MATLAB Flux Evaluation routines

Table of Contents

I. Installation guidelines

This section is just a brief summary of the different steps needed to installed and run automatically the flux package. For more details on the various steps, refer to the steps' corresponding page number of chapter II, Getting started.

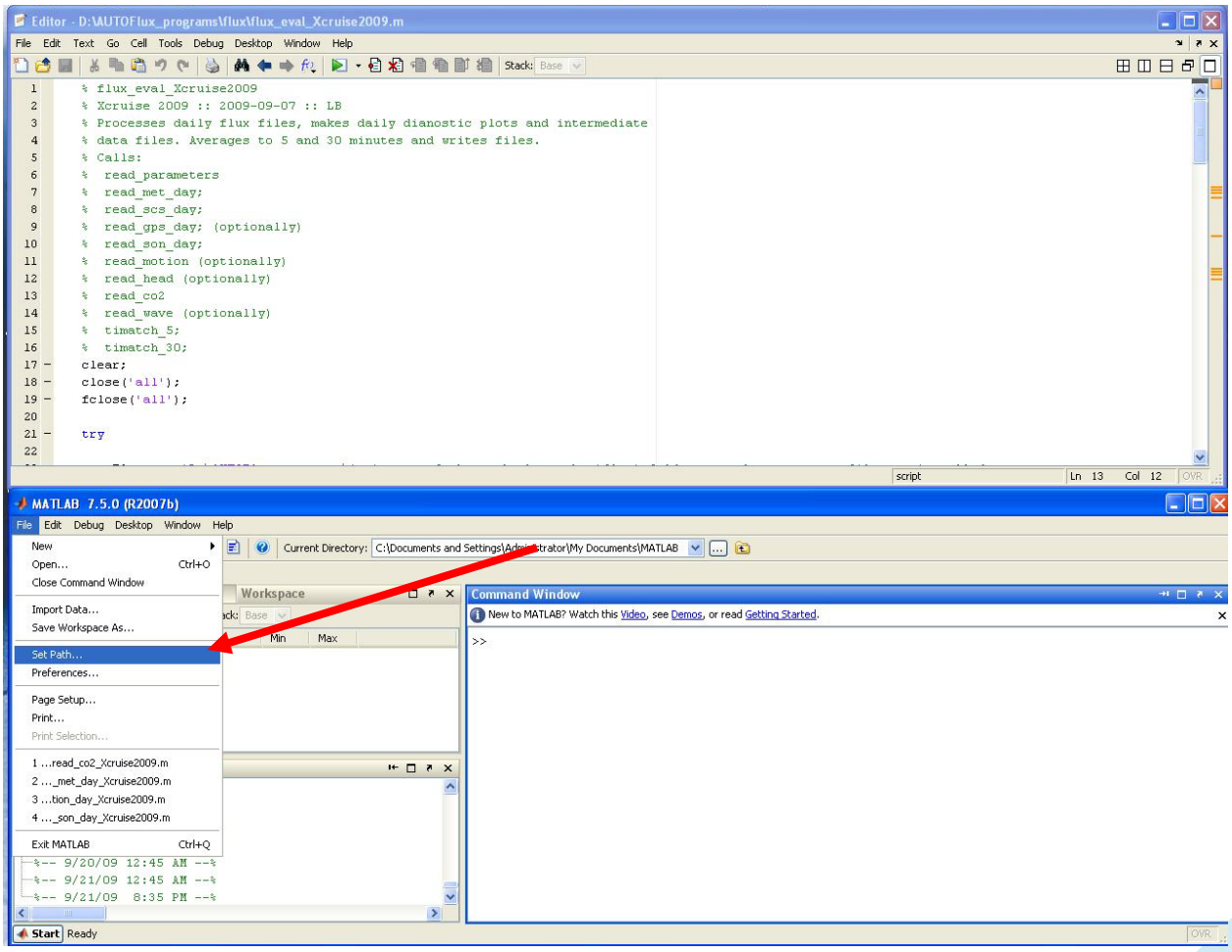
II. Getting started

First, get all the programs and put them in a directory of your choice. For instance, *D:\ AUTOFlux_programs*.

1. Adding folders to the Search Path using the Set Path Dialog Box

Now, start up MATLAB and add the folders to the MATLAB search path. The search path is the list of directories in which MATLAB searched to find M-files. When searching for an M-file, MATLAB will look in these directories in order and uses the first M-file it finds.
To view or change the MATLAB search path, use the Set Path dialog box:

