CatINT Documentation

Release 1.0

Stefan Ringe

May 11, 2021

Contents

CHAPTER 1

Installation

1.1 Installing CatINT

CatINT is a program package which combines mass transport simulations with micro- kinetic modeling. Installing CatINT requires:

- python 2.5 or greater
- numpy
- scipy
- matplotlib
- tables
- logging

After installing all dependencies, you can install CatINT into a folder $CATINT via:

```
$ mkdir $CATINT
$ cd $CATINT
$ git clone git@github.com:sringe/CatINT.git
```

As usual update your $PYTHONPATH variable with the $CATINT folder location and verify that everything worked by importing the central transport module of CatINT:

```
$ python >>>from catint import transport
```

1.2 Installing CatMAP

For installing CatMAP, please see https://catmap.readthedocs.io/en/latest/installation.html.

For running CatMAP with CatINT, it is required to download the version from https://github.com/sringe/catmap-1.git, which enables the Frumkin voltage corrections and surface charge corrections of energetics which are not yet contained in the main repository.

1.3 Installing COMSOL

For installing COMSOL ®, please see www.comsol.de. CatiNT writes a java input file for COMSOL ® which is then compiled and executed. The COMSOL ® executable location is automatically searched for. However, if this fails, or the executable is out of reach for the `locate` linux command, specify the path of the executable manually in the `tp.comsol_args['bin_path']` variable (where `tp` is an instance of the `Transport` class).

Documentation (this wiki) is located in the catmap/docs folder, and is available online at http://catint.readthedocs.org/.

CHAPTER 2

General Usage and Information

CatINT is a program package which combines mass transport simulations with micro- kinetic modeling. Mean-field micro-kinetics can be used via

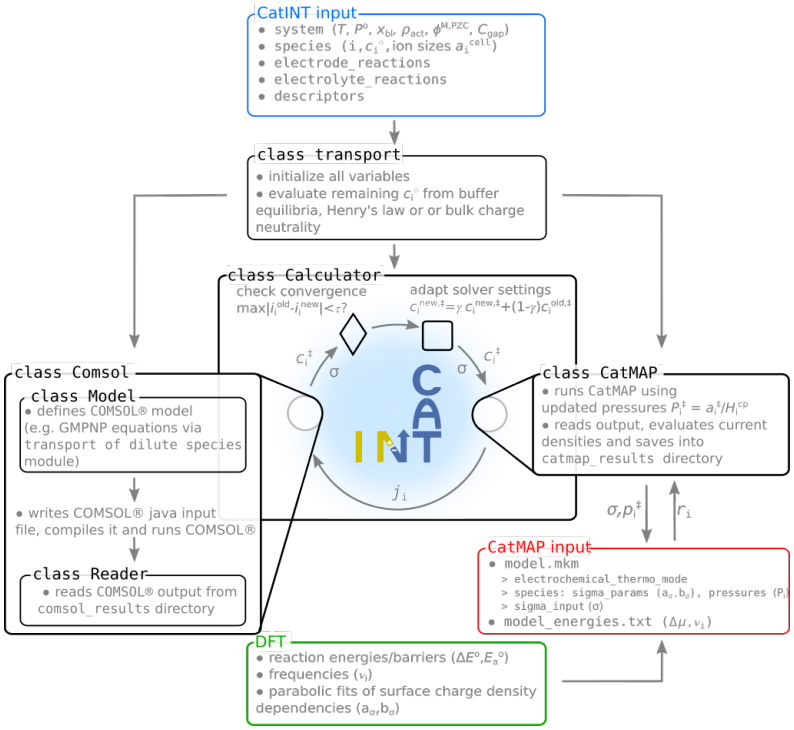
- analytic rate-equations (simple models)
- CatMAP (more involved models)

Mass transport (currently only 1D) is implemented via

- simple finite difference solver of the Poisson-Nernst Planck equations (simple models)
- finite-element program package COMSOL (more involved models, very robust)

The general procedure is shown the following figure:

Documentation (this wiki) is located in the catint/docs folder, and is available online at http://catint.readthedocs.org/.


CatINT input
• system (T, P°, x_bl, ρ_act, φ^M,PZC, C_gap)
• species (i, c_i°, ion sizes a_i^cell)
• electrode_reactions
• electrolyte_reactions
• descriptors
class transport
• initialize all variables
• evaluate remaining c_i° from buffer equilibria, Henry's law or or bulk charge neutrality
class Calculator
check convergence max|i_i^old-i_i^new|<τ?
adapt solver settings
c_i^new,‡=γ c_i^new,‡+(1-γ)c_i^old,‡
c_i‡
σ
j_i
CAT
INT
class Comsol
class Model
• defines COMSOL® model (e.g. GMPNP equations via transport of dilute species module)
• writes COMSOL® java input file, compiles it and runs COMSOL®
class Reader
• reads COMSOL® output from comsol_results directory
class CatMAP
• runs CatMAP using updated pressures P_i‡ = a_i‡/H_i^cp
• reads output, evaluates current densities and saves into catmap_results directory
σ,p_i‡
r_i
CatMAP input
• model.mkm
> electrochemical_thermo_mode
> species: sigma_params (a_σ,b_σ), pressures (P_i)
> sigma_input (σ)
• model_energies.txt (Δμ, ν_i)
DFT
• reaction energies/barriers (ΔE°,E_a°)
• frequencies (ν_i)
• parabolic fits of surface charge density dependencies (a_σ,b_σ)

CHAPTER 3

Tutorials

3.1 Using a CatMAP defined micro-kinetic model: CO2 reduction at polycrystalline Au

3.1.1 Creation of the CatINT input file

Import

We start with the creation of the CatINT python input file which is contained in $CATINT/examples/02_CO2R_Au_CatMAP/run.py. We import the main transport class and the calculator:

```
import numpy as np
from catint.transport import Transport
from catint.calculator import Calculator
from catint.units import unit_NA
```

CatINT comes with it's own units package.

