



ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Chương 0: Mở đầu

Trịnh Anh Phúc, Vũ Văn Thiệu, Đinh Việt Sang, Nguyễn Đức Nghĩa

Viện CNTT & TT
Trường Đại Học Bách Khoa Hà Nội

Giới thiệu

- ① Tính toán khoa học là gì ?
- ② Cách tiếp cận dùng trong tính toán khoa học
- ③ Sai số tính toán trong các phương pháp số
- ④ Các loại sai số
- ⑤ Về điều kiện của bài toán
- ⑥ Độ chính xác của máy tính
- ⑦ Sự ổn định số
- ⑧ Hậu quả của các sai sót phần mềm
- ⑨ Tổng kết

Giới thiệu

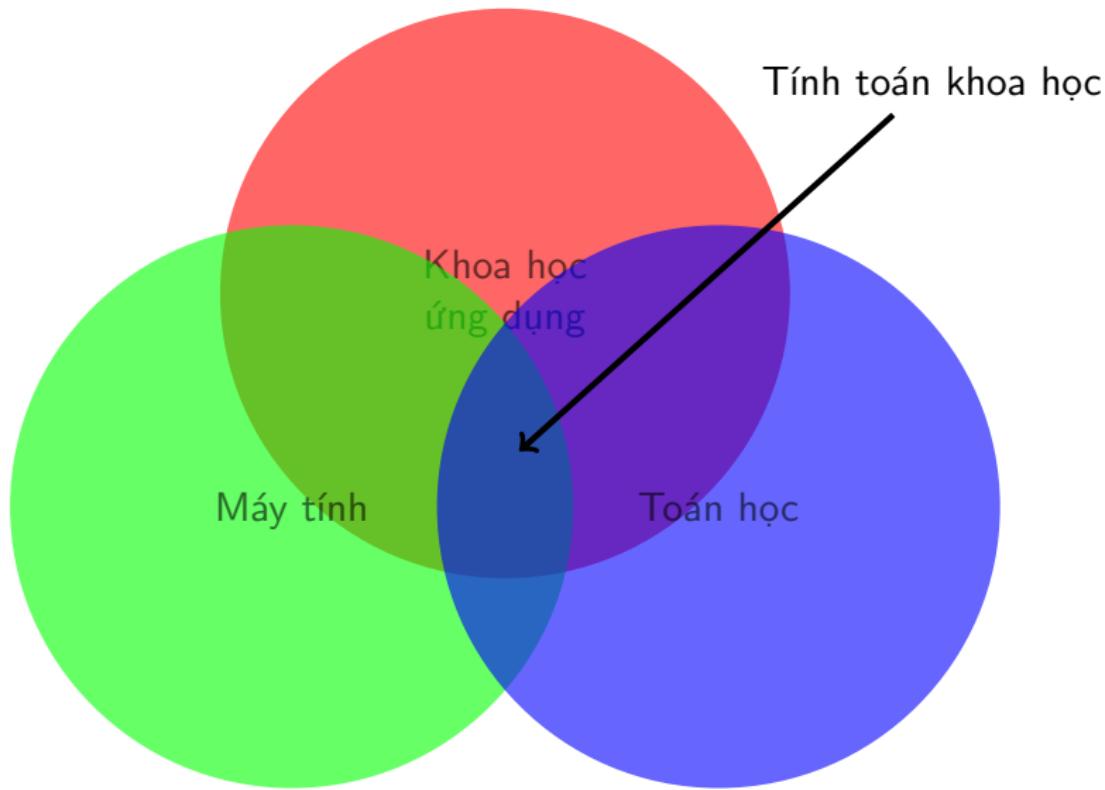
Định nghĩa

Tính toán khoa học là tập hợp tất cả công cụ, kỹ thuật và lý thuyết cần thiết để phát triển và giải quyết các mô hình toán học trong khoa học và kỹ thuật máy tính.

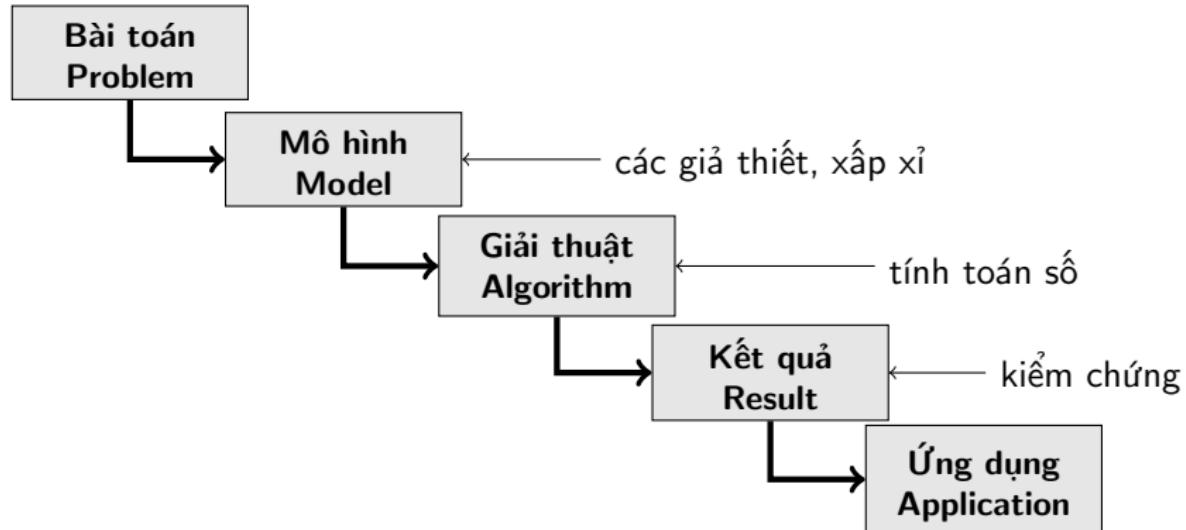
Đặc điểm của Tính toán khoa học:

- Sử dụng sức mạnh của máy tính
- Áp dụng các phương pháp số
- Giải quyết các bài toán kích thước lớn

Giới thiệu



Cách tiếp cận dùng trong tính toán khoa học



Tiếp cận tính toán khoa học

Sai số tính toán trong các phương pháp số

Đặt vấn đề

Khi thiết kế giải thuật để giải các bài toán trong máy tính, ta cần phải kiểm soát được *khoảng cách* giữa lời giải tìm được và lời giải toán học chính xác. Khoảng cách này chính là *sai số* - error - có thể xuất hiện trong quá trình tính lời giải gần đúng cho các mô hình toán học trong máy tính.

Sai số tính toán trong các phương pháp số

Nguồn gốc của sai số

- ① Sai số trong dữ liệu đầu vào (input data error) : Các dữ liệu đầu vào của bài toán có thể là kết quả của các phép đo hoặc phép toán thực hiện trước đó
- ② Sai số do mô hình (model error) : Xuất hiện khi ta lý tưởng hóa bài toán trong việc xây dựng mô hình toán học
- ③ Sai số rút gọn (truncation error) : Xuất hiện khi ta phải ngắt các quá trình vô hạn
- ④ Sai số làm tròn (round-off error) : Xuất hiện khi ta phải làm việc với số vô tỉ (chẳng hạn với số π ta ngắt sau dấu phẩy bao nhiêu số) hoặc khi biểu diễn số trên máy tính (phạm vi biểu diễn luôn hữu hạn)

Sai số tính toán trong các phương pháp số

Ví dụ minh họa

Diện tích bề mặt trái đất tính bởi $S = 4\pi r^2$

- Sai số dữ liệu đầu vào $r = 6370\text{km}$, dữ liệu thực nghiệm
- Sai số mô hình, trái đất là hình cầu tròn tria
- Sai số rút gọn, xuất hiện khi thực hiện các phép toán
- Sai số làm tròn, số π



Các loại sai số

Các loại sai số

Giả sử \hat{x} là xấp xỉ của số thực x .

Ta phân biệt các loại sai số sau:

- Sai số tuyệt đối : $|\hat{x} - x|$
- Sai số tương đối : $\frac{|\hat{x} - x|}{|x|}$ nếu $x \neq 0$

Ta nói \hat{x} là lời giải xấp xỉ chính xác tương đối đến r chữ số sau dấu phẩy khi

$$0.5 \times 10^{-r} < \frac{|\hat{x} - x|}{|x|} \leq 5 \times 10^{-r}$$

Các loại sai số

Ví dụ minh họa

Xét số

$$x = \pi = 3.141592\dots$$

khi đó

$$\hat{x} = 3.1419, \hat{x} = 3.1421, \hat{x} = 3.143$$

là lời giải chính xác tương đối đến $r = 4$ chữ số sau dấu phẩy

Về điều kiện của bài toán

Các định nghĩa

Khái niệm liên quan đến độ nhạy cảm của lời giải tìm được phụ thuộc vào sai số dữ liệu đầu vào

- Bài toán được gọi là có điều kiện tốt (well-conditioned) nếu sai số nhỏ trong dữ liệu dẫn đến sai số nhỏ trong lời giải
- Bài toán được gọi là có điều kiện tồi (ill-conditioned) nếu sai số nhỏ trong dữ liệu dẫn đến sai số lớn trong lời giải

Độ chính xác của máy tính

Định nghĩa

Gọi $fl(x)$ là biểu diễn số thực dấu phẩy động - floating point number - trong máy tính, và ϵ_M là **độ chính xác của máy tính**. Ta có

$$\frac{|fl(x) - x|}{|x|} \leq \epsilon_M$$

Độ chính xác này của máy tính tạo ra giới hạn sai số tương đối mà ta có thể mong đợi từ các thuật toán số.

Sự ổn định số

Sự ổn định số

Khái niệm ổn định số liên quan đến độ chính xác của thuật toán khi làm tròn. Một thuật toán được gọi là *không ổn định* nếu sai số làm tròn có thể dẫn đến sai số lớn trong kết quả.

Hậu quả của các sai sót phần mềm

Ngày 25 tháng 2 năm 1991, tên lửa Scud bắn trúng trại lính Mỹ giết chết 28 lính. Sai số làm tròn thanh ghi 24 bit trong tên lửa đánh chặn Patriot, chậm 0.34 giây tương đương khoảng cách 1/2 km do tốc độ tên lửa 1676m/s.



Hậu quả của các sai sót phần mềm

Ngày 4 tháng 6 năm 1996, tên lửa Arian 5 nổ tung sau 40 giây rời bệ phóng Kourou, Guiana, Pháp. Tốc độ ngang của tên lửa đáng nhẽ dùng dấu phẩy động 64 bit thì bị ép về kiểu số nguyên 16 bit.



Hậu quả của các sai sót phần mềm

Ngày 13 tháng 12 năm 1994, hàng Intel phải thu hồi các sản phẩm chip Pentium vỉ lỗi làm tròn thương số phép chia đến 5 chữ số. Giả sử cho $A = 4195835.0$ và $B = 3145727.0$, thì kết quả công thức trở thành

$$A = A - (A/B) * B = 256$$



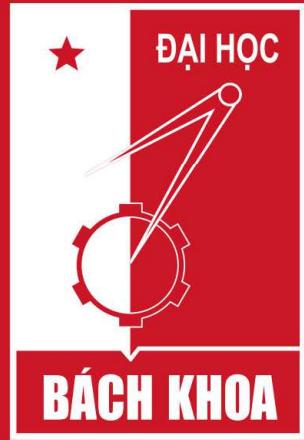


25
SOICT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

**Thank you for
your attentions!**





25 YEARS ANNIVERSARY
SCHOOL OF COMPUTER SCIENCE AND TECHNOLOGY

ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG



ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

CHƯƠNG 1

NHẬP MÔN MATLAB

Vũ Văn Thiệu, Đinh Viết Sang, Nguyễn Khánh Phương

TÍNH TOÁN KHOA HỌC

Nội dung

1. Giới thiệu chung về MATLAB
2. Làm việc với MATLAB
3. Lập trình với MATLAB
4. Các phép tính ma trận nâng cao
5. Đồ thị nâng cao
6. Vào ra dữ liệu

Giới thiệu chung về MATLAB

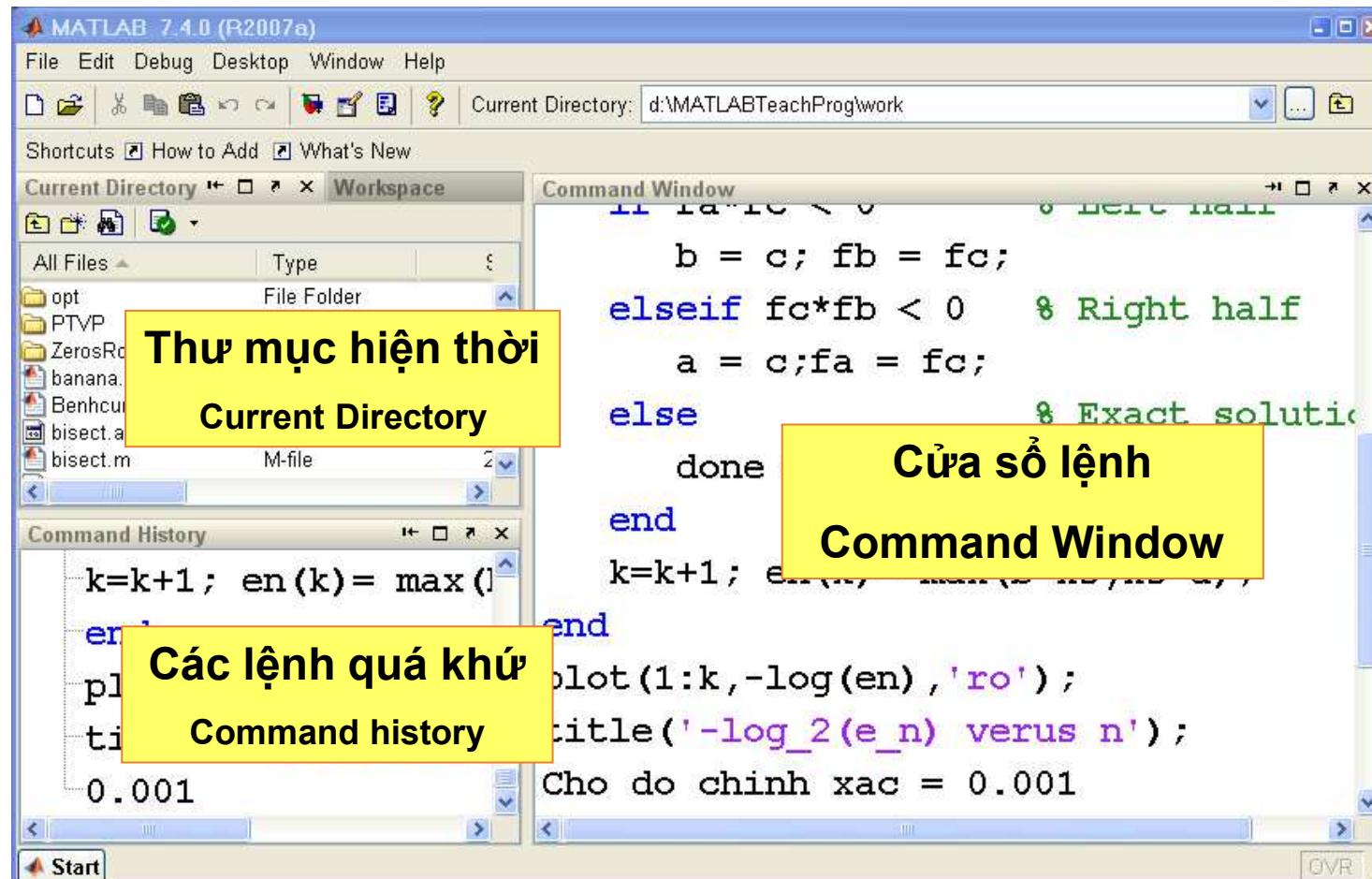
- MATLAB (Matrix Laboratory) là phần mềm của hãng MathWorks Inc.
- Đối tượng là các ma trận.
- MATLAB tích hợp các phương pháp tính toán, hiển thị và ngôn ngữ lập trình mạnh mẽ để cung cấp cho người sử dụng một môi trường làm việc thuận tiện để giải các vấn đề tính toán khoa học.
- Cấu trúc mở của MATLAB cho phép sử dụng MATLAB và các thành phần của nó để khảo sát dữ liệu, nghiên cứu các thuật toán và tạo các công cụ tiện ích của người sử dụng.

Giới thiệu chung về MATLAB

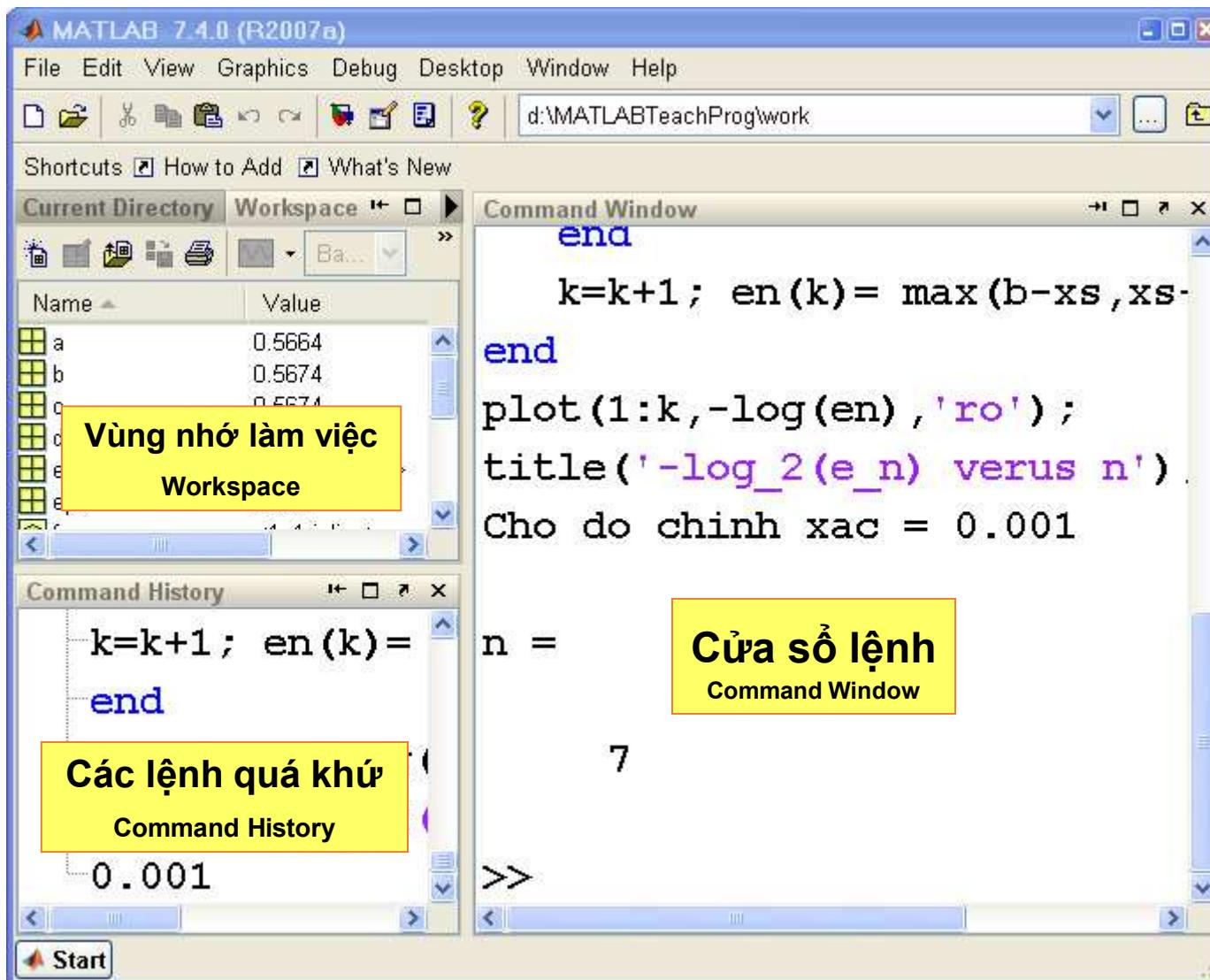
- Matlab cũng đã tạo sẵn rất nhiều công cụ tiện ích như:
 - *Khai phá dữ liệu (Data acquisition)*
 - *Phân tích và khảo sát dữ liệu (Data analysis and exploration)*
 - *Hiển thị và xử lý ảnh (Visualization and image processing)*
 - *Dựng mẫu và Phát triển thuật toán (Algorithm prototyping and development)*
 - *Mô hình hóa và mô phỏng (Modeling and simulation)*
- MATLAB là công cụ được các nhà khoa học, kỹ sư sử dụng để phát triển các phần mềm giải các bài toán tính toán trong khoa học kỹ thuật.
- Bản thân MATLAB cũng cung cấp công cụ để giải nhiều bài toán của khoa học kỹ thuật.
- MATLAB được dùng trong nhiều trường đại học để hỗ trợ việc giảng dạy các giáo trình toán, đặc biệt là các giáo trình liên quan đến tính toán số như đại số tuyến tính ứng dụng, giải tích số, tính toán khoa học,

LÀM VIỆC VỚI MATLAB

Màn hình làm việc của Matlab



Màn hình làm việc của Matlab

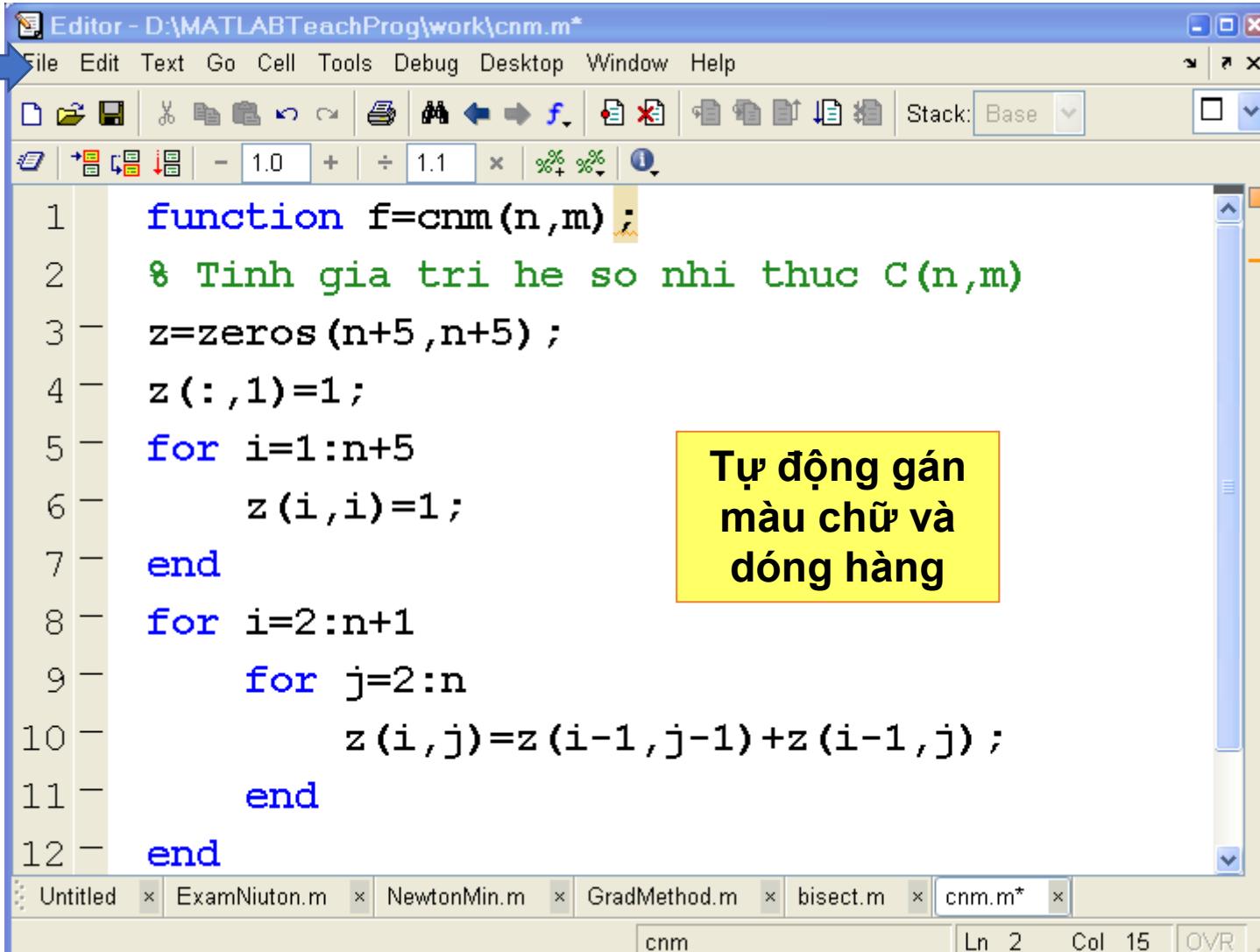


Chương trình trên Matlab

- Matlab có thể làm việc như là một siêu máy tính cầm tay nếu chúng ta chỉ cần Matlab thực hiện một số lệnh bằng cách đánh trực tiếp trên cửa sổ lệnh...
- Chương trình được thực hiện bằng cách nào?
- Chương trình trong Matlab có thể là:
 - **Kịch bản (Scripts)**, hoặc
 - **Các hàm (Functions)**
- **Scripts**: Dãy lệnh Matlab ghi trong một file được đưa vào cửa sổ lệnh và được thực hiện tức thì
- **Functions**: Các môđun chương trình tiếp nhận dữ liệu vào và trả lại kết quả (ví dụ hàm sin nhận đầu vào x và trả lại giá trị $\sin(x)$)
- Chương trình có thể được soạn thảo bằng bất cứ bộ soạn thảo văn bản nào (tuy nhiên Matlab cũng cung cấp bộ soạn thảo chương trình của riêng mình)

Matlab Editor

Các chức
năng



Editor - D:\MATLABTeachProg\work\cnm.m*

```
function f=cnm(n,m);
% Tinh gia tri he so nhanh thuc C(n,m)
z=zeros(n+5,n+5);
z(:,1)=1;
for i=1:n+5
    z(i,i)=1;
end
for i=2:n+1
    for j=2:n
        z(i,j)=z(i-1,j-1)+z(i-1,j);
    end
end
```

Tự động gán
màu chữ và
dòng hàng

Untitled ExamNiuton.m NewtonMin.m GradMethod.m bisect.m cnm.m* cnm Ln 2 Col 15 OVR

Cơ cấu làm việc của Matlab

- Matlab là ngôn ngữ thông dịch (interpreted language)
 - Các câu lệnh được đánh trực tiếp trong **cửa sổ lệnh** và được thực hiện tức thì
 - Các biến được phân bổ bộ nhớ ngay lần đầu tiên chúng được khởi tạo
 - Muốn thực hiện lại một lệnh chỉ việc gõ lại lệnh đó
- Tất cả các biến được sử dụng trong cửa sổ lệnh được cất giữ vào **Vùng nhớ làm việc Base Workspace**
 - Có thể gán giá trị mới cho các biến nếu cần thiết
 - Có thể chọn để xoá bỏ một số biến khỏi vùng nhớ làm việc
 - Vùng nhớ làm việc có thể cất giữ vào một file dữ liệu
 - Phần mở rộng của file dữ liệu là **.mat** (ví dụ: **mydata.mat**)
 - File là file nhị phân
 - Các file dữ liệu (đuôi **.mat**) có thể nạp trở lại vào Vùng nhớ làm việc

Câu lệnh, Chỉ thị & Biến

- Tại dấu nhắc của cửa sổ lệnh, người sử dụng có thể gõ:
 - Lệnh (Command):
 - `save mydata` (cất giữ vùng nhớ làm việc vào `mydata.mat`)
 - `whos` (hiển thị danh mục các biến trong vùng nhớ làm việc)
 - Chỉ thị gán (Assignment Statement):
 - `A = width * length;`
 - `B = 267;`
 - Câu lệnh gán chỉ có một tên biến ở vế trái của toán tử gán (`=`)
 - Vế phải sẽ được tính dựa vào giá trị hiện thời của các biến và kết quả tính được sẽ gán cho biến ở vế trái.
 - Giá trị có thể có dạng số hoặc dạng ký tự
 - **Kiểu của biến sẽ được cập nhật mỗi khi nó được gán giá trị**
- Biến
 - Phân biệt 31 ký tự đầu tiên; Phân biệt chữ hoa hay thường

Làm việc trong chế độ hội thoại

- Khi sử dụng chế độ hội thoại, người sử dụng đánh trực tiếp câu lệnh vào sau dấu nhắc của MATLAB. Khi ấn nút “Enter”, dñng lệnh sẽ được thực hiện.

- Ví dụ,

```
>> x = 1;
```

```
>> 4*atan(x)
```

- Kết quả sẽ được đưa ra màn hình dưới dạng

ans = 3.1416

- Dấu chấm phẩy ";" ở cuối dñng lệnh được sử dụng để ngăn MATLAB khung đưa kết quả của phép thao tác.

Các từ khoá

- Matlab sử dụng một loạt các từ khoá (reserved words) mà để tránh xung đột, không nên sử dụng để đặt tên biến...

for
end
if
while
function
return
elseif
case
otherwise

switch
continue
else
try
catch
global
persistent
break

Các tên biến và tên hàm chuẩn của MATLAB

Tên biến hằng	Mô tả
<code>ans</code>	Biến ngầm định chứa kết quả
<code>beep</code>	Phát tiếng kêu
<code>pi</code>	Hằng số pi
<code>eps</code>	Số 0 của Matlab
<code>inf</code>	infinity
<code>NaN</code>	not a number
<code>i</code> (<i>hoặc</i>) <code>j</code>	Đơn vị phức
<code>realmin</code> , <code>realmax</code>	Số thực dương nhỏ nhất và lớn nhất
<code>bitmax</code>	Số nguyên lớn nhất
<code>nargin</code> , <code>nargout</code>	Số lượng biến vào/ra của lệnh gọi hàm
<code>varargin</code>	Số lượng biến trong lệnh gọi hàm
<code>varaout</code>	Số lượng biến đầu ra trong lệnh gọi hàm

Các tên biến và tên hàm chuẩn của MATLAB

Tên hàm	Ý nghĩa
abs (x)	Giá trị tuyệt đối
cos (x)	Cos
sin (x)	Sin
tan (x)	Tang
acos (x)	Arcos
asin(x)	Arsin
exp (x)	Hàm mũ của e

Các tên biến và tên hàm chuẩn của MATLAB

Tên hàm	Ý nghĩa
imag (x)	phần ảo của số phức
log (x)	Logarithm cơ số e
log10 (x)	Logarithm cơ số 10
real (x)	phần thực của x
sign (x)	Hàm dấu, trả lại dấu của đối số
pow (x,y)	hàm mũ
sqrt (x)	Căn bậc hai
tan (x)	hàm tang

Khuôn dạng dữ liệu

- Mặc dù tất cả các tính toán số trong Matlab đều được thực hiện với độ chính xác kép (double precision), nhưng khuôn dạng của dữ liệu đưa ra có thể định dạng lại nhờ các lệnh định dạng của Matlab.
- Các biến ngầm định cũng như các biến của người sử dụng định nghĩa đều có thể đưa ra theo nhiều khuôn dạng khác nhau.
- Khuôn dạng được chọn nhờ sử dụng lệnh format:
 - **FORMAT SHORT** số dấu phẩy động có 4 chữ số sau dấu.
 - **FORMAT LONG** số dấu phẩy động có 14 chữ số.
 - **FORMAT SHORTE** số dấu phẩy động có 4 chữ số với số mũ.
 - **FORMAT LONGE** số dấu phẩy động có 15 chữ số với số mũ biểu diễn đúng hoặc gần đúng dưới dạng phân số
 - **FORMAT RAT**

Khuôn dạng dữ liệu: Ví dụ

>> pi

ans = 3.1416

>> format long, pi

ans = 3.14159265358979

>> format long e, pi

ans = 3.141592653589793e+000

>> format short e, pi

ans = 3.1416e+000

>> format rat, pi

ans = 355/113



VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Khởi tạo biến vectơ và ma trận

- Một trong những điểm mạnh của MATLAB là nó cho phép làm việc với các ma trận và vectơ. Để sử dụng một biến ta cần khởi tạo nó. Có thể khởi tạo biến vectơ và ma trận theo nhiều cách.
- Đối với vectơ (hay ma trận chỉ có một dòng) ta có thể sử dụng các cách khởi tạo sau

```
>> a = [1 2 3 4 5 6 7 8 9 10];  
>> b = [1:10];  
>> c = [1:0.5:5.5];  
>> d = sin(a);  
>> e = [5 d 6];
```

Khởi tạo biến vectơ và ma trận

- Một trong những cách khởi tạo vectơ thường dùng là sử dụng toán tử

first : increment : last

- Câu lệnh:

a= first : increment : last

khởi tạo vectơ a bắt đầu từ phần tử *first* và kết thúc tại phần tử *last* với độ dài bước là *increment*. Nếu không chỉ ra *increment*, thì giá trị ngầm định nó là bằng 1.

Khởi tạo biến vectơ và ma trận

- **Ví dụ:**

1) Để khởi tạo vectơ $a = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$ có thể thực hiện

```
>> a=1:10
```

```
a = 1 2 3 4 5 6 7 8 9 10
```

2) Độ dài bước có thể là số âm

```
>> a=[100:-10:20]
```

```
a = 100 90 80 70 60 50 40 30 20
```

3) Các giá trị của thông số có thể xác định bởi biểu thức toán học

```
>> c=0:pi/6:2*pi
```

```
c = Columns 1 through 6
```

```
0 0.5236 1.0472 1.5708 2.0944 2.6180
```

```
Columns 7 through 12
```

```
3.1416 3.6652 4.1888 4.7124 5.2360 5.7596
```

```
Column 13 6.2832
```

Khởi tạo biến vectơ và ma trận

- Để khởi tạo ma trận ta có thể làm như sau

```
>> A = [1 2 3; 4 5 6; 7 8 9];
```

```
>> B = [x; y; z];
```

```
>> C = [A; 10 11 12];
```

- Chú ý là các dòng của ma trận được phân tách nhau bởi dấu ";", trong khi đó các phần tử trên một dòng được phân tách bởi dấu cách (hoặc dấu phẩy).

Địa chỉ của mảng

- Các phần tử của một mảng tổng quát (là vectơ hay ma trận) có thể địa chỉ hóa theo nhiều cách.
- Cách đơn giản nhất là chỉ ra phần tử bởi vị trí dòng và cột của nó trong mảng
- Các dòng và cột được đánh số bắt đầu từ 1.

Địa chỉ của mảng

- Ví dụ

```
>> x = [10 5 3 7 -2 8];  
>> x(5)  
ans =  
-2  
>> A = [3 4 9; 2 5 1; 7 4 2]  
A =  
3 4 9  
2 5 1  
7 4 1  
>> A(1,3)  
ans =  
9
```

Địa chỉ của mảng

- Lấy một hàng

>> A(3,:)

- Lấy một cột

>> A(:,2)

- Lấy một ma trận con trong một ma trận lớn

>> Asub = A(i:j,k:l);

- Đổi vị trí các hàng hoặc cột

>> B = A(:,[3 1 2]);

Địa chỉ của mảng

- Ghép 2 ma trận theo cột
$$>> C = [A \ B];$$
- Ghép 2 ma trận theo hàng
$$>> C = [A; B];$$
- Xóa một cột
$$>> A(:,2) = [];$$

Các phép toán với vector và ma trận

- Tính độ dài của véc tơ (đếm số phần tử)
length
- Tích thước của ma trận
size

>> [rows cols] = size(A);

Các phép toán với vector và ma trận

- Cộng (và trừ) các ma trận cùng kích thước được thực hiện từng thành phần.
- Ví dụ
 - >> A=[5 -1 2; 3 4 7]; B=[2 2 1; 5 0 3];
 - >> A+B
 - ans = 7 1 3
8 4 10
- Chú ý các ma trận phải có cùng kích thước:
 - >> C=[3 1; 6 4];
 - >> A+C
 - ??? Error using ==> + Matrix dimensions must agree.

Các phép toán với vector và ma trận

- Nhân với một số (chia cho một số khác không) cũng được thực hiện theo từng thành phần. Ví dụ:

```
>> A=[5 -1 2; 3 4 7];
```

```
>> 2*A
```

```
ans =
```

```
10 -2 4
```

```
6 8 14
```

- Phép toán * trong tích trên là bắt buộc phải có:

```
>> 2A
```

```
??? 2 | Missing operator, comma, or  
semi-colon.
```

Các phép toán với vector và ma trận

- Cộng vectơ và nhân vectơ với một số được thực hiện tương tự.

```
>> v=[3; 5]; w=[-2; 7];
```

```
>> 10*v-5*w
```

```
ans =
```

```
40
```

```
15
```

Các phép toán với vector và ma trận

- Phép nhân hai ma trận cũng có thể thực hiện được trên Matlab.
- Để nhân hai ma trận hay nhân ma trận với vector chỉ việc dùng toán tử * giống như phép nhân của các đại lượng vô hướng.
- Matlab nhận dạng kích thước của đầu vào và thực hiện phép nhân.
- Điều mà người sử dụng quan tâm là kích thước của các ma trận và vector phải phù hợp để có thể thực hiện được phép toán.

Các phép toán với vector và ma trận

- Ví dụ:

```
>> x = [1 2 3];
```

```
>> A = [4 5 6; 5 4 3];
```

```
>> b = A*x
```

```
??? Error using ==> *
```

Inner matrix dimensions must agree.

```
>> y = [1; 2; 3];
```

```
>> b = A*y
```

b =

32

22

Các phép toán với vector và ma trận

- Chuyển vị véc tơ

```
>> A = [ 4 5; 5 4; 6 3]
```

```
>> A'
```

```
ans =
```

$$\begin{matrix} 4 & 5 & 6 \\ 5 & 4 & 3 \end{matrix}$$

Các phép toán với vector và ma trận

- **Tích theo từng thành phần** của hai ma trận cùng kích thước A và B là ma trận A.*B với các phần tử là tích của các phần tử tương ứng của A và B.
- Ví dụ:

```
>> A=[ 1 2 3; 4 5 6]; B=[3 2 1;-1 2 2];  
>> A.*B  
ans = 3 4 3  
      -4 10 12
```

Các phép toán với vector và ma trận

- **Phép chia và luỹ thừa theo từng thành phần:**

$A./B$ và $A.^B$ được định nghĩa tương tự. Lưu ý là các phép tính với các thành phần phải là có nghĩa, nếu không MATLAB sẽ báo lỗi.

- Ví dụ:

```
>> A ./B  
ans = 1/3 1 3  
      -4 5/2 3
```

```
>> A.^B  
ans = 1 4 3  
      1/4 25 36
```

Các phép toán với vector và ma trận

- **Phép cộng ma trận với vô hướng:** Giá trị của vô hướng được cộng vào từng thành phần của ma trận.
- **Ví dụ:**

```
>> A=[1 2 3; 2 3 4];  
>> A+5  
ans =  
6 7 8  
7 8 9
```

Các hàm véc tơ hóa

- Các hàm của Matlab để được véc tơ hóa. Nghĩa là nếu input là một mảng thì output cũng là một mảng
- **Ví dụ:** Vẽ đồ thị hàm $y=\sin(x)$ trên đoạn $[0, 2\pi]$

```
>> x = (0:.1:2*pi);  
>> y = sin(x);  
>> plot(x,y)
```

- **Ví dụ:**

```
x = (-5:.1:5);  
>> y = x./(1+x.^2);  
>> plot(x,y)
```

Các hàm tạo ma trận đặc biệt

- zeros(m,n);
- ones(m,n);
- eye(n);
- diag(v);

Biến xâu trong MATLAB

- MATLAB cho phép sử dụng biến xâu ký tự: Sử dụng lệnh gán

S = 'Any Characters'

cho phép tạo mảng ký tự (xâu ký tự).

- Ví dụ:

>> msg = 'You''re right!'

msg =

You're right!

- Lưu ý: Để tạo dấu ‘ trong xâu ký tự cần đánh hai dấu ‘

Biến xâu trong MATLAB

- Lệnh $S = [S_1 \ S_2 \ \dots]$ ghép các xâu S_1, S_2, \dots thành một xâu mới S .
- Ví dụ:

```
>> name = ['Michel' ' Paul ' ' Heath ']  
name =  
      Michel Paul Heath  
>> name(1)  
ans =  
      M  
>> name(1)+name(9)  
ans =  
      174
```

- Lưu ý đến cách truy cập đến các thành phần trong xâu.

Biến xâu trong MATLAB

- $S = \text{char}(X)$ tạo S là ký tự có mã ASCII là X .
- $X = \text{double}(S)$ chuyển ký tự thành số
- Ví dụ:

```
>> char([65 66 67])
```

```
ans =
```

```
ABC
```

```
>> x = double('ABC')
```

```
x =
```

```
65      66      67
```

Biến xâu trong MATLAB

- Để khởi tạo biến mảng mà mỗi phần tử là một xâu, sử dụng:

```
>> S = { 'Hello' 'Yes' 'No' 'Goodbye' }

S =

    'Hello'      'Yes'      'No'
'Goodbye'

>> S(4)

ans =

    'Goodbye'
```

LẬP TRÌNH TRÊN MATLAB



VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Câu lệnh trong Matlab

- Câu lệnh hay gấp nhất trong Matlab có dạng
variable = expression

Câu lệnh này sẽ gán giá trị của biểu thức *expression* cho biến *variable*.

Ví dụ:

```
>> x=2^10*pi  
x =  
3.216990877275948e+003
```

Câu lệnh trong Matlab

- Câu lệnh của Matlab cũng còn có thể có dạng đơn giản như sau:

expression

trong trường hợp này giá trị của biểu thức sẽ được gán cho biến ngầm định có tên là ***ans***.

Ví dụ:

>> 2*pi*exp(-5)

ans =

0.04233576958521

Câu lệnh trong Matlab

- Cuối cùng, một dạng nữa của câu lệnh trong Matlab là ***variable***
 - Nếu như biến đã được gán giá trị trước đó thì nội dung của biến được đưa ra màn hình,
 - Nếu trái lại sẽ có thông báo rằng biến chưa được xác định.
- Người sử dụng có thể tận dụng điều này để kiểm tra xem một tên biến đã được dùng hay chưa.

Câu lệnh trong Matlab

- Câu lệnh của Matlab sẽ được thực hiện ngay sau khi nhấn phím Enter. Nếu câu lệnh Matlab kết thúc bởi Enter, theo ngầm định, Matlab sẽ đưa kết quả thực hiện lên thiết bị ra chuẩn (ngầm định là màn hình).
- Muốn tránh việc đưa kết quả ra ngay trực tiếp sau câu lệnh, cần kết thúc câu lệnh bởi dấu ; sau đó mới đến Enter.
- Khi câu lệnh của Matlab quá dài có thể ngắt thành hai hoặc nhiều dòng sử dụng dấu nối dòng ... ở cuối mỗi dòng chứa câu lệnh.

Câu lệnh trong Matlab

- **Ví dụ:**

```
>> avariablewithlongname = 100 + (32-17.33)*5 ...  
+ 2^3 - log(10)/log(2);
```

- Để xóa tất cả biến trong Matlab, ta dùng lệnh

```
>> clear
```

- Để xóa một số biến cụ thể, ta dùng lệnh
>> clear var1 var 2 ...

trong đó *var1*, *var2*, .. Là các tên của cá biến cần xóa.

Các phép toán quan hệ và logic

Phép toán quan hệ	Ý nghĩa
<	Nhỏ hơn
\leq	Nhỏ hơn hoặc bằng
>	Lớn hơn
\geq	Lớn hơn hoặc bằng
$=$	Bằng
\neq	Không bằng
Phép toán lôgic	Ý nghĩa
&	AND (hội)
	OR (tuyễn)
~	NOT (phủ định)
0	FALSE
Số khác không	TRUE

Câu lệnh if

- Dạng tổng quát của câu lệnh if là

if expr1

statements1

elseif expr2

statements2

 ...

else

statements

end

elseif và **else** là tùy chọn

- Nhóm lệnh đi ngay sau biểu thức (*expr*) đầu tiên có giá trị khác 0 sẽ được thực hiện.
- Nếu không có *expr* nào khác 0 thì nhóm lệnh sau else được thực hiện

Câu lệnh if

- Ví dụ:

```
>> t = rand(1);
>> if t > 0.75
    s = 0;
elseif t < 0.25
    s = 1;
else
    s = 1-2*(t-0.25);
end
>> s
s =
0
>> t
t =
0.7622
```

Câu lệnh if

- Các phép toán quan hệ:

$<$, $>$, \leq , \geq , $==$, $\sim=$

- Ví dụ:

$>> 5>3$

ans =

1

$>> 5<3$

ans =

0

$>> 5==3$

ans =

0

Câu lệnh vòng lặp for

- Vòng lặp for lặp lại các câu lệnh trong thân của nó đối với các giá trị của biến chạy được lấy từ một vector dòng cho trước.
- Ví dụ

```
>> for i=[1,2,3,4]  
    disp(i^2)
```

```
end
```

```
1
```

```
4
```

```
9
```

```
16
```

Câu lệnh vòng lặp for

- Chú ý đến việc sử dụng hàm nội trú **disp**: hàm này đưa ra màn hình nội dung của biến.
- Vòng lặp, cũng giống như câu lệnh if, phải kết thúc bởi **end**.
Vòng lặp ở trên thường được viết dưới dạng như sau.

```
>> for i=1:4  
    disp(i^2)  
end  
1  
4  
9  
16
```

- Nhớ lại cách khởi tạo 1:4 cũng chính là [1, 2, 3, 4].

Câu lệnh vòng lặp for

- Đoạn lệnh

```
n=4; x = [] ; for i=1:n, x = [x, i^2], end  
hay  
x=[];  
for i= 1:n  
    x = [x, i^2];  
end
```

sẽ tạo ra vector $x = [1, 4, 9, 16]$.

- Đoạn lệnh

```
n=4;x=[]; for i=n:-1:1, x = [x, i^2],end
```

sẽ tạo ra vector $x = [16, 9, 4, 1]$.

Câu lệnh while

- Câu lệnh có dạng sau:

```
while expr  
    Statements  
end
```

- Các câu lệnh trong thân của vòng lặp while sẽ được lặp lại chừng nào biểu thức *expr* còn là true (có giá trị khác 0).

Câu lệnh while

- Ví dụ:

```
>> x=1;  
>> while 1+x > 1  
      x = x/2;  
end  
>> x  
x =  
    1.1102e-16
```

- Chú ý: Về mặt chính xác toán học thì $1 + x > 1$ đối với mọi $x > 0$.

Câu lệnh switch

- Câu lệnh **switch** cho phép thực hiện rẽ nhánh dựa trên giá trị biểu thức.
- Dạng tổng quát của câu lệnh **switch** là:

switch bieuthuc

case giatri

 cac cau lenh

case {giatri1, giatri2, giatri3, ...}

 cac cau lenh

...

otherwise

 cac cau lenh

end

Ví dụ:

```
>> date= 'Sunday';
>> switch lower(date)
    case {'sunday','saturday'}
        disp('the weekend.')
    otherwise
        disp('the workday.')
end
```

Kết quả: the weekend.

Câu lệnh break

- Lệnh break dùng để ngắt việc thực hiện vòng lặp while hoặc for.
- Trong vòng lặp lồng nhau, break chỉ thoát khỏi vòng lặp trong cùng
- Nếu sử dụng break ngoài vòng lặp trong một kịch bản, nó sẽ chấm dứt thực hiện kịch bản
- Lệnh break trong cấu trúc IF hoặc Switch Case sẽ chấm dứt thực hiện câu lệnh đó

Kịch bản (Script)

- Kịch bản là dãy các câu lệnh của Matlab được ghi lại trong một m – file, file đuôi .m.
- Khi đánh tên của file (không cần .m), dãy các lệnh này sẽ được thực hiện.
- Lưu ý: m – file phải được đặt tại một trong các thư mục mà Matlab sẽ tự động tìm kiếm m – file trong đó; danh mục các thư mục như thế có thể xem nhờ lệnh **path**.
- Một trong những thư mục mà Matlab luôn khảo sát chính là thư mục hiện thời. Thư mục này được hiển thị trong cửa sổ Current Directory.
- Người sử dụng có thể thay đổi thư mục hiện thời nhờ dùng các tiện ích trong cửa sổ này.
- **Câu hỏi:** thay đổi thư mục hiện thời như thế nào?

Kịch bản: Ví dụ

- Ví dụ kịch bản: plotsin.m

x = 0:2*pi/N:2*pi;

y = sin(w*x);

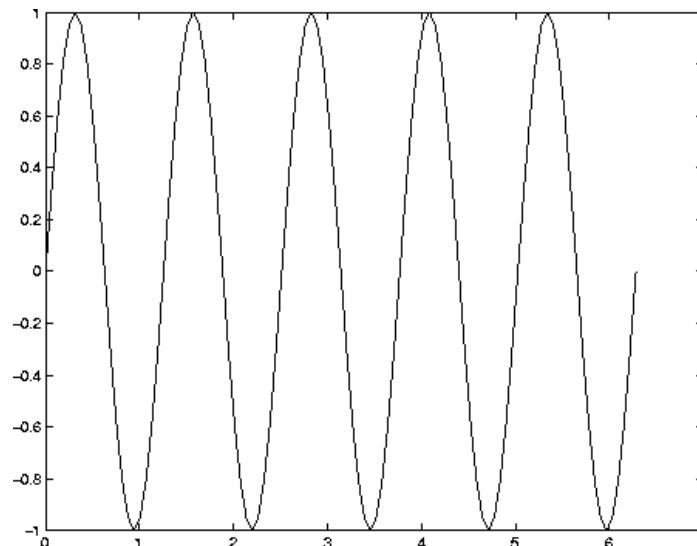
plot(x,y)

- Khi gõ lệnh

>> N=100;w=5;

>> plotsin

sẽ ra đồ thị sau:



Hàm (function)

- Cách định nghĩa hàm:

function [out1,out2,...] = funcname(inp1, inp2,...)

