intro

November 1, 2019

1 Introduction

This lesson is a brief introduction to TOAST and its data representations. This next cell is just initializing some things for the notebook.

```
In [1]: # Load common tools for all lessons
        import sys
        sys.path.insert(0, "..")
        from lesson_tools import (
            fake_focalplane
        )

        # Capture C++ output in the jupyter cells
        %load_ext wurlitzer
```

1.1 Runtime Environment

You can get the current TOAST runtime configuration from the "Environment" class.

```
In [2]: import toast

        env = toast.Environment.get()
        print(env)
```

```
<toast.Environment
  Source code version = 2.3.1.dev1428
  Logging level = INFO
  Handling enabled for 0 signals:
  Max threads = 2
  MPI build enabled
  MPI runtime disabled
  Cannot use MPI on NERSC login nodes
>
```

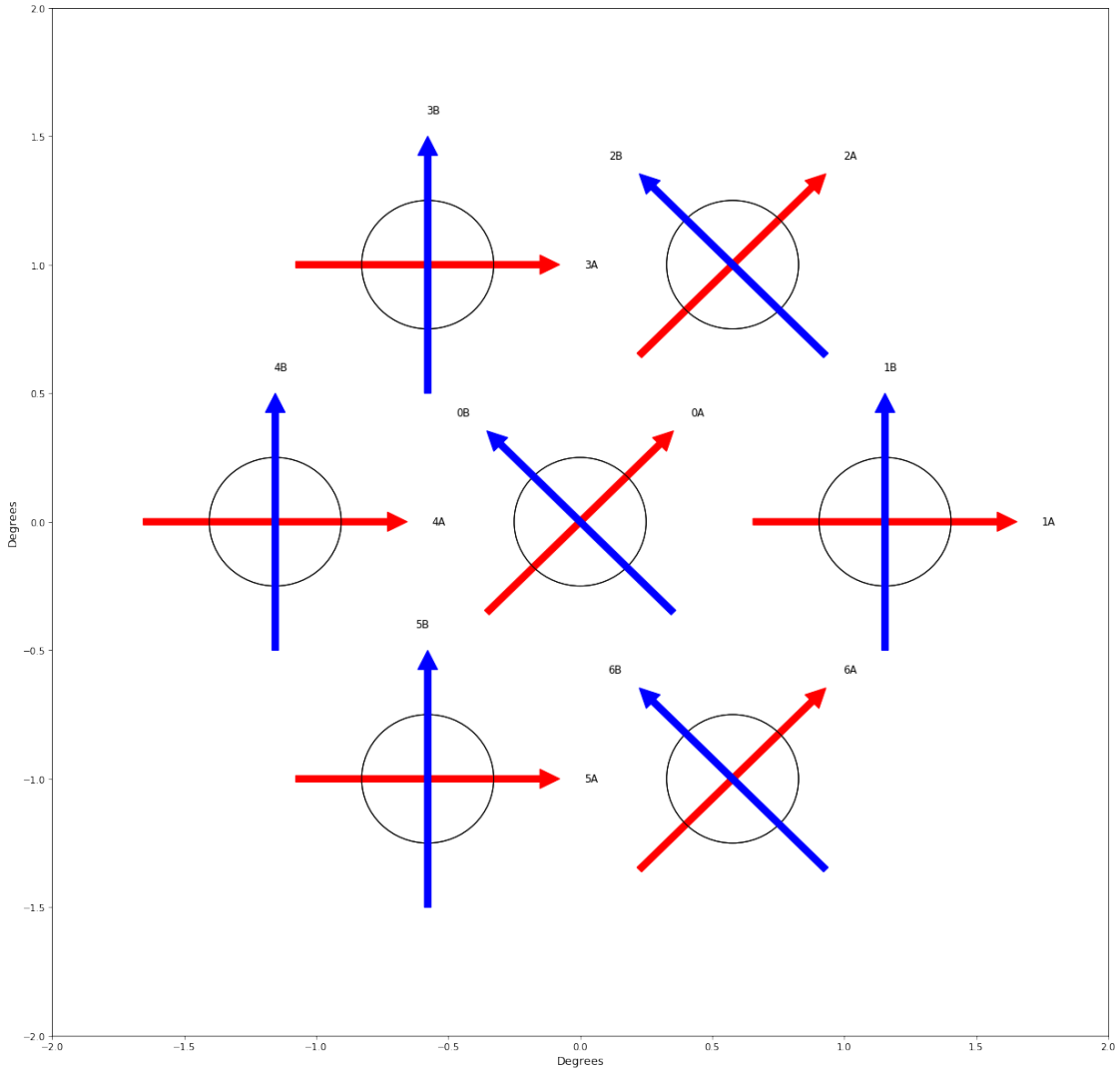
1.2 Data Model

Before using TOAST for simulation or analysis, it is important to discuss how data is stored in memory and how that data can be distributed among many processes to parallelize large workflows.

First, let's create a fake focalplane of detectors to use throughout this example.

In [3]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

# Generate a fake focalplane with 7 pixels, each with 2 detectors.

fp = fake_focalplane()
```

In [4]:
```
# Make a plot of this focalplane layout.

detnames = list(sorted(fp.keys()))
detquat = {x: fp[x]["quat"] for x in detnames}
detfwhm = {x: fp[x]["fwhm_arcmin"] for x in detnames}
detlabels = {x: x for x in detnames}
detpolcol = {x: "red" if i % 2 == 0 else "blue" for i, x in enumerate(detnames)}

toast.tod.plot_focalplane(
    detquat, 4.0, 4.0, None, fwhm=detfwhm, polcolor=detpolcol, labels=detlabels
);
```



1.2.1 Observations with Time Ordered Data

TOAST works with data organized into *observations*. Each observation is independent of any other observation. An observation consists of co-sampled detectors for some span of time. The intrinsic detector noise is assumed to be stationary within an observation. Typically there are other quantities which are constant for an observation (e.g. elevation, weather conditions, satellite spin axis, etc).

An observation is just a dictionary with at least one member ("tod") which is an instance of a class that derives from the `toast.TOD` base class.

The inputs to a TOD class constructor are at least:

1. The detector names for the observation.
2. The number of samples in the observation.
3. The geometric offset of the detectors from the boresight.

4. Information about how detectors and samples are distributed among processes. More on this below.

The TOD class can act as a storage container for different "flavors" of timestreams as well as a source and sink for the observation data (with the read_*() and write_*() methods):

```
In [5]: import toast.qarray as qa

        nsamples = 1000

        obs = dict()
        obs["name"] = "20191014_000"

In [6]: # The type of TOD class is usually specific to the data processing job.
        # For example it might be one of the simulation classes or it might be
        # a class that loads experiment data.  Here we just use a simple class
        # that is only used for testing and which reads / writes data to internal memory
        # buffers.

        tod = toast.tod.TODCache(None, detnames, nsamples, detquats=detquat)
        obs["tod"] = tod

In [7]: # Print the tod to get summary info:
        print(tod)

<TODCache
  14 total detectors and 1000 total samples
  Using MPI communicator None
    In grid dimensions 1 sample ranks x 1 detranks
  Process at (0, 0) in grid has data for:
    Samples 0 - 999 (inclusive)
    Detectors:
      0A
      0B
      1A
      1B
      2A
      2B
      3A
      3B
      4A
      4B
      5A
      5B
      6A
      6B
    Cache contains 0 bytes
>
```

In [8]:
```
# The TOD class has methods to get information about the data:

print("TOD has detectors {}".format(", ".join(tod.detectors)))
print("TOD has {} total samples for each detector".format(tod.total_samples))
```

```
TOD has detectors 0A, 0B, 1A, 1B, 2A, 2B, 3A, 3B, 4A, 4B, 5A, 5B, 6A, 6B
TOD has 1000 total samples for each detector
```

In [9]:
```
# Write some data.  Not every TOD derived class supports writing (for example,
# TOD classes that represent simulations).

t_delta = 1.0 / fp[detnames[0]]["rate"]
tod.write_times(stamps=np.arange(0.0, nsamples * t_delta, t_delta))
tod.write_boresight(
    data=qa.from_angles(
        (np.pi / 2) * np.ones(nsamples),
        (2 * np.pi / nsamples) * np.arange(nsamples),
        np.zeros(nsamples)
    )
)
for d in detnames:
    tod.write(detector=d, data=np.random.normal(scale=fp[d]["NET"], size=nsamples))
    tod.write_flags(detector=d, flags=np.zeros(nsamples, dtype=np.uint8))
```

In [10]:
```
# Read it back

 print("TOD timestamps = {} ...".format(tod.read_times()[:5]))
 print("TOD boresight = \n{} ...".format(tod.read_boresight()[:5,:]))
 for d in detnames:
     print("TOD detector {} = {} ...".format(d, tod.read(detector=d, n=5)))
     print("TOD detector {} flags = {} ...".format(d, tod.read_flags(detector=d, n=5)))
```

```
TOD timestamps = [0.   0.05 0.1  0.15 0.2 ] ...
TOD boresight =
[[ 7.07106781e-01  0.00000000e+00  7.07106781e-01  1.11022302e-16]
 [ 7.07103292e-01  2.22143781e-03  7.07103292e-01 -2.22143781e-03]
 [ 7.07092824e-01  4.44285371e-03  7.07092824e-01 -4.44285371e-03]
 [ 7.07075377e-01  6.66422575e-03  7.07075377e-01 -6.66422575e-03]
 [ 7.07050951e-01  8.88553201e-03  7.07050951e-01 -8.88553201e-03]] ...
TOD detector 0A = [ 0.4160208  -0.39853823  0.30923787 -0.05840396  0.63575638] ...
TOD detector 0A flags = [0 0 0 0 0] ...
TOD detector 0B = [ 1.66789846  0.22500044  0.02281229 -0.35858763  1.38152883] ...
TOD detector 0B flags = [0 0 0 0 0] ...
TOD detector 1A = [ 1.01885935 -0.50995926 -0.94325906 -0.63304825 -0.38301689] ...
TOD detector 1A flags = [0 0 0 0 0] ...
TOD detector 1B = [-1.04668157  0.53042507  0.43963395  0.90080722 -1.72627901] ...
TOD detector 1B flags = [0 0 0 0 0] ...
TOD detector 2A = [-0.92469301  1.528144   -0.62495257 -0.42823959 -0.58968093] ...
```

```
TOD detector 2A flags = [0 0 0 0 0] ...
TOD detector 2B = [ 1.50664351 -0.06504427 -1.0535867  -0.11695765 -1.90191136] ...
TOD detector 2B flags = [0 0 0 0 0] ...
TOD detector 3A = [ 1.46601394  0.12227309 -1.50616774 -1.29255702  0.51476471] ...
TOD detector 3A flags = [0 0 0 0 0] ...
TOD detector 3B = [-0.47062199 -0.55157412 -1.33499313 -0.20891842 -1.2546331 ] ...
TOD detector 3B flags = [0 0 0 0 0] ...
TOD detector 4A = [-1.56684962  1.1158182  -0.05114078 -0.61027597  0.37299293] ...
TOD detector 4A flags = [0 0 0 0 0] ...
TOD detector 4B = [ 0.59707082  0.50935595  0.53863295  0.35191755 -1.01376266] ...
TOD detector 4B flags = [0 0 0 0 0] ...
TOD detector 5A = [-0.63125525  0.13216398 -1.15961638 -0.1650204   2.29205647] ...
TOD detector 5A flags = [0 0 0 0 0] ...
TOD detector 5B = [-0.03208431 -0.22108621  0.12870683  1.38101898 -0.47175519] ...
TOD detector 5B flags = [0 0 0 0 0] ...
TOD detector 6A = [-0.36146864  0.66954733 -0.80967795 -1.54563102  0.24238121] ...
TOD detector 6A flags = [0 0 0 0 0] ...
TOD detector 6B = [ 1.92747613  1.5150788  -2.23038363 -0.23790163  0.70023375] ...
TOD detector 6B flags = [0 0 0 0 0] ...
```

```
In [11]: # Store some data in the cache.  The "cache" member variable looks like a dictionary
         # numpy arrays, but the memory used is allocated in C, so that we can actually clear
         # these buffers when needed.

         for d in detnames:
             processed = tod.read(detector=d)
             processed /= 2.0
             # By convention, we usually name buffers in the cache by <prefix>_<detector>
             tod.cache.put("processed_{}".format(d), processed)
         print("TOD cache now contains {} bytes".format(tod.cache.report(silent=True)))
```

```
TOD cache now contains 278000 bytes
```

1.2.2 Comm : Groups of Processes

A toast.Comm instance takes the global number of processes available (MPI.COMM_WORLD) and divides them into groups. Each process group is assigned one or more observations. Since observations are independent, this means that different groups can be independently working on separate observations in parallel. It also means that inter-process communication needed when working on a single observation can occur with a smaller set of processes.

At NERSC, this notebook is running on a login node, so we cannot use MPI. Constructing a default `toast.Comm` whenever MPI use is disabled will just produce a single group of one process. See the parallel example at the end of this notebook for a case with multiple groups.

```
In [12]: comm = toast.Comm()
         print(comm)
```

```
<toast.Comm
  World MPI communicator = None
  World MPI size = 1
  World MPI rank = 0
  Group MPI communicator = None
  Group MPI size = 1
  Group MPI rank = 0
  Rank MPI communicator = None
>
```

1.2.3 Data : a Collection of Observations

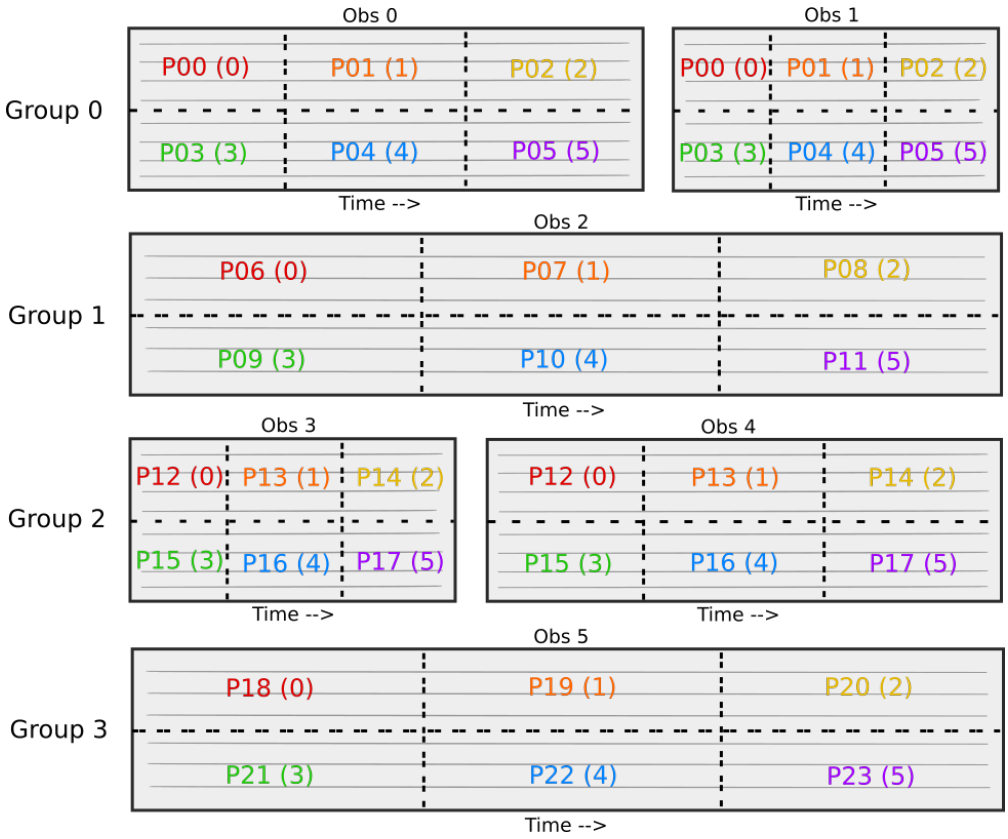
A toast.Data instance is mainly just a list of observations. However remember that each process group will have a different list. Since we have only one group of one process, this example is not so interesting. See the parallel case at the end of the notebook.

```
In [13]: data = toast.Data(comm)
         data.obs.append(obs)
```

1.2.4 Data Distribution

Recapping previous sections, we have some groups of processes, each of which has a set of observations. Within a single process group, the detector data is distributed across the processes within the group. That distribution is controlled by the size of the communicator passed to the TOD class, and also by the `detranks` parameter of the constructor. This detranks number sets the dimension of the process grid in the detector direction. For example, a value of "1" means that every process has all detectors for some span of time. A value equal to the size of the communicator results in every process having some number of detectors for the entire observation. The detranks parameter must divide evenly into the number of processes in the communicator and determines how the processes are arranged in a grid.

As a concrete example, imagine that MPI.COMM_WORLD has 24 processes. We split this into 4 groups of 6 procesess. There are 6 observations of varying lengths and every group has one or 2 observations. Here is a picture of what data each process would have. The global process number is shown as well as the rank within the group:



The parallel script at the bottom of this notebook has further examples of data distribution.

1.3 Utilities

There are many utilities in the TOAST package that use compiled code "under the hood". These include:

- `toast.rng`: Streamed random number generation, with support for generating random samples from any location within a stream.
- `toast.qarray`: Vectorized quaternion operations.
- `toast.fft`: API Wrapper around different vendor FFT packages.
- `toast.cache`: Class for dictionary of C-allocated numpy arrays.
- `toast.healpix`: Subset of pixel projection routines, simd vectorized and threaded.
- `toast.timing`: Simple serial timers, global named timers per process, a decorator to time calls to functions, and MPI tools to gather timing statistics from multiple processes.

1.3.1 Random Number Example

Here is a quick example of a threaded generation of random numbers drawn from a unit-variance gaussian distribution. Note the "key" pair of uint64 values and the first value of the "counter" pair determine the stream, and the second value of the counter pair is effectively the sample in that stream. We can drawn randoms from anywhere in the stream in a reproducible fashion (i.e. this random generator is stateless). Under the hood, this uses the Random123 package on each thread.

```
In [14]: import toast.rng as rng

         # Number of random samples
         nrng = 10

In [15]: # Draw randoms from the beginning of a stream
         rng1 = rng.random(
             nrng, key=[12, 34], counter=[56, 0], sampler="gaussian", threads=True
         )

In [16]: # Draw randoms from some later starting point in the stream
         rng2 = rng.random(
             nrng, key=[12, 34], counter=[56, 4], sampler="gaussian", threads=True
         )

In [17]: # The returned objects are buffer providers, so can be used like a numpy array.
         print("Returned RNG buffers:")
         print(rng1)
         print(rng2)

Returned RNG buffers:
<AlignedF64 10 elements: -1.93456 0.0142729 ... 1.00982 0.190195>
<AlignedF64 10 elements: -0.661366 0.0715733 ... -0.696467 1.31699>


In [18]: # Compare the elements.  Note how the overlapping sample indices match.  The
         # randoms drawn for any given sample agree regardless of the starting sample.
         print("------ rng1 ------")
         for i in range(nrng):
             print("rng1 {}:  {}".format(i, rng1[i]))
         print("------ rng2 ------")
         for i in range(nrng):
             print("rng2 {}:  {}".format(i + 4, rng2[i]))

------ rng1 ------
rng1 0:  -1.9345557154123538
rng1 1:  0.014272943598316932
rng1 2:  0.45226011897093527
rng1 3:  -1.957844901847933
rng1 4:  -0.6613662026904461
rng1 5:  0.07157330335682659
rng1 6:  -0.6785937530814287
rng1 7:  1.1896892565821966
rng1 8:  1.0098155098534913
rng1 9:  0.1901951884823849
------ rng2 ------
rng2 4:  -0.6613662026904461
rng2 5:  0.07157330335682659
rng2 6:  -0.6785937530814287
```

```
rng2 7:  1.1896892565821966
rng2 8:  1.0098155098534913
rng2 9:  0.1901951884823849
rng2 10:  0.04850349309129753
rng2 11:  0.6481059989834956
rng2 12:  -0.6964671519913512
rng2 13:  1.3169924602685523
```

1.3.2 Quaternion Array Example

The quaternion manipulation functions internally attempt to improve performance using OpenMP SIMD directives and threading in cases where it makes sense. The Python API is modelled after the quaternionarray package (https://github.com/zonca/quaternionarray/). There are functions for common operations like multiplying quaternion arrays, rotating arrays of vectors, converting to and from angle representations, SLERP, etc.

```
In [19]: import toast.qarray as qa

         # Number points for this example

         nqa = 5
```

```
In [20]: # Make some fake rotation data by sweeping through theta / phi / pa angles

         theta = np.linspace(0.0, np.pi, num=nqa)
         phi = np.linspace(0.0, 2 * np.pi, num=nqa)
         pa = np.zeros(nqa)
         print("----- input angles -----")
         print("theta = ", theta)
         print("phi = ", phi)
         print("pa = ", pa)
```

```
----- input angles -----
theta =  [0.         0.78539816 1.57079633 2.35619449 3.14159265]
phi =  [0.         1.57079633 3.14159265 4.71238898 6.28318531]
pa =  [0. 0. 0. 0. 0.]
```

```
In [21]: # Convert to quaternions

         quat = qa.from_angles(theta, phi, pa)

         print("\n----- output quaternions -----")
         print(quat)
```

```
----- output quaternions -----
[[ 0.00000000e+00  0.00000000e+00  1.00000000e+00  2.22044605e-16]
```

```
 [ 2.70598050e-01  2.70598050e-01  6.53281482e-01 -6.53281482e-01]
 [ 0.00000000e+00  7.07106781e-01  1.11022302e-16 -7.07106781e-01]
 [-6.53281482e-01  6.53281482e-01 -2.70598050e-01 -2.70598050e-01]
 [-1.00000000e+00  1.11022302e-16 -6.12323400e-17 -1.84889275e-32]]
```

In [22]:
```
# Use these to rotate a vector

zaxis = np.array([0.0, 0.0, 1.0])
zrot = qa.rotate(quat, zaxis)

print("\n---- Z-axis rotated by quaternions ----")
print(zrot)
```

```
---- Z-axis rotated by quaternions ----
[[ 0.00000000e+00  0.00000000e+00  1.00000000e+00]
 [ 0.00000000e+00  7.07106781e-01  7.07106781e-01]
 [-1.00000000e+00  1.57009246e-16  2.22044605e-16]
 [-2.77555756e-16 -7.07106781e-01 -7.07106781e-01]
 [ 1.22464680e-16 -5.05741657e-32 -1.00000000e+00]]
```

In [23]:
```
# Rotate different vector by each quaternion

zout = qa.rotate(quat, zrot)

print("\n---- Arbitrary vectors rotated by quaternions ----")
print(zout)
```

```
---- Arbitrary vectors rotated by quaternions ----
[[ 0.00000000e+00  0.00000000e+00  1.00000000e+00]
 [ 7.07106781e-01  5.00000000e-01  5.00000000e-01]
 [-4.44089210e-16  3.14018492e-16 -1.00000000e+00]
 [ 7.07106781e-01  5.00000000e-01  5.00000000e-01]
 [ 0.00000000e+00  7.39557099e-32  1.00000000e+00]]
```

In [24]:
```
# Multiply two quaternion arrays

qcopy = np.array(quat)
qout = qa.mult(quat, qcopy)

print("\n---- Product of two quaternion arrays ----")
print(qout)
```

```
---- Product of two quaternion arrays ----
```

```
[[ 0.00000000e+00  0.00000000e+00  4.44089210e-16 -1.00000000e+00]
 [-3.53553391e-01 -3.53553391e-01 -8.53553391e-01 -1.46446609e-01]
 [ 0.00000000e+00 -1.00000000e+00 -1.57009246e-16  2.22044605e-16]
 [ 3.53553391e-01 -3.53553391e-01  1.46446609e-01 -8.53553391e-01]
 [ 3.69778549e-32  0.00000000e+00  0.00000000e+00 -1.00000000e+00]]
```

```
In [25]: # SLERP quaternions

         qtime = 3.0 * np.arange(nqa)
         qtargettime = np.arange(3.0 * (nqa - 1) + 1)
         qslerped = qa.slerp(qtargettime, qtime, quat)

         print("\n---- SLERP input ----")
         for t, q in zip(qtime, quat):
             print("t = {} : {}".format(t, q))

         print("\n---- SLERP output ----")
         for t, q in zip(qtargettime, qslerped):
             print("t = {} : {}".format(t, q))
```

```
---- SLERP input ----
t = 0.0 : [0.00000000e+00 0.00000000e+00 1.00000000e+00 2.22044605e-16]
t = 3.0 : [ 0.27059805  0.27059805  0.65328148 -0.65328148]
t = 6.0 : [ 0.00000000e+00  7.07106781e-01  1.11022302e-16 -7.07106781e-01]
t = 9.0 : [-0.65328148  0.65328148 -0.27059805 -0.27059805]
t = 12.0 : [-1.00000000e+00  1.11022302e-16 -6.12323400e-17 -1.84889275e-32]

---- SLERP output ----
t = 0.0 : [0.00000000e+00 0.00000000e+00 1.00000000e+00 2.22044605e-16]
t = 1.0 : [ 0.10093174  0.10093174  0.95929668 -0.24367078]
t = 2.0 : [ 0.19364697  0.19364697  0.84050023 -0.46750515]
t = 3.0 : [ 0.27059805  0.27059805  0.65328148 -0.65328148]
t = 4.0 : [ 0.19364697  0.45739433  0.46750515 -0.7312525 ]
t = 5.0 : [ 0.10093174  0.60695567  0.24367078 -0.74969471]
t = 6.0 : [ 0.00000000e+00  7.07106781e-01  1.11022302e-16 -7.07106781e-01]
t = 7.0 : [-0.24367078  0.74969471 -0.10093174 -0.60695567]
t = 8.0 : [-0.46750515  0.7312525  -0.19364697 -0.45739433]
t = 9.0 : [-0.65328148  0.65328148 -0.27059805 -0.27059805]
t = 10.0 : [-0.84050023  0.46750515 -0.19364697 -0.19364697]
t = 11.0 : [-0.95929668  0.24367078 -0.10093174 -0.10093174]
t = 12.0 : [-1.00000000e+00  1.11022302e-16 -6.12323400e-17 -1.84889275e-32]
```

1.3.3 FFT Example

The internal FFT functions in TOAST are very limited and focus only on batched 1D Real FFTs. These are used for simulated noise generation and timestream filtering. Internally the compiled

code can use either FFTW or MKL for the backend calculation.

```
In [26]: # Number of batched FFTs

         nbatch = 5

         # FFT length

         nfft = 65536
```

```
In [27]: # Create some fake data

         infft = np.zeros((nbatch, nfft), dtype=np.float64)
         for b in range(nbatch):
             infft[b, :] = rng.random(nfft, key=[0, 0], counter=[b, 0], sampler="gaussian")

         print("----- FFT input -----")
         print(infft)
```

```
----- FFT input -----
[[ 0.70824414  0.37958204 -0.62191667 ...  0.77954278  1.12705665
   0.21430745]
 [ 0.61372834  0.68541016 -0.82076732 ... -0.10495521 -0.09820802
   0.1019775 ]
 [-0.25952652 -0.41435533  0.57094265 ...  0.32334547  2.15186219
  -0.10320734]
 [-0.18910596 -1.50802991 -1.65130707 ...  0.42610591  1.17170788
   1.54472861]
 [-1.4962988  -0.00235431 -1.50373344 ...  0.97027549  0.94612694
   0.80572723]]
```

```
In [28]: # Forward FFT

         outfft = toast.fft.r1d_forward(infft)

         print("\n----- FFT output -----")
         print(outfft)
```

```
----- FFT output -----
[[ 428.42060388 -138.03674787 -181.0470772  ... -403.91871131
   -80.09506675 -210.21894707]
 [-236.80183938 -293.36911153 -299.89827436 ... -265.36475328
    79.30045561 -152.79215102]
 [ 407.62843731 -321.18213065  149.8744358  ...  115.39517
   161.42894263   -8.53315689]
 [-353.92938439 -163.58217755  199.29345241 ...   41.98100849
  -151.62213508  227.94024389]
```

```
 [-287.73541094  -94.54962966  -41.95149462 ...  121.73873742
   200.54675492  233.86103108]]
```

```
In [29]: # Reverse FFT

         backfft = toast.fft.r1d_backward(outfft)

         print("\n----- FFT inverse output -----")
         print(backfft)
```

```
----- FFT inverse output -----
[[ 0.70824414  0.37958204 -0.62191667 ...  0.77954278  1.12705665
   0.21430745]
 [ 0.61372834  0.68541016 -0.82076732 ... -0.10495521 -0.09820802
   0.1019775 ]
 [-0.25952652 -0.41435533  0.57094265 ...  0.32334547  2.15186219
  -0.10320734]
 [-0.18910596 -1.50802991 -1.65130707 ...  0.42610591  1.17170788
   1.54472861]
 [-1.4962988  -0.00235431 -1.50373344 ...  0.97027549  0.94612694
   0.80572723]]
```

1.3.4 Cache Example

The Cache class provides a mechanism to work around the Python memory pool. There are times when we want to allocate memory and explicitly free it without waiting for garbage collection. Every instance of a `toast.Cache` acts as a dictionary of numpy arrays. Internally, the memory of each entry is a flat-packed std::vector with a custom allocator that ensures aligned memory allocation. Aligned memory is required for SIMD operations both in TOAST and in external libraries. Buffers in a Cache instance can be used directly for such operations.

```
In [30]: from toast.cache import Cache

         # Example array dimensions

         cnames = ["c1", "c2"]
         cshapes = {
             "c1" : (20,),
             "c2" : (2, 3, 2)
         }
         ctyps = {
             "c1" : np.float64,
             "c2" : np.uint16
         }
```

```
In [31]: # A cache instance

         cache = Cache()

In [32]: # Create some empty arrays in the cache

         for cn in cnames:
             cache.create(cn, ctyps[cn], cshapes[cn])

         print("---- Cache object ----")
         print(cache)
         print("\n---- Now contains ----")
         for cn in cnames:
             print("{}:  {}".format(cn, cache.reference(cn)))
         print("Size = ", cache.report(silent=True), " bytes")
```

```
---- Cache object ----
<toast.cache.Cache object at 0x2aaae7643128>

---- Now contains ----
c1:  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
c2:  [[[0 0]
  [0 0]
  [0 0]]

 [[0 0]
  [0 0]
  [0 0]]]
Size =  184  bytes
```

```
In [33]: # Fill existing buffers

         # Get a reference to the buffer
         cdata = cache.reference("c1")

         # Assign elements.
         cdata[:] = np.random.random(cshapes["c1"])

         # Delete the reference
         del cdata

In [34]: cdata = cache.reference("c2")
         idx = 0
         for x in range(cshapes["c2"][0]):
             for y in range(cshapes["c2"][1]):
                 for z in range(cshapes["c2"][2]):
                     cdata[x, y, z] = idx
                     idx += 1
```

```
        del cdata

        print("\n---- Contents after filling ----")
        for cn in cnames:
            print("{}:  {}".format(cn, cache.reference(cn)))
        print("Size = ", cache.report(silent=True), " bytes")
```

```
---- Contents after filling ----
c1:  [0.71355378 0.45563995 0.80373043 0.01834192 0.59432237 0.71313407
 0.22963687 0.80161985 0.72275687 0.0762314  0.80865342 0.97946115
 0.16575266 0.80597926 0.24744005 0.40421217 0.17940692 0.77569579
 0.82491902 0.74607358]
c2:  [[[ 0  1]
  [ 2  3]
  [ 4  5]]

 [[ 6  7]
  [ 8  9]
  [10 11]]]
Size =  184  bytes
```

```
In [35]: # We can also "put" existing numpy arrays which will then be copied into
         # the cache

         np1 = np.random.normal(size=10)
         np2 = np.random.randint(0, high=255, dtype=np.uint16, size=12).reshape((2, 3, 2))

         cache.put("p1", np1)
         cache.put("p2", np2)

         print("\n---- Contents after putting numpy arrays ----")

         for cn in list(cache.keys()):
             print("{}:  {}".format(cn, cache.reference(cn)))
         print("Size = ", cache.report(silent=True), " bytes")
```

```
---- Contents after putting numpy arrays ----
c1:  [0.71355378 0.45563995 0.80373043 0.01834192 0.59432237 0.71313407
 0.22963687 0.80161985 0.72275687 0.0762314  0.80865342 0.97946115
 0.16575266 0.80597926 0.24744005 0.40421217 0.17940692 0.77569579
 0.82491902 0.74607358]
c2:  [[[ 0  1]
  [ 2  3]
  [ 4  5]]
```

```
 [[ 6  7]
  [ 8  9]
  [10 11]]]
p1:  [ 0.29977207 -1.14387477 -0.06382724 -0.10071886 -0.34697721  1.57076212
  1.05939354  0.60970975  0.81818009  0.75791165]
p2:  [[[ 53 234]
  [  0  94]
  [209  96]]

 [[247  65]
  [ 78 250]
  [245  94]]]
Size =  288  bytes
```

1.4 Running the Test Suite

TOAST includes extensive tests built in to the package. Running all of them takes some time, but you can also run just one test by specifying the name of the file in the toast/tests directory (without the ".py" extension):

```
In [36]: import toast.tests

         # Run just a couple simple tests in toast/tests/env.py
         toast.tests.run("env")
```

```
test_comm (toast.tests.env.EnvTest) ...

<toast.Comm
  World MPI communicator = None
  World MPI size = 1
  World MPI rank = 0
  Group MPI communicator = None
  Group MPI size = 1
  Group MPI rank = 0
  Rank MPI communicator = None
>


ok
test_env (toast.tests.env.EnvTest) ...

<toast.Environment
  Source code version = 2.3.1.dev1428
  Logging level = INFO
  Handling enabled for 0 signals:
  Max threads = 2
  MPI build enabled
```

```
  MPI runtime disabled
  Cannot use MPI on NERSC login nodes
>


ok

----------------------------------------------------------------------
Ran 2 tests in 0.003s

OK


Out[36]: 0

In [37]: # Now run **ALL** the (serial) tests
         toast.tests.run()


test_comm (toast.tests.env.EnvTest) ...

Note: Google Test filter = TOAST*
[==========] Running 46 tests from 9 test cases.
[----------] Global test environment set-up.
[----------] 2 tests from TOASTenvTest
[ RUN      ] TOASTenvTest.print
TOAST ENV: Source code version = 2.3.1.dev1428
TOAST ENV: Logging level = INFO
TOAST ENV: Handling enabled for 0 signals:
TOAST ENV: Max threads = 2
TOAST ENV: MPI build enabled
TOAST ENV: MPI runtime disabled
TOAST ENV: Cannot use MPI on NERSC login nodes
[       OK ] TOASTenvTest.print (0 ms)
[ RUN      ] TOASTenvTest.setlog
[       OK ] TOASTenvTest.setlog (1 ms)
[----------] 2 tests from TOASTenvTest (1 ms total)

[----------] 3 tests from TOASTutilsTest
[ RUN      ] TOASTutilsTest.logging
Testing level CRITICAL
TOAST CRITICAL: This message level is CRITICAL
TOAST CRITICAL: This message level is CRITICAL at  (file "/global/homes/k/kisner/git/toast/src/
Testing level ERROR
TOAST CRITICAL: This message level is CRITICAL
TOAST CRITICAL: This message level is CRITICAL at  (file "/global/homes/k/kisner/git/toast/src/
TOAST ERROR: This message level is ERROR
TOAST ERROR: This message level is ERROR at  (file "/global/homes/k/kisner/git/toast/src/libtoa
Testing level WARNING
```

```
TOAST CRITICAL: This message level is CRITICAL
TOAST CRITICAL: This message level is CRITICAL at  (file "/global/homes/k/kisner/git/toast/src/
TOAST ERROR: This message level is ERROR
TOAST ERROR: This message level is ERROR at  (file "/global/homes/k/kisner/git/toast/src/libtoa
TOAST WARNING: This message level is WARNING
TOAST WARNING: This message level is WARNING at  (file "/global/homes/k/kisner/git/toast/src/li
Testing level INFO
TOAST CRITICAL: This message level is CRITICAL
TOAST CRITICAL: This message level is CRITICAL at  (file "/global/homes/k/kisner/git/toast/src/
TOAST ERROR: This message level is ERROR
TOAST ERROR: This message level is ERROR at  (file "/global/homes/k/kisner/git/toast/src/libtoa
TOAST WARNING: This message level is WARNING
TOAST WARNING: This message level is WARNING at  (file "/global/homes/k/kisner/git/toast/src/li
TOAST INFO: This message level is INFO
TOAST INFO: This message level is INFO at  (file "/global/homes/k/kisner/git/toast/src/libtoast
Testing level DEBUG
TOAST CRITICAL: This message level is CRITICAL
TOAST CRITICAL: This message level is CRITICAL at  (file "/global/homes/k/kisner/git/toast/src/
TOAST ERROR: This message level is ERROR
TOAST ERROR: This message level is ERROR at  (file "/global/homes/k/kisner/git/toast/src/libtoa
TOAST WARNING: This message level is WARNING
TOAST WARNING: This message level is WARNING at  (file "/global/homes/k/kisner/git/toast/src/li
TOAST INFO: This message level is INFO
TOAST INFO: This message level is INFO at  (file "/global/homes/k/kisner/git/toast/src/libtoast
TOAST DEBUG: This message level is DEBUG
TOAST DEBUG: This message level is DEBUG at  (file "/global/homes/k/kisner/git/toast/src/libtoa
[       OK ] TOASTutilsTest.logging (0 ms)
[ RUN      ] TOASTutilsTest.singletimer
TOAST INFO: Test timer stopped:  0.20 seconds (1 calls)
TOAST ERROR: Timer is still running! (file "/global/homes/k/kisner/git/toast/src/libtoast/src/t
This should throw since timer not stopped...
Timer is still running!
TOAST INFO: Original:  0.20 seconds (1 calls)
TOAST INFO: Copied:  0.20 seconds (1 calls)
TOAST INFO: Original was running:  0.20 seconds (1 calls)
TOAST INFO: Original was stopped:  0.00 seconds (1 calls)
[       OK ] TOASTutilsTest.singletimer (605 ms)
[ RUN      ] TOASTutilsTest.globaltimer
TOAST ERROR: Cannot stop timer timer1 which does not exist (file "/global/homes/k/kisner/git/to
This should throw since timer timer1 not yet created
Cannot stop timer timer1 which does not exist
TOAST ERROR: Cannot stop timer timer2 which does not exist (file "/global/homes/k/kisner/git/to
This should throw since timer timer2 not yet created
Cannot stop timer timer2 which does not exist
TOAST ERROR: Cannot stop timer timer3 which does not exist (file "/global/homes/k/kisner/git/to
This should throw since timer timer3 not yet created
Cannot stop timer timer3 which does not exist
TOAST INFO: Global timer: timer1:  0.20 seconds (1 calls)
```

```
TOAST INFO: Global timer: timer2:  0.40 seconds (1 calls)
TOAST INFO: Global timer: timer3:  0.60 seconds (1 calls)
TOAST INFO: Global timer: timer1:  0.20 seconds (1 calls)
TOAST INFO: Global timer: timer2:  0.20 seconds (1 calls)
TOAST INFO: Global timer: timer3:  0.20 seconds (1 calls)
[       OK ] TOASTutilsTest.globaltimer (801 ms)
[----------] 3 tests from TOASTutilsTest (1406 ms total)

[----------] 5 tests from TOASTsfTest
[ RUN      ] TOASTsfTest.trig
[       OK ] TOASTsfTest.trig (1 ms)
[ RUN      ] TOASTsfTest.fasttrig
[       OK ] TOASTsfTest.fasttrig (0 ms)
[ RUN      ] TOASTsfTest.sqrtlog
[       OK ] TOASTsfTest.sqrtlog (0 ms)
[ RUN      ] TOASTsfTest.fast_sqrtlog
[       OK ] TOASTsfTest.fast_sqrtlog (1 ms)
[ RUN      ] TOASTsfTest.fast_erfinv
[       OK ] TOASTsfTest.fast_erfinv (0 ms)
[----------] 5 tests from TOASTsfTest (2 ms total)

[----------] 8 tests from TOASTrngTest
[ RUN      ] TOASTrngTest.reprod
[       OK ] TOASTrngTest.reprod (0 ms)
[ RUN      ] TOASTrngTest.reprod_multi
[       OK ] TOASTrngTest.reprod_multi (0 ms)
[ RUN      ] TOASTrngTest.uniform11
[       OK ] TOASTrngTest.uniform11 (0 ms)
[ RUN      ] TOASTrngTest.uniform11_multi
[       OK ] TOASTrngTest.uniform11_multi (0 ms)
[ RUN      ] TOASTrngTest.uniform01
[       OK ] TOASTrngTest.uniform01 (0 ms)
[ RUN      ] TOASTrngTest.uniform01_multi
[       OK ] TOASTrngTest.uniform01_multi (0 ms)
[ RUN      ] TOASTrngTest.uint64
[       OK ] TOASTrngTest.uint64 (0 ms)
[ RUN      ] TOASTrngTest.uint64_multi
[       OK ] TOASTrngTest.uint64_multi (0 ms)
[----------] 8 tests from TOASTrngTest (0 ms total)

[----------] 18 tests from TOASTqarrayTest
[ RUN      ] TOASTqarrayTest.arraylist_dot1
[       OK ] TOASTqarrayTest.arraylist_dot1 (0 ms)
[ RUN      ] TOASTqarrayTest.arraylist_dot2
[       OK ] TOASTqarrayTest.arraylist_dot2 (0 ms)
[ RUN      ] TOASTqarrayTest.inv
[       OK ] TOASTqarrayTest.inv (0 ms)
[ RUN      ] TOASTqarrayTest.norm
```

```
[       OK ] TOASTqarrayTest.norm (0 ms)
[ RUN      ] TOASTqarrayTest.mult
[       OK ] TOASTqarrayTest.mult (0 ms)
[ RUN      ] TOASTqarrayTest.multarray
[       OK ] TOASTqarrayTest.multarray (0 ms)
[ RUN      ] TOASTqarrayTest.rot1
[       OK ] TOASTq<toast.Comm
  World MPI communicator = None
  World MPI size = 1
  World MPI rank = 0
  Group MPI communicator = None
  Group MPI size = 1
  Group MPI rank = 0
  Rank MPI communicator = None
>


ok
test_env (toast.tests.env.EnvTest) ...

arrayTest.rot1 (0 ms)
[ RUN      ] TOASTqarrayTest.rotarray
[       OK ] TOASTqarrayTest.rotarray (0 ms)
[ RUN      ] TOASTqarrayTest.slerp
[       OK ] TOASTqarrayTest.slerp (1 ms)
[ RUN      ] TOASTqarrayTest.rotation
[       OK ] TOASTqarrayTest.rotation (0 ms)
[ RUN      ] TOASTqarrayTest.toaxisangle
[       OK ] TOASTqarrayTest.toaxisangle (0 ms)
[ RUN      ] TOASTqarrayTest.exp
[       OK ] TOASTqarrayTest.exp (0 ms)
[ RUN      ] TOASTqarrayTest.ln
[       OK ] TOASTqarrayTest.ln (0 ms)
[ RUN      ] TOASTqarrayTest.pow
[       OK ] TOASTqarrayTest.pow (0 ms)
[ RUN      ] TOASTqarrayTest.torotmat
[       OK ] TOASTqarrayTest.torotmat (1 ms)
[ RUN      ] TOASTqarrayTest.fromrotmat
[       OK ] TOASTqarrayTest.fromrotmat (0 ms)
[ RUN      ] TOASTqarrayTest.fromvectors
[       OK ] TOASTqarrayTest.fromvectors (0 ms)
[ RUN      ] TOASTqarrayTest.thetaphipa
[       OK ] TOASTqarrayTest.thetaphipa (0 ms)
[----------] 18 tests from TOASTqarrayTest (2 ms total)

[----------] 4 tests from TOASTfftTest
[ RUN      ] TOASTfftTest.roundtrip_single
[       OK ] TOASTfftTest.roundtrip_single (0 ms)
```

```
[ RUN      ] TOASTfftTest.roundtrip_multi
[       OK ] TOASTfftTest.roundtrip_multi (0 ms)
[ RUN      ] TOASTfftTest.plancache_single
[       OK ] TOASTfftTest.plancache_single (0 ms)
[ RUN      ] TOASTfftTest.plancache_multi
[       OK ] TOASTfftTest.plancache_multi (0 ms)
[----------] 4 tests from TOASTfftTest (0 ms total)

[----------] 1 test from TOASThealpixTest
[ RUN      ] TOASThealpixTest.pixelops
[       OK ] TOASThealpixTest.pixelops (1 ms)
[----------] 1 test from TOASThealpixTest (1 ms total)

[----------] 3 tests from TOASTcovTest
[ RUN      ] TOASTcovTest.accumulate
[       OK ] TOASTcovTest.accumulate (0 ms)
[ RUN      ] TOASTcovTest.eigendecompose
[       OK ] TOASTcovTest.eigendecompose (58 ms)
[ RUN      ] TOASTcovTest.matrixmultiply
[       OK ] TOASTcovTest.matrixmultiply (0 ms)
[----------] 3 tests from TOASTcovTest (58 ms total)

[----------] 2 tests from TOASTpolyfilterTest
[ RUN      ] TOASTpolyfilterTest.filter
[       OK ] TOASTpolyfilterTest.filter (1 ms)
[ RUN      ] TOASTpolyfilterTest.filter_with_flags
[       OK ] TOASTpolyfilterTest.filter_with_flags (0 ms)
[----------] 2 tests from TOASTpolyfilterTest (1 ms total)

[----------] Global test environment tear-down
[==========] 46 tests from 9 test cases ran. (1471 ms total)
[  PASSED  ] 46 tests.
<toast.Environment
  Source code version = 2.3.1.dev1428
  Logging level = DEBUG
  Handling enabled for 0 signals:
  Max threads = 2
  MPI build enabled
  MPI runtime disabled
  Cannot use MPI on NERSC login nodes
>


ok
test_alias (toast.tests.cache.CacheTest) ... ok
test_clear (toast.tests.cache.CacheTest) ... ok
test_create (toast.tests.cache.CacheTest) ... ok
test_create_none (toast.tests.cache.CacheTest) ... ok
```

```
test_memfree (toast.tests.cache.CacheTest) ...

