CHƯƠNG 1. HƯỚNG DẪN SỬ DỤNG MATLAB VÀ SIMULINK

Sau khi học xong bài này, sinh viên có thể:

- Sử dụng phần mềm MATLAB.
- Thực hiện tạo các script file hay function và lưu trữ trên MATLAB.
- Biết được các công cụ cơ bản tạo mô hình bằng Simulink trên MATLAB.

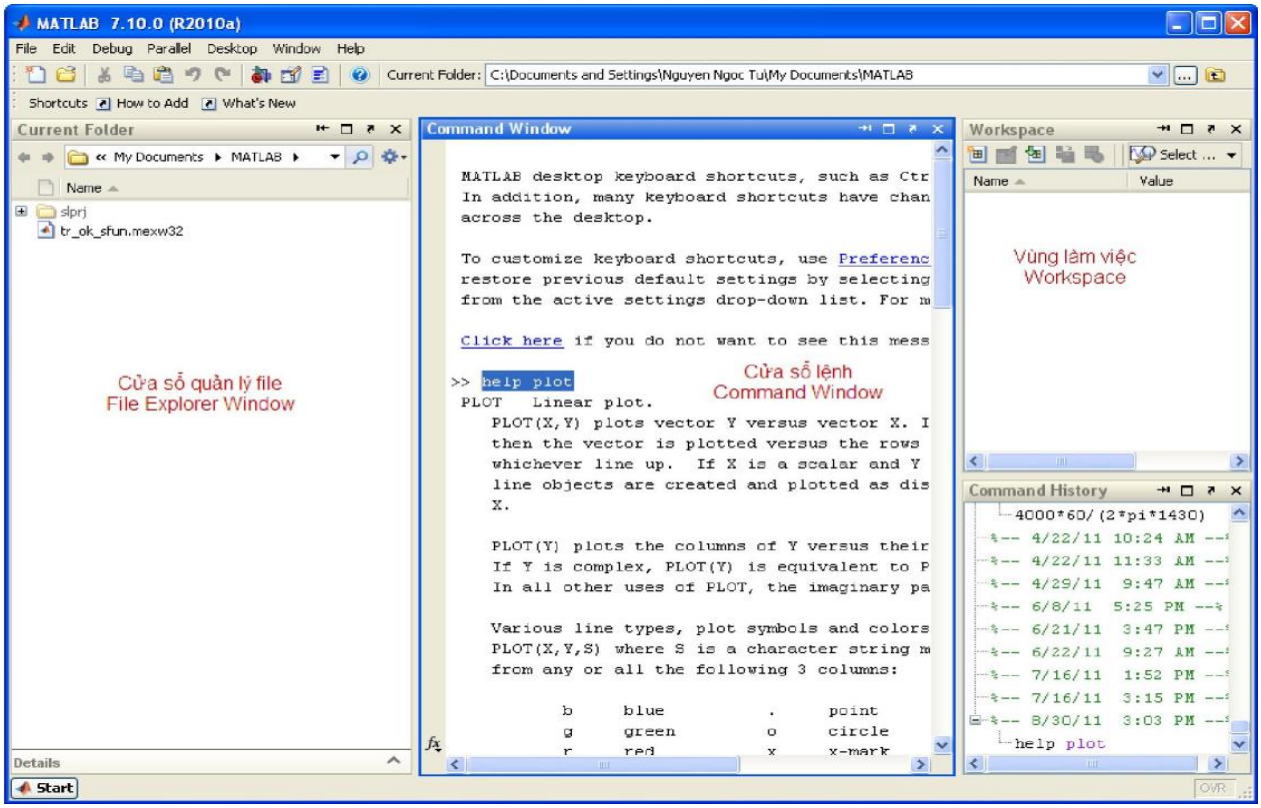
1.1 Hướng dẫn sử dụng Matlab

MATLAB (**Mat**rix **Lab**oratory) là một phần mềm khoa học được thiết kế để cung cấp việc tính toán số và hiển thị đồ họa bằng ngôn ngữ lập trình cấp cao. MATLAB cung cấp các tính năng tương tác tuyệt vời cho phép người sử dụng thao tác dữ liệu linh hoạt dưới dạng ma trận để tính toán và quan sát. Các dữ liệu vào của MATLAB có thể được nhập từ "Command line" hoặc từ "m files", trong đó tập lệnh được cho trước bởi MATLAB.

MATLAB cung cấp cho người dùng các toolbox tiêu chuẩn tùy chọn. Người dùng cũng có thể tạo ra các hộp công cụ riêng của mình gồm các "mfiles" được viết cho các ứng dụng cụ thể.

Chúng ta có thể sử dụng các tập tin trợ giúp của MATLAB cho các chức năng và các lệnh liên quan với các toolbox có sẵn (dùng lệnh help).
Ví dụ: Command Window: >> **help** plot

Màn hình tiêu chuẩn sau khi khởi động Matlab:



Vị trí của các cửa sổ con như workspace, current folder, command window được hiển thị từ khi chạy MATLAB và có thể di chuyển thay đổi vị trí cũng như đóng mở theo ý muốn người sử dụng.

Command Window: Đây là cửa sổ làm việc chính của MATLAB. Tại đây ta thực hiện tòan bộ việc nhập dữ liệu và xuất kết quả tính tóan. Dấu nháy >> báo hiệu chương trình sẵn sàng cho việc nhập dữ liệu. Ta kết thúc việc nhập dữ liệu bằng cách nhấn phím Enter. MATLAB sẽ thực thi dòng lệnh mà ta nhập vào *Command Window* và trả kết quả trong *Command Window.*

Command History: Lưu lại tất cả các lệnh mà ta đã nhập vào trong *Command Window*. Ta có thể xem lại tất cả các lệnh bằng cách dùng scroll bar, hay thực hiện lại lệnh đó bằng cách nhấp kép lên dòng lệnh. Ngòai ra ta còn có thể cut, paste, delete các lệnh.

Workspace browser: trong MATLAB các dữ liệu được lưu trong biến. *Workspace browser* liệt kê tất cả các biến mà ta đang sử dụng trong MATLAB. Nó cung cấp thông tin về kích thước, loại dữ liệu. Ta có thể truy cập trực tiếp vào dữ liệu bằng cách nhấn kép vào biến để hiển thị *Array editor*.

Launch pad: cho phép người dùng truy cập nhanh vào các bộ *Toolbox*, phần *Help*.

Editor window: dùng để sọan thảo và debug các M-file của MATLAB.

Current Directory: xem các file trong thư mục hiện hành. Lưu ý, khi thực thi (run) file script hay gọi hàm mà không đưa đường dẫn chi tiết thì các file script hay hàm phải được để ở trong thư mục hiện hành.

