

知能機械設計演習
Practicum in Intelligent Machine Design

MATLAB/Simulinkの基礎1
MATLAB/Simulink Basics Tutorial 1

生命体工学研究科
人間知能システム工学専攻
s-yasukawa@brain.kyutech.ac.jp
安川 真輔
Shinsuke Yasukawa

Outline

Third period

13:00-13:10 Introduction

13:10-13:50 MATLABの基礎

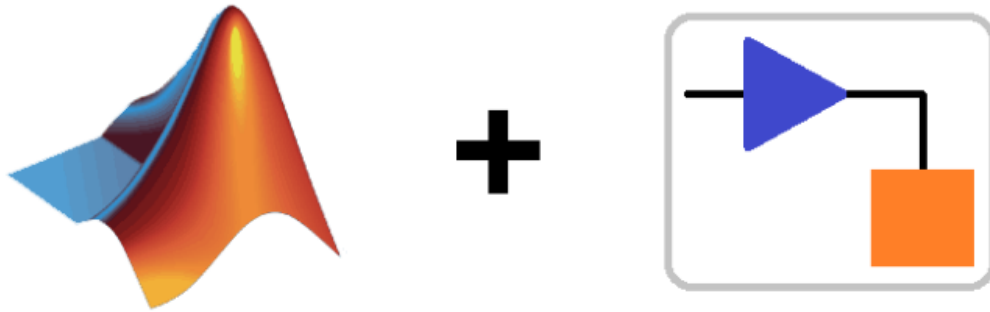
13:50-14:30 GUIプログラミング

Fourth period

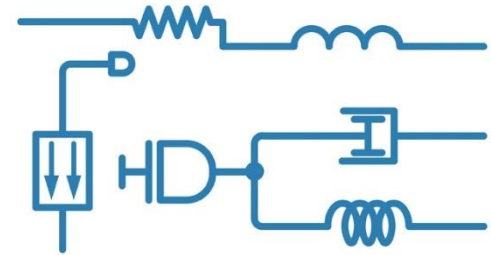
14:40-15:25 Simulinkの基礎

15:25-16:10 M-ファイル S-function

About MATLAB/Simulink



MATLAB
SIMULINK®



multidomain
model

MATLAB, the language of technical computing, is a programming environment for algorithm development, data analysis, visualization, and numeric computation.

Simulink is a graphical environment for simulation and Model-Based Design of multidomain dynamic and embedded systems.

MATLABの基礎

1. コマンド操作

- A. 基本演算, 変数の確認
- B. 基本コマンド (clear, clc)
- C. 組み込み変数・組み込み関数 (sin, cosなど)
- D. 配列の宣言と演算
- E. データの保存と読み込み
- F. データのプロット

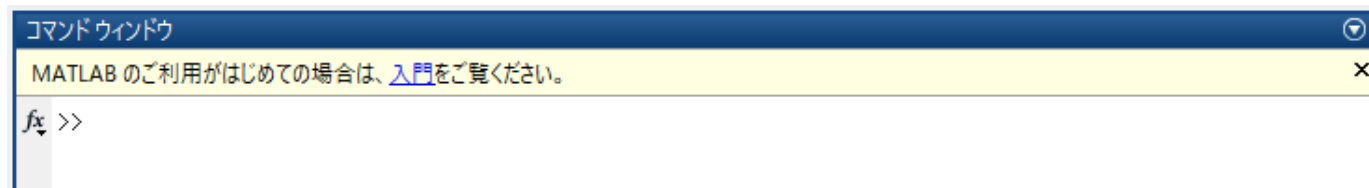
2. MATLABエディタ

- A. スクリプトの書き方
- B. 関数化
- C. ODEソルバー

3. GUI プログラミング

コマンド操作

コマンド ウィンドウで MATLAB のプロンプト (`>>`) の後にコマンド名を入力して Enter キーを押すと、そのコマンドを実行できる。



- A. 基本演算, 変数の確認
- B. 基本コマンド (clear, clc)
- C. 組み込み変数・組み込み関数 (sin, Cosなど)
- D. 配列の宣言と演算
- E. データの保存と読み込み
- F. データのプロット

