Российский государственный университет нефти и газа им. И.М. Губкина

Кафедра Информатики

Дисциплина: Программные комплексы общего назначения

Преподаватель:

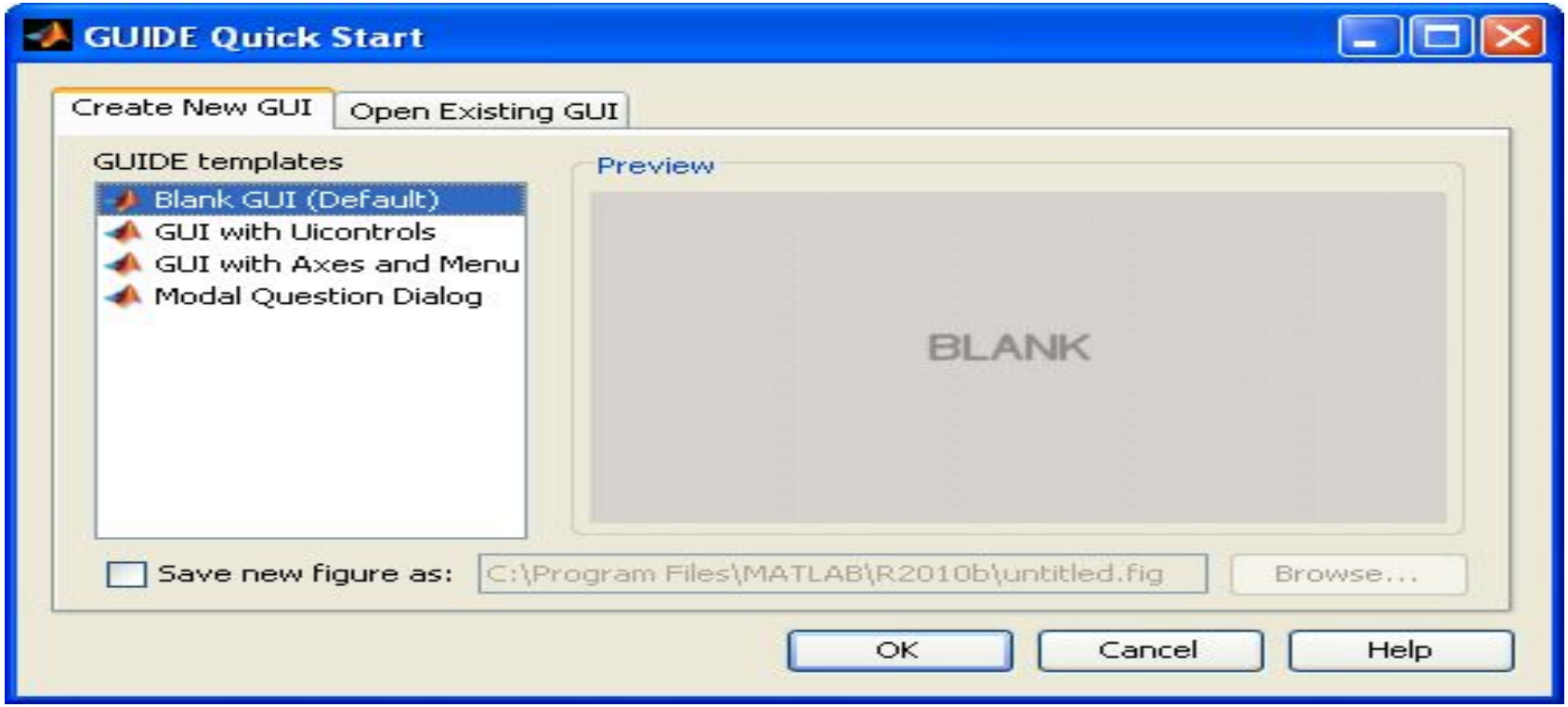
к.т.н., доцент
Коротаев
Александр Фёдорович

