Построение пользовательского интерфейса GUI

Использование интерфейса при программировании в MatLab для согласования различных действий (выполнения процедур, задания входных параметров, выбора необходимых опций, получения результатов в интерактивном режиме в виде графиков или внешних файлов и т.п.) удобно, хотя и требует дополнительных усилий, связанных с GUI программированием.

Поле GUI – Figure.

Родительским объектом для всех элементов управления, которые составляют интерфейс, является figure (они все являются окнами в смысле операционной системы Windows), в его окне и располагаются кнопки, оси-графики, окна для ввода информации и т.д. Мы помним, что figure – это структура, а значит, анализ и редактирование ее свойств происходит соответственно помощью команд set и get заданием значений полям структуры. Дескриптором активного figure является gcf (graphical current figure).

Любой элемент управления, равно как и графический объект более высокого уровня, имеет свойство position, которое определяет размер и положение окна. Значением свойства является вектор, состоящий из четырех элементов, первый и второй – координаты левого нижнего угла предполагаемого элемента, третий и четвертый – длины сторон по горизонтали и вертикали (или вдоль осей x и y). Чтобы соотнести размер figure с экраном компьютера, целесообразно определить его размеры, привязать к произвольному монитору и при программировании это учитывать. Заметим, что родительским объектом для figure является root, т.е. графический объект с нулевым дескриптором.

Важным свойством структуры figure является поле единиц измерения units, по умолчанию оно измеряется в пикселях (pixels), однако для GUI весьма важен режим normalized, который является безразмерным и всегда обеспечивает размещение элементов управления инвариантное от изменения размера поля GUI (изменяемый режим размера). Значение normalized для units может быть выбрано для Root и любого его потомка

Следует помнить, что окно figure, как и все окна, имеет свернутый и развернутый режим; если программист, создавая элементы управления, оставил для units режим по умолчанию, разворачивание окна нарушит гармоничное расположение объектов на поле figure, и в этом случае самым недорогим способом избежать дисгармонии - наложить запрет на разворачивание окна. Далее в примере, предлагаются коды рассмотренных ситуаций.

Пример 1. Подготовка figure

```
% Определение размера текущего монитора:
CSS=get(0,'screensize') % CSS-CurrentScreenSize
% x0,y0,hx,hy – идентификаторы, соответствующие (x0,y0) – левой
```

```
% нижней точке и (hx,hy) – длинам; команда последовательно при-
сваивает значения своих аргументов выходным параметрам:
[x0,y0,hxR,hyR]= deal(CSS(1),CSS(2),CSS(3),CSS(4))
% Запрет разворачивания окна figure
set(gcf,'resize','off')
% Переход к безразмерным единицам измерения:
set(0,'units','normalized') % root
set(gcf,'units','normalized') % figure
% создание нового графического окна:
figure % заметим, что дескриптор figure –натуральное число, на
единицу большее номера последнего созданного объекта figure
% его размер – по умолчанию определяется:
 [xf0,yf0,hxf,hyf]=get(0,'position')
```

Заметим, что в примере 1 (xf0, yf0) – координаты левой нижней точки окна figure на плоскости экрана, (hxf, hyf) – длина и высота этого окна, а в режиме normalized параметры position для root (xf0,yf0,hxf,hyf $\in$[0,1]) не превосходят единицы. Заметим, что выбор безразмерных единиц измерения для figure (set(gcf,'units',' normalized') означает, что координаты всех точек внутри figure , и размеры всех элементов GUI не бут превосходить единицы, такой выбор делает обязательным проектирование всех элементов управления *более низкого уровня* для этого GUI также безразмерными.

Пример 2. Подготовка figure

```
%  удаление всех окон figure, начиная с первого до текуще-
го(активного)
delete(1:gcf);
```

Построение элементов управления.

Прежде чем заняться программированием GUI, научимся строить элементы общего вида, из которого состоит любой интерфейс.

Конструктор для элементов управления общего вида

Построение элементов управления общего вида обеспечивает функция (конструктор) uicontrol, у которой первый параметр – дескриптор родительского окна, далее следуют имена свойств полей и их значения; не принципиальные в момент создания свойства получают значения по умолчанию; построение конкретного типа элемента определяется свойством поля style. Все элементы управления делятся на неизменяемые – статические и изменяемы в процессе диалога – динамические. Динамические элементы обеспечивают передачу информации, ее получение осуществляется посредством обработки события, которое возникает вследствие нажатия кнопки, ввода данных, задания опций-переключателей и т.п.

