Some Notes on Climax

Updated, 20 March 2015

Erik M. Leitch

Contents

1 Quickstart Guide

1.1 Introduction

Currently, CLIMAX consists of a single binary, ${CLIMAX_SRC}/bin/**climaxRunManager** which is entirely driven by parameter files; no command-line options except the parameter file name are supported, so that runs can be exactly reproduced from the contents of the parameter file. The binary can be run like:

```
climaxRunManager file=myParameterFile
```

Additionally, a simple Python hook is provided with the code, which is compiled into the loadable module ${CLIMAX_SRC}/python/climaxPyTest.so (if PYTHON_INC_PATH is defined in Makefile.directives when the code is built). To use the code from Python, do:

```
import climaxPyTest
res = climaxPtTest.run('myParameterFile')
```

where 'res' will contain a dictionary of parameter values, 1-sigma confidence limits, and other information.

One goal of the code documentation is that it be self-describing as much as possible; this means that any error in the parameter file should generate a usage message with information on valid options, or instructions on how to fix syntactical errors. Run **climaxRunManager** with no options to get started.

1.2 Theory of Operation

The general structure of parameter files is designed around defining datasets and models, using the commands `adddataset` and `addmodel`. To see a list of recognized datasets (or models), just enter one of these commands with in invalid type. Datasets in CLIMAX are aware of which models can apply to them, and by default all models will be fit to any datasets to which they apply. This behavior can be modified by using the `exclude` command to exclude models from particular datasets (see § 1.3).

Once datasets and models are declared, **climaxRunManager** can do one of several things, depending on control parameters in the parameter file: by default it will run a Markov chain for any variable model components, but it can also display datasets, models and residuals, compute chi-squared for defined datasets and models, load and display output chains, or generate simulated data for a dataset type if the dataset supports it. What it does is determined by global control parameters, detailed in § 1.5.

1.3 Defining Datasets

Once a dataset is declared, the user can set individual parameters of the dataset by name. For example, to declare an interferometric UVF dataset, you would use:

```
adddataset type=uvf name=duvf;
```

which declares a new dataset of type 'uvf' and associates the name 'duvf' with it. Parameters of this dataset can subsequently be declared like:

```
duvf.file = /path/to/my/uvf/filename;
```

To see a list of recognized parameters for a dataset, just set an invalid parameter for that dataset.

If desired, models can be excluded from particular datasets by using the `exclude` meethod. For example,

```
duvf.exclude(model1);
```

would prevent `model1` from being fit to `duvf`. This allows greater flexibility in the way that chains are run. For example, for data from an instrument like the SZA, you can dramatically increase the speed of fitting cluster and point source models by loading the data twice, once with a uvmax of $2k\lambda$ (short-baseline data) and once with a uvmin of $2k\lambda$ (long-baseline data), fitting a point-source model to both, but excluding the cluster model from the long baseline data, which have no sensitivity to it anyway. This is because the cluster fitting will only require Fourier inversion of the short-baseline data, which can be coarsely gridded and quickly inverted, while the point-source models are fit directly on the Fourier plane and require no inversion of the models for either dataset.

1.4 Defining Models

1.4.1 Adding Models

Models can contain both parameters and potentially variable components. To see a list of recognized parameters and components for a model, just set an invalid parameter for that model.

If a component is assigned a specific value in the parameter file, it is held fixed at that value. For example

```
addmodel type=betamodel name=model1;
model1.beta = 0.8;
```

defines a β-model with a fixed value for the β parameter. If a component is instead specified with a prior, then that component is treated as variable. For example,

```
model1.beta = 0.6:0.8;
```

would define a β-model with a uniform prior of $\{0.6, 0.8\}$ for the β parameter. On a more subtle note, for the Metropolis-Hastings algorithm, Climax by default selects the center of any uniform prior as the starting mean for the jumping distribution. You can override this behavior by optionally specifying a starting mean explicitly. The specification

```
model1.beta = 0.65:0.6:0.8;
```

for example would set the same uniform prior but initialize the jumping distribution with a mean of 0.65. Alternately, you can assign a Gaussian prior, via:

```
model1.beta = 0.7 +- 0.1;
```

Model components with units must be specified in a valid unit for that component. For example, to specify the radio normalization of the β-model you could use:

```
model1.Sradio = -2000:-500 muK;
```

to set a prior in micro-Kelvin. Enter an invalid unit to see what units are supported for each model component. Additionally, CLIMAX will allow any numerical scaling from a recognized unit. For example:

```
m.m500 = 1e15:5e15 Msolar;
```

and

```
m.m500 = 1:5 1e15Msolar;
```

are both valid specifications for a model with mass parameter 'm500'.

1.4.2 Removing Models

Models can be removed from the data prior to any operations by using the `remmodel` keyword. Models are defined just as with `addmodel`, but are subtracted from the data prior to fitting or imaging.

1.4.3 Correlating Model Parameters

You can set any model parameter equal to another (symbolic) model parameter, provided they are of the same type. This links the two parameters together, with only the first parameter sampled by the Markov chain. For example:

```
addmodel type=betamodel name=beta;
addmodel type=ptsrc name=pt;

beta.xoff = -0.01:0.01 deg;
beta.yoff = -0.01:0.01 deg;

pt.xoff = beta.xoff;
pt.yoff = beta.yoff;
```

would fit a cluster and a point source, with the location of the point source fixed to the cluster center. This can be useful in other contexts too; for example, i n combination with model exclusion (see §1.3), it could be used to tie together physical model parameters when two datasets have an unphysical offset between them, e.g.:

```
adddataset type=uvf name=sz;
adddataset type=xrayimage name=xray;

addmodel type=betamodel name=msz;
msz.Sradio = -2000:0 muK;
msz.xoff = -0.01:0.01 deg;
msz.yoff = -0.01:0.01 deg;
msz.beta = 0.1:0.8;
msz.thetaCore = 0.1:1.0';

addmodel type=betamodel name=mx;
mx.Sxray = 0:2000 counts;
mx.xoff = -0.01:0.01 deg;
mx.yoff = -0.01:0.01 deg;
mx.beta = msz.beta;
mx.thetaCore = msz.thetaCore;

sz.exclude(mx);
xray.exclude(msz);
```

defines a beta model with independent offsets between the radio and xray datasets, but with linked `beta` and `thetaCore` parameters.

1.5 Global Control Parameters

In addition to commands declaring datasets and models, Climax parameters files support global parameters that control what Climax does with the parameter file.

1.5.1 Running Markov Chains

By default, Climax will attempt to run a Markov chain for any variable model components. The total length of the chain and the length of the burn-in sequence can be controlled by parameters `ntry` amd `nburn`, respectively.

By default, Climax will attempt to display parameter histogram plots, data and residuals with the best-fit model at the end of the chain. You can use the `dev` parameter to redirect the plots to device other than the screen (`/dev/null` or a hardcopy device). Climax uses `pgplot` for graphics and expects the usual pgplot device specifications. For example, to run a chain with 10000 total iterations and burn-in length of 3000, and redirect the plot to a postscript file `climax.ps`, do

```
ntry = 10000;
nburn = 3000;
dev = climax.ps/vcps;
```

1.5.2 Displaying Datasets and Models

If the `display` parameter is set to `true`, Climax will instead attempt to display any defined datasets, along with residuals against any defined models. Naturally this requires that models be defined with fixed parameters, and Climax will report an error if any required model parameters are not defined, or are specified with priors.

In addition, some datasets support their own `display` parameters, which are executed when the datasets are loaded, independent of what the parameter file instructs Climax to do. Thus:

```
adddataset type=uvf name=duvf;
duvf.display = true;
```

will cause the dataset to be displayed first, whether running a Markov chain or creating residual plots.

1.5.3 Outputting Chains

By default, parameter chains are stored internally, and are displayed as histogram plots at the end of a chain. The chain is not however written to disk unless you explicitly instruct Climax to do so. Both of these behaviors can be modified by using the `store` and `output` keywords. To turn off internal storage (say, for a very long chain, for which storage would create a large memory footprint), use

```
store = false;
```

To output the chain to a file, use:

```
output file=/path/to/my/output/filename;
```

1.5.4 Reading Chains Back into Climax

You can instruct **climaxRunManager** to load chain files written by Climax (or even *Markov*) by using the command

```
load file=/path/to/my/output/filename;
```

This will load the file, in whatever units the parameters are specified, and create parameter histograms, as at the end of an internal Climax Markov chain run.

1.6 Examples

```
//============================================================
// An example of fitting a beta model to short-baseline SZA data
//============================================================

//------------------------------------------------------------
// Add the data set
//------------------------------------------------------------

adddataset name=duvf type=uvf;
duvf.file = ~eml/projects/climax/climaxTestSuite/A1914.uvf;
duvf.uvmin = 0;
duvf.uvmax = 2000;

//------------------------------------------------------------
// Add the beta model
//------------------------------------------------------------

addmodel name=m_cluster type=betamodel;

//------------------------------------------------------------
// Assign values to fixed model parameters
//------------------------------------------------------------

m_cluster.beta = 0.8;
m_cluster.axialRatio = 1;
m_cluster.rotang    = 0 degrees;
m_cluster.spectralType = sz;
m_cluster.normalizationFrequency = 30 GHz;

//------------------------------------------------------------
// Assign priors to parameters we want to fit
//------------------------------------------------------------

m_cluster.thetaCore = 5:295";
m_cluster.Sradio = -5:0 mK;
m_cluster.xoff   = -60:60";
m_cluster.yoff   = -60:60";

//------------------------------------------------------------
// Directives controlling the run itself
//------------------------------------------------------------

ntry  = 10000;
nburn = 3000;
nbin  = 50;

//------------------------------------------------------------
// Display to Pgplot window 1
//------------------------------------------------------------

dev = 1/xs;
```

2 The SZ Effect

The SZ-effect, or *SZE*, in clusters is a spectral distortion of the CMB radiation due to inverse Compton scattering of the relatively cool CMB photons off hot ICM electrons. At frequencies less than 218 GHz, the intensity of the CMB radiation is diminished compared to the unscattered CMB, and the SZ effect is manifested as a brightness temperature decrement towards the cluster. This decrement, $\Delta T_{SZ}/T_{CMB}$, has a magnitude proportional to the Compton y-parameter, i.e., the total number of scatterers (electrons) weighted by their associated temperature:

$$y = \int_{-\infty}^{\infty} n_e \frac{k_B T_e}{m_e c^2} \sigma_T \, d\ell, \tag{1}$$

where I treat the cluster as effectively infinitely far away and define $\ell = 0$ to be at the cluster center. This equation holds for any line of sight through the cluster. What we measure in SZ observations is the projected 2D image of the cluster, where the value at each location θ is given by this line integral, or

$$y(\theta) = \frac{\sigma_T}{m_e c^2} \int_{-\infty}^{\infty} P_e(r(\ell, \theta)) \, d\ell \tag{2}$$

if we identify $P_e = n_e k_B T_e$.

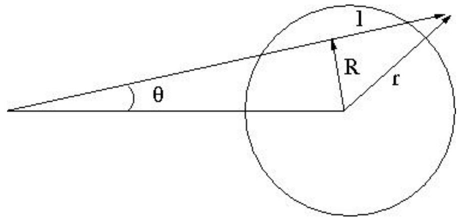

Figure 1: Schematic of the line integration through a cluster for constructing a projected 2D profile from a 3D radially-symmetric model.