Формирование пользовательского интерфейса

Будем использовать утилиту **GUIDE** из меню

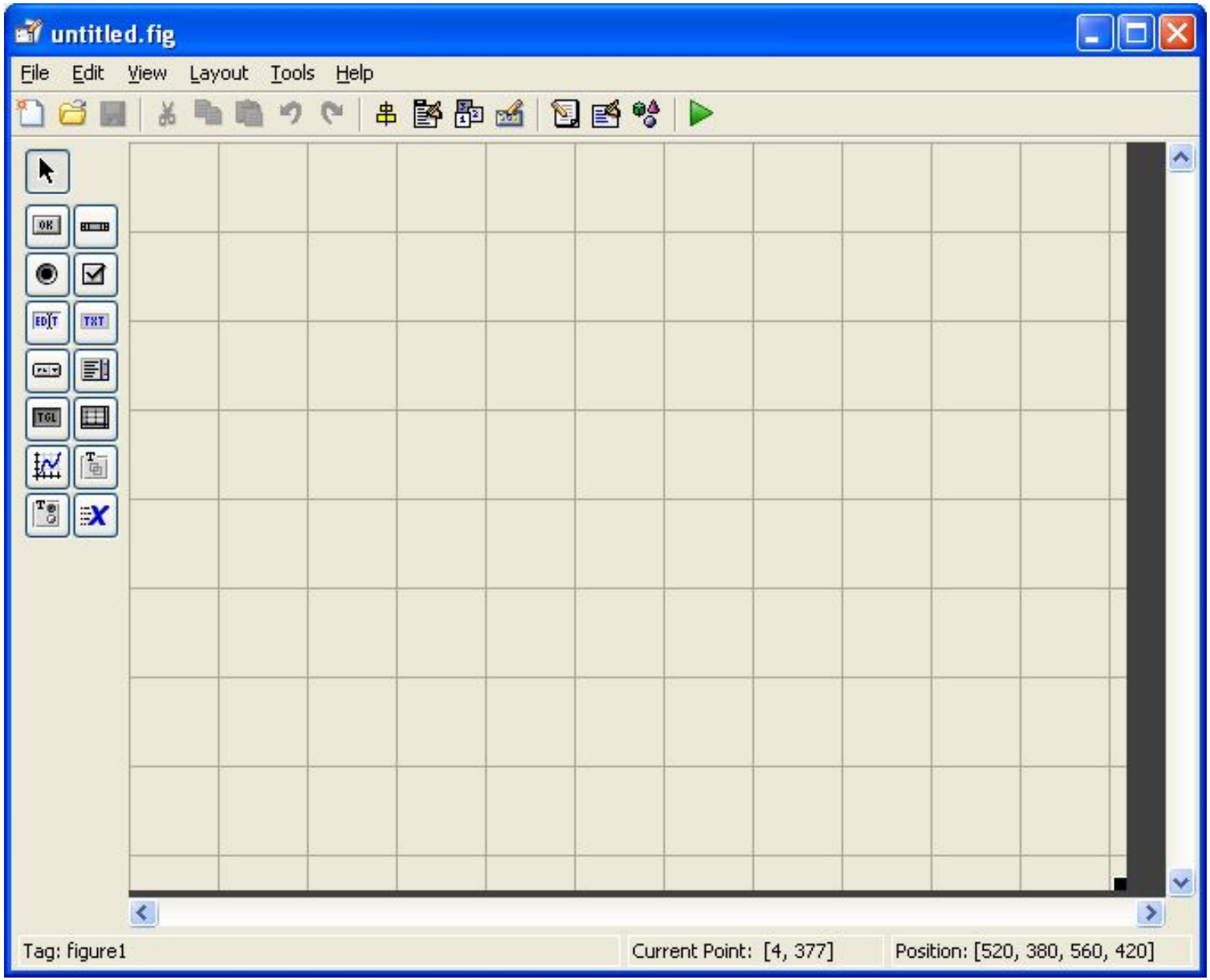
File->New->GUI

или из панели инструментов

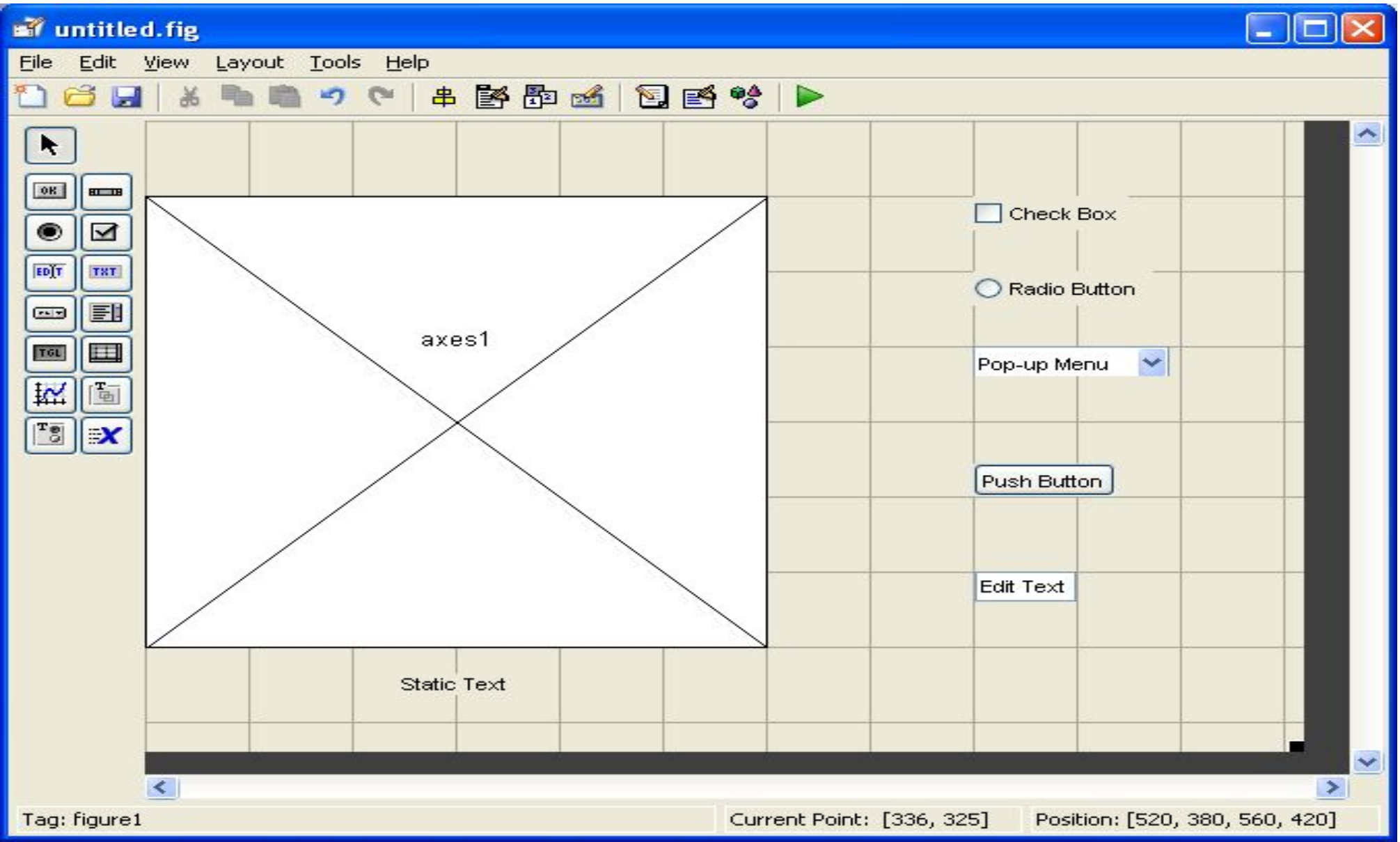


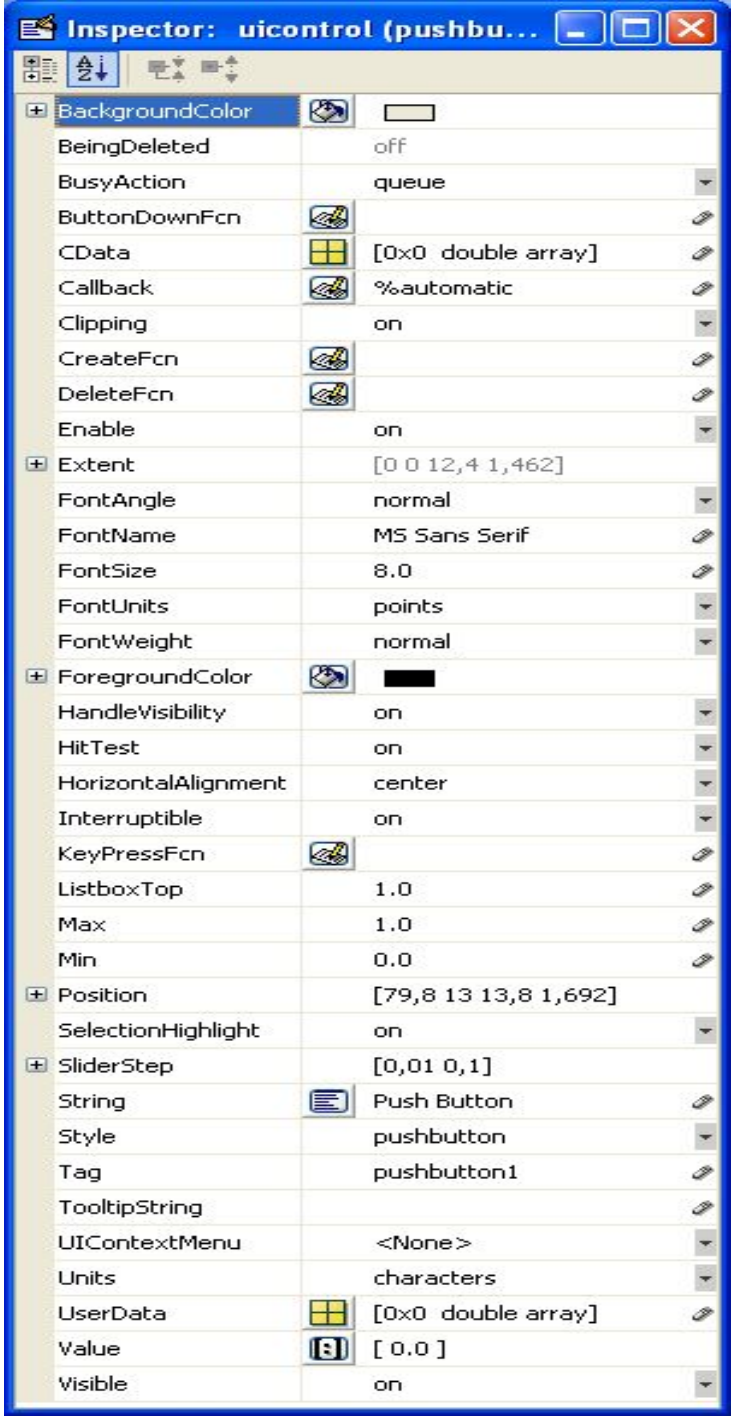
Здесь можно открыть существующую заготовку GUI-приложения или создать новую , выбрав

Появится окно редактора, содержащее саму заготовку и **палитру графических элементов управления**



С помощью мыши можно перетаскивать необходимые элементы на создаваемое графическое окно





Когда элемент управления размещен на поверхности графического окна, двойной щелчок по нему мышью вызовет редактор свойств этого элемента **Property Inspector**, что и показано для кнопки **Push Button**.

Здесь расположен список всех свойств данного графического элемента. Выбрав конкретное свойство, можно задать его значение.

Сохранить текущее состояние работы можно выполнив команду меню **File->Save As…** ->>

Задаём имя файла и место его размещения. Кроме графического файла с выбранным именем и расширением **.fig** создастся ещё и текстовый **m**-файл с тем же именем, который будет содержать программный код на **М-языке**.

