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.

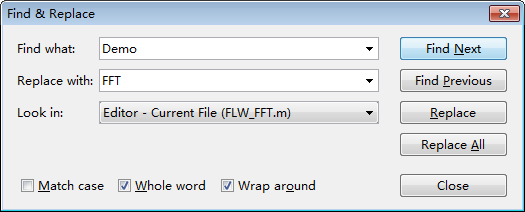
In these two functions, they almost do nothing, just add a prefix and save the dataset.

## 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.



1. 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

1. 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.

1. 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>*