Nils Schöneberg Monday 05.06.2023 **nils.science@gmail.com**

European Research Council

Established by the European Commission

Institut de Ciències del Cosmos

UNIVERSITAT DE BARCELONA

EXCELENCIA MARÍA DE MAEZTU 2020–2023

The CLASS code

Layout of the Tutorial

- Lecture separated into three parts:
  - What is CLASS and how do I get it?

Layout of the Tutorial

- Lecture separated into three parts:
  - What is CLASS and how do I get it?
  - Inputs and outputs – How to work with CLASS?

Layout of the Tutorial

- Lecture separated into three parts:
  - What is CLASS and how do I get it?
  - Inputs and outputs – How to work with CLASS?
  - The layout of CLASS – internal and external

Layout of the Tutorial

- Lecture separated into three parts:
  - What is CLASS and how do I get it?
  - Inputs and outputs – How to work with CLASS?
  - The layout of CLASS – internal and external

- What we probably won't cover:
  - Nitty-gritty details of the C code → ask me!

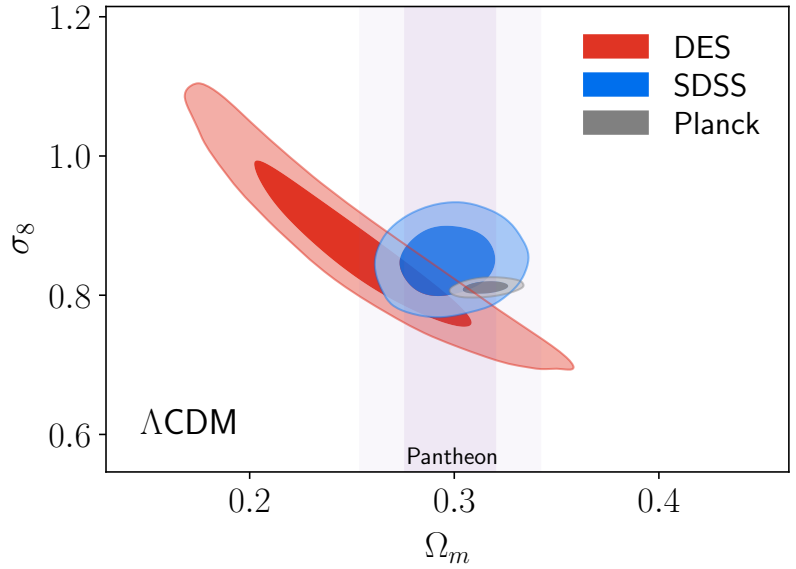
Part One

What is class ?

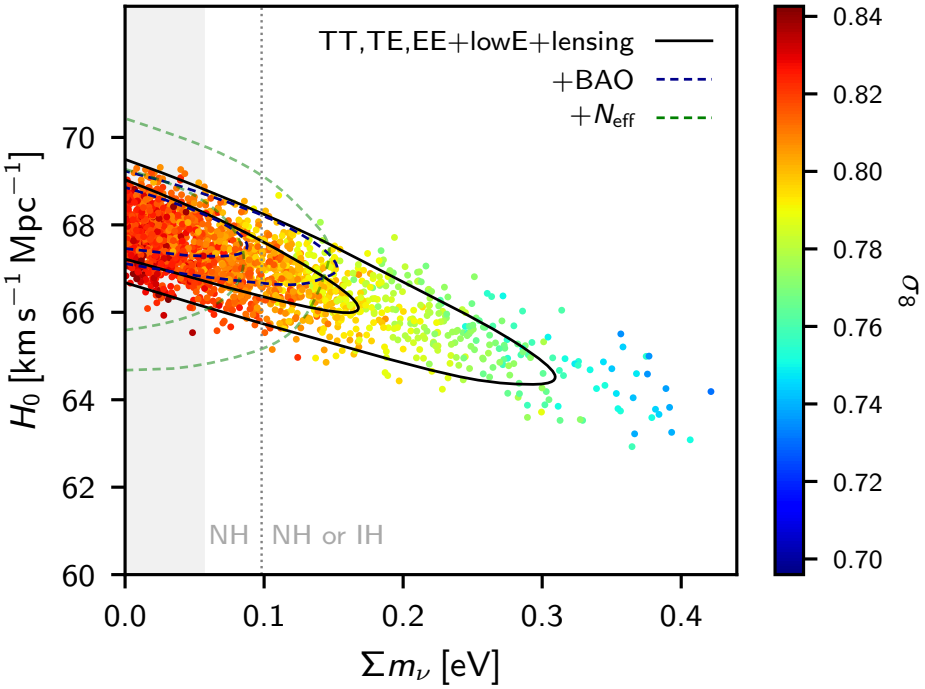
How do I get it?

Nils Schöneberg nils.science@gmail.com

Einstein-Boltzmann solver (EBS)



Asgari+2020 | Astron.Astrophys. 645 (2021) A104, arXiv:2007.15633

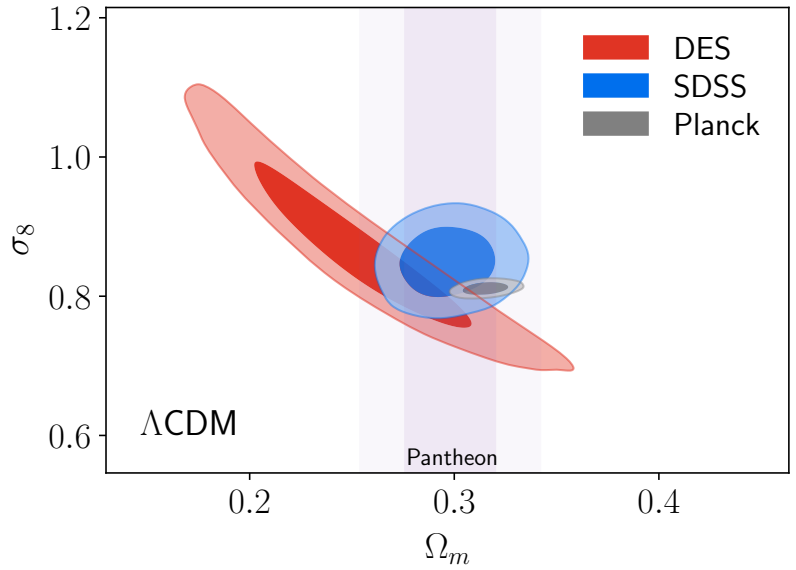


Planck Collaboration+2018 | Astron.Astrophys. 641 (2020) A6, arXiv:1807.06209

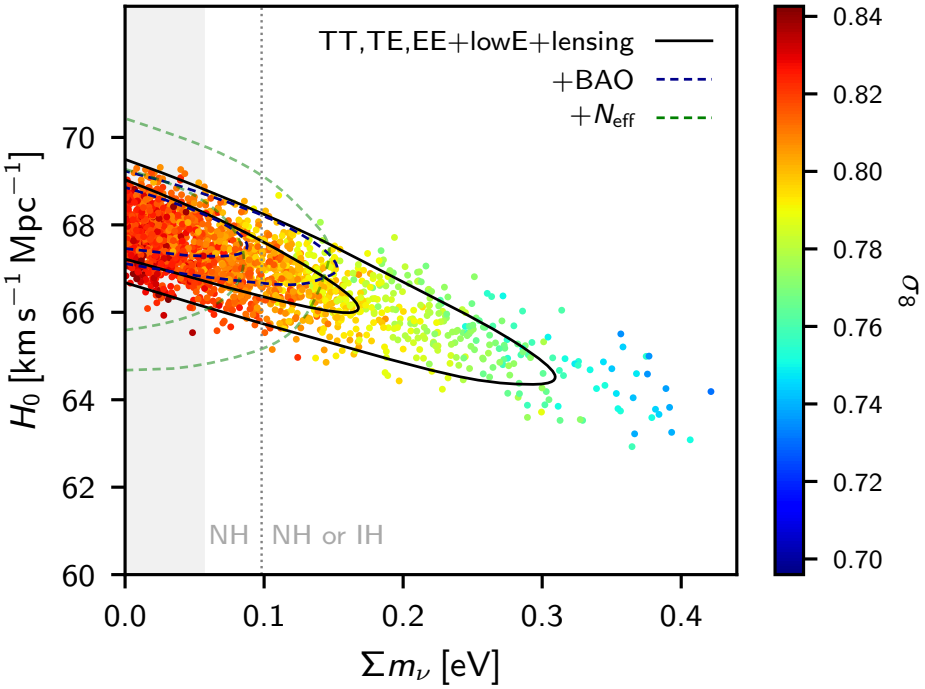
Nils Schöneberg nils.science@gmail.com

Einstein-Boltzmann solver (EBS)

- **linear perturbations** = basis of modern cosmology



Asgari+2020 | Astron.Astrophys. 645 (2021) A104, arXiv:2007.15633



Planck Collaboration+2018 | Astron.Astrophys. 641 (2020) A6, arXiv:1807.06209

Einstein-Boltzmann solver (EBS)

- **linear perturbations** = basis of modern cosmology
  - Direct observables

Einstein-Boltzmann solver (EBS)

- **linear perturbations** = basis of modern cosmology
  - Direct observables
    - CMB : C_L^{TT}, C_L^{TE}, C_L^{EE}, $C_L^{\phi\phi}$

Nils Schöneberg nils.science@gmail.com

Einstein-Boltzmann solver (EBS)

- **linear perturbations** = basis of modern cosmology
  - Direct observables
    - CMB : C_L^{TT}, C_L^{TE}, C_L^{EE}, $C_L^{\phi\phi}$
    - Cosmic shear : $\xi^{\pm}(r,\vartheta)$

Nils Schöneberg nils.science@gmail.com

Einstein-Boltzmann solver (EBS)

- **linear perturbations** = basis of modern cosmology
  - Direct observables
    - CMB : C_L^{TT} , C_L^{TE} , C_L^{EE} , $C_L^{\phi\phi}$
    - Cosmic shear : $\xi^{\pm}(r,\vartheta)$
    - Linear galaxy counts : C_L^{dd} , $P_{lin}(k,z)$

Nils Schöneberg nils.science@gmail.com

Einstein-Boltzmann solver (EBS)

- **linear perturbations** = basis of modern cosmology
  - Direct observables
    - CMB : C_L^{TT}, C_L^{TE}, C_L^{EE}, $C_L^{\phi\phi}$
    - Cosmic shear : $\xi^{\pm}(r,\vartheta)$
    - Linear galaxy counts : C_L^{dd}, $P_{lin}(k,z)$
  - Input for other codes:
    - Initial conditions for simulations (N-body, 21cm, Lyman-α forest, ...)

Einstein-Boltzmann solver (EBS)

- Fundamental objective:
  - Predict linear perturbations and corresponding summary statistics

Einstein-Boltzmann solver (EBS)

- Fundamental objective:
  - Predict linear perturbations and corresponding summary statistics
  - linear perturbations follow the Einstein-Boltzmann equations

Einstein-Boltzmann solver (EBS)

- Fundamental objective:
  - Predict linear perturbations and corresponding summary statistics
  - linear perturbations follow the Einstein-Boltzmann equations

$$G_{\mu\nu} = 8\pi G \cdot T_{\mu\nu} \qquad \frac{df}{dt} = C[f]$$

Einstein **Boltzmann**

Nils Schöneberg nils.science@gmail.com

Einstein-Boltzmann solver (EBS)

- Fundamental objective:
  - Predict linear perturbations and corresponding summary statistics
  - linear perturbations follow the Einstein-Boltzmann equations

$$G_{\mu\nu} = 8\pi G \cdot T_{\mu\nu} \qquad \frac{df}{dt} = C[f]$$

Einstein **Boltzmann**

$$k^2(\phi + \mathcal{H}\psi) = 4\pi G a^2(\rho + P)\theta \qquad \delta_b' + \theta_b = 3\phi'$$

History of EBS

- Initially request from Planck science team to check for bias in parameter extraction

History of EBS

- Initially request from Planck science team to check for bias in parameter extraction

  $\rightarrow$ Strong CLASS-CAMB competition leading to 0.1% agreement

History of EBS

1995 – COSMICS (Fortran77, Bertschinger)

History of EBS

1995 – COSMICS (Fortran77, Bertschinger)

1996 – CMBFAST (Fortran 77, Seljak & Zaldarriaga)

History of EBS

1995 – COSMICS (Fortran77, Bertschinger)

1996 – CMBFAST (Fortran 77, Seljak & Zaldarriaga)

- 1999 – CAMB (Fortran 90/2000, Lewis & Challinor)

History of EBS

1995 – COSMICS (Fortran77, Bertschinger)

1996 – CMBFAST (Fortran 77, Seljak & Zaldarriaga)

- 1999 – CAMB (Fortran 90/2000, Lewis & Challinor)

2003 – CMBEASY (C++, Doran, discontinued)

History of EBS

1995 – COSMICS (Fortran77, Bertschinger)

1996 – CMBFAST (Fortran 77, Seljak & Zaldarriaga)

- 1999 – CAMB (Fortran 90/2000, Lewis & Challinor)

2003 – CMBEASY (C++, Doran, discontinued)

- 2011 – CLASS (C, Lesgourgues & Tram & growing team)

History of EBS

1995 – COSMICS (Fortran77, Bertschinger)

1996 – CMBFAST (Fortran 77, Seljak & Zaldarriaga)

- 1999 – CAMB (Fortran 90/2000, Lewis & Challinor)

2003 – CMBEASY (C++, Doran, discontinued)

- 2011 – CLASS (C, Lesgourgues & Tram & growing team)

These are all current "competitors" (focus shifted to emulation)

History of EBS

These are all current "competitors" (focus shifted to emulation)

2002 DASh $\rightarrow$ C_L^{TT} (Kalpinghat & Knox & Skordis)

Nils Schöneberg nils.science@gmail.com

History of EBS

These are all current "competitors" (focus shifted to emulation)

2002 DASh $\rightarrow$ C_L^{TT} (Kalpinghat & Knox & Skordis)

2004 CMBwarp $\rightarrow$ C_L^{TT}, C_L^{TE}, C_L^{EE} (Jimenez & Verde & Peiris & Kosowsky)

History of EBS

These are all current "competitors" (focus shifted to emulation)

2002 DASh $\rightarrow {C_L}^{TT}$ (Kalpinghat & Knox & Skordis)

2004 CMBwarp $\rightarrow {C_L}^{TT}$, ${C_L}^{TE}$, ${C_L}^{EE}$ (Jimenez & Verde & Peiris & Kosowsky)

2006 Pico $\rightarrow {C_L}^{TT}$, ${C_L}^{TE}$, ${C_L}^{EE}$ (Fendt & Wandelt)

History of EBS

These are all current "competitors" (focus shifted to emulation)

2002 DASh $\rightarrow C_L^{TT}$ (Kalpinghat & Knox & Skordis)

2004 CMBwarp $\rightarrow C_L^{TT}$, C_L^{TE}, C_L^{EE} (Jimenez & Verde & Peiris & Kosowsky)

2006 Pico $\rightarrow C_L^{TT}$, C_L^{TE}, C_L^{EE} (Fendt & Wandelt)

- **2019 CosmicNet** $\rightarrow$ perturbations (Albers & Fidler & Lesgourgues & Schöneberg & Torrado)

History of EBS

These are all current "competitors" (focus shifted to emulation)

2002 DASh → C_L^{TT} (Kalpinghat & Knox & Skordis)

2004 CMBwarp → C_L^{TT}, C_L^{TE}, C_L^{EE} (Jimenez & Verde & Peiris & Kosowsky)

2006 Pico → C_L^{TT}, C_L^{TE}, C_L^{EE} (Fendt & Wandelt)

- **2019 CosmicNet** → perturbations (Albers & Fidler & Lesgourgues & Schöneberg & Torrado)
- 2021 Bacco → P(k,z) (Arico & Angulo & Zennaro)

History of EBS

These are all current "competitors" (focus shifted to emulation)

2002 DASh → C_L^{TT} (Kalpinghat & Knox & Skordis)

2004 CMBwarp → C_L^{TT}, C_L^{TE}, C_L^{EE} (Jimenez & Verde & Peiris & Kosowsky)

2006 Pico → C_L^{TT}, C_L^{TE}, C_L^{EE} (Fendt & Wandelt)

- **2019 CosmicNet** → perturbations (Albers & Fidler & Lesgourgues & Schöneberg & Torrado)
- 2021 Bacco → P(k,z) (Arico & Angulo & Zennaro)
- **2021 CosmoPower** → observables (Mancini & Piras & Alsing & Joachimi & Hobson)

History of EBS

These are all current "competitors" (focus shifted to emulation)

2002 DASh → C_L^{TT} (Kalpinghat & Knox & Skordis)

2004 CMBwarp → C_L^{TT}, C_L^{TE}, C_L^{EE} (Jimenez & Verde & Peiris & Kosowsky)

2006 Pico → C_L^{TT}, C_L^{TE}, C_L^{EE} (Fendt & Wandelt)

- **2019 CosmicNet** → perturbations (Albers & Fidler & Lesgourgues & Schöneberg & Torrado)
- 2021 Bacco → P(k,z) (Arico & Angulo & Zennaro)
- **2021 CosmoPower** → observables (Mancini & Piras & Alsing & Joachimi & Hobson)
- 2022 Connect → C_L^{TT}, C_L^{TE}, C_L^{EE} (Nygaard & Holm & Hannestad & Tram)

History of EBS

These are all current "competitors" (focus shifted to emulation)

2002 DASh → C_L^{TT} (Kalpinghat & Knox & Skordis)

2004 CMBwarp → C_L^{TT}, C_L^{TE}, C_L^{EE} (Jimenez & Verde & Peiris & Kosowsky)

2006 Pico → C_L^{TT}, C_L^{TE}, C_L^{EE} (Fendt & Wandelt)

- **2019 CosmicNet** → perturbations (Albers & Fidler & Lesgourgues & Schöneberg & Torrado)
- 2021 Bacco → P(k,z) (Arico & Angulo & Zennaro)
- **2021 CosmoPower** → observables (Mancini & Piras & Alsing & Joachimi & Hobson)
- 2022 Connect → C_L^{TT}, C_L^{TE}, C_L^{EE} (Nygaard & Holm & Hannestad & Tram)
- 2023 ....

Why class?

CLASS is

Why class?

class is

- modular

Why class?

class is

- modular
- structured & documented

Why class?

class is

- modular
- structured & documented
- accurate and fast

Why class?

CLASS is

- modular
- structured & documented
- accurate and fast
- explicit (no hard-coding/global variables)

Why class?

CLASS is

- modular
- structured & documented
- accurate and fast
- explicit (no hard-coding/global variables)

There is also a helpful community:

- Development continuous
- **Suggestions are welcome!**

Documentation example

```
/** - --> (a) fix current approximation scheme. */

ptw->ptdw->ap_current = index_interval;

/** - --> (b) define the vector of quantities to be integrated
    over. If the current interval starts from the
    initial time zinitial, fill the vector with initial
    conditions. If it starts from an approximation
    switching point, redistribute correctly the values
    from the previous to the new vector. For both
    RECFAST and HYREC, the vector consists of Tmat, x_H,
    x_He, + others for exotic models */

class_call(thermodynamics_vector_init(ppr,
                                      pba,
                                      pth,
                                      interval_limit[index_interval],
                                      ptw),
           pth->error_message,
           pth->error_message);

/* find the value of last_index_back at z = - interval_limit[index_interval], in order to speed up
   subsequent interpolations in thermodynamics_derivs */
class_call(background_at_z(pba,
...
```

Installation

- Hopefully works:

```
pip install classy
```

Installation

- Hopefully works:

```
pip install classy
```

- Most likely works:

```
git clone git@github.com:lesgourg/class_public.git
cd class
make clean; make -j
```

Installation

- Hopefully works:

```
pip install classy
```

- Most likely works:

```
git clone git@github.com:lesgourg/class_public.git
cd class
make clean; make -j
```

Check if it works:
`python -c "import classy"`

Installation

- Hopefully works:

```
pip install classy
```

- Most likely works:

```
git clone git@github.com:lesgourg/class_public.git
cd class
make clean; make -j
```

Check if it works:
`python -c "import classy"`

Special care for MAC/Windows due to openMP

Installation

- Hopefully works:

```
pip install classy
```

- Most likely works:

```
git clone git@github.com:lesgourg/class_public.git
cd class
make clean; make -j
```

Check if it works:
`python -c "import classy"`

Special care for MAC/Windows due to openMP

Nils Schöneberg

nils.science@gmail.com

Installation

- Hopefully works:

```
pip install classy
```

- Most likely works:

```
git clone git@github.com:lesgourg/class_public.git
cd class
make clean; make -j
```

Check if it works:
`python -c "import classy"`

Special care for MAC/Windows due to openMP

https://github.com/lesgourg/class_public/wiki/Installation

Notebook (Installation)

Usage Overview

- CLASS is written in C

Usage Overview

- CLASS is written in C
- CLASSY is the python wrapper for CLASS

Usage Overview

- CLASS is written in C
- CLASSY is the python wrapper for CLASS
- They share (almost) all input/output functionalities

  → Only differ slightly in format

Usage Overview

- CLASS is written in C
- CLASSY is the python wrapper for CLASS
- They share (almost) all input/output functionalities

  → Only differ slightly in format

  (input python dictionary vs `.ini` file)

  (output python functions vs `.dat` files)

Usage (Command Line)

- Simple and easy file

  → `./class default.ini`

Usage (Command Line)

- Simple and easy file

  → `./class default.ini`

- Slim file matching Planck 2018 "baseline model" bestfit

  → `./class base_2018_plikHM_TTTEEE_lowl_lowE_lensing.ini`

Usage (Command Line)

- Simple and easy file

  → `./class default.ini`

- Slim file matching Planck 2018 "baseline model" bestfit

  → `./class base_2018_plikHM_TTTEEE_lowl_lowE_lensing.ini`

- All input is presented in detail in `explanatory.ini` (apart from precision parameters)

  → `./class explanatory.ini`

Usage (Command Line)

- Simple and easy file

  → `./class default.ini`

- Slim file matching Planck 2018 "baseline model" bestfit

  → `./class base_2018_plikHM_TTTEEE_lowl_lowE_lensing.ini`

- All input is presented in detail in `explanatory.ini` (apart from precision parameters)

  → `./class explanatory.ini`

- This is a *reference* file; we advise you to not modify it:
  - either start from a slim file (like `default.ini`),

Usage (Command Line)

- Simple and easy file

  → `./class default.ini`

- Slim file matching Planck 2018 "baseline model" bestfit

  → `./class base_2018_plikHM_TTTEEE_lowl_lowE_lensing.ini`

- All input is presented in detail in `explanatory.ini` (apart from precision parameters)

  → `./class explanatory.ini`

- This is a *reference* file; we advise you to not modify it:
  - either start from a slim file (like `default.ini`),
  - or copy it and reduce it to a shorter and more friendly file,

Usage (Command Line)

- Simple and easy file

  → `./class default.ini`

- Slim file matching Planck 2018 "baseline model" bestfit

  → `./class base_2018_plikHM_TTTEEE_lowl_lowE_lensing.ini`

- All input is presented in detail in `explanatory.ini` (apart from precision parameters)

  → `./class explanatory.ini`

- This is a *reference* file; we advise you to not modify it:
  - either start from a slim file (like `default.ini`),
  - or copy it and reduce it to a shorter and more friendly file,
  - or write your own from scratch with only needed input lines.

