Introduction to Monte Python

Benjamin Audren

Institute of Theoretical Physics
École Polytechnique Fédérale de Lausanne

13/05/2014

Outline

Outline

Goals of Monte Python

Wish list

- **Interfacing** with CLASS (C), other likelihoods

Goals of Monte Python

Wish list

- **Interfacing** with CLASS (C), other likelihoods
- **Modular design** (use other Boltzmann codes, algorithms)

Goals of Monte Python

Wish list

- **Interfacing** with CLASS (C), other likelihoods
- **Modular design** (use other Boltzmann codes, algorithms)
- **Readability** (a code is read more than it is written)

Goals of Monte Python

Wish list

- **Interfacing** with CLASS (C), other likelihoods
- **Modular design** (use other Boltzmann codes, algorithms)
- **Readability** (a code is read more than it is written)
- **Manipulating files** - read, write, renaming

Goals of Monte Python

Wish list

- **Interfacing** with CLASS (C), other likelihoods
- **Modular design** (use other Boltzmann codes, algorithms)
- **Readability** (a code is read more than it is written)
- **Manipulating files** - read, write, renaming
- Using **Existing Library** (for complex computations)

Goals of Monte Python

Wish list

- **Interfacing** with CLASS (C), other likelihoods
- **Modular design** (use other Boltzmann codes, algorithms)
- **Readability** (a code is read more than it is written)
- **Manipulating files** - read, write, renaming
- Using **Existing Library** (for complex computations)
- **Easy to learn, easy to use**

Goals

Goals

~~Wish~~ Tick list

- ✓ **Interfacing** with CLASS (C), other likelihoods
- ✓ **Modular design** (use other Boltzmann codes, algorithms)
- ✓ **Readability** (a code is read more than it is written)
- ✓ **Manipulating files** - read, write, renaming
- ✓ Using **Existing Library** (for complex computations)
- ✓ **Easy to learn, easy to use**

Design policy of Monte Python

Guidelines

- **Modular structure**: separate I/O, parser, data structures, sampler, interface with the cosmological module, plotting: **Tuesday**

Design policy of Monte Python

Guidelines

- **Modular structure**: separate I/O, parser, data structures, sampler, interface with the cosmological module, plotting: **Tuesday**
- **Remembering a run**: you must be able to reproduce a run done some time ago (version control, folder based)

Design policy of Monte Python

Guidelines

- **Modular structure**: separate I/O, parser, data structures, sampler, interface with the cosmological module, plotting: **Tuesday**
- **Remembering a run**: you must be able to reproduce a run done some time ago (version control, folder based)
- **Intelligent communication with** CLASS: if CLASS knows how to do it, you can ask for it.

Design policy of Monte Python

Guidelines

- **Modular structure**: separate I/O, parser, data structures, sampler, interface with the cosmological module, plotting: **Tuesday**
- **Remembering a run**: you must be able to reproduce a run done some time ago (version control, folder based)
- **Intelligent communication with CLASS**: if CLASS knows how to do it, you can ask for it.
- **Convenient Plotting**: since a folder will be self contained, with all the information, producing a plot out of this folder should be easy.

Design policy of Monte Python

Guidelines

- **Modular structure**: separate I/O, parser, data structures, sampler, interface with the cosmological module, plotting: **Tuesday**
- **Remembering a run**: you must be able to reproduce a run done some time ago (version control, folder based)
- **Intelligent communication with CLASS**: if CLASS knows how to do it, you can ask for it.
- **Convenient Plotting**: since a folder will be self contained, with all the information, producing a plot out of this folder should be easy.
- **Using mock data**: there must be a way to handle mock data easily.

Conclusion on design

You tell me !

We'll see in the coming days if it works !

Outline

Installation

Compile the wrapper

```
cd class; make;
cd python; python setup.py install --user
```

Download

http://montepython.net or
https://github.com/audren/montepython_public/releases

Unzip

```
bunzip2 montepython_v2.0.2.tar.bz2
tar -xvf montepython_v2.0.2.tar
```

Configure

```
cp default.conf.template default.conf
```

edit it to match your path

Parser

Common arguments

`python montepython/MontePython.py run` plus...

- *A minima*: `-o chains/planck -p example.param`

Parser

Common arguments

`python montepython/MontePython.py run` plus...

- *A minima*: `-o chains/planck -p example.param`
- `-N 1000` **Number of proposed steps**

Parser

Common arguments

`python montepython/MontePython.py run` plus...