Edit

Edit - динамический элемент предназначен для ввода и редактирования входной инфор-

мации (числовой и текстовой); текстовые поля с возможностью редактирования создаются командой:

```
he=uicontrol( gcf, 'style', 'edit', 'position', [ x0 y0 Hx Hy],…
                   'BackgroundColor', 'white',…
                   'HorizontalAlignment', 'left', …
                   'string', 0.01 );
```

(многоточие означает продолжение строки), здесь gcf – активное figure - родительский объект GUI (gcf можно заменить идентификатором, например, hf=figure), задан размер окна - 'position', цвет фона – белый, левое выравнивание текста, заметим, что задание поля string обычно опускается, но это целесообразно на этапе построения и отладки GUI, чтобы уже было задано некоторое, например, начальное значение, равное 0.01. Оценить все параметры, в том числе по умолчанию, оставшиеся без объяснений, и узнать их значения можно, используя дескриптор объекта he и команду get (he).

Обработка события:

Поскольку элемент edit является динамическим, нужно уметь интерактивно получать информацию, которую передают с его помощью. Поле 'string' служит для передачи набираемой информации. В нашем случае мы задали число (0.01) по умолчанию, но пользователь может изменить предлагаемую на необходимую ему для диалога. В этом случае, зная дескриптор этого окна – he, мы получим введенное значение, но поскольку свойство поля ‘string’ является строкой, типа char, то для вычислений в программе преобразуем его в числовой формат, если это необходимо:

```
X0=num2str(get(he,’string ’))
```

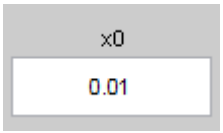
Идентификатор переменной выберем совпадающим с надписью в окне text над этим полем edit, далее построим окно text.

Text

Элемент управления TEXT является статическим, задается как поясняющий текст для EDIT, поясним he, созданный в предыдущем пункте, для этого воспользуемся тем же конструктором

```
ht=uicontrol( gcf, 'style','text','position',[ x0 y0+Hy Hx Hy],…
                   'BackgroundColor', 'white', 'String','x0',…
                   'HorizontalAlignment', 'center' );
```

Обратим внимание, что в position второй элемент равен y0+Hy, т.е. поясняющее (текстовое) окно для значения 0.01 из Edit, которое содержит надпись x0, находится сразу над окном Edit ицвет его фона, как видно, совпадает с цветом фона figure.



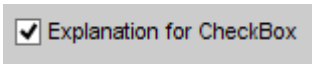
CheckBox

Создадим теперь элемент управления, называемый "флажком" (checkbox):

```
HCb=uicontrol(gcf, 'style', 'checkbox', …
```

```
'position'[x0HCb y0HCb HxHCb HyHCb],…
'string','Explanation for CheckBox', …
'backgroundcolor',get(gcf,'color'));
```

Здесь в string поясняющая надпись, появляющаяся на поле окна, position, задающая его размеры и положение; цвет фона выбран совпадающим с фоном окна figure. Этот элемент управления может пребывать в одном из двух состояний: отмеченном (выбранном) или неотмеченном (невыбранном). В первом случае должна стоять "галочка" (отметка), которую проставляют щелчком левой кнопки мыши на этом элементе управления, во втором - отсутствовать. Повторный щелчок убирает "галочку". Этот элемент управления также является динамическим.



Обработка события:

Система в случае наличия галочки генерирует числовое значение равное единице, в противном случае – нулю. Программист может получить его с как значение свойства поля 'value' элемента HCb:

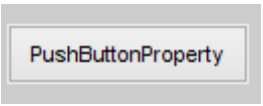
```
Priznak=get (HCb,'value')
```

b воспользоваться при обработке соответствующего условия.

Pushbutton

Кнопка имеет привилегированное место среди остальных элементов управления. Лишь только она создается, ее уже можно нажимать (и неважно, что алгоритм не запускается, потому что этой кнопке не приписано никаких функций); цвет ее фона зафиксирован разработчиками, он неизменяемый. В действительности, в построенном GUI после нажатия кнопки запускается процесс. Создадим кнопку на поверхности конкретного графического окна:

```
hF = figure;
hPb=uicontrol( hF,'style','pushbutton','position',[x0hP y0hP
HxhP HyhP] 'String', 'PushButtonProperty',…
'CallBack','NameFile');
```



PushButtonProperty – надпись на кнопке, которая помогает определить ее назначение; position, как и ранее, размер и расположение окна кнопки. Заметим, что для многооконного интерфейса полями для элементов управления могут быть любые созданные программистом figure, не только с дескриптором gcf.

