CERFACS

EUROPEAN CENTRE FOR RESEARCH AND ADVANCED TRAINING IN SCIENTIFIC COMPUTING

Tools for the numerical simulation with complex chemistry

–

Open-source code CANTERA

Monday 14th November 2022

www.cerfacs.fr

Organisation of the day

9h30 – 10h30 : Talk about Cantera

10h30 – 12h30 : Jupyter notebook tutorials

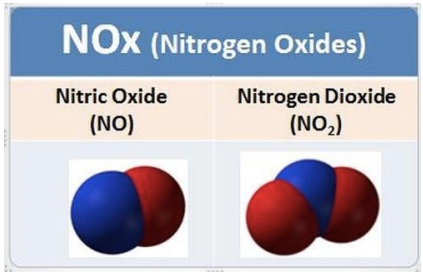
12h30 – 14h : Lunch break

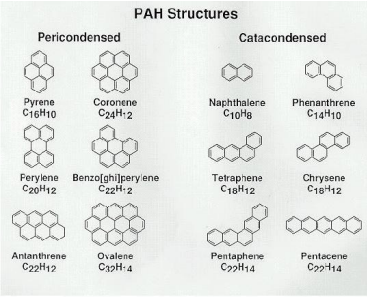
14h – 14h30 : Talk about cantera-avbp features

14h30 – 17h : End of jupyter notebook tutorials + create your own scripts

Why do we need chemistry ?

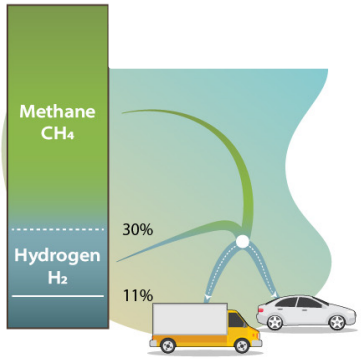
Pollutants





Chemistry driven processes

Ignition



Fuel blending

Content

I. Presentation of CANTERA

II. Governing equations and numerical methods

III. Practical use

Content

What is CANTERA ?

Cantera is an open-source suite of tools for problems involving:

- Chemical kinetics
- Thermodynamics
- Transport processes

Multiple Interfaces :

- Python
- Matlab
- C/C++
- Fortran 90

Broad fields of applications :

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

User friendly :

- Object-oriented
- Easy custom inputs

What is CANTERA ?

Cantera is an open-source suite of tools for problems involving:

- Chemical kinetics
- Thermodynamics
- Transport processes

Multiple Interfaces :

- Python
- Matlab
- C/C++
- Fortran 90

Broad fields of applications :

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

User friendly :

- Object-oriented
- Easy custom inputs

What is CANTERA ?

Cantera was originally written by Dave Goodwin in 2004

- He was a confirmed user of the CHEMKIN suite but disagreed with:
  - the charged software philosophy
  - the fixed structure of CHEMKIN
- Two main idea drive the Cantera development:
  - An open source sofware
  - An object-oriented structure with multiple interfaces
- Currently, Cantera is in version 2.6.0.

At Cerfacs, we use a modified 2.3.0 version !

What is CANTERA ?

Cantera vs ANSYS Chemkin

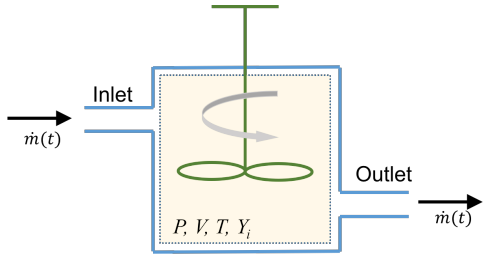
- Cantera replicates most of the Chemkin functionalities and adds new capabilities (multiphase equilibrium, electrochemistry,...)
- Cantera can use Chemkin files through a file converter
- Cantera uses interfaces with scripts whereas Chemkin is based on keywords input files
- The Cantera documentation is scarce, relying on an automatic documentation and an active community

What can CANTERA do ?

Cantera proposes different configurations for the calculations :

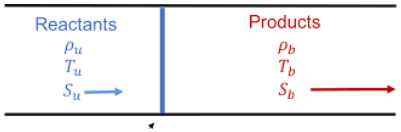
0D

- *Equilibrium State*
- *Constant Pressure/Volume reactor*
- *Steady-state Plug Flow Reactor*

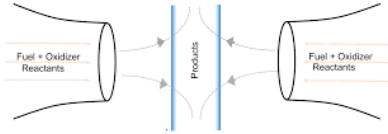


1D

- Freely propagating flame



- Diffusion flame in a counterflow



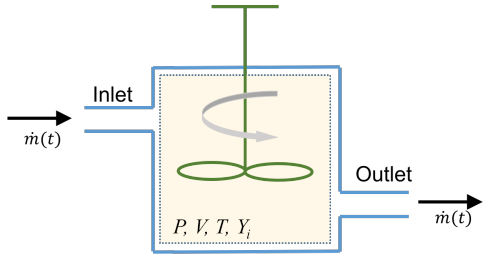
- Premixed strain flame

What can CANTERA do ?

Cantera proposes different configurations for the calculations :

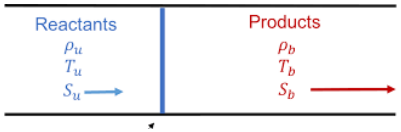
0D

- *Equilibrium State*
- *Constant Pressure/Volume reactor*
- *Steady-state Plug Flow Reactor*

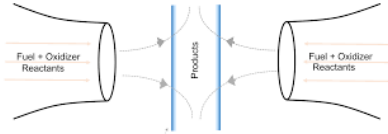


1D

- Freely propagating flame



- Diffusion flame in a counterflow



- Premixed strain flame

and analysis tools :

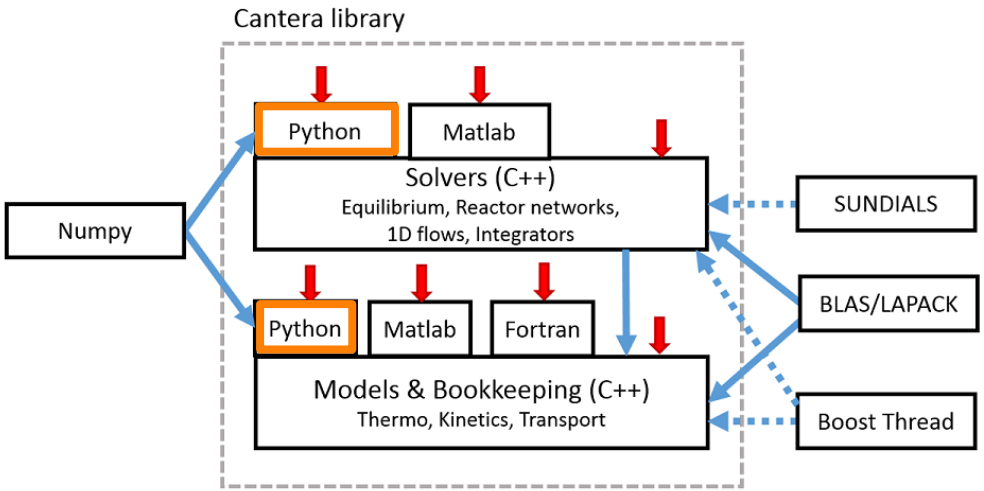
- Easy access to data, Sensitivity Analysis, ...

How do I use Cantera ?

Cantera is used through interfaces in different languages:

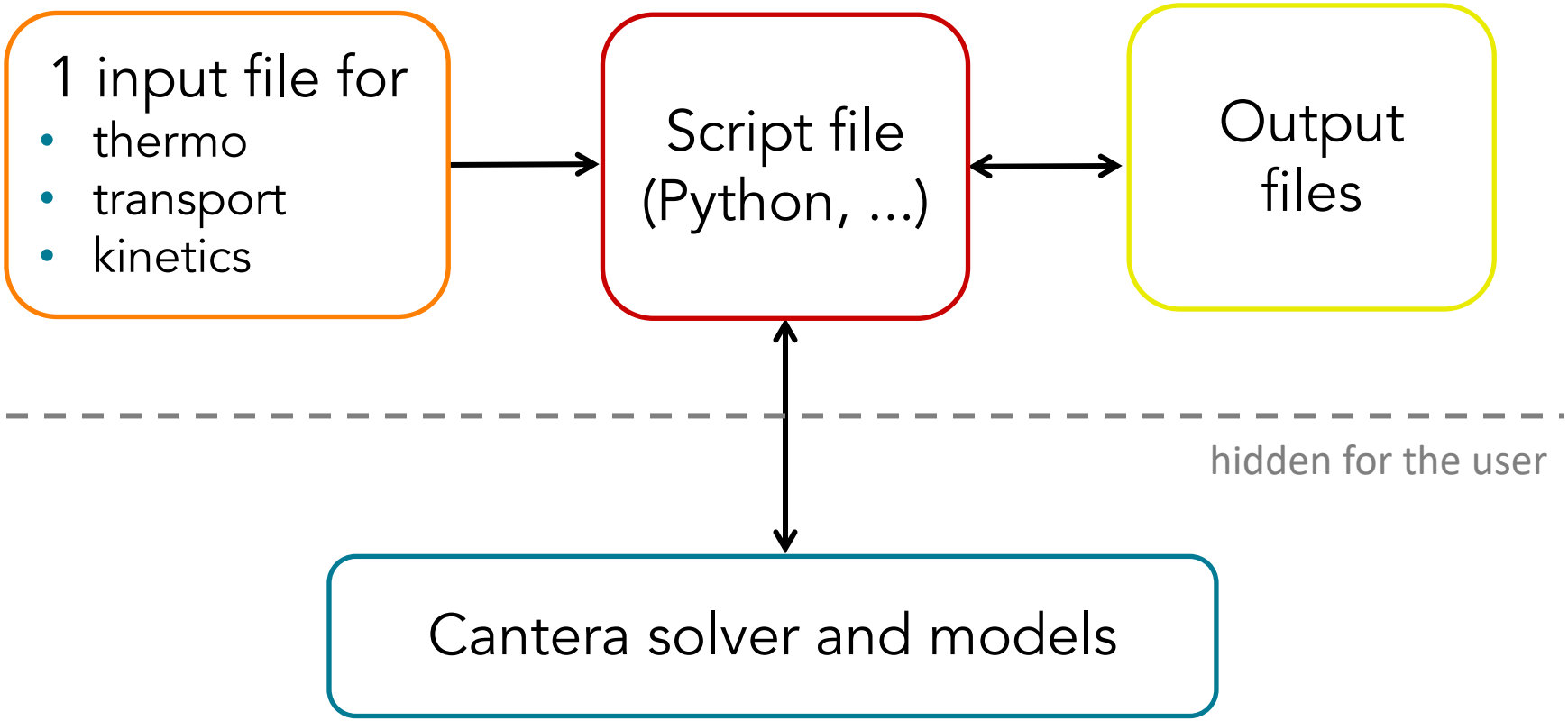
- Python
- Matlab
- C/C++
- Fortran 90

- Those interfaces are only front ends: calculations are done in an optimized, compiled code that is really efficient and fast !!



