Results reproducibility with KROME

aka DOCMAKE

Documentation

Benchmarking - things that can go wrong with chemistry/microphysics:

- Different chemical species
- Different reactions
- Different rate coefficients
- Missing ingredients (e.g. cosmic rays)
- Mismatching data (e.g. enthalpy, binding energies, . . .)
- Different photo cross-sections
- . . .

Documentation

KROME is

open-source

Documentation

KROME is

open-source

because we like

reproducibility

Documentation

KROME is

open-source

because we like

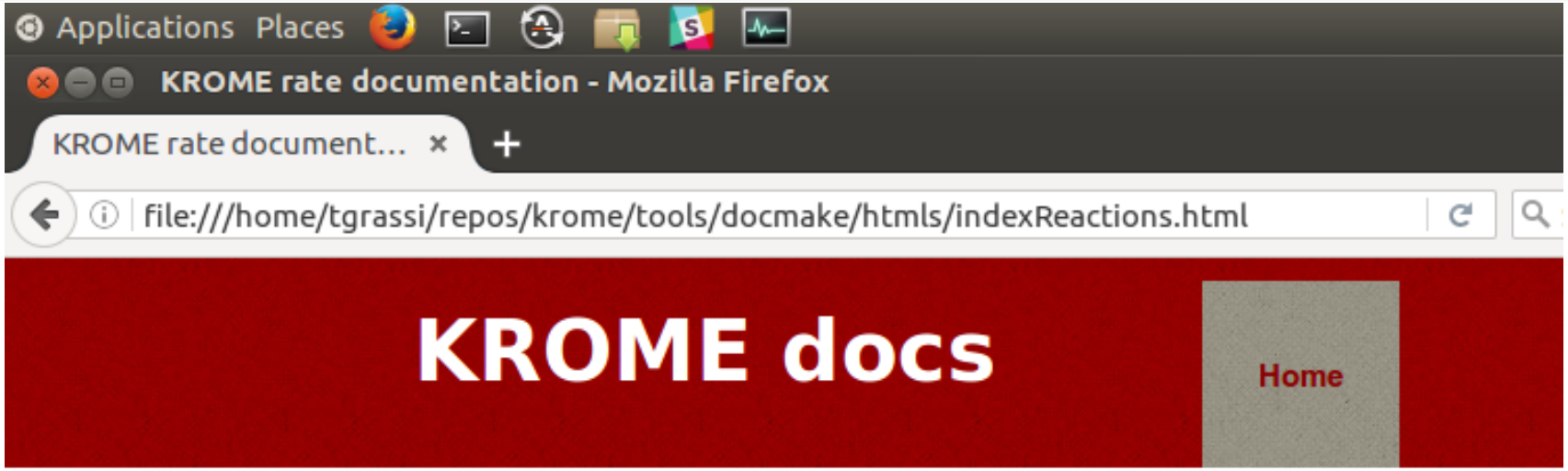
reproducibility

and we want to do

benchmarks

Documentation

your network → DOCMAKE → HTML documentation → your server



Reactions

H^-	+	H	→	H	+	H	+ e^-	details
H_2	+	e^-	→	H	+	H	+ e^-	details
H^-	+	H	→	H_2	+	e^-		details
H^-	+	e^-	→	H	+	e^-	+ e^-	details
H_2	+	e^-	→	H	+	H^-		details
H^-	+	H^+	→	H	+	H		details
He^+	+	e^-	→	He^{++}	+	e^-	+ e^-	details
H	+	e^-	→	H^-				details

Documentation

$C + He^+ \rightarrow C^+ + He$

back
search in KIDA
get rate evaluation in JSON format

KIDA link

Langevin rate: 1.71×10^{-9} $cm^3 s^{-1}$

Langevin

Species involved: C, C^+, He, He^+

rate	:	8.58d-17*Tgas**(0.757)
Tmax	:	.LE.2e2
rate	:	3.25d-17*Tgas**(0.968)
Tmin	:	>2e2
Tmax	:	.LE.2e3
rate	:	2.77d-19*Tgas**(1.597)
Tmin	:	>2e3
Tmax	:	1e8

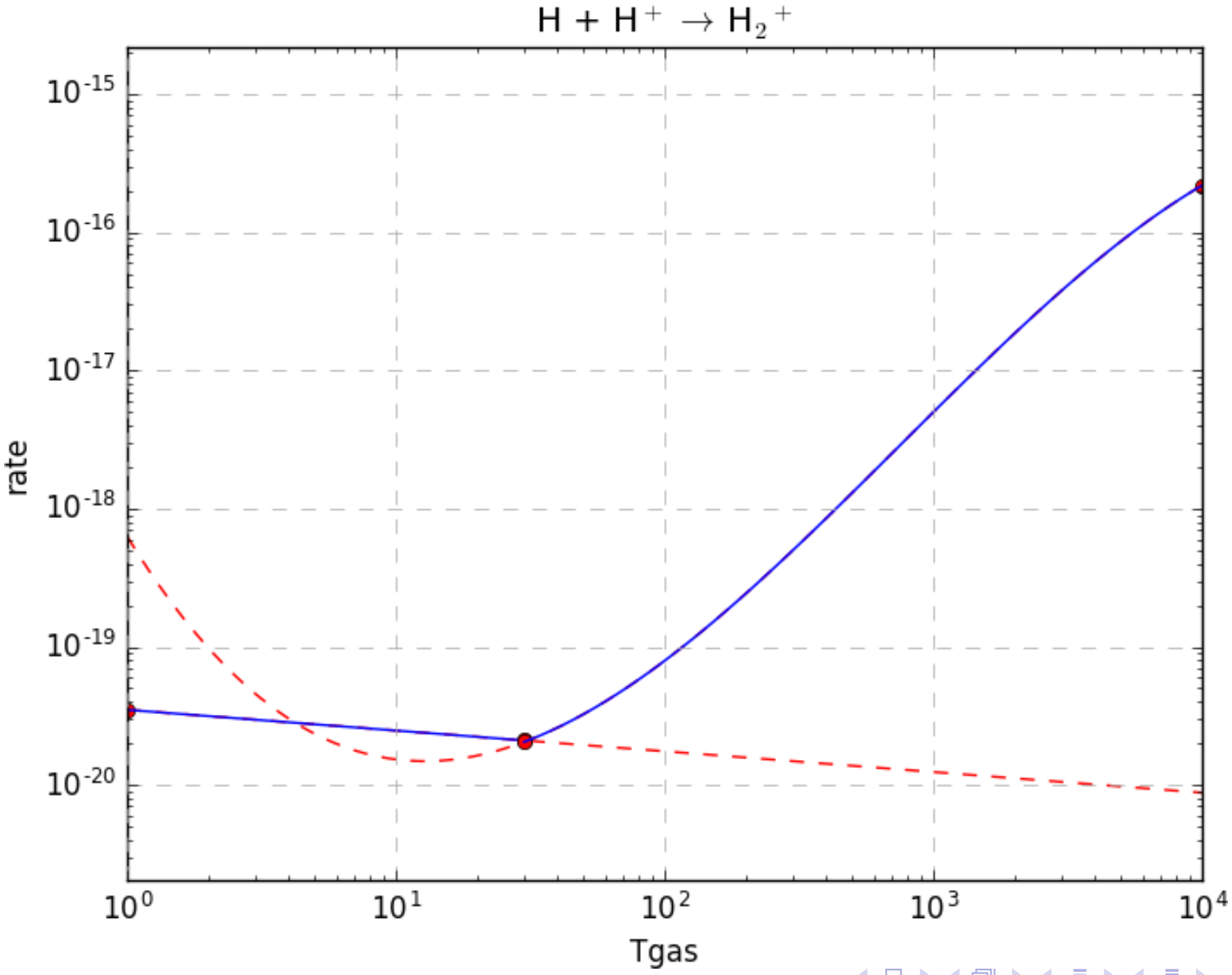
F90 rate

joints evaluation

Evaluated joints:

