Creation of a Steam Generation System Digital Twin for Industrial Energy Optimization

A R&D report (dissertation) submitted in partial fulfillment

of the requirements for the degree

of

Bachelor of Engineering (Hons)

at

The University of Waikato

by

David Dickson

Supervised by:

James Carson & Benjamin Lincoln

THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

Abstract

Most industrial energy consumption is from process heating via fossil fuel combustion, making decarbonisation of industrial process heating systems critical for climate change initiatives. Digital modelling for these systems is essential for approaching the decarbonisation challenge for the analysis of optimisation and fuel switching opportunities of existing industrial sites. This project contributed by developing process model sub-components bespoke to the purpose of boiler steam system modelling. These components addressed current limitations of commercial process model, most importantly the lacking accessibility to modelling complex renewable materials, and a robust and flexible model solving framework for the future with Industry 4.0 that process engineering will be a part of. This was achieved by creating custom models within the IDAES-PSE framework. By leveraging the internal and overall code structure architecture of the framework, custom property packages, reaction packages, and unit operation models were developed that were dedicated to boiler steam system modelling. They were successfully implemented and used in the framework to produce boiler flowsheet models that could characterise real boiler metrics and behaviour.

Chapter 2 focused on developing property and reaction packages that were developed for the boiler-relevant combustion reactions of Methane, coal, wood biomass, and black liquor. Combustion stoichiometry for the impure fuels was estimated based on literature compositional and emissions data. The code was verified by a reaction process against the expected calculations in a spreadsheet model with less than 1% error. Shomate parameter equations for ideal gases were validated against literature measurements with predominantly under 0.01 [J/mol/K] error in heat capacity.

Chapter 3 focused on developing a custom combustion reactor unit operation model dedicated to boiler system modelling. The model was an adaptation of an existing IDAES unit model which added new variables and constraints that are better suited to specifying the conditions of a boiler system. The synergistic boiler modelling capabilities of the developed property package, reaction package, and custom unit model were demonstrated in a boiler flowsheet with flexible boiler conditions and expected efficiency behaviour.

Chapter 4 focused on taking the approach to boiler flowsheet modelling from Chapter 3 and applying it to a real case study. Measurement data from a pulp & paper mill black liquor recovery boiler was inputted into a flowsheet built from the components developed in this project. The case study flowsheet model solved the efficiency and could explore other conditions that the system may experience with varying resulting efficiencies. This paves the way not just for renewable boiler scenario modelling, but also for bi-directional digital twin applications on a digital twin platform.

Acknowledgements

Thank you to the University of Waikato for providing the opportunity of my Bachelors Degree and honours research and development project. Thank you, James Carson, for being my primary supervisor and seeking me out to undertake this project. Thank you, Ben Lincoln, being ever so supportive throughout the duration of this project. I appreciated the guidance, wisdom, and encouragement you gave, and of course the time you sacrificed in your busy schedule. Thank you, Bert Downs, for supporting me in my software and platform integration ventures. Thank you, Keegan Hall, for your support, particularly the data and information you provided for the Chapter 4 case study. Thank you for the whole team at Ahuora who I worked alongside during my project. Thank you, Amir Askarov, for doing running your CFD simulations beside me and reminding I could be farther behind than I was currently. Thank you, Bjorn Enriquez, for being a close friend and chemical engineering peer throughout. Thank you to the rest of my chemical engineering peers for being kind and for sharing this final year journey with me. Thank you, Aiden Lewis, for going to the gym with me and always giving me a good laugh. Thank you to my parents for your continual support. Thank you to my miscellaneous close friends for keeping it real.

Declaration of Authorship

I, David Dickson, declare that this dissertation entitled "Creation of a Steam Generation System Digital Twin for Industrial Energy Optimization" is the result of my own work, thoughts, and ideas, and includes nothing that is the outcome of work done in collaboration, except as specified in the text and acknowledgements.

I confirm that this work has not been submitted in whole or in part for any other degree or qualification at this or any other institution. All sources of information have been acknowledged and referenced appropriately. I have read and understood the University of Waikato's regulations on academic integrity and plagiarism.

Signed: David Dickson

Date: 9th October

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ADTP	Ahuora Digital Twin Platform
AML	Algebraic Modelling Language
EECA	Energy Efficiency & Conservation Authority
EOS	Equation of State
FTPx	Total Flow, Temperature, Pressure, and Mole Fraction Composition
GCV	Gross Calorific Value
GHG	Greenhouse Gas
IDAES	Institute for the Design of Advanced Energy Systems
IoT	Internet of Things
Ipopt	Interior Point Optimizer
MBIE	Ministry of Business, Innovation & Employment
NCV	Net Calorific Value
NIST	National Institute of Standards and Technology
PSE	Process Systems Engineering
UNSDG	United Nations Sustainable Development Goals

1 Introduction and Summary of Literature Review

1.1 Background

1.1.1 Anthropogenic Climate Change

The industrial revolution and consequent global anthropogenic climate change primarily by greenhouse gas emissions, have been so significant it warrants recognising a new geologic epoch, the Anthropocene (Steffen et al., 2011). The far-reaching harmful consequences have motivated international efforts to reverse climate change, most significant is The Paris Agreement (*The Paris Agreement | UNFCCC*, n.d.). The situation becomes increasingly urgent because of compounding and irreversible changes to Earth's climate systems that may ensue beyond 2°C warming above pre-industrial levels (Park et al., 2023). To solve this issue, the most significant industrial activities that contribute to climate change must evolve for planetary sustainability.

1.1.2 New Zealand Decarbonisation

New Zealand has its own national decarbonisation goals, namely New Zealand's Emissions Reduction Plan which aims to reach net zero carbon emissions by 2050 (Ministry of Business & Employment, 2022). The largest stationary energy end use category in New Zealand is industrial process heating. Steam boiler systems are the largest sole technology for stationary energy consumption, consuming 26% of total stationary energy usage (EECA, 2023).

New Zealand has a variety of economically important industries that rely on industrial process heating. The largest of which are meat, dairy, and pulp & paper processing. Process heat decarbonisation is a key but challenging step for planetary sustainability in New Zealand.

1.1.3 Decarbonisation for Industrial Steam Boilers

Steam boilers are the core technology which must evolve to decarbonise industrial process heating. There are two distinct approaches to achieving this, one is fuel switching for boilers in existing industrial steam systems. This is a viable approach that has been successfully executed in New Zealand such as for the dairy industry (EECA, 2020). Renewable fuel switching options include biomass, biogas, and renewable electricity. New Zealand is unique in its geographical contexts in that it has an abundant forestry industry and a high electricity grid composition of renewables. This context lends itself well to fuel switching pathways involving electrification or biomass fuel.

Another more established approach that can contribute to decarbonisation is improving boiler efficiency to reduce the overall fossil fuel demand. This approach does not match the potential emission reduction of fuel switching. However, it will have a higher economic benefit from lowering

operational costs and avoiding the higher capital and operation costs of fuel switching (Han et al., 2017).

Decarbonising industrial boiler systems for New Zealand is a major and complex engineering challenge because of economic dependence on fossil fuels, the variety of scale in industrial sites, and retrofitting for existing sites. Tools and solutions must be used and developed to address these aspects of the challenge. The goal is an accelerated and effective uptake of renewable industrial energy solutions.

1.2 Summary of Literature & Technology Review

1.2.1 Digitalization for Process Engineering

In context of the boiler decarbonisation challenge, modelling the system is an essential first step for all decarbonisation projects to ensure technological feasibility, economic viability, and environmental sustainability.

In process systems engineering, models have always been used for optimising and designing new processes. Digitalization is the development and use of digital tools to create value for existing systems; such is the case for digital process modelling solutions. Digital process modelling tools allow chemical process systems engineers to productively design, simulate, optimise, and analyse complex systems with reasonable accuracy.

1.2.2 Review of Process Simulation Tools

Current commercial chemical process simulation tools that have the technical capacity to be effectively used for steam boiler system decarbonisation. Notable and well used examples include Aspen Hysys and DWSIM. Aspen Hysys is highly popular, especially for use in industry. DWSIM is more unique as it is an open-source software. Both share the typical suite of chemical process simulation capabilities such as property packages, unit operations, flow sheeting, and optimisation. Both are CAPE-OPEN compliant, a common standard for commercial process simulators for interoperability with third-party modelling components (*About CAPE-OPEN | the CAPE-OPEN Laboratories Network*, n.d.). This means that both have the capacity for third-party software development to extend their base capabilities.

Being commercial software products, they typically lend themselves to more mainstream and historically relevant applications. For example, Aspen Hysys is targeted towards modelling for the petrochemical industry. Both have a generally wide range of potential applications but are limited when it comes to new and emerging applications such as renewable processes for decarbonisation.

For a process simulator to accessibly facilitate boiler steam system decarbonisation, it should be readily equipped with the appropriate compound properties and unit operations such as biomass. Hysys and DWSIM do not include biomass in their default sets of compounds. However, both allow custom specified compounds using built-in technical features, which can then be used for studies on renewable processes (Brinkmann & Seyfang, 2024; Kartal & Özveren, 2021). The issue is that technical features assume the user is technically proficient at understanding the thermochemistry and the software features to accurately specify a custom compound. This barrier to entry slows the uptake of decarbonisation projects such as for steam boilers. On the other end of the spectrum are spreadsheeting tools like Microsoft Excel that are highly generalisable and can be adapted to many applications, including boiler system decarbonisation modelling. But making such a comprehensive tool becomes elaborate, awkward, and slow to use in a spreadsheet format.

As new ideas and paradigms of digitalization emerge, the limitations of older process simulators like Hysys, DWSIM, or Excel can start to be identified and addressed. Such an emerging paradigm is 'Industry 4.0', characterised by greater connectivity across both digital and physical systems to enhance design and operability of industrial processes. From the themes of Industry 4.0 emerge the concept of 'Digital Twins'.

A digital twin is a digital model that is synchronous with the physical system it is simulating (Semeraro et al., 2021). It uses feeds of data from the physical system to simulate and make decision scenarios. A decision can be used to return control data back to the physical system. Various types of digital twins can be categorised as design, manufacturing, or service digital twins. A design digital twin would be a necessary type of digital twin for steam boiler system decarbonisation.

As described by de Beer & Depew (2021), "Process simulators are typically poorly integrated into engineering workflows beyond the process world, and if so, with a single directional information flow." This highlights the insufficient connectivity to high level workflows, interconnected systems, and adaptability to flow of information as the presiding limitations that digital twins can address.

A unique candidate is the IDAES-PSE framework which is an open-source framework and library, coded in python, and purpose-built for process modelling (Lee et al., 2021). It has the potential to enable digital twin tools to address these limitations. IDAES-PSE offers the essential components of a process simulator. These include the flowsheet, unit operations, property & reaction packages, and a tool for solving the flowsheet model. All of these are built upon the Pyomo Algebraic Modelling Language (AML) The open-source nature of IDAES means that the code and its components can be easily customized or adapted. For example, the IDAES generic property package framework can be used to create otherwise inaccessible custom property packages for renewable compounds. IDAES-PSE framework can also address the tediousness that comes with complex spreadsheet models, by

using pre-developed and packaged unit operations that must otherwise be developed in a bulky spreadsheet.

Older process simulators were developed without considerations for leveraging the unique characteristics of Industry 4.0. The best way to take advantage of concepts from Industry 4.0 in a way that contributes to decarbonisation is to build a new digital twin tool that is readily equipped with digital twin and renewable process modelling capabilities. The IDAES framework offers a solid foundation for such a digital twin tool. The independent technical features of Digital Twins are not new as they derive from past phases of industrial paradigms. However, a tool which comprehensively and cohesively embodies Digital Twin characteristics, especially one geared towards specific applications, is novel.

Table 1.1. Comparison of process simulation software solutions.

Process Simulator:	Aspen Hysys	DSWIM	Excel Spreadsheets	Proposed IDAES-Based Platform
CAPE-OPEN	Yes	Yes	No	n/a
Open-Source	No	Yes	n/a (low level)	Yes
Readily equipped for renewables processing	No	No	No	Yes
Readily equipped for Digital Twin applications	No	No	No	Yes
Solver method	SM	SM	I	EO

Note.

Solver methods are Sequential Modular (SM), Equation Orientated (EO), and Iterative (I).

1.3 Aim

The Ahuora Digital Twin Platform (ADTP) is a project working on leveraging the benefits of the IDAES-PSE framework to develop a new digital twin process simulator tool (Ahuora, 2025). The aim for this project is to develop and synthesize together the necessary components for a digital renewable boiler modelling tool that is accurate, comprehensive, and digitally adaptable. These components include compound property packages, reaction packages, unit operation models, flowsheet models, and extended modules for ADTP integration. The boiler model is to be validated and applied to a case study to demonstrate validity and value. The target applications are fuel switching scenario analysis and general boiler system optimisation.

