**Министерство образования и науки Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования «КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ» ИНСТИТУТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ**

**Отчет**

    По учебной (эксплуатационной) практике

(наименование практики)

Обучающийся  гр. 09-832 Гусев В.Е.            (ФИО, группа) (подпись)

Руководитель практики на

кафедре вычислительной математики       ассистент КВМ Глазырина О.В. (должность, ФИО)

оценка за практику

(подпись руководителя практики)

Дата сдачи отчета

КАЗАНЬ – 2020 г.

Оглавление

[Введение 2](#_Toc41050934)

[Создание проекта 3](#_Toc41050935)

[Построение графиков 5](#_Toc41050936)

[Сетка 7](#_Toc41050937)

[Маркеры 9](#_Toc41050938)

[Цвет линии 12](#_Toc41050939)

[Толщина линии 14](#_Toc41050940)

[Заключение 15](#_Toc41050941)

[Список использованной литературы 16](#_Toc41050942)

[Листинг 17](#_Toc41050943)

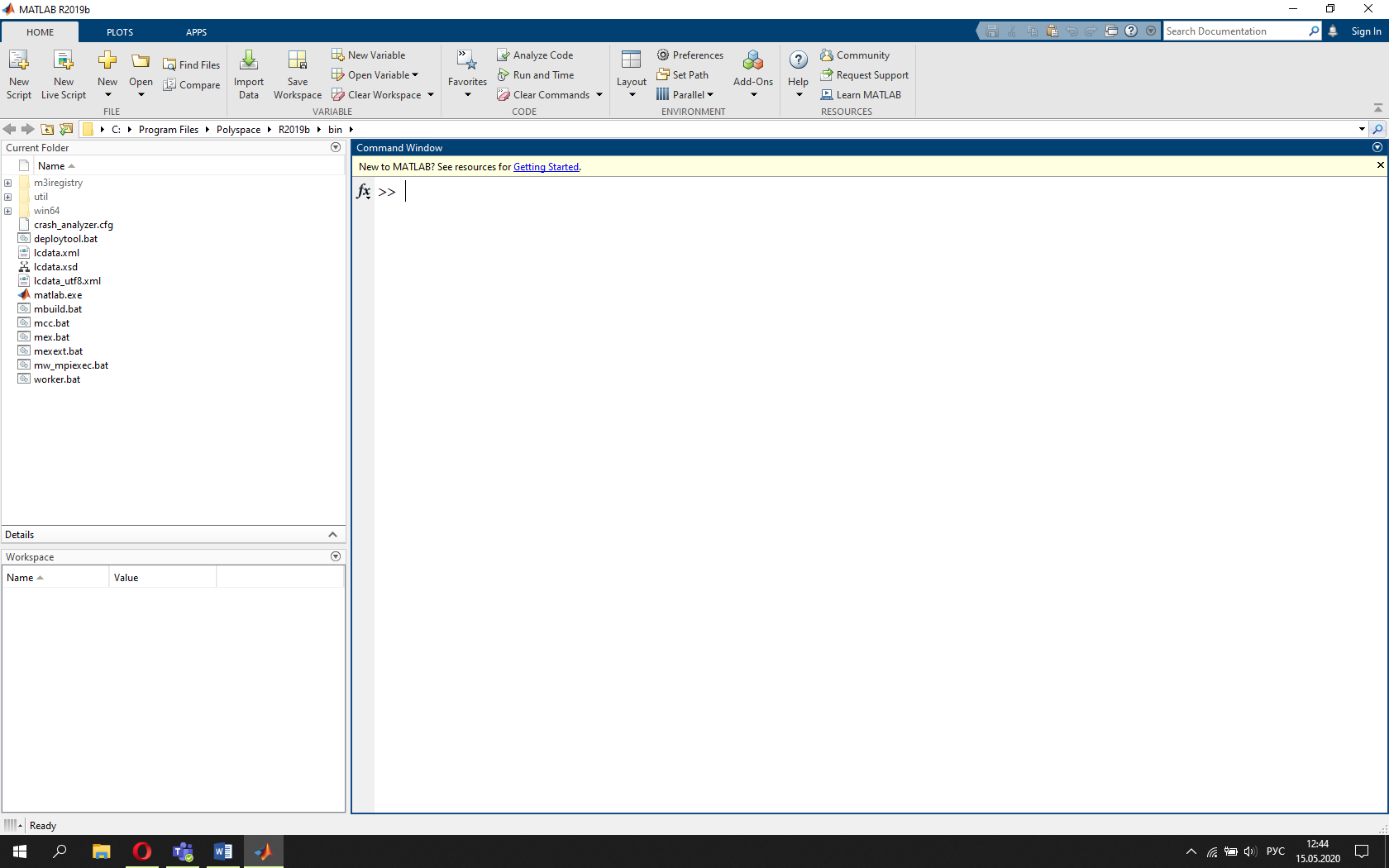
# Введение

В данном отчете идёт описание пошаговой разработки Графического интерфейса в высокоуровневом интерпретируемом языке программирования MATLAB.

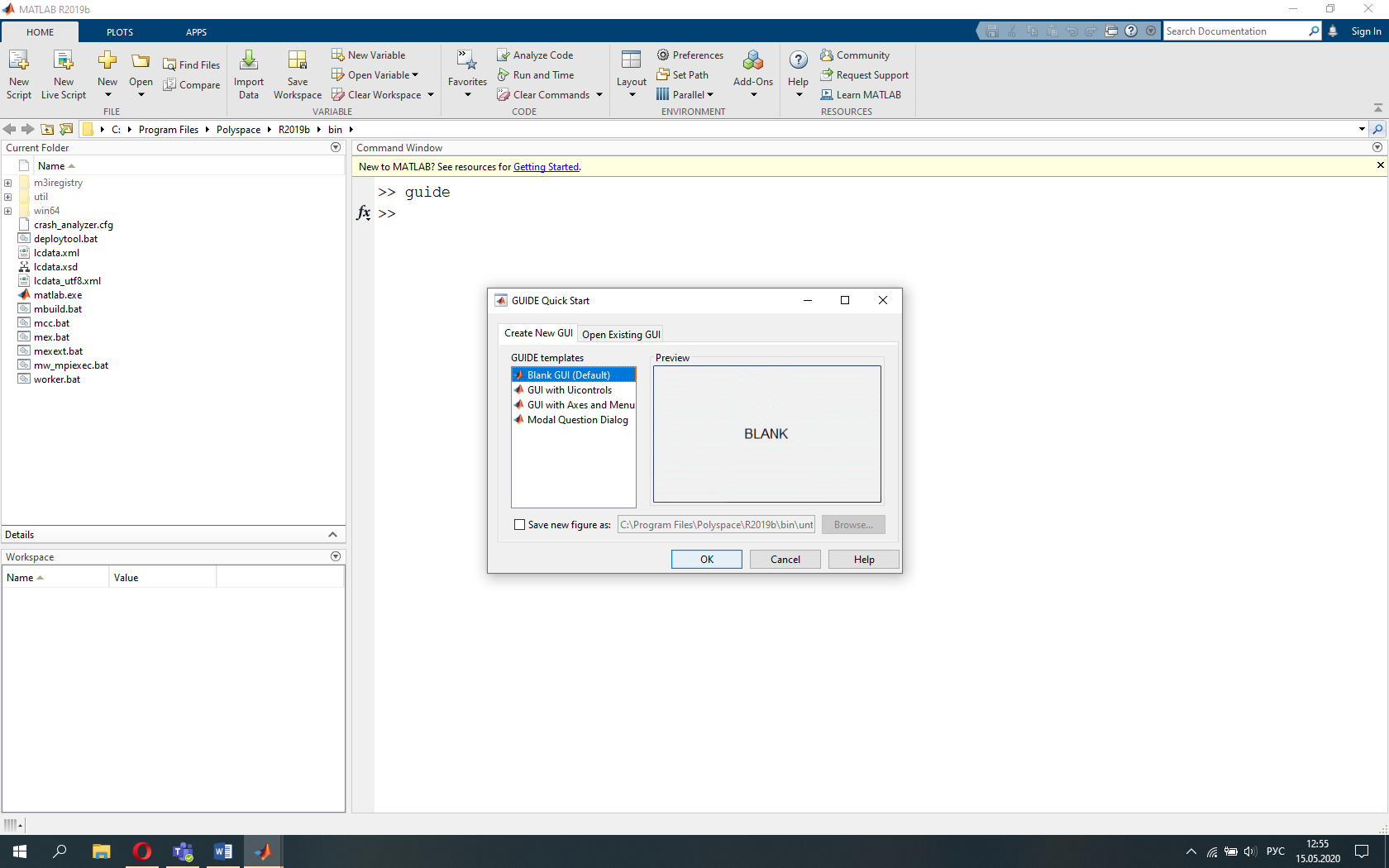
MATLAB (сокращение от англ. «Matrix Laboratory», в русском языке произносится как Матла́б) — пакет прикладных программ для решения задач технических вычислений. Пакет используют более миллиона инженерных и научных работников, он работает на большинстве современных операционных систем, включая Linux, Mac OS, Solaris (начиная с версии R2010b поддержка Solaris прекращена) и Windows.