		Tgas/K	rate
limit1 (Tgas,rate)	:	200.0	4.73551257895e-15
limit2 (Tgas,rate)	:	200.0	5.48631067361e-15
extrapolated (Tgas,rate)	:	200.0	5.48631067361e-15
error	:	15.85% ⚠	
limit1 (Tgas,rate)	:	2000.0	5.09659820321e-14
limit2 (Tgas,rate)	:	2000.0	5.1787467307e-14
extrapolated (Tgas,rate)	:	2000.0	5.1787467307e-14
error	:	1.61% ⚠	

Documentation



Documentation

CH_2

enthalpy

back
Enthalpy @ 298.15K: **391.199425643** kJ/mol
α: **2.12** $Å^3$
All plots in a single page

polarizability

reactions

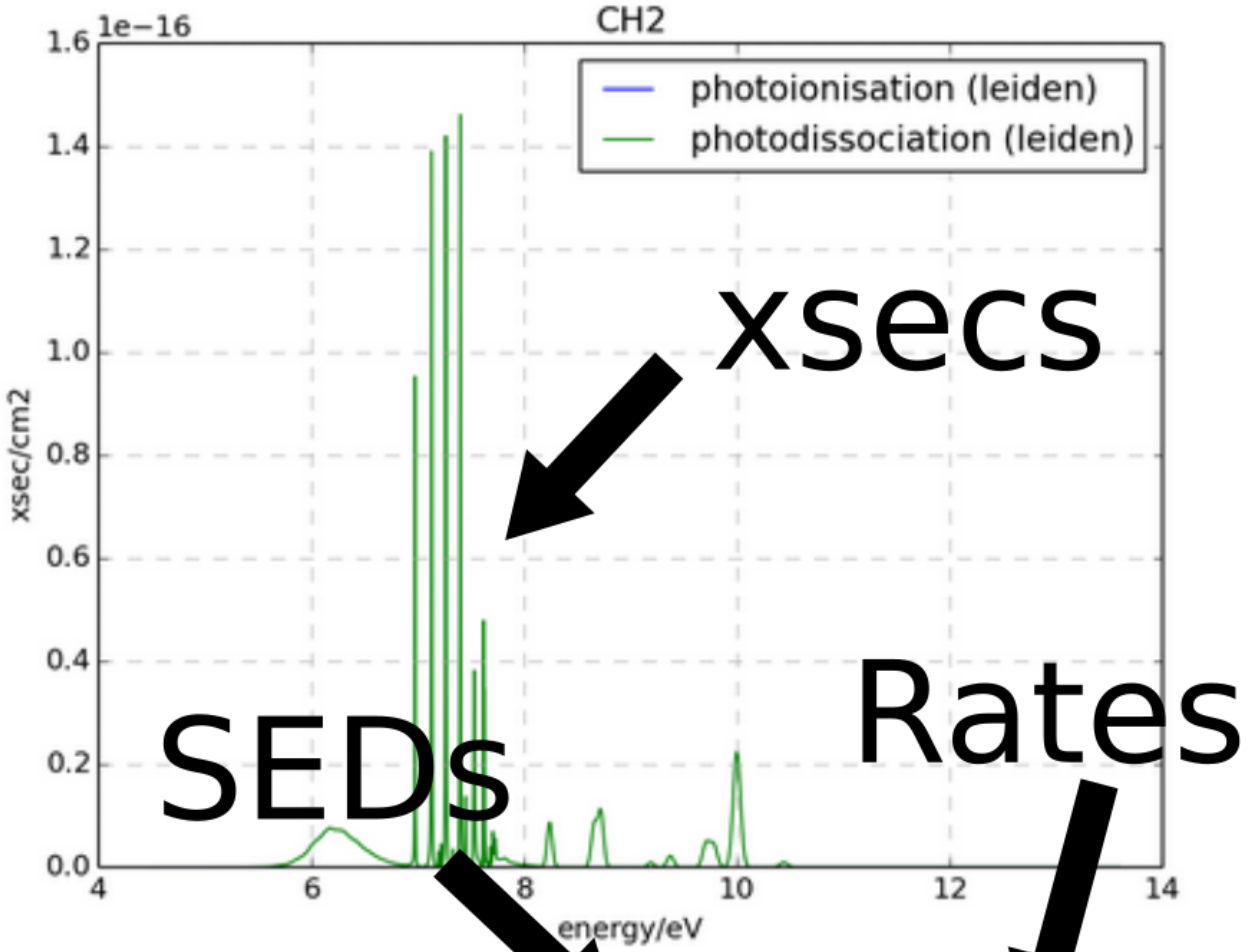
Formation channels

CH_3^+	+	e^-	→	**CH_2**	+	H		details
C	+	H_2	→	**CH_2**				details
CH	+	H_2	→	**CH_2**	+	H		details
C^-	+	H_2	→	**CH_2**	+	e^-		details

Destruction channels

CH_2	+	O	→	HCO	+	H		details
CH_2	+	O	→	CO	+	H_2		details
CH_2			→	CH	+	H		details
CH_2	+	H	→	CH	+	H_2		details
CH_2	+	O	→	CH	+	OH		details

