EUROPEAN CENTRE FOR RESEARCH AND ADVANCED TRAINING IN SCIENTIFIC COMPUTING

Tools for the numerical simulation with complex chemistry
Open-source code CANTERA

Tuesday 16th November 2020

Jonathan WIRTZ, Théo OGIER - PhD CERFACS

Organisation of the day

9h30 – 10h30 : Talk about Cantera (J.W)

10h30 – 12h30 : jupyter notebook tutorial (J.W + T.O)

12h30 – 14h : Lunch break (yourself :D)

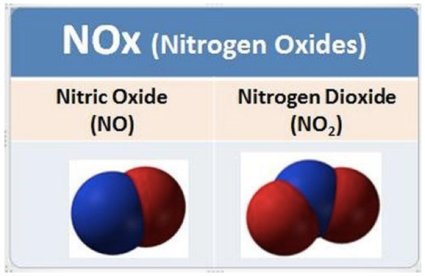
14h – 14h15 : Talk about cantera-avbp features (J.W)

14h15 – 17h : end of jupyter notebook tutorial + create your own script (J.W + T.O)

Everything will be on teams.

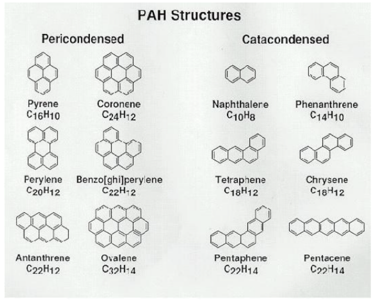
Why studying chemistry at CERFACS ?

Pollutants



NOx (Nitrogen Oxides)

Nitric Oxide (NO)	Nitrogen Dioxide (NO_2)



Chemistry driven processes

Ignition

Fuel addition

Content of the presentation

Content of the presentation

What is CANTERA ?

CANTERA is an **open-source suite of tools** for problems involving **chemical kinetics, thermodynamics, and transport processes.**

Multiple Interfaces
- **Python**
- Matlab
- C/C++
- Fortran 90.

Automative
Users can efficiently incorporate chemistry, thermodynamics and transport.

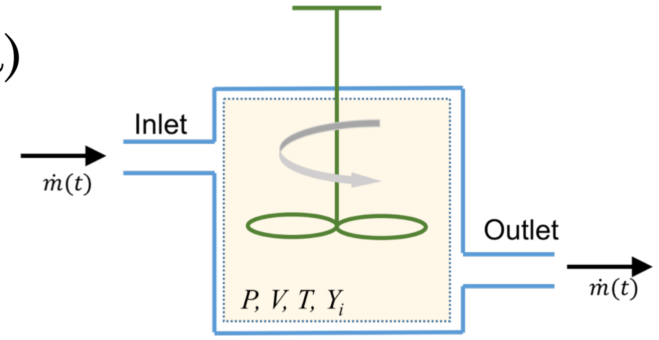
Object-Oriented
Can use different phase without too much change

Broad Applications
- **Combustion**
- Detonations
- Electrochemical energy
- Conversion and storage
- Fuel cells
- Batteries
- Aqueous electrolyte solutions
- Plasmas
- Thin film deposition

What can CANTERA do?

In terms of calculations …

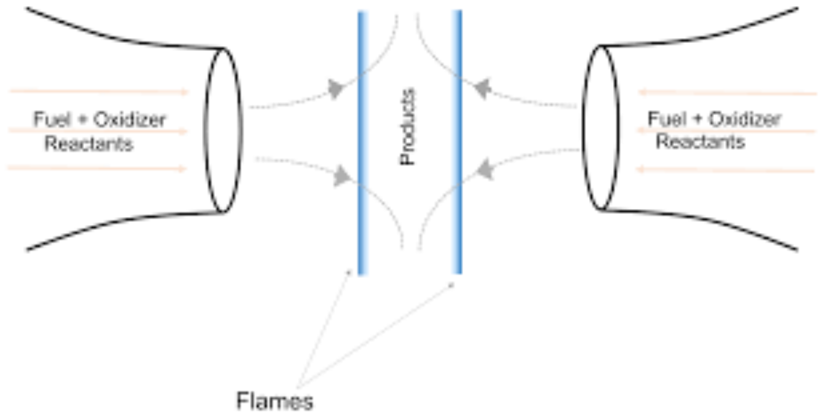
- Initialize a mixture (tutorial part 1)
- 0D (tutorial part 2)
  - Equilibrium state
- 0D with time evolution (tutorial part 3)
  - Constant Pressure/Volume batch reactor
  - Steady-state Plug Flow Reactor (PFR)



What can CANTERA do?

…

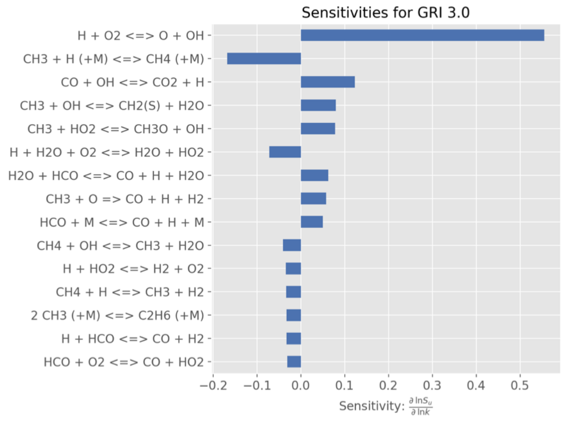
- 1D (tutorial part 4)
  - Burner stabilized and propagating premixed flat flame
  - Diffusion flame in counterflow configuration
  - Premixed flame in counterflow configuration (strained)



What can CANTERA do?

… but also in terms of analysis !

- Data extraction and post processing:
  - Temperature, Pressure, Mass Fractions …
  - Path Flow analysis, sensitivity analysis
- Add surface chemistry and heat losses …



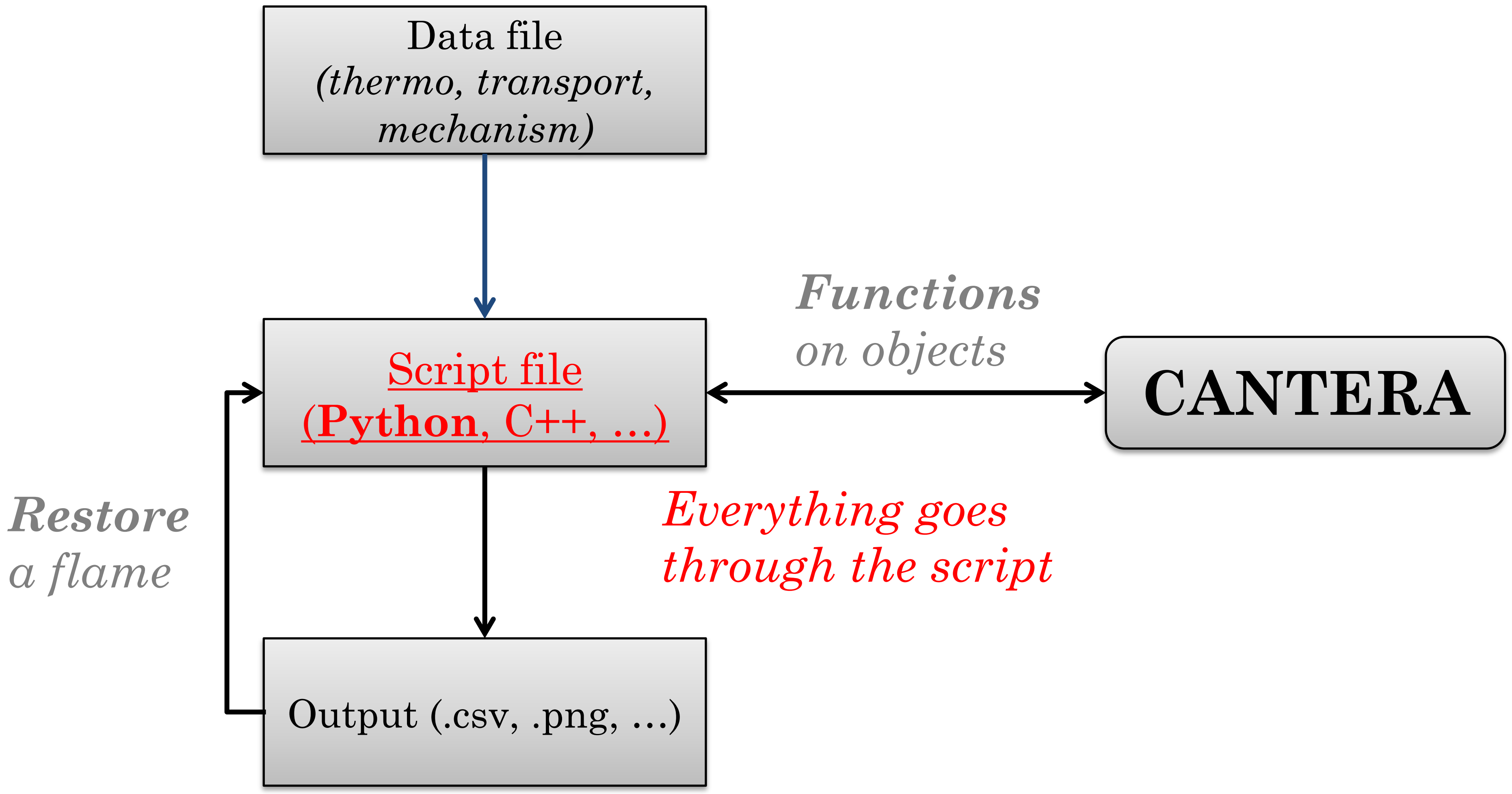
What is CANTERA ?

- **CANTERA's history**
  - We owe it all to **Dave Goodwin**, the original developer of Cantera whom made it available under BSD license.
  - He started off with an extensive overhaul of the CHEMKIN suite …
  - … to steer the modeling software towards an **object-oriented structure** with multiple interfaces (C++, Python, Matlab, FORTRAN).
  - Currently, **we are at the version 2.4.0** ***(but we will use version 2.3 in this training).***

What is CANTERA ?

- **CANTERA has the advantages of an object-oriented code (info in appendix):**
  - *Think of a Lego box* : it allows you to **form complex** kinetic/thermodynamic systems, or **networks from a set of « building blocks »**.
  - Each « building block » (or object) represents a well-defined small component of the global structure.
  - Those « building blocks » are defined and coded in C++, the core language of CANTERA …
  - **… But CANTERA provides a user-friendly interface in Python** (or Matlab, or FORTRAN)

What is CANTERA ?



Why is it a good choice ?

- ✓ CANTERA has **most of the functionality of the familiar CHEMKIN-II** + additional capabilities (non-ideal phases, multiphase equilibrium, electrochemistry…) See appendix to know more the difference between CANTERA and CHEMKIN.
- ✓ **CANTERA can (re)use CHEMKIN files** (libraries, mechanisms …).
- ✓ Users can interface with Cantera through Python, C++, the Matlab toolbox and Fortran as well : **no language excuse !**
- ✓ Those interfaces are only front ends; calculations are done in an optimized, compiled code that is really efficient and fast.
- ✓ **Basically, it is possible to create anything that pops into your head …**

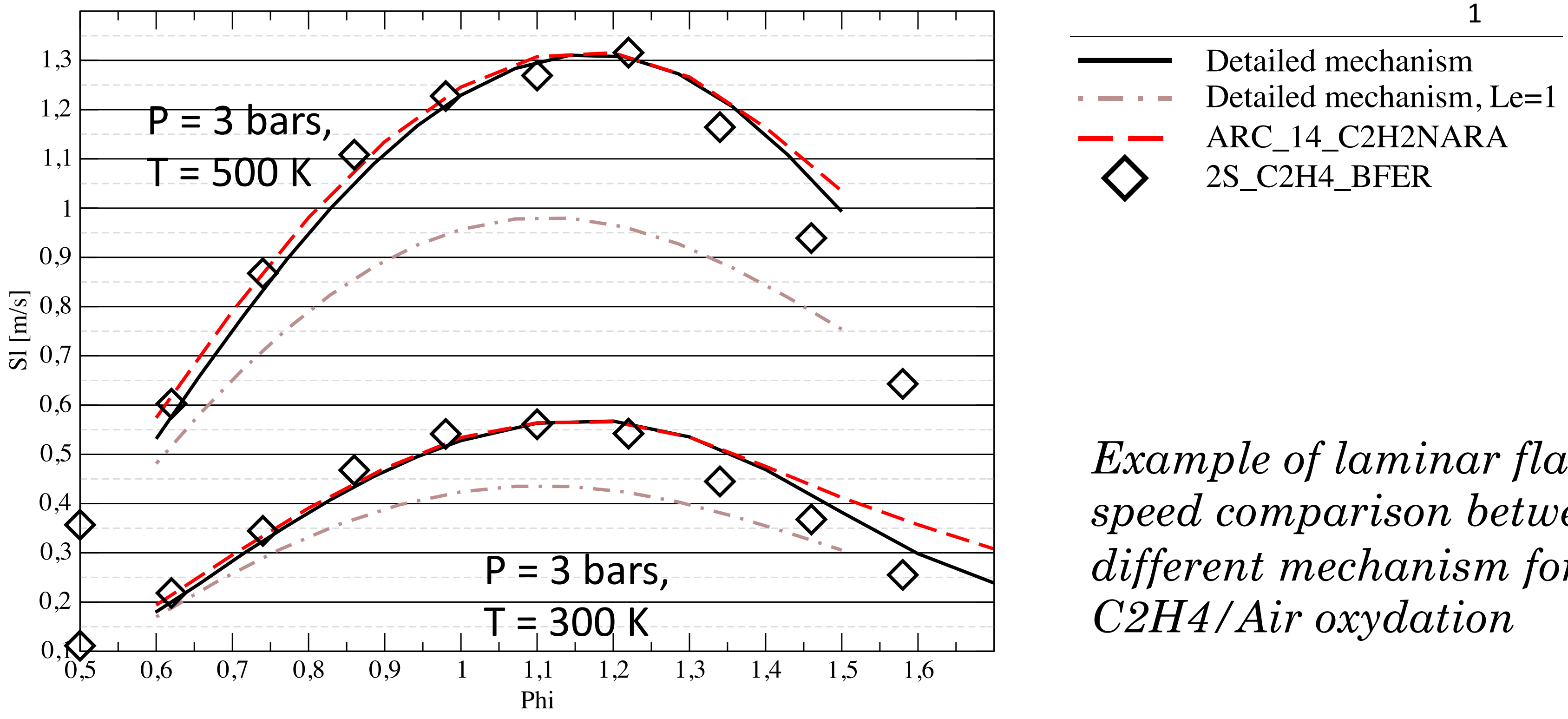
Why is it a good choice ?

- **On the downside :** CANTERA does not provide any robust documentation. Basically, everything that exists can be accessed through : http://www.cantera.org
- ✓ **But** there is **a big community of users**, they will help solve all bugs and scripting issues : **cantera google group**
- ✓ **CANTERA comes with a series of scripts and examples** in all of the interface languages : currently **in the folder 'samples'**

How will it be helpful to you ?

1. Validation phase

✓ **Compare different chemistries with CANTERA**, including skeletal and reduced chemistries, in a fast and efficient way.



Example of laminar flame speed comparison between different mechanism for C2H4/Air oxydation

[1] K. Narayanaswamy, G. Blanquart, H. Pitsch "A consistent chemical mechanism for oxidation of substituted aromatic species ". Combustion and Flame, Vol.157 pp. 1879–1898, 2010

How will it be helpful to you ?

1. Validation phase

✓ **Simulate chemistry-related combustion features** directly

- **Evaluate the impact of simplified transport model** (constant Schmidt number, fitted viscosity …)
- **Develop and test various combustion models**. At CERFACS, we use a flame front thickening option (DTFLES) in our CFD code AVBP
- **Evaluate the effects of strain on flame behavior**: axisymmetric stagnation flows reduce to pseudo 1D simulations (counterflow flames)

How will it be helpful to you ?

1. **Validation phase**
2. **Reduce some chemistries**

✓ **Reduce chemistries**
  - **Implement your own via the main script** (Sensitivity Analysis, …)
  - **Extract relevant data for post processing** with reduction tools (ARCANE)

How will it be helpful to you ?

