KROME: a first look into the package

Download, explore, and prepare a network file

July 20, 2015

Stefano Bovino

stefano.bovino@uni-hamburg.de

Hamburger Sternwarte
Hamburg University
Germany

Aims and goals

what we will learn during this talk (hopefully!)

- What is KROME?
- Git and direct download
- How KROME is structured
- How to run KROME
- How to prepare a simple chemical network
- Quick overview of the pre-built networks and tests

Why KROME?
Motivation

- the chemistry is a complex problem
- no general codes to solve microphysics (in HD)
- only very specific codes around (env-dependent)
- chemistry mostly hard-coded
- often not updated chemistry
- approximations/assumptions (env-dependent) to speed-up
- numerically unstable solvers (sometimes)
- some attempts to have external libraries but not enough

Why KROME?
Motivation

- the chemistry is a complex problem
- no general codes to solve microphysics (in HD)
- only very specific codes around (env-dependent)
- chemistry mostly hard-coded
- often not updated chemistry
- approximations/assumptions (env-dependent) to speed-up
- numerically unstable solvers (sometimes)
- some attempts to have external libraries but not enough

Why KROME?
chemistry bottlenecks

$$\frac{dn_i}{dt} = \overbrace{\sum_{lm} k_{lm}(T) n_l(t) n_m(t)}^{formation} - \overbrace{\sum_{j} k_{ij}(T) n_i(t) n_j(t)}^{destruction} \quad (1)$$

$$\frac{dT}{dt} = \frac{\gamma - 1}{k_B n_{tot}} (\Gamma(T, n_i) - \Lambda(T, n_i)) \quad (2)$$

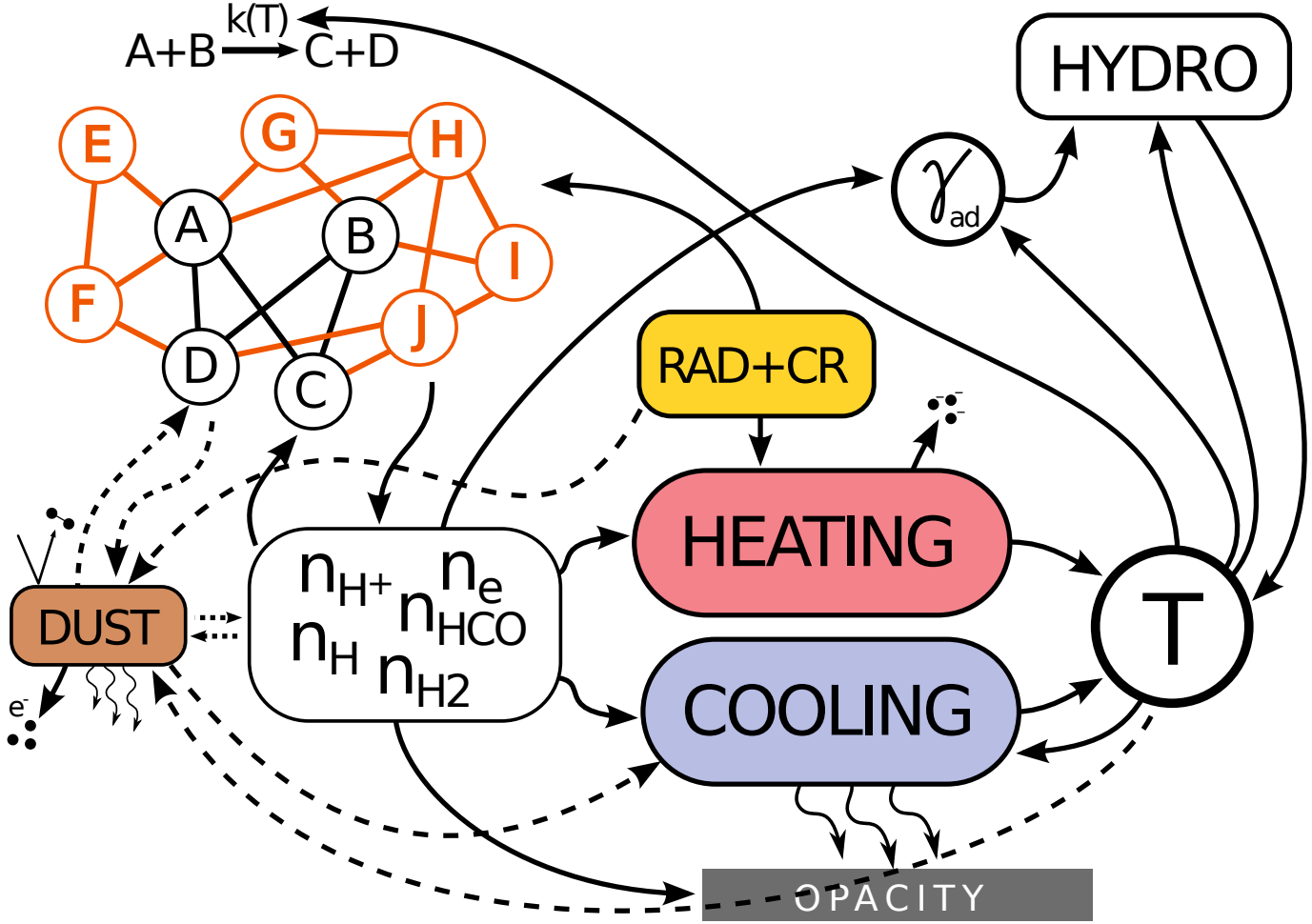
(i) Stiffness $\rightarrow$ need for implicit solvers
- widely varying time scales
- very stiff problems $\rightarrow$ computationally intensive
- accurate and efficient solver

(ii) Network complexity $\rightarrow$ need for reduction techniques
- *a-priori* $\rightarrow$ reduce ODEs number (T. Grassi, **SB**+, MNRAS 2013)
- *on the fly* $\rightarrow$ reduce RHS terms (T. Grassi, **SB**+, MNRAS 2012)

(iii) Rates availability and accuracy
- basic information

(iv) Connected with many physical processes
- very CPU demanding
- hydro-chemistry-radiation strongly coupled

Why KROME?
chemistry bottlenecks



KROME

T. Grassi, **SB**+ MNRAS, 2014

KROME

better science through chemistry

- KROME is a package which helps users to build their own microphysics
- KROME is open source
- KROME is a pre-processor
- the core of KROME is a mix of PYTHON and FORTRAN 90.
- a chemical network is compulsory to run KROME
- to use KROME in your simulations you need to run ./krome first
- users functions are provided to an easy usage (see Tommaso's talk)
- KROME reduces the hard-coding from the users
- to embed KROME into an external code requires a simple call
  - `call krome(x(:), Tgas, dt)`

Bitbucket and KROME
a happy marriage

Atlassian Bitbucket

KROME better science through chemistry

KROME is developed and maintained on Bitbucket

What is bitbucket?