Memory usage Before large buffer creation
       total :  503.054 GB
   available :  196.486 GB
     percent :   60.900 %
        used :  289.185 GB
        free :  209.875 GB
      active :  267.938 GB
    inactive :    9.509 GB
     buffers :  354.637 MB
      cached :    3.648 GB
      shared :  184.684 MB
        slab :    1.870 GB

Memory usage After large buffer creation
       total :  503.054 GB
   available :  196.392 GB
     percent :   61.000 %
        used :  289.278 GB
        free :  209.781 GB
      active :  268.031 GB
    inactive :    9.509 GB
     buffers :  354.637 MB
      cached :    3.648 GB
      shared :  184.684 MB
        slab :    1.870 GB

Cache now has 100000000 bytes
Memory usage After cache clear
       total :  503.054 GB
   available :  196.485 GB
     percent :   60.900 %
        used :  289.186 GB
        free :  209.874 GB
      active :  267.938 GB
    inactive :    9.509 GB
     buffers :  354.637 MB
      cached :    3.648 GB
      shared :  184.684 MB
        slab :    1.870 GB

Cache now has 0 bytes
Memory usage After creation of 100 small buffers
       total :  503.054 GB
   available :  196.398 GB
     percent :   61.000 %
        used :  289.273 GB
```

```
        free :  209.787 GB
      active :  268.027 GB
    inactive :    9.509 GB
     buffers :  354.637 MB
      cached :    3.648 GB
      shared :  184.684 MB
        slab :    1.870 GB

Cache now has 100000000 bytes
Memory usage After cache clear
       total :  503.054 GB
   available :  196.398 GB
     percent :   61.000 %
        used :  289.273 GB
        free :  209.787 GB
      active :  268.027 GB
    inactive :    9.509 GB
     buffers :  354.637 MB
      cached :    3.648 GB
      shared :  184.684 MB
        slab :    1.870 GB

Cache now has 0 bytes


ok
test_put (toast.tests.cache.CacheTest) ... ok
test_put_none (toast.tests.cache.CacheTest) ... ok
test_refcount (toast.tests.cache.CacheTest) ... ok
test_comm (toast.tests.timing.TimingTest) ...

--- Rank 0
---
TOAST INFO: Global timer: timer1:  0.20 seconds (1 calls)
TOAST INFO: Global timer: timer2:  0.20 seconds (1 calls)
TOAST INFO: Global timer: timer3:  0.20 seconds (1 calls)
timer1 timing:
  call_max = 1
  call_mean = 1.0
  call_median = 1.0
  call_min = 1
  participating = 1
  time_max = 0.200092744
  time_mean = 0.200092744
  time_median = 0.200092744
  time_min = 0.200092744
timer2 timing:
  call_max = 1
```

```
  call_mean = 1.0
  call_median = 1.0
  call_min = 1
  participating = 1
  time_max = 0.200093448
  time_mean = 0.200093448
  time_median = 0.200093448
  time_min = 0.200093448
timer3 timing:
  call_max = 1
  call_mean = 1.0
  call_median = 1.0
  call_min = 1
  participating = 1
  time_max = 0.200093751
  time_mean = 0.200093751
  time_median = 0.200093751
  time_min = 0.200093751


ok
test_global (toast.tests.timing.TimingTest) ...

Successful exception:  stop an already stopped timer
Successful exception:  stop an already stopped timer
Successful exception:  stop an already stopped timer
Successful exception:  seconds() on running timerTOAST ERROR: Timer is still running! (file "/g

Successful exception:  seconds() on running timerTOAST ERROR: Timer is still running! (file "/g

Successful exception:  seconds() on running timer
TOAST ERROR: Timer is still running! (file "/global/homes/k/kisner/git/toast/src/libtoast/src/t


ok
test_single (toast.tests.timing.TimingTest) ...

TOAST INFO: Global timer: timer1:  0.20 seconds (1 calls)
TOAST INFO: Global timer: timer2:  0.40 seconds (1 calls)
TOAST INFO: Global timer: timer3:  0.60 seconds (1 calls)


ok

Successful exception:  elapsed_seconds() from a stopped timer
Successful exception:  report_elapsed() for a stopped timer
TOAST INFO: Test timer elapsed:  0.20 seconds (1 calls)
TOAST ERROR: Timer is not running! (file "/global/homes/k/kisner/git/toast/src/libtoast/src/toa
TOAST INFO: Test timer stopped:  0.20 seconds (1 calls)
```

```
Successful exception:  report running timer



test_rng_01 (toast.tests.rng.RNGTest) ... ok
test_rng_01_multi (toast.tests.rng.RNGTest) ... ok
test_rng_gaussian (toast.tests.rng.RNGTest) ... ok
test_rng_gaussian_multi (toast.tests.rng.RNGTest) ... ok
test_rng_m11 (toast.tests.rng.RNGTest) ... ok
test_rng_m11_multi (toast.tests.rng.RNGTest) ... ok
test_rng_uint64 (toast.tests.rng.RNGTest) ... ok
test_rng_uint64_multi (toast.tests.rng.RNGTest) ... ok
test_roundtrip (toast.tests.fft.FFTTest) ...

TOAST ERROR: Timer is still running! (file "/global/homes/k/kisner/git/toast/src/libtoast/src/t
TOAST INFO: original:  0.20 seconds (1 calls)
TOAST INFO: pickle roundtrip:  0.20 seconds (1 calls)


ok
test_construction (toast.tests.dist.DataTest) ... ok
test_none (toast.tests.dist.DataTest) ... ok
test_split (toast.tests.dist.DataTest) ... ok
test_angles (toast.tests.qarray.QarrayTest) ... ok
test_depths (toast.tests.qarray.QarrayTest) ... ok
test_exp (toast.tests.qarray.QarrayTest) ... ok
test_fp (toast.tests.qarray.QarrayTest) ... ok
test_fromrotmat (toast.tests.qarray.QarrayTest) ... ok
test_fromvectors (toast.tests.qarray.QarrayTest) ... ok
test_inv (toast.tests.qarray.QarrayTest) ... ok
test_ln (toast.tests.qarray.QarrayTest) ... ok
test_mult_onequaternion (toast.tests.qarray.QarrayTest) ... ok
test_mult_qarray (toast.tests.qarray.QarrayTest) ... ok
test_norm (toast.tests.qarray.QarrayTest) ... ok
test_pow (toast.tests.qarray.QarrayTest) ... ok
test_rotate_onequaternion (toast.tests.qarray.QarrayTest) ... ok
test_rotate_qarray (toast.tests.qarray.QarrayTest) ... ok
test_rotation (toast.tests.qarray.QarrayTest) ... ok
test_slerp (toast.tests.qarray.QarrayTest) ... ok
test_toaxisangle (toast.tests.qarray.QarrayTest) ... ok
test_torotmat (toast.tests.qarray.QarrayTest) ... ok
test_cached_read (toast.tests.tod.TODTest) ... ok
test_local_intervals (toast.tests.tod.TODTest) ... ok
test_local_signal (toast.tests.tod.TODTest) ... ok
test_props (toast.tests.tod.TODTest) ...

<TODCache
  4 total detectors and 10 total samples
  Using MPI communicator None
```

```
    In grid dimensions 1 sample ranks x 1 detranks
  Process at (0, 0) in grid has data for:
    Samples 0 - 9 (inclusive)
    Detectors:
      1a
      1b
      2a
      2b
    Cache contains 1730 bytes
>


ok
test_read (toast.tests.tod.TODTest) ... ok
test_read_pntg (toast.tests.tod.TODTest) ... ok
test_phase (toast.tests.tod_satellite.TODSatelliteTest) ... ok
test_precession (toast.tests.tod_satellite.TODSatelliteTest) ... ok
test_todclass (toast.tests.tod_satellite.TODSatelliteTest) ... ok
test_regular (toast.tests.intervals.IntervalTest) ... ok
test_tochunks (toast.tests.intervals.IntervalTest) ... ok
test_gauss (toast.tests.ops_simnoise.OpSimNoiseTest) ... ok
test_sim (toast.tests.ops_simnoise.OpSimNoiseTest) ... ok
test_sim_correlated (toast.tests.ops_simnoise.OpSimNoiseTest) ... ok
test_sss (toast.tests.ops_sim_sss.OpSimScanSynchronousSignalTest) ...

TOAST INFO: TODGround: simulate scan:  0.02 seconds (1 calls)
TOAST INFO: TODGround: list valid intervals:  0.00 seconds (1 calls)
TOAST INFO: TODGround: call base class constructor:  0.00 seconds (1 calls)
TOAST INFO: TODGround: translate scan pointing:  0.03 seconds (1 calls)
TOAST INFO: TODGround: simulate scan:  0.02 seconds (1 calls)
TOAST INFO: TODGround: list valid intervals:  0.00 seconds (1 calls)
TOAST INFO: TODGround: call base class constructor:  0.00 seconds (1 calls)
TOAST INFO: TODGround: translate scan pointing:  0.05 seconds (1 calls)
TOAST INFO: 0 : test-0-0 : Setting up SSS simulation
Sigma is 254.796540 arcmin (0.074117 rad)
-> fwhm is 600.000000 arcmin
Sigma is 0.000000 arcmin (0.000000 rad)
-> fwhm is 0.000000 arcmin
TOAST INFO: 0 : test-0-0 : Observing the scan-synchronous signal


ok
test_op_applygain (toast.tests.ops_applygain.TestApplyGain) ... ok
test_write_calibration_file (toast.tests.ops_applygain.TestApplyGain) ... ok
test_hpix_hwpnull (toast.tests.ops_pmat.OpPointingHpixTest) ... ok
test_hpix_simple (toast.tests.ops_pmat.OpPointingHpixTest) ... ok
test_pointing_matrix_healpix (toast.tests.ops_pmat.OpPointingHpixTest) ... ok
test_pointing_matrix_healpix2 (toast.tests.ops_pmat.OpPointingHpixTest) ... ok
```

```
test_accum (toast.tests.cov.CovarianceTest) ... ok
test_distpix_init (toast.tests.cov.CovarianceTest) ...

TOAST INFO: 0 : test-0-0 : Setting up SSS simulation
Sigma is 254.796540 arcmin (0.074117 rad)
-> fwhm is 600.000000 arcmin
Sigma is 0.000000 arcmin (0.000000 rad)
-> fwhm is 0.000000 arcmin
TOAST INFO: 0 : test-0-0 : Observing the scan-synchronous signal
TOAST INFO: Gains written to file /global/u2/k/khcheung/git/fork/toast-workshop-ucsd-2019/lesso


ok
test_fitsio (toast.tests.cov.CovarianceTest) ...

NSIDE = 32
ORDERING = NESTED in fits file
INDXSCHM = IMPLICIT
Ordering converted to RING
NSIDE = 32
ORDERING = NESTED in fits file
INDXSCHM = IMPLICIT
Ordering converted to RING
NSIDE = 32
ORDERING = NESTED in fits file
INDXSCHM = IMPLICIT
Ordering converted to RING
NSIDE = 32
ORDERING = NESTED in fits file
INDXSCHM = IMPLICIT
Ordering converted to RING


ok
test_invert (toast.tests.cov.CovarianceTest) ... ok
test_invnpp (toast.tests.cov.CovarianceTest) ... ok
test_multiply (toast.tests.cov.CovarianceTest) ... ok
test_dipole_func (toast.tests.ops_dipole.OpSimDipoleTest) ... ok
test_dipole_func_total (toast.tests.ops_dipole.OpSimDipoleTest) ... ok
test_sim (toast.tests.ops_dipole.OpSimDipoleTest) ...

NSIDE = 64
ORDERING = RING in fits file
INDXSCHM = IMPLICIT
NSIDE = 64
ORDERING = RING in fits file
INDXSCHM = IMPLICIT
```

```
ok
test_filter (toast.tests.ops_groundfilter.OpGroundFilterTest) ... ok
test_cart_quat (toast.tests.sim_focalplane.SimFocalplaneTest) ...

TOAST INFO: TODGround: simulate scan:  0.02 seconds (1 calls)
TOAST INFO: TODGround: list valid intervals:  0.00 seconds (1 calls)
TOAST INFO: TODGround: call base class constructor:  0.00 seconds (1 calls)
TOAST INFO: TODGround: translate scan pointing:  0.05 seconds (1 calls)


ok
test_hex_nring (toast.tests.sim_focalplane.SimFocalplaneTest) ... ok
test_vis_hex_large (toast.tests.sim_focalplane.SimFocalplaneTest) ... ok
test_vis_hex_medium (toast.tests.sim_focalplane.SimFocalplaneTest) ... ok
test_vis_hex_small (toast.tests.sim_focalplane.SimFocalplaneTest) ... ok
test_vis_hex_small_rad (toast.tests.sim_focalplane.SimFocalplaneTest) ... ok
test_vis_rhombus (toast.tests.sim_focalplane.SimFocalplaneTest) ... ok
test_filter (toast.tests.ops_polyfilter.OpPolyFilterTest) ... ok
test_counter (toast.tests.ops_memorycounter.OpMemoryCounterTest) ... ok
test_scrambler (toast.tests.ops_gainscrambler.OpGainScramblerTest) ... ok
test_autocov_psd (toast.tests.psd_math.PSDTest) ... ok
test_madam_gradient (toast.tests.ops_madam.OpMadamTest) ... ok
test_madam_output (toast.tests.ops_madam.OpMadamTest) ... ok
test_mapmaker_madam (toast.tests.ops_mapmaker.OpMapMakerTest) ...

libmadam not available, skipping tests
libmadam not available, skipping tests
TOAST INFO: Flag gaps:  0.00 seconds (1 calls)
TOAST INFO: Get detector weights:  0.00 seconds (1 calls)
TOAST INFO: Identify local submaps:  0.01 seconds (1 calls)
TOAST INFO: Accumulate N_pp'^1:  0.01 seconds (1 calls)
TOAST INFO: All reduce N_pp'^1:  0.00 seconds (1 calls)
TOAST INFO: Wrote hits to /global/u2/k/khcheung/git/fork/toast-workshop-ucsd-2019/lessons/01_In
TOAST INFO: Write hits:  0.01 seconds (1 calls)
TOAST INFO: Wrote inverse white noise covariance to /global/u2/k/khcheung/git/fork/toast-worksh
TOAST INFO: Write N_pp'^1:  0.02 seconds (1 calls)
TOAST INFO: Compute reciprocal condition numbers:  0.00 seconds (1 calls)
TOAST INFO: Wrote reciprocal condition numbers to /global/u2/k/khcheung/git/fork/toast-worksho
TOAST INFO: Write rcond:  0.01 seconds (1 calls)
TOAST INFO: Invert N_pp'^1:  0.01 seconds (1 calls)
TOAST INFO: Wrote white noise covariance to /global/u2/k/khcheung/git/fork/toast-workshop-ucsd-
TOAST INFO: Write N_pp':  0.02 seconds (1 calls)
TOAST INFO:   Build noise-weighted map:  0.00 seconds (0 calls)
TOAST INFO:   Apply noise covariance:  0.00 seconds (0 calls)
TOAST INFO:   Write map to /global/u2/k/khcheung/git/fork/toast-workshop-ucsd-2019/lessons/01_I
TOAST INFO: Read processing mask from /global/u2/k/khcheung/git/fork/toast-workshop-ucsd-2019/l
TOAST INFO: Apply processing mask:  0.01 seconds (1 calls)
TOAST INFO: Initializing offset template, step_length = 1
```

```
TOAST INFO: Initialize templates:  0.29 seconds (1 calls)
TOAST INFO: Initialize projection matrix:  0.00 seconds (1 calls)
TOAST INFO: Initialize projection matrix:  0.00 seconds (1 calls)
TOAST INFO: Initialize PCG solver:  0.04 seconds (1 calls)
TOAST INFO: Initial residual: 31025245359171.836
TOAST INFO: Iter =    0 relative residual:   1.5459e-02:  0.13 seconds (1 calls)
TOAST INFO: Iter =    1 relative residual:   2.2468e-03:  0.08 seconds (1 calls)
TOAST INFO: Iter =    2 relative residual:   6.5868e-04:  0.08 seconds (1 calls)
TOAST INFO: Iter =    3 relative residual:   8.8820e-05:  0.08 seconds (1 calls)
TOAST INFO: Iter =    4 relative residual:   4.6969e-05:  0.08 seconds (1 calls)
TOAST INFO: Iter =    5 relative residual:   2.0606e-05:  0.08 seconds (1 calls)
TOAST INFO: Iter =    6 relative residual:   1.1758e-05:  0.09 seconds (1 calls)
TOAST INFO: Iter =    7 relative residual:   5.7747e-06:  0.09 seconds (1 calls)
TOAST INFO: Iter =    8 relative residual:   2.6707e-06:  0.08 seconds (1 calls)
TOAST INFO: Iter =    9 relative residual:   1.9917e-06:  0.09 seconds (1 calls)
TOAST INFO: Iter =   10 relative residual:   1.4165e-06:  0.08 seconds (1 calls)
TOAST INFO: Iter =   11 relative residual:   7.7161e-07:  0.08 seconds (1 calls)
TOAST INFO: Iter =   12 relative residual:   6.2295e-07:  0.08 seconds (1 calls)
TOAST INFO: Iter =   13 relative residual:   4.3296e-07:  0.08 seconds (1 calls)
TOAST INFO: Iter =   14 relative residual:   2.8488e-07:  0.08 seconds (1 calls)
TOAST INFO: Iter =   15 relative residual:   1.6527e-07:  0.08 seconds (1 calls)
TOAST INFO: Iter =   16 relative residual:   1.0608e-07:  0.08 seconds (1 calls)
TOAST INFO: Iter =   17 relative residual:   5.8429e-08:  0.08 seconds (1 calls)
TOAST INFO: Iter =   18 relative residual:   3.0966e-08:  0.08 seconds (1 calls)
TOAST INFO: Iter =   19 relative residual:   2.0165e-08:  0.08 seconds (1 calls)
TOAST INFO: Iter =   20 relative residual:   1.1556e-08:  0.08 seconds (1 calls)
TOAST INFO: Iter =   21 relative residual:   8.0575e-09:  0.08 seconds (1 calls)
TOAST INFO: Iter =   22 relative residual:   5.4327e-09:  0.08 seconds (1 calls)
TOAST INFO: Iter =   23 relative residual:   3.7765e-09:  0.08 seconds (1 calls)
TOAST INFO: Iter =   24 relative residual:   2.5908e-09:  0.08 seconds (1 calls)
TOAST INFO: Iter =   25 relative residual:   1.6324e-09:  0.08 seconds (1 calls)
TOAST INFO: Iter =   26 relative residual:   1.1368e-09:  0.08 seconds (1 calls)
TOAST INFO: Iter =   27 relative residual:   7.4918e-10:  0.08 seconds (1 calls)
TOAST INFO: Iter =   28 relative residual:   5.8417e-10:  0.08 seconds (1 calls)
TOAST INFO: Iter =   29 relative residual:   4.1219e-10:  0.08 seconds (1 calls)
TOAST INFO: Iter =   30 relative residual:   2.8770e-10:  0.08 seconds (1 calls)
TOAST INFO: Iter =   31 relative residual:   2.1173e-10:  0.08 seconds (1 calls)
TOAST INFO: Iter =   32 relative residual:   1.3643e-10:  0.08 seconds (1 calls)
TOAST INFO: Iter =   33 relative residual:   1.0149e-10:  0.08 seconds (1 calls)
TOAST INFO: Iter =   34 relative residual:   8.2024e-11:  0.08 seconds (1 calls)
TOAST INFO: Iter =   35 relative residual:   6.3968e-11:  0.08 seconds (1 calls)
TOAST INFO: Iter =   36 relative residual:   6.1209e-11:  0.08 seconds (1 calls)
TOAST INFO: Iter =   37 relative residual:   5.1889e-11:  0.08 seconds (1 calls)
TOAST INFO: Iter =   38 relative residual:   5.1683e-11:  0.09 seconds (1 calls)
TOAST INFO: Iter =   39 relative residual:   6.6672e-11:  0.09 seconds (1 calls)
TOAST INFO: Iter =   40 relative residual:   8.4721e-11:  0.09 seconds (1 calls)
TOAST INFO: Iter =   41 relative residual:   1.2548e-10:  0.09 seconds (1 calls)
TOAST INFO: Iter =   42 relative residual:   1.9966e-10:  0.09 seconds (1 calls)
```

```
TOAST INFO: Iter =   43 relative residual:   3.0122e-10:  0.09 seconds (1 calls)
TOAST INFO: Iter =   44 relative residual:   4.4938e-10:  0.09 seconds (1 calls)
TOAST INFO: Iter =   45 relative residual:   6.5880e-10:  0.09 seconds (1 calls)
TOAST INFO: Iter =   46 relative residual:   1.0454e-09:  0.09 seconds (1 calls)
TOAST INFO: Iter =   47 relative residual:   1.5247e-09:  0.10 seconds (1 calls)
TOAST INFO: Iter =   48 relative residual:   2.1989e-09:  0.09 seconds (1 calls)
TOAST INFO: Iter =   49 relative residual:   3.4409e-09:  0.09 seconds (1 calls)
TOAST INFO: Iter =   50 relative residual:   4.3654e-09:  0.09 seconds (1 calls)
TOAST INFO: PCG stalled after 50 iterations:  4.32 seconds (1 calls)
TOAST INFO: 0 : Solution: template amplitudes:
"offset" :
[ -6.75727997  -6.3120565   -5.55929247 ... -29.49541514 -30.16792006
 -13.49735431]
TOAST INFO: Solve amplitudes:  4.32 seconds (1 calls)
TOAST INFO: Clean TOD:  0.01 seconds (1 calls)
TOAST INFO:   Build noise-weighted map:  0.00 seconds (0 calls)
TOAST INFO:   Apply noise covariance:  0.00 seconds (0 calls)
all_timers:TOAST INFO:   Write map to /global/u2/k/khcheung/git/fork/toast-workshop-ucsd-2019/l
 {'cov_accum_diag': {'participating': 1, 'call_min': 2, 'call_max': 2, 'call_mean': 2.0, 'call
                                                - OpAccumDiag.exec.apply_flags...     0.2
                                                - OpAccumDiag.exec.global_to_local...     0.3
                                                - cov_accum_zmap...     0.3
                                                - OpSimScan.exec.global_to_local...     0.3
                                                - OpSimScan.exec.scan_map...     0.5



ok
test_boresight_null (toast.tests.map_satellite.MapSatelliteTest) ...

Saved timers in /global/u2/k/khcheung/git/fork/toast-workshop-ucsd-2019/lessons/01_Introductio
libmadam not available, skipping mapmaker comparison


ok
test_grad (toast.tests.map_satellite.MapSatelliteTest) ... ok
test_hwpconst (toast.tests.map_satellite.MapSatelliteTest) ...

libmadam not available, skipping tests


ok
test_hwpfast (toast.tests.map_satellite.MapSatelliteTest) ...

libmadam not available, skipping tests


ok
test_noise (toast.tests.map_satellite.MapSatelliteTest) ...
```

```
libmadam not available, skipping tests


ok
test_scanmap (toast.tests.map_satellite.MapSatelliteTest) ...

libmadam not available, skipping tests


ok
test_azel (toast.tests.map_ground.MapGroundTest) ...

libmadam not available, skipping tests
TOAST INFO: TODGround: simulate scan:  0.02 seconds (1 calls)
TOAST INFO: TODGround: list valid intervals:  0.00 seconds (1 calls)
TOAST INFO: TODGround: call base class constructor:  0.00 seconds (1 calls)
TOAST INFO: TODGround: translate scan pointing:  0.05 seconds (1 calls)
TOAST INFO: TODGround: simulate scan:  0.02 seconds (1 calls)
TOAST INFO: TODGround: list valid intervals:  0.00 seconds (1 calls)
TOAST INFO: TODGround: call base class constructor:  0.00 seconds (1 calls)
TOAST INFO: TODGround: translate scan pointing:  0.05 seconds (1 calls)
TOAST INFO: TODGround: simulate scan:  0.02 seconds (1 calls)
TOAST INFO: TODGround: list valid intervals:  0.00 seconds (1 calls)
TOAST INFO: TODGround: call base class constructor:  0.00 seconds (1 calls)


ok
test_grad (toast.tests.map_ground.MapGroundTest) ...

TOAST INFO: TODGround: translate scan pointing:  0.05 seconds (1 calls)
TOAST INFO: TODGround: simulate scan:  0.02 seconds (1 calls)
TOAST INFO: TODGround: list valid intervals:  0.00 seconds (1 calls)
TOAST INFO: TODGround: call base class constructor:  0.00 seconds (1 calls)
TOAST INFO: TODGround: translate scan pointing:  0.05 seconds (1 calls)
TOAST INFO: TODGround: simulate scan:  0.02 seconds (1 calls)
TOAST INFO: TODGround: list valid intervals:  0.00 seconds (1 calls)
TOAST INFO: TODGround: call base class constructor:  0.00 seconds (1 calls)
TOAST INFO: TODGround: translate scan pointing:  0.05 seconds (1 calls)
TOAST INFO: TODGround: simulate scan:  0.02 seconds (1 calls)
TOAST INFO: TODGround: list valid intervals:  0.00 seconds (1 calls)
TOAST INFO: TODGround: call base class constructor:  0.00 seconds (1 calls)


ok
test_hwpconst (toast.tests.map_ground.MapGroundTest) ...

TOAST INFO: TODGround: translate scan pointing:  0.05 seconds (1 calls)
libmadam not available, skipping tests
TOAST INFO: TODGround: simulate scan:  0.02 seconds (1 calls)
```

```
TOAST INFO: TODGround: list valid intervals:  0.00 seconds (1 calls)
TOAST INFO: TODGround: call base class constructor:  0.00 seconds (1 calls)
TOAST INFO: TODGround: translate scan pointing:  0.05 seconds (1 calls)
TOAST INFO: TODGround: simulate scan:  0.03 seconds (1 calls)
TOAST INFO: TODGround: list valid intervals:  0.00 seconds (1 calls)
TOAST INFO: TODGround: call base class constructor:  0.00 seconds (1 calls)


ok
test_hwpfast (toast.tests.map_ground.MapGroundTest) ...
```

```
TOAST INFO: TODGround: translate scan pointing:  0.05 seconds (1 calls)
TOAST INFO: TODGround: simulate scan:  0.02 seconds (1 calls)
TOAST INFO: TODGround: list valid intervals:  0.00 seconds (1 calls)
TOAST INFO: TODGround: call base class constructor:  0.00 seconds (1 calls)
TOAST INFO: TODGround: translate scan pointing:  0.05 seconds (1 calls)
libmadam not available, skipping tests
TOAST INFO: TODGround: simulate scan:  0.02 seconds (1 calls)
TOAST INFO: TODGround: list valid intervals:  0.00 seconds (1 calls)
TOAST INFO: TODGround: call base class constructor:  0.00 seconds (1 calls)
TOAST INFO: TODGround: translate scan pointing:  0.05 seconds (1 calls)
TOAST INFO: TODGround: simulate scan:  0.02 seconds (1 calls)
TOAST INFO: TODGround: list valid intervals:  0.00 seconds (1 calls)
TOAST INFO: TODGround: call base class constructor:  0.00 seconds (1 calls)
TOAST INFO: TODGround: translate scan pointing:  0.05 seconds (1 calls)
TOAST INFO: TODGround: simulate scan:  0.02 seconds (1 calls)
TOAST INFO: TODGround: list valid intervals:  0.00 seconds (1 calls)
TOAST INFO: TODGround: call base class constructor:  0.00 seconds (1 calls)
TOAST INFO: TODGround: translate scan pointing:  0.05 seconds (1 calls)


ok
test_noise (toast.tests.map_ground.MapGroundTest) ...
```

```
libmadam not available, skipping tests
TOAST INFO: TODGround: simulate scan:  0.02 seconds (1 calls)
TOAST INFO: TODGround: list valid intervals:  0.00 seconds (1 calls)
TOAST INFO: TODGround: call base class constructor:  0.00 seconds (1 calls)
TOAST INFO: TODGround: translate scan pointing:  0.05 seconds (1 calls)
TOAST INFO: TODGround: simulate scan:  0.02 seconds (1 calls)
TOAST INFO: TODGround: list valid intervals:  0.00 seconds (1 calls)
TOAST INFO: TODGround: call base class constructor:  0.00 seconds (1 calls)
TOAST INFO: TODGround: translate scan pointing:  0.05 seconds (1 calls)
TOAST INFO: TODGround: simulate scan:  0.02 seconds (1 calls)
TOAST INFO: TODGround: list valid intervals:  0.00 seconds (1 calls)
TOAST INFO: TODGround: call base class constructor:  0.00 seconds (1 calls)


ok
test_scanmap (toast.tests.map_ground.MapGroundTest) ...
```

```
TOAST INFO: TODGround: translate scan pointing:  0.05 seconds (1 calls)
libmadam not available, skipping tests
TOAST INFO: TODGround: simulate scan:  0.02 seconds (1 calls)
TOAST INFO: TODGround: list valid intervals:  0.00 seconds (1 calls)
TOAST INFO: TODGround: call base class constructor:  0.00 seconds (1 calls)
TOAST INFO: TODGround: translate scan pointing:  0.05 seconds (1 calls)
TOAST INFO: TODGround: simulate scan:  0.02 seconds (1 calls)
TOAST INFO: TODGround: list valid intervals:  0.00 seconds (1 calls)
TOAST INFO: TODGround: call base class constructor:  0.00 seconds (1 calls)
TOAST INFO: TODGround: translate scan pointing:  0.05 seconds (1 calls)


ok
test_binned (toast.tests.binned.BinnedTest) ...

TOAST INFO: TODGround: simulate scan:  0.02 seconds (1 calls)
TOAST INFO: TODGround: list valid intervals:  0.00 seconds (1 calls)
TOAST INFO: TODGround: call base class constructor:  0.00 seconds (1 calls)
TOAST INFO: TODGround: translate scan pointing:  0.05 seconds (1 calls)
libmadam not available, skipping tests
libmadam not available, skipping tests


ok
test_op_pysm_nosmooth (toast.tests.ops_sim_pysm.OpSimPySMTest) ...

TOAST DEBUG: Collecting, Broadcasting map
TOAST DEBUG: Running PySM on fake_0A


ok
test_pysm_local_pix (toast.tests.ops_sim_pysm.OpSimPySMTest) ...

TOAST DEBUG: Assemble PySM map on rank0, shape of local map is (3, 768)
TOAST DEBUG: Communication completed
TOAST DEBUG: PySM map min / max pixel value = -97.8028335571289 / 28006.533203125
TOAST DEBUG: Broadcasting the map to other processes
TOAST DEBUG: Running OpSimScan
TOAST DEBUG: Rank 0 timeline min / max after smoothing = 40.63530136412361 / 28131.66487772626
TOAST INFO: PySM Operator:  7.38 seconds (1 calls)


ok
test_pysm_ring_distribution (toast.tests.ops_sim_pysm.OpSimPySMTest) ... ok
test_op_pysm_smooth (toast.tests.ops_sim_pysm.OpSimPySMTestSmooth) ...

TOAST DEBUG: Collecting, Broadcasting map
TOAST DEBUG: Running PySM on fake_0A
TOAST DEBUG: Executing Smoothing with libsharp on fake_0A
```

```
ok
test_atm (toast.tests.ops_sim_atm.OpsSimAtmosphereTest) ...

TOAST DEBUG: Smoothing completed on fake_0A
TOAST DEBUG: Assemble PySM map on rank0, shape of local map is (3, 49152)
TOAST DEBUG: Communication completed
TOAST DEBUG: PySM map min / max pixel value = -77.15636998906243 / 15322.48714828821
TOAST DEBUG: Broadcasting the map to other processes
TOAST DEBUG: Running OpSimScan
TOAST DEBUG: Rank 0 timeline min / max after smoothing = 49.696532725306845 / 15434.553192717955
TOAST INFO: PySM Operator:  0.55 seconds (1 calls)
TOAST INFO: TODGround: simulate scan:  0.00 seconds (1 calls)
TOAST INFO: TODGround: list valid intervals:  0.00 seconds (1 calls)
TOAST INFO: TODGround: call base class constructor:  0.00 seconds (1 calls)
TOAST INFO: TODGround: translate scan pointing:  0.00 seconds (1 calls)
TOAST INFO: TODGround: simulate scan:  0.00 seconds (1 calls)
TOAST INFO: TODGround: list valid intervals:  0.00 seconds (1 calls)
TOAST INFO: TODGround: call base class constructor:  0.00 seconds (1 calls)
TOAST INFO: TODGround: translate scan pointing:  0.00 seconds (1 calls)


ok
test_atm_caching (toast.tests.ops_sim_atm.OpsSimAtmosphereTest) ...

No MPI available, skipping MPI/serial test
TOAST INFO: TODGround: simulate scan:  0.00 seconds (1 calls)
TOAST INFO: TODGround: list valid intervals:  0.00 seconds (1 calls)
TOAST INFO: TODGround: call base class constructor:  0.00 seconds (1 calls)
TOAST INFO: TODGround: translate scan pointing:  0.00 seconds (1 calls)
TOAST INFO: TODGround: simulate scan:  0.00 seconds (1 calls)
TOAST INFO: TODGround: list valid intervals:  0.00 seconds (1 calls)
TOAST INFO: TODGround: call base class constructor:  0.00 seconds (1 calls)
TOAST INFO: TODGround: translate scan pointing:  0.00 seconds (1 calls)
Creating /global/u2/k/khcheung/git/fork/toast-workshop-ucsd-2019/lessons/01_Introduction/toast_
TOAST INFO: 0 : test-0-0 : Setting up atmosphere simulation
TOAST INFO: 0 : test-0-0 : Instantiating atmosphere for t = 0.0
TOAST INFO: 0 : test-0-0 : OpSimAtmosphere: Initialize atmosphere:  0.00 seconds (1 calls)
TOAST INFO: 0 : test-0-0 : Simulating the atmosphere for t = 0.0
TOAST INFO: Volume compressed in:  0.00 seconds (1 calls)
140 / 144(97.2222 %) volume elements are needed for the simulation
nx = 4 ny = 6 nz = 6
wx = -1.84353 wy = 3.59223 wz = -3.19308
TOAST INFO: 0 : test-0-0 : OpSimAtmosphere: Simulated atmosphere:  1.56 seconds (1 calls)
TOAST INFO: 0 : test-0-0 : Observing the atmosphere
TOAST INFO: 0 : test-0-0 : OpSimAtmosphere: Observe atmosphere:  0.00 seconds (1 calls)
TOAST INFO: 0 : test-0-0 : OpSimAtmosphere: Initialize atmosphere:  0.00 seconds (1 calls)
TOAST INFO: 0 : test-0-0 : Simulating the atmosphere for t = 0.0
TOAST INFO: Volume compressed in:  0.00 seconds (1 calls)
```

```
80 / 80(100 %) volume elements are needed for the simulation
nx = 5 ny = 4 nz = 4
wx = -1.84353 wy = 3.59223 wz = -3.19308
TOAST INFO: 0 : test-0-0 : OpSimAtmosphere: Simulated atmosphere:  1.58 seconds (1 calls)
TOAST INFO: 0 : test-0-0 : Observing the atmosphere
TOAST INFO: 0 : test-0-0 : OpSimAtmosphere: Observe atmosphere:  0.00 seconds (1 calls)
TOAST INFO: 0 : test-0-0 : OpSimAtmosphere: Initialize atmosphere:  0.00 seconds (1 calls)
TOAST INFO: 0 : test-0-0 : Simulating the atmosphere for t = 0.0
TOAST INFO: Volume compressed in:  0.00 seconds (1 calls)
36 / 36(100 %) volume elements are needed for the simulation
nx = 4 ny = 3 nz = 3
wx = -1.84353 wy = 3.59223 wz = -3.19308
TOAST INFO: 0 : test-0-0 : OpSimAtmosphere: Simulated atmosphere:  1.66 seconds (1 calls)
TOAST INFO: 0 : test-0-0 : Observing the atmosphere
Creating /global/u2/k/khcheung/git/fork/toast-workshop-ucsd-2019/lessons/01_Introduction/toast_
TOAST INFO: 0 : test-0-0 : Simulated and observed atmosphere:  4.81 seconds (1 calls)



ok

----------------------------------------------------------------------
Ran 107 tests in 136.378s

OK


TOAST INFO: 0 : test-0-0 : Setting up atmosphere simulation
TOAST INFO: 0 : test-0-0 : Instantiating atmosphere for t = 0.0
TOAST INFO: 0 : test-0-0 : OpSimAtmosphere: Initialize atmosphere:  0.00 seconds (1 calls)
TOAST INFO: 0 : test-0-0 : Loading the atmosphere for t = 0.0 from /global/u2/k/khcheung/git/fo
TOAST INFO: 0 : test-0-0 : OpSimAtmosphere: Loaded atmosphere:  0.00 seconds (1 calls)
TOAST INFO: 0 : test-0-0 : Observing the atmosphere
TOAST INFO: 0 : test-0-0 : OpSimAtmosphere: Observe atmosphere:  0.00 seconds (1 calls)
TOAST INFO: 0 : test-0-0 : OpSimAtmosphere: Initialize atmosphere:  0.00 seconds (1 calls)
TOAST INFO: 0 : test-0-0 : Loading the atmosphere for t = 0.0 from /global/u2/k/khcheung/git/fo
TOAST INFO: 0 : test-0-0 : OpSimAtmosphere: Loaded atmosphere:  0.00 seconds (1 calls)
TOAST INFO: 0 : test-0-0 : Observing the atmosphere
TOAST INFO: 0 : test-0-0 : OpSimAtmosphere: Observe atmosphere:  0.00 seconds (1 calls)
TOAST INFO: 0 : test-0-0 : OpSimAtmosphere: Initialize atmosphere:  0.00 seconds (1 calls)
TOAST INFO: 0 : test-0-0 : Loading the atmosphere for t = 0.0 from /global/u2/k/khcheung/git/fo
TOAST INFO: 0 : test-0-0 : OpSimAtmosphere: Loaded atmosphere:  0.00 seconds (1 calls)
TOAST INFO: 0 : test-0-0 : Observing the atmosphere
TOAST INFO: 0 : test-0-0 : OpSimAtmosphere: Observe atmosphere:  0.00 seconds (1 calls)
TOAST INFO: 0 : test-0-0 : Simulated and observed atmosphere:  0.00 seconds (1 calls)
```

```
Out[37]: 0
```

intro_mpi

November 1, 2019

1 Introduction - MPI Example

This lesson is a brief introduction to TOAST and its data representations. This next cell is just initializing some things for the notebook.

In [ ]:
```
# Are you using a special reservation for a workshop?
# If so, set it here:
nersc_reservation = None

# Load common tools for all lessons
import sys
sys.path.insert(0, "..")
from lesson_tools import (
    check_nersc
)
nersc_host, nersc_repo, nersc_resv = check_nersc(reservation=nersc_reservation)

# Capture C++ output in the jupyter cells
%reload_ext wurlitzer
```

In [ ]:
```
%%writefile intro_mpi.py

import toast
from toast.mpi import MPI

# Load common tools for all lessons
import sys
sys.path.insert(0, "..")
from lesson_tools import (
    fake_focalplane
)

import numpy as np
import matplotlib.pyplot as plt

import toast.qarray as qa
```

```
# Runtime Environment

env = toast.Environment.get()
if MPI.COMM_WORLD.rank == 0:
    print(env, flush=True)

# Create a fake focalplane

fp = fake_focalplane()

detnames = list(sorted(fp.keys()))
detquat = {x: fp[x]["quat"] for x in detnames}
detfwhm = {x: fp[x]["fwhm_arcmin"] for x in detnames}
detlabels = {x: x for x in detnames}
detpolcol = {x: "red" if i % 2 == 0 else "blue" for i, x in enumerate(detnames)}

if MPI.COMM_WORLD.rank == 0:
    outfile = "intro_focalplane_mpi.pdf"
    toast.tod.plot_focalplane(
        detquat, 4.0, 4.0, outfile, fwhm=detfwhm, polcolor=detpolcol, labels=detlabels
    )

# Different communicators

# Default communicator (one group using COMM_WORLD)

comm = toast.Comm()
if comm.world_rank == 0:
    print("----- Default Comm -----")
for p in range(MPI.COMM_WORLD.size):
    if p == comm.world_rank:
        print(comm, flush=True)
    comm.comm_world.barrier()

# Communicator with group size of one.

comm = toast.Comm(world=MPI.COMM_WORLD, groupsize=1)
if comm.world_rank == 0:
    print("----- Comm: groupsize = 1 -----")
for p in range(MPI.COMM_WORLD.size):
    if p == comm.world_rank:
        print(comm, flush=True)
    comm.comm_world.barrier()

# Communicator with 2 groups

gsize = MPI.COMM_WORLD.size // 2
```

```
    comm = toast.Comm(world=MPI.COMM_WORLD, groupsize=gsize)
    if comm.world_rank == 0:
        print("----- Comm: groupsize = {} -----".format(gsize))
    for p in range(MPI.COMM_WORLD.size):
        if p == comm.world_rank:
            print(comm, flush=True)
        comm.comm_world.barrier()
```

In [ ]:
```
import subprocess as sp

command = "python intro_mpi.py"
runstr = None

if nersc_host is not None:
    runstr = "srun -N 1 -C haswell -n 32 -c 2 --cpu_bind=cores -t 00:03:00"
    if nersc_resv is not None:
        runstr = "{} --reservation {}".format(runstr, nersc_resv)
else:
    # Just use mpirun
    runstr = "mpirun -np 4"

runcom = "{} {}".format(runstr, command)
print(runcom, flush=True)
sp.check_call(runcom, stderr=sp.STDOUT, shell=True)
```

In [ ]:

simscan_ground

November 1, 2019

1 Ground observing schedules

In this notebook we learn about creating ground observing schedules.

