

## Практическая работа 10. Проектирование графического интерфейса пользователя в среде GUIDE

**Цель работы:** изучение основных команд, используемых для проектирования графического интерфейса пользователя, приобретение навыков построения графического интерфейса пользователя в среде GUIDE.

### Теоретические сведения

#### Проектирование интерфейса в среде GUIDE

Как и любой процесс проектирования, процесс построения графического интерфейса пользователя можно разбить на следующие этапы: постановка задачи, создание формы интерфейса и создание на нее элементов управления, написание кода программы и кода обработки событий.

На первом этапе проводится анализ поставленной задачи и определяется количество и состав элементов управления.

На втором этапе создается форма графического интерфейса, на ней создаются и размещаются элементы управления. Здесь же описываются их свойства.

На третьем этапе создания графического интерфейса пользователя, пишется код основной программы вычисления и код для обработки событий. Код основной программы вычисления пишется на языке программирования операционной среды MatLab, в виде М-файлов. Созданные М-файлы закрепляются за событием какого-нибудь элемента управления или формы.

#### Вызов редактора GUIDE

Редактор GUIDE запускается из командного окна с помощью команды *guide*. После запуска редактора появляется окно, содержащее две вкладки (рисунок 1). Вкладка *<create new GUI>* дает возможность начать проектирование нового интерфейса. Вкладкой *<open existing GUI>* следует пользоваться в том случае, если интерфейс ранее был уже создан.

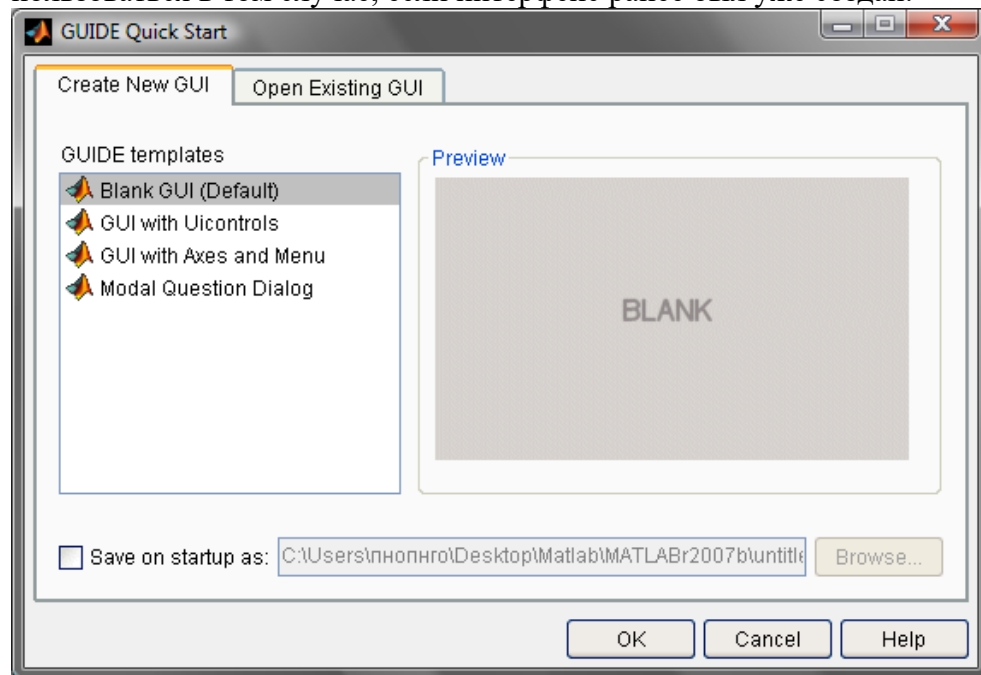


Рис. 1. Стартовое окно

Список шаблонов (*GUIDE template*) предусматривает следующие варианты компоновки, которые могут находиться в основе разрабатываемого приложения:

- 1) *Blank GUI* (Шаблон интерфейса) – используется в случае, когда приложение разрабатывается с нуля.
- 2) *GUI with Uicontrols* (Типовой интерфейс) – этот выбор предусматривает использование готового приложения в качестве прототипа.
- 3) *GUI with Axes and Menu* (Интерфейс с координатными осями и меню) – этот вариант предполагает использование приложения, которое воспроизводит график одной из шести функций. Эти функции могут быть выбраны из раскрывающегося списка.
- 4) *Modal Question Dialog* (Модальное диалоговое окно) – в этом случае в качестве прототипа можно использовать приложение, содержащее диалоговое окно с двумя вариантами ответа.










#### *Управление конструктором графического интерфейса*

На рабочем поле конструктора нанесена вспомогательная сетка, позволяющая осуществлять выравнивание элементов графического интерфейса по узлам и линиям.

В разделе *Tools* (Сервис) находятся команды, предусматривающие вызов инструментов построения графического интерфейса, например, *Align Objects*, *Grid and Rules*, *Menu Editor*, *Tab Order Options*.

В таблице 2 перечислены некоторые элементы управления, используемые при построении графического интерфейса приложения.

Таблица 2. Элементы управления

Элемент	Назначение
	Режим выделения объекта
	Режим создания кнопки
	Режим создания переключателя
	Режим создания флажка
	Режим создания поля для ввода текстовой информации
	Режим создания поля для ввода текстовой информации без возможности редактирования
	Режим создания списка
	Режим создания раскрывающегося списка
	Режим создания поля для вывода информации в виде графиков и диаграмм

#### *Инспектор свойств (Property Inspector)*

Инспектор свойств вызывается по команде *View/Property Inspector*. Список свойств выделенного объекта представлен в левой половине окна, а в правой половине окна находятся их значения.

Инспектор свойств на стадии проектирования приложения в среде GUIDE выполняет те же действия, которые осуществляют функции *get* и *set* при выполнении работы приложения. При необходимости текущие значения свойств объектов заменяются новыми значениями.

#### *Просмотр объектов (Object Browser)*

Вызов браузера объектов, расположенных на форме, осуществляется командой *View/Object Browser*. Все объекты отображаются в порядке их размещения на форме. Ветви дерева объектов подчиняются взаимоотношениям между объектами-родителями и объектами-потомками.

### Создание меню (Menu Editor)

В нижней части редактора меню находятся две вкладки, которые позволяют создавать либо главное меню, либо всплывающие меню.

Для создания нового раздела меню используется кнопка *New Menu*. А для создания новой команды меню – кнопка *New Menu Item*.

В качестве примера построим графический интерфейс для вывода графика функции в заданном интервале.

Для выбора и размещения объектов будем использовать панель инструментов, изображенную на рисунке 2. На рисунке 3 показан инспектор свойств, позволяющий осуществлять редактирование значения свойств выделенного объекта.



Рис. 2. – Панель элементов управления

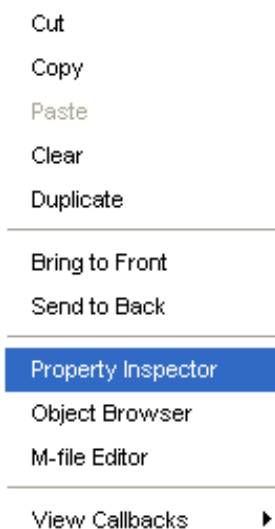



Рис. 3. – Инспектор свойств

Инспектор свойств может быть вызван также с помощью кнопки . Создадим две области. Одна область предназначена для ввода выражения, а другая область – для ввода границ интервала вычисления выражения.

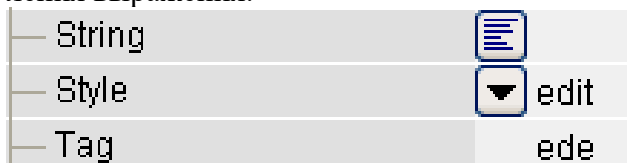



Рис. 4. Свойства Tag и String

Для первой области (области ввода функции) свойству *Tag* присвоим значение *ede*, а для второй области (области ввода заданного интервала) значение *edi*. Для осей графика присвоим свойству *Tag* значение *ay*. Значения свойства *String* необходимо очистить (рисунок 4).

Создаем кнопку с помощью элемента управления кнопка - . Список свойств этого элемента появляется при двойном нажатии кнопке, созданной в выбранной области графического интерфейса. Свойству *Tag* присвоим значение *Bp*, а свойству *String* – значение *Plot*. Для построения графика необходимо выполнение ряда команд. При нажатии правой кнопки мыши в контекстном меню выбираем команду *Callback* (рисунок 5).