1. **Validation phase**
2. **Reduce some chemistries**
3. **Initialization phase for other codes**

- ✓ Use 1D profiles to initialize a simulation (CFD codes such as AVBP at CERFACS)
- ✓ Use 1D profiles to generate flamelet tables for use in CFD codes (Mutagen at CORIA)

Content of the presentation

Content of the presentation

Equilibrium calculations

“The equilibrium state is that corresponding to ***a minimum of a property called the energy function*** *under specified conditions”*

Cantera related pages : https://cantera.org/tutorials/python-tutorial.html / https://cantera.org/tutorials/cxx-guide/equil-example.html

Governing equations
Equilibrium equations

"The equilibrium state is that corresponding to a minimum of a property called the energy function under specified conditions"

- Several methods exist, based on different energy functions.
- Usually, **the Gibbs free energy function is used, with two constant quantities**
- 2 different methods exist :
  - **Non-stoichiometric methods** where the conservation of mass is treated separately
  - **Stoichiometric methods**, more robust but slower
- CANTERA tries a non-stoichiometric method first, and turns to a stoichiometric method (VCS algorithm[2]) if it fails.

[1] C.H.Wong ,"Chemical equilibrium analysis of combustion proudcts at constant volume", 2001

[2] Smith, W.R. and Missen, R.W. "Chemical Reaction Equilibrium Analysis: Theory and Algorithms", 1982

The non-stoichiometric method

Once the λ_l are determined, since **T & P are constant**, the mole fractions are automatically deduced.

$$\boxed{\mu_k = \sum_{l=1}^{L} \lambda_l n_{kl} \Rightarrow X_k = \frac{P_o}{P} \exp\left(-\frac{g_k^0(T)}{RT} + \sum_{l=1}^{L} n_{kl} \frac{\lambda_l}{RT}\right)}$$

- **General procedure (see appendix for details)** :
- The g_k^0 are tabulated.
- The user provides a guess for enough (L)X_k - with the knowledge that $\sum_{k=1}^{L} X_k = 1$
- The chemical potential of the elemental species per atom λ_l are calculated (see appendix).
- The unknown X_k are calculated with those estimated λ_l and $\sum_{k=1}^{L} X_k = 1$ is evaluated.
- If $\sum_{k=1}^{L} X_k = 1$ is « too far » from 1, a new guess for the X_k is provided and the procedure reiterates with well chosen (L)X_k.

(Note: no need to provide reactions information !)

Adiabatic flame temperature

Equilibrium calculation are also possible maintaining other quantities constant
Ex: **H and P constant, through a loop on T**

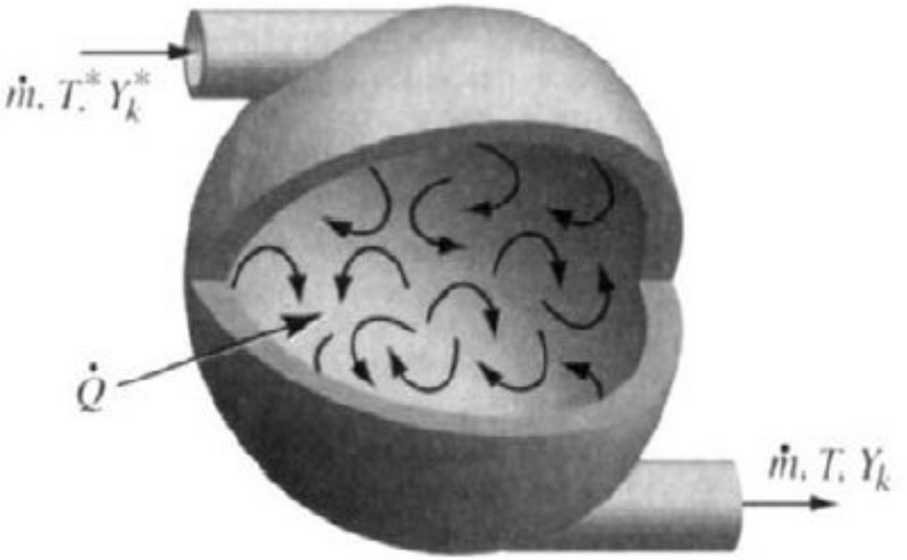
$$\boxed{\begin{array}{l} X_k = \dfrac{P_o}{P} \exp\left(-\dfrac{g_k^0(T)}{RT} + \displaystyle\sum_{l=1}^{L} n_{kl} \dfrac{\lambda_l}{RT}\right) \\ \textit{and} \\ \Delta H = 0 \end{array}}$$

Using H at initial conditions

$$H = \int_{T_0}^{T} C_p dT + \sum_{k=1}^{K} X_k H_{f,k}^{0,m}, \quad C_p = \sum_{k=1}^{K} X_k C_{p,k}^{m}$$

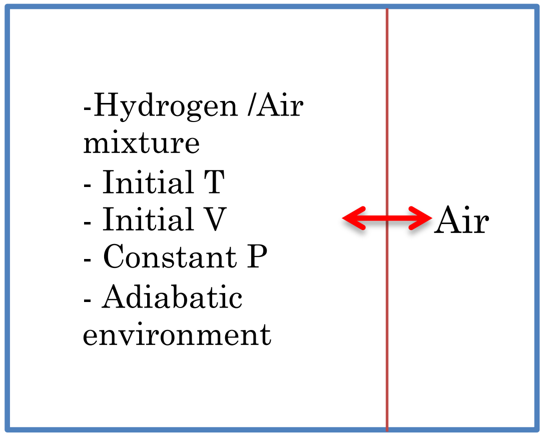
→ Gives the **gas composition** and the **adiabatic flame temperature.**

Batch and Stirred Reactor equations[3]



Conceptual illustration of a continuously stirred tank reactor.

Initial state



[3] Picture from Robert J. Kee, Michael E. Coltrin and Peter Glarborg ,"Chemically Reacting Flow: Theory and Practice", 2003 – chap 16
Cantera related page : https://cantera.org/science/reactors.html

Governing equations
Batch and Stirred Reactor[3][4]

The **state variables** for Cantera's general reactor model are

- m, the mass of the reactor's contents
- V, the reactor's volume
- U, the total internal energy of the reactors contents
- Yk , the mass fractions for each species

- Reactor Volume:

$$\boxed{\frac{dV}{dt} = \sum_w f_w A_w v_w(t)}$$

Where :
- A_w is the wall Area
- v_w(t) is the velocity of the wall
- $f_w = \pm 1$ is the wall facing

- Mass Conservation:

$$\boxed{\frac{dm}{dt} = \sum_{in} \dot{m}_{in} - \sum_{out} \dot{m}_{out} + \dot{m}_{wall}}$$

Where $\dot{m}_{in,out}$ are inlets and outlets mass flow rate and $\dot{m}_{wall}$ stands for the production of homogeneous phase species on the reactor walls.

[3] Robert J. Kee, Michael E. Coltrin and Peter Glarborg ,"Chemically Reacting Flow: Theory and Practice", 2003 – chap 16
[4] http://cantera.github.io/docs/sphinx/html/reactors.html

Governing equations
Batch and Stirred Reactor[3][4]

The **state variables** for Cantera's general reactor model are

- m, the mass of the reactor's contents
- V, the reactor's volume
- U, the total internal energy of the reactors contents
- Yk , the mass fractions for each species

- Species conservation:

$$\boxed{\frac{d(mY_k)}{dt} = \sum_{in} \dot{m}_{in} Y_{k,in} - \sum_{out} \dot{m}_{out} Y_k + \dot{m}_{k,gen}}$$

Where

$\dot{m}_{k,gen}$ is the rate at which species k is generated through homogeneous phase reactions in the reactor and on the walls

- Energy Conservation (*reactor case*) :

$$\boxed{\frac{dU}{dt} = -p\frac{dV}{dt} - \dot{Q} + \sum_{in} \dot{m}_{in} h_{in} - h\sum_{out} \dot{m}_{out}}$$

Where $\dot{Q}$ is the total rate of heat transfer through all walls

[3] Robert J. Kee, Michael E. Coltrin and Peter Glarborg ,"Chemically Reacting Flow: Theory and Practice", 2003 – chap 16
[4] http://cantera.github.io/docs/sphinx/html/reactors.html

Governing equations
Batch and Stirred Reactor[3][4]

Resulting system of Ordinary Differential Equations (ODE) :

- Usually stiff
- Integrated for a user-specified timestep by the Sundials' solver CVODE.

→ **Gives the temporal evolution of the quantities inside a vessel**

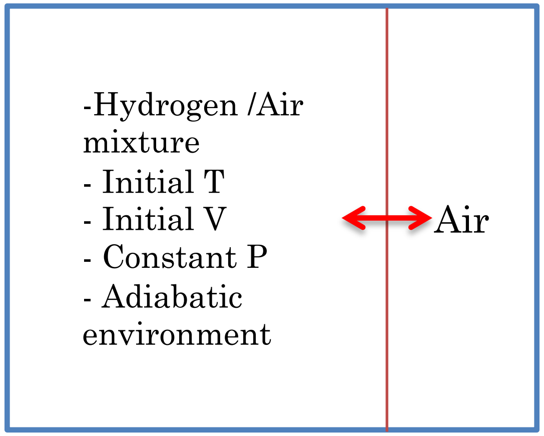
[3] Robert J. Kee, Michael E. Coltrin and Peter Glarborg ,"Chemically Reacting Flow: Theory and Practice", 2003 – chap 16

[4] http://cantera.github.io/docs/sphinx/html/reactors.html

Governing equations
Batch and Stirred Reactor[3][4]

The constant pressure batch reactor

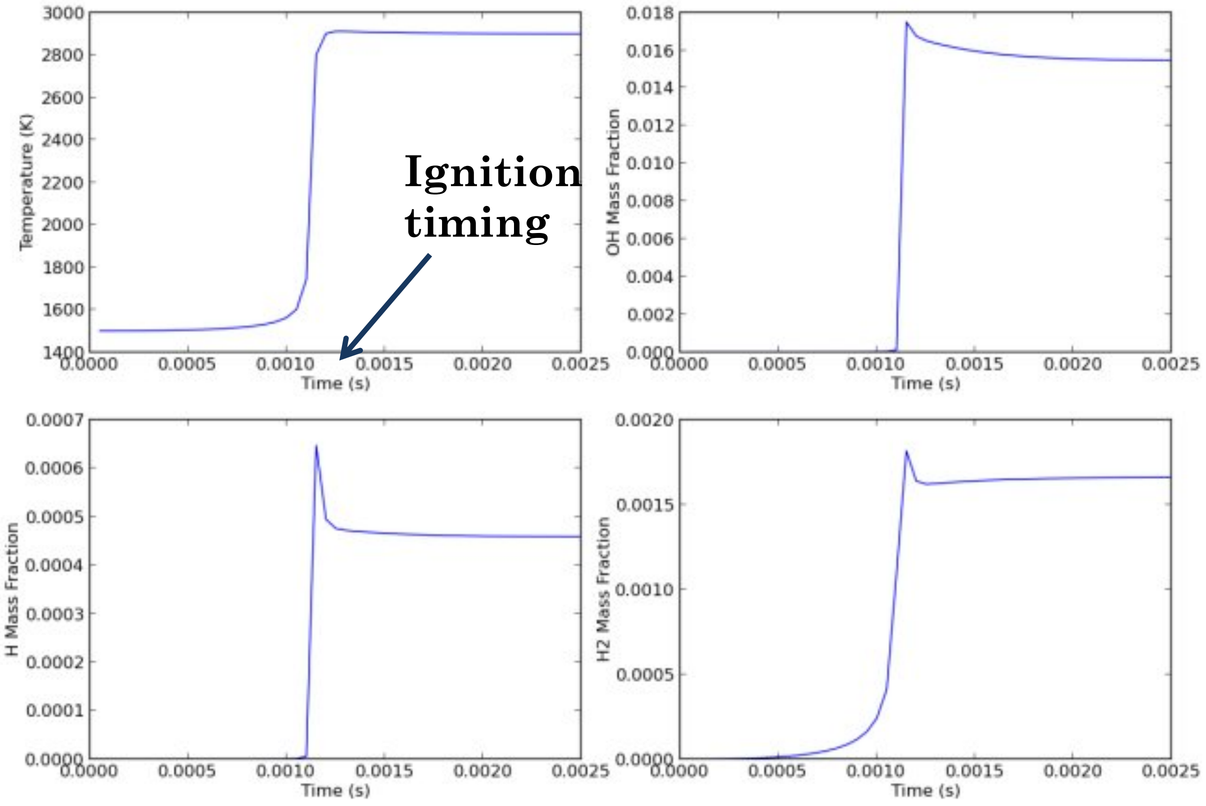
Initial state



Separating moveable wall, to keep a constant P

A « piston-like » system

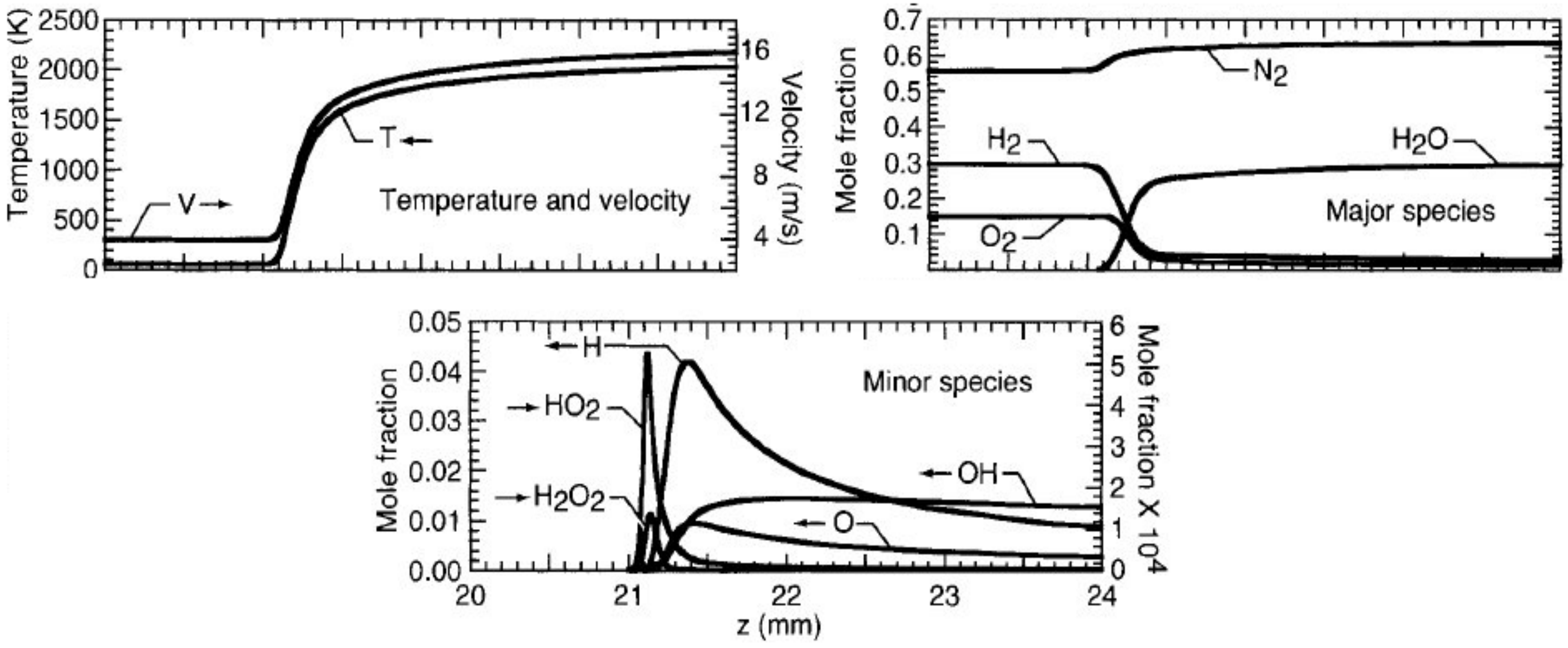
Time evolution of the state with CANTERA



[3] Robert J. Kee, Michael E. Coltrin and Peter Glarborg ,"Chemically Reacting Flow: Theory and Practice", 2003 – chap 16
[4] http://cantera.github.io/docs/sphinx/html/reactors.html

1D computations equations

Example : Laminar premixed flame equations



Computed solution to an atmospheric-pressure, freely propagating, stoichiometric, premixed, hydrogen-air, flat flame.