System Settings

First, we define the system reaction conditions:

```
pH = 6.8
system=\
    {
    #ENVIRONMENTAL CONDITIONS
    'temperature':  298,          #K
    'pressure':     1.013,        #bar
    'bulk_pH': pH,
    #MASS TRANSPORT
    'boundary thickness': 8.E-05, #m
```

(continues on next page)

(continued from previous page)

```
    #ELECTROSTATICS
    'epsilon': 78.36,            #eps_0
    'migration': True,
    #REACTIONS
    'electrode reactions': True,
    'electrolyte reactions': True,
    #CHARGING
    'charging_scheme':'comsol',
    'phiM':-0.5,
    'phiPZC': 0.16,              #V vs. SHE
    'Stern capacitance': 20.,    #micro F/cm2
    #KINETICS
    'potential drop':'Stern',
    'active site density': 9.61e-05/unit_NA*(1e10)**2, #mol sites/m^2
    #INITIALIZATION
   # 'init_folder':init_folder,
    }
```

`temperature` and `pressure` are important for the finite temperature free energy corrections that are applied in `CatMAP`, but the temperature also enters the mass transport equations. The `pH` should be adjusted to the reaction conditions. *boundary thickness'* refers to the boundary thickness which defines the cell dimensions. `epsilon` is the bulk dielectric constant that enters the mass transport equations. `migration` turns the migrational motion of species on or off. Further, electrode and electrolyte reactions can be turned on or off. The charging of the surface is controlled via the `charging_scheme` tag. Currently, `CatINT` supports different methods to define a surface charge density - potential relation $\sigma(\phi^{\mathrm{M}})$ which is used to evaluate potential-dependent free energies:

- `'comsol'`: Get the relation from `COMSOL` directly via the electrostatics defined there.
- **`'input'`: Define a relation manually. This enables us to take the charging relation directly from the `CatMAP` input file. T**
  - `sigma_input=['CH',20.]`. Define surface charge density via $\sigma = C_{\mathrm{H}}(\phi^{\mathrm{M}} - \phi^{\mathrm{PZC}})$, where C_{H} is a capacitance.
  - `sigma_input='file.txt'`: Define surface charge density from $\sigma(\phi^{\mathrm{M}})$ relation given in file name. The discrete data in the file is interpolated

For the electrostatics within the mass transport (PNP equations), the *Stern capacitance* (also called gap capacitance) is needed to define the Robin boundary condition. The same is valid for `phiPZC`, $\phi^{\mathrm{M,PZC}}$ which is given on an SHE scale. `phiM` is the starting potential for the calculation which is usually overwritten when defining a descriptor range (see below). The *potential drop* defines how to calculate the driving electrochemical potential in `CatMAP`, i.e. if a Frumkin correction is considered or not. If `'potential_drop':'Stern'` is selected a Frumkin correction is applied, so that the driving potential drop is $\phi^{\mathrm{M}} - \phi^{\ddagger}$ with $\phi^{\ddagger}$ is the potential at the reaction plane.

Finally `CatINT` supports to initialize a run form a previous one by using the `init_folder` tag.

Electrolyte Reactions

We define electrolyte reactions which should be included in the mass transport. Currently `CatINT` supports multiple buffer reactions which are defined in the `catint/data.py` file. Here we include bicarbonate buffer reactions both using water and protons as donors (acidic and alkaline conditions) as well as the water self dissociation equilibria

```
electrolyte_reactions=['bicarbonate-base','water-diss',{'additional_cell_reactions':
↪'bicarbonate-acid'}]
```

`CatINT` evaluates concentrations at the boundary layer based on these buffer equilibria reactions. Since the acidic buffer reactions are already contained in the combination of alkaline and water dissociation, they are included as

additional_cell_reactions, meaning that they are active during the mass transport simulation, but not used for initializing the concentrations.

Electrode Reactions

Now it is time to think about the reactions at the electrode, the electrochemical reactions. In our case, we consider the reduction of CO2 with water as a proton donor, which is defined as:

```
electrode_reactions={
    'CO': {'reaction': 'CO2 + H2O + 2 e- -> CO + 2 OH-'},
}
```

Species Definitions

After defining the reactions, we need to also think about all species that our system, we define them as a dictionary:

```
species=\
    {
    'K+':               {'bulk_concentration':   'charge_neutrality',
                         'MPB_radius':           2*3.5e-10},
    'CO2':              {'bulk_concentration':   'Henry'},
    'OH-':              {'bulk_concentration':   10**(pH-14.)*1000.0}, #mol/m^3
    'CO':               {'bulk_concentration':   0.0}
    }
```

All species that are part of the electrolyte reactions are already automatically added to the species dictionary in CatINT. All other species have to be added here. Charges are automatically assigned according to the species name. We add potassium cations which should neutralize all anions so that the system is charge neutral in the bulk solution (boundary layer). We chose a MPB_radius of 3.5 Angstrom which is important since the negative electrode potental dramatically increases the potassium concentrations. We then define the CO2 concentration at the boundary layer to be given by Henry's law which will use the pressure defined in system settings and the Henry constant in data/henry_constants.txt to evaluate the equilibrium CO2 concentration. The buffer component concentrations are now evaluated using the equilibrium buffer equations and must not be specified. It is also possible though to specify a buffer concentration and let CatINT calculate the CO2 concentration via the buffer equations. Finally, we set the concentration of hydroxide anions according to the pH (proton concentration are automatically evaluated using the water dissociation equilibrium). All concentrations are given in $\mathrm{mol}/\mathrm{m}^3$.