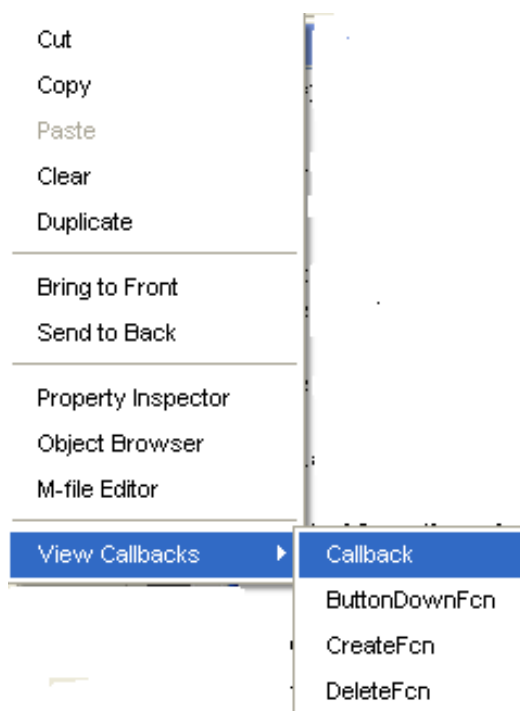



Рис. 5. Свойство Callback

Открывается содержимое М-файла, где надо найти свойство *Callback*, возникающее при нажатии кнопки *Вр*.

Укажем следующую последовательность команд:

```
cla
interval=str2num(get(handles.edi,'String'));
f=inline(get(handles.ede,'String'));
[x,y]=fplot(f,interval);
handles.line=plot(x,y,'b-');
handles.line=plot(x,y,'Color', My_color);
guidata(gcbo,handles);
hold on
```

В данном случае функция *get()* использует конкретное свойство заданного объекта. Функция *str2num()* выполняет преобразование строкового значения в числовое значение. Полученное значение используется переменной *interval*. Функция *inline()* преобразует строку, содержащее выражение, в функцию. Функция *fplot(f,interval)* осуществляет построение графика функции *f* на заданном интервале. Запуск программы осуществляется с помощью кнопки .

### Последовательность действий для построения графического интерфейса в среде GUIDE

**Задание:** Получить график аналитического выражения  $30\sin(x)/x$  в заданном интервале  $[-30 \ 30]$ . Осуществить ввод аналитического выражения, ввод значений границ интервала, построить график.

1. Создаем новый файл для построения графического интерфейса в среде GUIDE (рисунок 6).

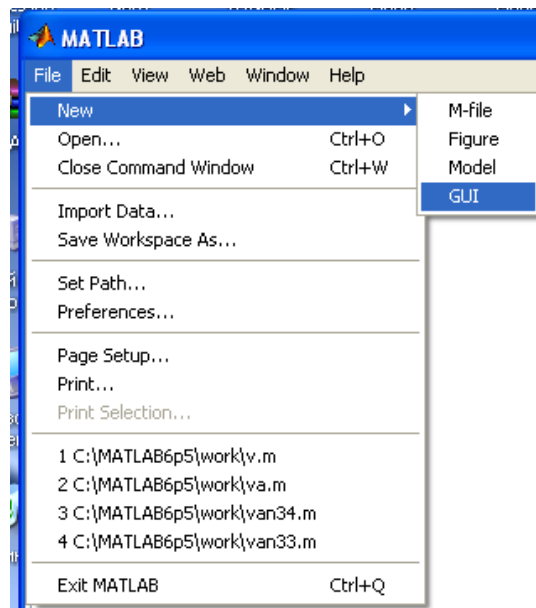


Рис. 6. Создание файла

2. Стартовое окно (рисунок 7).

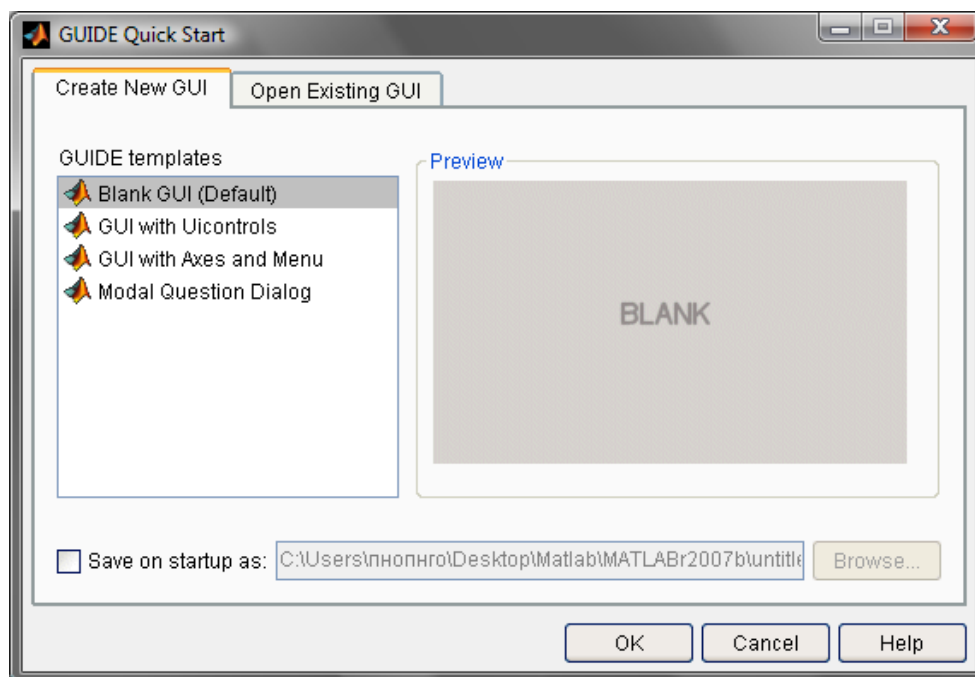


Рис. 7. Стартовое окно

3. Начальный экран (рисунок 8).

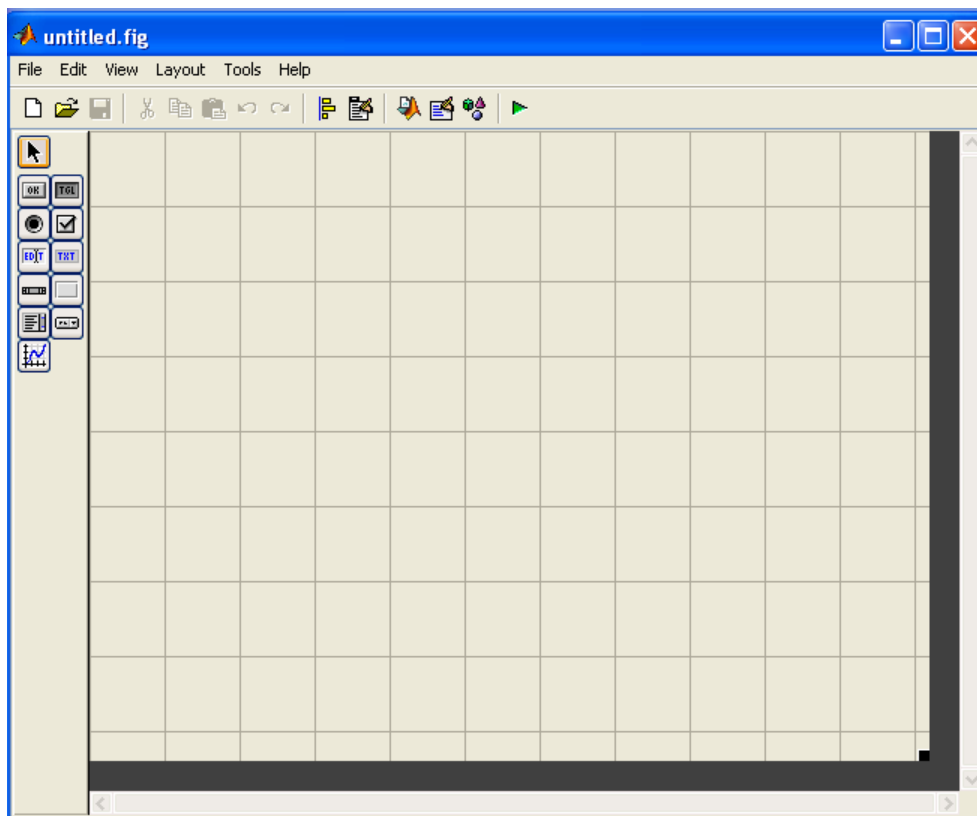


Рис. 8. Начальный экран


4. Режим создания поля для ввода текстовой информации с помощью кнопки  (она подсвечена желтым цветом, рисунок 9).



Рис. 9. Режим создания поля для ввода текстовой информации

5. Создание области для ввода аналитического выражения (рисунок 10).



Рис. 10. Поле для ввода аналитического выражения

6. Создание области для ввода значений границ интервала (рисунок 11).

