

Ví dụ 1:

+ >>solve ('f(x) = 0') để giải phương trình f(x) = 0.

```
>>solve ('2x + 1')
>>ans = −1/2
```

+ solve(' f(x) = g(x) ') để giải phương trình f(x) = g(x)

```
>>s = solve(`cos(2*x) + sin(x) = 1`)
s =
[      0]
[     pi]
[ 1/6*pi]
[ 5/6*pi]
```

+ solve('f(x)','g(x)','h(x)',...): giải hệ nhiều phương trình.

1.6. Tính tích phân

+ Để tính tích phân ta dùng hàm quad (tính tích phân theo phương pháp Simpson) và hàm quadl (tính tích phân theo phương pháp Lobatto).

Ví dụ 1:

```
f = inline('1./((x-0.3).^2+0.01)+1./((x-0.9).^2+0.04)-6');
- q = quad(f,0,1)
q =
29.8583
- r = quadl(f,0,1)
r =
29.8583
```

Ví dụ 2:

```
y = sin(x)
- quad('sin',0,pi)
ans =
2.00001659104794
- quadl('sin',0,pi)
ans =
1.99999999999989
```

+ Ngoài ra, ta có thể dùng hàm trapz (phương pháp hình thang) để tính tích phân:

```
y = sin(x)
x = [0:pi/100:pi];
y = sin(x);
trapz(x,y)
ans =
1.99983550388744
```

1.7. Số phức

Số phức trong Matlab được sử dụng theo nhiều cách

```
+ >>c1=1-2i  % Chèn thêm kí tự i vào phần ảo
c1=
1.0000-2.0000i
+ >>c2=6+sin(.5)*i
```

c2=

6.0000-0.4795i

+ Đối với Matlab tất cả các phép toán học đều sử dụng như đối với số thực

```
>>c3=c1+c2
c3=
7.0000-2.4795i
```

+ Ngoài ra, chúng ta có thể sử dụng hàm real và imag để kiểm tra phần thực và ảo của số phức

```
>>real-c1=real(c1)  % Tính phần thực
real-c1=1
>>imag-c1=imag(c1)  % Tính phần ảo
imag -c1=-2
```

1.8. Các cấu trúc lặp

1.8.1. Câu lệnh rẽ nhánh (if và switch)

Câu lệnh: if...else if...else...end

Ví dụ:

```
if isinf(x) ! ¯ isreal(x)
disp('So lieu dau vao xau!')
y = NaN;
elseif (x == round(x)) && (x > 0)
y = prod(1:x-1);
else
y = gamma(x);
end
```

1.8.2. Câu lệnh switch...case...case... case...otherwise...end

Bộ câu lệnh if / elseif chỉ hữu ích trong trường hợp chỉ có một vài lựa chọn. Còn khi có một số lượng lớn các lựa chọn khả dĩ người ta dùng switch để thay thế.

Ví dụ:

```
switch donvi
case 'Chieudai'
disp('met')
case 'The tich'
```

```
disp('lit')
case 'Thoi gian'
disp('giay')
otherwise
disp('Toi chiu thua')
end
```

Trường hợp phù hợp với case thì các lệnh sẽ được thực thi.

Chú ý: Có thể sử dụng otherwise hoặc không. Trong trường hợp có sử dụng thì Matlab thực thi các lệnh sau otherwise, hoặc không nếu không có trường hợp nào phù hợp với case.

1.8.3. Vòng lặp (for và while)

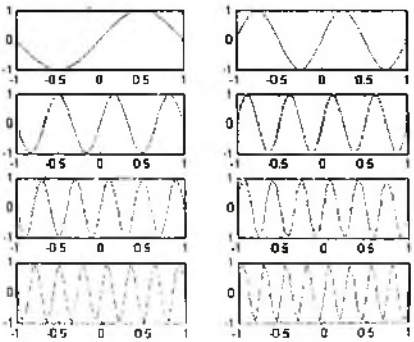
1.8.3.1. Vòng lặp for...end

Vòng lặp for...end được sử dụng khi lặp một đoạn mã lệnh với một số lần tùy ý.

Ví dụ 1: Vẽ đồ thị hàm số sin(n.x); với $-1 < x < 1$ và n = 1, 2, ...8.

Để thực hiện, ta phải sử dụng 8 lệnh vẽ riêng rẽ. Nhưng sẽ dễ dàng hơn nhiều nếu ta sử dụng một vòng lặp *for...end*.

```
>> x = -1:0.5:1; % bước nhảy 0.5
>> for n = 1:8
subplot(4,2,n), plot(x,sin(n*pi*x))
end
```



Chú ý: + Tất cả các lệnh giữa hai dòng bắt đầu bằng 'for' và kết thúc bằng 'end' đều được lặp đi lặp lại với n = 1 lần thứ nhất, n = 2 lần thứ 2... cho tới khi n = 8.

+ Lệnh subplot tạo ra một mảng 4×2, cửa sổ đồ thị con trong một đồ thị chính. Ở lần lặp thứ n, một hình sẽ được vẽ lên cửa sổ đồ thị con thứ n.

1.8.3.2. Vòng lặp while...end

Để thực hiện việc lặp đi lặp lại một đoạn mã lệnh của Matlab cho tới khi một điều kiện (logic) nào đó được thỏa mãn mà không biết số lần lặp cụ thể, khi đó ta có thể sử dụng vòng lặp *while...end*

Ví dụ 1: Tìm giá trị lớn nhất của n sao cho tổng dưới đây nhỏ hơn 100?

```
>> S = 1; n = 1;
>> while S+ (n+1)^2 < 100
n = n+1; S = S + n^2;
end
>> [n, S]
ans =
6 91
```

1.9. Vectơ và Ma trận

1.9.1. Vectơ

1.9.1.1. Giới thiệu

Đại số tuyến tính là trái tim của Matlab. Vì vậy hơn bất kì ngôn ngữ nào khác, Matlab khuyến khích ta tận dụng mọi khả năng của mảng, vectơ và ma trận.

* Một vài thuật ngữ:

+ Mảng là một tập hợp các số, được gọi là các 'phần tử' hay các 'đầu số', được biết đến với một hoặc nhiều chỉ số chạy suốt các tập hợp chỉ số.

+ Số chiều của một mảng là số các chỉ số cần thiết để định nghĩa một phần tử trong mảng. Chẳng hạn mảng 2 chiều sẽ cần 2 chỉ số i và j để đặc trưng cho một phần tử của mảng.

+ Kích thước của mảng là một danh sách các kích thước của các tập hợp chỉ số.

Ví dụ 1:

```
>> r = [1 2 3; −1 −2 −7]
```

>> size(r) nghĩa là kích thước của mảng r sẽ là 2×3 (2 hàng, 3 cột).

+ Ma trận là một mảng hai chiều (kích thước m × n với các quy luật đặc biệt cho phép cộng, nhân và các tính toán khác).

+ Vectơ là một ma trận một chiều có chỉ số bằng 1.

+ Các phần tử đơn lẻ trong ma trận có thể được tiếp cận hoặc sửa đổi bằng cách sử dụng chỉ số phần tử (subscripting). Trong Matlab, phần tử thứ i của vectơ V được biểu diễn bằng kí hiệu V(i), chỉ số được viết trong ngoặc đơn.

Ví dụ 2:

```
>> V = [10 20 30]
V =
10 20 30
>> V(2)
ans =
20
```

1.9.1.2. Vectơ hàng

+ Vectơ hàng là chuỗi các số được phân cách bởi dấu phẩy hoặc khoảng trống. Số lượng các đầu số được gọi là 'chiều dài' của vectơ, mỗi đầu số thường được nhắc đến như 'phần tử', hoặc 'hợp phần' của vectơ.

+ Cú pháp: Để nhập 1 vectơ là một chuỗi các giá trị được bao trong cặp ngoặc vuông [ ].

Ví dụ 1:

```
>> v = [ 1 3 sqrt(5)]
```

+ ngoài ra, ta có thể sử dụng các dấu phẩy (,)

```
>> v = [1, 3, sqrt(5)]
```

1.9.1.3. Vectơ cột

Vectơ cột có cấu tạo tương tự như vectơ hàng. Khi định nghĩa vectơ cột, các phần tử được phân cách nhau bởi kí tự ';' hoặc bởi 'newlines'.

Ví dụ:

```
>> c = [ 1; 3; sqrt(5)]
```

1.9.1.4. Toán tử hai chấm (:)

Toán tử này dùng để tạo ra vectơ hàng một cách nhanh chóng:

```
>> x = 1:4
```

Tổng quát:

a : b : c sẽ tạo ra một vectơ với các phần tử bắt đầu từ giá trị của a, tăng dần với bước tăng bằng giá trị của b, cho tới khi đạt tới giá trị của c.

```
>> 0.32:0.1:0.6
ans =
0.3200 0.4200 0.5200
```

Toán tử ':' còn được dùng để trích xuất một phần của vectơ.

Giả thiết chúng ta có vectơ

```
>> r = [1:2:6, -1:-2:-7]
r =
1 3 5 -1 -3 -5 -7
```

Để trích ra các phần tử từ thứ 3 đến thứ 6 ta có thể dùng lệnh:

```
>> r(3:6)
ans =
5 -1 -3 -5
```

1.9.1.5. Làm việc với vectơ và ma trận

a. Các phép toán số học

+ Cộng, trừ ứng với các vectơ có cùng chiều dài.

Ví dụ 1:

```
>> v1 = [1 2 3];
>> v2 = [4 5 6];
>> v1+v2
ans =
5 7 9
```

+ Một vectơ cũng có thể nhân được với một đại lượng vô hướng (một số), bằng cách Matlab tiến hành với từng phần tử.

Để thực hiện các phép tính toán: Nhân, chia và lũy thừa, Matlab đưa ra các toán tử '.*', './' và '.^'.

Ví dụ 2:

```
>> v1.*v2
ans =
4 10 18
```

+ Toán tử lũy thừa có thể được sử dụng theo hai cách, lũy thừa số vô hướng hoặc lũy thừa vectơ:

```
>> v2.^2
ans =
```

16 25 36

Trong đó:

+ Dấu (.*) có ý nghĩa của một phép nhân ma trận hay vectơ.

+ Tất cả các hàm số học dựng sẵn của Matlab được thiết kế để hoạt động với các vectơ và ma trận, vì vậy chúng ta có thể xây dựng các diễn giải đại số hoạt động với từng phần tử của vectơ.

Ví dụ 3:

```
>> x = [1 2 3]; y = [4 5 6];
>> s = 2*sqrt(x) + x./y – x.^3.*cos(pi*y)
s =
1.2500 11.2284 – 23.0359
```

Chú ý: Các phép tính của các đại lượng vô hướng trên các vectơ khác với cách làm việc phần tử với phần tử.

2*sqrt(x) là nhân số vô hướng với vectơ, trong khi x/y thì ta cần phải sử dụng (x./y).

+ Các phép cộng và trừ phần tử với phần tử không cần thiết phải sử dụng (.+) và (.–).

b. Ghép các vectơ

Để tạo ra một vectơ từ những vectơ có trước thì kích thước của chúng phải tương thích với nhau.

Ví dụ 1:

```
>> w = [1 2 3], z = [8 9]
>> cd = [2*z,-w]
>>sort(cd)
w =
1 2 3
z =
8 9
cd =
16 18 –1 –2 –3
ans =
–3 –2 –1 16 18
```

Trong đó:

Câu lệnh cuối cùng (sort) dùng để sắp xếp các phần tử của vectơ theo chiều tăng dần. Ngoài ra, ta cũng có thể sử dụng các lệnh cat, vertcat, horzcat để ghép nối các vectơ.

c. Các lệnh cho biết thông tin về ma trận (vectơ)

size - kích thước theo mỗi chiều

length - kích thước của chiều dài nhất (đặc biệt là cho vectơ)

ndims - số chiều

find - các chỉ số của các phần tử khác 0

d. Chuyển vị ma trận

Để chuyển đổi một vectơ hàng thành một vectơ cột (và ngược lại) bằng một quá trình gọi là 'chuyển vị' kí hiệu bằng: (').

Ví dụ 1:

```
>> w, w', c, c'
w =
1 −2 3
ans =
1
−2
3
c =
1.0000
3.0000
2.2361
ans =
1.0000 3.0000 2.2361
```

e. Xử lí dữ liệu với các hàm dựng sẵn cho vectơ

- sort sắp xếp dữ liệu,
- max tìm giá trị lớn nhất,
- min tìm giá trị nhỏ nhất,
- sum tính tổng,
- mean tìm giá trị trung bình,
- std tìm độ lệch quân phương.

1.9.2. Ma trận

1.9.2.1. Cú pháp

+ Giống với vectơ, các khoảng trống (hoặc dấu phẩy) phân cách các phần tử trong một hàng và các dấu chấm phẩy là kí hiệu cho biết bắt đầu một hàng mới sau đó.

Ví dụ 1:

>> A = [2 -1 0 0; 1 1 2 3; -1 4 0 5]

Matlab sẽ đưa ra kết quả

A =

2 -1 0 0

1 1 2 3

-1 4 0 5

+ Các phần tử đơn lẻ của một ma trận có thể được chỉnh sửa như với các vectơ.

Ví dụ 2:

Lệnh A(3,2) = 0 sẽ thay thế giá trị phần tử cột 2, hàng 3 của ma trận A thành 0.

Có nhiều lệnh để khởi tạo một số dạng ma trận đặc biệt

zeros(n,m) - tạo ma trận với tất cả các phần tử = 0,

ones(n,m) - tạo ma trận với tất cả các phần tử = 1,

eye(n) - tạo ma trận đơn vị kích thước n × n.

+ Để khởi tạo một ma trận vuông đặc biệt, ta có thể sử dụng dạng ngắn zeros(n), lệnh này ngầm định số hàng và số cột của ma trận là bằng nhau.

1.9.2.2. Một số ma trận đặc biệt

eye - ma trận đơn vị,

zeros - ma trận với tất cả các phần tử = 0,

ones - ma trận với tất cả các phần tử = 1,

diag - ma trận đường chéo (hoặc chích xuất một đường chéo),

toeplitz - ma trận với mỗi đường chéo bằng một hằng số,

triu - ma trận tam giác trên,

tril - ma trận tam giác dưới,

rand - ma trận với các phần tử ngẫu nhiên (từ −1 đến 1),

linspace - ma trận với các phần tử cách đều nhau,

cat - móc nối các ma trận với nhau theo một chiều đã định,

repmat - xây dựng ma trận mới bằng cách lặp một vectơ theo một chiều.

1.9.2.3. Các phép toán với từng phần tử trong ma trận

Cộng: X= A + B

Trừ: X= A – B

Nhân: X= A * B

X.*A nhân các phần tử tương ứng với nhau

Chia: X = A/B lúc đó X*B = A

X = A\B lúc đó A*X = B

X = A./B chia các phần tử tương ứng với nhau

Luỹ thừa: X = A^2

và X = A.^2

Nghịch đảo: X = inv(A)

Định thức: d = det(A)

cond(A) điều kiện

norm(A) chuẩn hoá ma trận

eig(A) giá trị riêng, vectơ riêng.

§2. CÁC PHƯƠNG PHÁP GẦN ĐÚNG

2.1. Phương pháp chia đôi

Để sử dụng phương pháp chia đôi ta xây dựng hàm bisection()

```
function [x,err,xx] = bisection(f, a, b, tolx, maxiter)
%bisection.m de giai pt f(x) = 0 bang phuong phap chia doi cung
%vao: f = ham can tim nghiem
% a/b = bien cua doan can tim nghiem
% tolx = sai so mong muon
% maxiter lan lap max
%ra: x = nghiem
% err = sai so
% xx = cac gia tri trung gian
tol = eps;
fa = feval(f, a);
fb = feval(f, b);
if fa*fb > 0
```

```
error('Nghiem khong o trong doan nay');
end
for k = 1: maxiter
xx(k) = (a + b)/2;
fx = feval(f, xx(k));
err = (b – a)/2;
if abs(fx) < tol | abs(err) < tolx
break;
elseif fx*fa > 0
a = xx(k);
fa = fx;
else b = xx(k);
end
end
x = xx(k);
if k = = maxiter
fprintf('Khong hoi tu sau %d lan lap\n', maxiter),
else
fprintf('Hoi tu sau %d lan lap\n',k),
end
```

Ví dụ 1: Để tìm nghiệm của hàm f(x) = tan(pi - x) - x

```
clear all, clc
f = inline('tan(pi – x) – x');
[x, ss, xx] = bisection(f, 1.6, 3, 1e–4 , 50).
```

2.2. Phương pháp lặp

Để sử dụng phương pháp lặp ta dùng hàm simpiter()

```
function [x, err, xx] = simpiter(g, x0, tolx, maxiter)
% giai pt x = g(x) tu x0 bang cach lap
%vao: g, x0 = ham va gia tri dau
% tolx = sai so mong muon
% maxiter = so lan lap max
%ra: x = nghiem
```

```
% err = sai so |x(k) – x(k – 1)|
% xx = cac gia tri trung gian
if nargin < 4
maxiter = 100;
end
if nargin < 3
tolx = 1e–6;
end
xx(1) = x0;
for k = 2:maxiter
xx(k) = feval(g, xx(k – 1));
err = abs(xx(k) – xx(k – 1));
if err < tolx
break;
end
end
x = xx(k);
if k == maxiter
fprintf('Khong hoi tu sau %d lan lap\n', maxiter)
else
fprintf('Hoi tu sau %d lan lap\n',k)
end
```

Ví dụ 1: Để tính lại ví dụ trên ta dùng hàm simpiter()

```
clear all, clc
f = inline('-0.5*((x - 1).^2 - 3)');
[x, ss, xx] = simpiter(f, 0.5,.00001,200).
```

2.3. Đa thức nội suy lagrange

Dùng hàm lagrange() để thực hiện việc nội suy hàm số:

```
function [l, L] = lagrange(x, y)
%Dua vao: x = [x0 x1 ... xn], y = [y0 y1 ... yn]
%ket qua: l = He so cua da thuc Lagrange bac n
% L = Da thuc Lagrange
```

```
n = length(x) → 1; %bac cua da thuc/
l = 0;
for m = 1:n + 1
p = 1;
for k = 1:n + 1
if k ~= m
p = conv(p, [1 →x(k)])/(x(m) → x(k));
end
end
L(m, :) = p; %da thuc Lagrange
l = l + y(m)*p;
end
```

Ví dụ 1: Cho hàm dưới dạng bảng:

X	−2	−1	1	2
Y	−6	0	0	6

và tìm y(2.5) ta dùng hàm largrange():

```
clear all, clc
x = [−2 −1 1 2];
y = [−6 0 0 6];
l = lagrange(x, y);
yx = polyval(l, 2.5)
```

2.4. Đa thức Newton

Dùng hàm newton() để nội suy:

```
function [n,DD] = newton(x,y)
%Dua vao : x = [x0 x1 ... xN]
% y = [y0 y1 ... yN]
%Lay ra: n = he so cua da thuc newton bac N
N = length(x) - 1;
DD = zeros(N + 1, N + 1);
DD(1:N + 1, 1) = y';
for k = 2:N + 1
```

```
for m = 1: N + 2 - k
DD(m,k) = (DD(m + 1, k – 1) – DD(m, k – 1))/(x(m + k – 1) – x(m));
end
end
a = DD(1, :);
n = a(N+1);
for k = N:–1:1
n = [n a(k)] – [0 n*x(k)];
end
```

Ví dụ 1: Cho hàm dưới dạng bảng:

X	-2	-1	1	2	4
Y	-6	0	0	6	60

Ta dùng hàm newton() để nội suy:

```
clear all, clc
x = [-2 -1 1 2 4];
y = [-6 0 0 6 60];
a = newton(x, y)
yx = polyval(a, 2.5)
```

2.5. Cách tìm cực trị của hàm

Phương pháp newton: Việc tìm điểm cực tiểu của hàm f(x) tương đương với việc xác định x để cho gradient g(x) của hàm f(x) bằng zero. Nghiệm của g(x) = 0 tìm được bằng cách dùng phương pháp newton cho hệ phương trình phi tuyến.

Hàm newtons(x) dùng để tìm nghiệm của phương trình g(x) = 0 là:

```
function [x, fx, xx] = newtons(f, x0, tolx, maxiter)
h = 1e-4;
tolfun = eps;
EPS = 1e-6;
fx = feval(f, x0);
nf = length(fx);
nx = length(x0);
if nf ~= nx
```

```
error('Kich thuoc cua g va x0 khong tuong thich!');
end
if nargin < 4
maxiter = 100;
end
if nargin < 3
tolx = EPS;
end
xx(1, :) = x0(:).';
for k = 1: maxiter
dx = -jacob(f, xx(k, :), h)\fx(:);%-[dfdx]^-1*fx
xx(k + 1, :) = xx(k, :) + dx.';
fx = feval(f, xx(k + 1, :));
fxn = norm(fx);
if fxn < tolfun | norm(dx) < tolx
break;
end
end
x = xx(k + 1, :);
if k == maxiter
fprintf('Ket qua tot nhat sau %dlan lap\n', maxiter)
end
function g = jacob(f, x, h) %Jacobian cua f(x)
if nargin < 3
h = 1e-4;
end
h2 = 2*h;
n = length(x);
x = x(:).';
I = eye(n);
for n = 1:n
g(:, n) = (feval(f, x + I(n, :)*h) - feval(f, x - I(n,:)*h))'/h2;
end
```

Ví dụ 1:

Để tìm cực tiểu của hàm bằng phương pháp newton ta có:

```
clear all, clc
f = inline('x(1).^2 - x(1)*x(2) - 4*x(1) + x(2).^2 - x(2)');
g = inline('[(2*x(1) - x(2) -4)  (2*x(2) - x(1) - 1)]');
x0 = [0.1  0.1];
tolx = 1e-4;
maxiter = 100;
[x0, f0] = newtons(g, x0, tolx, maxiter)
```

2.6. Tính đạo hàm bằng phương pháp nội suy

Giả sử ta có hàm cho dưới dạng bảng

X	x_0	x_1	x_0	...	x_n
Y	y_0	y_0	y_0	...	x_n

Để tìm đạo hàm của hàm tại một điểm nào đó, ta nội suy hàm đó rồi tính đạo hàm của hàm tại điểm đã cho.

Dùng hàm diffinterp() để thực hiện công việc trên.

```
function df = diffinterp(x, y, n, x0)
%Tinh dao ham cap 1 hoặc 2 bang phuong phap noi suy
px = lagrange(x, y); % Tim da thuc noi suy Lagrange (x, y)
[p, dp, ddp] = peval(px, x0);
fprintf('Tri so cua ham la:  %f\n',p)
if n ==1
df = dp;
else
df = ddp;
end
fprintf('Dao ham cap %d la: %f\n',n, df);
```

Ví dụ 1:

Để tính đạo hàm ta dùng hàm diffinterp ():

```
clear, clc
x0 = pi/4;
x = [2:6]*pi/16;
```

```
y = sin(x);
x = [1.5  1.9  2.1  2.6  3.2];
y = [1.0628  1.3961  1.5432  1.8423  2.0397];
n = 2;
df = diffinterp(x, y, n, x0);
```

2.7. Tính vi phân

2.7.1. Phương pháp chuỗi Taylor

Dùng hàm Taylor() để tính vi phân:

```
function [xout, yout] = taylor(deriv, x, y, x1, h)
% Tich phan chuoi Taylor bac 4.
% x, y = cac gia tri dau; i la vecto hang.
% x1 = gia tri cuoi cua x
if size(y,1) > 1;
y = y';
end
xout = zeros(2, 1);
yout = zeros(2, length(y));
xout(1) = x;
yout(1, :) = y;
k = 1;
while x < x1
h = min(h,x1 - x);
d = feval(deriv,x,y); % Dao ham cua [y]
hh = 1;
for j = 1:4 % tao chuoi Taylor
hh = hh*h/j;
y = y + d(j, :)*hh;
end
x = x + h;
k = k + 1;
xout(k) = x;
yout(k,:) = y;
end
```

Ví dụ 1: Để giải phương trình

Ta dùng hàm taylor ()

```
clear all, clc
y = @f5;
a = 0;
b = 2;
ya = [0 1];
h = 0.2;
[x, y] = taylor(y, a, ya, b, h)
plot(x, y);
```

2.7.2. Phương pháp Euler

Dùng hàm euler() để thực hiện:

```
function [X, Y] = euler(fxy, xo, xf, yo, n)
%Giai phuong trinh  y'(x) = f(x,y(x)) hay y' = f(x)
if n < 2
n = 2;
end
h = (xf – xo)/n;
X = zeros(n+1, 1);
M = max(size(yo));% so phuong trinh (so cot cua ma tran Y)
Y = zeros(n+1, M);
%dat dieu kien dau
x = xo;
X(1) = x;
y = yo;
Y(1,:) = y';
for i = 1:n
if nargin(fxy) > 1
k1 = h*feval(fxy, x, y);
else
k1 = h*feval(fxy, x);
end
```

```
y = y + k1;
x = x + h;
X(i+1) = x;
Y(i+1, :) = y';
end
function dy = f1(t, y)
dy = zeros(3, 1);
dy(1) = y(2) * y(3);
dy(2) = -y(1) * y(3);
dy(3) = -0.51 * y(1) * y(2);
```

Ví dụ 1:

Để giải phương trình cho bởi hàm f1(x, y) ta dùng hàm euler ()

```
clear all, clc
a = 0;
b = 1;
y = @f1;
ya = [0 1 1]';
m = 200;
[x, y] = euler(y, a, b, ya, m)
plot(x, y);
```

2.8. Tính tích phân

2.8.1. Công thức hình thang

Dùng hàm trapezoid() để thực hiện thuật toán.

```
function J = trapezoid(f, a, b, maxiter, tol)
% Quy tac hinh thang lap.
% Cu phap: J = trapezoid(f, a, b, k)
fa = feval(f, a);
fb = feval(f, b);
J1 = (fa + fb)*(b - a)/2;
for k = 2:maxiter
n = 2^(k -2 ); % so diem moi
h = (b – a)/n; % khoang chia moi
```

```
x = a + h/2.0; % toa do diem moi thu nhat
sum = 0.0;
for i = 1:n
fx = feval(f, x);
sum = sum + fx;
x = x + h;
end
J = (J1 + h*sum)/2;
if abs(J1 – J) < tol
break;
end
J1 = J;
end
```

Ví dụ 1: Để tính tích phân ta dùng hàm trapezoid ()

```
clear all, clc
f = inline('(x^3+1)*sin(x)','x');
a = 0;
b = 1;
maxiter = 50;
tol = 1e-6;
J = trapezoid(f, a, b, maxiter, tol)
```

2.8.2. Công thức Simpson

Dùng hàm simpson() để thực hiện thuật toán:

```
function s = simpson(f, a, b, n)
%n so khoang chia
% neu  f chua trong mot file dung ki hieu @ de goi
% s = simpson(@f, a, b, n).
% neu f la ham inline
% s = simpson(f, a, b, n).
if mod(n, 2) ~= 0
n = n + 1
end
h = (b – a)/(2*n);
```

```
s1 = 0;
s2 = 0;
for k = 1:n
x = a + h*(2*k-1);
s1 = s1+ f(x);
end
for k = 1:(n-1)
x = a + h*2*k;
s2 = s2 + f(x);
end
s = h*(f(a) + f(b) + 4*s1 + 2*s2)/3;
clc
```

Ví dụ 1: Để tính tích phân ta dùng hàm simpson ():

```
clear all, clc
f = inline('exp(x).*sin(x)','x');
a = 0;
b = 1;
n = 6;
s = simpson(f, a, b, n).
```

§3. M-FILE

Trong mục trước chúng ta đã biết cách sử dụng các phép toán với các biến, để mở rộng khả năng ứng dụng Matlab, cần sử dụng M-file nhằm hỗ trợ tương tác trong quá trình lập trình và tính toán.

3.1. Giới thiệu M-file

M-file là các file ASCII (file text) chứa các (câu) lệnh Matlab, các file này có phần mở rộng là '.m'.

Có hai loại M-file: Script và Function. Các Script và Function files cũng hoạt động gần như các Procedures và Functions trong các ngôn ngữ lập trình thông dụng khác. Về cơ bản nội dung của một script file được hiểu giống như khi nội dung đó được gõ vào tại dấu nhắc cửa sổ nhập lệnh. Hiểu đơn giản thì nó chỉ thực hiện một chuỗi các câu lệnh của Matlab.

Ưu điểm của phương pháp sử dụng script là:

– Tạo ra và xem xét, chỉnh sửa một chuỗi nhiều dòng lệnh (thường là 4, 5 dòng trở lên).

– Có thể dễ dàng xem lại hoặc thực hiện lại công việc sau này.

– Chạy các tính toán (công việc) đòi hỏi cường độ cao của CPU trên nền, xử lí kết quả, lưu lại tự động và cho phép log-off (liên quan tới UNIX).

3.2. Biên soạn và thực thi M-file

+ *Biên soạn:* Matlab cung cấp công cụ biên soạn các M-file khá tốt, đó là Matlab editor. Tuy nhiên có thể tự do sử dụng các ứng dụng soạn thảo khác cho file text của Windows như Notepad, Textpad...

+ *Khởi động Matlab Editor:* Từ menu File/New/M-file, hoặc nhấn tổ hợp phím tắt 'Ctrl – N', hay cách nhấn vào nút 'New Doccument' trên thanh công cụ, cách đánh vào cửa sổ nhập lệnh 'edit' và tên file (nếu file chưa tồn tại trong thư mục hiện thời, Matlab sẽ hỏi để khẳng định rằng người dùng muốn tạo ra một file mới với tên như vậy).

+ *Soạn thảo các câu lệnh và Save:*

Để biết trong thư mục hiện tại (current directory) có những M-file nào, có thể sử dụng lệnh

>> what

Để xem nội dung của một M-file, hãy nháy đúp vào file đó để mở nó ra hoặc đánh lệnh

>> type tên–file

+ *Thực thi:* Để có thể thực thi một M-file, thì M-file cần phải tồn tại trong thư mục hiện thời (xem cửa sổ Current Directory), ta có thể di chuyển giữa các thư mục trong ổ cứng gần giống như với trình duyệt Exprorer của Windows hoặc dùng lệnh editpath (path là đường dẫn đến thư mục mà Matlab sẽ tìm kiếm file ở đó).

+ *Biên dịch:* Không cần thiết biên dịch cả hai loại M-file của Matlab. Muốn thực hiện các lệnh chứa trong file này chỉ cần đánh tên file (không cần phần mở rộng '.m') từ dấu nhắc cửa sổ lệnh. Các chỉnh sửa đã tiến hành với file và ghi lại vào ổ đĩa sẽ được thực thi khi gọi function hay script đó lần sau.

Ví dụ: Gọi thực thi các lệnh có trong file baitap2.m như sau:

>> baitap2

Chỉ có các thông số đầu ra sẽ được thể hiện trên màn hình, chứ không phải bản thân các câu lệnh. Để có thể xem các câu lệnh có trong file cùng lúc với các thông số đầu ra, đánh lệnh

>> echo on và lệnh 'echo off' sẽ tắt chức năng này.

3.3. Chú thích (comments)

Một dạng quan trọng trong M-file là câu chú thích, được bắt đầu bằng kí tự phần trăm (%). Bất cứ phần text nào sau kí tự '%' trên một dòng lệnh sẽ được Matlab bỏ qua không thực hiện (trừ trường hợp kí tự % là một phần của chuỗi kí tự giữa hai dấu nháy ' ').

Mục đích chính của tính năng này là cho phép bổ sung các câu chú thích (comments) vào script file, mô tả rõ ràng mục đích, tính năng các lệnh, đoạn, vòng lặp, biến... để tiện theo dõi và dễ dàng chỉnh sửa chương trình.

Ví dụ: Giả sử trong file baitap2.m có nội dung sau:

```
% script nay giai quyet cac bai tap ve nha
% lien quan toi kien thuc chuyen nganh ki thuat bien
% z= rand(1);
% muc nuoc bien
a=omega*t*sin(2*pi); % bien do song...
```

Khi một người dùng khác ngồi vào máy tính của bạn, muốn biết những thông tin cơ bản nhất xem file baitap2.m viết về vấn đề gì, họ có thể đánh vào cửa sổ lệnh:

```
>>help baitap2
```

và kết quả nhận được sẽ là: 'script nay giai quyet cac bai tap ve nha', 'lien quan toi kien thuc chuyen nganh ki thuat bien'.

3.4. Các hàm M-file (function M-files)

Trước tiên chúng ta cần phân biệt các hàm M-file và các hàm số dựng sẵn, hàm trong một dòng.

+ Hàm dựng sẵn: sqrt( ), log( ), exp( ), sin( )...

+ Hàm trong một dòng (inline function): Là cách đơn giản nhất mà người dùng có thể định nghĩa một hàm.

Ví dụ 1:

```
>> f = inline('x*sin(x)+2'), f(5)
f =
Inline function:
f(x) = x*sin(x)+2
ans =
-2.7946
```

+ Hàm với M-file: Dùng cho các hàm phức tạp hơn, chẳng hạn như các vòng lặp, câu điều kiện... bạn cần dùng M-file để khai báo các hàm đó.

Tiếp theo cần phân biệt các hàm M-file và các script-file:

+ Script M-file không phải là một hàm. Nó không có các tham số đầu vào cũng như đầu ra và đơn giản nó chỉ thực hiện một chuỗi các câu lệnh của Matlab với các biến được định nghĩa trong không gian làm việc.

+ Hàm M-file khác với script M-file ở chỗ nó có một dòng định nghĩa hàm, qua đó liên hệ giữa các tham số đầu vào và đầu ra. Hàm là cách chủ yếu để phát huy khả năng của Matlab. So với các script, các hàm có khả năng phân chia nhiệm vụ tốt hơn nhiều.

Các bước chính cần tuân theo khi khai báo một hàm trong Matlab là:

– Đặt tên cho hàm, lưu ý rằng tên đó không được xung đột với các tên đã được Matlab dành trước. Dòng đầu tiên của file này cần có dạng thức như sau:

function[các outputs] = tên–hàm(các inputs)

Một M-file hàm hoàn chỉnh có thể có dạng như sau:

```
function [A] = dientich(a,b,c)
% Tinh dien tich cua mot tam giac
% khi biet chieu dai 3 canh la a, b va c.
% Dau vao:
% a,b,c: Chieu dai cua cac canh
% Dau ra:
% A: Dien tich tam giac
% Cach su dung (cu phap):
% Dientichcantinh = dientich(2,3,4);
% Nguoi viet: Ng.Ba.Tuyen, 2007.
s = (a+b+c)/2;
A = sqrt(s*(s-a)*(s-b)*(s-c));
%%%%%%%%% ket thuc dientich %%%%%%%%%
```

Một khía cạnh quan trọng khác của hàm M-file là hầu hết các hàm xây dựng trong Matlab (trừ những hàm lõi toán học) đều là các M-file mà ta có thể đọc và copy.

3.5. Đọc dữ liệu từ file và ghi ra file

Nhập dữ liệu trực tiếp từ bàn phím sẽ trở nên không thể (không thực tế) khi lượng dữ liệu quá lớn. Dữ liệu đó được dùng cho phân tích nhiều lần. Trong những trường hợp này thì người sử dụng Matlab sẽ chọn cách nhập/xuất dữ liệu với file dữ liệu. Hai lệnh save và load có chức năng ghi và đọc giá trị của các biến vào/ra từ đĩa.

Khi làm việc với file dữ liệu, điều cốt yếu cần lưu ý là định dạng của dữ liệu phải đúng. Định dạng dữ liệu là chìa khóa quyết định việc biên dịch dữ liệu.

Có hai dạng file dữ liệu: formatted và unformatted (có định dạng và không định dạng).

– File dữ liệu có định dạng sử dụng các định dạng chuỗi để khai báo chính xác xem dữ liệu được lưu ở vị trí nào.

– File dữ liệu không định dạng thì khác, nó chỉ định rõ định dạng của số.

Ví dụ 1: Giả sử file dữ liệu dạng số được lưu với tên 'table.dat' trong thư mục hiện hành:

100 2256
200 4564
300 3653
400 6798
500 6432

Để sử dụng ta dùng 3 lệnh sau

```
>> fid = fopen('table.dat','r');
>> a = fscanf(fid,'%3d%4d');
>> fclose(fid);
```

Giải thích:

– Mở một file để đọc, việc này được chỉ định bằng chuỗi 'r', (r là viết tắt của read). Biến fid được gán cho một giá trị bằng một số nguyên tố duy nhất, đặc trưng cho file sẽ sử dụng (số này còn gọi là số chỉ thị của file).

– Đọc vào bộ nhớ từng cặp số từ file (file có số chỉ thị là fid), một số có 3 chữ số và một số có 4 chữ số.

– Đóng file (file có số chỉ thị là fid).

Quá trình này tạo ra một vectơ cột chứa các phần tử 100 2256

200 4564

...

500 6432.

§4. SYMBOLIC MATH TOOLBOXE

4.1. Khái niệm chung

Symbolic Math Toolboxe kết hợp tính toán bằng chữ vào môi trường Matlab.

Các toolbox này bổ sung các tiện ích số và đồ thị với các kiểu tính toán học khác nhau.

Tiện ích	Nội dung
Calculus	Đạo hàm, tích phân, giới hạn, tổng và chuỗi Taylor
Linear Algebra	Nghịch đảo, định thức, giá trị riêng, phân tích và dạng chính tắc của ma trận
Simplification	Phương pháp rút gọn các biểu thức đại số
Solution of Equations	Giải bằng số, chữ các phương trình đại số và vi phân
Variable-Precision Arithmetic	Đánh giá độ chính xác của các biểu thức đại số
Transform	Biến đổi Laplace, Fourrier và z
Special Mathematical Function	Các hàm toán học đặc biệt của các ứng dụng toán học kinh điển

4.2. Khởi động Toolbox

4.2.1. Các đối tượng chữ

Symbolic Math Toolbox định nghĩa một kiểu dữ liệu Matlab mới gọi là đối tượng chữ hay sym. Một đối tượng chữ là một cấu trúc số liệu mà nó biểu diễn chuỗi các kí tự.

Symbolic Math Toolbox dùng các đối tượng chữ để biểu diễn các biến chữ, các biểu thức chữ và các ma trận chữ.

4.2.2. Tạo các biến và các biểu thức chữ

Lệnh sym cho phép ta xây dựng các biến và các biểu thức chữ.

Ví dụ 1:

```
x = sym('x')
a = sym('alpha')
```

Tạo ra các biến chữ là x và a, với x là x và a là alpha.

Ví dụ 2: Giả sử khi muốn dùng biến chữ để biểu diễn tỉ lệ: $\rho = \frac{1+\sqrt{5}}{2}$, ta dùng lệnh:

rho = sym('(1 + sqrt(5))/2')

Ví dụ 3: Khi giải phương trình bậc 2: $f = a.x^2 + b.x + c = 0$ ta dùng lệnh:

f = sym('a*x^2 + b*x + c')

Gán biểu thức chữ $a.x^2 + b.x + c$ cho biến f. Tuy nhiên trong trường hợp này Symbolic Math Toolbox không tạo ra các biến tương ứng với các số hạng a, b, c và x trong biểu thức. Để thực hiện các phép toán bằng chữ (tích phân, đạo hàm,...) ta phải tạo các biến một cách rõ ràng, nghĩa là

a = sym('a')

b = sym('b')

c = sym('c')

x = sym('x')

hay đơn giản là:

syms a b c x

Chú ý: Có thể dùng sym hay syms để tạo các biến chữ nhưng nên dùng syms để tiết kiệm thời gian.

4.3. Biến đổi giữa số và chữ

4.3.1. Tạo các biến thực và phức

Lệnh sym cho phép mô tả các thuộc tính toán học của các biến chữ bằng cách dùng tuỳ chọn real.

x = sym('x','real');

y = sym('y', 'real');

hay hiệu quả hơn:

syms x y real

z = x + i*y

Chú ý:

– Để xóa thuộc tính real của x ta dùng lệnh: syms x unreal

hay: x = sym('x', 'unreal')

– Lệnh clear x không xoá thuộc tính số real của x.

4.3.2. Tạo các hàm trừu tượng

Để tạo một hàm trừu tượng (nghĩa là một hàm không xác định) f(x) ta dùng lệnh

f = sym('f(x)')

lúc này f hoạt động như là f(x) và có thể xử lí bằng các lệnh toolbox.

Ví dụ 1: Để tính vi phân bậc 1, ta viết:

```
df = (subs(f, 'x', 'x+h') – f)/ 'h'
```

hay:

```
syms x h
df = (subs(f,x,x+h)–f)/h
```

Kết quả:

```
df =
(f(x+h)–f(x))/h
```

4.3.3. Dùng sym để truy cập các hàm của Matlab

Có thể truy cập hàm giai thừa k! của Matlab khi dùng sym.

```
kfac = sym('k!')
```

Ví dụ 1: Để tính 6! ta viết:

```
syms k n
subs(kfac,k,6)
ans =
720
```

4.3.4. Tạo ma trận chữ

Một ma trận vòng là ma trận mà hàng sau có được bằng cách dịch các phần tử của hàng trước đi một lần.

Ví dụ 1: Tạo ma trận vòng A bằng các phần tử a, b và c:

```
syms a b c
A = [a b c; b c a; c a b]
```

kết quả:

```
A =
    [a, b, c]
    [b, c, a]
    [c, a, b]
```

Do A là ma trận vòng tổng mỗi hàng và cột như nhau:

```
sum(A(1,:))
ans =
a+b+c
sum(A(1,:)) = = sum(A(:,2))
```

ans =

1

Bây giờ ta thay A(2, 3) bằng beta và b bằng alpha:

syms alpha beta

A(2,3) = beta;

A = subs(A,b,alpha)

A =

[a, alpha, c]

[alpha, c, beta]

[c, a, alpha]

Trong Matlab các đối tượng chữ cũng dùng tương tự như các đối tượng số.

4.3.5. Biến chữ

Khi sử dụng các hàm toán học, việc chọn các biến độc lập thường rất rõ ràng như bảng hàm 1.

Bảng 1. Bảng hàm 1

Hàm toán học	Lệnh Matlab
f = xn	f = x^n
g = sin(at + b)	g = sin(a*t+b)
h = Jv(z)	h = besselj(nu,z)

Tuy nhiên ta có thể lấy đạo hàm của f theo n bằng cách viết biến độc lập ra.

Ví dụ 1:

syms x n

f = x^n;

Để đạo hàm hàm f ta có:

diff(f):

ans =

x^n*n/x

Trong ví dụ trên x là biến độc lập. Nếu muốn tính đạo hàm của f theo n ta có:

diff(f,n)

ans =

x^n*log(x)

4.4. Tạo các hàm toán học bằng chữ

4.4.1. Các biểu thức chữ

Ví dụ 1:

```
x y z
r = sqrt(x^2 + y^2 + z^2)
t = atan(y/x)
f = sin(x*y)/(x*y)
```

Tạo ra các biểu thức chữ r, t và f.

Ngoài ra ta có thể dùng các lệnh diff, int, subs hay các lệnh Symbolic Math Toolbox khác để tính các biểu thức.

4.4.2. Tạo các M-file

M-file cho phép tính các hàm tổng quát hơn.

Ví dụ 1: Tạo ra hàm sinc = sin(x)/x ta sẽ viết một M-file có nội dung như sau:

```
function z = sinc(x)
if isequal(x, sym(0))
z = 1;
else
z = sin(x)/x;
end
```

4.5. Tính toán

4.5.1. Đạo hàm

Ví dụ 1: Tạo biểu thức chữ:

```
syms a x
f = sin(a*x)
df = diff(f)
df =
cos(a*x)*a
```

Chú ý: Để tính đạo hàm của hàm số bất kì chúng ta sử dụng bảng 2.

Bảng 2. Bảng hàm 2

Hàm toán học	Lệnh Matlab
$f = xn$ $f' = nxn^{-1}$	F = x^n diff(f) hay diff(f, x)
g = sin(at+b) g' = acos(at+b)	g = sin(a*t+b) diff(g) hay diff(g, t)

Ví dụ 2: Để tính đạo hàm bậc 2 của f theo x và a ta có:

```
diff(f,2)
ans =
-sin(a*x)*a^2
diff(f,x,2)
ans =
-sin(a*x)*x^2
```

Hàm diff dùng đối số là ma trận, trong trường hợp này đạo hàm được thực hiện trên từng phần tử.

Ví dụ 3:

```
syms a x
A = [cos(a*x),sin(a*x);-sin(a*x),cos(a*x)]
A =
[cos(a*x), sin(a*x)]
[-sin(a*x), cos(a*x)]
dy = diff(A)
dy =
[-sin(a*x)*a,    cos(a*x)*a]
[cos(a*x)*a,     -sin(a*x)*a]
```

Bảng 3. Bảng tổng hợp hàm diff và hàm jacobian

Toán tử toán học	Lệnh Matlab
f = exp(ax + b)	syms a b x f = exp(a*x + b)
$\frac{df}{dx}$	diff(x) hay diff(f, x)
$\frac{df}{da}$	diff(f,a)
$\frac{d^2f}{d^2a}$	diff(f,a,2)
$r = u^2 + v^2$ $t = \arctan(v/u)$	syms r t u v r = u^2 + v^2 t = atan(v/u)

Toán tử toán học	Lệnh Matlab
$J = \frac{\partial(r,t)}{\partial(u,v)}$	J = jacobian ([r; t], [u, v])

Chuyển đổi từ toạ độ Euclid (x, y, z) sang toạ độ cong (r, λ, φ) bằng các công thức biến đổi tọa độ:

x = rcosλcosφ

y = rcosλsinφ

z = rsinλ

Để tính ma trận Jacobi J của phép biến đổi ta dùng hàm jacobian. Có dạng:

$$J = \frac{\partial(x,y,z)}{\partial(r,\lambda,\varphi)}$$

Để dễ viết ta dùng kí tự *l* thay cho λ và f thay cho φ rồi vào lệnh

```
syms r l f
x = r*cos(l)*cos(f);
y = r*cos(l)*sin(f);
z = r*sin(l);
J = jacobian([x; y; z], [r l f])
```

cho ta kết quả:

```
J =
[cos(l)*cos(f),   -r*sin(l)*cos(f),   -r*cos(l)*sin(f)]
[cos(l)*sin(f),   -r*sin(l)*sin(f)    r*cos(l)*cos(f)]
[sin(l),          r*cos(l),                        0]
```

và lệnh:

```
detJ = simple(det(J))
```

cho:

```
detJ =
-cos(l)*r^2
```

4.5.2. Giới hạn

Symbolic Math Toolbox cho phép tính trực tiếp giới hạn của một hàm.

```
dc = limit( (cos(x+h) - cos(x))/h, h, 0 )
dc =
-sin(x)
```

Trong một số trường hợp limit(f) tương đương với limit(f,x,0).