コマンド操作 -A. 基本演算, 変数の確認-

1. $3*5$

2. $m=3*5$

3. $m=m+1$

4. $k=5$

5. $k=m+k$

コマンド操作 -B. 基本コマンド (clear, clc) -

1. clear

Remove items from workspace, freeing up system memory

ワークスペースからアイテムを削除し、
システムメモリを解放する

2. clc

Clear Command Window

コマンド ウィンドウのクリア

コマンド操作 -C. 組み込み変数・組み込み関数 (sin, cosなど) -

```
a = pi
```

```
a = sin(-5)
```

```
a=sqrt(2)
```


コマンド操作 -D. 配列の宣言と演算- (1/2)

`x1=[3 5]`

行ベクトル ($1 \times n$)

`x2=[1;3]`

列ベクトル ($n \times 1$)

`x3 = [3 4 5;6 7 8]`

行列

`x4 = [(-4)^2 4^2]`

配列の中での演算

`x5 = 5:8`

等間隔ベクトル

`x6 = 20:2:26`

各要素の間隔がわかっているとき

`x7 = linspace(0,1,5)`

ベクトル内の要素の数がわかっているとき

`x8 = x7'`

転置

`x9 = rand(2)`

コマンド操作 -D. 配列の宣言と演算- (2/2)

```
x10 = zeros(6,3)
```

```
x11=x10(2,1)
```

配列のインデックス指定

```
x12 =x10(end,2)
```

```
x13 = x10(2,:)
```

その次元のすべての要素を指定

```
x14 = x10(2,1:3)
```

```
x14 = x10(2,1:3)
```

```
x15 = [3;4] * [10 20]
```

行列乗算

```
x16=[3 4] .* [10 20]
```

要素単位の乗算

```
[xrow,xcol] = size(x14)
```

```
x11(2)=20
```

コマンド操作 -E. データのプロット-

例：正弦波関数のプロット

```
% x : 0 と2πの間の線形に等間隔な値のベクトルとして作成  
% 値の間はπ/100刻み  
% yはxを引数とした正弦波関数  
% データのラインプロットを作成
```

```
x = 0:pi/100:2*pi;  
y = sin(x);
```

```
figure(1)  
plot(x,y)  
title('sin curve')  
xlabel('x')  
ylabel('sin(x)')
```

```
figure(2)  
plot(x,y,'r--o')  
title('red circle & Dashed line')  
xlabel('x')  
ylabel('sin(x)')
```

コマンド操作 -F. データの保存と読み込み-

```
>> x=rand(10,10)
```

```
>> save datafile x
```

```
>> clear
```

```
>> load datafile
```