- *A minima*: `-o chains/planck -p example.param`
- `-N 1000`
- `-c covmat/old.covmat` **better proposal density**

Parser

Common arguments

`python montepython/MontePython.py run` plus...

- *A minima*: `-o chains/planck -p example.param`
- `-N 1000`
- `-c covmat/old.covmat`
- `-j fast` **for a fast Cholesky decomposition sampling**

Parser

Common arguments

`python montepython/MontePython.py run` plus...

- *A minima*: `-o chains/planck -p example.param`
- `-N 1000`
- `-c covmat/old.covmat`
- `-j fast` for a fast Cholesky decomposition sampling
- `-m NS` **to use MultiNest and the Nested Sampling algorithm**

Input Parameter File

```
data.experiments=['fake_planck_bluebook']

# Cosmological parameters list
data.parameters['omega_b']   = [2.249,  None, None, 0.016, 0.01,'cosmo']
data.parameters['omega_cdm'] = [0.1120, None, None, 0.0016,1,   'cosmo']
data.parameters['n_s']       = [0.963,  None, None, 0.004, 1,   'cosmo']
data.parameters['A_s']       = [2.42,   None, None, 0.038, 1e-9,'cosmo']
data.parameters['h']         = [0.703,  None, None, 0.0065,1,   'cosmo']
data.parameters['tau_reio']  = [0.085,  None, None, 0.0044,1,   'cosmo']

# Derived parameter list
data.parameters['z_reio']        = [0,   -1, -1, 0,1,  'derived']
data.parameters['Omega_Lambda'] = [0,   -1, -1, 0,1,  'derived']
```

```
data.cosmo_arguments['N_eff'] = 3.046
```

```
data.N=10
```

```
data.write_step=5
```

Input Parameter File

```
data.experiments=['fake_planck_bluebook']

# Cosmological parameters list
data.parameters['omega_b']    = [2.249,  None, None, 0.016, 0.01,'cosmo']
data.parameters['omega_cdm']  = [0.1120, None, None, 0.0016,1,   'cosmo']
data.parameters['n_s']        = [0.963,  None, None, 0.004, 1,   'cosmo']
data.parameters['A_s']        = [2.42,   None, None, 0.038, 1e-9,'cosmo']
data.parameters['h']          = [0.703,  None, None, 0.0065,1,   'cosmo']
data.parameters['tau_reio']   = [0.085,  None, None, 0.0044,1,   'cosmo']

# Derived parameter list
data.parameters['z_reio']        = [0,   -1, -1, 0,1,  'derived']
data.parameters['Omega_Lambda'] = [0,   -1, -1, 0,1,  'derived']
```

Mean values

```
data.cosmo_arguments['N_eff'] = 3.046
```

```
data.N=10
```

```
data.write_step=5
```

Input Parameter File

```
data.experiments=['fake_planck_bluebook']

# Cosmological parameters list
data.parameters['omega_b']   = [2.249,  None, None, 0.016, 0.01,'cosmo']
data.parameters['omega_cdm'] = [0.1120, None, None, 0.0016,1,   'cosmo']
data.parameters['n_s']       = [0.963,  None, None, 0.004, 1,   'cosmo']
data.parameters['A_s']       = [2.42,   None, None, 0.038, 1e-9,'cosmo']
data.parameters['h']         = [0.703,  None, None, 0.0065,1,   'cosmo']
data.parameters['tau_reio']  = [0.085,  None, None, 0.0044,1,   'cosmo']

# Derived parameter list
data.parameters['z_reio']       = [0,   -1,  -1, 0,1,  'derived']
data.parameters['Omega_Lambda'] = [0,   -1, -1, 0,1,  'derived']
```

Lower-Upper bounds

```
data.cosmo_arguments['N_eff'] = 3.046

data.N=10

data.write_step=5
```

Input Parameter File

```
data.experiments=['fake_planck_bluebook']

# Cosmological parameters list
data.parameters['omega_b']    = [2.249,  None, None, 0.016, 0.01,'cosmo']
data.parameters['omega_cdm']  = [0.1120, None, None, 0.0016,1,   'cosmo']
data.parameters['n_s']        = [0.963,  None, None, 0.004, 1,   'cosmo']
data.parameters['A_s']        = [2.42,   None, None, 0.038, 1e-9,'cosmo']
data.parameters['h']          = [0.703,  None, None, 0.0065,1,   'cosmo']
data.parameters['tau_reio']   = [0.085,  None, None, 0.0044,1,   'cosmo']

# Derived parameter list
data.parameters['z_reio']        = [0,   -1, -1, 0,1,  'derived']
data.parameters['Omega_Lambda'] = [0,   -1, -1, 0,1,  'derived']

data.cosmo_arguments['N_eff'] = 3.046

data.N=10

data.write_step=5
```

