Sensitivity

Version 1.0.0

Generated by Doxygen 1.8.13

Contents

1 Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

2 Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

object

3 Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

4 File Index

4.1 File List

Here is a list of all files with brief descriptions:

5 Namespace Documentation

5.1 NH3_storage_sensitivity Namespace Reference

Sensitivity Analysis of the NH3 Storage and Aging Model.

Functions

- def NH3_Storage_Model_v0 (params, conds)

  The previous NH3 storage model contains the following params and conds...
- def NH3_Storage_Model_v0_1 (params, conds)

  The current NH3 storage model contains the following params and conds...

Variables

- dictionary params = {}
- dictionary conds_lb = {}
- dictionary conds_ub = {}
- dictionary conds_tuples = {}
- analysis = SensitivitySweep(NH3_Storage_Model_v0_1, params, conds_tuples)
- string file_name_simple = "NH3-Analysis-Results-Simple.txt"
- string file_name_full = "NH3-Analysis-Results-Exhaustive.txt"
- bool rel = True
- int per = 10

5.1.1 Detailed Description

Sensitivity Analysis of the NH3 Storage and Aging Model.

Python script using the sensitivity.py script and object to perform a simple sensitivity analysis on the NH3 storage model to see which parameters the model is most or least sensitive to. This can be used to determine whether or not all the reaction schemes and aging mechanisms proposed are useful in determining the NH3 storage capacity on Cu-SSZ-13.

Author

Austin Ladshaw

Date

02/13/2020

Copyright

5.1.2 Function Documentation

5.1.2.1 NH3_Storage_Model_v0()

```
def NH3_storage_sensitivity.NH3_Storage_Model_v0 (
              params,
              conds )
```

The previous NH3 storage model contains the following params and conds...

params["dH1"] = reaction enthalpy (J/mol) for reaction 1

params["dS1"] = reaction entropy (J/K/mol) for reaction 1

params["dH2"] = reaction enthalpy (J/mol) for reaction 2

params["dS2"] = reaction entropy (J/K/mol) for reaction 2

params["dH3"] = reaction enthalpy (J/mol) for reaction 3

params["dS3"] = reaction entropy (J/K/mol) for reaction 3

params["dH4"] = reaction enthalpy (J/mol) for reaction 4

params["dS4"] = reaction entropy (J/K/mol) for reaction 4

params["A1"] = pre-exponential factor (1/hr/kPa^0.25) for aging reacion 1

params["E1"] = reaction rate energy (J/mol) for aging reaction 1

params["A2"] = pre-exponential factor (1/hr/kPa^0.25) for aging reacion 2

params["E2"] = reaction rate energy (J/mol) for aging reaction 2

params["A3f"] = pre-exponential factor (1/hr) for Forward aging reacion 3

params["E3f"] = reaction rate energy (J/mol) for Forward aging reaction 3

params["A3r"] = pre-exponential factor (1/hr) for Reverse aging reacion 3

params["E3r"] = reaction rate energy (J/mol) for Reverse aging reaction 3

params["Z1CuOH_o"] = initial site density (mol/L) for Z1 Cu sites

params["Z2Cu_o"] = initial site density (mol/L) for Z2 Cu sites

params["ZH_o"] = initial site density (mol/L) for solitary Bronsted sites

params["ZH-ZCu_o"] = initial site density (mol/L) for Bronsted sites near inactive Z1 Cu sites

