

BÀI THỰC HÀNH SỐ 0

LÀM QUEN VỚI PHẦN MỀM MATLAB

Mục đích của bài thực hành này là :

1. Giới thiệu tổng quan cho sinh viên phần mềm tính toán số Matlab.
2. Thực hành thao tác dùng Matlab để giải quyết các bài toán hóa học đơn giản.


1 Giới thiệu :

MATLAB là một phần mềm tính toán số và ma trận, được sử dụng rộng rãi cho các khối ngành kỹ thuật (cơ, điện-điện tử, xây dựng và kỹ thuật hóa học...). Matlab được tạo thành từ một mô-đun tổng quát chứa các hàm cơ bản và các hộp công cụ riêng (Toolbox)+Simulink.

Trong bài thực hành này, phần mềm Matlab được cài đặt trên máy tính với hệ điều hành thông thường (Microsoft Windows XP...).

2 Môi trường làm việc

Để xác nhận cấu hình Matlab cài đặt, cần khởi động Matlab bằng một trong các cách sau

- Nhấp chuột vào biểu tượng  MATLAB
- Hoặc từ menu khởi động START của Windows.

Từ cửa sổ làm việc (Command Window) vừa khởi động, nhập vào

```
>> help
```

Các hộp công cụ có sẵn sẽ hiển thị trên màn hình. Tiếp theo chọn mục mà ta cần sự giúp đỡ. Ngoài ra, Matlab còn cung cấp sự trợ giúp trực tuyến (online). Để truy xuất vào các thông tin trợ giúp của một hàm/lệnh nào đó, chỉ cần nhập vào:

```
>> help TÊN_CỦA_HÀM
```

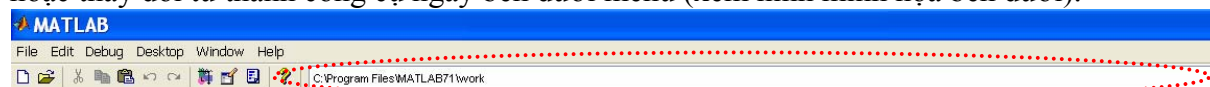
Để nhận biết thư mục làm việc hiện hành (nơi mà chúng ta sẽ lưu giữ các file làm việc tạo ra), cần nhập vào:

```
>> pwd
```

Để thay đổi thư mục làm việc, nhập vào dòng lệnh sau

```
>> cd('directory')
```

hoặc thay đổi từ thanh công cụ ngay bên dưới menu (xem hình minh họa bên dưới).



Để nhận biết vị trí thư mục của một hàm nào đó, nhập vào

```
>> which TÊN_CỦA_HÀM
```

3 Các hàm cơ sở

3.1 Quản lý dữ liệu

Bài thực hành môn học **Mô hình hóa, mô phỏng và tối ưu hóa các quá trình hóa học**

Nhìn chung, mọi dữ liệu cơ bản được lưu trữ dưới dạng ma trận.

Thông thường, tiếp sau một sự khai báo ma trận hoặc thực thi một hàm, kết quả sẽ được hiển thị nếu không có dấu chấm phẩy “;” ở cuối câu lệnh.

Một ma trận được khai báo giữa hai dấu ngoặc vuông “[...]”, các khoảng trống phân chia các cột và dấu chấm phẩy “;” phân chia các dòng. Ví dụ:

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

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

A =

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

Một vector có thể được định nghĩa bởi phần tử đầu tiên, bước tăng và phần tử cuối cùng của nó. Ví dụ:

```
>> v=1:0.5:2.5
```

v =

```
1.0000 1.5000 2.0000 2.5000
```

Chú ý: nếu không chỉ rõ bước tăng, Matlab lấy giá trị mặc định là 1.

Có thể chọn một phần của ma trận bằng cách chỉ ra giữa hai dấu ngoặc đơn “(...)” tọa độ vị trí của các phần tử cần lấy. Ví dụ:

```
>> matrancon=A(2:3,2:3)
```

matrancon =

```
5 6
8 9
```

Lệnh trên cho phép lưu giữ trong *matrancon* ma trận được tạo thành từ các phần tử ở các vị trí (2,2), (2,3), (3,2) và (3,3) của ma trận A.