2 Creation of Property and Reaction Packages

2.1 Introduction

Process modelling and simulation relies on accurate parameters that fix the mathematical equations used for characterising chemical properties and reactions. Accurately defined parameters provide the sound basis for the model's wider system of equations representing various processes and physical phenomena.

A property or reaction package is the data structure containing the set of parameters for code to use in a model simulation. In this study the IDAES framework facilitated the creation and implementation of property and reaction packages in a flowsheet model simulation.

Each property package must contain sufficient parameters for all the compounds desired to be modelled throughout an unbroken process stream. For example, a property package for a natural gas combustion reaction must at least contain the compounds CH_4, O_2, H_2O, and CO_2, for both the combustion feed and products. The reaction package similarly must contain all the desired reactions to be simulated.

For boiler decarbonisation modelling, both non-renewable and renewable types of boilers should be modelled, of which four have been identified. Natural gas and coal boilers are common types for non-renewable boilers. Electric and biomass boilers are common types for renewable boilers. The energetics of electric boilers can be sufficiently characterised simply by a direct energy stream to steam generation. The other boiler types require the parameters for their compounds and combustion reactions to be specified in property and reaction packages.

The required compounds needed to model the boiler combustion processes were identified and summarised in Table 2.1. Carbon Dioxide (CO_2) and Water (H_2O) are essential products of combustion reactions. Combustion feed air was assumed to only consist of Oxygen (O_2) and Nitrogen (N_2). Methane (CH_4), biomass, liquor, coal and black liquor were included as the essential fuel compounds. Incombustibles were included to represent ash products such as from biomass or coal. Steam was included as the generated product of a boiler. Four reaction packages were required for the combustion reactions of methane, biomass, coal, and black liquor.

Table 2.1. Summary of compounds that were modelled within property and reaction packages.

Compound	Phase	Purity	EOS
CH_4	Vapour	Pure	Ideal Gas
CO_2	Vapour	Pure	Ideal Gas

H_2O	Vapour	Pure	Ideal Gas
O_2	Vapour	Pure	Ideal Gas
N_2	Vapour	Pure	Ideal Gas
NO	Vapour	Pure	Ideal Gas
Biomass	Solid	Impure	Ideal Solid
Incombustibles	Solid	Impure	Ideal Solid
Coal	Solid	Impure	Ideal Solid
Black Liquor	Liquid	Impure	Ideal Liquid
Steam	Vapour/Liquid	Pure	IAPWS-95

2.2 Methods

2.2.1 Defining Physical Property Parameters

Three distinct categories of compounds were identified for combustion boilers, these were steam, ideal gas, and ideal solids. Steam used the property package known as IAPWS95 which is built into IDAES. The formulation used for IAPWS95 was the 2018 revised release on the IAPWS formulation 1995 for the thermodynamic properties of ordinary water substance for general and scientific use (International Association for the Properties of Water and Steam (IAPWS), 2018).

Both ideal gas and ideal solid compounds were specified in custom property packages and shared certain property parameters. These included the elemental composition, molecular weight, and the valid phase type.

The ideal gas assumption was used for all gaseous compounds. This was deemed valid as the boilers of interest would be relatively low pressure and high temperature. H_2O in Table 2.1 was listed as an Ideal Gas. This is not accurate for H_2O below 100 [°C] at atmospheric pressure, regarding both density and latent heat of vaporisation. H_2O or moisture in feed streams are expected to be below 100 [°C] but are assumed to be at negligibly low compositions. Combustion temperatures are expected to greatly exceed 100 [°C] so the ideal gas assumption for H_2O is reasonable here. The NCV specified for the combustion reactions accounted for the latent heat of vaporisation from fuel moisture, but latent heat of condensing would need to be accounted for by the property package.

Shomate parameters, A, B, C, D, E, F, G and H, are used for calculating heat capacity, enthalpy, and entropy as a function of absolute temperature. Shomate parameters were retrieved from the National Institute of Standards and Technology (NIST), and were based on measured data from Chase, (1998). A given gas compound may have had multiple sets of Shomate parameters suitable at different temperature ranges. The flame temperature could be expected to reach up to 2,000°C. The Shomate parameters which captured the expected temperature range for combustion were selected for. Some Shomate parameters will yield negative heat capacities for temperatures outside the accurate range.

This would cause significantly inaccurate simulation results, so these parameters were selected against in favour of parameters with higher overall temperature ranges.

The compounds, biomass, coal, black liquor, and incombustibles, are impure materials with many different constituents. The molecular weight specified for these compounds could be arbitrary as it was essentially only used as a conversion factor, for example between mass-based calorific values from literature and mole-based stoichiometry. For consistency, the molecular weight of these compounds was derived a molecular weight-weighted average of their estimated elemental or molecular compositions. Heat capacities were estimated based on literature values.

The biomass compound was based on wood biomass such as wood pellet fuel that is used in industry. Wood is a complex composite material making it difficult to comprehensively characterise its thermochemical properties. Instead, biomass was assumed to be composed of pure cellulose. The main molecular components of wood are cellulose, hemicellulose, and lignin. Cellulose is the majority component making up to 50% on a dry basis (Dongre et al., 2024). Also, it can be calculated that the relative elemental compositions of Carbon, Hydrogen, and Oxygen across these three molecular components have a standard deviation of 0.056. It is reasonable to simplify the total composition of biomass as cellulose only. This is especially so for combustion modelling purposes where the biomass NCV was not calculated from first principles (see Section 2.2.2). The arbitrary molecular weight of biomass was 162.1394 [g/mol], based on cellulose. A nominal heat capacity of 1500 [J/kg/°C] was based on wood and cellulose (Green & Perry, 2007).

Subbituminous coal specific heat capacity was based on literature data and specified to 1300 [J/kg/°C] at 300 [K] (Leśniak et al., 2013). The arbitrary molecular weight of subbituminous coal was 11.86 [g/mol], based on the weighted average of a Carbon, Hydrogen, Oxygen, Nitrogen, and ash from an ultimate analysis of coal. The coal ultimate analysis was taken from an investigation of the Huntly New Zealand subbituminous coal field (Mares, 2009). The density of coal was assumed from a literature value of 1300 [kg/m^3] (Jing et al., 2022).

Black liquor heat capacity was calculated by a bespoke correlation using temperature and solids percentage (Fardim & Tikka, 2011). It was assumed to be constant as calculated at a nominal temperature and solids percentage. The density of black liquor also used a correlation at a nominal temperature and solids percentage. The arbitrary molecular weight of black liquor was 18.02 [J/mol], the same as water.

Incombustible solid properties represented combustion ash products. The significant components of incombustible wood ash were identified to be SiO_2, Al_2O_3, CaO, Fe_2O_3, and MgO. A range of wood ash compositions from literature were used to calculate the weighted heat capacity and molecular

weight based on these major components (see Appendix A1)(AL-Kharabsheh et al., 2022a). Based on this analysis the heat capacity was 68.27 [J/mol/K] and the arbitrary molecular weight was 66.37 [g/mol].

2.2.2 Defining Reaction Parameters

The extent of detail specified for combustion reactions included the stoichiometry, heat of reaction, and the base component. Reaction kinetic parameters to calculate the rate of reaction were not needed as the combustion reaction was assumed to react instantly and completely and could be accurately modelled in a conversion reactor unit operation. For all fuels, the fuel itself was specified as the base component which was used for conversion equations.

The combustion stoichiometry for impure fuels like coal was estimated based on their assumed compositions, and in the case of coal and black liquor, their estimated emissions factor. These estimations and simplifications were reasonable due to the focus of this study being the heat energy content of fuels and not their ultimate combustion products. The comprehensive approach, assumptions, and calculations used were detailed by Dickson et al. (2025)(see Appendix C2).

The heat of reaction was defined either by first principles using enthalpy of formation parameters from the property package, or it was externally defined in the reaction package. The enthalpy of formation was specified in property packages by the Shomate 'H' Parameter. The first principles heat of reaction was calculated for reactions with only ideal gases, for example methane combustion. The heats of reaction for biomass, coal and black liquor used externally defined values.

$$dh_{reaction} = \sum h_{form,products} - \sum h_{form,reactants} \qquad (2.1)$$

For wood biomass, a correlation for NCV was provided in *Industrial Bioenergy Use – Updated methodology to estimate demand* (MBIE, 2016). This equation was based on radiata pine wood and accounts for water and Hydrogen content in wood combustion (see Eq. 2.2). The stoichiometry of biomass combustion was based on cellulose combustion.

$$NCV = GCV\left(1 - \frac{w}{100}\right) - 2.447\frac{w}{100} - (2.447 \times 9.01)\frac{h}{100}\left(1 - \frac{w}{100}\right) \qquad (2.2)$$

Where:

NCV = net calorific value or the wet basis calorific value [MJ/kg]

GCV = gross calorific value or the dry basis calorific value [MJ/kg]

w = water content [wt%]

h = Hydrogen content [wt%]

The NCV for subbituminous coal used was 24 [MJ/kg] based on the proximate analysis of coal from Huntly New Zealand subbituminous coal field (Mares, 2009). 24 [MJ/kg] is also the lower limit for bituminous coal NCV as per the *IPCC Guidelines for National Greenhouse Gas Inventories* (IPCC, 2006). This makes sense as subbituminous coal is less pure in combustible material compared to bituminous coal.

The reaction stoichiometry of coal combustion was estimated using a subbituminous emissions factor of 26.2 [tC/TJ] from IPCC (2006), and the ultimate composition analysis from Mares (2009). It was assumed during combustion that all Nitrogen (N) formed into Nitric Oxide (NO), all Hydrogen (H) formed into water H_2O, and total CO_2 production was exactly equivalent to the emissions factor used.

Black liquor used an externally calculated NCV which accounted for its solids content (see Eq. 2.3).

$$NCV_{Black\ Liquor} = 15x - 3\ [MJ/kg] \qquad (2.3)$$

Where x is the [wt%] solids content of the black liquor.

The stoichiometry for black liquor combustion was estimated using an emissions factor and the CO_2 and H_2O balance within literature black liquor gasification reaction stoichiometry (Helbig & Mawdsley, 2021; Naqvi et al., 2010).

2.2.3 Creation of IDAES Property Packages

Property packages were made using the IDAES Modular Property Package Framework. More information can be found from *Modular Property Package Framework* from idaes-pse.readthedocs.io. The property package for all properties used in this study can be found in Appendix B1. In a modular property package, parameter data is stored in a 'dictionary of dictionaries' data structure called the configuration. Each compound is specified in its own dictionary within the configuration. An example modular property package configuration dictionary is shown in Figure 2.1. When only certain compounds were required in a property package in this study, the package in Appendix B1 was copied with unnecessary compound dictionaries removed.

The elemental composition was specified. For impure compounds, not composed of a pure molecule, the name of the compound was arbitrarily parsed and given an elemental composition of 1. Methods for molar enthalpy and molar heat capacity were specified for all compounds. For solid and liquid compounds an additional density or specific volume method was specified. For ideal gas compounds, 'NIST' methods provided by IDAES was used. For impure compounds that required simplifying estimations, the 'Constant' method was used, also provided by IDAES. The valid phase type was specified to either liquid, vapour, or solid phases. Parameters were specified in the 'parameter_data'

dictionary. The parameters that were included depended on the compound and the methods selected. The determination of these parameters was discussed in Section 2.2.1.

```
"CH4": {
        "type": Component,
        "elemental_composition": {"C": 1, "H": 4},
        "enth_mol_ig_comp": NIST,
        "cp_mol_ig_comp": NIST,
        'valid_phase_types': PT.vaporPhase,
        "parameter_data": {
                "mw": (16.0425, pyunits.g / pyunits.mol),
                "cp_mol_ig_comp_coeff": {
                "A": (-0.703029 , pyunits.J / pyunits.mol / pyunits.K),
                "B": (108.4773, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-1),
                "C": (-42.52157, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-2),
                "D": (5.862788, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-3),
                "E": (0.678565, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**2),
                "F": (-76.84376, pyunits.kJ / pyunits.mol),
                "G": (158.7163, pyunits.J / pyunits.mol /pyunits.K),
                "H": (-74.87310, pyunits.kJ / pyunits.mol)}}}
```

Figure 2.1. Example Python code snippet of a modular property package configuration dictionary for the compound Methane. *Note:* Metadata dictionaries are not shown here.

Besides specifications for each compound, property packages also contain additional meta-data which apply globally within that property package. Each is also stored in dictionaries within the configuration dictionary. The valid phase list contains the phases that the property package supports, each with a corresponding equation of state. The state definition is set from a list of valid options supported by IDAES-PSE. The total flow, temperature, pressure, and mole fraction (FTPx) state definition was used in this study. The state bounds define the lower limit, upper limit, and nominal values for the state variables, total molar flow, temperature, and pressure. These are for used by the solver for optimisation and initialisation purposes. The base units are Pyomo units specifying time, length, mass, amount, and temperature so that other values generated by the framework share a common unit basis to derive from. These are specified to the SI-units seconds, metres, kilograms, moles, and Kelvin respectively. Reference pressure and reference temperature values are also defined for relative thermophysical properties such as entropy. For IDAES to build and use the modular property package, the GenericParameterBlock class was used with the modular property package's configuration dictionary as its argument.