For each θ on the sky, there is a cylindrical radius $R_\theta = D_A \theta$ that corresponds to a physical separation of that line of sight from the center of the cluster. With ℓ defined to be zero at the cluster center, for an axially-symmetric pressure model $P(r)$ we have

$$\begin{aligned} r^2 &= \ell^2 + R_\theta^2 & (3) \\ r\,dr &= \ell\,d\ell & (4) \\ d\ell &= \frac{r\,dr}{\sqrt{r^2 - R_\theta^2}} & (5) \end{aligned}$$

and

$$y(\theta) = \frac{\sigma_T}{m_e c^2} \int_{-\infty}^{\infty} \frac{P(r) r}{\sqrt{r^2 - R_\theta^2}} \, dr \tag{6}$$

In practice we treat $P(r)$ as a shape function $P(r) = P_0\, p(r)$ and compute the 2D projection of $p(r)$. Furthermore, we define the model in terms of $x \equiv r/r_c$, or $dr = r_c\, dx$ yielding:

$$\begin{aligned} y(\theta) &= \frac{\sigma_T}{m_e c^2} P_0 r_c \int_{-\infty}^{\infty} \frac{p(x)x}{\sqrt{x^2 - x_\theta^2}}\, dx \\ &= \frac{\sigma_T}{m_e c^2} P_0 D_A \theta_c \int_{-\infty}^{\infty} \frac{p(x)x}{\sqrt{x^2 - x_\theta^2}}\, dx \end{aligned} \tag{7}$$

where $x_\theta \equiv R_\theta/r_c$. Internally, CLIMAX actually computes the unity-normalized version of the integral, $N(\theta)$ i.e.,

$$\begin{aligned} I(\theta) &= \int_{-\infty}^{\infty} \frac{p(x)x}{\sqrt{x^2 - x_\theta^2}}\, dx \\ N(\theta) &= I(\theta)/I(0) \end{aligned} \tag{8}$$

and fits $y(\theta) = y(0)N(\theta)$, so that

$$\begin{aligned} y(0)N(\theta) &= \frac{\sigma_T}{m_e c^2} P_0 D_A \theta_c I(\theta) \\ &\equiv \frac{\sigma_T}{m_e c^2} P_0 D_A \theta_c I(0) N(\theta) \end{aligned} \tag{9}$$

This means that for CLIMAX fits to radially-symmetric pressure models, the pressure normalization can be recovered from:

$$P_0 = y(0) \left(\frac{m_e c^2}{\sigma_T D_A \theta_c} \right) \frac{1}{I(0)} \tag{10}$$

when models are normalized in Compton-y units. Note that I explicitly retain $I(0)$ in the normalization, since the models $P(r)$ are not in general defined to yield P_0 at $\theta = 0$. (The GNFW model for example (see 3.1.2) is actually singular at $r = 0$ and therefore not technically integrable at $\theta = 0$. And the Arnaud model is only fit to $r = 0.03\, R_{500}$, for which $I(\theta) \sim 0.89$).

Since the proportionality between Compton y and ΔT_{SZ} is given by:

$$\frac{\Delta T_{SZ}}{T_{CMB}} = f(\nu)\, y \tag{11}$$

where $f(\nu)$ is a function that encapsulates the frequency dependence of the SZ effect ($f(\nu) \sim -2$ at $\nu = 30$ GHz, see Eq. 21) the pressure normalization is given by:

$$P_0 = \frac{\Delta T_{SZ}(0)}{T_{CMB} f(\nu)} \left(\frac{m_e c^2}{\sigma_T D_A \theta_c} \right) \frac{1}{I(0)} \tag{12}$$

when models are normalized in temperature units in Climax.

2.1 Comptonization of the CMB

If we define the planck spectrum to be

$$I(x) = \frac{2(k_B T)^3}{(hc)^2} \frac{x^3}{e^x - 1} = i_0 \, i(x), \tag{13}$$

then change in intensity due to inverse-Compton scattering of a blackbody distribution of photons by an isotropic distribution of electrons is given in the optically thin limit ($\tau \ll 1$) by:

$$\Delta i(x) \equiv \frac{\Delta I(x)}{i_0} = \{j(x) - i(x)\}\, \tau \tag{14}$$

where

$$\tau = \sigma_T \int n_e \, d\ell, \tag{15}$$

is the optical depth, $i(x)\tau$ is the flux scattered to other frequencies and $j(x)\tau$ is the flux scattered from other frequencies to $x = h\nu/(kT)$.

It is useful to rewrite Eq. 14 as

$$\Delta i(x) = \tilde{g}(x)\, \tilde{y}, \tag{16}$$

where

$$\tilde{y} \quad = \quad \frac{\sigma_T}{m_e c^2} \int n_e k_B \tilde{T}_e \, d\ell$$

and $\tilde{T}_e$ is defined by

$$k_B \tilde{T}_e = \frac{P_e}{n_e} \tag{17}$$

(in the case of a thermal distribution of electrons, $\tilde{T}_e \equiv T_e$). Defining

$$\left\langle k_B \tilde{T}_e \right\rangle = \frac{\int n_e k_B \tilde{T}_e \, d\ell}{\int n_e \, d\ell} \tag{18}$$

we have

$$\tilde{g}(x) = \{j(x) - i(x)\} \frac{m_e c^2}{\left\langle k_B \tilde{T}_e \right\rangle} \tag{19}$$

For a thermal distribution of electrons, $\left\langle k_B \tilde{T}_e \right\rangle \equiv k_B T_e$, and in the non-relativistic regime, we have

$$\tilde{g}(x) = \frac{x^4 e^x}{(e^x - 1)^2} \left(x \frac{e^x + 1}{e^x - 1} - 4 \right) \frac{m_e c^2}{k_B T_e}, \tag{20}$$

i.e., the standard non-relativistic solution to the Kompaneets equation. To make contact with Eq. 11, we can identify the frequency dependence of the SZ temperature decrement in the non-relativistic case as the central term of Eq. 20:

$$f(x) = x \frac{e^x + 1}{e^x - 1} - 4. \tag{21}$$

In general, though, for an arbitrary electron momentum distribution $f_e(p)$, we have:

$$\left\langle k_B \tilde{T}_e \right\rangle = \int_\infty^0 f_e(p) \frac{1}{3} p v(p) m_e c \, dp, \tag{22}$$

where $p = \beta_e \gamma_e$ is the normalized electron momentum. The scattered spectrum is now given by

$$j(x) = \int_\infty^0 P(t) i(x/t) \, dt, \tag{23}$$

where $P(t)$ is the probability that a photon is scattered to a frequency $t = \nu'/\nu$ times its original frequency. The photon redistribution function can be written as

$$P(t) = \int_\infty^0 f_e(p) P(t;p) \, dp, \tag{24}$$

where $P(t;p)$ is the redistribution function for a mono-energetic electron distribution, which in the Thomson regime ($h\nu \ll \gamma_e m_e c^2$) has an analytic solution

$$\begin{aligned} P(t;p) = & - \frac{3|1-t|}{32p^6 t}[1 + (10 + 8p^2 + 4p^4)t + t^2] \\ & + \frac{3(1+t)}{8p^5} \left[\frac{3 + 3p^2 + p^4}{\sqrt{1+p^2}} - \frac{3 + 2p^2}{2p}(2\,\mathrm{arcsinh(p)} - |\ln t|) \right], \end{aligned}$$

where $P(t;p) = 0$ for $|\ln t| > 2\,\mathrm{arcsinh(p)}$.

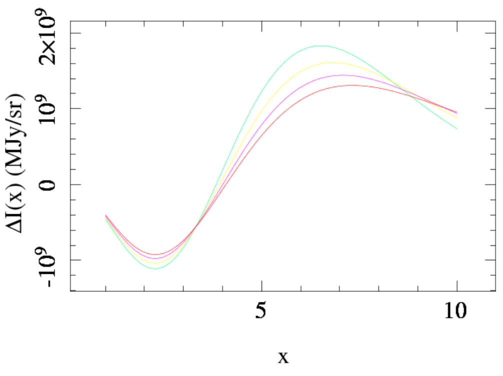

Figure 2: Comparison of the non-relativistic $\Delta I(x)$ computed with Eq. 20 (green) with the correct relativistic calculation using Eq. 19, for $T_e = 10, 20, 30$ keV.

2.2 Integrated Y Estimates

Given a 3D radial model, we can also compute the spherically integrated Compton-y parameter, $Y_{sph}(R)$

$$\begin{aligned} Y_{sph}(R) &= \frac{\sigma_T}{m_e c^2} \int_0^R P(r)\, dV \qquad (25) \\ &= \frac{\sigma_T}{m_e c^2} \int_0^R P(r) 4\pi r^2\, dr \\ &= \frac{\sigma_T P_0}{m_e c^2} \int_0^R p(r) 4\pi r^2\, dr \\ &= y(0) \frac{1}{D_A \theta_c} \frac{1}{I(0)} \int_0^R p(r) 4\pi r^2\, dr \qquad (26) \end{aligned}$$

if we substitute $P_e(0)$ from Eq. 10. If we make the transformation $x = r/r_c$, to transform to dimensionless coordinates, we have:

$$\begin{aligned} Y_{sph}(R) &= y(0) \frac{1}{D_A \theta_c I(0)} r_c^3 \int_0^{x_R} p(x) 4\pi x^2\, dx \qquad (27) \\ &= y(0) \frac{1}{D_A \theta_c I(0)} D_A^3 \theta_c^3 \int_0^{x_R} p(x) 4\pi x^2\, dx \\ &= y(0) D_A^2 \theta_c^2 \frac{1}{I(0)} \int_0^{x_R} p(x) 4\pi x^2\, dx \end{aligned}$$

for which we require the cosmology (i.e., D_A).

2.3 Mass estimates

Similarly, we see that

$$\begin{aligned}
Y_{sph}(R) &= \frac{\sigma_T}{m_e c^2}\int_0^R P_e(r)\,dV \qquad (28)\\
&= \frac{\sigma_T}{m_e c^2}\int_0^R P_e(r)4\pi r^2\,dr\\
&= \frac{\sigma_T P_e(0)}{m_e c^2}\int_0^R p(r)4\pi r^2\,dr\\
&= \frac{k_B\sigma_T T_e(0)}{m_e c^2}\frac{1}{m_p\mu_e}\int_0^R n_e(0)m_p\mu_e p(r)4\pi r^2\,dr\\
&= \frac{k_B\sigma_T T_e(0)}{m_e c^2}\frac{1}{m_p\mu_e}M_{gas}(R)
\end{aligned}$$

from which we have

$$M_{gas}(R) = m_p\mu_e\left(\frac{m_e c^2}{k_B T_e(0)}\right)\left(\frac{1}{\sigma_T}\right)Y_{sph}(R) \qquad (29)$$

or

$$M_{gas}(R) = m_p\mu_e\frac{y(0)}{I(0)}\left(\frac{m_e c^2}{k_B T_e(0)}\right)\left(\frac{D_A^2\theta_c^2}{\sigma_T}\right)\int_0^{x_R} p(x)4\pi x^2\,dx \qquad (30)$$

for which we require the cosmology (i.e., D_A) and the temperature normalization ($T_e(0)$). Here μ_e is the mean molecular weight per electron, defined as the ratio of the mean particle mass per electron to the mass of the hydrogen atom.

2.3.1 Some sanity checks

Let's take an example, to see what we get for a typical case. We can write the beta-model as a radially symmetric pressure model:

$$p(x) = \frac{1}{(1+x^2)^{3\beta/2}} \qquad (31)$$

In the particular case $\beta = 2/3$, the integrals become analytic. In particular, we have

$$\int p(x)\,dx = \int \frac{1}{(1+x^2)}\,dx = \arctan x \qquad (32)$$

and

$$\int p(x)x^2\,dx = \int \frac{x^2}{(1+x^2)}\,dx = x - \arctan x \tag{33}$$

whence

$$I(0) = \int_{-\infty}^{\infty} p(x)\,dx = \int_{-\infty}^{\infty} \frac{1}{(1+x^2)}\,dx = \pi \tag{34}$$

(which I confirmed numerically)

and

$$\int_0^{x_R} p(x)4\pi x^2\,dx = 4\pi \int_0^{x_R} \frac{x^2}{(1+x^2)}\,dx = 4\pi(x_R - \arctan x_R) \tag{35}$$

(which I also confirm numerically, for $x_R = 1$).