Ví dụ

- Lập Hàm giải phương trình bậc 2



VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

CÁC PHÉP TÍNH MA TRẬN NÂNG CAO

Các phép toán nâng cao

- Matlab cung cấp nhiều hàm liên quan đến xử lý ma trận như:
 - eig : tính trị riêng
 - lu : phân tích LU
 - chol : phân tích Cholesky
 - qr : phân tích qr
 - svd : tính single value
 - cond, condest, rcond: tính các số điều kiện
 - norm : tính chuẩn của ma trận

Đồ họa nâng cao

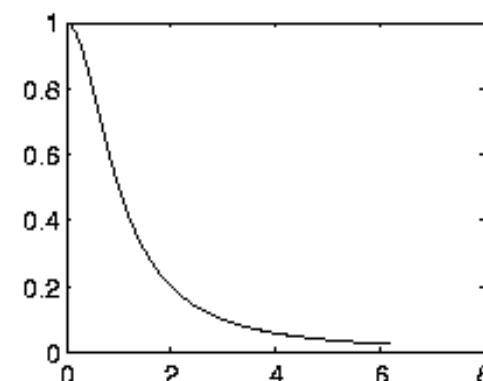
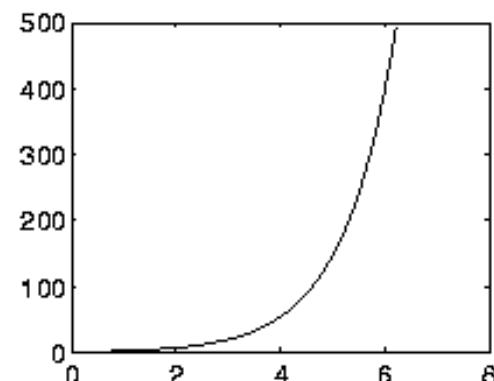
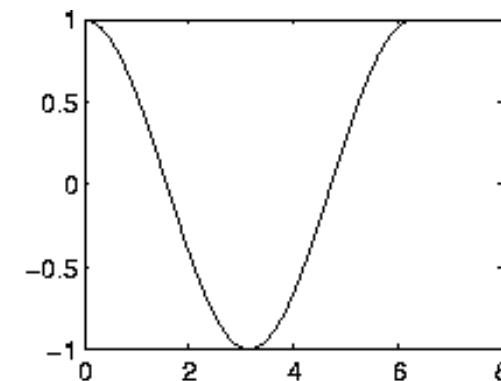
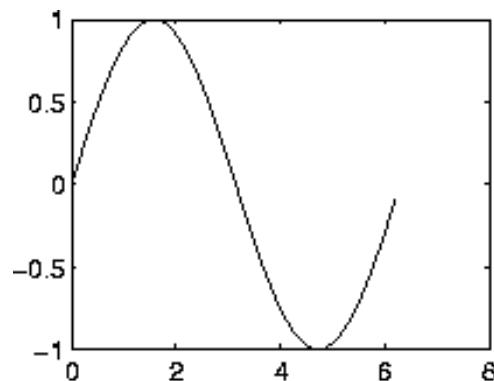
- Vẽ đường cong 2D (2D curves),
- Vẽ mặt 3D (3D surfaces),
- Vẽ đường cong 3D (contour plots of 3D surfaces),

Vẽ nhiều đồ thị trên cùng một cửa sổ

- Ví dụ:

```
>> t = (0:.1:2*pi)';
>> subplot(2,2,1)
>> plot(t,sin(t))
>> subplot(2,2,2)
>> plot(t,cos(t))
>> subplot(2,2,3)
>> plot(t,exp(t))
>> subplot(2,2,4)
>> plot(t,1./(1+t.^2))
```

Đưa ra nhiều đồ thị



Vẽ đường

- `plot(x,y)`
- `plot(x1, y1, linestyle1,...,xn,yn,linestyle)`

Vẽ đường

Màu nét vẽ	Ký tự đánh dấu	Nét vẽ
y yellow	.	(point)
m magenta	o	(circle)
c cyan	x	(x-mark)
r red	+	(plus)
g green	*	(star)
b blue	s	(square)
w white	d	(diamond)
k black	v	(triangular)
	^	(triangular)

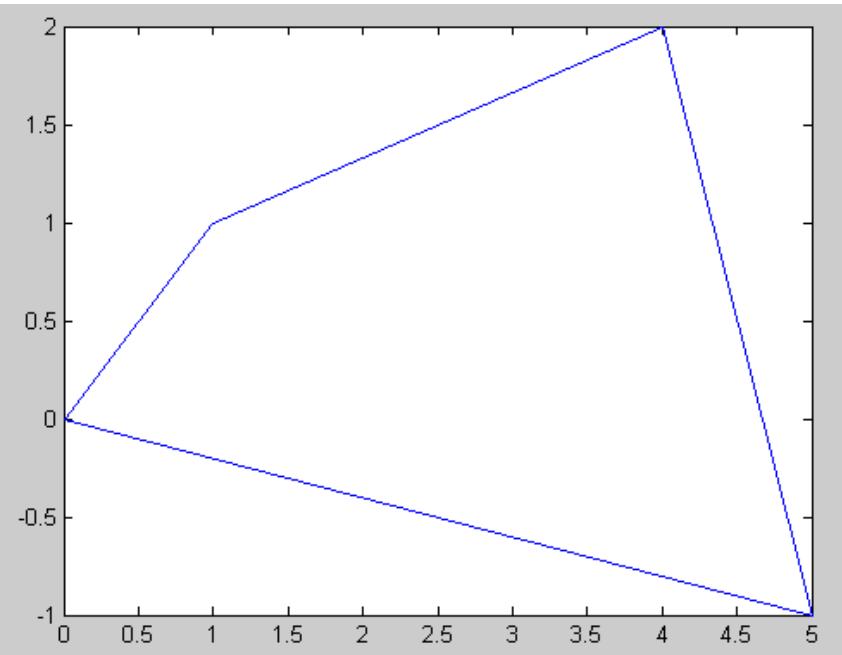
Vẽ đường

- Ví dụ:

```
>> x=[0 1 4 5 0];
```

```
>> y=[0 1 2 -1 0];
```

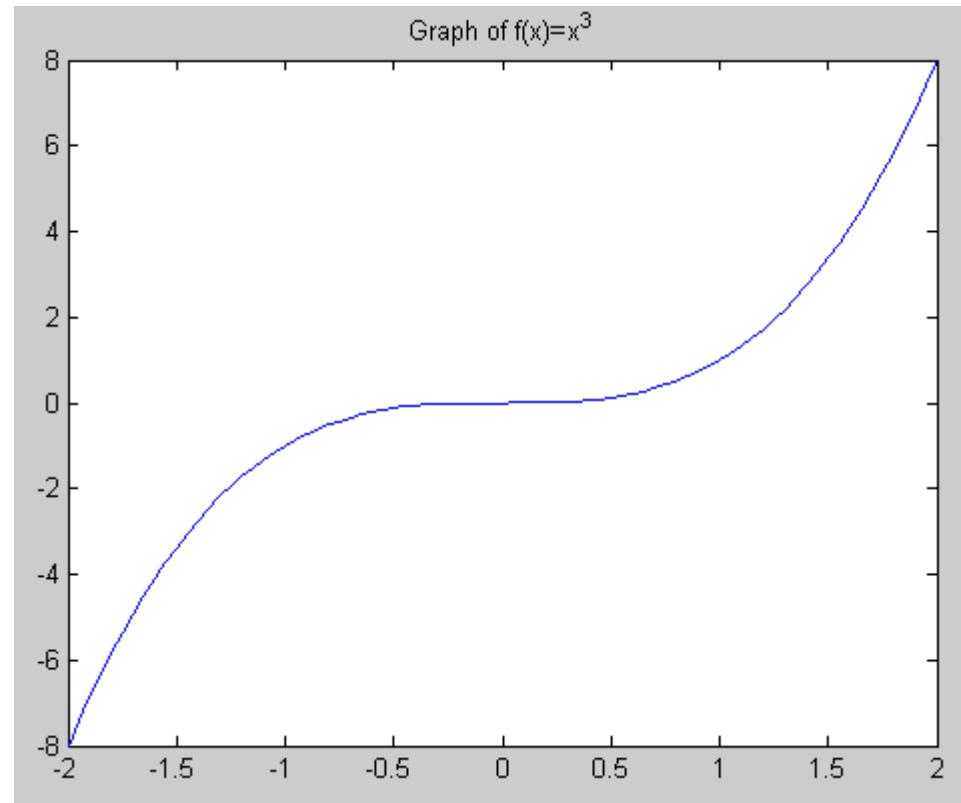
```
>> plot(x,y)
```



Vẽ đường

- Ví dụ: Vẽ đồ thị hàm số $y=x^3$ trên $[-2,2]$,
 - `>> x=-2:.05:2;`
 - `>> y=x.^3;`
 - `>> plot(x,y)`
 - `>> title('Graph of f(x)=x^3')`

Đồ thị hàm số $y = x^3$



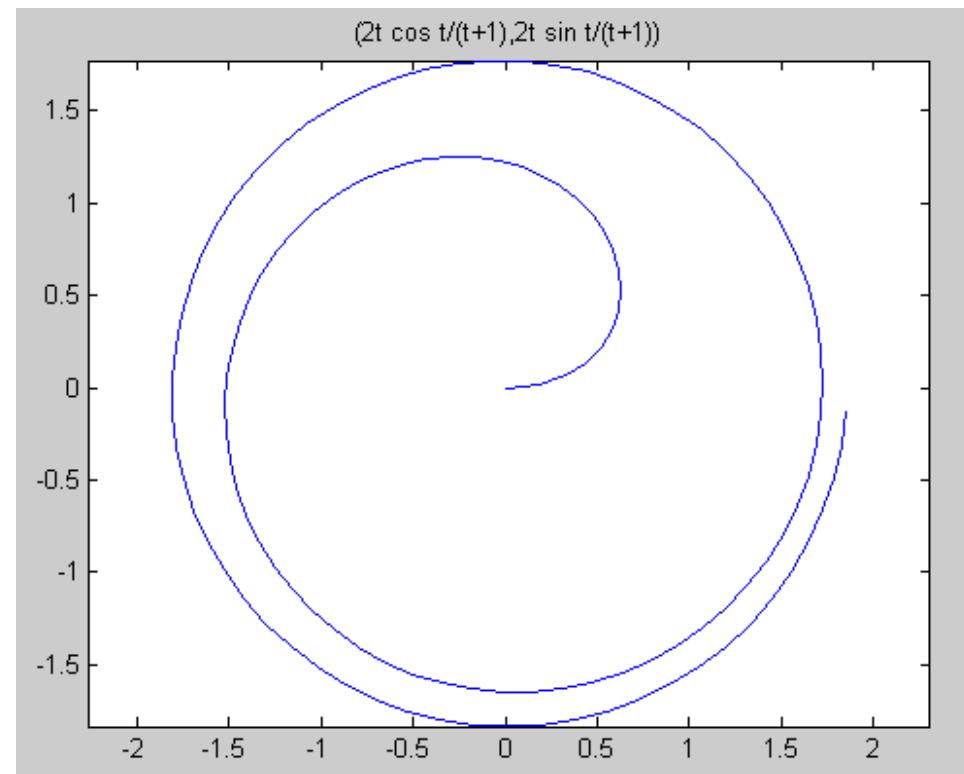
Vẽ đường cong tham số

- Ví dụ: Vẽ đường cong tham số:

```
r(t) = (2tcost/(t+1), 2tsint/(t+1)) với t ∈ [0,4π],  
>> t=0:.1:4*pi;  
>> x=2*t.*cos(t)./(t+1);  
>> y=2*t.*sin(t)./(t+1);  
>> plot(x,y);  
>> title('(2tcost/(t+1),2tsint/(t+1))')
```

Vẽ đường cong tham số

- Cân bằng trực
>> axis equal



Vẽ đường

- Vẽ nhiều đường trên một khung hình:
 - Dùng hold on

```
>> t=0:pi/20:2*pi;  
>> plot(2*cos(t),2*sin(t))  
>> hold on  
>> plot(1+cos(t),1+sin(t))  
>> axis equal  
>> title('The circles x^2+y^2=4 and (x-1)^2  
+(y-1)^2=1')
```

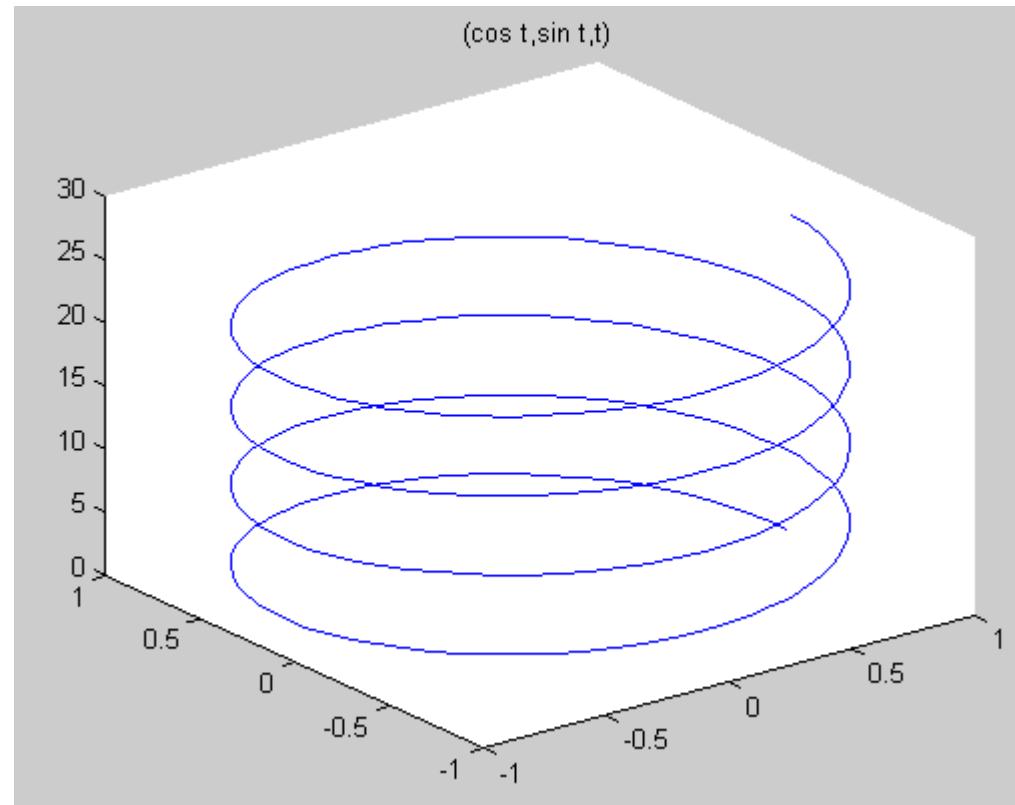
Vẽ đường cong 3D

- **plot3**
- **Ví dụ:**

$\mathbf{r}(t) = (\cos(t), \sin(t), t)$ với $t \in [0, 8\pi]$

```
>> t=0:.1:8*pi;  
>> plot3(cos(t),sin(t),t)  
>> title('(\cos t, \sin t, t)')
```

Vẽ đường cong 3D



Vẽ mặt

- Vẽ đồ thị hàm số $f(x,y)$ trên hình:

$$R = [a,b] \times [c,d] = \{(x,y) \mid a \leq x \leq b; c \leq y \leq d\},$$

```
>> x=0:4;  
>> y=0:.5:3;  
>> [X,Y]=meshgrid(x,y))
```

Vẽ mặt

- Ví dụ: vẽ đồ thị hàm số sau:

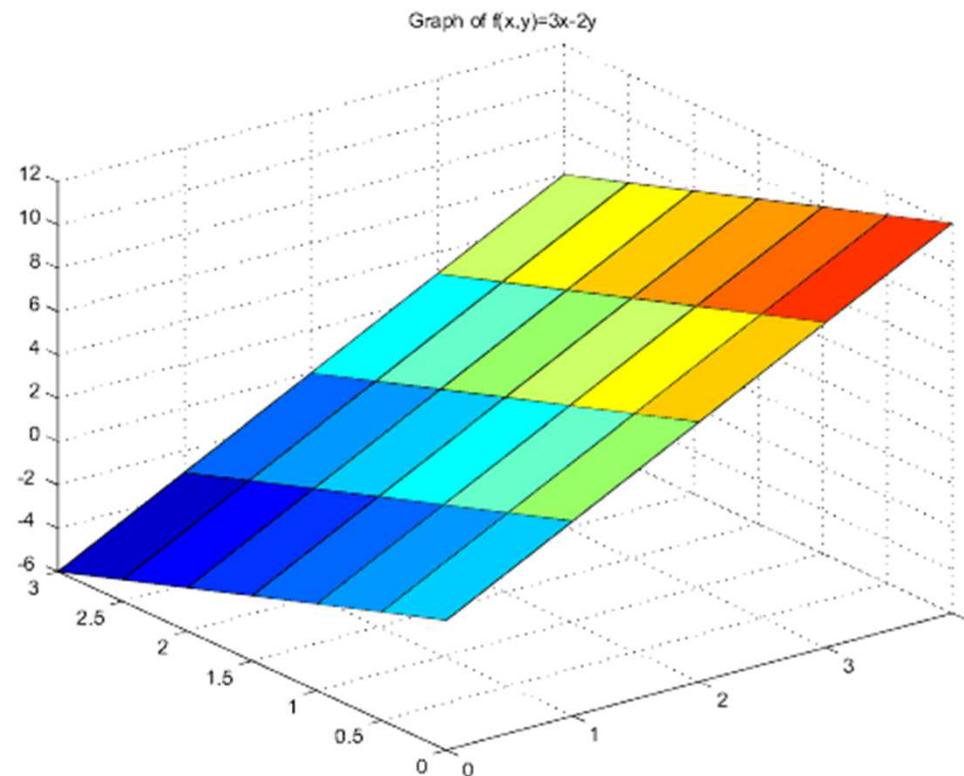
$$f(x,y)=3x - 2y.$$

```
>> Z=3*X-2*Y
```

```
>> surf(X,Y,Z)
```

```
>> title('Graph of f(x,y)=3x-2y')
```

Vẽ mặt



Vẽ mặt

- Ví dụ: Vẽ đồ thị hàm số

$f(x,y)=x^2y - 2y$ trên miền $[-2,2] \times [-1,1]$.

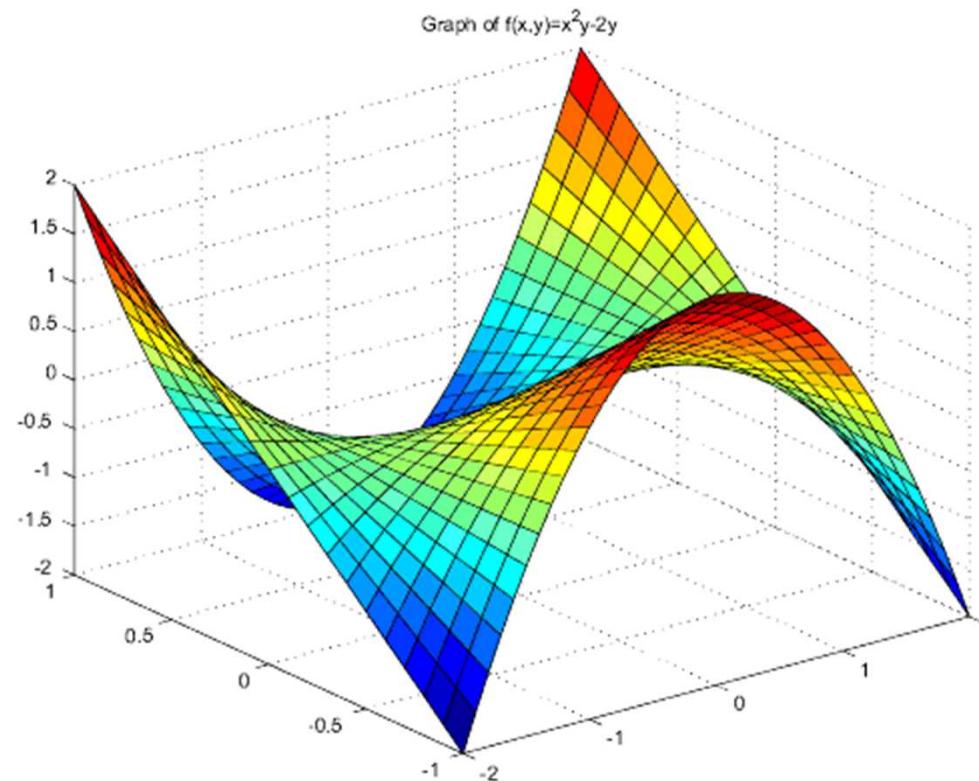
```
>> [X,Y]=meshgrid(-2:.1:2,-1:.1:1);
```

```
>> Z=(X.^2).*Y-2*Y;
```

```
>> surf(X,Y,Z)
```

```
>> title('Graph of f(x,y)=x^2y-2y')
```

Vẽ mặt



Đa thức trong Matlab

- Đa thức bậc n

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

- Biểu diễn bằng vector hàng $n+1$ phần tử

- Sắp xếp hệ số từ bậc cao nhất đến bậc 0

- Ví dụ $p = 3x^4 + x^2 - 3x + 5$

- $>>p=[3 0 1 -3 5]$

- $p = 3 0 1 -3 5$

Các phép tính với đa thức

HÀM	Ý NGHĨA
<code>conv(p1,p2)</code>	Nhân hai đa thức
<code>[k,d]=deconv (p1,p2)</code>	Chia hai đa thức (k= kết quả; d =phần dư)
<code>k=polyder(p)</code>	Tìm đạo hàm của đa thức p
<code>k=polyder(p,q)</code>	Tìm đạo hàm của đa thức tích (p^*q)
<code>[n,d]=polyder(num,den)</code>	Tìm đạo hàm (dạng n/d) của phân thức (num/den)
<code>roots(p)</code>	Tìm nghiệm đa thức p
<code>p=poly(r)</code>	Lập đa thức p từ vectơ r chứa các nghiệm.
<code>polyval(p,x)</code>	Tính giá trị của đa thức tại x (x có thể là mảng)
<code>[r,p,k]= residue(num,den)</code>	Tìm các thành phần tối giản của phân thức
<code>[num,den]=residue(r,p,k)</code>	Chuyển các thành phần tối giản thành 1 phân thức
<code>printsys(num,den,'s')</code>	in phân thức có dạng tỉ số 2 đa thức theo s
<code>[z,p,k]=tf2zp(num,den)</code>	Tìm các zero z, cực p, độ lợi k của phân thức

Các phép tính với đa thức

- **Ví dụ 1:**

```
>> r=[1 3];
```

```
>> p=poly(r)
```

```
p = 1 - 4 3
```

```
>>polyval(p,[1 3 5])
```

```
ans =
```

```
0 0 8
```

```
>> printsys(r,p,'x')
```

```
num/den =
```

```
x + 3
```

```
-----
```

```
x^2 - 4 x + 3
```

- **Ví dụ 2:**

$$\frac{x^4 + 3x - 14}{x^2 - 4} = \frac{(x^4 - 16) + 3x + 2}{x^2 - 4}$$
$$= x^2 + 4 + \frac{1}{x+2} + \frac{2}{x-2}$$

$$k(x)=x^2 + 4$$

$$r=[1 2]$$

$$p=[-2 2]$$

Các phép tính với đa thức

Ví dụ 3:

$$G(s) = \frac{4s^2 + 16s + 12}{s^4 + 12s^3 + 44s^2 + 48s}$$

```
>> num=[4 16 12]
>> den=[1 12 44 48 0]
>> [z,p,k]=tf2zp(num,den)
z =
    -3
    -1
p =
    0
   -6.0000
   -4.0000
   -2.0000
k =
    4
```

Do đó :

$$G(s) = \frac{K(s - z_1)(s - z_2)}{(s - p_1)(s - p_2)(s - p_3)} = \frac{4(s + 3)(s + 1)}{(s + 6)(s + 4)(s + 2)}$$

Vào-Ra dữ liệu

- **Vào dữ liệu từ bàn phím**
 - Các lệnh liên quan đến nhập dữ liệu từ bàn phím: input, keyboard, menu và pause.
- **Lệnh input:** Lệnh này đưa ra thông báo nhắc người sử dụng nhập dữ liệu và nhận dữ liệu đánh từ bàn phím.
Lệnh có dạng

R = input(*string*)

trong đó R là tên biến, *string* là xâu ký tự chứa thông báo nhắc người sử dụng biết về dữ liệu cần nạp. Dữ liệu có thể là biểu thức bất kỳ. Nếu người sử dụng ấn Enter ngay thì R sẽ là ma trận rỗng.

Vào dữ liệu từ bàn phím

- Ví dụ:

```
>> m=input('Nhập số dòng của ma trận:  
' )
```

Nhập số dòng của ma trận: 3

m =

3

```
>> n = input('Hay nhập số cột của ma  
trận n =')
```

Hay nhập số cột của ma trận n =5

n =

5

Vào dữ liệu từ bàn phím

- **Ví dụ**

```
>> a =input('Hay nhap cac phan tu cua ma tran a  
theo khuon dang:\n [Cacphan tu dong 1 ;\n ...  
\n Cac phan tu dong m] : \n\n')
```

Hay nhap cac phan tu cua ma tran a theo khuon dang:
[Cacphan tu dong 1 ;

...

Cac phan tu dong m] :

```
[1 2 3 4 5;  
6 7 8 9 10;  
11 12 13 14 15]
```

Vào-Ra dữ liệu

- **Lệnh keyboard**

- Khi lệnh này được đặt trong m – file thì nó ngắt việc thực hiện chương trình.
- Màn hình xuất hiện dấu nhắc **>>K**.
- Người sử dụng có thể xem hoặc biến đổi giá trị các biến bằng các lệnh của Matlab.
- Chấm dứt chế độ dùng **Enter**.
- Tiện lợi cho việc gỡ lỗi (debug) chương trình.

Vào-Ra dữ liệu

- **Lệnh menu:** Cho phép tạo bảng lựa chọn. Lệnh có dạng sau.

CHOICE = menu (HEADER, ITEM1, . . . , ITEMn)

- Lệnh sẽ hiển thị tiêu đề của bảng chọn chứa trong xâu HEADER, tiếp đến là dãy các lựa chọn trong các xâu: ITEM1, ..., ITEMn.
- Chỉ số của lựa chọn sẽ được trả lại cho biến CHOICE.
- Trong màn hình đồ họa Matlab sẽ hiển thị bảng lựa chọn trong một cửa sổ MENU với các phím ấn chọn.

Vào-Ra dữ liệu

- Ví dụ:

```
>> CHOICE = menu('Hay chon cach vao du  
lieu: ','Keyboard', 'File' , 'Random' )
```

- Trên màn hình đồ họa hiển thị bảng lựa chọn:



- Người sử dụng chọn bằng việc click chuột vào lựa chọn cần thiết.

Vào-Ra dữ liệu

- Cách sử dụng khác của menu

CHOICE = menu (HEADER, ITEMlist)

trong đó ITEMlist là mảng xâu.

- Ví dụ:

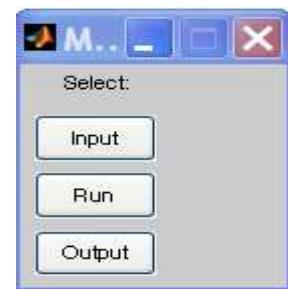
```
>> ItemList= {'Input' , 'Run' , 'Output' } ;
```

```
>> HEADER='Select: ';
```

```
>> CHOICE = MENU (HEADER, ItemList)
```

```
CHOICE =
```

```
1
```



Đưa ra màn hình

- **Đưa ra màn hình nội dung mảng**

- Gõ tên biến không có ; cuối lệnh

- **Ví dụ:**

```
>> x = [1 2 3];
```

```
>> y=[2; 3; 4];
```

```
>> x*y
```

```
ans =
```

```
20
```

Đưa ra màn hình

- **Lệnh disp:**

- Dùng để hiển thị nội dung của biến ra màn hình và không hiển thị tên biến
- Làm việc dù có; hay không
- Lệnh có dạng
disp(X)

với X là biến hoặc biểu thức cần hiển thị.

- **Ví dụ:**

```
>> B = [1 2 3; 4 5 6; 7 8 9]
```

```
>> disp(B^2) ;
```

30	36	42
----	----	----

66	81	96
----	----	----

102	126	150
-----	-----	-----

```
>> disp(B.^2) ;
```

1	4	9
---	---	---

16	25	36
----	----	----

49	64	81
----	----	----

Đưa ra màn hình

- Ví dụ: Lập bảng tính giá trị hàm sin

```
clc  
n = input('Give number of point n = ') ;  
x = linspace(0,1,n) ;  
y = sin(2*pi*x) ;  
disp(' ')  
disp(' ! k ! x(k) ! sin(x(k)) ! ')  
disp('-----')  
for k=1:n degrees = (k-1)*360/(n-1) ;  
disp(sprintf(' ! %2.0f ! %3.0f ! %6.3f  
' ,k,degrees,y(k))) ;  
end  
disp(' ') ;  
disp('x(k) is given in degrees.')  
disp(sprintf('One Degree = %5.3e Radians' ,pi/180))
```

Đưa ra màn hình

- Kết quả thực hiện có thể có dạng:

Give number of point n = 10

!	k	!	x(k)	!	sin(x(k))	!
---	---	---	------	---	-----------	---

!	1	!	0	!	0.000	!
!	2	!	40	!	0.643	!
!	3	!	80	!	0.985	!
!	4	!	120	!	0.866	!
!	5	!	160	!	0.342	!
!	6	!	200	!	-0.342	!
!	7	!	240	!	-0.866	!
!	8	!	280	!	-0.985	!
!	9	!	320	!	-0.643	!
!	10	!	360	!	-0.000	!

x(k) is given in degrees.

One Degree = 1.745e-002 Radians

Đưa ra màn hình

- Lệnh **fprintf**
- Lệnh có dạng

`fprintf(dialog_format, danh_sach_bien)`

trong đó **dialog_format** là biến (hằng) xâu ký tự thông báo và các ký tự định khuôn dạng dữ liệu ra.

Đưa ra màn hình

- Ví dụ:

```
>> A = rand(3,3)
A =
    0.1389    0.6038    0.0153
    0.2028    0.2722    0.7468
    0.1987    0.1988    0.4451
>> fprintf('Length of matrix A: %i%i',length(A))
Length of matrix A: 3
>> fprintf('Square of A:\n
    %i%i%i\n%i%i%i\n%i%i%i\n',A^2)
Square of A:
    1.447541e-001 2.317550e-001 1.563636e-001
    2.512430e-001 3.449860e-001 2.625931e-001
    4.598234e-001 5.387547e-001 3.496177e-001
```

Vào ra với file văn bản

- Matlab cung cấp các lệnh cho phép làm việc với các file
- Ở đây chỉ hạn chế trình bày các thao tác với file văn bản
- Các lệnh chính liên quan đến làm việc với file văn bản của Matlab là:
 - fopen, fclose
 - frewind, fread, fwrite
 - fscanf, fprintf..

Vào ra với file văn bản

- **Lệnh fopen.** Dùng để mở file

fileID = fopen(FILENAME, PERMISSION)

- Lệnh này mở file văn bản có tên cho bởi FILENAME (hằng xâu ký tự hoặc biến xâu)
- Chế độ làm việc được chỉ ra bởi PERMISSION.
- PERMISSION có thể là:
 - ‘rt’ mở file để đọc
 - ‘wt’ mở file để ghi (nếu chưa có file sẽ tạo ra file mới có tên FILENAME)
 - ‘at’ mở file nối đuôi (tạo file mới có tên FILENAME nếu chưa có)
 - ‘rt+’ mở file để đọc và ghi (không tạo file mới)
- Biến nhận dạng *fileID* nhận giá trị nguyên và ta sẽ sử dụng nó để thâm nhập vào file.

Vào ra với file văn bản

- Trong trường hợp muốn kiểm tra lỗi mở file có thể sử dụng lệnh

`[FID, MESSAGE] = fopen(FILENAME, PERMISSION)`

nếu gặp lỗi mở file biến MESSAGE sẽ chứa thông báo lỗi.

Vào ra với file văn bản

- Ví dụ:

```
>> [fileID, MESSAGE] = fopen('ETU.txt','rt');  
>> fileID  
fileID =  
      -1  
>> MESSAGE  
MESSAGE =  
No such file or directory
```

Vào ra với file văn bản

- **Lệnh đóng file fclose:** Đóng file làm việc.

ST = fclose (FID)

- Lệnh này sẽ đóng file ứng với biến nhận dạng FID
- fclose trả lại biến ST giá trị 0 nếu nó hoàn thành việc đóng file và -1 nếu gặp lỗi.
- Lệnh ST=fopen('all') đóng tất cả các file đang mở ngoại trừ 0, 1, 2.
- **Lệnh frewind.**

frewind (FID)

đặt con trỏ file của FID vào đầu file.

Vào ra với file văn bản

- **Lệnh fscanf:** Đọc dữ liệu vào từ file

[A, COUNT] = fscanf(FID, FORMAT, SIZE)

- Lệnh này đọc dữ liệu từ file tương ứng với FID
- Chuyển đổi dữ liệu về khuôn dạng được xác định bởi biến xâu FORMAT
- Gán giá trị cho mảng A
- COUNT là biến ra tùy chọn dùng để chứa số lượng phần tử đọc được.
- SIZE là biến tùy chọn chỉ giới hạn số phần tử được đọc vào từ file, nếu vắng mặt thì toàn bộ file được xét. Các giá trị có thể có của biến là:
 - N : đọc không quá N phần tử từ file vào vector cột
 - Inf : đọc không quá kết thúc file
 - [M, N] : đọc không quá M*N phần tử và đưa vào ma trận kích thước không quá MxN theo từng cột, N có thể là inf nhưng M thì phải là hữu hạn.

Vào ra với file văn bản

- Nếu ma trận A là kết quả của việc chuyển khuôn dạng ký tự và biến SIZE không có dạng [M, N] thì vector dòng sẽ được trả lại.
- FORMAT là biến chứa các ký tự chuyển đổi khuôn dạng của ngôn ngữ C.
- Các ký tự định khuôn dạng bao gồm: %, các ký hiệu thay thế, độ dài trường, các ký tự chuyển đổi khuôn dạng: d, i, o, u, x, e, f, g, s, c và [...] (liệt kê tập hợp).
- Nếu %s được sử dụng thì khi đọc một phần tử có thể dẫn đến phải sử dụng một loạt các thành phần của ma trận, mỗi thành phần giữ một ký tự.
- Hãy sử dụng %c để đọc ký tự trắng (space) và khuôn dạng %s bỏ qua các ký tự trắng.

Vào ra với file văn bản

- Nếu chỉ thị định dạng gồm cả số lẫn ký tự thì ma trận kết quả sẽ là ma trận số và mỗi ký tự sẽ được chuyển thành số bằng giá trị mã ASCII của nó.
- fscanf khác với lệnh này trong C ở chỗ nó là lệnh vector hóa trả lại đối số là ma trận.
- Biến xâu định dạng sẽ được sử dụng lặp lại cho đến khi gặp kết thúc file hoặc đọc đủ số lượng phần tử chỉ ra bởi SIZE.

Vào ra với file văn bản

- Ví dụ:

- Lệnh

`S = fscanf(fid,'%s')`

đọc (và trả lại) một xâu.

- Lệnh

`A = fscanf(fid,'%5d')`

đọc các số nguyên có 5 chữ số thập phân.

Vào ra với file văn bản

- **Ví dụ:** Giả sử có file văn bản với tên “kq” chứa xâu “Day la ket qua dua ra”. Khi đó ta có thể đọc dữ liệu vào như sau:

```
>> fid=fopen('kq','rt')
fid =
      3
>> x = fscanf(fid,'%s')
x =
Daylaketquaduara
>>rewind(fid)
>> x = fscanf(fid,'%s%c')
x =
Day la ket qua dua ra
```

Vào ra với file văn bản

- **Ví dụ:** Giả sử file văn bản Matrix.txt chứa hai dòng

```
1 2 3 4 5  
6 7 8 9 10
```

- Hãy theo dõi kết quả làm việc của các lệnh sau để thấy tác động của lệnh fscanf

```
>> fopen ('matrix.txt' , 'rt')
```

```
ans =
```

```
4
```

```
>> A=fscanf(4 , '%i' , [2,5])
```

```
A =
```

1	3	5	7	9
2	4	6	8	10

Vào ra với file văn bản

```
>> frewind(4);  
>> B = fscanf(4, '%i', [5,2])  
B =  
    1     6  
    2     7  
    3     8  
    4     9  
    5    10  
  
>> frewind(4);  
>> C = fscanf(4, '%i', 6)  
C =  
    1  
    2  
    3  
    4  
    5  
    6
```

Vào ra với file văn bản

- **Lệnh fprintf :** ghi dữ liệu theo khuôn dạng ra file.

COUNT = FPRINTF(FID, FORMAT, A, ...)

- Lệnh này sẽ định dạng các thành phần của ma trận A (và các biến tiếp theo trong danh sách) theo khuôn dạng được xác định bởi biến xâu format, và ghi ra file tương ứng với biến nhận dạng FID.
- Biến:
 - COUNT đếm số byte dữ liệu được ghi ra.
 - FID là số nguyên nhận dạng tên file thu được từ lệnh fopen. Có thể dùng số 1 nếu sử dụng thiết bị ra chuẩn (màn hình). Nếu lệnh này không có FID thì mặc định đưa kết quả ra màn hình.
 - FORMAT là biến xâu chứa các ký tự mô tả định dạng dữ liệu giống như trong ngôn ngữ C
 - Các ký tự \n, \r, \t, \b, \f có thể dùng để tạo linefeed, carriage return, tab, backspace, và formfeed character tương ứng.
 - Sử dụng \\ để tạo dấu \ và %% để tạo ký tự %.

Vào ra với file văn bản

- Ví dụ:

```
x = 0:.1:1; y = [x; exp(x)];  
fid = fopen('exp.txt','w');  
fprintf(fid,'%6.2f %12.8f\r',y);  
fclose(fid);
```

Tạo ra file văn bản có tên ‘exp.txt’ chứa bảng giá trị của hàm mũ.

0.00 1.00000000

0.10	1.10517092
0.20	1.22140276
0.30	1.34985881
0.40	1.49182470
0.50	1.64872127

...



ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

CHƯƠNG 2: HỆ PHƯƠNG TRÌNH TUYẾN TÍNH

Vũ Văn Thiệu, Đinh Viết Sang, Nguyễn Khánh Phương

TÍNH TOÁN KHOA HỌC

Giới thiệu

- ① Thế nào là hệ phương trình tuyến tính ?
- ② Ví dụ 3 chiều
- ③ Ma trận hoán vị và ma trận tam giác
- ④ Phân tích LU
- ⑤ Vai trò của phần tử trụ
- ⑥ Hiệu ứng của sai số làm tròn
- ⑦ Hệ xác định tồi và số điều kiện của ma trận
 - Chuẩn ma trận
 - Số điều kiện ma trận
 - Đánh giá sai số khi biết số điều kiện của ma trận
- ⑧ Giải hệ phương trình tuyến tính bằng phân tích ma trận
- ⑨ Tổng kết

Hệ phương trình tuyến tính

Định nghĩa

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1$$

$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2$$

...

$$a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m$$

Ký hiệu

$A = (a_{ij})$ với $i = 1, \dots, m$ và $j = 1, \dots, n$ là ma trận hệ số A.

$b = (b_1, b_2, \dots, b_m)^T$ là vectơ về phải.

$x = (x_1, x_2, \dots, x_n)^T$ là vectơ biến.

ta có thể viết lại hệ phương trình tuyến tính dưới dạng ma trận

$$Ax = b$$

Hệ phương trình tuyến tính

Ví dụ 1 :

Xét hệ phương trình tuyến tính có

- Ma trận hệ số $A = \begin{pmatrix} 3 & 2 \\ 1 & -1 \end{pmatrix}$

- Vec tơ về phải là $b = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$

thì hệ có nghiệm duy nhất $x = \begin{pmatrix} 0.2 \\ -0.8 \end{pmatrix}$

Hệ phương trình tuyến tính

Ví dụ 2 :

Xét hệ phương trình tuyến tính có

- Ma trận hệ số $A = \begin{pmatrix} 1 & 0 & 3 \\ 0 & 1 & -5 \end{pmatrix}$

- Vec tơ về phải là $b = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$

thì hệ có vô số nghiệm $x = \begin{pmatrix} 1 - 3t \\ 2 + 5t \\ t \end{pmatrix}$ với mọi $t \in \mathbb{R}$.

Hệ phương trình tuyến tính

Ví dụ 3 :

Xét hệ phương trình tuyến tính có

- Ma trận hệ số $A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 3 & 4 \end{pmatrix}$

- Vec tơ về phải là $b = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$

thì hệ vô nghiệm.

Hệ phương trình tuyến tính

Đối với hệ phương trình tuyến tính có thể xảy ra

- $m = n$: hệ vuông (số phương trình bằng số ẩn, thường có nghiệm duy nhất)
- $m < n$: hệ thiêú (số phương trình ít hơn số ẩn số, hệ thường vô số nghiệm)
- $m > n$: hệ dư (số phương trình nhiều hơn số ẩn số, hệ thường vô nghiệm)

Giải hệ phương trình tuyến tính

Hệ phương trình vuông

$$Ax = b$$

trong đó $A \in \mathbb{R}^{n \times n}$ còn x và b là các vec tơ $\in \mathbb{R}^n$

Giải hệ phương trình vuông

- Nếu ma trận A không suy biến (singular) thì nghiệm duy nhất của phương trình là

$$x = A^{-1}b$$

Matlab

» **$x=inv(A)*b$**

Giải hệ phương trình tuyến tính

Ví dụ 4 :

Giải hệ phương trình $A = (7)$ và $b = (21)$ hay phương trình

$$7x = 21$$

- Cách 1 : Giải trực tiếp phép chia $x = 21/7 = 3$
- Cách 2 : Nghịch đảo 7^{-1} rồi nhân với 21 sẽ dẫn đến

$$x = 7^{-1} \times 21 = 0.142857 \times 21 = 2.99997$$

Rõ ràng cách 1 tốt hơn cách 2, thêm nữa cách 2 còn có khôi lượng tính toán lớn hơn khi xác định nghịch đảo 7^{-1} .

Giải hệ phương trình tuyến tính

Nhận xét

Việc sử dụng ma trận nghịch đảo cho lời giải kém chính xác.

Khi giải HPT tuyến tính ta thường tìm lời giải trực tiếp và chỉ trong một số ít tình huống mới dùng đến ma trận nghịch đảo A^{-1} .

Một số phương pháp:

- Phân tích LU (LU Factorization)
- Phân tích Cholesky (Cholesky Factorization)
- Phân rã QR (QR Decomposition)

Giải hệ phương trình tuyến tính

Toán tử chia ma trận trong Matlab

Nếu A là ma trận bất kỳ và B là ma trận có số hàng giống A thì lời giải của hệ phương trình

$$AX = B$$

thì ta dùng phép *chia trái* $X = A \setminus B$.

Còn lời giải của hệ phương trình

$$XA = B$$

thì ta dùng phép *chia phải* $X = B / A$.

Giải hệ phương trình tuyến tính

Ví dụ 5 : Chia trái

» $A = [3 \ 2; 1 \ -1]; b = [-1; 1];$

» $x = A \setminus b;$

$x = 0.2000$

-0.8000

Ví dụ 6 : Chia phải

» $AA = A'; bb = b';$

» $xx = bb / AA;$

$xx = 0.2000 \quad -0.8000$

Giải hệ phương trình tuyến tính

Các định lượng cơ bản khi giải hệ phương trình vuông

- **Định thức (Determinant)** là đặc trưng số quan trọng của ma trận vuông, cho phép xác định số nghiệm của HPT (vô nghiệm hoặc vô số nghiệm, hay nghiệm duy nhất).
- **Vết (Trace)** là tổng các phần tử đường chéo chính.
- **Hạng (Rank)** là số dòng hay cột độc lập tuyến tính lớn nhất của ma trận.

Matlab

```
» D=det(A)  
» T=trace(A)  
» R=rank(A)
```

Giải hệ phương trình tuyến tính

Định lý Kronecker-Capelli

Hệ phương trình tuyến tính $Ax = b$ có nghiệm khi và chỉ khi

$$\text{rank}(A) = \text{rank}(Ab)$$

Ví dụ 7 : cùng hạng

» $A=[1\ 2\ 3; 4\ 5\ 6; 8\ 10\ 12];$

» $b=[5;6;12];$

» $rA=\text{rank}(A);$

» $rAb=\text{rank}([Ab])$

$rA = 2$

$rAb = 2$

Giải hệ phương trình tuyến tính vuông

Các trường hợp khi giải $Ax = b$

- Hệ phương trình có nghiệm duy nhất nếu $\det(A) \neq 0$.
- Khi $\det(A) = 0$ hệ phương trình có thể có vô số nghiệm hoặc vô nghiệm (Ta có thể áp dụng định lý Kronecker-Capelli để xác định rõ nó vô nghiệm hay vô số nghiệm).
- Khi $\det(A) \neq 0$ thì tồn tại ma trận nghịch đảo của A và A được gọi là ma trận không suy biến.
- Khi $\det(A) = 0$ thì ma trận nghịch đảo A^{-1} không tồn tại và A được gọi là ma trận suy biến.

Giải hệ phương trình tuyến tính vuông

Ví dụ 8 :

» $A1=[-1\ 1; -2\ 2]; b1=[1\ ; 0];$

% hệ vô nghiệm

» $x1 = A1 \setminus b1, D1 = \det(A1)$

Warning : Matrix is singular to working precision.

$x1 = Inf$

Inf

$D1=0$

Giải hệ phương trình tuyến tính vuông

Ví dụ 9 :

» $A2=[-1\ 1; -2\ 2]; b2=[1\ ; 2];$

% hệ vô số nghiệm

» $x2 = A2 \setminus b2$

Warning : Matrix is singular to working precision.

$x2 = -1$

0

Giải hệ phương trình tuyến tính vuông



- 1 Thê nào là hệ phương trình tuyến tính ?
- 2 Ví dụ 3 chiều
- 3 Ma trận hoán vị và ma trận tam giác
- 4 Phân tích LU
- 5 Vai trò của phần tử trụ
- 6 Hiệu ứng của sai số làm tròn
- 7 Hệ xác định tồi và số điều kiện của ma trận
 - Chuẩn ma trận
 - Số điều kiện ma trận
 - Đánh giá sai số khi biết số điều kiện của ma trận
- 8 Giải hệ phương trình tuyến tính bằng phân tích ma trận
- 9 Tổng kết

Giải hệ phương trình tuyến tính vuông

Ví dụ 3 chiều

Cho hệ phương trình cấp 3 sau :

$$\begin{pmatrix} 10 & -7 & 0 \\ -3 & 2 & 6 \\ 5 & -1 & 5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 7 \\ 4 \\ 6 \end{pmatrix}$$

ta viết lại dưới dạng hệ phương trình tuyến tính

$$\begin{aligned} 10x_1 - 7x_2 &= 7 \\ -3x_1 + 2x_2 + 6x_3 &= 4 \\ 5x_1 - x_2 + 5x_3 &= 6 \end{aligned}$$

Giải hệ phương trình tuyến tính vuông

Ví dụ 3 chiều (tiếp)

Hệ phương trình tuyến tính

$$10x_1 - 7x_2 = 7 \quad (1)$$

$$-3x_1 + 2x_2 + 6x_3 = 4 \quad (2)$$

$$5x_1 - x_2 + 5x_3 = 6 \quad (3)$$

ta tiến hành giải

- Khử $x_1 \Rightarrow (2) - (1) \times (-0.3)$ và $(3) - (1) \times 0.5$

Hệ số 10 của ẩn x_1 trong (1) đc gọi là **phần tử trù**, các hệ số -0.3 và 0.5 đc gọi là **nhân tử**.

Giải hệ phương trình tuyến tính vuông

Ví dụ 3 chiều (tiếp)

Hệ phương trình tuyến tính sau khi khử x_1

$$10x_1 - 7x_2 = 7 \quad (4)$$

$$-0.1x_2 + 6x_3 = 6.1 \quad (5)$$

$$2.5x_2 + 5x_3 = 2.5 \quad (6)$$

ta tiếp tục giải

- Khử $x_2 \Rightarrow$ do phần tử trụ của x_2 trong (5) là -0.1 có trị tuyệt đối nhỏ, ta tiến hành đổi chỗ hai phương trình (5) và (6) rồi mới tiến hành khử x_2 .

Việc làm này gọi là **phép xoay**

Giải hệ phương trình tuyến tính vuông

Ví dụ 3 chiều (tiếp)

Hệ phương trình tuyến tính sau khi thực hiện phép xoay

$$10x_1 - 7x_2 = 7 \quad (7)$$

$$2.5x_2 + 5x_3 = 2.5 \quad (8)$$

$$-0.1x_2 + 6x_3 = 6.1 \quad (9)$$

ta tiếp tục giải

- Khử $x_2 \Rightarrow (9) - (8) \times (-0.04)$

Giải hệ phương trình tuyến tính vuông

Ví dụ 3 chiều (tiếp)

Hệ phương trình tuyến tính sau khi thực hiện phép khử x_2

$$10x_1 - 7x_2 = 7 \quad (10)$$

$$2.5x_2 + 5x_3 = 2.5 \quad (11)$$

$$6.2x_3 = 6.2 \quad (12)$$

ta tiếp tục giải

- Từ phương trình (12) $\Rightarrow x_3 = 1$.
- thay x_3 vào (11) thì $2.5x_2 + 5 \times (1) = 2.5 \Rightarrow x_2 = -1$.
- thay x_2 vào (10) thì $10x_1 - 7 \times (-1) = 7 \Rightarrow x_1 = 0$.

Giải hệ phương trình tuyến tính vuông

Các ma trận L,U,P

Toàn bộ cách giải vừa trình bày có thể được gói gọn trong các ma trận sau

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 0.5 & 1 & 0 \\ -0.3 & -0.04 & 1 \end{pmatrix}, U = \begin{pmatrix} 10 & -7 & 0 \\ 0 & 2.5 & 5 \\ 0 & 0 & 6.2 \end{pmatrix}, P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Với

- L là ma trận chứa các nhân tử
- U là ma trận hệ số cuối cùng
- P là ma trận hoán vị mô tả phép xoay

thì chúng ta có

$$LU = PA$$

Giải hệ phương trình tuyến tính vuông



- 1 Thê nào là hệ phương trình tuyến tính ?
- 2 Ví dụ 3 chiều
- 3 Ma trận hoán vị và ma trận tam giác
- 4 Phân tích LU
- 5 Vai trò của phần tử trù
- 6 Hiệu ứng của sai số làm tròn
- 7 Hệ xác định tồi và số điều kiện của ma trận
 - Chuẩn ma trận
 - Số điều kiện ma trận
 - Đánh giá sai số khi biết số điều kiện của ma trận
- 8 Giải hệ phương trình tuyến tính bằng phân tích ma trận
- 9 Tổng kết

Giải hệ phương trình tuyến tính vuông

Ma trận hoán vị

Ma trận hoán vị là thu được từ ma trận đơn vị / bằng cách hoán vị các hàng của nó.

- Với ma trận hoán vị : $P^{-1} = P^T$
- Phép nhân PX dùng để hoán vị hàng ma trận X
- Phép nhân XP dùng để hoán vị cột ma trận X

Ví dụ 10

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Giải hệ phương trình tuyến tính vuông

Ví dụ 10 (tiếp)

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Dạng rút gọn trong Matlab

$$p = [2 \ 4 \ 3 \ 1]$$

Giải hệ phương trình tuyến tính vuông

Ma trận tam giác

Ma trận $X \in \mathbb{R}^{n \times n}$ là **ma trận tam giác trên** nếu $x_{ij} = 0 \in X \quad \forall i < j$ nghĩa là có dạng

$$X = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ 0 & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & x_{nn} \end{pmatrix}$$

- Khi ma trận này có các phần tử đường chéo chính $x_{ii} = 1 \quad \forall i = 1, \dots, n$ thì được gọi là **ma trận tam giác trên đơn vị**.
- Định thức ma trận tam giác trên khác không khi và chỉ khi tất cả các phần tử nằm trên đường chéo chính là khác không.

⇒ Định nghĩa tương tự ta có **ma trận tam giác dưới** và **ma trận tam giác dưới đơn vị**.

Giải hệ phương trình tuyến tính vuông

Ma trận tam giác

Hệ phương trình với ma trận hệ số tam giác có thể giải dễ dàng. Bắt đầu giải phương trình hàng cuối cùng để tìm ẩn số cuối; sau đó lần lượt thay vào các phương trình trên để tìm ra các ẩn số còn lại.

Ví dụ 11 :

Giải hệ phương trình tam giác trên $Ux = b$

»x = zeros(n,1);

for k = n:-1:1

x(k) = b(k)/U(k,k);

i=(1:k-1)';

b(i) = b(i) - x(k) * U(i,k);

end

Giải hệ phương trình tuyến tính vuông

Ví dụ 12 :

Giải hệ phương trình tuyến tính

$$3x_1 + 4x_2 + 5x_3 = 7$$

$$2x_2 - 3x_3 = 8$$

$$5x_3 = 11$$

% Chương trình Matlab

```
» U=[3,4,5;0,2,-3;0,0,5]; b = [7;8;11];n=3;x=zeros(n,1);
» for k=n:-1:1
»     x(k) = b(k)/U(k,k);
»     i=(1:k-1)';
»     b(i) = b(i) - x(k) * U(i,k);
» end;
```

- 1 Thê nào là hệ phương trình tuyến tính ?
- 2 Ví dụ 3 chiều
- 3 Ma trận hoán vị và ma trận tam giác
- 4 Phân tích LU
- 5 Vai trò của phần tử trù
- 6 Hiệu ứng của sai số làm tròn
- 7 Hệ xác định tồi và số điều kiện của ma trận
 - Chuẩn ma trận
 - Số điều kiện ma trận
 - Đánh giá sai số khi biết số điều kiện của ma trận
- 8 Giải hệ phương trình tuyến tính bằng phân tích ma trận
- 9 Tổng kết

Giải hệ phương trình tuyến tính vuông

Phân tích LU

Thuật toán phổ biến dùng để giải hệ phương trình tuyến tính vuông có hai giai đoạn

- Khử xuôi (Forward elimination) chính là phép chuyển ma trận vuông về dạng tam giác trên dùng để khử từng ẩn số, với nhân tử, phần tử trụ tương thích kết hợp phép xoay.
 - gồm $n - 1$ bước
 - tại bước $k = 1, \dots, n - 1$ nhân phương trình thứ k với nhân tử rồi trừ các phương trình còn lại để khử ẩn số x_k .
 - Nếu hệ số của x_k nhỏ thì ta nên đổi chỗ các phương trình.
- Thế ngược (Backward substitution) giải phương trình hàng cuối cùng để tìm ẩn cuối cùng, sau đó thế ngược lần lượt lên các hàng trên để tìm ra các ẩn còn lại. (xem lại ví dụ 12)

Giải hệ phương trình tuyến tính vuông

Phân tích LU (tiếp)

- Gọi P_k là các ma trận hoán vị tại các bước $k = 1, \dots, n - 1$
- Gọi M_k là các ma trận tam giác dưới đơn vị thu được bằng cách chèn các nhân tử "cộng" được sử dụng ở bước k xuống dưới vị trí đường chéo của cột k của ma trận đơn vị.
- Gọi U là ma trận tam giác trên thu được cuối cùng khi kết thúc giai đoạn khử xuôi.

Quá trình khử được viết lại dưới dạng ma trận như sau

$$U = M_{n-1}P_{n-1} \cdots M_1P_1A$$

Giải hệ phương trình tuyến tính vuông

Phân tích LU (tiếp)

Phương trình có thể viết lại tương đương như sau

$$L_1 L_2 \cdots L_{n-1} U = P_{n-1} \cdots P_1 A$$

trong đó L_k thu được từ M_k bằng cách hoán vị và đổi dấu các nhân tử ở dưới đường chéo. Vậy nếu ta đặt

$$L = L_1 L_2 \cdots L_{n-1}$$

$$P = P_{n-1} \cdots P_2 P_1$$

thì ta thu được công thức ban đầu

$$LU = PA$$

Giải hệ phương trình tuyến tính vuông

Ví dụ 12 :

Quay lại ví dụ 3 chiều đầu tiên, ta có $A = \begin{pmatrix} 10 & -7 & 0 \\ -3 & 2 & 6 \\ 5 & -1 & 5 \end{pmatrix}$ thì các ma trận xác định trong quá trình khử xuôi là

$$P_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, M_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0.3 & 1 & 0 \\ -0.5 & 0 & 1 \end{pmatrix}$$

$$P_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, M_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0.04 & 1 \end{pmatrix}$$

Giải hệ phương trình tuyến tính vuông

Ví dụ 12 (tiếp):

Các ma trận L_1, L_2 tương ứng là

$$L_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0.5 & 1 & 0 \\ -0.3 & 0 & 1 \end{pmatrix}, L_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -0.04 & 1 \end{pmatrix}$$

Chú ý :

Khi tính toán giai đoạn khử, ta sẽ tính toán trực tiếp trên các hàng của ma trận chứ không thực hiện phép nhân ma trận như trên.

Phân tích LU

- Hệ thức $LU = PA$ vừa trình bày được gọi là *phân tích LU* hay *phân rã tam giác* của ma trận A .

Giải hệ phương trình tuyến tính vuông

Giải hệ phương trình bằng phân tích LU

Với hệ phương trình

$$Ax = b$$

với ma trận A là không suy biến đồng thời $PA = LU$ là phân tích LU của A thì hệ phương trình có thể được giải bởi hai bước.

- **Khử xuôi** Giải hệ

$$Ly = Pb$$

để tìm y , do L là ma trận đơn vị dưới nên y có thể tìm được nhờ phép khử xuôi (từ trên xuống dưới).

- **Thế ngược** Giải hệ

$$Ux = y$$

bằng cách thế ngược để tìm được x .

- 1 Thê nào là hệ phương trình tuyến tính ?
- 2 Ví dụ 3 chiều
- 3 Ma trận hoán vị và ma trận tam giác
- 4 Phân tích LU
- 5 Vai trò của phần tử trụ
- 6 Hiệu ứng của sai số làm tròn
- 7 Hệ xác định tồi và số điều kiện của ma trận
 - Chuẩn ma trận
 - Số điều kiện ma trận
 - Đánh giá sai số khi biết số điều kiện của ma trận
- 8 Giải hệ phương trình tuyến tính bằng phân tích ma trận
- 9 Tổng kết

Vai trò của phần tử trụ

Phần tử trụ

- Các phần tử nằm trên đường chéo chính của ma trận U .
- Phần tử trụ thứ k là hệ số của ẩn x_k trong phương trình thứ k tại bước k của giai đoạn khử.
- Trong cả hai bước khử xuôi và thế ngược đều cần chia cho phần tử trụ nên chúng không thể có giá trị không.

Trực giác :

hệ phương trình giải tồi nếu phần tử trụ gần không.

Vai trò của phần tử trù

Ví dụ 13 :

Thay đổi chút ít tại hàng hai trong các ví dụ trên

$$\begin{pmatrix} 10 & -7 & 0 \\ -3 & 2.099 & 6 \\ 5 & -1 & 5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 7 \\ 3.901 \\ 6 \end{pmatrix}$$

Như vậy, giả sử mọi tính toán được thực hiện trên máy tính giả định trang bị phép toán số học với số thực dấu phẩy động 5 chữ số.

- Hệ số ẩn x_2 tại hàng hai thay đổi từ 2.000 thành 2.099
- Đồng thời về phải tương ứng thay đổi từ 4.000 thành 3.901

mục đích là giữ nguyên nghiệm $(0, -1, 1)^T$ của hệ phương trình.

Vai trò của phần tử trù

Ví dụ 13 (tiếp) :

Bước đầu tiên của giai đoạn khử

$$\begin{pmatrix} 10 & -7 & 0 \\ 0 & -0.001 & 6 \\ 0 & 2.5 & 5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 7 \\ 6.001 \\ 2.5 \end{pmatrix}$$

tiếp tục thực hiện khử dù phần tử trù dù -0.001 là nhỏ so với các hệ số khác của ma trận mà không thực hiện phép xoay. Vậy ta

- Nhân phương trình hàng thứ hai với 2.5×10^3 rồi cộng với hàng thứ ba.
- Về phải của phương trình này, khi nhân 6.001 với 2.5×10^3 thì kết quả 1.50025×10^4 làm tròn thành 1.5002×10^4

Vai trò của phần tử trù

Ví dụ 13 (tiếp) :

$$\begin{pmatrix} 10 & -7 & 0 \\ 0 & -0.001 & 6 \\ 0 & 0 & 1.5005 \times 10^4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 7 \\ 6.001 \\ 1.5004 \times 10^4 \end{pmatrix}$$

tiếp tục ...

- Kết quả về phải phương trình hai làm tròn 1.5002×10^4 được cộng với 2.5 là về phải của phương trình thứ ba và lại được làm tròn.

Vậy phương trình ba trở thành $1.5005 \times 10^4 x_3 = 1.5004 \times 10^4$ giải ra ta có

$$x_3 = \frac{1.5004 \times 10^4}{1.5005 \times 10^4} = 0.99993$$

Rõ ràng, với giá trị chính xác của ẩn số $x_3 = 1$ thì giá trị giải được bởi phương trình này chưa đáng ngại.

Vai trò của phần tử trù

Ví dụ 13 (tiếp) :

$$\begin{pmatrix} 10 & -7 & 0 \\ 0 & -0.001 & 6 \\ 0 & 0 & 1.5005 \times 10^4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 7 \\ 6.001 \\ 1.5004 \times 10^4 \end{pmatrix}$$

tiếp tục ...

- đối với ẩn số x_2

$$-0.001x_2 + 6 \times (0.99993) = 6.001$$

$$\text{nên } x_2 = \frac{1.5 \times 10^{-3}}{-1.0 \times 10^{-3}} = -1.5$$

- Cuối cùng thế lên phương trình đầu tìm ẩn x_1

$$10x_1 - 7 \times (-1.5) = 7$$

suy ra $x_1 = -3.5$

Vai trò của phần tử trù

Ví dụ 13 (tiếp) :

Như vậy khi không thực hiện phép xoay chọn phần tử trù

$$\begin{pmatrix} 10 & -7 & 0 \\ -3 & -0.001 & 6 \\ 5 & 2.5 & 5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 7 \\ 6.001 \\ 2.5 \end{pmatrix}$$

thay vì được nghiệm $(0, -1, 1)^T$ ta lại được nghiệm
 $(-0.35, -1.5, 0.99993)^T$.

Vì sao có sự cố này ?

Sai số là do chúng ta chọn **phần tử trù quá nhỏ**. Vậy ta nên chọn phần tử trù có trị tuyệt đối lớn nhất tại mỗi bước khử k.

- ① Thế nào là hệ phương trình tuyến tính ?
- ② Ví dụ 3 chiều
- ③ Ma trận hoán vị và ma trận tam giác
- ④ Phân tích LU
- ⑤ Vai trò của phần tử trù
- ⑥ Hiệu ứng của sai số làm tròn
- ⑦ Hệ xác định tồi và số điều kiện của ma trận
 - Chuẩn ma trận
 - Số điều kiện ma trận
 - Đánh giá sai số khi biết số điều kiện của ma trận
- ⑧ Giải hệ phương trình tuyến tính bằng phân tích ma trận
- ⑨ Tổng kết

Hiệu ứng của sai số làm tròn

Cách đo sự khác biệt

Thông thường, khi thu được lời giải x^* khác với lời giải đúng x ta thường sử dụng hai cách đo sự khác biệt

- Sai số : $e = x - x^*$
- Độ lệch : $r = b - Ax^*$

Về lý thuyết nếu A không suy biến thì hai đại lượng này cùng bằng không, tuy nhiên khi tính toán trong máy tính hai đại lượng này không đồng điệu.

Hiệu ứng của sai số làm tròn

Ví dụ 14 :

Xét hệ phương trình

$$0.780x_1 + 0.563x_2 = 0.217$$

$$0.913x_1 + 0.659x_2 = 0.254$$

Khử Gauss như ví dụ trước, áp dụng quy tắc chọn phần tử trụ lớn nhất tuy nhiên mọi tính toán chỉ chính xác đến 3 chữ số thập phân.

Hiệu ứng của sai số làm tròn

Ví dụ 14 (tiếp):

tiếp tục

- Thực hiện phép xoay, để 0.913 trở thành phần tử trụ.

$$0.913x_1 + 0.659x_2 = 0.254$$

$$0.780x_1 + 0.563x_2 = 0.217$$

Tính hệ số $0.780/0.913 = 0.854$

- Nhân hệ số 0.854 với pt thứ nhất rồi trừ đi pt thứ hai. Ta được

$$0.913x_1 + 0.659x_2 = 0.254$$

$$0.001x_2 = 0.001$$

Hiệu ứng của sai số làm tròn

Ví dụ 14 (tiếp):

$$0.913x_1 + 0.659x_2 = 0.254$$

$$0.001x_2 = 0.001$$

tiếp tục

- Ấn $\hat{x}_2 = 0.001/0.001 = 1.000$ (chính xác)
- Thế lên pt trên, $x_1 = (0.254 - 0.659\hat{x}_2)/0.913 = -0.443$

Cuối cùng ta thu được nghiệm $x^* = (-0.443, 1.000)^T$

Hiệu ứng của sai số làm tròn

Ví dụ 14 (tiếp):

Đo sự khác biệt, rõ ràng nghiệm đúng của hệ $x = (1, -1)^T$

- Sai số : $e = x - x^* = (1.433, -2)^T$
- Độ lệch :

$$\begin{aligned} r = b - Ax^* &= \begin{pmatrix} 0.217 - (0.780(-0.443) + 0.563(1.000)) \\ 0.254 - (0.913(-0.443) + 0.659(1.000)) \end{pmatrix} \\ &= \begin{pmatrix} -0.000460 \\ -0.000541 \end{pmatrix} \end{aligned}$$

Rõ ràng trong khi độ lệch chấp nhận được khi ta làm tròn sai số 3 số thập phân sau dấu phẩy thì sai số thậm chí còn lớn hơn cả lời giải.