'include_enthalpy_of_formation' is a Boolean configuration option. When set to 'True' the heat of reaction was calculated by the enthalpies of formation as part of the enthalpy balance of chemical reactions. It was turned off when the heat of reaction was specified externally, independent of

formation enthalpies. Otherwise, the heat of reaction from both formation enthalpies and external specification would be unintentionally combined to create an excessive overestimation of heat.

2.2.4 Creation of IDAES Reaction Packages

Reaction packages were made using both the ReactionParameterBlock class and the GenericReactionParameterBlock class with modular reaction property framework. Both classes have arguments for a property package which must contain all compounds present in the specified reaction stoichiometries.

In the GenericReactionParameterBlock class the build method is fixed, and it receives parameters from a configuration dictionary, just like for modular property packages. An example modular reaction configuration dictionary is shown in Figure 2.2. In the modular reaction package configuration dictionary base units were specified the same as they were in the property package. Reactions are organised into either 'rate_reaction' or 'equilibrium_reaction' dictionaries, both have a fixed config structure for their required parameters. All reactions in this study were specified using the 'rate_reaction' config structure. The stoichiometry dictionary specifies the reaction stoichiometry. Methods for heat of reaction, rate constant, rate form, and concentration form are specified and provided by IDAES. The methods used were constant heat of reaction, Arrhenius rate constant, power law rate, and mole fraction concentration. The parameter data dictionary specifies, according to the methods selected above, the heat of reaction, Arrhenius constant, and activation energy, all with Pyomo units. The Arrhenius constant and activation energy were specified arbitrarily as these values would be ignored in the custom combustion reactor unit model used in this study (see Section 3.2.2). The heat of reaction was set to 0 [J/mol] when it was intended to be derived from heat of formation enthalpies.

```
reaction_configuration = {
  "base_units": {"time": pyunits.s,
                 "length": pyunits.m,
                 "mass": pyunits.kg,
                 "amount": pyunits.mol,
                 "temperature": pyunits.K},
  "rate_reactions":{
    "RCH4": {
      “stoichiometry”: {
                       (“Vap”, “CH4”): -1,
                       (“Vap”, “O2”): -2,
                       (“Vap”, “CO2”): 1,
                       (“Vap”, “H2O”): 2,
                       },
      “heat_of_reaction”: constant_dh_rxn,
        “rate_constant”: arrhenius,
```

```
        "rate_form": power_law_rate,
        "concentration_form": ConcentrationForm.moleFraction,
        "parameter_data": {
          "dh_rxn_ref": (0, pyunits.J/pyunits.mol),
          "arrhenius_const": (0, pyunits.mol/pyunits.m**3/pyunits.s),
          "energy_activation": (0, pyunits.J/pyunits.mol)},
    }}}
```

Figure 2.2. Example Python code snippet of a modular reaction package configuration dictionary for Methane combustion.

In the ReactionParameterBlock class, reaction parameters are instantiated at a lower level directly within its build method. Only stoichiometry and heat of reaction parameters were specified, omitting the Arrhenius constant or activation energy.

Modular reaction packages are simple to develop and specify when interacting with just the configuration dictionary. But to add new custom variables or constraints, the build method by IDAES must be edited. This is difficult as the GenericReactionParameterBlock build method by IDAES is robust within its scope and does not facilitate appending new variables or constraints. The build method in the standard ReactionParameterBlock class is written at a lower level so is more suitable for appending new variables and constraints. This class was therefore predominantly used in this study as it was important to create bespoke custom variables and parameters (see Section 3.2.2). An example when the lower-level build method was leveraged was for indicating specific types of reactions within Pyomo set lists. This could not otherwise be done within the modular reaction package.

2.3 Physical Property and Reaction Parameter Results

2.3.1 Summarised Physical Property Parameters

The physical properties derived from the methods described in Section 2.2.1 are summarised into Table 2.2 and Table 2.3.

Table 2.2. Summary of physical property parameters used for ideal gases in this study.

Compound		O2	N2	CO2	H2O	CH4
Molecular Weight	[g/mol]	31.9988	28.0134	44.0095	18.0153	16.0425
Valid Temp. Range	[K]	700 to 2000	500 to 2000	298 to 1200	500 to 1700	298 to 1300
Shomate Parameters	A	30.032	19.506	24.997	30.092	-0.703
	B	8.773	19.887	55.187	6.833	108.477
	C	-3.988	-8.599	-33.691	6.793	-42.522
	D	0.788	1.370	7.948	-2.534	5.863
	E	-0.742	0.528	-0.137	0.082	0.679
	F	-11.325	-4.935	-403.608	-250.881	-76.844
	G	236.166	212.390	228.243	223.397	158.716

	H [kJ/mol]	0.000	0.000	-393.522	-241.826	-74.873
Cp at 1000 K	[J/mol/K]	34.86	32.69	54.3	41.27	71.79

Table 2.3. Summary of physical property parameters used for ideal solids and liquids in this study.

Compound		Biomass	Coal	Black Liquor	Ash
Arbitrary Molecular Weight	[g/mol]	162.14	11.86	18.02	66.37
Cp Constant	[J/mol/K]	243.21	15.42	75.68	68.27
	[J/kg/K]	1300	1300	4200	1028.63
Constant Density	[mol/m3]	2960	109599	76471	2960
	[kg/m3]	480	1300	1378	480
Valid Phase		Solid	Solid	Liquid	Solid

2.3.2 Summarised Reaction Property Parameters

The reaction parameters were derived from the methods described in Chapter 2.2.2 and are summarised in Table 2.4. These summarised parameters were enough to sufficiently specify IDAES reaction packages.

Table 2.4. Summary of the combustion reactions formulated and used in this study.

Combustion Reaction		Methane (CH4)	Subbituminous Coal	Wood Biomass	Black Liquor
Stoichiometry [mol]	Fuel	-1	-1	-1	-1
	O_2	-2	-0.643	-6	-0.169
	CO_2	1	0.716	5.950	0.3
	H_2O	2	0.025	4.959	0.038
	N_2	0	0.005	0	0
	Ash	0	0.005	0.044	0.144
Heat of Combustion	[MJ/mol]	-0.80	-0.33	-2.75[A]	-0.14[B]
	[MJ/kg]	-50.00	-24.00	-16.96[A]	-7.50[B]
Molecular Weight of Fuel	[g/mol]	16.04	11.86	162.14	18.02

Note: [A]The heat of reaction for Wood Biomass at 9 [wt%] water and 6 [wt%] Hydrogen content as per Equation 2.2. [B]The heat of reaction for Black Liquor at 70 [wt%] solids content as per Equation 2.3.

2.3.3 Verification of Code Behaviour

It was important to verify that the code operating on the property and reaction packages would function in the intended manner. A stoichiometric reactor process flow model was devised to verify

that the thermochemical characteristics of the compounds and reactions would be modelled by the intended mathematical methods. The reactor process was modelled in both a Python IDAES framework flowsheet, and an Excel spreadsheet (see Appendices B4 and A2 respectively). A process flow diagram of the model used for code verification is displayed in Figure 2.3.

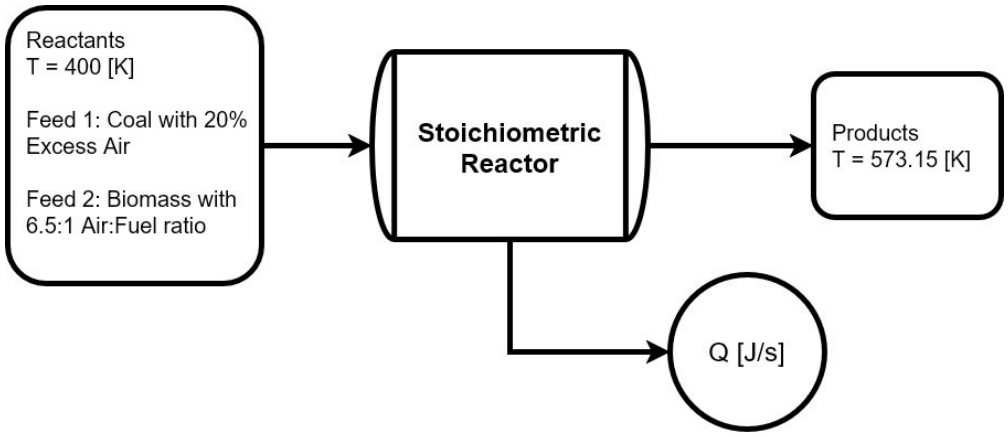

Figure 2.3. Process flow diagram for verifying Python code operation on property and reaction packages.

Both coal and biomass combustion were separately tested using the conditions specified in Figure 2.3. Both Excel and Python models solved for heat duty (Q) and were compared. Both coal and biomass combustion demonstrated <1% deviation between the Excel and Python models.

2.3.4 Validation of Shomate Parameters

The thermophysical parameters for pure compounds were validated against measured data from the literature. Many parameters were estimated directly from literature data, in which case validation methods were not directly used for these parameters.

Measured values for the specific heat capacity of pure compounds were taken from *NIST-JANAF Thermochemical Tables* (Chase, 1998). A control volume of the pure compound was modelled in an IDAES flowsheet to simulate the heat capacity at a given temperature and pressure (see Appendix B5). The absolute difference of heat capacity between the measured data and simulation is presented as heatmap in Figure 2.4.

	Absolute difference						
Temperature	O2	O2	N2	CO2	H2O(g)	NO	CH4
[K]	100 to 700 [K]	700 to 2000 [K]	500 to 2000 [K]	298 to 1200 [K]	500 to 1700 [K]	298 to 1200 [K]	298 to 1300 [K]
298.15	0.007	5.404	1.518	0.001	0	0.012	0.009
300	0.007	5.298	1.472	0.003	0.001	0.008	0.003
400	0.002	1.787	0.201	0.011	0.001	0.013	0.001
500	0.012	0.537	0.001	0.012	0.008	0.028	0.01
600	0.017	0.119	0.006	0.003	0.007	0.012	0
700	0.006	0.005	0.006	0.003	0.007	0.016	0.003
800	0.267	0.01	0.005	0.007	0.015	0.022	0.005
900	0.995	0.002	0.001	0.002	0.011	0.007	0.005
1000	2.43	0.006	0.005	0.003	0.002	0.014	0.001
1100	4.814	0.007	0.004	0.006	0.014	0.02	0.006
1200	8.387	0.003	0.001	0.004	0.017	0.015	0.008
1300	12.255	0.002	0.002	0.047	0.009	0.114	0.006
1400	7.28	0.006	0.004	0.163	0.006	0.297	0.048
1500	1.269	0.005	0.004	0.358	0.019	0.583	0.128
1600	5.901	0.003	0.002	0.664	0.016	0.991	0.253
1700	14.358	0.003	0.002	1.133	0.018	1.534	0.423
1800	0.37	0.006	0.005	1.786	0.103	2.227	0.637
1900	-	0.005	0.004	2.658	0.259	3.084	0.886
2000	-	0.006	0.005	3.809	0.505	4.117	1.162

Figure 2.4. Heatmap of absolute difference between measured Cp [J/mol/K] data of pure components from *NIST-JANAF Thermochemical Tables* (Chase et al., 1998) and the Cp simulated in IDAES flowsheet control volumes from 298.15 to 2000 [K]. The thick bordered boxes indicate the invalid temperature range listed for each set of Shomate parameters.

2.4 Discussion

2.4.1 Verification of Code Behaviour

A verification was done by modelling combustion reactions in both spreadsheet and flowsheet models using the developed property and reaction packages. It demonstrated that solving flowsheet models on the IDAES framework uses the same equations as the intended equations used in a spreadsheet model. Behind-the-scenes code in the IDAES framework may be operating on alternative equations or methods, so verifying ensured the equations being used for modelling were known.

2.4.2 Accuracy of Parameters

The correlation used for the specific heat capacity of ideal gases was the Shomate equation and parameters. Individual sets of parameters for a given ideal gas component had limited valid temperature ranges. The validation analysis showed that most ideal gas compounds and their parameter set aligned well with literature measurements across a suitably wide range from 298.15 to 2000 [K]. The compound that deviated the most from literature was Oxygen as seen in Figure 2.4. Two sets of parameters with valid temperature ranges, 100 to 700 [K] and 700 to 2000 [K], were considered for selection. If the 100 to 700 [K] Oxygen parameters were selected, low temperature instances of Oxygen would be thermodynamically accurate but any oxygen exiting the hot boiler

model could exceed an inaccuracy of 10 [J/mol/K]. This is convenient for combustion applications where most of the oxygen is expected to be consumed. If the 700 to 2000 [K] Oxygen parameters were selected, all higher temperature instances of Oxygen would be accurate. Lower instances of Oxygen could reach at most a difference of around 5 [J/mol/K]. Even though most of the Oxygen in a combustion reaction would be present in the feed at a lower temperature, the overall accuracy of the higher temperature parameters was more greatly appreciated. Therefore, 700 to 2000 [K] O_2 parameters were selected and used for the remainder of this study.