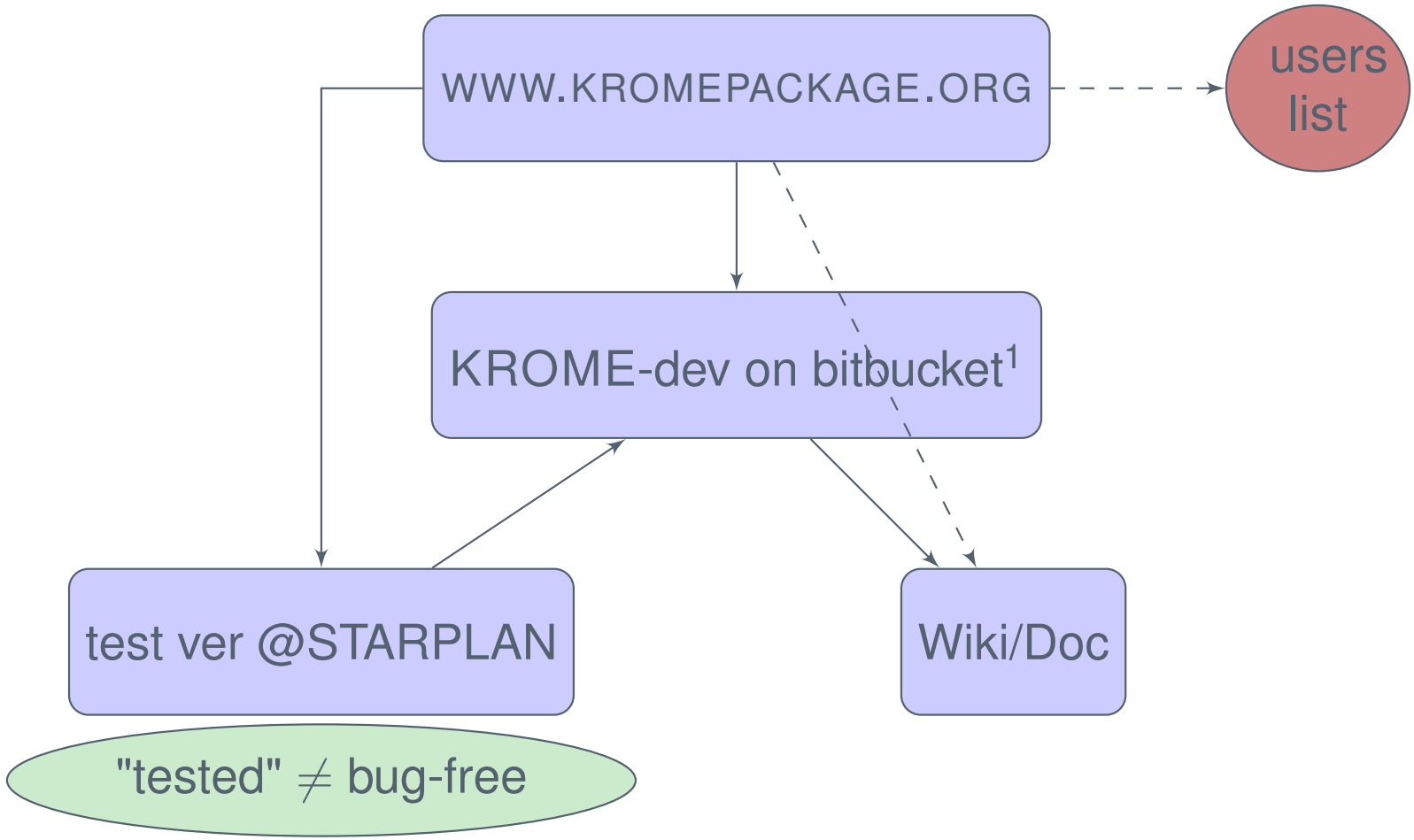
Bitbucket is a web-based hosting service for projects that use **distributed revision control system (DRCS)**

- Git
- Mercurial

Main advantages

- it is not a centralized system (there is no central server)
- it is a peer-to-peer network
- you can work without being connected to a network