Bảng 4. Bảng giới hạn

Hàm toán học	Lệnh Matlab
$\lim_{x\to 0} f(x)$	limit(f)
$\lim_{x\to a} f(x)$	limit(f, x, a) hay limit(f, a)
$\lim_{x\to a^-} f(x)$	limit(f, x, a, 'left')
$\lim_{x\to a^+} f(x)$	limit(f, x, a, 'right')

4.5.3. Tích phân

Bảng 5. Bảng tích phân

Hàm toán học	Lệnh Matlab
$\int x^n dx = \frac{x^{n+1}}{n+1}$	int(x^n) hay int(x^n, x)
$\int_0^{\frac{\pi}{2}} \sin(2x)dx = 1$	int(sin(2*x), 0, pi/2) hay int(sin(2*x), x, 0, pi/2)
$g = \cos(at + b)$ $\int f(t)dt = \frac{1}{a}\sin(at + b)$	g = cos(a*t) + b) int(g) hay int(g, t)

Nếu f là một biểu thức chữ thì int(f) tìm một biểu thức khác F sao cho diff(F) = f. Như vậy int(f) cho ta tích phân bất định của f. Tương tự như đạo hàm int(f, v) lấy tích phân theo biến độc lập v.

Khi Matlab không tìm được tích phân thì sẽ tự động viết lại lệnh đã nhập vào.

Ví dụ 1:

Cách 1: syms x

f = exp(−(k*x)^2);

int(f, x);

ezplot(f)

Cách 2: syms x

```
f = exp(-(k*x)^2);
a = int(f, x, 0, 1);
a = double(a)
```

4.5.4. Tính tổng

Tính tổng bằng cách dùng lệnh: symsum

Ví dụ 1: Tính: $1+\frac{1}{2^2}+\frac{1}{3^2}...$

Ta có:

```
syms x k
s1 = symsum(1/k^2, 1, inf)
```

Ví dụ 2

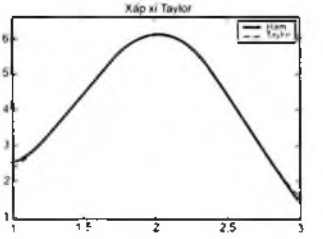
```
syms x k
s1 = symsum(1/k^2,1,inf)
s2 = symsum(x^k,k,0,inf)
s1 = 1/6*pi^2
s2 = -1/(x-1)
```

4.5.5. Chuỗi Taylor

```
syms x
g = exp(x*sin(x))
t = taylor(g,12,2)
```

Để khai triển Taylor hàm f(x) tại x = 2 và chứa đến 12 số hạng khác 0. Ta vẽ các hàm này trên cùng một đồ thị để thấy được khả năng xấp xỉ của chuỗi Taylor với hàm thực g:

```
xd = 1:0.05:3;
yd = subs(g,x,xd);
ezplot(t, [1,3]);
hold on;
plot(xd, yd, 'r-.')
title('Xap xi Taylor');
legend('Ham','Taylor')
```



4.6. Rút gọn biểu thức

Xét 3 biểu thức:

```
syms x
f = x^3-6*x^2+11*x-6
g = (x-1)*(x-2)*(x-3)
h = x*(x*(x-6)+11)-6
```

Dùng các lệnh pretty(f), pretty(g), pretty(h) ta được:

$f = x^3 - 6x^2 + 11x - 6$

$g = (x - 1)(x - 2)(x - 3)$

$h = x(x(x - 6) + 11) - 6$

Symbolic Math Toolbox cung cấp một số hàm dùng để rút gọn các biểu thức đại số và lượng giác thành các biểu thức đơn giản, bao gồm: collect, expand, horner, factor, simplify và simple.

4.6.1. Collect

Cú pháp: collect(f)

Ví dụ 1:

f	collect(f)
(x – 1)(x – 2)(x – 3)	x^3 – 6*x^2 + 11*x – 6
x*(x*(x – 6) + 11) – 6	x^3 – 6*x^2 + 11*x – 6
(1 + x)*t + x*t	2*x*t + t

4.6.2. Expand

Cú pháp: expand(f)

Ví dụ 1:

f	expand(f)
a*(x + y)	a*x + a*y
(x – 1)*(x – 2)*(x – 3)	x^3 – 6*x^2 + 11*x – 6
x*(x*(x – 6) + 11) – 6	x^3 – 6*x^2 + 11*x – 6
exp(a + b)	exp(a) + exp(b)
cos(x + y)	cos(x)*cos(y) – sin(x)*sin(y)
cos(3*acos(x))	4*x^3 – 3*x

4.6.3. Horner

Cú pháp: horner(f)

Ví dụ 1:

f	horner(f)
x^3 - 6*x^2 + 11*x - 6	-6 + (11 + (-6 + x)*x)*x
1.1 + 2.2*x + 3.3*x^2	11/10 + (11/5 + 33/10*x)*x

4.6.4. Factor

Cú pháp: factor(f)

Trong đó: f là đa thức, hàm factor biểu diễn f như là tích của các đa thức có bậc thấp hơn.

Ví dụ 1:

f	factor(f)
x^3 - 6*x^2 + 11*x - 6	(x - 1)*(x - 2)*(x - 3)
x^3 - 6*x^2 + 11*x - 5	x^3 - 6*x^2 + 11*x - 5
x^6 + 1	(x^2 + 1)*(x^4 - x^2 + 1)

Ví dụ 2: Phân tích đa thức $x^n + 1$ thành thừa số:

```
syms x;
n = 1:9;
x = x(ones(size(n)));
p = x.^n + 1;
f = factor(p);
[p; f].'
```

Kết quả trả về ma trận với các đa thức ở cột thứ nhất và các kết quả ở cột thứ 2:

```
[   x+1,                                  x + 1]
[x^2+1,                                   x^2+1]

[x^3+1,                          (x+1)*(x^2-x+1)]
[x^4+1,                                   x^4+1]
[x^5+1,                  (x+1)*(x^4-x^3+x^2-x+1)]
[x^6+1,                  (x^2+1)*(x^4-x^2+1)]
[x^7+1,    (x+1)*(1-x+x^2-x^3+x^4-x^5+x^6)]
```

```
[x^8+1,                          x^8+1]
[x^9+1,          (x+1)*(x^2-x+1)*(x^6-x^3+1]
```

Hàm factor có thể phân tích các đối tượng chữ có chứa số nguyên thành thừa số.

Ví dụ 3:

```
one = '1'
for n = 1:11
N(n ,:) = sym(one(1, ones(1, n)));
end
[N factor(N)]
```

cho kết quả:

```
[           1,                  1]
[          11,                 (11]
[         111,               (3)*(37)]
[        1111,              (11)*(101)]
[       11111,              (41)*(271)]
[      111111,     (3)*(7)*(11)*(13)*(37)]
[     1111111,           (239)*(4649)]
[    11111111,     (11)*(73)*(101)*(137)]
[   111111111,       (3)^2*(37)*(333667)]
[  1111111111,     (11)*(41)*(271)*(9091)]
[ 11111111111,          (513239)*(21649)]
```

4.6.5. Simplify

Hàm simplify là một hàm mạnh, dùng rút gọn các biểu thức. Sau đây là một số ví dụ:

f	simplify(f)
x*(x*(x – 6) + 11) – 6	x^3 – 6*x^2 + 11*x – 6
(1 – x^2)/(1 – x)	x + 1
(1/a^3 + 6/a^2 + 12/a + 8)^(1/3)	((2*a + 1)^3/a^3)^(1/3)
syms x y positive log(x*y)	log(x) + log(y)
exp(x)* exp(y)	exp(x + y)
cos(x)^2 + sin(x)^2	1

4.6.6. Simple

Hàm simple đưa ra dạng ngắn nhất có thể có của một biểu thức.

Ví dụ 1:

syms x

simple(cos(x)^2 + sin(x)^2)

Sau đây là một số ví dụ:

f	simple(f)
cos(x)^2 + sin(x)^2	1
2*cos(x)^2 – sin(x)^2	3*cos(x)^2 – 1
cos(x)^2 – sin(x)^2	cos(2*x)
cos(x) + (–sin(x)^2)^(1/2)	cos(x) + i*sin(x)
cos(x) + i*sin(x)	exp(i*x)
cos(3*acos(x))	4*x^3 – 3*x

4.6.7. Nhân hai đa thức

Để nhân hai đa thức ta sử dụng lệnh conv

Ví dụ 1: Nhân hai đa thức $y1 = 2x^2 + 3x + 1$ và $y2 = 3x^2 + 4x$

>> y1 = [2 3 1]

>> y2 = [3 4 0]

>> y3 = conv(y1,y2)

>> y3 = 6 17 15 4 0

Trong đó: Hàm conv dùng để nhân hai đa thức với nhau, nếu muốn nhân nhiều đa thức thì ta phải sử dụng nhiều lệnh conv.

4.7. Thay số

Xét ví dụ giải phương trình bậc hai $ax^2 + bx + c = 0$.

syms a b c x

s = solve(a*x^2 + b*x + c);

Muốn giải cụ thể giá trị của x với a = 1, b = 2, c = 4 thì dùng các lệnh:

a = 1;

b = 2;

c = 4;

x = subs(s)

4.8. Giải phương trình

4.8.1. Phương trình đại số

+ Nếu S là biểu thức chữ thì dùng lệnh solve(S) → Tìm giá trị của biến kí tự trong S để S = 0.

Ví dụ 1:

```
syms a b c x
S = a*x^2 + b*x + c;
solve(S)
```

cho kết quả:

```
ans =
[ 1/2/a*(-b+(b^2-4*a*c)^(1/2))]
[ 1/2/a*(-b-(b^2-4*a*c)^(1/2))]
```

Đây là vectơ chữ mà các phần tử của nó là 2 nghiệm của phương trình.

+ Nếu muốn tìm nghiệm với một biến, ta phải chỉ rõ biến như một thông số.

Ví dụ 2: Nếu muốn giải S theo b ta có:

```
b = solve(S,b)
```

kết quả:

```
b =
-(a*x^2+c)/x
```

+ *Chú ý:* Nếu muốn giải phương trình có dạng f(x) = q(x) ta phải sử dụng chuỗi.

Ví dụ 3: Giải phương trình x^3-2*x^2 = x-1.

```
s = solve('x^3-2*x^2 = x-1')
```

kết quả:

```
s =
[1/6*(28+84*i*3^(1/2))^(1/3)+14/3/(28+84*i*3^(1/2))^(1/3)+2/3]
[-1/12*(28+84*i*3^(1/2))^(1/3)-7/3/(28+84*i*3^(1/2))^(1/3
+2/3+1/2*i*3^(1/2))^(1/3)-7/3/(28+84*i*3^(1/2))^(1/3)
-14/3/(28+84*i*3^(1/2))^(1/3))]
[-1/12*(28+84*i*3^(1/2))^(1/3)-7/3/(28+84*i*3^(1/2))^(1/3)
+2/3-1/2*i*3^(1/2))*(1/6*(28+84*i*3^(1/2))^(1/3)
-14/3/(28+84*i*3^(1/2))^(1/3))]
```

4.8.2. Hệ phương trình đại số

Ví dụ 1: Giải hệ phương trình: $\begin{cases} x^2.y^2 = 0 \\ x - \frac{y}{2} = \alpha \end{cases}$

```
syms x y alpha
[x, y] = solve(x^2*y^2, x – (y/2) – alpha)
```

kết quả:

```
x =
[      0]
[      0]
[ alpha]
[ alpha]
y =
[ –2*alpha]
[ –2*alpha]
[          0]
[          0]
```

Sau đó viết vectơ nghiệm:

```
v = [x, y]
```

cho ta:

```
v =
[      0, –2*alpha]
[      0, –2*alpha]
[ alpha,       0]
[ alpha,       0]
```

+ Cách gán các nghiệm trên chỉ thích hợp với hệ có ít phương trình. Với hệ có nhiều phương trình dùng solve tạo ra một cấu trúc các trường là các nghiệm.

Ví dụ 2: Giải hệ phương trình: $\begin{cases} u^2 + v^2 = a^2 \\ u + v = 1 \\ a^2 - 2a = 3 \end{cases}$

```
S = solve('u^2–v^2 = a^2', 'u + v = 1','a^2–2*a = 3')
```

kết quả:

S =

a: [2x1 sym]

u: [2x1 sym]

v: [2x1 sym]

Các nghiệm là các trường của S.

S.a

Tạo ra:

ans =

[−1]

[3]

Tương tự ta tìm được nghiệm u và v. Cấu trúc S bây giờ có thể được xử lí bằng trường và chỉ số để truy cập đến các phần riêng biệt của nghiệm.

Ví dụ 3: Nếu ta muốn kiểm tra nghiệm thứ 2 ta có:

s2 = [S.a(2), S.u(2), S.v(2)]

s2 =

[3, 5, −4]

Ví dụ 4: Nếu hệ phương trình là tuyến tính, ta có thể dùng ma trận để giải hệ.

clear u v x y

syms u v x y

S = solve(x+2*y−u, 4*x+5*y−v);

sol = [S.x;S.y]

A = [1 2; 4 5];

b = [u; v];

z = A\b

kết quả:

sol =

[−5/3*u+2/3*v]

[4/3*u−1/3*v]

z =

[−5/3*u+2/3*v]

[4/3*u−1/3*v]

4.8.3. Phương trình vi phân

Để giải phương trình vi phân ta dùng hàm dsolve.

Cú pháp của dsolve được mô tả trong bảng sau:

Bảng 6. Bảng cú pháp của dsolve giải phương trình vi phân không có điều kiện

Cú pháp	Phạm vi
y = dsolve('Dyt = y0*y')	Một phương trình, một nghiệm
[u,v] = dsolve('Du = v', 'Dv = u')	Hai phương trình, hai nghiệm
S = dsolve('Df = g', 'Dg = h', 'Dh = –f S.f, S.g, S.h)	Ba phương trình, giải ra cấu trúc nghiệm

Ví dụ 1:

```
syms t
dsolve('Dy = 1 + y^2')
```

kết quả:

```
ans =
tan(t-C1)
```

Để mô tả điều kiện ban đầu, ta dùng:

```
y = dsolve('Dy = 1+y^2', 'y(0) = 1')
```

kết quả:

```
y =
tan(t + 1/4*pi)
```

Ví dụ 2: Các phương trình phi tuyến có thể có nhiều nghiệm, dù đã cho điều kiện đầu.

```
x = dsolve(' (Dx)^2 + x^2 = 1','x(0) = 0')
```

kết quả:

```
x =
[-sin(t)]
[ sin(t)]
```

Ví dụ 3: Giải phương trình bậc 2 với 2 điều kiện đầu.

```
y = simplify(dsolve('D2y = cos(2*x) - y', 'y(0) = 1', 'Dy(0) = 0', 'x'))
```

kết quả:

y =

−2/3*cos(x)^2 + 1/3 + 4/3*cos(x)

Ví dụ 4: Để giải phương trình: $\frac{d^3u}{dx^3} = u$

Với điều kiện u(0) = 1, u'(0) = 1, u''(0) = π.

ta có:

u = dsolve('D3u = u', 'u(0) = 1', 'Du(0) = −1', 'D2u(0) = pi', 'x')

4.8.4. Hệ phương trình vi phân

Ví dụ 1: Hệ phương trình: y' = 3f + 4g

g' = −4f + 3g

S = dsolve('Df = 3*f + 4*g', 'Dg = −4*f + 3*g')

Nghiệm được tính và trả về dưới dạng cấu trúc S:

S =

f: [1x1 sym]

g: [1x1 sym]

Ta có thể xác định giá trị của f và g bằng lệnh:

f = S.f

f =

exp(3*t)*(cos(4*t)*C1 + sin(4*t)*C2)

g = S.g

g =

−exp(3*t)*(sin(4*t)*C1 − cos(4*t)*C2)

Nếu cho cả điều kiện đầu ta có:

[f, g] = dsolve('Df = 3*f + 4*g, Dg = −4*f + 3*g', 'f(0) = 0, g(0) = 1')

f =

exp(3*t)*sin(4*t)

g =

exp(3*t)*cos(4*t)

Bảng 7. Bảng cú pháp của dsolve giải phương trình vi phân có điều kiện

Phương trình vi phân	Lệnh Matlab
$\frac{dy}{dt} + 4y(t) = e^{-t}$ $y(0) = 1$	y = dsolve('Dy + 4*y = exp(-t)', 'y(0) = 1')
$\frac{d^2y}{dx^2} + 4y(x) = e^{-2x}$ $y(0) = 0, y(\pi) = 0$	y = dsolve('D2y – 4*y = exp(–2*x)', 'y(0) = 0', 'y(pi) = 0', 'x')
$\frac{d^2y}{dx^2} = xy(x)$ $y(0) = 0, y(3) = \frac{1}{\pi}K_{\frac{1}{3}}(2\sqrt{3})$ (phương trình Airy)	y = dsolve('D2y = x*y','y(0) = 0', 'y(3) = besselk(1/3, 2*sqrt(3))/pi','x')

4.8.5. Phương trình bậc cao

Nghiệm phương trình bậc cao có thể tìm bằng lệnh Roots.

Ví dụ 1: Giải phương trình sau: $y = x^5 - 2x^4 + 5x^2 - 1$

```
>> y = [ 1  –2   0  5  0  –1]
y =
 1   –2    0    5    0   –1
>> kq=roots(y)
kq =
  1.5862 + 1.1870i
  1.5862 – 1.1870i
 –1.1606
 –0.4744
  0.4627
```

4.8.6. Biết nghiệm tìm lại phương trình

Biết nghiệm ta có thể tìm lại phương trình bằng lệnh poly

```
 >>A = [1 –1  2;1 3 4;2 –1 1];
  >>poly(A)
ans =
   1  –5   8   14
```

4.8.7. Chuyển từ hệ số sang phương trình

Chuyển từ hệ số sang phương trình bằng lệnh poly2sym

```
>> poly2sym([1 0 -2 -5])
ans =
x^3-2*x-5
>> y = [1 2 3 0 1]
y =   1   2   3   0   1
>> poly2sym(y)
ans =  x^4+2*x^3+3*x^2+1
```

4.8.8. Hệ phương trình tuyến tính

$$\begin{cases} 2x + 3y + z = 7 \\ 3x + 6y - 4z = 19 \\ x + y + z = 2 \end{cases}$$

```
>> A=[2 3 1;3 6 -4;1 1 1]
A =
 2   3   1
 3   6  -4
 1   1   1
>> B=[7;19;2]
B =
7
19
2
>> C=inv(A)
C =
-2.5000   0.5000   4.5000
1.7500  -0.2500  -2.7500
0.7500  -0.2500  -0.7500
>> kq=C*B
kq =
1.0000
2.0000
-1.0000
```

4.8.9. Hệ phương trình phi tuyến

Giải hệ phương trình phi tuyến bằng lệnh solve

$$\begin{cases} \sin(x)+y^2+\log(z)=7 \\ 3x+2^y+z^3=4 \\ x+y+z=2 \end{cases}$$

```
>>[x,y,z]=solve('sin(x)+y^2+log(z)=7','3*x+2^y+z^3=4','x+y+z=2')
x =  -2.34957562245720321874105364003 68
y =  2.68352691947852194272702390790 10
z =  1.66604870297868127601402973213 5
```

4.8.10. Hệ phương trình tham số

```
>>[a,u] = solve('a*u^2 + v^2= 0','u – v = 1','a,u')
a =
–v^2/(v^2+2*v+1)
u =
v+1
```

4.8.11. Hệ phương trình vi phân thường

```
>>y = dsolve('(D2y) =1','y(0) = 1')
y =  1/2*t^2+C1*t+1
>>[x,y]=dsolve('Dx = y', 'Dy = –x')
x=   cos(t)*C1+sin(t)*C2
y =  –sin(t)*C1+cos(t)*C2
```

4.9. Giới hạn

Ví dụ 1:

```
sym h n x
limit((cos(x + h) – cos(x))/h,h,0)
ans = –sin(x)
limit((1 + x/n)^n,n,inf)
ans=exp(x)
limit(x/abs(x),x,0,'left')
ans = –1
limit(x/abs(x),x,0,'right')
ans=1
```

limit(x/abs(x),x,0)

• ans = NaN

4.10. Biến đổi Fourier và Fourier ngược

4.10.1. Biến đổi Fourier

Biến đổi Fourier dùng để biến đổi phương trình vi phân thành phương trình đại số.

Cú pháp: F = fourier(f)

F = fourier(f, v)

F = fourier(f, v, u)

Các ví dụ biến đổi Fourier trong bảng sau:

Biến đổi Fourier	Lệnh Matlab
$f(x) = e^{-x^2}$ $F[f](w) = \int_{-\infty}^{\infty} f(x)e^{-iwx}dx = \sqrt{\pi}e^{-w^2/4}$	f = exp(-x^2) fourier(f) cho: pi^(1/2)*exp(-1/4*w^2)
$g(w) = e^{-\lvert w\rvert}$ $F[g](t) = \int_{-\infty}^{\infty} g(w)e^{-iwt}dt = \frac{2}{1+t^2}$	g = exp(-abs(w)) fourier(g) cho 2/(1+t^2)
$f(x) = xe^{-\lvert x\rvert}$ $F[f](u) = \int_{-\infty}^{\infty} f(x)e^{-ixu}dx = -\frac{4i}{(1+u^2)^2u}$	f = x*exp(-abs(x)) f = x*exp(-abs)(x)) cho -4*i/(1+u^2)^2*u

4.10.2. Biến đổi Fourier ngược

Khi biết hàm ảnh Fourier dùng biến đổi Fourier ngược ta tìm được hàm gốc.

Cú pháp: f = ifourier(F)

f = ifourier(F, u)

f = ifourier(F, v, u)

Ví dụ 1:

Biến đổi Fourier ngược	Lệnh Matlab
$f(w) = e^{w^2/4a^2}$ $F^{-1}[f]w(x) = \int_{-\infty}^{\infty} f(w)e^{iwx}dw = \frac{a}{\sqrt{\pi}}e^{-a^2x^2}$	syms a real f = exp(-w^2/(4*a^2)) F = ifourier(f) F = simple(F) cho

Biến đổi Fourier ngược	Lệnh Matlab
$g(x) = e^{-\|x\|}$	ha*exp(-x^2*a^2)/pi^(1/2) g = exp(-abs(x))
$F^{-1}[g](t) = \int_{-\infty}^{\infty} g(x)e^{ftx}dx = \frac{\pi}{1+t^2}$	ifourier(g) cho 1/(1+t^2)/pi
$f(w) = 2e^{-\|w\|} - 1$ $F-1[f](t) =$ $\int_{-\infty}^{\infty} f(w)e^{fwt}dw = \frac{2-\pi\delta(t)(1-t^2)}{\pi(1+t)}$	f = 2*exp(-abs(w))-1 simple(ifourier(f,t)) cho (2-pi*Dirac(t)-pi*Dirac(t)*t^2)/ (pi+pi*t^2)

4.11. Biến đổi Laplace và Laplace ngược

4.11.1. Biến đổi Laplace

Biến đổi Laplace dùng biến đổi phương trình vi phân thành phương trình đại số.

Cú pháp: laplace(F)

laplace(F, t)

laplace(F, w, z)

Ví dụ:

Biến đổi Laplace	Lệnh Matlab
$f(t) = t4$ $L[f] = \int_0^{\infty} F(t)e^{-st}dt = \frac{24}{s^5}$	f = t^4 laplace(f) cho 24/s^5
$g(s) = \frac{1}{\sqrt{s}}$ $L[g](t) = \int_0^{\infty} g(s)e^{-st}ds = \frac{\pi}{\sqrt{s}}$	g = 1/sqrt(s) laplace(g) cho 1/(s^(1/2))*pi^(1/2)
$f(t) = e-zt$ $L[f](x) = \int_0^{\infty} f(t)e^{-tx}dt = \frac{1}{x+a}$	f = exp(-a*t) laplace(f) cho 1/(x+a)

4.11.2. Biến đổi Laplace ngược

Khi có ảnh của hàm, ta có thể tìm lại hàm gốc bằng biến đổi Laplace ngược.

Cú pháp: F = ilaplace(L)

F = ilaplace(L, y)

F = ilaplace(L, y, x)

Ví dụ:

Biến đổi Laplace ngược	Lệnh Matlab
$f(s) = \frac{1}{s^2}$ $L^{-1}[f] = \frac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty} f(s)e^{st}ds = t$	f = 1/s^2 ilaplace(f) cho t
$g(t) = \frac{1}{t-a}$ $L^{-1}[g] = \frac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty} g(t)e^{xt}dt = xe^{ax}$	g = 1/(t-a) ilaplace(g) cho x*exp(a*x)
$f(u) = \frac{1}{u^2-a^2}$ $L^{-1}[f] = \frac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty} g(u)e^{xu}du = \frac{1}{2ae^{ax}} - \frac{1}{2ae^{-ax}}$	f = 1/(u^2-a^2) ilaplace(f) cho 1/(2*a*exp(a*x)) - 1/(2*a*exp(-a*x))

4.12. Các hàm giao tiếp

4.12.1. Lệnh fclose

Lệnh fclose giúp đóng file đang mở sau khi truy xuất xong.

Cú pháp: fclose(fid)

fid: tên biến trỏ đến file đang mở.

4.12.2. Lệnh fopen

Lệnh fopen mở file hoặc truy xuất dữ liệu của file đang mở.

Cú pháp: fid = fopen('fn')

fid = fopen('fn', 'p')

fid: tên biến trỏ đến file đang mở

fn: tên file (có thể đặt đường dẫn).

Tham số p được xác định:

'r': đọc.

'r+': đọc - ghi.

'w': xoá tất cả nội dung của file hoặc tạo một file mới và mở file đó để ghi.

'w+': xoá tất cả nội dung của file hoặc tạo một file mới và mở file đó để ghi và đọc.

4.12.3. Lệnh fprintf

Lệnh fprintf giúp ghi đoạn dữ liệu thành file.

Cú pháp: fprintf(fid, f)

fid: tên biến trỏ đến file cần ghi

f: các tham số để định dạng

Ví dụ 1: Tạo file exp.txt có nội dung sau

```
x = 0:2:10;
y = [x, x/2];
fid = fopen('exp.txt', 'w');
fprintf(fid, '%d', [2, inf]);
```

Gán file exp.txt vào biến a để xem nội dung

```
fid = fopen('exp.txt')
a = fscanf(fid, '%d', [2,inf]);
disp(a);
fclose(fid);
```

Kết quả

```
0246810
012345
```

4.12.4. Lệnh fread

Lệnh fread giúp đọc dữ liệu dạng nhị phân từ file.

Cú pháp: [a, c] =fscanf(fid)

[a, c] = fscanf(fid,s)

a: tên biến chứa dữ liệu được đọc vào

c: số phần tử được đọc vào

fid: tên biến trỏ đến file cần đọc

s: kích thước dữ liệu đầu vào, s được định dạng bởi các thông số

n: chỉ cần đọc n phần tử vào cột vectơ a.

inf: đọc đến hết file

[m,n]: chỉ đọc vào m cột, n hàng, n có thể là inf còn m thì không.

Ví dụ 1: file vd.txt có nội dung sau:

```
A B C
1 2 3
fid = fopen(vd.txt');
```

```
[a,c] = fread(fid);
disp(a);
disp(c);
a =
65
32
66
32
67
13
10
49
32
50
32
51
c =
12
```

Ví dụ 2:

```
fid = fopen('vd1.txt');
[a,c] = fread(fid, 4);
disp(a);
disp(c);
a=
65
32
66
32
c =
4
```

4.12.5. Lệnh fwrite

Lệnh fwrite giúp ghi dữ liệu dạng nhị phân thành file.

Cú pháp: fwrite (fid,a)

fid: tên biến trỏ đến file cần ghi

a: tên biến chứa dữ liệu

Ví dụ 1: Ghi đoạn dữ liệu của biến a thành file a.txt

```
a = [65  66  67]
fid = fopen('a.txt', 'w');
fwrite(fid, '%');
fwite(fid,a);
```

Gán file a.txt vào biến b để xem nội dung

```
fid = fopen('a.txt');
b = fscanf(fid, '%');
disp(b);
fclose(fid);
```

Kết quả

```
b = ABC
```

4.12.6. Lệnh sprintf

Lệnh sprintf làm hiển thị thông tin lên màn hình.

Cú pháp: s = sprintf('ts',ds)

s: biến chứa chuỗi số hiển thị trên màn hình

ts: các tham số định dạng

ds: danh sách các đối số

Ví dụ 1: Một số tham số định dạng

1) %d: đối số là số nguyên được viết dưới dạng thập phân

```
s = sprintf('đây là số: %d',–24)
s = đây là số: –2
```

2) %u: đối số là số nguyên được viết dưới dạng thập phân không dấu

```
s = sprintf('đây là số: %u',24)
s = đây là số: 24
```

3)%s: đối số là chuỗi kí tự

```
s = sprintf('đây là chuỗi: %s', 'Matlab')
s = đây là chuỗi: Matlab
```

4.12.7. Lệnh sscanf

Lệnh sscanf thực hiện đọc và định dạng lại chuỗi kí tự.

Cú pháp: [a,count] = sscanf(s, 'format', size)

a: tên biến chứa chuỗi kí tự sau khi được định dạng.

count: đếm số phần tử được đọc vào.

size: kích thước sẽ được đọc vào.

format: phần định dạng.

Ví dụ 1:

```
s = '3.12  1.2  0.23  2.56';
[a, count] = sscanf(s, '%f',3)
a =
3.1200
1.2000
0.2300
count =
3
```

BÀI TẬP

3.1. Tính giá trị của các hàm số cơ bản tại giá trị của biến số lần lượt là:

13.15; 12 + i.9; i.4

3.2. Tính tích phân:

1) $I = \int_{-\infty}^{+\infty} \frac{x^2+1}{x^4+1}.dx$

2) $I = \int_{-\infty}^{+\infty} \frac{1}{(x^2+1)^n}.dx \quad (n \in \mathbb{Z}^+)$

3) $I = \int_{0}^{+\infty} \frac{\cos x}{x^2+4}.dx$

4) $I = \int_{-\infty}^{+\infty} \frac{x.\sin x}{x^2-6x+20}.dx$

5) $I = \int_{-\infty}^{+\infty} \frac{2x.\sin x}{x^2-5x+6}.dx$

6) $I = \int_{0}^{+\infty} \frac{\sin x}{x.(x^4+x^2+1)^2}.dx$

7) $I = \int_{0}^{+\infty} \left(\frac{\sin x}{x}\right)^2.dx$

8) $I = \int_{-\infty}^{+\infty} \frac{\cos x}{(x^2+9).(x-1)}.dx$

3.3. Giải phương trình và hệ phương trình:

1) $\frac{1}{z-i} + \frac{2+i}{1-i} = \sqrt{2}$

2) $2x - e^x = \int_{0}^{x^2} \sin x^2 dx$.

3) $\sin z + e^{2z} = 17$ 4) $z^4 + 1 = 0$

5) $\begin{cases} z_1 + iz_2 = 1 \\ 2z_1 + z_2 = 1 + i \end{cases}$ 6) $\tan(2 - z) + z = e^z$

3.4. Tính các giá trị:

1) $z = e^{i\frac{\pi}{2}}$, 2) $z = \sin(4 - 3i)$,

3) $z = \ln(3 + i)$, 4) $z = \arctan(-i)$.

3.5. Giải các bất phương trình

1) $\sin x + x > e^{x+1}$, 2) $x^3 + 2x - 1 \le \sin(x + 1)$

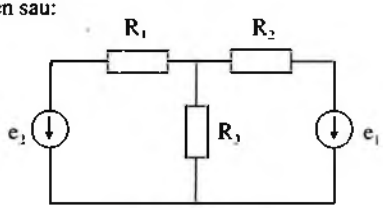
3.6. Một vật khối lượng $m = n.10^{-2}$ (kg) được gắn vào con lắc lò xo có độ cứng $k = 200$ N/m. Đầu kia của con lắc được gắn chặt vào tường, hệ dao động dọc theo trục Ox trên một mặt phẳng nhẵn nằm ngang với hệ số ma sát $\eta = 0,05$. Hệ tọa độ được chọn sao cho vật m khi ở trạng thái cân bằng nằm trùng với gốc tọa độ. Tại thời điểm ban đầu người ta cung cấp một động năng cho vật, sau đó đo đạc thực nghiệm xác định vị trí của vật có kết quả theo bảng sau:

T (s)	0	0,3	0,7	0,9	1,2	1,6	2	2,5	3,2
x(n cm)	0	0,7	1,95	1,8	0,3	–0,92	–1,1	0,1	–0,13

a. Chứng minh rằng còn tồn tại ngoại lực khác nữa tác dụng lên cơ hệ.

b. Xác định phương trình chuyển động, vận tốc, gia tốc của vật.

3.7. Cho mạch điện sau:



Cho thông số: $R_1 = 10(\Omega)$, $R_2 = 20(\Omega)$, $R_3 = 10(\Omega)$, $e_1 = 20(V)$, $e_2 = 30(V)$.

Tính dòng điện I_1, I_2, I_3 tương ứng đi qua R_1, R_2, R_3 bằng cách lập theo dạng hệ phương trình đại số tuyến tính ba ẩn số.

3.8. Cho phương trình vi phân mô tả dao động của một con lắc đơn:

$$\ddot{\varphi} + 0,002.\dot{\varphi} + \omega^2.\sin 2\varphi = 0$$

Với $\varphi = \varphi(t)$, $\varphi(0) = 0,005.n.(30 - n)$, $\varphi'(0) = 0$, $\omega = 2$.

Giải gần đúng nghiệm của phương trình nói trên.

Chương 4
ĐỒ HOẠ

Sau khi lập trình tính toán để giải quyết bài toán vật lí, cần phải thể hiện kết quả tính toán. Minh hoạ kết quả bằng hình ảnh là một trong những phương thức phổ biến nhất, vừa giúp chúng ta có cái nhìn tổng quát về kết quả lại vừa giúp chúng ta dễ dàng đánh giá, so sánh và tiếp tục nghiên cứu sâu hơn vào những điểm quan trọng của bài toán.

§1. ĐỒ THỊ HAI CHIỀU

1.1. Cú pháp

Cú pháp của đồ thị hai chiều là: plot(t, z)

Ví dụ 1: Biểu diễn sự biến thiên tăng giảm số liệu trong một dãy

z = [–0.05 0.18 0.28 0.33 0.19 0 –0.26 –0.35 –0.31 –0.22 0.05 0.14 0.31 0.38 0.18 0.09 –0.11 –0.20 –0.36 –0.11 0.08];

>>plot(z)

Trong đó:

– Lệnh plot(z) dùng vẽ biểu đồ dạng đường với số liệu cho bởi dãy số z.

– Trường hợp này trục hoành sẽ đánh số thứ tự lần lượt 1, 2, ...

Để biểu diễn số liệu z tại các thời điểm 0s, 10s, 20s... (cách nhau 10 giây) thì ta dùng

figure;

t = 0:10:200;

plot(t, z);

Trong đó: Lệnh figure có tác dụng tạo ra một hình mới.

Chú ý:

– Các dãy số cần vẽ thường có rất nhiều phần tử, do đó ta cần dùng dấu ';' ở cuối câu lệnh để ngăn không cho máy hiện lại nội dung của toàn bộ dãy.

– Để vẽ biểu đồ dạng đường nói chung ta thực hiện 3 bước sau:

+ Xây dựng một dãy số chứa các tọa độ x của các điểm,

+ Xây dựng một dãy số chứa các tọa độ y (Có thể là từ số liệu của một hàm giải tích hoặc từ đo đạc thực nghiệm),

+ Thực hiện lệnh vẽ plot(x, y).

1.2. Các tuỳ chọn của đồ thị hai chiều

1.2.1. Ghi nhãn và chọn kích thước các trục

Trong biểu đồ ta phải ghi nhãn (tiêu đề) của biểu đồ, điền các đại lượng và đơn vị lên các trục (x, y) bằng cách:

title('Tiêu đề biểu đồ ')

xlabel('Tiêu đề trục x')

ylabel('Tiêu đề trục y')

Ví dụ 1: Vẽ mẫu quan trắc mực nước theo thời gian, ta có:

title('Qua trinh muc nuoc thuc do')

xlabel('t (s)')

ylabel('z (m)')2

Trong một số trường hợp ta cần thể hiện từng phần của biểu đồ hoặc vì tính thẩm mĩ có thể chỉnh sửa trong phạm vi các trục.

Câu lệnh thực hiện như sau:

axis([xmin xmax ymin ymax])

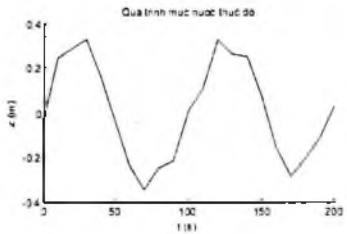
Trong đó:

xmin, xmax lần lượt là giới hạn trái và phải của trục hoành.

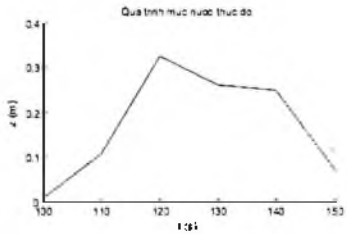
ymin, ymax lần lượt là giới hạn dưới và trên của trục tung.

Ví dụ 2: Khi muốn hiển thị mực nước chỉ trong khoảng thời gian từ t = 100s đến t = 150s và hạn chế cao độ mặt nước từ 0 đến 0.4m, ta gõ lệnh:

axis([100 150 0 0.4])



Hình 4.1. Ví dụ về biểu đồ dạng đường



Hình 4.2. Biểu đồ dạng đường sau khi chỉnh lại phạm vi các trục

1.2.2. Xóa

Để xoá toàn bộ đồ thị hiện thời, ta gõ:

>>clf

1.2.3. Lựa chọn màu vẽ, nét vẽ

Trường hợp đồ thị có nhiều đường nét vẽ khác nhau, ta phân biệt bằng những kiểu nét vẽ và màu sắc khác nhau.

plot(x, y, 'lựa chọn')

Trong đó: Lựa chọn là một chuỗi kí tự có 3 phần quy định như sau: md--

Với m là một kí tự chỉ màu vẽ, thường chữ đầu của từ tiếng Anh tương ứng d là kí hiệu đánh dấu các điểm nút.

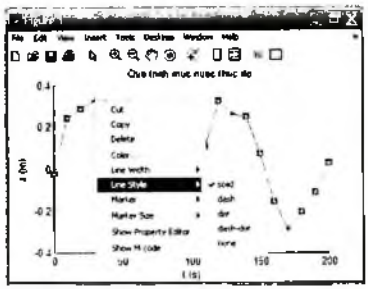
-- là một hay hai kí tự thể hiện kiểu nét vẽ.

Ví dụ 1:

– plot(x, y, 'r') vẽ đồ thị nét vẽ màu đỏ (red). Nếu không có lựa chọn kiểu nét vẽ cụ thể thì máy vẽ kiểu nét liền (mặc định).

– plot(x, y, 'g--') vẽ đồ thị nét vẽ xanh lục (green), nét vẽ được chọn là kiểu nét đứt (--).

– plot(x, y, 'b*-') vẽ đồ thị nét vẽ xanh lam (blue), nét vẽ được chọn là kiểu nét liền với các điểm dấu sao (*-).



Hình 4.3. Kiểu nét vẽ cùng các thuộc tính khác có thể lựa chọn trực tiếp

Sau khi chọn chế độ Edit Plot và nháy phải chuột vào đường biểu đồ.

Tổ hợp các kiểu màu và nét vẽ khác nhau theo bảng sau đây:

Màu vẽ		Nét vẽ		Điểm nút	
R	Red, đỏ	-	Nét liền	*	Dấu sao
G	Green, xanh lục	--	Nét đứt	+	Dấu cộng
B	Blue, xanh lam	:	Nét chấm	S	Hình vuông
K	Black, đen	.-	Nét chấm gạch	^	Tam giác

Trong đó:

– Theo mặc định, khi có một đường mới được vẽ thì đường cũ sẽ biến mất.

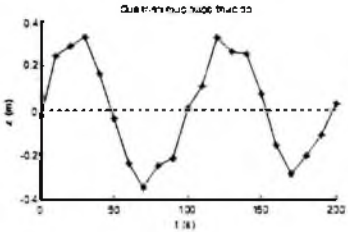
– Để vẽ nhiều đường trên cùng một đồ thị ta cần gõ lệnh: hold on trước khi vẽ các đường tiếp theo.

Ví dụ 2: Có thể vẽ thêm một đường nét đứt nằm ngang biểu thị mực nước bằng 0 theo cách sau:

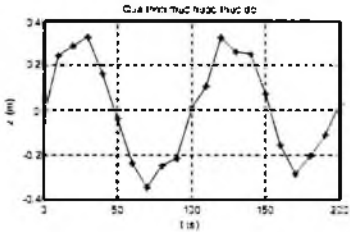
```
hold on;
plot([0 200],[0 0],':');
```

Ngoài ra trên đường mực nước có thể thêm các điểm dấu *:

```
plot(t,z,'*-');
```



Hình 4.4. Biểu đồ dạng đường với hai nét vẽ có kiểu khác nhau



Hình 4.5. Biểu đồ có khung và đường đóng

– Nếu muốn đóng khung đồ thị và tạo các đường đóng ta cần gõ vào các lệnh:

```
box on
grid on
```

– Ngược lại, để xóa các đường đóng khung và đường đóng, chỉ cần gõ:

```
box off
grid off
```

1.2.4. Tạo các chú thích, chú giải trên hình vẽ

1.2.4.1. Tạo các chú thích

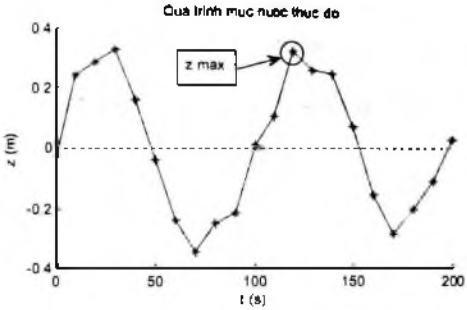
Cách tạo các chú thích thực hiện theo các bước sau:

– Vào menu Insert – Ellipse, vẽ một vòng tròn vào vị trí đỉnh của đường quá trình.

– Vào menu Insert – Text Box, vạch ra một khung chữ nhật và gõ vào z max.

– Vào menu Insert – Arrow, vạch mũi tên chỉ từ khung chữ vào vòng tròn.

– Muốn xóa các chú thích, chỉ cần chọn đối tượng cần xóa ấn Delete.



Hình 4.6. Biểu đồ với các chú thích

1.2.4.2. Chú giải (legend)

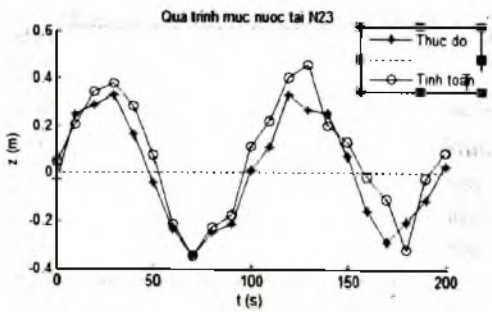
Khi có nhiều đường trên một đồ thị, cần có chú giải (legend) để phân biệt chúng. Giả sử trên đồ thị quá trình mực nước ở trên, bổ sung thêm một đường mực nước tính toán được từ mô hình:

z2 = [0.05 0.2 0.34 0.38 0.28 0.08 –0.21 –0.35 –0.23 –0.18 0.11 0.22 0.4 0.45 0.2 0.13 –0.02 –0.11 –0.32 –0.02 0.09];

hold on; plot(t,z2,'–o');

title('Qua trinh muc nuoc tai N23');

Trên thanh công cụ, ấn nút menu Insert – Legend. Sau đó nháy đúp thì phần chú giải xuất hiện, gõ vào tên chú giải cần đặt (Hình 4.7).



Hình 4.7. Chỉnh sửa tên chú giải tương ứng với các đường quá trình

1.2.5. Xóa đường biểu đồ, lưu biểu đồ

+ Nếu vẽ sai một đường biểu đồ nào đó, ta có thể xóa bằng cách chọn menu Tools–Edit Plot, chọn đường biểu đồ cần xóa, ấn Delete hoặc chọn Delete trong danh mục khi nháy phải chuột.

+ Matlab có thể lưu lại biểu đồ dưới nhiều dạng file ảnh chuẩn hiện nay: *.gif, *.png, *.jpeg, *.emf, *.eps v.v... Bên cạnh đó, Matlab còn có một dạng file riêng gọi là *.fig, trong đó lưu toàn bộ thông tin của các đường, nét, điểm... trên biểu đồ.

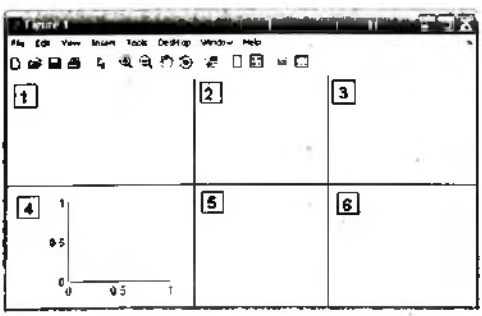
Để lưu biểu đồ trong Matlab ta chọn Menu File – Save, sau đó nhập tên cho file hình mà ta muốn lưu. Cũng có thể nhấn vào biểu tượng Save ( ) trên thanh công cụ.