1.1. Open the dialog box by selecting **File** > **Set Path**.

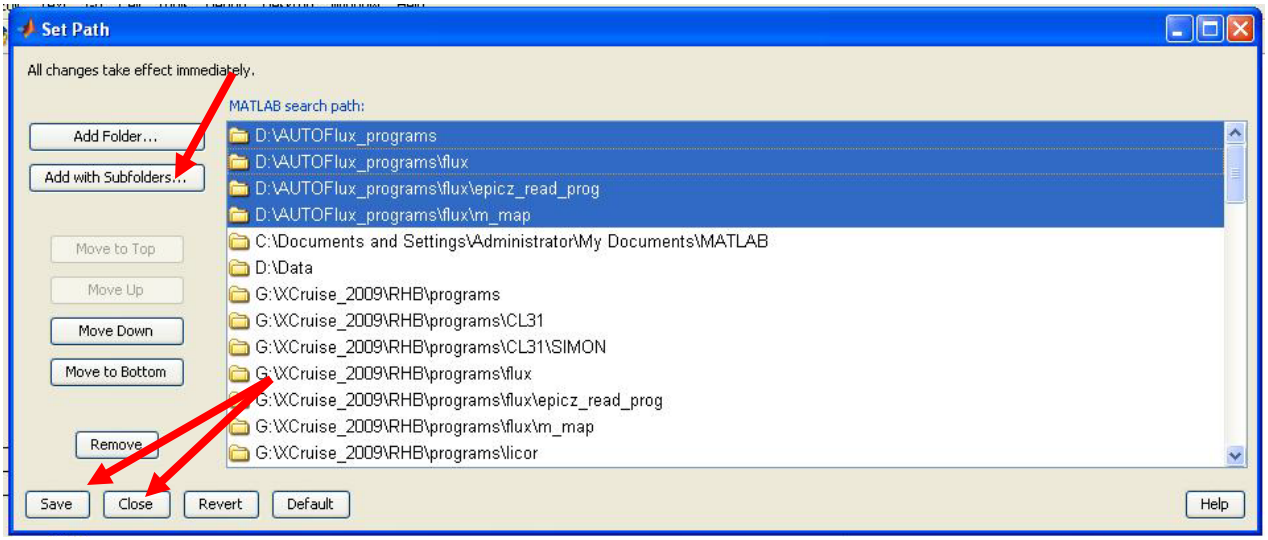


The “Set Path” dialog box opens, listing all folders on the search path.

1.2. Select the option **Add with Subfolders**

1.3. In the resulting Browse for Folder dialog box, select the folder you want to add to the search path (for instance, *D:\AUTOFlux_programs*) and click **ok**.

1.4. You should now see the specified folder with all its subfolders to the top of the search path.



1.5. Finally, click **SAVE** to keep changes for use in the current and future sessions. Then click **Close**.

NOTE: for more details on the search path, type “search path” in the MATLAB help.

2. Flux M-files nomenclature and filename renaming procedure

Once the paths to the routines have been defined, the next step is to rename the programs to match the current research project. The main codes are renamed for every project to make it easier for archiving the programs with the data at the end of the cruise. This also allows tracking down changes within a specific project and helps to prevent modified programs interfering with other projects.

Xcruise_yyyy (*yyyy* being the year) is a generic name that corresponds to the name of the project. This is used as a version qualifier in the naming of the MATLAB scripts of interests, as well as in the cruise database format. Try to respect this arrangement as it might otherwise create conflict

with the processing routines. For instance, rename *flux_eval_Xcruise_yyyy.m* to *flux_eval_STRATUS_2009.m* for a project called STRATUS_2009.

There are quite a few programs to rename. You can either do it manually, or you can use the function *filerename.m* to facilitate the task. Here is how to use that function:

* to rename read*_parameters_Xcruise_yyyy.m* to read_parameters*_STRATUS_2009*.m, type the following in the Matlab prompt:

```
filerename('D:\AUTOFlux_programs\read_parameters_Xcruise_yyyy.m','STRATUS_2009')
```

This will replace the string *Xcruise_yyyy* by *STRATUS_2009*.

* to rename all the M-files containing the string *Xcruise_yyyy*, type:

```
filerename('D:\AUTOFlux_programs\flux\*Xcruise_yyyy*.m','STRATUS_2009')
```

3. Configuring *read_parameters_Xcruise_yyyy.m*

The next step is to specify the configuration used for a particular project. To do so, open the parameter file *read_parameters_Xcruise_yyyy.m* with Matlab editor, and set the appropriate parameters for your project. Note that this script is usually located one directory above the flux programs as it might be used by other scripts (for other instruments).

Here is a configuration example:

```
% ********************************************************************
% ********************************************************************
% ***  Parameter m-file flux_eval_XXX.m.                          ***
% ***  Written by LB on July 06                                   ***
% ***                                                             ***
% ***  Note: - Do not forget to put the character '\' at the end    ***
% ***      of each path!                                             ***
% ***        - Comment style is '%'                                  ***
% ********************************************************************
% ********************************************************************
% Modified by SPdeS 17 October 2008
% Modified by LB 12 June 2009
% Modified by LB 07 September 2009
%
%This routine sets the parameters for the cruise.
```

In this section, you specify the name, the year and the place where the project occurs

```
%################### Cruise informations  ######################################
cruise='STRATUS_2009';     % Cruise name and year
ship='RHB';                % Research vessel
```

Here, you specify the path of the raw data as well as the paths of where to put the processed data and images. This should follow the PSD cruise database format *D:\cruise\ship\instrument* and means that you normally will only have to change the drive letter if necessary.

```
%################### Flux programs and files ###################################
way_raw_data_flux=['D:\' cruise  '\' ship '\flux\Raw\'];      % Raw data path
way_proc_data_flux=['D:\' cruise  '\' ship '\flux\Processed\'];     % Processed
data path
way_spec_data_flux=['D:\' cruise  '\' ship '\flux\Processed\' ];
way_images_flux=['D:\' cruise  '\' ship '\flux\Processed_Images\' ];

%####################%  ceilo files %###################################
way_images_ceilo=['D:\' cruise  '\' ship '\ceilometer\Raw_Images\']; % daily
saved impates
way_raw_data_ceilo=['D:\' cruise  '\' ship '\ceilometer\Raw\'];  % Raw Data
way_proc_data_ceilo=['D:\' cruise  '\' ship '\ceilometer\Processed\']; %
Processed Data

%####################%  Mailbox files %###################################
way_raw_images_mb=['D:\' cruise  '\' ship '\radiometer\Mailbox\Raw_Images\'];
%daily saved images
way_proc_images_mb=['D:\' cruise  '\' ship
'\radiometer\Mailbox\Processed_Images\'];  %processed images
way_raw_data_mb=['D:\' cruise  '\' ship '\radiometer\Mailbox\Raw\'];  % Raw Data
way_proc_data_mb=['D:\' cruise  '\' ship '\radiometer\Mailbox\Processed\']; %
Processed Data

%####################%  balloon files %###################################
way_raw_images_balloon=['D:\' cruise  '\' ship '\balloon\Raw_Images\'];  %daily
saved images
way_proc_images_balloon=['D:\' cruise  '\' ship '\balloon\Processed_Images\'];
%processed images
way_raw_data_balloon=['D:\' cruise  '\' ship '\balloon\Raw\edt\'];  % Raw Data
way_proc_data_balloon=['D:\' cruise  '\' ship '\balloon\Processed\']; %
Processed Data

%####################%  Lacie backup directory###################################
way_backup_data=['G:\' cruise  '\' ship '\'];      % Data back-up path on
external hard disk

%Other paths not in use are commented out (not really necessary, but helps for
clarity
%####################%  mwr files %###################################
% way_raw_images_mwr=['D:\' cruise  '\' ship '\radiometer\90Ghz\Raw_Images\'];
%daily saved images path
% way_raw_data_mwr=['D:\' cruise  '\' ship '\radiometer\90GHz\Raw\Sci\'];
% Raw Data
% way_proc_data_mwr=['D:\' cruise  '\' ship '\radiometer\90GHz\Processed\']; %
Processed Data
```