1σ proposal

Input Parameter File

```
data.experiments=['fake_planck_bluebook']

# Cosmological parameters list
data.parameters['omega_b']   = [2.249,  None, None, 0.016, 0.01,'cosmo']
data.parameters['omega_cdm'] = [0.1120, None, None, 0.0016,1,   'cosmo']
data.parameters['n_s']       = [0.963,  None, None, 0.004, 1,   'cosmo']
data.parameters['A_s']       = [2.42,   None, None, 0.038, 1e-9,'cosmo']
data.parameters['h']         = [0.703,  None, None, 0.0065,1,   'cosmo']
data.parameters['tau_reio']  = [0.085,  None, None, 0.0044,1,   'cosmo']

# Derived parameter list
data.parameters['z_reio']        = [0,   -1, -1, 0,1,  'derived']
data.parameters['Omega_Lambda'] = [0,   -1, -1, 0,1,  'derived']
```

scale

```
data.cosmo_arguments['N_eff'] = 3.046
```

```
data.N=10
```

```
data.write_step=5
```

Input Parameter File

```
data.experiments=['fake_planck_bluebook']

# Cosmological parameters list
data.parameters['omega_b']   = [2.249,  None, None, 0.016, 0.01,'cosmo']
data.parameters['omega_cdm'] = [0.1120, None, None, 0.0016,1,   'cosmo']
data.parameters['n_s']       = [0.963,  None, None, 0.004, 1,   'cosmo']
data.parameters['A_s']       = [2.42,   None, None, 0.038, 1e-9,'cosmo']
data.parameters['h']         = [0.703,  None, None, 0.0065,1,   'cosmo']
data.parameters['tau_reio']  = [0.085,  None, None, 0.0044,1,   'cosmo']

# Derived parameter list
data.parameters['z_reio']        = [0,   -1, -1, 0,1,  'derived']
data.parameters['Omega_Lambda'] = [0,   -1, -1, 0,1,  'derived']
```