From Equation 30 then, we have:

$$\begin{aligned} M_{gas}(R) &= m_p\mu_e \frac{y(0)}{I(0)}\left(\frac{m_e c^2}{k_B T_e(0)}\right)\left(\frac{D_A^2\theta_c^2}{\sigma_T}\right)\int_0^{x_R} p(x)4\pi x^2\,dx \\ &= y(0)\left(\frac{T_e}{1\text{ keV}}\right)^{-1}\left(\frac{D_A}{1\text{ Gpc}}\right)^2\left(\frac{\theta_c}{1\text{ arcsec}}\right)^2(x_R - \arctan x_R) \\ &\quad \times 6.8\times 10^{14} M_\odot \end{aligned} \tag{36}$$

A β-model fit to A1914 with $\beta = 2/3$ gives $\theta_c = 20$ arcsec and $\Delta T(0) = -2$ mK, or $y(0) \sim 3.7\times10^{-4}$.

For A1914 we also have $T_e \sim 10$ keV, $D_A \sim 0.6$ Gpc, yielding

$$M_{gas}(R) = y(0)(x_R - \arctan x_R) \times 9.8\times10^{15} M_\odot \tag{37}$$

In Tony's paper, he estimates an R_{500} of $R \sim 1.3$ Mpc, or $x_R = R/(D_A\theta_c) \sim 22$, yielding

$$M_{gas}(R) = 7.7\times10^{13} M_\odot. \tag{38}$$

Note that this is not what Tony reports for A1914, but is within 30% of his value ($11\times10^{13}M_\odot$).

2.3.2 Self-Similar Models — Theory

The pressure normalization of SZ models can also be related directly to the mass of the cluster, through considerations of self-similarity.

We start from the characteristic temperature for an isothermal sphere of mass M_{500}

$$kT_{500} = \mu m_p \frac{GM_{500}}{2R_{500}} \tag{39}$$

where M_{500} is given by

$$M_{500} = 500\rho_c(z)\frac{4\pi}{3}{R_{500}}^3 \tag{40}$$

and

$$\rho_c(z) = \frac{3H(z)^2}{8\pi G}. \tag{41}$$

The characteristic gas density is

$$\rho_{g,500} = 500 f_B \rho_c(z) \tag{42}$$

and the electron density is

$$n_{e,500} = \frac{\rho_{g,500}}{\mu_e m_p} \tag{43}$$

Putting it all together, we have

$$\begin{aligned} P_{500} = n_{e,500} \times kT_{500} &= \frac{\rho_{g,500}}{\mu_e m_p} \times \mu m_p \frac{GM_{500}}{2R_{500}} \\ &= \frac{\mu}{\mu_e}\rho_{g,500} \times \frac{GM_{500}}{2R_{500}} \\ &= \frac{\mu}{\mu_e} 500 f_B \rho_c(z) \times \frac{GM_{500}}{2R_{500}} \end{aligned} \tag{44}$$

From Eq.40 we have

$$R_{500} = \left(\frac{3}{4\pi}\frac{M_{500}}{500\rho_c(z)}\right)^{1/3} \tag{45}$$

whence

$$\begin{aligned} P_{500} &= \frac{\mu}{\mu_e} 500 f_B \rho_c(z) \times \frac{GM_{500}}{2R_{500}} \\ &= \frac{\mu}{\mu_e} 500 f_B \rho_c(z) \times \frac{GM_{500}}{2}\left(\frac{4\pi}{3}\frac{500\rho_c(z)}{M_{500}}\right)^{1/3} \end{aligned}$$

$$\begin{aligned}
&= \frac{\mu}{\mu_e} f_B G \frac{3}{8\pi} \left(\frac{4\pi}{3} 500 \rho_c(z)\right)^{4/3} M_{500}^{2/3} \\
&= \frac{\mu}{\mu_e} f_B G \frac{3}{8\pi} \left(\frac{4\pi}{3} 500 \frac{3H(z)^2}{8\pi G}\right)^{4/3} M_{500}^{2/3} \\
&= \frac{\mu}{\mu_e} f_B \frac{3}{8\pi} \left(\frac{500 G^{-1/4} H(z)^2}{2}\right)^{4/3} M_{500}^{2/3} \qquad (46)
\end{aligned}$$

3 Models

3.1 SZ Models

3.1.1 Beta Model (β-model)

The β-model is an analytic shape function widely used in X-ray analysis to fit the radial density profile:

$$n_e(x) = \frac{n_{e0}}{[1 + x^2]^{(3\beta/2)}} \tag{47}$$

The line-integral of this function is therefore commonly used to fit the projected 2D cluster profiles in SZ observations:

$$p(x) = \frac{1}{[1 + x^2]^{(1-3\beta/2)}} \tag{48}$$

The β-model can bs used in Climax by invoking `addmodel` with `type = betamodel`.

3.1.2 Generalized NFW (GNFW) Model

In the context of the pressure models of Nagai et al. (2007), hereafter N07, the generalized NFW profile is given by

$$p(x_g) = \frac{p_0}{(c_{500}x_g)^\gamma \left[1 + (c_{500}x_g)^\alpha\right]^{(\beta-\gamma)/\alpha}} \tag{49}$$

where c_{500} is a dimensionless concentration parameter and $x_g = r/R_{500}$.

In Climax, the generalized NFW model is implemented as a dimensionless shape function that can be used to fit cluster profiles in any units:

$$p(x) = \frac{1}{x^\gamma \left[1 + x^\alpha\right]^{(\beta-\gamma)/\alpha}} \tag{50}$$

where $x = r/r_c$. In the context of the GNFW pressure models, therefore, the r_c returned by Climax (actually θ_c), is equivalent to $r_c = R_{500}/c_{500}$.

The generalized NFW model can be used in Climax by invoking `addmodel` with `type = gnfwmodel`.

From self-similarity arguments (see §3.2), the pressure normalization (see Eq 10) of a cluster can be related to its mass via Eq 53. In Climax `m500` can therefore also be used with any GNFW model as the primary variable, given a cosmology.

3.1.3 Nagai07 GNFW Model

N07 find that a good description of high-T_X *Chandra* clusters can be fit with a model with $p_0 = 3.3$, $c_{500} = 1.8$ and $(\alpha, \beta, \gamma) = (1.3, 4.3, 0.7)$.

This specialization of the GNFW model can be used in CLIMAX by invoking `addmodel` with `type = nagai07model`.

3.1.4 Arnaud GNFW Model

Arnaud et al. (2010), hereafter A10, determine that $p_0 = 8.403\, h_{70}^{-3/2}$, $c_{500} = 1.17$ and $(\alpha, \beta, \gamma) = (1.0510, 5.4905, 0.3081)$ yield the best fit to REXCESS clusters.

This specialization of the GNFW model can be used in CLIMAX by invoking `addmodel` with `type = arnaudmodel`.

A10 also determine the normalization of the pressure model fits to be

$$\begin{aligned} P(r) &= 1.65 \times 10^{-3} h(z)^{8/3} \left[\frac{M_{500}}{3 \times 10^{14} h_{70}^{-1} M_\odot}\right]^{2/3+\alpha_P+\alpha'_P(x)} \\ &\times\ p_{\rm A}(x_g)\, h_{70}^2\, \mathrm{keV\,cm^{-3}}, \end{aligned} \tag{51}$$

where $p_{\rm A}(x_g)$ is the GNFW profile with the Arnaud et al fit parameters given above, $\alpha_P = 0.12$, and

$$\alpha'_P(x_g) = 0.1 - (\alpha_P + 0.1)\frac{(x_g/0.5)^3}{1 + (x_g/0.5)^3} \tag{52}$$

The small second-order x_g-dependent term represents a departure from self-similarity, and also a significant increase in computational overhead, and it is presently ignored in CLIMAX.

3.2 Problems with the GNFW model

I have explored the GNFW model pretty extensively at this point, and have encountered a number of issues with it:

1 The A10 and N07 normalizations don't agree, even for the same set of fit parameters

2 For a given M500, the A10 normalization predicts a pressure (and therefore an SZ decrement) that is significantly lower than what we actually measure for real clusters thought to be close to that M500

  Another way of saying the same thing is that for a fixed SZ decrement, the GNFW model normalization predicts an M500 that is significantly larger than what you get from joint fits to SZ + Xray

3 The A10 model is not self-consistent. If you start with an M500 and use the A10 normalization to construct the corresponding pressure profile, the M500 that you get by integrating that pressure profile out to R500 does not agree with the M500 you started with

In the next couple sections, I address each of these in turn.

3.2.1 Comparing the A10 and N07 P500

Substituting $f_B = 0.175, \mu = 0.59, \mu_e = 1.14, G = 6.67384 \times 10^{-8}\mathrm{cm}^3/(\mathrm{g\,s}^2)$, $h(z) \equiv H(z)/H_0$, we have

$$\begin{aligned}
P_{500} &= \frac{\mu}{\mu_e} f_B \frac{3}{8\pi} \left(\frac{500 G^{-1/4} H(z)^2}{2} \right)^{4/3} M_{500}^{2/3} \\
&= (4197.54)\, H(z)^{8/3} M_{500}^{2/3} \\
&= (4197.54)\, h(z)^{8/3} H_0^{8/3} M_{500}^{2/3} \\
&= (4197.54)\, (70 \text{ km/s/Mpc})^{8/3}\, (3 \times 10^{14} M_\odot)^{2/3}\, h(z)^{8/3} h_{70}^{8/3} \left[\frac{M_{500}}{3 \times 10^{14} M_\odot} \right]^{2/3} \\
&= 1.65 \times 10^{-3}\, h(z)^{8/3} h_{70}^{8/3} \left[\frac{M_{500}}{3 \times 10^{14} M_\odot} \right]^{2/3} \mathrm{keV\,cm}^{-3} \qquad (53)
\end{aligned}$$

This is to be compared to Eq 13 of A10, with which it agrees exactly. I therefore conclude that there are no arithmetic errors in the derivation of Arnaud's P500/M500 relationship.

Note that if we instead normalize M_{500} to $1 \times 10^{15} M_\odot$, and take $h_{70} = 1$ or $h = 0.7$, as Nagai et al do, we have:

$$P_{500} = 5.89 \times 10^{-12}\, h(z)^{8/3} \left[\frac{M_{500}}{1 \times 10^{15} M_\odot} \right]^{2/3} \mathrm{erg\,cm}^{-3} \qquad (54)$$

This is to be compared to Eq 3 of Nagai et al, with $h = 0.7$, which yields

$$P_{500} = 1.14 \times 10^{-11}\, h(z)^{8/3} \left[\frac{M_{500}}{1 \times 10^{15} M_\odot} \right]^{2/3} \mathrm{erg\,cm}^{-3} \qquad (55)$$

(Note that $h(z)$ is equivalent to the $E(z)$ of Nagai et al. And N07 actually take $h = 0.72$, but I've used $h = 0.7$ for simplicity)

The ratio of these two expressions is very close to 2, and the reason is that N07 have dropped a factor of h^2 in their Eq 3, which as-written is proportional to $h^{2/3}$ and not $h^{8/3}$, which is clearly required by Eq 46. So that mystery amounts to nothing more than typo in N07.

3.2.2 Comparing P500 and M500

To make contact with P_0 of Equation 10, we have

$$P_0 = p_0 \times 1.65 \times 10^{-3} h(z)^{8/3} \left[\frac{M_{500}}{3 \times 10^{14} h_{70}^{-1} M_\odot} \right]^{2/3+\alpha_P} h_{70}^2 \, \mathrm{keV\, cm^{-3}} \tag{56}$$

(if we ignore α'_P). This suggests, on the face of it, that given a cosmology, we can relate an observed central decrement $y(0)$ to the Arnaud pressure normalization and therefore M_{500} by plugging P_0 into Equation 10. However, I can't quite make sense of the numbers I get if I attempt to do this.