```
In [1]: # Are you using a special reservation for a workshop?
        # If so, set it here:
        nersc_reservation = None

        # Load common tools for all lessons
        import sys
        sys.path.insert(0, "..")
        from lesson_tools import (
            check_nersc,
            fake_focalplane
        )
        nersc_host, nersc_repo, nersc_resv = check_nersc(reservation=nersc_reservation)

        # Capture C++ output in the jupyter cells
        %reload_ext wurlitzer
```

```
Running on NERSC machine 'cori'
  with access to repos: mp107
Using default repo mp107
```

TOAST `pipelines` include a tool called `toast_ground_schedule.py`, also known as the *opportunistic scheduler*. It builds observing schedules heuristically by building a list of available targets and scheduling and always choosing the highest priority target. `toast_ground_schedule.py` can be used to create site-specific observing schedules subject to a number of constraints. At the minimum, the tool needs the location of the observatory, observing window and at least one target. Here is a minimal example:

```
In [2]: ! toast_ground_schedule.py \
            --site-lat "-22.958064" \
            --site-lon "-67.786222" \
            --site-alt 5200 \
            --site-name Atacama \
```

```
            --telescope LAT \
            --start "2020-01-01 00:00:00" \
            --stop "2020-01-01 12:00:00" \
            --patch-coord C \
            --patch small_patch,1,40,-40,44,-44 \
            --out schedule.txt

TOAST INFO: Adding patch "small_patch"
TOAST INFO: Rectangular format
TOAST INFO: Global timer: toast_ground_schedule:  0.16 seconds (1 calls)
```

Let's look at the contents of the schedule file.

```
In [3]: ! cat schedule.txt
```

```
#Site            Telescope        Latitude [deg] Longitude [deg]   Elevation [m]
 Atacama         LAT                      -22.958         -67.786          5200.0
#Start time UTC       Stop time UTC        Start MJD      Stop MJD       Patch name
 2020-01-01 00:50:00  2020-01-01 01:04:45    58849.034722   58849.044965 small_patch
 2020-01-01 01:04:55  2020-01-01 01:19:40    58849.045081   58849.055324 small_patch
 2020-01-01 01:19:50  2020-01-01 01:34:35    58849.055440   58849.065683 small_patch
 2020-01-01 01:34:45  2020-01-01 01:49:00    58849.065799   58849.075694 small_patch
 2020-01-01 01:50:40  2020-01-01 02:02:00    58849.076852   58849.084722 small_patch
 2020-01-01 02:02:10  2020-01-01 02:13:30    58849.084838   58849.092708 small_patch
 2020-01-01 02:13:40  2020-01-01 02:24:40    58849.092824   58849.100463 small_patch
 2020-01-01 02:26:20  2020-01-01 02:40:20    58849.101620   58849.111343 small_patch
 2020-01-01 02:40:30  2020-01-01 02:54:20    58849.111458   58849.121065 small_patch
 2020-01-01 02:56:00  2020-01-01 03:08:00    58849.122222   58849.130556 small_patch
 2020-01-01 03:08:10  2020-01-01 03:20:00    58849.130671   58849.138889 small_patch
 2020-01-01 03:21:40  2020-01-01 03:32:40    58849.140046   58849.147685 small_patch
 2020-01-01 03:32:50  2020-01-01 03:43:40    58849.147801   58849.155324 small_patch
 2020-01-01 03:45:20  2020-01-01 03:55:20    58849.156481   58849.163426 small_patch
 2020-01-01 03:55:30  2020-01-01 04:05:20    58849.163542   58849.170370 small_patch
 2020-01-01 04:07:00  2020-01-01 04:16:00    58849.171528   58849.177778 small_patch
 2020-01-01 04:16:10  2020-01-01 04:25:00    58849.177894   58849.184028 small_patch
 2020-01-01 04:26:40  2020-01-01 04:35:10    58849.185185   58849.191088 small_patch
 2020-01-01 04:35:20  2020-01-01 04:43:40    58849.191204   58849.196991 small_patch
 2020-01-01 04:45:20  2020-01-01 04:53:50    58849.198148   58849.204051 small_patch
 2020-01-01 04:54:00  2020-01-01 05:02:20    58849.204167   58849.209954 small_patch
 2020-01-01 05:04:00  2020-01-01 05:13:00    58849.211111   58849.217361 small_patch
 2020-01-01 05:13:10  2020-01-01 05:22:00    58849.217477   58849.223611 small_patch
```

The rectangular patch definition takes the form `--patch <name>,<priority>,<RA left>,<DEC top>,<RA right>,<DEC bottom>`. No spaces are allowed in the definition. Other patch definition formats will be discussed below.

The start and stop times are given in UTC.

The resulting schedule is a plain ASCII file. The header defines the telescope and each line after that defines a constant elevation scan (CES) with a fixed azimuth range. When a full pass of the target takes longer than allowed observation time, `--ces-max-time`, the CES is broken up into sub passes that use the same observing elevation but adjust the azimuth range. The above schedule includes 10 passes of the target "small_patch" that fit in the given 12-hour observing window. Some passes are split into as many as 4 sub passes, each no longer than 20 minutes (default).

Let's add another patch, this time using the circular patch definition format, set the observing elevation limits and enable Sun avoidance. We'll also increase `ces-max-time` so we get fewer entries in the schedule. The circular patch format is

```
--patch <name>,<priority>,<RA>,<DEC>,<radius>
```

```
In [4]: ! toast_ground_schedule.py \
            --site-lat "-22.958064" \
            --site-lon "-67.786222" \
            --site-alt 5200 \
            --site-name Atacama \
            --telescope LAT \
            --start "2020-01-01 00:00:00" \
            --stop "2020-01-04 00:00:00" \
            --patch-coord C \
            --patch small_patch,1,80,-13,10 \
            --patch large_patch,1,80,-33,20 \
            --el-min 30 \
            --el-max 60 \
            --ces-max-time 86400 \
            --sun-avoidance-angle 20 \
            --out schedule.txt \
            --debug

        ! cat schedule.txt
```

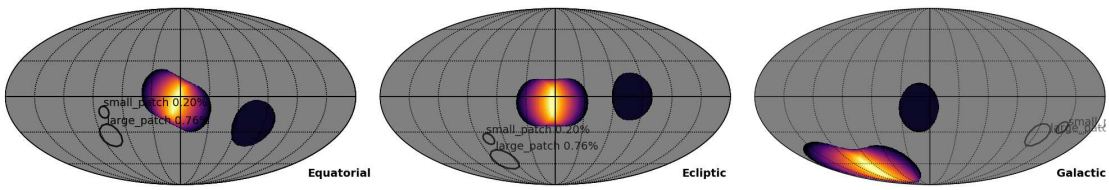
```
TOAST INFO: Adding patch "small_patch"
TOAST INFO: Center-and-width format
TOAST INFO: Adding patch "large_patch"
TOAST INFO: Center-and-width format
TOAST INFO: small_patch corners:
 lon = [80.0, 82.52889698392441, 84.40387023305097, 85.13152053896695, 84.49355559956005, 82.6
 lat= [-8.000000000000002, -8.669872981077807, -10.5, -13.0, -15.5, -17.33012701892219, -18.0,
TOAST INFO: large_patch corners:
 lon = [80.0, 85.48776585034872, 89.80834435462137, 91.92363292835948, 90.99002012118036, 86.69
 lat= [-23.0, -24.339745962155618, -28.0, -33.0, -38.0, -41.66025403784439, -43.0, -41.6602540
TOAST INFO: small_patch corners:
 lon = [80.0, 82.52889698392441, 84.40387023305097, 85.13152053896695, 84.49355559956005, 82.6
 lat= [-8.000000000000002, -8.669872981077807, -10.5, -13.0, -15.5, -17.33012701892219, -18.0,
TOAST INFO: large_patch corners:
 lon = [80.0, 85.48776585034872, 89.80834435462137, 91.92363292835948, 90.99002012118036, 86.69
 lat= [-23.0, -24.339745962155618, -28.0, -33.0, -38.0, -41.66025403784439, -43.0, -41.6602540
TOAST INFO: small_patch corners:
```

```
 lon = [80.0, 82.52889698392441, 84.40387023305097, 85.13152053896695, 84.49355559956005, 82.6
 lat= [-8.000000000000002, -8.669872981077807, -10.5, -13.0, -15.5, -17.33012701892219, -18.0,
TOAST INFO: large_patch corners:
 lon = [80.0, 85.48776585034872, 89.80834435462137, 91.92363292835948, 90.99002012118036, 86.69
 lat= [-23.0, -24.339745962155618, -28.0, -33.0, -38.0, -41.66025403784439, -43.0, -41.6602540
TOAST INFO: Global timer: toast_ground_schedule:  9.73 seconds (1 calls)
#Site            Telescope        Latitude [deg] Longitude [deg]   Elevation [m]
 Atacama         LAT                      -22.958         -67.786          5200.0
#Start time UTC       Stop time UTC        Start MJD      Stop MJD       Patch name
 2020-01-01 00:00:00  2020-01-01 00:44:00    58849.000000   58849.030556 small_patch
 2020-01-01 00:45:40  2020-01-01 01:30:40    58849.031713   58849.062963 small_patch
 2020-01-01 04:32:20  2020-01-01 06:11:20    58849.189120   58849.257870 large_patch
 2020-01-01 06:13:00  2020-01-01 06:56:00    58849.259028   58849.288889 small_patch
 2020-01-01 06:57:40  2020-01-01 07:40:40    58849.290046   58849.319907 small_patch
 2020-01-01 21:52:20  2020-01-01 23:28:20    58849.911343   58849.978009 large_patch
 2020-01-01 23:30:00  2020-01-02 01:06:00    58849.979167   58850.045833 large_patch
 2020-01-02 04:27:40  2020-01-02 06:06:40    58850.185880   58850.254630 large_patch
 2020-01-02 06:08:20  2020-01-02 06:51:20    58850.255787   58850.285648 small_patch
 2020-01-02 06:53:00  2020-01-02 07:36:00    58850.286806   58850.316667 small_patch
 2020-01-02 21:47:40  2020-01-02 23:23:40    58850.908102   58850.974769 large_patch
 2020-01-02 23:25:20  2020-01-03 00:08:20    58850.975926   58851.005787 small_patch
 2020-01-03 00:10:00  2020-01-03 00:54:00    58851.006944   58851.037500 small_patch
 2020-01-03 04:25:40  2020-01-03 06:03:40    58851.184491   58851.252546 large_patch
 2020-01-03 06:05:20  2020-01-03 07:41:20    58851.253704   58851.320370 large_patch
 2020-01-03 21:43:00  2020-01-03 23:19:00    58851.904861   58851.971528 large_patch
```

Note that we added the `--debug` option to the command line. This produces a helpful diagnostic plot, `patches.png`, that shows the locations of your patches, the Sun, the Moon and their avoidance areas. The plot is shown below. The motion of the Moon is already apparent in this 3-day schedule. The Sun (on the right) is effectively stationary. `--debug` can be expensive, especially if you have lots of patches or request a long observing schedule.

```
In [5]: from IPython.display import Image
        Image("patches.png")
```

Out[5]:



We deliberately chose the locations of the patches so that they compete over the observing time. This allows us to point out some advanced features of the scheduler. If you examine the

very end of the observing schedule, you can note that both `small_patch` and `large_patch` were observed 7 times. Given that `large_patch` is twice as wide and only takes twice as long to observe, equal number of observations actually implies that `large_patch` will end up with *half* as many hits per sky pixel.

The scheduler offers two ways to remedy this issue. First, one can simply increase the priority of the large patch to dedicate more observing time to it. **All things being equal, the number of visits to a given patch is inversely proportional to the `priority` in the patch definition**:

```
In [6]: ! toast_ground_schedule.py \
            --site-lat "-22.958064" \
            --site-lon "-67.786222" \
            --site-alt 5200 \
            --site-name Atacama \
            --telescope LAT \
            --start "2020-01-01 00:00:00" \
            --stop "2020-01-04 00:00:00" \
            --patch-coord C \
            --patch small_patch,1,80,-13,10 \
            --patch large_patch,0.5,80,-33,20 \
            --el-min 30 \
            --el-max 60 \
            --ces-max-time 86400 \
            --sun-avoidance-angle 20 \
            --out schedule.txt

        ! cat schedule.txt

TOAST INFO: Adding patch "small_patch"
TOAST INFO: Center-and-width format
TOAST INFO: Adding patch "large_patch"
TOAST INFO: Center-and-width format
TOAST INFO: Global timer: toast_ground_schedule:  0.62 seconds (1 calls)
#Site            Telescope        Latitude [deg] Longitude [deg]   Elevation [m]
 Atacama         LAT                      -22.958         -67.786          5200.0
#Start time UTC       Stop time UTC        Start MJD      Stop MJD       Patch name
 2020-01-01 00:00:00  2020-01-01 01:37:00    58849.000000   58849.067361 large_patch
 2020-01-01 04:28:40  2020-01-01 06:07:40    58849.186574   58849.255324 large_patch
 2020-01-01 06:09:20  2020-01-01 06:52:20    58849.256481   58849.286343 small_patch
 2020-01-01 06:54:00  2020-01-01 08:30:00    58849.287500   58849.354167 large_patch
 2020-01-01 21:51:40  2020-01-01 23:27:40    58849.910880   58849.977546 large_patch
 2020-01-01 23:29:20  2020-01-02 00:12:20    58849.978704   58850.008565 small_patch
 2020-01-02 00:14:00  2020-01-02 00:58:00    58850.009722   58850.040278 small_patch
 2020-01-02 04:29:40  2020-01-02 06:07:40    58850.187269   58850.255324 large_patch
 2020-01-02 06:09:20  2020-01-02 06:52:20    58850.256481   58850.286343 small_patch
 2020-01-02 06:54:00  2020-01-02 07:37:00    58850.287500   58850.317361 small_patch
 2020-01-02 21:48:40  2020-01-02 23:24:40    58850.908796   58850.975463 large_patch
 2020-01-02 23:26:20  2020-01-03 01:02:20    58850.976620   58851.043287 large_patch
 2020-01-03 04:24:00  2020-01-03 06:03:00    58851.183333   58851.252083 large_patch
```

```
 2020-01-03 06:04:40  2020-01-03 07:40:40    58851.253241   58851.319907 large_patch
 2020-01-03 21:42:20  2020-01-03 23:18:20    58851.904398   58851.971065 large_patch
```

Now the large patch is observed 9 times and the small patch is observed 4 times. Typically though, we don't use the priority field to normalize the depths. Instead, the user can balance the integration depths with two command line arguments: `--equalize-area` and `--equalize time`.

With `--equalize-area` the scheduler will automatically modulate the user-given priorities with the area of each patch.

With `--equalize-time` the scheduler will balance the actual time spent in each patch rather than the number of visits. There is a difference, because the observing time per pass can vary greatly depending on the patch shape and orientation

```
In [7]: ! toast_ground_schedule.py \
            --site-lat "-22.958064" \
            --site-lon "-67.786222" \
            --site-alt 5200 \
            --site-name Atacama \
            --telescope LAT \
            --start "2020-01-01 00:00:00" \
            --stop "2020-01-04 00:00:00" \
            --patch-coord C \
            --patch small_patch,1,80,-13,10 \
            --patch large_patch,1,80,-33,20 \
            --el-min 30 \
            --el-max 60 \
            --ces-max-time 86400 \
            --sun-avoidance-angle 20 \
            --equalize-area \
            --equalize-time \
            --out schedule.txt

        ! cat schedule.txt

TOAST INFO: Adding patch "small_patch"
TOAST INFO: Center-and-width format
TOAST INFO: Adding patch "large_patch"
TOAST INFO: Center-and-width format
TOAST INFO: Global timer: toast_ground_schedule:  5.93 seconds (1 calls)
#Site            Telescope        Latitude [deg] Longitude [deg]   Elevation [m]
 Atacama         LAT                      -22.958         -67.786          5200.0
#Start time UTC       Stop time UTC        Start MJD      Stop MJD       Patch name
 2020-01-01 00:00:00  2020-01-01 01:37:00    58849.000000   58849.067361 large_patch
 2020-01-01 04:28:40  2020-01-01 06:07:40    58849.186574   58849.255324 large_patch
 2020-01-01 06:09:20  2020-01-01 06:52:20    58849.256481   58849.286343 small_patch
 2020-01-01 06:54:00  2020-01-01 08:30:00    58849.287500   58849.354167 large_patch
 2020-01-01 21:51:40  2020-01-01 23:27:40    58849.910880   58849.977546 large_patch
 2020-01-01 23:29:20  2020-01-02 00:12:20    58849.978704   58850.008565 small_patch
```

```
 2020-01-02 00:14:00  2020-01-02 00:58:00    58850.009722   58850.040278 small_patch
 2020-01-02 04:29:40  2020-01-02 06:07:40    58850.187269   58850.255324 large_patch
 2020-01-02 06:09:20  2020-01-02 06:52:20    58850.256481   58850.286343 small_patch
 2020-01-02 06:54:00  2020-01-02 07:37:00    58850.287500   58850.317361 small_patch
 2020-01-02 21:48:40  2020-01-02 23:24:40    58850.908796   58850.975463 large_patch
 2020-01-02 23:26:20  2020-01-03 01:02:20    58850.976620   58851.043287 large_patch
 2020-01-03 04:24:00  2020-01-03 06:03:00    58851.183333   58851.252083 large_patch
 2020-01-03 06:04:40  2020-01-03 07:40:40    58851.253241   58851.319907 large_patch
 2020-01-03 21:42:20  2020-01-03 23:18:20    58851.904398   58851.971065 large_patch
```

As with the by-hand-modulated priorities, `large_patch` ends up with twice as many visits.

1.1 Binning the schedule

We take an observing schedule from `toast_ground_sim.py` and translate it into a depth map.

First, we need a focalplane. If one does not already exist, TOAST `pipelines` includes a tool for generating mock hexagonal focalplanes:

```
In [8]: ! toast_fake_focalplane.py --help
```

```
usage: toast_fake_focalplane.py [-h] [--minpix MINPIX] [--out OUT]
                                [--fwhm FWHM] [--fwhm_sigma FWHM_SIGMA]
                                [--fov FOV] [--psd_fknee PSD_FKNEE]
                                [--psd_NET PSD_NET] [--psd_alpha PSD_ALPHA]
                                [--psd_fmin PSD_FMIN]
                                [--bandcenter_ghz BANDCENTER_GHZ]
                                [--bandcenter_sigma BANDCENTER_SIGMA]
                                [--bandwidth_ghz BANDWIDTH_GHZ]
                                [--bandwidth_sigma BANDWIDTH_SIGMA]
                                [--random_seed RANDOM_SEED]

Simulate fake hexagonal focalplane.

optional arguments:
  -h, --help            show this help message and exit
  --minpix MINPIX       minimum number of pixels to use
  --out OUT             Root name of output pickle file
  --fwhm FWHM           beam FWHM in arcmin
  --fwhm_sigma FWHM_SIGMA
                        Relative beam FWHM distribution width
  --fov FOV             Field of View in degrees
  --psd_fknee PSD_FKNEE
                        Detector noise model f_knee in Hz
  --psd_NET PSD_NET     Detector noise model NET in K*sqrt(sec)
  --psd_alpha PSD_ALPHA
                        Detector noise model slope
  --psd_fmin PSD_FMIN   Detector noise model f_min in Hz
  --bandcenter_ghz BANDCENTER_GHZ
```

```
                        Band center frequency [GHz]
  --bandcenter_sigma BANDCENTER_SIGMA
                        Relative band center distribution width
  --bandwidth_ghz BANDWIDTH_GHZ
                        Bandwidth [GHz]
  --bandwidth_sigma BANDWIDTH_SIGMA
                        Relative bandwidth distribution width
  --random_seed RANDOM_SEED
                        Random number generator seed for randomized detector
                        parameters
```

Now we create a focalplane with 10-degree FOV and a mininimum of 20 pixels:

```
In [9]: ! toast_fake_focalplane.py \
            --minpix 20 \
            --out focalplane \
            --fwhm 30 \
            --fov 10 \
            --psd_fknee 5e-2 \
            --psd_NET 1e-3 \
            --psd_alpha 1 \
            --psd_fmin 1e-5

TOAST INFO: using 37 pixels (74 detectors)
```

The actual focalplane ends up having 37 pixels, instead of the minimum of 20. This is because regular packing of the hexagon is quantized. Notice that the final name of the focalplane is `focalplane_37.pkl`. We'll need the name to run the simulation script.

We will use the versatile ground simulation pipeline, `toast_ground_sim.py`, to bin the map. It will be covered in detail in lesson 7 so here we simply write out a parameter file:

```
In [10]: %%writefile bin_schedule.par
         --sample-rate
         1
         --scan-rate
         0.3
         --scan-accel
         10.0
         --nside
         64
         --focalplane
         focalplane_37.pkl
         --schedule
         schedule.txt
         --out
         out
         --simulate-noise
```

```
        --freq
        100
        --no-destripe
        --no-binmap
        --hits
        --wcov

Writing bin_schedule.par
```

Then run the pipeline. Because the pipeline uses `libMadam`, an MPI code, we must submit the job to a compute node.

```
In [11]: import subprocess as sp

        command = "toast_ground_sim.py @bin_schedule.par"
        runstr = None

        if nersc_host is not None:
            runstr = "srun -N 1 -C haswell -n 32 -c 2 --cpu_bind=cores -t 00:05:00"
            if nersc_resv is not None:
                runstr = "{} --reservation {}".format(runstr, nersc_resv)
        else:
            # Just use mpirun
            runstr = "mpirun -np 4"

        runcom = "{} {}".format(runstr, command)
        print(runcom, flush=True)
        sp.check_call(runcom, stderr=sp.STDOUT, shell=True)

srun -N 1 -C haswell -n 32 -c 2 --cpu_bind=cores -t 00:05:00 toast_ground_sim.py @bin_schedule
Launched in background. Redirecting stdin to /dev/null
ModuleCmd_Load.c(244):ERROR:105: Unable to locate a modulefile for 'altd'
ModuleCmd_Load.c(244):ERROR:105: Unable to locate a modulefile for 'darshan'
srun: job 25265601 queued and waiting for resources
srun: job 25265601 has been allocated resources
TOAST INFO: Running with 32 processes at 2019-10-19 16:08:08.600211
TOAST INFO: All parameters:
TOAST INFO: group_size = None
TOAST INFO: do_daymaps = False
TOAST INFO: do_seasonmaps = False
TOAST INFO: debug = False
TOAST INFO: scan_rate = 0.3
TOAST INFO: scan_accel = 10.0
TOAST INFO: sun_angle_min = 30.0
TOAST INFO: schedule = schedule.txt
TOAST INFO: weather = None
TOAST INFO: timezone = 0
TOAST INFO: sample_rate = 1.0
```

```
TOAST INFO: coord = C
TOAST INFO: split_schedule = None
TOAST INFO: sort_schedule = True
TOAST INFO: hwp_rpm = None
TOAST INFO: hwp_step_deg = None
TOAST INFO: hwp_step_time_s = None
TOAST INFO: nside = 64
TOAST INFO: single_precision_pointing = False
TOAST INFO: common_flag_mask = 1
TOAST INFO: flush = False
TOAST INFO: apply_polyfilter = False
TOAST INFO: poly_order = 0
TOAST INFO: apply_groundfilter = False
TOAST INFO: ground_order = 0
TOAST INFO: simulate_atmosphere = False
TOAST INFO: focalplane_radius_deg = None
TOAST INFO: atm_verbosity = 0
TOAST INFO: atm_lmin_center = 0.01
TOAST INFO: atm_lmin_sigma = 0.001
TOAST INFO: atm_lmax_center = 10.0
TOAST INFO: atm_lmax_sigma = 10.0
TOAST INFO: atm_gain = 3e-05
TOAST INFO: atm_zatm = 40000.0
TOAST INFO: atm_zmax = 200.0
TOAST INFO: atm_xstep = 10.0
TOAST INFO: atm_ystep = 10.0
TOAST INFO: atm_zstep = 10.0
TOAST INFO: atm_nelem_sim_max = 1000
TOAST INFO: atm_wind_dist = 500.0
TOAST INFO: atm_z0_center = 2000.0
TOAST INFO: atm_z0_sigma = 0.0
TOAST INFO: atm_T0_center = 280.0
TOAST INFO: atm_T0_sigma = 10.0
TOAST INFO: atm_cache = atm_cache
TOAST INFO: simulate_noise = True
TOAST INFO: apply_gainscrambler = False
TOAST INFO: gain_sigma = 0.01
TOAST INFO: madam_prefix = toast
TOAST INFO: madam_iter_max = 1000
TOAST INFO: madam_precond_width = 100
TOAST INFO: madam_precond_width_min = None
TOAST INFO: madam_precond_width_max = None
TOAST INFO: madam_baseline_length = 10000.0
TOAST INFO: madam_baseline_order = 0
TOAST INFO: madam_noisefilter = False
TOAST INFO: madam_parfile = None
TOAST INFO: madam_allreduce = False
TOAST INFO: madam_concatenate_messages = True
```

```
TOAST INFO: destripe = False
TOAST INFO: write_binmap = False
TOAST INFO: write_hits = True
TOAST INFO: write_wcov = True
TOAST INFO: write_wcov_inv = True
TOAST INFO: conserve_memory = True
TOAST INFO: input_map = None
TOAST INFO: pysm_model = None
TOAST INFO: pysm_apply_beam = True
TOAST INFO: pysm_precomputed_cmb_K_CMB = None
TOAST INFO: ground_map = None
TOAST INFO: ground_nside = 128
TOAST INFO: ground_fwhm_deg = 10
TOAST INFO: ground_lmax = 256
TOAST INFO: ground_scale = 0.001
TOAST INFO: ground_power = -1
TOAST INFO: simulate_ground = False
TOAST INFO: tidas = None
TOAST INFO: spt3g = None
TOAST INFO: MC_start = 0
TOAST INFO: MC_count = 1
TOAST INFO: outdir = out
TOAST INFO: focalplane = focalplane_37.pkl
TOAST INFO: freq = 100
TOAST INFO: Parsed parameters:  0.04 seconds (1 calls)
TOAST INFO: Load 15 (sub)scans in schedule.txt:  0.01 seconds (1 calls)
TOAST INFO: Loading schedule(s):  0.02 seconds (1 calls)
TOAST INFO: Loading focalplanes:  0.02 seconds (1 calls)
TOAST INFO: Group #    0 has 15 observations.
TOAST INFO: Simulated scans:  0.61 seconds (1 calls)
TOAST INFO: Expanding pointing
TOAST INFO: Pointing generation:  0.10 seconds (1 calls)
TOAST INFO: Scanning local submaps
TOAST INFO: Identify local submaps:  0.02 seconds (1 calls)
TOAST INFO: Processing frequency 100.0GHz 1 / 1, MC = 0
TOAST INFO: Simulating noise
TOAST INFO: Simulate noise:  0.12 seconds (1 calls)
TOAST INFO: Making maps
TOAST INFO: Mapping toast_100_telescope_all_time_all
TOAST INFO: OpMadam: 100.00 % of samples are included in valid intervals.
TOAST INFO: Collect period ranges:  0.00 seconds (1 calls)
TOAST INFO: Collect dataset dimensions:  0.00 seconds (1 calls)
TOAST INFO: Stage time:  0.00 seconds (1 calls)
<toast.Environment
  Source code version = 2.3.1.dev1428
  Logging level = INFO
  Handling enabled for 0 signals:
  Max threads = 4
```

```
  MPI build enabled
  MPI runtime enabled
>
Memory usage after staging time
       total :  125.785 GB  <  125.785 +-    0.000 GB  <  125.785 GB
   available :  117.358 GB  <  117.358 +-    0.000 GB  <  117.358 GB
     percent :    6.700 %   <    6.700 +-    0.000 %   <    6.700 %
        used :    7.252 GB  <    7.252 +-    0.000 GB  <    7.252 GB
        free :  103.900 GB  <  103.900 +-    0.000 GB  <  103.900 GB
      active :    4.147 GB  <    4.147 +-    0.000 GB  <    4.147 GB
    inactive :   13.818 GB  <   13.818 +-    0.000 GB  <   13.818 GB
     buffers :   20.824 MB  <   20.824 +-    0.000 MB  <   20.824 MB
      cached :   14.613 GB  <   14.613 +-    0.000 GB  <   14.613 GB
      shared :  396.680 MB  <  396.680 +-    0.000 MB  <  396.680 MB
        slab :    2.217 GB  <    2.217 +-    0.000 GB  <    2.217 GB

TOAST INFO: Stage signal:  0.07 seconds (1 calls)
Node has 0.300 GB allocated in TOAST TOD caches and 0.017 GB in Madam caches (0.317 GB total) a
Memory usage after staging signal
       total :  125.785 GB  <  125.785 +-    0.000 GB  <  125.785 GB
   available :  117.322 GB  <  117.322 +-    0.000 GB  <  117.322 GB
     percent :    6.700 %   <    6.700 +-    0.000 %   <    6.700 %
        used :    7.288 GB  <    7.288 +-    0.000 GB  <    7.288 GB
        free :  103.863 GB  <  103.863 +-    0.000 GB  <  103.863 GB
      active :    4.194 GB  <    4.194 +-    0.000 GB  <    4.194 GB
    inactive :   13.818 GB  <   13.818 +-    0.000 GB  <   13.818 GB
     buffers :   20.824 MB  <   20.824 +-    0.000 MB  <   20.824 MB
      cached :   14.614 GB  <   14.614 +-    0.000 GB  <   14.614 GB
      shared :  396.750 MB  <  396.750 +-    0.000 MB  <  396.750 MB
        slab :    2.217 GB  <    2.217 +-    0.000 GB  <    2.217 GB

TOAST INFO: Stage pixels 1 / 32:  0.07 seconds (1 calls)
TOAST INFO: Stage pixels 2 / 32:  0.01 seconds (1 calls)
TOAST INFO: Stage pixels 3 / 32:  0.01 seconds (1 calls)
TOAST INFO: Stage pixels 4 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixels 5 / 32:  0.01 seconds (1 calls)
TOAST INFO: Stage pixels 6 / 32:  0.01 seconds (1 calls)
TOAST INFO: Stage pixels 7 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixels 8 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixels 9 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixels 10 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixels 11 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixels 12 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixels 13 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixels 14 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixels 15 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixels 16 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixels 17 / 32:  0.00 seconds (1 calls)
```

```
TOAST INFO: Stage pixels 18 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixels 19 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixels 20 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixels 21 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixels 22 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixels 23 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixels 24 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixels 25 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixels 26 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixels 27 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixels 28 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixels 29 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixels 30 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixels 31 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixels 32 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixels:  0.13 seconds (1 calls)
Node has 0.300 GB allocated in TOAST TOD caches and 0.056 GB in Madam caches (0.356 GB total) a
Memory usage after staging pixels
       total :  125.785 GB  <  125.785 +-    0.000 GB  <  125.785 GB
   available :  117.301 GB  <  117.301 +-    0.000 GB  <  117.301 GB
     percent :    6.700 %   <    6.700 +-    0.000 %   <    6.700 %
        used :    7.310 GB  <    7.310 +-    0.000 GB  <    7.310 GB
        free :  103.841 GB  <  103.841 +-    0.000 GB  <  103.841 GB
      active :    4.217 GB  <    4.217 +-    0.000 GB  <    4.217 GB
    inactive :   13.820 GB  <   13.820 +-    0.000 GB  <   13.820 GB
     buffers :   20.824 MB  <   20.824 +-    0.000 MB  <   20.824 MB
      cached :   14.614 GB  <   14.614 +-    0.000 GB  <   14.614 GB
      shared :  396.891 MB  <  396.891 +-    0.000 MB  <  396.891 MB
        slab :    2.218 GB  <    2.218 +-    0.000 GB  <    2.218 GB

TOAST INFO: Stage pixel weights 1 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixel weights 2 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixel weights 3 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixel weights 4 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixel weights 5 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixel weights 6 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixel weights 7 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixel weights 8 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixel weights 9 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixel weights 10 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixel weights 11 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixel weights 12 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixel weights 13 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixel weights 14 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixel weights 15 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixel weights 16 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixel weights 17 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixel weights 18 / 32:  0.00 seconds (1 calls)
```

```
TOAST INFO: Stage pixel weights 19 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixel weights 20 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixel weights 21 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixel weights 22 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixel weights 23 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixel weights 24 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixel weights 25 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixel weights 26 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixel weights 27 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixel weights 28 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixel weights 29 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixel weights 30 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixel weights 31 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixel weights 32 / 32:  0.00 seconds (1 calls)
TOAST INFO: Stage pixel weights:  0.17 seconds (1 calls)
Node has 0.266 GB allocated in TOAST TOD caches and 0.076 GB in Madam caches (0.341 GB total) a
Memory usage after staging pixel weights
       total :  125.785 GB  <  125.785 +-    0.000 GB  <  125.785 GB
   available :  117.244 GB  <  117.244 +-    0.000 GB  <  117.244 GB
     percent :    6.800 %   <    6.800 +-    0.000 %   <    6.800 %
        used :    7.366 GB  <    7.366 +-    0.000 GB  <    7.366 GB
        free :  103.783 GB  <  103.783 +-    0.000 GB  <  103.783 GB
      active :    4.275 GB  <    4.275 +-    0.000 GB  <    4.275 GB
    inactive :   13.820 GB  <   13.820 +-    0.000 GB  <   13.820 GB
     buffers :   20.824 MB  <   20.824 +-    0.000 MB  <   20.824 MB
      cached :   14.615 GB  <   14.615 +-    0.000 GB  <   14.615 GB
      shared :  397.031 MB  <  397.031 +-    0.000 MB  <  397.031 MB
        slab :    2.218 GB  <    2.218 +-    0.000 GB  <    2.218 GB

TOAST INFO: Stage all data:  0.51 seconds (1 calls)
TOAST INFO: Collect PSD info:  0.00 seconds (1 calls)
Node has 0.148 GB allocated in TOAST TOD caches and 0.135 GB in Madam caches (0.282 GB total) a
Memory usage just before calling libmadam.destripe
       total :  125.785 GB  <  125.785 +-    0.000 GB  <  125.785 GB
   available :  117.244 GB  <  117.244 +-    0.000 GB  <  117.244 GB
     percent :    6.800 %   <    6.800 +-    0.000 %   <    6.800 %
        used :    7.367 GB  <    7.367 +-    0.000 GB  <    7.367 GB
        free :  103.782 GB  <  103.782 +-    0.000 GB  <  103.782 GB
      active :    4.276 GB  <    4.276 +-    0.000 GB  <    4.276 GB
    inactive :   13.821 GB  <   13.821 +-    0.000 GB  <   13.821 GB
     buffers :   20.824 MB  <   20.824 +-    0.000 MB  <   20.824 MB
      cached :   14.616 GB  <   14.616 +-    0.000 GB  <   14.616 GB
      shared :  397.070 MB  <  397.070 +-    0.000 MB  <  397.070 MB
        slab :    2.218 GB  <    2.218 +-    0.000 GB  <    2.218 GB

OMP: 32 tasks with 2 procs per node, 4 threads per task.

 Program MADAM
```

```
 Destriping of CMB data with a noise filter
 Version 3.7


Examining periods

Flagged 0 samples on 0 periods that had less than  0.100% of unflagged samples
Total number of samples (single detector): 2275200
Zero-weight fraction (all detectors):   0.000 %
Flagged fraction (all detectors):   2.316 %
Total number of intervals: 480
Max number of samples per task: 71100

 Initializing parameters
 Adjusting noise weights using noise spectra
 Polarized detectors present:  Will produce polarization maps
 noise_weights_from_psd =  T
 radiometers =  T
 mode_detweight =            0

 istart_mission        =            0
 nosamples_tot         =      2275200
 Initializing parallelization
 ntasks                =           32  Number or processes
 nosamples_tot         =      2275200  Total samples
 nosamples_proc_max    =        71100  Samples/process

 MCMode                =            F
 write_cut             =            F
 basis_func            = Legendre      Destriping function basis
 basis_order           =            0  Destriping function order

 bin_subsets           =            F

 ntasks                =           32  Number of processes
 nthreads              =            4  Number of threads per process
 info                  =            3  Screen output level
 fsample               =       1.0000  Sampling frequency (Hz)
 nmap                  =            3  Polarization included
 ncc                   =            6  Independent wcov elements
 Input files:

 nside_map             =           64  Healpix resolution (output map)
 nside_cross           =           32  Healpix resolution (destriping)
 nside_submap          =           16  Submap resolution
 concatenate_messages  =            T  use mpi_alltoallv to communicate
 reassign_submaps      =            T  minimize communication by reassigning submaps
 pixmode_map           =            2  Pixel rejection criterion (output map)
```

```
pixmode_cross         =           2  Pixel rejection criterion (destriping)
pixlim_map            =  1.00000E-02  Pixel rejection limit (output map)
pixlim_cross          =  1.00000E-03  Pixel rejection limit (destriping)

Standard mode
psdlen                =     1000000  Length of requested noise PSD
psd_downsample        =          10  PSD downsampling factor
kfirst                =           F  First destriping OFF

cglimit               =  1.00000E-12  Iteration convergence limit
iter_min              =           3  Minimum number of iterations
iter_max              =        1000  Maximum number of iterations
precond_width_min     =           0  Min width of the preconditioner band matrix
precond_width_max     =           0  Max width of the preconditioner band matrix
No preconditioning
flag_by_horn          =           F  Flags are independent
mode_detweight        =           0  Detector weighting mode: sigma from simulation file


time_unit             =          pp  Time unit = pointing period
mission_time          =         480  Mission length in time units
nosamples_tot         =     2275200  Total samples
                      =    632.0000  hours

Detectors available on the FIRST process and noise according to the FIRST period
detector              sigma         weight 1/sqrt(weight)
fake_00A        0.10488E-02    0.90911E+06    0.10488E-02
fake_00B        0.10488E-02    0.90911E+06    0.10488E-02
fake_01A        0.10488E-02    0.90911E+06    0.10488E-02

Output files
file_root    = toast_100_telescope_all_time_all
file_hit     = out/00000000/100/toast_100_telescope_all_time_all_hmap.fits
file_matrix  = out/00000000/100/toast_100_telescope_all_time_all_wcov_inv.fits
file_wcov    = out/00000000/100/toast_100_telescope_all_time_all_wcov.fits
binary_output      =  F
concatenate_binary =  F

TOD memory               min =      1.70 MB   max =     2.24 MB   total =    59.67 MB
Baseline memory          min =      0.00 B    max =     0.00 B    total =     0.00 B
Pointing memory          min =      2.71 MB   max =     4.07 MB   total =   100.35 MB
Basis function memory    min =      0.00 B    max =     0.00 B    total =     0.00 B
Clock =     0.014 s
Total submaps = 3072  submap size = 16
Local submaps:  min = 74  max = 94  mean =    85.28
Submap table memory      min =    384.00 kB   max =   384.00 kB   total =    12.00 MB
local maps memory        min =    115.72 kB   max =   146.97 kB   total =     4.17 MB
All2allv memory          min =     97.62 kB   max =   186.00 kB   total =     4.68 MB
```

```
Map memory                min =    156.00 kB   max =   156.00 kB   total =     4.88 MB
Building pixel matrices...
Counting hits...
Binning TOD...
Clock =     0.062 s

Finalization begins

Clock =     0.065 s
Writing pixel matrix...
 0 : Write matrix completed in   0.258 s
Inverting pixel matrices...
Pixel matrix written in out/00000000/100/toast_100_telescope_all_time_all_wcov_inv.fits

    1417 pixels solved
   47735 pixels unsolved
       0 pixels had decoupled intensity
 0 : Invert pixelmatrix_map completed in   0.317 s
Constructing output map...