При необходимости внесения изменений можно из основного окна **МАТЛАБ** вызвать

File->New->GUI

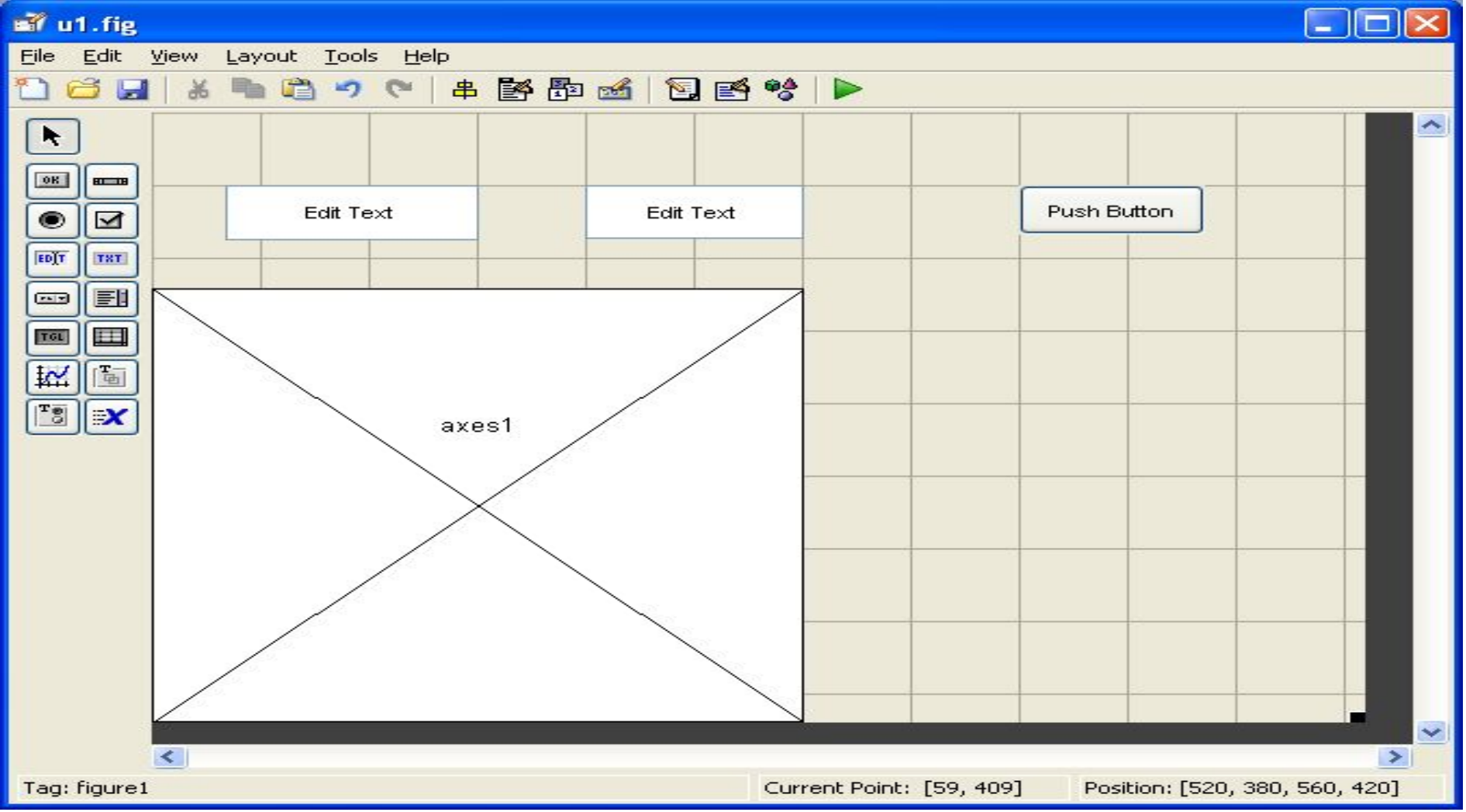
и на вкладке **Open Existing GUI** выбрать нужный файл.

Задача

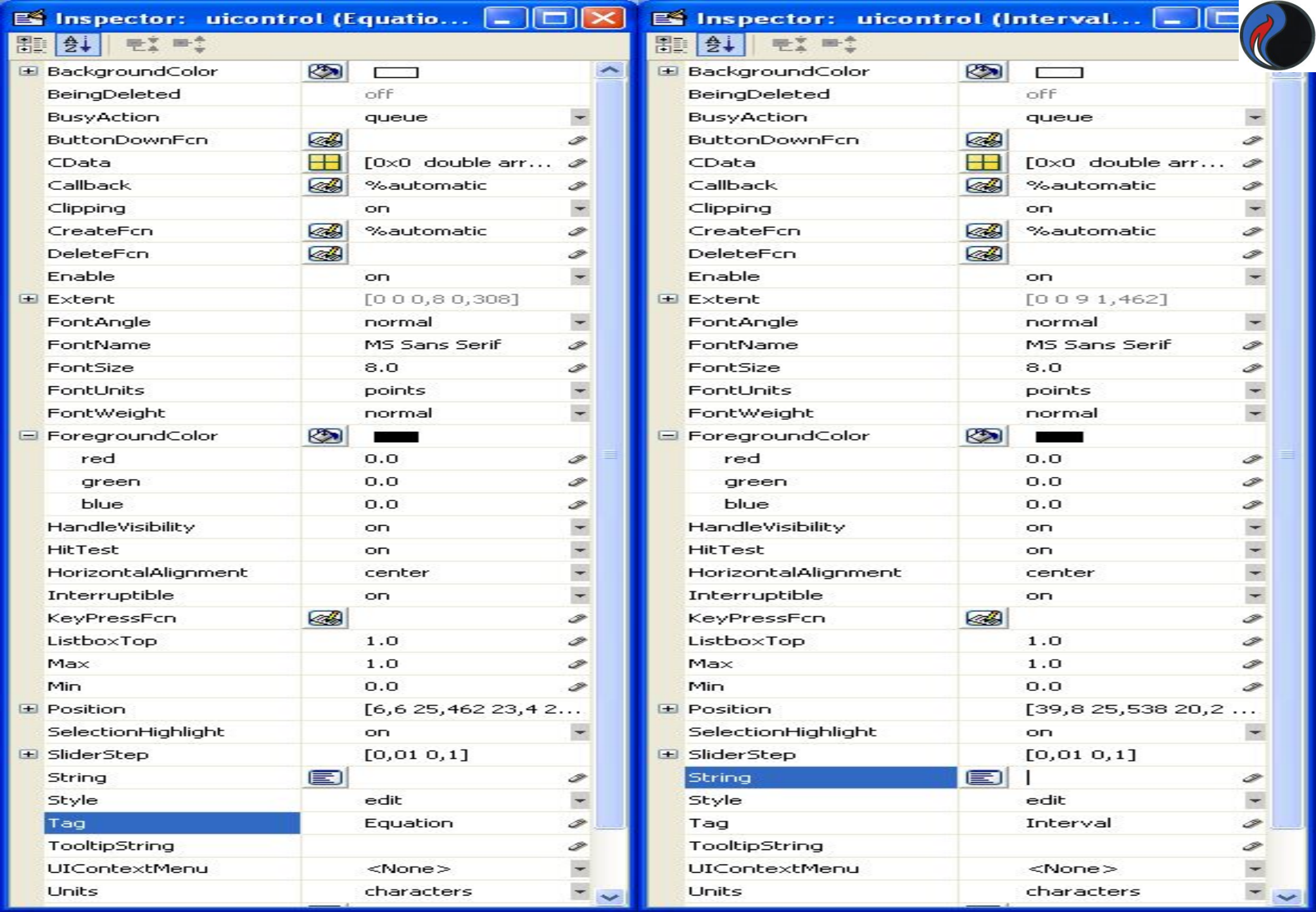
Построить графический интерфейс, в котором с клавиатуры вводится:

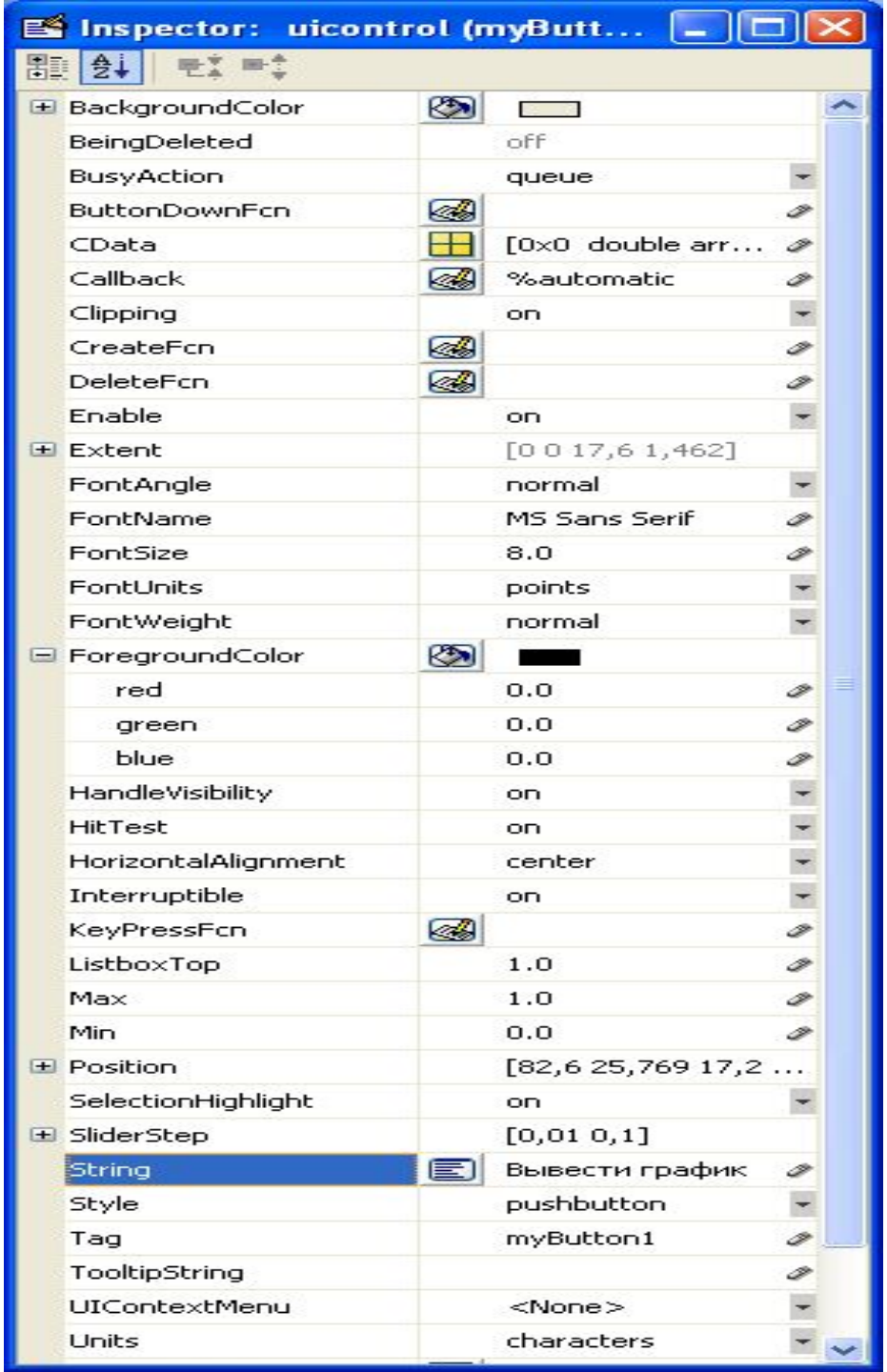
- **формула функции** одной переменной
- **границы интервала** построения графика функции

После нажатия на кнопку «**Вывести график**» изображался бы график этой функции



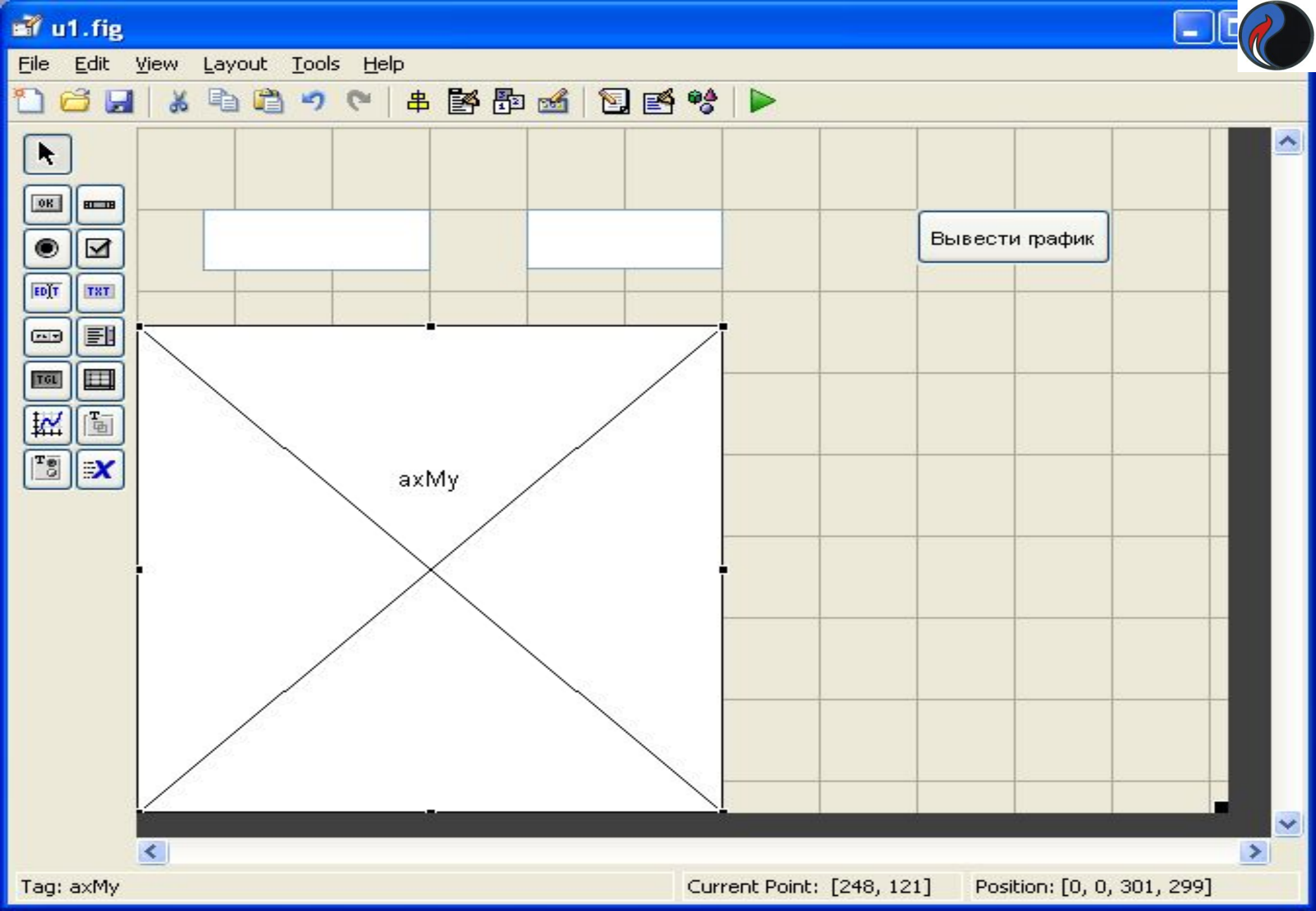
Для первой области ввода текста (куда будем вводить имя функции) элементу **Tag** присвоим имя **Equation**, для второй (для ввода интервала) - **Interval .** Уберем строку “**Edit Text**” в поле **String** (в обоих областях ввода текста). Элементу **Tag** для области вывода графика присвоим имя **axMy**.


Inspector: uicontrol (Equatio...
BackgroundColor
BeingDeleted off
BusyAction queue
ButtonDownFcn
CData [0x0 double arr...
Callback %automatic
Clipping on
CreateFcn %automatic
DeleteFcn
Enable on
Extent [0 0 0,8 0,308]
FontAngle normal
FontName MS Sans Serif
FontSize 8.0
FontUnits points
FontWeight normal
ForegroundColor
red 0.0
green 0.0
blue 0.0
HandleVisibility on
HitTest on
HorizontalAlignment center
Interruptible on
KeyPressFcn
ListboxTop 1.0
Max 1.0
Min 0.0
Position [6,6 25,462 23,4 2...
SelectionHighlight on
SliderStep [0,01 0,1]
String
Style edit
Tag Equation
TooltipString
UIContextMenu <None>
Units characters
Inspector: uicontrol (Interval...
BackgroundColor
BeingDeleted off
BusyAction queue
ButtonDownFcn
CData [0x0 double arr...
Callback %automatic
Clipping on
CreateFcn %automatic
DeleteFcn
Enable on
Extent [0 0 9 1,462]
FontAngle normal
FontName MS Sans Serif
FontSize 8.0
FontUnits points
FontWeight normal
ForegroundColor
red 0.0
green 0.0
blue 0.0
HandleVisibility on
HitTest on
HorizontalAlignment center
Interruptible on
KeyPressFcn
ListboxTop 1.0
Max 1.0
Min 0.0
Position [39,8 25,538 20,2 ...
SelectionHighlight on
SliderStep [0,01 0,1]
String
Style edit
Tag Interval
TooltipString
UIContextMenu <None>
Units characters



Разместим необходимые надписи на кнопке

Кликнув по ней дважды , получим на экране список свойств. Заменим в строке **String** значение по умолчанию “**Push Button**”на “**Вывести график**”. Зададим значение свойства **Tag**. Назовем его **myButton1**. Сохраним построенный интерфейс в файле. Получим на экране:


u1.fig
File
Edit
View
Layout
Tools
Help
Вывести график
axMy
Tag: axMy
Current Point: [248, 121]
Position: [0, 0, 301, 299]

Запустим приложение командой **Run**.