It was not useful to validate parameters estimated based on literature data because those parameters in isolation could only be validated by the same literature data. Where possible, multiple points of literature data were used to estimate these parameters more accurately. For example, the estimated reaction stoichiometry for the combustion of impure compounds referenced literature data for both emissions factors and compositions.

Constant parameters were estimated from the literature for the impure compounds, biomass, coal, black liquor, and ash. For density, heat capacity, and arbitrary molecular weight. It was important for specific heat capacity to be very accurate, but for the purposes of this study, the constant heat capacity parameters used were sufficiently accurate.

2.4.3 Future Work

Further work is to be done to fully integrate these custom property and reaction packages into the ADTP. Especially for user-inputted reactions, for which there is no current platform solution for yet. The user should be able to both specify their own reactions, and pre-formulated reaction packages such as those from this study.

Of the assumptions on physical property made, only the estimated constant heat capacity assumption could significantly limit the thermophysical accuracy of these compounds. This is a limitation of the property packages that could be addressed in future work that would formulate dedicated temperature dependent heat capacity correlations for novel compounds.

More accurate equations of state such as the Peng-Robinson equation of state could be implemented for more accurate property packages.

Since water in combustion flue was assumed to be an ideal gas, the latent heat qualities of water in air have room be more accurately characterised. This would be useful for increasing the accuracy of a condensing economizer process. This would involve implementing a humid air property package.

It would be important to support a wider range of renewable processes by developing the appropriate property and reaction packages. This would involve characterising the unique physical properties and

any relevant reactions. Many of the estimated parameters in this study were based on literature data. Research experiments to add more thermochemical data to the literature would be important, especially for novel and complex compounds and processes that are increasingly important to renewable processes.

2.5 Conclusion

This chapter set out to develop the physical property and reaction parameters that were relevant to the combustion process in the context of industrial boiler decarbonisation. Ideal gas components were simply characterised by the ideal gas law and Shomate parameters. The combustion fuels coal, wood biomass, and black liquor are complex impure materials so required more estimations to yield adequately accurate parameters. Stoichiometry parameters were estimated mostly based on emissions factors and literature composition analysis. Once formulated, the physical property and reaction parameters were encoded into IDAES property and reaction packages. These coded components were important as part of a larger process modelling and simulation framework. To verify that the property and reaction parameters were being used in the intended calculations by the code, a code verification test was done. A simple reaction process was modelled both in IDAES using the packages and in an Excel spreadsheet. The IDAES code model was accurate to the Excel to within a percentage error. The Shomate parameters used for ideal gases were validated against literature measured data of heat capacity. This showed accuracy of Shomate equation and helped select parameters that would be more accurate. The property and reaction packages developed in this chapter could be reliable used within IDAES flowsheet models in this study.

3 Creation of a Boiler Flowsheet Model

3.1 Introduction

Fuel switching and energy optimisation are the primary methods for industrial process heat decarbonisation in New Zealand. There is a significant barrier to this energy transition to renewable energy as existing industrial sites are complex. A fuel switching or optimisation solution for a given site will be nontrivial, costly, and specific to that site to some extent. Digital boiler modelling is critical as it grants the ability to easily and accurately navigate these obstacles to lower the both the technical and economic barriers of industrial decarbonisation. The flowsheet modelling architecture of the IDAES framework was leveraged in this chapter to develop a boiler model that could be used in industry for this purpose.

3.1.1 High Level Code structure of IDAES Process Flowsheet Modelling

This study required the creation of a process flowsheet model of a steam boiler system using the IDAES framework. Process flowsheet models are built from unit operation models interconnected by material process streams. Unit operations are assigned property packages and reaction packages that support the selected process compounds and reactions respectively. Figure 3.1 illustrates how these distinct components interrelate with each other in a hierarchical structure to form a complete process flowsheet model. An example of the code implementation for this hierarchy is shown in Figure 3.2. The methods in this section were to abide by this class structure and hierarchy.

3.1.2 Approach and Aim for Boiler Model Creation

This chapter sets out to create a specific boiler unit operation model designed to use the property and reaction packages created in Chapter 2. This model can then be used as part of a wider steam system flowsheet model. Existing unit operation models from the IDAES code repository are part of the IDAES framework. Some of these models were directly used for simple and general unit operations within flowsheets. Examples include heaters, coolers, heat exchangers, mixers, and ideal separators. The IDAES stoichiometric reactor unit model could be used to model the combustion reaction process. But it lacks details important to a boiler. To address this a custom combustion reactor unit model was developed by building upon the existing IDAES stoichiometric reactor model. This unit would simulate the fire-side combustion process of a boiler that generates hot flue gas that proceeds to transfer heat to water. A comprehensive boiler flowsheet model would then be created to demonstrate the use of the custom reactor model for boiler modelling. The developed custom unit model should be integrated into the ADTP, as well as the other unit models to form a flowsheet model, also in the ADTP.

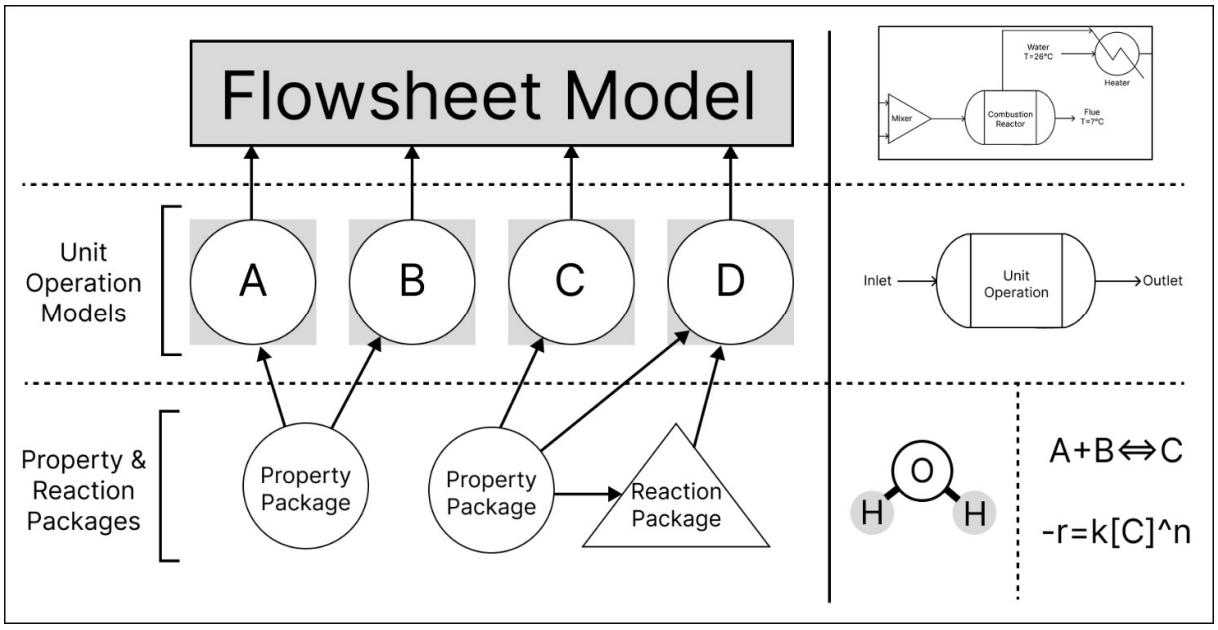

Figure 3.1. Illustrative hierarchy of property/reaction packages, unit operation models, and the culminating flowsheet model.

3.2 Methods

3.2.1 Code Structure of IDAES Unit Operation and Flowsheet Models

The code structure follows the hierarchy illustrated in Figure 3.1. Figure 3.2 shows an example code implementation of the hierarchy and is described as follows. The property package, m.fs.properties, was built with the IDAES GenericParameterBlock with the modular configuration dictionary as an argument. The reaction package, m.fs.reaction, was built with the BMCombRxnParameterBlock class with the property package as an argument. Note that the property package can also be built by a non-modular parameter block class, and the reaction package by a modular configuration dictionary. Both packages were entered as arguments for the StoichiometricReactor unit model. Multiple process unit operation models were connected via pyomo Arcs which represent interconnecting process streams. In this example an Arc form the process stream between a mixer outlet and the reactor inlet.

```
configuration = {
    "components": {
        "biomass": { #woody biomass
            "type": Component,
            "elemental_composition": {"C":6, "H": 10, "O": 5},
@declare_process_block_class('BMCombRxnParameterBlock')
class MultiCombReactionParameterData(ReactionParameterBlock):
    def build(self):
        super(MultiCombReactionParameterData, self).build()
m.fs.properties = GenericParameterBlock(**configuration)
m.fs.reaction = BMCombRxnParameterBlock(property_package=m.fs.properties)
m.fs.react = StoichiometricReactor(
    property_package = m.fs.properties,
    reaction_package = m.fs.reaction,
m.fs.stream1 = Arc(source=m.fs.mixer.outlet, destination=m.fs.react.inlet)
TransformationFactory("network.expand_arcs").apply_to(m)
```

Figure 3.2. The code implementation of the hierarchy of components as illustrated in Figure 3.1. (Pink) and (orange) boxes represent the property package and reaction package respectively. These are assigned to the unit operation model in the (green) box. Finally, unit operation models are connected by their inlet/outlet streams via Pyomo 'Arc()'components.

IDAES Unit Models are instantiated as a class that inherits from the 'UnitModelBlockData' base class. In this class, configuration options are declared to decide specific model behaviour and methods. Important configuration options were for specifying the property and reaction packages associated with a reactor unit model. After the configuration was the build method. The build method declared variables and constraints which define the model behaviour. This is where additional variables and constraints were declared to extend the base unit model. The IDAES stoichiometric reactor was adapted as such with additional variables and constraints to create a bespoke boiler combustion reactor.

IDAES flowsheets are Pyomo models instantiated with the IDAES 'FlowsheetBlock' class and were declared as 'm.fs' in this study. Unit operation blocks and arcs are attributed to the flowsheet by following 'm.fs.' with the name of the component as shown in Figure 3.2. Adding unit operations introduces various degrees of freedom. These are variables that must be directly fixed with numeric values, or indirectly fixed with equations, Pyomo constraints, and Pyomo arcs. When the degrees of freedom summed to zero the model could be initialised, then computationally solved. Pyomo expressions evaluate metrics of the model such as boiler efficiency.

3.2.2 Creation of a Custom Combustion Reactor – IDAES Unit Operation Model

Making a custom combustion reactor by adapting an existing unit model was necessary for adding details important to a bespoke combustion boiler model. These adaptations created additional variables that were not present in the original model and are as follows:

- conversion of fuel combustion,
- casing heat loss,
- excess stoichiometric air of combustion,
- air-fuel ratio,

and where applicable depending on the fuel:

- net calorific value correlation,
- ash mass content.

The following methods are for the custom combustion reactor unit model; its code is presented in Appendix B6. The methods emphasise the non-code aspects due to most of the code was borrowed from the existing IDAES stoichiometric reactor unit model that was adapted. Methods only include instances where code was added or significantly altered from the original.

Because of the bespoke nature of the custom unit model and the assumptions incurred, the reaction package was hardcoded to the control volume, and the reaction package configuration option was thusly removed. The main assumption was the inclusion of custom index sets that are relied upon when declaring variables and constraints. A user may want to specify an alternate reaction package that is incompatible by lack of these index sets. Hardcoding removes this possibility.

Index sets helped variables and constraints refer to common types of parameters across multiple reactions. Parameters in the reaction package were stored in dictionaries indexed by keys. The 'rate_reaction_idx' index set contained a key for each reaction in the reaction package. A variable could now refer to all reaction keys of a parameter by iterating through this index set. Segregating reactions for special variables was also done by creating a smaller index set. For example, the 'uncombs_set' index set only contained keys for combustion reactions that have ash content. Ash-specific variables and constraints could iterate through this set for ash-only reactions.