Một số thao tác cơ bản trong MATLAB

Trong MATLAB, thanh trình đơn thay đổi tùy theo cửa sổ mà ta lựa chọn. Tuy vậy các trình đơn *File, Desktop, Window, Help* có mặt hầu hết trong các thanh trình đơn.

- Trình đơn *File*:

- *New:* tạo một đối tượng mới (biến, m-file, figure, model, GUI).
- *Open:* mở một file theo định dạng của MATLAB (*.m, *.mat, *.mdl)
- *Import data…*: nhập dữ liệu từ các file khác vào MATLAB.
- *Save workspace…*: lưu các biến trong MATLAB vào file *.mat.
- *Set path:* khai báo các đường dẫn của các thư mục chứa các m-file.
- *Preferences:* thay đổi các định dạng về font, font size, color cũng như các tùy chọn cho Editor, Command Window v.v.
- *Page Setup:* định dạng trang in.
- *Print:* in.

- Trình đơn *Desktop*:

- *Desktop layout:* sắp xếp các cửa sổ trong giao diện.
- *Save layout:* lưu cách sắp xếp cửa sổ.

(trong các version mới hơn các chức năng này được tổ chức trong **Home**).

1.1.1 Khai báo biến, các phép toán và toán tử

Khai báo biến:

- Phân biệt chữ hoa và chữ thường.
- Không cần phải khai báo kiểu biến.
- Tên biến phải bắt đầu bằng ký tự và không được có khoảng trắng.
- Không đặt tên trùng với các tên đặc biệt của MATLAB.
- Để khai báo biến toàn cục (sử dụng được trong tất cả chương trình con), phải dùng thêm từ khoá ***global*** phía trước

Các phép toán: + , - , * , / , \ (chia trái) , ^ (mũ) , **'** (chuyển vị hay số phức liên hiệp).

Các toán tử quan hệ : < , <= , > , >= , == , ~=

Các toán tử logic : & , | (or) , ~ (not)

Các hằng:

- pi 3.14159265
- i, j: số ảo.
- inf: vô cùng.
- NaN: không phải là số

Chú ý :

- Các lệnh kết thúc bằng dấu chấm phẩy, MATLAB sẽ không thể hiện kết quả trên màn hình.
- Các chú thích được đặt phía sau dấu %.
- Trong quá trình nhập nếu các phần tử trên một hàng dài quá ta có thể xuống dòng bằng toán tử ba chấm (. . .)

1.1.2 Thao tác với ma trận

Ma trận là một mảng có m hàng và n cột. Trường hợp ma trận chỉ có một phần tử (ma trận 1x1) ta có một số. Ma trận chỉ có một cột hay một hàng được gọi là một vector.

- Khi nhập ma trận ta phải tuân theo các quy định sau:
- Ngăn cách các phần tử trong một hàng của ma trận bằng dấu "," hay khoảng trắng.
- Dùng dấu ";" để kết thúc một hàng và bắt đầu hàng kế tiếp.
- Các phần tử của ma trận phải để trong cặp dấu ngoặc vuông [ ].
- Phần tử ở hàng i cột j của ma trận có ký hiệu là A(i,j). Lưu ý rằng các chỉ số của ma trận thường bắt đầu từ 1.
- Toán tử '**:**'

Toán tử ":" là một toán tử quan trọng của MATLAB. Nó xuất hiện ở nhiều dạng khác nhau. Biểu thức 1:10 là một vector hàng chứa 10 số nguyên từ 1 đến 10.

```
>>1:10

>>100:-7:50 %tạo dãy số từ 100 đến 51, cách đều nhau 7

>>0: pi/4: pi %tạo một dãy số từ 0 đến π, cách đều nhau π/4
```

Các biểu thức chỉ số có thể tham chiếu tới một phần của ma trận. A(1:k,j) xác định k phần tử đầu tiên của cột j. Ngoài ra toán tử ":" tham chiếu tới tất cả các phần tử của một hàng hay một cột.

```
>>A(:,3)

>>A(3, :)

>>B = A(:, [1 3 2 4]) %tạo ma trận B từ ma trận A bằng cách đổi thứ tự các cột từ
                       [1 2 3 4] thành [1 3 2 4]

>>b(:, 2) = [] ; %xoá cột thứ 2
```

MATLAB cung cấp một số hàm để tạo các ma trận cơ bản:

- ***zeros*** tạo ra ma trận mà các phần tử đều là 0.
- ***ones*** tạo ra ma trận mà các phần tử đều là 1.
- ***rand*** tạo ra ma trận mà các phần tử ngẫu nhiên phân bố đều.
- ***randn*** tạo ra ma trận mà các phần tử ngẫu nhiên phân bố chuẩn.

1.1.3 Xử lý ma trận

Cộng : X = A + B

Trừ : X = A – B

Nhân : X = A*B

X = A.*B nhân các phần tử tương ứng với nhau, yêu cầu 2 ma trận A và B phải có cùng kích thước.

Chia : X = A/B lúc đó X = A * inv(B)

X = A\B lúc đó X = inv(A) * B

X=A./B chia các phần tử tương ứng với nhau, 2 ma trận A và B có cùng kích thước.

Luỹ thừa : X = A^2

X = A.^2

1.1.4 Đồ họa

MATLAB cung cấp một loạt hàm để vẽ biểu diễn các vector cũng như giải thích và in các đường cong này.

- **plot**: đồ họa 2-D với số liệu 2 trục vô hướng và tuyến tính.
- **plot3**: đồ họa 3-D với số liệu 2 trục vô hướng và tuyến tính
- **loglog**: đồ hoạ với các trục x, y ở dạng logarit
- **semilogx**: đồ hoạ với trục x logarit và trục y tuyến tính
- **semilogy**: đồ hoạ với trục y logarit và trục x tuyến tính

1.1.4.1 Tạo hình vẽ

Hàm plot có các dạng khác nhau phụ thuộc vào các đối số đưa vào. Ví dụ nếu y là một vector thì plot(y) tạo ra một đường quan hệ giữa các giá trị của y và chỉ số của nó. Nếu ta có 2 vector x và y thì plot(x,y) tạo ra đồ thị quan hệ giữa x và y.

```
>>t = [0:pi/100:2*pi];
>>y = sin(t);
>>plot(t,y); % Vẽ hình sin từ 0->2π
```

Ta có thể dùng các kiểu đường vẽ khác nhau khi vẽ hình.

```
>>plot(t,y, '. ' ) % vẽ bằng đường chấm chấm
```

Các dạng đường thẳng xác định bằng:

'**-**' đường liền; '**--**'đường đứt nét; '**:**' đường chấm chấm; '**-.**' đường chấm gạch

1.1.4.2 Thao tác với các trục tọa độ

Khi ta tạo một hình vẽ, MATLAB tự động chọn các giới hạn trên trục toạ độ và khoảng cách đánh dấu dựa trên dữ liệu dùng để vẽ.