Descriptors

In a common application, CatINT calculations should be run for a specified parameter or descriptor range. In this example, we want to simulate a polarization curve, our descriptor is therefore the electrode potential ϕ^{M}, we define it like this:

```
phimin=-0.5
phimax=-2.0
dphi=0.01
descriptors={'phiM':list(np.linspace(phimin,phimax,-(phimax-phimin)/dphi+1))}
```

Currently CatINT supports only the potential as descriptor, others could be implemented, if needed. The CatINT calculator iterates over the descriptor list and solves the coupled mass transport – microkinetics model at each potential.

COMSOL arguments

There are a couple of predefined COMSOL arguments which are saved in the `tp.comsol_args` dictionary (suppose that `tp` is the transport instance that we create in the end of this tutorial page). We can define COMSOL variables and parse them to CatINT via the `comsol_args` tag:

```
comsol_args={}
#parameter
comsol_args['parameter']={}
comsol_args['parameter']['grid_factor']=[str(100),'Grid factor']
comsol_args['parameter']['grid_factor_domain']=[str(100),'Grid factor for domain']
comsol_args['parameter']['grid_factor_bound']=[str(200),'Grid factor for boundary']
#solver_settings
comsol_args['solver_settings']={}
comsol_args['solver_settings']['direct']={}
comsol_args['solver_settings']['direct']['nliniterrefine']=True
comsol_args['solver_settings']['ramp']={}
comsol_args['solver_settings']['ramp']['names']=['PZC','CS']
comsol_args['solver_settings']['ramp']['dramp']=0.01
#par_method
comsol_args['par_method']='internal'

#SOLVER SEQUENCE
#comsol_args['solver_settings']['solver_sequence']='tds_elstat'
#OTHER PARAMETER
#comsol_args['parameter']['RF']=[1,'Roughness Factor']
```

This is in particular useful for modifying numerical solver settings. In our case, we first define a `grid_factor` which tells `COMSOL` about the minimal finite element mesh width. A higher factor means a finer mesh and the mesh can be defined for the domain and boundary separately. Parameter definitions always a list of two entries, the value parsed as a string and the name or description inside COMSOL. Some more specific numerical parameters can be edited and changed here to help convergence. In particular, the `'ramp'` flag enables to slowly ramp non-linearities in the equations, in our case it slowly ramps up the PZC and the Helmholtz/Stern/gap capacitance which can be useful if the systems has a PZC far from the initial potential to be evaluated. A flux ramping is always applied and controlled by the *'dramp'* flag which defines the interval in which the fluxes are ramped from 0 to 100 %. Additional possible settings involve the definition of solver sequences to improve convergence (e.g. first solving electrostatics only, then the coupled mass transport/electrostatic problem). Also, it is possible to define a roughness factor that multiplies all fluxes by a constant.

Inside `CatINT`, a couple of `COMSOL` variables are assigned by default. The iterations over `tp.descriptors` are performed inside `CatINT` and `COMSOL` is compiled and run for each descriptor value. This behavior is defined via the `COMSOL` key:

```
comsol_args['desc_method'] = 'external'
```

Inside `COMSOL`, we have the possibility to sweep over a particular parameter space to enable convergence. A common way to do this, is to ramp up the non-linearities in the equations as the flux of the species. This is the default in `CatINT`:

```
comsol_args['par_name'] = 'flux_factor'
comsol_args['par_values'] = 'range(0,'+str(comsol_args['solver_settings']['ramp'][
↪'dramp'])+',1)'
comsol_args['par_method'] = 'internal'
```

The range can be modified by the `dramp` key as discussed before. The `par_method` key indicates the way that `COMSOL` should treat the parametric sweep: `'internal'` uses an auxiliary parameter sweep, while `'external'` uses a regular parameter sweep. Although both should do in principle the same, there fine differences between both

methods inside COMSOL, and generally the 'internal' sweep seems to be more stable.

CatMAP arguments

Some additional arguments can be parsed to the CatMAP calculator:

```
catmap_args={}
#CATMAP DESCRIPTOR RAMPING
catmap_args['desc_method']='automatic'
#catmap_args['min_desc']=0.0
catmap_args['min_desc_delta']=0.2
catmap_args['max_desc_delta']=0.2
#INTERACTIONS
catmap_args['n_inter']='automatic'
```

In a regular CatMAP-COMSOL iteration loop, a single CatMAP calculation is required at the descriptor value of choice. This is referred to as desc_method='single_point'. Sometimes, however, some potential values are hard to converge and it is better to provide CatMAP with a range of potentials. CatMAP will then try to find stable starting points and solve for all potentials and CatINT selects the potential that was actually needed. This happens if we choose desc_method='automatic'. For this case, min_desc specifies the minimum descriptor value in the new created list of descriptors. Alternatively, min/max_desc_delta can be used to create a new list of descriptors around the current descriptor value. The descriptor range can be also taken from the CatMAP input file (desc_method='from_input').

Flux definition

Now it is time to define how fluxes are evaluated within the CatINT model. In our case, we will use CatMAP to define fluxes but multiple options are available (cf. *Defining fluxes*).

```
species['CO']['flux']='catmap' #CO production rate
species['CO2']['flux']='catmap' #CO2 consumption rate
```

Defining transport instance and assigning calculator

We are now ready to define the transport instance to which we parse all the previously defined dictionaries:

```
nx=200
tp=Transport(
    species=species,
    electrode_reactions=electrode_reactions,
    electrolyte_reactions=electrolyte_reactions,
    system=system,
    catmap_args=catmap_args,
    comsol_args=comsol_args,
    model_name='CO2R',
    descriptors=descriptors,
    nx=nx)
```

nx is hereby the starting number of finite elements. By using the COMSOL calculator this is usually relevant, because the 'grid_factor_?' keys will define the mesh. However, tp.nx will be updated and thus the final size of the mesh within COMSOL can be accessed via this.