The associated constraint equations for the added variables are presented in Equations 3.1 to 3.4. These variables and constraints were inserted into the 'def build(self)' build method of the IDAES 'StoichiometricReactor' unit model.

$$X = \xi/(N_{total} * C_{fuel}) \qquad (3.1)$$

where:

X = conversion

ξ = extent of reaction [mol/s]

N_{total} = total molar flow [mol/s]

C_{fuel} = molar feed concentration of fuel

$$Q_{casing} = UA\left(T_{surface} - T_{outlet}\right) \quad (3.2)$$

where:

Q_{casing} = convection heat loss from boiler casing [J/s]

U = overall heat transfer coefficient of convection [W/m^2/K]

A = heat transfer area [m^2]

$T_{surface}$ = casing surface temperature [K]

T_{outlet} = boiler outlet temperature [K]

$$(1 + E) \times \frac{v_{O_2} \dot{N_{fuel}}}{0.21 \ _{fuel}} = \dot{N_{air}} \quad (3.3)$$

$$AFR = \dot{N_{air}} M_{air} / \dot{m_{fuel}} \quad (3.4)$$

where:

E = excess air beyond 100% stoichiometric air ratio

v_{O_2} = Oxygen stoichiometry

$\dot{N_{fuel}}$ = fuel molar flow [mol/s]

v_{fuel} = fuel stoichiometry

$\dot{N_{air}}$ = air molar flow [mol/s]

AFR = air fuel ratio of feed on a mass basis

M_{air} = molar weight of air [g/mol]

$\dot{m_{fuel}}$ = mass flow of fuel [g/s]

By default, IDAES reactions are specified by the absolute extent of reaction in [mol/s]. Instead, the percent conversion of the base reactant is more useful to specifying for relative reaction extents independent of the flow rate and was thusly added as a variable and constraint.

Further constraint equations were required to correctly link user-variables to base idaes-variables. For example, linking the air flow rate calculated by excess air or air-fuel ratio must be linked to the control volume inlet flow by Equation 3.5.

$$0.21 \sum \dot{N_{air}} = C_{O_2} \dot{N_{total}} \qquad (3.5)$$

A loop was used to declare separate sets of variables for each reaction. Figure 3.3 shows the loop iterating through the 'rate_reaction_idx' index set and assigning variables suffixes using 'setattr(self,f"variable_{r}")'. Constraints also iterate through index sets and refer to these variables by their suffix using 'getattr(self,f"variable_{r}")' as shown in Figure 3.4.

```
for r in self.reaction_package.rate_reaction_idx:
    p,l = self.reaction_package.limit_reactant_dict[r]
    setattr(self,f"conversion_{r}", Var(initialize=1,bounds=(0,1), units="dimensionless"))
    setattr(self,f"dh_rxn_{r}", Var(initialize=1000000, units=pyunits.J/pyunits.mol))
    setattr(self,f"excess_air_percent_{r}", Var(initialize = 15, units="dimensionless"))
    setattr(self,f"air_fuel_mass_ratio_{r}", Var(initialize=6, units="dimensionless"))
    setattr(self,f"mass_flow_kg_{l}", Var(initialize=1 ,doc="mass flow of fuel in kg/s"))
    setattr(self,f"mole_flow_{l}", Var(initialize=100, doc="fuel molar flow in mol/s"))
    setattr(self,f"mole_flow_air_{r}", Var(initialize=1000, doc="mole flow air in mol/s"))
```

Figure 3.3. For-loop for declaring variables on a per-reaction basis.

```
def build(self):

    @self.Constraint(self.flowsheet().time,self.reaction_package.rate_reaction_idx)
    def conversion_performance_eqn(b, t, r):
        p,l = self.reaction_package.limit_reactant_dict[r]
        return getattr(b, f"conversion_{r}") == (
        b.control_volume.rate_reaction_extent[t,r]
         /(b.control_volume.properties_in[t].mole_frac_comp[l]
         *b.control_volume.properties_in[t].flow_mol
        ))
```

Figure 3.4. Code implementation of the Pyomo constraint for reaction conversion within the unit model build method.

Developing constraints for fuel ash mass content was nontrivial. Consider the case where the user specifies an ash mass content that is different from the default as initially implied by the default reaction stoichiometry. A simple solution for this is to modify the stoichiometry of the ash products, but doing so disobeys the mass balance that the default stoichiometry assumed. A set of equations to address this issue must conserve both the mass balance of the reaction. It should also account for conserving NCV [J/mol] that used the default stoichiometry as a basis.

The constraint equations used that address this issue were formulated by (Dickson et al., 2025)(Appendix C2). Their approach involved adjusting both fuel and ash stoichiometries whilst accounting for molar weight ratios, initial stoichiometry, and initial NCV. The main assumption for

the equations was that the initial default stoichiometry obeyed the mass balance, which is reasonable as the default stoichiometry was specified as such in an immutable parameter.

The resulting constraint equations used are presented in Equations 3.6 and 3.7. Equation 3.6 fixes the new ash stoichiometry and Equation 3.7 fixes the new fuel stoichiometry. The calorific values of combustion were also adjusted by simply by taking the existing value and multiplying by the initial stoichiometry and dividing by the final stoichiometry.

$$v_{ash,n} = (\left(\left(x_{as} - v_{as,i} mw_{as}\right)\left(-v_{fuel,i} mw_{fuel}\right)\frac{1}{1-\left(x_{as} - v_{as,i} mw_{ash}\right)}\right) + v_{as,i} mw_{as})/mw_{as} \quad (3.6)$$

$$v_{fuel,n} = -(\left(\left(x_{ash} - v_{as,i} mw_{ash}\right)\left(-v_{fuel,i} mw_{fuel}\right)\frac{1}{1-\left(x_{as} - v_{as,i} mw_{ash}\right)}\right) + v_{fuel,i} mw_{\text{fuel}})/ mw_{\text{fuel}} \quad (3.7)$$

where:

$v_{as,n}$ = new stoichiometry for ash [mol]

$v_{fuel,n}$ = new stoichiometry for fuel [mol]

$v_{ash,i}$ = initial stoichiometry for ash [mol]

$v_{fuel,i}$ = initial stoichiometry for fuel [mol]

x_{as} = fuel ash content [mol%]

mw_{as} = molecular weight of ash [g/mol]

mw_{fuel} = molecular weight of fuel [g/mol]

3.2.3 Creation of a Boiler Flowsheet Model

The custom boiler combustion unit model was used as part of a hypothetical superheated steam boiler system flowsheet model as. The process flow diagram for this model is displayed in Figure 3.5. The unit operations in the flowsheet were as follows.

- Combustion reactor.
- Ash separator.
- Superheating heat exchanger.
- Main boiler heat exchanger.
- Blowdown water separator.

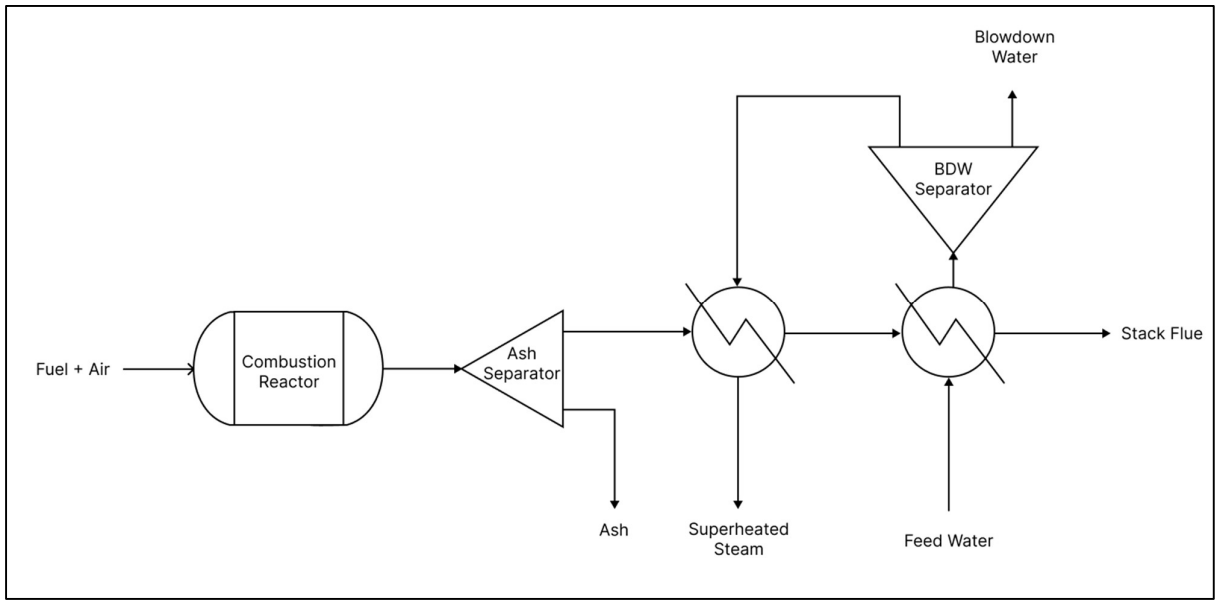

Figure 3.5. Process flow diagram for a superheated steam boiler.

The python file for this flowsheet can be seen in Appendix B7. It follows the code structure as described in Section 3.2.1. The combustion reactor used the custom combustion reactor model developed in section 3.2.2 All other unit models were directly used and remained unmodified from the IDEAS code repository. The ash separator used an ideal phase separator to remove any solids, mostly ash, from the products flue stream. 0-dimensional heat exchangers were used for the superheating and main boiler heat exchangers. The blowdown water separator used an ideal phase separator to separate liquids the main boiler outlet. The blowdown ratio was specified by fixing the quality of the main boiler steam-side outlet to 1 minus the blowdown ratio as a decimal. Then the exact amount of blowdown water would be in the liquid phase of the outlet and separated out, leaving saturated steam which proceeds to the superheater.

The model as presented in Appendix B7 shows its variables fixed to specify the steam demand; it then calculates the required total feed flow of fuel and air. A Pyomo 'Sequential Decomposition' process was used to determine the ideal order in which to initialise each unit operation (Bynum et al., 2021). Sequential decomposition identifies the order in which unit models are to be initialised and any 'tear streams' which require an initial guess. The sequential decomposition identified the flue-stream in between the heat exchangers as the 'tear stream'. Then the solver was called to solve the system. The solver package used in this study was the Interior Point Optimizer (Ipopt) (Wächter & Biegler, 2006).

An expression for boiler efficiency, calculated by Equation 3.8, was called after the solve. The efficiency was the ratio between heat transferred to water for steam generation and total reaction heat generated in the reactor.

$$\eta = \left(Q_{superhea} + Q_{main}\right)/\left(\sum(\xi_r * dh_{comb,r}\right) \qquad (3.8)$$

where:

η = boiler efficiency

Q = heat transferred to water [J/s]

ξ_r = extent of reaction 'r' [mol/s]

$dh_{comb,r}$ = specific heat of reaction 'r' [J/mol]

The solved flowsheet results were analysed by printing evaluated expressions, unit data, and stream data to the Python terminal. The '.report()' function is a built-in method that was used for printing summary tables of unit operations to the terminal.

To demonstrate the simulated flexibility of this flowsheet model, it was simulated across a range of part loads, flue stack temperatures, and fuel types to generate part-load boiler efficiency curves as shown in Figure 3.6. This was done by repeatedly solving the model with nested loops.

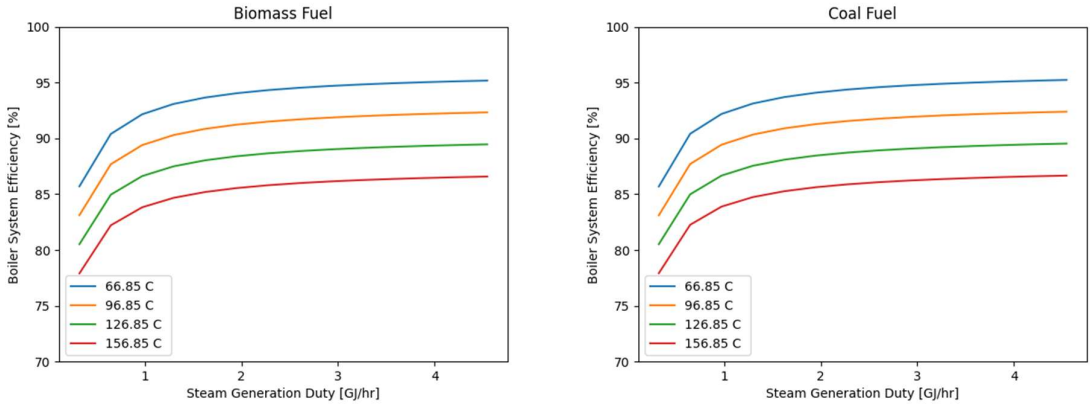

Figure 3.6. The flexibility and range of this model is demonstrated by boiler efficiency curve plots for different fuels across various stack flue temperature conditions.

To show that these sets of simulated boiler efficiency curves were characteristic of expected real boiler efficiency curves, a simple correlation by Shang (2000) was fitted to one of the simulated curves and is presented in Equation 3.9.

$$\eta_{Boiler} = \frac{1}{a+b\frac{m_{max}}{m_{steam}}} \qquad (3.9)$$

where:

η_{Boiler} = Boiler Efficiency

a,b = Fitting parameters

m_{max} = maximum boiler steam load [kg/s]

m_{steam} = actual boiler steam load [kg/s]

3.3 Discussion

3.3.1 Results

The custom unit model was seamlessly integrated as part of the IDAES flowsheet architecture. It was then used in a boiler system flowsheet model which successfully modelled the expected behaviour of a boiler system. A boiler efficiency curve correlation intended for real boiler systems was fitted to the simulation curve. Figure 3.7 shows the correlation fitted to one of the coal part-load efficiency curves using parameters a=1.14, b=0.011, and m_{max} assumed to be 0.5 [kg/s]. With those parameters the fit was evaluated to have a percent deviation of 0.14% and a root mean square deviation (RMSD) of 0.23. This showed that the simulated boiler behaviour was characteristic of a real boiler system.

