GIÁO TRÌNH MATLAB CƠ BẢN

CHƯƠNG 1: MATLAB CƠ BẢN

§1. CÁC TOÁN TỬ CƠ BẢN CỦA MATLAB

1. Các toán tử cơ bản: Matlab là một phần mềm cao cấp dùng để giải các bài toán. Để khởi động MATLAB ta bấm đúp vào icon của nó. Các file MATLAB có dạng *.m và chỉ chạy trong môi trường MATLAB. MATLAB xử lí số liệu như là ma trận. Khi ta đánh lệnh vào cửa sổ lệnh, nó sẽ được thi hành ngay và kết quả hiện lên màn hình. Nếu ta không muốn cho kết quả hiện lên màn hình thì sau lệnh ta đặt thêm dấu ";". Nếu lệnh quá dài, không vừa một dòng dòng có thể đánh lệnh trên nhiều dòng và cuối mỗi dòng đặt thêm dấu ... rồi xuống dòng. Khi soạn thảo lệnh ta có thể dùng các phím tắt :

↑	Ctrl-P	gọi lại lệnh trước đó
↓	Ctrl-N	gọi lệnh sau
←	Ctrl-B	lùi lại một kí tự
→	Ctrl-F	tiến lên một kí tự
Ctrl-→	Ctrl-R	sang phải một từ
Ctrl-←	Crtl-L	sang phải một từ
home	Ctrl-A	về đầu dòng
end	Ctrl-E	về cuối dòng
esc	Ctrl-U	xoá dòng
del	Ctrl-D	xoá kí tự tại chỗ con nháy đứng
backspace	Ctrl-H	xoá kí tự trước chỗ con nháy đứng

☞ Các phép toán cơ bản của MATLAB gồm:

+	cộng
-	trừ
*	nhân
/	chia phải
\	chia trái
^	luỹ thừa
'	chuyển vị ma trận hay số phức liên hợp

☞ Các toán tử quan hệ :

<	nhỏ hơn
<=	nhỏ hơn hay bằng
>	lớn hơn
>=	lớn hơn hoặc bằng
==	bằng

~=	không bằng

☞ Các toán tử logic :

&	và
|	or
~	not

☞ Các hằng :

pi	3.14159265
i	số ảo
j	tương tự i
eps	sai số 2^{-52}
realmin	số thực nhỏ nhất 2^{-1022}
realmax	số thực lớn nhất 2^{1023}
inf	vô cùng lớn
NaN	Not a number

2. Nhập xuất dữ liệu từ dòng lệnh: MATLAB không đòi hỏi phải khai báo biến trước khi dùng. MATLAB phân biệt chữ hoa và chữ thường. Các số liệu đưa vào môi trường làm việc của MATLAB được lưu lại suốt phiên làm việc cho đến khi gặp lệnh *clear all*. MATLAB cho phép ta nhập số liệu từ dòng lệnh. Khi nhập ma trận từ bàn phím ta phải tuân theo các quy định sau :

- ngăn cách các phần tử của ma trận bằng dấu "," hay dấu trống
- dùng dấu ";" để kết thúc một hàng
- bao các phần tử của ma trận bằng cặp dấu ngoặc vuông [ ]

Để nhập các ma trận sau:

$$A = \begin{bmatrix} 1 & 2 & 4 \\ 3 & -2 & 5 \\ 1 & 5 & 3 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 4 & -2 & 1 \end{bmatrix} \qquad C = \begin{bmatrix} 1 \\ 4 \\ 7 \end{bmatrix}$$

ta dùng các lệnh:

A = [*1 2 3; 3 -2 4; 1 5 3*]
B = [*1 4 2 1*]
C = [*1; 4; 7*]

3. Nhập xuất dữ liệu từ file: MATLAB có thể xử lí hai kiểu file dữ liệu: file

nhị phân *.mat và file ASCII *.dat. Để lưu các ma trận A, B, C dưới dạng file nhị phân ta dùng lệnh:

save ABC A B C

và nạp lại các ma trận A, B bằng lệnh:

load ABC A B

Nếu muốn lưu số liệu của ma trận B dưới dạng file ASCII ta viết:

save b.dat B /ascii

Ta viết chương trình ***ct1_1.m*** như sau:

```
clear
A = [1 2 3; 4 5 6]
B = [3; -2; 1];
C(2) = 2; C(4) = 4
disp('Nhan phim bat ky de xem nhap/xuat du lieu tu file')
save ABC A B C %luu A,B & C duoi dang MAT-file co ten 'ABC.mat'
clear('A', 'C') %xoa  A va C khoi bo nho
load ABC A C %doc MAT - file de nhap A va C vao bo nho
save b.dat B /ascii %luu B duoi dang file ASCII co ten 'b.dat'
clear B
load b.dat %doc ASCII
b
x = input('Nhap x:')
format short e
x
format rat, x
format long, x
format short, x
```

4. Nhập xuất dữ liệu từ bàn phím: Lệnh ***input*** cho phép ta nhập số liệu từ bàn phím. Ví dụ:

```
x = input('Nhap x: ')
```

Lệnh ***format*** cho phép xác định dạng thức của dữ liệu. Ví dụ:

```
format rat % so huu ti
format long % so sẽ có 14 chu so sau dau phay
format long e % so dang mu
format hex % so dang hex
format short e %so dang mu ngan
format short %tro ve so dang ngan (default)
```

Một cách khác để hiển thị giá trị của biến và chuỗi là đánh tên biến vào cửa số lệnh MATLAB. Ta cũng có thể dùng ***disp*** và ***fprintf*** để hiển thị các biến. Ví dụ:

```
disp('Tri so cua  x = '), disp(x)
```

Ta viết chương trình ***ct1_2.m*** như sau:

```
clc
f = input('Nhap nhiet do Fahrenheit[F]:');
c = 5/9*(f - 32);
fprintf('%5.2f(do Fahrenheit) la %5.2f(do C).\n', f, c)
fid = fopen('ct1_2.dat', 'w');
fprintf(fid, '%5.2f(do Fahrenheit) la %5.2f(do C).\n', f, c);
fclose(fid);
```

Trong trường hợp ta muốn nhập một chuỗi từ bàn phím, ta cần phải thêm kí tự s vào đối số. Ví dụ:

```
ans = input('Ban tra loi  <co> hoac  <khong>: ','s')
```

5. Các hàm toán học:

a. Các hàm toán học cơ bản:

exp(x)	hàm e^x
sqrt(x)	căn bậc hai của x
log(x)	logarit tự nhiên

log10(x)	logarit cơ số 10
abs(x)	modun của số phức x
angle(x)	argument của số phức a
conj(x)	số phức liên hợp của x
imag(x)	phần ảo của x
real(x)	phần thực của x
sign(x)	dấu của x

cos(x)
sin(x)
tan(x)
acos(x)
asin(x)
atan(x)
cosh(x)
coth(x)
sinh(x)
tanh(x)
acosh(x)
acoth(x)
asinh(x)
atanh(x)

b. Các hàm toán học tự tạo: MATLAB cho phép ta tạo hàm toán học và lưu nó vào một file để dùng như là hàm có sẵn của MATLAB. Ví dụ ta cần tạo hàm:

$$f_1(x) = \frac{1}{1+8x^2}$$

và hàm:

$$f_2(x) = \begin{bmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{bmatrix} = \begin{bmatrix} x_1^2 + 4x_2^2 - 5 \\ 2x_1^2 - 2x_1 - 3x_2 - 2.5 \end{bmatrix}$$

Muốn thế ta tạo ra file ***f1.m*** như sau:

```
function y = f1(x)
y = 1./(1+8*x.^2);
```

và file ***f2.m***:

```
function y = f2(x)
y(1) = x(1)*x(1)+4*x(2)*x(2) -5;
y(2) = 2*x(1)*x(1)-2*x(1)-3*x(2) -2.5;
```

Khi nhập lệnh ***f1(2)*** ta có giá trị của hàm f1 tại x = 2. Khi nhập lệnh ***f2([2 4])*** ta có giá trị của hàm f2 tại $x_1 = 2$ và $x_2 = 4$. Lệnh ***feval***(‘***f1***’, ***2***) và ***feval***(‘***f2***’, [***2 4***]) cũng cho kết quả tương tự.

Cách thứ hai để biểu diễn một hàm toán học một biến trên dòng lệnh là tạo ra một đối tượng inline từ một biểu thức chuỗi. Ví dụ ta có thể nhập từ dòng lệnh hàm như sau:

```
f1 = inline('1./(1 + 8*x.^2)','x');
f1([0 1]), feval(f1, [0 1])
```

Ta cũng có thể viết:

```
f1 = '1./(1 + 8*x.^2)';
x = [0 1];
eval(f1)
```

Nếu hàm là đa thức ta chỉ cần nhập ma trận các hệ số từ số mũ cao nhất. Ví dụ với đa thức $P_4(x) = x^4 + 4x^3 + 2x + 1$ ta viết:

```
P = [1   4   0   2   1]
```

Để nhân hai đa thức ta dùng lệnh ***conv***; để chia 2 đa thức ta dùng lệnh ***deconv***. Muốn tính trị số của đa thức ta dùng lệnh ***polyval*** và lệnh ***polyvalm*** dùng khi đa thức là ma trận.

c. Các lệnh xử lí hàm: Lệnh ***fplot*** vẽ đồ thị hàm toán học giữa các giá trị đã cho. Ví dụ:

```
fplot('f1', [-5  5 ])
grid on
```

Cho một hàm toán học một biến, ta có thể dùng lệnh ***fminbnd*** của MATLAB để tìm cực tiểu địa phương của hàm trong khoảng đã cho. Ví dụ:

```
f = inline('1./((x - 0.3).^2+0.01) + 1./((x - 0.9).^2 + 0.04) - 6 ');
x = fminbnd(f, 0.3, 1)
```

Lệnh ***fminsearch*** tương tự hàm ***fminbnd*** dùng để tìm cực tiểu địa phương của hàm nhiều biến. Ta có hàm 3 biến lưu trong file ***three_var.m*** như sau:

```
function b = three_var(v)
    x = v(1);
    y = v(2);
    z = v(3);
    b = x.^2 + 2.5*sin(y) - z^2*x^2*y^2;
```

Bây giờ tìm cực tiểu đối với hàm này bắt đầu từ x = -0.6 , y = -1.2 và z = 0.135 bằng các lệnh:

```
v = [-0.6 -1.2  0.135];
a = fminsearch('three_var', v)
```

Lệnh ***fzero*** dùng để tìm điểm zero của hàm một biến. Ví dụ để tìm giá trị không của hàm lân cận giá trị -0.2 ta viết:

```
f = inline('1./((x - 0.3).^2 + 0.01) + 1./((x - 0.9).^2 + 0.04) - 6');
a = fzero(f, -0.2)

Zero found in the interval: [-0.10949, -0.264].
a =
     -0.1316
```

6. Các phép toán trên ma trận và vec tơ:

a. Khái niệm chung: Giả sử ta tạo ra các ma trận a và b bằng các lệnh:

```
a = [1  2  3; 4  5  6];
b = [3  -2  1];
```

Ta có thể sửa đổi chúng:

A = [*a; 7 8 9*]
B = [*b*; [*1 0 -1*]]'

Toán tử ‘ dùng để chuyển vị một ma trận thực và chuyển vị liên hợp một ma trận phức. Nếu chỉ muốn chuyển vị ma trận phức, ta dùng thêm toán tử “.” nghĩa là phải viết “.’”. Ví dụ:

C = [*1 + 2*i 2 - 4*i; 3 + i 2 - 2*j*];
X = *C’*
Y = *C.’*

b. Chỉ số: Phần tử ở hàng i cột j của ma trận m×n có kí hiệu là A(i, j). Tuy nhiên ta cũng có thể tham chiếu tới phần tử của mảng nhờ một chỉ số, ví dụ A(k) với k = i + (j - 1)m. Cách này thường dùng để tham chiếu vec tơ hàng hay cột. Trong trường hợp ma trận đầy đủ thì nó được xem là ma trận một cột dài tạo từ các cột của ma trận ban đầu. Như vậy viết A(5) có nghĩa là tham chiếu phần tử A(2, 2).