We now choose a calculator for the transport instance, in our case the COMSOL calculator and then assign the transport instance to a calculator object.

```
tp.set_calculator('comsol')
c=Calculator(transport=tp,tau_scf=0.008,ntout=1,dt=1e-1,tmax=10,mix_scf=0.02)
```

Now, we can run the calculation:

```
c.run()
```

Mass transport-free CatMAP simulation

We can also use CatINT just to run CatMAP. This can be useful for easily comparing non-mass transport corrected and mass transport corrected results, or to just use the analysis tools of CatINT for CatMAP (cf. :ref:analysis). In order to do this, instead of assigning the tp instance to a Calculator object, we define a CatMAP instance and then run CatMAP for each potential:

```
cm=CatMAP(transport=tp,model_name='CO2R')
for pot in descriptors['phiM']:
    tp.system['phiM']=pot
    tp.descriptors['phiM']=[pot]
    tp.system['use_activities']=False
    cm.run()
```

Mass transport extrapolation

Sometimes, COMSOL does not converge any more, but we have sufficient data that we think we can try to extrapolate all species concentrations at the reaction plane to a different potential region. This is done by polynomial functions, but these can be in principle specified by the user. We first import the extrapolate function:

```
from tools.extrapolate_surface_conc import extrapolate
```

We then start by running a mass-transport corrected CatINT simulations for the region of potentials which converges. Then we run the input file again until the assignment of a calculator:

```
tp.set_calculator('comsol')
extra=extrapolate(tp=tp,extrapol_folder='CO2R_results_to_extrapolate')
extra.plot()
```

This first initializes the tp instance as before, but that uses the old CatINT results folder which we named CO2R_results_to_extrapolate here to extrapolate concentrations of species at the reaction plane. The plot function enables to visualize the extrapolation and play around with the extrapolation functions.

If one is satisfied, one can remove the two last lines and instead define a CatMAP only calculator and use the extrapolated surface concentrations at each potential for the CatMAP calculation:

```
tp.set_calculator('comsol')
cm=CatMAP(transport=tp,model_name='CO2R')
for pot in descriptors['phiM']:
   for sp in tp.species:
      tp.species[sp]['surface_concentration']=10**extra.extrapol_func[sp](pot)
   tp.system['potential']=[extra.extrapol_func['voltage_diff_drop'](pot)]
   tp.system['surface_pH']=extra.extrapol_func['surface_pH'](pot)
cm.run()
```

It is important to also extrapolate the voltage_diff_drop, the Frumkin correction for the driving force as well as the pH at the reaction plane which both enter the CatMAP kinetics.

3.1.2 Running the model

Running the model, requires also to set-up a CatMAP mkm file and txt file containing all energies, see https://catmap.readthedocs.io/en/latest/ for details. Having created the CatINT input file run.py and the CatMAP files co2r.mkm and co2r.txt, we can simply run CatINT via

```
python run.py
```

This will create a new folder preceding the name given in the definition of the transport instance and write all output files into this folder.

catmap_input

Contains subfolders desc_? for each descriptor value. This folder contains all input files and model definitions (also given in terms of free energy diagrams).

catmap_output

Contains subfolders desc_? for each descriptor value. This folder contains all output files of CatMAP, as coverages cov_[species].tsv, reaction rates j_[species].tsv, elementary step reaction rates jelem_?.tsv and degree of rate control rc_[product]_[species].tsv

comsol_results

The folders comsol_results_id000_[index of descriptor 1]_[index of descriptor 2] contain all COMSOL results for a specific descriptor pair (In our case only a single descriptor, the potential, is used). The results outputted by COMSOL are defined in the transport.comsol_args['outputs'] dictionary. The default outputs that are always generated are:

```
comsol_args['outputs']=['concentrations','electrostatics','electrode_flux','rho_charge
↪']
```

These refer to species concentrations, electrostatics (potential and electric field), electrode fluxes and the charge density. Usually these outputs are enough, but if more are needed they can be easily asked for using the comsol_args['outputs'] key in the CatINT input file and extending the catint/comsol_model.py if needed for the required output.

The folder also contains the compile and run log files for debugging purposes as well as copies of the COMSOL input file (mph file) and .java input and compiled class .class file used for the particular calculations. Backups of the .java files with timestamps for debugging purpose are also saved in the results folder.

Convergence problems

Convergence problems can happen in particular at high potentials. We found that the reason is often the very small CO2 concentration at the electrode which results in negative CO2 concentrations due to numerical issues. CatINT tries to solve this issue by disregarding the CatINT-COMSOL iteration step and using the previous CO2 concentration (but all updated species fluxes and surface concentrations). This mostly helps, but can also fail at higher potentials. Other possibilites of influencing convergence are a change of the grid_factor if COMSOL has problems converging. A number of default steps are implemented in CatINT to try to get convergence, sometimes it even helps to restart the same COMSOL calculation due to slight numerical differences. If all does not help, CatINT stops and raises and error message. If that is the case, we suggest to also try a finer potential descriptor range. The latter is important, since CatINT initializes the next descriptor step with the species surface concentrations of the previous step which yields

a better initialization and avoid very bad first iteration steps which can break the convergence. Also the use of solver sequences might be possible to increase convergence.

