Deliverable 6.2: Commander module

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

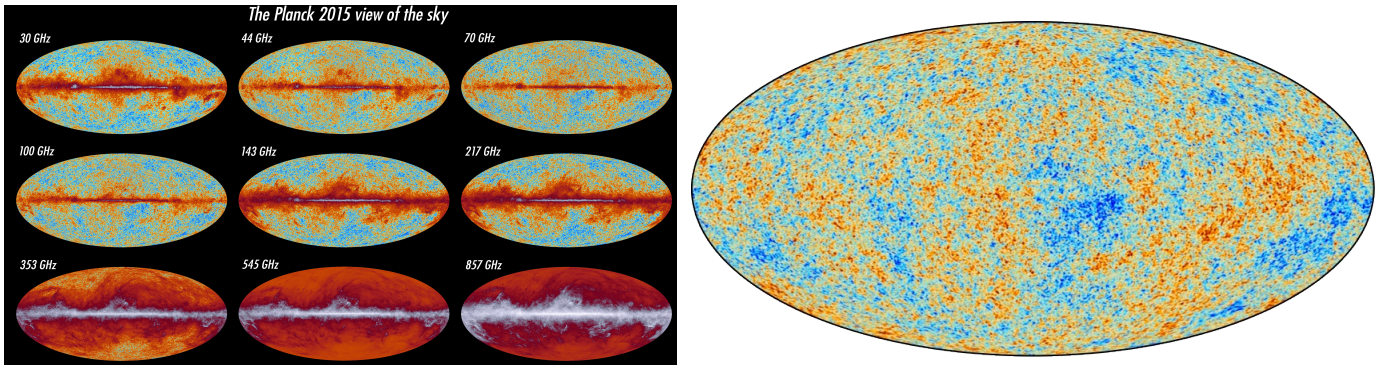

Figure 1: Component separation is the process of going from frequency maps to component maps.The left panel shows the Planck 2015 temperature maps, while the right panel shows the Planck 2015 CMB temperature map. We employ the Commander program to perform this operation in this project.

1 Overview

Component separation is the process of extracting physical component maps from complicated data sets, and forms a critical component of the BeyondPlanck project. For this task, we will employ the Commander computer program, which is a Bayesian Gibbs sampler that fits an explicit parametric model to given data. This program is described in Eriksen et al. (2008), and has already been used extensively in the Planck project.

The development of the Commander suite has happened over more than a decade, and we will take full advantage of these existing investments. For the BeyondPlanck project, we have adapted this suite into our GitLab project repository, with suitable modifications in order to allow it to compile and run in a stand-alone mode.

Two different versions of the Commander algorithm has been adapted, referred to as Commander1 and Commander2. Commander1 is a very mature implementation, with support for a wide range of parameters, but only supports data sets with uniform angular resolution across frequencies. Commander2 is currently under active development, and supports multi-resolution component separation. Both will be used in the BeyondPlanck processing.

An important part of the work regarding the Commander module has been training in their use for both Kristian Joten Andersen and Trygve Leithe Svalheim. They are now both able to run these codes independently of Wehus and Eriksen, and this will be essential in the further BeyondPlanck processing.

2 Software and compilation

The Commander1 and Commander2 modules are available in the GitLab repository, under the directories

- https://gitlab.com/BeyondPlanck/repo/tree/master/commander1
- https://gitlab.com/BeyondPlanck/repo/tree/master/commander2

To compile these codes, the user must have access to the following compilers and libraries:

- Fortran MPI compiler, for instance Intel
- CFITSIO, https://heasarc.gsfc.nasa.gov/fitsio/fitsio.html
- Lapack, for instance MKL
- HEALPix, http://healpix.sourceforge.net/
- HDF5, https://support.hdfgroup.org/HDF5/

Options for compilers and libraries are defined in {path_to_commander}/config. Several different configuration files are available, and which one being used is controlled with the COMMANDER environment variable.

For compilation and execution within the owl compute environment, we recommend adding the following flags to ~/.bashrc:

- export COMMANDER=ita_owl
- export OMPI_FC=ifort
- module load Intel_composer_xe/2017/1.132
- module load openmpi/Intel/2.1.1
- export HEALPIX={path_to_healpix}

With these environment variables defined in the correct configuration file, the code can be compiled as follows:

- cd git_repository/commander{1,2}
- make

The resulting executable is available in git_repository/commander{1,2}/src/commander

3 Usage

Both commander1 and commander2 are run as follows:

- mpirun -n N git_repository/commander{1,2}/src/commander param.txt

Example parameter files are available in git_repository/commander{1,2}/params. In addition to the above basic MPI command, we recommend the following useful statements, to ensure both correct OpenMP parallelization and proper recording of input and output:

- export OMP_NUM_THREADS=1; cp param.txt chains/; mpirun -n N git_repository/commander{1,2}/src/commander param.txt 2>&1 | tee chains/slurm.txt

where chains/ is the desired output directory and slurm.txt is a log file.

The number of cores, N, is determined differently for commander1 and commander2. For commander1, it must be equal to the product of {number of chains} x {number of bands} x {number of cores per band}. For commander2, it may be set to any number, typically the same number of cores as is available on the compute nodes employed for the analysis.

Both commander1 and commander2 employ hybrid MPI and OpenMP parallelization, and the above flags may be modified accordingly to increase the number of OpenMP threads. However, MPI parallelization is typically more efficient.

For interpretation of outputs, see Planck Collaboration (2015) and Svalheim's Master thesis, “Improving the Planck sky maps with Bayesian component separation”.

4 References

Joint Bayesian component separation and CMB power spectrum estimation, Eriksen, H. K., Jewell, J. B., Dickinson, C., Banday, A. J., Górski, K. M. and Lawrence, C. R., 2008, ApJ, 676, 10

Planck 2015 results. X. Diffuse component separation: Foreground maps, Planck Collaboration, 2016, A&A, 594, A10

Improving the Planck sky maps with Bayesian component separation, Svalheim, T. L., 2018, Master thesis, in preparation