# Создание проекта

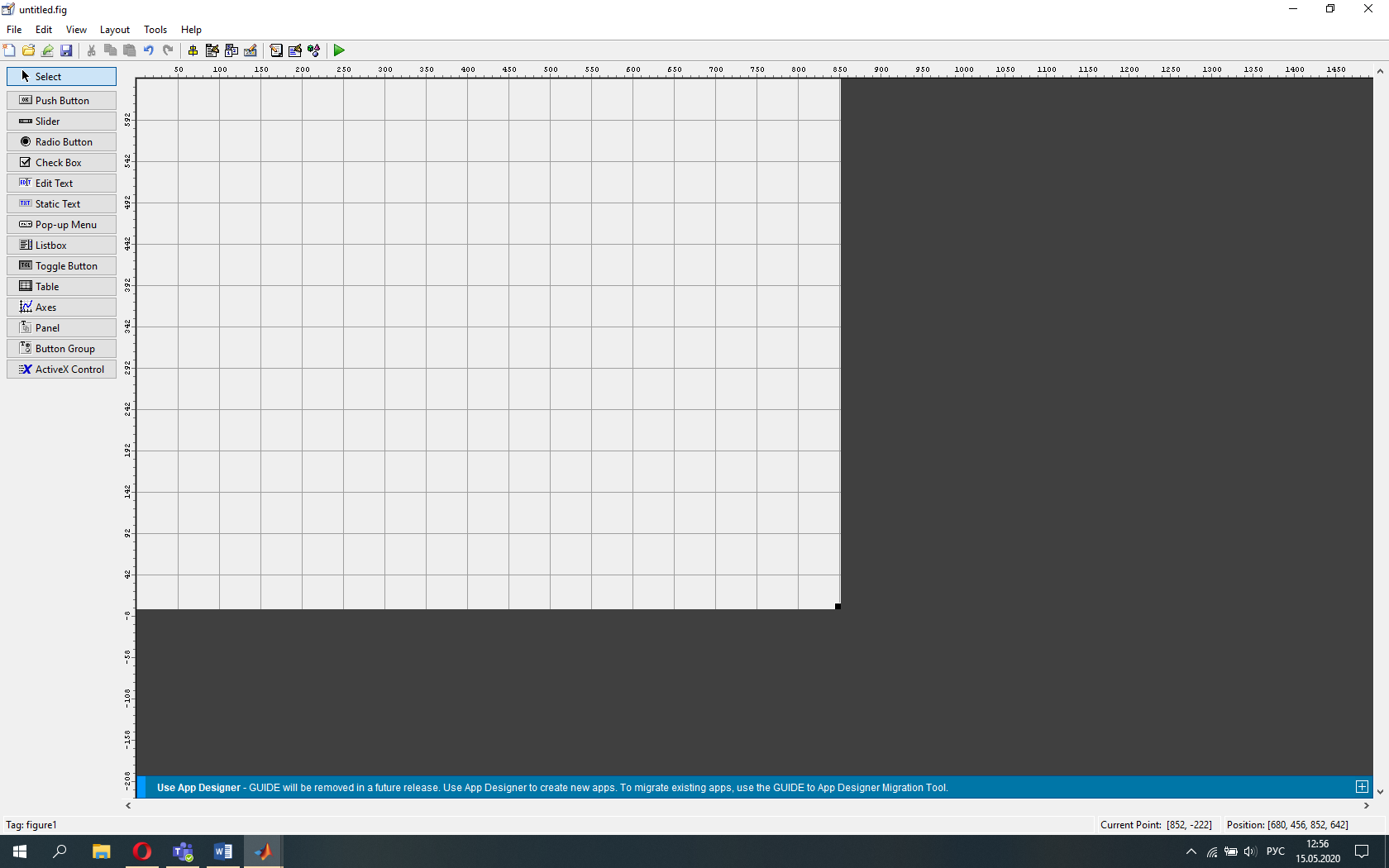
После установки и запуска программы открывается среда программирования.



В Command window можно вводить различные команды и математические функции. Чтобы открыть «GUIDE quick start», нужно ввести «guide». Здесь создается новый проект во вкладке «Create New GUI», либо открывается существующий во вкладке «Open Existing GUI». Для создания нового проекта нужно выбрать в «GUIDE templates» пункт «Blank GUI (default)».



Так открывается пустой интерфейс, с которым ведется работа.

