Introduction to Use Association NRTL-SAC Python Code

This document explains how to use the attached Association NRTL-SAC activity coefficient model python code. An example file is also included for illustration. The reference for the Association NRTL-SAC model is “*Hao, Y., & Chen, C. C. (2019). Nonrandom Two-Liquid Segment Activity Coefficient Model with Association Theory. Industrial & Engineering Chemistry Research, 58(28), 12773-12786.”.*

# Setup external python libraries

Several open-source third-party python libraries, e.g. numpy, scipy, matplotlib, have been used for numerical computation, optimization, and plotting. A “requirements.txt” file is attached for installing all dependent libraries by using the command “pip install -r requirements.txt”. Python virtual environment is recommended.

# Example Association NRTL-SAC model application

Activity coefficient calculation for a binary system, chloroform-water, is illustrated here to show how to use the Association NRTL-SAC model python code. The example code is in the attached “example\_AssociationNRTLSAC.py” file.

## Create “AssociationNRTLSACMolecule” object to hold molecular parameters

# molecular parameters can be added when initializing the object

chloroform **=** AssociationNRTLSACMolecule**(**name**=**'chloroform'**,**

r**=**2.87**,**

X**=**0.269**,**

Y**=**0.297**,**

nu\_D**=**1**,**

delta\_D**=**0.145**)**

# molecular parameters can also be added from the attributes

water **=** AssociationNRTLSACMolecule**(**name**=**'water'**)**

water**.**r **=** 0.76

water**.**Y **=** 0.492

water**.**nu\_D **=** water**.**nu\_A **=** 2

water**.**delta\_D **=** water**.**delta\_A **=** 1.0

r: Normalized volume parameters for the component.

X and Y: Conceptual segment numbers for nonpolar and polar segments.

nu\_A and nu\_D: number of HB acceptor and donor sites in each molecule.

delta\_A and delta\_D: Association strength parameters of HB acceptor and donor sites.

Association NRTL-SAC molecular parameters of common solvents are reported in Table 4 in the reference.

## Create “AssociationNRTLSAC” object for the multicomponent system

# AssociationNRTLSAC class is constructed by wrapping all AssociationNRTLSACMolecule objects in the

# "molecules" argument

ChloroformWaterBinary **=** AssociationNRTLSAC**(**molecules**=(**chloroform**,** water**))**

The “AssociationNRTLSAC” object can be created for the multi-component system by passing the “molecules” argument, which holds all “AssociationNRTLSACMolecule” objects in a list or tuple.

## Calculate activity coefficients from “AssociationNRTLSAC” object

# activity coefficient is calculated by compute function by pass in temperature and mole fraction

x **=** np**.**array**([**0.2**,** 0.8**])**

T **=** 300.0 # K

lnGamma **=** ChloroformWaterBinary**.**compute**(**x**,** T**)**

**print(**f'Calculated ln(gamma):\n{lnGamma}'**)**

After the “AssociationNRTLSAC” object is created from the input parameters, activity coefficients can be calculated repeatedly at any temperature and composition using the “compute” function.

x: mole fraction of each component, which is specified as numpy 1-dimensional array of size equal to the number of components (NOC) in the system.

T: temperature in Kelvin.

Output: the “compute” function returns as numpy 1-dimentional array with the size of NOC in the system.

# Additional calculated details (optional)

# additional information of calculated values can be retrieved by passing in an empty dictionary

info **=** **{}**

lnGamma **=** ChloroformWaterBinary**.**compute**(**x**,** T**,** info**)**

**print(**f'Additional calculated values:\n{info}'**)**

If an empty dictionary (info) is passed to the “compute” function (optional), some calculated details will be stored in that dictionary object. An example “info” object holding information is shown below:

**{**'xA'**:** **[**0.13804127110109904**,** 0.6125265667831137**,** 0.18647545025320697**],**

'lnGammaC'**:** **[-**0.25208731207917334**,** **-**0.028968923750010422**],**

'lnGammaR'**:** **[**0.4838221528364055**,** 0.0808723270678825**],**

'lnGammaA'**:** **[**2.8584956519828877**,** 0.29794976343586255**]}**

xA: unbonded site fraction in the mixture in the order of all HB acceptor sites, and all HB donor sites.

lnGammaC, lnGammaR, lnGammaA: combinatorial, residual, and association contributions to the activity coefficients.

# Association NRTL-SAC molecular parameter database

A database file is attached containing all molecular parameters reported in Table 4 in the reference. Users could retrieve existing molecular parameters, add new molecular parameters, and modify or delete existing parameters on their own copy of the database file.

All data in the database can be printed:

import database as db

db**.**show\_all\_data**()**

Molecular parameters can be retrieved by their names in the database (Same names as in the reference and case insensitive):

# molecular parameters can be retrieved from the component name (case insensative)

# and return as AssociationNRTLSACMolecule object

acetone **=** db**.**get\_molecular\_data**(**'acetone'**)**

Returned as “AssociationNRTLSACMolecule” object. If the name does not exist in the database, an exception will be raised.

New molecular parameters can be added by passing “AssociationNRTLSACMolecule” object:

# new molecular parameters could be added by AssociationNRTLSACMolecule object

test\_molecule **=** AssociationNRTLSACMolecule**(**name**=**'test molecule'**,**

r**=**1.23**,**

X**=**0.5**,**

nu\_A**=**2**,**

delta\_A**=**1.3**)**

db**.**add\_to\_db**(**test\_molecule**)**

Existing molecular parameters can be modified by passing “AssociationNRTLSACMolecule” object:

# existing molecular parameters could be modified

test\_molecule**.**r **=** 3.14

db**.**update\_molecule\_in\_db**(**test\_molecule**)**

Existing molecular parameters can be deleted from its name:

# existing molecular parameters could be deleted

db**.**delete\_molecule\_from\_db**(**'test molecule'**)**