Let's take Abell 1914 as a test case. This cluster has a redshift $z = 0.168$, which yields $h(z) = 1.089$ and $D_A = 0.59$ Gpc. For A1914, the central decrement is roughly -2 mK, or $y(0) \sim 4 \times 10^{-4}$. The Arnaud fits give a typical $\theta_c \sim 2'$, yielding:

$$P_0 = 0.29 \text{ keV/cm}^3 \tag{57}$$

from Equation 10. Plugging the numbers into Equation 56, we have:

$$P_0 = 1.74 \times 10^{-2} \left[\frac{M_{500}}{3 \times 10^{14} M_\odot} \right]^{2/3+\alpha_P} \text{keV/cm}^3, \tag{58}$$

or

$$M_{500} = 1 \times 10^{16} M_\odot \tag{59}$$

which seems about an order of magnitude too large (estimates of the virial mass for A1914 I've seen are somewhere in the neighborhood of $2 - 3 \times 10^{15} M_\odot$).

3.2.3 The A10 Model is not Self-consistent

As noted above, if you start with an M500, you can use Eq 53 to determine P_{500} and thus the normalization of the cluster pressure profile, via Eq 10, in particular:

$$P(r) = P_{500}\, p(r). \tag{60}$$

where $p(r)$ is the GNFW profile in Eq 49.

This suggests that if the cluster were an isothermal sphere, then $P(R_{500})/P_{500} = p(R_{500}) = 1.0$. For A10's parameter values, however, $p(R_{500}) = 0.48$ (which is suspiciously close to the ratio of $(70/100)^2$). It is also consistent with my empirical observation that the P_{500} normalization predicts an SZ decrement that is about a factor of 2 smaller than observed, for a set of clusters for which we have independent SZ + X-ray mass estimates.

From Eq 26 and Eq 30 we also have:

$$M_{500} = \frac{1}{f_{gas}} \left(\frac{m_p \mu_e}{k_B T_e} \right) \int_0^{R_{500}} P(r)\, dV \tag{61}$$

$$= \frac{1}{f_{gas}} \left(\frac{m_p \mu_e}{k_B T_e} \right) P_{500}\, p_0 \int_0^{R_{500}} p(r)\, dV \qquad (62)$$

If I take measured X-ray temperatures for a set of clusters and the A10 pressure profile fits, I can integrate this expression and compare it to what the A10 normalization would predict for M_{500}, and again I find that the masses I obtain in this manner are factors of several lower than the best-fit M_{500}.

All of which suggests that an appropriate way to proceed with my simulations is to rescale p_0 to force self-consistency with the input M_{500}. That is, I find the value of p_0 that gives me the same integrated M_{500} that I put in the model in the first place. This factor is shown in Figure 3.

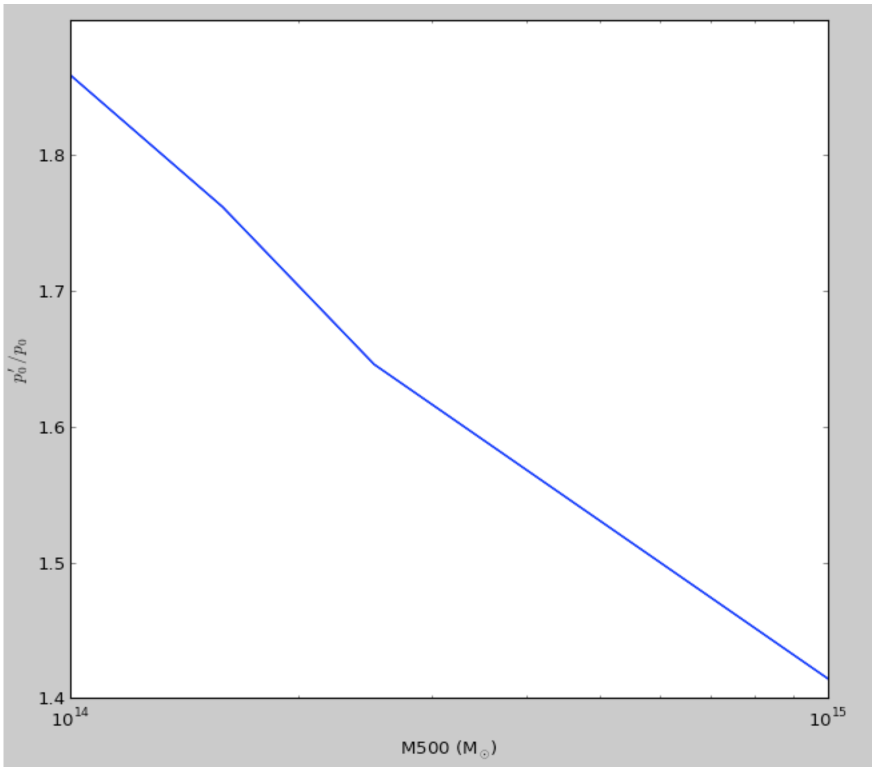

Figure 3: Plot of factor by which the A10 p_0 needs to be rescaled to recover the same integrated M_{500} that is put into the model normalization

3.3 Other Models

3.3.1 Image Model

Generic images can also be used in Climax as models, via `adddataset` with `type = image`. Supported image types, and manipulation are as described in §4.5.1.

Image models support only the components common to all 2D models: translation (offsets), rotation, and scaling (normalization). When reading in images, you must specify either a background level, or provide a region of the image from which it can be estimated. This allows for real data images to be used as models, for which a background (i.e., constant offset) would significantly skew any goodness-of-fit estimator.

On read-in, this background will be subtracted from the image, and the resulting image normalized to unity at the peak. When comparing image models to radio or xray data, the usual requirements for specifying the appropriate normalization apply (i.e., `Sradio` or `Sxray` in appropriate units). Note that no assumptions are made about the source of the image data; for example, if the source is an xray surface brightness image, for example, you will probably want to apply an appropriate transformation (i.e., `trans = sqrt`) to the image before using it as an SZ model.

Currently, two operations are supported for evaluating an image model at a requested offset, by using the `oper` keyword: returning the value of the nearest pixel (`oper = nearest`), or interpolating at the requested point (`oper = interpolate`). For offsets that lie outside of a model image, a zero value is returned, and the pixel is marked as invalid.

Invalid pixels will not be used in evaluating the likelihood for image-plane models, however note that Fourier-plane datasets cannot exclude individual pixels, and the zero values will be incorporated into the transform and hence the likelihood evaluation. You should think carefully about what makes sense for your use-case. Supplying a valid image that is large enough to accomodate any offset explored by the Markov chain can eliminate this potential problem.

An example of using an X-ray-derived image model to fit SZ data is included below:

```
adddataset name=sz type=uvf;
sz.file=A1914.uvf;
sz.uvmax = 2000;
sz.display = true;

addmodel name=im type=image;
im.file=acisf00542N004_cntr_img2.fits;
im.npix = 64;
im.thetaMinErr = 0.04 deg;
im.display = true;
im.interactive = true;
im.trans = sqrt;
im.sigmasmooth = 20";

im.Sradio = -2000:0 muK;
im.xoff = -0.01:0.01 deg;
im.yoff = -0.01:0.01 deg;
im.rotang = -90:90 deg;

nburn = 3000;
ntry  = 10000;
```

4 Datasets

4.1 Visibility Datasets

Visibility datasets in UVF format are currently supported (I have a stub for miriad file format and the code to read it exists, but it is not yet implemented), by using `adddataset` with `type = uvf`, i.e.:

```
adddataset type=uvf name=duvf;
```

On read-in, visibility data are automatically gridded in Fourier space by unique baseline type (for heterogeneous array data) and frequency. By default the size of the Fourier grid (effective resolution in image space) is chosen for each baseline/frequency combination according to the maximum uv radius present in the data. The resolution in Fourier space (effective size in image space) is chosen so that the integrated overlap of the autocorrelation functions (what I refer to as the *correlation percentage*) for visibility data in each cell is at least `perc`. This defaults to 95%, or `duvf.perc` = `0.95`. The automated gridding will, in general, result in different model and primary beam resolution for each baseline/frequency pair, and Fourier transforms that are not square, leading to the fastest performance for image-plane models that require Fourier inversion.

Although dirty images are produced for visibility datasets as if the array were homogeneous (i.e., no correction is made for the different primary beams), the fitting is handled correctly for heterogeneous data as well. If you want to make beam-corrected images (as for a mosaic) of a heterogeneous dataset, use the mosaicked dataset type instead (see §4.3 below).

For heterogeneous data, antennas can be included/excluded by type (if the type can be determined from the data), or by antenna number. Thus

```
duvf.useanttypes = sza,bima;
```

would cause Climax to use only data from SZA and BIMA antennas in a CARMA23 dataset, and

```
duvf.excludeants = 1,4;
```

would use only data matching any specified type, and not including antennas 1 or 4.

4.1.1 Making Images of Visibility Data

Data can also be gridded to a fixed resolution if desired, though this is not recommended for running chains, since there is no guarantee that the correlation percentage is uniform, or that the data are gridded to the minimum required transform size. It can however be useful for producing images of visibility datasets at fixed resolution. For example, to produce a 128×128-pixel image that is 0.3 degrees on the sky, you can do:

```
adddataset type=uvf name=duvf;
duvf.size = 0.3 degrees;
duvf.npix = 128;
duvf.display = true;
```

To select a uv range for imaging or fitting in a Markov chain, use the `uvmin` and `uvmax` parameters:

```
duvf.uvmin = 0;
duvf.uvmax = 2000;
```

This would apply a cutoff at 2 $k\lambda$ when gridding the visibility data. You can also apply a uvtaper to the data for imaging, but it is not recommended for fitting, since it changes the likelihood ratio associated with a given $\Delta\chi^2$. For example:

```
duvf.uvtaper = 0.5, 2000;
```

would apply a gaussian uvtaper whose half-power point lies at 2 $k\lambda$.

For all visibility datasets you can have CLIMAX output the data/model/residual images as FITS files by using the `dataimage/modelimage/resimage` keywords.