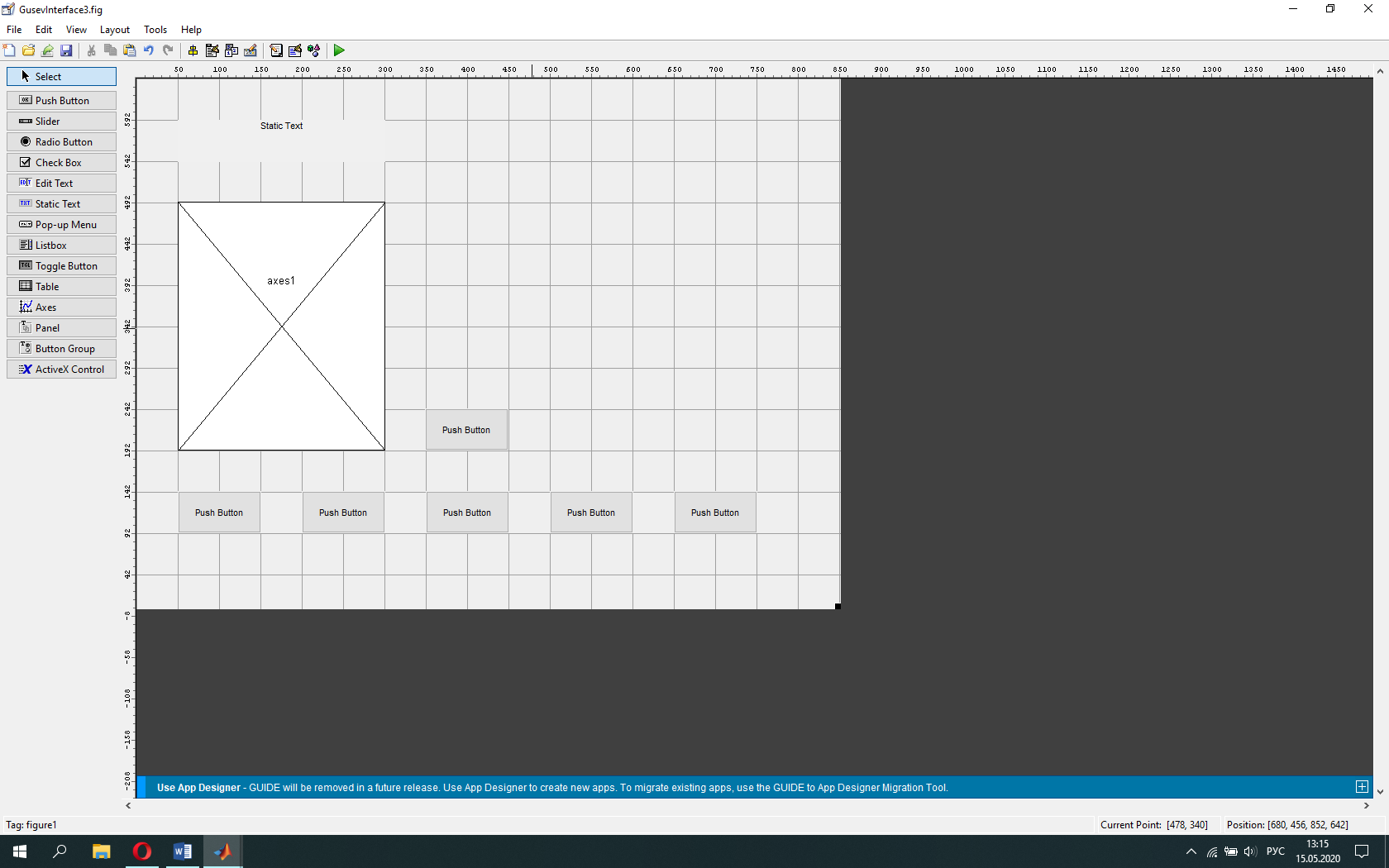


Проект сохраняется комбинацией клавишей Ctrl+S или нажатием на знак «Сохранить» сверху. Выбирается нужное место для сохранения и задается название интерфейса. С данным файлом ведется дальнейшая работа.

# Построение графиков

Слева располагаются различные элементы, которые накладываются на форму и задаются нужные размеры.

Для начала нужно расположить на форму Axes, 6-шт Push Button и Static Text.



При нажатии на элемент и сверху на Property Inspector, открывается настройка элемента на форме. В поле String вводится текст, который виден пользователю на элементе. В Property Inspector задается Background color (цвет фона), FontSize (размер текста) и т.д.

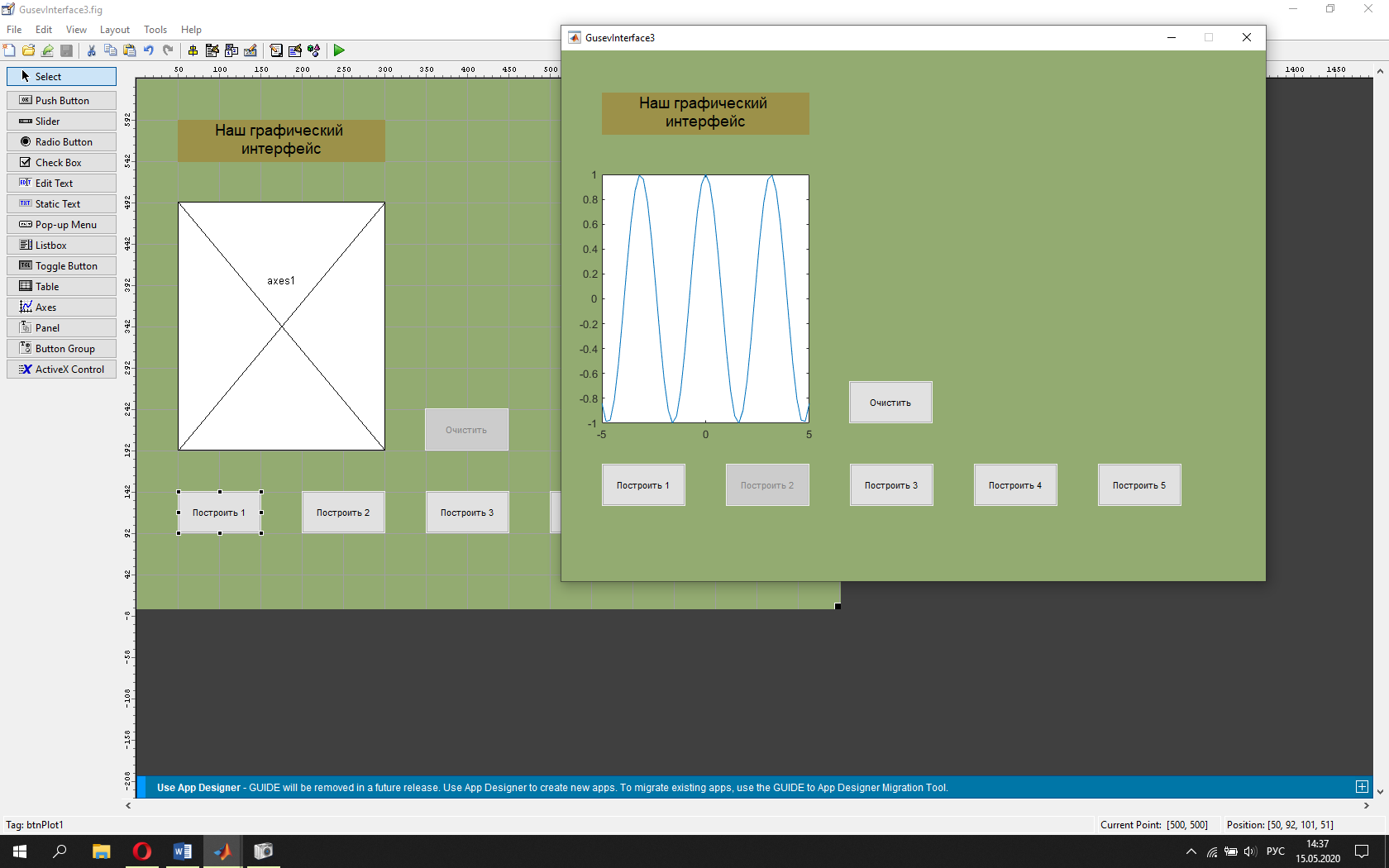
Таким образом настраиваются любые элементы на форме. Для настройки первой кнопки в поле String прописывается «Построить 1», в поле Tag (здесь пишется название переменной, к которой идет обращение при программировании) вводится «btnPlot1», в Tooltip (при наведении на элемент будет показываться текст) - «cos(x)». Аналогично настраиваются остальные кнопки.

