

MATLAB PDE Solving

Boundary Value Problems

Computational Design Laboratory
Department of Automotive Engineering
Hanyang University, Seoul, Korea



CONTENTS

- **Review ODE Solving**
- **Boundary Value Problems for ODE**
- **Boundary Value Problems for PDE**
- **Case Study**
- **Assignment**
- **Appendix : MATLAB GUI**

EXAMPLE 27.10

van der Pol equation

$$\frac{d^2y_1}{dt^2} - \mu(1 - y_1^2) \frac{dy_1}{dt} + y_1 = 0$$

initial condition

$$t = 0, y_1 = 1, \frac{dy_1}{dt} = 1$$

convert process

$$\begin{cases} \frac{dy_1}{dt} = y_2 \\ \frac{dy_2}{dt} = \mu(1 - y_1^2)y_2 + y_1 \end{cases}$$



stiff 한 정도가 μ 값에 따라 변하는 van der Pol equation.

In 1920 the Dutch physicist Balthasar van der Pol studied a differential equation that describes the circuit of a vacuum tube.

It has been used to model other phenomenon such as the human heartbeat by Johannes van der Mark.

```

Editor - C:\Users\sean\Desktop\vanderpol.m
File Edit Text Go Cell Tools Debug Desktop
File Edit Text Go Cell Tools Desktop
1 function yp = vanderpol(t,y,mu)
2     yp = [y(2); mu*(1-y(1)^2)*y(2)-y(1)];
3 end

```

vanderpol predprey.m dydt.m example_III.m vanderpol.m

vanderpol Ln 3 Col 4 OVR



매틀랩 함수

EXAMPLE 27.10: MAIN & RESULT

Editor - C:\Users\sean\Desktop\example.m

```

1 - clc; clear all; close all;
2 - %% ode45 solution when mu equals 1
3 - [t_45a,y_45a] = ode45(@vanderpol,[0 20],[1,1],[],1);
4 - %% ode45 solution when mu equals 1000
5 - [t_45b,y_45b] = ode45(@vanderpol,[0 20],[1,1],[],1000);
6 - %% ode23s solution when mu equals 1000
7 - [t_23s,y_23s] = ode23s(@vanderpol,[0 6000],[1,1],[],1000);
8 - %% plot solution
9 - subplot(3,1,1)
10 - plot(t_45a,y_45a)
11 - title('ode45 solution when mu equals 1')
12 - subplot(3,1,2)
13 - plot(t_45b,y_45b(:,1))
14 - title('ode45 solution when mu equals 1000')
15 - subplot(3,1,3)
16 - plot(t_23s,y_23s(:,1))
17 - title('ode23s solution when mu equals 1000')

Heun.m x example II.m x predprev.m x dvdt.m x ex
t_23s 1827x1 double
t_45a 213x1 double
t_45b 67705x1 double
y_23s 1827x2 double
y_45a 213x2 double
y_45b 67705x2 double

```

The figure shows the MATLAB Editor window with the script file 'example.m'. The code defines three solutions: one for mu=1 using ode45, one for mu=1000 using ode45, and one for mu=1000 using ode23s. The results are plotted in three subplots:

- ode45 solution when mu equals 1:** A smooth oscillating curve between -5 and 5.
- ode45 solution when mu equals 1000:** A smooth oscillating curve between -2 and 2.
- ode23s solution when mu equals 1000:** A piecewise constant function with steps at approximately 1000, 2000, 3000, 4000, and 5000.

- 1 mu = 1 일 때 실행 코드
- 2 mu = 1000 일 때 ode45로 푸는 코드
결과는 그래프 2 번째 그림
이며 20초 계산하는데
67705 pnt. 필요
- 3 mu = 1000 일 때 ode23s로 푸는 코드
결과는 그래프 3번째 그림
이며 6000초 계산하는데
1827 pnt. 필요(ode45로 20초 푸는 pnt개수의 약 1/37)
- 따라서 ode45로는 해당 미분방정식을 푸는 것은 매우 비 효율적(또한 결과의 정확도 보장하기 어려움)

CASE STUDY I

Background. Electric circuits where the current is time-variable rather than constant are common. A transient current is established in the right-hand loop of the circuit shown in Fig. 28.11 when the switch is suddenly closed.

Equations that describe the transient behavior of the circuit in Fig. 28.11 are based on Kirchhoff's law, which states that the algebraic sum of the voltage drops around a closed loop is zero (recall Sec. 8.3). Thus,

$$L \frac{di}{dt} + Ri + \frac{q}{C} - E(t) = 0 \quad (28.9)$$

where $L(di/dt)$ = voltage drop across the inductor, L = inductance (H), R = resistance (Ω), q = charge on the capacitor (C), C = capacitance (F), $E(t)$ = time-variable voltage source (V), and

$$i = \frac{dq}{dt} \quad (28.10)$$

RLC circuit equation

$$L \frac{d^2q}{dt^2} + R \frac{dq}{dt} + \frac{q}{C} - E(t) = 0$$

$$\frac{d^2q}{dt^2} = \frac{1}{L} \left(E(t) - R \frac{dq}{dt} - \frac{q}{C} \right)$$

initial condition

$$t = 0, q = 0, i = 0$$

$\xrightarrow{\text{convert process}}$

$$\begin{cases} q = y_1 \\ i = \frac{dq}{dt} = y_2 \\ \frac{di}{dt} = \frac{d^2q}{dt^2} = \frac{1}{L} \left(E(t) - Ry_2 - \frac{y_1}{C} \right) \end{cases}$$

$$E = E_0 \sin(\omega t)$$

$$L = 1 \text{ H}$$

$$E_0 = 1 \text{ V}$$

$$C = 0.25 \text{ F}$$

$$\omega^2 = 3.5 \text{ s}^2$$

$$R = 0$$

CASE STUDY I

Editor - C:\Users\sean\Desktop\circuit.m

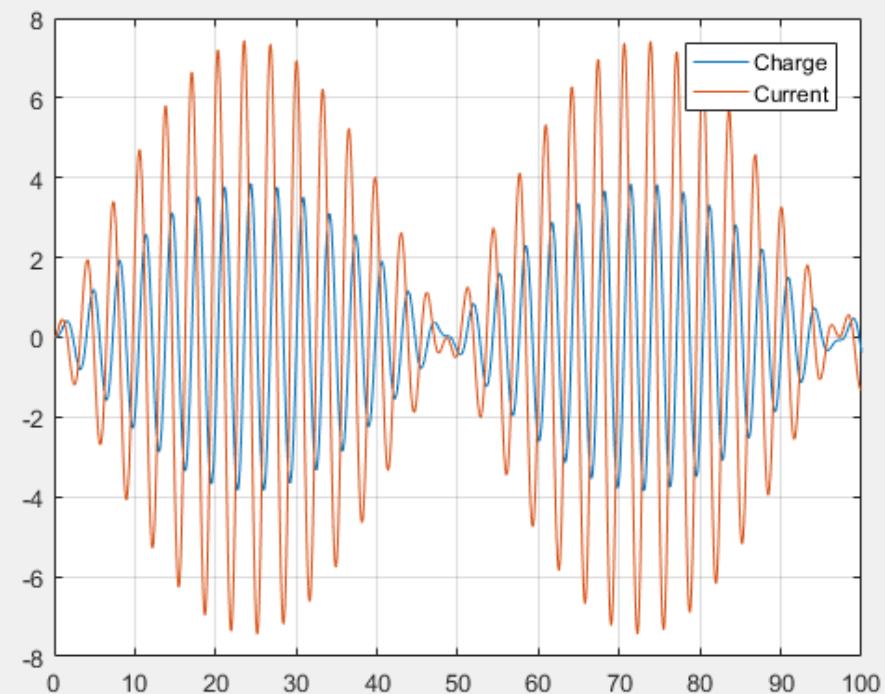
```
File Edit Text Go Cell Tools Debug Desktop Window Help
- 1.0 + ÷ 1.1 × % + % - ⓘ
1 function yp = circuit(t,y,L,E0,omega,R,C)
2     yp = [y(2); 1/L*E0*sin(omega*t)-R*y(2)-y(1)/C];
3 end
case_Im x circuit.m x
Ln 3 Col 4 OVR ...
```

circuit

Editor - C:\Users\sean\Desktop\case_Im.m

```
File Edit Text Go Cell Tools Debug Desktop Window Help
- 1.0 + ÷ 1.1 × % + % - ⓘ
1 clc; clear all; close all;
2 %% ode45 solution
3 [t,y] = ode45(@circuit,[0 100],[0,0],[],1,1,sqrt(3.5),0,0.25);
4 %% plot solution
5 plot(t,y)
case_Im x circuit.m x
script
```

Ln



CASE STUDY II

Therefore, in polar coordinates ($\alpha = d^2\theta/dt^2$),

$$-W \sin \theta = \frac{Wl}{g} \alpha = \frac{Wl}{g} \frac{d^2\theta}{dt^2}$$

or

$$\frac{d^2\theta}{dt^2} + \frac{g}{l} \sin \theta = 0 \quad (28.15)$$

This apparently simple equation is a second-order nonlinear differential equation. In general, such equations are difficult or impossible to solve analytically. You have two choices regarding further progress. First, the differential equation might be reduced to a form that can be solved analytically (recall Sec. PT7.1.1), or second, a numerical approximation technique can be used to solve the differential equation directly. We will examine both of these alternatives in this example.

pendulum equation

$$\frac{d^2\theta}{dt^2} + \frac{g}{l} \sin \theta = 0$$

$$\frac{d^2\theta}{dt^2} = -\frac{g}{l} \sin \theta$$

initial condition

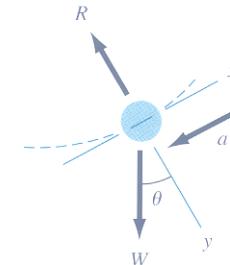
$$t = 0, \theta = 0, \frac{d\theta}{dt} = 0$$

convert process

$$\begin{cases} \theta = y_1 \\ \frac{d\theta}{dt} = y_2 \\ \frac{d^2\theta}{dt^2} = -\frac{g}{l} \sin \theta \end{cases}$$

FIGURE 28.16

A free-body diagram of the swinging pendulum showing the forces on the particle and the acceleration.



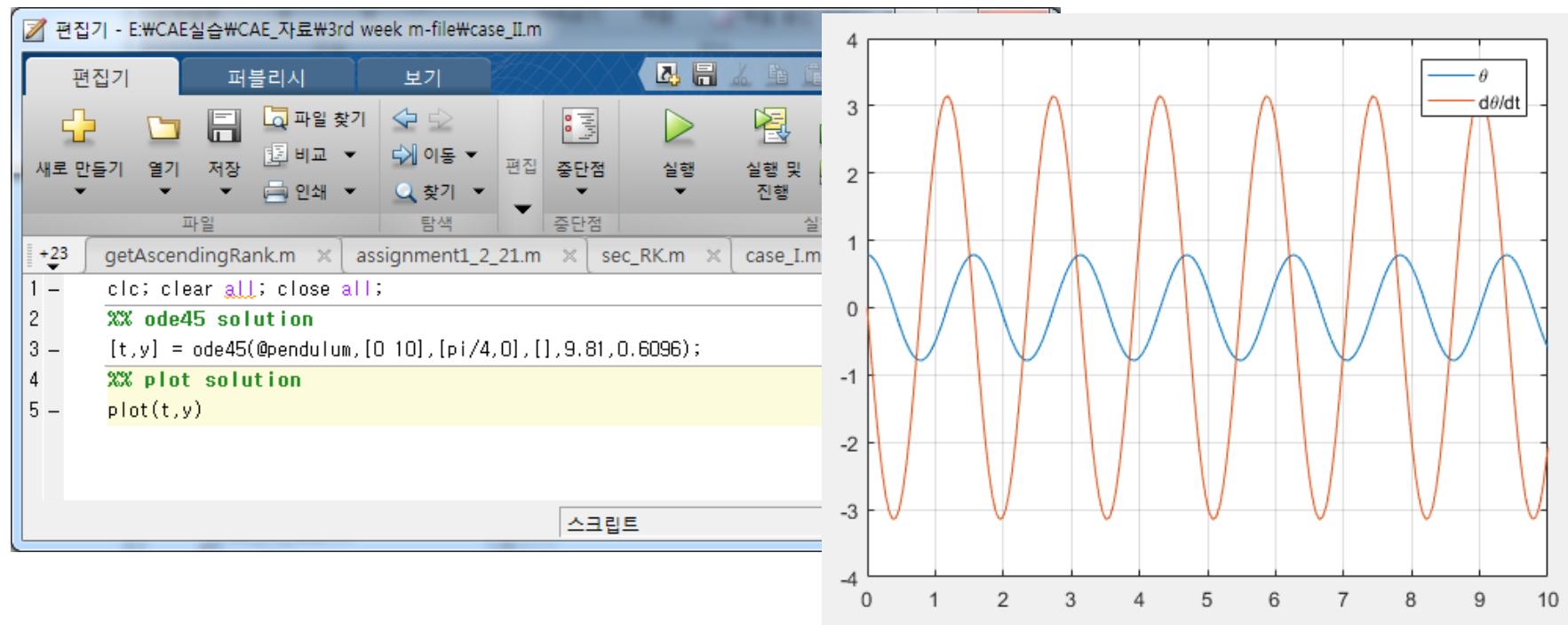
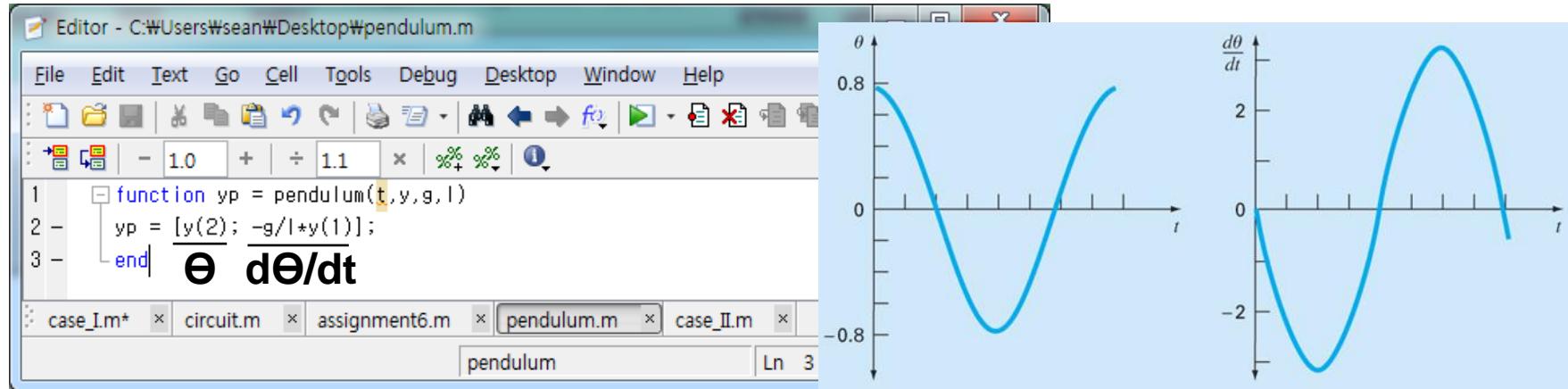
$$\sin \theta \approx \theta$$

$$g = 9.81 \text{ m/s}^2$$

$$\theta_0 = \pi / 4$$

$$l = 0.6096 \text{ m}$$

CASE II: RESULT



- **Boundary Value Problems for ODE**
 - ✓ **Shooting method for linear ODE**
 - ✓ **Shooting method for nonlinear ODE**
 - ✓ **Finite difference method**

EXAMPLE

- Heat balance for a long, thin rod
 - Not insulated along its length
 - Steady state

$$\frac{d^2T}{dx^2} + h'(T_a - T) = 0$$

$$\left. \begin{array}{l} T(0) = T_1 = 40^\circ C \\ T(L) = T_2 = 200^\circ C \end{array} \right\} \text{boundary conditions}$$

$T_a = 20^\circ C$ (temperature of the surrounding air)

$L = 10 m$

$h' = 0.01 m^{-2}$ (heat transfer coefficient)

rate of heat dissipation to the surrounding air

analytic solution: $T = 73.4523e^{0.1x} - 53.4523e^{-0.1x} + 20$

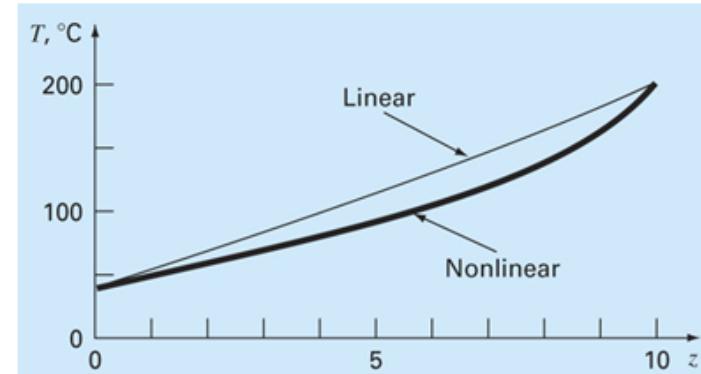
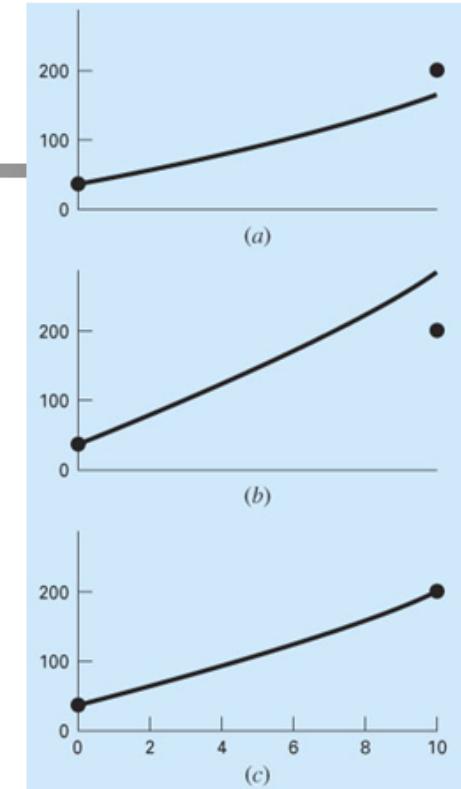
EXAMPLE

Shooting Method

$$\begin{cases} \frac{dT}{dx} = z \\ \frac{dz}{dx} = h'(T - T_a) \text{ [linear]} \quad \frac{dz}{dx} = h''(T - T_a)^4 \text{ [nonlinear]} \end{cases}$$

4th RK, $h = 2, T(10) = 200$

$$\rightarrow \begin{cases} \text{guess / linear: } \begin{cases} z(0) = 10 \rightarrow T(10) = 168.3797 \\ z(0) = 20 \rightarrow T(10) = 285.8980 \end{cases} \\ \xrightarrow{\text{linear interpolation}} z(0) = 12.6907 \\ \text{non-linear: } T(10) = f(z(0)) \\ \rightarrow g(z(0)) = f(z(0)) - 200 \end{cases}$$



SHOOTING METHOD: LINEAR

```

Editor - C:\Users\sean\Desktop\heatfun.m
File Edit Text Go Cell Tools Debug Desktop Window Help
function dy = heatfun(x,y)
dy = [y(2); -0.01*(20-y(1))];
```

The code defines a function `heatfun` that takes `x` and `y` as inputs. It returns a vector `dy` containing two elements: `y(2)` and `-0.01*(20-y(1))`. A red box highlights the second line of the function definition.

```

Editor - C:\Users\sean\Desktop\ex1.m
File Edit Text Go Cell Tools Debug Desktop Window Help
clc; clear all;
[x1,y1] = ode45(@heatfun, [0 10], [40 10]);
Tb1 = y1(length(y1(:,1)));
[x2,y2] = ode45(@heatfun, [0 10], [40 20]);
Tb2 = y2(length(y2(:,1)));
za = 10+(20-10)/(Tb2-Tb1)*(200-Tb1);
[x3,y3] = ode45(@heatfun, [0 10], [40 za]);
plot(x3,y3(:,1))
Tb = y3(length(y3(:,1)))
```

The script starts by clearing the workspace. It then calls the `heatfun` function twice using `ode45` to solve the differential equation over the interval [0, 10]. The first call corresponds to T_{b1} and the second to T_{b2} . The initial conditions for both are $[40, 10]$. The script then calculates the intermediate value z_a using the shooting method formula. Finally, it solves the equation again from T_{b1} to T_b with the calculated z_a as the initial condition, plots the result, and finds the final value T_b .

1 아래에 표기된 시스템 ODE 함수를 `heatfun` 이름으로 저장

$$\begin{aligned} \frac{dT}{dx} &= z \\ \frac{dz}{dx} &= -0.01(20-T) \end{aligned} \quad \left. \begin{array}{l} y(1)=T \\ y(2)=z \end{array} \right\}$$

$$\frac{dy(1)}{dx} = y(2)$$

$$\frac{dy(2)}{dx} = -0.01(20 - y(1))$$

2 z_{a1} 이 10 일 때와 z_{a2} 가 20 일 때 값을 T_{b1} 과 T_{b2} 에 저장

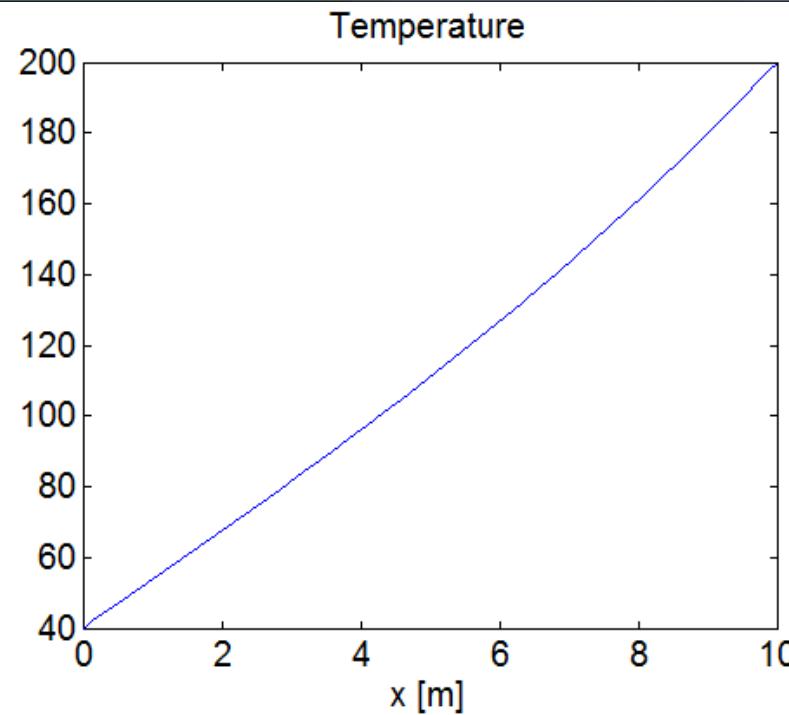
3 아래 수식을 이용하여 z_a 를 선형 보간

$$z_a = z_{a1} + \frac{z_{a2} - z_{a1}}{T_{b2} - T_{b1}} (T_b - T_{b1})$$

RESULT

Command Window

```
File Edit Debug Desktop Window Help
za =
12.6905
Tb =
fx 200.0000
```



1 z_a 가 12.6905 일 때 B.C. 0 일치

2 결과 그래프

SHOOTING METHOD: NONLINEAR

$$\begin{cases} \frac{dT}{dx} = z \\ \frac{dz}{dx} = -h'(T_{\infty} - T) - h''(T_{\infty}^4 - T^4) \end{cases}$$

```

Editor - C:\Users\sean\Desktop\dydxn.m
File Edit Text Go Cell Tools Debug Desktop Window Help
function dy = dydxn(x,y)
dy = [y(2); -0.05*(200-y(1))-2.7e-9*(1.6e9-y(1)^4)];

```

assignment6.m ex.m heatfun.m ex1.m dydxn.m res.m Untitled5

dydxn Ln 2 Col 53 OVR

```

Editor - C:\Users\sean\Desktop\res.m
File Edit Text Go Cell Tools Debug Desktop Window Help
function r=res(za)
[x,y] = ode45(@dydxn, [0 10], [300 za]);
r = y(length(x),1)-400;

```

assignment6.m ex.m heatfun.m ex1.m dydxn.m res.m Untitled5

res Ln 3 Col 24 OVR



아래에 표기된 상수대로 왼쪽의 ODE 함수를 입력

$$h' = 0.05 \text{ m}^{-2}$$

$$h'' = 2.7 \times 10^{-9} \text{ K}^{-3} \text{ m}^{-2}$$

$$T_{\infty} = 200 \text{ K}$$

$$T(0) = 300 \text{ K}$$

$$T_{\infty}(10) = 400 \text{ K}$$



근을 찾기 위한 함수 설정

r 값(residual: 해석해와 수치해의 차이)이 0 일 경우 정확한 z_a 값

SHOOTING METHOD: NONLINEAR

The screenshot shows the MATLAB Editor window with the file name 'ex2_1.m'. The code in the editor is:

```

1 - clc; clear all;
2 - fzero(@res,-50)

```

A red box highlights the second line of code, and a red footprint icon with the number '1' is placed next to it.

