ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA TOÁN - CƠ - TIN HỌC

Đỗ Văn Chung

Tính toán chuẩn $\mathbf{H}_\infty$ của hệ thống điều khiển bằng phương pháp miền tần số

LUẬN VĂN TỐT NGHIỆP ĐẠI HỌC

Ngành Toán Tin ứng dụng
(Chương trình đào tạo: Chuẩn)

Hà Nội - 2021

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA TOÁN - CƠ - TIN HỌC

Đỗ Văn Chung

Tính toán chuẩn $\mathbf{H}_\infty$ của hệ thống điều khiển bằng phương pháp miền tần số

LUẬN VĂN TỐT NGHIỆP ĐẠI HỌC

Ngành Toán Tin ứng dụng
(Chương trình đào tạo: Chuẩn)

Cán bộ hướng dẫn: TS Hà Phi

Hà Nội - 2021

Lời cảm ơn

Luận văn này được hoàn thành dưới sự hướng dẫn của TS Hà Phi.

Tôi xin chân thành cảm ơn TS Hà Phi đã hướng dẫn và cho những lời khuyên quý báu để tôi có thể hoàn thành bài luận văn này. Tôi cũng chân thành cảm ơn các thầy cô trong Khoa Toán - Cơ - Tin học đã cung cấp và truyền đạt cho tôi những kiến thức và kinh nghiệm quý giá trong quá trình học tập.

Do thời gian và trình độ còn hạn chế nên luận văn của tôi không thể tránh khỏi những thiếu sót. Tôi rất kính mong nhận được sự góp ý của thầy cô và bạn bè để bài luận văn này được hoàn thiện hơn.

Tôi xin chân thành cảm ơn.

Hà Nội, ngày 24 tháng 12 năm 2020,

Sinh viên: Đỗ Văn Chung
Lớp K62 Toán - Tin ứng dụng.

Mục lục

Danh sách hình vẽ

MỞ ĐẦU

Một trong những đòi hỏi đặt ra trong lý thuyết điều khiển chính là việc thiết kế hệ thống điều khiển, giúp cho máy móc có một hiệu suất làm việc hợp lý dưới nhiều điều kiện đầu vào và các yếu tố gây nhiễu khác nhau. Lý thuyết điều khiển H_∞ được sử dụng để có thể giảm các lỗi mô hình hóa và các nhiễu không xác định trong một hệ thống, đồng thời cung cấp khả năng tối ưu hóa một cách định lượng được đối với một bài toán quy mô lớn có nhiều biến tham gia. Trong thực tế hầu hết các nghiệm của bài toán điều khiển H_∞ thực ra chỉ là các bộ điều khiển dưới mức tối ưu, nghĩa là hàm chuyển của bộ điều khiển đạt đến một giới hạn định trước với chuẩn H_∞ . Trong cách thiết lập bài toán này, kết quả là bền vững đối với một giới hạn cho trước. Tuy nhiên, nếu ta tìm thấy chuẩn H_∞ , ta sẽ có câu trả lời cho sự tồn tại của bộ điều khiển trong một phạm vi nhiễu động nhất định đối với hệ thống.

George Zames, [**23**] đưa ra lý thuyết điều khiển H_∞ bằng cách xây dựng việc giảm độ nhạy như một vấn đề trong tối ưu với toán tử chuẩn, cụ thể là chuẩn H_∞ [**15**]. Tại đó H_∞ là không gian của mọi hàm giải tích và bị chặn có giá trị ma trận, bị giới hạn trong nửa bên phải mặt phẳng phức. Cách thiết lập bài toán này hoàn toàn dựa trên miền tần số. Zames gợi ý rằng việc sử dụng chuẩn H_∞ làm thước đo hiệu suất sẽ đáp ứng tốt hơn nhu cầu ứng dụng so với thiết kế điều khiển dạng toàn phương tuyến tính Gauss [**25**]. Từ đó, ta có thiết kế của điều khiển tối ưu H_∞ nhằm mục đích tìm ra một bộ điều khiển có thể ổn định được hệ thống trong khi giảm thiểu tác động của nhiễu gây ra. Chuẩn H_∞ được sử dụng để đánh giá số độ nhạy, độ bền vững và hiệu suất của bộ điều khiển của hệ thống phản hồi vòng kín.

Phương pháp miền tần số là một trong hai phương pháp chính để nghiên cứu H_∞ của hệ thống điều khiển, chính vì vậy trong luận văn này chúng ta sẽ đi tìm hiểu hai thuật toán kinh điển đầu tiên, được trình bày trong các bài báo [**3**], [**4**] để tính toán chuẩn này. Các thuật toán này vẫn còn được tham khảo và trích dẫn cho đến nay (2021).

Chương 1

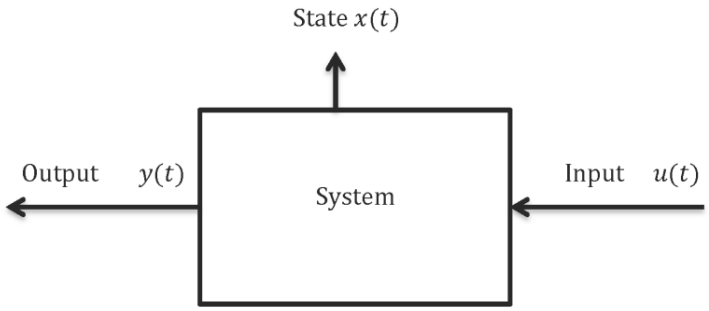
Kiến thức chuẩn bị

1.1 Lập công thức không gian trạng thái

Ta xét hệ thống điều khiển với thời gian liên tục sau

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t),\ x(t_0) = x_0 \\ y(t) &= Cx(t) + Du(t). \end{aligned} \tag{1.1}$$

trong đó A là ma trận cỡ $n \times n$, B cỡ $n \times m$, C là $r \times n$ và D cỡ $r \times m$. Vector x biểu diễn trạng thái, u là vector điều khiển và y là đầu ra. Ta minh họa bằng sơ đồ khối sau:



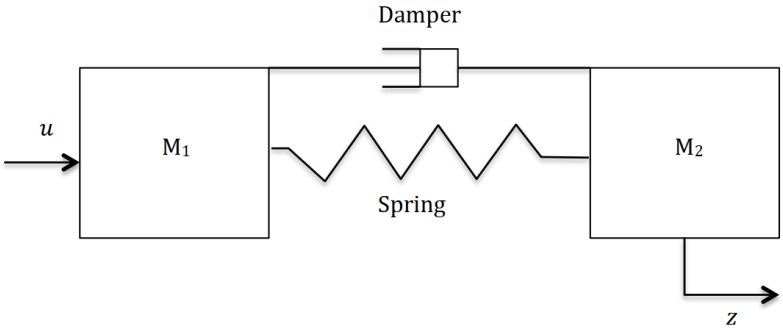
Hình 1.1: Sơ đồ khối mô hình không-thời gian liên tục.