Picture from Robert J. Kee, Michael E. Coltrin and Peter Glarborg ,"Chemically Reacting Flow: Theory and Practice", 2003 – chap 3
Cantera related page : https://cantera.org/science/flames.html

Governing equations
Laminar Premixed Flame

For a steady laminar premixed flame, **we consider 3+K equations, derived from the general conservation equations for perfect gas** in cylindrical coordinates[3]:

Steady-state version

State
$$\rho = \frac{p\bar{W}}{RT}$$

Continuity
$$\frac{\partial \rho u}{\partial z} = 0$$

Species
$$\rho \cancel{\frac{\partial Yk}{\partial t}} + \rho u \frac{\partial Yk}{\partial z} = -\frac{\partial j_{k,z}}{\partial z} + \dot{\omega} Wk$$

Energy
$$\rho c_p \cancel{\frac{\partial T}{\partial t}} + \rho u c_p \frac{\partial T}{\partial z} = \frac{\partial}{\partial z}\left(\lambda \frac{\partial T}{\partial z}\right) - \sum_{k=1}^{K} c_{pk} j_{k,z} \frac{\partial T}{\partial z} - \sum_{k=1}^{K} h_k \dot{\omega}_k W_k$$

$\bar{W}$, is the *mean molecular weight* of the mixture
$\dot{\omega}_k$, is the *molar production rate* of species k
$j_{k,z}$, is the *diffusive mass flux vector* of species k
c_p, is the *mass averaged specific heat*
$c_{p,k}$, is the *specific heat of species k*
λ, is the *thermal conductivity*
z, is the *spatial coordinate*

[3] Robert J. Kee, Michael E. Coltrin and Peter Glarborg ,“Chemically Reacting Flow: Theory and Practice”, 2003 – chap 16

Numerical Resolution

The ***steady state*** version of the previous equations (1D flames) are discretized on a generally **non uniform mesh**, if needed.

$$\left[\frac{dT}{dx}\right]_j = \frac{T_j - T_{j-1}}{x_j - x_{j-1}}$$

Example with the windward difference in space

Let's introduce

– **the discretized solution vector y** (density, velocity, mole fractions and temperature at each space point)
– **the equation vector F** (the 3+K conservation equations).
– **the discretized residual F(y)** at each space point.

Governing equations
Numerical Resolution

The computational problem consists of **finding the smallest residual F**, (at all space points j …).

$$F(y) = 0$$

with $y = (solution\_vector\_pt1, \ldots, solution\_vector\_ptJ)$

A Damped modified Newton solver with internal time integration is used
(but then, transient terms are needed)

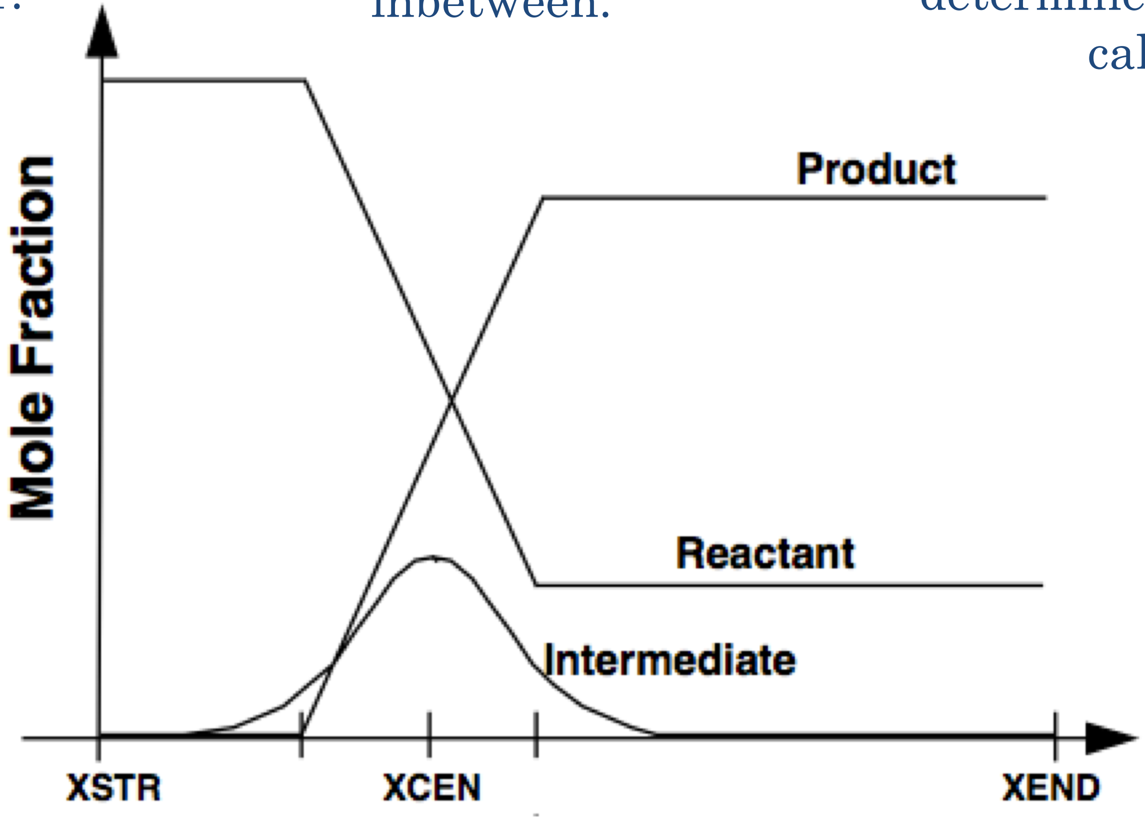
Convergence relies on the **initial estimate, y^0, usually provided by equilibrium calculation** at the inlet temperature T_1^0 and pressure P_1^0.

An initial y^0 is required.

Inlet composition, set by user.

Linear profile, inbetween.

Outlet composition, determined by equilibrium calculation.



Governing equations
Numerical Resolution

A Damped modified Newton solver with internal time integration

The **Newton solver** is : the method used by Cantera to solve the system and try to find the solution.

It is **damped** because a damping coefficient is added to help for the convergence.

It is **with internal integration** as an “artificial temporal term” is added to help for the convergence if the damping failed.

The Newton solver

What we seek at point m is $F(y) = 0$

which is used to iterate $\dfrac{\partial F}{\partial y} = \dfrac{0 - F(y^m)}{y^{m+1} - y^m}$

or in other words

$$\boxed{y^{m+1} = y^m - \left[\frac{\partial F}{\partial y}\right]^{-1}_{y^m} F(y^m)}$$

– **Convergence is reached when** $\Delta y^m = y^{m+1} - y^m$ **becomes negligibly small.**

– The mesh might be automatically refined in the region of high gradients

The **Damped modified** Newton solver

Evaluating the Jacobian matrices

$$\left(\frac{\partial F}{\partial y}\right)_{y^m} = J^m$$

is a **time consuming process**, so it is not done at each iteration

The solver evaluates the Jacobian each specified number of iteration step and inbetween

Scaling parameter

$$(J^m)^{-1} = \lambda^m \left(\frac{\partial F}{\partial y}\right)_{y^k}^{-1} = \lambda^m (J^k)^{-1}; \quad 1 < k \le m \quad 0 < \boxed{\lambda^m} \le 1$$

so that the problem becomes

$$\boxed{(J^k)\Delta y^m = -\lambda^m F(y^m)}$$

The **Damped modified** Newton solver

$$F(y) = 0$$

$$(J^k)\Delta y^m = -\lambda^m F(y^m)$$

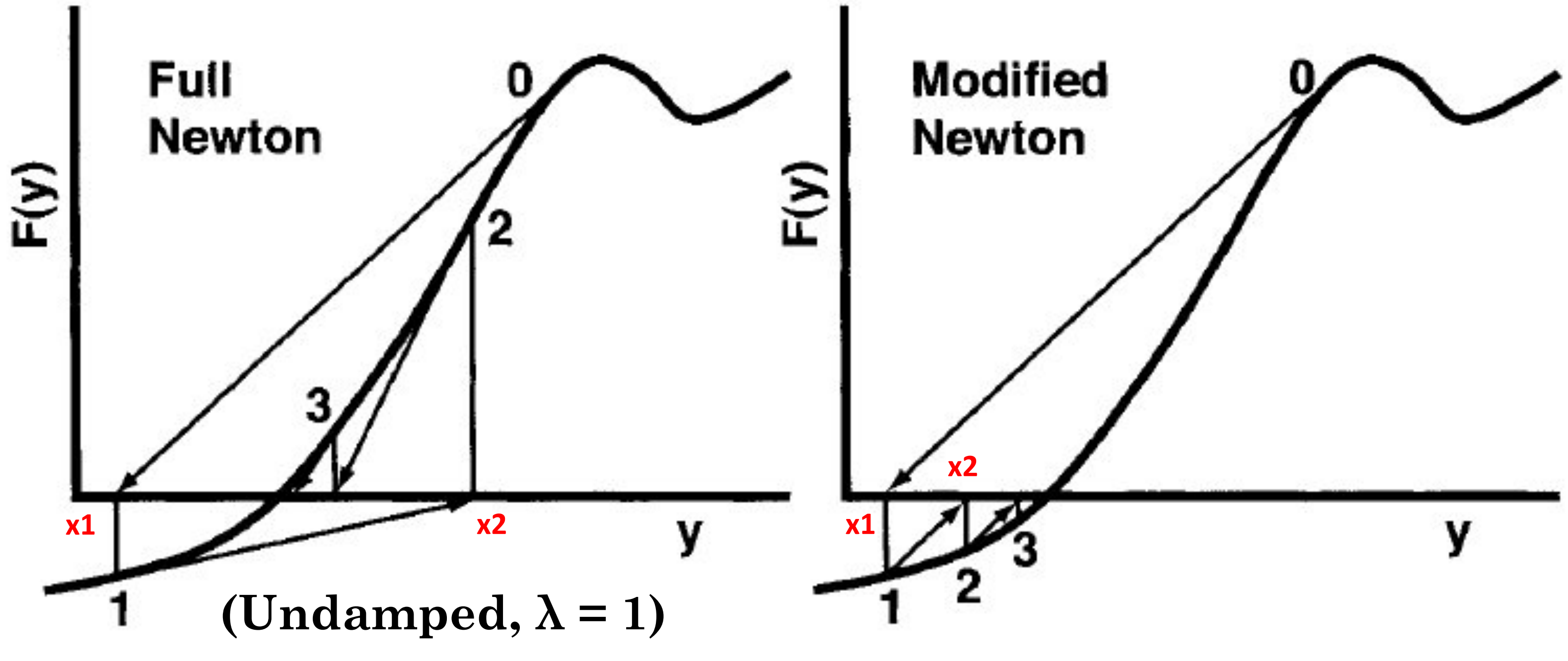


Illustration of the full Newton and modified Newton algorithm

The **Damped modified** Newton solver

$$\boxed{(J^k)\Delta y^m = -\lambda^m F(y^m)}$$

The λ^m must satisfy the condition that the subsequent undamped step be smaller.

$$\boxed{\begin{array}{l} \left|\Delta y^{m+1}\right| \le \left|\Delta y^m\right| \\ \left|(J^m)^{-1} F(y^{m+1})\right| \le \left|(J^m)^{-1} F(y^m)\right| \end{array}}$$

If not verified $\left\{ \begin{array}{l} \text{another damping parameter is tested} \\ \text{a new Jacobian is computed} \end{array} \right.$

set_max_jac_age

The Damped modified Newton solver with internal time integration

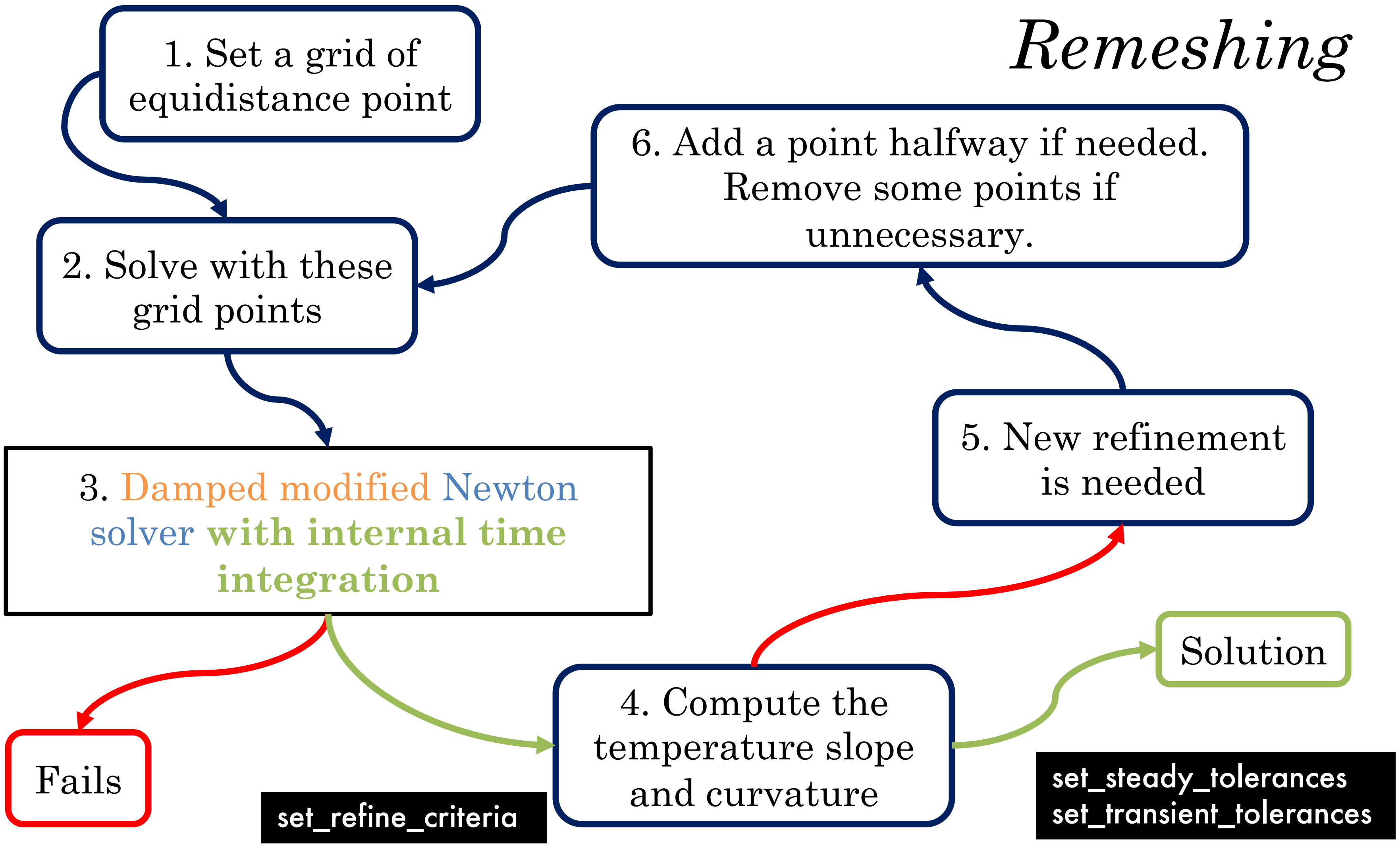
Whenever both damping parameters and the new Jacobian fail :

- ***Transient equations*** are used $$F(y_{t=n+1}) = \frac{y_{t=n+1} - y_{t=n}}{\Delta t}$$
- Time derivatives are approximated by **first-order, implicit backwards finite differences schemes**.
- Time steps are specified by the user $\Delta t = h$
- Same Newton method to solve the system of equations **for each time step** $$G(y) = F(y) - \frac{dy}{dt} = 0 \qquad \boxed{\left(J - \frac{I}{h}\right)(y_{n+1} - y_n) = -G(y_n)}$$
- The new y is used **as a new starting estimate** for the steady state problem

set_time_step

Governing equations
Numerical Resolution

Remeshing


1. Set a grid of equidistance point
2. Solve with these grid points
3. Damped modified Newton solver with internal time integration
Fails
4. Compute the temperature slope and curvature
Solution
5. New refinement is needed
6. Add a point halfway if needed. Remove some points if unnecessary.