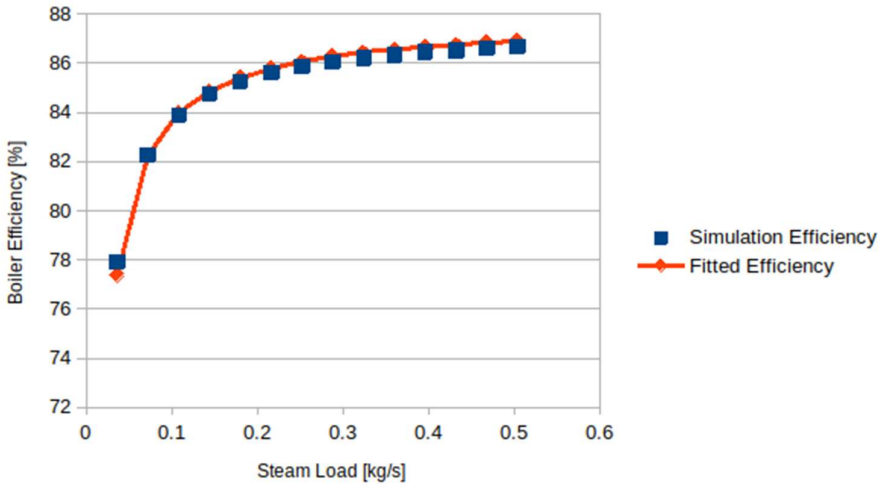

Figure 3.7. Simple part-load boiler efficiency curve

Reasonably estimating boiler efficiency behaviour is impactful to industry as it allows for design improvements to be reasonably evaluated for their energy efficiency. Having a variety of common renewable fuel types like coal and biomass to seamlessly switch between within the same model is also a major benefit to energy transitioning industry.

3.3.2 Future Work

The detail for heat exchanger units was kept simple in this study, limited to estimated overall heat transfer coefficient and surface area parameters. IDAES offers more detailed heat exchanger unit models such as a shell and tube heat exchanger that calculates U and A based on design geometry. One of the challenges to be addressed when using more elaborate unit models is developing an initialisation routine that is robust to various types of heat exchangers. Having flowsheet models initialise reliably with multiple heat exchangers was a challenge in this study. Detailed heat exchangers would extend the offered boiler modelling capabilities to that of a more comprehensive design digital tool. The added detail may also elevate the boiler behaviour to express a characteristic non-monotonic boiler efficiency curve which is closer to reality compared to what was expressed in Figure 3.6.

The current unit and flowsheet models in this study can be integrated into the Ahuora Digital Twin Platform which would extend the model usability with a flow sheeting user interface. In a boiler flowsheet model the user must navigate between different unit models to fully specify the boiler system. Having all required variables within a single packaged boiler unit that abstracts the underlying sub-unit detail would be a more concise user experience. Developing a packaged boiler unit model would use similar methods as the creation of a custom combustion reactor in section 3.2.2. The packaged boiler unit model would differ by including additional control volumes blocks to simulate the sub-unit processes.

A packaged boiler unit would also address the inherent issue with calculating the boiler efficiency at the flowsheet level. A flowsheet-wide boiler system transfers heat across multiple unit operations that must be summed together. The current architecture of the ADTP requires flowsheet level expressions to be manually entered by the user which is unreasonable for such a commonly used metric. An efficiency expression written within a packaged boiler unit would bypass this. An alternative approach would be to analyse heat behaviour of connected units across a flowsheet to calculate boiler system efficiency. This approach is more adaptive to a variety of potential boiler system flowsheets.

3.4 Conclusion

The IDAES framework was used to develop bespoke unit operation models and flowsheet models for boiler systems. These models were developed with the context of intending to integrate them into a digital twin platform such as the ADTP. These models were to be highly beneficial to helping lower technical and economic barriers to the energy transition for industrial boiler steam systems. The models were developed with the IDAES framework by following its code structure hierarchy as

illustrated in Figure 3.1. Lower-level code structure elements were taken advantage of to customise the existing IDAES stoichiometric reactor unit model. The reactor unit model was adapted into a bespoke combustion unit model for the fire-side process of a boiler. This was done by adding variables and constraints that are important to a boiler model. The unit model was then used as part of a boiler system flowsheet model. The boiler flowsheet model was successfully simulated with its boiler efficiency analysed across a range of conditions. A two-parameter equation for a part-load boiler efficiency curve was fitted to a simulated efficiency curve. It showed that the simulated model followed the expected part-load efficiency behaviour. The modelling flexibility, behaviour, and metrics from this boiler model are important for a digital tool solution to industrial boiler decarbonisation.

4 Case Study for Industrial Boiler

4.1 Introduction

Instead of modelling an arbitrary hypothetical boiler system as was done in section 3.2.3, the components developed in this study should be able to accurately model real life boiler systems. This chapter set out to model and analyse a case study of an industrial boiler system. This was useful for both validating the model components developed in previous chapters and for demonstrating their practical use in an industrial context.

The case study used data from a New Zealand pulp and paper mill, specifically its black liquor recovery boiler. This case study was what motivated the creation of black liquor property and reaction packages in Chapter 2. Black liquor is a byproduct of the pulp and paper Kraft process. Its solids are mostly composed of lignin. During gasification black liquor releases heat and produces biogenic CO_2 emissions, making it a renewable process heating solution for pulp and paper mills.

The application possibilities for modelling such a system included scenario analysis of feed costs, system efficiency, and emissions across various conditions. Modelling real systems is also a critical step towards adaptive digital twins that can bi-directionally communicate between itself and its physical counterpart.

4.2 Methods

The data provided was continuous time series data at 5-minute intervals across several years. The data included various measurements of the black liquor recovery boiler. A single 24-hour period was isolated, and its data was averaged to fix the system at a single set of conditions. The types of measurements used were as follows:

- Natural gas flow [NM^3/hr]
- Black liquor flow [L/min]
- Black liquor solids content [wt%]
- Black liquor temperature [°C]
- Feed air flow [NM^3/s]
- Feed air temperature [°C]
- Steam pressure [bara]
- Steam temperature [°C]
- Feed water temperature [°C]

From the data provided a simple process flow diagram was devised as the basis for the developed flowsheet model as shown in Figure 4.1. The data of stream conditions were summarised in Table 4.1

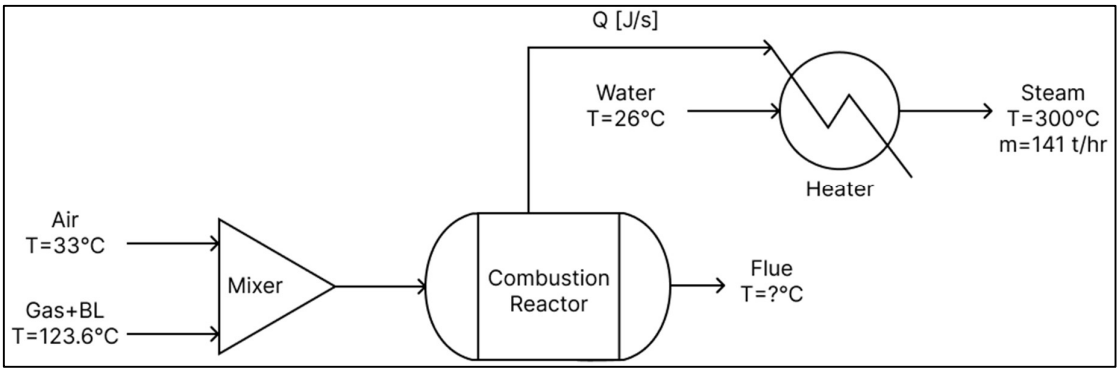

Figure 4.1. Process flow diagram for black liquor recovery boiler case study.

Bespoke correlations for black liquor were used to calculate and fix its important properties. These correlations were retrieved from Fardim & Tikka (2011) and are shown in Equations 2.3 and 4.1. These equations fixed NCV and density with respect to temperature and solids content [wt%] and were assumed to be constant in this model. Enthalpy equations were retrieved from the same source but were omitted. Instead, the heat capacity was assumed to be constant 4.2 [kJ/kg/K] which aligned well with the enthalpy equation.

$$\rho_{BL} = (997 + 649x) * \left(1.008 - 0.237\frac{T}{1000} - 1.94\left(\frac{T}{1000}\right)^2\right) \qquad (4.1)$$

Where:

ρ_{BL} = density of black liquor [kg/m^3]

x = solids content of black liquor [wt%]

T = temperature [°C]

Table 4.1. Summary table of specified stream conditions for the process in Figure 4.1.

Stream	Compound	Mole Flow	Mass Flow	Temperature	Pressure	Molar Weight
		[mol/s]	[t/hr]	[°C]	[bara]	mw [g/mol]
Combustion Feed	Air	975.69	101.76	33	1.013	28.97
	Gas	0.17	0.01	123.6	1.013	16.0425
	BL	998.14	64.75	123.6	1.013	18.02
Combustion Flue	Flue	1288	166.52	195.66	1.013	-

Water feed	Water	2177.69	141.23	136.39	45	18.015
Steam Gen.	Steam	2177.69	141.23	400	45	18.015

This case study flowsheet model used the combustion reactor developed in Chapter 3 was stripped down of the added features that were not necessary here. The air flow was directly fixed from the data without A/F ratio or excess air specifications. There was no need for variable ash content. And there was no data provided from the case study to characterise convection boiler heat loss. Instead of transferring heat by a heat exchanger, heat was directly transferred via the reactor heat duty by using a constraint. Part of the heat duty was additional heat loss which was solved for. Heat was then assumed constant for when solving across a range of flue stack temperatures.

4.3 Discussion

4.3.1 Results

The developed flowsheet was able to accurately model the case study based on the data provided. The model solved the outlet flue temperature for the given case study conditions. The model was then simulated across a range of flue stack temperatures to analyse its impact on boiler efficiency in Figure 4.2. This demonstrated the use of this tool to evaluate a baseline simulation of real boiler system conditions, which could then be developed upon by analysing the impact of changing different parameters. In this case, variable flue stack temperature was explored to mimic the impact that improved flue-steam heat transfer could achieve.

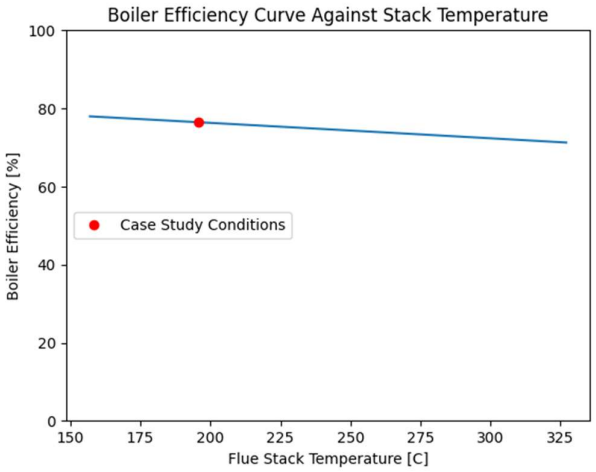

Figure 4.2. Boiler efficiency curve against flue stack outlet temperature. The efficiency of the case study conditions was marked in red.

Overall, it was a simple and streamlined workflow to take the gathered case study data, construct the flowsheet, and enter the data to solve a flexible simulation of a real boiler system. Being able to easily model the real system in the digital space allows industry to quickly adapt different scenarios on existing systems for optimisation and fuel switching.

4.3.2 Future Work

In future work, more data could be gathered for the case study to analyse modelling accuracy across a wider range of reference conditions. Improved accuracy would increase the validity and reliability of this digital modelling tool for informing real engineering design decisions. Comparing against dynamic time-series data would also give the opportunity for modelling dynamic system behaviour. For this to be done, more advanced unit operations would have to be used that facilitate constraints for dynamic behaviour. In the current implementation, ‘multi steady state' simulations could be done, but this does not take into consideration real dynamic relations between variables.

One aspect of digital twins that could be explored is live data feeds entering and updating the model in real time. This would look like on-site edge-computing and would help with model-predictive control and overall design decision making. Much more work will have to be put into the software implementation of the models developed in this study for this to be successful.

4.4 Conclusion

A case study on a real industrial black liquor recovery boiler was done in Chapter 4. The purpose was to validate the model for simulation on real system data and to show potential of analysing design improvements of the system. A process flow diagram was drawn based on the data retrieved. Then a flowsheet model of that process was developed, it was simplified to account scope of measurements from the case study. The boiler efficiency and heat losses were calculated from the conditions of the case study. Then the flue stack temperature was variated across a range to show impact on boiler efficiency (Figure 4.2). This demonstration of the models developed in this study on a real case study is a critical first step towards a full digital twin implementation with bi-directional data streams between physical system and digital model.

