Beyond PLANCK

Deliverable 1.1:
Proto Gibbs sampler

Authors Kristian Joten Andersen
Hans Kristian Eriksen
Trygve Leithe Svalheim
Ingunn Kathrine Wehus

Date May 31st, 2018

Work Package WP6 – Component separation

Revision History

Version	Authors	Date	Changes
1.0	Kristian Joten Andersen Hans Kristian Eriksen Trygve Leithe Svalheim Ingunn Kathrine Wehus	May 24th, 2018	Initial Version

Contents

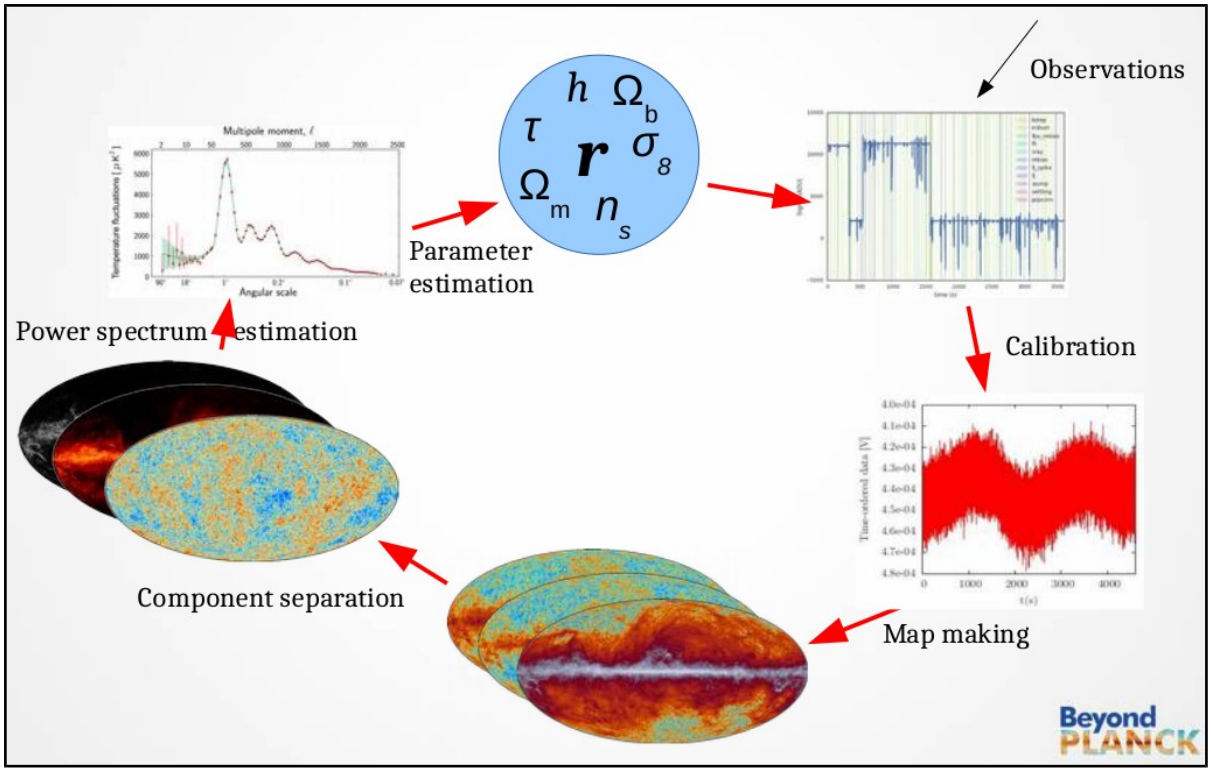

Figure 1: Overview of Gibbs sampling chain.

1 Overview

The main goal of the BeyondPlanck project is to build an end-to-end Gibbs sampler for the Planck LFI data, and use this to improve the overall calibration and fidelity of the final LFI sky maps. This Gibbs sampler is illustrated in Figure 1, and may be summarized by the following steps:

1. Data selection and calibration
2. Map making
3. Component separation
4. Power spectrum estimation
5. Parameter estimation

These five steps are then iterated until convergence, where each step uses information produced by previous steps. The development of this Gibbs sampler is scheduled to take about 12 months. The “proto Gibbs sampler” presented in this deliverable is essentially a “skeleton code” or “blueprint” with placeholders for each module, but no actual functionality, that will allow different groups to structure their common work.

2 Software

We provide two different proto Gibbs samplers, adopting different balances between integration and efficiency versus code effort in the two approaches.

The first version takes the form of a bash script called “BPipe” that runs existing codes sequentially. The script is available here:

- https://gitlab.com/BeyondPlanck/repo/tree/master/BPipe

In this case, the individual Gibbs sampling modules are available in the following repository directories:

1. Data selection and calibration: calib/
2. Map making: madam/
3. Component separation: commander/
4. Power spectrum estimation: like/
5. Parameter estimation: like/

The main advantage of this approach is cheap development efforts, since most codes already exist in their basic form, and most coding efforts go into defining common interfaces and data formats between the codes. The main disadvantage of this approach is the fact that all communication between codes happens via disk IO, which takes a non-negligible run time.

To run the code, the user must compile each of the above codes, and then call the executable as follows:

- ./BPipe path_to_repo/BPipe/param_bpipe.txt

In parallel with the above low-risk/low-cost approach, we also develop a computationally more efficient approach, based on tight on-the-fly integration within the existing Commander2 Gibbs sampling code. The main advantage of this approach is faster run times, since no intermediate data needs to be written to disk. This proto-Gibbs sampler is available within the Commander2 code:

- https://gitlab.com/BeyondPlanck/repo/tree/master/commander2

The existing code already supports component separation and power spectrum and parameter estimation, but lacks time-ordered data capabilities. We have therefore added new support for these operations through the following Fortran modules :

- commander2/src/commander/comm_tod_mod.f90
- commander2/src/commander/comm_gain_mod.f90
- commander2/src/commander/comm_pointing_mod.f90
- commander2/src/commander/comm_N_tod_mod.f90

We stress that the current version of these modules is not yet operational. Instead, only interfaces have been defined at this early stage, allowing different developers to contribute to the code in parallel.

For how to compile and run Commander2, see Deliverable 6.2 “Commander module”.