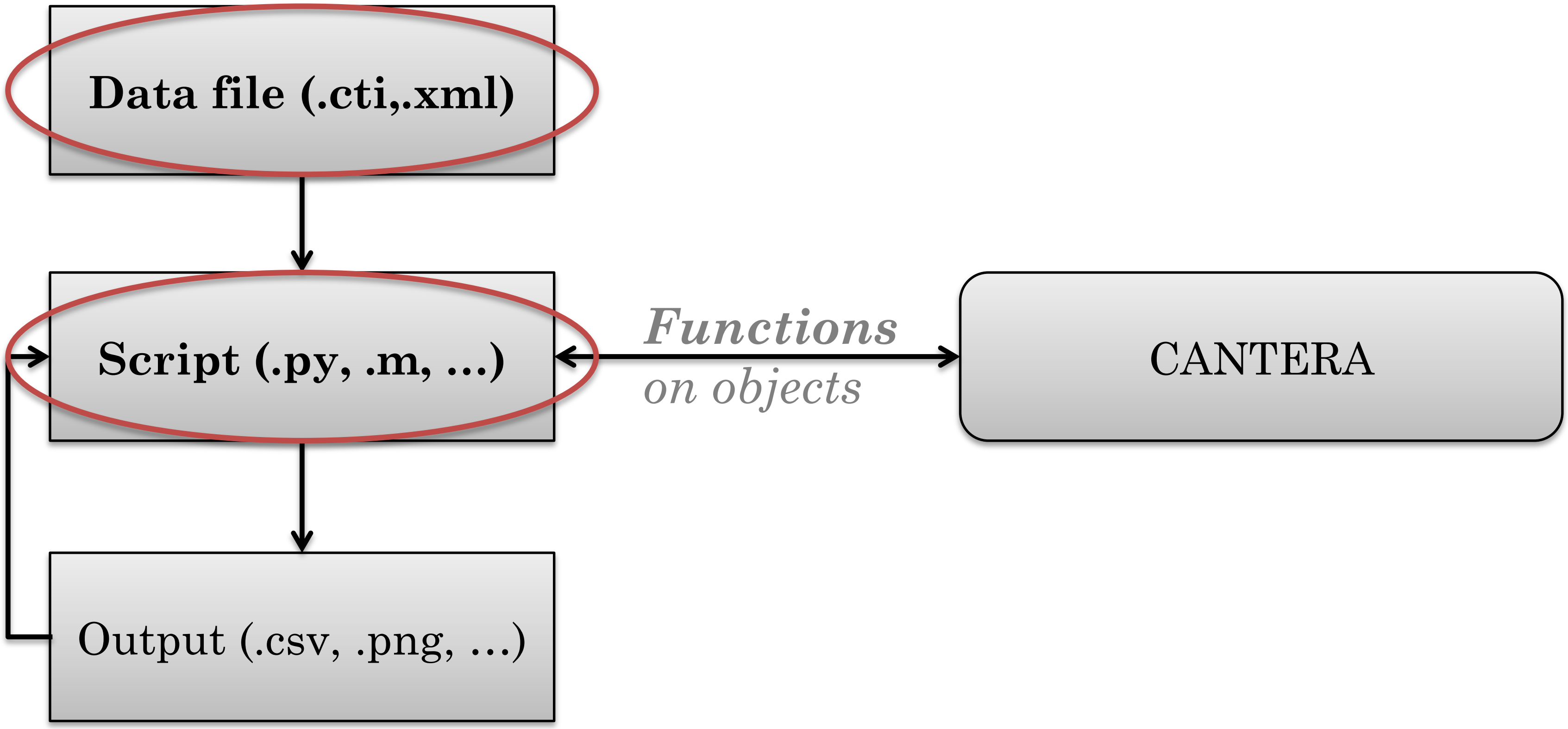
```
set_refine_criteria
```

```
set_steady_tolerances
set_transient_tolerances
```

Content of the presentation

Practical use

What do you need to run Cantera ?



Practical use

- Where do you find the data file ?
  - From CHEMKIN (‘mech.inp’, ‘therm.dat’ & ‘trans.dat’) via the tool ck2cti.
  - From a dataset provided with CANTERA (air, GRIMech, …).
  - You can generate it by hand.
- And the script ?
  - You generate it from « building blocks ».

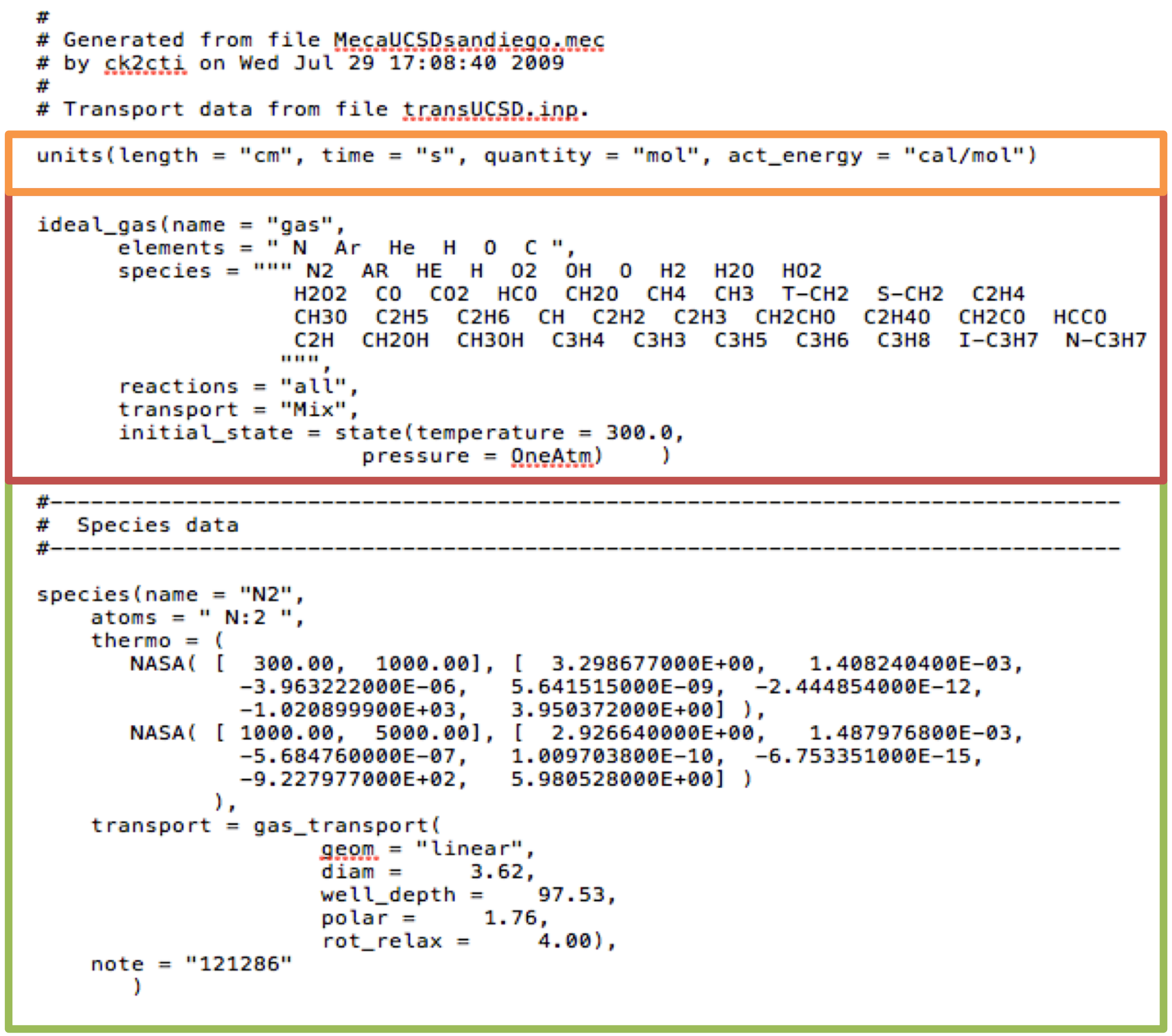
What is a data file (format '.cti') ?

It contains information about :

- **Phases and interfaces** (species involved, thermo and transport models ...)
- **Elements and species data**
- **Reaction data** (expression, rate coefficients, pressure dependence, ...)
- Appendix gives you information about technical keywords

Practical use

Data file



```
#
# Generated from file MecaUCSDsandiego.mec
# by ck2cti on Wed Jul 29 17:08:40 2009
#
# Transport data from file transUCSD.inp.

units(length = "cm", time = "s", quantity = "mol", act_energy = "cal/mol")

ideal_gas(name = "gas",
      elements = " N  Ar  He  H  O  C ",
      species = """ N2  AR  HE  H  O2  OH  O  H2  H2O  HO2
                   H2O2  CO  CO2  HCO  CH2O  CH4  CH3  T-CH2  S-CH2  C2H4
                   CH3O  C2H5  C2H6  CH  C2H2  C2H3  CH2CHO  C2H4O  CH2CO  HCCO
                   C2H  CH2OH  CH3OH  C3H4  C3H3  C3H5  C3H6  C3H8  I-C3H7  N-C3H7
                  """,
      reactions = "all",
      transport = "Mix",
      initial_state = state(temperature = 300.0,
                        pressure = OneAtm)     )

#-------------------------------------------------------------------------------
#  Species data
#-------------------------------------------------------------------------------

species(name = "N2",
    atoms = " N:2 ",
    thermo = (
       NASA( [  300.00,  1000.00], [  3.298677000E+00,   1.408240400E-03,
               -3.963222000E-06,   5.641515000E-09,  -2.444854000E-12,
               -1.020899900E+03,   3.950372000E+00] ),
       NASA( [ 1000.00,  5000.00], [  2.926640000E+00,   1.487976800E-03,
               -5.684760000E-07,   1.009703800E-10,  -6.753351000E-15,
               -9.227977000E+02,   5.980528000E+00] )
             ),
    transport = gas_transport(
                     geom = "linear",
                     diam =      3.62,
                     well_depth =     97.53,
                     polar =      1.76,
                     rot_relax =     4.00),
    note = "121286"
       )
```

Units

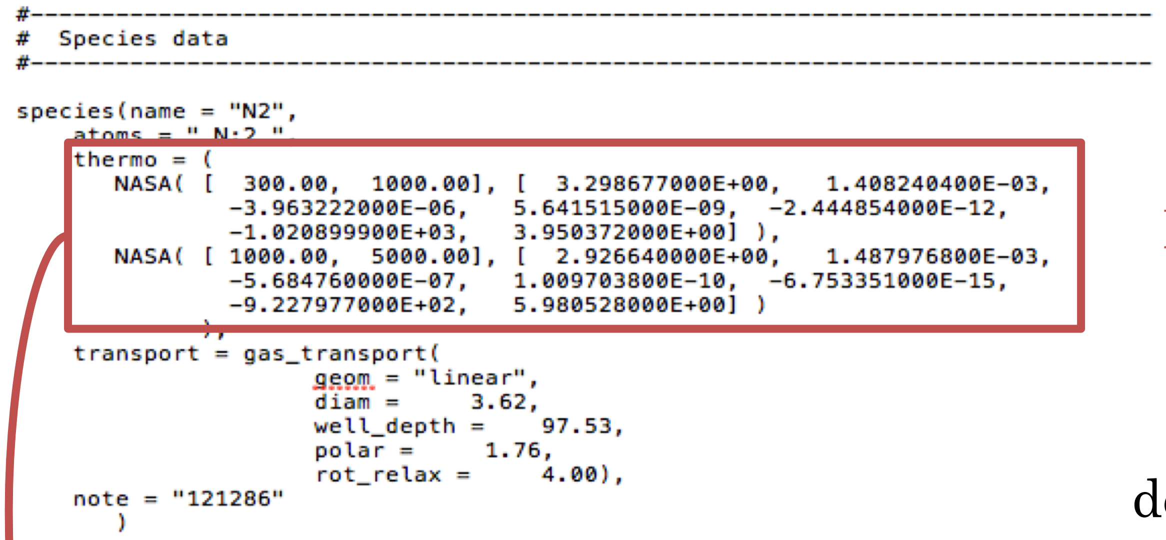
Phase data

Species N_2 data

thermo

transport

Practical use
Data file



```
#-------------------------------------------------------------------------------
#  Species data
#-------------------------------------------------------------------------------

species(name = "N2",
    atoms = " N:2 ",
    thermo = (
       NASA( [  300.00,  1000.00], [  3.298677000E+00,   1.408240400E-03,
                -3.963222000E-06,   5.641515000E-09,  -2.444854000E-12,
                -1.020899900E+03,   3.950372000E+00] ),
       NASA( [ 1000.00,  5000.00], [  2.926640000E+00,   1.487976800E-03,
                -5.684760000E-07,   1.009703800E-10,  -6.753351000E-15,
                -9.227977000E+02,   5.980528000E+00] )
             ),
    transport = gas_transport(
                     geom = "linear",
                     diam =      3.62,
                     well_depth =     97.53,
                     polar =     1.76,
                     rot_relax =     4.00),
    note = "121286"
       )
```

NASA7

There can be other definitions but this one is the most popular.

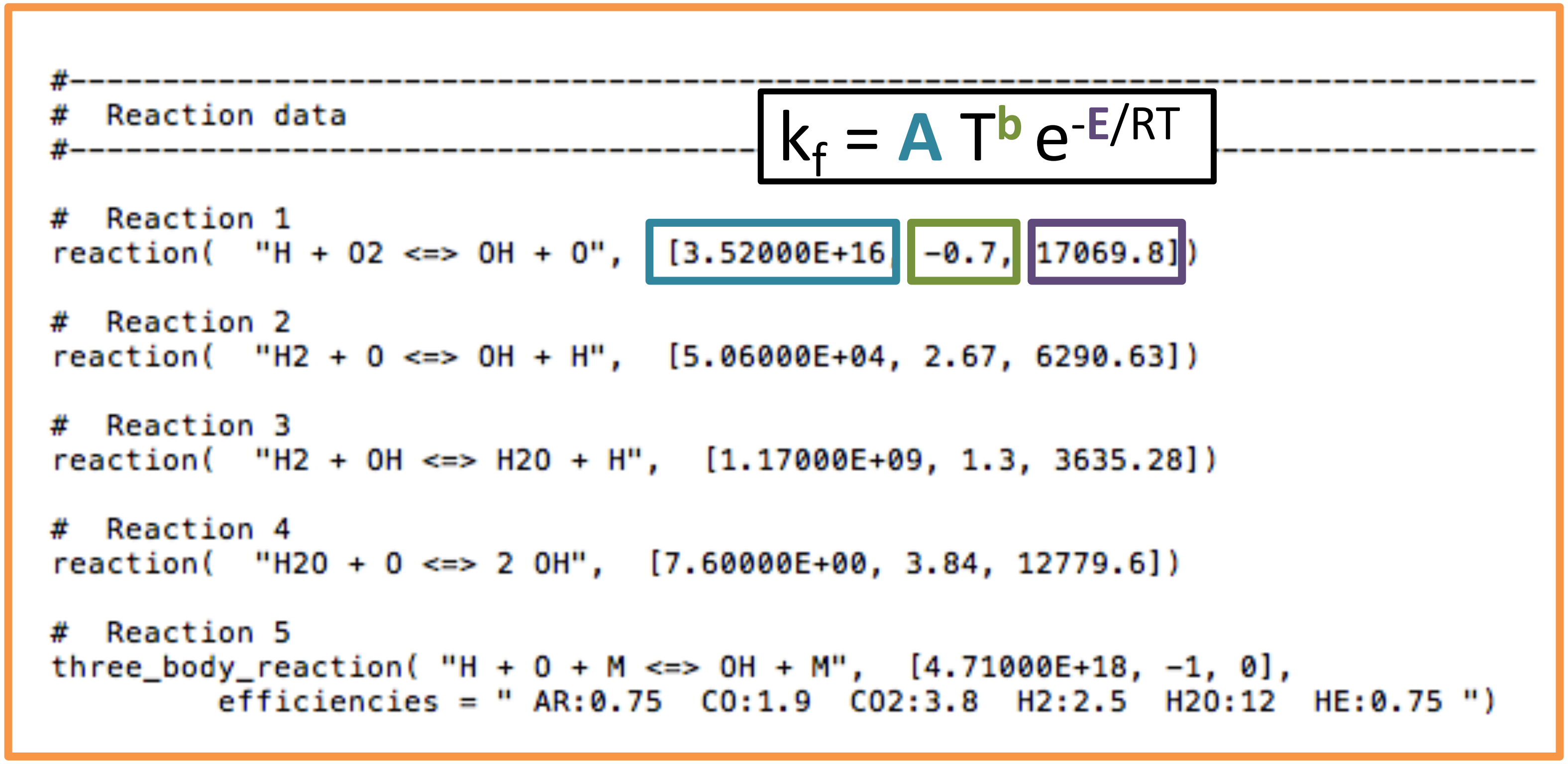
$$\begin{cases} \dfrac{Cp}{R} = a_1 + a_2 T + a_3 T^2 + a_4 T^3 + a_5 T^4 \\ \dfrac{H}{RT} = a_1 + a_2 \dfrac{T}{2} + a_3 \dfrac{T^2}{3} + a_4 \dfrac{T^3}{4} + a_5 \dfrac{T^4}{4} + \dfrac{a_6}{T} \\ \dfrac{S}{R} = a_1 \ln(T) + a_2 T + a_3 \dfrac{T^2}{2} + a_4 \dfrac{T^3}{3} + a_5 \dfrac{T^4}{4} + a_7 \end{cases}$$