Các hàm cơ bản cho tính toán ma trận là có sẵn trong thư viện *matfun*. Ví dụ các giá trị riêng của một ma trận vuông được tính bằng cách sử dụng hàm *eig(...)*:

```
>> eig(A)
```

ans =

```
16.1168
-1.1168
-0.0000
```

Chúng ta sẽ có thể bằng cách tương tự, tính định thức, chuẩn, kernel, vết, vector riêng... của một ma trận.

Để nhận biết kích thước của một ma trận, nhập vào :

```
>> dim=size(A)
```

dim =

3 3

Lệnh size trên sẽ trả về kích thước của ma trận A và lưu giữ kết quả trong biến vector dim.

Để chuyển vị một ma trận dùng kí hiệu «'», ví dụ nhập vào :

```
>> transpose=A'
```

transpose =

**1 4 7
2 5 8
3 6 9**

Chú ý : «'» sẽ trả về ma trận chuyển vị liên hợp phức trong trường hợp các ma trận chứa các hệ số phức.

Biến vector ans mặc định của Matlab sẽ ghi nhớ kết quả tính toán cuối cùng không được lưu lại.

3.2 Thao tác với đa thức :

3.2.1 Biểu diễn đa thức với Matlab :

Một đa thức được biểu diễn với Matlab dưới dạng vector. Điều đó có nghĩa rằng chúng ta kết hợp đa thức cần biểu diễn với một vector chỉ chứa các hệ số theo chiều giảm của số mũ.

Ví dụ chúng ta có đa thức $P(x)$ như sau :

$$P(x) = x^5 + 4x^4 - 2x^3 + x + 7$$

Khai báo $P(x)$ với Matlab chỉ đơn giản khai báo một vector chứa các hệ số của $P(x)$ theo chiều giảm của số mũ :

```
>> P=[1 4 -2 0 1 7]
```

P =

1 4 -2 0 1 7

Một cách khác để biểu diễn đa thức là định nghĩa từ các nghiệm của nó. Như vậy, chúng ta sử dụng hàm *poly*. Ví dụ một đa thức $Q(x)$ có các nghiệm -1 và 2, được định nghĩa $Q(x) = (x+1)(x-2) = x^2 - x - 2$. Với Matlab, chúng ta khai báo :

```
>> Q=poly([-1,2])
```

Q =

1 -1 -2

Chú ý : đa thức cho bởi Matlab với lệnh *poly* là đa thức mà hệ số của số mũ cao nhất là đơn vị.

3.2.2 Nghiệm của một đa thức :

Nghiệm của một đa thức nhận được nhờ hàm *roots*. Chẳng hạn nghiệm của đa thức $Q(x)$ bên trên được tính :

```
>> solutions=roots(Q)
```

solutions =

**2
-1**

Chú ý : hàm *fsolve* cho phép tìm nghiệm của một phương trình phi tuyến.

3.2.3 Phép nhân và nhân tử hóa các đa thức

Phép nhân và nhân tử hóa các đa thức được thực hiện bằng các hàm *conv* (convolution) và *deconv* (deconvolution). Nếu chúng ta muốn nhân hai đa thức $P_1(x) = x^2 + 4x - 12$ và $P_2(x) = x^3 - 8$, chúng ta viết :

```
>> P1=[1 4 -12];  
>> P2=[1 0 0 -8];  
>> P=conv(P1,P2)
```

P =

1 4 -12 -8 -32 96

Như vậy, đa thức kết quả sẽ là $P(x) = x^5 + 4x^4 - 12x^3 - 8x^2 - 32x + 96$. Một cách tương tự, nếu chúng ta muốn nhân tử hóa $P(x)$ bằng $P_1(x)$, chúng ta sử dụng các lệnh sau đây :

```
>> P2p=deconv(P,P1);
```

P2p =

1 0 0 -8

Rõ ràng rằng $P2p = P_2(x)$

3.3. Thao tác với ma trận

Các hàm xử lý ma trận là nằm trong thư viện *matfun*. Chúng ta chỉ đề cập ở đây những hàm được sử dụng phổ biến.