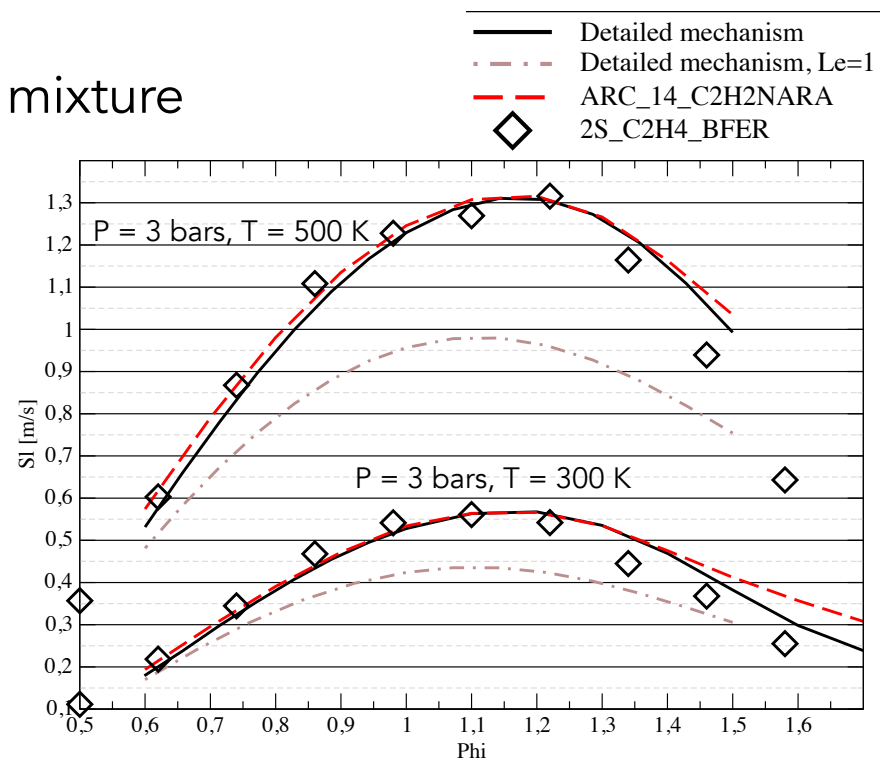
How do I use Cantera ?

Cantera calculations follow then the following structure:



What can I use Cantera for ?

- Compute elementary combustion characteristics:
  - Adiabatic flame temperature
  - Equilibrium composition of a mixture
  - Laminar flame speed
  - …
- Compare models:
  - Transport
  - Thicken flame
- Initialize other codes:
  - 1D flame profiles



Example of laminar flame speed comparison between different mechanisms for C2H4/Air[1]

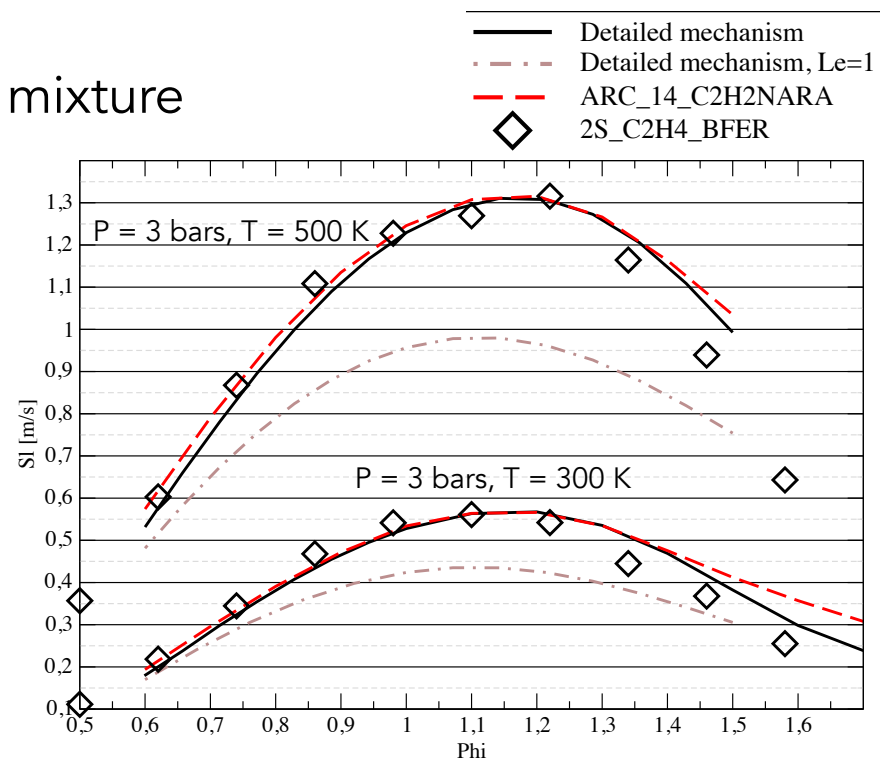
[1] K. Narayanaswamy, G. Blanquart, H. Pitsch "A consistent chemical mechanism for oxidation of substituted aromatic species ". Combustion and Flame, Vol.157 pp. 1879–1898, 2010

What can I use Cantera for ?

- Compute elementary combustion characteristics:
  - Adiabatic flame temperature
  - Equilibrium composition of a mixture
  - Laminar flame speed
  - …

⇒ Cerfacs version features

- Compare models:
  - Transport
  - Thicken flame
- Initialize other codes:
  - 1D flame profiles



Example of laminar flame speed comparison between different mechanisms for C2H4/Air[1]

[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

Content

Governing equations and numerical methods

Three methods are detailed:

- Equilibrium states (0D)
- Homogeneous Reactor (0D with time evolution)
- Laminar premixed flame (1D steady)

Governing equations and numerical methods

Three methods are detailed:

- Equilibrium states (0D)
- Homogeneous Reactor (0D with time evolution)
- Laminar premixed flame (1D steady)

Equilibrium state

The equilibrium state corresponds to a minimum of a property called the energy function under specified conditions

- The Gibbs free energy is defined as

$$G = H - TS$$

- The variation of the Gibbs free energy can be expressed as

$$\Delta G = \Delta H - \Delta(TS)$$

⇒ A reaction spontaneously occurs if $\Delta G < 0$
⇒ A reaction does NOT spontaneously occur if $\Delta G > 0$

⇒ Equilibrium is reached when $\Delta G = 0$

Equilibrium state

- For a mixture, the variation of the total Gibbs free energy of the mixture can be expressed as:

$$dG = \sum_{i=1}^{N} \mu_i \, dn_i = 0$$

  - n_i is the number of moles of component i
  - μ_i is the chemical potential of component i

- With the constraint that the number of moles of every element that must be conserved

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

Equilibrium state

- For a mixture, the variation of the total Gibbs free energy of the mixture can be expressed as:

$$dG = \sum_{i=1}^{N} \mu_i \, dn_i = 0$$

  - n_i is the number of moles of component i
  - μ_i is the chemical potential of component i

- With the constraint that the number of moles of every element that must be conserved
  - ➢ There are no kinetics data, only thermodynamic data are involved !

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

Equilibrium state

- Do we need equilibrium states when doing combustion ?

Yes !

We use :

- Constant T and P equilibrium for the calculation of the *Lower Heating Value (LHV)* [J/kg] :

LHV corresponds to the energy released when Fuel and O_2 are transformed in CO_2 and H_2O:

$$Fuel + \left(x + \frac{y}{4}\right)(O_2 + 3.76\, N_2) \rightarrow xCO_2 + yH_2O$$

$$LHV = \sum_{i}^{C} n_i \bar{h}_i(T_{ref})$$

LHV can be compared to the integral of heat release rate:

$$\int HRRdV = \dot{m}_F \cdot LHV$$

Equilibrium state

- Do we need equilibrium states when doing combustion ?

Yes !

We use :

- Constant H and P equilibrium for *the adiabatic flame temperature T_{ad}* :

$$X_k^{equil} = \frac{P_o}{P} \exp\left(-\frac{g_k^0(T_{ad})}{RT} + \sum_{l=1}^{L} n_{kl} \frac{\lambda_l}{RT}\right)$$

$$\Delta \mathrm{H} = 0$$

$\Rightarrow$ Provides the final gas composition and the adiabatic flame temperature

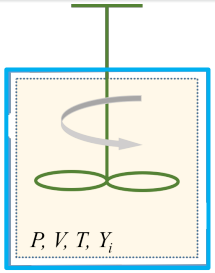
Governing equations and numerical methods

Three methods are detailed:

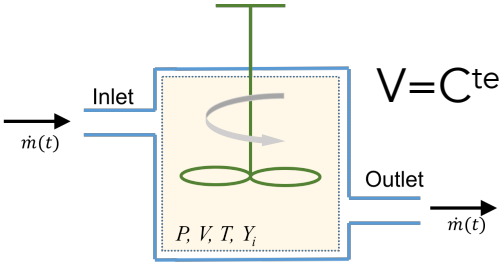
- Equilibrium State (0D)
- Homogeneous Reactor (0D with time evolution)
- Laminar premixed flame (1D steady)

Several types of reactors

- **Batch Reactor** at Constant Volume or at Constant Pressure

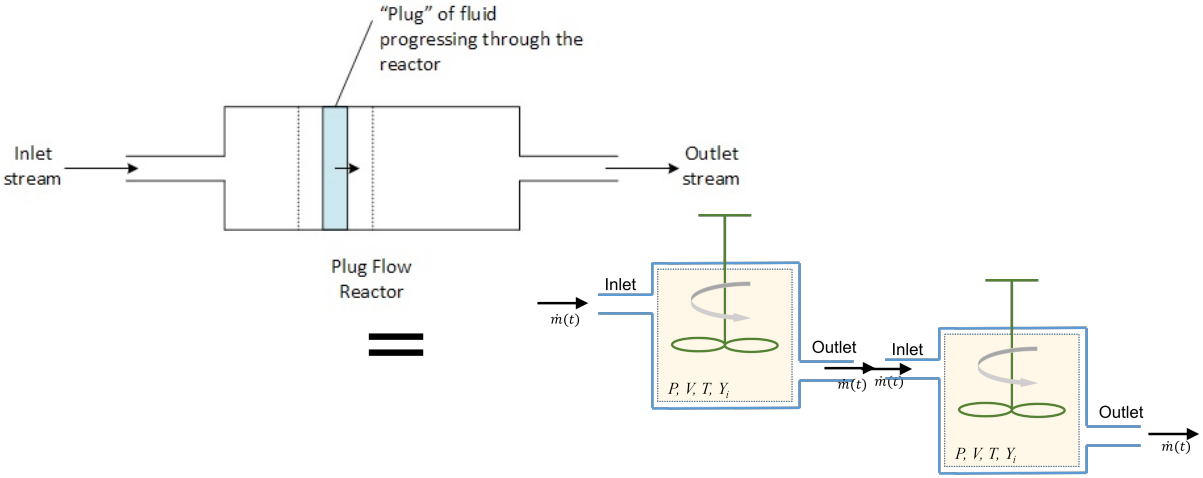


- **Continuously Stirred Tank Reactor (CSTR)** or Well-Stirred Reactor (WSR) or Perfectly Stirred Reactor (PSR) or Longwell reactor



- **Plug-Flow Reactor (PFR)**

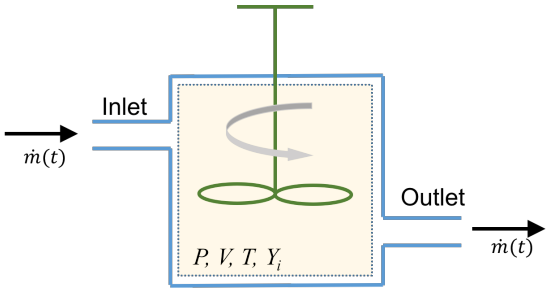
⇒ Does not exist in Cantera
⇒ Is replaced by CSTRs