Hiệu ứng của sai số làm tròn

Các câu hỏi đặt ra khi có hiện tượng sai số khi làm tròn

- Vì sao độ lệch lại có giá trị nhỏ?
- Vì sao sai số lại có giá trị quá lớn?
- Định thức của hệ $0.780 \times 0.659 - 0.913 \times 0.563 = 10^{-6}$ gần không có phải là nguyên nhân gây hiện tượng này?

Hiệu ứng của sai số làm tròn

Ví dụ 14 (tiếp):

Thay giả thiết làm tròn với 3 số thập phân thành làm tròn với 6 số thập phân sau dấu phẩy, ta thu được hệ phương trình sau khi khử Gauss

$$\begin{pmatrix} 0.913000 & 0.659000 \\ 0.000000 & 0.000001 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0.254000 \\ -0.000001 \end{pmatrix}$$

Chú ý, giá trị về phải của pt thứ hai đã thay đổi. Thực tế **nghiệm gần đúng cũng là nghiệm chính xác** của hệ

$$x_1 = \frac{-0.000001}{0.000001} = -1.000000$$

$$x_2 = \frac{0.254 - 0.659x_1}{0.913} = 1.000000$$

Hiệu ứng của sai số làm tròn

Giải thích tại sao độ lệch nhỏ

- Độ lệch của phương trình hai nhỏ do ma trận gần suy biến $\det(A) = 10^{-6}$ dẫn đến hai phương trình gần như phụ thuộc tuyến tính.
 - Vì thế cặp ẩn (x_1, x_2) thỏa mãn phương trình thứ nhất cũng thỏa mãn phương trình thứ hai
- ⇒ Nếu như ta biết chắc định thức bằng không thì ta không cần phải quan tâm đến phương trình thứ hai vì mọi nghiệm của hệ phương trình thứ nhất đều thỏa mãn hệ phương trình thứ hai.

Kết luận quan trọng :

Khi ta tiến hành khử Gauss với phần tử trụ tối đại trên cột **đảm bảo** cho độ lệch $r = b - Ax^*$ là nhỏ.

Hiệu ứng của sai số làm tròn



Cài đặt thuật toán trên Matlab

```
function [L, U ,p]=lutx(A)
[n,n]=size(A);
p=(1:n)';
for k=1:n-1
    [r,m]=max(abs(A(k:n,k)));
    m=m+k-1;
    if (A(m,k) ~= 0)
        if (m ~= k)
            A([k m], :)=A([m k], :);
            p([k m])=p([m k]);
        end
        i=k+1:n;
        A(i,k)=A(i,k)/A(k,k);
        j=k+1:n;
        A(i,j)=A(i,j)-A(i,k)*A(k,j);
    end
end
L=tril(A,-1)+eye(n,n);
U=triu(A);
end
```

Bài tập

Viết hàm bslashtx cài đặt phép toán chia trái (được đơn giản hóa) của MatLab giải hệ phương trình tuyến tính.

```
function x=bslashtx(A,b)
n=size(A,1);
%Su dung lutx(A);
...
%Khu xuoi
...
%The nguoc
...
```

- 1 Thê nào là hệ phương trình tuyến tính ?
- 2 Ví dụ 3 chiều
- 3 Ma trận hoán vị và ma trận tam giác
- 4 Phân tích LU
- 5 Vai trò của phần tử trù
- 6 Hiệu ứng của sai số làm tròn
- 7 Hệ xác định tồi và số điều kiện của ma trận
 - Chuẩn ma trận
 - Số điều kiện ma trận
 - Đánh giá sai số khi biết số điều kiện của ma trận
- 8 Giải hệ phương trình tuyến tính bằng phân tích ma trận
- 9 Tổng kết

Hệ xác định tồi và số điều kiện của ma trận

Các hệ số ít khi được biết tới một cách chính xác

bởi vì

- Đối với các hệ phương trình xuất hiện trong ứng dụng, các hệ số thường được quy cho giá trị thực nghiệm nên gắn với sai số quan sát.
- Nhiều hệ phương trình khác có các hệ số được tính bởi các công thức và vì thế chúng được biết chính xác tới sai số làm tròn khi tính toán theo công thức đã cho.
- Ngay cả đôi với các hệ phương trình được cất giữ chính xác trong máy tính cũng không thể tránh khỏi sai số (Xem lại các cách biểu diễn số nguyên, số nguyên có dấu, số thực dấu phẩy động trong giáo trình Tin học đại cương)

Vậy câu hỏi được đặt ra là :

Nếu có sai số trong biểu diễn các hệ số của hệ phương trình tuyến tính thì điều đó ảnh hưởng đến lời giải như thế nào ? Hay khi giải $Ax = b$ làm sao có thể đo được độ nhạy của x khi có thay đổi trong A, b ?

Hệ xác định tồi và số điều kiện của ma trận

Có một vài nhận xét sau

- Nếu A suy biến thì đối với b nào đó x hoặc vô nghiệm hoặc vô số nghiệm. Trong trường hợp A có định thức nhỏ thì một sự thay đổi nhỏ trong A và b có thể dẫn sự thay đổi lớn trong lời giải.
- Hãy nghĩ đến kích thước các phần tử trụ và khái niệm gần suy biến. Bởi vì nếu các phép toán số học được thực hiện chính xác thì tất cả các phần tử trụ khác không khi và chỉ khi ma trận không suy biến. Từ đó rút ra khẳng định sau : 'Nếu các phần tử trụ là nhỏ thì ma trận gần suy biến' điều ngược lại không đúng, hay nói cách khác có ma trận gần suy biến mà các phần tử trụ đều không nhỏ.

Hệ xác định tồi và số điều kiện của ma trận

Chuẩn vec tơ

Định nghĩa : Hàm $v : \mathbb{R}^n \mapsto \mathbb{R}$ được gọi là chuẩn vec tơ (vector norm) trên \mathbb{R}^n khi và chỉ khi

- ① $v(x) \geq 0 \quad \forall x \in \mathbb{R}^n$ và $v(x) = 0$ khi và chỉ khi $x = 0$
- ② $v(\alpha x) = |\alpha|v(x) \quad \forall x \in \mathbb{R}^n, \forall \alpha \in \mathbb{R}$
- ③ $v(x + y) \leq v(x) + v(y) \quad \forall x, y \in \mathbb{R}^n$ đây là bất đẳng thức tam giác.

Thông thường $v(x)$ được ký hiệu là $\|x\|$

Hệ xác định tồi và số điều kiện của ma trận

Chuẩn vec tơ (tiếp)

Một số chuẩn thường dùng

- $\|x\|_2 = \sqrt{\sum_{i=1}^n x_i^2}$ là (l_2) hay chuẩn Euclid
- $\|x\|_1 = \sum_{i=1}^n |x_i|$ là (l_1)
- $\|x\|_\infty = \max_{1 \leq i \leq n} |x_i|$ là (l_∞)
- $\|x\|_p = (\sum_{i=1}^n |x_i|^p)^{1/p}$ là (l_p)

Matlab

norm(x,p) dùng cho l_p

còn với $p = 2$ thì hàm đơn giản hơn **norm(x)**

Hệ xác định tồi và số điều kiện của ma trận

Chuẩn ma trận

Định nghĩa : Hàm $\|.\| : \mathbb{R}^{n \times n} \mapsto \mathbb{R}$ được gọi là chuẩn ma trận nếu

$$\|A\| = \max_{\|x\|=1, x \in \mathbb{R}^n} \|Ax\| = \max_{\|x\| \neq 0, x \in \mathbb{R}^n} \frac{\|Ax\|}{\|x\|}$$

trong đó $\|Ax\|$ là chuẩn của vec tơ Ax . Tất nhiên, ta có bất đẳng thức $\|Ax\| \leq \|A\| \|x\|$

Hệ xác định tồi và số điều kiện của ma trận

Chuẩn ma trận (tiếp)

Các tính chất của chuẩn ma trận

- ① $\|A\| \geq 0; \|A\| = 0$ khi và chỉ khi $A = 0$.
- ② $\|\alpha A\| = |\alpha| \|A\|, \alpha \in \mathbb{R}$
- ③ $\|A + B\| \leq \|A\| + \|B\|$
- ④ $\|AB\| \leq \|A\| \times \|B\|$
- ⑤ $\|Ax\| \leq \|A\| \|x\|$

Hệ xác định tồi và số điều kiện của ma trận

Chuẩn ma trận (tiếp)

Các chuẩn vec tơ sinh ra các chuẩn ma trận tương ứng

- Chuẩn Euclid : $\|A\|_2 = \max_{\|x\|_2=1} \|Ax\|_2$
- Chuẩn max "tổng dòng" :
$$\|A\|_\infty = \max_{\|x\|_\infty=1} \|Ax\|_\infty = \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}|$$
- Chuẩn max "tổng cột" :
$$\|A\|_1 = \max_{\|x\|_1=1} \|Ax\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^n |a_{ij}|$$
- Chuẩn Frobenius : $\|A\|_F = (\text{Tr}(A^T A)^{1/2}) = \left(\sum_{i,j=1}^n a_{ij}^2 \right)^{1/2}$

Trong Matlab

norm(A,p) trong đó $p = 1, 2, \text{inf}$

Hệ xác định tồi và số điều kiện của ma trận

Số điều kiện của ma trận

Định nghĩa : Số điều kiện (condition number) $\text{cond}(A)$, thường được ký hiệu là $\kappa_p(A)$, của ma trận vuông A tính đối với một chuẩn ma trận p cho trước là số

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

trong đó, ta quy ước $\text{cond}(A) = \infty$ khi A là suy biến. Bởi vì,

$$\|A\| \cdot \|A^{-1}\| = \frac{\max_{x \neq 0} \frac{\|Ax\|}{\|x\|}}{\min_{x \neq 0} \frac{\|Ax\|}{\|x\|}}$$

nên số điều kiện đo tỷ số giữa độ giãn nở lớn nhất và độ co hẹp lớn nhất mà ma trận có thể tác động đối với một vec tơ khác không.

Hệ xác định tồi và số điều kiện của ma trận

Số điều kiện của ma trận (tiếp)

Số điều kiện cho biết ma trận gần suy biến đến mức độ nào : ma trận càng lớn càng **gần suy biến** (hệ phương trình tương ứng xác định tồi), trái lại ma trận với số điều kiện càng gần 1 càng xa với **gần suy biến**.

Chú ý :

Định thức ma trận không là đặc trưng tốt cho tính gần suy biến. Mặc dù khi $\det(A) = 0$ thì ma trận suy biến nhưng độ lớn hay nhỏ của định thức không chứa thông tin về việc ma trận có gần suy biến hay không.

Ví dụ, cho ma trận $\det(\alpha \mathbb{I}_n) = \alpha^n$ có thể là số rất nhỏ khi $|\alpha| < 1$ nhưng ma trận $\alpha \mathbb{I}_n$ lại có điều kiện tốt với $\text{cond}(\alpha \mathbb{I}_n) = 1$. Trong đó \mathbb{I}_n là ma trận đơn vị n chiều.

Hệ xác định tồi và số điều kiện của ma trận

Một số tính chất số điều kiện của ma trận

- ① Với mọi ma trận A : $cond(A) \geq 1$
- ② Với mọi ma trận đơn vị \mathbb{I} : $cond(\mathbb{I}) = 1$
- ③ Với mọi ma trận hoán vị P : $cond(P) = 1$
- ④ Với mọi ma trận A và số thực khác không α : $cond(\alpha A) = cond(A)$
- ⑤ Với mọi ma trận đường chéo $D = diag(d_i)$: $cond(D) = \frac{\max\{|d_i|\}}{\min\{|d_i|\}}$
- ⑥ Số điều kiện có ý nghĩa quan trọng trong việc đánh giá tính chính xác của lời giải của hệ phương trình tuyến tính.

Hệ xác định tồi và số điều kiện của ma trận

Matlab với số điều kiện

cond(A,p) để tính $\kappa_p(A)$ với $p = 1, 2, \text{inf}$.

cond(A) hoặc **cond(A,2)** tính $\kappa_2(A)$. Sử dụng **svd(A)**. Nên sử dụng với ma trận cỡ nhỏ.

cond(A,1) tính $\kappa_1(A)$. Sử dụng hàm **inv(A)**. Đòi hỏi thời gian tính toán ít hơn **cond(A,2)**.

cond(A,inf) tính $\kappa_\infty(A)$. Sử dụng hàm **inv(A)**. Đòi hỏi thời gian tính toán ít hơn **cond(A,1)**.

condest(A) để đánh giá $\kappa_1(A)$. Sử dụng hàm **lu(A)** và thuật toán Higham-Tisseur. Nên sử dụng cho ma trận cỡ lớn.

rcond(A) để đánh giá $1/\kappa_1(A)$

Hệ xác định tồi và số điều kiện của ma trận

Đánh giá sai số khi biết số điều kiện của ma trận

Gọi x là lời giải chính xác của $Ax = b$, còn x^* là lời giải của hệ $Ax^* = b + \Delta b$ (chú ý ta chỉ coi b bị nhiễu cộng). Đặt $\Delta x = x^* - x$, ta có $b + \Delta b = Ax^* = A(x + \Delta x) = Ax + A\Delta x$ do $Ax = b$ thế vào suy ra $\Delta x = A^{-1}\Delta b$.

$$b = Ax \Rightarrow \|b\| \leq \|A\| \|x\| \quad (13)$$

$$\Delta x = A^{-1}\Delta b \Rightarrow \|\Delta x\| \leq \|A^{-1}\| \|\Delta b\| \quad (14)$$

Nhân hai bất đẳng thức (13) (14) và sử dụng định nghĩa $cond(A) = \|A\| \|A^{-1}\|$ ta có đánh giá

$$\frac{\|\Delta x\|}{\|x\|} \leq cond(A) \frac{\|\Delta b\|}{\|b\|}$$

Hệ xác định tồi và số điều kiện của ma trận

Đánh giá sai số khi biết số điều kiện của ma trận (tiếp)

tiếp tục...

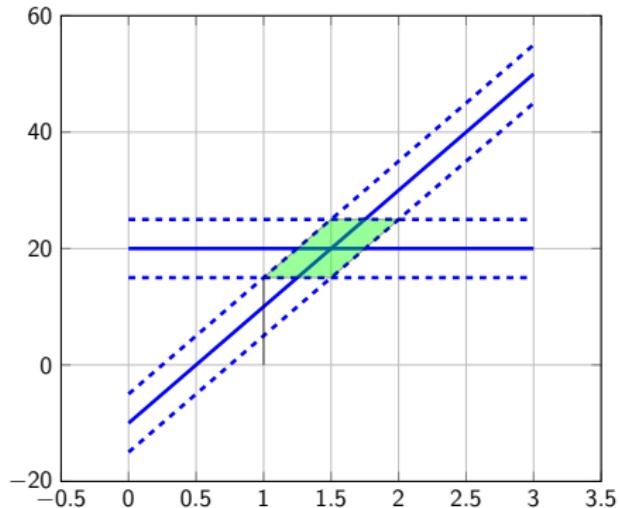
$$\frac{||\Delta x||}{||x||} \leq \text{cond}(A) \frac{||\Delta b||}{||b||}$$

Vậy số điều kiện cho phép ta xác định khả năng biến đổi sai số tương đối trong lời giải $\frac{||\Delta x||}{||x||}$ khi biết sự thay đổi tương đối trong vế phải $\frac{||\Delta b||}{||b||}$

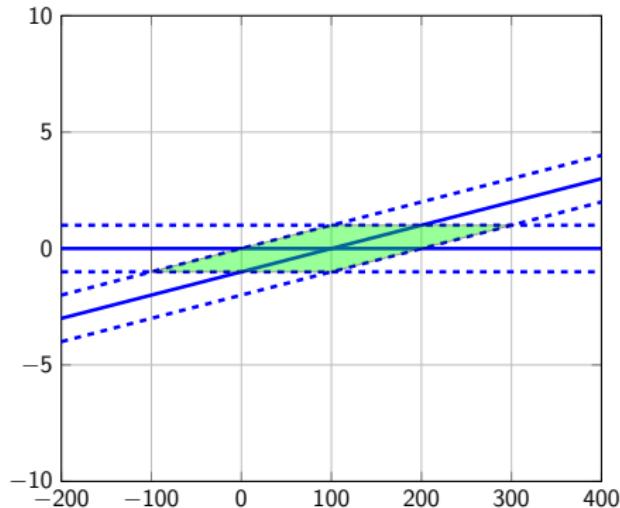
- Khi $\text{cond}(A)$ lớn hay hệ gần suy biến thì sự biến đổi tương đối của vế phải sẽ 'ép' sự thay đổi sai số tương ứng trong lời giải.
- Ngược lại, khi $\text{cond}(A)$ tiến về 1 hay hệ có điều kiện tốt thì sự biến đổi tương đương của vế phải và lời giải là như nhau.

Hệ xác định tồi và số điều kiện của ma trận

Hệ điều kiện tốt



Hệ điều kiện tồi



Hệ xác định tồi và số điều kiện của ma trận

Đánh giá sai số khi biết số điều kiện của ma trận (Kết luận)

Nếu dữ liệu vào được biểu diễn gần đúng với độ chính xác máy tính thì đánh giá sai số tương đối của lời giải tính được sẽ cho bởi công thức:

$$\frac{\|x^* - x\|}{\|x\|} \approx \text{cond}(A)\epsilon_M$$

lời giải tính được sẽ mất đi một quãng $\log_{10}(\text{cond}(A))$ chữ số thập phân trong sai số tương đối so với độ chính xác của dữ liệu.

Kết luận

Hệ phương trình tuyến tính $Ax = b$ là có điều kiện tồi nếu $\text{cond}(A)$ là lớn, khi đó sự thay đổi không lớn của dữ liệu có thể dẫn đến sự thay đổi lớn của lời giải.

Hệ xác định tồi và số điều kiện của ma trận

Ví dụ 15 :

Xét hệ phương trình

$$0.789x_1 + 0.563x_2 = 0.127$$

$$0.913x_1 + 0.659x_2 = 0.254$$

Kết quả khi dùng Matlab

```
» A=[0.789 0.563;0.913 0.659];  
» fprintf('cond(A)=%d ; det(A)=%d ',cond(A),det(A))  
» cond(A) = 2.193219e+006 ; det(A)=1.000000e-006
```

Hệ xác định tồi và số điều kiện của ma trận

Ví dụ 16 :

Xét hệ phương trình

$$\begin{pmatrix} 4.1 & 2.8 \\ 9.7 & 6.6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 4.1 \\ 9.7 \end{pmatrix}$$

Đây là hệ có điều kiện tồi do $\text{cond}(A, 1) = 2494.4$ đồng thời nghiệm chính xác của hệ là $x = (1, 0)^T$. Nếu ta thay về phải $b + \Delta b = (4.11, 9.70)^T$ thì nghiệm của hệ sẽ là $x^* = (0.34, 0.97)^T$.

Trong Matlab ta có

```
» A = [4.1 2.8; 9.7 6.6]; b = [4.1 ; 9.7]; b1=[4.11 ; 9.7];
```

```
» x = (A \ b)', x1 = (A \ b1)'
```

```
x = 1 0
```

```
x1 = 0.3400 0.9700
```

Hệ xác định tồi và số điều kiện của ma trận

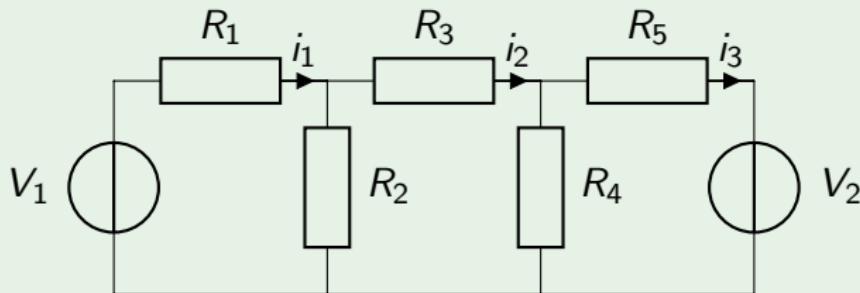


- 1 Thê nào là hệ phương trình tuyến tính ?
- 2 Ví dụ 3 chiều
- 3 Ma trận hoán vị và ma trận tam giác
- 4 Phân tích LU
- 5 Vai trò của phần tử trù
- 6 Hiệu ứng của sai số làm tròn
- 7 Hệ xác định tồi và số điều kiện của ma trận
 - Chuẩn ma trận
 - Số điều kiện ma trận
 - Đánh giá sai số khi biết số điều kiện của ma trận
- 8 Giải hệ phương trình tuyến tính bằng phân tích ma trận
- 9 Tổng kết

Giải hệ phương trình tuyến tính bằng phân tích ma trận

Phân tích mạng điện

Giải hệ phương trình tuyến tính có ứng dụng quan trọng trong phân tích mạng điện. Ví dụ cho mạng điện sau



Giải hệ phương trình tuyến tính bằng phân tích ma trận (tiếp)

Phân tích mạng điện (tiếp)

Theo định luật Kirchoff, hiệu điện thế các mạch vòng phải bằng không. Ta có hệ phương trình tuyến tính như sau

$$-V_1 + R_1 i_1 + R_2(i_1 - i_2) = 0$$

$$R_2(i_2 - i_1) + R_3 i_2 + R_4(i_2 - i_3) = 0$$

$$R_4(i_3 - i_2) + R_5 i_3 + V_2 = 0$$

chuyển thành

$$\begin{pmatrix} R_1 + R_2 & -R_2 & 0 \\ -R_2 & R_2 + R_3 + R_4 & -R_4 \\ 0 & -R_4 & R_4 + R_5 \end{pmatrix} \begin{pmatrix} i_1 \\ i_2 \\ i_3 \end{pmatrix} = \begin{pmatrix} V_1 \\ 0 \\ -V_2 \end{pmatrix}$$

Hệ phương trình tuyến tính

Tổng kết

- Đặc điểm hệ phương trình tuyến tính (đại số tuyến tính)
- Nhắc lại về ma trận hoán vị và ma trận tam giác
- Phân tích LU dùng để giải hệ phương trình tuyến tính
- Vai trò phần tử trụ trong phân tích LU có thể tạo sai số trong kết quả
- Hiệu ứng làm tròn sai số
- Xác định điều kiện của ma trận trong phương trình tuyến tính gây sai số kết quả
- Một ví dụ áp dụng hệ phương trình tuyến tính

Giải hệ phương trình tuyến tính bằng phân tích ma trận

Bài đọc thêm về nhà

Chúng ta có thể sử dụng nhiều phương pháp ngoài phân tích LU để giải hệ phương trình

- Phân tích Cholesky

- ▶ Khái niệm ma trận bán xác định dương
- ▶ Nếu A là ma trận xác định dương thì tồn tại ma trận tam giác dưới dương L sao cho $A = LL^T$
- ▶ Khử xuôi $Ly = b$, thế ngược $L^T x = y$

- Phân rã QR

- ▶ Khái niệm ma trận trực giao
- ▶ Phân rã QR : nếu $A \in \mathbb{R}^{m \times n}$ với $m \geq n$ và có hạng n thì tồn tại ma trận trực giao $Q \in \mathbb{R}^{m \times n}$ và ma trận tam giác trên $R \in \mathbb{R}^{n \times n}$ với các phần tử dương trên đường chéo sao cho $A = QR$
- ▶ Giải bài toán bình phương tối thiểu

$$\min\{||Ax - b||^2 | x \in \mathbb{R}\}$$

vậy nghiệm x là điểm dừng khi cực tiểu hóa bài toán trên.



25
SOICT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

**Thank you for
your attentions!**





ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

CHƯƠNG 3: ĐƯỜNG CONG KHỚP

Vũ Văn Thiệu, Đinh Việt Sang, Nguyễn Khánh Phương

TÍNH TOÁN KHOA HỌC

Giới thiệu

① Đặt vấn đề

② Nội suy

- Nội suy Lagrange
- Nội suy spline

③ Hồi qui

- Hồi qui tuyến tính
- Đường cong khớp bậc cao
- Đường cong khớp tổng quát

Đặt vấn đề

Vấn đề đặt ra là :

Giả sử cần tính xấp xỉ hàm $f(x)$ không biết rõ công thức, nhưng bằng thực nghiệm ta xác định được giá trị $f(x)$ tại $n + 1$ điểm rời rạc

$$f_i = f(x_i) \quad i = 0, 1, \dots, N$$

Cho bộ dữ liệu $\{(x_i, f_i), i = 0, 1, \dots, N\}$, cần tìm cách xấp xỉ hàm $f(x)$ bởi một hàm (có thể được tính theo công thức giải tích) $p(x)$.

Đặt vấn đề (tiếp)

tiếp tục...

Dễ giải quyết vấn đề

Ta sẽ tìm hiểu hai phương pháp dùng để để xây dựng đường cong khớp (curve fitting)

- Nội suy (Interpolation) : hàm $p(x)$ phải đi qua tất cả các điểm thuộc bộ dữ liệu.
- Hồi quy (Regression): cho trước dạng $p(x)$ có tham số, ta phải xác định các tham số để cực tiểu hóa một tiêu chí sai số nào đó (thường dùng tiêu chí bình phương bé nhất (least squares)).

Mở đầu

Phép nội suy có thể thực hiện với các hàm $p(x)$ là

- Hàm đa thức
- Hàm hữu tỷ
- Hàm chuỗi Fourier

Nội suy

Nội suy

Định nghĩa : Ta nói hàm $p(x)$ là hàm nội suy bộ dữ liệu $\{(x_i, f_i), i = 0, 1, \dots, N\}$ nếu thỏa mãn các điều kiện sau

$$p_i = f_i, i = 0, 1, \dots, N$$

Hệ $N + 1$ phương trình này gọi là **điều kiện nội suy**. Sở dĩ hàm đa thức được chọn vì việc tính giá trị, đạo hàm, vi phân dễ dàng (tính chất giải tích của hàm rõ ràng).

Đa thức nội suy bộ dữ liệu được gọi là đa thức nội suy (interpolating polynomial)

Định nghĩa : Ta gọi đa thức bậc không quá K là hàm

$$p_K(x) = a_0 + a_1x + \cdots + a_Kx^K$$

trong đó a_0, a_1, \dots, a_K là các hằng số.

Nội suy

Nội suy tuyến tính

Ta có hai điểm dữ liệu (x_0, f_0) và (x_1, f_1) cần nội suy bởi đa thức $p_1(x) \equiv a_0 + a_1x$ thỏa mãn hệ phương trình

$$p_1(x_0) \equiv a_0 + a_1x_0 = f_0$$

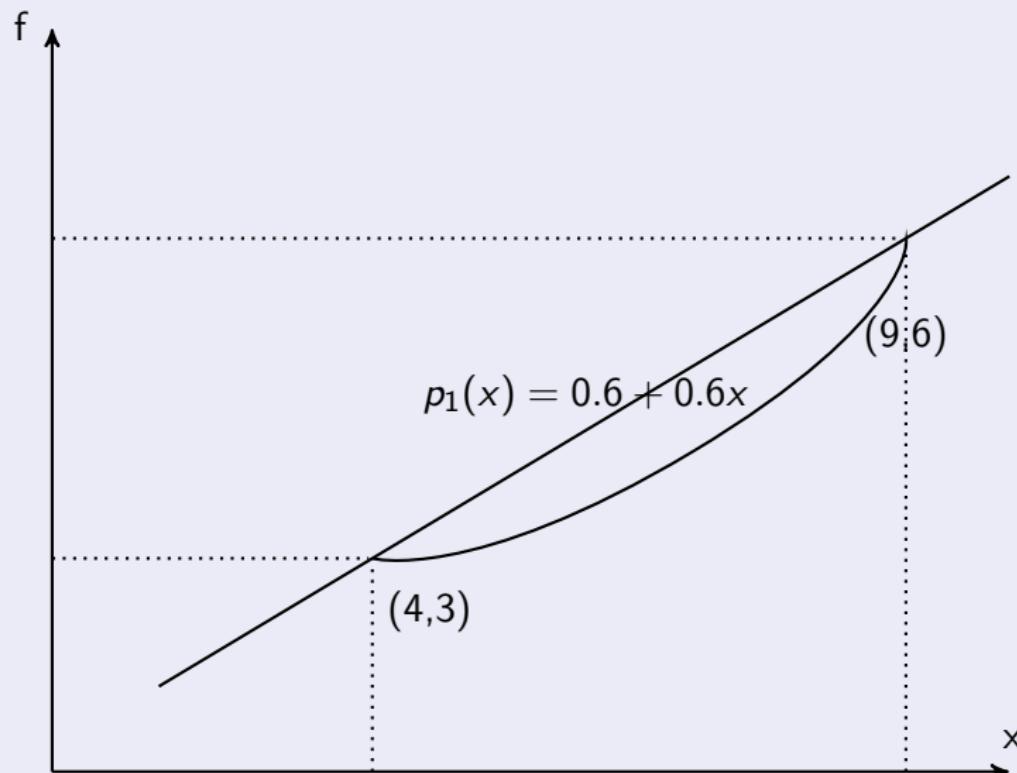
$$p_1(x_1) \equiv a_0 + a_1x_1 = f_1$$

Giải ra, ta có

- Khi $x_0 \neq x_1$ xảy ra hai trường hợp
 - ▶ nếu $f_0 \neq f_1$ thì $p_1(x) = \frac{x_1f_0 - x_0f_1}{x_1 - x_0} + \frac{f_1 - f_0}{x_1 - x_0}x$
 - ▶ nếu $f_0 = f_1$ thì lời giải là hằng số \Rightarrow đa thức bậc không.

Nội suy

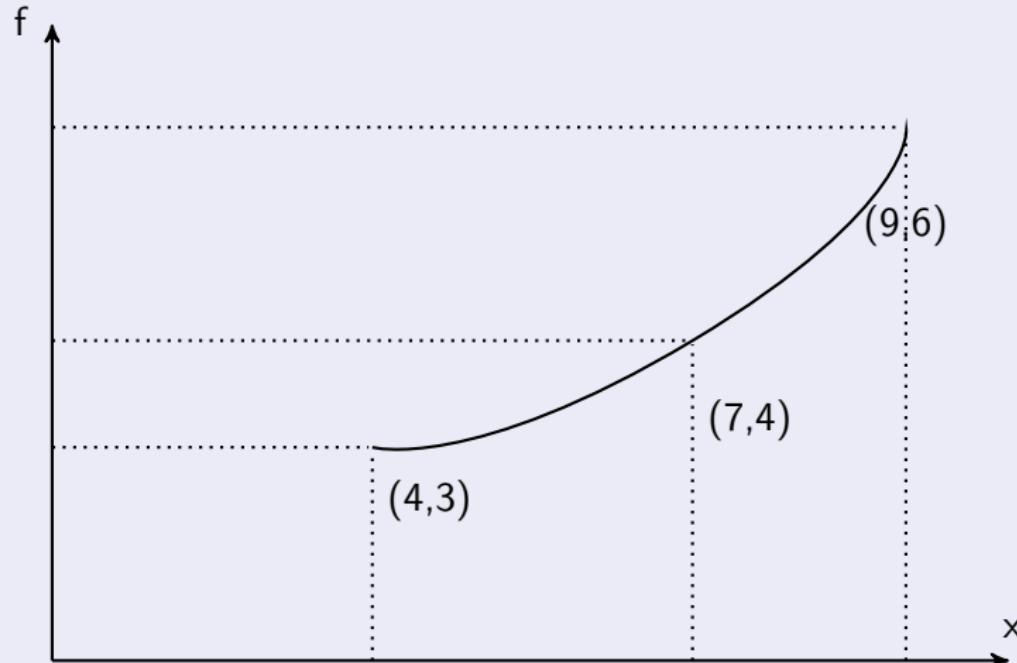
Nội suy tuyến tính (tiếp)



Nội suy

Công thức nội suy Lagrange

Cần xây dựng đường cong đi qua 3,4 hay nhiều điểm (xem hình).



Nội suy

Công thức nội suy Lagrange (tiếp)

Xét đa thức

$$p_M(x) = a_0 + a_1x + a_2x^2 + \cdots + a_Mx^M$$

Vậy đa thức $p_M(x)$ nội suy bộ dữ liệu $\{(x_i, f_i), i = 0, 1, \dots, N\}$ nếu các điều kiện nội suy :

$$p_M(x_0) \equiv a_0 + a_1x_0 + a_2x_0^2 + \cdots + a_Mx_0^M = f_0$$

$$p_M(x_1) \equiv a_0 + a_1x_1 + a_2x_1^2 + \cdots + a_Mx_1^M = f_1$$

...

$$p_M(x_N) \equiv a_N + a_1x_N + a_2x_N^2 + \cdots + a_Mx_N^M = f_N$$

Đây là một hệ phương trình tuyến tính nên ta có thể biểu diễn bằng công thức dạng ma trận.

Nội suy

Công thức nội suy Lagrange (tiếp)

Như vậy dạng ma trận của điều kiện nội suy

$$\begin{pmatrix} 1 & x_0 & \cdots & x_0^M \\ 1 & x_1 & \cdots & x_1^M \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_N & \cdots & x_N^M \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_M \end{pmatrix} = \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_N \end{pmatrix}$$

Giả thiết là x_0, x_1, \dots, x_N là phân biệt, vậy theo lý thuyết giải hệ phương trình tuyến tính trong chương trước ta có

- $N=M$: hệ phương trình có nghiệm duy nhất
- $M < N$: có thể chọn bộ dữ liệu để nó vô nghiệm
- $M > N$: nếu hệ có nghiệm thì nó có vô số nghiệm

Nội suy

Công thức nội suy Lagrange (tiếp)

Khi $N = M$ ma trận hệ số của hệ phương trình trên là ma trận Vandermonde (Không phải Voldemort)

$$V_N = \begin{pmatrix} 1 & x_0 & \cdots & x_0^M \\ 1 & x_1 & \cdots & x_1^M \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_N & \cdots & x_N^M \end{pmatrix}$$

Có định thức của V_n là :

$$V_N = \prod_{i>j} (x_i - x_j)$$

suy ra hệ có nghiệm khi x_0, x_1, \dots, x_N là phân biệt từng đôi.

Nội suy

Công thức nội suy Lagrange (tiếp)

Định lý (Tính duy nhất của đa thức nội suy) : Nếu các nút dữ liệu x_0, x_1, \dots, x_N là khác nhau từng đôi thì tồn tại duy nhất đa thức nội suy $p_N(x)$ bậc không quá N nội suy bộ dữ liệu $\{(x_i, f_i), i = 0, 1, \dots, N\}$.

Bậc của đa thức nội suy có thể nhỏ hơn N

Ví dụ, khi 3 điểm dữ liệu $(x_0, f_0), (x_1, f_1), (x_3, f_3)$ thẳng hàng thì phương trình nội suy trở thành bậc không quá hai do nó là đường thẳng.

Nội suy

Ví dụ 1 :

Xác định các hệ số của đa thức nội suy $p_2(x) = a_0 + a_1x + a_2x^2$ nội suy bộ dữ liệu $\{(-1,0),(0,1),(1,3)\}$, ta có điều kiện nội suy

$$a_0 + (-1)a_1 + 1a_2 = 0$$

$$a_0 + 0a_1 + 0a_2 = 1$$

$$a_0 + 1a_1 + 1a_2 = 3$$

Giải hệ phương trình ta có $a_0 = 1$, $a_1 = 1.5$ và $a_2 = 0.5$ vậy đa thức nội suy là

$$p_2(x) = 1 + 1.5x + 0.5x^2$$

Công thức nội suy Lagrange (tiếp)

Việc dùng giải trực tiếp hệ phương trình điều kiện nội suy dùng phép khử Gauss chẳng hạn cần đòi hỏi nhiều tính toán $\mathcal{O}(N^3)$. Thêm nữa, vì lũy thừa của hệ số x có thể rất cao nên độ sai số lại càng lớn do hiệu ứng việc làm tròn số, thậm chí còn tràn số (nhớ lại là số dấu phẩy động IEEE 754/85 độ chính xác kép chỉ là $\approx \pm 10^{38}$).

⇒ ta có thể dùng **công thức nội suy Lagrange** để tính hằng số của đa thức nội suy mà không cần giải hệ phương trình tuyến tính trên.

Nội suy

Công thức nội suy Lagrange (tiếp)

Xét bộ dữ liệu $\{(x_i, f_i), i = 0, 1, \dots, N\}$. Dạng Lagrange của đa thức nội suy đối với bộ dữ liệu là như sau :

$$p_N(x) = f_0 V_0(x) + f_1 V_1(x) + \cdots + f_N V_N(x)$$

trong đó $V_i(x) : i = 0, 1, \dots, N$ là các đa thức bậc N thỏa mãn điều kiện :

$$V_i(x_j) = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

Họ các đa thức $V_i(x) : i = 0, 1, \dots, N$ được gọi là họ các đa thức cơ sở.
Đễ dàng kiểm tra được rằng :

$$p_N(x_i) = f_i \quad i = 0, 1, \dots, N$$

Nội suy

Công thức nội suy Lagrange (tiếp)

Một trong lớp đa thức cơ sở có thể xây dựng theo công thức như sau

$$V_i(x) = \frac{(x - x_0)(x - x_1) \cdots (x - x_{i-1})(x - x_{i+1}) \cdots (x - x_n)}{(x_i - x_0)(x_i - x_1) \cdots (x_i - x_{i-1})(x_i - x_{i+1}) \cdots (x_i - x_n)}$$

rõ ràng thỏa mãn

$$V_i(x_j) = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

Vậy ta có thể tóm tắt lại công thức trên như sau

$$V_i(x) = \frac{\prod_{i \neq j} (x - x_j)}{\prod_{i \neq j} (x_i - x_j)}$$

Nội suy

Công thức nội suy Lagrange (tiếp)

Vậy công thức nội suy Lagrange

$$p_N(x) = \frac{(x - x_1)(x - x_2) \cdots (x - x_N)}{(x_0 - x_1) \cdots (x_0 - x_N)} f_0$$

...

$$+ \frac{(x - x_0)(x - x_1) \cdots (x - x_{i-1})(x - x_{i+1}) \cdots (x - x_n)}{(x_i - x_0)(x_i - x_1) \cdots (x_i - x_{i-1})(x_i - x_{i+1}) \cdots (x_i - x_n)} f_i \quad (1)$$

...

$$+ \frac{(x - x_0)(x - x_1) \cdots (x - x_{N-1})}{(x_N - x_0) \cdots (x_N - x_{N-1})} f_N$$

Công thức (1) tuy dài nhưng dễ nhớ và tính toán hơn.

Nội suy

Cài đặt $V_i(x)$ bằng Matlab

`v = polyinterp(x,y,u)` tính $v(j) = P(u(j))$ trong đó P là đa thức bậc $d = \text{length}(x) - 1$ thỏa mãn $P(x(i)) = y(i)$. Sử dụng công thức nội suy Lagrange. Hai vectơ x,y chứa hoàng độ và tung độ của bộ dữ liệu.

```
function v = polyinterp(x,y,u)
n = length(x);
v = zeros(size(u));
for k = 1:n
    w = ones(size(u));
    for j = [1:k-1 k+1:n]
        w = (u-x(j))./(x(k)-x(j)).*w;
    end
    v = v + w*y(k);
end
```

Nội suy

Cài đặt $V_i(x)$ bằng Matlab (tiếp)

```
function v = polyinterp(x,y,u)
n = length(x);
v = zeros(size(u));
for k = 1:n
    w = ones(size(u));
    for j = [1:k-1 k+1:n]
        w = (u-x(j))./(x(k)-x(j)).*w;
    end
    v = v + w*y(k);
end
```

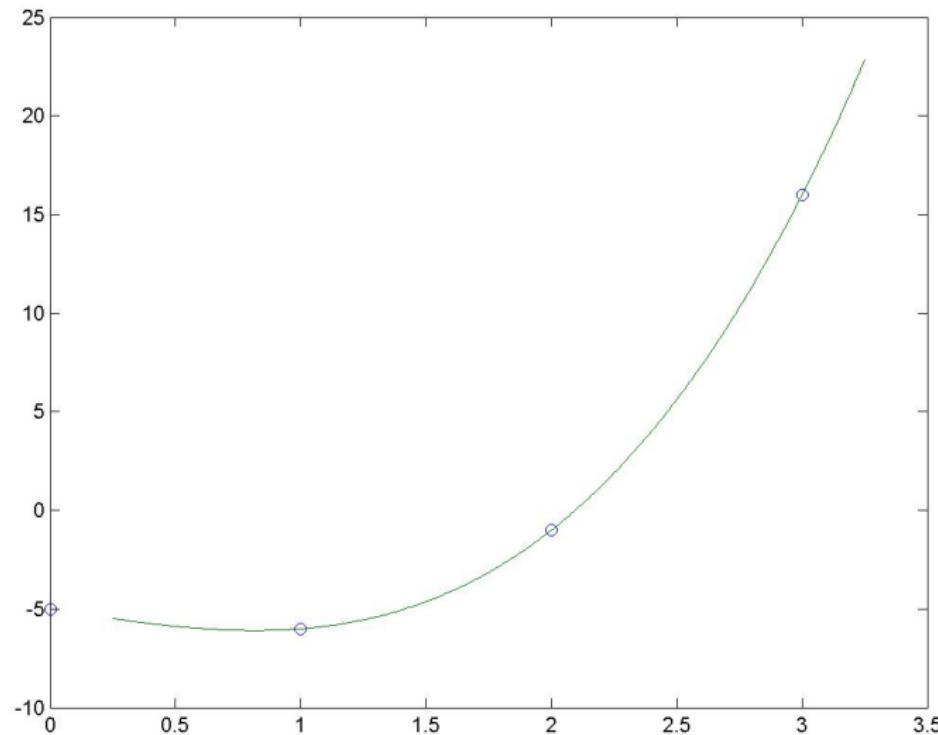
- Hai vectơ x, y chứa hoàng độ và tung độ của bộ dữ liệu.
- u là vec tơ chứa các hoàng độ của các điểm cần tính theo công thức nội suy Lagrange.
- v là vectơ chứa giá trị hàm nội suy tại các điểm cho trong u

Nội suy

Ví dụ 2 :

```
x = [0 1 2 3];  
y = [-5 -6 -1 16];  
u= [0.25:0.01:3.25];  
v = polyinterp(x,y,u);  
plot(x,y,'o',u,v,'-')
```

Nội suy



Nội suy

Sai số nội suy

Định nghĩa : Sai số của đa thức nội suy $g(x)$ được xác định bởi công thức :

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

Độ lớn của $e(x)$ phụ thuộc vào

- Hoàng độ của các điểm dữ liệu.
- Kích thước của miền nội suy $D = x_n - x_0$.
- Độ lớn của đa thức nội suy

Đối với đa thức nội suy Lagrange, sai số được xác định bởi công thức :

$$e(x) = f(x) - g(x) = L(x)f^{(N+1)}(\xi)$$

trong đó $f^{(N+1)}$ là đạo hàm bậc $N + 1$ của hàm $f(x)$ còn

$$L(x) = \frac{(x-x_0)(x-x_1)\cdots(x-x_n)}{(N+1)!}$$

Nội suy

Sai số nội suy (tiếp)

tiếp tục ... công thức :

$$e(x) = f(x) - g(x) = L(x)f^{(N+1)}(\xi)$$

ξ phụ thuộc x nhưng thỏa mãn điều kiện $x_0 < \xi < x_N$. Cũng theo công thức trên nếu $f(x)$ là đa thức bậc không quá N thì đạo hàm $f^{(N+1)}$ sẽ triệt tiêu, nghĩa là sai số bằng không. Do $L(x) = \frac{(x-x_0)(x-x_1)\cdots(x-x_n)}{(N+1)!}$ nên ta có các kết luận sau

- Với một lưới cách đều, biên độ dao động của $L(x)$ nhỏ nhất tại tâm và tăng dần về biên của miền nội suy.
- Kích thước nội suy tăng lên, độ lớn của dao động tăng nhanh chóng.

Nội suy Lagrange



Nội suy spline

Đặt vấn đề

Trong trường hợp này ta xét phép nội suy bởi một tập các các đa thức bậc thấp thay vì một đa thức bậc cao duy nhất

- Nội suy bởi spline tuyến tính
- Nội suy bởi spline bậc ba

Nhớ lại công thức đánh giá sai số nội suy Lagrange

$$f(x) - p_N(x) = [w_{N+1}(x)/(N+1)!]f^{(N+1)}(\xi_x)$$

trong đó $w_{N+1}(x) = (x - x_0)(x - x_1) \cdots (x - x_N)$ và ξ_x là một giá trị trong khoảng (x_0, x_N) .

Nội suy spline

Đặt vấn đề (tiếp)

tiếp tục ... Giả sử mọi điểm dữ liệu $x_i \in [a, b]$ thì ta có

$$\max_{x \in [a, b]} |f(x) - p_N(x)| \leq |b - a|^{N+1} \times \frac{\max_{x \in [a, b]} |f^{(N+1)}(x)|}{N!}$$

Dánh giá này cho biết nếu muốn giảm sai số mà vẫn giữ nguyên số lượng điểm dữ liệu N thì cần tìm cách giảm kích thước của $|b - a|$.

Hướng tiếp cận của ta là xấp xỉ đa thức từng khúc (piecewise polynominal approximation): Như vậy đoạn $[a, b]$ sẽ được chia thành nhiều đoạn nhỏ không giao nhau và **các đa thức khác nhau sẽ được xấp xỉ trên từng đoạn con.**

Nội suy spline

Spline tuyến tính từng khúc

Đối với bộ dữ liệu

$$\{(x_i, f_i) : i = 0, 1, \dots, N\}$$

trong đó

$$a = x_0 < x_1 < \dots < x_N = b, h \equiv \max_i |x_i - x_{i-1}|,$$

spline tuyến tính $S_{1,N}(x)$ là hàm liên tục nội suy dữ liệu đã cho và được xây dựng từ các hàm tuyến tính được xác định bởi các đa thức nội suy hai điểm dữ liệu :

Nội suy spline

Spline tuyến tính từng khúc (tiếp)

tiếp tục...

$$S_{1,N}(x) = \begin{cases} \frac{f_1 - f_0}{x_1 - x_0}(x - x_1) + f_1, & \text{nếu } x \in [x_0, x_1] \\ \vdots \\ \frac{f_i - f_{i-1}}{x_i - x_{i-1}}(x - x_i) + f_i, & \text{nếu } x \in [x_{i-1}, x_i] \\ \vdots \\ \frac{f_N - f_{N-1}}{x_N - x_{N-1}}(x - x_N) + f_N, & \text{nếu } x \in [x_{N-1}, x_N] \end{cases}$$

Dễ thấy

$$L_i(x) = \frac{f_i - f_{i-1}}{x_i - x_{i-1}}(x - x_i) + f_i$$

là phương trình đường thẳng qua hai điểm (x_i, f_i) và (x_{i-1}, f_{i-1})

Nội suy spline

Spline tuyến tính từng khúc (tiếp)

Sử dụng công thức sai số cho đa thức nội suy với $x \in [a, b]$ cho từng đoạn $[x_{i-1}, x_i]$ ta được

$$\begin{aligned} \max_{z \in [x_{i-1}, x_i]} |f(z) - S_{1,N}(z)| &\leq \frac{|x_i - x_{i-1}|^2}{8} \times \max_{x \in [x_{i-1}, x_i]} |f^{(2)}(x)| \\ &\leq \frac{h^2}{8} \times \max_{x \in [x_{i-1}, x_i]} |f^{(2)}(x)| \end{aligned}$$

trong đó $h \equiv \max_i |x_i - x_{i-1}|$.

Nội suy spline

Ví dụ 3 :

Cho bộ dữ liệu $\{(-1,0),(0,1),(1,3)\}$, ta xây dựng spline tuyết tính như sau

$$S_{1,2}(x) = \begin{cases} \frac{f_1-f_0}{x_1-x_0}(x - x_1) + f_1 = \frac{f_1-f_0}{x_1-x_0}(x - x_0) + f_0, & \text{nếu } x \in [x_0, x_1] \\ \frac{f_2-f_1}{x_2-x_1}(x - x_2) + f_2 = \frac{f_2-f_1}{x_2-x_1}(x - x_1) + f_1, & \text{nếu } x \in [x_1, x_2] \end{cases}$$

suy ra với bộ dữ liệu tương ứng

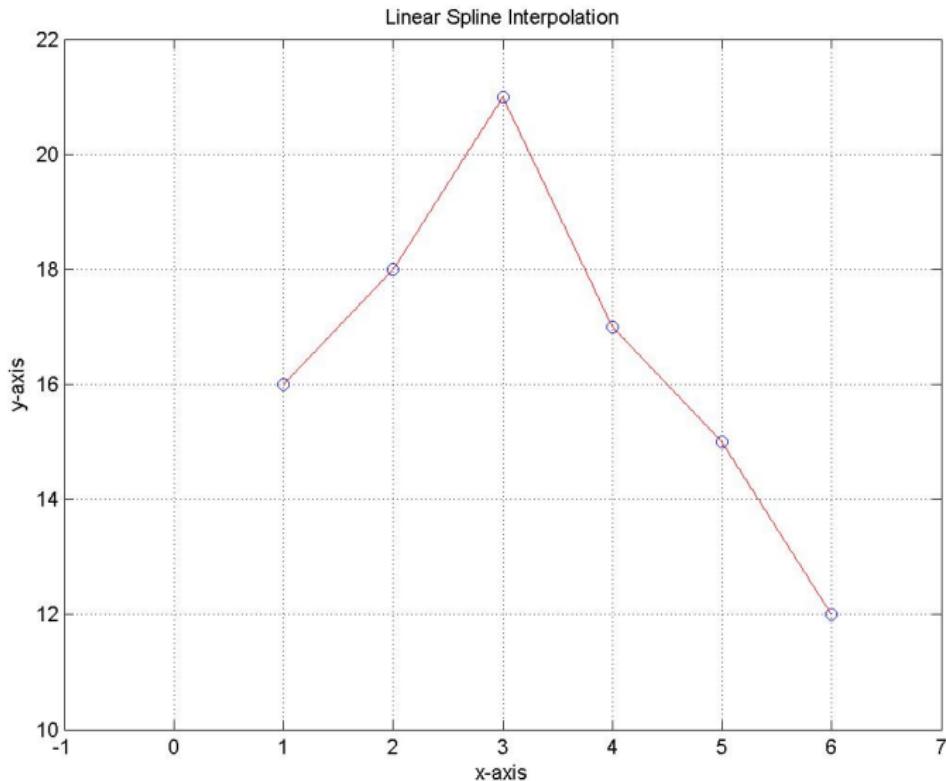
$$S_{1,2}(x) = \begin{cases} \frac{1-0}{0-(-1)}x + 1 = x + 1, & \text{nếu } x \in [-1, 0] \\ \frac{3-1}{1-0}(x - 1) + 3 = 2x + 1, & \text{nếu } x \in [0, 1] \end{cases}$$

Nội suy spline

Ví dụ 4 :

```
» clear;
» x=1:6;
» y=[16 18 21 17 15 12];
» plot(x,y,'o')
» hold on; grid on;
» axis([-1 7 10 22])
» xx = [1:0.01:6];
» yy = piecelin(x,y,xx);
» plot(xx,yy,'r')
» hold off
» xlabel('x-axis');ylabel('y-axis');
» title('Linear Spline Interpolation');
```

Nội suy spline



Nội suy spline

Spline bậc ba

Thay vì các đoạn thẳng, Spline bậc 3 sử dụng các đa thức bậc 3 để xấp xỉ đa thức từng khúc. Ta cũng có bộ dữ liệu

$$\{(x_i, f_i) : i = 0, 1, \dots, N\}$$

trong đó

$$a = x_0 < x_1 < \cdots < x_N = b, h \equiv \max_i |x_i - x_{i-1}|,$$

Nội suy Lagrange



Nội suy spline

Spline bậc ba (tiếp)

Định nghĩa : Spline bậc ba $S_{3,N}(x)$ nội suy bộ dữ liệu đã cho

- $S_{3,N}(x)$ là hàm bậc ba giữa hai nút $[x_{i-1}, x_i]$

$$S_{3,N}(x) = \begin{cases} p_1(x) = a_{1,0} + a_{1,1}x + a_{1,2}x^2 + a_{1,3}x^3, & \text{nếu } x \in [x_0, x_1] \\ \vdots \\ p_i(x) = a_{i,0} + a_{i,1}x + a_{i,2}x^2 + a_{i,3}x^3, & \text{nếu } x \in [x_{i-1}, x_i] \\ \vdots \\ p_N(x) = a_{N,0} + a_{N,1}x + a_{N,2}x^2 + a_{N,3}x^3, & \text{nếu } x \in [x_{N-1}, x_N] \end{cases}$$

- $S_{3,N}(x)$ là hàm liên tục có đạo hàm cấp một và hai liên tục trên đoạn $[x_0, x_N]$ (kể cả tại các nút)

Nội suy spline

Spline bậc ba (tiếp)

Với $x_{i-1} < x < x_i$, $S_{3,N}(x)$ có giá trị
 $p_i(x) = a_{i,0} + a_{i,1}x + a_{i,2}x^2 + a_{i,3}x^3$

Với $x_i < x < x_{i+1}$, $S_{3,N}(x)$ có giá trị
 $p_{i+1}(x) = a_{i+1,0} + \dots + a_{i+1,3}x^3$

Giá trị của $S_{3,N}(x)$ khi $x \rightarrow x_i^-$

$$p_i(x_i)$$

$$p'_i(x_i)$$

$$p''_i(x_i)$$

Giá trị của $S_{3,N}(x)$ khi $x \rightarrow x_i^+$

$$p_{i+1}(x_i)$$

$$p'_{i+1}(x_i)$$

$$p''_{i+1}(x_i)$$

Nhận xét :

- Điều kiện về tính trơn tại điểm x_i thì

$$p'_i(x_i) = p'_{i+1}(x_i); \quad p''_i(x_i) = p''_{i+1}(x_i); \quad i = 1, 2, \dots, N-1$$

- Điều kiện nội suy dữ liệu đảm bảo

$$p_i(x_i) = p_{i+1}(x_i) = f_i; \quad i = 0, 1, 2, \dots, N-1$$

Nội suy spline

Spline bậc ba (tiếp)

- Điều kiện biên tự nhiên (Natural Boundary Condition)

$$p_1''(x_0) = 0; p_N''(x_N) = 0;$$

- Điều kiện đạo hàm cấp hai (Second Derivative Condition)

$$p_1''(x_0) = f''(x_0); p_N''(x_N) = f''(x_N);$$

- Điều kiện sát biên (Not-a-knot Condition)

$$p_1'''(x_1) = p_2'''(x_1); p_{N-1}'''(x_{N-1}) = p_N'''(x_{N-1});$$

Nội suy spline

Hàm tính Spline nội suy bậc 3 trong MatLab:

p=spline(x,y): trả lại dạng đa thức từng khúc của spline bậc 3 đối với bộ dữ liệu (x,y).

v=ppval(p,x): Tính giá trị của đa thức nội suy tại đầu vào x (p xác định nhờ hàm spline).

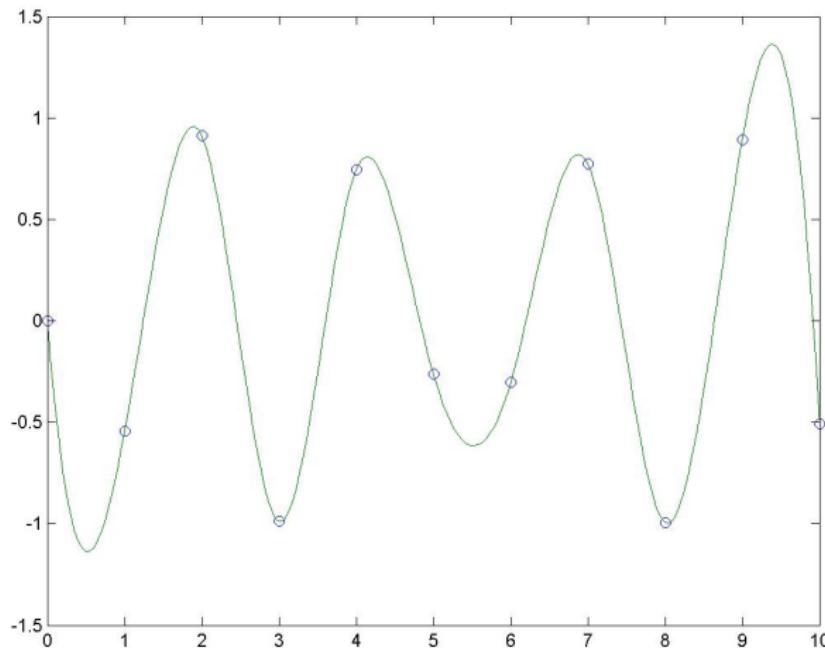
yy=spline(x,y,xx): Xác định spline nội suy bậc 3 đối với bộ dữ liệu (x,y) và trả lại yy là vectơ giá trị của spline nội suy tại các điểm trong vectơ xx.