`fzero(@funname, IV)`

funname의 근을 찾는 매틀
랩 함수
IV 는 initial value 를 뜻함

[주의] 비선형함수이므로 초기값에 따른 영향 존재

The screenshot shows the MATLAB Command Window. The command entered was:

```

ans =
-41.7434

```

A red box highlights the output value '-41.7434', and an orange footprint icon with the number '2' is placed next to it.

결과 $z_a = -41.7434$

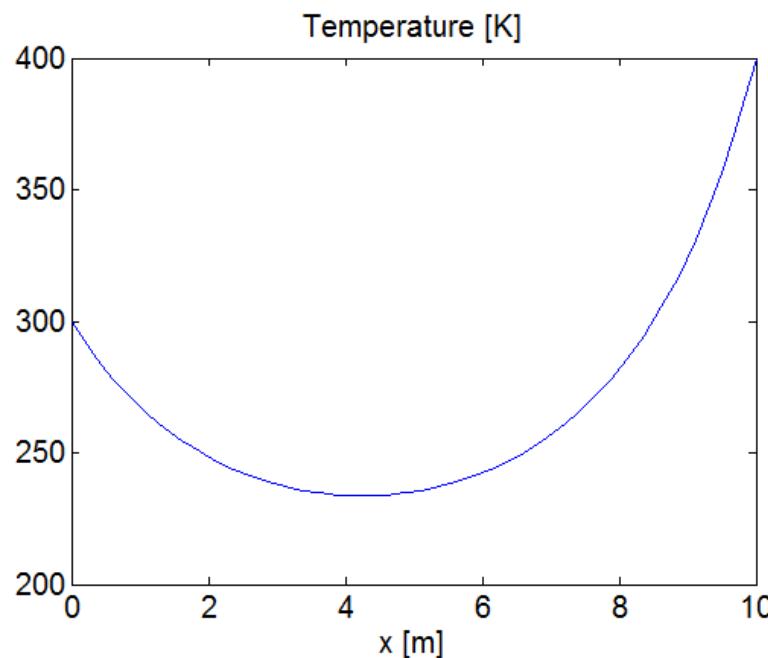
즉, 이 값을 이용하여 ode 함수를 풀면 nonlinear BC를 만족하는 해

RESULT

Editor - C:\Users\sean\Desktop\ex2_2.m

```
File Edit Text Go Cell Tools Debug Desktop Window Help
- 1.0 + ÷ 1.1 × %% %% 1
1 - clc; clear all;
2 - [x,y] = ode45(@dydxn,[0 10],[300 fzero(@res,-50)]);
3 - plot(x,y(:,1))
```

脚印图标 1 指向代码中的 `fzero(@res,-50)` 部分，脚印图标 2 指向结果图形。



1 혹은 initial 값을 입력하는
곳에 fzero 함수를 직접 입
력 가능

2 결과 그래프

FINITE DIFFERENCE METHOD

Forward FDM	$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h}$	$O(h)$
	$f'(x_i) = \frac{-f(x_{i+2}) + 4f(x_{i+1}) - 3f(x_i)}{2h}$	$O(h^2)$
	$f''(x_i) = \frac{f(x_{i+2}) - 2f(x_{i+1}) + f(x_i)}{h^2}$	$O(h)$
	$f''(x_i) = \frac{-f(x_{i+3}) + 4f(x_{i+2}) - 5f(x_{i+1}) + 2f(x_i)}{h^2}$	$O(h^2)$
Backward FDM	$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{h}$	$O(h)$
	$f'(x_i) = \frac{3f(x_i) - 4f(x_{i-1}) + f(x_{i-2})}{2h}$	$O(h^2)$
	$f''(x_i) = \frac{f(x_i) - 2f(x_{i-1}) + f(x_{i-2})}{h^2}$	$O(h)$
	$f''(x_i) = \frac{2f(x_i) - 5f(x_{i-1}) + 4f(x_{i-2}) - f(x_{i-3})}{h^2}$	$O(h^2)$
Centered FDM	$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h}$	$O(h^2)$
	$f'(x_i) = \frac{-f(x_{i+2}) + 8f(x_{i+1}) - 8f(x_{i-1}) + f(x_{i-2})}{12h}$	$O(h^4)$
	$f''(x_i) = \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{h^2}$	$O(h^2)$
	$f''(x_i) = \frac{-f(x_{i+2}) + 16f(x_{i+1}) - 30f(x_i) + 16f(x_{i-1}) - f(x_{i-2})}{12h^2}$	$O(h^4)$

FINITE DIFFERENCE METHOD

$$\frac{d^2T}{dx^2} = -h'(T_\infty - T) \rightarrow \frac{T_{i-1} - 2T_i + T_{i+1}}{\Delta x^2} = -h'(T_\infty - T_i)$$

$$\rightarrow -T_{i-1} + (2 + h' \Delta x^2) T_i - T_{i+1} = h' \Delta x^2 T_\infty$$

```

Editor - C:\Users\sean\Desktop\FDM_ex1.m
File Edit Text Go Cell Tools Debug Desktop Window Help
File Edit Text Go Cell Tools Debug Desktop Window Help
1 - clc; clear all
2
3 - step_size = 2; h = 0.05; T_inf = 200;
4 - length = 10; T_a = 300; T_b = 400;
5
6 - number = length/step_size;
7
8 - K = zeros((number+1),(number+1));
9 - f = ones((number-1),1)*h*step_size^2*T_inf;
10
11 - f(1,1) = f(1,1) + T_a;
12 - f((number-1),1) = f((number-1),1) + T_b;
13
14 - i = 1:3;
15 - a = [-1, 2+h*step_size^2,-1];
16
17 - for j = 1:(number+1)
18 -     K(j,i+j-1) = a;
19 - end
20
21 - K = K(1:(number-1),2:number);
22
23 - T_sol = K\f;
24 - T_total = [T_a;T_sol;T_b]
25 - plot([0:step_size:length],T_total)

```

The code implements the finite difference method for a 1D heat transfer problem. It starts by initializing variables: step_size (2), h (0.05), T_inf (200), length (10), T_a (300), and T_b (400). It calculates the number of nodes (51) and initializes the coefficient matrix K and right-hand side vector f. Boundary conditions are applied at the first and last nodes. A loop constructs the tridiagonal matrix K, which is then solved for the temperature distribution T. Finally, the total temperature profile is plotted.

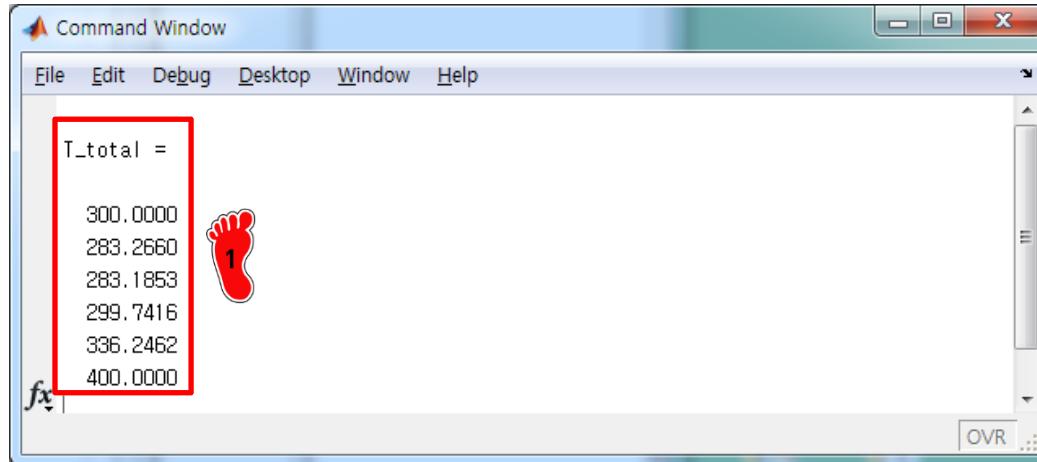


RESULT

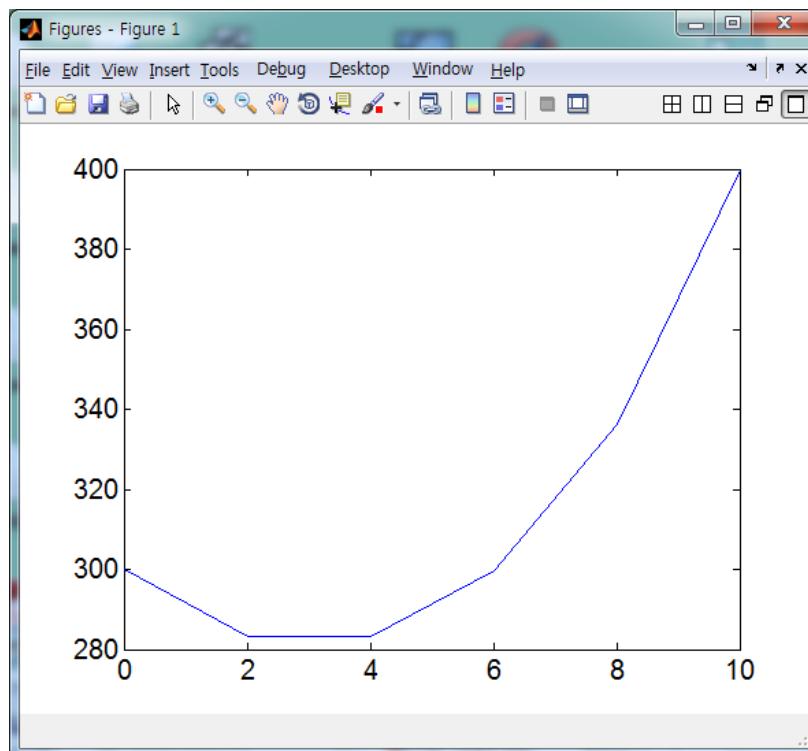
Command Window

```
File Edit Debug Desktop Window Help
T_total =
300.0000
283.2660
283.1853
299.7416
336.2462
400.0000
fx|
```

1



- 1 결과 온도 벡터
2 결과 그래프
- 



- **Boundary Value Problems for PDE**
 - ✓ **Finite Difference Method: Elliptic equation**

LAPLACE EQUATION

- PDE → algebraic difference equation

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0 \leftarrow \begin{cases} \frac{\partial^2 T}{\partial x^2} = \frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{\Delta x^2} \\ \frac{\partial^2 T}{\partial y^2} = \frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{\Delta y^2} \end{cases}$$

$$\rightarrow \frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{\Delta x^2} + \frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{\Delta y^2} = 0$$

$$\xrightarrow{\Delta x = \Delta y} T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1} - 4T_{i,j} = 0$$

apply boundary conditions (fixed/Dirichlet)

$$@ (1,1) : T_{21} + \underbrace{T_{01}}_{75^\circ C} + T_{12} + \underbrace{T_{10}}_{0^\circ C} - 4T_{11} = 0 \rightarrow 4T_{11} - T_{21} - T_{12} = 75$$

$$@ (2,1) : T_{31} + T_{11} + T_{22} + \underbrace{T_{20}}_{0^\circ C} - 4T_{21} = 0 \rightarrow -T_{11} + 4T_{21} - T_{31} - T_{22} = 0$$

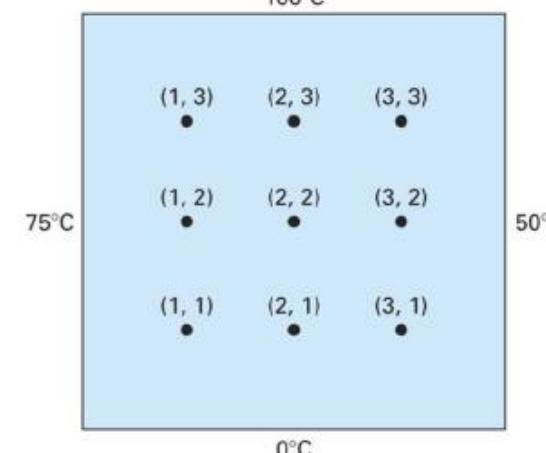
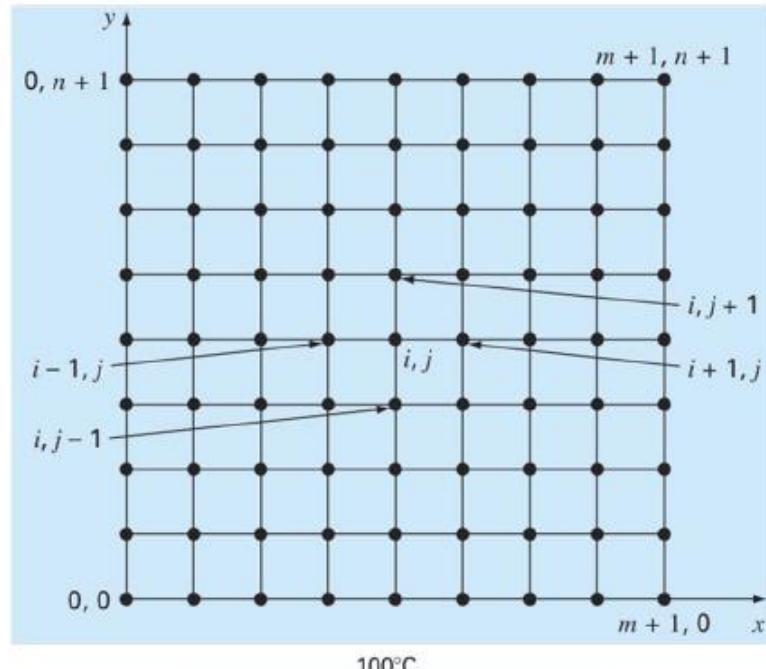
$$@ (3,1) : \underbrace{T_{41}}_{50^\circ C} + T_{21} + T_{32} + \underbrace{T_{30}}_{0^\circ C} - 4T_{31} = 0 \rightarrow -T_{21} + 4T_{31} - T_{32} = 50$$

⋮
9 equations

$$[K_{ij}] \{T_i\} = \{f_i\}$$

CAE

$[K_{ij}]$: coefficient matrix, $\{T_i\}$: solution vector, $\{f_i\}$: force vector



DE - 11

MATLAB CODE

```

1 - clc; clear all;
2 - T_left = 75; T_right = 50; T_bottom = 0; T_upper = 100;
3 - nx = 3; ny = 3;
4 - T_numbering = [];
5 - iter = 0;
6 - i = [1:nx]';
7 - j = ones(nx,1);
8 - for k = 1:ny
9 -     T_numbering = [T_numbering;i,j*k];
10 - end
11 - f = zeros(nx*ny,1);
12 - for j = 1:ny
13 -     for i = 1:nx
14 -         T_numbering(:,3) = zeros(nx*ny,1);
15 -         a = find(T_numbering(:,1) == i+1 & T_numbering(:,2) == j);
16 -         b = find(T_numbering(:,1) == i-1 & T_numbering(:,2) == j);
17 -         c = find(T_numbering(:,1) == i & T_numbering(:,2) == j+1);
18 -         d = find(T_numbering(:,1) == i & T_numbering(:,2) == j-1);
19 -         e = find(T_numbering(:,1) == i & T_numbering(:,2) == j);
20 -         T_numbering(a,3) = -1;
21 -         T_numbering(b,3) = -1;
22 -         T_numbering(c,3) = -1;
23 -         T_numbering(d,3) = -1;
24 -         T_numbering(e,3) = 4;
25 -         iter = iter + 1;
26 -         I(iter,:) = T_numbering(:,3)';
27 -         a = [a 1];
28 -         b = [b 1];
29 -         c = [c 1];
30 -         d = [d 1];
31 -         if length(a) == 1
32 -             f(iter) = f(iter) + T_right;
33 -         end
34 -         if length(b) == 1
35 -             f(iter) = f(iter) + T_left;
36 -         end
37 -         if length(c) == 1
38 -             f(iter) = f(iter) + T_upper;
39 -         end
40 -         if length(d) == 1
41 -             f(iter) = f(iter) + T_bottom;
42 -         end
43 -     end
44 - end
45 - Temp_temp1 = T#f;
46 - iter = 0;
47 - for i = ny:-1:1
48 -     for j = 1:nx
49 -         iter = iter + 1;
50 -         Temp_temp2(i,j) = Temp_temp1(iter,1);
51 -     end
52 - end
53 - Temp = zeros(ny+2,nx+2);
54 - Temp(:,1) = T_left;
55 - Temp(:,nx+2) = T_right;
56 - Temp(1,:) = T_upper;
57 - Temp(ny+2,:) = T_bottom;
58 - Temp(2:ny+1,2:nx+1) = Temp_temp2;
59 - x=[0:1:nx+1]; y=[ny+1:-1:0];
60 - surf(x,y,Temp)
61 - xlabel('x'); ylabel('y'); zlabel('Temp'); colorbar

```

INITIALIZATION

```
clc; clear all;
```

```
T_left = 75; T_right = 50; T_bottom = 0; T_upper = 100;
```

```
nx = 3; ny = 3;
```



```
T_numbering = [];
```

```
iter = 0;
```

```
i = [1:nx]';
j = ones(nx,1);
```

```
for k = 1:ny
    T_numbering = [T_numbering;i,j*k];
end
```

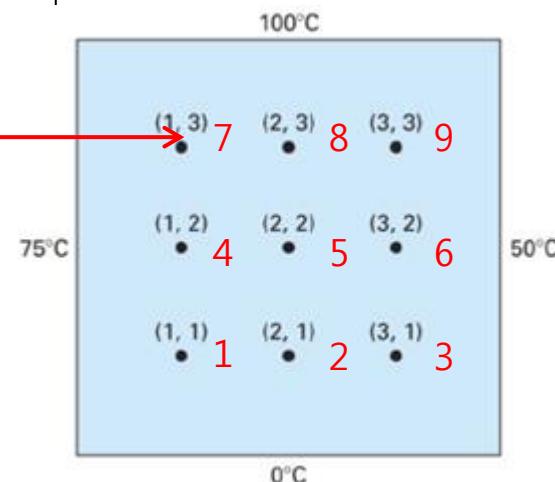
```
f = zeros(nx*ny,1);
```



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

numbering

- 1 경계조건 선언
- 2 경계를 제외한 node 개수 선언
- 3 매트릭스를 구성하기 위한 정보를 저장



COEFFICIENT MATRIX

```
for j = 1:ny
    for i = 1:nx
```

```
T_numbering(:,3) = zeros(nx*ny,1);
a = find(T_numbering(:,1) == i+1 & T_numbering(:,2) == j);
b = find(T_numbering(:,1) == i-1 & T_numbering(:,2) == j);
c = find(T_numbering(:,1) == i & T_numbering(:,2) == j+1);
d = find(T_numbering(:,1) == i & T_numbering(:,2) == j-1);
e = find(T_numbering(:,1) == i & T_numbering(:,2) == j);
T_numbering(a,3) = -1;
T_numbering(b,3) = -1;
T_numbering(c,3) = -1;
T_numbering(d,3) = -1;
T_numbering(e,3) = 4;
iter = iter + 1;
```

$T(\text{iter},:) = \text{T_numbering}(:,3)'$; 

 매트릭스 생성

 nx = 3, ny = 3 일 때의 매트릭스 요소 정보



T <9x9 double>									9
1	2	3	4	5	6	7	8	9	0
1	4	-1	0	-1	0	0	0	0	0
2	-1	4	-1	0	-1	0	0	0	0
3	0	-1	4	0	0	-1	0	0	0
4	-1	0	0	4	-1	0	-1	0	0
5	0	-1	0	-1	4	-1	0	-1	0
6	0	0	-1	0	-1	4	0	0	-1
7	0	0	0	-1	0	0	4	-1	0
8	0	0	0	0	-1	0	-1	4	-1
9	0	0	0	0	0	-1	0	-1	4

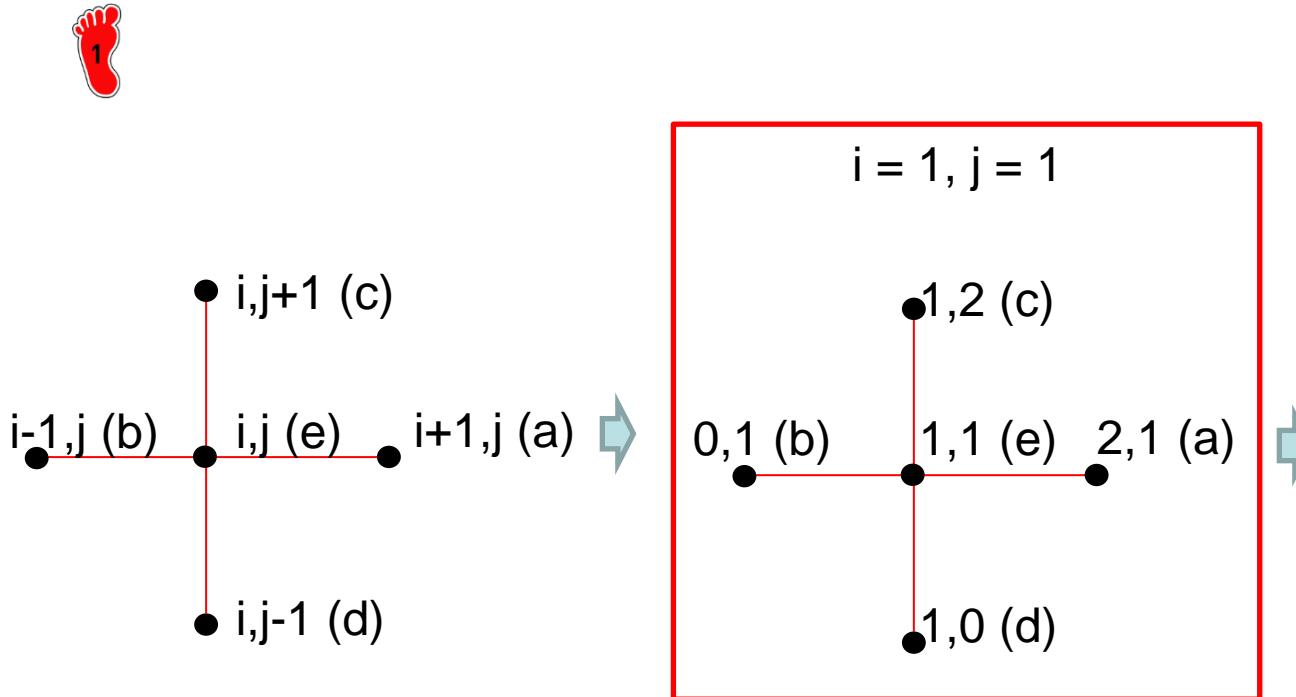
COEFFICIENT MATRIX

```
a = find(T_numbering(:,1) == i+1 & T_numbering(:,2) == j);
b = find(T_numbering(:,1) == i-1 & T_numbering(:,2) == j);
c = find(T_numbering(:,1) == i & T_numbering(:,2) == j+1);
d = find(T_numbering(:,1) == i & T_numbering(:,2) == j-1);
e = find(T_numbering(:,1) == i & T_numbering(:,2) == j);
```



find 는 벡터 혹은 매트릭스
내 요소 중 같은 값을 갖는
요소의 위치정보를 저장하
는 명령어

$i = 1, j = 1$ 일 때
 $T_{numbering}$ 벡터의 위치
정보를 저장



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

$a=1$
 $b=[]$
 $c=4$
 $d=[]$
 $e=1$

COEFFICIENT MATRIX

```
T_numbering(a,3) = -1;
T_numbering(b,3) = -1;
T_numbering(c,3) = -1;
T_numbering(d,3) = -1;
T_numbering(e,3) = 4;
```



	1	2	3	4
1	1	1	1	4
2	2	1	1	-1
3	3	1	1	0
4	1	2	2	-1
5	2	2	2	0
6	3	2	2	0
7	1	3	3	0
8	2	3	3	0
9	3	3	3	0



위치 정보를 이용하여 계수
값을 T_numbering 내에 입
력



입력된 3번째 벡터를
coefficient matrix 의 첫 번
째 열로 저장

$T(\text{iter},:) = T_numbering(:,3)'$;



1	2	3	4	5	6	7	8	9
1	4	-1	0	-1	0	0	0	0

FORCE VECTOR

```
a = [a 1];
b = [b 1];
c = [c 1];
d = [d 1];
```

\rightarrow

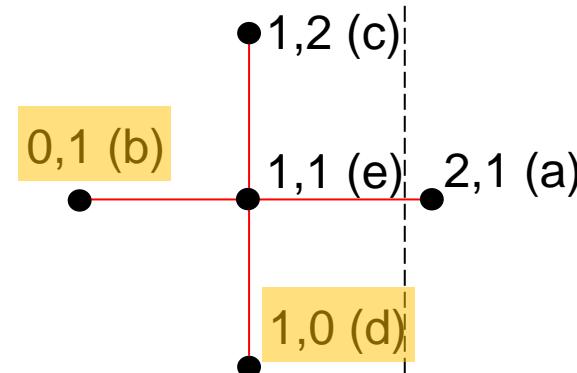
```
a = [2,1]
b = 1
c = [4,1]
d = 1
```



경계조건을 입력할 번호를
구분, length 명령어를 이용
하여 값이 1 일 경우 경계에
존재하는 번호

a 가 경계일 경우 항상 오른
쪽 경계
마찬가지 논리로 입력할 경
계 위치를 결정

```
if length(a) == 1
    f(iter) = f(iter) + T_right;
end
if length(b) == 1
    f(iter) = f(iter) + T_left;
end
if length(c) == 1
    f(iter) = f(iter) + T_upper;
end
if length(d) == 1
    f(iter) = f(iter) + T_bottom;
end
end
end
```



경계조건이 입력될 부분

POST-PROCESSING

```
Temp_temp1 = T\f;
```



```
iter = 0;
for i = ny:-1:1
    for j = 1:nx
        iter = iter + 1;
        Temp_temp2(i,j) = Temp_temp1(iter,1);
    end
end
```



```
Temp = zeros(ny+2,nx+2);
Temp(:,1) = T_left;
Temp(:,nx+2) = T_right;
Temp(1,:) = T_upper;
Temp(ny+2,:) = T_bottom;
Temp(2:ny+1,2:nx+1) = Temp_temp2;
x=[0:1:nx+1]; y=[ny+1:-1:0];
surf(x,y,Temp)
xlabel('x'); ylabel('y'); zlabel('Temp'); colorbar
```

Temp

5x double

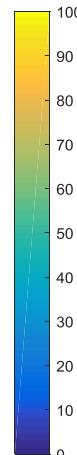
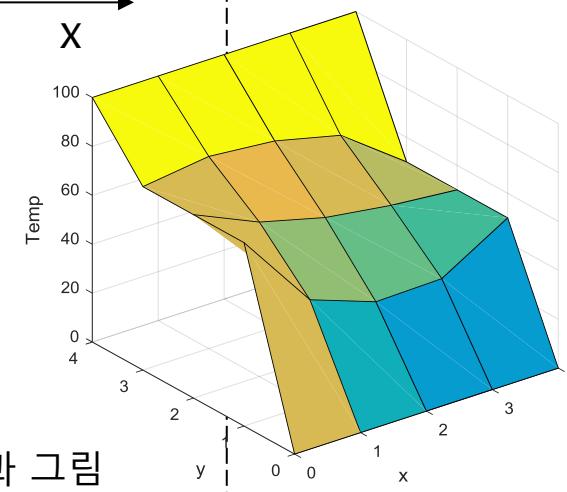
	1	2	3	4	5
1	100	100	100	100	100
2	75	78.5714	76.1161	69.6429	50
3	75	63.1696	56.2500	52.4554	50
4	75	42.8571	33.2589	33.9286	50
5	0	0	0	0	0

y
0

0



결과 그림



1 solve

2 솔루션 매트릭스 재배열

재배열을 하는 것은 온도분
포가 1열 벡터로 출력되기
때문

surf 명령어로 후처리를 할
예정이므로 매트릭스로 재
배열이 필요

3 후처리

ASSIGNMENT

918

CHAP. 21 Numerics for ODEs and PDEs

EXAMPLE 1**Mixed Boundary Value Problem for a Poisson Equation**

Solve the mixed boundary value problem for the Poisson equation

$$\nabla^2 u = u_{xx} + u_{yy} = f(x, y) = 12xy$$

shown in Fig. 457a.