1.3. Đồ thị hai chiều nâng cao

Bằng lệnh figure có thể tạo ra nhiều hình vẽ độc lập trên nhiều cửa sổ. Tuy vậy, khi muốn có một dãy (hoặc bảng) các biểu đồ xếp kế tiếp nhau, có kích thước bằng nhau để tiện việc so sánh thì Matlab hỗ trợ hệ thống subplot (biểu đồ nhỏ) với câu lệnh có dạng như sau:

subplot(m,n,k);

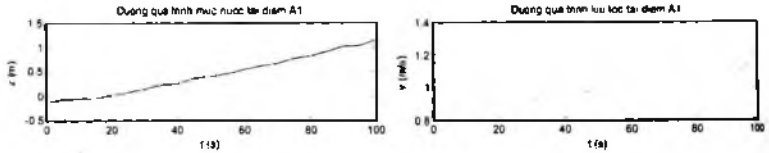
Lệnh này sẽ tạo ra một bảng gồm (m × n) biểu đồ nhỏ (m hàng và n cột). Tiếp đó hình thứ k (tính từ trên xuống dưới, trái qua phải) sẽ được kích hoạt và chuẩn bị được vẽ.



Hình 4.8. Vị trí các biểu đồ trong dãy xác định bằng lệnh subplot

Ví dụ 1: Ta cần vẽ đường quá trình mực nước (z) và vận tốc dòng chảy (v) theo thời gian (t) trên hai biểu đồ khác nhau. Để có sự đối chiếu về thời gian giữa hai biểu đồ ta nên xếp chúng theo một cột dọc. Như vậy m = 2 và n = 1.

```
t = 0:5:100;
    z = [-0.14 -0.08 -0.05 -0.04 0.01 0.07 0.15 0.23 0.25 0.37 0.4 0.45 0.55 0.6
0.66 0.76 0.82 0.91 1 1.03 1.14];
    v = [0.84 0.89 0.91 0.89 0.91 0.88 0.92 0.97 0.97 0.97 0.99 1.02 0.96 1.05
0.98 1.04 1.04 1.11 1.04 1.12 1.16];
    subplot 211
    plot(t, z); xlabel('t (phut)'); ylabel('z (m)');
    title('Duong qua trinh muc nuoc tai diem A1');
    subplot 212
    plot(t, v, 'g'); xlabel('t (phut)'); ylabel('v (m/s)');
    title('Duong qua trinh luu toc tai diem A1');
```



Hình 4.9. Hai đường quá trình trên hai subplot

§2. ĐỒ THỊ BA CHIỀU

2.1. Các lệnh cơ bản

Lệnh *mesh* và *surf* tạo ra mặt 3D từ ma trận số liệu.

Gọi ma trận số liệu là z mà mỗi phần tử của nó z(i,j) xác định cao độ của mặt thì *mesh(z)* tạo ra một lưới có màu thể hiện mặt z còn *surf(z)* tạo ra một mặt có màu z.

2.2. Đồ thị các hàm hai biến

Để thể hiện hàm hai biến z = f(x,y) là ma trận x và y (chứa các toạ độ trong miền xác định của hàm) ta dùng hàm meshgrid.

Ví dụ 1: Cho hàm z = sin(r)/r. Để tính hàm trong khoảng $-\infty < x, y < +\infty$ ta chỉ cần chuyển đổi số cho *meshgrid*

```
[x,y] = meshgrid(-∞:.5: +∞);
r = sqrt(x.^2 + y.^2) + 0.005;
```

Tiếp theo ta dùng hàm mesh để vẽ hàm.

```
z = sin(r)./r;
mesh(z)
```

2.3. Đồ thị đường đẳng mức

Các hàm contour tạo, hiển thị và ghi chú các đường đẳng mức của một hay nhiều ma trận gồm:

Clabel: tạo các nhãn sử dụng ma trận contour và hiển thị nhãn.

contour: hiển thị các đường đẳng mức tạo bởi một giá trị cho trước của ma trận Z.

contour3: hiển thị các mặt đẳng mức tạo bởi một giá trị cho trước của ma trận Z.

contourf: hiển thị đồ thị contour 2D và tô màu vùng giữa các đường

contourc: hàm cấp thấp để tính ma trận contour

Hàm meshc hiển thị contour và lưới và surfc hiển thị mặt contour.

Ví dụ 1:

```
[X,Y,Z] = peaks;
contour(X,Y,Z,20)
```

Mỗi contour có một giá trị gắn với nó. Hàm *clabel* dùng giá trị này để hiển thị nhãn đường đồng mức 2D.

Ma trận contour chứa giá trị clabel dùng cho các đường contour 2D. Ma trận này được xác định bởi *contour*, *contour3* và *contourf*.

Ví dụ 2: Để hiển thị 10 đường đẳng mức của hàm peak ta viết :

```
Z = peaks;
[C,h] = contour(Z,10);
clabel(C,h)
title({'Cac contour co nhan','clabel(C,h)'})
```

Hàm *contourf* hiển thị đồ thị đường đẳng mức trên một mặt phẳng và tô màu vùng còn lại giữa các đường đẳng mức. Để kiểm soát màu tô ta dùng hàm *caxis*.

```
Z = peaks;
[C,h] = contourf(Z,10);
caxis([20 20])
title({'Contour co to mau', 'contourf(Z,10)'})
```

Các hàm contour(z,n) và contour(z,v) cho phép chỉ rõ số lượng mức contour hay một mức contour cần vẽ nào đó với z là ma trận số liệu, n là số đường contour và v là vectơ các mức contour. Như vậy, *contour(z,v)* có vai trò giống như *contour(z,n)*.

Để hiển thị một đường đẳng mức ta cần cho v là một vectơ có hai phần tử và cả hai phần tử đều bằng mức mong muốn.

Ví dụ 3: Tạo ra một đường đẳng mức 3D bằng hàm peaks

```
xrange = -3:.125:3;
yrange = xrange;
[X,Y] = meshgrid(xrange,yrange);
Z = peaks(X,Y);
contour3(X,Y,Z)
```

Để hiển thị một mức ở Z = 1, ta cho v là [1 1]

```
v = [1 1]
contour3(X,Y,Z,v)
```

Hàm *ginput* cho phép ta dùng chuột hay các phím mũi tên để chọn các điểm vẽ, trả về toạ độ của vị trí con trỏ.

Ví dụ 4: Dùng hàm *ginput* và hàm spline để tạo ra đường cong nội suy hai biến.

```
disp('Chuot phai tro cac diem tren duong ve')
disp('Chuot trai tro diem cuoi cua duong ve')
axis([0 10 0 10])
hold on
x = [];
y =[];
n = 0;
but = 1;
while but = =1
[xi,yi,but] = ginput(1);
plot(xi,yi,'go')
n = n +1;
x(n,1) = xi;
y(n,1) = yi;
end
```

```
t = 1:n;
ts = 1:0.1:n;
xs = spline(t,x,ts);
ys = spline(t,y,ts);
plot(xs,ys,'c');
hold off
```

§3. MỘT VÀI DẠNG ĐỒ THỊ ĐẶC BIỆT

3.1. Vẽ các vectơ

Có nhiều cách để hiển thị các vectơ có hướng và vectơ vận tốc. Ta định nghĩa một vectơ bằng cách dùng một hay hai đối số. Các đối số này mô tả thành phần x và thành phần y của vectơ.

Nếu ta dùng hai đối số thì đối số thứ nhất sẽ mô tả thành phần x và đối số thứ hai mô tả thành phần y.

Nếu ta chỉ dùng một đối số thì Matlab xử lí đối số này như một số phức, phần thực là thành phần x và phần ảo là thành phần y.

Các hàm vẽ vectơ gồm:

compass	vẽ các vectơ bắt đầu từ gốc toạ độ của hệ toạ độ cực
feather	vẽ các vectơ bắt đầu từ một đường thẳng
quiver	vẽ các vectơ 2D có các thành phần (u, v)
quiver3	vẽ các vectơ 3D có các thành phần (u, v, w).

3.1.1. Hàm compass

Ví dụ 1: Vẽ hướng và tốc độ gió. Trong đó: Các vectơ xác định hướng góc tính bằng độ) và tốc độ gió (km/h) là:

```
hg = [45 90 90 45 360 335 360 270 335 270 335 335];
td = [6 6 8 6 3 9 6 8 9 10 14 12];
```

biến đổi hướng gió thành radian trước khi chuyển sang toạ độ vuông góc

```
hg1 = hg * pi/180;
[x,y] = pol2cart(hg1,td);
compass(x,y)
```

Tạo ghi chú trên đồ thị:

```
gc = ('Huong gio va suc gio tai san bay Da Nang')
text(-28,15,gc)
```

3.1.2. Hàm feather

Hàm feather hiển thị các vectơ bắt đầu từ một đường thẳng song song với trục x.

Ví dụ 1: Để tạo ra các vectơ có góc từ 90^0 đến 0^0 và cùng độ dài ta có:

```
theta = 90:-10:0;
r = ones(size(theta));
```

Trước khi vẽ, chuyển các số liệu sang toạ độ vuông góc và tăng độ lớn thành r để dễ nhìn

```
[u,v] = pol2cart(theta*pi/180,r*10);
feather(u,v)
axis equal
```

Nếu đối số là số phức z thì feather coi phần thực là x và phần ảo là y:

```
t = 0:0.3:10;
s = 0.05 + i;
z = exp(-s*t);
feather(z)
```

3.1.3. Hàm quiver

Hàm quiver hiển thị các vectơ ở các điểm đã cho trong mặt phẳng. Các vectơ này được xác định bằng các thành phần x và y.

Ví dụ 1: Để tạo ra 10 contour của hàm peaks ta có:

```
n = -2.0:.2:2.0;
[X,Y,Z] = peaks(n);
contour(X,Y,Z,10)
```

Bây giờ dùng hàm gradient để tạo các thành phần của vectơ dùng làm đối số cho quiver:

```
[U,V] = gradient(Z,.2);
```

Đặt hold on để thêm đường contour:

```
hold on
quiver(X,Y,U,V)
hold off
```

3.1.4. Hàm quiver3

Hàm quiver3 hiển thị các vectơ có các thành phần (u, v, w) tại điểm (x, y, z).

Ví dụ 1: Tính các thành phần của vectơ vận tốc của một chuyển động nhanh dần đều. Trước hết ta gán vận tốc ban đầu và gia tốc a:

v0 = 10; % Van toc ban dau

a = −32; % gia toc

Tiếp theo tính z tại các thời điểm:

t = 0:.1:1;

*z = vz*t + 1/2*a*t.^2;*

Tính vị trí theo hướng x và y:

vx = 2;

*x = vx*t; vy = 3; y = vy*t;*

Tính các thành phần của vectơ vận tốc và hiển thị bằng cách dùng quiver3:

u = gradient(x);

v = gradient(y);

w = gradient(z);

scale = 0;

quiver3(x,y,z,u,v,w,scale)

axis square

3.2. Đồ thị Logarit

Trong một số trường hợp, các đồ thị logarit cần được sử dụng, chẳng hạn biểu đồ đường cấp phối hạt. Muốn đặt thang logarit với trục hoành ta chỉ cần thay tên lệnh plot bằng semilogx.

Ví dụ 1: Ta cần vẽ đường cấp phối hạt với mẫu bùn cát sau:

Đường kính (mm)	Khối lượng (mg)
d < 0.15	900
0.15 < d < 0.21	2900
0.21 < d < 0.3	16000
0.3 < d < 0.42	20100
0.42 < d < 0.6	8900
0.6 < d	1200
toàn bộ	50000

Trước khi vẽ đồ thị, tính tỉ lệ bùn cát tương ứng với mỗi khoảng đường kính và tỉ lệ cộng dồn:

KhoiLuong = [900 2900 16000 20100 8900 1200];

TiLe = KhoiLuong / 50000;

Hàm cumsum giúp ta tính cộng dồn, chẳng hạn:

>> cumsum(TiLe)

ans =

0.0180 0.0760 0.3960 0.7980 0.9760 1.0000

tính tỉ lệ P theo phần trăm, ta có:

P = cumsum(TiLe) * 100;

Đường kính (mm)	Khối lượng (mg)	Tỉ lệ (%)	Tỉ lệ cộng dồn P (%)	Đường kính d (mm)
d < 0.15	900	1.8	1.8	0.15
0.15 < d < 0.21	2900	4.8	7.6	0.21
0.21 < d < 0.3	16000	32	39.6	0.3
0.3 < d < 0.42	20100	40.2	79.8	0.42
0.42 < d < 0.6	8900	17.8	97.6	0.6
0.6 < d	1200	2.4	100	1
Toàn bộ	50000	100		

Số liệu dùng để vẽ đồ thị là hai cột sau cùng: Tỉ lệ cộng dồn (P) và đường kính (d).

Ở đây giả thiết đường kính lớn nhất bằng 1mm như là giới hạn trên của biểu đồ.

d = [0.15 0.21 0.30 0.42 0.60 1];

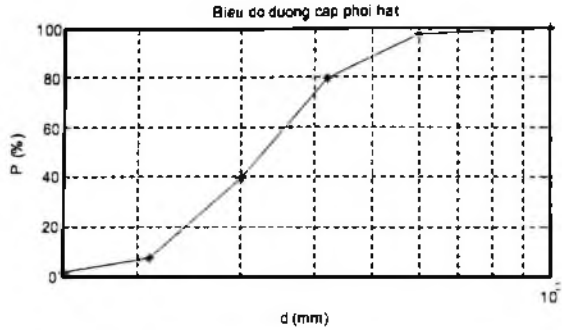
title('Bieu do duong cap phoi hat');

xlabel('d (mm)');

ylabel('P (%)');

semilogx(d,P,'*-');

grid on; box on;



Hình 4.10. Ví dụ biểu đồ có trục theo thang logarit

Các đồ thị với thang logarit trên trục y và trên cả hai trục cũng được thực hiện tương tự với các câu lệnh lần lượt là semilogy và loglog.

3.3. Các hiệu ứng hoạt hình

Ta có thể tạo ra hình chuyển động bằng 2 cách

– Tạo và lưu nhiều hình khác nhau và lần lượt hiển thị chúng.

– Vẽ và xoá liên tục một đối tượng trên màn hình, mỗi lần vẽ lại có sự thay đổi.

Với cách thứ nhất ta thực hiện hình chuyển động qua 3 bước:

+ Dùng hàm moviein để dành bộ nhớ cho một ma trận đủ lớn nhằm lưu các khung hình.

+ Dùng hàm getframes để tạo các khung hình.

+ Dùng hàm movie để hiển thị các khung hình.

Ví dụ 1: Sử dụng movie để quan sát hàm fft(eye(n)).

```
axis equal
M = moviein(16,gcf);
set(gca,'NextPlot','replacechildren')
h = uicontrol('style','slider','position',[100 10 500 20],'Min',1,'Max',16)
for j = 1:16
plot(fft(eye(j + 16)))
set(h,'Value',j)
M(:,j) = getframe(gcf);
end
```

clf;

axes('Position',[0 0 1 1]);

movie(M,30)

Bước đầu để tạo hình ảnh chuyển động phải gán ma trận. Tuy nhiên trước khi gọi hàm *moviein*, cần tạo ra các trục toạ độ có cùng kích thước với kích thước muốn hiển thị hình.

Trong ví dụ này để hiển thị các số liệu cách đều trên vòng tròn đơn vị ta dùng lệnh *axis equal* để xác định tỉ lệ các trục. Hàm *moviein* tạo ra ma trận đủ lớn để chứa 16 khung hình.

Cú pháp: set(gca, 'NextPlot', 'replacechildren' ngăn hàm plot đưa tỉ lệ các trục về axis normal mỗi khi được gọi.

Hàm *getframe* trả lại các điểm ảnh của trục hiện hành.

Mỗi khung hình gồm các số liệu trong một vectơ cột.

Hàm *getframe(gcf)* chụp toàn bộ phần trong của một cửa sổ hiện hành. Sau khi tạo ra hình ảnh ta có thể chạy chúng với số lần nhất định như 30 lần nhờ hàm *movie(M,30)*.

Để tạo hình chuyển động bằng cách vẽ và xoá, nghĩa là vẽ một đối tượng đồ hoạ rồi thay đổi vị trí của nó bằng cách thay đổi toạ độ x, y và z một lượng nhỏ nhờ một vòng lặp. Ta có thể tạo ra các hiệu ứng khác nhau nhờ cách xoá hình khác nhau như:

– none	Matlab không xoá đối tượng khi nó di chuyển
– background	Matlab xoá đối tượng bằng cách vẽ nó có màu nền
– xor	Matlab chỉ xoá đối tượng

Ví dụ 2:

A = [– 8/3 0 0; 0 –10 10; 0 28 –1];

y = [35 –10 –7];

h = 0.01;

p = plot3(y(1),y(2),y(3),..., ...

('EraseMode','none','MarkerSize',5); % dat EraseMode ve none

axis([0 50: 25 25: 25 25])

hold on

for i = 1:4000

A(1,3) = y(2);

A(3,1) = –y(2);

```
ydot = A*y;
y = y + h*ydot;
set(p,'Xdata',y(1),'Ydata',y(2),'Zdata',y(3)) % thay doi toa do
drawnow
i = i + 1;
end
```

BÀI TẬP

4.1. Cho phương trình vi phân mô tả dao động của một con lắc đơn:

$$\ddot{\varphi} + 0{,}002.\dot{\varphi} + \omega^2.\sin 2\varphi = 0$$

Với $\varphi = \varphi(t)$, $\varphi(0) = 0{,}005.n.(30 - n)$, $\varphi'(0) = 0$, $\omega = 2$.

Mô tả dao động nói trên bằng hình ảnh động.

4.2. Một sợi dây đồng chất có chiều dài L, hai đầu mút gắn chặt, khi cân bằng nó nằm ngang và trùng với Ox. Lúc ban đầu người ta nâng vị trí $x = c$ lên phía trên có độ cao h ($0 < c < L$) và buông tay không vận tốc ban đầu. Hãy mô tả hình ảnh dao động tự do của sợi dây.

Chương 5
MÔ PHỎNG BẰNG SIMULINK
POWER SYSTEM BLOCKSET VÀ GUI

§1. MÔ PHỎNG BẰNG SIMULINK

1.1. Khởi động Simulink

Khởi động Simulink bằng cách lick vào icon của Simulink trên Matlab toolbar hay đánh lệnh Simulink trong cửa sổ Matlab. Lúc này trên màn hình xuất hiện cửa sổ Simulink Library Browser, trong đó có các thư viện các khối của Simulink.

1.2. Tạo một mô hình mới

Để tạo một mô hình mới, hãy click vào icon trên cửa sổ Simulink Library Browser hay chọn menu File → New → Model trên cửa sổ Matlab.

1.3. Thay đổi một mô hình đã có

Click vào icon trên cửa sổ Simulink Library Browser hay chọn Open trên cửa sổ Matlab. File chứa mô hình sẽ mở và ta có thể thay đổi các thông số cũng như bản thân mô hình.

1.4. Chọn một đối tượng

Để chọn một đối tượng, click lên nó. Khi đó đối tượng sẽ có một hình chữ nhật có các góc là các hạt bao quanh.

1.5. Chọn nhiều đối tượng

Có thể chọn nhiều đối tượng cùng một lúc bằng cách dùng phím Shift và nháy chuột hay vẽ một đường bao quanh các đối tượng đó bằng cách bấm chuột kéo thành hình chữ nhật và thả khi hình chữ nhật đó đã bao lấy các đối tượng cần chọn.

1.6. Chọn tất cả các đối tượng

Để chọn tất cả các đối tượng trong cửa sổ ta chọn menu Edit → Select All.

1.7. Các khối

Khối các phần tử mà Simulink dùng để tạo mô hình. Có thể mô hình hoá bất kì một hệ thống động học nào bằng cách tạo mối liên hệ giữa các khối theo cách thích hợp. Khi tạo một mô hình cần thấy rằng các khối của Simulink có hai loại cơ bản:

– Các khối không nhìn thấy được đóng vai trò quan trọng trong việc mô phỏng một hệ thống. Nếu thêm hay loại bỏ một khối không nhìn thấy được thì thuộc tính của mô hình sẽ thay đổi.

– Các khối nhìn thấy được không đóng vai trò quan trọng trong mô hình hóa, chúng chỉ giúp ta xây dựng mô hình một cách trực quan bằng đồ hoạ.

Một vài khối của Simulink có thể thấy được trong một số trường hợp và lại không thấy được trong một số trường hợp khác. Các khối như vậy được gọi là các khối nhìn thấy có điều kiện.

1.8. Copy các khối từ một cửa sổ sang một cửa sổ khác

Khi xây dựng một mô hình, thường phải copy các khối từ thư viện khối của Simulink sang cửa sổ mô hình. Để làm việc này cần thực hiện các bước sau:

– Mở cửa sổ thư viện khối,

– Kéo khối muốn dùng từ cửa sổ thư viện vào cửa sổ mô hình và thả.

Có thể copy các khối bằng cách dùng lệnh Copy và Paste trong menu Edit qua các bước sau:

– Chọn khối muốn copy

– Chọn Copy từ menu Edit

– Làm cho cửa sổ cần copy hoạt động

– Chọn Paste từ menu Edit

Simulink gán một tên cho mỗi bản copy. Nếu là khối đầu tiên trong mô hình thì tên của nó giống như trong thư viện Simulink. Nếu là bản thứ 2 hay thứ 3 thì sau nó sẽ có chỉ số 1 hay 2, v.v... Trên cửa sổ mô hình có lưới, để hiển thị lưới này từ cửa sổ Matlab hãy gõ vào:

set–param('<model name>','showgrid','on')

Để thay đổi khoảng cách ô lưới ta dùng lệnh:

set–param('<model name>','gridspacing',<number of pixels>)

Ví dụ: Để thay đổi ô lưới thành 20 pixels, gõ lệnh:

set–param('<model name>','gridspacing',20)

Để nhân bản một khối giữ phím Ctrl, kéo khối tới một vị trí khác và thả.

1.9. Mô tả thông số của khối

Để mô tả thông số của khối dùng hộp thoại Block Properties. Để hiển thị hộp thoại này ta chọn khối và chọn Block Properties từ menu Edit. Có thể nhấp đúp chuột lên khối để hiển thị hộp thoại này. Hộp thoại Block Properties gồm:

– Description: Mô tả ngắn gọn về mục đích của khối,

– Priority: Thực hiện quyền ưu tiên của khối so với các khối khác trong mô hình,

– Tag: Trường văn bản được lưu cùng với khối,

– Open function: Các hàm Matlab được gọi khi mở khối này,

– Attributes format string: Thông số này sẽ mô tả thông số nào được hiển thị dưới icon của khối.

1.10. Deleting Blocks

Muốn xoá một hay nhiều khối, chọn khối đó và nhấn phím Del.

1.11. Thay đổi hướng của khối

Có thể xoay hướng của khối bằng cách vào menu Format sau đó:

– Chọn Flip Block để quay khối 180^0.

– Chọn Rotate Block để quay khối 90^0.

1.12. Định lại kích thước của khối

Để thay đổi kích thước của khối, đưa con trỏ chuột vào một góc của khối rồi bấm và kéo cho đến kích thước mong muốn và thả.

1.13. Xử lí tên khối

Mỗi khối có tên, phải là duy nhất và phải chứa ít nhất một kí tự. Mặc định tên khối nằm dưới khối. Với tên khối ta có thể thực hiện các thao tác sau đây:

– Thay đổi tên khối bằng cách bấm chuột vào tên đã có và nhập lại tên mới. Nếu muốn thay đổi font chữ dùng cho tên khối hãy chọn khối, vào menu Format và chọn Font.

– Thay đổi vị trí đặt tên khối từ dưới lên trên hay ngược lại bằng cách kéo tên khối tới vị trí mong muốn.

– Không cho hiển thị tên khối bằng cách vào menu Format và chọn Hide Names hay Show Names.

1.14. Hiển thị các thông số bên dưới khối

Simulink có thể hiển thị một hay nhiều thông số bên dưới khối. Để làm điều này ta nhập vào một dòng trong trường Attributes format string ở hộp thoại Block Properties.

1.15. Cắt các khối

Để cắt khối khỏi sơ đồ ta bấm phím Shift và kéo khối đến vị trí mới.

1.16. Nhập và xuất các vectơ

Hầu hết các khối chấp nhận đại lượng đầu vào là vectơ hay vô hướng và biến đổi thành đại lượng đầu ra là vectơ hay vô hướng.

Có thể xác định đầu vào nào nhận đại lượng vectơ bằng cách chọn mục Wide Vector Lines từ menu Format. Khi tùy chọn này được chọn, các đường nhận vectơ được vẽ đậm hơn các đường mang số liệu vô hướng.

Nếu thay đổi mô hình sau khi chọn Wide Vector Lines thì phải cập nhật hình vẽ bằng cách chọn Update Diagram từ menu Edit, khởi động lại Simulink để cập nhật sơ đồ.

1.17. Mở rộng vô hướng các đầu vào và các thông số

Mở rộng vô hướng là biến đổi đại lượng vô hướng thành vectơ với số phần tử không thay đổi. Simulink áp dụng mở rộng vô hướng cho các đại lượng vào và thông số đối với hầu hết các khối.

Mở rộng đầu vào: Khi dùng khối với nhiều đầu vào có thể trộn lẫn các đại lượng vectơ và đại lượng vô hướng. Lúc này các đầu vào vô hướng được mở rộng thành vectơ với số phần tử như của đầu vào vectơ, các phần tử đều có trị số như nhau.

Mở rộng thông số: Có thể đặt các thông số đối với khối được vectơ hoá thành đại lượng vectơ hay đại lượng vô hướng. Khi đặt các thông số vectơ, mỗi phần tử thông số được kết hợp với phần tử tương ứng trong vectơ đầu vào, Simulink áp dụng mở rộng vô hướng để biến đổi chúng thành vectơ có kích thước phù hợp.

1.18. Gán độ ưu tiên cho khối

Có thể gán độ ưu tiên cho khối không nhìn thấy trong mô hình. Khối có độ ưu tiên cao hơn được đánh giá trước khối có độ ưu tiên nhỏ hơn. Độ ưu tiên được gán bằng cách dùng lệnh tương tác hay dùng chương trình. Để dùng chương trình ta sử dụng lệnh:

set-param(b,'Priority','n')

trong đó b là khối và n là một số nguyên, số càng thấp, độ ưu tiên càng cao.

Để gán độ ưu tiên bằng lệnh ta nhập độ ưu tiên vào trường Priority trong hộp thoại Block Priorities của khối.

1.19. Sử dụng Drop Shadows

Có thể thêm Drop Shadow vào khối đã chọn bằng cách chọn Show Drop Shadow từ menu Format.

1.20. Tạo một thư viện

Để tạo một thư viện, chọn Library từ menu con New của menu File. Simulink sẽ hiển thị một cửa sổ mới, có tên là Library: untitled.

1.21. Mở một thư viện đã có

Khi mở một thư viện, nó tự động khoá lại và không thể thay đổi các thành phần của thư viện được.

Muốn mở khoá ta chọn Unlock từ menu Edit.

1.22. Copy một khối từ thư viện vào mô hình

Có thể copy một khối từ thư viện vào mô hình bằng copy hay paste, hay kéo và thả vào cửa sổ mô hình.

1.23. Vẽ đường nối giữa các khối

Để nối cổng ra của một khối với cổng vào của một khối khác ta thực hiện như sau:

– Đặt con trỏ chuột lên cổng ra của khối đầu tiên, con trỏ có dạng dấu +,

– Nhấn và giữ chuột,

– Kéo con trỏ chuột tới cổng vào của khối thứ hai,

– Thả chuột.

Để vẽ đường gấp khúc, nhấn phím Shift khi vẽ.

1.24. Vẽ đường nhánh

Đường nhánh là đường nối từ một đường đã có và mang tín hiệu của nó tới cổng vào của một khối. Để thêm đường nhánh ta thực hiện như sau:

– Đưa con trỏ chuột tới đường cần phân nhánh,

– Nhấn phím chuột đồng thời nhấn phím Ctrl,

– Kéo con trỏ chuột tới cổng vào tiếp theo đồng thời thả chuột và phím Ctrl.

Tuy nhiên, có thể dùng phím phải chuột thay vì dùng phím Ctrl và phím trái chuột.

1.25. Chèn khối vào một đường

Có thể chèn một khối vào một đường bằng cách kéo và thả khối đó lên đường nối. Khối chèn vào chỉ có một đầu vào và một đầu ra.

1.26. Nhãn của tín hiệu

Nhãn của tín hiệu để ghi chú cho mô hình. Nhãn có thể nằm trên hay dưới đường nối nằm ngang, bên phải hay bên trái đường nối thẳng đứng.

– Để tạo nhãn tín hiệu, bấm đúp chuột lên đường nối và ghi nhãn.

– Để di chuyển nhãn, sửa một nhãn, click lên nhãn rồi đánh nhãn mới sau khi xóa nhãn cũ.

1.27. Ghi chú

Ghi chú là đoạn văn bản cung cấp thông tin về mô hình. Có thể thêm ghi chú vào bất kì chỗ nào của mô hình.

– Để tạo một ghi chú, nhấn đúp chuột vào vùng trống của mô hình, trên màn hình xuất hiện một hình chữ nhật có con nháy ở trong và có thể đánh văn bản ghi chú vào khung này.

– Khi muốn di chuyển phần ghi chú đến một vị trí khác, bấm chuột vào đó và kéo đến vị trí mới rồi thả chuột.

– Để sửa một ghi chú, bấm chuột vào nó để hiển thị khung văn bản và bắt đầu sửa.

1.28. Các kiểu dữ liệu

Simulink chấp nhận các kiểu dữ liệu sau:

Double số thực với độ chính xác gấp đôi

single số thực với độ chính xác đơn

int8 số nguyên có dấu 8 bit

uint8 số nguyên không dấu 8 bit

int16 số nguyên có dấu 16 bit

uint16 số nguyên không dấu 16 bit

int32 số nguyên có dấu 32 bit

uint32 số nguyên không dấu 32 bit

Các khối đều chấp nhận kiểu dữ liệu double.

Khi nhập vào tham số của một khối, kiểu dữ liệu của nó được mô tả bằng lệnh: type(value), với type là tên của kiểu dữ liệu và value là giá trị của tham số.

Ví dụ 1: Single(1.0) dữ liệu là số thực có trị là 1

int8(2) dữ liệu là số nguyên có trị là 2

int32(3+2i) dữ liệu là số phức, phần thực và phần ảo là số nguyên 32 bit

Tạo tín hiệu có kiểu dữ liệu được mô tả: Có thể đem vào mô hình một tín hiệu có kiểu dữ liệu được mô tả bằng một trong các phương pháp sau đây:

– Nạp tín hiệu có kiểu dữ liệu mong muốn từ Matlab.

– Tạo một khối hằng và đặt thông số của nó có kiểu dữ liệu mong muốn.

– Sử dụng khối biến đổi kiểu dữ liệu.

Hiển thị các kiểu dữ liệu của cổng: Để hiển thị kiểu dữ liệu của cổng trong mô hình, chọn Port Data Types từ menu Format.

1.29. Tín hiệu phức

Mặc định các giá trị của tín hiệu Simulink là số thực. Tuy nhiên các mô hình có thể tạo và xử lí các tín hiệu là số phức. Có thể đưa một tín hiệu là số phức vào mô hình bằng một trong các phương pháp sau:

– Nạp tín hiệu phức từ Matlab.

– Tạo một khối hằng trong mô hình và cho nó giá trị phức.

– Tạo một tín hiệu thực tương ứng với phần thực và phần ảo của tín hiệu phức và kết hợp các phần này thành tín hiệu phức bằng cách sử dụng khối biến đổi tín hiệu thực - ảo thành tín hiệu phức.

Có thể xử lí tín hiệu phức nhờ các khối chấp nhận tín hiệu phức. Phần lớn các khối của Simulink chấp nhận tín hiệu vào là số phức.

1.30. Tạo hệ thống con

Tạo một hệ thống con bằng cách thêm khối hệ thống con: Để tạo một khối hệ thống con trước khi thêm các khối trong nó ta phải thêm khối hệ thống con vào mô hình rồi thêm các khối tạo nên hệ thống con này vào khối hệ thống con bằng cách sau:

– Copy khối hệ thống con từ thư viện Signal & System vào mô hình.

– Mở khối hệ thống con bằng cách click đúp lên nó.

– Trong cửa sổ khối con rỗng, tạo hệ thống con. Sử dụng các khối inport để biểu diễn đầu vào và các khối outport để biểu diễn đầu ra.

Tạo hệ thống con bằng cách nhóm các khối đã có: Nếu mô hình đã có một số khối mà ta muốn nhóm thành khối hệ thống con thì có thể nhóm các khối này thành khối hệ thống con bằng cách bao các khối và đường nối giữa chúng bằng một đường đứt nét (bấm chuột và kéo từ góc này đến góc kia của các khối) rồi thả chuột, chọn Create Subsystem từ menu Edit.

Gán nhãn cho các cổng của hệ thống con: Simulink gán nhãn cho các cổng của hệ thống con. Nhãn là tên của các khối inport và outport nối khối hệ thống con với các khối bên ngoài qua các cổng này. Có thể dấu các nhãn này bằng cách chọn khối hệ thống con rồi chọn Hide Port Labels từ menu Format.

Có thể đấu một hay nhiều nhãn bằng cách chọn các khối inport hay outport thích hợp trong khối hệ thống con và chọn Hide Name từ menu Format.

1.31. Một số ví dụ mô phỏng

1.31.1. Mô phỏng một phương trình

Phương trình dùng để biến đổi độ Celcius thành độ Fahrenheit là :

$$T^0F = (9/5)T^0C + 32$$

Trước hết khảo sát các khối cần để tạo mô hình:

– Khối ramp trong thư viện Sources để input tín hiệu nhiệt độ,

– Khối Constant trong thư viện Sources để tạo hằng số 32,

– Khối Gain trong thư viện Math để tạo ra hệ số 9/5,

– Khối Sum trong thư viện Math để cộng hai đại lượng,

– Khối Scope trong thư viện Sinks để hiển thị kết quả.

Tiếp đó đưa các khối vào cửa sổ mô hình, gán các giá trị thông số cho Gain và Constant bằng cách nhấp đúp lên chúng để mở khối.

Sau đó nối các khối.

– Khối Ramp đưa nhiệt độ Celcius và mô hình. Mở khối này và thay đổi giá trị khởi gán Initial output về 0.

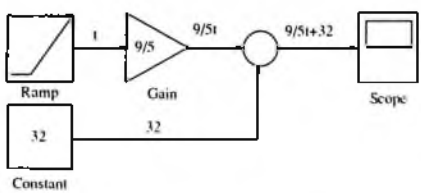
– Khối Gain nhân nhiệt độ này với hệ số 9/5.

– Khối Sum cộng giá trị 32 với kết quả và đưa ra nhiệt độ Fahrenheit.

– Khối Scope để xem kết quả.

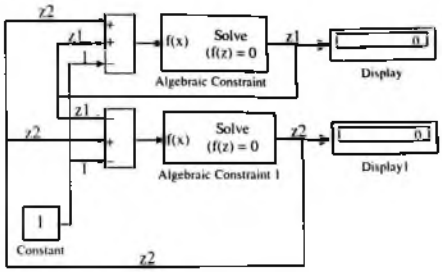
Sơ đồ mô phỏng như hình 5.1. Bây giờ Start từ menu Simulation để chạy simulation.

Simulation chạy 10 giây, tương ứng với nhiệt độ Celcius biến đổi từ 0 đến 10^0.



Hình 5.1. Sơ đồ mô phỏng biến đổi thang đo nhiệt độ

1.31.2. Mô phỏng giải một hệ phương trình tuyến tính



Hình 5.2. Sơ đồ mô phỏng giải một hệ hai phương trình tuyến tính hai ẩn

Xét hệ phương trình tuyến tính có hai ẩn:

$$\begin{cases} z_1 + z_2 = 1 \\ -z_1 + z_2 = 1 \end{cases}$$

Để mô phỏng ta dùng các khối:

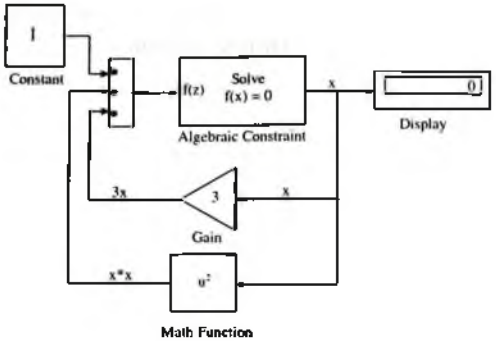
- Hai khối Algebric Constraint trong thư viện Math để giải phương trình,
- Hai khối Sum trong thư viện Math để tạo phép tính,
- Hai khối Display trong thư viện Sink để hiển thị giá trị nghiệm,
- Khối Constant trong thư viện Sources để tạo giá trị 1.

1.31.3. Mô phỏng giải một phương trình bậc cao

Xét phương trình: $x^2 + 3x + 1 = 0$.

Để mô phỏng ta dùng các khối:

- Khối Algebric Constraint trong thư viện Math để giải phương trình,
- Khối Display trong thư viện Sink để hiển thị trị số của nghiệm,
- Khối Constant trong thư viện Sources để tạo giá trị 1,
- Khối Sum trong thư viện Math để tạo phép cộng,
- Khối Math Function trong thư viện Math để tạo hàm x^2,
- Khối Gain trong thư viện Math để tạo hệ số 3.



Hình 5.3. Sơ đồ mô phỏng giải một phương trình bậc hai

1.31.4. Mô phỏng hệ thống liên tục đơn giản

Ta mô hình hoá hệ mô tả bởi phương trình vi phân

$x'(t) = -2x(t) + u(t)$

với u(t) là một sóng hình chữ nhật có biên độ bằng 1 và tần số 1 rad/s.

Để mô phỏng hệ ta dùng các khối:

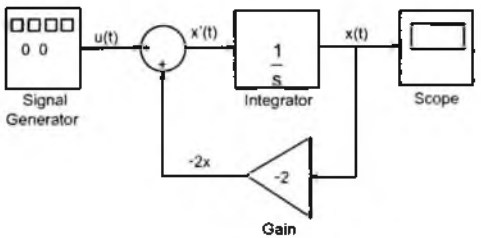
– Khối Gain trong thư viện Math để tạo hệ số 2,

– Khối Sum trong thư viện Math để tạo phép tính,

– Khối Scope trong thư viện Sink để xem kết quả,

– Khối Signal Generator trong thư viện Sources để tạo nguồn,

– Khối Integrator trong thư viện Continuous để tích phân.



Hình 5.4. Sơ đồ mô phỏng một hệ thống liên tục

1.31.5. Mô phỏng giải hệ phương trình vi phân bậc cao

Xét hệ mô tả bởi phương trình vi phân bậc hai sau:

$$\frac{d^2x}{dt^2} + 3\frac{dx}{dt} + 2x(t) = 4u(t),$$

trong đó u(t) là hàm bước nhảy, x'(0) = 0 và x(0) = 0. Biến đổi Laplace của hệ cho ta:

$$p^2x(p) + 3p.x(p) + 2x(p) = 4u(p).$$

Hàm truyền của hệ là:

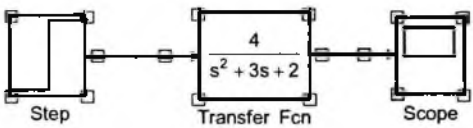
$$T(p) = \frac{4}{p^2 + 3p + 2}.$$

Ta mô phỏng hệ bằng các phần tử:

– Khối Step trong thư viện Sources để tạo hàm bước nhảy u(t),

– Khối Transfer Fcn trong thư viện Continuous để tạo hàm truyền,

– Khối Scope trong thư viện Sink để xem kết quả.



Hình 5.5. Sơ đồ mô phỏng giải một phương trình vi phân

1.32. Lưu mô hình

Có thể lưu mô hình bằng cách chọn Save hay Save as từ menu File. Dùng Save khi mở mô hình cũ, sửa và lưu lại còn Save as dùng khi mô hình có tên là untitled nghĩa là chưa được đặt tên. Simulink sẽ lưu mô hình bằng một file có tên với phần mở rộng là .mdl.

1.33. In sơ đồ khối

Có thể in sơ đồ khối bằng cách chọn Print từ menu File, lúc này hộp thoại Print sẽ xuất hiện. Nó cho phép:

– In hệ thống hiện hành,

– In hệ thống hiện hành và các hệ thống dưới nó trong phân lớp mô hình,

– In hệ thống hiện hành và các hệ thống trên nó trong phân lớp mô hình,

– In tất cả các hệ thống trong mô hình,

– In mỗi mô hình một khung overlay.

1.34. Duyệt qua mô hình

Cửa sổ Model Browser cho phép:

– Duyệt qua mô hình có phân lớp,

– Mở các hệ thống trong các mô hình,

– Xác định nội dung các khối trong một mô hình,

Để hiển thị Model Browser, chọn nó từ menu View. Cửa sổ xuất hiện được chia làm 2 phần. Phía trái là Browser. Cấu trúc cây của mô hình hiển thị ở bên phải. Mỗi dấu + tương ứng với một hệ thống con.

§2. MÔ PHỎNG BẰNG POWER SYSTEM BLOCKSET

2.1. Khái niệm chung

Power System Blockset được thiết kế nhằm cung cấp một công cụ hiệu quả và tiện lợi để mô phỏng nhanh và dễ dàng các mạch điện, các hệ thống điện.

Thư viện của nó chứa các phần tử cơ bản của mạch điện như máy biến áp, đường dây, các máy điện và các thiết bị điện tử công suất.

Giao diện đồ hoạ cung cấp các thành phần của hệ thống điện. Các thành phần này được lưu trong thư viện powerlib. Để mở thư viện này từ cửa sổ Matlab ta đánh lệnh powerlib.

Lúc này Matlab mở một cửa sổ chứa các khối hệ thống con khác nhau. Các hệ thống con bao gồm:

– Electrical Sources,

– Elements,

– Power Electronics,

– Machines,

– Connectors,

– Measuremets,

– Extras,

– Demos.

Có thể mở các hệ thống con này để tạo ra các cửa sổ chứa các khối cần copy vào mô hình. Mỗi thành phần được biểu diễn bằng một icon đặc biệt.

2.2. Mô hình hoá một mạch điện đơn giản

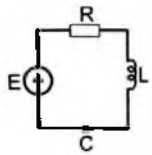
Power System Blockset cho phép xây dựng và mô phỏng một mạch điện chứa các phần tử tuyến tính cũng như phi tuyến. Xét mạch điện như hình vẽ:

$e = \sqrt{2}.220.\sin(100\pi t + \varphi_0)(V)$,

$R = 10\Omega$,

$L = 0.1$ H,

$C = 100\mu F$.



Để mô phỏng mạch điện này ta dùng các khối: nguồn, điện trở, điện kháng, điện dung và dụng cụ đo.

Để đo điện áp cần dùng khối Vmet sẽ thu được trị số tức thời của điện áp.

Để thấy được giá trị hiệu dụng ta dùng khối RMS.

Các bước thực hiện như sau:

– Từ menu File của cửa sổ powerlib chọn New rồi chọn Model sẽ chứa mạch điện và gọi là ctcircuit.mdl.

– Mở thư viện Electrical Sources để copy AC Voltage Source Block vào cửa sổ ctcircuit.mdl.

– Mở hộp thoại AC Voltage Source Block bằng cách nhấp đúp lên nó để nhập vào biên độ, phase và tần số theo các giá trị đã cho trong sơ đồ.

Chú ý: + Biên độ là giá trị max của điện áp.

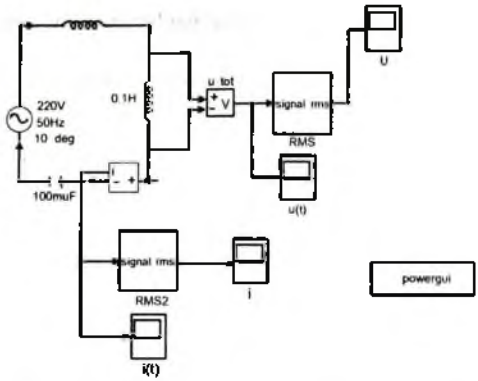
+ Do khối điện trở không có nên copy khối Series RLC Branch và đặt giá trị điện trở như đã cho và đặt L là vô cùng và C là zero. Thực hiện tương tự với phần tử L và C.

– Lấy khối đo điện áp trong hệ thống con Measurement.

– Để xem điện áp, dùng khối Scope của Simulink chuẩn.

– Mở Simulink và copy khối Scope vào mô hình ctcircuit.mdl. Nếu khối Scope được nối trực tiếp với đầu ra của thiết bị đo điện áp nó sẽ hiển thị điện áp theo V.

Để hoàn thành mạch điện, cần nối các phần tử với nhau. Sơ đồ mô phỏng (lưu trong ctcircuit.mdl) như sau:



Hình 5.12. Sơ đồ mô phỏng hệ RLC mắc nối tiếp

Tiếp theo có thể bắt đầu mô phỏng từ menu simulation. Vào menu này, chọn các thông số cho quá trình mô phỏng và bấm nút start.

Để dễ dàng cho việc phân tích trạng thái xác lập của mạch điện, thư viện powerlib cung cấp giao diện đồ hoạ (GUI). Copy khối giao diện Powergui vào cửa sổ ctcircuit.mdl và nhấn đúp vào icon để mở nó.

Mỗi dụng cụ đo đại lượng được xác định bằng chuỗi tương ứng với tên của nó.

Các biến trạng thái được hiển thị tương ứng với các giá trị xác lập của dòng điện và điện áp. Tên các biến chứa tên các khối, bắt đầu bằng tiếp đầu ngữ Il hay Uc. Dấu quy ước được sử dụng với dòng điện, điện áp và các biến trạng thái được xác định bằng hướng của các khối.

Dòng điện cảm chạy theo hướng mũi tên tương ứng với dấu dương điện áp trên tụ C bằng điện áp ra trừ đi điện áp vào.

– Chọn menu Tool → Steady – State Voltages and Currents để xem các trị số xác lập của dòng điện và điện áp.

– Tiếp theo chọn menu Tool → Initial Value of State Variables để hiển thị các giá trị khởi đầu của các biến trạng thái. Các giá trị khởi đầu này được đặt để bắt đầu simulation ở trạng thái xác lập.

