TASK 1: The Viewer GUI

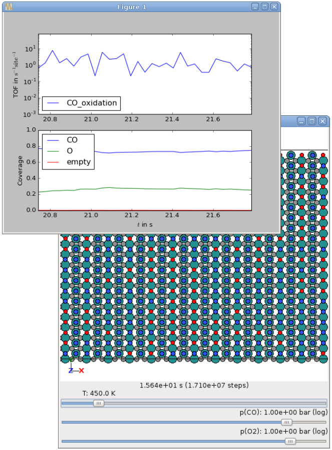
Objectives

Get used to quickly observe model features using kmos' viewer GUI

- Fire up the Viewer GUI with

  `kmos view`

  from the folder containing the model
- Familiarize yourself with the information in the two windows
- Use the sliders to alter the model parameters and observe the effect on reactivity and surface state.
- Follow the instructions in the handout.



The kmos runner API

Client scripts

- Most efficient way to use kmos
- Automatize tasks
- Fine access to all model features
- Easy integration with other python tools (e.g. plotting with matplotlib)

Example

Study dependence of reactivity (turnover frequency, TOF) on the temperature for fixed partial pressures. Plot $\log(TOF)$ vs $1/T$ (Arrhenius plot).

The kmos runner API

First: Initialization

- Import and initialize an instance of `kmos.run.KMC_Model`

```
from kmos.run import KMC_Model
model = KMC_Model(banner = False)
```

Second: General setup

- Set the pressure parameters to the desired values

```
model.parameters.p_COgas = 2.e-1
model.parameters.p_O2gas = 1.e-1
```

Third: Main calculation loop.

- Define the number of relaxation and sample steps (numerical parameters).
- Generate a list of temperature values, and an empty list to store the calculated TOFs.
- Run the model in a loop for each temperature.

```
nrel = 1e7; nsample = 1e7  # numerical parameters
Ts = range(450,650,20)     # 20 values between 450 and 650 K
TOFs = []                  # empty list for output
# Loop over the temperature
for T in Ts:
    model.parameters.T = T   # Set the temperature
    model.do_steps(nrel)     # Relax the system
    # Sample the reactivity
    output = model.get_std_sampled_data(1, nsample, output='dict')
    # Collect output
    TOFs.append(output['CO_oxidation'])
```

Finally: Plot

- Transform the variables
- Plot the results using matplotlib.

```
# Transform the variables
invTs = [1/float(T) for T in Ts]
logTOFs = [math.log(TOF,10.) for TOF in TOFs]

# and plot
import matplotlib.pyplot as plt
plt.plot(invTs, logTOFs, '-o')
plt.xlabel('1/T␣[1/K]')
plt.ylabel('log(TOF)␣/␣events␣(sites␣s)^-1')
plt.savefig('arrhenius.pdf') # Optionally, save plot
plt.show()
```

Alternative

Store the results in a file before plotting

Main calculation loop.

- Open a file for writing.
- Write in results of `get_std_sampled_data` (omit `output='dict'`!)

```
nrel = 1e7; nsample = 1e7           # numerical parameters
Ts = range(450,650,20)              # 20 values between 450 and 650 K
fout = open('arrhenius.dat', 'w')   # open an output file
fout.write(model.get_std_header()) # print the header
# Loop over the temperature
for T in Ts:
    model.parameters.T = T    # Set the desired temperature
    model.do_steps(nrel)      # Relax the system
    # Sample the reactivity and print data to file
    output = model.get_std_sampled_data(1, nsample)
    fout.write(output)
fout.close() # Close output file
```

Alternative

Store the results in a file before plotting

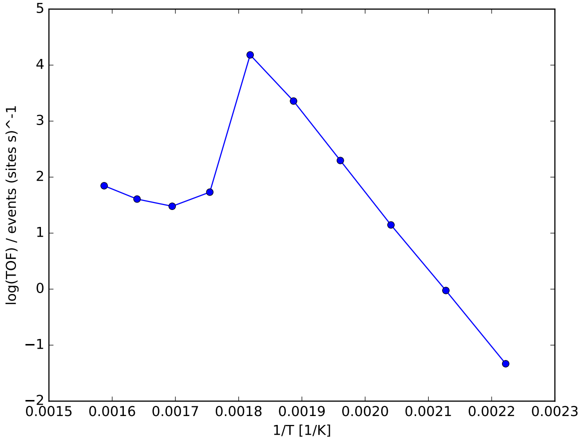
Plot

- With a file, you can use an external application or numpy
- numpy arrays are handy to manipulate

```
# We can read the file with numpy...
import numpy as np
data = np.loadtxt('arrhenius.dat')
iT = 0; iTOF = 3 # columns for each variable
# ... and then plot
import matplotlib.pyplot as plt
# numpy arrays can be transformed much more easily
plt.plot(1/data[:,iT], np.log10(data[:,iTOF]), '-o')
plt.xlabel('1/T␣[1/K]')
plt.ylabel('log(TOF)␣/␣events␣(sites␣s)^-1')
# plt.savefig('arrhenius.pdf') # Optionally, save plot
plt.show()
```

The kmos runner API

Result

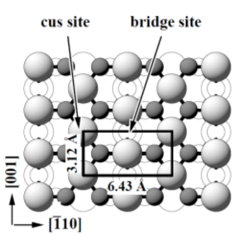


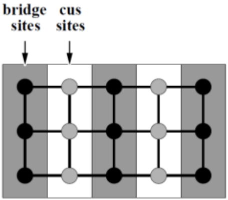
TASK 2: TOF and Coverages vs. pressure

Objectives

Write a client script to run a model and inspect it's behavior.

- Write a kmos client scripts that runs the model at fixed $T = 600\,\mathrm{K}$ and $p_{O_2} = 10^{-1}$ bar for 10 values of 10^{-2} bar $< p_{CO} < 10^{1}$ bar (important: distribute these values in a log-scale).
- As in the previous case, collect the TOF results for each value of p_{CO}.
- Also collect CO and O coverage for each site type (bridge and cus).
- Make a TOF and a coverage plot

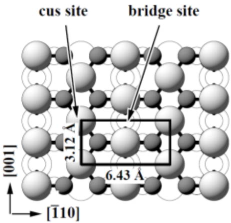


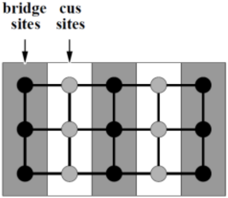


TASK 2: TOF and Coverages vs. pressure

Hints

- Important output for this exercise is labeled by `'CO_oxidation'`, `'CO_ruo2_bridge'`, `'CO_ruo2_cus'`, `'O_ruo2_bridge'` and `'O_ruo2_cus'`
- If using an output file: Default columns for these output parameters are 3, 4, 5, 6 and 7. `'p_COgas'` is in column 1. (First column is nr. 0).
- You can use `numpy.logspace` to generate points equidistant in a log-scale





TASK 2: TOF and Coverages vs. pressure