How to write a FLW file

Here we will take three examples to show how to edit the FLW file.

Example 1. FLW_Demo

Class `FLW_Demo` is one of the FLW file, which actually do nothing in this function.

```
classdef FLW_Demo<CLW_generic
```

All FLW class is inheriting from class CLW_generic. Class CLW_generic contains the common UI and function that would be used by all the FLW class, which the constructor function `CLW_generic`, `get_option`, `set_option`, `view_Script`, `get_Script`, `header_update` and `GUI_update`.

```
    properties
        % set the type of the FLW class
        % 0 for load (only input);
        % 1 for the function dealing with single dataset (1in-1out)
        % 2 for other functions related to multiple dataset like merge
        %   (Nin-1out, 1in-Nout or Nin-Nout)
        FLW_TYPE=1;
    end
```

In the properties, we define the type of the FLW class, which are determined by the input and output of the function.

- Class FLW_load which load the letwave dataset with on output, will be set `FLW_TYPE=0`.
- For the most of the FLW files, they operate single dataset with one input and one output. We set `FLW_TYPE=1`.
- Other FLW files which are related multiple dataset with N input one output, one input one output, or even N input N output would be set as `FLW_TYPE=2`.

```
    methods
        % the constructor of this class
        function obj = FLW_Demo(tabgp)
            %call the constructor of the superclass
            %CLW_generic(tabgp,fun_name,affix_name,help_str)
            obj@CLW_generic(tabgp,'Demo','demo',...
                'Just make a demo for how to the FLW file.');
            %to be edited...

        end

        %get the parameters setting from the GUI
        function option=get_option(obj)
            option=get_option@CLW_generic(obj);
            %to be edited...
```

```
        end

        %set the GUI via the parameters setting
        function set_option(obj,option)
            set_option@CLW_generic(obj,option);
            %to be edited...

        end

        %get the script for this operation
        %run this function, normally we will get a script
        %with two lines as following
        %      option=struct('affix','demo','is_save',1);
        %      lwdata= FLW_Demo.get_lwdata(lwdata,option);
        function str=get_Script(obj)
            option=get_option(obj);
            stript_slide=[];
            %to be edited...

            str=get_Script@CLW_generic(obj,stript_slide,option);
        end
    end
```

For the methods related to the GUI, there are four functions, `FLW_Demo`, `get_option`, `set_option` and `get_Script`. Since FLW_`Demo` does nothing to the dataset, the four functions are almost empty.

```
    methods (Static = true)
        function header_out= get_header(header_in,option)
            header_out=header_in;
            %to be edited...

            if ~isempty(option.affix)
                header_out.name=[option.affix,' ',header_out.name];
            end
            option.function=mfilename;
            header_out.history(end+1).option=option;
        end

        function lwdata_out=get_lwdata(lwdata_in,varargin)
            option.affix='demo';
            option.is_save=1;
            option=CLW_check_input(option,{'affix','is_save'},varargin);
            header=FLW_Demo.get_header(lwdata_in.header,option);

            data=lwdata_in.data;
            %to be edited...


            lwdata_out.header=header;
            lwdata_out.data=data;
            if option.is_save
                CLW_save(lwdata_out);
```

```
            end
        end
    end
```

There mainly two function related to the operation, which are `get_header` and `get_lwdata`.

- `get_header` just returns the header after the processing.
- `get_lwdata` returns both header and data after the processing.