Cantera link : https://cantera.org/science/science-species.html

Practical use

Data file

Reactions Data



```
#---------------------------------------------------------------------------------
#   Reaction data
#---------------------------------------------------------------------------------

#   Reaction 1
reaction(  "H + O2 <=> OH + O",  [3.52000E+16, -0.7, 17069.8])

#   Reaction 2
reaction(  "H2 + O <=> OH + H",  [5.06000E+04, 2.67, 6290.63])

#   Reaction 3
reaction(  "H2 + OH <=> H2O + H",  [1.17000E+09, 1.3, 3635.28])

#   Reaction 4
reaction(  "H2O + O <=> 2 OH",  [7.60000E+00, 3.84, 12779.6])

#   Reaction 5
three_body_reaction( "H + O + M <=> OH + M",  [4.71000E+18, -1, 0],
         efficiencies = " AR:0.75  CO:1.9  CO2:3.8  H2:2.5  H2O:12  HE:0.75 ")
```

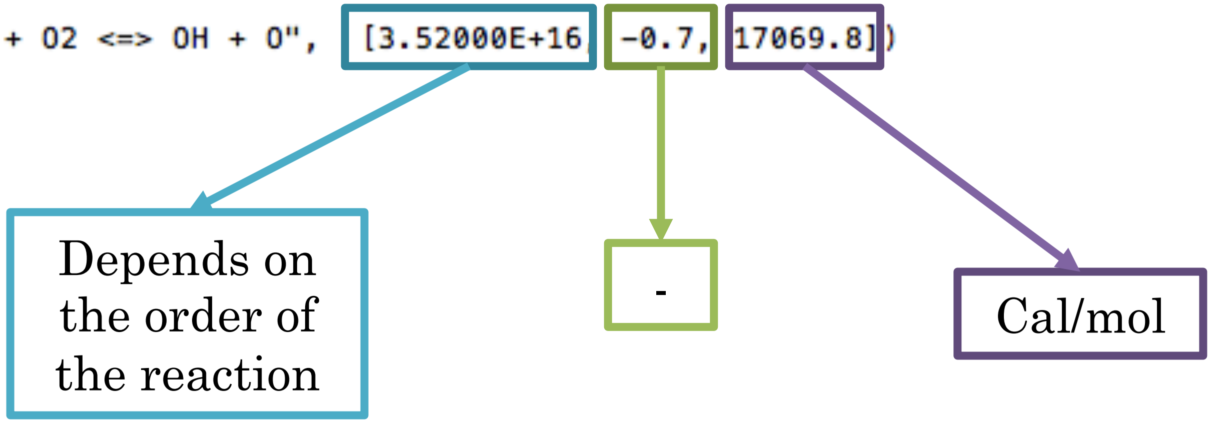
$$k_f = A\, T^{b}\, e^{-E/RT}$$

Practical use

Data file

```
units(length = "cm", time = "s", quantity = "mol", act_energy = "cal/mol")


  #------------------------------------------------------------------------------
  #  Reaction data
  #------------------------------------------------------------------------------

  #  Reaction 1
  reaction(  "H + O2 <=> OH + O",  [3.52000E+16, -0.7, 17069.8])
```

$$\boxed{k_f = A\, T^{b}\, e^{-E/RT}}$$



Here : $\omega_{f,j} = k_f[O_2][H]$

So the order is 2 and the unit is :
$[mol/m^3/s]/[mol/m^3]^2 = [mol^{-1}m^{-3}s^{-1}]$

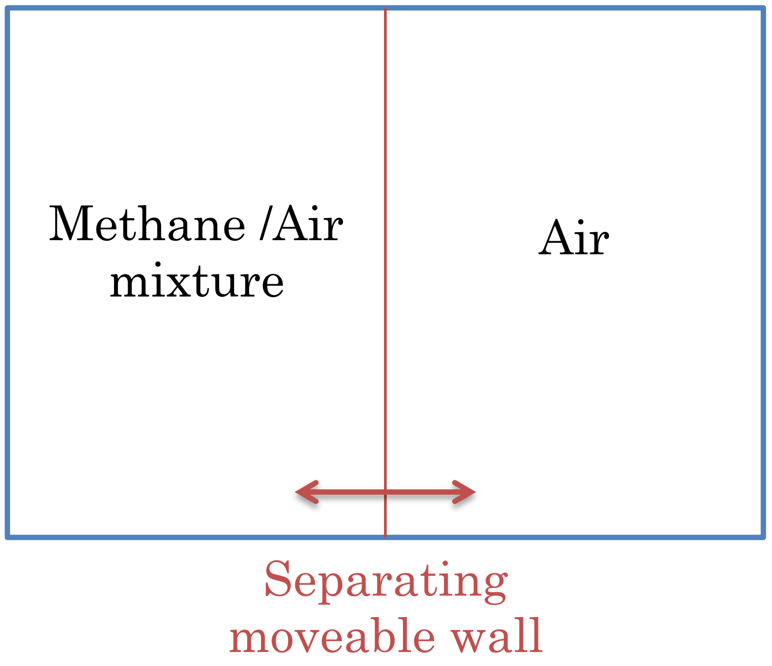
More data on the type of equations can be found in appendix.

Example
The constant pressure batch reactor

It is composed of 5 main « building blocks »

- 2 Reactors
- Methane-air mixture
- Non reactive air mixture
- Moveable wall

They form a constant pressure reactor !



The piston-cylinder system

The constant pressure batch reactor

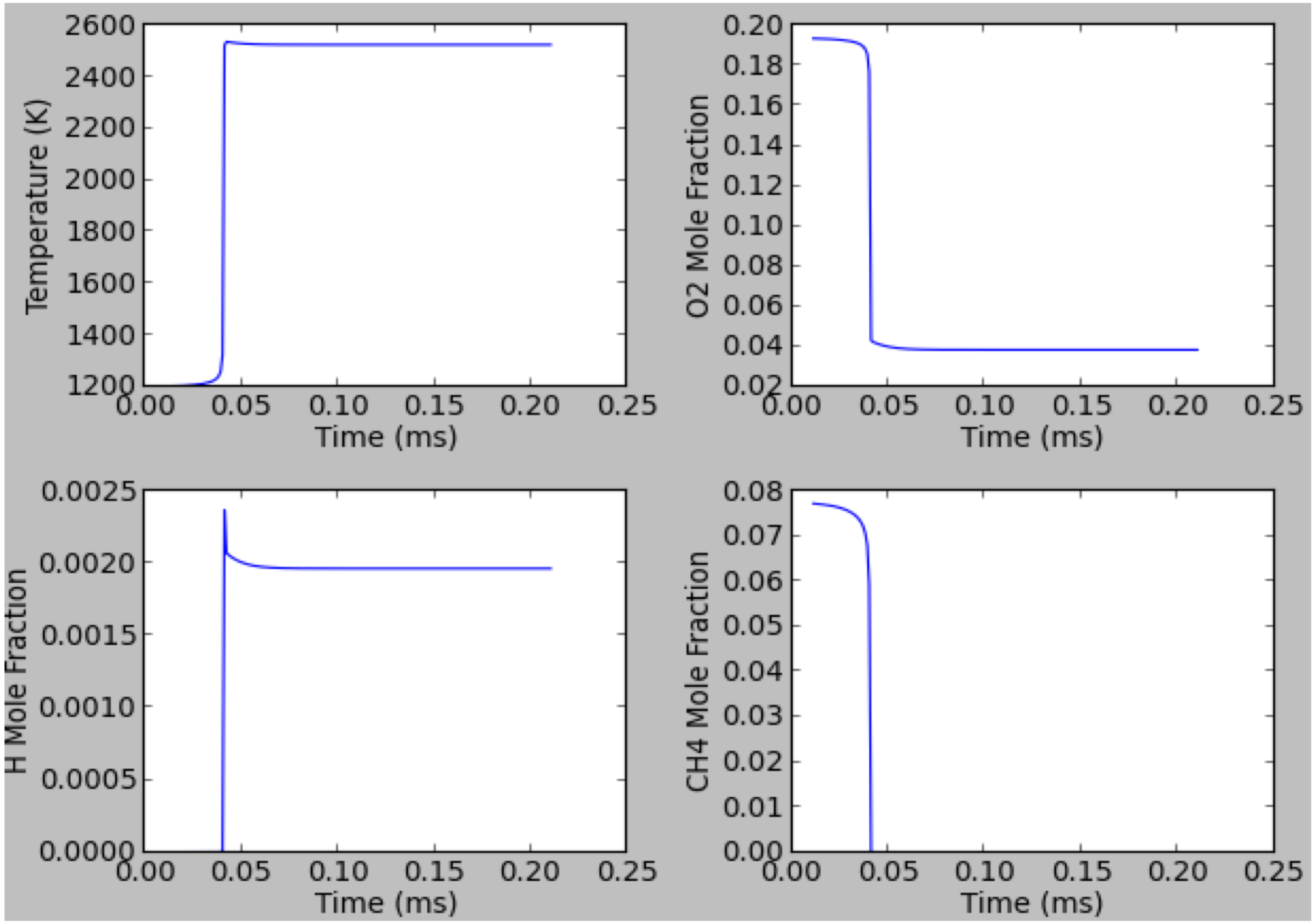
What does it look like, « program-wise » ?

Building blocks

```
      """  Constant pressure reactor ‘.py’ script. """

#Mechanism used for the methane-air mixture
cti = importPhase('gri30.cti')
#Set initial conditions
cti.set(T = 1200, P = 1, X = 'CH4:0.40, O2:1, N2:3.76')
# Create the batch reactor
r   = Reactor(cti)

#Create a reactor for the environment containing air at
#atmospheric conditions
env = Reservoir(Air())

# Define a wall between the reactor and the environment, and
# make it flexible, so that the pressure in the reactor is held
# at the environment pressure.
w = Wall(r,env)
w.set(K = 1.0e6) # set expansion parameter
w.set(A = 1.0)    # set wall area

# Now create a reactor network consisting of the single batch
reactor
# Reason: the only way to advance reactors in time is through a
sim = ReactorNet([r])
```

The constant pressure batch reactor

What will you observe ?



Practical use
Helpful links

- **Github**

https://github.com/Cantera/cantera

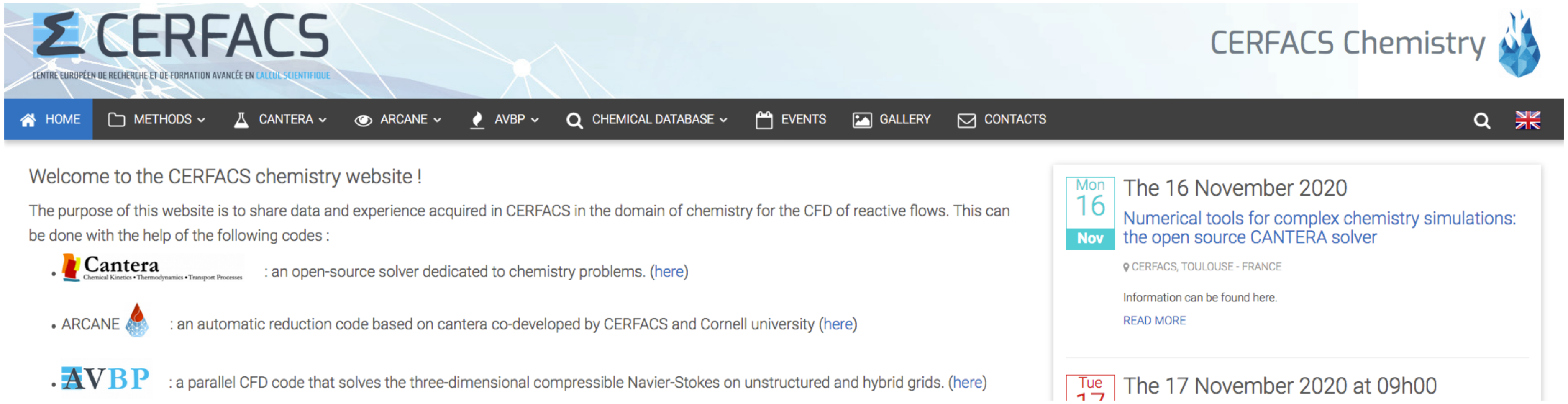
- **Google Groups page for Cantera**

http://groups.google.com/group/cantera-users

- **Cantera SourceForge Page**
  http://sourceforge.net/projects/cantera/files/

To download all Cantera versions, source code or (Windows) binaries and find more documentation.

CERFACS CANTERA website : https://chemistry.cerfacs.fr



Welcome to the CERFACS chemistry website !

The purpose of this website is to share data and experience acquired in CERFACS in the domain of chemistry for the CFD of reactive flows. This can be done with the help of the following codes :

- Cantera : an open-source solver dedicated to chemistry problems. (here)
- ARCANE : an automatic reduction code based on cantera co-developed by CERFACS and Cornell university (here)
- AVBP : a parallel CFD code that solves the three-dimensional compressible Navier-Stokes on unstructured and hybrid grids. (here)

Mon 16 Nov — The 16 November 2020

Numerical tools for complex chemistry simulations: the open source CANTERA solver

CERFACS, TOULOUSE - FRANCE

Information can be found here.

READ MORE

Tue 17 — The 17 November 2020 at 09h00

CERFACS knowledge and experience in chemistry for CFD in one website !!!

- ***Chemical database*** : Detailed, reduced and global kinetics mechanisms
- ***Cantera*** : CERFACS' version with installation walkthrough, scripts and tutorials
- ***Private documentation*** : ARCANE, AVBP
- ***Events*** : such as this training
- ***CFD gallery*** : nice pictures with great chemistry

Content of the presentation

Installing CANTERA

Official versions

You can install CANTERA on both LINUX, Windows and Mac.

To install the official version, instructions can be found here
https://cantera.org/install/index.html

The current version is Cantera 2.4.0.

Installing CANTERA

CERFACS versions

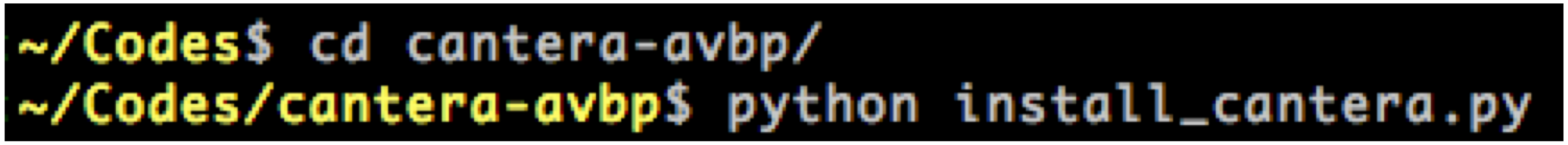
If you are from CERFACS get the sources with

```
git clone git@nitrox.cerfacs.fr:cantera/cantera-avbp.git
```

From outside, you can get them from :
https://chemistry.cerfacs.fr/en/cantera-installation/

Installing CANTERA

The advised procedure for a local installation of the CERFACS Cantera version is



```
~/Codes$ cd cantera-avbp/
~/Codes/cantera-avbp$ python install_cantera.py
```

The script is only valid for NFS machines installation and Mac OSX installation.