3.1.3 Analyzing the results

Polarization curve and coverages

We first analyze the results, by using the `$CATINT/tools/plotting_catmap.py` utility. We call

```
python $CATINT/tools/plotting_catmap.py --file CO2R_results --scale SHE --expdata\
    --product CO --exp_add_pH 7.2 --system pc-Au --fit_tafel --exp_colormode dataset
```

The command plots the current density and coverages of the calculation. Experimental data can be added, if placed into the `$CATINT/data/CO2R/CSV` folder in the form of CSV files folder in the form of CSV files (here we use the digitized and original data of various references which is not publically available). In case experimental data has been placed into the folder, the `$CATINT/catint/experimental.py` file also needs to be modified, so that this data is considered. Then the data can be plotted by invoking the above command using also the `--expdata` keyword. The experimental data is filtered with respect to the simulated pH. Data for different pH values can also be plotted using the `--exp_add_pH` argument. `--system` specifies the system for which experimental data will be searched for, `--product` filters the product for which experimental partial current densites should be plotted. `--exp_colormode` defines how to color the experimental data curves, the possible options are `'dataset'` (color with respect to data set/reference) and `'species'` (color with respect to species). `--fit_tafel` fits a Tafel line to experimental points selected in the `skip_dict` dictionary in the `$CATINT/catint/experimental.py` file. In case no experimental data is available, just remove all the keywords referring to the experimental data.

The resulting figure (including experimental data) is:

As seen from the figure, all experimental curves are pretty close to each other and also the Tafel slopes are close. Also the theory predicts the experimental curves reasonably well.

Coverages are shown in the bottom window and indicate no major surface coverage of any species.

In order to analyze the rate-limiting step, we plot the rate control analysis using `--ratecontrol` and remove the experimental data for clarity:

```
python $CATINT/tools/plotting_catmap.py --file CO2R_results --scale SHE\
    --product CO --system pc-Au --fit_tafel --ratecontrol
```

The resulting figure is:

As seen from the figure, the *COOH to *CO transition state limits the overall conversion at low overpotentials, why CO2 adsorption limits over the remaining part.

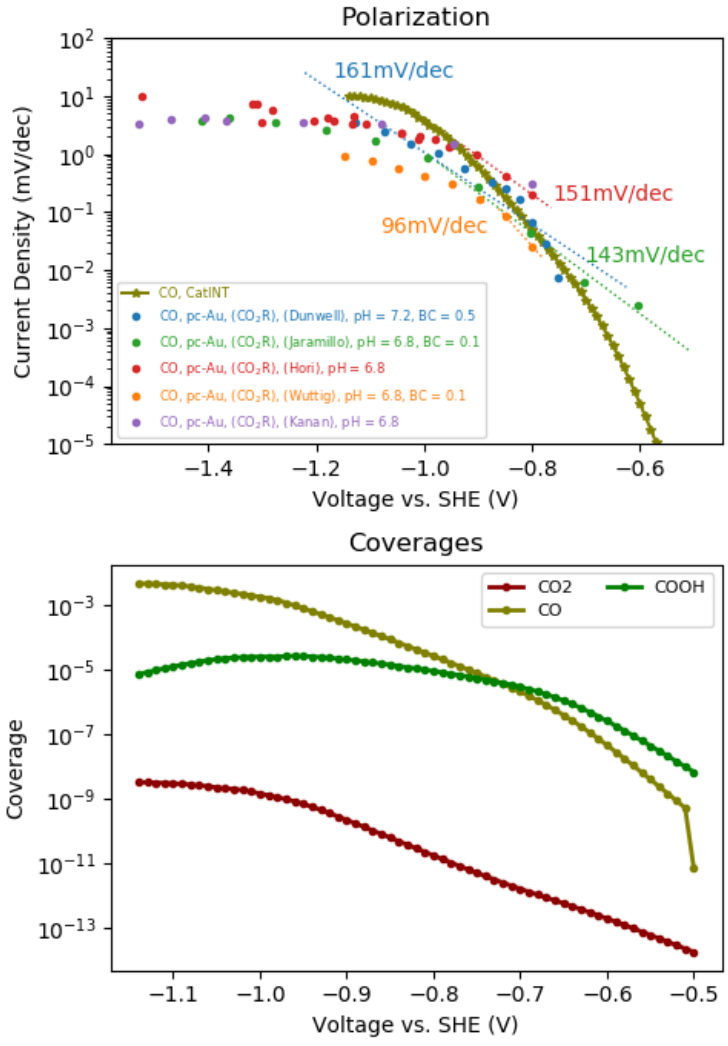
Mass transport properties

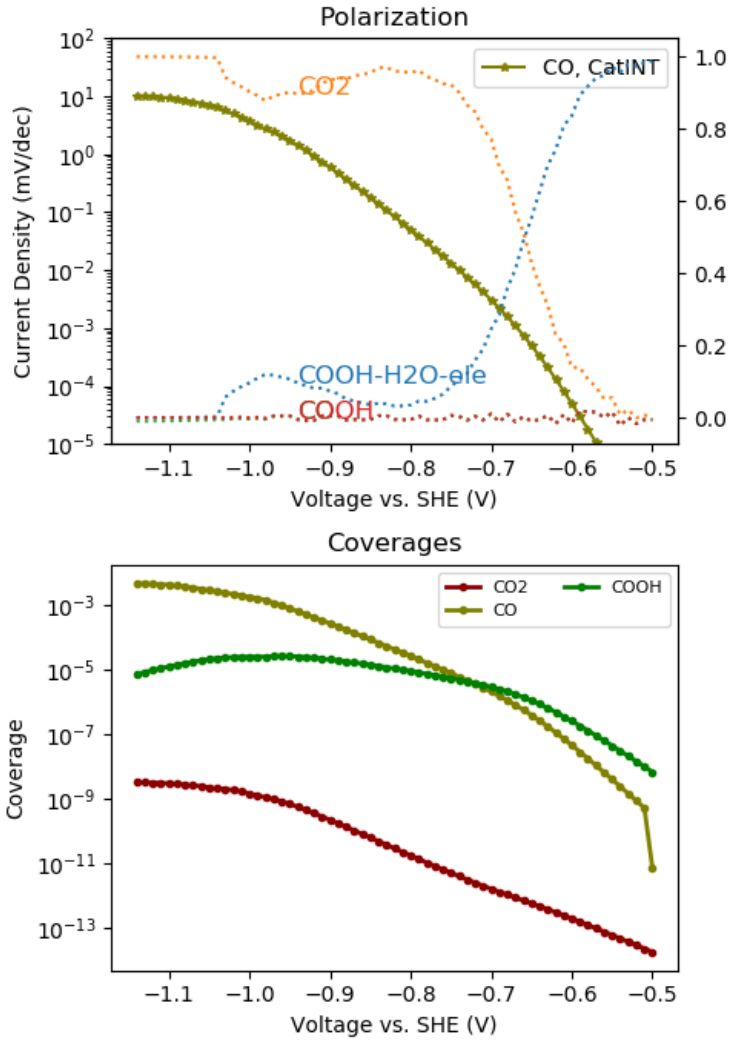
Transport properties are plotted using the `$CATINT/tools/plotting_catint.py` utility. Running it as

