Реализация вычислений с помощью графического интерфейса (GUI) в MatLab

Лабораторная работа № 11

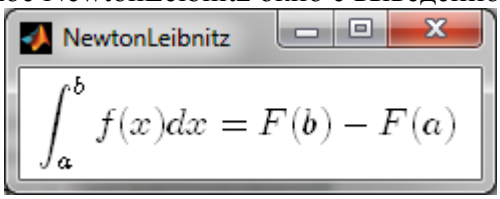
Вывод формулы в графическое окно

Размещение объекта Static Text в окне приложения с графическим интерфейсом пользователя позволяет только вывести простой текст без верхних и нижних индексов и, тем более, без греческих букв или знаков суммы, интеграла. Однако, в MATLAB имеется возможность выводить математические формулы на оси. Для этого на осях размещается текстовый объект, создаваемый при помощи функции text, а сами оси можно сделать невидимыми. Тогда будет казаться, что текстовый объект размещен в самом графическом окне. У текстового объекта есть свойство Interpreter, которое может принимать три значения:

- 'tex' - поддержка тегов TeX;
- ‘latex' - поддержка тегов LaTeX (больше возможностей по набору красивых формул, чем при задании значения tex);
- 'none' - вывод всех символов текстового объекта так, как они заданы, без учета тегов TeX или LaTeX.

При задании свойству Interpreter текстового объекта значений 'tex' или 'latex' можно управлять начертанием символов, т.е. делать курсив, жирный шрифт, изменять размер шрифта.

Создадим графическое NewtonLeibnitz окно с выведенной в него формулой:



Объект содержит следующие графические объекты:

1. графическое окно белого цвета без стандартных меню и панели инструментов (при помощи функции figure);
2. невидимые оси, совпадающие по размеру с графическим окном, (при помощи функции axes);
3. соответствующий текстовый объект с использование интерпретатора LaTeX (при помощи функции text).

Создайте m.файл со следующим кодом:

```
function NewtonLeibnitz
% создание графического окна белого цвета без стандартной строки меню
% и панели инструментов с заголовком NewtonLeibnitz шириной 230 и высотой 55 пикселей
figure('MenuBar','none','Color','w','Name','NewtonLeibnitz',...
    'NumberTitle','off','Position',[400 300 230 55])
% создание невидимых осей, совпадающих по размерам с графическим окном
axes('Position',[0 0 1 1],'Visible','off')
% задание строки с формулой в формате LaTeX
str='$$\int_a^b f(x)dx=F(b)-F(a)$$';
% создание текстового объекта на осях, шрифт 14пт
text('Interpreter','latex','String',str,'FontSize',14,...
    'Units','pixels','Position',[10 30])
```

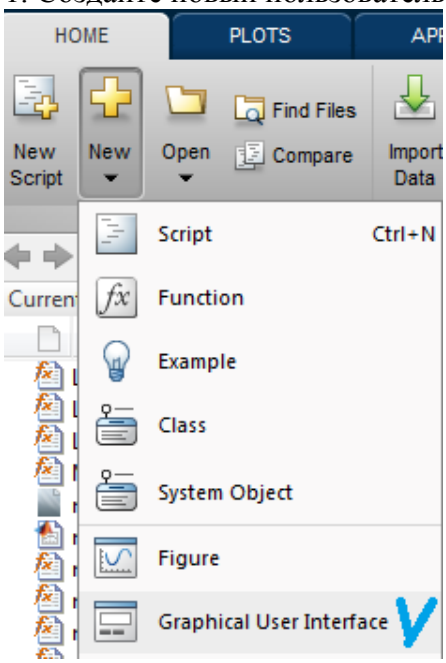
Лабораторная работа № 12

Графическое окно для вычисления сумма двух чисел

Описание работы по созданию GUI описано в мультимедийной справке: см. закладку Help-> Examples->Creating Graphical User Interfaces.

Рассмотрим простейший пример GUI-программы, вычисляющей сумму двух чисел.

1. Создайте новый пользовательский интерфейс.