In the sections below, you specify the instrument configuration, adjustments, limits, post-calibration values, etc… If you need to, you can add your own variables for a particular instrument.
(Do not change unless you know what you are doing)

In here, we define the various temperature adjustments

```
% Adjustments & postcalibrations for flux observations
tsnake_adjustment=0; %degC;
ta_im_adjustment=0;
ta_etl_adjustment=0;
```

This section is where you define the sonic model, its frequency and configuration. Is it rotated 30deg from the main axis? Is the output temperature in degC or is it the speed of sound?

```
%################### Sonic configuration ##################################
sonicmodel='R3';          %Sonic model
fsonic=10;                %frequency (Hz)
rotationsonic=false;      %rotate 30 deg = true
outputsonic=1;      %Sonic output. 0=speed of sound, 1=Temperature (degC)
```

Here, you declare the licor sampling frequency

```
%################### Licor configuration  #################################
flicor=10;            %frequency (Hz)
```

As the title of this section stated, this setup some variables for the clear sky model. Leave as it is if you don't know what to put.

```
%################### Clear sky model configuration ########################
k1=0.05;                %aerosol optical depth, band 1
k2=0.05;                %aerosol optical depth, band 2
oz=0.2;                 %column ozone
iv=3.5;                 %column water vapor (cm), if not calculated from obs.
```

Here, you define the limits of your track plot.

```
%################### Track map limits #####################################
Lonmin=-65;
Lonmax=-50;
Latmin=5;
Latmax=20;
```

Finally, this section declares the height of the main flux instrumentation.

```
%###################  Other parameters  ###################################
zus=18.5;               %wind speed measurement height (m)
zts=15;                 %air T measurement height (m)
zqs=15;                 %air q measurement height (m)
```

The parameter M-file shall be unique for each project and it is entirely customizable. You can add variables that you want to, you can also write some comments at the bottom of the file, for instance what was the configuration of a particular sensor. This M-file will be used in conjunction with the logbook for post-processing and will help document the setup configuration.

4. Editing the MANUAL and AUTO flux programs

Now that you have added the folders to the Matlab Search Path, renamed your files and configured the *read_parameters_Xcruise_yyyy.m*, you are almost good to go! In the flux program folder, there

are two main M-files. *AUTOflux_eval_Xcruise_yyyy.m* is the MATLAB program which runs automatically (once scheduler step II.5. is done), whereas *MANUALflux_eval_Xcruise_yyyy.m* is the code which allows the user to manually look at the data and plots with Matlab. Both programs are very similar and differ slightly due to their usage (see section III.1. and III.2. for description). The final step is then to edit these two main programs *AUTOflux_eval_Xcruise_yyyy.m* and *MANUALflux_eval_Xcruise_yyyy.m*. First, open the files with the Matlab editor.

4.1. The first thing to be changed (in both programs) is the name of the root path for the flux processing package. This is declared in the variable Fluxroot. For instance, if your flux package is in *D:\AUTOFlux_programs*, put: `Fluxroot= D:\AUTOFlux_programs\';  % name of the path where the 'flux' folder + read_parameters file are installed.`
The idea is to make sure that every time *AUTOflux_eval_Xcruise_yyyy.m* runs, the root folder and all its subdirectories are at the top of the search path. This will avoid issues if two files have the same name on the processing computer. That is why the following commands are used:

```
cd([Fluxroot, 'flux'])          %Changes current directory to fluxroot
addpath(genpath(Fluxroot))      %adds flux folder and all of its subdirectories to the top of the MATLAB search path.
```

Note that in Matlab, the current directory is always searched first, before any directories in the search path.

4.2. The second step is to enter the valid name of the *read_parameters_Xcruise_yyyy.m* file. Currently the program is written so that it gets the last uploaded parameter file from the Fluxroot directory. If you know what you're doing, you can manually select your own *read_parameters_Xcruise_yyyy.m* file. Otherwise leave it as it is.

```
%%%%%%%%%%  Read parameters %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     read_parameters_CalNEX_2010;     %reads the parameters for the cruise.
%     currentMname='CalNEX_2010';
currentMname=mfilename;   %Gets the name of currently running M-file to pick
currentMname=currentMname(17:end);

%Gets the last uploaded parameter file from Fluxroot directory
dirfluxroot=dir([Fluxroot 'read_parameters_*.m']);
MostrecentPARfile=dirfluxroot(cellfun(@(x)
x==max([dirfluxroot.datenum]),{dirfluxroot(:).datenum})).name;
disp(['Most recent parameters being used: ' MostrecentPARfile])
eval(strtok(MostrecentPARfile, '.'));
```

Note: If the data format or the filename change, then the program reading these data needs to be updated. Usually this is checked in the lab prior to the experiment, but it's good to keep that in mind if you have weird results…

5. Running *MANUALflux_eval_Xcruise_yyyy*

You are now ready to manually start the flux evaluation routines. From the MATLAB prompt, type the following terminated by the Return key:

MANUALflux_eval_Xcruise_yyyy (you should of course modify this line to match the name of your program, like *MANUALflux_eval_STRATUS_2009* for instance).