Lien : https://testing.cantera.org/science/reactors.html

Homogeneous Reactor

Temporal variations can be expressed as:



- Volume: $$\frac{dV}{dt} = \sum_w f_w A_w v_w(t)$$
- Mass: $$\frac{dm}{dt} = \sum_{in} \dot{m}_{in} - \sum_{out} \dot{m}_{out} + \dot{m}_{wall}$$
- Species: $$\frac{d(mY_k)}{dt} = \sum_{in} \dot{m}_{in} Y_{k,in} - \sum_{out} \dot{m}_{out} Y_{k,out} + \dot{m}_{k,gen}$$
- Energy: $$\frac{dU}{dt} = -p\frac{dV}{dt} - \dot{Q} + \sum_{in} \dot{m}_{in} h_{in} - \sum_{out} \dot{m}_{out} h_{out}$$

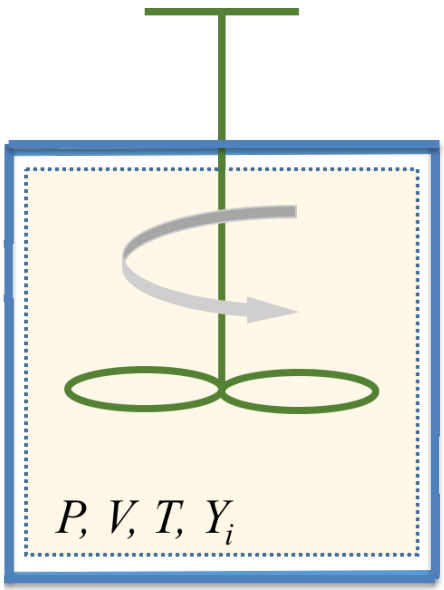
$\Rightarrow$ Stiff system of ODEs that are integrated with a solver from an external library (Sundials CVODE)

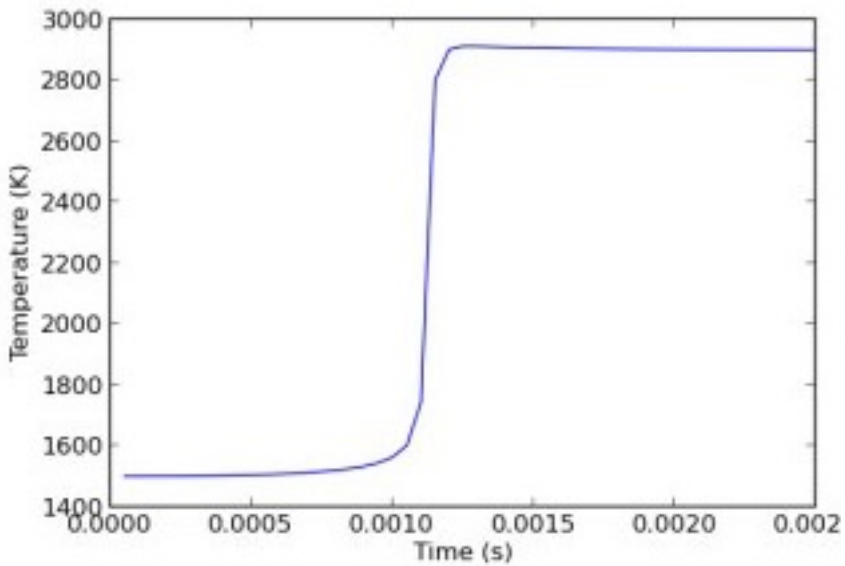
Homogeneous Reactor

- Do we need homogeneous reactor calculation ?

Yes !

The ignition delay time of a mixture can be estimated with constant volume reactor calculation





Governing equations and numerical methods

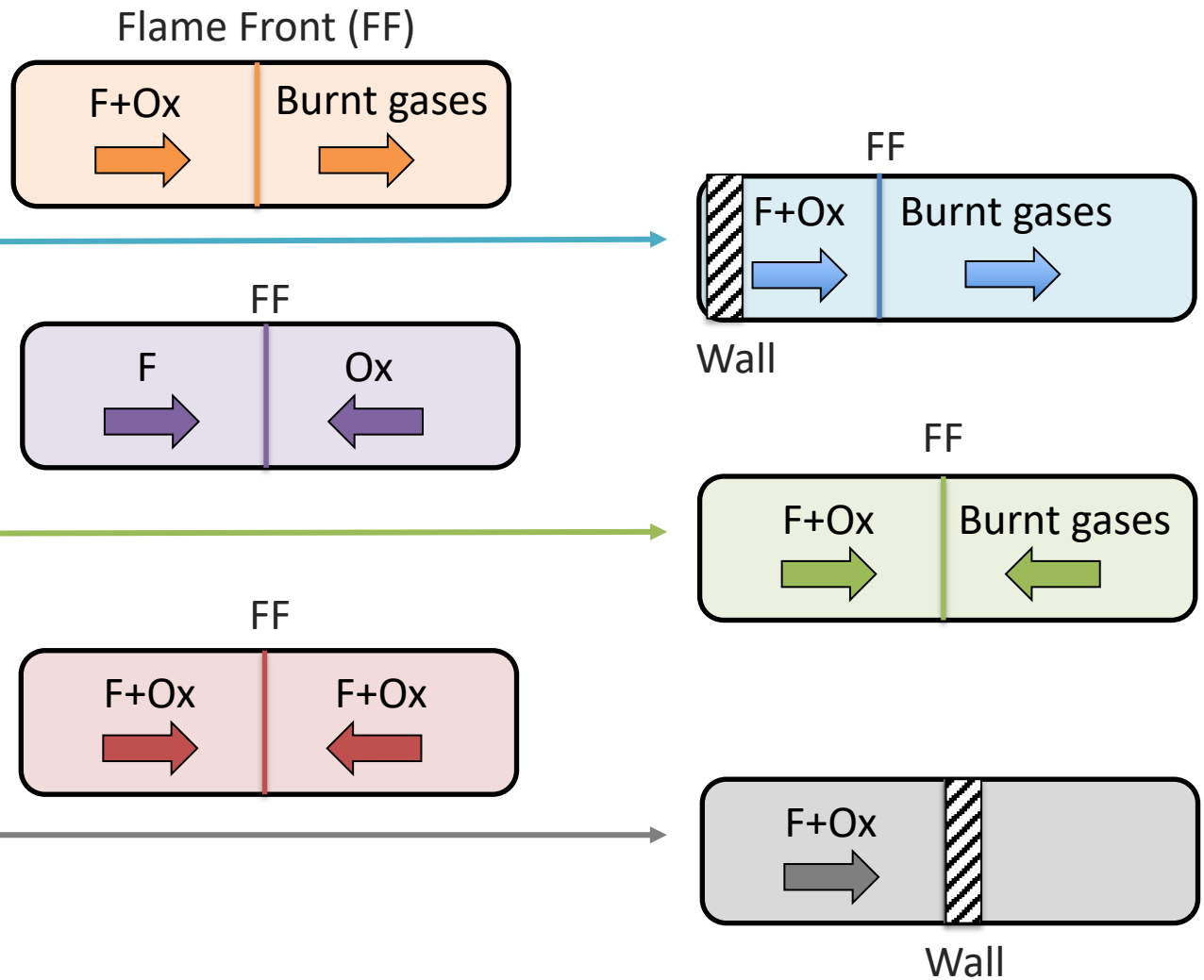
Three methods are detailed:

- Equilibrium State (0D)
- Homogeneous Reactor (0D with time evolution)
- Laminar premixed flame (1D steady)

Several types of flames

Examples provided in the tutorial

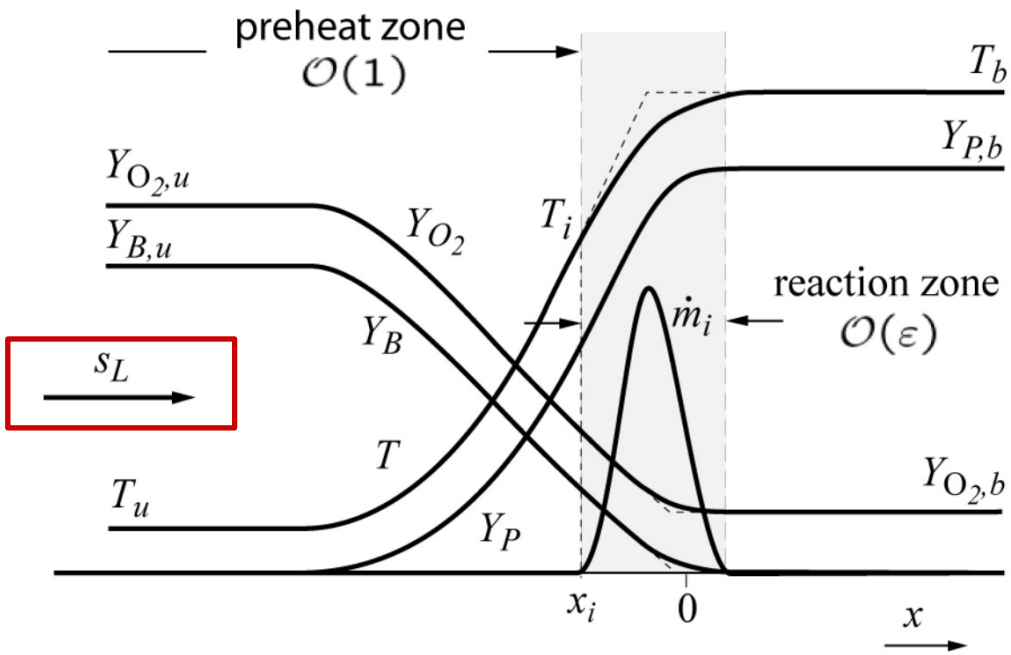
- Freeflame
- Burner flame
- Counterflow diffusion flame
- Counterflow premixed flame
- Counterflow twin premixed flame
- Impinging Jet flame



Lien : https://cantera.org/science/flames.html

Laminar premixed flame

https://cefrc.princeton.edu/sites/g/files/toruqf1071/files/Files/2010%20Lecture%20Notes/Norbert%20Peters/Lecture4.pdf



For a given inlet velocity, called laminar flame speed, the system is in steady state !

Laminar premixed flame

- The system to solve is the following:

$$\cancel{\frac{\partial \rho}{\partial t}} + \frac{\partial \rho u}{\partial x} = 0$$

$$\cancel{\frac{\partial \rho Y_k}{\partial t}} + \frac{\partial}{\partial x}(\rho(u + V_k)Y_k) = \dot{\omega}_k$$

$$\rho C_p \left(\cancel{\frac{\partial T}{\partial t}} + u \frac{\partial T}{\partial x} \right) = \dot{\omega}'_T + \frac{\partial}{\partial x}\left(\lambda \frac{\partial T}{\partial x} \right) - \rho \frac{\partial T}{\partial x} \left(\sum_{k=1}^{N} C_{p,k} Y_k V_k \right)$$

- The system can be written as

$$\mathcal{L}(U_i) = 0$$

Poinsot, Thierry & Veynante, Denis. (2005). Theoretical and Numerical Combustion. 2nd Edition.

