ORNL/TM-2017/748

Chemics-Reactors: A Preliminary Python Program for Implementing Network Models of Multiphase Reactors

Jonathan E. Sutton
Gavin M. Wiggins
C. Stuart Daw

December 18, 2018

OAK RIDGE NATIONAL LABORATORY
MANAGED BY UT-BATTELLE FOR THE US DEPARTMENT OF ENERGY

ORNL/TM-2017/748

Chemical Sciences Division
Engineering & Transportation Sciences Division

Chemics-Reactors: A Preliminary Python Program for Implementing Network Models of Multiphase Reactors

Jonathan E. Sutton
Gavin M. Wiggins
C. Stuart Daw

Date Published:
December 18, 2017

Prepared by
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, TN 37831-6283
managed by
UT-BATTELLE, LLC
for the
US DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

We thank our colleagues Rick French, Kristiina Iisa, and Danny Carpenter at the National Renewable Energy Laboratory for providing information on the equipment setup for the vapor phase upgrading system. We thank Jeremy Leong, Trevor Smith, and Kevin Craig of the U.S. Department of Energy Bioenergy Technologies Office, the Consortium for Computational Physics and Chemistry (CCPC, www.cpcbiomass.org), and ChemCatBio (www.chemcatbio.org) for funding support.

ABSTRACT

We discuss the design and implementation of a preliminary software package written in Python 3 that is intended to represent complex multiphase reactors as networks of ideal continuous stirred tank reactors. This software also implements statistical design of experiments, uncertainty quantification, and global sensitivity analysis. These advanced features can provide important qualitative and quantitative insights into the effect of operating conditions and model parameters on predicted reactor performance. We demonstrate the utility of the program by modeling the vapor phase catalytic upgrading of bio-oil in a bubbling fluidized bed reactor.

1. INTRODUCTION

Low-order modeling with reactor networks is useful for bridging the gap between an ideal reactor model and a full computational fluid dynamics (CFD) treatment of the reactor. Ideal reactor models are very inexpensive to solve and can handle complex kinetics easily, but they can fail to capture important practical hydrodynamic features, such as dead zones and bypassing. A full CFD treatment can readily capture important hydrodynamic features with high fidelity, but the addition of energy balances and reaction kinetics can impose extremely high computational costs, making many types of research and design studies with such models impractical. Reactor network models can effectively help to bridge this gap by dividing the reactor into multiple coupled hydrodynamic regions that can be approximated by a network of connected ideal reactors. Such models can capture many of the most important global features dominating overall reactor performance, including complex reaction kinetics and heat transfer, while greatly reducing the computational overhead.

In this technical report, we describe the design and implementation of an object-oriented program Chemics-Reactors that is written in Python 3 for the purpose of facilitating steady-state network modeling of complex multiphase reactors. It has specifically been written to incorporate advanced experimental design and model analysis features, including uncertainty quantification and global sensitivity analysis. We demonstrate some of its key features and capabilities with a case study: the upgrading of bio-oil over HZSM-5 in a bubbling bed reactor.

2. CHEMICS-REACTORS

2.1 DESIGN PHILOSOPHY AND GENERAL ARCHITECTURE

Chemics-Reactors is designed to employ object oriented modules with good data encapsulation and few dependencies. Certain classes and methods require the use of functions found in the Chemics library, and the program is complementary to and a natural outgrowth of the modeling efforts that led to that library. A diagram of the modules and their dependencies is given in Figure 1. The purpose of each module as well as external dependencies is given in Table 1. The program has been written to be as flexible as possible, with models specified using ordinary Python objects (dictionaries, lists, ordered dictionaries, and named tuples) in the special module `model_input.py`.

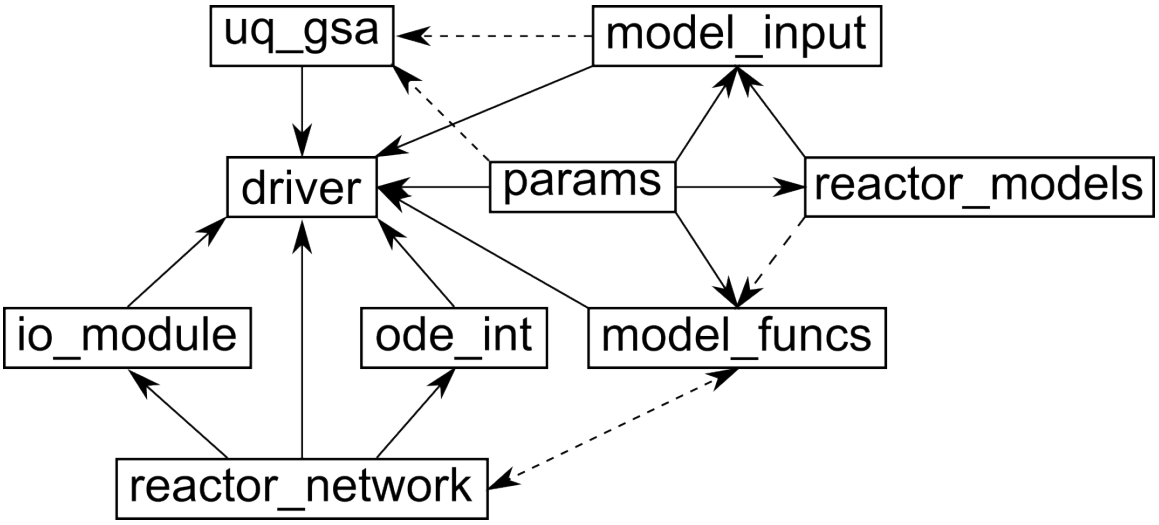

Figure 1. Module dependencies in Chemics-Reactors. The reactor_network module depends on the Chemics library, and the model_funcs module may also depend on the Chemics library (not shown). Solid lines indicates a module import relationship (with the arrow pointing to the module doing the importing). Dashed lines indicate that one or more module objects (or the module itself) are passed as an argument to a function in the dependent module.

Table 1. Overview of module purposes and external dependencies.

Module	Purpose	External Dependencies
`driver.py`	Main driver routine to run program	`NumPy, mpi4py`
`reactor_network.py`	Implements network object (see Tables 2-5 for an overview of the classes and objects in this module)	`NumPy, chemics`
`io_module.py`	Implements output routines for basic runs as well as for design of experiments, uncertainty quantification, and global sensitivity analysis (types of output are described in Section 2.6)	`NumPy`
`ode_int.py`	Wraps ODE solvers and provides an interface between the balance equations defined in the reactor_network module (see Section 2.4) and the functional form required by the ODE solver	`NumPy, SciPy`
`uq_gsa.py`	Uncertainty quantification, global sensitivity analysis, and design of experiments routines (see Section 2.5 for how to use the features of this module and an overview of the related theory)	`NumPy, pyDOE, sobol_seq`
`model_input.py`	Main input module for defining model parameters (see Sections 2.3, 2.4, and 2.5 for setting the options in this module)	`NumPy`
`model_funcs.py`	User-defined functions (see Section 2.7)	Variable dependencies
`params.py`	Module for storing global variables used/modified by multiple objects	Not applicable
`reactor_models.py`	Functions for defining standard reactor models	Not applicable

The main program is contained in the driver source file. The driver routine converts the input parameters in `model_input` into a `Network` object and then constructs class objects that interact with the

`Network` object in useful ways. It also handles the solution of the system of ODEs and any advanced model analysis requested. The `params` module is also very important to the software package. Its function is to serve as a central repository of global variables that can be accessed by other key objects during the simulation. For instance, it is used to store the current values of the key operating conditions used in the user-defined mass flow rate calculation function that is passed to and stored in the network object. It can also be used to store the kinetic parameters required by the rate law or anything else that the user might need.

The basic philosophy of Chemics-Reactors is that the steady-state peformance of any real reactor can be represented as a network of smaller ideal stirred tank sub-reactors with mass flows between individual sub-reactors in the network. In many respects, this is essentially what is done with the finite volume approach in certain computational fluid dynamics (CFD) programs (e.g., MFiX), and CFD simulations can be converted into networks of ideal reactors. The primary difference is in the total number of reactors. Instead of dividing the computational domain into many thousands or even millions of ideal reactors, Chemics-Reactors aims to reduce the computational burden by approximating each region of similar behavior in the macroscopic reactor as a single ideal reactor. A CFD simulation with thousands of cells can thus be approximated in Chemics-Reactors as the collective behavior of a small number of interacting reactor sub-units.

Conceptually, a reactor network can be thought of in terms of graph theory. Each network is composed of a number of nodes (sources, sinks, junctions, and reactors) and the edges (flow streams) connecting them. The graph of the reactor network is always node-terminated. Nodes can be of four types: sources, sinks, junctions, and reactors. These are implemented in the software package by the `Source`, `Sink`, `Junction`, and `Reactor` subclasses of the `Node` class in the `reactor_network` module. Sources, sinks, and junctions are different from reactors in one key respect; these nodes have no associated volume and cannot contain mass. They differ from each other in the number and type of streams associated with them. Sources and sinks have only a single stream exiting and entering them, respectively. Junctions may have any number of influent and effluent streams, provided there is at least one of each. Reactors differ from the other nodes in that they can have mass resident in the system, and that mass can undergo chemical reactions. Streams are implemented in the software package by the `Stream` class in the `reactor_network` module. Streams are required to be phase pure, but this incurs no loss of generality as a two-phase stream can be represented as two parallel pure streams with the same start and end points. The choice of the ideal stirred tank as the basic reactor design also incurs no loss of generality. Other common types of ideal reactors (batch, semi-batch, and tubular) are readily represented as either a single stirred tank (batch, which is a reactor with no inlet and no outlet, and semi-batch, which is a reactor with only an inlet) or as a series of tanks (tubular). Nodes and streams are then bundled together into a single reactor network object encapsulating all information about the reactor model. This object is implemented by the `Network` class in the `reactor_network` module. Some examples of networks that can be modeled in Chemics-Reactors are shown in Figure 2. In Figure 2 sources, sinks, and junctions are represented by small boxes, reactors are represented by large boxes, and streams are represented by arrows between boxes.

