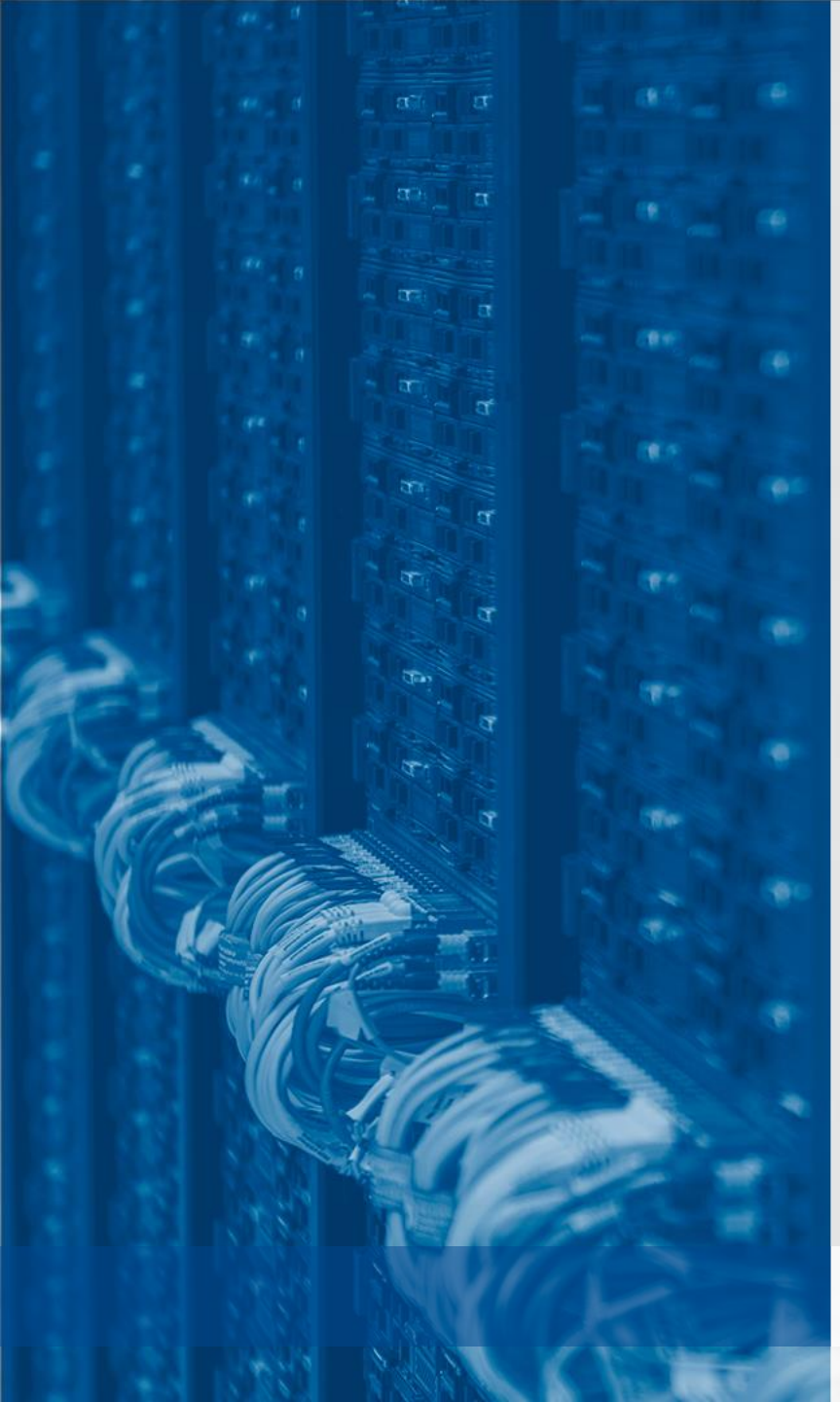
Generalitat de Catalunya
Departament de Recerca i Universitats

GOBIERNO DE ESPAÑA
MINISTERIO DE CIENCIA E INNOVACIÓN

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

UNIÓN EUROPEA
Fondo Europeo de Desarrollo Regional



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Hands on session: Cantera

Emiliano M. Fortes, Daniel Mira

Hands on session: Cantera

- A review of premixed one-dimensional flames theory
  - Setup
  - Solving

- Using Cantera to solve the premixed flame
  - Aided by some code blocks
  - Scripts for H2 flames that you can run during the presentation
  - Exercise to complete from scratch

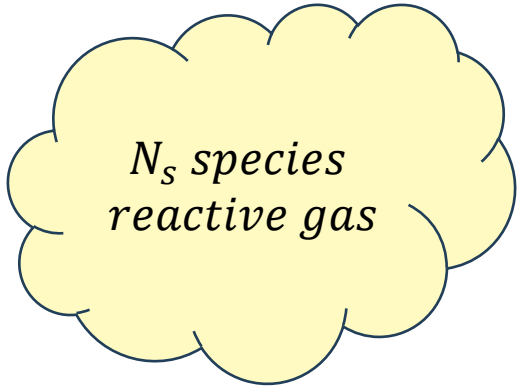
One-dimensional laminar flame

- Goal: Solve transport equations for N_s reactive gas in a 1D domain

Pump fully premixed at ϕ_{mix}

$x = 0$
Dirichlet BC

$x = L$
null Neumann BC

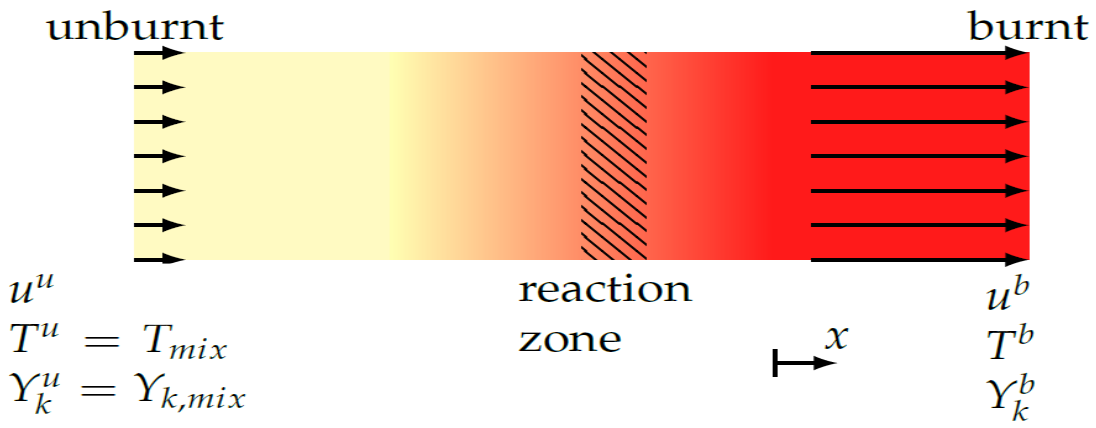


$$
\begin{cases}
\dfrac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0 \\
\dfrac{\partial (\rho Y_k)}{\partial t} + \nabla \cdot (\rho \boldsymbol{u} Y_k) + \nabla \cdot \underbrace{(\rho \mathbf{V}_k Y_k)}_{\mathrm{j}_k} = \dot{\omega}_k \\
\dfrac{\partial (\rho \boldsymbol{u})}{\partial t} + \nabla \cdot (\rho \boldsymbol{u} \otimes \boldsymbol{u}) = -\nabla p + \boldsymbol{\nabla} \cdot \boldsymbol{\tau}, \\
\dfrac{\partial (\rho h)}{\partial t} + \nabla \cdot (\rho \boldsymbol{u} h) + \nabla \cdot \underbrace{\left(-\lambda \nabla T + \rho \sum_{k=1}^{N_s} (\mathbf{V}_k + \mathbf{V}_\mathbf{c}) Y_k h_k\right)}_{\mathrm{j}_h} = 0
\end{cases}
$$