- Mô tả lại phạm vi giá trị trên trục và khoảng cách đánh dấu theo ý riêng. Ta có thể dùng các hàm sau:

axis đặt lại các giá trị trên trục toạ độ.

axes tạo một trục toạ độ mới với các đặc tính được mô tả.

- Ghi nhãn lên các trục toạ độ

MATLAB cung cấp các lệnh ghi nhãn lên đồ hoạ gồm:

title thêm nhãn vào đồ hoạ.

xlabel thêm nhãn vào trục x.

ylabel thêm nhãn vào trục y.

zlabel thêm nhãn vào trục z.

l***egend*** thêm chú giải vào đồ thị.

text hiển thị chuỗi văn bản ở vị trí nhất định.

1.1.5 Lập trình trong MATLAB

- Các phát biểu điều kiện **if, else, elseif**
  Cú pháp của if:
  if <biểu thức điều kiện>

  <phát biểu>

  End

Nếu <biểu thức điều kiện> cho kết quả đúng thì phần lệnh trong thân của if được thực hiện. Các phát biểu else và elseif cũng tương tự.

- Phát biểu **Switch**
  Cú pháp của switch như sau:

  switch <biểu thức>

  case n1

  <lệnh 1>

  case n2

  <lệnh 2>

  . . . . . .

  case nn

  <lệnh n>

  Otherwise

  <lệnh n+1>

  end

- Phát biểu **While**
  Vòng lặp while dùng khi không biết trước số lần lặp. Cú pháp của nó như sau:

  while <biểu thức>

  <phát biểu>

  end

- Phát biểu **For**
  Vòng lặp for dùng khi biết trước số lần lặp. Cú pháp như sau:
  for <chỉ số>=<giá trị đầu>:<mức tăng>:<giá trị cuối>
  <phát biểu>
  end

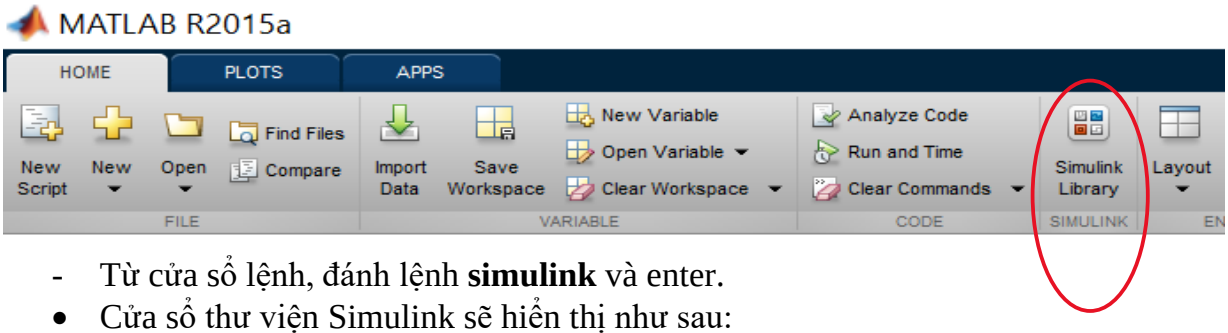
- Phát biểu **Break:**

Phát biểu break để kết thúc vòng lặp for hay while mà không quan tâm đến điều kiện kết thúc vòng lặp đã thoả mãn hay chưa.

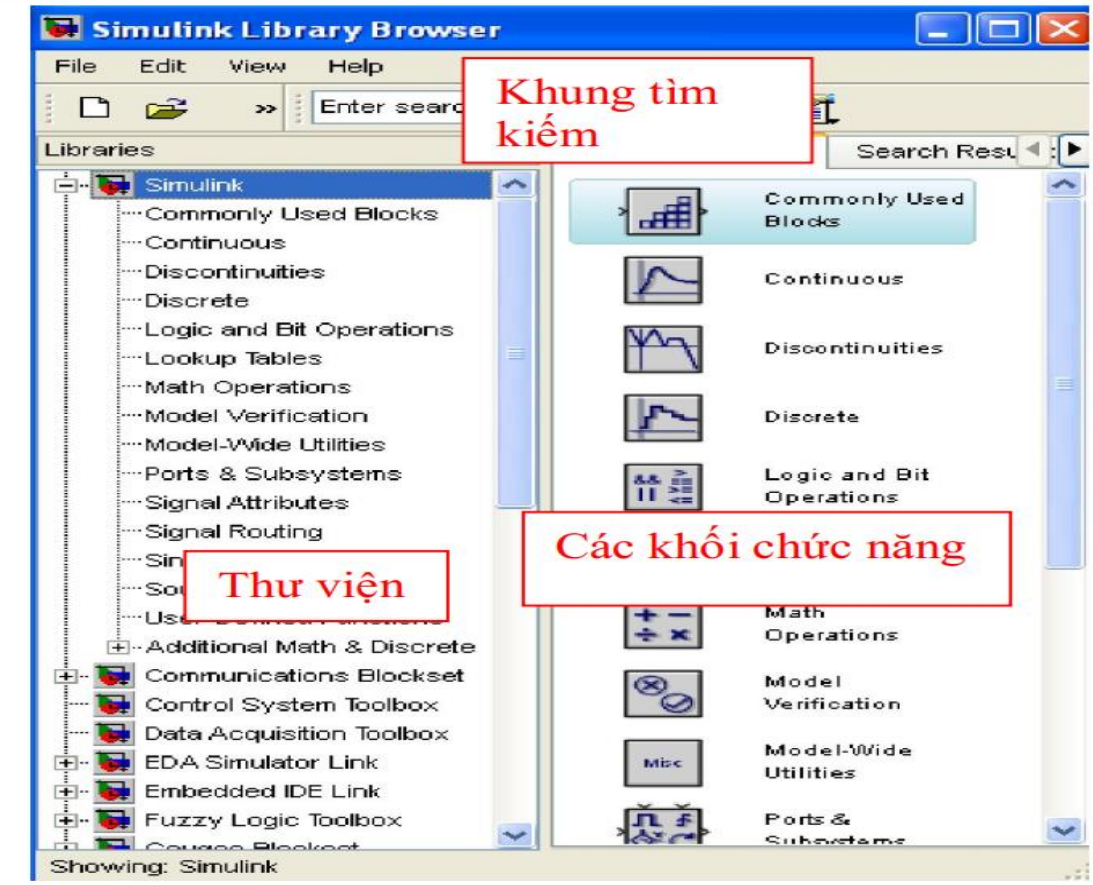
1.2 Hướng dẫn sử dụng Simulink

Simulink là một công cụ trong Matlab dùng để tạo mô hình, mô phỏng và phân tích các hệ thống động với môi trường giao diện sử dụng bằng đồ họa. Việc xây dựng mô hình được đơn giản hóa bằng các hoạt động nhấp chuột và kéo thả. Simulink bao gồm một bộ thư viện khối với các hộp công cụ toàn diện cho cả việc phân tích tuyến tính và phi tuyến. Simulink là một phần quan trọng của Matlab và có thể dễ dàng chuyển đổi qua lại trong quá trình phân tích, và vì vậy người dùng có thể tận dụng được ưu thế của cả hai môi trường.