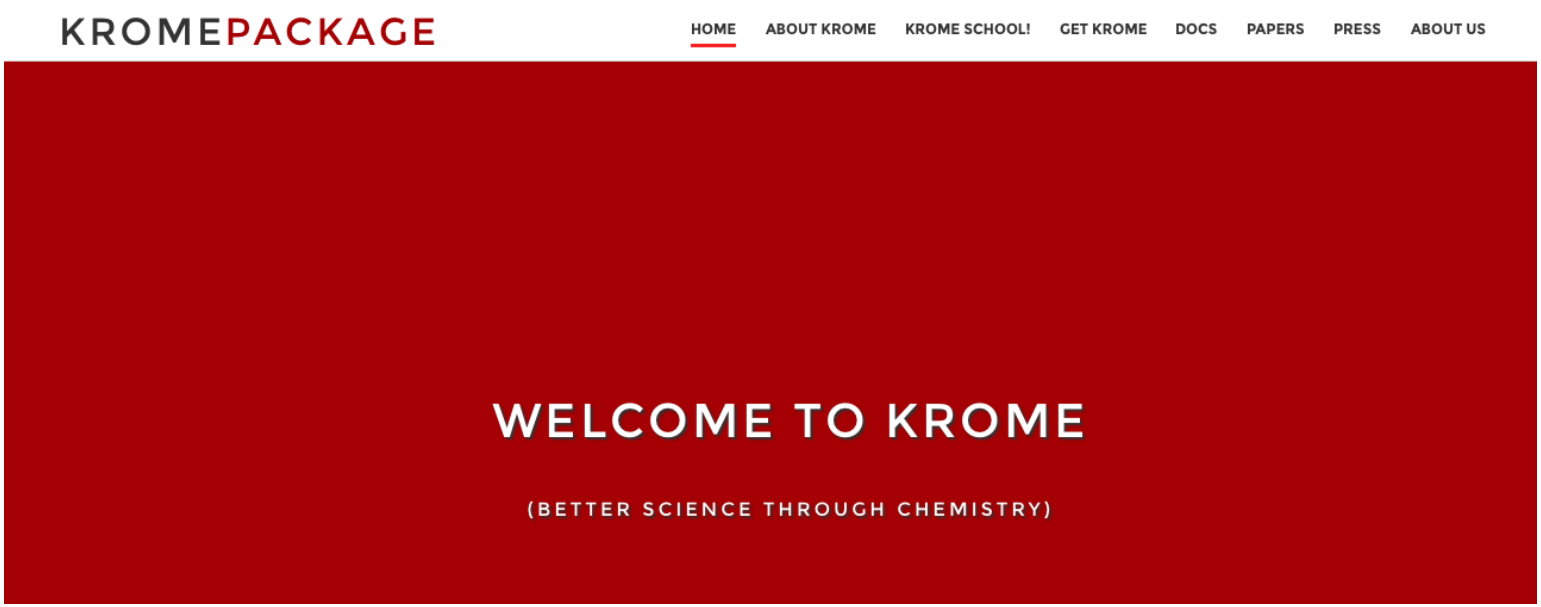
How to get KROME



[1] https://bitbucket.org/tgrassi/krome

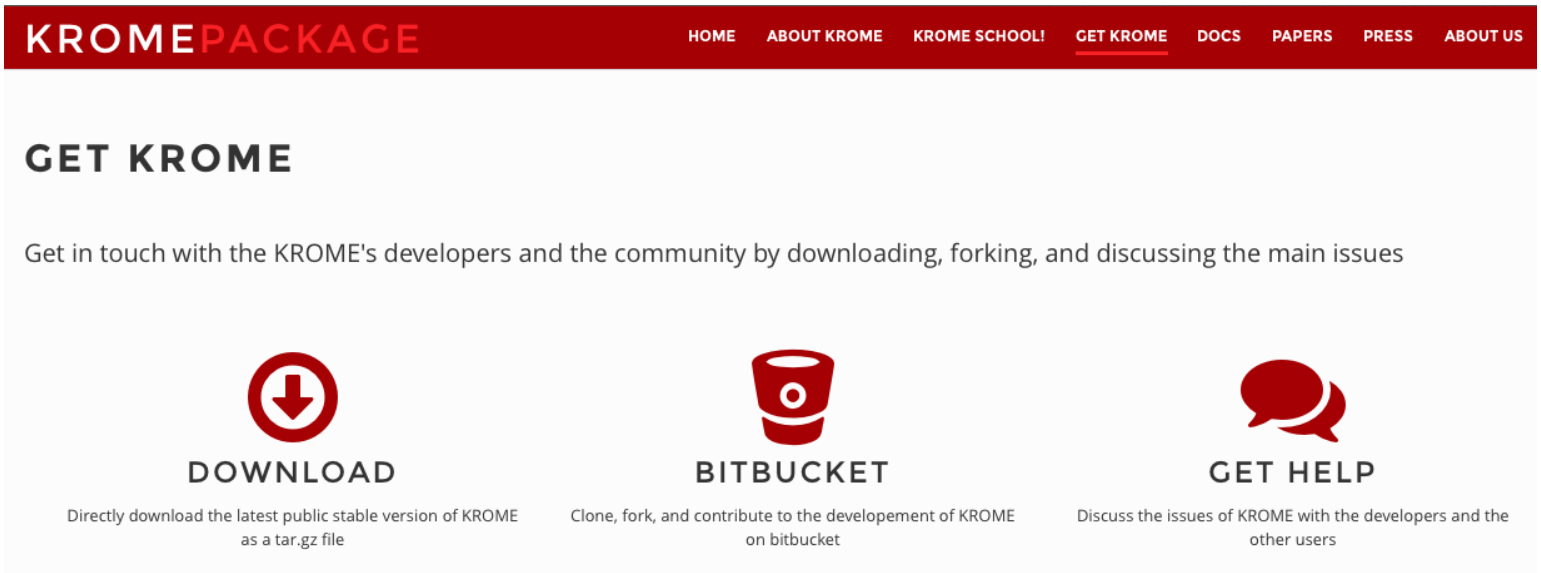
How to get KROME
via the website

- simply go to `http://www.kromepackage.org`



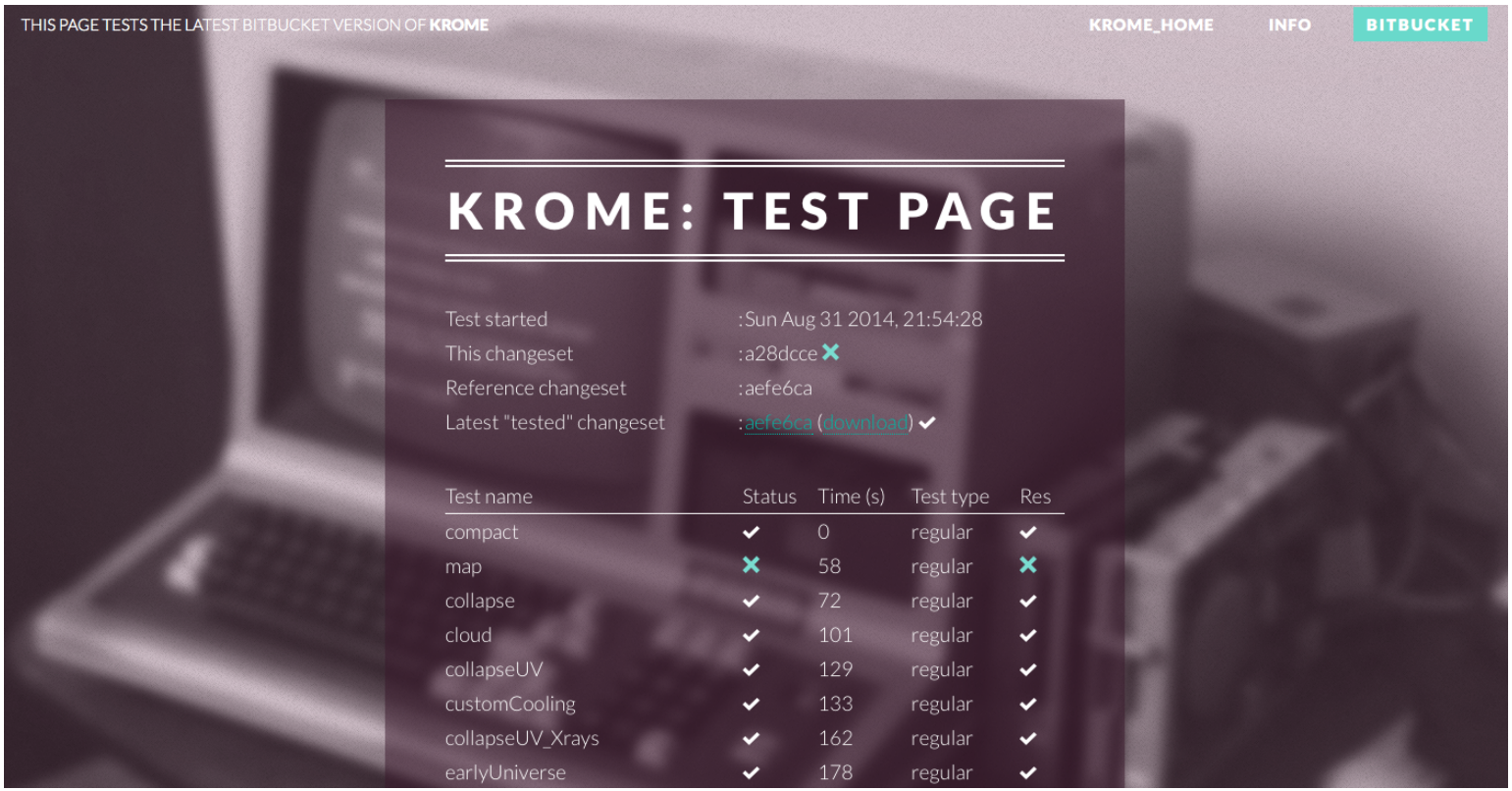
How to get KROME
via the website

- simply go to `www.kromepackage.org`



How to get KROME

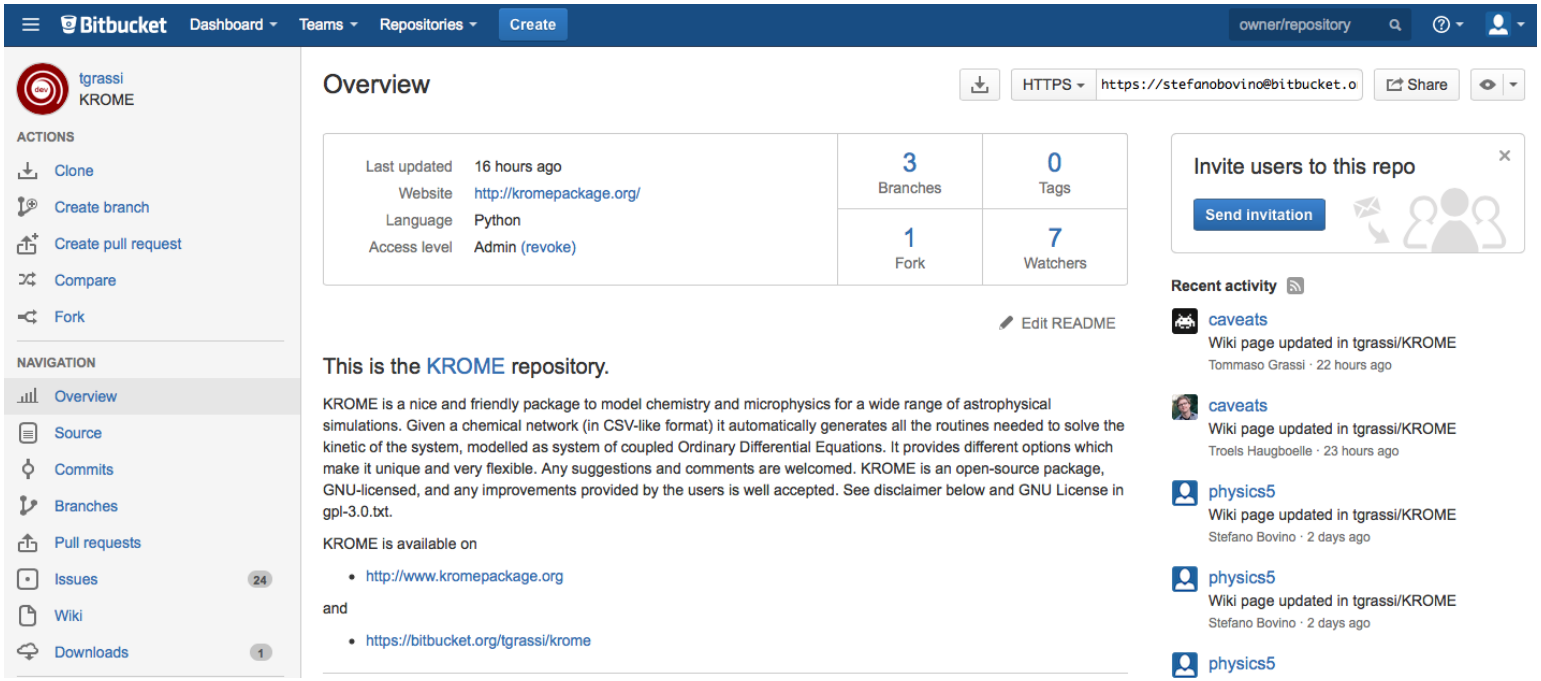
- from the website you can access the repository
- a version of KROME is "tested" if the test called "regular" are working



How to get KROME
via bitbucket

requirements:

- basic knowledge of the **git** commands
- `https://bitbucket.org/tgrassi/krome`



How to get KROME
via bitbucket


```
$ git clone
    https://bitbucket.org/tgrassi/krome.git
Cloning into 'krome'...
remote: Counting objects: 4193, done.
remote: Compressing objects: 100%
    (2063/2063), done.
remote: Total 4193 (delta 2675), reused 3278
    (delta 2094)
Receiving objects: 100% (4193/4193), 16.81
    MiB | 673.00 KiB/s, done.
Resolving deltas: 100% (2675/2675), done.
Checking connectivity... done.
```

- you will generate a folder named "krome"
- note that you're downloading the "dev" non-tested version