Ví dụ 1.1.1. Để minh họa rõ hơn việc lập công thức không gian trạng thái, ta xét hệ 2 vật nặng được nối với nhau bởi một lò xo và giảm chấn. Lực u tác động vào vật M_1, làm thay đổi vị trí z của vật M_2. Đầu vào của hệ này là lực tác động, đầu ra chính là vị trí của vật M_2. Ở đây thứ ta cần quan tâm chính là việc ta có thể điều khiển được vị trí của M_2.

Áp dụng định luật II Newton, ta có:

$$M_1\ddot{w} = -b(\dot{w} - \dot{z}) - k(w - z) + u, \tag{1.2}$$

$$M_2\ddot{z} = b(\dot{w} - \dot{z}) + k(w - z). \tag{1.3}$$



Hình 1.2: Hệ vật lò xo giảm chấn.

Ở đây, b là hệ số tắt dần, w và z liên tục là sự dịch chuyển M_1 và M_2, k là độ cứng của lò xo. Ta đưa hệ đạo hàm bậc 2 này về bậc 1 bằng cách:
Đặt $x_1 = \text{w}$, $\dot{x_1} = \dot{w} = x_2$, $x_3 = \text{z}$, $\dot{x_3} = \dot{z} = x_4$. Khi đó,

$$\dot{x}_2 = -\frac{b}{m_1}(x_2 - x_4) - \frac{k}{m_1}(x_1 - x_3) + u, \tag{1.4}$$

$$\dot{x}_4 = \frac{b}{m_2}(x_2 - x_4) + \frac{k}{m_1}(x_1 - x_3) + u, \tag{1.5}$$

$$y = x_3. \tag{1.6}$$

Ta viết lại (1.4)-(1.6) hệ dưới dạng ma trận

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k}{m_1} & -\frac{b}{m_1} & \frac{k}{m_1} & \frac{b}{m_1} \\ 0 & 0 & 0 & 1 \\ \frac{k}{m_2} & \frac{b}{m_2} & -\frac{k}{m_2} & -\frac{b}{m_2} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m_1} \\ 0 \\ 0 \end{bmatrix} \cdot u, \tag{1.7}$$

$$y = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}. \tag{1.8}$$

Ta đưa hệ lò xo-giảm chấn này về dạng (1.1) ta có

$$\mathrm{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k}{m_1} & -\frac{b}{m_1} & \frac{k}{m_1} & \frac{b}{m_1} \\ 0 & 0 & 0 & 1 \\ \frac{k}{m_2} & \frac{b}{m_2} & -\frac{k}{m_2} & -\frac{b}{m_2} \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ \frac{1}{m_1} \\ 0 \\ 0 \end{bmatrix} \quad C = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \quad D = 0.$$

1.2 Nghiệm không gian trạng thái

Nếu ta đã biết vector u, hệ (1.1) có thể giải được. Trước khi đi vào cụ thể, ta cần biết một vài định nghĩa [**5**].

Định nghĩa 1.2.1. Cho ma trận A cỡ n × n, khi đó ta định nghĩa **hàm mũ ma trận** là

$$e^{At} = \sum_{k=0}^{\infty} \frac{(At)^k}{k!}.$$

Hai điều lưu ý về hàm mũ ma trận:
1. $e^{A0} = A^0 = I$.
2. $\frac{d}{dt}e^{At} = \sum_{k=0}^{\infty} \frac{d}{dt}\frac{A^k t^k}{k!} = \sum_{k=1}^{\infty} \frac{A^k t^{k-1}}{(k-1)!} = \sum_{k=0}^{\infty} A\frac{A^k t^k}{k!} = Ae^{At}$.
Ta biết nghiệm của bài toán giá trị ban đầu

$$\dot{x}(t) = Ax(t),\ x(t_0) = x_0, \tag{1.9}$$

là x(t) = $e^{A(t-t_0)}x_0$.
Trở lại với phương trình tổng quát với u cho trước,

$$\dot{x}(t) = Ax(t) + Bu(t),\ x(t_0) = x_0 \tag{1.10}$$

$$y(t) = Cx(t) + Du(t). \tag{1.11}$$

ta sẽ tìm nghiệm qua biến đổi Laplace.

Định nghĩa 1.2.2 (Biến đổi Laplace)**.** Hàm f(x), x ∈ [0,∞) có biến đổi Laplace là tích phân $\mathcal{L}(f(x))(s) = \hat{f}(\mathrm{s}) = \int_1^{\infty} f(x)e^{-sx}dx$ với giá trị s ∈ $\mathbb{C}$ mà tích phân có nghĩa.

Hàm Laplace ngược của $\hat{f}$, ký hiệu bởi $\mathcal{L}^{-1}\hat{f}$ = f, là hàm f sao cho biến đổi Laplace của hàm đó chính bằng $\hat{f}$.
Để có thể tìm được nghiệm cho hệ phương trình (1.10) - (1.11), ta biến đổi Laplace phương trình (1.10) và tìm nghiệm $\hat{x}$, cho ta kết quả là

$$\hat{x}(s) = (sI - A)^{-1}x_0 + (sI - A)^{-1}B\hat{u}(s). \tag{1.12}$$

Áp dụng hàm Laplace ngược (1.12) cho ta nghiệm

$$\begin{aligned} x(t) &= L^{-1}((sI-A)^{-1}x_0) + L^{-1}((sI-A)^{-1}B\hat{u}(t)) \\ &= e^{A(t-t_0)}x_0 + \int_{t_0}^{t} e^{A(t-s)}Bu(s)ds. \end{aligned} \tag{1.13}$$

Ta thay phương trình (1.13) vào phương trình (1.11) ta tìm được $y(t) = Ce^{A(t-t_0)}x_0 + \int_{t_0}^{t} Ce^{A(t-s)}Bu(s)ds + Du(t)$.

Định nghĩa 1.2.3. Ma trận $e^{A(t-t_0)}$ được gọi là **ma trận chuyển trạng thái**.

Nếu ta không biết hàm u, ta có nhiều phương án điều khiển khác nhau để tìm ra hệ điều khiển có thể cho ra đầu ra mong muốn.

1.3 Hàm truyền

Hàm truyền sẽ cho ta thấy mối quan hệ giữa đầu vào và đầu ra của hệ thống và sự ảnh hưởng của nhiễu đối với đầu ra như thế nào. Để tìm ra hàm chuyển của hệ (1.1)

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t),\ x(0) = x_0 \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{1.14}$$

ta sử dụng biến đổi Laplace như (1.12) để thu được nghiệm

$$\hat{x}(s) = (sI-A)^{-1}x_0 + (sI-A)^{-1}B\hat{u}(s), \tag{1.15}$$

$$\hat{y}(s) = C\hat{x}(s) + D\hat{u}(s). \tag{1.16}$$

Trừ vế theo vế phương trình (1.15) cho phương trình (1.16) và đặt G(s) = $\frac{\hat{y}(s)}{\hat{u}(s)}$. Phương trình (1.16) trở thành

$$\hat{y}(s) = G(s)\hat{u}(s),$$

hay

$$G(s) = \frac{\hat{y}(s)}{\hat{u}(s)}. \tag{1.17}$$

Hàm G(s) phản ánh tỷ số giữa đầu ra hệ thống $\hat{y}$(s) và đầu vào hệ thống $\hat{u}(s)$ và được gọi là hàm truyền ma trận hoặc đơn giản chỉ là ma trận truyền. Thành phần G_{ij} trong ma trận biểu diễn hàm truyền từ đầu vào thứ j sang đầu ra thứ i. Hàm truyền này đại diện cho các thuộc tính của hệ thống và không đại diện cho độ lớn hoặc tính chất của các đầu vào. Nó cũng không phục vụ cho việc cung cấp bất kỳ thông tin nào về cấu trúc của hệ thống.