- Có thể mở Simulink bằng 2 cách:
- Click vào biểu tượng Simulink library như hình dưới (Simulink icon).

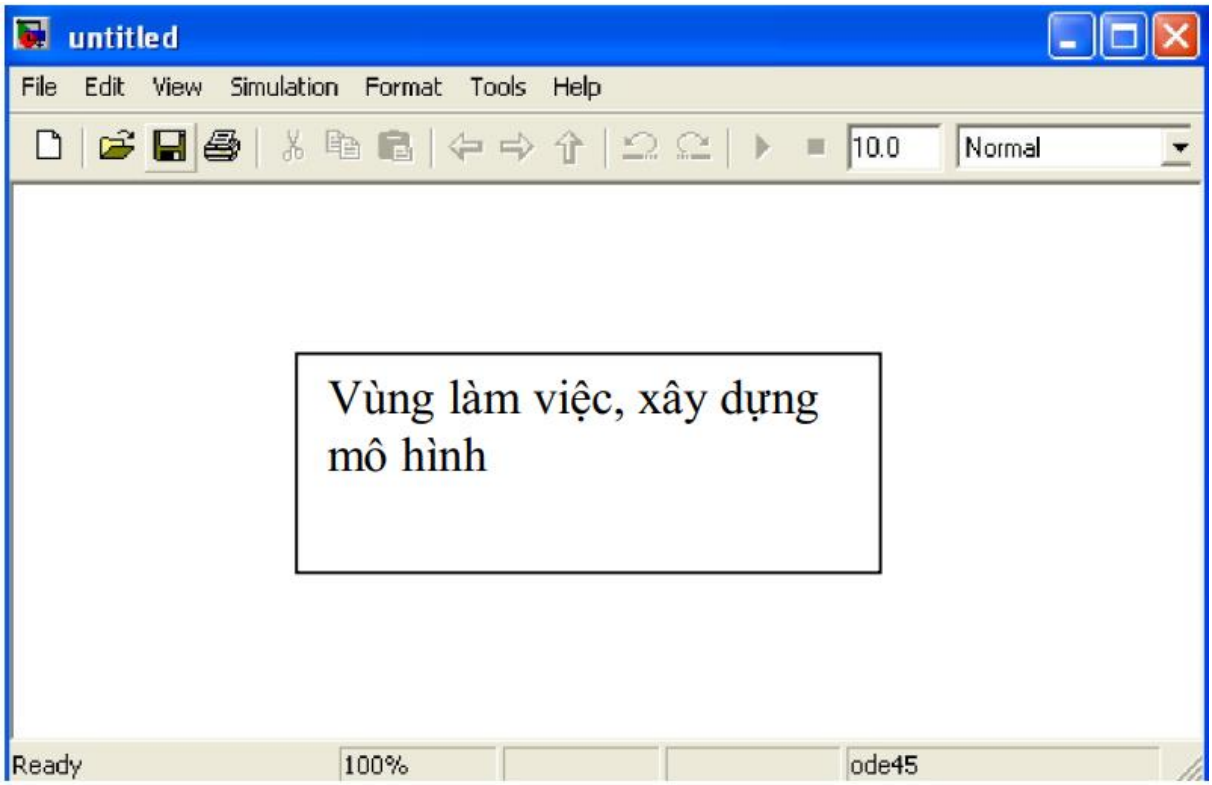


- Từ cửa sổ lệnh, đánh lệnh **simulink** và enter.
- Cửa sổ thư viện Simulink sẽ hiển thị như sau:

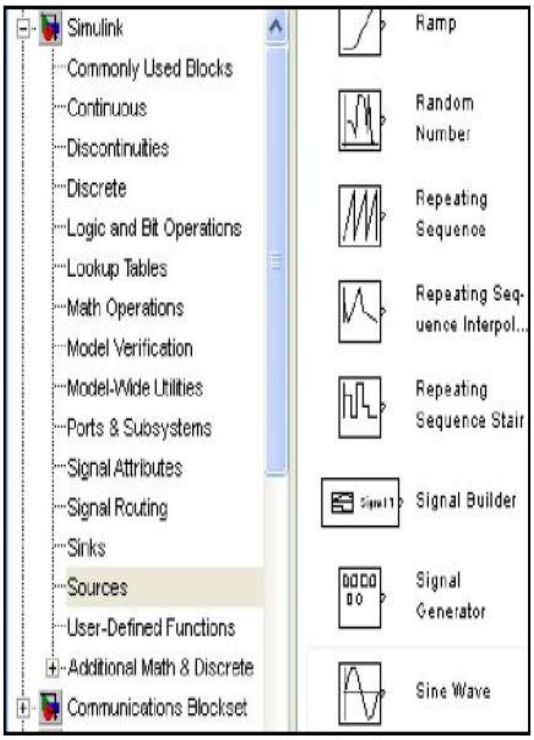


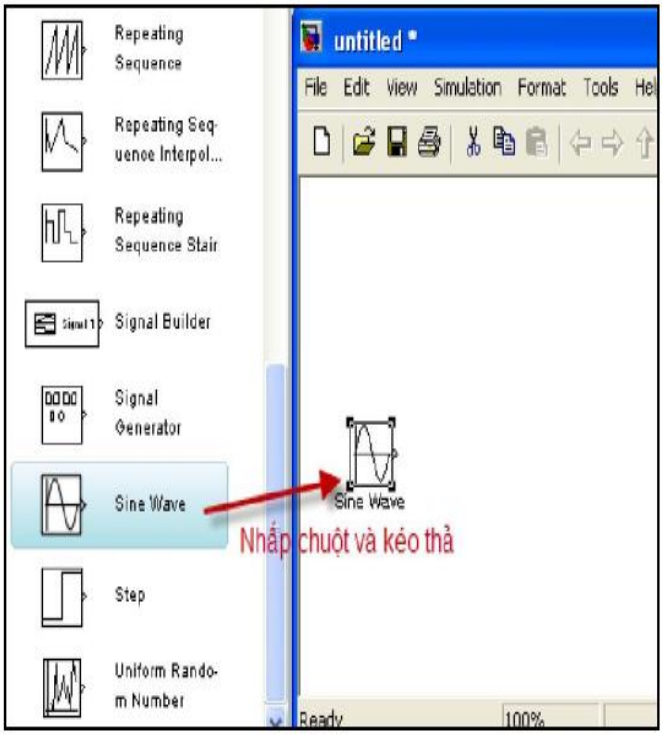
- Tạo một mô hình mới bằng cách:
- Click vào icon **New model** hoặc gõ **Ctrl-N**