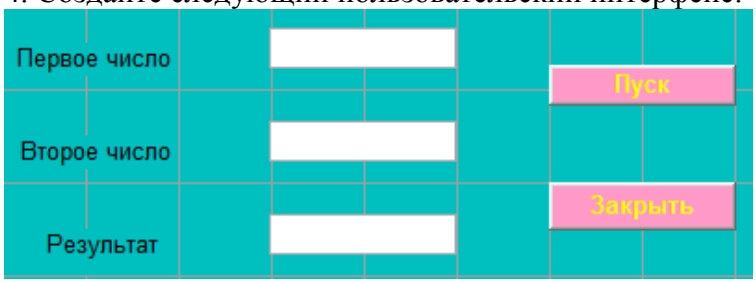


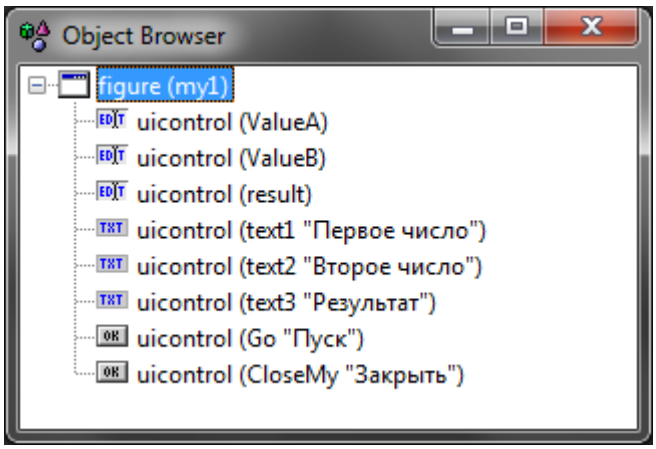
или >> guide

2. Выберите шаблон Blank GUI (Default).

3. Отобразите два окна – Инспектора Свойств и Обозревателя Объектов  .

4. Создайте следующий пользовательский интерфейс:





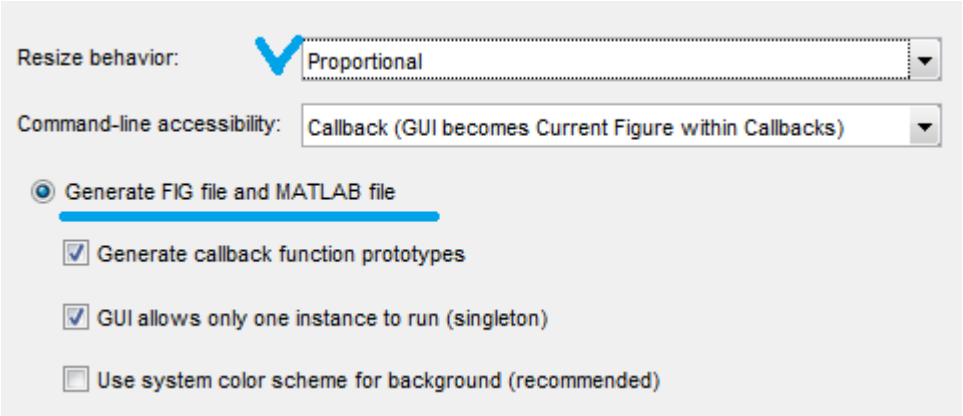
Текстовым полям дайте имена ValueA, ValueB и result (свойство tag).

Кнопки Push Button – «Закрыть» и «Пуск» (свойство String) назовите CloseMy и Go (свойство tag).

Созданное окно тождественно объекту класса figure.

5. Посмотрите опции для генерации шаблона GUI: Tools->GUI Options.

Должно стоять «Сгенерировать и М-файл, и рисуночный файл *.fig».



Выберите параметр масштабируемости окна (proportional).

Сохраните файлы под именем my1, М-файл откроется автоматически.

6. Дополнительно создайте М-файл для вычисления суммы двух чисел, my1exec.m:

```
function q=my1exec(p)
    q=p(1)+p(2);
end
```

7. Отредактируйте код my1.m-файла.