5 Conclusion and Future Work

This project goal was to find if novel renewable processes could be developed into a flexible digital modelling tool for industrial steam boiler systems. The motivation was to lower the technologic and economic barriers to industrial decarbonisation for New Zealand industries. Such digital tools for decarbonisation would allow model exploration for optimisation and renewable fuel switching. This project made successful contributions towards its goal by using the IDAES-PSE framework to create custom property packages, reaction packages, and unit operation models to model realistic boiler behaviour in a process flowsheet model. The foundations of accuracy for a boiler system lay in the property and reaction packages where significant attention and detail were given, especially where estimations were made for novel renewable fuels. This project modelled the combustion reaction process for coal, wood biomass, and black liquor, all of which are necessary to model for industrial process heat decarbonisation. The same methodology could go on to be used for other renewable fuels.

The boiler models developed in this study could simulate realistic boiler system behaviour that streamlined access to renewable fuel packages and modelling of real systems. The models possessed advantages for renewable boiler modelling compared to commercial flowsheet simulators by having accessible renewable fuel compounds for combustion. And it possessed advantages over commercial spreadsheeting by having advanced solving capabilities for flowsheet models. And by packaging variables and constraints into properties and unit models, it avoids the often elaborate and bulky nature of spreadsheet models.

As the contributions made were software solutions, new features and updated detail of the models can be added over continual development. Some thought was given to the integration of these models to the Ahuora Digital Twin Platform (ADTP), but as this platform was an ongoing project further and up-to-date work must be done for full and maintained integration. This project contributes to the advancement to Industry 4.0 of a more interconnected and sustainable process engineering future.

Bibliography

About CAPE-OPEN | the CAPE-OPEN Laboratories Network. (n.d.). Retrieved September 13, 2025, from https://www.colan.org/general-information-on-co-lan/

Ahuora. (2025). *Ahuora Adaptive Digital Twin Platform* [Computer software]. https://github.com/waikato-ahuora-smart-energy-systems

AL-Kharabsheh, B. N., Arbili, M. M., Majdi, A., Ahmad, J., Deifalla, A. F., & Hakamy, A. (2022a). A Review on Strength and Durability Properties of Wooden Ash Based Concrete. *Materials*, *15*(20), 7282. https://doi.org/10.3390/ma15207282

AL-Kharabsheh, B. N., Arbili, M. M., Majdi, A., Ahmad, J., Deifalla, A. F., & Hakamy, A. (2022b). A Review on Strength and Durability Properties of Wooden Ash Based Concrete. *Materials*, *15*(20), 7282. https://doi.org/10.3390/ma15207282

Brinkmann, S., & Seyfang, B. C. (2024). Building a Code-Based Model to Describe Syngas Production from Biomass. *ChemEngineering*, *8*(5). https://doi.org/10.3390/chemengineering8050094

Bynum, M. L., Hackebeil, G. A., Hart, W. E., Laird, C. D., Nicholson, B. L., Siirola, J. D., Watson, J.-P., & Woodruff, D. L. (2021). *Pyomo—Optimization Modeling in Python* (Vol. 67). Springer International Publishing. https://doi.org/10.1007/978-3-030-68928-5

Chase, Jr., M. W. (1998). *NIST-JANAF Thermochemical Tables, Fourth Edition, Monograph 9 (Part I and Part II)*. American Chemical Society and American Institute of Physics for the National Institute of Standards and Technology. https://www.nist.gov/data/janaf-thermochemical-tables

de Beer, J., & Depew, C. (2021). The role of process engineering in the digital transformation. *Computers & Chemical Engineering*, *154*, 107423. https://doi.org/10.1016/j.compchemeng.2021.107423

Dongre, P., Nagardeolekar, A., Corbett, D., & Bujanovic, B. M. (2024). Chemical aspects of the composite structure of wood and its recalcitrance to enzymatic hydrolysis. In *Sustainable Biorefining of Woody Biomass to Biofuels and Biochemicals* (pp. 1–41). Elsevier. https://doi.org/10.1016/B978-0-323-91187-0.00012-6

EECA. (2020). *Fonterra coal boiler conversion*. Fonterra coal boiler conversion

EECA. (2023). *Energy End Use Database*. EECA. https://www.eeca.govt.nz/insights/data-tools/energy-end-use-database/

Fardim, P., & Tikka, P. (2011). *Chemical pulping* (2nd ed). Paper Engineers' Association/Paperi ja Puu Oy.

Green, D. W., & Perry, R. H. (Eds.). (2007). *Perry's Chemical Engineers' Handbook* (8th ed.). McGraw-Hill Professional.

Han, Y., Shen, B., & Zhang, T. (2017). A Techno-economic Assessment of Fuel Switching Options of Addressing Environmental Challenges of Coal-Fired Industrial Boilers: An analytical work for China. *Energy Procedia*, *142*, 3083–3087. https://doi.org/10.1016/j.egypro.2017.12.448

Helbig, T., & Mawdsley, I. (2021). *Reporting of biomass in CRF tables and reallocation of emissions from spent liquor combustion*.

International Association for the Properties of Water and Steam (IAPWS). (2018). *Revised Release on the IAPWS Formulation 1995 for the Thermodynamic Properties of Ordinary Water Substance for General and Scientific Use* (No. IAPWS R6-95(2018)). International Association for the Properties of Water and Steam. https://iapws.org/private/downloads/k3PSK5LkHCEfXuMarafO6Q/IAPWS95-2018.pdf

IPCC. (2006). *IPCC Guidelines for National Greenhouse Gas Inventories*.

Jing, D., Meng, X., Ge, S., Zhang, T., Ma, M., & Wang, G. (2022). Structural Model Construction and Optimal Characterization of High-Volatile Bituminous Coal Molecules. *ACS Omega*, *7*(22), 18350–18360. https://doi.org/10.1021/acsomega.2c00505

Kartal, F., & Özveren, U. (2021). A comparative study for biomass gasification in bubbling bed gasifier using Aspen HYSYS. *Bioresource Technology Reports*, *13*, 100615. https://doi.org/10.1016/j.biteb.2020.100615

Lee, A., Ghouse, J. H., Eslick, J. C., Laird, C. D., Siirola, J. D., Zamarripa, M. A., Gunter, D., Shinn, J. H., Dowling, A. W., Bhattacharyya, D., Biegler, L. T., Burgard, A. P., & Miller, D. C. (2021). The IDAES process modeling framework and model library—Flexibility for process simulation and optimization. *Journal of Advanced Manufacturing and Processing*, *3*(3), e10095. https://doi.org/10.1002/amp2.10095

Leśniak, B., Słupik, L., & Jakubina, G. (2013). The determination of the specific heat capacity of coal based on literature data. *Chemik*, *67*, 560–571.

Mares, T. E. (2009). *An investigation of the relationship between coal and gas properties in the Huntly coalfield, New Zealand*. University of Canterbury.

MBIE. (2016). *Industrial Bioenergy Use—Updated methodology to estimate demand*. https://www.mbie.govt.nz/dmsdocument/125-industrial-bioenergy-demand-methodology-2016-pdf

Ministry of Business, I., & Employment. (2022). *New Zealand Emissions Reduction Plan*. Ministry of Business, Innovation & Employment (MBIE).

Naqvi, M., Yan, J., & Dahlquist, E. (2010). Black liquor gasification integrated in pulp and paper mills: A critical review. *Bioresource Technology*, *101*(21), 8001–8015. https://doi.org/10.1016/j.biortech.2010.05.013

Park, T., Hashimoto, H., Wang, W., Thrasher, B., Michaelis, A. R., Lee, T., Brosnan, I. G., & Nemani, R. R. (2023). What Does Global Land Climate Look Like at 2°C Warming? *Earth's Future*, *11*(5), e2022EF003330. https://doi.org/10.1029/2022EF003330

Semeraro, C., Lezoche, M., Panetto, H., & Dassisti, M. (2021). Digital twin paradigm: A systematic literature review. *Computers in Industry*, *130*, 103469. https://doi.org/10.1016/j.compind.2021.103469

Steffen, W., Grinevald, J., Crutzen, P., & Mcneill, J. (2011). The anthropocene: Conceptual and historical perspectives. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *369*(1938), 842–867. https://doi.org/10.1098/rsta.2010.0327

The Paris Agreement | UNFCCC. (n.d.). Retrieved September 11, 2025, from https://unfccc.int/process-and-meetings/the-paris-agreement

Wächter, A., & Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, *106*(1), 25–57. https://doi.org/10.1007/s10107-004-0559-y

Appendix A1. Weighted Properties of Wood Ash

Component	Wood Ash Compositions from Literature[A]					Average	Cp@ 1000[K][B]	Weighted Cp	mw	Weighted mw
							[J/mol/K]	[J/mol/K]	[g/mol]	[g/mol]
SiO_2	48.96	25	55.52	2.7	29.1	32.3	69	2225.7	60.1	1938.1
Al_2O_3	11.24	0.76	3.11	1.3	10.3	5.3	125	667.8	144.9	774.3
CaO	11.59	35.6	9.92	61	35.6	30.7	54	1660.1	56.1	1723.9
Fe_2O_3	0.6	2.76	0.41	1.3	6.55	2.3	150	348.6	159.7	371.1
MgO	5.05	1.76	2.32	8.7	4.7	4.5	51	229.8	40.3	181.6
Sum:	77.44	65.88	71.28	75	86.25	75.2		5131.9		4989.0
Weighted:	1.03	0.88	0.95	1.00	1.15	1		68.27043		66.36964

Note.

[A] (AL-Kharabsheh et al., 2022b)

[B] (Chase, 1998)

Appendix A2. Verification Spreadsheet Model

Inlet	mole rate	mole frac	Cp (J/mol/K)	temp K	A	B	C	D	E
biomass	0.616770128	0.026744	243.2091	400					
o2	4.71341156	0.20438	28.31890	400	30.03235	8.772972	-3.98813	0.788313	-0.7416
co2	0	0	41.33622	400	24.99735	55.18696	-33.6914	7.948387	-0.13664
h2o	0	0	34.26312	400	30.092	6.832514	6.793435	-2.53448	0.082139
n2	17.73167994	0.76887	29.47006	400	19.50583	19.88705	-8.59854	1.369784	0.527601
total(mol/s)	23.062		34.9508427						
h			13980.3371	J/mol					
H			322414.534	J/s					
Outlet	mole rate	mole frac	Cp (J/mol/K)	temp K	A	B	C	D	E
biomass	1.68897E-05	6.6159E-07	243.2091	573.15					
o2	1.01289213	0.03967633	31.64137	573.15	30.03235	8.772972	-3.98813	0.788313	-0.7416
co2	3.70051943	0.14495427	46.64069	573.15	24.99735	55.18696	-33.6914	7.948387	-0.13664
h2o	3.083766191	0.12079523	36.01255	573.15	30.092	6.832514	6.793435	-2.53448	0.082139
n2	17.73167994	0.69457351	29.94346	573.15	19.50583	19.88705	-8.59854	1.369784	0.527601
total(mol/s)	25.52887458		33.164						
h			19,008	J/mol					
H			485,258	J/s					
dh_heating	139565.1912	J/s							
dh_rxn	-1695798	J/s							
excel	-1556233	J/s							
idaes	-1553900	J/s							
	-0.15%	%error							