type

```
data.cosmo_arguments['N_eff'] = 3.046
```

```
data.N=10
```

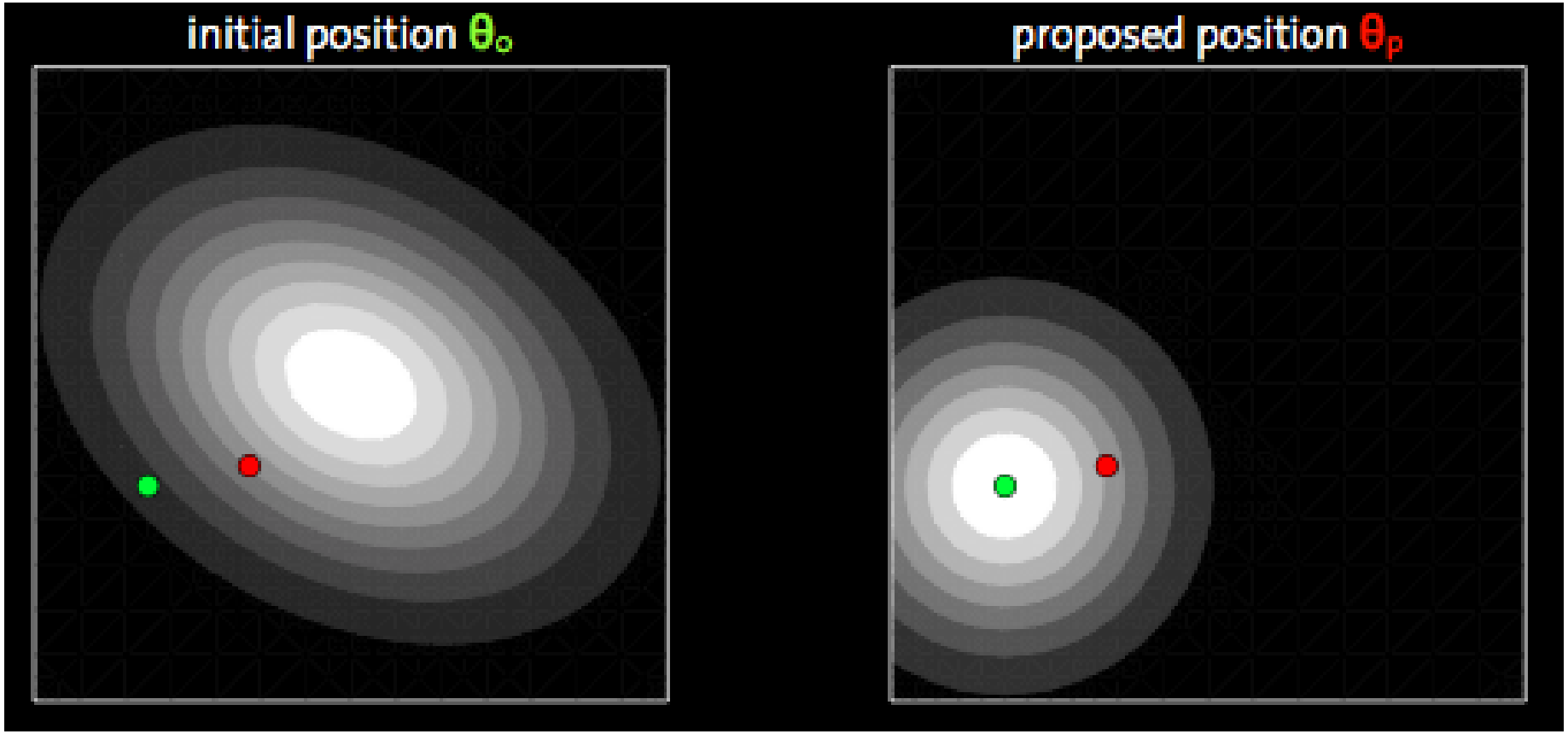
```
data.write_step=5
```

Input Parameter File

```
data.experiments=['fake_planck_bluebook']

# Cosmological parameters list
data.parameters['omega_b']   = [2.249,  None, None, 0.016, 0.01,'cosmo']
data.parameters['omega_cdm'] = [0.1120, None, None, 0.0016,1,   'cosmo']
data.parameters['n_s']       = [0.963,  None, None, 0.004, 1,   'cosmo']
data.parameters['A_s']       = [2.42,   None, None, 0.038, 1e-9,'cosmo']
data.parameters['h']         = [0.703,  None, None, 0.0065,1,   'cosmo']
data.parameters['tau_reio']  = [0.085,  None, None, 0.0044,1,   'cosmo']

# Derived parameter list
data.parameters['z_reio']        = [0,   -1, -1, 0,1,  'derived']
data.parameters['Omega_Lambda'] = [0,   -1, -1, 0,1,  'derived']

data.cosmo_arguments['N_eff'] = 3.046

data.N=10

data.write_step=5
```

Number of steps, write buffer

Reminder from lecture 1



Types of parameter

cosmological

- **known to Class (can define tricks)**
- **must be compatible (as in `explanatory.ini`)**

Types of parameter

cosmological

- known to Class (can define tricks)
- must be compatible (as in `explanatory.ini`)

derived

- **known to Class**
- **can be redundant**

Types of parameter

cosmological

- known to Class (can define tricks)
- must be compatible (as in `explanatory.ini`)

derived

- known to Class
- can be redundant

nuisance

- **As the name indicates...**
-

Types of parameter

cosmological

- known to Class (can define tricks)
- must be compatible (as in `explanatory.ini`)

derived

- known to Class
- can be redundant

nuisance

- As the name indicates. . .
- **Needed pain**

Warning!

Input parameter file and log.param

- When a folder is **created**, a file `log.param` is written, copying information from param and likelihoods.
- When a new chain is launched, **the input file is not read any more**, only the `log.param`

Configuration File

```
root = '/Users/benjaminaudren/Desktop/professional/codes'

path['cosmo']    = root+'/class/'
```

Usage: summary

After installation

- `cp default.conf.template default.conf` and edit
- `python montepython/MontePython.py -o chains/test -p example.param`

Running strategies

Starting

- Choosing **experiments** to combine - **careful with that**
- Varying **parameters**, proposal distribution
- After M ($\simeq 10$) chains of N ($\simeq 10000$) points, **analyze**

Analyzing

```
python montepython/MontePython.py info folder
```

Main

- Feed the new found **covariance matrix** to new runs.
- Analyze **only these new chains**
- Compare best-fit likelihood, or evidence, read parameter constraints

What Monte Python does for you

Convenience

- No need to choose a name for your chain

What Monte Python does for you

Convenience

- No need to choose a name for your chain
- Can use a covariance matrix with partial information

What Monte Python does for you

Convenience

- No need to choose a name for your chain
- Can use a covariance matrix with partial information
- Can analyze only certain chains