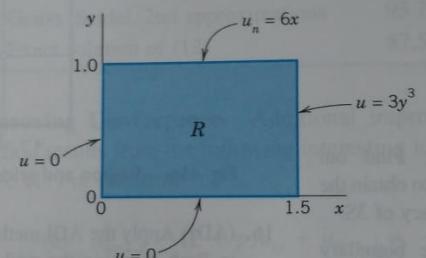
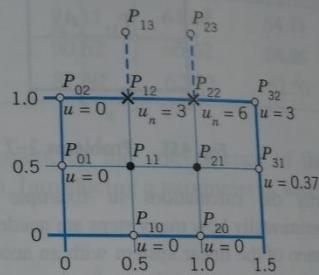
(a) Region R and boundary values(b) Grid ($h = 0.5$)

Fig. 457. Mixed boundary value problem in Example 1

(3)

$$\begin{bmatrix} -4 & 1 & 1 & 0 \\ 1 & -4 & 0 & 1 \\ 2 & 0 & -4 & 1 \\ 0 & 2 & 1 & -4 \end{bmatrix} \begin{bmatrix} u_{11} \\ u_{21} \\ u_{12} \\ u_{22} \end{bmatrix} = \begin{bmatrix} 0.75 \\ 1.125 \\ 1.5 - 3 \\ 0 - 6 \end{bmatrix} = \begin{bmatrix} 0.75 \\ 1.125 \\ -1.5 \\ -6 \end{bmatrix}.$$

(The entries 2 come from u_{13} and u_{23} , and so do -3 and -6 on the right). The solution of (3) (obtained by Gauss elimination) is as follows; the exact values of the problem are given in parentheses.

$$\begin{aligned} u_{12} &= 0.866 \quad (\text{exact } 1) & u_{22} &= 1.812 \quad (\text{exact } 2) \\ u_{11} &= 0.077 \quad (\text{exact } 0.125) & u_{21} &= 0.191 \quad (\text{exact } 0.25). \end{aligned}$$

$$u_{xx} + u_{yy} = f(x, y) = 12xy$$

Dirichet B.C

$$u(0, y) = u(y, 0) = 0, u(1.5, y) = 3y^3$$

Neumann B.C

$$u_y(x, 1) = 6x$$

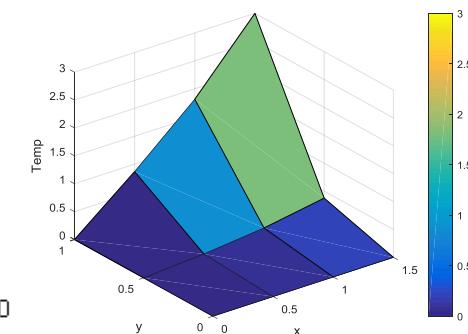
$$1) h = 0.5$$

$$u_{12} = \quad u_{22} =$$

$$0.8665 \quad 1.8121$$

$$u_{11} = \quad u_{21} =$$

$$0.0769 \quad 0.1910$$



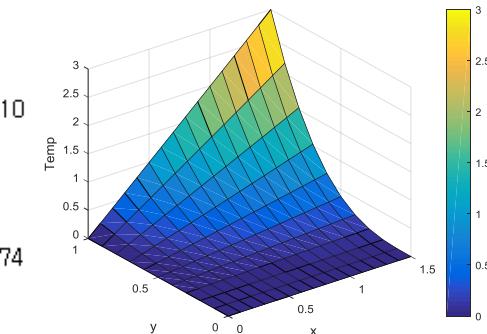
$$2) h = 0.1$$

$$u_{12} = \quad u_{22} =$$

$$0.9941 \quad 1.9910$$

$$u_{11} = \quad u_{21} =$$

$$0.1229 \quad 0.2474$$



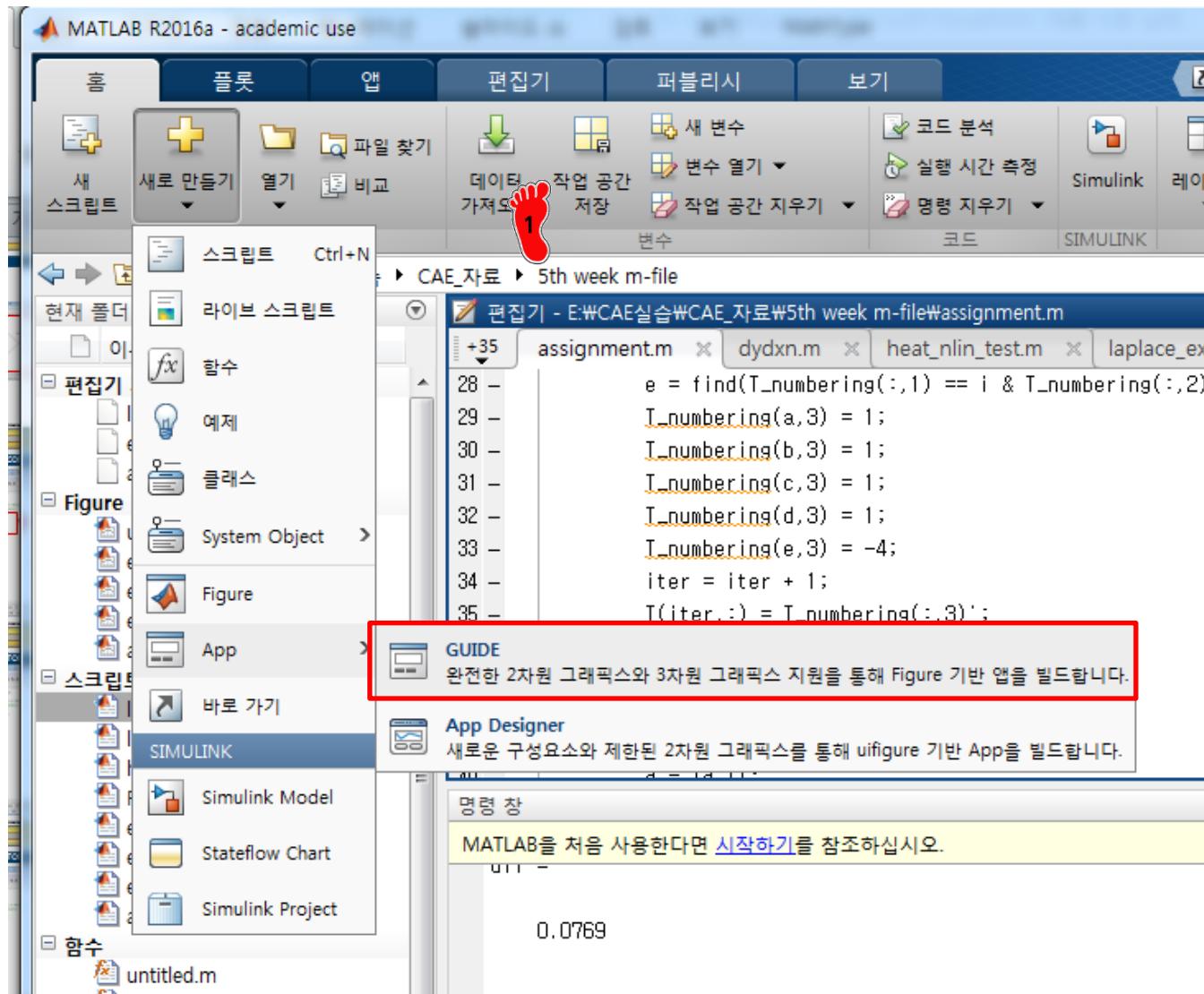
APPENDIX

- **MATLAB GUI implementation**
 - ✓ GUI
 - ✓ **Example**

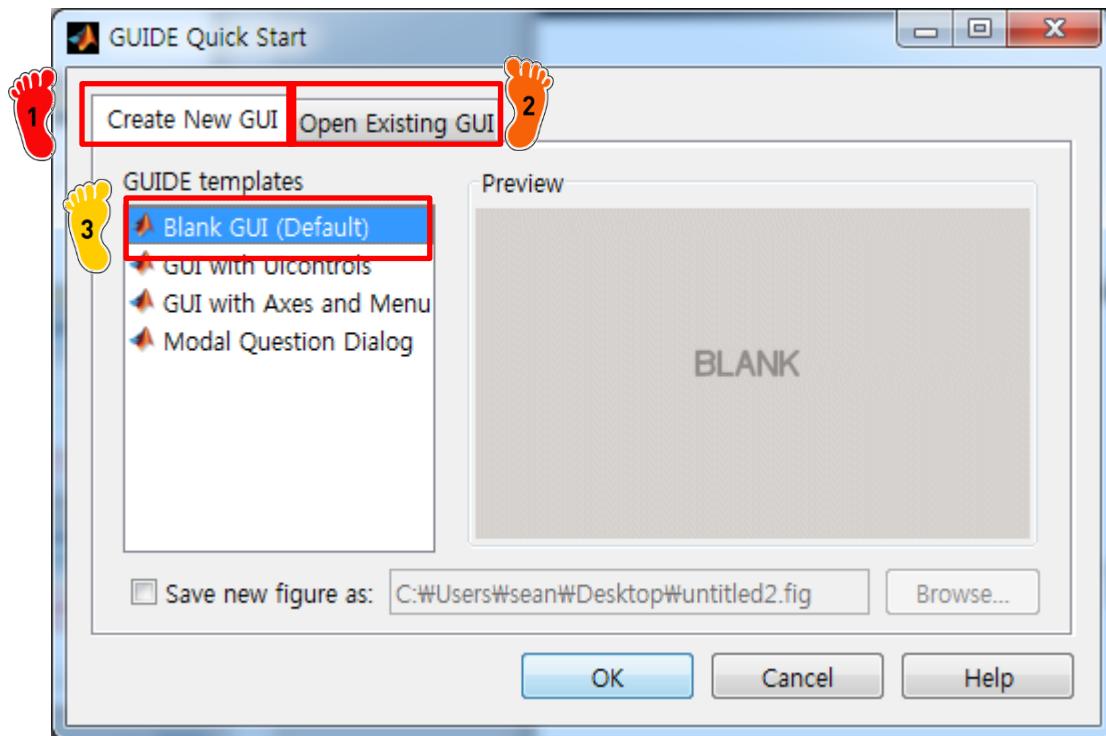
START



home – New – APP –
GUIDE 클릭



GUI SETTING WINDOW



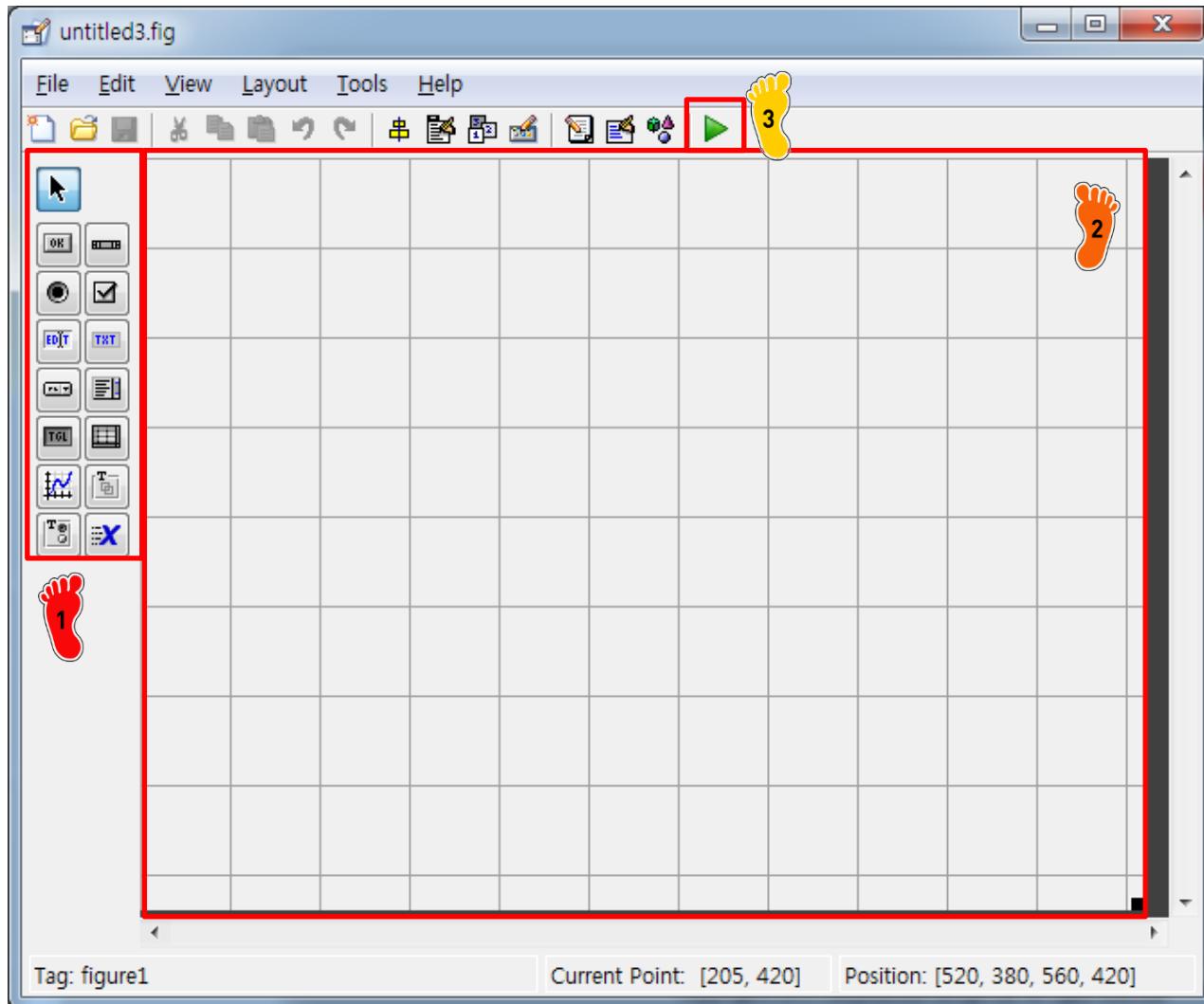
1 새로운 GUI 를 만드는 텁
메뉴

2 기존 GUI 파일을 불러오는
텝 메뉴

3 빈 GUI 창 생성(기본)

그 아래 메뉴는 예제

GUI WINDOW

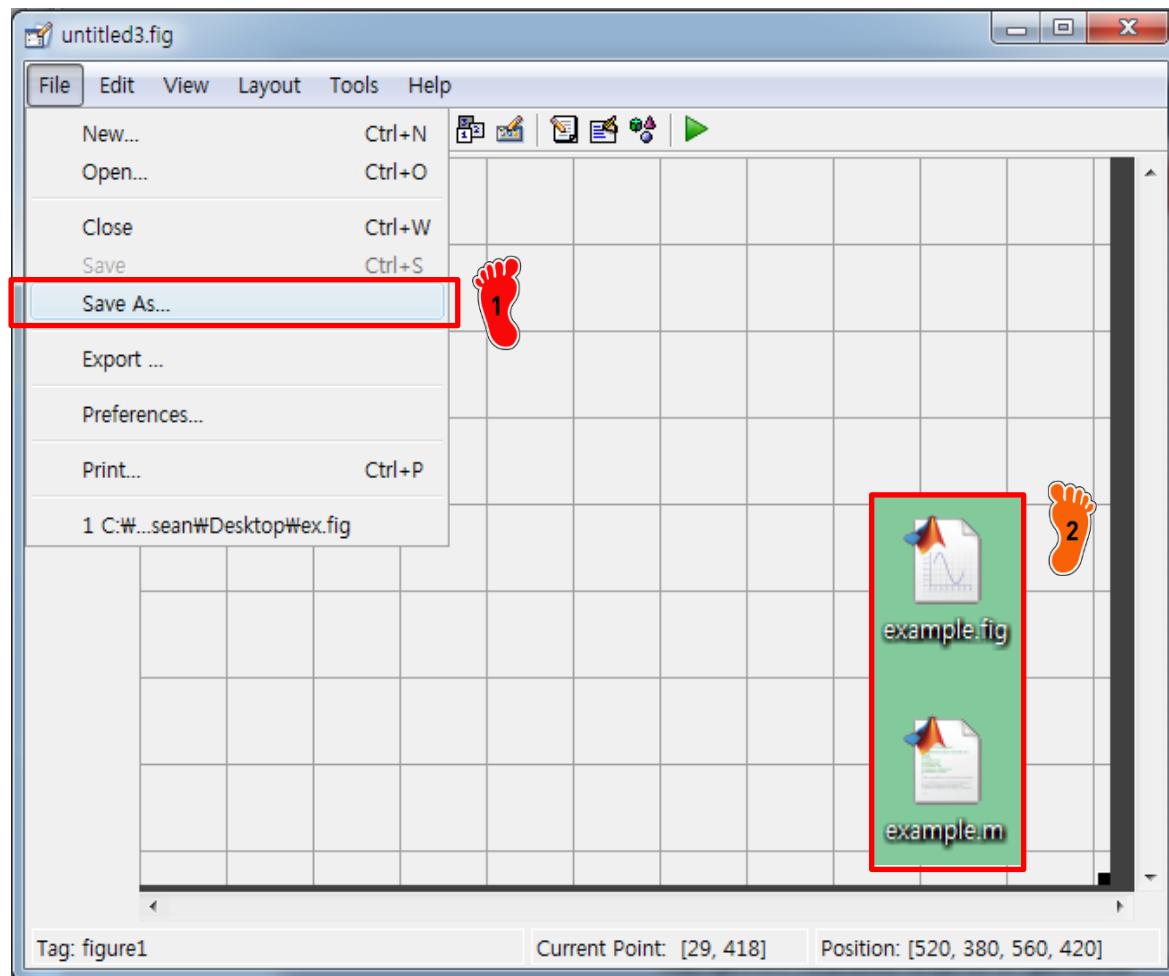


1 GUI 아이콘

2 GUI 아이콘을 배치하는 창

3 GUI 실행 아이콘

GUI FILES



- 1 현재 GUI 창을 저장
- 2 저장하면 *.fig 파일과 *.m 파일이 동시에 생성됨

GUI M-FILE

The screenshot shows a MATLAB interface. At the top, there's a window titled "example" which is currently empty. Below it is the MATLAB editor window with the file name "example.m". The code in the editor is:

```

1 function varargout = example(varargin)
2 % EXAMPLE M-file for example.fig
3 %
4 % EXAMPLE, by itself, creates a new EXAMPLE or raises the existing
5 % singleton*.
6 %
7 % H = EXAMPLE returns the handle to a new EXAMPLE or the handle to
8 % the existing singleton*.
9 %
10 % EXAMPLE('CALLBACK',hObject,eventData,handles,...) calls the local
11 % function named CALLBACK in EXAMPLE.M with the given input arguments.
12 %
13 % EXAMPLE('Property','Value',...) creates a new EXAMPLE or raises the
14 % existing singleton*. Starting from the left, property value pairs are
15 % applied to the GUI before example_OpeningFcn gets called. An
16 % unrecognized property name or invalid value makes property application
% stop. All inputs are passed to example_OpeningFcn via varargin.