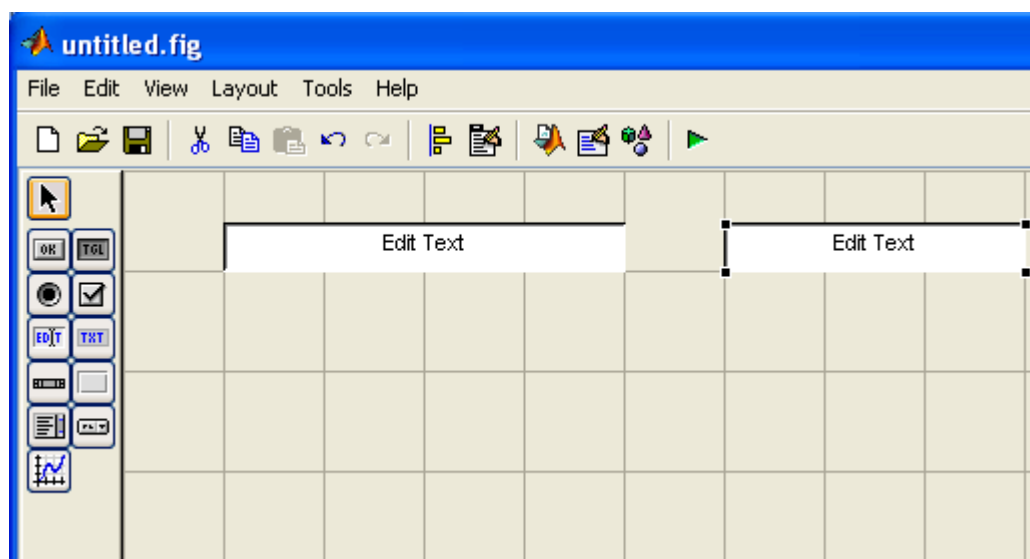


Рис. 11. Поле для ввода значений границ интервала

7. Режим создания поля для вывода информации в виде графиков и диаграмм с помощью кнопки



(она подсвечена желтым цветом, рисунок 12).



Рис. 12. Режим создания поля для вывода информации в виде графиков и диаграмм

8. Создание области для вывода информации в виде графиков и диаграмм (рисунок 13).

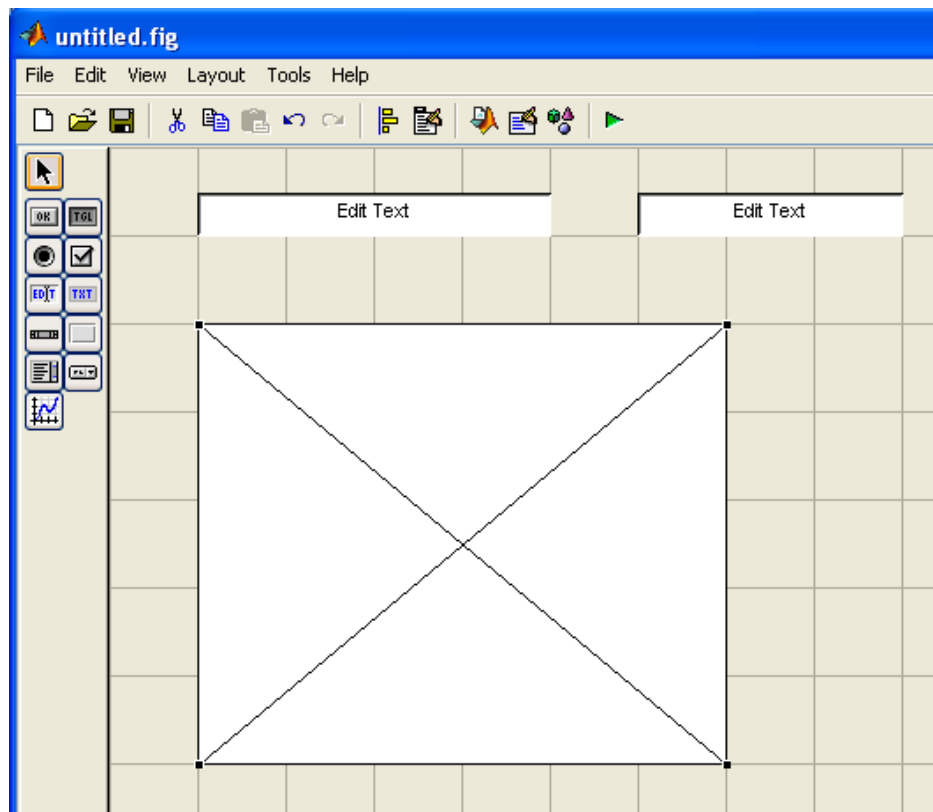


Рис. 13. Поле для вывода информации в виде графиков и диаграмм

9. Для области ввода аналитического выражения вызовом *Property Inspector* (при нажатии правой кнопки мыши, рисунок 14).

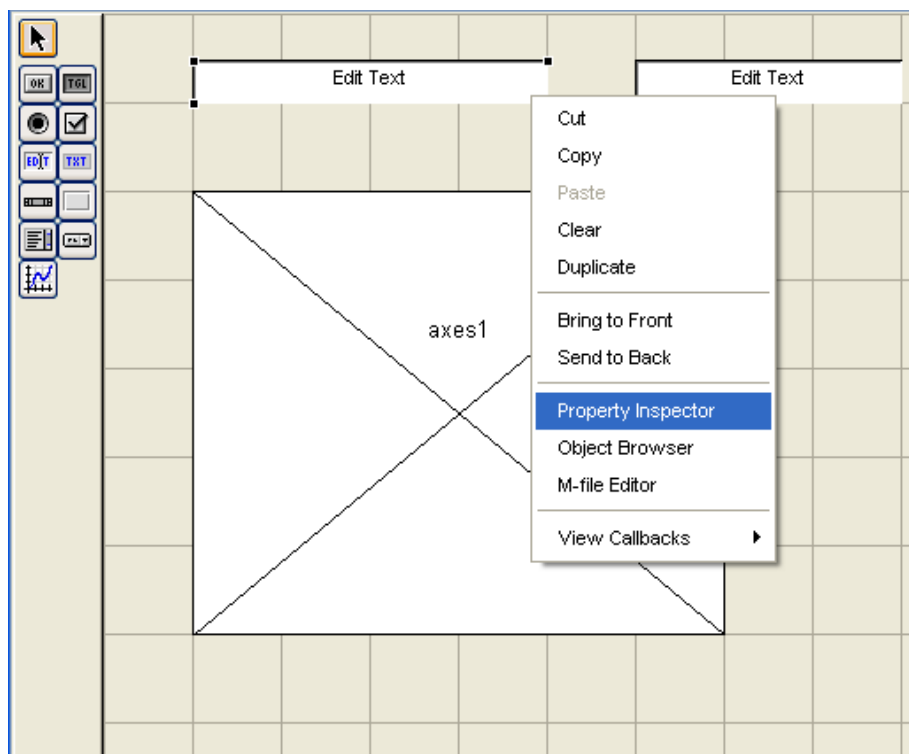


Рис. 14. *Property Inspector* для области ввода аналитического выражения

10. Свойство *Tag* и свойство *String* для области ввода аналитического выражения – исходные значения (рисунок 15).



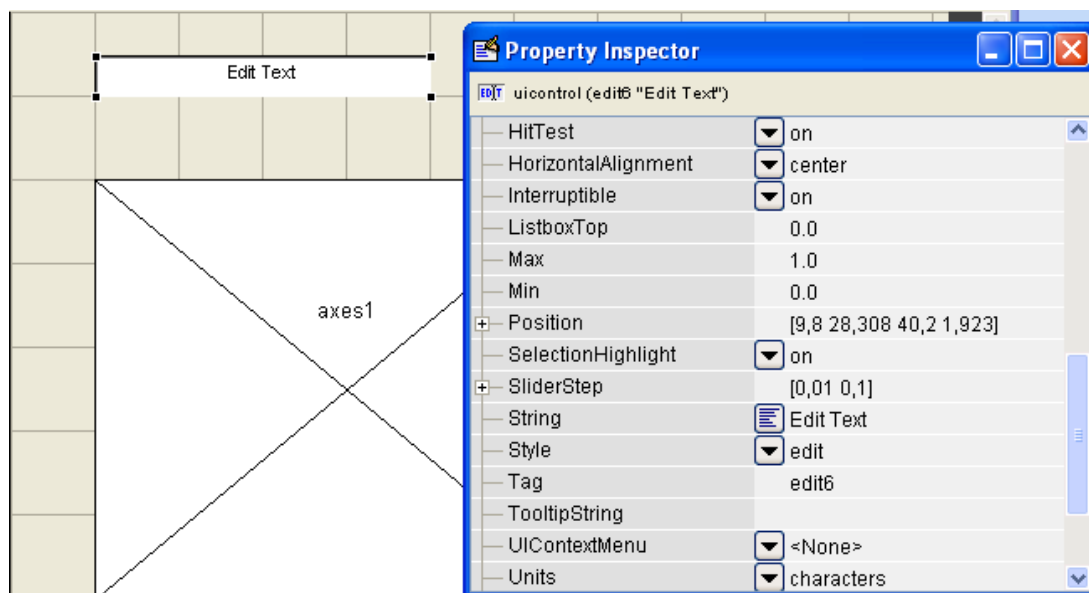


Рис. 15. Свойство *Tag* и свойство *String*

11. Для первой области (области ввода функции) свойству *Tag* присвоим значение *ede*, очистить значение свойства *String* (рисунок 16).