Destriped map:
Map                     1                2                3
Std           0.00000E+00 K    0.00000E+00 K    0.00000E+00 K
Mean          0.00000E+00 K    0.00000E+00 K    0.00000E+00 K
Min           0.00000E+00 K    0.00000E+00 K    0.00000E+00 K
Max           0.00000E+00 K    0.00000E+00 K    0.00000E+00 K

Writing pixel matrix...
 0 : Write matrix completed in   0.244 s
Writing hits...
Pixel matrix written in out/00000000/100/toast_100_telescope_all_time_all_wcov.fits
Hit count written in out/00000000/100/toast_100_telescope_all_time_all_hmap.fits
Clock =     0.966 s

MEMORY (MB):
Detector pointing         min =      2.71 MB   max =     4.07 MB   total =   100.35 MB
TOD buffer                min =      1.70 MB   max =     2.24 MB   total =    59.67 MB
Maps                      min =    156.00 kB   max =   156.00 kB   total =     4.88 MB
Baselines                 min =      0.60 kB   max =     0.60 kB   total =    19.22 kB
Basis functions           min =      0.00 B    max =     0.00 B    total =     0.00 B
Noise filter              min =      0.00 B    max =     0.00 B    total =     0.00 B
Preconditioner            min =      0.00 B    max =     0.00 B    total =     0.00 B
Submap table              min =    384.00 kB   max =   384.00 kB   total =    12.00 MB
Temporary maps            min =    115.72 kB   max =   146.97 kB   total =     4.17 MB
All2All buffers           min =     97.62 kB   max =   186.00 kB   total =     4.68 MB
CG work space             min =      0.00 B    max =     0.00 B    total =     0.00 B
NCM                       min =      0.00 B    max =     0.00 B    total =     0.00 B
Total                     min =      5.16 MB   max =     7.15 MB   total =   185.76 MB
```

```
 WALL-CLOCK TIME (s):
 Inverting pixel matrices        mean =     0.3  min =     0.3  max =     0.3
 Finalization and output         mean =     0.9  min =     0.9  max =     0.9
 Total                          1.0 s  (        0.03 CPU hours)

TOAST INFO: Unstage signal 1 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixels 1 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixel weights 1 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage data 1 / 32:  0.01 seconds (1 calls)
TOAST INFO: Unstage signal 2 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixels 2 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixel weights 2 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage data 2 / 32:  0.01 seconds (1 calls)
TOAST INFO: Unstage signal 3 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixels 3 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixel weights 3 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage data 3 / 32:  0.01 seconds (1 calls)
TOAST INFO: Unstage signal 4 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixels 4 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixel weights 4 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage data 4 / 32:  0.01 seconds (1 calls)
TOAST INFO: Unstage signal 5 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixels 5 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixel weights 5 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage data 5 / 32:  0.01 seconds (1 calls)
TOAST INFO: Unstage signal 6 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixels 6 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixel weights 6 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage data 6 / 32:  0.01 seconds (1 calls)
TOAST INFO: Unstage signal 7 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixels 7 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixel weights 7 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage data 7 / 32:  0.01 seconds (1 calls)
TOAST INFO: Unstage signal 8 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixels 8 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixel weights 8 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage data 8 / 32:  0.01 seconds (1 calls)
TOAST INFO: Unstage signal 9 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixels 9 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixel weights 9 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage data 9 / 32:  0.01 seconds (1 calls)
TOAST INFO: Unstage signal 10 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixels 10 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixel weights 10 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage data 10 / 32:  0.01 seconds (1 calls)
TOAST INFO: Unstage signal 11 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixels 11 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixel weights 11 / 32:  0.00 seconds (1 calls)
```

```
TOAST INFO: Unstage data 11 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage signal 12 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixels 12 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixel weights 12 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage data 12 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage signal 13 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixels 13 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixel weights 13 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage data 13 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage signal 14 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixels 14 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixel weights 14 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage data 14 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage signal 15 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixels 15 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixel weights 15 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage data 15 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage signal 16 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixels 16 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixel weights 16 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage data 16 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage signal 17 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixels 17 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixel weights 17 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage data 17 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage signal 18 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixels 18 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixel weights 18 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage data 18 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage signal 19 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixels 19 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixel weights 19 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage data 19 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage signal 20 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixels 20 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixel weights 20 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage data 20 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage signal 21 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixels 21 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixel weights 21 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage data 21 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage signal 22 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixels 22 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixel weights 22 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage data 22 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage signal 23 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixels 23 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixel weights 23 / 32:  0.00 seconds (1 calls)
```

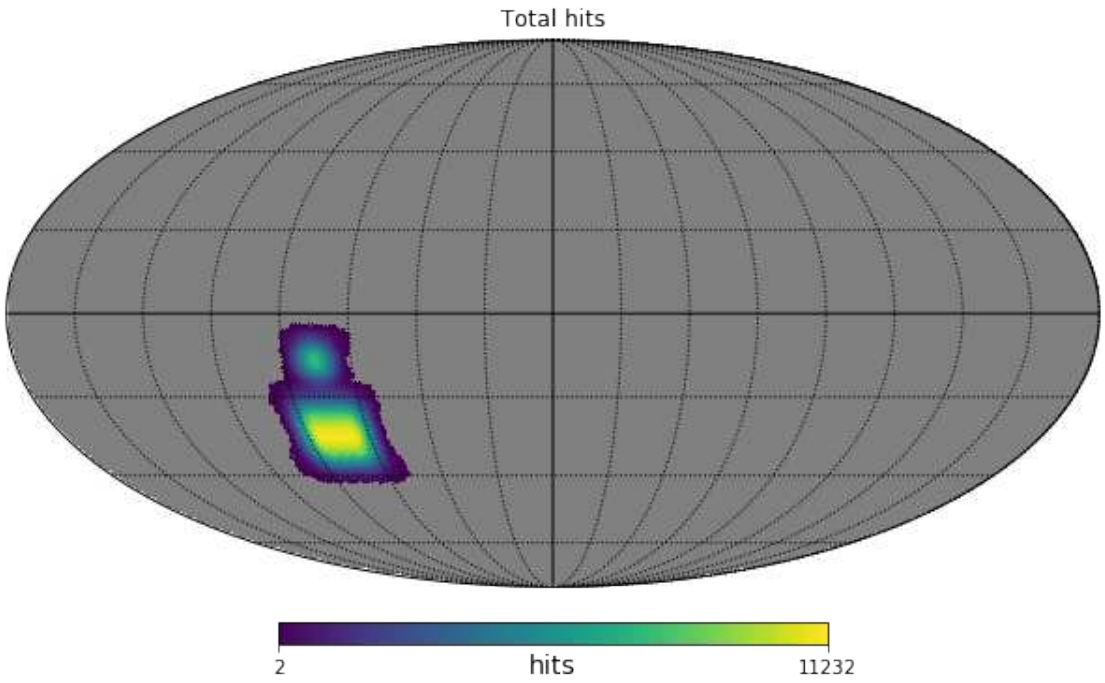
```
TOAST INFO: Unstage data 23 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage signal 24 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixels 24 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixel weights 24 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage data 24 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage signal 25 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixels 25 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixel weights 25 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage data 25 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage signal 26 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixels 26 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixel weights 26 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage data 26 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage signal 27 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixels 27 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixel weights 27 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage data 27 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage signal 28 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixels 28 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixel weights 28 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage data 28 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage signal 29 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixels 29 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixel weights 29 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage data 29 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage signal 30 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixels 30 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixel weights 30 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage data 30 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage signal 31 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixels 31 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixel weights 31 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage data 31 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage signal 32 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixels 32 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage pixel weights 32 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage data 32 / 32:  0.00 seconds (1 calls)
TOAST INFO: Unstage all data:  0.17 seconds (1 calls)
TOAST INFO: Mapping toast_100_telescope_all_time_all:  1.72 seconds (1 calls)
TOAST INFO: Madam total:  1.74 seconds (1 calls)
TOAST INFO: Gather and dump timing info:  0.01 seconds (1 calls)
TOAST INFO: toast_ground_sim.py:  2.72 seconds (1 calls)
```

```
Out[11]: 0
```

Let's examine the resulting hits and depth map. The file naming convention may seem a little awkward but follows from the fact that a single run of `toast_ground_sim.py` may map multiple telescopes, frequencies and time splits.

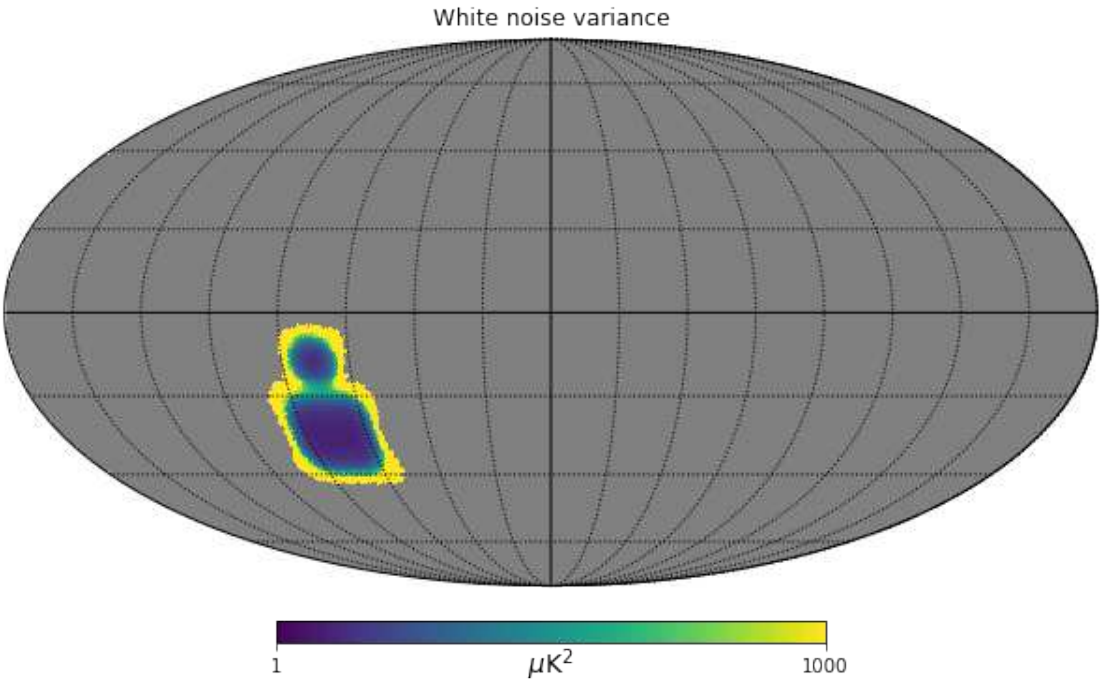
In [12]:
```
import matplotlib.pyplot as plt
%matplotlib inline
import healpy

hits = healpy.read_map("out/00000000/100/toast_100_telescope_all_time_all_hmap.fits")
hits[hits == 0] = healpy.UNSEEN
healpy.mollview(hits, unit="hits", title="Total hits")
healpy.graticule(22.5, verbose=False)
```

```
NSIDE = 64
ORDERING = NESTED in fits file
INDXSCHM = IMPLICIT
Ordering converted to RING
```



In [13]:
```
wcov = healpy.read_map("out/00000000/100/toast_100_telescope_all_time_all_wcov.fits")
wcov *= 1e12  # from K^2 to uK^2
wcov[wcov == 0] = healpy.UNSEEN
healpy.mollview(wcov, unit="$\mu$K$^2$", title="White noise variance", min=1e0, max=1e
healpy.graticule(22.5, verbose=False)
```

```
NSIDE = 64
ORDERING = NESTED in fits file
INDXSCHM = IMPLICIT
Ordering converted to RING
```



1.2 Advanced topics in ground scheduling

1.2.1 Cooler cycle format

it is possible to instruct the scheduler to add regular breaks in the schedule to cycle the cooler or to perform other maintenance activities. The cooler cycle is a pseudo patch that the scheduler considers like other targets when deciding what to observe next. The full syntax is:

`--patch <name>,COOLER,<weight>,<power>,<hold_time_min>,<hold_time_max>,<cycle_time>,<az>,<el`

All of the time arguments are given in hours. The priority of the patch depends on the time since the last cycle occurred. It is `infinity` until `hold_time_min` has elapsed and then begins to decrease according to a power law set by `power`. Priority at `hold_time_max` is zero.

1.2.2 Planet scans

The scheduler can target planets just like stationary patches. The SSO (solar system object) format is

`--patch <name>,SSO,<priority>,<radius [deg]>`

All orbiting bodies recognized by `pyEphem` are supported.

1.2.3 Oscillating patches

The scheduler designs the scans so that the azimuth range is kept fixed and the boresight sweeps the entire patch. This usually implies a certain amount of spillover integration time outside the patch. This can produce an excess of hits at the boundary of two patches. The scheduler offers a

way to smear the spillover by systematically shifting the position of the patches in RA and DEC. The arguments to accomplish this are

```
--ra-period <period [visits]>
--ra-amplitude <amplitude [deg]>
--dec-period <period [visits]>
--dec-amplitude <amplitude [deg]>
```

Patches will systematically shift after each visit, returning to their fiducial positions after each period.

1.2.4 Horizontal (high cadence) patch definition

Horizontal patch definition specifies the observing elevation and the azimuth range. The scheduler parks the telescope at the given elevation and scans until the constraints (Sun, Moon, cooler hold time) prevent continuing. If possible, scanning is continued by switching between rising and setting scan.

```
--patch <name>,HORIZONTAL,<priority>,<az min [deg]>,<az max [deg]>,<el
[deg]>,<scan time [min]>
```

1.2.5 Polygon patch definition

Patches do not need to be rectangular or circular. An arbitrary polygon shape can be specified by giving the corner coordinates.

```
--patch <name>,<priority>,<RA_0 [deg]>,<DEC_0 [deg]>,...„<RA_N-1
[deg]>,<DEC_N-1 [deg]>
```

1.2.6 Elevation penalty

Lower observing elevations are subject to higher levels of photon noise from the atmosphere. It is possible to instruct the scheduler to modulate the relative priorities of the available patches based on their elevation.

```
--elevation-penalty-limit <elevation [deg]>
--elevation-penalty-power <power>
```

If the available patch is below `elevation-penalty-limit`, the priority is modulated by $\left(\frac{limit}{elevation}\right)^{power}$. This way low elevation scans are reserved for targets that cannot be observed at higher elevation or when no targets are available higher.

1.2.7 Block-out

January and February weather in the Atacama is known to be problematic for observing. It is possible to instruct the scheduler to skip certain periods of the calendar year with

```
--block-out <start month>/<start day>-<end month>/<end day>
```

or with

```
--block-out <start year>/<start month>/<start day>-<end year>/<end month>/<end
day>
```

All fields are integers. The dates are in UTC.

simscan_satellite

November 1, 2019

1 Simulated Satellite Scan Strategies

```
In [1]: # Load common tools for all lessons
        import sys
        sys.path.insert(0, "..")
        from lesson_tools import (
            fake_focalplane
        )
```

1.1 Overview

A generic satellite scanning strategy can be described in terms of precession and spin axes, the opening angles about each of them, and the rates of rotation. The precession axis itself is typically oriented in the anti-sun direction and that orientation must be updated either in steps or with a continuous slewing motion. Here is a cartoon (drawn from an old, public LiteBIRD talk) showing a sketch of these angles:

For a real satellite experiment such as Planck or LiteBIRD, there are many custom details, such as simulating repointing maneuvers, simulating any lost time due to cooler cycling, nutation effects, etc. Those are contained in classes for the specific experiments. The tools built in to the core TOAST package are intended for rough simulations to study things like scan strategy choices.

1.1.1 TOD Class for Simulations

In the introductory lesson we saw the use of a TOD derived class providing things like telescope pointing. Here we introduce the `TODSatellite` class which serves that purpose for generic satellite simulations.

```
In [2]: import numpy as np

        import toast
        from toast.todmap import (
            slew_precession_axis,
            TODSatellite
        )
```

```
In [3]: # Default Comm (one group for this example)

        comm = toast.Comm()
```

In [4]:
```
# Create our fake focalplane

fp = fake_focalplane()

detnames = list(sorted(fp.keys()))
detquat = {x: fp[x]["quat"] for x in detnames}
```

In [5]:
```
# Scan parameters (made up, not physically motivated)

samplerate = 10.0
precperiod = 90.0
precangle = 45.0
spinperiod = 1.0
spinangle = 45.0
```

In [6]:
```
# We can simulate a simplistic HWP

hwprpm = 6.0
```

In [7]:
```
# Number of samples

nsamples = 100
```

In [8]:
```
# Instantiate a TOD

tod = TODSatellite(
    comm.comm_group,
    detquat,
    nsamples,
    firstsamp=0,
    firsttime=0.0,
    rate=samplerate,
    spinperiod=spinperiod,
    spinangle=spinangle,
    precperiod=precperiod,
    precangle=precangle,
    coord="E",
    hwprpm=hwprpm
)
```

In [9]:
```
# The TOD constructor above specifies the scan parameters, but the boresight
# simulation is not done until we set the location of the precession axis as a
# function of time.  There is a reason for this delayed construction.  The data
# distribution occurs during the above construction, and we might want to only
# simulate the precession axis motion for our local data.  In reality this is
# cheap enough to do on one process and distribute during the construction.

qprec = np.empty(4 * tod.local_samples[1], dtype=np.float64).reshape((-1, 4))
```

```
        deg_per_day = 1.0

        slew_precession_axis(
            qprec,
            firstsamp=tod.local_samples[0],
            samplerate=samplerate,
            degday=deg_per_day,
        )

        tod.set_prec_axis(qprec=qprec)
```

In [10]:
```
# Now we can read from this TOD object

         print("TOD timestampes = {} ...".format(tod.read_times()[:5]))
         print("TOD boresight = \n{} ...".format(tod.read_boresight()[:5,:]))
         for d in detnames:
             print("TOD detector {} = {} ...".format(d, tod.read(detector=d, n=5)))
             print("TOD detector {} flags = {} ...".format(d, tod.read_flags(detector=d, n=5)))
             print("TOD detector {} pointing = {} ...".format(d, tod.read_pntg(detector=d, n=5)
```

```
TOD timestampes = [0.  0.1 0.2 0.3 0.4] ...
TOD boresight =
[[ 0.5         0.5        -0.5         0.5       ]
 [ 0.50372461  0.50002202 -0.49626167  0.49996385]
 [ 0.50743541  0.50002989 -0.49250974  0.49991357]
 [ 0.51113228  0.50002363 -0.4887443   0.49984914]
 [ 0.51481514  0.50000321 -0.48496547  0.49977059]] ...
TOD detector 0A = [0. 0. 0. 0. 0.] ...
TOD detector 0A flags = [0 0 0 0 0] ...
TOD detector 0A pointing = [[ 0.65328148  0.27059805 -0.27059805  0.65328148]
 [ 0.656731    0.26919305 -0.26715812  0.65181749]
 [ 0.66016234  0.26778026 -0.26371103  0.65033523]
 [ 0.66357541  0.26635974 -0.26025687  0.64883474]
 [ 0.66697012  0.26493151 -0.25679575  0.64731607]] ...
TOD detector 0B = [0. 0. 0. 0. 0.] ...
TOD detector 0B flags = [0 0 0 0 0] ...
TOD detector 0B pointing = [[ 0.65328148 -0.27059805  0.27059805  0.65328148]
 [ 0.65472717 -0.27403071  0.27199525  0.64981388]
 [ 0.65615451 -0.27745603  0.27338459  0.64632831]
 [ 0.65756345 -0.2808739   0.27476605  0.64282484]
 [ 0.65895396 -0.28428423  0.27613957  0.6393036 ]] ...
TOD detector 1A = [0. 0. 0. 0. 0.] ...
TOD detector 1A flags = [0 0 0 0 0] ...
TOD detector 1A pointing = [[ 0.50501286  0.50501286 -0.49493637  0.49493637]
 [ 0.50869961  0.50503451 -0.4911607   0.4949    ]
 [ 0.51237241  0.50504188 -0.48737156  0.49484964]
 [ 0.51603116  0.50503497 -0.48356907  0.49478528]
 [ 0.51967576  0.50501376 -0.47975332  0.49470694]] ...
```

```
TOD detector 1B = [0. 0. 0. 0. 0.] ...
TOD detector 1B flags = [0 0 0 0 0] ...
TOD detector 1B pointing = [[ 7.14196038e-01 -5.55111512e-17 -1.11022302e-16  6.99945726e-01]
 [ 7.16818276e-01 -2.59161549e-03  2.64408538e-03  6.97250208e-01]
 [ 7.19420549e-01 -5.18346747e-03  5.28779671e-03  6.94535272e-01]
 [ 7.22002785e-01 -7.77548483e-03  7.93106150e-03  6.91800997e-01]
 [ 7.24564909e-01 -1.03675965e-02  1.05738073e-02  6.89047458e-01]] ...
TOD detector 2A = [0. 0. 0. 0. 0.] ...
TOD detector 2A flags = [0 0 0 0 0] ...
TOD detector 2A pointing = [[ 0.65417834  0.27764851 -0.26352011  0.6523183 ]
 [ 0.65759801  0.27622036 -0.26005044  0.65087705]
 [ 0.6609995   0.27478423 -0.25657382  0.64941756]
 [ 0.66438269  0.27334016 -0.25309032  0.64793988]
 [ 0.6677475   0.2718882  -0.24960006  0.64644404]] ...
TOD detector 2B = [0. 0. 0. 0. 0.] ...
TOD detector 2B flags = [0 0 0 0 0] ...
TOD detector 2B pointing = [[ 0.65890108 -0.26624679  0.27492183  0.64759555]
 [ 0.6603093  -0.26967473  0.27635614  0.64412301]
 [ 0.66169902 -0.27309543  0.27778248  0.64063265]
 [ 0.66307019 -0.27650882  0.2792008   0.63712456]
 [ 0.66442277 -0.27991479  0.28061107  0.63359886]] ...
TOD detector 3A = [0. 0. 0. 0. 0.] ...
TOD detector 3A flags = [0 0 0 0 0] ...
TOD detector 3A pointing = [[ 0.49309224  0.50181874 -0.49813049  0.50685699]
 [ 0.49683581  0.50180832 -0.49441092  0.50685345]
 [ 0.50056576  0.50178371 -0.49067781  0.50683559]
 [ 0.50428199  0.5017449  -0.48693124  0.50680341]
 [ 0.50798438  0.50169191 -0.48317134  0.50675689]] ...
TOD detector 3B = [0. 0. 0. 0. 0.] ...
TOD detector 3B flags = [0 0 0 0 0] ...
TOD detector 3B pointing = [[ 0.7035083   0.00617057  0.00617057  0.71063346]
 [ 0.70614804  0.00351609  0.0087982   0.70800083]
 [ 0.70876811  0.00086122  0.01142528  0.70534849]
 [ 0.71136844 -0.00179399  0.01405174  0.70267651]
 [ 0.71394896 -0.00444945  0.01667751  0.69998497]] ...
TOD detector 4A = [0. 0. 0. 0. 0.] ...
TOD detector 4A flags = [0 0 0 0 0] ...
TOD detector 4A pointing = [[ 0.49493637  0.49493637 -0.50501286  0.50501286]
 [ 0.49869846  0.49495875 -0.50131225  0.50497694]
 [ 0.50244687  0.49496713 -0.4975979   0.50492673]
 [ 0.50618151  0.49496151 -0.49386991  0.50486225]
 [ 0.50990226  0.49494189 -0.49012838  0.5047835 ]] ...
TOD detector 4B = [0. 0. 0. 0. 0.] ...
TOD detector 4B flags = [0 0 0 0 0] ...
TOD detector 4B pointing = [[ 6.99945726e-01  0.00000000e+00 -1.11022302e-16  7.14196038e-01]
 [ 7.02621751e-01 -2.64437272e-03  2.59132230e-03  7.11553911e-01]
 [ 7.05278207e-01 -5.28897433e-03  5.18226587e-03  7.08891968e-01]
 [ 7.07915019e-01 -7.93373230e-03  7.77275966e-03  7.06210285e-01]
```

```
 [ 7.10532113e-01 -1.05785741e-02  1.03627326e-02  7.03508937e-01]] ...
TOD detector 5A = [0. 0. 0. 0. 0.] ...
TOD detector 5A flags = [0 0 0 0 0] ...
TOD detector 5A pointing = [[ 0.50181874  0.49309224 -0.50685699  0.49813049]
 [ 0.50556168  0.49314706 -0.50313781  0.49806195]
 [ 0.50929075  0.49318794 -0.49940483  0.49797932]
 [ 0.51300585  0.49321485 -0.49565816  0.49788261]
 [ 0.51670688  0.49322781 -0.49189789  0.49777183]] ...
TOD detector 5B = [0. 0. 0. 0. 0.] ...
TOD detector 5B flags = [0 0 0 0 0] ...
TOD detector 5B pointing = [[ 0.7035083  -0.00617057 -0.00617057  0.71063346]
 [ 0.70619373 -0.00877846 -0.00358917  0.70795514]
 [ 0.70885948 -0.01138641 -0.00100798  0.7052571 ]
 [ 0.71150548 -0.01399435  0.00157293  0.70253942]
 [ 0.71413167 -0.01660221  0.0041535   0.69980217]] ...
TOD detector 6A = [0. 0. 0. 0. 0.] ...
TOD detector 6A flags = [0 0 0 0 0] ...
TOD detector 6A pointing = [[ 0.65890108  0.26624679 -0.27492183  0.64759555]
 [ 0.66234515  0.26487916 -0.2714774   0.64609439]
 [ 0.66577088  0.26350386 -0.26802568  0.64457512]
 [ 0.66917818  0.26212094 -0.26456677  0.64303779]
 [ 0.67256696  0.26073044 -0.26110078  0.64148243]] ...
TOD detector 6B = [0. 0. 0. 0. 0.] ...
TOD detector 6B flags = [0 0 0 0 0] ...
TOD detector 6B pointing = [[ 0.65417834 -0.27764851  0.26352011  0.6523183 ]
 [ 0.65564659 -0.28105089  0.26489422  0.64882123]
 [ 0.65709647 -0.28444573  0.26626067  0.64530622]
 [ 0.65852793 -0.28783293  0.26761942  0.64177334]
 [ 0.65994092 -0.2912124   0.26897045  0.63822271]] ...
```

Notice that the signal data for all detectors is zero. For simulated TOD classes, there is no data to "read". Instead, simulated timestreams are constructed and stored in the `tod.cache` member variable.

1.1.2 Low Resolution Example

Imagine the case of a satellite telescope with detector beams that are a 5 degrees FWHM. We'll use a healpix resolution of NSIDE = 32 (approximately 2 degrees) for this example. Let's use made-up angles for the spin and precession angles of 40 and 50 degrees, respectively:

$$\alpha = 50^\circ \tag{1}$$
$$\beta = 40^\circ \tag{2}$$
$$\omega_\alpha = \text{precession rate} \tag{3}$$
$$\omega_\beta = \text{spin rate} \tag{4}$$
$$\tag{5}$$

When computing the precession rate, we want the precession motion to be slow enough so that the speed of the boresight on the sky does not vary enough to change our effective beams. The speed variation on the sky due to precession is

$$v_{\mathrm{min}} = \beta \cdot \omega_\beta - \alpha \cdot \omega_\alpha \tag{6}$$
$$v_{\mathrm{max}} = \beta \cdot \omega_\beta + \alpha \cdot \omega_\alpha \tag{7}$$
$$v_{\mathrm{diff}} = v_{\mathrm{max}} - v_{\mathrm{min}} = 2\alpha\omega_\alpha \tag{8}$$

This change, integrated over a sample, must be a small fraction (here called "X") of the beam FWHM:

$$\frac{2\alpha\omega_\alpha}{f_{\mathrm{sample}}} = X \cdot \mathrm{FWHM} \tag{9}$$
$$f_{\mathrm{sample}} = \frac{2\alpha\omega_\alpha}{X \cdot \mathrm{FWHM}} \tag{10}$$
$$\tag{11}$$

The speed of the boresight on the sky in degrees per second due to the spin axis motion is

$$v_{bore} = \alpha \cdot \omega_\alpha \cdot \frac{1}{60}$$

If we want to have 3 hits per pixel with two degree pixels (2/3 degree per second), this gives us

$$v_{bore} = \frac{2}{3} = \alpha \cdot \omega_\alpha \cdot \frac{1}{60} \tag{12}$$
$$\omega_\alpha = \frac{60}{3 \cdot \alpha} = 0.8 \text{ RPM} \tag{13}$$

If we assume five percent for our "X" fraction above, then this in turn forces our sample rate to be:

$$f_{\mathrm{sample}} = \frac{2 \cdot 50 \cdot 0.8}{0.05 \cdot 3.0 \cdot 60} = 8.9 \text{ Hz}$$

The precession rate is slower than the spin rate. The spin rate above corresponds to a period of 1.25 minutes. We choose a precession period 20 times longer (25 minutes). We will assume a very simple satellite motion where the precession axis slews continuously in the anti-sun direction.

NOTE: For the serial example in the next cell, we have artificially decreased the sample rate to 0.5 Hz and the resolution to NSIDE=16. This is so that this small example fits into reasonable RAM while still covering the sky. See the parallel notebook for an example with proper sampling.

In [11]:
```
# Scan parameters

alpha = 50.0       # precession opening angle, degrees
beta = 45.0        # spin opening angle, degrees
p_alpha = 25.0     # precession period, minutes
```

```
p_beta = 1.25     # spin period, minutes
samplerate = 0.5  # sample rate, Hz
hwprpm = 5.0      # HWP rotation in RPM
nside = 16        # Healpix NSIDE
```

In [12]:
```
# We will use one observation per day, with no gaps in between, and
# run for one year.

obs_samples = int(24 * 3600.0 * samplerate) - 1
nobs = 366
```

In [13]:
```
# Slew the precession axis so that it completes one circle

deg_per_day = 360.0 / nobs
```

In [14]:
```
# Create distributed data

data = toast.Data(comm)
```

In [15]:
```
# Append observations

for ob in range(nobs):
    obsname = "{:03d}".format(ob)
    obsfirst = ob * (obs_samples + 1)
    obsstart = 24 * 3600.0
    tod = TODSatellite(
        comm.comm_group,
        detquat,
        obs_samples,
        firstsamp=obsfirst,
        firsttime=obsstart,
        rate=samplerate,
        spinperiod=p_beta,
        spinangle=beta,
        precperiod=p_alpha,
        precangle=alpha,
        coord="E",
        hwprpm=hwprpm
    )
    qprec = np.empty(4 * tod.local_samples[1], dtype=np.float64).reshape((-1, 4))
    slew_precession_axis(
        qprec,
        firstsamp=obsfirst,
        samplerate=samplerate,
        degday=deg_per_day,
    )
    tod.set_prec_axis(qprec=qprec)
    obs = dict()
```

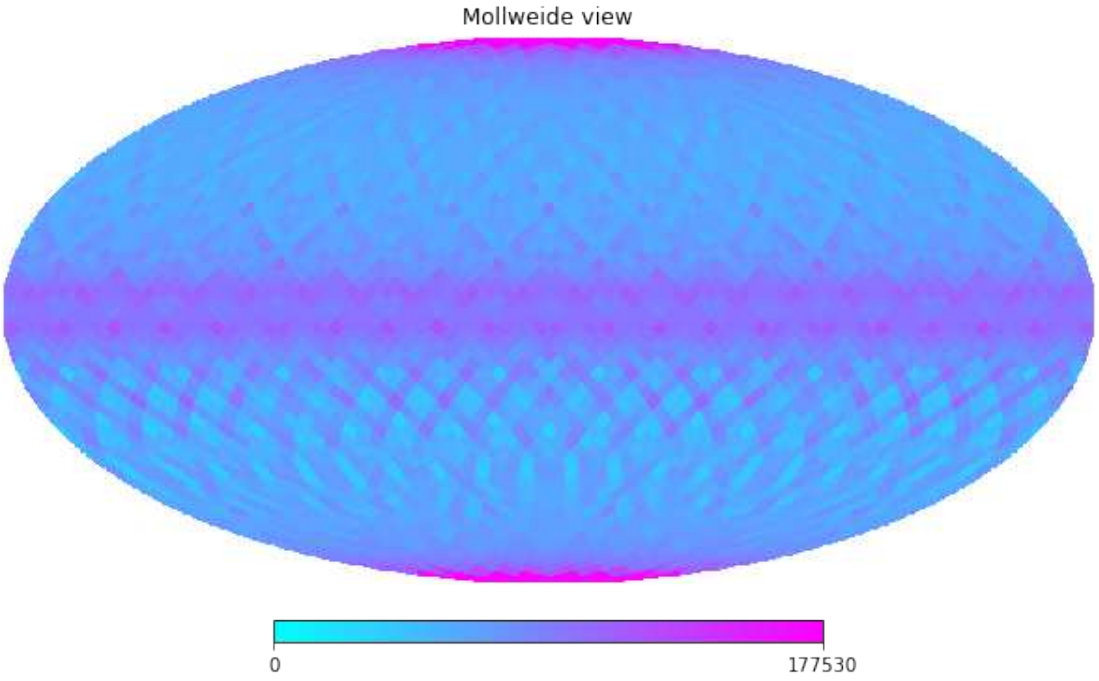
```
            obs["tod"] = tod
            data.obs.append(obs)
```

Now that we have simulated our scan strategy, we can make a simple hit map to visualize this

```
In [16]: from toast.todmap import (
             get_submaps_nested,
             OpPointingHpix,
             OpAccumDiag
         )
         from toast.map import (
             DistPixels
         )

In [17]: # Make a simple pointing matrix

         pointing = OpPointingHpix(nside=nside, nest=True, mode="IQU")
         pointing.exec(data)

In [18]: # Compute the locally hit pixels

         localpix, localsm, subnpix = get_submaps_nested(data, nside)

In [19]: # Construct a distributed map to store the hit map

         npix = 12 * nside**2

         hits = DistPixels(
             comm=data.comm.comm_world,
             size=npix,
             nnz=1,
             dtype=np.int64,
             submap=subnpix,
             local=localsm,
         )
         hits.data.fill(0)

In [20]: # Accumulate the hit map locally

         build_hits = OpAccumDiag(hits=hits)
         build_hits.exec(data)

In [21]: # Reduce the map across processes (a No-op in this case)

         hits.allreduce()

In [22]: # Write out the map

         hitsfile = "simscan_satellite_hits.fits"
         hits.write_healpix_fits(hitsfile)
```

In [23]:
```
# Plot the map.  If we were running on multiple processes, then
# only rank zero would do this...

import healpy as hp
import matplotlib.pyplot as plt
%matplotlib inline

hitdata = hp.read_map(hitsfile, nest=True)
hp.mollview(hitdata, xsize=800, nest=True, cmap="cool", min=0)
plt.show()
```

```
NSIDE = 16
ORDERING = NESTED in fits file
INDXSCHM = IMPLICIT
```



simscan_satellite_mpi

November 1, 2019

1 Simulated Satellite Scan Strategies - MPI Example

In [1]:
```
# Are you using a special reservation for a workshop?
# If so, set it here:
nersc_reservation = None

# Load common tools for all lessons
import sys
sys.path.insert(0, "..")
from lesson_tools import (
    check_nersc,
)
nersc_host, nersc_repo, nersc_resv = check_nersc(reservation=nersc_reservation)

# Capture C++ output in the jupyter cells
%reload_ext wurlitzer
```

```
Running on NERSC machine 'cori'
  with access to repos: mp107
Using default repo mp107
```

In [2]:
```
%%writefile simscan_satellite_mpi.py

import toast
from toast.mpi import MPI

# Load common tools for all lessons
import sys
sys.path.insert(0, "..")
from lesson_tools import (
    fake_focalplane
)

import numpy as np
import healpy as hp
import matplotlib.pyplot as plt
```

```
from toast.todmap import (
    slew_precession_axis,
    TODSatellite,
    get_submaps_nested,
    OpPointingHpix,
    OpAccumDiag
)
from toast.map import (
    DistPixels
)

env = toast.Environment.get()

# We have many small observations, so we should use a small
# group size.  Here we choose a group size of one process.

comm = toast.Comm(world=MPI.COMM_WORLD, groupsize=1)
if comm.world_rank == 0:
    print(env)

# Create our fake focalplane

fp = fake_focalplane()

detnames = list(sorted(fp.keys()))
detquat = {x: fp[x]["quat"] for x in detnames}

# Scan parameters

alpha = 50.0      # precession opening angle, degrees
beta = 45.0       # spin opening angle, degrees
p_alpha = 25.0    # precession period, minutes
p_beta = 1.25     # spin period, minutes
samplerate = 8.9  # sample rate, Hz
hwprpm = 5.0      # HWP rotation in RPM
nside = 32        # Healpix NSIDE

# We will use one observation per day, with no gaps in between, and
# run for one year.

obs_samples = int(24 * 3600.0 * samplerate) - 1
nobs = 366

# Slew the precession axis so that it completes one circle

deg_per_day = 360.0 / nobs
```

```
# Create distributed data

data = toast.Data(comm)

# Append observations

for ob in range(nobs):
    # Am I in the group that has this observation?
    if (ob % comm.ngroups) != comm.group:
        # nope...
        continue
    obsname = "{:03d}".format(ob)
    obsfirst = ob * (obs_samples + 1)
    obsstart = 24 * 3600.0
    tod = TODSatellite(
        comm.comm_group,
        detquat,
        obs_samples,
        firstsamp=obsfirst,
        firsttime=obsstart,
        rate=samplerate,
        spinperiod=p_beta,
        spinangle=beta,
        precperiod=p_alpha,
        precangle=alpha,
        coord="E",
        hwprpm=hwprpm
    )
    qprec = np.empty(4 * tod.local_samples[1], dtype=np.float64).reshape((-1, 4))
    slew_precession_axis(
        qprec,
        firstsamp=obsfirst,
        samplerate=samplerate,
        degday=deg_per_day,
    )
    tod.set_prec_axis(qprec=qprec)
    obs = dict()
    obs["tod"] = tod
    data.obs.append(obs)

# Make a simple pointing matrix

pointing = OpPointingHpix(nside=nside, nest=True, mode="IQU")
pointing.exec(data)

# Compute the locally hit pixels

localpix, localsm, subnpix = get_submaps_nested(data, nside)
```

```
        # Construct a distributed map to store the hit map

        npix = 12 * nside**2

        hits = DistPixels(
            comm=data.comm.comm_world,
            size=npix,
            nnz=1,
            dtype=np.int64,
            submap=subnpix,
            local=localsm,
        )
        hits.data.fill(0)

        # Accumulate the hit map locally

        build_hits = OpAccumDiag(hits=hits)
        build_hits.exec(data)

        # Reduce the map across processes (a No-op in this case)

        hits.allreduce()

        # Write out the map

        hitsfile = "simscan_satellite_hits_mpi.fits"
        hits.write_healpix_fits(hitsfile)

        # Plot the map.

        if comm.world_rank == 0:
            hitdata = hp.read_map(hitsfile, nest=True)
            hp.mollview(hitdata, xsize=800, nest=True, cmap="cool", min=0)
            plt.savefig("{}.png".format(hitsfile))
            plt.close()
```

```
Writing simscan_satellite_mpi.py
```

```
In [3]: import subprocess as sp

        command = "python simscan_satellite_mpi.py"
        runstr = None

        if nersc_host is not None:
            runstr = "export OMP_NUM_THREADS=4; srun -N 2 -C haswell -n 32 -c 4 --cpu_bind=cor
            if nersc_resv is not None:
```

```
            runstr = "{} --reservation {}".format(runstr, nersc_resv)
        else:
            # Just use mpirun
            runstr = "mpirun -np 4"

        runcom = "{} {}".format(runstr, command)
        print(runcom, flush=True)
        sp.check_call(runcom, stderr=sp.STDOUT, shell=True)
```

```
export OMP_NUM_THREADS=4; srun -N 2 -C haswell -n 32 -c 4 --cpu_bind=cores -t 00:05:00 python s
Launched in background. Redirecting stdin to /dev/null
ModuleCmd_Load.c(244):ERROR:105: Unable to locate a modulefile for 'altd'
ModuleCmd_Load.c(244):ERROR:105: Unable to locate a modulefile for 'darshan'
srun: job 25265611 queued and waiting for resources
srun: job 25265611 has been allocated resources
<toast.Environment
  Source code version = 2.3.1.dev1428
  Logging level = INFO
  Handling enabled for 0 signals:
  Max threads = 4
  MPI build enabled
  MPI runtime enabled
>
NSIDE = 32
ORDERING = NESTED in fits file
INDXSCHM = IMPLICIT


Out[3]: 0
```

simsky_mapdomain

November 1, 2019

1 Simulated Sky Signal in map domain

This lesson is about simulating the input sky signal using PySM 3.

1.1 PySM 3

If you used `PySM` in the past, you most probably used `PySM 2` from https://github.com/bthorne93/PySM_public.

`PySM 3` is a rewrite of `PySM` which offers all the same functionality and the same models of `PySM 2` but is focused on:

- improving performance using just-in-time compilation and multi-threading with `numba`
- lowering memory requirements by reworking the underlying algorithms
- improved capability of running in parallel with MPI

It is available from https://github.com/healpy/pysm, it is still missing a few models and the documentation but is already integrated into TOAST to overcame the strong performance limits of `PySM 2`.

If anyone is interested in learning more about PySM 3, check the PySM 3 tutorial, we can work through this during the hack day.

1.2 PySMSky

The lower level TOAST class is `PySMSky`, it performs the following operations: * initialize `PySM` with the input sky configuration * loop through all channels and for each calls `PySM` to generate the sky emission at all frequencies in the bandpass and integrate

```
In [1]: # Load common tools for all lessons
        import sys
        sys.path.insert(0, "..")
        from lesson_tools import (
            fake_focalplane
        )

        # Capture C++ output in the jupyter cells
        %reload_ext wurlitzer
```

simsky_timedomain

November 1, 2019

1 Simulated Sky Signal in time domain

In this lesson we will use the TOAST Operator `OpSimPySM` to create timestreams for an instrument given a sky model.

In [1]:
```
# Load common tools for all lessons
import sys
sys.path.insert(0, "..")
from lesson_tools import (
    fake_focalplane
)

# Capture C++ output in the jupyter cells
%reload_ext wurlitzer
```

In [2]:
```
import toast
import healpy as hp
import numpy as np
```

In [3]:
```
env = toast.Environment.get()
env.set_log_level("DEBUG")
```

1.1 Scanning strategy

Before being able to scan a map into a timestream we need to define a scanning strategy and get pointing information for each channel.

We use the same **satellite** scanning used in lesson 2 about scanning strategies, see the `02_Simulated_Scan_Strategies/simscan_satellite.ipynb` for more details.

In [4]:
```
focal_plane = fake_focalplane()
```

In [5]:
```
focal_plane.keys()
```

Out[5]:
```
dict_keys(['0A', '0B', '1A', '1B', '2A', '2B', '3A', '3B', '4A', '4B', '5A', '5B', '6A
```

In [6]:
```
focal_plane["0A"]["fwhm_arcmin"]
```

Out[6]:
```
30
```

In [7]:
```
# Scan parameters

alpha = 50.0       # precession opening angle, degrees
beta = 45.0        # spin opening angle, degrees
p_alpha = 25.0     # precession period, minutes
p_beta = 1.25      # spin period, minutes
samplerate = 0.5   # sample rate, Hz
hwprpm = 5.0       # HWP rotation in RPM
nside = 64         # Healpix NSIDE

# We will use one observation per day, with no gaps in between, and
# run for one year.

obs_samples = int(24 * 3600.0 * samplerate) - 1
nobs = 366

# Slew the precession axis so that it completes one circle

deg_per_day = 360.0 / nobs
```

In [8]:
```
from toast.todmap import TODSatellite, slew_precession_axis
```

In [9]:
```
detquat = {ch: focal_plane[ch]["quat"] for ch in focal_plane}
```

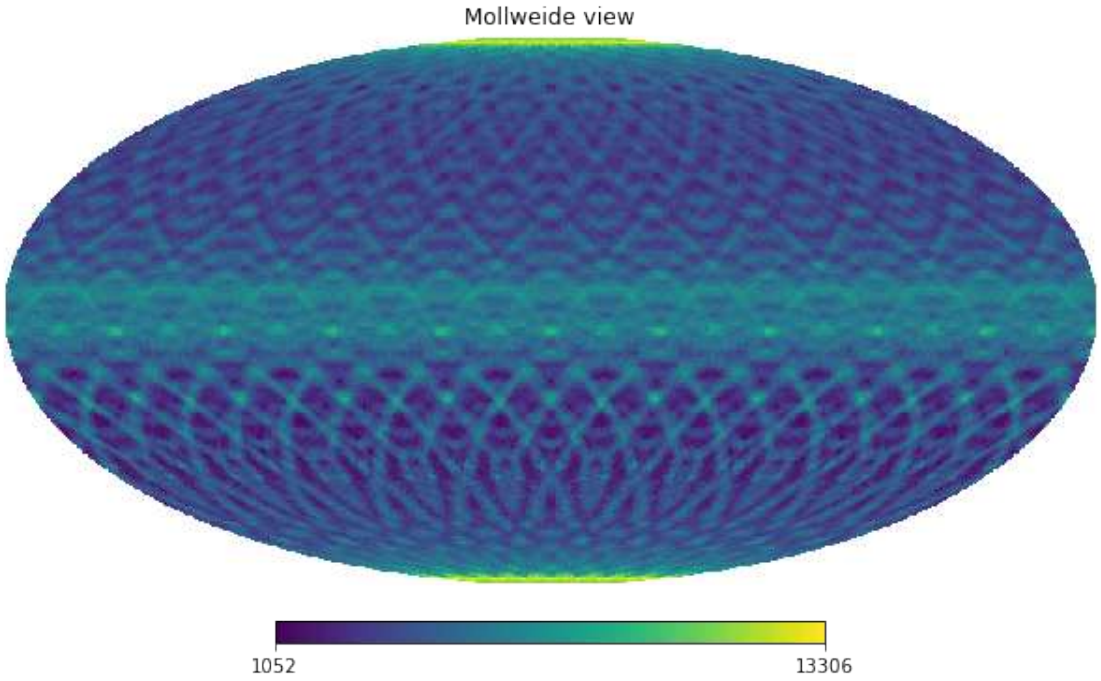
In [10]:
```
# Create distributed data

comm = toast.Comm()
data = toast.Data(comm)

# Append observations

for ob in range(nobs):
    obsname = "{:03d}".format(ob)
    obsfirst = ob * (obs_samples + 1)
    obsstart = 24 * 3600.0
    tod = TODSatellite(
        comm.comm_group,
        detquat,
        obs_samples,
        firstsamp=obsfirst,
        firsttime=obsstart,
        rate=samplerate,
        spinperiod=p_beta,
        spinangle=beta,
        precperiod=p_alpha,
        precangle=alpha,
        coord="E",
        hwprpm=hwprpm
```

```
            )
            qprec = np.empty(4 * tod.local_samples[1], dtype=np.float64).reshape((-1, 4))
            slew_precession_axis(
                qprec,
                firstsamp=obsfirst,
                samplerate=samplerate,
                degday=deg_per_day,
            )
            tod.set_prec_axis(qprec=qprec)
            obs = dict()
            obs["tod"] = tod
            data.obs.append(obs)
```

In [11]:
```
from toast.todmap import (
    get_submaps_nested,
    OpPointingHpix,
    OpAccumDiag
)
from toast.map import (
    DistPixels
)

# Make a simple pointing matrix

pointing = OpPointingHpix(nside=nside, nest=True, mode="IQU")
pointing.exec(data)

# Compute the locally hit pixels

localpix, localsm, subnpix = get_submaps_nested(data, nside)

# Construct a distributed map to store the hit map

npix = 12 * nside**2

hits = DistPixels(
    comm=data.comm.comm_world,
    size=npix,
    nnz=1,
    dtype=np.int64,
    submap=subnpix,
    local=localsm,
)
hits.data.fill(0)

# Accumulate the hit map locally

build_hits = OpAccumDiag(hits=hits)
```

```
build_hits.exec(data)

# Reduce the map across processes (a No-op in this case)

hits.allreduce()
```

```
In [12]: %matplotlib inline
         hp.mollview(hits.data.flatten(), nest=True)
```



1.2 Define PySM parameters and instrument bandpasses

Then we define the sky model parameters, choosing the desired set of `PySM` models and then we specify the band center and the bandwidth for a top-hat bandpass. Currently top-hat bandpasses are the only type supported by the operator, in the future we will implement arbitrary bandpasses.

Then bandpass parameters can be added directly to the `focal_plane` dictionary:

```
In [13]: for ch in focal_plane:
             focal_plane[ch]["bandcenter_ghz"] = 70
             focal_plane[ch]["bandwidth_ghz"] = 10
             focal_plane[ch]["fwhm"] = 60*2

In [14]: pysm_sky_config = ["s1", "f1", "a1", "d1"]
```

siminst

November 1, 2019

1 Simulated Instrument Signal

In this lesson we cover three TOAST operators for simulating noise and systematics: * `OpSimNoise` for simulating instrumental noise * `OpSimAtmosphere` for simulating atmospheric noise * `OpSimSSS` for simulating scan-synchronous signal (typically ground pick-up) We also introduce `TODGround`, a synthetic `TOD` class that simulates constant elevation scanning.

In [1]:
```
# Are you using a special reservation for a workshop?
# If so, set it here:
nersc_reservation = None

# Load common tools for all lessons
import sys
sys.path.insert(0, "..")
from lesson_tools import (
    check_nersc,
    fake_focalplane
)
nersc_host, nersc_repo, nersc_resv = check_nersc(reservation=nersc_reservation)

# Capture C++ output in the jupyter cells
%reload_ext wurlitzer
```

```
Running on NERSC machine 'cori'
  with access to repos: mp107
Using default repo mp107
```

1.1 Generating a TOAST data object with one constant elevation scan

We begin by generating an observing schedule containing a single observation. We define one circular patch at (40°, -40°) RA, DEC with a 10° radius.

In [2]:
```
! toast_ground_schedule.py \
    --site-lat "-22.958064" \
    --site-lon "-67.786222" \
    --site-alt 5200 \
    --site-name Atacama \
```

```
    --telescope LAT \
    --start "2020-01-01 04:00:00" \
    --stop "2020-01-01 05:00:00" \
    --patch-coord C \
    --patch small_patch,1,40,-40,10 \
    --ces-max-time 86400 \
    --out schedule.txt

! cat schedule.txt
```

```
TOAST INFO: Adding patch "small_patch"
TOAST INFO: Center-and-width format
TOAST INFO: Global timer: toast_ground_schedule:  0.03 seconds (1 calls)
#Site            Telescope        Latitude [deg] Longitude [deg]   Elevation [m]
 Atacama         LAT                     -22.958         -67.786          5200.0
#Start time UTC       Stop time UTC        Start MJD      Stop MJD       Patch name
 2020-01-01 04:00:00  2020-01-01 04:53:00    58849.166667   58849.203472 small_patch
```

For simulating atmospheric noise, we'll also need a TOAST weather file, which contains the historic distributions of temperature, wind, pressure and water vapor at a fixed site. The file is arranged by month and hour of the day to capture seasonal and diurnal variation in these parameters. Here we fetch the weather file for Atacama. There is a another file for Pole.