Nội suy spline

Ví dụ 4 :

```
» x = 0:10; y = sin(10*x);  
» p = spline(x,y);  
» xx = linspace(0,10,200);  
» yy = ppval(p,xx);  
» plot(x,y,'o',xx,yy)
```

Nội suy spline cấp 3



Nội suy spline

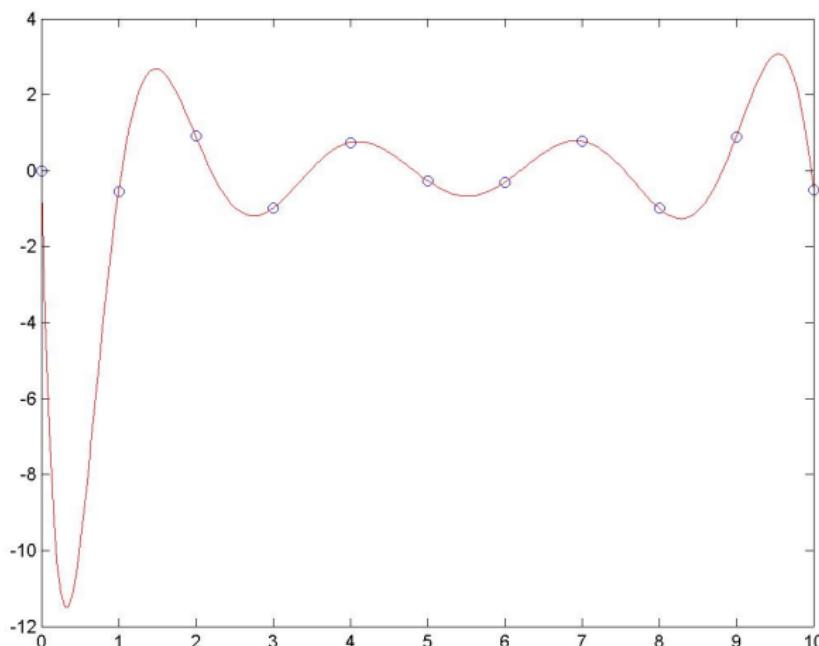
Ví dụ 5 :

```
» x = 0:10; y = sin(10*x);  
» xx = linspace(0,10,200);  
» coef = polyfit(x,y,10);
```

Warning: Polynomial is badly conditioned. Remove repeated data points or try centering and scaling as described in HELP POLYFIT.
(Type "warning off MATLAB:polyfit:RepeatedPointsOrRescale" to suppress this warning.)

```
» yy = polyval(coef,xx);  
» plot(x,y,'o',xx,yy,'r')
```

Nội suy spline cấp 3



Như vậy khi ta nội suy bằng spline bậc 3 thì tốt hơn nội suy đa thức với
hàm $\sin(10x)$

Nội suy spline

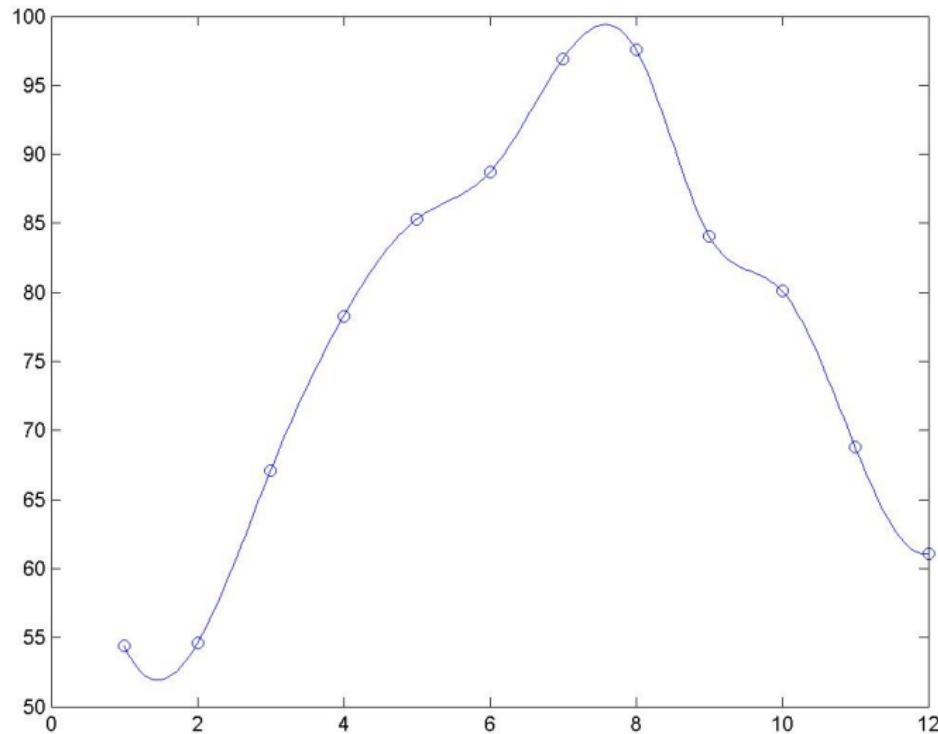
Ví dụ 6 :

Công suất tiêu thụ điện của một thiết bị trong 12 giờ như sau

1	2	3	4	5	6
54.4	54.6	67.1	78.3	85.3	88.7
7	8	9	10	11	12
96.9	97.6	84.1	80.1	68.8	61.1

Dùng spline bậc ba để nội suy tập điểm dữ liệu trên, ta có được

Nội suy spline cấp 3



Nội suy spline cấp 3



Đặt vấn đề

Cho bộ dữ liệu $\{(x_i, f_i), i = 1, \dots, N\}$, cần tìm cách xấp xỉ hàm $f(x)$ bởi một hàm $p(x)$. Ta cần quan tâm đo sai số tổng cộng trên toàn đoạn.

- Sai số cực đại $E_\infty(f) = \max_{1 \leq k \leq n} |f(x_k) - y_k|$
- Sai số trung bình $E_1(f) = \frac{1}{n} \sum_{k=1}^n |f(x_k) - y_k|$
- Căn bậc hai của tổng các bình phương sai số

$$E_2(f) = \sqrt{\frac{1}{n} \sum_{k=1}^n (f(x_k) - y_k)^2}$$

Hồi qui

Hồi qui tuyến tính

Hồi qui tuyến tính là việc tìm một đường thẳng khớp với các điểm dữ liệu theo nghĩa bình phương tối thiểu.

- Cho tập N cặp điểm dữ liệu $\{(x_i, f_i) : i = 1, \dots, N\}$
- Tìm hệ số m và hằng số tự do b của đường thẳng

$$y(x) = mx + b$$

sao cho đường thẳng này khớp với dữ liệu theo tiêu chí bình phương tối thiểu.

$$L(m, b) = \sum_{i=1}^N (f_i - (mx_i + b))^2$$

Hồi qui

Hồi qui tuyến tính (tiếp)

Nghĩa là ta cần tìm m và b để đạt cực tiểu hàm

$$L(m, b) = \sum_{i=1}^N (f_i - (mx_i + b))^2$$

để tìm cực tiểu này, ta cần giải hệ phương trình tìm điểm dừng

$$\frac{\partial L}{\partial m} = \sum_{i=1}^N 2(f_i - (mx_i + b))(-x_i) = 0$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^N 2(f_i - (mx_i + b))(-1) = 0$$

Hỏi qui

Hỏi qui tuyến tính (tiếp)

$$\left[\sum_{i=1}^N x_i^2 \right] m + \left[\sum_{i=1}^N x_i \right] b = \sum_{i=1}^N x_i f_i$$

$$\left[\sum_{i=1}^N x_i \right] m + \left[\sum_{i=1}^N 1 \right] b = \sum_{i=1}^N f_i$$

hay dưới dạng ma trận,

$$\begin{pmatrix} \sum_{i=1}^N x_i^2 & \sum_{i=1}^N x_i \\ \sum_{i=1}^N x_i & N \end{pmatrix} \begin{pmatrix} m \\ b \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^N x_i f_i \\ \sum_{i=1}^N f_i \end{pmatrix}$$

Giải hệ phương trình ta có được hệ số m và b .

Hỏi qui

Ví dụ 6 :

Xây dựng đường cong khớp cho bộ dữ liệu sau

i	1	2	3	4
x_i	1	3	4	5
f_i	2	4	3	1

Vậy hệ phương trình cần giải là

$$13m + 4b = 10$$

$$51m + 13b = 31$$

từ đó tìm được $b = 107/35$ và $m = -6/35$, đường thẳng cần tìm là
 $y = (-6/35)x + 107/35$

Hồi qui

Đường cong khớp bậc cao

Xây dựng đường cong khớp

$$p_M(x) = a_0 + a_1x + a_2x^2 + \cdots + a_Mx^M$$

khớp với bộ dữ liệu $\{(x_i, f_i) | i = 1, \dots, N\}$ theo tiêu chí bình phương tối thiểu

$$\sigma_M \equiv (p_M(x_1) - f_1)^2 + (p_M(x_2) - f_2)^2 + \cdots + (p_M(x_N) - f_N)^2$$

Giải hệ phương trình tìm điểm dừng

$$\frac{\partial}{\partial a_0} \sigma_M = 0; \quad \frac{\partial}{\partial a_1} \sigma_M = 0; \cdots \frac{\partial}{\partial a_M} \sigma_M = 0;$$

Đường cong khớp bậc cao (tiếp)

Viết dưới dạng ma trận hệ phương trình chuẩn

$$\begin{pmatrix} \sum_{i=1}^N 1 & \sum_{i=1}^N x_i & \cdots & \sum_{i=1}^N x_i^M \\ \sum_{i=1}^N x_i & \sum_{i=1}^N x_i^2 & \cdots & \sum_{i=1}^N x_i^{M+1} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^N x_i^M & \sum_{i=1}^N x_i^{M+1} & \cdots & \sum_{i=1}^N x_i^{2M} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_M \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^N f_i \\ \sum_{i=1}^N f_i x_i \\ \vdots \\ \sum_{i=1}^N f_i x_i^M \end{pmatrix}$$

Điểm đặc biệt của ma trận trên là đối xứng xác định dương nên ta có thể dùng phép khử Gauss mà không cần thay đổi dòng, để giải hệ phương trình chuẩn.

Hỏi qui

Ví dụ 7 :

Xác định các tham số a, b, c của đường cong

$$y = ax^2 + bx + c$$

khớp với bộ dữ liệu sau

x	1	2	3	4	5	6	7	8
y	1	8	27	64	125	216	350	560

Hồi qui

Ví dụ 7 : (tiếp)

`x = [1 2 3 4 5 6 7 8];`

`y = [1 8 27 64 125 216 350 560];`

`s0 = length(x); s1 = sum(x); s2 = sum(x.^2);`

`s3 = sum(x.^3); s4 = sum(x.^4);`

`A = [s4 s3 s2; s3 s2 s1; s2 s1 s0];`

`b = [sum(x.^2.*y); sum(x.*y);sum(y)];`

`c0 = A\b;`

`c = polyfit(x,y,2); % Tìm đa thức hồi qui khớp dữ liệu`

`c0 % Đưa ra hệ số tính được`

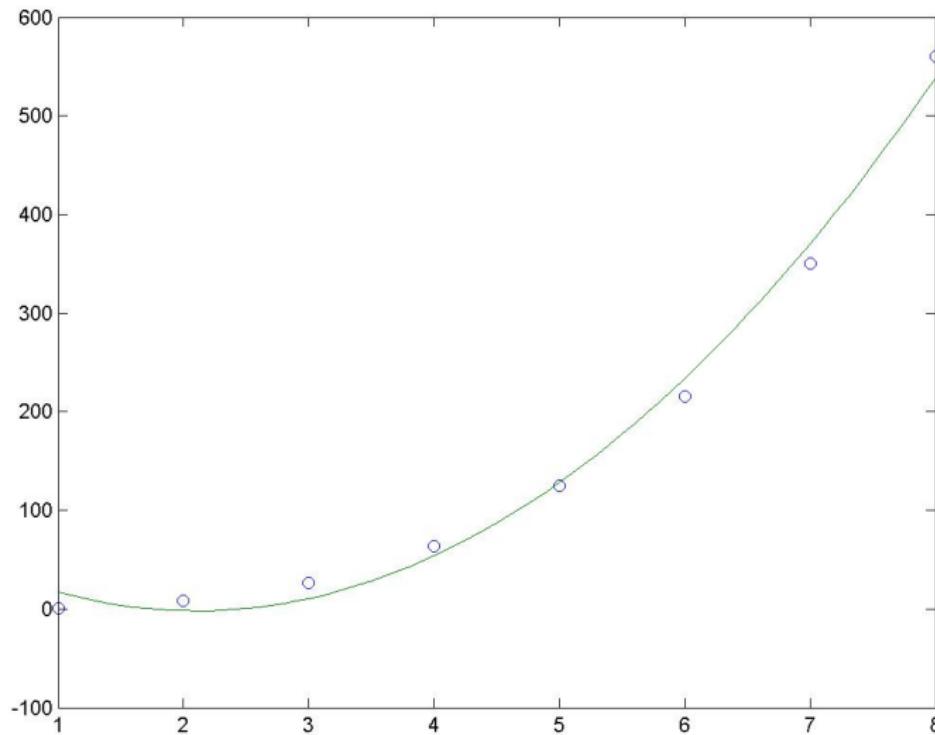
`c % Đưa ra hệ số tìm được bởi polyfit`

`xx = 1:0.1:8;`

`yy = polyval(c0,xx);`

`plot(x,y,'o',xx,yy)`

Hỏi qui



Hồi qui

Đường cong khớp tổng quát

Giả sử ta muốn khớp N điểm đa cho $\{(x_i, f_i) : i = 1, \dots, N\}$ bởi họ gồm m hàm độc lập tuyến tính $\varphi_j(x), j = 1, 2, \dots, m$ tức là hàm $f(x)$ cần tìm có dạng

$$f(x) = \sum_{j=1}^m c_j \varphi_j(x)$$

Chẳng hạn các họ hàm sau

- $\varphi_j(x) = x^j, j = 1, 2, \dots, m$
- $\varphi_j(x) = \sin(jx), j = 1, 2, \dots, m$
- $\varphi_j(x) = \cos(jx), j = 1, 2, \dots, m$

Hồi qui

Đường cong khớp tổng quát (tiếp)

Ta cần xác định các số c_1, c_2, \dots, c_m để cực tiểu hóa sai số căn bậc hai của tổng các bình phương sai số :

$$E(c_1, c_2, \dots, c_m) = \sum_{k=1}^n (f(x_k) - f_k)^2 = \sum_{k=1}^n \left[\left(\sum_{j=1}^m c_j \varphi_j(x) \right) - f_k \right]^2$$

Để tìm các hệ số c_1, c_2, \dots, c_m , ta cần giải hệ phương trình

$$\frac{\partial E}{\partial c_j} = 0, j = 1, 2, \dots, m$$

Hồi qui



Tổng kết

Tổng kết

- Đặt vấn đề đường cong khớp
- Định nghĩa nội suy - interpolation
 - ▶ Nội suy Lagrange
 - ▶ Nội suy Spline
- Định nghĩa hồi qui - regression
 - ▶ Hồi qui tuyến tính
 - ▶ Hồi qui đa thức bậc cao



25
SOLCT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

**Thank you for
your attentions!**





ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

CHƯƠNG 3: GIẢI PHƯƠNG TRÌNH PHI TUYẾN

Vũ Văn Thiệu, Đinh Việt Sang, Nguyễn Khánh Phương

TÍNH TOÁN KHOA HỌC

Giới thiệu

1 Đặt vấn đề

- Sự tồn tại và duy nhất của nghiệm
- Độ nhạy và điều kiện của bài toán giải phương trình phi tuyến
- Thủ tục lặp

2 Phương pháp chia đôi

3 Phương pháp dây cung

4 Phương pháp Newton

5 Phương pháp cát tuyến

6 Phương pháp lặp

7 Phương pháp Bairstow

8 Tổng kết

Giải phương trình phi tuyến

Bài toán

Cho hàm phi tuyến (non-linear) $f(x)$, ta cần tìm x thỏa mãn

$$f(x) = 0.$$

Lời giải x là nghiệm của phương trình và cũng gọi là nghiệm (không điểm) của hàm $f(x)$.

Bài toán tìm x , được gọi là bài toán tìm nghiệm (root finding).

Giải phương trình phi tuyến

Các ví dụ về bài toán tìm nghiệm của các phương trình phi tuyến

① $1 + 4x - 16x^2 + 3x^3 - 3x^4 = 0$

② $\frac{x\sqrt{(2.1-0.5x)}}{(1-x)\sqrt{(1.1-0.5x)}} - 369 = 0$ với $(0 < x < 1)$

③ $\operatorname{tg}(x) - \tanh(x) = 0$ trong đó $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

Giải phương trình phi tuyến

Đặt vấn đề

Nếu các phương trình $f(x)$ là phi tuyến thì

- thường không có lời giải dưới dạng công thức tường minh
- các phương pháp số cho phép ta tìm nghiệm dựa trên *thủ tục lặp*

Giải phương trình phi tuyến

Khoảng phân ly nghiệm

Đối với hàm $f : \mathbb{R} \mapsto \mathbb{R}$ đoạn $[a, b]$ được gọi là **khoảng phân ly nghiệm** nếu hàm f có giá trị trái dấu ở hai đầu mút a, b , tức là $f(a)f(b) < 0$.

Sự tồn tại nghiệm trong khoảng phân ly nghiệm

Nếu f là hàm liên tục trên đoạn $[a, b]$ và $f(a)f(b) < 0$ thì tồn tại $x^* \in [a, b]$ sao cho $f(x^*) = 0$.

Giải phương trình phi tuyến

Các ví dụ về số nghiệm của các phương trình phi tuyến

- ① $e^x + 1 = 0$ không có nghiệm
- ② $e^{-x} - x = 0$ có một nghiệm
- ③ $x^2 - 4\sin(x) = 0$ có hai nghiệm
- ④ $x^3 - 6x^2 + 11x - 6 = 0$ có ba nghiệm
- ⑤ $\cos(x) = 0$ có vô số nghiệm

Giải phương trình phi tuyến

Điều kiện của bài toán giải phương trình

- Giá trị tuyệt đối của số điều kiện bài toán tìm nghiệm x^* của hàm $f : \mathbb{R} \mapsto \mathbb{R}$ là $\frac{1}{|f'(x^*)|}$.
- Nghiệm x^* đc gọi là có *điều kiện tồi* (tốt) nếu đường tiếp tuyến nó với đồ thị tại điểm có hoành độ x^* gần như nằm ngang (thẳng đứng).

Giải phương trình phi tuyến

Thủ tục lặp

Các phương trình phi tuyến thường không có lời giải tường minh. Vì vậy để tìm nghiệm của chúng ta thường phải dùng phương pháp số dựa trên thủ tục lặp.

- **Điều kiện dừng:** $|f(x_k)| < \epsilon$ hay $|x^* - x_k| < \epsilon$ với ϵ là *độ chính xác* cho trước còn x_k là lời giải xấp xỉ thu được ở bước k
- **Tốc độ hội tụ:** Ta ký hiệu *sai số* bước lặp k là: $e_k = x_k - x^*$. Dãy $\{e_k\}$ được gọi là *hội tụ* với *cấp độ* r nếu

$$\lim_{k \rightarrow \infty} \frac{|e_{k+1}|}{|e_k|^r} = C$$

trong đó C là hằng số khác không

Tính toán khoa học

└ Đặt vấn đề

└ Thủ tục lặp

└ Giải phương trình phi tuyến

Thủ tục lặp

Các phương trình phi tuyến thường không có lời giải tường minh. Vì vậy để tìm nghiệm của chúng ta thường phải dùng phương pháp số đưa trên thủ tục lặp

- ♦ Điều kiện dừng: $|f(x_k)| < \epsilon$ hay $|x^* - x_k| < \epsilon$ với ϵ là độ chính xác cho trước còn x_k là lời giải xấp xỉ thu được ở bước k

- ♦ Tốc độ hội tụ: Ta ký hiệu sau số bước lặp k là: $e_k = x_k - x^*$. Dãy $\{e_k\}$ được gọi là hội tụ với cấp độ r nếu

$$\lim_{k \rightarrow \infty} \frac{|e_{k+1}|}{|e_k|^r} = C$$

trong đó C là hằng số khác không

Tên gọi của tốc độ hội tụ trong một số trường hợp

- $r = 1$ - tốc độ hội tụ tuyến tính
- $r > 1$ - tốc độ hội tụ trên tuyến tính
- $r = 2$ - tốc độ hội tụ bình phương

Câu hỏi



1 Đặt vấn đề

- Sự tồn tại và duy nhất của nghiệm
- Độ nhạy và điều kiện của bài toán giải phương trình phi tuyến
- Thủ tục lặp

2 Phương pháp chia đôi

3 Phương pháp dây cung

4 Phương pháp Newton

5 Phương pháp cát tuyến

6 Phương pháp lặp

7 Phương pháp Bairstow

8 Tổng kết

Phương pháp chia đôi

Thủ tục lặp

Giả thiết khoảng phân li nghiệm $[a, c]$ chỉ có một nghiệm

- ① Thu nhỏ dần khoảng phân li nghiệm thông qua phép chia nhỏ
- ② Phép chia đc thực hiện là phép chia đôi $b = \frac{(a+c)}{2}$
Nếu $f(b) = 0$ thì b là nghiệm đúng cần tìm, ngược lại nếu $f(b) \neq 0$ thì ta có
 - ▶ $f(a)f(b) < 0$ thì khoảng phân li nghiệm mới là $[a, b]$
 - ▶ Ngược lại khoảng phân li nghiệm mới là $[b, c]$

Hai bước 1-2 được lặp lại cho đến khi $|a - c| < \epsilon$ cho trước

Tính toán khoa học

└ Phương pháp chia đôi

└ Phương pháp chia đôi

Thủ tục lặp

Giả thiết khoảng phân lì nghiệm $[a, c]$ chỉ có một nghiệm

• Thu nhỏ dần khoảng phân lì nghiệm thông qua phép chia đôi

• Phép chia đôi thực hiện là phép chia đôi $b = \frac{a+c}{2}$ Nếu $f(b) = 0$ thì b là nghiệm đúng cần tìm, ngược lại nếu $f(b) \neq 0$ thì ta có $f(a)f(b) < 0$ thì khoảng phân lì nghiệm mới là $[a, b]$ Ngược lại khoảng phân lì nghiệm mới là $[b, c]$ Hai bước 1-2 được lặp lại cho đến khi $|a - c| < \epsilon$ cho trước

Nếu cho trước độ chính xác ϵ thì số bước lặp là số nguyên n thỏa mãn

$$n \geq \log_2 \frac{c - a}{\epsilon}$$

bởi vì

$$\frac{c - a}{2^n} < \epsilon$$

Phương pháp chia đôi

Ví dụ 1:

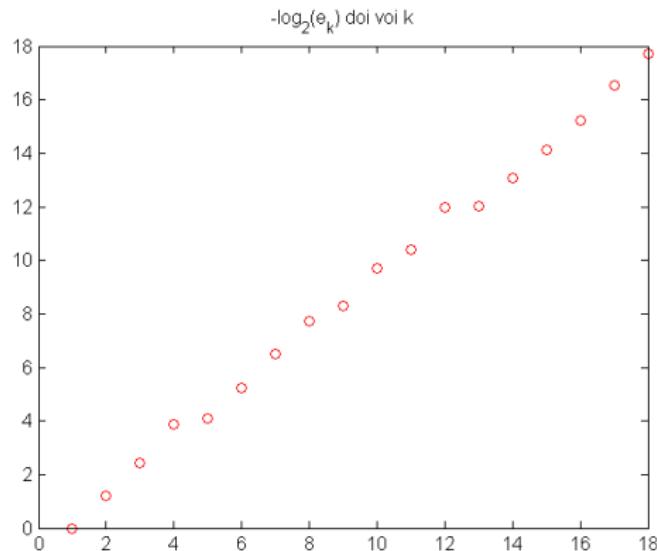
Tìm nghiệm của phương trình $e^x - 2 = 0$ có khoảng phân li nghiệm $[0, 2]$ với độ chính xác $\epsilon = 0.01$

Lần lặp	a	b	c	f(a)	f(b)	f(c)	sai số
1	0.0000	1.0000	2.0000	-1.0000	0.7183	5.0389	2.0000
2	0.0000	0.5000	1.0000	-1.0000	-0.3513	0.7183	1.0000
3	0.5000	0.7500	1.0000	-0.3513	0.1170	0.7183	0.5000
4	0.5000	0.6250	0.7500	-0.3513	-0.1318	0.1170	0.2500
5	0.6250	0.6875	0.7500	-0.1318	-0.0113	0.1170	0.1250
6	0.6875	0.7188	0.7500	-0.0113	0.0519	0.1170	0.0625
7	0.6875	0.7031	0.7188	-0.0113	0.0201	0.0519	0.0313
8	0.6875	0.6953	0.7031	-0.0113	0.0043	0.0201	0.0156
9	0.6875	0.6914	0.6953	-0.0113	-0.00349	0.0043	0.0078

Phương pháp chia đôi

Ví dụ 2

Xét lời giải của phương trình $f(x) = 1/(x - e^{-x})$ có khoảng phân li nghiệm $[0, 1]$ với độ chính xác $\epsilon = 0.00001$



Sai số:

$$e_k = \max\{x^* - a_k, c_k - x^*\}$$

Trục hoành: số lần lặp k

Trục tung: $-\log_2(e_k)$

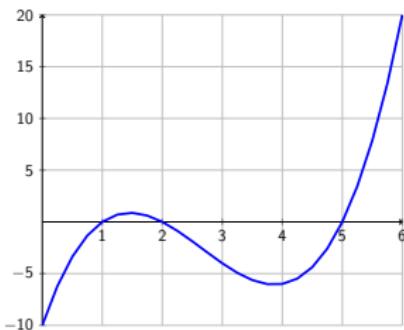
Rõ ràng $e_k \approx 2^{-k}$

Phương pháp chia đôi

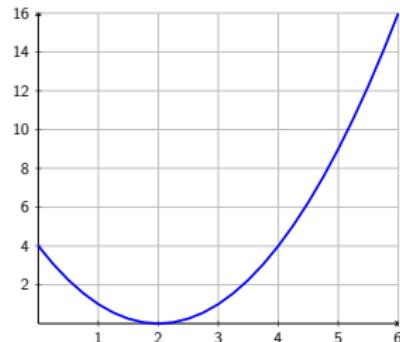
Nhận xét về phương pháp chia đôi

- **Điểm mạnh:** Làm việc ngay cả với hàm không có dưới dạng giải tích.
- **Điểm yếu:**
 - ▶ Cần xác định khoảng phân li nghiệm và chỉ tìm được một nghiệm.
 - ▶ Không tìm được nghiệm kép.
 - ▶ Khi hàm f có những điểm kỳ dị, phương pháp chia đôi có thể coi chúng là nghiệm.

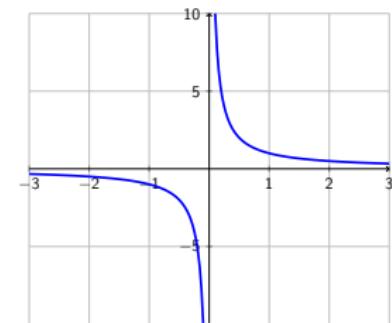
Phương pháp chia đôi



a) Nhiều nghiệm



b) Nghiệm kép



c) Điểm kỳ dị

Phương pháp chia đôi



1 Đặt vấn đề

- Sự tồn tại và duy nhất của nghiệm
- Độ nhạy và điều kiện của bài toán giải phương trình phi tuyến
- Thủ tục lặp

2 Phương pháp chia đôi

3 Phương pháp dây cung

4 Phương pháp Newton

5 Phương pháp cát tuyến

6 Phương pháp lặp

7 Phương pháp Bairstow

8 Tổng kết

Phương pháp dây cung

Thủ tục lắp

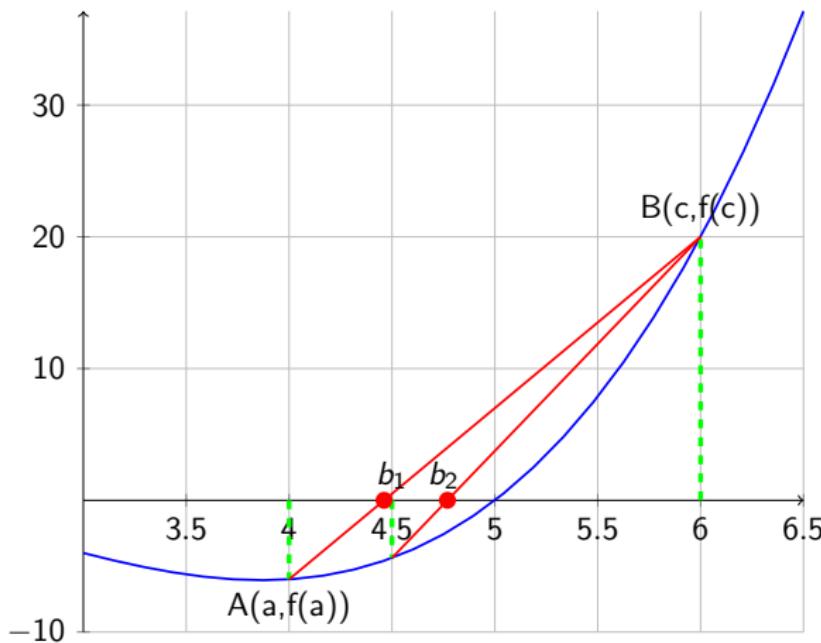
Giả thiết khoảng phân li nghiệm $[a, c]$ chỉ có một nghiệm

- ① Thu nhỏ dần khoảng phân li nghiệm thông qua phép chia nhỏ
- ② Phép chia đc thực hiện là $b = a - \frac{c-a}{f(c)-f(a)} f(a) = \frac{af(c)-cf(a)}{f(c)-f(a)}$
Nếu $f(b) = 0$ thì b là nghiệm cần tìm. Ngược lại nếu $f(b) \neq 0$, ta có:
 - ▶ Nếu $f(a)f(b) < 0$ thì khoảng phân li nghiệm mới là $[a, b]$
 - ▶ Ngược lại khoảng phân li nghiệm mới là $[b, c]$

Hai bước 1-2 được lặp lại cho đến khi $[a, c] < \epsilon$ cho trước.

Vậy b là giao trực hoành với đoạn thẳng nối $A(a,f(a))$ với $B(c,f(c))$

Phương pháp dây cung



Phương pháp dây cung

Nhận xét

- **Ưu điểm:** cũng như phương pháp chia đôi (bisection), ta không cần dạng giải tích của phương trình f
- **Nhược điểm:**
 - ▶ Cần biết khoảng phân li nghiệm
 - ▶ Hội tụ một phía nên chậm, đặc biệt chậm khi đoạn chứa nghiệm lớn
 - ▶ Có thể cải tiến bằng cách sử dụng cùng phương pháp chia đôi

Phương pháp dây cung



1 Đặt vấn đề

- Sự tồn tại và duy nhất của nghiệm
- Độ nhạy và điều kiện của bài toán giải phương trình phi tuyến
- Thủ tục lặp

2 Phương pháp chia đôi

3 Phương pháp dây cung

4 Phương pháp Newton

5 Phương pháp cát tuyến

6 Phương pháp lặp

7 Phương pháp Bairstow

8 Tổng kết

Phương pháp Newton

Mô tả

Ý tưởng cơ bản của phương pháp là thay phương trình $f(x) = 0$ phi tuyến bằng một phương trình gần đúng, tuyến tính đối với x . Được xây dựng dựa trên nền tảng khai triển Taylor.

Giả sử $f(x)$ khả vi liên tục đến bậc $n + 1$ thì tồn tại $\xi \in (a, b)$

$$\begin{aligned}f(b) &= f(a) + f'(a)(b-a) + \frac{(b-a)^2}{2!} f''(a) + \dots \\&\quad + \frac{(b-a)^n}{n!} f^{(n)}(a) + \frac{(b-a)^{(n+1)}}{(n+1)!} f^{(n+1)}(\xi)\end{aligned}$$

Phương pháp Newton

Mô tả (tiếp)

Khai triển Taylor của $f(x)$ tại lân cận nghiệm xấp xỉ ban đầu x_0 :

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + O(h^2)$$

trong đó $h = x - x_0$.

Giải phương trình xấp xỉ đối với x :

$$f(x_0) + f'(x_0)(x - x_0) = 0$$

$$\text{Thu được: } x = x_0 - \frac{f(x_0)}{f'(x_0)}$$

x là nghiệm không chính xác, nhưng lời giải này sẽ gần với nghiệm đúng hơn giá trị khởi tạo x_0 .

Phương pháp Newton

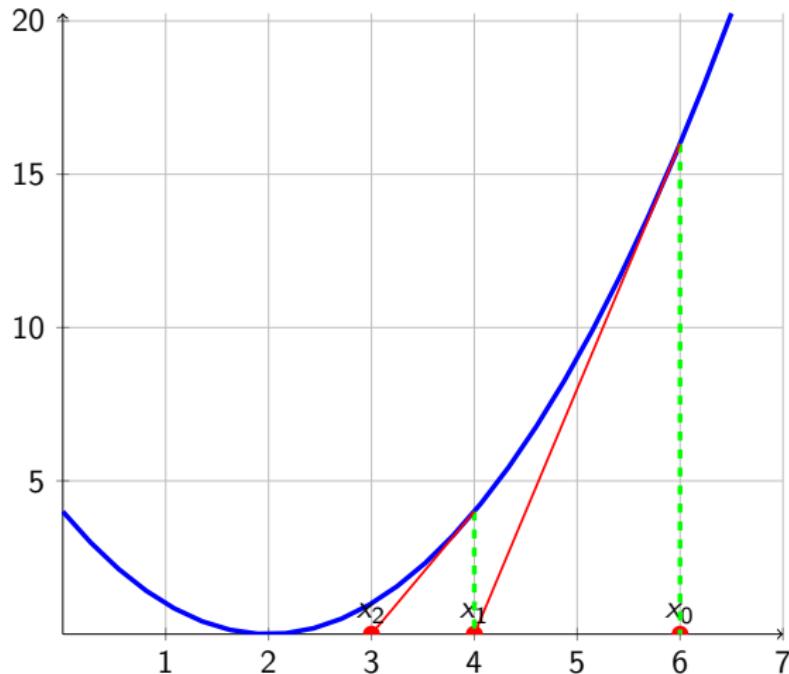
Thủ tục lặp

- ① Khởi tạo với x_0
- ② Tính với $k > 0$

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

- ③ Lặp lại bước 2 đến khi $|f(x_k)| < \epsilon$ với ϵ là độ chính xác cho trước

Phương pháp Newton



Phương pháp Newton

Nhận xét

- **Ưu điểm:**

- ▶ Đối với hàm đủ trơn và ta bắt đầu từ điểm gần nghiệm thì tốc độ hội tụ của phương pháp là bình phương hay $r = 2$
- ▶ Không cần biết khoảng phân ly nghiệm chỉ cần điểm ban đầu x_0

- **Nhược điểm:**

- ▶ Cần tính đạo hàm bậc một $f'(x_k)$, ta cũng có thể tính gần đúng bằng công thức $f'(x_k) = \frac{f(x_k+h)-f(x_k-h)}{2h}$ với h là giá trị rất nhỏ e.g. $h = 0.001$
- ▶ Không phải lúc nào thủ tục lặp cũng hội tụ

Phương pháp Newton

Ví dụ 1

Sử dụng phương pháp Newton để tìm nghiệm của phương trình

$$f(x) = x^2 - 4 \sin(x) = 0$$

Đạo hàm bậc một của $f(x)$

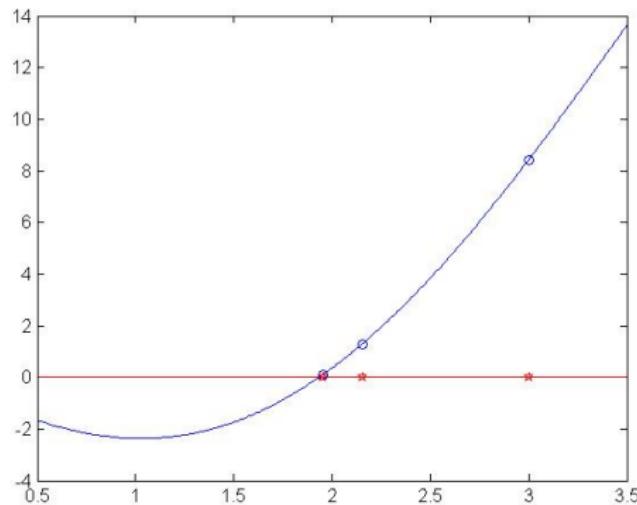
$$f'(x) = 2x - 4 \cos(x)$$

Công thức lặp của phương pháp Newton là

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} = x_k - \frac{x_k^2 - 4 \sin(x_k)}{2x_k - 4 \cos(x_k)}$$

Điểm sấp xỉ xuất phát $x_0 = 3$

Phương pháp Newton



k	x_k	$f(x_k)$
0	3.000000	8.435520
1	2.153058	1.294773
2	1.954039	0.108439

Phương pháp Newton

Ví dụ 2

Giải phương trình $f(x) = x^2 - 2 = 0$ vì $f'(x) = 2x$ nên công thức lặp sẽ là
 $x_{k+1} = x_k - \frac{x_k^2 - 2}{2x_k}$ sai số $e_k = x_k - x^* = x_k - \sqrt{2}$

k	x_k	e_k
0	4.000000000	2.5857864376
1	2.250000000	0.8357864376
2	1.569444444	0.1552308821
3	1.421890364	0.0076768014
4	1.414234286	0.0000207236
5	1.414213563	0.0000000002

Phương pháp Newton

Ví dụ 3

Giải phương trình

$$f(x) = \text{sign}(x - a)\sqrt{|x - a|}$$

Phương trình này thỏa mãn:

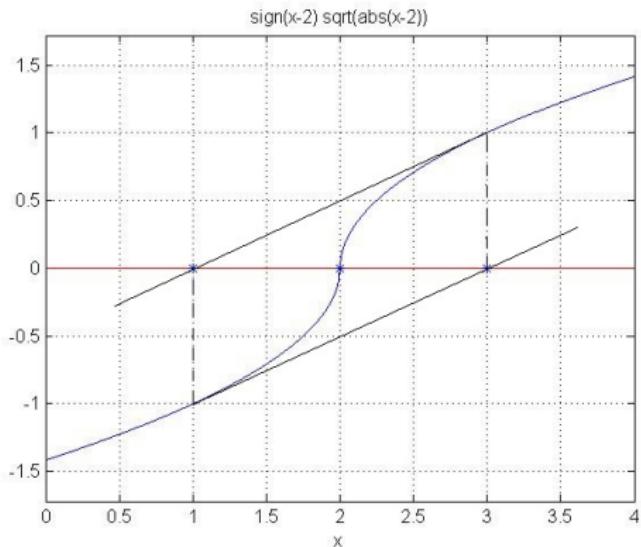
$$x - a - \frac{f(x)}{f'(x)} = -(x - a)$$

Không điểm của hàm là $x^* = a$.

Nếu ta vẽ tiếp tuyến đồ thị tại điểm bất kì thì nó luôn cắt trục hoành tại điểm đối xứng với đường thẳng $x = a$.

Phương pháp Newton lặp vô hạn, không hội tụ và cũng không phân kì.

Phương pháp Newton



Phương pháp Newton



1 Đặt vấn đề

- Sự tồn tại và duy nhất của nghiệm
- Độ nhạy và điều kiện của bài toán giải phương trình phi tuyến
- Thủ tục lặp

2 Phương pháp chia đôi

3 Phương pháp dây cung

4 Phương pháp Newton

5 Phương pháp cát tuyến

6 Phương pháp lặp

7 Phương pháp Bairstow

8 Tổng kết

Phương pháp cát tuyến

Thủ tục lặp

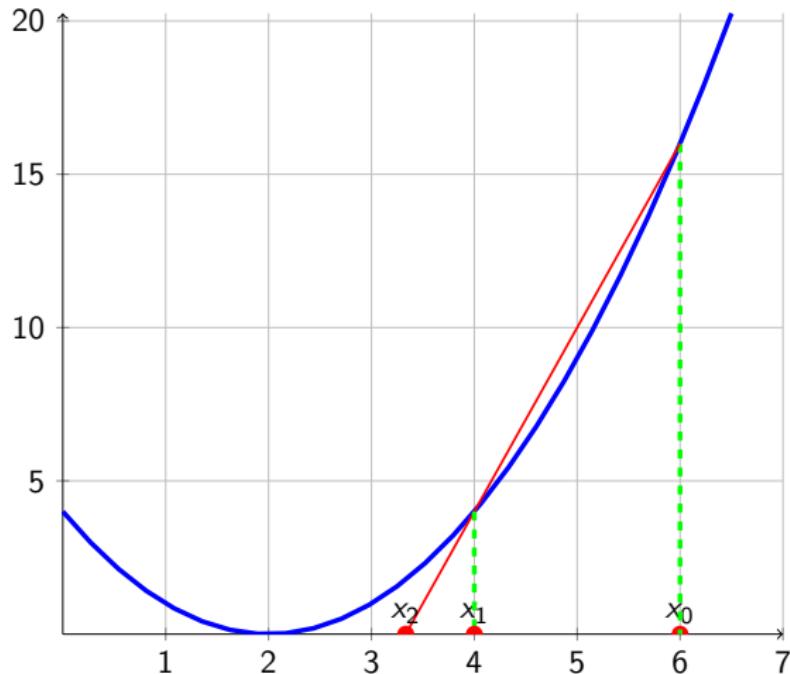
Cải tiến của phương pháp Newton, thay vì ta dùng tiếp tuyến $f'(x)$ thì ta dùng sai phân xấp xỉ dựa trên hai bước lặp liên tiếp.

- ① Bắt đầu với hai điểm xuất phát x_0 và x_1
- ② Với $k \geq 2$, ta lặp theo công thức

$$s_k = \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$$
$$x_{k+1} = x_k - \frac{f(x_k)}{s_k}$$

- ③ Lặp lại bước 2 đến khi $|f(x_k)| < \epsilon$ dương nhỏ cho trước.

Phương pháp cát tuyến



Phương pháp cát tuyến

Nhận xét

- Ưu điểm:
 - ▶ Không cần biết khoảng phân ly nghiệm chỉ cần hai điểm ban đầu x_0 và x_1
 - ▶ Không cần tính đạo hàm bậc một $f'(x_k)$
- Nhược điểm:
 - ▶ Cần có hai điểm khởi tạo
 - ▶ Tốc độ hội tụ của phương pháp trên tuyến tính $1 < r < 2$, cụ thể tỉ lệ vàng $r \approx \frac{1+\sqrt{5}}{2} = 1.618$

Phương pháp cát tuyến

Ví dụ 1

Giải lại phương trình

$$f(x) = \text{sign}(x - 2)\sqrt{|x - 2|} = 0$$

Với hai điểm xuất phát là $x_0 = 4$, $x_1 = 3$ với $\epsilon = 0.001$

$$\begin{aligned}s_k &= \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}} \\&= \frac{\text{sign}(x_k - 2)\sqrt{|x_k - 2|} - \text{sign}(x_{k-1} - 2)\sqrt{|x_{k-1} - 2|}}{x_k - x_{k-1}} \\x_{k+1} &= x_k - \frac{f(x_n)}{s_n} = x_k - \frac{\text{sign}(x_k - 2)\sqrt{|x_k - 2|}}{s_k}\end{aligned}$$

Phương pháp cát tuyến

k	x_k	e_k
0	4.000000000	2.000000000
1	3.000000000	1.000000000
2	0.585786438	1.4142135624
3	1.897220119	0.1027798813
:	:	:
26	1.999989913	0.0000100868
27	1.999998528	0.0000014716
28	2.000003853	0.0000038528
29	2.000000562	0.0000005621

Phương pháp cát tuyến

Ví dụ 2

Giải phương trình

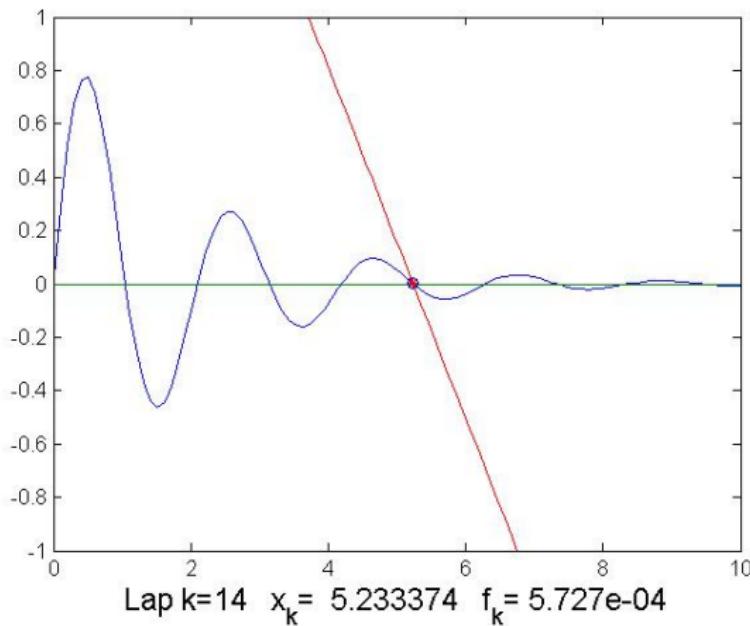
$$f(x) = e^{-x/2} \sin(3x) = 0$$

Với hai điểm xuất phát là x_0, x_1 với độ chính xác ϵ được nhập từ bàn phím

$$\begin{aligned}s_k &= \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}} \\&= \frac{e^{-x_k/2} \sin(3x_k) - e^{-x_{k-1}/2} \sin(3x_{k-1})}{x_k - x_{k-1}}\end{aligned}$$

$$x_{k+1} = x_k - \frac{f(x_n)}{s_n} = x_k - \frac{e^{-x_k/2} \sin(3x_k)}{s_k}$$

Phương pháp cát tuyến



Hai điểm xuất phát
 $x_0 = 4$
 $x_1 = 5$
Độ chính xác
 $\epsilon = 0.001$

Phương pháp cát tuyển



1 Đặt vấn đề

- Sự tồn tại và duy nhất của nghiệm
- Độ nhạy và điều kiện của bài toán giải phương trình phi tuyến
- Thủ tục lặp

2 Phương pháp chia đôi

3 Phương pháp dây cung

4 Phương pháp Newton

5 Phương pháp cát tuyến

6 Phương pháp lặp

7 Phương pháp Bairstow

8 Tổng kết

Phương pháp lắp

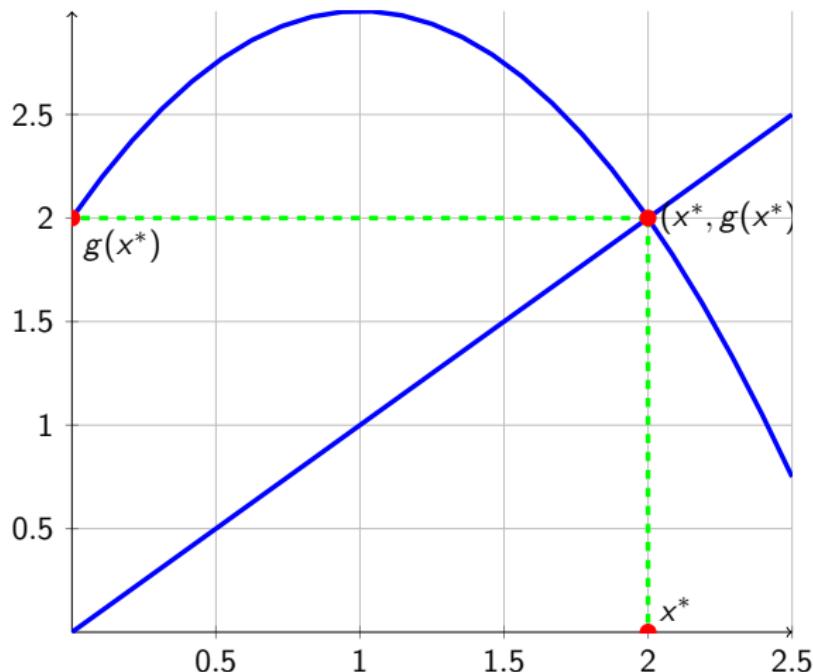
Điểm bất động

Thay vì viết phương trình dưới dạng $f(x) = 0$, ta viết lại dưới dạng bài toán

Tìm x thỏa mãn $x = g(x)$

Điểm x^* gọi là *điểm bất động* của hàm $g(x)$ nếu $x^* = g(x^*)$, nghĩa là điểm x^* không bị biến đổi bởi ánh xạ g

Phương pháp lắp



Phương pháp lặp

Các ví dụ

- Phương pháp Newton, theo công thức $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$, hàm g mà ta cần tìm điểm bất động x^* sẽ là $g(x) = x - f(x)/f'(x)$
- Tìm nghiệm $f(x) = x - e^{-x} \Rightarrow g(x) = e^{-x}$
- Tìm nghiệm $f(x) = x^2 - x - 2 \Rightarrow g(x) = \sqrt{x+2}$ hoặc $g(x) = x^2 - 2$
- Tìm nghiệm $f(x) = 2x^2 - x - 1 \Rightarrow g(x) = 2x^2 - 1$

Phương pháp lặp

Thủ tục lặp

Cách tiếp cận để giải bài toán

$$x_{k+1} = g(x_k) \text{ với } k = 1, 2, \dots$$

Thủ tục lặp trên thường được gọi là thủ tục lặp **tìm điểm bất động** với điểm xuất phát x_1 cho trước

Phương pháp lắp

Nhận xét

- **Ưu điểm:**

- ▶ Không cần biết khoảng phân li nghiệm

- **Nhược điểm:**

- ▶ Không phải lúc nào cũng hội tụ

Phương pháp lắp

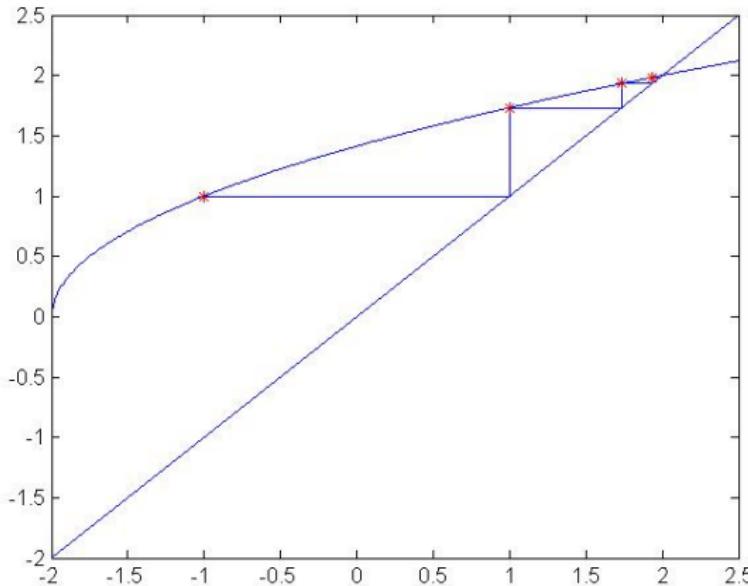
Ví dụ 1

Tìm nghiệm của phương trình $f(x) = x^2 - x - 2 = 0$ có thể dẫn về tìm điểm bất động

$$g(x) = \sqrt{x + 2}$$

Điểm xấp xỉ xuất phát $x_1 = -1$

Phương pháp lặp



Điểm xuất phát

$$x_1 = -1$$

Số bước lặp

$$n = 3$$

Phương pháp lắp

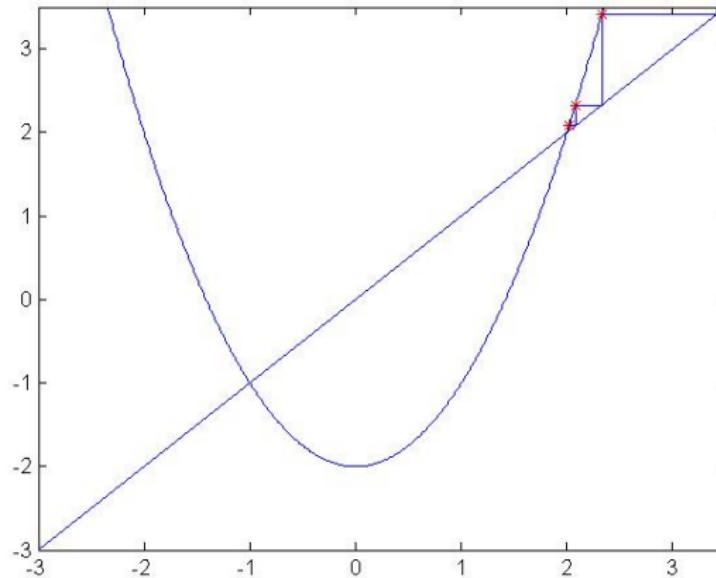
Ví dụ 2

Tìm nghiệm phương trình $f(x) = x^2 - x - 2$ nhờ tìm điểm bất đồng của hàm

$$g(x) = x^2 - 2$$

Điểm xuất phát $x_1 = 2.02$ rất gần nghiệm

Phương pháp lặp



Điểm xuất phát

$$x_1 = 2.02$$

Số bước lặp

$$n = 50$$

Phương pháp lặp

Định lý về sự hội tụ của phương pháp lặp

Định lý 1: Giả sử hàm $g(x)$ là liên tục và dãy lặp

$$x_{k+1} = g(x_k), k = 1, 2, \dots$$

khi đó nếu $x_k \rightarrow x^*$ khi $k \rightarrow \infty$ thì x^* là điểm bất động của g .

Định lý 2: Giả sử $g \in C^1$ và $|g'(x)| < 1$ trong một khoảng nào đó chứa điểm bất động x^* . Nếu x_0 cũng thuộc khoảng này thì dãy lặp $\{x_k\}$ hội tụ tới x^* .

Định lý 3: Nếu g là một **hàm co** thì nó có duy nhất một điểm bất động và dãy lặp $\{x_k\}$ hội tụ tới x^* với mọi điểm xuất phát x_0 .

Lưu ý: Hàm g được gọi là hàm co nếu tồn tại hằng số $L < 1$ sao cho với mọi x, y ta có: $|g(x) - g(y)| < L|x - y|$.

Phương pháp lắp



1 Đặt vấn đề

- Sự tồn tại và duy nhất của nghiệm
- Độ nhạy và điều kiện của bài toán giải phương trình phi tuyến
- Thủ tục lặp

2 Phương pháp chia đôi

3 Phương pháp dây cung

4 Phương pháp Newton

5 Phương pháp cát tuyến

6 Phương pháp lặp

7 Phương pháp Bairstow

8 Tổng kết

Phương pháp Bairstow

Mô tả

Đây là phương pháp dùng để tìm nghiệm của đa thức,:

$$y = a_0 + a_1x + \cdots + a_Nx^N$$

Có thể viết nó lại dưới dạng nhân tử bậc hai cộng một phần dư

$$y = (x^2 + px + q)G(x) + R(x)$$

trong đó,

- p và q là các giá trị tùy ý.
- $G(x)$ là đa thức bậc $N - 2$
- $R(x)$ là phần dư, thường là đa thức bậc nhất.

Phương pháp Bairstow

Mô tả (tiếp)

Vậy đa thức $G(x)$ và phần dư $R(x)$ có dạng

$$G(x) = b_2 + b_3x + b_4x^2 + \cdots + b_Nx^{N-2}$$

$$R(x) = b_0 + b_1x$$

Giá trị của b_0 và b_1 phụ thuộc lựa chọn p và q , mục đích là ta tìm $p = p^*$ và $q = q^*$ sao cho

- $b_0(p^*, q^*) = b_1(p^*, q^*) = 0 \Rightarrow R(x) = 0$
- $(x^2 + p^*x + q^*) \Rightarrow$ nhân tử bậc hai của y

Phương pháp Bairstow

Thủ tục lắp

- ① Khởi tạo giá trị p và q và tính b_0 và b_1 (xem ct trong sách)
- ② Tính các giá trị $(b_0)_p, (b_1)_p$ và $(b_0)_q, (b_1)_q$ (xem ct trong sách)
- ③ Tìm Δp và Δq khi giải phương trình (9)
- ④ Thu được p^* và q^* theo công thức $p^* = p + \Delta p$ và $q^* = q + \Delta q$

Phương pháp Bairstow

Nhận xét

- Ưu điểm:
 - ▶ phương pháp hội tụ đến nhân tử bậc hai ($x^2 + px + q$) bất kể giá trị khởi tạo p, q
 - ▶ các hệ số của đa thức $G(x)$ cũng tự động thu được
- Nhược điểm:
 - ▶ độ chính xác của nghiệm thu được không cao
 - ▶ để cải thiện có thể dùng thêm phương pháp Newton để tính chính xác lại từng nghiệm

Tổng kết

Phương pháp	Biết khoảng phân ly nghiệm	Đòi hỏi liên tục của đạo hàm bậc 1	Kiểu phương trình	Tính năng đặc biệt
Chia đôi	Có	Không	Bất kỳ	Áp dụng đc với hàm không có dạng giải tích
Dây cung	Có	Có	Bất kỳ	Hội tụ chậm khi khoảng phân ly lớn
Newton	Không	Có	Bất kỳ	Hội tụ nhanh Cần tính $f'(x)$
Cát tuyến	Không	Có	Bất kỳ	nt
Lặp	Không	Có	Bất kỳ	Có thể ko hội tụ
Bairstow	Không	Có	Đa thức	Nhân tử bậc 2 Có thể tìm nghiệm phức

Tính toán khoa học

└ Tổng kết

└ Tổng kết

Phương pháp	Bất khoảng phân ly nghiệm	Có bài liên tục của đạo hàm bậc 1	Kiểu phương trình	Tính năng đặc biệt
Chia đôi	Có	Không	Bản kỳ	Áp dụng dc với hàm không có dạng giải tích
Dây cung	Có	Có	Bản kỳ	Hồi tu chậm kh không phân ly lớn
Newton	Không	Có	Bản kỳ	Hồi tu nhanh Cần tính $f'(x)$
Cát tuyến	Không	Có	Bản kỳ	at
Lập	Không	Có	Bản kỳ	Có thể ko hội tụ
Bairstow	Không	Có	Đa thức	Nhận tử bậc 2 Có thể tìm nghiệm phức

Hai phương pháp đầu (chia đôi, dây cung) đều đòi hỏi khoảng phân ly nghiệm. Phương pháp Newton và cát tuyến cần có giá trị phỏng đoán ban đầu. Phương pháp lặp có trở ngại là không phải lúc nào cũng hội tụ. Phương pháp Bairstow hạn chế giải phương trình đa thức, cũng có thể không hội tụ.