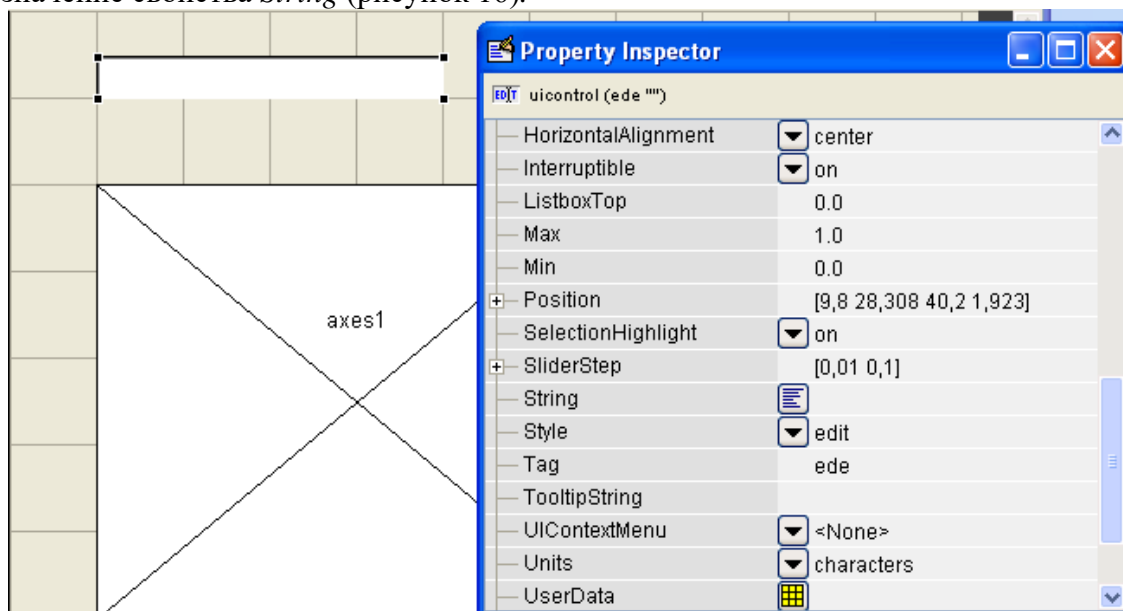


Рис. 16. Свойство *Tag* - значение *ede*

12. Аналогично для области ввода значений границ интервала вызовем *Property Inspector* (при нажатии правой кнопки мыши, рисунок 17).

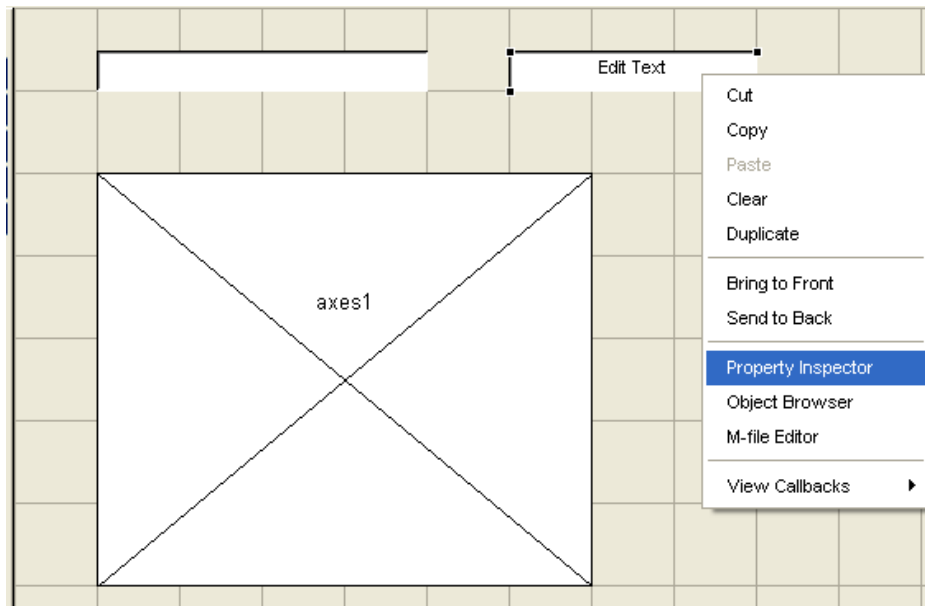


Рис. 17. *Property Inspector* для области ввода значений границ интервала

13. Для области ввода значений границ интервала свойству *Tag* присвоим значение *edi*, очистить значение свойства *String* (рисунок 18).

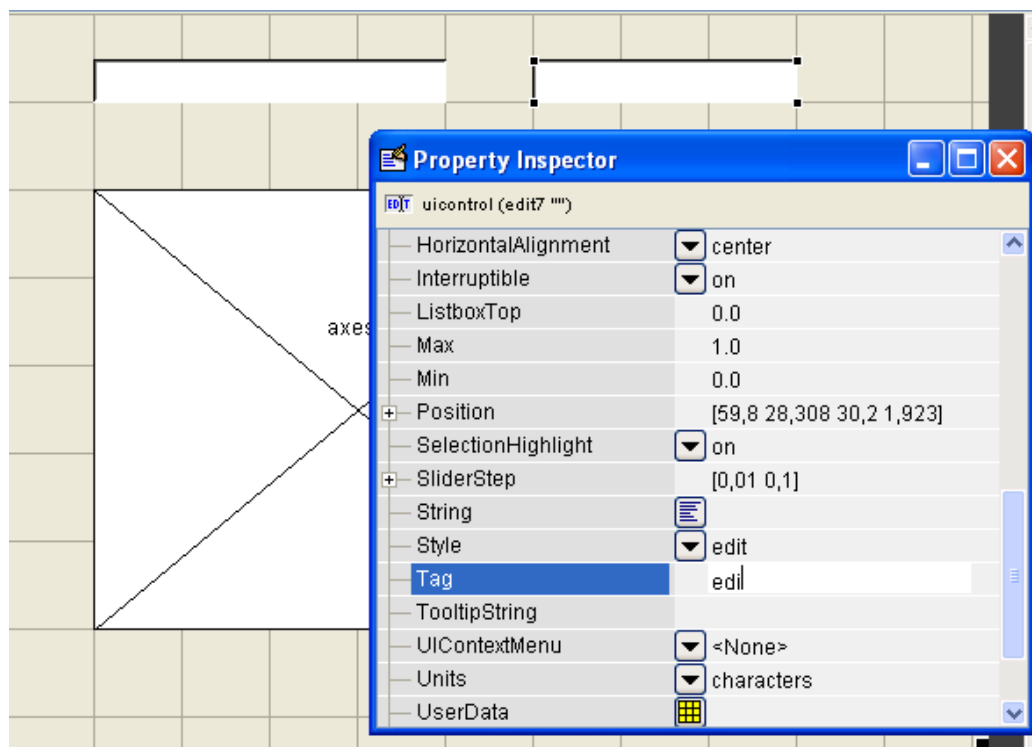


Рис. 18. Свойство *Tag* и свойство *String* для области ввода значений границ интервала

14. Аналогично для области вывода информации в виде графиков и диаграмм вызовем *Property Inspector* (при нажатии правой кнопки мыши, рисунок 19)

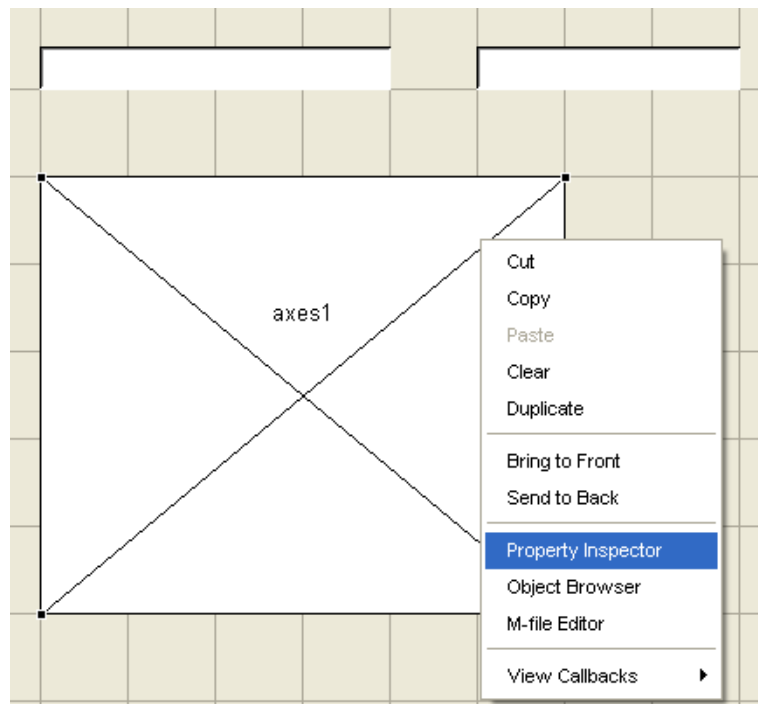


Рис. 19. *Property Inspector* для области вывода информации в виде графиков и диаграмм

15. Для области вывода информации в виде графиков и диаграмм свойству *Tag* присвоим значение *av* (рисунок 20).