Để xác định kích thước của một ma trận ta dùng lệnh ***length***(trả về kích thước lớn nhất) hay ***size***(số hàng và cột). Ví dụ:

c = [*1 2 3 4; 5 6 7 8*];
length(c)
[*m, n*] = *size(c)*

c. Toán tử “:” : Toán tử “:” là một toán tử quan trọng của MATLAB. Nó xuất hiện ở nhiều dạng khác nhau. Ví dụ:

1:10

tạo một vec tơ hàng chứa 10 số nguyên từ 1 đến 10. Lệnh:

100: -7: 50

tạo một dãy số từ 100 đến 51, giảm 7 mỗi lần. Lệnh:

0: pi/4: pi

tạo một dãy số từ 0 đến pi, cách đều nhau pi/4

Các biểu thức chỉ số tham chiếu tới một phần của ma trận. Viết A(1:k, j) là tham chiếu đến 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. Ví dụ:

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

tạo ra ma trận B từ ma trận A bằng cách đổi thứ tự các cột từ [1 2 3] thành [1 3 2]

d. Tạo ma trận bằng hàm có sẵn: MATLAB cung cấp một số hàm để tạo các ma trận cơ bản:

zeros tạo ra ma trận mà các phần tử đều là zeros

z = zeros(2, 4)

ones tạo ra ma trận mà các phần tử đều là 1

x = ones(2, 3)
*y = 5*ones(2, 2)*

rand tạo ra ma trận mà các phần tử ngẫu nhiên phân bố đều

d = rand(4, 4)

randn tạo ra ma trận mà các phần tử ngẫu nhiên phân bố trực giao

e = randn(3, 3)

magic(n) tạo ra ma trận cấp n gồm các số nguyên từ 1 đến n^2 với tổng các hàng bằng tổng các cột n phải lớn hơn hay bằng 3.

pascal(n) tạo ra ma trận xác định dương mà các phần tử lấy từ tam giác Pascal.

pascal(4)

eye(n) tạo ma trận đơn vị

eye(3)

eye(m, n) tạo ma trận đơn vị mở rộng

eye(3, 4)

e. Lắp ghép: Ta có thể lắp ghép(concatenation) các ma trận có sẵn thành một ma trận mới. Ví dụ:

a = ones(3, 3)
*b = 5*ones(3, 3)*
c = [a + 2; b]

f. Xoá hàng và cột : Ta có thể xoá hàng và cột từ ma trận bằng dùng dấu []. Để xoá cột thứ 2 của ma trận b ta viết:

b(:, 2) = []

Viết *x(1: 2: 5) = []* nghĩa là ta xoá các phần tử bắt đầu từ đến phần tử thứ 5 và cách 2 rồi sắp xếp lại ma trận.

g. Các lệnh xử lí ma trận:

Cộng	: X= A + B
Trừ	: X= A - B
Nhân	: X= A * B
	: X.*A nhân các phần tử tương ứng với nhau
Chia	: X = A/B lúc đó X*B = A
	: X = A\B lúc đó A*X = B
	: X=A./B chia các phần tử tương ứng với nhau
Luỹ thừa	: X = A^2
	: X = A.^2
Nghịch đảo	: X = inv(A)
Định thức	: d = det(A)

7. Tạo số ngẫu nhiên: MATLAB có các lệnh tạo số ngẫu nhiên là ***rand*** và ***randn*** tạo ra các số ngẫu nhiên theo phân bố Gauss.

rand(m, n) tạo ra ma trận các số ngẫu nhiên phân bố đồng nhất.

randn(m, n) tạo ra ma trận các số ngẫu nhiên theo phân bố chuẩn Gauss.

rand(3, 3)

randn(3, 3)

8. Các lệnh dùng lập trình:

a. 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 leseif cũng tương tự.

Ví dụ: Ta xét chương trình) ***ct1_4. m*** để đoán tuổi như sau:

```
clc
disp('Xin chao! Han hanh duoc lam quen');
x = fix(30*rand);
disp('Tuoi toi trong khoang 0 - 30');
gu = input('Xin nhap tuoi cua ban: ');
if gu < x
    disp('Ban tre hon toi');
elseif gu > x
    disp('Ban lon hon toi');
else
    disp('Ban bang tuoi toi');
end
```

b. 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

c. while: vòng lặp while dùng khi không biết trước số lần lặp. Cú pháp của nó như sau:

while <biểu thức>
<phát biểu>
end

Xét chương trình in ra chuoi "Xin chao" lên mà hình với số lần nhập từ bàn phím ***ct1_5.m*** như sau:

```
clc
disp('xin chao');
gu = input('Nhap so lan in: ');
i = 0;
while i ~= gu
    disp(['Xin chao' i]);
    i = i + 1
end
```

d. 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>

Ta xây dựng chương trình đoán số ***ct1_6.m***:

```
clc
x = fix(100*rand);
n = 7;
t = 1;
for k = 1:7
  num = int2str(n);
  disp(['Ban co quyen du doan ', num, '  lan']);
  disp('So can doan nam trong khoang 0 - 100');
  gu = input('Nhap so ma ban doan: ');
  if gu < x
    disp('Ban doan nho hon');
  elseif gu > x
    disp('So ban doan lon hon');
  else
    disp('Ban da doan dung. Xin chuc mung');
    t = 0;
    break;
  end
```

```
  n = n - 1;
end
if t > 0
  disp('Ban khong doan ra roi');
  numx = int2str(x);
  disp(['Do la so: ', numx]);
end
```

e. 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.

§2. ĐỒ HOẠ TRONG MATLAB

1. Các lệnh vẽ: MATLAB cung cấp một loạt hàm để vẽ biểu diễn các vec tơ số liệu 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
polar	đồ hoạ trong hệ toạ độ cực
loglog	đồ hoạ với các trục 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
plotyy	đồ hoạ với trục y có nhãn ở bên trái và bên phải

2. Tạo hình vẽ: Hàm ***plot*** có các dạng khác nhau phụ thuộc vào các đối số đưa vào. Ví dụ nếu y là một vec tơ thì ***plot(y)*** tạo ra một đường thẳng quan hệ giữa các giá trị của y và chỉ số của nó. Nếu ta có 2 vec tơ 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)
grid on
polar(t, y)
```

3. Đặc tả kiểu đường vẽ: Ta có thể dùng các kiểu đường vẽ khác nhau khi vẽ hình. Muốn thế ta chuyển kiểu đường vẽ cho hàm ***plot***. Ta viết chương trình ***ct1_7.m*** tạo ra đồ thị hàm hình sin:

```
t = [0: pi/100: 2*pi];
y = sin(t);
plot(t, y, '. ') % vẽ bằng đường chấm chấm
grid on
```

4. Đặc tả màu và kích thước đường vẽ: Để đặc tả màu và kích thước đường vẽ ta dùng các tham số sau:

LineWidth	độ rộng đường thẳng,tính bằng số điểm
MarkerEdgeColor	màu của các cạnh của khối đánh dấu
MarkerFaceColor	màu của khối đánh dấu
MarkerSize	kích thước của khối đánh dấu

Màu được xác định bằng các tham số:

Mã	Màu	Mã	Màu
r	red	m	magenta
g	green	y	yellow
b	blue	k	black
c	cyan	w	white

Các dạng điểm đánh dấu xác định bằng:

Mã	Kiểu đánh dấu	Mã	Kiểu đánh dấu
+	dấu cộng	.	điểm
o	vòng tròn	x	chữ thập
*	dấu sao	s	hình vuông
d	hạt kim cương	v	điểm tam giác hướng xuống
^	điểm tam giác hướng lên	<	tam giác sang trái
>	tam giác sang phải	h	lục giác
p	ngũ giác		

Các dạng đường thẳng xác định bằng:

Mã	Kiểu đường	Mã	Kiểu đường
-	đường liền	:	đường chấm chấm
--	đường đứt nét	-.	đường chấm gạch

Ta xét chương trình ***ct1_8.m*** như sau:

```
x = -pi : pi/10 : pi;
y = tan(sin(x)) - sin(tan(x));
 plot(x, y, '--rs', 'LineWidth', 2, 'MarkerEdgeColor', 'k',...
      'MarkerFaceColor', 'g', 'MarkerSize', 10)
```

Chương trình này sẽ vẽ đường cong y = f(x) có các đặc tả sau :

- đường vẽ là đường đứt nét(--)
- khối đánh dấu hình vuông (s), đường vẽ màu đỏ(r)
- đường vẽ rộng 2 point
- các cạnh của khối đánh màu đen
- khối đánh dấu màu green
- kích thước khối đánh dấu 10 point

5. Thêm đường vẽ vào đồ thị đã có: Để làm điều này ta dùng lệnh ***hold***. Khi ta đánh lệnh ***hold on*** thì MATLAB không xoá đồ thị đang có. Nó thêm số liệu vào đồ thị mới này. Nếu phạm vi giá trị của đồ thị mới vượt quá các giá trị của trục toạ độ cũ thì nó sẽ định lại tỉ lệ xích.

6. Chỉ vẽ các điểm số liệu: Để vẽ các điểm đánh dấu mà không nối chúng lại với nhau ta dùng đặc tả nói rằng không có các đường nối giữa các điểm, nghĩa là ta gọi hàm ***plot*** chỉ với đặc tả màu và điểm đánh dấu. Ta xét chương trình ***ct1_9.m*** như sau:

```
x = -pi : pi/10 : pi;
y = tan(sin(x)) - sin(tan(x));
plot(x, y, 's', 'MarkerEdgeColor', 'k')
```

7. Vẽ các điểm và đường: Để vẽ cả các điểm đánh dấu và đường nối giữa chúng ta cần mô tả kiểu đường và kiểu điểm. Ta xét chương trình ***ct1_10.m***:

```
x = 0:pi/15:4*pi;
y = exp(2*sin(x));
plot(x, y, '-r', x, y, 'ok')
```

dùng vẽ đường cong y = f(x) có đường nối liền, màu đỏ. Điểm đánh dấu là

chữ o có màu đen.

8. Vẽ với hai trục y: Lệnh ***plotyy*** cho phép tạo một đồ thị có hai trục y. Ta cũng có thể dùng ***plotyy*** để cho giá trị trên hai trục y có kiểu khác nhau nhằm tiện so sánh. Ta xét chương trình ***ct1_11.m***:

```
t = 0:900;
A = 1000;
b = 0.005;
a = 0.005;
z2 = sin(b*t);
z1 = A*exp(-a*t);
[haxes, hline1, hline2] = plotyy(t, z1, t, z2,'semilogy', 'plot');
```

9. Vẽ đường cong với số liệu 3 - D: Nếu x, y, z là 3 vec tơ có cùng độ dài thì ***plot3*** sẽ vẽ đường cong 3D. Ta viết chương trình ***ct1_12.m***:

```
t = 0:pi/50:10*pi;
plot3(sin(t),cos(t),t)
axis square;
grid on
```

10. Đặt các thông số cho trục: Khi ta tạo một hình vẽ, MATLAB tự động chọn các giới hạn trên trục toạ độ và khoảng cách đánh dấu dựa trên số liệu dùng để vẽ. Tuy nhiên ta có thể mô tả lại phạm vi giá trị trên trục và khoảng cách đánh dấu theo ý riêng. Ta có thể dùng các lệnh 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
get và set	cho phép xác định và đặt các thuộc tính của trục toạ độ đang có
gca	trở về trục toạ độ cũ

MATLAB chọn các giới hạn trên trục toạ độ và khoảng cách đánh dấu dựa trên số liệu dùng để vẽ. Dùng lệnh ***axis*** có thể đặt lại giới hạn này. Cú pháp của lệnh:

axis[xmin , xmax , ymin , ymax]

Ta xét chương trình ***ct1_13.m*** như sau:

```
x = 0:0.025:pi/2;
plot(x, tan(x), '-ro')
axis([0 pi/2 0 5])
```

MATLAB chia vạch trên trục dựa trên phạm vi dữ liệu và chia đều. Ta có thể mô tả cách chia nhờ thông số ***xtick*** và ***ytick*** bằng một vec tơ tăng dần. Ví dụ xét chương trình ***ct1_14.m***:

```
x = -pi: .1: pi;
y = sin(x);
plot(x, y)
set(gca, 'xtick', -pi :pi/2:p);
set(gca, 'xticklabel', {'-pi', '-pi/2', '0', 'pi/2', 'pi'})
```

11. 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
legend	thêm chú giải vào đồ thị
text	hiển thị chuỗi văn bản ở vị trí nhất định
gtext	đặt văn bản lên đồ hoạ nhờ chuột
\bf	bold font
\it	italics font
\sl	oblique font (chữ nghiêng)
\rm	normal font

Các kí tự đặc biệt xem trong ***String properties*** của Help.

Ta dùng các lệnh ***xlabel , ylabel , zlabel*** để thêm nhãn vào các trục toạ độ. Ta có thể thêm văn bản vào bất kì chỗ nào trên hình vẽ nhờ hàm text. Ta có chương trình ***ct1_15.m***:

```
x = -pi: .1: pi;
y = sin(x);
plot(x, y)
xlabel('t = 0 to 2\pi', 'Fontsize', 16)
ylabel('sin(t)', 'Fontsize', 16)
```

```
title('\it{Gia tri cua sin tu zero đến 2 pi}', 'Fontsize', 16)
text(3*pi/4, sin(3*pi/4),'\leftarrowsin(t ) = 0.707', 'FontSize', 12)
```

12. Định vị văn bản trên hình vẽ: Ta có thể sử dụng đối tượng văn bản để ghi chú các trục ở vị trí bất kì. MATLAB định vị văn bản theo đơn vị dữ liệu trên trục. Ví dụ để vẽ hàm $y = Ae^{\alpha t}$ với A = 0.25 , t = 0 đến 900 và $\alpha = 0.005$ ta viết chương trình ***ct1_16.m***:

```
t = 0: 900;
plot(t, 0.25*exp(-0.005*t))
plot(t, y)
text(300, .25*exp(-.005*300),...
'\bullet\leftarrow\fontname{times}0.25{\ite}^{-   0.005{\itt}} tai,...
 {\itt} = 300', 'FontSize', 14)%ghi chu tai t = 300
```

Tham số ***HorizontalAlignment*** và ***VerticalAlignment*** định vị văn bản so với các toạ độ x, y, z đã cho.

13. Đồ hoạ đặc biệt:

a. Khối và vùng: Đồ hoạ khối và vùng biểu diễn số liệu là vec tơ hay ma trận. MATLAB cung cấp các hàm đồ hoạ khối và vùng :

bar hiển thị các cột của ma trận m*n như là m nhóm, mỗi nhóm có n bar

barh hiển thị các cột của ma trận m*n như là m nhóm, mỗi nhóm có n bar nằm ngang

bar3 hiển thị các cột của ma trận m*n như là m nhóm, mỗi nhóm có n bar dạng 3D

bar3h hiển thị các cột của ma trận m*n như là m nhóm, mỗi nhóm có n bar dạng 3D nằm ngang

Mặc định, mỗi phần tử của ma trận được biểu diễn bằng một bar. Ta xét chương trình ***ct1_17.m***:

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

```
bar(y)
```

b. Mô tả dữ liệu trên trục: Ta dùng các hàm ***xlabel*** và ***ylabel*** để mô tả các dữ liệu trên trục. Ta xét chương trình ***ct1_18.m***:

```
nhdo = [29 23 27 25 20 23 23 27];
ngay = 0: 5: 35;
bar(ngay, nhdo)
xlabel('Ngay')
ylabel('Nhiet do (^{o}C)')
set(gca,'YLim',[15 30],'Layer','top')
grid on
set(gca,'YLim',[15 30])
```

Mặc định,phạm vi giá trị của trục y là từ 0 đến 30. Để xem nhiệt độ trong khoảng từ 15 đến 30 ta thay đổi phạm vi giá trị của trục y:

```
set(gca,'YLim',[15 30],'Layer','top')
```

và trên đồ thị, phạm vi giá trị của trục y đã thay đổi.

c. Xếp chồng đồ thị: Ta có thể xếp chồng số liệu trên đồ thị thanh bằng cách tạo ra một trục khác trên cùng một vị trí và như vậy ta có một trục y độc lập với bộ số liệu khác.