– Tiếp theo tính các biểu diễn của không gian trạng thái của mô hình ctcircuit bằng hàm power2sys. Nhập dòng lệnh sau đây vào cửa sổ Matlab

[A, B, C, D, x0, states, inputs, outputs] = power2sys('ctcircuit');

Hàm power2sys trả về mô hình không gian trạng thái của mạch trong 4 ma trận A, B, C, D, x0 là vectơ các điều kiện đầu vừa hiển thị với Powergui. Tên của

các biến trạng thái, các đại lượng vào và các đại lượng ra được trả về trong 3 ma trận chuỗi.

Một khi mô hình trạng thái đã biết, có thể phân tích được trong vùng tần số.

Ví dụ 1: Các mode của mạch này có thể tìm từ các giá trị riêng của ma trận A (dùng lệnh Matlab eig(A)):

```
eig(A)
ans =
1.0e+002 * -0.5000 + 3.1225i -0.5000 - 3.1225i
```

Hệ thống này có dao động tắt dần vì phần thực âm. Nếu dùng Control System Toolbox, có thể vẽ đồ thị Bode.

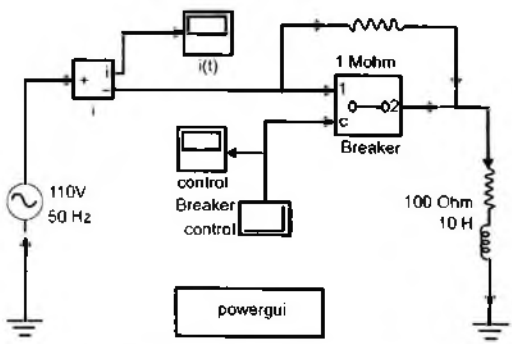
Các lệnh Matlab như sau:

```
freq = 0:1500;
w = 2*pi*freq;
[bien, pha, w] = bode(A, B, C, D);
semilogy(w, mag1(:, 2));
semilogy(w, mag1(:, 2));
```

2.3. Mô hình hoá quá trình quá độ

Một trong những ứng dụng của Power System Blockset là simulation quá trình quá độ trong các mạch điện. Điều này có thể làm được cả với cầu dao cơ khí và mạch điện tử.

Xét quá trình quá độ khi đóng một mạch RL vào nguồn điện xoay chiều. Sơ đồ mô phỏng như sau:

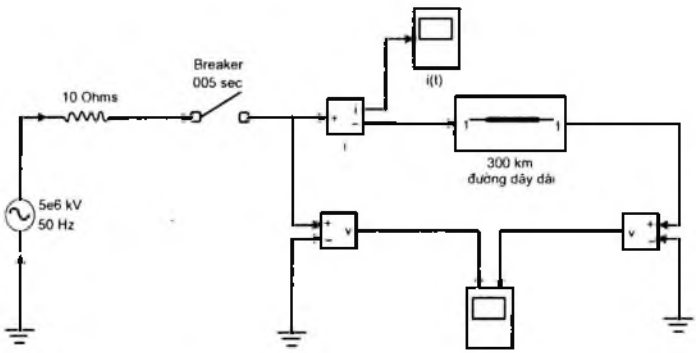


Hình 5.13. Sơ đồ mô phỏng một quá trình quá độ

Trước quá trình quá độ, cầu dao (được mô phỏng bằng phần tử breaker) ở trạng thái mở. Sau khoảng thời gian 1.5 chu kì, cầu dao đóng, nối mạch RL vào nguồn e = 2 sin314t.

2.4. Mô hình hoá đường dây dài

Đường dây dài là đường dây có thông số rải, được mô phỏng bằng khối Distributed Parameter Line và được xây dựng trên cơ sở xét quá trình truyền sóng trên đường dây. Xét một đường dây dài 1000km có mô hình như sau:



Hình 5.14. Sơ đồ mô phỏng một đường dây dài

Khi sử dụng mô hình phải khai báo điện trở, điện dung và điện cảm của đường dây trên một đơn vị dài, số pha và chiều dài của đường dây.

2.5. Mô hình hoá đường dây bằng các đoạn hình π

Mục đích của mô hình này là thực hiện đường dây 1 pha với thông số được tập trung trên từng đoạn.

Khối PI Section Line thực hiện đường dây truyền tải một pha với thông số tập trung trên từng đoạn π.

Đối với đường dây truyền tải, điện trở, điện cảm và điện dung phân bố đều trên suốt chiều dài.

Một mô hình xấp xỉ đường dây thông số phân bố có được bằng cách nối nhiều đoạn π giống nhau. Không giống như đường dây thông số rải có số trạng thái là vô hạn, mô hình tuyến tính các đoạn π có số hữu hạn các trạng thái cho phép mô hình không gian - trạng thái được dùng để rút ra đáp ứng tần số. Số

đoạn được dùng phụ thuộc vào tần số được biểu diễn. Xấp xỉ tốt nhất thực hiện theo phương trình:

$$f_{max} = \frac{Nv}{8l}$$

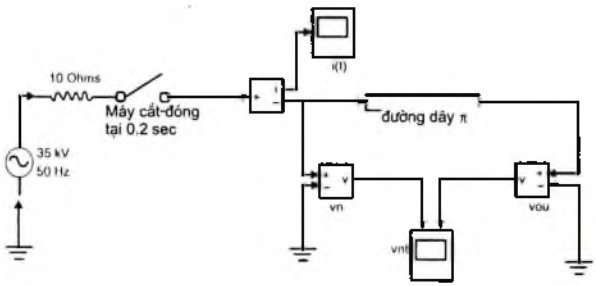
trong đó: N là số đoạn π,

v là tốc độ truyền sóng (km/s = $\frac{1}{\sqrt{LC}}$(H/km)C(F/km),

l là chiều dài đường dây (km).

Ta xét đường dây trên không dài 100km có tốc độ truyền sóng 300000km/s, tần số lớn nhất biểu diễn được khi dùng một đoạn π là 375Hz. Mô hình đơn giản này đủ dùng trong hệ thống truyền tải năng lượng.

Xây dựng mô hình như sau:



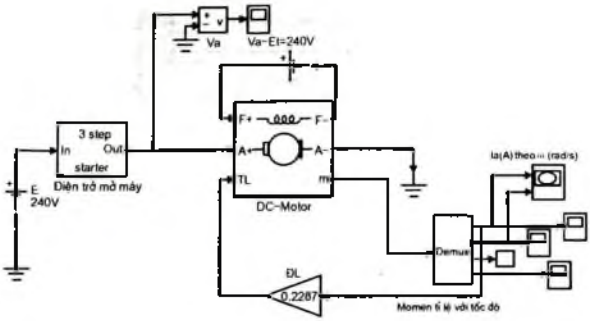
Hình 5.15. Mô hình đường dây bằng các đoạn π

Nhập điện trở, điện cảm và điện dung trên một đơn vị dài vào 3 ô đầu tiên của hộp thoại. Nhập độ dài và số đoạn π mong muốn vào 2 ô cuối.

2.6. Mô hình hoá máy điện

Các máy điện nằm trong thư viện Machines, được mô phỏng dựa trên các phương trình cơ bản của nó và được chia thành 2 dạng: Máy điện trong hệ đơn vị tương đối và máy điện trong hệ đơn vị SI.

Xét quá trình mở máy bằng điện trở một động cơ điện một chiều như sau:

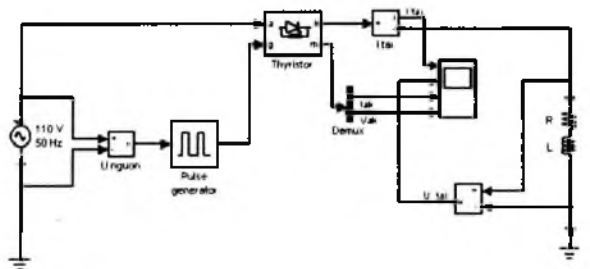


Hình 5.16. Sơ đồ mô phỏng một động cơ điện một chiều

2.7. Giới thiệu về điện tử công suất

Power System Blockset được thiết kế để simulation các thiết bị điện tử công suất.

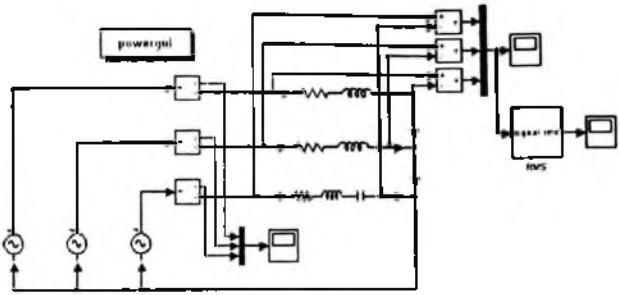
Chúng ta có thể khảo sát một mạch điện có thyristor cung cấp cho một mạch RL.



Hình 5.17. Sơ đồ mô phỏng một mạch điện có thyristor cung cấp cho một mạch RL

2.8. Mô hình hoá mạch điện 3 pha

Mô hình hoá một mạch điện 3 pha có nguồn đối xứng nhưng tải không đối xứng. Điện áp các nguồn có trị hiệu dụng là 231V. Tải pha thứ nhất là R = 1Ω, L = 1H, pha thứ hai R = 15Ω, L = 2H và pha thứ 3 là R = 10Ω, L = 1H và C = 1μF. Sơ đồ mô phỏng như sau:

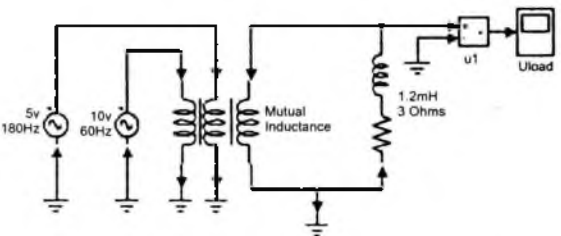


Hình 5.18. Sơ đồ mô phỏng một mạch điện ba pha

2.9. Mô hình điện kháng hỗ cảm

Phần tử điện kháng hỗ cảm thực hiện mối liên hệ từ giữa 2 hay 3 dây quấn.

Khối Mutual Inductance thực hiện liên hệ từ giữa 3 dây quấn riêng biệt. Ta mô tả điện trở và điện cảm của từng dây quấn trên vào mục thứ nhất của hộp thoại và điện trở, điện cảm hỗ cảm vào mục cuối cùng.



Hình 5.19. Sơ đồ mô phỏng một mô hình điện kháng hỗ cảm

+ Nếu mục vào của dây quấn thứ 3 bị bỏ trống, nghĩa là chỉ có hỗ cảm giữa 2 dây quấn.

+ Các đầu vào của khối Mutual Inductance là cùng cực tính tại một thời điểm.

+ Do simulation nên cần: $Rs > 0$, $Rs > Rm$, $Lm \neq 0$, $Ls \neq Lm$

+ Điện trở của dây quấn phải dương và lớn hơn điện trở hỗ cảm.

+ Điện cảm hỗ cảm phải khác 0 nhưng điện trở hỗ cảm có thể bằng 0.

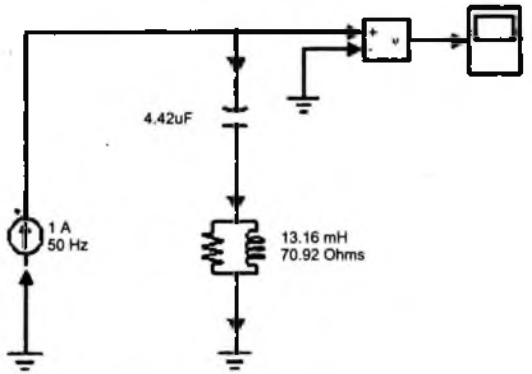
+ Dây quấn có thể thả nổi, nghĩa là không nối với tổng trở hay phần còn lại của mạch.

2.10. Mô hình nhánh RLC nối song song

Xét một nhánh RLC nối song song.

– Khối Parallel RLC Branch thực hiện điện trở, điện cảm và điện dung nối song song. Để bỏ một phần tử R, L hay C ta phải đặt các thông số tương ứng là Inf, Inf và 0.

– Ta có thể dùng giá trị âm cho các thông số. Để có đáp ứng tần của bộ lọc tần số bậc 7 ở 660Hz ta dùng mạch như trong file ctpararlc.mdl.



Hình 5.20. Sơ đồ mô phỏng một nhánh RLC nối song song

Tổng trở của mạch:

$$Z(s) = \frac{V(s)}{I(s)} = \frac{RLCs^2 + Ls + R}{LCs^2 + RCs}$$

Để đáp ứng tần của tổng trở ta phải xác định mô hình không gian - trạng thái (ma trận A B C D) của hệ thống.

```
[A, B, C, D] = power2sys('ctpararlc');
freq = logspace(1, 4, 500);
w = 2*pi*freq;
[Z,phaseZ] = bode(A, B, C, D, 1, w);
subplot(2, 1, 1)
loglog(freq, Z)
grid
```

```
title('Bo loc song hai bac 11')
xlabel('Tan so, Hz')
ylabel('Tong tro Z')
subplot(2, 1, 2)
semilogx(freq, phaseZ)
xlabel('Tan so, Hz')
ylabel('Pha Z')
grid
```

2.11. Mô hình tải RLC nối song song

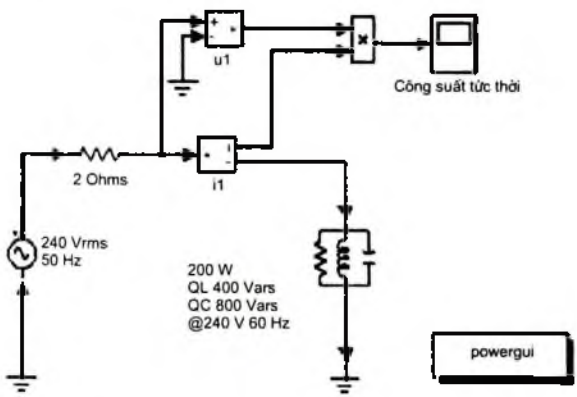
Xét một tải RLC nối song song. Khối Parallel RLC Load thực hiện tải tuyến tính như tổ hợp nối song song của các phần tử R, L và C.

– Để xác định tham số ta nhập điện áp định mức và tần số định mức vào 2 mục đầu tiên.

– Nhập công suất tác dụng, công suất phản kháng trên cuộn dây và công suất phản kháng trên tụ điện vào 3 mục cuối. Các công suất phản kháng phải dương.

Tại tần số đã mô tả, tải sẽ có tổng trở và công suất tỉ lệ với bình phương điện áp đặt vào.

Tìm các giá trị xác lập của điện áp và dòng điện tải trong mạch trong file ctloadrclp.mdl.



Hình 5.21. Sơ đồ mô phỏng một tải RLC nối song song

2.12. Mô hình nhánh RLC mắc nối tiếp

Xét một RLC mắc nối tiếp. Khối Series RLC Branch bao gồm điện trở, điện cảm và điện dung mắc nối tiếp.

Để loại trừ R, L hay C ta cho chúng bằng 0, 0 hay Inf. Các giá trị này có thể đặt là số âm. Xét một mô hình như trong file ctserierlc.mdl.

Tổng trở của nhánh là:

$$Z(s) = \frac{V(s)}{I(s)} = \frac{LCs^2 + RCs + 1}{Cs}$$

Để nhận được đáp ứng tần số của tổng trở, phải xây dựng mô hình không gian - trạng thái của hệ thống:

```
[A,B,C,D] = power2sys('ctserierlc');
freq = logspace(1, 4, 500);
w = 2*pi*freq;
[Y, phaseY] = bode(A, B, C, D, 1, w);
Z = 1./Y;
phaseZ = phaseY;
subplot(2, 1, 1)
loglog(freq, Z)
grid
title('Bo loc song bac 5')
xlabel('Tan so, Hz')
ylabel('Tong tro  Z')
subplot(2,1,2)
semilogx(freq,phaseZ)
xlabel('Tan so, Hz')
ylabel('Pha  Z')
grid
```

2.13. Mô hình tải RLC mắc nối tiếp

Xét một tải RLC mắc nối tiếp tuyến tính.

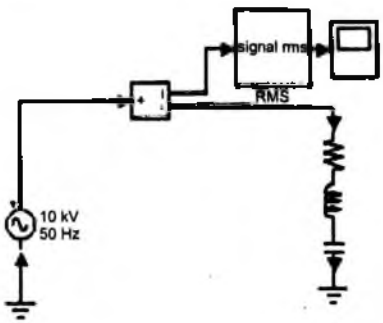
Khối Series RLC Load thực hiện tải RLC mắc nối tiếp tuyến tính.

– Nhập giá trị điện áp và tần số định mức vào 2 ô đầu của hộp thoại. Nhập

công suất tác dụng, công suất phản kháng trên điện cảm và công suất tác dụng trên điện dung vào 3 ô cuối.

– Các công suất phản kháng phải có trị số dương. Tại tần số đã mô tả, tải có tổng trở xác định và công suất của nó tỉ lệ với bình phương điện áp đặt vào.

Ta tìm giá trị xác lập của điện áp và dòng điện của tải trong file ctloadrlcs.mdl.



Hình 5.22. Sơ đồ mô phỏng một tải RLC nối tiếp

§3. GUI TRONG MATLAB

3.1. Giao diện phần mềm

3.1.1. Mở phần mềm

Gõ lệnh sau vào cửa sổ Command để mở phần mềm

>> guide

Trong cửa sổ GUIDE Quick Start có nhiều lựa chọn theo một trong các khuôn mẫu sau:

– *Create New GUI*: Tạo một hộp thoại GUI mới theo một trong các loại sau.

– *Blank GUI (Default)*: Hộp thoại GUI trống, không có một điều khiển uicontrol nào cả.

– *GUI with Uicontrols*: Hộp thoại GUI với một vài uicontrol như button,...

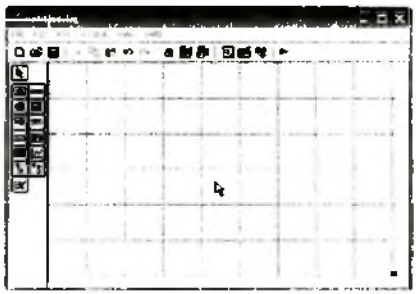
– *GUI with Axes and Menu*: Hộp thoại GUI với một uicontrol axes và button các menu để hiển thị đồ thị.

– *Modal Question Dialog*: Hộp thoại đặt câu hỏi Yes, No.

– *Open Existing GUI*: Mở một project có sẵn.

Hình 5.23. Giao diện GUI Quick Start trong Matlab

3.1.2. Giao diện phần mềm



Hình 5.24. Giao diện GUI trong Matlab

Giao diện rất giống với các chương trình lập trình như Visual Basic, Visual C++,... khi di chuột qua các biểu tượng ở bên trái sẽ thấy tên của các điều khiển.

3.1.3. Một số nút điều khiển hay dùng

– Push Button: giống như nút Command Button trong VB, đây là các nút bấm giống nút OK, Cancel.

– Slider: là thanh trượt có một con trượt chạy trên đó.

– Radio Button: là nút nhỏ hình tròn để chọn lựa

+ Check Box,

+ Edit Text,

+ Static Text,

+ Pop-up Menu,

+ List Box,

+ Axes,

+ Panel,

+ Button Group,

+ ActiveX Control,

+ Toggle Button.

– Ngoài ra menu quan trọng nhất là *menu Tools:* Run (Ctr + T): nhấn vào để chạy chương trình đã viết, khi có lỗi là hiện ra ngay.

– Align Object: dùng để làm cho các điều khiển sắp xếp gọn đẹp, như cùng căn lề bên trái,...

– Grid and Rulers: dùng để cấu hình về lưới trong giao diện vì nó sẽ coi giao diện như một ma trận các ô vuông nhỏ, ta sẽ thay đổi giá trị này để cho các điều khiển có thể thả ở đâu tùy ý cho đẹp.

– Menu Editor: để tạo menu cho điều khiển.

– Tab Order Editor: dùng để sắp xếp Tab order là thứ tự khi nhấn phím Tab.

– Gui Options: giúp lựa chọn cho giao diện GUI.

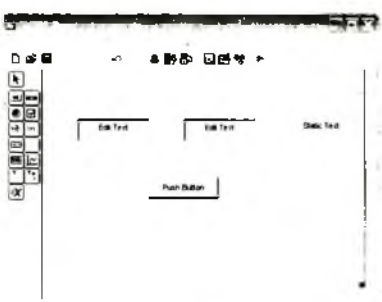
Đầu tiên nên vào menu Help để xem hướng dẫn thêm trong Help. Ví dụ khi save dưới tên: dung1, khi đó đồng thời xuất hiện cửa sổ Editor và đang mở file dung1.m. Trong thư mục save sẽ có 2 file là:

+ dung1.fig: File này chứa giao diện của chương trình,

+ dung1.m: File chứa các mã thực thi cho chương trình như các hàm khởi tạo, các hàm callback,...

3.1.4. Kéo thả các điều khiển

Hãy kéo vào trong giao diện 2 edit box, 1 static box và 1 Push Button.

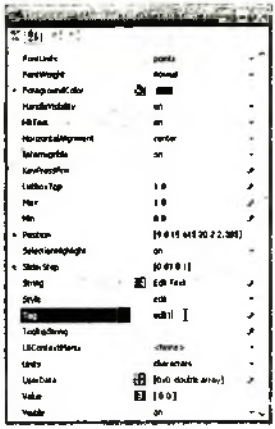


Hình 5.25. Bảng điều khiển

Chương trình có chức năng khi nhấn vào nút bấm thì kết quả của phép tính cộng giữa hai số được gõ vào hai ô sẽ hiện lên trong Static Text.

3.1.5. Thay đổi các thuộc tính của các điều khiển

Click đúp vào Edit Text bên trái để xuất hiện cửa sổ các thuộc tính của điều khiển. Có thể sắp xếp theo chức năng hoặc theo thứ tự A-Z của tên thuộc tính bằng cách gõ nút hiện bên trái.



Hình 5.26. Thay đổi thuộc tính bằng điều khiển

Thuộc tính quan trọng của *Edit Box* bao gồm:

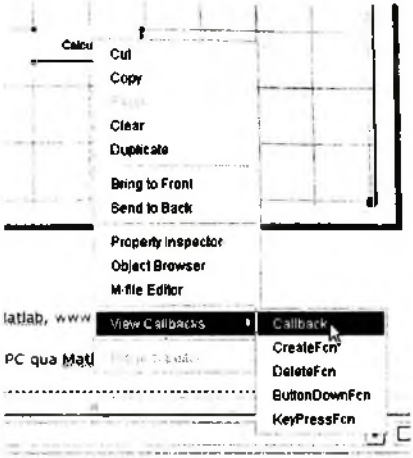
– Tag: Đây là thuộc tính giống như Caption trong Visual Basic để đặt tên điều khiển. Dùng tên này có thể thao tác đến các thuộc tính của đối tượng, đặt tên là: *editStr1*.

– String: Là xâu kí tự hiện lên *Edit Box*. Tương tự, khi thay đổi thuộc tính tag của Edit Box thứ 2 thành *editStr2*. *Static Box* cũng tương tự thành staticStr3.

– Push Button: thuộc tính tag = buttonCalculate, string = calculate.

3.1.6. Viết lệnh cho chương trình

Chương trình có tác dụng khi nhấn vào nút Push Button sẽ hiện lên kết quả ở Static Box. Vì thế sẽ phải viết vào hàm mà khi nhấn vào Push Button sẽ gọi, đó chính là hàm Callback. Điều khiển nào cũng có hàm callback, giống hàm ngắt trong vi điều khiển. Click chuột phải vào nút Calculate chọn Callback.



Hình 5.27. Mã lệnh cho chương trình điều khiển

Nhìn vào định nghĩa của hàm trong Editor

```
% --- Executes on button press in buttonCalculate.
function buttonCalculate_Callback(hObject, eventdata, handles)
% hObject    handle to buttonCalculate (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

Hàm này được thực hiện khi nhấn vào nút buttonCalculate. Hàm có một số tham số:

– hObject: handle của điều khiển button Calculate eventdata.

– handles: Là một cấu trúc chứa tất cả các điều khiển và dữ liệu người dùng. Dùng tham số này để truy xuất các điều khiển khác. Qua thuộc tính tag của các điều khiển ta sẽ truy xuất đến thuộc tính string của các điều khiển editStr1, editStr2, editStr3 bằng lệnh *get* và *set*.

+ get(handles.tag–dieu–khien, 'ten thuoc tinh'),

+ set(handles.tag–dieu–khien, 'ten–thuoc–tinh', gia–tri),

và hàm quan trọng nữa biến từ string sang số là hàm: str2num và num2str để biến trở lại.

Ví dụ 1:

```
% --- Executes on button press in buttonCalculate.
function buttonCalculate_Callback(hObject, eventdata, handles)
% hObject    handle to buttonCalculate (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
    val1 = get(handles.editStr1,'String');
    val2 = get(handles.editStr2,'String');

    val1 = str2num(val1);
    val2 = str2num(val2);

    val3 = val1 + val2;

    set(handles.staticStr3, 'String',num2str(val3));
```

Nhấn nút Run kiểm tra kết quả:

3.1.7. Tính chất và hàm trong GUI

Để hiện cửa sổ các tính chất *Property Inspector* của một điều khiển có 3 cách sau:

– Nhấn đúp chuột vào mỗi điều khiển.

– Chọn điều khiển rồi vào menu View, chọn *Property Inspector*.

– Chọn điều khiển rồi nhấn vào biểu tượng *Property Inspector*, gần chỗ M-file editor. Khi đó, cửa sổ *Property Inspector* sẽ hiện ra. Nhấn vào điều khiển nào thì cửa sổ này sẽ hiện thông tin tương ứng cho điều khiển đó.

Một vài tính chất chung của các điều khiển nên chú ý:

Tính chất (property)	Giá trị (value)	Miêu tả
Enable	on, inactive, off. Mặc định là: on	Xác định khi nào thì điều khiển hiển thị lên giao diện. Đặt = off, thì điều khiển sẽ không xuất hiện.

Tính chất (property)	Giá trị (value)	Miêu tả
Max	Vô hướng. Mặc định là 1	Giá trị lớn nhất, tùy thuộc vào từng điều khiển.
Min	Vô hướng. Mặc định là 0	Giá trị nhỏ nhất, tùy thuộc vào từng điều khiển.
Position	Vector gồm 4 phần tử (left, bottom, width, height)	Kích thước của điều khiển và vị trí tương đối của nó với điều khiển chứa nó.
String		
Units		Đơn vị đo lường dùng trong xác định vị trí
Value	Vô hướng hoặc vectơ	Giá trị của component, tùy thuộc vào từng component

Ngoài ra, thuộc tính quan trọng phải thay ngay từ đầu là: *tag*. Thuộc tính này giống thuộc tính Caption gặp trong Visual Basic, chính là tên để phân biệt giữa các điều khiển. Mỗi điều khiển chỉ có một tên duy nhất và nên quy định tên này cho dễ nhớ.

Ví dụ 1: Một ô nhập dữ liệu giá trị tiền, thuộc Edit Box đặt tag là: editMoney.

3.1.7.1. Push Button

Thay đổi chữ hiển thị (label) hiển thị trên điều khiển bằng cách thay đổi thuộc tính *String*. Chữ hiển thị trên Button chỉ có thể là 1 dòng, nên nếu gõ nhiều dòng trong thuộc tính String thì chỉ hiển thị dòng đầu tiên. Nếu số kí tự dòng đầu tiên lớn hơn bề rộng có thể hiển thị chữ được của bề mặt Button thì Matlab tự rút ngắn String đó với dấu 3 chấm (...).

Thay đổi vị trí của PushButton = thay đổi thuộc tính Position. Có thể code trong MFile (thay đổi trong quá trình thực thi) hoặc thay đổi ngay lúc thiết kế (gắp thả).

Để thêm một ảnh vào PushButton ta gán thuộc tính CData bằng một ma trận $m \times n \times 3$ của giá trị RGB. Thực hiện trong MFile ở hàm Open của điều khiển để ngay khi chạy chương trình thì ảnh này đã được load.

```
img = rand(16,64,3);
set(handles.pushbutton1, 'CData',img);
```

Chú ý: Có thể tạo biểu tượng riêng cho các nut Push Button bằng cách dùng Icon Editor, sau đó dùng hàm ind2rgb để chuyển sang ảnh gán vào thuộc tính CData.

Các sự kiện xảy ra khi nhấn Push Button được viết trong các hàm ngắt như Callback,..

3.1.7.2. Slider (Thanh trượt)

Có thể thay đổi khoảng giá trị của Slider bằng cách thay đổi thuộc tính Min và Max. Phải chú ý để Min < Max.

Giá trị hiện tại của Slider được cho bằng giá trị của thuộc tính *Value* nên khi set hoặc get thì ta lấy giá trị này.

Khi click vào 2 cái mũi tên 2 bên thì thanh trượt sẽ trượt tương ứng về 2 phía theo một bước nào đó. Thay đổi thuộc tính bằng cú pháp

SliderStep = [min–step, max–step]: min–step là giá trị bước nhảy khi click vào 2 mũi tên, còn Max–step là giá trị khi click vào trong vùng trượt.

3.1.7.3. Radio Button

Để biết nút Radio có được đánh dấu hay không phải xem thuộc tính Value của nó.

– Value = 1 thì đang đánh dấu,

– Value = 0 thì không đánh dấu.

3.1.7.4. Check Box

Check Box có thuộc tính quan trọng tương tự như Radio Button.

3.1.7.5. Edit Text

Thuộc tính quan trọng là String, chính là xâu kí tự hiển thị trên Edit Text. Để hiển thị Edit Text dạng Multi-line cần thay đổi thuộc tính Max, Min. Max > Min.

Ví dụ: Max = 2, Min = 0 và hiển thị 2 dòng đó.

3.1.7.5. Tổng quan về hàm Callback

Sau khi tạo giao diện xong tiếp tục lập trình các hành vi của các điều khiển để đáp ứng lại các sự kiện như nhấn phím, kéo thanh trượt, khi chọn menu,... đó chính là các hàm Callback.

Thế nào là hàm Callback? Callback là một hàm miêu tả hành vi của một thành phần GUI xác định hoặc là của chính GUI figure, điều khiển các hành vi của

chúng bằng cách thực hiện một số hành động được viết trong hàm, để đáp ứng lại một sự kiện của chính thành phần đó.

Cách lập trình này thường gọi là: Lập trình lái sự kiện (event driven programming).

Ví dụ 1:

Khi nhấn một Button thì vẽ đồ thị đúng không? Như vậy khi nhấn phím thì hiển nhiên đã gọi hàm Callback nhấn phím của Button đó và trong hàm Callback này đã có lệnh vẽ đồ thị.

Các loại hàm Callback: Mỗi thành phần có nhiều hàm Callback khác nhau. Bảng dưới đây liệt kê các loại hàm Callback và các điều khiển có thể có hàm này.

Callback property	Sự kiện xảy ra	Thành phần có hàm này
ButtonDownFcn	Thực hiện khi người dùng nhấn chuột lên hoặc trong 5 pixels của component hoặc figure. Nếu là component thì thuộc tính Enable phải on.	Axes, figure, button group, panel, user interfacecontrols
Callback	Hành động của các component, ví dụ như thực thi khi người dùng click lên Push Button hoặc chọn một thành phần menu.	Contextmenu, menu, userinterface controls
CloseRequestFcn	Thực thi trước khi figure đóng.	Figure
CreateFcn	Tạo các thành phần, được dùng để khởi tạo các thành phần khi nó được tạo ra. Nó thực thi sau khi thành phần hoặc figure được tạo, nhưng trước khi hiển thị lên trên giao diện người dùng.	Axes, figure, button group, contextmen, menu, panel, user interfacecontrols
DeleteFcn	Xóa thành phần, nó có thể được dùng để thực hiện hành động xóa bỏ trước khi component hoặc figure bị hủy bỏ.	Axes, figure, button group, contextmen, menu, panel, user interfacecontrols
KeyPressFcn	Thực thi khi người dùng nhấn một phím trong keyboard và component hoặc figure của hàm callback đó đang được focus.	Figure, userinterface controls
KeyReleaseFcn	Thực thi khi người dùng nhả một phím đang bấm và figure vẫn đang được focus.	Figure

Callback property	Sự kiện xảy ra	Thành phần có hàm này
ResizeFcn	Thực thi khi người dùng thay đổi kích thước của panel, button group hoặc figure với điều kiện thuộc tính *Resize* của figure = on.	Buttongroup, figure, panel
SelectiononChangeFcn	Thực thi khi người dùng lựa chọn một nút Radio Button khác hoặc toggle button khác trong thành phần Button Group.	Buttongroup
WindowButtonDownFcn	Thực thi khi bạn nhấn chuột (trái hoặc phải) trong khi con trỏ vẫn nằm trong vùng cửa sổ figure.	Figure
WindowButtonMotionFcn	Thực thi khi bạn di chuyển con trỏ trong vùng cửa sổ figure.	Figure
WindowButtonUpFcn	Ban đầu bạn nhấn chuột (trái hoặc phải) thì khi nhả phím đó ra hàm này sẽ được gọi.	Figure
WindowScrollWheelFcn	Thực thi khi nút cuộn của chuột cuộn trong khi figure vẫn trong tầm focus.	Figure

Ví dụ 2: Lập trình giao tiếp RS232 qua Matlab

Đối tượng Serial Object

Trước hết đấu tắt 2 chân 2 và 3 (TX và RX) của cổng COM lại. Sau đó viết một chương trình đơn giản nhằm giới thiệu cách tạo đối tượng, kết nối, viết hàm callback.

* Tạo đối tượng:

```
>> s = serial('COM1')
Serial Port Object: Serial-COM1
Communication Settings
Port:           COM1
BaudRate:       9600
Terminator:     'LF'
Communication State
Status:         closed
RecordStatus:   off
```

Read/Write State
TransferStatus: idle
BytesAvailable: 0
ValuesReceived: 0
ValuesSent: 0

Như vậy đối tượng là Serial-COM1, tốc độ 9600,..

Tiếp theo, xem các tham số của đối tượng như thế nào bằng lệnh get(s):

```
>> get(s)
ByteOrder = littleEndian
BytesAvailable = 0
BytesAvailableFcn =
BytesAvailableFcnCount = 48
BytesAvailableFcnMode = terminator
BytesToOutput = 0
ErrorFcn =
InputBufferSize = 512
Name =
Serial-COM1
ObjectVisibility = on
OutputBufferSize = 512
OutputEmptyFcn =
RecordDetail = compact
RecordMode = overwrite
RecordName = record.txt
RecordStatus = off
Status = closed
Tag =
Timeout = 10
TimerFcn =
TimerPeriod = 1
TransferStatus = idle
Type = serial
UserData = []
```

ValuesReceived = 0

ValuesSent = 0

SERIAL specific properties:

BaudRate = 9600

BreakInterruptFcn =

DataBits = 8

DataTerminalReady = on

FlowControl = none

Parity = none

PinStatus = [1x1 struct]

PinStatusFcn =

Port = COM1

ReadAsyncMode = continuous

RequestToSend = on

StopBits = 1

Terminator = LF

Có rất nhiều tham số nhưng ở đây ta chỉ quan tâm đến tham số: BytesAvailableFcn, tham số này chưa thiết lập và chính là hàm callback mà nó sẽ gọi khi có byte nhận được ở bộ đệm nhận. Vậy khi viết hàm này chính là viết hàm OnComm đáp ứng sự kiện ReceiveEvent như trong MSCOMM của MS. Thiết lập này phải thực hiện trước khi mở cổng để giao tiếp, vì vậy ta sẽ viết hàm callback trước.

Viết một m-file với tên Serial–Callback.m như sau:

function Serial–Callback(obj,event)

ind = fscanf(obj)

Cú pháp của hàm callback như trên với obj là đối tượng kiểu Serial. Hàm có tác dụng đọc dữ liệu và hiển thị luôn kết quả lên command window.

Đưa tham số tên hàm vào đối tượng s:

>> s.BytesAvailableFcn = @Serial–Callback;

Tiếp theo, bắt đầu giao tiếp:

>>fopen(s);

>>fprintf(s,"chao cac ban");

Sau đó kiểm tra kết quả và thử truyền các kí tự khác bằng lệnh fprintf(s,...), hoặc thử với vi xử lí, khi truyền lên.

Khi không giao tiếp thì đóng cổng lại:

>>fclose(s);

Ví dụ 3: Chương trình RS232 Communication

Đây là chương trình viết bằng GUI, đã test với mạch FPGA Spartan 3E.

Cách dùng:

1/ Chọn tham số cho Rs232 rồi ấn nút Connect để bắt đầu kết nối với RS232.

2/ Nhập dữ liệu vào ô TX rồi nhấn nút Send để gửi dữ liệu.

3/ Để thay đổi tham số (tốc độ,...) cho RS232 thì phải nhấn Disconnect trước rồi chỉnh tham số, Sau đó quay lại bước 1.

Chương trình được viết dưới dạng mở tức là có thể thêm code vào các hàm để phục vụ mục đích riêng. Đó chính là các hàm ngắt nhận, ngắt gửi,... Bytes AvailableFcnCount là số byte nhận được trong bộ đệm nhận trước khi xảy ra ngắt nhận.

Các hàm đó là:

– function ByteAvailable–Callback(obj, event)

– function OutputEmpty–Callback(obj, event)

– function Error–Callback (obj, event)

– function PinStatus–Callback(obj, event)

– function Timer–Callback (obj,event)

– function BreakInterrupt–Callback(obj, event)

Khi muốn thao tác với dữ liệu vừa nhận được thì edit thêm code trong hàm sau:

– function ByteAvailable–Callback(obj, event)

3.2. Tạo file .exe

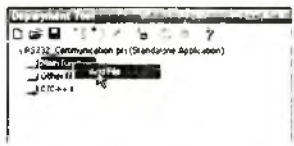
3.2.1. Tạo file .exe cho giao diện GUI Matlab

Giả sử muốn dịch một project GUI RS232 Communication thành file .exe

Bước 1: Thiết lập môi trường dịch

Bước 2: Gõ lệnh deploytool và vào menu File chọn New Project,. chọn Standalone Application, chọn tên project và nơi lưu project.

Hình 5.28. Giao diện Deployment Tool



Hình 5.29. Thêm file trong Deployment Tool

Chú ý: Deploytool là một tool matlab không tự động thiết lập khi cài đặt. Nếu muốn sử dụng thì khi cài matlab cần chọn mục Custom và tích hết các mục rồi cài đặt bình thường. Giao diện Deployment Tool sẽ gồm có Main function, Other files và C/C++ files.

Hình 5.30. Vào thư mục RS232 Communication

Hình 5.31. Tiếp tục, click chuột phải vào Other files, chọn Add files

Một project GUI sẽ gồm có 2 file (.m và .fig) và có thể thêm các file .m và .fig nếu gọi các figure khác. Như vậy file add vào trong Main Function là file .m và là file chính sẽ chạy project GUI đó.

Khi vào browse đến thư mục RS232 Communication sẽ thấy nó chỉ cho phép

chọn file .m. Ở đây chọn file RS232–Communication.m vì là file chính, còn modaldlg.m là file để mở cửa sổ yêu cầu chọn Yes, No.

Tiếp tục, click chuột phải vào Other files, chọn Add files và add các file còn lại trong project (chỉ các file .m và .fig). Khi kết thúc thì giao diện sẽ hiện ra như hình 5.32.

Hình 5.32. Giao diện hiện ra cuối cùng của Deployment Tool

Bước 3: Dịch project, vào menu Tools → Build (Ctr + B).

```
mcc -B "D:\My Documents\MATLAB\Make exe\RS232_Communication.prj"

mcc -o RS232_Communication -W main -d D:\My Documents\MATLAB\Make exe\RS232_Communi
mcc -o RS232_Communication -W main -d D:\My Documents\MATLAB\Make exe\RS232_Communi
Compiler version: 4.6 (R2007a)
Processing C:\Program Files\MATLAB\R2007a\toolbox\matlab\mcc.enc
Processing D:\My Documents\MATLAB\GUI Programming\RS232 Communication\RS232 Communi
Processing D:\My Documents\MATLAB\GUI Programming\RS232 Communication\RS232 Communi
Processing C:\Program Files\MATLAB\R2007a\toolbox\instrument\mcc.enc
Processing C:\Program Files\MATLAB\R2007a\toolbox\control\mcc.enc
Processing C:\Program Files\MATLAB\R2007a\toolbox\database\mcc.enc
Processing C:\Program Files\MATLAB\R2007a\toolbox\ident\mcc.enc
Processing C:\Program Files\MATLAB\R2007a\toolbox\shared\controllib\mcc.enc
Processing C:\Program Files\MATLAB\R2007a\toolbox\signal\mcc.enc
Processing C:\Program Files\MATLAB\R2007a\toolbox\shared\optimlib\mcc.enc
Processing include files...
2 item(s) added.
Processing directories installed with MCR...
The file D:\My Documents\MATLAB\Make exe\RS232_Communication\src\mccExcludedFiles.l
```

Tiếp tục thư vào thư mục \distrib sẽ thấy file .exe, nhấn vào chạy. Nếu muốn chuyển sang máy khác thì phải đóng gói, vào Tools chọn Package (Ctr + P):

Packaging output(2008-04-27 11:44:37)

Files to package

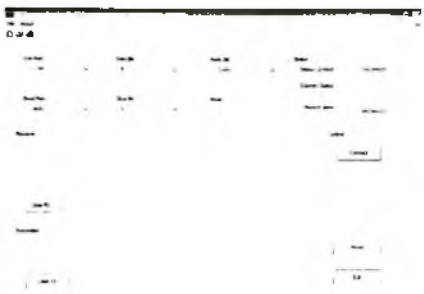
_install.bat

RS232_Communication.exe

RS232_Communication.ctf

RS232_Communication_pkg.exe created in the directory D:\My Documents\MATLAB\Make exe\RS232_Communication\distrib

The size of the package is: 5.797 MB



Hình 5.33. Giao diện hiện ra khi chạy chương trình trên máy không cài Matlab

Trường hợp khi chuyển sang máy không cài Matlab thì copy đồng thời file package và McrInstaller vào một thư mục, khi chạy sẽ hiện ra như hình 5.33.

3.2.2. Tạo file .exe trong Matlab

Chương trình viết bằng Matlab có thể dịch ra file EXE bằng công cụ gọi là Matlab Compiler (có sẵn trong bộ cài Matlab). Khi chạy chương trình đã dịch này thì không cần Matlab trong máy.

Chú ý:

– Nếu không muốn dịch ra EXE thì có thể tạo ra đối tượng COM từ chương trình Matlab với công cụ Matlab COM Builder. Sau đó có thể dùng nó trong VB, Excel và các công cụ phát triển nhanh (RAD) khác.

– Không phải lệnh nào, toolbox nào cũng có thể dịch được ra file EXE hay COM.

– Có thể viết chương trình bằng C/C++ hay Fortran, ADA, và khi cần tính toán phức tạp thì gọi thư viện tính toán của Matlab.

– Nếu lập trình bằng Java thì có thể từ Java gọi thư viện tính toán của Matlab, thậm chí control Matlab/Simulink.

– Một tính năng hay là tính năng cho phép link với Simulink, do đó có thể tạo GUI khá đẹp cho Simulink simulation mà không mất công lập trình M-file.

Sau đây là cách tạo một chương trình .exe chạy độc lập trên máy tính khác.

Bước 1: Chọn công cụ biên dịch C: (Bước này chỉ cần thực hiện 1 lần duy nhất ban đầu, nếu muốn chọn công cụ biên dịch khác thì chạy lại bước này).

Bước 2: Chạy công cụ Development Tool

>> deploytoo

Hình 5.34. Giao diện chương trình Development Tool

Hình 5.35. New Development Tool

Vào File → new Deployment Tool

Có nhiều lựa chọn, tuy nhiên để chạy độc lập nên chọn Standardalone Application.

Hình 5.36. Giao diện chương trình sau khi tạo

Sau đó vào menu Project → Add file để thêm file.

Chú ý:

– Có thể click chuột vào từng loại như Main Function, other files,... và chọn Add File.

– Phải chọn hết các file cần dịch trong dự án vào trong Project vừa tạo.

Bước 3: Dịch chương trình:

– Nếu muốn dịch thì chọn Tools → Build (Ctr + B) để dịch.

– Nếu muốn đóng gói sản phẩm thì chọn Tools → Package (Ctr + P) để đóng gói.

Trong đó: Để chạy được trên máy khác cần phải copy file MCRInstaller.exe của phiên bản Matlab đang sử dụng cho vào cùng thư mục với file đóng gói vừa tạo ra. Khi chuyển sang máy khác thì chạy file .exe, máy sẽ tự động cài MCR lần đầu, các lần sau không phải cài nữa.

Vị trí của MCRInstaller là:

>> mcrinstaller The WIN32 MCR Installer, version 7.7, is: C:\Program Files\MATLAB\R2007b\toolbox\compiler\deploy\win32\MCRInstaller.exe MCR installers for other platforms are located in: C:\Program Files\MATLAB\R2007b\ toolbox\compiler\ deploy\<ARCH> <ARCH> is the value of COMPUTER('arch') on the target machine.

Để biết version của MCR hiện tại, gõ:

>> [mcrmajor,mcrminor]=mcrversion mcrmajor = 7 mcrminor = 7.

3.3. Ứng dụng

Tạo hình nền hoặc một phần của giao diện GUI.

Ý tưởng: Dùng một Axes control để load hình ảnh. Như vậy, ngay khi chạy GUI thì ảnh đã hiển thị lên axes giống như hình nền của GUI. Vì thế, phải viết lệnh hiển thị hình ảnh vào hàm CreateFcn là hàm được gọi lên trước khi giao diện GUI hiển thị trước người dùng.

Tạo một giao diện GUI đơn giản gồm có: axes control với kích thước chính là vùng cần đặt hình nền. Các điều khiển khác đặt tại đây nên chỉ thêm vài điều khiển nhưng không viết lệnh cho các điều khiển này.

Click vào Axes control chọn hàm CreateFunction:

Khi đó chúng ta chỉ cần gõ lệnh để hiển thị hình ảnh vào hàm này:

```
function axes1_CreateFcn(hObject, eventdata, handles)
  imshow(
```

Chọn hình ở trong *C:\Documents and Settings\All Users\Documents\My Pictures\Sample Pictures*, có thể chọn đường dẫn tùy ý đến hình cần hiển thị.

Để biết thêm về lệnh hiển thị hình ảnh, gõ lệnh: help imshow. Khi chạy chương trình sẽ hiển thị như sau:

§4. THIẾT KẾ PHẦN MỀM MÔ PHỎNG TRÊN MATLAB

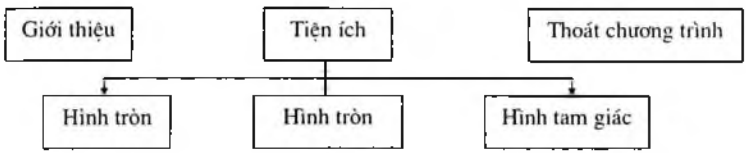
Trước khi lập trình một phần mềm thì người lập trình cần phải phân tích được cấu trúc của chương trình cần lập trình: xây dựng được sơ đồ khối, lựa chọn ngôn ngữ lập trình phù hợp và các danh mục thiết kế.

Giả sử khi cần thiết kế một phần mềm vẽ các hình: hình tròn, hình vuông và hình tam giác, chúng ta cần thực hiện các bước sau:

4.1. Phân tích cấu trúc của phần mềm cần thiết kế

4.1.1. Xây dựng sơ đồ khối

Sơ đồ khối của phần mềm có dạng như sau:



4.1.2. Lựa chọn ngôn ngữ lập trình

Giả sử ta lựa chọn lập trình trên Matlab.

4.1.3. Các danh mục thiết kế

Để lập trình một phần mềm ta cần thực hiện:

– Thiết kế giao diện phần mềm.

– Thiết kế menu.

– Thiết kế các tiện ích.

4.2. Các bước thực hiện thiết kế

4.2.1. Thiết kế giao diện phần mềm

```
clear all; % xóa hết các biến cũ đã lưu trong bộ nhớ của Matlab
%TAO GIAO DIEN
h=figure('name','Ve hinh','color',[0.5 0.5 0.8],'numbertitle','on','units',...
'normalized','position',[0.1 0.1 0.85 0.85]);
% h là một biến gán
% lệnh figure dùng để tạo giao diện.
```

color gồm 3 chỉ số tương ứng với việc trộn màu. Chỉ số 1 ứng với màu red, chỉ số 2 ứng với màu green, chỉ số 3 ứng với màu blue.

Position gồm 4 vị trí ứng với cạnh trái, cạnh trên, cạnh dưới, cạnh phải.

4.2.2. Thiết kế menu

Để thiết kế menu cần thiết kế menu chính và menu con.

4.2.2.1. Menu chính

```
%tao menu ngang
menu1=uimenu(h,'label','Gioi thieu'); % lệnh uimenu dùng để tạo ra menu ngang
menu2=uimenu(h,'label','Tien ich');
menu3=uimenu(h,'label','Thoat');
```

4.2.2.2. Menu con

Menu con là menu nằm trong menu chính (menu ngang). Theo cấu trúc trong menu tiện ích có các menu con: hình tròn, hình vuông, hình tam giác.