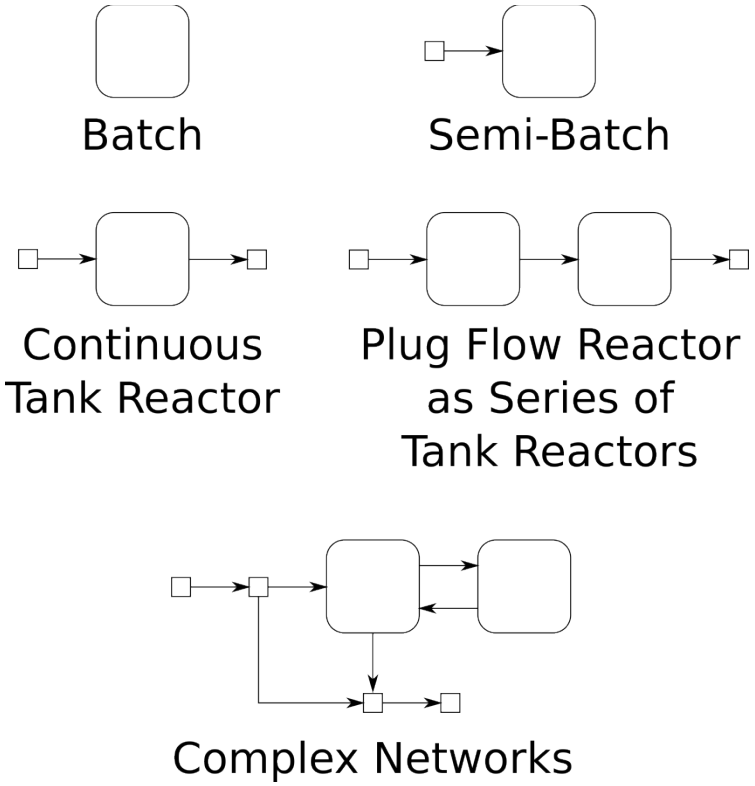

Figure 2. Network diagrams illustrating different types of reactor networks that might be constructed with Chemics-Reactors. Large rounded squares are reactors, and small squares are other nodes (sources, sinks, or junctions). Arrows represent streams connecting reactors.

Chemics-Reactors networks are capable of handling gas, solid, and liquid phase kinetics as well as interphase transport, and in principle could be also used to model a separation unit. Gases are either assumed to be ideal (the default), or a user-defined equation of state can be provided. Solid phases are composed of mixtures of particles with the capability to specify independent particle properties for each species in the mixture. Each species can be chemically distinct (e.g., fresh and spent catalyst), or it can represent different size fractions of the same species, or a combination of the two. Liquid phases are similar to solids phases, except the array of particle diameters is replaced by a mixture-averaged viscosity. The reactor network is permitted to be non-isothermal with individual reactors having either adiabaticity or an overall heat transfer coefficient specified. We summarize the key features of the `reactor_network` module in Table 2, the `Node` class and its `Source`, `Sink`, `Junction`, and `Reactor` subclasses in Table 3, the `Stream` class in Table 4, and the `Network` class in Table 5.

Table 2. Overview of classes in the reactor network module.

Class	Description
`Node`	Base type for Source, Sink, Junction, and Reactor node subclasses
`Source`	Node for terminating the start of an inlet stream
`Sink`	Node for terminating the end of an outlet stream
`Junction`	Node for mixing/splitting two or more streams
`Reactor`	Node that can have mass inventories in multiple phases; any number of inlet/outlet streams
`Stream`	Connects two nodes to permit mass flow between them
`Network`	Bundles lists of Node and Stream objects with additional metadata describing the network topology

Table 3. Overview of node properties and data structures.

Property	Description	Notes
`pos`	Position in Cartesian coordinates	
`node_id`	ID number of the node, as assigned by the library	
`name`	name of the node, as assigned by the user in the `model_input` module	
`num_phases`	The number of phases present	Present for all `Node` subclasses, but only used by `Reactors`
`streams_out`	A list of all the stream ID numbers leaving the node	Empty for `Sink` nodes, at least one present for `Junction` nodes
`streams_in`	A list of all the stream ID numbers entering the node	Empty for `Source` nodes, at least one present for `Junction` nodes
`diam`	Diameter	Only for `Reactor` nodes
`height`	Height	Only for `Reactor` nodes
`T`	Temperature	Only for `Reactor` nodes
`P`	Pressure	Only for `Reactor` nodes
`vol`	Volume	Only for `Reactor` nodes
`species`	List of lists of species in each phase	Only for `Reactor` nodes
`mass_fracs`	List of arrays of mass fraction of each species in each phase	Only for `Reactor` nodes
`masses`	List of total mass in each phase	Only for `Reactor` nodes
`mw`	List of arrays of molecular mass of each species in each phase	Only for `Reactor` nodes
`dp`	List of arrays of particle diameters of each species in each phase	Only for `Reactor` nodes and solid phases (gas and liquid phases have `None`)
`rho`	List of arrays of densities of each species in each phase	Only for `Reactor` nodes and solid/liquid phases (gas phase is just the average density)
`rho_avg`	List of average density in each phase	Only for `Reactor` nodes
`eps`	List of volume fraction in each phase	Only for `Reactor` nodes
`mu`	List of dynamic viscosity in each phase	Only for `Reactor` nodes and gas/liquid phases (solid phase has `None`)
`phase_type`	List of characters describing type of phase: 'g' (gas), `'s'` (solid), `'l'` (liquid)	Only for `Reactor` nodes
`Q_balance_type`	Specifies calculation method for energy balance: one of `'isothermal'`, `'adiabatic'`, or `'Ucoeff'` (heat transfer coefficient, area, and external temperature specified)	Only for `Reactor` nodes
`cp`	List of heat capacity in each phase	Only for `Reactor` nodes
`Ucoeff`	Heat transfer coefficient for the reactor	Only for `Reactor` nodes, if `Q_balance_type` is `'Ucoeff'`

Text	External temperature for the reactor	Only for Reactor nodes, if Q_balance_type is 'Ucoeff'
Asurf	Heat transfer surface area	Only for Reactor nodes, if Q_balance_type is 'Ucoeff'
rtd	List of residence time distribution in each phase	Only for Reactor nodes, not presently used
rate_law	Function that can calculate species production/consumption rates and reaction enthalpy	Only for Reactor nodes

Table 4. Overview of stream properties and data structures.

Property	Description
stream_id	ID number of stream, as assigned by the library
name	Name of the stream, as assigned by the user in the model_input module
source_name	Name of the source node, as assigned by the user in the model_input module
source	ID number of the source node, as assigned by the library
sink_name	Name of the sink node, as assigned by the user in the model_input module
sink	ID number of the sink node, as assigned by the library
T	Temperature
P	Pressure
species	List of names of species in the stream
mass_fracs	Array of mass fractions of each species
mw	Array of molecular masses of each species
vol_flow	Volumetric flow rate
H	Array of enthalpies of each species
phase	ID number of phase (corresponds to the index of the phase in each Reactor)
rhop	Array of species densitiies (for liquid/solid phases, None for gas phase)
mu	Dynamic viscosity (gas/liquid phases, None for solid phases)
dp	Particle diameters (solid phases only)
phase_type	Character describing type of phase: 'g' (gas), 's' (solid), 'l' (liquid)
rho_avg	Average density
mass_flow	Mass flow rate

Table 5. Overview of network properties and data structures.

Property	Description
nodes	List of Node objects in the network
streams	List of Stream objects in the network
num_reactors	Number of Reactor nodes in the network (excludes Sources, Sinks, and Junctions)
num_streams	Number of Streams in the network
node_name_map	Dictionary giving the ID number of a node when given the user-assigned name
node_map	Dictionary giving the index of a node in the list of nodes when given the ID number of the node
stream_name_map	Dictionary giving the ID number of a stream when given the user-assigned name
stream_map	Dictionary giving the index of a stream in the list of streams when given the ID number of the stream

2.2 CHEMICS LIBRARY

The Chemics library is a collection of functions for calculating a variety of quantities that may be useful for implementing reactor sub-models. Currently, the library consists of the following: engineering correlations for fluidized reactors, gas phase physical property estimation functions, utility functions, and functions for calculating residence time distributions. The engineering correlations include a variety of functions for estimating minimum fluidization velocity, bubble diameter, transport disengagement, solids flux at the bed surface, and more. The gas phase physical property functions included thermal conductivity estimates for important species and the calculation of the mass density of the gas via the ideal gas law. The utility functions are for calculating average molecular weights, average densities, and for converting between mole and mass fractions. Residence time distributions are also available, but these are not currently used by Chemics-Reactors and are not discussed here.

2.3 SPECIFYING REACTOR NETWORKS

2.3.1 OVERVIEW

Reactor networks are specified by combining an arbitrary number of nodes and streams, each of which has defined properties. All properties of the nodes and streams are specified in the model_input module. The Chemics-Reactors driver program then reads the model_input module, creates lists of node and stream objects (one for each type), checks each node and stream for validity, and then assembles the final network and checks it for validity. Once the final network is assembled and validated, the Network object is returned to the driver program and all additional simulation is done by interacting with this object.

2.3.2 NETWORK TOPOLOGY

Reactor networks are specified by defining the properties of the nodes and streams in ordered dictionaries named nodes and streams, respectively. For convenience, we provide several functions in the reactor_models module that facilitate the construction of node and stream dictionaries for common reactor types (currently batch, semi-batch, and a series of tanks). Ordered dictionaries were chosen for the purpose of ensuring a repeatable (deterministic) user-defined order for iterating over the nodes and streams in the network. Keys to the nodes and streams ordered dictionaries are user-defined names. These

names should be meaningful to the user for purposes of model analysis. Note that the ordered dictionaries are used only in the `model_input` module as an easy way of specifying the input. The program reads these ordered dictionaries and constructs `Node` and `Stream` objects from them. The `Node` and `Stream` objects are then bundled together in a `Network` object.

When specifying nodes, the value associated with the key (node name) is an ordinary dictionary. All nodes have at least two required keys: `'type'` and `'pos'`, giving the node type (selected from `'source'`, `'sink'`, `'junction'`, and `'reactor'`) and physical coordinates. The physical coordinates are not currently used by Chemics-Reactors, but may be useful in the future or in user-defined routines. Reactor sub-units have two additional required keys: `'diam'` and `'height'`, specifying the diameter and height of the equivalent cylindrical tank. An example of defining a nodes dictionary for a single ideal tank reactor with an inlet and outlet is provided in Figure 3.

```
nodes = OrderedDict([
    ('in',   {'type': 'source', 'pos': [0.0, 0.0, 0.0]}),
    ('out',  {'type': 'sink', 'pos': [1.0, 0.0, 0.0]}),
    ('cstr', {'type': 'reactor',
              'diam': 1.0,
              'height': 1.0,
              'pos': [0.5, 0.0, 0.0]})
    ])
```

Figure 3. Code snippet for specifying the nodes of a single CSTR network in the model_input module. The 'pos' coordinates are not implemented in the current version of the framework but act as a placeholder for future code development.