Обработка события:

Кнопка – динамический объект, и он инициирует действие, происходящее при ее нажатии. Так, свойство поля CallBack может иметь либо строку, которая является именем файла NameFile.m, как в hPb (заметим, расширение *.m не пишется, а добавляется системой). В

случае, когда кнопка предусматривает выполнение нескольких команд, то вместо имени файла может быть идентификатор строки или сама строка, состоящая из набора необходимых действий:

```
'CallBack',StringOfActions);
```

Впоследствии мы приведем пример использования **StringOfActions.**

Listbox

Рассмотрим элемент listbox - список, назначение которого - выбор конкретного элемента из предлагаемого набора. Принимая во внимание конкретные задачи, можно отметить, что это может быть набор реализованных в GUI конкретных методов или свойств различной природы, функций, необходимых программисту и т.п.. В конструкторе перечислены некоторые свойства

```
hL=uicontrol(hF,'style','listbox',…
      'position',[hLx0 hLy0 hLHx hLHy],…
      'string',{'method1','method2','method3'},…
      'HorizontalAlignment', 'left');
```

Понятно, что значением свойства 'style' должно быть 'listbox', а поле структуры 'String' содержит массив ячеек, предназначенный для объединения нескольких строковых значений типа char, объясняющих возможный выбор.

По стрелке или щелчком мыши по списку в GUI выбирается требуемое значение списка, оно выделяется цветом.

Обработка события:

Алгоритм запускается по нажатию кнопки в момент, когда заданы все значения и выбран метод. Система *узнает* о выбранном методе следующим образом:

```
Index=get(hL,'value')
```

здесь Index – индекс выбранного элемента в массиве ячеек из списка, представляющего все методы в GUI, эти методы можно получить из поля string объекта с дескриптором hL

```
Methods=get(hL,'string')
ChoosenMetod= Methods{Index}
```

а выбранный метод определяется по индексу, переданному системой через свойство поля value.

Slider

Этот элемент управления представляет собой движок, который содержит поясняющую надпись и возможность менять положение с помощью движения мыши с зажатой левой

кнопкой, это положение позволяет задать некий масштабирующий множитель (коэффициент), на отрезке [0,1].

```
Hs=uicontrol(gcf,'style','slider',…
                          'position',[ Hsx0 Hsy0 HsHx HsHy], …
                          'HorizontalAlignment','center',…
                          'string','explanation',…
                          'sliderstep',[stepx stepy])
```

В дополнение к уже рассмотренным полям добавлено поле 'sliderstep', которое задает размер самого движка вдоль x и y, перетаскивание его по полю элемента slider влияет на величину масштабирующего коэффициента. Этот элемент является динамическим, изменяющимся, и чтобы получить выбранное значение требуется обработка события.

Обработка события:

Выбранный пользователем коэффициент может быть считан как значение поля value

```
Coefficient=get(Hs,'value')
```

Переменная Coefficient имеет числовую природу, принадлежит классу Double.

Заметим, что все возможные элементы управления могут быть получены как значения поля style для любого элемента управления, например, для последнего Hs, но по команде set:

```
set(Hs)
```

В результате получим такой перечень:

```
Style: [pushbutton | togglebutton | radiobutton | checkbox | edit | text | {slider} | frame | listbox | popupmenu ]
```

Элементы управления, оставшиеся за пределами обсуждения, их создание не является исключительным, они имеют следующие назначения: Togglebutton - для построения toolbars; radiobutton - аналог checkbox, только окошко для индикации – круглое; frame – рамка; popupmenu - создает минименю форму. Справка (help uicontrol) поможет разобраться в деталях.