4.1.2 Dirty Maps and Pixel Covariance

When we form a dirty map by Fourier transform of the visibilities, for each pixel j we construct the sum:

$$\begin{aligned} d_j &= \int\int \widetilde{V}(u,v)e^{2\pi i(ux_j+vy_j)}du\,dv \\ &= \int_{HP}\left\{\widetilde{V}(u,v)e^{2\pi i(ux_j+vy_j)} + \widetilde{V}(-u,-v)e^{-2\pi i(ux_j+vy_j)}\right\} \\ &= \int_{HP}\left\{\widetilde{V}(u,v)e^{2\pi i(ux_j+vy_j)} + \widetilde{V}^*(u,v)e^{-2\pi i(ux_j+vy_j)}\right\} \end{aligned} \tag{63}$$

where the integral is over the half-plane, and the last step assumes that the Fourier transform is Hermitian. If we rewrite the transform as a discrete sum, we have:

$$\begin{aligned} d_j &= \sum_k \left\{\left(V_k^R + iV_k^I\right)\{\cos(a_{jk}) + i\sin(a_{jk})\} + \left(V_k^R - iV_k^I\right)\{\cos(a_{jk}) - i\sin(a_{jk})\}\right\} \\ &= \sum_k \left\{V_k^R\cos(a_{jk}) - V_k^I\sin(a_{jk})\right\} \end{aligned} \tag{64}$$

up to a normalization (with $a_{jk} = 2\pi(x_ju_k + y_jv_k)$). For maps made with "natural weighting", we form the dirty map from the weighted mean of the visibilities

$$d_j = \frac{\sum_k w_kV_k^R\cos(a_{jk}) - \sum_k w_kV_k^I\sin(a_{jk})}{\sum_k w_k}, \tag{65}$$

with $w_k = 1/\sigma_k^2$, i.e., the inverse variance of the Re/Im part of the visibility (since that is the estimator that maximizes the likelihood of our visibility data). What is the variance of $\{d_j\}$? It's given by

$$\begin{aligned}
\sigma^2 \equiv < d_j^2 > &= \frac{\sum_k w_k^2 \left\langle {V_k^R}^2 \right\rangle \cos^2(a_{jk}) + \sum_k w_k^2 \left\langle {V_k^I}^2 \right\rangle \sin^2(a_{jk})}{\left(\sum_k w_k\right)^2} \\
&= \frac{\sum_k w_k^2 \sigma_k^2 \cos^2(a_{jk}) + \sum_k w_k^2 \sigma_k^2 \sin^2(a_{jk})}{\left(\sum_k w_k\right)^2} \\
&= \frac{\sum_k w_k^2 \sigma_k^2}{\left(\sum_k w_k\right)^2} = \frac{\sum_k w_k}{\left(\sum_k w_k\right)^2} = \frac{1}{\sum_k w_k}
\end{aligned} \tag{66}$$

4.1.3 Synthesized beam

From examination of Eqs. 63 and Equation 65, it is clear that we can express the discrete sampling of the Fourier transform as the multiplication of $\widetilde{V}$ with a sampling function $\widetilde{W}$, where

$$\widetilde{W} = \frac{\sum_k \delta(u_k, v_k) w_k}{\sum_k w_k}. \tag{67}$$

By the convolution theorem, the inverse transform $W \equiv \mathcal{F}^{-1}\{\widetilde{W}\}$ therefore represents the function by which the sky signal has been convolved, or effectively the point-spread function of the interferometer. This function is commonly referred to as the *synthesized beam*, and can have significant spatial sidelobe structure compared to a typical filled-aperture instrument, due to its sharp discontinuities in Fourier space.

4.1.4 Estimating the clean beam width

We want a gaussian approximation to the synthesized beam.

A gaussian approximation to any function can be calculated from considering the Taylor expansion of its logarithm:

$$\ln f(x) \simeq \ln f(x_0) - \left.\frac{\partial \ln f}{\partial x}\right|_{x_0} (x - x_0) + \frac{1}{2} \left.\frac{\partial^2 \ln f}{\partial x^2}\right|_{x_0} (x - x_0)^2 + \ldots \tag{68}$$

Since we are expanding about the maximum, $\left.\frac{\partial \ln f}{\partial x}\right|_{x_0} = 0$, and

$$\ln f(x) \simeq \ln f(x_0) + \frac{1}{2} \left.\frac{\partial^2 \ln f}{\partial x^2}\right|_{x_0} (x - x_0)^2 + \ldots \tag{69}$$

To the extent that we can ignore higher-order terms in the expansion, the function will be dominated by the quadratic term, and we can write

$$f(x) \simeq f(x_0) \exp\left(\frac{1}{2} \left. \frac{\partial^2 \ln f}{\partial x^2} \right|_{x_0} (x - x_0)^2 \right) \tag{70}$$

which is just a Gaussian of width

$$\sigma_x = \left(- \left. \frac{\partial^2 \ln f}{\partial x^2} \right|_{x_0} \right)^{-1/2} . \tag{71}$$

To calculate the gaussian approximation to the synthesized beam, we start with the (1D) expression for the synthesized beam:

$$W(x) = \frac{\sum_k w_k \cos(2\pi x u_k)}{\sum_k w_k} \tag{72}$$

whence

$$\ln W(x) = \ln \sum_k w_k \cos(2\pi x u_k) + \ln \sum_k w_k, \tag{73}$$

$$\frac{\partial \ln W(x)}{\partial x} = - \frac{2\pi \sum_k w_k u_k \sin(2\pi x u_k)}{\sum_k w_k \cos(2\pi x u_k)} \tag{74}$$

and

$$\frac{\partial^2 \ln W(x)}{\partial x^2} = - \frac{(2\pi)^2 \sum_k w_k u_k^2 \cos(2\pi x u_k)}{\sum_k w_k \cos(2\pi x u_k)} + \frac{(2\pi)^2 \left(\sum_k w_k u_k \sin(2\pi x u_k)\right)^2}{\left(\sum_k w_k \cos(2\pi x u_k)\right)^2}. \tag{75}$$

When we evaluate this at zero, the sin terms vanish, and we are left with:

$$\sigma = \frac{1}{\sqrt{2\pi^2 \langle u^2 \rangle}} \tag{76}$$

with

$$\left\langle u^2 \right\rangle \equiv \frac{\sum u_k^2 w_k}{\sum w_k}. \tag{77}$$

4.1.5 Clean images

Rewriting $\Delta V_k \equiv V_k^R cos(a_{jk}) - V_k^R sin(a_{jk})$, we see that if we divide the visibilities into subsets, we have:

$$\begin{aligned} d_j &= \frac{\sum_{k1} w_{k1}\Delta V_{k1} + \sum_{k2} w_{k2}\Delta V_{k2}}{\sum_{k1} w_{k1} + \sum_{k2} w_{k2}} && (78) \\ &= \frac{\sum_{k1} w_{k1}}{\sum_k w_k}\frac{\sum_{k1} w_{k1}\Delta V_{k1}}{\sum_{k1} w_{k1}} + \frac{\sum_{k2} w_{k2}}{\sum_k w_k}\frac{\sum_{k2} w_{k2}\Delta V_{k2}}{\sum_{k2} w_{k2}} && (79) \\ &= \frac{\sum_i w_i d_{ij}}{\sum_i w_i}, && (80) \end{aligned}$$

that is, the combined dirty map is the weighted sum of the individual dirty maps, with w_i given by the sum of the weights for each subset of visibilities.

What is generally referred to as a *clean* image is the equivalent sky signal that would be seen by a filled aperture instrument with resolution equivalent to the main lobe of the synthesized beam. We can use Eq. 80 to construct this image in the same way as the dirty image, but with d_{ij} replaced by the model convolved with a Gaussian approximation to the synthesized beam, i.e.:

$$c_j = \frac{\sum_i w_i c_{ij}}{\sum_i w_i} + r_j \tag{81}$$

where $c_{ij} = \{M * \hat{W}_i\}_j$, i.e., the model convolved with the i^{th} approximate PSF, and r_j is the residual image, obtained from Eq. 65 by replacing $\tilde{V}_k$ with the delta between the data and model, i.e., $\tilde{V}^m_k - \tilde{V}^d_k$.

Two different types of clean images can be made in Climax: an image where the model to be convolved with the synthesized beam is taken from the parameter file (or best-fit model from a Markov chain), or an image where the model is built up iteratively out of delta functions via the Högbom CLEAN algorithm (Högbom, 1974).

The type of clean image to produce is controlled by the `clean` and `cleantype` keywords. If `clean = true` then a clean image will be generated in place of the default model image. To produce the first type of clean image, use:

```
adddataset type=uvf name=vis;
vis.clean = true;
vis.cleantype = model;
```

Note that this will construct an image using an equivalent filled-aperture synthesized beam. As-such, if your model contains large-scale power that is not sampled by the actual (i.e., incomplete Fourier sampling) instrument, then your clean image will also contain large-scale power that is not present in the dirty maps.

An alternative is the Högbom algorithm, which iteratively constructs a model from the dirty maps using only delta-function components; i.e., it is a filled-aperture model approximation to the structure seen in the dirty maps using only components that the actual instrument has sensitivity to. This can be requested by specifying `cleantype = delta` instead.

The Högbom algorithm works by iteratively finding the highest peaks in the dirty map, at each step in the iteration subtracting a copy of the synthesized beam, whose amplitude is a fraction of the flux at that location. The iteration proceeds until a specific cutoff flux is reached, or the maximum number of iterations is reached if no cutoff is specified.

The number of iterations is controlled by the `cleaniter` keyword. The fraction of the synthesized beam to subtract at each step is specified by the `cleangain` keyword. A cutoff threshold (in Jy) can be specified using the `cleancutoff` keyword.

Additionally, the user must specify windows that the clean algorithm will search. You can specify clean windows in a couple different ways:

```
cleanwindow = [abs +- 0.01, abs +- 0.01, deg];
```

sets up a clean window of ± 0.1 deg around the position of the absolute maximum. You can also use `max` or `min` instead of `abs`. Similarly

```
cleanwindow = [-2:2, -2:2, '];
```

sets up an arbitrary rectangle window. Lastly, you can use:

```
cleanwindow += [-2:2, -2:2, '];
```

to set up more than one window.

4.2 Writing Visibility Datasets

Data, model or residual visibilities can be written out using the `datauvf`, `modeluvf` or `resuvf` keywords, in combination with the `writedata` directive. Note that the dataset's `store` keyword must be set to true to use this function.

For unstacked visibility data, all data (including flagged or unselected visibilities) are written, with weights as in the original file.

For stacked data, only unflagged data matching your current selection criteria (selected IFs, antennas or UV range) will be written out, with any requested weight scaling applied to the data.

4.3 Stacking Visibility Datasets

For each dataset, we construct the weighted mean of each visibility according to:

$$\hat{V}_k = \frac{\sum V_j w_{jk}}{\sum w_{jk}} \tag{82}$$

When combining datasets for which estimators $\hat{V}_k$ have already been formed, we ultimately want:

$$\begin{aligned}\hat{V} &= \frac{\sum_k \sum_j V_j w_{jk}}{\sum_k \sum w_{jk}} \\ &= \frac{\sum_k \hat{V}_k \sum w_{jk}}{\sum_k \sum w_{jk}} \\ &= \frac{\sum_k \hat{V}_k W_k}{\sum_k W_k}\end{aligned}$$
.

In other words, the combined maximum likelihood estimator for V is just the weighted mean of the separate estimators $\hat{V}_k$, with weights $W_k = \sum w_{jk}$.

From Eq. 80, we see that if we break up the visibilities into subsets, the combined dirty map can be written as the weighted sum of the dirty maps from each subset. If the subsets of visibilities represent different observations with the same pointing, the result is trivially the dirty map you would obtain by combining the visibilities and Fourier-transforming.

What if the subsets of visibilities represent offset pointings? By shifting each subset of visibilities to a common center, we can construct a dirty map that has the correct center, but the primary beam envelope for each subset will now be shifted with respect to the others, i.e.

$$d_j = \frac{\sum_i w_i d_{ij}}{\sum_i w_i}, \tag{83}$$

but

$$d_{ij} = b_{ij} s_j + n_{ij}, \tag{84}$$

where

$$b_{ij} = b(x_j + \Delta x_i, y_j + \Delta y_i). \tag{85}$$

Thus the dirty image is the sky signal multiplied by an effective primary beam $\bar{b}_j$, where

$$\bar{b}_j = \frac{\sum_i w_i \, b(x_j + \Delta x_i, y_j + \Delta y_i)}{\sum_i w_i}. \tag{86}$$

4.4 Mosaicked Visibility Datasets

Mosaicked visibility datasets can also be handled in Climax, by using the by using `adddataset` with `type = mos`, i.e.:

```
adddataset type=mos name=dmos;
```

Once defined, datasets can be added to the mosaic using the `file` keyword:

```
dmos.file  = MS0735_p1.uvf;
dmos.file += MS0735_p2.uvf;
```

The mosaic dataset supports the same set of keywords as the `uvf` visibility dataset (e.g., `wtscale, uvmin, uvmax`) plus some additional ones, primarily for mapmaking.