Laminar premixed flame

- The system to solve is the following:

$$\cancel{\frac{\partial \rho}{\partial t}} + \frac{\partial \rho u}{\partial x} = 0$$

$$\cancel{\frac{\partial \rho Y_k}{\partial t}} + \frac{\partial}{\partial x}(\rho(u + V_k)Y_k) = \boxed{\dot{\omega}_k}$$

⇒ The chemistry is hidden here !!

$$\rho C_p \left(\cancel{\frac{\partial T}{\partial t}} + u \frac{\partial T}{\partial x} \right) = \boxed{\dot{\omega}'_T} + \frac{\partial}{\partial x}\left(\lambda \frac{\partial T}{\partial x} \right) - \rho \frac{\partial T}{\partial x} \left(\sum_{k=1}^{N} C_{p,k} Y_k V_k \right)$$

- The system can be written as

$$\mathcal{L}(U_i) = 0$$

Poinsot, Thierry & Veynante, Denis. (2005). Theoretical and Numerical Combustion. 2nd Edition.

Laminar premixed flame

- How do Cantera solve the system ?

$$\mathcal{L}(U_i) = 0$$

$\Rightarrow$ Newton solver are well suited to find the roots iteratively

$\Rightarrow$ An initial solution is needed by the Newton solver

- Cantera uses

A Damped modified Newton solver with internal time integration

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 $$\mathcal{L}(U) = 0$$

which is used to iterate $$\frac{\partial \mathcal{L}}{\partial U} = \frac{0 - \mathcal{L}(U)}{U^{m+1} - U^{m}}$$

or in other words $$U^{m+1} = U^{m} - \left[\frac{\partial \mathcal{L}}{\partial U}\right]_{y^{m}}^{y^{m1}} \mathcal{L}(U^{m})$$

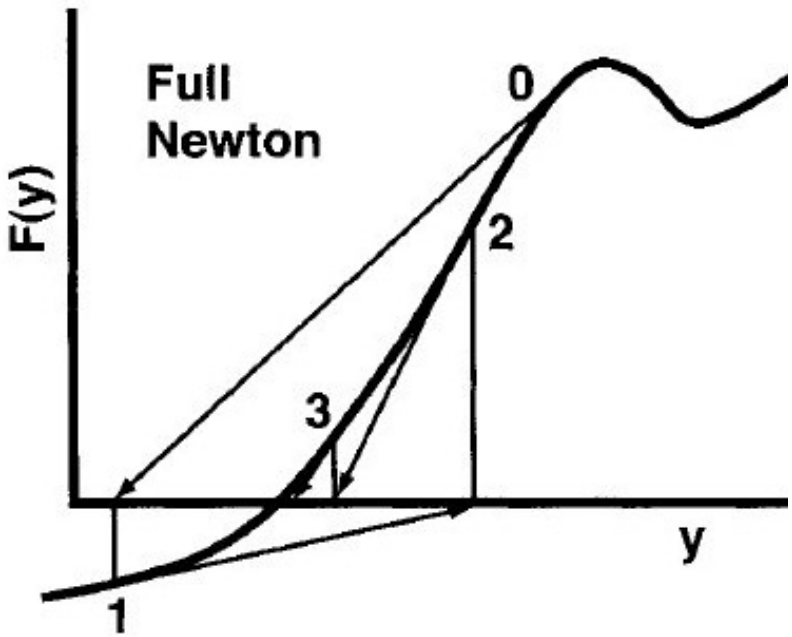
– Convergence is reached when $\Delta U^{m} = U^{m+1} - U^{m}$ becomes negligibly small

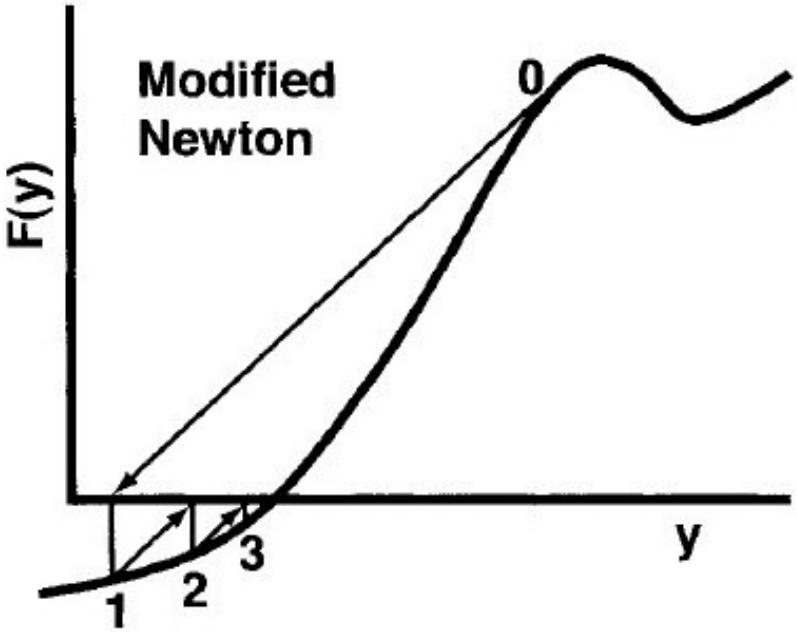
– The mesh might be automatically refined in the region of high gradients

The Damped modified Newton solver

$$\mathcal{L}(U) = 0$$

$$(J^k)\Delta U^m = -\lambda^m \mathcal{L}(U^m)$$





$0 < \lambda < 1$

The Damped modified Newton solver with internal time integration

Whenever both damping parameters and the new Jacobian fail:

- Transient equations are used $\mathcal{L}(\mathrm{U}_{\mathrm{t=n+1}}) = \frac{y_{t=n+1} - y_{t=n}}{\Delta t}$

  (first-order, implicit backward finite differences schemes)

- The Newton method to solve the system of equations for each time step:

$$\mathcal{H}(U) = \mathcal{L}(U) - \frac{dU}{dt} = 0$$

- The new U is used as a new starting estimate for the steady state problem

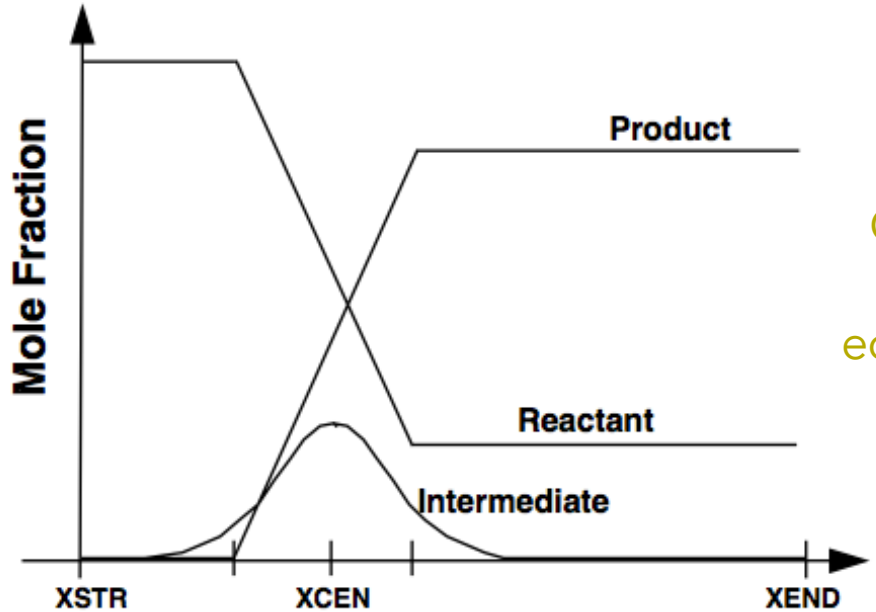
Laminar premixed flame

- What are the inputs required by Cantera?

  ⇒ The Newton solver requires an initial solution

Linear profile in between

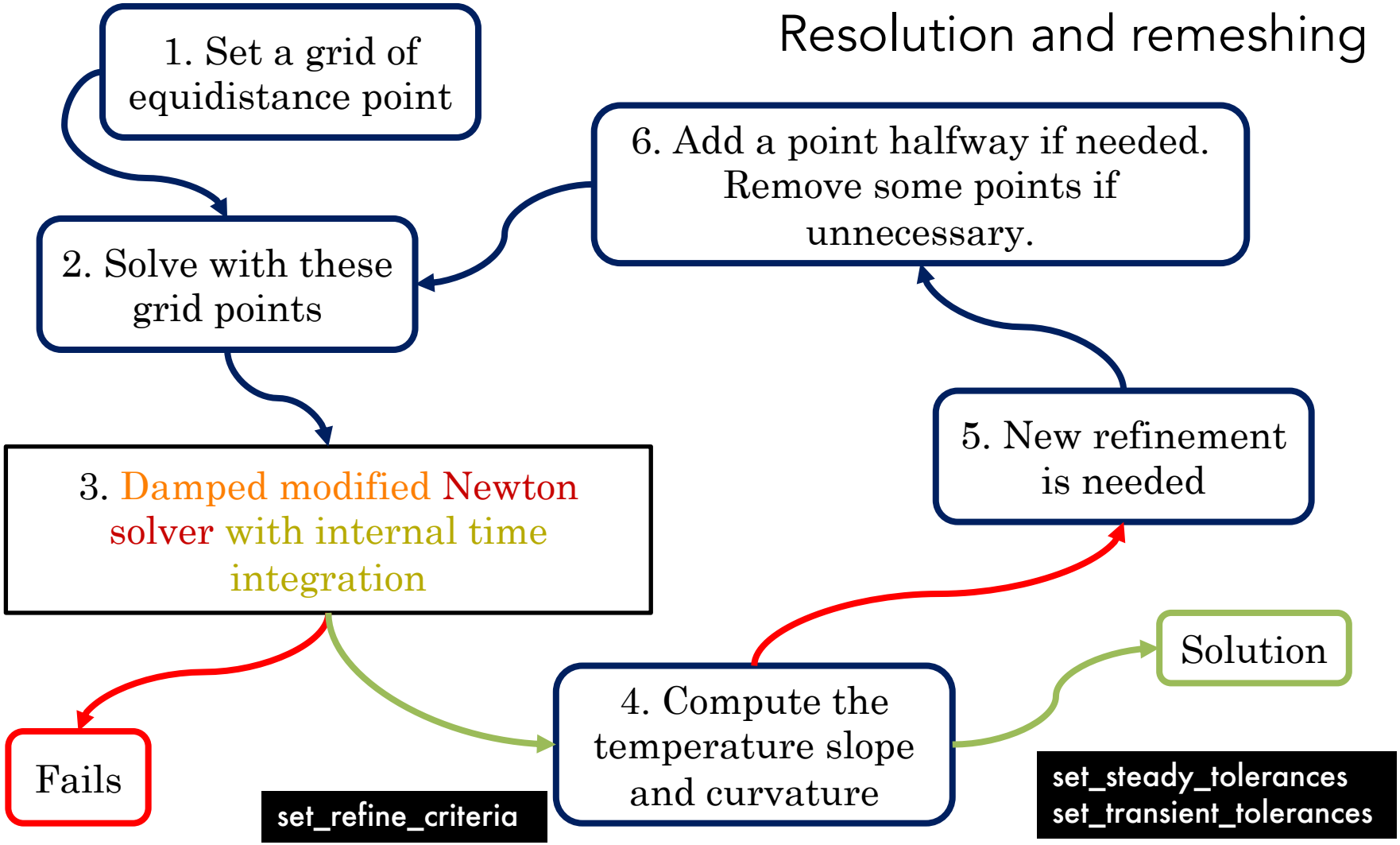
Inlet composition:
set by the user !!