MATLABエディタ

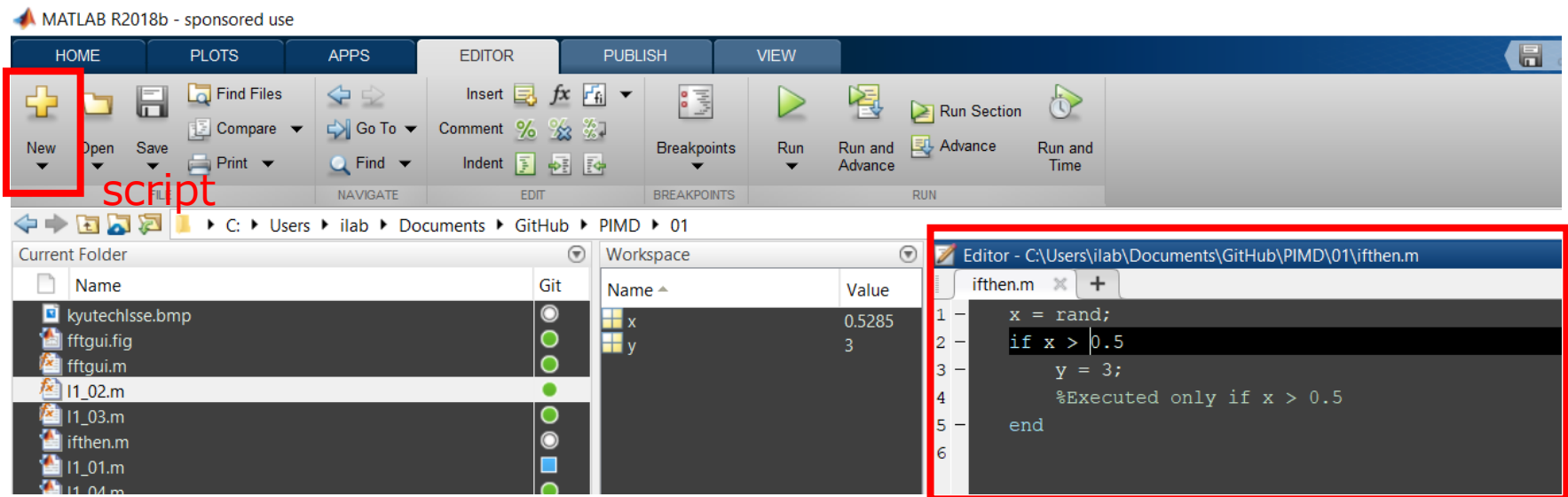
A. スクリプトの書き方

B. 関数化

C. ODEソルバー

MATLABエディター-A. スクリプトの書き方- (1/2)

How to write MATAALB script



Editor

```
dx = 0.1;
x = 0:dx:2*pi;
y = sin(x);
plot(x,y);
ylabel('sine');
xlabel('t');
```

-MATLABの基礎-

MATLABエディター-A. スクリプトの書き方- (2/2)

If-else

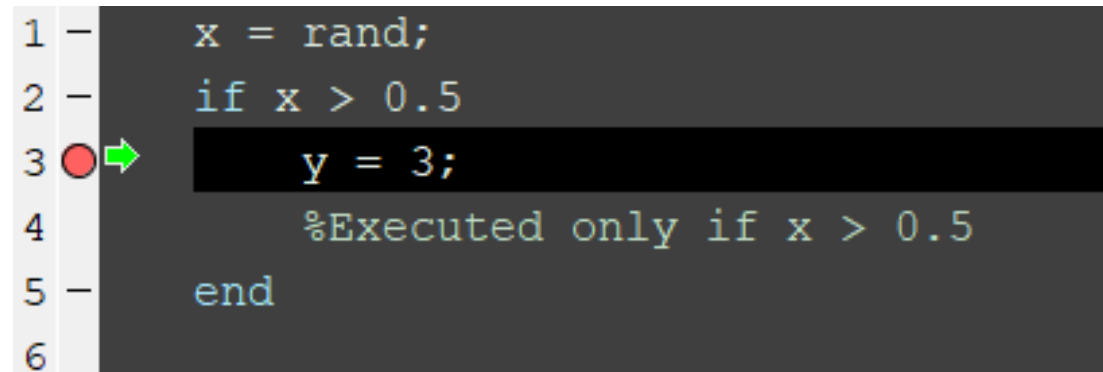
```
x = rand;  
if x > 0.5  
    y = 3; %Executed only if x >  
0.5  
End
```

```
x = rand;  
if x > 0.5  
    y = 3;  
else  
    y = 4;  
end
```

For loops

```
for i = 1:3  
    disp(i)  
end
```

breakpoint



MATLABエディタ -B. 関数化-

関数Mファイル

- 基本的にはMファイルと同じ.
- スクリプトの先頭にfunctionキーワードをつける.