Streams are specified by another ordered dictionary. Streams are key to defining which reactor sub-units are permitted to exchange mass with each other. To define a mass flow between two nodes, a corresponding stream must be defined by giving it a key (name) and a two element list with the names of the source and sink nodes. This two element list is stored in a sub-dictionary under a key named `'nodes'` (i.e., defining which nodes it connects). Streams are always unidirectional from source to sink; to specify bidirectional flow, simply specify a second stream with a unique name with the order of the source and sink nodes reversed. Streams must always have a node at each end; to satisfy this requirement for inlet and outlet streams, special source and sink node types have been defined. Source nodes have a single stream exiting the node, whereas sink nodes have a single stream entering the node. Junction nodes (representing mixing and splitting points) can also be defined. An example of defining the set of streams for our single tank reactor is shown in Figure 4. Network diagrams of some possible reactors are given in Figure 2.

```
streams = OrderedDict([
    ('gas_in',  {'nodes': ['in', 'cstr']}),
    ('gas_out', {'nodes': ['cstr', 'out']})
    ])
```

Figure 4. Code snippet for specifying the streams of a single CSTR network in the model_input module.

2.3.3 NODE PROPERTIES

`Source`, `Sink`, and `Junction` nodes do not have any mass inventory or physical properties associated with them. `Reactor` nodes, on the other hand, do have mass inventory and associated physical properties. Reactor properties are assigned to the dictionary associated with the reactor (i.e., in addition to the position, diameter, and height). Two of these parameters (temperature and pressure) are assigned directly to the dictionary via an associated key and are common to all phases in the reactor. Most of the remaining properties are phase-specific and are assigned to a common key as a dictionary of dictionaries. The outer dictionary is keyed by the phase number, and the inner dictionary is keyed by the corresponding physical property tags for that phase. The phase number is defined at the network level so that all phases in the network have a unique index. The gas phase should always be given the index 0 (other phase types may be in arbitrary order). Reactors do not have to have all phases in the network; it is permissible and frequently necessary for certain reactors to only have a one or two of the phases in the network (a common example would be a reactor with only gas in a multiphase network). In such a case, the properties of any network phases that are not present in a given reactor will be undefined. The phase type is determined by which of its properties are specified. The thermal behavior of the reactor is defined by a dictionary stored under the `'Q_balance_params'` key. This dictionary is used to define the type of energy balance as well as any additional parameters required by the energy balance calculation for the reactor. A full list of property names, their associated phases, and the Python object types used to define them is given in Table 3. An example of defining a two-phase (gas/solids) ideal tank reactor is shown in Figure 5.

```
nodes['cstr']['T'] = 573.0
nodes['cstr']['P'] = 101.325
nodes['cstr']['phases'] = {
    0: {'species': ['a', 'b', 'c'],
        'mass_fracs': [0.25, 0.75, 0.0],
        'mw': [8.0, 10.0, 12.0],
        'mu': 3e-5,
        'eps': 0.54},
    1: {'species': ['catalyst'],
        'mass_fracs': [1.0],
        'mw': [100.0],
        'dp': [0.005],
        'rho': 1.0e3,
        'eps': 0.46}
    }
```

Figure 5. Code snippet demonstrating how to specify the properties of a two phase (gas/solid) reactor. The phase type is determined automatically from the specified properties.

2.3.4 STREAM PROPERTIES

Stream properties are specified in much the same way as the node properties. The primary difference is that streams are phase pure, so all properties can be specified in a single flat dictionary. This properties dictionary is then stored as the value of the `'properties'` key in the sub-dictionary associated with the stream name. A list of all properties required to fully define a stream is given in Table 4, and an

example of a stream property declaration is given in Figure 6. In the example declaration, we assume that no solid streams are present.

```
streams['gas_in']['properties'] = {'T': 573,
                                   'P': 101.325,
                                   'vol_flow': 1.0,
                                   'species': ['a', 'b', 'c'],
                                   'mass_fracs': [0.25, 0.75, 0.0],
                                   'mw': [8.0, 10.0, 12.0],
                                   'mu': 3e-5,
                                   'phase': 0}
streams['gas_out']['properties'] = deepcopy(
    streams['gas_in']['properties'])
```

Figure 6. Code snippet for specifying the streams in the network for a single CSTR, assuming that the inlet and outlet properties are identical at the start of the simulation.

2.3.5 SIMULATION UNITS

Chemics-Reactors input units are in SI (m, kg, s) and are mass based. The only exceptions to this are pressure (kPa) and molecular masses (g/mol). A complete list of quantities and units is given in Table 6. Molar units may be used in the reaction rate calculation, but it is up to the user to make the proper unit conversions. Likewise, we place no restrictions on the units of the kinetic parameters or the units used in intermediate calculations in any of the user-defined functions. Our only requirement is that these functions accept and return quantities using the units used by the rest of the program.

Table 6. Units used by Chemics-Reactors in the `model_input` module and during simulations. See Tables 3 and 4 for where the quantities are employed in the library.

Quantity	Unit
Temperature	K
Pressure	kPa
Length (particle diameter, reactor dimensions, etc.)	m
Mass	kg
Density	kg/m^3
Dynamic Viscosity	kg/m/s
Mass Fraction	kg/kg
Volume Fraction	m^3/m^3
Time	s
Molecular Weight	g/mol
Mass Flow Rates	kg/s
Specific Enthalpy	J/kg
Heat Capacity	J/kg/K
Heat Transfer Coefficient	$W/m^2/K$
Heat Transfer Area	m^2
Species Production Rate	kg/s
Reaction Enthalpy Rate	J/s

2.4 SOLUTION OF MODEL BALANCE EQUATIONS

For each reactor sub-unit, we need to solve a number of balance equations. These balances are defined by various methods in the `Network` object in the `reactor_network` module. The balance equations solved by the program are summarized in Table 7.

Table 7. Summary of balance equations and the corresponding method in the `Network` object responsible for evaluating the associated time rate of change.

Balance	Method Name
Total Mass	get_mass_change
Species Mass Fraction	get_species_mass_frac_change
Temperature	get_T_change

For multiphase reactors, the material balances must also account for interphase reactions or transfer of material from one phase to another. For species i, phase j, inlet stream k, and outlet stream k', the transient (differential) total mass balance can be written:

$$\frac{dm_j}{dt} = \sum_k \dot{m}_{jk} - \sum_{k'} \dot{m}_{jk'} + \sum_i M_i r_{ij}$$

Here we have chosen to work with mass rather than moles because it simplifies the balances somewhat for single phase reactors (in such cases the reaction source term is identically zero, assuming there are no nuclear reactions). The mass in the reactor is m_j and the inlet and outlet mass flow rates are m_{jk} and $m_{jk'}$,

respectively. Currently mass flow rates must be specified in the input or explicitly calculated by the user in one of the user-defined model functions (discussed later); future versions of Chemics-Reactors may relax this requirement by developing advanced algorithms for calculating mass flow rates that are entirely internal to the network (i.e., not connected to one of the source nodes). The species production rates r_{ij} are in molar units and are converted to mass units by multiplication by the species molecular mass M_i. Species production rates can be due to chemical reactions or simple mass transfer (e.g., due to absorption of a gaseous species by a liquid or adsorption of a gas on a solid). For individual species in a given phase, we solve the mass balances in terms of the mass fraction (other choices could be made as well, but mass fractions are consistent with our choice to work with mass, rather than moles, when solving the overall material balance). The resulting balance in terms of mass fractions (w_{ij}, for the reactor and outlet streams, as all outlet streams have the same composition as the reactor for an ideal continuous tank reactor, and w_{ijk} for inlet streams) is:

$$\frac{dw_{ij}}{dt} = \frac{1}{m_j}\left[\sum_k (w_{ijk} - w_{ij})\dot{m}_{jk} + M_i r_{ij} - w_{ij}\sum_i M_i r_{ij}\right]$$

Besides the material balances, we must also solve the energy balance. This is solved in terms of temperature assuming that the total masses and heat capacities are constant:

$$\frac{dT}{dt} = \frac{\sum_j \left[\sum_k \dot{m}_{jk}\sum_i w_{ijk}H_{ijk} - \sum_{k'} \dot{m}_{jk'}\sum_i w_{ijk'}H_{ijk'}\right] + \Delta H_{rxn} + Q}{\sum_j m_j c_{P,j}}$$

where H_{ijk} is the mass specific enthalpy of species i in phase j and inlet stream k ($H_{ijk'}$ is similar but for outlet stream k'), $c_{P,j}$ is the heat capacity of phase j in the reactor, ΔH_{rxn} is the total reaction enthalpy for all reactions in all phases, and Q is the total amount of sensible heat evolved. The amount of heat evolved can currently be calculated in two ways: as adiabatic (i.e., $Q = 0$) or with an overall heat transfer coefficient:

$$Q = UA(T - T_{ext})$$

where U is the heat transfer coefficient, A is the heat transfer area, and T_{ext} is the externally applied temperature for heat transfer. It is also possible to specify isothermal operation, in which the temperature derivative is explicitly set to 0.

These balance equations may be solved by a variety of numerical solvers. We have chosen to implement a generic wrapper class `ode_integrator` that can interface with the numerical solver of the user's choice. We have also implemented functions specifically tailored to the VODE solver package (which has been wrapped in the SciPy library). In particular, we provide functions for extracting the current solution vector (named `vode_y`) and the values of the right hand sides of the differential equations (`vode_rhs`) for use by the integrator. In the VODE solution vector, the masses of each phase in each reactor are listed first, followed by the mass fractions of each species in each phase of each reactor, and the temperatures of each reactor are listed last. Solver-specific options are specified in the `model_input` module via the `solver_settings` dictionary, and the solver is selected by assigning a value to the `solver` variable (currently only the value `'vode'` is allowed). The start time (`t_initial`), stop time (`t_final`), and time increment for each call to the ODE solver (`t_res`) are also specified in the `model_input` module.

2.5 STATISTICAL MODEL ANALYSIS

2.5.1 OVERVIEW

One of the key sets of features implemented in Chemics-Reactors is the ability to perform large numbers of calculations for the purpose of efficiently screening for optimal operating conditions (Design of Experiments), understanding how parametric uncertainty impacts the variability of model predictions (uncertainty quantification), and identifying which parameters have the most impact on prediction variability (sensitivity analysis). If one of the statistical analysis features is selected, then a `uq_gsa_seq` object is created. This class has functions for setting up a trajectory of calculations, sampling the parameter space, setting model parameters, and calculating metrics for the specified trajectory type (typically only employed by a separate user-supplied post-processing program).