```
TCE = [515 420 370 250 135 120 60 20];
nhdo = [29 23 27 25 20 23 23 27];
ngay = 0:5:35;
bar(ngay, nhdo)
xlabel('Ngay')
ylabel('Nhiet do (^{o}C)')
```

Để xếp chồng một số liệu lên một đồ thị thanh ở trên, có trục thứ 2 ở cùng vị trí như trục thứ nhất ta viết:

```
h1 = gca;
```

và tạo trục thứ 2 ở vị trí trục thứ nhất trước nhất vẽ bộ số liệu thứ 2:

```
h2 = axes('Position',get(h1,'Position'));
```

```
plot(days,TCE,'LineWidth',3)
```

Để trục thứ 2 không gây trở ngại cho trục thứ nhất ta viết:

```
set(h2,'YAxisLocation','right','Color','none','XTickLabel',[])
set(h2,'XLim',get(h1,'XLim'),'Layer','top')
```

Để ghi chú lên đồ thị ta viết:

```
text(11,380,'Mat do','Rotation',--55,'FontSize',16)
ylabel('TCE Mat do (PPM)')
title('Xep chong do thi','FontSize',16)
```

(lưu trong ***ct1_19.m***)

d. ***Đồ hoạ vùng***: Hàm ***area*** hiển thị đường cong tạo từ một vec tơ hay từ một cột của ma trận. Nó vẽ các giá trị của một cột của ma trận thành một đường cong riêng và tô đầy vùng không gian giữa các đường cong và trục x. ta xét chương trình ***ct1_20.m***:

```
Y =  [5 1 2
      8 3 7
      9 6 8
      5 5 5
      4 2 3];
area(Y)
```

hiển thị đồ thị có 3 vùng, mỗi vùng một cột. Độ cao của mỗi đồ thị vùng là tổng các phần tử trong một hàng. Mỗi đường cong sau sử dụng đường cong trước làm cơ sở. Để hiển thị đường chia lưới ta dùng lệnh:

```
set(gca,'Layer','top')
set(gca,'XTick',1:5)
grid on
```

f. Đồ thị pie: Đồ thị pie hiển thị theo tỉ lệ phần trăm của một phần tử của một vec tơ hay một ma trận so với tổng các phần tử. Các lệnh ***pie*** và ***pie3*** tạo ra đồ thị 2D và 3D. ta xét chương trình ***ct1_21.m***:

```
X =  [19.3   22.1   51.6;
      34.2   70.3   82.4;
      61.4   82.9   90.8;
```

```
        50.5   54.9   59.1;
        29.4   36.3   47.0];
x = sum(X);
explode = zeros(size(x));
[c,offset] = max(x);
explode(offset) = 1;
h = pie(x,explode)
%A = [ 1 3 6];
%pie3(A)
```

Khi tổng các phần tử trong đối số thứ nhất bằng hay lớn hơn 1, ***pie*** và ***pie3*** chuẩn hoá các giá trị. Như vậy cho vec tơ x, mỗi phần có diện tích $x_i / \text{sum}(x_i)$ với x_i là một phần tử của x. Giá trị được chuẩn hoá mô tả phần nguyên của mỗi vùng. Khi tổng các phần tử trong đối số thứ nhất nhỏ hơn 1, ***pie*** và ***pie3*** không chuẩn hoá các phần tử của vec tơ x. Chúng vẽ một phần pie.

```
x = [.19 .22 .41];
pie(x)
```

g. Làm hình chuyển động: Ta có thể tạo ra hình chuyển động bằng 2 cách

- tạo và lưu nhiều hình khác nhau và lần lượt hiển thị chúng
- vẽ và xoá liên tục một đối tượng trên màn hình,mỗi lần vẽ lại có sự thay đổi.

Với cách thứ nhất ta thực hiện hình chuyển động qua 3 bước:

- dùng hàm ***moviein*** để dành bộ nhớ cho một ma trận đủ lớn nhằm lưu các khung hình.
- dùng hàm ***getframes*** để tạo các khung hình.
- dùng hàm ***movie*** để hiển thị các khung hình.

Sau đây là ví dụ sử dụng ***movie*** để quan sát hàm ***fft(eye(n))***.Ta tạo chương trình ***ct1_22.m*** như sau :

```
axis equal
M = moviein(16, gcf);
set(gca, 'NextPlot', 'replacechildren')
h = uicontrol('style', 'slider', 'position',[100 10 500 20], 'Min', 1, 'Max', 16)
for j = 1:16
        plot(fft(eye(j + 16)))
```

```
        set(h, 'Value', j)
M(:, j) = getframe(gcf);
end
clf;
axes('Position', [0 0 1 1]);
movie(M, 30)
```

Bước đầu tiên để tạo hình ảnh chuyển động là khởi gán ma trận. Tuy nhiên trước khi gọi hàm ***moviein***, ta cần tạo ra các trục toạ độ có cùng kích thước với kích thước mà ta muốn hiển thị hình. Do trong ví dụ này ta hiển thị các số liệu cách đều trên vòng tròn đơn vị nên ta dùng lệnh ***axis equal*** để xác định tỉ lệ các trục. Hàm ***moviein*** tạo ra ma trận đủ lớn để chứa 16 khung hình. Phát biểu:

```
set(gca, 'NextPlot', 'replacechildren')
```

ngăn hàm ***plot*** đưa tỉ lệ các trục về ***axis normal*** mỗi khi nó được gọi. Hàm ***getframe*** không đối số trả lại các điểm ảnh của trục hiện hành ở hình hiện có. Mỗi khung hình gồm các số liệu trong một vec tơ cột. Hàm ***getframe(gcf)*** chụp toàn bộ phần trong của một cửa sổ hiện hành. Sau khi tạo ra hình ảnh ta có thể chạy chúng một số lần nhất định ví dụ 30 lần nhờ hàm ***movie(M, 30)*** .

Một phương pháp nữa để tạo hình chuyển động là vẽ và xoá, nghĩa là vẽ một đối tượng đồ hoạ rồi thay đổi vị trí của nó bằng cách thay đổi toạ độ x, y và z một lượng nhỏ nhờ một vòng lặp. Ta có thể tạo ra các hiệu ứng khác nhau nhờ các cách xoá hình khác nhau. Chúng gồm:

- none — MATLAB không xoá đối tượng khi nó di chuyển
- background — MATLAB xoá đối tượng bằng cách vẽ nó có màu nền
- xor — MATLAB chỉ xoá đối tượng

Ta tạo ra M-file có tên là ***ct1_23.m*** như sau:

```
A = [ -8/3 0 0; 0 -10 10; 0 28 -1 ];
y = [35 -10 -7]';
h = 0.01;
p = plot3(y(1), y(2), y(3),'.', ...
'EraseMode', 'none', 'MarkerSize', 5);
axis([0 50 -25 25 -25 25])
```

```
hold on
for i = 1:4000
        A(1,3) = y(2);
        A(3,1) = -y(2);
        ydot = A*y;
        y = y + h*ydot;
        set(p, 'XData', y(1), 'YData', y(2), 'ZData', y(3)) % thay doi toa do
        drawnow
        i = i + 1;
end
```

13. Đồ hoạ 3D:

a.Các lệnh cơ bản: Lệnh ***mesh*** và ***surf*** tạo ra lưới và mặt 3D từ ma trận số liệu. Gọi ma trận số liệu là z mà mỗi phần tử của nó z(i, j) xác định tung độ của mặt thì ***mesh(z)*** tạo ra một lưới có màu thể hiện mặt z còn ***surf(z)*** tạo ra một mặt có màu z.

b. Đồ thị các hàm hai biến: Bước thứ nhất để thể hiện hàm 2 biến z=f(x,y) là tạo ma trận x và y chứa các toạ độ trong miền xác định của hàm. Hàm meshgrid sẽ biến đổi vùng xác định bởi 2 vec tơ x và y thành ma trận x và y. Sau đó ta dùng ma trận này để đánh giá hàm.

Ta khảo sát hàm ***sin(r)/r***. Để tính hàm trong khoảng -8 và 8 theo x và y ta chỉ cần chuyển một vec tơ đối số cho ***meshgrid***:

```
[x,y] = meshgrid(-8:.5:8);
r = sqrt(x.^2 + y.^2) + 0.005;
```

ma trận r chứa khoảng cách từ tâm của ma trận. Tiếp theo ta dùng hàm ***mesh*** để vẽ hàm.

```
z = sin(r)./r;
mesh(z)
```

c. Đồ thị đường đẳng mức: Các hàm contour tạo, hiển thị và ghi chú các đường đẳng mức của một hay nhiều ma trận. Chúng gồm:

clabel	tạo các nhãn sử dụng ma trận contour và hiển thị nhãn
contour	hiển thị các đường đẳng mức tạo bởi một giá trị cho trước của ma trận Z.

contour3 hiển thị các mặt đẳng mức tạo bởi một giá trị cho trước của ma trận Z.

contourf hiển thị đồ thị contour 2D và tô màu vùng giữa 2 các đường

contourc hàm cấp thấp để tính ma trận contour

Hàm ***meshc*** hiển thị contour và lưới và ***surfc*** hiển thị mặt contour.

```
[X,Y,Z] = peaks;
contour(X,Y,Z,20)
```

Mỗi contour có một giá trị gắn với nó. Hàm ***clabel*** dùng giá trị này để hiển thị nhãn đường đồng mức 2D. Ma trận contour chứa giá trị clabel dùng cho các đường contour 2D. Ma trận này được xác định bởi ***contour, contour3*** và ***contourf***.

Để hiển thị 10 đường đẳng mức của hàm ***peak*** ta viết:

```
Z = peaks;
[C,h] = contour(Z,10);
clabel(C,h)
title({'Cac contour co nhan','clabel(C,h)'})
```

Hàm ***contourf*** hiển thị đồ thị đường đẳng mức trên một mặt phẳng và tô màu vùng còn lại giữa các đường đẳng mức. Để kiểm soát màu tô ta dùng hàm ***caxis*** và ***colormap***. Ta viết chương trình ***ct1_26.m***:

```
Z = peaks;
[C, h] = contourf(Z, 10);
caxis([-20 20])
colormap autumn;
title({'Contour co to mau', 'contourf(Z, 10)'})
```

Các hàm ***contour(z, n)*** và ***contour(z, v)*** cho phép ta chỉ rõ số lượng mức contour hay một mức contour cần vẽ nào đó với z là ma trận số liệu, n là số đường contour và v là vec tơ các mức contour. MATLAB không phân biệt giữa vec tơ một phần tử hay đại lượng vô hướng. Như vậy nếu v là vec tơ một phần tử mô tả một contour đơn ở một mức hàm ***contour*** sẽ coi nó là số lượng đường contour chứ không phải là mức contour. Nghĩa là, ***contour(z, v)*** cũng như ***contour(z, n)***. Để hiển thị một đường đẳng mức ta cần cho v là một

vec tơ có 2 phần tử với cả hai phần tử bằng mức mong muốn. Ví dụ để tạo ra một đường đẳng mức 3D của hàm peaks ta viết chương trình ***ct1_27.m***:

```
xrange = -3: .125: 3;
yrange = xrange;
[X,Y] = meshgrid(xrange, yrange);
Z = peaks(X, Y);
contour3(X, Y, Z)
```

Để hiển thị một mức ở Z = 1, ta cho v là [1 1]

```
v = [1 1]
contour3(X, Y, Z, v)
```

Hàm ***ginput*** cho phép ta dùng chuột hay các phím mũi tên để chọn các điểm vẽ. Nó trả về toạ độ của vị trí con trỏ. Ví dụ sau sẽ minh hoạ các dùng hàm ***ginput*** và hàm spline để tạo ra đường cong nội suy hai biến.

Ta tạo một M-file có tên ***ct1_28.m*** như sau:

```
disp('Chuot phai tro cac diem tren duong ve')
disp('Chuot trai tro diem cuoi cua duong ve')
axis([0 10 0 10])
hold on
x = [];
y  = [];
n = 0;
but = 1;
while but = =1
        [xi,yi,but] = ginput(1);
        plot(xi, yi, 'go')
        n = n  + 1;
        x(n, 1) = xi;
        y(n,1) = yi;
end
t = 1:n;
ts = 1: 0.1: n;
xs = spline(t, x, ts);
```

```
ys = spline(t, y, ts);
plot(xs, ys, 'c-');
hold off
```

14. Vẽ các vectơ: Có nhiều hàm MATLAB dùng hiển thị các vec tơ có hướng và vec tơ vận tốc. Ta định nghĩa một vec tơ bàng cách dùng một hay 2 đối số. Các đối số mô tả thành phần x và thành phần y của vec tơ. Nếu ta dùng 2 đối số thì đối số thứ nhất sẽ mô tả thành phần x và đối số thứ ha mô tả thành phần y. Nếu ta chỉ dùng một đối số thì MATLAB xử lí nó như một số phức, phần thực là thành phần x và phần ảo là thành phần y.

Các hàm vẽ vec tơ gồm:

compass vẽ các véc tơ bắt đầu từ gốc toạ độ của hệ toạ độ cực
feather vẽ các vec tơ bắt đầu từ một đường thẳng
quiver vẽ các vec tơ 2D có các thành phần (u, v)
quiver3 vẽ các vec tơ 3D có các thành phần (u, v, w)

a. Hàm compass: Ta xét ví dụ vẽ hướng và tốc độ gió. Các vec tơ xác định hướng (góc tính bằng độ) và tốc độ gió (km/h) là:

```
hg = [45 90 90 45 360 335 360 270 335 270 335 335];
td = [6 6 8 6 3 9 6 8 9 10 14 12];
```

Ta biến đổi hướng gió thành radian trước khi biến đổi nó thành toạ độ vuông góc.