Đọc thêm

Các hàm tìm nghiệm trong Matlab

- **X = roots(C)** tìm nghiệm đa thức
- **X = fzero(FUN,X0)** tìm nghiệm phương trình phi tuyến

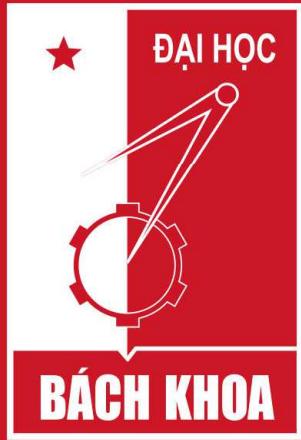


25
SOICT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

**Thank you for
your attentions!**





25 YEARS ANNIVERSARY
SCHOOL OF COMPUTER SCIENCE AND TECHNOLOGY

ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG



ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Chương 5

Tính gần đúng

Đạo hàm và Tích phân

Vũ Văn Thiệu, Đinh Viết Sang, Nguyễn Khánh Phương

TÍNH TOÁN KHOA HỌC

Nội dung

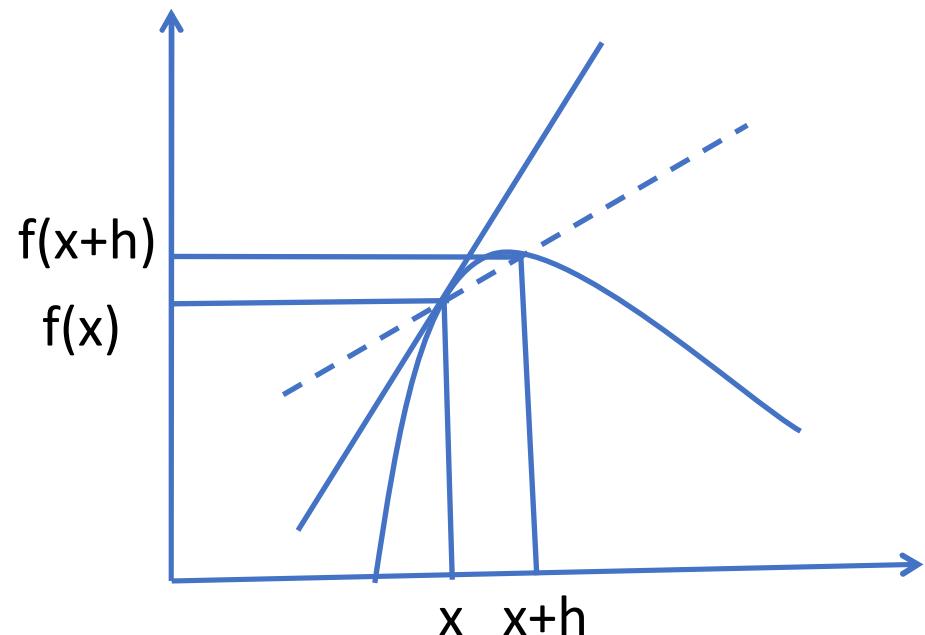
1. Tính gần đúng đạo hàm
2. Tính gần đúng tích phân

5.1. Tính gần đúng đạo hàm: đặt vấn đề

- Định nghĩa đạo hàm bậc nhất:

$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

- Ý nghĩa hình học:
 - $f'(x)$ là hệ số góc của tiếp tuyến tại điểm x
- Tính gần đúng đạo hàm:
 - $h \neq 0$
 - $f'(x)$ là hệ số góc của cát tuyến



5.1.1. Công thức sai phân thuận (Forward difference)

- Xây dựng công thức : Xét khai triển Taylor của hàm f tại lân cận x:

$$f(x+h) = f(x) + f'(x)h + f''(\zeta) \frac{h^2}{2!} \quad (1)$$

Trong đó ξ thuộc đoạn $[x, x+h]$.

Từ (1) ta có:

$$f'(x) = \frac{f(x+h) - f(x)}{h} + f''(\zeta) \frac{h}{2!} \quad (2)$$

Coi số hạng $f''(\xi) h/2$ là sai số rút gọn, từ (2) suy ra:

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} \quad (3)$$

Là công thức tính gần đúng ĐH theo PP sai phân thuận

CT sai phân thuận: Phân tích sai số

- Sai số rút gọn là: $f''(\xi) h^2 = O(h)$

⇒ Phương pháp có độ chính xác bậc nhất

- Sai số làm tròn: Giả sử khi tính $f(x)$ và $f(x+h)$ có sai số làm tròn, công thức tính f' :

$$\frac{f(x+h)(1 + \delta_1) - f(x)(1 + \delta_2)}{h} = \frac{f(x+h) - f(x)}{h} + \frac{\delta_1 f(x+h) - \delta_2 f(x)}{h}$$

Do $|\delta_i|$ nhỏ hơn độ chính xác của máy tính ε nên sai số làm tròn khi tính f' là:

$$\frac{\varepsilon(|f(x+h)| + |f(x)|)}{h}$$

- Sai số tổng cộng đạt tối thiểu khi: $h \approx \sqrt{\varepsilon}$

CT sai phân thuận: Ví dụ

- Xét hàm: $f(x) = \sin x$. Sử dụng CT sai phân thuận để tính gần đúng $f'(\pi/3)$. Phân tích sai số.
 - Tính với $h=10^{-k}$, $k = 1, \dots, 16$
 - Tìm h để có sai số nhỏ nhất

Kết quả

h	Đạo hàm	Sai số
10^{-1}	0.455901885410761	-0.044098114589239
10^{-2}	0.495661575773687	-0.004338424226313
10^{-3}	0.499566904000770	-0.000433095999230
10^{-4}	0.499956697895820	-0.000043302104180
10^{-5}	0.499995669867026	-0.000004330132974
10^{-6}	0.499999566971887	-0.000000433028113
10^{-7}	0.499999956993236	-0.000000043006764
10^{-8}	0.499999996961265	-0.000000003038736
10^{-9}	0.500000041370186	0.000000041370185

5.1.2. Công thức sai phân ngược (Backward difference)

- Xây dựng công thức: Tương tự như trong CT sai phân thuận, khai triển Taylor với $x-h$ thay vì $x+h$, ta có:

$$f'(x) \approx \frac{f(x) - f(x-h)}{h} \quad (1)$$

- Sai số: Tương tự như trong CT sai phân thuận
 - Độ chính xác bậc nhất
 - Sai số nhỏ nhất khi:

$$h \approx \sqrt{\varepsilon}$$

- Bài tập: Sử dụng CT sai phân ngược để tính gần đúng $f'(\pi/3)$, biết $f(x) = \sin x$

5.1.3. Công thức sai phân trung tâm (Central difference)

- Xây dựng công thức : Xét khai triển Taylor của hàm f tại lân cận x:

$$f(x-h) = f(x) - f'(x)h + f''(x)\frac{h^2}{2!} - f'''(\zeta^-)\frac{h^3}{3!} \quad (1)$$

$$f(x+h) = f(x) + f'(x)h + f''(x)\frac{h^2}{2!} + f'''(\zeta^+)\frac{h^3}{3!} \quad (2)$$

Trong đó ξ^+ thuộc đoạn $[x, x+h]$, ξ^- thuộc đoạn $[x-h, x]$.

Từ (1) và (2) ta có công thức tính gần đúng ĐH theo PP sai phân trung tâm

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} \quad (3)$$

CT sai phân trung: Phân tích sai số

- Sai số rút gọn:

$$-\frac{1}{6} f'''(\zeta)h^2, \quad \zeta \in [x-h, x+h]$$

- CT có độ chính xác bậc 2;
- Sai số tổng cộng bé nhất khi $h = \varepsilon^{1/3}$
- Bài tập: Sử dụng PP sai phân trung tâm để tính gần đúng $f'(\pi/3)$, biết $f(x) = \sin x$. So sánh với PP sai phân thuận và sai phân ngược

So sánh sai số 3 phương pháp

h tâm	Sai phân thuận	Sai phân ngược	Sai phân trung
10^{-1}	$\sim 10^{-2}$	$\sim 10^{-2}$	$\sim 10^{-4}$
10^{-2}	$\sim 10^{-3}$	$\sim 10^{-3}$	$\sim 10^{-6}$
10^{-3}	$\sim 10^{-4}$	$\sim 10^{-4}$	$\sim 10^{-8}$
10^{-4}	$\sim 10^{-5}$	$\sim 10^{-5}$	$\sim 10^{-10}$
10^{-5}	$\sim 10^{-6}$	$\sim 10^{-6}$	$\sim 10^{-12}$
10^{-6}	$\sim 10^{-7}$	$\sim 10^{-7}$	$\sim 10^{-11}$
10^{-7}	$\sim 10^{-8}$	$\sim 10^{-8}$	$\sim 10^{-10}$
10^{-8}	$\sim 10^{-9}$	$\sim 10^{-9}$	$\sim 10^{-9}$
10^{-9}	$\sim 10^{-8}$	$\sim 10^{-8}$	$\sim 10^{-8}$

5.1.4. Tính gần đúng đạo hàm cấp cao: Đạo hàm cấp 2

- Xét khai triển Taylor của hàm f tại lân cận x:

$$f(x-h) = f(x) - f'(x)h + f''(x)\frac{h^2}{2!} - f'''(x)\frac{h^3}{3!} + f''''(x)\frac{h^4}{4!} - f'''''(x)\frac{h^5}{5!} + \dots \quad (1)$$

$$f(x+h) = f(x) + f'(x)h + f''(x)\frac{h^2}{2!} + f'''(x)\frac{h^3}{3!} + f''''(x)\frac{h^4}{4!} + f'''''(x)\frac{h^5}{5!} + \dots \quad (2)$$

Từ (1) và (2) ta có công thức tính gần đúng ĐH bậc 2

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \quad (3)$$

- Sai số rút gọn:

- Sai số bé nhất khi $h = \varepsilon^{1/4}$

$$-\frac{1}{12} f''''(\zeta) h^2, \quad \zeta \in [x-h, x+h]$$

5.1.5. Tính gần đúng đạo hàm riêng

- Tương tự, ta có thể xây dựng các PP tính gần đúng đạo hàm riêng, ví dụ PP sai phân trung tâm tính đạo hàm riêng cho hàm $f(x,y)$ như sau:

$$\frac{\partial f(x,y)}{\partial x} = \frac{f(x+h,y) - f(x-h,y)}{2h}$$

$$\frac{\partial f(x,y)}{\partial y} = \frac{f(x,y+h) - f(x,y-h)}{2h}$$

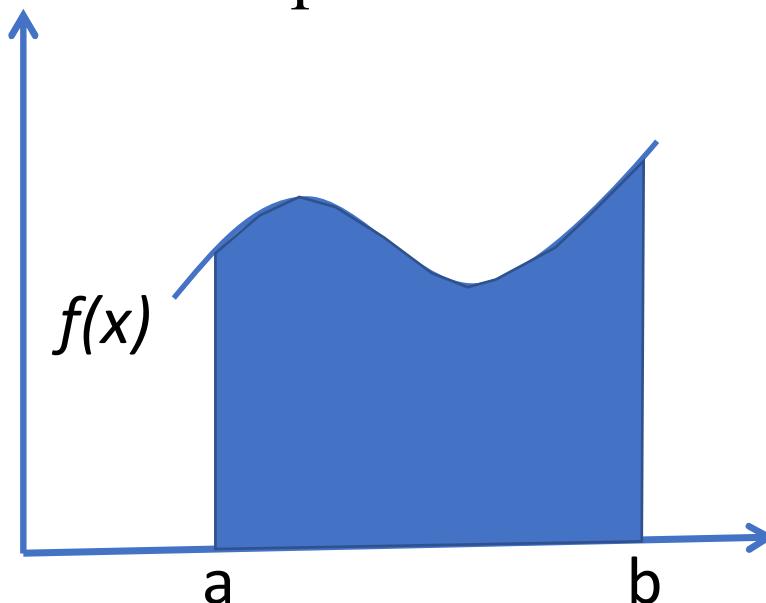
5.2. Tính gần đúng tích phân: đặt ván đề

- Tích tích phân:

$$I = \int_a^b f(x)dx,$$

trong đó $f(x)$ là hàm khả tích trên đoạn $[a,b]$

- Ý nghĩa hình học của tích phân:



5.2.1. Tính gần đúng tích phân: Tổng Riemann

- Giả sử hàm f xác định trên $[a,b]$ và Δ là phép chia đoạn $[a,b]$ thành n đoạn đóng $I_k = [x_{k-1}, x_k]$, $k=1, \dots, n$, trong đó $a = x_0 < x_1 < \dots < x_{n-1} < x_n = b$. Chọn n điểm $\{c_k\}$: $k=1, \dots, n\}$, mỗi điểm thuộc đoạn con, nghĩa là: c_k thuộc I_k với mọi k . Tổng

$$\sum_{k=1}^n f(c_k) \Delta x_k = f(c_1) \Delta x_1 + f(c_2) \Delta x_2 + \dots + f(c_n) \Delta x_n$$

được gọi là tổng Riemann của hàm $f(x)$ tương ứng với phép chia Δ và các điểm chọn lọc $\{c_k\}$: $k=1, \dots, n\}$.

5.2.2. Tính gần đúng tích phân: Định nghĩa

- Tích phân xác định của hàm $f(x)$ theo x từ a đến b là giới hạn của tổng Riemann

$$\int_a^b f(x)dx = \lim_{n \rightarrow \infty} \sum_{k=1}^n f(c_k)\Delta x_k,$$

Với giả thiết là giới hạn này tồn tại.

- Hàm $f(x)$ gọi là hàm cần tích phân
- a, b là các cận tích phân
- $[a,b]$ là khoảng tích phân

5.2.3. Tính gần đúng tích phân: Các tính chất của tích phân xác định

$$\int_a^a f(x) dx = 0$$

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

$$\int_a^b C \cdot f(x) dx = C \cdot \int_a^b f(x) dx$$

$$\int_a^b (f(x) + g(x)) dx = \int_a^b f(x) dx + \int_a^b g(x) dx$$

$$\int_a^b f(x) dx = \int_a^c f(x) dx + \int_c^b f(x) dx , \quad c \in [a, b]$$

5.2.4. Tính gần đúng tích phân: Các định lý

- ĐL1: Nếu f là liên tục trên $[a,b]$ và F là nguyên hàm của hàm f ($F' = f$) thì:

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

- ĐL2 (ĐL về giá trị trung bình): Nếu f là liên tục trên $[a,b]$ thì tồn tại số c trong đoạn $[a,b]$ sao cho:

$$f(c) = \frac{1}{b-a} \int_a^b f(x)dx$$

5.2.5. Tính gần đúng tích phân: Công thức Newton-Cotes (1)

- Cách tiếp cận đầu tiên để xây dựng công thức tính gần đúng tích phân là xấp xỉ hàm $f(x)$ trên khoảng tích phân $[a,b]$ bởi một đa thức. Trong mỗi khoảng con ta xấp xỉ hàm $f(x)$ bởi một đa thức:

$$p_m(x) = a_0 + a_1x + a_2x^2 + \dots + a_mx^m$$

(1)

Ta có thể dễ dàng tính chính xác tích phân của (1)

- Đơn giản nhất ta có thể thay hàm $f(x)$ bởi đa thức nội suy.

Tính gần đúng tích phân: PP Newton-Cotes (2)

- Thay $f(x)$ bằng đa thức nội suy Lagrange ta có:

$$\begin{aligned} \int_a^b f(x) dx &= \int_b^a \left(\sum_{i=0}^m \prod_{\substack{j=0 \\ j \neq i}}^m \frac{x - x_j}{x_i - x_j} f(x_i) \right) dx \\ &= \sum_{i=0}^m f(x_i) \int_a^b \prod_{\substack{j=0 \\ j \neq i}}^m \frac{x - x_j}{x_i - x_j} dx \end{aligned} \quad (1)$$

Tính gần đúng tích phân: PP Newton-Cotes (3)

- Sai số của PP được đánh giá bởi:

$$\int_a^b f(x)dx - \int_a^b p_m(x)dx = \frac{1}{(m+1)!} \int_b^a f^{(m+1)}(\zeta_x) \left(\prod_{i=0}^m (x - x_i) \right) dx$$
$$\zeta_x \in [a, b] \quad (2)$$

Tính gần đúng tích phân: PP Newton-Cotes (4)

- Các công thức tính gần đúng tích phân thu được theo cách tiếp cận này trong đó sử dụng lưới chia cách đều trong khoảng tích phân, nghĩa là:

$$x_i = a + i * h; \quad i=0, 1, \dots, m; \quad h = (b-a)/m,$$

được gọi là công thức Newton-Cotes.

- Với m khác nhau, ta có các PP khác nhau

m	Bậc đa thức	Công thức	Sai số
1	Tuyến tính	Hình thang	$O(h^2)$
2	Bậc 2	Simpson 1/3	$O(h^4)$
3	Bậc 3	Simpson 3/8	$O(h^4)$

5.2.6. Tính gần đúng tích phân: Công thức hình thang (Trapezoidal rule)

- Với $n=1$, đa thức nội suy có dạng:

$$\begin{aligned} p_1(x) &= f(a) + \frac{f(b) - f(a)}{b - a}(x - a) \\ \Rightarrow I &= \int_a^b f(x)dx \approx \int_a^b p_1(x)dx = \int_a^b \left(f(a) + \frac{f(b) - f(a)}{b - a}(x - a) \right) dx \\ \Rightarrow I &= \frac{(f(a) + f(b))}{2}(b - a) \quad (1) \end{aligned}$$

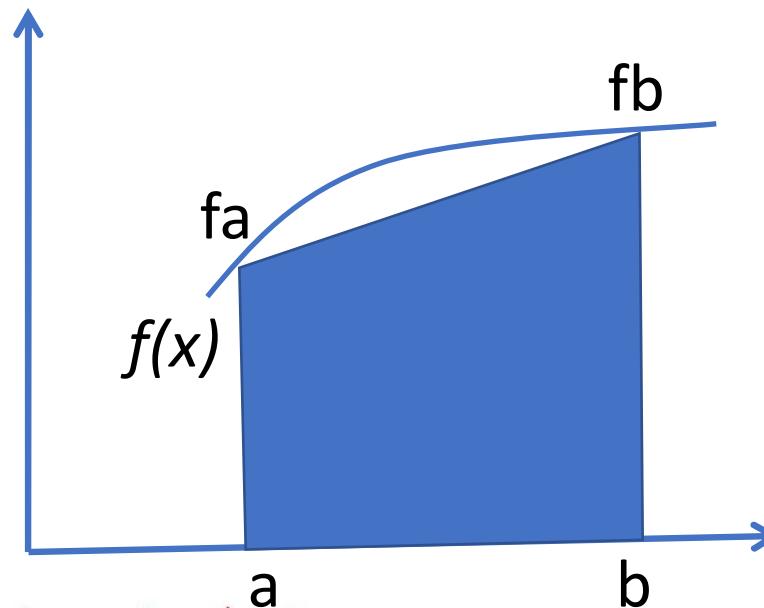
- (1) gọi là công thức hình thang tính gần đúng tích phân

Tính gần đúng tích phân: Công thức hình thang (2)

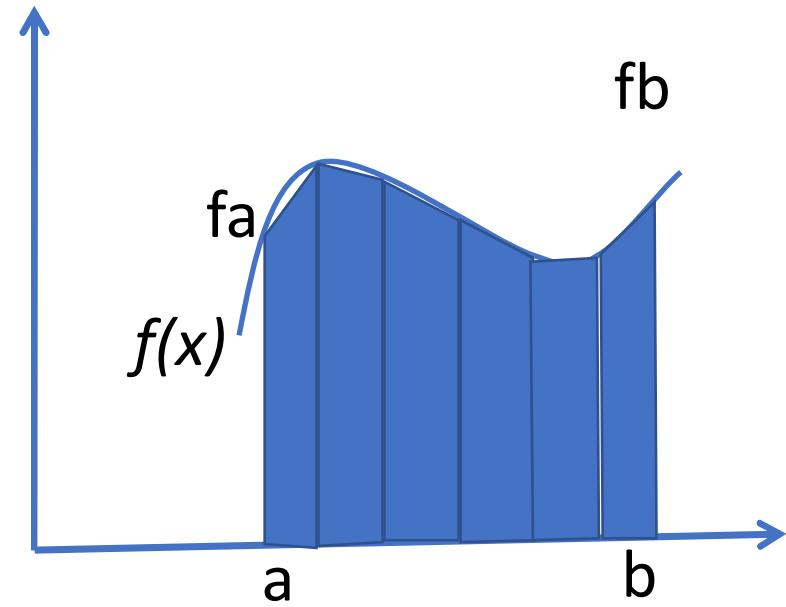
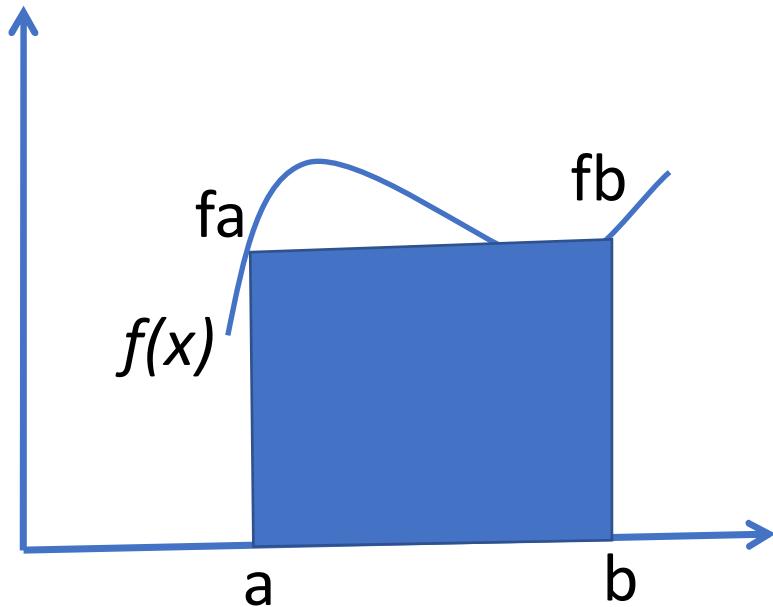
- Sai số của CT hình thang:

$$-\frac{b-a}{12} f''(\zeta) h^2, \quad h = b-a, \quad \zeta \in [a,b]$$

- Ý nghĩa hình học:



5.2.7. Tính gần đúng tích phân: Công thức hình thang mở rộng (1)



- Ý tưởng công thức hình thang mở rộng: Chia nhỏ đoạn $[a,b]$ để giảm sai số

Tính gần đúng tích phân: Công thức hình thang mở rộng (2)

- Chia đoạn $[a,b]$ thành n khoảng bằng nhau dùng $n+1$ điểm: $x_0 = a$, $x_1 = a + h$, $x_{n-1} = a + (n-1)*h$, $x_n = a + n*h$ trong đó $h = (b-a)/n$, ta có:

$$I = \int_a^b f(x)dx = \int_a^{a+h} f(x)dx + \int_{a+h}^{a+2h} f(x)dx + \dots + \int_{a+(n-1)h}^{a+nh} f(x)dx \quad (1)$$

- Áp dụng công thức hình thang cho mỗi đoạn ta có:

$$I = \frac{h}{2} \left[f(a) + 2 \sum_{i=1}^{n-1} f(a + ih) + f(b) \right] \quad (2)$$

- (2) gọi là công thức hình thang mở rộng

5.2.8. Tính gần đúng tích phân: Công thức Simpson 1/3

- Thay n=2 vào công thức Newton-Cotes rồi tính tích phân, ta được:

$$I = \int_a^b f(x)dx = \frac{h}{3} [f(x_0) + 4f(x_1) + f(x_2)]$$
$$x_0 = a, \quad x_1 = a + h, \quad x_2 = a + 2h = b, \quad (1)$$

- (1) gọi là công thức Simpson 1/3

5.2.9. Tính gần đúng tích phân: Công thức Simpson 1/3 mở rộng

- Giống như CT hình thang mở rộng, ta chia đoạn tích phân $[a,b]$ thành nhiều khoảng con và áp dụng CT Simpson 1/3 cho mỗi khoảng con, ta thu được CT Simpson mở rộng:

$$I = \int_a^b f(x)dx = (b-a) \frac{f(x_0) + 4 \sum_{i=1,3,5,\dots}^{n-1} f(x_i) + 2 \sum_{j=2,4,6,\dots}^{n-2} f(x_j) + f(x_n)}{3n}$$

$$x_0 = a, \quad x_i = a + ih, \quad i = 1, \dots, n, \quad (1)$$

- Chú ý: Ta cần số khoảng con chẵn, hay số điểm lẻ.

5.2.10. Tính gần đúng tích phân: Công thức Simpson 3/8

- Thay n=3 vào công thức Newton-Cotes rồi tính tích phân, ta được:

$$I = \int_a^b f(x)dx = \frac{3h}{8} [f(x_0) + 3f(x_1) + 3f(x_2) + f(x_3)]$$

$$x_0 = a, \quad x_1 = a + h, \quad x_2 = a + 2h, \quad x_3 = a + 3h \quad (1)$$

- (1) gọi là công thức Simpson 3/8



25 YEARS ANNIVERSARY
SOICT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Thank you
for your
attentions!

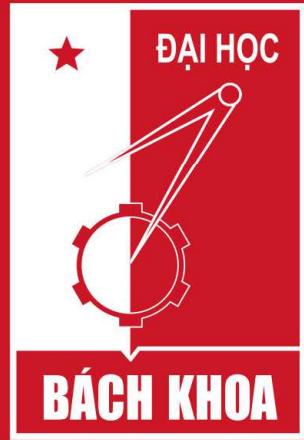


soict.hust.edu.vn/



fb.com/groups/soict





25 YEARS ANNIVERSARY
SCHOOL OF COMPUTER SCIENCE AND TECHNOLOGY

ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG



ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Chương 6

Bài toán giá trị ban đầu đối với phương trình vi phân thường

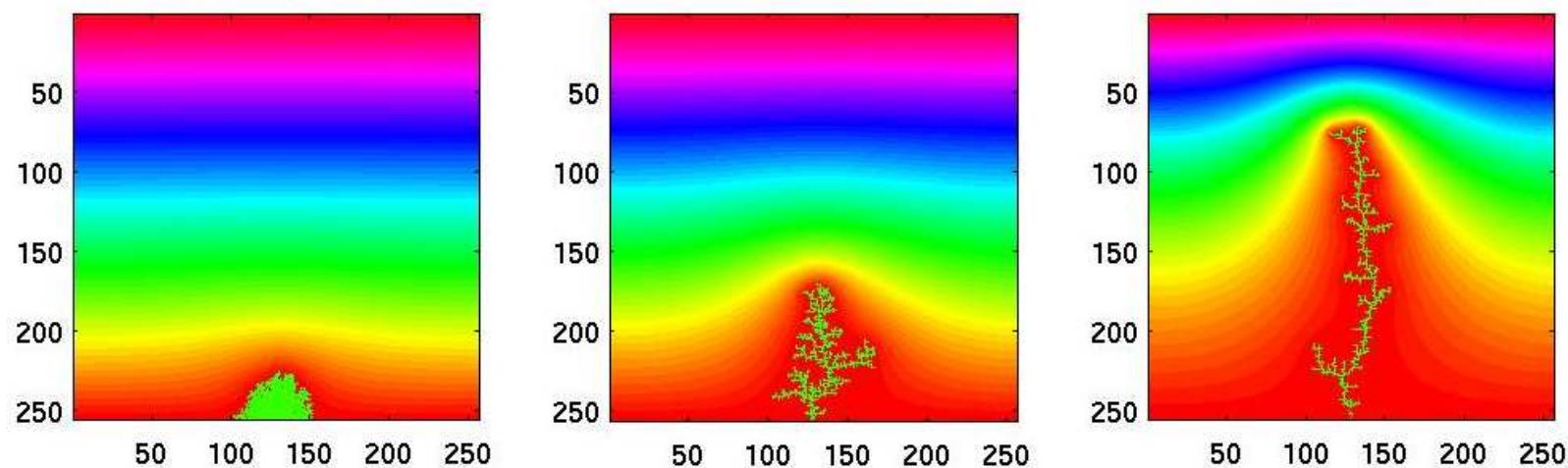
Vũ Văn Thiệu, Đinh Viết Sang, Nguyễn Khánh Phương

TÍNH TOÁN KHOA HỌC

Nội dung

- Mở đầu: Phương trình vi phân
- Phương pháp Euler
 - Phương pháp Euler thuận
 - Phương pháp Euler cải biên
 - Phương pháp Euler ngược
- Phương pháp Runge – Kutta
 - Phương pháp Runge – Kutta bậc 2
 - Phương pháp Runge – Kutta bậc 3
 - Phương pháp Runge – Kutta bậc 4
- Các hàm trên Matlab
- Bài tập

Ví dụ



Mở đầu (1)

- Bài toán tìm nghiệm của các phương trình vi phân (PTVP) thường được chia làm 2 loại: bài toán giá trị ban đầu và bài toán điều kiện biên phụ thuộc vào việc ta cần tìm nghiệm thỏa mãn điều kiện ban đầu hay điều kiện biên.
- Đa số các bài toán giá trị ban đầu mô tả các hệ thống được xét phụ thuộc thời gian và lời giải của bài toán phụ thuộc vào điều kiện tại thời điểm ban đầu

Mở đầu (2)

- Bài toán giá trị ban đầu (IVP: Input Value Problem) đối với PTVP cấp 1 có thể viết dưới dạng:

$$y'(t) = f(y, t); y(t_0) = y_0$$

- y' là đạo hàm bậc nhất của y , $f(y, t)$ là hàm của hai biến y và t , $y(t_0)=y_0$ là điều kiện ban đầu của bài toán.
- Nếu f không phụ thuộc vào y thì có thể tính y' bằng cách lấy tích phân của hàm f .
- Nếu f phụ thuộc vào y ?

Ví dụ

- VD1: Tốc độ tăng trưởng dân số phụ thuộc vào dân số. Nếu dân số tại thời điểm t là $y(t)$ thì tốc độ tăng dân số tại thời điểm t là

$$y'(t) = k y(t)$$

trong đó k là một hằng số dương.

- VD2: Phương trình Lotka-Volterra về thú săn mồi (cáo)- con mồi (thỏ): Gọi $F(t)$ là số lượng cáo và $R(t)$ là số lượng thỏ tại thời điểm t , ta có:

$$R'(t) = (a - bF)R$$

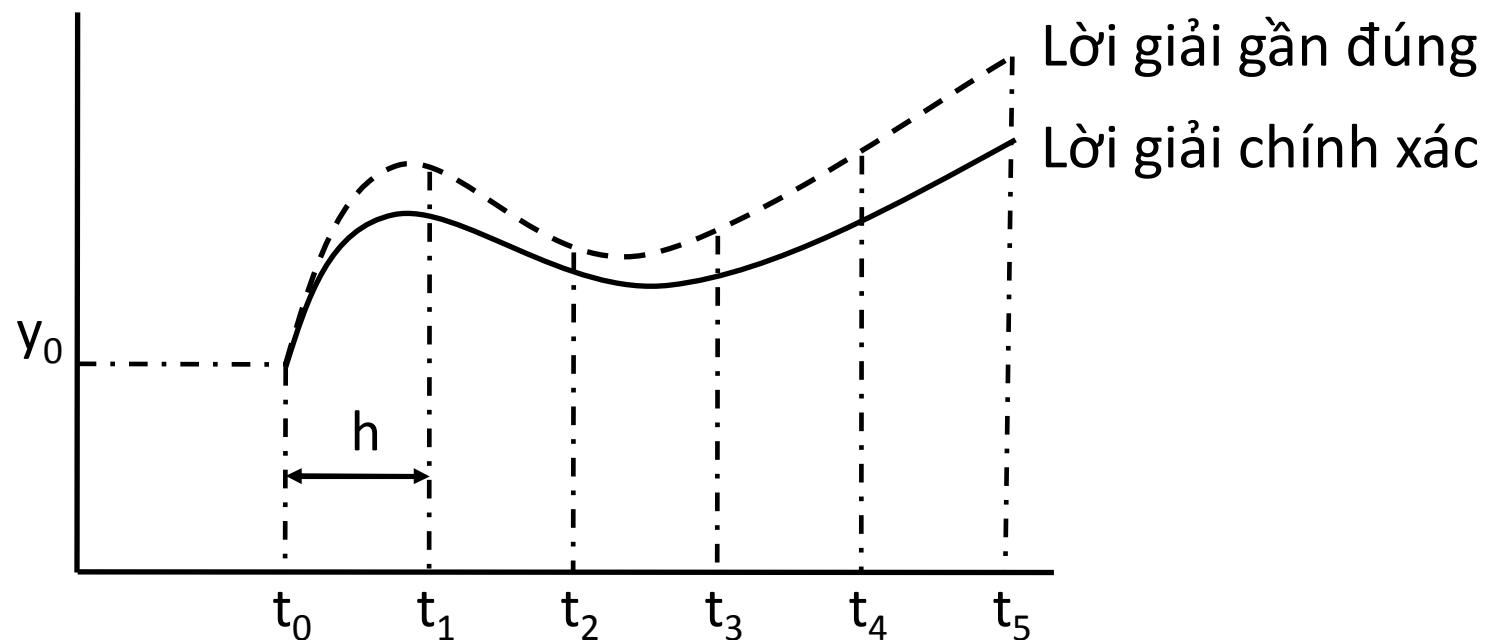
$$F'(t) = (cR - d)F$$

trong đó a,b,c,d là các hằng số dương

- Các PTVP trên rất khó có thể giải bằng phương pháp giải tích

Giải PTVP bằng phương pháp gần đúng (phương pháp số)

- Phương pháp số đòi hỏi tính giá trị tại lưới điểm theo thời gian
 $t_n = t_{n-1} + h$, $n = 1, 2, \dots$, h là độ dài bước



Sự tồn tại và tính duy nhất của nghiệm

- **Định lý 1:** Nếu f là hàm liên tục trên hình chữ nhật:

$$R = \{(t,y) : |t - t_0| \leq \alpha, |y - y_0| \leq \beta\}$$

thì IVP có nghiệm $y(t)$ với $|t - t_0| \leq \min\{\alpha, \beta/M\}$, trong đó $M = \max\{|f(t,y)| : (t,y) \in R\}$.

- **Định lý 2:** Nếu f và $\delta f/\delta y$ liên tục trên hình chữ nhật

$$R = \{(t,y) : |t - t_0| \leq \alpha, |y - y_0| \leq \beta\}$$

thì IVP có nghiệm duy nhất $y(t)$ với $|t - t_0| \leq \min\{\alpha, \beta/M\}$, trong đó $M = \max\{|f(t,y)| : (t,y) \in R\}$.

- **Định lý 3:** Giả sử t_0 nằm trong đoạn $[a,b]$. Nếu f liên tục với $a \leq t \leq b$, $-\infty \leq y \leq \infty$ và liên tục lipschitz đều theo y , nghĩa là tìm được hằng số L sao cho với mọi y_1, y_2 và t thuộc $[a,b]$ ta có $|f(t,y_2) - f(t,y_1)| \leq L|y_2 - y_1|$
thì IVP có nghiệm duy nhất $y(t)$ trên đoạn $[a,b]$.

Phương pháp Euler thuận (1)

- Xét PTVP: $y' = f(y, t)$, PP Euler thuận thu được bằng cách sử dụng sai phân xấp xỉ thuận

$$\frac{y_{n+1} - y_n}{h} \approx y_n' = f(y_n, t_n) \quad (1)$$

$$(1) \Rightarrow y_{n+1} = y_n + hf(y_n, t_n) \quad (2)$$

- Sử dụng (2), y_n được tính đệ quy như sau:

$$\begin{aligned} y_1 &= y_0 + hf(y_0, t_0) \\ y_2 &= y_1 + hf(y_1, t_1) \\ &\dots\dots \\ y_n &= y_{n-1} + hf(y_{n-1}, t_{n-1}) \end{aligned} \quad (3)$$

Phương pháp Euler thuận (2): Ví dụ

- Xét PTVP: $y' = -20y + 7e^{-0.5t}$, $y(0) = 5$ (4)

Giải PTVP (4) với t thuộc $[0,0.04]$, $h=10^{-2}; 10^{-3}; 10^{-4}$

Đánh giá sai số biết nghiệm chính xác của (4) là:

$$y = 5e^{-20t} + (7/19.5)(e^{-0.5t} - e^{-20t})$$

Phương pháp Euler thuận (2): Ví dụ

t	$h = 0.01$		$h = 0.001$		$h = 0.0001$	
	Kết quả	Sai số	Kết quả	Sai số	Kết quả	Sai số
0.01	4.07000	0.08693	4.14924	0.0769	4.15617	0.0076
0.02	3.32565	0.14072	3.45379	0.1259	3.46513	0.0124
0.03	2.72982	0.17085	2.88524	0.1544	2.89915	0.0153
0.04	1.87087	0.18440	2.42037	0.1684	2.43554	0.0167

Phương pháp Euler thuận (3)

- PP Euler thuận rất đơn giản, nhưng có hai nhược điểm:
 - Sai số làm tròn lớn như trong ví dụ 1
 - Tính không ổn định xuất hiện khi hằng số thời gian của phương trình âm, trừ khi bước thời gian h đủ nhỏ.
- VD xét PTVP: $y' = -\alpha y$, $y(0) = y_0$
trong đó $y_0 > 0$, $\alpha > 0$.

Lời giải chính xác của bài toán là: $y = y_0 e^{-\alpha t}$ tiến tới 0 khi t tăng. Nếu giải bằng PP Euler thuận thì:

- Nếu $\alpha h < 1$ thì lời giải được thu nhỏ và dương
- Nếu $\alpha h > 1$ thì dấu của lời giải là xen kẽ nhau. Đặc biệt nếu $\alpha h > 2$ thi biên độ của lời giải tăng theo từng bước, và lời giải dao động.

=> Không ổn định

Phương pháp Euler thuận (4) đối với hệ PTVP

- Xét hệ PTVP thường cấp 1:

$$y' = f(y, z, t), \quad y(0) = y_0$$

$$z' = g(y, z, t), \quad z(0) = z_0 \quad (5)$$

Phương pháp Euler thuận đối với hệ PTVP (4) được viết như sau:

$$y_{n+1} = y_n + h f(y_n, z_n, t_n)$$

$$z_{n+1} = z_n + h g(y_n, z_n, t_n) \quad (6)$$

Phương pháp Euler thuận (5) đối với PTVP bậc cao

- Để giải các PTVP bậc cao ta có thể phân rã nó thành hệ các PTVP bậc 1.
- VD: Xét PTVP bậc 2:

$$\begin{aligned}y''(t) - 0.05 y'(t) + 0.15 y(t) &= 0 \\y'(0) &= 0 \\y(0) &= 1\end{aligned}\tag{7}$$

Đặt $y' = z$ và viết lại (7) dưới dạng

$$\begin{aligned}y' &= z, & y(0) &= 1, \\z' &= 0.05z - 0.15 y, & z(0) &= 0\end{aligned}\tag{8}$$

Giải (8) như là hệ PTVP, dùng công thức (6)

Phương pháp Euler cải biên (1)

- PP Euler cải biên chính xác và ổn định hơn PP Euler thuận
- PP Euler cải biên dựa trên quy tắc hình thang để tính tích phân
 $y' = y(t)$:

$$y_{n+1} = y_n + h/2 [f(y_{n+1}, t_{n+1}) + f(y_n, t_n)] \quad (9)$$

- Nếu f là tuyến tính với y thì (8) là tuyến tính với y_{n+1} , do đó ta có thể dễ dàng xác định y_{n+1}
- Nếu f là không tuyến tính với y thì (8) là phi tuyến tính y_{n+1} . Việc tìm y_{n+1} giống việc giải phương trình phi tuyến như đã học trong Chương 4.

Phương pháp Euler cải biên (2): VD

- Sử dụng PP Euler cải biên với $h=0.1$ để giải PTVP:

$$y' = -y^{1.5} + 1, \quad y(0) = 10, \text{ với } 0 \leq t \leq 1$$

- Áp dụng PP Euler cải biên ta được:

$$y_{n+1} = y_n + h/2 [-(y_{n+1})^{1.5} - (y_n)^{1.5} + 2] \quad (10)$$

Với $n = 0$ ta có

$$y_1 = y_0 + h/2 [-(y_1)^{1.5} - (y_0)^{1.5} + 2]$$

xấp xỉ tốt nhất cho y_1 ở vế phải là y_0 .

Đặt $y_1 = y_0$, ta có:

$$y_1 = y_0 + h/2 [-(y_0)^{1.5} - (y_0)^{1.5} + 2] \quad (11)$$

- Tương tự ta tính được y_n

Phương pháp Euler ngược

- Xét PTVP: $y' = f(y, t)$, PP Euler ngược thu được bằng cách sử dụng sai phân xấp xỉ ngược

$$\frac{y_{n+1} - y_n}{h} \approx y'_{n+1} = f(y_{n+1}, t_{n+1}) \quad (1)$$

$$(1) \Rightarrow y_{n+1} = y_n + hf(y_{n+1}, t_{n+1}) \quad (2)$$

- Chú ý: (2) chưa cho ta công thức hiện bởi vì ta phải tính giá trị của hàm f đối với đối số y_{n+1} còn chưa biết. Để tìm y_{n+1} ta có thể giải (2) như là phương trình phi tuyến như đã trình bày trong Chương 4

Phương pháp Euler ngược: VD

- Xét PTVP: $y' = y^3$, $y(0) = 1$. Thực hiện PP Euler ngược với $h=0.5$, ta có:

$$y_1 = y_0 + h f(y_1, t_1) = 1 + 0.5 (y_1)^3 \quad (3)$$

(3) có thể giải bằng phương pháp Newton:

$$y_k = y_{k-1} - f(y_{k-1})/f'(y_{k-1})$$

(3) giải bằng Matlab:

```
fzero('x+0.5*x^3-1',1)
```

Tổng kết các phương pháp Euler

- Phương pháp Euler thuận dựa trên xấp xỉ sai phân thuận. Sai số trong một khoảng lặp của nó tỉ lệ với h^2 và sai số toàn cục tỉ lệ với h . PP này đơn giản nhưng sai số lớn và độ không ổn định cao.
- Phương pháp Euler cải biên dựa trên quy tắc hình thang. Sai số trong một khoảng lặp của nó tỉ lệ với h^3 và sai số toàn cục tỉ lệ với h^2 .
- Phương pháp Euler ngược dựa trên xấp xỉ sai phân ngược. Sai số của nó tương tự như phương pháp sai phân thuận. Tuy nhiên phương pháp này ổn định, vì vậy được dùng để giải những bài toán không trơn (rất khó giải bằng các phương pháp khác).

Phương pháp Runge-Kutta

- Nhược điểm của PP Euler là bậc của độ chính xác giảm dần. Muốn có độ chính xác cao đòi hỏi h phải rất nhỏ.
- Trong PP Runge-Kutta, bậc của độ chính xác được tăng lên bằng cách sử dụng các điểm trung gian trong mỗi bước lặp.
- Xét PTVP: $y' = f(y, t)$, $y(0) = y_0$ (1)

Để tính y_{n+1} tại $t_{n+1} = t_n + h$ với y_n đã biết, ta lấy tích phân phương trình trên trong khoảng $[t_n, t_{n+1}]$ như sau:

$$y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} f(y, t) dt \quad (2)$$

PP Runge-Kutta được phát triển nhờ áp dụng các PP tính tích phân số để tính tích phân ở bên phải của (2).

Phương pháp Runge-Kutta bậc 2 (1)

- Chúng ta khảo sát một ứng dụng của quy tắc hình thang vào vế phải của (2) như sau:

$$\int_{t_n}^{t_{n+1}} f(y, t) dt = \frac{1}{2} h [f(y_n, t_n) + f(\bar{y}_{n+1}, t_{n+1})] \quad (3)$$

Trong (3) thì y_{n+1} chưa biết. Như vậy số hạng thứ 2 được xấp xỉ bởi $f(\bar{y}_{n+1}, t_{n+1})$, trong đó \bar{y}_{n+1} là ước tính đầu tiên của y_{n+1} được tính theo PP Euler thuận.

- PP thu được theo cách này gọi là PP Runge-Kutta bậc 2

Phương pháp Runge-Kutta bậc 2 (2)

- Công thức PP Runge-Kutta bậc 2:

$$\overline{y_{n+1}} = y_n + hf(y_n, t_n) \quad (4)$$

$$y_{n+1} = y_n + \frac{h}{2} [f(y_n, t_n) + f(\overline{y_{n+1}}, t_{n+1})]$$

hoặc ta có thể viết dưới dạng sau:

$$\begin{aligned} k_1 &= hf(y_n, t_n) \\ k_2 &= hf(y_n + k_1, t_{n+1}) \end{aligned} \quad (5)$$

$$y_{n+1} = y_n + \frac{1}{2} [k_1 + k_2]$$

Phương pháp Runge-Kutta bậc 2 (3)

- PP Runge-Kutta bậc 2 tương đương với PP Euler cải biên chỉ áp dụng với một bước lặp.
- Độ chính xác của PP Runge-Kutta bậc 2 là h^2 , trùng với PP Euler cải biên với điều kiện thủ tục lặp giải phương trình phi tuyến trong nó là hội tụ. Như vậy việc sử dụng PP Runge-Kutta bậc 2 với bước nhảy h đủ nhỏ là tốt hơn so với sử dụng PP Euler cải biên.
- Việc sử dụng PP Runge-Kutta khá đơn giản.

Phương pháp Runge-Kutta bậc 2 (4)

- VD: Xét PTVP bậc 2 sau:

$$y''(t) + a y'(t) + b y(t) = q(t)$$

$$y(0) = 1, \quad y'(0) = 0, \quad (6)$$

trong đó a, b là các hằng số, $q(t)$ đã biết. Đặt $z=y'(t)$ ta có:

$$y' = z, \quad y(0) = 1,$$

$$z' = q(t) - a z - b y, \quad z(0) = 0 \quad (7)$$

Phương pháp Runge-Kutta bậc 2 (5)

- PP Runge-Kutta cho (7) có dạng như sau:

$$k_1 = h z_n$$

$$l_1 = h (q_n - a z_n - b y_n)$$

$$k_2 = h (z_n + l_1)$$

$$l_2 = h (q_{n+1} - a (z_n + l_1) - b (y_n + k_1)) \quad (8)$$

$$y_{n+1} = y_n + \frac{1}{2} (k_1 + k_2)$$

$$z_{n+1} = z_n + \frac{1}{2} (l_1 + l_2)$$

Phương pháp Runge-Kutta bậc 3 (1)

- PP Runge-Kutta bậc 3 dựa trên việc áp dụng sơ đồ tích phân bậc chính xác cao hơn cho số hạng thứ 2 trong phương trình (2). Sử dụng quy tắc Simson 1/3, (2) được xấp xỉ bởi:

$$y_{n+1} = y_n + h/6 [f(y_n, t_n) + 4f(\bar{y}_{n+1/2}, t_{n+1/2}) + f(\bar{y}_{n+1}, t_1)] \quad (9)$$

trong đó $\bar{y}_{n+1/2}$ và \bar{y}_{n+1} là ước tính (vì $y_{n+1/2}$ và y_{n+1} chưa biết) như sau:

$$\bar{y}_{n+1/2} = y_n + h/2 f(y_n, t_n) \quad (10)$$

$$\bar{y}_{n+1} = y_n + h[\theta f(y_n, t_n) + (1-\theta) f(\bar{y}_{n+1/2}, t_{n+1/2})] \quad (11)$$

trong đó θ (chưa biết) dùng để xác định độ chính xác cực đại của phương pháp.

Phương pháp Runge-Kutta bậc 3 (2)

- Thay (10) và (11) vào (9), ta thu được PP Runge-Kutta bậc 3 dưới dạng sau:

$$k_1 = h f(y_n, t_n)$$

$$k_2 = h f(y_n + \frac{1}{2} k_1, t_n + \frac{1}{2} h)$$

$$k_3 = h f(y_n + \theta k_1 + (1-\theta)k_2, t_n + h) \quad (12)$$

$$y_{n+1} = y_n + \frac{1}{6} (k_1 + 4k_2 + k_3)$$

- Có thể chứng minh được rằng $\theta = -1$ là tối ưu.

Phương pháp Runge-Kutta bậc 4 (1)

- Việc phát triển PP Runge-Kutta bậc 4 cũng tương tự như PP Runge-Kutta bậc 3, ngoại trừ có thêm bước trung gian tính đạo hàm. PP Runge-Kutta bậc 4 có sai số địa phương tỉ lệ với h^3 .
- PP Runge-Kutta bậc 4 dựa trên quy tắc Simson 1/3:

$$k_1 = h f(y_n, t_n)$$

$$k_2 = h f(y_n + \frac{1}{2} k_1, t_n + \frac{1}{2} h)$$

$$k_3 = h f(y_n + \frac{1}{2} k_2, t_n + \frac{1}{2} h) \quad (13)$$

$$k_4 = h f(y_n + k_3, t_n + h)$$

$$y_{n+1} = y_n + \frac{1}{6} (k_1 + 2k_2 + 2k_3 + k_4)$$

Phương pháp Runge-Kutta bậc 4 (2)

- PP Runge-Kutta bậc 4 dựa trên quy tắc Simson 3/8:

$$k_1 = h f(y_n, t_n)$$

$$k_2 = h f(y_n + 1/3 k_1, t_n + 1/3 h)$$

$$k_3 = h f(y_n + 1/3 k_1 + 1/3 k_2, t_n + 2/3 h) \quad (14)$$

$$k_4 = h f(y_n + k_1 - k_2 + k_3, t_n + h)$$

$$y_{n+1} = y_n + 1/8 (k_1 + 3k_2 + 3k_3 + k_4)$$

Phương pháp Runge-Kutta bậc 4 (3)

- VD1: Tính $y(1)$ bằng cách giải PTVP

$$y' = -1/(1+y^2),$$

$$y(0)=1,$$

sử dụng PP Runge-Kutta bậc 4 với $h=1$

Các hàm để giải PTVP trên Matlab

- Các hàm giải PTVP: ODE45, ODE113, ODE15S, ODE23S, ODE23T, ODE23TB
- Các hàm thiết lập tùy chọn: ODESET, ODEGET
- Các hàm đưa ra kết quả: ODEPLOT, ODEPHAS2, ODEPHAS3, ODEPRINT
- Tìm công thức giải tích của nghiệm: dsolve

Giải PTVP trên Matlab: ODE23 (1)

- Lệnh gọi:

$$[T, Y] = \text{ODE23}(\text{ODEFUN}, \text{TSPAN}, Y_0)$$

- Các tham số đầu vào:

- TSPAN là khoảng tích phân $[t_0, t_1]$
- Y_0 là giá trị ban đầu
- $\text{ODEFUN}(T, Y)$ trả lại véc tơ cột tương ứng với giá trị $f(t, y)$

- Kết quả:

- Mỗi dòng trong mảng kết quả Y tương ứng với thời gian trong véc tơ cột T

Giải PTVP trên Matlab: ODE23 (2)

- VD1: Giải PTVP $y' = \sin t$, $y(0) = 1$, $0 \leq t \leq 2\pi$.
(Nghiệm chính xác là: $y = -\cos t + 2$)
- Chương trình Matlab

%Viết hàm tính giá trị của hàm vế phải:

```
function dydt = fp(t,y)
```

% Tính giá trị cho PTVP $y' = \sin t$

```
dydt = [sin(t)];
```

% Giải PTVP bằng hàm ODE23

```
[t,y] = ode23(@fp,[0,2*pi],[1]);
```

```
plot(t,y);
```

```
hold on
```

```
plot(t,-cos(t),'r')
```



25 YEARS ANNIVERSARY
SOICT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Thank you
for your
attentions!



soict.hust.edu.vn/



fb.com/groups/soict





ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

CHƯƠNG 7: CÁC PHƯƠNG PHÁP CỰC TIỂU HÓA KHÔNG RÀNG BUỘC

Vũ Văn Thiệu, Đinh Viết Sang, Nguyễn Khánh Phương

TÍNH TOÁN KHOA HỌC

Giới thiệu

- ① Nhắc lại một số khái niệm từ giải tích
- ② Bài toán qui hoạch phi tuyến không ràng buộc
- ③ Các phương pháp cực tiểu một biến
 - Hàm đơn cực trị
 - Phương pháp Fibonacci
 - Phương pháp lát cắt vàng
- ④ Các phương pháp số cực tiểu không ràng buộc
 - Các phương pháp gradient
 - Phương pháp Newton

Nhắc lại một số khái niệm từ giải tích

Không gian Euclid n-chiều

Ký hiệu \mathbb{R}^n - tập các vec tơ thực n-chiều

$$\mathbb{R}^n = \{x = (x_1, x_2, \dots, x_n)^T : x_i \in \mathbb{R}, i = 1, 2, \dots, n\}$$

trong đó \mathbb{R} là tập số thực. Trên đó ta xác định các phép toán

- Phép cộng hai vec tơ $u = (u_1, u_2, \dots, u_n)^T$ và $v = (v_1, v_2, \dots, v_n)^T$

$$u + v = (u_1 + v_1, u_2 + v_2, \dots, u_n + v_n)$$

- Phép nhân vec tơ với một số thực α

$$\alpha u = (\alpha u_1, \alpha u_2, \dots, \alpha u_n)^T$$

\mathbb{R}^n cùng các phép toán vừa định nghĩa lập thành một không gian tuyến tính. Các phần tử của \mathbb{R}^n đôi khi là các điểm.

Nhắc lại một số khái niệm từ giải tích

Không gian Euclid n-chiều (tiếp)

- Nếu ta đưa vào thêm khái niệm tích vô hướng của hai vec tơ $u, v \in \mathbb{R}^n$:

$$\langle u, v \rangle = \sum_{i=1}^n u_i \times v_i$$

thì \mathbb{R}^n cùng với tích vô hướng sẽ trở thành không gian Euclid n-chiều.

- Độ dài chuẩn của vec tơ $u \in \mathbb{R}^n$ là số

$$\|u\| = \langle u, u \rangle^{1/2} = \left(\sum_{i=1}^n u_i^2 \right)^{1/2}$$

Nhắc lại một số khái niệm từ giải tích

Không gian Euclid n-chiều (tiếp)

- Khoảng cách giữa hai điểm $u, v \in \mathbb{R}^n$

$$\rho(u, v) = \|u - v\| = \left(\sum_{i=1}^n (u_i - v_i)^2 \right)^{1/2}$$

- Đối với $u, v, w \in \mathbb{R}^n$ ta có bất đẳng thức tam giác

$$\|u - v\| \leq \|u - w\| + \|w - v\|$$

Nhắc lại một số khái niệm từ giải tích

Không gian Euclid n-chiều (tiếp)

- Giả sử $\{u^k, k = 1, 2, \dots\}$ dãy điểm trong \mathbb{R}^n , nghĩa là $u^k \in \mathbb{R}^n$, $k = 1, 2, \dots$, điểm v được gọi là điểm tới hạn của dãy $\{u^k\}$ nếu tìm được dãy con $\{u^{k(i)}\}$ hội tụ đến v .
- Dãy $\{u^k\}$ được gọi là bị chặn nếu tìm được hằng số $M \geq 0$ sao cho $\|u^k\| \leq M$, với mọi $k = 1, 2, \dots$
- Tập $O(x, \epsilon) = \{u \in \mathbb{R}^n : \|u - x\| < \epsilon\}$ là khối cầu tâm tại x và bán kính $\epsilon > 0$ được gọi là **lân cận** ϵ của x .
- Điểm $v \in \mathbb{R}^n$ được gọi là **điểm tới hạn** của tập $U \subset \mathbb{R}^n$, nếu mọi lân cận ϵ của nó luôn chứa điểm của U khác với v .

Nhắc lại một số khái niệm từ giải tích

Không gian Euclid n-chiều (tiếp)

- Điểm $x \in X$ được gọi là **điểm trong** của tập X nếu tồn tại một ϵ lân cận của nó nằm trọn trong X . Tập các điểm trong của X được ký hiệu là $int(X)$.
- Điểm $x \in X$ được gọi là **điểm biên** của tập X nếu trong mọi ϵ lân cận của nó có điểm những điểm thuộc X và không thuộc X . Tập các điểm trong của X được ký hiệu là $\partial(X)$.
- Tập X được gọi là **tập mở** nếu mỗi điểm $x \in X$ đều là điểm trong của X .
- Tập X trong không gian \mathbb{R}^n được gọi là **bị chặn** hay **giới nội**, nếu tìm được hằng số $L > 0$ sao cho $\|u\| \leq L$ với mọi $u \in X$.

Nhắc lại một số khái niệm từ giải tích

Không gian Euclid n-chiều (tiếp)

- Tập X trong không gian \mathbb{R}^n được gọi là **tập đóng** nếu nó chứa tất cả các điểm tới hạn.
- Giả sử $\{x^k\}$ là dãy điểm trong tập đóng X và $\lim_{k \rightarrow +\infty} x^k = \bar{x}$, khi đó $\bar{x} \in X$
- Tập X được gọi là **compact** nếu nó đóng và giới nội.
- Giả sử $\{x^k\}$ là dãy điểm trong tập compact X . Khi đó từ $\{x^k\}$ ta luôn có thể trích ra dãy con hội tụ $\{x^{k(i)}\}$ sao cho $\lim_{k(i) \rightarrow +\infty} x^{k(i)} = \bar{x}$, khi đó $\bar{x} \in X$

Không gian Euclid n-chiều



Nhắc lại một số khái niệm từ giải tích

Vi phân hàm nhiều biến

Định nghĩa 1 : Giả sử hàm f xác định tại lân cận $O(x, \epsilon)$ của điểm x . Ta nói hàm f là khả vi tại x nếu tìm được vec tơ $f'(x) \in \mathbb{R}^n$ sao cho số gia của hàm số tại x : $\Delta f(x) = f(x + \Delta x) - f(x), ||\Delta x|| \leq \epsilon$ có thể viết dưới dạng

$$\Delta f(x) = \langle f'(x), \Delta x \rangle + o(x, \Delta x)$$

trong đó $o(x, \Delta x)$ là vô cùng bé bậc cao hơn $||\Delta x||$, nghĩa là $\lim_{||\Delta x|| \rightarrow 0} \frac{o(x, \Delta x)}{||\Delta x||} = 0$.

Hàm $f'(x)$ được gọi là gradient của hàm f tại x và thường được ký hiệu là $\nabla f(x)$.

Nhắc lại một số khái niệm từ giải tích

Vì phân hàm nhiều biến (tiếp)

Định nghĩa 2 : Giả sử hàm f xác định tại lân cận $O(x, \epsilon)$ của điểm x . Ta nói hàm f là hai lần khả vi tại x nếu cùng với vec tơ $f'(x)$, tồn tại ma trận đối xứng $f''(x) \in \mathbb{R}^{n \times n}$ sao cho số gia của hàm số tại x có thể viết dưới dạng

$$\Delta f(x) = f(x + \Delta x) - f(x) = \langle f'(x), \Delta x \rangle + \frac{\langle f''(x)\Delta x, \Delta x \rangle}{2} + o(x, \Delta x)$$

trong đó $\lim_{\|\Delta x\| \rightarrow 0} \frac{o(x, \Delta x)}{\|\Delta x\|^2} = 0$.