These features are all controlled by the `uq_gsa_settings` dictionary in the `model_input` module and auxiliary dictionaries that specify allowable parameter ranges (whether for operating conditions, kinetic parameters, or internal mass flow rates). The type of analysis is set by specifying the value of the `'type'` key (allowed options are given in Table 8), and the number of replicates is specified by assigning an integer value to the `'replicates'` key. Random seeds for methods requiring Monte Carlo sampling (uncertainty quantification and global sensitivity analysis) are specified by assigning a list to the `'rand_seed'` key. Granular control of the output is achieved by specifying which reactors or streams are of explicit interest in a dictionary of lists (assigned to the key `'output'`). This sub-dictionary has two keys: `'Reactor'` and `'Stream'`, and the lists assigned to each are the names of the reactors or streams to write output for. The base names (i.e., without extensions) of the output files are set by strings specified in a similar dictionary assigned to the key `'output_filenames'`.

Table 8. Summary of how to specify advanced model analysis options.

Desired Analysis	Value to Assign `'type'` Key
None	`None`
Design of experiments	`'DOE'`
Uncertainty Quantification	`'UQ'`
Global Sensitivity Analysis	`'VGSA1'` (only first order and total effects) or `'VGSA2'` (also generates second order effects)

All of these features require defining a distribution of allowable values. To do this, we have chosen to define a named tuple called `param_dist`. This named tuple has four fields: '`base`' (for the base value), '`a`' (for the low value of uniform or log-uniform distributions or the mean for normal or log-normal distributions), '`b`' (for the high value of uniform or log-uniform distributions or the standard deviation for normal or log-normal distributions), and '`type`' (specifying the type of distribution). Allowed distributions are uniform (`'u'`), log-uniform (`'lu'`), normal (`'n'`), and log-normal (`'ln'`). The named tuple and all distribution and parameter dictionaries for design of experiments, uncertainty quantification, and global sensitivity analysis are stored in the `params` module.

Any settings for post-processing the raw results should also be set in the `uq_gsa_settings` dictionary. As it is not possible to anticipate every possible form of post-processing, we do not discuss such options here (we have written a post-processing code for our own use, but we expect that every user will need to write one for his/her specific use case). To facilitate Monte Carlo sampling of many thousands (or more) of parameter combinations on supercomputing clusters, we have implemented very basic parallelization with `mpi4py` that will start independent trajectories (to turn this on, set `use_mpi` to `True` in the `model_input` module).

2.5.2 DESIGN OF EXPERIMENTS

The design of experiments feature is designed to efficiently sample the operating space (currently this only works with inlet stream properties) to identify the key parameters controlling the model predictions of interest (e.g., conversions, selectivities, yields, etc.). This feature relies on two external packages: `pyDOE` and `sobol_seq`. The package `pyDOE` creates full factorial, fractional factorial, Plackett-Burman, Box-Behnken, central composite, and Latin hypercube designs in coded units. The package `sobol_seq` implements the Sobol' low discrepancy sequence for quasi-random sampling, also in coded units. The coding for Latin hypercube and the Sobol' sequence map points to the unit hypercube, whereas the coding for the other design methods are with respect to a specified center point. Options for the design methods are specified by creating a dictionary assigned to the `'doe_args'` key of the `'DOE_design'` key in the `uq_gsa_settings` dictionary. The `'doe_args'` dictionary has two keys: `'type'` (which specifies the design type, with possible options given in Table 9) and `'args'`, which is a dictionary of keyword arguments that is passed unchanged to the underlying routine. The arguments for the `pyDOE` designs are outlined in the documentation for the package. The only required argument for the Sobol' sequence is the number of samples (stored in key `'samples'` of the `'args'` dictionary), which specifies the total number of points to sample in the n-dimensional space. After generation of the coded design matrix, the program automatically translates it into a series of operating conditions in real units.

Table 9. Summary of how to specify design of experiments design types.

DOE Design Option Tag	Design Type
`'full2'`	Full, two level factorial
`'frac'`	Fractional factorial (requires a special, user-provided alias structure)
`'pb'`	Plackett-Burman
`'bb'`	Box-Behnken
`'cc'`	Central Composite
`'lh'`	Latin Hypercube
`'sob'`	Sobol' sequence

The operating space parameters should be specified by assigning `param_dist` named tuples to an ordered dictionary named `op_space_params_dist` (in the `params` module) whose keys are space delimited strings defined by up to three fields. The first field is the name of the stream, the second field is the type of stream property to vary, and if the stream property depends on the species, the third field is the name of the species whose property is to be varied (otherwise there is no third field). If the species property that varies is a mass fraction, then the first mass fraction in the list of species mass fractions is always constrained to make the sum of mass fractions equal unity. A dictionary named `op_space_params` with keys corresponding to the `op_space_params_dist` ordered dictionary and values corresponding to the current parameter values as set by the design of experiments matrix also needs to be defined in the `model_input` module. This dictionary acts as the interface between the uncertainty quantification/global sensitivity analysis module and the user-supplied mass flow rates function, permitting the inlet stream properties to be altered at arm's length in the user-supplied function.

2.5.3 UNCERTAINTY QUANTIFICATION

Uncertainty quantification calculations may be performed for quantities in any user-defined function. Since the possibilities are endless, the uncertainty quantification code is written to be extremely generic. The only requirement is that any distribution dictionaries are stored in a container dictionary named `param_dists` with keys that are accessible to the user-supplied functions and that correspond to

identically named dictionaries in the `model_input` module (these dictionaries contain the current values sampled from the corresponding distribution dictionaries, in analogy with the design of experiments `op_space_params_dist`/`op_space_params` pair of dictionaries in the `params` module). As a special feature, uncertainty quantification is automatically performed for design of experiments runs if the total number of replicates is greater than one, and a single replicate is always performed at the baseline parameter set (for parallel runs this is the first replicate on the first MPI rank). Post-processing codes can use the special `uq_gsa_seq` method `calculate_UQ_metrics` to calculate quantiles of a user-supplied data set.

2.5.4 GLOBAL SENSITIVITY ANALYSIS

Inputs for the global sensitivity analysis feature are essentially identical to those for the uncertainty quantification feature. We have implemented a variance-based global sensitivity analysis, which is essentially a set of numerical experiments on the chosen set of parameters designed to extract information about which parameters contribute most to the total variance. We have constructed our parameter sampling to use the estimators of Saltelli and coworkers.[1-3] We can compute three different global sensitivity indices: the first order indices (S_i), the total effect indices (S_i^T), and optionally the second order indices (S_{ij}). The first order indices give information on linear contributions of each parameter to the total variance, the second order indices give information on the contributions of two parameter interactions, and the total effect indices give information on the contribution of each parameter on its own and in conjunction with all other parameters to the total variance. If requested, the second order indices (giving insight into two-parameter interactions) require slightly less than double the number of sample points as the reduced set giving only the first order and the total effect indices ($2n$+2 vs. n+2, where n is the number of parameters), but also gives double estimates of each of the sensitivity metrics, so the cost tradeoff is a good one.

The global sensitivity analysis requires two independent parameter vectors denoted **A** and **B**. The algorithm for the first order and total effect indices requires model solutions at these points as well as using vectors constructed by successively substituting the value of the i^{th} parameter from the second vector for the corresponding value in the first vector. For example, if the first vector is **A**, and the second parameter is the one being perturbed, then the current vector is the **A** vector with its second element replaced by the second element from the **B** vector. We denote this substitution as the $\mathbf{A}_{\mathbf{B}i}$ vector. If the first vector is the **B** vector, then the substitution comes from the **A** vector, and the substituted vector is the $\mathbf{B}_{\mathbf{A}i}$ vector. The second order indices do not use the **A** and **B** vectors directly but rather a set of three specially selected $\mathbf{A}_\mathbf{B}$ and $\mathbf{B}_\mathbf{A}$ vectors for each estimate of the second order indices. For the first estimate, one of these is an $\mathbf{A}_\mathbf{B}$ vector and the other two are $\mathbf{B}_\mathbf{A}$ vectors, whereas for the second estimate, one is a $\mathbf{B}_\mathbf{A}$ vector and the others are $\mathbf{A}_\mathbf{B}$ vectors. As an example, in the first estimate, if the first parameter in the two-term interaction is i and the second is j, the $\mathbf{A}_\mathbf{B}$ vector is chosen such that the j^{th} element from the **B** vector is substituted into the **A** vector ($\mathbf{A}_{\mathbf{B}j}$). The $\mathbf{B}_\mathbf{A}$ vectors are chosen such that the **B** vector has either the i^{th} or the j^{th} elements from the **A** vector are used (i.e., generating both $\mathbf{B}_{\mathbf{A}i}$ and $\mathbf{B}_{\mathbf{A}j}$). For both estimates, a total of four vectors are thus needed, but only three of them are used in each estimator. A complete set of solutions at all of the parameter vectors constitutes a single replicate block. Well-converged sensitivity indices typically require thousands of such blocks.

Once the raw data are collected, post-processing is required to estimate the sensitivity indices. Denoting $f(.)$ as the value of the function giving the model prediction of some quantity for a given parameter set, the post-processing code first estimates the partial variances. For the first order indices, we need the estimators in Table 10 (notice the parallel structures of the two independent estimates). The total variance is calculated from the model predictions taken only from the corresponding vector. We note that we have modified the estimators for the second order indices slightly to only use a normalization factor of n. The original reference[1] uses mixed normalization factors (n-1 in one term and n in the second term). The n-1

normalization factor is related to eliminating bias in the estimate and is only important for small samples. In the limit of a large number of samples, the bias correction is negligible, and we prefer the simpler structure of the estimator that arises when employing the same normalization factor. We also note that the estimators for the second order indices are given very tersely as a passing comment in the original reference[1], and the estimators given here represent our best effort to expound on that work given what we know about the principles of global sensitivity analysis; it is possible we have made mistakes in deriving formulas for the second order indices.

If calculating only the first and total effect indices with the reduced set of model solutions (option `'VGSA1'`), only the first estimates are calculated, and it is not possible to calculate second order indices. The partial and total variances calculated via the formulas in Table 10 are then converted into sensitivity indices with the formulas in
Table 11 and the double estimates of the sensitivity indices are further averaged to produce a single metric.

Table 10. Estimators required for the calculation of global sensitivity indices. The total variance for each sensitivity index should be estimated from model predictions corresponding to the specified vector.