Внесите изменения согласно таблице:

Название процедуры	Добавленный код

`my1_OpeningFcn`	`my.A=0;`
	`my.B=0;`
	`setappdata(hObject,'mydata',my);`

`ValueA_Callback`	`my=getappdata(handles.Summa,'mydata');`
	`my.A=str2double(get(hObject,'String'));`
	`setappdata(handles.Summa,'mydata',my);`

`ValueB_Callback`	`my=getappdata(handles.Summa,'mydata');`
	`my.B=str2double(get(hObject,'String'));`
	`setappdata(handles.Summa,'mydata',my);`

`CloseMy_Callback`	`rmappdata(handles.Summa,'mydata');` `delete(handles.Summa);`

`Go_Callback`	`my=getappdata(handles.Summa,'mydata');`
	`res=my1exec([my.A my.B]);`
	`set(handles.result,'String',num2str(res));`

Все подфункции написаны параллельно, отражая специфику ориентированного на события программирования. Каждому компоненту окна соответствуют две по названию связанные с ним процедуры – Callback и CreateFcn. Вторая функция автоматически вызывается при его создании (в языке Си аналогично «конструктору» объекта при его инициализации), а вторая – при наступлении некоторых событий, с ним связанных. Напомним, что событием в Windows называется нажатие клавиши, щелчок мыши в определенном месте экрана и прочее вызванное внешними причинами (например, пользователем) изменение среды. Функция OpeningFcn, относящаяся к figure целиком, является конструктором окна. Путем промежуточного переприсвоения указателя на нее выводится вовне (OutputFcn).

Важнейшее значение имеет уникальная для конкретного окна GUI-структура, имя которой автоматически генерируется как handles. По сути это объект класса структура,

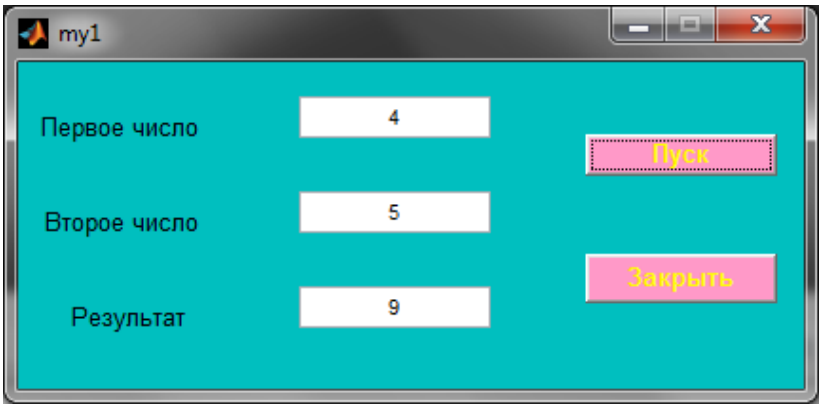
поля которого являются указателями на все дочерние объекты, включая и сам figure. Она передается параметром в каждую процедуру и позволяет программисту изнутри обмениваться данными между дочерними компонентами окна. В нее можно записывать и пользовательские данные; поскольку значения переменных среды регистрируются циклически, то при изменении handles (путем прибавления лишнего поля, не обязательно указателя) простого присвоения недостаточно, а следует применять дополнительно команду guidata.

Ключевой момент в создании GUI-программы – это организация обмена данными между расчетной программой и вызывающей ее оболочкой. В первую очередь для этого нужны те переменные, доступ к которым является общим.

В разбираемом примере реализован путь добавления данных с помощью организации пользовательских данных. Они непосредственно не видны из различных функций (например, Callback-ов), но к ним можно получить доступ парой функций – getappdata, setappdata.