Documentation

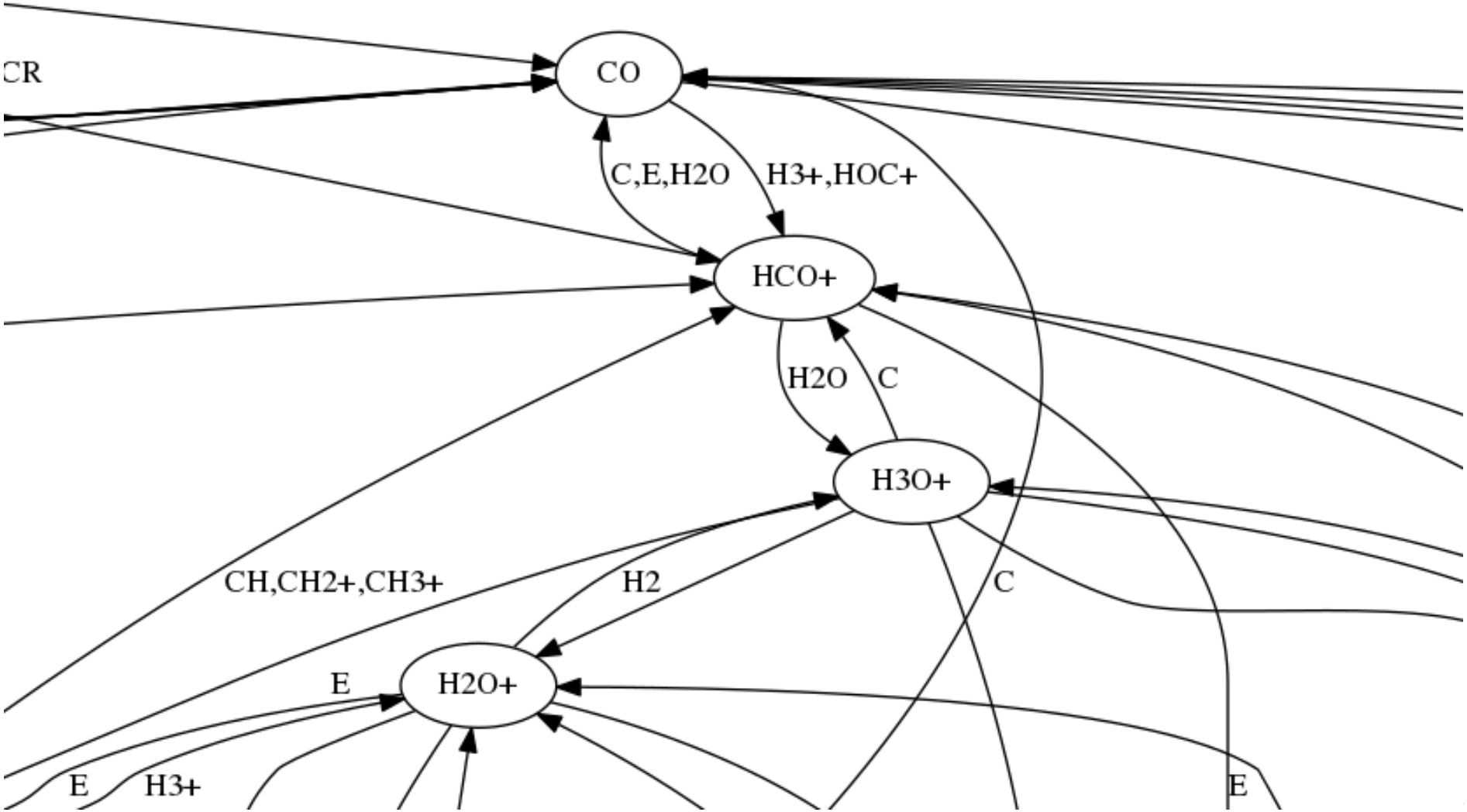


Photochemical rates (1/s)

process	database	radiation	rate (1/s)
photodissociation	leiden	BB@1e4K	1.07×10^{-9}
photodissociation	leiden	BB@2e4K	6.73×10^{-10}
photodissociation	leiden	BB@4e3K	2.24×10^{-9}

Documentation

DOCMAKE creates graphs



Documentation

DOCMAKE creates tables ready for papers

Table A1 – *continued*

No.	Reaction	Rate coefficient	Notes	Ref.
9	$H_2 + He^+ \longrightarrow He + H_2^+$	$k_9 = 7.2 \times 10^{-15}$		8
10	$H_2 + He^+ \longrightarrow He + H + H^+$	$k_{10} = 3.7 \times 10^{-14} \exp\left(-\frac{35}{T}\right)$		8
11	$H_2 + He^+ \longrightarrow He^+ + H + H$	$k_{11} = 3 \times 10^{-11} \sqrt{(T/300)} \exp\left(-\frac{52000}{T}\right)$		9
12	$He^{++} + e^- \longrightarrow He^+ + \gamma$	$k_{12} = \frac{1.891 \times 10^{-10}}{\sqrt{\frac{T}{9.37}}\left(1+\sqrt{\frac{T}{9.37}}\right)^{0.2476}\left(1+\sqrt{\frac{T}{2774000}}\right)^{1.7524}}$		10
13	$H + e^- \longrightarrow H^- + \gamma$	$k_{13} = 1.4 \times 10^{-18} T^{0.928} \exp\left(-\frac{T}{16200}\right)$		10
14	$H^- + H \longrightarrow H_2 + e^-$	$k_{14} = \frac{1.35 \times 10^{-9}\left(T^{0.09849} + 0.32852 T^{0.5561} + 2.771 \times 10^{-7} T^{2.1826}\right)}{1 + 0.00619 T^{1.0461} + 8.9712 \times 10^{-11} T^{3.0424} + 3.2576 \times 10^{-14} T^{3.7741}}$		11
15	$H + H^+ \longrightarrow H_2^+ + \gamma$	$k_{15} = 2.1 \times 10^{-20} (T/30)^{-0.15}$	$T < 30\,K$	12
		$= 10^{\wedge}\left(-18.2 - 3.194 \log(T) + 1.786 \log(T)^2 - 0.2072 \log(T)^3\right)$	$T \geqslant 30\,K$	
16	$H_2^+ + H \longrightarrow H_2 + H^+$	$k_{16} = 6 \times 10^{-10}$		13
17	$H_2 + H^+ \longrightarrow H^+ + H$	$k_{17} = 10^{\wedge}\left(-8875.5774\right.$	$T > 10^5\,K$	14

(Frostholm+2018, arXiv:1809.05541)

Documentation

KROME-DOCMAKE as a flexible Python class

import

load network

reactions as objects

species as objects

plots

```
import docmake

ntw = docmake.network("cloud.csv")

rea = ntw.reactions[0]
print "reactants:", [x.name for x in rea.reactants]
print "reactant mass, g: ", [x.mass for x in rea.reactants]

sp = ntw.getSpeciesByName("H3+")
print "name: ", sp.name
print "LaTex: ", sp.nameLatex
print "enthalpy @ 298.15 K: ", sp.getEnthalpy(ntw.thermochemicalData), "kJ/mol"
print "photorate: ", sp.getPhotoRate("leiden", "photodissociation", "BB@1e4K"), "1/s"

sp.plotXsec("xsec.png")
rea.plotRate(myOptions, "plot.png")
```