- Menu **File → New → Model**
- Cửa sổ xây dựng mô hình xuất hiện:

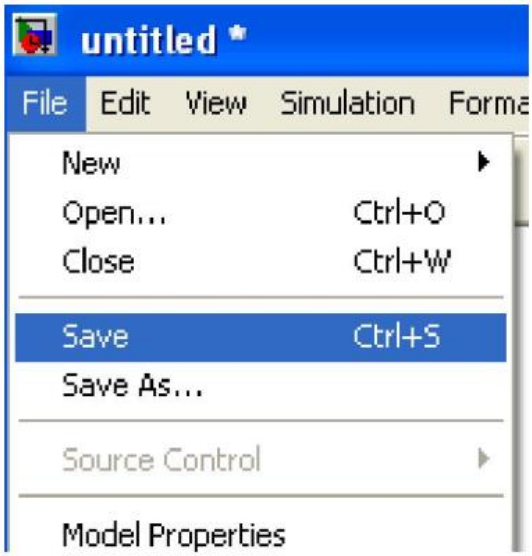


- Tạo các khối bằng cách: từ thư viện Simulink → chọn thư mục con chứa các khối chức năng cần sử dụng → chọn khối cần dùng, nhấp chuột vào và kéo ra ra cửa sổ mô hình

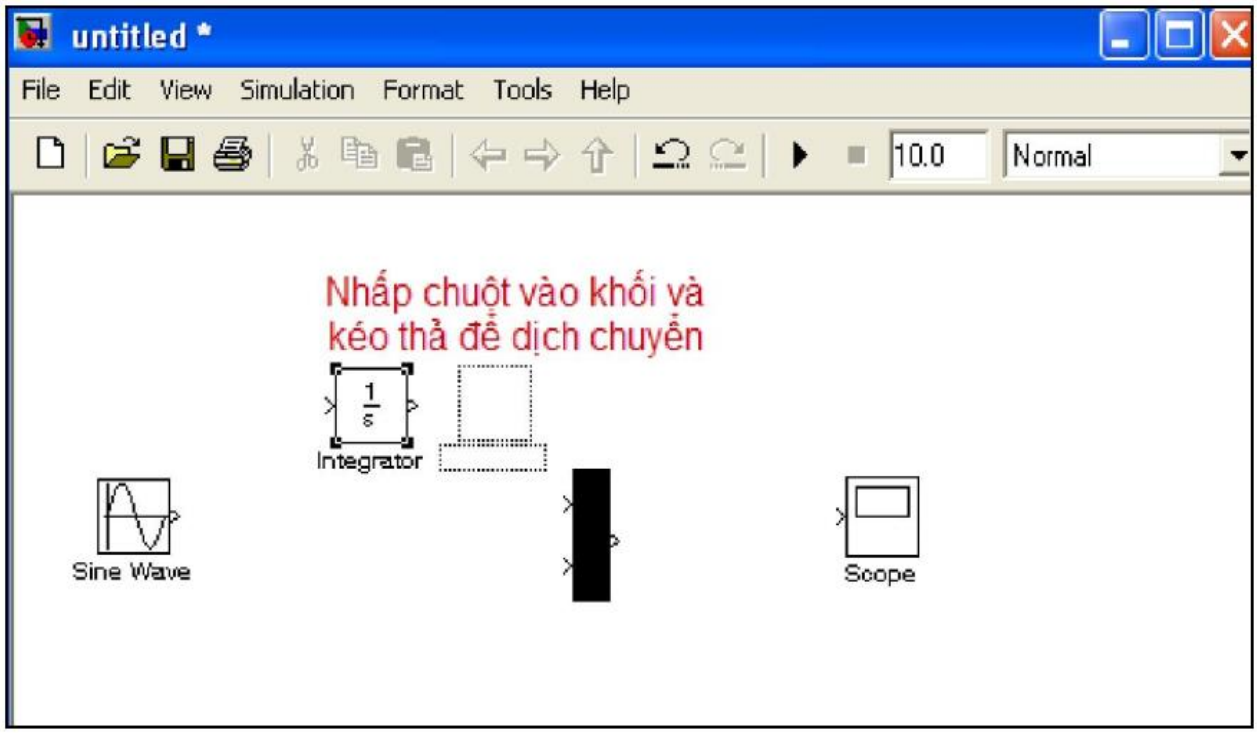




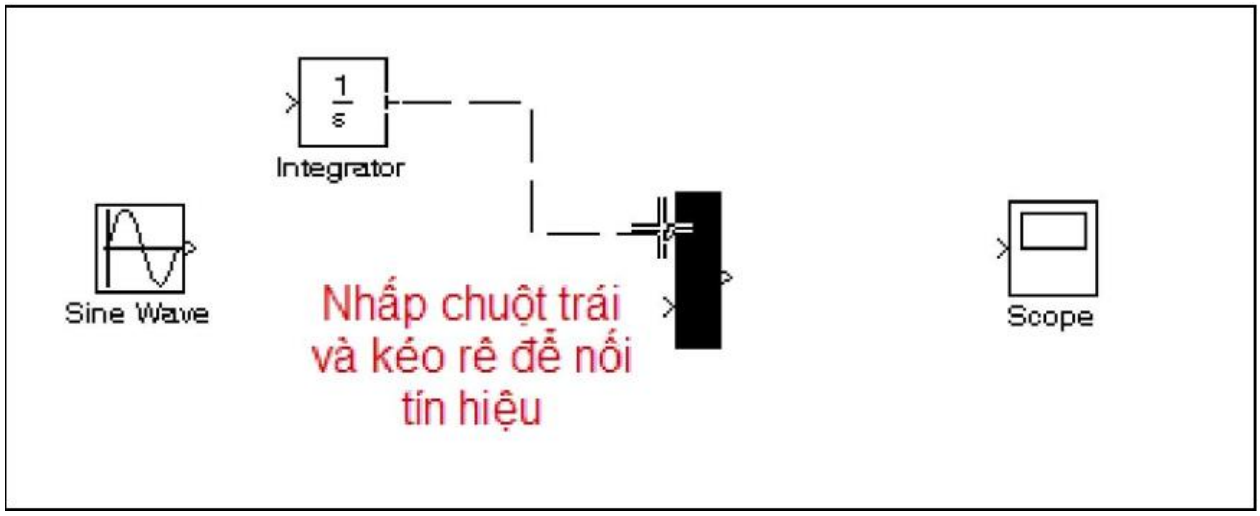
- Lưu trữ mô hình bằng lệnh Save (**File → Save**) hoặc nhấp vào icon **Save**