Usage (Command Line)

- Input .ini files

Usage (Command Line)

- Input .ini files

`parameter_name = value`

```
omega_b = 0.022
```

Usage (Command Line)

- Input .ini files

`parameter_name = value` `omega_b = 0.022`

`option = yes / no` `lensing = yes`

Usage (Command Line)

- Input .ini files

```
parameter_name = value
option = yes / no
selection = list, of, things
```

```
omega_b = 0.022
lensing = yes
output = tCl, pCl, lCl
```

Usage (Command Line)

- Input .ini files

```
parameter_name = value
option = yes / no
selection = list, of, things
# comment
```

```
omega_b = 0.022
lensing = yes
output = tCl, pCl, lCl
# To be or not to be?
```

Usage (Command Line)

- Files written to

  `<root>_xxx.dat`

Usage (Command Line)

- Files written to

  `<root>_xxx.dat`

- With `<root>` = `output/<filename>`, except when `root` is explicitly passed as input

Usage (Command Line)

- Files written to

  `<root>_xxx.dat`

- With `<root>` = `output/<filename>`, except when `root` is explicitly passed as input
- Command line file outputs: `write_background`/`write_thermodynamics`/ `write_primordial`

Usage (Command Line)

- Files written to

  `<root>_xxx.dat`

- With `<root>` = `output/<filename>`, except when `root` is explicitly passed as input
- Command line file outputs: `write_background`/`write_thermodynamics`/ `write_primordial`
- Other outputs: same as what we will discuss later

Usage (Python)

```
→ import classy
→ cosmo = classy.Class()
```

Usage (Python)

```
→ import classy
→ cosmo = classy.Class()
→ cosmo.set(input_dictionary)
```

Usage (Python)

```
→ import classy
→ cosmo = classy.Class()
→ cosmo.set(input_dictionary)
→ cosmo.compute()
```

Usage (Python)

```
→ import classy
→ cosmo = classy.Class()
→ cosmo.set(input_dictionary)
→ cosmo.compute()
→ cosmo.lensed_cl()
→ cosmo.get_pk_all(k,z)
→ cosmo.get_background()
→ cosmo.Hubble(z)
→ ...
```

Part Two

Inputs and outputs

How to work with class?

Input Parameters

- If nothing given:
  - Planck 2013 with $m_\nu=0$

Input Parameters

- If nothing given:
  - Planck 2013 with $m_\nu=0$
- Parameters overwritten by `cosmo.set` or `xxx.ini`

Input Parameters

- If nothing given:
  - Planck 2013 with m_ν=0
- Parameters overwritten by `cosmo.set` or `xxx.ini`
- precision parameter file:

  `cl_ref.pre` → close to 'optimal' precision

Input Parameters

- If nothing given:
  - Planck 2013 with m_ν=0
- Parameters overwritten by `cosmo.set` or `xxx.ini`
- precision parameter file:

  `cl_ref.pre` → close to 'optimal' precision
- Can also be included in `.ini`

Input Parameters

- **Which output to generate:**

  `output =`

  `tCl`

  C_L^{TT}

Input Parameters

- **Which output to generate:**

```
output =
tCl , pCl
```

C_L^{TT} C_L^{EE}

C_L^{TE}

Input Parameters

- **Which output to generate:**

```
output =
tCl , pCl , lCl
```

C_L^{TT} C_L^{EE} $C_L^{\phi\phi}$

C_L^{TE}

$C_L^{E\phi}$ $C_L^{T\phi}$

Nils Schöneberg nils.science@gmail.com

Input Parameters

- **Which output to generate:**

```
output =
tCl , pCl , lCl , mPk
```

C_L^{TT} C_L^{EE} $C_L^{\phi\phi}$ $P(k,z)$

C_L^{TE}

$C_L^{E\phi}$ $C_L^{T\phi}$

Nils Schöneberg nils.science@gmail.com

Input Parameters

- **Which output to generate:**

```
output =
tCl , pCl , lCl , mPk , mTk , vTk
```

C_L^{TT} C_L^{EE} $C_L^{\phi\phi}$ $P(k,z)$ $\delta_x(k,z^*)$ $\theta_x(k,z^*)$

C_L^{TE}

$C_L^{E\phi}$ $C_L^{T\phi}$

Nils Schöneberg nils.science@gmail.com

Input Parameters

- **Which output to generate:**

```
output =
tCl , pCl , lCl , mPk , mTk , vTk, nCl , sCl
```

C_L^{TT} C_L^{EE} $C_L^{\phi\phi}$ $P(k,z)$ $\delta_x(k,z^*)$ $\theta_x(k,z^*)$ $C_L^{\delta\delta}$ $C_L^{\Upsilon\Upsilon}$

C_L^{TE}

$C_L^{E\phi}$ $C_L^{T\phi}$

Input Parameters

- **Specifying output ranges:**

```
l_max_scalars = 5000
```

Input Parameters

- **Specifying output ranges:**

```
l_max_scalars = 5000
P_k_max_1/Mpc = 10
```

Input Parameters

- **Specifying output ranges:**

```
l_max_scalars = 5000
P_k_max_1/Mpc = 10
z_pk = 0, 1, 5, …        ← Command line
z_max_pk = 10            ← Python interface
```

Input Parameters

- **Specifying output ranges:**

```
l_max_scalars = 5000
P_k_max_1/Mpc = 10
z_pk = 0, 1, 5, …        ← Command line
z_max_pk = 10            ← Python interface
```

- **Useful things:**

```
format = class/CAMB      ← ! for N-body ICs
write_warnings =yes      ← ! for command line
```

Input Parameters

- **Verbosity (useful for debugging):**

```
input_verbose = 1
background_verbose = 1
thermodynamics_verbose = 1
perturbations_verbose = 1
transfer_verbose = 1
primordial_verbose = 1
harmonic_verbose = 1
fourier_verbose = 1
lensing_verbose = 1
distortions_verbose = 1
output_verbose = 1
```

Model Inputs

- **Usual ΛCDM model:** $\{H_0, A_s, n_s, \Omega_{\rm m}, \Omega_b h^2, \tau_{\rm reio}\}$
  - Hubble constant (basis of Hubble law)
  - `h = 0.67`
  - `H0 = 67`

Nils Schöneberg nils.science@gmail.com

Model Inputs

- **Usual ΛCDM model:** $\{H_0, A_s, n_s, \Omega_{\rm m}, \Omega_b h^2, \tau_{\rm reio}\}$
  - Hubble constant (basis of Hubble law)
  - `h = 0.67`
  - `H0 = 67`
  - `100*theta_s = 1.042`

$$\theta_s = \frac{r_s}{r_A} = \frac{H_0 r_s}{\int_0^{z_{\rm rec}} H_0/H(z) dz}$$

Model Inputs

- **Usual ΛCDM model:** $\{H_0, A_s, n_s, \Omega_{\rm m}, \Omega_b h^2, \tau_{\rm reio}\}$
  - Primordial amplitude of nearly scale-invariant power spectrum at pivot scale
  - `A_s = 2.01e-9`
  - `ln10^{10}A_s = 3.0` $\ln(10^{10} A_s)$
  - `ln_A_s_1e10 = 3.0`

Nils Schöneberg nils.science@gmail.com

Model Inputs

- **Usual ΛCDM model:** $\{H_0, A_s, n_s, \Omega_{\rm m}, \Omega_b h^2, \tau_{\rm reio}\}$
  - Primordial amplitude of nearly scale-invariant power spectrum at pivot scale
  - `A_s = 2.01e-9`
  - `ln10^{10}A_s = 3.0` $\ln(10^{10} A_s)$
  - `ln_A_s_1e10 = 3.0`
  - `sigma8` = **Notebook!**

$$\sigma_8(z) = \int \frac{k^3 P(k,z)}{2\pi^2} T^2(kR_8) d\ln k$$

Nils Schöneberg nils.science@gmail.com

Model Inputs

- **Usual ΛCDM model:** $\{H_0, A_s, n_s, \Omega_{\rm m}, \Omega_b h^2, \tau_{\rm reio}\}$
  - Primordial amplitude of nearly scale-invariant power spectrum at pivot scale
  - `A_s = 2.01e-9`
  - `ln10^{10}A_s = 3.0` $\ln(10^{10} A_s)$
  - `ln_A_s_1e10 = 3.0`
  - `sigma8 = 0.825`

$$\sigma_8(z) = \int \frac{k^3 P(k,z)}{2\pi^2} T^2(kR_8) d\ln k$$

Model Inputs

- **Usual ΛCDM model:** $\{H_0, A_s, n_s, \Omega_{\rm m}, \Omega_b h^2, \tau_{\rm reio}\}$
  - Primordial tilt of nearly scale-invariant power spectrum at pivot scale
  - `n_s = 0.96`

Nils Schöneberg nils.science@gmail.com

Model Inputs

- **Usual ΛCDM model:** $\{H_0, A_s, n_s, \Omega_\mathrm{m}, \Omega_b h^2, \tau_\mathrm{reio}\}$
  - Matter abundance as fraction of current-day energy density
  - `Omega_m = 0.3` Ω_m
  - `omega_m = 0.15` $\Omega_m h^2$
  - `Omega_cdm = 0.25` Ω_cdm
  - `omega_cdm = 0.125` $\Omega_\mathrm{cdm} h^2$

Model Inputs

- **Usual ΛCDM model:** $\{H_0, A_s, n_s, \Omega_{\rm m}, \Omega_b h^2, \tau_{\rm reio}\}$
  - Baryon abundance as fraction of current-day energy density
  - `Omega_b = 0.05` Ω_b
  - `omega_b = 0.025` $\Omega_b h^2$

Model Inputs

- **Usual ΛCDM model:** $\{H_0, A_s, n_s, \Omega_{\rm m}, \Omega_b h^2, \tau_{\rm reio}\}$
  - Reionization as optical depth/redshift
  - `tau_reio = 0.05`
  - `z_reio = 7`

$$\tau(z) = \int \kappa' d\eta = \int \sigma n_H(z) x_e(z) d\eta$$

Further Parameters

- **One-parameter extensions:**

```
Omega_k = 0
N_ur = 3.046
T_cmb = 2.7255
A_L = 1.0
alpha_s = 0
```

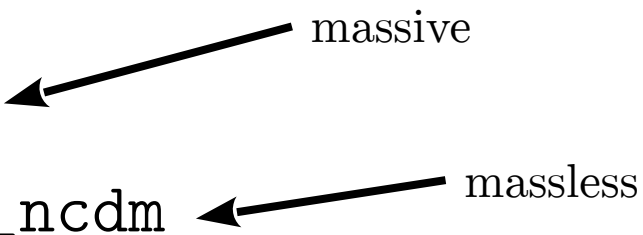
Dark Energy Models

- Cosmological constant `Omega_Lambda`
- fluid dark energy `Omega_fld`
  - `CPL` parameterization (`w0_fld`, `wa_fld`)
  - `EDE` is not the usual axion EDE → ***https://github.com/PoulinV/AxiCLASS*** and ***https://github.com/flo1984/TriggerCLASS*** still to be merged
- Quintessence `Omega_scf`

Neutrino parameters

- Ordinary free-streaming neutrinos:

```
N_ncdm = 1/2/3
m_ncdm = 0.02, 0.03, 0.05
N_ur = 3.046 - 1.0132 * N_ncdm
```

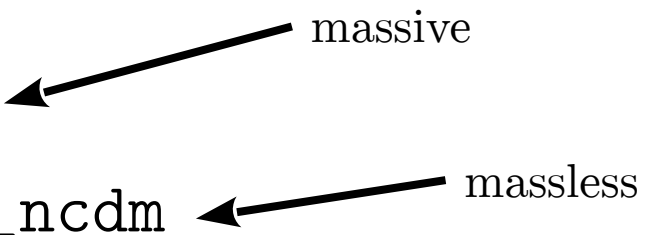


Neutrino parameters

- Ordinary free-streaming neutrinos:

```
N_ncdm = 1/2/3
m_ncdm = 0.02, 0.03, 0.05
N_ur = 3.046 - 1.0132 * N_ncdm

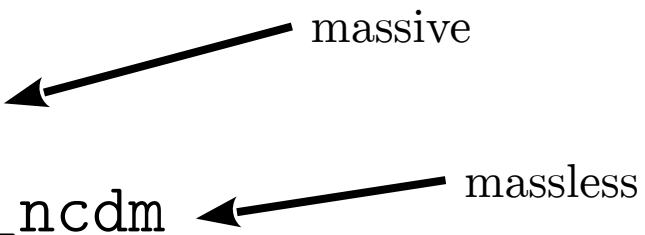
deg_ncdm = 1, 2
```



Neutrino parameters

- Ordinary free-streaming neutrinos:

```
N_ncdm = 1/2/3
m_ncdm = 0.02, 0.03, 0.05
N_ur = 3.046 - 1.0132 * N_ncdm
```




```
deg_ncdm = 1, 2

omega_ncdm = omega_cdm * f_wdm
T_ncdm = 0.71611 * (omega_ncdm*94.13/m_ncdm)^(1/3)
```

Nils Schöneberg nils.science@gmail.com

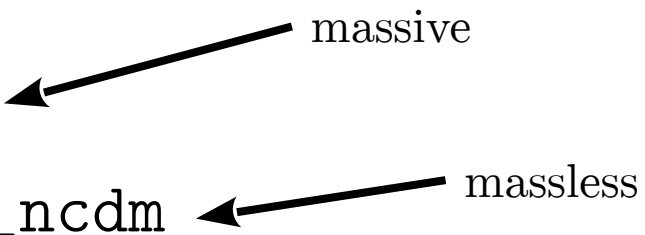
Neutrino parameters

- Ordinary free-streaming neutrinos:

```
N_ncdm = 1/2/3
m_ncdm = 0.02, 0.03, 0.03
N_ur = 3.046 - 1.0132 * N_ncdm

deg_ncdm = 1, 2

omega_ncdm = omega_cdm * f_wdm
T_ncdm = 0.71611 * (omega_ncdm*94.13/m_ncdm)^(1/3)
```



Also: non-trivial phase-space, neutrino flavor mixing, neutrino chemical potential

Dark Matter/Radiation Models

- Dark radiation extensions:
  - fluid/self-interacting dark radiation (`idr`)
  - Viscous dark radiation (`ceff2_ur`, `cvis_ur`)
  - `https://github.com/PoulinV/class_interacting_neutrinos`

Dark Matter/Radiation Models

- Dark radiation extensions:
  - fluid/self-interacting dark radiation (`idr`)
  - Viscous dark radiation (`ceff2_ur`, `cvis_ur`)
  - `https://github.com/PoulinV/class_interacting_neutrinos`
- Dark matter extensions:
  - Thermal WDM (see before)
  - Decaying dark matter (`dcdm`)
  - Interactions with photons, baryons, neutrinos (`idm`)

Inflation and ICs

A_s = 2.1e-9 / ln10^{10}A_s = 3.044 / sigma8 = 0.82

n_s = 0.966

alpha_s = 0

Inflation and ICs

```
A_s = 2.1e-9 / ln10^{10}A_s = 3.044 / sigma8 = 0.82
n_s = 0.966
alpha_s = 0
```

```
f_bi = 0
```
b isocurvature fraction

```
c_ad_bi = 0
```
b isocurvature correlation

Nils Schöneberg nils.science@gmail.com

Inflation and ICs

```
A_s = 2.1e-9 / ln10^{10}A_s = 3.044 / sigma8 = 0.82
n_s = 0.966
alpha_s = 0
```

```
f_bi = 0
```
b isocurvature fraction

```
c_ad_bi = 0
```
b isocurvature correlation

```
r = 1
```
tensor-scalar ratio

```
n_t = scc
```
tensor tilt (self-consistency single-field inflation)

Precision Parameters

- **Precision parameters (`include/precisions.h`):**

```
P_k_max_h/Mpc = 1.
```

Precision Parameters

- **Precision parameters (`include/precisions.h`):**

  `P_k_max_h/Mpc = 1.`

  `k_per_decade_for_pk = 10`

Precision Parameters

- **Precision parameters (`include/precisions.h`):**

```
P_k_max_h/Mpc = 1.
k_per_decade_for_pk = 10
k_max_tau0_over_lmax = 2.4
```

Precision Parameters

- **Precision parameters (include/precisions.h):**

```
P_k_max_h/Mpc = 1.
k_per_decade_for_pk = 10
k_max_tau0_over_lmax = 2.4
k_min_tau0 = 0.1
tight_coupling_approximation /
radiation_streaming_approximation /
ur_fluid_approximation / ncdm_fluid_approximation
```

Precision Parameters

- **Precision parameters (include/precisions.h):**

```
P_k_max_h/Mpc = 1.

k_per_decade_for_pk = 10

k_max_tau0_over_lmax = 2.4

k_min_tau0 = 0.1

tight_coupling_approximation /
radiation_streaming_approximation /
ur_fluid_approximation / ncdm_fluid_approximation

accurate_lensing = 1
```

Precision Parameters

- **Precision parameters (`include/precisions.h`):**

  `background_Nloga = 3000`

  `tol_background_integration = 1e-10`

Precision Parameters

- **Precision parameters (`include/precisions.h`):**

  `background_Nloga = 3000`

  `tol_background_integration = 1e-10`

  `tol_ncdm = 1e-3 / tol_ncdm_synchronous = 1e-3 / tol_ncdm_bg = 1e-5`

Precision Parameters

- **Precision parameters (`include/precisions.h`):**

  `background_Nloga = 3000`

  `tol_background_integration = 1e-10`

  `tol_ncdm = 1e-3 / tol_ncdm_synchronous = 1e-3 / tol_ncdm_bg = 1e-5`

  `tol_shooting_deltax = 1e-4 / tol_shooting_deltaF = 1e-6 /`
  `tol_shooting_deltax_rel = 1e-5 (1D)`

Precision Parameters

- **Precision parameters (`include/precisions.h`):**

  `background_Nloga = 3000`

  `tol_background_integration = 1e-10`

  `tol_ncdm = 1e-3 / tol_ncdm_synchronous = 1e-3 / tol_ncdm_bg = 1e-5`

  `tol_shooting_deltax = 1e-4 / tol_shooting_deltaF = 1e-6 /`
  `tol_shooting_deltax_rel = 1e-5 (1D)`

  `thermo_z_initial = 5e6 / thermo_Nz_lin = 20000 / thermo_Nz_log = 5000`

  `tol_thermo_integration = 1e-6`

Precision Parameters

- **Precision parameters (include/precisions.h):**

  background_Nloga = 3000

  tol_background_integration = 1e-10

  tol_ncdm = 1e-3 / tol_ncdm_synchronous = 1e-3 / tol_ncdm_bg = 1e-5

  tol_shooting_deltax = 1e-4 / tol_shooting_deltaF = 1e-6 /
  tol_shooting_deltax_rel = 1e-5 (1D)

  thermo_z_initial = 5e6 / thermo_Nz_lin = 20000 / thermo_Nz_log = 5000

  tol_thermo_integration = 1e-6

  perturbations_sampling_stepsize = 0.1

  tol_perturbations_integration = 1e-5

Quick moment to catch your breath!

$\rightarrow$ On to outputs

Nils Schöneberg nils.science@gmail.com

Output functions (classy)

`raw_cl(lmax=-1)`	$C_\ell^{\rm unlensed}$
`lensed_cl(lmax=-1)`	$C_\ell^{\rm lensed}$
`density_cl(lmax=-1)`	$C_\ell^{\rm galaxy}$

Nils Schöneberg nils.science@gmail.com

Output functions (classy)

`raw_cl(lmax=-1)`	$C_\ell^{\rm unlensed}$
`lensed_cl(lmax=-1)`	$C_\ell^{\rm lensed}$
`density_cl(lmax=-1)`	$C_\ell^{\rm galaxy}$

`get_pk_all(k,z, nonlinear=True, cdmbar=False)` $P(k,z)$

Nils Schöneberg nils.science@gmail.com

Output functions (classy)

`raw_cl(lmax=-1)`	$C_\ell^{\rm unlensed}$
`lensed_cl(lmax=-1)`	$C_\ell^{\rm lensed}$
`density_cl(lmax=-1)`	$C_\ell^{\rm galaxy}$

`get_pk_all(k,z, nonlinear=True, cdmbar=False)`	$P(k,z)$
`get_pk_and_k_and_z(nonlinear=True, only_clustering_species=False, h_units=False)`	
`get_transfer_and_k_and_z(output_format='class', h_units=False)`	$\delta(k,z), \theta(k,z)$

Nils Schöneberg nils.science@gmail.com

Output functions (classy)

`raw_cl(lmax=-1)` $C_\ell^{\rm unlensed}$

`lensed_cl(lmax=-1)` $C_\ell^{\rm lensed}$

`density_cl(lmax=-1)` $C_\ell^{\rm galaxy}$

`get_pk_all(k,z, nonlinear=True, cdmbar=False)` $P(k,z)$

`get_pk_and_k_and_z(nonlinear=True, only_clustering_species=False, h_units=False)`

`get_transfer_and_k_and_z(output_format='class', h_units=False)` $\delta(k,z), \theta(k,z)$

`get_background()`, `get_thermodynamics()`, `get_primordial()`, `get_perturbations()`, `get_transfer(z)`

Output functions (classy)

```
sigma8()
h(), n_s(), theta_s_100()
tau_reio(), z_reio()
Omega_m()
Omega_r(), Omega_Lambda(), Omega_g(), Omega_b()
Neff()
keq(), age(), rs_drag()
```

Output functions (classy)

```
sigma8()
h(), n_s(), theta_s_100()
tau_reio(), z_reio()
Omega_m()
Omega_r(), Omega_Lambda(), Omega_g(), Omega_b()
Neff()
keq(), age(), rs_drag()

get_current_derived_parameters(names) → classy.pyx
```

Output functions (classy)

- `z_of_r(z)` → `conf_distance(z)`, `H(z)` [1/Mpc]
- `luminosity_distance(z)`, `angular_distance(z)`, `angular_distance_from_to(z1,z2)`, `comoving_distance(z)`

Output functions (classy)

- `z_of_r(z)` → `conf_distance(z)`, `H(z)` [1/Mpc]
- **`luminosity_distance(z)`, `angular_distance(z)`, `angular_distance_from_to(z1,z2)`, `comoving_distance(z)`**
- **`scale_independent_growth_factor(z)`** → `D(z)`
- `scale_independent_growth_factor_f(z)` → `f(z) = dlnD/dlna`

Nils Schöneberg nils.science@gmail.com

Output functions (classy)

- z_of_r(z) → conf_distance(z), H(z) [1/Mpc]
- **luminosity_distance(z), angular_distance(z), angular_distance_from_to(z1,z2), comoving_distance(z)**
- **scale_independent_growth_factor(z)** → D(z)
- scale_independent_growth_factor_f(z) → f(z) = dlnD/dlna

- **sigma(R,z,hunits=False)** / sigma8() → sigma(8,0,hunits=True)
- scale_independent_f_sigma8()
- effective_f_sigma8() → dsigma8/dlna

Nils Schöneberg nils.science@gmail.com

Output functions (classy)

- z_of_r(z) → conf_distance(z), H(z) [1/Mpc]
- **luminosity_distance(z), angular_distance(z), angular_distance_from_to(z1,z2), comoving_distance(z)**
- **scale_independent_growth_factor(z)** → D(z)
- scale_independent_growth_factor_f(z) → f(z) = dlnD/dlna

- **sigma(R,z,hunits=False)** / sigma8() → sigma(8,0,hunits=True)
- scale_independent_f_sigma8()
- effective_f_sigma8() → dsigma8/dlna

- z_of_tau(tau)
- **Hubble(z)**
- Om_m(z) /Om_b(z), Om_cdm(z), Om_ncdm(z)

Nils Schöneberg nils.science@gmail.com

Output functions (classy)

- z_of_r(z) → conf_distance(z), H(z) [1/Mpc]
- **luminosity_distance(z), angular_distance(z), angular_distance_from_to(z1,z2), comoving_distance(z)**
- **scale_independent_growth_factor(z)** → D(z)
- scale_independent_growth_factor_f(z) → f(z) = dlnD/dlna

- **sigma(R,z,hunits=False)** / sigma8() → sigma(8,0,hunits=True)
- scale_independent_f_sigma8()
- effective_f_sigma8() → dsigma8/dlna

- z_of_tau(tau)
- **Hubble(z)**
- Om_m(z) /Om_b(z), Om_cdm(z), Om_ncdm(z)

- pk_tilt(k,z)

- ionization_function(z) / baryon_temperature(z)

Short notebook break!