Exercise: coupling a 1D "radiative transfer" code with krome

Exercises plan

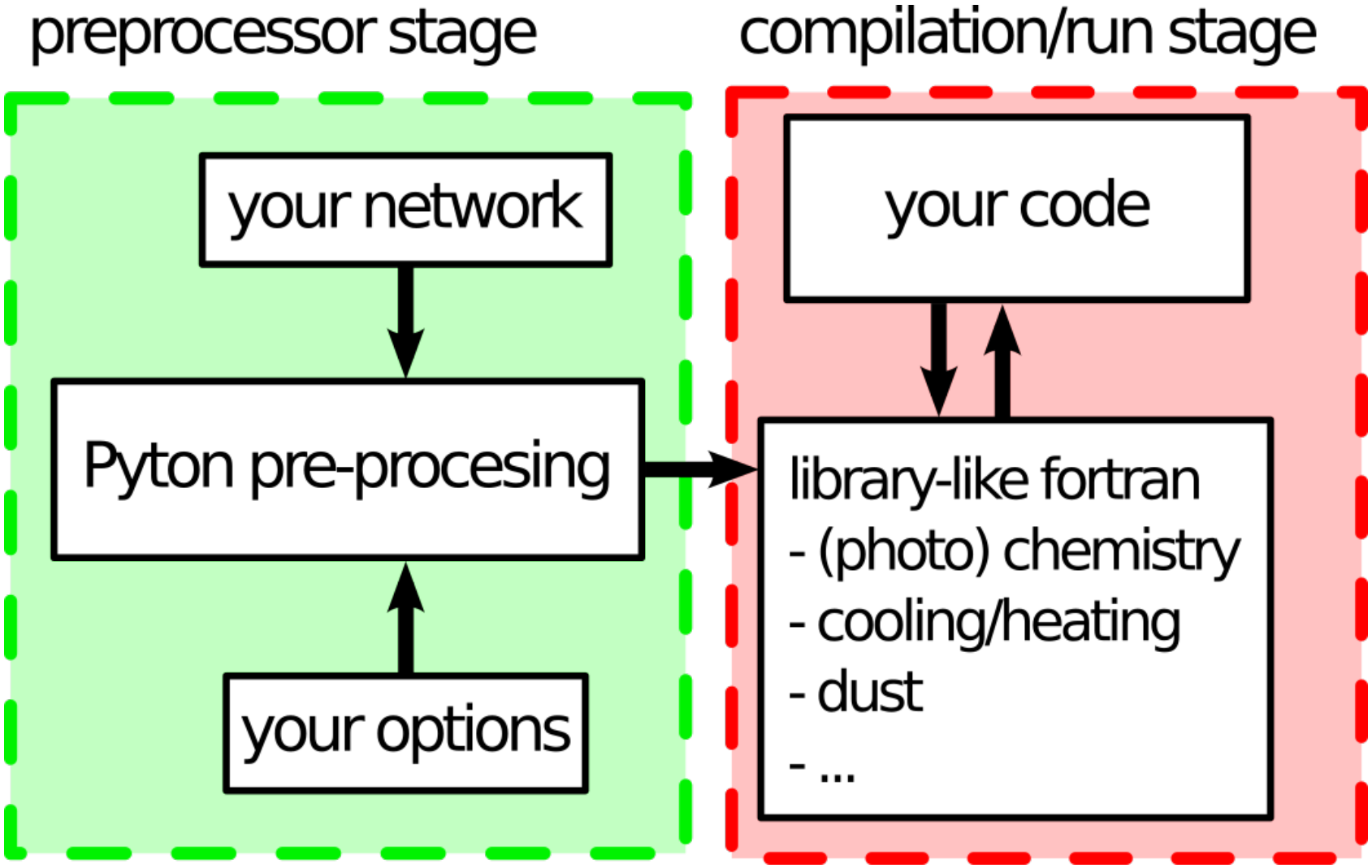
- Part 0: stand-alone chemical network with KROME
- **Part 1: adding chemistry to PROTO**
- Part 2: adding thermal processes to PROTO
- Part 3: adding photochemistry to PROTO
- Part 4: adding dust to PROTO

Legacy from yesterday's exercise

1. a chemical network for water formation
2. KROME has two stages
3. learn how to feed reactions and options to KROME

Aims of today's exercise

1. learn the basics of PROTO "RT" code
2. learn to couple KROME with PROTO
3. add water network to PROTO

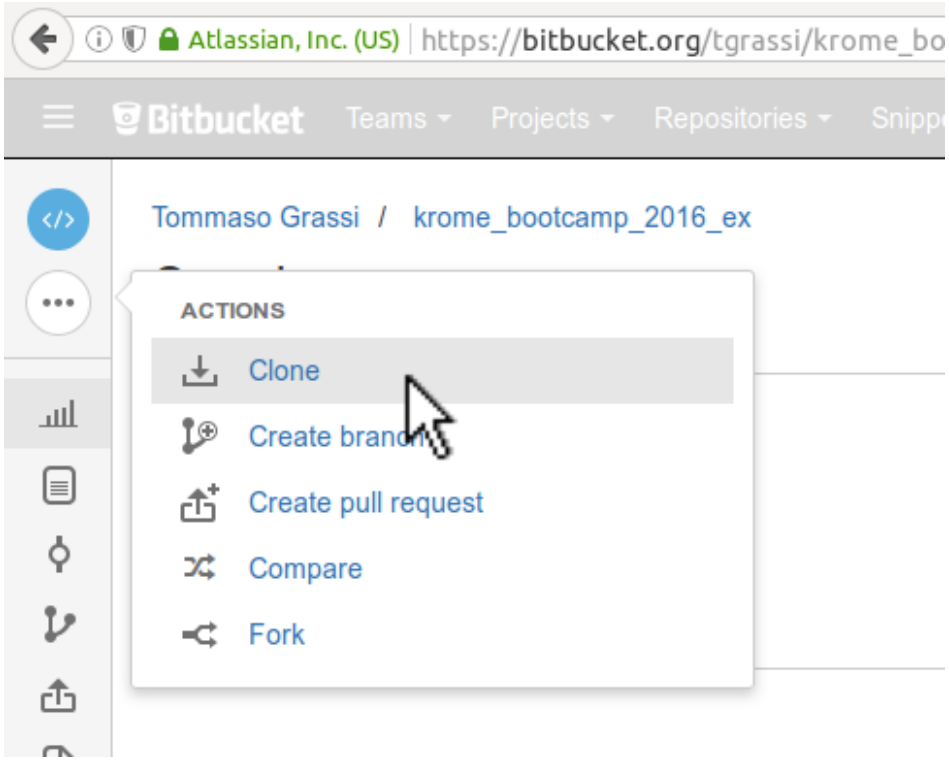


GIGO philosophy!

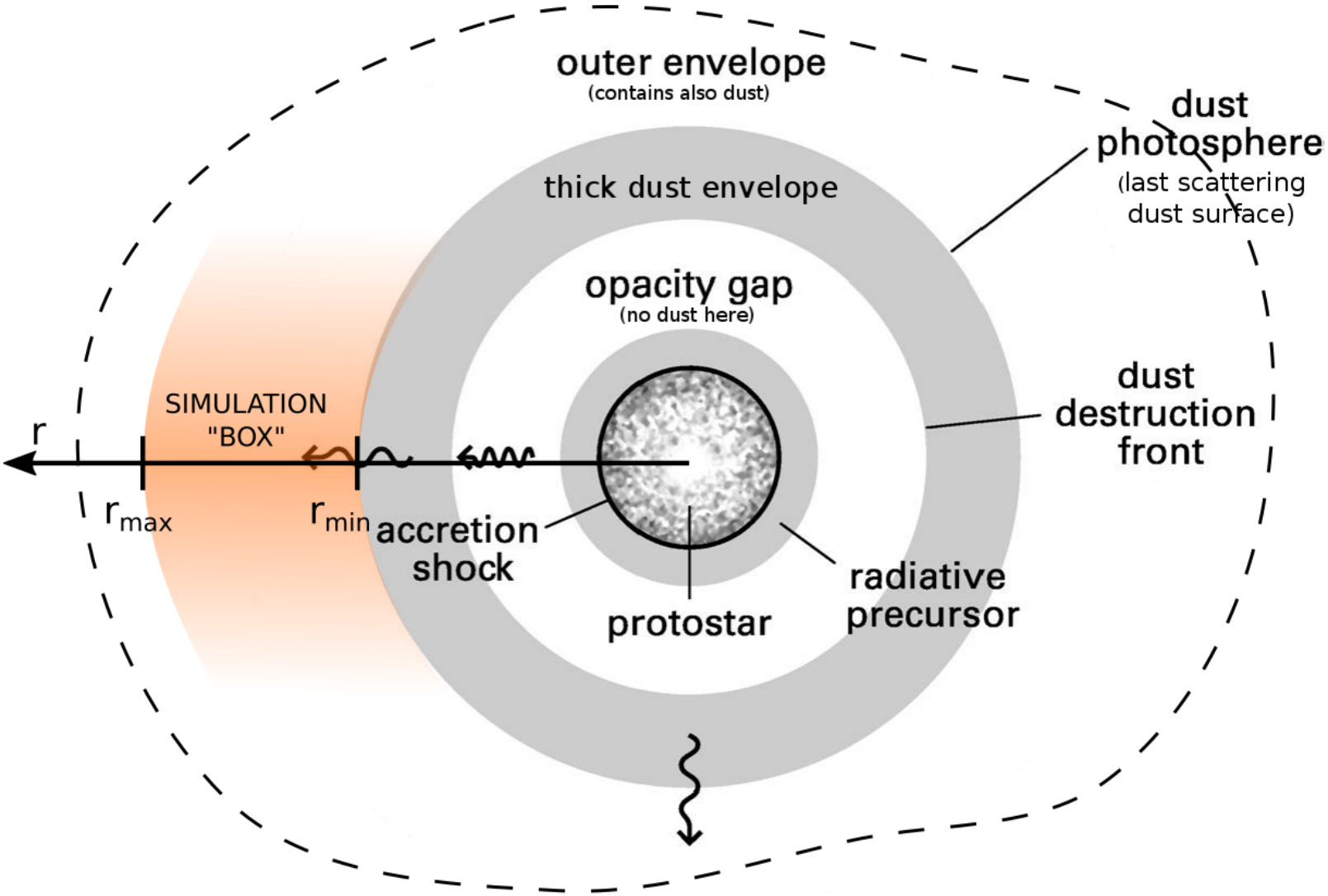
The material for the exercises is provided in a bitbucket repository