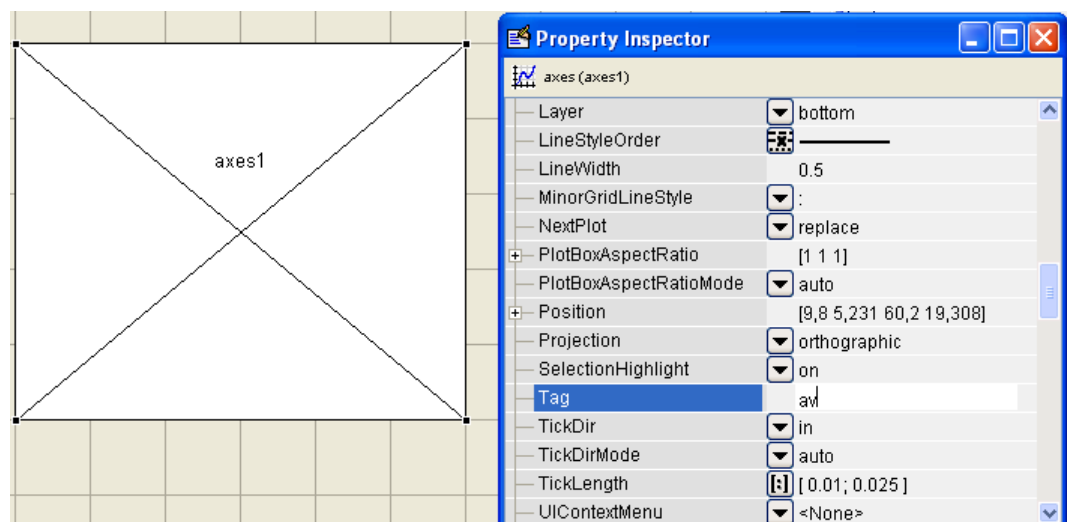


Рис. 20. Свойство *Tag* для области вывода информации в виде графиков и диаграмм



16. Режим создания кнопки (она подсвечена желтым цветом, рисунок 21).



Рис. 21. Режим создания поля кнопки.

17. Создание кнопки для построения графика функции (рисунок 22).

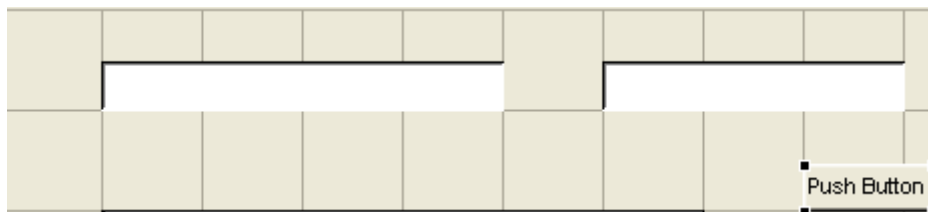


Рис. 22. Создание кнопки для построения графика функции

18. Для кнопки построения графика функции вызовем *Property Inspector* (при нажатии правой кнопки мыши, рисунок 23).

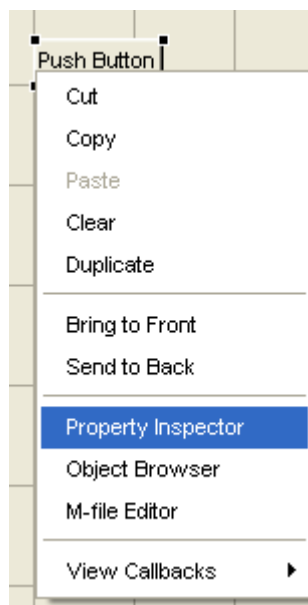


Рис. 23. *Property Inspector* для кнопки построения графика функции

19. Для кнопки построения графика функции свойству *Tag* присвоим значение *Bp*, свойству *String* присвоим значение *Plot* (рисунок 24).

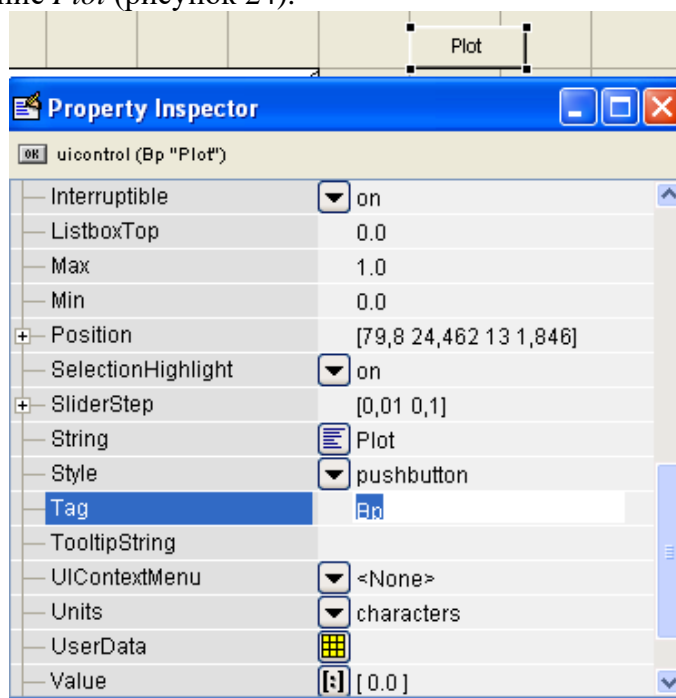


Рис. 24. Свойство *Tag* и свойство *String* для кнопки построения графика функции

20. Кнопка построения графика функции (рисунок 25).

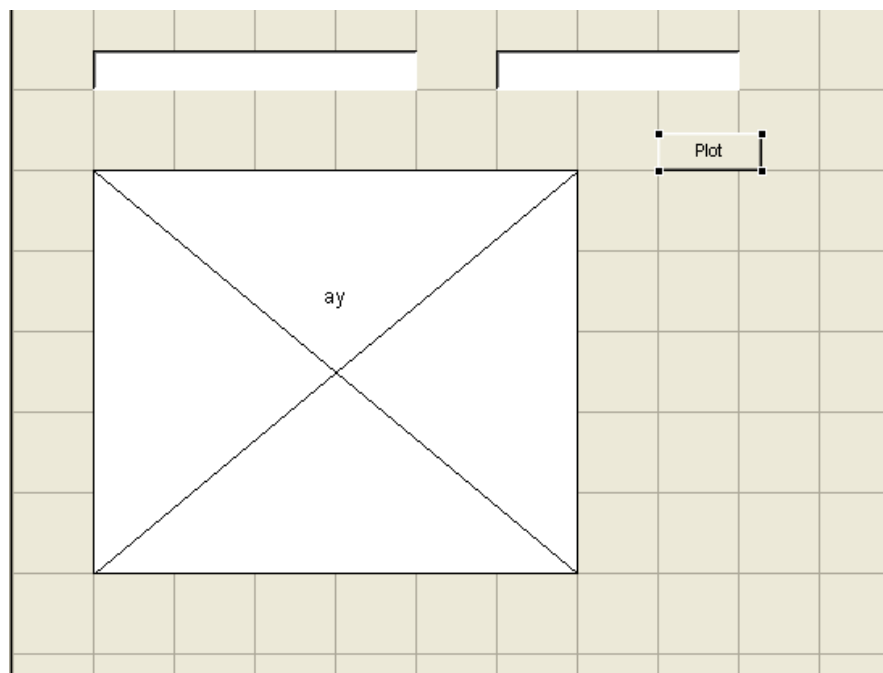


Рис. 25. Кнопка построения графика функции

21. Для кнопки построения графика функции вызовем событие *Callback* (рисунок 26).