```



1 GUI 를 실행하면 설정한 아
이콘이 없기 때문에 빈 창이
pop up 됨



2 m-file 내용은 GUI 를 이용
하는 방법 설명과 현재 GUI
창을 pop up 시키는 명령어
로 이루어짐

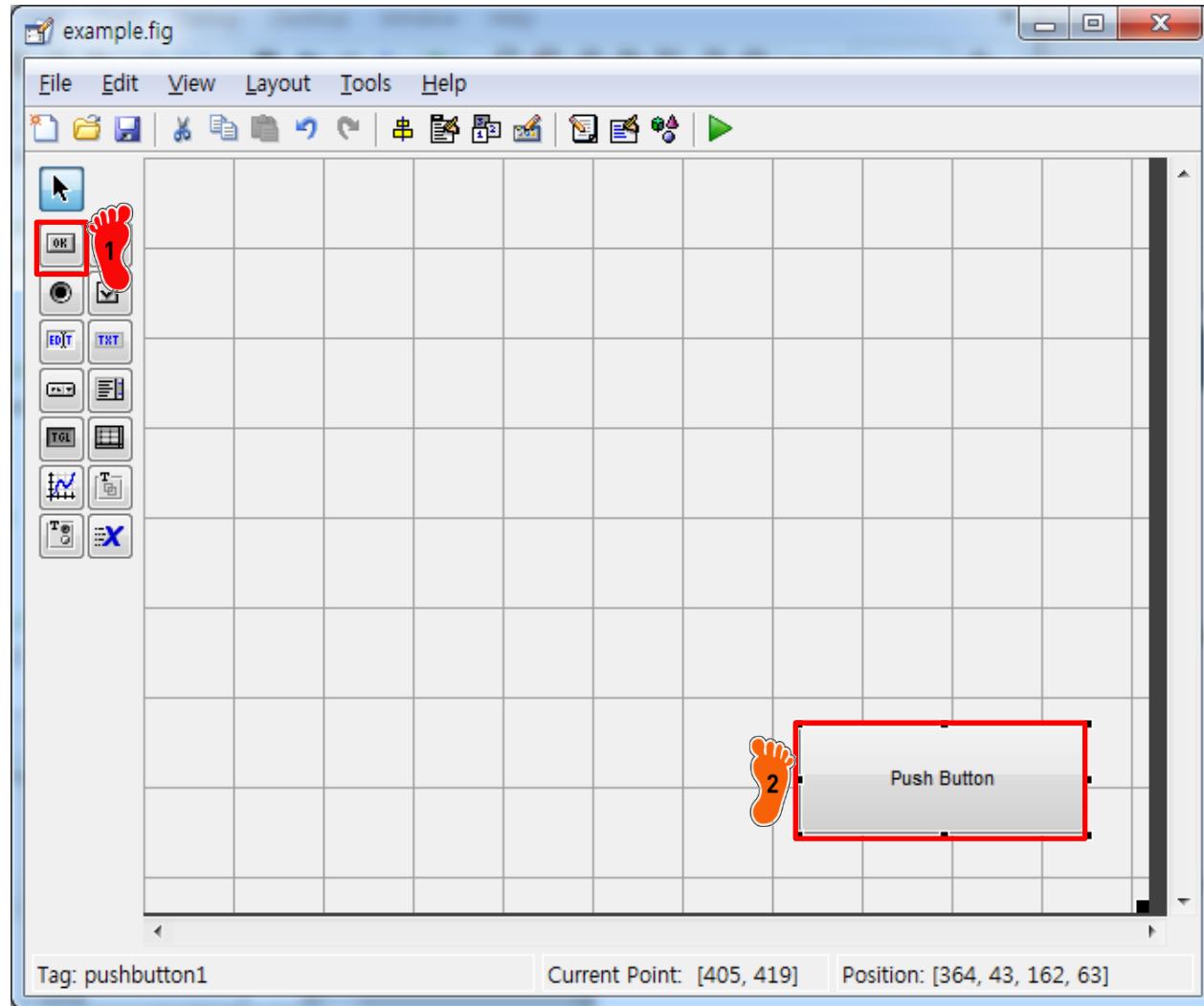
APPENDIX

- **MATLAB GUI implementation**

- ✓ **GUI**
- ✓ **Example (Prob.27-27에 참고)**
- ✓ **ODE 문제를 입력하여 다양한 방법으로 수치해를 구하고 비교할 수 있는 GUI 구축**

고 비교할 수 있는 GUI 구축

PUSH BUTTON

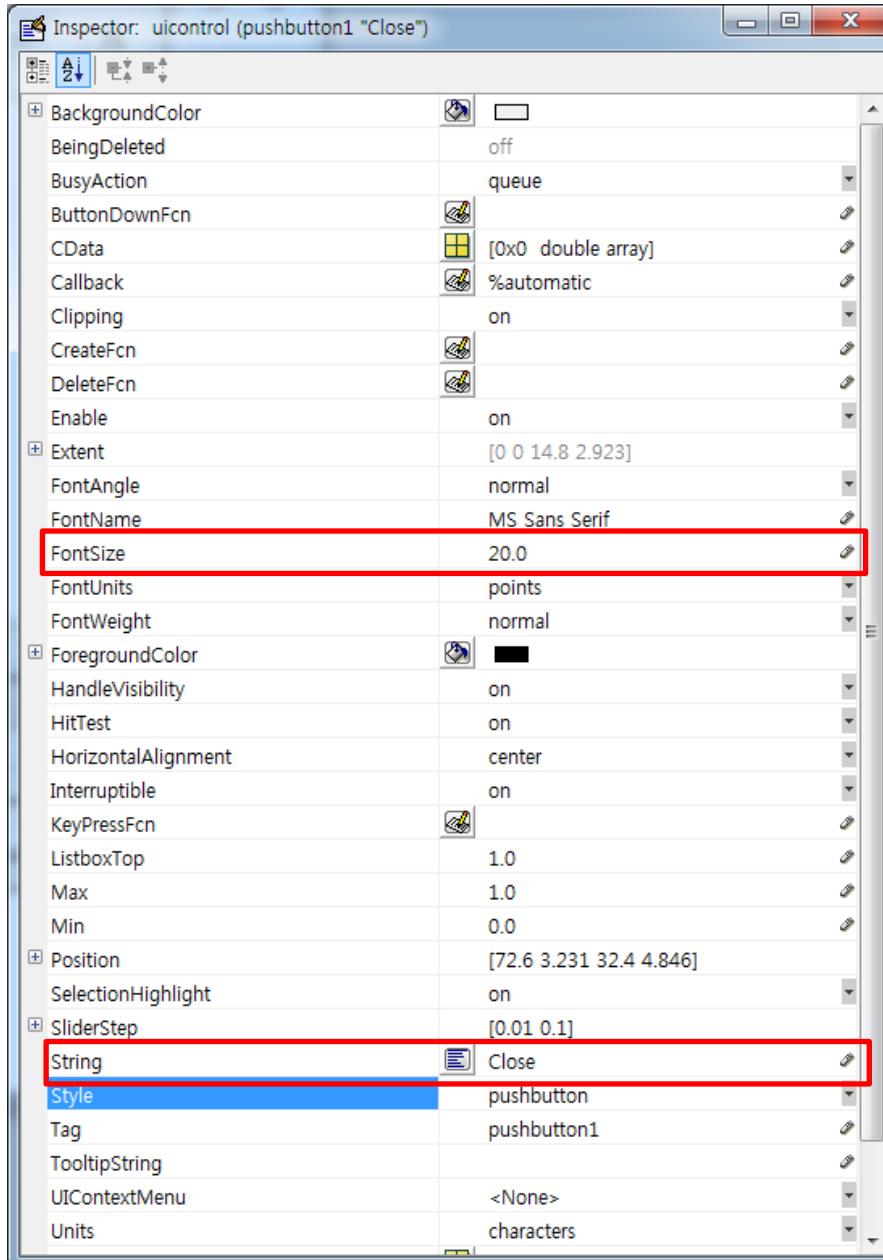


1 Push Button 아이콘 클릭

2 마우스를 드래그해서 Push
Button 아이콘 생성

그 후 아이콘 더블 클릭

PUSH BUTTON: INSPECTOR

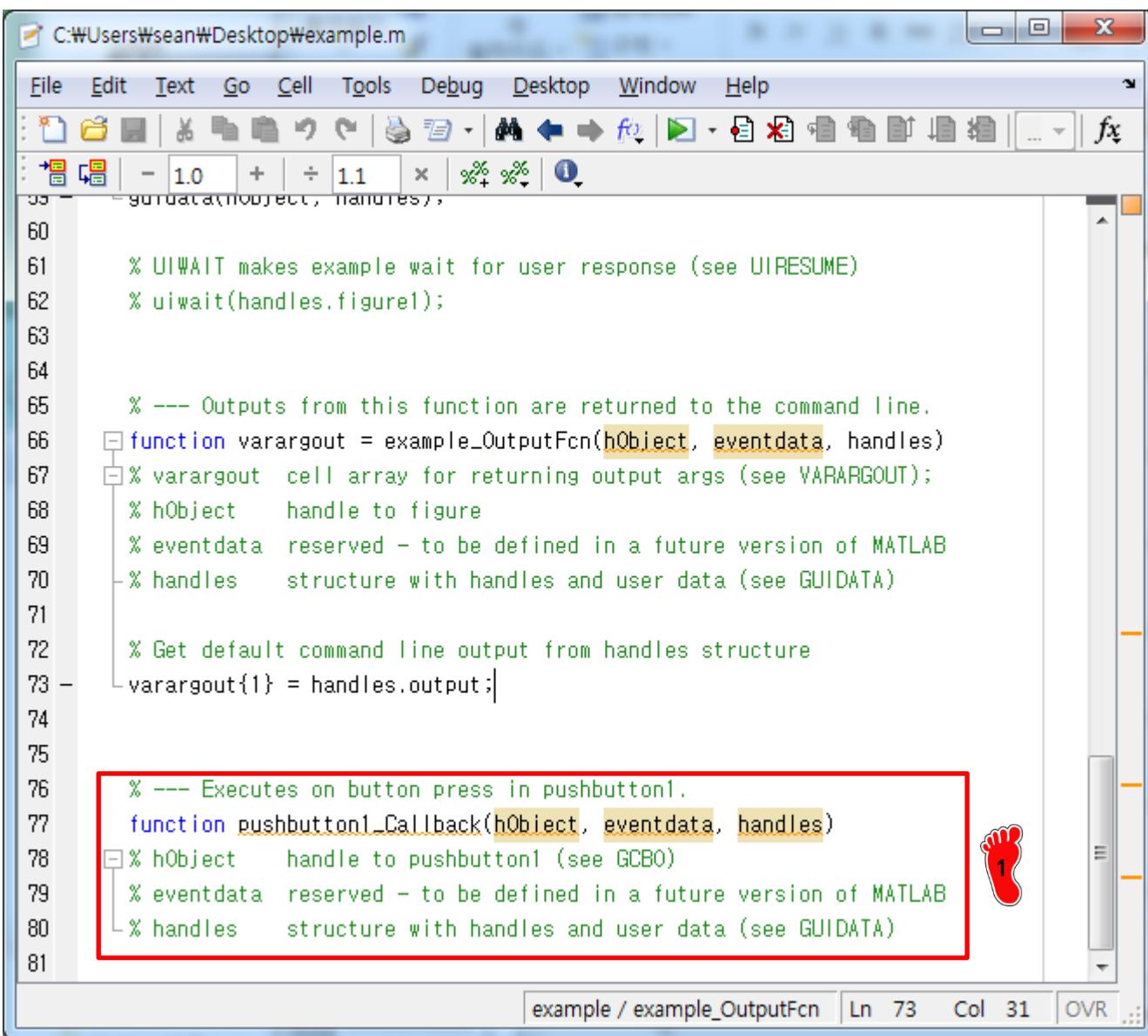


1 String 을 Close 로 변경

2 FontSize 20 으로 변경



PUSH BUTTON: M-FILE



The screenshot shows the MATLAB Editor window with the file `C:\Users\sean\Desktop\example.m` open. The code defines a function `example_OutputFcn` and a callback function `pushbutton1_Callback`.

```

C:\Users\sean\Desktop\example.m
File Edit Text Go Cell Tools Debug Desktop Window Help
File Edit Text Go Cell Tools Desktop Window Help
- + 1.0 ÷ 1.1 × % % 1
55 - guidata(hObject, handles);
56
57 % UIWAIT makes example wait for user response (see UIRESUME)
58 % uiwait(handles.figure1);
59
60
61 % --- Outputs from this function are returned to the command line.
62
63 function varargout = example_OutputFcn(hObject, eventdata, handles)
64 % varargout cell array for returning output args (see VARARGOUT);
65 % hObject handle to figure
66 % eventdata reserved - to be defined in a future version of MATLAB
67 % handles structure with handles and user data (see GUIDATA)
68
69 % Get default command line output from handles structure
70 varargout{1} = handles.output;
71
72
73 % --- Executes on button press in pushbutton1.
74
75 function pushbutton1_Callback(hObject, eventdata, handles)
76 % hObject handle to pushbutton1 (see GCBO)
77 % eventdata reserved - to be defined in a future version of MATLAB
78 % handles structure with handles and user data (see GUIDATA)
79
80
81

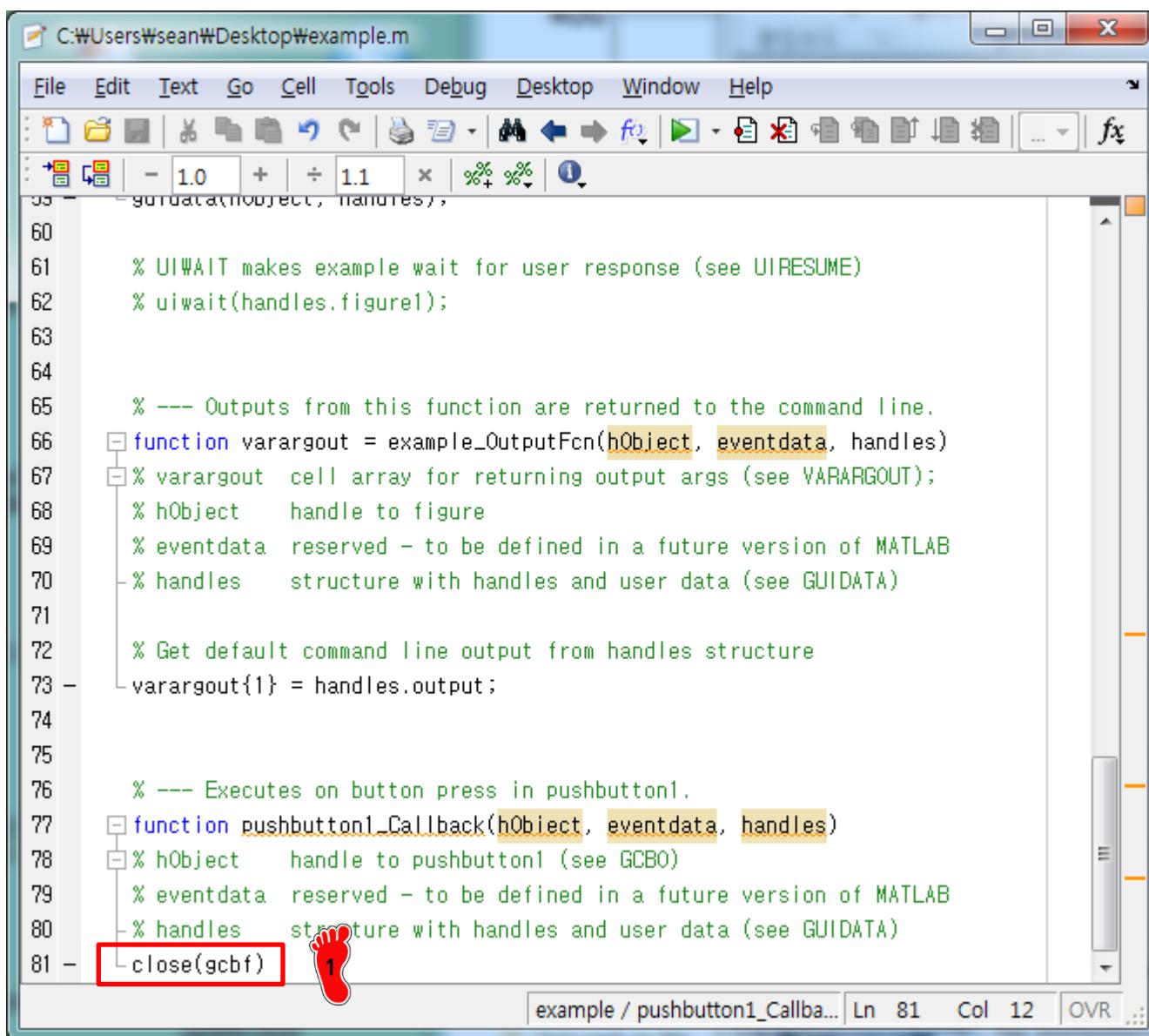
```

The line `% --- Executes on button press in pushbutton1.` and the definition of `pushbutton1_Callback` are highlighted with a red box.

1 M-file 에
pushbutton1_Callback 함수
가 생성됨

이 함수 밑에 m-file 명령어
를 입력하고 저장하면
pushbutton 을 클릭했을 때
실행됨

PUSH BUTTON: M-FILE



The screenshot shows the MATLAB Editor window with the file `example.m` open. The code defines a function `example` with a push button callback. The callback function is named `pushbutton1_Callback`. The code includes comments explaining the use of `UIWAIT` and `GUIDATA`.

```

C:\Users\sean\Desktop\example.m
File Edit Text Go Cell Tools Debug Desktop Window Help
File Edit Text Go Cell Tools Desktop Window Help
- + ÷ × % %
1.0 1.1
35 - guidata(hObject, handles);
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
% --- Outputs from this function are returned to the command line.
function varargout = example_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
close(gcf);

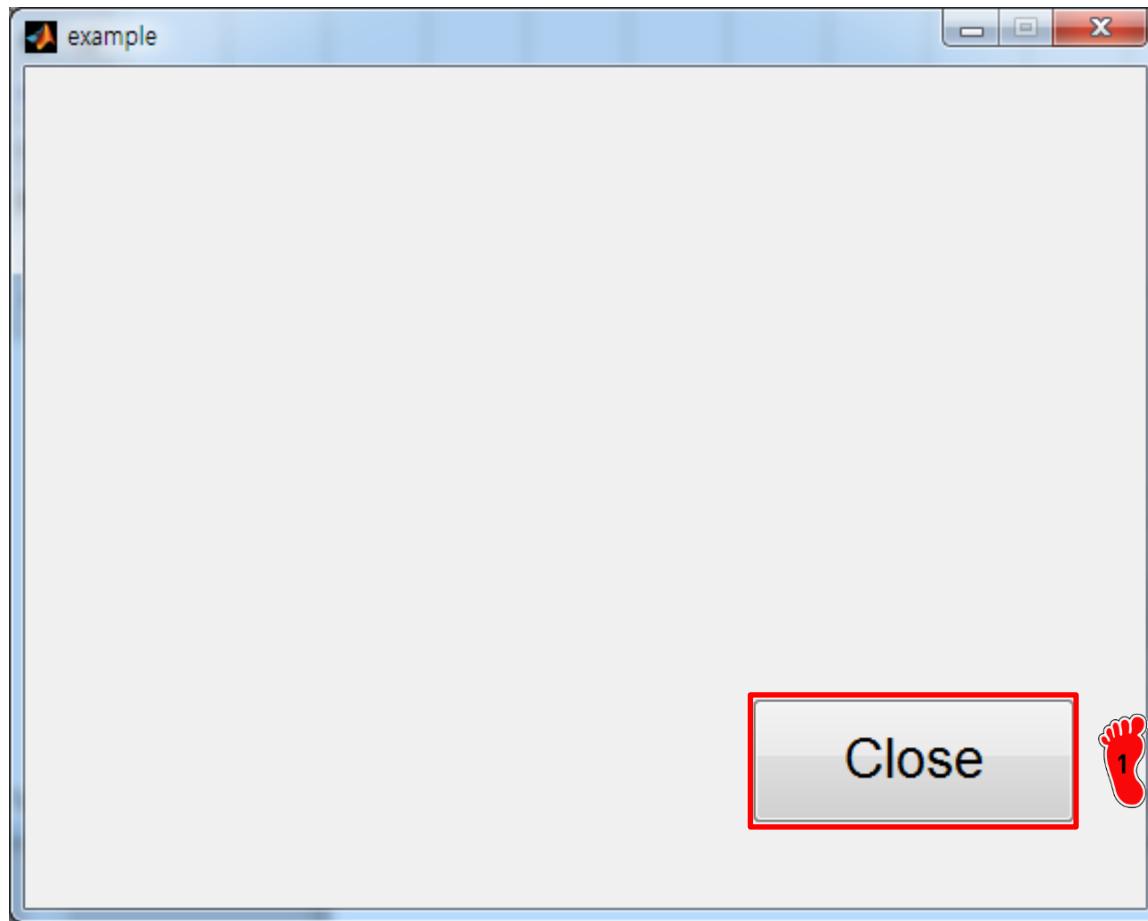
```