Basic commands for cloning the repository

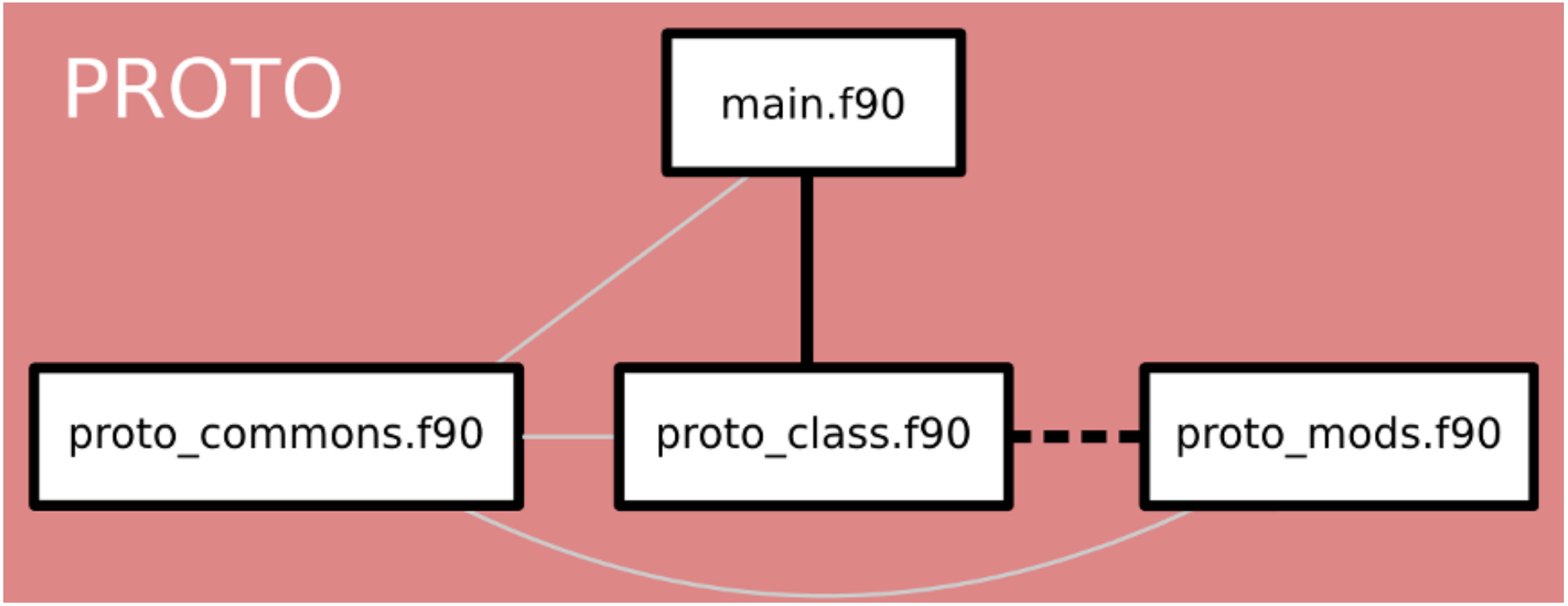
- `git pull origin`
- note that modifications on your local version cannot be ovewritten by the `pull` command



PROTOstar

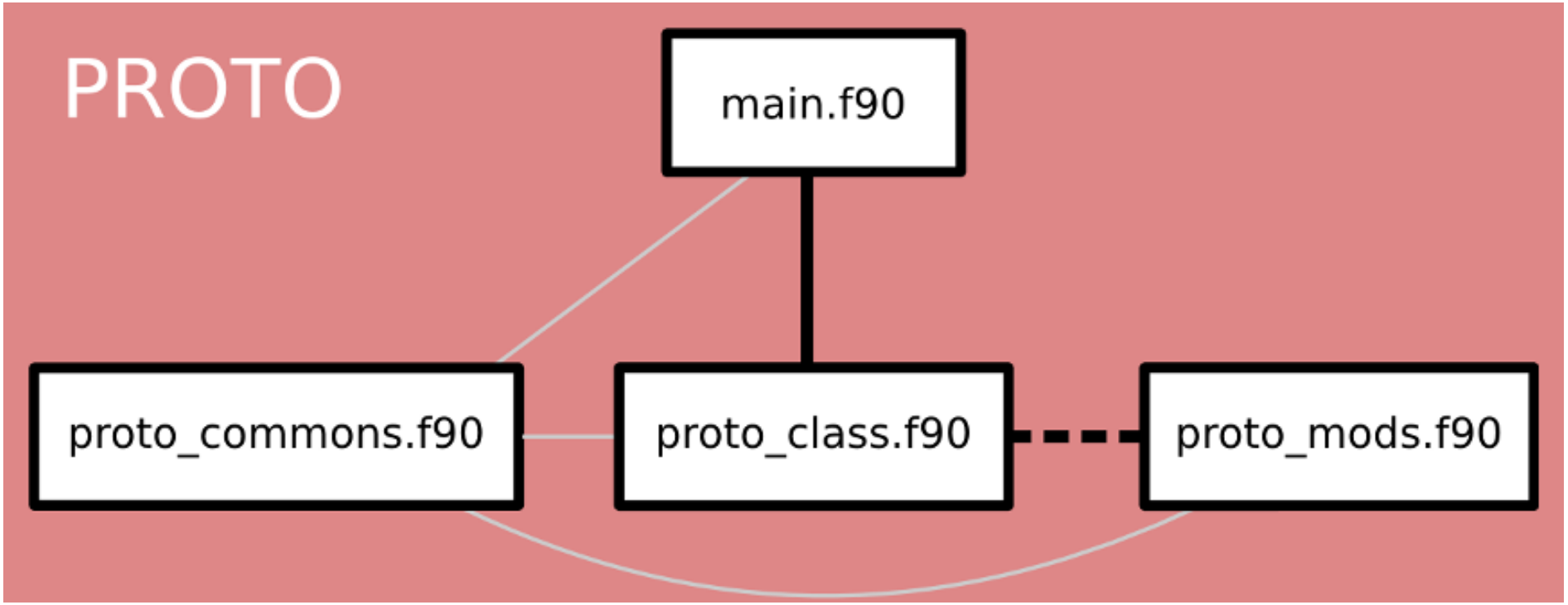


see Sect.11 from Sthaler+Palla (2005)