Sensitivity Index	Partial Variance Formula	Vector for Total Variance
First order, estimate 1	$V_i = \frac{1}{n}\sum_{r=1}^{n} f(\mathbf{B})\left[f(\mathbf{A_{Bi}}) - f(\mathbf{A})\right]$	**B**
First order, estimate 2	$V_i = \frac{1}{n}\sum_{r=1}^{n} f(\mathbf{A})\left[f(\mathbf{B_{Ai}}) - f(\mathbf{B})\right]$	**A**
Total effect, estimate 1	$V_{-i} = \frac{1}{2n}\sum_{r=1}^{n} \left[f(\mathbf{A}) - f(\mathbf{A_{Bi}})\right]^2$	**A**
Total effect, estimate 2	$V_{-i} = \frac{1}{2n}\sum_{r=1}^{n} \left[f(\mathbf{B}) - f(\mathbf{B_{Ai}})\right]^2$	**B**
Second order, estimate 1	$V_{ij}^c = \frac{1}{n}\sum_{r=1}^{n} f(\mathbf{A_{Bj}})\left[f(\mathbf{B_{Ai}}) - f(\mathbf{B_{Aj}})\right]$	$\mathbf{A_B}$
Second order, estimate 2	$V_{ij}^c = \frac{1}{n}\sum_{r=1}^{n} f(\mathbf{B_{Ai}})\left[f(\mathbf{A_{Bj}}) - f(\mathbf{A_{Bi}})\right]$	$\mathbf{B_A}$

Table 11. Formulas for calculating individual estimates of the sensitivity indices. If double estimates of the sensitivity indices are available, the best value of the sensitivity index is the arithmetic mean of the two estimates.

Sensitivity Index	Formula
First order	$S_i = \frac{V_i}{V}$
Total effect	$S_i^T = 1 - \frac{V_{-i}}{V}$
Second order	$S_{ij} = \frac{V_{ij}^c - V_i - V_j}{V}$

2.6 MODEL OUTPUT

Steady state results for the baseline parameter set for all reactors and streams are routinely generated for all simulations. These results include temperatures, pressures, compositions, flow rates (streams) or mass inventories (reactors), and phase properties. Transient results are also available (in an unformatted text file) for the baseline model. These results are primarily intended for use in determining when steady state has been reached is simply the solution vector obtained by VODE (or other integrator) as a function of time. The exact order of the output values are determined by the structure of the solution vector outlined in Section 2.4.

If an advanced model analysis feature is selected (design of experiments, uncertainty quantification, or global sensitivity analysis), then additional output is also generated for later post-processing. This additional output is only generated for the specifically requested streams and reactors in order to reduce the amount of disk space required. The output is written to disk in a tabular format. Each row in the table corresponds to a different parameter set, and the parameter names and values precede the results of the model for each reactor or stream in the network. For reactors, the results include the temperature and pressure of the reactor followed by the total mass and composition (in mass fractions) of each phase in the reactor, and these results are grouped by column for each requested reactor. The stream data follows a format similar to the reactor data, except that mass flows are reported instead of mass inventories, and the phase purity of the streams means that the output is organized purely by stream rather than by stream and by phase. For convenience in the post-processing stage, the same data written to the text files is also saved as `NumPy` binary files.

The names of the output files can be specified for both the routine and specialized output. For the routine output, values need to be set in a dictionary assigned to the variable `output_filenames` under the keys `'Reactor'` (reactor results), `'Stream'` (stream results), and `'ty'` (for the transient t, **y** data). The filenames of the advanced model analysis results are set in a dictionary in the `uq_gsa_settings` dictionary variable under the key `'output_filenames'`. The keys for this dictionary are `'Reactor'` and `'Stream'`, with the same meaning as for the routine output (there is no transient data for the advanced model analysis output).

2.7 USER-SPECIFIED FUNCTIONS

2.7.1 OVERVIEW

Although Chemics-Reactors is designed to be general purpose modeling software, it requires the user to supply certain custom functions and permits it in other cases. These functions should be written in the `model_funcs` module (replacing the existing functions in that module) or they may be written in a separate module and assigned in the model_input module to the appropriate variable in the `model_funcs` module (by calling the Python function `setattr`). The two required functions are for the reaction rates (named `rate_law`) and for the inter-node mass flow rates (named `get_mass_flow_rates`). Minimal working (do-nothing) examples of the required functions are provided in the `model_funcs` module to provide a template for further customization. The optional function is for an equation of state for calculating the density of a fluid.

2.7.2 REACTION RATES

The first user-defined function that should be supplied is for calculating the reaction rates. This function takes the temperature, pressure, reactor volume, a list of the molecular mass arrays for each phase, a list or array of the masses of each phase, a list of arrays of the mass fractions for each phase, a list or array of

the volume fractions of each phase, a list of the arrays of the particle diameters for each phase, and a list or array of the dynamic viscosities of each phase. It should return a list of arrays with the species production (positive) or consumption (negative) rates for each phase, and the overall enthalpy of reaction. Kinetic parameters may be stored in the `params` module if desired.

2.7.3 INTER-NODE CIRCULATION RATES

The second required user-defined function is for calculating the mass flow rates between nodes in the network. The only argument to this function is the reactor network object. This function has no return values; instead it sets the mass flow rates and properties for each stream directly based on any constraints imposed by the user. It might be possible to replace this user-defined function with a more sophisticated algorithm that could iteratively solve for a set of self-consistent flow rates. Such an algorithm, while desirable from a user-convenience perspective, is outside the scope of the current work. As with the rate expression, any parameters required by this user-defined function may be stored in the `params` module (and any parameters that should be altered during the simulation must be stored there).

2.7.4 EQUATION OF STATE

The third (and optional) user defined function is for an equation of state for calculating the density of a gas or liquid. This function should take as arguments the temperature, the pressure, the mass fractions, and the molecular masses of the species. Any number of equations of state can be defined and used. The only requirement is that the name of the appropriate function is assigned as a value in the phase properties dictionaries in the `model_input` module for each stream or reactor under the key `'eos'`. The program will then assign the correct functions to the underlying reactor or stream objects.

2.8 EXPERIMENTAL FEATURES AND UNTESTED CODE

We would like to note that certain implemented features have not been tested and should be considered experimental. While we believe the functionality is correct, undiscovered bugs could exist. In particular, we have not tested any of the liquid phase functionality, the equation of state feature, or the non-isothermal energy balances. In the `Reactor` class, we permit the definition of a residence time distribution (RTD) associated with the rate law. However, we have not exposed this feature to the user, and it is not currently employed in any code outside this class. Exposing this feature to the user should be fairly straightforward; simply add the RTD function as an argument to the `create_nodes` function. Any code utilizing an RTD would also have to be added by the user.

3. CASE STUDY: MODELING BIO-OIL UPGRADING IN A BUBBLING FLUIDIZED BED REACTOR

3.1 OVERVIEW

As an example of a real problem that is being solved with Chemics-Reactors, we present a model of a bench-scale bubbling bed reactor at the National Renewable Energy Laboratory (NREL). This reactor is intended to generate laboratory data for developing process economics models, exploring new catalysts, and identifying optimal operating conditions and strategies.[4-6] We are employing Chemics-Reactors to model this reactor with the intent of understanding its operational behavior in order to guide experimental design and interpretation.

3.2 METHODS

3.2.1 NETWORK MODEL

The physical reactor has a gas distributor at the bottom of the bed, receives fresh solid catalyst feed above the bed, removes used catalyst via spillover from the bed (the solid catalyst feed location is above the solid catalyst removal location), and a large section of freeboard above the fresh catalyst feed so that the solids ejected from the bed will disengage and fall back into the bed rather than elutriate. The disengagement is promoted by the freeboard having a larger diameter than the bed. The vapor phase products are drawn off through an outlet at the top of the freeboard. The conical region where the fresh catalyst enters and the spent catalyst exits is referred to as the splash zone. Physical dimensions for the reactor zones are given in Table 12.

Table 12. Geometric dimensions of the bubbling bed reactor.

Reactor Zone	Diameter (m)	Height (m)
Bed	0.052	0.1245
Splash	0.052 (bottom) 0.078 (top)	0.0155
Freeboard	0.078	0.3820

We have chosen to model the physical reactor with a network of four ideal reactors and several gas and solids streams circulating between reactors in the network. The placement of the reactors and streams are presented in Figure 7. There are three major zones in the reactor: the bed, the splash zone, and the freeboard. In the network model, the bed is divided into two reactors: one with only gas (labeled ‘bubbles’ and one with both gas and solids (labeled ‘emulsion’). The ‘bubbles’ reactor represents the gas in bubbles that bypasses the solid catalyst and thus does not react. The ‘emulsion’ reactor represents the dense mixture of gas and solid catalyst where reaction can occur. Above the bed, the splash zone is represented by a single reactor with a fresh solid catalyst feed stream and a spent solid catalyst exit stream. The freeboard is above the splash zone and has the exit stream for the gas as well as a secondary exit stream for the solids to quantify possible elutriation rates. Solid catalyst is permitted to travel between the emulsion and the splash zone and between the splash zone and the freeboard. Gas generally flows upward from the distributor to the freeboard. However, there are two junctions present in order to capture the behavior of two different bubbling regimes. One junction is at the bottom of the bed (referred to as the distributor junction), and the other is between the bed and the splash zone (referred to as the splash junction). During slow bubbling, the distributor junction at the bottom of the bed acts as a splitter, sending some gas to the bubbles and some gas to the emulsion. These two gas streams then rejoin at the splash junction before traveling upwards through the splash zone and the freeboard to the gas exit. In the case of fast bubbling, the gas flow in the emulsion will reverse and travel downwards toward the distributor. The distributor junction then acts as a mixer, mixing converted gas with fresh feed gas before sending it to the bubbles reactor. Likewise, the splash junction acts as a splitter, recycling some of the bubble gas back to the emulsion.

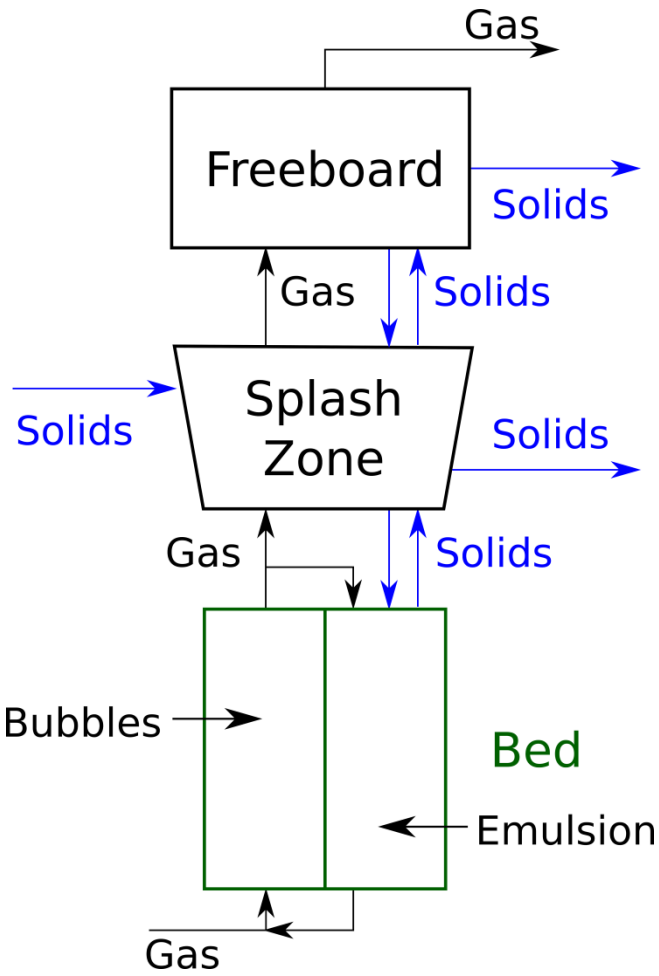

Figure 7. Diagram of the network model employed for the bubbling bed reactor simulation. The configuration depicted is for fast bubbling where gas flows downward through the emulsion. Slow bubbling is also possible, and our model handles this by reversing the direction of the gas flow through the emulsion.