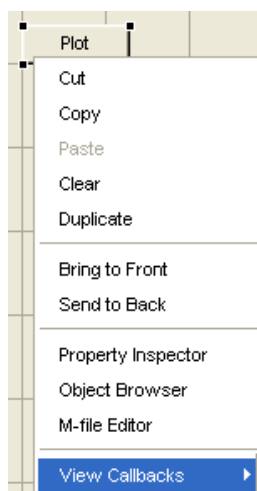


Рис. 26. Событие *Callback* для кнопки построения графика функции

22. Под найденным именем функции запишем следующую последовательность действий (рисунок 27).

```

function Bp_Callback(hObject, eventdata, handles)
% hObject    handle to Bp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

cla
interval=str2num(get(handles.edi,'String'));
f=inline(get(handles.ede,'String'));
[x,y]=fplot(f,interval);
handles.line=plot(x,y,'c-');
guidata(gcbo,handles);
hold on

```

Рис. 27. Последовательность действий для события *Callback* (построение графика функции)

23. Осуществим запуск приложения с помощью кнопки  (ввести данные, рисунок 28).

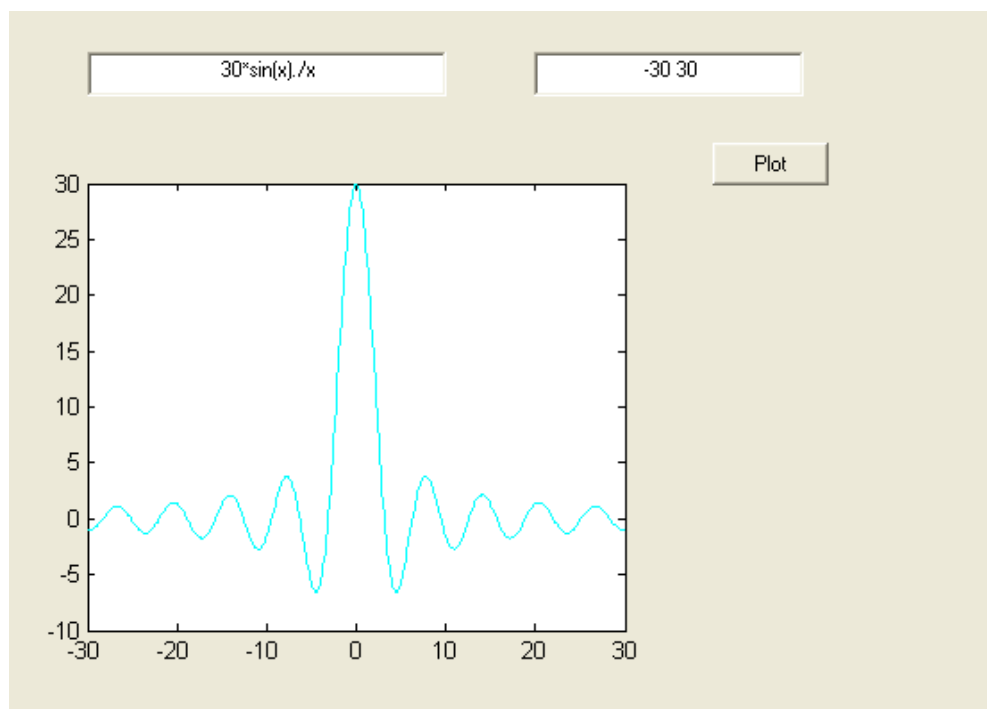


Рис. 28. Построение графика функции

24. Построим кнопку *Zero* для нахождения локального минимума функции  $\sin(x)$  на интервале  $[-3 \ 3]$ . Свойству *Tag* присвоим значение *Bzero*, свойству *String* присвоим значение *Zero* (рисунок 29).

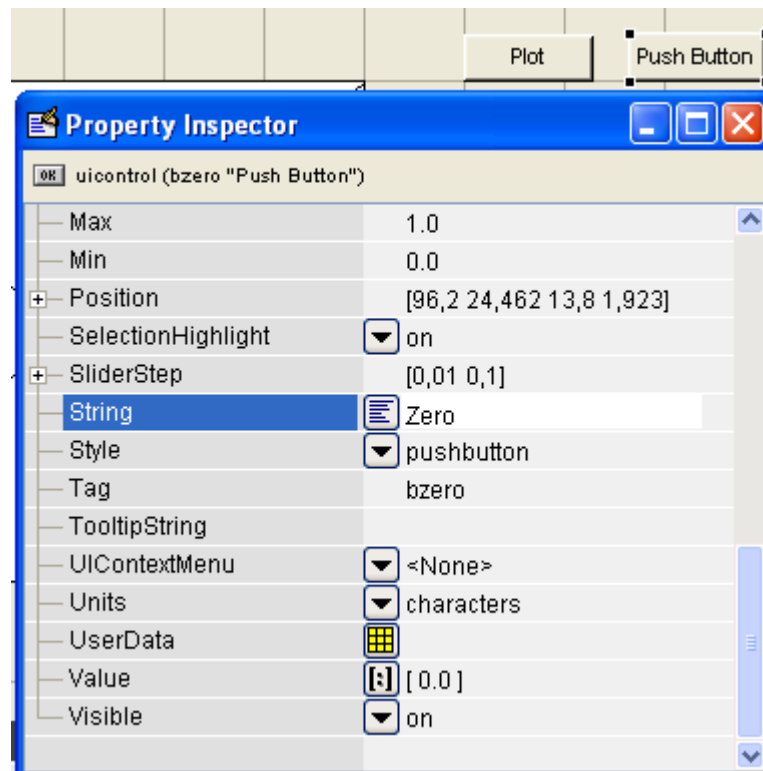


Рис. 29. Свойство *Tag* и свойство *String* для кнопки *Zero*

25. Кнопка нахождения локального минимума функции  $\sin(x)$  на интервале  $[-3 \ 3]$ . (рисунок 30).

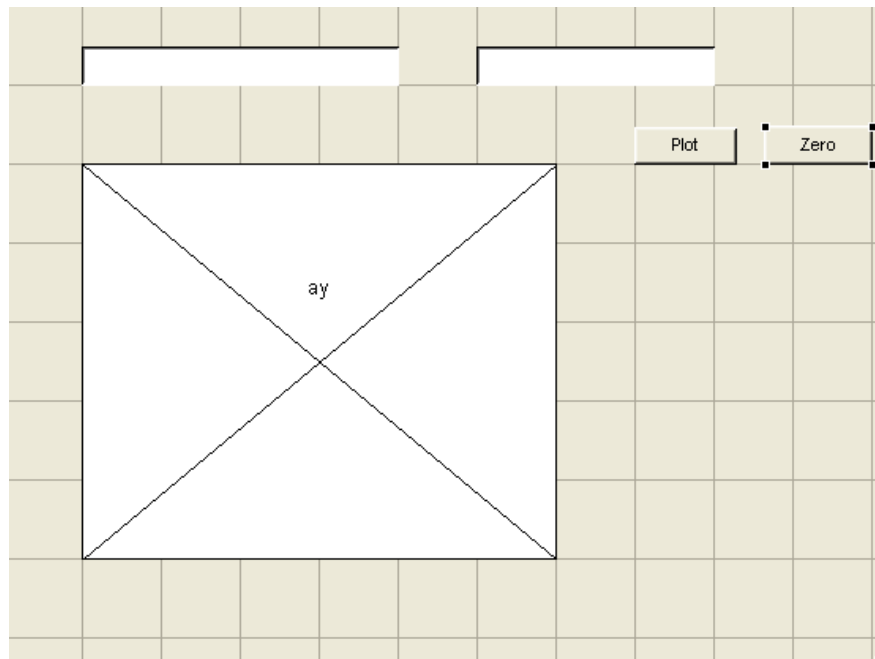


Рис. 30. Кнопка нахождения локального минимума

26. Вызовем событие *Callback* для кнопки нахождения локального минимума. Под найденным именем функции запишем следующую последовательность действий (рисунок 31).

```

function bzero_Callback(hObject, eventdata, handles)
% hObject    handle to bzero (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

interval=str2num(get(handles.edi,'String'));
x1=interval(1);
x2=interval(2);
f=inline(get(handles.ede,'String'));
x=fzero(f,(x1+x2)/2);
y=f(x);
plot(x,y,'g.', 'MarkerSize',25)

```

Рис. 31. Последовательность действий для события *Callback*

27. Осуществим запуск приложения с помощью кнопки  (ввести данные, рисунок 32).