The program will ask you for the year and Day-Of-Year you want to look at. The code will take approximately 12 minutes and it will display graphics in MATLAB figure windows (see section III.3. for description of the figures).

6. Adding the flux package to the task scheduler

To activate the *AUTOflux_eval_Xcruise_yyyy.m* M-file, you have to edit the *Run_Flux_Eval.bat* batch file and add it to the task scheduler.

6.1. Editing *Run_Flux_Eval.bat*.

In the same directory as the *flux_eval_Xcruise_yyyy.m* program, there is a batch file called *Run_Flux_Eval.bat*. To edit the batch file, right click on the file and select edit. A notepad window will appear and you should read the following line:

matlab -r AUTOflux_eval_Xcruise_yyyy

This says to run *AUTOflux_eval_Xcruise_yyyy.m* with MATLAB. You should of course modify this line to match the name of your *AUTOflux_eval* program. For instance, matlab -r *AUTOflux_eval_STRATUS_2009* for a project called STRATUS_2009.

6.2. Adding the batch file to the Scheduled Tasks

* First, click Start, click **All Programs**, point to **Accessories**, point to **System Tools**, and then click **Scheduled Tasks**.
* Double-click **Add Scheduled Task** to start the Scheduled Task Wizard, and then click **Next** in the first dialog box.
* The next dialog box displays a list of programs that are installed on your computer, either as part of the Windows XP operating system, or as a result of software installation.
* To run the batch file "Run_Flux_Eval.bat", click **Browse**, click the corresponding folder and file, and then click **Open**.
* Type a name for the task, for instance, run_flux_eval_Xcruise_yyyy, and then choose the **Daily** option.
* Click **Next**, specify the information about the day and time to run the task, for instance 30minutes past midnight. Leave **Every Day** option selected, and then click **Next**.
* Type the name and password of the user who is associated with this task. Make sure that you choose a user with sufficient permissions to run the program. By default, the wizard selects the name of the user who is currently logged on. On PC DAS04, the username is PS3DAS04\Administrator and the corresponding password is spyderman.
* Click **Next**, and then click **Finish** after you verify the choices that you have made.
Note: If you want to change the configuration of the task, open the **Properties** dialog box by doing a right-click on the task, and then click **Properties**.

III. Files description

The flux package routines are designed to monitor the instrument performance through some preliminary processing and display functions. These include computation and display of sensible and latent heats, evaluation and display of the Licor window cleanliness, display of sea surface, air temperature, and relative humidity from both the IMET ship system (if available) and our own sensors. In addition, a few selected time series and Fourier spectra are plotted for evaluation of sensor performance.

1. *MANUALflux_eval_Xcruise_yyyy.m*

Below is a description of the program.

This part gets the year and current day-of-year (doy) from the user, and computes the current date.

```
%%%%%%%%%%%%%%%%%%%%%%%% Gets day-of-year and year %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if ~exist('yyyy','var'); yyyy=input('Input year to evaluate.... ','s'); end;
if ~exist('ddd','var'); ddd=input('Input yearday to evaluate.... '); end;
[m,d]=yd2md(str2num(yyyy), ddd);
Vdate=[str2num(yyyy) m d];
```

See section I.4. for description of the following:

```
%%%%%%%%%%%%%%%%%%%%Modifies current directory and search %%%%%%%%%%%%%%%%%%%%%
Fluxroot='D:\STRATUS_2009\RHB\Scientific_analysis\programs\';  % name of the
path where the 'flux' folder + read_parameters file are installed.
cd([Fluxroot, 'flux'])         %Changes current directory to fluxroot/flux
addpath(genpath(Fluxroot))     %adds flux folder and all of its subdirectories
to the top of the MATLAB search path.

%%%%%%%%%%  Read parameters %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     read_parameters_STRATUS_2009;     %reads the parameters for the cruise.
%     currentMname=' STRATUS_2009';
currentMname=mfilename;   %Gets the name of currently running M-file to pick
currentMname=currentMname(17:end);
%     eval([ 'read_parameters_', currentMname(11:end)])

%Gets the last uploaded parameter file from Fluxroot directory
dirfluxroot=dir([Fluxroot 'read_parameters_*.m']);
MostrecentPARfile=dirfluxroot(cellfun(@(x)
x==max([dirfluxroot.datenum]),{dirfluxroot(:).datenum})).name;
disp(['Most recent parameters being used: ' MostrecentPARfile])
eval(strtok(MostrecentPARfile, '.'));
```

Section below defines some options for the plots and data files. There are also other options about clearing variables, data ftp confirmation, etc…

```
%%%%%%%%%%  Defines some options %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
plotit=true; % for making plots.
prtit=true;  % for printing plots--ineffective unless plotit is also true going
to a temp directory
graphformat='.png';  %select graphics format files
graphdevice='-dpng'; %select graphic device
saveit=true; % for saving data files going to a temp directory
clearit=true;% for clearing data sometimes to save memory
report=[];

data_ftp=false;     %to activate ftp to Boulder server
data_backup=false;  %to activate backup to Lacie hardrive (if present)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Next the raw flux data in *D:\Data\YYDOY* (from FileManager) is copied to *D:\cruise_yyy\ship\flux*.

```
%****   copy raw data folder from D:\Data (on DAS), to way_raw_data_flux
(D:\cruise_yyy\ship\flux) **************
if data_backup
    copyfile(['D:\data\' yyyy(3:4) sprintf('%03i',ddd)],[way_raw_data_flux
yyyy(3:4) sprintf('%03i',ddd)]);
end
```

The next lines read the data, do some processing and some plots.