```
m2=uimenu(menu2,'label','Hinh tron','callback','hinh tron');
m3=uimenu(menu2,'label','Hinh vuong','callback','hinh vuong');
m4=uimenu(menu2,'label','Hinh tam giac','callback','hinh tam giac');
```

Chú ý: - Đối với menu giới thiệu: sau khi tạo menu giới thiệu ta gõ

```
menu1=uimenu(h,'label','Gioi thieu'); % lệnh uimenu dùng để tạo ra menu lệnh
m1=uimenu(menu1,'label','Ly do chon de tai','callback','gioithieu');
a=uicontrol(h,'string','GIOI THIEU PHAN MEM VE HINH',...'style','text','
FontSize',20,'units','normalized','position', [0 1 1 0.92],'backgroundcolor',[1 1 1],
'foregroundcolor',[1 1 0]);
```

– Đối với menu thoát: sau khi tạo menu thoát ta gõ

```
%tao menu exit
menu3=uimenu(h,'Label','Thoat');
```

m5=uimenu(menu3,'Label','dong lai','callback','close'); % để thoát ta chỉ cần gọi lệnh close để đóng cửa sổ giao diện.

4.2.3. Thiết kế các tiện ích

Để vẽ hình tròn, hình vuông và hình tam giác thì trước hết ta mở m-file và gõ chương trình vào, sau đó save với tên tương ứng: hinhtron.m, hinhvuong.m, hinhtamgiac.m. Trong các chương trình đó cần phải có công cụ thực hiện:

– Cửa sổ nhập lệnh

– Tạo hệ trục toạ độ

– Tạo nút vẽ hình

– Tạo nút đóng chương trình

4.2.3.1. Tạo cửa sổ nhập lệnh

```
%Tao o nhap toa do
h1=uicontrol(h,'style','edit','backgroundcolor','w','horizontalalignment',...
'left','units','normalized','position',[0.08 0.2 0.2 0.04], 'callback', 'h1=get(h1, "string")');
```

4.2.3.2. Tạo hệ trục toạ độ

```
T=axes('color','w','units','normalized',...'position',[0.1 0.13 0.75 0.6]);
```

4.2.3.3. Tạo nút vẽ hình

```
syms t y
t=0:0.5:30;
y=t.^0+(-1).^t+ sin(2*pi*50*t);
plot(t,y)
axis([0 30 0 2])
```

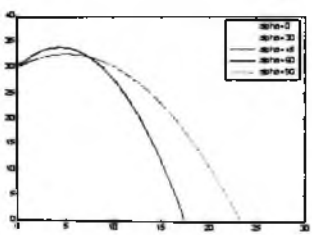
4.2.3.4. Tạo nút đóng chương trình

```
% tao nut dong
thoat=uicontrol(h, 'style', 'pushbutton', 'string', 'Thoat', 'FontSize', 14,'units',
'normalized', ... 'position', [0.8 0 0.2 0.06], 'backgroundcolor', [0.01 0.3 0.2],
'foregroundcolor', 'r', ... 'callback','close(h)');
```

BÀI TẬP

5.1. Từ một đỉnh tháp cao H = 30m người ta ném một hòn đá xuống đất với vận tốc v = 10m/s (H và v có thể thay đổi) theo phương hợp với mặt phẳng ngang một góc α. Với α = 0, 30^0, 45^0, 60^0, 90^0, hãy mô phỏng bài toán.

Kết quả mô phỏng: Ta được đồ thị như hình vẽ:



Dựa vào hình vẽ này ta thấy tầm xa cực đại khi góc ném bằng 45 độ.

5.2. Ném một vật có khối lượng 0,1kg ở độ cao 15m với vận tốc ban đầu 30m/s và góc ném 30°. Biết rằng lực ma sát tỉ lệ với vận tốc với hệ số bằng 0,15kg/s và gia tốc trường hấp dẫn bằng 9,8m/s^2.

a) Vẽ đồ thị x, y, v_x, v_y theo thời gian và quỹ đạo của vật từ khi ném đến khi chạm đất.

b) Nếu không có ma sát thì vật sẽ đi xa hơn bao nhiêu?

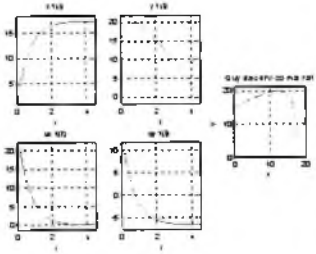
c) Vẽ quỹ đạo của vật khi có ma sát và khi không có ma sát trên cùng một hình?

d) Xác định thời điểm và vị trí khi vật đạt tới độ cao cực đại.

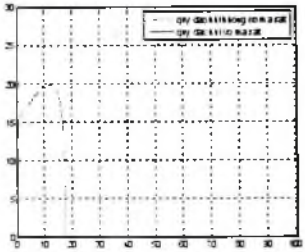
e) Xác định góc ném tối ưu để vật có thể xa nhất.

g) Vẽ hình ảnh chuyển động của vật.

Kết quả mô phỏng:



Đồ thị x, y, v_x, v_y theo thời gian và quỹ đạo của vật từ khi ném đến khi chạm đất.



Quỹ đạo của vật khi có ma sát và khi không có ma sát.

5.3. Ném một quả bóng có khối lượng 0,1kg ở độ cao 10m với vận tốc ban đầu 25m/s, góc ném nghiêng với phương nằm ngang một góc bằng 30° theo phương vuông góc với một bức tường cao thẳng đứng, cách vị trí ném 15m. Biết rằng lực ma sát tỉ lệ với vận tốc, hệ số tỉ lệ bằng 0,1 kg/s và gia tốc trường hấp dẫn bằng 9,8m/s^2, quả bóng va chạm đàn hồi với tường.

a) Xác định thời điểm khi quả bóng chạm tường?

b) Xác định độ cao của quả bóng khi chạm tường?

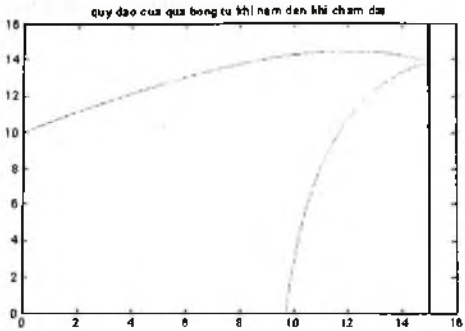
c) Góc này của quả bóng khi chạm tường?

d) Xác định thời điểm quả bóng chạm đất?

e) Khi chạm đất, quả bóng cách tường một khoảng bằng bao nhiêu?

g) Tính và vẽ quỹ đạo của quả bóng từ khi ném tới khi chạm đất?

Kết quả mô phỏng:



5.4. Xét dao động điều hoà của một con lắc lò xo với hệ số đàn hồi k = 1,5, khối lượng m = 1kg. Từ vị trí ban đầu của con lắc là x_0 = 1cm, ta thả cho con lắc dao động. Hãy:

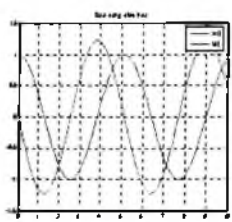
a) Biểu diễn đồ thị của toạ độ và vận tốc của con lắc theo thời gian.

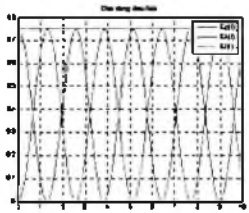
b) Biểu diễn đồ thị của động năng, thế năng và năng lượng theo thời gian.

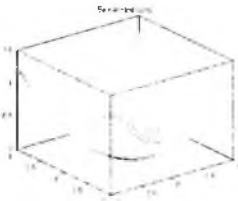
c) Biểu diễn mặt phẳng năng lượng.

d) Biểu diễn đường đẳng trị năng lượng trên mặt phẳng pha.

Kết quả mô phỏng:







5.5. Cho một hệ 100 dao động tử tuyến tính là những hạt có khối lượng bằng nhau m = 1, gắn với nhau bằng các lò xo có hệ số đàn hồi bằng nhau k = 10, khoảng cách giữa các hạt a = 1, các dao động tử chịu tác dụng của lực ma sát tỉ lệ với vận tốc của hạt với hệ số tỉ lệ γ = 0,1. Tại t = 0 các hạt đứng yên $x_j(0) = 0$, $v_j(0) = 0$. Ngoại lực tác dụng lên hạt thứ nhất bằng $F_e = b\cos(w_e t)$, với b = 0.5, $w_e = 1$.

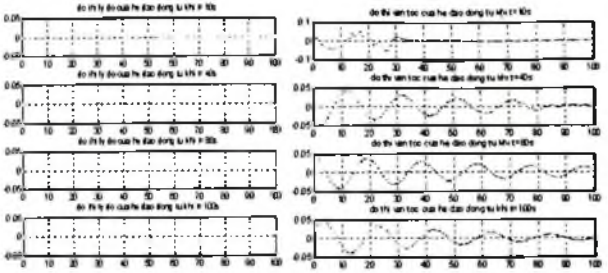
a) Tính và vẽ li độ của các dao động tử thứ 1, 45, 80, 100 theo thời gian.

b) Tính và vẽ vận tốc của các dao động tử thứ 1, 45, 80, 100 theo thời gian.

c) Tính và vẽ li độ của hệ dao động tử khi t = 10, 40, 80, 100.

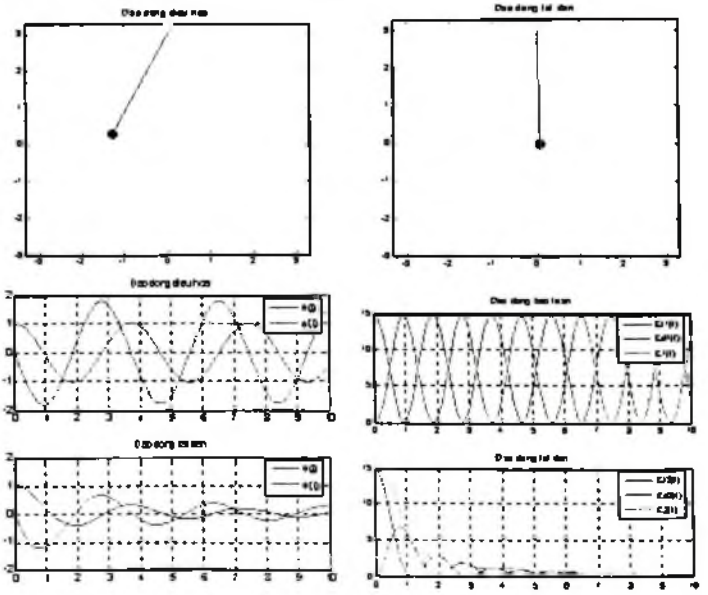
d) Tính và vẽ vận tốc của hệ dao động tử khi t = 10, 40, 80, 100.

Kết quả mô phỏng:



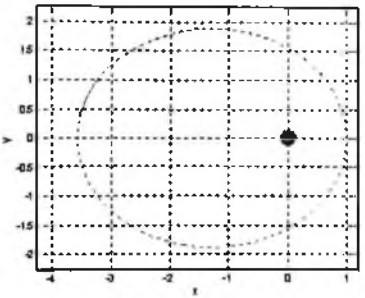
5.6. Mô phỏng con lắc có độ dài *l* dao động trong trường hấp dẫn

Kết quả mô phỏng:

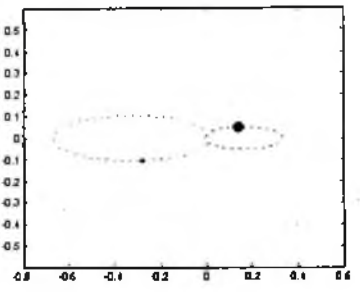


5.7. Mô phỏng chuyển động của hành tinh.

Kết quả mô phỏng:



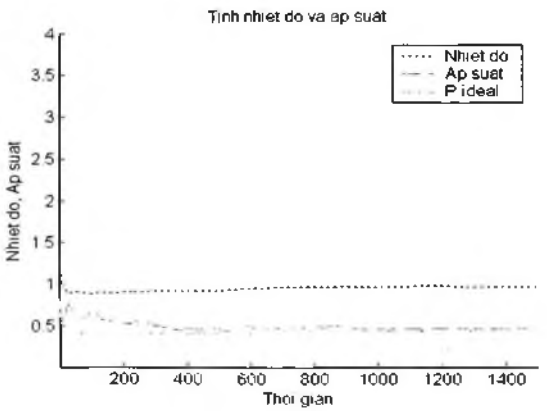
Hành tinh chuyển động dưới tác dụng của lực hấp dẫn giữa Mặt Trời và hành tinh



Hệ sao kép chỉ hấp dẫn lẫn nhau

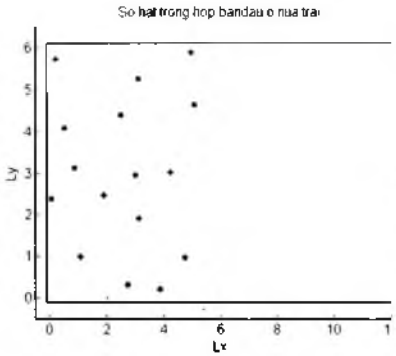
5.8. Tính toán áp suất và nhiệt độ của hệ động học phân tử và biểu diễn các đại lượng trên đồ thị theo thời gian.

Kết quả mô phỏng ta thu được:

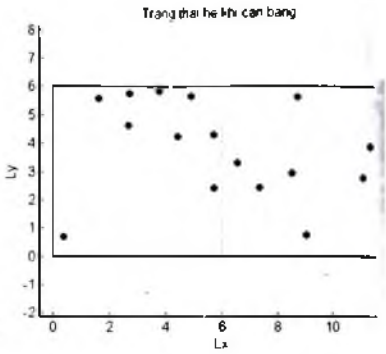


5.9. Biểu diễn số hạt trong hộp lúc ban đầu và sau khi hệ tiến tới trạng thái cân bằng. Vẽ đồ thị biểu diễn số hạt ở nửa trái của hộp theo thời gian.

Kết quả mô phỏng ta thu được:



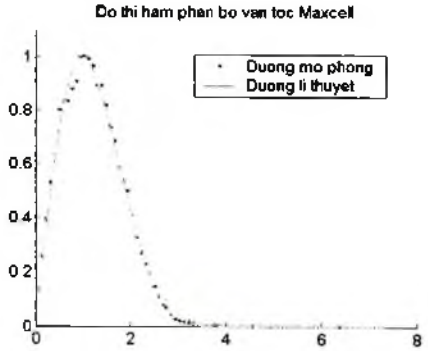
Ban đầu các hạt được phân bố ở về phía bên trái của hộp



Sau một khoảng thời gian hệ tiến tới cân bằng, số hạt ở nửa trái sẽ ổn định.

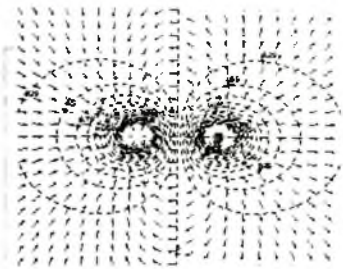
Kiểm tra tính xác suất giữa lí thuyết theo phân bố Maxcell–Boltmann:

$$\omega(v) = \sqrt{\frac{2}{\pi}}\left(\frac{m}{kT}\right)^{\frac{3}{2}} v^2 e^{\frac{-mv^2}{2kT}}$$

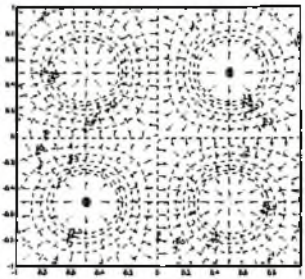


5.10. Mô phỏng đường sức của điện tích điểm.

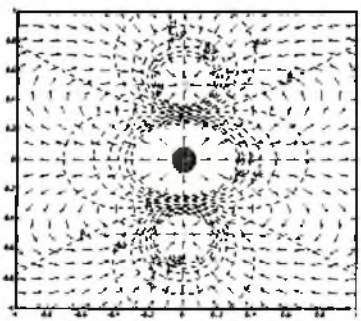
Kết quả mô phỏng ta thu được:



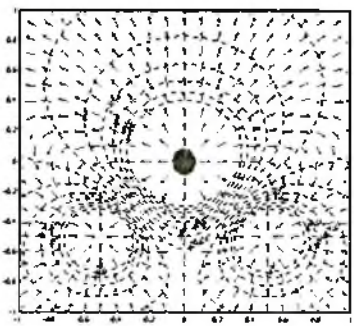
Mặt đẳng thế và điện trường của hai điện tích điểm: Điện tích âm biểu thị ở vị trí (−5,0), điện tích dương biểu thị ở vị trí (+5,0).



Hình biểu diễn hệ gồm bốn điện tích điểm đặt ở bốn đỉnh của một hình vuông. Cho hai điện tích âm −q ở toạ độ (d/2, 0, −d/2) và (−d/2, 0 ,0 d/2), còn hai điện tích dương q ở toạ độ (d/2, 0, d/2) và (−d/2, 0, −d/2).



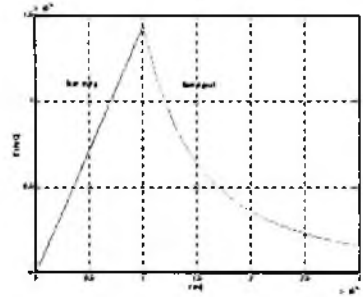
Hình biểu diễn hệ gồm ba điện tích điểm đặt trên cùng một đường thẳng. Cho điện tích thứ nhất có điện tích −q ở vị trí z = d/2 và điện tích thứ hai có điện tích −q ở vị trí z = −d/2. Điện tích thứ ba ở gốc toạ độ và có điện tích 2q.

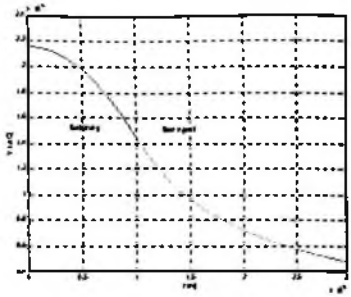


Hình biểu diễn hệ gồm 3 điện tích điểm đặt ở ba đỉnh của một tam giác. Cho điện tích thứ nhất có điện tích là −q ở toạ độ (d/2, 0, −d/2); điện tích thứ hai có điện tích là −q ở toạ độ (−d/2, 0, d/2); Điện tích thứ ba ở gốc toạ độ và có điện tích là 2q.

5.11. Xác định điện trường và điện tích tại những điểm bên trong (r < R) và bên ngoài (r > R) quả cầu. Biểu diễn bằng đồ thị kết quả đó.

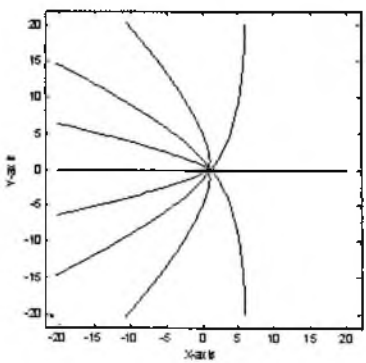
Kết quả mô phỏng ta thu được:





5.12. Mô phỏng trường điện sinh bởi một điện tích chuyển động đều.

Kết quả mô phỏng ta thu được:



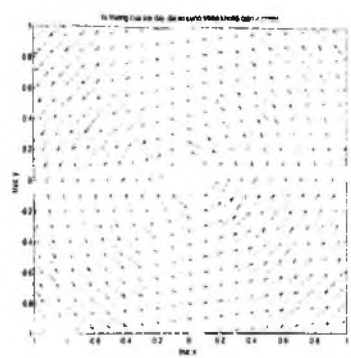
5.13. Thế từ vectơ của sợi dây thẳng dài.

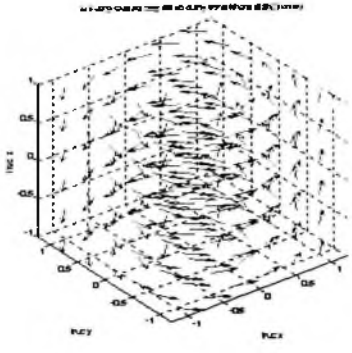
Khảo sát một sợi dây dẫn có chiều dài 2L, dòng i_0 chạy qua.

a) Xác định thế từ vectơ dọc theo đường thẳng đi qua tâm của đường dây dẫn. (Giả sử L đủ dài sao cho có thể bỏ qua hiệu ứng của các đầu dây).

b) Vẽ trường vectơ B do dây dài vô cùng trong trường hợp hai và ba chiều.

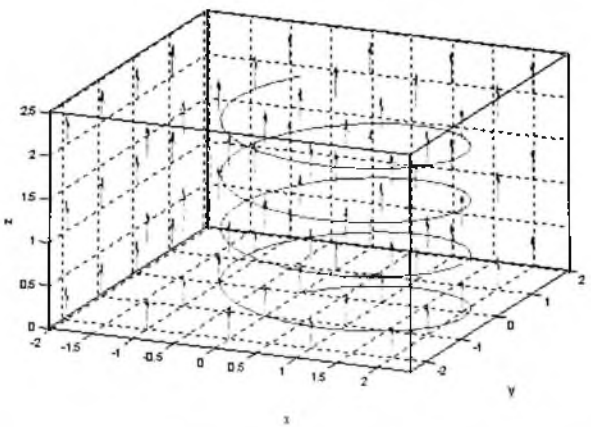
Kết quả mô phỏng ta thu được:





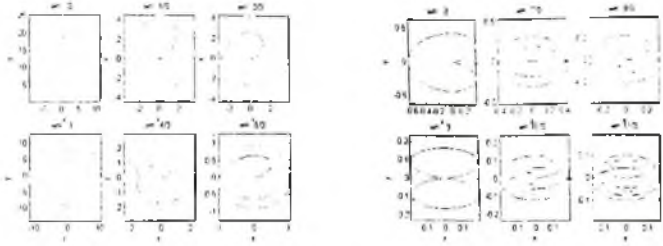
5.14. Khảo sát chuyển động của hạt tích điện trong từ trường không đổi.

Kết quả mô phỏng ta thu được:



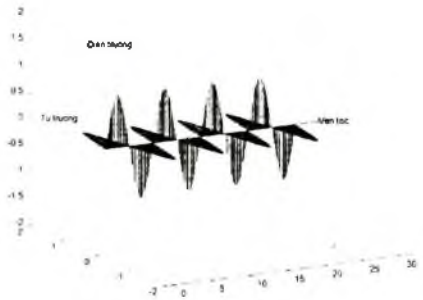
5.15. Khảo sát một hạt tích điện chuyển động trong từ trường đều không đổi và điện trường biến đổi theo thời gian. Cho điện trường hướng dọc theo trục x và có dạng $E_x = E_0\cos(\omega t)$. Từ trường B không đổi và hướng dọc theo trục z. Tại thời điểm t = 0 hạt đứng yên ở gốc toạ độ.

Kết quả mô phỏng ta thu được:



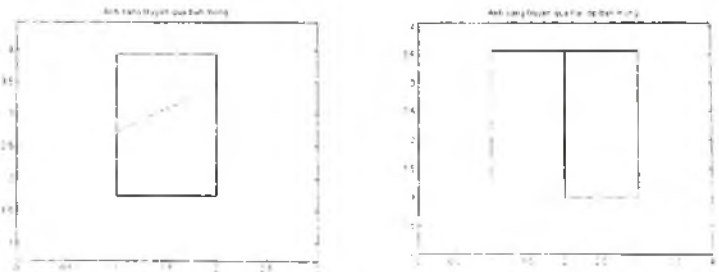
5.16. Mô phỏng sự lan truyền của sóng điện từ.

Kết quả mô phỏng ta thu được:



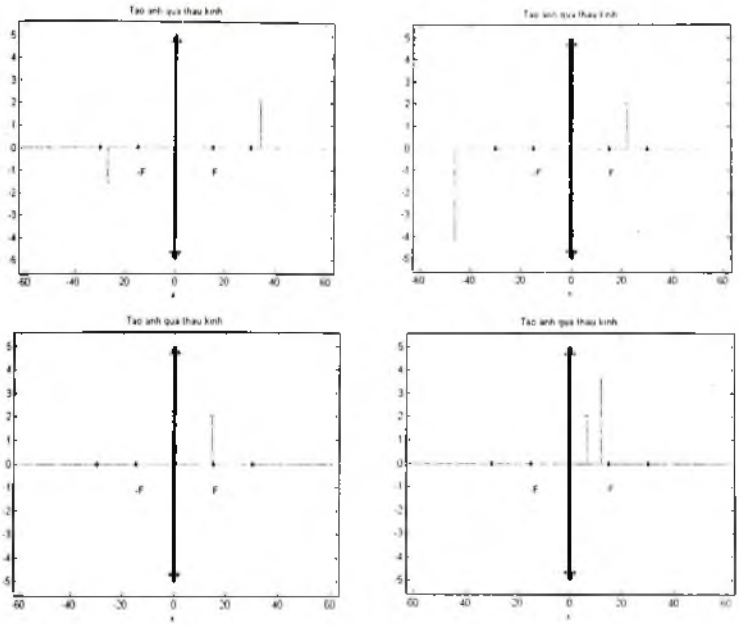
5.17. Mô phỏng sự khúc xạ qua một hoặc hai lớp bản mỏng.

Kết quả mô phỏng ta thu được:



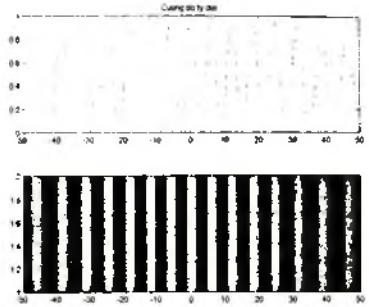
5.18. Vật sáng AB được đặt song song với màn và cách màn một khoảng cố định L. Một thấu kính hội tụ có trục chính qua điểm A và vuông góc với màn, có thể di chuyển giữa vật và màn. Mô phỏng sự tạo ảnh qua thấu kính khi vật dịch chuyển.

Kết quả mô phỏng ta thu được:

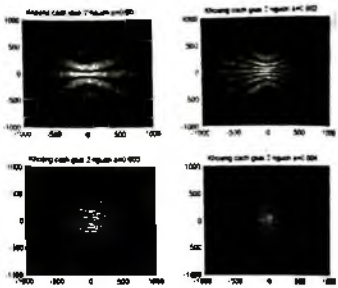


5.19. Một nguồn sáng đơn sắc phát ra ánh sáng có bước sóng λ = 0,6μm. Chiếu ánh sáng trên vào hai khe hở hẹp song song cách nhau a = 1mm và cách đều nguồn sáng. Trên một màn ảnh đặt song song và cách mặt phẳng chứa hai khe một đoạn D = 1m, ta thu được một hệ thống vân giao thoa. Mô phỏng hình ảnh giao thoa.

Kết quả mô phỏng ta thu được:



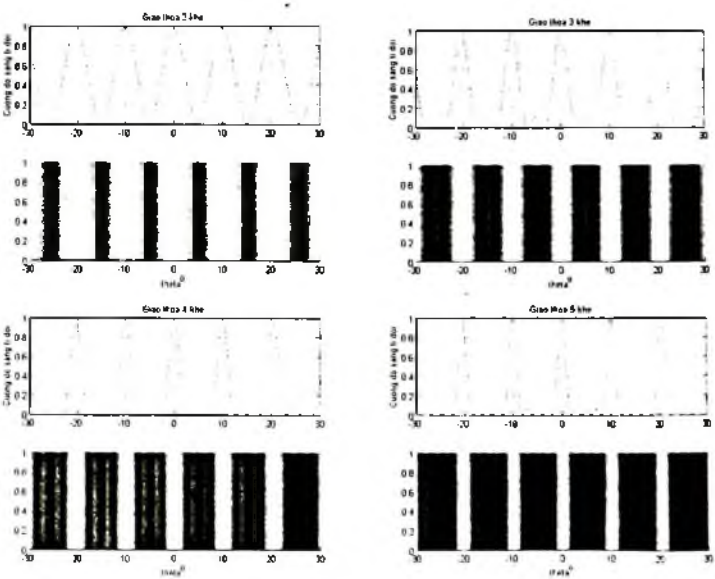
Phổ cường độ các vân giao thoa qua hai khe hở hẹp song song.



Khảo sát hệ vân khi khoảng cách a giữa hai khe thay đổi. a = 1; 2; 3; 4mm.

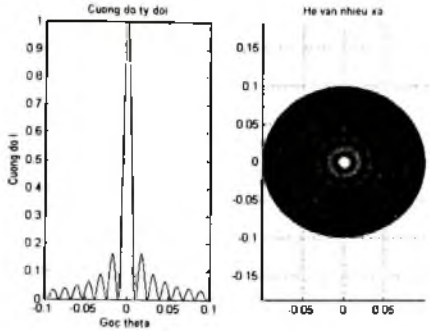
5.20. Mô phỏng hình ảnh giao thoa nếu số khe là 2; 3; 4 hoặc 5, biết khoảng cách các tia bằng 10 lần bước sóng.

Kết quả mô phỏng ta thu được:



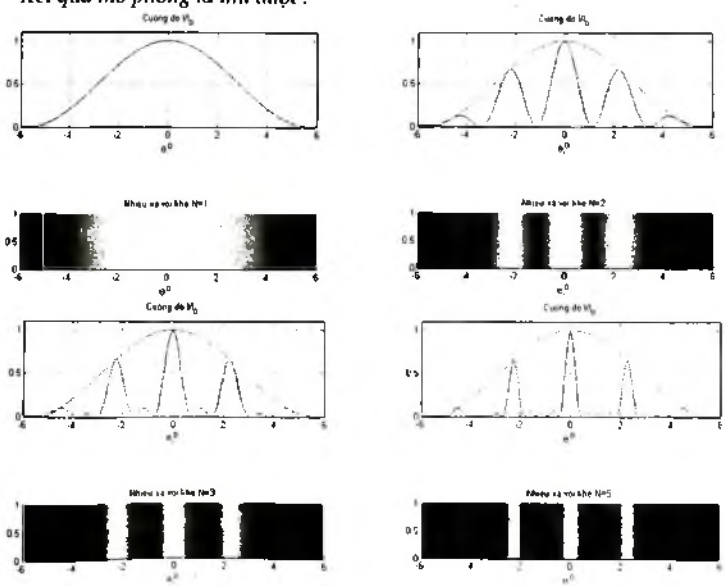
5.21. Mô phỏng hệ vân nhiễu xạ đặt trước lỗ tròn truyền sáng bán kính r.

Kết quả mô phỏng ta thu được:



5.22. Mô phỏng hệ vân nhiễu xạ với số khe N = [1, 2, 3, 5].

Kết quả mô phỏng ta thu được:



5.23. Vẽ đồ thị năng lượng bức xạ theo công thức Planck.

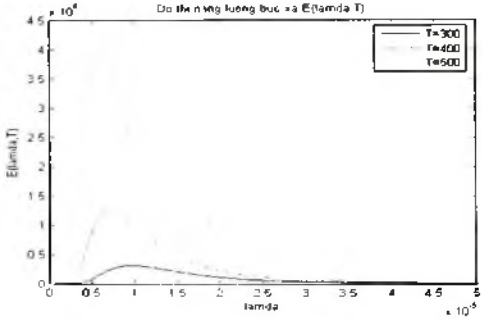
Kết quả: Planck coi lượng tử năng lượng ε_i ứng với tần số υ và có trị số bằng $h\upsilon$.

$h = 6{,}625.10^{-34}$ (J.s) gọi là hằng số Planck. Công thức Planck như sau:

$$\varepsilon_{\lambda,T} = \frac{2\pi hc^2}{\lambda^5} \frac{1}{e^{\frac{hc}{KT\lambda}} - 1}$$

Công thức này được nghiệm đúng với mức chính xác cao ở mọi nhiệt độ thực nghiệm và mọi bước sóng đo được.

Vẽ đồ thị năng lượng bức xạ $\varepsilon_{\lambda,T}$ với T = 300K; 400K; 500K



5.24. Dùng Matlab GUI, viết chương trình mô phỏng quá trình biến đổi từ một tín hiệu tương tự sang tín hiệu số. (Gồm các bước prefilter, lấy mẫu, lượng tử). Cho phép chọn một trong các dạng sóng tương tự vào sẵn có, nhập tần số tín hiệu, nhập tần số cắt của bộ prefilter (lí tưởng), nhập tần số lấy mẫu, tầm lượng tử và số bit mã hóa. Cho phép chọn các kiểu lượng tử khác nhau. Vẽ tín hiệu và phổ tương ứng ở mỗi bước trong quá trình biến đổi. (Tùy người dùng chọn mà hiển thị đồ thị tương ứng).

5.25. Dùng Matlab GUI, viết chương trình mô phỏng quá trình biến đổi từ một tín hiệu số sang tín hiệu tương tự (DAC).

5.26. Viết một chương trình Matlab (thể hiện trên GUI) cho phép nhập chiều dài khối ngõ vào, giá trị ngõ vào, chiều dài đáp ứng xung, giá trị đáp ứng xung. Tính chiều dài ngõ ra và giá trị ngõ ra (thể hiện trên tất cả các phương pháp đã học: bằng tích chập, dạng LTI, dạng ma trận...).

5.27. Viết chương trình bằng Matlab GUI để thực hiện xử lí và tạo hiệu ứng âm thanh, với âm thanh vào thu từ ngõ vào của soundcard, sau đó lưu thành file hoặc xuất ra ngõ ra để nghe.

5.28. Tìm hiểu về xử lí ảnh và viết chương trình GUI trên MATLAB thực hiện Edge Detection.

5.29. Viết một chương trình nhỏ bằng Matlab mô phỏng bộ lọc – chập trong đó ngõ vào là những khối riêng lẻ, đáp ứng xung là h(n) = (0.8)n. Tìm ngõ ra (Viết giao diện GUI cho phép nhập các giá trị ngõ vào).

Chương 6
MỘT SỐ ỨNG DỤNG CỦA NGÔN NGỮ LẬP TRÌNH MATLAB

§1. MÔ PHỎNG MỘT SỐ CƠ HỆ

1.1. Tạo giao diện trang giới thiệu và các menu

Trước tiên chúng ta xây dựng trang giới thiệu mở đầu. Ví dụ, khi thực hiện một bài tập lớn giải và mô phỏng kết quả một số bài tập cơ học, cần có một trang giới thiệu vấn đề, bố cục phần mềm...

Hình 6.1. Giao diện những trang giới thiệu: Lí do chọn đề tài, bố cục vấn đề...

Lập trình:

```
clear all
clc
% Trang gioi thieu
h=figure('name','MO PHONG MOT SO CO HE','menubar','none',...
  'numbertitle','off','color',[1 1 1]);
a=uicontrol(h,'string','MO PHONG MOT SO CO HE TRONG VAT LI',...
  'style','text','FontSize',18,'units','normalized','position',...
  [0 0 1 0.92],'backgroundcolor',[1 1 1],'foregroundcolor',[0 0 1]);
% Tao menu Gioi thieu
menu1=uimenu(h,'label','Gioi thieu');
m11=uimenu(menu1,'label','Li do chon de tai','callback','lido');
m13=uimenu(menu1,'label','Bo cuc phan mem','callback','bocuc');
```

```
% Tao menu Cac bai tap
menu2=uimenu(h,'label','Cac bai tap');
m21=uimenu(menu2,'label','Chuyen dong nem xien','callback','bai1cs1');
m22=uimenu(menu2,'label','Quy dao Kepler','callback','bai2cs1');
m23=uimenu(menu2,'label','Bai toan nhay tren ho nuoc','callback','bai3cs1');
% Tao menu exit
menu5=uimenu(h,'Label','Exit');
m5=uimenu(menu5,'Label','Exit','callback','close');
```

1.2. Thiết kế giao diện các bài tập cụ thể

1.2.1. Bài toán chuyển động ném xiên

1.2.1.1. Bài toán

Một vật được ném lên từ một độ cao H ở nơi có gia tốc trọng trường g với vận tốc ban đầu V_0 tạo với phương nằm ngang một góc α. Hãy:

– Xác định phương trình quỹ đạo của vật.

– Xác định thời gian vật rơi đến khi chạm đất.

– Xác định tầm xa cực đại của vật.

– Vẽ đồ thị quỹ đạo chuyển động.

1.2.1.2. Lập trình (tạo giao diện nhập dữ liệu)

```
clear all
clc
% A1. Tao cua so
h=figure('name','Chuyen dong nem xien','color',[0.95 0.95 0.95],...
    'numbertitle','off',...
    'units','normalized','position',[0.08 0.08 0.85 0.85])
% B1. Tao nhan tieu de
h1=uicontrol(h,'foregroundcolor','k',...
      'style','text','backgroundcolor',[0.95 0.95 0.95],...
      'string','Chuyen dong nem xien','FontSize',20,...
      'units','normalized','position',[0.1 0.85 0.65 0.07]);
% C1. Tao nhan moi nhap Vo
h2=uicontrol(h,'style','text','string','Nhap Vo (m/s)',...
      'foregroundcolor','k','horizontalalignment','left',...
```

```
        'backgroundcolor',[0.95 0.95 0.95],'FontSize',13,...
        'units','normalized','position',[0.06 0.72 0.15 0.07]);
% D1. Tao hop nhap Vo
h3=uicontrol(h,'style','edit','units','normalized',...
        'position',[0.18 0.72 0.15 0.07],'FontSize',14,...
        'callBack','Vo=str2num(get(h3,"string"))');
% E1. Tao nhan moi nhap H
h5=uicontrol(h,'style','text','string','Nhap H (m)',...
        'foregroundcolor','k','horizontalalignment','left',...
        'backgroundcolor',[0.95 0.95 0.95],'FontSize',13,...
        'units','normalized','position',[0.06 0.62 0.15 0.07]);
% F1. Tao hop nhap H
h6=uicontrol(h,'style','edit','units','normalized',...
        'position',[0.18 0.62 0.15 0.07],'FontSize',14,...
        'callBack','H=str2num(get(h6,"string"))');
% G1. Tao nhan moi nhap g
h7=uicontrol(h,'style','text','string','Nhap g (m/s^2)',...
        'foregroundcolor','k','horizontalalignment','left',...
        'backgroundcolor',[0.95 0.95 0.95],'FontSize',13,...
        'units','normalized','position',[0.06 0.52 0.15 0.07]);
% H1. Tao hop nhap g
h8=uicontrol(h,'style','edit','units','normalized',...
        'position',[0.18 0.52 0.15 0.07],'FontSize',14,...
        'callBack','g=str2num(get(h8,"string"))');
% I1. Tao nhan moi nhap alpha
h9=uicontrol(h,'style','text','string','Nhap Alpha (rad)',...
        'foregroundcolor','k','horizontalalignment','left',...
        'backgroundcolor',[0.95 0.95 0.95],'FontSize',13,...
        'units','normalized','position',[0.06 0.42 0.15 0.07]);
% J1. Tao hop nhap alpha
h10=uicontrol(h,'style','edit','units','normalized',...
        'position',[0.18 0.42 0.15 0.07],'FontSize',14,...
        'callBack','alpha=str2num(get(h10,"string"))');
```

```
% K1. Tao hop Ket qua phuong trinh quy dao
KQ1=uicontrol(gcf,'units','normalize','fontsize',14,...
  'backgroundcolor',[0.96 1 1],'position',[0.24 0.32 0.26 0.07],'style','edit');
syms x
KQ=uicontrol('style','pushbutton','backgroundcolor',[0.96 1 1],' foregroundcolor',
[0.57 0 1],'fontsize',14,...
   'string','PT quy dao',...
   'callback','d=-
g*x^2/((2*Vo*cos(alpha))^2)+tan(alpha)*x+H,y=char(d),set(KQ1,"string",y)',...
   'units','normalized','position',[0.02 0.32 0.2 0.07]);
% L1. Tao nut ket qua thoi gian bay
TG1=uicontrol(gcf,'units','normalize','fontsize',15,...
  'backgroundcolor',[0.96 1 1],'position',[0.24 0.22 0.26 0.07],'style','edit');
TG=uicontrol('style','pushbutton','backgroundcolor',[0.96 1 1],'foregroundcolor',
[0.57 0 1],...
   'string','Thoi gian cham dat (s)','callback',['t=(Vo*sin(alpha)+sqrt (Vo^2*
(sin(alpha))^2+2*g*H))/g,set(TG1,"string",t)'],...
   'fontsize',15,'units','normalized','position',[0.02 0.22 0.2 0.07])
% M1. Tao nut ket qua tam xa
TX1=uicontrol(gcf,'units','normalize','fontsize',15,...
  'backgroundcolor',[0.96 1 1],'position',[0.24 0.12 0.26 0.07],'style','edit');
xmax=uicontrol('style','pushbutton','backgroundcolor',[0.96 1 1],'foregroundcolor',
[0.57 0 1],...
'string','Tam bay xa (m)','callback',...
['L=(cos(alpha)*sin(alpha)*Vo^2+Vo*cos(alpha)*sqrt((Vo*sin(alpha))^2+2*
g*H))/g,set(TX1,"string",L)'],...
   'fontsize',15,'units','normalized','position',[0.02 0.12 0.2 0.07])
% N1. Tao he truc toa do
h2=axes('parent',h,'color','w',...
     'units','normalized','position',[0.55 0.1 0.4 0.7]);
xlabel ('Khoang cach x')
ylabel ('Do cao y')
grid on
```