После запуска приложения проверим его функциональность.

Введем в первое текстовое поле (ввод функции) - **sin(x)** ,

во второе поле (для интервала) – **-pi pi**

Нажмем на кнопку **Вывести график**.

На первый взгляд, все элементы функционируют, вот только график не строится

Мы уже знаем, что для построения графика **sin(x)** достаточно задать вектор **x=-pi:0.1:pi**,
а затем команду построения графиков **plot(x,sin(x))**

Другой вариант: использовать функцию **fplot**
(не требуется вычислять вектор **x**)

```
f=inline('sin(x)'); fplot(f,[-pi pi])
```

График будет построен в окне **Figure 1**

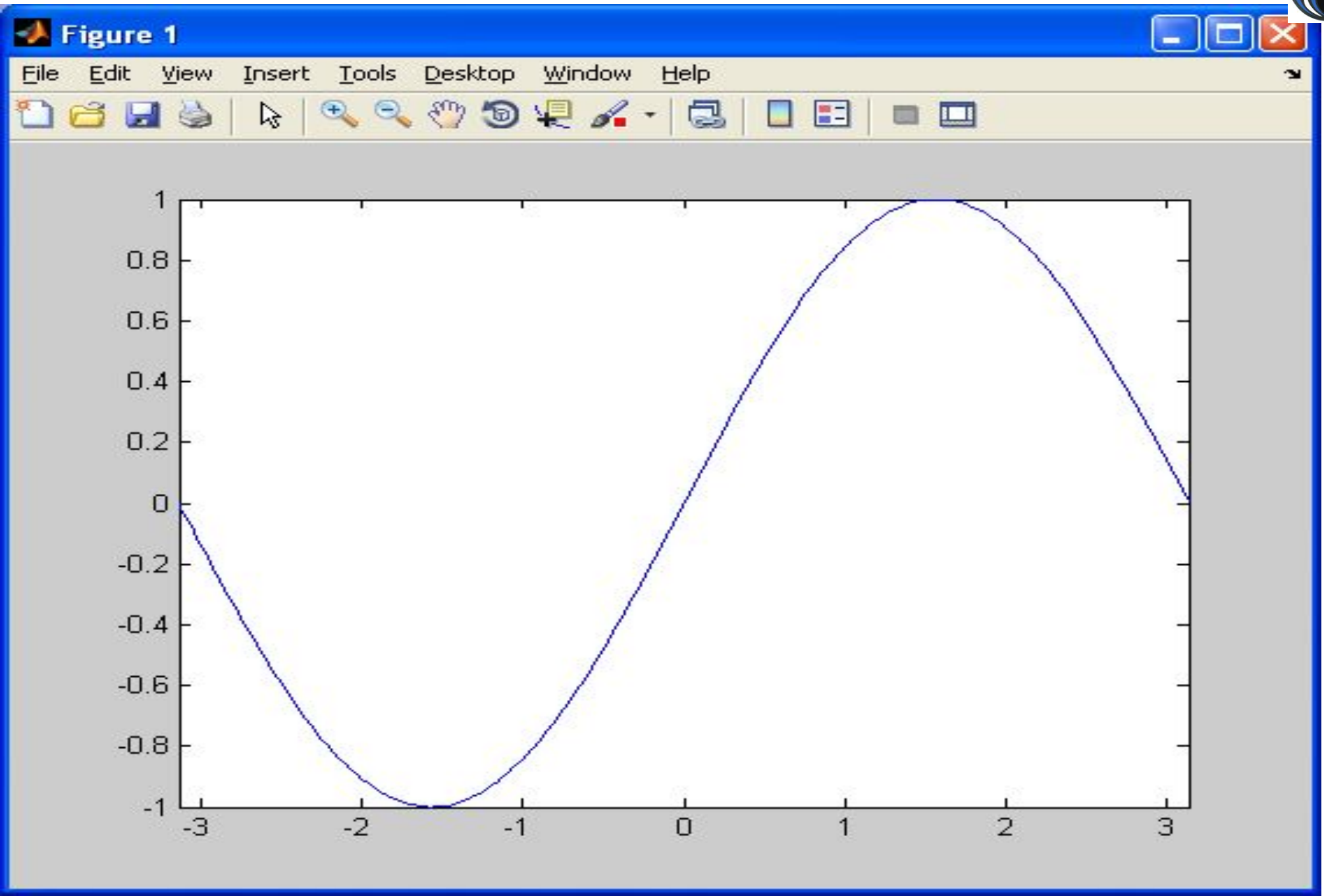


Figure 1
File Edit View Insert Tools Desktop Window Help
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-3 -2 -1 0 1 2 3

Для **построения графика в интерфейсе** необходимо, чтобы при нажатии кнопки **Вывести график** выполнялась некоторая последовательность команд.

Для этого вызовем на редактирование m-файл.

Найдем в нём строки:

```
function myButton1_Callback(hObject, eventdata, handles)
% hObject    handle to myButton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in radiobutton1.
```

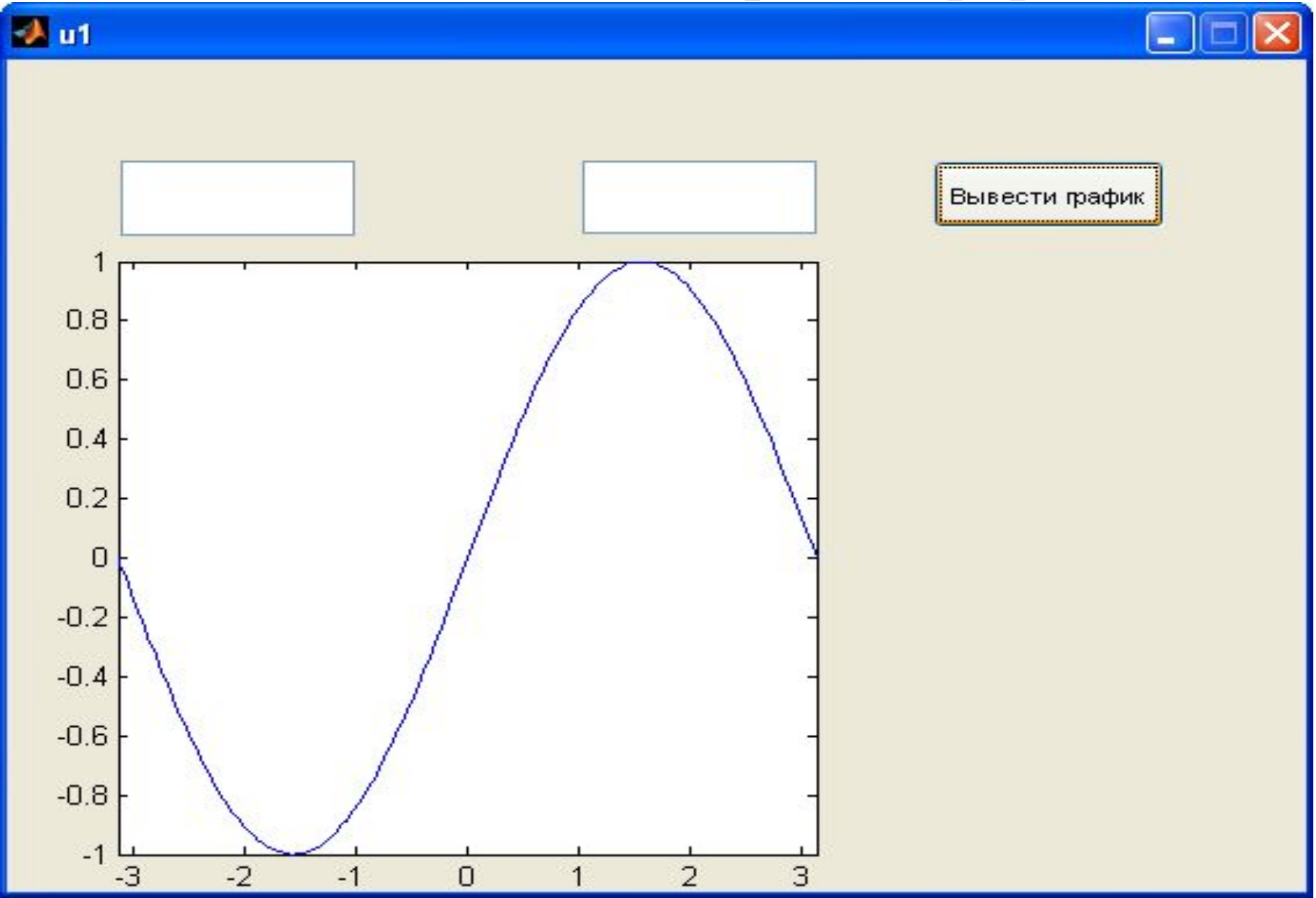
Название функции **myButton1_Callback** говорит о том, что функция обслуживает событие **Callback** элемента **myButton1**

