SharpInteg Instructional Manual

1. Summary of Code

Sharpinteg is a program that takes a cycle of half-wave plate (hereafter hwp) measurements from Sharp and process it to find I,Q,and U along with the corresponding errors. It effectively replaces Integ and the majority of Stokes (save the sorting of points from multiple sets of hwp cycles) from the Hertz analysis method.

The first step in the program is reading and processing the command line arguments (see user instructions section). Next the raw data is read in and stored along with the rgm/pixel mask file. After all data for a file is read, all bad pixels are set to a value of zero so they will not affect the f calculation while the rest of the pixels are multiplied by their gain. The chop is then demodulated. This procedure checks the chop offset to see if it is at the high or low chop position (within a certain tolerance set in program head at 4") and that is is at the left or right nod position (done by checking that azerr and elerr are within a tolerance of 5") Any data sample that fulfills those requirements is given a nonzero weight which is equal to the number of samples at that chop and nod position. Weights for samples that are not within chop or nod tolerances are set to 0. Sample in the 'hi' position have nonzero weights while the 'lo' are negative. Anytime that the chop state switches, i.e. it goes from hi to lo or vice versa, the counts are checked to see if they are within a factor of 2 of each other (locount > hicount/2 && hicount > locount/2). If so then the number of 'chops' is increased by 1 and weights are set to 1/hi(lo)count for both the high and low samples. If this is not true, the code assumes only half of the chop samples have been checked and continues on until checking samples until the chop state switches again. Once it is through all samples it writes a new strip chart of data that is at the chop sampling rate. The values for this data are

$$choppeddata = \sum_{i=1stsample}^{lastsampleinchop} raw * weight. \tag{1}$$

The next step is the calculation of f. f is the gain factor between the h and v arrays. It is calculated by taking all of the samples from a particular hwp position and fitting to the line

$$v = ah + b \tag{2}$$

using numerical recipe code. After this is done for all four hwp positions, the median value is taken and f is set to the inverse value of the median.

After the calculation of f, the h and v array samples are combined as follows

$$poldata[i = 0 - 143] = h - fv \tag{3}$$
$$poldata[i = 144 - 287] = h + fv. \tag{4}$$

Once this is complete, the samples that are left beam are summed and the right beam is summed. The mean and error are then computed for the left and right beam. Deglitching is then performed by either statistical deglitching with a threshold value set in the code (a value of 3 in the current version) or using chauvenet's theorem. The samples which do not satisfy whichever criterion was selected are removed and the sum, mean, and error are recalculated. The sample value and error are then set to

$$nodsignal[i] = leftmean[i] - rightmean[i] \tag{5}$$
$$noderror[i] = sqrt(leftvar[i]/leftcount[i] + rightvar[i]/rightcount[i]) \tag{6}$$

where ****mean is the mean for the **** beam, ****var is the variance for the **** beam, and ****count is the number of samples in the ****beam and i is between 0 and 287. The polarization signal is then set to the nod signal for values of i between 0 and 143. The flux signal is set to the mean value from the signals from each hwp position (i values between 144-287). The flux signal is calculated as follows

$$fluxerror = sqrt(\frac{(\sum_{i=0}^{4} noderror * noderror) - (\sum_{i=0}^{4} nodsignal * nodsignal)/4}{3} \tag{7}$$

U and Q are then calculated with the following method

$$Q = 0.5 * (nodsignal(hwp = 0) - nodsignal(hwp = 45)) \tag{8}$$
$$U = 0.5 * (nodsignal(hwp = 67.5) - nodsignal(hwp = 22.5)) \tag{9}$$

Errors on Q and U are the noderrors processed through the above equation. All values for bad pixels are then set to -1000000.0. The data is then written out to a fits file entitled solve.fits.

2. Command Line Arguments

The command line arguments for version 1 are as follows

sharpinteg hwp=0file hwp=22.5file hwp=45file hwp=67.5file -r rgmfile -c -f # -t # -a # # -nh -nc

0 -c selects chauvenet theorem, -f # sets f to be #, -t sets the chop tolerance to #, -a sets the pixel offsets (x,y) to be (#,#), -y # file causes the program to get the smooth tau for date # and from tau data in file, -nh causes the program to not check hwp angles (needs to be used for January 2006 data), and nc stops the program from exiting when chop and nod do not agree.

For Sharpinteg version 2, the command line arguments are as follows

sharpinteg file.fits -r rgmfile -c -f # -t # -a # # -q -o -d str -y # str -nh -nc -sil

All duplicate arguments from version 1 are the same, -q sets quicklook mode printing the h and v arrays values in seperate fits files, -o flag prints out cal/debug values, -d reads in string date to print to file from -o, and -sil is silent mode which suppresses some deglitching debug statments to the screen.