Part Three

The layout of class (internal and external)

Layout of CLASS

- Goal: Computing *summary statistics*

Layout of CLASS

- Goal: Computing *summary statistics*
- For this, we require *perturbations*

Layout of CLASS

- Goal: Computing *summary statistics*
- For this, we require *perturbations*
- For this, we require *initial conditions* (inflation) and *interaction rates* (thermodynamics) and *background quantities* (like Hubble, densities)

Layout of CLASS

- Goal: Computing *summary statistics*
- For this, we require *perturbations*
- For this, we require *initial conditions* (inflation) and *interaction rates* (thermodynamics) and *background quantities* (like Hubble, densities)

  $\rightarrow$ This suggests a causal code layout

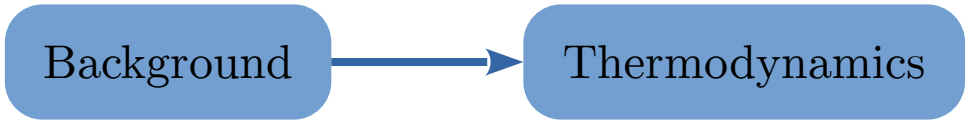
Nils Schöneberg nils.science@gmail.com

Layout of CLASS

Background

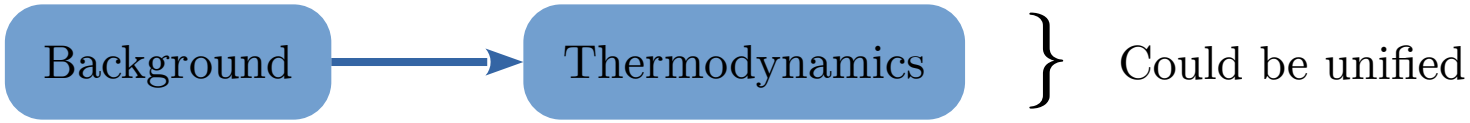
Nils Schöneberg nils.science@gmail.com

Layout of CLASS



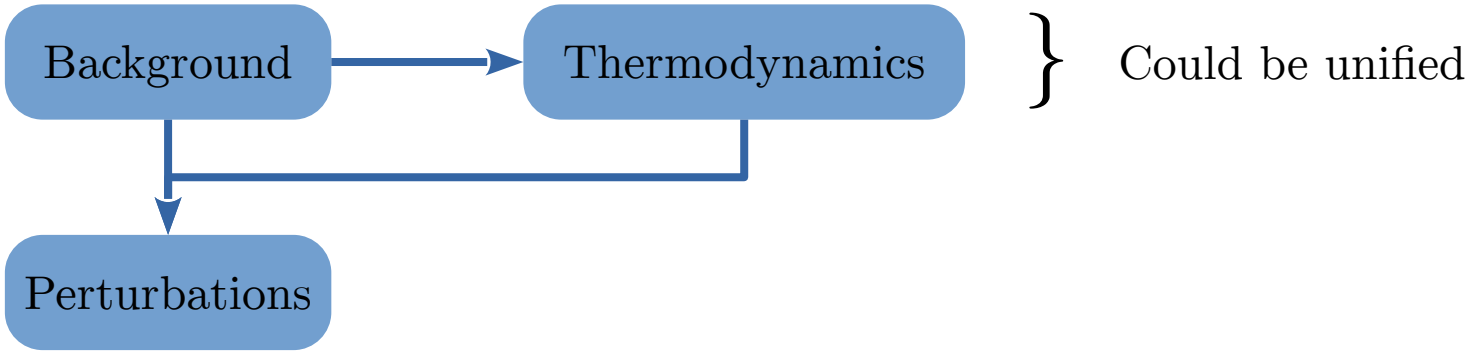
Nils Schöneberg nils.science@gmail.com

Layout of CLASS



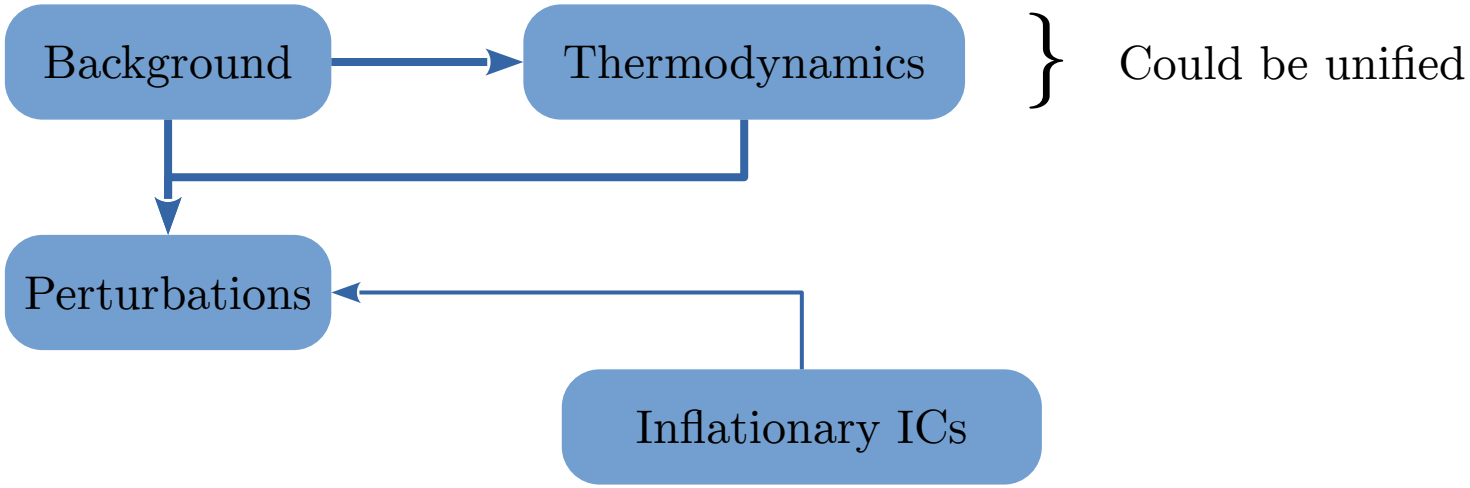
Nils Schöneberg nils.science@gmail.com

Layout of CLASS



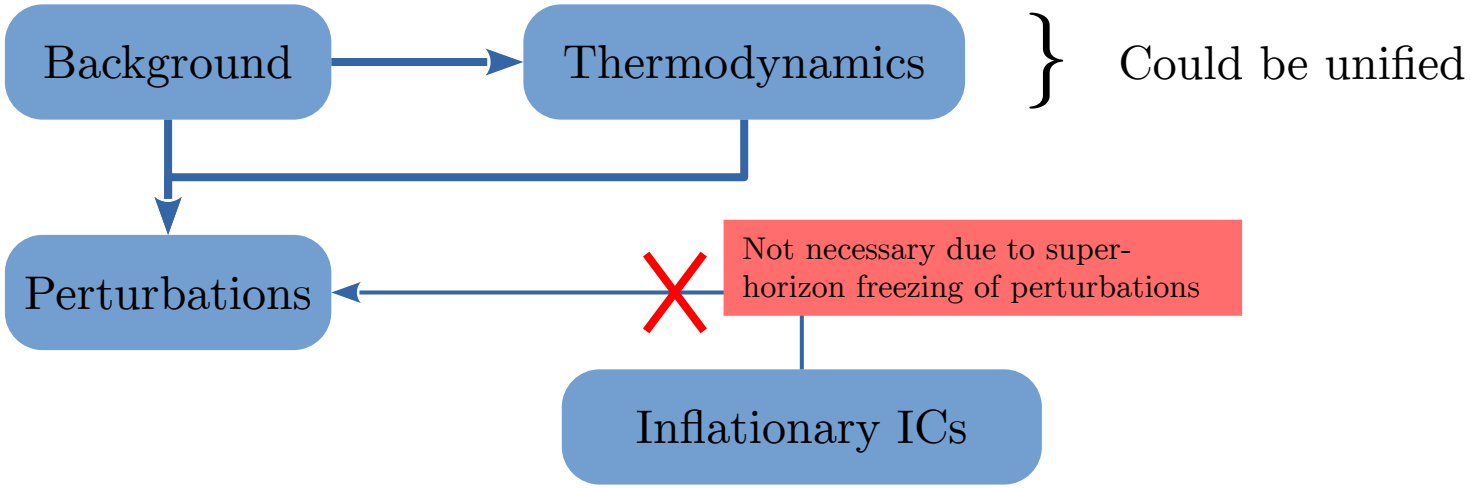
Nils Schöneberg nils.science@gmail.com

Layout of CLASS

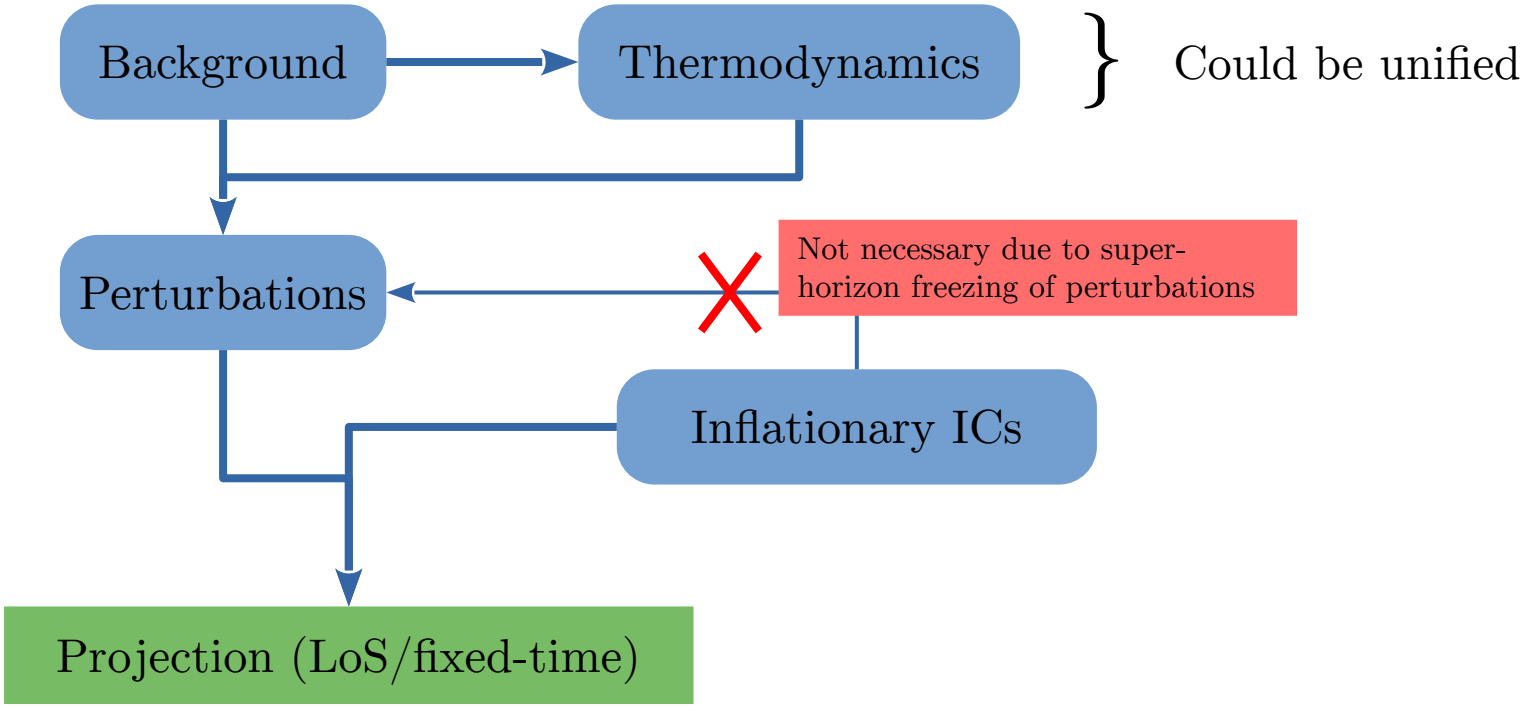


Nils Schöneberg nils.science@gmail.com

Layout of CLASS

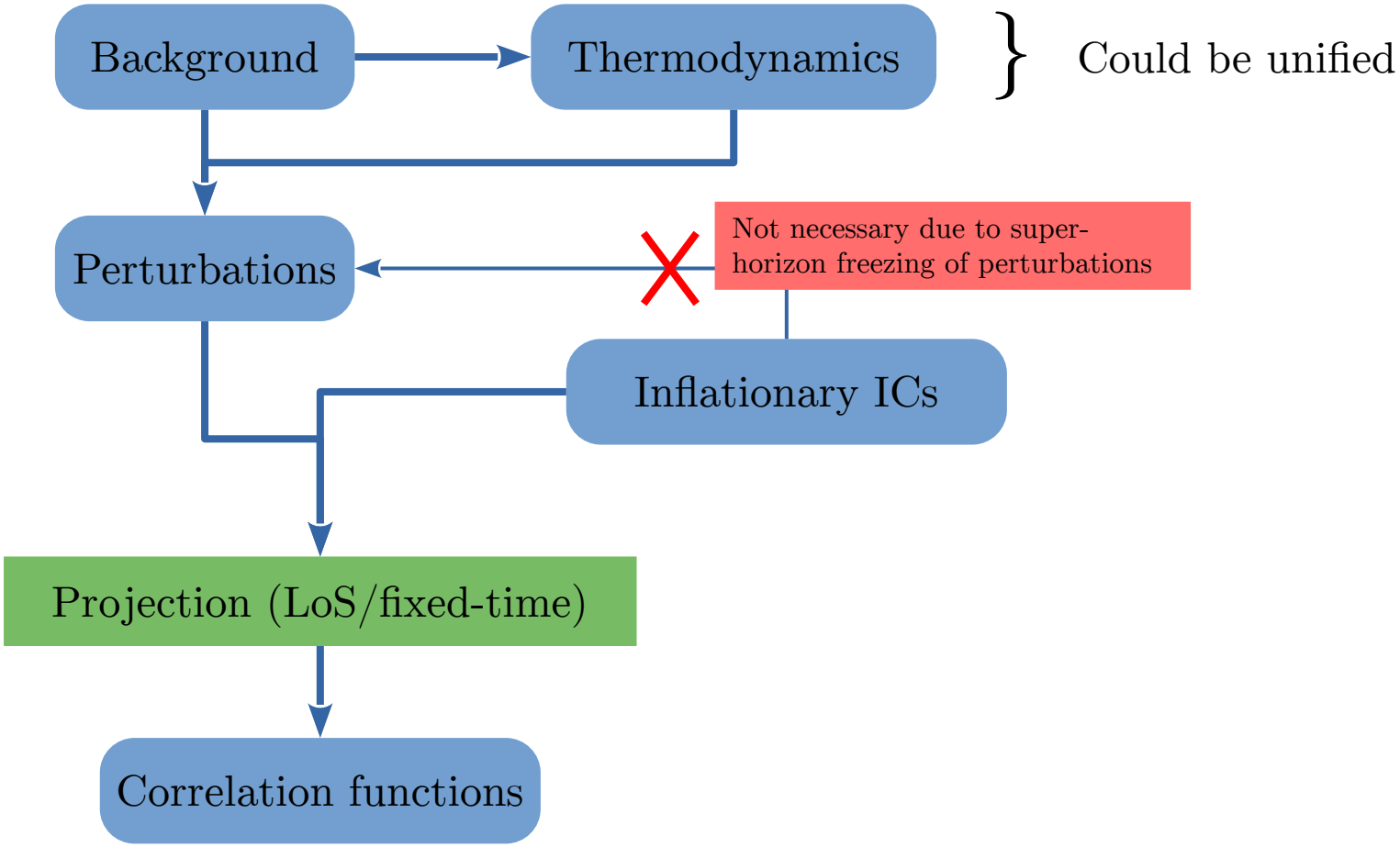


Layout of CLASS



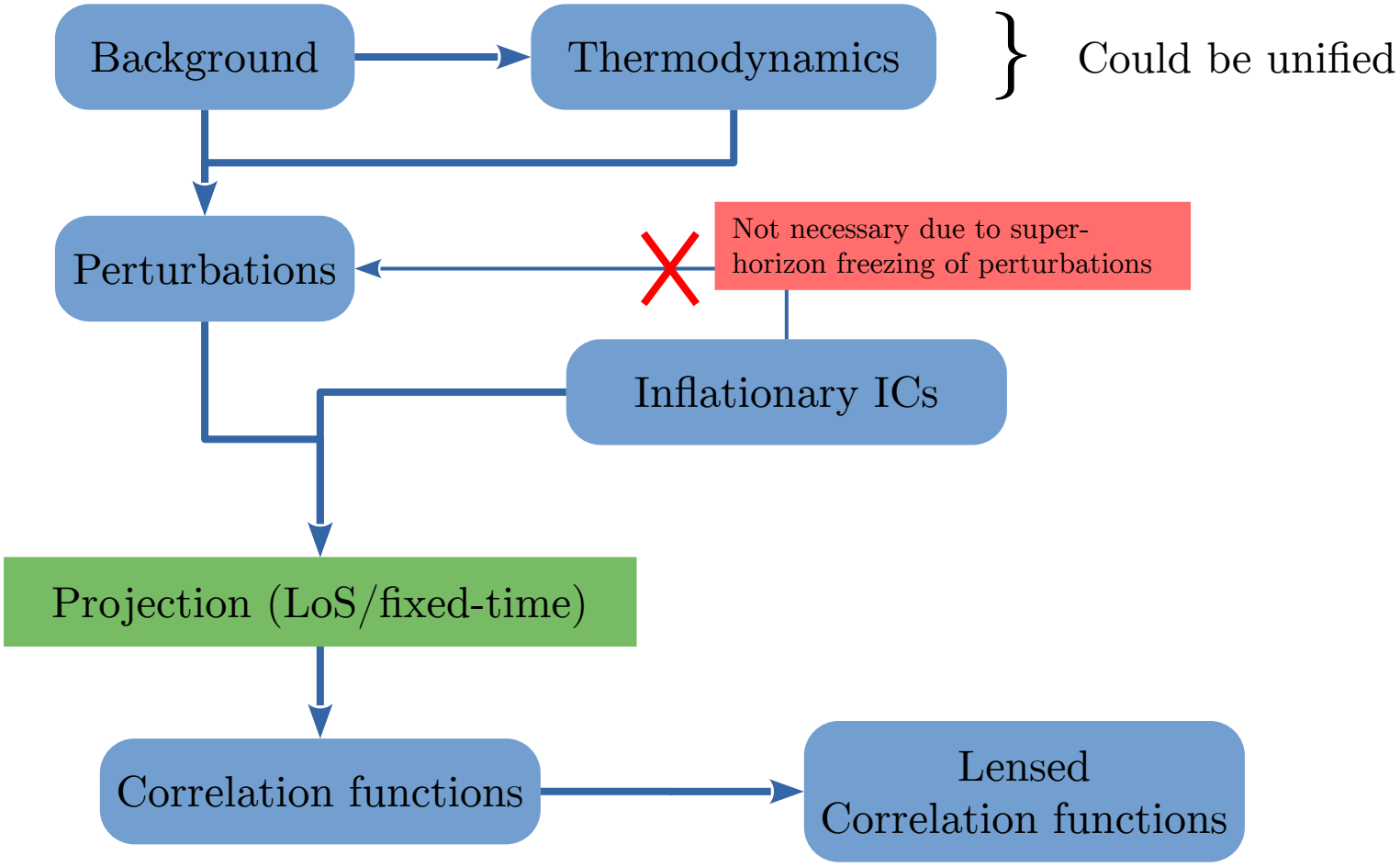
Nils Schöneberg nils.science@gmail.com

Layout of CLASS



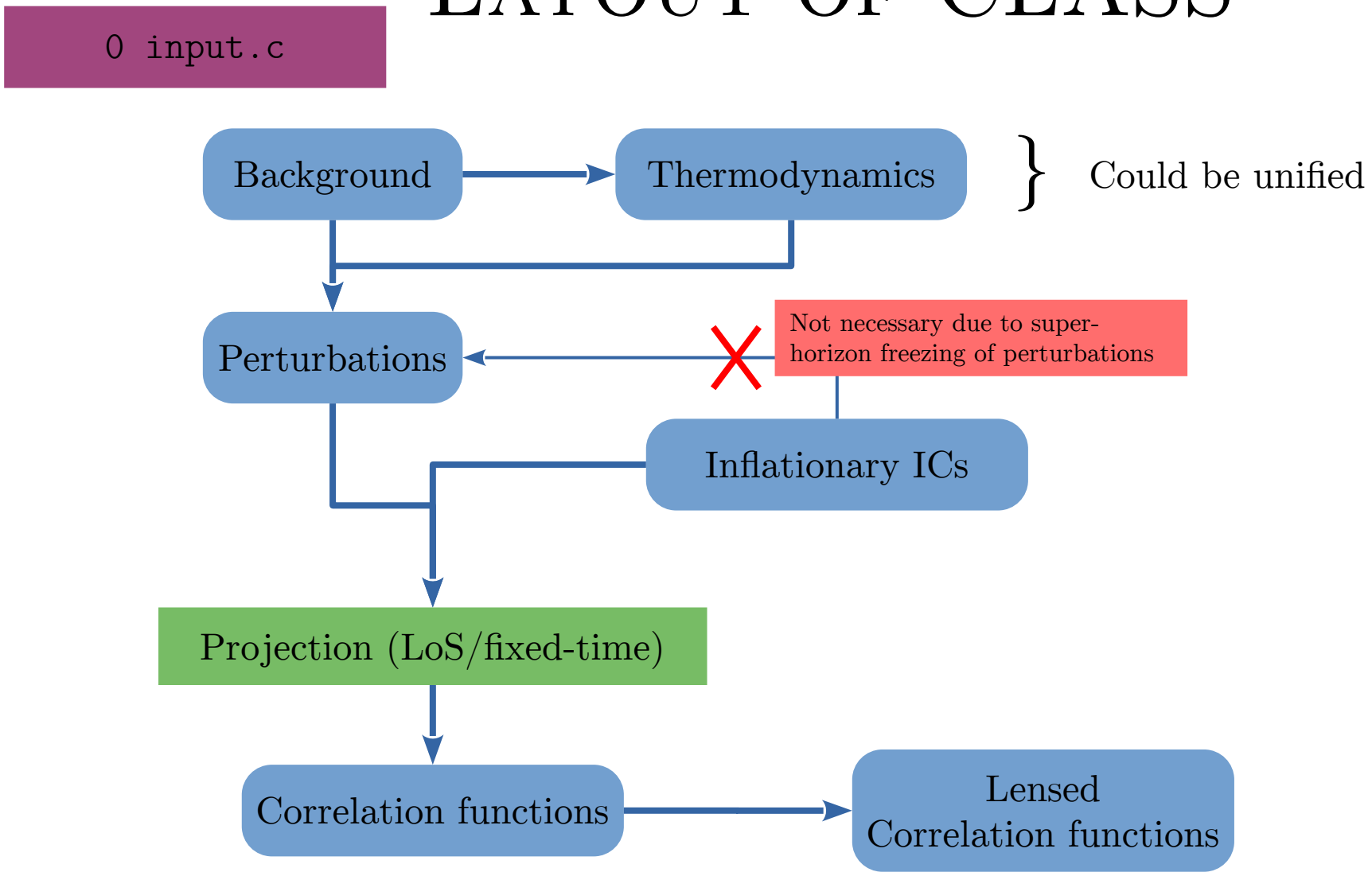
Nils Schöneberg nils.science@gmail.com

Layout of CLASS



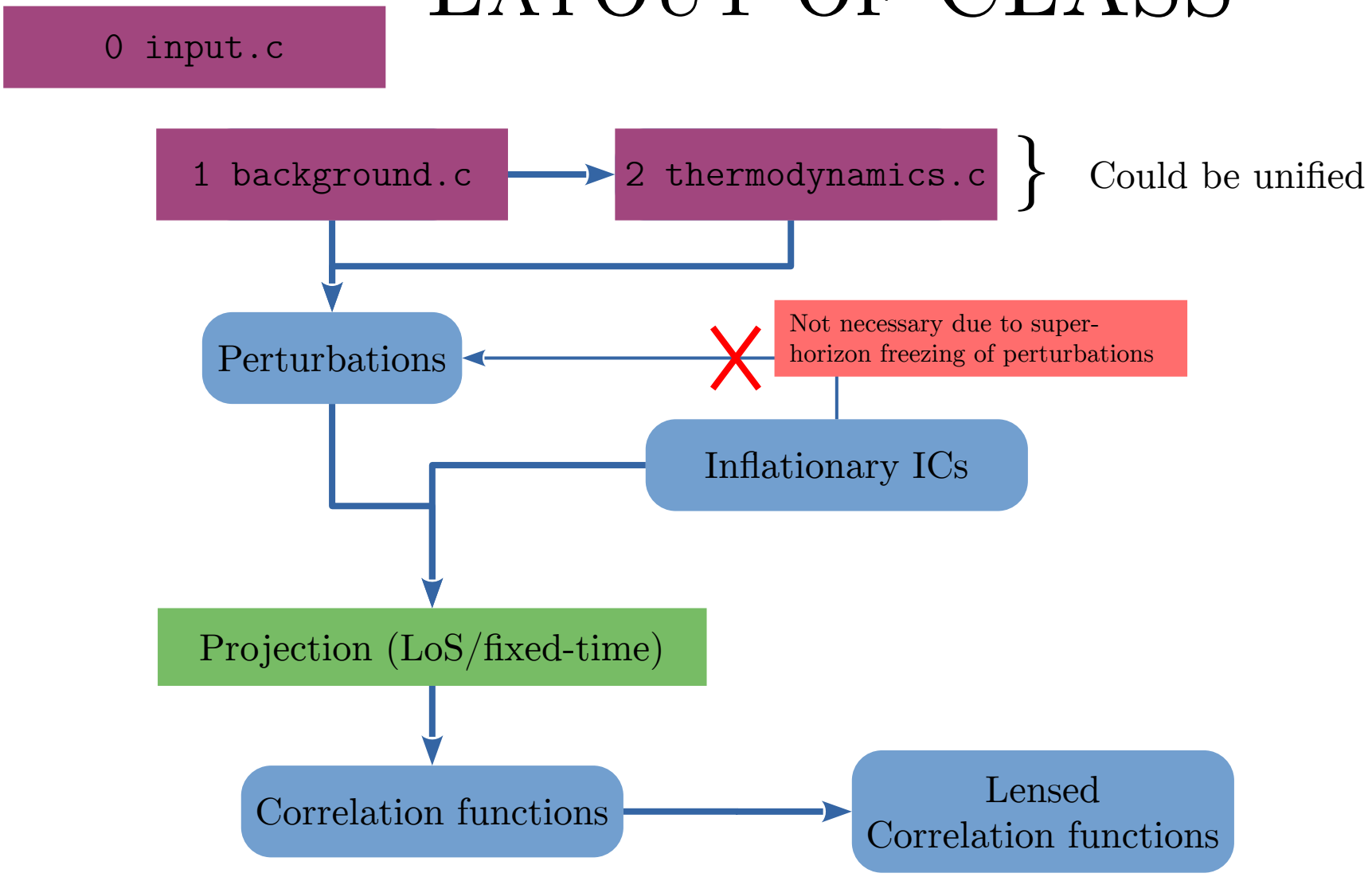
Nils Schöneberg nils.science@gmail.com

Layout of CLASS



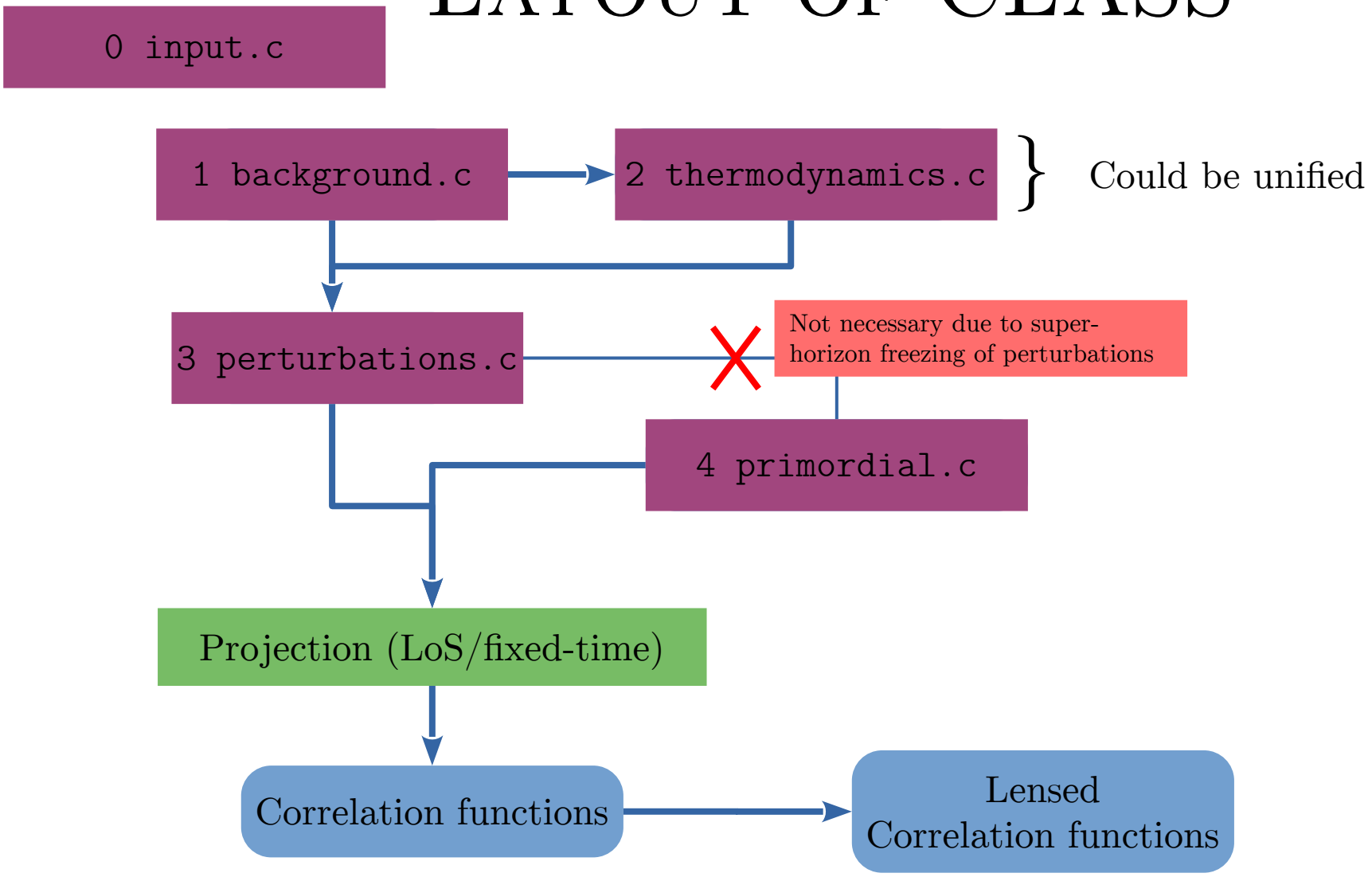
Nils Schöneberg nils.science@gmail.com

Layout of CLASS



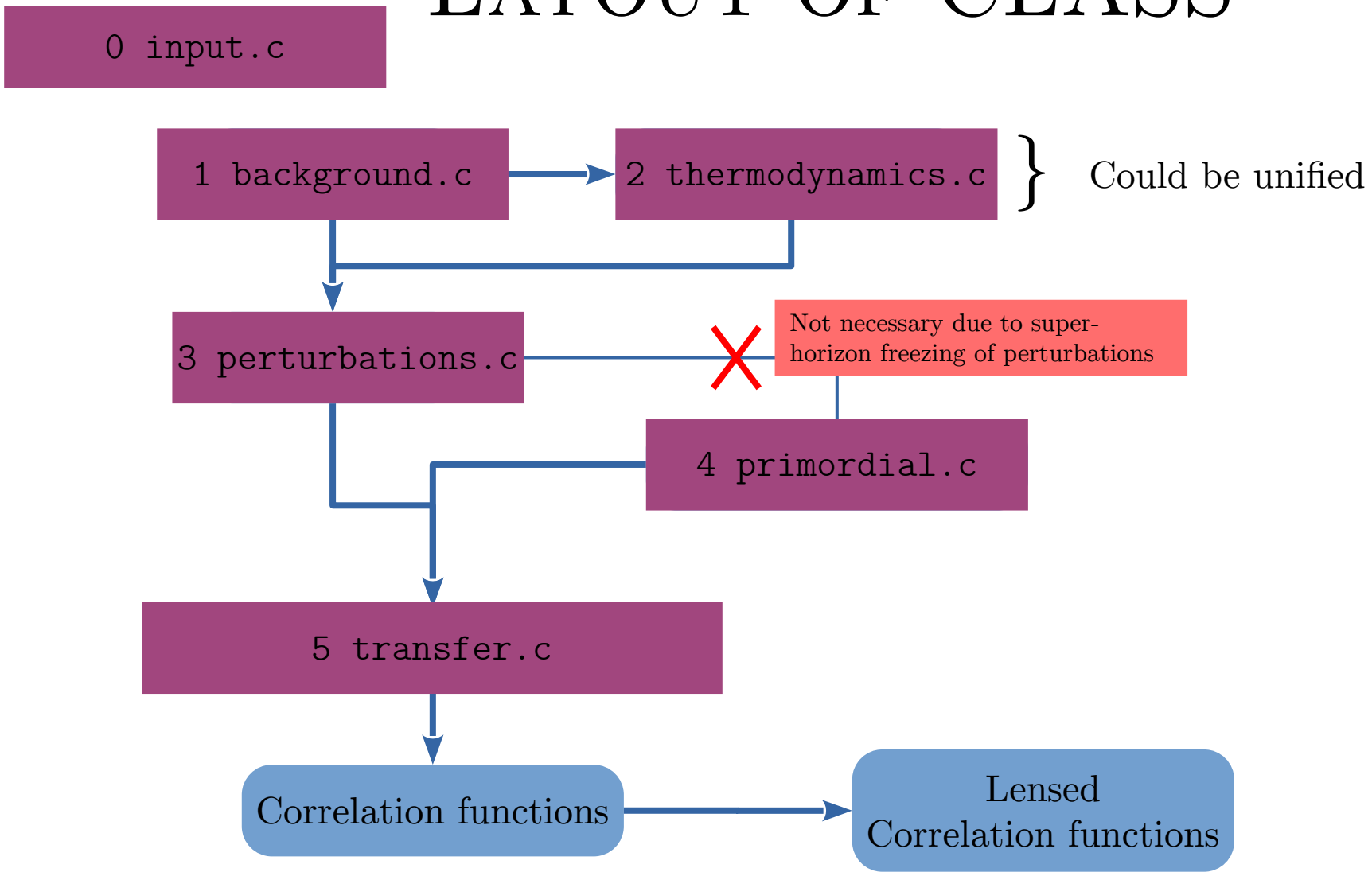
Nils Schöneberg nils.science@gmail.com

Layout of CLASS



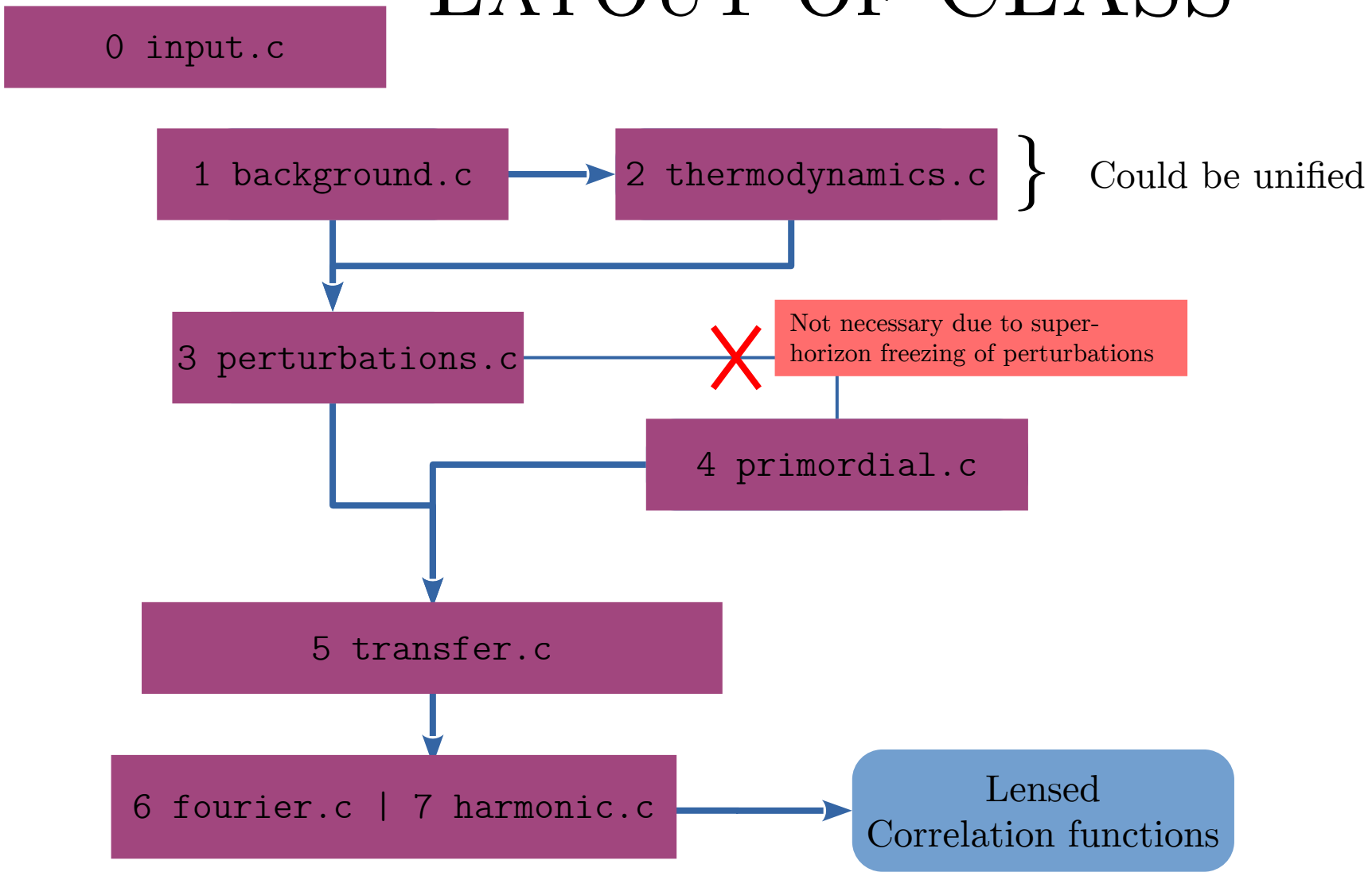
Nils Schöneberg nils.science@gmail.com

Layout of CLASS



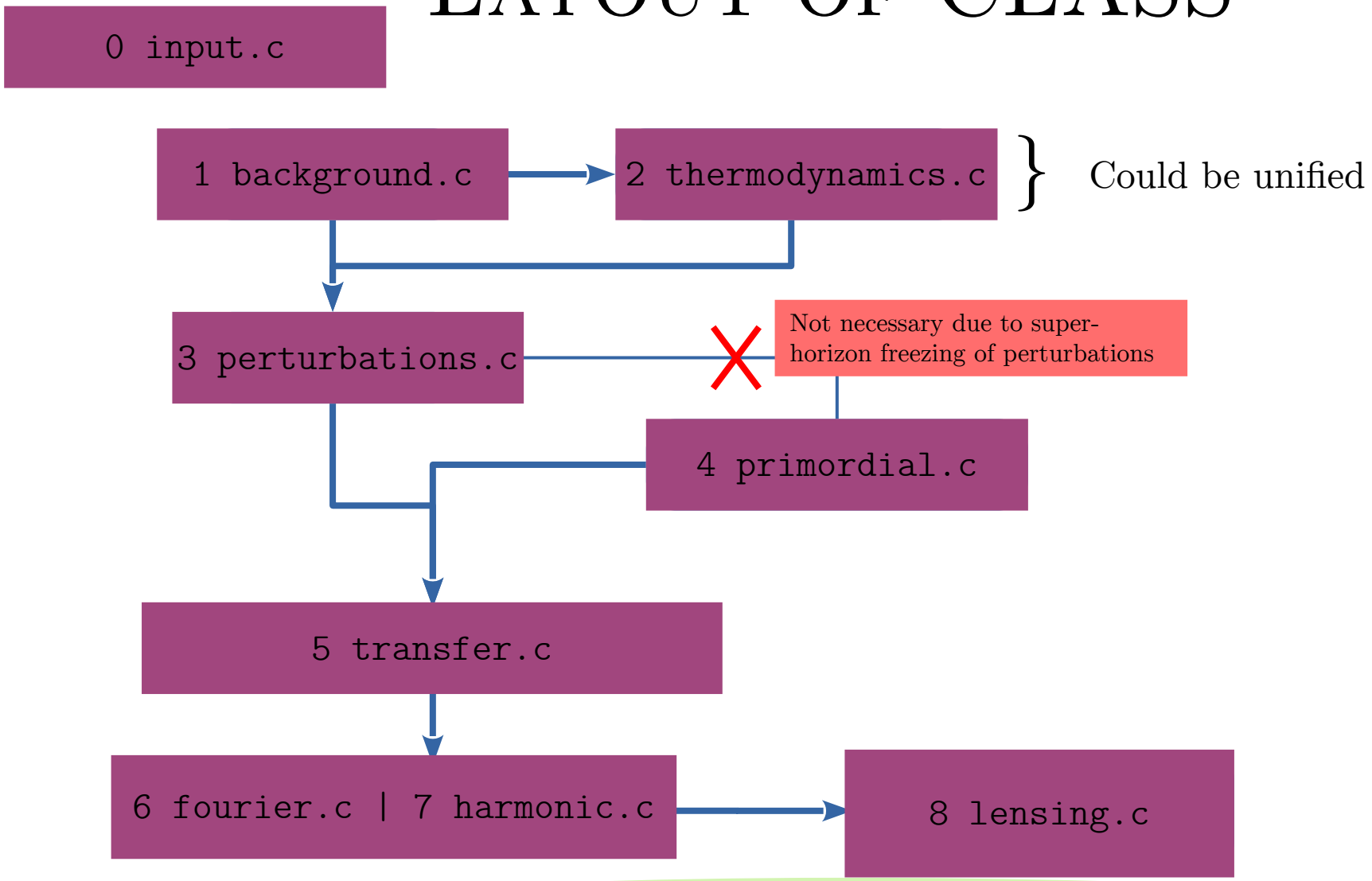
Nils Schöneberg nils.science@gmail.com

Layout of CLASS



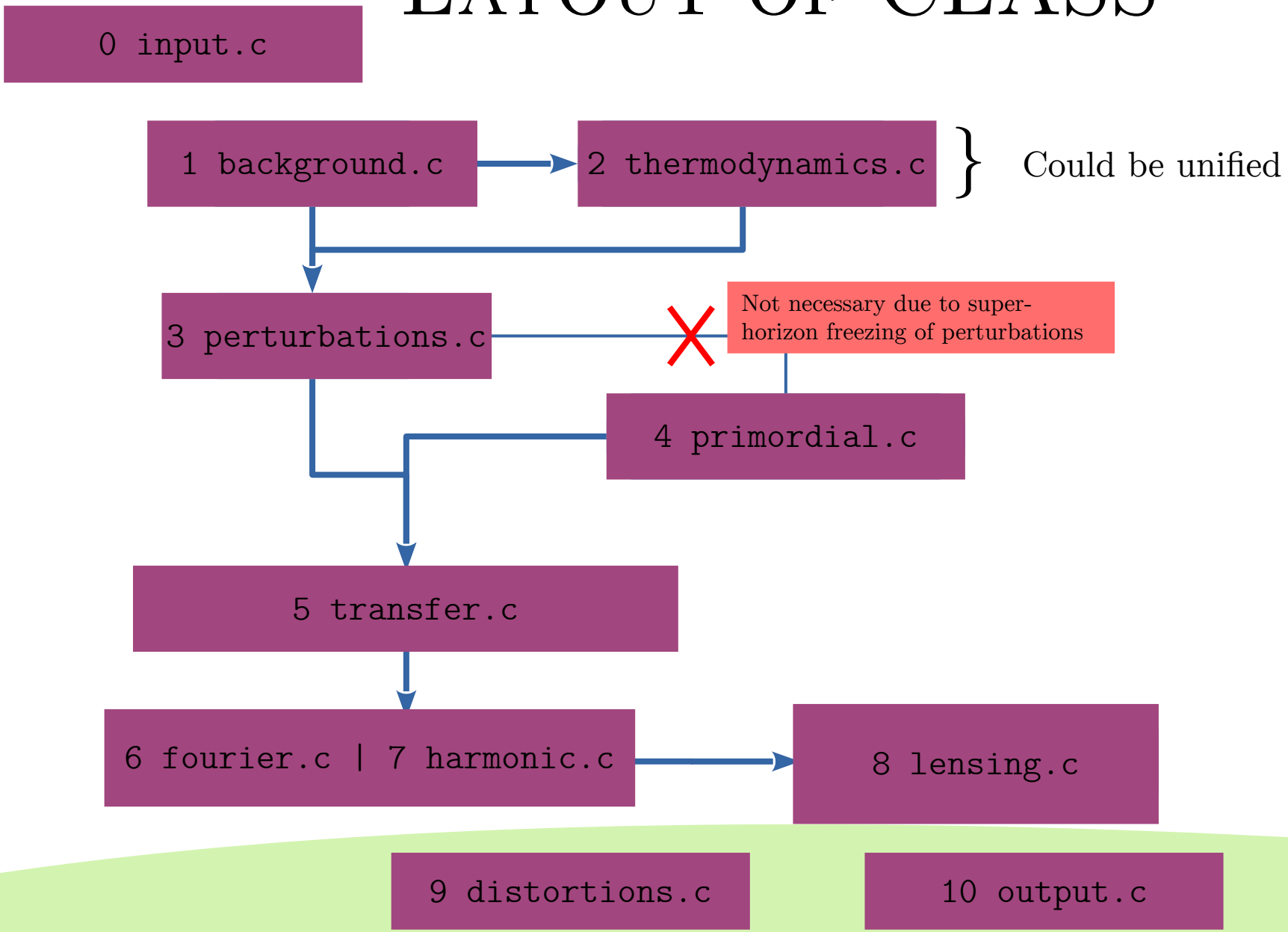
Nils Schöneberg nils.science@gmail.com

Layout of CLASS



Nils Schöneberg nils.science@gmail.com

Layout of CLASS



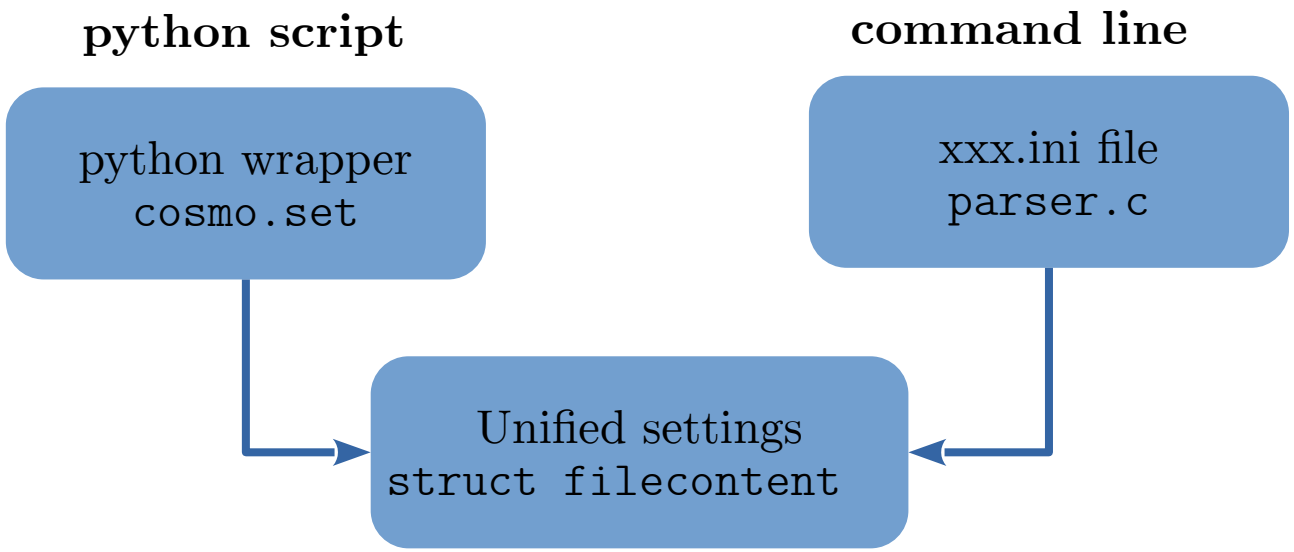
Nils Schöneberg nils.science@gmail.com

Input Layout of CLASS

python script **command line**

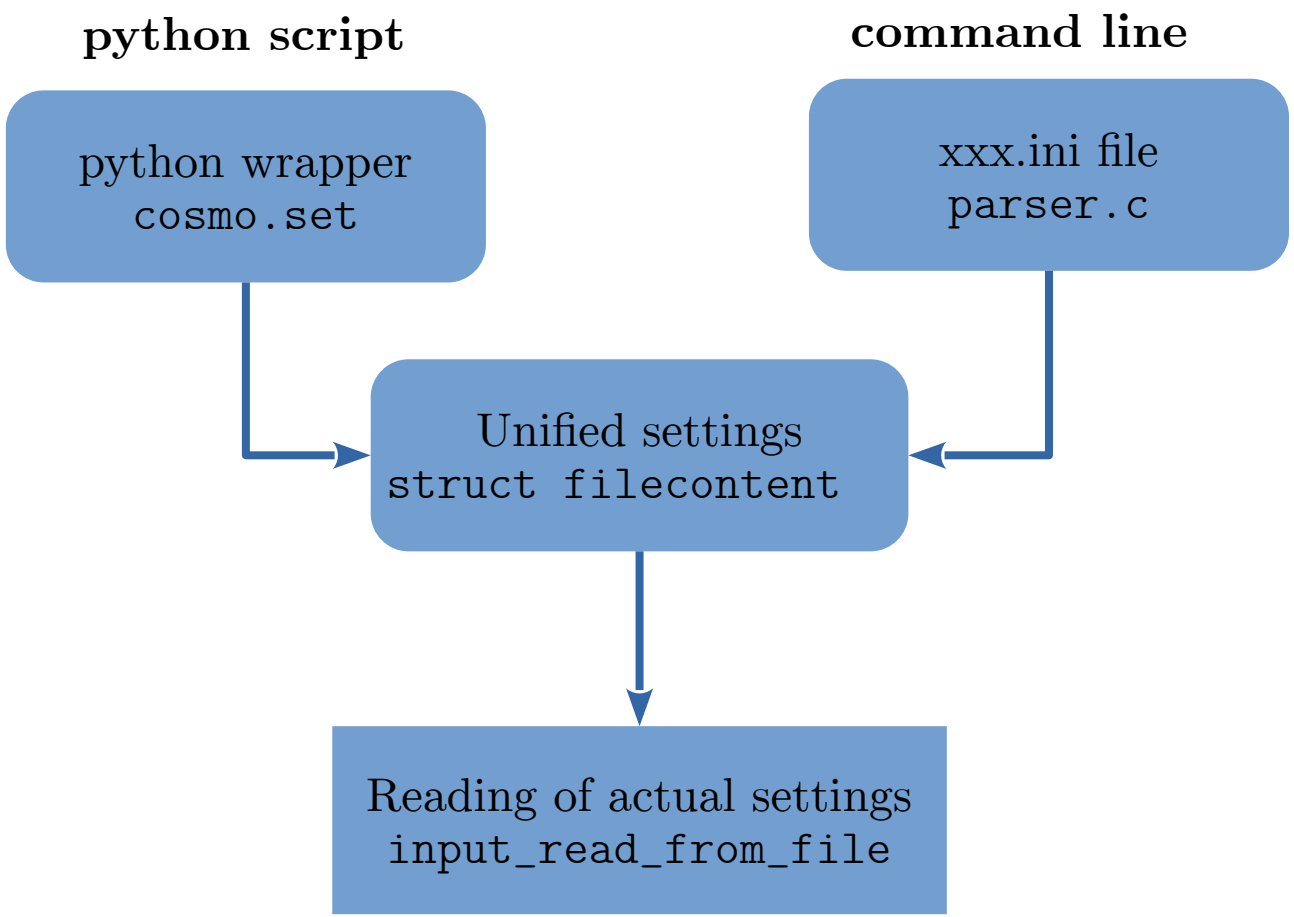
Nils Schöneberg nils.science@gmail.com

Input Layout of CLASS

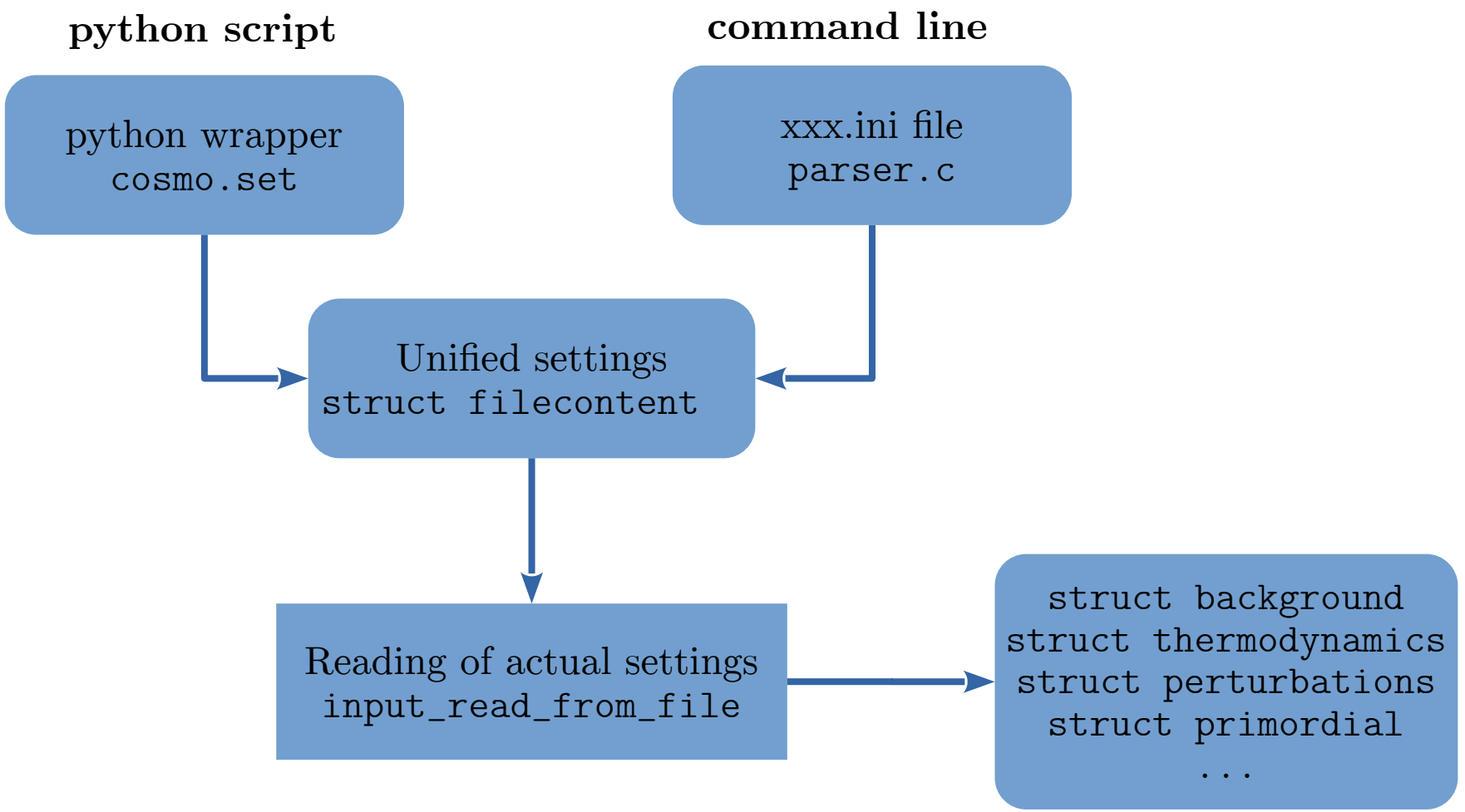


Nils Schöneberg nils.science@gmail.com

Input Layout of CLASS



Input Layout of CLASS



Shooting

Allows us to use parameters like $100_{\,s}$ or σ_8 even though they are technically outputs not inputs