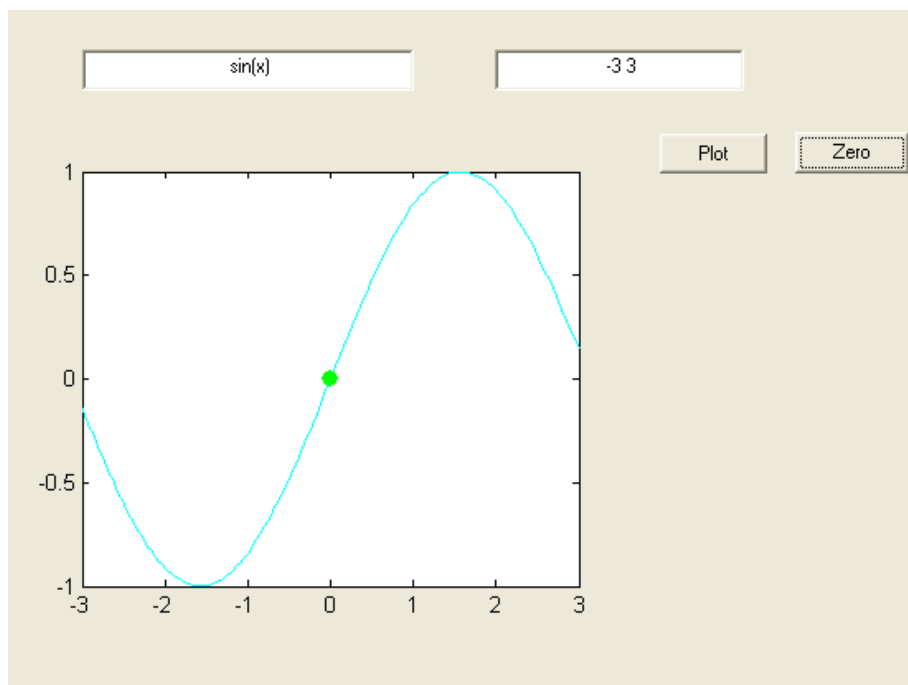


Рис. 32. Построение графика функции

28. Для управления появлением и удалением координатной сетки используем элемент управления *CheckBox* (рисунок 33).



Рис. 33. Режим создания флажка

29. Элемент управления *CheckBox* (рисунок 34).



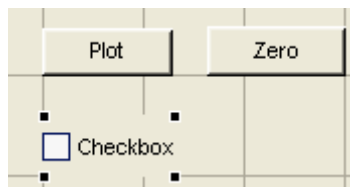


Рис. 34. Элемент управления *CheckBox*

30. Для элемента управления *CheckBox* вызовем *Property Inspector* (при нажатии правой кнопки мыши, рисунок 35).

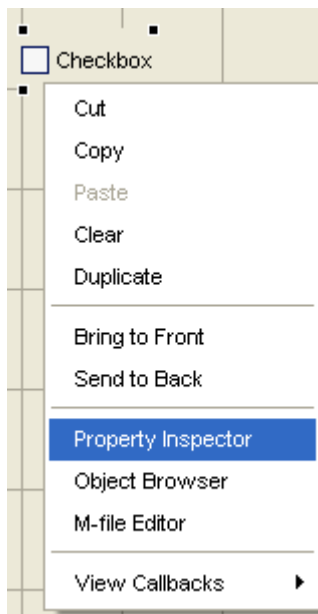


Рис. 35. *Property Inspector* для элемента управления *CheckBox*

31. Свойству *Tag* присвоим значение *cX*, свойству *String* присвоим значение *GridX* (рисунок 36).

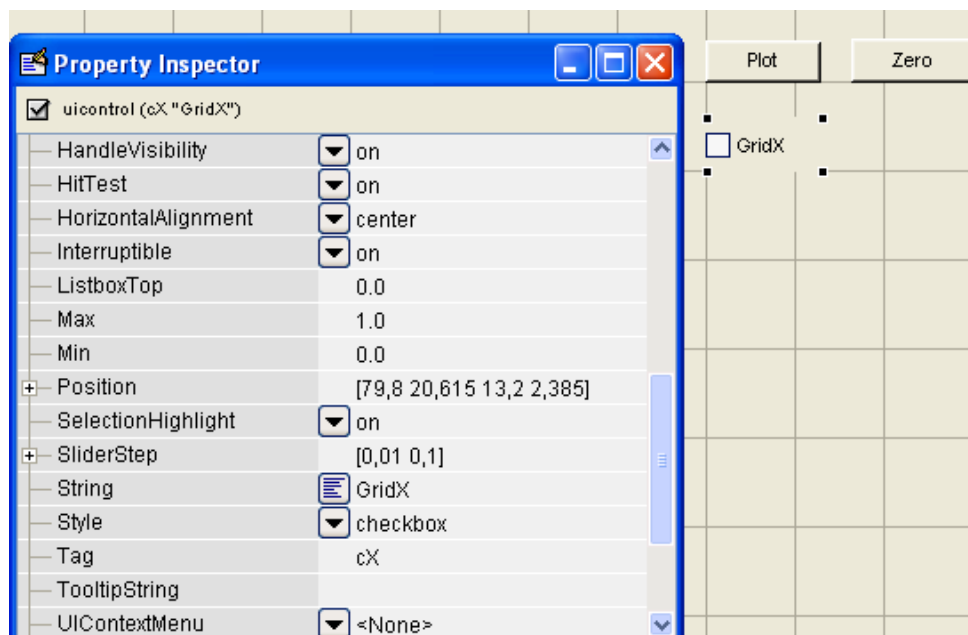


Рис. 36. Свойство *Tag* и свойство *String* для элемента управления *CheckBox*

32. Вызовем событие *Callback* для элемента управления *CheckBox* (при нажатии правой кнопки мыши, рисунок 37).

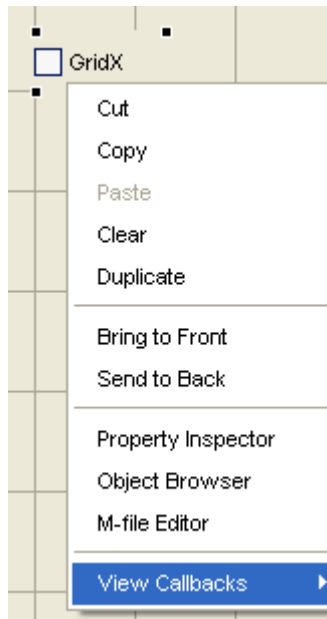


Рис. 37. Событие *Callback* для элемента управления *CheckBox*


33. Под найденным именем функции запишем следующую последовательность действий (рисунок 38).

```
function cX_Callback(hObject, eventdata, handles)
% hObject    handle to cX (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

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

if get(hObject, 'Value')
    set(gca, 'XGrid', 'on')
else
    set(gca, 'XGrid', 'off')
end
```

Рис. 38. Последовательность действий для события *Callback*

34. Осуществим запуск приложения с помощью кнопки  (ввести данные, рисунок 39).

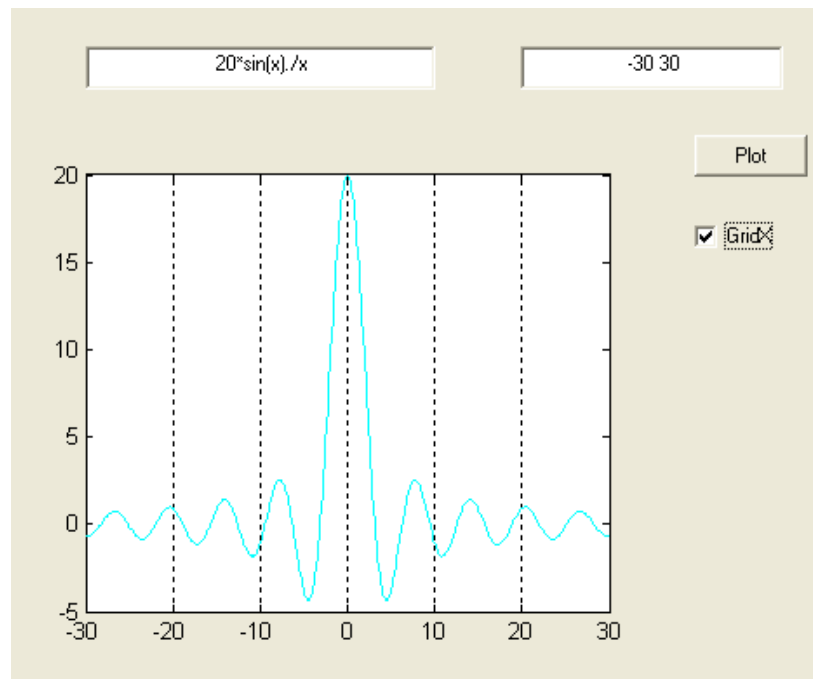


Рис. 39. Построение графика функции

35. Для создания элемента управления, изменяющего цвет (свойство *Color*) построения кривой воспользуемся режимом создания раскрывающегося списка (рисунок 40).



