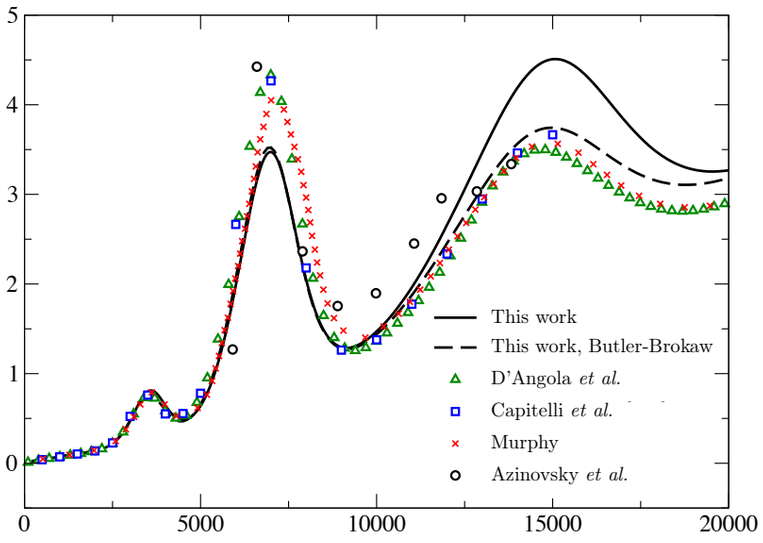
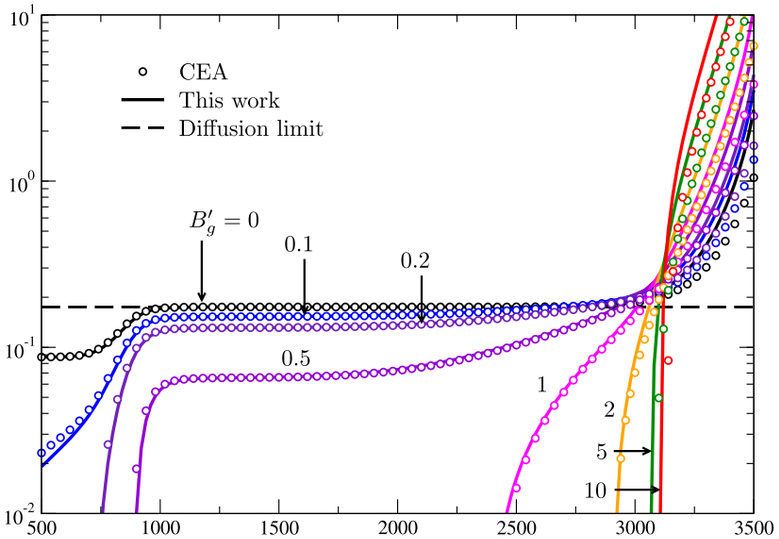
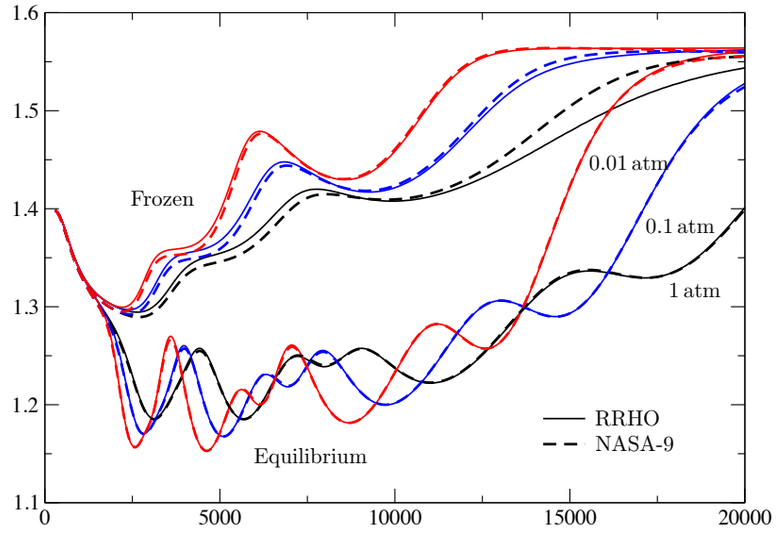
M++ Mutation

Multicomponent Thermodynamic And Transport properties for IONized gases in C++

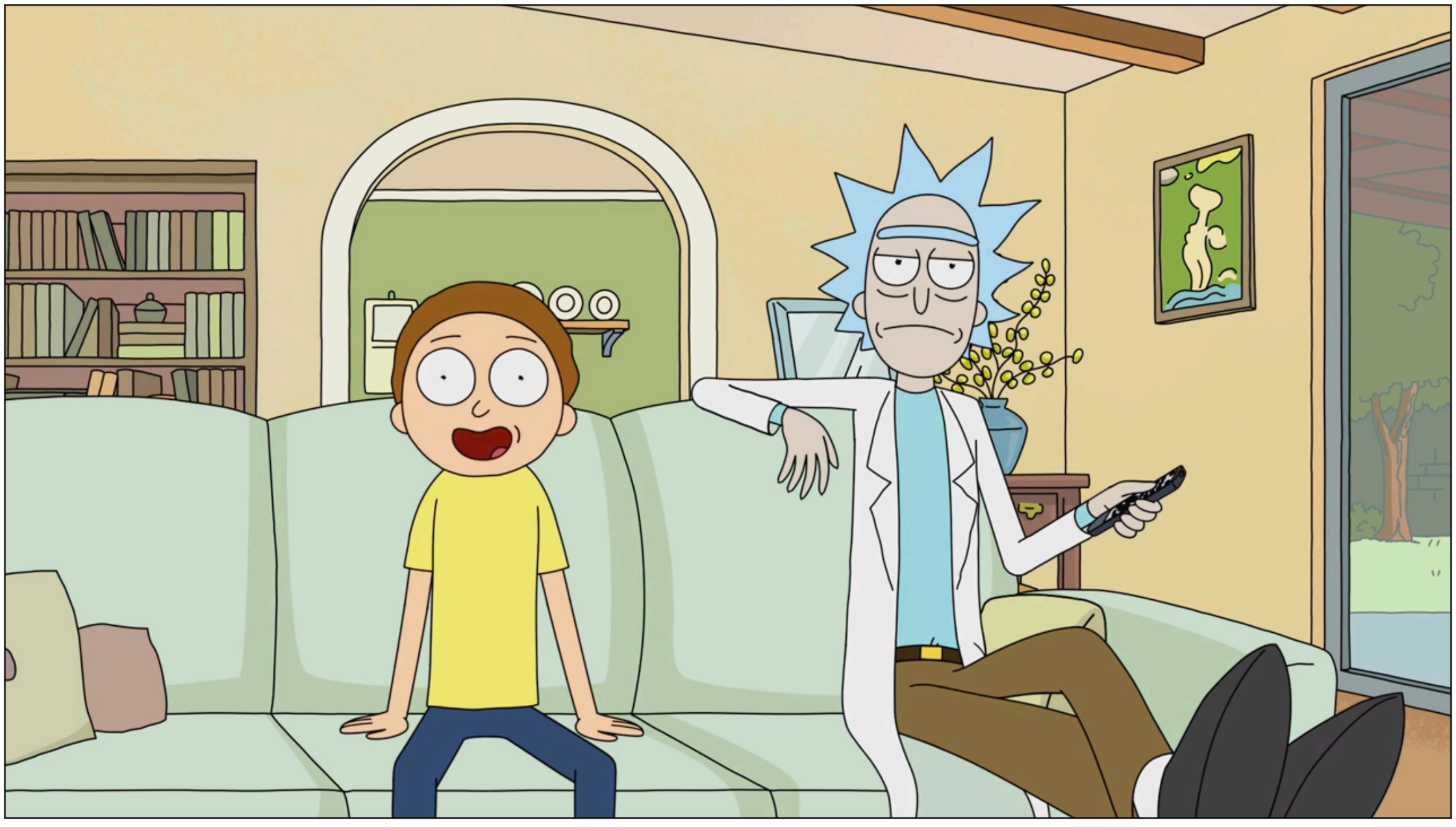




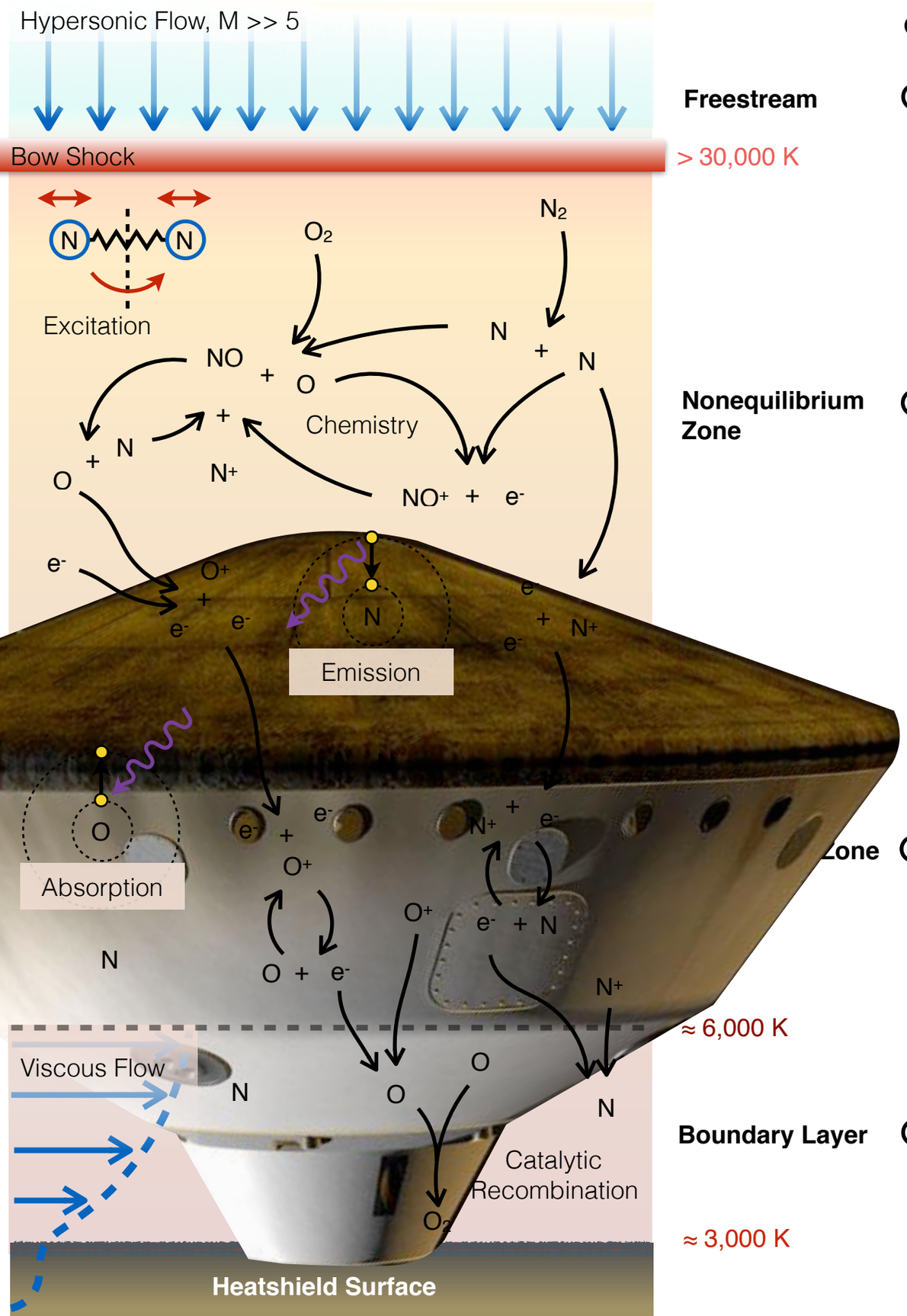


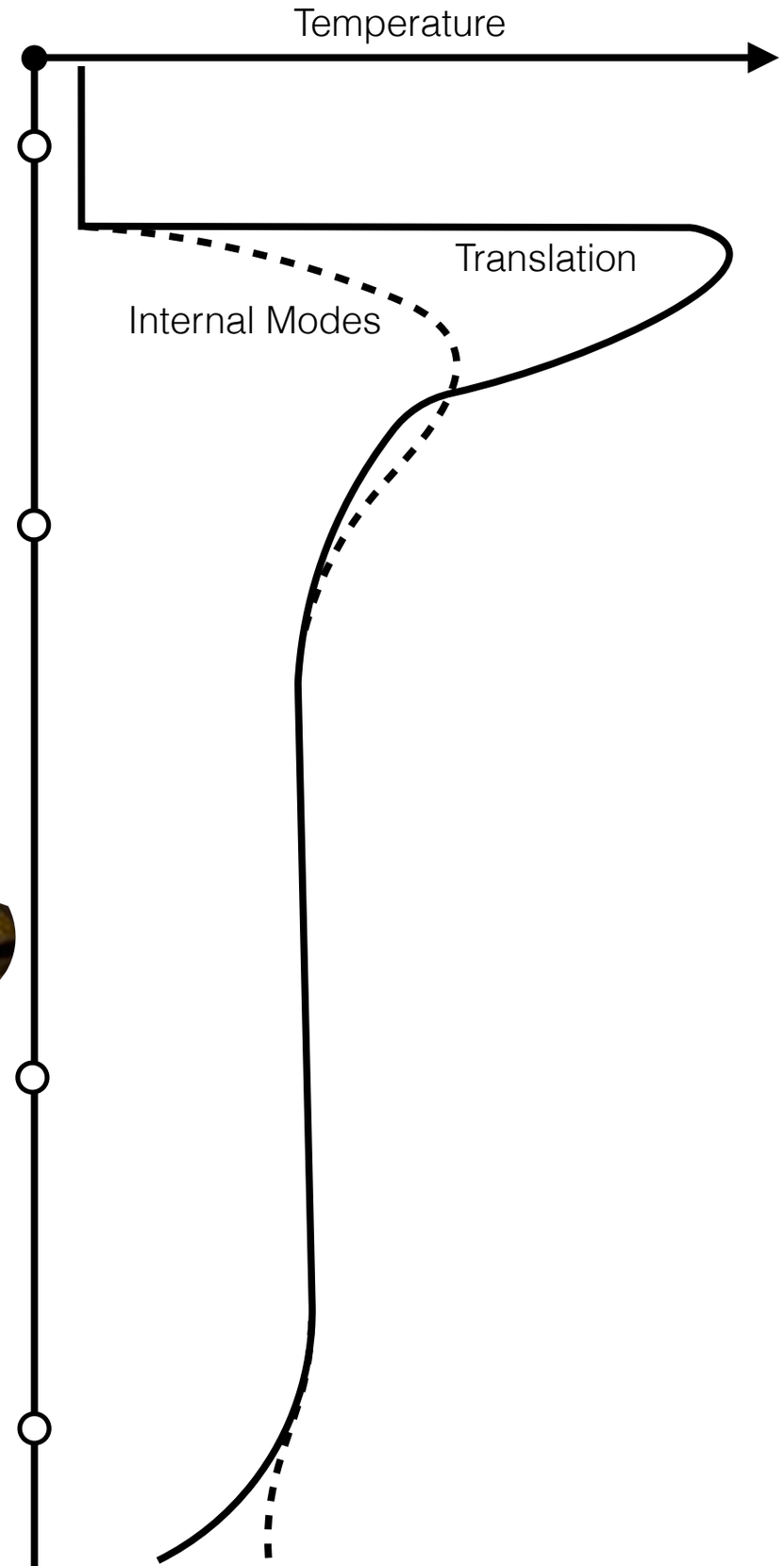
Presented by J.B. Scoggins, Georgios Bellas-Chatzigeorgis

An improvisational feel…

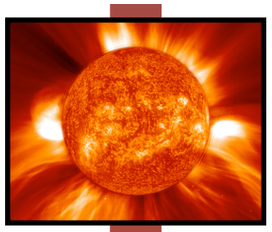


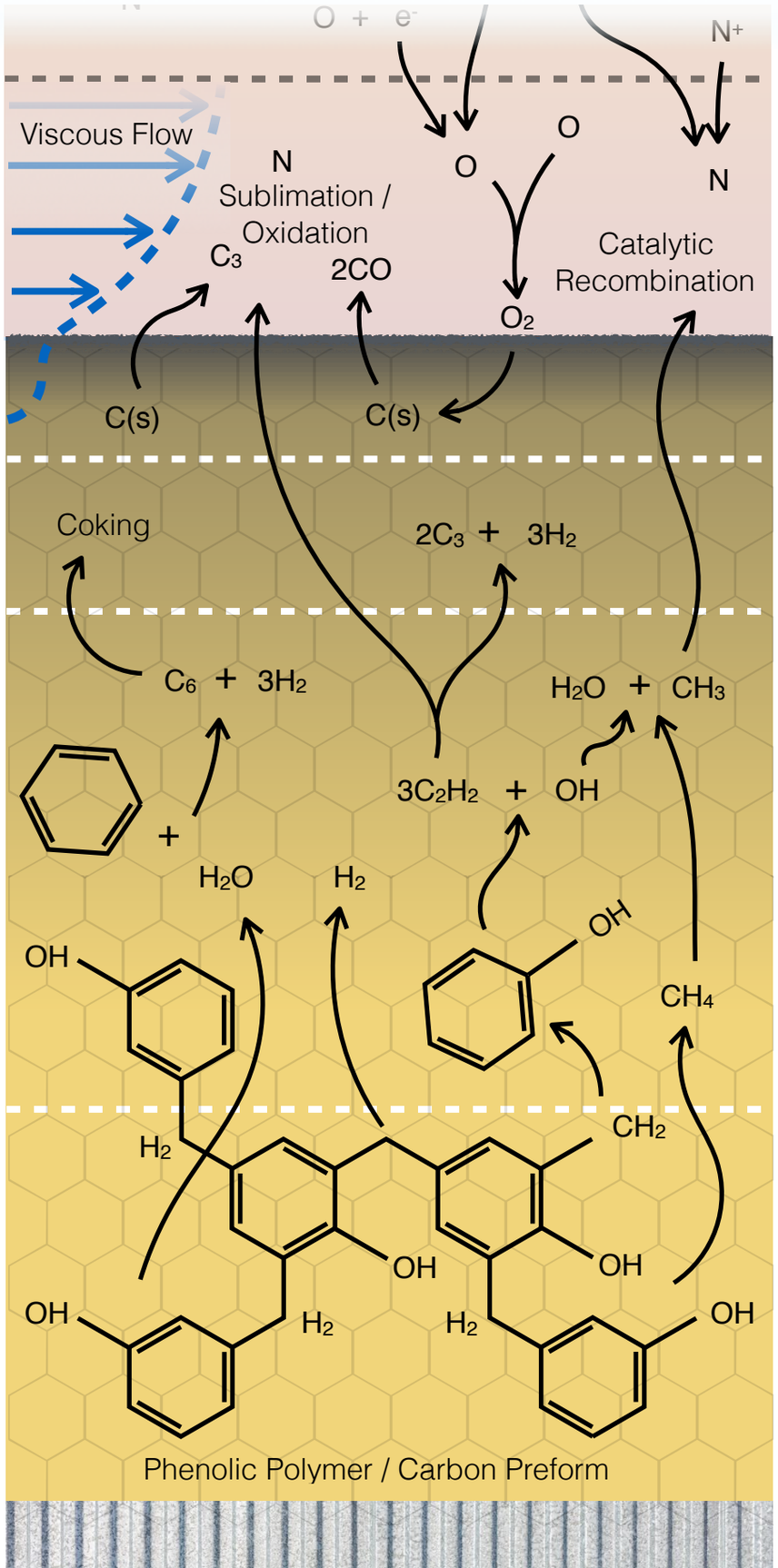
Atmospheric entry physics (Earth entries)





Thermal decomposition of the TPS





≈ 6,000 K

Boundary Layer

≈ 3,000 K

Ablation Zone

≈ 1,400 K

Coking Zone

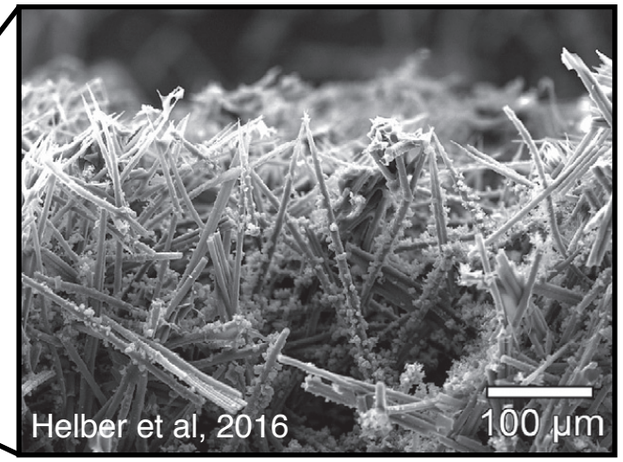
≈ 1,200 K

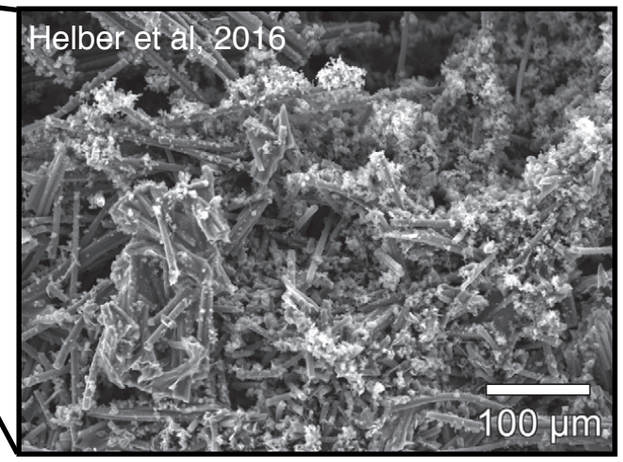
Pyrolysis Zone

≈ 400 K

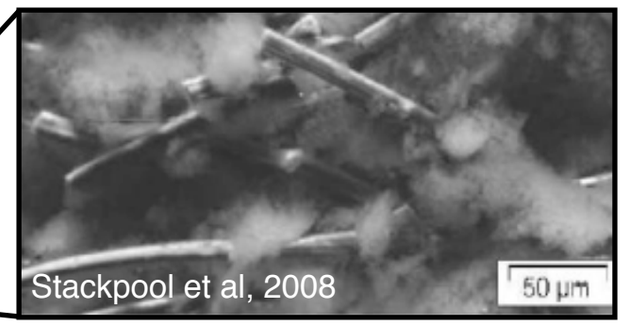
Virgin Material

Insulation





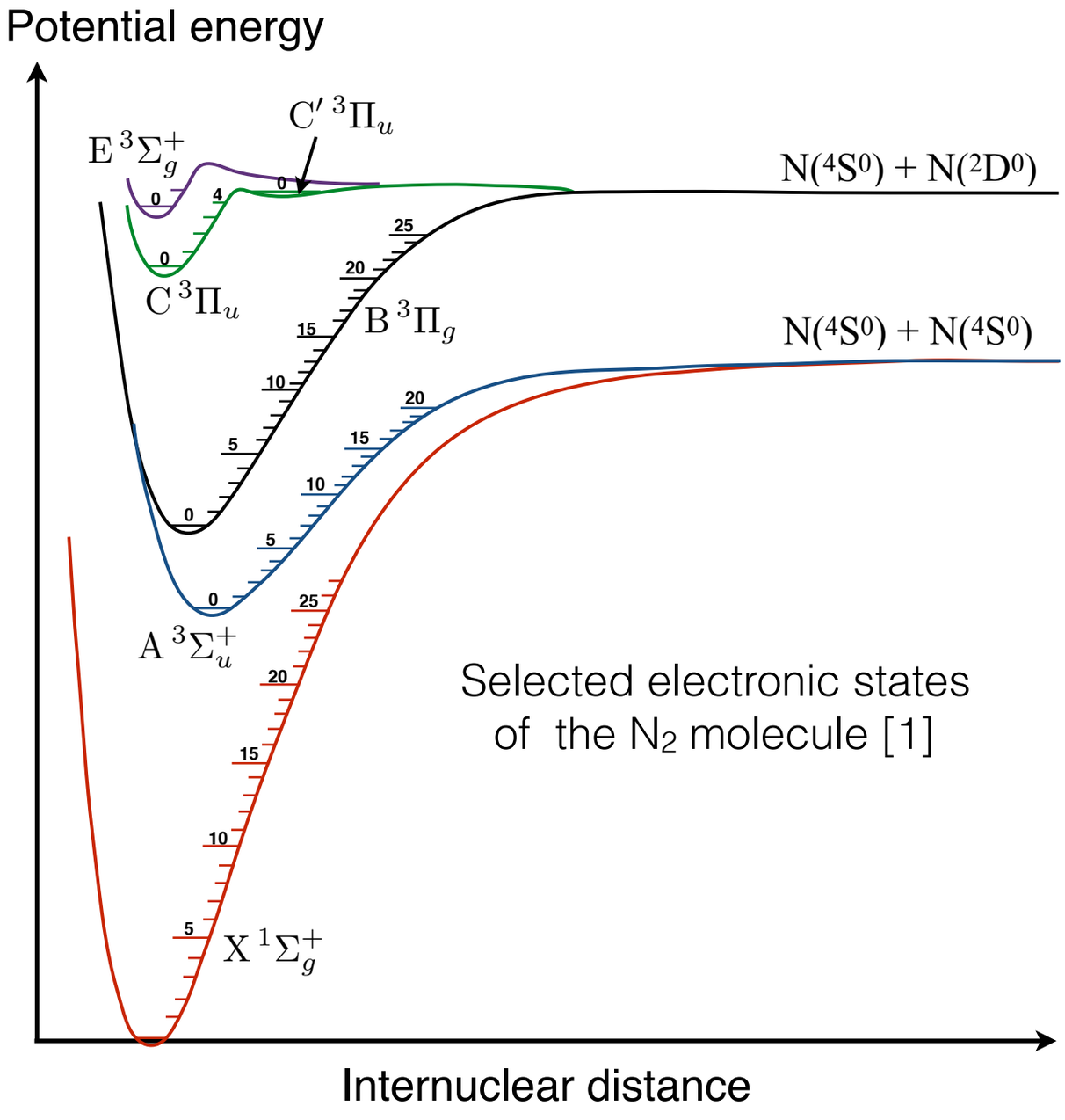
Section-cut micrographs of Asterm.



Micrographs of virgin PICA depicting carbon fibers impregnated with phenolic resin.

Energy partitioning regimes in hypersonics

Quantum mechanics dictates that atoms and molecule are permitted only **discrete energy levels**



Selected electronic states of N_2 molecule [1]

Thermal Equilibrium

$$\tau_{\mathrm{kin}} \ll \tau_{\mathrm{flow}}$$

Boltzmann Distribution

$$n_j^l = \frac{n_j}{Q_j^{\mathrm{int}}(T)} a_j^l \exp\left(-\frac{E_j^l}{k_B T}\right)$$

Multitemperature

State-specific

Energy binning

State-to-state

Nonequilibrium

$$\tau_{\mathrm{kin}} \approx \tau_{\mathrm{flow}}$$

[1] Gilmore. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 5:369–390, 1965.

Motivation

Governing equations for hypersonic flows

Detailed balance yields the following relationships for the conservation of mass, momentum, and energy

Pseudo-species mass conservation

$$\frac{\partial}{\partial t}(\rho_k) + \boldsymbol{\nabla} \cdot (\rho_k \boldsymbol{u}) + \boldsymbol{\nabla} \cdot (\rho_k \boldsymbol{V}_k) = \dot{\omega}_k + \dot{\phi}_k, \quad \forall\, k \in \mathcal{S}^*$$

Mass continuity

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

Momentum conservation

$$\frac{\partial}{\partial t}(\rho \boldsymbol{u}) + \boldsymbol{\nabla} \cdot (\rho \boldsymbol{u} \otimes \boldsymbol{u}) = -\boldsymbol{\nabla} p + \boldsymbol{\nabla} \cdot \bar{\boldsymbol{\Pi}} + nq\boldsymbol{E}' + \boldsymbol{j} \times \boldsymbol{B}$$

Free-electron energy conservation

$$\frac{\partial}{\partial t}(\rho_e e_e) + \boldsymbol{\nabla} \cdot (\rho_e e_e \boldsymbol{u}) = -p_e \boldsymbol{\nabla} \cdot \boldsymbol{u} - \boldsymbol{\nabla} \cdot \boldsymbol{q}_e + \Omega_e + \boldsymbol{j}_e \cdot \boldsymbol{E}' + \mathcal{P}_e$$

Internal energy conservation of mode m

$$\frac{\partial}{\partial t}(\rho e^m) + \boldsymbol{\nabla} \cdot (\rho e^m \boldsymbol{u}) = -\boldsymbol{\nabla} \cdot \boldsymbol{q}^m + \Omega^m + \mathcal{P}_m$$

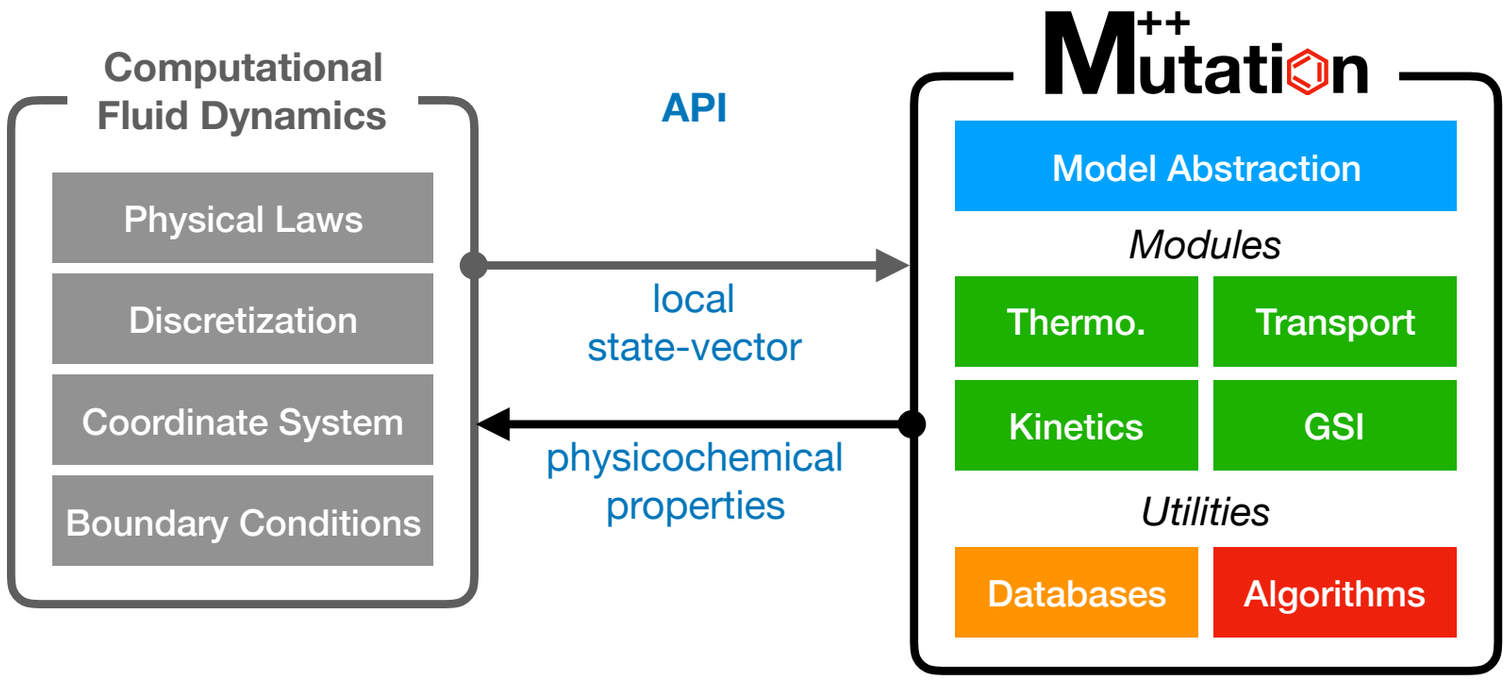
Total energy conservation

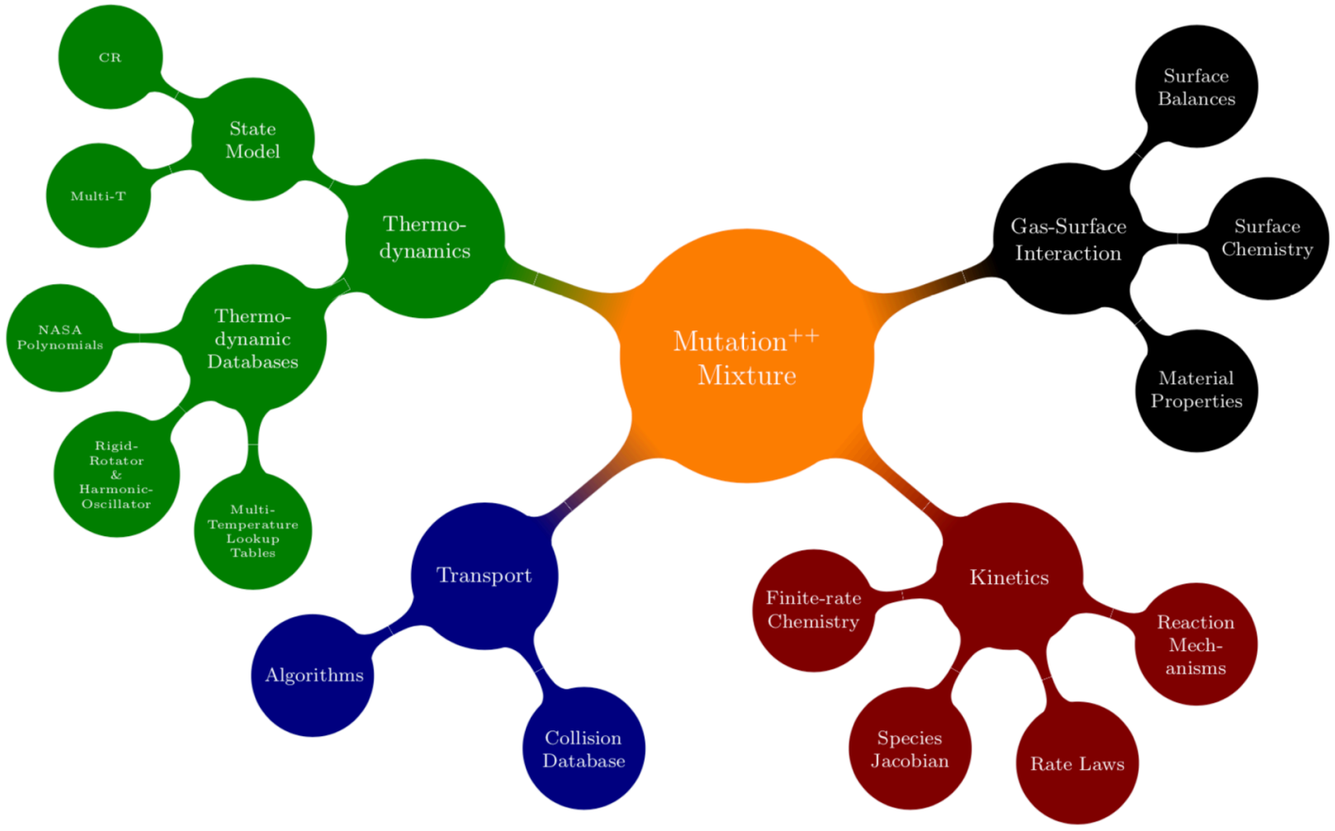
$$\frac{\partial}{\partial t}(\rho E) + \boldsymbol{\nabla} \cdot (\rho H \boldsymbol{u}) = \boldsymbol{\nabla} \cdot (\bar{\boldsymbol{\Pi}} \boldsymbol{u}) - \boldsymbol{\nabla} \cdot \boldsymbol{q} + \mathcal{P} + \boldsymbol{j} \cdot \boldsymbol{E}'$$

Requires knowledge of:

- Thermodynamic and transport properties
- Chemical kinetics
- Energy exchange mechanisms
- Radiative source terms

Development of Mutation++





Objectives:

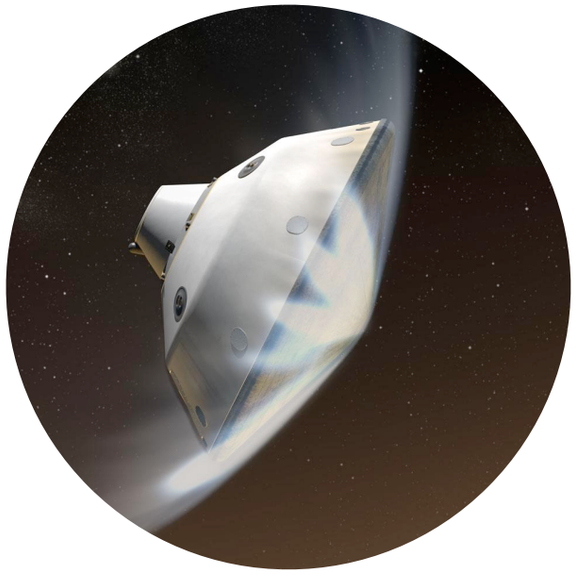
- Accurate property evaluation
- Efficient
- Extensible
- Interface to CFD
- Self documenting DBs
- Open source community

Scoggins, Magin. AIAA Paper 2014-2966. 2014.

Scoggins et al., submitted to SoftwareX.

Design allows for broad application domain

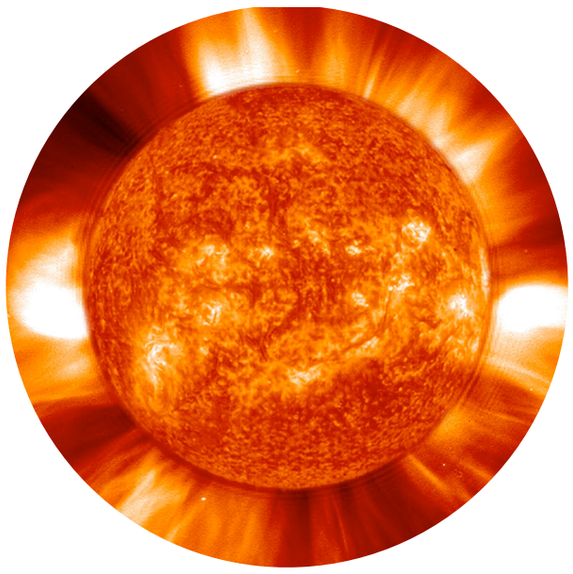
Atmospheric entry



Magnetized plasmas

Meteor phenomena

Solar physics



Biomass pyrolysis

Flexible coupling strategies: “on-the-fly” or through tables

Let's get started…

Project Homepage

https://github.com/mutationpp/Mutationpp

Exercise

1) *From the homepage, navigate to the installation instructions.*
2) *Download and install the library.*

Test the library installation

Exercise

Run the tests and verify that the library is correctly installed.

Locating and modifying data

Types and locations of database files.

Type	Data Directory
Mixture definitions	`data/mixtures`
Thermodynamic data	`data/thermo`
Transport data	`data/transport`
Reaction mechanisms	`data/mechanisms`
Energy transfer data	`data/transfer`

Search order for database files.

Local Directory	Data Directory

Exercise

1) *Locate the “air5” mixture*
2) *Locate the reaction mechanism named in the mixture file.*

Built-in Tool: `checkmix`

Loads a mixture and presents elements, species, and reactions in the mixture.

- ✓ Check if a mixture is found.
- ✓ Check exact location of the mixture file being loaded.
- ✓ Check if a mixture and corresponding support database files are formatted correctly.
- ✓ Check that no data is missing.
- ✓ Check which order the species and reactions are ordered internally.

Usage 1:

```
checkmix mixture
```

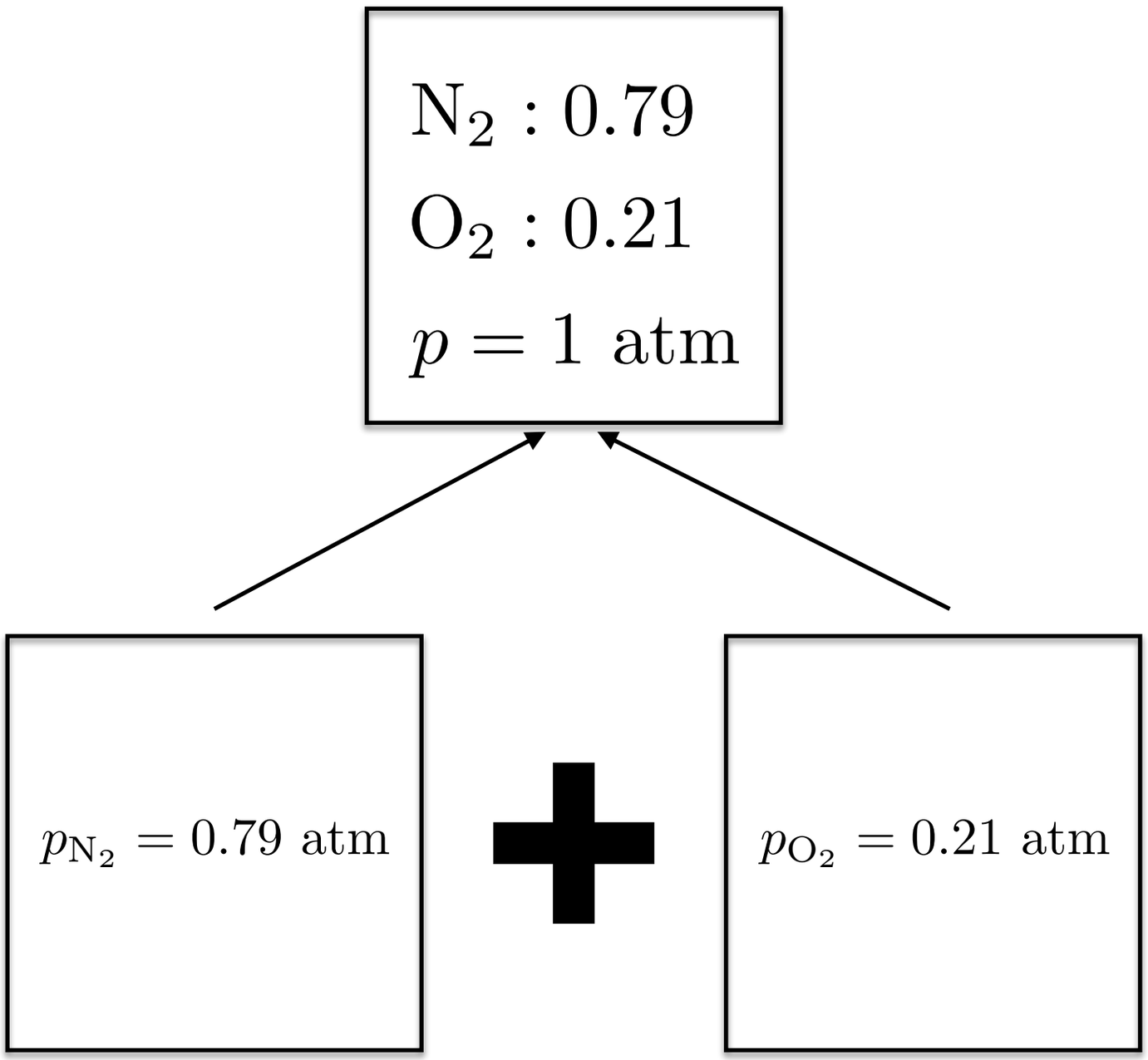
Usage 2:

```
checkmix [NASA-7 | NASA-9 | RRHO] species_list
```

Exercise

1) Run `checkmix` for "air5".
2) Try the second usage with the same species as "air5".

We are dealing with Mixtures



Perfect Gas Law

$$p = \rho R T$$

Dalton's Law

$$p = \sum_{i}^{n_s} p_i$$

Describing Chemical Mixtures

Mass Fractions

$$y_i = \frac{\rho_i}{\rho} \quad \Rightarrow \quad \frac{\text{Mass of species i}}{\text{Mass of mixture}} \qquad \sum_i^{n_s} y_i = 1$$

Mole Fractions

$$x_i = \frac{n_i}{n} \quad \Rightarrow \quad \frac{\text{Moles of species i}}{\text{Moles of mixture}} \qquad \sum_i^{n_s} x_i = 1$$

Elemental Mole Fraction

$$x_e = \frac{n_e}{n_{e,\text{tot}}} \quad \Rightarrow \quad \frac{\text{Elements e}}{\text{Total Elements}}$$

Species Properties → **Mixture Properties**

Describing Chemical Mixtures

Mass Fractions

$$y_i = \frac{\rho_i}{\rho} \quad \Rightarrow \quad \frac{\text{Mass of species i}}{\text{Mass of mixture}} \qquad \sum_i^{n_s} y_i = 1$$

Mole Fractions

$$x_i = \frac{n_i}{n} \quad \Rightarrow \quad \frac{\text{Moles of species i}}{\text{Moles of mixture}} \qquad \sum_i^{n_s} x_i = 1$$

Elemental Mole Fraction

$$x_e = \frac{n_e}{n_{e,\text{tot}}} \quad \Rightarrow \quad \frac{\text{Elements e}}{\text{Total Elements}}$$

Example:

$$x_{\text{CO}_2} = 0.8 \quad x_{\text{CN}} = 0.2$$

$$n_{e,\text{tot}} = 0.8 \cdot 1 + 0.8 \cdot 2 \quad \text{CO}_2$$
$$+ 0.2 \cdot 1 + 0.2 \cdot 1 \quad \text{CN}$$

Species Properties ⟹ **Mixture Properties**

Species Properties

NASA-9 Polynomials

- Computing properties based on polynomial fits

Enthalpy:

$$\frac{h_i}{R_i T} = -a_{0i} T^{-2} + a_{1i} \frac{\ln T}{T} + a_{2i} + \frac{a_{3i}}{2} T + \frac{a_{4i}}{3} T^2 + \frac{a_{5i}}{4} T^3 + \frac{a_{6i}}{5} T^4 + \frac{a_{7i}}{T}$$

Entropy:

$$\frac{s_i}{R_i} = -\frac{a_{0i}}{2} T^{-2} - a_{1i} T^{-1} + a_{2i} \ln T + a_{3i} T + \frac{a_{4i}}{2} T^2 + \frac{a_{5i}}{3} T^3 + \frac{a_{6i}}{4} T^4 + a_{8i}$$

Gibbs Free Energy (or any other thermodynamic property):

$$\frac{g_i}{R_i T} = \frac{s_i}{R_i T} - \frac{h_i}{T}$$

Rigid Rotator-Harmonic Oscillator

- Computing properties with analytical formulas (statistical thermodynamics)

e.g.: $e_i^{\text{trans}} = \frac{3}{2} R_i T$

Species to Mixture Properties

Energy and Enthalpy

$$e = \sum_{i=1}^{\mathrm{n}_s} y_i e_i \qquad \text{and} \qquad h = \sum_{i=1}^{\mathrm{n}_s} y_i h_i$$

Entropy

$$s = \sum_{i=1}^{\mathrm{n}_s} y_i s_i - \frac{nk_B}{\rho} \sum_{i}^{\mathrm{n}_s} x_i \ln\ x_i$$

Entropy of Mixing

Here all properties are in **kg**. When the properties are in **mole,** one should multiply with mole fractions!

Constrained multiphase equilibrium

A mixture is at equilibrium when the system Gibbs free energy is minimized, under mass balance constraints.

$$\tilde{G} \equiv \frac{G}{R_u T} = \sum_{m \in \mathcal{P}} \sum_{j \in \mathcal{S}_m} N_j \left(\tilde{g}_j + \ln N_j - \ln \bar{N}_m \right)$$

$$\bar{N}_m = \sum_{j \in \mathcal{S}_m} N_j, \quad \forall\, m \in \mathcal{P}$$

$$\sum_{j \in \mathcal{S}} B^{\mathrm{e}}_{ji} N_j = c^{\mathrm{e}}_i \quad \forall\, i \in \mathcal{E}$$

yields small non linear system of equations to be solved

Can use the Lagrange multiplier technique to solve constrained minimization problem.

$$L = \tilde{G} - \sum_{i \in \mathcal{C}} \lambda_i \left(\sum_{j \in \mathcal{S}} B_{ji} N_j - c_i \right)$$

$$= \sum_{j \in \mathcal{S}} N_j \left(\tilde{g}_j + \ln N_j - \ln \sum_{k \in \mathcal{S}_{p_j}} N_k \right) - \sum_{i \in \mathcal{C}} \lambda_i \left(\sum_{j \in \mathcal{S}} B_{ji} N_j - c_i \right)$$

$$\frac{\partial L}{\partial N_j} = \frac{\partial \tilde{G}}{\partial N_j} - \sum_{i \in \mathcal{C}} \lambda_i B_{ji}$$

$$= \tilde{g}_j + \ln N_j - \ln \sum_{k \in \mathcal{S}_{p_j}} N_k - \sum_{i \in \mathcal{C}} \lambda_i B_{ji} = 0, \quad \forall\, j \in \mathcal{S}.$$

necessary conditions for minimization

Species	C	O	$\bar{N}_{\mathrm{mix}}$	Gas	C(gr)
	B columns			**P** columns	
C	1	0	1	1	0
CO	1	1	1	1	0
CO_2	1	2	1	1	0
O_2	0	2	1	1	0
C(gr)	1	0	1	0	1

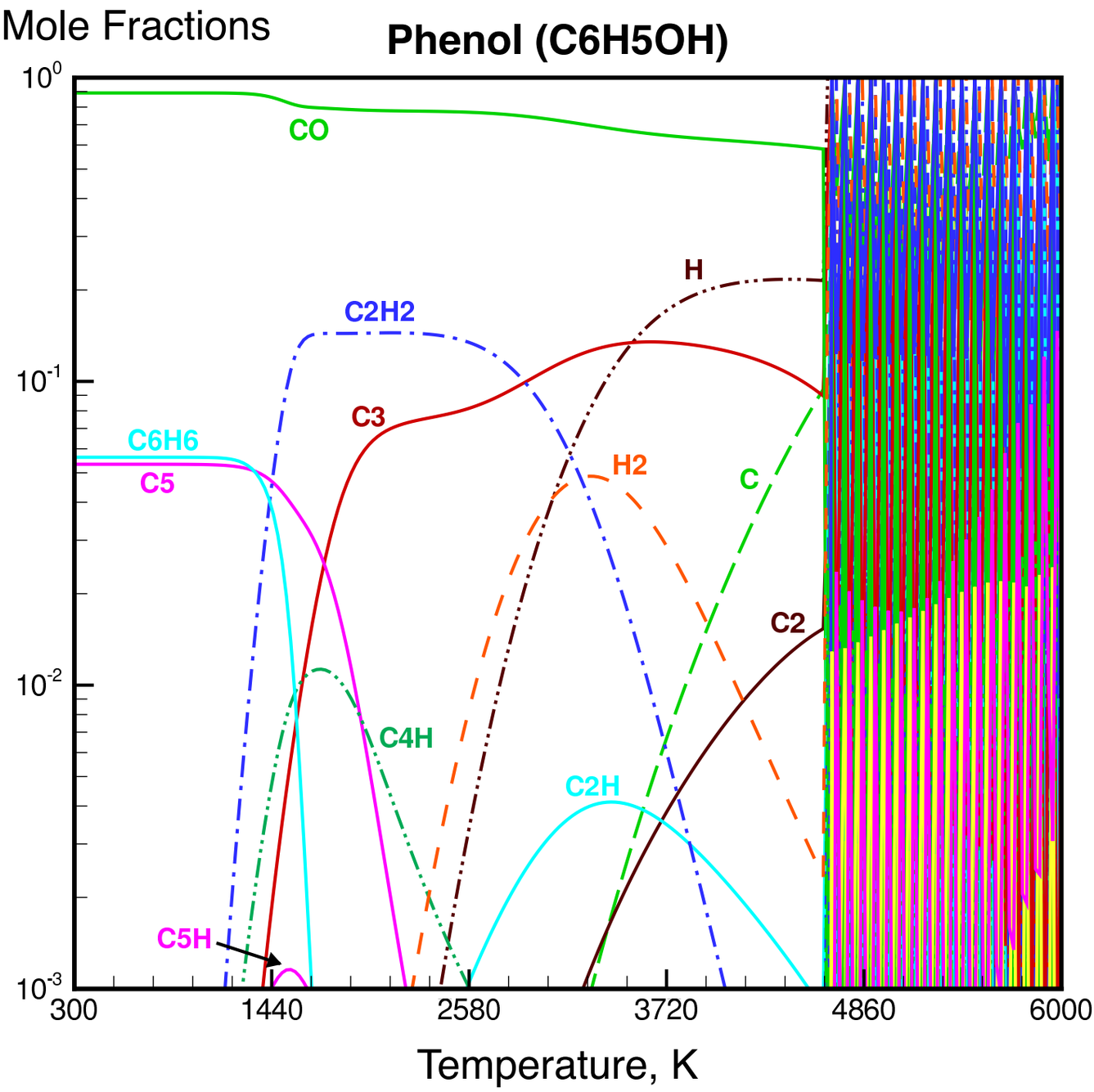
general linear constriants

Multiphase Gibbs function continuation

Minimization of Gibbs energy under constraints

Solution of nonlinear system for phase moles and constraint potentials

- Newton iterations may fail to converge
- Pope [1] introduced Gibbs function continuation (GFC) method for one phase
- Have extended Popes formulation to multiple phases (MPGFC) in [2]
- Proof of guaranteed convergence when constraints are valid
- Showed that Pope convergence proof is a special case of the general proof

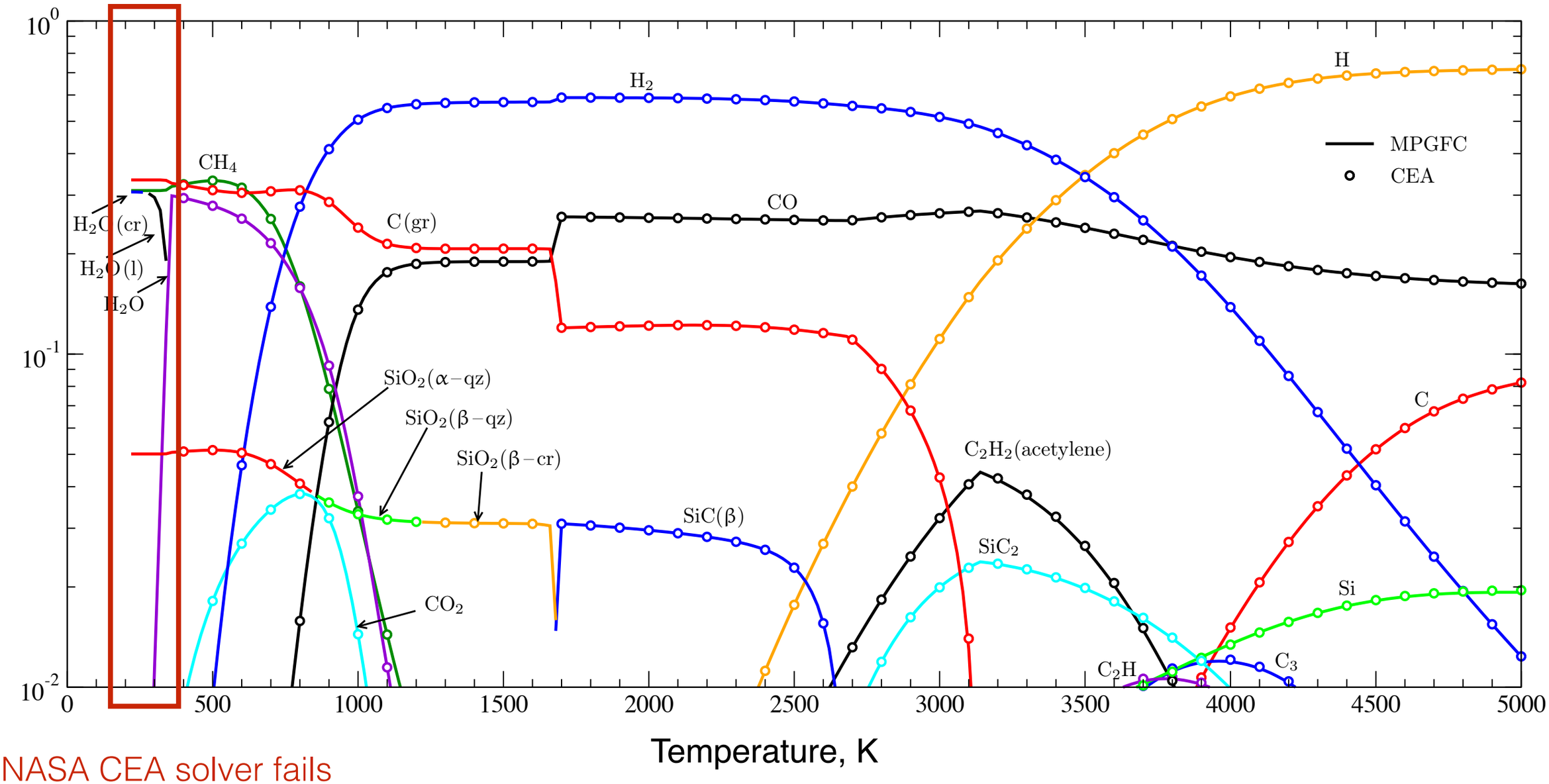


[1] Pope. *Combustion and Flame*, 139(3):222–226, 2004.

[2] Scoggins, Magin. *Combustion and Flame, 162(12):4514–4522, 2015.*

Multiphase Gibbs function continuation

Equilibrium mixture of 90% phenol (C_6H_5OH) with 10% silicon



NASA CEA solver fails to converge at low temperatures

[1] Scoggins, Magin. *Combustion and Flame, 162(12):4514–4522, 2015.*

Built-in Tool: `mppequil`

Generates a table of equilibrium properties for a given mixture and conditions.

Usage:

```
mppequil [options] mixture
```

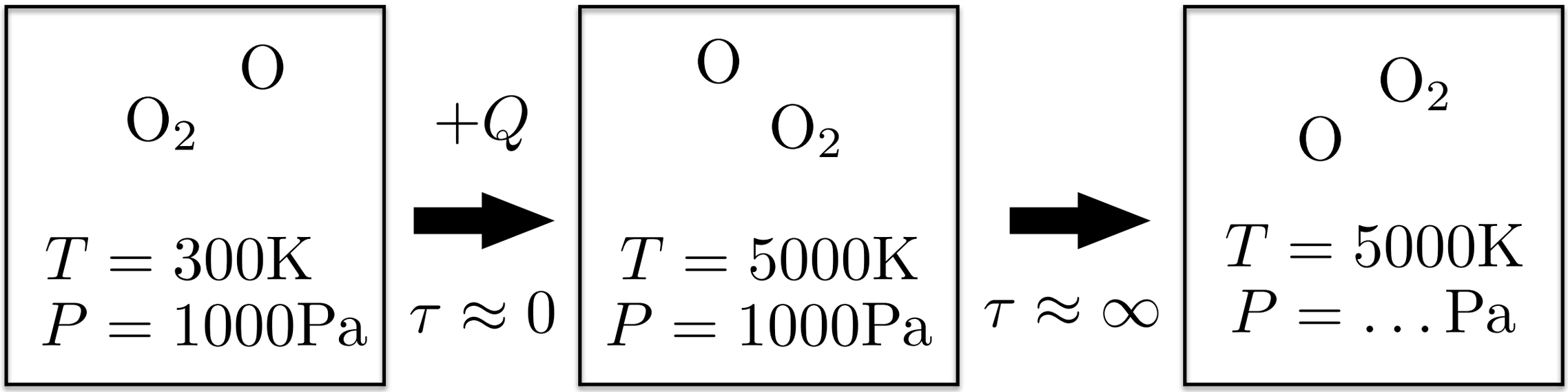
Help:

```
mppequil -h
```

Exercise

1) *Check out the help for mppequil.*
2) *Compute the equilibrium mole fractions for 5-species air from 500K to 8000K at 1atm.*

Reaction occur at finite rate till equilibrium



$\rho_{\text{O}} = 0\ \text{kg/m}^3$

$\rho_{\text{O}_2} = \ldots\ \text{kg/m}^3$

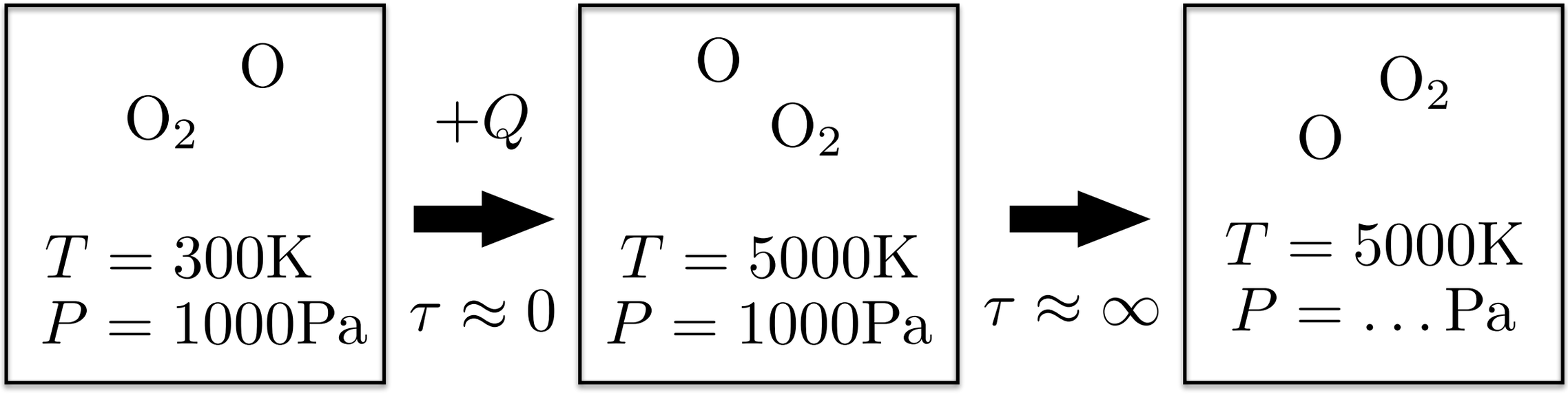
$\rho_{\text{O}} = 0\ \text{kg/m}^3$

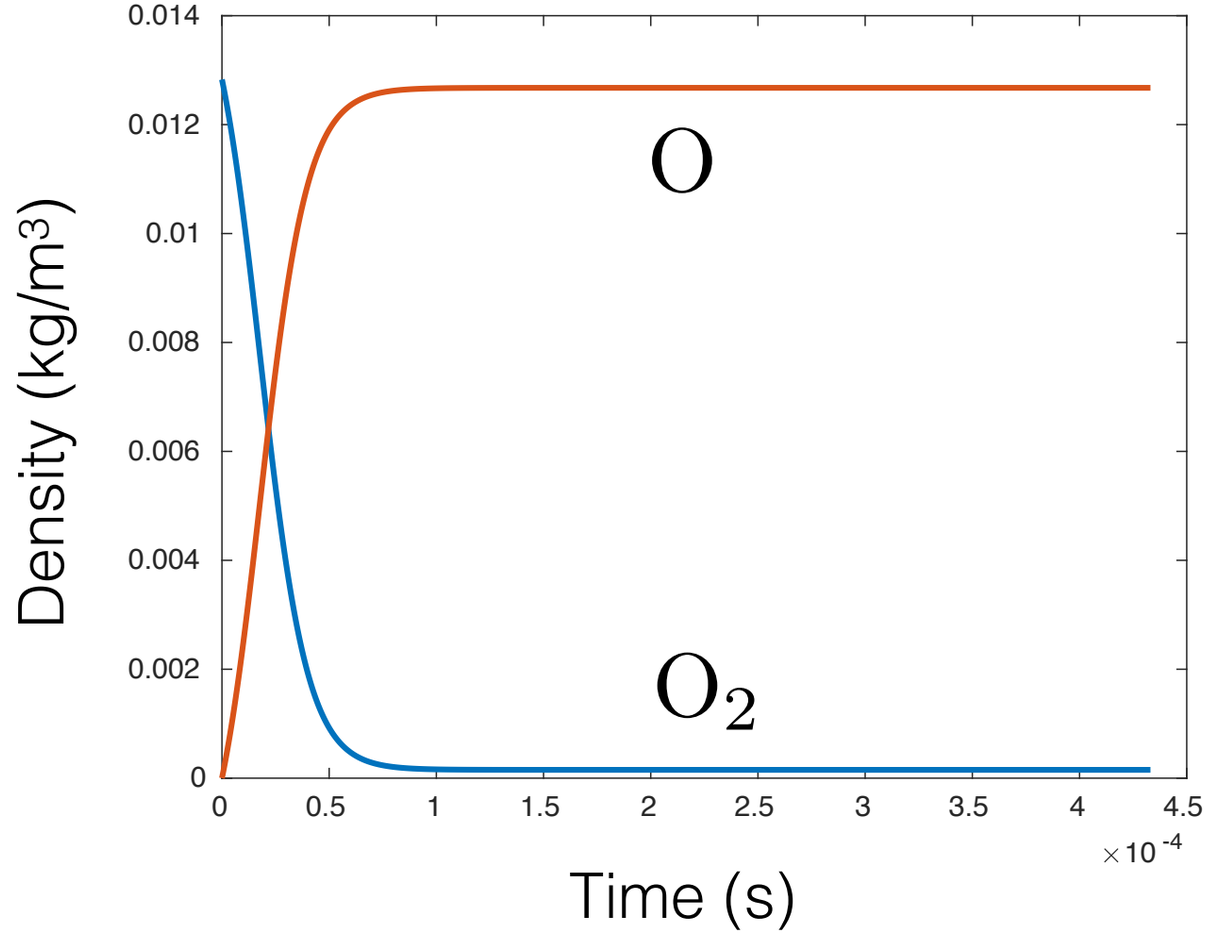
$\rho_{\text{O}_2} = \ldots\ \text{kg/m}^3$

$\rho_{\text{O}} = \ldots\ \text{kg/m}^3$

$\rho_{\text{O}_2} = 0\ \text{kg/m}^3$

Reaction occur at finite rate till equilibrium

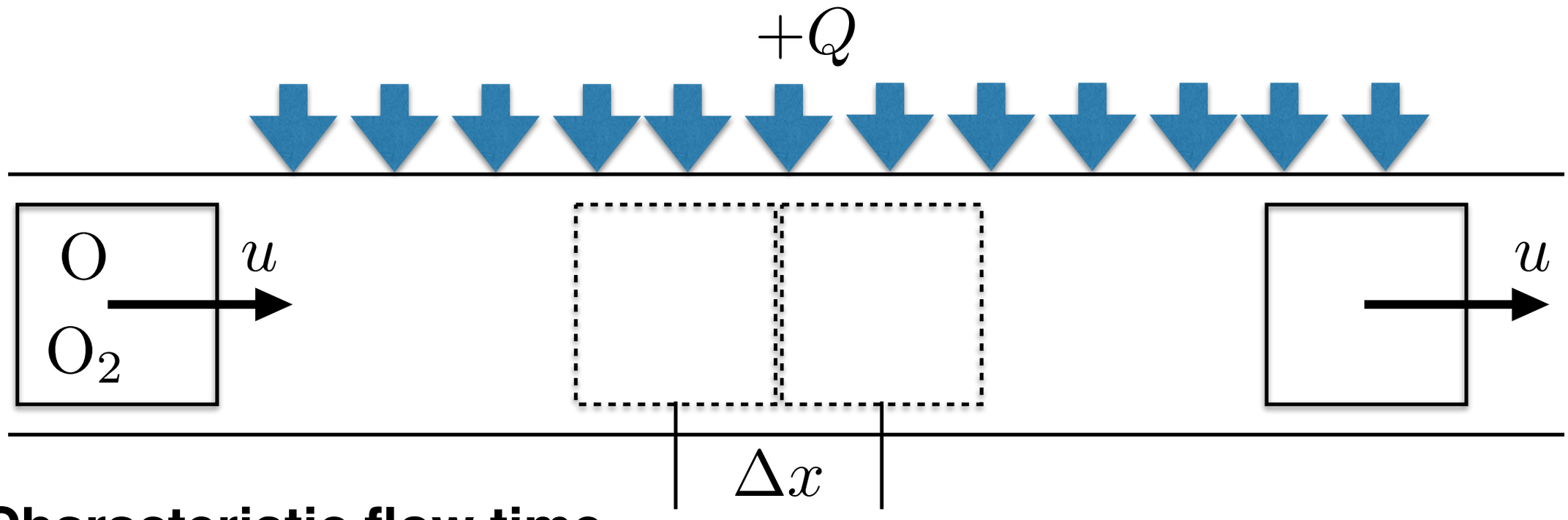




Finite Rate Chemistry

$$\mathrm{O} + \mathrm{O} \longrightarrow \mathrm{O}_2$$

When is finite-rate chemistry necessary?



Characteristic flow time

$$\tau_{\text{flow}} = \frac{\Delta x}{u}$$

Damköhler Number:

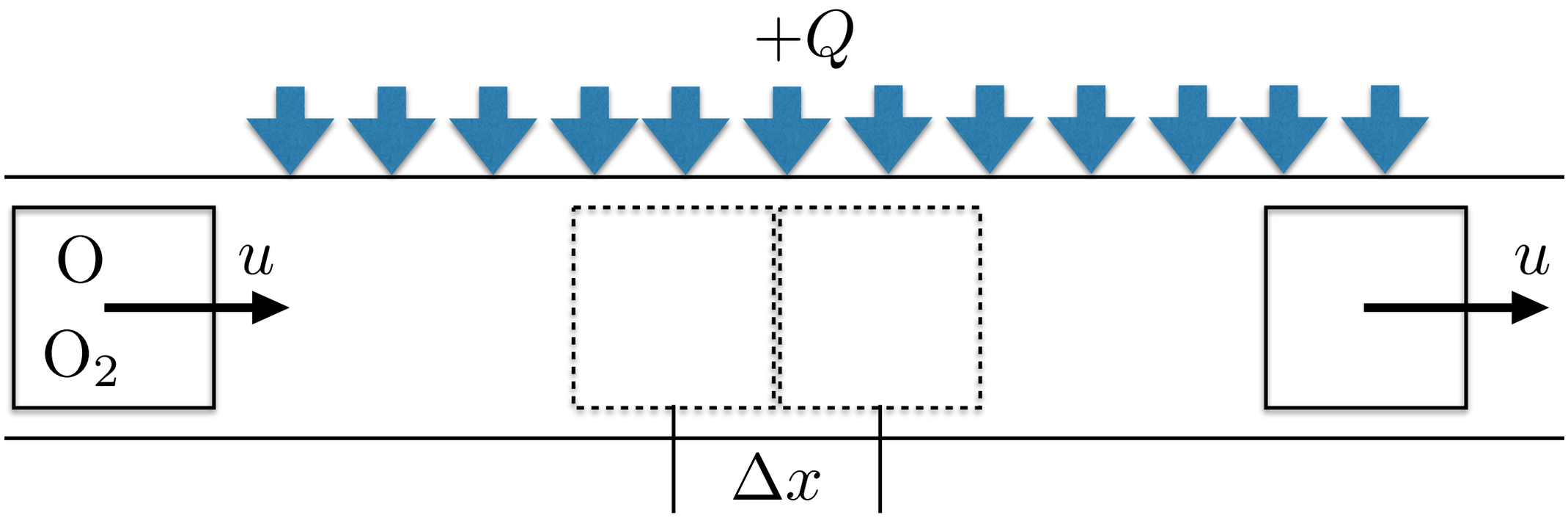
$$\text{Da} = \frac{\tau_{\text{flow}}}{\tau_{\text{chem}}}$$

$\text{Da} \ll 1$ Frozen

$\text{Da} \gg 1$ Equilibrium

Real flows are neither frozen nor in equilibrium

When is finite-rate chemistry necessary?



Frozen/Equilibrium:

$$\frac{\partial \rho}{\partial t} + \nabla \left(\rho u \right) = 0$$

Finite Rate Chemistry

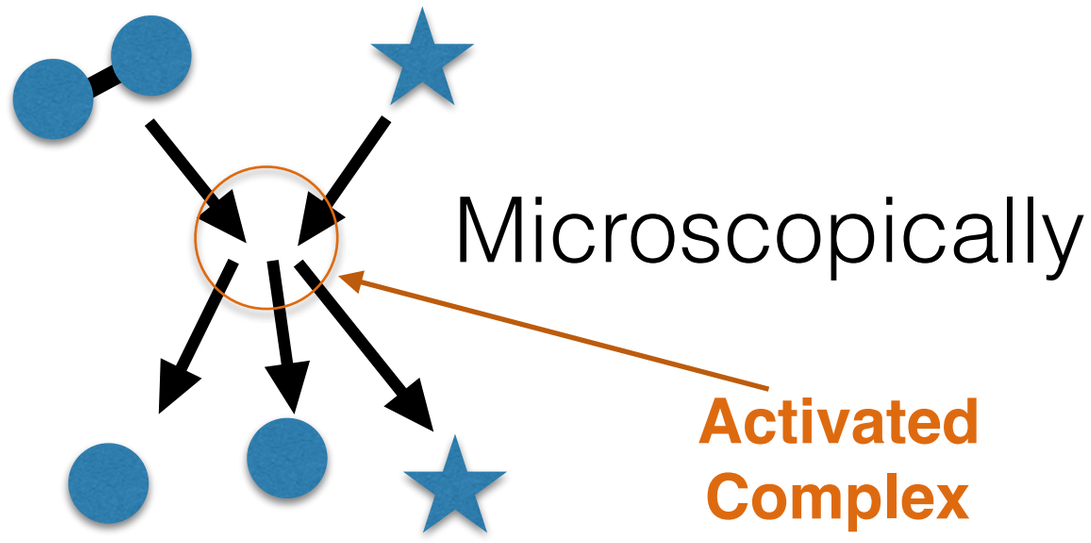
$$\frac{\partial \rho_i}{\partial t} + \nabla \left(\rho_i u + j_{\text{diff}} \right) = \dot{\omega}_i$$

Chemical Production: $\dot{\omega}_i \sim \frac{d[O]}{dt}$

Chemical Reactions and their rate

Oxygen Dissociation

$$\mathrm{O_2} + \mathrm{M} \longrightarrow \mathrm{O} + \mathrm{O} + \mathrm{M}$$



Law of Mass Action

$$r = k_f[\mathrm{O_2}][\mathrm{M}] \qquad \text{Reaction Rate}$$

Chemical Production Rate of Species

$$\frac{d\mathrm{O_2}}{dt} = -k_f[\mathrm{O_2}][\mathrm{M}] \qquad \frac{d\mathrm{O}}{dt} = 2k_f[\mathrm{O_2}][\mathrm{M}]$$

Stoichiometry

Generic Chemical Reaction

$$\mathrm{aA} + \mathrm{bB} \ldots \longrightarrow \mathrm{cC} + \mathrm{dD} \ldots$$

Reactants

$$\frac{d[\mathrm{A}]}{dt} = -\mathrm{a}k_f[\mathrm{A}]^{\mathrm{a}}[\mathrm{B}]^{\mathrm{b}}$$

$$\frac{d[\mathrm{B}]}{dt} = -\mathrm{b}k_f[\mathrm{A}]^{\mathrm{a}}[\mathrm{B}]^{\mathrm{b}}$$

Products

$$\frac{d[\mathrm{D}]}{dt} = \mathrm{d}k_f[\mathrm{A}]^{\mathrm{a}}[\mathrm{B}]^{\mathrm{b}}$$

$$\frac{d[\mathrm{C}]}{dt} = \mathrm{c}k_f[\mathrm{A}]^{\mathrm{a}}[\mathrm{B}]^{\mathrm{b}}$$

To go from **moles** to **kg** we multiply by the **molar mass**

Reaction Rate Coefficient

Activation Energy

Arrhenius Law: $k_f = C \exp \frac{-E_a}{RT}$

Pre-exponential

Kinetic Energy of Atoms

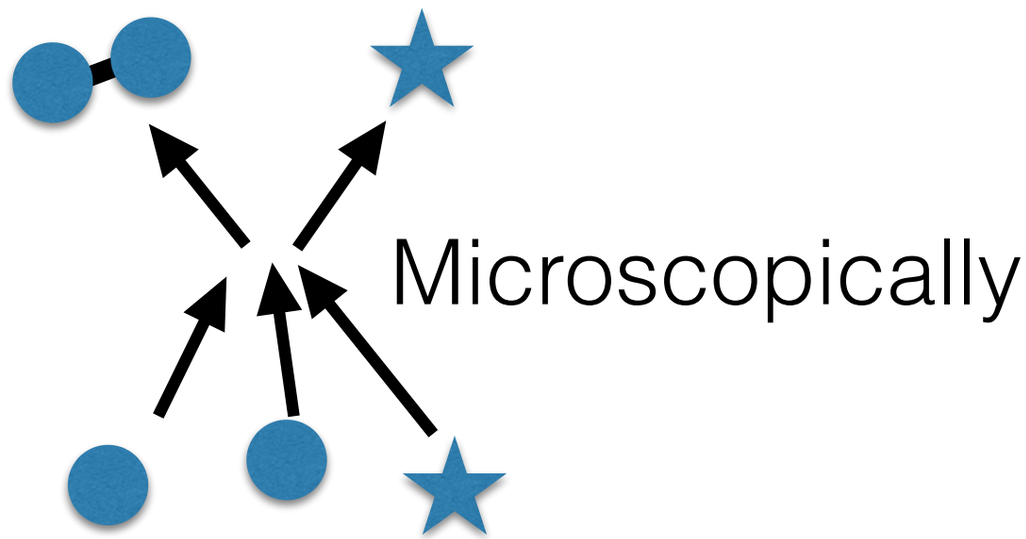
Generalized Arrhenius Law: $k_f = CT^b \exp \frac{-E_a}{RT}$

Determined Experimentally or from Quantum Chemistry

Backward Reactions (micro-reversibility)

$$\mathrm{O} + \mathrm{O} + \mathrm{M} \longrightarrow \mathrm{O_2} + \mathrm{M}$$

$$r_b = k_b[\mathrm{O}]^2[\mathrm{M}]$$



Forward and backward rates are interdependent!

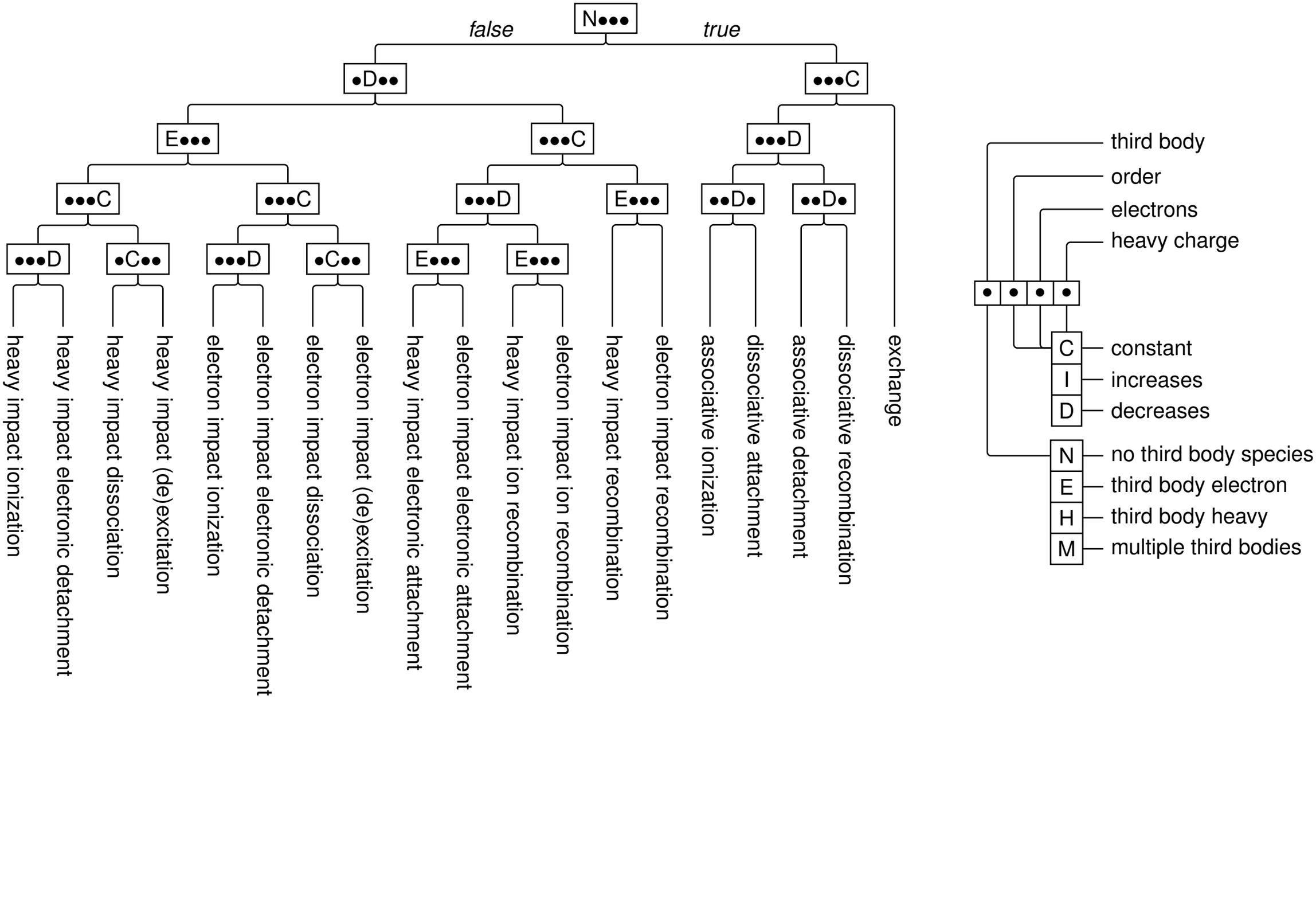
At Equilibrium: $r_f = r_b$

or: $\dfrac{k_f(T)}{k_b(T)} = \dfrac{[\mathrm{O_2}]}{[\mathrm{O}]^2}$

Equilibrium Constant: $K_C(T)$

Provided rigorously from thermodynamics

Automatic reaction classification



What happens when I have multiple reactions?

```
<!-- Park Air-5 Reaction Mechanism -->
<mechanism name="air5_park">

    <arrhenius_units A="mol,cm,s,K" E="kcal,mol,K" />

    <!-- 1 -->
    <reaction formula="N2+M=2N+M">
        <arrhenius A="3.0E+22" n="-1.6" T="113200.0" />
        <M>N2:0.2333, NO:0.2333, O2:0.2333</M>
    </reaction>

    <!-- 2 -->
    <reaction formula="O2+M=2O+M">
        <arrhenius A="1.0E+22" n="-1.5" T="59360.0" />
        <M>N2:0.2, NO:0.2, O2:0.2</M>
    </reaction>

    <!-- 3 -->
    <reaction formula="NO+M=N+O+M">
        <arrhenius A="5.0E15" n="+0.0" T="75500.0" />
        <M>NO:20.0, N:22.0, O:22.0</M>
    </reaction>

    <!-- 4 -->
    <reaction formula="N2+O=NO+N">
        <arrhenius A="5.69E+12" n="+0.42" T="42938.0" />
    </reaction>

    <!-- 5 -->
    <reaction formula="O2+N=NO+O">
        <arrhenius A="2.49E+09" n="+1.18" T="4005.5" />
    </reaction>
</mechanism>
```

What happens when I have multiple reactions?

```
<!-- Park Air-5 Reaction Mechanism -->
<mechanism name="air5_park">

    <arrhenius_units A="mol,cm,s,K" E="kcal,mol,K" />

    <!-- 1 -->
    <reaction formula="N2+M=2N+M">
        <arrhenius A="3.0E+22" n="-1.6" T="113200.0" />
        <M>N2:0.2333, NO:0.2333, O2:0.2333</M>
    </reaction>

    <!-- 2 -->
    <reaction formula="O2+M=2O+M">
        <arrhenius A="1.0E+22" n="-1.5" T="59360.0" />
        <M>N2:0.2, NO:0.2, O2:0.2</M>
    </reaction>

    <!-- 3 -->
    <reaction formula="NO+M=N+O+M">
        <arrhenius A="5.0E15" n="+0.0" T="75500.0" />
        <M>NO:20.0, N:22.0, O:22.0</M>
    </reaction>

    <!-- 4 -->
    <reaction formula="N2+O=NO+N">
        <arrhenius A="5.69E+12" n="+0.42" T="42938.0" />
    </reaction>

    <!-- 5 -->
    <reaction formula="O2+N=NO+O">
        <arrhenius A="2.49E+09" n="+1.18" T="4005.5" />
    </reaction>
</mechanism>
```

Pre-exp

b

Activation Temperature Ea/R

Third Body Efficiency

What happens when I have multiple reactions?

```
<!-- Park Air-5 Reaction Mechanism -->
<mechanism name="air5_park">

    <arrhenius_units A="mol,cm,s,K" E="kcal,mol,K" />

    <!-- 1 -->
    <reaction formula="N2+M=2N+M">
        <arrhenius A="3.0E+22" n="-1.6" T="113200.0" />
        <M>N2:0.2333, NO:0.2333, O2:0.2333</M>
    </reaction>

    <!-- 2 -->
    <reaction formula="O2+M=2O+M">
        <arrhenius A="1.0E+22" n="-1.5" T="59360.0" />
        <M>N2:0.2, NO:0.2, O2:0.2</M>
    </reaction>

    <!-- 3 -->
    <reaction formula="NO+M=N+O+M">
        <arrhenius A="5.0E15" n="+0.0" T="75500.0" />
        <M>NO:20.0, N:22.0, O:22.0</M>
    </reaction>

    <!-- 4 -->
    <reaction formula="N2+O=NO+N">
        <arrhenius A="5.69E+12" n="+0.42" T="42938.0" />
    </reaction>

    <!-- 5 -->
    <reaction formula="O2+N=NO+O">
        <arrhenius A="2.49E+09" n="+1.18" T="4005.5" />
    </reaction>
</mechanism>
```

Example:

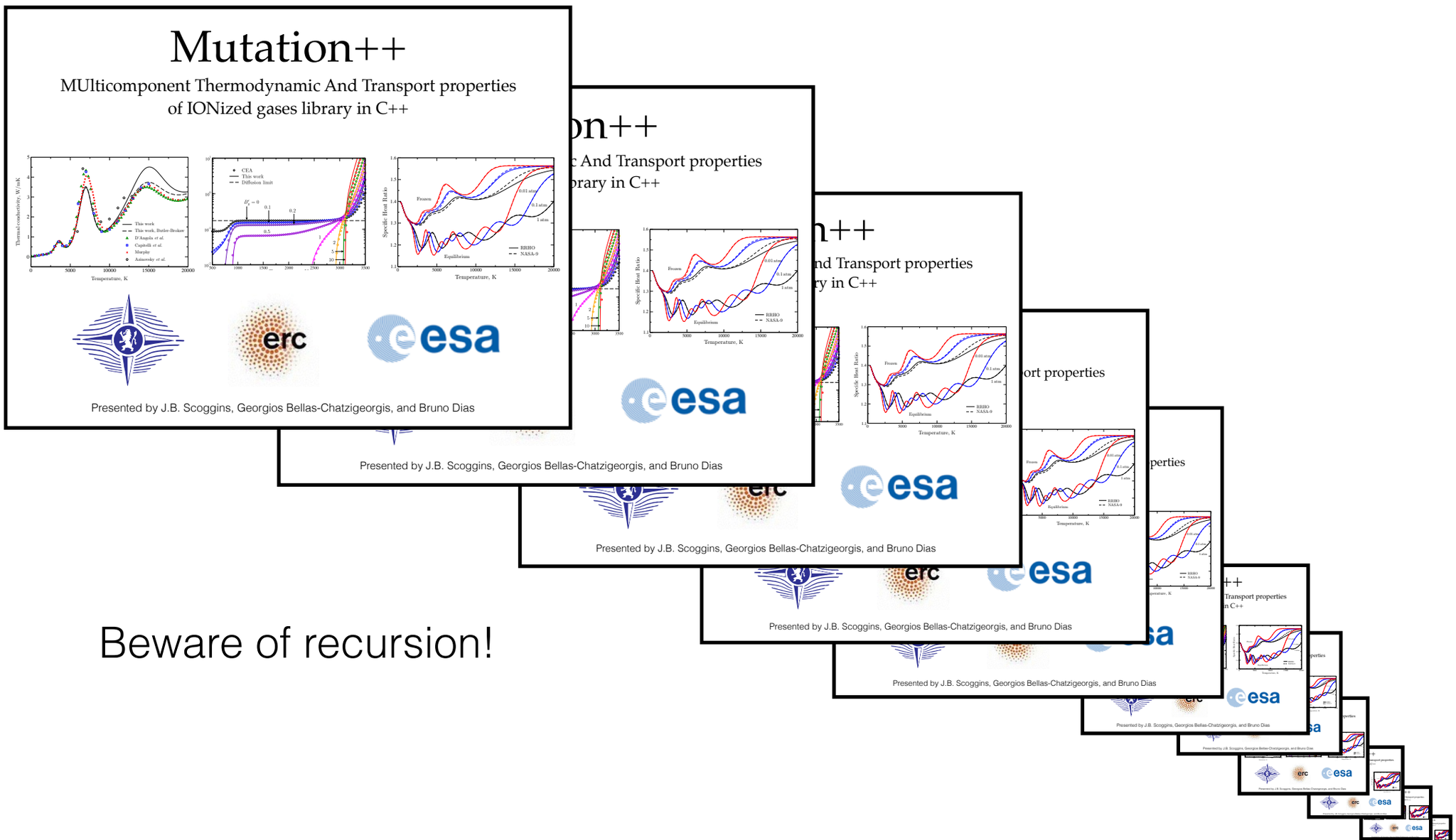
$$\frac{d[\mathrm{O}]}{dt} = \begin{array}{|l} \textbf{Forward} \\ +2k_{f,2}[\mathrm{O_2}][\mathrm{M}] \\ +k_{f,3}[\mathrm{NO}][\mathrm{M}] \\ -k_{f,4}[\mathrm{N_2}][\mathrm{O}] \\ +k_{f,5}[\mathrm{O_2}][\mathrm{N}] \\ \hline \textbf{Backward} \\ -2k_{b,2}[\mathrm{O}]^2[\mathrm{M}] \\ -k_{b,3}[\mathrm{N}][\mathrm{O}][\mathrm{M}] \\ +k_{b,4}[\mathrm{NO}][\mathrm{N}] \\ -k_{b,5}[\mathrm{NO}][\mathrm{O}] \end{array}$$

Lunch Time!

Before we begin…

Please download the training materials, including the example programs and this presentation:

https://tinyurl.com/yxk467g4



Hello World!

Hello ~~World!~~ Mixture

Printing species in a mixture

hello.cpp

```cpp
#include "mutation++.h"
#include <iostream>

int main(int argc, char* argv[])
{
    // Load air5 mixture
    Mutation::Mixture mix("air_5");

    // Print the species
    std::cout << "Species: ";
    for (auto& s : mix.species())
        std::cout << s.name() << ' ';
    std::cout << '\n';

    return 0;
}
```

hello.f90

```f90
program main
    use mutationpp
    implicit none
    character(len=10) :: mixture
    character(len=12) :: species, state_model
    integer :: j, ns

    mixture     = "air_5"
    state_model = "ChemNonEq1T"

    ! Initialize the mutation++ library
    call mpp_initialize(mixture, state_model)
    ns = mpp_nspecies()

    write(*,'(A12)',advance='no') "Species: "
    do j = 1,ns
        call mpp_species_name(j, species)
        write(*,'(A12)',advance='no') species
    end do
    write(*,*)

    ! Clean up the memory in the mutation++ library
    call mpp_destroy()

end program main
```

Using M++ in your own projects

Setting up dependencies

Makefile for C++ code.

```
# Change this path to your Mutation++ install path
MPP_INSTALL_DIR = $(MPP_DIRECTORY)/install

CXX_FLAGS = -I $(MPP_INSTALL_DIR)/include \
            -I $(MPP_INSTALL_DIR)/include/mutation++ \
            -I $(MPP_DIRECTORY)/thirdparty/eigen \
            -O3 -std=c++11

# use .dylib for Mac, .so for Unix
UNAME_S := $(shell uname -s)
ifeq ($(UNAME_S),Linux)
     CXX      = g++
     CXX_LIBS = $(MPP_INSTALL_DIR)/lib/libmutation++.so
endif
ifeq ($(UNAME_S),Darwin)
     CXX      = c++
     CXX_LIBS = $(MPP_INSTALL_DIR)/lib/libmutation++.dylib
endif

hello : hello.o
     $(CXX) -o $@ $(CXX_FLAGS) $^ $(CXX_LIBS)

%.o : %.cpp
     $(CXX) -c $(CXX_FLAGS) $<
```

Makefile for Fortran code.

```
# Change this path to your Mutation++ install path
MPP_INSTALL = $(MPP_DIRECTORY)/install

FC_FLAGS = -J$(MPP_INSTALL)/include/
           -O3 -g -fdefault-real-8


# use .dylib for Mac, .so for Unix
UNAME_S := $(shell uname -s)
ifeq ($(UNAME_S),Linux)
     FC       = gfortran
     FC_LIBS = $(MPP_INSTALL)/lib/libmutation++_fortran.so
endif
ifeq ($(UNAME_S),Darwin)
     FC       = gfortran
     FC_LIBS = $(MPP_INSTALL)/lib/libmutation++_fortran.dylib
endif

hello : hello.o
     $(FC) -o $@ $(FC_FLAGS) $^ $(FC_LIBS)

%.o : %.f90
     $(FC) -c $(FC_FLAGS) $<
```

Exercise

Compile and run the hello.cpp or hello.f90 programs.

Creating a Mixture from scratch

Understanding the MixtureOptions class

hello.cpp

```cpp
#include "mutation++.h"
#include <iostream>

int main(int argc, char* argv[])
{
    // Load air5 mixture
    Mutation::Mixture mix("air_5");

    // Print the species
    std::cout << "Species: ";
    for (auto& s : mix.species())
        std::cout << s.name() << ' ';
    std::cout << '\n';

    return 0;
}
```

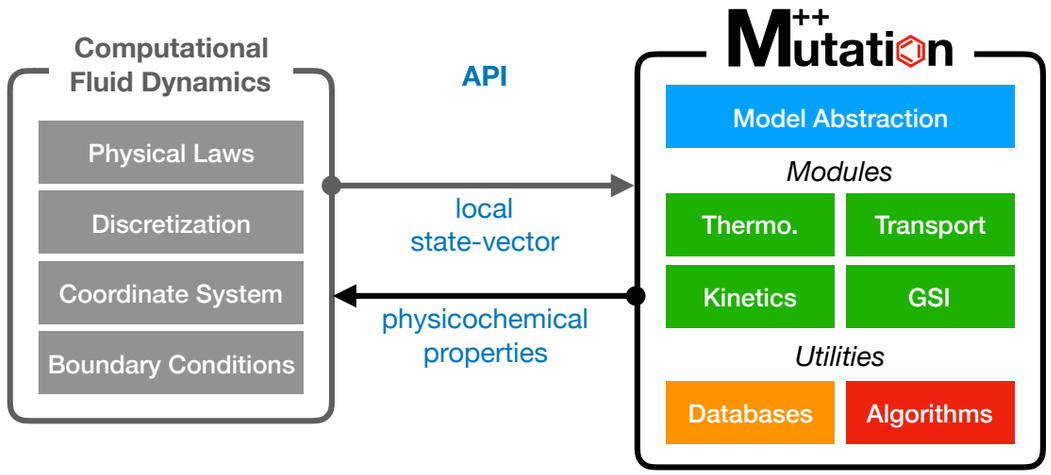
mixture_options.cpp

```cpp
#include "mutation++.h"
#include <iostream>

int main(int argc, char* argv[])
{
    // Setup mixture options
    Mutation::MixtureOptions options;
    options.setSpeciesDescriptor("N O NO N2 O2");
    options.setThermodynamicDatabase("RRHO");

    // Create the mixture
    Mutation::Mixture mix(options);

    // Print the species
    std::cout << "Species: ";
    for (auto& s : mix.species())
        std::cout << s.name() << ' ';
    std::cout << '\n';

    return 0;
}
```

Exercise

1) *Modify hello.cpp to use MixtureOptions as above.*
2) *Try using more generic species descriptor to load all species with N and O (including ions).*

Setting the state of the mixture




```
mix.setState(const double* const vec1, const double* const vec2, int vars = 0);
```

State Model	vars	vec1	vec2
Equil	0	mixture density	mixture energy density
	1	static pressure	temperature
	2	element mole fractions	pressure, temperature
ChemNonEq1T	0	species densities	mixture energy density
	1	species densities	temperature
	2	species mass fractions	pressure, temperature
ChemNonEqTTv	0	species densities	total and vibronic energy densities
	1	species densities	trans. and vibronic temperatures

*Natural variables for CFD are always the default (0) variable set.

Setting the state of the mixture

Example equilibrium code

set_state_equil.cpp

```
#include "mutation++.h"
#include <iostream>

int main(int argc, char* argv[])
{
    // Create the mixture
    Mutation::MixtureOptions options(“air_5");
    options.setStateModel("Equil");

    Mutation::Mixture mix(options);
    mix.addComposition("N2:0.79, O2:0.21", true);

    double P = Mutation::ONEATM;
    double T = 7000.0;
    mix.setState(&P, &T, 1);

    for (int i = 0; i < mix.nSpecies(); ++i) {
        std::cout << std::setw(10) << mix.speciesName(i);
        std::cout << std::setw(15) << mix.X()[i] << '\n';
    }

    return 0;
}
```

Exercise

Try above code with a pure nitrogen mixture.

Setting the state of the mixture

Example non-equilibrium code

set_state_noneq.cpp

```
int main(int argc, char* argv[])
{
    // Same setup as before…

    // Equilibrate the solution to T and P
    double P = Mutation::ONEATM; double T = 7000.0;
    mix.equilibrate(T, P);

    // Print mole fractions
    std::cout << "-- Before changing temperature --\n";
    std::cout << "Mole fractions:\n";
    printMoleFractions(mix);

    // Print the species production rates
    std::cout << "Production rates:\n";
    printProductionRates(mix);

    // Change the temperature but leave the densities the same
    std::vector<double> rho(mix.nSpecies());
    mix.densities(rho.data());
    T = 1000.0;
    mix.setState(rho.data(), &T, 1);

    // Print mole fractions
    std::cout << "-- After changing temperature --\n";
    std::cout << "Mole fractions:\n";
    printMoleFractions(mix);

    // Print the species production rates
    std::cout << "Production rates:\n";
    printProductionRates(mix);

    return 0;
}
```

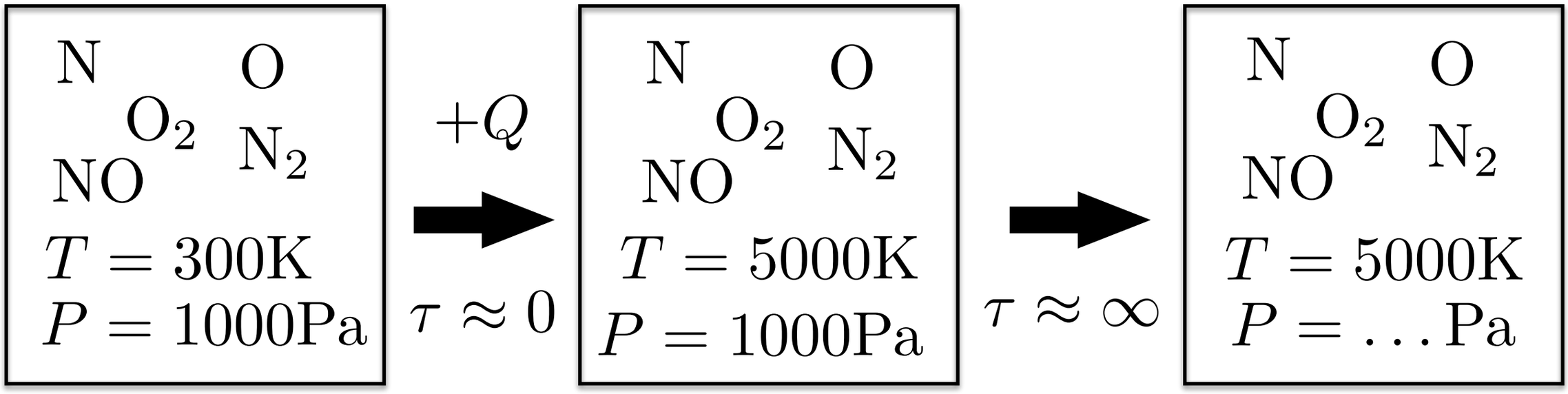
Output

```
-- Before changing temperature --
Mole fractions:
          N       0.499208
          O       0.258275
         NO     0.00258515
         N2       0.239898
         O2     3.42487e-05
Production rates:
          N      4.125e-13
          O   -1.66924e-13
         NO   -2.81131e-13
         N2   -2.81269e-13
         O2    3.16824e-13
-- After changing temperature --
Mole fractions:
          N       0.499208
          O       0.258275
         NO     0.00258515
         N2       0.239898
         O2     3.42487e-05
Production rates:
          N       -1036.07
          O       -331.057
         NO         320.88
         N2         886.286
         O2         159.962
```

Atoms are recombining at the lower temperature.

Now for a more interesting example

Implementing a 0D reactor at constant volume and temperature



$$\frac{d\rho_i}{dt} = \dot{\omega}_i \quad \xrightarrow{\text{Explicit}} \quad \rho_i^{n+1} = \rho_i^n + \Delta t \cdot \dot{\omega}_i^n$$

Instead: Runge-Kutta 4

$$\rho_i^{n+1} = \rho_i^n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$t^{n+1} = t^n + \Delta t$$

$$k_1 = \Delta t \cdot \dot{\omega}_i(\rho^n)$$

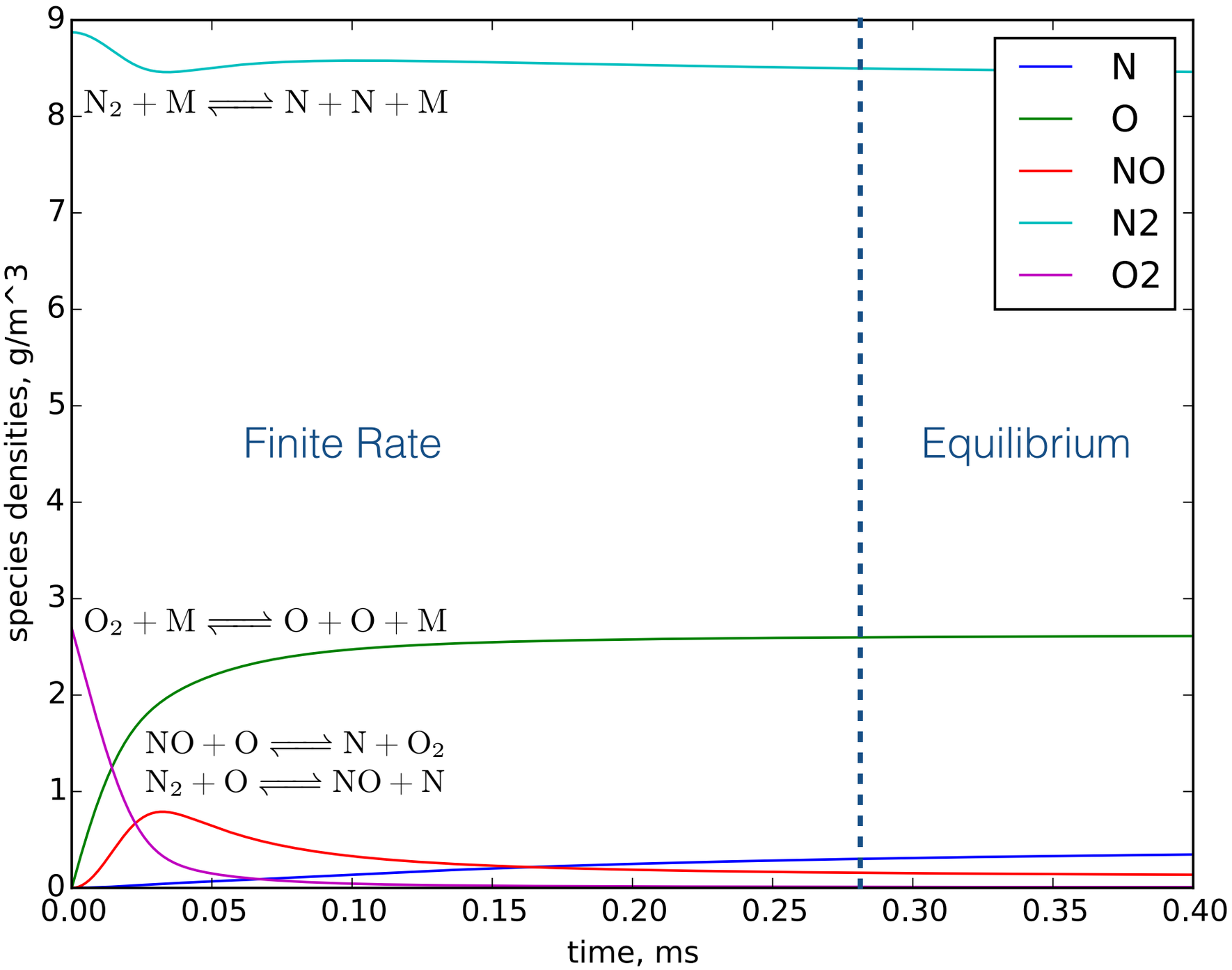
$$k_2 = \Delta t \cdot \dot{\omega}_i(\rho^n + \frac{k_1}{2})$$

$$k_3 = \Delta t \cdot \dot{\omega}_i(\rho^n + \frac{k_2}{2})$$

$$k_4 = \Delta t \cdot \dot{\omega}_i(\rho^n + k_3)$$

Results (cvisoRK4)

Implementing a 0D reactor at constant volume and temperature



What else is available in the library?
Building the API documentation

What you will need: Doxygen (http://www.stack.nl/~dimitri/doxygen/index.html)

1. Change to mutation++/build directory
2. Run `ccmake ..`
3. Turn `BUILD_DOCUMENTATION` to `ON`
4. Type ‘c’ ‘c’ ‘g’
5. Run `make docs`
6. Open mutation++/docs/html/index.html in you favorite browser

Happy Coding!