README
SPT/EBEX IDL simulation package

Catherine Laflamme

August 2008

Abstract

These simulations were created as an efficient method to produce accurate and realistic data structures for both the South Pole Telescope (SPT) and the E and B Experiment (EBEX). The goal of the simulations is to provide data analysts with data where we know exactly what is contained within it, so we can see the effects of processing the data. Included in the simulations are a simulated CMB and simulated galaxy clusters from Laurie Shaw [1], atmospheric effects taken from the Kolmogorov power spectrum,a unique power spectrum of instrument noise, as well as the stokes parameters measured using a rotating half wave plate and fixed polarization grid. The simulations are designed to run in IDL; the program most commonly used by project collaborators. This README file provides a detailed outline of each code, including the purpose of each individual code, the method or algorithm applied, noting any externally called programs and each programs proper inputs and outputs.

[1]Department of Physics, McGill University, Montreal QC

Contents

1 SPT Pipeline

The goal of the SPT pipeline was to create a simulated timestream which would include simulations of the CMB, galaxy clusters, realistic instrument noise, and the effects of atmospheric fluctuations. These timestreams should be saved in such a manner that any SPT data analyser would be able to easily access it. Polarization simulations can be easily adapted to fit the SPT model to provide simulations for the future polarimetry to be performed by SPT. The end of this pipeline is an standard SPT data structure which includes simulated sky information. The entire simulation can be run through the program spt_timestream.pro which takes no arguments and intself calls on the following programs.

1.1 get_laurie_data.pro

PURPOSE: The function get_laurie_data.pro is designed to read in information from Laurie Shaw's fits data files, into a usable IDL variable. The resulting images will be tiled in a checkerboard to cover an area of sky large enough to cover the SPT scan.

METHOD: This program calls on *readfits.pro* which reads a .FITS file into an IDL variable.

INPUTS:

Input .fits files.

OUTPUTS:

SKY_DATA[2] - an anonymous structure containing the map of the CMB and SZ effects, in units of Y, resolution in arcminutes/pixel, number of pixels, and degrees subtended by the map

1.2 get_spt_scope.pro

PURPOSE: To create an array which will completely define the bolometer array, namely, by giving the offsets of bolometers from the central pixel.

METHOD: This program calls on *read_bolo_info.pro* which reads in x and y offsets from the SPT directories. From the data structure we find only the 'good' bolometers (not flagged out for this observation) and we keep only the information on these bolometers.

INPUTS:

DATA - An existing SPT data structure.

[2]Note: any SMALL CAPITALIZED font indicates an IDL structure variable. These codes pass information through these IDL variables.

OUTPUTS:

SCOPE An anonymous structure containing an array of x and y offsets for each bolometer. Of dimension 2 x number bolometers.

1.3 laurie_sky_image.pro

PURPOSE: The function laurie_sky_image.pro is designed to read in information from sky_data and scope and turns the Y-Map into a smoothed map in units of μK.

METHOD: This program first changes the Y-map into a map in μK. This is done by the equation [3]:

$$\Delta Y = y \left(e \frac{e^x + 1}{e^x - 1} - 4 \right) T$$

At 150 GHz $x = 150/56.84.$ and T is the temperature of the CMB in μK. The next task is to smooth this map. Taking the FWHM from the structure, scope, I call on the program written by Micheal Fogel, *smooth_sky.pro*, to smooth with a gaussian beam of this FWHM.

INPUTS:

SKY_DATA
SCOPE

OUTPUTS:

SKY_IMAGE An anonymous structure containing the map of the CMB and SZ effects, in units of μK, resolution of the map in arcminutes/pixel, number of pixels and degrees on one side of the map.

1.4 make_spt_scan.pro

PURPOSE: The function make_spt_scan.pro is designed to read in information from an existing SPT data structure to reproduce its scan strategy.

METHOD: This program first copies the RA/DEC poiting of the scan in the input data structure. Then using *zero.pro* it excludes any erroneous 0 points (where the telescope missed information) and interpolates from the nearest neighbouring non-zero points. The next step is to create a list of the time of each point(each point differeing by 1/(data rate)). We can then transform our pointing into az and alt using the program *eq2hor.pro* to have streams in both telescope and sky coordinates.

INPUTS:

[3]see asto-ph/0210667

DATA - A SPT data structure containing the scan strategy you want to simulate.
ARRAY- a 2x6 array containing (by row) the day, month, year, hour, minute, second of the start and end of your scan. (start times in the first column, end times in the second.)

OUTPUTS:

SCAN An anonymous structure containing streams of the central pointing in RA/DEC and AZ/ALT. As well the this structure contains the time at each sample, the data rate and the total time of the scan.

1.5 make_sky_timestream.pro

PURPOSE: The function make_sky_timestream.pro is designed to turn the information stored in sky_image into a timestream, based on the scan information given from the structure scan, and scope information given from the structure scope.

METHOD: The first step in making the timestream is to associate with every RA/DEC in the sky, a pixel in the simulated sky image. To do this I will first center the RA and DEC vectors in my scan at 0 by subtracting the median value. the RA vector is scaled by a factor of $\cos(declination)$ to convert from sky coordinates into square coordinates. Based on the resolution of my sky image (ie. the arcminutes on the side of one pixel) I am able to convert these newly created RA and DEC vectors into pixels. Because I want the center of my scan to be in the center of the map, and not at the [0,0] pixel, I add the mean pixel number to both vectors, resulting in X and Y pixel coordinates of my simulated scan. Individual bolometer offsets, once multiplied by the same resolution as above, can be directly added to these X and Y vectors, to give 2 arrays of size [n_bolos, n_datapoints], each corresponding the the X and Y cooridnates of each bolometer at each time. The last step is to find the data point associated with the X and Y coordinated in my simulated sky image, and associate that data point with this specific time, for this specific bolometer. This will give me an accurate timestream for each bolometer across the array.

INPUTS:

SKY_IAMGE
SCOPE
SCAN

OUTPUTS:

TIMESTREAM An anonymous structure containing the stream, an array of size n_bolo x n_datapoints which contains the timestream of each bolometer.

1.6 add_noise2stream.pro

PURPOSE: The function add_noise2stream.pro is designed to add, on top of timestreams containing data of the CMB and SZ effect, realistically simulated noise.

METHOD: This program calls on a code written by Micheal Fogel *t_noise.pro*. This program creates a noise timestream by first creating the noise power spectrum which includes 1/f noise, white noise, and takes into account bolometer and electornic roll off. Once the power specturm is created, it the phases are randomizes, and then transformed back into time space. Taking caution of normalizations that IDL uses in an FFT we have a noise timestream. This process is repeated for each bolometer stream.

INPUTS:

TIMESTREAM
White noise level in $\mu K \cdot \sqrt{s}$
1/f knee, bolometer time constant(3 dB), and electronic time constant (3dB) roll off levels in Hz

OUTPUTS:

TIMESTREAM The same strucutre as was inputted, with noise added into the timestreams.

1.7 add_spt_atmosphere.pro

PURPOSE: The function add_spt_atmosphere.pro is designed to add to an already created timestream, information on an atmosphere moving at constant velocity.

METHOD: Using information on the height of the atmosphere, and the resolution of the map, we can turn a speed from m/s at the height of the atmosphere into pixel/s. The indices of the atmosphere image, instead of staying constant, are shifted through time by this speed. The information of the atmosphere image is then simply added to the old timestream.

INPUTS:

ATMOSPHERE - an array of an atmosphere image, in units of μK. This array is the output of the program make_spt_atmos.pro, which is an already used SPT code.
TIMESTREAM
speed_x/y - speeds of the atmosphere

OUTPUTS:

TIMESTREAM The inputed timestream structure with added atmosphere information.

1.8 make_spt_map.pro

PURPOSE: The function make_spt_map.pro is designed as to map out information created in the simulated timesreams. Not crucial in the pipeline, this program acts as an error check; a way to visually check what information is carried in the timestream.

METHOD: Because the X and Y pixel of each bolometer is still stored in the input strucute TIMESTREAM a map is easily created by just putting the infomration stored in the timestream, in the map at the pixel specified above. By keeping track of map hits and dividing by this factor in the final step, we efficiently create an accurate map from our timestream. We can display this final map using the procedure *spt_plot.pro.*

INPUTS:

TIMESTREAM
SCOPE
SCAN
N_pix - number of pixels you want on the side of your output map.

OUTPUTS:

MAP An anonymous structure containing the map of whatever information was contained in the timestream, and the map which recorded the number of hits per pixel.

1.9 make_spt_struc.pro

PURPOSE: The function make_spt_struc.pro is to save the timestream information in the standardized SPT data structure.

METHOD: The procedure first turns the timestream which is in units of μK into units of ADC counts. This is done by a constant conversion factor taken from the chicago database. There is a distinct conversion factor specific to each bolometer.[4] Then I simply take an existing structure which is already in the accepted format and overwirte the timestream and pointing information with the information pertaining to my simulation.

INPUTS:

DATA - an existing SPT data structure
TIMESTREAM

OUTPUTS:

DATA An SPT formatted data structure containing simulated timestreams.

[4]see get_chi_cal.pro for more information

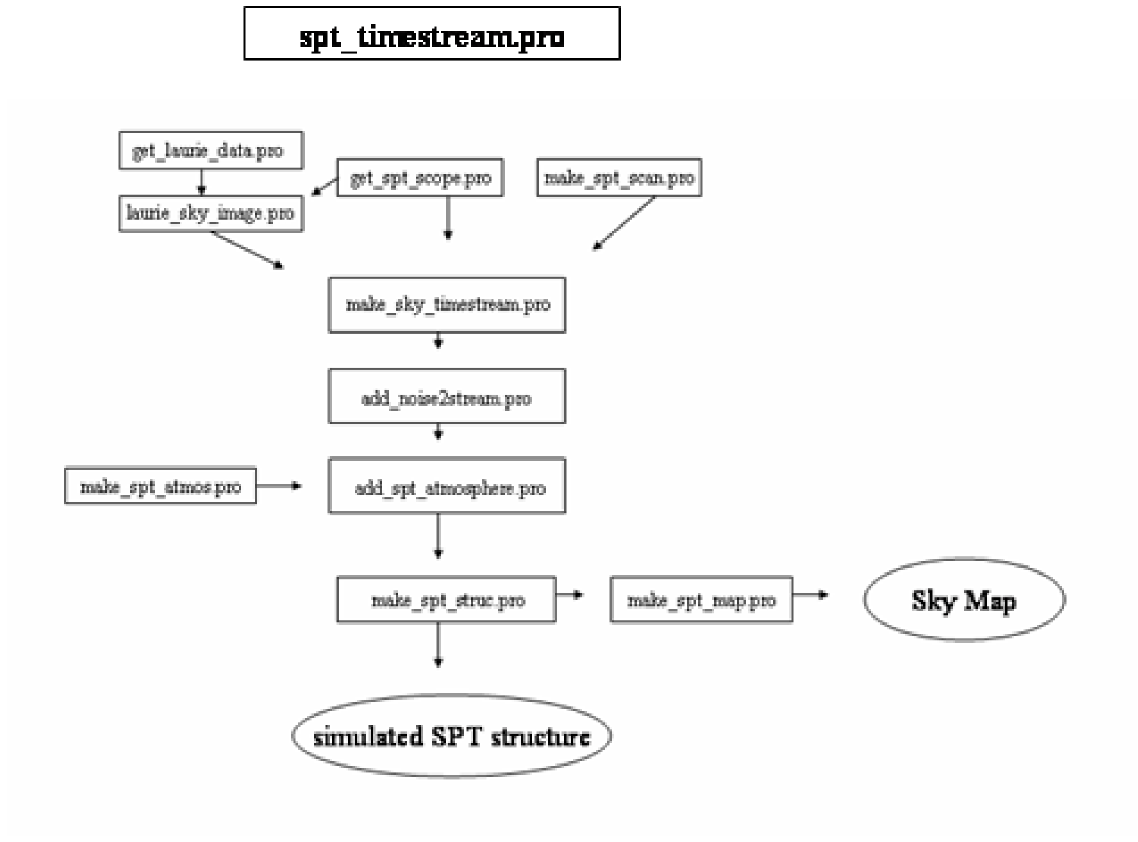

Figure 1: The complete SPT simulation pipeline

2 EBEX PIPELINE

The goal of the EBEX pipeline was to create a simulated timestream which would include simulations of the polarization of the CMB, which uses a realistic EBEX scan strategy. These timestreams should be saved in such a manner that any EBEX data analyser would be able to easily access them. The end of this pipeline is a standard EBEX data structure which includes simulated sky information.

2.1 ebex_scan.pro

PURPOSE: The function get_ebex_scan.pro is to read, into an IDL variable, the binary files which are the output of Sam Leech's C-code.

METHOD: The procedure reads the input binary file with the IDL function read_binary.

INPUTS:

none

OUTPUTS:

SCAN An accurate EBEX scan stream of ra, dec, beta (angle of the focal array) , alt and az.

2.2 ebex_bolos.pro

PURPOSE: The function ebex_bolos.pro is to read in az/el offsets of the ebex focal plane at boresight point of (0,0) and convert them into ra/dec offsets.

METHOD: The procedure uses a program from the IDL library altaz2hadec.pro, to transform between coordinate systems.

INPUTS:

none

OUTPUTS:

SCOPE An array of ra/dec and az/alt offsets for each bolometer, given in units of arcminutes.

2.3 ebex_pointing.pro

PURPOSE: The function ebex_pointing.pro applies the az/el offsets which apply at (0,0) to any point on the sphere (namely wherever the boresight is pointing), as well as rotates the entire focal array by angle beta.

METHOD: The procedureuses calls on two prewritten procedures, *apply_pixel_offset_scan.pro* and *altaz2hadec.pro.* The former is available in the SPT repository, the latter on the NASA idl library. To rotate the focal array by the parameter beta, the coordinates are transformed into a coordinate system where the focal plane of bolometers lies on the x,y plane. The coordinates then undergo a 2d rotation by the angle beta, before transforming back into the oringinal coordinate system.

INPUTS:

lst -The lst at every time in the timestream.
az/el/beta - The streams of boresight pointing, and angle beta as taken from the structure SCAN. SCOPE - The structure Scope.

OUTPUTS:

POINTING An array of each bolometers pointing information for each time in the simulated scan.

2.4 make_synfast_timestream.pro

PURPOSE: The function make_synfast_timestream.pro is to create a timestream of CMB polarisation taking into account the half wave plate and fixed grid used in EBEX.

METHOD: The procedure uses POINTING to find the ra/dec of each bolometer at each time. Using a HEALpix IDL procedure ang2pix_ring.pro [5] which will associate for each ra/dec an index in the synfast output array. This allows me to associate the 3 stokes parameters, I, Q and U with each bolometer at each time. Then I call on a function *polarization_fast.pro* which takes into account the HWP and fixed grid.

2.4.1 polarization_fast.pro

polarization_fast.pro uses Jones calculus to compute the effects of the rotating HWP and fixed grid on polarized light. When given inputed stokes parameters I, Q and U it will output the temperature recorded by an EBEX detector.

INPUTS:

POINTING - Pointing information of each bolometer.
SYNFAST_ARRAY - information from synfast, written in an IDLvariable from the HEALpix procedure[6] *read_fits_s.pro.*
scope
scan

[5]http://healpix.jpl.nasa.gov/html/idlnode31.htm
[6]http://healpix.jpl.nasa.gov/html/idlnode37.htm

speed / scramble - Optional keywords which allow for altering of the angular speed of the HWP and altering the initial position of the fast axis with respect to the stationary y axis.

OUTPUTS:

TIMESTREAM A data structure containing the polarisation stream as detected by EBEX, the data rate, and the number of bolometers.

2.5 plot_polarisation.pro

PURPOSE: The function plot_polarisation.pro is to plot maps of A_x and A_y polarisation, from the simulated timestreams. Again this is just an easy way to error check the timestreams, and see what information is contained in them, before any noise subtraction etc...

METHOD: The procedure starts by first calling on the procedure *plot_polarisation_one.pro* which takes into account, backwards, the HWP and fixed grid. This will effectivly take the temperature recorded by EBEX and output the A_x and A_y polarisation of the incident photon. Then using the same code writted for *make_sky_timestream.pro* I obtain pixel coordinates for each bolometer at each moment in the timestream. Once I have these coordinates, using the timestream information I can plot a map.

2.5.1 plot_polarisation_one.pro

plot_polarisation_one.pro uses Jones calculus to compute the effects of the rotating HWP and fixed grid on polarized light, just as *polarization_fast.pro* did for the case when we were making the timestream. By taking the inverse of the matrix used in that procedure we are effectvly computing the sequence in reverse order. Now we begin with one recorded temperature, we run first through the fixed plate, then the HWP giving us the incident beam A_x and A_y polarisation information.

INPUTS:

TIMESTREAM
SCOPE
SCAN
speed - Optional keyword which allows for altering of the angular speed of the HWP.

OUTPUTS:

MAPS A data structure containing maps of the A_x and A_y polarisations.

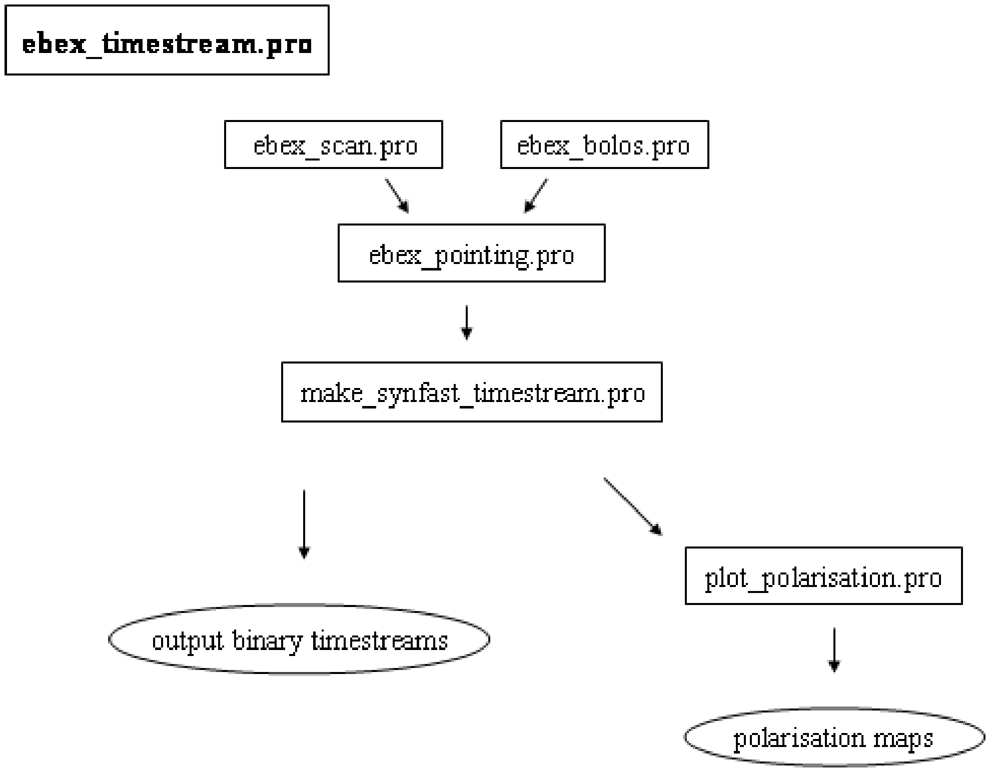

Figure 2: The complete EBEX simulation pipeline

A A run through the simulation pipelines

A.1 SPT

```
IDL> spt_timestream
```

spt_timestream calls the following sequence of commands:

```
IDL> scope = get_spt_scope()
```

Scope is an anonymous array which includes information of the bolometer offsets (in arcminutes) and the FWHM of the beam (arcminutes).

```
IDL> scan = make_spt_scan(spt_data)
```

Scan includes the boresight pointing in ra/dec and in az/alt. The scan corresponds to the scan in the input spt data file.

```
IDL> data = get_laurie_data()
```

Data is an anonymous structure which includes: a tiled map which includes CMB and galaxy clusters, in units of y, the resolution of the map (in arcminutes/pixel) and the number of pixels to each side of the map.

```
IDL> image = laurie_sky_image(data,scope)
```

Image contains the sky map(CMB and clusters) smoothed to the FWHM of the scope's beam, in units of microK.

```
IDL> stream = make_sky_timestream(scope, image, scan)
```

Stream is an anonymous structure including: streams for each bolometer as it pans the image, in units of microK, stream containing x and y pixel locations on the image map at each time for each bolometer, and the data rate at which the image was sampled.

```
IDL> stream_withnoise = add_noise2tstream(stream, w_noise = 300, f_knee=1, bolo_roll=18, lp_roll = 40)
```

Stream_withnoise is the same structure as stream with accurate instrument noise added individually to each bolometer's timestream.

```
IDL> atmos = make_spt_atmos(10.e6, 1300, 6005, 0.25, el_av_degrees = 52.4)
```

Atmos gives a map of an atmosphere, based on the Kolmogrov spectrum, with an amplitude of 10K, at a height of 1300m above ground level, with 6005 pixels to a side and at a resolution of 0.25 arcminutes. 52.4 is the average elevation of the scan in degrees.

```
IDL> stream_withatmos = add_spt_atmoshere(stream_withnoise, atmos, 1.08, 1.08)
```

Stream_withamos keeps the same structure as stream_with noise but adds the timestream of atmosphere temperature onto the existing timestream data.

```
IDL> new_data_struc = make_spt_struc(stream_withatmos, spt_data)
```

New_data_struc overwirtes the spt_data structure with the simulated timestream, and has the standard SPT data format.

A.2 EBEX

IDL> ebex_timestream Ebex_timestream calls the following sequence of commands:

```
IDL> scan = get_ebex_scan()
```

Scan includes ra/dec/beta of the boresight for all times in my timestream

```
IDL> scope = ebex_bolos()
```

Scope includes offsets of the bolometers in ra/dec and in az/alt.

```
IDL> pointing = ebex_pointing(scan.az, scan.alt, scan.lst scan.beta, scope)
```

Pointing contains ra/dec coordinates of every bolometer at every time.

```
IDL> read_fits_s, ‘./synfast_output.fits’, hdr, data
```

Data now contains all information of the polarization from the synfast program.

```
IDL> stream = make_synfast_timestream(pointing, data)
```

Stream is a timestream including: the measured polarization signal, in microK, at each time for each bolometer, and the data rate at which it was sampled.

```
IDL> openw, lun, ‘directory/bolo_’+num+‘.dat’
IDL> writeu, lun, stream
IDL> free_lun, lun
```

This sequence of commands writes the stream to a binary data file, ready to be included in a standard EBEX DIR file.