params["ZH-CuO_o"] = initial site density (mol/L) for Bronsted sites near CuO species

```
        conds["T"] = temperature (K) for gas stream

        conds["P_O2"] = partial pressure (kPa) for O2 in gas stream

        conds["P_H2O"] = partial pressure (kPa) for H2O in gas stream

        conds["P_NH3"] = partial pressure (kPa) for NH3 in gas stream

        conds["aging_time"] = time spent aging (hr)

        conds["T_aging"] = temperature during aging (K) for gas stream

        conds["P_O2_aging"] = partial pressure during aging (kPa) for O2 in gas stream

        conds["P_H2O_aging"] = partial pressure during aging (kPa) for O2 in gas stream


        NOTE:

           aging_time = 0 for "de-greened" catalyst

        NOTE2:

               You can put more information in params if you want sensitivity analysis
                to also cover the model sensitivity to things like temperature and
                partial pressures.

        ------------ MODEL INFORMATION GIVEN BELOW -------------

        Capacity Reactions:

            (1)      (Z1CuOH) + NH3 <== ==> [(Z1CuOH)-NH3]

            (2)      (Z2Cu) + NH3 <== ==> [(Z2Cu)-NH3]

            (3)      (ZH) + NH3 <== ==> [(ZH)-NH3]

            (4)      (Z1CuOH) + H2O <== ==> [(Z1CuOH)-H2O]


        Aging Reactions:

            (1)      (ZH)(ZCu) + 0.25 O2 --> (Z2Cu) + 0.5 H2O

            (2)      (ZH) + 0.25 O2 --> (Z) + 0.5 H2O

            (3)      (Z1CuOH) <-- --> (ZH)(CuO)

        w1 = (availability of Z1CuOH after aging)

        w2 = (availability of Z2Cu after aging)

        w3 = (availability of total ZH sites after aging)
                total ZH sites = (ZH) + (ZH)(ZCu) + (ZH)(CuO)
```

5.1.2.2 NH3_Storage_Model_v0_1()

```
def NH3_storage_sensitivity.NH3_Storage_Model_v0_1 (
              params,
              conds )
```

The current NH3 storage model contains the following params and conds...

params["dH1"] = reaction enthalpy (J/mol) for reaction 1

params["dS1"] = reaction entropy (J/K/mol) for reaction 1

params["dH2"] = reaction enthalpy (J/mol) for reaction 2

params["dS2"] = reaction entropy (J/K/mol) for reaction 2

params["dH3"] = reaction enthalpy (J/mol) for reaction 3

params["dS3"] = reaction entropy (J/K/mol) for reaction 3

params["dH4"] = reaction enthalpy (J/mol) for reaction 4

params["dS4"] = reaction entropy (J/K/mol) for reaction 4

params["k1"] = aging rate (1/hr/kPa^0.25) for aging reacion 1

params["k2"] = aging rate (1/hr/kPa^0.25) for aging reacion 2

params["k3f"] = aging rate (1/hr) for Forward aging reacion 3

params["k3r"] = aging rate (1/hr) for Reverse aging reacion 3

params["Z1CuOH_o"] = initial site density (mol/L) for Z1 Cu sites

params["Z2Cu_o"] = initial site density (mol/L) for Z2 Cu sites

params["ZH_o"] = initial site density (mol/L) for solitary Bronsted sites

params["ZH-ZCu_o"] = initial site density (mol/L) for Bronsted sites near inactive Z1 Cu sites

params["ZH-CuO_o"] = initial site density (mol/L) for Bronsted sites near CuO species

```
conds["T"] = temperature (K) for gas stream

conds["P_O2"] = partial pressure (kPa) for O2 in gas stream

conds["P_H2O"] = partial pressure (kPa) for H2O in gas stream

conds["P_NH3"] = partial pressure (kPa) for NH3 in gas stream

conds["aging_time"] = time spent aging (hr)

conds["T_aging"] = temperature during aging (K) for gas stream

conds["P_O2_aging"] = partial pressure during aging (kPa) for O2 in gas stream

conds["P_H2O_aging"] = partial pressure during aging (kPa) for O2 in gas stream


NOTE:

   aging_time = 0 for "de-greened" catalyst

NOTE2:

       You can put more information in params if you want sensitivity analysis
        to also cover the model sensitivity to things like temperature and
        partial pressures.

------------ MODEL INFORMATION GIVEN BELOW -------------

Capacity Reactions:

    (1)      (Z1CuOH) + NH3 <== ==> [(Z1CuOH)-NH3]

    (2)      (Z2Cu) + NH3 <== ==> [(Z2Cu)-NH3]

    (3)      (ZH) + NH3 <== ==> [(ZH)-NH3]
```

```
        (4)      (Z1CuOH) + H2O <== ==> [(Z1CuOH)-H2O]

    Aging Reactions:

        (1)      (ZH)(ZCu) + 0.25 O2 --> (Z2Cu) + 0.5 H2O

        (2)      (ZH) + 0.25 O2 --> (Z) + 0.5 H2O

        (3)      (Z1CuOH) <-- --> (ZH)(CuO)

    w1 = (availability of Z1CuOH after aging)

    w2 = (availability of Z2Cu after aging)

    w3 = (availability of total ZH sites after aging)
            total ZH sites = (ZH) + (ZH)(ZCu) + (ZH)(CuO)
```

