KARL-FRANZENS-UNIVERSITÄT GRAZ
UNIVERSITY OF GRAZ
Institut für Physik | Geophysik, Astrophysik und Meteorologie

# Data Analysis in Astrophysics

WS 2025/2026

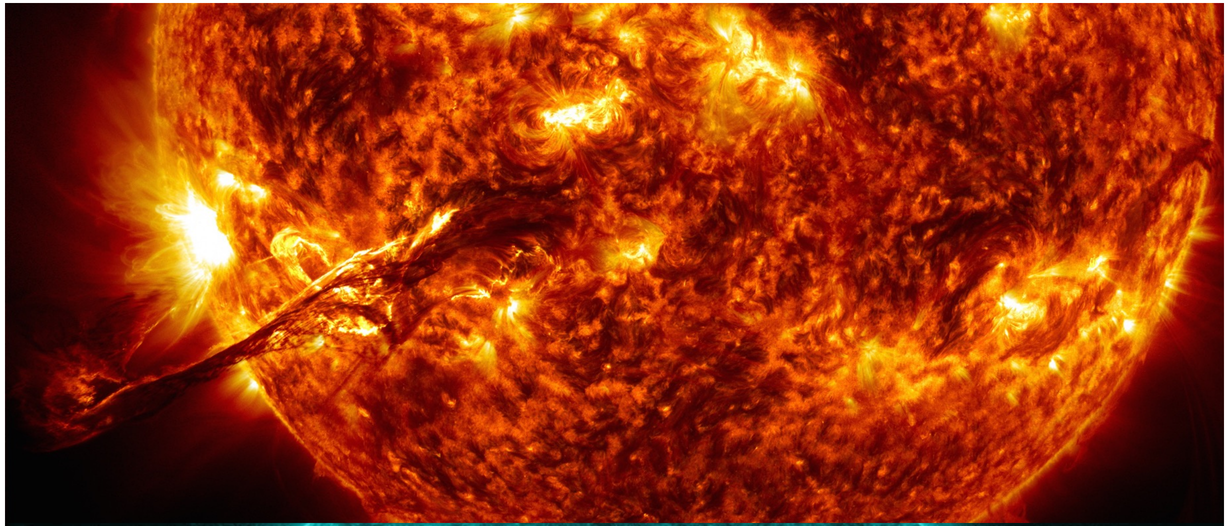
VO: PHM.013UB; Tue, 15:45-18:15
UE: PHM.014UB; Mon, 09:00-10:45

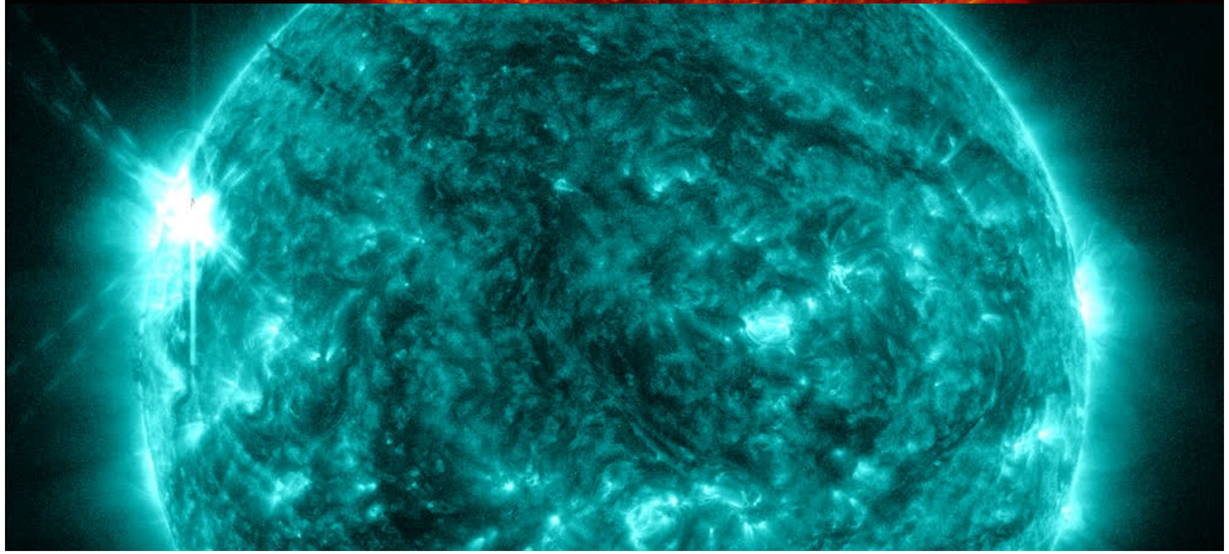
**Dr. Karin Dissauer & Dr. Veronika Witzke**

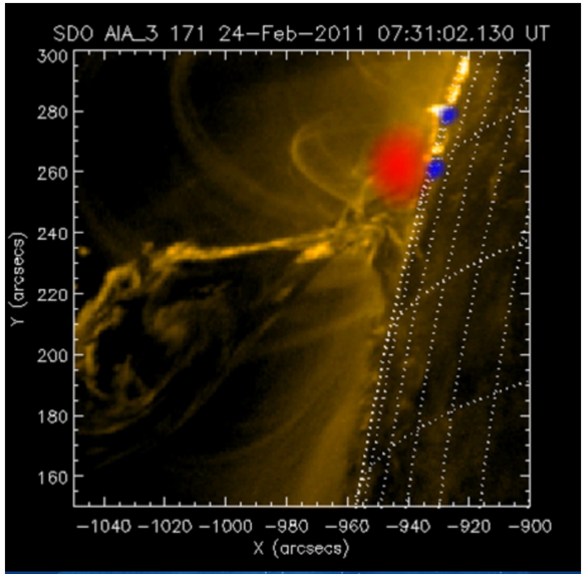
**Technical Support: Ing. Roland Maderbacher**

# Lecture 4: Solar Soft (SSW) IDL

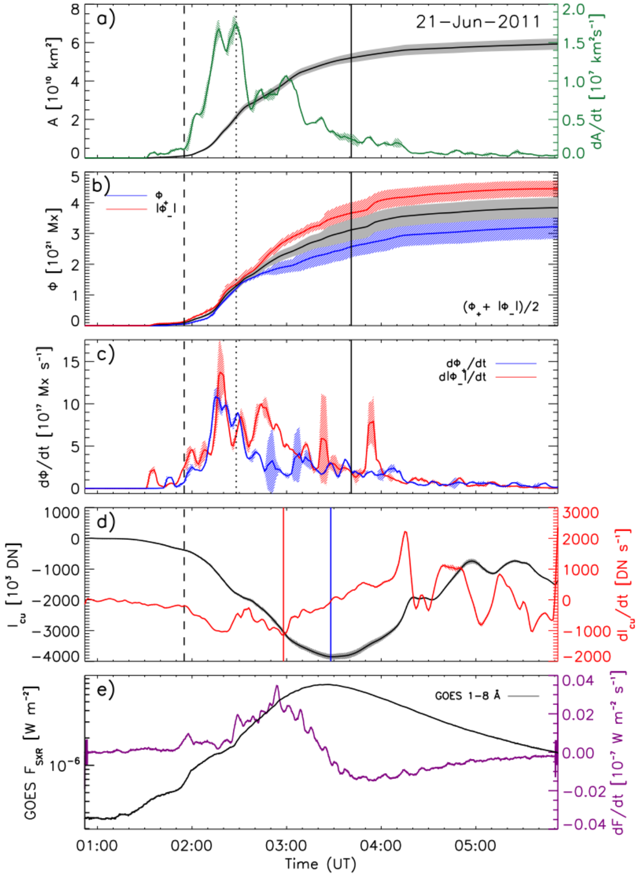
# Solar SoftWare (SSW) IDL











# Solar SoftWare (SSW) IDL

- *"SolarSoft is a set of integrated software libraries, data bases, and system utilities which provide a common programming and data analysis environment for Solar Physics. The SolarSoftWare (SSW) system is built from Yohkoh, SOHO, SDAC and Astronomy libraries and draws upon contributions from many members of those projects. It is primarily an IDL based system, although some instrument teams integrate executables written in other languages.*
- *The SSW environment provides a consistent look and feel at widely distributed co-investigator institutions to facilitate data exchange and to stimulate coordinated analysis. Commonalities and overlap in solar data and analysis goals are exploited to permit application of fundamental utilities to the data from many different solar instruments. The use of common libraries, utilities, techniques and interfaces minimizes the learning curve for investigators who are analyzing new solar data sets, correlating results from multiple experiments or performing research away from their home institution."*
  http://www.lmsal.com/solarsoft/

# Solar SoftWare (SSW) IDL

- SSW contains relevant routines (and partially also data necessayrs for data reduction) for many solar physics missions and satellites, like Yohkoh, SOHO – Solar and Heliospheric Observatory, TRACE – Transition Region and Coronal Explorer, RHESSI – Reuven Ramaty High Energy Solar Spectroscopic Imager, GOES – Geostationary Operational Enviromental Satellites, Hinode, STEREO – Solar TErrestrial RElations Observatory, SDO – Solar Dynamics Observatory, IRIS – Interface Region Imaging Spectrograph, PSP – Parker Solar Probe, etc. as well as for several ground-based observatories.
- It facilitates the coordinated data analysis in solar physics, the exchange of data, and pushes the compliance of common standards.
- Note: In almost all solar phyiscs missions and observatories, an „open data policy“ prevails, i.e. the data are accessible to everybody of the scientific community (not only the instrument teams) as well as to the public.

# Solar SoftWare (SSW) IDL

- SSW can be run under Unix, Linux, Windows, Mac. It is a dynamic system, i.e. continuously new programs are added, and existing programs are updated.
- To call IDL in the SSW environment, type:

```
sswidl [idl_startupfile]
```

- On the Linux machines in the computer room, first open a shell: `tcsh -l` .
- SSW includes programs for:
  - File input/output, to work with FITS-files
  - IDL data manipulation (structures, strings, arrays, mathematics, statistics, ...)
  - Data reduction in solar physics
  - Time series analysis, time format manipulations, `utplot`
  - Visualisation and processing of images and image series
  - Geometry (coordinate transformations, ...)
  - WWW presentations (HTML conversions, file conversions, movies, ...), etc.

# Solar SoftWare (SSW) IDL

- An index and search for SSW routines by cateories and names is available at:
  http://www.sipwork.org/zarro/xdoc/
- To obtain help to programs in SSW IDL, one can use:
  `xdoc [, ’programmname’]`
- SSW web pages (in particular relavant, when you want to instal it on your own computer):
  - SSW overview: http://www.lmsal.com/solarsoft/
  - Installation: http://www.lmsal.com/solarsoft/ssw_install.html
  - SSW setup: http://www.lmsal.com/solarsoft/ssw_setup.html
  - Automatic upgrades: http://www.lmsal.com/solarsoft/ssw_upgrades.html
  - SSW standards: http://www.lmsal.com/solarsoft/ssw_standards.html
  - SSW under Windows: http://www.lmsal.com/solarsoft/ssw_windows.html

# Solar SoftWare (SSW) IDL

- **FITS Files**

  - FITS (Flexible Image Transport System) was introduced as a standardized data format for the storage, processing and transfer of 2D astronomical data.
  - FITS is the most commonly used digital format in astronomy.
  - The specific advantages of fits files is that:
    - 1) The data are saved in binary format: a) independent of the internal coding of the machine on which they were processed, and b) efficient in terms of memory size.
    - 2) In addition to the data (in binary format), metadata information to the data is saved in human-readable ASCII format, which can be accessed independently of the data themselves.

# Solar SoftWare (SSW) IDL

- **FITS Files**

  - A FITS file consists of a header and a data block.
  - The data block contains the data themselves (e.g., an image or a spectrum), the header contains relevant information on the data.
  - The header can only contain human-readable ASCII characters, while the data block is a binary array.
  - Both header and data block are built from records, which are elementary units in FITS and contain exactly 2880 Bytes (= 36 x 80 characters).
  - The header consists of a list of keywords, which provide information on the organisation and format of the data in the data block as well as necessary meta-information on the observations.
  - Each keyword has a length of 80 Bytes and consists of three fields: the name of the keyword (maximum 8 string characters), the value of the keyword (starts with the 10th character) and an optional comment field (after a "/"), which can extend up to the 80th character.

# Solar SoftWare (SSW) IDL

- **FITS Files**

- The number of keywords is not limited, and the names can be mostly freely chosen (but typically they follow some standards).

- The beginning of the header needs to include the following obligatory keywords:

  - SIMPLE – logical (T[rue]/F[alse]), indicates that it is a standard FITS file
  - BITPIX – Integer, represents the data format („bits per pixel“: number of bits that describe a data entry);
  - NAXIS – Integer, number of dimensions (e.g., 2 for an image);
  - NAXIS1 – Integer, gives the number of pixels of the first dimension (columns);
  - NAXIS2 – Integer, gives the number of pixels of the second dimension (rows).

- Example for the beginning of a FITS header:
  - `SIMPLE = T / Standard FITS format`
  - `BITPIX = 12 / Number of bits per data pixel`
  - `NAXIS = 2 / Dimensions`
  - `NAXIS1 = 1008 / Number of columns`
  - `NAXIS2 = 1016 / Number of rows`

# Solar SoftWare (SSW) IDL

- **FITS Files**

  - After this beginning of a FITS file, typically a number of optional keywords are following, which contain information on the observations and data (which instrument, wavelength, date/time of observation, exposure time, pixel scale, position, physical units, etc.)
  - The FITS header is closed by the necessary keyword `END` (has no value).
  - In addition, there exist two specific keywords, `HISTORY` und `COMMENT`, which are used to give additional information on the data.

# Solar SoftWare (SSW) IDL

- **FITS Files**

  - Example of a FITS header of an Halpha full-disk image recorded at Kanzelhöhe Observatory (the keywords used follow the SOHO standard):

```
SIMPLE = T / file does conform to FITS standard
BITPIX = 16 / number of bits per data pixel
NAXIS = 2 / number of data axes
NAXIS1 = 2048 / length of data axis 1
NAXIS2 = 2048 / length of data axis 2
BUNIT = 'CCD COUNTS'
BSCALE = 1 / default scaling factor
BZERO = 32768 / offset data range to that of unsigned short
DATAMIN = 37
DATAMAX = 945
CTYPE1 = 'SOLAR_X'
CTYPE2 = 'SOLAR_Y'
CUNIT1 = 'arcsec '
CUNIT2 = 'arcsec '
```

# Solar SoftWare (SSW) IDL

- **FITS Files**

○ Example of a FITS header of an Halpha full-disk image recorded at Kanzelhöhe Observatory (the keywords used follow the SOHO standard):

```
CRPIX1 = 1024.50 / [pix]
CRPIX2 = 1024.50 / [pix]
CRVAL1 = -5.2528
CRVAL2 = -24.8314
CDELT1 = 1.02369 / [arcsec/pix]
CDELT2 = 1.02369 / [arcsec/pix]
DATE_OBS= '2015-02-02T07:54:19Z`
DATE-OBS= '2015-02-02T07:54:19`
CENTER_X= 1029.63 / [pix]
CENTER_Y= 1048.76 / [pix]
SOLAR_R = 955.886 / [pix]
ANGLE  = -12.9652 / [deg]
CROTA1 = 12.9652 / [deg]
```

# Solar SoftWare (SSW) IDL

- **FITS Files**

- Example of a FITS header of an Halpha full-disk image recorded at Kanzelhöhe Observatory (the keywords used follow the SOHO standard):

```
SOLAR_P0= -12.5680 / [deg]
SOLAR_B0= -6.08757 / [deg]
OBJECT = 'SUN ' / NOAA # or SUN for full disk
OBS_TYPE= 'HALPH`
OBS_MODE= 'FULL DISK`
OBS_PROG= 'HALPHA PATROL`
TYPE-DP = 'ARCHIVE ' / Data processing Type
TELESCOP= 'KHPI`
INSTRUME= 'HA2`
DETECTOR= 'TM4200`
WAVELNTH= 6563 / [ANG], FWHM=0.7 [ANG]
EXP_TIME= 2.716 / Exposure Time [ms]
EXP_MODE= 0 / Exp. Mode (0=auto,1=dbl,2=fix,3=both)
```

# Solar SoftWare (SSW) IDL

- **FITS Files**

  - Example of a FITS header of an Halpha full-disk image recorded at Kanzelhöhe Observatory (the keywords used follow the SOHO standard):

```
PRE_INT = 700 / Preselected PixInt in AOI
A_O_INT = '695,674,1057,819' / Rect. for PixInt [X0,Y0,X1,Y1]
QUALITY = 1 / image quality [1-3]
ORIGIN = 'KANZELHOEHE OBSERVATORY, A-9521 TREFFEN, AUSTRIA`
DATE = '2015-02-02' / Date of FITS Creation
COMMENT Orientation: N up, W right, first pix is left bottom
HISTORY No intesity processing applied
END
```

# Solar SoftWare (SSW) IDL

- **Data I/O: Fits Files**

- In SSWIDL, there exist different routines to read FITS files, e.g.

```
data = readfits('filename.fts' [, header] [, keywords=keywords])
```

- Reads the data into the variable `data`; the keyword `header` can be used to get the FITS header (as String array)

```
mreadfits, 'filename.fts', index [, data] [, keywords=keywords]
```

- Reads the FITS header into the variable `index` and the data into the variable `data`. In contrast to readfits, the FITS header is read into an easily accessible structure. mreadfits can also handle 3D FITS files (image series).

- For writing FITS files, there are e.g. the routines

```
writefits, filename, data [, header] [, keywords=keywords]
mwritefits, index, data, [outfile=outfile,] [outdir=outdir,] [,
keywords=keywords]
```

# Solar SoftWare (SSW) IDL

- **Data I/O: Fits Files**

- Note: Sometimes there exist also specific routines for an instrument to read FITS files. Often, these are wrappers to mreadfits or readfits, but also may consider some instrument-specific properties, e.g.

```
rd_mdi, mdifitsfiles, mdiindex [, mdidata] [, keywords=keywords]
```

- For most instruments on satellite mission, there are routines provided for the standard data reduction (dark current subtraction, flatfield, etc.), e.g.

```
eit_prep, eitindex, data=eitdata, eitindex_out, eitdata_out
```

- Note: EIT/SOHO: Extreme Ultraviolet Imaging Telescope
  MDI/SOHO: Michelson Doppler Imager

# Solar SoftWare (SSW) IDL

- **Data I/O: Fits Files**

- Often used technique is to first read only the FITS header of an images series, check for specific properties that are of interest (e.g. certain wavelengths, time range, exposure time, etc) and only thereafter to read the data - only of the subset of interest.
- This is in particular an efficient approach in case of big data arrays, which may easily exceed the resources/memory available. The separation between meta-data (header) and data offered by the FITS format makes this approach very easy.

Expl.:

```
path = /sswidl/scratch/kad/eit_14-Sep-99/‘
files = file_search( path + 'efz*.*' ) ;Search all data files in
the directory path, which start with ‘efz’
help, files ; FILES STRING = Array[118]
mreadfits, files, index ;Read all FITS headers into index
help, index, /str ;Information on the structure variable index
wl = where( index.wavelnth eq '304' ) ;Check for certain
attributes (obs. wavelengths)
```

# Solar SoftWare (SSW) IDL

- **Data I/O: Fits Files**

Expl. cont.:

```
help, wl ;c WL LONG = Array[4]
mreadfits, files[ wl ], index, data ;Read the data that
                                     fulfill the attribute
eit_prep, index, data=data, index2, data2 ;Apply the instrument's data
                                           reduction
eit_colors, 304 ;Load EIT 304 Å color table
tvscl, congrid( data2[ *, *, 0 ], 512, 512 ) ;Rescale and display
                                              first image of series
a = where( data2 le 0 )
data2[ a ] = 0.1 ;Set negative intensities to value 0.1
tvscl, congrid( alog10( data2[ *, *, 0 ]), 512, 512 ) ;logarithmic
                                                       scaling for display
xstepper, alog10( data2 ), xsize=512, ysize=512 ;interactive tool for
                                                 image cubes
xmovie, congrid( alog10( data2 ), 512, 512, 4 ) ;Animates the 4 images
```

# Solar SoftWare (SSW) IDL

- **Data I/O: Fits Files**

Alternatively to `file_search` one could also select files interactively by using `dialog_pickfile`:

```
files = dialog_pickfile( path=path, /multi )
```

- **Data download for various solar missions & Kanzelhöhe Observatory**

  - Virtual Solar Observatory (VSO):
    - via Webform: [https://sdac.virtualsolar.org/cgi/search](https://sdac.virtualsolar.org/cgi/search) (Instrument/Source/Provider → Instrument / Source (not provider dependent) → Generate VSO search form)
    - via sswidl: first the routine `vso_search` is used to search for data in the specified time range and with the given attributes (instrument, filter, cadenc etc.)

      In the next step, we can download the data via the routine `vso_get`.

# Solar SoftWare (SSW) IDL

- **Data download for various solar missions & Kanzelhöhe Observatory**

- Virtual Solar Observatory (VSO):
  - via sswidl

  Some examples:

```
sel = vso_search('2019/10/22 14:00:00', '2019/10/22 14:01:00',
instr='aia')
status = vso_get( sel, out_dir ='/home/kad/aia_data/22-10-2019/')
```

  The variable `status` contains info about the downloaded data, see e.g.

```
help, status, /str
```

  When searching for data, we can also specify further parameters/attributes, e.g.

```
s304 = vso_search('2014/10/22 14:00:00', '2014/10/22 15:00:00',
instr='aia', wave = '304', sample = 600)
```

- Data download from Kanzelhöhe Observatory: https://cesar.kso.ac.at and https://kanzelhohe.uni-graz.at

# Solar SoftWare (SSW) IDL

- **Mapping Software (“maps”)**

- An IDL “map” in SSW ist basically a structure variable, which contains 2D image data plus related properties, like information on the observing instrument, exposure time, pixel scale, image center, image orientation, etc.
- Using such maps, an image can be manipulated in an easy manner, independent of its origin, and can be easily combined with other images.
- Typical manipulations are: image rotation to Solar North up, selection of an image subfield, plotting of a heliographic grid, overplotting of several images (e.g. magnetic contours on an continuum image), compensation of differential solar rotation, co-alignment of images from different instruments/satellites, etc.

# Solar SoftWare (SSW) IDL

- **Mapping Software (“maps”)**

  - Creation and structure of a map:

```
image = dist( 500, 300 ) & map = make_map( image )& help, map, /structure
```

```
c ** Structure <138eee8>, 13 tags, length=600104, data length=600097, refs=1:
DATA FLOAT Array[500, 300] ; data
XC FLOAT 0.00000 ; Deviation of x-coordinate of solar center from image center
YC FLOAT 0.00000 ; -„- y-coordinate -„-
DX FLOAT 1.00000 ; Pixel scale in x-direction [arcsec/pixel]
DY FLOAT 1.00000 ; Pixel scale in y-direction [arcsec/pixel]
TIME STRING '29-Nov-2015 08:52:00.739' ; Observing time
ID STRING '' ; Observing instrument
ROLL_ANGLE FLOAT 0.00000 ; Rotation of image with respect to Solar North (°)
ROLL_CENTER FLOAT Array[2] ; Center of rotation
DUR FLOAT 0.00000 ; Exposure time
XUNITS STRING 'arcsecs' ; Units in x-direction
YUNITS STRING 'arcsecs' ; Units of y-direction
SOHO BYTE 0 ; SOHO Instrument yes/no
```

# Solar SoftWare (SSW) IDL

- **Mapping Software (“maps”)**

  - Normally, when creating a map, also all relevant information should be included, e.g.:

```
map2 = make_map (image, xc = 249, yc = 149, dx = 2.3, dy = 2.3, $
time = '22-Oct-15 07:13:20.2', duration = 0.12, id='KSO H-alpha', $
roll_center = [249,149], roll_angle = -21.55)
```

  - Creation of a map from a FITS file:

If our fits file complies with the standards, the map will contain all the correct and necessary info needed.

Vers. 1: **`mreadfits, filename, index, data`** `; Read the` *`data`* `(from data block) and image properties` *`index`* `(from Fits header)`

```
;Alternatively: use specific FITS read routine of instruments
;also: one should apply the data reduction
```

```
index2map, index, data, map3 ; Convert data to map structure, using the
;information in the fits header
```

# Solar SoftWare (SSW) IDL

- **Mapping Software (“maps”)**

Vers. 2: **`fits2map, filename, map3b`** ; Read data and image properties + convert to map

Expl.: `rd_mdi, ’/home/kad/Teaching/DataAnalysis/data/fd_Ic.fits’, index, data`

```
index2map, index, data, mdi_map
```

Expl.: `files = file_search( path + 'efz*.*' )`

```
files = files[ 0:4 ]
```

`mreadfits, files, index, data` ; Read fits file

`eit_prep, index, data=data, index2, data2` ; Apply instrument-specific data reduction

`index2map, index2, data2, map` ; Convert the *reduced* EIT data to map

# Solar SoftWare (SSW) IDL

- **Mapping Software (“maps”)**

- **Display of maps:**
- Maps can be easily displayed via the procedure `plot_map`, which also offers a variety of keywords

Expl.: `plot_map, map3 [, keywords=keywords]`

`plot_map, map3, dmax=2000, dmin=0` ;maximum intensity value displayed is 2000, minimum instensity value displayed is 0

`plot_map, map3, dmax=2000, dmin=0, grid = 15` ;plots also a heligraphic grid in 15°

`plot_map, map3, xrange = [-300, 400], yrange = [-600, -100]` ;specifies the x- and y-ranges displayed (in arcseconds)

`plot_map, map2, /over` ;plots map2 as contour over previously plotted map3
;corrects automatically for field displayed, different orientation,
;different pixel scale of the 2 images combined, etc.

# Solar SoftWare (SSW) IDL

- **Mapping Software ("maps")**

  - For further keywords, use: `xdoc, 'plot_map_index'`
  - Note, multiple maps can be configured in one window using `!p.multi`, i.e. the maps are handled like a plot command rather than a `tvscl` command.
  - Rotation of a map:

```
rmap = rot_map(map, roll_angle = 0)  ;Rotates the image to 0° = North up)
rmap2 = rot_map(map, roll_angle = 60)  ;Rotates the image by  60° (clockwise)
rmap3 = rot_map( map, angle = -30 )  ;Rotates the image by 30°(anti-clockwise)
plot_map, rmap2, grid = 15           ;Draws heliographic grid (in 15° intervals)
```

# Solar SoftWare (SSW) IDL

- **Mapping Software ("maps")**

  - Further ways of manipulating maps (correction of differential rotation, selection of subregions, overlays, creation of difference maps, contour representations, movies, etc.) are described on Dominic Zarro's homepage: http://hesperia.gsfc.nasa.gov/rhessidatacenter/complementary_data/maps/

  - Some relevant routines to work with maps:

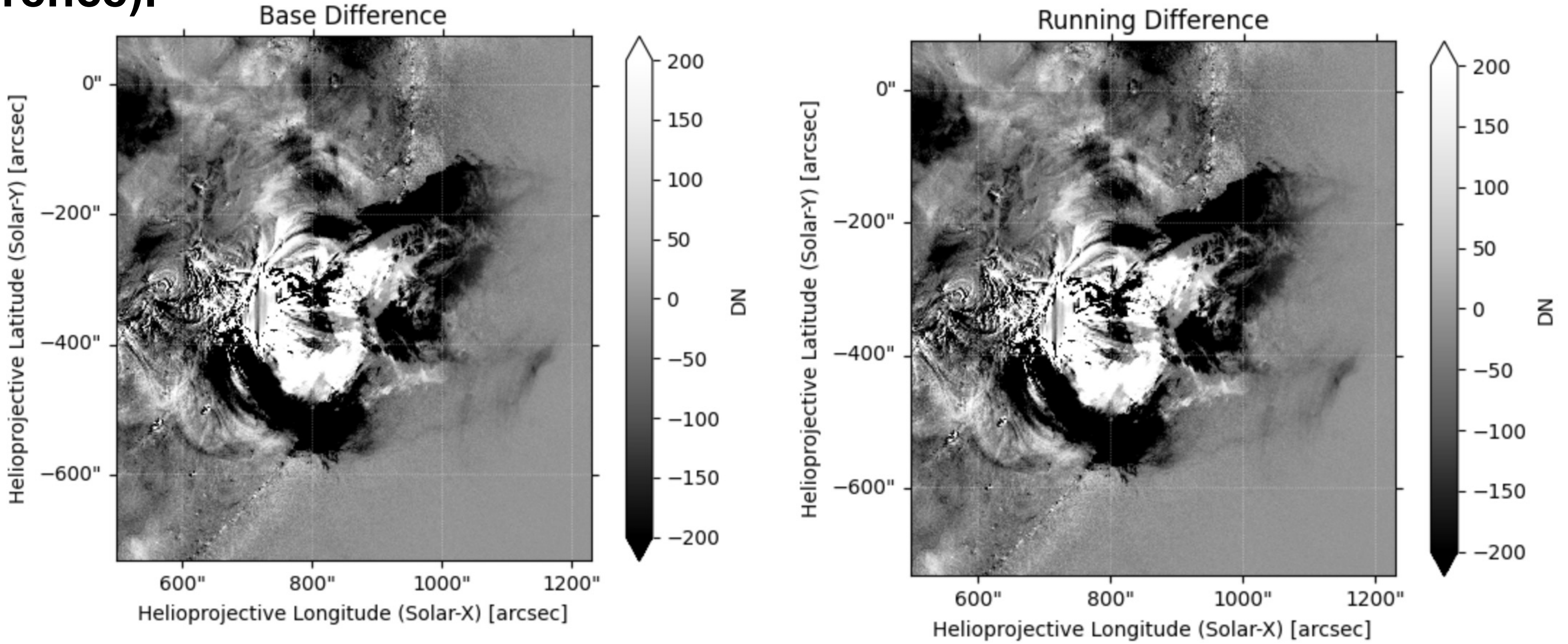
| | |
|---|---|
| `plot_map` | plot an image map |
| `rot_map` | rotate an image within a map structure |
| `drot_map` | apply solar differential rotation to an image within a map structure |
| `diff_map` | build the difference or ratio (check keywords) between two maps |
| `sub_map` | extract a new map covering only a subregion |
| `movie_map` | play a movie of an array of maps |
| `align_map` | align an input map to a reference map |
| `coreg_map` | co-register input map to a reference map (i.e. same image center, pixel size etc.) |

# Solar SoftWare (SSW) IDL

- **Mapping Software (“maps”)**

  - Base-difference/ratio maps & Running-difference/ratio maps

  - When analyzing solar imaging data it is often useful to look at the **difference from one time step to the next (running difference)** or the **difference from the start of the sequence (base difference).**



# Solar SoftWare (SSW) IDL

- **Mapping Software (“maps”)**
  - Base-difference/ratio maps & Running-difference/ratio maps

Syntax:

```
diff=diff_map(map1,map2) ;map1-map2
ratio=diff_map(map1,map2, /ratio) ;map1/map2
```

Running difference/ratio: map2 is previous image

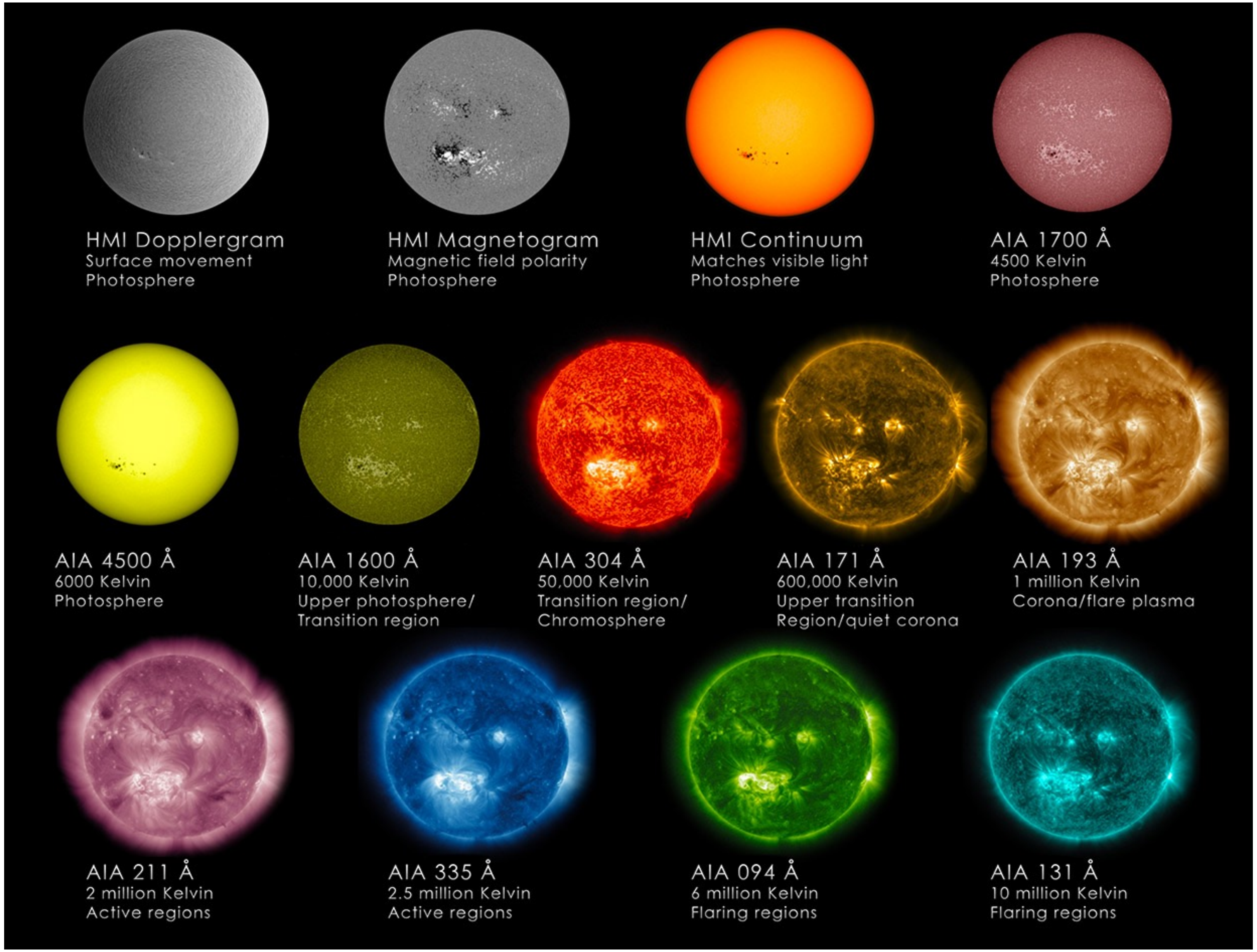
Base difference/ratio: map2 is “base” map, i.e. map at a specific reference time

# Solar SoftWare (SSW) IDL

- **Solar Dynamics Observatory (SDO)**

  - The Solar Dynamics Observatory (SDO) is a NASA mission that was launched in 2010.
  - It has three instruments onboard to observe the Sun:
    - **the Atmospheric Imaging Assembly (AIA)** , which observes the full solar disk in 10 different filters in the UV and EUV wavelength range;
    - **the Helioseismic and Magnetic Imager (HMI)** , which obtains magnetic field images of the full Sun: line-of-sight (LOS) magnetic field maps and 3D vector fields, LOS dopplergrams as well as continuum images; and
    - **the Extreme Ultraviolet Variability Experiment (EVE)** , which measures the spatially integrated solar EUV irradiance
  - The SDO imaging is obtained with high spatial and temporal resolution (e.g. AIA: 12 s cadence in the EUV filters and 24 s cadence in the UV filters, 1.5 arcsec resolution; CCD: 4096 × 4096 pixel, 32-bit, 0.6 arcsec per pixel). Consequently, the SDO data rate is enormous, about 1.5 TeraByte per day

# Solar SoftWare (SSW) IDL




Credit: NASA

# Solar SoftWare (SSW) IDL

- **Solar Dynamics Observatory (SDO)**

o **Reading SDO data in IDL:**

  - AIA as well as HMI Level-1 can both be read in IDL with `read_sdo`.
  - In the following, we show a simple example of selection and reading of AIA images that were obtained with its EUV 94 Å filter:

```
f = file_search('/sswidl/scratch/kad/aia_11-Apr-13/*.fits')
read_sdo, f, index
a = where(index.wavelnth eq 94)
f94 = f[a]
read_sdo, f94, index94, data94
index2map, index94, data94, map94
help, map94[0]
aia_lct, wavelnth = 94, /load
movie_map, map94, /log, xr=[-500,100], yr=[0,500]
```

# Solar SoftWare (SSW) IDL

- **Solar Dynamics Observatory (SDO)**

  - **Reading SDO data in IDL:**
    - The uncompressed HMI and AIA data have typically a size of 64 MB.
    - This means, if we work with image sequences, we need to be careful not to read in too many images at once.
    - Otherwise, the internal memory of our machines won't be able to cope with it.
    - Also note that it is possible to download SDO data in compressed format (RICE compression).
    - In this format, an AIA image needs only about 12 MB of memory, an HMI image about 15 MB. These data are then decompressed when they are read in IDL, and saved to a directory /tmp.
    - To delete these decompressed data again, one should set the keyword `uncomp_delete` when reading the data:

```
read_sdo, f94, index94, data94, /uncomp_delete,
[keywords=keywords]
```

# Solar SoftWare (SSW) IDL

- **Solar Dynamics Observatory (SDO)**

o **Reading SDO data in IDL:**

  - Sometimes we do not need the full resolution in the analysis. This may be the case, e.g., when analysing large-scale objects like coronal holes, or when doing a first check on the data.
  - When downscaling the images, we have less demands on the memory needed for the analysis.
  - To this aim the keyword `outsize` can be used, which allows us to downscale the dimension of the images (in integers of the full resolution of 4096 × 4096 pixels).
  - Check the different keywords with `xdoc, ’read_sdo’`

# Solar SoftWare (SSW) IDL

- **Solar Dynamics Observatory (SDO)**

  - **Reading SDO data in IDL:**
    - In the SDO Level-1 data, already several important data reductions have been applied, like dark current, flat-field, bad pixel correction, de-spiking.
    - This means the Level-1 data are calibrated, and suitable for scientific analysis.
    - However, if we want to make use of the full capacity of the simultaneneous multi wavelengths imagery of the different AIA filters and also combining them with HMI data, one should use Level-1.5 data.
    - The additional calibration that is applied from Level-1 to Level-1.5 is that the corrections for the slightly different angular orientations of the four AIA telescopes, the slightly different pointings and different image scale due to the different focal length in the different wavelengths are applied (i.e. the corrections regard rotation, translation and spatial scaling).
    - The AIA data provided in data archives are typically Level-1.
    - If we want to work with Level-1.5 data instead, then we need to use the procedure `aia_prep` instead of `read_sdo` to read the data

# Solar SoftWare (SSW) IDL

- **Solar Dynamics Observatory (SDO)**

o **Reading SDO data in IDL:**

  - Example:

```
aia_prep, f94, -1, index94, data94
```

  - The second parameter (-1) indicates that all data shall be read. If we want to read, e.g., only each fifth image from the file list `f94`, then we would instead specifiy the corresponding indices:

```
n = n_elements(f94)
ii = indgen(n/5)*5
aia_prep, f94, ii, index94, data94
```

# Solar SoftWare (SSW) IDL

- **Solar Dynamics Observatory (SDO)**

  - **Reading SDO data in IDL:**

    - For a detailled description of AIA Level-1 data see http://jsoc.stanford.edu/new/AIA/AIA_lev1.html
    - A description of the AIA instruments and the different AIA data products is given in *Lemen et al., 2002, Solar Phys. 275, 17-40*.
    - A description of the HMI instrument and data products is given in *Scherrer et al., 2002, Solar Phys. 275, 207-227*, and *Schou et al. 2002, Solar Phys. 275, 229-259*.

# Exercises

**Exercise 16:** Read the fits header of all EIT images in the directory /hosts/pauli/home/DataAnalysis/data/eit_14-Sep-99/. To this aim, make use of the routines `file_search` and `mreadfits`.

a) Check the first header. Determine from the fits headers, at which wavelengths the images were recorded.

b) Read the data of the 3 first images that were recorded in the 195 Å filter. Apply the standard EIT data reduction. First, for testing, apply it only to the first 3 images.

c) Display the first image of the sequence. Compare how the image looks when applyig a linear and when applying a logarithmic intensity scaling (take care for negative values). Why do the images look so different with different scalings?

d) Play the image sequence as movie (in logarithmic scaling), using `xmovie`. Reduce the size of the images for the movie play (see the example above).

e) Create and plot the histogram of the first image. Choose a proper binsize

f) Compare the HISTORY keyword of the original FITS headers with that of the corrected data.

# Solar SoftWare (SSW) IDL

- **Solar Dynamics Observatory (SDO)**

  - **Download of SDO data:**

    - There exist different ways to search and download SDO data. Some of them are listed below. Note, the recommended way is version 1 (JSOC).
    - 1) JSOC (Joint Science Operations Center) Lookdata: http://jsoc.stanford.edu/ajax/lookdata.html
    - The JSOC Lookdata tool ist an Interface for the selection of SDO data. It may appear somewhat complicated at the beginning, but when used to it, it provides a very compact way of data search and download. One first selects the data with the attributes wanted, and sends the requests.
    - After the data are then prepared, one obtains an email with a link to the server where the data can be downloaded (the best may be to choose the download option ftptar in the data selection). An example for the syntax in the JSOC search mask:

```
aia.lev1_euv_12s[2013-04-11T06:45:00Z/150m@2m][94,171,193]
```

# Solar SoftWare (SSW) IDL

- **Solar Dynamics Observatory (SDO)**

o **Download of SDO data:**

  - An example for the syntax in the JSOC search mask:

```
aia.lev1_euv_12s[2013-04-11T06:45:00Z/150m@2m][94,171,193]
```

  - With this command, we select AIA images of three EUV filters (94, 171 and 193 Å) over a time range of 150min, starting with the indicated time, and with a cadence of 2 min. This means that not each available image in the selected time range and filter will be provided (which would be 12 s) but only one each 120 seconds, i.e. each 10th image.

  - Take care of the exact syntax, time format, etc. In case, we want to get each image of the series in the 193 Å filter, the corresponding input command would be:

```
aia.lev1_euv_12s[2013-04-11T06:45:00Z/150m][193]
```

# Solar SoftWare (SSW) IDL

- **Solar Dynamics Observatory (SDO)**

## Download of SDO data:

- For the download of AIA UV 1600 Å filtergrams (available at 24 s cadence), the syntax would be:

  ```
  aia.lev1_uv_24s[2013-04-11T06:45:00_TAI/150m@4m][1600]
  ```

- The HMI/SDO instrument provides differerent data products. For the LOS magnetic field maps, there exists a 720 s data product (which has a higher signal-to-noise ratio) and a 45 s data product (which is better suited when studying short-term magnetic field dynamics), see details at http://jsoc.stanford.edu/HMI/Magnetograms.html

- To download a sequence of 720 s LOS magnetic field images over a time range of 60 min from the given start time, the search input would be:

  ```
  hmi.M_720s[2013-04-11T06:45:00_TAI/60m]
  ```

- For the download of HMI continuums images for the same time range, use:

  ```
  hmi.Ic_720s[2013-04-11T06:45:00_TAI/60m]
  ```

# Solar SoftWare (SSW) IDL

- **Solar Dynamics Observatory (SDO)**

  - **Download of SDO data:**

    - 2) VSO (Virtual Solar Observatory): see previous slides

    - Examples for the HMI data search with VSO:

```
hmi = vso_search( '2019/10/22 14:00:00', '2019/10/22 14:30:00',$
instr = 'hmi', physobs = 'LOS_magnetic_field' )
dir = '/home/kad/Teaching/Dataanalysis/hmi_data/22-10-2019/'
status = vso_get( hmi, out_dir = dir )
print, sel.fileid
f = file_search( dir + '*.fits' )
read_sdo, f, index, data
index2map, index, data, map_hmi
```

    - With this sequence of commands, we identify for the given time range (start/end) the HMI LOS magnetic field data available (keywords `instr` and `physobs`), and download them into the directory (keyword `out_dir`).

# Solar SoftWare (SSW) IDL

- **Solar Dynamics Observatory (SDO)**

  - **Download of SDO data:**

    - More information to VSO and the specifications of the data search are available on the VSO webpage: https://sdac.virtualsolar.org/ .
    - An advantage of the download via VSO is that it can directly be included in an IDL
    - program, and can thus be automatized, and that it searches for data from many different sites and and instruments.
    - The disadvantage is that the data download can take long and that an IDL session needs to be open.
    - For SDO data and in particular bigger data volumes, JSCO ist to be preferred.

# Solar SoftWare (SSW) IDL

- **Solar Dynamics Observatory (SDO)**

○ **AIA cutout und JHelioViewer:**

  - Options for getting a quick but still extensive overview on the SDO and AIA observations (without necessity of data download) are offered by the AIA cutout-service and the interactive JHelioViewer.
  - The JHelioViewer is a visualization software based on JPEG2000 compression standard (enabling data in highly compressed, quality-progressive, region-of-interest-based stream), and allows interactive combination and overlay of different data. It is very useful for a first check of the data for an event, and also enables to produce presentation-ready movies:
  - http://www.lmsal.com/get_aia_data/
  - http://www.jhelioviewer.org/
  - However, note that these data are not perfectly suited for proper scientific analysis, for scientific analysis better use the data downloaded from JSOC.
  - A quick multiwavelength view of the Sun can be also obtained at http://solarmonitor.org
  - At https://helio.cfa.harvard.edu/trace/SSXG/ynsu/Ji/sdo_primer_V1.1.pdf you can find a tutorial with a detailed description of the SDO data search and data analysis.

# Exercises

**Exercise 17: JHelioviewer:** Load SDO/AIA 211 Å data for September 6, 2011 from 22:00 – 22:30 UT.
a) Use the options for base difference and running difference images to display the data. What do you see? Which images are better suited to study the dynamics of the event and why?
b) Produce movies for the direct images, base-difference images as well as running difference images and include a time stamp to the field-of-view. Adjust the image layer settings to for enhanced visibility of faint, dynamic structures.
c) Load STEREO-A or -B/EUVI as well as COR1 data. What do you see. Visit [https://stereo-ssc.nascom.nasa.gov/cgi-bin/make_where_gif](https://stereo-ssc.nascom.nasa.gov/cgi-bin/make_where_gif) to determine where SDO and STEREO are located with respect to each other

# Exercises

**Exercise 18: JHelioviewer:** Load SDO/AIA 193 Å data for February 15, 2011 from 01:15 – 03:00 UT. Change the cadence of the data to 1 min.
Display AIA filters that reflect the different layers of the solar atmosphere including the HMI magnetogram on top of each other.
Use the opacity slider to connect structures observed in different wavelengths.
What do you see around the flare region?
To which areas in the magnetic field is the flare emission related to?

# Solar SoftWare (SSW) IDL

- **GOES (object oriented)**

  - The Geostationary Operational Environmental Satellites (GOES) measure the spatially integrated soft X-ray flux of the Sun in two wavelength bands (0.5-4 Å and 1-8 Å).
  - The data are tyically available in 2-sec (or 3-sec) and 1-min integration.
  - The X-ray importance classification of solar flares is based on the peak flux measured in the GOES 1-8 Å channel.
  - The GOES classes in increasing order are: A, B, C, M, X.
  - These are logarithmic classes (i.e. each class differs by the next by a factor of 10), and are followed by a number that gives the multiplication factor.
  - For example: X4.3 = 4.3 x $10^{-4}$ W $m^{-2}$, C7.8 = 7.8 x $10^{-6}$ W $m^{-2}$.
  - The GOES software in SSW ist programmed object-based.
  - In traditional programming, there is a distinct separation between programs and data, which are accessed by these programs.
  - In object-oriented programming, this border is loosened: an object can contain both, data and the programs (methods) that can be applied to the data.

# Solar SoftWare (SSW) IDL

- **GOES (object oriented)**

  - In applying a method to an object, the „method invocation operator“ (->) is used.
  - Syntax:

    Definition of an object: `obj = obj_new( 'classname‘ )`
    Application of a procedure method: `obj -> methodname [, parameter]`
    Application of a function method: `result = obj -> methodname( [parameter] )`
    Example of GOES. To read a GOES light curve:

```
gobj = obj_new( 'goes' )  ; Creates a GOES light curve object
gobj -> read, ’28-Oct-03 08:00’, ’28-Oct-03 18:00’, /GOES10
```

    ; Reads 3-s data from the GOES10 satellite (read method)
    Plotting of a GOES light curve:

```
gobj -> plot  ; Plots the light curve of the data read in (plot method)
gobj -> plotman  ; Plot the light curve of the data in a GUI (plotman method)
```

    Checking of the object settings:

```
gobj -> help
```

# Solar SoftWare (SSW) IDL

- **GOES (object oriented)**

  - Extract the GOES data and the corresponding time vector from the object:

```
data = gobj -> getdata( )  ; Extracts both GOES channels, 2D array
low = gobj -> getdata( /low )  ; Extracts only the “low”-energy channel
                                 (1–8 Å), 1D array
high = gobj -> getdata( /high )  ; Extracts only the “high”-energy channel
                                   (0.5–4 Å), 1D array
times = gobj -> getdata( /times ); Extracts the corresponding time vector
utbase = gobj -> get( /utbase )  ; Extracts the corresponding UT basis (start
                                   time)
utplot, times, low, utbase, /ylog, yrange=[1e-7,1e-2]
; Plots the low channel data as function of time (UT plot format)
outplot, times, high, utbase  ; Overplots the high channel data
```

# Solar SoftWare (SSW) IDL

- **GOES (object oriented)**

- There exists also a graphical user interface (IDL widget) to work with GOES data. It is started with the command `goes`
- One can also extract a list of SXR flares (event list) from a GOES object for the defined time range, e.g.:

```
gev = gobj -> get_gev('06-Sep-2017', '10-Sep-2017‘,
/structure, /class_decode, /show, file='output.txt‘)
```

- You can find further information on: http://hesperia.gsfc.nasa.gov/rhessidatacenter/complementary_data/goes.html

# Exercises

**Exercise 19:** Read the GOES X-ray data for 2017 September 10, 6 UT to September 11, 18 UT and plot them using `plotman`. Extract the data of the 1–8 Å channel and plot them using `utplot`. Choose a logarithmic scaling for plotting the X-ray fluxes. Calculate the maximum of the des 1–8 Å flux and when it occurs.
What is the corresponding GOES flare classification?
In the next step, plot only the range 2017 September 10, 15–23 UT.
Use the keyword `timerange` to specify the x-range to be plotted (or the keywords `tstart` and `tend`).
For further info and keywords for the SSWIDL procedure `utplot` use: `xdoc, ’utplot’`

# Solar SoftWare (SSW) IDL

- **World Coordinate System (WCS) Routines**

  - The FITS World Coordinate System (WCS) (Greisen and Calabretta, 2002) is a formal system for embedding coordinate information within FITS files.
  - At its simplest level, the WCS is an extension of the coordinate system described in the original FITS paper (Wells et al., 1981) with some ambiguities resolved, and with a structure that allows the system to be extended to handle more complicated kinds of data.
  - The most important of these extensions is the system of projections for spherical coordinates (Calabretta and Greisen, 2002), which itself has been adapted for solar coordinates (Thompson, 2006).
  - Another extension has been published for spectral coordinates (Greisen et al., 2006), and extensions are in the works for distortions (Calabretta et al., 2010) and time axes (Rots et al., 2010).

# Solar SoftWare (SSW) IDL

- **World Coordinate System (WCS) Routines**

  - WCS tutorial by William Thompson: https://hesperia.gsfc.nasa.gov/ssw/gen/idl/wcs/wcs_tutorial.pdf
  - Data to work through the tutorial: `/sswidl/scratch/kad/` SDO and STEREO fits files