```
In [3]: ! [[ ! -e weather_Atacama.fits ]] && wget http://portal.nersc.gov/project/cmb/toast_da
```

1.1.1 TODGround object

Typical TOAST pipelines like toast_ground_sim.py create synthetic observations by loading the schedule from file and then creating instances of TODGround according to the observing schedule. This is most easily accomplished with toast.pipeline_tools.

```
In [4]: import toast
        import toast.pipeline_tools
        from toast.mpi import MPI

        import numpy as np
        import matplotlib.pyplot as plt

        mpiworld, procs, rank = toast.mpi.get_world()
        comm = toast.mpi.Comm(mpiworld)

        # A pipeline would create the args object with argparse

        class args:
            split_schedule = None
            schedule = "schedule.txt"
            sort_schedule = False  # Matters for parallelization
            weather = "weather_Atacama.fits"
```

```
    sample_rate = 10  # Hz
    scan_rate = 1.0  # deg / s
    # Use an artifially low acceleration to show the turn-arounds better
    scan_accel = 0.1  # deg / s^2
    hwp_rpm = None
    hwp_step_deg = None
    hwp_step_time_s = None
    fov = 3.0  # Field-of-view in degrees

# Create a fake focalplane, we could also load one from file.
# The Focalplane class interprets the focalplane dictionary
# created by fake_focalplane() but it can also load the information
# from file.

focalplane = toast.pipeline_tools.Focalplane(
    fake_focalplane(fov=args.fov), sample_rate=args.sample_rate
)

# Load the observing schedule, append weather and focalplane to it

schedules = toast.pipeline_tools.load_schedule(args, comm)
toast.pipeline_tools.load_weather(args, comm, schedules)
# There could be more than one observing schedule, but not this time
schedule = schedules[0]
schedule.telescope.focalplane = focalplane

# Create a TODGround object based on the only entry in the schedule

ces = schedule.ceslist[0]  # normally we would loop over entries
totsamples = int((ces.stop_time - ces.start_time) * args.sample_rate)

if comm.comm_group is not None:
    # Available detectors should be split between processes in the group
    ndetrank = comm.comm_group.size
else:
    ndetrank = 1

telescope = schedule.telescope  # shorthand

tod = toast.todmap.TODGround(
    comm.comm_group,
    telescope.focalplane.detquats,
    totsamples,
    detranks=ndetrank,
    firsttime=ces.start_time,
    rate=args.sample_rate,
    site_lon=telescope.site.lon,
    site_lat=telescope.site.lat,
```

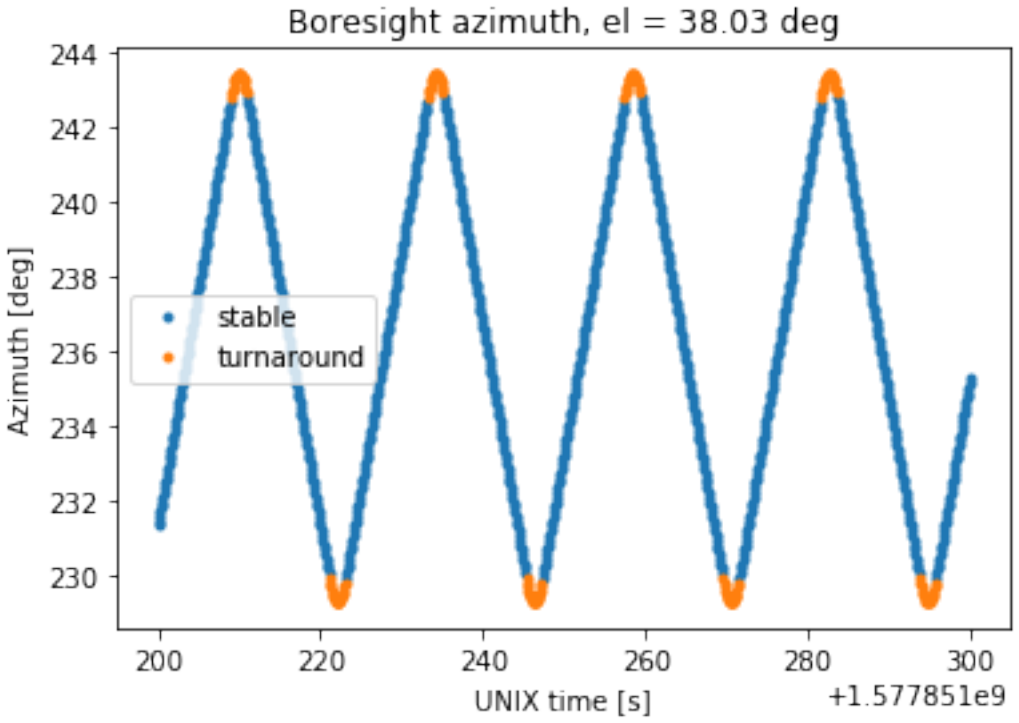
```
            site_alt=telescope.site.alt,
            azmin=ces.azmin,
            azmax=ces.azmax,
            el=ces.el,
            scanrate=args.scan_rate,
            scan_accel=args.scan_accel,
            #CES_start=None,
            #CES_stop=None,
            #sun_angle_min=args.sun_angle_min,
            #coord=args.coord,
            #sampsizes=None,
            #report_timing=args.debug,
            hwprpm=args.hwp_rpm,
            hwpstep=args.hwp_step_deg,
            hwpsteptime=args.hwp_step_time_s,
        )
```

```
TOAST INFO: Load 1 (sub)scans in schedule.txt:  0.00 seconds (1 calls)
TOAST INFO: Loading schedule(s):  0.00 seconds (1 calls)
TOAST INFO: Load weather_Atacama.fits:  0.30 seconds (1 calls)
TOAST INFO: TODGround: simulate scan:  0.01 seconds (1 calls)
TOAST INFO: TODGround: list valid intervals:  0.00 seconds (1 calls)
TOAST INFO: TODGround: call base class constructor:  0.00 seconds (1 calls)
TOAST INFO: TODGround: translate scan pointing:  0.03 seconds (1 calls)
```

The `TODGround` objects have capabilities that regular TOD objects do not. Specifically, they can produce detector and boresight pointing in both celestial and horizontal coordinate systems. Here we plot the boresight azimuth and identify the stable science scans.

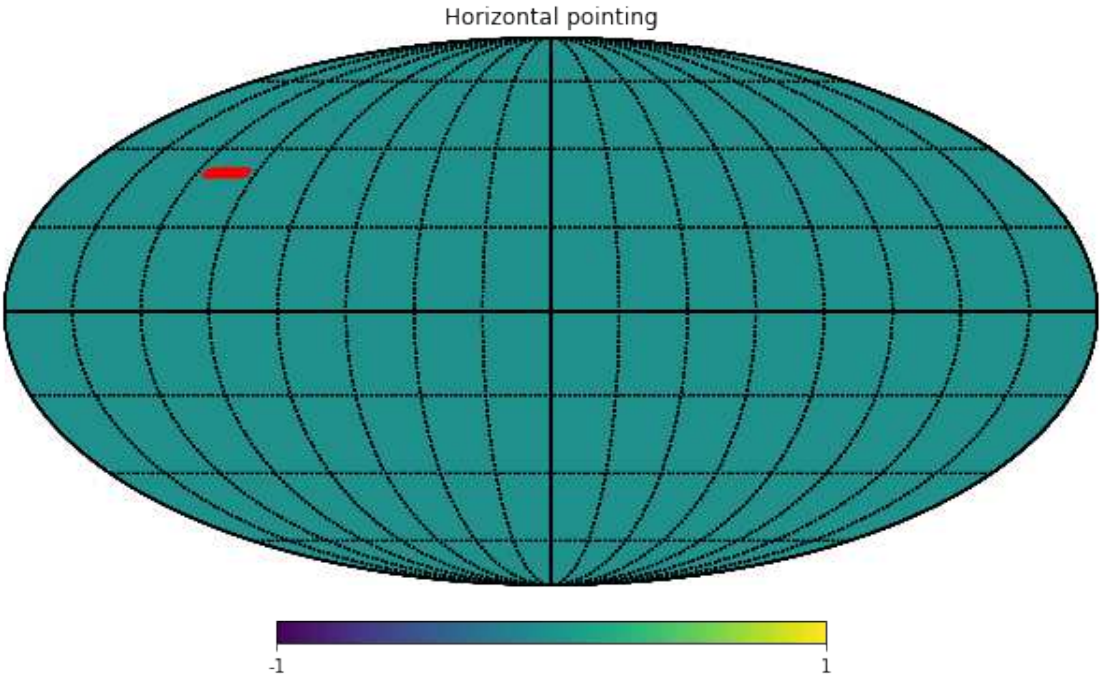
```
In [5]: import matplotlib.pyplot as plt
        %matplotlib inline

        ind = slice(0, 1000)
        times = tod.local_times()[ind]
        cflags = tod.local_common_flags()[ind]
        az = tod.read_boresight_az()[ind]

        turnaround = cflags & tod.TURNAROUND != 0
        stable = np.logical_not(turnaround)

        plt.plot(times[stable], np.degrees(az[stable]), '.', label="stable")
        plt.plot(times[turnaround], np.degrees(az[turnaround]), '.', label="turnaround")
        ax = plt.gca()
        ax.set_xlabel("UNIX time [s]")
        ax.set_ylabel("Azimuth [deg]")
        ax.set_title("Boresight azimuth, el = {} deg".format(np.degrees(tod._el)))
        plt.legend()
```

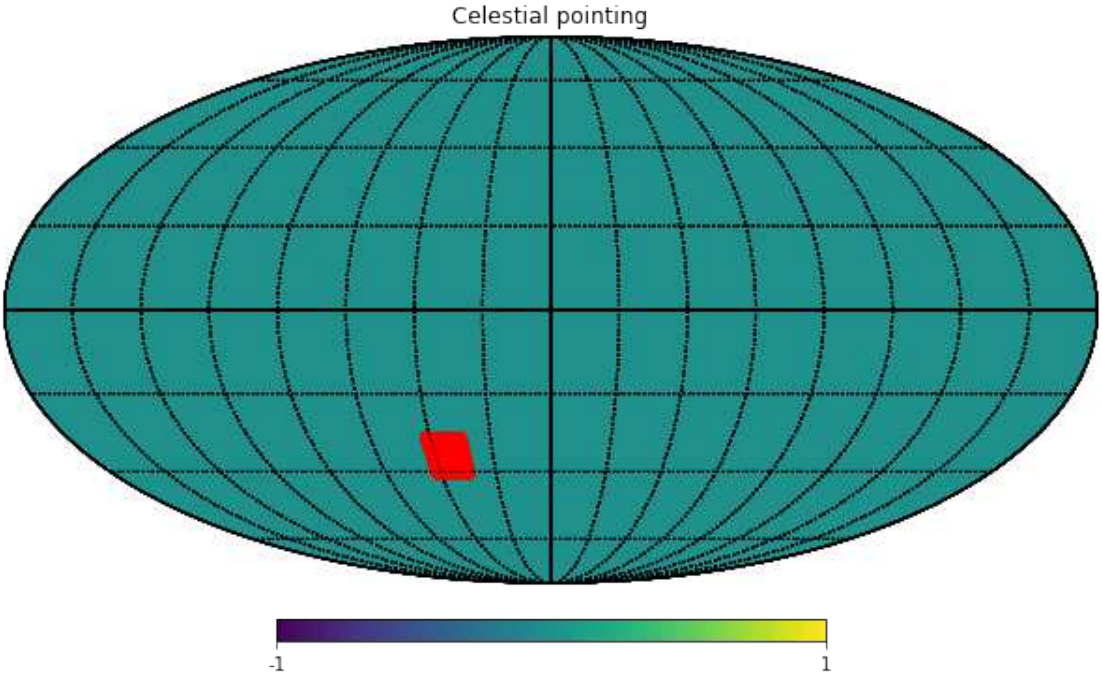
```
Out[5]: <matplotlib.legend.Legend at 0x2aaaf41b0668>
```



Here we get the horizontal and celestial pointing and plot the location of every 10th sample

```
In [6]: import healpy

        healpy.mollview(np.zeros(12), title="Horizontal pointing")
        for det in tod.local_dets:
            quat_azel = tod.read_pntg(det, azel=True)[::10]
            az, el = toast.qarray.to_position(quat_azel)
            healpy.projplot(az, el, 'r-')
            healpy.graticule(22.5, verbose=False)

        healpy.mollview(np.zeros(12), title="Celestial pointing")
        for det in tod.local_dets:
            quat_radec = tod.read_pntg(det)[::10]
            ra, dec = toast.qarray.to_position(quat_radec)
            healpy.projplot(ra, dec, 'r.', ms=1)
            healpy.graticule(22.5, verbose=False)
```


Horizontal pointing
-1
1


Celestial pointing
-1
1

1.1.2 TOAST data

In [7]:
```
# Now embed the TOD in an observation dictionary and add other necessary metadata

obs = {}
obs["name"] = "CES-{}-{}-{}-{}-{}".format(
    telescope.site.name, telescope.name, ces.name, ces.scan, ces.subscan
)
obs["tod"] = tod
obs["baselines"] = None
obs["noise"] = telescope.focalplane.noise
obs["id"] = int(ces.mjdstart * 10000)
obs["intervals"] = tod.subscans
obs["site"] = telescope.site
obs["site_name"] = telescope.site.name
obs["site_id"] = telescope.site.id
obs["altitude"] = telescope.site.alt
obs["weather"] = telescope.site.weather
obs["telescope"] = telescope
obs["telescope_name"] = telescope.name
obs["telescope_id"] = telescope.id
obs["focalplane"] = telescope.focalplane.detector_data
obs["fpradius"] = telescope.focalplane.radius
obs["start_time"] = ces.start_time
obs["season"] = ces.season
obs["date"] = ces.start_date
obs["MJD"] = ces.mjdstart
obs["rising"] = ces.rising
obs["mindist_sun"] = ces.mindist_sun
obs["mindist_moon"] = ces.mindist_moon
obs["el_sun"] = ces.el_sun
```

In [8]:
```
for key, value in obs.items():
    if key == "intervals":
        print("intervals = [{} ... {}] ({} in total)".format(value[0], value[-1], len(v
    else:
        print("{} = {}".format(key, value))
```

```
name = CES-Atacama-LAT-small_patch-0-0
tod = <TODGround
  14 total detectors and 31800 total samples
  Using MPI communicator None
    In grid dimensions 1 sample ranks x 1 detranks
  Process at (0, 0) in grid has data for:
    Samples 0 - 31799 (inclusive)
    Detectors:
      0A
      0B
```

```
      1A
      1B
      2A
      2B
      3A
      3B
      4A
      4B
      5A
      5B
      6A
      6B
    Cache contains 2575800 bytes
>
baselines = None
noise = <toast.tod.sim_noise.AnalyticNoise object at 0x2aaaf5228e10>
id = 588491666
intervals = [<Interval 1577851199.0 - 1577851209.0 (-10 - 89)> ... <Interval 1577854369.1 - 157
site = (Site 'Atacama' : ID = 0, lon = -67.786, lat = -22.958, alt = 5200.0 m, weather = (Weath
site_name = Atacama
site_id = 0
altitude = 5200.0
weather = (Weather : 'weather_Atacama.fits', site = 0, time = None, year = None, month = None,
telescope = (Telescope 'LAT' : ID = 225, Site = (Site 'Atacama' : ID = 0, lon = -67.786, lat =
telescope_name = LAT
telescope_id = 225
focalplane = {'0A': {'quat': array([0.        , 0.        , 0.38268343, 0.92387953]), 'epsilon
fpradius = 1.1558552389176189
start_time = 1577851200.0
season = 2020
date = 2020-01-01
MJD = 58849.166667
rising = False
mindist_sun = 88.71160018273949
mindist_moon = 47.56306383473445
el_sun = -43.25
```

```
In [9]: data = toast.Data(comm)
        data.obs.append(obs)
```

1.2 Simulating instrument noise

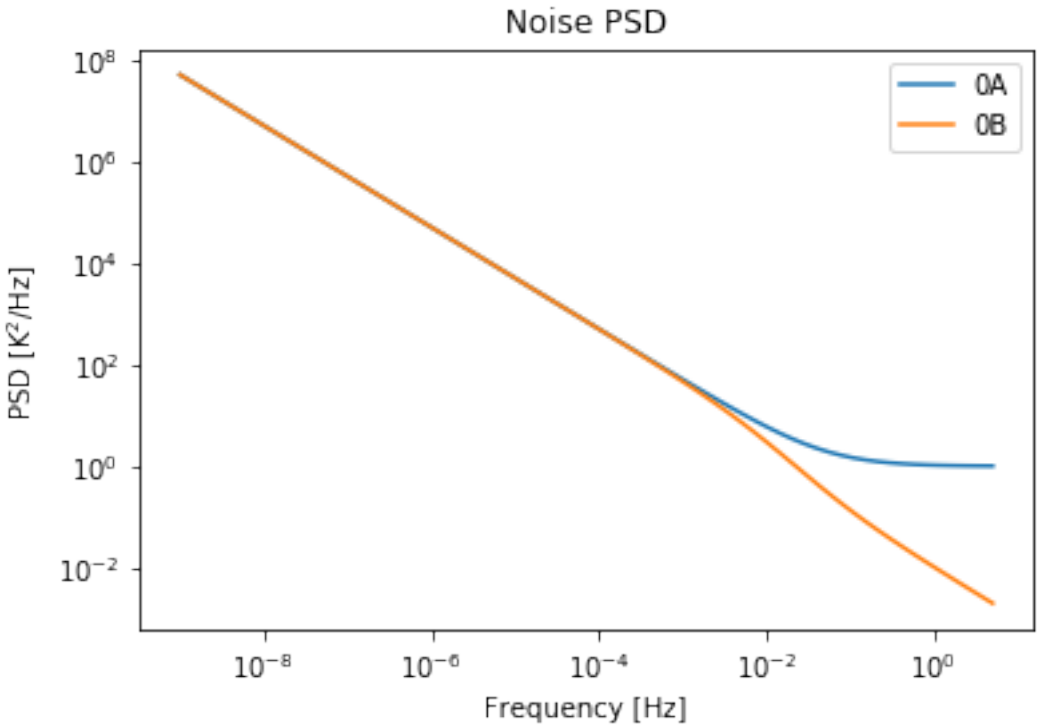
Instrumental noise simulation is based on the TOAST Noise object appended to each observation.
src/toast/tod/sim_det_noise.py

```
In [10]: obs = data.obs[0]
         noise = obs["noise"]
```

```
print("detectors:", noise.detectors)

# Get and plot noise PSD for det1
det1 = noise.detectors[0]
freq1 = noise.freq(det1)
psd1 = noise.psd(det1)
plt.loglog(freq1, psd1, label=det1)

# get and plot det2 but manipulate the noise PSD to suppress high frequency noise
det2 = noise.detectors[1]
freq2 = noise.freq(det2)
psd2 = noise.psd(det2)
psd2 *= 1e-2 / (freq2 + 1e-2)
plt.loglog(freq2, psd2, label=det2)

ax = plt.gca()
ax.set_xlabel("Frequency [Hz]")
ax.set_ylabel("PSD [K$^2$/Hz]")
ax.set_title("Noise PSD")
plt.legend(loc="best")
```

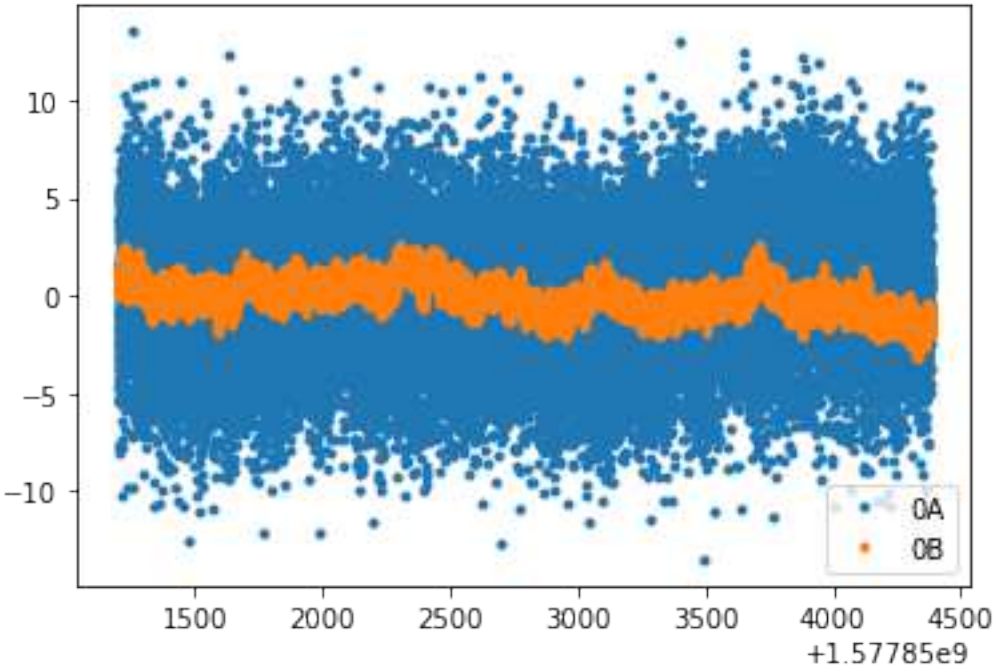
```
detectors: ['0A', '0B', '1A', '1B', '2A', '2B', '3A', '3B', '4A', '4B', '5A', '5B', '6A', '6B']
```

Out[10]: <matplotlib.legend.Legend at 0x2aaaf5228eb8>



```
In [11]: simnoise = toast.tod.OpSimNoise(out="noise", realization=0)
         toast.tod.OpCacheClear("noise").exec(data)
         simnoise.exec(data)
         tod = obs["tod"]
         times = tod.local_times()
         for det in [det1, det2]:
             simulated = tod.local_signal(det, "noise")
             plt.plot(times, simulated, ".", label=det)
         ax.set_xlabel("Unix time [s]")
         ax.set_ylabel("Noise [K]")
         ax.set_title("Simulated noise")
         plt.legend(loc="best")
```

```
Out[11]: <matplotlib.legend.Legend at 0x2aaaf4dd07f0>
```



By default the `Noise` object includes a diagonal mixing matrix and produces uncorrelated detector noise. Let's add a strongly correlated low frequency component to the `Noise` object.

```
In [12]: def print_mixmatrix(noise):
             print("{:10}key".format(""))
             print("{:10}".format("detector"), end="")
             for key in noise.keys:
                 print("{:>4}".format(key), end="")
             print()
             print("-" * 80)
             for det in noise.detectors:
```

```
                print("{:8} :".format(det), end="")
                for key in noise.keys:
                    weight = noise.weight(det, key)
                    if np.abs(weight) < 1e-10:
                        print("{:4}".format(""), end="")
                    else:
                        print("{:4}".format(noise.weight(det, key)), end="")
                print()

        print("Original mixing matrix")
        print_mixmatrix(noise)
```

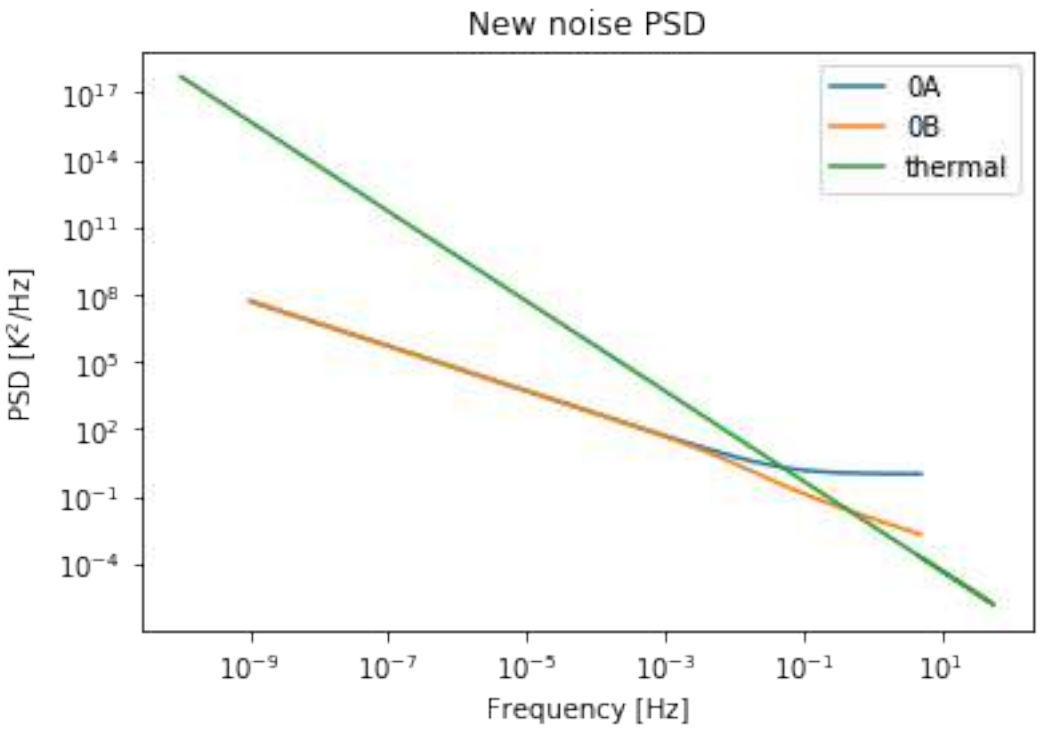
```
Original mixing matrix
          key
detector    0A   0B   1A   1B   2A   2B   3A   3B   4A   4B   5A   5B   6A   6B
--------------------------------------------------------------------------------
0A       :   1
0B       :        1
1A       :             1
1B       :                  1
2A       :                       1
2B       :                            1
3A       :                                 1
3B       :                                      1
4A       :                                           1
4B       :                                                1
5A       :                                                     1
5B       :                                                          1
6A       :                                                               1
6B       :                                                                    1
```

We will instantiate a new Noise object with all the original inputs and an additional thermal mode. We could also manipulate the existing noise object but this approach is more representative of how pipelines currently use `Noise`.

```
In [13]: detectors = []
         freqs = {}
         psds = {}
         mixmatrix = {}
         indices = {}
         for det in noise.detectors:
             detectors.append(det)
             freqs[det] = noise.freq(det).copy()
             psds[det] = noise.psd(det).copy()
             indices[det] = noise.index(det)
             # We do not just copy the old mixing matrix because it is not stored for the triv
             mixmatrix[det] = {}
```

```
            for key in noise.keys:
                weight = noise.weight(det, key)
                if weight != 0:
                    mixmatrix[det][key] = weight

        # Create a new PSD and add it to the Noise object inputs
        correlated_name = "thermal"
        freq = np.logspace(-10, 2)
        freq[freq < args.sample_rate / 2]
        freq[-1] = args.sample_rate / 2
        psd = freq ** -2 * 5e-3
        freqs[correlated_name] = freq
        psds[correlated_name] = psd
        indices[correlated_name] = 999999
        for det in noise.detectors:
            mixmatrix[det][correlated_name] = 1

        # Create a completely new Noise object
        new_noise = toast.tod.Noise(
            detectors=detectors, freqs=freqs, psds=psds, mixmatrix=mixmatrix, indices=indices
        )

        plt.figure()
        for key in [det1, det2, correlated_name]:
            freq = new_noise.freq(key)
            psd = new_noise.psd(key)
            plt.loglog(freq, psd, label=key)
        ax = plt.gca()
        ax.set_xlabel("Frequency [Hz]")
        ax.set_ylabel("PSD [K$^2$/Hz]")
        ax.set_title("New noise PSD")
        plt.legend(loc="best")
```

```
Out[13]: <matplotlib.legend.Legend at 0x2aaaf49b5278>
```

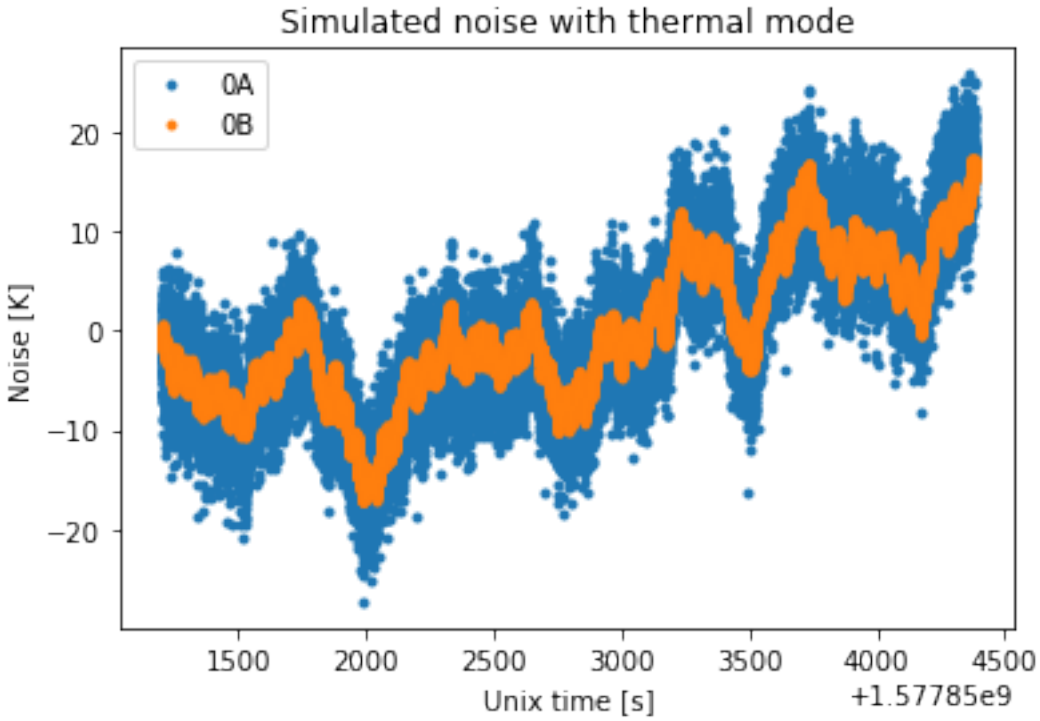



```
In [14]: print("\nNew mixing matrix")
         print_mixmatrix(new_noise)
```

```
New mixing matrix
          key
detector    0A  0B  1A  1B  2A  2B  3A  3B  4A  4B  5A  5B  6A  6Bthermal
--------------------------------------------------------------------------------
0A        :   1                                                         1
0B        :       1                                                     1
1A        :           1                                                 1
1B        :               1                                             1
2A        :                   1                                         1
2B        :                       1                                     1
3A        :                           1                                 1
3B        :                               1                             1
4A        :                                   1                         1
4B        :                                       1                     1
5A        :                                           1                 1
5B        :                                               1             1
6A        :                                                   1         1
6B        :                                                       1     1
```

We will now replace the `Noise` in the observation with the one that includes the correlated mode and rerun the simulation

```
In [15]: obs["noise"] = new_noise
         toast.tod.OpCacheClear("new_noise").exec(data)
         simnoise = toast.tod.OpSimNoise(out="new_noise", realization=0)
         simnoise.exec(data)

         plt.figure()
         tod = obs["tod"]
         times = tod.local_times()
         for det in [det1, det2]:
             simulated = tod.local_signal(det, "new_noise")
             plt.plot(times, simulated, ".", label=det)
         ax = plt.gca()
         ax.set_xlabel("Unix time [s]")
         ax.set_ylabel("Noise [K]")
         ax.set_title("Simulated noise with thermal mode")
         plt.legend(loc="best")

Out[15]: <matplotlib.legend.Legend at 0x2aaaf4c8df98>
```



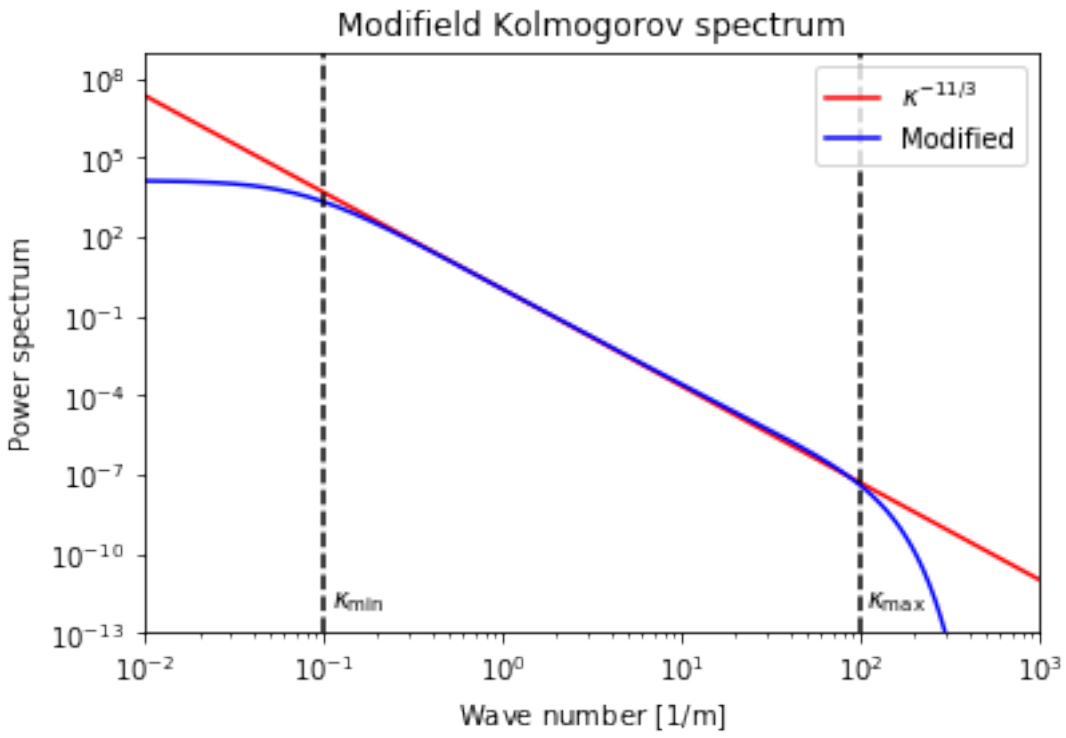
1.3 Atmospheric noise simulation

The atmospheric noise module in TOAST 1. defines a rectangular volume of atmosphere that contains all of the planned lines-of-sight factoring in the wind 2. compresses the volume by determining which elements are actually needed 3. builds the element-element covariance matrix matching the Errard. et al. model and instantiates it on the compressed volume 4. optionally caches the volume elements for future use 5. simulates the detector signal by performing line-of-sight integrals through the simulated volume while moving the volume with wind

src/toast/todmap/sim_det_atm.py

The atmospheric simulation is based on creating realization of the modified Kolmogorov spectrum (Andrews, L. C. 2004, Field Guide to Atmospheric Optics (SPIE), 112):

```
In [16]: dissipation_scale = .01  # in meters
         injection_scale = 10.  # in meters
         kappamin = 1 / injection_scale
         kappamax = 1 / dissipation_scale

         n = 1000000
         k = np.linspace(1e-2, kappamax * 10, n)

         kl = 0.9 / dissipation_scale
         k0 = 0.75 / injection_scale

         kkl = k / kl
         phi = (1 + 1.802 * kkl - .254 * (kkl) ** (7 / 6)) * np.exp(-kkl ** 2) * (k ** 2 + k0 *

         plt.figure()
         plt.loglog(k, k ** (-11 / 3), "r-", label="$\kappa^{-11/3}$")
         plt.loglog(k, phi, "b-", label="Modified")

         ax = plt.gca()
         for k, label in [(kappamin, "$\kappa_\mathrm{min}$"), (kappamax, "$\kappa_\mathrm{max
             ax.axvline(k, linestyle="--", color="black")
             plt.annotate(label, xy=[k * 1.08, 1e-12])

         ax.set_xlabel("Wave number [$1/\mathrm{m}$]")
         ax.set_ylabel("Power spectrum")
         ax.set_title("Modifield Kolmogorov spectrum")

         ax.set_xlim([1e-2, 1e3])
         ax.set_ylim([1e-13, 1e9])

         plt.legend(loc="upper right");
```



Here we run an example atmospheric simulation for two different injection scales. Be patient, they take about a minute to run.

```
In [17]: fig = plt.figure(figsize=[6 * 2, 4 * 3])
         name = "atmosphere"

         # Delete old atmosphere from disk.  We could also set cachedir=None
         ! rm -rf atm_cache*

         # Run the simulation with two different injection scales and plot the resulting TOD

         for iplot, lmax in enumerate([10, 300]):
             cachedir = "atm_cache_{}".format(lmax)
             # Wipe old atmospheric signal
             toast.tod.OpCacheClear(name).exec(data)

             atmsim = toast.todmap.OpSimAtmosphere(
                 out=name,
                 realization=0,  # Each MC will have a different realization
                 zmax=1000,  # maximum altitude to integrate
                 lmin_center=0.01,  # Dissipation scale
                 lmin_sigma=0,
                 lmax_center=lmax,  # Injection scale
                 lmax_sigma=0,
```

```
    xstep=30, # Volume element size
    ystep=30,
    zstep=30,
    nelem_sim_max=10000,  # Target number of volume elements to consider at a tim
    gain=3e-5,  # This gain was calibrated against POLARBEAR data
    # If the wind is strong or the observation is long, the volume becomes
    # too large.  This parameter controls breaking the simulation into
    # disjoint segments
    wind_dist=10000,
    cachedir=cachedir,
    freq=100,
    verbosity=1,  # Print progress to stdout and write out the correlation functi
)
atmsim.exec(data)

# Pick every second detector because the simulated atmosphere is not polarized
dets = tod.local_dets[::2]

# Plot the Kolmogorov correlation function
ax = fig.add_subplot(3, 2, 1 + iplot)
kolmo = np.genfromtxt("kolmogorov.txt").T
ax.semilogx(kolmo[0], kolmo[1])
ax.set_xlabel("Separation [m]")
ax.set_ylabel("Correlation factor")
ax.set_title("Kolmogorov correlation, lmax = {}m".format(lmax))
ax.grid()

ax = fig.add_subplot(3, 2, 3 + iplot)
ind = slice(0, 1000)
tod = obs["tod"]
times = tod.local_times()
for det in dets:
    simulated = tod.local_signal(det, name)
    ax.plot(times[ind], simulated[ind], "-", label=det)
ax.set_xlabel("Unix time [s]")
ax.set_ylabel("Atmosphere [K]")
ax.set_title("Simulated atmosphere, lmax = {}m".format(lmax))
ax.grid()

ax = fig.add_subplot(3, 2, 5 + iplot)
for det in dets:
    simulated = tod.local_signal(det, name)
    ax.plot(times, simulated, "-", label=det)
ax.set_xlabel("Unix time [s]")
ax.set_ylabel("Atmosphere [K]")
ax.set_title("Simulated atmosphere, lmax = {}m".format(lmax))
ax.grid()
```

```
        plt.legend(loc="upper right", bbox_to_anchor=(1.2, 1.00))
        plt.subplots_adjust(hspace=0.3)
```

```
Creating atm_cache_10/6/6/6
TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : Setting up atmosphere simulation
TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : Instantiating atmosphere for t = 0.0
TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : OpSimAtmosphere: Initialize atmosphere:  0.0


atmsim constructed with 1 processes, 2 threads per process.

Input parameters:
             az = [227.807 - 244.881] (17.0743 degrees)
             el = [36.8741 - 39.1859] (2.31171 degrees)
              t = [1.57785e+09 - 1.57785e+09] (3179.9 s)
           lmin = 0.01 +- 0 m
           lmax = 10 +- 0 m
              w = 1.80032 +- 0 m
           wdir = -172.765 +- 0 degrees
             z0 = 2000 +- 0 m
             T0 = 276.965 +- 0 K
           zatm = 40000 m
           zmax = 1000 m
       scan frame:
          xstep = 30 m
          ystep = 30 m
          zstep = 30 m
 horizontal frame:
         xxstep = 5.14843 m
         yystep = 30 m
         zzstep = 42.1129 m
  nelem_sim_max = 10000
         corrlim = 0.001
      verbosity = 1
           rmin = 0 m
           rmax = 100 m

Atmospheric realization parameters:
 lmin = 0.01 m
 lmax = 10 m
    w = 1.80032 m/s
 easthward wind = -1.78599 m/s
 northward wind = -0.226741 m/s
  az0 = 236.344 degrees
  el0 = 38.03 degrees
   wx = -1.26998 m/s
   wy = 0.801063 m/s
   wz = -0.993287 m/s
```

```
 wdir = -172.765 degrees
   z0 = 2000 m
   T0 = 276.965 K

Simulation volume:
   delta_x = 4170.96 m
   delta_y = 2631.66 m
   delta_z = 3223.23 m
Observation cone along the X-axis:
   delta_y_cone = 24.3586 m
   delta_z_cone = 4.68029 m
    xstart = -4038.4 m
    ystart = -42.1793 m
    zstart = -3190.62 m
   maxdist = 102.559 m
        nx = 140
        ny = 88
        nz = 108
        nn = 1330560

Evaluating Kolmogorov correlation at 1000 different separations in range 0 - 5950.59 m
kappamin = 0.1 1/m, kappamax =  100 1/m. nkappa = 1000000
Compressing volume, N = 1330560
Flagged hits, flagging neighbors
Creating compression table
Resizing realization to 27350
X-slice: 0 -- 1560(52 30 m layers) m out of  4200 m indices 0 -- 10058 ( 10058 ) out of 27350

0 : Allocated 3.99903 MB for the sparse covariance matrix. Compression: 0.00518133
0 : Analyzing sparse covariance ...
0 : Factorizing sparse covariance ...