Shooting

Try out a *reasonable* value (with a pre-programmed guess)

Then iterate in *correct* direction

Allows us to use parameters like $100_{\,\mathrm{s}}$ or σ_8 even though they are technically outputs not inputs

Shooting

Try out a *reasonable* value (with a pre-programmed guess)

Then iterate in *correct* direction

1D :	More robust Ridder's method
2D or higher:	Less robust but faster Newton's method

Nils Schöneberg nils.science@gmail.com

Shooting

Try out a *reasonable* value (with a pre-programmed guess)

Then iterate in *correct* direction

1D : More robust Ridder's method

2D or higher: Less robust but faster Newton's method

τ_{reio} (mini-shooting in th.c)
σ_8
$100\ _s$
(f_{ede} , ...)

Allows us to use parameters like $100\ _s$ or σ_8 even though they are technically outputs not inputs

Nils Schöneberg nils.science@gmail.com

Shooting

τ_{reio} (mini-shooting in th.c)
σ_8
$100\ \theta_s$
(f_{ede} , ...)

Try out a *reasonable* value (with a pre-programmed guess)

Then iterate in *correct* direction

1D : More robust Ridder's method

2D or higher: Less robust but faster Newton's method

h	$100\ \theta_s$
0.70000000000	1.0522492086422521

← Precision reached

Allows us to use parameters like $100\ \theta_s$ or σ_8 even though they are technically outputs not inputs

Nils Schöneberg nils.science@gmail.com

Shooting

τ_{reio} (mini-shooting in th.c)
σ_8
$100\ \theta_s$
(f_{ede} , ...)

Try out a *reasonable* value (with a pre-programmed guess)

Then iterate in *correct* direction

1D : More robust Ridder's method

2D or higher: Less robust but faster Newton's method

h	$100\ \theta_s$
0.70000000000	1.0522492086422521
0.65000000000	1.0270326366580724

← Precision reached

Allows us to use parameters like $100\ \theta_s$ or σ_8 even though they are technically outputs not inputs

Nils Schöneberg nils.science@gmail.com

Shooting

τ_{reio} (mini-shooting in th.c)
σ_8
$100\ \theta_s$
(f_{ede} , ...)

Try out a *reasonable* value (with a pre-programmed guess)

Then iterate in *correct* direction

1D : More robust Ridder's method

2D or higher: Less robust but faster Newton's method

h	$100\ \theta_s$
0.70000000000	1.0522492086422521
0.65000000000	1.0270326366580724
0.68215616173	1.0437999980620178

← Precision reached

Allows us to use parameters like $100\ \theta_s$ or σ_8 even though they are technically outputs not inputs

Nils Schöneberg nils.science@gmail.com

Shooting

τ_{reio} (mini-shooting in th.c)
σ_8
$100\ \theta_s$
(f_{ede} , ...)

Try out a *reasonable* value (with a pre-programmed guess)

Then iterate in *correct* direction

1D : More robust Ridder's method

2D or higher: Less robust but faster Newton's method

h	$100\ \theta_s$
0.70000000000	1.0522492086422521