Một ưu điểm chính của việc sử dụng hàm truyền là chúng cho phép ta làm việc với một hệ thống phức tạp trong miền thời gian và biểu diễn nó như một hàm của một biến trong miền tần số. Thật không may, cách tiếp cận này chỉ hoạt động đối với các hệ thống đơn đầu ra - đơn đầu vào (SISO). Điều này có nghĩa là, đối với nhiều hệ thống đa đầu vào - đa đầu ra, thì sẽ có nhiều hàm truyền được sử dụng để biểu diễn mỗi quan hệ đầu vào - đầu ra. Cần lưu ý rằng, nhiều hệ thống có thể chia sẻ cùng một hàm truyền, không liên quan đến độ lớn của hệ thống hoặc tính chất của đầu vào.

Định nghĩa 1.3.1. Các điểm p mà tại đó hàm truyền G(p) = ∞ được gọi là các cực của hệ.

Nếu G(∞) là một ma trận hằng thì hàm truyền được gọi là **chính** và nếu G(∞) = 0, hàm truyền được gọi là **chính ngặt**. Chúng ta hãy sử dụng hệ thống giảm chấn khối lượng lò xo từ Ví dụ (1.1) để minh họa cách tìm hàm truyền của một hệ đã cho được mô tả bằng biểu diễn không gian trạng thái.

Ví dụ 1.3.2. Từ Ví dụ (1.1) ta có $A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k}{m_1} & -\frac{b}{m_1} & \frac{k}{m_1} & \frac{b}{m_1} \\ 0 & 0 & 0 & 1 \\ \frac{k}{m_2} & \frac{b}{m_2} & -\frac{k}{m_2} & -\frac{b}{m_2} \end{bmatrix}$, $B = \begin{bmatrix} 0 \\ \frac{1}{m_1} \\ 0 \\ 0 \end{bmatrix}$,

$C = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}$, $D = 0$.

Suy ra

$$G(s) = C(sI-A)^{-1}B+D = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \left(\begin{bmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & s & 0 \\ 0 & 0 & 0 & s. \end{bmatrix} - \begin{bmatrix} 0 & 1 & 0 & 0 \\ -4 & -2 & 4 & 2 \\ 0 & 0 & 0 & 1 \\ 4 & 2 & -4 & -2 \end{bmatrix} \right)^{-1} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}.$$

Giản ước đi ta tìm ra

$$G(s) = \frac{s+2}{s^4+3s^3+6s^2}.$$

Định nghĩa 1.3.3. Trong không gian thời gian - trạng thái biểu diễn ở (1.14), ma trận

$$G(iw) = C(iwI - A)^{-1}B + D \tag{1.18}$$

được gọi là **ma trận phản hồi tần số**, $\omega \in \mathbb{R}$ chính **tần số**.

1.4 Tính điều khiển được và tính quan sát được

Tính điều khiển và tính quan sát được là những khái niệm cơ bản của lý thuyết điều khiển. Để trình bày những khái niệm này, ta tham khảo trong mục [**5**]

Định nghĩa 1.4.1. Hệ (1.1)

$$\dot{x}(t) = Ax(t) + Bu(t),$$
$$y(t) = Cx(t) + Du(t)$$

được gọi là có thể điều khiển được nếu hệ thống đạt được bất kì trạng thái bất kì $x_1 = x(t_1)$, trong thời gian hữu hạn t_1, từ bất kỳ trạng thái ban đầu x(0) bằng cách chọn điều khiển u(t), $0 \leq t \leq t_1$ phù hợp.

Tính điều khiển của hệ thống (1.1), được gọi là tính điều khiển của cặp (A, B). Định lý sau cung cấp các điều kiện có thể kiểm chứng được về việc một hệ thống có thể điều khiển được hay không.

Định lý 1.4.2. *Cho $A \in \mathbb{R}^{n\times n}$ và $B \in \mathbb{R}^{n\times m}(m \leq n)$. Các điều sau đây là tương đương:*

1. *Hệ (1.1) có thể điều khiển được.*
2. *Ma trận điều khiển $n \times nm$ là $C_M = [B, AB, A^2B, ..., A^{n-1}B]$ có đủ hạng dòng.*
3. *Ma trận*

$$W_C = \int_0^{t_1} e^{At} BB^T e^{A^T t} dt$$

   không suy biến với bất kỳ $t_1 > 0$.

4. *Nếu (λ, x) là một cặp giá trị riêng và vector riêng tương ứng của A^T, thì $x^T B \neq 0$.*
5. *Rank(A - λ I, B) = n với mọi giá trị riêng λ của A.*
6. *Các giá trị riêng của A - BK có thể được phân bố tùy ý bằng một lựa chọn thích hợp của K.*

Định nghĩa 1.4.3. Hệ thời gian liên tục (1.1) gọi là **quan sát được** nếu tồn tại $t_1 >$ 0 sao cho trạng thái ban đầu x(t_0) xác định duy nhất nếu biết u(t) và y(t) với mọi $0 \leq t \leq t_1$.

Tính quan sát được của hệ thống (1.1) được gọi là tính quan sát được của cặp (A, C).Tính đối ngẫu có nghĩa là cặp (A, C) có thể quan sát được nếu cặp (A^T, C^T) có thể điều khiển được. Do tính hai mặt của khả năng quan sát và khả năng điều khiển, khả năng quan sát có các đặc điểm tương tự như tính điều khiển được trong Định lý 2.4.1.

Định lý 1.4.4. *Những điều sau đây là tương đương:*

1. *Hệ* (1.1) *có thể quan sát được.*
2. *Ma trận của khả năng quan sát* $O_M = [CCA...CA^{n-1}]^T$ *có hạng đủ.*
3. *Ma trận*

$$W_O = \int_0^{t_0} e^{A^T t} C^T C e^{At} dt$$

   không suy biến với mọi t_1 *dương.*

4. *Ma trận* $\begin{bmatrix} \lambda I - A \\ C \end{bmatrix}$ *có hạng bằng n với mọi giá trị riêng của A.*
5. *Nếu (λ, y) là 1 cặp giá trị riêng và vector riêng của A, thì* $C_y \neq 0$.
6. *Các giá trị riêng cho A - LC có thể được gán tùy ý bằng một lựa chọn thích hợp của L.*

Khả năng điều khiển và khả năng quan sát đóng vai trò quan trọng trong sự tồn tại của các nghiệm xác định dương và nửa xác định dương cho phương trình Lyapunov.