3.2.2 CORRELATIONS AND CLOSURES OF FREE MODEL PARAMETERS

As it stands, the network model has a number of free parameters that must be closed. These fall into several categories: feed conditions, solid catalyst volume fractions, internal gas flow rates, and internal catalyst flow rates. The solid catalyst volume fractions were set from known experimental trends, and the internal flow rates were estimated via literature correlations, primarily appearing in *Fluidization Engineering,* 2nd ed., by Kunii and Levenspiel (hereafter referred to as KL).[7] The flow rates are specified in the `get_mass_flow_rates` function in the `model_funcs` module.

Feed conditions (flow rates, compositions, etc.) were provided by collaborators at NREL for a typical setup employing fresh pyrolysis vapors and HZSM-5 catalyst. Catalyst properties were measured by collaborators at the National Energy Technology Laboratory (NETL). They are summarized in Table 13.

Table 13. Inlet flow rates and material properties.

Property	Value
Gas temperature	773 K
Gas pressure	1 atm
Gas flow rate	20 L/min at STP
Gas composition	10 wt% bio-oil, 90 wt% N_2
Solids temperature	773 K
Solids flow rate (pure fresh HZSM-5)	300 g/hr
Solids mean particle diameter	599 μm
Solids density	1.19 g/cm^3
Solids mean particle sphericity	0.83

The volume fraction of solids at minimum fluidization in the emulsion was determined experimentally by collaborators at NETL to be 0.46. The solids volume fractions in the splash zone and freeboard were set to 0.30 and 0.01, respectively, in agreement with typical values reported in the literature.[7]

The internal gas mass flow rates were estimated with a combination of literature correlations and experimental data. The minimum fluidization velocity was determined experimentally at NETL to be 10.11 cm/s. The bed geometry (bed height, bed diameter, and distributor plate orifice diameter) were used with the Mori-Wen correlation (KL equations 5.15, 5.19, 6.4, and 6.5) to estimate the effective diameter of the bubble phase at both the midpoint and top of the bed. First the bubble diameter at the distributor is estimated with KL equations 5.15 and 5.19. For slow bubbling (determined by an initial bubble size smaller than or equal to the spacing between orifices in the distributor plate) KL equation 5.15 is used. This equation is

$$d_{b0} = \frac{1.30}{g^{0.2}}\left[\frac{u_0 - u_{mf}}{N_{or}}\right]^{0.4}$$

where g is the acceleration due to gravity, u_0 is the superficial velocity, u_{mf} is the minimum fluidization velocity, and N_{or} is the orifice number, which is $\frac{1}{l_{or}^2}$ for a square array of holes in the distributor and $\frac{2}{\sqrt{3}l_{or}^2}$ for an equilateral triangular array of holes, with the spacing between the holes given by l_{or}. For fast bubbling (initial bubble size larger than the spacing between the orifices), KL equation 5.19 is used. This equation is

$$d_{b0} = \frac{2.78}{g}(u_0 - u_{mf})^2$$

with the same nomenclature as the preceding equation. Then the maximum bubble diameter is calculated with KL equation 6.5, which is

$$d_{bm} = 0.65\left[\frac{\pi}{4}d_t^2(u_0 - u_{mf})\right]^{0.4}$$

where d_t is the diameter of the bed. The bubble diameter d_b at a specified height z is then calculated with KL equation 6.4

$$\frac{d_{bm} - d_b}{d_{bm} - d_{b0}} = e^{-0.3z/d_t}$$

In these equations, distances are in cm and velocities are in cm/s. Effective bubble diameters were constrained to be no more than 80% of the bed diameter as this represents the maximum bubble diameter achievable under slugging conditions.

These bubble diameters were then used in conjunction with the bubble position, superficial velocity in the bed, and the minimum fluidization velocity to estimate the bubble rise velocity (KL equations 6.7 and 6.8). Combining these two equations yields an estimate of the bubble velocity u_b

$$u_b = u_0 - u_{mf} + 0.711\sqrt{(gd_b)}$$

This velocity was then used to calculate the mass flow rate of gas passing through the bubble phase. The mass flow rate of gas through the emulsion was then calculated by enforcing mass conservation on the total mass flow of gas entering the distributor junction. If the bubble gas flow rate were below the inlet gas flow rate, then bubbling was slow, and the distributor junction acted as a splitter. Otherwise, the bubbling was fast, and the distributor junction acted as a mixer (i.e., emulsion gas was traveling downward). The relative sizes of the bubble and emulsion phase were determined by calculating the fraction of the bed volume in the bubble phase with KL equations 6.26-6.29. To calculate the fraction of the bed δ that is in the bubble phase, we first determine the bubbling regime by calculating the ratio of the bubble velocity to the emulsion gas velocity: $\frac{u_b \varepsilon_{mf}}{u_{mf}}$ (we call this the bubbling ratio). This ratio indicates the bubbling regime; slow bubbling occurs when the ratio is less than unity, intermediate bubbling occurs when the ratio is from one to five, fast bubbling occurs from five to ten, and very fast bubbling is when the ratio is more than ten (see KL chapter 6 for a deeper discussion of the bubbling regimes). The equations used for calculating the bubble fraction are given in Table 14. The gas flow rates through the splash zone and freeboard were explicitly set to be equal to the gas mass flow rate in the inlet (i.e., assuming a constant mass flow of gas through the reactor).

Table 14. Summary of equations used to estimate the fraction of the bed in the bubble phase (see KL chapter 6).

Regime	Bubbling Ratio	Bubble Fraction δ
Slow	<1	$\frac{u_0 - u_{mf}}{u_b + 2u_{mf}}$
Intermediate, Slow	~1	$\frac{u_0 - u_{mf}}{u_b + u_{mf}}$
Intermediate, Fast	~5	$\frac{u_0 - u_{mf}}{u_b}$
Fast	5-10	$\frac{u_0 - u_{mf}}{u_b - u_{mf}}$
Very Fast	>10	$\frac{u_0}{u_b}$

The internal solids mass flow rates were estimated with a combination of flux matching at the bed surface and transport decay theory in the freeboard. The bubble diameter, bed diameter, minimum fluidization velocity, superficial velocity, gas density, and gas viscosity were used in a modified version of the Wen-Chen correlation (ORNL TM-7847 by Wells, Kulver, and Krishnan) to estimate the flux of solids being ejected upwards from the bed surface into the splash zone. This correlation is

$$F_0 = 3.07 \times 10^{-9} \frac{\frac{\pi d_t^2}{4} d_b (u_0 - u_{mf})^{2.5} \rho_g^{2.5} \sqrt{g}}{\mu_g^{2.5} u_{mf}}$$

For wide, shallow beds (i.e., where the depth of the bed is less than the bed diameter), the flux F_0 is multiplied by $\left(\frac{d_b}{d_t}\right)^2$. This correlation uses SI units in all quantities.

The solids mass flow rate back into the bed from the splash zone was assumed to equal the mass flow rate of solids leaving the bed surface. The mass flow rate of solids from the splash zone to the freeboard was assumed to decrease exponentially with height above the bed surface *z* according to the equation

$$\dot{m}_z = \dot{m}_0 e^{-az}$$

where *a* is the transport decay constant, $\dot{m}_z$ is the mass flow rate at height z, and $\dot{m}_0$ is the mass flow rate at the bed surface. The transport decay constant for estimating the fraction of solids of diameter d_p remaining fluidized was calculated with a curve fit to the data reported in Figure 7.12 of *Fluidization Engineering*[7] and the superficial velocity in the freeboard. The equation for the curve fit that was used is

$$au_0 = 0.03918 d_p^{0.65883}$$

The transport decay constant is in m^{-1}, the superficial velocity is in m/s, and the particle diameter is in μm. The power law fit is valid for particle diameters in the range of 50 μm to 800 μm. The total solids mass flow rate was then the fraction of solids remaining fluidized multiplied by the solids mass flow rate at the bed surface. As before, the mass flow rate of solids back into the splash zone from the freeboard was assumed to match the mass flow rate of solids from the splash zone to the freeboard. The solids elutriation rate was calculated similarly to the mass flow rate of solids from the splash zone to the freeboard, except the height of the gas outlet above the bed surface was used instead of the height of the transition from the splash zone to the freeboard. The total mass flow rate of solids leaving the splash zone was set to enforce no solids mass accumulation in the reactor (i.e., by subtracting the solids elutriation mass flow rate from the inlet solids mass flow rate).

3.2.3 KINETIC SCHEME

We employed a kinetic scheme from the literature for this work.[8] This scheme was originally developed for bio-oil produced via high pressure liquefaction and is the most relevant model that we have yet found. It is sufficient for initial screening and code demonstration purposes, but we recommend that data for constructing a kinetic model of fast pyrolysis bio-oil should be collected as soon as possible.

The kinetic scheme is summarized in Figure 8. All reactions are considered to be unimolecular (something of a misnomer since the 'species' represent groups of similar compounds, rather than chemically pure species) and irreversible. Raw oil can decompose into nonvolatiles (presumably heavy tars) that further convert into coke (carbonaceous material left on the catalyst), residue (carbonaceous material deposited on the reactor walls, as char, etc.), or volatile compounds (presumably heavy oxygenates such as sugars and polyols). Raw oil can also decompose into volatiles that further convert into additional fractions. The undesired products are light gases, coke, residue, and an aqueous phase with water and water-soluble compounds, whereas the desired products produced by the volatile species are organic distillates (mainly polyaromatic phenolics) and hydrocarbons (mainly polyaromatic

hydrocarbons). Molecular weights for the various bio-oil species were taken from an appendix of the PhD thesis of J.D. Adjaye. We believe that the volatile compounds have the highest similarity to fast pyrolysis bio-oil, and we therefore assume that fast pyrolysis bio-oil is equivalent to the volatiles fraction in the discussion that follows. We also assumed first order kinetics for converting the fresh catalyst into deactivated (coked) catalyst.