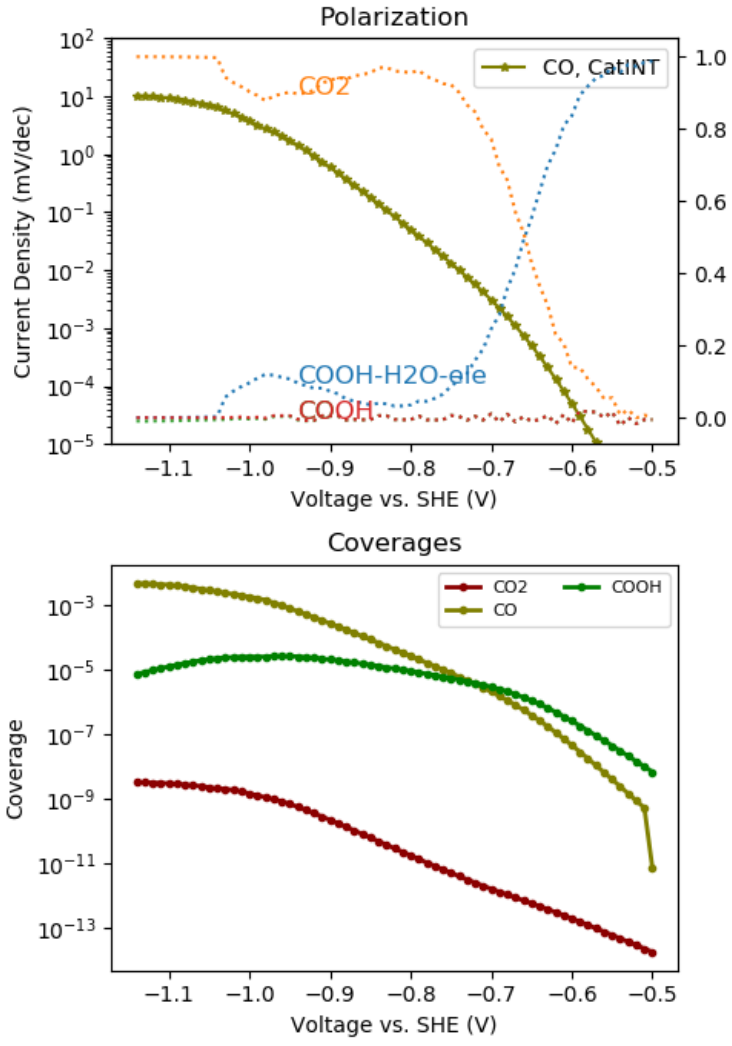
```
python $CATINT/tools/plotting_catint.py --file CO2R_results --prop concentration␣
↪potential surface_pH activity --desc -0.9
```

`--prop` selects the property to plot (see *Analysis tools* for all available properties), `--desc` selects a descriptor value (potential).

The resulting figure shows the species concentrations, potential and activities as a function of x at -0.9 V vs. SHE, as well as the pH as a function of potential:


Polarization
161mV/dec
151mV/dec
96mV/dec
143mV/dec
Current Density (mV/dec)
Voltage vs. SHE (V)
CO, CatINT
CO, pc-Au, (CO2R), (Dunwell), pH = 7.2, BC = 0.5
CO, pc-Au, (CO2R), (Jaramillo), pH = 6.8, BC = 0.1
CO, pc-Au, (CO2R), (Hori), pH = 6.8
CO, pc-Au, (CO2R), (Wuttig), pH = 6.8, BC = 0.1
CO, pc-Au, (CO2R), (Kanan), pH = 6.8
Coverages
CO2
COOH
CO
Coverage
Voltage vs. SHE (V)


Polarization
CO, CatINT
CO2
COOH-H2O-ele
COOH
Current Density (mV/dec)
Voltage vs. SHE (V)
Coverages
CO2
COOH
CO
Coverage
Voltage vs. SHE (V)


Polarization
CO, CatINT
CO2
COOH-H2O-ele
COOH
Current Density (mV/dec)
Voltage vs. SHE (V)
Coverages
CO2
COOH
CO
Coverage
Voltage vs. SHE (V)

3.2 Using an analytic rate-equation model: CO2 reduction at polycrystalline Cu

3.2.1 CO2 reduction at polycrystalline Cu

Creation of the CatINT input file

COMSOL arguments

COMSOL has to be run only once instead of iteratively as in the case of the CatMAP calculator. In that case, it is possible to let COMSOL do the iterations over the descriptor range, by setting

```
comsol_args['desc_method'] = 'internal'
```

Since the potential is correlated with the flux of species, it serves as a kind of non-linearity ramping and we do not need the 'flux_factor' as an additional ramping. We thus define

```
comsol_args['par_name'] = 'phiM'
comsol_args['par_method'] = 'internal-reinit'
```

Note, that 'par_name' must be a name from the transport.descriptors list and the values are assigned automatically, if not changed here. We can perform the iterations by using the 'internal-reinit' (solutions are reinitialized at each potential – default) or 'internal-cont' (solutions are initialized by the previous potential) methods.