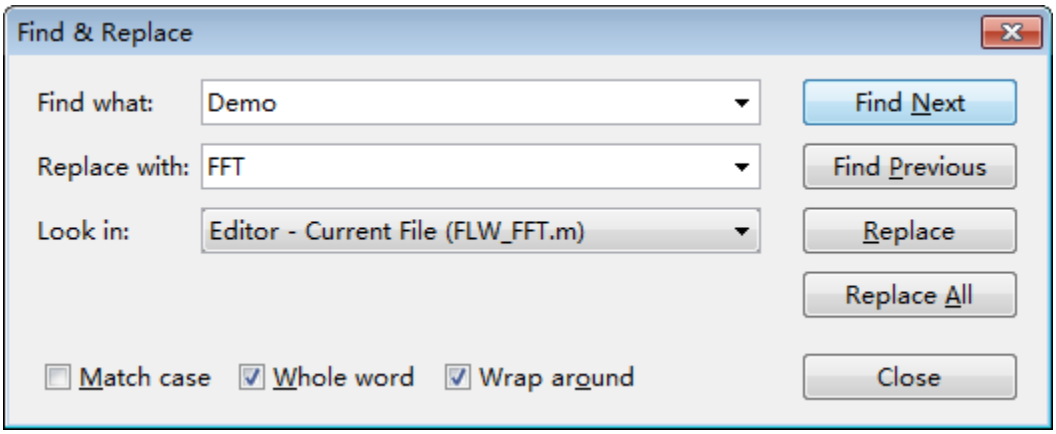
In these two functions, they almost do nothing, just add a prefix and save the dataset.

Example 2. FLW_FFT

Class `FLW_FFT` is used for fast Fourier transform on time domain dataset. We can edit it with the following steps.

1) Copy the class from `FLW_Demo`

`Copy FLW_Demo.m to FLW_FFT.m`. Open it in Matlab, replace all (Ctrl+F) the `demo` to `FFT`.



2) Graphic User Interface (GUI) Design

```
function obj = FLW_FFT(tabgp)
    obj@CLW_generic(tabgp,'FFT','fft',...
        'make the Fast Fourier Transform(fft) for the data.');

    uicontrol('style','text','position',[35,520,200,20],...
        'string','Output','HorizontalAlignment','left',...
        'parent',obj.h_tab);
    obj.h_output_pop=uicontrol('style','popupmenu',...
        'String',{'amplitude','power','phase angle','real part',...
        'imagery part','complex'},'value',1,...
        'position',[35,490,200,30],'parent',obj.h_tab);
    obj.h_half_spectrum_chx=uicontrol('style','checkbox',...
        'String','Output only first half of spectrum','value',1,...
        'position',[35,440,250,30],'parent',obj.h_tab);
    uicontrol('style','text','position',[35,400,200,20],...
        'string','Normalize:','HorizontalAlignment','left',...
```

```
        'parent',obj.h_tab);
    obj.h_normalized_pop=uicontrol('style','popupmenu',...
        'String',{'no normalization','normalized(divided by N)',...
        'normalized(divided by N/2)'},'value',1,...
        'position',[35,370,200,30],'parent',obj.h_tab);
end
```

FLW_FFT makes fast Fourier transform on time domain data, which includes several ways for output, like `amplitude`, `power`, `phase angle`, `real part`, `imagery part` and `complex`. Due to the output is symmetric, the user interface also has an option for output only the first half of the spectrum. Also for normalization, the user interface provides 3 options like `no normalization`, `divided by N`, or `divided by N/2`. Hence, we need a popupmenu for the type of output, a checkbox for choose half of the spectrum or not, and another popupmenu for the selection of normalization. Some other texts are also needed. All these are implemented in the constructor `FLW_FFT`. Since programmatic GUI is used for edit the FLW class, we could not edit the GUIs interactively like GUIDE GUI. Here `FLW_design.m` is used to help the design of the FLW_FFT class in LW_Batch. To make a view of the layout of these uicontrol, you can use `FLW_design.m`. Run it just after changing line 24 into

```
batch{1}=FLW_FFT(handle.tabgp);
```

Remember the uicontrol we used in the following, such as `h_output_pop`, `h_half_spectrum_chx`, `h_normalized_pop`, must be defined in the properties of the class.

```
    properties
        FLW_TYPE=1;

        h_output_pop;
        h_half_spectrum_chx;
        h_normalized_pop;
    end
```