Function [return vector] = function_name(argument list)

平均値計算関数(avefunc.m)

```
function y = l1_02(x)
if ~isvector(x)
    error('Input must be a vector')
end
y = sum(x)/length(x);
end
```

```
>>avefunc([3 7 8])
```


MATLABエディタ -C. ODEソルバー-

例：振り子の解析

常微分方程式 (Ordinary Differential Equation : ODE)

$$ml\ddot{\theta} + kl\dot{\theta} + mg\sin\theta = 0$$

pendulum.m

```
function dy = pendulum (t,y,m,l,k)
```

```
% 振り子の振動関数
```

```
% y(1)=θ
```

```
% y(2)=d
```

```
% 重力加速度9.8
```

```
% 入力引数
```

```
% m:質量
```

```
% l:ロッドの長さ
```

```
% k:粘性減衰
```

```
g=9.8;
```

```
dy=[y(2);-l/(m*l).*(k.*l.*y(2) + m.*g.*sin(y(1)))];
```

```
>>[T Y]=ode45(@pendulum,t, y,[], m, l, k)
```

```
>>plot(T, Y)
```

$$\begin{aligned} \theta = y_1 &\Rightarrow \dot{y}_1 = y_2 \\ \dot{\theta} = y_2 &\Rightarrow \dot{y}_2 = \frac{-(kly_2 + mgsin y_1)}{ml} \end{aligned}$$

初期値

$y(1) = 1, y(2) = 0$

ロッドの長さ

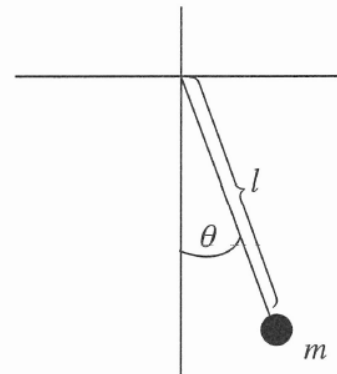
1.5

シミュレーション時間

0 ~ 30 秒

粘性減衰

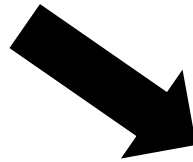
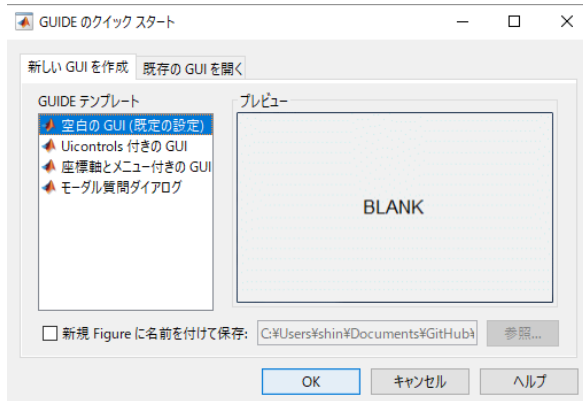
0.2



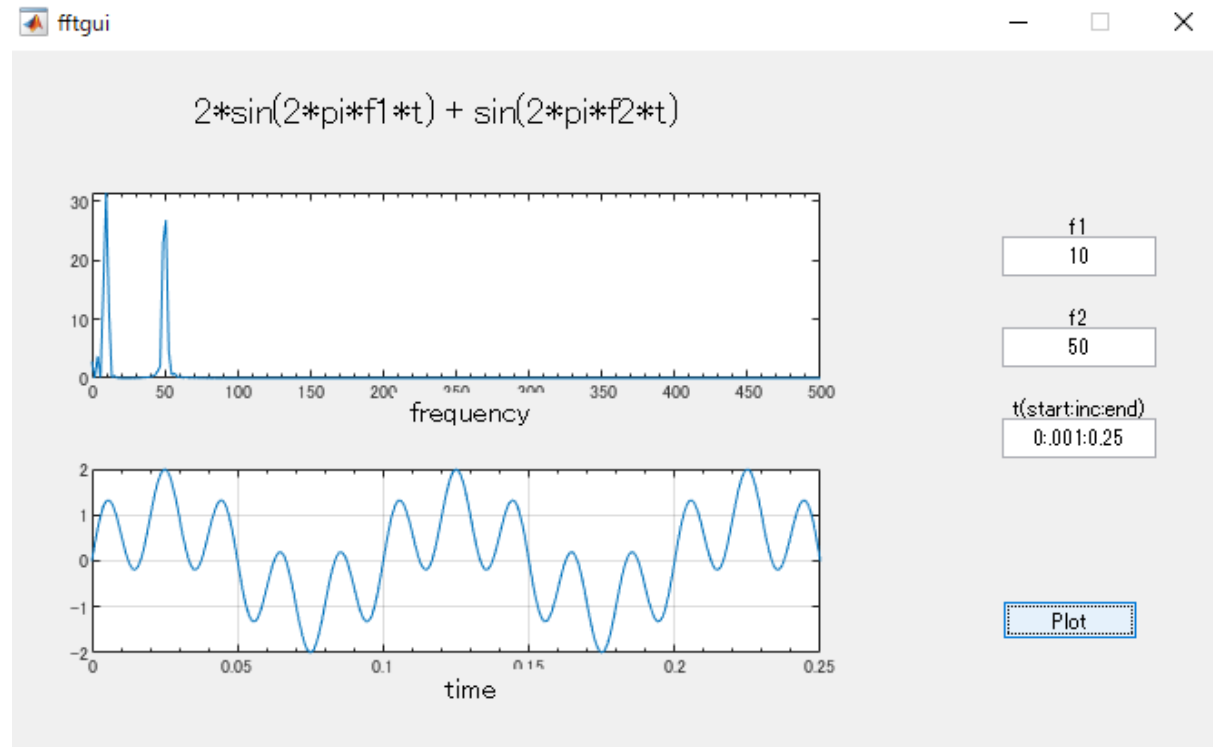
変数名	意味	単位
m	錘の質量	kg
l	ロッドの長さ	m
θ	角度	rad
g	重力加速度	m/s ²
k	粘性減衰	kg/(m/s)