One-dimensional laminar flame

$T_{mix}, p_{mix}, \phi_{mix}$
$\Rightarrow \{Y^u_{mix}\}_{k=1}^{N_s}$

Fresh gas $(300 - 800)K$



$T^b, u^b,$
$\{Y^b\}_{k=1}^{N_s}$

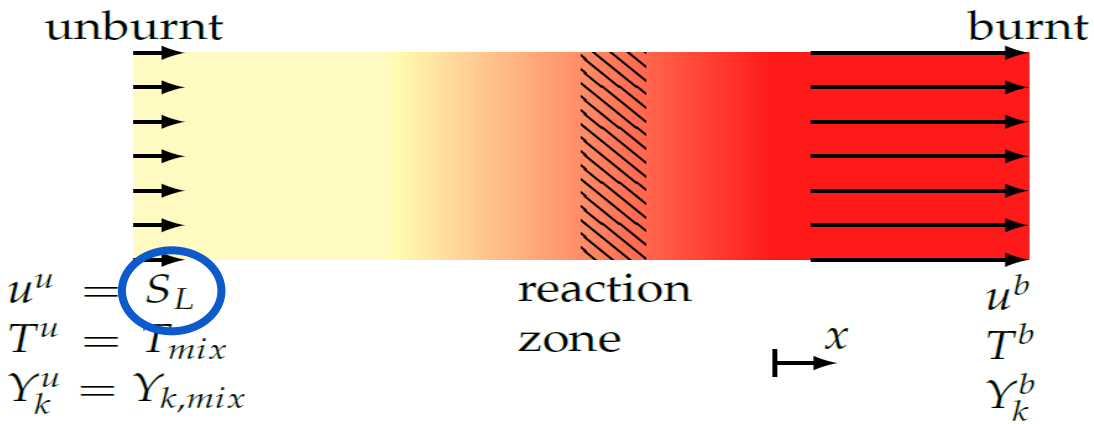
- Seen from reaction zone heat, reaction products, and radicals diffuse towards unburnt mixture
- Through this process, at certain point of the reaction zone T becomes sufficiently high and radicals sufficiently abundant
- Strong reactions occur, consuming the fuel and oxidizer.
- At some time: Steady state; Unburnt gases velocity = s_L
- Two ways of solving the problem

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

FIGURE 3.1 from **High-fidelity numerical simulations of reacting flows with tabulated chemistry,** Ambrus Both Ph.D. thesis

One-dimensional laminar flame (steady)

- Solving the problem in the steady state



$$\cancel{\frac{\partial \rho}{\partial t}} + \frac{\partial}{\partial x}(\rho u) = 0 \;\Rightarrow\; \rho u = \rho^u s_L = cst$$

Laminar flame speed

Characteristic value of the problem: dependent only of the inlet mixture state

- Only left to solve:

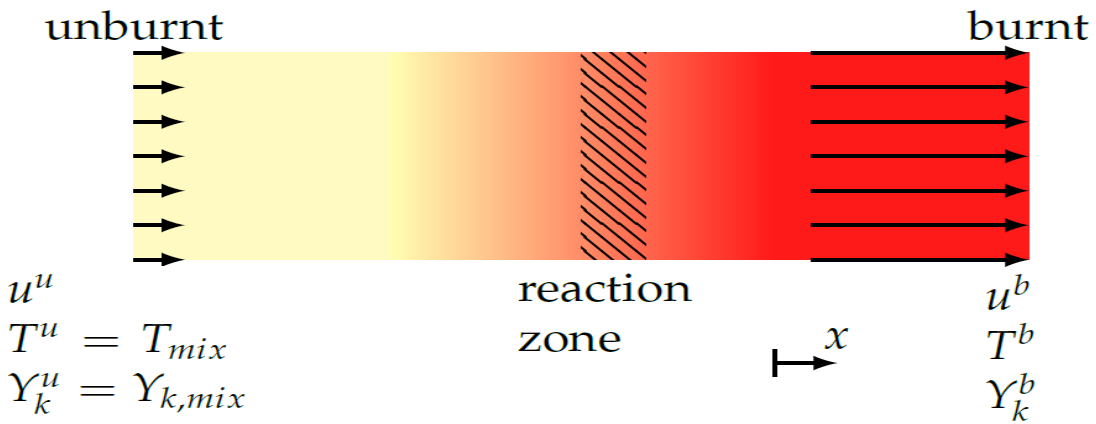
$$\forall k: \quad \frac{\partial}{\partial x}(\rho u Y_k) + \frac{\partial}{\partial x}(\rho \mathrm{V}_k Y_k) = \dot{\omega}_k$$

$$\frac{\partial}{\partial x}(\rho u h) - \frac{\partial}{\partial x}\left(\lambda \frac{\partial T}{\partial x} + \rho \sum_{k=1}^{N_s} (V_k + V_c)\, h_k Y_k\right) = 0$$

- Solving the steady state needs dedicated care
  - A “cold flow” solution exists, where species mass fractions are practically constant
  - 1D solvers take dedicated care in forcing the reacting solution (i.e. Cantera)

One-dimensional laminar flame (unsteady)

- Solving the problem in the unsteady state



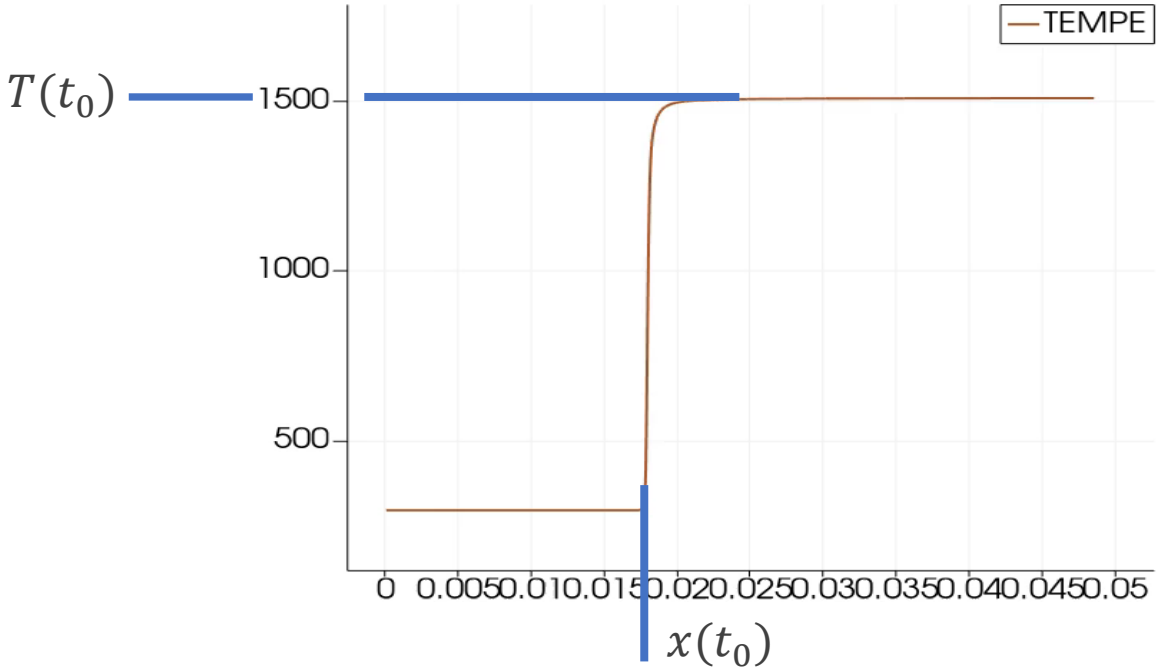
- Inlet velocity u^u is not an unknown
- The reaction zone moves (unless $u^u = s_L$)
- After initial transient state, fluid reaches a moving flame front (if the reacting zone is sufficiently far from boundaries)

- Choose reference frame fixed to the flame front:

$$u_{RF} = u - (u^u - s_L)$$

$$\rho u_{RF} = constant = \rho^u u_{RF}^u = \rho^b u_{RF}^b$$

$$\Rightarrow s_L = \frac{u^b - u^u}{(\rho^u/\rho^b) - 1}$$



Hands-on session: Practice

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

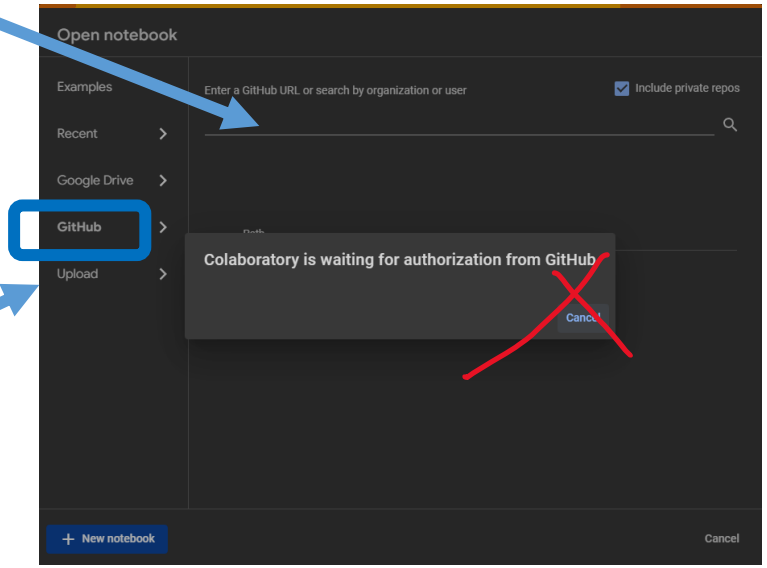
Material

- Material https://github.com/burn-research/ercoftac2023-workshop

- Option 1 (RECOMMENDED - Fully remote with google colab)
  - https://colab.research.google.com/

- Option 2 (Mixed remote with google colab)
  - Download material from ‘Cantera’ directory of Git
  - Upload scripts using the Upload function

- Option 3 (Local installation of python)
  - Requirements: cantera, jupyter, matplotlib, numpy, pandas, tqdm



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
- **User friendly**: Object-oriented with easy custom inputs
- **Broad range of applications**:
  - **Combustion**
  - **Detonation**
  - **Electrochemical energy conversion and storage**
  - **Fuel cells**
  - **Batteries**
  - **Aqueous electrolyte solutions**
  - **Plasmas**
  - **Thin flame deposition**

Cantera: Mixtures

Cantera defines mixtures as Solution objects (A class for chemically-reacting solutions) (0_mixtures.ipynb)

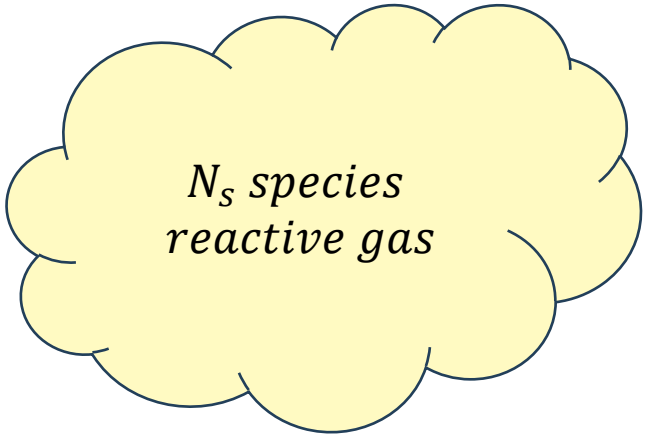
```
import cantera as ct
mechanism = "gri30.yaml"
gas = ct.Solution(mechanism)
# Can epresent any type of solution, i.e. –
# mixture of gases, a liquid solution, or a solid
# solution.
print(gas())
```

```
gri30:

       temperature   300 K
          pressure   1.0133e+05 Pa
           density   0.081894 kg/m^3
  mean mol. weight   2.016 kg/kmol
   phase of matter   gas

                      1 kg            1 kmol
                  ---------------   ---------------
         enthalpy            26469             53361  J
  internal energy      -1.2108e+06        -2.441e+06  J
          entropy            64910        1.3086e+05  J/K
    Gibbs function      -1.9447e+07       -3.9204e+07  J
 heat capacity c_p            14311             28851  J/K
 heat capacity c_v            10187             20536  J/K

                    mass frac. Y      mole frac. X     chem. pot. / RT
                  ---------------   ---------------   ---------------
                 H2                 1                 1          -15.717
     [  +52 minor]                  0                 0
```

N_s *species reactive gas*

- gri30.yaml (GRI-Mech 3.0)
  - Comes installed with Cantera
  - 325 reactions that involve 53 species.
  - Thermochemical properties of species
  - Kinetic reactions

Cantera: Mixtures



$$T_{mix}, p_{mix}, \phi_{mix}\ (Z_{mix}) \Rightarrow \{Y_{mix}^{u}\}_{k=1}^{N_s}$$

- Fuel: H2
- Oxidizer: Air

```
# Define gas properties
T_mix = 300.0 # K
p_mix = ct.one_atm
phi_mix = 0.5
X_fuel = {"H2": 1.0}
X_oxidizer = {"O2": 1.0, "N2": 3.76}
```

```
# Setting gas state method 1
gas.TP = T_mix, p_mix
gas.set_equivalence_ratio(phi_mix, X_fuel, X_oxidizer, basis="mole")
```

DEFAULT MOLE

```
# Setting gas state method 2
X_mix = {"H2": 0.17361, "O2": 0.17361, "N2": 0.65278}
gas.TPX = T_mix, p_mix, X_mix
```

```
# Setting gas state DANGEROUS
gas.TP = T_mix, p_mix
gas.X = X_mix
```

Cantera: Mixtures

(0_mixtures.ipynb)

- h2o2.yaml
  - Comes installed with Cantera
  - Hydrogen-Oxygen submechanism extracted from GRI-Mech 3.0.
  - Reduced to 10 species (much faster than gri30)
  - Modified from the original to include N2.

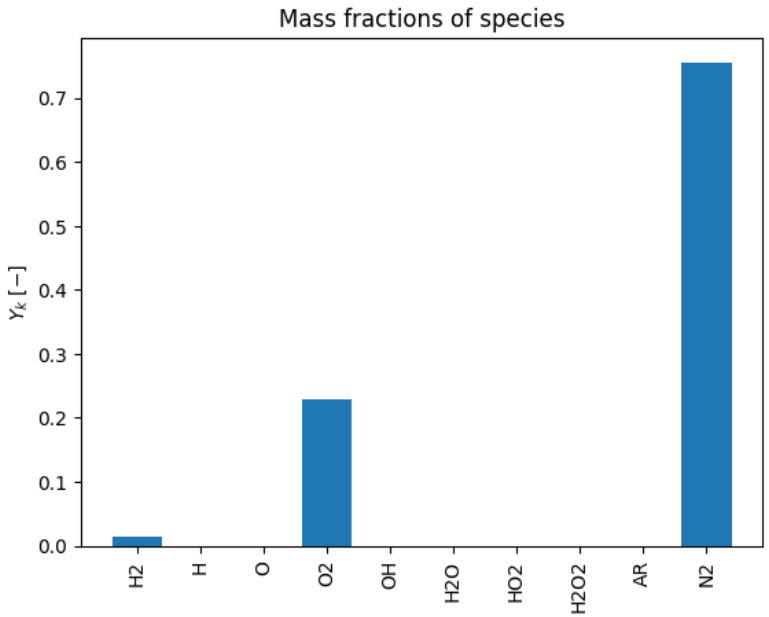
https://www.cerfacs.fr/cantera/mechanisms/hydro.php (More mechanisms and better details)

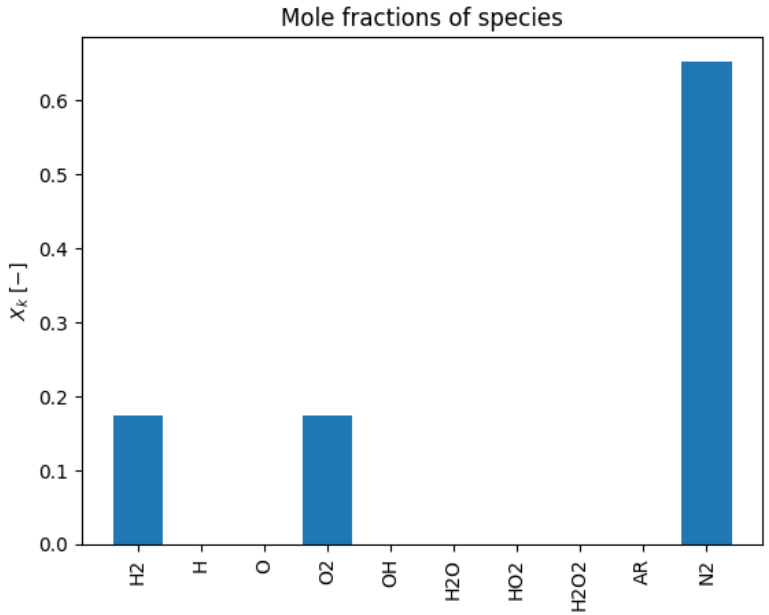
https://cantera.org/tutorials/input-files.html (Converting mechanism formats)

```
T_mix = 300.0
p_mix = ct.one_atm
phi_mix = 0.5
X_fuel = {"H2": 1.0}
X_oxidizer = {"O2": 1.0, "N2": 3.76}
mechanism = "h2o2.yaml"
gas = ct.Solution(mechanism)
gas.TP = T_mix, p_mix
gas.set_equivalence_ratio(phi_mix, X_fuel,
X_oxidizer)

print(gas())
```

Cantera: Mixtures

```
ohmech:

      temperature   300 K
         pressure   1.0133e+05 Pa
          density   0.98273 kg/m^3
 mean mol. weight   24.192 kg/kmol
  phase of matter   gas

                         1 kg            1 kmol
                   ---------------   ---------------
        enthalpy           2262.9             54745  J
  internal energy      -1.0084e+05       -2.4396e+06  J
          entropy          7889.7        1.9087e+05  J/K
   Gibbs function      -2.3646e+06       -5.7206e+07  J
heat capacity c_p          1202.5             29091  J/K
heat capacity c_v           858.8             20776  J/K

                    mass frac. Y      mole frac. X     chem. pot. / RT
                   ---------------   ---------------   ---------------
              H2         0.014468          0.17361           -17.468
              O2          0.22963          0.17361           -26.425
              N2           0.7559          0.65278            -23.46
    [   +7 minor]                0                0
```





Barcelona Supercomputing Center
Centro Nacional de Supercomputación

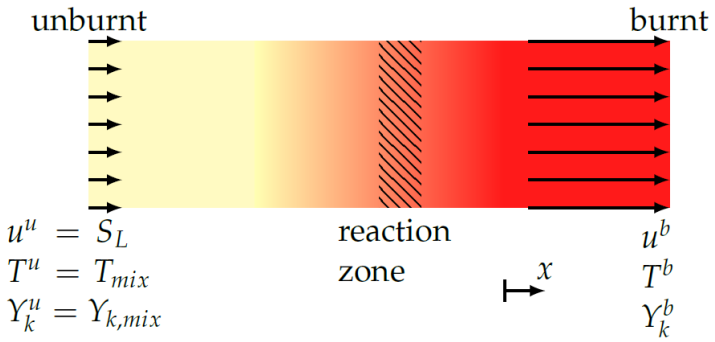
Cantera: Flames

- Cantera defines flames as Flame objects which NEED a Solution objects
- This tutorial: Premixed flames

```
# Define gas properties
T = 300.0
p = ct.one_atm
phi = 0.5
X_fuel = {"H2": 1.0}
X_ox = {"O2": 1.0, "N2": 3.76}
```

```
# Setting gas state
gas.TP = T, p gas.set_equivalence_ratio(phi, X_fuel, X_ox)
```

Create object:
ct.FreeFlame



Cantera: Flames

- Cantera defines flames as Flame objects which NEED a Solution objects
- This tutorial: Premixed flames (1_first_flame.ipynb)

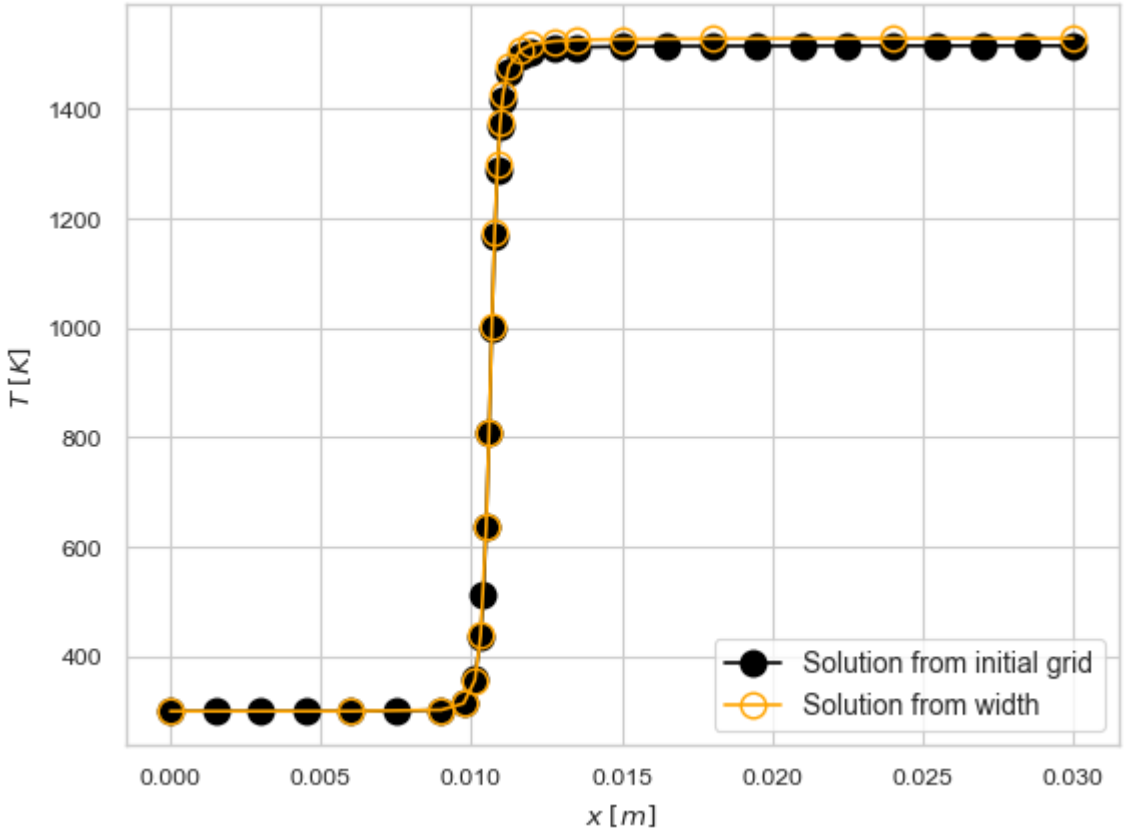
```
# Define gas properties
T = 300.0
p = ct.one_atm
phi = 0.5
X_fuel = {"H2": 1.0}
X_ox = {"O2": 1.0, "N2": 3.76}
mechanism = "h2o2.yaml"
```

```
width = 0.03 # m
npoints = 19 # grid points - 1
initial_grid = [x * width / npoints for x in range(npoints+1)]

gas1 = ct.Solution(mechanism)
gas1.TP = T, p
gas1.set_equivalence_ratio(phi_mix, X_fuel, X_ox)

flame1 = ct.FreeFlame(gas1, grid=initial_grid)
flame1.solve(loglevel=1)
```

```
width = 0.03 # m

gas2 = ct.Solution(mechanism)
gas2.TP = T, p
gas2.set_equivalence_ratio(phi, X_fuel, X_ox)

flame2 = ct.FreeFlame(gas2, width=width)
flame2.solve(loglevel=1)
```

CAREFUL: After solving, Cantera changes gas object thermochemical state

Cantera: Flames



- Plot flame.grid vs flame.T
- Adaptive mesh refinement

Disgression: Transport models

Diffusive term of species

$$\frac{\partial(\rho Y_k)}{\partial t} + \nabla \cdot (\rho \boldsymbol{u} Y_k) + \nabla \cdot \underbrace{(\rho \mathbf{V}_k Y_k)}_{\mathrm{j}_k} = \dot{\omega}_k$$

- **Multicomponent species diffusion**

To determine $\mathbf{V}_k$: $\forall p = 1, N_s$:

$$\nabla X_p = \sum_{k=1}^{N_s} \frac{X_p X_k}{\mathcal{D}_{pk}} (V_k - V_p) + (Y_p - X_p) \frac{\nabla P}{P} + \frac{\rho}{p} \sum_{k=1}^{N_s} Y_p Y_k (f_p - f_k)$$

- Complex and costful
  - Linear system of size N_s^2
  - Solve at every time
  - Solve at every physical space point

- **Mixture-averaged approximation**
  - Hirschfelder and Curtiss approximation: best first-order approximation to the exact resolution

$$V_k X_k = -D_k \nabla X_k, \qquad D_k = \frac{1 - Y_k}{\Sigma_{j \neq k} X_j / \mathcal{D}_{jk}}$$

$$\Rightarrow \frac{\partial \rho Y_k}{\partial t} + \frac{\partial \rho u_i Y_k}{\partial x_i} = \frac{\partial}{\partial x_i} \left(\rho D_k \frac{W_k}{W} \frac{\partial X_k}{\partial x_i} \right) + \dot{\omega}_k$$

Similar but different to Fick's Law $\left(\rho D_k \frac{\partial Y_k}{\partial x_i} \right)$

- **Unity-Lewis approximation**

Ratio of thermal diffusivity and mass difussivity

$$Le_k = \frac{\lambda}{\rho c_p D_k}$$

$$\forall k: 1, N_s: \quad Le_k = 1 \Rightarrow D_k = \frac{\lambda}{\rho c_p}$$

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Recommended reads:
Theoretical and Numerical Combustion – Thierry Poinsot, Denis Veynante
https://cantera.org/science/flames.html

Cantera: Transport models

- Cantera offers the three options
  - Setting up transport model (Before 3.0)
    - “mixture-averaged” (“Mix”)
    - “multicomponent” (“Multi”)
    - “unity-Lewis-number” (“UnityLewis”)

- By default, cantera selects mixture average approximation

```
gas = ct.Solution(mechanism)
print(gas.transport_model)
```

```
'mixture-averaged’
```

- Two ways of setting it up

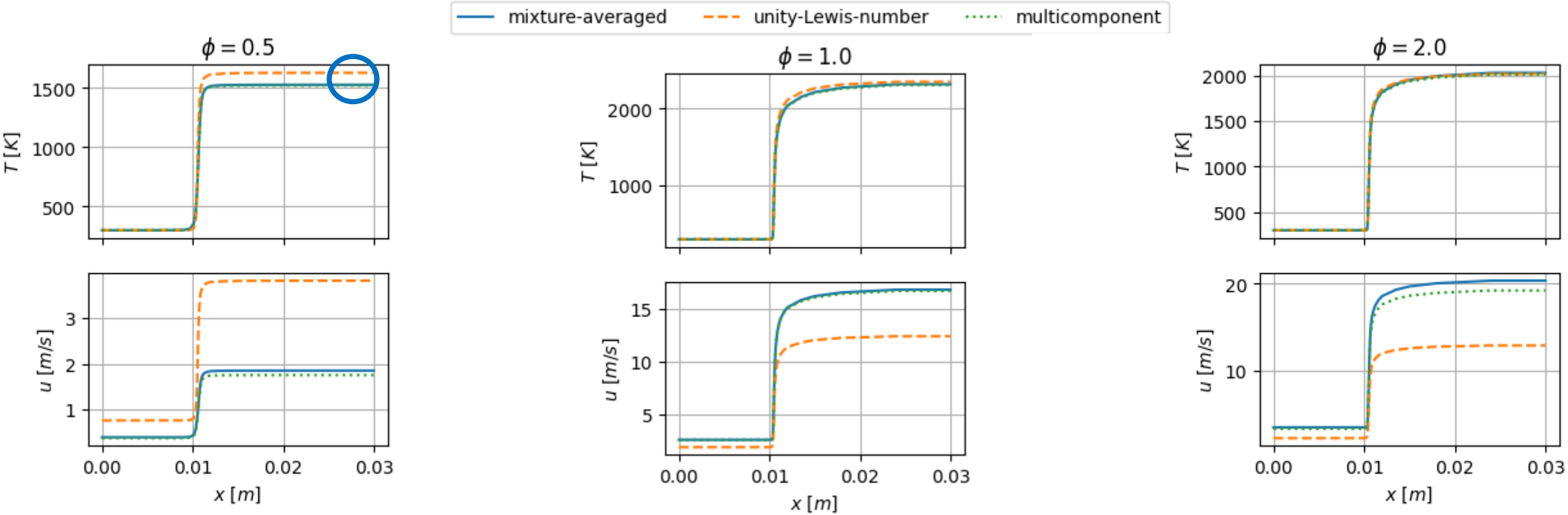
```
# 1) Directly to the mixture object
gas.transport_model = “multicomponent”
```

```
# 2) Set the flame object

gas = ct.Solution(mechanism)
flame = ct.FreeFlame(gas, width=0.3)
flame.transport_model = “multicomponent”
print(gas.transport_model)
```

```
‘multicomponent’
```

Cantera: Transport models

- (2_flame_vs_transport_model.py)
- Part 1) Play around with parameters T, p, ϕ



Cantera: Numerics

- Summary of (2_flame_vs_transport_model.py)

```
transport_models = ["mixture-averaged", "unity-Lewis-number", "multicomponent"]
```

```
flames = {}
properties = {}
for transport_model in transport_models:
    gas = ct.Solution(mechanism)
    gas.TP = T, p gas.set_equivalence_ratio(phi, X_fuel, X_ox)

    flame = ct.FreeFlame(gas, width=width)
    flame.transport_model = transport_model
    flame.solve(loglevel=1)
    flames[transport_model] = flame
```

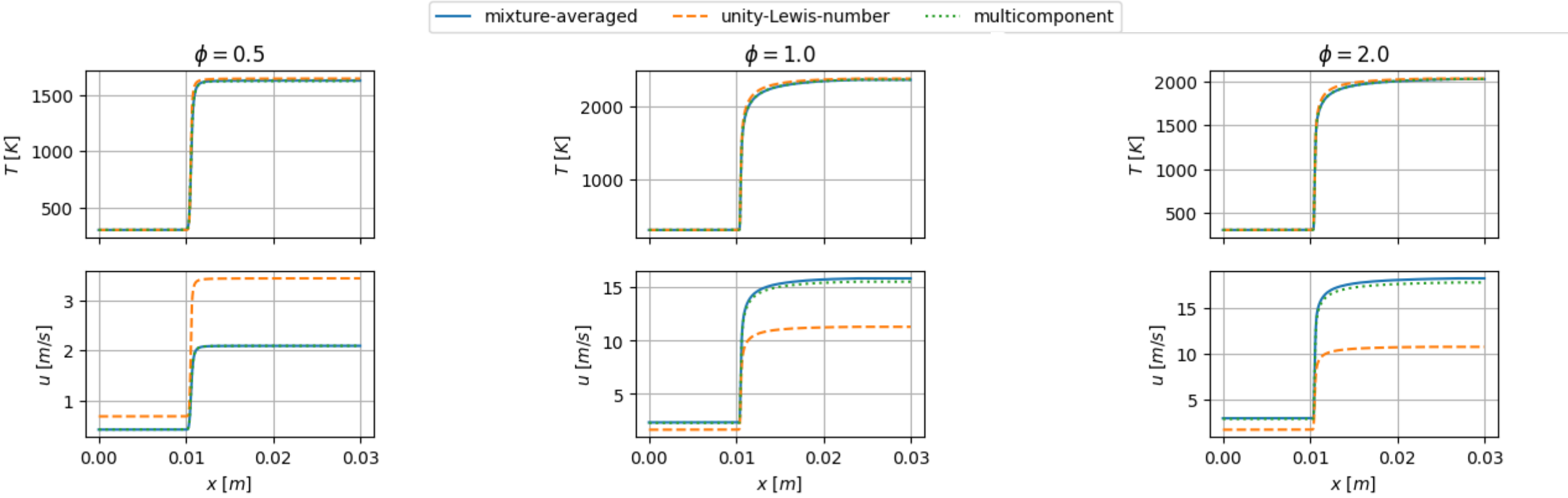
```
flame.energy_enabled = True # Solve T equation

# Number of times reusing Jacobian before recalculation
flame.set_max_jac_age(10, 10)

# Try 10 steps of 1e-8 s, then if it fails try 20 steps …
flame.set_time_step(1e-8, [10, 20, 40, 80, 100, 100, 150])

# [rtol, atoll] for steady-state problem
flame.flame.set_steady_tolerances(default=[1.0e-5, 1.0e-9])
flame.flame.set_transient_tolerances(default=[1.0e-5, 1.0e-9])

# Max timestep
flame.max_time_step_count = 5000

flame.set_refine_criteria(ratio=2, slope=0.05, curve=0.05,
prune=0.01) # Grid refinement criteria
```

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Details at https://coec-project.eu/training/ercoftac-course-understanding-and-predicting-hydrogen-combustion/ (Hands-on 1 Hydrogen Flames With Cantera)

Cantera: Transport models

- (3_flame_vs_transport_model_better_numerics.py)
- Part 1) Play around with parameters T, p, ϕ



Cantera: Mixture fraction

- 1D Flames are the ideal fully premixed combustion setup
  - Completely characterized by equivalence ratio ϕ
  - Bilger, Stårner, and Kee (1990) introduced the local mixture fraction
  - Elements are not created/destroyed by chemical reactions (C, H, O)

From elemental mass fractions

$$Z_{mass,m} = \sum_{k}^{Ns} \frac{a_{m,k} W_m}{W_k} Y_k \Rightarrow b = 2\frac{Z_{mass,C}}{W_C} + 0.5\frac{Z_{mass,H}}{W_H} - \frac{Z_{mass,O}}{W_O},$$

number of atoms of element m in species k

```
# From mixture
Z_H = gas.elemental_mass_fraction("H")  # A unique value

# From the flame
Z_H = flame.elemental_mass_fraction("H") # Across all points in grid
```

- Given any fuel + oxidizer, calculate b_{fuel} and $b_{oxidizer}$

$$Z(\phi) = \frac{b(\phi) - b_{oxidizer}}{b_{fuel} - b_{oxidizer}}$$

Cantera: Mixture fraction

$$Z(\phi) = \frac{b(\phi) - b_{oxidizer}}{b_{fuel} - b_{oxidizer}} = Z_{st} + \frac{1}{b_f - b_o} \sum_{k=1}^{N_C} b_k Y_k,$$

where

$$b_k = \left(2\frac{a_{C,k}}{W_k} + 0.5\frac{a_{H,k}}{W_k} - \frac{a_{O,k}}{W_k} \right).$$

It follows that

$$\boxed{\phi = \frac{Z}{1-Z}\frac{1-Z_{\mathrm{st}}}{Z_{\mathrm{st}}}}$$

Finally, using the fact that Z =linear combination of Y_k $\Rightarrow \frac{\partial(\rho Z)}{\partial t} + \nabla \cdot (\rho \boldsymbol{u} Z) + \nabla \cdot (\mathbf{j}_Z) = 0$

- Exercise at home $Z_{in} = Z_{out}$
- The same argument holds for enthalpy $h_{in} = h_{out}$

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Details at https://cantera.org/documentation/docs-3.0/sphinx/html/cython/importing.html

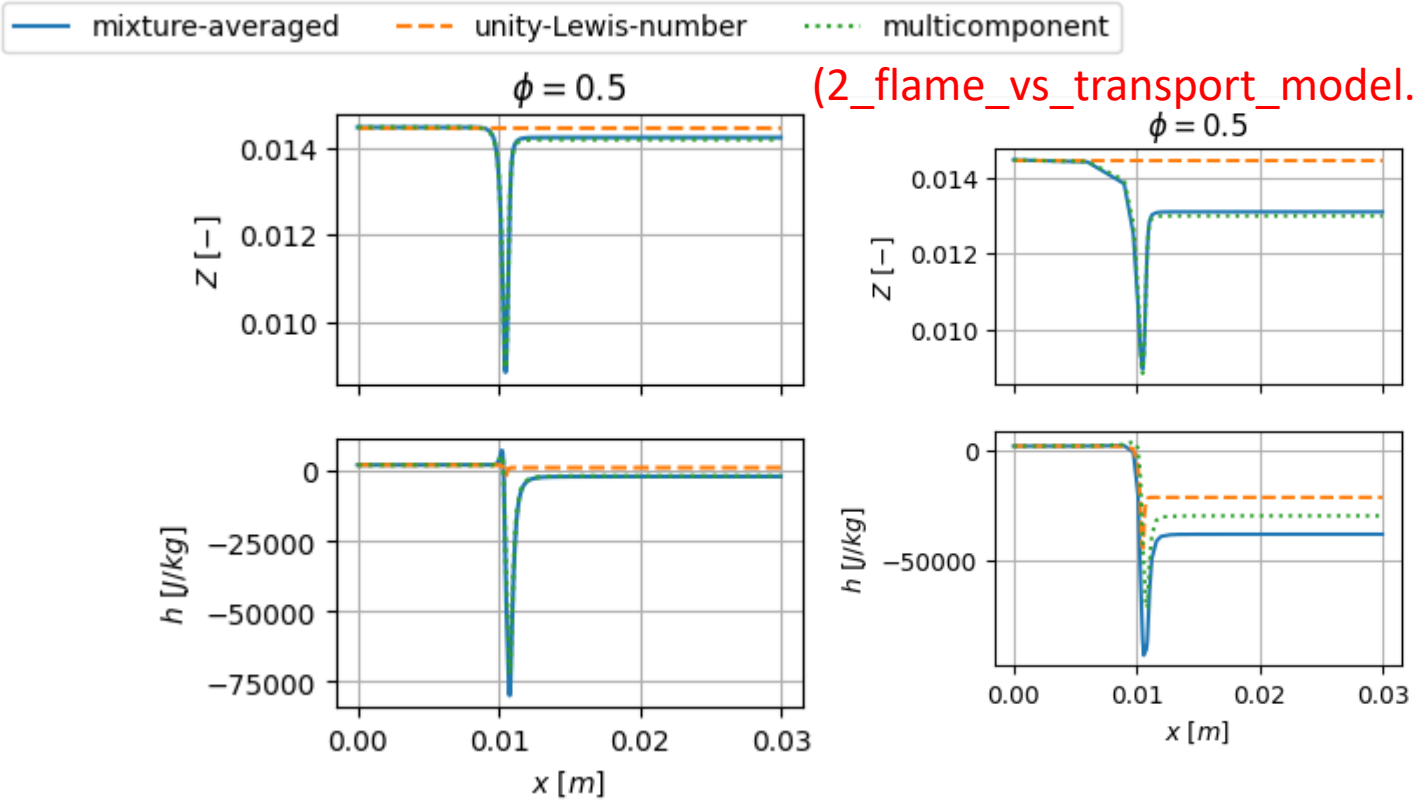
Cantera: Transport models

- (3_flame_vs_transport_model_better_numerics.py)
- Part 2) 1D Flames are the perfect setup to study preferential diffusion

$$\frac{\partial(\rho Z)}{\partial t} + \nabla \cdot (\rho \boldsymbol{u} Z) + \nabla \cdot (\mathbf{j}_Z) = 0$$

Exercise at home $Z_{in} = Z_{out}$, the same argument holds for enthalpy $h_{in} = h_{out}$

```
properties = {}
for transport_model in transport_models:

    previous_loop()

    b = bilger_from_flame(flame, gas, bilger_weights)
    Z = (b - b_ox) / (b_fuel - b_ox)
    phi_flame = phi_from_Z(Z, Z_st)
```

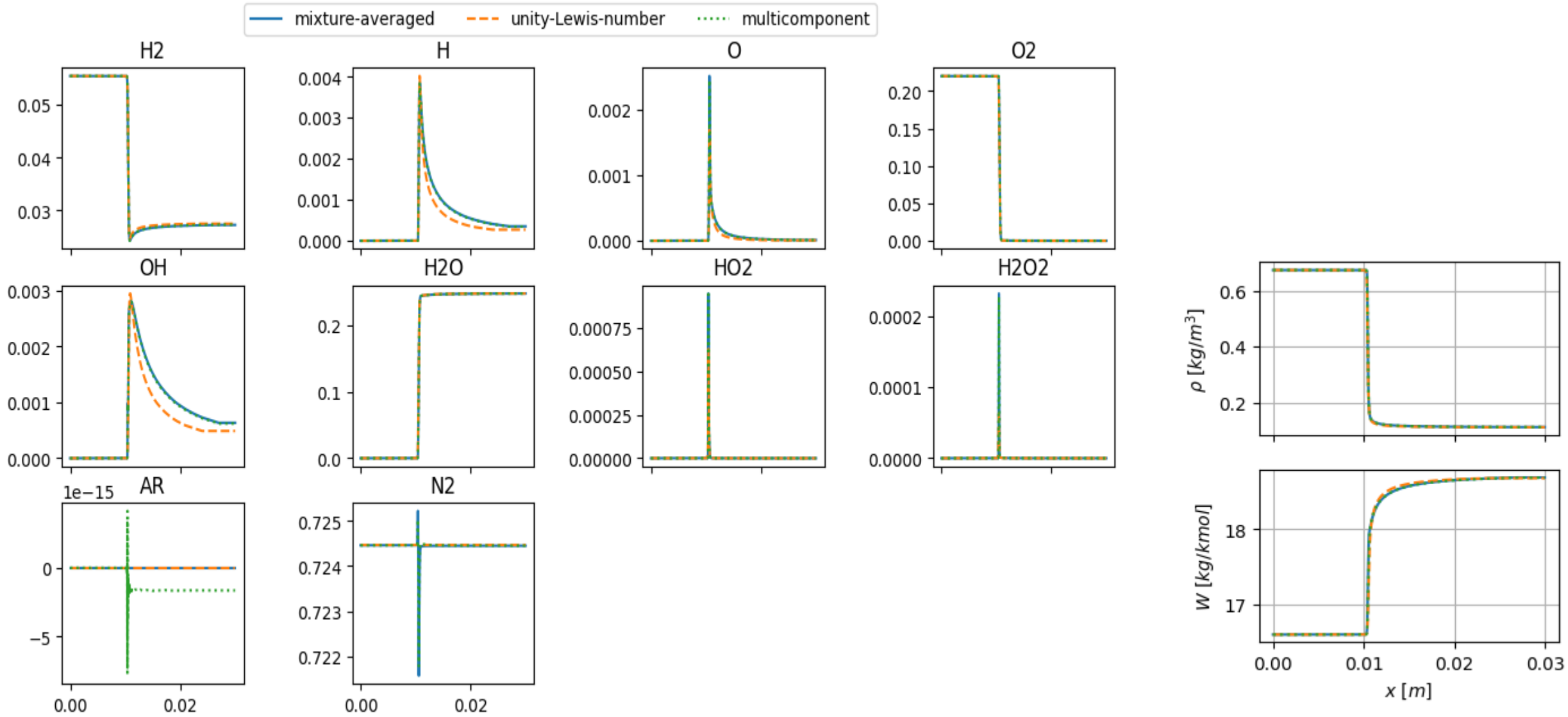


(2_flame_vs_transport_model.py)

(3_flame_vs_transport_model_better_numerics.py)

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

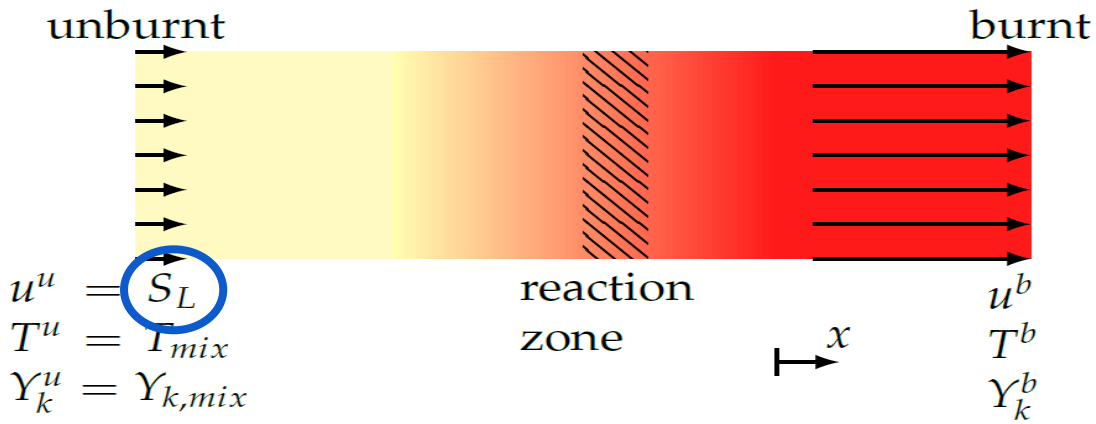
Cantera: Transport models



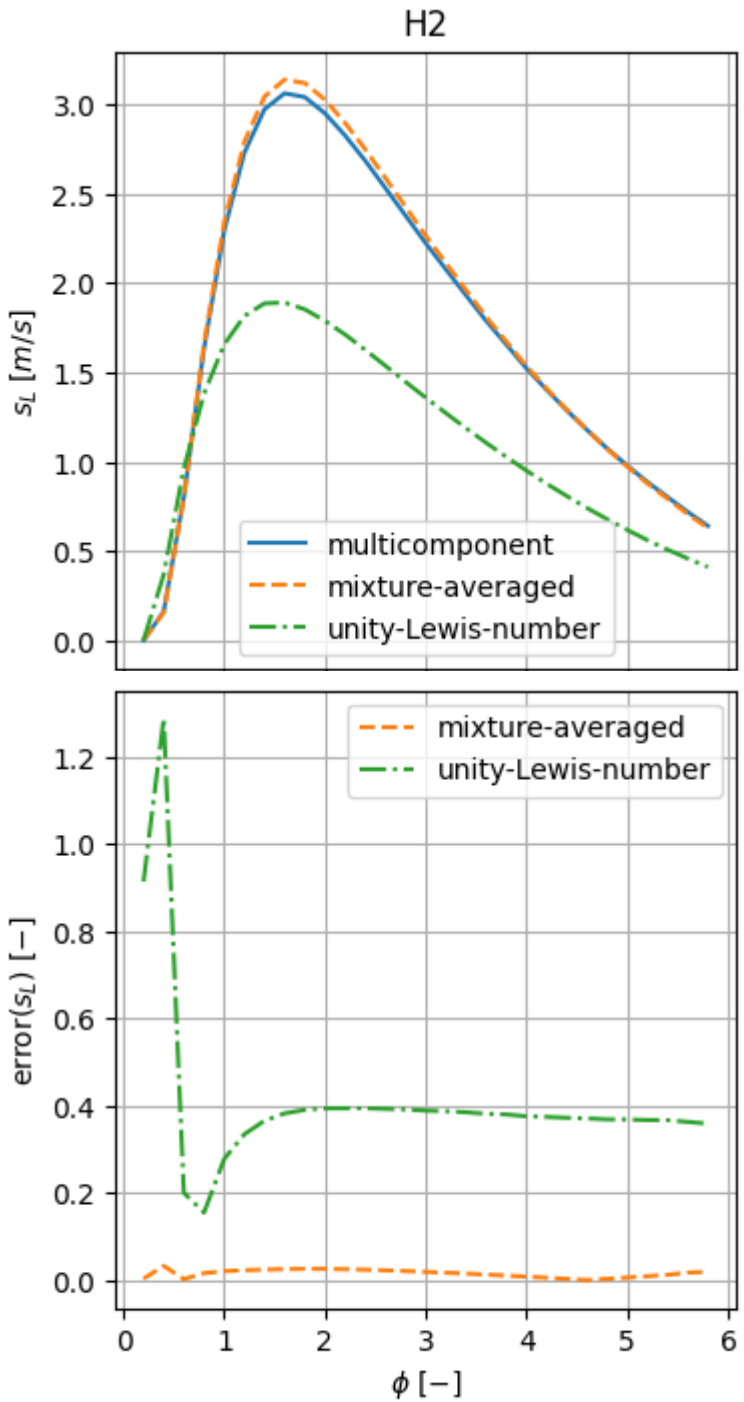
Cantera: Flame speed

(4_flame_spped.py); take some time to run

- Transport comparison:
  - Loop through transport models
  - Loop through $\phi \in [0.5, 6]$
- Where is the laminar flame speed in Cantera s_L ?



$$error(s_L) = \left| \frac{s_L^{transport} - s_L^{multicomponent}}{s_L^{multicomponent}} \right|$$

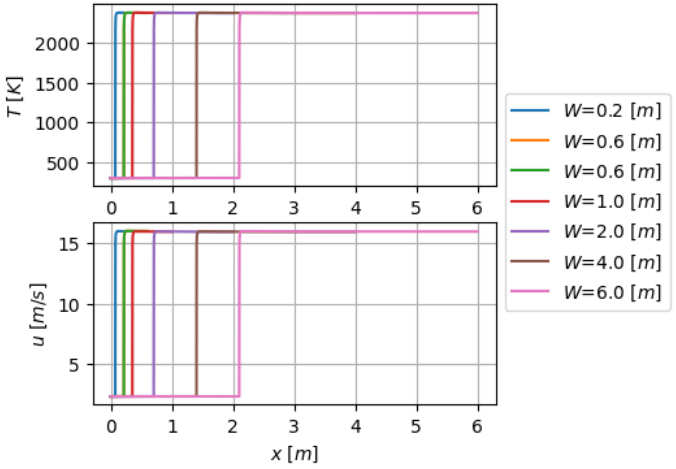


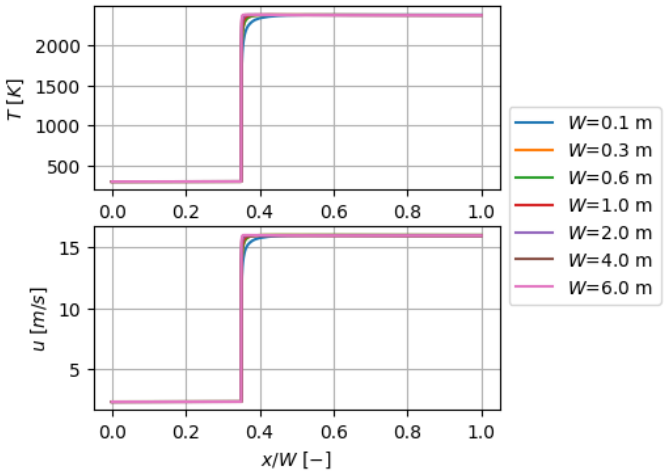
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

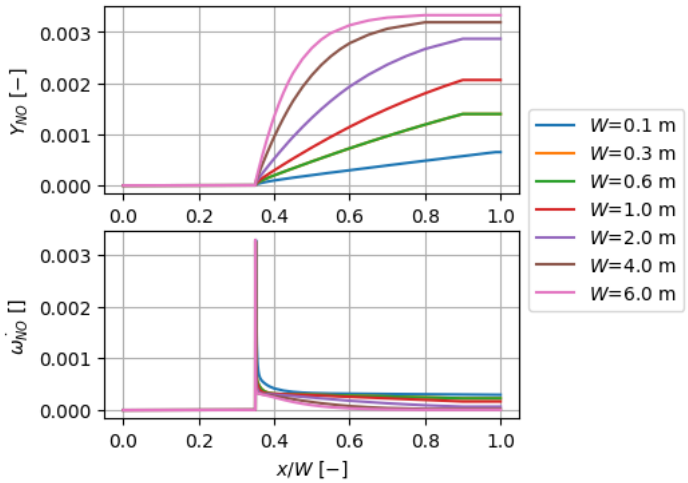
NOx production

(5_nox_problem.py)

- Nitrogen-oxide (NOx) emissions are an unavoidable side-effect of combustion in air.
- Nitric oxide builds up with residence time in the premixed flamelet domain
- GRI mechanisms are not the best to compute NOx, still they highlight the fact that even if the NOx chemical pathways are known, uncertainty remains on their contributions.
- Better chemical description can be found in mechanisms from CRECK modeling group and NUI Galway for example
  - Detailed mechanisms (CRECK) are accurate but may be difficult to handle even in 1D flamelet solvers.

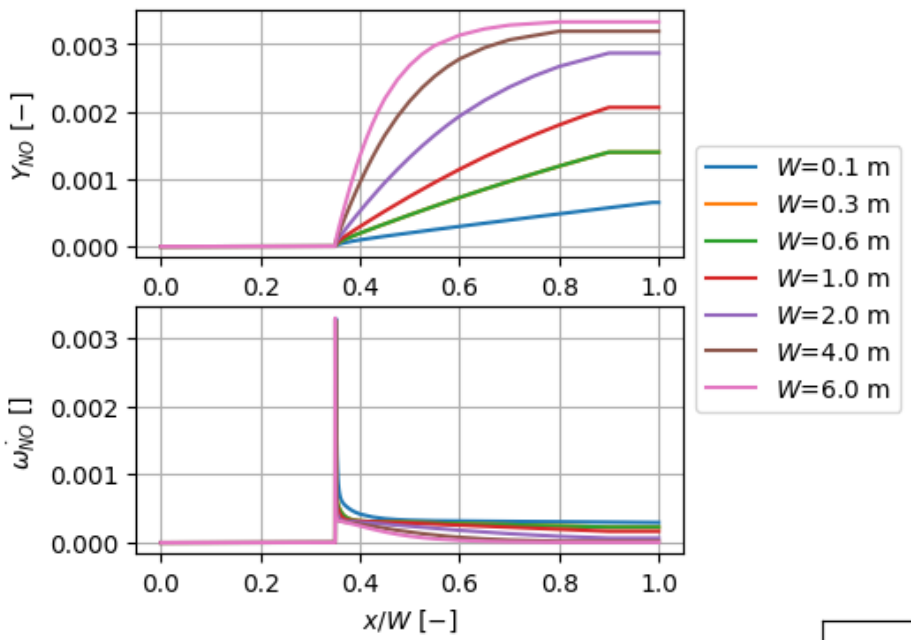




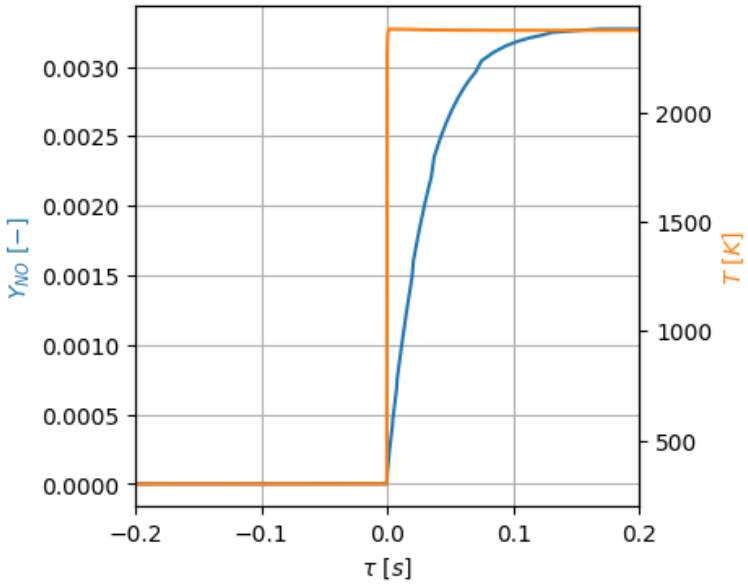


Macroscopic seems to match

NOx production



- Higher equilibrium NO content at larger domains
- NOx formation must be disrupted before the equilibrium is reached!
- NO production is concentrated at the reacting layer, where unfavorable conditions coexist:
  - High temperature
  - Available O2
- Nitric oxide builds up with residence time in the premixed flamelet domain $dt = dx/u$



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

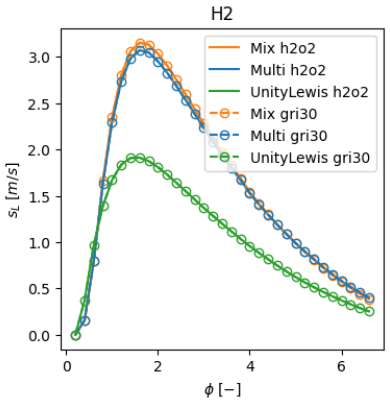
Cantera: Other exercises

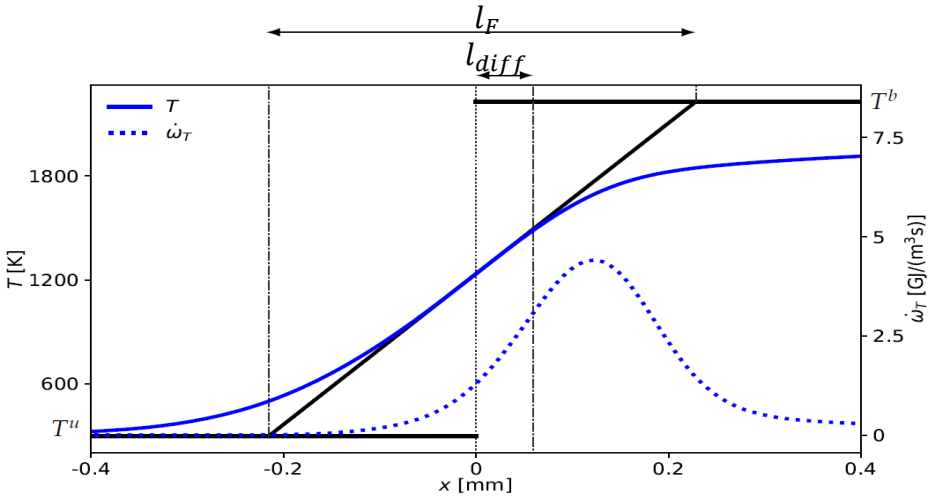
(6_flame_thickness.py)

- Calculate thermal flame thickness measures l_F (thermal flame thickness), l_{diff} (diffusive thickness)

$$l_F = \frac{T^b - T^u}{max\left(\frac{\mathrm{d}T}{\mathrm{d}x}\right)}, \quad l_{diff} = \frac{D_t^u}{s_L} = \left(\frac{\lambda}{\rho c_p}\right)^u \frac{1}{s_L}$$

  - Compare results with different transports
  - Compare results vs ϕ or Z





- Study flame speed and flame thickness vs p and T
- Study the flammability range by finding lower and upper limits of equivalence ratio of solutions
- Compare results of other fuels and compare the impact of preferential diffusion on results

(7_transport_vs_time.py)

- Compares average execution speed of flame solver using the different transports (Slow)

- Slides last year: (https://coec-project.eu/training/ercoftac-course-understanding-and-predicting-hydrogen-combustion/)
  - Try out diffusion flames
  - Play around with solver parameters to improve accuracy of solutions