```
% P1. Tao nut ve
Vedc=uicontrol(h,'style','pushbutton',...
        'string','Ve do thi','FontSize',14,...
        'units','normalized','foregroundcolor',[0.57 0 1],...
        'position',[0.06 0.02 0.2 0.07],...
        'callback',['x1=0:0.01:(round(L)+0.5);','y1=H+tan(alpha).*x1-
g.*x1.^2/(2.*Vo.^2*(cos(alpha))^2);',...
        'plot(x1,y1);','grid on;','xlabel ("Khoang cach x");','ylabel ("Do cao
y");'])
% Q1. Tao nut dong
Dong=uicontrol(h,'style','pushbutton','string','CLOSE',...
        'Fontsize',14,'units','normalized',...
        'foregroundcolor',[0.57 0 1],...
        'position',[0.3 0.02 0.2 0.07],'callback','close');
```

1.2.1.3. Kết quả

- Lập trình được chương trình giải bài toán chuyển động ném xiên trong trường hợp tổng quát cho phép nhập các đại lượng đầu vào từ điều kiện bài toán để xuất ra các đại lượng đầu ra theo yêu cầu của bài tập.

+ Các đại lượng nhập vào:

Vận tốc ban đầu V_0 (m/s)

Độ cao H (m)

Gia tốc trọng trường g (m/s^2)

Góc ném alpha (rad)

+ Các đại lượng xuất ra:

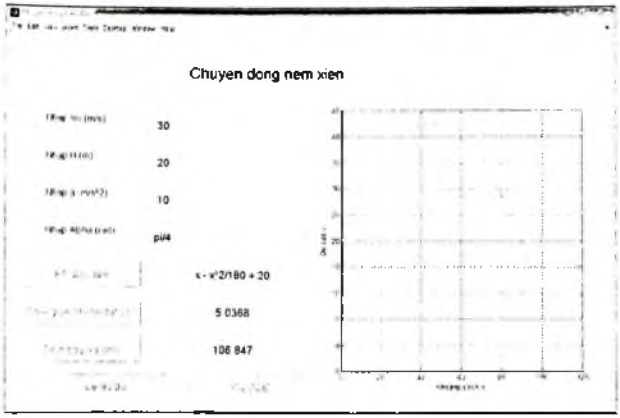
Phương trình quỹ đạo

Thời gian chuyển động đến khi chạm đất

Tầm xa cực đại

Đồ thị quỹ đạo

- Trong trường hợp góc alpha = 0 (rad), ta có bài toán mô tả chuyển động của vật ném ngang.



Hình 6.2. Giao diện quan sát mô phỏng của chuyển động ném xiên

1.2.2. Quỹ đạo Kepler

1.2.2.1. Bài toán

Khảo sát 8 hành tinh trong hệ Mặt Trời ta có bảng số liệu sau:

Planet	Mass (kg)	Axes (m)	Period (năm)	ep
Mercury	3.300*10^23	0.5791*10^11	0.241	0.026
Venus	4.870*10^24	1.0821*10^11	0.615	0.007
Earth	5.974*10^24	1.4960*10^11	1.000	0.017
Mars	6.419*10^23	2.2794*10^11	1.881	0.093
Jupiter	18.99*10^26	7.7840*10^11	11.860	0.048
Sarturn	5.680*10^26	14.2700*10^11	29.420	0.054
Uranus	8.680*10^25	28.7100*10^11	83.750	0.047
Neptune	10.20*10^25	44.9800*10^11	163.7	0.009

1/ Với mỗi hành tinh, hãy kiểm lại định luật III Kepler:

$$\frac{4a^2\pi^2}{T^2G} = M_{sun} + M_{planet} \approx M_{sun}$$

Trong đó: + T là chu kì quay (tính theo giây)

+ G: hằng số hấp dẫn

+ a: bán trục chính của quỹ đạo

2/ Vẽ quỹ đạo tương ứng của các hành tinh đó.

1.2.2.2. Lập trình (Tạo giao diện nhập dữ liệu)

```
clear all
clc
% A2. Tao cua so
h=figure('name','Quy dao Kepler','color',[0.95 0.95 0.95],...
    'numbertitle','off',...
    'units','normalized','position',[0.08 0.08 0.85 0.85])
% B2. Tao nhan tieu de
h1=uicontrol(h,'foregroundcolor','k',...
        'style','text','backgroundcolor',[0.95 0.95 0.95],...
        'string','Quy dao Kepler','FontSize',20,...
        'units','normalized','position',[0.1 0.9 0.65 0.07]);
% C2. Tao nhan khoi luong cua Mat Troi
h0=uicontrol(h,'style','text','string','Khoi luong Mat Troi Ms=1.99*10^30
kg',...
        'foregroundcolor','k','horizontalalignment','left',...
        'backgroundcolor',[0.95 0.95 0.95],'FontSize',13,...
        'units','normalized','position',[0.08 0.74 0.35 0.07]);
    Ms=1.99*10^30;
% D2. Tao nhan hang so hap dan G
h01=uicontrol(h,'style','text','string','Hang so hap dan G=6.67*10^-11 m^3*
kg^-1*s^-2',...
        'foregroundcolor','k','horizontalalignment','left',...
        'backgroundcolor',[0.95 0.95 0.95],'FontSize',13,...
        'units','normalized','position',[0.08 0.66 0.35 0.07]);
    G=6.67*10^-11;
% E2. Tao nhan moi nhap ban truc a (m)
h5=uicontrol(h,'style','text','string','Nhap ban truc quy dao a (m)',...
        'foregroundcolor','k','horizontalalignment','left',...
```

```
        'backgroundcolor',[0.95 0.95 0.95],'FontSize',13,...
        'units','normalized','position',[0.08 0.58 0.3 0.07]);
% F2. Tao hop nhap a
h6=uicontrol(h,'style','edit','units','normalized',...
        'position',[0.36 0.58 0.15 0.07],'FontSize',14,...
        'callBack','a=str2num(get(h6,"string"))');
% G2. Tao nhan moi nhap chu ki T (nam)
h7=uicontrol(h,'style','text','string','Nhap chu ki T (nam)',...
        'foregroundcolor','k','horizontalalignment','left',...
        'backgroundcolor',[0.95 0.95 0.95],'FontSize',13,...
        'units','normalized','position',[0.08 0.50 0.3 0.07]);
% H2. Tao hop nhap T
h8=uicontrol(h,'style','edit','units','normalized',...
        'position',[0.36 0.50 0.15 0.07],'FontSize',14,...
        'callBack','T=str2num(get(h8,"string"))');
% I2. Tao nhan moi nhap ep
h9=uicontrol(h,'style','text','string','Nhap ep',...
        'foregroundcolor','k','horizontalalignment','left',...
        'backgroundcolor',[0.95 0.95 0.95],'FontSize',13,...
        'units','normalized','position',[0.08 0.42 0.3 0.07]);
% J2. Tao hop nhap ep
h10=uicontrol(h,'style','edit','units','normalized',...
        'position',[0.36 0.42 0.15 0.07],'FontSize',14,...
        'callBack','ep=str2num(get(h10,"string"))');
% K2. Tao hop Ket qua ve trai
KQ1=uicontrol(gcf,'units','normalize','fontsize',14,...
  'backgroundcolor',[0.96 1 1],'position',[0.36 0.34 0.15 0.07],'style','edit');
VT=uicontrol('style','pushbutton','backgroundcolor',[0.96 1 1],'foregroundcolor',
[0.57 0 1],'fontsize',14,...
'string','Ve trai 4*a^3*pi^2/(T^2*G)',... 'callback','nam=365*86400+ 6*3600
+ 9*60+10,Ts=T*nam,VT=4*a^3*pi^2/(Ts^2*G),set(KQ1,"string",VT)',...
  'units','normalized','position',[0.08 0.34 0.25 0.07]);
```

```
% L2. Tao he truc toa do
h2=axes('parent',h,'color','w',...
    'units','normalized','position',[0.55 0.2 0.4 0.6]);
grid on
% M2.Tao nut ve
Vedc=uicontrol(h,'style','pushbutton',...
        'string','Ve do thi','FontSize',14,...
        'units','normalized','foregroundcolor',[0.57 0 1],...
        'position',[0.08 0.18 0.2 0.07],...
      'callback','theta=0:0.01:2*pi;r=(a*(1-ep^2))./(1+ep.*cos(theta));,polar(theta,r);,
hold on')
% N2. Tao nut dong
Dong=uicontrol(h,'style','pushbutton','string','CLOSE',...
          'Fontsize',14,'units','normalized',...
          'foregroundcolor',[0.57 0 1],...
          'position',[0.3 0.18 0.2 0.07],'callback','close');
```

1.2.2.3. Kết quả

Lập trình được chương trình giải bài toán khảo sát tám hành tinh trong hệ Mặt Trời:

+ Nhập vào các số liệu tương ứng của mỗi hành tinh để kiểm chứng định luật III Kepler.

+ Vẽ quỹ đạo tương ứng của các hành tinh theo số liệu nhập vào.

Hình 6.3. Định luật III Kepler đối với Trái Đất. Quỹ đạo Kepler của Trái Đất và Hỏa Tinh

1.2.3. Bài toán nhảy trên hồ nước

1.2.3.1. Bài toán

Một người đứng trên độ cao cách mặt hồ một khoảng H có buộc một sợi dây đàn hồi chiều dài *l* và nhảy xuống mặt nước. Giả sử rằng lực cản của không khí tỉ lệ bậc nhất với vận tốc của vật và phương trình động lực học có dạng:

$$\vec{F} = m\vec{a} = m\vec{g} - k_1\vec{v} - k_2\Delta\vec{l}$$

Xác định phương trình dao động của người và vẽ đồ thị quỹ đạo chuyển động trong trường hợp này.

1.2.3.2. Lập trình (Tạo giao diện nhập dữ liệu)

```
clear all
clc
% A3. Tao cua so
h=figure('name','Bai toan nhay tren ho nuoc','color',[0.95 0.95 0.95],...
    'numbertitle','off',...
    'units','normalized','position',[0.08 0.08 0.85 0.85])
% B3. Tao nhan tieu de
h1=uicontrol(h,'foregroundcolor','k',...
    'style','text','backgroundcolor',[0.95 0.95 0.95],...
    'string','Bai toan nhay tren ho nuoc','FontSize',20,...
    'units','normalized','position',[0.1 0.85 0.65 0.07]);
% C3. Tao nhan moi nhap khoi luong cua nguoi
h2=uicontrol(h,'style','text','string','Nhap khoi luong m (kg)',...
    'foregroundcolor','k','horizontalalignment','left',...
    'backgroundcolor',[0.95 0.95 0.95],'FontSize',13,...
    'units','normalized','position',[0.02 0.72 0.16 0.07]);
% D3. Tao hop nhap m
h3=uicontrol(h,'style','edit','units','normalized',...
    'position',[0.2 0.72 0.15 0.07],'FontSize',14,...
    'callBack','m=str2num(get(h3,"string"))');
% E3. Tao nhan moi nhap he so can cua khong khi
h4=uicontrol(h,'style','text','string','Nhap he so can k1',...
    'foregroundcolor','k','horizontalalignment','left',...
```

```
        'backgroundcolor',[0.95 0.95 0.95],'FontSize',13,...
        'units','normalized','position',[0.02 0.62 0.16 0.07]);
% F3. Tao hop nhap k1
h5=uicontrol(h,'style','edit','units','normalized',...
        'position',[0.2 0.62 0.15 0.07],'FontSize',14,...
        'callBack','k1=str2num(get(h5,"string"))');
% G3. Tao nhan moi nhap he so dan hoi k2
h6=uicontrol(h,'style','text','string','Nhap he so dan hoi k2 (N/m)',...
        'foregroundcolor','k','horizontalalignment','left',...
        'backgroundcolor',[0.95 0.95 0.95],'FontSize',13,...
        'units','normalized','position',[0.02 0.52 0.16 0.07]);
% H3. Tao hop nhap k2
h7=uicontrol(h,'style','edit','units','normalized',...
        'position',[0.2 0.52 0.15 0.07],'FontSize',14,...
        'callBack','k2=str2num(get(h7,"string"))');
% I3. Tao nhan moi nhap do cao cua thac
h8=uicontrol(h,'style','text','string','Nhap do cao H (m)',...
        'foregroundcolor','k','horizontalalignment','left',...
        'backgroundcolor',[0.95 0.95 0.95],'FontSize',13,...
        'units','normalized','position',[0.02 0.42 0.16 0.07]);
% J3. Tao hop nhap H
h9=uicontrol(h,'style','edit','units','normalized',...
        'position',[0.2 0.42 0.15 0.07],'FontSize',14,...
        'callBack','H=str2num(get(h9,"string"))');
% K3. Tao nhan moi nhap chieu dai day
h10=uicontrol(h,'style','text','string','Nhap chieu dai L (m)',...
        'foregroundcolor','k','horizontalalignment','left',...
        'backgroundcolor',[0.95 0.95 0.95],'FontSize',13,...
        'units','normalized','position',[0.02 0.32 0.16 0.07]);
% L3.Tao hop nhap L
h11=uicontrol(h,'style','edit','units','normalized',...
        'position',[0.2 0.32 0.15 0.07],'FontSize',14,...
        'callBack','L=str2num(get(h11,"string"))');
```

```
% M3. Tao nut hien thi phuong trinh dong luc hoc
g=9.8;
KQ1=uicontrol(gcf,'units','normalize','fontsize',13,...
  'backgroundcolor',[0.96 1 1],'position',[0.12 0.22 0.3 0.07],'style','edit');
PT=uicontrol(h,'style','pushbutton',...
        'string','PT','FontSize',13,...
        'units','normalized','foregroundcolor',[0.57 0 1],...
        'position',[0.02 0.22 0.1 0.07],...
        'callback',...
        'ODE="D2x=-g-(k1/m)*Dx+(k2/m)*(H-L-x)",DE=subs(ODE),
a=char(DE),set(KQ1,"string",a)')
% N3. Tao nut hien thi nghiem cua phuong trinh:
KQ2=uicontrol(gcf,'units','normalize','fontsize',13,...
  'backgroundcolor',[0.96 1 1],'position',[0.12 0.12 0.3 0.07],'style','edit');
PT2=uicontrol(h,'style','pushbutton',...
        'string','x=','FontSize',13,...
        'units','normalized','foregroundcolor',[0.57 0 1],...
        'position',[0.02 0.12 0.1 0.07],...
  'callback',['b=str2mat(get(KQ1,"string")),c=dsolve(b,"x(0)=H","Dx(0)=0","t"),
x=char(subs(c)),set(KQ2,"string",x)'])
% P3. Tao he truc toa do
h0=axes('parent',h,'color','w',...
    'units','normalized','position',[0.55 0.1 0.4 0.7]);
xlabel ('Thoi gian t')
ylabel ('Li do x')
grid on
% Q3. Tao nut ve
Ve=uicontrol(h,'style','pushbutton',...
        'string','Do thi','FontSize',13,...
        'units','normalized','foregroundcolor',[0.57 0 1],...
        'position',[0.02 0.02 0.1 0.07],...
        'callback',['d=str2mat(get(KQ2,"string")),ezplot(d,[0.44]),grid
on,xlabel ("Thoi gian t"),ylabel ("Li do x")'])
```

```
% R3. Tao nut dong
Dong=uicontrol(h,'style','pushbutton','string','CLOSE',...
        'Fontsize',14,'units','normalized',...
        'foregroundcolor',[0.57 0 1],...
        'position',[0.2 0.02 0.15 0.07],'callback','close');
```

1.2.3.3. Kết quả

Lập trình được chương trình giải bài toán nhảy trên hồ nước đã nêu ở trên trong trường hợp tổng quát:

+ Các đại lượng nhập vào:

Khối lượng của người nhảy m

Hệ số cản của không khí k1

Hệ số đàn hồi của dây k2

Độ cao H

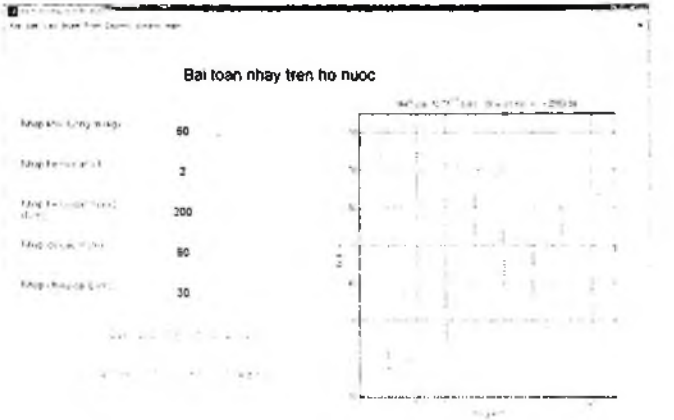
Chiều dài của dây L

+ Các đại lượng xuất ra:

Phương trình động lực học

Nghiệm của phương trình

Đồ thị quỹ đạo



Hình 6.4. Giao diện kết quả của bài toán nhảy trên hồ nước

1.2.4. Khảo sát thực nghiệm chuyển động rơi của vật trong không khí

1.2.4.1. Bài toán

Nghiên cứu bài thực hành sự rơi tự do của một vật trong không khí. Hãy thiết kế một phần mềm mô phỏng chuyển động rơi của vật từ kết quả thực nghiệm đo đạc được (ví dụ vẽ đồ thị li độ theo thời gian và đồ thị vận tốc theo thời gian).

1.2.4.2. Lập trình thiết kế giao diện phần mềm

* Khởi tạo trang chủ

```
h=figure('name','khao sat thuc nghiem chuyen dong roi cua vat trong khong
khi','menubar','none',...
  'numbertitle','off','color',[1 1 1]);
a=uicontrol(h,'string','BAI TAP LON NHOM PHUONG PHAP',...
  'style','text','FontSize',18,'units','normalized','position',...
  [0 0.1 1 0.92],'backgroundcolor',[1 1 1],'foregroundcolor',[0 0 1]);
b=uicontrol(h,'string','Nhung nguoi thuc hien:',...
  'style','text','FontSize',15,'units','normalized','position',...
  [0 0.1 1 0.82],'backgroundcolor',[1 1 1],'foregroundcolor',[0 0 1]);
%tao menu GIOI THIEU
menu1=uimenu(h,'label','Gioi thieu');
m1=uimenu(menu1,'label','Ly do chon de tai','callback','gioithieu');
m2=uimenu(menu1,'label','Matlab la gi?','callback','gioithieu1');
m3=uimenu(menu1,'label','y nghia khoa hoc','callback','gioithieu2');
m4=uimenu(menu1,'label','Bo cuc bai tap lon','callback','gioithieu3');
%tao menu chuc nang
menu2=uimenu(h,'label','chuc nang');
m4=uimenu(menu2,'label','dua ket qua thuc nghiem','callback','NHAPTT');
m4=uimenu(menu2,'label','Ve do thi','callback','ve');
m5=uimenu(menu2,'label','do thi van toc','callback','vantoc');
m6=uimenu(menu2,'label','nhan xet rut ra','callback','nhanxet');
%tao menu exit
menu3=uimenu(h,'Label','Thoat');
m7=uimenu(menu3,'Label','dong lai','callback','close');
```

* Tạo giao diện nhập số liệu

```
clear all;
```

```
%TAO GIAO DIEN
h1=figure('name','NHAP THONGTIN','color',[0.1 0.4 0.5],'numbertitle', 'off',
'units',...
  'normalized','position',[0.1 0.1 0.7 0.7]);
%TAO TIEU DE
h2=uicontrol(gcf,'style','text','backgroundcolor','w','foregroundcolor','b',...
  'string','NHAP THONG TIN TU THUC NGHIEM:','FontSize', 20,'units',
'normalized',...
  'position',[0.1 0.9 0.8 0.06]);
%TAO TIEU DE NHAP LI DO
h2=uicontrol(gcf,'style','text','backgroundcolor','w','foregroundcolor','b',...
  'string','LI DO','FontSize',20,'units','normalized',...
  'position',[0.15 0.65 0.3 0.07]);
%TAO TIEU DE NHAP THOI GIAN
h2=uicontrol(gcf,'style','text','backgroundcolor','w','foregroundcolor','b',...
  'string','THOI GIAN','FontSize',20,'units','normalized',...
  'position',[0.6 0.65 0.3 0.07]);
%TAO NUT TIEU DE NHAP LI DO 1
LD1=uicontrol(gcf,'style','edit',
'FontSize',18,'backgroundcolor','w','horizontalalignment',...
'left','units','normalized','position',[0.15 0.55 0.3 0.07], 'callback', 'LD1=str2num
(get(LD1,"string"))');
%TAO NUT NHAP THOI GIAN 1
TG1=uicontrol(gcf,'style','edit',
'FontSize',18,'backgroundcolor','w','horizontalalignment',...
'left','units','normalized','position',[0.6 0.55 0.3 0.07], 'callback', 'TG1=str2num
(get(TG1,"string"))');
%TAO NUT TIEU DE NHAP LI DO 2
LD2=uicontrol(gcf,'style','edit', 'FontSize',18,'backgroundcolor','w', 'horizonta
lalignment',...
'left','units','normalized','position',[0.15 0.45 0.3 0.07], 'callback', 'LD2= str2num
(get(LD2,"string"))');
%TAO NUT NHAP THOI GIAN 2
TG2=uicontrol(gcf,'style','edit','FontSize',18,'backgroundcolor','w','horizontala
```

lignment','left',... 'horizontalalignment', 'left', 'units', 'normalized', 'position',... [0.6 0.45 0.3 0.07], 'callback', 'TG2=str2num(get(TG2,"string"))');

%TAO NUT TIEU DE NHAP LI DO3

LD3=uicontrol(gcf,'style','edit',
'FontSize',18,'backgroundcolor','w','horizontalalignment',...

'left','units','normalized','position',[0.15 0.35 0.3 0.07], 'callback', 'LD3= str2 num(get(LD3,"string"))');

%TAO NUT NHAP THOI GIAN 3

TG3=uicontrol(gcf,'style','edit', 'FontSize', 18, 'backgroundcolor', 'w', 'horizon talalignment', 'left',...'horizontalalignment','left','units','normalized','position',...[0.6 0.35 0.3 0.07], 'callback', 'TG3=str2num(get(TG3,"string"))');

%TAO NUT TIEU DE NHAP LI DO 4

LD4=uicontrol(gcf,'style','edit',
'FontSize',18,'backgroundcolor','w','horizontalalignment',...

'left','units','normalized','position',[0.15 0.25 0.3 0.07], 'callback', 'LD4= str2 num(get(LD4,"string"))');

%TAO NUT NHAP THOI GIAN 4

TG4=uicontrol(gcf,'style','edit', 'FontSize', 18, 'backgroundcolor', 'w', 'horizon talalignment', 'left',...'horizontalalignment', 'left','units','normalized','position',...[0.6 0.25 0.3 0.07], 'callback', 'TG4=str2num(get(TG4,"string"))');

%TAO NUT TIEU DE NHAP LI DO 5

LD5=uicontrol(gcf,'style','edit','FontSize',18,'backgroundcolor','w','horizontala lignment',...

'left','units','normalized','position',[0.15 0.15 0.3 0.07], 'callback', 'LD5=str2num (get(LD5,"string"))');

%TAO NUT NHAP THOI GIAN 5

TG5=uicontrol(gcf,'style','edit', 'FontSize', 18, 'backgroundcolor', 'w', 'horizontalalignment','left',...'horizontalalignment','left','units','normalized','position' ...[0.6 0.15 0.3 0.07], 'callback', 'TG5=str2num(get(TG5,"string"))');

%TAO NUT DONG

h6=uicontrol(gcf,'style','pushbutton','string','CLOSE','units','normalized',...'pos ition',[0.7 0.05 0.1 0.05], 'backgroundcolor',[0.01 0.3 0.2], 'foregroundcolor', 'Y',...'callback','close(gcf)');

Vẽ đồ thị biến thiên li độ theo thời gian

```
%TAO GIAO DIEN
h1=figure('name','ve do thi','color',[0.1 0.4 0.5], 'numbertitle', 'on','units',...
'normalized','position',[0.1 0.1 0.7 0.7]);
%TAO TIEU DE
h2=uicontrol(gcf,'style','text','backgroundcolor','w','foregroundcolor','b',...'stri
ng', 've do thi', 'FontSize', 20, 'units', 'normalized', ...'position', [0.07 0.95 0.2
0.05]);
x=[TG1,TG2,TG3,TG4,TG5];
y=[LD1,LD2,LD3,LD4,LD5]
plot(x,y)
xlabel('thoi gian')
ylabel('li do')
```

Vẽ đồ thị vận tốc theo thời gian

```
%TAO GIAO DIEN
h1=figure('name','ve do thi','color',[0.1 0.4 0.5],'numbertitle','on','units',
'normalized', 'position', [0.1 0.1 0.7 0.7]);
%TAO TIEU DE
h2=uicontrol(gcf,'style','text','backgroundcolor','w','foregroundcolor','b',...
'string', 'do thi van toc', 'FontSize', 18, 'units','normalized',... 'position',[0.07 0.95
0.5 0.05]);
v1=0;
v2=(LD2-LD1)/(TG2-TG1);
v3=(LD3-LD2)/(TG3-TG2);
v4=(LD4-LD3)/(TG4-TG3);
v5=(LD5-LD4)/(TG5-TG4);
v=[v1,v2,v3,v4,v5];
plot(v)
xlabel('thoi gian(s)')
ylabel('van toc(m/s)')
%TAO NUT DONG
h6=uicontrol(gcf,'style','pushbutton','string','CLOSE','units','normalized',...
'position', [0.7 0.01 0.1 0.05], 'backgroundcolor', [0.01 0.3 0.2], 'foregroundcolor',
'R',... 'callback','close(gcf)')
```

1.2.4.3. Kết quả thực hiện

Lập trình được chương trình mô phỏng thực nghiệm sự rơi tự do trong không khí:

+ Các đại lượng nhập vào từ đo đạc thực nghiệm:

Li độ của vật

Thời gian

+ Các kết quả xuất ra:

Đồ thị li độ theo thời gian

Đồ thị vận tốc theo thời gian

Hình 6.5. Giao diện nhập số liệu

Hình 6.6. Đồ thị li độ theo thời gian

Hình 6.7. Đồ thị vận tốc theo thời gian

Hình 6.8. Giao diện đánh giá kết quả

§2. MÔ PHỎNG MỘT SỐ BÀI TOÁN PHẦN ĐIỆN - TỪ

2.1. Giao diện giới thiệu

a) Phần giao diện giới thiệu và các menu được thực hiện tương tự như mục §1.

b) Lập trình

```
% Các menu
h=figure('name',' Giai mot so bai tap vat li dai cuong phan dien – tu ','menubar','none', ...'numbertitle','off','color',[1 1 1]);
a1=uicontrol(h,'string','Bai tap lon phan Dien – Tu hoc!',...'style', 'text','FontSize',40,'units','normalized','position',[0 0 1 0.92], 'backgroundcolor', [1 1 1],'foregroundcolor',[0 0 1]);
% Tao menu GIOI THIEU
menu1=uimenu(h,'label','Gioi thieu');
m1=uimenu(menu1,'label','Li do chon de tai','callback','Lidochondetai');
% Tao menu chuc nang
menu2=uimenu(h,'label','Tien ich');
m2=uimenu(menu2,'label','Luong cuc dien','callback','Bai1');
m2=uimenu(menu2,'label','Dia tron','callback','Bai2');
m2=uimenu(menu2,'label','Chuyen dong cua hat mang dien trong tu truong deu va dien truong bien doi theo thoi gian','callback','Bai3');
% Tao menu exit
menu3=uimenu(h,'Label','Exit');
m5=uimenu(menu3,'Label','Exit','callback','close');
% Li do chon đề tài
h=figure('name',' Giai mot so bai tap vat li dai cuong phan dien – tu ','menubar','none',... 'numbertitle','off','color',[1 1 1]);
a=uicontrol(h,'string','Kha nang ung dung cua Matlab ',...'style','text', 'FontSize',20, 'units','normalized','position',[0 1 1 0.92],'backgroundcolor',[1 1 0], 'foregroundcolor',[1 0 1]);
a1=uicontrol(h,'string','Matlab la mot ngon ngu lap trinh thuc hanh bac cao duoc su dung de giai cac bai toan ve ki thuat. Matlab tich hop duoc nhieu tinh toan, the hien ket qua, cho phep lap trinh, giao dien lam viec de dang cho nguoi su dung. Du lieu cung voi thu vien duoc lap trinh san cho phep nguoi su dung co the co duoc nhieu ung dung. Trong mon vat li dai cuong, phan dien-tu hoc co nhieu
```

bai tap ung dung quan trong trong ki thuat va thuc tien. Matlab la cong cu huu hieu de giai cac bai tap do mot cach tong quat va trong cac truong hop cu the (voi nhung so lieu cu the).',...'style','text','FontSize',30,'units','normalized','position',[0 0 1 0.92],'backgroundcolor',[1 1 1],'foregroundcolor',[0 0 1]);

2.2. Thiết kế giao diện các bài tập cụ thể

2.2.1. Bài tập về lưỡng cực điện

2.2.1.1. Bài toán

Lưỡng cực điện là một hệ gồm 2 điện tích giống nhau nhưng trái dấu đặt cách nhau một khoảng rất bé so với khoảng cách r từ điểm quan sát đến tâm của lưỡng cực điện.

a) Tính điện thế tại điểm quan sát có toạ độ r, góc θ.

b) Tính điện trường của lưỡng cực tại chính điểm đó.

2.2.1.2. Lập trình

```
disp('Luong cuc dien la mot he gom hai dien tich giong nhau nhung trai dau')
disp('dat cach nhau mot khoang l rat be so voi khoang cach r tu diem quan sat den tam cua luong cuc.')
disp('a. Tinh dien the tai diem quan sat co toa do r, theta.')
disp('b. Tinh dien truong cua luong cuc tai chinh diem do.')
syms l q r r1 r2 V theta ep0 Ex Ey E;
[x,y] = pol2cart(theta,r)
r1=r+l*cos(theta)/2;
r2=r-l*cos(theta)/2;
%a. Dien the tai diem co toa do r, theta la:
V=q*(r1-r2)/(4*pi*ep0*r1*r2);
disp('a. Dien the tai diem co toa do r, theta la:')
V1=subs(simplify(V),l^2,0)
V=subs(V1,'cos(theta)','x/r');
V=subs(V1,r,'sqrt(x^2+y^2)');
%b. Dien truong cua luong cuc tai chinh diem do la:
Ex=-diff(V,'x');
Ey=-diff(V,'y');
E=sqrt(Ex^2+Ey^2);
E=simplify(subs(E,'x','r*cos(theta)'));
E=simplify(subs(E,'y','r*sin(theta)'));
```

```
disp('b. Dien truong cua luong cuc tai chinh diem do la:')
E=simplify(subs(E,'(sin(theta))^2','1-(cos(theta))^2'))
%Nhap cac gia tri l, r, theta, q
disp('Nhap cac gia tri l, r, theta')
l=input('Nhap l=')
r=input('Nhap r=')
q=input('Nhap q=')
theta=input('Nhap theta=')
disp('Tinh dien the:')
V=subs(V1,'l',l);
V=subs(V,'r',r);
V=subs(V,'theta',theta);
V=subs(V,'q',q)
disp('Tinh dien truong:')
E=subs(E,'l',l);
E=subs(E,'r',r);
E=subs(E,'theta',theta);
E=subs(E,'q',q)
```

2.2.1.3. Kết quả

Luong cuc dien la mot he gom hai dien tich giong nhau nhung trai dau dat cach nhau mot khoang *l* rat be so voi khoang cach r tu diem quan sat den tam cua luong cuc.

```
a. Tinh dien the tai diem quan sat co toa do r, theta.
b. Tinh dien truong cua luong cuc tai chinh diem do.
x = r*cos(theta)
y = r*sin(theta)
Dien the tai diem co toa do r, theta la:
V1 = 1/4*q*l*cos(theta)/pi/ep0/r^2
Dien truong cua luong cuc tai chinh diem do la:
E = 1/2/pi*(q^2*l^2*cos(theta)^2/ep0^2/r^6)^(1/2)
Nhap cac gia tri l, r, theta
Nhap l=0.0005
l = 5.0000e^-004
```

Nhap r=1

r = 1

Nhap q=10^-8

q = 1.0000e^-008

Nhap theta=pi/3

theta = 1/3*pi

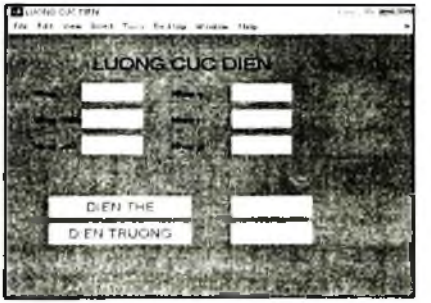
Tinh dien the:

V = 1/1600000000000/pi/ep0

Tinh dien truong:

E =

1/32000000000000000000000/pi*4000000^(1/2)*4^(1/2)*100000000000000000^(1/2)*(1/ep0^2)^(1/2)



Hình 6.9. Giao diện đầu vào của bài toán lưỡng cực điện

2.2.2. Bài tập về đĩa tròn

2.2.2.1. Bài toán

Một đĩa tròn bán kính R, tích điện với mật độ điện tích mặt không đổi sigma=q/(pi*R^2)

a) Xác định điện thế, điện trường dọc theo trục của đĩa.

b) Xét trường hợp khi R → ∞ để đĩa trở thành mặt phẳng tích điện đều.

2.2.2.2. Lập trình

disp('Mot dia tron ban kinh R, tich dien voi mat do dien tich mat khong doi sigma=q/(pi*R^2')

disp('a. Xac dinh dien the, dien truong doc theo truc cua dia.')

```
disp('b. Xet truong hop khi R->vo cung de dia tro thanh mat phang tich dien deu.')
syms z r x y k sigma R dq dr
%Xac dinh dien the doc theo truc cua dia
disp('Dien the doc theo truc cua dia la:')
dS=2*pi*r*dr;
dq=sigma*dS;
dV=k*dq/sqrt(z^2+r^2);
V=int(dV/dr,r,0)
%Xac dinh dien truong doc theo truc cua dia
disp('Dien truong doc theo truc cua dia la:')
E=[ -diff(V,x) -diff(V,y) -diff(V,z)]
%Xet truong hop R->vo cung
disp('Tinh dien truong khi R-> vo cung')
Et=limit(E,R,inf)
%Nhap cac gia tri R, sigma, k, z
disp('Nhap cac gia tri R, sigma, k, z')
R=input('Nhap gia tri R=')
sigma=input('Nhap gia tri sigma=')
k=input('Nhap gia tri k=')
z=input('Nhap gia tri z=')
%Tinh V, E va Et theo cac gia tri vua nhap
disp('Tinh V, E va Et theo cac gia tri vua nhap')
V=subs(V,'R',R);
V=subs(V,'sigma',sigma);
V=subs(V,'k',k);
V=subs(V,'z',z)
E=subs(E,'R',R);
E=subs(E,'sigma',sigma);
E=subs(E,'k',k);
E=subs(E,'z',z)
Et=subs(Et,'sigma',sigma);
Et=subs(Et,'k',k);
Et=subs(Et,'z',z)
```

2.2.2.3. Kết quả

Mot dia tron ban kinh R, tich dien voi mat do dien tich mat khong doi sigma=q/(pi*R^2)

a. Xac dinh dien the, dien truong doc theo truc cua dia.

b. Xet truong hop khi R->vo cung de dia tro thanh mat phang tich dien deu.

Dien the doc theo truc cua dia la:

V = 2*k*sigma*pi*(z^2+R^2)^(1/2)-2*(z^2)^(1/2)*k*sigma*pi

Dien truong doc theo truc cua dia la:

E = [0, 0, -2*k*sigma*pi/(z^2+R^2)^(1/2)*z+2/(z^2)^(1/2)*k*sigma*pi*z]

Tinh dien truong khi R-> vo cung

Et = [0, 0, 2/(z^2)^(1/2)*k*sigma*pi*z]

Nhap cac gia tri R, sigma, k, z

Nhap gia tri R=0.03

R = 0.0300

Nhap gia tri sigma=10^-9

sigma = 1.0000e^-009

Nhap gia tri k=9*10^9

k = 9.0000e^+009

Nhap gia tri z=0.01

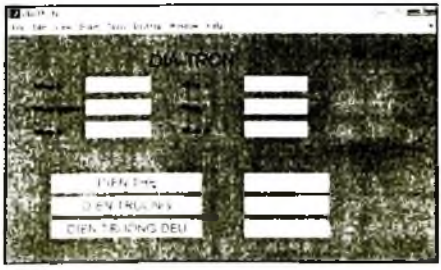
z = 0.0100

Tinh V, E va Et theo cac gia tri vua nhap

V = 1.2227

E = 0 0 38.6664

Et = 0 0 56.5487



Hình 6.10. Giao diện đầu vào của bài toán đĩa tròn

2.2.3. Bài tập về chuyển động của hạt mang điện trong điện trường và từ trường

2.2.3.1. Bài toán

Xét chuyển động của một điện tích trong từ trường đều và điện trường biến đổi theo thời gian. Hãy dùng đồ thị mô phỏng chuyển động đó.

2.2.3.2. Lập trình

```
disp('Bai toan hat mang dien chuyen dong trong truong dien tu')
disp('Dien truong huong doc theo truc x va co dang E=E0*cos(omega*t)')
disp('Tu truong khong doi B=m*omega0/q huong doc theo truc z')
disp('Phuong trinh chuyen dong: m.r''=q.E + q[vxB]')
disp('-----------------------------------------')
ode1='D2x-omega0*Dy=q/m*E0*cos(omega*t)';
ode2='D2y+omega0*Dx=0';
ode3='D2z=0';
%Giai he pt voi dieu kien ban dau
%'x(0)=0','Dx(0)=0','y(0)=0','Dy(0)=0','z(0)=0','Dz(0)=0','t'
[x,y,z]=dsolve(ode1,ode2,ode3,'x(0)=0','Dx(0)=0','y(0)=0','Dy(0)=0','z(0)=0','
Dz(0)=0','t');
x=Simplify(x)
y=Simplify(y)
z=Simplify(z)
%Nhap du lieu dau vao m,q,E0,omega0,omega
disp('Nhap du lieu dau vao m,q,E0,omega0,omega')
m=input('Nhap m=');
q=input('Nhap q=');
E0=input('Nhap E0=');
omega0=input('Nhap omega0=');
ome=input('Nhap omega=');
disp(' Gioi han thoi gian [tmin,tmax] ')
tmin=input('Nhap tmin=');
tmax=input('Nhap tmax=');
%Tinh x, y, z voi cac tham so dau vao
x=subs(x,'m',m);
x=subs(x,'q',q);
x=subs(x,'E0',E0);
```

```
x=subs(x,'omega0',omega0);
y=subs(y,'m',m);
y=subs(y,'q',q);
y=subs(y,'E0',E0);
y=subs(y,'omega0',omega0);
%Gia tri x,y khi omega = ome
syms omega;
x=limit(x,omega,ome)
y=limit(y,omega,ome)
%Cua So
cs=figure('name','Ve do thi','color',[0.96,1,1],'numbertitle','off',...
  'resize','on','units','Normalized','position',[0.1,0.1,0.85,0.85]);
%ve do thi 3D
syms t Fx Fy Fz;
Fx=x;
Fy=y;
Fz=z;
ezplot3(Fx,Fy,Fz,[tmin:tmax])
title('Do thi 3D hat mang dien chuyen dong:')
xlabel('Truc x')
ylabel('Truc y')
zlabel('Truc z')
grid on
```

2.2.3.3. Kết quả

Hình 6.11. Giao diện đầu vào, kết quả của bài toán hạt mang điện chuyển động trong từ trường và điện trường

Bai toan hat mang dien chuyen dong trong truong dien tu

Dien truong huong doc theo truc x va co dang E=E0*cos(omega*t)

Tu truong khong doi B=m*omega0/q huong doc theo truc z

Phuong trinh chuyen dong: m.r''=q.E + q[vxB]

---

x = -(cos(omega0*t)-cos(omega*t))*E0*q/m/(omega0^2-omega^2)

y = -(omega0*sin(omega*t)-sin(omega0*t)*omega)* E0*q/omega/m/ (omega 0^2-omega^2)

z = 0

Nhap du lieu dau vao m,q,E0,omega0,omega

Nhap m=9*10^-29

Nhap q=10^-9

Nhap E0=10^5

Nhap omega0=100*pi

Nhap omega=200*pi

Gioi han thoi gian [tmin,tmax]

Nhap tmin=1

Nhap tmax=2

x=1/9408126880630107421875*(348449143727040986586495598010130648530944*cos(100*pi*t)-348449143727040986586495598010130648530944*cos(200*pi*t))/pi^2

y=1/9408126880630107421875*(17422457186352049329324779900506532 4265472*sin(200*pi*t)-348449143727040986586495598010130648530944*sin (100*pi*t))/pi^2

Chương 7
NGÔN NGỮ LẬP TRÌNH FORTRAN 9.0

Để mô phỏng một số hệ vật lí ta không chỉ dùng Mathematica, Maple, Matlab mà còn có thể sử dụng Fortran. Ngôn ngữ lập trình Fortran 9.0 có ưu điểm là lập trình, kết nối cơ sở dữ liệu rất mạnh.

§1. CẤU TRÚC CHƯƠNG TRÌNH FORTRAN

1.1. Cấu trúc chung của một chương trình Fortran

Một chương trình Fortran về cơ bản có cấu trúc như sau:

[PROGRAM TenChuongTrinh]

[Cac-cau-lenh-khai-bao]

[Cac-cau-lenh-thuc-hien]

END [PROGRAM [TenChuongTrinh]]

trong đó, chỉ có một câu lệnh bắt buộc trong chương trình Fortran là END. Câu lệnh này báo cho chương trình dịch rằng không còn câu lệnh nào nữa để dịch.

Chú ý:

END [PROGRAM [TenChuongTrinh]] có nghĩa rằng có thể bỏ qua TenChuongTrinh trong câu lệnh END, nhưng nếu có TenChuongTrinh thì từ khoá PROGRAM là bắt buộc.

Cac-cau-lenh-khai-bao là những câu lệnh khai báo biến, hằng,... và kiểu dữ liệu tương ứng của chúng để trình biên dịch cấp phát bộ nhớ, phân luồng xử lí. Cac-cau-lenh-thuc-hien là những câu lệnh xác định quy tắc và trình tự thực hiện tính toán, xử lí để đạt được kết quả.

Trong cấu trúc trên, các mục bắt buộc phải xuất hiện theo trình tự như đã mô tả. Có nghĩa là sau câu lệnh mô tả tên chương trình sẽ là các câu lệnh khai báo, tiếp theo là các câu lệnh thực hiện. Câu lệnh END phải đặt ở cuối chương trình.

1.2. Cấu trúc câu lệnh

Tất cả các câu lệnh, trừ câu lệnh gán (ví dụ Sotien = 1000.0), đều bắt đầu bằng các từ khoá (keyword) như END, PRINT, PROGRAM, và REAL.

Nói chung trên mỗi dòng có một câu lệnh. Trong nhiều trường hợp có thể

xuất hiện nhiều câu lệnh trên một dòng thì chúng phải được cách nhau bởi các dấu chấm phẩy (;).

Để cho rõ ràng, dễ hiểu và dễ kiểm tra ta chỉ nên viết những câu lệnh gán rất ngắn, như: A = 1; B = 1; C = 1.

1.3. Lời chú thích

Mọi kí tự theo sau dấu chấm than (!) (ngoại trừ trong xâu kí tự) là lời chú thích và được chương trình dịch bỏ qua.

Toàn bộ nội dung trên cùng một dòng có thể là lời chú thích hay dòng trắng cũng được dịch như dòng chú thích. Lời chú thích có thể được dùng một cách tuỳ ý để làm cho chương trình dễ đọc.

1.4. Dòng nối tiếp

Nếu câu lệnh quá dài thì có thể được chuyển một phần xuống dòng tiếp theo bằng cách thêm kí hiệu nối dòng (&) vào cuối cùng của dòng trước khi ngắt phần còn lại xuống dòng dưới.

Ví dụ 1:

```
A = 174.6 *   &
        (T - 1981.2) * 3
```

1.5. Kiểu dữ liệu

Fortran 90 định nghĩa 5 kiểu dữ liệu chuẩn, được chia thành hai lớp là lớp các kiểu số (numeric) gồm số nguyên (integer), số thực (real) và số phức (complex) và lớp các kiểu không phải số (non-numeric) gồm kiểu kí tự (character) và kiểu lôgic (logical).

Gắn liền với các kiểu dữ liệu còn có các thuộc tính dữ liệu.

– PARAMETER: Thuộc tính hằng,

– DIMENSION: Thuộc tính mảng,

– ALLOCATABLE: Thuộc tính cấp phát động,

– POINTER: Thuộc tính con trỏ,...

1.5.1. Lớp các kiểu số (Integer, Real, Complex)

Kiểu số nguyên: INTEGER [([KIND=]kind)][,attrs] ::] vname

trong đó: + kind là loại, nhận một trong các giá trị 1, 2, 4 hoặc 8 (đối với UNIX hoặc LINUX).

+ attrs là thuộc tính, nhận một, hoặc nhiều hơn, trong các giá trị PARAMETER, DIMENSION, ALLOCATABLE, POINTER,...

+ vname là danh sách biến hoặc hằng, được viết cách nhau bởi các dấu phẩy.

Ví dụ 1

INTEGER, DIMENSION(:), POINTER :: days, hours

INTEGER(2), POINTER :: k, limit

INTEGER(1), DIMENSION(10) :: min

Kiểu số thực: REAL [([KIND=]kind)][[,attrs] ::] vname

Đối với số thực độ chính xác kép (hay độ chính xác gấp đôi) ta còn có thể sử dụng câu lệnh khai báo:

DOUBLE PRECISION [[,attrs] ::] vname

trong đó: + kind là loại, nhận giá trị 4, 8 hoặc 16 (đối với UNIX hoặc LINUX).

+ attrs là thuộc tính, nhận một, hoặc nhiều hơn, trong các giá trị PARAMETER, DIMENSION, ALLOCATABLE, POINTER,...

+ vname là danh sách biến hoặc hằng, viết cách nhau bởi các dấu phẩy, mô tả chúng.

Ví dụ:

REAL X

X = 123456789.0

PRINT2 '(F30.2)', X

end

ta sẽ nhận được kết quả trên màn hình là:

X = 123456800.00

Kiểu số phức: COMPLEX [([KIND =]kind)] [[,attrs]::] vname

trong đó: + kind nhận giá trị 4 hoặc 8;

+ attrs là một hoặc nhiều thuộc tính, nhận các giá trị PARAMETER, DIMENSION, ALLOCATABLE, POINTER,...;

+ vname là danh sách biến hoặc hằng, viết cách nhau bởi các dấu phẩy.

Số phức được định nghĩa như một cặp có thứ tự của hai số thực được gọi là phần thực và phần ảo.

Ví dụ 3:

COMPLEX (4), DIMENSION (8) :: cz, cq

Khai báo hai biến phức cz và cq, mỗi biến là một mảng gồm 8 phần tử phức tức là 8 cặp số thực, mỗi số thực chiếm 4 byte. Câu lệnh này tương đương với hai câu lệnh sau:

COMPLEX(4) cz, cq

DIMENSION(8) cz, cq

1.5.2. Lớp các kiểu không phải số

Kiểu kí tự: Câu lệnh tổng quát khai báo biến, hằng kiểu kí tự có thể là một trong các cách sau.

Cách 1: CHARACTER (length) vname

trong đó: + length là một số nguyên dương chỉ độ dài cực đại của vname;

+ vname là danh sách tên biến, hằng có kiểu xâu kí tự, viết cách nhau bởi dấu phẩy.

Cách 2: CHARACTER(type[,type...])[attrib[,attrib]...]::vname

với: * type là tham số độ dài và loại, nhận một trong các dạng:

(LEN = type-value)

(KIND = expr)

(KIND = expr, LEN = type-value)

([LEN =] type-value, KIND = expr)

trong đó: + type-value có thể là dấu sao (*), hằng nguyên không dấu, hoặc biểu thức nguyên.

+ expr là biểu thức xác định giá trị hằng nguyên tương ứng với phương pháp biểu diễn kí tự (chẳng hạn, chữ cái Latinh, chữ cái Hy Lạp,...).

* attrib là một hoặc nhiều thuộc tính, viết cách nhau bởi dấu phẩy. Nếu chỉ ra thuộc tính thì sau đó phải sử dụng dấu (::). Các thuộc tính có thể là: ALLOCATABLE, DIMENSION, PARAMETER, POINTER,...

Cách 3: CHARACTER [*chrs] vname [*lengths][(dim)] & [/values/][,vname [*lengths] [(dim)]] [/values/]

trong đó: + chrs là độ dài (cực đại) của các xâu, có thể là một số nguyên không dấu, biểu thức nguyên nằm trong ngoặc đơn, hoặc dấu sao nằm trong ngoặc đơn (*).

+ lengths là độ dài (cực đại) của xâu, có thể là số nguyên không dấu, biểu thức nguyên nằm trong ngoặc đơn, hoặc dấu sao nằm trong ngoặc đơn (*).

+ dim: khai báo mảng, tức vname như là mảng.

+ /values/ là liệt kê các hằng kí tự, tức giá trị của các biến, hằng vname.

Ví dụ 1:

CHARACTER (20) St1, St2*30

CHARACTER wt*10, city*80, ch

CHARACTER (LEN = 10), PRIVATE :: vs

CHARACTER*(*) arg

CHARACTER name(10)*20

CHARACTER(len=20), dimension(10):: plume

CHARACTER(2) susan,patty,alice*12,dotty, jane(79)

CHARACTER*5 word /'start'/

Các khai báo trên đây có ý nghĩa như sau:

Biến St1 có độ dài cực đại bằng 20 kí tự.

Biến St2 có độ dài cực đại bằng 30 kí tự.

Các biến wt, city, ch tương ứng có độ dài cực đại là 10, 80 và 1 kí tự; biến vs có độ dài cực đại bằng 10 kí tự và có thuộc tính PRIVATE.

Biến arg có độ dài không xác định.

Các biến mảng một chiều name, plume mỗi mảng gồm 10 phần tử, mỗi phần tử là một xâu có độ dài cực đại 20 kí tự.

Các biến susan, patty, dotty có độ dài cực đại 2 kí tự.

Biến alice có độ dài cực đại 12 kí tự.

Biến mảng jane gồm 79 phần tử, mỗi phần tử là một xâu dài 2 kí tự.

Biến word dài tối đa 5 kí tự và được khởi tạo giá trị đầu bằng 'start'.

Kiểu lôgic: LOGICAL [([KIND=]kind)] [, attrs ::] vname

trong đó: + kind là độ dài tính bằng byte, nhận các giá trị 1, 2, hoặc 4.

+ attrs là các thuộc tính, có thể nhận một hoặc nhiều giá trị, phân cách nhau bởi dấu phẩy.

+ vname là danh sách các biến, hằng, phân cách nhau bởi dấu phẩy.

Dữ liệu kiểu lôgic chỉ nhận các giá trị .TRUE. hoặc .FALSE.

Ví dụ 2:

LOGICAL, ALLOCATABLE :: flag1, flag2

LOGICAL (2), SAVE :: doit, dont = .FALSE.

LOGICAL switch

1.6. Một số lệnh cơ bản

* *Lệnh gán:* vname = expr

* *Lệnh vào/ra đơn giản:* READ* và PRINT*

** Đọc dữ liệu từ file TEXT:* OPEN

Giả sử ta có file số liệu với tên là SOLIEU.TXT mà nội dung của nó chỉ gồm 3 số ở dòng đầu tiên của file: 3 4 5

Bây giờ ta hãy gõ chương trình sau đây vào máy và chạy thử:

```
PROGRAM ThuFile
REAL A, B, C
OPEN(1, FILE = 'SOLIEU.TXT') ! Mở file
READ(1, *) A, B, C          ! Đọc số liệu từ file
PRINT*, A, B, C
END
```

Câu lệnh OPEN kết nối số 1 với file SOLIEU.TXT trên đĩa. Số 1 này được gọi là UNIT, mang hàm nghĩa chỉ thị số hiệu file (hay kênh vào/ra).

** Lệnh kết xuất dữ liệu:* PRINT*, list

Ví dụ 1:

```
PRINT*
PRINT*, "Can bac hai cua ", 2, 'la', SQRT(2.0)
```

Nếu muốn in ở dạng cầu kì, có quy cách, ta có thể sử dụng lệnh định dạng FORMAT.

Ví dụ: Để in số 123.4567 dưới dạng dấu phẩy tĩnh trên 8 cột, với 2 chữ số sau dấu chấm thập phân, ta có thể viết:

```
X = 123.4567
PRINT 10, X
10 FORMAT( F8.2 )
```

+ Kết xuất ra máy in: Nếu muốn kết xuất ra máy in, ta chỉ cần đặt tham số FILE = 'PRN' trong câu lệnh OPEN và kết hợp với việc sử dụng lệnh WRITE.

Ví dụ 3:

```
OPEN (2, FILE = 'prn' )
WRITE(2, *) 'In ra may in'
PRINT*, 'In ra man hinh'
```

1.7. Một số ví dụ minh hoạ

Ví dụ 1: Chuyển động trong trường trọng lực

```
PROGRAM ChuyenDongThangDung
```

```
! Chuyen dong thang dung xuong duoi trong truong trong luc
IMPLICIT NONE   ! Xóa bỏ quy tắc kiểu ẩn
REAL, PARAMETER :: G = 9.8 ! Gia tốc trọng trường
REAL S          ! Quãng đường (m)
REAL T          ! Thời gian
REAL U          ! Tốc độ ban đầu (m/s)
PRINT*, ' Thoi gian   Quang duong'
PRINT*
U = 60
T = 6
S = U * T - G / 2 * T ** 2
PRINT*, T, S
END PROGRAM ChuyenDongThangDung
```

Ví dụ 2: Giả sử A, B, C là ba đỉnh của một tam giác. Kí hiệu AB, AC, BC là các cạnh của tam giác, Alfa là góc kẹp giữa hai cạnh AB và AC. Cho biết độ dài của các cạnh AB, AC và số đo bằng độ của góc Alfa, có thể tính độ dài của cạnh BC theo công thức:

$$BC^2 = AB^2 + AC^2 - 2.AB.AC.Cos(Alfa)$$

Viết chương trình nhập vào độ dài các cạnh AB, AC và góc Alfa (độ) rồi tính độ dài của cạnh BC.