```
hg1 = hg * pi/180;
[x, y] = pol2cart(hg1, td);
compass(x, y)
```

và tạo ra ghi chú trên đồ thị:

```
gc = {'Huong gio và suc gio tai san bay Da Nang')
text(–28, 15, gc)
```

b. Hàm feather: Hàm feather hiển thị các vec từ bắt đầu từ một đường thẳng song song với trục x. Ví dụ để tạo ra các vec tơ có góc từ 90^0 đến 0^0 và cùng độ dài ta viết chương trình ***ct1_30.m***:

```
theta = 90: –10: 0;
```

```
r = ones(size(theta));
```

trước khi vẽ, chuyển các số liệu sang toạ độ vuông góc và tăng độ lớn thành r để dễ nhìn:

```
[u, v] = pol2cart(theta*pi/180, r*10);
feather(u, v)
axis equal
```

Nếu đối số là số phức z thì ***feather*** coi phần thực là x và phần ảo là y. Ta xét chương trình ***ct1_31.m***:

```
t = 0: 0.3: 10;
s = 0.05 + i;
Z = exp(–s*t);
feather(Z)
```

c. Hàm quiver: Hàm quiver hiển thị các vec tơ ở các điểm đã cho trong mặt phẳng. Các vec tơ này được xác định bằng các thành phần x và y.
Ví dụ để tạo ra 10 contour của hàm peaks ta dùng chương trình ***ct1_32.m***:

```
n = –2.0: .2: 2.0;
[X,Y,Z] = peaks(n);
contour(X, Y, Z, 10)
```

Bây giờ dùng hàm gradient để tạo các thành phần của vec tơ dùng làm đối số cho ***quiver***:

```
[U, V] = gradient(Z, .2);
```

Đặt ***hold on*** để thêm đường contour:

```
hold on
quiver(X,Y,U,V)
hold off
```

d. Hàm quiver3: Hàm ***quiver3*** hiển thị các vec tơ có các thành phần (u,v,w) tại điểm (x, y, z). Ví dụ ta biểu diễn quỹ đạo của một vật được ném đi theo t. Phương trình của chuyển động là:

$$z(t) = v_0 t + \frac{at^2}{2}$$

Ta viết chương trình ***ct1_33.m***. Trước hết ta gán vận tốc ban đầu và gia tốc a:

```
v0 = 10; % Van toc ban dau
a = –32; % gia toc
```

Tiếp theo tính z tại các thời điểm:

```
t = 0:.1:1;
z = vz*t + 1/2*a*t.^2;
```

Tính vị trí theo hướng x và y:

```
vx = 2;
x = vx*t;
vy = 3;
y = vy*t;
```

Tính các thành phần của vec tơ vận tốc và hiển thị bằng các dùng ***quiver3***:

```
u = gradient(x);
v = gradient(y);
w = gradient(z);
scale = 0;
quiver3(x, y, z, u, v, w, scale)
axis square
```

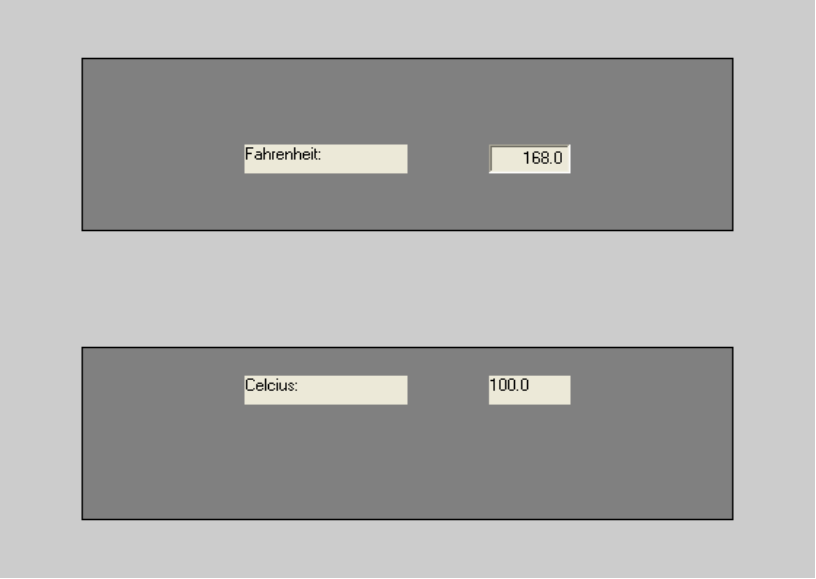
§3. GIAO DIỆN ĐỒ HOẠ

1. Khái niệm chung: Để tiện dụng ta có thể tạo nên giao diện đồ hoạ(GUI - Graphic User Interface) giữa người dùng và MATLAB. Trong giao diện này ta có thể xuất dữ liệu dưới 2 dạng: văn bản và đồ hoạ. Mỗi một GUI có một hay nhiều layout(diện mạo). Việc tạo GUI tạo nên một công cụ đồ hoạ phục vụ

nhập xuất dữ liệu một cách trực giác, rất thuận tiện. Ngoài ra có thể dùng GUI để giám sát các quá trình, hiển thị các đối tượng.

2. Nhập xuất kí tự, số liệu ra GUI:

a. Tạo khung hình: Ta xét các lệnh sau(***ct1_35.m***):




```
f = input('Nhap nhiet do(do K): ');
c = (f - 32)*5/9;
fprintf(1,'nhiet do(do C) la: %g\n', c)
```

Ba dòng lệnh trên thực hiện các công việc sau:

- nhập giá trị đầu vào
- thực hiện phép tính quy đổi nhiệt độ
- xuất kết quả ra màn hình

Bây giờ ta tìm cách cài các dòng lệnh trên sao cho chúng thực hiện trên khuôn khổ một khung đồ hoạ có dạng như trên

Các lệnh sau(***ct1_36.m***) thực hiện công việc trên:

```
set(gcf,'DefaultUicontrolUnit', 'Normalized')
frame_1 = uicontrol(gcf, 'Style', 'Frame',...
                    'Position', [0.1 0.1  0.8 0.3]);
frame_2 = uicontrol(gcf, 'Style', 'Frame',...
                    'Position', [0.1 0.6  0.8 0.3]);
set(frame_1, 'BackgroundColor',  [0.5 0.5 0.5]);
set(frame_2, 'BackgroundColor',  [0.5 0.5 0.5]);
```

```
text_f = uicontrol(gcf, 'Style', 'Text',...
                   'String',      'Fahrenheit: ',...
                   'Position',  [0.3 0.7 0.2 0.05],'HorizontalAlignment','Left');
edit_f = uicontrol(gcf, 'Style',  'Edit',...
                   'String',  '168.0',...
                   'Position',  [0.6 0.7 0.1 0.05 ],...
                   'HorizontalAlignment',  'Right',...
                   'Callback',  'ct1_38');
text_c1 = uicontrol(gcf,'Style',  'Text',...
                     'String',  'Celcius: ',...
                     'Position',  [0.3 0.3 0.2 0.05],...
                     'HorizontalAlignment',  'Left');
text_c2 = uicontrol(gcf,'Style',  'Text',...
                     'String',  '100.0',...
                     'Position',  [0.6 0.3 0.1 0.05],...
                     'HorizontalAlignment',  'Left');
```

Bây giờ ta sẽ xem các lệnh trên hoạt động như thế nào. Các lệnh sau:

```
set(gcf,'DefaultUicontrolUnit',  'Normalized')
frame1 = uicontrol(gcf,'Style',  'Frame',...
                       'Position',  [0.1 0.1 0.8 0.3]);
frame2 = uicontrol(gcf,'Style',  'Frame',...
                       'Position',  [0.1 0.6 0.8 0.3]);
set(frame1,'BackgroundColor',  [0.5 0.5 0.5]);
set(frame2,'BackgroundColor',  [0.5 0.5 0.5]);
```

tạo hai khung hình chữ nhật trong cửa sổ Figure hiện hành với nền màu xám. Hai khung (Frames) có toạ độ các góc dưới trái là (0.1, 0.1) và (0.1, 0.6), cùng chiều cao 0.3 đơn vị và bề rộng 0.8 đơn vị. Đơn vị được tính bằng % của kích cỡ ngoài của Figure. Vậy ta có thể diễn giải như sau:

- Khung thứ nhất có góc trái dưới tại điểm có toạ độ 10% chiều ngang và 10% chiều cao của khung ngoài Figure.

- Khung thứ 2 có góc trái phía dưới tại điểm có toạ độ ứng với 10% chiều ngang và 60% chiều cao của khung ngoài Figure.

- Cả hai khung có chiều cao bằng 30% chiều cao và bề ngang bằng 80% bề ngang của khung ngoài Figure.

b. Dùng lệnh edit và text để nhập xuất kí tự và số liệu: Trên đây ta đã dùng lệnh uicontrol để tạo và xác định vị trí hai khung hình. Đoạn lệnh sau sử dụng uicontrol để viết chuỗi kí tự "Fahrenheit" lên khung bên trên:

```
text_f = uicontrol(gcf,'Style','Text','String','Fahrenheit: ',...
                   'Position',[0.3 0.7 0.2 0.05],'HorizontalAlignment','Left');
```

Chuỗi kí tự "Fahrenhaeit" được đặt vào đúng vị trí dồn trái của ô có Position ghi trong đoạn chương trình trên. Đoạn lệnh sau dùng Edit để viết chuỗi kí tự "68.0" vào vị trí bên cạnh của "Fahrenheit". Chuỗi kí tự có vị trí dồn phải trong ô (Position Box).

```
edit_f = uicontrol(gcf,'Style',  'Edit',...
                   'String',  '168.0',...
                   'Position',  [0.6 0.7 0.1 0.05 ],...
                   'HorizontalAlignment',  'Right',...
                   'Callback',  'ct1_38');
```

Do sử dụng edit, chuỗi kí tự "68.0" là chuỗi có thể viết lại được trực tiếp trên GUI. Sau khi nhấn nút trên, giá trị mới viết lại được tiếp nhận và MATLAB sẽ gọi lệnh viết trong phần callback ***ct1_38.m***.

Cuối cùng ta còn phải dùng uicontrol để tạo ta chuỗi text, hiển thị chuỗi "Celcius" và "20.0" trong khung bên dưới.

```
text_c1 =  uicontrol(gcf,'Style','Text','String','Celcius: ',...
                     'Position',[0.3 0.3 0.2 0.05],'HorizontalAlignment','Left');
text_c2 =  uicontrol(gcf,'Style','Text','String','20.0','Position',...
                     [0.6 0.3 0.1 0.05],'HorizontalAlignment','Left');
```

c. Tự động cập nhật giá trị lên GUI: Để hoàn thiện ví dụ GUI ta thực hiện chương trình với nhiệm vụ tính quy đổi từ độ K sang độ C và tự động điền kết quả vào các ô bên cạnh chuỗi Celcius. Đoạn mã sau phục vụ mục đích callback (hoàn trả giá trị) được lưu vào file ***ct1_38.m*** và có nội dung như sau:

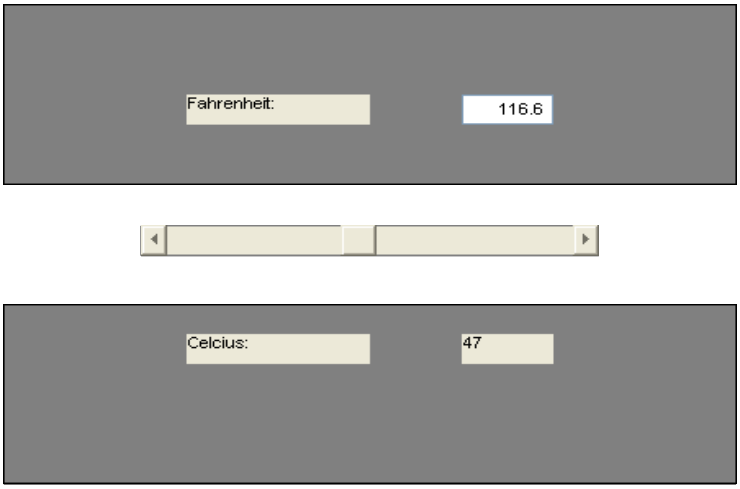
```
f = get(edit_f, 'String');
f = str2num(f);
c = ( f - 32)*5/9;
```

```
c = num2str(c);
set(text_c2, 'String',c);
```

Đoạn mã trên nhận giá trị do lệnh uicontrol "edit" đọc vào dưới dạng chuỗi (string) và sau đó:

- biến đổi từ dạng string sang dạng số
- tính quy đổi từ nhiệt độ fahrenheit sang nhiệt độ celcius
- biến đổi từ số sang string
- xuất kết quả dưới dạng string ra GUI nhờ text_c2

3. Nhập số liệu từ thanh trượt: Ngoài cách nhập số liệu từ bàn phím, ta có thể nhập số liệu từ thanh trượt. Ta muốn tạo ra một giao diện như sau:



Trong giao diện này, con trượt sẽ làm thay đổi giá trị nhiệt độ đua vào và nhiệt độ quy đổi tính theo độ C cũng sẽ thay đổi tương ứng. Các lệnh tạo ra giao diện này (***ct1_37.m***) là:

```
set(gcf, 'DefaultUicontrolUnit', 'Normalized')
frame_1 = uicontrol(gcf, 'Style', 'Frame', 'Position', [0.1 0.1  0.8 0.3]);
frame_2 = uicontrol(gcf, 'Style', 'Frame', 'Position', [0.1 0.6  0.8 0.3]);
set(frame_1, 'BackgroundColor' ,[0.5 0.5 0.5]);
set(frame_2, 'BackgroundColor', [0.5 0.5 0.5]);
text_f = uicontrol(gcf, 'Style', 'Text', 'String', 'Fahrenheit: ','Position',...
            [0.3 0.7 0.2 0.05], 'HorizontalAlignment', 'Left');
edit_f = uicontrol(gcf, 'Style',  'Edit',...
                    'String',   '168.0',,,,
```

```
                    'Position',  [0.6 0.7 0.1 0.05 ],...
                    'HorizontalAlignment',  'Right',...
                    'Callback',  'ct1_38');
text_c1 =  uicontrol(gcf,'Style',  'Text',...
                    'String',  'Celcius: ',...
                    'Position',  [0.3 0.3 0.2 0.05],...
                    'HorizontalAlignment',  'Left');
text_c2 =  uicontrol(gcf,'Style',  'Text',...
                    'String',  '100.0',...
                    'Position',  [0.6 0.3 0.1 0.05],...
                     'HorizontalAlignment',  'Left');
slider_f =  uicontrol(gcf,'Style',  'Slider',...
                     'Min',  32.0, 'Max',  212.0,...
                     'Value',  68.0,...
                     'Position',  [0.6 0.8 0.2 0.05],...
                  'Callback',  'ct1_39; ct1_38');
```

Để tạo thanh trượt ta dùng lệnh:

```
slider_f = uicontrol(gcf,'Style','Slider','Min',32.0,'Max',212.0,...
                    'Value',68.0,'Position',[0.6 0.8 0.2 0.05],...
               'Callback','ct1_39; ct1_38');
```

Như vậy Callback có thể gọi một chuỗi các lệnh MATLAB, phân cách nhau bằng dấu chấm than hay dấu phẩy. Chuỗi callback gọi ***ct1_39.m***:

```
f = get(slider_f,'Value');
f = num2str(f);
set(edit_f,'String',f,'CallBack','ct1_40; ct1_38');
```

với tác dụng nhập nhiệt độ giữ tại 'Value' của slider_f vào vị trí bên cạnh ô chứa chuỗi "Fahrenheit". Sau đó Callback gọi tiếp ***ct1_38.m*** để tính quy đổi giá trị nhiệt độ và gán vào ô cạnh chuỗi "Celcius". File ***ct1_40.m*** như sau:

```
f = get(edit_f,'String');
f = str2num(f);
set(slider_f,'Value',f);
```

có nhiệm vụ cập nhật giá trị giữ tại 'Value' của slider_f để rồi sau đó ***ct1_38.m*** làm nốt phần việc còn lại: tính đổi nhiệt độ và gán vào vị trí cạnh ô chứa chuỗi "Celcius".

4. Chọn lựa khi xuất số liệu:

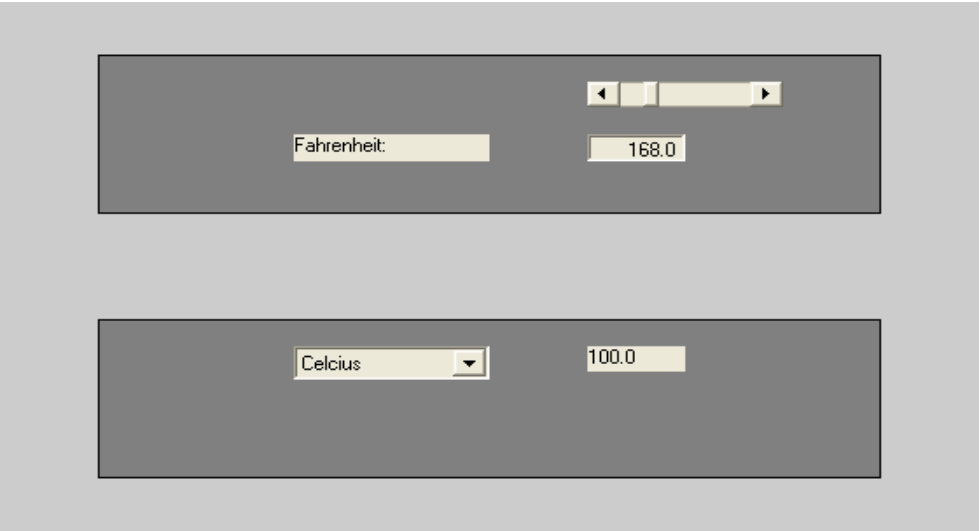
a. Khái niệm chung: Ngoài khả năng xuất dữ liệu cố định theo kiểu string hay kiểu số, ta có thể xuất dữ liệu theo một danh mục nào đó. Để minh hoạ, ta tạo file ***ct1_41.m*** như sau:

```
f = input('Nhap nhiet do: ');
r = f + 459.7;
c = (f - 32)*5/9;
k = c + 273.15;
choice = input(['Nhap 1 cho Rankie', '2 cho Celcius', '3 cho Kelvin: ']);
if choice = = 1
    fprintf(1, 'Nhiet do (do R) la: %g\n', r);
elseif choice = = 2
    fprintf(2, 'Nhiet do (do C) la: %g\n', c);
elseif choice = = 3
    fprintf(2, 'Nhiet do (do C) la: %g\n', c);
end
```

Từ cửa sổ lệnh, nhập lệnh ***ct1_41*** thì MATLAB sẽ hỏi nhiệt độ và đích quy đổi rồi hiển thị kết quả. Tuy nhiên công cụ GUI của MATLAB cho phép ta thực hiện việc lựa chọn thuận lợi hơn. Ta có thể chọn một trong 4 phương xuất dữ liệu sau đây:

- dùng popupmenu
- dùng list box
- dùng radio button
- dùng check box

b. Dùng popupmenu: Ta tạo ra giao diện như sau:



Các lệnh thực hiện công việc trên (***ct1_42.m***) là:

```
set(gcf, 'DefaultUicontrolUnit', 'Normalized')
frame_1 = uicontrol(gcf, 'Style', 'Frame',...
                    'Position', [0.1 0.1  0.8 0.3]);
frame_2 = uicontrol(gcf, 'Style', 'Frame',...
                    'Position', [0.1 0.6  0.8 0.3]);
set(frame_1, 'BackgroundColor', [0.5 0.5 0.5]);
set(frame_2, 'BackgroundColor' ,[0.5 0.5 0.5]);
text_f = uicontrol(gcf,'Style', 'Text',...
                   'String', 'Fahrenheit: ',...
                   'Position', [0.3 0.7 0.2 0.05],...
                   'HorizontalAlignment', 'Left');
edit_f = uicontrol(gcf,'Style', 'Edit',...
                   'String',...'168.0',...
                    'Position', [0.6 0.7 0.1 0.05 ],...
                    'HorizontalAlignment', 'Right',...
                    'Callback', 'ct1_38');
popup_c = uicontrol(gcf,...
                     'Style','Popupmenu',...
                  'String','Rankine|Celcius|Kelvin',...
                  'Value',2,...
                  'Position',[0.3 0.3 0.2 0.05],...
                  'Callback','ct1_43; ct1_45');
text_c2 =  uicontrol(gcf, 'Style', 'Text',...
```

```
                    'String',   '100.0',...
                    'Position',   [0.6 0.3 0.1 0.05],...
                    'HorizontalAlignment',   'Left');
slider_f = uicontrol(gcf, 'Style',   'Slider',...
                    'Min',  32.0, 'Max',  212.0,...
                     'Value',   68.0,...
                    'Position',   [0.6 0.8 0.2 0.05],...
                'Callback', 'ct1_39; ct1_45');
```

Khi kích chuột vào Popupmenu , có ba khả năng chọn lựa sẽ xuất hiện. Tiếp tục nháy chuột vào một trong 3 khả năng đó , ***Popupmenu*** biến mất chỉ còn lại đơn vị được chọn. Khi dùng chuột kéo thanh trượt ở frame phía trên, ta có được giá trị quy đổi sang đơn vị được chọn hiển thị ở phía dưới. Trong đoạn mã trên, giá trị 'Value' đặt sẵn là 2. Khi Callback gọi ***ct1_43.m***:

```
choice = get(popup_c,'Value');
```

thì giá trị của biến choice được đưa tới 'Value'. Sau đó Callback gọi tiếp ***ct1_45.m*** để xem kết quả giữ trong choice. File ***ct1_45.m*** như sau:

```
f = get(edit_f, 'String');
 f = str2num(f);
r = f  +  459.7;
 c = (f - 32)*5/9;
 k = c + 273.15;
choice = input(['Nhap 1 cho Rankie', '2 cho Celcius', '3 cho Kelvin: ']);
if choice = = 1
  t = r;
elseif choice = = 2
  t = c;
elseif choice = = 3
   t = k
end
t = num2str(t);
set(text_c2, 'String',t);
```

Bằng cách thay '***Popupmenu***' bằng '***Radiobutton***' uicontrol ta có phương án ***Radiobutton***. Giao diện sẽ có dạng:

Các lệnh thực hiện công việc này (***ct1_46.m***) là:

```
set(gcf, 'DefaultUicontrolUnit',   'Normalized')
frame_1 = uicontrol(gcf, 'Style', 'Frame', 'Position',  [0.1 0.1  0.8 0.3]);
frame_2 = uicontrol(gcf, 'Style', 'Frame', 'Position',  [0.1 0.6  0.8 0.3]);
set(frame_1,'BackgroundColor',  [0.5 0.5 0.5]);
set(frame_2,'BackgroundColor',  [0.5 0.5 0.5]);
text_f = uicontrol(gcf, 'Style',  'Text', 'String',  'Fahrenheit: ','Position',...
                        [0.3 0.7 0.2 0.05], 'HorizontalAlignment', 'Left');
edit_f = uicontrol(gcf, 'Style', 'Edit', 'String','168.0', 'Position',...
                       [0.6 0.7 0.1 0.05 ], 'HorizontalAlignment',...
                       'Right', 'Callback','ct1_41');
strings = ['Rankine'; 'Celcius'; 'Kelvine'];
show   = [    0;       1;        0];
ys     = [    3;       2;        1]*0.075 + 0.075;
for i = 1:3
   radio_c(i) = uicontrol(gcf,...
                              'Style',  'Radiobutton',...
```

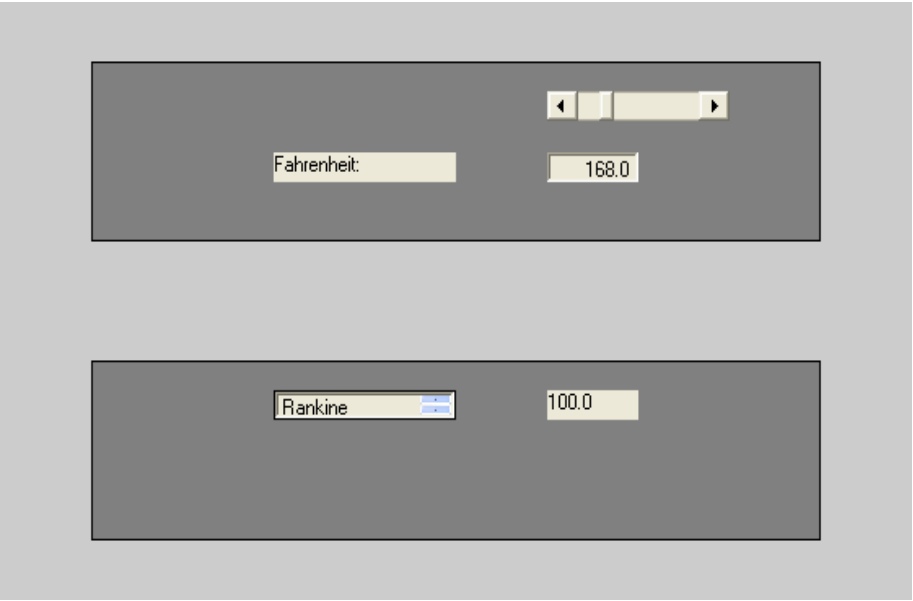
```
                    'String',    strings(i),...
                    'Value',     show(i),...
                          'Position',  [0.3 ys(i)  0.2 0.05],...
                    'Callback', 'ct1_47; ct1_45');
end
text_c2= uicontrol(gcf, 'Style', 'Text', 'String','100.0', 'Position',...
            [0.6 0.3 0.1 0.05], 'HorizontalAlignment', 'Left');
slider_f = uicontrol(gcf, 'Style', 'Slider', 'Min',32.0, 'Max', 212.0,...
                'Value', 68.0, 'Position', [0.6 0.8 0.2 0.05],...
                     'Callback', 'ct1_39; ct1_45');
```

File ***ct1_47.m***:

```
for i = 1:3
   if gcbo = = radio_c(i)
            choice = i;
      set(radio_c(i), 'Value', 1);
  elseif
      set(radio_c(i), 'Value', 0);
        end;
end;
```

Đoạn lệnh trên là một vòng lặp, so sánh số (handle) Callback thu được (giá trị do hàm gcbo trả về) với handle của mỗi nút. Nút nào có số trùng sẽ được đóng (turn on, 'Value' = 1) và nút nào khác số sẽ bị ngắt (turn off,'Value' = 0). Cuối cùng Callback gọi ***ct1_45.m*** để thực hiện việc tính quy đổi được chọn và hiển thị kết quả. Điểm khác duy nhất là khi chọn, Popupmenu chỉ chứa một phần tử thì radiobutton có thể đồng thời chứa nhiều phần tử.

Cuối cùng ta xét phương án dùng ***listbox***. Giao diện cần tạo như sau:

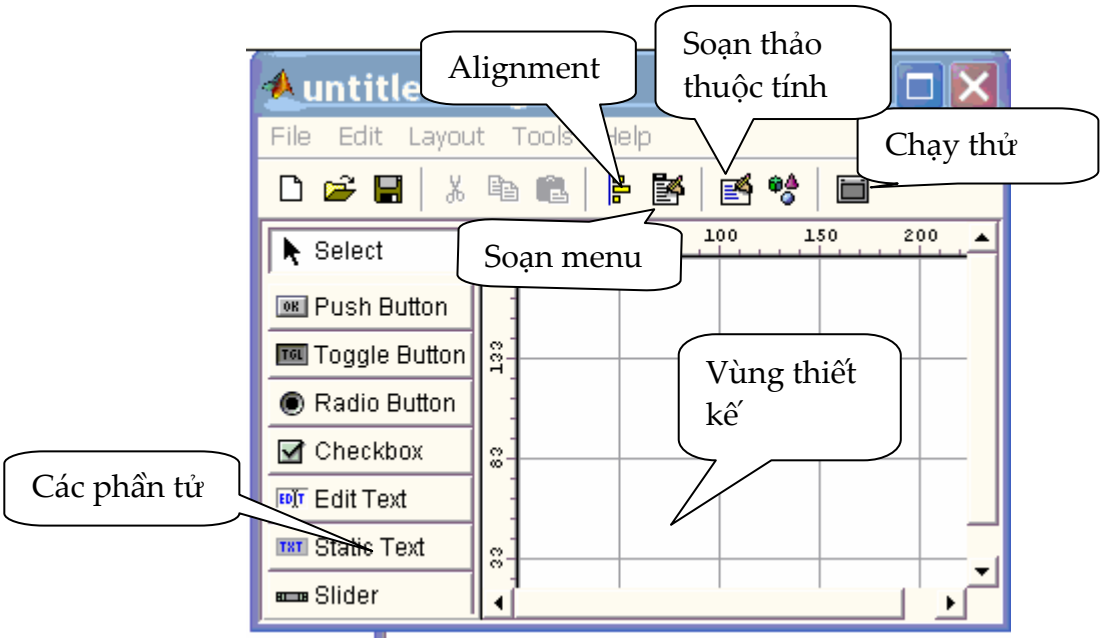


Các mã tạo ra giao diện trên (***ct1_48.m***) là:

```
set(gcf, 'DefaultUicontrolUnit', 'Normalized')
frame_1 = uicontrol(gcf, 'Style', 'Frame', 'Position', [0.1 0.1  0.8 0.3]);
frame_2 = uicontrol(gcf, 'Style', 'Frame', 'Position', [0.1 0.6  0.8 0.3]);
set(frame_1, 'BackgroundColor', [0.5 0.5 0.5]);
set(frame_2, 'BackgroundColor', [0.5 0.5 0.5]);
text_f = uicontrol(gcf, 'Style', 'Text', 'String', 'Fahrenheit: ', 'Position',...
            [0.3 0.7 0.2 0.05], 'HorizontalAlignment', 'Left');
edit_f = uicontrol(gcf, 'Style', 'Edit', 'String', '168.0', 'Position',...
            [0.6 0.7 0.1 0.05 ], 'HorizontalAlignment',...
            'Right', 'Callback', 'ct1_38');
listbox_c = uicontrol(gcf,...
            'Style', 'Listbox',...
            'String', 'Rankine|Celcius|Kelvin',...
            'Value', 2,...
            'Position', [0.3 0.3 0.2 0.05],...
            'Callback', 'ct1_49;ct1_45');
text_c2 =  uicontrol(gcf, 'Style', 'Text', 'String', '100.0', 'Position',...
            [0.6 0.3 0.1 0.05], 'HorizontalAlignment', 'Left');
slider_f = uicontrol(gcf, 'Style', 'Slider', 'Min',32.0, 'Max', 212.0,...
              'Value', 68.0, 'Position', [0.6 0.8 0.2 0.05],...
                    'Callback', 'ct1_39; ct1_45');
```

5. Công cụ đồ hoạ tạo GUI

a. Tạo GUI bằng công cụ đồ hoạ: Trên đây ta đã xem xét cách tạo GUI bằng phương pháp thủ công. Ta có thể tạo GUI bằng công cụ đồ hoạ. Khi nhập lệnh guide ta gọi trình đồ hoạ (Graphics User Interface Development Environment) để soạn thảo layout. Kết quả đầu tiên là ta có một layout rỗng như sau:

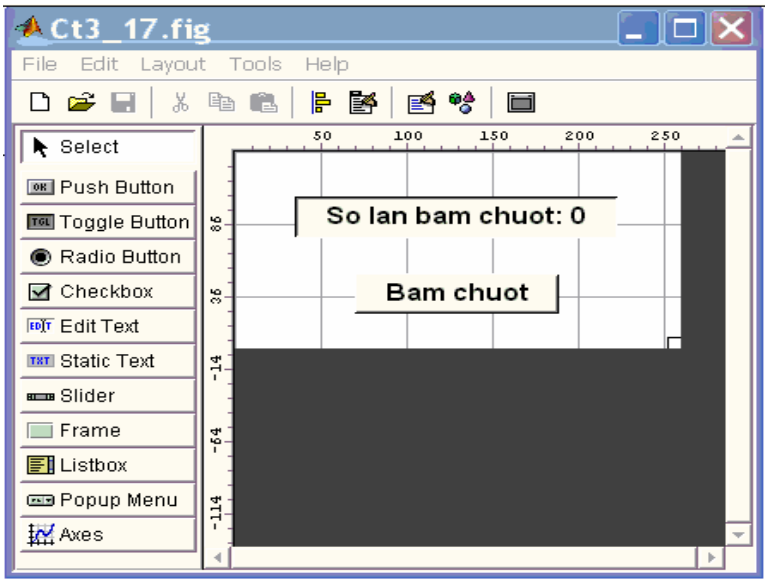


Việc đầu tiên là ta thiết kế giao diện mong muốn. Ta sẽ dùng chuột kéo các phần tử cần dùng từ bên trái và thả vào layout rỗng bên phải. Ta có thể dịch chuyển các phần tử này đế các vị trí mong muốn và cân chỉnh bằng công cụ Alignment. Với mỗi phần tử ta cấn xác định thuộc tính cho nó bằng cách bấm đúp vào phần tử hay bấm vào công cụ soạn thảo thộc tính

Sau khi thiết kế xong ta lưu nó lại. Lúc này MATLAB tự động tạo ra file *.fig dùng lưu giao diện vừa tạo và file *.m chưa các mã lệnh cần thực hiện. Việc cuối cùng là viết các mã lệnh vào file *.m. Trong quá trình thiết kế ta có thể chạy thử xem sau mỗi bước thiết kế đã đạt yêu cầu chưa bằng cách bấm vào ô chạy thử

b. Một số ví dụ tạo GUI:

☞ ***Đếm số lần bấm chuột***: Ta thiết kế một giao diện như sau:



Ta muốn là khi bấm chuột, số lần bấm sẽ được đếm và ghi lại. Trước hết ta gọi guide và có được một layout rỗng. Vào Property Inspector (ô soạn thảo thuộc tính) và ghi vào Name chuỗi "ct1_52" và chấp nhận thuộc tích Tag mặc định của nó là figure1; dùng Font chữ mặc định, cỡ chữ 12, bold. Ta dùng ô Edit Text để ghi lại số lần bấm. Ta vào Property Inspector rồi chọn String. Ta nhập vào ô này chuỗi "So lan bam chuot: 0". Ta ghi vào ô Tag chuỗi "editmot" và cũng dùng Font chữ mắc định, cỡ chữ 12 và bold. Tiếp theo kéo Pushbutton vào layout và soạn thảo thuộc tính cho nó với Font chữ mặc định, cỡ chứ 12, bold. Trong thuôc tính String ghi chuỗi " Bam chuot"; ghi và Tag chuỗi "pushbuttonmot". Như vậy là ta đã thiết kế xong. Bây giờ ta lưu lại với tên là ***ct1_52.fig*** và ***ct1_52.m***.

Nhiệm vụ tiếp theo là ghi các lệnh cần thiết vào file ***ct1_52.m***. File này đã được MATLAB tự động tạo ra. Ta phải thêm vào đó các mã lệnh để khi bấm chuột thì số lần bấm được thể hiện trên ô Edit Text. Ta sẽ ghi các mã lệnh này vào phần:

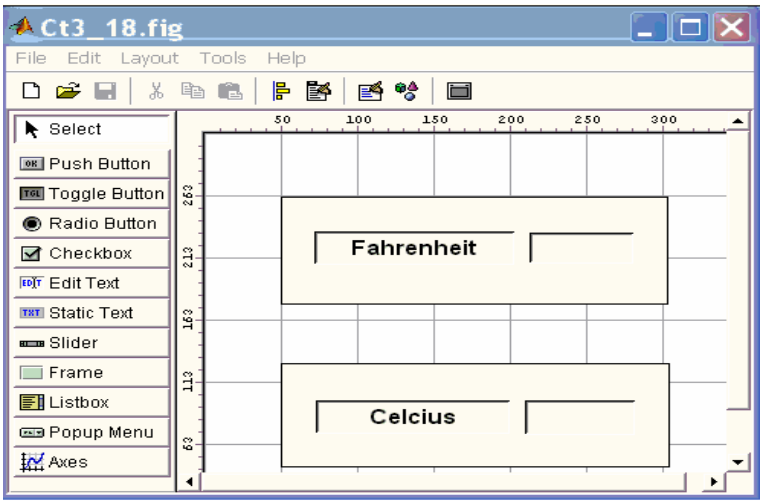
function varargout = pushbuttonmot_Callback(h, eventdata, handles, varargin)

do lệnh cần được thực hiện khi gọi pushbutton. Nội dung của ***ct1_52.m*** là:

```
function varargout = Ct1_52(varargin)
if nargin = = 0
        fig = openfig(mfilename,'reuse');
        set(fig, 'Color', get(0, 'defaultUicontrolBackgroundColor'));
```

```
        handles = guihandles(fig);
        guidata(fig, handles);
        if nargout > 0
            varargout{1} = fig;
        end
    elseif
        ischar(varargin{1})
        try
            [varargout{1:nargout}] = feval(varargin{:});
        catch
            disp(lasterr);
        end
    end
    function varargout = pushbuttonmot_Callback(h, eventdata, handles, varargin)
    persistent dem;%bien dem la persistent de no ton tai giua lan goi ham
    if isempty(dem)
       dem = 0;
    end
    dem = dem  + 1;
    str = sprintf('So lan bam chuot: %d',dem);
    set(handles.editmot,'String',str);
```

☞ ***Chuyển đổi từ độ Fahrenheit sang độ Celcius***: Ta thiết kế một GUI để chuyển đổi nhiệt độ. Giao diện có dạng như sau:



Thuộc tính của Layout được ghi Name: ***ct1_53*** còn các thuộc tính khác là mặc định.

Ta dùng hai Frame với các Tag là frmmot và frame2. Các thuộc tính khác chấp nhận giá trị mặc định.

Edit Text thứ nhất có các thuộc tính FontName: Arial, FontSize: demi, FontWeight: demi, String: Fahrenheit, Tag: editmot còn các thuộc tính khác là mặc định.

Edit Text thứ hai có các thuộc tính FontName: Arial, FontSize: demi, FontWeight: demi, String: để trống, Tag: edithai còn các thuộc tính khác là mặc định.

Edit Text thứ ba có các thuộc tính FontName: Arial, FontSize: demi, FontWeight: demi, String: Celcius, Tag: editba còn các thuộc tính khác là mặc định.

Edit Text thứ tư có các thuộc tính FontName: Arial, FontSize: demi, FontWeight: demi, String: để trống, Tag: editbon còn các thuộc tính khác là mặc định.

Sau khi thiết kế xong, lưu nó với tên ***ct3_18.fig***. MATLAB tạo thêm ***ct1_53.m***. Bây giờ ta cần viết mã cho nó. Nhiệm vụ của đoạn mã là khi ta nhập nhiệt độ Fahrenheit vào ô Edit text thứ hai thì trong ô Edit Text thứ 4 phải xuất hiện giá trị nhiệt độ Celcius tương ứng. Do vậy nội dung của ***ct1_53.m*** là:

```
function varargout = Ct1_53(varargin)
if nargin == 0  % LAUNCH GUI
        fig = openfig(mfilename,'reuse');
        set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));
        handles = guihandles(fig);
        guidata(fig, handles);
        if nargout > 0
                varargout{1} = fig;
        end
elseif ischar(varargin{1})
        try
                [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
        catch
                disp(lasterr);
        end
end
function varargout = edithai_Callback(h, eventdata, handles, varargin)
f = get(handles.edithai,'String');
```

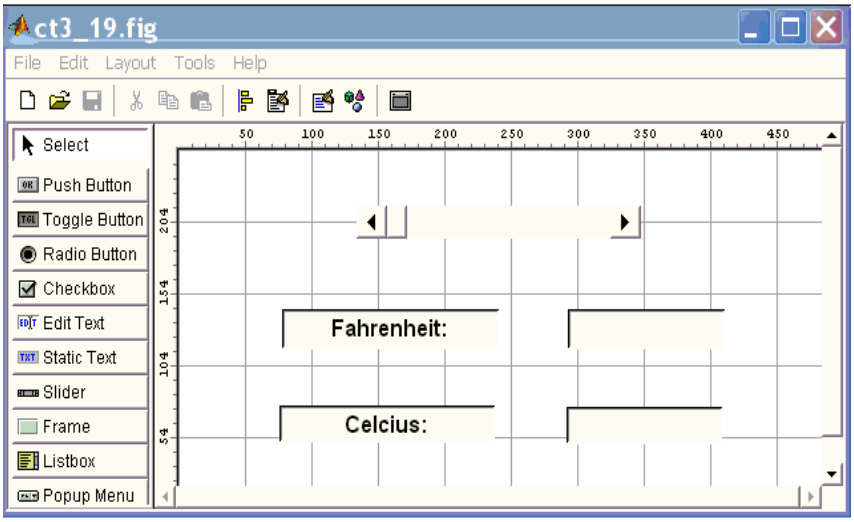
f = str2num(f);
*c = (f - 32)*5/9;*
c = num2str(c);
set(handles.editbon,'String',c);

Trong đó đoạn mã cần viết nằm trong đoạn:

function varargout = edithai_Callback(h, evendata, handles, varargin)

Các lệnh khác là do MATLAB tự động tạo ra.

☞ ***Dùng slider để nhập số liệu***: Ta dùng ví dụ chuyển đổi nhiệt độ trên nhưng bây giờ sẽ thêm slider để thay đổi nhiệt độ đầu vào. Giao diện sẽ có dạng:



Như vậy ta cần 5 phần tử, trong đó có một phần tử là slider và 4 phần tử Edit Text.

Layout có thuộc tính Name: ***ct1_54***, còn các thuộc tính khác ta chấp nhận giá trị mặc định.

Slider có thuộc tính Max: 1.0 và Min: 0.0.

Edit Text thứ nhất có thuộc tính FontSize: 12, FontWeight: bold, String: Fahrenheit còn các thuộc tính khác chấp nhận giá trị mặc định.

Edit Text thứ 2 có thuộc tính FontSize: 12, FontWeight: bold, String: để trống.

Edit Text thứ 3 có thuộc tính FontSize: 12, FontWeight: bold, String: Celcius.

Edit Text thứ 4 có thuộc tính FontSize: 12, FontWeight: bold, String: để
trống. (Các thuộc tính mà ta không nhắc đến có nghĩa là chấp nhận giá trị mặc định).

Layout được lưu với tên ***ct1_54.fig***.

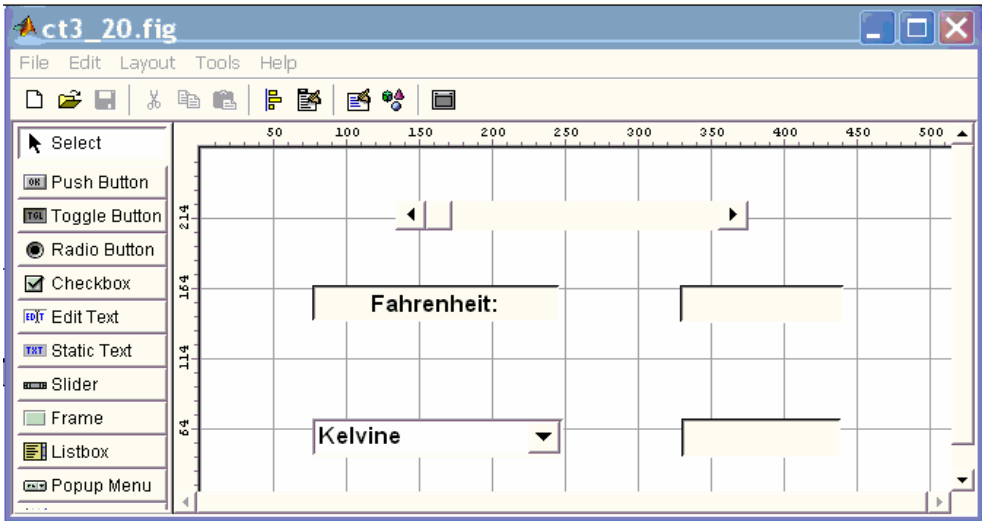
Bây giờ ta viết mã cho phần ***ct1_54.m*** mà MATLAB đã tự động tạo ra. Nhiệm vụ của nó là nhận giá trị thay đổi từ con trượt, cập nhật cho Edit Text 2 và Edit Text 4. Ta có nội dung của ***ct1_54.m***:

```
function varargout = ct1_54(varargin)
if nargin = = 0
        fig = openfig(mfilename,'reuse');
        handles = guihandles(fig);
        guidata(fig, handles);
        if nargout > 0
                varargout{1} = fig;
        end
elseif ischar(varargin{1})
        try
                [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
        catch
                disp(lasterr);
        end
end

function varargout = slider1_Callback(h, eventdata, handles, varargin)
f = get(handles.slider1,'Value');%nhan gia tri tu con truot
f = f*180 + 32;%tinh ra do Fahrenheit
a = num2str(f);%bien lai thanh chuoi
set(handles.edit2,'String',a);%ghi vao o do Fahrenheit
b = (f-32)*5/9;%doi thanh do Celcius
b = num2str(b);%doi lai thanh chuoi
set(handles.edit4,'String',b);%ghi vao o do Celcius
```

☞ ***Xuất số liệu có lựa chọn***: Ta vẫn dùng ví dụ trên nhưng bây giờ nhiệt độ quy đổi có thể được tính theo thang nhiệt độ Kenvine, Celcius hay

Rankine. Để có thể chọn lựa ta dùng một trong các phương án: Popupmenu, Rdiobutton, Listbox hay Checkbox. Giao diện khi dùng Popupmenu như sau:



Như vậy là ta cần một Slider, ba Edit Text và một Popupmenu. Layout có thuộc tính Name: ***ct13_55***.

Slider có thuộc tính Max: 1 và Min: 0

Edit Text thứ nhất có thuộc tính FontSize: 12, FontWeight: bold và String: Fahrenheit.

Edit Text thứ hai có thuộc tính FontSize: 12, FontWeight: bold và String để trống.

Edit Text thứ 3 có thuộc tính FontSize: 12, FontWeight: bold và String để trống.

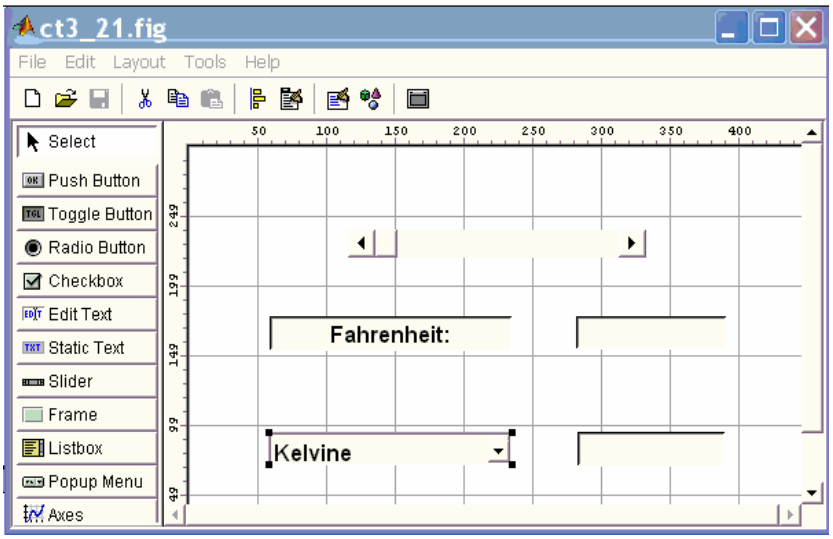
Popupmenu có thuộc tính FontSize: 12, FontWeight: bold. Để ghi vào thuộc tính String ta bấm đúp chuột vào icon của nó và viết 3 dòng: Kelvine, Celcius và Rankine.

File được lưu với tên ***ct1_55.fig***. Vấn đề còn lại là viết mã trong file ***ct1_55.m***. Mã cần thực hiện nhận giá trị từ Slider, xem Popupmenu nào được chọn để hiển thị nhiệt độ tương ứng. File ***ct1_55.m*** như sau:

```
function varargout = ct1_55(varargin)
if nargin == 0  % LAUNCH GUI
        fig = openfig(mfilename,'reuse');
        set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));
        handles = guihandles(fig);
        guidata(fig, handles);
```

```
        if nargout > 0
                varargout{1} = fig;
        end
elseif ischar(varargin{1})
        try
                [varargout{1:nargout}] = feval(varargin{:});
        catch
                disp(lasterr);
        end
end
function varargout = slider1_Callback(h, eventdata, handles, varargin)
f = get(handles.slider1,'Value');
f = f*180 + 32;
a = num2str(f);
set(handles.edit2,'String',a);
r = f + 495.7;
c = (f - 32)*5/9;
k = c + 273.15;
chon = get(handles.popupmenu1,'Value');
if chon = = 1
   t = k;
elseif chon = = 2
   t = c;
elseif chon = = 3
   t = r;
end
t = num2str(t);
set(handles.edit3,'String',t);
```

Tiếp theo ta xét trường hợp dùng listbox. Thay vì dùng Popupmenu ta dùng Listbox. Các phần tử khác và thuộc tính của nó không thay đổi. Thuộc tính Name của Layout là ***ct1_56***. Ta vào ô String của Listbox và ghi vào đó 3 dòng Kelvine, Celcius và Rankine. Giao diện như sau:



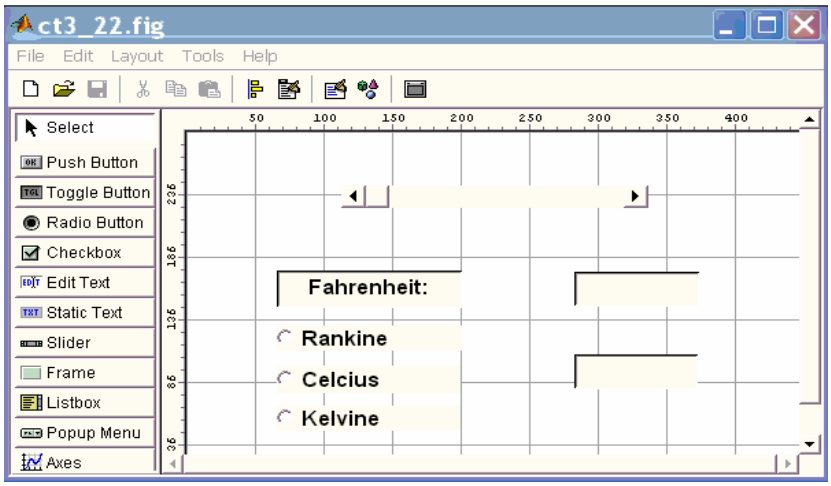
File được lưu với tên ***ct1_56.fig***. Tiếp theo viết lệnh cho ***ct1_56.m***. Ta có file này như sau:

```
function varargout = ct1_56(varargin)
if nargin = =
        fig = openfig(mfilename,'reuse');
        set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));
        handles = guihandles(fig);
        guidata(fig, handles);
        if nargout > 0
            varargout{1} = fig;
        end
elseif ischar(varargin{1})
        try
            [varargout{1:nargout}] = feval(varargin{:});
        catch
            disp(lasterr);
        end
end

function varargout = slider1_Callback(h, eventdata, handles, varargin)
f = get(handles.slider1,'Value');
f = f*180 + 32;
a = num2str(f);
set(handles.edit2,'String',a);
r = f + 495.7;
```

```
c = (f - 32)*5/9;
k = c + 273.15;
chon = get(handles.listbox1,'Value');
if chon = = 1
   t = k;
elseif chon = = 2
   t = c;
elseif chon = = 3
   t = r;
end
t = num2str(t);
set(handles.edit3,'String',t);
```

Ta tiếp tục xét phương án dùng Radiobutton. Giao diện có dạng:



Ta dùng ba Radiobutton thay cho Listbox. Radiobutton thứ nhất có thuộc tính FontSize: 12, FontWeight: bold và String: Rankine. Radiobutton thứ 2 có thuộc tính FontSize: 12, FontWeight: bold và String: Celcius. Radibutton thứ 3 có thuộc tính FontSize: 12, FontWeight: bold và String: Kelvine. Các phần tử khác và thuộc tính của chúng vẫn như cũ. Layout có thuộc tính Name: ***ct1_57***. Lưu GUI với tên ***ct1_57.fig***.

Tiếp theo ta viết các mã lệnh trong ***ct1_57.m***:

```
function varargout = ct1_57(varargin)
if nargin = = 0
    fig = openfig(mfilename,'reuse');
    set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));
    handles = guihandles(fig);
```

```
    guidata(fig, handles);
    if nargout > 0
      varargout{1} = fig;
    end
elseif ischar(varargin{1})
    try
        [varargout{1:nargout}] = feval(varargin{:});  catch
        disp(lasterr);
    end
end
function mutual_exclude(off)
set(off,'Value',0);
function varargout = slider1_Callback(h, eventdata, handles, varargin)
global chon
f = get(handles.slider1,'Value');
f = f*180 + 32;
a = num2str(f);
set(handles.edit2,'String',a);
r = f + 495.7;
c = (f - 32)*5/9;
k = c + 273.15;
if chon = = 1
   t = r;
elseif chon = = 2
   t = c;
elseif chon == 3
   t = k;
end
t = num2str(t);
set(handles.edit3,'String',t);
function varargout = radiobutton1_Callback(h, eventdata, handles, varargin)
global chon;
off = [handles.radiobutton2, handles.radiobutton3];
mutual_exclude(off);
chon = 1;
function varargout = radiobutton2_Callback(h, eventdata, handles, varargin)
global chon;
off = [handles.radiobutton1, handles.radiobutton3];
```