CHAPTER 4

Topics

4.1 Analysis tools

In general, `CatINT` includes two main tools for analyzing the results of the simulation:

- `$CATINT/tools/plotting_catint.py`
- `$CATINT/tools/plotting_catmap.py`

4.1.1 `plotting_catmap.py`

`$CATINT/tools/plotting_catmap.py` reads the results from the `catmap_output` folder and plots the current densities, coverages and degree of rate control. It supports also the comparison with experimental data, that is contained in the `$CATINT/data/[reaction]/CSV` folder.

4.1.2 `plotting_catint.py`

`$CATINT/tools/plotting_catint.py` reads the results from the `comsol_results_?` folder and plots all desired variables. Currently, `CatINT` supports a range of variables to be plotted. In general, for a variable named `surface_[variable]`, the variable is plotted at the reaction plane at a potential specified via `--desc` (maps to closest actually calculated potential). In all other cases, the variable is just plotted as a function of space (x):

- **properties (`--prop`) supporting reaction plane (prepend `surface_`) and spatial plot:**
  - `concentration`: Concentrations of all species
  - `activity`: Activities of all species
  - `activity_coefficient`: Activity coefficients of all species
  - `efield`: Electric field
  - `potential`: Electrostatic potential
  - `charge_density`: Charge density

  - `pH`: pH calculated using the proton activity (if in species list, otherwise the hydroxide activity)

- **Other supported properties are:**
  - `electrode_current_density`: Partial current density of all products, should give the same results as the plotting of the `catmap_output` data using the `$CATINT/tools/plotting_catmap.py` script
  - `pKw`: The water self-dissociation product at the reaction plane as a function of potential
  - `concentration_at_x`: Plot the concentrations of species at a specified distance x (specified using ``--xfixed`)

Also remember to set the potential scale correctly by specifying `--scale RHE/SHE`.

4.2 Defining fluxes

4.2.1 Fixed species fluxes

We start by introducing into the definition of fluxes in CatINT as constants within descriptor space. We look at HER and CO2R to CO in alkaline conditions. At the electrode, we define the two reactions. The dictionary key must be the product name as also appearing in the species dictionary:

```
electrode_reactions=
    {
    'H2': {'reaction': 'H2O + 2 e- -> H2 + 2 OH-'}
    'CO': {'reaction': 'H2O + CO2 + 2 e- -> CO + 2 OH-'}
    }
```

All appearing species must be included in the species list (in spite of H_2O and e^-):

```
species=
    {
    'H2': { 'symbol': 'H_2',
            'name': 'hydrogen',
            'diffusion': 4.50e-009, #(m^2/s)
            'bulk concentration': 1e-4}, #(mol/m^3)
    'OH-': {'symbol': 'OH^-',
            'name': 'hydroxide',
            'diffusion': '5.273e-9', #(m^2/s)
            'bulk concentration': 1e-7}, #(mol/m^3)
    'CO2': {'symbol': 'CO_2',
            'name': 'carbon dioxide',
            'diffusion': 1.91e-009, #(m^2/s)
            'bulk concentration': 1e-3} #(mol/m^3)
    }
```

We can now add fluxes. These can be given as production/consumption rate (flux) of the individual species:

```
species['H2']['flux'] = 1e-4 #(mol/s/m^2)
species['CO']['flux'] = 1e-5 #(mol/s/m^2)
```

Alternatively, we can provide them as current densities:

```
species['H2']['flux'] = 1e-4 #(mol/s/m^2)
species['CO']['flux'] = 1e-5 #(mol/s/m^2)
```

Not, however, that only one flux should be defined per equation. The fluxes of the remaining species will be calculated automatically from the reaction equation, for example:

$j_{\mathrm{OH^-}} = 2 \times j_{\mathrm{H_2}} + 2 \times j_{\mathrm{CO}}$ $j_{\mathrm{CO_2}} = -j_{\mathrm{CO}}$

Alternatively, fluxes can be also given as a current density, e.g. if experimental data is at hand that should be tested. This can be defined as:

```
species['H2']['current density'] = 1e-4 #(C/s/m^2=A/m^2)
species['CO']['current density'] = 1e-5 #(C/s/m^2=A/m^2)
```

Internally, these will be then converged into fluxes via:

$j_{\mathrm{OH^-}} = -2 \times i_{\mathrm{OH^-}}$ $j_{\mathrm{CO}} = -2 \times i_{\mathrm{CO}}$

Finally, fluxes can be also given as faradaic efficiencies (FE):

```
species['H2']['FE'] = 0.6 #(%)
species['CO']['FE'] = 0.2 #(%)
```

This, however, requires to also define the total current density in the system dictionary as:

```
system['current density'] = 1e-4
```

If a total current density is given, a check will be performed if the partial current density add up to the total current density. If they do not, the program will stop with a warning. Please define then an unknown species with the missing current density.