Outlet composition:
determined by
equilibrium calculation

Laminar premixed flame

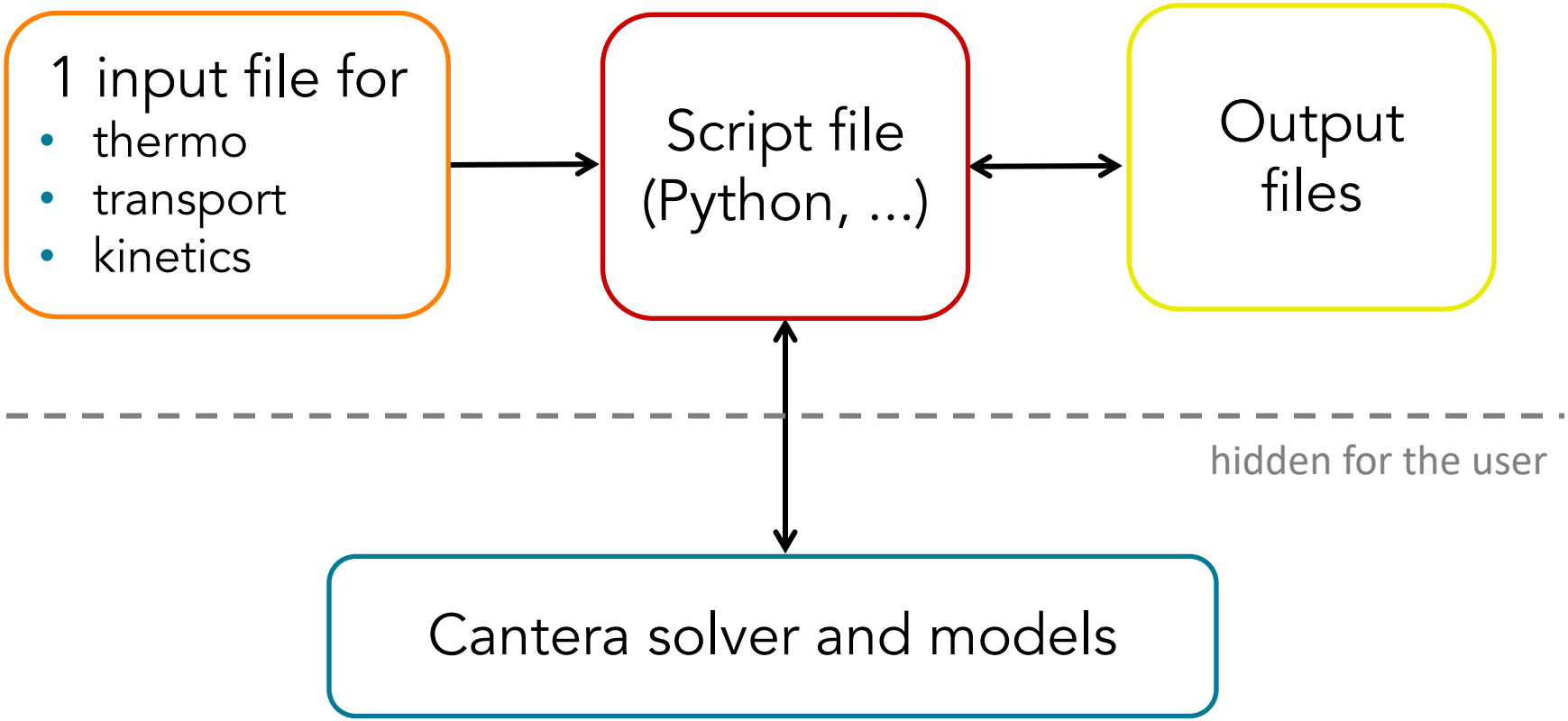
Resolution and remeshing



Content

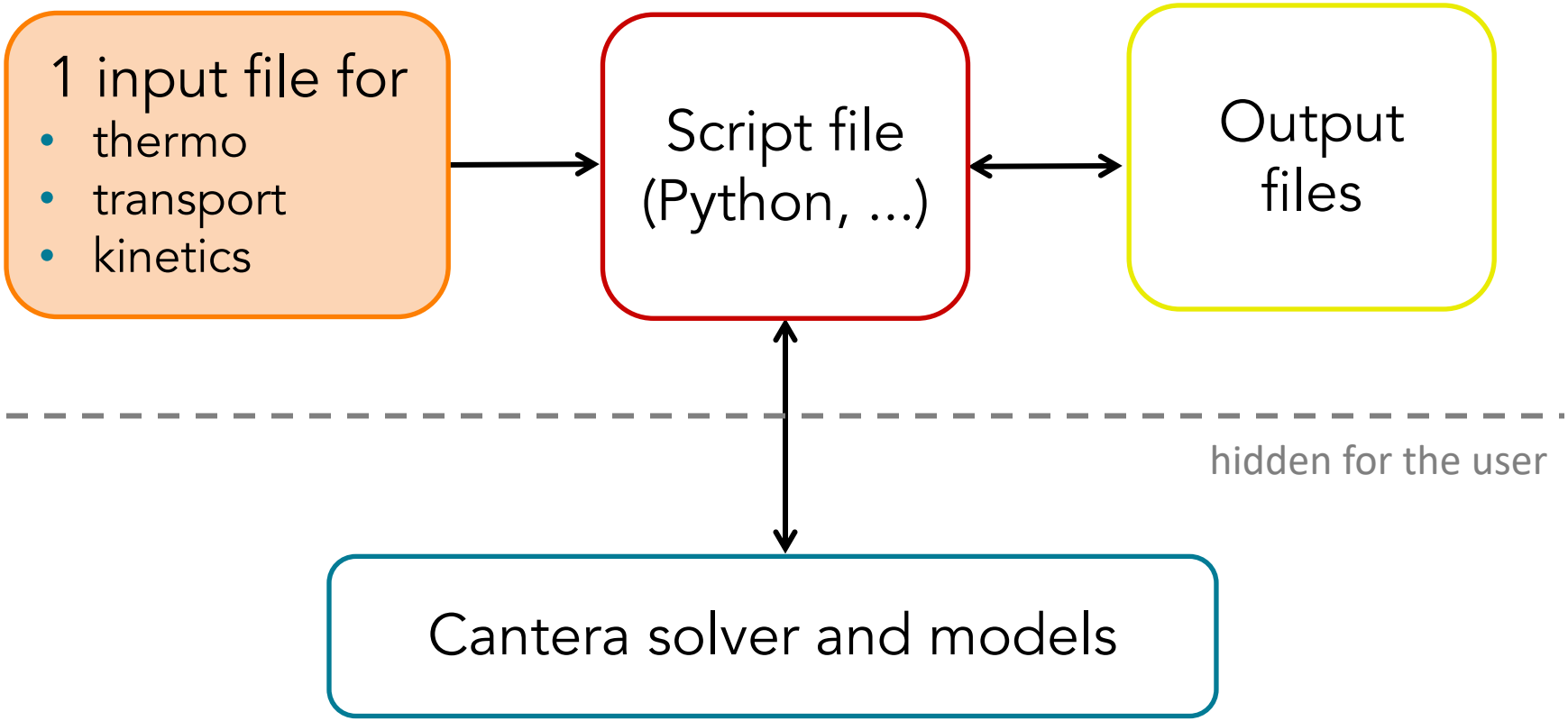
How do I use Cantera ?

Cantera calculations follow then the following structure:



How do I use Cantera ?

Cantera calculations follow then the following structure:



Cantera Input file

- Cantera can read different input files format:
  - cti, default format for version 2.3.0
  - yaml, new default format as of version 2.6.0 !!

- The input file contains information about:
  - Phases and interfaces (species, thermo and transport models, ...)
  - Elements and species data
  - Reactions data (expression, rate coefficients, pressure dependence, ...)

Cantera Input 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

Cantera Input 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

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

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

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

Cantera Input file

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

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

Why do we need that information ? Remember...

$$\cancel{\frac{\partial \rho Y_k}{\partial t}} + \frac{\partial}{\partial x}(\rho(u + V_k)Y_k) = \boxed{\dot{\omega}_k}$$

$\Rightarrow$ The chemistry is hidden ;)

Let's develop $\dot{\omega}_k$:

$\dot{\omega}_k = W_k \sum_{j=1}^{M} \upsilon_{kj}\ Q_j$ with Q_j being the progress rate of reaction j

$Q_j = k_{f,j} \prod_{k=1}^{N} [X_k]^{\upsilon'_{kj}}$ for irreversible reactions

Cantera Input file

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

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

Let's develop $\dot{\omega}_k$:

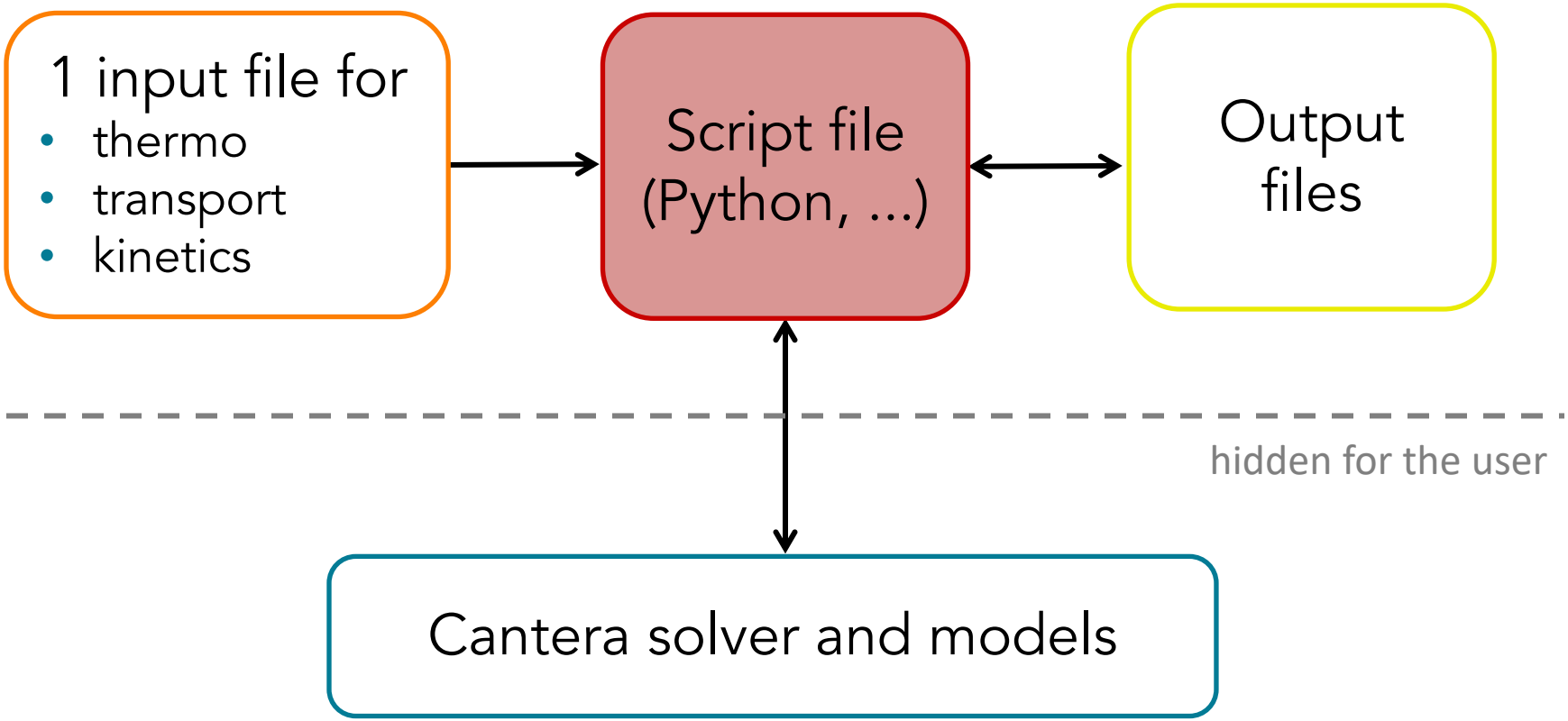
$$\dot{\omega}_k = W_k \sum_{j=1}^{M} \upsilon_{kj}\ Q_j$$ with Q_j being the progress rate of reaction j

$$Q_j = k_{f,j} \prod_{k=1}^{N} [X_k]^{\upsilon'_{kj}}$$ for irreversible reactions

with $$k_{f,i} = A_{f,i}\, T^{\beta_j} \exp(-\frac{E_j}{RT})$$

How do I use Cantera ?