The other three function should be implanted as follows

```
function option=get_option(obj)
    option=get_option@CLW_generic(obj);
    str=get(obj.h_output_pop,'String');
    str_value=get(obj.h_output_pop,'value');
    option.output=str{str_value};
    option.half_spectrum=get(obj.h_half_spectrum_chx,'value');
    option.normalize=get(obj.h_normalized_pop,'value');
end

function set_option(obj,option)
    set_option@CLW_generic(obj,option);
    switch option.output
        case 'amplitude'
            set(obj.h_output_pop,'value',1);
        case 'power'
            set(obj.h_output_pop,'value',2);
        case 'phase angle'
```

```
            set(obj.h_output_pop,'value',3);
        case 'real part'
            set(obj.h_output_pop,'value',4);
        case 'imagery part'
            set(obj.h_output_pop,'value',5);
        case 'complex'
            set(obj.h_output_pop,'value',6);
    end
    set(obj.h_half_spectrum_chx,'value',option.half_spectrum);
    set(obj.h_normalized_pop,'value',option.normalize);
end

function str=get_Script(obj)
    option=get_option(obj);
    stript_slide=[];
    stript_slide=[stript_slide,'''output'',''',...
        option.output,''','];
    stript_slide=[stript_slide,'''half_spectrum'',',...
        num2str(option.half_spectrum),','];
    stript_slide=[stript_slide,'''normalize'',',...
        num2str(option.normalize),','];
    str=get_Script@CLW_generic(obj,stript_slide,option);
end
```

3) function `get_header`, `get_lwdata`

```
function header_out= get_header(header_in,option)
    if ~strcmpi(option.output,'time_amplitude');
        warning('!!! WARNING : input data is not of format
time_amplitude!');
    end
    header_out=header_in;
    header_out.xstart=0;
    header_out.xstep=1/(header_out.xstep*header_out.datasize(6));
    switch option.output
        case 'amplitude'
            header_out.filetype='frequency_amplitude';
        case 'power'
            header_out.filetype='frequency_power';
        case 'phase angle'
            header_out.filetype='frequency_phase';
        case 'real part'
            header_out.filetype='frequency_realpart';
        case 'imagery part'
            header_out.filetype='frequency_imagpart';
        case 'complex'
            header_out.filetype='frequency_complex';
            %used for ifft
            option.events=header_in.events;
            option.xstart=header_in.xstart;
            option.xstep=header_in.xstep;
            option.datasize=header_in.datasize;
    end
    header.events=[];
```

```
    if option.half_spectrum==1
        header_out.datasize(6)=ceil(header_out.datasize(6)/2);
    end

    if ~isempty(option.affix)
        header_out.name=[option.affix,' ',header_out.name];
    end
    option.function=mfilename;
    header_out.history(end+1).option=option;
end

function lwdata_out=get_lwdata(lwdata_in,varargin)
    option.affix='fft';
    option.is_save=1;
    option=CLW_check_input(option,{'output','half_spectrum',...
        'normalize','affix','is_save'},varargin);
    header=FLW_FFT.get_header(lwdata_in.header,option);
    option=header.history(end).option;
    data=fft(lwdata_in.data,[],6);
    switch option.output
        case 'amplitude'
            data=abs(data);
        case 'power'
            data=abs(data).^2;
        case 'phase angle'
            data=angle(data);
        case 'real part'
            data=real(data);
        case 'imagery part'
            data=imag(data);
    end
    switch option.normalize
        case 1
            data=data/size(data,6);
        case 2
            data=data/size(data,6)*2;
    end
    if option.half_spectrum==1
        data=data(:,:,:,:,:,1:ceil(size(data,6)/2));
    end

    lwdata_out.header=header;
    lwdata_out.data=data;
    if option.is_save
        CLW_save(lwdata_out);
    end
end
```

These two functions are implemented the FFT of the dataset.

4) Add FLW_FFT into the menu

Here, we put FLW_FFT into the menu process->frequency domain

```
<menu Label="frequency domain">
        . . . . . .
        . . . . . .
        <submenu Label="FFT" callback="FLW_FFT"/>
        . . . . . .
</menu>
```