0.65000000000	1.0270326366580724
0.68215616173	1.0437999980620178
0.68110138476	1.0432819283581667

← Precision reached

Allows us to use parameters like $100\ \theta_s$ or σ_8 even though they are technically outputs not inputs

Nils Schöneberg nils.science@gmail.com

Shooting

τ_{reio} (mini-shooting in th.c)
σ_8
$100\ \theta_s$
(f_{ede} , ...)

Try out a *reasonable* value (with a pre-programmed guess)

Then iterate in *correct* direction

1D : More robust Ridder's method

2D or higher: Less robust but faster Newton's method

h	$100\ \theta_s$
0.70000000000	1.0522492086422521
0.65000000000	1.0270326366580724
0.68215616173	1.0437999980620178
0.68110138476	1.0432819283581667
0.6810337942	1.0432499363679562

← Precision reached

Allows us to use parameters like $100\ \theta_s$ or σ_8 even though they are technically outputs not inputs

Nils Schöneberg nils.science@gmail.com

Shooting

τ_{reio} (mini-shooting in th.c)
σ_8
$100\ \theta_s$
(f_{ede} , ...)

Try out a *reasonable* value (with a pre-programmed guess)

Then iterate in *correct* direction

1D : More robust Ridder's method

2D or higher: Less robust but faster Newton's method

h	$100\ \theta_s$
0.70000000000	1.0522492086422521
0.65000000000	1.0270326366580724
0.68215616173	1.0437999980620178
0.68110138476	1.0432819283581667
0.68103637942	1.0432499363679562
0.68103650871	1.0432499921072458

← Precision reached

Allows us to use parameters like $100\ \theta_s$ or σ_8 even though they are technically outputs not inputs

Nils Schöneberg nils.science@gmail.com

Shooting

τ_{reio} (mini-shooting in th.c)
σ_8
$100\ \theta_s$
(f_{ede} , ...)

Try out a *reasonable* value (with a pre-programmed guess)

Then iterate in *correct* direction

1D : More robust Ridder's method

2D or higher: Less robust but faster Newton's method

h	$100\ \theta_s$
0.70000000000	1.0522492086422521
0.65000000000	1.0270326366580724
0.68215616173	1.0437999980620178
0.68110138476	1.0432819283581667
0.68103637942	1.0432499363679562
0.68103650871	1.0432499921072458
0.68103652701	1.0432500079710365
...	...

← Precision reached

Allows us to use parameters like $100\ \theta_s$ or σ_8 even though they are technically outputs not inputs

Nils Schöneberg nils.science@gmail.com

Shooting

τ_{reio} (mini-shooting in th.c)
σ_8
$100\ \theta_s$
(f_{ede} , ...)

Try out a *reasonable* value (with a pre-programmed guess)

Then iterate in *correct* direction

1D : More robust Ridder's method

2D or higher: Less robust but faster Newton's method

Notebook!

h	$100\ \theta_s$
0.70000000000	1.0522492086422521
0.65000000000	1.0270326366580724
0.68215616173	1.0437999980620178
0.68110138476	1.0432819283581667
0.68103637942	1.0432499363679562
0.68103650871	1.0432499921072458
0.68103652701	1.0432500079710365
...	...