GUIプログラミングの基礎 (1/5)

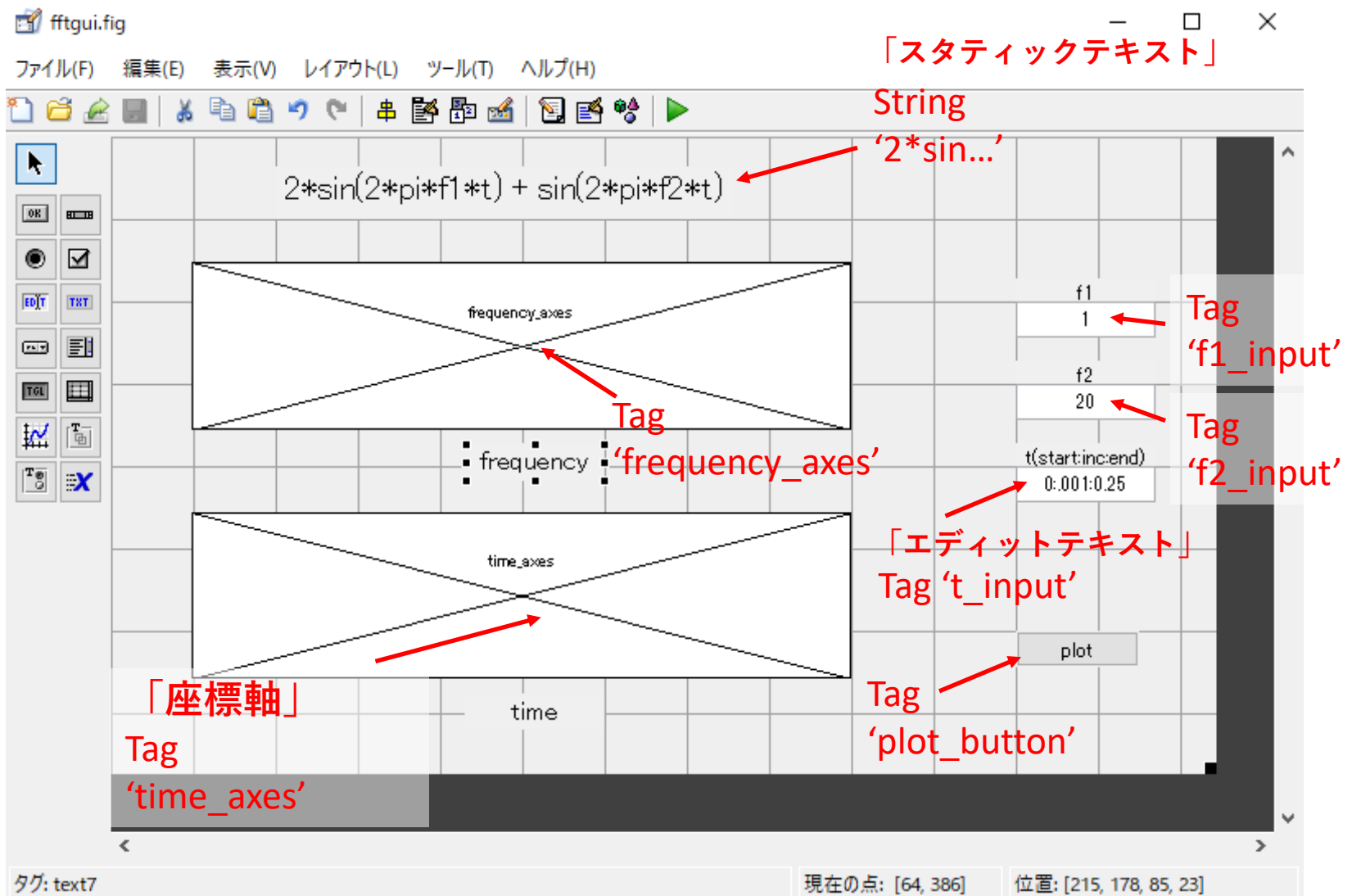
>> guide



2つの正弦波の和を示す関数について
周波数および時間領域表現で表示するアプリ



GUIプログラミングの基礎 (2/5)



→保存すると、紐づけられているmファイルも更新

- GUIプログラミング-

GUIプログラミングの基礎 (3/5)

f1_input_Callback と f2_input_Callback

- 関数 f1_input_Callback
[f1] 編集フィールドで値を変更すると実行
- 関数 f2_input_Callback
[f2] フィールドの変更に応答し、関数 f1_input_Callback とほぼ同じ

どちらの関数も、有効なユーザー入力をチェックする。
編集フィールドの値が無効な場合、[Plot] ボタンは無効

```
f1 = str2double(get(hObject,'String'));  
if isnan(f1) || ~isreal(f1)  
    % Disable the Plot button and change its string to  
    say why  
    set(handles.plot_button,'String','Cannot plot f1');  
    set(handles.plot_button,'Enable','off');  
    % Give the edit text box focus so user can correct  
    the error  
    uicontrol(hObject);  
else  
    % Enable the Plot button with its original name  
    set(handles.plot_button,'String','Plot');  
    set(handles.plot_button,'Enable','on');  
end
```

GUIプログラミングの基礎 (4/5)

t_input_Callback

• [t] 編集フィールドの値を変更すると実行

1. 値が数値であること、
2. 値の長さが 2 ~ 1000 であること、
3. ベクトルが単調増加していることを確認する.

catch ブロックで、[Plot] ボタンのラベルを変更し、入力値が無効であることを示す

• uicontrol コマンド

エラーのある値を含むフィールドにフォーカスを設定する

```
try
    t = eval(get(handles.t_input,'String'));
    if ~isnumeric(t)
        % t is not a number
        set(handles.plot_button,'String','t is not numeric')
    elseif length(t) < 2
        % t is not a vector
        set(handles.plot_button,'String','t must be vector')
    elseif length(t) > 1000
        % t is too long a vector to plot clearly
        set(handles.plot_button,'String','t is too long')
    elseif min(diff(t)) < 0
        % t is not monotonically increasing
        set(handles.plot_button,'String','t must increase')
    else
        % Enable the Plot button with its original name
        set(handles.plot_button,'String','Plot')
        set(handles.plot_button,'Enable','on')
        return
    end
catch EM
    % Cannot evaluate expression user typed
    set(handles.plot_button,'String','Cannot plot t');
    uicontrol(hObject);
end
```