Use the keyword `power` to specify the point of the primary beam beyond which images for each antenna pair/frequency combination will be masked before coadding to make the final mosaicked map.

Use the keyword `wtmin` to set the minimum weight in the combined map. The final image will be truncated at the boundary beyond which the weight is below this limit.

These keywords have no effect on fitting of mosaicked data in Climax.

Note that when using the mosaicked dataset, you will in general need to specify absolute coordinates (RA/DEC) for any models, so that the relative separations from the RA/DEC of component datasets are correctly accounted for.

4.4.1 Mosaicking Theory

The "dirty image" from an interferometer is enveloped by the primary beam of the antennas. As such, if d_{ij} represents the intensity in pixel j of map i, we have

$$d_{ij} = b_{ij} s_j + n_{ij}, \tag{87}$$

that is, the pixel consists of the sky signal s_j, multiplied by the ith beam at location j, b_{ij}, plus some additive noise. We wish to form the weighted mean of the true sky signal at each pixel, or:

$$\hat{s}_{ij} = d_{ij}/b_{ij}, \tag{88}$$

with variance

$$\sigma^2_{\hat{s}_{ij}} = \sigma^2_i / b^2_{ij}, \tag{89}$$

where σ^2_i is given by Eq. 66. Thus the jth pixel in the mosaicked map is given by:

$$\begin{aligned} \hat{s}_j &= \frac{\sum_i \hat{s}_{ij}/\sigma^2_{\hat{s}_{ij}}}{\sum_i 1/\sigma^2_{\hat{s}_{ij}}} && (90) \\ &= \frac{\sum_i b_{ij} d_{ij}/\sigma^2_i}{\sum_i b^2_{ij}/\sigma^2_i}. && (91) \end{aligned}$$

The uncertainty on the weighted mean sky signal in each pixel is given by the variance of Eq. 91:

$$V\left[\frac{\sum_i b_{ij}d_{ij}/\sigma_i^2}{\sum_i b_{ij}^2/\sigma_i^2}\right] = \frac{\sum_i V\left[b_{ij}d_{ij}/\sigma_i^2\right]}{\left(\sum_i b_{ij}^2/\sigma_i^2\right)^2} = \frac{\sum_i V[b_{ij}d_{ij}]/\sigma_i^4}{\left(\sum_i b_{ij}^2/\sigma_i^2\right)^2} = \frac{\sum_i b_{ij}^2/\sigma_i^2}{\left(\sum_i b_{ij}^2/\sigma_i^2\right)^2} = \frac{1}{\sum_i b_{ij}^2/\sigma_i^2}, \tag{92}$$

or

$$\sigma_{\hat{s}_j}^2 = \frac{1}{\sum_i b_{ij}^2/\sigma_i^2}, \tag{93}$$

with *snr* given by:

$$\hat{s}_j/\sigma_{\hat{s}_j} = \frac{\sum_i b_{ij}d_{ij}/\sigma_i^2}{\sqrt{\sum_i b_{ij}^2/\sigma_i^2}}. \tag{94}$$

As for datasets of type `uvf` you can have CLIMAX output the data/model/residual images as FITS files by using the `dataimage/modelimage/resimage` keywords (for `mos` datasets, these will be in units of *snr* instead of Jy/beam). For mosaicked data, you can additionally output the noise image in Jy/beam, ($\sqrt{Eq.\ 93}$) by using the keyword `noiseimage`.

4.4.2 Mosaicked clean images

Equation 91 is the primary-beam corrected analog of Equation 80. We can construct a clean image in the same way as before by replacing d_{ij} with c_{ij}, the model convolved with a Gaussian approximation to the synthesized beam.

If the Högbom algorithm is used to construct a clean image, the keyword `cleancutoff` for mosaicked datasets is interpreted as *snr*.

4.5 Heterogeneous Visibility Datasets

As discussed in §4.1 above, heterogeneous interferometric datasets are handled correctly by the default `uvf` dataset type. For imaging purposes, however, beam-corrected images of heterogeneous datasets can be made by using the mosaicked datasets type, `mos`, as discussed above.

Note that heterogeneous datasets can be treated in the same way as mosaicked datasets, for map-making purposes. In this case the beam b_{ij} of Eq.87 represents the primary beam of each distinct antenna pairing i (and technically, at each individual frequency) in the heterogeneous array (for three antenna types there are 6), at the same spatial location j.

For the heterogeneous array, the beam for each pair consists of the cross-product of the aperture fields of the two antenna types. For example, if the current grading across dishes 1 and 2 are J_0 Bessel functions (for some parameters ρ), then:

$$\begin{aligned} A_1(\theta) &= \tilde{\mathcal{F}}(J_0(x,\rho_1)) \\ A_2(\theta) &= \tilde{\mathcal{F}}(J_0(x,\rho_2)) \end{aligned}$$

where $\tilde{\mathcal{F}}$ denotes inverse Fourier transform, with primary beam given by

$$B_{12}(\theta) = A_1 A_2{}^*, \tag{95}$$

or

$$b_{ij} = A_1(\theta_j) A_2(\theta_j)^*, \tag{96}$$

4.6 Image Datasets

4.6.1 Image Handling Basics

Reading Images

For all image datasets or models, images can be read in FITS format, either as standard FITS 2D images, or as columns from a FITS binary table — Climax attempts to detect the format automatically. If a binary table is detected, you will have to specify the binary table extension from which the read the data (`im.extname`), and the x, y and data columns of the table that comprise the image ((`im.xcol`), (`im.ycol`), and (`im.dcol`). For example:

```
im.extname = EVENTS;
im.xcol = x;
im.ycol = y;
im.dcol = energy;
```

would read an image from columns named `x`, `y` and `energy` of the binary table with extension name `EVENTS`.

Sub-image Selection

For all image datasets (or models), images can be manipulated on readin, to extract a particular region, resample an image at a different pixelation, smooth or otherwise transform the data. For example the following code:

```
im.region = 0,0 deg +- 0.1 deg;
```

would extract a ± 0.1 deg region about the center position. Note that the `region` specification is somewhat complex and can take many forms. For example `im.region = 0,0 +- 0.1 deg` in the example above would extract ± 0.1 deg about *pixel* (0,0) instead. Specifying `region = 65:192,193:320` would extract an subimage from pixel 65 to 192 in x and 193 to 320 in y, etc. Parsing a run file with an invalid region specification (i.e., `region = 0`) will show valid usages.

Image Resampling

Images can also be resampled after extraction, by using the `npix` keyword. Setting `im.npix = 128` would resample the image to 128×128 pixels; setting `im.npix = 64 x 128` would resample to 64 × 128. To preserve the noise properties of the image plane data, resampling is done by simple binning, rather than convolution, which would correlate the pixel noise. As a result, for fitting purposes, you should not rebin an image with finer resolution than the original, or you will duplicate pixels, and you will generate sampling artifacts (i.e., moiré patterns) if you resample an image by anything other than an integral divisor of the original image sampling.

Image Smoothing

For image models, images can be smoothed by convolution with a gaussian kernel, by using the `sigmasmooth` keyword. For example

```
im.sigmasmooth = 2'';
```

would convolve the extracted (and possibly resampled) image with gaussian smoothing kernel of $\sigma = 2$ arcseconds. Note that smoothing by convolution is done via FFT, and the image should be sampled to a power of 2, or you will produce unexpected artifacts. You should not convolve image datasets, or you will produce correlated pixel noise in the image plane.

Transforms

Simple transformations can also be applied to images, by using the `trans` keyword. Setting `im.trans = sqrt` for example, would take the square root of an image. Setting `trans` to any invalid transform name will show supported transform specifications.

4.6.2 Generic Image Dataset

Generic image-plane datasets, including support for point-spread functions (PSF) can be defined by using `adddataset` with `type = psfimage`.

Point-spread functions can be defined by using the `psf` keyword. For example:

```
adddataset type=psfimage name=im;
im.file = myfile.fits;
im.psf = gauss;
im.obs.ant.diameter = 10m;
im.obs.freqs = 90 GHz;
im.obs.bws = 1 GHz;
```

would convolve all models with an approximate gaussian PSF corresponding to a 10 m antenna diameter at 90 GHz. Note that the bandwidth is also required whenever a frequency is specified. Using `im.psf = realistic` calculates a more realistic primary beam by assuming a uniform current grading across the specified aperture. Using `psf = none` will compare models to the image data without any intermediate convolution (the default if `psf` is not specified).

Additionally, the `psfimage` dataset supports evaluation of either Gaussian or Poisson likelihoods. Poisson likelihood evaluation may be more appropriate for data dominated by low-count Poisson statistics (e.g., X-ray images). The type of distribution to use can be selected by using the `dist` keyword. The default for `psfimage` datasets is `dist = gauss`.

For Gaussian evaluation, a background (mean) and noise estimate (rms) are required. For Poisson evaluation, the background count rate is required.

These can be specified either manually, via the `background` and `noiserms` keywords, or you can specify a minimum radius for estimating these quantities from the image by using the `thetaMinErr` keyword, or you can specify a region from which to estimate them, by using the `errRegion` keyword.

An additional parameter is provided, via the `errImage` keyword for specifying whether quantities should be estimated from the original image or from the extracted (and possibly resampled) image. Think carefully about what you are doing here: if you are using Poisson evaluation and you are resampling an image, the background count rate should be estimated from the resampled image, or the estimated background rate will not match the data. Likewise, the noise rms estimated for Gaussian evaluation must match the noise in the final resampled image, or your statistics will not make sense.

Note that for Gaussian evaluation, the estimated background will be subtracted from the image.

4.6.3 Xray Image Dataset

Climax also provides a specialization of the PSF dataset type for x-ray image data, by using `adddataset` with `type = xrayimage`. The extension is currently useful for fitting of cluster models, where the shape of the cluster profile is inherently different for X-ray and SZ observations.

Additionally, the `xrayimage` dataset defaults to Poisson likelihood evaluation.

4.6.4 Radio Image Dataset

Climax also provides a specialization of the PSF dataset type for radio image data, by using `adddataset` with `type = radioimage`. The extension is currently useful for fitting of cluster models, where the shape of the cluster profile is inherently different for X-ray and SZ observations.

5 Consistency Tests

Here I collect together a number of internal and external consistency checks to verify correctness of the code and performance.

5.1 Side-by-side Comparison of β-model fitting with *Markov*

As a sanity check, Stephen and I ran a comparison of Climax and *Markov* fitting an asymmetric β-model to SZA data for Abell1914. Results are in Figure 4. The resulting chains give nearly identical results for all parameters (except that *Markov* prefers a slight shift (~ 5 arcsec) in the x-centroid for some reason that I have not investigated; I believe this corresponds roughly to the image pixelation in *Markov* and I suspect that the model centroid is erroneously shifted by 1 pixel). The Climax chain is significantly longer, since it took two orders of magnitude less time to run.