Ma trận $f''(x)$ được gọi là ma trận đạo hàm cấp hai hay Hessian của hàm f tại x và đôi khi còn được ký hiệu là $\nabla^2 f(x)$

Nhắc lại một số khái niệm từ giải tích

Vi phân hàm nhiều biến (tiếp)

Định nghĩa 3 : Giả sử hàm f xác định trên tập mở X . Ta nói hàm f là khả vi liên tục trên tập X nếu f là khả vi tại mọi điểm x của X và

$$||f'(x + \Delta x) - f'(x)|| \rightarrow 0 \text{ khi } ||\Delta x|| \rightarrow 0, \quad \forall x, x + \Delta x \in X$$

Tập các hàm thỏa mãn tính chất này được ký hiệu là $C^1(X)$.

Định nghĩa 4 : Giả sử hàm f xác định trên tập mở X . Ta nói hàm f là hai lần khả vi liên tục trên tập X nếu f là hai lần khả vi tại mọi điểm x của X và

$$||f''(x + \Delta x) - f''(x)|| \rightarrow 0 \text{ khi } ||\Delta x|| \rightarrow 0, \quad \forall x, x + \Delta x \in X$$

Tập các hàm thỏa mãn tính chất này được ký hiệu là $C^2(X)$.

Nhắc lại một số khái niệm từ giải tích

Vi phân hàm nhiều biến (tiếp)

Công thức Taylor : Giả sử $f(x)$ là hai lần khả vi liên tục tại một ϵ lân cận nào đó của x^o , khi đó ta có

$$\begin{aligned}f(x) = & f(x^o) + \langle f'(x^o), x - x^o \rangle \\& + \frac{1}{2} \langle f''(x^o)(x - x^o), x - x^o \rangle + \alpha(x, x^o) \|x - x^o\|^2\end{aligned}$$

trong đó $\lim_{x \rightarrow x^o} \alpha(x, x^o) = 0$, sai số có thể được viết dưới dạng $o(\|x - x^o\|^2)$

Nhắc lại một số khái niệm từ giải tích

Vi phân hàm nhiều biến (tiếp)

Công thức số gia hữu hạn : Giả sử hàm f là khả vi liên tục trên tập mở S và x là một vec tơ nào đó trong S . Khi đó mọi vec tơ y thỏa mãn $x + y \in S$, luôn tìm được số $\alpha \in [0, 1]$ sao cho

$$f(x + y) - f(x) = \langle f'(x + \alpha y), y \rangle = \int_0^1 \langle f'(x + ty), y \rangle dt$$

Nếu f hai lần khả vi liên tục thì ta có công thức:

$$f(x + y) - f(x) = \langle f'(x), y \rangle + \frac{1}{2} \langle f''(x + \alpha y)y, y \rangle.$$

Vi phân hàm nhiều biến



Nhắc lại một số khái niệm từ giải tích

Bài toán cực trị hàm nhiều biến

Xét bài toán tối ưu

$$f(x) \rightarrow \min, x \in X$$

trong đó $X \subset \mathbb{R}^n$, còn f là hàm xác định trên X .

Định nghĩa 5 : Điểm $x^* \in X$ được gọi là điểm **cực tiểu toàn cục** của f trên X nếu $f(x^*) \leq f(x)$, $\forall x \in X$.

- Giá trị $f(x^*)$ là giá trị cực tiểu của f trên X và ta sẽ ký hiệu $\min\{f(x) : x \in X\}$
- Điểm $x^* \in X$ được gọi là điểm **cực tiểu địa phương** của f trên X nếu tìm được lân cận $O(x, \epsilon)$, $\epsilon > 0$ sao cho $f(x^*) \leq f(x)$, với $x \in O(x, \epsilon) \cap X$.

Nhắc lại một số khái niệm từ giải tích

Bài toán cực trị hàm nhiều biến (tiếp)

Định nghĩa 6 : Giả sử hàm f bị chặn trên X . Số f^* được gọi là cận dưới của f trên X nếu

- ① $f^* \leq f(x)$ với mọi $x \in X$
- ② Với mọi số $\epsilon > 0$ luôn tìm đc $u^\epsilon \in X$ sao cho $f(u^\epsilon) < f^* + \epsilon$

Khi đó ta ký hiệu : $\inf_{x \in X} f(x) = f^*$

Chú ý

Rõ ràng nếu hàm f đạt cực tiểu toàn cục trên X thì

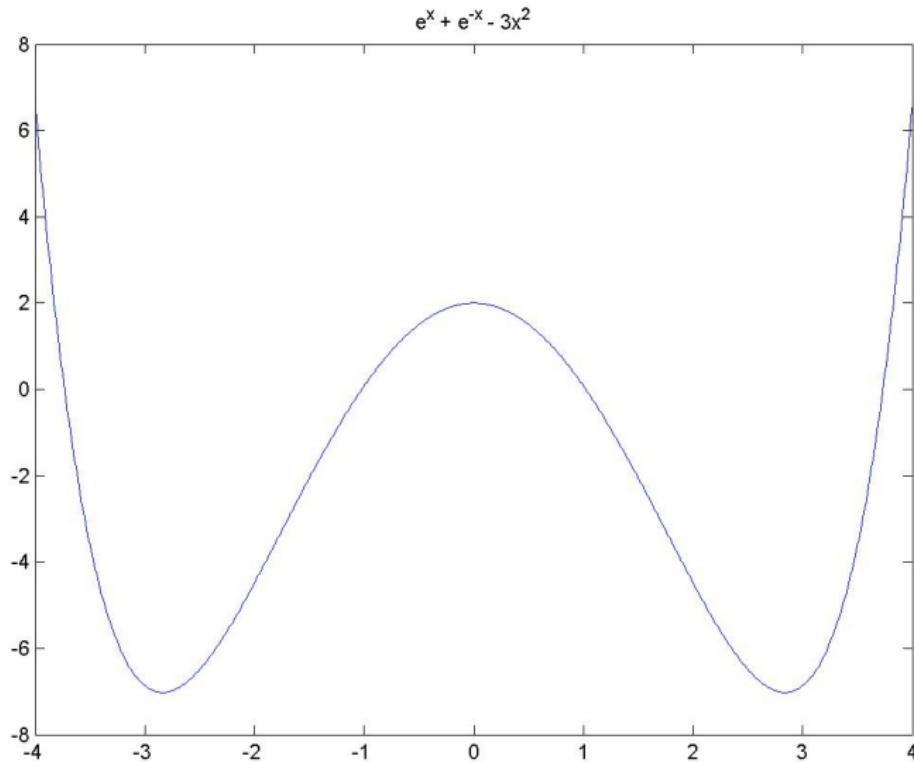
$$\inf_{x \in X} f(x) = \min_{x \in X} f(x)$$

Nhắc lại một số khái niệm từ giải tích

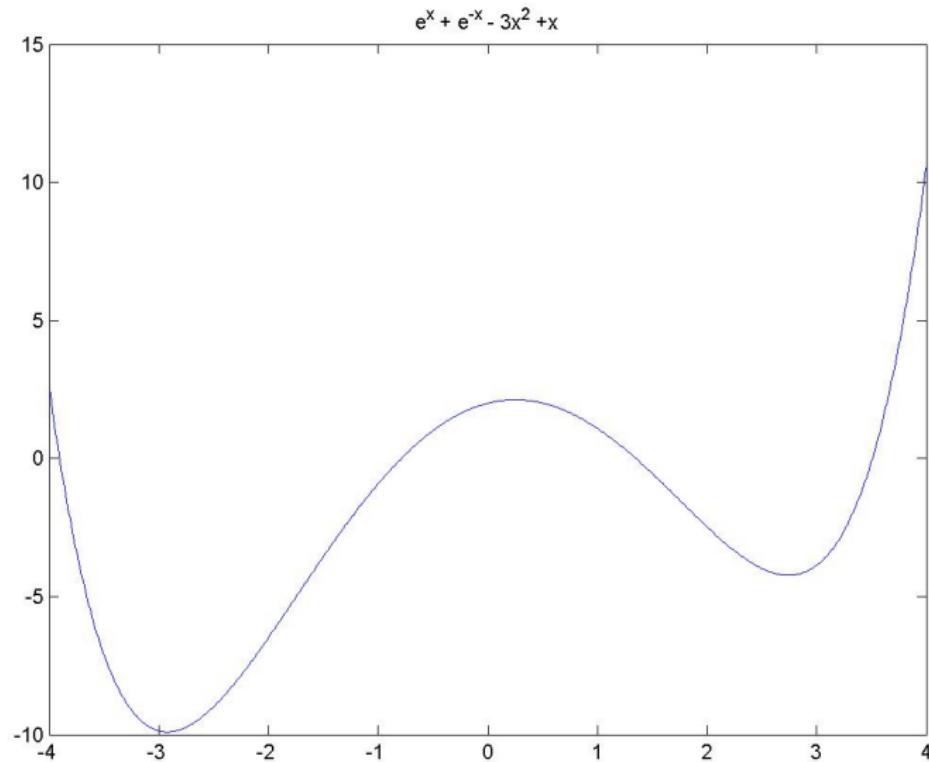
Một số ví dụ

- $f(x) = (x - 1)^2$ có cực tiểu toàn cục tại $x^* = 1$ với $f(x^*) = 0$.
- $f(x) = e^x + e^{-x} - 3x^2$. Giá trị tối ưu của hàm $f(x) = -7.02$. Bài toán có cực tiểu toàn cục tại hai điểm $x = \pm 2.84$, không có cực tiểu địa phương.
- $f(x) = e^{-x}$ cận dưới bằng không nhưng không đạt được. Không có cực tiểu địa phương cũng như cực tiểu toàn cục.
- $f(x) = -x + e^{-x}$ Hàm mục tiêu không bị chặn dưới, không có cực tiểu địa phương cũng như cực tiểu toàn cục.
- $f(x) = e^x + e^{-x} - 3x^2 + x$ Bài toán có hai cực tiểu địa phương $x_1 = -2.9226$ và $x_2 = 2.7418$, trong đó x_1 là cực tiểu toàn cục. Giá trị tối ưu của hàm là -9.9040

Nhắc lại một số khái niệm từ giải tích



Nhắc lại một số khái niệm từ giải tích



Bài toán cực trị hàm nhiều biến



Bài toán qui hoạch phi tuyến không ràng buộc

Định nghĩa

Xét bài toán qui hoạch phi tuyến không ràng buộc (unconstrained nonlinear programming)

$$\min\{f(x) : x \in \mathbb{R}^n\} \quad (1)$$

trong đó $f(x)$ khả vi liên tục.

Các định lý

Định lý 1 (Điều kiện cần tối ưu) : Điều kiện cần để x^0 là cực tiểu địa phương là

$$\nabla f(x^0) = 0 \quad (2)$$

Điều kiện (2) được gọi là điều kiện dừng, điểm x^0 thỏa mãn (2) đc gọi là điểm dừng. Như vậy việc giải bài toán (1) có thể qui về giải hệ phương trình (2).

Bài toán qui hoạch phi tuyến không ràng buộc

Các ví dụ

- $f(x) = x^2 - 3x - 1$ phương trình $f'(x) = 2x - 3 = 0$ có nghiệm duy nhất $x^0 = 3/2$ là điểm cực tiểu địa phương, đồng thời là điểm cực tiểu toàn cục.
- $f(x_1, x_2) = x_1^2 + x_2^2 - 2x_1x_2 + x_1$ phương trình
 $\nabla f(x) = \begin{pmatrix} 2x_1 - 2x_2 + 1 \\ -2x_1 + 2x_2 \end{pmatrix} = 0$ có nghiệm duy nhất
 $x^0 = (-1/4, 1/4)$. Tuy nhiên, nghiệm x^0 không là phương án tối ưu của bài toán $\min\{f(x) : x \in \mathbb{R}^2\}$ vì ta có
 $f(-1/4, 1/4) = -1/8 > -1 = f(0, 1)$.

Bài toán qui hoạch phi tuyến không ràng buộc

Các định lý (tiếp)

Định lý 2 (Điều kiện đủ tối ưu) : Giả sử f là hai lần khả vi liên tục. Điểm dừng x^0 là cực tiểu địa phương nếu ma trận $f''(x^0)$ là ma trận xác định dương.

Để biết ma trận có tính xác định dương hay không có thể sử dụng tiêu chuẩn Silvestra sau đây.

Tiêu chuẩn Silvestra : Ma trận $A = (a_{ij})_{n \times n}$ là xác định không âm (bán xác định dương) khi và chỉ khi tất cả các định thức con của nó là không âm

$$\Delta_{i_1, i_2, \dots, i_k} = \det \begin{vmatrix} a_{i_1, i_1} & a_{i_1, i_2} & \cdots & a_{i_1, i_k} \\ a_{i_2, i_1} & a_{i_2, i_2} & \cdots & a_{i_2, i_k} \\ \vdots & & & \\ a_{i_k, i_1} & a_{i_k, i_2} & \cdots & a_{i_k, i_k} \end{vmatrix} \geq 0$$

trong đó $\forall 1 \leq i_1 < i_2 < \cdots < i_k \leq n, \forall k = 1, 2, \dots, n$

Bài toán qui hoạch phi tuyến không ràng buộc

Các ví dụ

- Xét $f(x_1, x_2) = e^{x_1^2 + x_2^2}$ giải hệ phương trình

$$\nabla f(x) = \begin{pmatrix} 2x_1 e^{x_1^2 + x_2^2} \\ 2x_2 e^{x_1^2 + x_2^2} \end{pmatrix} = 0$$

có nghiệm duy nhất $x^0 = (0, 0)$ do $f''(0, 0) = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$ có định thức

$\det|f''(x^0)| > 0$ suy ra x^0 là điểm cực tiểu địa phương đồng thời là phương án tối ưu của bài toán.

Bài toán qui hoạch phi tuyến không ràng buộc

Các ví dụ (tiếp)

- Xét $f(x_1, x_2) = -x_1^2 + x_2^2 - 2x_1x_2 - x_1$ giải hệ phương trình

$$\nabla f(x) = \begin{pmatrix} -2x_1 - 2x_2 - 1 \\ -2x_1 + 2x_2 \end{pmatrix} = 0$$

có nghiệm duy nhất $x^0 = (-1/4, -1/4)$ do $f''(x^0) = \begin{pmatrix} -2 & -2 \\ -2 & 2 \end{pmatrix}$ có
định thức $\det|f''(x^0)| < 0$ vậy x^0 không là cực tiểu của hàm $f(x)$.

Bài toán qui hoạch phi tuyến không ràng buộc



Các phương pháp cực tiểu một biến

Hàm đơn cực trị

Hàm đơn cực trị (unimodal function) là hàm chỉ có một điểm cực đại hay cực tiểu trên đoạn xác định.

Định nghĩa : Hàm $f(x)$ được gọi là đơn cực tiểu nếu

- $x_1 < x_2 < x^*$ kéo theo $f(x_2) < f(x_1)$
- $x_2 > x_1 > x^*$ kéo theo $f(x_1) < f(x_2)$

trong đó x^* là điểm cực tiểu.

Chú ý

Các phương pháp tìm kiếm (sẽ được giới thiệu) đều sử dụng giả thuyết là hàm đơn cực trị.

Các phương pháp cực tiểu một biến

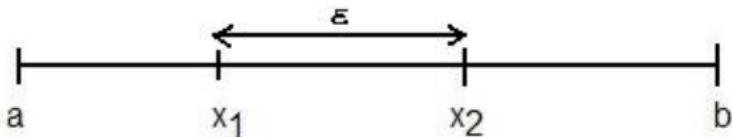
Ví dụ

Giả sử đoạn chứa cực tiểu $[0,1]$, ta tính giá trị hàm tại hai điểm $x_1 < x_2$: $f(x_1) = f_1, f(x_2) = f_2$ có 3 khả năng xảy ra :

- ① $f_1 < f_2$: Điểm cực tiểu x không thể nằm bên phải x_2 vì thế đoạn chứa cực tiểu mới là $[0, x_2]$
- ② $f_1 > f_2$: Điểm cực tiểu x không thể nằm bên trái x_1 vì thế đoạn chứa cực tiểu mới là $[x_1, 1]$
- ③ $f_1 = f_2$: Hai đoạn $[0, x_1]$ và $(x_2, 1]$ có thể loại bỏ và đoạn chứa cực tiểu mới là $[x_1, x_2]$

Các phương pháp cực tiểu một biến

Sơ đồ tìm kiếm



Giả sử ta có đoạn chứa cực tiểu xuất phát là $[a, b]$

- ① Tính $x_1 = a + (b - a)/2 - e/2$ và $x_2 = a + (b - a)/2 + e/2$ với e là sai số.
- ② Tính $f_1 = f(x_1)$ và $f_2 = f(x_2)$
- ③
 - ▶ Nếu $f_1 < f_2$ thì đặt $b = x_2$ (loại bỏ đoạn $x > x_2$);
 - ▶ Nếu $f_1 > f_2$ thì đặt $a = x_1$ (loại bỏ đoạn $x < x_1$);
 - ▶ Nếu $f_1 = f_2$ thì đặt $a = x_1, b = x_2$ (loại bỏ đoạn $x < x_1$ và $x > x_2$);
- ④ Nếu $|b - a| < 2e$ thì kết thúc; trái lại quay về bước 1

Các phương pháp cực tiểu một biến

Ví dụ :

Viết đoạn kịch bản tìm cực tiểu của $f(x) = x(x - 1.5)$ trên đoạn $(a, b) = (0, 1)$ với chênh lệch $e = 0.01$ so với nghiệm đúng $x^* = 0.75$

```
f = inline('x.*(x-1.5)','x');
eps = 0.01;
a = 0; b = 1; k = 0;
while abs(b-a)>= 2*eps
    x1=a + (b-a)/2 - eps/2; x2=a + (b-a)/2 + eps/2;
    f1=f(x1); f2=f(x2); k=k+1;
    if f1<f2 b=x2;
        elseif f1>f2 a=x1;
        else a=x1;b=x2;
    end
end
...
```

Các phương pháp cực tiểu một biến

Ví dụ (tiếp) :

```
...
fprintf('So buoc lap k= %d ',k);
fprintf('Do dai doan : b-a = %d',b-a);
fprintf('x= %d',x1);
```

Kết quả

» So buoc lap k=7

Do dai doan : b-a = 1.773438e-002

x=7.502344e-001

Các phương pháp cực tiểu một biến

Phương pháp Fibonacci

Định nghĩa : Dãy Fibonacci được định nghĩa đệ quy như sau

$$F_0 = 1; F_1 = 1;$$

$$F_k = F_{k-1} + F_{k-2}, k \geq 2;$$

Ta phải xác định số bước lặp N trước khi thực hiện tính cực tiểu. Việc chọn hai điểm $x_1^{(k)}$ và $x_2^{(k)}$ ở bước lặp k đc xác định theo công thức :

$$x_1^{(k)} = \frac{F_{N-1-k}}{F_{N+1-k}}(b_k - a_k) + a_k, k = 0, 1, \dots, N-1$$

$$x_2^{(k)} = \frac{F_{N-k}}{F_{N+1-k}}(b_k - a_k) + a_k, k = 0, 1, \dots, N-1$$

Các phương pháp cực tiểu một biến

Phương pháp lát cắt vàng

Để cải tiến phương pháp Fibonacci, khi không cần cho trước số bước lặp N , ta áp luôn tỉ lệ cố định khi phân chia khoảng $b_k - a_k$

$$\lim_{N \rightarrow \infty} \frac{F_{N-1}}{F_{N+1}} = 0.382; \quad \lim_{N \rightarrow \infty} \frac{F_N}{F_{N+1}} = 0.618$$

Do đó phương pháp lát cắt vàng đề xuất chọn hai điểm thử $x_1^{(k)}$ và $x_2^{(k)}$ theo công thức cập nhật tại bước lặp như sau

$$x_1^{(k)} = 0.382(b_k - a_k) + a_k,$$

$$x_2^{(k)} = 0.618(b_k - a_k) + a_k, \quad k = 0, 1, 2, \dots$$

Các phương pháp cực tiểu một biến

Ví dụ

Cài đặt thuật toán lát cắt vàng để tìm cực tiểu hàm trong đoạn [0,2]

$$f(x) = x^2 - 2x + 1$$

```
f = inline('x^2-2*x+1');
a = 0; b = 2; eps = 0.00001;
x1 = a + (b-a)*0.382;
x2 = a + (b-a)*0.618;
f1 = f(x1);
f2 = f(x2);
while abs(b-a)>2*eps
    if f1 > f2
        a = x1; x1 = x2; f1 = f2;
        x2 = a + (b-a)*0.618;
        f2 = f(x2);...
```

Các phương pháp cực tiểu một biến

Ví dụ (tiếp)

```
...
else
    b = x2; x2 = x1; f2 = f1;
    x1 = a + (b-a)*0.382;
    f1 = f(x1);
end
end
fprintf('Khoang co hep nghiem : [%f,%f]',a,b);
»
Khoang co hep nghiem : [0.999990,1.000004]
```

Các phương pháp cực tiểu một biến



Các phương pháp số cực tiểu không ràng buộc

Mở đầu

Xét bài toán qui hoạch phi tuyến không ràng buộc

$$f(x) \rightarrow \min, x \in \mathbb{R}^n \quad (3)$$

trong đó $f(x)$ là khả vi liên tục. Để giải (3), nếu tồn tại có thể tìm được trong số các nghiệm của phương trình

$$\nabla f(x) = 0 \quad (4)$$

Tuy vậy, việc giải hệ phương trình (4) trong trường hợp tổng quát cũng không kém phần phức tạp. Dẫn đến ta phải dùng các phương pháp hiệu quả để giải (3).

Các phương pháp số cực tiểu không ràng buộc

Mở đầu (tiếp)

Hướng thường dùng để giải quyết (3) là dùng các phương pháp lặp từ giá trị khởi tạo x^0 rồi dịch chuyển dần 'về hướng' giá trị tối ưu x^* , theo mỗi bước lặp cập nhật là :

$$x^{k+1} = x^k + \alpha_k p^k, \quad k = 1, 2, \dots \quad (5)$$

trong đó

- p^k là vec tơ định hướng dịch chuyển từ điểm x^k .
- α_k là độ dài của bước dịch chuyển theo hướng p^k .

Rõ ràng thủ tục (5) là xác định khi ta xác định được hướng dịch chuyển p^k và cách tính độ dài bước dịch chuyển α_k .

Các phương pháp số cực tiểu không ràng buộc

Mở đầu (tiếp)

Phụ thuộc vào các cách xây dựng p^k và α^k khác nhau mà ta có các thủ tục lặp với các đặc tính khác nhau. Ta đặc biệt quan tâm đến hai đặc tính sau :

- Sự thay đổi giá trị của hàm mục tiêu f của dãy $\{x^k\}$
- Sự hội tụ của dãy $\{x^k\}$ đến lời giải x^* .

Cũng cần chú ý là việc xác định p^k và α_k khác nhau cũng đòi hỏi khôi lượng tính toán khác nhau.

Các phương pháp số cực tiểu không ràng buộc

Các phương pháp gradient

Ta chọn hướng p^k sao cho

$$\langle \nabla f(x^k), p^k \rangle < 0 \quad (6)$$

Bởi vì khi chọn α^k đủ nhỏ, ta có

$$f(x^{k+1}) = f(x^k + \alpha_k) = f(x^k) + \alpha_k \langle \nabla f(x^k), p^k \rangle + o(\alpha_k) < f(x^k)$$

tức là dịch chuyển theo hướng p^k với độ dài đủ nhỏ ta sẽ đến được điểm x^{k+1} với giá trị hàm mục tiêu nhỏ hơn. Vậy hướng p^k thỏa mãn (6) được gọi là hướng giảm (hướng tụt) của hàm mục tiêu $f(x)$.

Các phương pháp số cực tiểu không ràng buộc

Các phương pháp gradient (tiếp)

Một trong các vec tơ thỏa mãn bất đẳng thức (6) có thể chọn là vec tơ đối gradient của hàm f tại x^k :

$$p_k = -\alpha_k \nabla f(x^k), \alpha_k > 0, k = 0, 1, 2, \dots$$

Khi đó ta có thủ tục lặp

$$x^{k+1} = x^k - \alpha_k \nabla f(x^k), \alpha_k > 0, k = 0, 1, 2, \dots \quad (7)$$

Thủ tục lặp tuân theo công thức (7) được gọi là **các phương pháp gradient**.

Các phương pháp số cực tiểu không ràng buộc

Các phương pháp gradient (tiếp)

Do hướng dịch chuyển là cố định, nên các phương pháp gradient khác nhau do cách chọn α_k . Ta liệt kê ra một số cách chọn cơ bản sau

- Thủ tục 1 : Giải bài toán cực tiểu hàm một biến

$$\min\{\varphi_k(\lambda) : \lambda \geq 0\}, \text{ với } \varphi_k(\lambda) = f(x^k - \lambda \nabla f(x^k))$$

Phương án tối ưu của bài toán được lấy làm giá trị của α_k .

- Thủ tục 2 : Để chọn α_k ta tiến hành theo thủ tục sau

- Đặt $\alpha = \bar{\alpha} > 0$
- Đặt $u = x^k - \alpha \nabla f(x^k)$, tính $f(u)$
- Kiểm tra

$$f(u) - f(x^k) \leq -\epsilon \alpha \|\nabla f(x^k)\|^2, \text{ với } 0 < \epsilon < 1 \quad (8)$$

- Nếu bđt (8) thỏa mãn thì đặt $\alpha_k = \alpha$, ngược lại đặt $\alpha = \alpha/2$ và quay lại bước 2.

Các phương pháp số cực tiểu không ràng buộc

Các định lý với phương pháp gradient

- Định lý 1 : Giả sử $f(x)$ bị chặn dưới và gradient của nó $f'(x)$ thỏa mãn điều kiện Lipchitz :

$$||f'(x) - f'(y)|| \leq L||x - y||$$

với mọi $x, y \in \mathbb{R}^n$, việc chọn α_k được tiến hành theo thủ tục 2. Khi đó theo công thức lặp (7) sẽ sinh ra dãy $\{x^k\}$ thỏa mãn điều kiện

$$||f'(x)|| \rightarrow 0, \text{ khi } k \rightarrow \infty$$

với mọi điểm xuất phát x^0

Các phương pháp số cực tiểu không ràng buộc

Các định lý với phương pháp gradient (tiếp)

Định lý 2 : Giả sử hàm f là hai lần khả vi liên tục và ma trận Hessian của nó thỏa mãn điều kiện

$$m\|y\|^2 \leq \langle H(x)y, y \rangle \leq M\|y\|^2, M \geq m > 0$$

với mọi $x, y \in \mathbb{R}^n$, dãy $\{x^k\}$ được xây dựng theo thủ tục lặp (7) với α_k được xác định theo thủ tục 2. Khi đó với mọi điểm xuất phát x^0 ta có

$$x^k \rightarrow x^*, f(x^k) \rightarrow f(x^*), \text{ khi } k \rightarrow \infty$$

trong đó x^* là điểm cực tiểu của $f(x)$, đồng thời ta có các đánh giá sau:

$$\begin{aligned} f(x^k) - f(x^*) &\leq q^k [f(x^0) - f(x^*)], \\ \|x^k - x^*\| &\leq Cq^{1/2} \end{aligned}$$

với $0 < C < +\inf, 0 < q < 1$ là các hằng số.

Các phương pháp số cực tiểu không ràng buộc

Ví dụ :

Xét bài toán cực tiểu hóa không ràng buộc

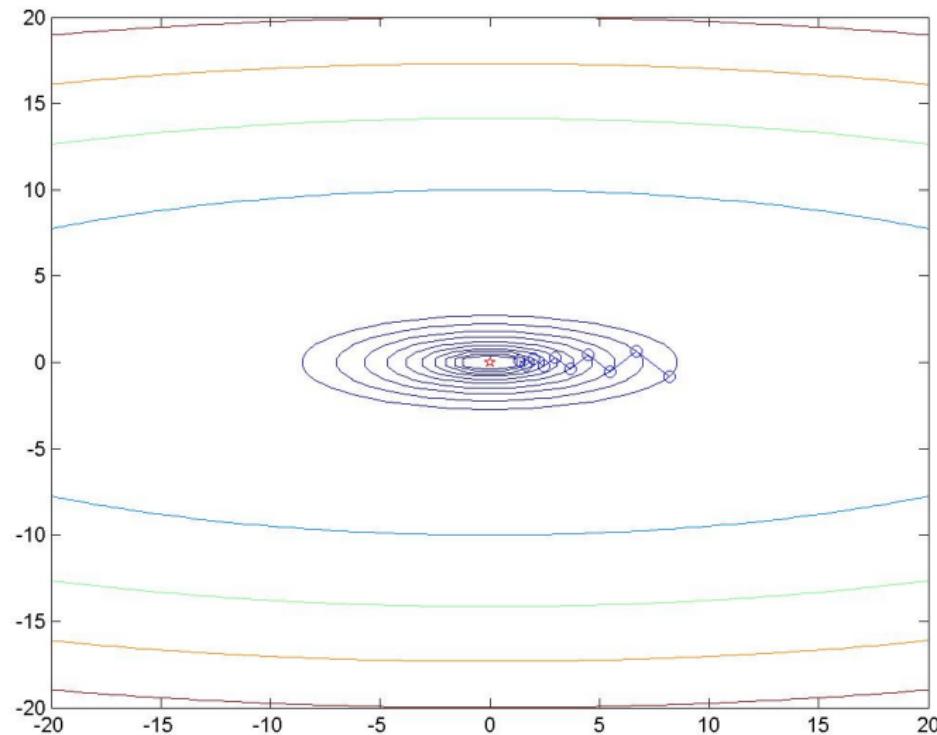
$$f(x_1, x_2) = \frac{1}{2}(x_1^2 + \gamma x_2^2) \quad (\gamma > 0)$$

Thực hiện thuật toán gradient bắt đầu từ phương án $x^0 = (\gamma, 1)$, độ dài bước xác định theo thủ tục 1, ta thu được dãy các phương án xấp xỉ $x^k = (x_1^k, x_2^k)$ trong đó

$$x_1^k = \gamma \left(\frac{\gamma - 1}{\gamma + 1} \right)^k, x_2^k = \left(-\frac{\gamma - 1}{\gamma + 1} \right)^k$$

Dãy xấp xỉ hội tụ chậm đến phương án tối ưu $x^* = (0, 0)$ khi $\gamma >> 1$ và $\gamma << 1$

Các phương pháp số cực tiểu không ràng buộc



Các phương pháp số cực tiểu không ràng buộc

Phương pháp Newton

Trong trường hợp hàm f là hai lần khả vi liên tục và việc tính toán $f'(x)$ và $f''(x)$ là không khó khăn ta có thể sử dụng đến số hạng bậc hai của khai triển Taylor.

$$f_k(x) \approx f(x^k) + \langle f'(x^k), x - x^k \rangle + \frac{1}{2} \langle H(x^k)(x - x^k), x - x^k \rangle \quad (9)$$

là xấp xỉ bậc hai của hàm f tại lân cận điểm x^k .

Các phương pháp số cực tiểu không ràng buộc

Phương pháp Newton (tiếp)

Dễ thấy là khi x^k rất gần với x^* thì $f(x^k)$ rất gần với 0 và vì thế số hạng bậc hai của (9) xấp xỉ sẽ cho ta thông tin chính xác hơn về sự biến thiên của hàm f trong lân cận của x^k .

Ta xác định vec tơ xấp xỉ phụ u^k từ điều kiện

$$f_k(u^k) = \min f_k(x) \quad (10)$$

và xây dựng phương án xấp xỉ tiếp theo

$$x^{k+1} = x^k + \alpha_k(u^k - x^k) \quad (11)$$

Vậy phụ thuộc các phương án chọn α_k mà ta có các phương pháp khác nhau. Nếu $\alpha_k = 1$, với mọi k , ta có phương pháp Newton.

Các phương pháp số cực tiểu không ràng buộc

Phương pháp Newton (tiếp)

Từ (11) ta có $x^{k+1} = u^k$ khi chọn $\alpha_k = 1$ thế vào điều kiện (10) trở thành

$$f_k(x^{k+1}) = \min f_k(x), k = 1, 2, \dots \quad (12)$$

Từ (12) ta suy ra x^{k+1} là điểm dừng của hàm $f_k(x)$, tức là

$$\nabla f_k(x) = \nabla f(x^k) + H(x^k)(x - x^k) = 0$$

Vậy nếu $H(x^k)$ không suy biến thì ta có công thức Newton sau đây

$$x^{k+1} = x^k - [H(x^k)]^{-1} \nabla f(x^k), k = 1, 2, \dots \quad (13)$$

Các phương pháp số cực tiểu không ràng buộc

Phương pháp Newton (tiếp)

Định lý hội tụ : Giả sử $H(x)$ là khả nghịch, $[H(x)]^{-1}$ bị chặn, $H(x)$ thỏa mãn điều kiện Liptchitz

$$\|H(x) - H(y)\| \leq L\|x - y\|, \forall x, y \in \mathbb{R}^n$$

Khi đó dãy $\{x^k\}$ xây dựng theo (13) sẽ thỏa mãn

$$\|x^{k+1} - x^*\| \leq c\|x^k - x^*\|^2$$

với x^* là nghiệm của phương trình $f'(x) = 0$. Vậy tốc độ hội tụ của phương pháp Newton là bình phương khoảng cách sau mỗi bước lặp.

Các phương pháp số cực tiểu không ràng buộc

Ví dụ

Minh họa giải thuật Newton cho hàm mục tiêu không ràng buộc sau

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

trong giải thuật có các hàm $f(x)$, $gf(x)$ và $hf(x)$.

```
f = inline('100*(x(2)-x(1)^2)^2 + (1-x(1))^1');
gf = inline('[100*2*(x(2)-x(1)^2)*(-2*x(1)) +
2*x(1)-2;200*(x(2)-x(1)^2)]');
hf = inline('[1200*x(1)^2 - 400*x(2)+ 2,
-400*x(1);-400*x(1),200]');
```

Các phương pháp số cực tiểu không ràng buộc

Ví dụ (tiếp)

```
fval = f(x);gval = gf(x);H = hf(x);ng = norm(gval);nf = 1;tol = 0.05;iter = 0; alpha = 1;
```

```
while ng >= tol
```

```
    iter = iter + 1;nf = 0;alpha = 1;p = -inv(H)*gval;pass = 0;
```

```
    while pass == 0
```

```
        ftest = f(x+alpha*p);
```

```
        nf = nf+1;
```

```
        if ftest <= fval + 0.01*alpha*gval'*p
```

```
            pass = 1;
```

```
            x = x+alpha*p;
```

```
            fval = ftest;
```

```
            gval = gf(x);
```

```
            H = hf(x);
```

```
    ...
```

Các phương pháp số cực tiểu không ràng buộc

Ví dụ (tiếp)

```
...
ng = norm(gval);
else
    alpha = alpha/2;
end
end
fprintf('%3i %3.2e %3.2e %3.2e %3.2e
%i',iter,fval,ng,norm(x-xstart),alpha,nf);
end
```

Các phương pháp số cực tiểu không ràng buộc

iter	f	$\ g\ $	$\ x-x^*\ $	alpha	nf
1	2.18e+000	4.64e+000	2.21e+000	1.00e+000	1
2	2.08e+000	1.10e+001	2.06e+000	6.25e-002	5
3	2.00e+000	1.51e+001	1.93e+000	2.50e-001	3
4	1.91e+000	1.63e+001	1.83e+000	5.00e-001	2
5	1.79e+000	1.68e+001	1.72e+000	1.00e+000	1
6	1.46e+000	7.79e+000	1.65e+000	1.00e+000	1
7	1.36e+000	8.04e+000	1.59e+000	5.00e-001	2
8	1.23e+000	8.23e+000	1.50e+000	1.00e+000	1
9	9.87e-001	3.70e+000	1.40e+000	1.00e+000	1
10	9.82e-001	1.06e+001	1.23e+000	1.00e+000	1
11	6.72e-001	1.17e+000	1.12e+000	1.00e+000	1
.....					
16	1.42e-001	4.59e+000	2.82e-001	1.00e+000	1
17	9.04e-002	4.41e-001	1.96e-001	1.00e+000	1
18	2.24e-002	2.13e+000	4.88e-002	1.00e+000	1
19	9.92e-003	2.86e-002	2.22e-002	1.00e+000	1

Các phương pháp số cực tiểu không ràng buộc

Các phương pháp tìm kiếm

Trong thực tế không phải lúc nào hàm f cũng trơn và khả vi hai lần. Ngoài ra, việc tính toán các đạo hàm này đòi hỏi tính toán khối lượng lớn nên không thích hợp. Vậy ta cần các phương pháp chỉ cần đòi hỏi tính giá trị hàm.

$$f(x) \rightarrow \min, x \in \mathbb{R}^n \quad (14)$$

Ký hiệu

$$e^i = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix}$$

là vec tơ chỉ gồm các số không và số 1 duy nhất ở hàng thứ i . Dương nhiên $e^i \in \mathbb{R}^n$, đây là vec tơ trục tọa độ đơn vị của không gian n -chiều.

Các phương pháp tìm kiếm

Giải thuật

Giả sử ta có x^0 là một xấp xỉ xuất phát. Chọn $\alpha_0 > 0$ là một thông số của thuật toán. Trong các bước lặp $k = 0, 1, 2, \dots$ tại các xấp xỉ x^k và $\alpha_k > 0$.
Đặt

$$p^k = e^{i_k}, \quad i_k = k - \left[\frac{k}{n} \right] n + 1 \quad (15)$$

trong đó $[k/n]$ là số nguyên lớn nhất không vượt quá k/n .

Tính $f(x^k + \alpha_k p^k)$ nếu

$$f(x^k + \alpha_k p^k) < f(x^k) \quad (16)$$

thỏa mãn thì gán

$$x^{k+1} = x^k + \alpha_k p^k, \quad \alpha_{k+1} = \alpha_k$$

rồi chuyển sang $k + 1$.

Các phương pháp tìm kiếm

Giải thuật (tiếp)

Nếu bđt (16) không đc thực hiện thì tính $f(x^k - \alpha_k p^k)$ và kiểm bất đẳng thức

$$f(x^k - \alpha_k p^k) < f(x^k) \quad (17)$$

thỏa mãn thì gán

$$x^{k+1} = x^k - \alpha_k p^k, \quad \alpha_{k+1} = \alpha_k$$

rồi chuyển sang $k + 1$.

Một bước lặp thành công nếu một trong hai bđt (16) và (17) thỏa mãn.
Ngược lại, ta xử lý như sau

$$x^{k+1} = x^k,$$

Các phương pháp tìm kiếm

Giải thuật (tiếp)

$$\alpha_{k+1} = \begin{cases} \lambda\alpha_k, & \text{nếu } i_k = n, x^k = x^{k-n+1} \\ \alpha_k, & \text{nếu } i_k \neq n, \text{ hay } x^k \neq x^{k-n+1} \end{cases}$$

trong đó $0 < \lambda < 1$ là thông số của thuật toán.

- Công thức (15) đảm bảo việc thực hiện chuyển đổi hướng dịch chuyển một cách có chu kỳ
- $p^0 = e^1, p^1 = e^2, \dots, p^{n-1} = e^n, p^n = e^1, \dots, p^{2n-1} = e^n, p^{2n} = e^1,$
- Khi độ dài $\alpha_{k+1} = \lambda\alpha_k$ cần thay đổi theo công thức trên chỉ khi sau n chu kỳ liên tiếp mà ta không thành công trong việc giảm giá trị hàm mục tiêu khi thỏa mãn một trong hai bđt (16) và (17).

Các phương pháp số cực tiểu không ràng buộc

Các phương pháp tìm kiếm (tiếp)

Định lý hội tụ : Giả sử f là hàm lồi khả vi liên tục trên \mathbb{R}^n , còn x^0 được chọn sao cho tập

$$M(x^0) = \{x : f(x) \leq f(x^0)\}$$

là giới nội. Khi đó dãy $\{x^k\}$ xây dựng theo phương pháp mô tả ở trên là dãy cực tiểu hóa của bài toán (14), nghĩa là

$$f(x^k) \rightarrow f^*, k \rightarrow \infty$$

trong đó f^* là giá trị tối ưu của bài toán (14).

Các phương pháp số cực tiểu không ràng buộc



Bài tập về nhà :

tìm hiểu các hàm tối ưu hóa trong Matlab

- FMINBND
- FMINUNC
- FMINSEARCH
- OPTIMSET
- INLINE

Xét bài toán cực tiểu hóa không điều kiện:

$$f(x) = e^{x_1}(4x_1^2 + 2x_2^2 + 4x_1x_2 + 2x_2 + 1)$$

Điểm xuất phát $x^0 = (-1, 1)$

Xét bài toán quy hoạch phi tuyến không ràng buộc:

$$f(x_1, x_2) = (3x_1 - 9)^2 + (4x_2 - 10)^2 \rightarrow \min, x = (x_1, x_2) \in \mathbb{R}^2.$$

Thực hiện giải thuật gradient giải bài toán đặt ra bắt đầu từ phương án xuất phát $x^0 = (1, 2)$. Để xác định độ dài bước hãy sử dụng thủ tục 1 (trong đó đòi hỏi giải bài toán cực tiểu hóa một chiều)

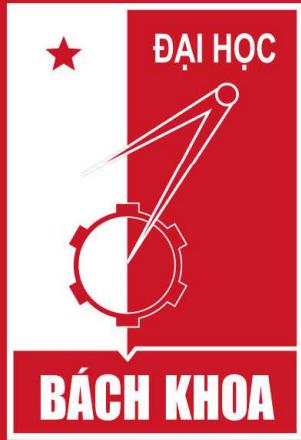


25
SOLCT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

**Thank you for
your attentions!**





25 YEARS ANNIVERSARY
SCHOOL OF COMPUTER SCIENCE AND TECHNOLOGY

ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG



ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Chương 8

Quy hoạch tuyến tính

Vũ Văn Thiệu, Đinh Việt Sang, Nguyễn Khánh Phương

TÍNH TOÁN KHOA HỌC

Nội dung

1) Thuật toán đơn hình

1. Bài toán QHTT dạng chính tắc và dạng chuẩn
2. Phương án cơ sở chấp nhận được
3. Công thức số gia hàm mục tiêu. Tiêu chuẩn tối ưu
4. Thuật toán đơn hình dạng ma trận nghịch đảo
5. Thuật toán đơn hình dạng bảng
6. Tính hữu hạn của thuật toán đơn hình
7. Thuật toán đơn hình hai pha

2) Lý thuyết đối ngẫu

1. Xây dựng bài toán đối ngẫu
2. Các định lý đối ngẫu
3. Một số ứng dụng của lý thuyết đối ngẫu

THUẬT TOÁN ĐƠN HÌNH

*Thuật toán đơn hình là thuật toán cơ bản giải bài toán
Quy hoạch tuyến tính*

Nội dung

1. Bài toán QHTT dạng chính tắc và dạng chuẩn
2. Phương án cơ sở chấp nhận được
3. Công thức số gia hàm mục tiêu. Tiêu chuẩn tối ưu
4. Thuật toán đơn hình dạng ma trận nghịch đảo
5. Thuật toán đơn hình dạng bảng
6. Tính hữu hạn của thuật toán đơn hình
7. Thuật toán đơn hình hai pha

Bài toán QHTT dạng chính tắc và dạng chuẩn

Bài toán QHTT tổng quát

- Bài toán QHTT tổng quát là bài toán tối ưu hoá mà trong đó chúng ta phải tìm **cực đại** (**cực tiểu**) của hàm mục tiêu tuyến tính với điều kiện biến số phải thoả mãn hệ thống phương trình và bất phương trình tuyến tính. Mô hình toán học của bài toán có thể phát biểu như sau

$$f(x_1, x_2, \dots, x_n) = \sum_{j=1}^n c_j x_j \rightarrow \min(\max), \quad (1)$$

với điều kiện

$$\sum_{j=1}^n a_{ij} x_j = b_i, i = 1, 2, \dots, p \quad (p \leq m) \quad (2)$$

$$\sum_{j=1}^n a_{ij} x_j \geq b_i, i = p+1, p+2, \dots, m \quad (3)$$

$$x_j \geq 0, j = 1, 2, \dots, q \quad (q \leq n) \quad (4)$$

$$x_j < 0, j = q+1, q+2, \dots, n \quad (5)$$

- Ký hiệu $x_j < 0$ để chỉ ra rằng biến x_j không có đòi hỏi trực tiếp về dấu.

Bài toán QHTT tổng quát

- Ràng buộc:

$$\sum_{j=1}^n a_{ij}x_j = b_i, i = 1, \dots, p$$

được gọi là ràng buộc cơ bản dạng đẳng thức.

- Ràng buộc:

$$\sum_{j=1}^n a_{ij}x_j \geq b_i, i = p+1, \dots, m$$

được gọi là ràng buộc cơ bản dạng bất đẳng thức.

- Ràng buộc:

$$x_j \geq 0, j = 1, \dots, q$$

được gọi là ràng buộc về dấu của biến số.

Bài toán QHTT dạng chính tắc

- Ta gọi bài toán QHTT dạng chính tắc là bài toán sau:

$$f(x_1, x_2, \dots, x_n) = \sum_{j=1}^n c_j x_j \rightarrow \min,$$

$$\sum_{j=1}^n a_{ij} x_j = b_i, i = 1, 2, \dots, m$$

$$x_j \geq 0, j = 1, 2, \dots, n$$

Bài toán QHTT dạng chuẩn

- Ta gọi bài toán QHTT dạng chuẩn là bài toán sau:

$$f(x_1, x_2, \dots, x_n) = \sum_{j=1}^n c_j x_j \rightarrow \min,$$

$$\sum_{j=1}^n a_{ij} x_j \geq b_i, i = 1, 2, \dots, m$$

$$x_j \geq 0, j = 1, 2, \dots, n$$

Đưa BT QHTT tổng quát về dạng chính tắc

- Rõ ràng Bài toán QHTT dạng chính tắc là trường hợp riêng của QHTT tổng quát.

$$f(x_1, x_2, \dots, x_n) = \sum_{j=1}^n c_j x_j \rightarrow \min(\max), \quad (1)$$

$$\sum_{j=1}^n a_{ij} x_j = b_i, i = 1, 2, \dots, p \quad (p \leq m) \quad (2)$$

$$\sum_{j=1}^n a_{ij} x_j \geq b_i, i = p+1, p+2, \dots, m \quad (3)$$

$$x_j \geq 0, j = 1, 2, \dots, q \quad (q \leq n) \quad (4)$$

$$x_j < 0, j = q+1, q+2, \dots, n \quad (5)$$

$$f(x_1, x_2, \dots, x_n) = \sum_{j=1}^n c_j x_j \rightarrow \min, \quad (2)$$

$$\sum_{j=1}^n a_{ij} x_j = b_i, i = 1, 2, \dots, m$$

$$x_j \geq 0, j = 1, 2, \dots, n$$

- Mặt khác, một bài toán QHTT bất kỳ luôn có thể đưa về dạng chính tắc nhờ các phép biến đổi sau:

Đưa BT QHTT tổng quát về dạng chính tắc

a) *Đưa ràng buộc bất đẳng thức dạng “≤” về dạng “≥”.* Bất phương trình tuyến tính

$$\sum_{j=1}^n a_{ij}x_j \leq b_i$$

là tương đương với bất phương trình tuyến tính sau

$$-\sum_{j=1}^n a_{ij}x_j \geq -b_i$$

Đưa BT QHTT tổng quát về dạng chính tắc

b) *Đưa ràng buộc dạng “=” về dạng “≥”.* Phương trình tuyến tính

$$\sum_{j=1}^n a_{ij} x_j = b_i$$

là tương đương với hệ gồm 2 bất phương trình tuyến tính sau

$$\sum_{j=1}^n a_{ij} x_j \geq b_i$$

$$-\sum_{j=1}^n a_{ij} x_j \geq -b_i$$

Đưa BT QHTT tổng quát về dạng chính tắc

c) *Đưa ràng buộc dạng “ \geq ” về dạng “ $=$ ”.* Bất phương trình tuyến tính

$$\sum_{j=1}^n a_{ij}x_j \geq b_i$$

là “tương đương” với hệ gồm 1 phương trình tuyến tính và một điều kiện không âm đối với biến số sau đây

$$\sum_{j=1}^n a_{ij}x_j - y_i = b_i$$

$$y_i \geq 0$$

- Tương đương hiểu theo nghĩa: Nếu $(x_1, x_2, \dots, x_n, y_i)$ là nghiệm của hệ thì (x_1, x_2, \dots, x_n) là nghiệm của bất phương trình.
- Biến y_i được gọi là **biến bù** (hay biến phụ).

Đưa BT QHTT tổng quát về dạng chính tắc

d) *Thay mỗi biến không có điều kiện dấu x_j bởi hiệu hai biến có điều kiện về dấu:*

$$x_j = x_j^+ - x_j^-,$$

$$x_j^+ \geq 0, x_j^- \geq 0.$$

e) *Đưa bài toán tìm cực đại về bài toán tìm cực tiểu.* Bài toán tối ưu hoá

$$\max \{f(x): x \in D\}$$

là tương đương với bài toán tối ưu hoá

$$\min \{-f(x): x \in D\}$$

theo nghĩa: Lời giải của bài toán này cũng là lời giải của bài toán kia và ngược lại, đồng thời ta có đẳng thức:

$$\max \{f(x): x \in D\} = - \min \{-f(x): x \in D\}$$

Đưa BT QHTT tổng quát về dạng chính tắc

f) *Đưa ràng buộc dạng “≤” về dạng “=”*. Bất phương trình tuyến tính

$$\sum_{j=1}^n a_{ij}x_j \leq b_i$$

là “tương đương” với hệ gồm 1 phương trình tuyến tính và một điều kiện không âm đối với biến số sau đây

$$\begin{aligned} \sum_{j=1}^n a_{ij}x_j + y_i &= b_i \\ y_i &\geq 0 \end{aligned}$$

- Tương đương hiểu theo nghĩa: Nếu $(x_1, x_2, \dots, x_n, y_i)$ là nghiệm của hệ thì (x_1, x_2, \dots, x_n) là nghiệm của bất phương trình.
- Biến y_i được gọi là **biến bù** (hay biến phụ).

Ví dụ:

- Bài toán QHTT

$$x_1 + 2x_2 - 3x_3 + 4x_4 \rightarrow \max,$$

$$x_1 + 5x_2 + 4x_3 + 6x_4 \leq 15,$$

$$x_1 + 2x_2 - 3x_3 + 3x_4 = 9,$$

$$x_1, x_2, x_4 \geq 0, x_3 < > 0,$$

là tương đương với bài toán QHTT dạng chính tắc:

$$-x_1 - 2x_2 + 3(x_3^+ - x_3^-) - 4x_4 \rightarrow \min,$$

$$x_1 + 5x_2 + 4(x_3^+ - x_3^-) + 6x_4 + x_5 = 15,$$

$$x_1 + 2x_2 - 3(x_3^+ - x_3^-) + 3x_4 = 9,$$

$$x_1, x_2, x_3^+, x_3^-, x_4, x_5 \geq 0,$$

tức là nếu $(\bar{x}_1, \bar{x}_2, \bar{x}_3^+, \bar{x}_3^-, \bar{x}_4, \bar{x}_5)$ là phương án tối ưu của bài toán dạng chính tắc thì $(\bar{x}_1, \bar{x}_2, \bar{x}_3, \bar{x}_4)$ với $\bar{x}_3 = \bar{x}_3^+ - \bar{x}_3^-$ sẽ là phương án tối ưu của bài toán xuất phát.

Giải bài toán QHTT trong mặt phẳng

Giải bài toán QHTT trong mặt phẳng²

- Bài toán QHTT dạng chuẩn 2 biến số

$$f(x_1, x_2) = c_1x_1 + c_2x_2 \rightarrow \min,$$

với điều kiện

$$a_{i1}x_1 + a_{i2}x_2 \geq b_i, i = 1, 2, \dots, m$$

- Ký hiệu

$$D = \{(x_1, x_2) : a_{i1}x_1 + a_{i2}x_2 \geq b_i, i = 1, 2, \dots, m\}$$

là miền ràng buộc.

Giải bài toán QHTT trong mặt phẳng

- Từ ý nghĩa hình học, mỗi bất phương trình tuyến tính

$$a_{i1} x_1 + a_{i2} x_2 \geq b_i, i = 1, 2, \dots, m$$

xác định một nửa mặt phẳng.

⇒ Miền ràng buộc D xác định như giao của m nửa mặt phẳng sẽ là một đa giác lồi trên mặt phẳng.

Giải bài toán QHTT trong mặt phẳng

- Phương trình

$$c_1x_1 + c_2x_2 = \alpha$$

có vectơ pháp tuyến là (c_1, c_2)

khi α thay đổi sẽ xác định các đường thẳng song song với nhau mà ta sẽ gọi là các đường mức (với giá trị mức α).

Mỗi điểm $u=(u_1, u_2) \in D$ sẽ nằm trên đường mức với mức

$$\alpha_u = c_1u_1 + c_2u_2 = f(u_1, u_2)$$

Giải bài toán QHTT trong mặt phẳng

Bài toán đặt ra có thể phát biểu dưới ngôn ngữ hình học như sau: Trong số các đường mức cắt tập D , hãy tìm đường mức với giá trị mức nhỏ nhất.

Bây giờ, có thể nhận thấy là, nếu dịch chuyển song song các đường mức theo hướng vectơ pháp tuyến của chúng $c = (c_1, c_2)$ thì giá trị mức sẽ tăng, nếu dịch chuyển theo hướng ngược lại thì giá trị mức sẽ giảm. Vì vậy, để giải bài toán đặt ra ta có thể tiến hành như sau: Bắt đầu từ một đường mức cắt D ta dịch chuyển song song các đường mức theo hướng ngược với hướng vectơ $c = (c_1, c_2)$ cho đến khi nào việc dịch chuyển tiếp theo làm cho đường mức không còn cắt D nữa thì dừng. Các điểm của D nằm trên đường mức cuối cùng này sẽ là các lời giải cần tìm, còn giá trị của nó chính là giá trị tối ưu của bài toán.