При настройке кнопки «Очистить» полю Enable (активность кнопки) задается «off», чтобы при включении формы она была выключена.

Чтобы запрограммировать элемент на форме, нужно нажать правой кнопкой мыши на элемент, выбрать View Callbacks и нажать на Callback.

Открывается Editor, в котором программируются все элементы. В переменной x задается вектор, в y – функция. handles.Line=plot(x,y) рисует график. Set(hObject, ‘Enable’, ‘off’) выключает кнопку при нажатии, set(handles.btnPlot2, ‘Enable’, ‘on’) включает определенную кнопку. При нажатии на «Построить 1» будет рисоваться график. Аналогично программируются остальные кнопки.

Для кнопки «Очистить» пишется «cla», чтобы очистить график.



# Сетка

Для нанесения сетки на график, нужно перетащить на форму Panel. Полю Title присваивается значение «Сетка», TitlePosition (положение Title на Panel) – centertop (середина сверху), Fontsize – 10.0 и BackgroundColor по предпочтению.

На панель накладываются два Check Box. Одному присваивается String – «Сетка по X» и Tag – «chbxGridX», другому присваивается String – «Сетка по Y» и Tag – «chbxGridY».

Далее программируются Check Box.

function chbxGridX\_Callback(hObject, eventdata, handles)

if get(hObject, 'Value')

set(gca,'XGrid', 'on')

else

set(gca,'XGrid','off')

end

function chbxGridY\_Callback(hObject, eventdata, handles)

if get(hObject, 'Value')

set(gca,'YGrid', 'on')

else

set(gca,'YGrid','off')

end

Если включен Check Box, то рисуется сетка по X или Y, иначе она выключена.

Так же нужно дописать в функции каждой кнопки «Построить» проверку на наличие сетки. В случае ее включении, сетка будет рисоваться вместе с графиком. Без данных условий сетка будет стираться при рисовании нового графика.

if get(handles.chbxGridX, 'Value')

set(gca,'XGrid', 'on')

else

set(gca,'XGrid','off')

end

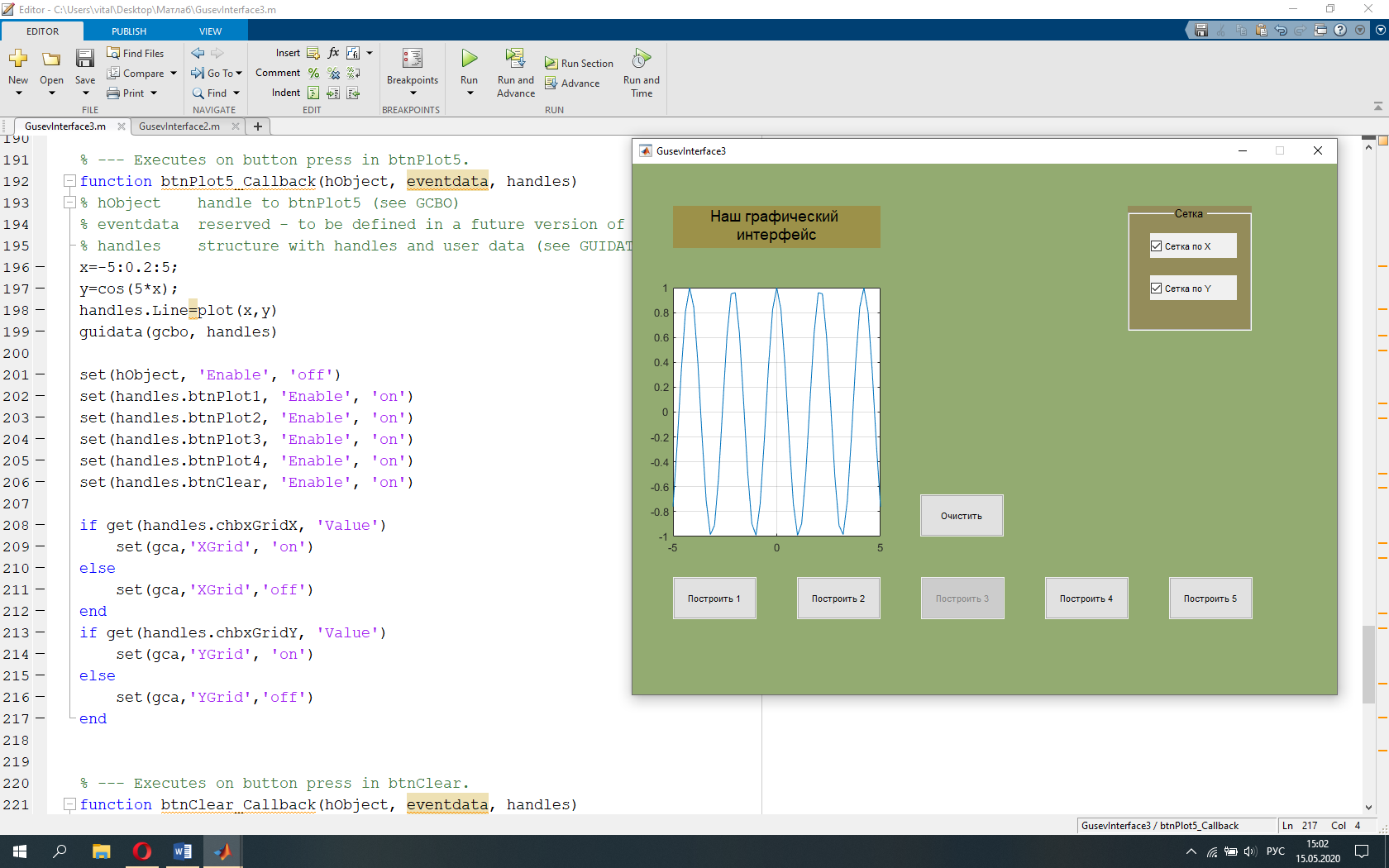
if get(handles.chbxGridY, 'Value')

set(gca,'YGrid', 'on')

else

set(gca,'YGrid','off')

end



# Маркеры

Для нанесения маркеров на построенные графики, нужно создать Button Group и 6-шт Radio Button на форме. В Button Group меняется поле Title на «Маркеры», TitlePosition – «centertop», FontSize на 10.0 и BackgroundColor по предпочтению.

Во всех Radio Button меняется поле «Enable» на «off», поля String на «Звездочки», «Треугольники», «Ромбики», «Круги», «Квадраты» и «Без маркера». Поля Tag на «rbMarkStar», «rbMarkTriangle», «rbMarkRhombus», «rbMarkCirc», «rbMarkSq» и «rbMarkNone». У «Без маркера» поле Value меняется на 1.0, чтобы при запуске данный Radio Button был включен.

Далее нужно запрограммировать все Radio Button. Ниже приведен код для нанесения маркеров-кругов, остальные маркеры программируются аналогично.

function rbMarkCirc\_Callback(hObject, eventdata, handles)

set(handles.Line, 'Marker', 'o')

set(hObject, 'Enable', 'inactive')

set(handles.rbMarkSq, 'Enable', 'on')

set(handles.rbMarkNone, 'Enable', 'on')

set(handles.rbMarkStar, 'Enable', 'on')

set(handles.rbMarkTriangle, 'Enable', 'on')

set(handles.rbMarkRhombus, 'Enable', 'on')

В первой функции всех Radio Button присваивается линии определенный маркер, в следующей функции данный Radio Button делается «inactive», чтобы после включения, на него нельзя было нажимать. Далее в функциях остальные Radio Button делаются активными.