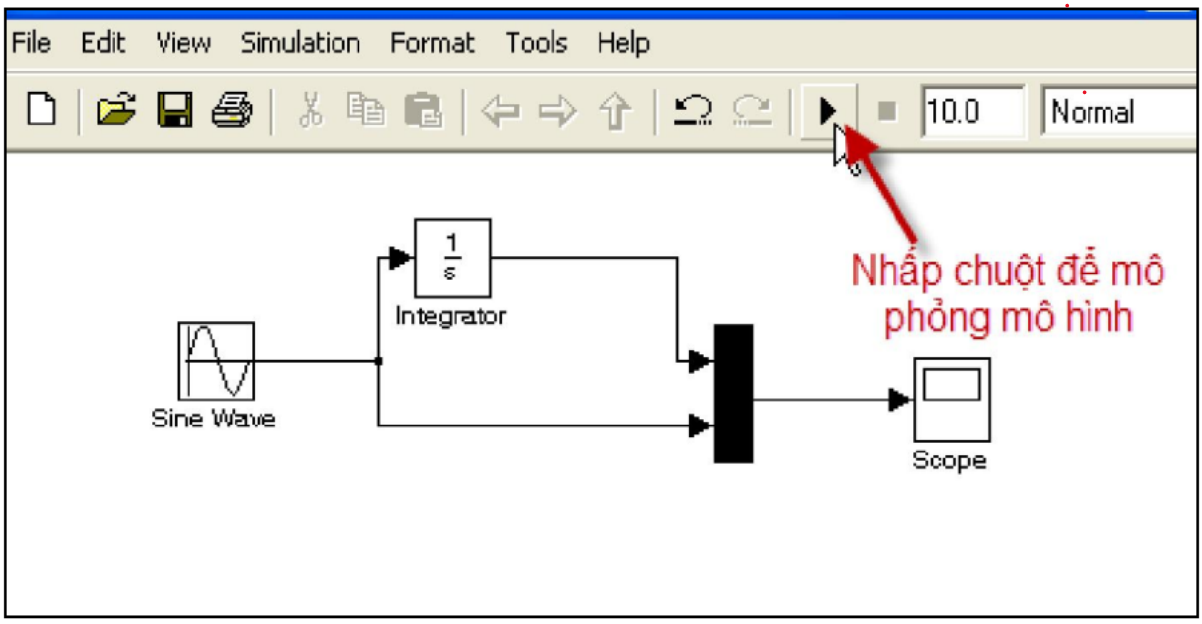
- Dịch chuyển các khối đơn giản bằng cách nhấp vào khối đó và kéo thả



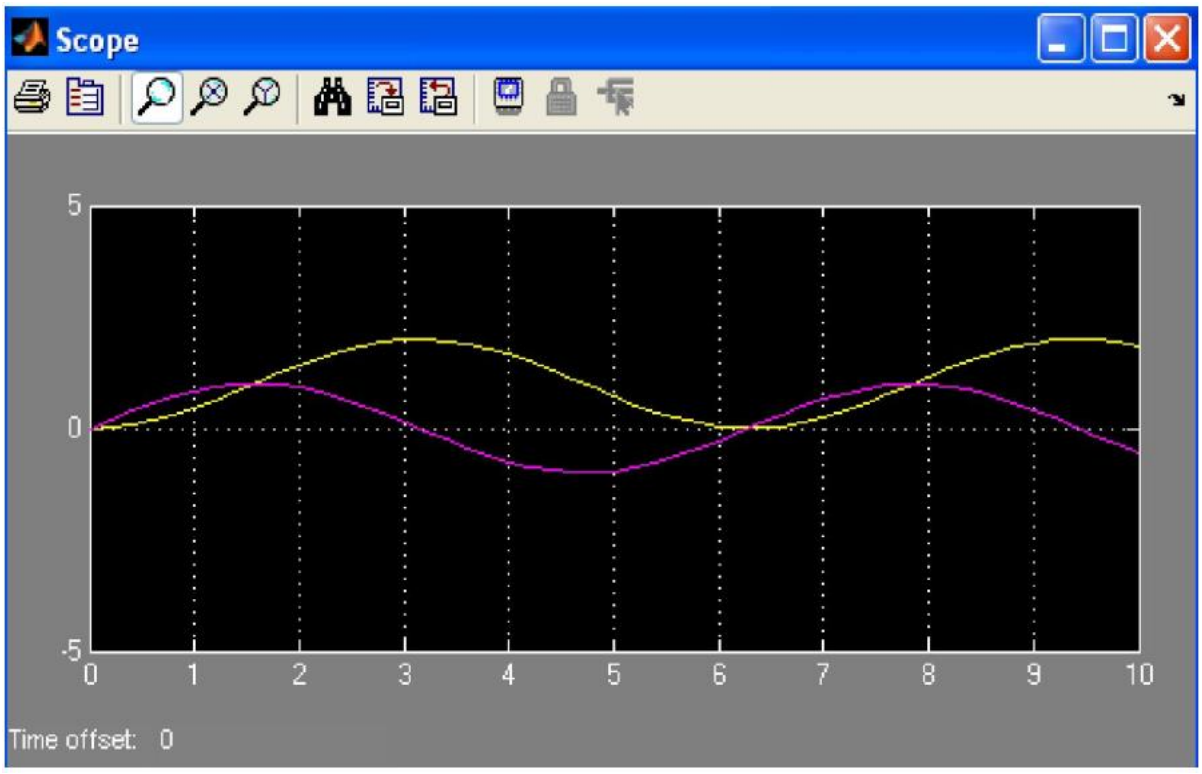
- Kết nối giữa các khối: Đưa con chuột tới ngõ ra của khối (dấu ">"), khi đó con chuột sẽ có dạng "+". Kéo rê chuột tới ngõ vào của một khối khác và thả ra để kết nối tín hiệu.