3.3.1 Hàm norm, rank, det

Bài thực hành môn học **Mô hình hóa, mô phỏng và tối ưu hóa các quá trình hóa học**

Các hàm norm, rank, det cho phép một cách tương ứng tính chuẩn, hạng và định thức của một ma trận.

3.3.2 Hàm inv, pinv, cond

Hàm inv (pinv) cho phép nghịch đảo các ma trận vuông (các ma trận không vuông).

Hàm cond cho phép xác định điều kiện của một ma trận. Số điều kiện phản ánh độ “khó khăn” khi nghịch đảo ma trận. Đó là tỉ số giữa “giá trị suy giảm” (singular value) lớn nhất và nhỏ nhất.

3.3.3 Hàm eig, svd

Các hàm này cho phép tính giá trị riêng và giá trị suy giảm của một ma trận vuông.

3.4 Hiện thị - đồ họa

MATLAB có các hàm cho phép biểu diễn đồ thị 1D, 2D và 3D các kết quả.

Chúng ta có thể biểu diễn hàm exp theo thời gian giữa 0 và 10 giây. Chúng ta làm theo các bước:

B1: Định nghĩa một vector thời gian

```
>> t=0:0.1:1;
```

Vector thời gian t có 11 thành phần.

B2: Tính giá trị của hàm exp tại các thành phần thời gian t

```
>> y=exp(t);
```

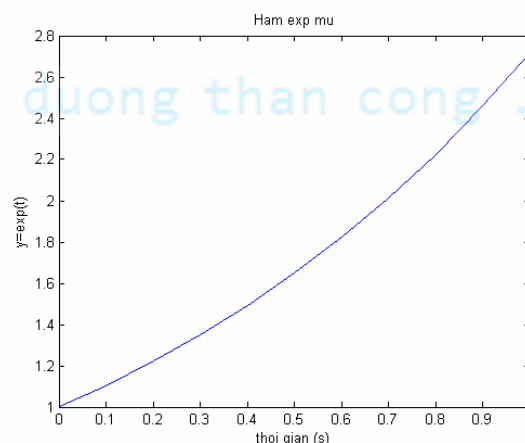
Vector y có 11 thành phần.

B3: Vẽ hàm y phụ thuộc vào t theo lệnh

```
>> plot(t,y);
```

Khi thực hiện lệnh vẽ, MATLAB sẽ tạo ra một khung hình (figure). Để làm việc với các khung hình, chúng ta có thể sử dụng các lệnh sau:

```
>> grid; % Tạo lưới  
>> xlabel('thời gian (s)'); % Tên trục x  
>> ylabel('exp(t)'); % Tên trục y  
>> title('Hàm e mu'); % Tên đồ thị
```



Các khung hình được đánh số lần lượt. Tuy nhiên chúng ta có thể chọn lựa một khung hình bằng cách nhập vào :

```
>> figure(1);
```

Tương tự, nếu chúng ta muốn làm việc trên một khung hình đã mở, phải sử dụng lệnh hold on. Lệnh này cho phép giữ lại hình ảnh đã có khi thực hiện một lệnh vẽ mới trên đó. Ngược lại ta có lệnh hold off. Một số hàm sau đây cũng có thể được sử dụng :

```
>> clf;           % Xóa khung hình
>> close all;     % Đóng các khung hình
>> subplot(221);
```

3.5 Các lệnh xử lý dữ liệu:

3.5.1 Hàm polyfit

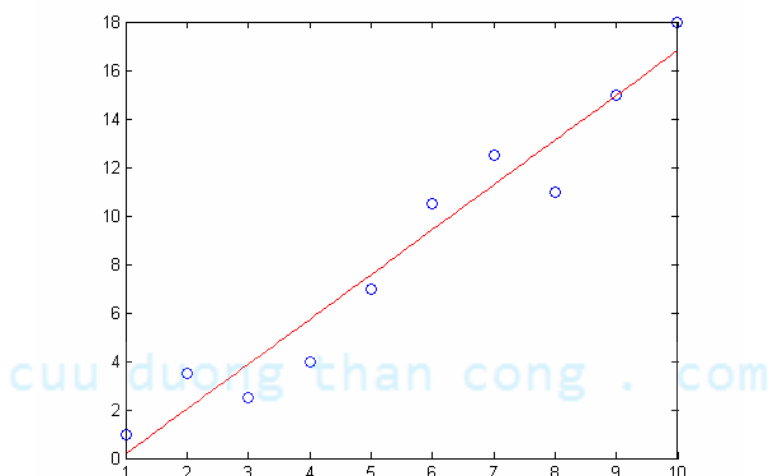
Cho phép « xấp xỉ » một tập dữ liệu/một hàm theo nghĩa bình phương cực tiểu.