Для всех кнопок «Построить» нужно прописать функции включения всех Radio Button и написать условия, при помощи которых будут задаваться нужные маркеры графику при построении и условия, при которых, в случае активности Radio Button, его делает «inactive».

set(handles.rbMarkCirc, 'Enable', 'on')

set(handles.rbMarkSq, 'Enable', 'on')

set(handles.rbMarkNone, 'Enable', 'on')

set(handles.rbMarkStar, 'Enable', 'on')

set(handles.rbMarkTriangle, 'Enable', 'on')

set(handles.rbMarkRhombus, 'Enable', 'on')

if get(handles.rbMarkCirc, 'Value')

set(handles.Line, 'Marker', 'o')

end

if get(handles.rbMarkSq, 'Value')

set(handles.Line, 'Marker', 's')

end

if get(handles.rbMarkStar, 'Value')

set(handles.Line, 'Marker', '\*')

end

if get(handles.rbMarkTriangle, 'Value')

set(handles.Line, 'Marker', '^')

end

if get(handles.rbMarkRhombus, 'Value')

set(handles.Line, 'Marker', 'd')

end

if get(handles.rbMarkCirc, 'Value')

set(handles.rbMarkCirc, 'Enable', 'inactive')

end

if get(handles.rbMarkSq, 'Value')

set(handles.rbMarkSq, 'Enable', 'inactive')

end

if get(handles.rbMarkNone, 'Value')

set(handles.rbMarkNone, 'Enable', 'inactive')

end

if get(handles.rbMarkStar, 'Value')

set(handles.rbMarkStar, 'Enable', 'inactive')

end

if get(handles.rbMarkTriangle, 'Value')

set(handles.rbMarkTriangle, 'Enable', 'inactive')

end

if get(handles.rbMarkRhombus, 'Value')

set(handles.rbMarkRhombus, 'Enable', 'inactive')

end

В функцию кнопки «Очистить» нужно прописать функции выключения всех Radio Button, чтобы не было возможности их менять, когда график не нарисован.

set(handles.rbMarkCirc, 'Enable', 'off')

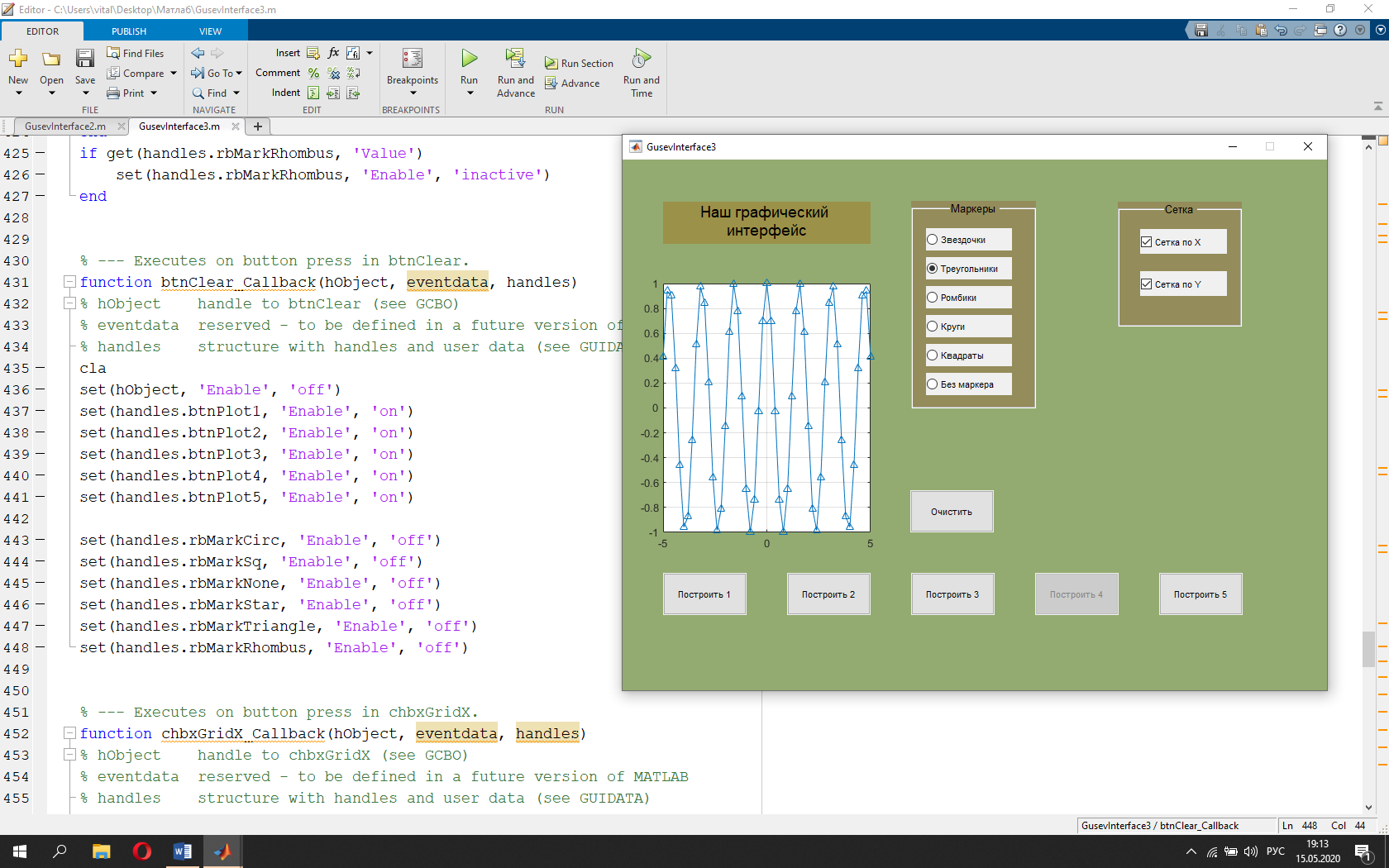
set(handles.rbMarkSq, 'Enable', 'off')

