**Example MCMC_SD Run**

The following is an example of how to derive astrometric parameters using the mcmc_sd codes. In this example we will derive the astrometric parameters of Epsilon Indi BC's photocenter based on the CTIOPI astrometric data set of Dieterich et al. 2018. That paper actually used a slightly different code to combine two datasets, and we will only be using one of them here. To compensate for the lack of data we will set the trigonometric parallax fixed to the value obtained by Dieterich et al. 2018, 276.88 mas. This will also illustrate how to set fixed parameters. The actual running of the MCMC will take several hours, so you may want to do that overnight and continue with the example on the next day.

It is recommended that users also refer to the explanations in instructions_mcmc.pdf throughout this example.

**Procedure**

1 - Set up a main directory containing all the mcmc_sd .pro files as well as the data file,

```
ctiopi_mcmc_data.txt
```

2 - In this example we will be running three simultaneous IDL sessions to speed up the code. Under the main directory create the directories for sessions 2 and 3:

```
mkdir session2
mkdir session3
```

Then copy all files to these subdirectories or alternatively put them in your IDL library. If you cannot run multiple IDL sessions or would rather run a single session then continue with a single session, but it will take roughly triple the time for the mcmc to run.

3 - The parameters inputbounds and parbounds define the range in which initial values for the chains should be generated and the range in which the chains are allowed to fluctuate, respectively. See instructions_mcmc.pdf for details. Because these parameters can be lengthy, let's define them as variables rather than entering them on the mcmc wrapper call. Cut and paste these definitions to all three IDL sessions:

```
IDL> inputbounds_var = [[276.88, 276.88], [10.7d,11d], [-6.8d, -6.5d],
      [150, 350d], [3000d, 7500d],[0.35d, 0.6d],[2.45d6, 2453216.77],
      [120d*!pi/180d, 180d*!pi/180d], [0d, 2d*!pi], [70d* pi/180d,  !pi
      /2d]]
IDL> parbounds_var = [[276.88, 276.88],[10d, 11.5d],[-7d, -6d],[100d,
      600d], [1000d, 8000d],[0.2d, 0.7d],[2.448e6, 2453216.77],[100d*!
      pi/180d, 200d*!pi/180d],[0,2d*!pi],[40d*!pi/180d,3d*!pi/2d]]
```

These parameters are appropriate for Epsilon Indi BC. Note that parameter index 0, the trigonometric parallax, has the same value of 276.88 for both elements. This declaration makes trigonometric parallax a fixed constant not to be explored by the mcmc. Any combination of parameters can be set to a fixed value. Also, if trigonometric parallax is fixed then the parallax factors in the data file can be junk numbers.

5 - We are ready to run the code. Cut and paste the following into all three IDL sessions. DO NOT PRESS ENTER YET.

```
IDL> result = mcmc_sd_wrapper_v1(astrofile = "ctiopi_mcmc_data.txt",
      n_chains = 17, n_iter = 10000, n_runs = 200, ra_jitter = 0,
      dc_jitter = 0, scalingpar = 3, stepscale = 1, stepmultiplier =
      10, inputbounds = inputbounds_var, parbounds = parbounds_var,
      /tictoc)
```

Take a few minutes to read the instructions file and understand what each one of these parameters does. Otherwise you will not know what is going on. If you are running a single IDL session then set n_chains = 50 ( and hunker down for a 30 hour run!)

6 - Now that you thoroughly understand the call in item 5, press enter in all three IDL sessions! The code will run for about 10 hours (MacBook Pro, 2.8 GHz Intel core i7, 16MB RAM), with each session taking roughly 100% of a processor core. Rudimentary histograms of the probability density functions of all astrometric parameters are produced at the end of each sub-run, set for 10,000 steps.

**Interpreting spider step output messages**

Refer to Appendix A of Dieterich et al. 2018 for an overview of the spider mechanism. The IDL terminal will produce copious lines of the form:

```
SPIDER ON THE MOVE!     200       3       4
SPIDER ON THE MOVE!     200       4       5
SPIDER ON THE MOVE!     200       9       3
     SPIDER FOUND NEW HOME!     299       3       4       3870.1048 ---->       1834.5907
     SPIDER FOUND NEW HOME!     299       4       5      0.53238440 ---->       0.39228140
     SPIDER FOUND NEW HOME!     299       9       3       379.75212 ---->       455.24986
```

The first line means that on step 200 of this sub-run chain 3 underwent a spider jump where parameter 4 (orbital period) jumped to a new location in parameter 4's allowed parameter space and that jump was provisionally accepted according the the standard Metropolis-Hastings acceptance criteria. The next two lines indicate that chains 4 and 9 also had spider jumps for parameters 5 (eccentricity) and 9 (inclination), respectively. The fourth line indicates that for chain 3 after 100 steps it was determined that the mean probability in the new location was better than the mean probability of the 100 steps before the jump. The chain will continue to evolve in the vicinity of orbital period = 1834.5907 days, as opposed to 3870.1048 days before the jump. The same format applies for the next 2 lines.