Explore KROME
through the files


```
$ ls
README.md       data              outtest.md5
alltest.py      gpl-3.0.txt       patches
argparse.py     krome             solver
build           kromelib.py       src
changelog.txt   kromeobj.py       tests
clean           networks          tools
custom.dat      options_example   wizard
```

- folders
- python files-the core of the package
- executable-pre-processor
- other files (e.g. clean to clean up your working dir)

Explore KROME
through the folders

Folders names are self-explanatory

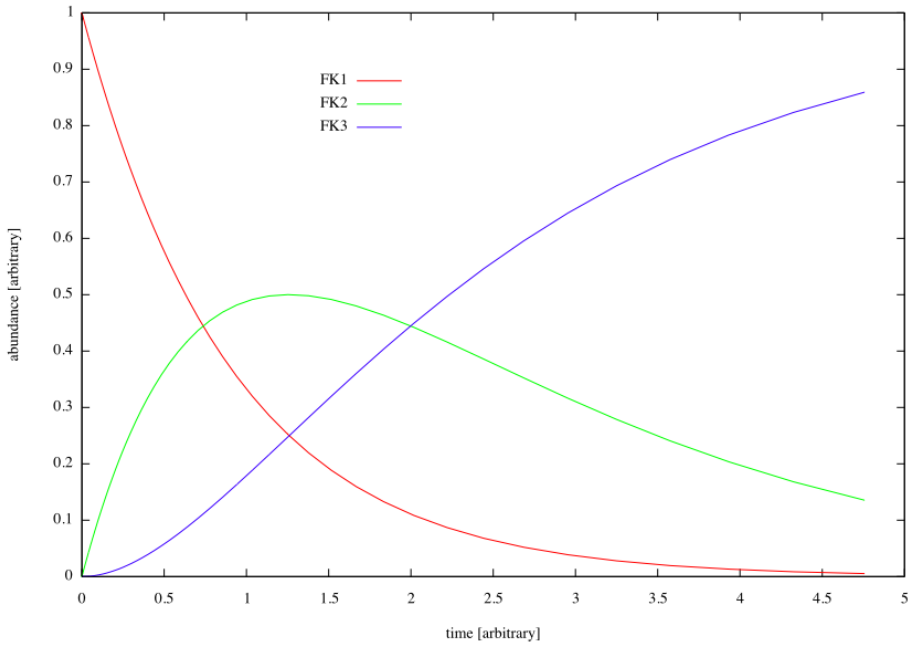
- BUILD → contains the fortran routines
- DATA → cross-sections, cooling tables...
- NETWORKS → all the pre-built networks
- PATCHES → 3D hydro-code patches source routines
- SOLVER → chemical solvers routines
- SRC → fortran sources routines/need to be pre-processed
- TESTS → contains the pre-built tests (ready to run)
- TOOLS → a series of useful tools (e.g. databases converter, for advanced users)

Sanity check

make sure what you downloaded is working

Before to start to work with KROME:

- prepare the "hello" test `./krome -test=hello`
- compile and run the test and check the results



NOTE: if you want to use gfortran `-compiler=gfortran`

Explore KROME
the first run

several ways to run KROME:

- from shell using in-line commands
- via the helper-script "wizard"
- by using an option file

KROME has internal check to verify the options

```
$ ./krome
*******************************
     WELCOME TO KROME
*******************************

ERROR: you must define -n FILENAME or -network
    FILENAME, where FILENAME is the reaction file!
```

The first run
using in-line commands

It requires you to know the options!

KROME help in unix format

```
$ ./krome -h
*******************************
     WELCOME TO KROME
*******************************

KROME a package for astrochemistry and microphysics

optional arguments:
  -h, --help show this help message and exit
  -ATOL ATOL set solver absolute tolerance to the
      float or double value ATOL
  -C create a simple C wrapper
  -n FILENAME reaction network file
  -network FILENAME same as -n
  -nochargeCheck skip reaction charge check
```

The first run
using the wizard

If you don’t know how to start you can use the "wizard" helper

```
$ ./wizard
**********************
 option wizard!
**********************
- Path of your chemical network
    [networks/react_COthin]:
```

The first run
using the wizard

It will ask questions about the possible options you can enable

```
$ ./wizard
**********************
 option wizard!
**********************
- Path of your chemical network
    [networks/react_COthin]:
    networks/react_primordial
added -n=networks/react_primordial

- Use number density (otherwise mass fractions)?
    [y]:
```

The first run
using the wizard


```
0) NONE (No cooling)
1) ATOMIC (Atomic from Cen 1992)
2) H2 (H2 from Glover+2007)
3) HD (HD from Lipovka+2007)
4) DH (Endothermic with thermochemical data)
5) DUST (Dust cooling)
6) H2GP98 (H2 from Galli+Palla 1998)
7) COMPTON (Compton)
8) EXPANSION (Isothermal expanding gas)
9) CIE (Collisional induced)
10) CONT (Continuum emission)
11) CHEM (Endothermic reactions)
...
21) FeII (FeII cooling)
- Cooling functions (use numbers above comma
    separated)? [0]:
```

The first run
using the wizard

- at the end of the process you will get the in-line commands to correctly run KROME
- additionally you will have an "options.kop" file in your folder

```
Your call to krome is:
$ ./krome -n=networks/react_primordial -useN
or use the option file
 options.kop
```

Note: during the school exercise strongly encouraged to use the option file!

The option file


```
-n networks/react_primordial
-useN
-cooling H2,ATOMIC
-heating CHEM
```