```
%%%%%%%%%%%%%%%%%%%%%%%%% Starts process %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
eval([ 'read_met_day_', currentMname])
```

read_met_day_Xcruise_yyyy.m : read and plot 1-minute data processed by the Campbell Scientific, create the ascii files *Xcruise_yyyy_proc_met_1min_yyyy_mm_dd_doy.txt* and *Xcruise_yyyy_proc_rad_1min_yyyy_mm_dd_doy.txt*. Data included in the met file are radiometer data, air and sea temperatures, relative humidity, rain gauge, and pressure.

```
eval([ 'read_scs_day_', currentMname])
```

read_scs_day_Xcruise_yyyy.m : when available, read and plot IMET ship data (SCS), and creates a 1-min ascii file *Xcruise_yyyy_proc_scs_1min_yyyy_mm_dd_doy.txt*. Data usually included in the scs file are radiometer data, air and sea temperatures, relative humidity, rain gauge, wind data, and navigation data.

```
eval([ 'read_gps_day_', currentMname])
```

read_gps_day_Xcruise_yyyy.m: read and plot GPS data, mainly latitude/ longitude, Speed-Over-Ground (SOG) and Course-Over-Ground (COG), and the map of the ship track.

```
eval([ 'read_head_pitch_day_', currentMname])
eval([ 'read_head_roll_day_', currentMname])
```

read_head_pitch_day_Xcruise_yyyy.m and *read_head_roll_day_Xcruise_yyyy.m*: when installed, read and plot the Crescent GPS data (differential GPS). This example shows that two systems were installed, thus 2 corresponding M-file. These programs plot the angle (pitch and/or roll) and the ship's heading. It then writes a 1-min ascii file *Xcruise_yyyy_proc_gpsnav_1min_yyyy_mm_dd_doy.txt*.

```
eval([ 'read_son_day_', currentMname])
```

read_son_day_Xcruise_yyyy.m : read and plot sonic anemometer data, and creates a 1-min ascii file *Xcruise_yyyy_proc_son_1min_yyyy_mm_dd_doy.txt*.

```
eval([ 'read_motion_day_', currentMname])
```

read_motion_day_Xcruise_yyyy.m : read and plot motion pack data.

```
licor_prefix='lic';
eval([ 'read_co2_day_', currentMname])
```

read_co2_day_Xcruise_yyyy.m : read and plot LICOR data which includes water vapor and CO2. Also save a 1-min ascii file *Xcruise_yyyy_proc_licor_1min_yyyy_mm_dd_doy.txt*.

```
% eval([ 'read_wave_day_', currentMname])
```

read_wave_day_Xcruise_yyyy.m : If deployed, could be used to read and plot Riegl 10Hz laser range data (wave height) . Currently commented.

```
eval([ 'timatch_5_', currentMname])
```
timatch_5_Xcruise_yyyy.m : Inputs data and computes bulk flux statistics over 5-minute intervals.
```
eval([ 'timatch_30_',currentMname])
```
timatch_30_Xcruise_yyyy.m : Inputs data and computes bulk flux statistics over 30-minute intervals.

The following section computes clear sky shortwave and longwave flux.
```
% compute radiation with model
jjx=min(length(latm),length(qvais));
rlclr=(.52+.13/60*abs(latm(1:jjx))+(.082-
.03/60.*abs(latm(1:jjx))).*sqrt(qvais(1:jjx)')).*(5.67e-
8*(Tvais(1:jjx)+273.15).^4);

clear jdy;
jdys=jd_pc';
latd=nanmean(latm);
lond=nanmean(lonm);
qa=qvais; % humidity needed for solar model
iv=ones(length(qa),1)*NaN;

% solar model
eval([currentMname,'_Rsclear_1' ])
```
Xcruise_yyyy_Rsclear_1.m : computes clear sky solar flux

Below are some final plots made and saved in the image path `way_images_flux`. You could add your own plots if you wanted to.

```
%%%%%%%%%%%%%%%%%%%%%%%%% Do plots  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if plotit
    if ~isempty(Rhvais(~isnan(Rhvais))) && exist('rhm','var')
        figure;plot(jd_pc,Rhvais,'b',jd_scs,rhm,'.g');axis([ddd ddd+1 0 100])
        title(sprintf('%s (%04i-%02i-%02i, DOY%03i).  PSD/SCS relative
        humidity',cruise,Vdate(1),Vdate(2),Vdate(3),ddd),'FontWeight','Bold','I
        nterpreter','none')
        xlabel('Hour (UTC)');ylabel('Relative Humidity (%)')
        legend('PSD', 'SCS','Location','Best');
        set(gca(gcf),'XTick',ddd:2/24:ddd+1);datetick('x','HH:MM','keepticks');
        xax=get(gca(gcf),'XTickLabel');xax(end,:)='24:00';set(gca(gcf),'XTickLa
        bel',xax(:,1:2))
        grid
        annotation(gcf,'textbox',[0.007154 0.01077 0.4498
        0.02462],'String',{'NOAA/ESRL/PSD/Weather & Climate
        Physics'},'FontSize',6,'FitBoxToText','off','LineStyle','none');
        if prtit; print(graphdevice,[way_images_flux cruise '_PSD-
        SCS_RH_comparison_'
        sprintf('%04i_%02i_%02i_%03i',Vdate(1),Vdate(2),Vdate(3),ddd)
        graphformat]);xlim([ddd ddd+1]); end
    end;

    if ~isempty(Tsea(~isnan(Tsea))) && exist('tsgm','var')
        figure;plot(jd_pc,Tsea,'b',jd_scs,tsgm,'.g');xlim([ddd ddd+1])
        title(sprintf('%s (%04i-%02i-%02i, DOY%03i).  PSD/SCS sea surface
        temperature',cruise,Vdate(1),Vdate(2),Vdate(3),ddd),'FontWeight','Bold'
        ,'Interpreter','none')
```