Вставим в неё функцию построения графика

```
f=inline('sin(x)'); fplot(f,[-pi pi])
```

В результате при нажатии кнопки **Вывести график** нарисуется график **sin(x)** в интервале **[-pi pi]**, независимо от заданного в текстовых полях

Только sin(x) в интервале [-pi pi]



Как изменить функцию **myButton1_Callback**, чтобы строить график функции **sin(x)** на интервале, введённом в соответствующее поле? Вставим в m-файл команду

interval=str2num(get(handles.Interval,'String'));

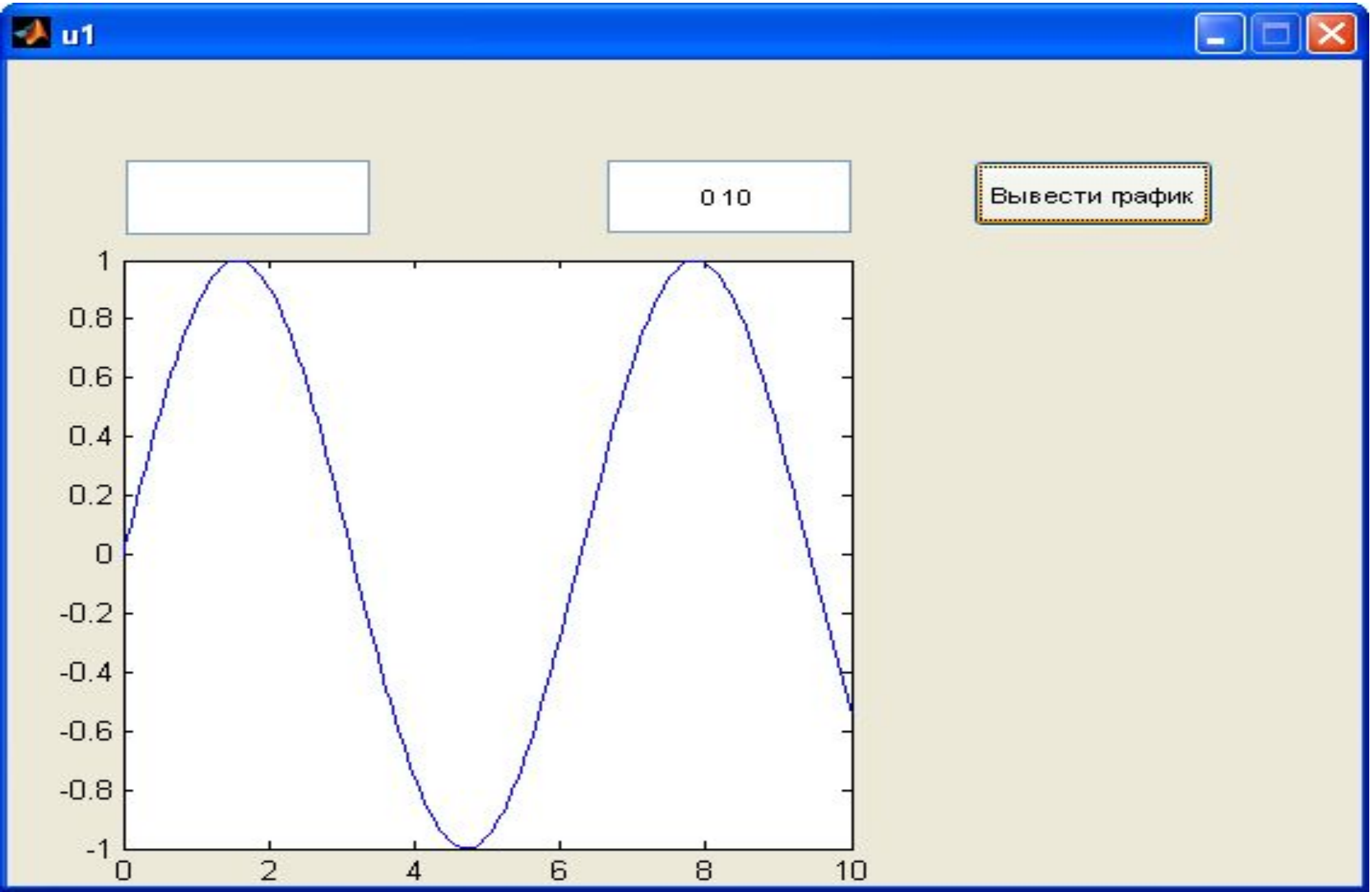
и поменяем второй параметр в функции **fplot(f,[-pi pi])** на

fplot(f,interval);

Получим функцию, рисующую график **sin** на заданном интервале

```
function myButton1_Callback(hObject, eventdata, handles)
% hObject    handle to myButton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
   MATLAB
% handles    structure with handles and user data (see
   GUIDATA)
% --- Executes on button press in radiobutton1.
 f=inline('sin(x)');
 interval=str2num(get(handles.Interval,'String'));
 fplot(f,interval);
```