Cantera calculations follow then the following structure:



Script file

AND NOW IT'S TUTO TIME !!!!

Practical use : Helpful links

- Github

https://github.com/Cantera/cantera

- Google Groups page for Cantera

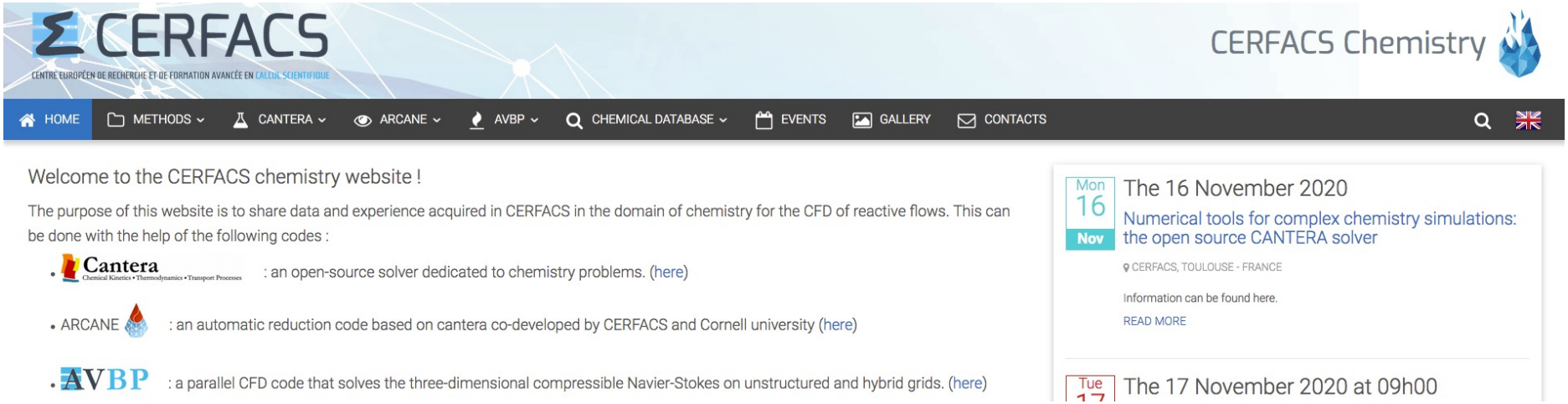
http://groups.google.com/group/cantera-users

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

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

Practical use : Helpful links

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



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

Appendix

1. Parameters for the Newton solver

Parameters for the Newton solver

- Allowed error tolerances for convergence (relative and absolute):

```
f.flame.set_steady_tolerances(default=[1.0E-5, 1.0E-9])          #[rtol atol] for steady-state problem
f.flame.set_transient_tolerances(default=[1.0E-5, 1.0E-9])       #[rtol atol] for time stepping method
```

- Number of times the Jacobian is used before its re-evaluation:

```
f.set_max_jac_age(50, 50)
```

- Time-stepping for the internal time integration:

```
f.set_time_step(1.0E-5, [2, 5, 10, 20, 80])  # Try 2 steps of 1.0E-5 seconds, then if it fails try 5 steps, ...
```

- Grid refinement:

```
f.set_refine_criteria(ratio = 10.0, slope = 1, curve = 1, prune = 0.05)
```

Old 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 (***Keywords***)	A script to arrange « building blocks » into a simulation (***Interface objects and functions***)
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 inpu	
« Inter	
A d … towar	…ing …n …l
Outp	…ript's …tlab, FORTRAN)

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… FOR… +
3 input file…
« Interprete… bi…
A driver… towards si… g
Outputs w… t's b, FORTRAN)