```
    xlabel('Hour (UTC)');ylabel('Sea Temperature (degC)')
    legend('PSD', 'SCS','Location','Best');
    set(gca(gcf),'XTick',ddd:2/24:ddd+1);datetick('x','HH:MM','keepticks');
    xax=get(gca(gcf),'XTickLabel');xax(end,:)='24:00';set(gca(gcf),'XTickLa
    bel',xax(:,1:2))
    grid
    annotation(gcf,'textbox',[0.007154 0.01077 0.4498
    0.02462],'String',{'NOAA/ESRL/PSD/Weather & Climate
    Physics'},'FontSize',6,'FitBoxToText','off','LineStyle','none');
    if prtit; print(graphdevice,[way_images_flux cruise '_PSD-
    SCS_Tsea_comparison_'
    sprintf('%04i_%02i_%02i_%03i',Vdate(1),Vdate(2),Vdate(3),ddd)
    graphformat]);xlim([ddd ddd+1]); end
end;

if ~isempty(Tvais(~isnan(Tvais))) && exist('tam','var')
    figure;plot(jd_pc,Tvais,'b',jd_scs,tam,'.g');xlim([ddd ddd+1])
    title(sprintf('%s (%04i-%02i-%02i, DOY%03i).  PSD/SCS air
    temperature',cruise,Vdate(1),Vdate(2),Vdate(3),ddd),'FontWeight','Bold'
    ,'Interpreter','none')
    xlabel('Hour (UTC)');ylabel('Air Temperature (degC)')
    legend('PSD','SCS','Location','Best');
    set(gca(gcf),'XTick',ddd:2/24:ddd+1);datetick('x','HH:MM','keepticks');
    xax=get(gca(gcf),'XTickLabel');xax(end,:)='24:00';set(gca(gcf),'XTickLa
    bel',xax(:,1:2))
    grid
    annotation(gcf,'textbox',[0.007154 0.01077 0.4498
    0.02462],'String',{'NOAA/ESRL/PSD/Weather & Climate
    Physics'},'FontSize',6,'FitBoxToText','off','LineStyle','none');
    if prtit; print(graphdevice,[way_images_flux cruise '_PSD-
    SCS_Tair_comparison_'
    sprintf('%04i_%02i_%02i_%03i',Vdate(1),Vdate(2),Vdate(3),ddd)
    graphformat]);xlim([ddd ddd+1]); end
end;

if ~isempty(rlclr(~isnan(rlclr))) && exist('irm','var')
    ii=find(~isnan(rlclr));

    figure;plot(jd_pc,pirm,'b',jd_scs,irm,'g',jd_scs(ii),rlclr(ii),'r');xli
    m([ddd ddd+1])
    title(sprintf('%s (%04i-%02i-%02i, DOY%03i).  PSD/SCS infrared
    flux',cruise,Vdate(1),Vdate(2),Vdate(3),ddd),'FontWeight','Bold','Inter
    preter','none')
    xlabel('Hour (UTC)');ylabel('IR Flux (W/m^2)')
    legend('PSD', 'SCS', 'Clearsky','Location','Best');
    set(gca(gcf),'XTick',ddd:2/24:ddd+1);datetick('x','HH:MM','keepticks');
    xax=get(gca(gcf),'XTickLabel');xax(end,:)='24:00';set(gca(gcf),'XTickLa
    bel',xax(:,1:2))
    grid
    annotation(gcf,'textbox',[0.007154 0.01077 0.4498
    0.02462],'String',{'NOAA/ESRL/PSD/Weather & Climate
    Physics'},'FontSize',6,'FitBoxToText','off','LineStyle','none');
    if prtit; print(graphdevice,[way_images_flux cruise '_PSD-
    SCS_IRflux_comparison_'
    sprintf('%04i_%02i_%02i_%03i',Vdate(1),Vdate(2),Vdate(3),ddd)
    graphformat]);xlim([ddd ddd+1]); end
```

```
    end;

    if ~isempty(Rscl(~isnan(Rscl))) && exist('solarm','var')
        ii=find(~isnan(Rscl));

        figure;plot(jd_pc,pspm,'.b',jd_scs,solarm,'.g',jd_pc(ii),Rscl(ii),'r');
        xlim([ddd ddd+1])
        title(sprintf('%s (%04i-%02i-%02i, DOY%03i).  PSD/SCS solar
        flux',cruise,Vdate(1),Vdate(2),Vdate(3),ddd),'FontWeight','Bold','Inter
        preter','none')
        xlabel('Hour (UTC)');ylabel('Solar Flux (W/m^2)')
        legend('PSD', 'SCS','Clearsky','Location','Best');
        set(gca(gcf),'XTick',ddd:2/24:ddd+1);datetick('x','HH:MM','keepticks');
        xax=get(gca(gcf),'XTickLabel');xax(end,:)='24:00';set(gca(gcf),'XTickLa
        bel',xax(:,1:2))
        grid
        annotation(gcf,'textbox',[0.007154 0.01077 0.4498
        0.02462],'String',{'NOAA/ESRL/PSD/Weather & Climate
        Physics'},'FontSize',6,'FitBoxToText','off','LineStyle','none');
        if prtit; print(graphdevice,[way_images_flux cruise '_PSD-
        SCS_Solarflux_comparison_'
        sprintf('%04i_%02i_%02i_%03i',Vdate(1),Vdate(2),Vdate(3),ddd)
        graphformat]);xlim([ddd ddd+1]); end
    end
end % if plotit
disp(['END of MANUALflux_eval_' currentMname ' yearday ' jd '.']);
```

If selected in the options, will backup flux data to external hard drive (if available).

```
%%%%%%%%%%%%%%%%%%%%%%%% backup raw and processed data to external Lacie hard
drive %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if data_backup
    backup_to_Lacie;
end
```

Finally, if selected in the options, the ascii data files, plots, and log files are zipped and sent to PSD ftp/incoming.