```
REAL AB, AC, BC, ALFA
REAL PI
PI = 4.0*ATAN (1.0)
PRINT*,'Cho do dai cac canh AB, AC: '
READ*, AB,AC
PRINT*,'Cho so do goc (do) giua AB va AC: '
READ*, ALFA
BC = SQRT (AB**2 + AC**2 – 2*COS(ALFA*PI/180.0))
PRINT*,'Do dai canh BC = ', BC
END
```

§2. CÁC CẤU TRÚC LỆNH LẶP CỦA FORTRAN

Trong phần trước chúng ta đã làm quen với một số câu lệnh của Fortran, như lệnh gán, các lệnh vào/ra đơn giản với READ và PRINT*, lệnh mở file OPEN để nhận dữ liệu từ file TEXT hoặc kết xuất thông tin ra máy in, lệnh định dạng FORMAT,... Với những câu lệnh đó, ta đã có thể viết được một số chương trình đơn giản. Tiếp theo đó trong phần này ta sẽ tìm hiểu thêm các cấu trúc lệnh lặp dùng để viết cho các chương trình tính toán phức tạp hơn.*

2.1. Lệnh chu trình (DO Loops)

Dạng 1:

```
DO m bdk = TriDau, TriCuoi [, Buoc]
   Các–câu–lệnh
m Câu–lệnh–kết–thúc
```

Dạng 2:

```
DO m bdk = TriDau, TriCuoi [, Buoc]
   Các–câu–lệnh
m CONTINUE
```

Dạng 3:

```
DO bdk = TriDau, TriCuoi [, Buoc]
   Các–câu–lệnh
END DO
```

trong đó: + bdk, TriDau, TriCuoi, Buoc phải có cùng kiểu dữ liệu.

+ m là nhãn của câu lệnh kết thúc chu trình, trong trường hợp không thể sử dụng câu lệnh kết thúc như vậy, có thể thay thế nó bằng câu lệnh m CONTINUE như ở dạng 2. Nếu TriDau < TriCuoi thì Buoc phải là một số dương, ngược lại nếu TriDau > TriCuoi thì Buoc phải là một số âm.

Ví dụ 1: Tính tổng các số nguyên liên tiếp từ N1 đến N2, trong đó N1 và N2 được nhập vào từ bàn phím.

```
INTEGER N1, N2, TONG, I
PRINT '(A\)', ' CHO GIA TRI N1, N2 (N1<=N2):'
READ*, N1, N2
TONG = 0
DO I = N1,N2,1
  TONG = TONG + I
```

```
  PRINT*, I
ENDDO
PRINT '(" TONG=",I5)', TONG
END
```

Khi chạy chương trình, các số nguyên liên tiếp từ N1 đến N2 sẽ được hiện lên màn hình và cuối cùng là thông báo kết quả tổng của các số từ N1 đến N2.

Các câu lệnh PRINT '(A\)', ' CHO GIA TRI N1, N2 (N1<=N2):' Và PRINT '(" TONG=",I5)', TONG đã chứa trong đó lệnh định dạng FORMAT.

2.2. Lệnh rẽ nhánh với IF

Trong Fortran, cấu trúc rẽ nhánh được cho khá đa dạng. Sau đây ta sẽ lần lượt xét từng trường hợp.

Dạng 1: IF (BThuc–Logic) Câu–lệnh

trong đó Câu–lệnh là một câu lệnh thực hiện nào đó và không thể là một trong các câu lệnh có cấu trúc khác, như IF, DO,... BThuc–Logic là điều kiện rẽ nhánh. Tác động của câu lệnh IF là, nếu BThuc–Logic nhận giá trị .TRUE. (đúng) thì chương trình sẽ thực hiện Câu–lệnh ngay sau đó, ngược lại, nếu BThuc–Logic nhận giá trị .FALSE. (sai) thì Câu–lệnh sẽ bị bỏ qua và chương trình tiếp tục với những câu lệnh khác sau IF.

Ví dụ 1. Hãy đọc vào một số và cho biết đó là số dương, số âm hay số 0.

```
! Vi du ve lenh re nhanh
REAL X
PRINT '(A\)',' CHO MOT SO:'
READ*, X
IF (X > 0) PRINT *, ' DAY LA SO DUONG'
IF (X < 0) PRINT *, ' DAY LA SO AM'
IF (X == 0) PRINT *, ' DAY LA SO 0'
END
```

Dạng 2:

```
IF (BThuc–Logic) THEN
  Các–câu–lệnh
END IF
```

Về nguyên tắc, tác động của câu lệnh này hoàn toàn giống với cấu trúc dạng 1. Sự khác nhau giữa chúng chỉ là ở chỗ, trong cấu trúc dạng 1, khi điều kiện được

thỏa mãn (Bthuc – Logic nhận giá trị .TRUE.) thì chỉ có một câu lệnh sau IF được thực hiện, còn trong tdạng 2, nếu BThuc–Logic nhận giá trị .TRUE. thì có thể có nhiều câu lệnh nằm giữa IF... THEN và END IF sẽ được thực hiện.

Ví dụ 2: Viết chương trình nhập vào hai số thực, nếu chúng đồng thời khác 0 thì tính tổng, hiệu, tích, thương của chúng.

```
REAL X, Y, TONG, HIEU, TICH, THUONG
PRINT*, ' CHO 2 SO THUC:'
READ*, X, Y          ! Đọc các số X, Y từ bàn phím
IF (X /= 0.AND.Y /= 0) THEN
                ! X và Y đồng thời khác 0
  TONG = X + Y
  HIEU = X – Y
  TICH = X * Y
  THUONG = X / Y
  PRINT*,' TONG CUA ',X,' VA ',Y,' LA:',TONG
  PRINT*,' HIEU CUA ',X,' VA ',Y,' LA:',HIEU
  PRINT*,' TICH CUA ',X,' VA ',Y,' LA:',TICH
  PRINT*,' THUONG CUA ',X,' VA ',Y,' LA:',&
      THUONG
END IF
IF (X == 0.OR.Y == 0) THEN  ! Một trong hai số = 0
  PRINT*,' MOT TRONG HAI SO VUA NHAP = 0'
END IF
END
```

Dạng 3:

```
IF (BThuc–Logic) THEN
  Các–câu–lệnh–1
ELSE
  Các–câu–lệnh–2
END IF
```

Khác với hai cấu trúc trên, trong cấu trúc dạng 3, việc thực hiện chương trình có thể rẽ về một trong hai "nhánh": Nếu BThuc–Logic nhận giá trị .TRUE. thì chương trình sẽ thực hiện Các–câu–lệnh–1, ngược lại, chương trình sẽ thực hiện Các–câu–lệnh–2.

Ví dụ 3. Viết chương trình nhập vào từ bàn phím ba số thực. Nếu ba số đó thỏa mãn điều kiện là ba cạnh của một tam giác thì tính diện tích của tam giác. Ngược lại thì đưa ra thông báo "BA SO NAY KHONG PHAI LA 3 CANH CUA TAM GIAC".

```
PROGRAM TAM-GIAC
REAL A,B,C  ! Ba số sẽ nhập vào
REAL P,S    ! Nửa chu vi và Diện tích
LOGICAL L1
LOGICAL L2
PRINT*, ' CHO 3 SO THUC:'
READ*, A,B,C
L1 = A>0.AND.B>0.AND.C>0  ! Ba số cùng Dương
L2 = A+B>C.AND.B+C>A.AND.C+A>B
      ! Ba số phải thỏa mãn bất đẳng thức tam giác
IF (L1.AND.L2) THEN ! Thỏa mãn điều kiện Tam giác
  P = (A+B+C)/2
  S = SQRT(P*(P-A)*(P-B)*(P-C))
  PRINT*,' DIEN TICH TAM GIAC = ',S
ELSE   ! Không thỏa mãn điều kiện Tam giác
  PRINT*,"BA SO NAY KHONG PHAI LA 3 CANH &
      &CUA TAM GIAC"
END IF
END
```

Dạng 4:

```
IF (BThuc-Logic-1) THEN
  Các-câu-lệnh-1
ELSE IF (BThuc-Logic-2) THEN
  Các-câu-lệnh-2
ELSE IF (BThuc-Logic-3) THEN
  Các-câu-lệnh-3
  ...
ELSE
  Các-câu-lệnh-n
```

END IF

Trước hết, chương trình sẽ kiểm tra BThuc–Logic–1.

* Nếu BThuc–Logic–1 nhận giá trị .TRUE. thì Các–câu–lệnh–1 sẽ được thực hiện;

* Nếu BThuc–Logic–1 nhận giá trị .FALSE. thì chương trình sẽ kiểm tra đến BThuc–Logic–2.

+ Nếu BThuc–Logic–2 nhận giá trị .TRUE. thì Các–câu–lệnh–2 sẽ được thực hiện;

+ Nếu BThuc–Logic–2 nhận giá trị .FALSE. thì chương trình sẽ kiểm tra BThuc–Logic–3,...

Quá trình cứ tiếp diễn như vậy cho đến khi nếu tất cả các BThuc–Logic đều nhận giá trị .FALSE. thì chương trình sẽ thực hiện Các–câu–lệnh–n. Nếu Các–câu–lệnh–k ở giai đoạn k nào đó của quá trình đã được thực hiện, chương trình sẽ thoát khỏi cấu trúc IF và chuyển điều khiển đến những câu lệnh ngay sau END IF, ngoại trừ trường hợp trong Các–câu–lệnh–k có lệnh chuyển điều khiển GOTO đến một vị trí khác trong chương trình.

Ví dụ 4: Viết chương trình nhập điểm trung bình chung học tập (TBCHT) của một sinh viên và cho biết sinh viên đó được xếp loại học tập như thế nào, nếu tiêu chuẩn xếp loại được quy định như sau: Loại xuất sắc nếu TBCHT >=9; Loại giỏi nếu 9<TBCHT<=8; Loại khá nếu 8<TBCHT<=7; Loại trung bình nếu 7<TBCHT<=5 và loại yếu nếu TBCHT<5.

```
PROGRAM XEPLOAI–1
INTEGER DIEM
WRITE (*,'(A\)') ' CHO DIEM TBCHT: '
READ*, DIEM
IF (DIEM < 0.OR.DIEM > 10) THEN
  PRINT*, 'DIEM KHONG HOP LE'
  STOP
  ! Dừng chương trình nếu điểm không hợp lệ
ELSE IF (DIEM >= 9) THEN
  PRINT*,' LOAI XUAT SAC'
ELSE IF (DIEM >= 8) THEN
  PRINT*,' LOAI GIOI'
ELSE IF (DIEM >= 7) THEN
```

```
    PRINT*,' LOAI KHA'
  ELSE IF (DIEM >= 5) THEN
    PRINT*,' LOAI TRUNG BINH'
  ELSE
    PRINT*,' LOAI YEU'
  END IF
  END
```

2.3. Lệnh nhảy vô điều kiện GOTO

Cú pháp: GOTO m

trong đó m là nhãn của một câu lệnh nào đó sẽ được chuyển điều khiển tới trong chương trình. Khi gặp lệnh 'GOTO m', ngay lập tức chương trình sẽ chuyển điều khiển tới câu lệnh có nhãn m.

Ví dụ 1: Ta có một đoạn chương trình

```
IF (L1) THEN
 I = 1
 J = 2
ELSE IF (L2) THEN
 I = 2
 J = 3
ELSE
 I = 3
 J = 4
END IF
```

Có thể được thay thế bởi đoạn chương trình sau nếu sử dụng lệnh GOTO

```
IF (.NOT.L1) GOTO 10
 I = 1
 J = 2
 GOTO 30
10 IF (.NOT.L2) GOTO 20
 I = 2
 J = 3
 GOTO 30
```

```
20 I = 3
   J = 4
30 CONTINUE
```

2.4. Lệnh IF số học

Cú pháp: IF (BThuc–SoHoc) m1, m2, m3

trong đó BThuc–SoHoc là một biểu thức số học, có thể có kiểu nguyên hoặc thực; m1, m2, m3 là nhãn của các câu lệnh có trong chương trình.

Ví dụ 1: Nhập vào một số nguyên và xác định xem số đó nhỏ hơn, lớn hơn hay bằng 50.

```
INTEGER N
PRINT*, ' CHO MOT SO NGUYEN '
READ*, N
IF (N–50) 10, 20, 30
10 PRINT*,' SO NAY NHO HON 50'
   GOTO 40
20 PRINT*,' SO NAY BANG 50'
   GOTO 40
30 PRINT*,' SO NAY LON HON 50'
40 CONTINUE
END
```

Nếu ta chỉ quan tâm đến việc số nhập vào có lớn hơn 50 hay không, thì cấu trúc rẽ nhánh chỉ cần chuyển điều khiển đến hai nhánh. Khi đó chương trình được viết lại thành:

```
INTEGER N
PRINT*, ' CHO MOT SO NGUYEN '
READ*, N
IF (N–50) 10, 10, 30
10 PRINT*,' SO NAY NHO HON HOAC BANG 50'
   GOTO 40
30 PRINT*,' SO NAY LON HON 50'
40 CONTINUE
END
```

2.5. Kết hợp DO và IF

Cú pháp tổng quát của các cấu trúc này như sau.

Dạng 1: Cấu trúc IF nằm trong chu trình DO:

```
DO bdk = TriDau, TriCuoi, Buoc
  ...
  IF (BThuc-Logic) THEN
    ...
  END IF
  ...
END DO
```

Dạng 2: Chu trình DO nằm trong cấu trúc IF:

```
IF (BThuc-Logic) THEN
  ...
  DO bdk = TriDau, TriCuoi, Buoc
    ...
  END DO
  ...
END IF
```

Ví dụ 1: Chương trình sau đây sẽ kết thúc sau 20 lần lặp, mặc dù số lần lặp được quy định bởi lệnh chu trình là 10000:

```
PROGRAM IFinDO !C.trúc IF nằm trong C.trình DO
Do i=1,10000
  If (mod(i,2)==0) write(*,*) i
  If (i == 20) then
    print*,' Ket thuc sau lan lap thu ',i
    stop
  End If
Enddo
END
```

2.6. Rẽ nhánh với cấu trúc SELECT CASE

```
SELECT CASE (BThuc-Chon)
    CASE (Chon1)
```

```
        Các–câu–lệnh–1
    CASE (Chon2)
        Các–câu–lệnh–2
    ...
    CASE DEFAULT
        Các–câu–lệnh–n
END SELECT
```

trong đó BThuc–Chon, Chon1, Chon2,... phải có cùng kiểu dữ liệu số nguyên, lôgic hoặc CHARACTER*1. BThuc–Chon là biểu thức được tính toán, nó còn được gọi là chỉ số chọn. Chon1, Chon2,... là các giá trị hoặc khoảng giá trị có thể có của BThuc–Chon.

Nếu có nhiều giá trị rời rạc, chúng phải được liệt kê cách nhau bởi các dấu phẩy. Nếu là khoảng giá trị liên tiếp, chúng phải được biểu diễn bởi hai giá trị đầu và cuối khoảng, phân cách nhau bằng dấu hai chấm (:). Các–câu–lệnh–1, Các–câu–lệnh–2,... là tập các câu lệnh thực hiện. Đoạn chương trình trên có ý nghĩa như sau:

Nếu giá trị của Bieu–Thuc–Chon thuộc tập (Chon1) thì Thực hiện Các–câu–lệnh–1

Nếu giá trị của Bieu–Thuc–Chon thuộc tập (Chon2) thì Thực hiện Các–câu–lệnh–2

...

Nếu Bieu–Thuc–Chon nhận các giá trị khác thì thực hiện Các–câu–lệnh–n

Kết thúc.

Ví dụ 1: Viết chương trình xem số ngày của một tháng nào đó trong năm.

```
INTEGER Month, Year
Print'(A\)',' Xem so ngay cua thang nao?'
Read*, Month
SELECT CASE (Month)
 CASE (1,3,5,7,8,10,12)!Các tháng có 31 ngày
  Print*,' Thang ', Month,' co 31 ngay'
 CASE (4,6,9,11)  ! Các tháng có 30 ngày
  Print*,' Thang ', Month,' co 30 ngay'
 CASE (2)  ! Có 28 hoặc 29 ngày
  Print'(A\)',' Nam nao?'
```

```
  Read*, Year
  IF (Mod(Year,4).EQ.0.AND.  &
   & Mod(Year,100).NE.0.OR. &
   & Mod(Year,400).EQ.0) then ! Năm nhuận
   Print*,' Thang ', Month,' Nam ',&
     Year, ' co 29 ngay'
  Else  ! Năm bình thường
   Print*,' Thang ', Month,' Nam ',&
     Year, ' co 28 ngay'
  End IF
 CASE DEFAULT
  Print*,' Khong co thang ', Month
END SELECT
END
```

Ví dụ 2: Gõ một kí tự và cho biết đó là chữ cái hay chữ số.

```
CHARACTER*1  char
Print*,' Hay go mot ki tu:'
Read*, Char
  SELECT CASE (char)
    CASE ('0':'9')
      WRITE (*, *) "Day la chu so ", Char
    CASE ('A':'Z','a':'z')
      WRITE (*, *) "Day la chu cai ",Char
    CASE DEFAULT
      WRITE (*, *) "Day khong phai chu so,
           &cung khong phai chu cai."
      WRITE (*, *) "Day la ki tu ", Char
  END SELECT
END
```

2.7. Thao tác với hằng và biến kí tự (CHARACTER)

Hằng kí tự là tập hợp các kí tự thuộc bảng mã ASCII, không bao gồm các kí tự điều khiển, lập thành một dãy đặt trong cặp dấu nháy đơn (' ') hoặc dấu nháy kép (" ").

Biến kí tự là biến có kiểu kí tự, được khai báo bởi lệnh CHARACTER. Các hằng và biến kí tự có thể được gộp với nhau để tạo thành một xâu kí tự mới.

Ví dụ 1: Chương trình sau đây sẽ đưa ra lời chào mừng nếu ta gõ tên mình vào khi được hỏi.

```
Program WelCome
CHARACTER *20 Name
Print *,'Ten ban la gi ?'
Read*, Name
Write(*,*) 'Xin chao ban ', Name
END
```

Trong chương trình này, lệnh PRINT in ra một hằng kí tự (Ten ban la gi ?), lệnh READ* đọc giá trị của biến kí tự Name do ta nhập vào, còn lệnh WRITE in ra một hằng kí tự (Xin chao ban) và giá trị của biến kí tự Name tiếp theo đó.

§3. CHƯƠNG TRÌNH CON

3.1. Các chương trình con trong

3.1.1. Hàm trong (Internal FUNCTION)

Hàm trong có thể được khai báo như sau.

```
[KiểuDL][RECURSIVE] FUNCTION TenHam &
      ([Các–đối–số]) [RESULT (TenKetQua)]
 [Các–câu–lệnh–khai–báo]
 [Các–câu–lệnh–thực hiện]
 [TenHam = ...]
END FUNCTION [TenHam]
```

trong đó: + KiểuDL là kiểu dữ liệu mà hàm sẽ trả về. Ta có thể bỏ qua tùy chọn này khi sử dụng tùy chọn RESULT.

+ RECURSIVE là tùy chọn để chỉ hàm là hàm đệ quy.

+ TenHam là tên của hàm, được dùng để gọi tới hàm.

+ Các–đối–số là danh sách các đối số hình thức, liệt kê cách nhau bởi dấu phẩy.

+ TenKetQua là tên biến chứa kết quả trả về của hàm. Nếu sử dụng tùy chọn này thì câu lệnh TenHam = ... không được phép xuất hiện. Ngược lại, nếu

không sử dụng tùy chọn RESULT thì phải có dòng lệnh TenHam = ... để trả về kết quả của hàm.

Hàm có thể được gọi tới bằng cách hoặc gán giá trị hàm cho biến, hoặc hàm tham gia vào biểu thức tính:

TenBien = TenHam ([Các–đối–số])

Ví dụ 1: Câu lệnh

Cx = COS (x) sẽ tính giá trị cosin của x bằng lời gọi hàm COS(x) rồi gán cho biến Cx.

Còn trong câu lệnh

Pi = 4.0 * ATAN (1.0)

giá trị của arctan(1.0) được tính thông qua lời gọi hàm ATAN(1.0), sau đó lấy kết quả nhận được nhân với 4.0 rồi mới gán giá trị của biểu thức cho biến Pi.

Khi xây dựng hàm, Các–đối–số là những đối số hình thức, nhưng khi gọi hàm thì Các–đối–số phải được thay vào đó là danh sách đối số thực.

Ví dụ 2: Hàm YNew được xây dựng với ba đối số hình thức X, Y, A:

```
FUNCTION YNew ( X, Y, A )
  ...
  YNew = ...
END FUNCTION YNew
```

3.1.2. Thủ tục trong (Internal SUBROUTINE)

Về cơ bản cú pháp khai báo thủ tục giống với khai báo hàm, chỉ có một số khác biệt sau:

– Không có giá trị nào được liên kết với tên thủ tục,

– Để gọi tới thủ tục phải dùng từ khóa CALL,

– Từ khóa SUBROUTINE được dùng để định nghĩa thủ tục thay cho từ khóa FUNCTION,

– Hàm không có đối số sẽ được gọi tới bằng cách thêm vào sau tên hàm cặp dấu ngoặc đơn rỗng ( ).

Ví dụ 1: MyFunction(), nhưng nếu thủ tục không có đối số thì khi gọi tới sẽ không cần cặp dấu ngoặc đơn này.

Cú pháp khai báo thủ tục như sau:

```
SUBROUTINE TenThuTuc [( Các–đối–số )]
  [Các–câu–lệnh–khai–báo]
```

[Các–câu–lệnh–thực–hiện]

END SUBROUTINE [TenThuTuc]

Trong đó Các–đối–số là danh sách đối số hình thức, được liệt kê cách nhau bởi dấu phẩy.

Lời gọi thủ tục:

CALL TenThuTuc [(Các–đối–số–thực)]

Trong đó danh sách các đối số hình thức và danh sách các đối số thực cũng phải tương ứng 1–1 với nhau.

Chú ý:

• Hàm là một chương trình con chỉ trả về duy nhất một giá trị: Giá trị của hàm ứng với các đối số. (Sau này ta sẽ thấy hàm có thể trả về nhiều giá trị).

• Trong định nghĩa hàm (FUNCTION), trước khi trả về chương trình gọi, giá trị của hàm luôn được xác định bởi một câu lệnh gán hoặc cho TenHam hoặc cho biến TenKetQua trong tùy chọn RESULT. Đối với các thủ tục thì kết quả có thể sẽ được trả về thông qua danh sách các đối số, cũng có thể là một hoặc một số nhiệm vụ nào đó.

• Hàm (và thủ tục) kết thúc ở câu lệnh END cuối cùng. Tuy nhiên cũng có thể sử dụng câu lệnh RETURN để trả về chương trình gọi. Khi gặp câu lệnh RETURN chương trình con sẽ được giải phóng và quay về chương trình gọi, mà không quan tâm đến việc sau nó có còn câu lệnh thực hiện nào hay không.

3.1.3. Câu lệnh CONTAINS

Câu lệnh CONTAINS là câu lệnh không thực hiện, dùng để phân cách thân chương trình chính (chính xác hơn là đơn vị chương trình) với các chương trình con trong thuộc nó. Các chương trình con trong được sắp xếp ngay sau câu lệnh CONTAINS và trước từ khóa END của chương trình chính. Bố cục tổng quát của chương trình có dạng như sau:

PROGRAM TenChuongTrinh

[Các–câu–lệnh–khai–báo]

[Các–câu–lệnh–thực–hiện]

[CONTAINS

Các–chương–trình–con–trong]

END [PROGRAM [TenChuongTrinh]]

ở đây, Các–chương–trình–con–trong là những hàm trong hoặc thủ tục trong chịu sự quản lí của chương trình TenChuongTrinh.

Ví dụ 1: Trong chương trình sau đây, CT–CHINH sẽ gọi đến chương trình con trong có tên là CT–CON.

```
PROGRAM CT–CHINH
REAL A(10)
...
CALL CT–CON (A)
CONTAINS
  SUBROUTINE CT–CON (B)
  REAL B(10)
  ...
  END SUBROUTINE CT–CON
END PROGRAM CT–CHINH
```

3.1.4. Một số ví dụ về chương trình con trong

Ví dụ 1: Tính tích phân xác định $I = \int_a^b f(x)dx$ bằng phương pháp hình thang.

```
PROGRAM TICHPHAN
INTEGER N, J
REAL S1,S2,DELX
REAL X, F1,F2, SS,EP, HSO
REAL, PARAMETER :: EP=1.E-4, A=0., B=3.
N=0
S1=0
DO
  N=N+1
  DELX = (B-A)/REAL(N)/2.0
  S2=0
  DO J=1,N
    X = A + (J-1)*DELX
    IF (J>1) THEN
      F1 = F2
    ELSE
      F1= F(X)
```

```
        END IF
        X = X + DELX
        F2= F(X)
        S2= S2 + (F1+F2)*DELX
      END DO
      SS = ABS((S2-S1)/S2)
      IF (SS < EP ) EXIT
      S1 = S2
      PRINT*,'SO HINH THANG =',N
    END DO
    PRINT '(" GIA TRI TP = ",F10.4)', S2
    CONTAINS
    , FUNCTION F(X) RESULT (Fr)
      Fr=1.0/SQRT(2.0*(4.0*ATAN(1.)))*EXP(-0.5*X*X)
     END FUNCTION F
    END
```

Ví dụ 2: Giải phương trình f(x) = 0 bằng phương pháp lặp Newton.

```
    PROGRAM Newton
    ! Giai PT f(x) = 0 bang PP Newton
    IMPLICIT NONE
    INTEGER :: Its   = 0   ! Dem lan lap
    INTEGER :: MaxIts = 20  ! So lan lap cuc dai
    LOGICAL :: Converged =.false.! Dieu kien hoi tu
    REAL :: Eps = 1E-6   ! Sai so cho phep
    REAL :: X = 2.       ! Gia tri nghiem khoi tao DO WHILE (.NOT. Converged
.AND. Its < MaxIts)
      X = X - F(X) / DF(X)
      PRINT*, X, F(X)
      Its = Its + 1
      Converged = ABS( F(X) ) <= Eps
    END DO
     IF (Converged) THEN
      PRINT*, 'Hoi tu'
```

```
ELSE
 PRINT*, 'Phan ki'
END IF
PRINT*,'Nghiem PT: X = ',X
CONTAINS
 FUNCTION F(X)
  REAL F, X
  F = X ** 3 + X - 3
 END FUNCTION F
 FUNCTION DF(X)
  REAL DF, X
  DF = 3 * X ** 2 + 1
 END FUNCTION DF
END PROGRAM Newton.
```

3.2. Các chương trình con ngoài

Các chương trình con trong là những chương trình con chỉ do một đơn vị chương trình kiểm soát (chẳng hạn, chương trình chính), chúng khu trú giữa hai câu lệnh CONTAINS và END của đơn vị chương trình.

Các chương trình con tồn tại ở ngoài dưới dạng các file độc lập được gọi là các chương trình con ngoài. Chúng có thể được tham chiếu bởi nhiều đơn vị chương trình khác nhau. Tuy nhiên, các chương trình con ngoài cũng có thể tồn tại ngay trong cùng một file với chương trình chính hoặc các đơn vị chương trình khác nhưng không nằm giữa các câu lệnh CONTAINS và END. Trong trường hợp đó, các đơn vị chương trình chứa trong các file khác sẽ không thể tham chiếu trực tiếp đến chúng được.

Các chương trình con ngoài cũng có thể có các chương trình con trong riêng của chúng. Nhưng các chương trình con trong lại không được phép chứa các chương trình con trong khác.

Cú pháp khai báo các chương trình con ngoài có thể có dạng:

Khai báo hàm:

```
[KiểuDL][RECURSIVE] FUNCTION  TenHam
        ([Các-đối-số]) [RESULT (TenKetQua)]
 [Các-câu-lệnh-khai-báo]
```

[Các–câu–lệnh–thực–hiện]
[CONTAINS
Các–chương–trình–con–trong]
END [FUNCTION [TenHam]]

Khai báo thủ tục:

SUBROUTINE TenThuTuc [(Các–đối–số)]
[Các–câu–lệnh–khai–báo]
[Các–câu–lệnh–thực–hiện]
[CONTAINS
Các–chương–trình–con–trong]
END [SUBROUTINE [TenThuTuc]]

Về cơ bản khai báo chương trình con ngoài tương tự như khai báo chương trình con trong, ngoại trừ các chương trình con ngoài được phép chứa các chương trình con trong, còn các chương trình con trong thì không được phép chứa các chương trình con trong khác. Việc tham chiếu đến các chương trình con ngoài hoàn toàn tương tự như khi tham chiếu đến các chương trình con trong.

Ví dụ 1: Ta có hàm tính tổng hai số nguyên sau đây:

```
INTEGER FUNCTION Tong(a, b) RESULT (X)
Integer a,b
X = a + b
END FUNCTION Tong
```

Khi đó hàm có thể được tham chiếu như sau:

```
PROGRAM GOI–HAM
IMPLICIT NONE
Integer I,j,N,M,Tong
I = 10
J = 23
N = Tong(I, J)
M = N * Tong (N, J)
print*,N, M
END
```

Câu lệnh EXTERNAL:

Để tránh nhầm lẫn trong việc sử dụng các chương trình thư viện của Fortran

và chương trình con ngoài có tên trùng nhau, ta nên khai báo tên các chương trình con ngoài bằng câu lệnh EXTERNAL.

Ví dụ 2: Chương trình sau đây định nghĩa chương trình con ngoài COS(X) có tên trùng với hàm COS(X) của thư viện Fortran. Khi chạy chương trình ta sẽ thấy chương trình con này không được gọi tới, mà thay cho nó, hàm COS(X) của Fortran sẽ được gọi.

```
PROGRAM EXT1
REAL A
PRINT*,'Cho so A:'
READ*, A
A = COS( A )
PRINT*, A
END
FUNCTION COS( X )
  COS = X + 5.
END FUNCTION
```

Ví dụ 3: Cũng chương trình trên đây, nhưng nếu ta thêm câu lệnh EXTERNAL COS vào ngay phần khai báo của chương trình, kết quả là chương trình con ngoài sẽ được gọi tới.

```
PROGRAM EXT2
REAL A
EXTERNAL COS
PRINT*,'Cho so A:'
READ*, A
A = COS( A )
PRINT*, A
END
FUNCTION COS( X )
  COS = X + 5.
END FUNCTION
```

Như vậy, trong quá trình xây dựng chương trình, ta cần phải hết sức thận trọng khi đặt tên cho các chương trình con ngoài. Biện pháp an toàn nhất là nếu không chắc chắn tên chương trình con không trùng với tên của các chương trình khác thì nên khai báo chúng bằng câu lệnh EXTERNAL.

3.3. Khai báo khối giao diện (INTERFACE BLOCK)

Trong nhiều trường hợp trình biên dịch có thể không hiểu ý đồ khi người sử dụng muốn sử dụng những chương trình con ngoài có cùng tên (có thể vô tình) với các chương trình con thư viện của Fortran. Để khắc phục tình trạng đó ta có thể sử dụng câu lệnh EXTERNAL. Câu lệnh này cung cấp cho trình biên dịch tên của chương trình con ngoài, cho phép nó tìm được và liên kết (LINK). Tuy nhiên, để trình biên dịch tạo ra lời gọi các chương trình con ngoài một cách chính xác, ngoài tên thì cần phải biết chắc chắn những thông tin về chương trình con, như số biến và kiểu của biến,... Tập hợp những thông tin đó gọi là phần giao diện của chương trình con.

Đối với các chương trình con trong, chương trình con modul, và các chương trình con thư viện của Fortran, phần giao diện luôn được trình biên dịch hiểu. Nhưng khi trình biên dịch phát sinh lời gọi đến một chương trình con ngoài, những thông tin thuộc phần giao diện hoàn toàn chưa sẵn có, tức nó ở trạng thái ẩn (implicit), và trong nhiều trường hợp phức tạp (như các đối số tùy chọn hoặc các đối số từ khóa) đòi hỏi phải cung cấp những thông tin giao diện đầy đủ hơn. Do đó cần có khối giao diện. Cú pháp khai báo khối giao diện như sau:

```
INTERFACE
  Thân-của-khối-giao-diện
END INTERFACE
```

trong đó Thân-của-khối-giao-diện nên được sao chép một cách chính xác phần đầu (header) của các chương trình con, cũng như những khai báo đối số và kết quả của chúng và cả câu lệnh END của chúng.

Ví dụ 1: Chương trình sau đây sẽ gọi đến một thủ tục ngoài có tên DOI-CHO.

```
IMPLICIT NONE
! Phan khai bao khoi giao dien
INTERFACE
 SUBROUTINE DOI-CHO( X, Y )
 REAL X, Y
 END SUBROUTINE
END INTERFACE
! Phan khai bao bien, hang
REAL A, B
```

```
! Than chuong trinh
PRINT*,' CHO 2 SO:
READ*, A, B
print*,A, B
CALL DOI–CHO ( A, B )
print*,A, B
END
! Chuong trinh con ngoai
SUBROUTINE DOI–CHO ( X, Y ) ! Đổi giá trị của hai biến
REAL X, Y
REAL TMP
  TMP = X
  X = Y
  Y = TMP
END SUBROUTINE
```

§4. MẢNG

4.1. Khái niệm về mảng trong FORTRAN

Là một tập hợp các phần tử có cùng kiểu dữ liệu, được sắp xếp theo một trật tự nhất định, trong đó mỗi phần tử được xác định bởi chỉ số và giá trị của chúng. Chỉ số của mỗi phần tử mảng được xem là "địa chỉ" của từng phần tử trong mảng mà nó được dùng để truy cập/ tham chiếu đến phần tử của mảng. Mỗi phần tử của mảng được xác định bởi duy nhất một "địa chỉ" trong mảng.

Mảng có thể là mảng một chiều hoặc nhiều chiều. Mảng một chiều có thể hiểu là một vectơ mà mỗi phần tử mảng là một thành phần của vectơ. Địa chỉ các phần tử mảng một chiều được xác định bởi một chỉ số là số thứ tự của chúng trong mảng. Mảng hai chiều được hiểu như một ma trận mà địa chỉ các phần tử của nó được xác định bởi hai chỉ số: chỉ số thứ nhất là số thứ tự hàng, chỉ số thứ hai là số thứ tự cột. Tương tự, mảng ba chiều được xem như là tập hợp các mảng hai chiều, trong đó các phần tử mảng được xác định bởi ba chỉ số: chỉ số thứ nhất, chỉ số thứ hai (tương ứng là hàng và cột của một ma trận) và chỉ số thứ ba (lớp – số thứ tự của ma trận),...

Kiểu dữ liệu của các phần tử mảng có thể là kiểu số hoặc không phải số. Mỗi

mảng được xác định bởi tên mảng, số chiều, kích thước cực đại và cách sắp xếp các phần tử của mảng. Tên mảng còn gọi là tên biến mảng, hay ngắn gọn hơn là biến mảng. Biến mảng là biến có ít nhất một chiều.

Mảng có thể là mảng tĩnh hoặc mảng động. Nếu là mảng tĩnh thì vùng bộ nhớ dành lưu trữ mảng là cố định và nó không bị giải phóng chừng nào chương trình còn hiệu lực. Kích thước của mảng tĩnh không thể bị thay đổi trong quá trình chạy chương trình. Nếu mảng là mảng động, vùng bộ nhớ lưu trữ nó có thể được gán, thay đổi và giải phóng khi chương trình đang thực hiện.

Các con trỏ (POINTER) là những biến động. Nếu con trỏ cũng là mảng thì kích thước của mỗi chiều cũng có thể bị thay đổi trong lúc chương trình chạy, giống như các mảng động. Các con trỏ có thể trỏ đến các biến mảng hoặc biến vô hướng.

4.2. Khai báo mảng

Khi khai báo mảng cần phải chỉ ra tên và số chiều của nó, nhưng có thể chưa cần chỉ ra kích thước và cách sắp xếp các phần tử mảng.

Khai báo biến của mảng: Có rất nhiều cách khai báo biến mảng. Sau đây sẽ liệt kê một số trường hợp ví dụ.

```
REAL A(10, 2, 3)     ! Mảng các số thực 3 chiều
DIMENSION A(10, 2, 3) ! Mảng các số thực 3 chiều
ALLOCATABLE B(:, :)   ! Mảng các số thực 2 chiều
POINTER C(:, :, :)    ! Mảng các số thực 3 chiều
REAL,DIMENSION (2,5) :: D ! Mảng các số thực 2 chiều
REAL,ALLOCATABLE :: E(:,:,:)! Mảng thực 3 chiều
REAL, POINTER :: F(:,:)  ! Mảng các số thực 2 chiều
```

Khai báo mảng: Nói chung có thể có nhiều cách khai báo mảng khác nhau tùy thuộc vào yêu cầu và bối cảnh cụ thể. Sau đây là một số dạng cú pháp tổng quát của câu lệnh khai báo mảng thường được sử dụng trong lập trình.

Dạng 1:

Kiểu–DL Tên–biến–mảng (Mô–tả)

Ví dụ 1:

```
REAL*4 X (0:100)
REAL Y(12,34)
```

Dạng 2:

Thuộc–tính Tên–biến–mảng (Mô–tả)

Ví dụ 2:

DIMENSION N (10,20)

ALLOCATABLE Y(:,:)

Dạng 3:

Kiểu-DL, Thuộc-tính (Mô-tả) :: Tên-biến-mảng

Ví dụ 3:

REAL, ALLOCATABLE(:,:) :: X

INTEGER, DIMENSION(12,34) :: Y

Dạng 4:

Kiểu-DL, Thuộc-tính :: Tên-biến-mảng(Mô-tả)

trong đó: + Kiểu-DL là kiểu dữ liệu của các phần tử mảng,

+ Thuộc-tính có thể là một trong các thuộc tính DIMENSION, ALLOCATABLE, POINTER,...,

+ Tên-biến-mảng là tên của các biến mảng (nếu có nhiều hơn một biến thì chúng được liệt kê cách nhau bởi các dấu phẩy),

+ Mô-tả là mô tả số chiều, kích thước mảng và cách sắp xếp các phần tử mảng. Nếu là mô tả ẩn thì cách sắp xếp các phần tử mảng chưa cần chỉ ra trong khai báo biến mảng.

Ví dụ 4:

REAL, ALLOCATABLE :: X (:,:)

REAL, DIMENSION Y(12,34)

4.3. Lưu trữ mảng trong bộ nhớ và truy cập đến các phần tử mảng

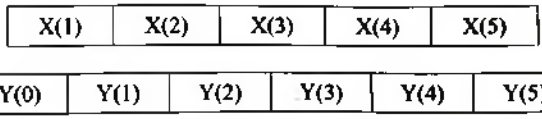
Nguyên tắc lưu trữ mảng trong bộ nhớ của Fortran là lưu trữ dưới dạng vectơ, cho dù đó là mảng một chiều hay nhiều chiều.

Đối với mảng một chiều, các phần tử mảng được sắp xếp theo thứ tự từ phần tử có địa chỉ mảng (chỉ số) nhỏ nhất đến phần tử có địa chỉ lớn nhất. Các phần tử của mảng hai chiều cũng được xếp thành một vectơ, trong đó các "đoạn" liên tiếp của vectơ này là các cột với chỉ số cột tăng dần.

Ví dụ: Đối với mảng một chiều, giả sử ta khai báo

REAL X(5), Y(0:5)

Khi đó các mảng X và Y được sắp xếp trong bộ nhớ như sau:



Chương trình sau đây minh họa cách truy cập đến các phần tử của các mảng này.