0 : Allocated 0.0383682 MB for the sparse factorization.

0 : Allocated 50.8239 MB for the sparse sqrt covariance matrix. Compression: 0.0658498
X-slice: 1560 -- 3000(48 30 m layers) m out of  4200 m indices 10058 -- 20097 ( 10039 ) out of

0 : Allocated 3.98053 MB for the sparse covariance matrix. Compression: 0.00517691
0 : Analyzing sparse covariance ...
0 : Factorizing sparse covariance ...

0 : Allocated 0.0382957 MB for the sparse factorization.

0 : Allocated 51.5798 MB for the sparse sqrt covariance matrix. Compression: 0.0670823
X-slice: 3000 -- 4170(39 30 m layers) m out of  4200 m indices 20097 -- 27350 ( 7253 ) out of 2

0 : Allocated 2.8653 MB for the sparse covariance matrix. Compression: 0.00713912
0 : Analyzing sparse covariance ...
```

```
0 : Factorizing sparse covariance ...

0 : Allocated 0.027668 MB for the sparse factorization.

0 : Allocated 33.8024 MB for the sparse sqrt covariance matrix. Compression: 0.0842213
Saved metadata to atm_cache_10/6/6/6/14869056_588491666_0_0_metadata.txt
Saved realization to atm_cache_10/6/6/6/14869056_588491666_0_0_realization.dat


TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : Simulating the atmosphere for t = 0.0
TOAST INFO: Kolmogorov initialized in:  3.73 seconds (1 calls)
TOAST INFO: Volume compressed in:  9.41 seconds (1 calls)
27350 / 1330560(2.05553 %) volume elements are needed for the simulation
nx = 140 ny = 88 nz = 108
wx = -1.26998 wy = 0.801063 wz = -0.993287
TOAST INFO: Sparse covariance evaluated in:  0.40 seconds (1 calls)
TOAST INFO: Sparse covariance constructed in:  0.41 seconds (2 calls)
TOAST INFO: Cholesky decomposition done in:  0.68 seconds (1 calls)
 s. N = 10058
TOAST INFO: Realization slice (0 -- 10058) var = 7.52128, constructed in:  0.04 seconds (1 cal
TOAST INFO: Sparse covariance evaluated in:  0.38 seconds (1 calls)
TOAST INFO: Sparse covariance constructed in:  0.38 seconds (2 calls)
TOAST INFO: Cholesky decomposition done in:  0.39 seconds (1 calls)
 s. N = 10039
TOAST INFO: Realization slice (10058 -- 20097) var = 3.02572, constructed in:  0.01 seconds (1
TOAST INFO: Sparse covariance evaluated in:  0.24 seconds (1 calls)
TOAST INFO: Sparse covariance constructed in:  0.25 seconds (2 calls)
TOAST INFO: Cholesky decomposition done in:  0.11 seconds (1 calls)
 s. N = 7253
TOAST INFO: Realization slice (20097 -- 27350) var = 1.39633, constructed in:  0.00 seconds (1
TOAST INFO: Realization constructed in:  2.35 seconds (1 calls)
TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : OpSimAtmosphere: Simulated atmosphere:  15.4
TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : Observing the atmosphere
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : OpSimAtmosphere: Observe atmosphere:  0.26 s
```

```
TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : OpSimAtmosphere: Initialize atmosphere:  0.00
TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : Simulating the atmosphere for t = 0.0
TOAST INFO: Kolmogorov initialized in:  3.88 seconds (1 calls)
TOAST INFO: Volume compressed in:  0.71 seconds (1 calls)
13171 / 63360(20.7876 %) volume elements are needed for the simulation
nx = 55 ny = 32 nz = 36
wx = -1.26998 wy = 0.801063 wz = -0.993287
TOAST INFO: Sparse covariance evaluated in:  0.34 seconds (1 calls)
TOAST INFO: Sparse covariance constructed in:  0.34 seconds (2 calls)
TOAST INFO: Cholesky decomposition done in:  0.00 seconds (1 calls)
 s. N = 10155
TOAST INFO: Realization slice (0 -- 10155) var = 4.4366, constructed in:  0.00 seconds (1 calls
TOAST INFO: Sparse covariance evaluated in:  0.03 seconds (1 calls)
TOAST INFO: Sparse covariance constructed in:  0.03 seconds (2 calls)
TOAST INFO: Cholesky decomposition done in:  0.00 seconds (1 calls)
 s. N = 3016
TOAST INFO: Realization slice (10155 -- 13171) var = 1.28794, constructed in:  0.00 seconds (1
TOAST INFO: Realization constructed in:  0.38 seconds (1 calls)
TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : OpSimAtmosphere: Simulated atmosphere:  4.99
TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : Observing the atmosphere
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)


atmsim constructed with 1 processes, 2 threads per process.

Input parameters:
             az = [227.807 - 244.881] (17.0743 degrees)
             el = [36.8741 - 39.1859] (2.31171 degrees)
              t = [1.57785e+09 - 1.57785e+09] (3179.9 s)
           lmin = 0.01 +- 0 m
           lmax = 10 +- 0 m
              w = 1.80032 +- 0 m
           wdir = -172.765 +- 0 degrees
             z0 = 2000 +- 0 m
             T0 = 276.965 +- 0 K
           zatm = 40000 m
           zmax = 1000 m
       scan frame:
          xstep = 94.8683 m
```

```
          ystep = 94.8683 m
          zstep = 94.8683 m
 horizontal frame:
         xxstep = 16.2808 m
         yystep = 94.8683 m
         zzstep = 133.173 m
  nelem_sim_max = 10000
        corrlim = 0.001
      verbosity = 1
           rmin = 100 m
           rmax = 1000 m

Atmospheric realization parameters:
 lmin = 0.01 m
 lmax = 10 m
    w = 1.80032 m/s
 easthward wind = -1.78599 m/s
 northward wind = -0.226741 m/s
  az0 = 236.344 degrees
  el0 = 38.03 degrees
   wx = -1.26998 m/s
   wy = 0.801063 m/s
   wz = -0.993287 m/s
 wdir = -172.765 degrees
   z0 = 2000 m
   T0 = 276.965 K

Simulation volume:
   delta_x = 5158.86 m
   delta_y = 2980.62 m
   delta_z = 3395.09 m
Observation cone along the X-axis:
   delta_y_cone = 243.586 m
   delta_z_cone = 46.8029 m
    xstart = -4038.4 m
    ystart = -216.661 m
    zstart = -3274.11 m
   maxdist = 1025.59 m
        nx = 55
        ny = 32
        nz = 36
        nn = 63360

Evaluating Kolmogorov correlation at 1000 different separations in range 0 - 6926.02 m
kappamin = 0.1 1/m, kappamax =  100 1/m. nkappa = 1000000
Compressing volume, N = 63360
Flagged hits, flagging neighbors
Creating compression table
```

```
Resizing realization to 13171
X-slice: 0 -- 3605(38 94.8683 m layers) m out of  5217.76 m indices 0 -- 10155 ( 10155 ) out o

0 : Allocated 0.154953 MB for the sparse covariance matrix. Compression: 0.000196947
0 : Analyzing sparse covariance ...
0 : Factorizing sparse covariance ...

0 : Allocated 0.154953 MB for the sparse factorization.

0 : Allocated 0.154953 MB for the sparse sqrt covariance matrix. Compression: 0.000196947
X-slice: 3605 -- 5122.89(16 94.8683 m layers) m out of  5217.76 m indices 10155 -- 13171 ( 3016

0 : Allocated 0.0460205 MB for the sparse covariance matrix. Compression: 0.00066313
0 : Analyzing sparse covariance ...
0 : Factorizing sparse covariance ...

0 : Allocated 0.0460205 MB for the sparse factorization.

0 : Allocated 0.0460205 MB for the sparse sqrt covariance matrix. Compression: 0.00066313
Saved metadata to atm_cache_10/6/6/6/14869056_588491666_1_0_metadata.txt
Saved realization to atm_cache_10/6/6/6/14869056_588491666_1_0_realization.dat


TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : OpSimAtmosphere: Observe atmosphere:  0.30 s
TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : OpSimAtmosphere: Initialize atmosphere:  0.0


atmsim constructed with 1 processes, 2 threads per process.

Input parameters:
             az = [227.807 - 244.881] (17.0743 degrees)
             el = [36.8741 - 39.1859] (2.31171 degrees)
              t = [1.57785e+09 - 1.57785e+09] (3179.9 s)
           lmin = 0.01 +- 0 m
           lmax = 10 +- 0 m
              w = 1.80032 +- 0 m
           wdir = -172.765 +- 0 degrees
             z0 = 2000 +- 0 m
             T0 = 276.965 +- 0 K
           zatm = 40000 m
           zmax = 1000 m
       scan frame:
          xstep = 300 m
          ystep = 300 m
```

```
          zstep = 300 m
 horizontal frame:
         xxstep = 51.4843 m
         yystep = 300 m
         zzstep = 421.129 m
  nelem_sim_max = 10000
        corrlim = 0.001
      verbosity = 1
           rmin = 1000 m
           rmax = 10000 m

Atmospheric realization parameters:
 lmin = 0.01 m
 lmax = 10 m
    w = 1.80032 m/s
 easthward wind = -1.78599 m/s
 northward wind = -0.226741 m/s
  az0 = 236.344 degrees
  el0 = 38.03 degrees
   wx = -1.26998 m/s
   wy = 0.801063 m/s
   wz = -0.993287 m/s
 wdir = -172.765 degrees
   z0 = 2000 m
   T0 = 276.965 K

Simulation volume:
   delta_x = 5961.58 m
   delta_y = 3532.82 m
   delta_z = 3832.63 m
Observation cone along the X-axis:
   delta_y_cone = 385.519 m
   delta_z_cone = 74.0743 m
    xstart = -4038.4 m
    ystart = -492.76 m
    zstart = -3491.29 m
   maxdist = 1623.18 m
        nx = 20
        ny = 12
        nz = 13
        nn = 3120

Evaluating Kolmogorov correlation at 1000 different separations in range 0 - 7998.17 m
kappamin = 0.1 1/m, kappamax =  100 1/m. nkappa = 1000000
Compressing volume, N = 3120
Flagged hits, flagging neighbors
Creating compression table
Resizing realization to 2364
```

```
X-slice: 0 -- 5700(19 300 m layers) m out of  6000 m indices 0 -- 2364 ( 2364 ) out of 2364

0 : Allocated 0.0360718 MB for the sparse covariance matrix. Compression: 0.000846024
0 : Analyzing sparse covariance ...
0 : Factorizing sparse covariance ...

0 : Allocated 0.0360718 MB for the sparse factorization.

0 : Allocated 0.0360718 MB for the sparse sqrt covariance matrix. Compression: 0.000846024
Saved metadata to atm_cache_10/6/6/6/14869056_588491666_2_0_metadata.txt
Saved realization to atm_cache_10/6/6/6/14869056_588491666_2_0_realization.dat


TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : Simulating the atmosphere for t = 0.0
TOAST INFO: Kolmogorov initialized in:  3.46 seconds (1 calls)
TOAST INFO: Volume compressed in:  0.04 seconds (1 calls)
2364 / 3120(75.7692 %) volume elements are needed for the simulation
nx = 20 ny = 12 nz = 13
wx = -1.26998 wy = 0.801063 wz = -0.993287
TOAST INFO: Sparse covariance evaluated in:  0.02 seconds (1 calls)
TOAST INFO: Sparse covariance constructed in:  0.02 seconds (2 calls)
TOAST INFO: Cholesky decomposition done in:  0.00 seconds (1 calls)
 s. N = 2364
TOAST INFO: Realization slice (0 -- 2364) var = 3.42455, constructed in:  0.00 seconds (1 calls
TOAST INFO: Realization constructed in:  0.02 seconds (1 calls)
TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : OpSimAtmosphere: Simulated atmosphere:  3.53
TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : Observing the atmosphere
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : OpSimAtmosphere: Observe atmosphere:  0.27 se
TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : Simulated and observed atmosphere:  24.85 sec
Creating atm_cache_300/6/6/6
TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : Setting up atmosphere simulation
TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : Instantiating atmosphere for t = 0.0
TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : OpSimAtmosphere: Initialize atmosphere:  0.00
TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : Simulating the atmosphere for t = 0.0
```

```
TOAST INFO: Kolmogorov initialized in:  3.73 seconds (1 calls)
TOAST INFO: Volume compressed in:  9.54 seconds (1 calls)
27350 / 1330560(2.05553 %) volume elements are needed for the simulation
nx = 140 ny = 88 nz = 108
wx = -1.26998 wy = 0.801063 wz = -0.993287
TOAST INFO: Sparse covariance evaluated in:  8.00 seconds (1 calls)
TOAST INFO: Sparse covariance constructed in:  11.94 seconds (2 calls)
TOAST INFO: Cholesky decomposition done in:  16.80 seconds (1 calls)
 s. N = 10058
TOAST INFO: Realization slice (0 -- 10058) var = 4.44867, constructed in:  0.07 seconds (1 cal
TOAST INFO: Sparse covariance evaluated in:  4.67 seconds (1 calls)
TOAST INFO: Sparse covariance constructed in:  8.55 seconds (2 calls)
TOAST INFO: Cholesky decomposition done in:  16.41 seconds (1 calls)
 s. N = 10039
TOAST INFO: Realization slice (10058 -- 20097) var = 2.08106, constructed in:  0.07 seconds (1
TOAST INFO: Sparse covariance evaluated in:  2.32 seconds (1 calls)
TOAST INFO: Sparse covariance constructed in:  4.39 seconds (2 calls)
TOAST INFO: Cholesky decomposition done in:  7.53 seconds (1 calls)
 s. N = 7253
TOAST INFO: Realization slice (20097 -- 27350) var = 0.892358, constructed in:  0.04 seconds (
TOAST INFO: Realization constructed in:  66.46 seconds (1 calls)
TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : OpSimAtmosphere: Simulated atmosphere:  79.7
TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : Observing the atmosphere
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : OpSimAtmosphere: Observe atmosphere:  0.15 s
TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : OpSimAtmosphere: Initialize atmosphere:  0.0


atmsim constructed with 1 processes, 2 threads per process.

Input parameters:
             az = [227.807 - 244.881] (17.0743 degrees)
             el = [36.8741 - 39.1859] (2.31171 degrees)
              t = [1.57785e+09 - 1.57785e+09] (3179.9 s)
           lmin = 0.01 +- 0 m
```

```
           lmax = 300 +- 0 m
              w = 1.80032 +- 0 m
           wdir = -172.765 +- 0 degrees
             z0 = 2000 +- 0 m
             T0 = 276.965 +- 0 K
           zatm = 40000 m
           zmax = 1000 m
       scan frame:
          xstep = 30 m
          ystep = 30 m
          zstep = 30 m
 horizontal frame:
         xxstep = 5.14843 m
         yystep = 30 m
         zzstep = 42.1129 m
  nelem_sim_max = 10000
        corrlim = 0.001
      verbosity = 1
           rmin = 0 m
           rmax = 100 m

Atmospheric realization parameters:
 lmin = 0.01 m
 lmax = 300 m
    w = 1.80032 m/s
 easthward wind = -1.78599 m/s
 northward wind = -0.226741 m/s
  az0 = 236.344 degrees
  el0 = 38.03 degrees
   wx = -1.26998 m/s
   wy = 0.801063 m/s
   wz = -0.993287 m/s
 wdir = -172.765 degrees
   z0 = 2000 m
   T0 = 276.965 K

Simulation volume:
   delta_x = 4170.96 m
   delta_y = 2631.66 m
   delta_z = 3223.23 m
Observation cone along the X-axis:
   delta_y_cone = 24.3586 m
   delta_z_cone = 4.68029 m
    xstart = -4038.4 m
    ystart = -42.1793 m
    zstart = -3190.62 m
   maxdist = 102.559 m
        nx = 140
```

```
        ny = 88
        nz = 108
        nn = 1330560

Evaluating Kolmogorov correlation at 1000 different separations in range 0 - 5950.59 m
kappamin = 0.00333333 1/m, kappamax =  100 1/m. nkappa = 1000000
Compressing volume, N = 1330560
Flagged hits, flagging neighbors
Creating compression table
Resizing realization to 27350
X-slice: 0 -- 1560(52 30 m layers) m out of  4200 m indices 0 -- 10058 ( 10058 ) out of 27350

0 : Allocated 578.957 MB for the sparse covariance matrix. Compression: 0.750124
0 : Analyzing sparse covariance ...
0 : Factorizing sparse covariance ...

0 : Allocated 0.0383682 MB for the sparse factorization.

0 : Allocated 578.957 MB for the sparse sqrt covariance matrix. Compression: 0.750124
X-slice: 1560 -- 3000(48 30 m layers) m out of  4200 m indices 10058 -- 20097 ( 10039 ) out of

0 : Allocated 576.772 MB for the sparse covariance matrix. Compression: 0.750125
0 : Analyzing sparse covariance ...
0 : Factorizing sparse covariance ...

0 : Allocated 0.0382957 MB for the sparse factorization.

0 : Allocated 576.772 MB for the sparse sqrt covariance matrix. Compression: 0.750125
X-slice: 3000 -- 4170(39 30 m layers) m out of  4200 m indices 20097 -- 27350 ( 7253 ) out of 1

0 : Allocated 301.083 MB for the sparse covariance matrix. Compression: 0.750172
0 : Analyzing sparse covariance ...
0 : Factorizing sparse covariance ...

0 : Allocated 0.027668 MB for the sparse factorization.

0 : Allocated 301.083 MB for the sparse sqrt covariance matrix. Compression: 0.750172
Saved metadata to atm_cache_300/6/6/6/14869056_588491666_0_0_metadata.txt
Saved realization to atm_cache_300/6/6/6/14869056_588491666_0_0_realization.dat
atmsim constructed with 1 processes, 2 threads per process.

Input parameters:
             az = [227.807 - 244.881] (17.0743 degrees)
             el = [36.8741 - 39.1859] (2.31171 degrees)
              t = [1.57785e+09 - 1.57785e+09] (3179.9 s)
           lmin = 0.01 +- 0 m
           lmax = 300 +- 0 m
              w = 1.80032 +- 0 m
```

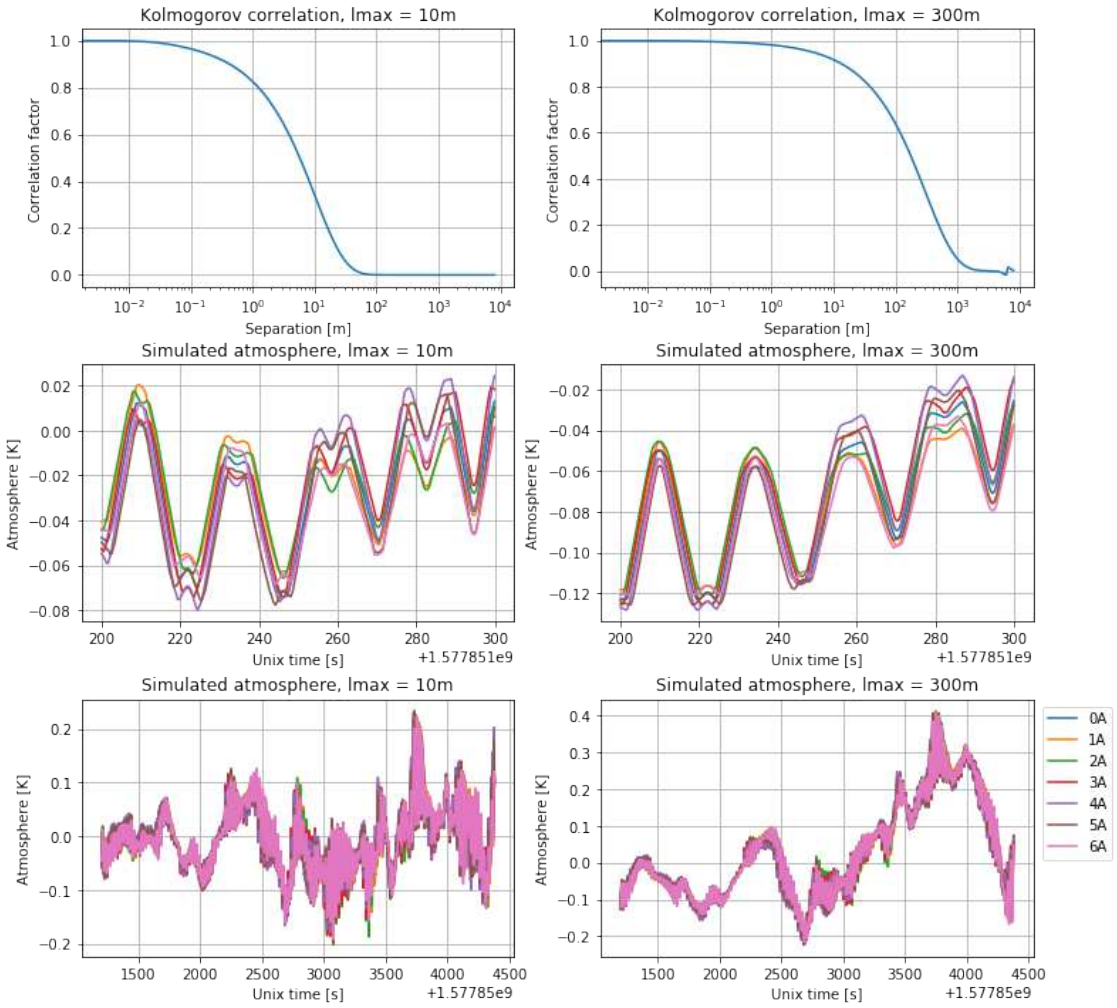
```
           wdir = -172.765 +- 0 degrees
             z0 = 2000 +- 0 m
             T0 = 276.965 +- 0 K
           zatm = 40000 m
           zmax = 1000 m
       scan frame:
          xstep = 94.8683 m
          ystep = 94.8683 m
          zstep = 94.8683 m
 horizontal frame:
         xxstep = 16.2808 m
         yystep = 94.8683 m
         zzstep = 133.173 m
  nelem_sim_max = 10000
        corrlim = 0.001
      verbosity = 1
           rmin = 100 m
           rmax = 1000 m


TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : Simulating the atmosphere for t = 0.0
TOAST INFO: Kolmogorov initialized in:  3.03 seconds (1 calls)
TOAST INFO: Volume compressed in:  0.71 seconds (1 calls)
13171 / 63360(20.7876 %) volume elements are needed for the simulation
nx = 55 ny = 32 nz = 36
wx = -1.26998 wy = 0.801063 wz = -0.993287
TOAST INFO: Sparse covariance evaluated in:  7.95 seconds (1 calls)
TOAST INFO: Sparse covariance constructed in:  11.19 seconds (2 calls)
TOAST INFO: Cholesky decomposition done in:  14.78 seconds (1 calls)
 s. N = 10155
TOAST INFO: Realization slice (0 -- 10155) var = 4.11504, constructed in:  0.08 seconds (1 cal
TOAST INFO: Sparse covariance evaluated in:  0.40 seconds (1 calls)
TOAST INFO: Sparse covariance constructed in:  0.71 seconds (2 calls)
TOAST INFO: Cholesky decomposition done in:  0.69 seconds (1 calls)
 s. N = 3016
TOAST INFO: Realization slice (10155 -- 13171) var = 1.10595, constructed in:  0.01 seconds (1
TOAST INFO: Realization constructed in:  27.75 seconds (1 calls)
TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : OpSimAtmosphere: Simulated atmosphere:  31.5(
TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : Observing the atmosphere
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
```

```
Atmospheric realization parameters:
 lmin = 0.01 m
 lmax = 300 m
    w = 1.80032 m/s
 easthward wind = -1.78599 m/s
 northward wind = -0.226741 m/s
  az0 = 236.344 degrees
  el0 = 38.03 degrees
   wx = -1.26998 m/s
   wy = 0.801063 m/s
   wz = -0.993287 m/s
 wdir = -172.765 degrees
   z0 = 2000 m
   T0 = 276.965 K

Simulation volume:
   delta_x = 5158.86 m
   delta_y = 2980.62 m
   delta_z = 3395.09 m
Observation cone along the X-axis:
   delta_y_cone = 243.586 m
   delta_z_cone = 46.8029 m
    xstart = -4038.4 m
    ystart = -216.661 m
    zstart = -3274.11 m
   maxdist = 1025.59 m
        nx = 55
        ny = 32
        nz = 36
        nn = 63360

Evaluating Kolmogorov correlation at 1000 different separations in range 0 - 6926.02 m
kappamin = 0.00333333 1/m, kappamax =  100 1/m. nkappa = 1000000
Compressing volume, N = 63360
Flagged hits, flagging neighbors
Creating compression table
Resizing realization to 13171
X-slice: 0 -- 3605(38 94.8683 m layers) m out of  5217.76 m indices 0 -- 10155 ( 10155 ) out o

0 : Allocated 477.712 MB for the sparse covariance matrix. Compression: 0.607179
0 : Analyzing sparse covariance ...
0 : Factorizing sparse covariance ...

0 : Allocated 0.0387383 MB for the sparse factorization.
```

```
0 : Allocated 590.177 MB for the sparse sqrt covariance matrix. Compression: 0.750123
X-slice: 3605 -- 5122.89(16 94.8683 m layers) m out of  5217.76 m indices 10155 -- 13171 ( 3016

0 : Allocated 52.0632 MB for the sparse covariance matrix. Compression: 0.750202
0 : Analyzing sparse covariance ...
0 : Factorizing sparse covariance ...

0 : Allocated 0.0115051 MB for the sparse factorization.

0 : Allocated 52.078 MB for the sparse sqrt covariance matrix. Compression: 0.750414
Saved metadata to atm_cache_300/6/6/6/14869056_588491666_1_0_metadata.txt
Saved realization to atm_cache_300/6/6/6/14869056_588491666_1_0_realization.dat


TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : OpSimAtmosphere: Observe atmosphere:  0.30 se
TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : OpSimAtmosphere: Initialize atmosphere:  0.00


atmsim constructed with 1 processes, 2 threads per process.

Input parameters:
             az = [227.807 - 244.881] (17.0743 degrees)
             el = [36.8741 - 39.1859] (2.31171 degrees)
              t = [1.57785e+09 - 1.57785e+09] (3179.9 s)
           lmin = 0.01 +- 0 m
           lmax = 300 +- 0 m
              w = 1.80032 +- 0 m
           wdir = -172.765 +- 0 degrees
             z0 = 2000 +- 0 m
             T0 = 276.965 +- 0 K
           zatm = 40000 m
           zmax = 1000 m
       scan frame:
          xstep = 300 m
          ystep = 300 m
          zstep = 300 m
 horizontal frame:
         xxstep = 51.4843 m
         yystep = 300 m
         zzstep = 421.129 m
  nelem_sim_max = 10000
        corrlim = 0.001
     verbosity = 1
```

```
           rmin = 1000 m
           rmax = 10000 m

Atmospheric realization parameters:
 lmin = 0.01 m
 lmax = 300 m
    w = 1.80032 m/s
 easthward wind = -1.78599 m/s
 northward wind = -0.226741 m/s
  az0 = 236.344 degrees
  el0 = 38.03 degrees
   wx = -1.26998 m/s
   wy = 0.801063 m/s
   wz = -0.993287 m/s
 wdir = -172.765 degrees
   z0 = 2000 m
   T0 = 276.965 K

Simulation volume:
   delta_x = 5961.58 m
   delta_y = 3532.82 m
   delta_z = 3832.63 m
Observation cone along the X-axis:
   delta_y_cone = 385.519 m
   delta_z_cone = 74.0743 m
    xstart = -4038.4 m
    ystart = -492.76 m
    zstart = -3491.29 m
   maxdist = 1623.18 m
        nx = 20
        ny = 12
        nz = 13
        nn = 3120

Evaluating Kolmogorov correlation at 1000 different separations in range 0 - 7998.17 m
kappamin = 0.00333333 1/m, kappamax =  100 1/m. nkappa = 1000000
Compressing volume, N = 3120
Flagged hits, flagging neighbors
Creating compression table
Resizing realization to 2364
X-slice: 0 -- 5700(19 300 m layers) m out of  6000 m indices 0 -- 2364 ( 2364 ) out of 2364

0 : Allocated 18.829 MB for the sparse covariance matrix. Compression: 0.441613
0 : Analyzing sparse covariance ...
0 : Factorizing sparse covariance ...

0 : Allocated 0.00901794 MB for the sparse factorization.
```

```
0 : Allocated 32.0002 MB for the sparse sqrt covariance matrix. Compression: 0.750529
Saved metadata to atm_cache_300/6/6/6/14869056_588491666_2_0_metadata.txt
Saved realization to atm_cache_300/6/6/6/14869056_588491666_2_0_realization.dat


TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : Simulating the atmosphere for t = 0.0
TOAST INFO: Kolmogorov initialized in:  2.60 seconds (1 calls)
TOAST INFO: Volume compressed in:  0.04 seconds (1 calls)
2364 / 3120(75.7692 %) volume elements are needed for the simulation
nx = 20 ny = 12 nz = 13
wx = -1.26998 wy = 0.801063 wz = -0.993287
TOAST INFO: Sparse covariance evaluated in:  0.39 seconds (1 calls)
TOAST INFO: Sparse covariance constructed in:  0.48 seconds (2 calls)
TOAST INFO: Cholesky decomposition done in:  0.29 seconds (1 calls)
 s. N = 2364
TOAST INFO: Realization slice (0 -- 2364) var = 3.07798, constructed in:  0.00 seconds (1 call
TOAST INFO: Realization constructed in:  0.79 seconds (1 calls)
TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : OpSimAtmosphere: Simulated atmosphere:  3.43
TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : Observing the atmosphere
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO:  samples observed in:  0.01 seconds (1 calls)
TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : OpSimAtmosphere: Observe atmosphere:  0.28 se
TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : Simulated and observed atmosphere:  115.40 se
```



1.4 Ground (scan-synchronous) signal

TOAST includes a new module for simulating scan-synchronous signals. It works by sampling a provided or on-the-fly synthesized low resolution map using the horizontal detector pointing instead of sky pointing.

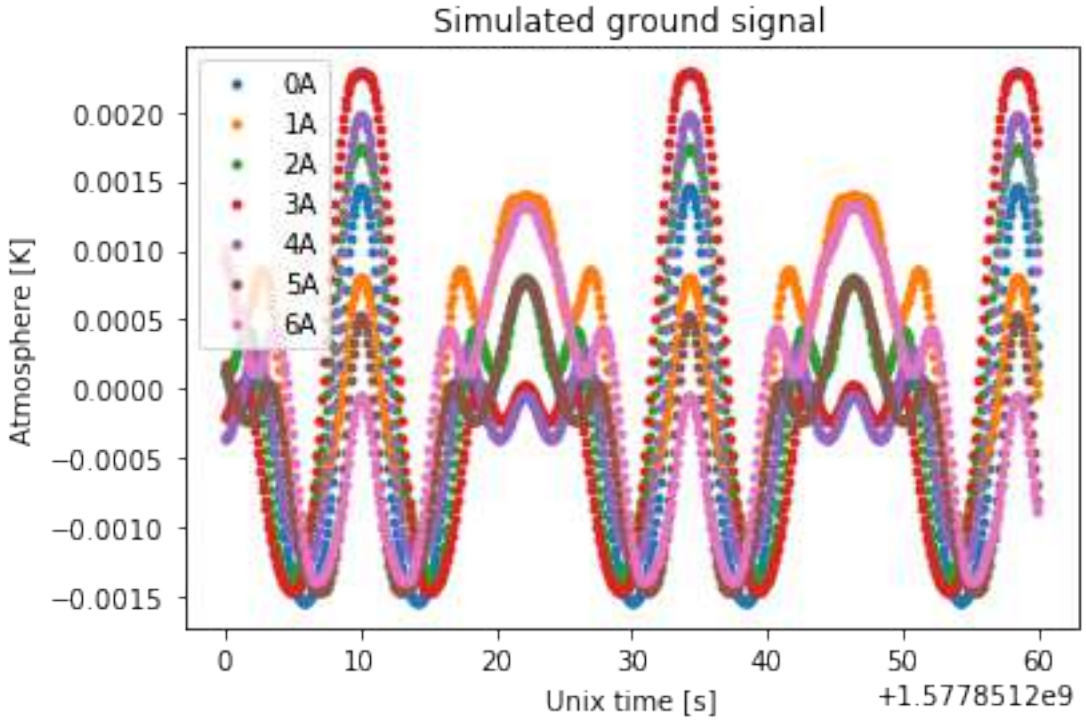
src/toast/todmap/sss.py

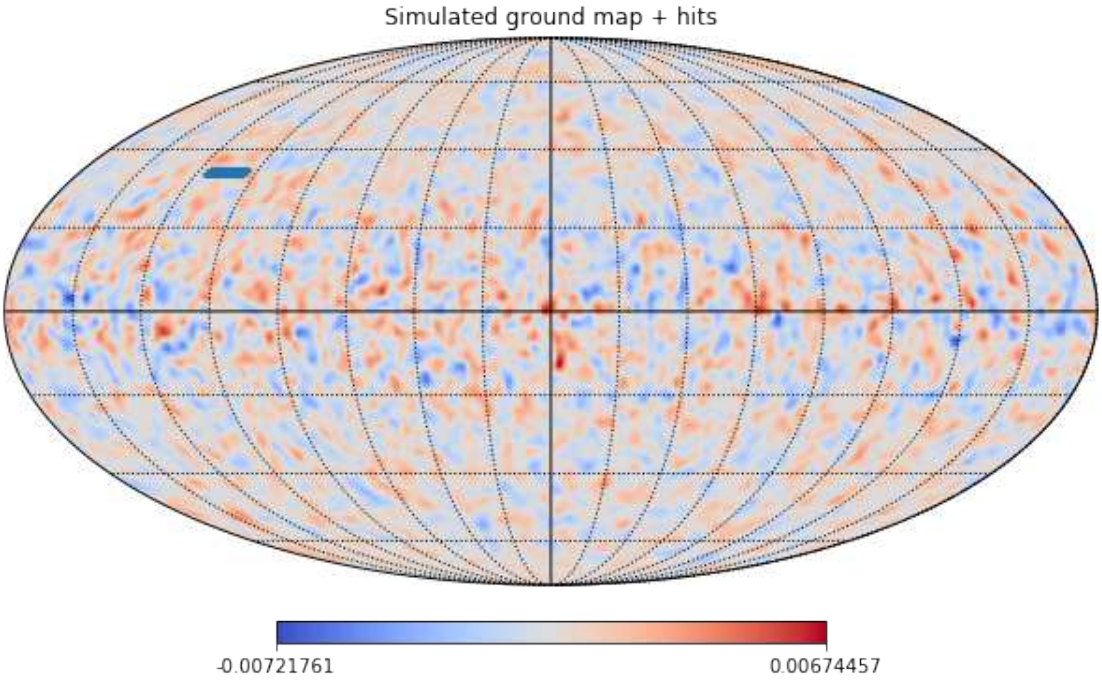
```
In [18]: toast.tod.OpCacheClear("sss").exec(data)

         sim_sss = toast.todmap.OpSimScanSynchronousSignal(
             out="sss",
             realization=0,
             nside=512,     # internal resolution for the simulated ground map
             fwhm=3,        # smoothing scale for the ground map
             scale=1e-3,    # RMS for observations at el=45 deg
             lmax=512,      # expansion lmax
```

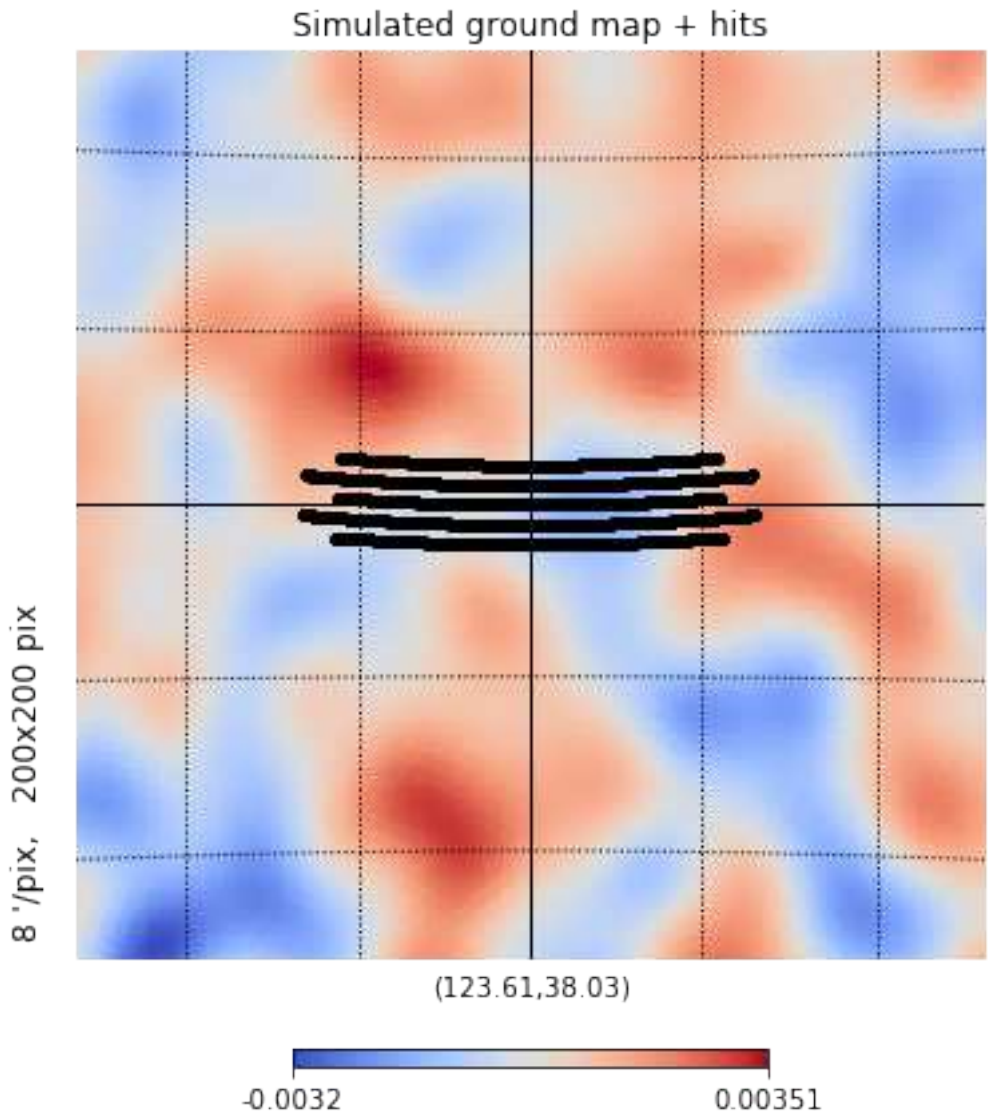
```
        power=-1,     # power law that suppresses ground signal at higher elevation
        path=None,    # alternative ground map in Healpix format, will override nside, sc
    )

    sim_sss.exec(data)

    tod = obs["tod"]
    times = tod.local_times()
    ind = slice(0, 600)
    for det in tod.local_dets[::2]:
        simulated = tod.local_signal(det, "sss")
        plt.plot(times[ind], simulated[ind], ".", label=det)
    ax = plt.gca()
    ax.set_xlabel("Unix time [s]")
    ax.set_ylabel("Atmosphere [K]")
    ax.set_title("Simulated ground signal")
    plt.legend(loc="best")

    # The ground map is simulated on-the-fly and discarded afterwards.  By default, each
    # an entirely independent ground map.  Here is an example:

    weather = obs["weather"]
    weather.set(0, 0, 0)
    key1, key2, counter1, counter2 = sim_sss._get_rng_keys(obs)
    ground_map = sim_sss._simulate_sss(key1, key2, counter1, counter2, weather, mpiworld)

    healpy.mollview(ground_map, cmap="coolwarm", title="Simulated ground map + hits")
    for det in tod.local_dets[::2]:
        quat_azel = tod.read_pntg(det, azel=True)
        theta, phi = toast.qarray.to_position(quat_azel)
        healpy.projplot(theta, phi, '-')
    healpy.graticule(22.5, verbose=False)

    healpy.gnomview(
        ground_map,
        cmap="coolwarm",
        title="Simulated ground map + hits",
        rot=[360 - .5 * (ces.azmin + ces.azmax), ces.el],
        reso=8,
    )
    for det in tod.local_dets[::2]:
        quat_azel = tod.read_pntg(det, azel=True)
        theta, phi = toast.qarray.to_position(quat_azel)
        healpy.projplot(theta, phi, 'k.')
    healpy.graticule(5, verbose=False)
```

```
TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : Setting up SSS simulation
Sigma is 76.438962 arcmin (0.022235 rad)
```

```
-> fwhm is 180.000000 arcmin
Sigma is 0.000000 arcmin (0.000000 rad)
-> fwhm is 0.000000 arcmin
TOAST INFO: 0 : CES-Atacama-LAT-small_patch-0-0 : Observing the scan-synchronous signal
Sigma is 76.438962 arcmin (0.022235 rad)
-> fwhm is 180.000000 arcmin
Sigma is 0.000000 arcmin (0.000000 rad)
-> fwhm is 0.000000 arcmin
```




Simulated ground map + hits
-0.00721761
0.00674457


Simulated ground map + hits
8 '/pix, 200x200 pix
(123.61,38.03)
-0.0032
0.00351

Interfacing to Real Data

TOAST Workshop - 2019 UCSD

Raw Data

- Usually a binary dump
  - CData
  - GCP / Archive files
  - Telemetry packets
  - spt3g (small frames)
- Naturally frame-like, since that is how most data acquisition systems work
- Often multiple sources of information from different systems

Data for Analysis

- Usually some merging of raw data sources
- Usually "more unpacked" (longer contiguous chunks)
  - Dirfiles (subdirectories, zipped archives)
  - Spt3g (with bigger frames)
  - HDF5 (2D datasets)
  - FITS (binary tables)
- Usually has been "indexed" somehow. Metadata like time range, sky patch, weather properties, etc are stored in a DB or some other format.

Data for Analysis

- Frequently data has natural split in time and detector directions
  - (Time) Changes in pointing configuration
    - Elevation changes
    - Thruster firing
  - (Time) Cooler cycling / thermal changes
  - (Time) Computer reboots
  - (Detector) Different readout cards
  - (Detector) Split by frequency
- These natural splits should be used to define TOAST observations.

TOAST Considerations

- Determine observation boundaries
- What (python) tools does your project have for querying metadata? Can you select data based on the observation splits you want?
- In order to read telescope and detector data for an observation, what information is needed? Examples:
    - Paths to files
    - Data ranges within files
    - Other metadata that is not contained in the files

TOAST Interface

- TOD class which takes in the constructor all information to read data for the observation.
- Python code to query metadata sources and instantiate observations:

```
def create_observations(selection, commworld):
    if commworld.rank == 0:
         # Only one process touches DB
        allobs = query(selection)
     allobs = commworld.bcast(allobs, root=0)
     comm = toast.Comm(world=commworld)
     # Distribute observations based on some criteria (i.e size)
     groupobs = distribute_obs(comm, allobs)
     for obs in groupobs:
        obs["tod"] = MyTOD(....)
```

TOAST Interface Example: Simons Observatory

S.O. has made the forward-looking decision to openly develop much of their software stack. **This is awesome!**

Example: Loading observations of realistic (simulated) data:

https://github.com/simonsobs/sotodlib/blob/master/sotodlib/data/toast_load.py

WARNING: all this is code under active development. It is very specific to the proposed S.O. data format / schema. It is just a concrete example.

realdata

November 1, 2019

1 Considerations for Real Data

This lesson is about...