Только sin(x) , но в любом интервале



Чтобы рисовался график любой вводимой функции,

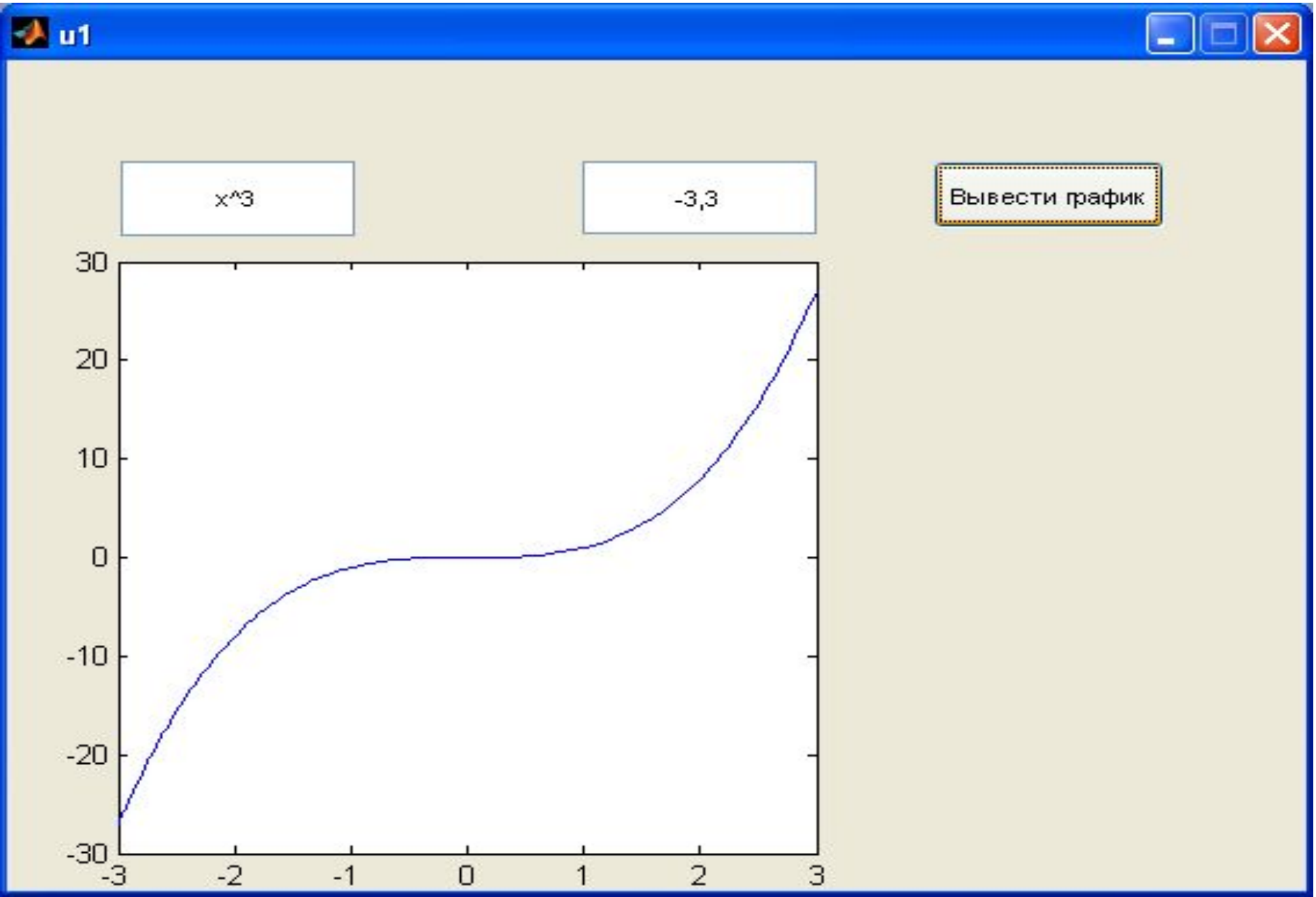
f=inline('sin(x)') необходимо заменить на

```
f=inline(get(handles.Equation,'String'));
```

Получим функцию

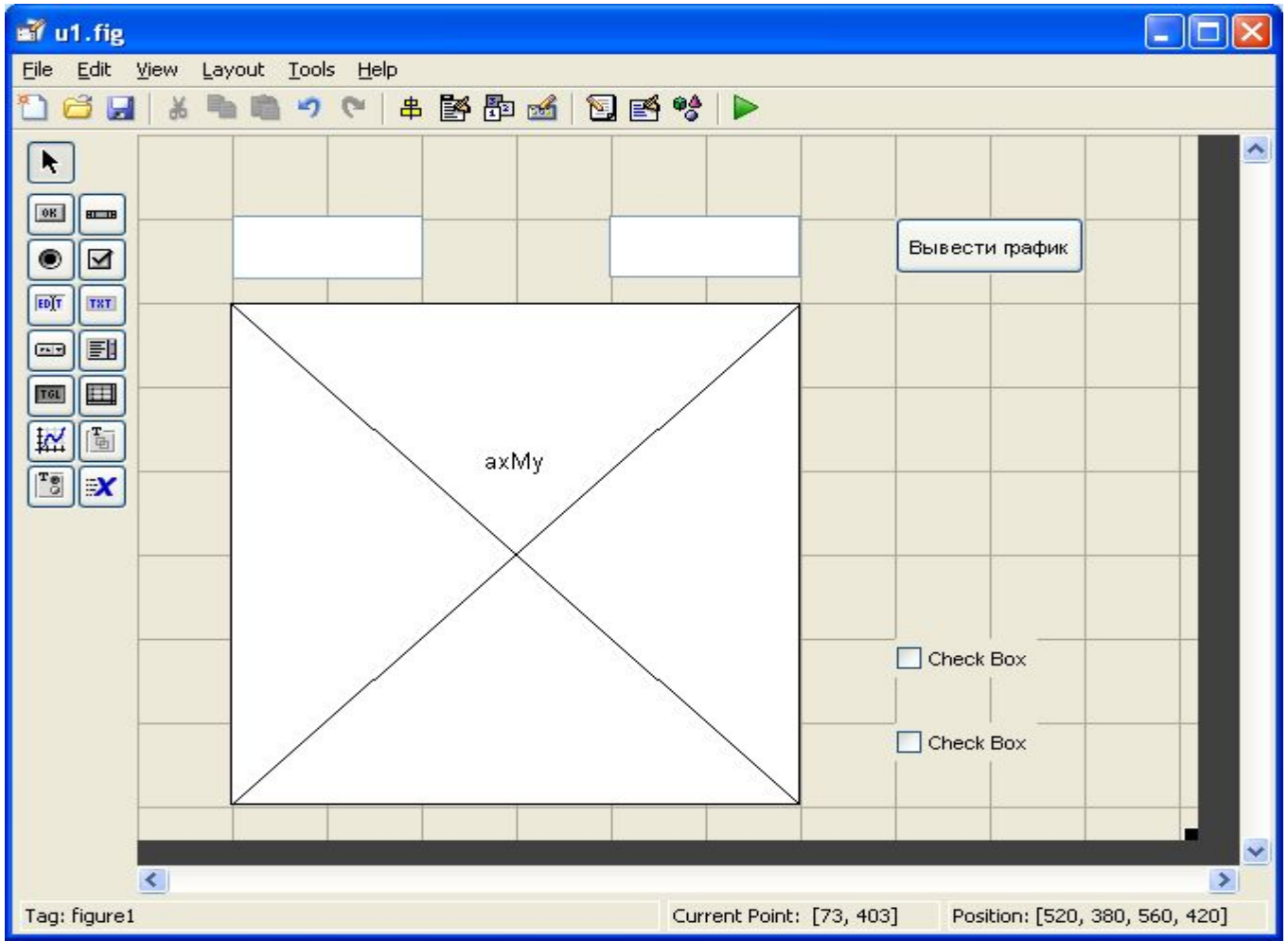
```
function myButton1_Callback(hObject, eventdata, handles)
% hObject    handle to myButton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
   MATLAB
% handles    structure with handles and user data (see
   GUIDATA)
% --- Executes on button press in radiobutton1.
 f=inline(get(handles.Equation,'String'));
 interval=str2num(get(handles.Interval,'String'));
 fplot(f,interval);
```

Любая функция в любом интервале



Для того, чтобы включать и выключать сетку осей на графике установим в нашем интерфейсе 2 флажка (для оси X и оси Y) .

Воспользуемся для этого элементом «**Check Box**»



Зададим в “**Инспекторе свойств**” этих элементов

- для оси **X : Tag – cbX , String – Сетка по X**
- для оси **Y : Tag – cbY , String – Сетка по Y**