Thank you for your attention !

$$\begin{tikzcd}
 & \text{Any questions ?} \arrow[dl, bend right] \arrow[dr, bend left] & \\
\text{Yes, please ask !} \arrow[ur, bend left] & & \text{No, it's fine !} \arrow[dl, bend left] \\
 & \text{Tutorial time !!} &
\end{tikzcd}$$

Appendix

1. Cantera VS CHEMKIN

Cantera VS CHEMKIN

- Its possibilities are **comparable to the CHEMKIN-II suite :**

CHEMKIN = a set of FORTRAN libraries	**CANTERA = a set of C++ libraries**
3 input files (thermo / transport / mechanism)	1 « data file » (everything)
« Interpreter step » to generate the binary input file	▨
A driver to stir the program towards simulations (***<u>Keywords</u>***)	A script to arrange « building blocks » into a simulation (***<u>Interface objects and functions</u>***)
Outputs written by the libraries	Outputs generated by the language of the script (Python, C++, Matlab, FORTRAN)

Cantera VS CHEMKIN

- Its possibilities are **comparable to the CHEMKIN-II suite :**

CHEMKIN = a set of FORTRAN libraries	**CANTERA = a set of C++ libraries**
3 input files (thermo / transport / mechanism)	**1 « data file » (everything)**
« Interpreter step » to generate the binary input file	▨
A driver to stir the program towards simulations (***Keywords***)	A script to arrange « building blocks » into a simulation (***Interface objects and functions***)
Outputs written by the libraries	Outputs generated by the script's language (Python, C++, Matlab, FORTRAN)

Cantera VS CHEMKIN

- It's possibilities are **comparable to the CHEMKIN-II suite :**

CHEMKIN = a set of FORTRAN libraries	**CANTERA = a set of C++ libraries**
3 input files (thermo / transport / mechanism)	1 « data file » (everything)
« Interpreter step » to generate the binary input file	
A driver to stir the program towards simulations (***Keywords***)	A script to arrange « building blocks » into a simulation (***Interface objects and functions***)
Outputs written by the libraries	Outputs generated by the script's language (Python, C++, Matlab, FORTRAN)

Cantera VS CHEMKIN

◆ It's p **I suite :**

C F	**C++**
3 inp	
« Inte	
A driv s	locks » ***bjects***
Out	ipt's lab,

CHEMKIN KEYWORD INPUT

```
/ flame configuration, burner stabilized with specified
temperature
BURN
TGIV
/ in the event of a Newton failure, take 100 timesteps
of 1.E-6
TIME 100 1.00E-6
/ begin on a uniform mesh of 6 points
NPTS 6
/ definition of the computational interval
XEND 10.0
XCEN 5.0
WMIX 10.0
/ pressure and inlet mass flow rate
PRES 0.0329 (atmospheres)
FLRT 4.63E-3 (g/cm**2-sec)
```

KEYWORDS

Cantera VS CHEMKIN

- It's possibilities are **comparable to the CHEMKIN-II suite :**

CHEMKIN = a set of FORTRAN libraries	**CANTERA = a set of C++ libraries**
3 input files (thermo / transport / mechanism)	1 « data file » (everything)
« Interpreter step » to generate the binary input file	
A driver to stir the program towards simulations (***Keywords***)	A script to arrange « building blocks » into a simulation (***Interface objects and functions***)
Outputs written by the libraries	Outputs generated by the script's language (Python, C++, Matlab, FORTRAN)

Cantera VS CHEMKIN

- It's possibilities are **comparable to the CHEMKIN-II suite :**

CHE FORT	**+**
3 input fil	
« Interpret bi	
A driver to simul	ks » ***cts***
Outputs	's),

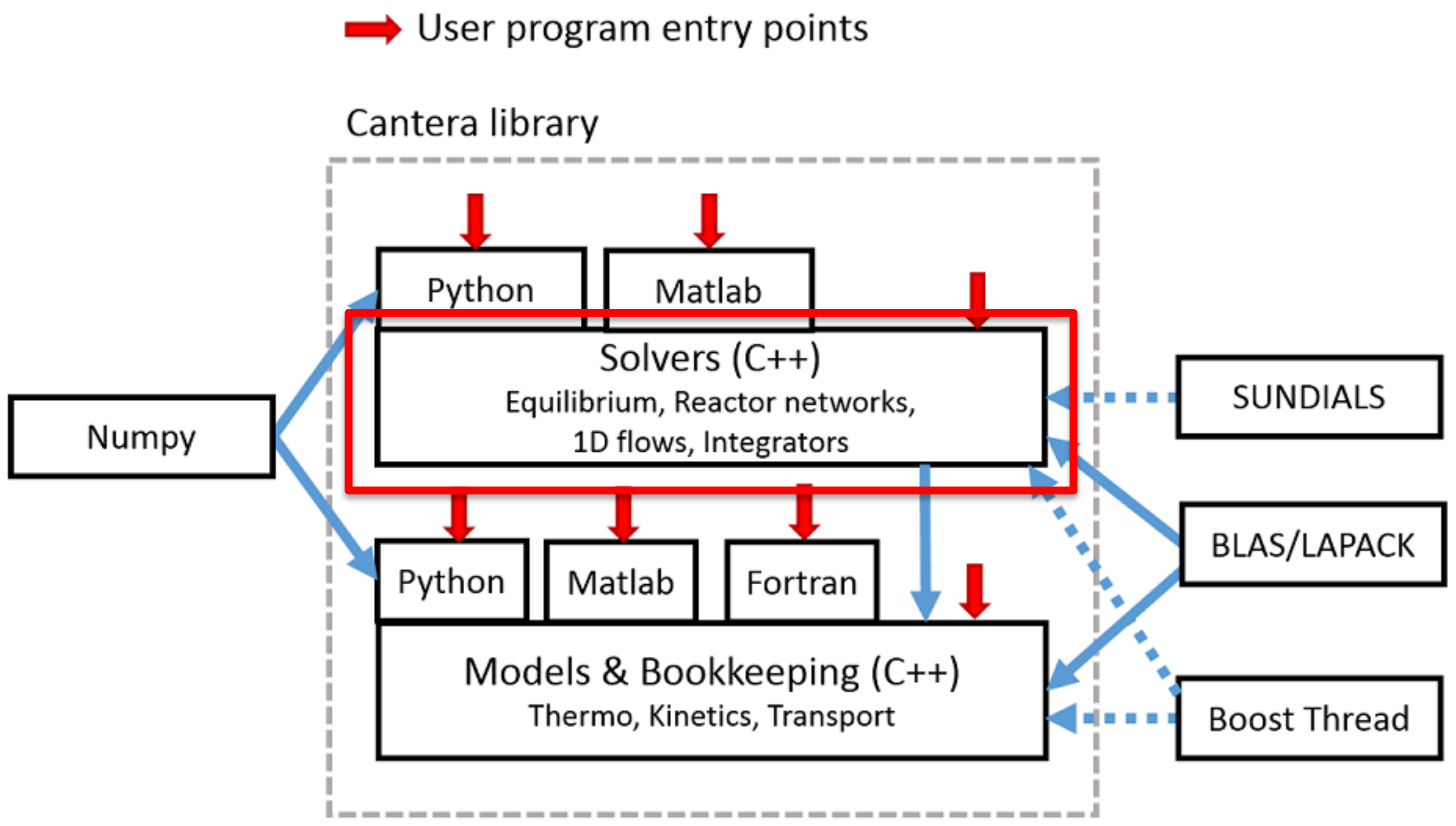
```
"""  Burner Stabilized flame '.py' script. """

initial_grid     = [0.0, 0.0025, 0.005, 0.0075, 0.0099, 0.01]

f                = ct.BurnerFlame(gas, initial_grid)
f.burner.T       = 400
f.burner.X       = reactants
f.burner.mdot    = 0.04
```

OBJECTS

FUNCTIONS

2. Detailed structure of Cantera

Structure of CANTERA



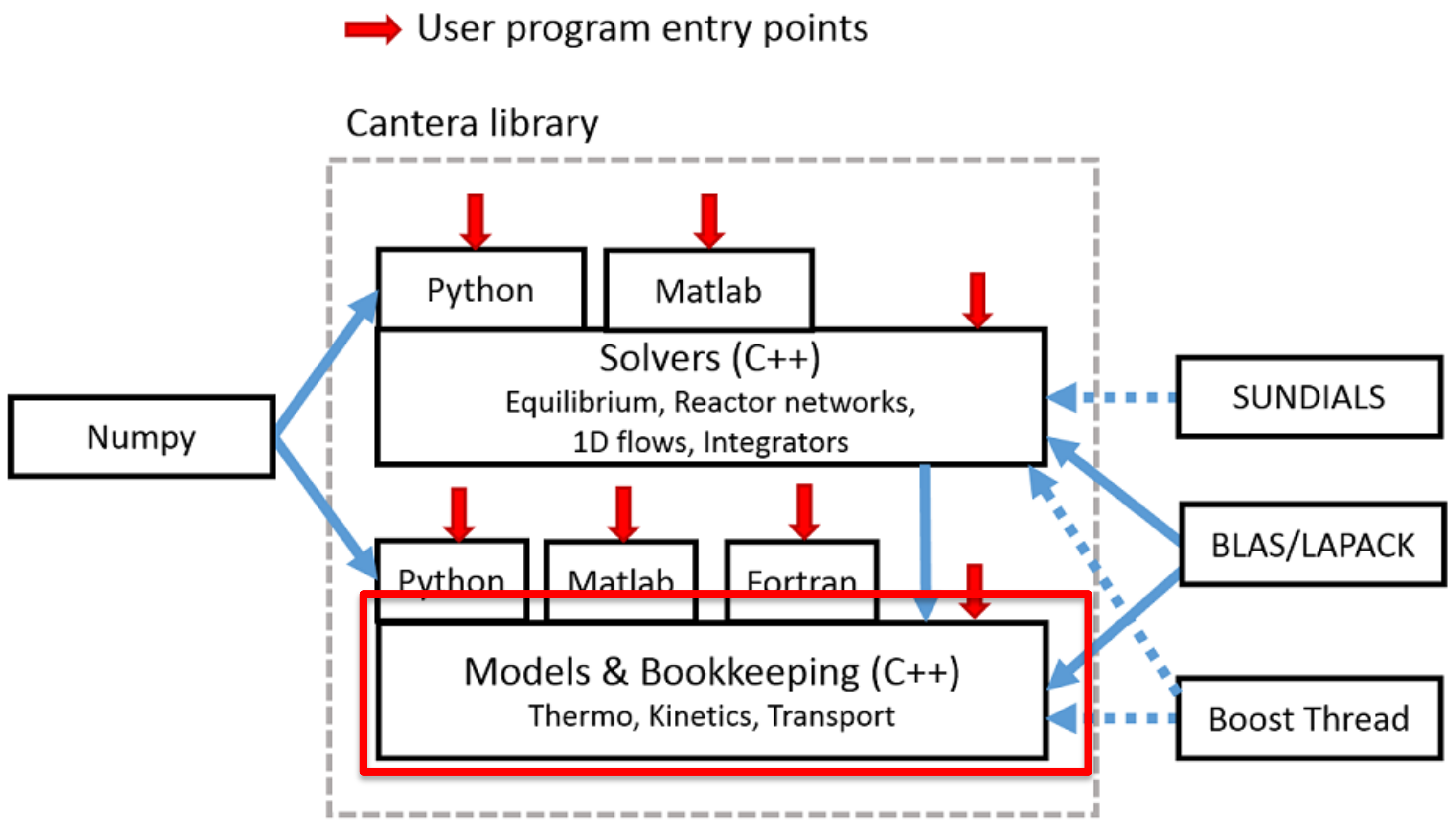
Structure of CANTERA

The « Solver » layer

Usually hidden from the user, and **borrows from famous « free » libraries** (LAPACK, BLAS, …) to perform

- Equilibrium calculations
- Reactor equations integration
- 1D calculations
- …

Structure of CANTERA



The « Bookkeeping » layer

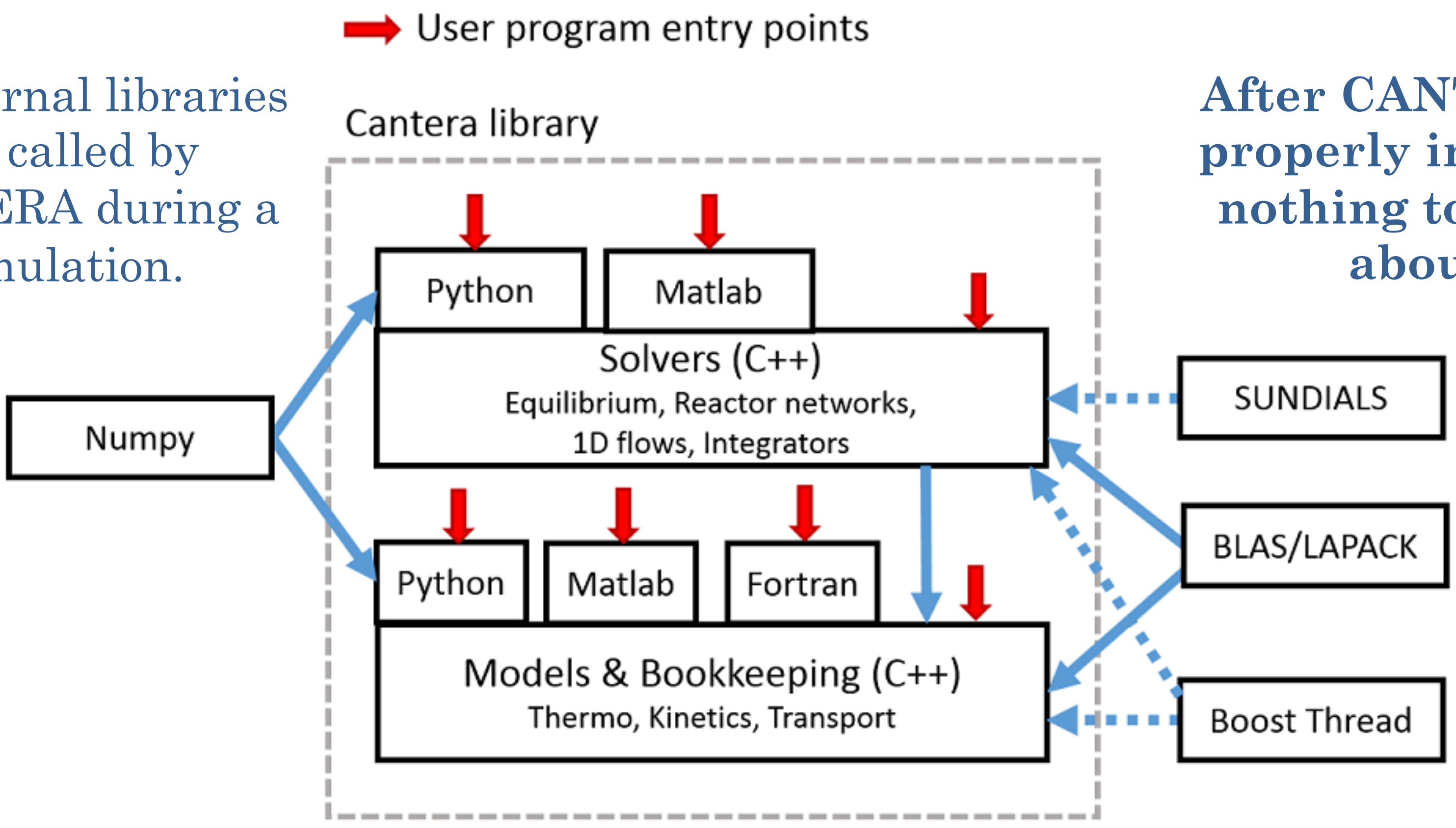
As we have just seen, it is the python script entry.
This layer contains all the methods that will

- **Initialize objects** defined in the script
  - If a phase object is defined, it will *calculate and set its thermodynamic state* and implement *their transport models* (example 1).
  - Set the *inlet conditions* of a "FreeFlame" object (example 2).
- **Link all objects** together
  - *Link two reactors* through a wall (example).
- **Organize the simulation**
  - Call the required solvers (so, the "solvers" layer)
  - Extract required output data