Ví dụ chúng ta có một tập dữ liệu sau :

x	1	2	3	4	5	6	7	8	9	10
y	1	3.5	2.5	4	7	10.5	12.5	11	15	18

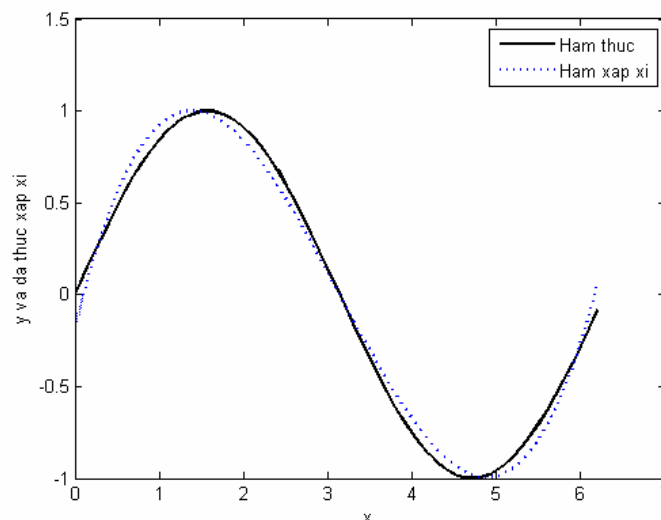
Khai báo tập dữ liệu trên với Matlab :

```
>> x = (1: 10)';
>> y = [1 3.5 2.5 4 7 10.5 12.5 11 15 18]';
>> result = polyfit(x,y,1); %tính các hệ số a, b của đường thẳng xấp xỉ
```



Ví dụ xấp xỉ một hàm sin:

```
>> x = (0: 0.1: 2*pi)'; % tạo vector x
>> y = sin(x);           % tính giá trị hàm sin tại các điểm của vector x
>> p = polyfit(x,y,3); % cho đa thức xấp xỉ bậc 3 của y theo nghĩa bình phương cực tiểu
>> f = polyval(p,x); % tính giá trị của đa thức xấp xỉ tại các điểm x
>> plot(x,y,'r',x,f,'b') % biểu diễn đồ thị so sánh
```



3.5.2 Hàm interp1

Cho phép nội suy các giá trị từ tập một dữ liệu (xdata,ydata) bằng các phương pháp tuyến tính, Spline... Giả sử với tập dữ liệu (x,y) trên đây, làm thế nào để tính giá trị của y tại x=5.5?

```
>> xi = 5.5;  
>> yi = interp1(x,y,xi); % phương pháp tuyến tính mặc định được chọn  
>> plot(x,y,'o',xi,yi,'r*');
```

3.6 Hàm giải phương trình vi phân thường

Các hàm ode45, ode23,... cho phép giải (hệ) phương trình vi phân thường ODE.

Xem chi tiết cấu trúc của lệnh và các ví dụ trong phần help của matlab bằng cách nhập vào >> **help ode45**

3.7 Lập trình với MATLAB

Trong phần này chúng ta sẽ làm quen với các câu lệnh lập trình cơ bản với MATLAB. Chú ý, cho các vòng lặp các chỉ số được sử dụng không nên trùng với các biến nội được định nghĩa trước bởi Matlab: ví dụ biến “i” tương ứng với căn bậc hai của -1.

3.7.1 Lệnh if

Trong vòng “if” một điều kiện sẽ được kiểm tra. Nếu điều kiện này đúng (giá trị logic 1 hoặc TRUE), chương trình sẽ đi vào và thực thi lệnh trong vòng “if”. Ngược lại sẽ không thực hiện.