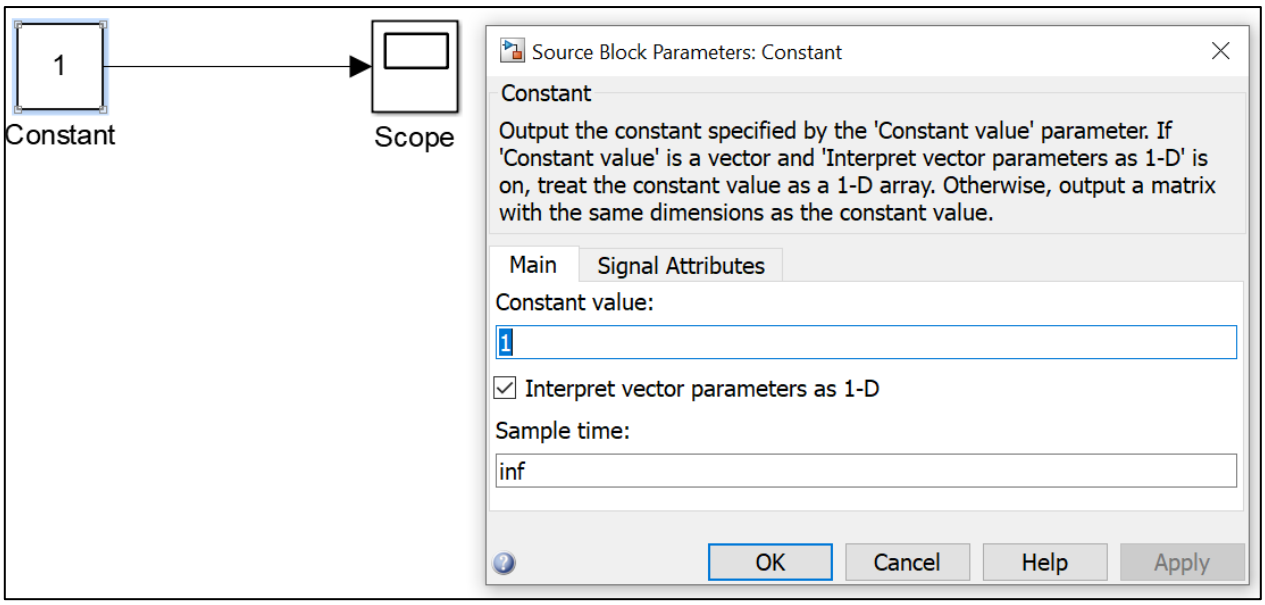
- Chạy mô phỏng mô hình: Dùng lệnh Start (Menu **Simulation → Start**) hoặc nhấp chuột vào icon **Start**



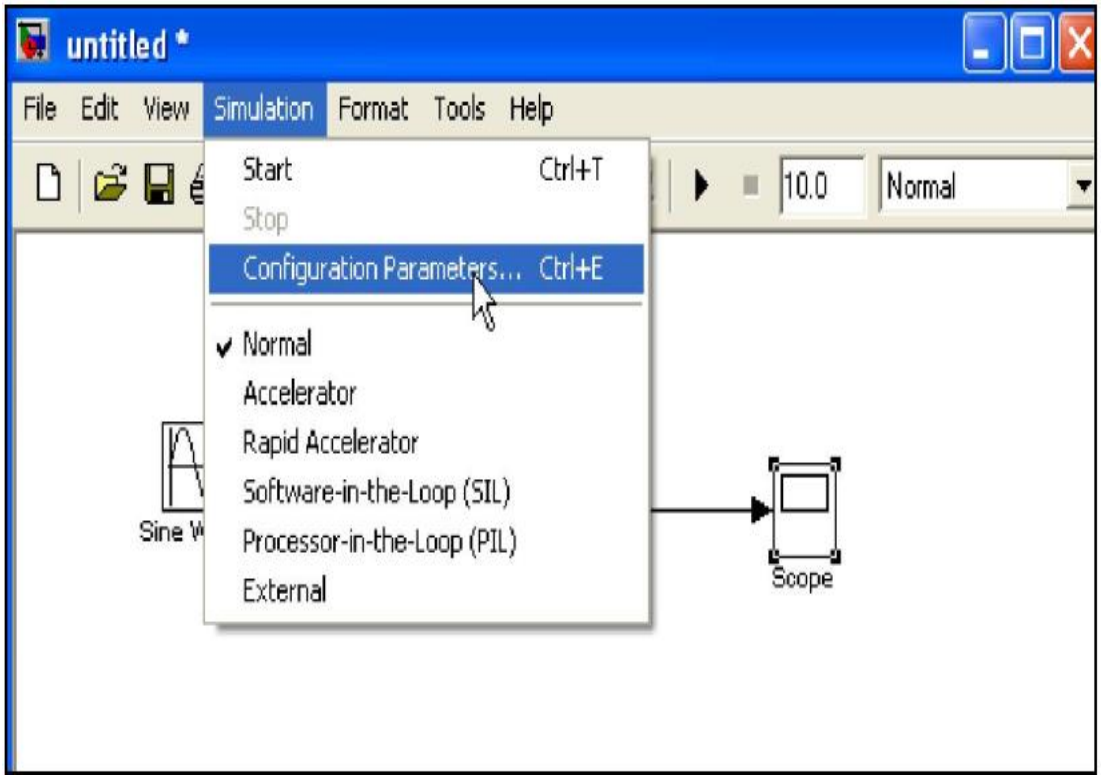
- Xem tín hiệu từ Scope: nhấp đôi vào khối **Scope**