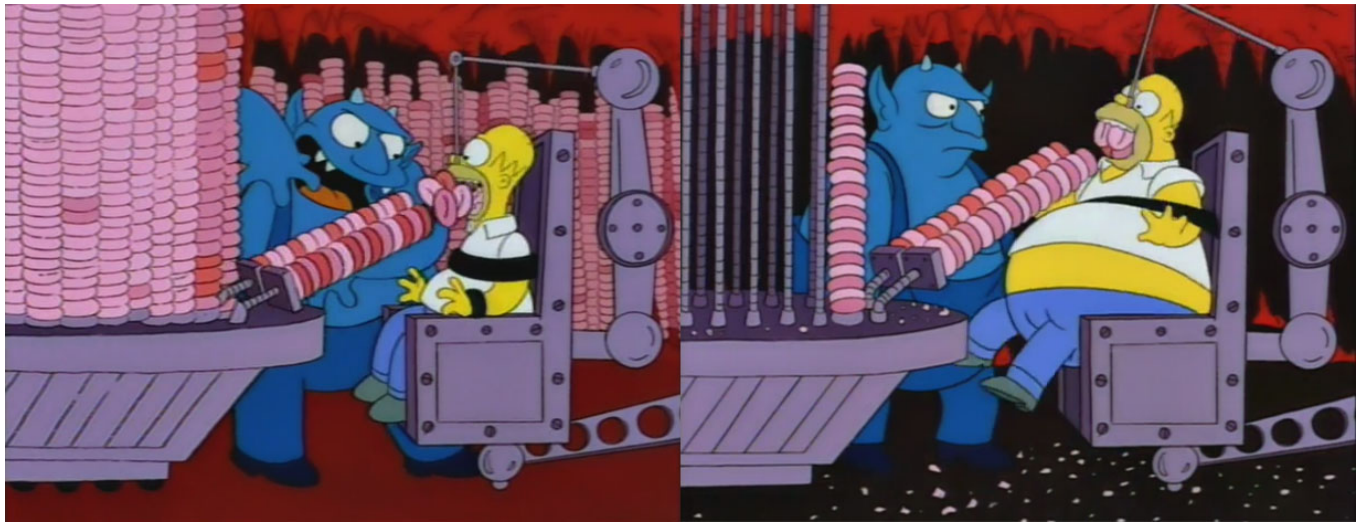
```
$ ./krome -options=options.kop
```

Summarizing

- use in-line commands (need to know the options, check -h, -help)
- use the wizard helper (much easier, core commands)
- use the option file provided into the KROME folder (follow the option_example)

Whatever you decide KROME will run!

KROME likes chemical networks!



Chemical Network files
quick overview

YOU NEED A CHEMICAL NETWORK

Three different approaches

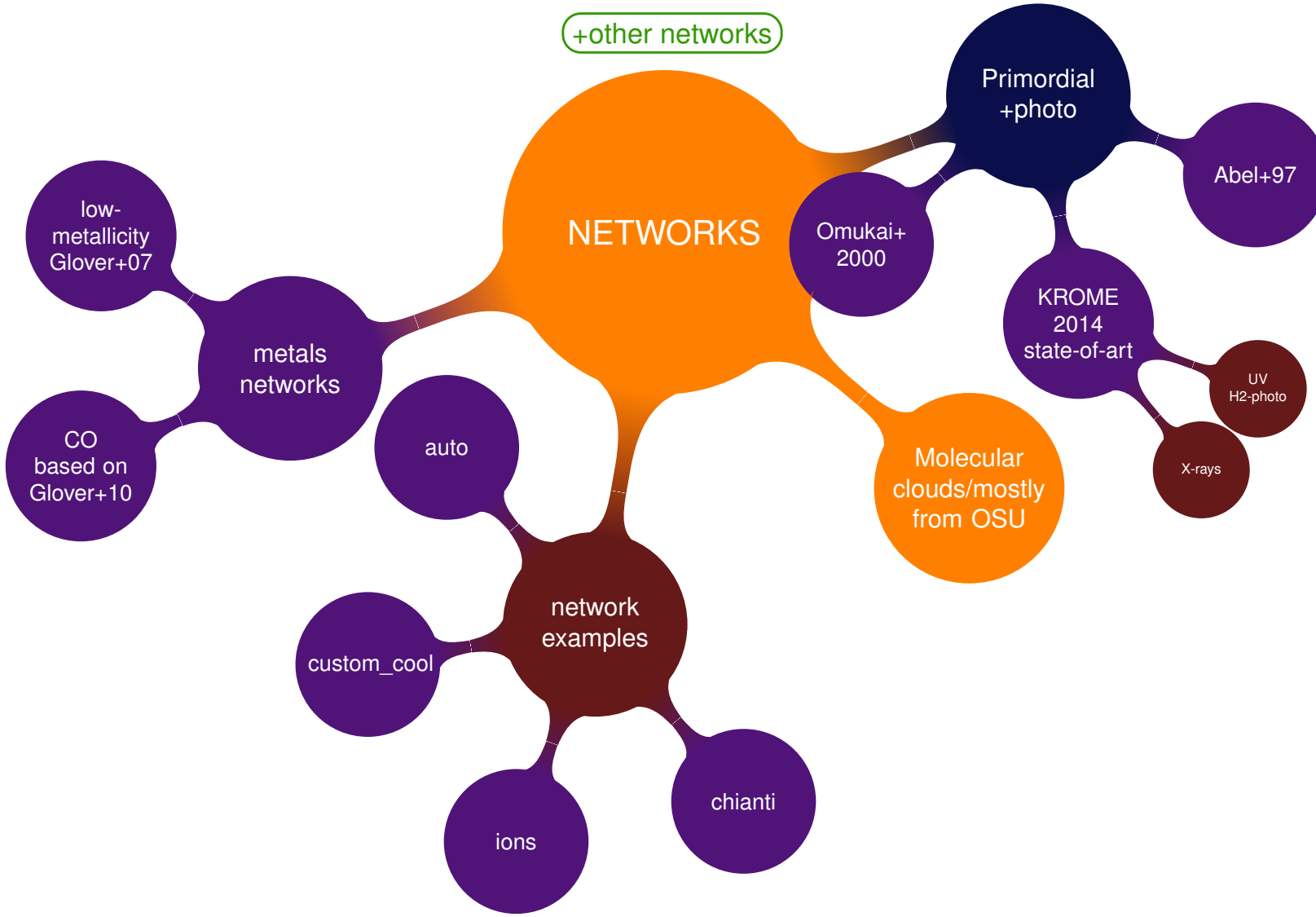
- use one of the pre-built KROME networks (under networks/ folder)
- use the KIDA $\rightarrow$ KROME converter tool to select reactions from standard database (advanced feature)
- prepare your own network based on databases, literature, and/or pre-existing networks

YOU NEED A CHEMICAL NETWORK

Three different approaches

- use one of the pre-built KROME networks (under networks/ folder)
- use the KIDA $\rightarrow$ KROME converter tool to select reactions from standard database (advanced feature)
- prepare your own networks based on databases, literature, and/or pre-existing networks

Pre-built networks
a graphical overview



Chemical Network files
quick overview

YOU NEED A CHEMICAL NETWORK

Three different approaches

- use one of the pre-built KROME networks (under networks/ folder)
- use the KIDA $\rightarrow$ KROME converter tool to select reactions from standard database (advanced feature)
- prepare your own network based on databases, literature, and/or pre-existing networks