PROTO

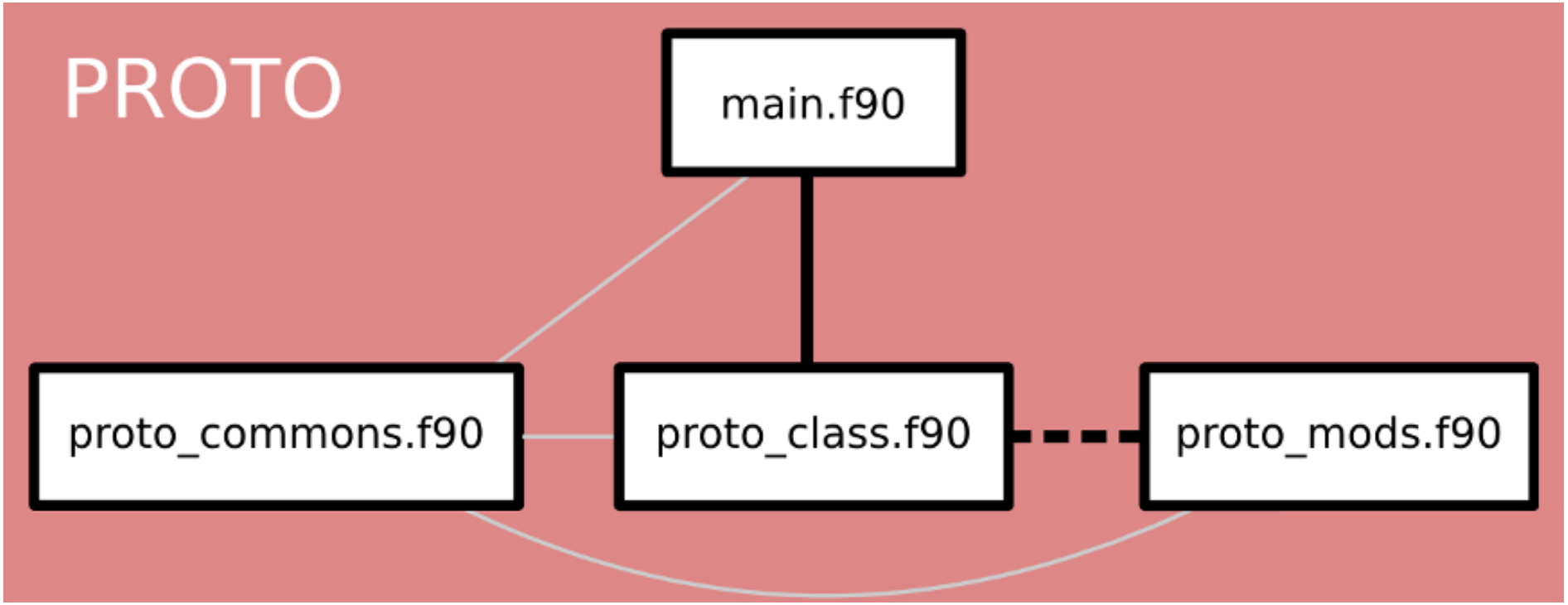
- Computes simple multi-frequency radiative transfer in 1D
- Single Fortran class named `proto`
- (i) geometrical attenuation (r^{-2}) and (ii) ν-dependent opacity, $e^{-\tau_\nu}$
- Wiki `https://bitbucket.org/tgrassi/krome_bootcamp_2018_ex/wiki/Home`



main.f90

This is the main program file that contains:

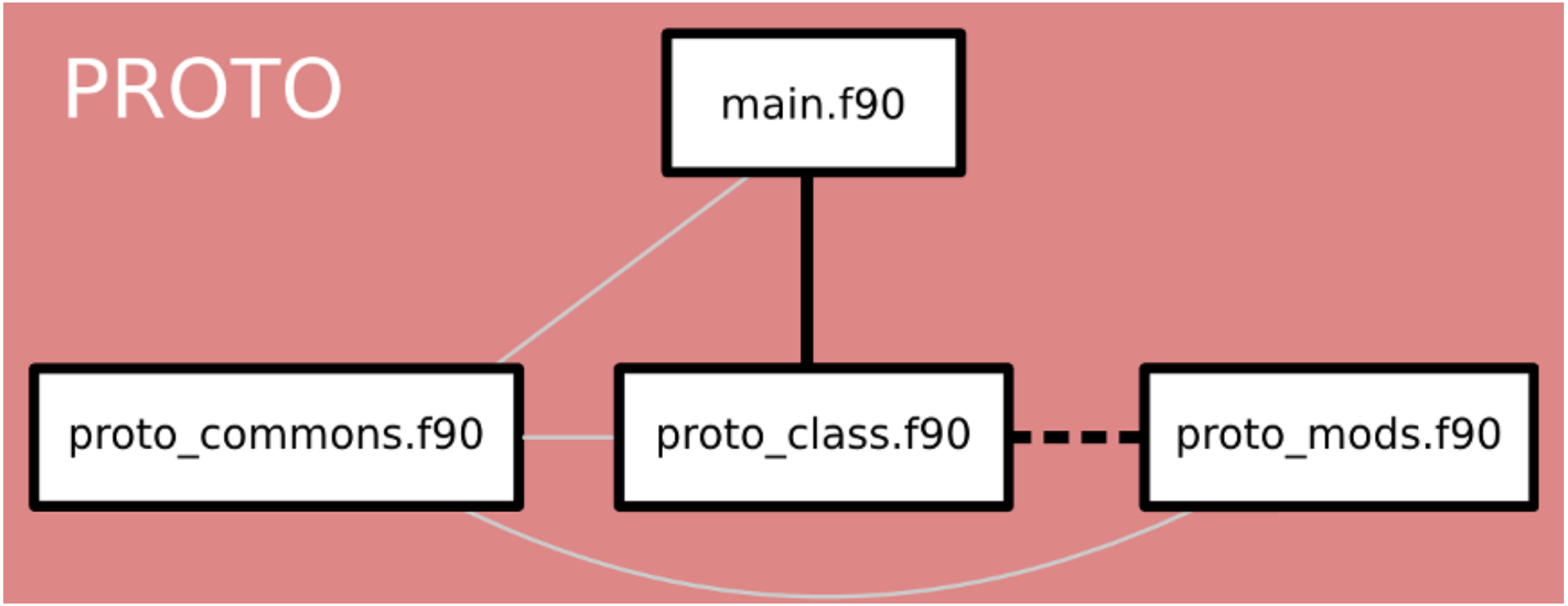
- a call to the class constructor
- various calls to initialization procedures (incl. chemistry)
- the loop over time (as yesterday's exercise)
- calls to output procedures



proto_commons.f90

Contains common variables used by the class and by main.f90:

- ncells: the number of cells (i.e. grid points)
- nchemistry: the number of chemical species (passive scalars)
- ndust: the number of dust bins (passive scalars)
- nenergy: the number of photo bins (in energy)



proto_class.f90

The class file. It contains *attributes* and *procedures*, like

- density: array of mass densities (g cm^{-3}, ncells)
- x: passive scalars, mass fractions, ncells $\times$ nchemistry