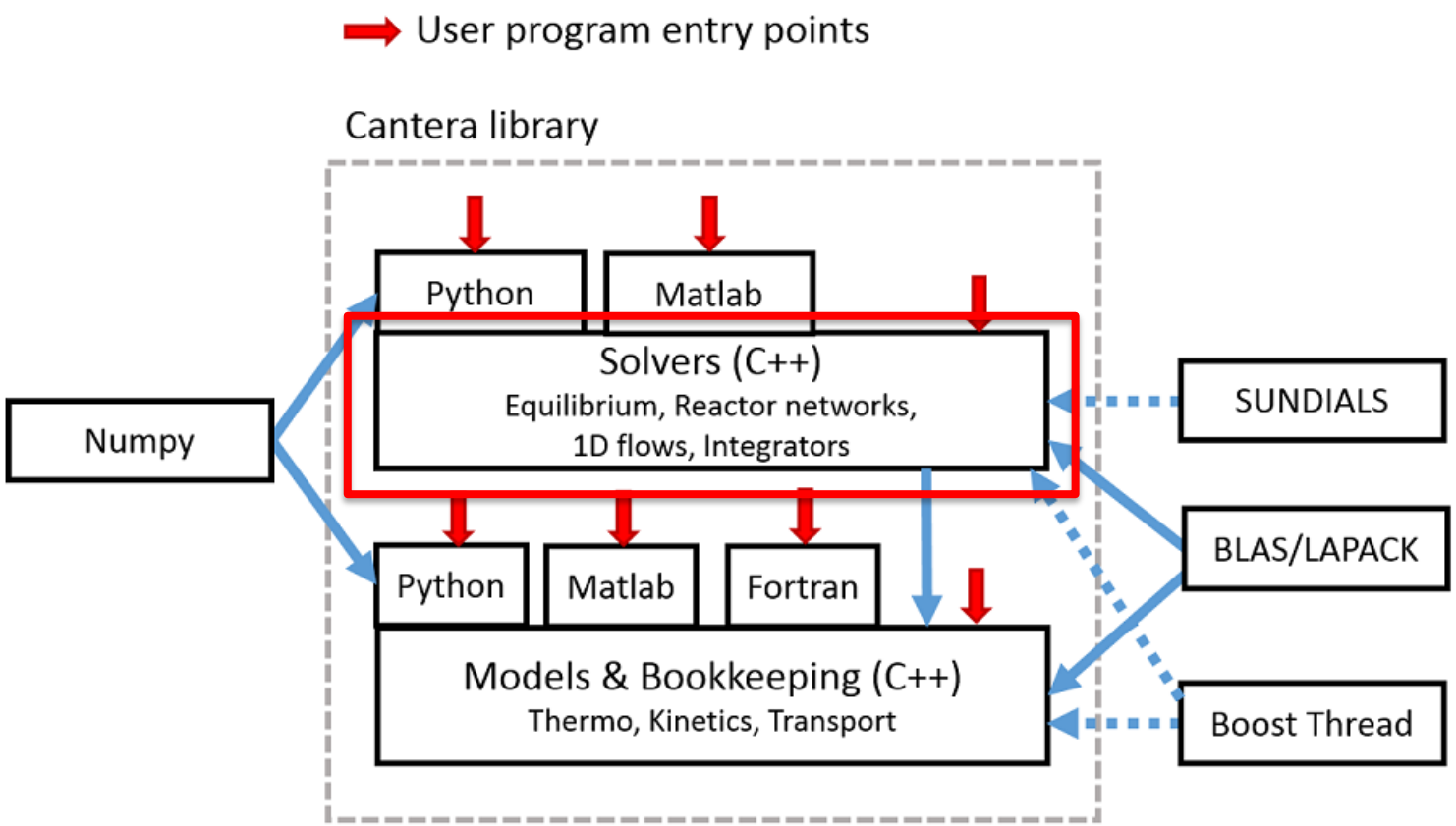
```
"""  Burner Stabilized flame '.py' script. """

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



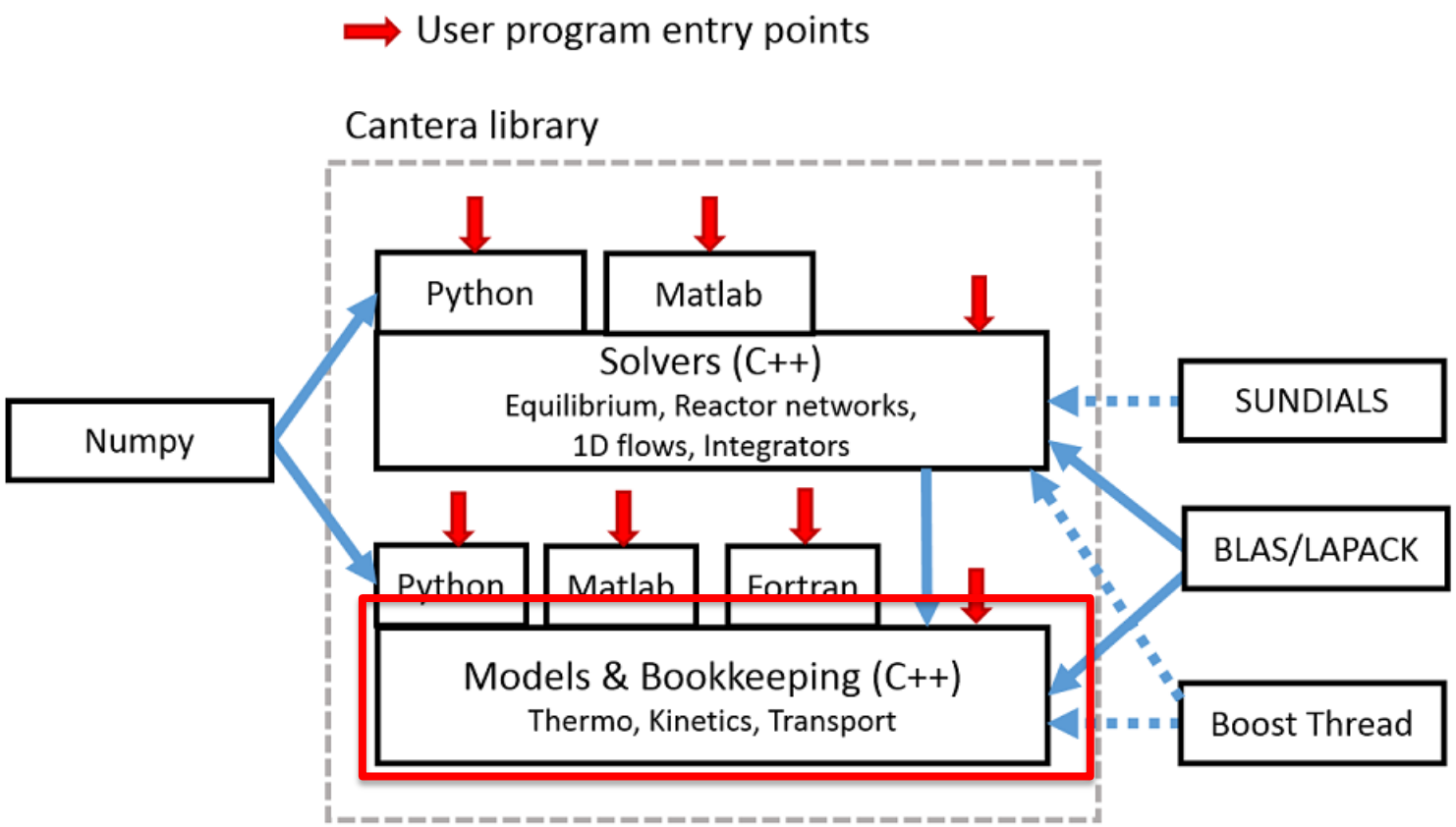
Structure of CANTERA

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

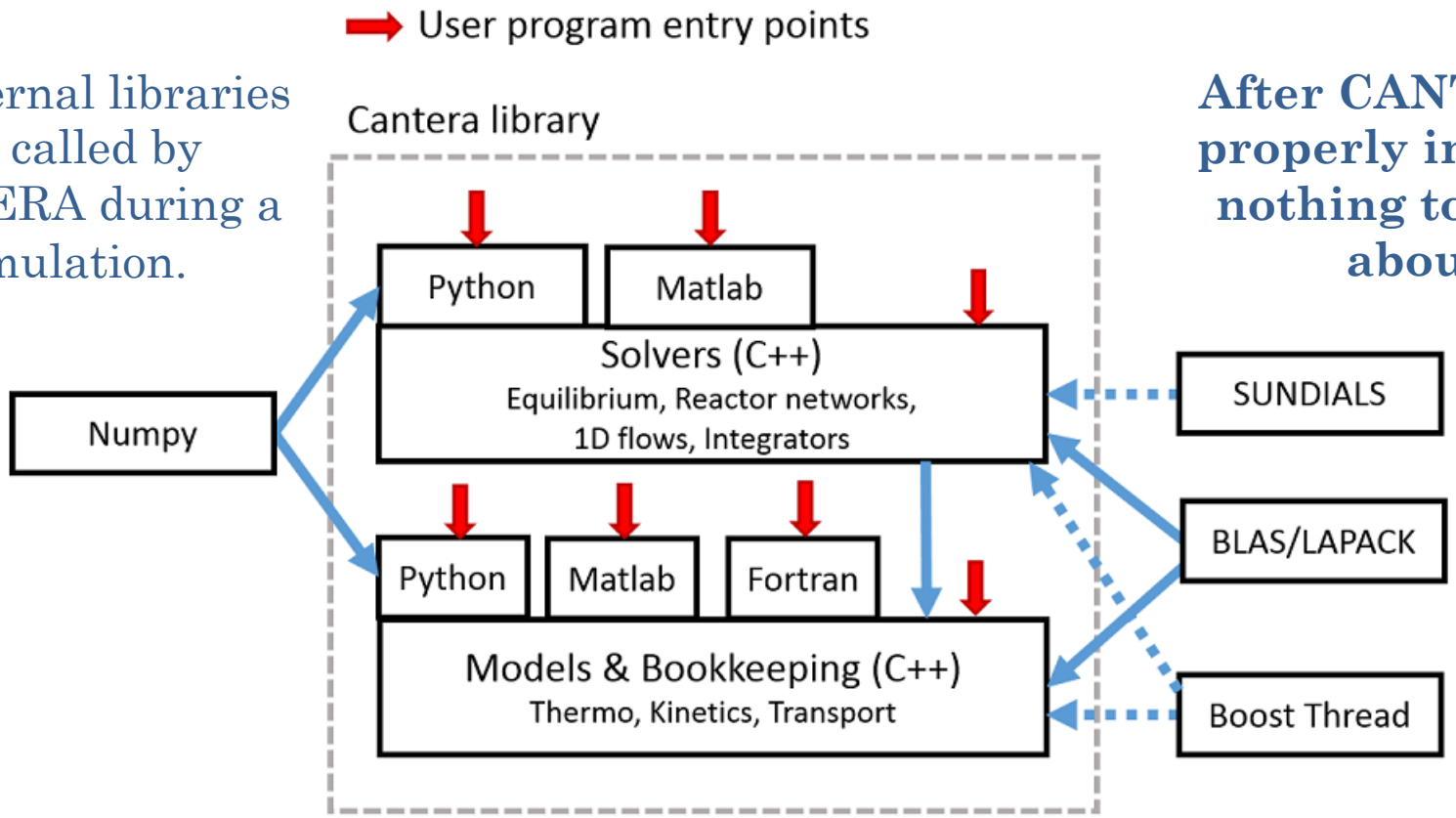
As we have just seen, it is the python script entry.
This layer contains all the methods that will

- **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

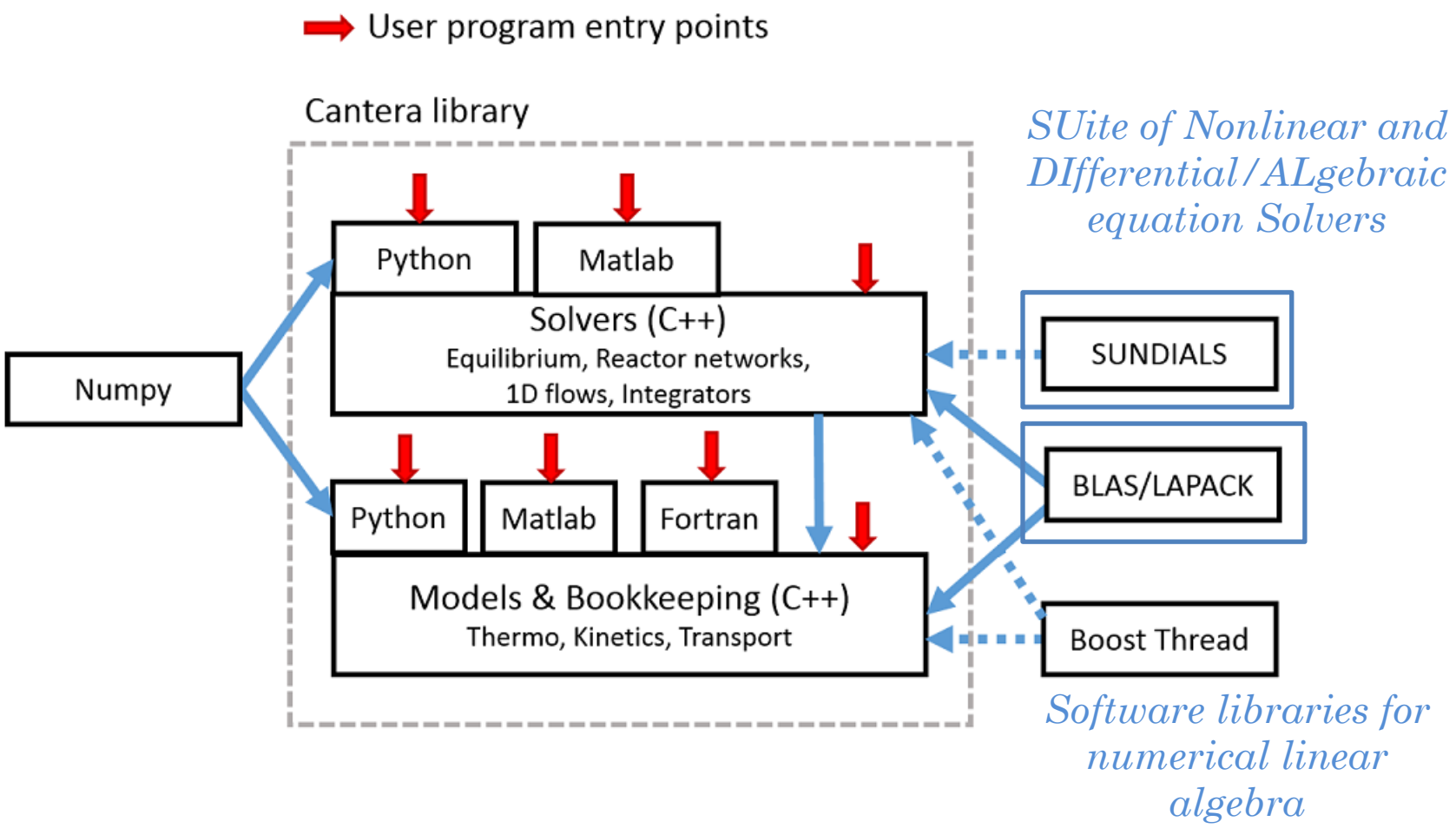
Cantera is a big lasagna, it has layers.

All external libraries are called by CANTERA during a simulation.



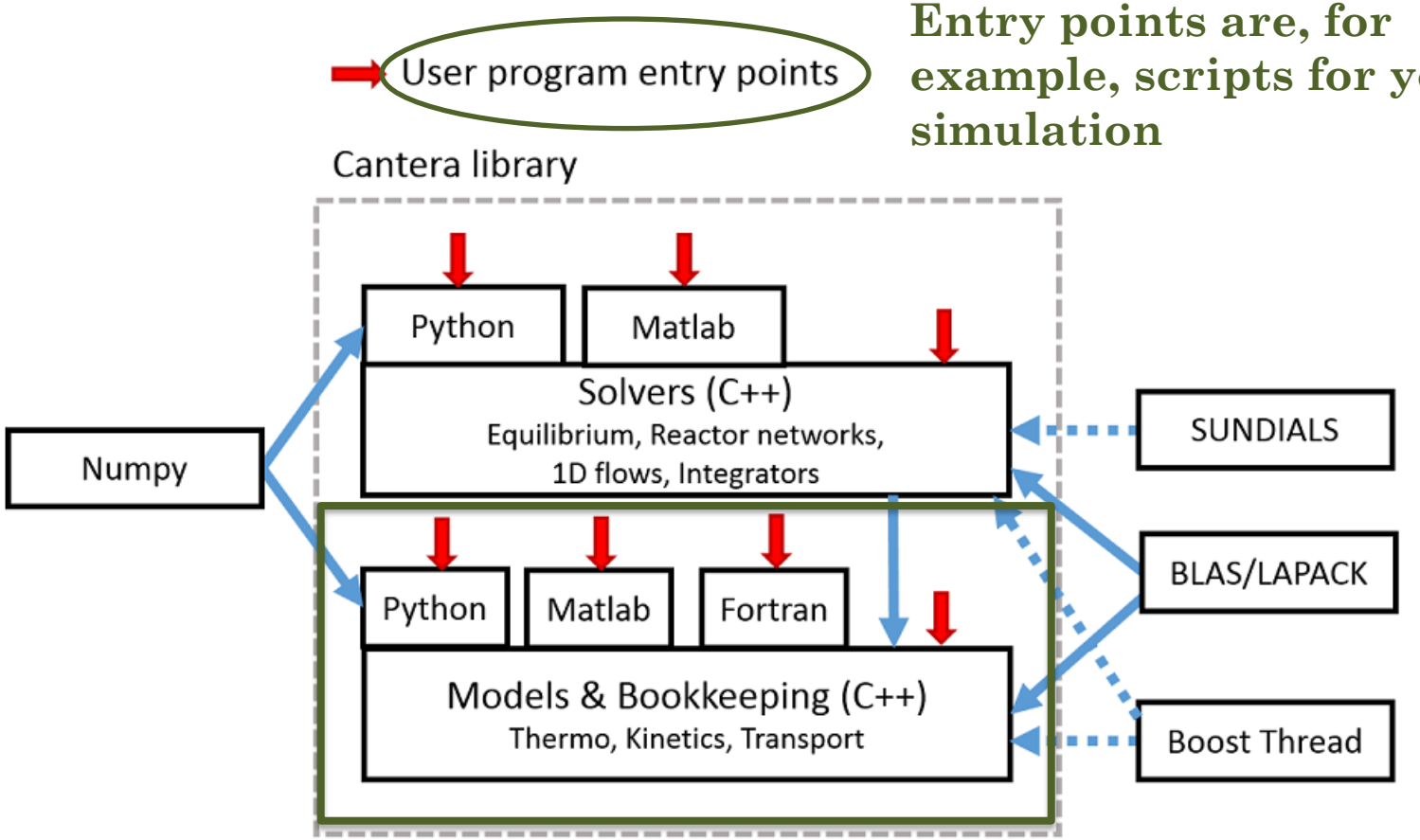
After CANTERA is properly installed, nothing to worry about.

Structure of CANTERA



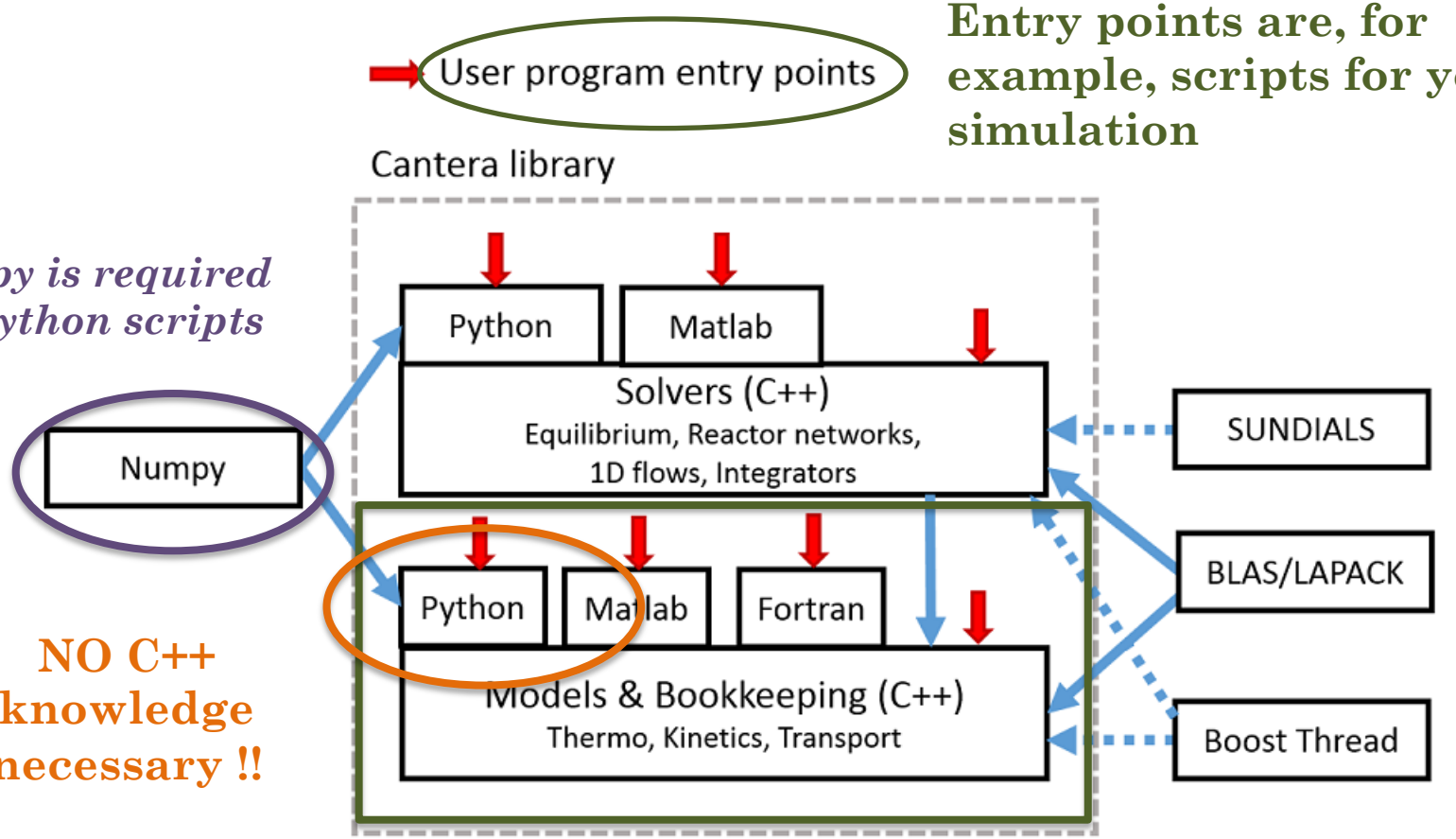
Structure of CANTERA

Entry points are, for example, scripts for your simulation



Structure of CANTERA

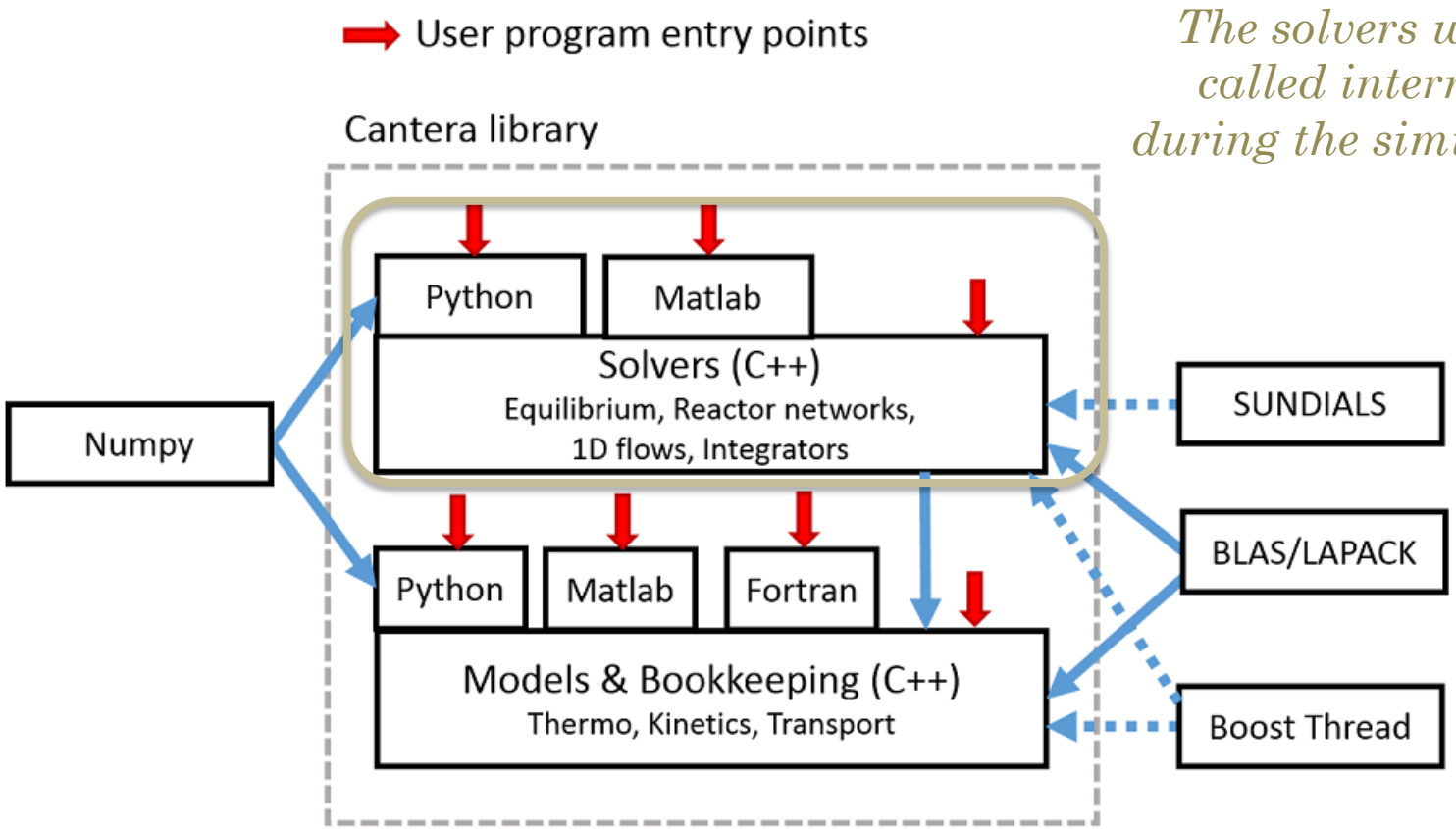
Entry points are, for example, scripts for your simulation



Numpy is required for Python scripts