YOU NEED A CHEMICAL NETWORK

Three different approaches

- use one of the pre-built KROME networks (under networks/ folder)
- use the KIDA $\rightarrow$ KROME converter tool to select reactions from standard database (advanced feature)
- prepare your own networks based on databases, literature, and/or pre-existing networks

Chemical network
without tokens

- KROME networks files are written in a simple Comma-Separated Values (CSV) format
- you can write simple networks without "tokens" by using the default format:

$$\mathrm{R_1 + R_2 + R_3 \xrightarrow{k[T_{min},T_{max}]} P_1 + P_2 + P_3 + P_4} \quad (3)$$

```
idx,R1,R2,R3,P1,P2,P3,P4,Tmin,Tmax,rate
```

Let's assume we want to implement the following network:

- $\mathrm{H_2^+ + H^- \rightarrow H + H_2}$
- $\mathrm{H_2^+ + H \rightarrow H^+ + H_2}$
- $\mathrm{H^- + H^+ \rightarrow e^- + H_2^+}$

Chemical network
without tokens

- $H_2^+ + H^- \rightarrow H + H_2 \quad k_1 = 5 \times 10^{-7}\sqrt{100/T_{gas}}$
- $H_2^+ + H \rightarrow H^+ + H_2 \quad k_2 = 6 \times 10^{-10}$
- $H^- + H^+ \rightarrow e^- + H_2^+ \quad k_3 = 10^{-8} T_{gas}^{-0.4}$

first_network

```
#Dalgarno & Lepp 1987
1,H2+,H-,,H,H2,,,1d1,1d8,5.d-7*sqrt(1.d2*invT)

//Karpas 1979
2,H2+,H,,H+,H2,,,NONE,NONE,6.0d-10

/*Poulart 1978*/
3,H-,H+,,E,H2+,,,NONE,NONE,1.d-8*Tgas**(-0.4)
```

Chemical network
advanced

With the use of the advanced "tokens" you can:

- change the format
- define variables and commons
- use auto photo reactions
- use your own functions
- define heating/cooling custom functions

See Tommaso's talk

Run KROME
analysing the out


```
$ ./krome -n networks/first_network.ntw
*******************************
     WELCOME TO KROME
*******************************

*****************************************************
WARNING: the folder build/ is not empty
 some items may be replaced. Do you want to
    proceed?
To avoid this message use -unsafe option.
*****************************************************
```

Run KROME
analysing the out

Network composed by: H_2^+, H_2, H^-, e^-, H^+, H

```
ODEs needed: 10
Reactions found: 3
Species found: 6
Species list saved in build/species.log
Species index initialization for gnuplot in
    build/species.gps
Heating cooling index init for gnuplot in
    build/heatcool.gps
Reactions saved in build/reactions.log
ODE partition: [6 atom/mols] + [1 CR] + [1
    PHOT] + [1 Tgas] + [1 dummy] = 10 ODEs
ODEs list: H2+, H, H2, H+, E, H-, CR, g,
    Tgas, dummy
```

Run KROME
analysing the out


```
Jacobian non-zero elements: 19 over 100
(19.0% of total elements, sparsity = 81.0%)
solver info:
 MF: 222
 MOSS+METH+MITER: 2+2+2
 LWM: 312 LRW: 422
 ...
 Prepearing files in /build...
- writing krome_commons.f90... done!
...
********************
Everything done, goodbye!
****************************************
41. COMPUTERS ARE LIKE OLD TESTAMENT GODS;
LOTS OF RULES AND NO MERCY
--- Joseph Campbell
****************************************
```

Access the build folder

files generated by KROME


```
Stefanos-MacBook-Air:build stefanobovino$ ls
Makefile                    krome_ode.f90               network.dot
README                      krome_photo.f90             opkda1.f
heatcool.gps                krome_reduction.f90         opkda2.f
krome.f90                   krome_stars.f90             opkdmain.f
krome_commons.f90           krome_subs.f90              reactions.log
krome_constants.f90         krome_tabs.f90              species.gps
krome_cooling.f90           krome_user.f90              species.log
krome_dust.f90              krome_user_commons.f90      test.f90
krome_heating.f90           list_user_functions.py
```

Look into the modules

krome_ode module


```
k(:) = coe_tab(n(:)) !compute coefficients

    !H2+
    dn(1) = &
        -k(1)*n(idx_H2j)*n(idx_Hk) &
        -k(2)*n(idx_H2j)*n(idx_H) &
        +k(3)*n(idx_Hk)*n(idx_Hj)

    !H-
    dn(2) = &
        -k(1)*n(idx_H2j)*n(idx_Hk) &
        -k(3)*n(idx_Hk)*n(idx_Hj)

    !H2
    dn(3) = &
        +k(1)*n(idx_H2j)*n(idx_Hk) &
        +k(2)*n(idx_H2j)*n(idx_H)
```

Look into the modules

krome_main module


```
!****************************
   !init DLSODES (see DLSODES manual)
   neq = nspec !number of eqns
   liw = size(iwork)
   lrw = size(rwork)
   iwork(:) = 0
   rwork(:) = 0.d0
   itol = 4 !both tolerances are arrays
   rtol(:) = 1.000000d-04 !relative tolerance
   atol(:) = 1.000000d-20 !absolute tolerance
   icount_max = 100 !maximum number of iterations
  itask = 1
   iopt = 0

   !MF=
   ! = 222 internal-generated JAC and sparsity
   ! = 121 user-provided JAC and internal generated
       sparsity
   ! = 22 internal-generated JAC but sparsity
       user-provided
   ! = 21 user-provided JAC and sparsity
   MF = 222
   !end init DLSODES
```

Additional useful files

In addition KROME creates the following files:

- species.log and reactions.log
  - the first one contains the list of species with their indexes (e.g. useful for initialization)
  - the second ones contains the list of reactions included in the network
- gnuplot scripts (to be used for testing, not in hydro-code)
  - heatcool.gps (to plot the cooling/heating contributions)
  - species.gps (to plot the evolution of the species)
- network.dot (next slides)