Замечание 1. Чтобы идентифицировать дескрипторы тех элементов управления, которые имеют одинаковые значения интересующих программиста свойств, например, цвета фона совпадающего с фоном figure, необходимо воспользоваться командой

```
AllDescriptors=findobj(gcf,'background',get(gcf,'Color'))
```

В качестве упражнения мы сможем разом изменить цвет фона у всех дескрипторов:

```
set(AllDescriptors, 'backgroundcolor', 'magenta')
```

Замечание 2. Чтобы объединить по некоторой логике программирования дескрипторы различных элементов управления, которые, например, в определенный момент нужно разом сделать невидимыми (или изменить какое другое свойство), необходимо при создании эти объекты снабдить объединяющим ярлыком с одинаковым именем, свойством

поля tag. Если это не было сделано при создании для некоторых элементов, то можно с помощью set добавить недостающие значения, например, для дескрипторов Hs и hL, а потом изменить свойство поля Visible:

```
set([Hs,hL],'tag','unvisible')

set(findobj(gcf, 'tag','unvisible'), 'visible', 'off')
```

Здесь findobj(gcf, 'tag','unvisible') выбирает все дескрипторы объектов, у которых родительский объект gcf и одинаковый ярлык , unvisible.

Axes

Оси – важный графический объект, который предназначен для визуализации графиков, рисунков и т.п. информации. Для создания осей существует свой конструктор:

```
ha =axes( 'parent', hF, 'color', [ 1 1 1], …
'units', 'points', 'position', [ ax0 ay0 hax hay ], …
                    'FontSize', 8 );
```

Здесь цвет поля осей задан белым, единицы измерения – точки, размер - координатой левой нижней точки и длинами вдоль осей.

Вспомним, что при создании кнопки мы указали возможность задания строки StringOfActions в поле Callback. Примером такой строки может быть следующая:

```
StringOfActions='axes(ha), cla'
```

По нажатию кнопки очистки экрана происходит выполнение строки StringOfActions, состоящей из выбора в качестве активных осей с дескриптором ha и последующей их очисткой по команде cla.

Организация структурных связей в GUI

Головной файл, в котором создаются все элементы управления, следует оформить как процедуру без входных и выходных параметров. Имя функции внутри файла и имя *.m файла должны совпадать. Функция, которая запускается нажатием кнопки, с именем, указанным в Callback, тоже должна быть оформлена как процедура (имя ее и внешнего файла совпадают).

Следует помнить, что те дескрипторы, которые используются в процедурах (только динамические элементы общего вида) необходимо описать в процедуре, указанной в Callback, и головном файле как глобальные. Заметим, что глобальные переменные отделяются пробелом!

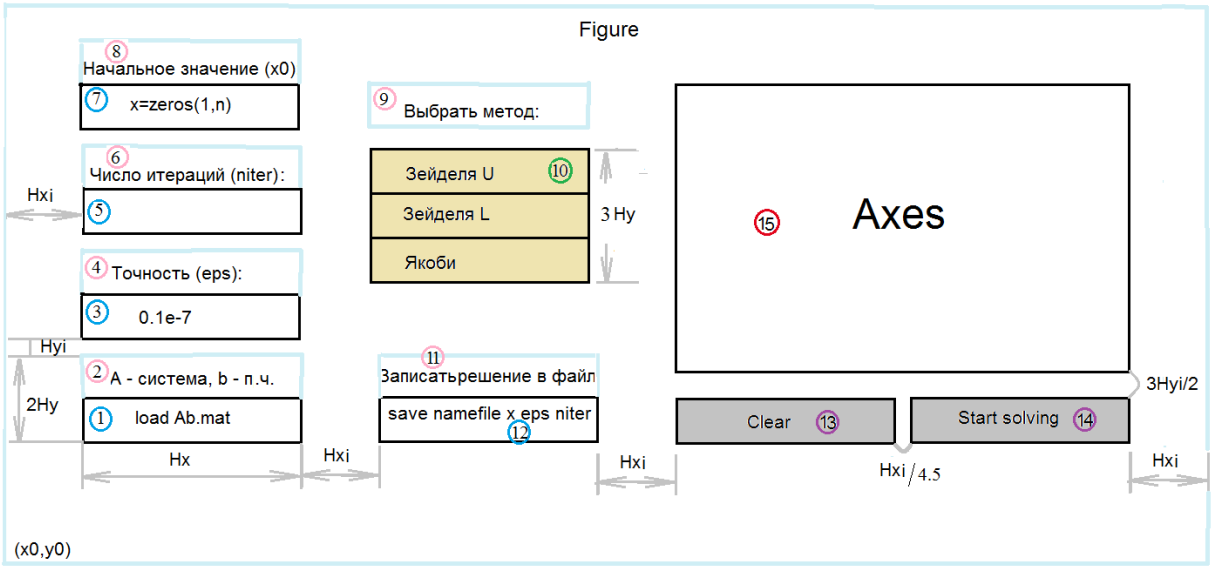
```
Global ha Hs he HCb;
```

Техника построения GUI

Для того чтобы освоить элементы техники построения интерфейса, воспользуемся примером поля, представленного на рисунке 6.1.