4.2.2 Rate-Equations

In COMSOL (and only here, not for any other of the FD solvers), we can define fluxes also as equations depending e.g. on the local concentrations of the species and the local potential. We can e.g. define the flux of H_2 in terms of a Butler-Volmer relation:

$j_{\mathrm{H_2}} = \rho \cdot \theta_{\mathrm{H}} \cdot A \cdot \exp(-[G_a^{\mathrm{eq}} + \alpha \cdot F \cdot \eta]/RT)$,

where ρ is the active site density, θ_{H} is the coverage of H, G_a is the activation barrier of the rate-determining step and α the transfer coefficient. η is the overpotential. Against a reference electrode which does not vary with pH, this is just given as the difference between potential at the reaction plane and potential at the electrode:

$$\eta = (\Phi^{\mathrm{M}} - \Phi^{\ddagger}) - (\Phi^{\mathrm{M,eq}} - \underbrace{\Phi^{\ddagger,\mathrm{eq}}}_{\approx 0}) = (\Phi^{\mathrm{M}} - \Phi^{\ddagger}) - \Phi^{\mathrm{M,eq}}.$$

We now define some COMSOL variables and parameters:

```
comsol_params['Ga']=[str(-0.3*unit_F)+'[J/mol]','Adsorption barrier H']
comsol_params['alpha']=['0.5','Transfer Coefficient']
comsol_params['A']=['1.e13[1/s]','Exponential prefactor']
comsol_params['rho']=['1e-05[mol/m^2]','Density of Active Sites']
comsol_params['theta_max']=['0.4','Maximum Coverage']
comsol_params['Lmol']=['1[l/mol]','unit conversion factor']
comsol_params['Kads']=['1e-4','Adsorption equilibrium constant']
comsol_params['phiEq']=['-0.1[V]','equilibrium potential']
```

What is missing now is to define the coverage of H. We can assume a Langmuir isotherm with a maximum coverage of $\theta_{\mathrm{H}}^{\mathrm{max}}$:

$\theta_{\mathrm{H}} = \frac{\sqrt{K_{\mathrm{ads}} \cdot a_{\mathrm{H_2}}}}{1+\sqrt{a_{\mathrm{H_2}} K_{\mathrm{ads}}}} \theta_{\mathrm{H}}^{\mathrm{max}}$

We add the coverage via a COMSOL variable:

```
comsol_variables['coverage']=['Kads*[[H2]]*Lmol/(1.+[[H2]]*Lmol*Kads)*theta_max','H␣
↪Coverage according to Langmuir isotherm']
```

Note that the surface concentrations of species are indicated here by the double brackets [[...]]. Any species surface concentration can be referred like this.

Finally, we can define the H_2 flux as:

```
species['H2']['flux-equation'] =
    'rho*coverage*exp(-(Ga+alpha*F_const*(phiM-phi-phiEq))/RT)' #(mol/s/m^2)
```

Fixed flux expressions can be combined with flux-equation expressions and the remaining species fluxes will be automatically calculated.

4.2.3 CatMAP

The most advanced method of defining reactant fluxes is via a mean-field kinetic model. This requires to evaluate all fluxes via CatMAP, by setting:

```
species['H2']['flux'] = 'catmap'
```

If any of the fluxes is set to 'catmap', a full CatMAP calculation will be started to evaluate the reaction fluxes. These will be passed to CatINT in order to evaluate the surface concentrations which again requires a CatMAP calculation. An SCF cycle is performed until convergence.

4.3 Using COMSOL within CatiNT

There is a range of predefined COMSOL variables which will be written by default to the COMSOL input file. These can be used when defining new COMSOL variables or requesting specific output.

4.3.1 General COMSOL Functions

There are some generalized COMSOL functions available for use, some of them listed here (`[var]` stands for the variable name):

Functional Call	Equation	Description
`[var]x, d([var],x])`	$\frac{\mathrm{d}[\mathrm{var}]}{\mathrm{d}x}$	derivative with respect to x-direction
`[var].at0(0,[var])`	$[\mathrm{var}](x=0)$	value of the variable at $x = 0$
`intop2([var]*(x<=dest(x)))`	$\int_0^x [\mathrm{var}]\mathrm{d}x$	integral of variable from 0 to x

`intop2` is defined as a domain integral as default integral.

4.3.2 Species-Dependent Variables:

Species dependent variables can be requested by putting the species name `sp` into double brackets behind the physical property symbol:

Variable Name `CatINT`	Variable Name `COMSOL`	Equation	Description
`j[[sp]]`	`j1,j2,...`	$j_i(x)$	Flux of species `sp`
`cp[[sp]]`	`cp1,cp2,...`	$c_i(x)$	Concentration of species `sp`
`ci[[sp]]`	`ci1,ci2,...`	$c_i(t=0)$	Initial Concentration

4.3.3 Global Variables

Variable Name `CatINT`	Mathmatical Expression	Equation	Description
`phi`	–	Electrostatic Potential in Solution	
`rho_s`	$\left(\Phi^{\mathrm{M}} - \Phi^{\mathrm{PZC}}\right) \cdot C_{\mathrm{S}}$	Surface Charge Density	
`rho_c`	$\rho_{\mathrm{c}} = F^2 \sum_i z_i^2 u_i c_i(x)$	Electrolyte Conductivity	
`i_el`	$i_{\mathrm{el}} = F \sum_i z_i j_i$	Electrolyte Current Density	
`delta_phi_iR`	$\Delta\Phi_{\mathrm{iR}}(x) = \int_0^x \frac{i(x)}{\rho_{\mathrm{c}}} \mathrm{d}x$	iR Drop in the Electrolyte as a function of distance to the electrode	
`delta_phi_diff`	$\Delta\Phi_{\mathrm{diff}}(x) = \int_0^x \frac{\sum_i z_i D_i \nabla c_i}{\rho_{\mathrm{c}}} \mathrm{d}x$	Diffusional Potential Drop	

CHAPTER 5

Citations

For usage of Catint, please cite:

- S. Ringe, C.G. Morales-Guio, L.D. Chen, M. Fields, T.F. Jaramillo, C. Hahn, K. Chan, *Double layer charging driven carbon dioxide adsorption limits the rate of electrochemical carbon dioxide reduction on Gold*, *Nat. Commun.* **2020**, *11*, 1.

For usage of CatMAP and COMSOL ®, please check the following web sites for the most current citations:

CatMAP – https://catmap.readthedocs.io

COMSOL ® – www.comsol.com

- search

CHAPTER 6

Indices and tables

- genindex
- modindex
- search