```
mutual_exclude(off);
chon = 2;
function varargout = radiobutton3_Callback(h, eventdata, handles, varargin)
global chon;
off = [handles.radiobutton1, handles.radiobutton2];
mutual_exclude(off);
chon = 3;
```

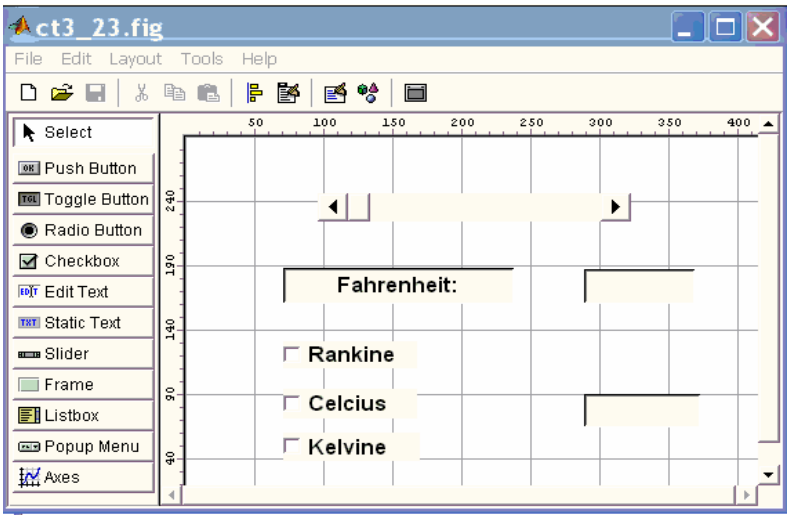
□o□n l□nh:

```
function mutual_exclude(off)
set(off,'Value',0);
```

l□m cho 3 nỳt l□nh tr□ th□nh m□t nhúm. Cỏc cõu l□nh:

```
off = [handles.radiobutton1, handles.radiobutton2];
mutual_exclude(off);
```

l□m cho khi ch□n m□t nỳt Radiobutton n□y thỡ khụng ch□n □□□c nỳt khỏc n□a. Cu□i cựng ta xột ph□□ng ỏn dựng Checkbox. Giao di□n nh□ sau:



C□ng nh□ ph□□ng ỏn dựng Radiobutton, ta dựng ba Checkbox.

Checkbox th□ nh□t cú thu□c tớnh FontSize: 12, F□ntWeight: bold v□ String: Rankine.

Checkbox th□ hai cú thu□c tớnh FontSize: 12, F□ntWeight: bold v□ String: Celcius.

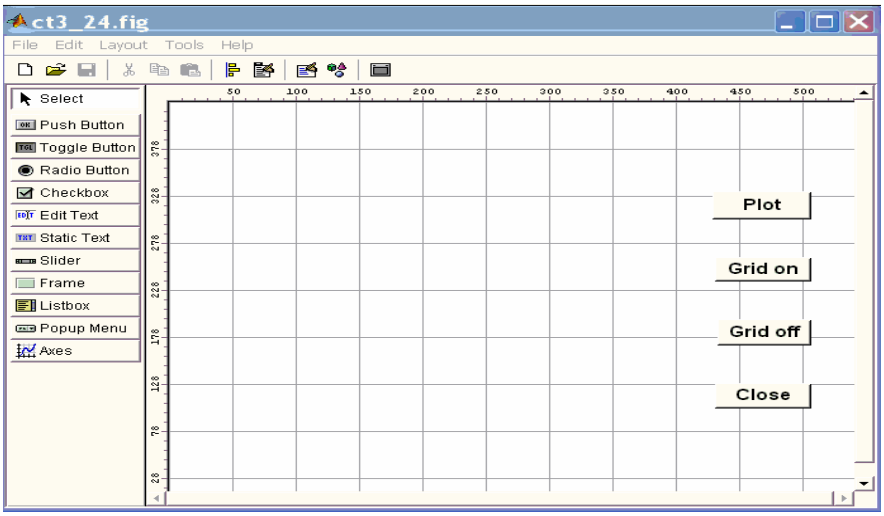
Checkbox th□ ba cú thu□c tớnh FontSize: 12, F□ntWeight: bold v□ String: Kelvine.

Cỏc ph□n t□ khỏc khụng co gỡ thay □□i. Ta l□u file v□i tờn ***ct1_58.fig***. Ti□p theo ta vi□t mó l□nh cho ***ct1_58.m***:

```
function varargout = ct1_58(varargin)
if nargin = = 0
fig = openfig(mfilename,'reuse');
        set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));
handles = guihandles(fig);
guidata(fig, handles);
```

```
if nargout > 0
            varargout{1} = fig;
end
elseif ischar(varargin{1})
try
        [varargout{1:nargout}] = feval(varargin{:});    catch
            disp(lasterr);
end
end
function mutual_exclude(off)
set(off,'Value',0);
function varargout = slider1_Callback(h, eventdata, handles, varargin)
global chon
f = get(handles.slider1,'Value');
f = f*180 + 32;
a = num2str(f);
set(handles.edit2,'String',a);
r = f + 495.7;
c = (f - 32)*5/9;
k = c + 273.15;
if chon = = 1
   t = r;
elseif chon = = 2
   t = c;
elseif chon = = 3
   t = k;
end
t = num2str(t);
set(handles.edit3,'String',t);
function varargout = checkbox1_Callback(h, eventdata, handles, varargin)
global chon;
off = [handles.checkbox2, handles.checkbox3];
mutual_exclude(off);
chon = 1;
function varargout = checkbox2_Callback(h, eventdata, handles, varargin)
global chon;
off = [handles.checkbox1, handles.checkbox3];
mutual_exclude(off);
chon = 2;
function varargout = checkbox3_Callback(h, eventdata, handles, varargin)
global chon;
off = [handles.checkbox2, handles.checkbox1];
mutual_exclude(off);
chon = 3;
```

☞ ***GUI cú dựng □□ ho□***: Ta xõy d□ng m□t GUI dựng □□ v□ □□ th□ h□m y=tsin(t). Giao di□n nh□ sau:



Ta dựng m□t Axes, b□n Pushbutton □□ t□o nờn giao di□n n□y. Khi nh□n Plot, □□ th□ c□a h□m y = tsin(t) □□□c v□. Khi nh□n Grid on, □□ th□ □□□c chia l□□i. Khi nh□n Grod off, l□□i b□ xoỏ. Nh□n Close □□ □úng □□ th□.

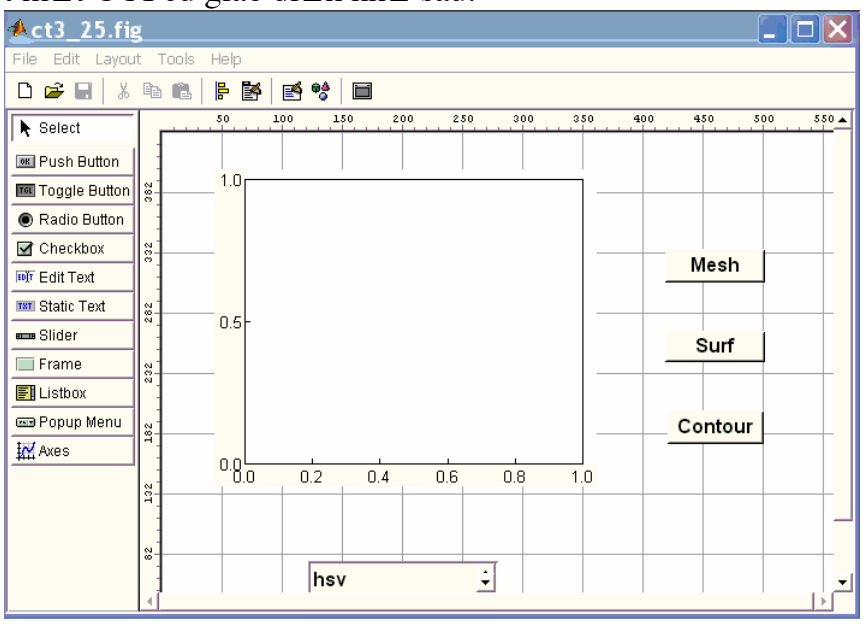
Layout cú thu□c tớnh Name: ***ct1_59***, HandleVisibility: callback.

Các Pushbutton đều có thuộc tính FontSize: 12, FontWeight: bold và các String là các tên lệnh. GUI được lưu với tên file là ***ct1_59.fig***. Tiếp theo ta soạn thảo lệnh cho ***ct1_59.m***:

```
function varargout = ct1_59(varargin)
if nargin = = 0
        fig = openfig(mfilename,'reuse');
        handles = guihandles(fig);
        guidata(fig, handles);
        if nargout > 0
                varargout{1} = fig;
        end
elseif ischar(varargin{1})
        try
                [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
        catch
                disp(lasterr);
        end
end
function varargout = pushbutton1_Callback(h, eventdata, handles, varargin)
grid on
function varargout = pushbutton2_Callback(h, eventdata, handles, varargin)
grid off
```

function varargout = pushbutton3_Callback(h, eventdata, handles, varargin)
close
function varargout = pushbutton4_Callback(h, eventdata, handles, varargin)
t = 0:0.01:20;
*y = t.*sin(t);*
plot(t,y);

Ti□p theo ta xột m□t GUI cú giao di□n nh□ sau:



Nhi□m v□ c□a GUI l□ v□ □□ th□ c□a h□m peaks theo cỏc d□ng khỏc nhau(mesh, surf v□ contour) v□i cỏc Colormap khỏc nhau(hsv, hot, gray, prism, cool, winter v□ summer). Vi□c v□ cỏc d□ng □□ th□ th□c hi□n nh□ cỏc Pushbutton. Vi□c ch□n Colormap th□c hi□n nh□ Listbox.

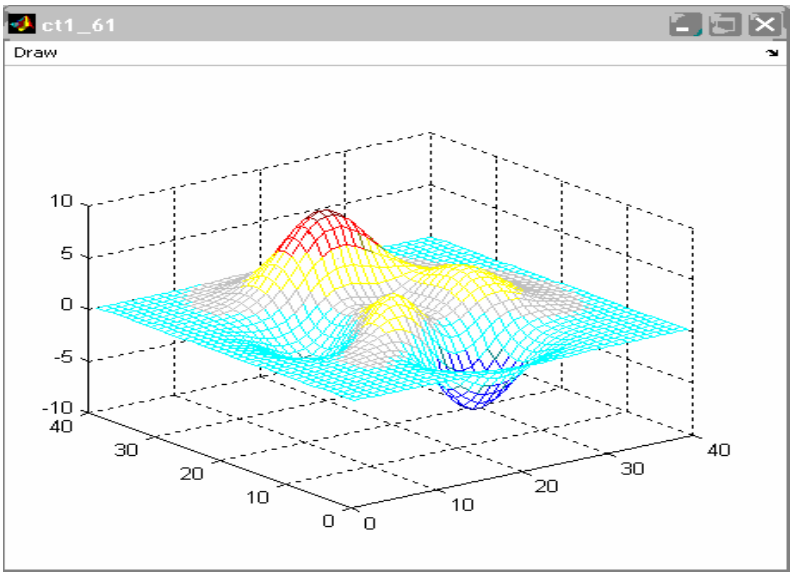
Layout cú thu□c tớnh Name: ***ct1_60*** v□ thu□c tớnh HandleVisbility: on. Cỏc Pushbutton □□u cú thu□c tớnh FontSize: 12 v□ F□ntWeight: bold. Ta l□u GUI v□i tờn ***ct1_60.fig***. Mó trong ***ct1_60.m*** g□m:

```
function varargout = ct1_60(varargin)
if nargin = = 0
        fig = openfig(mfilename,'reuse');
        set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));
        handles = guihandles(fig);
        guidata(fig, handles);
        if nargout > 0
            varargout{1} = fig;
        end
elseif ischar(varargin{1})
        try
            [varargout{1:nargout}] = feval(varargin{:});
        catch
```

```
            disp(lasterr);
        end
end
function varargout = pushbutton1_Callback(h, eventdata, handles, varargin)
z = peaks(40);
chon = get(handles.listbox1,'Value');
if chon = =1
   colormap(hsv(256));
elseif chon = =2
   colormap(hot(256));
elseif chon = =3
   colormap(gray(256));
elseif chon = =4
   colormap(prism(256));
elseif chon = =5
   colormap(cool(256));
elseif chon = =6
   colormap(winter(256));
elseif chon = =7
   colormap(summer(256));
end
mesh(z);
function varargout = pushbutton2_Callback(h, eventdata, handles, varargin)
z = peaks(40);
chon = get(handles.listbox1,'Value');
if chon = =1
   colormap(hsv(256));
elseif chon = =2
   colormap(hot(256));
elseif chon = =3
   colormap(gray(256));
elseif chon = =4
   colormap(prism(256));
elseif chon = =5
   colormap(cool(256));
elseif chon = =6
   colormap(winter(256));
elseif chon = =7
   colormap(summer(256));
end
surf(z);
function varargout = pushbutton3_Callback(h, eventdata, handles, varargin)
z = peaks(40);
chon = get(handles.listbox1,'Value');
if chon = =1
   colormap(hsv(256));
```

```
elseif chon = =2
   colormap(hot(256));
elseif chon = =3
   colormap(gray(256));
elseif chon = = 4
   colormap(prism(256));
elseif chon = = 5
   colormap(cool(256));
elseif chon = = 6
   colormap(winter(256));
elseif chon = = 7
   colormap(summer(256));
end
contour(z);
```

☞ ***GUI có dùng đồ hoạ***: Ta xây dựng một GUI dùng menu. Giao diện của GUI như sau:



Menu Draw gồm các menu con Mesh, Contour và Close. GUI được lưu trong file ***ct1_61.fig*** và chương trình được lưu trong file ***ct1_61.m***:

```
function varargout = ct1_61(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @ct1_61_OpeningFcn, ...
                   'gui_OutputFcn',  @ct1_61_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
```

```
                'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
handles.output = hObject;
function varargout = ct1_61_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
function mnumesh_Callback(hObject, eventdata, handles)
z = peaks(40);
mesh(z);
function Untitled_3_Callback(hObject, eventdata, handles)
z = peaks(40);
contour(z);
function mnuclose_Callback(hObject, eventdata, handles)
clf
close
function mnudraw_Callback(hObject, eventdata, handles)
```