```
In [1]: # Are you using a special reservation for a workshop?
        # If so, set it here:
        nersc_reservation = None

        # Load common tools for all lessons
        import sys
        sys.path.insert(0, "..")
        from lesson_tools import (
            check_nersc,
            fake_focalplane
        )
        nersc_host, nersc_repo, nersc_resv = check_nersc(reservation=nersc_reservation)

        # Capture C++ output in the jupyter cells
        %reload_ext wurlitzer
```

```
Running on NERSC machine 'cori'
  with access to repos: mp107
Using default repo mp107
```

1.1 Running in Parallel

The NERSC login nodes do not support MPI, so all of the previous examples are running serially. To run in parallel, we can submit a batch job version of the above examples:

```
In [2]: %%writefile realdata.py

        import toast
        from toast.mpi import MPI
```

```
Writing realdata.py
```

In [3]:
```
import subprocess as sp

command = "python realdata.py"
runstr = None

if nersc_host is not None:
    runstr = "srun -N 1 -C haswell -n 32 -c 2 --cpu_bind=cores -t 00:03:00"
    if nersc_resv is not None:
        runstr = "{} --reservation {}".format(runstr, nersc_resv)
else:
    # Just use mpirun
    runstr = "mpirun -np 4"

runcom = "{} {}".format(runstr, command)
print(runcom, flush=True)
sp.check_call(runcom, stderr=sp.STDOUT, shell=True)
```

```
srun -N 1 -C haswell -n 32 -c 2 --cpu_bind=cores -t 00:03:00 python realdata.py
Launched in background. Redirecting stdin to /dev/null
ModuleCmd_Load.c(244):ERROR:105: Unable to locate a modulefile for 'altd'
ModuleCmd_Load.c(244):ERROR:105: Unable to locate a modulefile for 'darshan'
srun: job 25309209 queued and waiting for resources
srun: job 25309209 has been allocated resources
```

Out[3]: 0

mapmaking

November 1, 2019

1 Map Making

In this lesson we cover the mapmaking problem and current and available TOAST mapmaking facilities * `OpMadam` – interface to `libMadam`, a parallel Fortran library for destriping and mapping signal * `OpMapmaker` – nascent implementation of a native TOAST mapmaker with planned support for a host of systematics templates

```
In [1]: # Are you using a special reservation for a workshop?
        # If so, set it here:
        nersc_reservation = None

        # Load common tools for all lessons
        import sys
        sys.path.insert(0, "..")
        from lesson_tools import (
            check_nersc,
            fake_focalplane
        )
        nersc_host, nersc_repo, nersc_resv = check_nersc(reservation=nersc_reservation)

        # Capture C++ output in the jupyter cells
        %reload_ext wurlitzer
```

```
Running on NERSC machine 'cori'
  with access to repos: mp107
Using default repo mp107
```

1.1 Mapmaking basics

(Apologies to the experts. You can freely skip the basics if this is trivial to you)

1.1.1 Binning a map

CMB experiments measure the Stokes I (intensity), Q and U linear polarization components in discrete sky pixels. A detector pointed at sky pixels p registers a linear combination of the three Stokes components:

$$d_p = I_p + \eta \cdot (Q_p \cos 2\psi + U_p \sin 2\psi) + n,$$

where η is the polarization efficiency of the detector, ψ is the polarization sensitive direction and n is the noise. ψ depends on the intrinsic polarization angle of the detector, ψ_0, and the relative orientation of the focalplane and the sky, ψ'. Furthermore, if the detector is observing the sky through a half-wave plate (HWP), then the HWP angle, ω must also be accounted for:

$$\psi = \psi_0 + \alpha + 2\omega$$

Regardless of how ψ is modulated (ψ_0, α or ω), one needs a bare minimum of 3 observations at different angles, ψ, to solve for $m = [I, Q, U]^T$. We encode the pointing weights (1, $\eta \cdot \cos 2\psi$, $\eta \cdot \sin 2\psi$) in a pointing matrix:

$$P = \begin{bmatrix} 1 & \eta \cos 2\psi_1 & \eta \sin 2\psi_1 \\ 1 & \eta \cos 2\psi_2 & \eta \sin 2\psi_2 \\ & \dots & \\ 1 & \eta \cos 2\psi_N & \eta \sin 2\psi_N \end{bmatrix}$$

A vector of samples drawn from m is then

$$d = Pm$$

and we can find a (generalized) least squares solution of m as

$$m = (P^T N^{-1} P)^{-1} P^T N^{-1} d,$$

where we have accounted for non-trivial noise correlations in

$$N = \langle n n^T \rangle.$$

If each detector sample is subject to same, uncorrelated noise fluctuations, N can be dropped. This recovers the regular least squares solution.

An important special case of P results when the angles ψ differ by 45 degrees and come in sets of 4:

```
In [2]: import numpy as np
        psi = np.radians(np.arange(4) * 45)
        print("psi =", np.round(psi, 2))
        P = np.vstack([np.ones_like(psi), np.cos(2*psi), np.sin(2*psi)]).T
        print("P = \n", np.round(P, 3))
        invcov = np.dot(P.T, P)
        print("P^T P = \n", np.round(invcov, 3))
        cov = np.linalg.inv(invcov)
        print("(P^T P)^{-1} = \n", np.round(cov, 3))
```

```
psi = [0.   0.79 1.57 2.36]
P =
 [[ 1.  1.  0.]
 [ 1.  0.  1.]
 [ 1. -1.  0.]
 [ 1. -0. -1.]]
```

```
P^T P =
 [[ 4. -0.  0.]
 [-0.  2.  0.]
 [ 0.  0.  2.]]
(P^T P)^{-1} =
 [[ 0.25  0.   -0.  ]
 [ 0.    0.5  -0.  ]
 [-0.   -0.    0.5 ]]
```

The quantity $P^T P$ is proportional to the noise covariance between IQU. It being diagonal means that the statistical error between them is uncorrelated. The reciprocal condition number of the above matrix is 0.5 which is the theoretical maximum. Low reciprocal condition number would indicate that the IQU solution is degenerate.

The mapmaking formalism in this section can be generalised for multiple sky pixels. Each sky pixel corresponds to its own I, Q and U columns in P. If the noise matrix, N, is diagonal, the pixel covariance matrix, $P^T N^{-1} P$ is 3×3 block diagonal with each pixel being solved independently.

1.1.2 Destriping

arXiv:0907.0367

In its simplest form, the time-ordered data (TOD) can be described as sky signal and a noise term:

$$d = Pm + n.$$

An optimal solution of m requires the sample-sample covariance of n. For that to exist, the noise term needs to be stationary, which is often not the case in presence of systematics. We may decompose n into a set of systematics templates and the actual, stationary noise term:

$$d = Pm + Fa + n.$$

Maximum likelihood solution of the template amplitudes, a, follows from the *destriping equation*:

$$(F^T N^{-1} Z F + C_a^{-1}) a = F^T N^{-1} Z d,$$

where

$$Z = \mathbf{1} - P(P^T N^{-1} P)^{-1} P^T N^{-1}$$

is a projection matrix that removes the sky-synchronous part of the signal and

$$C_a = \langle a a^T \rangle$$

TOAST provides an interface to a destriping library, libMadam, which solves a version of the destriping equation where the templates in F are disjoint steps of a step function. `libMadam` approximates that the covariance between the steps is stationary and can be calculated from the detector noise power spectral densities (PSDs). The remaining noise term, n is approximated as white noise with a diagonal noise covariance, N. Users can just have `libMadam` write out the destriped maps or continue processing the destriped TOD:

$$d' = d - Fa$$

The TOAST native mapmaker under development is designed to be more general. Columns of the template matrix, F can be populated with arbitrary forms, with or without an associated noise prior term, C_a. The templates are MPI-aware so a template may span data across process boundaries (think orbital dipole or far sidelobes).

1.2 Example

In this section we create a TOAST data object with simulated signal and noise and process the data into hit maps, pixels noise matrices and signal maps.

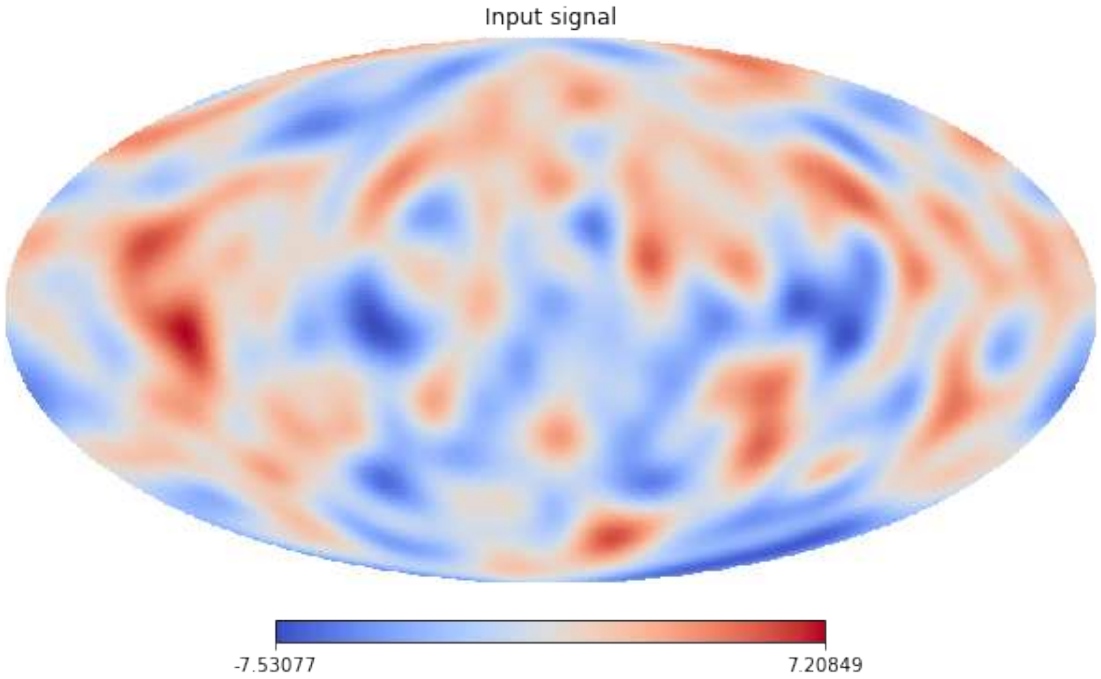
```
In [3]: import toast
        import toast.todmap
        from toast.mpi import MPI

        import numpy as np
        import matplotlib.pyplot as plt

        mpiworld, procs, rank = toast.mpi.get_world()
        comm = toast.mpi.Comm(mpiworld)

        # A pipeline would create the args object with argparse

        class args:
            sample_rate = 10  # Hz
            hwp_rpm = None
            hwp_step_deg = None
            hwp_step_time_s = None
            spin_period_min = 1 # 10
            spin_angle_deg = 20 # 30
            prec_period_min = 100 # 50
            prec_angle_deg = 30 # 65
            coord = "E"
            nside = 64
            nnz = 3
            outdir = "maps"

        # Create a fake focalplane, we could also load one from file.
        # The Focalplane class interprets the focalplane dictionary
        # created by fake_focalplane() but it can also load the information
        # from file.

        focalplane = fake_focalplane(samplerate=args.sample_rate, fknee=0.1, alpha=2)
        detectors = sorted(focalplane.keys())
        detquats = {}
        for d in detectors:
            detquats[d] = focalplane[d]["quat"]
```

```
        nsample = 100000
        start_sample = 0
        start_time = 0
        iobs = 0

        tod = toast.todmap.TODSatellite(
            comm.comm_group,
            detquats,
            nsample,
            coord=args.coord,
            firstsamp=start_sample,
            firsttime=start_time,
            rate=args.sample_rate,
            spinperiod=args.spin_period_min,
            spinangle=args.spin_angle_deg,
            precperiod=args.prec_period_min,
            precangle=args.prec_angle_deg,
            detranks=comm.group_size,
            hwprpm=args.hwp_rpm,
            hwpstep=args.hwp_step_deg,
            hwpsteptime=args.hwp_step_time_s,
        )

        # Constantly slewing precession axis
        precquat = np.empty(4 * tod.local_samples[1], dtype=np.float64).reshape((-1, 4))
        toast.todmap.slew_precession_axis(
            precquat,
            firstsamp=start_sample + tod.local_samples[0],
            samplerate=args.sample_rate,
            degday=360.0 / 365.25,
        )
        tod.set_prec_axis(qprec=precquat)

        noise = toast.pipeline_tools.get_analytic_noise(args, comm, focalplane)

        obs = {}
        obs["name"] = "science_{:05d}".format(iobs)
        obs["tod"] = tod
        obs["intervals"] = None
        obs["baselines"] = None
        obs["noise"] = noise
        obs["id"] = iobs

        data = toast.Data(comm)
        data.obs.append(obs)

TOAST INFO: Creating noise model:  0.00 seconds (1 calls)
```

Create a synthetic Gaussian map to scan as input signal

```
In [4]: import healpy as hp
        lmax = args.nside * 2
        cls = np.zeros([4, lmax + 1])
        cls[0] = 1e0
        sim_map = hp.synfast(cls, args.nside, lmax=lmax, fwhm=np.radians(15), new=True)
        hp.mollview(sim_map[0], cmap="coolwarm", title="Input signal")
        hp.write_map("sim_map.fits", hp.reorder(sim_map, r2n=True), nest=True, overwrite=True)
```

```
Sigma is 382.194810 arcmin (0.111176 rad)
-> fwhm is 900.000000 arcmin
```

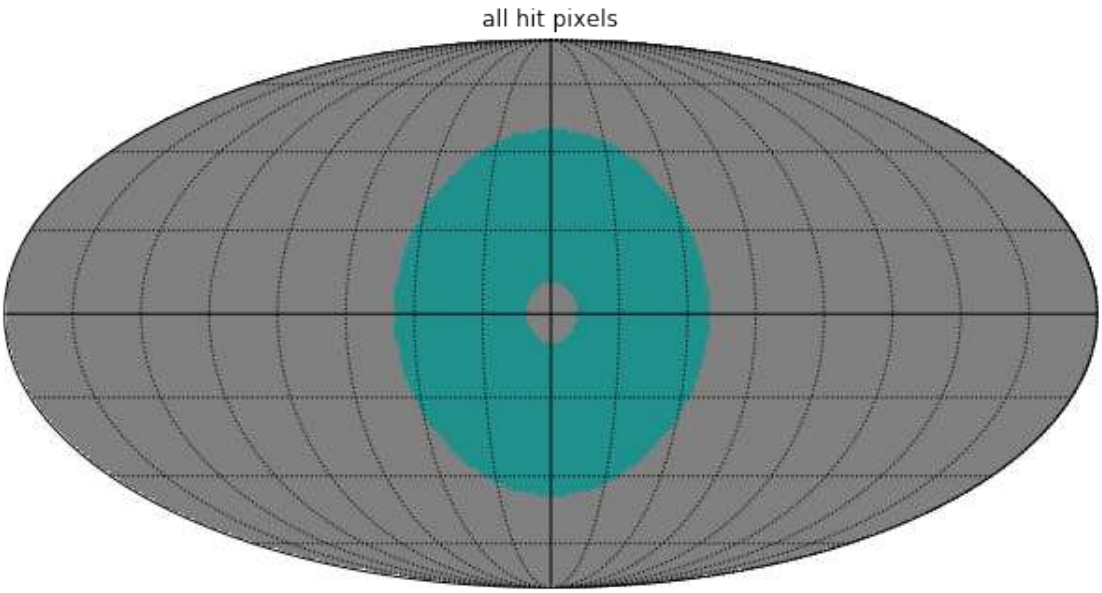


Now simulate sky signal and noise

```
In [5]: name = "signal"
        toast.tod.OpCacheClear(name).exec(data)

        toast.todmap.OpPointingHpix(nside=args.nside, nest=True, mode="IQU").exec(data)
```

```
In [6]: npix = 12 * args.nside ** 2
        hitmap = np.zeros(npix)
        tod = data.obs[0]["tod"]
        for det in tod.local_dets:
```

```
        pixels = tod.cache.reference("pixels_{}".format(det))
        hitmap[pixels] = 1
    hitmap[hitmap == 0] = hp.UNSEEN
    hp.mollview(hitmap, nest=True, title="all hit pixels", cbar=False)
    hp.graticule(22.5, verbose=False)
```



In [7]:
```
    # Scan the signal from a map

    localpix, localsm, subnpix = toast.todmap.get_submaps_nested(data, args.nside, subnside
    distmap = toast.map.DistPixels(
        comm=mpiworld,
        size=12 * args.nside **2,
        nnz=3,
        dtype=np.float32,
        submap=subnpix,
        local=localsm,
    )
    distmap.read_healpix_fits("sim_map.fits")
    toast.todmap.OpSimScan(distmap=distmap, out=name).exec(data)

    # Copy the sky signal

    toast.tod.OpCacheCopy(input=name, output="sky_signal", force=True).exec(data)

    # Simulate noise

    toast.tod.OpSimNoise(out=name, realization=0).exec(data)
```

```
        toast.tod.OpCacheCopy(input=name, output="full_signal", force=True).exec(data)
```

Destripe the signal and make a map. We use the nascent TOAST mapmaker because it can be run in serial mode without MPI. The TOAST mapmaker is still significantly slower so production runs should used `libMadam`.

```
In [8]: mapmaker = toast.todmap.OpMapMaker(
            nside=args.nside,
            nnz=3,
            name=name,
            outdir=args.outdir,
            outprefix="toast_test_",
            baseline_length=10,
            # maskfile=self.maskfile_binary,
            # weightmapfile=self.maskfile_smooth,
            # subharmonic_order=None,
            iter_max=100,
            use_noise_prior=False,
            # precond_width=30,
        )
        mapmaker.exec(data)
```

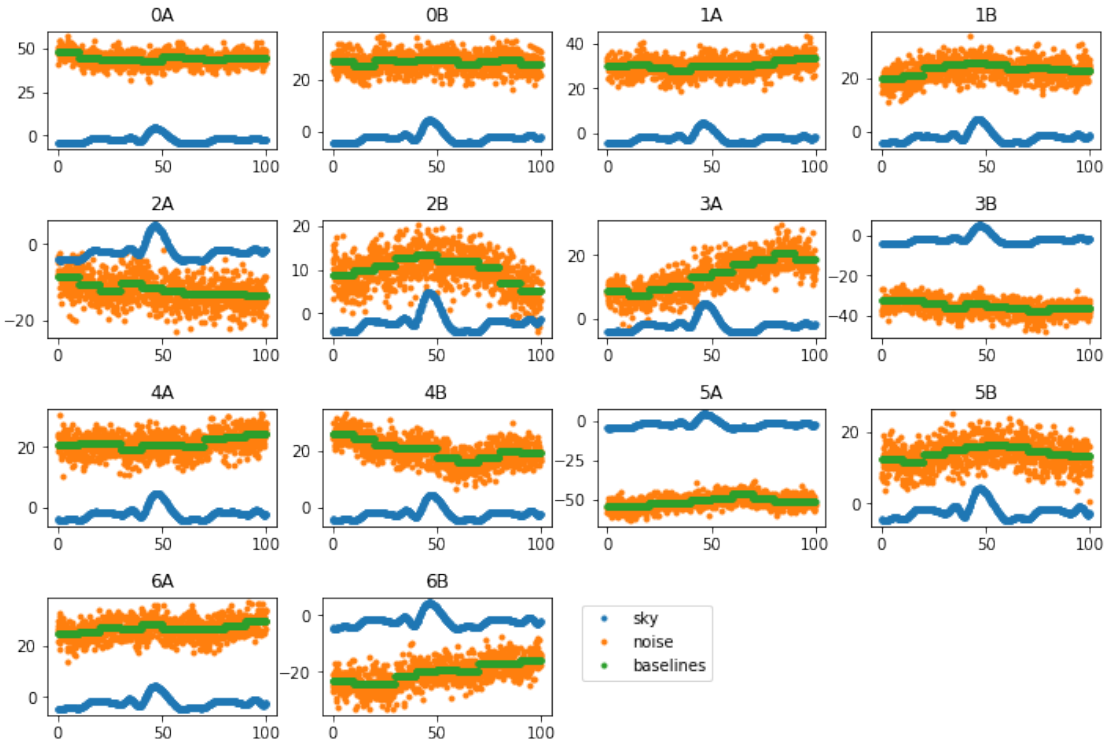
```
TOAST INFO: Flag gaps:  0.00 seconds (1 calls)
TOAST INFO: Get detector weights:  0.00 seconds (1 calls)
TOAST INFO: Identify local submaps:  0.01 seconds (1 calls)
TOAST INFO: Accumulate N_pp'^1:  0.03 seconds (1 calls)
TOAST INFO: All reduce N_pp'^1:  0.00 seconds (1 calls)
TOAST INFO: Wrote hits to maps/toast_test_hits.fits
TOAST INFO: Write hits:  0.01 seconds (1 calls)
TOAST INFO: Wrote inverse white noise covariance to maps/toast_test_invnpp.fits
TOAST INFO: Write N_pp'^1:  0.02 seconds (1 calls)
TOAST INFO: Compute reciprocal condition numbers:  0.22 seconds (1 calls)
TOAST INFO: Wrote reciprocal condition numbers to maps/toast_test_rcond.fits
TOAST INFO: Write rcond:  0.01 seconds (1 calls)
TOAST INFO: Invert N_pp'^1:  0.04 seconds (1 calls)
TOAST INFO: Wrote white noise covariance to maps/toast_test_npp.fits
TOAST INFO: Write N_pp':  0.02 seconds (1 calls)
TOAST INFO:   Build noise-weighted map:  0.00 seconds (0 calls)
TOAST INFO:   Apply noise covariance:  0.00 seconds (0 calls)
TOAST INFO:   Write map to maps/toast_test_binned.fits:  0.00 seconds (0 calls)
TOAST INFO: Initializing offset template, step_length = 10
TOAST INFO: Initialize templates:  0.30 seconds (1 calls)
TOAST INFO: Initialize projection matrix:  0.00 seconds (1 calls)
TOAST INFO: Initialize projection matrix:  0.00 seconds (1 calls)
TOAST INFO: Initialize PCG solver:  0.11 seconds (1 calls)
TOAST INFO: Initial residual: 885433649.2095621
TOAST INFO: Iter =    0 relative residual:   4.6278e-03:  0.22 seconds (1 calls)
```

```
TOAST INFO: Iter =    1 relative residual:   6.4700e-04:  0.14 seconds (1 calls)
TOAST INFO: Iter =    2 relative residual:   1.7046e-04:  0.15 seconds (1 calls)
TOAST INFO: Iter =    3 relative residual:   5.1771e-05:  0.13 seconds (1 calls)
TOAST INFO: Iter =    4 relative residual:   3.9861e-05:  0.14 seconds (1 calls)
TOAST INFO: Iter =    5 relative residual:   1.9196e-05:  0.16 seconds (1 calls)
TOAST INFO: Iter =    6 relative residual:   8.9093e-06:  0.15 seconds (1 calls)
TOAST INFO: Iter =    7 relative residual:   4.3047e-06:  0.15 seconds (1 calls)
TOAST INFO: Iter =    8 relative residual:   5.0802e-06:  0.15 seconds (1 calls)
TOAST INFO: Iter =    9 relative residual:   3.5688e-06:  0.14 seconds (1 calls)
TOAST INFO: Iter =   10 relative residual:   2.6579e-06:  0.14 seconds (1 calls)
TOAST INFO: Iter =   11 relative residual:   1.4963e-06:  0.14 seconds (1 calls)
TOAST INFO: Iter =   12 relative residual:   5.8145e-07:  0.16 seconds (1 calls)
TOAST INFO: Iter =   13 relative residual:   3.3869e-07:  0.16 seconds (1 calls)
TOAST INFO: Iter =   14 relative residual:   3.2408e-07:  0.16 seconds (1 calls)
TOAST INFO: Iter =   15 relative residual:   2.6685e-07:  0.14 seconds (1 calls)
TOAST INFO: Iter =   16 relative residual:   1.5142e-07:  0.15 seconds (1 calls)
TOAST INFO: Iter =   17 relative residual:   8.6206e-08:  0.15 seconds (1 calls)
TOAST INFO: Iter =   18 relative residual:   2.9340e-08:  0.13 seconds (1 calls)
TOAST INFO: Iter =   19 relative residual:   1.4302e-08:  0.15 seconds (1 calls)
TOAST INFO: Iter =   20 relative residual:   8.6289e-09:  0.16 seconds (1 calls)
TOAST INFO: Iter =   21 relative residual:   4.6083e-09:  0.14 seconds (1 calls)
TOAST INFO: Iter =   22 relative residual:   2.2566e-09:  0.14 seconds (1 calls)
TOAST INFO: Iter =   23 relative residual:   1.3089e-09:  0.13 seconds (1 calls)
TOAST INFO: Iter =   24 relative residual:   6.7816e-10:  0.15 seconds (1 calls)
TOAST INFO: Iter =   25 relative residual:   4.7313e-10:  0.13 seconds (1 calls)
TOAST INFO: Iter =   26 relative residual:   3.7748e-10:  0.15 seconds (1 calls)
TOAST INFO: Iter =   27 relative residual:   3.4232e-10:  0.16 seconds (1 calls)
TOAST INFO: Iter =   28 relative residual:   3.2723e-10:  0.16 seconds (1 calls)
TOAST INFO: Iter =   29 relative residual:   2.8890e-10:  0.14 seconds (1 calls)
TOAST INFO: Iter =   30 relative residual:   2.0482e-10:  0.16 seconds (1 calls)
TOAST INFO: Iter =   31 relative residual:   1.1737e-10:  0.15 seconds (1 calls)
TOAST INFO: Iter =   32 relative residual:   1.4366e-10:  0.15 seconds (1 calls)
TOAST INFO: Iter =   33 relative residual:   3.7676e-10:  0.15 seconds (1 calls)
TOAST INFO: Iter =   34 relative residual:   7.5089e-10:  0.14 seconds (1 calls)
TOAST INFO: Iter =   35 relative residual:   1.3755e-09:  0.16 seconds (1 calls)
TOAST INFO: Iter =   36 relative residual:   2.5279e-09:  0.15 seconds (1 calls)
TOAST INFO: Iter =   37 relative residual:   3.5890e-09:  0.15 seconds (1 calls)
TOAST INFO: Iter =   38 relative residual:   3.9316e-09:  0.15 seconds (1 calls)
TOAST INFO: Iter =   39 relative residual:   4.8039e-09:  0.15 seconds (1 calls)
TOAST INFO: Iter =   40 relative residual:   6.0234e-09:  0.18 seconds (1 calls)
TOAST INFO: PCG stalled after 40 iterations:  6.14 seconds (1 calls)
TOAST INFO: 0 : Solution: template amplitudes:
"offset" :
[ 48.40832879  44.45261853  44.15905326 ... -43.11081524 -42.79881051
 -45.01007091]
TOAST INFO: Solve amplitudes:  6.14 seconds (1 calls)
TOAST INFO: Clean TOD:  0.02 seconds (1 calls)
TOAST INFO:   Build noise-weighted map:  0.00 seconds (0 calls)
```

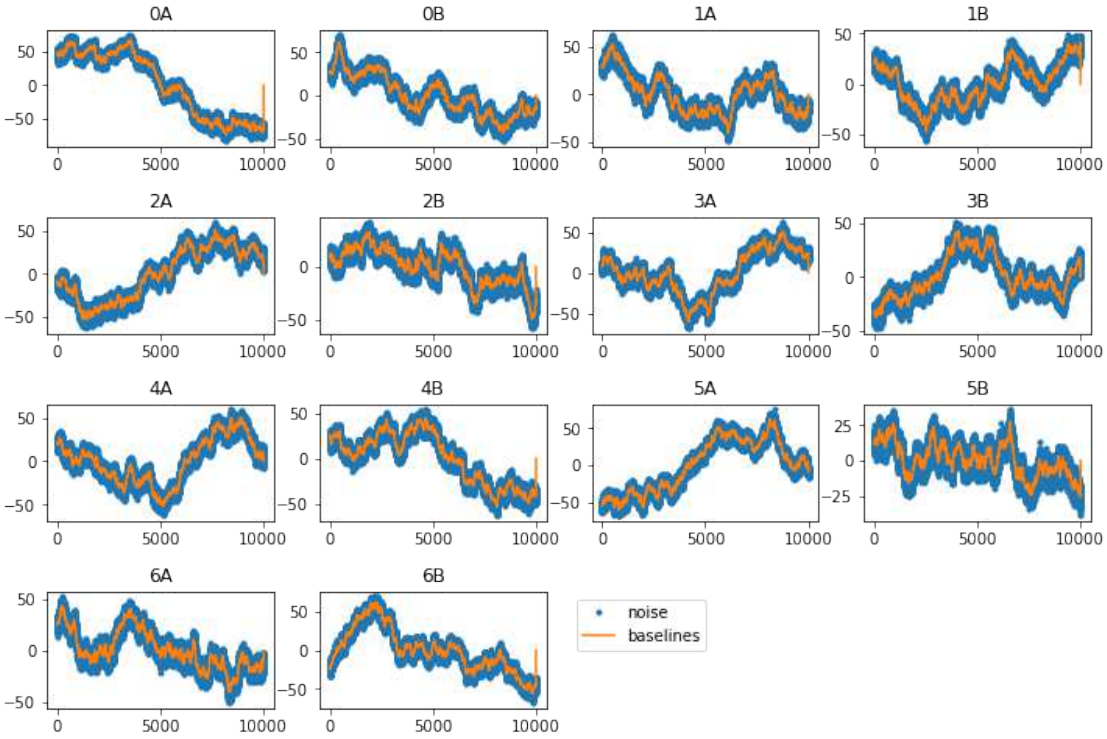
```
TOAST INFO:   Apply noise covariance:  0.00 seconds (0 calls)
TOAST INFO:   Write map to maps/toast_test_destriped.fits:  0.00 seconds (0 calls)
```

Plot a segment of the timelines

```
In [9]: tod = data.obs[0]["tod"]
        times = tod.local_times()

        fig = plt.figure(figsize=[12, 8])
        for idet, det in enumerate(tod.local_dets):
            sky = tod.local_signal(det, "sky_signal")
            full = tod.local_signal(det, "full_signal")
            cleaned = tod.local_signal(det, name)

            ind = slice(0, 1000)
            ax = fig.add_subplot(4, 4, 1 + idet)
            ax.set_title(det)
            ax.plot(times[ind], sky[ind], '.', label="sky", zorder=100)
            ax.plot(times[ind], full[ind] - sky[ind], '.', label="noise")
            ax.plot(times[ind], full[ind] - cleaned[ind], '.', label="baselines")
        ax.legend(bbox_to_anchor=(1.1, 1.00))
        fig.subplots_adjust(hspace=0.6)
```



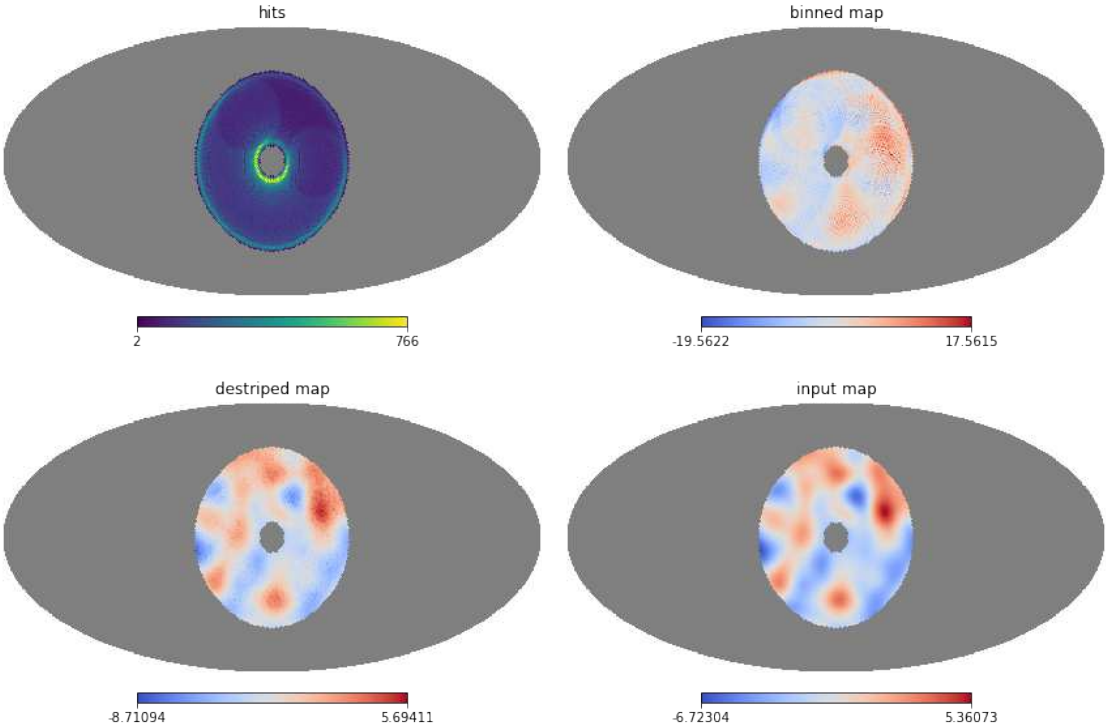
In [10]:
```
fig = plt.figure(figsize=[12, 8])
for idet, det in enumerate(tod.local_dets):
    sky = tod.local_signal(det, "sky_signal")
    full = tod.local_signal(det, "full_signal")
    cleaned = tod.local_signal(det, name)
    ax = fig.add_subplot(4, 4, 1 + idet)
    ax.set_title(det)
    #plt.plot(times[ind], sky[ind], '-', label="signal", zorder=100)
    plt.plot(times, full - sky, '.', label="noise")
    plt.plot(times, full - cleaned, '-', label="baselines")
ax.legend(bbox_to_anchor=(1.1, 1.00))
fig.subplots_adjust(hspace=.6)
```



In [11]:
```
plt.figure(figsize=[12, 8])

hitmap = hp.read_map("maps/toast_test_hits.fits")
hitmap[hitmap == 0] = hp.UNSEEN
hp.mollview(hitmap, sub=[2, 2, 1], title="hits")

binmap = hp.read_map("maps/toast_test_binned.fits")
binmap[binmap == 0] = hp.UNSEEN
hp.mollview(binmap, sub=[2, 2, 2], title="binned map", cmap="coolwarm")
```

```
destriped = hp.read_map("maps/toast_test_destriped.fits")
destriped[destriped == 0] = hp.UNSEEN
hp.mollview(destriped, sub=[2, 2, 3], title="destriped map", cmap="coolwarm")

inmap = hp.read_map("sim_map.fits")
inmap[hitmap == hp.UNSEEN] = hp.UNSEEN
hp.mollview(inmap, sub=[2, 2, 4], title="input map", cmap="coolwarm")
```

```
NSIDE = 64
ORDERING = NESTED in fits file
INDXSCHM = IMPLICIT
Ordering converted to RING
NSIDE = 64
ORDERING = NESTED in fits file
INDXSCHM = IMPLICIT
Ordering converted to RING
NSIDE = 64
ORDERING = NESTED in fits file
INDXSCHM = IMPLICIT
Ordering converted to RING
NSIDE = 64
ORDERING = NESTED in fits file
INDXSCHM = IMPLICIT
Ordering converted to RING
```

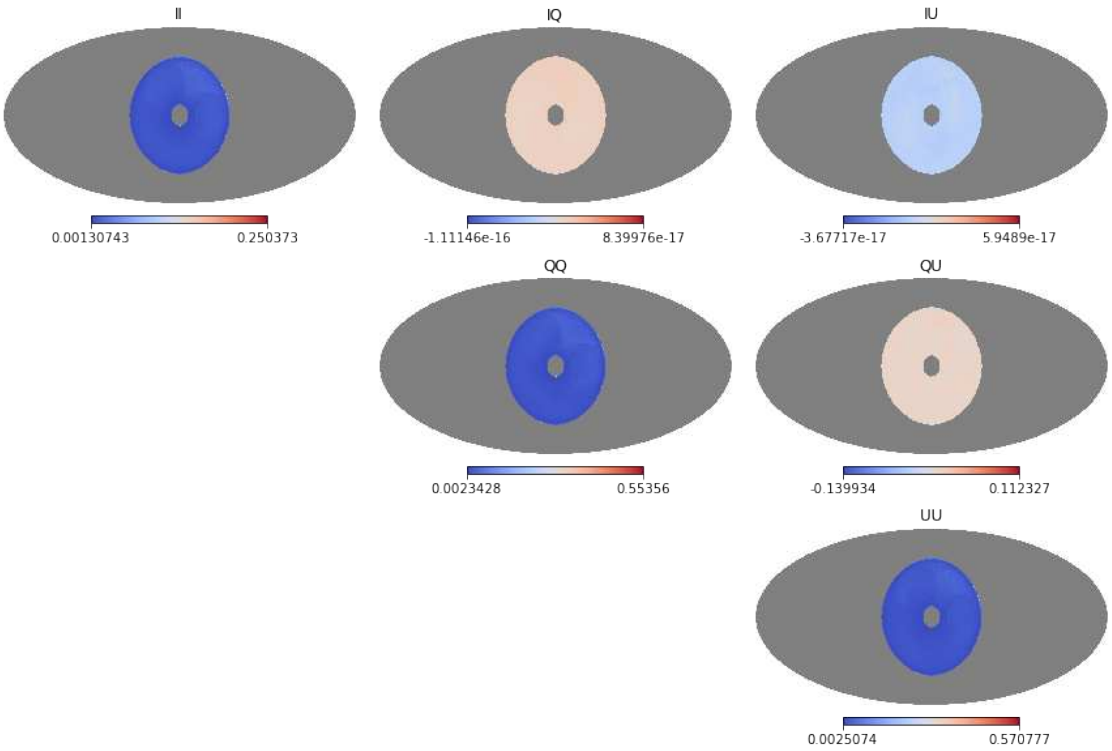


```
In [12]: print(np.sum(hitmap[hitmap != hp.UNSEEN]) / 1400000.0)
```

```
1.0
```

```
In [13]: # Plot the white noise covariance

         plt.figure(figsize=[12, 8])
         wcov = hp.read_map("maps/toast_test_npp.fits", None)
         wcov[:, wcov[0] == 0] = hp.UNSEEN
         hp.mollview(wcov[0], sub=[3, 3, 1], title="II", cmap="coolwarm")
         hp.mollview(wcov[1], sub=[3, 3, 2], title="IQ", cmap="coolwarm")
         hp.mollview(wcov[2], sub=[3, 3, 3], title="IU", cmap="coolwarm")
         hp.mollview(wcov[3], sub=[3, 3, 5], title="QQ", cmap="coolwarm")
         hp.mollview(wcov[4], sub=[3, 3, 6], title="QU", cmap="coolwarm")
         hp.mollview(wcov[5], sub=[3, 3, 9], title="UU", cmap="coolwarm")
```

```
NSIDE = 64
ORDERING = NESTED in fits file
INDXSCHM = IMPLICIT
Ordering converted to RING
Ordering converted to RING
Ordering converted to RING
Ordering converted to RING
Ordering converted to RING
Ordering converted to RING
```

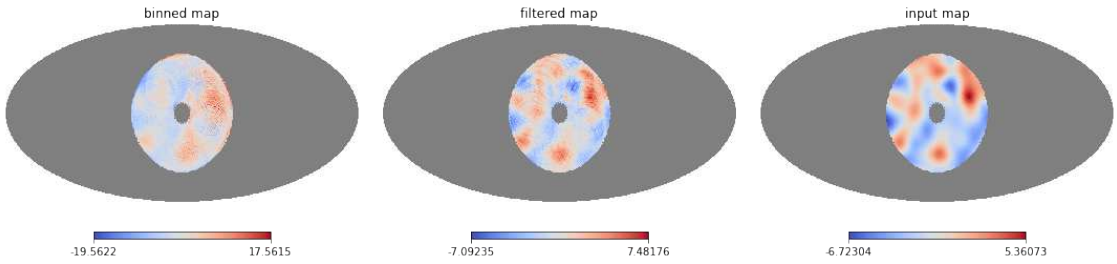


1.3 Filter & bin

A filter-and-bin mapmaker is easily created by combining TOAST filter operators and running the mapmaker without destriping:

```
In [14]: name_in = "full_signal"
         name_out = "full_signal_copy"
         toast.tod.OpCacheCopy(input=name_in, output=name_out, force=True).exec(data)

         toast.tod.OpPolyFilter(order=30, name=name_out).exec(data)

         mapmaker = toast.todmap.OpMapMaker(
             nside=args.nside,
             nnz=3,
             name=name_out,
             outdir=args.outdir,
             outprefix="toast_test_filtered_",
             baseline_length=None,
             # maskfile=self.maskfile_binary,
             # weightmapfile=self.maskfile_smooth,
             # subharmonic_order=None,
             iter_max=100,
             use_noise_prior=False,
             # precond_width=30,
         )
         mapmaker.exec(data)

         plt.figure(figsize=[15, 8])
         binmap = hp.read_map("maps/toast_test_binned.fits")
         filtered_map = hp.read_map("maps/toast_test_filtered_binned.fits")
         binmap[binmap == 0] = hp.UNSEEN

         hp.mollview(binmap, sub=[1, 3, 1], title="binned map", cmap="coolwarm")
         filtered_map[filtered_map == 0] = hp.UNSEEN
         hp.mollview(filtered_map, sub=[1, 3, 2], title="filtered map", cmap="coolwarm")

         inmap = hp.read_map("sim_map.fits")
         inmap[binmap == hp.UNSEEN] = hp.UNSEEN
         hp.mollview(inmap, sub=[1, 3, 3], title="input map", cmap="coolwarm")

TOAST INFO: Flag gaps:  0.00 seconds (1 calls)
TOAST INFO: Get detector weights:  0.00 seconds (1 calls)
TOAST INFO: Identify local submaps:  0.01 seconds (1 calls)
TOAST INFO: Accumulate N_pp'^1:  0.03 seconds (1 calls)
TOAST INFO: All reduce N_pp'^1:  0.00 seconds (1 calls)
```

```
TOAST INFO: Wrote hits to maps/toast_test_filtered_hits.fits
TOAST INFO: Write hits:  0.01 seconds (1 calls)
TOAST INFO: Wrote inverse white noise covariance to maps/toast_test_filtered_invnpp.fits
TOAST INFO: Write N_pp'^1:  0.03 seconds (1 calls)
TOAST INFO: Compute reciprocal condition numbers:  0.01 seconds (1 calls)
TOAST INFO: Wrote reciprocal condition numbers to maps/toast_test_filtered_rcond.fits
TOAST INFO: Write rcond:  0.01 seconds (1 calls)
TOAST INFO: Invert N_pp'^1:  0.01 seconds (1 calls)
TOAST INFO: Wrote white noise covariance to maps/toast_test_filtered_npp.fits
TOAST INFO: Write N_pp':  0.02 seconds (1 calls)
TOAST INFO:   Build noise-weighted map:  0.00 seconds (0 calls)
TOAST INFO:   Apply noise covariance:  0.00 seconds (0 calls)
TOAST INFO:   Write map to maps/toast_test_filtered_binned.fits:  0.00 seconds (0 calls)
TOAST INFO: No templates to fit, no destriping done.
TOAST INFO: Initialize templates:  0.00 seconds (1 calls)
NSIDE = 64
ORDERING = NESTED in fits file
INDXSCHM = IMPLICIT
Ordering converted to RING
NSIDE = 64
ORDERING = NESTED in fits file
INDXSCHM = IMPLICIT
Ordering converted to RING
NSIDE = 64
ORDERING = NESTED in fits file
INDXSCHM = IMPLICIT
Ordering converted to RING
```



pipelines

November 1, 2019

1 Pipelines

TOAST pipelines are standalone Python scripts that apply one or more TOAST operators to existing or simulated data. They typically divide into two major parts: 1. Creating the TOAST `data` object 2. Applying TOAST operators to the `data`

Experiments and simulations typically require specialized methods for creating the `data` object. This part is hard to generalize. For the subsequent data processing TOAST provides a number of convenience functions are kept in toast.pipeline_tools.

In [1]:
```
# Are you using a special reservation for a workshop?
# If so, set it here:
nersc_reservation = None

# Load common tools for all lessons
import sys
sys.path.insert(0, "..")
from lesson_tools import (
    check_nersc,
    fake_focalplane
)
nersc_host, nersc_repo, nersc_resv = check_nersc(reservation=nersc_reservation)

# Capture C++ output in the jupyter cells
%reload_ext wurlitzer
```

```
Running on NERSC machine 'cori'
  with access to repos: mp107
Using default repo mp107
```

1.1 Brief example

We demonstrate these concepts by writing a very simple pipeline that creates its own observations, fills them with noise and applies a polynomial filter.

In [2]:
```
%%writefile my_first_pipeline.py
```

```
import argparse

import numpy as np

# import lesson_tools

import toast
from toast.mpi import MPI
import toast.pipeline_tools


# Fake focaplane is copied here from lesson_tools to avoid import error

def fake_focalplane(
    samplerate=20,
    epsilon=0,
    net=1,
    fmin=0,
    alpha=1,
    fknee=0.05,
    fwhm=30,
    npix=7,
    fov=3.0
):
    """Create a set of fake detectors.