Định nghĩa 1.4.5. Ma trận A được gọi là **ma trận ổn định** nếu mọi giá trị riêng của A có phần thực âm thực sự.

Định nghĩa 1.4.6. Cho A là một ma trận ổn định, thì

$$C_G = \int_0^{t_1} e^{At} B B^T e^{A^T t} dt \tag{1.19}$$

gọi là **Grammian điều khiển.**

Định nghĩa 1.4.7. Cho A là một ma trận ổn định, thì

$$O_G = \int_0^{\infty} e^{A^T t} C^T C e^{At} dt \tag{1.20}$$

được gọi là **Grammian quan sát.**

Định lý 1.4.8. *Cho A là một ma trận ổn định. Grammian điều khiển C_G thỏa mãn phương trình Lyapunov*

$$AC_G + C_G A^T = -BB^T \tag{1.21}$$

và là đối xứng, xác định dương khi và chỉ khi (A, B) có thể điều khiển được.

Định lý 1.4.9. *Cho A là 1 ma trận ổn định. Grammian quan sát O_G thỏa mãn phương trình Lyapunov*

$$O_G A + A^T O_G = -C^T C \tag{1.22}$$

và là đối xứng, xác định dương khi và chỉ khi (A, B) có thể quan sát được.

Sau đây là một ví dụ minh họa tính toán của cả 2 giá trị C_G và O_G.

Ví dụ 1.4.10. Trở lại hệ (1.1), với

$$A = \begin{bmatrix} -4 & -8 & -12 \\ 0 & -8 & -4 \\ 0 & 0 & 0 \end{bmatrix},\ B = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix},\ C = [1 \quad 1 \quad 1],\ D = 0.$$

Ma trận A ổn định và sử dụng lệnh **lyap** trong MATLAB để giải các phương trình Lyapunov trong Định lý 1.4.5 và 1.4.6, chúng ta tìm được

$$C_G = \begin{bmatrix} 0.1458 & 0.0208 & 0.0208 \\ 0.0208 & 0.0833 & 0.0833 \\ 0.0208 & 0.0833 & 0.0833 \end{bmatrix} = O_G.$$

Ta nhận thấy ma trận này có hai dòng bằng nhau nên định thức ma trận bằng 0, suy ra ma trận suy biến, nên (A, B) không điều khiển được và (C, D) không quan sát được.

1.5 Tính ổn định hóa được và khả năng phát hiện được

Một bộ điều khiển thường được thiết kế để ổn định hóa một hệ thống tuyến tính. Điều này được thực hiện bằng cách chọn một điều khiển u thích hợp sao cho ma trận hệ đóng kín $(A - BK)$ ổn định, trong đó K là ma trận phản hồi mà $u(t) = Kx(t)$. Trong phần này, chúng ta xác định tính ổn định hóa được của một hệ thống và trình bày các đặc điểm của hệ thống có thể được ổn định hóa được thông qua điều khiển phản hồi. Đầu tiên chúng ta chọn hệ chưa điều khiển

$$\dot{x}(t) = Ax(t), \quad x(0) = x_0. \tag{1.23}$$

Định nghĩa 1.5.1. Trạng thái cân bằng của hệ (1.23), là vectơ x_e thỏa mãn

$$Ax_e = 0. \tag{1.24}$$

Nếu ma trận A không suy biến, phương trình trên chỉ có nghiệm tầm thường $x_e = 0$.

Định nghĩa 1.5.2. Gọi x_e là trạng thái cân bằng của (1.23), thì x_e được gọi là

1. ổn định nếu với mỗi $\epsilon > 0$ tồn tại $\delta > 0$ sao cho $\|x(t) - x_e\| < \epsilon$, $\forall$ t ≥ 0 nếu $\|x(t_0) - x_e\| < \delta$.

2. ổn định tiệm cận nếu trạng thái cân bằng ổn định và $\exists\delta > 0$ sao cho $\|x(t_0) - x_e\| < 0$ dẫn đến $\lim\limits_{x\to\infty} x(t) = x_e$.

Điều này có nghĩa là mọi nghiệm đúng của hệ nếu bắt đầu đủ gần điểm cân bằng ổn định tiệm cận, thì sẽ tiến gần đến điểm cân bằng khi thời gian tăng. Nên nếu $x_e = 0$ và A không suy biến, hệ không ổn định được ở trên ổn định tiệm cận nếu $x(t) \to 0$ *khi* $t \to \infty$. Các giá trị riêng của ma trận A trong (1.23) đặc trưng cho độ ổn định của hệ thống.

Định lý 1.5.3. *Hệ thống* (1.23) *ổn định tiệm cận khi và chỉ khi mọi giá trị riêng của A có phần thực âm thực sự.*

Cũng như với tính điều khiển, nghiệm của phương trình Lyapunov cũng đặc trưng cho tính ổn định của (1.23).

Định lý 1.5.4. *Hệ:*

$$\dot{x}(t) = Ax(t)$$

ổn định tiệm cận khi và chỉ khi với ma trận M đối xứng dương bất kỳ, thì phương trình sau có nghiệm duy nhất

$$XA + A^T X = -M.$$

Xét hệ tuyến tính (1.1),

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t). \end{aligned}$$

Ta xét việc tìm nghiệm điều khiển u để đưa hệ sang một trạng thái mong muốn. Điều đó có nghĩa là tìm 1 **ma trận phản hồi** K sao cho $A - BK$ ổn định. Không phải mọi hệ có thể đưa về một trạng thái nhất định, nên điều quan trọng nhất của phần này là tìm ra điều kiện cần và đủ sao cho tồn tại một ma trận phản hồi ổn định.

Hơn nữa giả sử ta có thể tìm được mọi thông tin của trạng thái *x(t)* ở mọi thời gian t bất kỳ. Sử dụng điều khiển tuyến tính, ta suy ra

$$u(t) = -Kx(t). \tag{1.25}$$

Trừ vế theo vế phương trình *u(t) = -Kx(t)* cho phương trình (1.1) ta được **hệ kín**:

$$\dot{x}(t) = (A - BK)x(t), \tag{1.26}$$
$$y(t) = (C - DK)x(t). \tag{1.27}$$

Ta nhắc lại hệ không điều khiển được (1.23) ổn định khi ma trận A ổn định. Vậy nên việc ổn định hóa hệ (1.26) quy về việc tìm ma trận K sao cho $(A - BK)$ là ổn định. Nếu tồn tại một ma trận K, thì K được gọi là **ma trận phản hồi** và ổn định hóa được, cặp ma trận (A, B) được gọi là **cặp ma trận ổn định hóa được**.
Những định lý sau đây cho ta điều kiện cần và đủ để đánh giá việc có thể ổn định được ma trận cho 1 hệ đang xét.