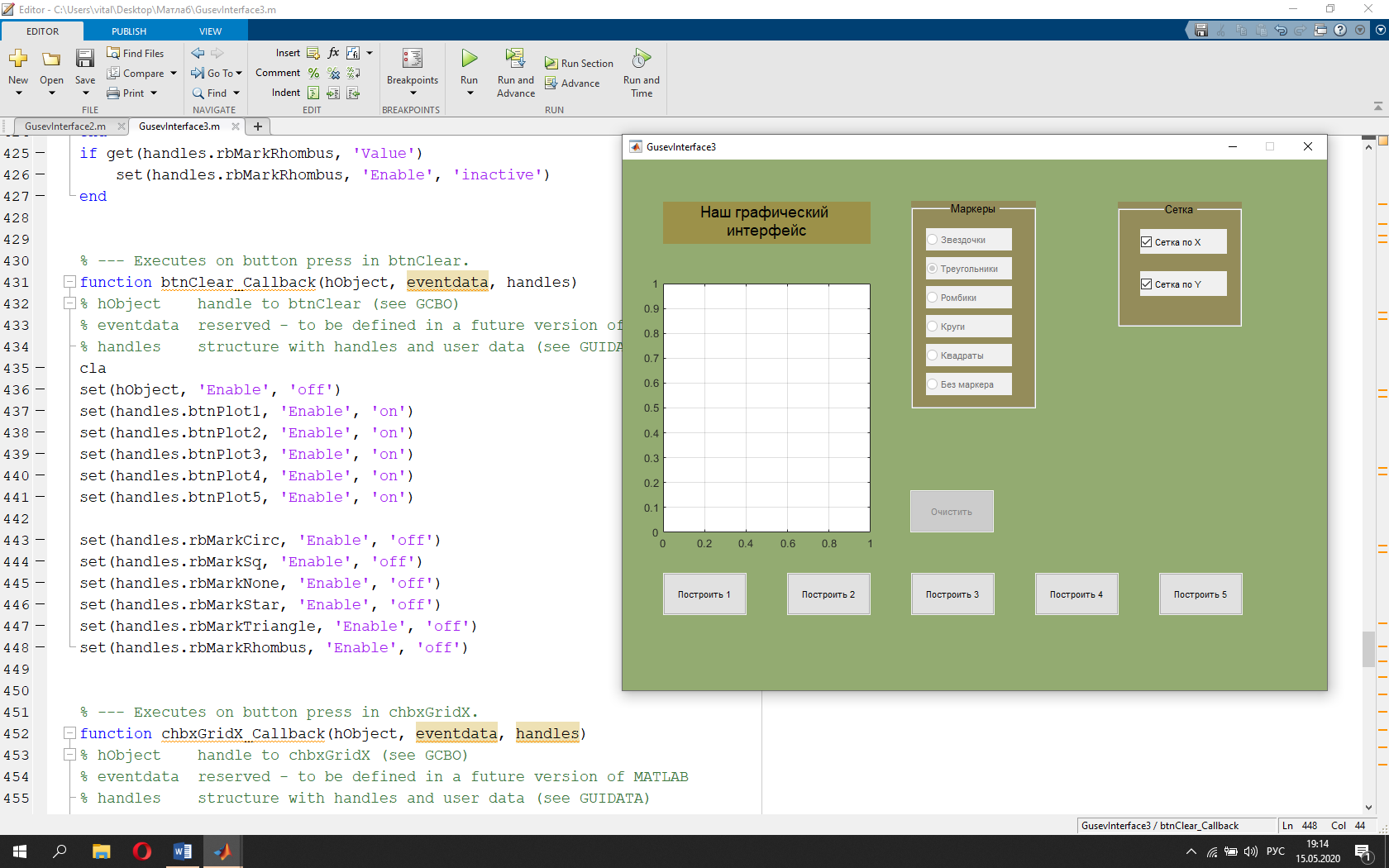
set(handles.rbMarkNone, 'Enable', 'off')

set(handles.rbMarkStar, 'Enable', 'off')

set(handles.rbMarkTriangle, 'Enable', 'off')

set(handles.rbMarkRhombus, 'Enable', 'off')





# Цвет линии

Для изменения цвета линии графика нужно нанести на форму Pop-up Menu и Static Text над ним. У Static Text в поле String написать «Цвет линии». В Pop-up Menu заменить поле String на «Синий, Красный, Зеленый, Желтый, Черный, Маджента и Циан», записывая каждый цвет с новой строки. Новая строка означает новый пункт в Pop-up Menu. Поле Enable нужно заменить на «off», чтобы при запуске к нему нельзя было обратиться, Tag заменить на «pmColor».

Для программирования Pop-up Menu нужно прописать:

Num=get(hObject, 'Value');

switch Num

case 1

set(handles.Line, 'Color', 'b')

case 2

set(handles.Line, 'Color', 'r')

case 3

set(handles.Line, 'Color', 'g')

case 4

set(handles.Line, 'Color', 'y')

case 5

set(handles.Line, 'Color', 'k')

case 6

set(handles.Line, 'Color', 'm')

case 7

set(handles.Line, 'Color', 'c')

end

Переменная Num принимает значение Pop-up Menu. Далее через switch задает цвет линии графика.

Для всех кнопок «Построить» нужно написать код:

set(handles.pmColor, 'Enable', 'on')

Num=get(handles.pmColor, 'Value');

switch Num

case 1

set(handles.Line, 'Color', 'b')

case 2

set(handles.Line, 'Color', 'r')

case 3

set(handles.Line, 'Color', 'g')

case 4

set(handles.Line, 'Color', 'y')

case 5

set(handles.Line, 'Color', 'k')

case 6

set(handles.Line, 'Color', 'm')

case 7

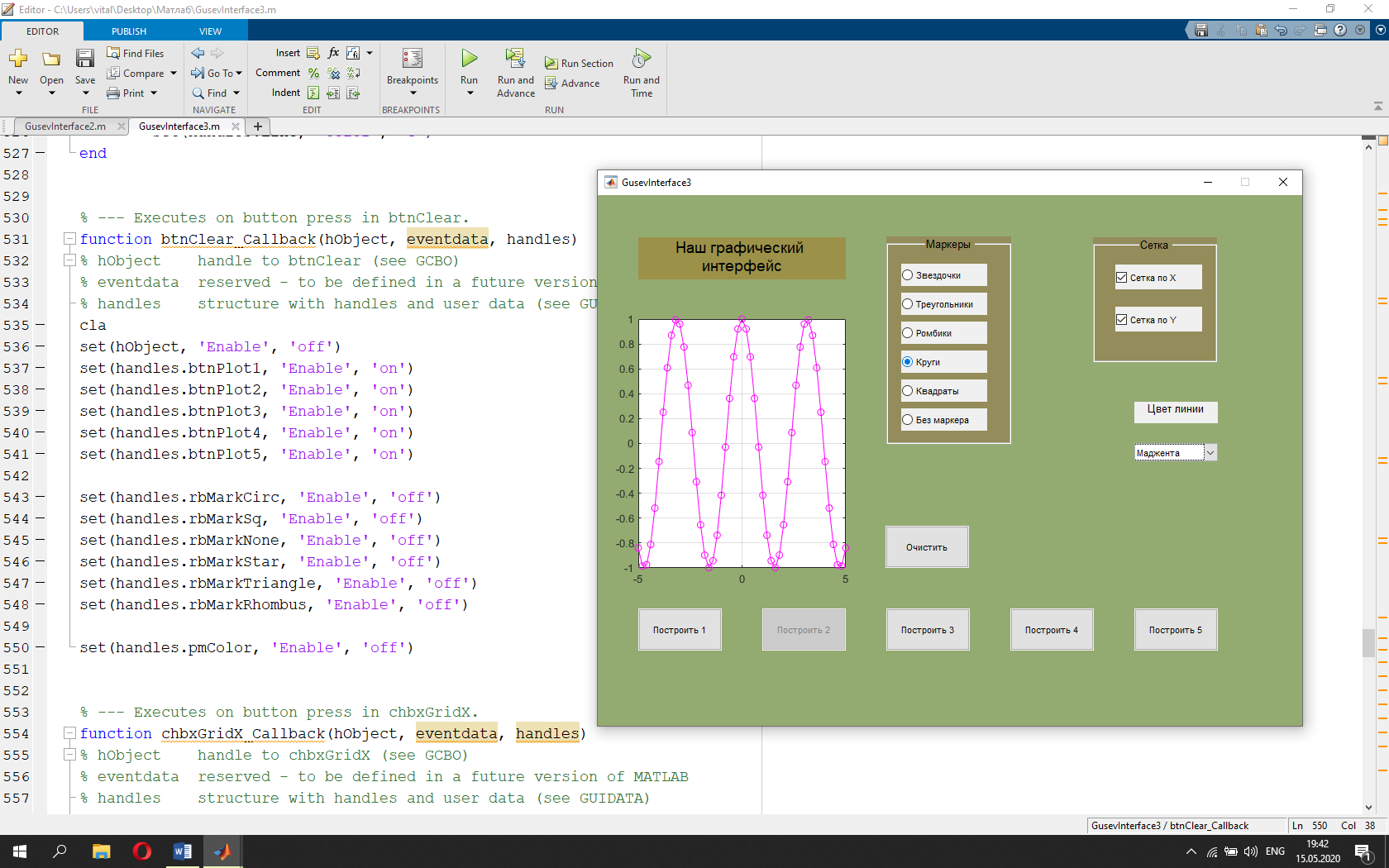
set(handles.Line, 'Color', 'c')