    This generates 7 pixels (14 dets) in a hexagon layout at the boresight
    and with a made up polarization orientation.

    Args:
        None

    Returns:
        (dict):  dictionary of detectors and their properties.

    """
    from toast.tod import hex_pol_angles_radial, hex_pol_angles_qu, hex_layout

    zaxis = np.array([0, 0, 1.0])

    pol_A = hex_pol_angles_qu(npix)
    pol_B = hex_pol_angles_qu(npix, offset=90.0)

    dets_A = hex_layout(npix, fov, "", "", pol_A)
    dets_B = hex_layout(npix, fov, "", "", pol_B)

    dets = dict()
    for p in range(npix):
```

```
            pstr = "{:01d}".format(p)
            for d, layout in zip(["A", "B"], [dets_A, dets_B]):
                props = dict()
                props["quat"] = layout[pstr]["quat"]
                props["epsilon"] = epsilon
                props["rate"] = samplerate
                props["alpha"] = alpha
                props["NET"] = net
                props["fmin"] = fmin
                props["fknee"] = fknee
                props["fwhm_arcmin"] = fwhm
                dname = "{}{}".format(pstr, d)
                dets[dname] = props
    return dets


def parse_arguments(comm):
    """ Declare and parse command line arguments

    """
    parser = argparse.ArgumentParser(fromfile_prefix_chars="@")

    # pipeline_tools provides arguments to supported operators
    toast.pipeline_tools.add_noise_args(parser)
    toast.pipeline_tools.add_polyfilter_args(parser)

    # The pipeline may add its own arguments
    parser.add_argument(
        "--nsample",
        required=False,
        default=10000,
        type=np.int,
        help="Length of the simulation",
    )

    args = parser.parse_args()
    return args


def create_observations(comm, args):
    """ Create the TOAST data object using `args`

    This method will look very different for different pipelines.
    """
    focalplane = fake_focalplane(samplerate=args.sample_rate, fknee=1.0)
    detnames = list(sorted(focalplane.keys()))
    detquats = {x: focalplane[x]["quat"] for x in detnames}
    tod = toast.tod.TODCache(None, detnames, args.nsample, detquats=detquats)
```

```
        # Write some auxiliary data
        tod.write_times(stamps=np.arange(args.nsample) / args.sample_rate)
        tod.write_common_flags(flags=np.zeros(args.nsample, dtype=np.uint8))
        for d in detnames:
            tod.write_flags(detector=d, flags=np.zeros(args.nsample, dtype=np.uint8))

        noise = toast.pipeline_tools.get_analytic_noise(args, comm, focalplane)

        observation = {"tod" : tod, "noise" : noise, "name" : "obs-000", "id" : 0}

        data = toast.Data(comm)
        data.obs.append(observation)

        return data


    def main():
        """ The `main` will instantiate and process the data.

        """
        mpiworld, procs, rank, comm = toast.pipeline_tools.get_comm()
        args = parse_arguments(comm)
        data = create_observations(comm, args)

        mc = 0
        name = "signal"
        tod = data.obs[0]["tod"]
        det = tod.local_dets[0]

        # Simulate noise, write out one detector
        toast.pipeline_tools.simulate_noise(args, comm, data, mc, cache_prefix=name)
        tod.local_signal(det, name).tofile("noise.npy")

        # Apply polyfilter, write the filtered data
        toast.pipeline_tools.apply_polyfilter(args, comm, data, cache_name=name)
        tod.local_signal(det, name).tofile("filtered.npy")


    if __name__ == "__main__":
        main()
```

```
Overwriting my_first_pipeline.py
```

In [3]:
```
! python3 my_first_pipeline.py --help
```

```
<toast.Environment
  Source code version = 2.3.1.dev1428
```

```
  Logging level = INFO
  Handling enabled for 0 signals:
  Max threads = 4
  MPI build enabled
  MPI runtime disabled
  Cannot use MPI on NERSC login nodes
>
TOAST INFO: Running serially with one process at 2019-10-22 12:38:38.584206
usage: my_first_pipeline.py [-h] [--noise] [--simulate-noise] [--no-noise]
                            [--no-simulate-noise] [--sample-rate SAMPLE_RATE]
                            [--polyfilter] [--no-polyfilter]
                            [--poly-order POLY_ORDER]
                            [--common-flag-mask COMMON_FLAG_MASK]
                            [--nsample NSAMPLE]

optional arguments:
  -h, --help            show this help message and exit
  --noise               Add simulated noise
  --simulate-noise      Add simulated noise
  --no-noise            Do not add simulated noise
  --no-simulate-noise   Do not add simulated noise
  --sample-rate SAMPLE_RATE
                        Detector sample rate (Hz)
  --polyfilter          Apply polynomial filter
  --no-polyfilter       Do not apply polynomial filter
  --poly-order POLY_ORDER
                        Polynomial order for the polyfilter
  --common-flag-mask COMMON_FLAG_MASK
                        Common flag mask
  --nsample NSAMPLE     Length of the simulation
```

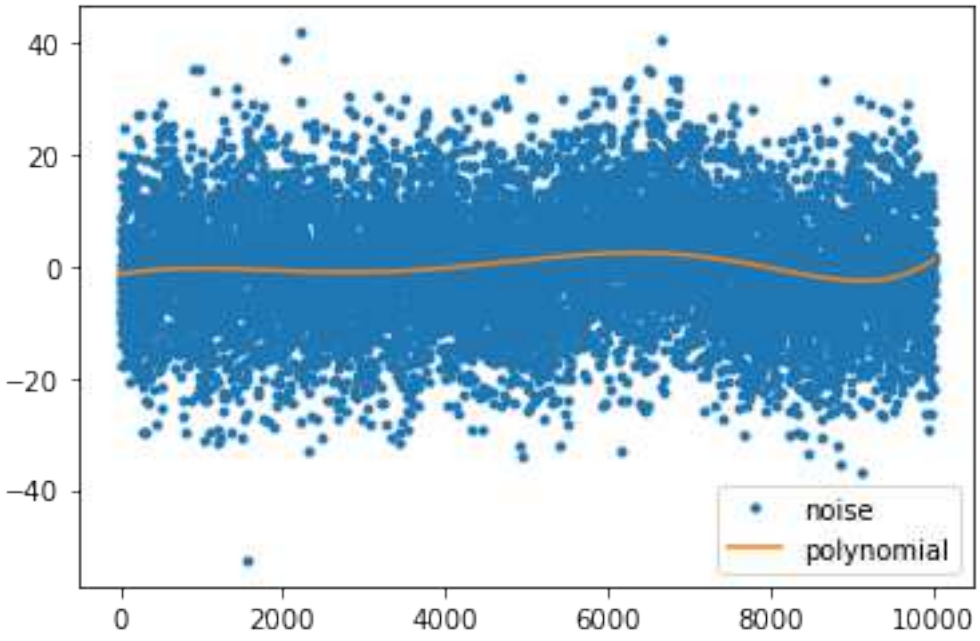
```
In [4]: ! python3 my_first_pipeline.py --simulate-noise --polyfilter --poly-order 6 --nsample
```

```
<toast.Environment
  Source code version = 2.3.1.dev1428
  Logging level = INFO
  Handling enabled for 0 signals:
  Max threads = 4
  MPI build enabled
  MPI runtime disabled
  Cannot use MPI on NERSC login nodes
>
TOAST INFO: Running serially with one process at 2019-10-22 12:38:40.821362
TOAST INFO: Creating noise model:  0.00 seconds (1 calls)
TOAST INFO: Simulating noise
TOAST INFO: Simulate noise:  0.10 seconds (1 calls)
TOAST INFO: Polyfiltering signal
```

```
TOAST INFO: Polynomial filtering:  0.34 seconds (1 calls)
```

```
In [5]: import matplotlib.pyplot as plt
        %matplotlib inline
        import numpy as np

        noise = np.fromfile("noise.npy")
        filtered = np.fromfile("filtered.npy")
        plt.plot(noise, '.', label="noise")
        plt.plot(noise - filtered, '-', label="polynomial")
        plt.legend()
```

```
Out[5]: <matplotlib.legend.Legend at 0x2aaaf0e44240>
```



1.2 Satellite simulation pipeline

pipelines/toast_satellite_sim.py

```
In [6]: ! toast_satellite_sim.py --help
```

```
<toast.Environment
  Source code version = 2.3.1.dev1428
  Logging level = INFO
  Handling enabled for 0 signals:
  Max threads = 4
```

```
  MPI build enabled
  MPI runtime disabled
  Cannot use MPI on NERSC login nodes
>
TOAST INFO: Running serially with one process at 2019-10-22 12:38:44.474550
usage: toast_satellite_sim.py [-h] [--group-size GROUP_SIZE] [--day-maps]
                              [--no-day-maps] [--season-maps]
                              [--no-season-maps] [--nside NSIDE]
                              [--coord COORD] [--single-precision-pointing]
                              [--no-single-precision-pointing]
                              [--common-flag-mask COMMON_FLAG_MASK] [--flush]
                              [--tidas TIDAS] [--spt3g SPT3G] [--dipole]
                              [--no-dipole] [--dipole-mode DIPOLE_MODE]
                              [--dipole-solar-speed-kms DIPOLE_SOLAR_SPEED_KMS]
                              [--dipole-solar-gal-lat-deg DIPOLE_SOLAR_GAL_LAT_DEG]
                              [--dipole-solar-gal-lon-deg DIPOLE_SOLAR_GAL_LON_DEG]
                              [--pysm-model PYSM_MODEL] [--pysm-apply-beam]
                              [--no-pysm-apply-beam]
                              [--pysm-precomputed-cmb-K_CMB PYSM_PRECOMPUTED_CMB_K_CMB]
                              [--MC-start MC_START] [--MC-count MC_COUNT]
                              [--noise] [--simulate-noise] [--no-noise]
                              [--no-simulate-noise]
                              [--sample-rate SAMPLE_RATE]
                              [--start-time START_TIME]
                              [--spin-period-min SPIN_PERIOD_MIN]
                              [--spin-angle-deg SPIN_ANGLE_DEG]
                              [--prec-period-min PREC_PERIOD_MIN]
                              [--prec-angle-deg PREC_ANGLE_DEG]
                              [--obs-time-h OBS_TIME_H] [--gap-h GAP_H]
                              [--obs-num OBS_NUM] [--hwp-rpm HWP_RPM]
                              [--hwp-step-deg HWP_STEP_DEG]
                              [--hwp-step-time-s HWP_STEP_TIME_S]
                              [--outdir OUTDIR] [--debug]
                              [--madam-prefix MADAM_PREFIX]
                              [--madam-iter-max MADAM_ITER_MAX]
                              [--madam-precond-width MADAM_PRECOND_WIDTH]
                              [--madam-precond-width-min MADAM_PRECOND_WIDTH_MIN]
                              [--madam-precond-width-max MADAM_PRECOND_WIDTH_MAX]
                              [--madam-baseline-length MADAM_BASELINE_LENGTH]
                              [--madam-baseline-order MADAM_BASELINE_ORDER]
                              [--madam-noisefilter]
                              [--madam-parfile MADAM_PARFILE]
                              [--madam-allreduce] [--no-madam-allreduce]
                              [--madam-concatenate-messages]
                              [--no-madam-concatenate-messages] [--destripe]
                              [--no-destripe] [--binmap] [--no-binmap]
                              [--hits] [--no-hits] [--wcov] [--no-wcov]
                              [--wcov-inv] [--no-wcov-inv] [--conserve-memory]
```

```
                              [--no-conserve-memory] [--zip] [--no-zip]
                              [--madam] [--no-madam] [--focalplane FOCALPLANE]
                              [--gain GAIN]

Simulate satellite boresight pointing and make a map.

optional arguments:
  -h, --help            show this help message and exit
  --group-size GROUP_SIZE
                        Size of a process group assigned to an observation
  --day-maps            Enable daily maps
  --no-day-maps         Disable daily maps
  --season-maps         Enable season maps
  --no-season-maps      Disable season maps
  --nside NSIDE         Healpix NSIDE
  --coord COORD         Sky coordinate system [C,E,G]
  --single-precision-pointing
                        Use single precision for pointing in memory.
  --no-single-precision-pointing
                        Use single precision for pointing in memory.
  --common-flag-mask COMMON_FLAG_MASK
                        Common flag mask
  --flush               Flush every print statement.
  --tidas TIDAS         Output TIDAS export path
  --spt3g SPT3G         Output SPT3G export path
  --dipole              Add simulated dipole
  --no-dipole           Do not add simulated dipole
  --dipole-mode DIPOLE_MODE
                        Dipole mode is 'total', 'orbital' or 'solar'
  --dipole-solar-speed-kms DIPOLE_SOLAR_SPEED_KMS
                        Solar system speed [km/s]
  --dipole-solar-gal-lat-deg DIPOLE_SOLAR_GAL_LAT_DEG
                        Solar system speed galactic latitude [degrees]
  --dipole-solar-gal-lon-deg DIPOLE_SOLAR_GAL_LON_DEG
                        Solar system speed galactic longitude[degrees]
  --pysm-model PYSM_MODEL
                        Comma separated models for on-the-fly PySM simulation,
                        e.g. "s1,d6,f1,a2"
  --pysm-apply-beam     Convolve sky with detector beam
  --no-pysm-apply-beam  Do not convolve sky with detector beam.
  --pysm-precomputed-cmb-K_CMB PYSM_PRECOMPUTED_CMB_K_CMB
                        Precomputed CMB map for PySM in K_CMBit overrides any
                        model defined in pysm_model"
  --MC-start MC_START   First Monte Carlo noise realization
  --MC-count MC_COUNT   Number of Monte Carlo noise realizations
  --noise               Add simulated noise
  --simulate-noise      Add simulated noise
  --no-noise            Do not add simulated noise
```

```
--no-simulate-noise   Do not add simulated noise
--sample-rate SAMPLE_RATE
                      Detector sample rate (Hz)
--start-time START_TIME
                      The overall start time of the simulation
--spin-period-min SPIN_PERIOD_MIN
                      The period (in minutes) of the rotation about the spin
                      axis
--spin-angle-deg SPIN_ANGLE_DEG
                      The opening angle (in degrees) of the boresight from
                      the spin axis
--prec-period-min PREC_PERIOD_MIN
                      The period (in minutes) of the rotation about the
                      precession axis
--prec-angle-deg PREC_ANGLE_DEG
                      The opening angle (in degrees) of the spin axis from
                      the precession axis
--obs-time-h OBS_TIME_H
                      Number of hours in one science observation
--gap-h GAP_H         Cooler cycle time in hours between science obs
--obs-num OBS_NUM     Number of complete observations
--hwp-rpm HWP_RPM     The rate (in RPM) of the HWP rotation
--hwp-step-deg HWP_STEP_DEG
                      For stepped HWP, the angle in degrees of each step
--hwp-step-time-s HWP_STEP_TIME_S
                      For stepped HWP, the time in seconds between steps
--outdir OUTDIR       Output directory
--debug               Write diagnostics
--madam-prefix MADAM_PREFIX
                      Output map prefix
--madam-iter-max MADAM_ITER_MAX
                      Maximum number of CG iterations in Madam
--madam-precond-width MADAM_PRECOND_WIDTH
                      Width of the Madam band preconditioner
--madam-precond-width-min MADAM_PRECOND_WIDTH_MIN
                      Minimum width of the Madam band preconditioner
--madam-precond-width-max MADAM_PRECOND_WIDTH_MAX
                      Maximum width of the Madam band preconditioner
--madam-baseline-length MADAM_BASELINE_LENGTH
                      Destriping baseline length (seconds)
--madam-baseline-order MADAM_BASELINE_ORDER
                      Destriping baseline polynomial order
--madam-noisefilter   Destripe with the noise filter enabled
--madam-parfile MADAM_PARFILE
                      Madam parameter file
--madam-allreduce     Use the allreduce commucation pattern in Madam
--no-madam-allreduce  Do not use the allreduce commucation pattern in Madam
--madam-concatenate-messages
```

```
                        Use the alltoallv commucation pattern in Madam
  --no-madam-concatenate-messages
                        Use the point-to-point commucation pattern in Madam
  --destripe            Write destriped maps [default]
  --no-destripe         Do not write destriped maps
  --binmap              Write binned maps [default]
  --no-binmap           Do not write binned maps
  --hits                Write hit maps [default]
  --no-hits             Do not write hit maps
  --wcov                Write white noise covariance [default]
  --no-wcov             Do not write white noise covariance
  --wcov-inv            Write inverse white noise covariance [default]
  --no-wcov-inv         Do not write inverse white noise covariance
  --conserve-memory     Conserve memory when staging libMadam buffers
                        [default]
  --no-conserve-memory  Do not conserve memory when staging libMadam buffers
  --zip                 Compress the map outputs
  --no-zip              Do not compress the map outputs
  --madam               Use libmadam for map-making
  --no-madam            Do not use libmadam for map-making [default]
  --focalplane FOCALPLANE
                        Pickle file containing a dictionary of detector
                        properties. The keys of this dict are the detector
                        names, and each value is also a dictionary with keys
                        "quat" (4 element ndarray), "fwhm" (float, arcmin),
                        "fknee" (float, Hz), "alpha" (float), and "NET"
                        (float). For optional plotting, the key "color" can
                        specify a valid matplotlib color string.
  --gain GAIN           Calibrate the input timelines with a set of gains from
                        aFITS file containing 3 extensions:HDU named DETECTORS
                        : table with list of detector names in a column named
                        DETECTORSHDU named TIME: table with common timestamps
                        column named TIMEHDU named GAINS: 2D image of floats
                        with one row per detector and one column per value.
```

`main` without the profiling commands:

```
def main():
    env = Environment.get()

    mpiworld, procs, rank, comm = get_comm()
    args, comm, groupsize = parse_arguments(comm, procs)

    # Parse options

    if comm.world_rank == 0:
        os.makedirs(args.outdir, exist_ok=True)
```

```
focalplane, gain, detweights = load_focalplane(args, comm)

data = create_observations(args, comm, focalplane, groupsize)

expand_pointing(args, comm, data)

localpix, localsm, subnpix = get_submaps(args, comm, data)

signalname = None
skyname = simulate_sky_signal(
    args, comm, data, [focalplane], subnpix, localsm, "signal"
)
if skyname is not None:
    signalname = skyname

diponame = simulate_dipole(args, comm, data, "signal")
if diponame is not None:
    signalname = diponame

# Mapmaking

if not args.use_madam:
    # THIS BRANCH WILL SOON BE REPLACED WITH THE NATIVE MAPMAKER
else:
    # Initialize madam parameters

    madampars = setup_madam(args)

    # Loop over Monte Carlos

    firstmc = args.MC_start
    nmc = args.MC_count

    for mc in range(firstmc, firstmc + nmc):
        # create output directory for this realization
        outpath = os.path.join(args.outdir, "mc_{:03d}".format(mc))

        simulate_noise(args, comm, data, mc, "tot_signal", overwrite=True)

        # add sky signal
        add_signal(args, comm, data, "tot_signal", signalname)

        if gain is not None:
            op_apply_gain = OpApplyGain(gain, name="tot_signal")
            op_apply_gain.exec(data)

        apply_madam(args, comm, data, madampars, outpath, detweights, "tot_signal")
```

1.3 Ground simulation pipeline

pipelines/toast_ground_sim.py

```
In [7]: ! toast_ground_sim.py --help
```

```
<toast.Environment
  Source code version = 2.3.1.dev1428
  Logging level = INFO
  Handling enabled for 0 signals:
  Max threads = 4
  MPI build enabled
  MPI runtime disabled
  Cannot use MPI on NERSC login nodes
>
TOAST INFO: Running serially with one process at 2019-10-22 12:38:46.760341
usage: toast_ground_sim.py [-h] [--group-size GROUP_SIZE] [--day-maps]
                           [--no-day-maps] [--season-maps] [--no-season-maps]
                           [--debug] [--no-debug] [--scan-rate SCAN_RATE]
                           [--scan-accel SCAN_ACCEL]
                           [--sun-angle-min SUN_ANGLE_MIN] --schedule SCHEDULE
                           [--weather WEATHER] [--timezone TIMEZONE]
                           [--sample-rate SAMPLE_RATE] [--coord COORD]
                           [--split-schedule SPLIT_SCHEDULE] [--sort-schedule]
                           [--no-sort-schedule] [--hwp-rpm HWP_RPM]
                           [--hwp-step-deg HWP_STEP_DEG]
                           [--hwp-step-time-s HWP_STEP_TIME_S] [--nside NSIDE]
                           [--single-precision-pointing]
                           [--no-single-precision-pointing]
                           [--common-flag-mask COMMON_FLAG_MASK] [--flush]
                           [--polyfilter] [--no-polyfilter]
                           [--poly-order POLY_ORDER] [--groundfilter]
                           [--no-groundfilter] [--ground-order GROUND_ORDER]
                           [--atmosphere] [--simulate-atmosphere]
                           [--no-atmosphere] [--no-simulate-atmosphere]
                           [--focalplane-radius-deg FOCALPLANE_RADIUS_DEG]
                           [--atm-verbosity ATM_VERBOSITY]
                           [--atm-lmin-center ATM_LMIN_CENTER]
                           [--atm-lmin-sigma ATM_LMIN_SIGMA]
                           [--atm-lmax-center ATM_LMAX_CENTER]
                           [--atm-lmax-sigma ATM_LMAX_SIGMA]
                           [--atm-gain ATM_GAIN] [--atm-zatm ATM_ZATM]
                           [--atm-zmax ATM_ZMAX] [--atm-xstep ATM_XSTEP]
                           [--atm-ystep ATM_YSTEP] [--atm-zstep ATM_ZSTEP]
                           [--atm-nelem-sim-max ATM_NELEM_SIM_MAX]
                           [--atm-wind-dist ATM_WIND_DIST]
                           [--atm-z0-center ATM_Z0_CENTER]
                           [--atm-z0-sigma ATM_Z0_SIGMA]
                           [--atm-T0-center ATM_T0_CENTER]
```

```
                              [--atm-T0-sigma ATM_T0_SIGMA]
                              [--atm-cache ATM_CACHE] [--noise]
                              [--simulate-noise] [--no-noise]
                              [--no-simulate-noise] [--gainscrambler]
                              [--no-gainscrambler] [--gain-sigma GAIN_SIGMA]
                              [--madam-prefix MADAM_PREFIX]
                              [--madam-iter-max MADAM_ITER_MAX]
                              [--madam-precond-width MADAM_PRECOND_WIDTH]
                              [--madam-precond-width-min MADAM_PRECOND_WIDTH_MIN]
                              [--madam-precond-width-max MADAM_PRECOND_WIDTH_MAX]
                              [--madam-baseline-length MADAM_BASELINE_LENGTH]
                              [--madam-baseline-order MADAM_BASELINE_ORDER]
                              [--madam-noisefilter]
                              [--madam-parfile MADAM_PARFILE] [--madam-allreduce]
                              [--no-madam-allreduce]
                              [--madam-concatenate-messages]
                              [--no-madam-concatenate-messages] [--destripe]
                              [--no-destripe] [--binmap] [--no-binmap] [--hits]
                              [--no-hits] [--wcov] [--no-wcov] [--wcov-inv]
                              [--no-wcov-inv] [--conserve-memory]
                              [--no-conserve-memory] [--input-map INPUT_MAP]
                              [--pysm-model PYSM_MODEL] [--pysm-apply-beam]
                              [--no-pysm-apply-beam]
                              [--pysm-precomputed-cmb-K_CMB PYSM_PRECOMPUTED_CMB_K_CMB]
                              [--ground-map GROUND_MAP]
                              [--ground-nside GROUND_NSIDE]
                              [--ground-fwhm-deg GROUND_FWHM_DEG]
                              [--ground-lmax GROUND_LMAX]
                              [--ground-scale GROUND_SCALE]
                              [--ground-power GROUND_POWER] [--simulate-ground]
                              [--no-simulate-ground] [--tidas TIDAS]
                              [--spt3g SPT3G] [--MC-start MC_START]
                              [--MC-count MC_COUNT] [--outdir OUTDIR]
                              [--focalplane FOCALPLANE] --freq FREQ

Simulate ground-based boresight pointing. Simulate atmosphere and make maps
for some number of noise Monte Carlos.

optional arguments:
  -h, --help            show this help message and exit
  --group-size GROUP_SIZE
                        Size of a process group assigned to an observation
  --day-maps            Enable daily maps
  --no-day-maps         Disable daily maps
  --season-maps         Enable season maps
  --no-season-maps      Disable season maps
  --debug               Enable extra debugging outputs
  --no-debug            Disable extra debugging outputs
```

```
--scan-rate SCAN_RATE
                      Scanning rate [deg / s]
--scan-accel SCAN_ACCEL
                      Scanning rate change [deg / s^2]
--sun-angle-min SUN_ANGLE_MIN
                      Minimum azimuthal distance between the Sun and the
                      bore sight [deg]
--schedule SCHEDULE   Comma-separated list CES schedule files (from
                      toast_ground_schedule.py)
--weather WEATHER     Comma-separated list of TOAST weather files for every
                      schedule. Repeat the same file if the schedules share
                      observing site.
--timezone TIMEZONE   Offset to apply to MJD to separate days [hours]
--sample-rate SAMPLE_RATE
                      Detector sample rate (Hz)
--coord COORD         Sky coordinate system [C,E,G]
--split-schedule SPLIT_SCHEDULE
                      Only use a subset of the schedule. The argument is a
                      string of the form "[isplit],[nsplit]" and only
                      observations that satisfy scan modulo nsplit == isplit
                      are included
--sort-schedule       Reorder the observing schedule so that observations of
                      thesame patch are consecutive. This will reduce the
                      sky area observed by individual process groups.
--no-sort-schedule    Do not reorder the observing schedule so that
                      observations of thesame patch are consecutive.
--hwp-rpm HWP_RPM     The rate (in RPM) of the HWP rotation
--hwp-step-deg HWP_STEP_DEG
                      For stepped HWP, the angle in degrees of each step
--hwp-step-time-s HWP_STEP_TIME_S
                      For stepped HWP, the time in seconds between steps
--nside NSIDE         Healpix NSIDE
--single-precision-pointing
                      Use single precision for pointing in memory.
--no-single-precision-pointing
                      Use single precision for pointing in memory.
--common-flag-mask COMMON_FLAG_MASK
                      Common flag mask
--flush               Flush every print statement.
--polyfilter          Apply polynomial filter
--no-polyfilter       Do not apply polynomial filter
--poly-order POLY_ORDER
                      Polynomial order for the polyfilter
--groundfilter        Apply ground filter
--no-groundfilter     Do not apply ground filter
--ground-order GROUND_ORDER
                      Ground template order
--atmosphere          Add simulated atmoshere
```

```
--simulate-atmosphere
                      Add simulated atmoshere
--no-atmosphere       Do not add simulated atmosphere
--no-simulate-atmosphere
                      Do not add simulated atmosphere
--focalplane-radius-deg FOCALPLANE_RADIUS_DEG
                      Override focal plane radius [deg]
--atm-verbosity ATM_VERBOSITY
                      Atmospheric sim verbosity level
--atm-lmin-center ATM_LMIN_CENTER
                      Kolmogorov turbulence dissipation scale center
--atm-lmin-sigma ATM_LMIN_SIGMA
                      Kolmogorov turbulence dissipation scale sigma
--atm-lmax-center ATM_LMAX_CENTER
                      Kolmogorov turbulence injection scale center
--atm-lmax-sigma ATM_LMAX_SIGMA
                      Kolmogorov turbulence injection scale sigma
--atm-gain ATM_GAIN   Atmospheric gain factor.
--atm-zatm ATM_ZATM   atmosphere extent for temperature profile
--atm-zmax ATM_ZMAX   atmosphere extent for water vapor integration
--atm-xstep ATM_XSTEP
                      size of volume elements in X direction
--atm-ystep ATM_YSTEP
                      size of volume elements in Y direction
--atm-zstep ATM_ZSTEP
                      size of volume elements in Z direction
--atm-nelem-sim-max ATM_NELEM_SIM_MAX
                      controls the size of the simulation slices
--atm-wind-dist ATM_WIND_DIST
                      Maximum wind drift to simulate without discontinuity
--atm-z0-center ATM_Z0_CENTER
                      central value of the water vapor distribution
--atm-z0-sigma ATM_Z0_SIGMA
                      sigma of the water vapor distribution
--atm-T0-center ATM_T0_CENTER
                      central value of the temperature distribution
--atm-T0-sigma ATM_T0_SIGMA
                      sigma of the temperature distribution
--atm-cache ATM_CACHE
                      Atmosphere cache directory
--noise               Add simulated noise
--simulate-noise      Add simulated noise
--no-noise            Do not add simulated noise
--no-simulate-noise   Do not add simulated noise
--gainscrambler       Add simulated noise
--no-gainscrambler    Do not add simulated noise
--gain-sigma GAIN_SIGMA
                      Simulated gain fluctuation amplitude
```

```
--madam-prefix MADAM_PREFIX
                      Output map prefix
--madam-iter-max MADAM_ITER_MAX
                      Maximum number of CG iterations in Madam
--madam-precond-width MADAM_PRECOND_WIDTH
                      Width of the Madam band preconditioner
--madam-precond-width-min MADAM_PRECOND_WIDTH_MIN
                      Minimum width of the Madam band preconditioner
--madam-precond-width-max MADAM_PRECOND_WIDTH_MAX
                      Maximum width of the Madam band preconditioner
--madam-baseline-length MADAM_BASELINE_LENGTH
                      Destriping baseline length (seconds)
--madam-baseline-order MADAM_BASELINE_ORDER
                      Destriping baseline polynomial order
--madam-noisefilter   Destripe with the noise filter enabled
--madam-parfile MADAM_PARFILE
                      Madam parameter file
--madam-allreduce     Use the allreduce commucation pattern in Madam
--no-madam-allreduce  Do not use the allreduce commucation pattern in Madam
--madam-concatenate-messages
                      Use the alltoallv commucation pattern in Madam
--no-madam-concatenate-messages
                      Use the point-to-point commucation pattern in Madam
--destripe            Write destriped maps [default]
--no-destripe         Do not write destriped maps
--binmap              Write binned maps [default]
--no-binmap           Do not write binned maps
--hits                Write hit maps [default]
--no-hits             Do not write hit maps
--wcov                Write white noise covariance [default]
--no-wcov             Do not write white noise covariance
--wcov-inv            Write inverse white noise covariance [default]
--no-wcov-inv         Do not write inverse white noise covariance
--conserve-memory     Conserve memory when staging libMadam buffers
                      [default]
--no-conserve-memory  Do not conserve memory when staging libMadam buffers
--input-map INPUT_MAP
                      Input map for signal
--pysm-model PYSM_MODEL
                      Comma separated models for on-the-fly PySM simulation,
                      e.g. "s1,d6,f1,a2"
--pysm-apply-beam     Convolve sky with detector beam
--no-pysm-apply-beam  Do not convolve sky with detector beam.
--pysm-precomputed-cmb-K_CMB PYSM_PRECOMPUTED_CMB_K_CMB
                      Precomputed CMB map for PySM in K_CMBit overrides any
                      model defined in pysm_model"
--ground-map GROUND_MAP
                      Fixed ground template map
```

```
  --ground-nside GROUND_NSIDE
                        Ground template resolution
  --ground-fwhm-deg GROUND_FWHM_DEG
                        Ground template smoothing in degrees
  --ground-lmax GROUND_LMAX
                        Ground template expansion order
  --ground-scale GROUND_SCALE
                        Ground template RMS at el=45 deg
  --ground-power GROUND_POWER
                        Exponential for suppressing ground pick-up at higher
                        observing elevation
  --simulate-ground     Enable simulating ground pickup.
  --no-simulate-ground  Disable simulating ground pickup.
  --tidas TIDAS         Output TIDAS export path
  --spt3g SPT3G         Output SPT3G export path
  --MC-start MC_START   First Monte Carlo noise realization
  --MC-count MC_COUNT   Number of Monte Carlo noise realizations
  --outdir OUTDIR       Output directory
  --focalplane FOCALPLANE
                        Pickle file containing a dictionary of detector
                        properties. The keys of this dict are the detector
                        names, and each value is also a dictionary with keys
                        "quat" (4 element ndarray), "fwhm" (float, arcmin),
                        "fknee" (float, Hz), "alpha" (float), and "NET"
                        (float).
  --freq FREQ           Comma-separated list of frequencies with identical
                        focal planes. They override the bandpasses in the
                        focalplane for the purpose of scaling the atmospheric
                        signal but not for simulating the sky signal.
```

`main` without profiling

```
def main():
    mpiworld, procs, rank, comm = get_comm()

    args, comm = parse_arguments(comm)

    # Initialize madam parameters

    madampars = setup_madam(args)

    # Load and broadcast the schedule file

    schedules = load_schedule(args, comm)

    # Load the weather and append to schedules
```

```
load_weather(args, comm, schedules)

# load or simulate the focalplane

detweights = load_focalplanes(args, comm, schedules)

# Create the TOAST data object to match the schedule.  This will
# include simulating the boresight pointing

data, telescope_data = create_observations(args, comm, schedules)

# Split the communicator for day and season mapmaking

time_comms = get_time_communicators(args, comm, data)

# Expand boresight quaternions into detector pointing weights and
# pixel numbers

expand_pointing(args, comm, data)

# Purge the pointing if we are NOT going to export the
# data to a TIDAS volume
if (args.tidas is None) and (args.spt3g is None):
    for ob in data.obs:
        tod = ob["tod"]
        tod.free_radec_quats()

# Prepare auxiliary information for distributed map objects

_, localsm, subnpix = get_submaps(args, comm, data)

if args.pysm_model:
    focalplanes = [s.telescope.focalplane.detector_data for s in schedules]
    signalname = simulate_sky_signal(
        args, comm, data, focalplanes, subnpix, localsm, "signal"
    )
else:
    signalname = scan_sky_signal(args, comm, data, localsm, subnpix, "signal")

# Set up objects to take copies of the TOD at appropriate times

totalname, totalname_freq = setup_sigcopy(args)

# Loop over Monte Carlos

firstmc = args.MC_start
nsimu = args.MC_count
```

```
freqs = [float(freq) for freq in args.freq.split(",")]
nfreq = len(freqs)

for mc in range(firstmc, firstmc + nsimu):

    simulate_atmosphere(args, comm, data, mc, totalname)

    # Loop over frequencies with identical focal planes and identical
    # atmospheric noise

    for ifreq, freq in enumerate(freqs):
        # Make a copy of the atmosphere so we can scramble the gains and apply
        # frequency-dependent scaling

        copy_signal(args, comm, data, totalname, totalname_freq)

        scale_atmosphere_by_frequency(
            args, comm, data, freq=freq, mc=mc, cache_name=totalname_freq
        )

        update_atmospheric_noise_weights(args, comm, data, freq, mc)

        # Add previously simulated sky signal to the atmospheric noise

        add_signal(args, comm, data, totalname_freq, signalname, purge=(nsimu == 1))

        mcoffset = ifreq * 1000000

        simulate_noise(args, comm, data, mc + mcoffset, totalname_freq)

        simulate_sss(args, comm, data, mc + mcoffset, totalname_freq)

        scramble_gains(args, comm, data, mc + mcoffset, totalname_freq)

        if (mc == firstmc) and (ifreq == 0):
            # For the first realization and frequency, optionally
            # export the timestream data

            output_tidas(args, comm, data, totalname)
            output_spt3g(args, comm, data, totalname)

        outpath = setup_output(args, comm, mc + mcoffset, freq)

        # Bin and destripe maps

        apply_madam(
            args,
            comm,
```

```
        data,
        madampars,
        outpath,
        detweights,
        totalname_freq,
        freq=freq,
        time_comms=time_comms,
        telescope_data=telescope_data,
        first_call=(mc == firstmc),
    )

    if args.apply_polyfilter or args.apply_groundfilter:

        # Filter signal

        apply_polyfilter(args, comm, data, totalname_freq)

        apply_groundfilter(args, comm, data, totalname_freq)

        # Bin filtered maps

        apply_madam(
            args,
            comm,
            data,
            madampars,
            outpath,
            detweights,
            totalname_freq,
            freq=freq,
            time_comms=time_comms,
            telescope_data=telescope_data,
            first_call=False,
            extra_prefix="filtered",
            bin_only=True,
        )
```

Here is a full working example of the ground simulation pipeline
First we need an observing schedule. This one is for one patch and covers 24 hours:

```
In [8]: ! toast_ground_schedule.py \
            --site-lat "-22.958064" \
            --site-lon "-67.786222" \
            --site-alt 5200 \
            --site-name Atacama \
            --telescope LAT \
            --start "2020-01-01 00:00:00" \
            --stop "2020-01-02 00:00:00" \
```

```
            --patch-coord C \
            --patch small_patch,1,40,-40,10 \
            --el-min 45 \
            --el-max 60 \
            --out schedule.txt

        ! cat schedule.txt
```

```
TOAST INFO: Adding patch "small_patch"
TOAST INFO: Center-and-width format
TOAST INFO: Global timer: toast_ground_schedule:  0.15 seconds (1 calls)
#Site            Telescope        Latitude [deg] Longitude [deg]   Elevation [m]
 Atacama         LAT                     -22.958         -67.786          5200.0
#Start time UTC       Stop time UTC        Start MJD      Stop MJD       Patch name
 2020-01-01 02:00:00  2020-01-01 02:14:30    58849.083333   58849.093403 small_patch
 2020-01-01 02:14:40  2020-01-01 02:29:10    58849.093519   58849.103588 small_patch
 2020-01-01 02:29:20  2020-01-01 02:43:50    58849.103704   58849.113773 small_patch
 2020-01-01 02:44:00  2020-01-01 02:58:00    58849.113889   58849.123611 small_patch
 2020-01-01 02:59:40  2020-01-01 03:12:55    58849.124769   58849.133970 small_patch
 2020-01-01 03:13:05  2020-01-01 03:26:20    58849.134086   58849.143287 small_patch
 2020-01-01 03:26:30  2020-01-01 03:39:45    58849.143403   58849.152604 small_patch
 2020-01-01 03:39:55  2020-01-01 03:52:40    58849.152720   58849.161574 small_patch
 2020-01-01 20:44:20  2020-01-01 20:57:35    58849.864120   58849.873322 small_patch
 2020-01-01 20:57:45  2020-01-01 21:11:00    58849.873438   58849.882639 small_patch
 2020-01-01 21:11:10  2020-01-01 21:24:25    58849.882755   58849.891956 small_patch
 2020-01-01 21:24:35  2020-01-01 21:37:20    58849.892072   58849.900926 small_patch
 2020-01-01 21:39:00  2020-01-01 21:52:45    58849.902083   58849.911632 small_patch
 2020-01-01 21:52:55  2020-01-01 22:06:40    58849.911748   58849.921296 small_patch
 2020-01-01 22:06:50  2020-01-01 22:20:35    58849.921412   58849.930961 small_patch
 2020-01-01 22:20:45  2020-01-01 22:34:00    58849.931076   58849.940278 small_patch
```

Then we need a focalplane

```
In [9]: ! toast_fake_focalplane.py \
            --minpix 100 \
            --out focalplane \
            --fwhm 30 \
            --fwhm_sigma 0.05 \
            --fov 10 \
            --psd_fknee 5e-2 \
            --psd_NET 1e-3 \
            --psd_alpha 1 \
            --psd_fmin 1e-5 \
            --bandcenter_ghz 100 \
            --bandcenter_sigma 0.01 \
            --bandwidth_ghz 10 \
            --bandwidth_sigma 0.1
```

```
TOAST INFO: using 127 pixels (254 detectors)
```

And of course we need the weather file

```
In [10]: ! [[ ! -e weather_Atacama.fits ]] && wget http://portal.nersc.gov/project/cmb/toast_da
```

Now write a parameter file that uses the above inputs and simulates sky signal (using PySM), atmosphere and instrument noise

```
In [11]: %%writefile toast_ground_sim.par
         --weather
         weather_Atacama.fits
         --scan-rate
         1
         --scan-accel
         3
         --schedule
         schedule.txt
         --sample-rate
         30
         --coord
         C
         --hwp-rpm
         2
         --nside
         512
         --common-flag-mask
         1
         --polyfilter
         --poly-order
         5
         --groundfilter
         --ground-order
         3
         --atmosphere
         --atm-lmin-center
         0.1
         --atm-lmax-center
         30
         --atm-gain
         3e-5
         --atm-zmax
         1000
         --atm-xstep
         30
         --atm-ystep
         30
         --atm-zstep
```

```
        30
        --atm-nelem-sim-max
        10000
        --atm-wind-dist
        5000
        --atm-cache
        atm_cache
        --noise
        --gainscrambler
        --gain-sigma
        0.05
        --madam-prefix
        groundsim000
        --madam-iter-max
        200
        --madam-precond-width
        100
        --madam-baseline-length
        1
        --madam-noisefilter
        --madam-allreduce
        --destripe
        --binmap
        --hits
        --wcov
        --ground-nside
        512
        --ground-fwhm-deg
        3
        --ground-lmax
        512
        --ground-scale
        1e-3
        --simulate-ground
        --focalplane
        focalplane_127.pkl
        --freq
        100
        --pysm-model
        s1,d1,f1,a1
        --pysm-apply-beam
Overwriting toast_ground_sim.par
```

With the PySM part (last three lines) the simulation will take about 20 minutes on a single Cori Haswell node. If you remove them, it will run in about 4 minutes.

Finally, instead of submitting the job interactively with srun, we write an actual script and submit it.

Future Directions & Roadmap

TOAST Workshop - 2019 UCSD

New Features

- Documentation!
- TOAST can call external map-making code (e.g. libmadam), but there are also built-in tools that can do some map-making operations. Performance improvements coming.
- Timestream processing- there may be general things usable from
- Supporting new architectures: next NERSC computer in ~1 - 1.5 years will have AMD Epyc processors with 64 or 128 "traditional" cores and NVIDIA GPUs on half the nodes.
- Better workflow interfacing with the spt3g package (both the data format and the HTC pipeline tools)

API Improvements

- Now that code has been applied in a variety of use cases, we can see some issues with interfaces.
- Any API changes need a well documented and discussed implementation and transition plan. Too much code depends on the current interfaces for a "quick" change.
- However, we should still aim to improve interfaces and make things easier:
    - Observation / TOD: Perhaps moving to a pure source and sink model, where the "observation" is just a data container (needs more thought).
    - Make it easier to instantiate operators from config files.