Định lý 1.5.5. *Những khẳng định sau là tương đương:*

1. *Cặp ma trận (A, B) là ổn định hóa được.*
2. *Rank(A - λI, B) = n, với mọi Re(λ) $\geq$ 0.*
3. *Với mọi λ và $x \neq 0$, nếu $x^*A = \lambda x^*$ và Re(λ) $\geq$ 0 thì $x^*B \neq 0$.*

Hệ quả 1.5.6. *Nếu cặp (A, B) là điều khiển được, thì nó cũng ổn định hóa được.*

Lưu ý khả năng điều khiển suy ra khả năng ổn định chỉ là chiều suy ra mà không suy ngược lại.
Cũng như khả năng quan sát được đối ngẫu với khả năng điều khiển được, khả năng ổn định được của hệ thống cũng đối ngẫu với khả năng phát hiện được.

Định nghĩa 1.5.7. Hệ được gọi là **phát hiện được** nếu tồn tại ma trận L sao cho với cặp ma trận (A, C) thì A - LC là ổn định.

Định lý 1.5.8. *Các điều sau đây là tương đương:*

1. *Cặp ma trận (A, C) là phát hiện được.*
2. *Ma trận* $\begin{bmatrix} \lambda I - A \\ C. \end{bmatrix}$ *có hạng đầy đủ, với mọi Re(λ) $\geq$ 0.*
3. *Với mọi λ và $x \neq 0$ sao cho $Ax = \lambda x$ và Re(λ) $\geq$ 0, thì $Cx \neq 0$.*
4. *Cặp ma trận (A^T, C^T) là ổn định hóa được.*

Có thể thấy rằng một hệ thống có thể điều khiển được thì các giá trị riêng của hệ điều khiển đóng kín có thể được gán tùy ý (bài toán gán giá trị riêng). Vấn đề với việc chỉ định các giá trị riêng tùy ý đó là không có phương pháp luận để tìm một bộ điều khiển tối ưu sao cho các đặc điểm thiết kế hệ thống có thể đáp ứng một cách hiệu quả nhất có thể.
Điều này dẫn đến nhiều phương pháp điều khiển tối ưu ví dụ như điều khiển tối ưu tuyến tính bậc hai. Phương pháp này tối thiểu hóa hàm phạt bậc 2 đã được xác định trước. Tuy nhiên, lưu ý [**7**] cho ta sự phát triển song song của điều khiển H_∞ và điều khiển H_2 tối ưu hay còn được gọi là điều khiển dạng toàn phương tuyến tính Gauss.

1.6 Chuẩn H_∞

Trước tiên, hãy lưu ý rằng hàm truyền của một hệ thống ổn định bất kỳ được mô hình hóa bởi phương trình vi phân tuyến tính hệ số hằng là hàm hữu tỉ và có các cực nằm hoàn toàn trong nửa mặt phẳng bên trái và là giải tích trong nửa mặt phẳng bên phải. Một không gian của các hàm như vậy được gọi là không gian Hardy, ký hiệu là H_∞ . Đối với hệ đa biến bất kỳ, hàm truyền G(iω) là một ma trận. Các giá trị kỳ dị của ma trận A, σ_j(A) được định nghĩa là

$$\sigma_j(A) = \sqrt{\lambda_j(AA^T)},$$

trong đó λ_j(E) là giá trị riêng thứ j của ma trận E.

Chuẩn Euclid của ma trận là:

$$\|A\| = \max_i \sigma_i(A).$$

Suy ra

$$\|G(i\omega)\| = \max_i \sigma_i(G(i\omega)) = \sigma_{max}(G(i\omega)).$$

Định nghĩa 1.6.1. Chuẩn H_∞ của hàm truyền G(s) được định nghĩa là

$$\|G\|_\infty = \sup_{\omega\in\mathbb{R}} \sigma_{max}(G(i\omega)). \tag{1.28}$$

Chuẩn này cung cấp một cách khác để phân tích tính ổn định của hệ thống, cụ thể là nếu G(s) $\in H_\infty$ thì hệ thống ổn định. Như được mô tả trong [**15**] cho trường hợp khi G là hữu tỉ, G $\in H_\infty$ khi và chỉ khi G không có cực trong nửa mặt phẳng đóng bên phải.

Chương 2

Thuật toán: Miền tần số

2.1 Tính toán chuẩn H_∞

Trong phần này, hai thuật toán để tìm chuẩn H_∞ của một hệ thống điều khiển tuyến tính được trình bày. Xấp xỉ chuẩn H_∞ của một hệ thống yêu cầu tính toán supremum của phản hồi trên tất cả tần số, do đó việc sử dụng quy trình lặp lại là bắt buộc. Mỗi liên hệ giữa chuẩn H_∞ của một hàm truyền ổn định (tức là không có cực trong nửa mặt phẳng bên phải) và ma trận Hamilton của nó đóng một vai trò quan trọng trong thuật toán cho trường hợp phản hồi trạng thái được trình bày sau đây.
Xét hệ

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t). \end{aligned} \tag{2.1}$$

Định nghĩa 2.1.1. Ta định nghĩa ma trận Hamilton M_r là

$$M_r = \begin{bmatrix} A + BR^{-1}D^TC & BR^{-1}B^T \\ -C^T(I + DR^{-1}D^T)C & -(A + BR^{-1}D^TC)^T \end{bmatrix} \tag{2.2}$$

với $R = r^2I - D^TD$, r là một số không âm.

Định lý 2.1.2. *Đặt $G(s)$ là hàm truyền ổn định và $r > 0$. Khi đó $\|G\|_\infty < r$ nếu và chỉ nếu $\sigma_{max}(D) < r$ và M_r không có giá trị riêng thuần ảo.*

Định lý này cho ta biết nếu r $> r^* = \|G(s)\|_\infty$ thì M_r sẽ không có giá trị riêng thuần ảo nào. Ngược lại nếu r $< r^*$ thì M_r sẽ có ít nhất một giá trị riêng thuần ảo.

2.2 Thuật toán phân đôi

Kết quả trên dẫn ta đến thuật toán phân đôi được trình bày bởi Boyd *et al* [**3**]. Trong mỗi vòng lặp của thuật toán, ta kiểm tra xem mỗi giá trị riêng của ma trận M_r có tồn

tại một giá trị riêng thuần ảo nào hay không. Cần lưu ý rằng thuật toán này hội tụ tuyến tính và xấp xỉ chuẩn H_∞ với độ chính xác tương đối ϵ. Để bắt đầu thuật toán, ta cần tính toán cận trên và cận dưới của khoảng chứa r. Thuật toán có thể bắt đầu ngay khi đặt $r_{lb} = 0$ và gán cho r_{ub} một giá trị đủ lớn. Tuy nhiên để có hai cận chặt hơn, ta có thể sử dụng các giá trị kỳ dị Hankel của hệ.

Định nghĩa 2.2.1. Các giá trị kỳ dị Hankel là căn bậc hai của các giá trị riêng của ma trận $C_G O_G$, với C_G và O_G lần lượt là giá trị Grammian điều khiển và Grammian quan sát, được định nghĩa trong (1.4.8) và (1.4.9).
Kí hiệu các giá trị kỳ dị Hankel là σH_i, được sắp xếp thứ tự giảm dần với σH_1 là giá trị lớn nhất.