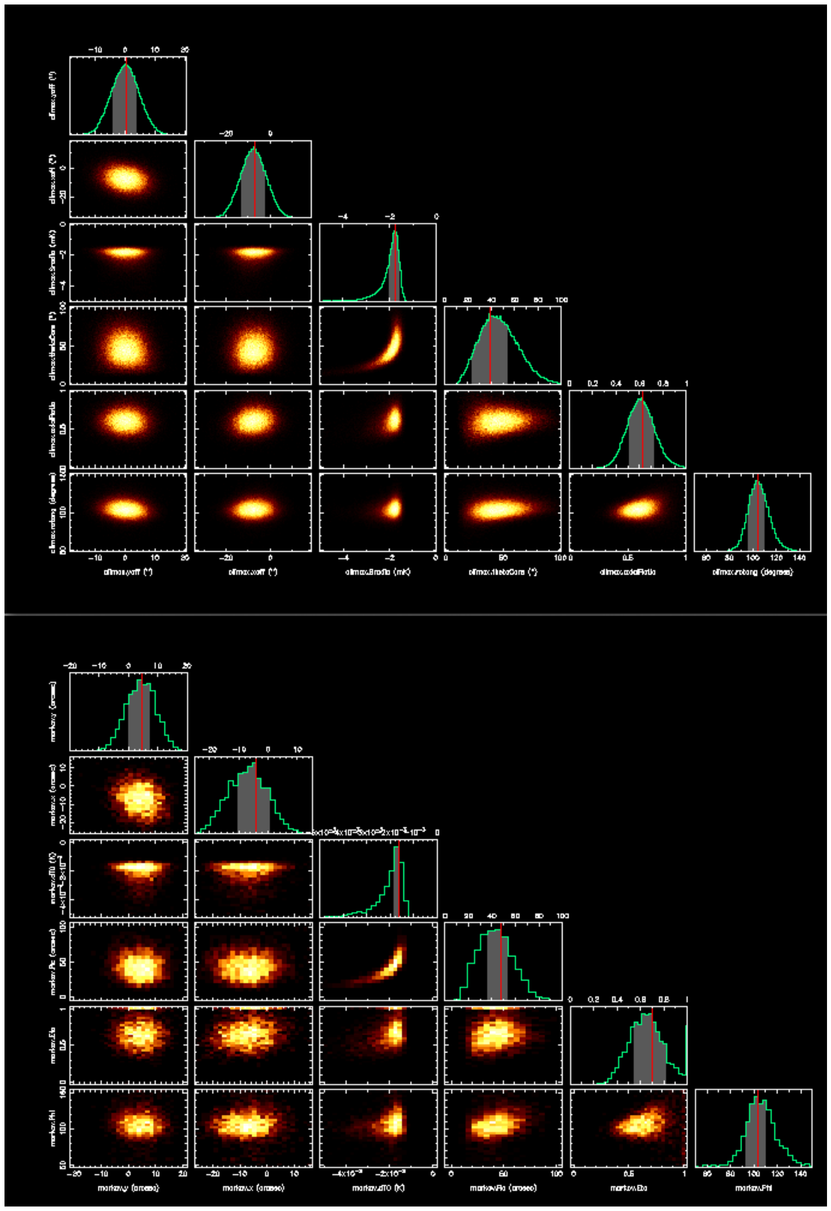

Figure 4: Top panel: Climax β-model run for A1914. Bottom panel: β-model run in *Markov* (for the same priors). Parameters are shown in the same order and to the same scale (note, however, that the Climax chain is significantly longer, hence the smoother distributions).

5.2 Sanity check on 3D radial-profile fitting

The β-model is defined as the 2D shape function

$$f(x) = \frac{1}{[1+x^2]^{(1-3\beta/2)}}, \tag{97}$$

frequently used to fit $y(\theta) = y(0)f(\theta)$, for example.

As a cross-check on Climax fitting using integrated 3D radial profiles, I can reconstruct the 2D beta model by starting the 3D radial profile (i.e., for an isothermal model $T(r) = T_0$ this would be a density model)

$$p(x) = \frac{1}{[1+x^2]^{3\beta/2}} \tag{98}$$

and performing the line integral to recover the 2D beta model shape.

Finally, I can use the GNFW model itself

$$p(x) = \frac{1}{x^{\gamma}\,[1+x^{\alpha}]^{(\epsilon-\gamma)/\alpha}} \tag{99}$$

which reduces to the β-model profile for $(\alpha, \epsilon, \gamma) = (2, 3\beta, 0)$. Then the identical code that generates the Arnaud and other variants of the GNFW models can be used to generate the integrated β-model as well.

The integrated profiles constructed in these last two ways match the analytic 2D β-model profiles to better than a part in 10^{-5}, and parameters derived from fitting either model show good agreement (see Figure 5). There are some small differences, however, notably a longer tail in θ_c for the β-model that is less pronounced in the integrated versions.

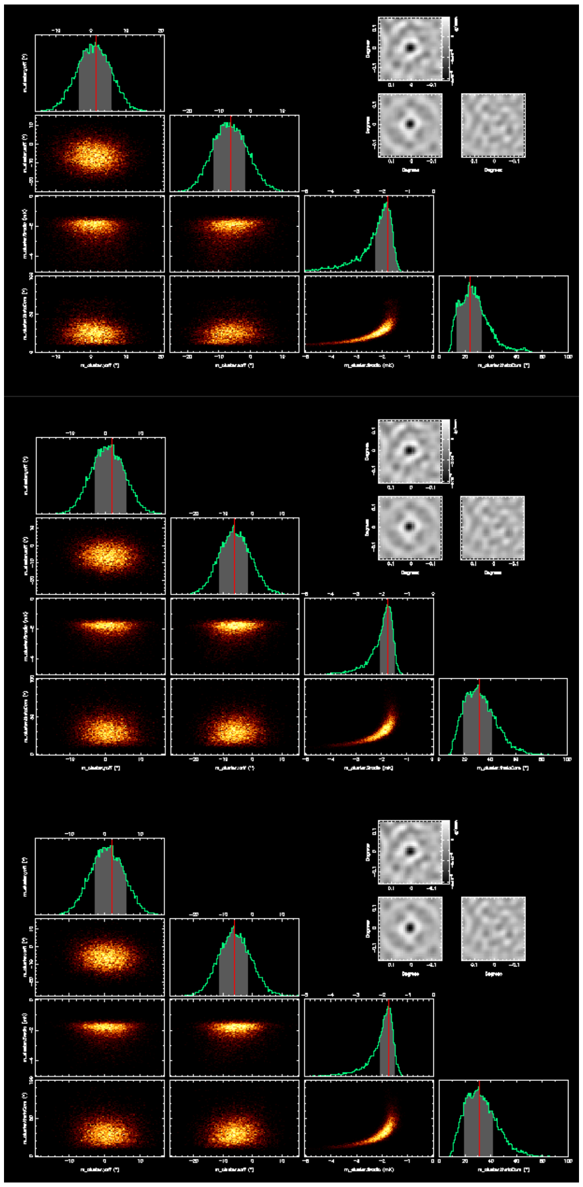

Figure 5: Top panel: Climax 2D β-model fits for A1914. Middle panel: integrated β-model fit in Climax using the 3D (i.e., density) profile. Bottom: Integrated β-model fit using the same GNFW profile code that generates the Arnaud model, etc. (with a generalization of the GNFW profile that allows it to reduce to the β-model profile for suitable choice of parameters). Images in the top right are, counter-clockwise from top left, dirty map of data, dirty map of model, residuals.

5.3 Side-by-side comparison of Nagai07 model fitting with *Markov*

Originally I was concerned that physical parameters I derived from my Arnaud09 model fits to A1914 didn't match the parameters given in Tony's paper (or even gave me ballpark numbers for A1914's total mass, see §3.2.2). This turns out to be an artifact of the particular data set I have been using for testing. Fits to the two datasets do indeed give fairly different results for θ_c but a direct comparison of fitting the Nagai07 model to the same dataset in CLIMAX and *Markov* shows good agreement (see Figure 6).

Moreover, using the Nagai07 model normalization, I determine

$$\begin{aligned} R_{500} &= 1.2 \pm 0.3 \text{ Mpc} \\ M_{tot500} &= 4.2 \pm 2.5 \times 10^{15}\ M_{\odot} \end{aligned}$$

where the R_{500} and M_{tot500} constraints comes directly from the Nagai07 model fits.

This is to be compared with $R_{500} = 1.3{\pm}0.1$ Mpc from Tony's N07 fits, and $M_{tot500} = 2{-}3{\times}10^{15}\ M_{\odot}$ I've seen in the literature.

By comparison, the Arnaud09 model fits I described in 3.2.2 yield $R_{500} = 0.43$ Mpc and $M_{tot500} = 8 \times 10^{15}\ M_{\odot}$. But those fits were not to the same dataset used in this analysis. I therefore conclude that the 3D radial profile-fitting in CLIMAX is consistent with the fitting in *Markov* and that the discrepancy between the results derived here and the results described in § 3.2.2 do not indicate any problem with the fitting code, but rather one of three things:

1 a problem with that data set

2 a disagreement between the Nagai07 model normalization and the Arnaud09 model normalization

3 they indicate that there are large systematic errors that are not showing up in the formal errors from the Markov chains. For instance, the likelihood for the GNFW model parameter space may be multimoded, implying that very different combinations of P_{e0} and θ_c can provide a good fit to the outer cluster profile and therefore to the SZ data, even though they imply very different physical results.

At this point, I think the first possibility is likely, since a Nagai07 model fit to that data give a similar normalization to the data fit in this section, but a θ_c that is smaller by nearly 60% (hence implying a larger pressure for the same y normalization and therfore a larger mass).

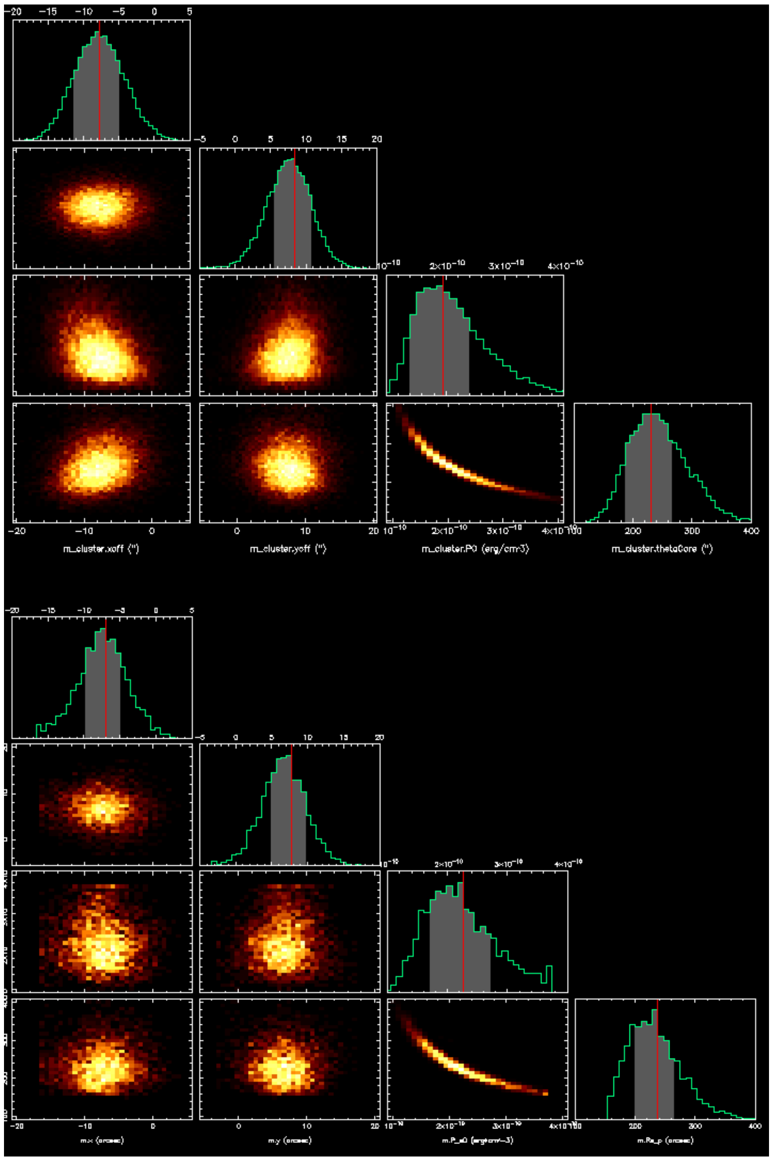

Figure 6: Top panel: CLIMAX 2D Nagai07 model fits for A1914 (NB: this is to the same dataset that Tony used for his profile paper, which gives significantly different results at least for θ_c to the dataset I used for the β-model comparison). Bottom: Nagai07 model fit in *Markov*. Third colum in the CLIMAX plot is the pressure derived from the temperature normalization fit to the data, in units of erg/cm^3, to compare to the P_{e0} output by *Markov*.