GUIプログラミングの基礎 (5/5)

plot_button_Callback

関数 plot_button_Callback
[Plot] ボタンをクリックすると実行

- ①コールバックは 3 つの編集フィールドで値を取得
- ②次に、コールバックは f1、f2 および t の値を使用して、時間領域で関数をサンプリングし、フーリエ変換を計算する
- ③その後、2つのプロットが更新される

```
f1 = str2double(get(handles.f1_input,'String'));  
f2 = str2double(get(handles.f2_input,'String'));  
t = eval(get(handles.t_input,'String'));
```

% Calculate data

```
x = sin(2*pi*f1*t) + sin(2*pi*f2*t);  
y = fft(x,512);  
m = y.*conj(y)/512;  
f = 1000*(0:256)/512;
```

% Create frequency plot in proper axes

```
plot(handles.frequency_axes,f,m(1:257));  
set(handles.frequency_axes,'XMinorTick','on');  
grid on
```

% Create time plot in proper axes

```
plot(handles.time_axes,t,x);  
set(handles.time_axes,'XMinorTick','on');  
grid on
```

Reporting assignment 1

- ①DialogBox上の
- ②「load」buttonを押すと, imageをload
- ③「compute」buttonを押すと, そのinput imageが二値化 (binarize)される

GUI programを作成せよ.



Image file

URL:

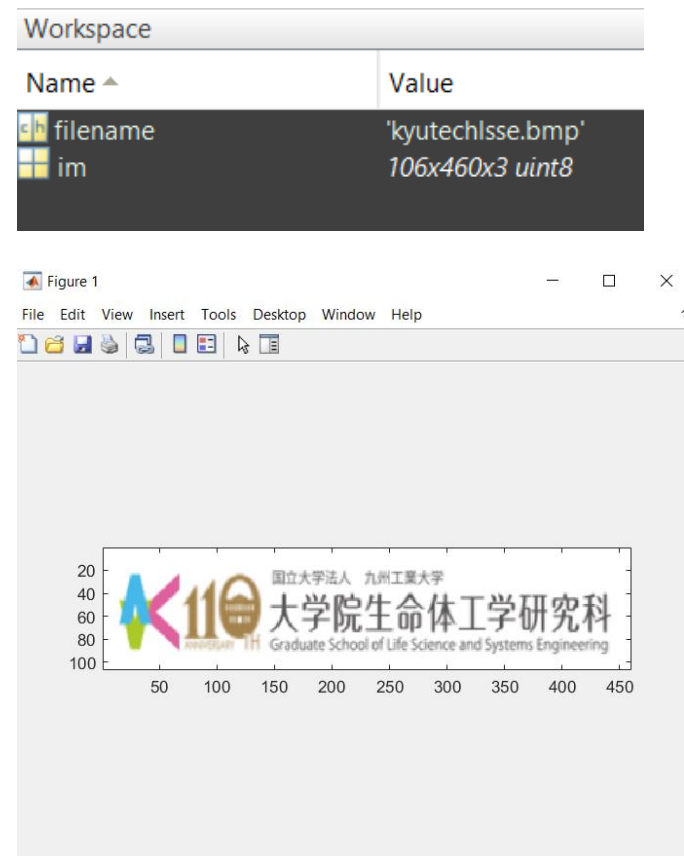


<https://github.com/syasukawa/PIMDst/tree/master/01>

Exercise 1 画像の読み込みと表示

imageread.m

```
% get filename
filename = uigetfile('*.bmp');
% read image
im=imread(filename);
% plot image
image(im);
% set xy ratio
set(gca,'dataAspectRatio',[1 1 1]);
```



Reporting assignment 1:hint (plot関数も参考に)

image(ax,___) creates the image in the axes specified by ax instead of in the current axes (gca). The option ax can precede any of the input argument combinations in the previous syntaxes.

Exercise 2 画像処理 (二値化)

imageproc.m

```
% get filename
filename = uigetfile('*.bmp');
% read image
im=imread(filename);
% compute image
imgray = rgb2gray(im);
BW = imbinarize(imgray);
% plot images
image(BW*255);
% set xy ratio
set(gca,'dataAspectRatio',[1 1 1]);
```