close(gcf) 입력

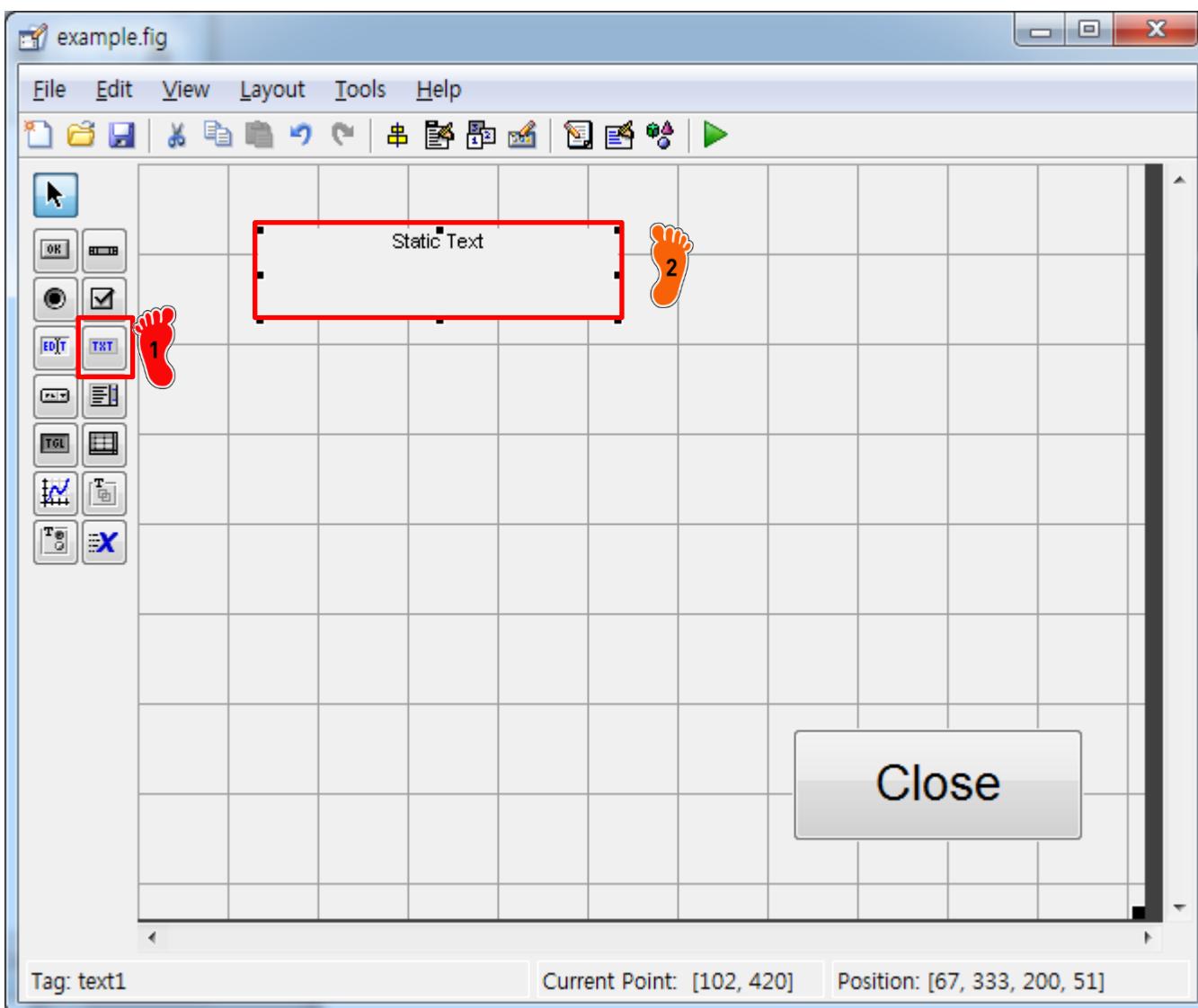
이 명령어는 현재 GUI 창을
닫는 명령어

PUSH BUTTON: GUI



1 GUI 창을 실행시켜 Close pushbutton 을 클릭하면 창이 닫힘

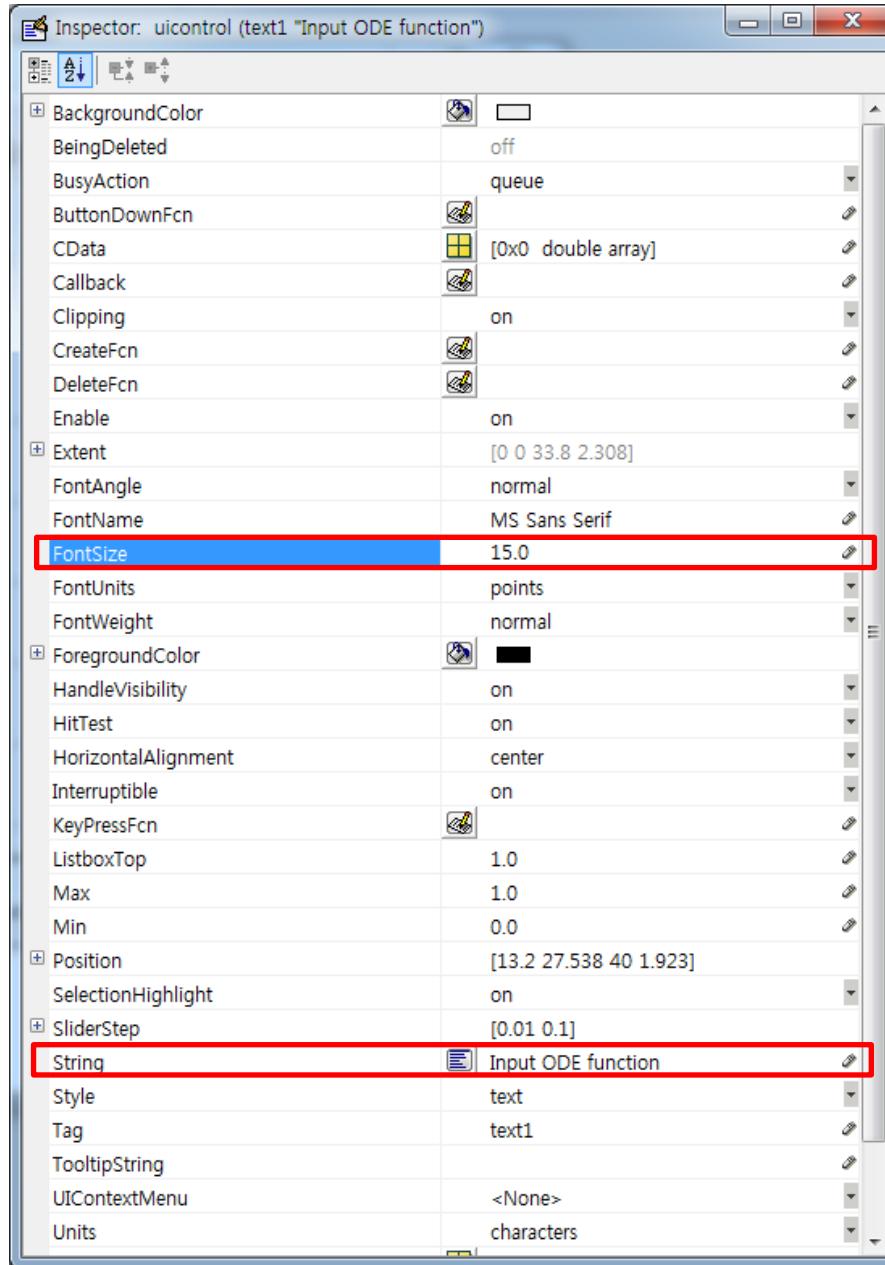
STATIC TEXT BOX



1 static text 아이콘 클릭

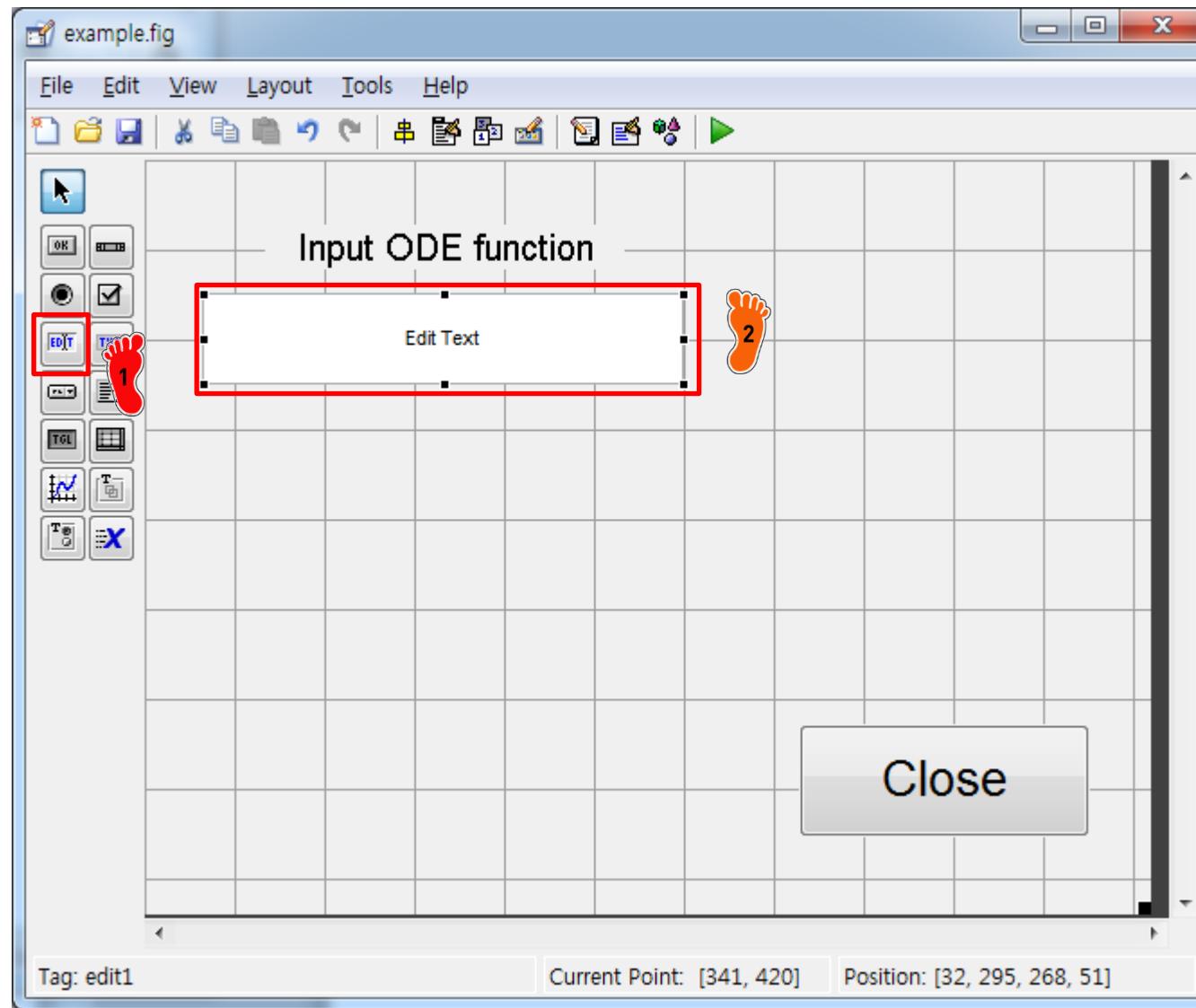
2 static text box 생성
더블 클릭

STATIC TEXT BOX: INSPECTOR



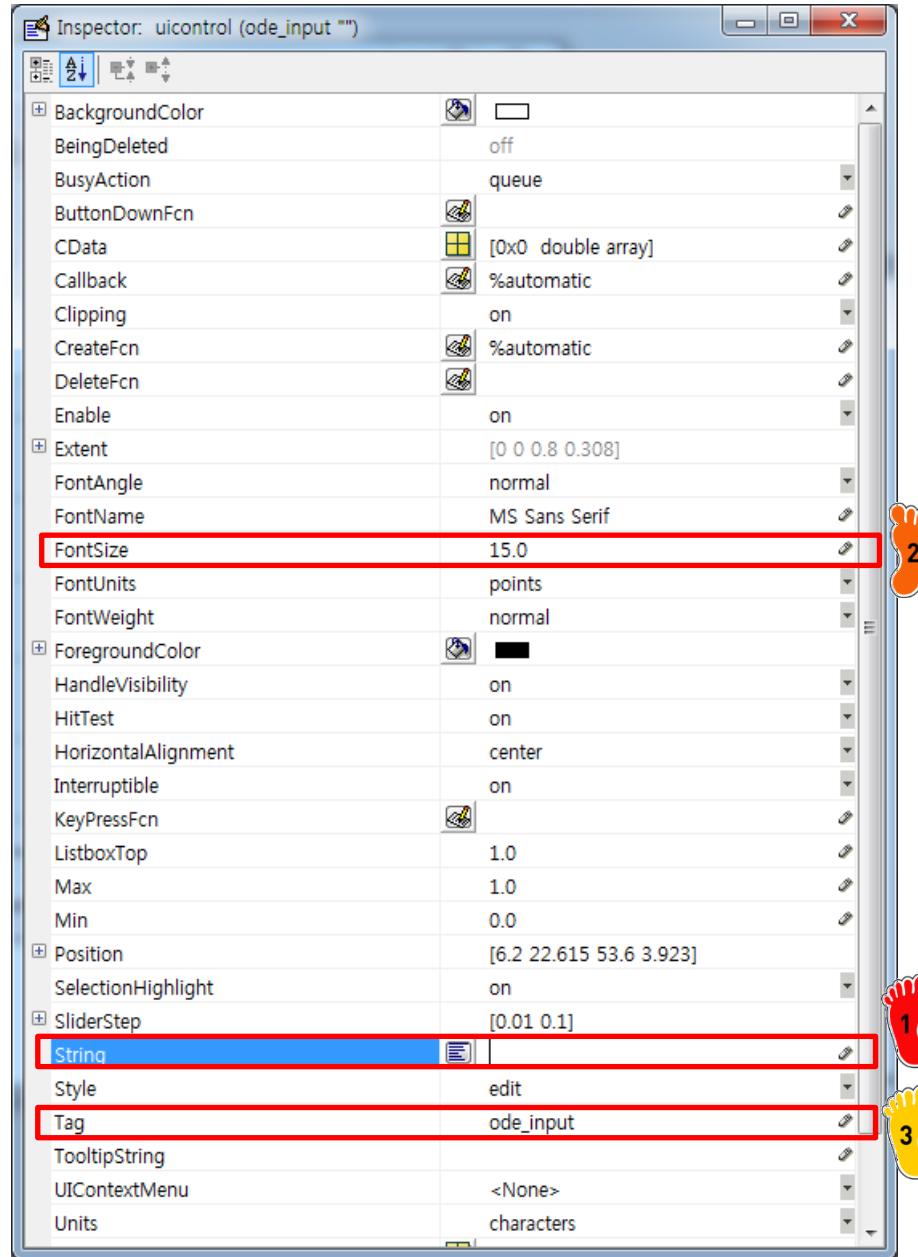
- 1 String 을 Input ODE function 으로 변경
2 FontSize 15로 변경

EDIT TEXT BOX



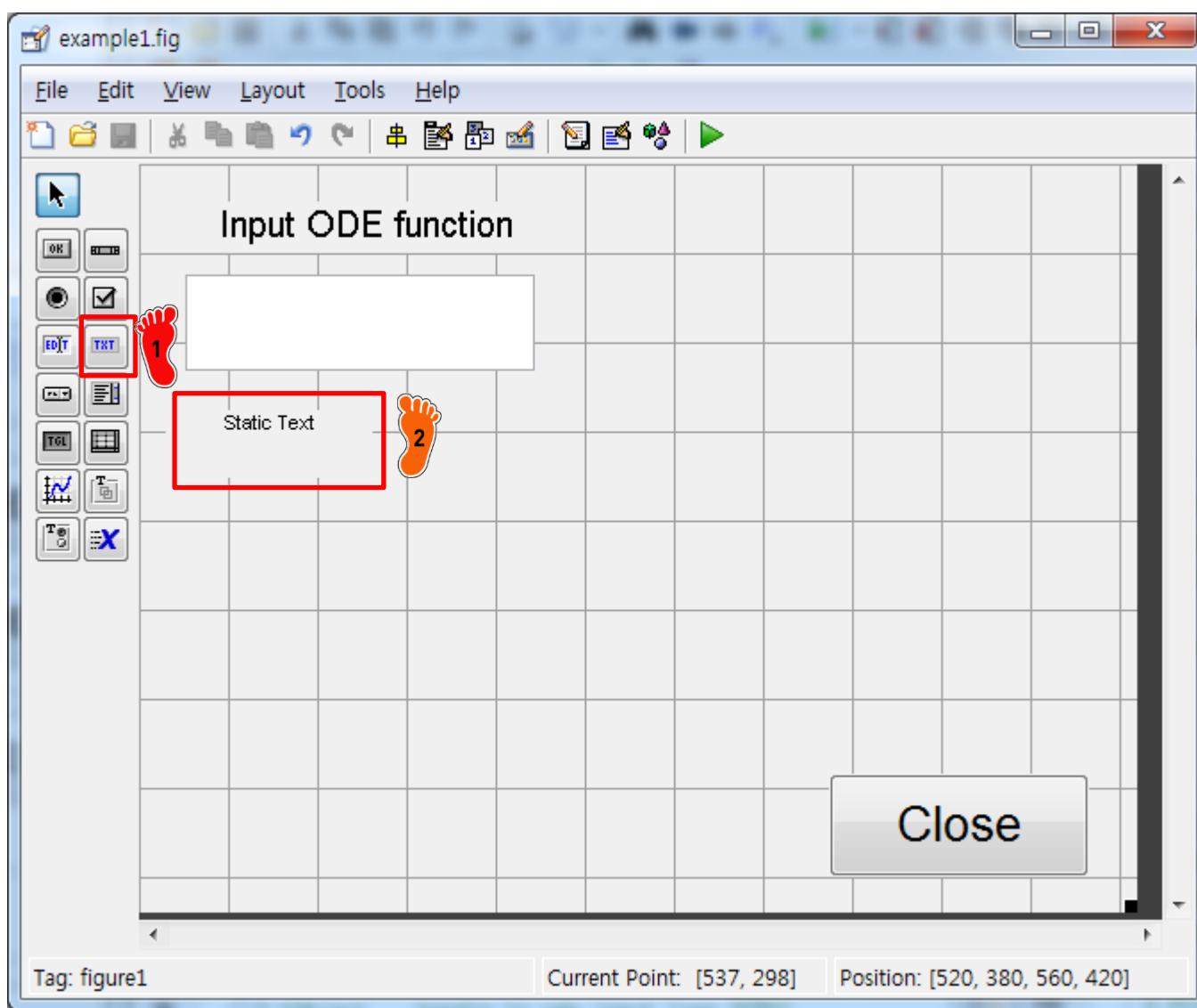
- 1 edit text 아이콘 클릭
- 2 edit text box 생성 더블 클릭