Alternatively, the message may be of the form:

```
SPIDER ON THE MOVE!     400       3       5
                    SPIDER RAN AWAY!     499       3       5
```

This means that the new location, which had been conditionally accepted on step 400, did not improve the mean probability of the next 100 steps. Chain 3 therefore "ran away" to its previous location before step 400 and continued to evolve in that vicinity.

7 - It's 10 hours later and the mcmc finished running! The histograms in the provisional plots should look a bit more localized, but don't worry if they are not too pretty. Their scaling is off and the chains from different sessions are not combined yet. To combine the chains enter the following in the top directory:

```
IDL> combined_results = reconstruct_multi4(3)
IDL> save, combined_results, filename="example_combined_results.sav",
     description="combination of 3 sessions using CTIOPI data. You could also
     cut and paste the values of parinputs and parbounds here for reference."
```

Where the second line saves your results, something that is NOT done automatically. You may now close or minimize IDL sessions 2 and 3. The chains are saved to the top directory and everything else will be done there.

8 - The next step is to plot the histograms representing the probability density functions of your results. Before we do that inspect the plots in "example_pdf_histogram_notrim.pdf". You will see that the distributions for the parameters closely resemble Gaussians, but there are residual chains that did not converge and indicate extremely low, but finite, probabilities away from the main Gaussian peak. These spurious chains must be trimmed out. The data for the plots in example_pdf_histogram_notrim.pdf are deliberately not included in the example distribution, so that you have no excuse to skip this trial run. So let's plot your results. To produce a simple untrimmed plot enter:

```
IDL> finalresult_v1,combined_results.chains,'your_pdf_histogram_notrim.ps'
```

The numbers on the top of each plot indicate the mean, median, and standard deviation, respectively. Check the instructions file and header of finalresult_v1.pro for instructions on how to change the binning for each parameter, if desired.

9 - We will now trim out non-convergent chains. This "noise" will likely look different in your plots than in the example plots, so adapt the trim criteria for something that will trim out bad chains in your plots in a manner analogous to what we will do here for the example plots.

First, look for a parameter where the spurious chains seem well localized and use that as a basis for trimming. In the example plots this occurs for proper motion of the declination, parameter 2. In that plot everything greater than -6.86 can be trimmed out:

```
finalresult_v1, combined_results.chains, 'example_pdf_histogram_trim1.ps',
trimstatement= "chains2 gt -6.86"
```

Now inspect the results in "example_pdf_histogram_trim1.pdf". There is an improvement, but we see that most Gaussians still have a very long tail of very low probabilities. This is clear in several parameters, but perhaps most clearly in the position angle of periastron, parameter 8. In addition to the first trim, let's trim out everything smaller than 300 in parameter 8. Note that we are saving the plots to a new file:

```
finalresult_v1, combined_results.chains, 'example_pdf_histogram_trim2.ps',
trimstatement= "chains2 gt -6.86 OR chains8 lt 300."
```

Now inspect "example_pdf_histogram_trim2.pdf". The last trim got us a good result. We can adopt those values as the mean, median, and standard deviation of each parameter.

As mentioned before, your case may be different. Also, if comparing to the results in Dieterich et al. 2018 keep in mind that this example uses less data and the result should not be as accurate or as precise as the published values. Keep trimming your results until you are satisfied that anything truly spurious has beed removed.

10 - Once the final trim criteria have been established we should save the trimmed data into an IDL .sav file. This can be useful later on for more advanced statistics. This is also done by finalresult_v1.pro:

```
IDL> finalresult_v1, combined_results.chains,
     'example_pdf_histogram_trim2.ps', trimstatement="chains2 gt -6.86 OR
      chains8 lt 300.", /savechains, chainsfile =
     "final_example_trimmed_chains.sav"
```

YOU ARE DONE INFERRING THE ASTROMETRIC PARAMETERS, WHICH ARE WRITTEN IN YOUR FINAL PLOTS! GOOD LUCK WITH YOUR SCIENCE!

You may, of course, wish to plot the astrometric motion for the parameters you just inferred, or solve for masses, etc. That is beyond the scope of this MCMC suite. Now that you have values for the 10 astrometric parameters equations 1 through 9 of Dieterich et al. 2018 and a bit of work is all you need to plot the astrometric motion. Equation 9 is transcendental and is solved by ecca.pro. You can modify the last few lines of that code to actually output the value for E, the eccentric anomaly, instead of an index number. See the header documentation for ecca.pro. If using ecca.pro be sure to use the mcmc wrapper function to establish the necessary common block. See the documentation for mcmc_sd_wrapper_v1.pro.