5.1.3 Variable Documentation

5.1.3.1 params

```
dictionary NH3_storage_sensitivity.params = {}
```

5.1.3.2 conds_lb

```
dictionary NH3_storage_sensitivity.conds_lb = {}
```

5.1.3.3 conds_ub

```
dictionary NH3_storage_sensitivity.conds_ub = {}
```

5.1.3.4 conds_tuples

```
dictionary NH3_storage_sensitivity.conds_tuples = {}
```

5.1.3.5 analysis

```
NH3_storage_sensitivity.analysis = SensitivitySweep(NH3_Storage_Model_v0_1, params, conds_↩
tuples)
```

5.1.3.6 file_name_simple

```
string NH3_storage_sensitivity.file_name_simple = "NH3-Analysis-Results-Simple.txt"
```

5.1.3.7 file_name_full

```
string NH3_storage_sensitivity.file_name_full = "NH3-Analysis-Results-Exhaustive.txt"
```

5.1.3.8 rel

```
bool NH3_storage_sensitivity.rel = True
```

5.1.3.9 per

```
int NH3_storage_sensitivity.per = 10
```

5.2 sensitivity Namespace Reference

Simple Sensitivity Analysis.

Classes

- class Sensitivity
  Sensivity class object for simple analyses.
- class SensitivitySweep
  SensitivitySweep class object for performing a full sensivitity analysis.

Functions

- def update_cond (cond_value, cond_limit_lower, cond_limit_upper)
  Helper function to iterate through all permutations of conditions.

5.2.1 Detailed Description

Simple Sensitivity Analysis.

Python script to perform a sensitivity analysis on a simple model by taking finite partial derivatives of that model given a set of parameters.

Author

Austin Ladshaw

Date

02/12/2020

Copyright

5.2.2 Function Documentation

5.2.2.1 update_cond()

```
def sensitivity.update_cond (
              cond_value,
              cond_limit_lower,
              cond_limit_upper )
```

Helper function to iterate through all permutations of conditions.

What this function does is update the given list of cond_value according to the corresponding limits given. This allows the sensitivity sweep object to iteratively move through all permutations of conditions, within the specified limits, to check all combinations of conditions for parameter sensivitity.

For Instance:

```
Consider a state machine that has 3 conditions (A, B, C), each of which can
have 3 different states (0, 1, 2). To exhaustively test all the states
possible, we have to iterate through all permutations of the variables
A, B, and C, at all possible states they can be in (0, 1, 2). In total,
there would be 3^3 (=27) permutations to produce.
```

The above example would need to produce the following...

A B C | A B C | A B C
---
0 0 0 | 0 0 1 | 0 0 2
1 0 0 | 1 0 1 | 1 0 2
2 0 0 | 2 0 1 | 2 0 2
0 1 0 | 0 1 1 | 0 1 2
1 1 0 | 1 1 1 | 1 1 2
2 1 0 | 2 1 1 | 2 1 2
0 2 0 | 0 2 1 | 0 2 2
1 2 0 | 1 2 1 | 1 2 2
2 2 0 | 2 2 1 | 2 2 2

The function only changes one state at a time and that changed state is based on the current state passed to it. It is meant to be coupled with a while loop that will start from an initial state and continue until this function returns true. When this function returns false, this means that states can still be updated.

Function will return True after the last permutation has been made

Parameters

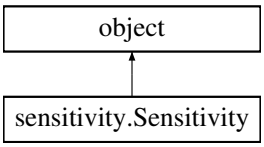
cond_value	current list of values of conditions that needs updating
cond_limit_lower	list of the upper limits of the conditions
cond_limit_upper	list of the upper limits of the conditions

6 Class Documentation

6.1 sensitivity.Sensitivity Class Reference

Sensivity class object for simple analyses.