```
REAL X(5), Y(0:5)
Y(0) = 1.
DO I=1,5
  X(I) = I*I  ! Gán giá trị cho các phần tử của X
  Y(I) = X(I) + I
     ! Nhận giá trị các phần tử của X, tính toán
     ! và gán cho các phần tử của Y
END DO
PRINT '(6F7.1))', (X(I), I=1,5) !In các phần tử của X
PRINT '(6F7.1))', (Y(I), I=0,5) !In các phần tử của Y
END
```

Khi chạy chương trình này ta sẽ nhận được kết quả trên màn hình là:

```
1.0   4.0   9.0  16.0  25.0
1.0   2.0   6.0  12.0  20.0  30.0
```

§5. XÂY DỰNG CƠ SỞ DỮ LIỆU

Có rất nhiều phần mềm chuyên dụng về cơ sở dữ liệu hiện đang được lưu hành. Mỗi một cơ sở dữ liệu đều có một kiểu cấu trúc dữ liệu riêng. Trong mục này ta sẽ tìm hiểu cách xây dựng một cơ sở dữ liệu bằng ngôn ngữ Fortran.

Việc xây dựng một cơ sở dữ liệu có thể bao gồm các nhiệm vụ sau:

1) Tạo một cấu trúc dữ liệu. Tùy thuộc vào từng mục đích cụ thể cũng như yêu cầu lưu trữ, khai thác thông tin, cấu trúc dữ liệu cần phải bảo đảm các nguyên tắc: rõ ràng, đầy đủ, dễ truy cập và chiếm ít không gian bộ nhớ nhất (cả bộ nhớ trong và bộ nhớ ngoài). Thực chất đây là giai đoạn thiết kế cấu trúc cơ sở dữ liệu.

2) Xây dựng chương trình quản trị, khai thác dữ liệu. Đây là một bộ chương trình cho phép thực hiện các chức năng tạo mới, cập nhật, bổ sung, sửa chữa, hiển

thị và kết xuất thông tin từ cơ sở dữ liệu. Trong nhiều trường hợp bộ chương trình này còn có thể có các chức năng tính toán, xử lí dữ liệu. Để có được một bộ chương trình hoàn thiện, ta cần phải tiến hành việc phân tích, thiết kế chương trình một cách kĩ lưỡng, tỉ mỉ, và có thể cần có cả những thuật toán tối ưu.

3) Tổ chức lưu trữ dữ liệu. Nếu khối lượng dữ liệu lớn, vấn đề tổ chức lưu trữ là rất quan trọng, vì nó liên quan đến sự an toàn, khả năng bảo mật (nếu cần), tính thuận tiện trong truy cập, khai thác,... đây là vấn đề mà người xây dựng cơ sở dữ liệu cần phải tính đến trước khi bắt tay vào thiết kế, xây dựng.

Khi muốn thiết lập một cơ sở dữ liệu về sinh viên, trong đó thông tin đầy đủ của mỗi sinh viên được mô tả bằng một bản ghi. Để đơn giản, ta giả thiết thông tin về mỗi sinh viên chỉ bao gồm họ tên và một số nguyên chỉ điểm thi nào đó. Ví dụ cấu trúc dữ liệu của một bản ghi trong cơ sở dữ liệu được khai báo như sau:

```
TYPE MauHSSV
 CHARACTER (NameLen) Name
 INTEGER  Mark
END TYPE MauHSSV
```

Ta cần phải tạo ra một số chương trình con để đọc và ghi các biến của cấu trúc này đối với file truy cập trực tiếp. Muốn vậy, trước hết cần phác thảo một "khung" chương trình bao gồm chương trình chính và các chương trình con có thể có. Ngoài ra, để tiện trình bày, các chương trình con đều được khai báo như là những chương trình con trong, và file chỉ được mở một lần ngay đầu chương trình chính và sẽ được đóng lại trước khi kết thúc chương trình.

Trong thực tế, các chương trình con nên được tổ chức thành modul với một số khai báo biến toàn cục, như định nghĩa kiểu dữ liệu (TYPE), tên file,... Trong trường hợp đó, mỗi chương trình con khi thao tác với file có thể mở và đóng file ngay trong đó.

```
PROGRAM HO-SO-SINH-VIEN
IMPLICIT NONE
INTEGER, PARAMETER :: NameLen = 20
TYPE MauHSSV
 CHARACTER (NameLen) Name
 INTEGER  Mark
END TYPE MauHSSV
TYPE (MauHSSV) Student
INTEGER  EOF, RecLen, RecNo
```

```
LOGICAL  IsThere
CHARACTER (NameLen) FileName
CHARACTER  Ans
CHARACTER (7) FileStatus
CHARACTER (*), PARAMETER :: NameChars = &
        " abcdefghijklmnopqrstuvwxyz" &
        "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
!
INQUIRE (IOLENGTH = RecLen) Student
WRITE (*, "('Ten File: ')", ADVANCE = "NO")
READ*, FileName
INQUIRE (FILE = FileName, EXIST = IsThere)
IF (IsThere) THEN
  WRITE (*, "('File dang ton tai. &
      Xoa va Thay the? (Y/N)? ')", &
      ADVANCE = "NO")
  READ*, Ans
  IF (Ans == "Y") THEN
    FileStatus = "REPLACE" ! Xóa file và tạo mới
  ELSE
    FileStatus = "OLD"  ! Cập nhật vào file cũ
  END IF
ELSE
  FileStatus = "NEW"  ! File không tồn tại. Tạo file mới.
END IF
OPEN (1, FILE = FileName, STATUS = FileStatus, &
       ACCESS = 'DIRECT', RECL = RecLen)
Ans = ""    ! Khởi tạo giá trị bằng trống rỗng
DO WHILE (Ans /= "Q")
  PRINT*
  PRINT*, "A: Them ban ghi moi"
  PRINT*, "D: Hien thi tat ca cac ban ghi"
  PRINT*, "Q: Thoat"
```

```
    PRINT*, "U: Cap nhat ban ghi dang co"
    PRINT*
    WRITE (*, "('Ban chon? (ENTER): ')", &
          ADVANCE = "NO")
    READ*, Ans
    SELECT CASE (Ans)
      CASE ("A", "a")
        CALL ThemBanGhi
      CASE ("D", "d")
        CALL HienThi
      CASE ("U", "u")
        CALL CapNhat
    END SELECT
  END DO
  CLOSE (1)
  CONTAINS
    SUBROUTINE ThemBanGhi
    ...
    SUBROUTINE HienThi
    ...
    SUBROUTINE DocSoNguyen(Num)
    ...
    SUBROUTINE XoaDauCach(Str)
    ...
    SUBROUTINE CapNhat
    ...
  END
```

Trong chương trình trên:

+ Độ dài trường Name của MauHSSV được khai báo là hằng vì nó sẽ được sử dụng trong các khai báo khác nữa.

+ Biến cơ bản trong chương trình là Student có kiểu dữ liệu MauHSSV.

+ Câu lệnh INQUIRE để xác định độ dài bản ghi cho câu lệnh OPEN sau đó. Chương trình sẽ dùng hội thoại để người sử dụng nhập tên file.

+ Câu lệnh INQUIRE tiếp theo xác định xem file có tồn tại không.

+ Tùy thuộc vào trạng thái tồn tại của file mà tham số STATUS trong lệnh OPEN sẽ mở file cho hợp lí.

+ Đoạn chương trình tiếp theo tạo thực đơn (Menu) dạng đối thoại, cho phép người sử dụng lựa chọn công việc sẽ làm.

+ Trong thực đơn của chương trình người sử dụng có thể gõ kí tự in thường hoặc in hoa. Tuy nhiên, sẽ thuận lợi hơn nếu ta xây dựng thêm một chương trình con đổi chữ thường thành chữ hoa như sau:

```
FUNCTION ChToUpper( Ch )
! Chuong trinh doi chu in thuong thanh chu in hoa
  CHARACTER Ch, ChToUpper
  ChToUpper = Ch
  SELECT CASE (Ch)
    CASE ( "a":"z" )
    ChToUpper = CHAR( ICHAR(Ch) + &
          ICHAR("A") - ICHAR("a") )
  END SELECT
END FUNCTION ChToUpper
```

Chương trình con ThemBanGhi làm nhiệm vụ chèn thêm một bản ghi mới vào cuối file dữ liệu đang tồn tại hoặc chèn vào đầu một file mới.

```
SUBROUTINE ThemBanGhi
RecNo = 0
EOF = 0
DO WHILE (EOF == 0)
  READ( 1, REC = RecNo+1, IOSTAT = EOF )
  IF (EOF == 0) THEN
    RecNo = RecNo + 1
  END IF
END DO
RecNo = RecNo + 1
Student = MauHSSV("a", 0)
DO WHILE ((VERIFY( Student % Name,NameChars )== 0))
  PRINT*, "Cho ten SV: "
```

```
    READ "(A20)", Student % Name
    IF (VERIFY(Student % Name,NameChars ) == 0) THEN
      PRINT*, "Mark: "
      CALL DocSoNguyen(Student % Mark)
      WRITE (1, REC = RecNo) Student
      RecNo = RecNo + 1
    END IF
  END DO
  END SUBROUTINE ThemBanGhi
```

Vì Fortran không có khả năng xác định số bản ghi trong file, nên ta phải thực hiện việc "đọc bỏ qua" các bản ghi để nhận biết số bản ghi này. Cách đọc bỏ qua này không tốn nhiều thời gian. Số bản ghi sẽ được xác định bởi biến RecNo trong vòng lặp DO WHILE. Câu lệnh WRITE sẽ ghi vào file toàn bộ thông tin của một sinh viên chứa trong bản ghi.

– Chương trình con DocSoNguyen được gọi để nhập điểm là một số nguyên hợp lệ.

– Chương trình con HienThi sử dụng vòng lặp DO WHILE để đọc và hiển thị nội dung file.

– Chương trình con CapNhat thực hiện việc tìm tên một sinh viên và sửa đổi nội dung thông tin về sinh viên này.

– Trong trường hợp trên, thông tin chỉ có một trường điểm (Mark).

– Chương trình con XoaDauCach dùng để xóa bỏ các dấu cách (nếu có) trong biến đưa vào để tìm (tên sinh viên) và trường Name của bản ghi để đảm bảo việc so sánh hai xâu kí tự.

```
  SUBROUTINE DocSoNguyen(Num)
  INTEGER Err, Num
  Err = 1
  DO WHILE (Err > 0)
    READ (*, *, IOSTAT = Err) Num
    IF (Err > 0) PRINT*, "Sai du lieu! Vao lai."
  END DO
  END SUBROUTINE DocSoNguyen
  SUBROUTINE HienThi
  RecNo = 1
```

```
  EOF = 0
  DO WHILE (EOF == 0)
    READ (1, REC = RecNo, IOSTAT = EOF) Student
    IF (EOF == 0) THEN
      PRINT "(A20, I3)", Student
    END IF
    RecNo = RecNo + 1
  END DO
  END SUBROUTINE HienThi
SUBROUTINE CapNhat
  CHARACTER (NameLen) Item, Copy
  LOGICAL Found
  Found = .false.
  EOF = 0
PRINT*, "Sua diem cho ai?"
  READ "(A20)", Item
  CALL XoaDauCach( Item )
  RecNo = 1
  DO WHILE (EOF == 0 .AND. .NOT. Found)
    READ (1, IOSTAT = EOF, REC = RecNo) Student
    IF (EOF == 0) THEN
      Copy = Student % Name
      CALL XoaDauCach( Copy )
      IF (Item == Copy) THEN
        Found = .true.   ! Tim thấy
        PRINT*, 'Found at recno', RecNo, &
             ' Enter new mark:'
        CALL DocSoNguyen( Student % Mark )
        WRITE (1, REC = RecNo) Student
                    ! Ghi vào file
      ELSE
        RecNo = RecNo + 1
      END IF
```

```
    END IF
  END DO
  IF (.NOT. Found) THEN
    PRINT*, Item, ' Khong tim thay...'
  END IF
  END SUBROUTINE CapNhat
 SUBROUTINE XoaDauCach( Str )
  CHARACTER (*) Str
  INTEGER I
 I = 1
  DO WHILE (I < LEN-TRIM( Str ))
    IF (Str(I:1) == " ") THEN
      Str(I:) = Str(I+1:)
    ELSE
      I = I + 1
    END IF
  END DO
  END SUBROUTINE XoaDauCach
```

§6. KIỂU FILE

6.1. Khái niệm

Trong hệ thống vào/ra của Fortran, dữ liệu được lưu trữ và chuyển đổi chủ yếu thông qua các file. Tất cả các nguồn vào/ra cung cấp và kết xuất dữ liệu được xem là các file. Các thiết bị như màn hình, bàn phím, máy in được xem là những file ngoài (external files), kể cả các file số liệu lưu trữ trên đĩa. Các biến trong bộ nhớ cũng có thể đóng vai trò như các file, đặc biệt chúng được sử dụng để chuyển đổi từ dạng biểu diễn mã ASCII sang số nhị phân (binary). Khi các biến được sử dụng theo cách này, chúng được gọi là các file trong.

Các file trong hoặc file ngoài đều được liên kết với thiết bị lôgic. Thiết bị lôgic là một khái niệm được sử dụng để tham chiếu đến các file. Ta có thể nhận biết một thiết bị lôgic liên kết với một file bằng định danh (UNIT=). Tại một thời điểm UNIT không thể kết nối với nhiều hơn một file, và file cũng không kết nối với nhiều hơn một thiết bị.

Ví dụ 1:

```
OPEN (UNIT = 10, FILE = 'TEST.dat')
WRITE(10,'(A18,\)' Ghi vao File TEST.dat &
          & da lien ket voi UNIT 10'
WRITE (*, '(1X, A30,\)') ' In ra man hinh.'
```

Fortran ngầm định một số thiết bị chuẩn liên kết với định danh UNIT như sau:

- Dấu sao (*): Màn hình hoặc bàn phím
- UNIT = 0: Màn hình hoặc bàn phím
- UNIT = 5: Bàn phím
- UNIT = 6: Màn hình

6.2. Lệnh mở (OPEN) và đóng (CLOSE) file

6.2.1. Lệnh mở file

Cú pháp lệnh mở file có dạng:

```
OPEN ([UNIT=] unit
    [, ACCESS=access]
    [, ACTION=action]
    [, BLANK=blanks]
    [, BLOCKSIZE=blocksize]
    [, CARRIAGECONTROL=carriagecontrol]
    [, DELIM=delim]
    [, ERR=err]
    [, FILE=file]
    [, FORM=form]
    [, IOFOCUS=iofocus]
    [, IOSTAT=iostat]
    [, PAD=pad]
    [, POSITION=position]
    [, RECL=recl]
    [, SHARE=share]
    [, STATUS=status])
```

trong đó: + UNIT là tham số dùng để khai báo thiết bị lôgic sẽ được liên kết với

file. Nếu bỏ qua 'UNIT=' thì unit cần phải là tham số đầu tiên. Ngược lại, các tham số có thể xuất hiện theo thứ tự bất kì.

+ unit là một số nguyên (INTEGER(4)) lớn hơn hoặc bằng 0, dùng như một thiết bị lôgic để liên kết với file ngoài hoặc thiết bị ngoài.

+ access: chỉ ra cách truy cập vào file, có thể nhận một trong các giá trị "APPEND" (ghi tiếp vào cuối file), "DIRECT" (truy cập trực tiếp), hoặc "SEQUENTIAL" (truy cập tuần tự – ngầm định).

+ action là tham số mô tả tác động dự định đối với file, có thể nhận giá trị 'READ' (file chỉ đọc), 'WRITE' (chỉ để ghi vào file), hoặc 'READWRITE' (cả đọc từ file và ghi vào file). Nếu bỏ qua action, chương trình sẽ cố gắng mở file với 'READWRITE'. Nếu không được, trước hết chương trình sẽ mở file với 'READ', sau đó với 'WRITE'. Việc bỏ qua action không giống với ACTION = 'READWRITE'. Nếu ACTION = 'READWRITE', khi mà file không thể truy cập bằng cả đọc và ghi thì việc mở file sẽ không thành công. Do đó việc bỏ qua action sẽ mềm dẻo và linh động hơn.

+ blanks có dạng kí tự (Character*(*)), dùng để điều khiển cách biểu diễn các kí tự trống (dấu cách) đối với vào/ra có định dạng, nhận một trong các giá trị 'NULL' hoặc 'ZERO'. Giá trị 'NULL' (ngầm định) để bỏ qua kí tự trống, giá trị 'ZERO' để xử lí các kí tự trống (các dấu cách ở đầu) như là những số 0.

+ blocksize ngầm định là một số nguyên ((INTEGER(4)), dùng để biểu diễn kích thước vùng đệm (tính bằng byte).

+ carriagecontrol có dạng kí tự (Character*(*)), dùng để chỉ việc hiểu kí tự đầu tiên của bản ghi trong file có định dạng như thế nào, carriagecontrol nhận một trong các giá trị 'FORTRAN' hoặc 'LIST'.

+ err là nhãn của câu lệnh thực hiện trong chương trình. Khi gặp lỗi mở file chương trình sẽ chuyển điều khiển đến câu lệnh có nhãn err. Nếu bỏ qua, hiệu ứng lỗi vào/ra sẽ được xác định bởi iostat.

+ file có dạng kí tự (Character*(*)), dùng để chỉ ra tên file cần mở, có thể là dấu cách, tên file hợp lệ, tên thiết bị hoặc tên biến xác định file trong. Đối với Windows NT và Windows 9x trở lên, tên file cho phép dài hơn 8 kí tự, phần mở rộng dài hơn 3 kí tự. Nếu file bị bỏ qua, trình biên dịch sẽ tạo một file tạp (file nháp) nào đó chỉ có tên (không có phần mở rộng) và file này sẽ bị xóa khi gặp lệnh đóng file hoặc khi chương trình kết thúc.

+ form xác định kiểu file sẽ được mở, nhận một trong các giá trị 'FORMATTED' (file có định dạng), 'UNFORMATTED' (file không định dạng), hoặc 'BINARY' (file nhị phân).

+ iofocus là tham số lôgic, dùng để chỉ việc có đặt cửa sổ con (child window) là cửa sổ hoạt động hay không. Giá trị .TRUE. (ngầm định) tạo ra lời gọi SETFOCUSQQ ngay lập tức trước khi thực hiện các lệnh READ, WRITE, hoặc PRINT.

+ iostat là tham số kết xuất, ngầm định là một số nguyên (INTEGER(4)), cho giá trị bằng 0 nếu không xuất hiện lỗi mở file, giá trị âm nếu gặp lỗi kết thúc bản ghi, hoặc một số xác định mã lỗi.

+ pad có dạng kí tự (Character*(*)), dùng để chỉ ra việc có đệm các dấu cách vào bản ghi khi đọc hay không nếu định dạng bản ghi có độ dài lớn hơn độ dài dữ liệu. pad nhận giá trị bằng 'YES' (ngầm định) hoặc 'NO'. Tham số này chỉ tác động khi đọc dữ liệu.

+ position có dạng kí tự (Character*(*)), chỉ ra vị trí con trỏ file truy cập tuần tự, nhận một trong các giá trị 'ASIS' (ngầm định), 'REWIND', hoặc 'APPEND'. Nếu position nhận giá trị 'REWIND', con trỏ file sẽ định vị tại đầu file; nếu nhận giá trị 'APPEND', con trỏ file sẽ định vị tại cuối file; nếu nhận giá trị 'ASIS', con trỏ file không thay đổi vị trí (tức giữ nguyên vị trí hiện thời trong file). Vị trí con trỏ file đối với file mới luôn luôn ở đầu file.

+ recl được ngầm định là số nguyên, để chỉ độ dài tính bằng byte của một bản ghi trong file truy cập trực tiếp, hoặc độ dài bản ghi cực đại đối với file tuần tự. Đối với file truy cập trực tiếp đây là tham số đòi hỏi phải có.

+ share có dạng kí tự (Character*(*)), chỉ ra quyền truy cập đối với file được mở. Các giá trị của share có ý nghĩa như sau:

share ='DENYRW': Cấm đọc/ghi

share ='DENYWR': Cấm ghi

share ='DENYRD': Cấm đọc

share ='DENYNONE': Cho phép cả đọc và ghi (ngầm định)

status: Có dạng kí tự (Character*(*)), dùng để mô tả trạng thái của file được mở.

6.2.2. Lệnh đóng file

Cú pháp lệnh đóng file CLOSE có dạng:

```
CLOSE ([UNIT=] unit
[, ERR=err]
[, IOSTAT=iostat]
[, STATUS=status])
```

trong đó: + nếu UNIT= bị bỏ qua thì unit phải là tham số đầu tiên, ngược lại các tham số có thể xuất hiện theo thứ tự bất kì.

+ unit là một số nguyên chỉ thiết bị lôgic liên kết với file được mở. Sẽ không xuất hiện lỗi nếu file không mở.

+ err là nhãn của câu lệnh thực hiện trong chương trình sẽ được chuyển điều khiển tới nếu lỗi xuất hiện khi đóng file. Nếu bỏ qua, hiệu ứng lỗi vào/ra được xác định bởi sự có mặt hoặc không có mặt của tham số iostat.

+ iostat là tham số kết xuất, ngầm định là một số nguyên (INTEGER*4), nhận giá trị bằng 0 nếu không xuất hiện lỗi khi đóng file, hoặc một số chỉ mã lỗi.

+ status là tham số vào, là một biểu thức kí tự (CHARACTER*(*)), nhận các giá trị "KEEP" hoặc "DELETE", ngầm định là "KEEP", ngoại trừ file nháp. Các file được mở không có tham số 'FILE=' được gọi là các file nháp ("scratch" files). Đối với những file này giá trị ngầm định của status là 'DELETE'. Nếu đặt status ='KEEP' đối với những file nháp sẽ gây nên lỗi run-time.

6.3. Các lệnh vào/ ra dữ liệu với file

6.3.1. Lệnh đọc dữ liệu từ file (READ)

Cú pháp lệnh READ làm việc với file có dạng:

```
READ ( [ fmt , ] nml ] |
    ( [UNIT=] unit
      [, [ [FMT=] fmt |
         [NML=] nml ] ]
      [, ADVANCE=advance]
      [, END=end]
      [, EOR=eor]
      [, ERR=err]
      [, IOSTAT=iostat]
      [, REC=rec]
      [, SIZE=size] ) ] iolist
```

trong đó dấu gạch đứng (|) dùng để phân chia các tham số thuộc một nhóm, c nghĩa là chỉ được phép chọn một trong các tham số của nhóm.

Ví dụ 1: Nếu đã chọn '[FMT=] fmt' thì không được phép chọn '[NML= nml'. Nếu bỏ qua 'UNIT=', thì unit phải là tham số đầu tiên. Nếu bỏ qua 'FMT= hoặc 'NML=', thì fmt hoặc nml phải là tham số thứ hai. Nếu muốn sử dụng fr không có 'FMT=', thì phải bỏ qua 'UNIT='. Trong các trường hợp khác các tha

số có thể xuất hiện theo thứ tự bất kì. Hoặc fmt hoặc nml cần được chỉ ra nhưng không phải cả hai.

Sau đây là ý nghĩa các tham số.

+ unit là tên thiết bị lôgic. Khi đọc từ một file ngoài unit là một biểu thức nguyên. Khi đọc từ một file trong unit là một xâu kí tự, một biến kí tự hoặc một phần tử của mảng kí tự,...

+ fmt là chỉ thị định dạng, có thể nhận một trong các trường hợp:

1) Nhãn của lệnh FORMAT,

2) Biểu thức kí tự (biến, thủ tục, hoặc hằng) xác định định dạng đọc vào, phân định bởi các dấu nháy đơn (') hoặc dấu nháy kép (''),

3) Dấu sao (*) đối với trường hợp định dạng tự do,

4) một biến nguyên ASSIGN.

Không được dùng fmt đối với NAMELIST.

Nếu lệnh READ bỏ qua các tùy chọn 'UNIT=', 'END=', 'ERR=', 'REC=', và chỉ có fmt và iolist thì câu lệnh sẽ đọc từ UNIT (*) là bàn phím.

+ nml chỉ tên của NAMELIST. Các tùy chọn iolist và fmt phải được bỏ qua. Lệnh đọc NAMELIST chỉ có thể được thực hiện đối với file truy cập tuần tự.

+ advance có dạng kí tự (Character*(*)), dùng để chỉ ra cách đọc vào từ file tuần tự có định dạng, nhận giá trị hoặc 'YES' (ngầm định) hoặc 'NO'. Nếu advance = 'YES', chương trình sẽ đọc theo định dạng fmt hết bản ghi này sang bản ghi khác và gán giá trị cho iolist. Nếu advance = 'NO', chương trình sẽ đọc các giá trị theo định dạng chỉ trên một dòng và gán cho iolist. Nếu số giá trị chứa trong bản ghi ít hơn số biến trong iolist sẽ xuất hiện lỗi. Yêu cầu tham số định dạng fmt phải khác (*).

+ end là nhãn của câu lệnh trong chương trình sẽ được chuyển điều khiển đến nếu con trỏ file đặt ở cuối bản ghi kết thúc file. Nếu bỏ qua end, sẽ xuất hiện lỗi khi con trỏ file đã ở cuối file nhưng quá trình đọc vẫn cố gắng đọc tiếp, trừ trường hợp có chỉ ra err hoặc iostat.

+ eor là nhãn của câu lệnh trong chương trình sẽ được chuyển điều khiển đến nếu con trỏ file đặt ở cuối bản ghi. Tham số này chỉ dùng khi đưa vào tham số ADVANCE='NO'. Nếu bỏ qua eor, hiệu ứng lỗi vào/ra sẽ được xác định bởi iostat.

+ err là nhãn của câu lệnh trong chương trình sẽ được chuyển đến nếu gặp lỗi trong quá trình đọc. Nếu bỏ qua err, hiệu ứng lỗi vào/ra sẽ được xác định bởi iostat.

+ iostat là tham số kết xuất, ngầm định là một số nguyên (INTEGER(4)).

iostat = 0 nếu không có lỗi, iostat = −1 nếu gặp kết thúc file (end-of-file), hoặc bằng một số chỉ thị thông báo lỗi.

+ rec là tham số vào, ngầm định là một số nguyên dương (INTEGER(4)) chỉ số thứ tự bản ghi cần đọc đối với file truy cập trực tiếp. Sẽ xuất hiện lỗi khi sử dụng tham số này cho file truy cập tuần tự hoặc file trong. Khi sử dụng tham số rec cần bỏ qua các tham số end và nml. Con trỏ file sẽ được định vị đến bản ghi có số thứ tự rec trước khi dữ liệu được đọc. Số thứ tự bản ghi bắt đầu từ 1. Ngầm định của rec là số thứ tự bản ghi hiện thời.

+ size ngầm định là một số nguyên (INTEGER(4)), thực hiện trả về số lượng kí tự được đọc và chuyển cho các biến nhận dữ liệu vào. Các dấu cách chèn đệm vào sẽ không được tính. Nếu sử dụng tham số này thì cần phải đặt tùy chọn ADVANCE='NO'.

+ iolist là danh sách các biến sẽ được nhận dữ liệu từ file.

6.3.2. Lệnh ghi dữ liệu ra file (WRITE)

Cú pháp câu lệnh như sau:

```
WRITE ( [UNIT=] unit
    [, {[ FMT=] fmt  |
       [ NML=] nml}]
    [, ADVANCE=advance]
    [, ERR=err]
    [, IOSTAT=iostat]
    [, REC=rec] ) iolist
```

trong đó: + dấu gạch đứng có ý nghĩa phân cách các tham số trong một nhóm mà chỉ có thể một trong chúng được phép xuất hiện. Nếu bỏ qua UNIT= thì unit phải là tham số đầu tiên và fmt hoặc nml phải là tham số thứ hai (FMT= hoặc NML= có thể được bỏ qua). Ngược lại, các tham số có thể xuất hiện theo thứ tự bất kì. Trong hai tham số fmt và nml chỉ được phép xuất hiện một.

+ unit là tên thiết bị lôgic. Khi ghi ra file ngoài unit là một biểu thức nguyên gắn với định danh UNIT. Khi ghi ra file trong unit phải là xâu kí tự, biến kí tự, mảng hoặc phần tử mảng kí tự,... Nếu unit chưa liên kết với một file cụ thể thì lệnh mở file ẩn (implicit) được thực hiện. *Ví dụ:*

```
OPEN (unit, FILE = ' ', STATUS = 'OLD',&
ACCESS = 'SEQUENTIAL', FORM = form)
```

trong đó form là 'FORMATTED' đối với lệnh đọc/ghi có định dạng hoặc 'UNFORMATTED' đối với lệnh đọc/ghi không định dạng.

+ fmt chỉ thị định dạng, có thể là nhãn câu lệnh FORMAT, biến, hàm hoặc hằng kí tự. Kiểu định dạng được chỉ ra trong các cặp dấu nháy đơn (') hoặc nháy kép ("), biến nguyên ASSIGN hoặc dấu sao (*).

+ nml chỉ ra tên của NAMELIST. Nếu tham số này được chọn thì các tham số iolist và fmt phải được bỏ qua. File chứa NAMELIST phải là file truy cập tuần tự.

+ advance có dạng kí tự (Character*(*)), chỉ ra cách ghi ra file là tiến (advancing) hay không. Nếu advance='YES' (ngầm định) sẽ tạo ra đánh dấu vị trí ở cuối bản ghi. Nếu advance='NO' sẽ ghi một phần của bản ghi (tức chưa tạo ra kết thúc bản ghi).

+ err sẽ chuyển điều khiển nhãn của câu lệnh thực hiện trong chương trình đến khi gặp lỗi. Nếu bỏ qua tham số này, hiệu ứng lỗi vào/ra sẽ được xác định bởi tham số iostat.

+ iostat là tham số kết xuất, ngầm định là một số nguyên (INTEGER(4)), bằng 0 nếu không có lỗi, hoặc bằng một số xác định mã lỗi.

+ rec là tham số vào, ngầm định là một số nguyên dương (INTEGER(4)), chỉ số thứ tự bản ghi trong file sẽ được ghi vào file truy cập trực tiếp. Khi sử dụng tham số rec thì các tham số end và nml cần phải bỏ qua. Con trỏ file phải được định vị tại bản ghi rec trước khi dữ liệu được ghi. Giá trị ngầm định của rec là số thứ tự bản ghi hiện thời.

+ iolist là danh sách các biến sẽ được ghi, liệt kê cách nhau bởi dấu phẩy (,).

6.3.3. Vào / ra dữ liệu với NAMELIST

Vào / ra dữ liệu bằng NAMELIST là một phương pháp rất hữu hiệu của Fortran. Bằng cách chỉ ra một hoặc nhiều biến trong nhóm tên danh sách ta có thể đọc hoặc ghi các giá trị của chúng chỉ với một câu lệnh đơn giản.

Nhóm namelist được tạo bởi câu lệnh NAMELIST có dạng:

NAMELIST / namelist / variablelist

trong đó namelist là tên nhóm và variablelist là danh sách các biến.

Lệnh NAMELIST khi đọc vào sẽ kiểm duyệt tên nhóm trong file NAMELIST. Sau khi tìm thấy tên nhóm nó sẽ kiểm duyệt các câu lệnh gán giá trị cho các biến trong nhóm.

Trong file NAMELIST, các nhóm được bắt đầu bởi dấu và (&) hoặc dấu đôla ($), và kết thúc bằng dấu gạch chéo (/), dấu và (&), hoặc dấu đôla ($).

Từ khóa END có thể xuất hiện liền ngay sau các dấu kết thúc (&) hoặc ($) (không có dấu cách) nhưng không được phép xuất hiện sau dấu gạch chéo (/).

6.4. Ví dụ thao tác với file

Tạo một file có tên file do ta xác định, sau đó đọc nội dung từng bản gh trong file và lần lượt hỏi ta có xóa hay không. Kết quả trung gian được ghi vào một file nháp. Nội dung của file được tạo sẽ được phục hồi lại từ file nháp này.

```
CHARACTER(80) Name, FileName, Ans
WRITE( *, '(A)', ADVANCE = 'NO' ) "Name of file:"
READ*, FileName
OPEN( 1, FILE = FileName )
OPEN( 2, STATUS = 'SCRATCH' )
write (1,'(A)') 'TEST1 Only'
write (1,'(A)') 'TEST2 Only'
write (1,'(A)') 'TEST3 Only'
rewind(1)
IO = 0
DO WHILE (IO == 0)
 READ( 1, '(A)', IOSTAT = IO ) Name
 print*,Name
 IF (IO == 0) THEN
  WRITE(*,'(A)',ADVANCE='NO')"Xoa khong (Y/N)?"
  READ*, Ans
  IF(Ans/='Y'.AND.Ans/='y') WRITE( 2, * ) Name
 END IF
END DO
REWIND( 2 )
CLOSE( 1, STATUS = 'DELETE' )
OPEN( 1, FILE = FileName )
IO = 0
DO WHILE (IO == 0)
 READ( 2, '(A)', IOSTAT = IO ) Name
 IF (IO == 0) WRITE( 1, * ) Name
END DO
CLOSE( 1)
CLOSE( 2)
END
```

§7. MỘT SỐ VÍ DỤ ỨNG DỤNG CỦA FORTRAN TRONG VẬT LÍ

Fortran là một trong những phần mềm tính số mạnh và được ứng dụng rộng rãi trong vật lí tính toán để tính toán và mô phỏng các hệ vật lí như hệ bán dẫn, hệ điện tử tương quan mạnh, hệ nano, tương tác của các hạt cơ bản... Điểm mạnh của Fortran là ngôn ngữ lập trình đơn giản, khả năng kết nối cơ sở dữ liệu rất tốt, tính toán với các bài toán phức tạp cần nhiều vòng lặp. Trong vật lí đôi khi người ta dùng kết hợp Fortran để tính toán lấy dữ liệu với một phần mềm khác (Matlab, Maple, Mathematica...) để vẽ hình, mô phỏng và giải tích hóa các hàm cần tìm. Qua một số ví dụ dưới đây, chúng ta sẽ thấy rõ hơn khả năng ứng dụng của Fortran trong giải các bài toán vật lí và nắm chắc hơn kĩ thuật lập trình tính toán.

7.1. Ví dụ 1: Giải phương trình truyền nhiệt

7.1.1. Bài toán

Xác định nhiệt độ trong một thanh đồng chất có chiều dài L ($0 \le x \le L$), hai đầu mút được giữ ở nhiệt độ bằng không. Biết rằng lúc ban đầu, phân bố nhiệt độ trong thanh có dạng:

$$U(x,0) = \begin{cases} 2x & \text{khi } 0 \le x \le \dfrac{L}{2} \\ 2.(1-x) & \text{khi } \dfrac{L}{2} \le x \le L \end{cases}$$

7.1.2. Lập trình

Phương trình truyền nhiệt của bài toán có thể được viết dưới dạng:

$$\frac{\partial u}{\partial t} - A\frac{\partial^2 u}{\partial x^2} = 0$$

Để lập trình giải bài toán ta giả sử cho L = 1 ($0 \le x \le 1$), h = 0.1 và k = 0.01, khi đó $r = k/h^2 = 1$. Số khoảng chia theo x là NX = 1/0.1 = 10 (i = 0,1,...,10)

Điều kiện biên được chọn là: U(0, t) = U(1, t) = 0 hay $u_{0,j} = u_{NX,j} = 0$ (với m

Điều kiện ban đầu là: $u_{i,0}$ = { 0, 0.2, 0.4, 0.6, 0.8, 1.0, 0.8, 0.6, 0.4, 0.2, 0 }.

Khi đó ta có chương trình tính như sau:

```
PROGRAM PT-Truyen-Nhiet
IMPLICIT NONE
INTEGER N, NT
REAL, ALLOCATABLE :: A(:), B(:), C(:), G(:), U(:,:), UX(:)
```

```
INTEGER I, J
REAL H, K, R, T, L, X
L = 1.  ! Độ dài thanh
K = 0.01 ! Bước thời gian
H = 0.1  ! Khoảng chia theo x
R = K / H ** 2
N = NINT(L/H) ! Số khoảng chia theo x
NT = 100     ! Số bước tích phân
ALLOCATE(A(2:N-1), B(1:N-1), C(1:N-2), &
      G(1:N-1), U(0:N,0:NT), UX(1:N-1))
A = -R       ! Xác định ba đường chéo của A
B = 2 + 2 * R
C = -R
DO I = 0, N ! Điều kiện ban đầu
 X = I*H
 U(I,0) = 2 * X
 IF (X > 0.5) U(I,0) = 2*(1-X)
END DO
U(0,:) = 0.  ! Điều kiện biên
U(N,:) = 0.
T = 0
PRINT "(11F7.4)", (I * H, I = 1, N-1)
DO J = 1, NT  ! Thời điểm tích phân
 G = R * (U(0:N-2,0) + U(2:N,0)) + &
   (2 - 2 * R) * U(1:N-1,0)   ! Vế phải
 G(1) = G(1) + R * U(0,J)
 G(N-1) = G(N-1) + R * U(N,J)
 CALL BaDuongCheo( A, B, C, UX, G )
 U(1:N-1,J) = UX
 U(1:N-1,0) = UX
END DO
DO J=1, NT
  PRINT "(11F7.4)", U(1:N,J)
```

```
ENDDO
CONTAINS
SUBROUTINE BaDuongCheo ( A, B, C, X, G )
IMPLICIT NONE
 REAL B(:)              ! Đường chéo chính
 REAL A(2:)             ! Đường chéo dưới
 REAL C(:)              ! Đường chéo trên
 REAL, INTENT(OUT) :: X(:)  ! ẩn
 REAL G(:)              ! Vế phải
 REAL W( SIZE(B) )      ! Mảng làm việc trung gian
 REAL T
 INTEGER I, J, N
 N = SIZE(B)
 W = B
 DO I = 2, N
  T = A(I) / W(I-1)
  W(I) = W(I) - C(I-1) * T
  G(I) = G(I) - G(I-1) * T
 END DO
 X(N) = G(N) / W(N)
 DO I = 1, N-1
  J = N-I
  X(J) = (G(J) - C(J) * X(J+1)) / W(J)
 END DO
END SUBROUTINE BaDuongCheo
END
```

7.2. Ví dụ 2

7.2.1. Bài toán

Giả sử các giá trị của các thành phần kinh hướng và vĩ hướng của tốc độ gió đã lưu trong file GIOKV.KQ1 thành hai cột, dòng đầu tiên của file ghi số dòng dữ liệu có trong file. Viết chương trình đọc file GIOKV.KQ1 và ghi kết quả tính tốc độ và hướng gió vào file mới GIO.KQ2 thành 4 cột dạng sau:

TT	k V	v V	m/s	HUONG
XX	XX.X	XX.X	XXX	XXX
XX	XX.X	XX.X	XXX	XXX

7.2.2. Lập trình

Khi lập chương trình giải quyết nhiệm vụ này ta nhận thấy cần tính mô đun của tốc độ gió và hướng gió nhiều lần. Do đó nên sử dụng các hàm: để tính tốc độ gió ta có thể dùng loại hàm lệnh, để tính hướng gió ta dùng hàm chương trình con

```
REAL GIOK (200), GIOV (200), V, H, TOCDO, HUONG
INTEGER I, N
%Mô tả hàm lệnh tính mô đun tốc độ gió
TOCDO (VK, VV) = SQRT (VK*VK+VV*VV)
OPEN (1, FILE = 'GIO.KQ1', STATUS = 'OLD')
READ(1,*) N
DO I = 1, N
READ(1,*) GIOK (I), GIOV (I)
END DO
CLOSE (1)
OPEN (1, FILE = 'GIO.KQ2', STATUS = 'UNKNOWN')
WRITE (1, 4) 'TT', 'VK', 'VV', 'M/S', 'HUONG'
FORMAT(1X, I3, 4F7.1)
DO I = 1, N
V = TOCDO (GIOK (I), GIOV (I))
H = HUONG (GIOV (I), GIOK (I))
WRITE (1, 5) I, GIOK (I), GIOV (I), V, H
FORMAT (1X, I3, 4F7.1)
END DO
END
% Hàm chương trình con
FUNCTION HUONG (VV, VK)
REAL VV, VK, HG
IF (VK .EQ. 0.0) THEN
IF (VV .GE. 0.0) THEN
HG = 90.0
```

```
ELSE
 HG = 270.0
ENDIF
ELSE
G = ATAN (ABS (VV / VK)) / 3.14159 * 180.0
 IF (VK .GT. 0.0) THEN
 IF (VV .GE. 0.0) THEN
 HG = G
ELSE
 HG = 360.0 - G
ENDIF
ELSE
 IF (VV .GE. 0.0) THEN
 HG = 180.0 - G
ELSE
 HG = 180.0 + G
ENDIF
ENDIF
ENDIF
 HUONG = HG
RETURN
END
```

7.3. Ví dụ 3

7.3.1. Bài toán

Giả sử có những số liệu quan trắc về nhiệt độ nước biển tại các tầng sâu ở điểm có tọa độ 120^0 KĐ-20°VB được cho trong bảng

Độ sâu	0	5	10	20	30	40	50	60
Nhiệt độ	24,31	24,26	24,20	24,18	24,13	24,05	23,98	23,89
Độ sâu	70	80	90	100	120	140	160	180
Nhiệt độ	23,87	23,57	23,14	22,74	21,31	20,03	18,49	17,58
Độ sâu	200	220	240	260	280	300	350	400
Nhiệt độ	16,66	15,61	14,73	13,97	13,47	12,93	11.40	10,18
Độ sâu	500	600	700	800	900	1000	1200	1400
Nhiệt độ	9,39	8,56	8,49	7,83	7,27	6,71	6.16	5,44

Hãy lập chương trình nhập những số liệu này và nội suy giá trị nhiệt độ c một độ sâu bất kì nhập từ bàn phím, thông báo lên màn hình kết quả nội suy d dạng như sau:

DO SAU = ... M
NHIET DO = ... DO C

7.3.2. Lập trình

```
INTEGER N, I, K
REAL H0, T0, H (40), T (40)
% In lời nhắc và nhập độ sâu cần nội suy nhiệt độ
PRINT * , ' NHAP DO SAU XAC DINH NHIET DO'
READ *, H0
% In lời nhắc và nhập 32 cặp giá trị độ sâu và nhiệt độ
N = 32
K = 1
PRINT *, ' NHAP DO SAU VA NHIET DO TANG ', K
READ *, H(K), T(K)
K = K +1
IF (K .GT. N)  GOTO 4
GOTO 5
% Nội suy giá trị nhiệt độ tại độ sâu H0
I = N - 1
IF (H0 .GT. H(N))  GOTO 1
I = 1
IF (H0 .GE. H (I) .AND. H0 .LE. H (I+1))  GOTO 1
I = I + 1
GOTO 2
T0 = T(I) + (T(I+1)-T (I))*(H0-H(I)) / (H(I+1)-H(I))
PRINT 3, H0
PRINT 6, T0
FORMAT (1X, 'DO SAU =  ', F6.1, ' M')
FORMAT (1X,'NHIET DO =  ', F5.1, ' DO C')
END
```

BÀI TẬP

7.1. Tính tích phân

a) $I = \int_{-\infty}^{+\infty} \frac{\sin 2x}{x^2 - 3x + 1} dx$ b) $I = \int_{-3}^{7} e^{2x} \sin(x^2) dx$

7.2. Mô phỏng dao động tự do của một sợi dây đồng chất hữu hạn có chiều dài L, hai đầu mút được gắn chặt. Biết rằng lúc ban đầu sợi dây nằm trùng với Ox ($0 \le x \le L$) và người ta cung cấp một vận tốc ban đầu cho sợi dây là $u'_t\big|_{t=0} = \sin\frac{x(L-x)}{L^2}$.

7.3. Mô phỏng dao động của một con lắc đơn có khối lượng m, chiều dài L, đặt trong trường trọng lực với g là gia tốc trọng trường và lực cản của không khí tỉ lệ với vận tốc của vật m. Coi dây không giãn trong quá trình dao động và các ma sát bỏ qua được.

TÀI LIỆU THAM KHẢO

1. Tôn Tích Ái. *Phương pháp số*. NXB Đại học Quốc gia Hà Nội.
2. Tạ Văn Đĩnh. *Phương pháp tính*. NXB Giáo dục.
3. Dương Thuỷ Vĩ. *Giáo trình phương pháp tính*. NXB Khoa học và Kĩ thuật.
4. Vũ Văn Hùng. *Phương pháp thống kê momen trong nghiên cứu tính chất n động và đàn hồi của tinh thể*. NXB Đại học Sư phạm.
5. Phan Văn Tân. *Giáo trình Ngôn ngữ lập trình Fortran 90*. NXB Đại Quốc gia Hà Nội.
6. Tôn Tích Ái. *Phần mềm toán cho kĩ sư*. NXB Đại học Quốc gia Hà Nội.
7. Đỗ Đình Thanh – Vũ Văn Hùng. *Phương pháp toán lí*. NXB Giáo dục.
8. Vũ Văn Hùng. *Cơ học lượng tử*. NXB Đại học Sư phạm.
9. Vũ Văn Hùng. *Bài tập cơ học lượng tử*. NXB Đại học Sư phạm.
10. Nguyễn Hữu Mình - Tạ Duy Lợi... *Bài tập vật lí lí thuyết*. NXB Giáo dục.
11. Nguyễn Chính Cương. *Giáo trình Phương pháp toán lí 1*. NXB Đại Sư phạm.
12. Nguyễn Chính Cương – Nguyễn Trọng Dũng. *Giáo trình tin học ứng d* NXB Đại học Sư phạm.
13. Nguyễn Phùng Quang. *Matlab và Simulink dành cho kĩ sư điều khiển tự đ* NXB Khoa học và Kĩ thuật.
14. Nguyễn Việt An: Xử lí ảnh số. *Lí thuyết và thực hành với Matlab*. NXB H Đức.
15. Nguyễn Hoàng Hải – Nguyễn Việt Anh. *Lập trình Matlab và ứng dụng*. N Khoa học và Kĩ thuật.
16. Trần Thị Thục Linh – Đặng Hoài Bắc. *Giải bài tập xử lí tín hiệu số và Ma* NXB Bưu điện.