← Precision reached

Allows us to use parameters like $100\ \theta_s$ or σ_8 even though they are technically outputs not inputs

Budget Equation

- To avoid over-constraining input, one of `Omega_Lambda` , `Omega_fld` , `Omega_scf` must be left free

Budget Equation

- To avoid over-constraining input, one of `Omega_Lambda` , `Omega_fld` , `Omega_scf` must be left free

- Then the first unconstrained one is set by

$$\sum \Omega_x = 1 + \Omega_k$$

Budget Equation

- To avoid over-constraining input, one of `Omega_Lambda` , `Omega_fld` , `Omega_scf` must be left free

- Then the first unconstrained one is set by

$$\sum \Omega_x = 1 + \Omega_k$$

- Example: Dark Energy with equation of state `Omega_Lambda=0.2` (`Omega_scf=0`)

Background Module

Nils Schöneberg nils.science@gmail.com

Background Module

Einstein Boltzmann Solver:

Einstein Equation:

$$H^2 = \frac{8\pi G}{3} \sum_{\text{species } i} \rho_i$$

Boltzmann Equations:

$$\frac{d\rho_i}{d\ln a} + 3(\rho_i + P_i) = 0$$

Nils Schöneberg nils.science@gmail.com

Background Module

Computation of the background quantities like $\rho(z)$, $H(z)$

AND derivation of **other useful quantities** for cosmology like

$$D_A(z) \ , \ r_s \ , \ t_{\text{age}} \ \ldots$$

Nils Schöneberg nils.science@gmail.com

Background Module

Computation of the background quantities like $\rho(z)$, $H(z)$

AND derivation of **other useful quantities** for cosmology like

$$D_A(z) \ , \ r_s \ , \ t_{\text{age}} \ \ldots$$

Quantities known analytically for each time-step:

$$\rho_b(z) = \rho_{b,0}(1+z)^3 \qquad \rho_r(z) = \rho_{r,0}(1+z)^4$$

Nils Schöneberg nils.science@gmail.com

Background Module

Computation of the background quantities like $\rho(z)$, $H(z)$

AND derivation of **other useful quantities** for cosmology like

$$D_A(z) \ , \ r_s \ , \ t_{\rm age} \ \dots$$

Quantities known analytically for each time-step:

$$\rho_b(z) = \rho_{b,0}(1+z)^3 \qquad \rho_r(z) = \rho_{r,0}(1+z)^4$$

Quantities that have to be evolved:

$$\frac{dt}{d\ln 1+z} = -\frac{1}{H(z)} \qquad \frac{d\ln \rho_{\rm fld}(z)}{d\ln 1+z} = 3(1+w_{\rm fld}(z))$$

$$\ddot{\phi} + 3H\dot{\phi} + V'(\phi) = 0$$

Nils Schöneberg nils.science@gmail.com

Thermodynamics module

Integration of the thermodynamic variables like temperature, ionization rate, and derivation of interaction rates

Thermodynamics module

Integration of the thermodynamic variables like temperature, ionization rate, and derivation of interaction rates

This is actually complicated, and there ∃ some codes to solve this

CLASS uses HYREC-2 (default) or RECFAST for recombination

Thermodynamics module

Integration of the thermodynamic variables like temperature, ionization rate, and derivation of interaction rates

This is actually complicated, and there ∃ some codes to solve this

CLASS uses HYREC-2 (default) or RECFAST for recombination

For reionization, simple tanh prescription like in CAMB (since mostly only τ_{reio} matters) → Other options available

Nils Schöneberg nils.science@gmail.com

Thermodynamics module

Integration of the thermodynamic variables like temperature, ionization rate, and derivation of interaction rates

This is actually complicated, and there ∃ some codes to solve this

CLASS uses HYREC-2 (default) or RECFAST for recombination

For reionization, simple tanh prescription like in CAMB (since mostly only τ_{reio} matters) → Other options available

$$x_e(z) \rightarrow \kappa'(z) \rightarrow g(z)$$

Thermodynamics module

Integration of the thermodynamic variables like temperature, ionization rate, and derivation of interaction rates

This is actually complicated, and there ∃ some codes to solve this

CLASS uses HYREC-2 (default) or RECFAST for recombination

For reionization, simple tanh prescription like in CAMB (since mostly only $\tau_{\rm reio}$ matters) → Other options available

$$x_e(z) \rightarrow \kappa'(z) \rightarrow g(z)$$

Saha recombination

- We know Maxwell-Boltzmann abundances of non-relativistic particles

$$n(\mu, T) \approx g e^{\mu/T} \left(\frac{mT}{2\pi}\right)^{3/2} e^{-m/T}$$

Saha recombination

- We know Maxwell-Boltzmann abundances of non-relativistic particles

$$n(\mu, T) \approx g e^{\mu/T} \left(\frac{mT}{2\pi}\right)^{3/2} e^{-m/T}$$

- Due to interactions, chemical potential must balance:

$$\mu_{\text{ionized}} + \mu_e = \mu_{\text{rec}}$$

Nils Schöneberg nils.science@gmail.com

Saha recombination

- We know Maxwell-Boltzmann abundances of non-relativistic particles

$$n(\mu, T) \approx g e^{\mu/T} \left(\frac{mT}{2\pi}\right)^{3/2} e^{-m/T}$$

- Due to interactions, chemical potential must balance:

$$\mu_{\text{ionized}} + \mu_e = \mu_{\text{rec}}$$

$$\frac{n_e n_{\text{ionized}}}{n_{\text{rec}}} \approx \left(\frac{m_e T}{2\pi}\right)^{3/2} e^{-E_{\text{bind}}/T} \times \underbrace{\left[e^{\mu_{\text{ionized}}+\mu_e-\mu_{\text{rec}}} \left(\frac{g_e g_{\text{ionized}}}{g_{\text{rec}}}\right)\left(\frac{m_{\text{ionized}}}{m_{\text{rec}}}\right)^{3/2}\right]}_{\approx 1}$$

Nils Schöneberg nils.science@gmail.com

Saha recombination

- We know Maxwell-Boltzmann abundances of non-relativistic particles

$$n(\mu, T) \approx g e^{\mu/T} \left(\frac{mT}{2\pi}\right)^{3/2} e^{-m/T}$$

- Due to interactions, chemical potential must balance:

$$\mu_{\text{ionized}} + \mu_e = \mu_{\text{rec}}$$

$$\frac{n_e n_{\text{ionized}}}{n_{\text{rec}}} \approx \left(\frac{m_e T}{2\pi}\right)^{3/2} e^{-E_{\text{bind}}/T} \times \underbrace{\left[e^{\mu_{\text{ionized}}+\mu_e-\mu_{\text{rec}}} \left(\frac{g_e g_{\text{ionized}}}{g_{\text{rec}}}\right)\left(\frac{m_{\text{ionized}}}{m_{\text{rec}}}\right)^{3/2}\right]}_{\approx 1}$$

$$\frac{x_e^2}{1-x_e} \approx \left(\frac{1.1 \cdot 10^{-10}}{n_{\text{H},0}/T_{\text{cmb},0}^3}\right)\left(\frac{\text{eV}}{T}\right)^{3/2} \exp(39.9 - 13.6\frac{\text{eV}}{T})$$

Nils Schöneberg nils.science@gmail.com

Saha recombination

- We know Maxwell-Boltzmann abundances of non-relativistic particles

$$n(\mu, T) \approx g e^{\mu/T} \left(\frac{mT}{2\pi}\right)^{3/2} e^{-m/T}$$

- Due to interactions, chemical potential must balance:

$$\mu_{\text{ionized}} + \mu_e = \mu_{\text{rec}}$$

$$\frac{n_e n_{\text{ionized}}}{n_{\text{rec}}} \approx \left(\frac{m_e T}{2\pi}\right)^{3/2} e^{-E_{\text{bind}}/T} \times \underbrace{\left[e^{\mu_{\text{ionized}}+\mu_e-\mu_{\text{rec}}} \left(\frac{g_e g_{\text{ionized}}}{g_{\text{rec}}}\right)\left(\frac{m_{\text{ionized}}}{m_{\text{rec}}}\right)^{3/2}\right]}_{\approx 1}$$

$$\frac{x_e^2}{1-x_e} \approx \left(\frac{1.1 \cdot 10^{-10}}{n_{\text{H},0}/T_{\text{cmb},0}^3}\right)\left(\frac{\text{eV}}{T}\right)^{3/2} \exp(39.9 - 13.6\frac{\text{eV}}{T})$$

$$T \approx \frac{13.6\text{eV}}{39.9} \approx 0.34\text{eV} \rightarrow z \approx 1400$$

Saha recombination

The effective multi-level atom is the basis for recombination codes.

Saha recombination

The effective multi-level atom is the basis for recombination codes.

We only use: 1s 2s 2p ~~3s 3p 3d ...~~ ionized

Saha recombination

The effective multi-level atom is the basis for recombination codes.

We only use: 1s 2s 2p ~~3s 3p 3d ...~~ ionized

- Reason: Intermediate transitions (4p→3s) or (3s→2p) are instant.

  Direct transition 2s → 1s is forbidden, and 2p → 1s is immediately reversed by 1s → 2p (optically thick medium)

Saha recombination

The effective multi-level atom is the basis for recombination codes.

We only use: 1s 2s 2p ~~3s 3p 3d ...~~ ionized

- Reason: Intermediate transitions (4p→3s) or (3s→2p) are instant.

  Direct transition 2s → 1s is forbidden, and 2p → 1s is immediately reversed by 1s → 2p (optically thick medium)

- Slow processes:
- 2p → 1s with subsequent redshifting of photon

  or 2s → 1s with two-photon decay

Thermodynamical choices

- Recombination code : RECFAST / HYREC-2

Thermodynamical choices

- Recombination code : RECFAST / HYREC-2
- Reionization parameterization:
  - `Camb`
  - `half_tanh`
  - `bins_tanh`
  - `inter`

$$\frac{1}{2}\left[1+\tanh\left(\frac{((1+z_r)^{3/2}-(1+z)^{3/2})}{\Delta_z\cdot(3/2\cdot(1+z_r)^{1/2})}\right)\right]\cdot\Delta x$$

Thermodynamical choices

- Recombination code : RECFAST / HYREC-2
- Reionization parameterization:
  - `Camb`
  - `half_tanh`
  - `bins_tanh`
  - `inter`

$$\frac{1}{2}\left[1+\tanh\left(\frac{((1+z_r)^{3/2}-(1+z)^{3/2})}{\Delta_z\cdot(3/2\cdot(1+z_r)^{1/2})}\right)\right]\cdot\Delta x$$

- Energy injection (PBH accretion/evaporation , DM decay/annihilation)

Thermodynamical choices

- Recombination code : RECFAST / HYREC-2
- Reionization parameterization:
  - `Camb`
  - `half_tanh`
  - `bins_tanh`
  - `inter`

$$\frac{1}{2}\left[1+\tanh\left(\frac{((1+z_r)^{3/2}-(1+z)^{3/2})}{\Delta_z\cdot(3/2\cdot(1+z_r)^{1/2})}\right)\right]\cdot\Delta x$$

- Energy injection (PBH accretion/evaporation , DM decay/annihilation)
- Spectral distortions (y, μ, PCA)

Thermodynamical choices

- Recombination code : RECFAST / HYREC-2
- Reionization parameterization:
  - `Camb`
  - `half_tanh`
  - `bins_tanh`
  - `inter`

$$\frac{1}{2}\left[1+\tanh\left(\frac{((1+z_r)^{3/2}-(1+z)^{3/2})}{\Delta_z\cdot(3/2\cdot(1+z_r)^{1/2})}\right)\right]\cdot\Delta x$$

- Energy injection (PBH accretion/evaporation , DM decay/annihilation)
- Spectral distortions (y, μ, PCA)
- Helium abundance: `YHe=0.25/BBN` `external/bbn/sBBN_2017.dat`

Nils Schöneberg nils.science@gmail.com

Perturbation Module

Nils Schöneberg nils.science@gmail.com

Perturbation Module

Einstein Boltzmann Solver:

Einstein Equations:

$$k^2\phi + 3\mathcal{H}(\phi' + \mathcal{H}\psi) = -4\pi G a^2 \delta\rho$$

Boltzmann Equations:

$$\delta' + (1+w)\theta = 3(1+w)\phi' - 3\mathcal{H}(c_s^2 - w)\delta$$

Nils Schöneberg nils.science@gmail.com

Perturbation Module

Einstein Boltzmann Solver:

Einstein Equations:

$$k^2\phi + 3\mathcal{H}(\phi' + \mathcal{H}\psi) = -4\pi G a^2 \delta\rho$$

Boltzmann Equations:

$$\delta' + (1+w)\theta = 3(1+w)\phi' - 3\mathcal{H}(c_s^2 - w)\delta$$

$$\theta' = -\mathcal{H}(1 - 3c_a^2)\theta + \frac{c_s^2}{1+w}k^2\delta - k^2\sigma + k^2\psi$$

Nils Schöneberg nils.science@gmail.com

Perturbation Module

Einstein Boltzmann Solver:

Einstein Equations:

$$k^2\phi + 3\mathcal{H}(\phi' + \mathcal{H}\psi) = -4\pi G a^2 \delta\rho$$

Boltzmann Equations:

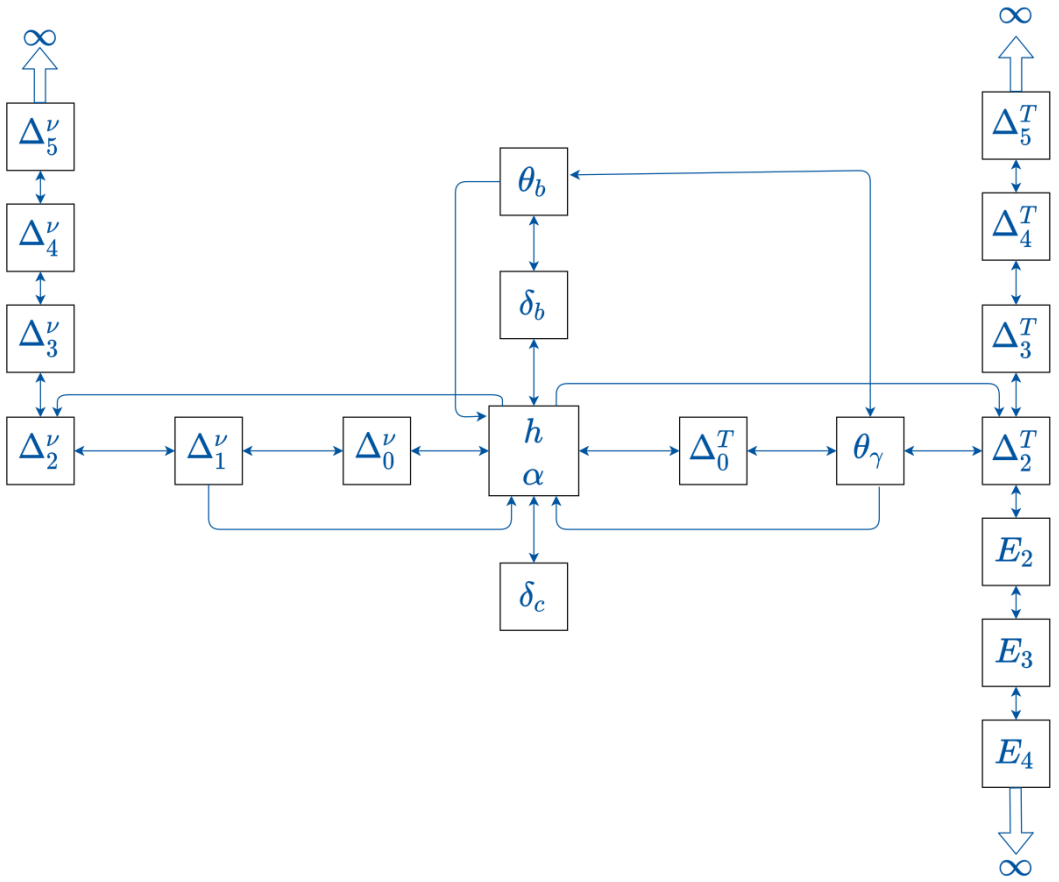
$$\delta' + (1+w)\theta = 3(1+w)\phi' - 3\mathcal{H}(c_s^2 - w)\delta$$

$$\theta' = -\mathcal{H}(1 - 3c_a^2)\theta + \frac{c_s^2}{1+w}k^2\delta - k^2\sigma + k^2\psi$$

Ma & Bertschinger astro-ph/9506072

Nils Schöneberg nils.science@gmail.com

Perturbations module



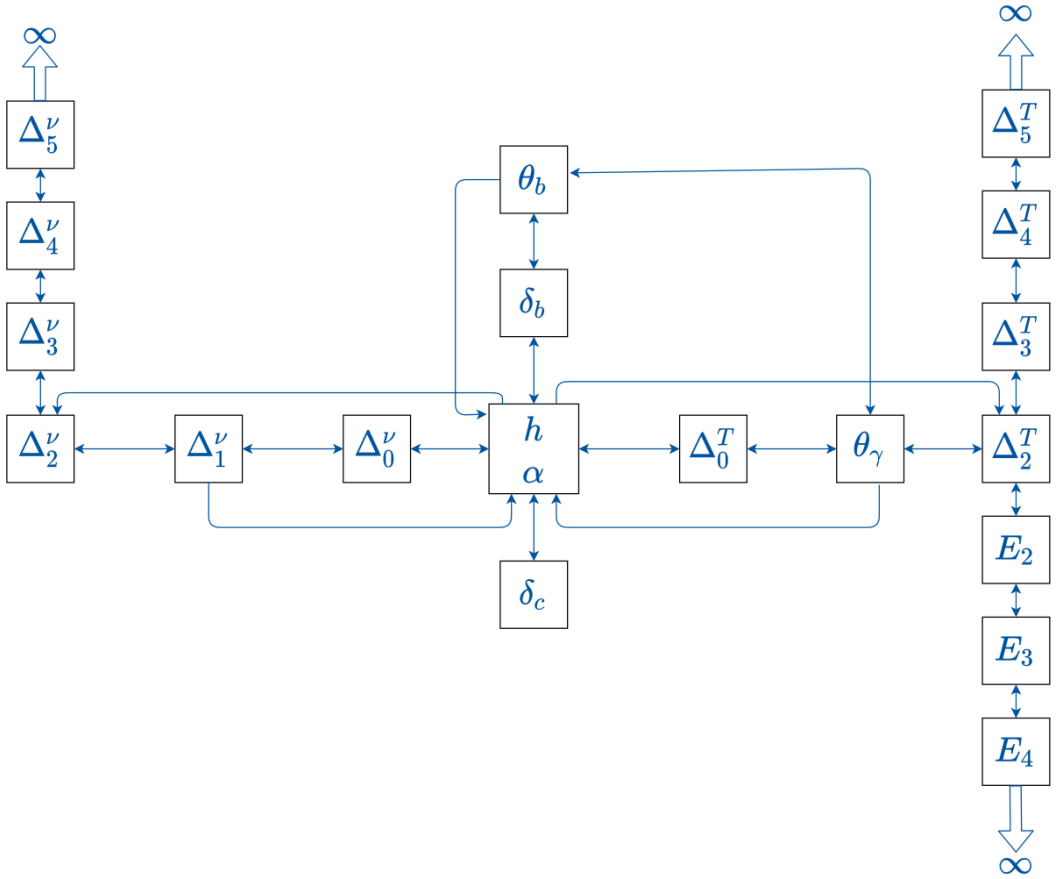
Ma & Bertschinger astro-ph/9506072

Nils Schöneberg nils.science@gmail.com

Perturbations module

Default:
CDM-**synchronous gauge**

Alternative:
Conformal Newtonian Gauge



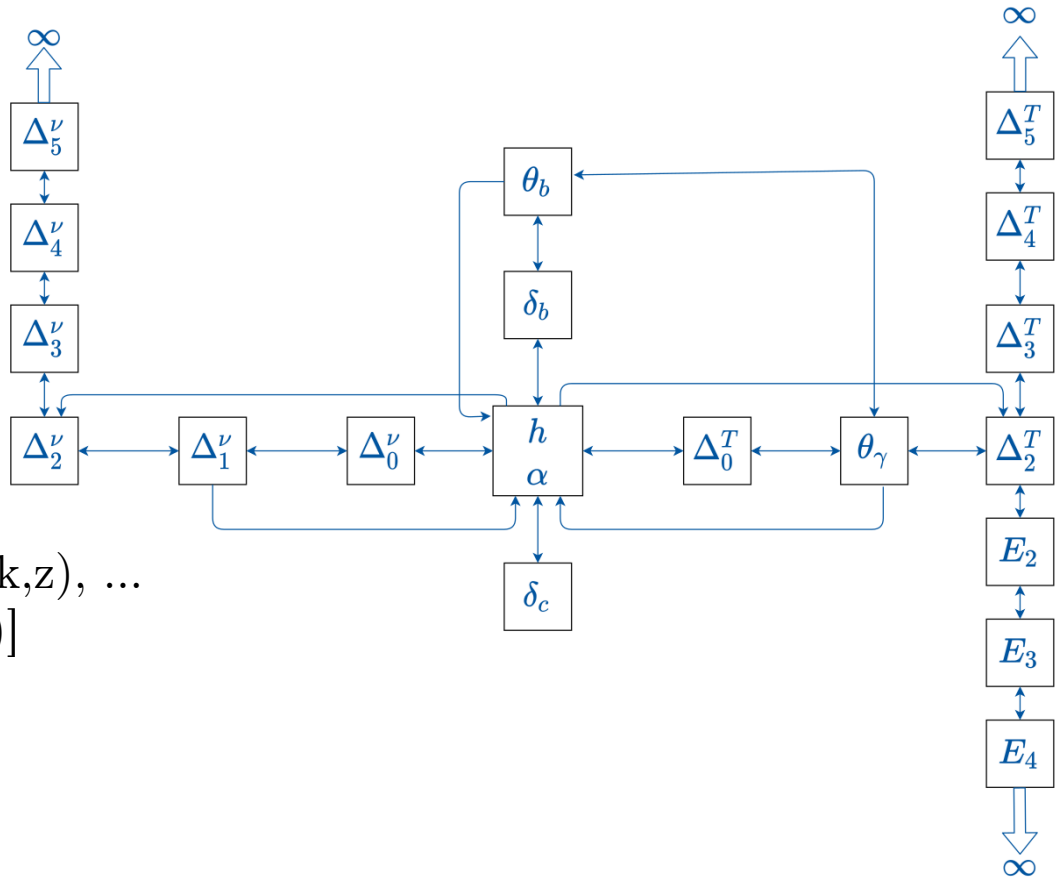
Ma & Bertschinger astro-ph/9506072

Nils Schöneberg nils.science@gmail.com

Perturbations module

Default:
CDM-**synchronous gauge**

Alternative:
Conformal Newtonian Gauge



Solve the hierarchy of perturbations
→ Get source functions of CMB, P(k,z), ...
[Non-trivial combinations of δ, , g(z)]