NO C++ knowledge necessary !!

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 $\mu k = \dfrac{\partial U}{\partial Nk}$

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

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

Example with the Gibbs function

The non-stoichiometric method

This becomes an optimisation problem where

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

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

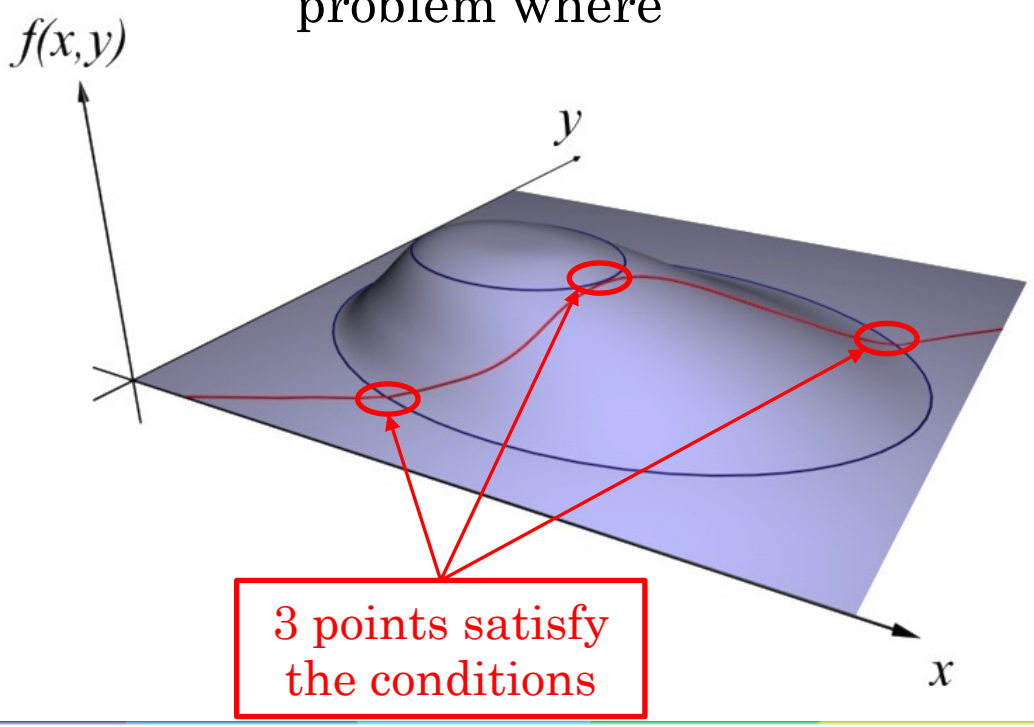


Illustration in 2D

- 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

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

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

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

$$\frac{\partial G^*}{\partial N_k} = \mu_k - \sum_{l=1}^{L} \lambda_l n_{kl} = 0$$

$$\frac{\partial G^*}{\partial \lambda_l} = p_l^* = 0$$

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 LX_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")
```

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 ")
```