В функции my1_OpeningFcn создается структура my, затем данные в ней копируются/инкапсулируются в пользовательскую структур mydata. Эта структура впредь будет ассоциирована с figure, указатель на которую временно совпадает с hObject, но статически совпадает с handles.Summa. Следующие две строки таблицы описывают изменение пользовательских данных – функция Callback. Триада команд является типической: считывание пользовательских данных (Application Data) в локальную переменную, ее изменение и запись их обратно. Во второй команде по указателю hObject=handles.ValueA(B) мы получаем доступ к тексту, набранному пользователем в боксе, затем преобразуем строку в число. В предпоследней строке таблицы уничтожаются пользовательские данные и закрывается окно командой уничтожения динамической переменной по указателю. Последняя строка таблицы содержит результат.

8. Протестируйте разработанную пользовательскую форму  .



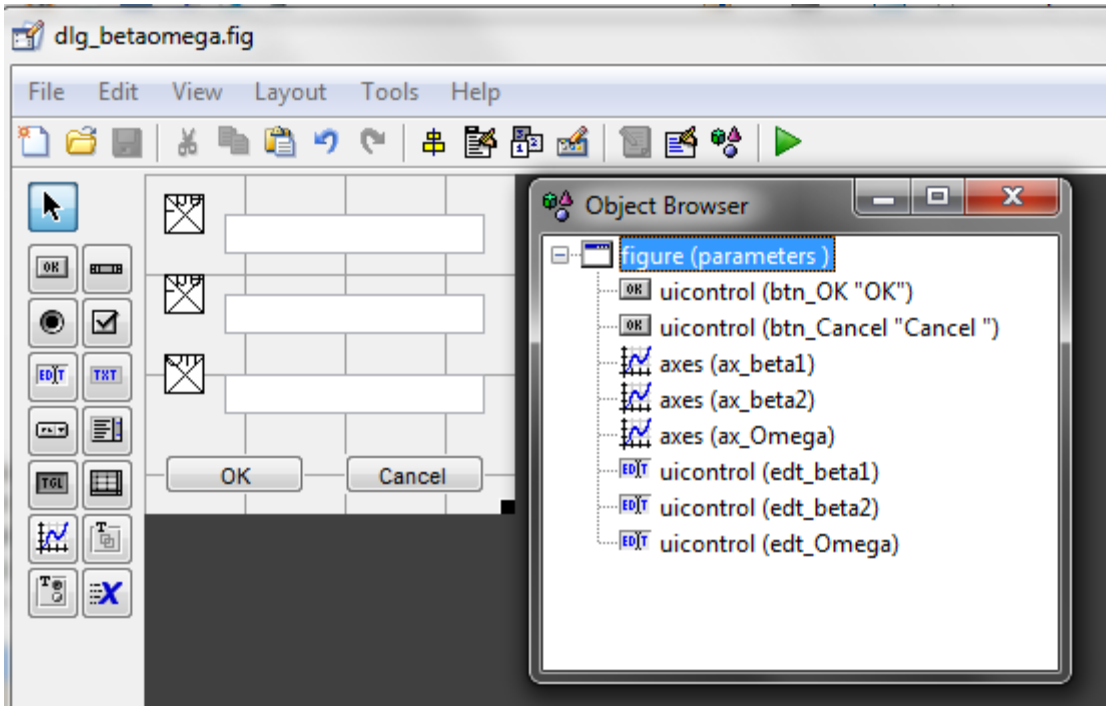
Лабораторная работа № 13

Графическое окно для вызова функции с греческими буквами

Рассмотрим создание диалогового окна `dlg_betaomega.`

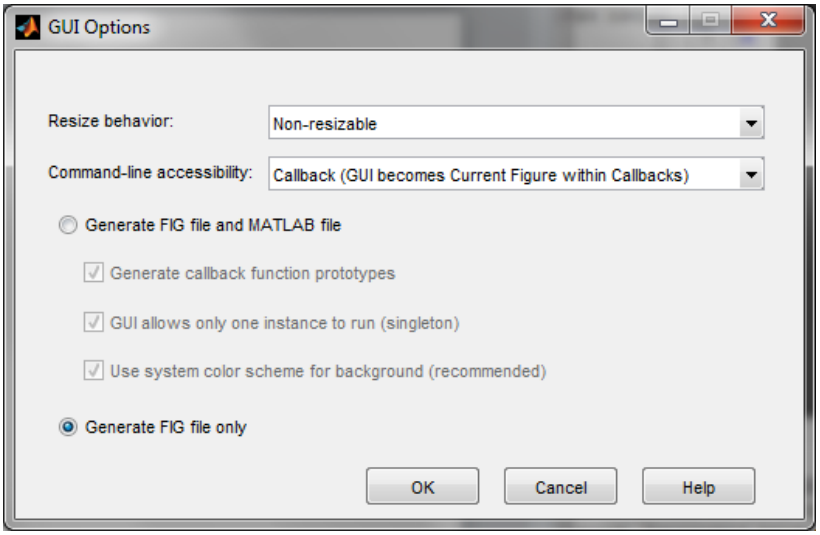
Размещение математических символов в диалоговых окнах, в котором будет три строки ввода для ввода параметров β_1, β_2 и Ω, а также кнопки OK и Cancel.

1. Создайте окно в среде визуального программирования GUIDE и задайте свойства его элементам.



Графическое окно - Tag - win - Units - pixels - Position - [400 300 185 170] - Name - parameters	
Кнопка ОК - Tag - btn_OK - String - OK - Units - pixels - Position - [10 10 70 20]	**Кнопка Cancel** - Tag - btn_Cancel - String - Cancel - Units - pixels - Position - [100 10 70 20]
Верхняя пара осей - Tag - ax_beta1 - Units - pixels - Position - [10 140 20 20] - Visible - off **Нижняя пара осей** - Tag - ax_Omega - Units - pixels - Position - [10 60 20 20] - Visible - off	**Средняя пара осей** - Tag - ax_beta2 - Units - pixels - Position - [10 100 20 20] - Visible - off
Верхняя строка ввода - Tag - edt_beta1 - Units - pixels - Position - [40 130 130 20] - String - пустая строка **Нижняя строка ввода** - Tag - edt_Omega - Units - pixels - Position - [40 50 130 20] - String - пустая строка	**Средняя строка ввода** - Tag - edt_beta2 - Units - pixels - Position - [40 90 130 20] - String - пустая строка

2. Сохраните окно только в одном файле dlg_betaomega.fig.



3. Напишите программный код m-файла dlg_betaomega.m:

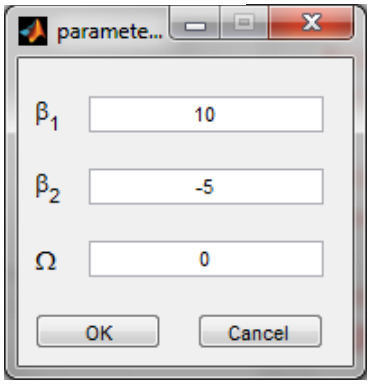
```
function [beta1,beta2,Omega,flag]=dlg_betaomega1
% основная функция, инициализирующая диалоговое окно для ввода
параметров
% beta1, beta2, Omega и возвращающая  их значения
% flag = =1 при закрытии окна кнопкой OK и flag == 0 при другом
способе его закрытия
% открываем диалоговое окно parameters и записываем указатель на него
в H
H=open('dlg_betaomega.fig');
% получаем структуру указателей на объекты окна,
% названия ее полей совпадают с тегами объектов
HANDLES=guihandles(H);
% временно делаем доступными указатели на объекты окна
set(H,'HandleVisibility','on')
% делаем текущими верхние оси
axes(HANDLES.ax_beta1)
% выводим на них текстовый объект с обозначением первого параметра
text('Interpreter','tex','String','\beta_1')
% делаем текущими средние оси
axes(HANDLES.ax_beta2)
% выводим на них текстовый объект с обозначением второго параметра
text('Interpreter','tex','String','\beta_2')
% делаем текущими нижние оси
axes(HANDLES.ax_Omega)
% выводим на них текстовый объект с обозначением третьего параметра
text('Interpreter','tex','String','\Omega')
% возвращаем состояние указателей на объекты окна, теперь они
недоступны
set(H,'HandleVisibility','off')
% с событием Callback кнопки OK связываем подфункцию OK_Callback
set(HANDLES.btn_Cancel,'Callback',@Cancel_Callback)
% с событием Callback кнопки Cancel связываем подфункцию
Cancel_Callback
set(HANDLES.btn_OK,'Callback',@OK_Callback)
% с событием CloseRequestFcn окна связываем подфункцию Cancel_Callback
```

```
set(HANDLES.win,'CloseRequestFcn',@Cancel_Callback)
% ожидаем закрытие диалогового окна
waitfor(H);

function OK_Callback(src,evt)
    % вложенная функция обработки нажатия на кнопку OK
    % присваиваем нужные значения выходным аргументам основной функции
    flag =1;
    beta1str=get(HANDLES.edt_beta1,'String');
    beta2str=get(HANDLES.edt_beta2,'String');
    Omegastr=get(HANDLES.edt_Omega,'String');
    beta1=str2num(beta1str);
    beta2=str2num(beta2str);
    Omega=str2num(Omegastr);
    % удаляем окно приложения
    delete(H)
end

function Cancel_Callback(src,evt)
    % вложенная функция обработки нажатия на кнопку OK
    % присваиваем нужные значения выходным аргументам основной функции
    flag =0;
    beta1=0;
    beta2=0;
    Omega=0;
    % удаляем окно приложения
    delete(H)
end
% конец основной функции
end
```