```
%%%%%%%%%%%%%%%%%%%%%%%% ftp selected data to our server %%%%%%%%%%%%%%%%%%%%%%%%
if data_ftp
    ftp_to_Boulder_v3;
end
```

Note: You can run a diagnostic program individually for a particular instrument. For instance, to check the sonic data for day-of-year 169, at the MATLAB prompt, type:

```
ddd=169
read_parameters_Xcruise_yyyy
plotit=true; prtit=true;  graphformat='.png'; graphdevice='-dpng';  saveit=true;
clearit=true
read_son_day_Xcruise_yyyy.m
```

2. *AUTOflux_eval_Xcruise_yyyy.m*

This program is basically almost the same code as *MANUALflux_eval_Xcruise_yyyy.m* except some processes are performed differently. For instance, this script gets the current date, compute the current day-of-year (doy) and runs the process for doy-1 (previous day)

```
%%%%%%%%%%%%%%%%%%%%%%%% Gets day-of-year, and year %%%%%%%%%%%%%%%%%%%%%%%%
dv = clock;  % gets current date and time as date vector.
Vdate=datevec(datenum(dv(1),dv(2),dv(3)-1));
ddd=datenum(Vdate)-datenum(Vdate(1),1,0);  %same as md2yd(V(1),V(2),V(3))
yyyy=sprintf('%04i',Vdate(1));
```

All the options are set to true in order to perform data backup and ftp.

```
%%%%%%%%%%  Defines some options %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
plotit=true; % for making plots.
prtit=true;  % for printing plots--ineffective unless plotit is also true
graphformat='.png';  %select graphics format files
graphdevice='-dpng'; %select graphic device
saveit=true; % for saving data files
clearit=true;% for clearing data sometimes to save memory
report=[];
data_ftp=true;     %to activate ftp to Boulder server
data_backup=true;  %to activate backup to Lacie hardrive (if present)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Then in the same fashion as *MANUALflux_eval_Xcruise_yyyy.m*, it reads the data, do some process and some plots.

Finally, the ascii data files, plots, and log files are zipped and sent to PSD ftp/incoming.

```
%%%%%%%%%%%%%%%%%%%%%%%% ftp selected data to our server %%%%%%%%%%%%%%%%%%%%%%%%
if data_ftp
   ftp_to_Boulder_v3;
end
```

The last section just creates a log file if an error occurs between try and catch command. This will help diagnose possible problems

```
%%%%%%%%%%%%%%%%%%%%%%%% catch error if any, and creates log report %%%%%%%%%%
catch fluxstatus1
    report = getReport(fluxstatus1);
    fid=fopen([way_proc_data_flux 'logerror' sprintf('%03i',ddd)  '.txt'],'w');
    fprintf(fid,'%s',report);
    fclose(fid);
    ftp_to_Boulder_v3;
end
```

At the end, the *exit* command terminates the current session of MATLAB.

3. *Additional MATLAB scripts of possible value*

Some additional MATLAB scripts which should be of value.

filecat.m : Use that function to concatenate all the files from the a specified source to a destination file. For instance, to concatenate all *proc_pcddd.txt* files into one unique file, type in the Matlab command: `filecat('your directory\proc_pc*.txt ','proc_pc_Xcruise_yyyy.txt')`
Type help filecat for more information on that function

epicz_read_Xcruise_yyyy.m : If you want to look at some composite plots for your entire project (to add to a report for example), you can use this script to produce composite plots. Similarly to *flux_eval_Xcruise_yyyy.m*, *epicz_read_Xcruise_yyyy.m* uses the *read_parameters_Xcruise_yyyy.m* script. For that reason, edit the program and enter the valid name for the *read_parameters_Xcruise_yyyy.m* file. The Fluxroot variable needs to be changes as well (as in section II.4.)

4. MATLAB figure windows description

The program *MANUALflux_eval_Xcruise_yyyy.m* will take approximately 12 minutes to run and it will display the following information in the MATLAB figure windows.

Figure1: Barometric pressure from Vaisala
Figure2: Temperatures (Tair_psd, Tseasnake, Case and Dome)
Figure3: Solar downwelling flux
Figure4: IR downwelling flux
Figure5: Relative humidity from the Vaisala
Figure6: Rain rate from Optical Raingauge
Figure7: Aspirator backflow indicator
Figure8: Optical Raingauge Function (VDC)
Figure9: Sea snake temperature

Note that more figures can be displayed if other systems are present, like the SCS or wave instruments for instance.

The figures can be paged through by pressing the 'Alt' key to the left or right of the space bar, simultaneously with the 'Tab' key. Repeated pressing of 'Alt'-'Tab' will page through the individual plots. Note that the user can reverse the paging by pressing the 'Shift' key with the 'Alt' and 'Tab' keys.

These plots are intended to be utilized by the user as diagnostic tools to periodically investigate data integrity. For example, the turbulence spectra (multiplied by frequency) should show a characteristic (-2/3) slope in the inertial-subrange frequencies, and a time series of ship motion variables will show oscillations corresponding to the ship's response to wave forcing. Furthermore, comparison of variables such as temperature with similar instruments will be helpful in diagnosing problems.