- new_proto: class constructor
- load_parameters_from_file: guess

How to access a class attribute/method

in Fortran

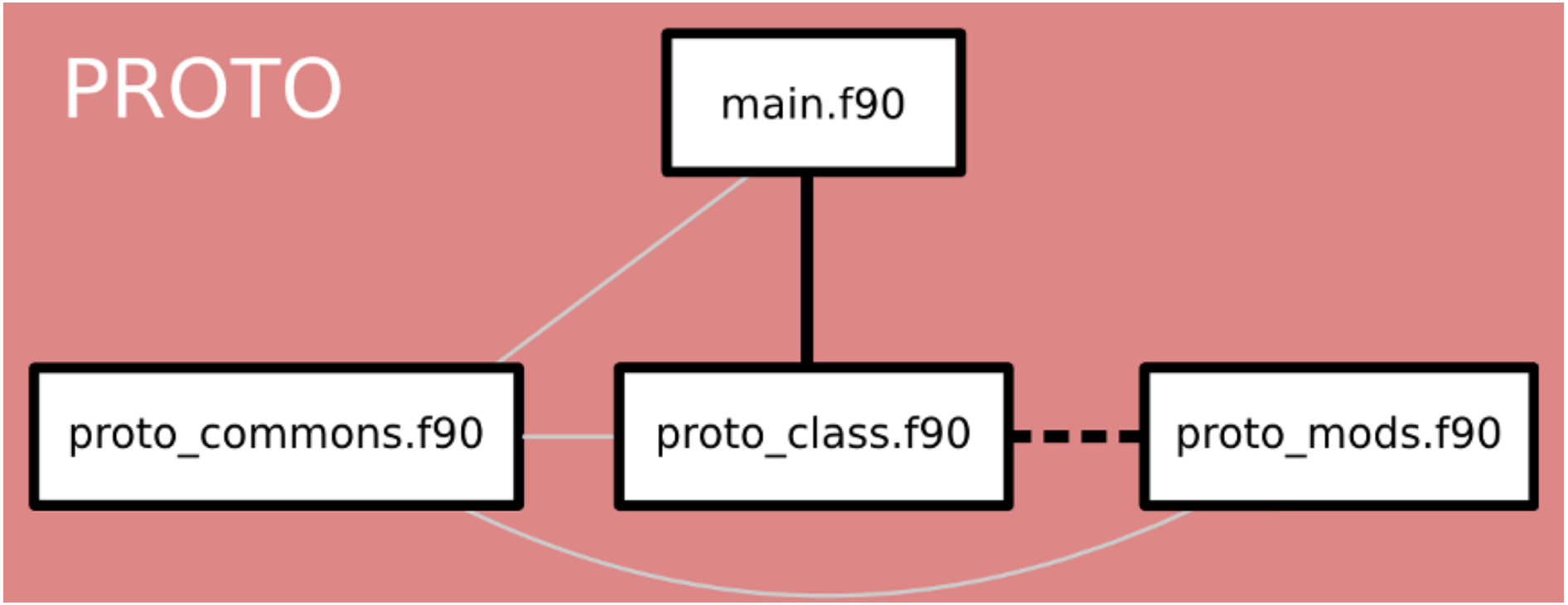
```
subroutine init_chemistry(this)
  implicit none
  class(proto),intent(inout)::this
  this%x(:, :) = 0d0
end subroutine init_chemistry
```

equivalent of Pythonesque

```
def init_chemistry(self):
  self.x[:, :] = 0e0
```

or C++

```
void init_chemistry(){
  this->x = 0e0;
}
```



proto_mods.f90

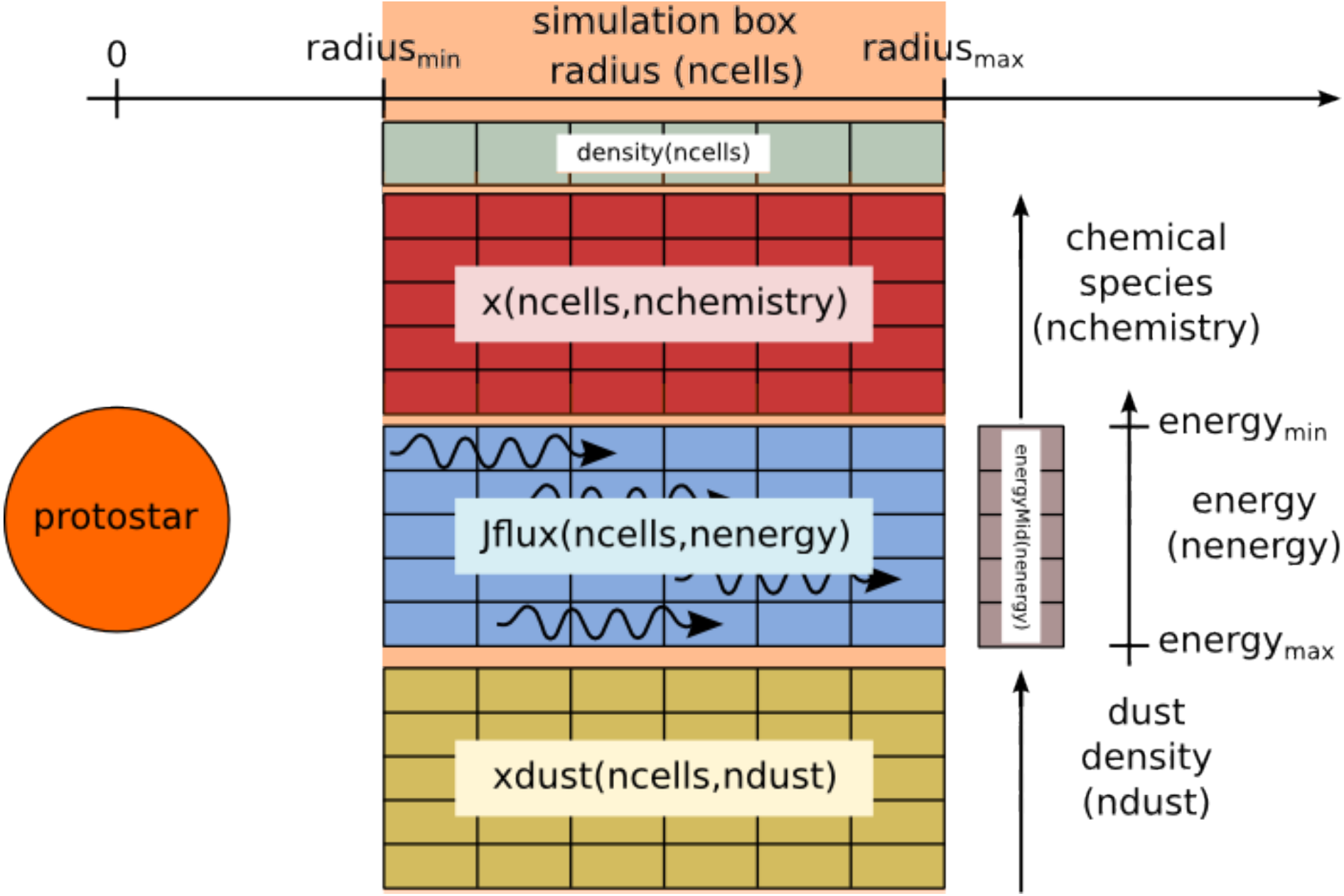
This is where you will add most of the modifications

- dump_density_profile: dump almost everything
- dump_radiation_flux: dump radiance (eV/cm2) $\forall$ energy and cell
- init_chemistry: where chemistry is initialized
- solve_chemistry: where chemistry is advanced by Δt