Hai công thức tính r_{lb} và r_{ub} trong Bước 1 của thuật toán dưới đây được gọi là **Công thức Enns-Glover**. Ta cũng có hai công thức khác để tính toán giá trị hai cận cho thuật toán phân đôi như sau:

$$\begin{aligned} r_{lb} &= max(\sigma_{max}(D), \sqrt{Trace(O_G C_G)/n}), \\ r_{ub} &= \sigma_{max}(D) + 2\sqrt{nTrace(C_G O_G)}. \end{aligned} \tag{2.3}$$

Algorithm 1: Thuật toán phân đôi

Result: Hai cận r_{lb}, r_{ub} và M_r có giá trị riêng thuần ảo hay không?
Input: Xét hệ (2.1) với các hệ số A, B, C, D, trong đó A ổn định và sai số $\epsilon > 0$

Tính toán các cận trên và cận dưới r_{ub} và r_{lb}

$$\begin{aligned} r_{lb} &= max\left\{\sigma_{max}(D), \sigma H_1\right\}, \\ r_{ub} &= \sigma_{max}(D) + 2\sum_{j=1}^{n} .\sigma H_i \end{aligned} \tag{2.4}$$

Đặt r = $(r_{lb} + r_{ub})/2$.
while *2(r_{ub} - r_{lb}) > ϵ* **do**
    Tính M_r
    if *M_r có giá trị riêng thuần ảo* **then**
        r_{lb} = r
    end
    else
        r_{ub} = r
    end
end

Sau đây là đoạn mã thực thi thuật toán phân đôi bằng ngôn ngữ Python.

```
import numpy as np
from scipy.linalg import *
from control import *
from slycot import *

sys = ss(A,B,C,D)
n = len(A)
r = len(C)
I = pow(A,0)
Wc = gram(sys,'c')
Wo = gram(sys,'o')
print(f'The controllability matrix: \n {Wc} \n \n The observability
    matrix is: \n {Wo}')

def is_stable(sys, A):
    if np.any(np.linalg.eigvals(sys.A).real >= 0.0):
        print("The system is unstable!")
    else: print("The system is stable!")

def Hamil_matrix(A,B,C,D,q):
  M_r = np.zeros([2,2,n,n])
  if np.all(D!=0):
    R = q**2*I - D.T*D
    M_r[0][0] = A + B*inv(R)*D.T*C
    M_r[0][1] = B*inv(R)*B.T
    M_r[1][0]= -C.T*(I + D*inv(R)*D.T)*C
    M_r[1][1]= -(A + B*inv(R)*D.T*C).T
    return np.matrix.round(M_r.transpose(0,2,1,3).reshape(6,6),4)
  else:
    M_r[0][0] = A
    M_r[0][1] = (1/q**2)*B*B.T
    M_r[1][0]= -C.T*C
    M_r[1][1]= -A.T
    return np.matrix.round(M_r.transpose(0,2,1,3).reshape(6,6),4)

e = 1e-6
r_lb = round(max(np.amax(svdvals(D)), np.sqrt(((Wo*Wc).trace())/n)),4)
r_ub = round(np.amax(svdvals(D)) + 2*np.sqrt(n*((Wc*Wo).trace())),4)
i = 0
while 2*(r_ub - r_lb) > e:
  y = 0.5*(r_ub + r_lb)
  M_r = np.matrix.round(Hamil_matrix(A,B,C,D,y),4)
  if np.any(np.matrix.round(np.linalg.eigvals(M_r),4).real == 0.0):
    r_lb = y
  else:
    r_ub = y
  i+=1
```

```
print(f'Number of iteration: {i}')
print(f'The upper bound and the result is:  {r_ub}')
print(f'The lower bound is: {r_lb}')

System specification: Intel Core i7-4810MQ 2.8 GHz, 12GB RAM
```

Listing 2.1: Thuật toán phân đôi

Theo Định lý 2.1.2 thì nếu ma trận M_r không có giá trị riêng thuần ảo nào, ta lấy r làm cận trên cho thuật toán. Tương tự nếu M_r có các giá trị riêng thuần ảo, ta có thể lấy r làm cận dưới mới. Để r^* chắc chắn là giá trị xấp xỉ chính xác, ta dừng thuật toán khi $2(r_{ub}$ - $r_{lb}) < \epsilon$.
Ta thấy rằng để bắt đầu thuật toán, cần phải tính toán hai cận bằng cách giải hai phương trình Lyapunov (1.4.8) và (1.4.8), thực hiện phép nhân hai ma trận cỡ n $\times$ n và tính giá trị riêng của tích hai ma trận. Đối với các ma trận cỡ lớn, việc tính toán rất tốn tài nguyên và gần như không thể. Hơn nữa thuật toán dựa vào việc tính chính xác các giá trị riêng, điều này cần những bộ giải riêng biệt, như trong ví dụ [**2**].

Ví dụ 2.2.2. Xét hệ (2.1) với các hệ số

$$A = \begin{bmatrix} -4 & -8 & -12 \\ 0 & -8 & 0 \\ 0 & 0 & -16 \end{bmatrix}, \; B = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \; C = [1 \quad 1 \quad 1], \; D = 0.$$

Đặt $\epsilon = 10^{-6}$.
Ta tính O_G và C_G bằng lệnh ***lyap*** trong MATLAB thu được:

$$O_G = \begin{bmatrix} 0.1250 & 0 & 0.1250 \\ 0 & 0.0625 & 0 \\ 0.1250 & 0 & 0.1250 \end{bmatrix}, \; C_G = \begin{bmatrix} 0.1250 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Sử dụng công thức (2.3) cho ra

$$r_{lb} = 0.0722, \; r_{ub} = 0.433.$$

suy ra r = 0.2526 ở bước lặp đầu tiên và

$$M_r = \begin{bmatrix} -4.0000 & -8.0000 & 12.0000 & 15.6375 & 0 & 0 \\ 0 & -8.0000 & 0 & 0 & 0 & 0 \\ 0 & 0 & -16.0000 & 0 & 0 & 0 \\ -1.0000 & -1.0000 & -1.0000 & 4.0000 & 0 & 0 \\ -1.0000 & -1.0000 & -1.0000 & 8.0000 & 8.0000 & 0 \\ -1.0000 & -1.0000 & -1.0000 & -12.0000 & 0 & 16.0000 \end{bmatrix}.$$

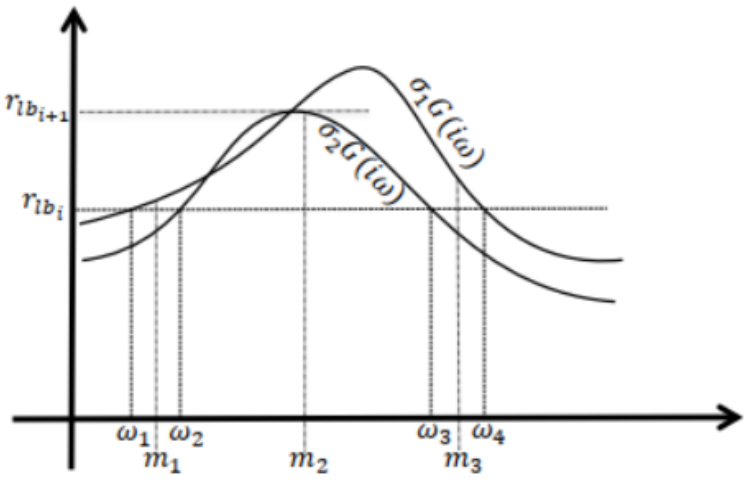
Các giá trị riêng của M_r là $8.0000, 16.0000, -0.5714, 0.5714, -8.0000, -16.0000$. Vì không có giá trị nào là số thuần ảo nên ta đặt r = r_{ub} và tiếp tục vòng lặp.
Sau 20 bước lặp, ta tìm được $r_{lb} = 0.249900$ và $r_{ub} = 0.250000$, thỏa mãn điều kiện dừng. Ta tính được $\|G(s)\|_\infty \approx .250000$.