Ví dụ 1

- Giải bài toán QHTT sau:

$$x_1 - x_2 \rightarrow \min$$

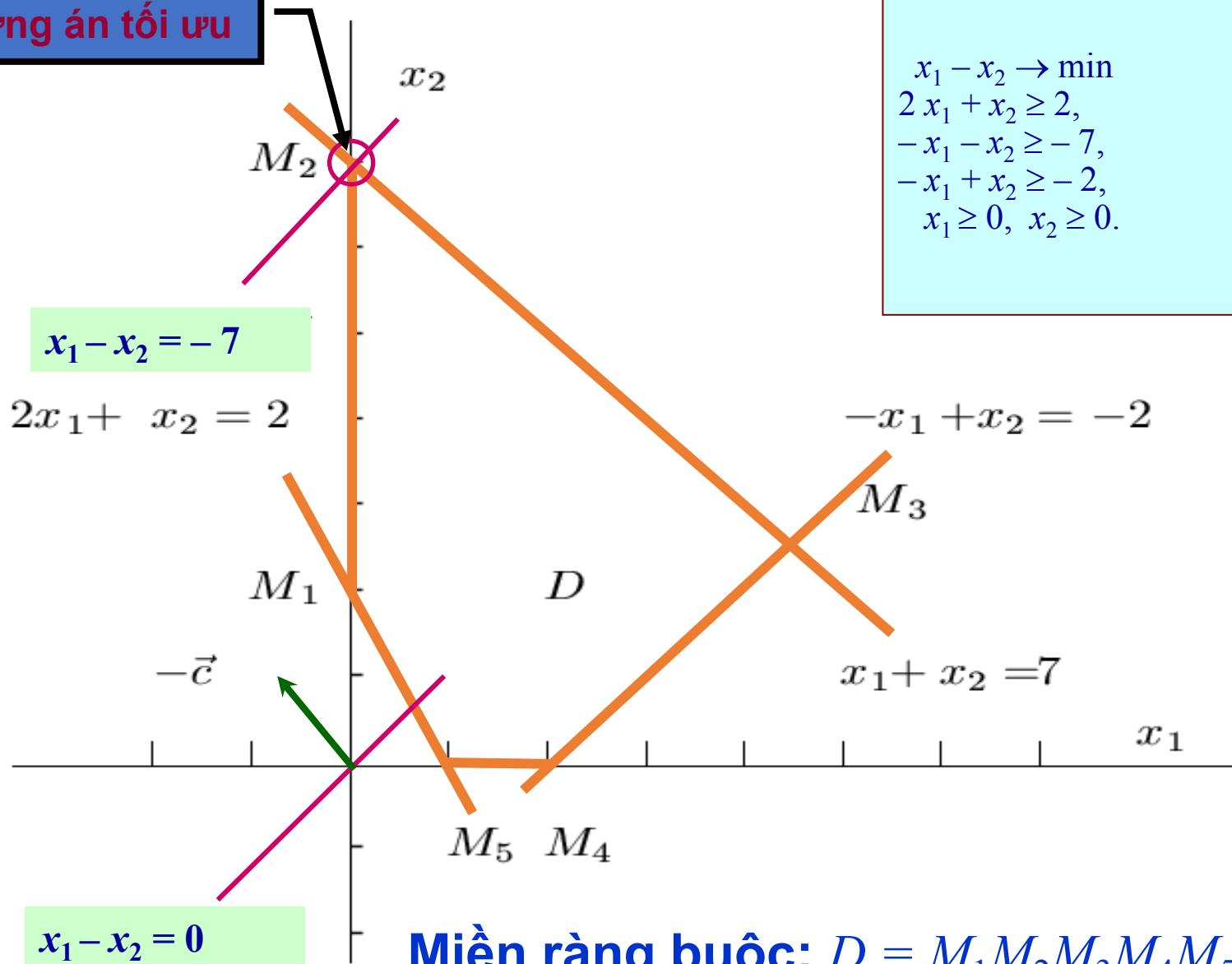
$$2x_1 + x_2 \geq 2,$$

$$-x_1 - x_2 \geq -7,$$

$$-x_1 + x_2 \geq -2,$$

$$x_1 \geq 0, \quad x_2 \geq 0.$$

Phương án tối ưu



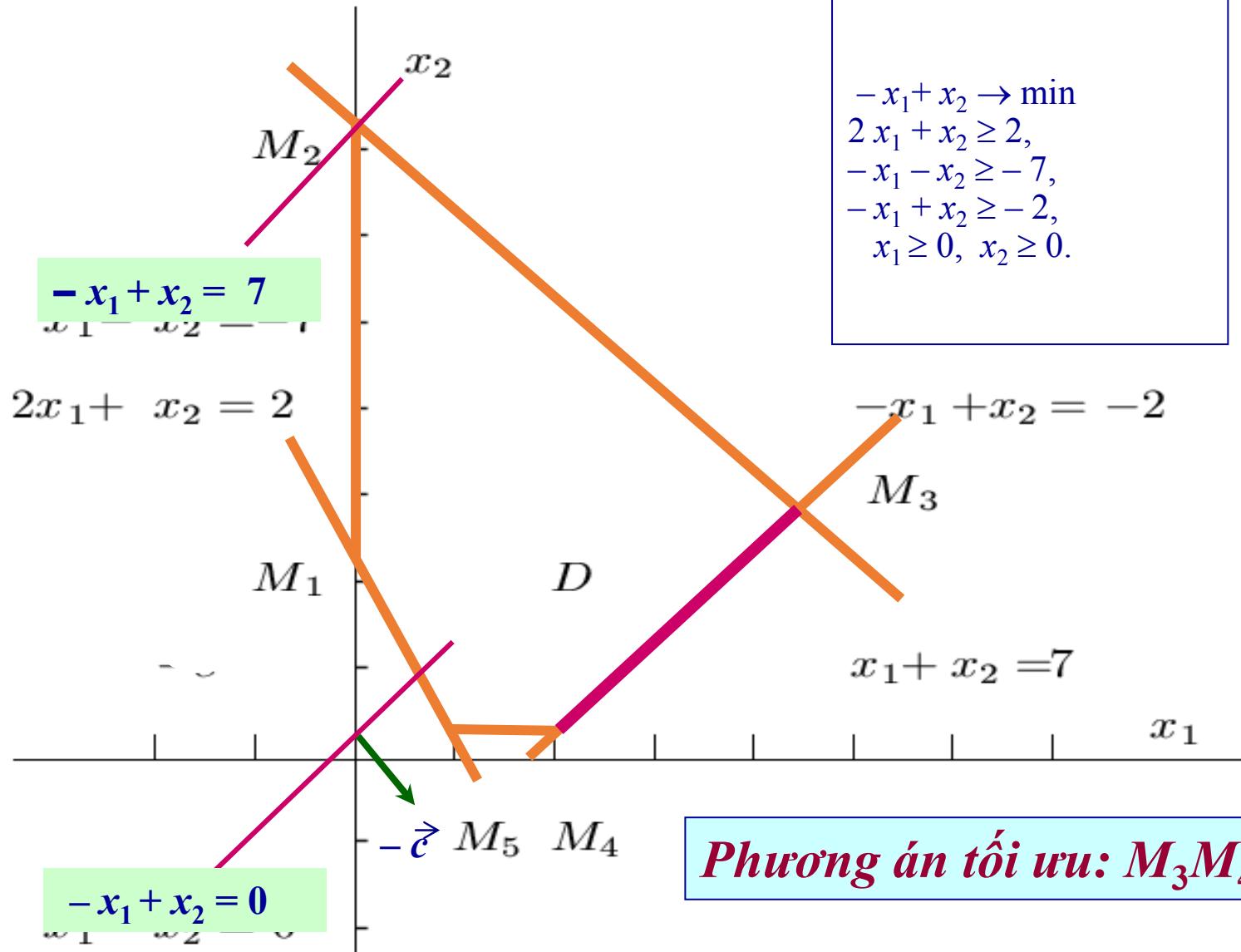
$$\begin{aligned}x_1 - x_2 &\rightarrow \min \\2x_1 + x_2 &\geq 2, \\-x_1 + x_2 &\geq -7, \\-x_1 + x_2 &\geq -2, \\x_1 &\geq 0, x_2 \geq 0.\end{aligned}$$

Ví dụ 1

- Giải theo phương pháp hình học vừa mô tả ta thu được lời giải tối ưu của bài toán tương ứng với điểm $M_2(0,7)$: $x^* = (0,7)$, với giá trị tối ưu là $f_* = -7$.
- Nếu thay hàm mục tiêu của bài toán bởi

$$-x_1 + x_2 \rightarrow \min,$$

thì giá trị tối ưu sẽ là -2 và tất cả các điểm nằm trên đoạn M_3M_4 đều là phương án tối ưu của bài toán. Chẳng hạn, có thể lấy phương án tối ưu của bài toán là $x^* = (2,0)$ (tương ứng với điểm M_4).



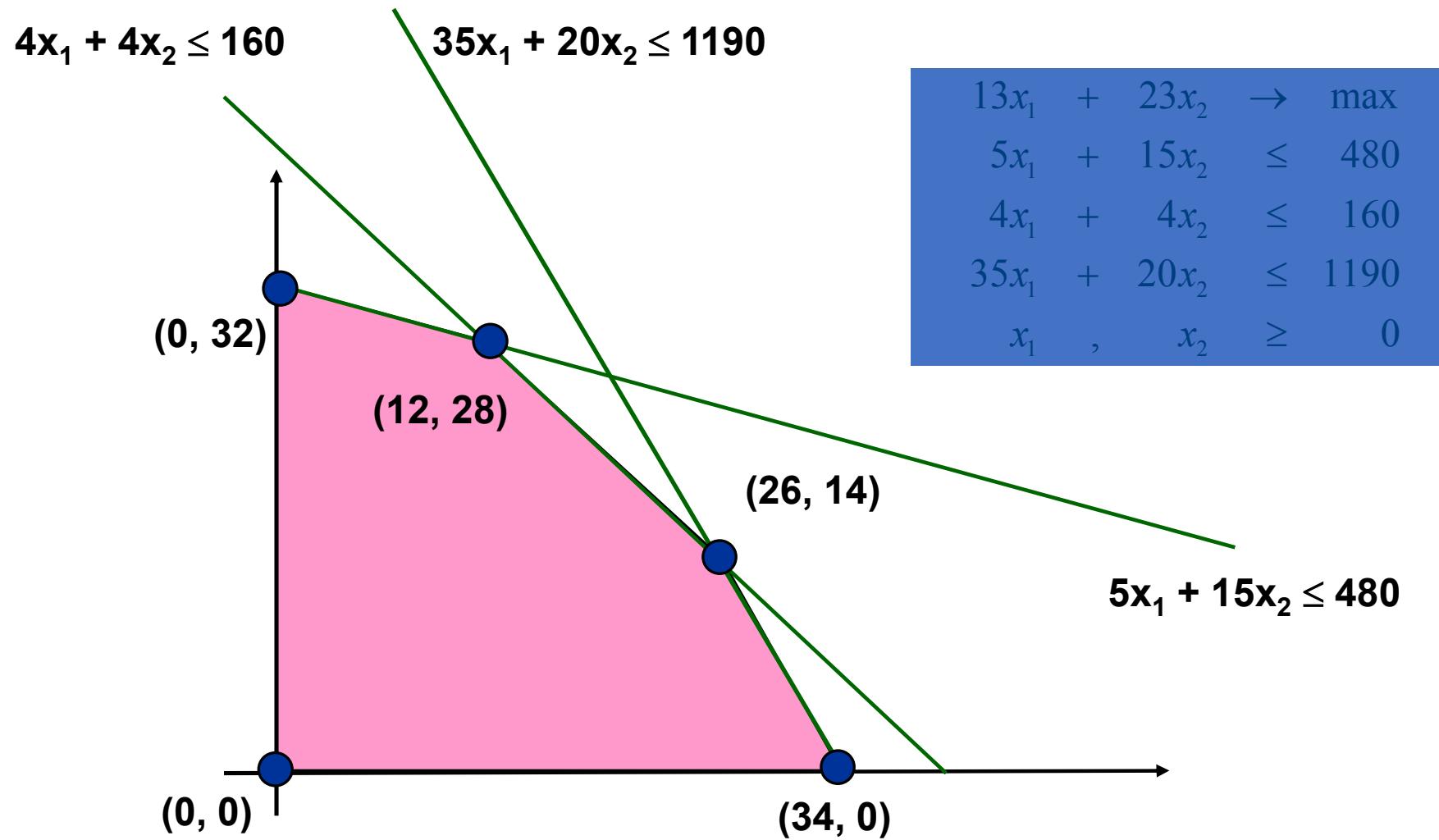
Nhận xét

- Trong cả hai trường hợp ta luôn tìm được phương án tối ưu là một đỉnh nào đó của miền ràng buộc.
- “*Bài toán QHTT trong mặt phẳng luôn có phương án tối ưu là đỉnh của miền ràng buộc.*”
- Nhận xét hình học quan trọng này đã dẫn tới việc đề xuất thuật toán đơn hình để giải bài toán QHTT.

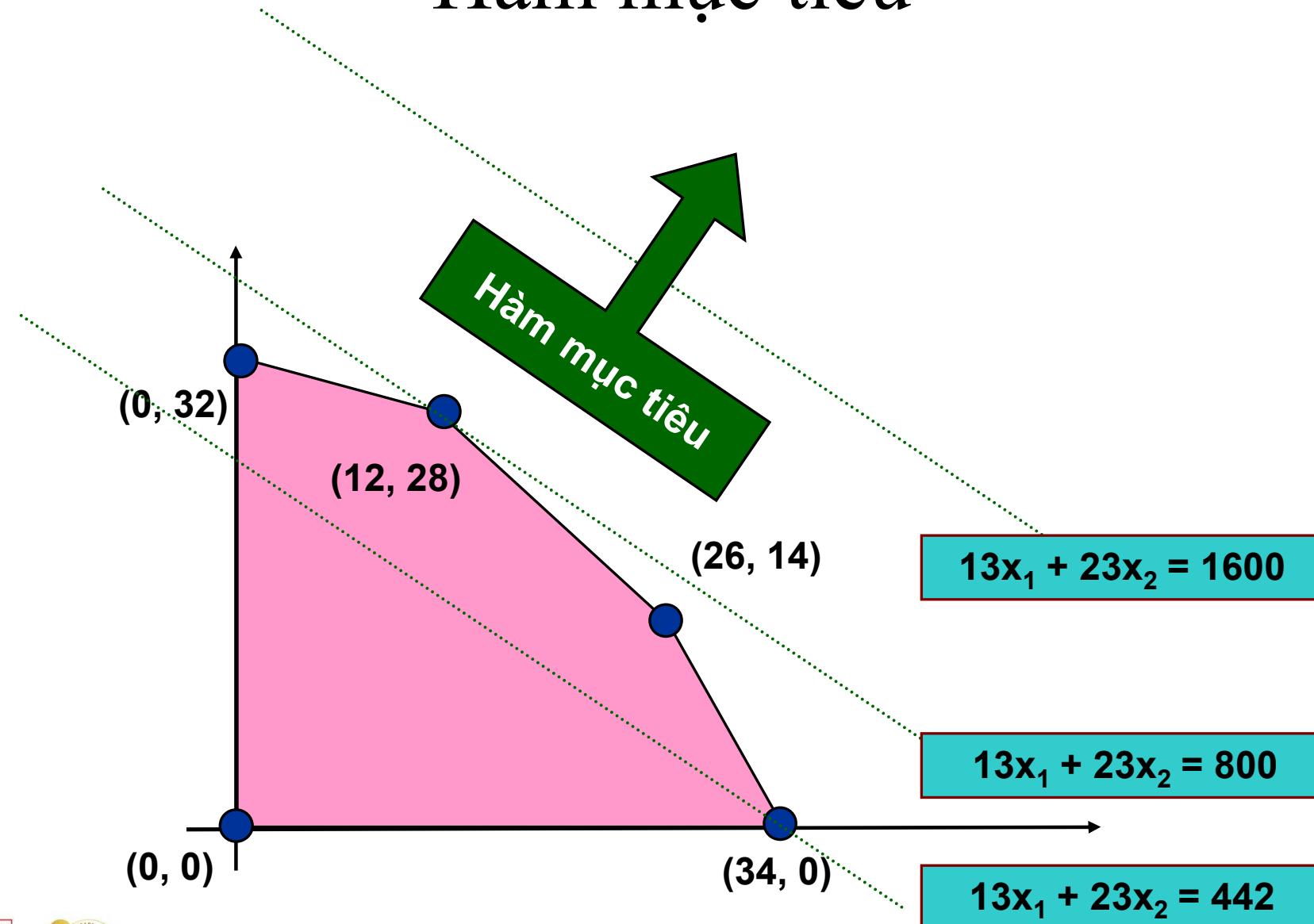
Ví dụ 2

$$\begin{aligned} 13x_1 + 23x_2 &\rightarrow \max \\ 5x_1 + 15x_2 &\leq 480 \\ 4x_1 + 4x_2 &\leq 160 \\ 35x_1 + 20x_2 &\leq 1190 \\ x_1, x_2 &\geq 0 \end{aligned}$$

Miền ràng buộc

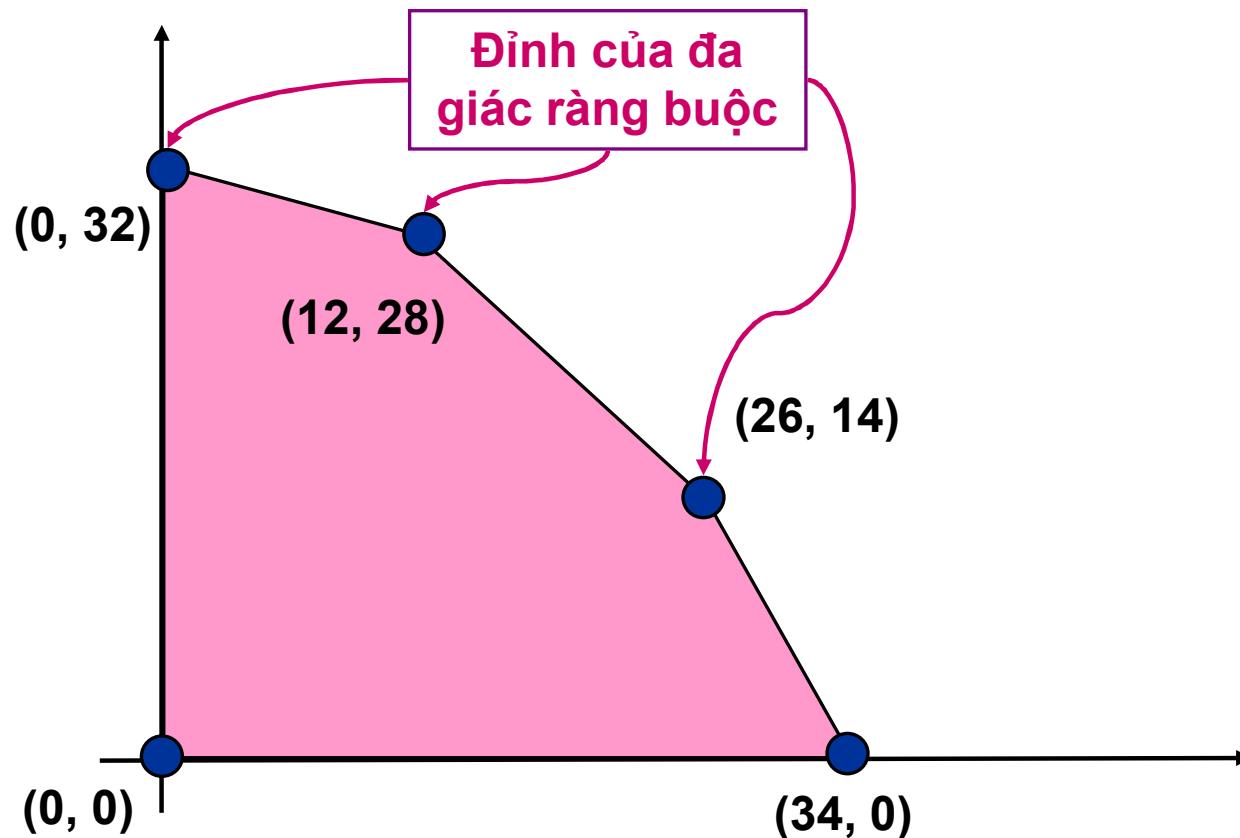


Hàm mục tiêu



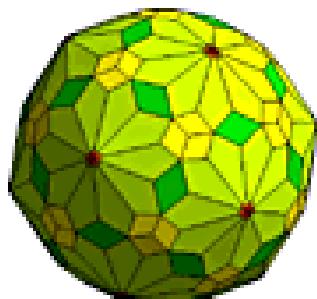
Ý nghĩa hình học

- Tồn tại lời giải tối ưu là đỉnh của đa giác ràng buộc

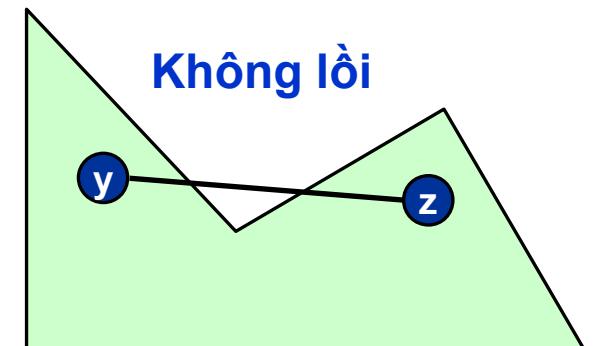
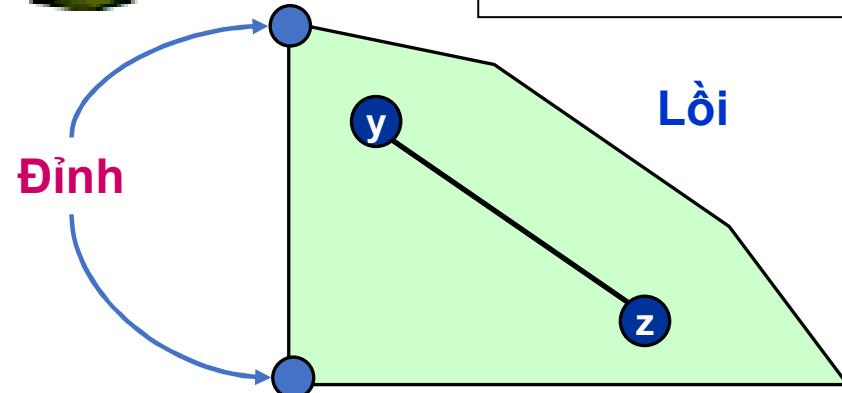


Ý nghĩa hình học của QHTT

- Miền ràng buộc là tập lồi đa diện.
 - Lồi:** nếu y và z là pacnd, thì $\alpha y + (1-\alpha)z$ cũng là pacnd với mọi $0 \leq \alpha \leq 1$.
 - Đỉnh:** pacnd x mà không thể biểu diễn dưới dạng $\alpha y + (1-\alpha)z$, $0 < \alpha < 1$, với mọi cặp pacnd phân biệt y và z .



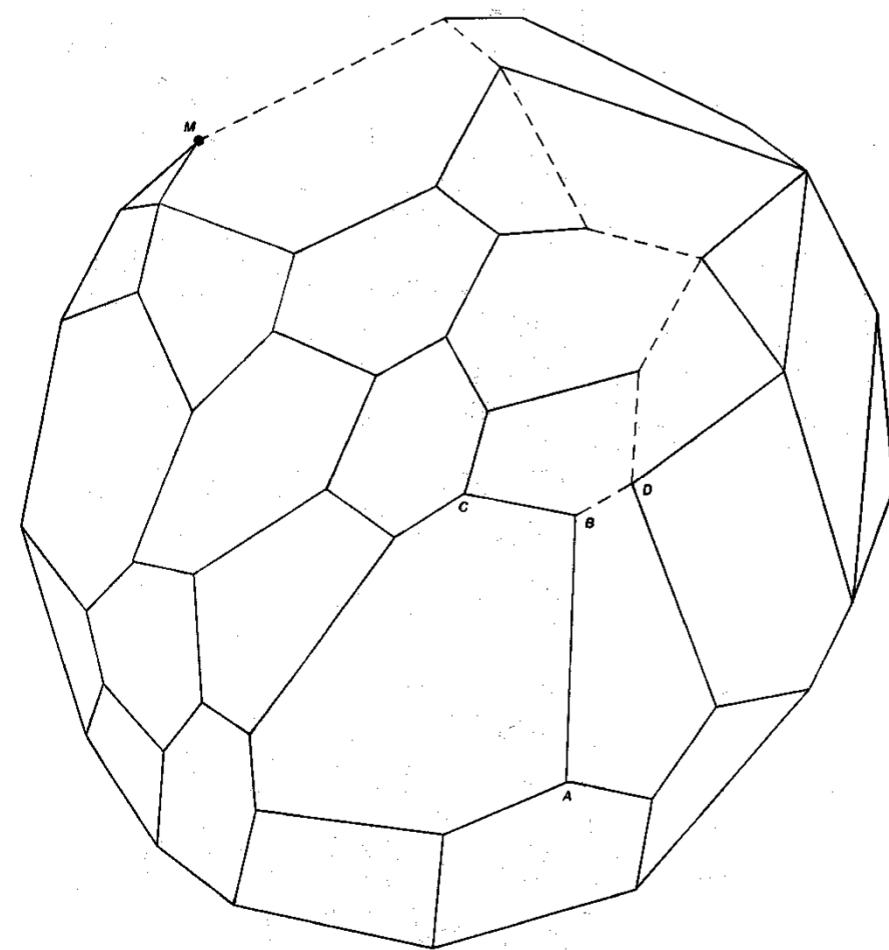
$$(P) \quad \begin{aligned} & \max \quad \sum_{j=1}^n c_j x_j \\ & \sum_{j=1}^n a_{ij} x_j \leq b_i \quad 1 \leq i \leq m \\ & x_j \geq 0 \quad 1 \leq j \leq n \end{aligned}$$



Ý nghĩa hình học

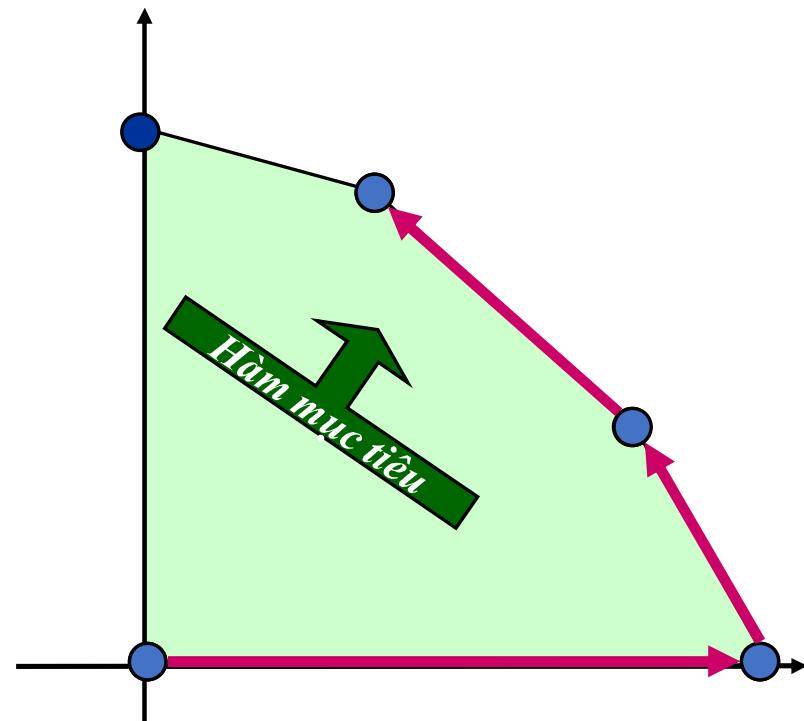
- **Kết luận:** Nếu bài toán có pa tối ưu thì nó luôn có pa tối ưu là đỉnh của miền ràng buộc vẫn đúng cho nhiều chiều.

⇒ Chỉ cần tìm pa tối ưu trong số hữu hạn phương án.



Thuật toán đơn hình

- Simplex Algorithm.
(Dantzig 1947)
 - Thuật toán thực hiện dịch chuyển từ một đỉnh sang một đỉnh kề tốt hơn cho đến khi đến được đỉnh tối ưu.
 - Thuật toán là hữu hạn nhưng có độ phức tạp hàm mũ.



Một số ký hiệu và định nghĩa

- Trong các phần tiếp theo ta sẽ chỉ làm việc với bài toán QHTT dạng chính tắc:

Tìm cực tiểu:

$$f(x_1, x_2, \dots, x_n) = \sum_{j=1}^n c_j x_j \rightarrow \min,$$

với điều kiện

$$\sum_{i=1}^n a_{ij} x_j = b_i, \quad i = 1, 2, \dots, m$$

$$x_j \geq 0, \quad j = 1, 2, \dots, n.$$

Ký hiệu và định nghĩa

- Đưa vào các ký hiệu:

- $x = (x_1, x_2, \dots, x_n)^T$ – vectơ biến số
- $c = (c_1, c_2, \dots, c_n)^T$ – vectơ hệ số hàm mục tiêu
- $A = (a_{ij})_{m \times n}$ – ma trận ràng buộc
- $b = (b_1, \dots, b_m)^T$ – vectơ ràng buộc (về phải)

Ký hiệu và định nghĩa

- Ta có thể viết lại bài toán dưới dạng ma trận:

$$f(x) = c^T x \rightarrow \min,$$

$$Ax = b, x \geq 0$$

hay

$$\min \{ f(x) = c^T x : Ax = b, x \geq 0 \}$$

- Bất đẳng thức vectơ:

$$y = (y_1, y_2, \dots, y_k) \geq 0$$

được hiểu theo nghĩa từng thành phần:

$$y_i \geq 0, i = 1, 2, \dots, k.$$

Ký hiệu và định nghĩa

- Ký hiệu các tập chỉ số:

➢ $J = \{1, 2, \dots, n\}$ tập chỉ số của các biến số

➢ $I = \{1, 2, \dots, m\}$ tập chỉ số của các ràng buộc

Khi đó ta sử dụng các ký hiệu sau

$x = x(J) = \{x_j : j \in J\}$ - vectơ biến số;

$c = c(J) = \{c_j : j \in J\}$ – vectơ hệ số hàm mục tiêu;

$A = A(I, J) = \{a_{ij} : i \in I, j \in J\}$ – ma trận ràng buộc

$A_j = (a_{ij} : i \in I)$ – vectơ cột thứ j của ma trận A .

Hệ phương trình ràng buộc cơ bản của bài toán QHTT dạng chính tắc còn có thể viết dưới dạng:

$$A_1x_1 + A_2x_2 + \dots + A_nx_n = b$$

Ký hiệu và định nghĩa

- Tập

$$D = \{x: Ax = b, x \geq 0\}$$

được gọi là miền ràng buộc (miền chấp nhận được)

x được gọi là phương án chấp nhận được.

- Phương án chấp nhận được x^* đem lại giá trị nhỏ nhất cho hàm mục tiêu, tức là

$$c^T x^* \leq c^T x \text{ với mọi } x \in D$$

được gọi là phương án tối ưu của bài toán và khi đó giá trị

$$f^* = c^T x^*$$

được gọi là giá trị tối ưu của bài toán

PHƯƠNG ÁN CƠ SỞ CHẤP NHẬN ĐƯỢC

Khái niệm phương án cơ sở chấp nhận được (pacscnd) là khái niệm trung tâm trong thuật toán đơn hình

Phương án cơ sở chấp nhận được

- Trước hết ta giả thiết rằng

$$\text{rank } (A) = m \quad (*)$$

nghĩa là hệ phương trình ràng buộc cơ bản gồm m phương trình độc lập tuyến tính.

- Chú ý: Trên thực tế giả thiết (*) là tương đương với giả thiết hệ phương trình tuyến tính $Ax = b$ có nghiệm.
- Về sau ta sẽ gỡ bỏ các giả thiết này.

Phương án cơ sở chấp nhận được

Định nghĩa 1. Ta gọi cơ sở của ma trận A là một bộ gồm m vectơ cột độc lập tuyến tính $B = \{A_{j_1}, A_{j_2}, \dots, A_{j_m}\}$ của nó.

Giả sử $B = A(I, J_B)$, trong đó $J_B = \{j_1, \dots, j_m\}$ là một cơ sở của ma trận A .

Khi đó vectơ $x = (x_1, \dots, x_n)$ thoả mãn:

$$x_j = 0, \quad j \in J_N = J \setminus J_B;$$

x_{j_k} là thành phần thứ k của vectơ $B^{-1}b$ ($k = 1, \dots, m$).

sẽ được gọi là *phương án cơ sở* tương ứng với cơ sở B .

Các biến x_j , $j \in J_B$ được gọi là *biến cơ sở*, còn x_j , $j \in J_N$ - *biến phi cơ sở*.

Phương án cơ sở chấp nhận được

- Như vậy, nếu ký hiệu

$$x_B = x(J_B), x_N = x(J_N)$$

thì phương án cơ sở x tương ứng với cơ sở B có thể xác định nhờ thủ tục sau:

1. *Đặt $x_N = 0$.*
 2. *Xác định x_B từ hệ phương trình $Bx_B = b$.*
- Từ giả thiết (*) suy ra bài toán luôn có phương án cơ sở.

Phương án cơ sở chấp nhận được

- Giả sử $x = (x_B, x_N)$ là phương án cơ sở tương ứng với cơ sở B . Khi đó bài toán QHTT dạng chính tắc có thể viết lại như sau:

$$f(x_B, x_N) = c_B x_B + c_N x_N \rightarrow \min$$

$$Bx_B + Nx_N = b,$$

$$x_B, x_N \geq 0,$$

trong đó $N = (A_j : j \in J_N)$ được gọi là ma trận phi cơ sở.

Phương án cơ sở chấp nhận được

- Xét bài toán QHTT

$$6x_1 + 2x_2 - 5x_3 + x_4 + 4x_5 - 3x_6 + 12x_7 \rightarrow \min$$

$$x_1 + x_2 + x_3 + x_4 = 4$$

$$x_1 + x_5 = 2$$

$$x_3 + x_6 = 3$$

$$3x_2 + x_3 + x_7 = 6$$

$$x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geq 0$$

Phương án cơ sở chấp nhận được

- Dưới dạng ma trận:

$$c = (6, 2, -5, 1, 4, -3, 12)^T;$$

$$b = (4, 2, 3, 6)^T;$$

$$A = \begin{vmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 3 & 1 & 0 & 0 & 0 & 1 \\ A_1 & A_2 & A_3 & A_4 & A_5 & A_6 & A_7 \end{vmatrix}$$

Phương án cơ sở chấp nhận được

- Xét cơ sở

$$B = \{A_4, A_5, A_6, A_7\} = E_4$$

- Phương án cơ sở $x = (x_1, x_2, \dots, x_7)$ tương ứng với nó thu được bằng cách đặt:

$$x_1 = 0, x_2 = 0, x_3 = 0$$

và các giá trị của $x_B = (x_4, x_5, x_6, x_7)$ thu được bằng việc giải hệ phương trình

$$Bx_B = b \text{ hay } E_4x_B = b$$

Từ đó ta tìm được: $x_B = (4, 2, 3, 6)$.

- Vậy pacs tương ứng với cơ sở B là

$$x = (0, 0, 0, 4, 2, 3, 6)$$

Phương án cơ sở chấp nhận được

- Xét cơ sở

$$B_1 = \{A_2, A_5, A_6, A_7\}$$

- Phương án cơ sở $y = (y_1, y_2, \dots, y_7)$ tương ứng với nó thu được bằng cách đặt:

$$y_1 = 0; y_3 = 0, y_4 = 0$$

và các giá trị của $y_B = (y_2, y_5, y_6, y_7)$ thu được bằng việc giải hệ

$$B_1 y_B = b$$

hay

$$y_2 = 4$$

$$y_5 = 2$$

$$y_6 = 3$$

$$3y_2 + y_7 = 6$$

Từ đó ta tìm được: $y_B = (4, 2, 3, -6)$.

- Vậy pacs tương ứng với cơ sở B_1 là

$$y = (0, 4, 0, 0, 2, 3, -6)$$

Phương án cơ sở chấp nhận được

- **Dễ thấy:**

- pacs tương ứng với cơ sở B

$$x = (0, 0, 0, 4, 2, 3, 6)$$

là phương án chấp nhận được

- còn pacs tương ứng với cơ sở B_1

$$y = (0, 4, 0, 0, 2, 3, -6)$$

không là phương án chấp nhận được

Định nghĩa. *Phương án cơ sở được gọi là phương án cơ sở chấp nhận được (lời giải cơ sở chấp nhận được) nếu như nó là phương án chấp nhận được.*

Phương án cơ sở chấp nhận được

- Bài toán QHTT có bao nhiêu pacscnd?
- Số pacscnd \leq Số cơ sở $\leq C(n,m)$
- Vậy một bài toán QHTT chỉ có thể có một số hữu hạn pacscnd

Phương án cơ sở chấp nhận được

- Bài toán QHTT luôn có pacscnd?
- Không đúng!
- Ví dụ: Nếu bài toán QHTT không có phương án chấp nhận được thì rõ ràng nó cũng không có pacscnd!
- Tuy nhiên ta có thể chứng minh kết quả sau:
- *Định lý 1. Nếu bài toán QHTT có pacnd thì nó cũng có pacscnd*



VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Công thức số gia hàm mục tiêu

Công thức số gia hàm mục tiêu

- Giả sử x là pacscnd với cơ sở tương ứng là $B=(A_j: j \in J_B)$. Ký hiệu:

$J_B = \{j_1, j_2, \dots, j_m\}$ – tập chỉ số biến cơ sở;

$J_N = J \setminus J_B$ – tập chỉ số biến phi cơ sở;

$B = (A_j: j \in J_B)$ – ma trận cơ sở;

$N = (A_j: j \in J_N)$ – ma trận phi cơ sở;

$x_B = x(J_B) = \{x_j: j \in J_B\}$, $x_N = x(J_N) = \{x_j: j \in J_N\}$ – vectơ biến cơ sở và phi cơ sở;

$c_B = c(J_B) = \{c_j: j \in J_B\}$, $c_N = c(J_N) = \{c_j: j \in J_N\}$ – vectơ hệ số hàm mục tiêu của biến cơ sở và phi cơ sở;

Công thức số gia hàm mục tiêu

- Xét pacnd $z=x+\Delta x$, trong đó $\Delta x = (\Delta x_1, \Delta x_2, \dots, \Delta x_n)$ – vectơ gia số của biến số. Ta tìm công thức tính số gia của hàm mục tiêu:

$$\Delta f = c'z - c'x = c'\Delta x.$$

- Do x, z đều là pacnd nên $Ax=b$ và $Az=b$. Vì vậy gia số Δx phải thoả mãn điều kiện $A\Delta x = 0$, hay là:

$$B\Delta x_B + N\Delta x_N = 0,$$

trong đó $\Delta x_B = (\Delta x_j : j \in J_B)$, $\Delta x_N = (\Delta x_j : j \in J_N)$.

Công thức số gia hàm mục tiêu

- Suy ra

$$\Delta x_B = -B^{-1}N\Delta x_N. \quad (1.10)$$

- Từ đó ta có

$$c' \Delta x = c'_B \Delta x_B + c'_N \Delta x_N = - (c'_B B^{-1} N - c'_N) \Delta x_N.$$

- Ký hiệu:

$u = c'_B B^{-1}$ – vectơ thế vị

$\Delta_N = (\Delta_j : j \in J_N) = uN - c'_N$ – vectơ ước lượng.

ta thu được công thức:

$$\Delta f = c' z - c' x = -\Delta_N \Delta x_N = - \sum_{j \in J_N} \Delta_j \Delta x_j$$

- Công thức thu được gọi là **công thức số gia hàm mục tiêu**

Tiêu chuẩn tối ưu

Tiêu chuẩn tối ưu

- **Định nghĩa.** Pacscnd x được gọi là *không thoái hoá* nếu như tất cả các thành phần cơ sở của nó là khác không. Bài toán QHTT được gọi là không thoái hoá nếu như tất cả các pacscnd của nó là không thoái hoá
- **Định lý 2.** (Tiêu chuẩn tối ưu) Bất đẳng thức

$$\Delta_N \leq 0 \quad (\Delta_j \leq 0, j \in J_N) \quad (1.13)$$

là điều kiện đủ và trong trường hợp không thoái hoá cũng là điều kiện cần để pacscnd x là tối ưu.

Tiêu chuẩn tối ưu

Chứng minh. Điều kiện đủ. Theo định nghĩa phương án cơ sở chấp nhận được ta có $x_N = 0$. Vì vậy, đối với mọi phương án chấp nhận được $\bar{x} = x + \Delta x$ ta có

$$\Delta x_N = \bar{x}_N - x_N = \bar{x}_N \geq 0. \quad (1.14)$$

Từ đó theo công thức số gia hàm mục tiêu suy ra

$$c' \bar{x} - c' x = -\Delta'_N \Delta x_N \geq 0$$

hay

$$c' \bar{x} \geq c' x.$$

Bất đẳng thức cuối cùng chứng tỏ x là phương án tối ưu.

Tiêu chuẩn tối ưu

Điều kiện cần. Giả sử x là phương án cơ sở chấp nhận được không thoái hóa tối ưu. Khi đó

$$x_B > 0. \quad (1.15)$$

Giả sử bất đẳng thức (1.13) không được thực hiện. Khi đó tìm được chỉ số $j_0 \in J_N$ sao cho

$$\Delta_{j_0} > 0. \quad (1.16)$$

Xây dựng vectơ $\bar{x} = x + \Delta x$, trong đó Δx được xác định như sau

$$\Delta x_{j_0} = \theta \geq 0, \quad \Delta x_j = 0, j \neq j_0, \quad j \in J_N,$$

còn số gia của các biến cơ sở được xác định từ (1.10), tức là

$$\Delta x_B = -B^{-1}N\Delta x_N = -\theta B^{-1}A_{j_0}.$$

Tiêu chuẩn tối ưu

Rõ ràng khi đó vector \bar{x} sẽ thỏa mãn hệ ràng buộc cơ bản: $A\bar{x} = A(x + \Delta x) = Ax + A\Delta x = Ax = b$. Ngoài ra, ta có

$$\bar{x}_N = \bar{x}(J_N) = x_N + \Delta x_N = \Delta x_N \geq 0$$

với mọi $\theta \geq 0$. Còn các thành phần của vector $\bar{x}_B = \bar{x}(J_B)$ thỏa mãn

$$\bar{x}_B = x_B + \Delta x_B = x_B - \theta B^{-1}A_{j_0}. \quad (1.17)$$

Do (1.15) nên từ (1.17) suy ra là với $\theta > 0$ đủ nhỏ ta có $\bar{x}_B \geq 0$. Như vậy, với giá trị θ đó \bar{x} sẽ là phương án chấp nhận được. Mặt khác, từ (1.15) và công thức số gia hàm mục tiêu ta có

$$c'\bar{x} - c'x = -\Delta'_N \Delta x_N = -\theta \Delta_{j_0} < 0$$

hay

$$c'\bar{x} < c'x.$$

Bất đẳng thức thu được là mâu thuẫn với tính tối ưu của x .

Định lý được chứng minh.

Điều kiện đủ để hàm mục tiêu không bị chặn dưới

Giả sử đối với phương án cơ sở chấp nhận được x tiêu chuẩn tối ưu không được thực hiện, tức là tìm được chỉ số $j_0 \in J_N$ sao cho $\Delta_{j_0} > 0$. Xét trường hợp khi các thành phần x_{jj_0} , $j \in J_B$ của vectơ $B^{-1}A_{j_0}$ là không dương:

$$x_{jj_0} \leq 0, \quad j \in J_B.$$

Trong trường hợp này $\bar{x}_B \geq 0 \ \forall \theta > 0$, tức là vectơ $\bar{x} = (x_B, x_N)$ sẽ là phương án chấp nhận được với mọi $\theta > 0$. Đồng thời khi đó ta có $c'\bar{x} = c'x - \theta\Delta_{j_0}$. Từ đó suy ra là khi θ càng lớn thì giá trị hàm mục tiêu tại phương án chấp nhận được \bar{x} càng nhỏ và $c'\bar{x} \rightarrow -\infty$ khi $\theta \rightarrow +\infty$. Vì vậy ta có

Định lý 1.3. (Điều kiện đủ để hàm mục tiêu không bị chặn dưới) *Nếu trong số các ước lượng của phương án cơ sở chấp nhận được x có ước lượng dương ($\Delta_{j_0} > 0$) mà ứng với nó các thành phần của vectơ $B^{-1}A_{j_0}$ là không dương ($B^{-1}A_{j_0} \leq 0$) thì hàm mục tiêu của bài toán là không bị chặn dưới.*



VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

THUẬT TOÁN ĐƠN HÌNH DẠNG MA TRẬN NGHỊCH ĐÁO

Bước lặp đơn hình

- Ta tiếp tục phân tích pacscnd x với cơ sở tương ứng B . Xét trường hợp khi tiêu chuẩn tối ưu và cả điều kiện đủ để hàm mục tiêu không bị chặn dưới không được thực hiện. Khi đó phải tìm được chỉ số j_0 sao cho $\Delta_{j_0} > 0$.
- Từ phần chứng minh điều kiện cần của tiêu chuẩn tối ưu ta thấy rằng có thể xây dựng được phương án chấp nhận được \bar{x} với giá trị hàm mục tiêu nhỏ hơn. Ta nhắc lại công thức xây dựng \bar{x} :

$$\bar{x} = x + \Delta x,$$

trong đó vectơ gia số Δx được xác định như sau

$$\Delta x_{j_0} = \theta, \Delta x_j = 0, j \neq j_0, j \in J_N$$

$$\Delta x_B = -\theta B^{-1} A_{j_0}$$

Bước lặp đơn hình

- Khi đó ta thu được

$$\bar{x}_N = \Delta x_N, \quad \bar{x}_B = x_B - \theta B^{-1} A_{j_0},$$

và giá trị hàm mục tiêu tại \bar{x} sẽ là

$$c\bar{x} = cx - \theta \Delta_{j_0}$$

- Rõ ràng: θ càng lớn thì giá trị hàm mục tiêu giảm được càng nhiều. Ta quan tâm đến giá trị θ lớn nhất có thể được. Do θ cần chọn sao cho \bar{x} là phương án chấp nhận được nên nó phải thoả mãn hệ bất phương trình tuyến tính sau:

$$\bar{x}_B = x_B - \theta B^{-1} A_{j_0} \geq 0, \quad \theta \geq 0.$$

- Ký hiệu $B^{-1} A_{j_0} = \{x_{i,j_0}: i \in J_B\}$, ta viết lại hệ bất phương trình trên dưới dạng

$$x_i - \theta x_{i,j_0} \geq 0, \quad i \in J_B,$$

$$\theta \geq 0.$$

Bước lặp đơn hình

- Ký hiệu

$$\theta_i = \begin{cases} x_i / x_{ij_0}, & \text{khi } x_{ij_0} > 0 \\ +\infty, & \text{khi } x_{ij_0} \leq 0 \end{cases}, \quad i \in J_B$$

$$\theta_0 = \theta_{i_0} = x_{i_0} / x_{i_0 j_0} = \min \{ \theta_i : i \in J_B \}$$

- Ta có nghiệm của hệ bất phương trình là

$$0 \leq \theta \leq \theta_0.$$

- Từ đó suy ra giá trị θ lớn nhất có thể chọn là θ_0 . Khi đó nếu thay x bởi phuong án $\bar{x}(\theta_0) = x + \Delta x$, trong đó vectơ gia số được xây dựng với $\theta = \theta_0$, giá trị hàm mục tiêu giảm đi được một lượng là $\theta_{i_0} \Delta_{j_0} > 0$.

Bước lặp đơn hình

Ta sẽ chỉ ra là \bar{x} xây dựng như vậy cũng là phương án cơ sở chấp nhận được. Rõ ràng $\bar{x}_j = 0$ với $j \in \bar{J}_N = J_N \setminus j_0 \cup i_0$. Gọi $\bar{J}_B = J_B \setminus i_0 \cup j_0$, $\bar{B} = \{A_j : j \in \bar{J}_B\}$. (\bar{B} thu được từ B bằng cách thay cột A_{j_0} bởi cột A_{i_0}). Ta có

$$B^{-1}\bar{B} = (B^{-1}A_{j_1}, \dots, B^{-1}A_{j_0}, \dots, B^{-1}A_{j_m}),$$

suy ra

$$B^{-1}\bar{B} = V = \begin{pmatrix} 1 & \dots & x_{j_1 j_0} & \dots & 0 \\ & & \dots & & \\ 0 & \dots & x_{i_0 j_0} & \dots & 0 \\ & & \dots & & \\ 0 & \dots & x_{j_m j_0} & \dots & 1 \end{pmatrix}. \quad (1.21)$$

Do đó $\det(B^{-1}\bar{B}) = \det V = x_{i_0 j_0} \neq 0$, suy ra $\det \bar{B} \neq 0$ hay \bar{B} là cơ sở của bài toán. Vì vậy \bar{x} là pacscnd.

Ta gọi việc chuyển từ pacscnd x sang pacscnd \bar{x} theo thủ tục mô tả ở trên là một bước lặp đơn hình thực hiện đối với pacscnd x .

Thuật toán đơn hình dạng ma trận nghịch đảo

- Trong các tính toán của một bước lặp đơn hình, ma trận B^{-1} giữ một vai trò quan trọng. Thuật toán dưới đây sẽ được mô tả trong ngôn ngữ của ma trận nghịch đảo B^{-1} .
- Bước khởi tạo.

Tìm một phần tử x với cơ sở tương ứng là B .

Tính B^{-1} .

Thuật toán đơn hình dạng ma trận nghịch đảo

Bước $k = 1, 2, \dots$ Ở đầu bước lặp ta đã có ma trận B_k^{-1} ($B_1^{-1} = B^{-1}$), tập chỉ số biến cơ sở J_B^k , $J_N^k = J \setminus J_B^k$, phương án cơ sở chấp nhận được $x^k = (x_B^k, x_N^k) = (B_k^{-1}b, 0)$.

- 1) Tính $u = c'_{B_k} B_k^{-1}$ (tương đương với giải hệ phương trình $u' B_k = c_{B_k}$).
- 2) Tính $\Delta_j = u' A_j - c_j$, $j \in J_N^k$.
- 3) Nếu $\Delta_j \leq 0 \forall j \in J_N^k$ thì thuật toán kết thúc và x^k là phương án tối ưu cần tìm.
- 4) Nếu trong số các Δ_j , $j \in J_N^k$ còn ước lượng dương, thì chọn $\Delta_{j_0} > 0$ ($j_0 = j_0(k)$).
- 5) Tính x_{j_0} , $j \in J_B^k$ là các thành phần của vectơ $y = B_k^{-1} A_{j_0}$ (tương đương với giải hệ phương trình $B_k y = A_{j_0}$).
- 6) Nếu $x_{j_0} \leq 0 \forall j \in J_B^k$ thì hàm mục tiêu của bài toán là không bị chặn dưới. Thuật toán kết thúc.

Thuật toán đơn hình dạng ma trận nghịch đảo

7) Tính

$$\theta_i = \begin{cases} x_i/x_{ij_0}, & \text{nếu } x_{ij_0} > 0, i \in J_B^k, \\ +\infty, & \text{nếu } x_{ij_0} \leq 0, i \in J_B^k. \end{cases}$$

$$\theta_0 = \theta_{i_0} = x_{i_0}/x_{i_0 j_0} = \min\{\theta_i : i \in J_B^k\}.$$

$(i_0 = i_0(k)).$

8) Đặt

$$J_B^{k+1} = J_B^k \setminus i_0 \cup j_0; J_N^{k+1} = J_N^k \setminus j_0 \cup i_0,$$

$$B_{k+1} = \{A_j : j \in J_B^{k+1}\}.$$

Tính ma trận nghịch đảo B_{k+1}^{-1} của B_{k+1} , chuyển sang bước $k+1$.

Thuật toán đơn hình dạng ma trận nghịch đảo

Chú ý. Trong 4) Δ_{j_0} là ước lượng dương tuỳ ý. Tuy nhiên, trong trường hợp $|J_N^k|$ là không quá lớn ta có thể chọn nó theo quy tắc sau

$$\Delta_{j_0} = \max \{ \Delta_j : j \in J_N^k \}.$$

Để tính ma trận nghịch đảo B_{k+1}^{-1} của $\bar{B} = B_{k+1}$, từ ma trận nghịch đảo B_k^{-1} của $B = B_k$, để ý đến mối quan hệ của chúng trong công thức (1.21): $B^{-1}\bar{B} = V$, suy ra

$$\bar{B}^{-1} = V^{-1}B^{-1}. \quad (1.22)$$

Do

$$V^{-1} = \begin{pmatrix} 1 & \dots & -x_{j_1 j_0}/x_{i_0 j_0} & \dots & 0 \\ & & \dots & & \\ 0 & \dots & 1/x_{i_0 j_0} & \dots & 0 \\ & & \dots & & \\ 0 & \dots & -x_{j_m j_0}/x_{i_0 j_0} & \dots & 1 \end{pmatrix}, \quad (1.23)$$

Thuật toán đơn hình dạng ma trận nghịch đảo

Do

$$V^{-1} = \begin{pmatrix} 1 & \dots & -x_{j_1 j_0}/x_{i_0 j_0} & \dots & 0 \\ & & \ddots & & \\ 0 & \dots & 1/x_{i_0 j_0} & \dots & 0 \\ & & \ddots & & \\ 0 & \dots & -x_{j_m j_0}/x_{i_0 j_0} & \dots & 1 \end{pmatrix}, \quad (1.23)$$

nên nếu ký hiệu $B_{k+1}^{-1} = \{ \bar{u}_{ij} : i \in J_B^{k+1}, j \in I \}$, $B_k^{-1} = \{ u_{ij} : i \in J_B^k, j \in I \}$ thì từ (1.22), (1.23) ta suy ra công thức sau đây để tính các phần tử của B_{k+1}^{-1}

$$\bar{u}_{i_0 j} = u_{i_0 j}/x_{i_0 j_0}, \quad j \in I,$$

$$\bar{u}_{ij} = u_{ij} - x_{ij_0} u_{i_0 j}/x_{i_0 j_0}, \quad i \neq i_0, \quad i \in J_B^{k+1}, \quad j \in I.$$

Thuật toán đơn hình dạng bảng

Thuật toán đơn hình dạng bảng

- Để thuận tiện cho những tính toán bằng tay ta sẽ mô tả một dạng khác của thuật toán đơn hình đó là thuật toán đơn hình dạng bảng.
- Giả sử ta có $pacscnd\ x$ với cơ sở tương ứng là B . Các ký hiệu vẫn giữ nguyên như trước.
- Ta gọi ***bảng đơn hình tương ứng với pacscnd x*** là bảng sau đây

Bảng đơn hình

c_j cơ sở	Cơ sở	Phương án	c_1	...	c_j	...	c_n	θ
			A_1	...	A_j	...	A_n	
c_{j_1}	A_{j_1}	x_{j_1}			$x_{j_1 j}$			θ_{j_1}
...
c_i	A_i	x_i			$x_{i j}$			θ_i
...
c_{j_m}	A_{j_m}	x_{j_m}			$x_{j_m j}$			θ_{j_m}
Δ			Δ_1	...	Δ_j	...	Δ_n	

Bảng đơn hình

- Cột đầu tiên ghi hệ số hàm mục tiêu của các biến cơ sở.
- Cột thứ hai dành để ghi tên của các cột cơ sở.
- Cột thứ ba ghi các giá trị của các biến cơ sở (các thành phần của vectơ $x_B = \{x_j : j \in J_B\} = B^{-1}b$).
- Các phần tử x_{ij} , $i \in J_B$ trong các cột tiếp theo được tính theo công thức:

$$\{x_{ij}, i \in J_B\} = B^{-1}A_j, j=1,2,\dots,n.$$

- Cột cuối cùng để ghi các tỷ số θ_i .
- Dòng đầu tiên của bảng ghi hệ số hàm mục tiêu của các biến (c_j).
- Dòng tiếp theo ghi tên của các cột A_1, \dots, A_n .
- Dòng cuối cùng gọi là dòng ước lượng:

$$\Delta_j = \sum_{i \in J_B} c_i x_{ij} - c_j, j=1,2,\dots,n.$$

- Có thể thấy rằng $\Delta_j = 0, j \in J_B$.

Thuật toán đơn hình dạng bảng

Với bảng đơn hình xây dựng được ta có thể tiến hành thực hiện một bước lặp đơn hình đối với phương án cơ sở chấp nhận được x như sau.

1. *Kiểm tra tiêu chuẩn tối ưu:* Nếu các phần tử của dòng ước lượng là không dương ($\Delta_j \leq 0, j = 1, \dots, n$) thì phương án cơ sở chấp nhận được đang xét là tối ưu, thuật toán kết thúc.

2. *Kiểm tra điều kiện đủ để hàm mục tiêu không bị chặn dưới:* Nếu có ước lượng $\Delta_{j_0} > 0$ mà các phần tử trong bảng đơn hình trên cột ứng với nó đều không dương ($x_{j_0} \leq 0, j \in J_B$), thì hàm mục tiêu của bài toán là không bị chặn dưới, thuật toán kết thúc.

3. *Tìm cột xoay:* Tìm $\Delta_{j_0} = \max \{ \Delta_j : j = 1, \dots, n \} > 0$.

Cột A_{j_0} gọi là cột xoay (cột đưa vào cơ sở), còn biến x_{j_0} gọi là biến đưa vào.

Tìm cột xoay

c_j cơ sở	Cơ sở	Phương án	c_1	...	c_{j_0}	...	c_n	θ
			A_1	...	A_{j_0}	...	A_n	
c_{j_1}	A_{j_1}	x_{j_1}			$x_{j_1 j_0}$			θ_{j_1}
...
c_i	A_i	x_i			$x_{i j_0}$			θ_i
...
c_{j_m}	A_{j_m}	x_{j_m}			$x_{j_m j_0}$			θ_{j_m}
Δ			Δ_1	...	Δ_{j_0}	...	Δ_n	

Thuật toán đơn hình dạng bảng

4. *Tìm dòng xoay:* Tính

$$\theta_i = \begin{cases} x_i/x_{ij_0}, & \text{nếu } x_{ij_0} > 0, \\ +\infty, & \text{nếu } x_{ij_0} \leq 0, \end{cases} i \in J_B,$$

$$\theta_0 = \theta_{i_0} = x_{i_0}/x_{i_0 j_0} = \min\{\theta_i : i \in J_B\}.$$

Dòng A_{i_0} gọi là dòng xoay (A_{i_0} - cột đưa ra cơ sở), còn biến x_{i_0} gọi là biến đưa ra. Phần tử nằm trên giao của dòng xoay và cột xoay của bảng đơn hình được gọi là *phần tử xoay*.

5. *Thực hiện phép biến đổi đơn hình chuyển từ phương án cơ sở chấp nhận được x sang phương án cơ sở chấp nhận được \bar{x} :* Bảng đơn hình tương ứng với \bar{x} (gọi tắt là bảng mới) có thể thu được từ bảng đơn hình tương ứng với x (gọi tắt là bảng cũ) theo các quy tắc biến đổi sau đây (suy trực tiếp từ các công thức (1.22), (1.23)):

Tìm dòng xoay

c_j cơ sở	Cơ sở	Phương án	c_1	...	c_{j_0}	...	c_n	θ
			A_1	...	A_{j_0}	...	A_n	
c_{j_1}	A_{j_1}	x_{j_1}			$x_{j_1 j_0}$			θ_{j_1}
...
c_{i_0}	A_{i_0}	x_{i_0}			$x_{i_0 j_0}$			θ_{i_0}
...
c_{j_m}	A_{j_m}	x_{j_m}			$x_{j_m j_0}$			θ_{j_m}
Δ			Δ_1	...	Δ_{j_0}	...	Δ_n	

Phép biến đổi đơn hình

- i) Các phần tử ở vị trí dòng xoay trong bảng mới ($\bar{x}_{i_0 j}$) bằng các phần tử tương ứng trong bảng cũ chia cho phần tử xoay: $\bar{x}_{i_0 j} = x_{i_0 j}/x_{i_0 j_0}$, $j \in J$.
- ii) Các phần tử ở vị trí cột xoay trong bảng mới, ngoại trừ phần tử nằm trên vị trí phần tử xoay bằng 1, còn tất cả là bằng 0.
- iii) Các phần tử cần tính còn lại trong bảng mới (\bar{x}_{ij} , $\bar{\Delta}_j$) được tính từ các phần tử tương ứng trong bảng cũ theo công thức sau:

$$\bar{x}_{ij} = x_{ij} - x_{i_0 j}x_{i_0 j_0}/x_{i_0 j_0}, \quad i \in J_B \ (i \neq i_0), \quad j \in J \ (j \neq j_0),$$

$$\bar{\Delta}_j = \Delta_j - x_{i_0 j}\Delta_{j_0}/x_{i_0 j_0}, \quad j \in J \ (j \neq j_0).$$

Thuật toán đơn hình dạng bảng

Các công thức trên dễ dàng ghi nhớ nhờ các hình chữ nhật sau



Chính vì vậy các công thức biến đổi vừa nêu còn được gọi là *quy tắc hình chữ nhật*.

Chú ý. Các phép biến đổi *i), ii), iii)* sẽ hoàn toàn được xác định nếu ta chỉ ra phần tử xoay $x_{i_0 j_0}$. Ta sẽ gọi chúng là phép biến đổi đơn hình với phần tử xoay được chọn là $x_{i_0 j_0}$.