4. Сохраните dlg_betaomega.m файл.
5. Протестируйте правильность работы функций. В командной строке введите:

```
>> [b1,b2,o,f] = dlg_betaomega
```

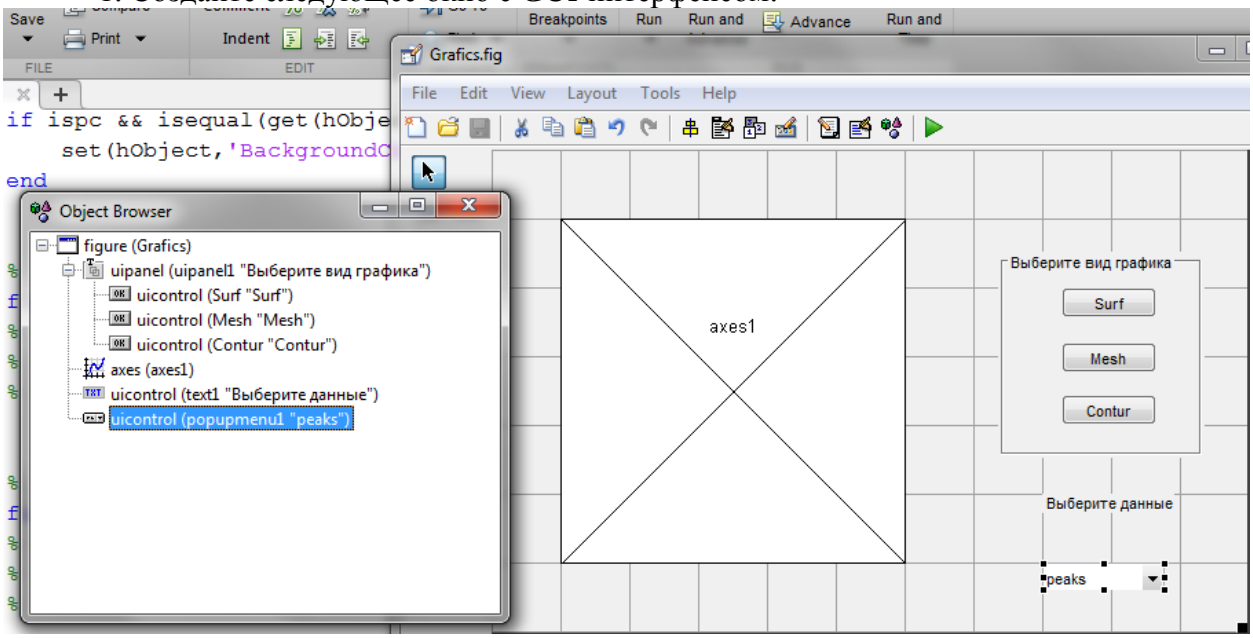
```
b1 =
    10
b2 =
    -5
o =
     0
f =
     1
```