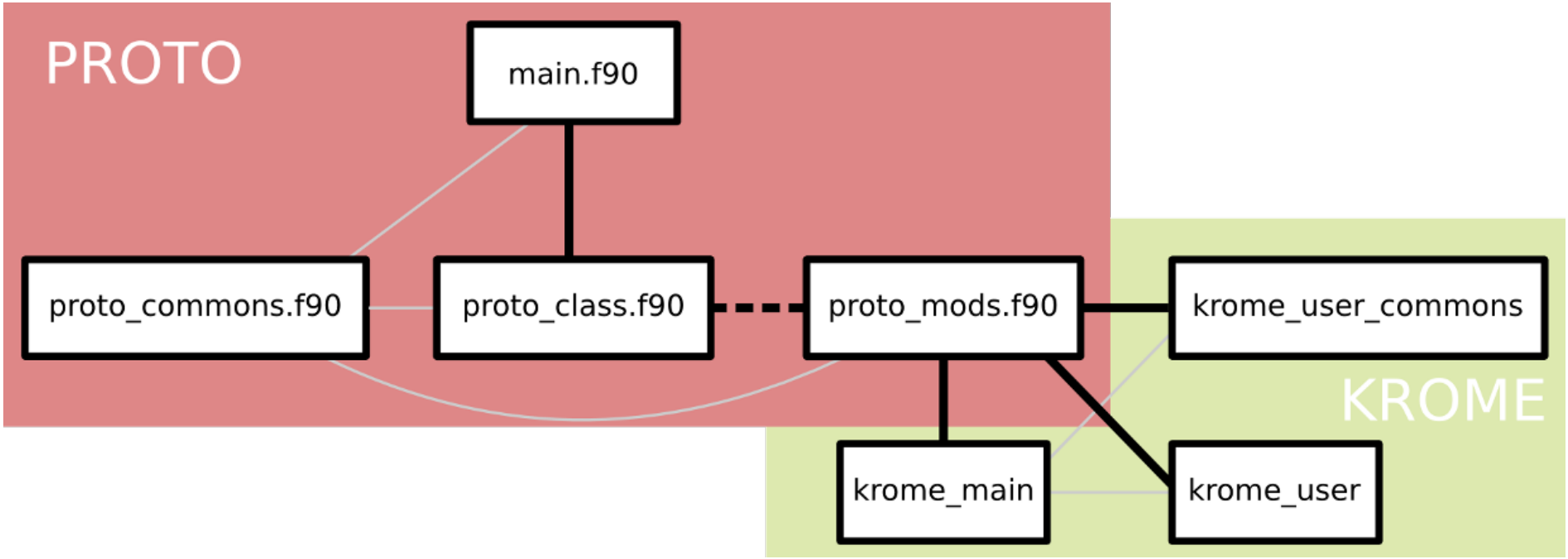
Use the wiki:

Recap:

- Relevant excerpt from Stahler+Palla (2005) (PDF);
- List of files, procedures, attributes;
- Variables data structure (schematic);
- Connections within PROTO and to KROME (schematic);
- How to access class procedures and attributes;
- How to add a new procedure to the class;
- Output procedures and default file format;
- How the Makefile works.


0
radius$_{min}$
simulation box
radius (ncells)
radius$_{max}$
density(ncells)
x(ncells,nchemistry)
chemical
species
(nchemistry)
protostar
Jflux(ncells,nenergy)
energyMid(nenergy)
energy$_{min}$
energy
(nenergy)
energy$_{max}$
xdust(ncells,ndust)
dust
density
(ndust)

KROME Bootcamp 2018 - PROTO



In `proto_mods.f90` you find `init_chemistry` and `solve_chemistry`

`init_chemistry` is called once by `main.f90`
`solve_chemistry` is called by the loop on time in `main.f90`

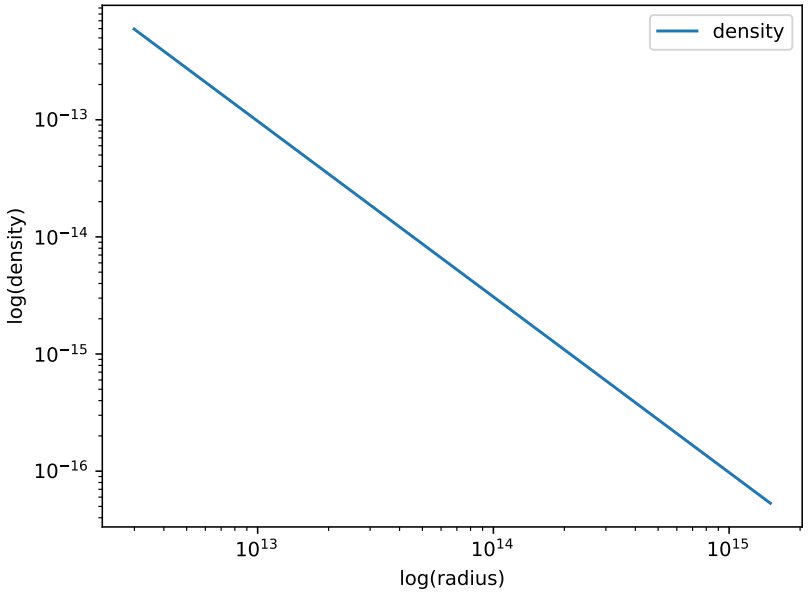
KROME Bootcamp 2018 - PROTO

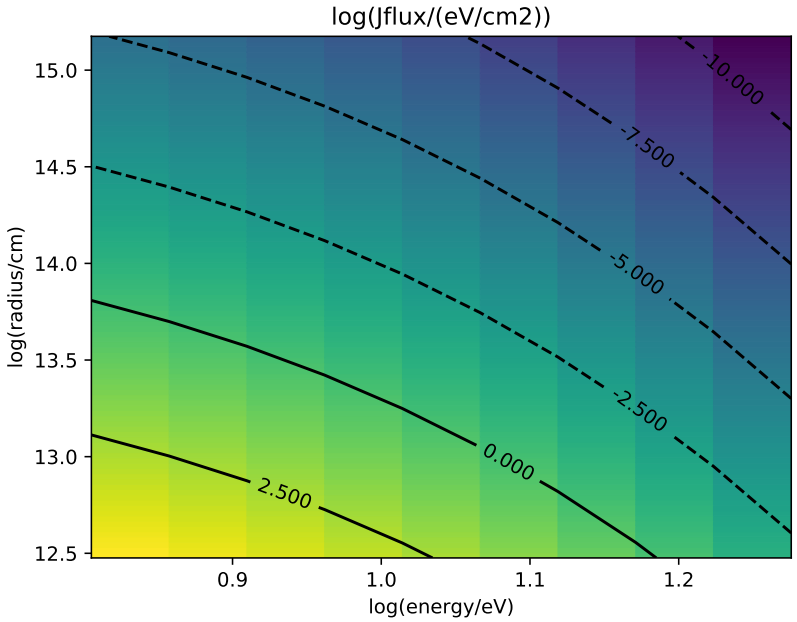
- in `proto/` folder `make` and run `./proto`
- PROTO already produces two output files `density.out` and `radiation.out`

Output format

- time/s, radius/cm, density/(g cm^{-3}), T/K, x(:)/#
- time/s, radius/cm, energy/eV, J/(eV cm^{-2}), opacity/#

p.plot2()
p.plot3()





What to do

1. run KROME and copy required files into PROTO folder
2. set passive scalars in PROTO
3. modify `init_chemistry`
4. modify `solve_chemistry`

Aim: Add water chemistry from Problem 0 to PROTO using KROME

GOOD WORK!