The species.log file


```
#This file contains a list of the species used with their
    indexes

1 H krome_idx_H
2 E krome_idx_E
3 H+ krome_idx_Hj
4 HE krome_idx_HE
5 HE+ krome_idx_HEj
6 HE++ krome_idx_HEjj
7 H- krome_idx_Hk
8 H2 krome_idx_H2
9 H2+ krome_idx_H2j
10 CR krome_idx_CR
11 g krome_idx_g
12 Tgas krome_idx_Tgas
13 dummy krome_idx_dummy
```

if you want to initialize H in your code

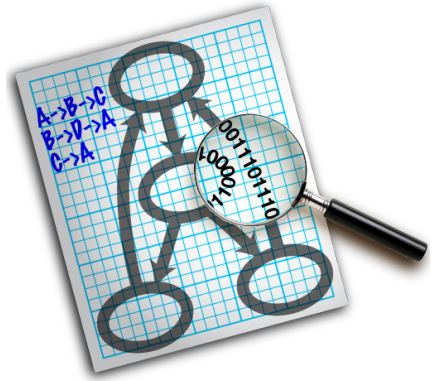
x(krome_idx_H) = HI_VARIABLE FROM YOUR CODE

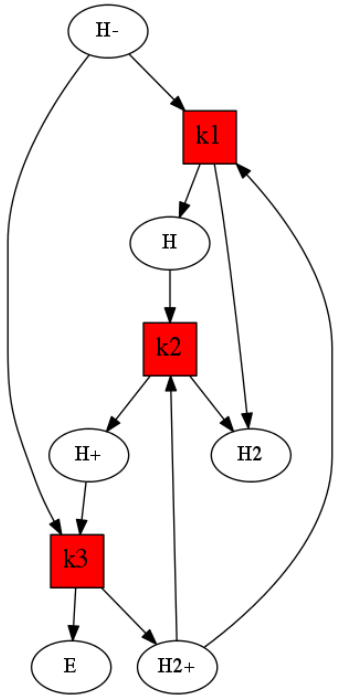
remember to include `use krome_user`

Nice tool
graphviz

Once KROME runs it generates a file "network.dot" which can be easily plotted with Graphviz[2].

```
dot -Tpng network.dot > output.png
```





- $H_2^+ + H^- \rightarrow H + H_2$
- $H_2^+ + H \rightarrow H^+ + H_2$
- $H^- + H^+ \rightarrow e^- + H_2^+$

[2] http://www.graphviz.org

Short summary
what we know and what we don't

What we know at this stage:

- how KROME is structured (folders/files etc.)
- how to run KROME (in-line commands, wizard, options file)
- how to prepare a simple network by using the default format
- how it looks the KROME output (at least partially)
- what is inside build (utilities/tools/routines)

What we don't know yet

- how to use the KROME-tokens to prepare a network (Tommaso's talk)
- how it is organized the `krome_user module` (Tommaso's talk)
- how to couple KROME with a "framework" code (Troels' and Daniel's talks)
- how to run a built-in test (next slides)

Built-in tests
overview

Under the folder "tests" you can find the following directories

- auto
- chianti
- cloud
- collapse
- collapseCO
- collapseUV
- collapseUV_Xrays
- collapseZ
- collapseZ_UV
- collapseZ_induced
- compact
- customCooling
- dust
- earlyUniverse
- hello
- lamda
- lotkav
- map
- reverse
- shock1D
- shock1Dcool
- shock1Dphoto
- slowmanifold
- stars
- wrapC

Built-in tests
overview

something to know

- mostly 0D and 1D
- some of the tests are stand-alone codes that can be used for applications, they are based on existing benchmark (e.g. Omukai+2000, Wakelam+2008, etc.)
- every test is described in the corresponding folder (look at the README file)
- not all the tests are at a stable stage (check http://www.kromepackage.org/test/)

To run a test with KROME

```
$ ./krome -test=NAME_OF_THE_TEST
```

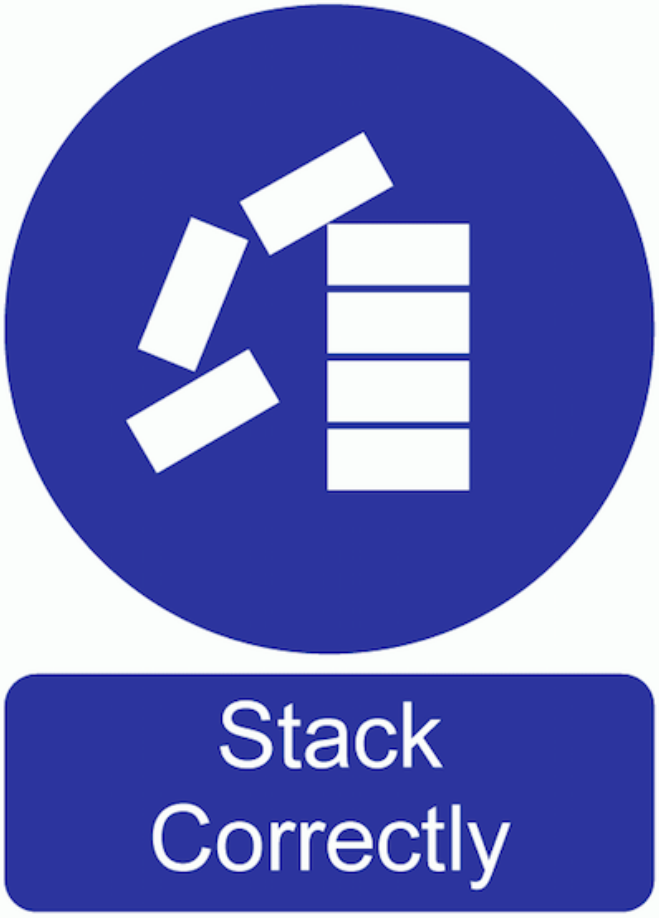
- Go into the "build", make and run!
- A gnuplot script is provided to plot the results, "load plot.gps".

Interactive practical example

Let's run a built-in test and analyse the output!

A small foretaste

on how to include KROME in your simulations



- enable options you need
- run KROME
- inlcude the krome modules
  - use krome
  - use krome_user
- call krome_init()
  - once and for all
- be sure to have initialized the species (make use of krome_nmols and other common arrays)
- change your Makefile accordingly
  - solver files (opkd*.F)
  - krome routine (`krome_all`)

Additional info

The information provided in this talk follow in part the Wiki page https://bitbucket.org/tgrassi/krome/wiki/

Thank you for your attention!