Inheritance diagram for sensitivity.Sensitivity:



Public Member Functions

- def __init__ (self, func, func_params, func_conds)

  Constructor for the object.
- def __str__ (self)

  Function to print the results of the analysis to the console.
- def eval_func (self)

  Function to call the users function with their parameters and conditions.
- def compute_partials (self, relative=False, per=1)

  Function to compute all partials.

Public Attributes

- errors
- func

  A function that produces a single output given a set of parameters and conditions.
- func_params

  A set of parameters that the sensitivity analysis will be performed on.
- func_conds

  A set of other conditions or information the model needs to use.
- partials

  Computed set of partial derivatives or percent changes.
- relative_sensitivity

  When set to True, the partials are computed based on a percent change to the variable.
- percent_change

  Percent change to apply to parameters when relative_sensitivity = True.
- partials_computed
- lowest_sensitivity
- highest_sensitivity
- sorted_param_sensitivity

  Stores a sorted map of most to least sensitve parameters.

6.1.1 Detailed Description

Sensivity class object for simple analyses.

This object is used to perform small to medium scale sensitivity analyses on a model function written in python. It provides a quick an easy way to check a simple model to see the responsiveness in the model to changes in its parameters. Changes can be computed as finite difference derivatives or percent changes in function response

NOTE:

```
func_params and func_conds are (or can be) used interchangibly. The
difference is when the routine looks to compute sensitivity, it only
does so for the func_params under the conditions of the system.
```

6.1.2 Constructor & Destructor Documentation

6.1.2.1 __init__()

```
def sensitivity.Sensitivity.__init__ (
              self,
              func,
              func_params,
              func_conds )
```

Constructor for the object.

Parameters

func	pointer to a func defined and written in python
func_params	map or dictionary of parameters the function depends on
func_conds	map or dictionary of conditions the function depends on

6.1.3 Member Function Documentation

6.1.3.1 __str__()

```
def sensitivity.Sensitivity.__str__ (
              self )
```

Function to print the results of the analysis to the console.

6.1.3.2 eval_func()

```
def sensitivity.Sensitivity.eval_func (
              self )
```

Function to call the users function with their parameters and conditions.

6.1.3.3 compute_partials()

```
def sensitivity.Sensitivity.compute_partials (
              self,
              relative = False,
              per = 1 )
```

Function to compute all partials.

Parameters

relative	if False, then partials are computed via finite difference derivatives if True, then partials are computed as percent changes in function for percent change in parameters
per	the percentage change to use in the parameters (only used if relative == True)

6.1.4 Member Data Documentation

6.1.4.1 errors

```
sensitivity.Sensitivity.errors
```

6.1.4.2 func

```
sensitivity.Sensitivity.func
```

A function that produces a single output given a set of parameters and conditions.

6.1.4.3 func_params

```
sensitivity.Sensitivity.func_params
```

A set of parameters that the sensitivity analysis will be performed on.

6.1.4.4 func_conds

```
sensitivity.Sensitivity.func_conds
```

A set of other conditions or information the model needs to use.

6.1.4.5 partials

`sensitivity.Sensitivity.partials`

Computed set of partial derivatives or percent changes.

6.1.4.6 relative_sensitivity

`sensitivity.Sensitivity.relative_sensitivity`

When set to True, the partials are computed based on a percent change to the variable.

6.1.4.7 percent_change

`sensitivity.Sensitivity.percent_change`

Percent change to apply to parameters when relative_sensitivity = True.

6.1.4.8 partials_computed

`sensitivity.Sensitivity.partials_computed`

6.1.4.9 lowest_sensitivity

`sensitivity.Sensitivity.lowest_sensitivity`

6.1.4.10 highest_sensitivity

`sensitivity.Sensitivity.highest_sensitivity`

6.1.4.11 sorted_param_sensitivity

`sensitivity.Sensitivity.sorted_param_sensitivity`

Stores a sorted map of most to least sensitve parameters.