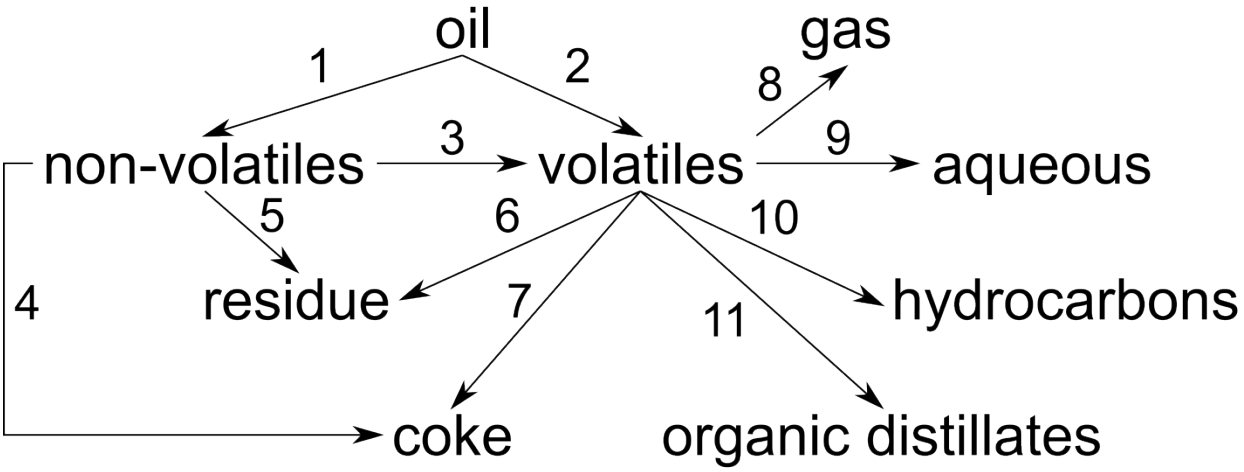

Figure 8. Kinetic scheme for vapor phase upgrading high pressure liquefaction bio-oil on HZSM-5. The catalyst deactivation reaction is not shown in the mechanism.

The rate constants at the operating temperature were estimated by combining the reported activation energies for each reaction with the rate constants at the highest available temperature (410 °C) and extrapolating to the operating temperature of 500 °C. The temperature was assumed to have a negligible effect on the reaction orders, so the reaction orders at the highest available temperature were used without change. The rates of the upgrading reactions 1-11 were artificially increased by a factor of 10^4 in order to obtain a baseline volatiles conversion of ~30%. The deactivation rate constant was tuned to provide approximately 10 wt% coked catalyst in the splash zone solids exit (this degree of coking was suggested by NREL as a typical value). The baseline kinetic parameters are provided in Table 15 (note that the parameters for reactions 1-11 do not reflect the artificial increase by a factor of 10^4).

Table 15. Baseline kinetic parameters for bio-oil upgrading scheme. The rates of reactions 1-11 were increased by a factor of 10^4 above the baseline values in order to achieve moderate levels of volatiles conversion.

Reaction Number	Reaction	Rate Constant	Reaction Order
1	oil → non-volatiles	6.33E+01	1
2	oil → volatiles	1.10E+01	1
3	non-volatiles → volatiles	6.85E-05	0.9
4	non-volatiles → coke	1.03E-04	1.1
5	non-volatiles → residue	3.35E-02	2.5
6	volatiles → residue	1.29E-03	0.7
7	volatiles → coke	2.62E-04	1.2
8	volatiles → gas	3.92E-04	0.8
9	volatiles → aqueous	3.03E+00	1.4
10	volatiles → hydrocarbons	7.85E-04	0.8
11	volatiles → organic	8.51E-04	0.9
12	catalyst deactivation	1.00E-04	1

We would like to point out that the kinetic scheme as implemented does not conserve mass. We discovered this when calculating the mass fractions in the gas stream leaving mixing junctions. We calculated the species mass fractions in the stream after the mixing point by calculating the species mass flow rates and dividing by the total mass flow rate through the system. In a steady state reactor with a properly defined kinetic scheme (and no nuclear reactions), the total mass flow through the system is constant. However, the total mass flow rate as calculated from the species mass flow rates does not equal the total mass flow rate through the system. In our calculation of the species mass fractions, we found that the N_2 mass fraction was decreasing when it should have been constant (as N_2 is an inert). We finally determined that the source of this error was the rate scheme (other, self-consistent kinetic schemes do not result in a loss of inert). Possible sources of error include extrapolation of the kinetic parameters to a higher temperature, poorly characterized species (including incorrect overall molecular weights for each species lump), and the overall poor quality of the underlying kinetic data and kinetic parameters.

3.3 PREDICTION OF OPTIMAL OPERATING CONDITIONS

3.3.1 ANALYSIS OF THE BASELINE MODEL

In order to identify the key operating conditions impacting the yield of hydrocarbons, we drew 4096 sample points from the Sobol' low discrepancy sequence in four operating space dimensions: gas flow rate, gas composition, fresh catalyst flow rate, and fresh catalyst mass fraction. The limits on the operating space are given in Table 16.

Table 16. Operating space limits for the operating space scan to identify the key operating conditions.

	Gas Flow (u/umf)	Oil Mass Fraction	Cat Flow Rate (g/s)	Fresh Cat Mass Fraction
Low	1.09	0.05	0.15	0.5
High	7.69	0.15	0.45	1.0

Although we scanned over both the mass fractions and the total flow rates for the gas and solids inlet streams, the fundamental underlying variable is the species flow rate (which is just the product of the species mass fraction and the total mass flow rate). We present an operating space map of the hydrocarbon yield as a function of the volatiles and fresh catalyst mass flow rates in Figure 9. Overall, the hydrocarbon yield depends most on the flow rate of the volatile species that are upgraded into desired products. Lower gas phase volatiles flow rates generally give higher yields because lower gas flow rates lead to longer gas residence times. Hydrocarbon yields also generally increase with increasing fresh catalyst flow rate, which leads to more fresh catalyst in the reactor available for converting bio-oil into products.

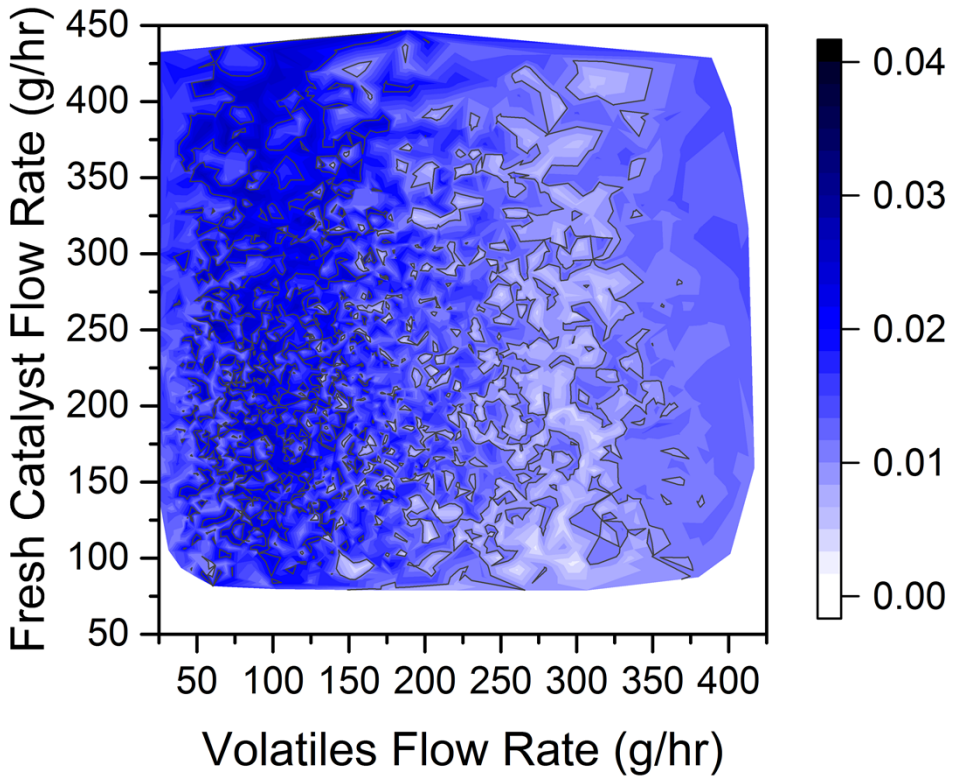

Figure 9. Yields of hydrocarbons (g/g oil fed) as a function of volatiles flow rate and fresh catalyst flow rate. The volatiles and fresh catalyst species flow rates are obtained by multiplying the total gas or solids mass flow rate and the mass fraction of the corresponding species. Deeper blue shades indicate higher yields.

3.3.2 UNCERTAINTY QUANTIFICATION OF THE YIELDS

We have also carried out uncertainty quantification calculations to investigate the impact of uncertainty in the kinetic parameters on the predicted product yields. For this aspect of the study, we chose the first eight Sobol' sequence points (due to the design of the Sobol' sequence, these are distributed approximately uniformly over the parameter space). Rate constants were sampled from a log-uniform distribution with upper and lower bounds that were ±1 order of magnitude from the baseline value. Reaction orders were sampled from a uniform distribution with upper and lower bounds that were ±0.25 from the baseline value. We did not change the reaction order of the catalyst deactivation reaction.

The results for the uncertainty quantification on the hydrocarbon yields are presented in Figure 10. The median predictions (open circles) fall within the range of the baseline predictions (filled squares). Uncertainties in the yields fall within about one order of magnitude from the median. Overall, the uncertainties in the kinetic parameters do not affect the qualitative conclusions of the baseline model.