Cú pháp chung của một vòng lệnh “if” là:

```
IF expression  
    statements  
ELSEIF expression  
    statements  
ELSE
```

```
statements  
END
```

3.7.2 Vòng lặp for

Thực hiện các lệnh với số vòng lặp xác định. Cú pháp của vòng lặp for:

```
for variable = expression  
statements  
end
```

3.7.3 Vòng lặp while

Thực hiện các lệnh trong khi điều kiện là còn đúng. Cú pháp của vòng lặp while:

```
while expression  
statements  
end
```

3.7.4 Lệnh break, return và keyboard

Lệnh break cho phép kết thúc một vòng lặp while hoặc for. Lệnh return cho phép thoát ra từ một hàm. Khi chúng ta muốn kiểm tra chương trình (gỡ lỗi), chúng ta sử dụng hàm keyboard.

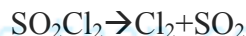
3.8 Bài tập thực hành

Bài tập 1: Giải phương trình tuyến tính/phi tuyến sau với độ chính xác tương đối 10^{-4} dùng Matlab (Trang 62, sách Ứng Dụng Tin Học Trong CNHH-Thực Phẩm, Trịnh Văn Dũng):

- a) $x^5 + 5x - 2 = 0, x \in [0, 1]$
- b) $\ln(8x) - x - 0.5 = 0$
- c) $\frac{dy}{dt} = \sin(t)(2y(t)t - 1) + 3, y(t=0) = 0$

Bài tập 2: (Bài tập 2.5, trang 58, sách Kỹ Thuật Phản Ứng, Tập 4, Vũ Bá Minh)

D.F Smith đã nghiên cứu phản ứng phân hủy pha khí của clorur sulfur, SO_2Cl_2 thành khí Clor và SO_2 tại $279,2^\circ\text{C}$:



Ở điều kiện thể tích của hỗn hợp không đổi, theo dõi áp suất tổng cộng theo thời gian phản ứng được kết quả sau:

t, ph	3,4	15,7	28,1	41,1	54,5	68,3	82,4	96,3
P_t, mmHg	325	335	345	355	365	375	385	395

Dùng Matlab, biểu diễn các cặp điểm (t, P_t) trên đồ thị. Quan hệ này là tuyến tính? Kết luận về bậc của phản ứng?

Xác định phương trình vận tốc phản ứng bằng đồ thị? Từ đây, có thể nói gì về độ chuyển hóa ở thời điểm vô cực.

Bài thực hành môn học **Mô hình hóa, mô phỏng và tối ưu hóa các quá trình hóa học**

Nội suy giá trị của P_t tại $t=20\text{ ph}$, 48 ph và 70 ph ?

Bài tập 3: Lập trình với Matlab

Tạo các tập file.m (vào menu, chọn **File**, chọn **New File**, chọn **File.m**) để giải các yêu cầu sau dùng Matlab (mỗi yêu cầu tạo một file và lưu nó với tên exo1.m, exo2.m...)

Yêu cầu 1 : Viết chương trình tính $S = \sum_{k=0}^n k$ với $n=5, 20$ và 50 ;

Yêu cầu 2 : Viết chương trình tính $S = \sum_{k=0}^n \frac{1}{k!}$ với $n=5, 20$ và 50 ; (so sánh kết quả với e^1)

Yêu cầu 3 : Viết chương trình cho phép hiển thị đồ họa hàm số sau :

$$f(x) = \text{sign}(x) = \begin{cases} -1 & \text{if } x < 0 \\ 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \end{cases}$$

Bài tập 4: Lập trình giải hệ tuyến tính (Khuyến khích)

Áp dụng các lệnh được giới thiệu trong bài thực hành để giải quyết các bài tập sau:

- Tạo một hàm cho phép tìm nghiệm của một hệ tuyến tính $Ax = b$ bằng phương pháp Gauss. Đầu vào là các ma trận A và b , đầu ra là các nghiệm cần tìm. Hãy hiện thực chương trình bằng Matlab.
- So sánh với nghiệm tìm được bằng lệnh $x = \text{inv}(A) * b$ (giả sử A khả nghịch)

Bài tập 5: (Khuyến khích) Sinh viên tự tham khảo các tài liệu khác để có các bài tập và dùng Matlab để giải quyết.