end

Первая функция включает Pop-up Menu, далее через switch задается цвет линии графика.

Для кнопки «Очистить» нужно написать функцию отключения Pop-up Menu.

set(handles.pmColor, 'Enable', 'off')



# Толщина линии

Для изменения толщины линии нужно нанести на форму Slider и выше Static Text. У Static Text поле в String написать «Толщина линии». У Slider поле Enable нужно заменить на «off», Max (максимальное значение Slider) на «8.0», Min (минимальное значение Slider) на «1.0», Tag – «scrWidth» и Value – «1.0».

Программирование Slider. Нужно написать в функцию:

width=get(hObject, 'Value');

set(handles.Line, 'LineWidth', round(width))

Переменная width принимает значение Slider, далее функция присваивает эту ширину линии.

Во всех кнопках «Построить» прописывается:

set(handles.scrWidth, 'Enable', 'on')

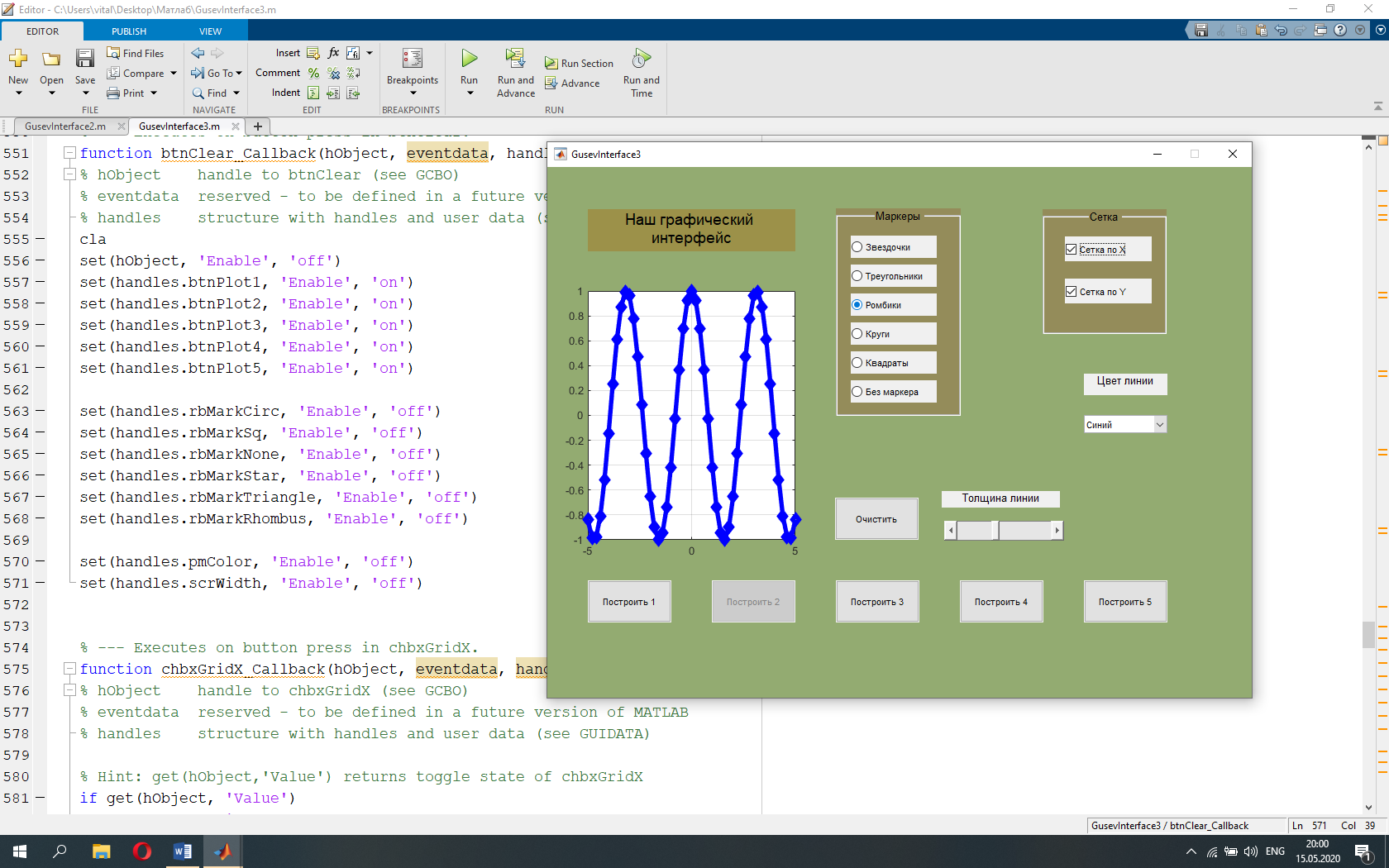
width=get(handles.scrWidth, 'Value');

set(handles.Line, 'LineWidth', round(width))

Первая функция включает Slider. Далее задается ширина линии.

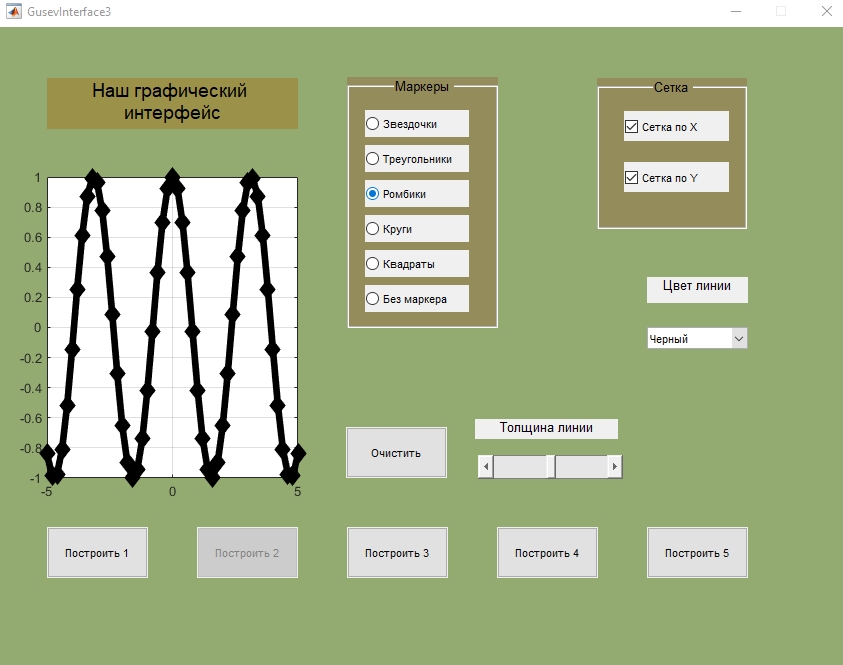
В кнопку «Очистить», чтобы выключить Slider:

set(handles.scrWidth, 'Enable', 'off')



# Заключение

На этом завершается разработка интерфейса. В результате был разработан графический интерфейс в высокоуровневом интерпретируемом языке программирования MATLAB.