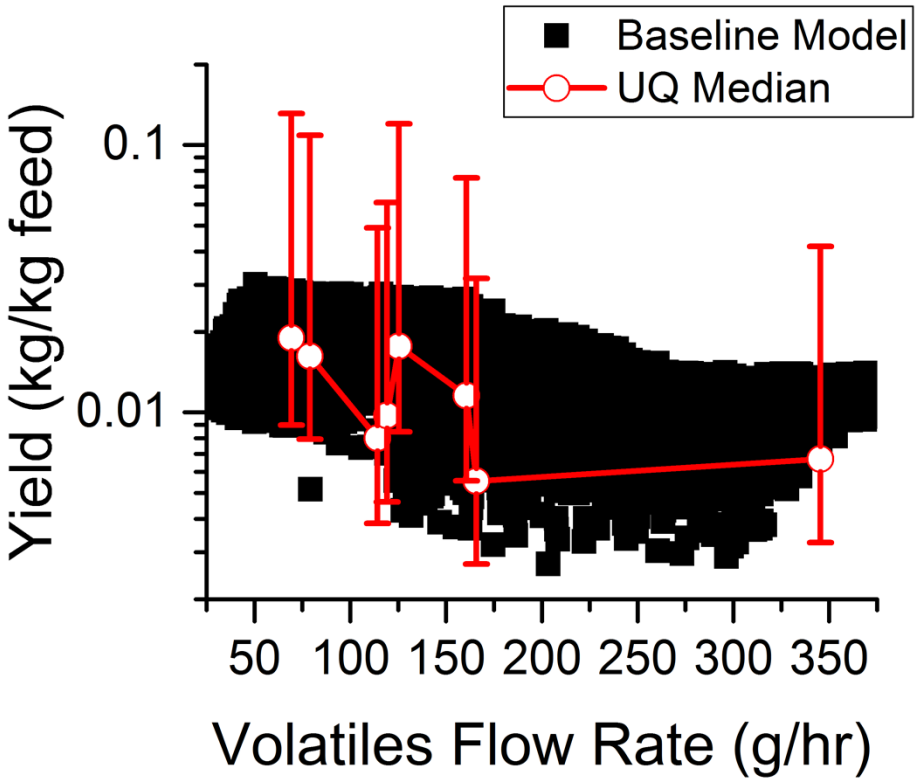

Figure 10. Uncertainty quantification of hydrocarbon yields. Solid squares are results for the baseline model, and open circles are the median values of the empirical yield distributions. Uncertainty envelopes (gray shaded regions) denote the 68% region around the median (i.e., 50% ± 34%). This uncertainty envelope corresponds to one standard deviation about the mean for a normal distribution.

3.3.3 GLOBAL SENSITIVITY ANALYSIS

We also carried out a global sensitivity analysis on the kinetic parameters at the center point of the operating space in order to better understand which parameters have the most impact on the yields of organic distillates and hydrocarbons. We note that, due to the relatively simple structure of the kinetic mechanism, the global sensitivity analysis is more a test of the method than a means of providing important insights. For the global sensitivity analysis, parameters were perturbed within the same range of values as outlined in Sec. 3.3.2. We employed the full global sensitivity analysis method (VGSA2) in order to calculate the second order sensitivity indices in addition to the first order and total effect sensitivity indices.

The results of the global sensitivity analysis are shown in Figure 11. Consistent with the kinetic mechanism, the parameters that have the largest impact on the production of organic distillates are the rate constant (K11) and reaction order (a11) for converting volatiles into organic distillates. Similarly, the yield of hydrocarbons is most impacted by the rate constant (K10) and reaction order (a10) for converting volatiles into hydrocarbons. The rate constant for catalyst deactivation (K12) is also important (we did not sample different reaction orders for the deactivation kinetics, so this parameter is identically zero). Comparing the first order and total effect indices in both cases suggests that there is a strong coupling between the rate constants and reaction orders. This is supported by the second order sensitivity indices measuring the coupling between the K10/a10 and K11/a11 pairs, as given in Table 17. In both cases, a significant fraction of the total variance in the product yields can be explained by interactions between the rate constants and the reaction orders.

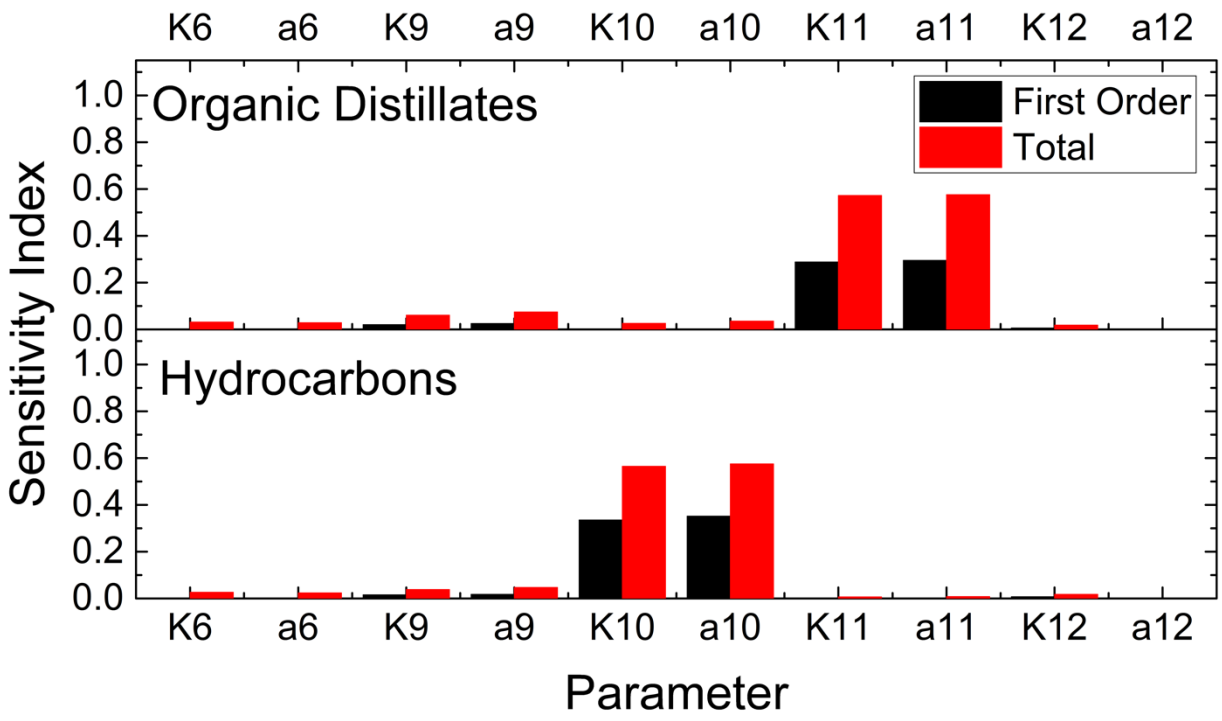

Figure 11. Global sensitivity analysis results for yields of organic distillates (top) and hydrocarbons (bottom) at the center point of the operating space for select kinetic parameters. Rate constants are labeled K_i, and reaction orders are labeled a_i. Other kinetic parameters had a negligible effect on the product yields of interest.

Table 17. Select second order sensitivity indices for the yield of hydrocarbons and organic distillates.

Yield of Product	Parameter Pair	Second Order SI
Hydrocarbons	K10/a10	0.19
Organic Distillates	K11/a11	0.23

3.4 SUMMARY OF BIO-OIL RESULTS

Our preliminary investigation of the upgrading of bio-oil over HZSM-5 in a bubbling bed reactor has shown that the most important operating parameters for maximizing product yield are the volatiles flow rate and the fresh catalyst flow rate. Low gas flow rates lead to longer gas residence times that promote conversion of the volatiles in the feed. Similarly, higher fresh catalyst flow rates lead to more mass of fresh catalyst in the reactor, higher reaction rates, and higher yields of hydrocarbons. Uncertainty quantification of product yields at select operating points demonstrate both that uncertainty in the kinetic parameters does not result in qualitative deviations from the baseline predictions and that the uncertainty in model predictions is approximately independent of the operating conditions. We also carried out a global sensitivity analysis calculation and found that the rate constants and reaction orders for the reactions leading to the desired products have the most influence on the yields of those products.

4. CONCLUSIONS AND FUTURE WORK

In this report, we have detailed the design and operation of the Chemics-Reactors reactor modeling program. It is a preliminary approach of handling multiphase flow (solids, liquids, and gases) under non-

isothermal conditions and can be applied to nearly any reactor. We have particularly demonstrated its use in modeling a bench-scale bubbling bed reactor at NREL employed in gathering experimental data related to upgrading bio-oil with HZSM-5. Our preliminary results showed that the gas flow rate through the NREL reactor was the single most important operating parameter. Specifically, low gas flow rates relative to minimum fluidization provide the best product yields.

Despite the great capabilities of the current version of Chemics-Reactors, we recommend some additions that would make it more powerful, flexible, and user friendly. If possible, a general-purpose algorithm for determining the properties and flow rates of streams should be implemented. Such an algorithm might take inspiration from process simulation packages. Additionally, the untested or experimental features (notably the non-isothermal energy balances and liquid phase capability) should be tested and revised as needed.

For the model of the vapor phase upgrading of bio-oil, we make some additional recommendations. An improved kinetic mechanism specific to pyrolysis bio-oil should be proposed and kinetic data collected to determine the associated activation energies, rate constants, and reaction orders. We also recommend designing a set of experiments to validate our model predictions.

5. REFERENCES

1. Saltelli, A., *Making best use of model evaluations to compute sensitivity indices.* Computer Physics Communications, 2002. **145**(2): p. 280-297.
2. Saltelli, A., et al., *Variance based sensitivity analysis of model output. Design and estimator for the total sensitivity index.* Computer Physics Communications, 2010. **181**(2): p. 259-270.
3. Campolongo, F., A. Saltelli, and J. Cariboni, *From screening to quantitative sensitivity analysis. A unified approach.* Computer Physics Communications, 2011. **182**(4): p. 978-988.
4. Iisa, K., et al., *In Situ and ex Situ Catalytic Pyrolysis of Pine in a Bench-Scale Fluidized Bed Reactor System.* Energy & Fuels, 2016. **30**(3): p. 2144-2157.
5. Yung, M.M., et al., *Multiscale Evaluation of Catalytic Upgrading of Biomass Pyrolysis Vapors on Ni- and Ga-Modified ZSM-5.* Energy & Fuels, 2016. **30**(11): p. 9471-9479.
6. Iisa, K., et al., *Production of low-oxygen bio-oil via ex situ catalytic fast pyrolysis and hydrotreating.* Fuel, 2017. **207**: p. 413-422.
7. Kunii, D. and O. Levenspiel, *Fluidization Engineering*. 2nd ed. 1991, Boston: Butterworth-Heinemann. xvii-xviii.
8. Adjaye, J.D. and N.N. Bakhshi, *CATALYTIC CONVERSION OF A BIOMASS-DERIVED OIL TO FUELS AND CHEMICALS .2. CHEMICAL-KINETICS, PARAMETER-ESTIMATION AND MODEL PREDICTIONS.* Biomass & Bioenergy, 1995. **8**(4): p. 265-277.