Рис. 40. Режим создания раскрывающегося списка

36. Элемент управления, изменяющий цвет (свойство *Color*) построения кривой (рисунок 41).

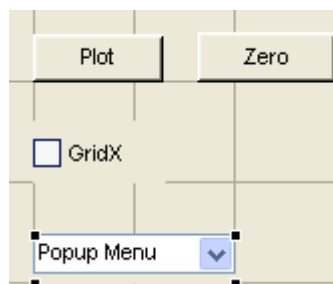


Рис. 41. Элемент управления, изменяющий цвет (свойство *Color*) построения кривой

37. Для элемента управления, изменяющего цвет (свойство *Color*) построения кривой вызовем *Property Inspector* (при нажатии правой кнопки мыши, рисунок 42).

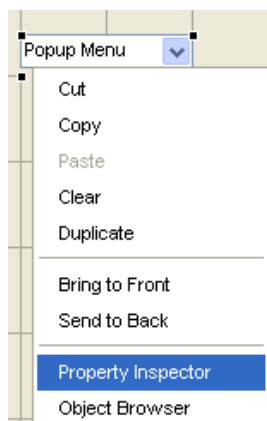


Рис. 42. *Property Inspector* для элемента управления *Popup Menu*

38. Свойству *Tag* присвоим значение *pColor* (рисунок 43).

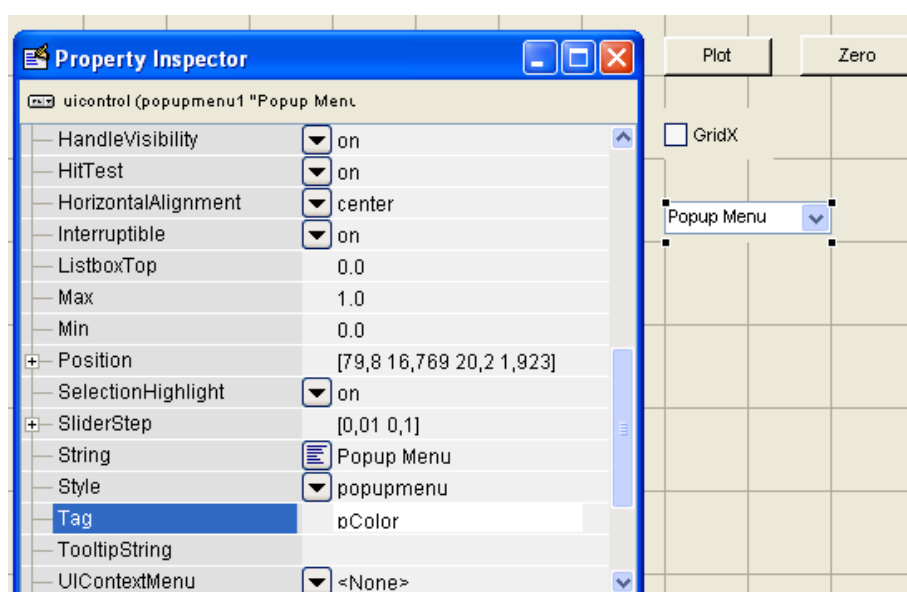


Рис. 43. Свойство *Tag*

39. На кнопке рядом со свойством *String* в окне свойств элемента возникает следующее окно (рисунок 44).

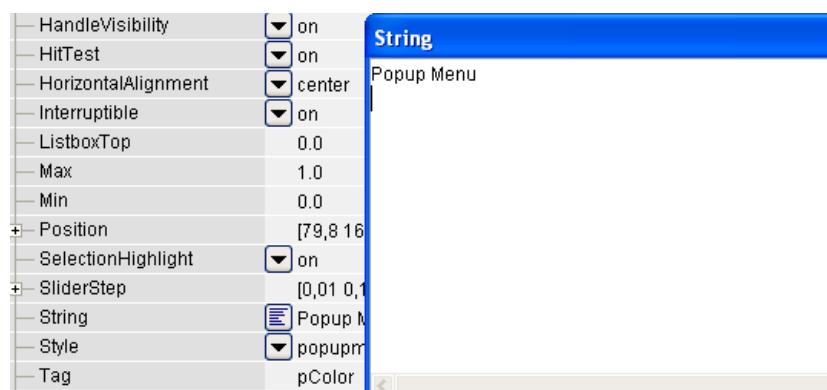


Рис. 44. Свойство *String*

40. В появившемся окне необходимо ввести название каждого пункта списка (рисунок 45). Название записывается одной строкой и отделяется от следующей нажатием клавиши *Enter*.

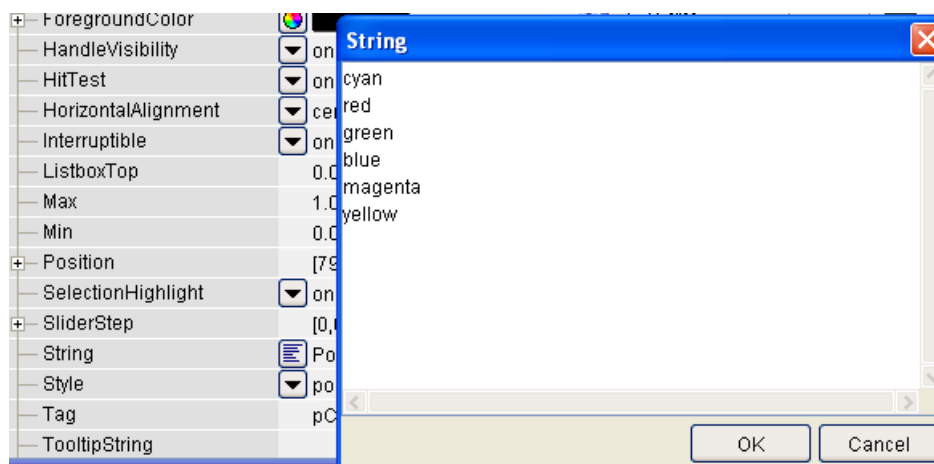


Рис. 45. Пункты списка для свойства *String*


41. Вызовем событие *Callback* для элемента управления *Popur Menu*. Под найденным именем функции запишем следующую последовательность действий (рисунок 46).

```
function pColor_Callback(hObject, eventdata, handles)
% hObject    handle to pColor (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns pColor contents as cell array
%          contents{get(hObject,'Value')} returns selected item from pColor

Num=get(hObject, 'Value');
switch Num
case 1
    set(handles.line, 'Color', 'cyan');
case 2
    set(handles.line, 'Color', 'red');
case 3
    set(handles.line, 'Color', 'green');
case 4
    set(handles.line, 'Color', 'blue');
case 5
    set(handles.line, 'Color', 'magenta');
case 6
    set(handles.line, 'Color', 'yellow');
end
```

Рис. 46. Последовательность действий для события *Callback*

42. Осуществим запуск приложения с помощью кнопки  (ввести данные, рисунок 47).

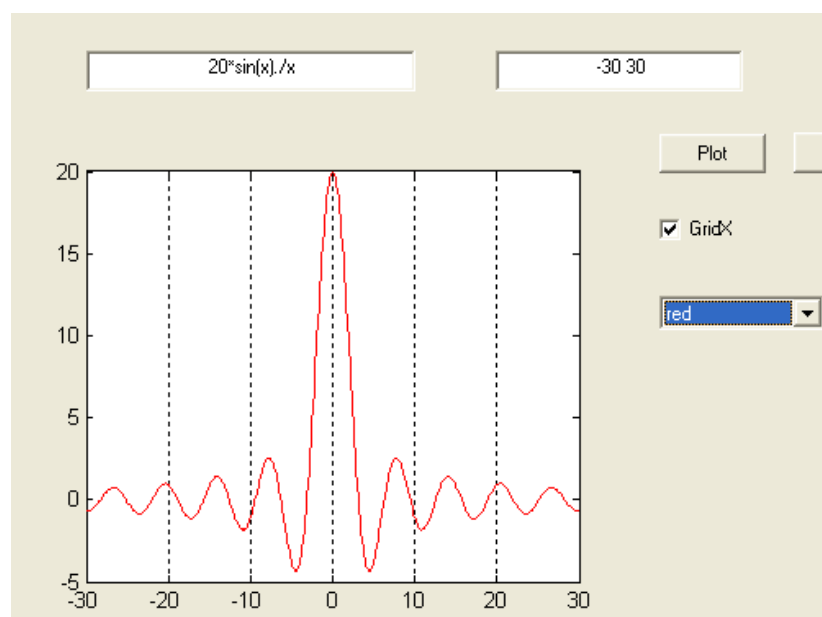


Рис. 47. Построение графика функции