6 Implementation Details

6.1 Sampling from multi-variate normal distributions with arbitrary correlations

The multi-variate distribution is given by:

$$p(\bar{x}) = \frac{1}{(2\pi)^{k/2}|\tilde{C}|^{1/2}} \exp -\frac{1}{2}(\bar{x}-\bar{\mu})^T \tilde{C}^{-1}(\bar{x}-\bar{\mu}) \tag{100}$$

If we write out the terms of the exponent $M \equiv (\bar{x}-\bar{\mu})^T \tilde{C}^{-1}(\bar{x}-\bar{\mu})$ we have:

$$M = {x_k}^2 C_{kk} - 2x_k\mu_k C_{kk} + 2\sum_{i\neq k} x_k(x_i-\mu_i)C_{ki} + C_{kk}B_k \tag{101}$$

where B_k/C_{kk} represents all terms that don't depend on x_k. If we rewrite this as:

$$\begin{aligned}
M &= C_{kk}{x_k}^2 + C_{kk}x_k\left(2\sum_{i\neq k}(x_i-\mu_i)C_{ki}/C_{kk} - 2\mu_k\right) + C_{kk}B_k && (102)\\
&= C_{kk}{x_k}^2 + C_{kk}x_kA_k + C_{kk}B_k && (103)\\
&= C_{kk}\left({x_k}^2 + x_kA_k + A_k^2/4\right) + C_{kk}\left(B_k - A_k^2/4\right) && (104)\\
&= C_{kk}\left(x_k + A_k/2\right)^2 + C_{kk}\left(B_k - A_k^2/4\right) && (105)
\end{aligned}$$

we see that if $\mu_k' \equiv -A_k/2$, the exponent can be written, up to a constant factor, as:

$$M_k = C_{kk}\left(x_k - \mu_k'\right)^2. \tag{106}$$

We see then that up to a normalization, the conditional distribution for x_k, $p(x_k|\{x_{i\neq k}\})$ is simply given by a univariate gaussian with:

$$\begin{aligned}
\mu_k' &= -A_k/2 && (107)\\
\sigma_k &= 1/\sqrt{C_{kk}} && (108)
\end{aligned}$$

Our prescription for generating samples from the joint distribution is then to Gibbs sample from the univariate conditional distributions, starting from a random point, and iterating each time from the previous point. The result is itself a Markov chain which collectively represents an unbiased sample from the joint distribution.

6.2 Convergence

Following Dunkley et al. (2008), we model the power spectrum of each variate as:

$$P(k) = P_0 \frac{(k^*/k)^\alpha}{(k^*/k)^\alpha + 1} \tag{109}$$

P_0 is an estimate of the sample variance, and k^* is the wavenumber where the power spectrum turns over to white noise.

Implementation Notes

We fit the power spectrum to this function, for which we require the derivatives w.r.t. P_0, k^* and α. Letting

$$f = (k^*/k)^\alpha, \tag{110}$$

we have:

$$\begin{aligned} \frac{df}{dk^*} &= f\,\frac{\alpha}{k^*} \\ \frac{df}{d\alpha} &= f\,\ln(k^*/k) \end{aligned}$$

and

$$\begin{aligned} \frac{dP}{dP_0} &= \frac{f}{f+1} \\ \frac{dP}{dk^*} = \frac{dP}{df}\frac{df}{dk^*} &= \frac{f}{f+1} \times P_0 \left[1 - \frac{f}{f+1}\right] \frac{\alpha}{k^*} \\ \frac{dP}{d\alpha} = \frac{dP}{df}\frac{df}{d\alpha} &= \frac{f}{f+1} \times P_0 \left[1 - \frac{f}{f+1}\right] \ln(k^*/k) \end{aligned}$$

Note that the convergence test relies on using the power spectrum of the chain $\tilde{P}(k)$ to estimate the true sample variance P_0. Thus the power spectrum should be normalized so that $P(k=0)$ is an estimate of the sample variance. For real transforms in FFTW, the discrete Fourier transform is unnormalized, or

$$Y_k = \sum_{j=0}^{N-1} X_j e^{-2\pi jk\sqrt{-1}/N} \tag{111}$$

For $k = 0$, we see that

$$Y_0 = \sum_{j=0}^{N-1} X_j, \tag{112}$$

and

$$
\begin{aligned}
Y_0^2 &= \left(\sum_{j=0}^{N-1} X_j\right)^2 && (113)\\
&= \sum_j X_j^2 + \sum_i \sum_{j\neq i} X_i X_j && (114)\\
&= N\left\langle X^2 \right\rangle + 0, && (115)
\end{aligned}
$$

assuming the X_i and X_j are uncorrelated. Therefore, in order for the power spectrum to reflect the variance, we instead compute the power spectrum of the normalized samples $X_j' = X_j/\sqrt{N}$.

Once P_0

6.3 Maximum Likelihood Estimators

6.3.1 Gaussian Data

The joint likelihood for a set of Gaussian-distributed data is simply the product of the individual probabilities:

$$
\begin{aligned}
L \equiv \prod p_k &\propto \prod_k e^{-(x_k-\mu)^2/2\sigma_k^2} && (116)\\
&= e^{-\sum_k (x_k-\mu)^2/2\sigma_k^2} && (117)\\
&\equiv e^{-\chi^2/2} && (118)\\
& && (119)
\end{aligned}
$$

The maximum likelihood estimate of a set of parameters is therefore, by definition, the parameters that minimize χ^2.

6.3.2 Poisson Data

What if our data are Poisson distributed, say a an X-ray image, where each pixel is the number of photons, and does not necessarily contain enough photons that the gaussian approximation is even remotely justified? In this case, the joint likelihood is still the product of the individual probabilities, but where each probability is given by the Poisson pdf:

$$
L \equiv \prod_k p_k = \prod_k \frac{e^{-\mu_k}\mu_k^k}{k!} \qquad (120)
$$

Here μ_k is the mean expected count rate in pixel k, the sum of the model prediction and the background count rate.

6.3.3 Joint Likelihood Calculation

For chains that involve a combination of Gaussian and Poisson-distributed data, the likelihood is straightforwardly calculated as

$$L \equiv \prod_i L_i, \tag{121}$$

where i ranges over the independent datasets, with L_i calculated as above for Gaussian or Poisson datasets. For purposes of reported goodness of fit (χ_r^2), an equivalent χ^2 distributed is constructed for Poisson data, noting that:

$$f_N(\chi^2) = \frac{1}{2\Gamma(N/2)} e^{-\chi^2/2} \left(\frac{\chi^2}{2}\right)^{N/2-1} \tag{122}$$

and

$$P(\mu; k) = \frac{1}{k!} e^{-\mu} \mu^k \tag{123}$$

with the identification of

$$\chi^2 = 2\mu \tag{124}$$

and

$$N = 2(k+1) \tag{125}$$

6.4 Model Selection

Given a model class M and a vector of model parameters θ, we can factor the joint distribution $P(\theta, D|M)$ as:

$$P(\theta, D|M) = P(\theta|D, M)\, P(D|M) = P(D|\theta, M)\, P(\theta|M), \tag{126}$$

which gives us Bayes' Theorem:

$$P(\theta|D, M) = \frac{P(D|\theta, M)\, P(\theta|M)}{P(D|M)} \tag{127}$$

We identify $P(\theta|M)$ as the prior, $P(D|\theta, M)$ as the likelihood, and $P(\theta|D, M)$ as the posterior distribution for θ, of which the chain is an estimate.

The quantity $P(D|M)$ is known as the *evidence*, and can be seen to play the role of a normalizing factor:

$$P(D|M) = \int P(D|\theta, M)\, P(\theta|M)\, d\theta. \tag{128}$$

In the context of parameter fitting, $P(D|M)$ is usually left unspecified, as only the ratio of the posterior is used to choose between different values of parameters.

When comparing different classes of models, however, the evidence plays a central role, as it represents the probability of the data given a particular model, marginalized over all possible values of that model's parameters θ.

When considering two classes of models, the evidence ratio $\mathcal{E}$ is defined as:

$$\frac{P(D|M_1)}{P(D|M_2)} = \frac{\int P(D|\alpha, M_1)\, P(\alpha|M_1)\, d\alpha}{\int P(D|\beta, M_2)\, P(\beta|M_2)\, d\beta} \tag{129}$$

Note that the evidence ratio contains the usual "goodness of fit" criterion in the form of the likelihood $P(D|M)$; i.e., if our two "hypotheses " just consist of two sets of parameter values $P(\alpha|M_1) = \delta(\alpha_0)$ and $P(\beta|M_2) = \delta(\beta_0)$, the we recover the usual likelihood ratio:

$$\mathcal{E} = \frac{P(D|\alpha_0)}{P(D|\beta_0)} \tag{130}$$

If the hypotheses are more complex, however, the evidence ratio retains a dependence on the integral over the priors, something that is often referred to as the *Ockham factor*; i.e., models are penalized if only a small part of the prior parameter range matches the data. If for example, our hypotheses consisted of two different priors for the same parameters $\vec{\alpha}$, where the likelihood was 1 for $\vec{\alpha} < \vec{\alpha}_{max}$ and 0 everwhere else, and where H_1 allowed only a fraction f of parameter space over which the likelihood was non-zero, while H_2 allowed the full range of parameter space over which the likelihood was non-zero, then

$$\mathcal{E} = \frac{\int P(\alpha|M_1)\, d\alpha}{\int P(\alpha|M_2)\, d\alpha} = f, \tag{131}$$

that is, H_1 would be disfavored over H_2 by that same factor f, even though both hypotheses contain some range of $\vec{\alpha}$ over which the model fits the data equally well.

In the context of Markov chains, one way to estimate the evidence proceeds as follows:

$$P(\theta|D, M)\, P(D|M) = P(D|\theta, M)\, P(\theta|M) \tag{132}$$

or

$$P(D|M)\, \frac{P(\theta|D, M)}{P(D|\theta, M)} = P(\theta|M) \tag{133}$$

whence

$$P(D|M) \int \frac{P(\theta|D,M)}{P(D|\theta,M)}\, d\theta = \int P(\theta|M)\, d\theta = 1. \tag{134}$$

The term

$$\left\langle P(D|\theta,M)^{-1} \right\rangle \equiv \int \frac{P(\theta|D,M)}{P(D|\theta,M)}\, d\theta \tag{135}$$

is just the average of $P(D|\theta,M)^{-1}$ over the posterior. In the context of Markov chains, then, the quantity

$$\hat{Z} \equiv \frac{1}{\langle P(D|\theta,M)^{-1} \rangle}, \tag{136}$$

where $\langle P(D|\theta,M)^{-1} \rangle$ is the average value of the inverse likelihood over the chain, can be taken as an estimate of the evidence.

7 To-do List

Sort out why sampling and unused variate screws up the chain, i.e., cosmo z when z is not used by any model

Change updateHessian to slew the mean, and investigate performance updating individual parameters

Read miriad files in visdataset

Remove static state variables for use in python

Make sure state memory doesn't screw up plotting, etc, when used from python

Add logging

add external setting of primary beams

References

Arnaud, M., Pratt, G. W., Piffaretti, R., Böhringer, H., Croston, J. H., & Pointecouteau, E. 2010, A&A, 517, A92

Dunkley, J., Bucher, M., Ferreira, P. G., Moodley, K., & Skordis, C. 2008, 000

Högbom, J. A. 1974, A&AS, 15, 417

Nagai, D., Kravtsov, A. V., & Vikhlinin, A. 2007, ApJ, 668, 1