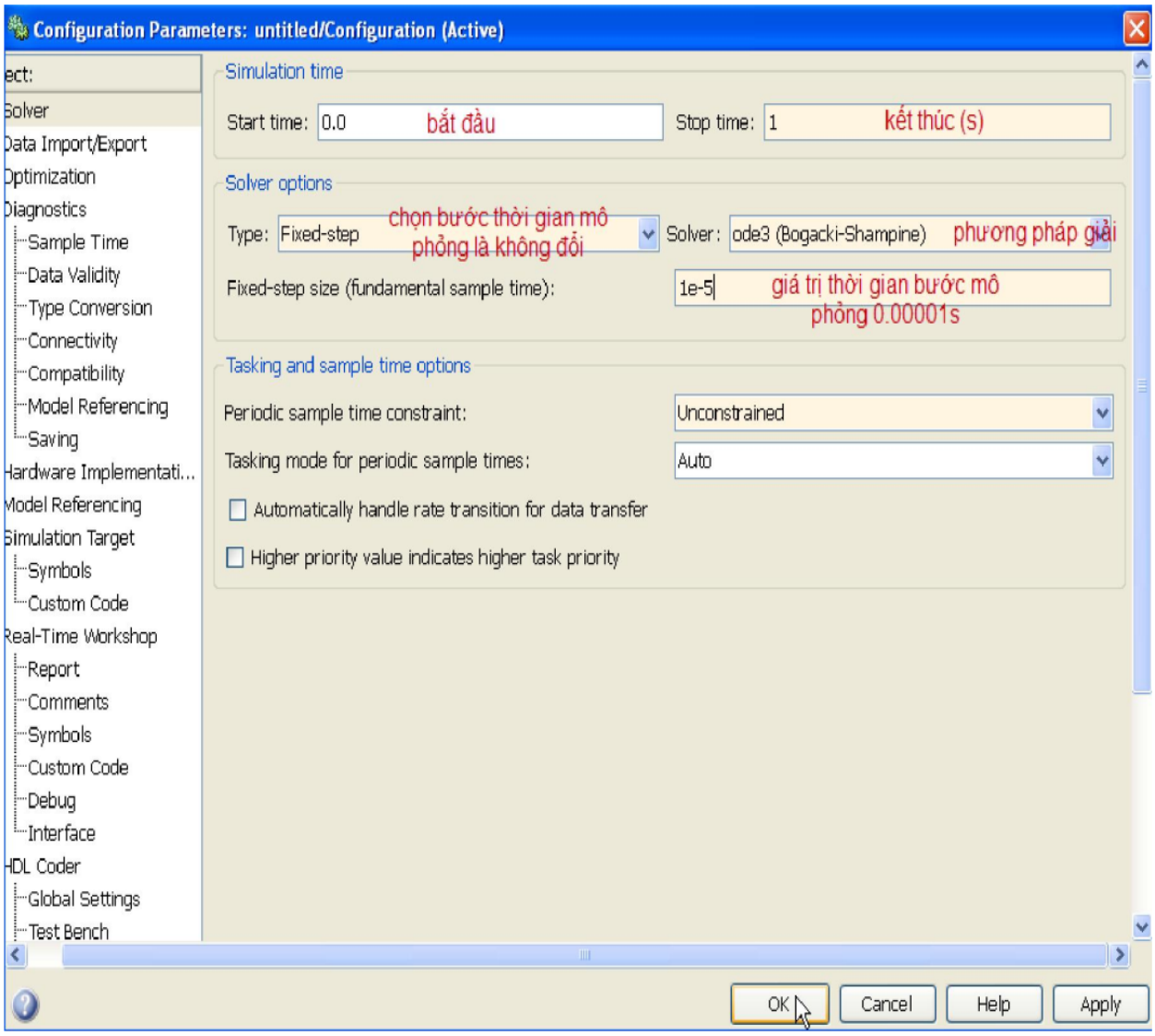
- Chỉnh thông số của một khối bằng cách nhấp đôi vào khối cần chỉnh:
  VD: khối tạo giá trị hằng số constant.



- Trước khi mô phỏng mô hình Simulink, chúng ta cần đặt các thông số mô phỏng bằng cách chọn menu **Simulation → Configuration Parameters**



Ở cửa sổ **Configuration Parameters**, chúng ta có thể đặt một số thông số như **Start time**, **Stop time** (second – giây), các bước thời gian mô phỏng,... sau đó nhấn nút OK.


Configuration Parameters: untitled/Configuration (Active)
ect:
Solver
Data Import/Export
Optimization
Diagnostics
Sample Time
Data Validity
Type Conversion
Connectivity
Compatibility
Model Referencing
Saving
Hardware Implementati...
Model Referencing
Simulation Target
Symbols
Custom Code
Real-Time Workshop
Report
Comments
Symbols
Custom Code
Debug
Interface
HDL Coder
Global Settings
Test Bench
Simulation time
Start time: 0.0
bắt đầu
Stop time: 1
kết thúc (s)
Solver options
Type: Fixed-step
chọn bước thời gian mô phỏng là không đổi
Solver: ode3 (Bogacki-Shampine)
phương pháp giải
Fixed-step size (fundamental sample time): 1e-5
giá trị thời gian bước mô phỏng 0.00001s
Tasking and sample time options
Periodic sample time constraint: Unconstrained
Tasking mode for periodic sample times: Auto
Automatically handle rate transition for data transfer
Higher priority value indicates higher task priority
OK
Cancel
Help
Apply

1.3 Thực hành

1) Sử dụng lệnh **help** hoặc **lookfor** để tìm kiếm thông tin cho các câu hỏi sau:
- Hãy tìm lệnh thể hiện phép toán hàm cosin
- Tìm thông tin về hàm logarithms

2) Cho hai số phức bất kỳ, ví dụ 10 - 4i và -4 – 7i. Hãy thực hiện các phép toán để cộng, trừ, nhân và chia hai số phức này với nhau sao cho đơn giản, nhanh nhất. (gợi ý khai báo biến).

3) Hãy thực hiện các bài tập sau đây (không sử dụng vòng lặp và câu điều kiện):
- Tạo một vector bao gồm những số chẵn trong khoảng từ 35 đến 63.
- Cho x = [23 59 45 16 15 94 45 52 15 84 89 54 62 87 51 42 56 84 87 98 14 25]
    - Trừ đi 15 ở mỗi thành phần của vector
    - Cộng 19 vào các thành phần có vị trí lẻ và trừ 23 vào các thành phần có vị trí chẵn.

4) Vẽ đồ thị của hàm y = sin(x) trong khoảng 0<x<30 thêm tiêu đề và mô tả của các trục vào đồ thị.

5) Cho ma trận **A** (m hàng, n cột), ví dụ A = [2 5 7 4 1; 6 1 7 8 2; 3 5 8 7 9; 9 4 8 3 8]. Thực thi các phép toán sau đối với A:
- Gán hàng thứ m-1 của A cho vector x
- Gán hàng đầu và hàng cuối của A cho y
- Gán 2 cột cuối của A cho z

6) Hãy xóa tất cả các biến (lệnh **clear**). Định nghĩa ma trận A = [10:14; 15:19; 1 1 1 1 1]. Hãy thực thi và kiểm tra kết quả của các phép tính sau:
- x = A(:, 3)
- y = A(3 : 3, 1 : 4)
- B = A(1 : 3, 2 : 2)
- A = [A; 2 1 7 8 7; 7 1 7 4 5]
- A(1, 1) = 9 + A(2, 3)
- C = A([1, 3], 2)
- A(2 : 3, 1 : 3) = [0 0 0; 0 0 0]
- D = A([2, 3, 5], [1, 3, 4])
- D(2, :) = [ ]

7) Vẽ một đường bằng nét gạch ngắn nối các đểm sau lại với nhau: (2, 4), (2.5, 15), (5, 14.5), (8.2, 16.5) và (4, 12).

8) Dùng Simulink vẽ tín hiệu sin như sau: (nhớ bỏ giới hạn bộ đệm data của scope)
- Vẽ 1 tín hiệu sin và biểu diễn bằng scope trong 5s, thay đổi số mẫu (sample) và nhận xét.
- Vẽ 3 tín hiệu sin khác nhau và biểu diễn trên cùng scope trong 5. (gợi ý: thêm số lượng input trong setup scope).