Qui tắc hình chữ nhật

c_j cơ sở	Cơ sở	Phương án	c_1	...	c_{j_0}	...	c_n	θ
			A_1	...	A_{j_0}	...	A_n	
c_{j_1}	A_{j_1}	x_{j_1}	x_{j_11}		$x_{j_1 j_0}$		$x_{j_1 n}$	θ_{j_1}
...
c_{i_0}	A_{i_0}	x_{i_0}	x_{i_01}		$x_{i_0 j_0}$		$x_{i_0 n}$	θ_{i_0}
...
c_{j_m}	A_{j_m}	x_{j_m}	x_{j_m1}		$x_{j_m j_0}$		$x_{j_m n}$	θ_{j_m}
Δ			Δ_1	...	Δ_{j_0}	...	Δ_n	

Ví dụ

Ví dụ. Giải bài toán quy hoạch tuyến tính sau bằng thuật toán đơn hình:

$$x_1 - 6x_2 + 32x_3 + x_4 + x_5 + 10x_6 + 100x_7 \rightarrow \min,$$

$$x_1 + x_4 + 6x_6 = 9,$$

$$3x_1 + x_2 - 4x_3 + 2x_6 + x_7 = 2,$$

$$x_1 + 2x_2 + x_5 + 2x_6 = 6,$$

$$x_j \geq 0, \quad j = 1, 2, \dots, 7.$$

Bắt đầu từ phương án cơ sở chấp nhận được $x' = (0, 0, 0, 9, 6, 0, 2)$ với cơ sở tương ứng là $B = (A_4, A_7, A_5)$. Do cơ sở B là ma trận đơn vị nên dễ dàng xây dựng bảng đơn hình ứng với phương án cơ sở chấp nhận được x .

Bảng đơn hình

c_j cơ sở	Cơ sở	Phương án	1	-6	32	1	1	10	100	θ
			A_1	A_2	A_3	A_4	A_5	A_6	A_7	
1	A_4	9	1	0	0	1	0	6	0	9
100	A_7	2	3	1	-4	0	0	2	1	2/3
1	A_5	6	1	2	0	0	1	2	0	6
Δ			301	108	-432	0	0	198	0	

Tìm cột xoay: Cột xoay là Cột có ước lượng lớn nhất

Bảng đơn hình

c_j cơ sở	Cơ sở	Phương án	1	-6	32	1	1	10	100	θ
			A_1	A_2	A_3	A_4	A_5	A_6	A_7	
1	A_4	9	1	0	0	1	0	6	0	9
100	A_7	2	3	1	-4	0	0	2	1	2/3
1	A_5	6	1	2	0	0	1	2	0	6
Δ			301	108	-432	0	0	198	0	

Tìm dòng xoay: Tính các tỷ số θ_i . Dòng xoay là dòng có tỷ số nhỏ nhất

Biến đổi bảng đơn hình

c_j cơ sở	Cơ sở	Phương án	1	-6	32	1	1	10	100	θ
			A_1	A_2	A_3	A_4	A_5	A_6	A_7	
1	A_4									
1	A_1	2/3	1	1/3	-4/3	0	0	2/3	1/3	2
1	A_5									
Δ										

Biến đổi bảng: Các phần tử trên dòng xoay ở bảng mới = Các phần tử tương ứng trong bảng cũ chia cho phần tử xoay.

Biến đổi bảng đơn hình

c_j cơ sở	Cơ sở	Phương án	1	-6	32	1	1	10	100	θ
			A_1	A_2	A_3	A_4	A_5	A_6	A_7	
1	A_4									
1	A_1	$2/3$	1	$1/3$	$-4/3$	0	0	$2/3$	$1/3$	2
1	A_5									
Δ										

Biến đổi bảng: Các phần tử trên các cột cơ sở là các vectơ đơn vị.

Biến đổi bảng đơn hình

c_j	Cơ sở	Phương án	1	-6	32	1	1	10	100	θ
cơ sở			A_1	A_2	A_3	A_4	A_5	A_6	A_7	
1	A_4		0			1	0			
1	A_1	$2/3$	1	$1/3$	$-4/3$	0	0	$2/3$	$1/3$	
1	A_5		0			0	1			
Δ			0			0	0			



Biến đổi bảng: Các phần tử trên các cột cơ sở là các vectơ đơn vị.

Bảng đơn hình ở bước 2

c_j cơ sở	Cơ sở	Phương án	1	-6	32	1	1	10	100	θ
			A_1	A_2	A_3	A_4	A_5	A_6	A_7	
1	A_4	$25/3$	0	$-1/3$	$4/3$	1	0	$16/3$	$-1/3$	-
1	A_1	$2/3$	1	$1/3$	$-4/3$	0	0	$2/3$	$1/3$	2
1	A_5	$16/3$	0	$5/3$	$4/3$	0	1	$4/3$	$-1/3$	$16/5$
Δ			0	$23/3$	$-92/3$	0	0	$-8/3$	$-301/3$	

Biến đổi bảng: Các phần tử còn lại tính theo qui tắc hình chữ nhật.

Bảng đơn hình ở bước 3

c_j cơ sở	Cơ sở	Phương án	1	-6	32	1	1	10	100	θ
			A_1	A_2	A_3	A_4	A_5	A_6	A_7	
1	A_4	9	1	0	0	1	0	6	0	
-6	A_2	2	3	1	-4	0	0	2	1	
1	A_5	2	-5	0	8	0	1	-2	-2	
Δ			-23	0	0	0	0	-18	-108	

Tiêu chuẩn tối ưu được thỏa mãn. Thuật toán kết thúc.

Phương án tối ưu: $x^* = (0, 2, 0, 9, 2, 0, 0)$. Giá trị tối ưu: $f_* = -1$



VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Tính hữu hạn của thuật toán đơn hình

Định nghĩa. *Thuật toán giải bài toán tối ưu hoá được gọi là hữu hạn nếu như nó cho phép sau một số hữu hạn phép tính tìm được phương án tối ưu của bài toán.*

Do mỗi bước lặp của thuật toán đơn hình có thể thực hiện xong sau một số hữu hạn phép tính. Vì vậy, để chứng minh tính hữu hạn của thuật toán đơn hình ta sẽ chứng minh rằng nó phải kết thúc sau hữu hạn bước lặp.

Định nghĩa. *Bài toán QHTT được gọi là không thoái hoá nếu tất cả các phương án cơ sở chấp nhận được của nó là không thoái hoá, trong trường hợp ngược lại bài toán được gọi là thoái hoá.*

Định lý 1.4. *Giả sử bài toán QHTT là không thoái hoá và có phương án tối ưu. Khi đó với mọi phương án cơ sở chấp nhận được xuất phát thuật toán đơn hình là hữu hạn.*

Tính hữu hạn của thuật toán đơn hình

Chứng minh. Giả sử x^1 là phương án cơ sở chấp nhận được xuất phát. Trong quá trình thực hiện thuật toán đơn hình ta sẽ xây dựng được dãy $x^k, B_k, k = 1, 2, \dots$ các phương án cơ sở chấp nhận được và cơ sở tương ứng với chúng của bài toán. Do bài toán là không thoái hoá nên mỗi phương án cơ sở chấp nhận được x^k là không thoái hoá, do đó ở mỗi bước lặp, khi chuyển từ phương án cơ sở chấp nhận được x^k sang phương án cơ sở chấp nhận được x^{k+1} hàm mục tiêu sẽ giảm thực sự (xem công thức (1.19)). Do

$$c'x^k = c'_{B_k}x_{B_k} = c'_{B_k}B_k^{-1}b$$

nên trong quá trình thực hiện thuật toán không có cơ sở nào bị lặp lại. Mặt khác, ma trận A chỉ có một số hữu hạn cơ sở (số cơ sở của A không vượt quá C_n^m), vì thế, sau một số hữu hạn bước lặp ta sẽ xây dựng được phương án cơ sở chấp nhận được $x^{k_0}, k_0 < +\infty$, mà tại nó tiêu chuẩn tối ưu được thực hiện.

Định lý được chứng minh.

THUẬT TOÁN ĐƠN HÌNH HAI PHA

Thuật toán đơn hình hai pha

Thuật toán đơn hình mô tả trong các mục trước để giải bài toán QHTT

$$\min \{c'x : Ax = b, x \geq 0\} \quad (1.25)$$

được xây dựng dựa trên các giả thiết sau đây

- i) $\text{rank } A = m$;
- ii) $D = \{x : Ax = b, x \geq 0\} \neq \emptyset$;
- iii) Biết một phương án cơ sở chấp nhận được.

Trong mục này ta sẽ xây dựng thuật toán giải bài toán đặt ra mà không cần sử dụng các giả thiết vừa nêu.

Bài toán phụ

- Bài toán xuất phát

$$\min \left\{ \sum_{j=1}^n c_j x_j : \sum_{j=1}^n a_{ij} x_j = b_i, \quad i = 1, 2, \dots, m, \quad x_j \geq 0, \quad j = 1, 2, \dots, n \right\} \quad (1.25)$$

- Không giảm tổng quát ta giả thiết là

$$b_i \geq 0, \quad i = 1, 2, \dots, m,$$

bởi vì: nếu có $b_i < 0$ thì chỉ cần nhân hai vế phương trình tương ứng với -1.

- Theo các thông số của bài toán đã cho ta xây dựng bài toán phụ sau đây

$$\begin{aligned} \sum_{i=1}^m x_{n+i} &\rightarrow \min, \\ \sum_{j=1}^n a_{ij} x_j + x_{n+i} &= b_i, \quad i = 1, 2, \dots, m, \\ x_j &\geq 0, \quad j = 1, 2, \dots, n, n+1, \dots, n+m. \end{aligned} \quad (1.27)$$

- Vecto $x_u = (x_{n+1}, x_{n+2}, \dots, x_{n+m})$ được gọi là vecto biến giả.

Bài toán phụ

Bổ đề sau đây cho thấy mối liên hệ giữa bài toán (1.25) và bài toán (1.27).

Bổ đề 1. *Bài toán (1.25) có phương án chấp nhận được khi và chỉ khi thành phần x_u^* trong phương án tối ưu (x^*, x_u^*) của bài toán (1.27) là bằng không.*

Chứng minh. Điều kiện cần. Giả sử x^* là phương án chấp nhận được của bài toán (1.25). Khi đó rõ ràng $(x^*, x_u^* = 0)$ là phương án chấp nhận được của bài toán (1.27). Mặt khác, do $e'x_u^* = 0 \leq e'x_u$, với mọi (x, x_u) là phương án chấp nhận được của bài toán (1.27) nên (x^*, x_u^*) là phương án tối ưu của nó.

Điều kiện đủ. Rõ ràng nếu $(x^*, x_u^* = 0)$ là phương án tối ưu của bài toán (1.27) thì x^* là phương án chấp nhận được của bài toán (1.25).

Pha thứ 1: Giải bài toán phụ

- Đối với bài toán phụ ta có ngay một pacscnd là

$$(x=0, x_u=b)$$

với cơ sở tương ứng là

$$B = \{A_{n+1}, A_{n+2}, \dots, A_{n+m}\} = E_m$$

trong đó A_{n+i} là vectơ cột tương ứng với biến giả x_{n+i} , $i=1, 2, \dots, m$.

- Vì vậy ta có thể áp dụng thuật toán đơn hình để giải bài toán phụ. Việc giải bài toán phụ bằng thuật toán đơn hình được gọi là **pha thứ nhất** của thuật toán đơn hình hai pha giải bài toán qui hoạch tuyến tính dạng chính tắc (1.25), và bài toán phụ còn được gọi là bài toán ở pha thứ nhất

Pha thứ 1: Giải bài toán phụ

Kết thúc pha thứ nhất ta sẽ xây dựng được phương án cơ sở chấp nhận được tối ưu (x^*, x_u^*) với cơ sở tương ứng là $B^* = \{A_j : j \in J_B^*\}$ của bài toán (1.27). Có thể xảy ra một trong 3 khả năng sau:

- i) $x_u^* \neq 0$;
- ii) $x_u^* = 0$ và ma trận B^* không chứa các cột ứng với biến giả, tức là nó chứa toàn bộ cột của ma trận ràng buộc của bài toán (1.25):

$$B^* = \{A_j : j \in J_B^*\}, J_B^* \cap J_u = \emptyset.$$

- iii) $x_u^* = 0$ và ma trận B^* có chứa cột ứng với biến giả, tức là

$$B^* = \{A_j : j \in J_B^*\}, J_B^* \cap J_u \neq \emptyset.$$

Pha thứ 1: Giải bài toán phụ

Ta sẽ xét từng trường hợp một.

i) Nếu $x_u^* \neq 0$, thì theo bối đề 1, bài toán (1.25) là không có phương án chấp nhận được, thuật toán kết thúc.

ii) Trong trường hợp này x^* là phương án cơ sở chấp nhận được của bài toán (1.25) với cơ sở tương ứng là B^* . Bắt đầu từ nó ta có thể tiến hành thuật toán đơn hình để giải bài toán (1.25). Giai đoạn này gọi là pha thứ hai của thuật toán đơn hình hai pha và toàn bộ thủ tục vừa trình bày được gọi là thuật toán đơn hình hai pha để giải bài toán QHTT (1.25).

iii) Ký hiệu $x_{i_*}, i_* \in J_B^* \cap J_u$ là một thành phần biến giả trong phương án cơ sở chấp nhận được tối ưu (x^*, x_u^*) của bài toán (1.27).

Ký hiệu $x_{i_* j}, j \in J \cup J_u$ là các phần tử của dòng i_* trong bảng đơn hình tương ứng với phương án tối ưu (x^*, x_u^*) của bài toán (1.27).

Bảng đơn hình

c_j cơ sở	Cơ sở	Phương án	c^*_1	...	c^*_j	...	c^*_{n+m}	θ
			A_1	...	A_j	...	A_{n+m}	
$c^*_{j(1)}$	$A_{j(1)}$	$x_{j(1)}$			$x_{j(1)j}$			
...			
$c^*_{i^*}$	A_{i^*}	x_{i^*}			x_{i^*j}			
...			
$c^*_{j(m)}$	$A_{j(m)}$	$x_{j(m)}$			$x_{j(m)j}$			
Δ			Δ_1	...	Δ_j	...	Δ_{n+m}	

Pha thứ 1: Giải bài toán phụ

Nếu tìm được chỉ số $j_* \in J \setminus J_B^*$ sao cho $x_{i_* j_*} \neq 0$ thì thực hiện một phép biến đổi bằng đơn hình với phần tử xoay được chọn là $x_{i_* j_*}$ ta sẽ đưa được thành phần biến giả x_{i_*} ra khỏi cơ sở và thay vào chỗ của nó là biến x_{j_*} .

Nếu $x_{i_* j} = 0, \forall j \in J \setminus J_B^*$, thì điều đó có nghĩa là phương trình tương ứng với nó trong hệ phương trình tuyến tính $Ax = b$ là hệ quả của các phương trình còn lại. Khi đó từ bảng đơn hình tương ứng với phương án tối ưu (x^*, x_u^*) của bài toán (1.27) ta có thể xoá bỏ dòng nói trên và đồng thời xoá bỏ luôn cột ứng với biến giả x_{i_*} .

Trong cả hai trường hợp vừa nêu ta đều loại bỏ được biến giả x_{i_*} khỏi cơ sở.

Thuật toán đơn hình hai pha

- Lần lượt điểm diện tất cả các thành phần biến giả trong cơ sở theo thủ tục vừa làm đối với x_i^* ta sẽ đi đến bảng đơn hình mới mà trong đó không còn thành phần biến giả trong cơ sở, tức là đến được trường hợp ii), đồng thời trong quá trình này ta cũng loại bỏ được tất cả các ràng buộc phụ thuộc tuyến tính trong hệ $Ax=b$.
- Từ bảng thu được ta có thể bắt đầu thực hiện pha thứ hai của thuật toán đơn hình hai pha.

Thuật toán đơn hình hai pha

- Như vậy thuật toán đơn hình hai pha áp dụng đối với một QHTT bất kỳ chỉ có thể kết thúc ở một trong ba tình huống sau đây:
 - 1) Bài toán không có pacnd.
 - 2) Bài toán có hàm mục tiêu không bị chặn.
 - 3) Tìm được phương án cơ sở tối ưu cho bài toán.
- Đồng thời trong quá trình thực hiện thuật toán ta cũng phát hiện và loại bỏ được tất cả các ràng buộc phụ thuộc tuyến tính trong hệ ràng buộc cơ bản $Ax=b$.

Một số kết quả lý thuyết

- **Định lý 1.** *Hết bài toán QHTT có pacnd thì nó cũng có pacscnd.*
- **Chứng minh.**

Áp dụng thuật toán đơn hình hai pha đối với bài toán đặt ra, kết thúc pha thứ nhất ta thu được phương án cơ sở chấp nhận được cho bài toán.

Một số kết quả lý thuyết

- **Định lý 1.6.** *Nếu bài toán QHTT có phương án tối ưu thì nó cũng có phương án cơ sở tối ưu.*
- **Chứng minh.**

Giả sử bài toán có phương án tối ưu. Khi đó, thuật toán đơn hình hai pha áp dụng để giải bài toán đặt ra chỉ có thể kết thúc ở tình huống 3), tức là thu được phương án cơ sở tối ưu cho nó.

Một số kết quả lý thuyết

- **Định lý 1.7.** *Điều kiện cần và đủ để bài toán QHTT có phương án tối ưu là hàm mục tiêu của nó bị chặn dưới trên miền ràng buộc khác rỗng.*
- **Chứng minh.**
- **Điều kiện cần.** Giả sử x^* là phương án tối ưu của bài toán. Khi đó, $f(x) \geq f(x^*)$, $\forall x \in D$, tức là hàm mục tiêu bị chặn dưới.
- **Điều kiện đủ.** Nếu bài toán có hàm mục tiêu bị chặn dưới trên miền ràng buộc khác rỗng, thì áp dụng thuật toán đơn hình hai pha để giải nó ta chỉ có thể kết thúc ở tình huống 3), tức là tìm được phương án cơ sở tối ưu cho nó.

Ví dụ

Ví dụ. Giải bài toán QHTT sau đây bằng thuật toán đơn hình hai pha:

$$2x_1 + x_2 + x_3 \rightarrow \min,$$

$$x_1 + x_2 + x_3 + x_4 + x_5 = 5,$$

$$x_1 + x_2 + 2x_3 + 2x_4 + 2x_5 = 8,$$

$$x_1 + x_2 = 2,$$

$$x_3 + x_4 + x_5 = 3,$$

$$x_j \geq 0, \quad j = 1, 2, \dots, 5.$$

Ví dụ

Bài toán phụ tương ứng có dạng:

$$\begin{array}{ccccccccc} & & & x_6 & +x_7 & +x_8 & +x_9 & \rightarrow min \\ x_1 & +x_2 & +x_3 & +x_4 & +x_5 & +x_6 & & = 5 \\ x_1 & +x_2 & +2x_3 & +2x_4 & +2x_5 & & +x_7 & = 8, \\ x_1 & +x_2 & & & & & +x_8 & = 2, \\ & x_3 & & +x_4 & +x_5 & & & +x_9 & = 3, \end{array}$$

$$x_j \geq 0, \quad j = 1, 2, \dots, 9.$$

Phương án cơ sở chấp nhận được của bài toán phụ là

$$(x, x_u) = (0, 0, 0, 0, 0, 5, 8, 2, 3),$$

Ví dụ: Pha thứ nhất

c_j	Cơ cơ	Ph/ sở	Ph/ án	0	0	0	0	1	1	1	1	θ	
				A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	
1		A_6	5	1	1	1	1	1	1	0	0	0	5
1		A_7	8	1	1	2	2	2	0	1	0	0	4
1		A_8	2	1	1	0	0	0	0	0	1	0	—
1		A_9	3	0	0	1*	1	1	1	0	0	1	3
Δ				3	3	4	4	4	0	0	0	0	

1	A_6	2	1*	1	0	0	0	1	0	0	-1	2
1	A_7	2	1	1	0	0	0	0	1	0	-2	2
1	A_8	2	1	1	0	0	0	0	0	1	0	2
0	A_3	3	0	0	1	1	1	0	0	0	1	-
Δ				3	3	0	0	0	0	0	-4	
0	A_1	2	1	1	0	0	0	1	0	0	-1	
1	A_7	0	0	0	0	0	0	-1	1	0	-1	
1	A_8	0	0	0	0	0	0	-1	0	1	1	
0	A_3	3	0	0	1	1	1	0	0	0	1	
Δ				0	0	0	0	-3	0	0	-1	

1	A_6	2	1*	1	0	0	0	1	0	0	-1	2
1	A_7	2	1	1	0	0	0	0	1	0	-2	2
1	A_8	2	1	1	0	0	0	0	0	1	0	2
0	A_3	3	0	0	1	1	1	0	0	0	1	-
Δ			3	3	0	0	0	0	0	0	-4	
0	A_1	2	1	1	0	0	0	1	0	0	-1	

1	A_8	0	0	0	0	0	0	-1	0	1	1	
0	A_3	3	0	0	1	1	1	0	0	0	1	
Δ			0	0	0	0	0	-3	0	0	-1	

1	A_6	2	1*	1	0	0	0	1	0	0	-1	2
1	A_7	2	1	1	0	0	0	0	1	0	-2	2
1	A_8	2	1	1	0	0	0	0	0	1	0	2
0	A_3	3	0	0	1	1	1	0	0	0	1	-
Δ			3	3	0	0	0	0	0	0	-4	
0	A_1	2	1	1	0	0	0	1	0	0	-1	

--	--	--	--	--	--	--	--	--	--	--	--	--

0	A_3	3	0	0	1	1	1	0	0	0	1	
Δ			0	0	0	0	0	-3	0	0	-1	

Ví dụ: Pha thứ hai

c_j	Co-	Ph/	2	1	1	0	0	θ
sở	sở	án	A_1	A_2	A_3	A_4	A_5	
2	A_1	2	1	1*	0	0	0	2
1	A_3	3	0	0	1	1	1	—
Δ			0	1	0	1	1	
1	A_2	2	1	1	0	0	0	—
1	A_3	3	0	0	1	1*	1	3
Δ			-1	0	0	1	1	
1	A_2	2	1	1	0	0	0	
0	A_4	3	0	0	1	1	1	
Δ			-1	0	-1	0	0	

Kết quả

- Phương án tối ưu:

$$x^* = (0, 2, 0, 3, 0);$$

- Giá trị tối ưu:

$$f^* = 2.$$

Hiệu quả của thuật toán đơn hình

- Một điểm yếu của TTĐH là về lý thuyết, nó có thời gian tính hàm mũ. Điều này được Klee-Minty chỉ ra bằng ví dụ sau:

$$\sum_{j=1}^n 10^{n-j} x_j \rightarrow \max,$$

$$2 \sum_{j=1}^{i-1} 10^{i-j} x_j + x_i \leq 100^{i-1}, \quad i = 1, 2, \dots, n,$$

$$x_j \geq 0, \quad j = 1, 2, \dots, n$$

- Để giải bài toán này, Thuật toán đơn hình đòi hỏi $2^n - 1$ bước lặp.

Hiệu quả của thuật toán đơn hình

- Ví dụ Klee-Minty với $n=3$:

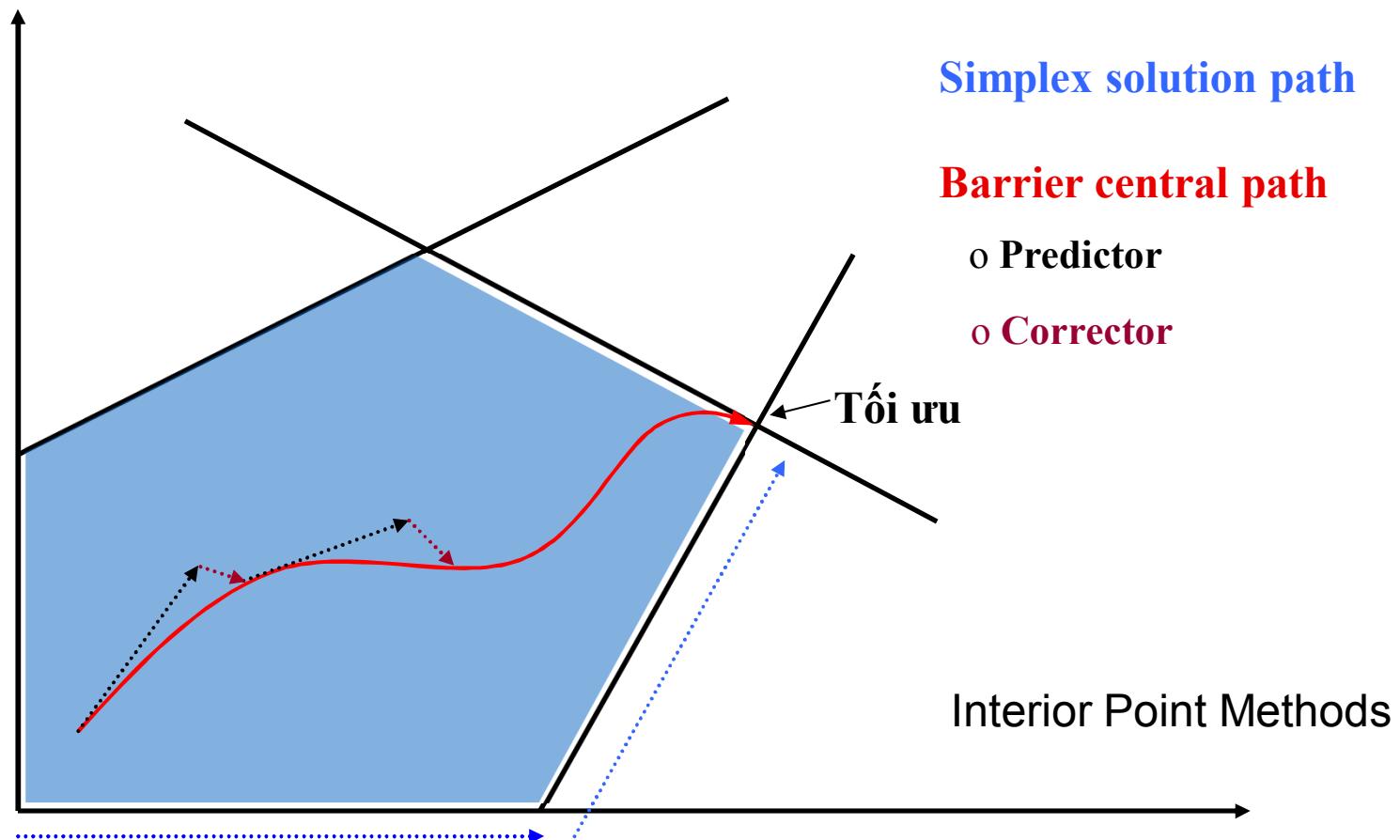
$$\begin{array}{lclcl} 100x_1 & + & 10x_2 & + & x_3 \rightarrow \max \\ x_1 & & & & \leq 1 \\ 20x_1 & + & x_2 & & \leq 100 \\ 200x_1 & + & 20x_2 & + & x_3 \leq 10000 \\ & & & & x_1, x_2, x_3 \geq 0 \end{array}$$

- Trên thực tế: Thuật toán đơn hình có thời gian tính $O(m^3)$

Thuật toán thời gian tính đa thức giải QHTT Polynomial Algorithms

- Ellipsoid. ([Khachian 1979, 1980](#))
 - đòi hỏi thời gian: $O(n^4 L)$.
 - n = số biến số
 - L = số bít cần thiết để biểu diễn dữ liệu vào
 - Đây là kết quả mang tính đột phá về mặt lý thuyết.
 - Chưa có hiệu quả thực tế.
- Karmarkar's algorithm. ([Karmarkar 1984](#))
 - $O(n^{3.5} L)$.
 - Có thời gian đa thức và có thể cài đặt hiệu quả.
- Các thuật toán điểm trong (Interior point algorithms).
 - $O(n^3 L)$.
 - Có thể sánh với thuật toán đơn hình!
 - Vượt trội so với tt đơn hình khi giải các bài toán kích thước lớn.
 - Phương pháp này được mở rộng để giải các bài toán tổng quát hơn.

Thuật toán hàm rào (Barrier Function Algorithms)



LÝ THUYẾT ĐỐI NGẦU

Lý thuyết đối ngẫu của QHTT là lĩnh vực nghiên cứu trong đó bài toán QHTT được khảo sát thông qua một bài toán QHTT bổ trợ liên hệ chặt chẽ với nó gọi là bài toán đối ngẫu

Nội dung

1. Bài toán đối ngẫu của QHTT tổng quát.
2. Định lý đối ngẫu.
3. Giải qui hoạch tuyến tính trên MATLAB

Bài toán đối ngẫu

Bài toán QHTT tổng quát

- Xét bài toán QHTT tổng quát

$$f(x_1, x_2, \dots, x_n) = \sum_{j=1}^n c_j x_j \rightarrow \min,$$

$$\sum_{j=1}^n a_{ij} x_j = b_i, i = 1, 2, \dots, p \quad (p \leq m)$$

$$\sum_{j=1}^n a_{ij} x_j \geq b_i, i = p+1, p+2, \dots, m$$

$$x_j \geq 0, j = 1, 2, \dots, n_1 \quad (n_1 \leq n)$$

$$x_j < 0, j = n_1 + 1, n_1 + 2, \dots, n$$

Bài toán QHTT tổng quát

- Đưa vào các ký hiệu:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & & & \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}$$

- ma trận ràng buộc,

Bài toán QHTT tổng quát

$a_i = (a_{i1}, a_{i2}, \dots, a_{in})'$ - dòng thứ i của ma trận A ,

$A_j = (a_{1j}, a_{2j}, \dots, a_{mj})'$ - cột thứ j của ma trận A ,

$M = \{1, 2, \dots, p\}$, $\bar{M} = \{p+1, p+2, \dots, m\}$ - các tập chỉ số ràng buộc,

$N = \{1, 2, \dots, n_1\}$, $\bar{N} = \{n_1 + 1, n_1 + 2, \dots, n\}$ - các tập chỉ số biến,

$b = (b_1, b_2, \dots, b_m)'$ - vectơ về phải,

$x = (x_1, x_2, \dots, x_n)'$ - vectơ biến số,

$c = (c_1, c_2, \dots, c_n)'$ vectơ hệ số hàm mục tiêu.

Xây dựng bài toán đối ngẫu

- Khi đó bài toán QHTT tổng quát có thể viết lại dưới dạng sau đây

$$f(x) = c'x \rightarrow \min,$$

$$a_i x = b_i, i \in M,$$

$$a_i x \geq b_i, i \in \overline{M},$$

$$x_j \geq 0, j \in N,$$

$$x_j <> 0, j \in \overline{N}.$$

Bài toán QHTT tổng quát

- Biến đổi bài toán QHTT tổng quát về dạng chính tắc bằng cách:
 - đưa vào các biến phụ x_i^s chuyển ràng buộc dạng bất đẳng thức về dạng đẳng thức,
 - thay mỗi biến không có điều kiện dấu bởi hiệu hai biến có điều kiện dấu: $x_j = x_j^+ - x_j^-$, khi đó mỗi cột A_j sẽ thay bởi hai cột A_j và $-A_j$,

ta thu được bài toán dạng chính tắc sau đây:

Xây dựng bài toán đối ngẫu

- Bài toán (2.3):

$$\hat{c}\hat{x} \rightarrow \min,$$

$$\hat{A}\hat{x} = b, \tag{2.3}$$

$$\hat{x} \geq 0,$$

trong đó

$$\hat{c} = (c_j : j \in N; (c_j, -c_j) : j \in \bar{N}; 0 : i \in \bar{M})',$$

$$\hat{x} = (x_j : j \in N; (x_j^+, x_j^-) : j \in \bar{N}; x_i^s : i \in \overline{\bar{M}})',$$

$$\hat{A} = [A_j : j \in N; (A_j, -A_j) : j \in \bar{N}; e_i : i \in \bar{M}],$$

$e_i = (0, 0, \dots, 0, -1, 0, \dots, 0)'$, ($i \in \bar{M}$) - vectơ có thành phần thứ i là bằng -1 còn tất cả các thành phần còn lại đều bằng không.

Xây dựng bài toán đối ngẫu

Từ tiêu chuẩn tối ưu của thuật toán đơn hình suy ra rằng nếu bài toán (2.3) có phương án tối ưu \hat{x}^0 và cơ sở tương ứng với nó là \hat{B} thì ta phải có

$$(\hat{c}'_B \hat{B}^{-1})' \hat{A} - \hat{c} \leq 0.$$

Do đó vectơ $y^0 = \hat{c}'_B \hat{B}^{-1}$ là phương án chấp nhận được của hệ bất phương trình tuyến tính sau:

$$y' \hat{A} \leq \hat{c}, \quad (2.4)$$

trong đó $y \in R^m$ - m -vectơ. Các bất phương trình trong (2.4) được phân làm ba nhóm phụ thuộc vào việc những vectơ cột nào của ma trận \hat{A} tham gia vào nó.

Xây dựng bài toán đối ngẫu

Nhóm ràng buộc đầu tiên là:

$$y' A_j \leq c_j, \quad j \in N. \quad (2.5)$$

Nhóm tiếp theo tương ứng các biến không có ràng buộc dấu x_j , $j \in \bar{N}$, các ràng buộc sẽ đi với nhau từng cặp một:

$$y' A_j \leq c_j, \quad -y' A_j \leq -c_j, \quad j \in \bar{N}$$

hay là chúng tương đương với hệ phương trình tuyến tính sau

$$y' A_j = c_j, \quad j \in \bar{N}. \quad (2.6)$$

Xây dựng bài toán đối ngẫu

Nhóm ràng buộc cuối cùng tương ứng với các biến phụ $x_i^s, i \in \bar{M}$

$$-y_i \leq 0, i \in \bar{M}$$

hay là

$$y_i \geq 0, i \in \bar{M}. \quad (2.7)$$

Các điều kiện (2.5)-(2.7) xác định miền ràng buộc của một bài toán quy hoạch tuyến tính mới gọi là *bài toán đối ngẫu* của bài toán xuất phát (2.1). Khi đó, bài toán xuất phát (2.1) sẽ được gọi là *bài toán gốc*.

Xây dựng bài toán đối ngẫu

- Khi đó vectơ

$$y^0 = \hat{c}_B B^{-1}$$

là phương án chấp nhận được của bài toán đối ngẫu. Nếu bây giờ ta định nghĩa hàm mục tiêu của bài toán đối ngẫu là:

$$y'b \rightarrow \max,$$

thì y^0 không những là pacnd mà còn là phương án tối ưu cho bài toán đối ngẫu.

- Các kết quả vừa nêu sẽ được phát biểu chính xác trong các định nghĩa và định lý dưới đây.

Bài toán đối ngẫu

- **Định nghĩa.** Giả sử cho bài toán QHTT tổng quát (được gọi là bài toán gốc). Khi đó bài toán đối ngẫu của nó là bài toán QHTT sau đây

Bài toán gốc (Primal Prob)

$$f(x) = c'x \rightarrow \min,$$

$$a_i x = b_i, \quad i \in M,$$

$$a_i x \geq b_i, \quad i \in \overline{M},$$

$$x_j \geq 0, \quad j \in N,$$

$$x_j < 0, \quad j \in \overline{N}$$

Bài toán đối ngẫu (Dual Prob)

$$g(y) = y'b \rightarrow \max,$$

$$y_i < 0,$$

$$y_i \geq 0,$$

$$yA_j \leq c_j,$$

$$yA_j = c_j,$$

Định lý đói ngẫu

Định lý đối ngẫu

- Trước hết ta chứng minh bổ đề sau:
- Bổ đề 2.1** (Định lý đối ngẫu yếu). Giả sử $\{x, y\}$ là cặp pacnd của bài toán gốc-đối ngẫu. Khi đó

$$f(x) = c'x \geq y'b = g(y).$$

$$\begin{aligned} f(x) &= c'x \rightarrow \min, & g(y) &= y'b \rightarrow \max, \\ a_i x &= b_i, \quad i \in M, & y_i &\leq 0, \\ a_i x &\geq b_i, \quad i \in \bar{M}, & y_i &\geq 0, \\ x_j &\geq 0, \quad j \in N, & yA_j &\leq c_j, \\ x_j &\leq 0, \quad j \in \bar{N} & yA_j &= c_j, \end{aligned}$$

Chứng minh. Do y là phương án chấp nhận được của bài toán đối ngẫu nên

$$c_j \geq y'A_j, \quad j \in N, \quad c_j = y'A_j, \quad j \in \bar{N}.$$

Mặt khác, ta có $x_j \geq 0, \quad j \in N$ (do x là phương án chấp nhận được của bài toán gốc), suy ra

$$c_j x_j \geq y'A_j x_j, \quad j \in N, \quad c_j x_j = y'A_j x_j, \quad j \in \bar{N}$$

Do đó

$$c'x \geq y'Ax.$$

Định lý đối ngẫu

Chứng minh tương tự ta thu được

$$y'Ax \geq y'b.$$

Vì vậy $c'x \geq y'Ax \geq y'b$. Bố đề được chứng minh.

Hệ quả 2.1. *Giả sử $\{\bar{x}, \bar{y}\}$ là cặp phương án chấp nhận được của bài toán gốc và đối ngẫu thoả mãn*

$$c'\bar{x} = \bar{y}'b.$$

Khi đó $\{\bar{x}, \bar{y}\}$ là cặp phương án tối ưu của bài toán gốc và đối ngẫu.

Chứng minh. Thực vậy, từ bố đề 2.1 ta suy ra

$$c'x \geq \bar{y}'b = c'\bar{x} \geq y'b,$$

đối với mọi cặp phương án chấp nhận được $\{x, y\}$ của bài toán gốc và đối ngẫu.

 Bất đẳng thức cuối cùng chứng tỏ tính tối ưu của cặp phương án chấp nhận được $\{\bar{x}, \bar{y}\}$.

Định lý đối ngẫu

Định lý 2.1. *Nếu bài toán quy hoạch tuyến tính (2.1) có phương án tối ưu thì bài toán đối ngẫu của nó cũng có phương án tối ưu và giá trị tối ưu của chúng là bằng nhau.*

Chứng minh. Giả sử bài toán quy hoạch tuyến tính có phương án tối ưu. Do đó bài toán quy hoạch tuyến tính dạng chính tắc tương ứng với nó (2.3) cũng có phương án tối ưu cơ sở \hat{x}^0 với cơ sở tương ứng là \hat{B} . Khi đó, như đã thấy ở trên, vector $y^0 = \hat{c}'_B \hat{B}^{-1}$ là phương án chấp nhận được của bài toán đối ngẫu. Gọi x^0 là phương án tối ưu của bài toán (2.1) thu được từ \hat{x}^0 , ta có

$$(y^0)' b = \hat{c}'_B \hat{B}^{-1} b = \hat{c}'_B \hat{x}_B^0 = c' x^0. \quad (2.9)$$

Từ hệ quả 2.1 suy ra y^0 là phương án tối ưu của bài toán đối ngẫu, đồng thời từ (2.9) suy ra là giá trị tối ưu của hai bài toán là bằng nhau.

Định lý được chứng minh.

Định lý đối ngẫu

Một trong những nét đặc trưng của đối ngẫu là tính đối xứng thể hiện trong định lý dưới đây.

Định lý 2.2. *Bài toán đối ngẫu của bài toán quy hoạch tuyến tính đối ngẫu là trùng với bài toán gốc.*

Chứng minh. Viết lại bài toán đối ngẫu dưới dạng

$$\begin{aligned} -y'b &\rightarrow \min, \\ -A'_j y &\geq -c_j, \quad j \in N, \\ -A'_j y &= -c_j, \quad j \in \bar{N}, \\ y_i &\geq 0, \quad i \in \bar{M}, \\ y_i &<> 0, \quad i \in M, \end{aligned}$$

và xét nó như bài toán gốc. Theo định nghĩa, bài toán đối ngẫu của nó có dạng:

$$\begin{aligned} -c'x &\rightarrow \max, \\ x_j &\geq 0, \quad j \in N, \\ x_j &<> 0, \quad j \in \bar{N}, \\ -a_i x &\leq -b_i, \quad i \in \bar{M}, \\ -a_i x &= -b_i, \quad i \in M, \end{aligned}$$

mà dễ thấy là trùng với bài toán gốc.

$$\begin{array}{lll} f(x) = c'x \rightarrow \min, & g(y) = y'b \rightarrow \max, \\ a_i x = b_i, & i \in M, & y_i <> 0, \\ a_i x \geq b_i, & i \in \bar{M}, & y_i \geq 0, \\ x_j \geq 0, & j \in N, & y A_j \leq c_j, \\ x_j <> 0, & j \in \bar{N} & y A_j = c_j, \end{array}$$

Định lý đối ngẫu

- Đối với một bài toán QHTT bất kỳ có thể xảy ra ba khả năng:
 - 1) *Bài toán có phương án tối ưu;*
 - 2) *Bài toán có hàm mục tiêu không bị chặn;*
 - 3) *Bài toán không có phương án chấp nhận được.*
- Vì vậy, đối với cặp bài toán QHTT gốc-đối ngẫu có thể xảy ra 9 tình huống mô tả trong bảng 2.1 sau đây:

Định lý đối ngẫu

Đối ngẫu Gốc	Có p/án tối ưu	Không bị chặn	Không có phương án
Có p/án tối ưu	1)	✓	✓
Không bị chặn	✓	✓	3)
Không có phương án	✓	3)	2)

Bảng 2.1.

Định lý đối ngẫu

Định lý 2.1 và 2.2 đã loại khỏi cột thứ nhất và dòng thứ nhất của bảng tất cả các ô, ngoại trừ ô đầu tiên ứng với trường hợp cả hai bài toán đều có phương án tối ưu. Các ô bị loại được đánh dấu bởi dấu \checkmark . Phần còn lại của bảng sẽ được điền theo định lý sau đây.

Định lý 2.3 (Định lý đối ngẫu). *Đối với cặp bài toán gốc đối ngẫu của quy hoạch tuyến tính chỉ có thể xảy ra một trong 3 tình huống sau đây*

- 1) Cả hai đều có phương án tối ưu và giá trị tối ưu của chúng là bằng nhau;
- 2) Cả hai đều không có phương án chấp nhận được;
- 3) Bài toán này có hàm mục tiêu không bị chặn còn bài toán kia không có phương án chấp nhận được.

Định lý đối ngẫu

Chứng minh. Từ bất đẳng thức (2.8) suy ra là nếu bài toán gốc (đối ngẫu) có hàm mục tiêu không bị chặn thì bài toán đối ngẫu (gốc) không thể có phương án chấp nhận được, tức là ta loại được ô (2,2) của bảng. Vì vậy chỉ còn 2 khả năng mà ta đánh dấu trên bảng là 2) và 3). Bằng các ví dụ ta sẽ chỉ ra rằng cả hai tình huống này đều có thể xảy ra.

Định lý đối ngẫu

Ví dụ 1. Cả hai bài toán đều không có phương án chấp nhận được.

Bài toán gốc

$$x_1 \rightarrow \min,$$

$$x_1 + x_2 \geq 1,$$

$$-x_1 - x_2 \geq 1,$$

$$x_1 <> 0, \quad x_2 <> 0.$$

Bài toán đối ngẫu

$$y_1 + y_2 \rightarrow \max,$$

$$y_1 - y_2 = 1,$$

$$y_1 - y_2 = 0,$$

$$y_1 \geq 0, \quad y_2 \geq 0.$$

Định lý đối ngẫu

Ví dụ 2. Bài toán gốc không có phương án chấp nhận được còn bài toán đối ngẫu có hàm mục tiêu không bị chặn.

Bài toán gốc

$$x_1 \rightarrow \min,$$

$$x_1 + x_2 \geq 1,$$

$$-x_1 - x_2 \geq 1,$$

$$x_1 \geq 0, x_2 \geq 0.$$

Bài toán đối ngẫu

$$y_1 + y_2 \rightarrow \max,$$

$$y_1 - y_2 \leq 1,$$

$$y_1 - y_2 \leq 0,$$

$$y_1 \geq 0, y_2 \geq 0.$$

Định lý về độ lệch bù

Nếu xem xét kỹ định nghĩa cặp bài toán gốc đối ngẫu của quy hoạch tuyến tính thì có thể cảm thấy có sự đối kháng giữa bài toán gốc và đối ngẫu: Ràng buộc càng đòi hỏi chặt ở bài toán này (ví dụ, $a_i x = b_i$, $i \in M$), thì để cân bằng lại, sang bài toán kia ràng buộc tương ứng lại được nói lỏng hơn ($y_i <> 0$, $i \in M$). Để diễn tả chính xác sự cân bằng này ta phát biểu điều kiện (gọi là điều kiện về độ lệch bù) cần và đủ để cặp phương án chấp nhận được x và y của bài toán gốc và đối ngẫu là tối ưu.

$$\begin{aligned} f(x) = c'x &\rightarrow \min, & g(y) = y'b &\rightarrow \max, \\ a_i x = b_i, & \quad i \in M, & y_i &<> 0, \\ a_i x \geq b_i, & \quad i \in \overline{M}, & y_i &\geq 0, \\ x_j \geq 0, & \quad j \in N, & yA_j &\leq c_j, \\ x_j <> 0, & \quad j \in \overline{N} & yA_j &= c_j, \end{aligned}$$

Định lý về độ lệch bù

- **Định lý 2.4.** Cặp phương án chấp nhận được của bài toán gốc-đối ngẫu $\{x, y\}$ là tối ưu khi và chỉ khi thực hiện các điều kiện sau đây:

$$\begin{aligned}(a_i x - b_i)y_i &= 0, \quad i = 1, 2, \dots, m; \\ x_j(c_j - yA_j) &= 0, \quad j = 1, 2, \dots, n.\end{aligned}$$

- **Chứng minh.** Đặt

$$\begin{aligned}f(x) = c'x \rightarrow \min, \quad g(y) = y'b \rightarrow \max, \\ a_i x = b_i, \quad i \in M, \quad y_i > 0, \\ a_i x \geq b_i, \quad i \in \overline{M}, \quad y_i \geq 0, \\ x_j \geq 0, \quad j \in N, \quad yA_j \leq c_j, \\ x_j < 0, \quad j \in \overline{N} \quad yA_j = c_j,\end{aligned}$$

$$u_i = (a_i x - b_i)y_i, \quad i = 1, 2, \dots, m;$$

$$v_j = x_j(c_j - yA_j), \quad j = 1, 2, \dots, n.$$

Do $\{x, y\}$ là cặp pacnd, nên

$$u_i \geq 0, \quad i = 1, 2, \dots, m; \quad v_j \geq 0, \quad j = 1, 2, \dots, n.$$

Định lý về độ lệch bù

Đặt

$$\alpha = \sum_i u_i \geq 0, \quad \beta = \sum_j v_j \geq 0.$$

Khi đó $\alpha = 0$ khi và chỉ khi thực hiện (2.10), và $\beta = 0$ khi và chỉ khi thực hiện (2.11). Suy ra, các điều kiện (2.10), (2.11) được thực hiện khi và chỉ khi $\alpha + \beta = 0$.

Bây giờ do

$$\begin{aligned}\alpha + \beta &= \sum_i y_i(a_i x - b_i) + \sum_j(c_j - y' A_j)x_j \\ &= -\sum_i y_i b_i + \sum_j c_j x_j \\ &= c' x - y' b.\end{aligned}$$

nên $\alpha + \beta = 0$ khi và chỉ khi $c' x = y' b$. Điều kiện $c' x = y' b$, theo hệ quả 2.1 và định lý đối ngẫu, là điều kiện cần và đủ để cặp phương án chấp nhận được gốc và đối ngẫu x và y là tối ưu. Định lý được chứng minh.

Hệ quả

- **Hệ quả.** Phương án chấp nhận được x^* là phương án tối ưu của bài toán QHTT khi và chỉ khi hệ phương trình và bất phương trình tuyến tính sau đây là có nghiệm:

$$(a_i x^* - b_i) y_i = 0, \forall i \in \overline{M},$$

$$(c_j - y A_j) x_j^* = 0, \forall j \in N,$$

$$y_i \geq 0, i \in \overline{M},$$

$$y A_j \leq c_j, j \in N,$$

$$y A_j = c_j, j \in \overline{N}.$$

$$\begin{array}{lll} f(x) = c'x \rightarrow \min, & g(y) = y'b \rightarrow \max, \\ a_i x = b_i, & i \in M, & y_i > 0, \\ a_i x \geq b_i, & i \in \overline{M}, & y_i \geq 0, \\ x_j \geq 0, & j \in N, & y A_j \leq c_j, \\ x_j > 0, & j \in \overline{N} & y A_j = c_j, \end{array}$$

Ví dụ

- Xét bài toán QHTT

$$\begin{array}{ccccccc} -2x_1 & -6x_2 & +5x_3 & -x_4 & -4x_5 & \rightarrow max \\ x_1 & -4x_2 & +2x_3 & -5x_4 & +9x_5 & = 3 \\ & x_2 & -3x_3 & +4x_4 & -5x_5 & = 6 \\ & x_2 & -x_3 & +x_4 & -x_5 & = 1 \end{array}$$

$$x_j \geq 0 \quad \forall j.$$

- Kiểm tra tính tối ưu của vecto

$$x^* = (0, 0, 16, 31, 14)$$

đối với bài toán QHTT đã cho

Ví dụ

- Dễ dàng kiểm tra được rằng x^* là phương án chấp nhận được của bài toán đã cho:

$$A = [1 \ -4 \ 2 \ -5 \ 9; \ 0 \ 1 \ -3 \ 4 \ -5; \ 0 \ 1 \ -1 \ 1 \ -1]; \\ x = [0; 0; 16; 31; 14]; \ A*x$$

ans =

3

6

1

- Theo hệ quả, x^* là phương án tối ưu khi và chỉ khi hệ phương trình và bất phương trình sau đây là có nghiệm

Ví dụ

$$(y_1 + 2) x^*_1 = 0$$

$$(-4y_1 + y_2 + y_3 + 6) x^*_2 = 0$$

$$(2y_1 - 3y_2 - y_3 - 5) x^*_3 = 0$$

$$(-5y_1 + 4y_2 + y_3 + 1) x^*_4 = 0$$

$$(9y_1 - 5y_2 - y_3 + 4) x^*_5 = 0$$

$$y_1 \geq -2$$

$$-4y_1 + y_2 + y_3 \geq -6$$

$$2y_1 - 3y_2 - y_3 \geq 5$$

$$-5y_1 + 4y_2 + y_3 \geq -1$$

$$9y_1 - 5y_2 - y_3 \geq -4$$

$$x^*_1 = 0$$

$$x^*_2 = 0$$

$$x^*_3 = 16$$

$$x^*_4 = 31$$

$$x^*_5 = 14$$

Bài toán đối ngẫu

$$3y_1 + 6y_2 + y_3 \rightarrow \min$$

$$y_1 \geq -2$$

$$-4y_1 + y_2 + y_3 \geq -6$$

$$2y_1 - 3y_2 - y_3 \geq 5$$

$$-5y_1 + 4y_2 + y_3 \geq -1$$

$$9y_1 - 5y_2 - y_3 \geq -4$$

Ví dụ

- Hệ cuối cùng là tương đương với hệ sau:

$$\begin{aligned}y_1 &\geq -2 \\-4y_1 + y_2 + y_3 &\geq -6 \\2y_1 - 3y_2 - y_3 &= 5 \\-5y_1 + 4y_2 + y_3 &= -1 \\9y_1 - 5y_2 - y_3 &= -4\end{aligned}$$

- Hệ ba phương trình cuối cùng có nghiệm duy nhất $y^* = (-1, 1, -10)$.
 $(A=[2 \ -3 \ -1; -5 \ 4 \ 1; \ 9 \ -5 \ -1]; \ b=[5; -1; -4]; \ y=A\bslash b)$
- Dễ dàng kiểm tra được rằng y^* thoả mãn hai bất phương trình đầu. Do đó y^* là nghiệm của hệ phương trình và bất phương trình ở trên. Theo hệ quả, điều đó chứng tỏ x^* là phương án tối ưu của bài toán QHTT đặt ra.

Giải qui hoạch tuyến tính trên MATLAB

Hàm LINPROG

- MATLAB cung cấp hàm **linprog** để giải bài toán QHTT.
- Dưới đây là một số cách sử dụng hàm này
 - **X=LINPROG (f ,A ,b)**
 - **X=LINPROG (f ,A ,b ,Aeq ,beq)**
 - **X=LINPROG (f ,A ,b ,Aeq ,beq ,LB ,UB)**
 - **X=LINPROG (f ,A ,b ,Aeq ,beq ,LB ,UB ,X0)**
 - **X=LINPROG (f ,A ,b ,Aeq ,beq ,LB ,UB ,X0 ,OPTIONS)**
 - **[X ,FVAL]=LINPROG (. . .)**
 - **[X ,FVAL ,EXITFLAG] = LINPROG (. . .)**
 - **[X ,FVAL ,EXITFLAG ,OUTPUT] = LINPROG (. . .)**
 - **[X ,FVAL ,EXITFLAG ,OUTPUT ,LAMBDA]=LINPROG (. . .)**

Hàm LINPROG

- Lệnh **X=LINPROG (f ,A,b)** giải bài toán QHTT:

$$\min \{ f'x : Ax \leq b \}$$

- Lệnh **X=LINPROG (f ,A,b ,Aeq,beq)** giải bài toán có thêm ràng buộc cơ bản dạng đẳng thức **Aeq*x = beq**.
- Lệnh **X=LINPROG (f ,A,b ,Aeq,beq,LB,UB)** xác định cận dưới và cận trên cho các biến số **LB <= x <= UB**.
 - Gán **Aeq=[]** (**A=[]**) và **beq=[]** (**b=[]**) nếu không có ràng buộc này
 - Gán **LB** và **UB** là ma trận rỗng (**[]**) nếu không có các cận này.
 - Gán **LB(i)=-Inf** nếu **x(i)** không bị chặn dưới và gán **UB(i)=Inf** nếu **x(i)** không bị chặn trên.

Hàm LINPROG

- Lệnh **X=LINPROG (f ,A ,b ,Aeq ,beq ,LB ,UB ,X0)** để xác định thêm phương án xuất phát **X0**.
 - **Chú ý:** Lựa chọn này chỉ được chấp nhận nếu sử dụng *thuật toán tập tích cực*. Phương pháp ngầm định để giải là *thuật toán điểm trong* sẽ không chấp nhận điểm xuất phát.
- Lệnh **X=LINPROG (f ,A ,b ,Aeq ,beq ,LB ,UB ,X0 ,OPTIONS)** thực hiện giải với các thông số tối ưu được xác định bởi biến có cấu trúc **OPTIONS**, được tạo bởi hàm **OPTIMSET**.
 - Gán **option=optimset ('LargeScale','off','Simplex','on')** để chọn thuật toán đơn hình để giải bài toán.
 - Hãy gõ **help OPTIMSET** để biết chi tiết.

Hàm LINPROG

- Lệnh **[X , FVAL]=LINPROG (. . .)** trả lại thêm giá trị hàm mục tiêu tại phương án **X**: **FVAL = f' * X**.
- Lệnh **[X , FVAL , EXITFLAG]=LINPROG (. . .)** trả lại **EXITFLAG** mô tả điều kiện kết thúc của **LINPROG**. Các giá trị của **EXITFLAG** có ý nghĩa sau
 - 1 LINPROG hội tụ đến lời giải X.
 - 0 Đạt đến giới hạn số bước lặp.
 - -2 Không tìm được pacnd.
 - -3 Bài toán có hàm mục tiêu không bị chặn.
 - -4 giá trị **NaN** xuất hiện trong quá trình thực hiện thuật toán.
 - -5 Cả hai bài toán gốc và đối ngẫu đều không tương thích.
 - -7 Hướng tìm kiếm quá nhỏ, không thể cải thiện được nữa.

Hàm LINPROG

- Lệnh **[X, FVAL, EXITFLAG, OUTPUT] = LINPROG(. . .)** trả lại biến cấu trúc **OUTPUT** với
 - **OUTPUT.iterations** - số bước lặp phải thực hiện
 - **OUTPUT.algorithm** - thuật toán được sử dụng
 - **OUTPUT.message** – thông báo
- Lệnh **[X, FVAL, EXITFLAG, OUTPUT, LAMBDA] = LINPROG(. . .)** trả lại nhân tử Lagrangian LAMBDA , tương ứng với lời giải tối ưu:
 - **LAMBDA.ineqlin** – tương ứng với ràng buộc bất đẳng thức A,
 - **LAMBDA.eqlin** – tương ứng với ràng buộc đẳng thức Aeq,
 - **LAMBDA.lower** – tương ứng với LB,
 - **LAMBDA.upper** – tương ứng với UB.

Ví dụ

- Giải bài toán qui hoạch tuyến tính:

$$\begin{aligned} 2x_1 + x_2 + 3x_3 &\rightarrow \min \\ x_1 + x_2 + x_3 + x_4 + x_5 &= 5 \\ x_1 + x_2 + 2x_3 + 2x_4 + 2x_5 &= 8 \\ x_1 + x_2 &= 2 \\ &x_3 + x_4 + x_5 = 3 \\ x_1, x_2, x_3, x_4, x_5 &\geq 0 \end{aligned}$$

- `f=[2 1 3 0 0]; beq=[5; 8; 2; 3];`
- `Aeq=[1 1 1 1 1; 1 1 2 2 2; 1 1 0 0 0; 0 0 1 1 1];`
- `A=[]; b=[]; LB=[0 0 0 0 0]; UB=[]; x0=[];`
- `[X,FVAL,EXITFLAG,OUTPUT,LAMBDA]=linprog(f,A,b,Aeq,b
eq,LB,UB,x0)`

Kết quả

• **X =**

0.0000

2.0000

0.0000

1.5000

1.5000

• **FVAL =**

2.0000

• **EXITFLAG =**

1

• **OUTPUT =**

iterations: 5

algorithm: 'large-scale: interior point'

cgitations: 0

message: 'Optimization terminated.'

• **LAMBDA =**

ineqlin: [0x1 double]

eqlin: [4x1 double]

upper: [5x1 double]

lower: [5x1 double]

Ví dụ

- Sử dụng thuật toán đơn hình:

```
opt=optimset('LargeScale','off','Simplex','on')
```

```
[x,fval,exitflag,output]=linprog(f,A,b,Aeq,beq,LB,UB,x0,opt)
```

ta thu được kết quả:

- **x = [0 2 0 3 0]**

- **fval = 2**

- **exitflag = 1**

- **output =**

```
iterations: 1
```

```
algorithm: 'medium scale: simplex'
```

```
cgiterations: []
```

```
message: 'Optimization terminated.'
```