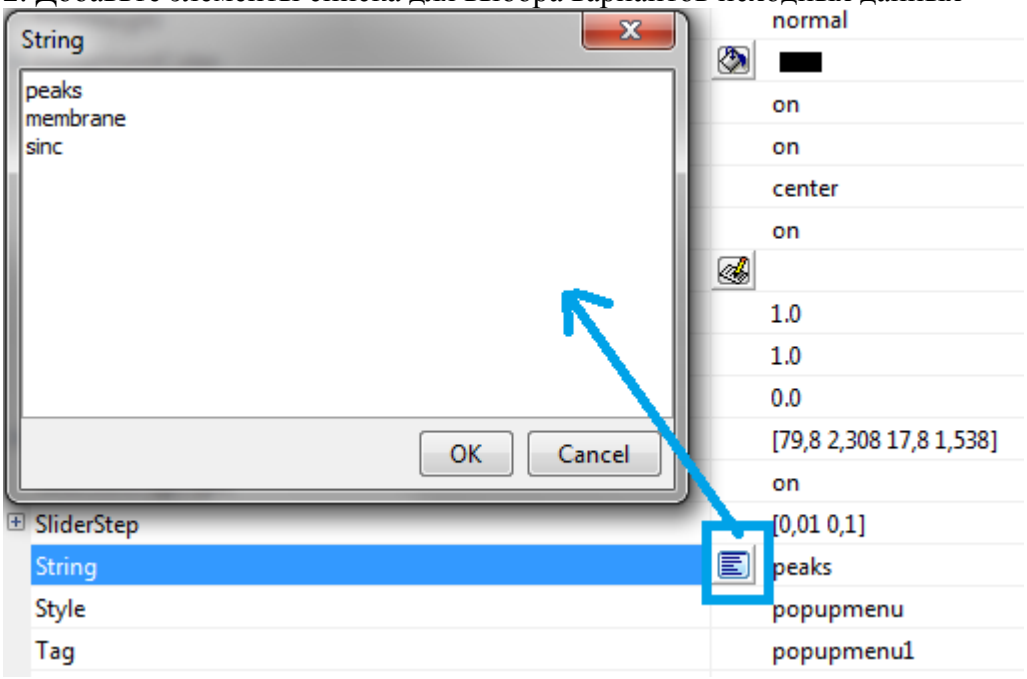
Лабораторная работа № 14

Построение графиков через GUI-интерфейс

1. Создайте следующее окно с GUI-интерфейсом:



2. Добавьте элементы списка для выбора вариантов исходных данных



3. Сохраните fig-файл, создав соответствующий m-файл.

4. Измените код загрузки окна (Выполняется перед тем, как окно становится видимым`% --- Executes just before Grafics is made visible`):

```
function Grafics_OpeningFcn(hObject, eventdata, handles,
varargin)
      handles.peaks=peaks(35);
      handles.membrane=membrane;
      [x,y]=meshgrid(-8:0.5:8);
      r=sqrt(x.^2+y.^2)+eps;
      sinc=sin(r)./r;
      handles.sinc=sinc;
      handles.current_data=handles.peaks;
      surf(handles.current_data);
      % Choose default command line output for Grafics
      handles.output = hObject;
      % Update handles structure
      guidata(hObject, handles);
```

5. Напишите код наступления события по изменению элементов списка:

```
function popupmenu1_Callback(hObject, eventdata, handles)
      val=get(hObject,'Value');
      switch val
       case 1
             handles.current_data=handles.peaks;
       case 2
             handles.current_data=handles.membrane;
        case 3
             handles.current_data=handles.sinc;
      end
      guidata(hObject, handles);
```

6. Самостоятельно продумайте код для кнопок Surf, Mesh и Contur.

7. Протестируйте разработку.