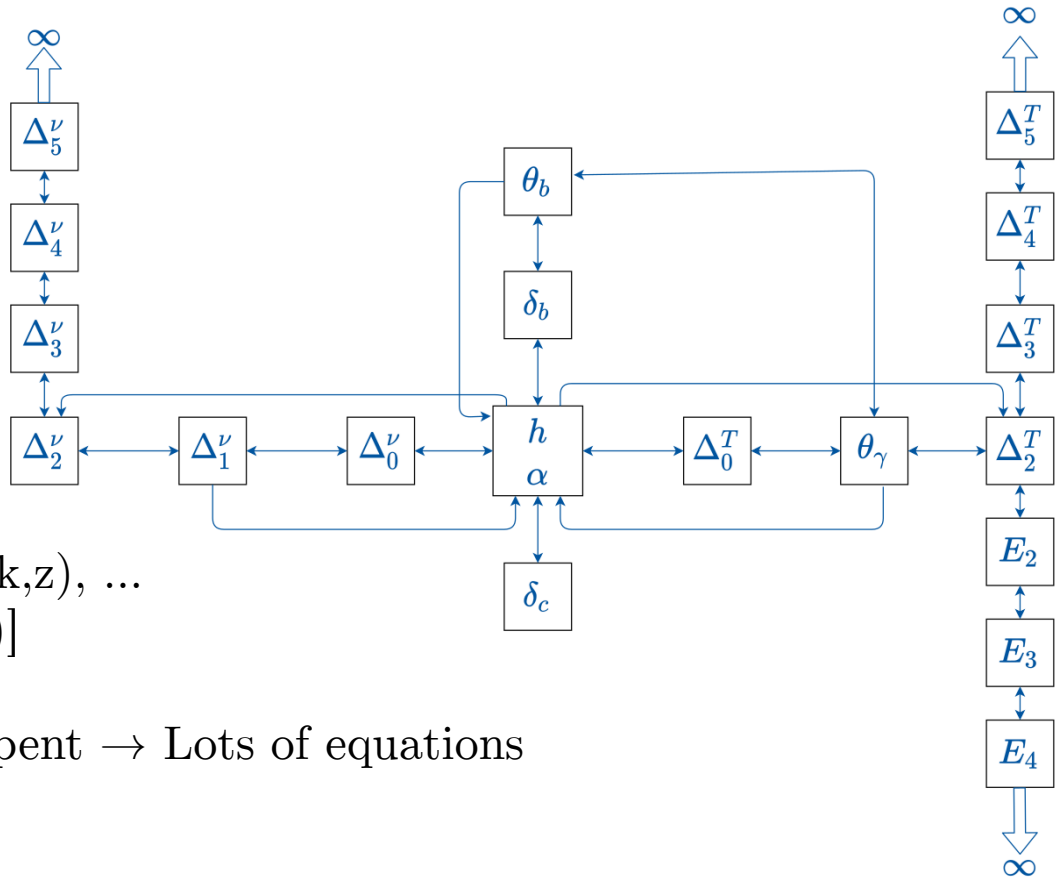
Ma & Bertschinger astro-ph/9506072

Nils Schöneberg nils.science@gmail.com

Perturbations module

Default:
CDM-**synchronous gauge**

Alternative:
Conformal Newtonian Gauge



Solve the hierarchy of perturbations
→ Get source functions of CMB, P(k,z), ...
[Non-trivial combinations of δ, , g(z)]

This is usually where most time is spent → Lots of equations
(minimal set is shown)

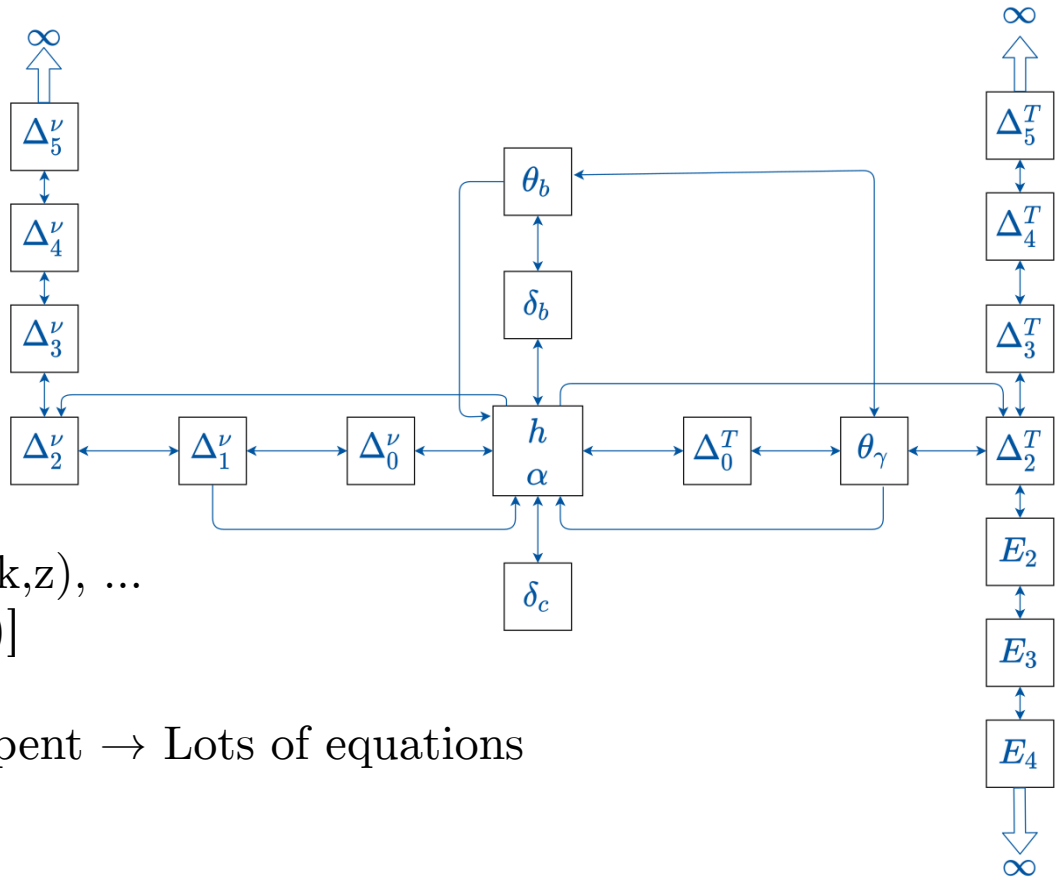
Ma & Bertschinger astro-ph/9506072

Nils Schöneberg nils.science@gmail.com

Perturbations module

Default:
CDM-**synchronous gauge**

Alternative:
Conformal Newtonian Gauge



Solve the hierarchy of perturbations
→ Get source functions of CMB, P(k,z), ...
[Non-trivial combinations of δ, , g(z)]

This is usually where most time is spent → Lots of equations
(minimal set is shown)

Equations relative to $\mathcal{R} = 1$ initial condition

Ma & Bertschinger astro-ph/9506072

Perturbations module

Approximations:
TCA $\rightarrow$ Simplify photons and baryons due to infinite interaction

Perturbations module

Approximations:
TCA $\rightarrow$ Simplify photons and baryons due to infinite interaction
UFA $\rightarrow$ Simplify (massless) neutrinos and photons by treating as a fluid ($\sigma = f(\delta, \theta)$)

Perturbations module

Approximations:
TCA $\rightarrow$ Simplify photons and baryons due to infinite interaction
UFA $\rightarrow$ Simplify (massless) neutrinos and photons by treating as a fluid ($\sigma = f(\delta, \theta)$)
RSA $\rightarrow$ Simplify (massless) neutrinos and photons by making them tracers of potential only

Perturbations module

Approximations:
TCA $\rightarrow$ Simplify photons and baryons due to infinite interaction
UFA $\rightarrow$ Simplify (massless) neutrinos and photons by treating as a fluid ($\sigma = f(\delta, \theta)$)
RSA $\rightarrow$ Simplify (massless) neutrinos and photons by making them tracers of potential only
NCDMFA $\rightarrow$ Simplify massive neutrinos by treating them as a fluid

Nils Schöneberg nils.science@gmail.com

Perturbations module

Approximations:
TCA $\rightarrow$ Simplify photons and baryons due to infinite interaction
UFA $\rightarrow$ Simplify (massless) neutrinos and photons by treating as a fluid ($\sigma = f(\delta, \theta)$)
RSA $\rightarrow$ Simplify (massless) neutrinos and photons by making them tracers of potential only
NCDMFA $\rightarrow$ Simplify massive neutrinos by treating them as a fluid

Stiff ODE solver (ndf15)
Implicit using Newton method, Jacobian recycling
On-the-fly adaptive stepsize

Nils Schöneberg nils.science@gmail.com

Primordial module

Primordial module

- Either: Analytical power spectrum $P_{\mathcal{R}}(k)$

Primordial module

- Either: Analytical power spectrum $P_{\mathcal{R}}(k)$
- Or: Solve inflation with variety of modes
  - `inflation_V` : using a potential Taylor expansion

Primordial module

- Either: Analytical power spectrum $P_{\mathcal{R}}(k)$
- Or: Solve inflation with variety of modes
  - `inflation_V` : using a potential Taylor expansion
  - `inflation_V_end` : using a potential Taylor expansion, but only in observable regime

Primordial module

- Either: Analytical power spectrum $P_{\mathcal{R}}(k)$
- Or: Solve inflation with variety of modes
  - `inflation_V` : using a potential Taylor expansion
  - `inflation_V_end` : using a potential Taylor expansion, but only in observable regime
  - `inflation_H` : using Taylor expansion of Hubble parameter in ϕ^n

Primordial stuff

- CLASS uses an inflation solver
- Taylor expansion in V(ϕ) / H(ϕ)
- Automatic determination of observable window
- Impose inflation duration

Nils Schöneberg nils.science@gmail.com

Primordial stuff

- CLASS uses an inflation solver
- Taylor expansion in V(ϕ) / H(ϕ)
- Automatic determination of observable window
- Impose inflation duration
- `Pk_ini_type = analytic_Pk`

$$P_{\mathcal{R}}(k) = A_s \left(\frac{k}{k_{\rm piv}}\right)^{n_s - 1 + \alpha_s \ln k/k_{\rm piv}}$$

- `inflation_V, inflation_H, inflationV_end, two_scales`

  → `V_0…V_4 , R_0…R_4, PSR_0…PSR_4, H_0…H_4`

- Read spectrum from file (`external_Pk`)

Nils Schöneberg nils.science@gmail.com

Fourier Module

Fourier Module

- Compute linear power spectrum of matter, cb

Nils Schöneberg nils.science@gmail.com

Fourier Module

- Compute linear power spectrum of matter, cb
- Compute σ_8

Fourier Module

- Compute linear power spectrum of matter, cb
- Compute σ_8
- Compute non-linear power spectrum using
  - Halofit
  - HMcode2016

Fourier Module

- Compute linear power spectrum of matter, cb
- Compute σ_8
- Compute non-linear power spectrum using
  - Halofit
  - HMcode2016
  - HMcode2020
  - (soon) perturbation theory

Model Inputs

Example: HMcode

$$\Delta^2_{\rm NL}(k,z) = \Big(\Delta^2_{\rm 1-halo}{}^{\alpha} + \Delta^2_{\rm 2-halo}{}^{\alpha}\Big)^{1/\alpha}$$

Nils Schöneberg nils.science@gmail.com

Model Inputs

Example: HMcode

$$\Delta^2_{\rm NL}(k,z) = \left(\Delta^2_{1-{\rm halo}}{}^{\alpha} + \Delta^2_{2-{\rm halo}}{}^{\alpha}\right)^{1/\alpha}$$

$$P_{1-{\rm halo}}(k,z) = \int n(\nu \cdot k^{\eta}) g(\nu) dm \qquad \nu = \delta_{\rm crit}/\sigma(R(m,z))$$

Nils Schöneberg nils.science@gmail.com

Model Inputs

Example: HMcode

$$\Delta^2_{\rm NL}(k,z) = \left(\Delta^2_{1-{\rm halo}}{}^{\alpha} + \Delta^2_{2-{\rm halo}}{}^{\alpha}\right)^{1/\alpha}$$

$$P_{1-{\rm halo}}(k,z) = \int n(\nu \cdot k^{\eta}) g(\nu) dm \qquad \nu = \delta_{\rm crit}/\sigma(R(m,z))$$

$$P_{2-{\rm halo}} = P_{\rm dewiggle}\left[1 - f_{\rm damp}\frac{\left(\frac{k}{k_{\rm dam}}\right)^{\alpha_{\rm damp}}}{1+\left(\frac{k}{k_{\rm dam}}\right)^{\alpha_{\rm damp}}}\right]$$

Transfer Module

- Convert from Fourier space (k, t) to harmonic space
- $e^{i\vec{k}\cdot\vec{r}} = \sum (2\ell+1) i^{\ell} j_{\ell}(kr) P_{\ell}(\cos(\vartheta))$

Nils Schöneberg nils.science@gmail.com

Transfer Module

- Convert from Fourier space (k, t) to harmonic space
- $e^{i\vec{k}\cdot\vec{r}} = \sum (2\ell+1) i^\ell j_\ell(kr) P_\ell(\cos(\vartheta))$
- This is the origin of line-of-sight integrals:

$$\Delta_\ell^x(k) = \int d\chi j_\ell(k\chi) S^x(k, \chi = \tau_0 - \tau)$$

Nils Schöneberg nils.science@gmail.com

Transfer Module

- Convert from Fourier space (k, t) to harmonic space
- $e^{i\vec{k}\cdot\vec{r}} = \sum (2\ell+1) i^\ell j_\ell(kr) P_\ell(\cos(\vartheta))$
- This is the origin of line-of-sight integrals:

$$\Delta_\ell^x(k) = \int d\chi j_\ell(k\chi) S^x(k, \chi = \tau_0 - \tau)$$

$$C_\ell^{x,y} = 4\pi \int d\ln k \Delta_\ell^x(k) \Delta_\ell^y(k)$$

Nils Schöneberg nils.science@gmail.com

Transfer Module

- Convert from Fourier space (k, t) to harmonic space
- $e^{i\vec{k}\cdot\vec{r}} = \sum (2\ell+1) i^\ell j_\ell(kr) P_\ell(\cos(\vartheta))$
- This is the origin of line-of-sight integrals:

$$\Delta_\ell^x(k) = \int d\chi j_\ell(k\chi) S^x(k, \chi = \tau_0 - \tau)$$

Harmonic module

$$C_\ell^{x,y} = 4\pi \int d\ln k \Delta_\ell^x(k) \Delta_\ell^y(k)$$

Lensing module

- In principle non-linear and very complicated

Lensing module

- In principle non-linear and very complicated
- Weak lensing, flat-sky limit, first order

$$\Theta^{\text{lensed}}(\vec{n}) = \Theta(\vec{n} + \vec{\alpha})$$

details: astro-ph/0502425

Nils Schöneberg nils.science@gmail.com

Lensing module

- In principle non-linear and very complicated
- Weak lensing, flat-sky limit, first order

$$\Theta^{\text{lensed}}(\vec{n}) = \Theta(\vec{n} + \vec{\alpha})$$

details: astro-ph/0502425

- $\boxed{\vec{\alpha} = \nabla\psi} \rightarrow \int \frac{d^2\vec{l}}{2\pi}(i\vec{l})\psi_{\vec{l}}e^{i\vec{l}\vec{n}}$

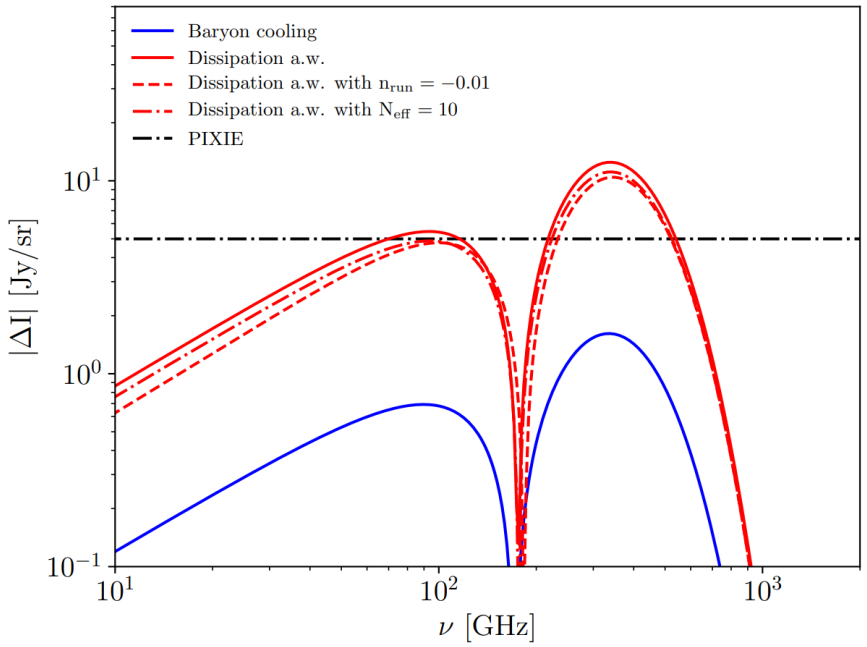
$$\langle \alpha_i(\vec{n})\alpha_j(\vec{n}')\rangle = \int \frac{d^2\vec{l}}{(2\pi)^2} l_i l_j C_\ell^\psi e^{i\vec{l}(\vec{n}-\vec{n}')}$$

$$C_\ell^{xy,\text{lensed}} = 2\pi\int_{-1}^{1}\xi^{xy}(\mu)d_{00}^\ell(\mu)d\mu \quad \xi^{xy}(\mu) \sim \int C_\ell^{xy}\sum_i\left(C_\ell^{GLa_n}J_{2n}(\ell r)\right) \quad C^{GL}(r) = \int\frac{dll^3}{2\pi}C_\ell^{\Psi\Psi}J_0(lr)$$

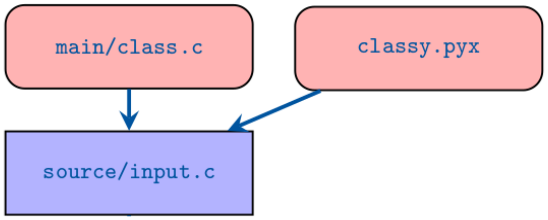
Nils Schöneberg nils.science@gmail.com

Distortions Module

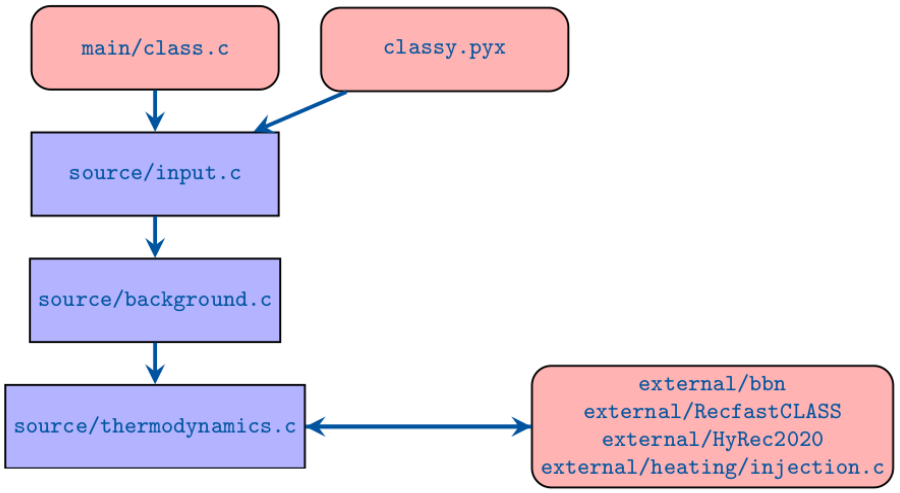
- Computes injected energy into photons
- Computes distortions y μ that could be measured in the CMB in the future