Structure of CANTERA

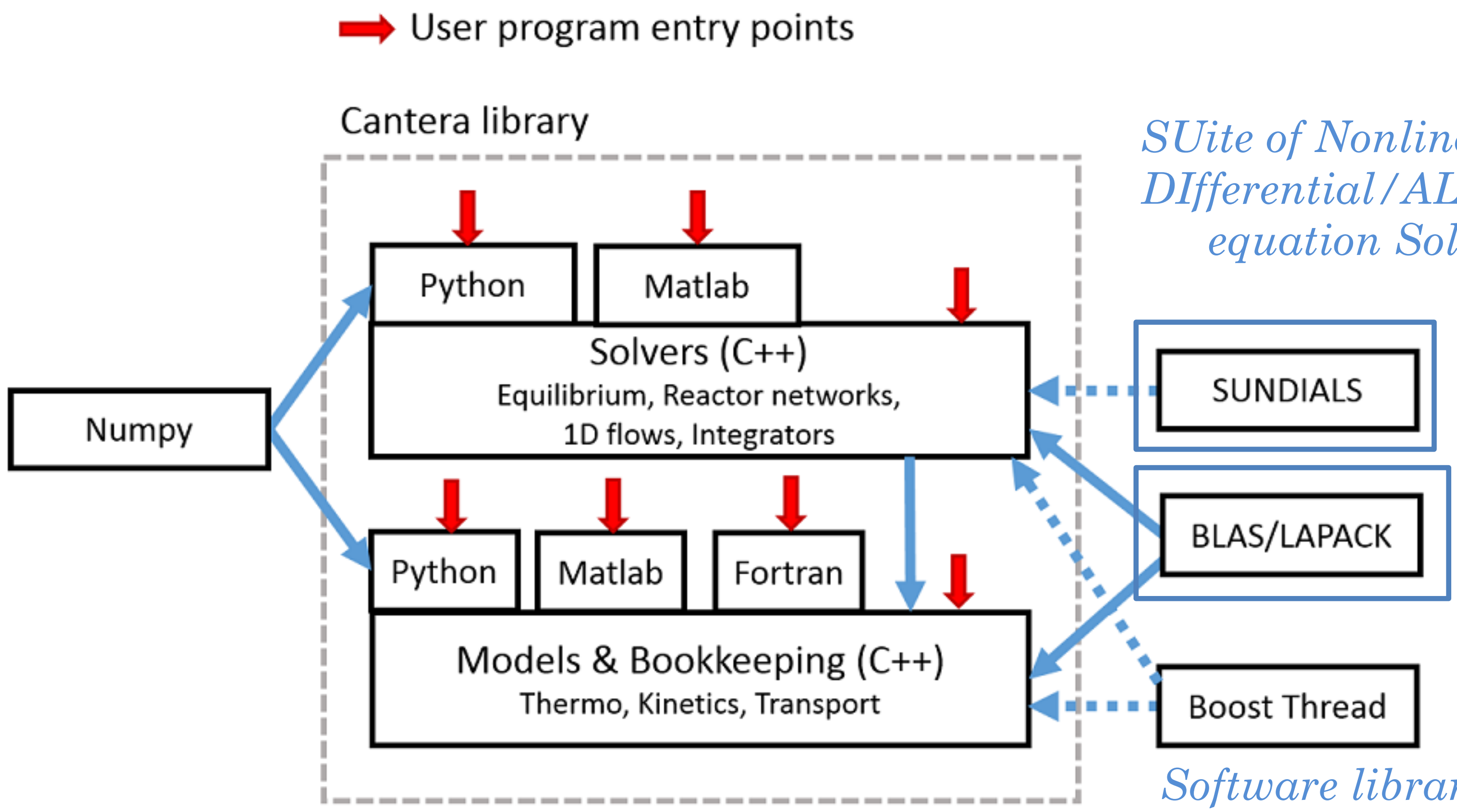
Cantera is a big lasagna, it has layers.

All external libraries are called by CANTERA during a simulation.


User program entry points
Cantera library
Python
Matlab
Solvers (C++)
Equilibrium, Reactor networks, 1D flows, Integrators
Numpy
Python
Matlab
Fortran
Models & Bookkeeping (C++)
Thermo, Kinetics, Transport
SUNDIALS
BLAS/LAPACK
Boost Thread

After CANTERA is properly installed, nothing to worry about.

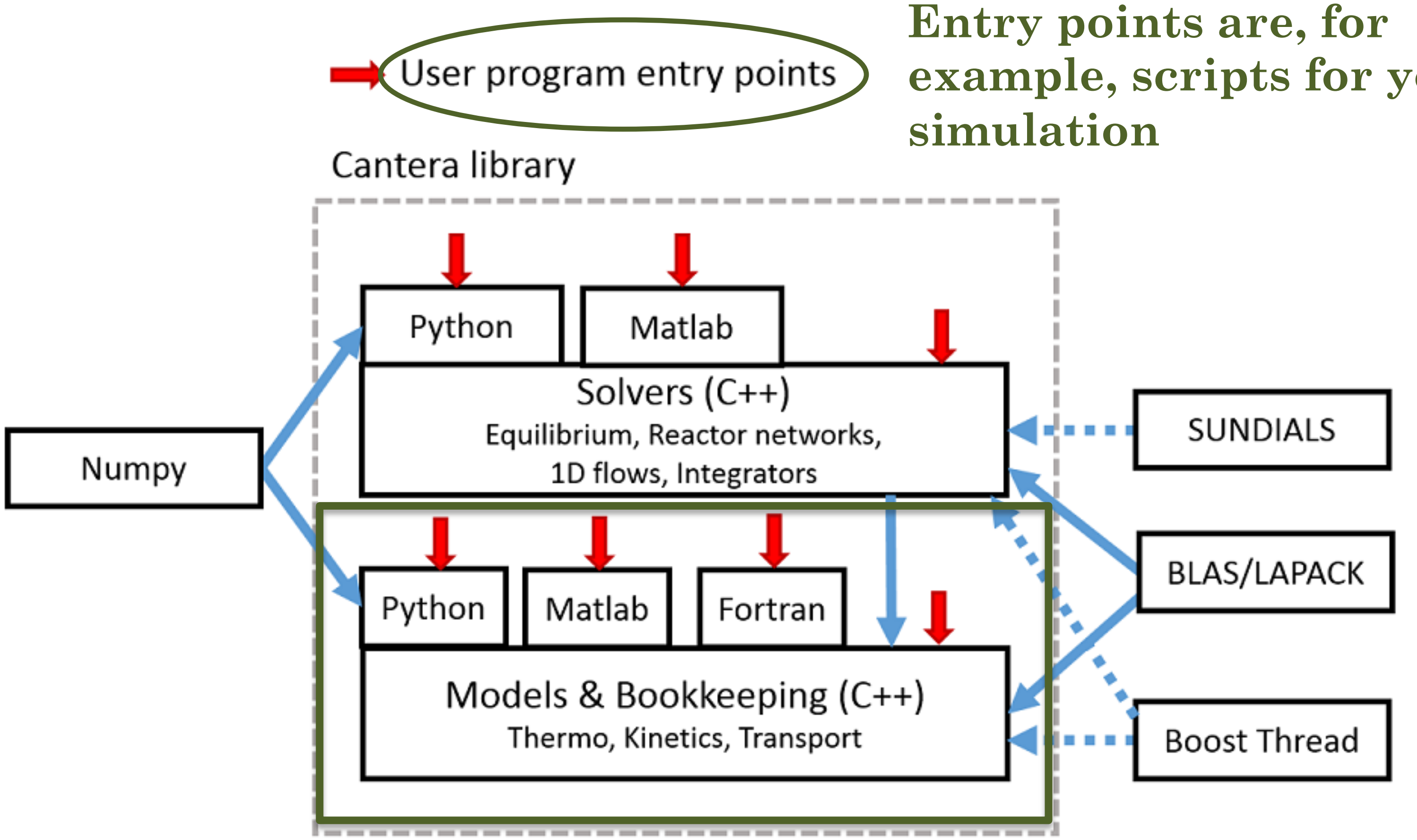
Structure of CANTERA



SUite of Nonlinear and DIfferential/ALgebraic equation Solvers

Software libraries for numerical linear algebra

Structure of CANTERA



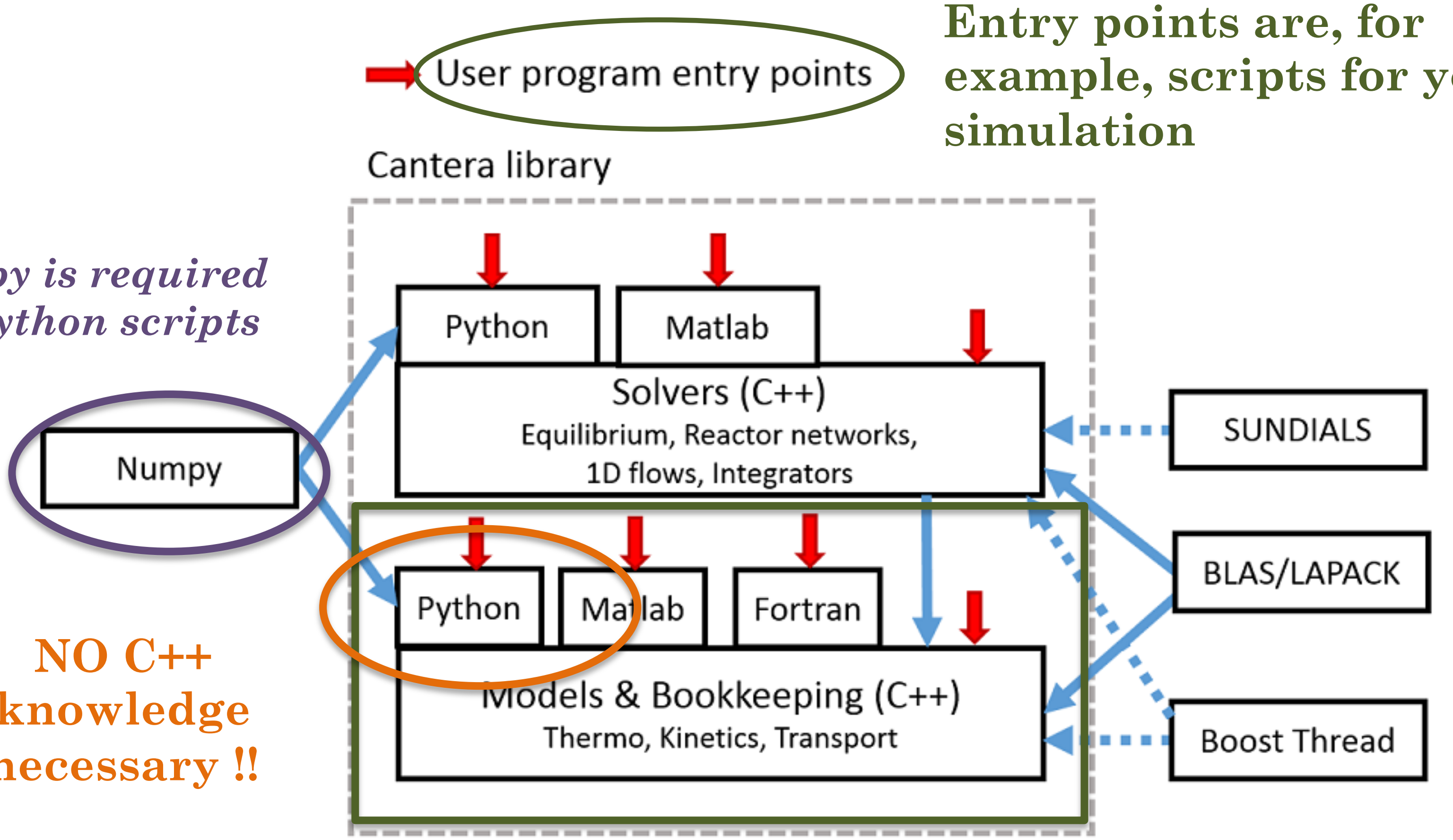
Entry points are, for example, scripts for your simulation

Structure of CANTERA

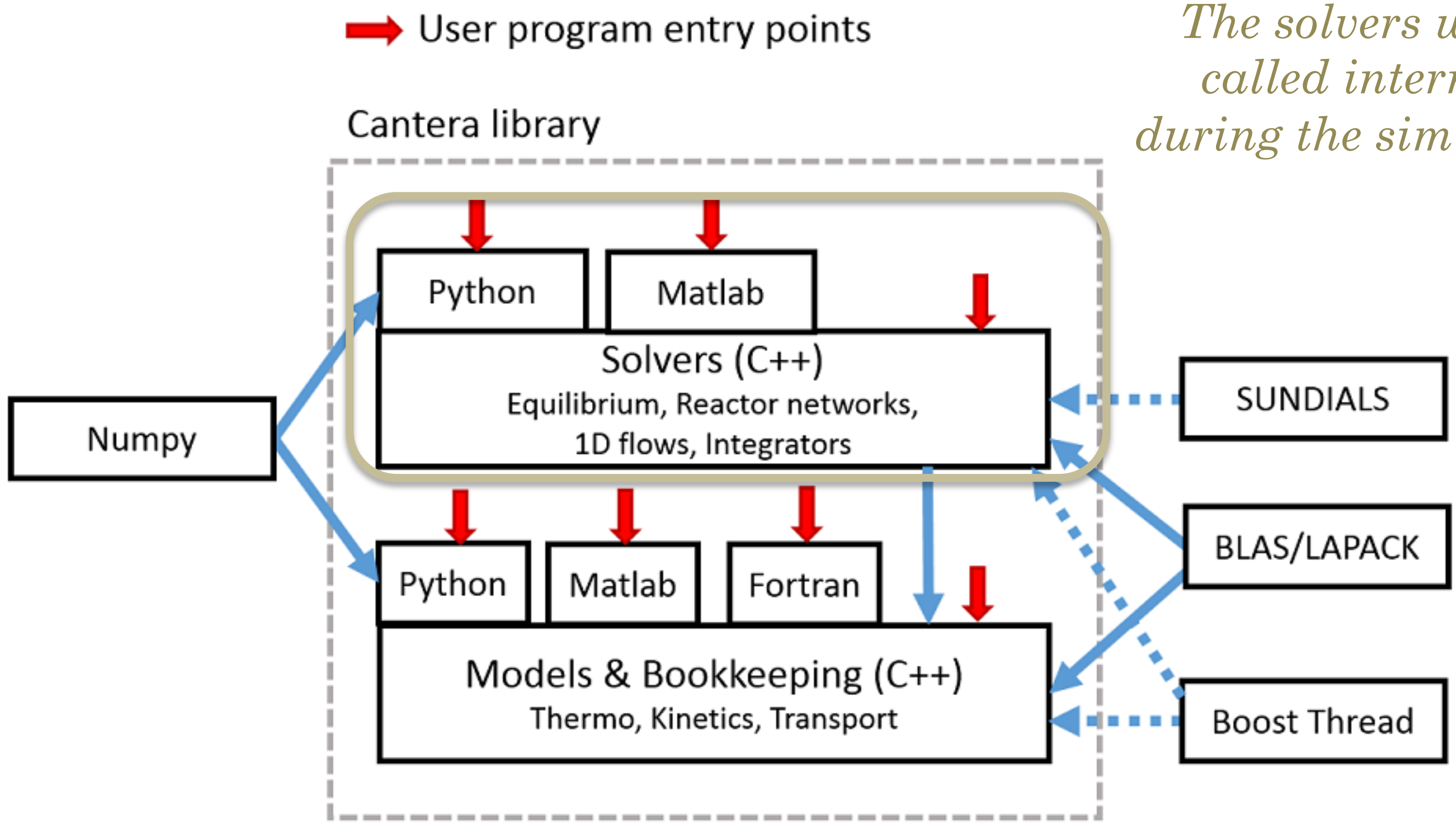
Numpy is required for Python scripts

NO C++ knowledge necessary !!

Entry points are, for example, scripts for your simulation



Structure of CANTERA



The solvers will be called internally during the simulation.