Внесем изменения в функции **Callback m-файла** соответствующих элементов

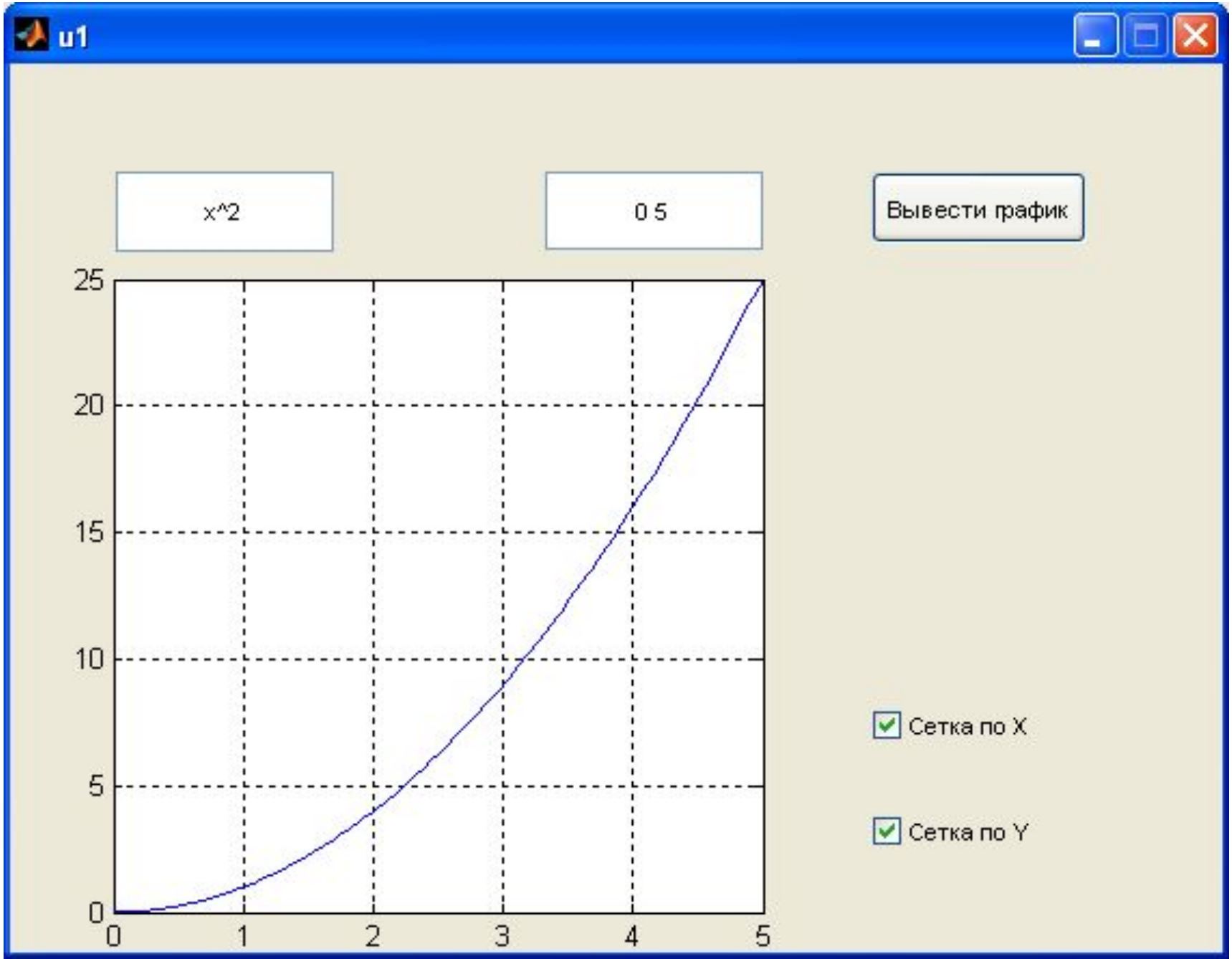
```
% --- Executes on button press in cbX.
function cbX_Callback(hObject, eventdata, handles)
% hObject    handle to cbX (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(hObject,'Value')
   set(gca,'XGrid','on')
else
   set(gca,'XGrid','off')
end
 % Hint: get(hObject,'Value') returns toggle state of cbX
```

Аналогично для функции **cbY_Callback**

```
% --- Executes on button press in cbY.
function cbY_Callback(hObject, eventdata, handles)
% hObject    handle to cbY (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(hObject,'Value')
set(gca,'YGrid','on')
else
   set(gca,'YGrid','off')
end

% Hint: get(hObject,'Value') returns toggle state of cbY
```

График с сеткой



Создадим в нашем интерфейсе кнопку,
при щелчке по которой будет найден минимум функции;
пусть этот минимум будет отмечен на графике красной точкой.

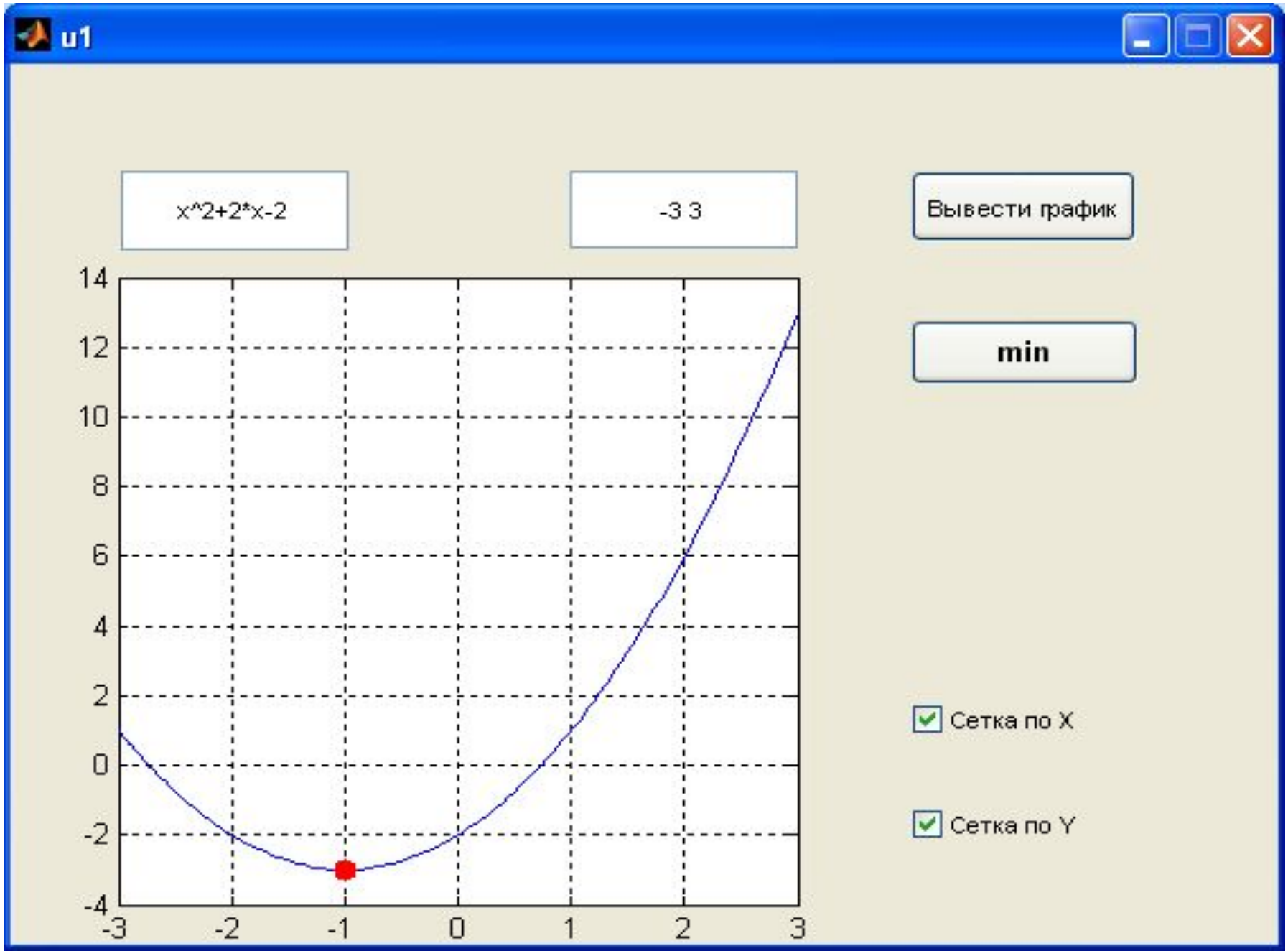
Зададим в “**Инспекторе свойств**” этой кнопки

Tag – pMinimum , String – min

Внесем изменения в функцию **pMinimum_Callback** :

```
interval=str2num(get(handles.Interval,'String'));
x1=interval(1);
x2=interval(2);
f=inline(get(handles.Equation,'String'));
[x,y]=fminbnd(f,x1,x2);
hold on;
plot(x,y,'r.','MarkerSize',25);
hold off;
```

График с точкой минимума



Для того, чтобы вывести в наш интерфейс координаты минимума в численном виде, создадим в нём элемент типа **Static Text**. Зададим в “**Инспекторе свойств**” его **Tag** – **min**

Добавим в функцию **pMinimum_Callback** ещё одну строку:

```
interval=str2num(get(handles.Interval,'String'));
x1=interval(1);
x2=interval(2);
f=inline(get(handles.Equation,'String'));
[x,y]=fminbnd(f,x1,x2);
hold on;
plot(x,y,'r.','MarkerSize',25);
hold off;
set(handles.min,'String',['x=',num2str(x),'   y=',num2str(y)])
```

Замечание: **num2str** - переводит числа в символы

str2num - переводит строку символов в числа

После добавления в окно интерфейса кнопок для построения графиков 1-й и 2-й производных, очистки графика (**функция cla**), нахождения нуля функции
Оно приобретёт примерно такой вид: