Opensource **M**icro**K**inetic **M**odeling Toolkit

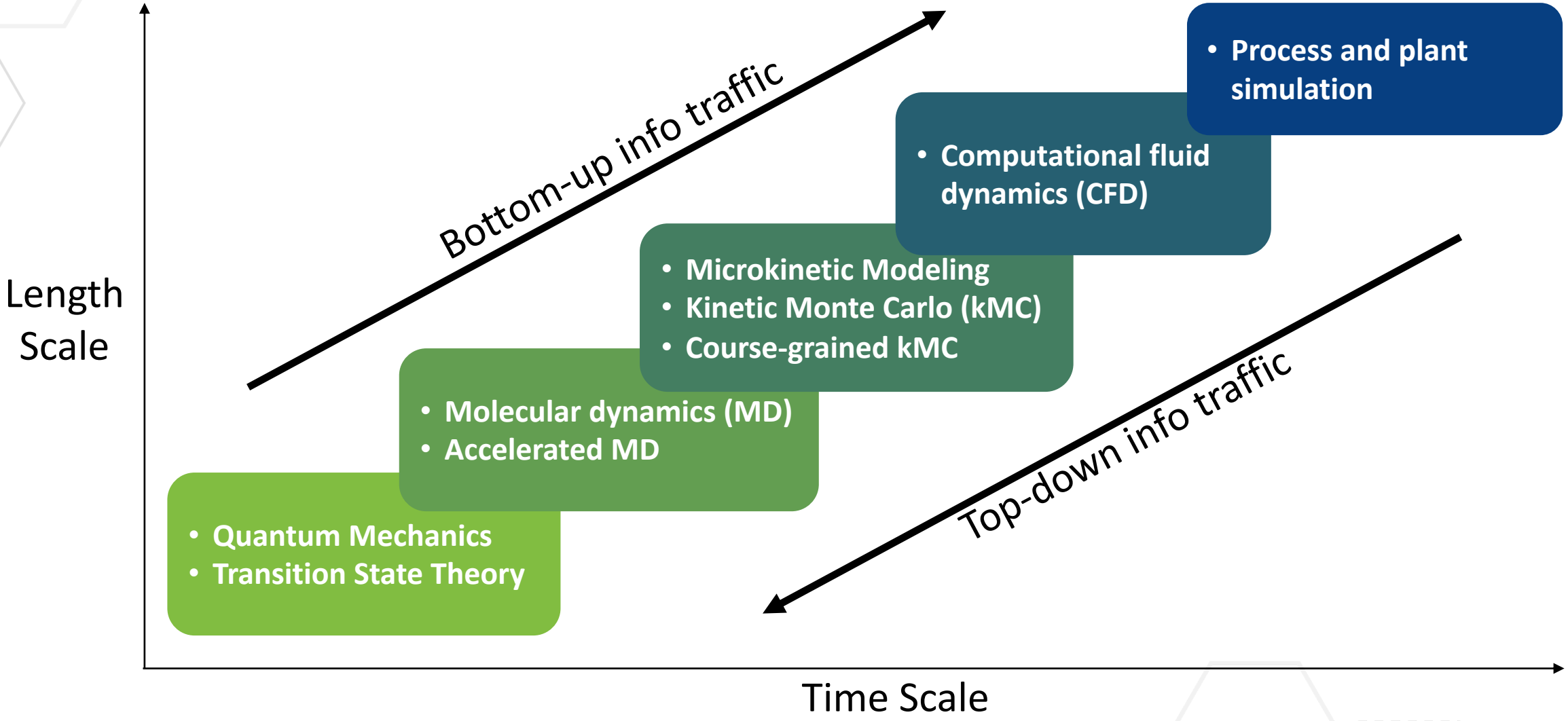
Bharat Medasani, Sashank Kasiraju, Dionisios Vlachos*
University of Delaware

https://github.com/VlachosGroup/openmkm

Medasani, B. et al. *J. Chem. Inf. Model.* **2023**, *63 (11)*, 4319–4355.

VlachosLab
AT THE UNIVERSITY OF DELAWARE

Multiscale Modeling

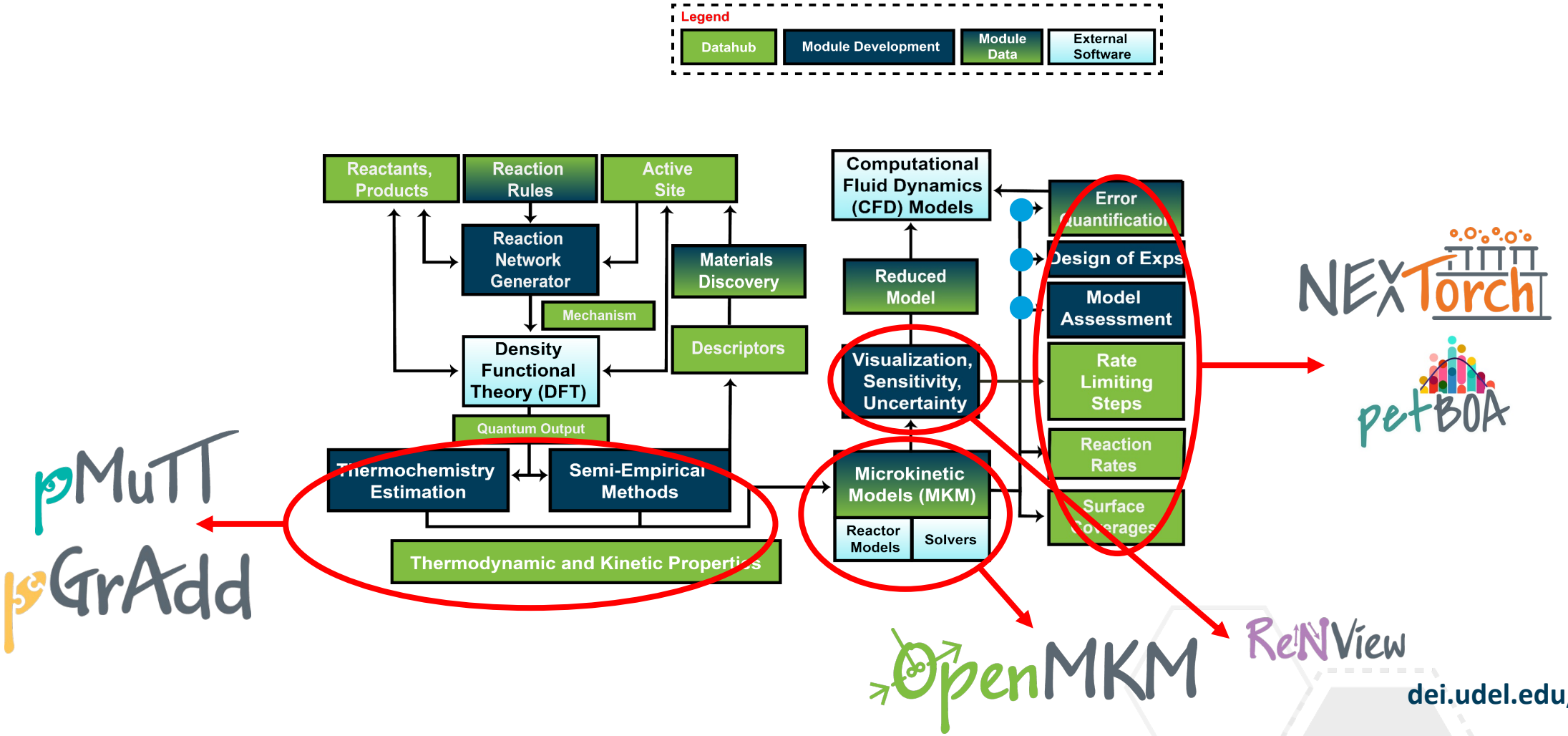


[1] Salciccioli, M. et al. *Chem. Eng. Sci.* **2011**, *66* (19), 4319–4355.

RAPID
Transforming Process Industries

Where does OpenMKM fit in V-Lab?

Bridge between atomic/meso scale data and macro-scale observables.



Microkinetic Model Inputs: Reactor Conditions

Legend

Set by user

Calculated by MKM

Derived from DFT

i – Species i

j – Reaction j

Microkinetic Model Inputs: Reactor Conditions

Q_0 = 1 mL s^{-1}

Inlet composition, x_0:

- N_2
- H_2
- Inert

T = 673 K

P = 100 bar

V = 1 cm^3

m_{cat} = 0.1 g Ru/Al_2O_3 catalyst

Legend

Set by user

Calculated by MKM

Derived from DFT

i – Species i

j – Reaction j

Microkinetic Model Inputs: Elementary Steps

Q_0 = 1 mL s^{-1}

Inlet composition, x_0:

- N_2
- H_2
- Inert

T = 673 K

P = 100 bar

V = 1 cm^3

m_{cat} = 0.1 g Ru/Al_2O_3 catalyst

Legend

Set by user

Calculated by MKM

Derived from DFT

i – Species i

j – Reaction j

Elementary reversible reactions:

- Adsorption/Desorption
  1. $N_2 + * \leftrightarrow N_2*$
  2. $H_2 + * \leftrightarrow 2H*$
  3. $NH_3 + * \leftrightarrow NH_3*$
- Surface Reactions
  4. $N_2* + * \leftrightarrow 2N*$
  5. $N* + H* \leftrightarrow NH*$
  6. $NH* + H* \leftrightarrow NH_2*$
  7. $NH_2* + H* \leftrightarrow NH_3*$

Microkinetic Model Inputs: Mass Balance for N_2^*

$$\frac{d}{dt}[N_2^*] = \underbrace{k_1^{fwd} P_{N_2}[*] - k_1^{rev}[N_2^*]}_{\text{Adsorption}} - \underbrace{k_4^{fwd}[N_2^*][*] + k_4^{rev}[N^*]^2}_{\text{Surface Reactions}}$$

Legend

Set by user

Calculated by MKM

Derived from DFT

i – Species i

j – Reaction j

Elementary reversible reactions:

- Adsorption/Desorption
  1. $N_2 + * \leftrightarrow N_2^*$
  2. $H_2 + * \leftrightarrow 2H^*$
  3. $NH_3 + * \leftrightarrow NH_3^*$
- Surface Reactions
  4. $N_2^* + * \leftrightarrow 2N^*$
  5. $N^* + H^* \leftrightarrow NH^*$
  6. $NH^* + H^* \leftrightarrow NH_2^*$
  7. $NH_2^* + H^* \leftrightarrow NH_3^*$

Microkinetic Model Inputs: Modified Arrhenius Rate Constant

$$\frac{d}{dt}[N_2^*] = k_1^{fwd} P_{N_2}[*] - k_1^{rev}[N_2^*] - k_4^{fwd}[N_2^*][*] + k_4^{rev}[N^*]^2$$

$$k_j^{fwd} = \underbrace{A_j' \left(\frac{T}{T_{ref,j}}\right)^{\beta_j}}_{A_{Arrhenius}} \exp\left(-\frac{E_{A,j}}{RT}\right)$$

Legend

Set by user

Calculated by MKM

Derived from DFT

i – Species i

j – Reaction j

- Arrhenius equation when $\beta_i = 1$ and $T_{ref,i} = 1\ K$

Microkinetic Model Inputs: Reverse Rate Constant

$$\frac{d}{dt}[N_2^*] = k_1^{fwd} P_{N_2}[*] - k_1^{rev}[N_2^*] - k_4^{fwd}[N_2^*][*] + k_4^{rev}[N^*]^2$$

$$k_j^{fwd} = A_j' \left(\frac{T}{T_{ref,j}}\right)^{\beta_j} \exp\left(-\frac{E_{A,j}}{RT}\right) \xrightarrow{\text{By microscopic reversibility}} k_j^{rev} = \frac{k_j^{fwd}}{K_j^{eq}}$$

$$K_j^{eq} = \exp\left(-\frac{\Delta G_j}{RT}\right)$$

Legend

Set by user

Calculated by MKM

Derived from DFT

i – Species i

j – Reaction j

Microkinetic Model Inputs: Adsorption Preexponential

$$\frac{d}{dt}[N_2^*] = k_1^{fwd} P_{N_2}[*] - k_1^{rev}[N_2^*] - k_4^{fwd}[N_2^*][*] + k_4^{rev}[N^*]^2$$

$$k_{ads,j}^{fwd} = A'_{ads,j}\left(\frac{T}{T_{ref,j}}\right)^{\beta_j} \exp\left(-\frac{E_{A,j}}{RT}\right)$$

By collision theory

$$A'_{ads,j} = \frac{s_j}{(\sigma)^{n_{surf}}}\sqrt{\frac{RT}{2\pi M_i}}$$

Legend

Set by user

Calculated by MKM

Derived from DFT

i – Species i

j – Reaction j

Microkinetic Model Inputs: Surface Reaction Preexponential

$$\frac{d}{dt}[N_2^*] = k_1^{fwd} P_{N_2}[*] - k_1^{rev}[N_2^*] - k_4^{fwd}[N_2^*][*] + k_4^{rev}[N^*]^2$$

$$k_{surf,j}^{fwd} = A'_{surf,j} \left(\frac{T}{T_{ref,j}}\right)^{\beta_j} \exp\left(-\frac{E_{A,j}}{RT}\right)$$

By transition-state theory

$$E_{A,j} = \Delta H_j^{\neq}$$

$$A'_{surf,j} = \frac{k_B}{h} \frac{1}{(\sigma)^{n_{surf}-1}} \exp\left(\frac{\Delta S_j^{\neq}}{R}\right)$$

Legend

Set by user

Calculated by MKM

Derived from DFT

i – Species i

j – Reaction j

Microkinetic Modeling Input Summary

Set by user

- Reactor conditions
  - T, P, Q, V, x_{i0}, m_{cat}
- Elementary steps
  - Adsorption
  - Surface reactions
- Reaction parameters
  - s_j, σ, $T_{ref,j}$, β_j

Calculated by MKM

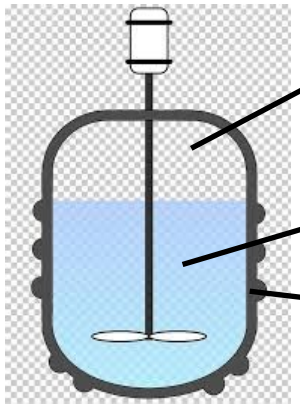
- Surface coverages
- Gas-phase compositions
- Rates

Derived from DFT/GA

- Enthalpies (H) and entropies (S) of:
  - Reactants
  - Products
  - Intermediates
  - Transition states

Stirred Tank Reactors: Heterogenous Catalysis

Batch Reactor



Volume
Temperature
Pressure

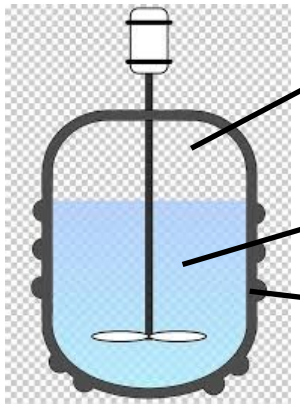
Composition of the fluid (chemical species)

Operational Mode: isothermal or adiabatic or external heat source/sink

A batch reactor is filled with reactants at a given temperature and pressure and the chemistry proceeds for a specified duration of time. No other inflow or outflow is allowed during the reaction.

Stirred Tank Reactors: Heterogenous Catalysis

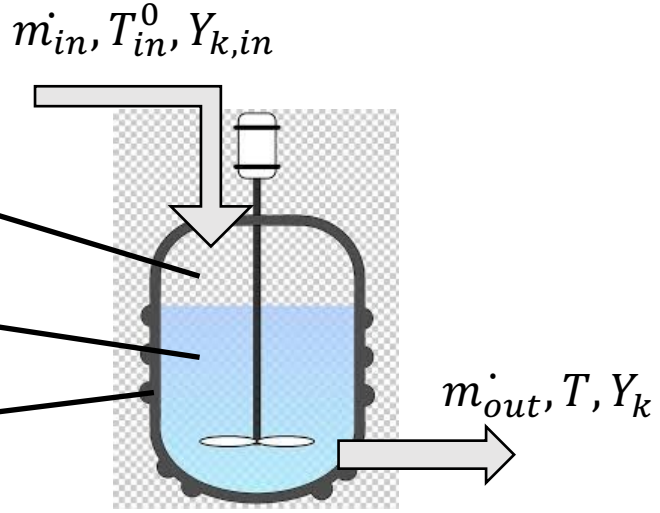
Batch Reactor



- Volume
  Temperature
  Pressure
- Composition of the fluid (chemical species)
- Operational Mode: isothermal or adiabatic or external heat source/sink

A batch reactor is filled with reactants at a given temperature and pressure and the chemistry proceeds for a specified duration of time. No other inflow or outflow is allowed during the reaction.

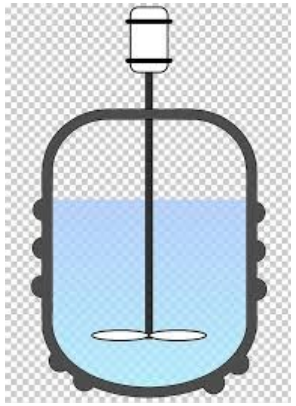
CSTR



A Continuously Stirred Tank Reactor (CSTR) has an inlet and an outlet to continuously add reactants and remove products. It is maintained at a given temperature and pressure and is typically operated at steady-state conditions.

Batch Reactor: With Catalytic Surface

Batch Reactor



Mass Balance

$$\frac{dm}{dt} = \cancel{\dot{m}_{in}}^{0} - \cancel{\dot{m}_{out}}^{0} + \dot{m}_{wall}$$

$$\frac{m\,dY_k}{dt} = \cancel{\dot{m}_{in}}^{0}\left(Y_{k,in} - Y_k\right) + \dot{m}_{k,gen} - Y_k\dot{m}_{wall}$$

$$\dot{m}_{k,gen} = V\dot{\omega}_k W_k + \dot{m}_{k,wall} \qquad \dot{m}_{k,wall} = W_k A_{cat}\dot{s}_k \qquad \dot{m}_{wall} = \sum_k \dot{m}_{k,wall}$$

m – Total mass of reactor contents
$\dot{m}_{wall}$ – Net mass flux from gas to catalyst surface
$\dot{m}_{k,gen}$ – Net mass production rate of species k
$\dot{\omega}_k$ – Net molar rate of species k from the gas phase
$\dot{s}_k$ – Net molar rate of species k from the catalyst surface
W_k – Molar weight of species k
Y_k – Mass fraction
A_{cat} – Surface area of the exposed catalyst

Energy Balance

$$mc_p\frac{dT}{dt} = \cancel{\dot{m}_{in}}^{0}\left(h_{in} - \sum_{k=1}^{K} h_k Y_{k,in}\right) - \sum_{k=1}^{K} h_k m_{k,gen} + \hat{h}A(T_{ext} - T)$$

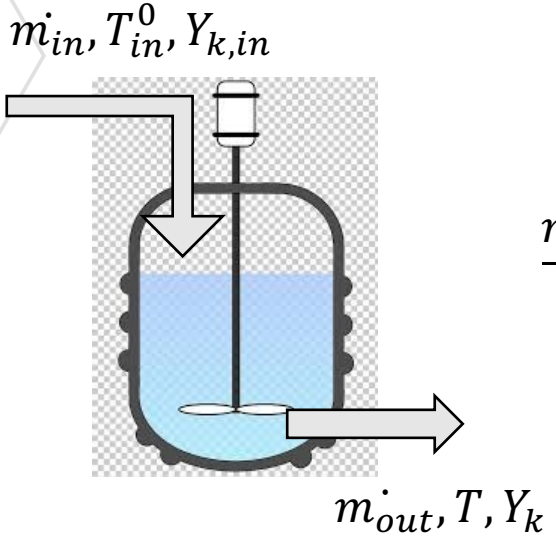
- c_p – Specific heat at const. pressure
- T – Temperature
- h_k – Specific enthalpy of kth species
- $\hat{h}$ – Heat conductance of wall
- T_{ext} – Temperature of heat source
- A – Area of the wall

Three Operational Modes

1. $\frac{dT}{dt} = 0$ – Isothermal
2. $\hat{h} = 0$ – Adiabatic
3. $T_{ext} \neq T$ – heat source/sink

CSTR: With Catalytic Surface

CSTR



Mass Balance

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

$$\frac{m\, dY_k}{dt} = \dot{m}_{in}\left(Y_{k,in} - Y_k\right) + \dot{m}_{k,gen} - Y_k \dot{m}_{wall}$$

$$\dot{m}_{k,gen} = V\dot{\omega}_k W_k + \dot{m}_{k,wall} \qquad \dot{m}_{k,wall} = W_k A_{cat} \dot{s}_k \qquad \dot{m}_{wall} = \sum_k \dot{m}_{k,wall}$$

- m – Total mass of reactor contents
- $\dot{m}_{wall}$ – Net mass flux from gas to catalyst surface
- $\dot{m}_{k,gen}$ – Net mass production rate of species k
- $\dot{\omega}_k$ – Net molar rate of species k from the gas phase
- $\dot{s}_k$ – Net molar rate of species k from the catalyst surface
- W_k – Molar weight of species k
- Y_k – Mass fraction
- A_{cat} – Surface area of the exposed catalyst

Energy Balance

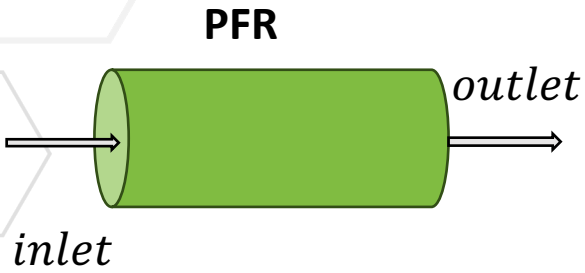
$$mc_p \frac{dT}{dt} = \dot{m}_{in}\left(h_{in} - \sum_{k=1}^{K} h_k Y_{k,in}\right) - \sum_{k=1}^{K} h_k m_{k,gen} + \hat{h}A(T_{ext} - T)$$

- c_p – Specific heat at const. pressure
- T – Temperature
- h_k – Specific enthalpy of kth species
- $\hat{h}$ – Heat conductance of wall
- T_{ext} – Temperature of heat source
- A – Area of the wall

Three Operational Modes

1. $\frac{dT}{dt} = 0$ – Isothermal
2. $\hat{h} = 0$ – Adiabatic
3. $T_{ext} \neq T$ – heat source/sink

PFR-1D (Numerical): With Catalytic Surface



ρ – density of reactor contents

$\dot{\omega}_k$ – Net molar rate of species k from the gas phase

$\dot{s}_k$ – Net molar rate of species k from the catalyst surface

W_k – Molar weight of species k

Y_k – Mass fraction

A_{cat} – Surface area of the exposed catalyst

Mass Balance

$$\rho v \frac{dY_k}{dz} + \frac{A_{Cat}}{V} Y_k \sum_k \dot{s_k} W_k = \left(\dot{\omega_k} + \frac{s_k \dot{A}_{cat}}{V} \right) W_k$$

Energy Balance

$$\rho c_p \frac{dT}{dz} = -\sum_{k=1}^{K} h_k W_k \dot{\omega}_k - \frac{A_{Cat}}{V} \sum_{k=1}^{K} h_k W_k \dot{s}_k + \frac{\hat{h} A_{wall}}{V} (T_{ext} - T)$$

- c_p – Specific heat at const. pressure
- T – Temperature
- h_k – Specific enthalpy of kth species
- $\hat{h}$ – Heat conductance of wall
- T_{ext} – Temperature of heat source
- A – Area of the wall

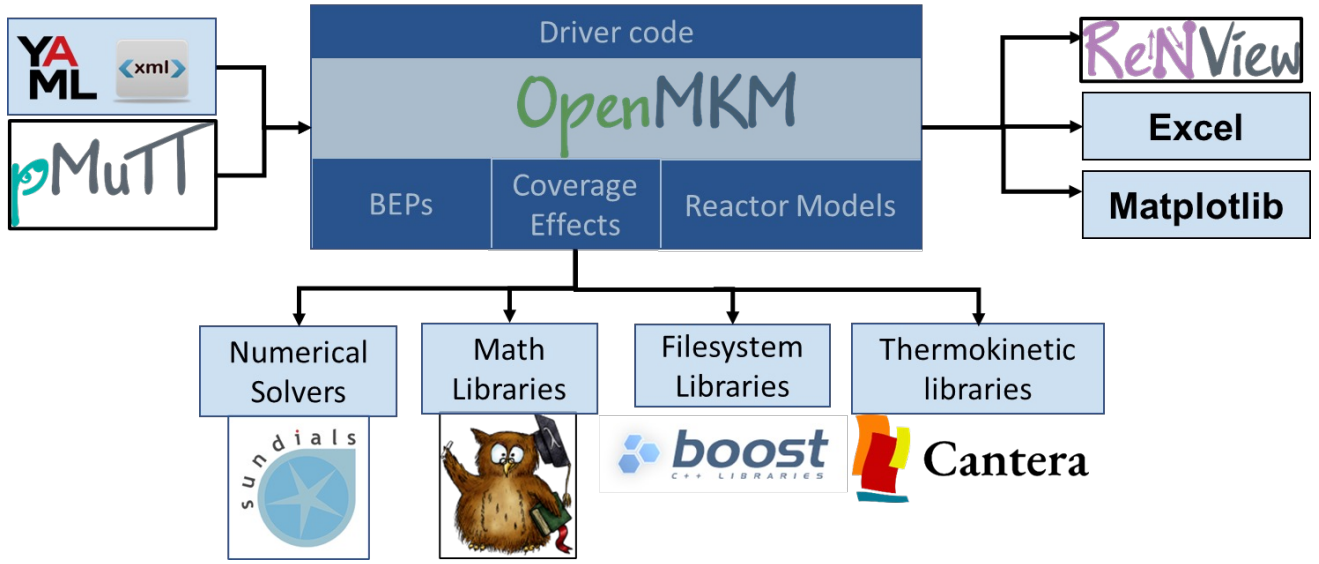
Three Operational Modes

1. $\frac{dT}{dz} = 0$ – Isothermal
2. $\hat{h} = 0$ – Adiabatic
3. $T_{ext} \neq T$ – Heat source/sink
4. $T(z) = \mathrm{f}(\mathrm{z})$ – Custom T-Profile

OpenMKM: Core Modules

Uses state-of-the art libraries for stability and robustness.

- OpenMKM's core modules are derived from:
  - The Cantera[1] library
  - Sundials[2] ODE/DAE solvers
  - Boost[3] C++ libraries
- YAML Input files are generated easily using
  - The pMuTT[4] library
  - Optionally also using YAML[2] API



[1] https://cantera.org
[2] https://computing.llnl.gov/projects/sundials
[3] https://www.boost.org/
[4] https://vlachosgroup.github.io/pMuTT/
[5] https://yaml.org/

https://github.com/VlachosGroup/openmkm
Medasani, B. et al. *J. Chem. Inf. Model.* **2023**, *63 (11)*, 4319–4355.

Cantera: Thermodynamic Specification

Units and Phases

```
units: {length: cm, time: s, quantity: mol, activation-energy: cal/mol}

phases:
- name: gri30
  thermo: ideal-gas
  elements: [O, H, C, N, Ar]
  species: [H2, H, O, O2, OH, H2O, HO2, H2O2, C, CH, CH2, CH2(S), CH3, CH4,
    CO, CO2, HCO, CH2O, CH2OH, CH3O, CH3OH, C2H, C2H2, C2H3, C2H4, C2H5,
    C2H6, HCCO, CH2CO, HCCOH, N, NH, NH2, NH3, NNH, NO, NO2, N2O, HNO, CN,
    HCN, H2CN, HCNN, HCNO, HOCN, HNCO, NCO, N2, AR, C3H7, C3H8, CH2CHO,
    CH3CHO]
  kinetics: gas
  transport: mixture-averaged
  state: {T: 300.0, P: 1 atm}
```

Species

```
species:
- name: H2
  composition: {H: 2}
  thermo:
    model: NASA7
    temperature-ranges: [200.0, 1000.0, 3500.0]
    data:
    - [2.34433112, 7.98052075e-03, -1.9478151e-05, 2.01572094e-08, -7.37611761e-12,
      -917.935173, 0.683010238]
    - [3.3372792, -4.94024731e-05, 4.99456778e-07, -1.79566394e-10, 2.00255376e-14,
      -950.158922, -3.20502331]
    note: TPIS78
```

Reactions

```
reactions:
- equation: 2 O + M <=> O2 + M  # Reaction 1
  type: three-body
  rate-constant: {A: 1.2e+17, b: -1.0, Ea: 0.0}
  efficiencies: {H2: 2.4, H2O: 15.4, CH4: 2.0, CO: 1.75, CO2: 3.6, C2H6: 3.0,
    AR: 0.83}
- equation: O + H + M <=> OH + M  # Reaction 2
  type: three-body
  rate-constant: {A: 5.0e+17, b: -1.0, Ea: 0.0}
  efficiencies: {H2: 2.0, H2O: 6.0, CH4: 2.0, CO: 1.5, CO2: 2.0, C2H6: 3.0,
    AR: 0.7}
- equation: O + H2 <=> H + OH  # Reaction 3
  rate-constant: {A: 3.87e+04, b: 2.7, Ea: 6260.0}
- equation: O + HO2 <=> OH + O2  # Reaction 4
  rate-constant: {A: 2.0e+13, b: 0.0, Ea: 0.0}
- equation: O + H2O2 <=> OH + HO2  # Reaction 5
  rate-constant: {A: 9.63e+06, b: 2.0, Ea: 4000.0}
- equation: O + CH <=> H + CO  # Reaction 6
```

http://combustion.berkeley.edu/gri-mech/version30/text30.html

https://cantera.org

Cantera[1] Modeling Concepts

- **Reactor**: Represents the simplest form of a homogeneous chemically reacting system.
- **Reservoir**: Represents an infinitely large volume which can depict a fixed thermodynamic state or reference.
- **Valve**: Implements mass flow as a function of pressure drop.
- **Mass Flow Controller**: Maintains mass flow rate either at a constant value or a function of time.
- **Wall**: Can be used to perform work on the reactor via expansion or contraction(e.g., an internal combustion engine)or to facilitate mass or heat transfer across boundaries.
- **Surface**: Supports heterogeneous reactions(e.g., surface of a catalyst).
- **ReactorNet**: Denotes a network of reactors connected to reservoirs, values, flow controllers, etc.

Template Cantera C++ reactor code (traditional way)

```
using namespace Cantera;

void runexample()
{
    // use reaction mechanism GRI-Mech 3.0
    auto sol = newSolution("gri30.yaml", "gri30", "none");
    auto gas = sol->thermo();

    // create a reservoir for the fuel inlet, and set to pure methane.
    Reservoir fuel_in;
    gas->setState_TPX(300.0, OneAtm, "CH4:1.0");
    fuel_in.insert(sol);
    double fuel_mw = gas->meanMolecularWeight();

    auto air = newSolution("air.yaml", "air", "none");
    double air_mw = air->thermo()->meanMolecularWeight();

    // create a reservoir for the air inlet
    Reservoir air_in;
    air_in.insert(air);

    // to ignite the fuel/air mixture, we'll introduce a pulse of radicals.
    // The steady-state behavior is independent of how we do this, so we'll
    // just use a stream of pure atomic hydrogen.
    gas->setState_TPX(300.0, OneAtm, "H:1.0");
    Reservoir igniter;
    igniter.insert(sol);


    // create the combustor, and fill it in initially with N2
    gas->setState_TPX(300.0, OneAtm, "N2:1.0");
    Reactor combustor;
    combustor.insert(sol);
    combustor.setInitialVolume(1.0);
```

https://cantera.org/examples/cxx/combustor.html

[1] https://cantera.org
[2] https://computing.llnl.gov/projects/sundials

Cantera[1] Modeling Concepts

- **Reactor**: Represents the simplest form of a homogeneous chemically reacting system.
- **Reservoir**: Represents an infinitely large volume which can depict a fixed thermodynamic state or reference.
- **Valve**: Implements mass flow as a function of pressure drop.
- **Mass Flow Controller**: Maintains mass flow rate either at a constant value or a function of time.
- **Wall**: Can be used to perform work on the reactor via expansion or contraction(e.g., an internal combustion engine)or to facilitate mass or heat transfer across boundaries.
- **Surface**: Supports heterogeneous reactions(e.g., surface of a catalyst).
- **ReactorNet**: Denotes a network of reactors connected to reservoirs, values, flow controllers, etc.

Template Cantera C++ reactor code (traditional way)

```
combustor.setInitialVolume(1.0);


// create a reservoir for the exhaust. The initial composition
// doesn't matter.
Reservoir exhaust;
exhaust.insert(sol);


// lean combustion, phi = 0.5
double equiv_ratio = 0.5;

// compute fuel and air mass flow rates
double factor = 0.1;
double air_mdot = factor*9.52*air_mw;
double fuel_mdot = factor*equiv_ratio*fuel_mw;

// create and install the mass flow controllers. Controllers
// m1 and m2 provide constant mass flow rates, and m3 provides
// a short Gaussian pulse only to ignite the mixture
MassFlowController m1;
m1.install(fuel_in, combustor);
m1.setMassFlowRate(fuel_mdot);

// Now create the air mass flow controller.  Note that this connects
// two reactors with different reaction mechanisms and different
// numbers of species. Downstream and upstream species are matched by
// name.
MassFlowController m2;
m2.install(air_in, combustor);
m2.setMassFlowRate(air_mdot);
```

https://cantera.org/examples/cxx/combustor.html

[1] https://cantera.org
[2] https://computing.llnl.gov/projects/sundials

Cantera[1] Modeling Concepts

- **Reactor**: Represents the simplest form of a homogeneous chemically reacting system.
- **Reservoir**: Represents an infinitely large volume which can depict a fixed thermodynamic state or reference.
- **Valve**: Implements mass flow as a function of pressure drop.
- **Mass Flow Controller**: Maintains mass flow rate either at a constant value or a function of time.
- **Wall**: Can be used to perform work on the reactor via expansion or contraction(e.g., an internal combustion engine)or to facilitate mass or heat transfer across boundaries.
- **Surface**: Supports heterogeneous reactions(e.g., surface of a catalyst).
- **ReactorNet**: Denotes a network of reactors connected to reservoirs, values, flow controllers, etc.

Rapidly evolving Python interface (currently popular) which uses Python/C++ bindings underneath it

```
import numpy as np
import matplotlib.pyplot as plt
import cantera as ct

# Use reaction mechanism GRI-Mech 3.0. For 0-D simulations,
# no transport model is necessary.
gas = ct.Solution('gri30.yaml')

# Create a Reservoir for the inlet, set to a methane/air mixture at a
# equivalence ratio
equiv_ratio = 0.5  # lean combustion
gas.TP = 300.0, ct.one_atm
gas.set_equivalence_ratio(equiv_ratio, 'CH4:1.0', 'O2:1.0, N2:3.76')
inlet = ct.Reservoir(gas)

# Create the combustor, and fill it initially with a mixture consist
# equilibrium products of the inlet mixture. This state corresponds
# the reactor would reach with infinite residence time, and thus pro
# initial condition from which to reach a steady-state solution on t
# branch.
gas.equilibrate('HP')
combustor = ct.IdealGasReactor(gas)
combustor.volume = 1.0

# Create a reservoir for the exhaust
exhaust = ct.Reservoir(gas)
```

https://cantera.org/examples/python/reactors/combustor.py.html

[1] https://cantera.org
[2] https://computing.llnl.gov/projects/sundials

OpenMKM – Integration with Cantera

- Current latest version of OpenMKM is based on Cantera v2.5
- We forked Cantera v2.5 to add new and modify existing source code which enable new additional features and implementations such as BEPs and adsorbate lateral interactions.
  - For a list of all changes between the forked Cantera version please see the GitHub comparison log between forks: Link
- Cantera simulations require a thermodynamic specification file. OpenMKM leverages the Cantera-fork to work with this file. This input file needs to be modified to enable additional OpenMKM features. Options for file-format:
  - CTI File (deprecated from Cantera v3.0), Not supported by OpenMKM
  - CTML/XML File (also deprecated from Cantera v3.0), Supported by OpenMKM
  - YAML (current standard), Supported by OpenMKM

OpenMKM thermodynamics specification

- File used for thermodynamics and kinetics
- File broken into blocks:
  - Units
  - Phases
  - Species
  - Reactions
  - **Lateral Interactions (optional)**
  - **BEPs (optional)**

```yaml
interactions:
    - species: ["N(T)", "N(T)"]
      strength: [-52.6 kcal/mol]
      coverage-threshold: [0, 1]
      id: "i0000"
    - species: ["N(T)", "H(T)"]
      strength: [-17.7 kcal/mol]
      coverage-threshold: [0, 1]
      id: "i0001"
    - species: ["H(T)", "N(T)"]
      strength: [-17.7 kcal/mol]
      coverage-threshold: [0, 1]
      id: "i0002"
    - species: ["H(T)", "H(T)"]
      strength: [-3.0 kcal/mol]
      coverage-threshold: [0, 1]
      id: "i0003"
```

```yaml
bep:
    - id: "NH2-H"
      slope: 0.71
      intercept: 23.69 kcal/mol
      direction: "cleavage"
      cleavage-reactions: ["NH2-H_cle_0001", "NH2-H_cle_0002"]

    - id: "NH-H"
      slope: 0.52
      intercept: 19.78 kcal/mol
      direction: "cleavage"
      cleavage-reactions: ["NH-H_cle_0001", "NH-H_cle_0002"]

    - id: "N-H"
      slope: 0.29
      intercept: 23.23 kcal/mol
      direction: "cleavage"
      cleavage-reactions: ["N-H_cle_0001"]
      synthesis-reactions: ["N-H_syn_0001"]
```

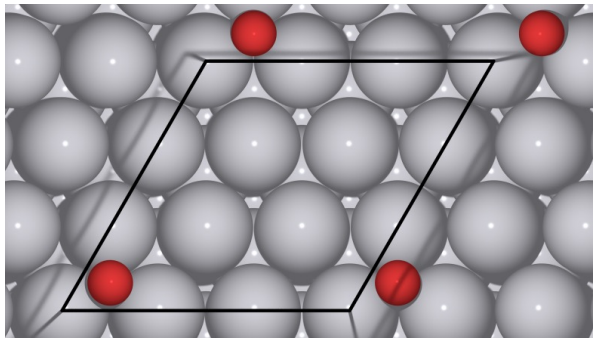
https://github.com/VlachosGroup/openmkm
https://github.com/VlachosGroup/openmkm/blob/master/docs/input.md
Medasani, B. et al. *J. Chem. Inf. Model.* **2023**, *63 (11)*, 4319–4355.

Modelling Coverage Effects using DFT

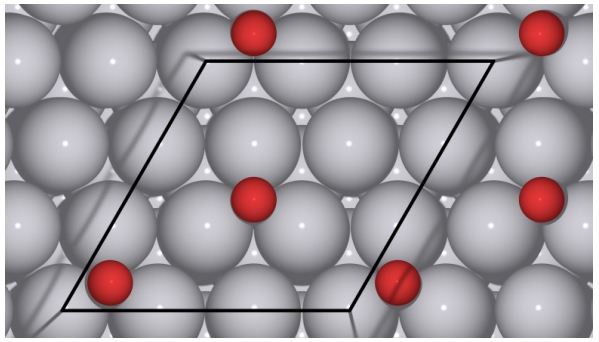
$$H_i(T, \boldsymbol{\theta}) = \underbrace{H_i(T, \theta_* \to 0)}_{\text{Low coverage}} + \underbrace{\sum_{j}^{\text{Adsorbates}} \omega_{ijk}\theta_j + b_{ijk}}_{\text{Finite coverage}}$$

- Gradually increase coverage in DFT cell
- Calculate binding energy as coverage increases
- Linear piece-wise is often sufficient

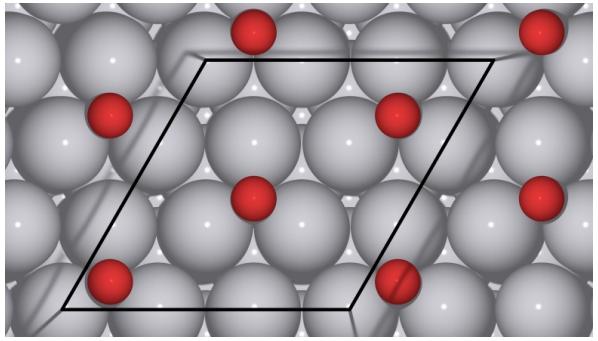
$\theta_O = \frac{1}{9}$ ML



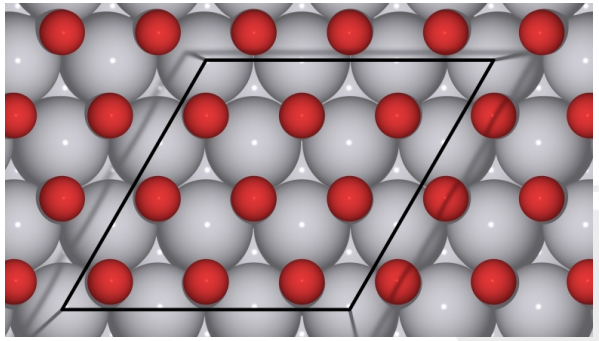
$\theta_O = \frac{2}{9}$ ML



$\theta_O = \frac{3}{9}$ ML



$\theta_O = 1$ ML



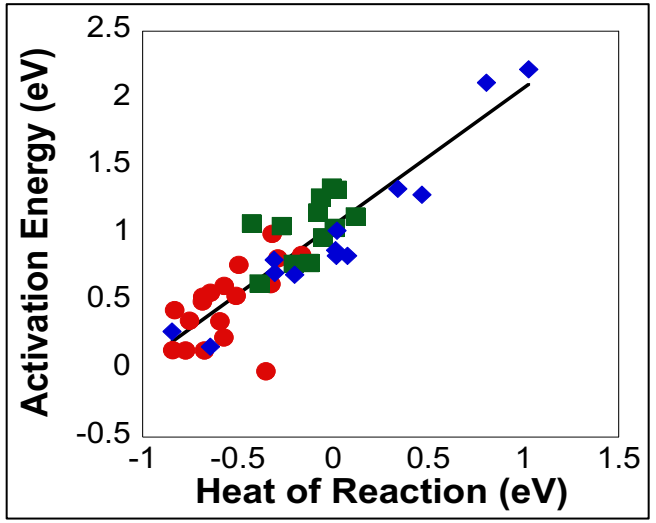
Mhadeshwar, A. B. et al. *Catal. Letters* **2004**, *96* (1–2), 13–22

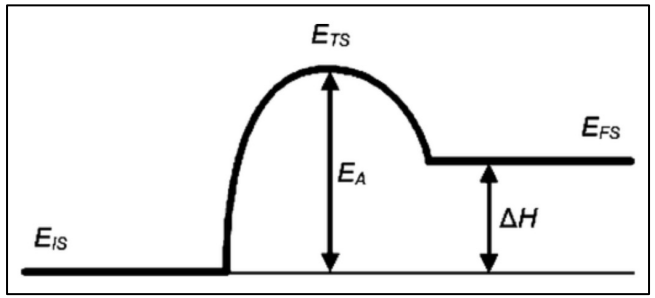
Brønsted–Evans–Polanyi (BEP) Relationships

$$E_{A,j} = \omega \Delta H_j + \Delta E_0$$

Directions supported:

- Synthesis ($NH_2^* + H^* \rightarrow NH_3^*$)
- Cleavage ($NH_3^* \rightarrow NH_2^* + H^*$)






```
bep(id="NH2-H",
    slope=0.71,
    intercept=23.69,
    direction="cleavage",
    cleavage_reactions=["0001 to 0002"],
    synthesis_reactions=[])
```

[1] Sutton and Vlachos, *ACS Catal.*, **2**, 1624 (2012)

OpenMKM: Reactor File Specification

- YAML data format
- Like a python dictionary (key: value pairs)
- 3 top level mandatory keys / YAML Nodes
  - **reactor**
  - **simulation**
  - **phases**
- One additional top-level node (mandatory for CSTR and PFR models)
  - **inlet_gas**

https://github.com/VlachosGroup/openmkm/blob/master/docs/input.md

OpenMKM[1]: Code Organization

OpenMKM Infrastructure

- **Solvers**: Classes and methods dedicated to interface with the Sundials solvers.
- **I/O Methods**: Classes and methods dedicated for parsing reactor specification file and printing out simulation output.
- **Reactor Models**: C++ classes that extend the reactor models implemented in Cantera as well as those that define new reactor models and utilize the corresponding I/O operations.
- **Driver Code**: Assembles and calls the required modules to solve the problem defined in the input file.

```
.
├── CMakeLists.txt  - CMAKE Build files
├── README.md
├── io.cpp - File input/output methods
├── main.cpp - Driver Code
├── onedReactor.cpp - One-D Reactor methods
├── pfr1d.cpp - Numerical PFR Model
├── pfr1d_solver.cpp - Setup Sundials IDAS DAE solver
├── reactor_parser.cpp - Parse the reactor.yaml file
├── util.cpp - Miscellaenous Utilities
└── zerodReactor.cpp - Zero-D Reactor models (Batch, CSTR etc.)
```

https://github.com/VlachosGroup/openmkm

Medasani, B. et al. *J. Chem. Inf. Model.* **2023**, *63 (11)*, 4319–4355.

Why use OpenMKM?

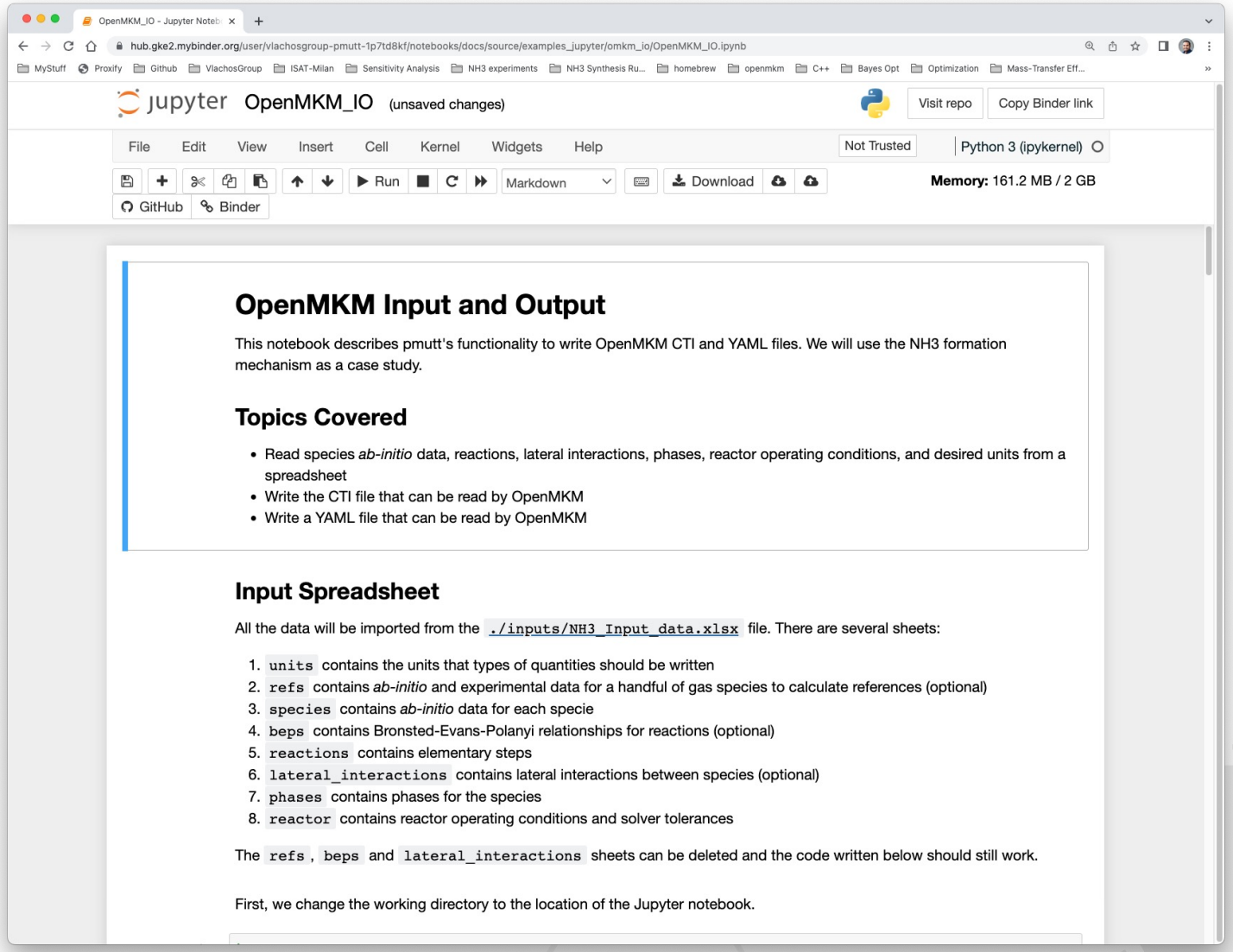
- **OpenMKM is super fast!** Uses state-of-the art libraries for stability and robustness.

- Model: Methane conversion to C2+ chemicals
  - 9286 gas reactions and 47surface reactions, 501 gas species, 38 surface species
  - Steady State, PFR as 20 CSTRs.
- Simulation time: OpenMKM (~**24 s**)

```
skasiraj@host-17-102:gas+surface % tail -3 omkm/general_info.out
CSTR #: 18
CSTR #: 19
Program ran for 23887 milliseconds
```

Why use OpenMKM?

- OpenMKM is super fast! Uses state-of-the art libraries for stability and robustness.
- **OpenMKM easy to use with just two inputs; and pMuTT can natively make them.**



[1] https://vlachosgroup.github.io/pMuTT/

Why use OpenMKM?

- OpenMKM is super fast! Uses state-of-the art libraries for stability and robustness.
- **OpenMKM easy to use with just two inputs; and pMuTT can natively make them.**

YAML

pMuTT

[1] https://vlachosgroup.github.io/pMuTT/

reactor.yaml

```yaml
# File generated by pMuTT (v 1.2.21) on 20
# See documentation for OpenMKM YAML file
# https://vlachosgroup.github.io/openmkm/i
inlet_gas:
    flow_rate: "1 cm3/s"
phases:
    bulk:
        name: bulk
    gas:
        initial_state: "N2:0.25, H2:0.75"
        name: gas
    surfaces:
    -   initial_state: "RU(T):1.0"
        name: terrace
    -   initial_state: "RU(S):1.0"
        name: step
reactor:
    cat_abyv: "1200 /cm"
    mode: "isothermal"
    pressure: "1 atm"
    temperature: 900
    type: "cstr"
    volume: "1 cm3"
simulation:
    end_time: "4 s"
    init_step: 1.0e-08
    output_format: "csv"
    solver:
        atol: 1.0e-10
        rtol: 1.0e-05
    stepping: "logarithmic"
    transient: true
```

Reactor specification

reactor.yaml | thermo.yaml

Users > skasiraj > work-temp > openMKM-SI-examples > Example-2

```yaml
phases:
- name: gas
  elements: [H, N]
  species: [N2, NH3, H2]
  thermo: ideal-gas
  kinetics: gas
  reactions: none
- name: bulk…
- name: terrace…
- name: step…


#-------------------------------------------------------------------------------
# SPECIES
#-------------------------------------------------------------------------------
species:
- name: N2
  composition: {N: 2.0}
  thermo:
```

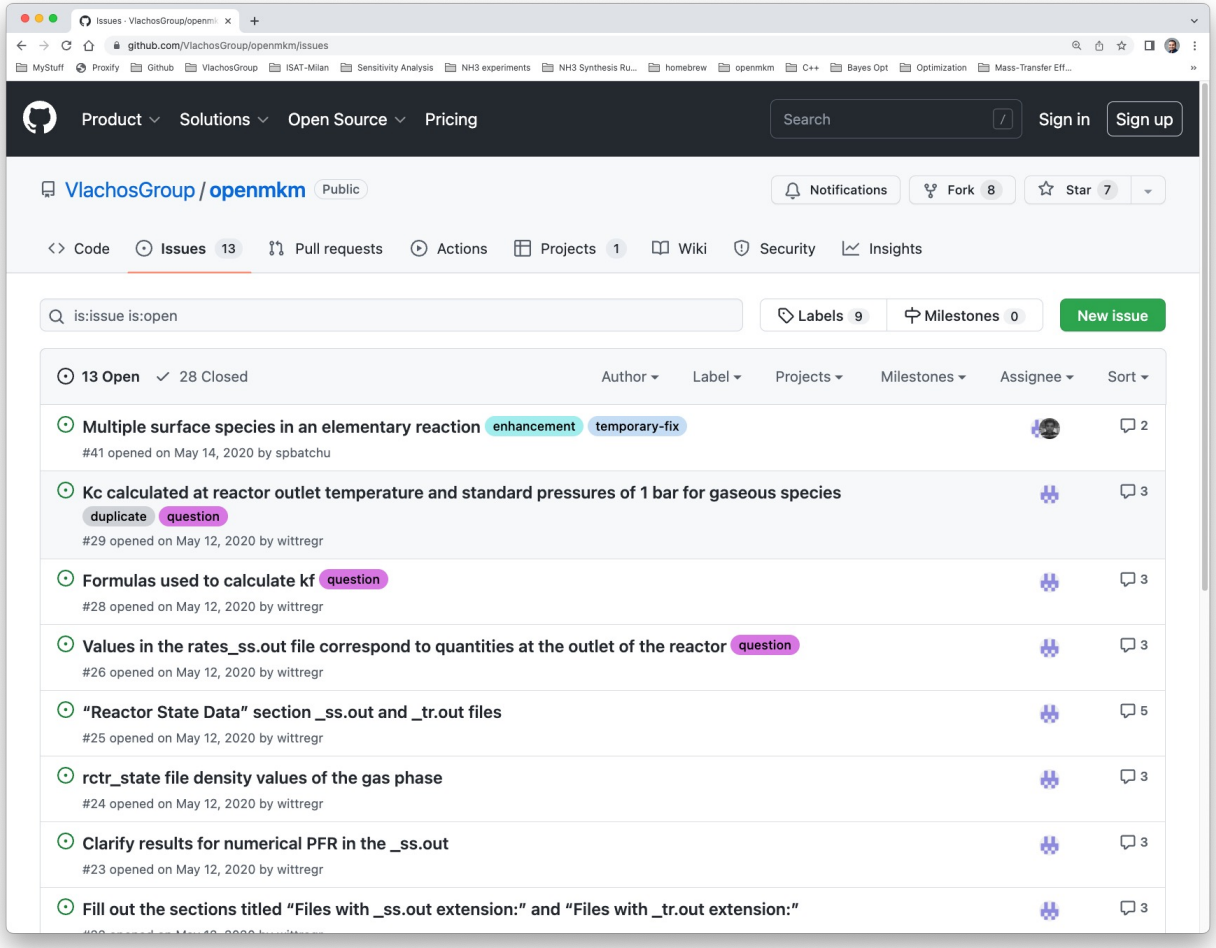
```yaml
step-reactions:
- equation: H2 + 2 RU(S) <=> 2 H(S) + 2 RU(B)
  sticking-species: H2
  Motz-Wise: false
  sticking-coefficient: {A: 0.5, b: 0, Ea: "0.0
  id: r_0007

#-------------------------------------------------------------------------------
# BEPS
#-------------------------------------------------------------------------------
beps:
- id: NH2-H(T)
  slope: 0.608
  intercept: "27.5 kcal/mol"
  direction: cleavage
  cleavage-reactions: ["r_0003"]
#-------------------------------------------------------------------------------
# INTERACTIONS
#-------------------------------------------------------------------------------
interactions:
- species: [N(T), N(T)]
  coverage-threshold: [0, 1]
```

Thermodynamic/kinetic specification

Why use OpenMKM?

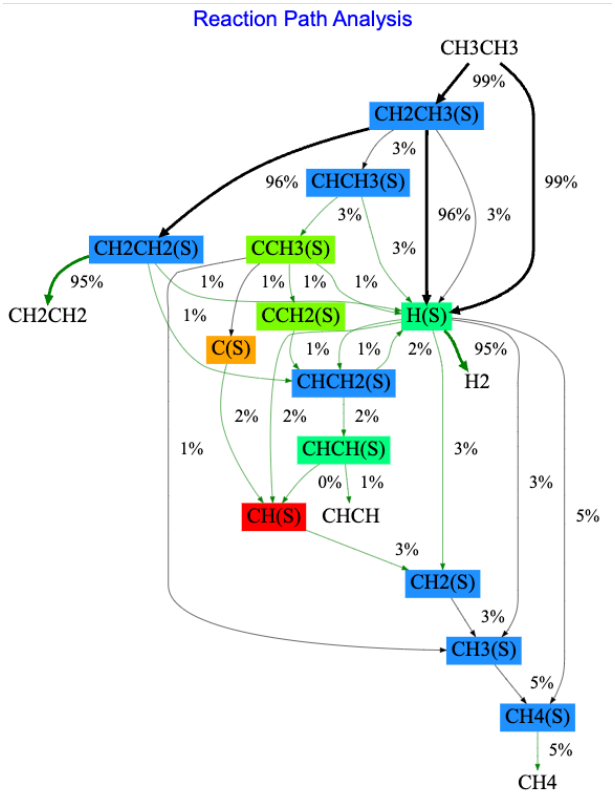
- OpenMKM is super fast! Uses state-of-the art libraries for stability and robustness.
- OpenMKM easy to use with just two inputs; and pMuTT can natively make them.
- **Open-source with active development, so new features can be added, or any bugs can be fixed.**

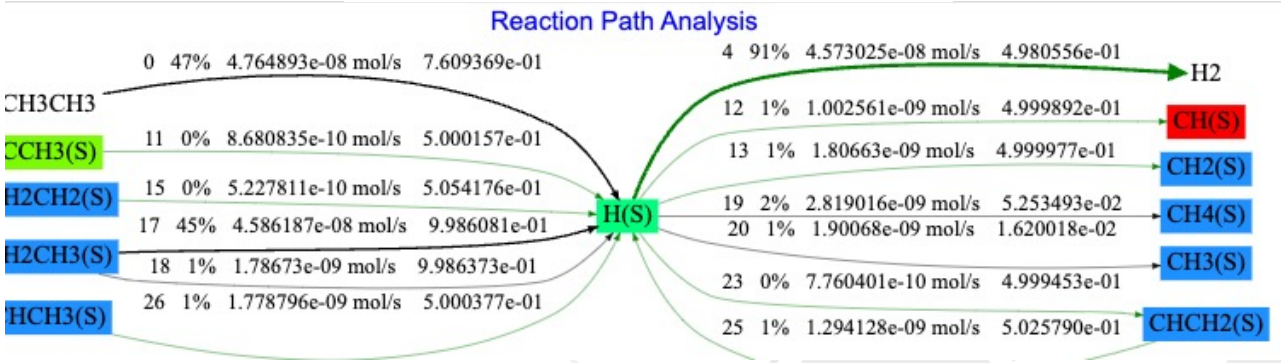


Relatively easier to identify and fix issues and bugs, compared to CHEMKIN, or some other code.

Why use OpenMKM?

- OpenMKM is super fast! Uses state-of-the art libraries for stability and robustness.
- OpenMKM easy to use with just two inputs; and pMuTT can natively make them.
- In-house developers, so new features can be added, or any bugs can be fixed.
- **Works natively with Renview for RPA visualization.**





Motivation: Summary

- OpenMKM is an open-source, fast, modular, C++ application (executable)
- It leverages Cantera for automated generation and simulation of the differential algebraic equations (DAE) and sensitivity analysis (SA)
- No hand-calculations or programming is required to run OpenMKM
- Uses standardized input files (YAML) and (generates) output files (OUT, CSV) which are human-readable
- Inputs can be generated systematically using pMuTT[1] (also part of VLab)
- Uses state-of-the-art numerical libraries (Cantera[2], Sundials[3], Boost C++[4])
- Interfaces to Python (sensitivity analysis), RenView[5] for Path Visualization

[1] https://vlachosgroup.github.io/pMuTT/
[2] https://cantera.org
[3] https://computing.llnl.gov/projects/sundials
[4] https://www.boost.org/
[5] https://github.com/VlachosGroup/renview

OpenMKM: Features Summary

Mechanisms

- Gas phase only
- Surface only
- Gas phase + multiple surfaces

Additional Features

- Support for BEP relations
- Coverage dependent enthalpies
- Parametric Studies
- Sensitivity Analysis
- Reaction Path Analysis
- Parameter Estimation

Supported Reactor Models

- Batch
- CSTR
- PFR
- PFR with T-profile

Temperature Modes

- Isothermal
- Adiabatic
- Heat transfer
- Temperature ramp (For TPD)
- Custom T-Profile (PFR 1-D only)

Demo 1: Install OpenMKM

Install and Run OpenMKM

See Example at: vlab_workshop_2023/OpenMKM/omkm_install_io

VlachosLab
AT THE UNIVERSITY OF DELAWARE

Demo 2: PFR-1D T-Profile

Ethane dehydrogenation MKM

See Example at: vlab_workshop_2023/openMKM/pfr1d_tprofile

VlachosLab
AT THE UNIVERSITY OF DELAWARE

Local Sensitivity Analysis

LSA or Normalized Sensitivity Coefficients [1-3]

$$\mathrm{NSC}_{ij} = \frac{\mathrm{d}\ln R_i}{\mathrm{d}\ln P_j} = \frac{P_i}{R_j}\frac{\mathrm{d}R_j}{\mathrm{d}P_i} \approx \frac{P_i}{R_j}\frac{R_j(P_i+\Delta P_i) - R_j(P_i)}{\Delta P_i}$$

$$S_{kj} = \frac{\partial \ln M_k}{\partial \ln A_j} \approx \frac{A_j \Delta M_k}{M_k \Delta A_j}$$

- Also known as brute-force or one-at-a time local sensitivity analysis (LSA)
- Each model response is evaluated by perturbing each parameter of interest
- Obtained via simple finite differences
- Responses could be species production rates, mass fractions or any response variable
- Rate is perturbed typically via the pre-exponential factor for that reaction

[1] Dionisios G. Vlachos, A Review of Multiscale Analysis: Examples from Systems Biology, Materials Engineering, and Other Fluid–Surface Interacting Systems, Advances in Chemical Engineering, 30, 1-61, **2005**.
[2] de Carvalho et. al., Microkinetic Modeling and Reduced Rate Expression of the Water–Gas Shift Reaction on Nickel, Industrial & Engineering Chemistry Research, 57 (31), 10269-10280, **2018**.

Degree of Rate Control

LSA or NSC

$$\mathrm{NSC}_{ij} = \frac{\mathrm{d}\ln R_i}{\mathrm{d}\ln P_j} = \frac{P_i}{R_j}\frac{\mathrm{d}R_j}{\mathrm{d}P_i} \approx \frac{P_i}{R_j}\frac{R_j(P_i + \Delta P_i) - R_j(P_i)}{\Delta P_i}$$

$$S_{kj} = \frac{\partial \ln M_k}{\partial \ln A_j} \approx \frac{A_j \Delta M_k}{M_k \Delta A_j}$$

Degree of rate control[1]

$$X_{RC,i} = \frac{k_i}{r}\frac{\partial r}{\partial k_i} = \frac{\partial \ln r}{\partial \ln k_i} \approx \frac{\Delta\left(\ln r\right)}{\Delta\left(\ln k_i\right)}$$

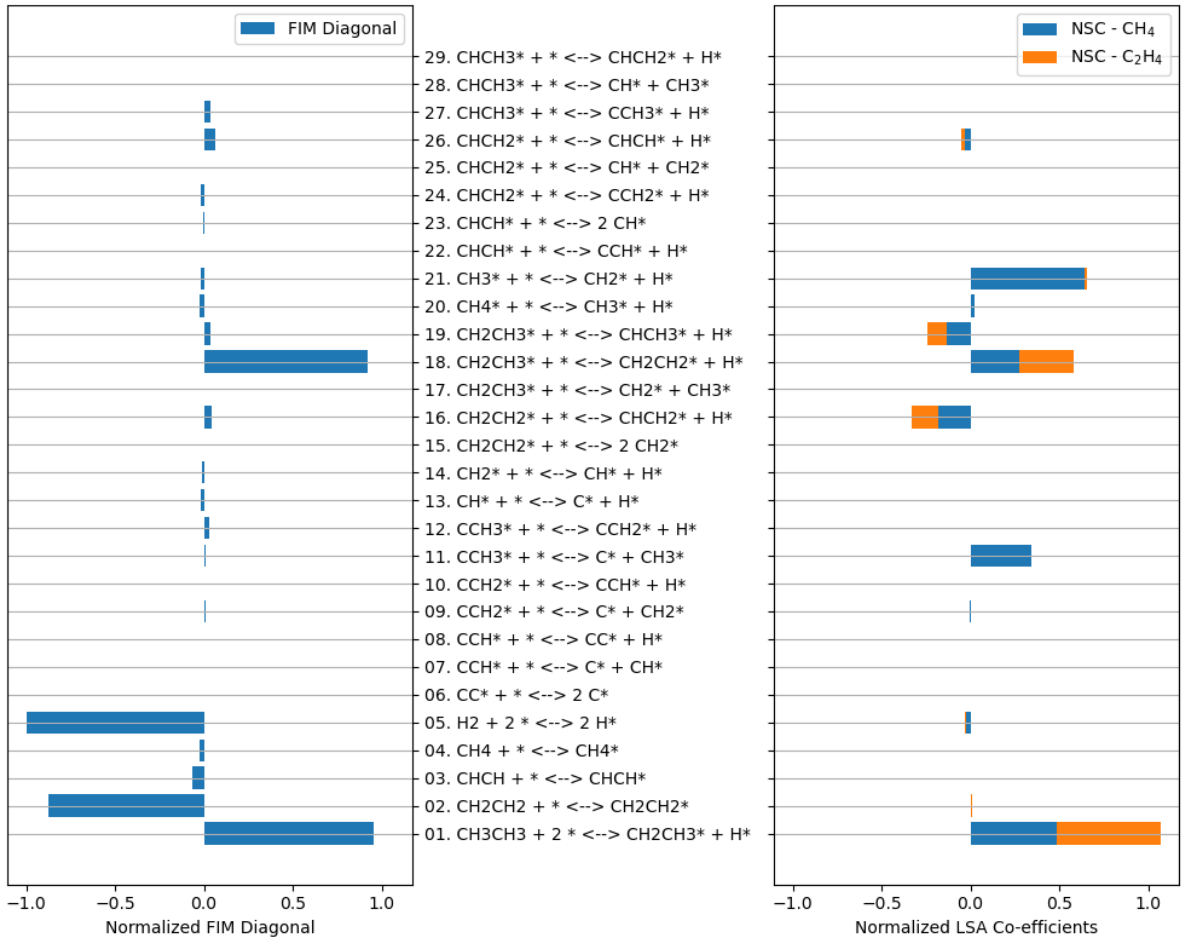
for very small $\Delta\left(\ln k_i\right)$

[1] Stegelmann, C., Andreasen, A. and Campbell, C. T. (2009) ,131, (8077-8082))', *Journal of the American Chemical Society*, 131(37)

Python interface for advanced modeling

- Use OpenMKM's built in LSA/FIM methods
- Simple Python interface wrapper enables
  - **(OAT/Manual) LSA**
  - **Degree of Rate Control**
  - **Global Sensitivity Analysis (SALib)**

1. Note: Manual Python-LSA enables sensitivity of any quantity of interest with respect to:
   1. Any reaction rate coefficients
   2. conc. or cov. of any other species
   3. Temperature, Pressure … (list is endless)



Demo 3: Sensitivity Analysis

LSA/FIM (OpenMKM) ; Manual LSA using Python

See Example at: vlab_workshop_2023/openMKM/lsa_nsc

VlachosLab
AT THE UNIVERSITY OF DELAWARE

Acknowledgements

1. Department of Energy
2. RAPID (AIChE)
3. State of Delaware

Dr. Jeffrey Frey, HPC
Kelly Walker – Logos
Jaynell Keely - Logos