3. Gibbs function

Example with the Gibbs function

"The equilibrium state is that corresponding to a minimum of a property called the energy function under specified conditions"

Use the Gibbs energy function G: $G = G(T, P, N_k)$

So that, when **P and T are constant**:

At equilibrium, we want to minimize G

$$\boxed{dG = \sum_{k=1}^{K} \mu_k dN_k}$$

with $\mu k = \frac{\partial U}{\partial Nk}$

$$p_l = \sum_{k=1}^{K} n_{kl} N_k$$

With the constraint that the number of moles p_l of every element l (N, O, H, …) is conserved:

Example with the Gibbs function

The non-stoichiometric method

This becomes an optimisation problem where

$$dG = \sum_{k=1}^{K} \mu_k dN_k = 0$$

$$p_l^* = p_l - \sum_{k=1}^{K} n_{kl} N_k = 0$$

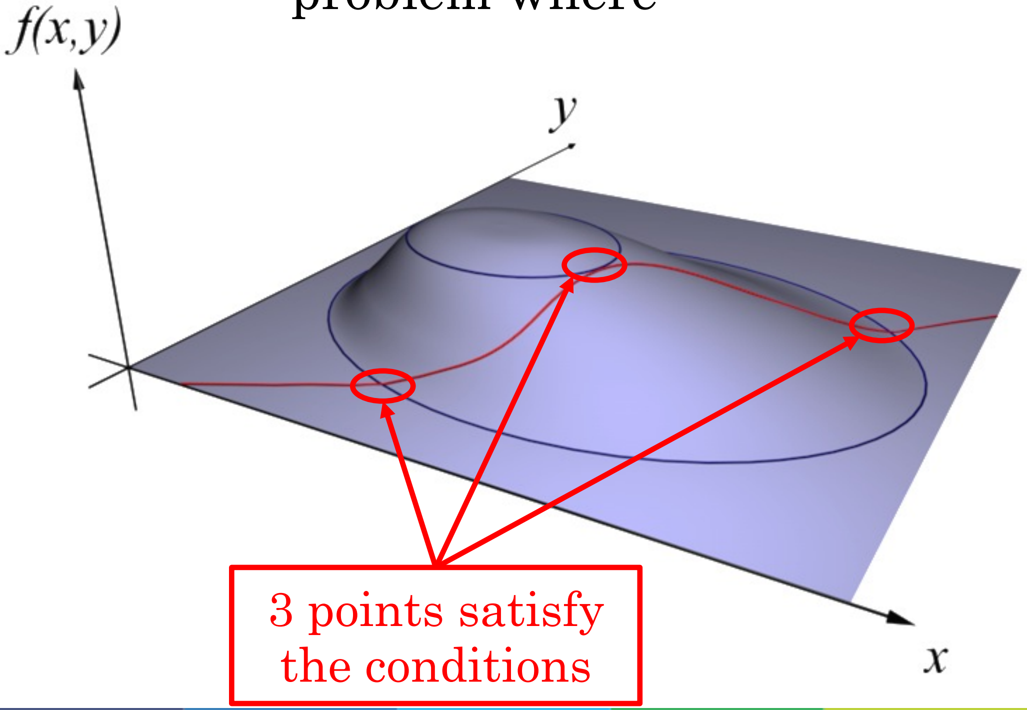


Illustration in 2D

- Find an extremum of the function G(x,y), represented by the blue lines

- that satisfies the condition $p_l^*(x,y)$=smthg represented by the red line

Example with the Gibbs function

The non-stoichiometric method

This becomes an optimisation problem where

$$\boxed{\begin{aligned} dG &= \sum_{k=1}^{K} \mu_k dN_k = 0 \\ p_l^* &= p_l - \sum_{k=1}^{K} n_{kl} N_k = 0 \end{aligned}}$$

Which is solved by **introducing Lagrange multipliers** λ_l such that

$$G^* = G + \sum_{l=1}^{L} \lambda_l p_l^*$$

And the problem can be posed as a solution of a set of
(K + l) nonlinear equations

$$\boxed{\begin{aligned} \frac{\partial G^*}{\partial N_k} &= \mu_k - \sum_{l=1}^{L} \lambda_l n_{kl} = 0 \\ \frac{\partial G^*}{\partial \lambda_l} &= p_l^* = 0 \end{aligned}}$$

The non-stoichiometric method

Once the λ_l are determined, since **T & P are constant**, the mole fractions are automatically deduced.

$$\boxed{\mu_k = \sum_{l=1}^{L} \lambda_l n_{kl} \Rightarrow X_k = \frac{P_o}{P} \exp\left(-\frac{g_k^0(T)}{RT} + \sum_{l=1}^{L} n_{kl} \frac{\lambda_l}{RT}\right)}$$

- **General procedure** (***Note:*** *no need to provide reactions information !*) :

- The g_k^0 are tabulated.
- The user provides a guess for enough (L) X_k - with the knowledge that $\sum_{k=1}^{K} X_k = 1$
- The λ_l can then be deduced from the previous K equations.
- The unkown X_k are calculated with those estimated λ_l and $\sum_{k=1}^{K} X_k$ is evaluated.
- If $\sum_{k=1}^{K} X_k$ is « too far » from 1, a new guess for the X_k is provided and the procedure reiterates with well chosen $L X_k$

4. Keywords in the cti format

So… how is it written (format '.cti') ?

It is **composed of « entries » and « directives »** recognized via keywords.

So… how is it written (format '.cti') ?

It is **composed of « entries » and « directives »** recognized via keywords.

A directive will tell the code how the entry parameters are to be interpreted.

For example, the 'units' directive

```
units(length = "cm", time = "s", quantity = "mol", act_energy = "cal/mol")
```

Practical use

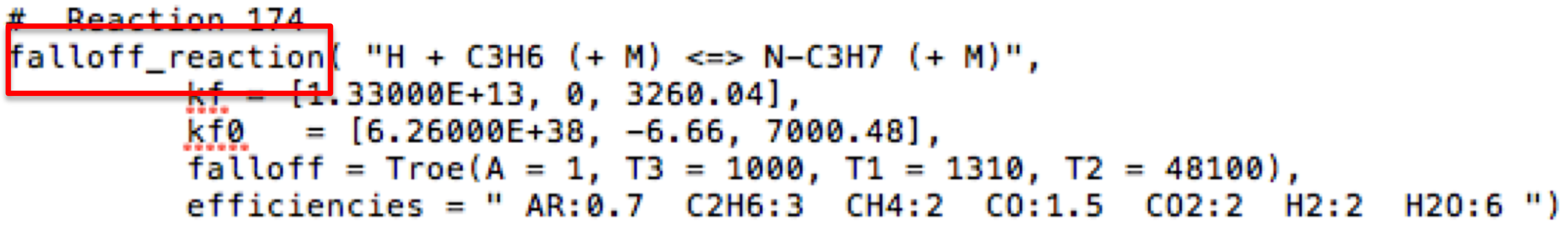
Data file

So… how is it written (format '.cti') ?

It is **composed of « entries » and « directives »** recognized via keywords.

An entry defines an object.

For example, a falloff reaction



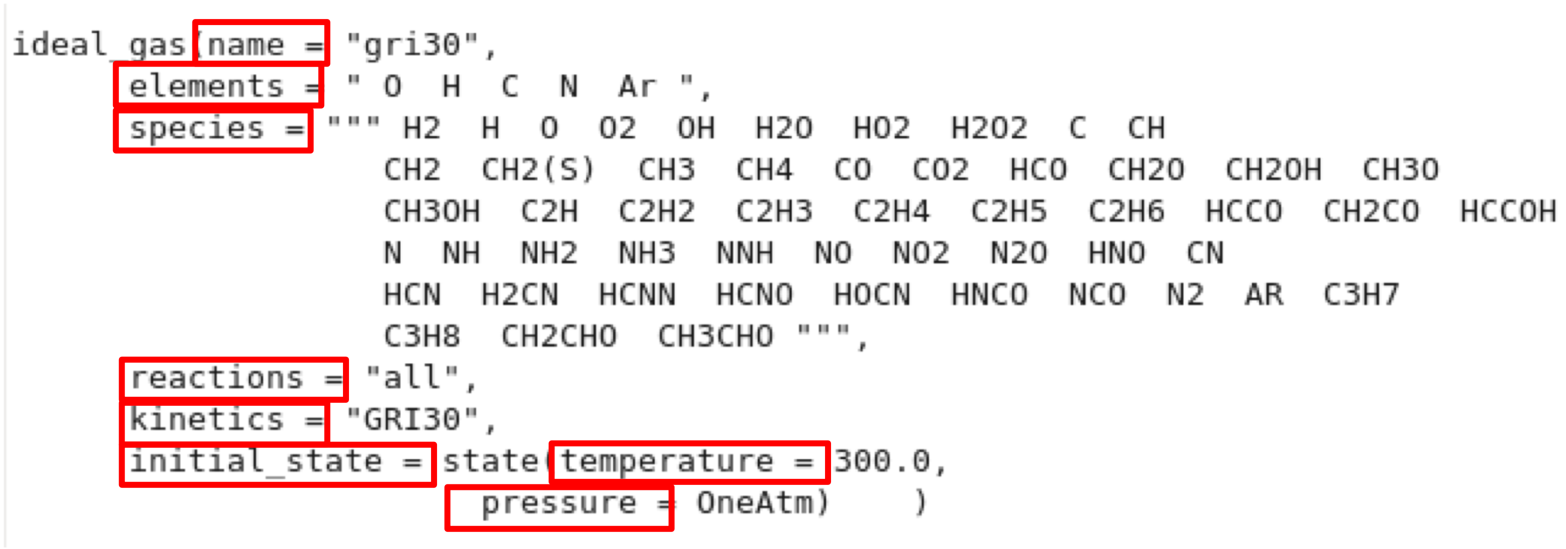
```
#  Reaction 174
falloff_reaction( "H + C3H6 (+ M) <=> N-C3H7 (+ M)",
         kf = [1.33000E+13, 0, 3260.04],
         kf0   = [6.26000E+38, -6.66, 7000.48],
         falloff = Troe(A = 1, T3 = 1000, T1 = 1310, T2 = 48100),
         efficiencies = " AR:0.7  C2H6:3  CH4:2  CO:1.5  CO2:2  H2:2  H2O:6 ")
```

So… how is it written (format ‘.cti’) ?

It is **composed of « entries » and « directives »** recognized via keywords.

Entries are composed of a series of <u>fields</u> that can be assigned values.

For example, ‘name’ or ‘elements’ for a phase

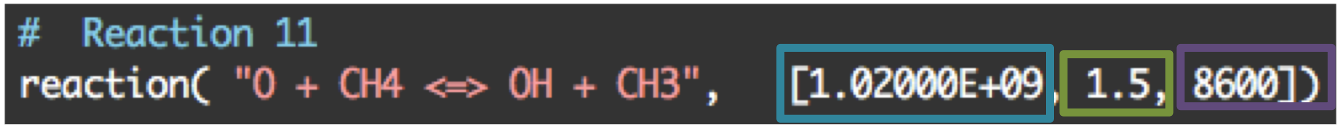


```
ideal_gas(name = "gri30",
      elements = " O  H  C  N  Ar ",
      species = """ H2  H  O  O2  OH  H2O  HO2  H2O2  C  CH
                   CH2  CH2(S)  CH3  CH4  CO  CO2  HCO  CH2O  CH2OH  CH3O
                   CH3OH  C2H  C2H2  C2H3  C2H4  C2H5  C2H6  HCCO  CH2CO  HCCOH
                   N  NH  NH2  NH3  NNH  NO  NO2  N2O  HNO  CN
                   HCN  H2CN  HCNN  HCNO  HOCN  HNCO  NCO  N2  AR  C3H7
                   C3H8  CH2CHO  CH3CHO """,
      reactions = "all",
      kinetics = "GRI30",
      initial_state = state(temperature = 300.0,
                            pressure = OneAtm)     )
```

5. Equations in the cti

Practical use
Data file

Simple forward constant rates coefficients



```
#  Reaction 11
reaction( "O + CH4 <=> OH + CH3",    [1.02000E+09, 1.5, 8600])
```

$$\boxed{k_f = A\, T^b\, e^{-E/RT}}$$

If the reaction is irreversible

$$\omega = k_f [R_1][R_2]$$

Net reaction rate
net_rates_of_progress

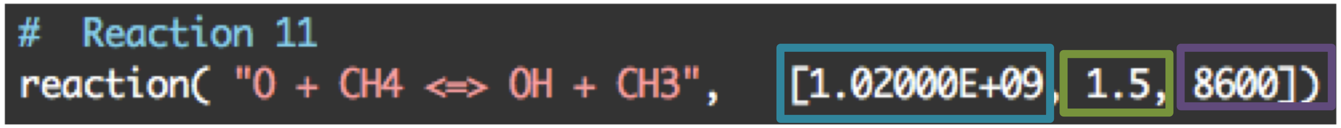
Arrhenius forward coefficient
forward_rate_constants

R_1, R_2 = reactants 1 and 2

Practical use

Data file

Simple forward constant rates coefficients



```
#  Reaction 11
reaction( "O + CH4 <=> OH + CH3",   [1.02000E+09, 1.5, 8600])
```

$$\boxed{k_f = A\,T^{b}\,e^{-E/RT}}$$

If the reaction is reversible

$$\omega = \boxed{k_f[R_1][R_2]} - \boxed{k_B[P_1][P_2]}$$

Forward reaction rate
forward_rates_of_progress

Forward reaction rate
reverse_rates_of_progress

with

Arrhenius backward coefficient

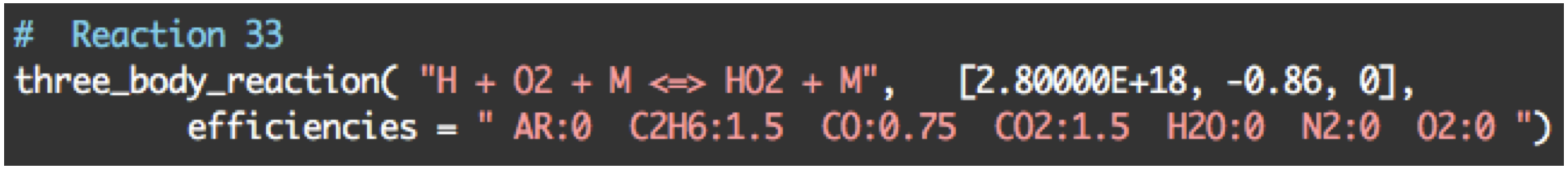
Equilibrium constants
equilibrium_constants

$$k_B = k_f/K_e \quad \text{and} \quad K_e = [P_1][P_2]\,/[R_1][R_2]$$

R_1, R_2 = reactants 1 and 2 and P_1, P_2 = products 1 and 2

Practical use

Data file



```
#  Reaction 33
three_body_reaction( "H + O2 + M <=> HO2 + M",    [2.80000E+18, -0.86, 0],
        efficiencies = " AR:0  C2H6:1.5  CO:0.75  CO2:1.5  H2O:0  N2:0  O2:0 ")
```

$$\omega = k_f[H][O_2][M] - k_B[HO_2][M]$$

with $$[M] = \sum_{k=1}^{K} \epsilon_k C_k$$ (by default, if not specified, $\epsilon_k = 1$)

efficiencies

Practical use

Data file

```
#  Reaction 12
falloff_reaction( "O + CO (+ M) <=> CO2 (+ M)",
        kf = [1.80000E+10, 0, 2385],
        kf0   = [6.02000E+14, 0, 3000],
        efficiencies = " AR:0.5  C2H6:3  CH4:2  CO:1.5  CO2:3.5  H2:2  H2O:6  O2:6 ")
```

Coefficient without third body
(high pressure)

$$P_r = \frac{k_0[M]}{k_\infty}$$

Coefficient with third body
(low pressure)

Lindemann fall-off

$$k_f(T, P_r) = k_\infty \left(\frac{P_r}{1 + P_r} \right)$$

Troe fall-off

$$k_f(T, P_r) = k_\infty \left(\frac{P_r}{1 + P_r} \right) \times F(T, P_r)$$

$$\log_{10} F(T, P_r) = \frac{\log_{10} F_{cent}(T)}{1 + f_1^2}$$

$$F_{cent}(T) = (1 - A)\exp(-T/T_3) + A\exp(-T/T_1) + \exp(-T_2/T)$$

$$f_1 = (\log_{10} P_r + C)/(N - 0.14\,(\log_{10} P_r + C))$$

$$C = -0.4 - 0.67 \log_{10} F_{cent}$$

$$N = 0.75 - 1.27 \log_{10} F_{cent}$$