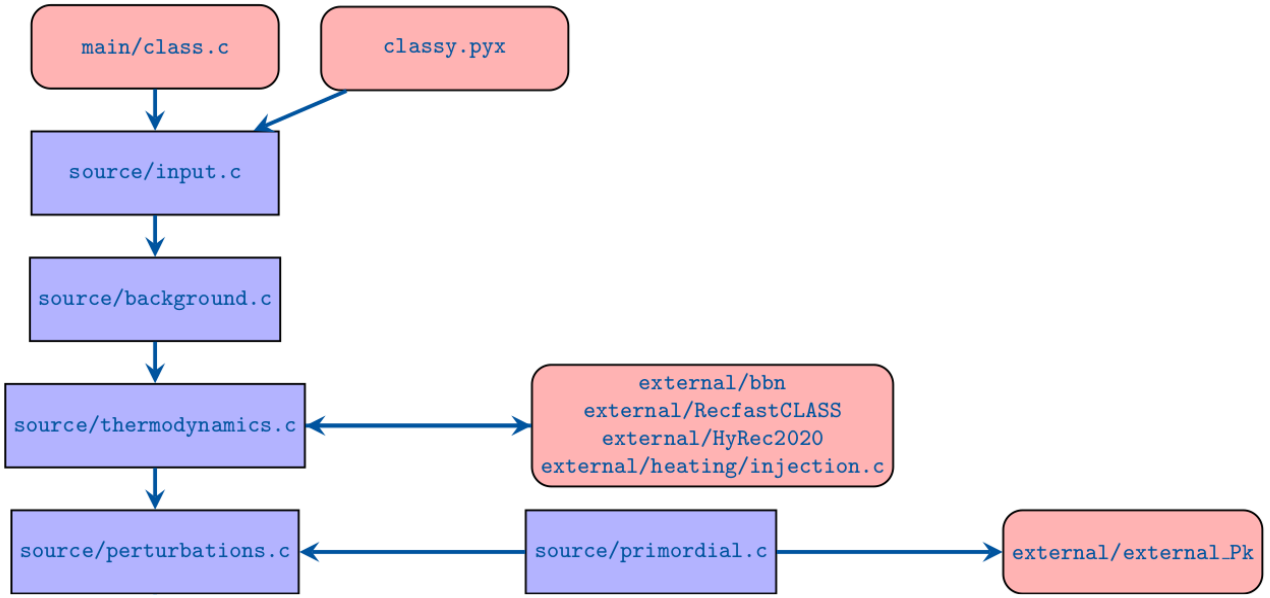
The complete structure



The complete structure

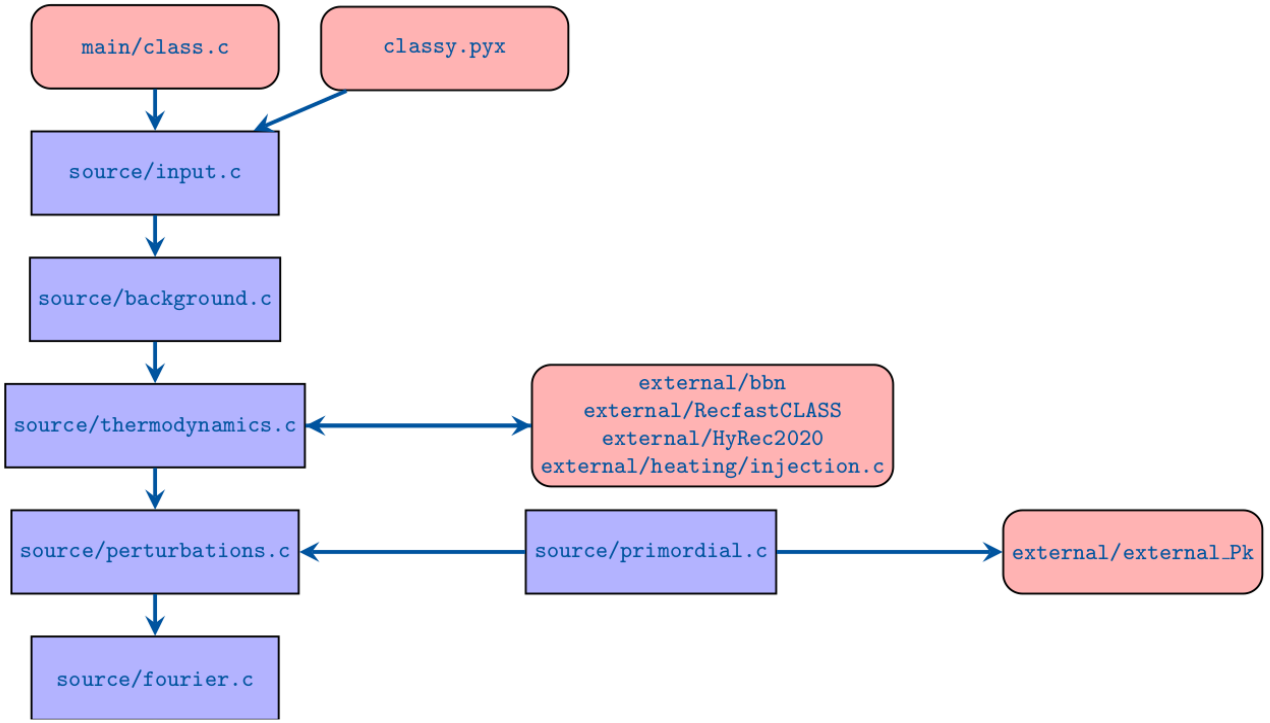


The complete structure



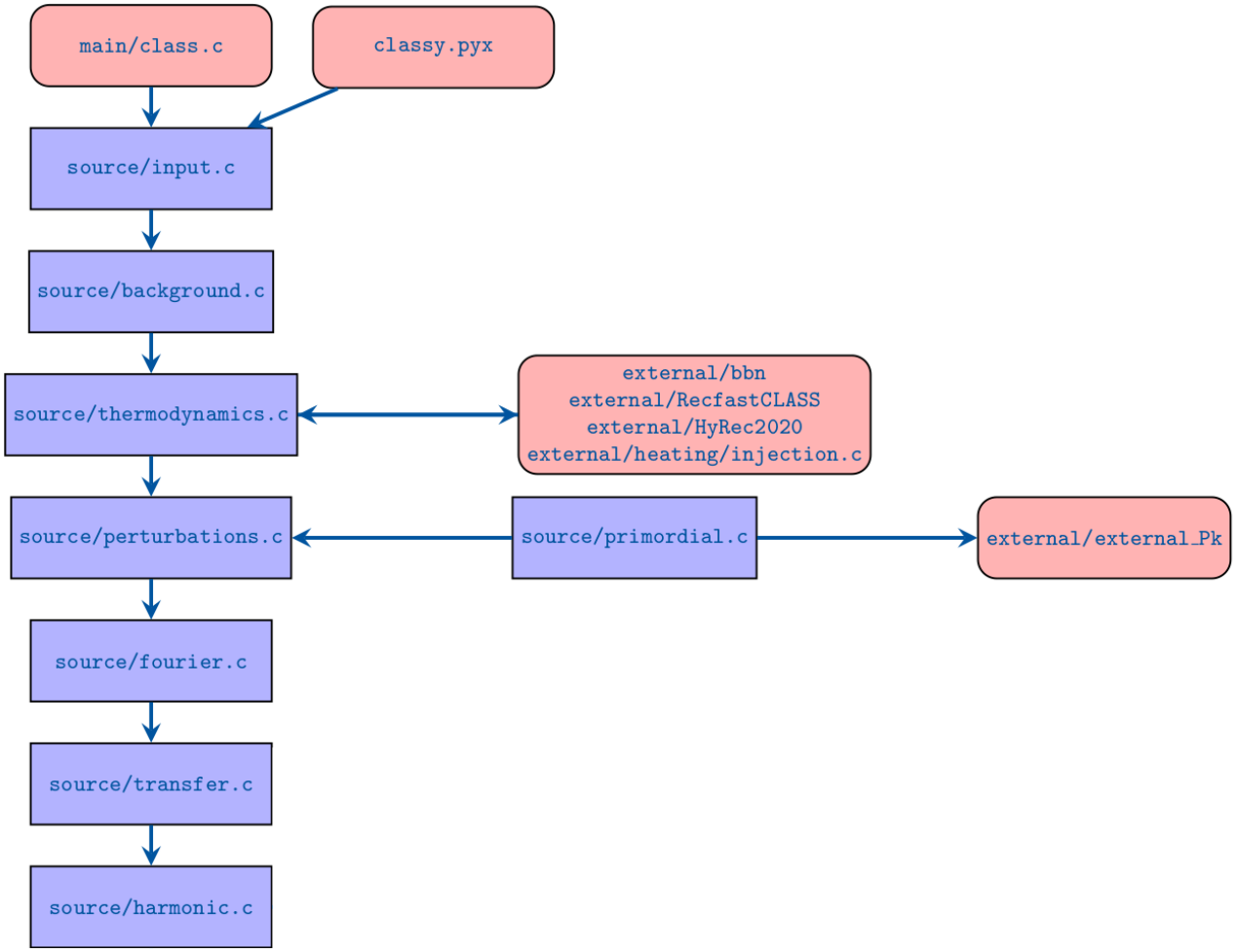
Nils Schöneberg nils.science@gmail.com

The complete structure



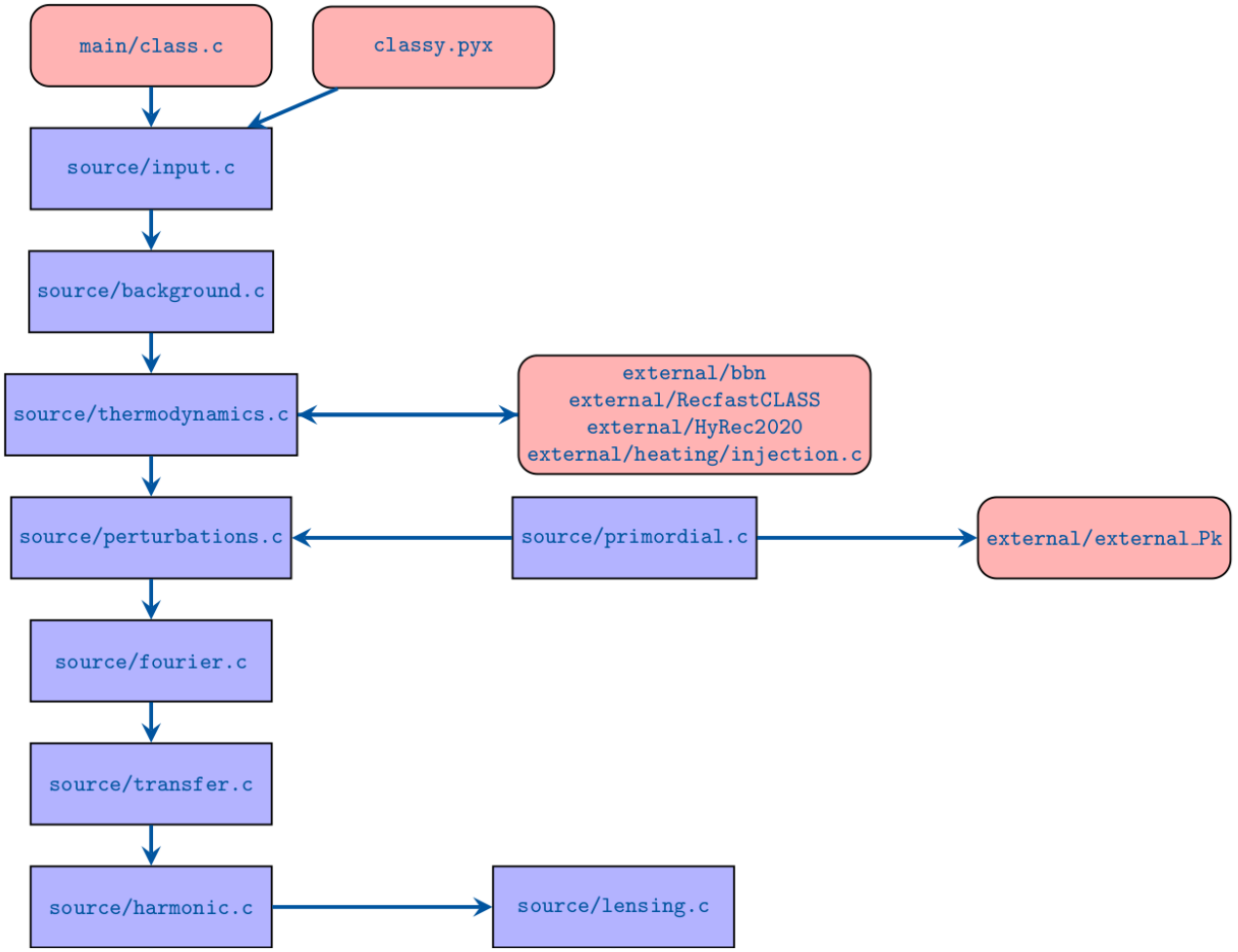
Nils Schöneberg nils.science@gmail.com

The complete structure



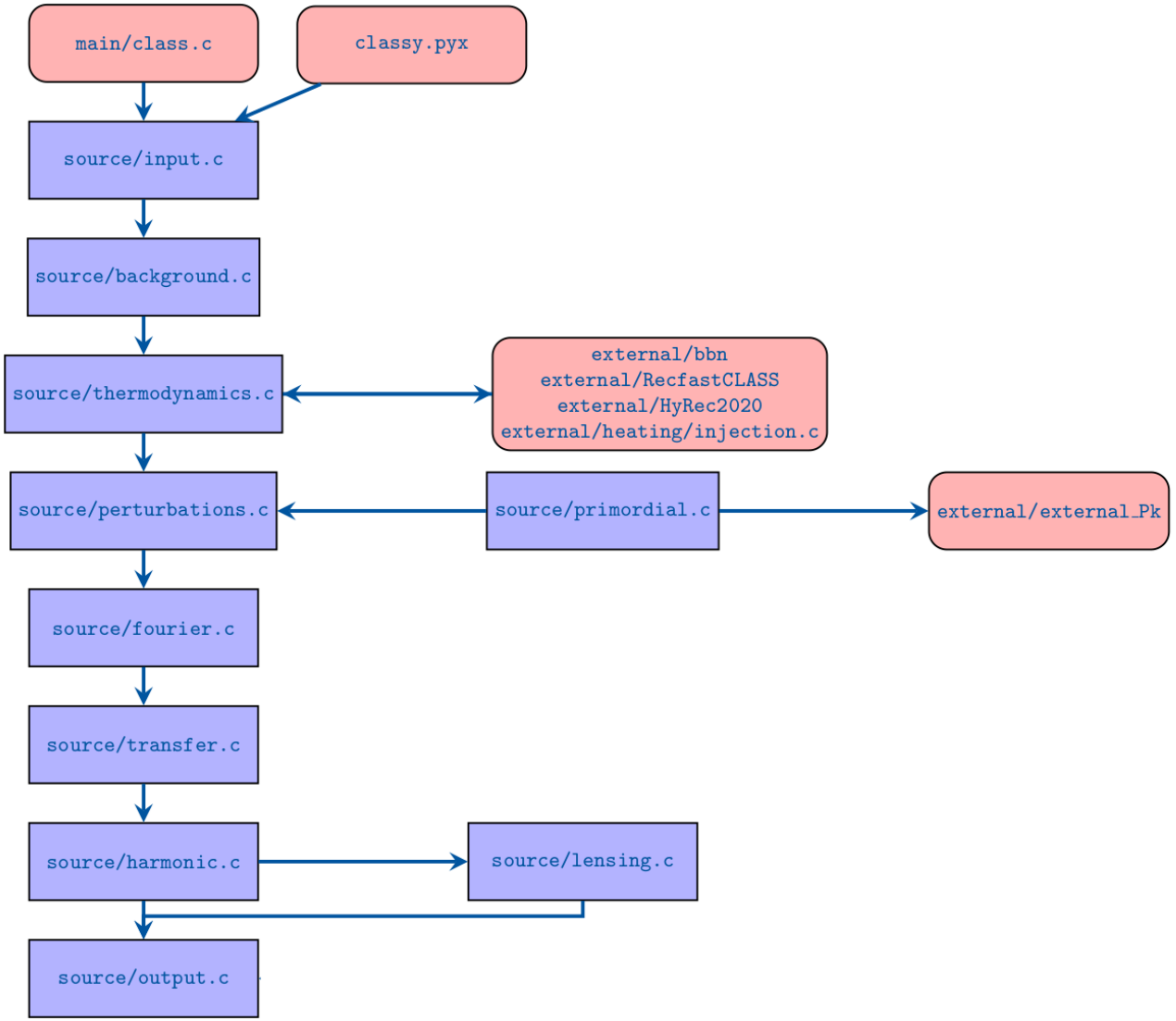
Nils Schöneberg nils.science@gmail.com

The complete structure



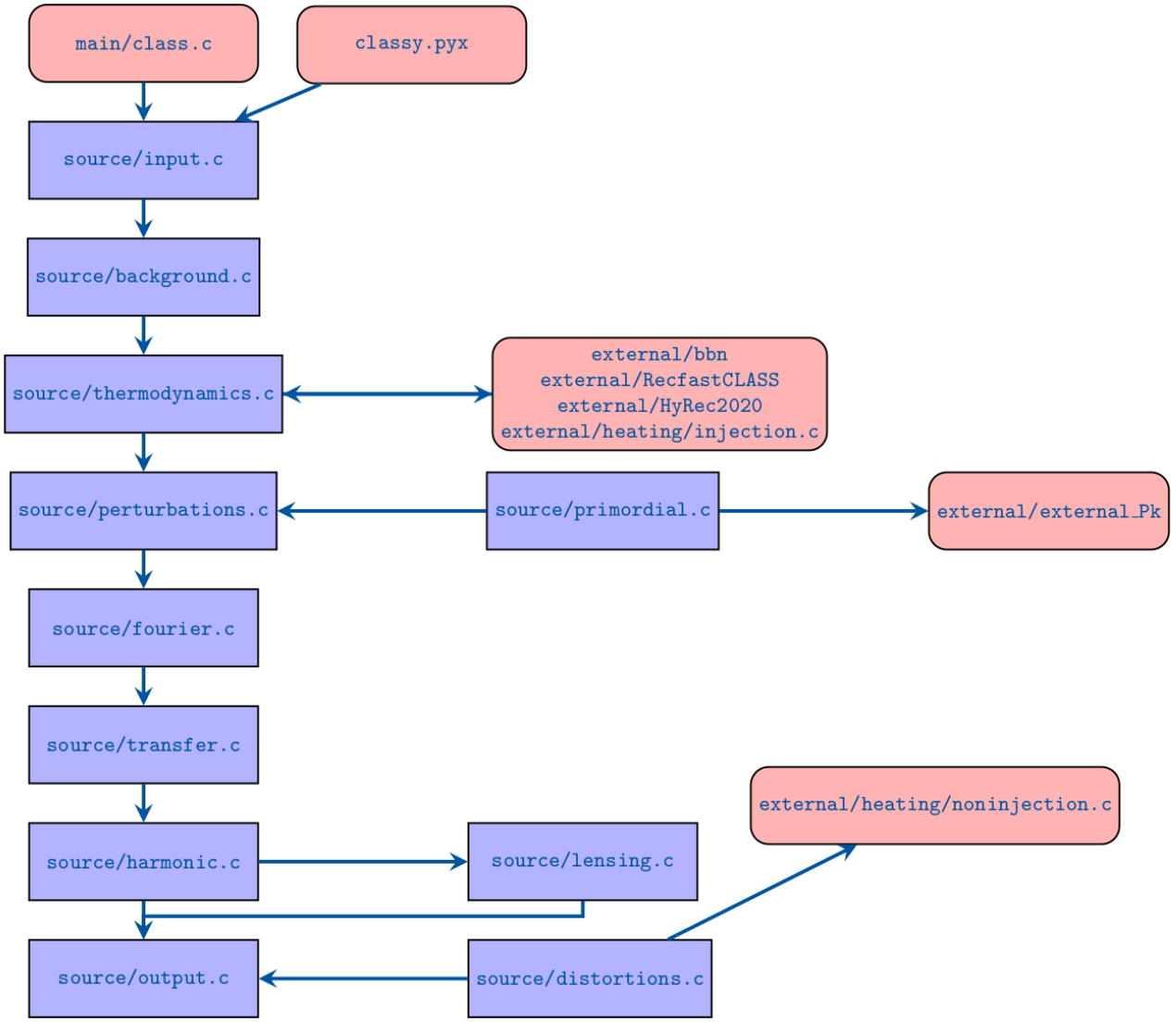
Nils Schöneberg nils.science@gmail.com

The complete structure



Nils Schöneberg nils.science@gmail.com

The complete structure



Careful

- Internal units are $\mathrm{Mpc}^{\mathrm{n}}$

Careful

- Internal units are $\mathrm{Mpc}^{\mathrm{n}}$

- Density units are even weirder:

$$\rho_i^{\mathrm{CLASS}} = \frac{8\pi G}{3}\rho_i^{\mathrm{physical}}$$

$$H^2 = \sum_{\mathrm{species}\, i} \rho_i^{\mathrm{CLASS}}$$

Implementing new models

Implementing new models

- Spoiler: There is no single answer, it will depend on your model, but...

Implementing new models

- Spoiler: There is no single answer, it will depend on your model, but...
- General guidelines:
  - Follow existing code conventions

Implementing new models

- Spoiler: There is no single answer, it will depend on your model, but...
- General guidelines:
  - Follow existing code conventions
  - Copy existing models and modify them

Implementing new models

- Spoiler: There is no single answer, it will depend on your model, but...
- General guidelines:
  - Follow existing code conventions
  - Copy existing models and modify them
  - Main modifications likely to be in
    - `input.c`, `background.c`, `thermodynamics.c`, `perturbations.c`

Any Questions?

nils.science@gmail.com

Nils Schöneberg nils.science@gmail.com

Nitty Gritty Coding Stuff

Structure	Pointer	Description
input.c	--	Convert input strings into C variables, shooting
background.c	pba	Homogeneous background evolution
thermodynamics.c	pth	Recombination, reionization, energy injection
Perturbations.c	ppt	Linear perturbation theory (EBS)
Primoridal.c	ppm	Inflation, primordial P(k)
fourier.c	pfo	Power spectrum, nonlinear treatment
transfer.c	ptr	Line-of-sight integral
Harmonic.c	pha	Two-point correlation functions
Lensing.c	ple	CMB lensing
Distortions.c	psd	Spectral distortions
Output.c	pop	Output formatting/files

Nitty Gritty Coding Stuff

- Input management
- `parser_read`: Name → value, flag (manual assignment)
- `class_read`: Name → value

Type	parser	class_read
double	parser_read_double	class_read_double
int	parser_read_int	class_read_int
boolean	---	class_read_flag
string	parser_read_string	class_read_string
double list	parser_read_list_of_doubles	class_read_list_of_doubles_or_default
int list	parser_read_list_of_integers	class_read_list_of_integers_or_default
string list	parser_read_list_of_strings	---

Nitty Gritty Coding Stuff

```
class_call(parser_read_double(pfc,"monkey",&param1,&flag1,errmsg),
           errmsg,
           Errmsg);

if (flag1 == _TRUE_){
  pba->monkey = param1;
}
```

→ Much simpler:

```
class_read_double("monkey",pba→monkey);
```

Nitty Gritty Coding Stuff

```
class_call(parser_read_double(pfc,"H0",&param1,&flag1,errmsg),
           errmsg,
           errmsg);
class_call(parser_read_double(pfc,"h",&param2,&flag2,errmsg),
           errmsg,
           errmsg);
/* Test */
class_test((flag1 == _TRUE_) && (flag2 == _TRUE_),
           errmsg,
           "You can only enter one of 'h' or 'H0'.");
/* Complete set of parameters */
if (flag1 == _TRUE_){
  pba->H0 = param1*1.e3/_c_;
  pba->h = param1/100.;
}
if (flag2 == _TRUE_){
  pba->H0 = param2*1.e5/_c_;
  pba->h = param2;
}
```

Indexing rules

- Always written as
  - `index_name_type`
  - `index_bg_rho`
  - `index_pt_delta_cdm`
  - `index_ct_tt`
- Here `name` is the shortcut for the specific table/array that is referenced to, and `type` is the specific entry column in the table

Indexing rules

```
/* initialize all indices */
  index_bg=0;
  class_define_index(pba->index_bg_rho_cdm,
                     pba->has_cdm,
                     index_bg,
                     1);
  class_define_index(pba->index_bg_rho_fld,
                     pba->has_fld,
                     index_bg,
                     1);
  ...
  pba->bg_size = index_bg;
```

Features

- Always use:

```
if (has_feature == _TRUE_)  {
    …
}
```

Error Management

- Always use:

```
class_call(function, from, to);
```

- Example:

```
class_call(background_at_z(pba,z),
                   pba→error_message,
                   ppt→error_message);
```

- Always `return _SUCCESS_;` at the end

Error Management

- Use `class_test` and `class_stop`:

```
class_test(xsize < 2, errmsg,
  “Something went wrong with the size!”)
```

```
class_stop(errmsg,
  “How did I get here?!”)
```

Error Management

- ```
  Error in thermodynamics_init

  =>thermodynamics_init(L:292) :error in
  thermodynamics_helium_from_bbn(ppr,pba,pth);

  =>thermodynamics_helium_from_bbn(L:1031) :condition
  (omega_b > omegab[num_omegab-1]) is true; You have
  asked for an unrealistic high value omega_b = 7.e-02.
  The corrresponding value of the primordial helium
  fraction cannot be found in the interpolation table.
  If you really want this value, you should fix YHe to
  a given value rather than to BBN
  ```
  ```

Module layout

- Each module contains:

  One function `xxx_init(...)` filling the structure `xx`

  One function `xxx_free(...)` freeing all the memory allocated to this structure

  Some functions `xxx_external_1(...)`, ..., `xxx_external_n(...)` that can be called from other modules (e.g. to read correctly or interpolate the content of the structure `xxx`)

  Some functions `xxx_internal_1(...)`, ..., `xxx_internal_m(...)` that are called only inside the module, within `xxx_init(...)`

- Following order in `xxx.c`:

  ```
  xxx_external_1(...)
  ...
  xxx_external_n(...)
  xxx_init(...)
  xxx_free(...)
  xxx_internal_1(...)
  ...
  xxx_internal_m(...)
  ```

Memory allocation

- Unsafe allocation:

```
ptr = malloc(number*sizeof(type));
```

- Safe allocation:

```
class_alloc(ptr, number*sizeof(type),
            errmsg);
```

- Error management (e.g. if `number==0`)

Wrapper Additions

- classy.pxd:

```
class background:
    double new_thing
```

- classy.pyx

```
def newfunction():
    return bg.new_thing
```

cosmo.newfunction() → new_thing