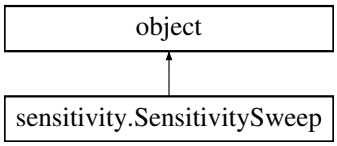
The documentation for this class was generated from the following file:

- sensitivity.py

6.2 sensitivity.SensitivitySweep Class Reference

SensitivitySweep class object for performing a full sensivitity analysis.

Inheritance diagram for sensitivity.SensitivitySweep:



Public Member Functions

- def __init__ (self, func, func_params, func_conds_tuples)

  Constructor for the sweep object.
- def __str__ (self)

  Function to print out results to console (Only useful for quick visualization.
- def run_sweep (self, sensitivity_file_name="SensitivitySweepAnalysis.dat", relative=False, per=1)

  Run the Sensitivity Sweep Analysis and print results to a file.
- def run_exhaustive_sweep (self, sensitivity_file_name="ExhaustiveSensitivitySweepAnalysis.dat", relative=False, per=1)

  Function to perform an Exhaustive Sensitivity Analysis.

Public Attributes

- errors
- sweep_computed
- cond_tuples
- sens_obj
- sens_maps

  Initialize a list of maps for sensitivity results to be stored digitally.
- max_sens_map

  Map of each parameter's maximum sensitivity The below objects (max_∗ and min_∗ sens_map) have the following format...
- min_sens_map

  Map of each parameter's minimum sensitivity The below objects (max_∗ and min_∗ sens_map) have the following format...

6.2.1 Detailed Description

SensitivitySweep class object for performing a full sensivitity analysis.

The SensitivitySweep object is an object that uses the Sensitivity object to calculation partials or changes in a model with changes in parameters, but also repeats this process for a ranged of conditions to produce sensitivity matrices that are output to a file. This is necessary for complex models as it is possible that a model will not be sensitive to a certain parameter under certain conditions, but becomes more sensitive as the conditions change.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 __init__()

```
def sensitivity.SensitivitySweep.__init__ (
              self,
              func,
              func_params,
              func_conds_tuples )
```

Constructor for the sweep object.

Parameters

func	pointer to a func defined and written in python
func_params	map or dictionary of parameters the function depends on
func_conds_tuples	map of tuples of conditions to sweep through where the first tuple arg is the lower_limit and the second is the upper_limit

NOTE:

```
 func_conds_tuples must be a dictionary whose keys are the simulation/model
condtions and whose values are tuples representing the lower and upper
bounds of the conditions, respectively.

    e.g.,  func_conds_tuples["Temp"] = (273, 373)

                a condition for temperature that spans 100 degrees
```

6.2.3 Member Function Documentation

6.2.3.1 __str__()

```
def sensitivity.SensitivitySweep.__str__ (
              self )
```

Function to print out results to console (Only useful for quick visualization.

Sweeps automatically puts this info in a text file)

6.2.3.2 run_sweep()

```
def sensitivity.SensitivitySweep.run_sweep (
              self,
              sensitivity_file_name = "SensitivitySweepAnalysis.dat",
              relative = False,
              per = 1 )
```

Run the Sensitivity Sweep Analysis and print results to a file.

User may also specify whether or not to use relative parameter changes and the percent to change

6.2.3.3 run_exhaustive_sweep()

```
def sensitivity.SensitivitySweep.run_exhaustive_sweep (
              self,
              sensitivity_file_name = "ExhaustiveSensitivitySweepAnalysis.dat",
              relative = False,
              per = 1 )
```

Function to perform an Exhaustive Sensitivity Analysis.

The exhaustive sweep uses the helper function update_cond() to go through all condition permutations within the specified boundaries of each condition variable.

6.2.4 Member Data Documentation

6.2.4.1 errors

```
sensitivity.SensitivitySweep.errors
```

6.2.4.2 sweep_computed

```
sensitivity.SensitivitySweep.sweep_computed
```

6.2.4.3 cond_tuples

```
sensitivity.SensitivitySweep.cond_tuples
```

6.2.4.4 sens_obj

```
sensitivity.SensitivitySweep.sens_obj
```

6.2.4.5 sens_maps

```
sensitivity.SensitivitySweep.sens_maps
```

Initialize a list of maps for sensitivity results to be stored digitally.