2.3 Thuật toán hai bước

Ngay sau khi thuật toán phân đôi được trình bày, một thuật toán hai bước đã được đề xuất bởi Bruinsma và Steinbuch [**4**], trong đó ta chỉ cần tính cận dưới. Cần lưu ý rằng một thuật toán tương tự đã được Boyd và Balakrishnan [**3**] đưa ra cùng thời điểm. Điều cần lưu ý là tính toán cận dưới không yêu cầu tìm O_G hoặc C_G, do đó tính toán ít tốn kém hơn. Thuật toán của Bruinsma và Steinbuch cũng dựa vào Định lý 2.1.2 cũng như định lý sau.

Định lý 2.3.1. *Giả sử $r > \sigma_{max}(D)$ và $\omega \in \mathbb{R}$. Khi đó $det(M_r - \omega i I) = 0$ khi và chỉ khi $\sigma_n(G(\omega i)) = r$ với một số n nào đó.*

Hệ quả của định lý này cho ta biết ωi là giá trị riêng của M_r khi và chỉ khi r là giá trị kỳ dị của $G(\omega i)$ với $\omega \in \mathbb{R}$. Trong hình 2.1, ωi biểu diễn tần số, r là bước chạy và m_i biểu diễn điểm ở giữa của ω_i và ω_{i+1}. Trong Định lý 2.3.1 cho ta mối liên hệ giữa các giá trị riêng thuần ảo và các giá trị của r. Nói rõ hơn là mỗi m_i liên hệ đến một giá trị của r. Trong hình 2.1, ta thấy vòng lặp tiếp theo của r sẽ là giá trị kỳ dị lớn nhất của G(s) với mỗi m_i.
Thuật toán của Bruinsma và Steinbuch được xây dựng dựa trên ý tưởng của thuật toán phân đôi, nhưng sử dụng Định lý 2.3.1 để tìm kiếm $\|G(s)\|_\infty$ bằng cách sử dụng nhiều giá trị của ω. Thuật toán phân đôi chỉ tìm kiếm ở một tần số mỗi lần lặp, trong khi thuật toán hai bước tìm kiếm nhiều tần số mỗi lần lặp. Đây là ưu điểm chính của việc sử dụng thuật toán này. Tác giả của thuật toán cho rằng thuật toán này hội tụ bậc hai (họ cũng cho rằng $r_i < r_{i+1}$, dù không có chứng minh nào).



Hình 2.1: Mối liên hệ giữa giá trị kỳ dị và giá trị riêng

Để bắt đầu thuật toán, ta đặt

$$r_{lb} = max\left\{\sigma_{max}G(0), \sigma_{max}G(\omega_p i), \sigma_{max}(D)\right\} \tag{2.5}$$

để tìm cận dưới, trong đó $\omega_p = |\ \lambda_i|$ và ω_i là cực của hàm truyền $G(s)$, được chọn theo tiêu chí sau:
Nếu hàm $G(s)$ có một hay nhiều cực là số ảo, thì λ_i sẽ là cực làm cực đại hóa giá trị

$$\left|\frac{Im(\lambda_i)}{Re(\lambda_i)}\frac{1}{\lambda_i}\right|. \tag{2.6}$$

Nếu $G(s)$ có các cực hoàn toàn là số thực, thì λ_i là cực làm cực đại hóa $\left|\frac{1}{\lambda_i}\right|$.
Cũng như thuật toán phân đôi, phương pháp này áp dụng với hệ có ma trận A ổn định.

Algorithm 2: Thuật toán hai bước

Result: Chuẩn H_∞

Input: Xét hệ (2.1) với các ma trận hệ số A, B, C, D. Trong đó ma trận A ổn định.

1 Tính hàm truyền $G(s)$ của hệ.
2 Tìm các cực của $G(s)$.
3 **if** *Các cực có giá trị thực chặt* **then**
4 | Đặt $\omega_p = max|\lambda_i|$.
5 **else**
6 | Đặt $\omega_p = \lambda_i$, với λ_i cực đại hóa (2.6).
7 **end**
8 Tính r_{lb} sử dụng công thức (2.5).
9 Tính $r = (1 + 2\epsilon)r_{lb}$.
10 Tính M_r bằng công thức (2.2). Sắp xếp toàn bộ các giá trị thuần ảo của M_r, gán nhãn cho chúng theo chiều giảm dần từ $\omega_1, ..., \omega_k$. Nếu M_r không có giá trị thuần ảo nào, đặt $r_{ub} = r$ và chuyển sang bước 16.
11 **for** $i \leftarrow 1$ **to** $k - 1$ **do**
12 | (i) Tính $m_i = \frac{1}{2}(\omega_i + \omega_{i+1})$.
13 | (ii) Tính $\omega_{max}(G(m_i i)) = svd_i$.
14 | Đặt $r_{lb} = max(svd_i)$.
15 **end**
16 Tính $\|G\|_\infty = \frac{1}{2}(r_{lb} + r_{ub})$.