EDIT TEXT BOX: INSPECTOR



- 1 String 빈칸으로 변경
- 2 FontSize 15로 변경
- 3 Tag 를 ode_input 으로 변경

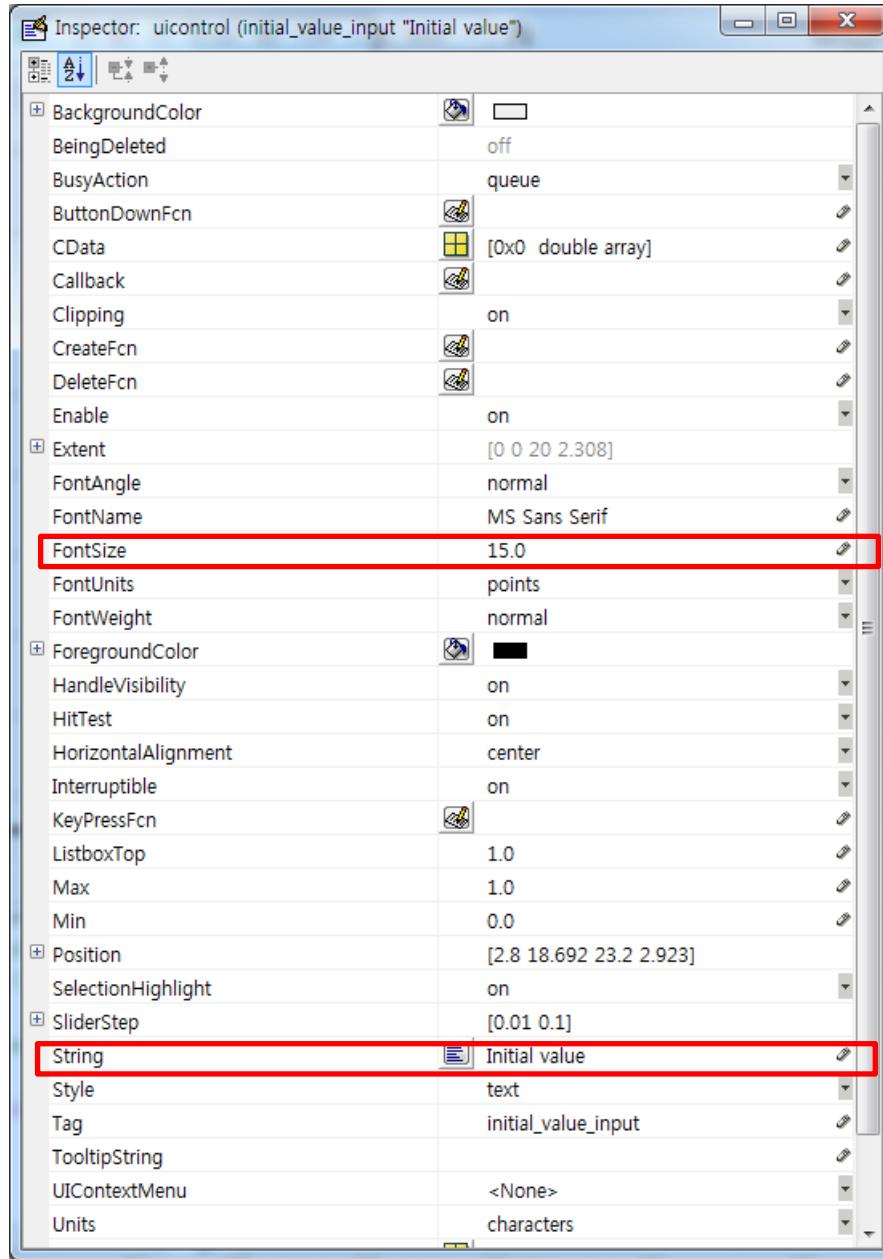
STATIC TEXT BOX



1 static text 아이콘 클릭

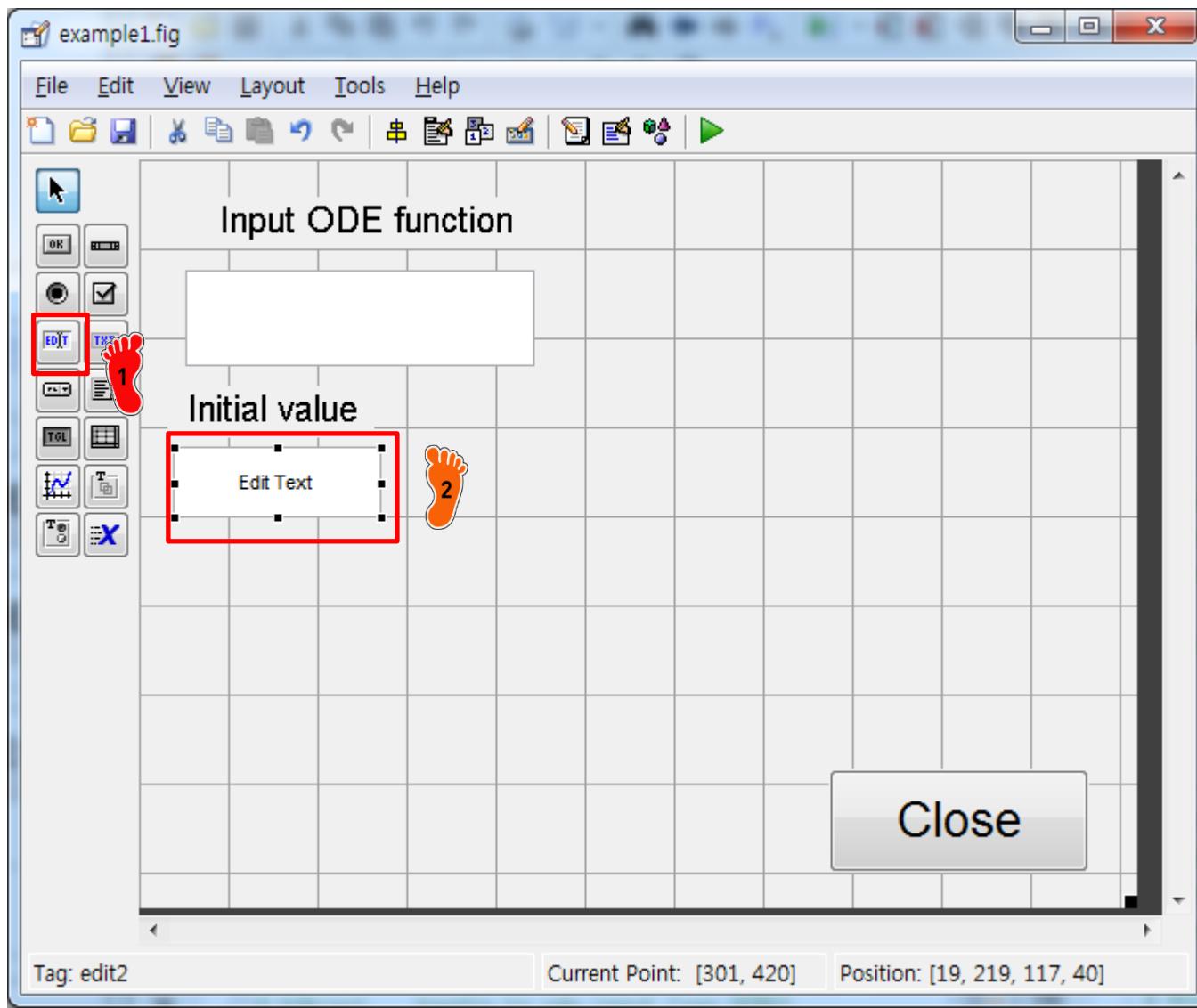
2 static text box 생성
더블 클릭

STATIC TEXT BOX: INSPECTOR



- 1 String 을 Initial value 로 변경
- 2 FontSize 15로 변경

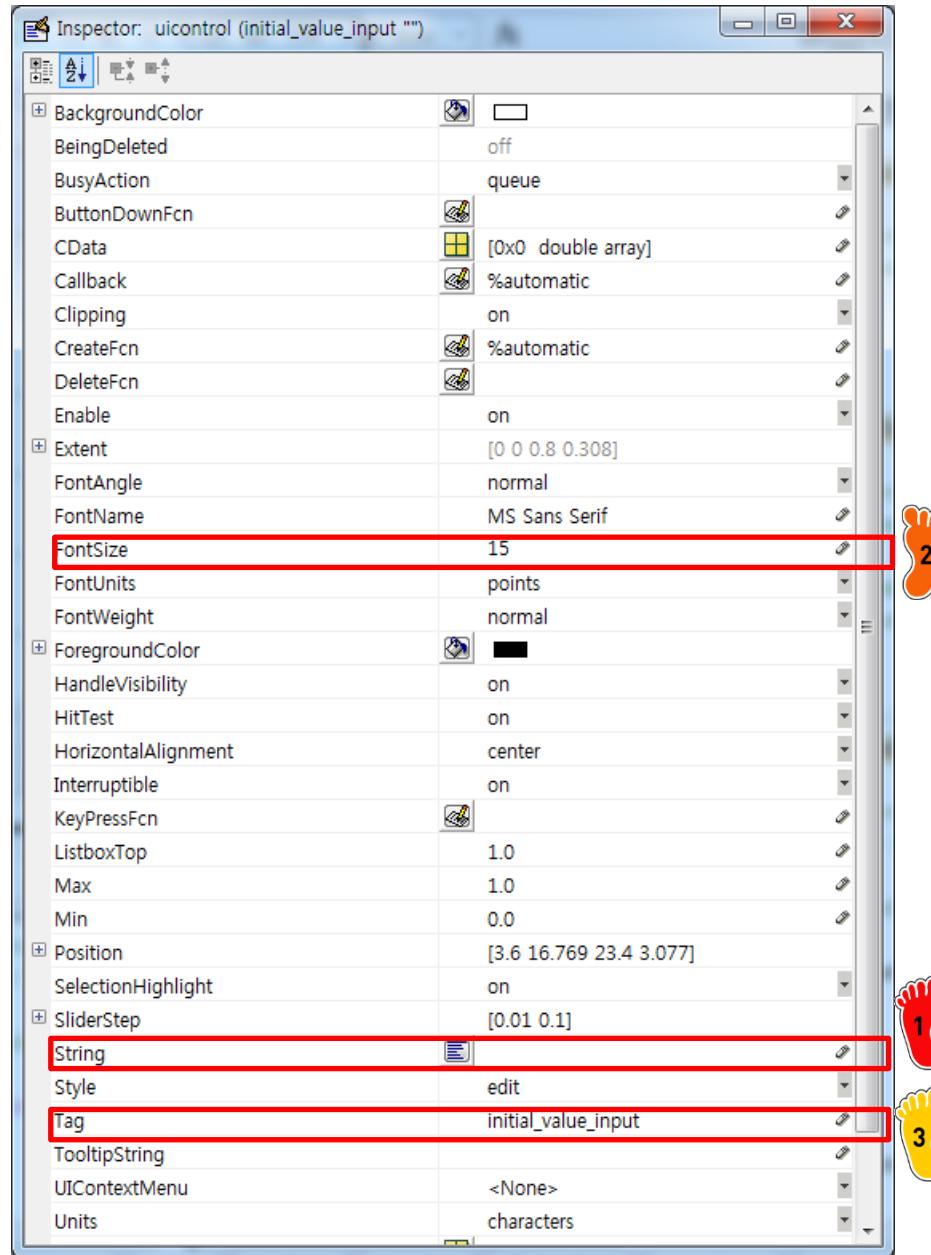
EDIT TEXT BOX



1 edit text 아이콘 클릭

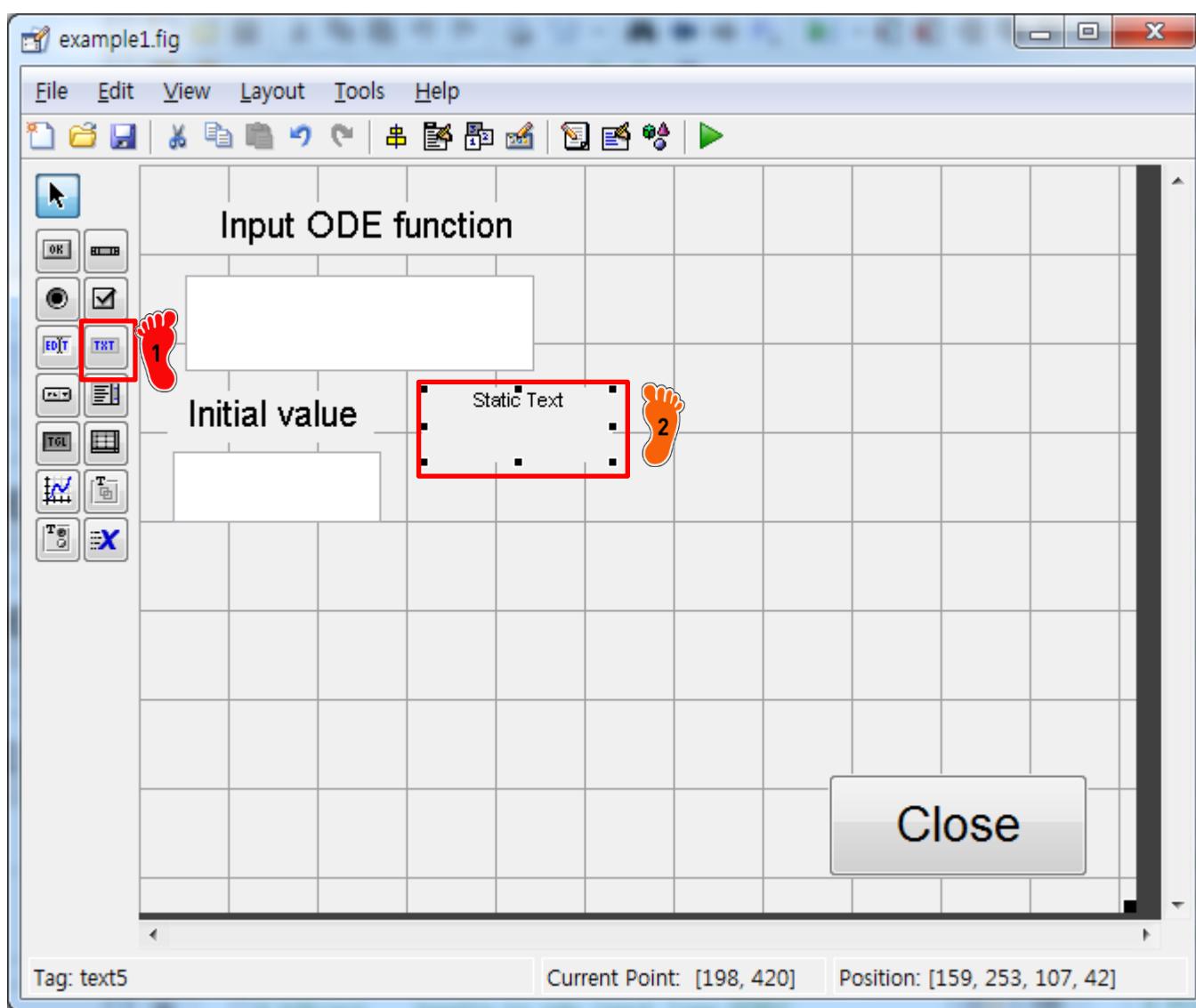
2 edit text box 생성
더블 클릭

EDIT TEXT BOX: INSPECTOR



- 1 String 빈칸으로 변경
- 2 FontSize 15로 변경
- 3 Tag 를 initial_value_input 으로 변경

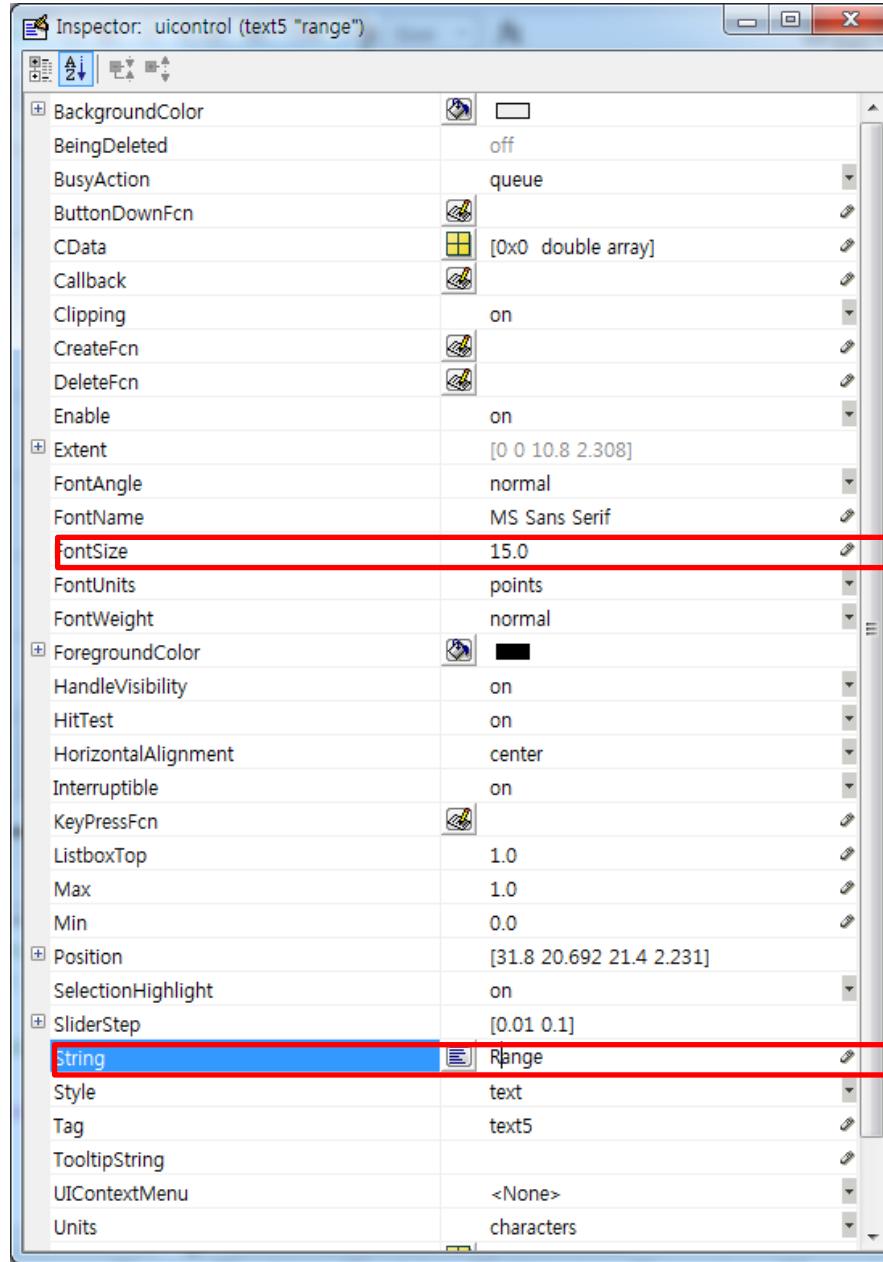
STATIC TEXT BOX



1 static text 아이콘 클릭

2 static text box 생성
더블 클릭

STATIC TEXT BOX: INSPECTOR



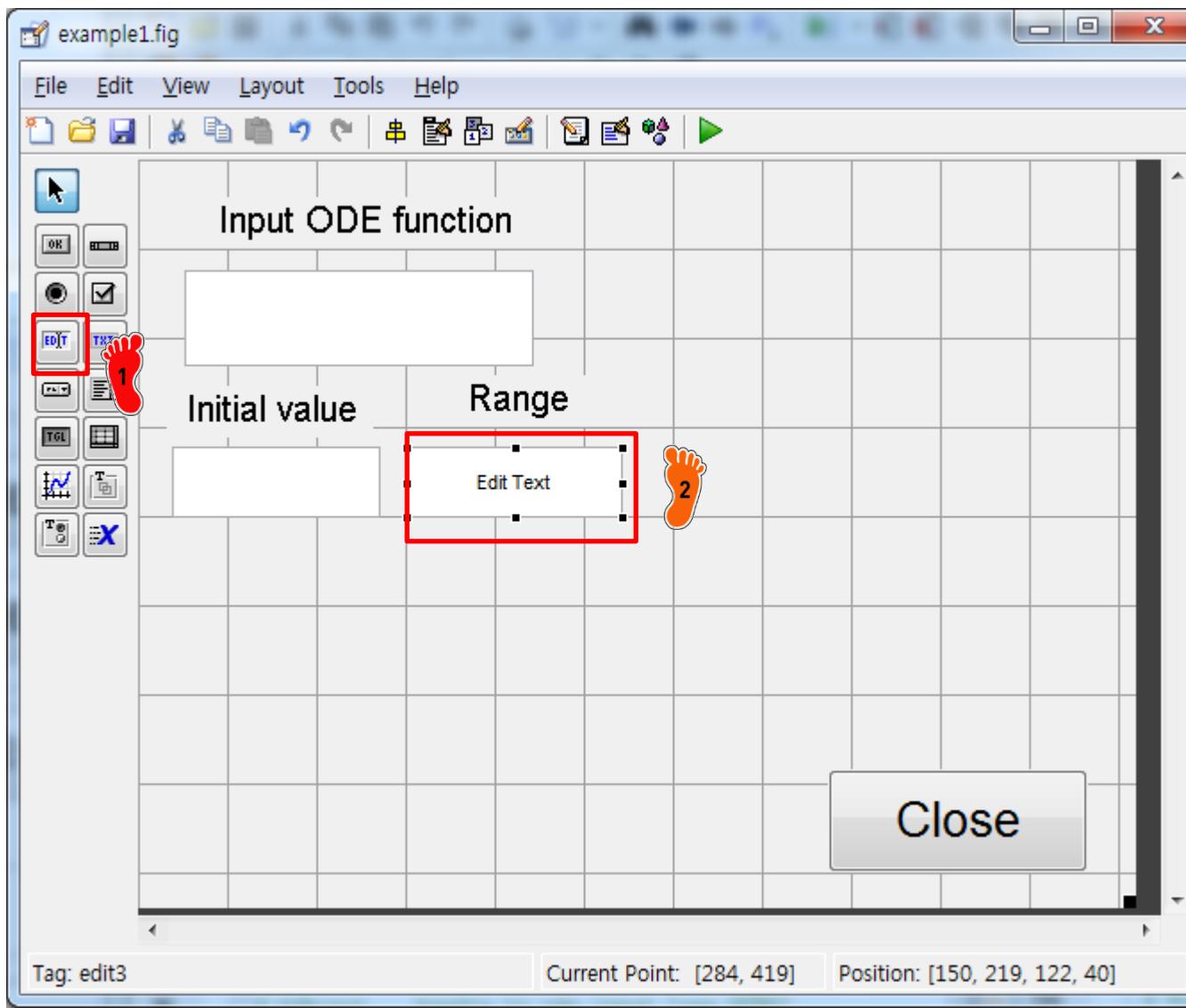
1 String 을 Range 로 변경

2 FontSize 15로 변경

2

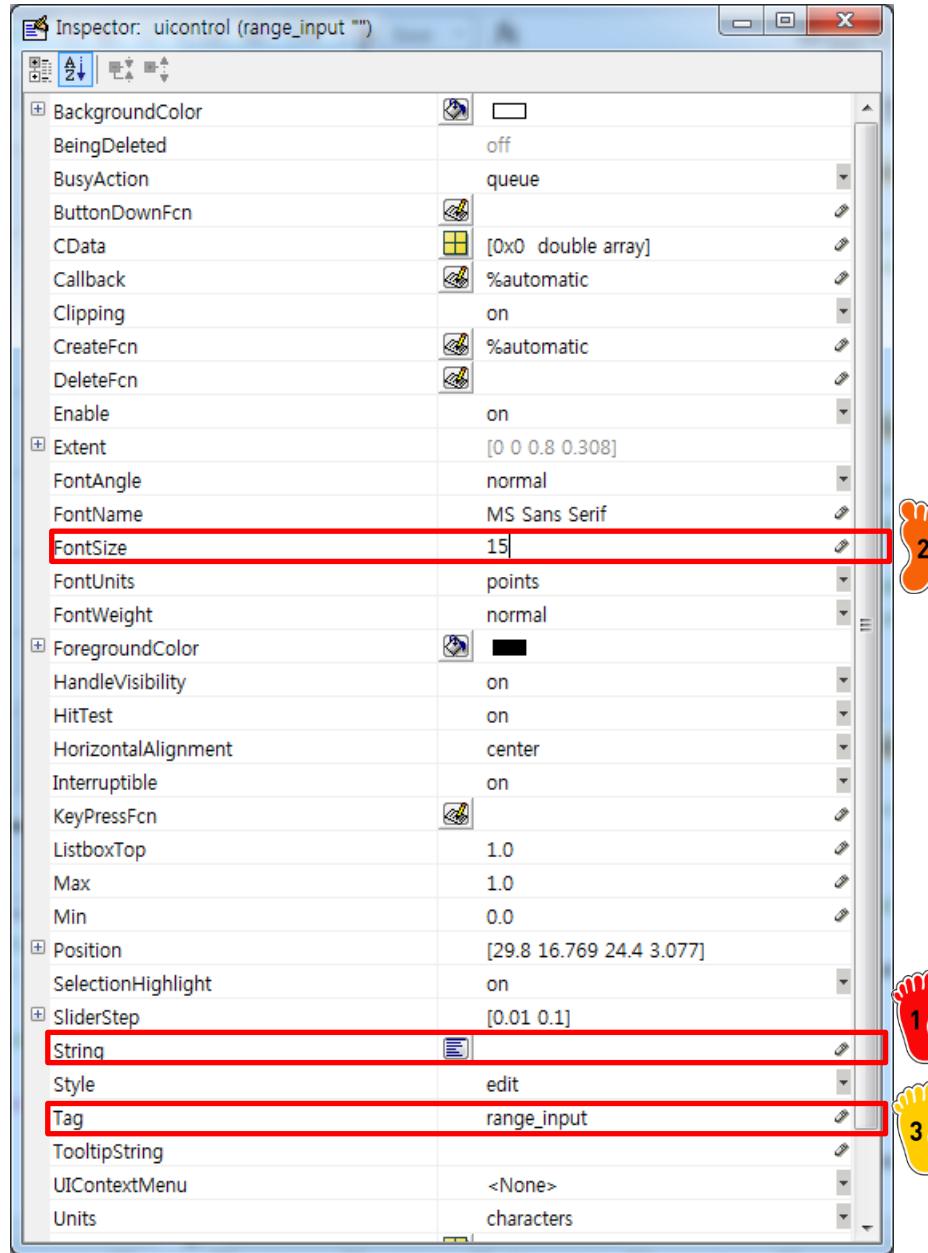
1

EDIT TEXT BOX



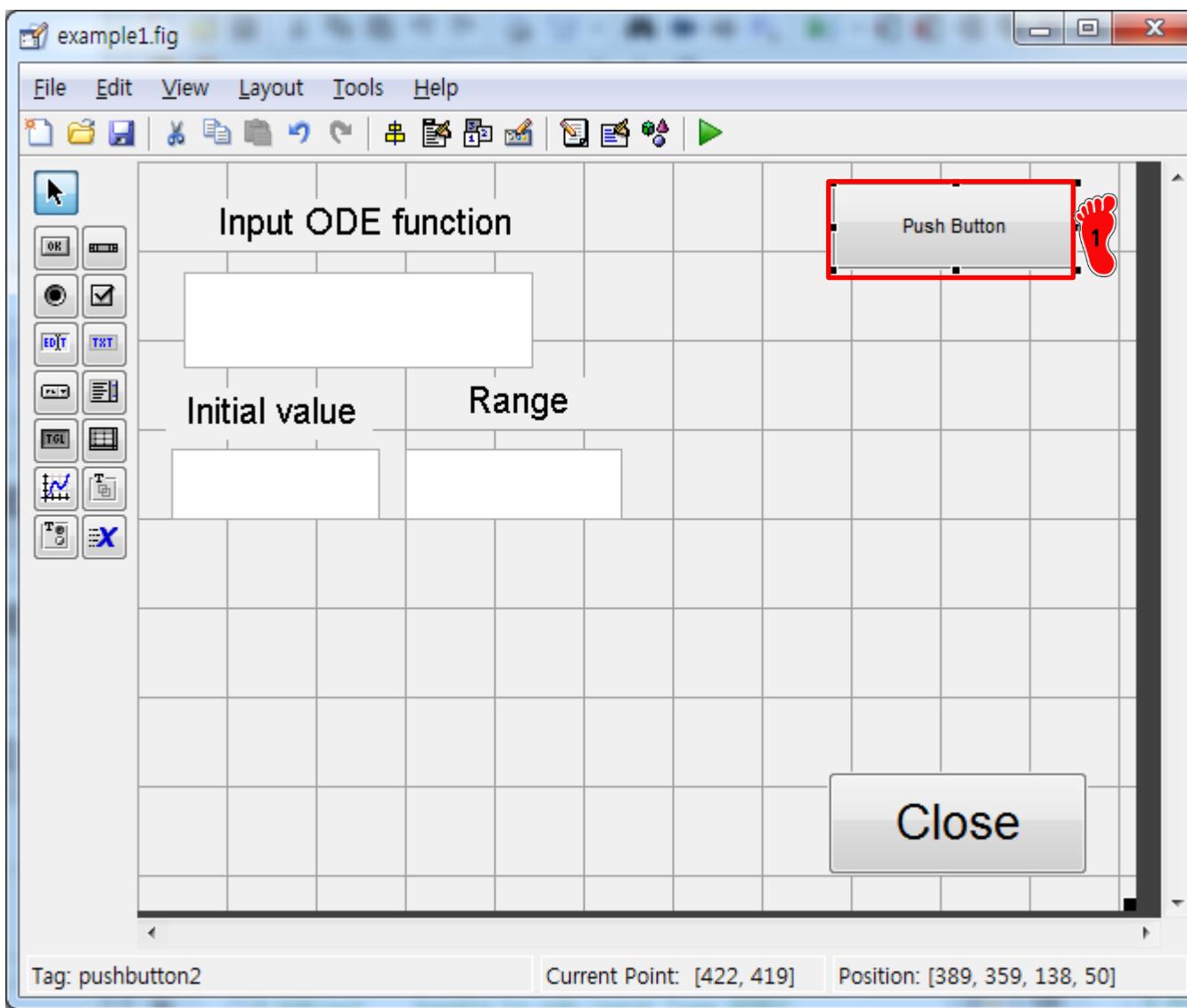
- 1 edit text 아이콘 클릭
- 2 edit text box 생성 더블 클릭

EDIT TEXT BOX: INSPECTOR



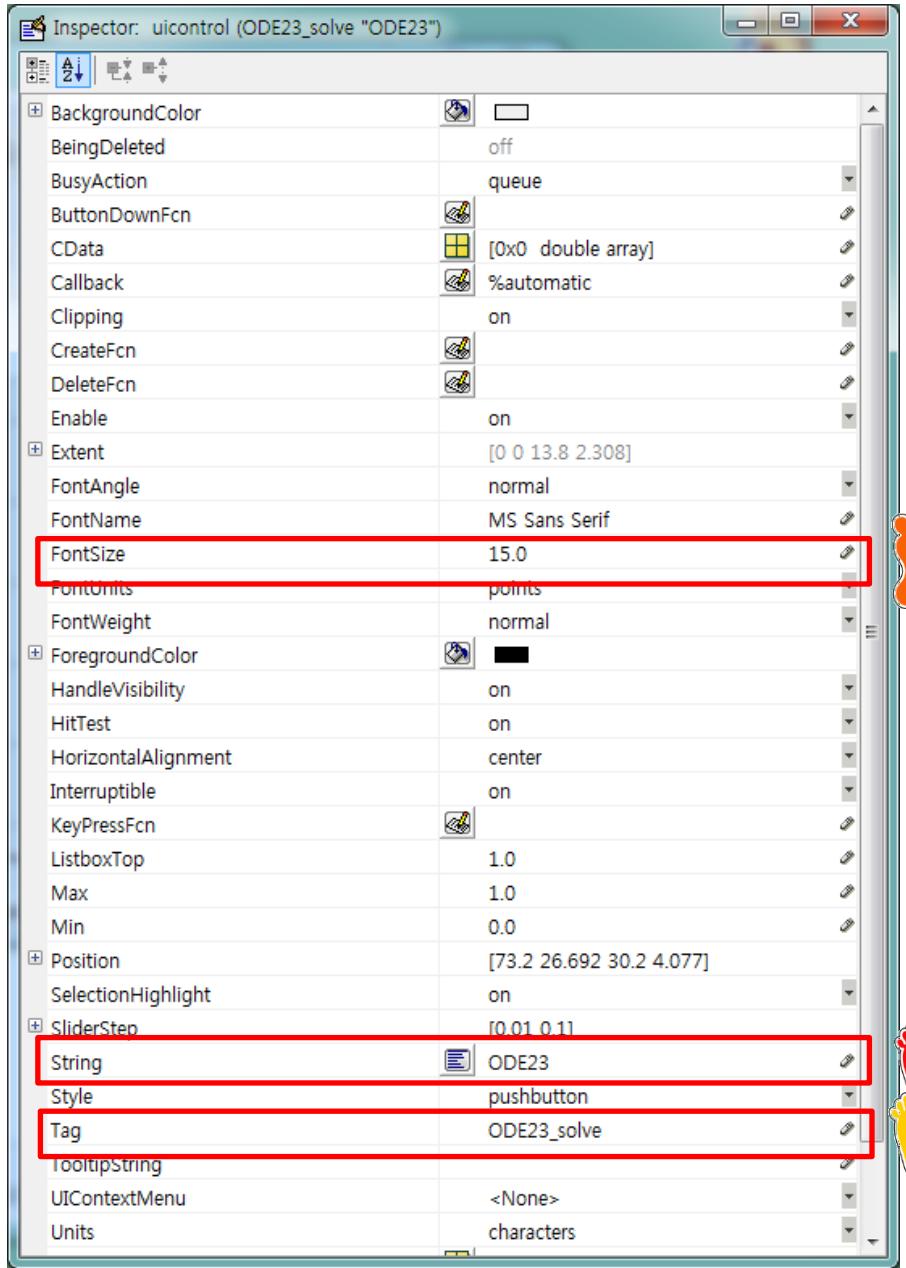
- 1 String 빈칸으로 변경
- 2 FontSize 15로 변경
- 3 Tag 를 range_input 으로 변경