The below object (self.sens_maps) has the following format...

```
   self.sens_maps[i] = {}                          // i = permutation number  -->  map of data
   self.sens_maps[i]["func_result"]                // = result of the function for that permutation
   self.sens_maps[i]["cond_set"] = {}              // map of conditions for the given permutation
   self.sens_maps[i]["param_response"] = {}        // map of function responses or partials for the parameters
   self.sens_maps[i]["cond_set"][cond]             // = value of the given condition (cond) for the given perm
   self.sens_maps[i]["param_response"][param]      // = value of the function response to a change in the give
```

6.2.4.6 max_sens_map

`sensitivity.SensitivitySweep.max_sens_map`

Map of each parameter's maximum sensitivity The below objects (max_∗ and min_∗ sens_map) have the following format...

self.∗_sens_map[param] = {} // map of max or min parameter results for the given param // Keys in this map include: func_result, param_response, and cond_set

self.∗_sens_map[param]["func_result"] // = result of the function for that max or min param sensitivity result

self.∗_sens_map[param]["param_response"] // = value of the function response to the param change under these conditions // This will be the max or min response for the parameter

self.∗_sens_map[param]["cond_set"] = {} // map of conditions for the max or min parameter response

self.∗_sens_map[param]["cond_set"][cond] // = value of the given condition (cond) for the max or min parameter response

6.2.4.7 min_sens_map

`sensitivity.SensitivitySweep.min_sens_map`

Map of each parameter's minimum sensitivity The below objects (max_∗ and min_∗ sens_map) have the following format...

self.∗_sens_map[param] = {} // map of max or min parameter results for the given param // Keys in this map include: func_result, param_response, and cond_set

self.∗_sens_map[param]["func_result"] // = result of the function for that max or min param sensitivity result

self.∗_sens_map[param]["param_response"] // = value of the function response to the param change under these conditions // This will be the max or min response for the parameter

self.∗_sens_map[param]["cond_set"] = {} // map of conditions for the max or min parameter response

self.∗_sens_map[param]["cond_set"][cond] // = value of the given condition (cond) for the max or min parameter response

The documentation for this class was generated from the following file:

- sensitivity.py

7 File Documentation

7.1 NH3_storage_sensitivity.py File Reference

Namespaces

- NH3_storage_sensitivity

  Sensitivity Analysis of the NH3 Storage and Aging Model.

Functions

- def NH3_storage_sensitivity.NH3_Storage_Model_v0 (params, conds)

  The previous NH3 storage model contains the following params and conds...
- def NH3_storage_sensitivity.NH3_Storage_Model_v0_1 (params, conds)

  The current NH3 storage model contains the following params and conds...

Variables

- dictionary NH3_storage_sensitivity.params = {}
- dictionary NH3_storage_sensitivity.conds_lb = {}
- dictionary NH3_storage_sensitivity.conds_ub = {}
- dictionary NH3_storage_sensitivity.conds_tuples = {}
- NH3_storage_sensitivity.analysis = SensitivitySweep(NH3_Storage_Model_v0_1, params, conds_tuples)
- string NH3_storage_sensitivity.file_name_simple = "NH3-Analysis-Results-Simple.txt"
- string NH3_storage_sensitivity.file_name_full = "NH3-Analysis-Results-Exhaustive.txt"
- bool NH3_storage_sensitivity.rel = True
- int NH3_storage_sensitivity.per = 10

7.2 sensitivity.py File Reference

Classes

- class sensitivity.Sensitivity

  Sensivity class object for simple analyses.
- class sensitivity.SensitivitySweep

  SensitivitySweep class object for performing a full sensivitity analysis.

Namespaces

- sensitivity

  Simple Sensitivity Analysis.

Functions

- def sensitivity.update_cond (cond_value, cond_limit_lower, cond_limit_upper)

  Helper function to iterate through all permutations of conditions.

Index