Sau đây là đoạn mã thực thi thuật toán hai bước bằng ngôn ngữ Python.

```
import numpy as np
from scipy.linalg import *
from control import *
from slycot import *

n = A.shape[0];
r = C.shape[0];
I = pow(A,0);
res = 0
temp1 = []
temp2 = []
e = 1e-6
tol = 1e-9

temp = signal.StateSpace(A,B,C,D);
sys = temp.to_ss()
G = sys.to_tf()
p = np.matrix.round(G.poles,6)

Wc = gram(sys,'c')
Wo = gram(sys,'o')

def is_stable(A):
    if np.any(np.linalg.eigvals(A).real >= 0.0):
        print("the system is unstable!")
    else: print("the system is stable!")

def find_pole(A):
  A_e_val=np.matrix.round(np.linalg.eigvals(A),6)
  w_p = 0
  if np.all(A_e_val.imag == 0):
    temp = max(abs(1/A_e_val[::]))
    for i in A_e_val:
      if(1/abs(i)==temp):
        w_p = max(abs(i))
    return w_p
  else:
    temp = max(abs((A_e_val[::].imag/A_e_val[::].real)*(1/A_e_val[::])
   ))
    for i in A_e_val:
      if(abs((i.imag/i.real)*(1/i))==temp):
        w_p = i
    return w_p

def Hamil_matrix(A,B,C,D,q):
  M_r = np.zeros([2,2,n,n],dtype=np.complex)
  if np.all(D!=0):
```

```
    R = q**2*I - D.T*D
    M_r[0][0] = A + B*inv(R)*D.T*C
    M_r[0][1] = B*inv(R)*B.T
    M_r[1][0]= -C.T*(I + D*inv(R)*D.T)*C
    M_r[1][1]= -(A + B*inv(R)*D.T*C).T
    return np.matrix.round(M_r.transpose(0,2,1,3).reshape(6,6),6)
  else:
    M_r[0][0] = A
    M_r[0][1] = (1/q**2)*B*B.T
    M_r[1][0]= -C.T*C
    M_r[1][1]= -A.T
    return np.matrix.round(M_r.transpose(0,2,1,3).reshape(6,6),6)

w = round(abs(find_pole(A)),6)
def G(w):
  z = complex(0,w)
  return C*inv(z*I-A)*B + D

r_lb = round(max((max(svdvals(G(w)))),(max(svdvals(D))),max(svdvals(G
    (0)))),6)
r_ub = round(np.amax(svdvals(D)) + 2*np.sqrt(n*((Wc*Wo).trace())),6)

while (abs(r - r_lb)<=tol):
  r = (1+2*e)*r_lb
  H = Hamil_matrix(A,B,C,D,r)
  e_vals = eigvals(H)
  if np.all(e_vals.imag == 0):
    r_ub = r
    break
  else:
    i = 0
    while(i<len(e_vals)-1):
      sorted_eivals = -np.sort(-e_vals)
      m_i_dummy = 0.5*(sorted_eivals[i]+sorted_eivals[i+1])
      m_i = abs(m_i_dummy)
      temp1 = svdvals(G(m_i))
      svd_i = max(temp1)
      dummy = svd_i.tolist()
      temp2.append(dummy)
      i+=1
  r_lb = max(temp2)
  del temp2[:]

res = 0.5*(r_lb+r_ub)
print(f'The solution is {res}')

System specification: Intel Core i7-4810MQ 2.8 GHz, 12GB RAM
```

Listing 2.2: Thuật toán hai bước

Chuẩn H_∞ được định nghĩa là giá trị kỳ dị cực đại trên trục ảo, nên ta có thể thử dọc theo trục ảo để tìm cận dưới cho chuẩn H_∞ . Để có thể chắc chắn tính toán được $\|G(s)\|_\infty$ bằng thuật toán hai bước, ta cần chọn cận dưới cho thuật toán một cách cẩn thận.

Sau khi tính các giá trị riêng của ma trận M_r, ta cần chọn xem r làm cận trên hay dưới bằng cách sử dụng Định lý 2.1.1, tương tự thuật toán phân đôi. Tuy nhiên nếu ta tìm được các giá trị riêng thuần ảo $\omega_1, \omega_2, ..., \omega_n$ thì theo Định lý 2.1.1, r là giá trị kỳ dị với mỗi hàm $G(\omega_i)$. Điều này cho ta biết rằng có tồn tại các phản hồi $m_1, ..., m_{n-1}$, mỗi m_i với hai điểm ω_i, ω_{i+1} sẽ có một vài m_i mà $G(\omega i) < G(m_i)$. Bằng cách tính σ_{max} với $\forall$i, thuật toán hai bước tìm kiếm ở một khoảng rộng hơn nhiều với thuật toán phân đôi. Sau khi tìm $\sigma_{max}(G(m_i))$ với mọi i, ta đặt giá trị này bằng r_{lb} và bắt đầu bước lặp tiếp theo.

Ví dụ 2.3.2. Có

$$\text{A} = \begin{bmatrix} -1.0000 + 1.0000i & -1.0000 & -1.0000 \\ -1.0000 & -2.0000 + 1.0000i & -1.0000 \\ -1.0000 & -1.0000 & -2.0000 - 1.0000i \end{bmatrix}.$$

Các cực của $G(s)$ là các giá trị riêng của A. Các giá trị đó là -3.3589 + 0.2858i, -0.3628 + 0.9534i, -1.2784 - 0.2392i. Vậy ma trận A ổn định.

Tiếp theo ta thấy -0.3628 + 0.9534i cực đại hóa (2.6) và sử dụng (2.3.1) ta tính được $r_{lb} = 0.9907$. Thực tế tính toán sẽ cho ra kết quả là 1.185206.

Sử dụng giá trị trên làm r_{lb} ta đi tính,

$$M_r = \begin{bmatrix} -1+i & -1 & -1 & 1.0188 & 0 & 0 \\ -1 & -2+i & -1 & 0 & 0 & 0 \\ -1 & -1 & -2-i & 0 & 0 & 0 \\ -1 & -1 & -1 & 1+i & 1 & 1 \\ -1 & -1 & -1 & 1 & 2+i & 1 \\ -1 & -1 & -1 & 1 & 1 & 2-i \end{bmatrix}.$$

Tính các giá trị riêng của M_r ta được:

$$\begin{aligned} -3.1927 + 0.2164i, \\ 3.1927 + 0.2164i, \\ -1.3224 - 0.1563i, \\ 1.3224 - 0.1563i, \\ -0.0000 + 0.8597i, \\ 0.0000 + 1.0201i. \end{aligned}$$

Ta tính và sắp xếp các giá trị riêng thuần ảo, ω_1 = -0.0000 + 0.8597i và ω_2 = 0.0000 + 1.0201i. Tính trung bình của 2 giá trị trên ta thu được m_1 = 0.0000 + 0.9399i.

Giá trị đơn lớn nhất của $\mathrm{G}(m_1 i) = 1.0121 = r_{lb}$.
Ta tiếp tục chạy vòng lặp tiếp theo khi có các giá trị riêng chặt của M_r. Sau bốn vòng chạy, ta thu được $\|G\|_\infty = 1.0121$. Thực tế tính toán cho ra kết quả là 1.608108.

Nhận xét. Ta thấy rằng một vấn đề chính nảy sinh trong việc triển khai thuật toán này là sự cần thiết của việc giải giá trị riêng chính xác, phụ thuộc vào việc các giá trị riêng thuần ảo hay không.

Chương 3

Kết luận

Luận văn đi tìm hiểu hai phương pháp tính toán chuẩn H_∞ cho các hệ điều khiển tuyến tính hệ số hằng với giả thiết là hệ không điều khiển là ổn định.

Đối với cả hai cách tiếp cận là phương pháp phân đôi và phương pháp hai bước, thì việc tính toán các giá trị riêng và giá trị kỳ dị đều được thực hiện bằng các hàm sẵn có trong MATLAB. Cần chú ý thêm rằng việc tính toán chính xác các giá trị riêng và giá trị kỳ dị có vai trò đặc biệt quan trọng trong việc chuyển từ cách tiếp cận miền tần số sang cách tiếp cận không gian-trạng thái, xem [**16**].