ODE23 PUSH BUTTON



1 pushbutton 생성

ODE23 PUSH BUTTON: INSPECTOR

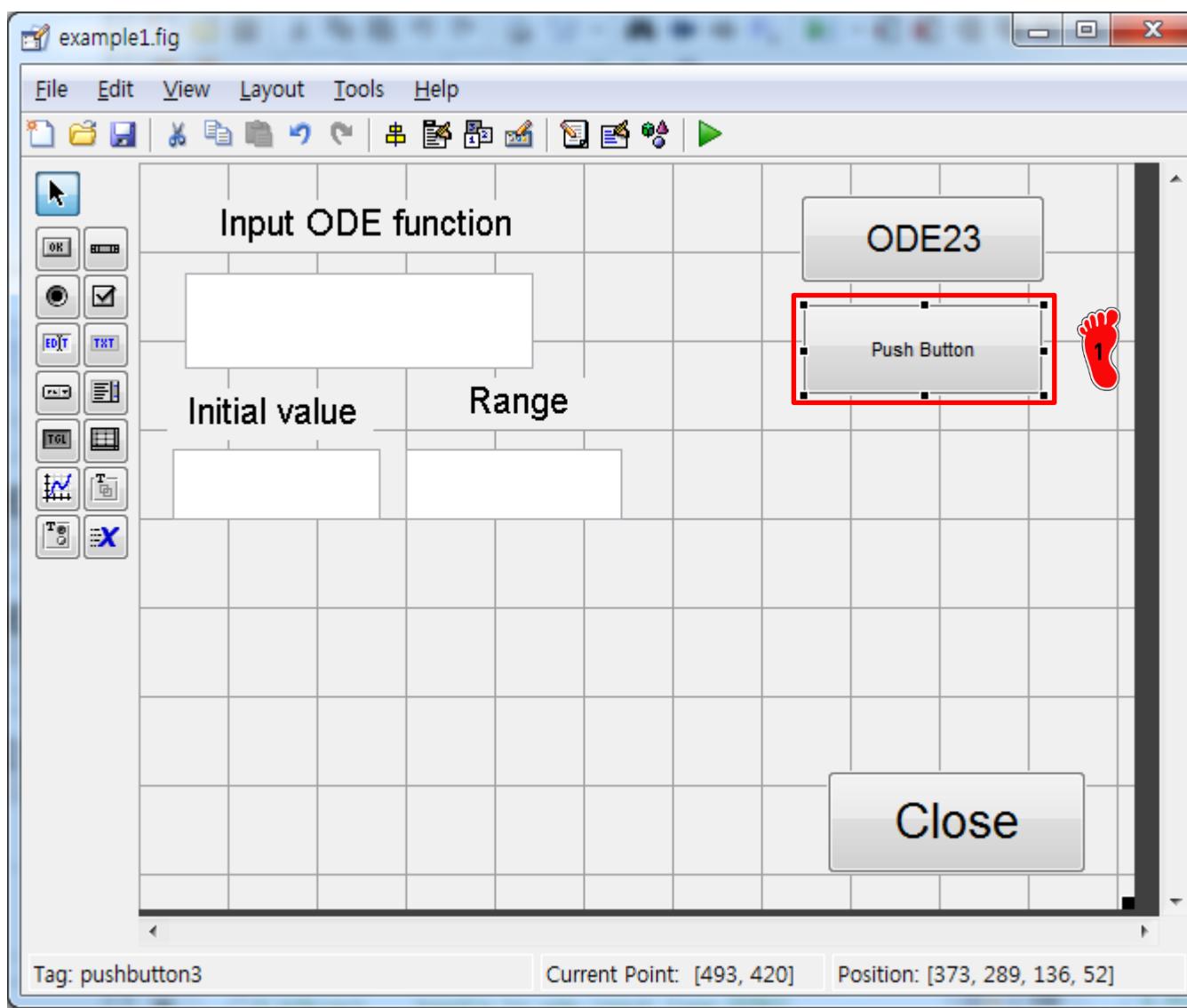


1 String ODE23으로 변경

2 FontSize 15로 변경

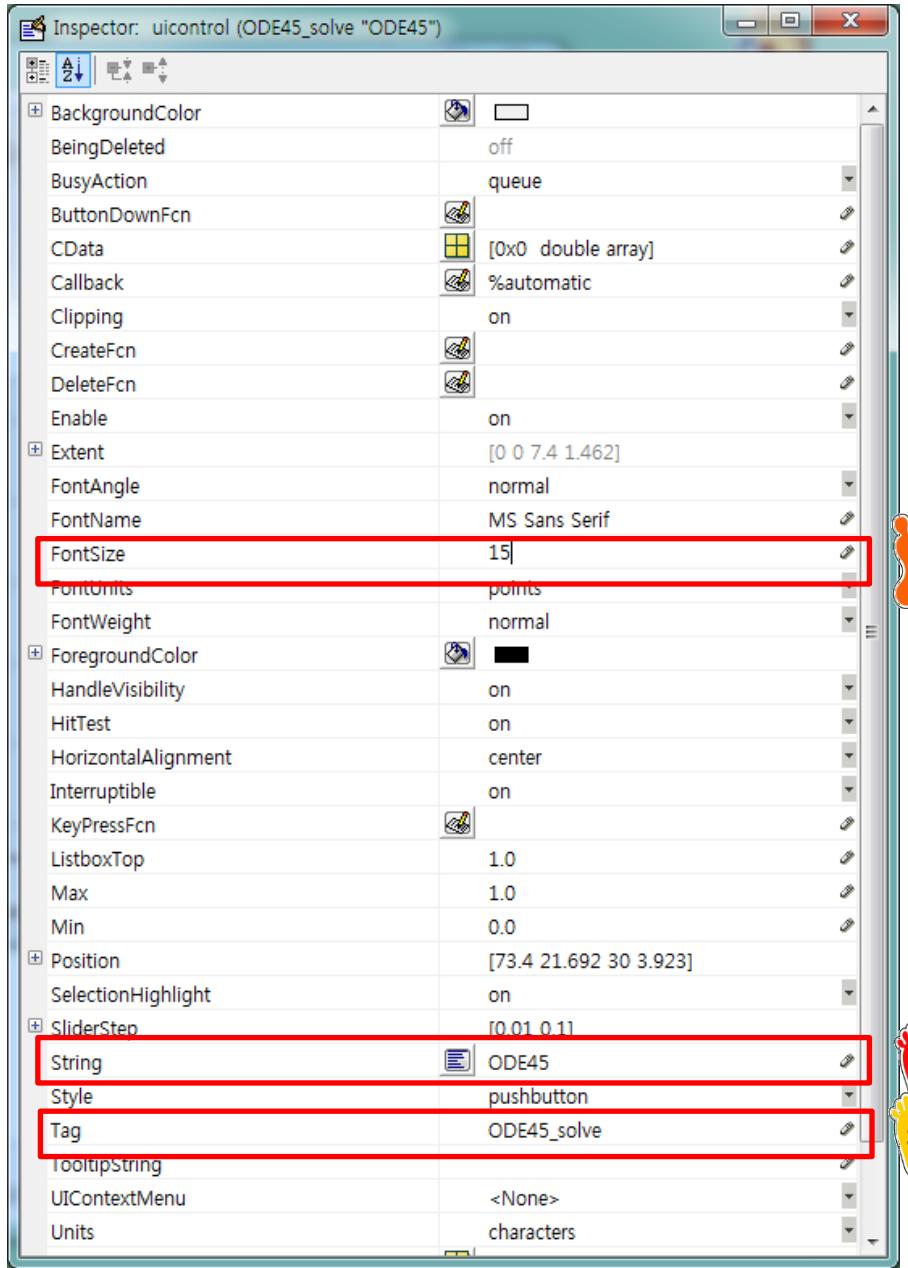
3 Tag 를 ODE23_solve 로 변경

ODE45 PUSH BUTTON



1 pushbutton 생성

ODE45 PUSH BUTTON: INSPECTOR

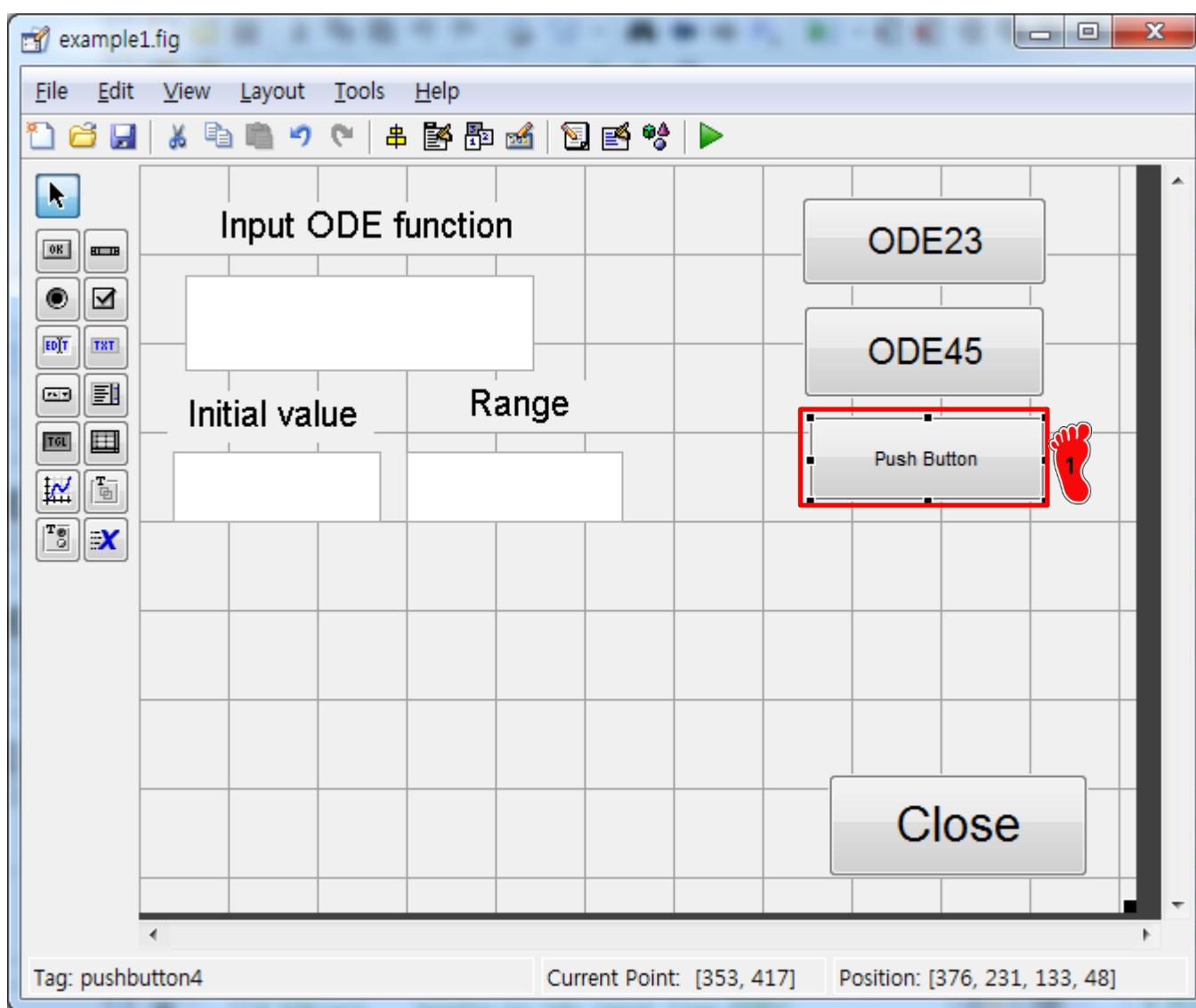


1 String ODE45 로 변경

2 FontSize 15 로 변경

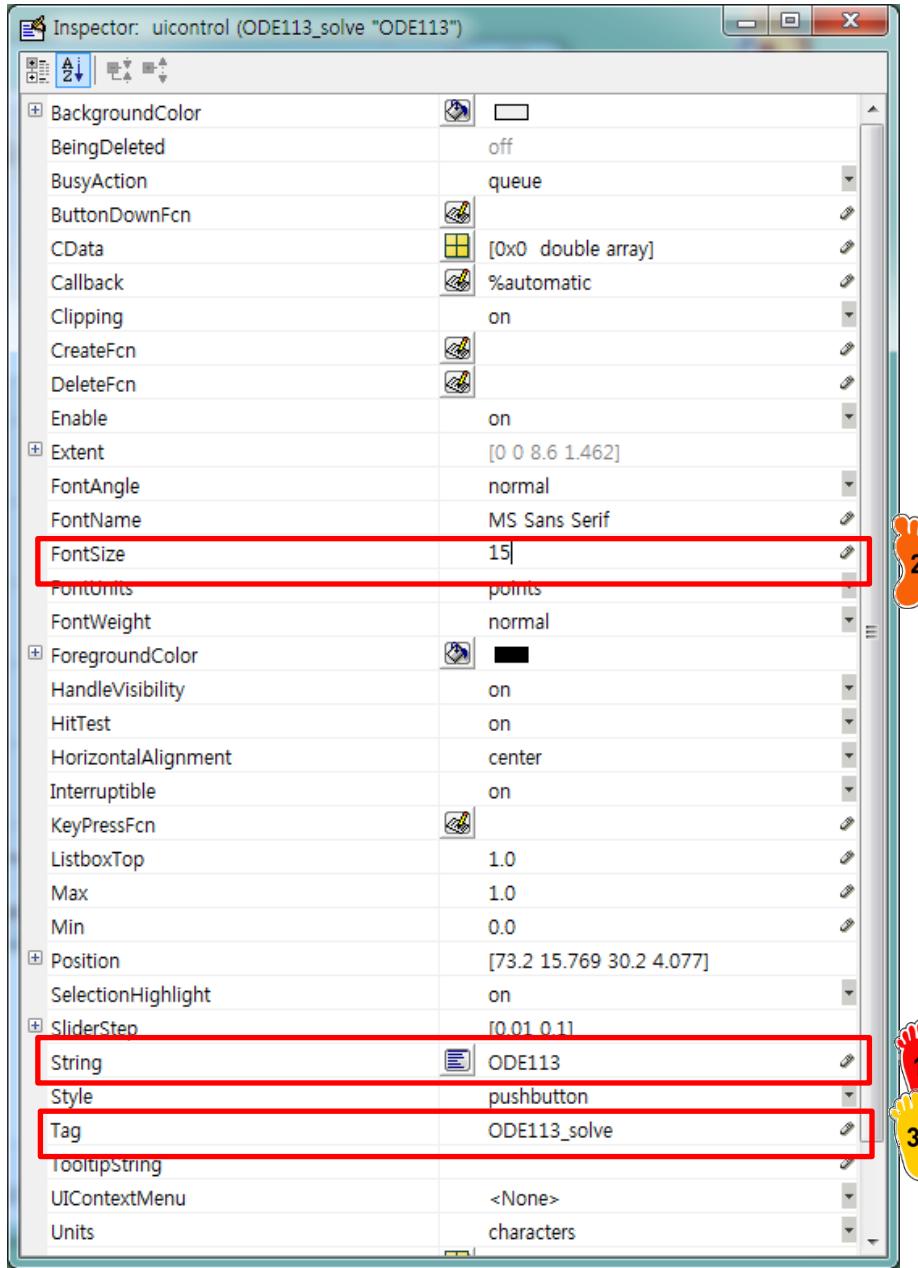
3 Tag 를 ODE45_solve 로 변경

ODE113 PUSH BUTTON



1 pushbutton 생성

ODE113 PUSH BUTTON: INSPECTOR

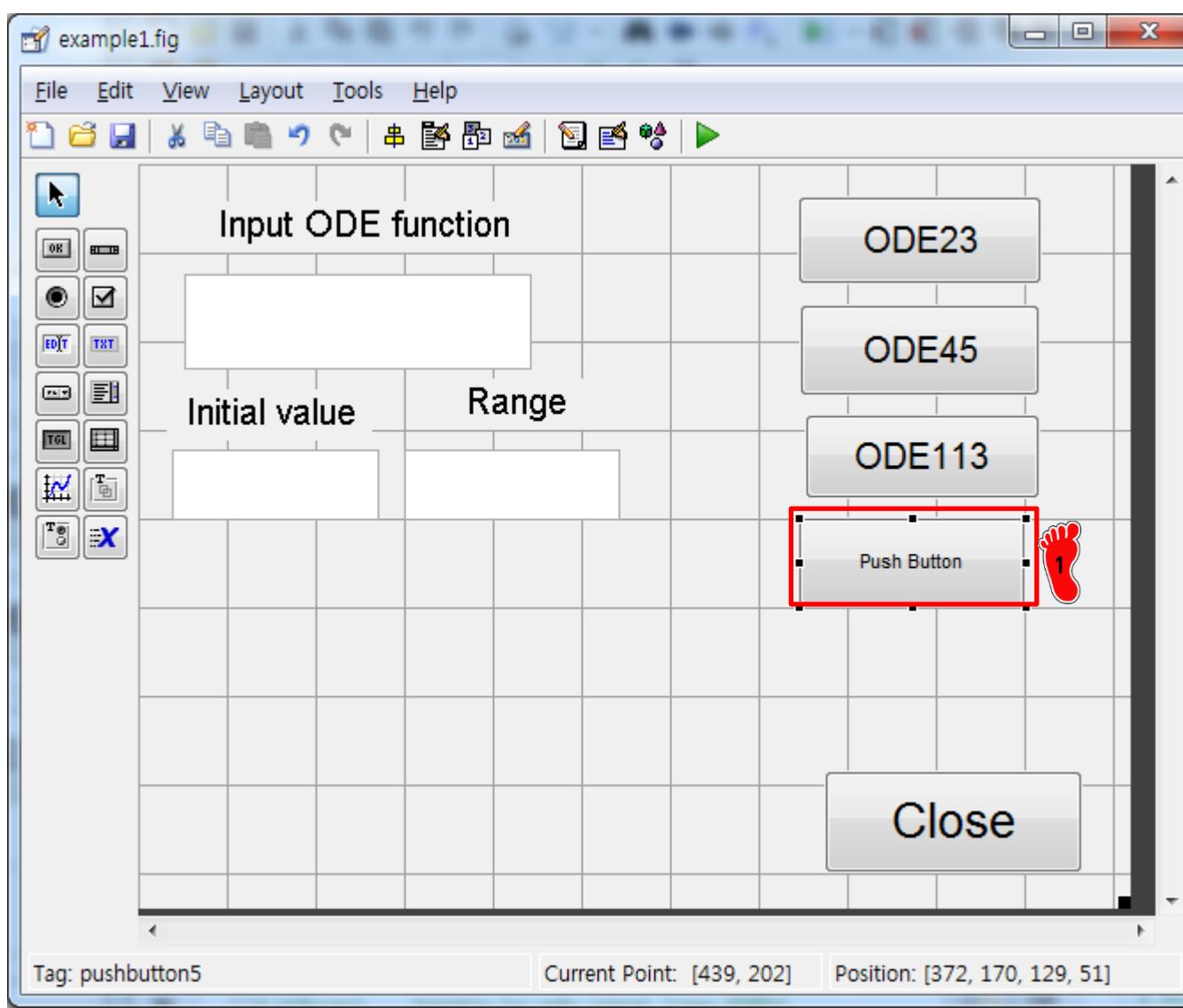


1 String ODE113 으로 변경

2 FontSize 15 로 변경

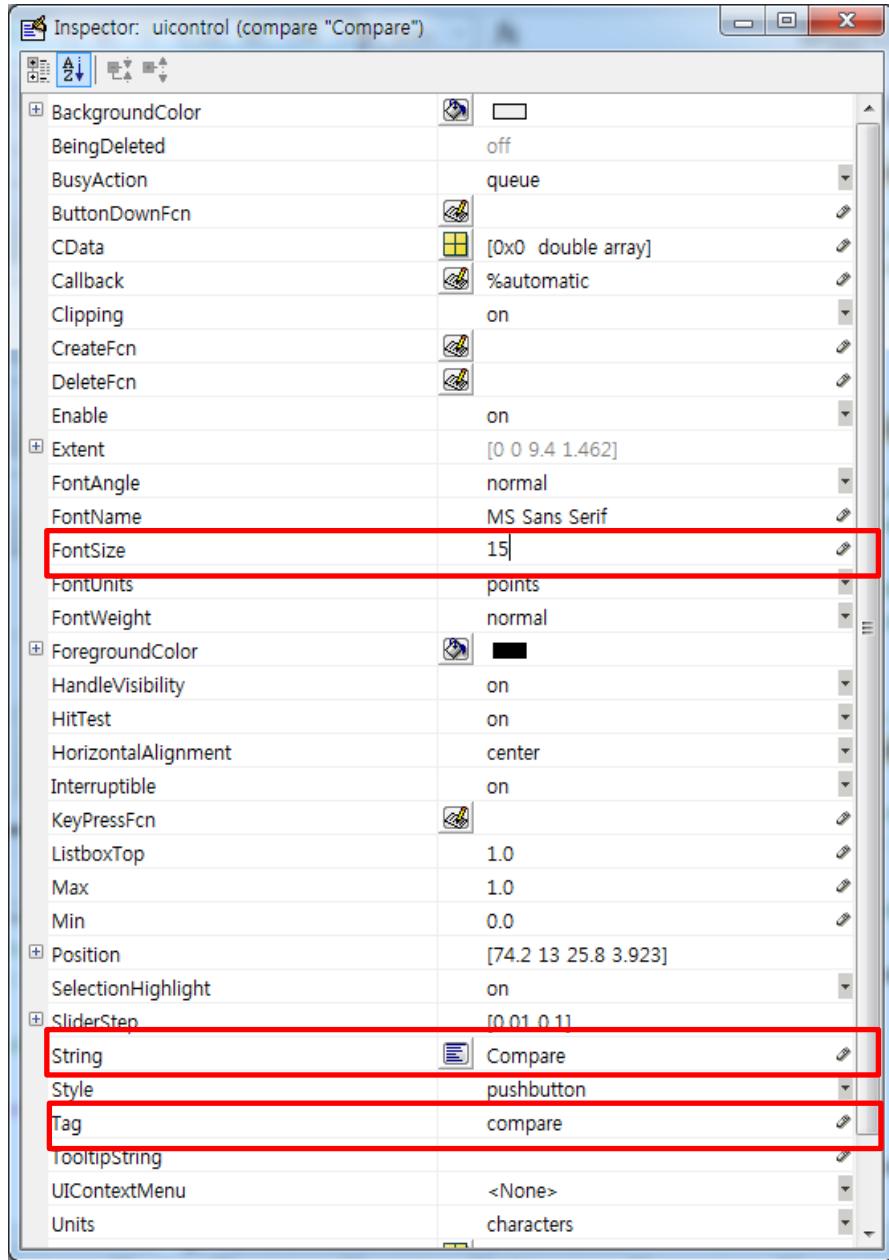
3 Tag 를 ODE113_solve 로 변경

COMPARE PUSH BUTTON



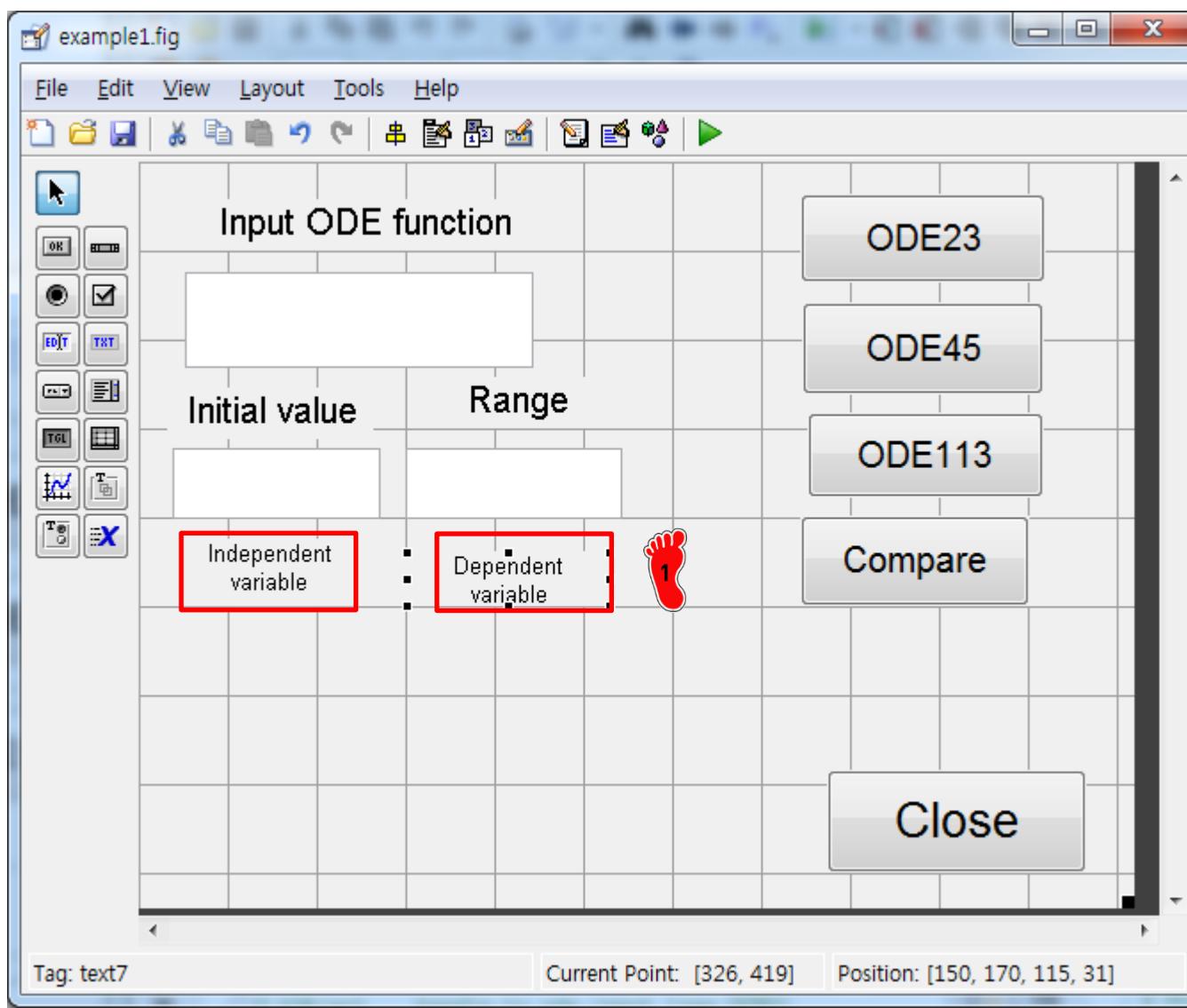
1 pushbutton 생성

COMPARE PUSH BUTTON: INSPECTOR



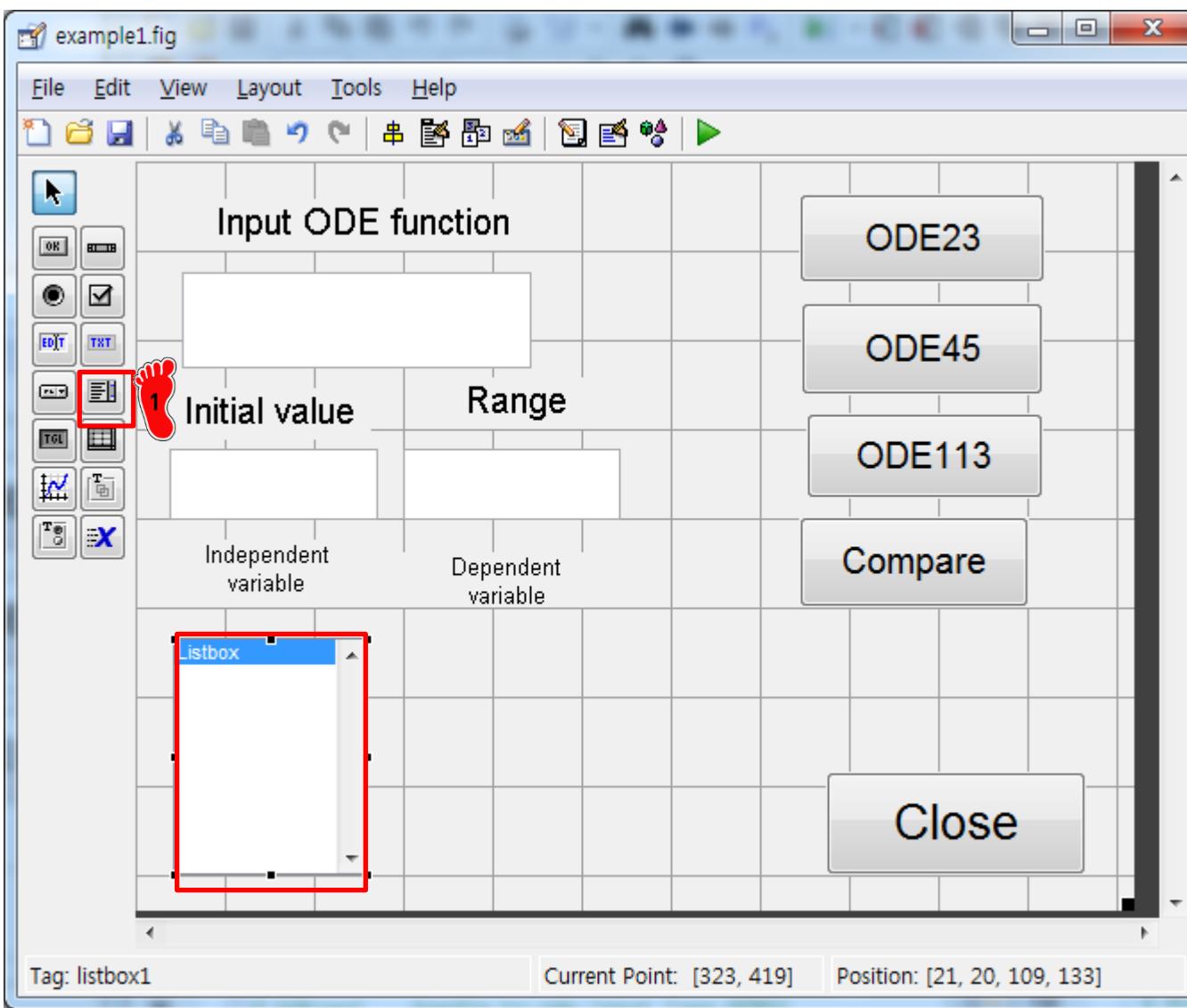
- 1 String Compare 으로 변경
- 2 FontSize 15 로 변경
- 3 Tag 를 compare 로 변경