Здесь окна 1,3,5,7,12 предназначены для ввода входных параметров, поля 2,4,6,8,9,11 содержат надписи, поясняющие назначение этих полей, заметим, что 10 – список набора переключателей, обеспечивающих выбор метода; 15 – оси для графической интерпретации результатов и две кнопки 13, 14, управляющие процессом решения и очистки осей.

Все элементы управления являются структурами, значения свойств полей регулируют их вид и функциональность. Для простоты изложения мы выберем одинаковые размеры Hx, Hy всех полей за исключением осей, (x0,y0) – координату левого нижнего угла figure и оставим units по умолчанию. Задавая значения Hx, Hxi, легко определить ширину всего figure, равную 4(Hx+Hxi)+Hxi/4.5. Для аналогичных рассуждений о высоте экрана программисту следует определиться с верхним и нижним полями.



Условимся, что уже выбраны значения Hy, Hx, Hxi, Hyi, равно как и точка левого нижнего угла (x0e1, y0e1) первого элемента (edit), заметим, согласно рисунку x0e1 может быть вычислена x0e1=x0+Hxi, если задано x0. Элементы 1-8 можно создать в цикле (пример 3)

Пример 3. Программирование полей GUI

```
% Привяжем поле GUI к текущему монитору и определим его размер:

CSS=get(0,'ScreenSize') % вектор CSS-CurrentScreenSize
% CSS(1),CSS(2),равны единице, т.к. начальная левая нижняя точка
%экрана; соответствующие  CSS(3),CSS(4) – длины экрана вдоль x и
%y;

[x0,y0,hxF,hyF]= deal(CSS(1)+100,CSS(2)+100,CSS(3)/2,CSS(4)/2)%
% --сдвинем нулевые значения для левого нижнего угла figure на
100 % пикселей (единицы измерения по умолчанию) вправо и вверх и
```

```
%последовательно присвоим [x0, y0, hxF, hyF] как входные
%переменные deal; выбираем длины hxF, hyF у figure как половины
%длин экрана

% Создание figure и задание ему выбранных размеров:
figure, set(gcf,'position', [x0,y0,hxF,hyF])
%
% Задание неизменяемых значений переменных, структурирующих
% элементы по полю, пусть выбраны так:
Hxi=50, Hyi=25,
Hx=150, Hy=25
x0e1=Hxi, y0e1=100
y0t1= y0e1+ Hy % абсцисса первого элемента та же, что и второго,
%а ордината отличается на Hy.
x0t1=x0e1

%Создание элементов типа EDIT TEXT:
% задание опорных параметров для построения первого EDIT:
%[x0e1, y0e1, Hx, Hy] %это position для первого EDIT
%[x0t1, y0t1, Hx, Hy] %это position для первого text

%Создание всех значений по умолчанию, предусмотренных  в Edit;
%храним в массиве ячеек StringEdit:
StringEdit={'load Ab.mat', ' 0.1e-7', ' ','x=zeros(1,n)'}

%Создание всех надписей; храним в массиве ячеек StringText:
StringText={'A - система, b - п.ч.', Точность (eps):', 'Число
итераций(niter):', 'Начальное значение (x0)'}

% Построение элементов 1-8:
for k=1:4
he(k)=uicontrol(gcf,'style', 'edit'…
             'position',[x0e1,y0e1+(k-1)*(2*Hy+Hyi),Hx,Hy],…
             'string', StringEdit{k},…
             'backgroundcolor', [1 1 1])
ht(k)= uicontrol(gcf,'style', 'text'…
             'position',[x0t1,y0t1+(k-1)*(2*Hy+Hyi),Hx,Hy],…
             'string', StringText{k},…
             'backgroundcolor', get(gcf,'color'))
end %k
```

Остальные элементы строятся в соответствии с описанием параметров конструкторов, предлагаемых ранее.