5. Processed ASCII files description

The 1-min daily ASCII files saved under D *way_proc_data_flux* have the following format:

The columns for files *Xcruise_yyyy_proc_met_1min_yyyy_mm_dd_doy.txt* are as follow:

```
doy=x(:,1);            % day of year
pir=x(:,2);            % averaged downward IR flux between Eppley unit and K&Z
unit (W/m^2)
psp=x(:,3);            % averaged downward solar flux between Eppley unit and
K&Z unit (W/m^2)
Tc1=x(:,4);            % case temperature of PIR Eppley unit (C)
Td1=x(:,5); );         % dome temperature of PIR Eppley unit 1 (C)
Tsea=x(:,6             % sea snake temperature, ~0.10 m depth (C)
Tvais=x(:,7);          % air temperature(C)
Rhvais=x(:,8);         % Relative Humidity (%
org=x(:,9);            % rainrate, STI optical rain gauge, uncorrected (mm/hr)
org_carrier =x(:,10);  % rain gauge function (V)
aspir_on=x(:,11);      % backflow indicator for RH/T sensor.
press=x(:,12);         % atmospheric pressure (mb)
```

The columns for files *Xcruise_yyyy_proc_rad_1min_yyyy_mm_dd_doy.txt* are as follow:

```
doy =x(:,1);      % day of year
pir1=x(:,2);      % downward IR flux from Eppley unit (W/m^2)
pir2=x(:,3);      % downward IR flux from K&Z unit (W/m^2)
psp1=x(:,4);      % downward solar flux flux from Eppley unit (W/m^2)
psp2=x(:,5);      % downward solar flux flux from K&Z unit (W/m^2)
Tc1=x(:,6);       % case temperature of PIR Eppley unit (C)
Td1=x(:,7);       % dome temperature of PIR Eppley unit (C)
```

```
Tkz=x(:,8);       % temperature of PIR K&Z unit (C)
Tvais=x(:,9);     % air temperature(C)
Rhvais=x(:,10);   % Relative Humidity (%)
Tsea=x(:,11);     % sea snake temperature, ~0.10 m depth (C)
```

The columns for files *Xcruise_yyyy_proc_gpsnav_1min_yyyy_mm_dd_doy.txt* are as follow:

```
doy=x(:,1);         % day of year
gpslat1i=x(:,2);      %decimal latitude (deg)
gpslon1i=x(:,3);      %decimal longitude (deg)
gpsspeedi=x(:,4);     %GPS Speed-Over-Ground (m/s)
gpsheadi=x(:,5);      %GPS Course-Over-Ground (deg)
headxi_pitch=x(:,6);  %Crescent GPS heading (deg)
pitchxi_pitch=x(:,7); %Crescent GPS angle (pitch) (deg)
pitchxi_roll=x(:,8);  %Crescent GPS angle (roll) (deg)
```

The columns for files *Xcruise_yyyy_proc_son_1min_yyyy_mm_dd_doy.txt* are as follow:

```
doy =x(:,1);    %day of year
U=x(:,2);       %u wind component (+boward) , m/s
V=x(:,3);       %v wind component (+portward) , m/s
W=x(:,4);       %w wind component (+up) , m/s
Tsonic=x(:,5);  %sonic temperature, C
dir =x(:,6)     %relative wind direction (from),clockwise rel ship's bow, deg
```

The columns for files *Xcruise_yyyy_proc_licor_1min_yyyy_mm_dd_doy.txt* are as follow:

```
jd_pc=x(:,1);             %day of year
Licor_H2O_mi=x(:,2);      %Licor H2O (g/kg)
Licor_CO2_ai=x(:,3);      %Licor CO2 (umol/mol)
Licor_Tempi=x(:,4);       %Licor box temperature (degC)
Licor_Pressi=x(:,5);      %Licor box pressure (hPa)
Licor_agci=x(:,6);        %Licor AGC value (%)
```

The columns for files *Xcruise_yyyy_PSD_flux_5min_yyyy_mm_dd_doy.txt* (5-min average) and *Xcruise_yyyy_PSD_flux_30min_yyyy_mm_dd_doy.txt* (30-min average) are as follow:

```
jad5=x(:,1);     %day of year
s1=x(:,2);       %psd true wind speed (m/s)
dir1=x(:,3);     %psd true wind direction (deg)
ts=x(:,4);       %psd seasnake temperature (degC)
ts_tsg=x(:,5);   %If available, ship thermosalinograph temperature (degC)
sal_tsg=x(:,6);  %If available, ship theromsalinograph salinity (psu)
ta=x(:,7);       %psd air temperature (degC)
qs=x(:,8);       %psd air specific humidity at sea surface (g/kg)
qa=x(:,9);       %psd air specific humidity (g/kg)
psp=x(:,10);     %psd solar flux (w/m^2)
pir=x(:,11);     %psd IR flux (w/m^2)
org=x(:,12);     %psd optical raingage precipitation rate (mm/hr)
shp_spd=x(:,13); %ship sog from psd gps (m/s)
shp_hed=x(:,14); %ship heading from psd differential gps (deg)
relsp=x(:,15);   %psd relative wind speed (m/s)
reldir=x(:,16);  %psd relative wind direction (deg)
lat=x(:,17);     %psd decimal latitude (deg)
lon=x(:,18);     %psd decimal longitude (deg)
zt=x(:,19);      %Depth of SST sensor used in heat flux calc (m)
```

```
sig_sp=x(:,20);  %standard deviation of ship speed (m/s)
taub=x(:,21);    %wind stress, coare 3.0 (N/m^2)
hsb=x(:,22);     %sensible heat flux, coare 3.0 (w/m^2)
hlb=x(:,23);     %latent heat flux, coare 3.0 (w/m^2)
rf=x(:,24);      %rain heat flux (w/m^2)
ta_im=x(:,25);   %If available, IMET air temp (degC)
qa_im=x(:,26);   %If available, IMET air specific humidity (g/kg)
s_shp=x(:,27);   %If available, IMET true wind speed (m/s)
dir_shp=x(:,28); %If available, IMET true wind direction (deg)
psp_im=x(:,29);  %If available, IMET solar flux (w/m^2)
pir_im=x(:,30);  %If available, IMET IR flux (w/m^2)
pressm=x(:,31);  %psd atmospheric pressure (mb)
rh_psd=x(:,32);  %psd relative Humidity (%)
```

IV. Contact

Send an email to ludovic.bariteau@noaa.gov if any issues or concerns with the MATLAB programs