STATIC TEXT BOX



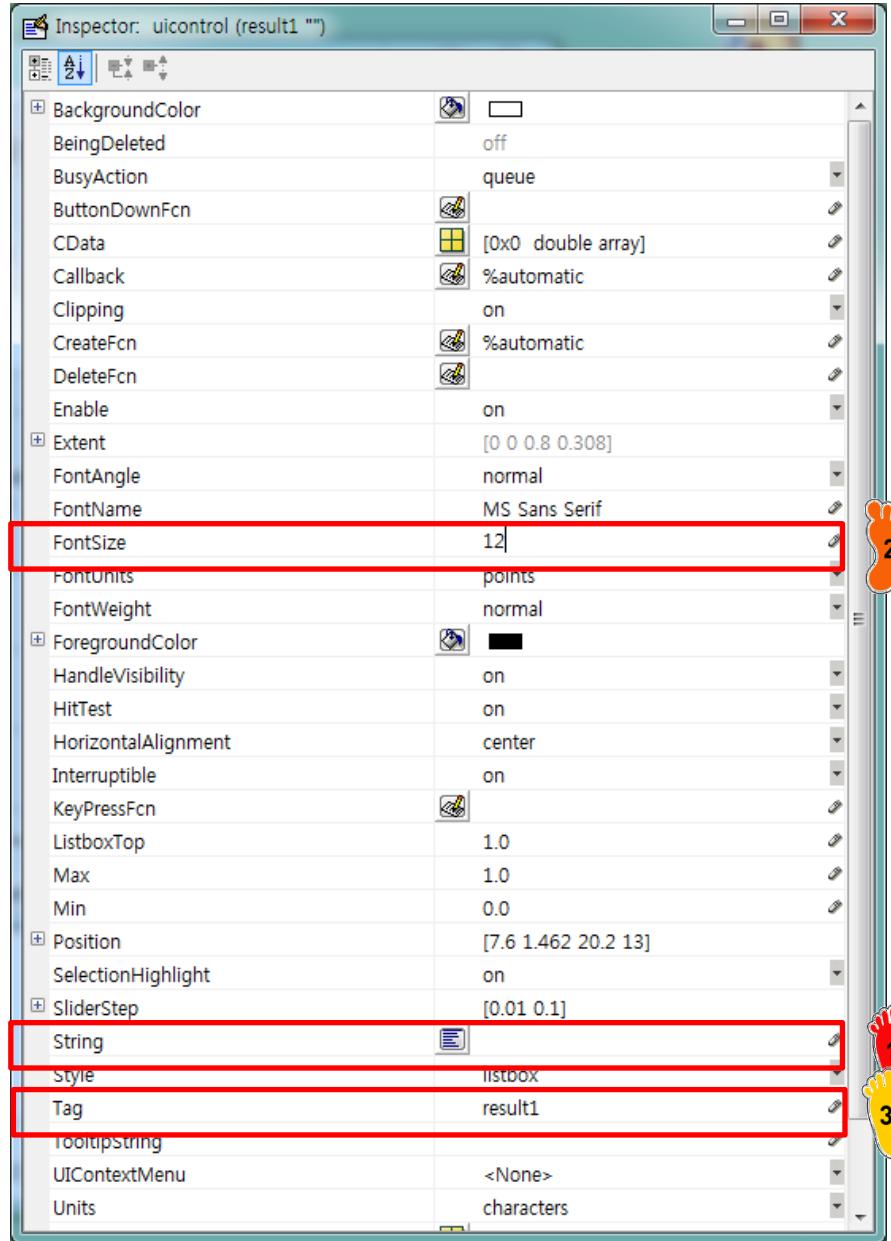
Independent variable 과
Dependent variable 이름을
갖는 static text box 2개 생
성

LIST BOX



- 1 List box icon 클릭
- 2 List box 생성

LIST BOX: INSPECTOR



1 String 빈칸으로 변경

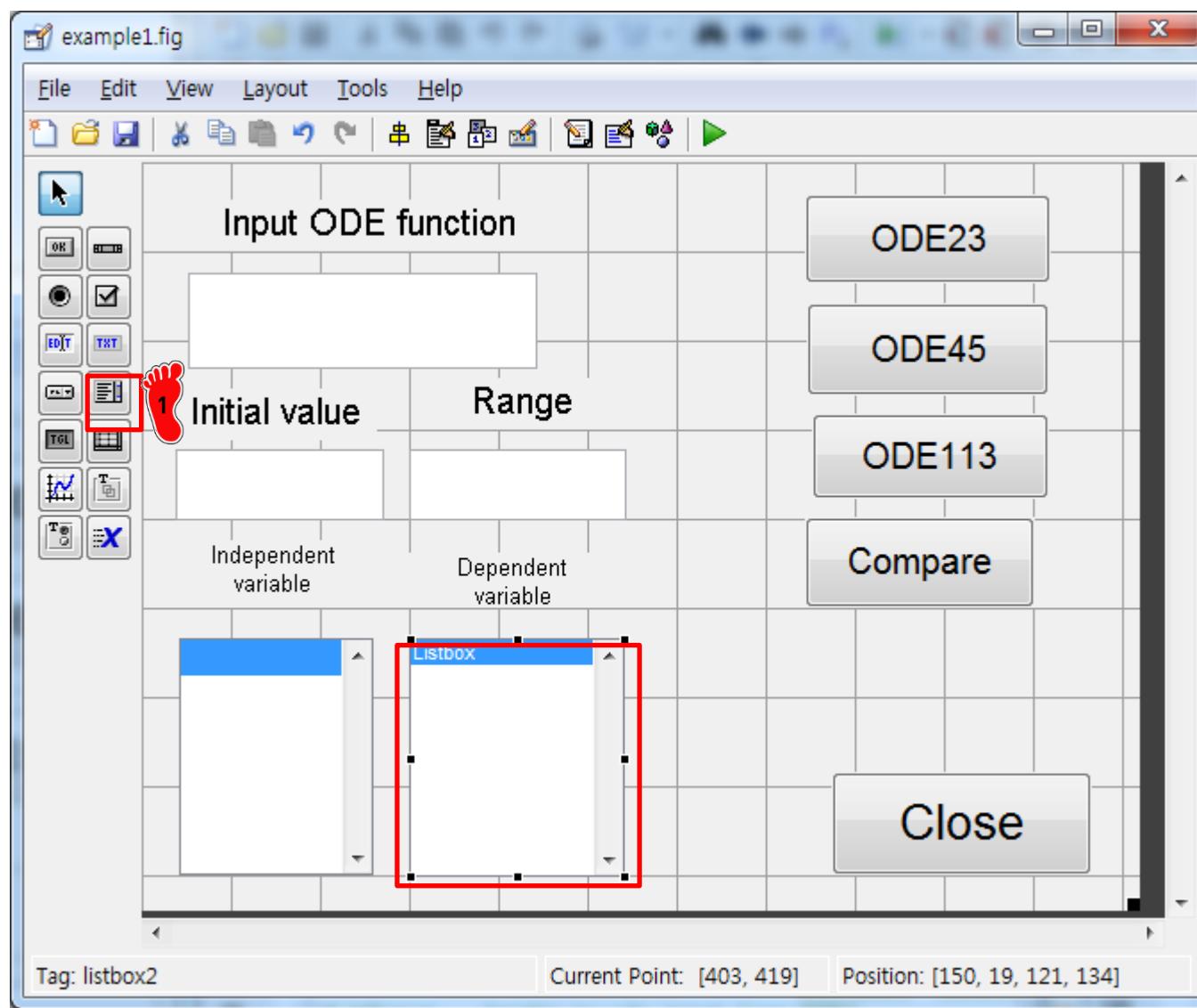


2 FontSize 12 로 변경



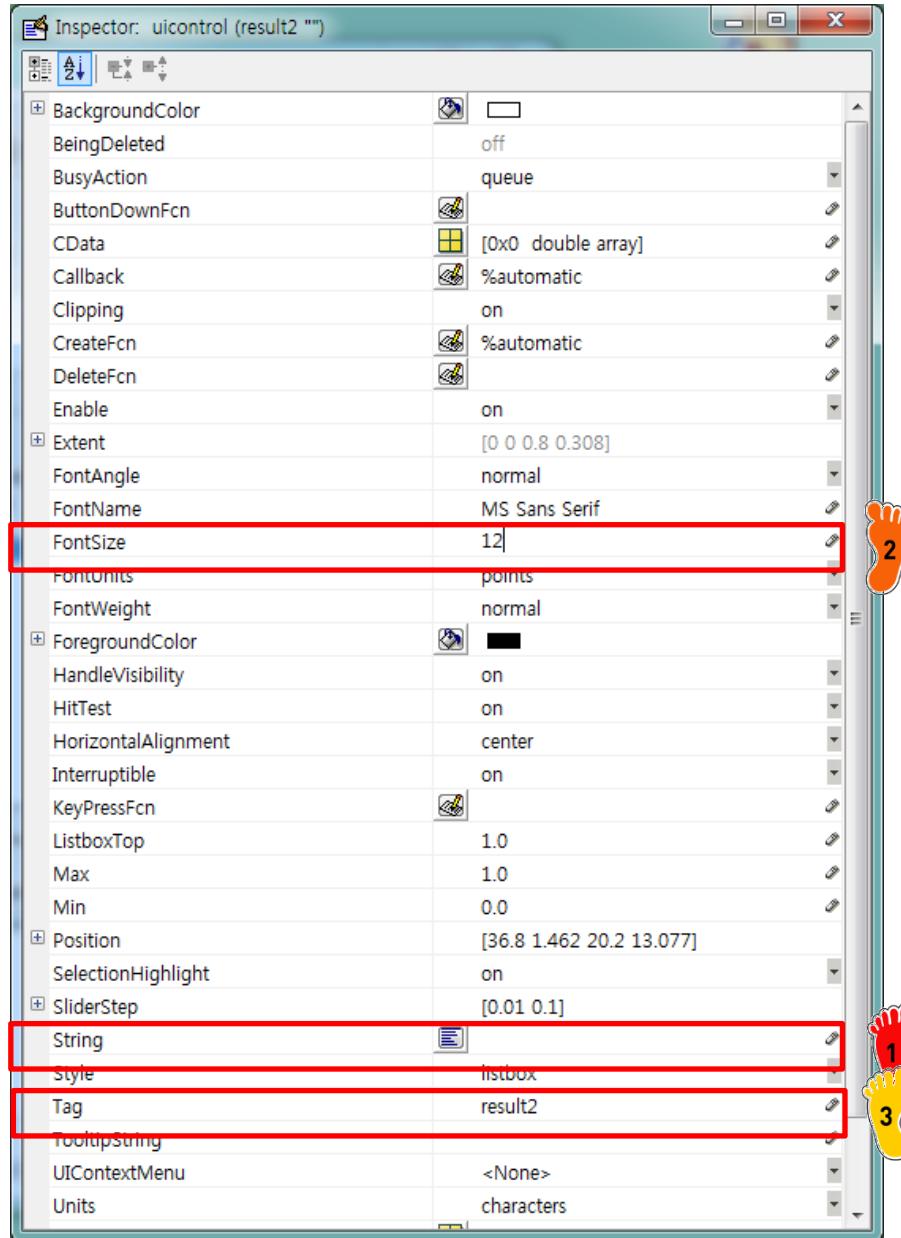
3 Tag 를 result1 으로 변경

LIST BOX



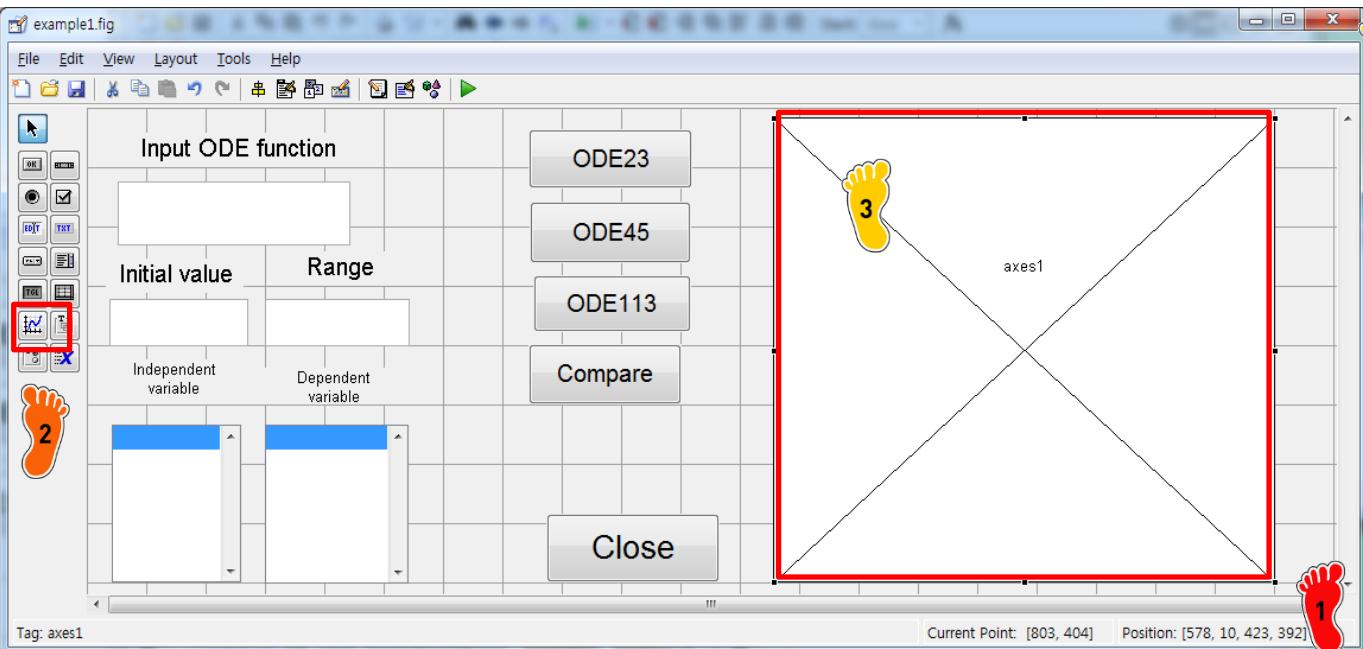
- 1 List box icon 클릭
- 2 List box 생성

LIST BOX: INSPECTOR



- 1 String 빈칸으로 변경
- 2 FontSize 12 로 변경
- 3 Tag 를 result2 로 변경

GRAPH WINDOW



- 1 GUI window 창을 마우스 드래그 하여 늘림
- 2 axes 아이콘 클릭
- 3 graph 를 표시해주는 창 생성

ODE23 PUSHBUTTON CODING

```

Editor - C:\Users\sean\Desktop\example1.m
File Edit Text Go Cell Tools Debug Desktop Window Help
1.0 + ÷ 1.1 × %% %% 1
153 % --- Executes on button press in ODE23_solve.
154 function ODE23_solve_Callback(hObject, eventdata, handles)
155 % hObject    handle to ODE23_solve (see GCBO)
156 % eventdata reserved - to be defined in a future version of MATLAB
157 % handles    structure with handles and user data (see GUIDATA)
158 fun = get(handles.ode_input,'string');
159 initial_temp = get(handles.initial_value_input,'string');
160 range_temp = get(handles.range_input,'string');
161 initial = str2num(initial_temp);
162 range = str2num(range_temp);
163
164 dydt = inline(fun,'t','y');
165 [t,y] = ode23(dydt,range,initial);
166
167 blank={};
168 set(handles.result1,'String',blank);
169 set(handles.result2,'String',blank);
170
171 ResultsStr1 = t;
172 ResultsStr2 = y;
173
174 set(handles.result1,'String',ResultsStr1);
175 set(handles.result2,'String',ResultsStr2);
176
177 plot(t,y)

```

The code in the MATLAB editor shows a callback function for a pushbutton. It reads 'fun', 'initial', and 'range' from edit boxes, converts them to numbers, calls the ode23 function, and then sets the results into two listboxes ('result1' and 'result2') and plots the solution.



ODE23_solve 태그로 이동

ode_input 태그로 지정된
edit box 입력 값을 get
함수로 호출



initial 과 range 값은 숫자
로 입력 해야 하기 때문에
str2num 명령어를 이용하
여 숫자로 변경 후 저장



호출한 함수를 inline 명령
어를 이용하여 함수로 지정



ode23 함수로 실행



result1 과 result2 의 태그
로 지정된 listbox 내용을
clear



결과 저장

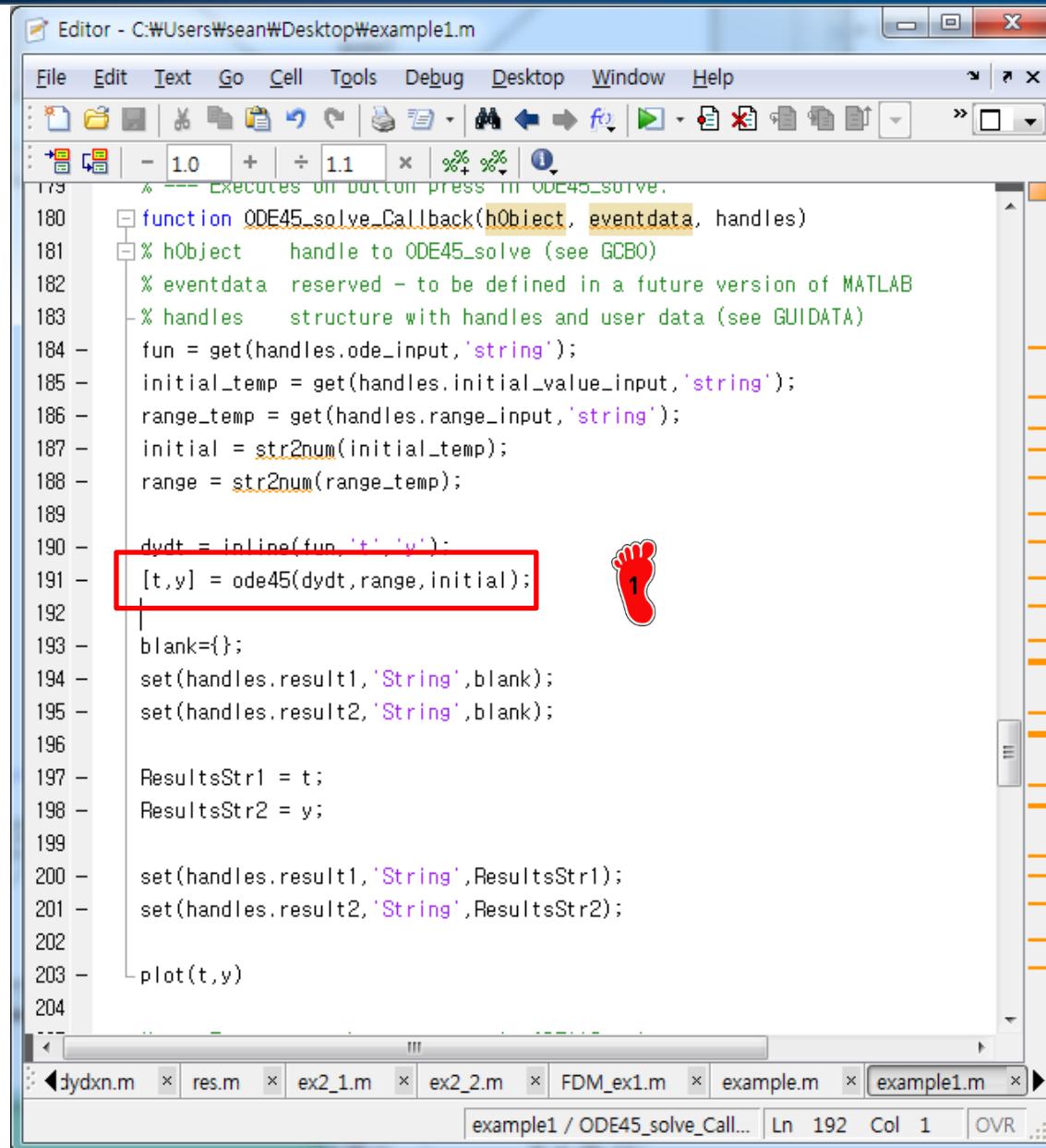


결과를 listbox에 출력



그래프 출력

ODE45&113 PUSHBUTTON CODING



The screenshot shows the MATLAB Editor window with the file `example1.m` open. The code is for a pushbutton callback function named `ODE45_solve_Callback`. The highlighted line of code is `[t,y] = ode45(dydt,range,initial);`. A red rectangular box surrounds this line, and a red footprint icon with the number '1' is placed next to it.

```

Editor - C:\Users\sean\Desktop\example1.m
File Edit Text Go Cell Tools Debug Desktop Window Help
1.0 + ÷ 11 × % % %
179 % ---- Executes on button press in ODE45_SOLVE.
180 function ODE45_solve_Callback(hObject, eventdata, handles)
181 % hObject    handle to ODE45_solve (see GCBO)
182 % eventdata reserved - to be defined in a future version of MATLAB
183 % handles    structure with handles and user data (see GUIDATA)
184 fun = get(handles.ode_input,'string');
185 initial_temp = get(handles.initial_value_input,'string');
186 range_temp = get(handles.range_input,'string');
187 initial = str2num(initial_temp);
188 range = str2num(range_temp);
189
190 dydt = inline(fun,'t','y');
191 [t,y] = ode45(dydt,range,initial); 1
192 blank={};
193 set(handles.result1,'String',blank);
194 set(handles.result2,'String',blank);
195
196 ResultsStr1 = t;
197 ResultsStr2 = y;
198
199 set(handles.result1,'String',ResultsStr1);
200 set(handles.result2,'String',ResultsStr2);
201
202 plot(t,y)
203

```



ode함수 명령어만 변경 후
나머지는 동일한 코드로 입
력

ODE45&113 PUSHBUTTON CODING

- 
 - 1 세 가지 ode 함수를 비교하기 위해 결과를 따로 저장
 - 2 listbox 내용 출력코드는 삭제
 - 3 세 가지 결과를 동시에 plot

RESULT



함수 및 초기값과 range 입력 후 실행 버튼을 클릭하여 결과 확인

$$\frac{dy}{dx} = f(x, y) = 4e^{0.8x} - 0.5y$$

with $y(0) = 2$ from $x = 0 \sim 4$