# Список использованной литературы

1. Юрий Лазарев «Начала программирования в среде MatLAB».

2. Игорь Ануфриев, Александр Смирнов, Елена Смирнова «MATLAB 7».

# Листинг

function varargout = GusevInterface3(varargin)

gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ...

'gui\_Singleton', gui\_Singleton, ...

'gui\_OpeningFcn', @GusevInterface3\_OpeningFcn, ...

'gui\_OutputFcn', @GusevInterface3\_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

function GusevInterface3\_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;

guidata(hObject, handles);

function varargout = GusevInterface3\_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;

function btnPlot1\_Callback(hObject, eventdata, handles)

x=-5:0.2:5;

y=cos(x);

handles.Line=plot(x,y)

guidata(gcbo, handles)

set(hObject, 'Enable', 'off')

set(handles.btnPlot2, 'Enable', 'on')

set(handles.btnPlot3, 'Enable', 'on')

set(handles.btnPlot4, 'Enable', 'on')

set(handles.btnPlot5, 'Enable', 'on')

set(handles.btnClear, 'Enable', 'on')

set(handles.rbMarkCirc, 'Enable', 'on')

set(handles.rbMarkSq, 'Enable', 'on')

set(handles.rbMarkNone, 'Enable', 'on')

set(handles.rbMarkStar, 'Enable', 'on')

set(handles.rbMarkTriangle, 'Enable', 'on')

set(handles.rbMarkRhombus, 'Enable', 'on')

set(handles.pmColor, 'Enable', 'on')

set(handles.scrWidth, 'Enable', 'on')

if get(handles.chbxGridX, 'Value')

set(gca,'XGrid', 'on')

else

set(gca,'XGrid','off')

end

if get(handles.chbxGridY, 'Value')

set(gca,'YGrid', 'on')

else

set(gca,'YGrid','off')

end

if get(handles.rbMarkCirc, 'Value')

set(handles.Line, 'Marker', 'o')

end

if get(handles.rbMarkSq, 'Value')

set(handles.Line, 'Marker', 's')

end

if get(handles.rbMarkStar, 'Value')

set(handles.Line, 'Marker', '\*')

end

if get(handles.rbMarkTriangle, 'Value')

set(handles.Line, 'Marker', '^')

end

if get(handles.rbMarkRhombus, 'Value')

set(handles.Line, 'Marker', 'd')

end

if get(handles.rbMarkCirc, 'Value')

set(handles.rbMarkCirc, 'Enable', 'inactive')

end

if get(handles.rbMarkSq, 'Value')

set(handles.rbMarkSq, 'Enable', 'inactive')

end

if get(handles.rbMarkNone, 'Value')

set(handles.rbMarkNone, 'Enable', 'inactive')

end

if get(handles.rbMarkStar, 'Value')

set(handles.rbMarkStar, 'Enable', 'inactive')

end

if get(handles.rbMarkTriangle, 'Value')

set(handles.rbMarkTriangle, 'Enable', 'inactive')

end

if get(handles.rbMarkRhombus, 'Value')

set(handles.rbMarkRhombus, 'Enable', 'inactive')

end

Num=get(handles.pmColor, 'Value');

switch Num

case 1

set(handles.Line, 'Color', 'b')

case 2

set(handles.Line, 'Color', 'r')

case 3

set(handles.Line, 'Color', 'g')

case 4

set(handles.Line, 'Color', 'y')

case 5

set(handles.Line, 'Color', 'k')

case 6

set(handles.Line, 'Color', 'm')

case 7

set(handles.Line, 'Color', 'c')

end

width=get(handles.scrWidth, 'Value');

set(handles.Line, 'LineWidth', round(width))

function btnPlot2\_Callback(hObject, eventdata, handles)

function btnPlot3\_Callback(hObject, eventdata, handles)

function btnPlot4\_Callback(hObject, eventdata, handles)

function btnPlot5\_Callback(hObject, eventdata, handles)

function btnClear\_Callback(hObject, eventdata, handles)

cla

set(hObject, 'Enable', 'off')

set(handles.btnPlot1, 'Enable', 'on')

set(handles.btnPlot2, 'Enable', 'on')

set(handles.btnPlot3, 'Enable', 'on')

set(handles.btnPlot4, 'Enable', 'on')

set(handles.btnPlot5, 'Enable', 'on')

set(handles.rbMarkCirc, 'Enable', 'off')

set(handles.rbMarkSq, 'Enable', 'off')

set(handles.rbMarkNone, 'Enable', 'off')

set(handles.rbMarkStar, 'Enable', 'off')

set(handles.rbMarkTriangle, 'Enable', 'off')

set(handles.rbMarkRhombus, 'Enable', 'off')

set(handles.pmColor, 'Enable', 'off')

set(handles.scrWidth, 'Enable', 'off')

function chbxGridX\_Callback(hObject, eventdata, handles)

if get(hObject, 'Value')

set(gca,'XGrid', 'on')

else

set(gca,'XGrid','off')

end

function chbxGridY\_Callback(hObject, eventdata, handles)

function rbMarkCirc\_Callback(hObject, eventdata, handles)

set(handles.Line, 'Marker', 'o')

set(hObject, 'Enable', 'inactive')

set(handles.rbMarkSq, 'Enable', 'on')

set(handles.rbMarkNone, 'Enable', 'on')

set(handles.rbMarkStar, 'Enable', 'on')

set(handles.rbMarkTriangle, 'Enable', 'on')

set(handles.rbMarkRhombus, 'Enable', 'on')

function rbMarkStar\_Callback(hObject, eventdata, handles)

function rbMarkTriangle\_Callback(hObject, eventdata, handles)

function rbMarkRhombus\_Callback(hObject, eventdata, handles)

function rbMarkSq\_Callback(hObject, eventdata, handles)

function rbMarkNone\_Callback(hObject, eventdata, handles)

function pmColor\_Callback(hObject, eventdata, handles)

Num=get(hObject, 'Value');

switch Num

case 1

set(handles.Line, 'Color', 'b')

case 2

set(handles.Line, 'Color', 'r')

case 3

set(handles.Line, 'Color', 'g')

case 4

set(handles.Line, 'Color', 'y')

case 5

set(handles.Line, 'Color', 'k')

case 6

set(handles.Line, 'Color', 'm')

case 7

set(handles.Line, 'Color', 'c')

end

function pmColor\_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function scrWidth\_Callback(hObject, eventdata, handles)

width=get(hObject, 'Value');

set(handles.Line, 'LineWidth', round(width))

function scrWidth\_CreateFcn(hObject, eventdata, handles)

if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor',[.9 .9 .9]);

end