Appendix B1. All-Compounds Property Package

```
"""
Property package of all compounds used in this study
-biomass
-subbituminous coal
-black liquor
-ash
-O2
-CH4
-CO2
-H2O
-N2
"""
# Import Pyomo units
from pyomo.environ import units as pyunits

# Import IDAES cores
from idaes.core import LiquidPhase, VaporPhase, Component, SolidPhase
import idaes.logger as idaeslog

from idaes.models.properties.modular_properties.state_definitions import FTPx
from idaes.models.properties.modular_properties.eos.ideal import Ideal
from idaes.models.properties.modular_properties.pure import ConstantProperties, ChapmanEnskog
from idaes.models.properties.modular_properties.phase_equil import SmoothVLE
from idaes.models.properties.modular_properties.phase_equil.bubble_dew import (
    IdealBubbleDew,
)
from idaes.models.properties.modular_properties.phase_equil.forms import fugacity
from idaes.models.properties.modular_properties.pure import Perrys
from idaes.models.properties.modular_properties.pure import RPP4
from idaes.models.properties.modular_properties.pure import NIST
from idaes.core import PhaseType as PT


# Set up logger
_log = idaeslog.getLogger(__name__)


configuration = {
    # Specifying components
    "components": {
        "biomass": { #woody biomass
            "type": Component,
            "elemental_composition": {"C":6, "H": 10, "O": 5}, #cellulose composition C6-H10-O5
            "enth_mol_sol_comp": ConstantProperties.Constant,
            "cp_mol_sol_comp": ConstantProperties.Constant,
            "dens_mol_sol_comp": ConstantProperties.Constant,
            "visc_d_phase_comp": {"Sol": ConstantProperties.Constant},
            'valid_phase_types': PT.solidPhase,
```

```
    "parameter_data": {
        "mw": (162.1394, pyunits.g / pyunits.mol),
        "cp_mol_sol_comp_coeff": (243.2091, pyunits.J/pyunits.mol/pyunits.K), #1500 J/kg/K
        "dens_mol_sol_comp_coeff": (2960.415544, pyunits.mol/pyunits.m**3),
        "enth_mol_form_sol_comp_ref": (0, pyunits.kJ/pyunits.mol),
        "enrt_mol_form_sol_comp_ref": (158.1, pyunits.J/pyunits.mol/pyunits.K),
        "visc_d_Sol_comp_coeff": (3.2833e-05, pyunits.Pa*pyunits.s)
    },
},
"coal": {
    "type": Component,
    "elemental_composition": {"coal":1},
    "enth_mol_sol_comp": ConstantProperties.Constant,
    "cp_mol_sol_comp": ConstantProperties.Constant,
    "dens_mol_sol_comp": ConstantProperties.Constant,
    'valid_phase_types': PT.solidPhase,
    "parameter_data": {
        "mw": (11.8615, pyunits.g / pyunits.mol),
        "cp_mol_sol_comp_coeff": (15.4199, pyunits.J/pyunits.mol/pyunits.K),
        "dens_mol_sol_comp_coeff": (109598.6169, pyunits.mol/pyunits.m**3),
        "enth_mol_form_sol_comp_ref": (0, pyunits.kJ/pyunits.mol),
        "enrt_mol_form_sol_comp_ref": (0, pyunits.J/pyunits.mol/pyunits.K),
    },
},
"BL": { #Black Liquor
    "type": Component,
    "elemental_composition":{"BL":1},
    "cp_mol_liq_comp": ConstantProperties.Constant,
    "enth_mol_liq_comp": ConstantProperties.Constant,
    "dens_mol_liq_comp": ConstantProperties.Constant,
    'valid_phase_types': PT.liquidPhase,
    "parameter_data": {
        "mw": (18.02, pyunits.g / pyunits.mol),
        "cp_mol_liq_comp_coeff": (75.684, pyunits.J/pyunits.mol/pyunits.K),
        "dens_mol_liq_comp_coeff": (76471.01, pyunits.mol/pyunits.m**3),
        "enth_mol_form_liq_comp_ref": (0, pyunits.kJ/pyunits.mol),
    },
},
"ash": { #wood ash
    "type": Component,
    "elemental_composition": {"ash":1},
    "cp_mol_sol_comp": ConstantProperties.Constant,
    "enth_mol_sol_comp": ConstantProperties.Constant,
    "dens_mol_sol_comp": ConstantProperties.Constant,
    'valid_phase_types': PT.solidPhase,
    "parameter_data": {
        "mw": (66.37, pyunits.g / pyunits.mol),
        "cp_mol_sol_comp_coeff": (68.27, pyunits.J/pyunits.mol/pyunits.K),
        "dens_mol_sol_comp_coeff": (2960.415544, pyunits.mol/pyunits.m**3),
        "enth_mol_form_sol_comp_ref": (0, pyunits.kJ/pyunits.mol),
```

```
        "enrt_mol_form_sol_comp_ref": (158.1, pyunits.J/pyunits.mol/pyunits.K),
    },
},
"O2": {
    "type": Component,
    "elemental_composition": {"O":2},
    "enth_mol_ig_comp": NIST,
    "cp_mol_ig_comp": NIST,
    'valid_phase_types': PT.vaporPhase,
    "parameter_data": {
        "mw": (31.9988, pyunits.g / pyunits.mol),
        "cp_mol_ig_comp_coeff": {  #valid range  700-2000
            "A": (30.03235  , pyunits.J / pyunits.mol / pyunits.K),
            "B": (8.772972, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-1),
            "C": (-3.988133, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-2),
            "D": (0.788313, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-3),
            "E": (-0.741599, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**2),
            "F": (-11.32468, pyunits.kJ / pyunits.mol),
            "G": (236.1663, pyunits.J / pyunits.mol /pyunits.K),
            "H": (0, pyunits.kJ / pyunits.mol)
        },
    },
},
"CH4": {
    "type": Component,
    "elemental_composition": {"C": 1, "H": 4},
    "enth_mol_ig_comp": NIST,
    "cp_mol_ig_comp": NIST,
    'valid_phase_types': PT.vaporPhase,
    "parameter_data": {
        "mw": (16.0425, pyunits.g / pyunits.mol),
        "cp_mol_ig_comp_coeff": { #valid range 298 K - 1300 K
            "A": (-0.703029 , pyunits.J / pyunits.mol / pyunits.K),
            "B": (108.4773, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-1),
            "C": (-42.52157, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-2),
            "D": (5.862788, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-3),
            "E": (0.678565, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**2),
            "F": (-76.84376, pyunits.kJ / pyunits.mol),
            "G": (158.7163, pyunits.J / pyunits.mol /pyunits.K),
            "H": (-74.87310, pyunits.kJ / pyunits.mol),
        },
    },
},
"CO2": {
    "type": Component,
    "elemental_composition": {"C":1,"O":2, },
    "enth_mol_ig_comp": NIST,
    "cp_mol_ig_comp": NIST,
    'valid_phase_types': PT.vaporPhase,
    "parameter_data": {
```

```
        "mw": (44.0095, pyunits.g / pyunits.mol),
        "pressure_crit": (73.80e5, pyunits.Pa),
        "temperature_crit": (304.18, pyunits.K),
        "cp_mol_ig_comp_coeff": { #valid range 298 K - 1200 K
           "A": (24.99735  , pyunits.J / pyunits.mol / pyunits.K),
           "B": (55.18696, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-1),
           "C": (-33.69137, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-2),
           "D": (7.948387, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-3),
           "E": (-0.136638, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**2),
           "F": (-403.6075, pyunits.kJ / pyunits.mol),
           "G": (228.2431, pyunits.J / pyunits.mol /pyunits.K),
           "H": (-393.5224, pyunits.kJ / pyunits.mol),
        },
    },
},
"H2O": {
   "type": Component,
   "elemental_composition":{"H":2,"O":1},
   "enth_mol_ig_comp": NIST,
   "cp_mol_ig_comp": NIST,
   'valid_phase_types': PT.vaporPhase,
   "parameter_data": {
      "mw": (18.0153e-3, pyunits.kg / pyunits.mol),
      "cp_mol_ig_comp_coeff": { #valid range 500 K- 1700 K
         "A": (30.09200,pyunits.J / pyunits.mol / pyunits.K,),
         "B": (6.832514,pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-1,),
         "C": (6.793435,pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-2,),
         "D": (-2.534480,pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-3,),
         "E": (0.082139,pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**2,),
         "F": (-250.8810, pyunits.kJ / pyunits.mol),
         "G": (223.3967, pyunits.J / pyunits.mol / pyunits.K),
         "H": (-241.8264, pyunits.kJ / pyunits.mol),
      },
   },
},
"N2": {
    "type": Component,
    "elemental_composition": {"N":2},
    "enth_mol_ig_comp": NIST,
    "cp_mol_ig_comp": NIST,
    "visc_d_phase_comp": {"Vap": ConstantProperties.Constant},
    'valid_phase_types': PT.vaporPhase,
    "parameter_data": {
       "mw": (28.0134, pyunits.g / pyunits.mol),
       "cp_mol_ig_comp_coeff": { #valid range 500 K to 2000 K
          "A": (19.50583  , pyunits.J / pyunits.mol / pyunits.K),
          "B": (19.88705, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-1),
          "C": (-8.598535 , pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-2),
          "D": (1.369784  , pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-3),
          "E": (0.527601, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**2),
```

```
                "F": (-4.935202, pyunits.kJ / pyunits.mol),
                "G": (212.3900, pyunits.J / pyunits.mol /pyunits.K),
                "H": (0, pyunits.kJ / pyunits.mol)
            },
        },
    },
},

# Specifying phases
"phases": {
    "Vap": {"type": VaporPhase, "equation_of_state": Ideal},
    "Sol": {"type": SolidPhase, "equation_of_state": Ideal},
    "Liq": {"type": LiquidPhase, "equation_of_state": Ideal}
},
# Set base units of measurement
"base_units": {
    "time": pyunits.s,
    "length": pyunits.m,
    "mass": pyunits.kg,
    "amount": pyunits.mol,
    "temperature": pyunits.K,
},
# Specifying state definition
"state_definition": FTPx,
"state_bounds": {
    "flow_mol": (0, 100, 1000, pyunits.mol / pyunits.s),
    "temperature": (273.15, 300, 2500, pyunits.K),
    "pressure": (5e3, 1e5, 1e6, pyunits.Pa),
},
"pressure_ref": (1e5, pyunits.Pa),
"temperature_ref": (300, pyunits.K),

"include_enthalpy_of_formation":(False),
}
```

Appendix B2. Biomass and Coal Combustion Reaction Package

```
"""
reaction package for the combustion of biomass, coal, and methane
"""
from pyomo.environ import Expression

# Import Python libraries
import logging

# Import Pyomo libraries
from pyomo.environ import (Constraint,
                           exp,
                           Set,
                           Var,
                           Param,
                           Reals,
                           Any,
                           units as pyunits)

# Import IDAES cores
from idaes.core import (declare_process_block_class,
                        MaterialFlowBasis,
                        ReactionParameterBlock,
                        ReactionBlockDataBase,
                        ReactionBlockBase)
from idaes.core.util.constants import Constants as const
from idaes.core.util.misc import add_object_reference

# Set up logger
_log = logging.getLogger(__name__)


@declare_process_block_class("BMCombReactionParameterBlock")
class BMCombReactionParameterData(ReactionParameterBlock):
    """
    Property Parameter Block Class
    Contains parameters and indexing sets associated with properties for
    superheated steam.
    """

    def build(self):
        '''
        Callable method for Block construction.
        '''
        super(BMCombReactionParameterData, self).build()

        self._reaction_block_class = BMReactionBlock

        # List of valid phases in property package
```

```
self.phase_list = Set(initialize=["Vap", "Sol"])

# Component list - a list of component identifiers
self.component_list = Set(initialize=["H2O",
                                      "CO2",
                                      "O2",
                                      "N2",
                                      "biomass",
                                      "ash",
                                      "coal",
                                      ])

# Reaction Index
self.rate_reaction_idx = Set(initialize=["R1","Rcoal","RCH4"])
self.uncombs_set = Set(initialize=["R1","Rcoal"])

self.reaction_set = Set(initialize=[("R1", "Vap", "H2O"),
                                    ("R1", "Vap", "CO2"),
                                    ("R1", "Vap", "O2"),
                                    ("R1", "Sol", "coal"),
                                    ("R1", "Sol", "biomass"),
                                    ("R1", "Vap", "N2"),
                                    ("R1", "Sol", "ash"),

                                    ("Rcoal", "Vap", "H2O"),
                                    ("Rcoal", "Vap", "CO2"),
                                    ("Rcoal", "Vap", "O2"),
                                    ("Rcoal", "Sol", "coal"),
                                    ("Rcoal", "Sol", "biomass"),
                                    ("Rcoal", "Vap", "N2"),
                                    ("Rcoal", "Sol", "ash"),

                                    ("RCH4", "Vap", "H2O"),
                                    ("RCH4", "Vap", "CO2"),
                                    ("RCH4", "Vap", "O2"),
                                    ("RCH4", "Vap", "CH4"),
                                    ])

self.stoich_init = Param(self.reaction_set, initialize={
                                    ("R1", "Vap", "H2O"): 4.95868,
                                    ("R1", "Vap", "CO2"): 5.95041556,
                                    ("R1", "Vap", "O2"): -6,
                                    ("R1", "Sol", "coal"): 0,
                                    ("R1", "Sol", "biomass"): -1,
                                    ("R1", "Vap", "N2"): 0,
                                    ("R1", "Sol", "ash"): 0.04409448,

                                    ("Rcoal", "Vap", "H2O"): 0.025,
                                    ("Rcoal", "Vap", "CO2"): 0.620971467,
                                    ("Rcoal", "Vap", "O2"): -0.548471467,
```

```
                                ("Rcoal", "Sol", "coal"): -1,
                                ("Rcoal", "Sol", "biomass"): 0,
                                ("Rcoal", "Vap", "N2"): 0.005,
                                ("Rcoal", "Sol", "ash"): 0.005361517,

                                ("RCH4", "Vap", "H2O"):2,
                                ("RCH4", "Vap", "CO2"):1,
                                ("RCH4", "Vap", "O2"):-2,
                                ("RCH4", "Vap", "CH4"):-1,
                                }
                                ,mutable=False)
        self.rate_reaction_stoichiometry = Var(self.reaction_set, initialize=self.stoich_init)
        self.rate_reaction_stoichiometry.fix()

        #fuel dict
        self.limit_reactant_dict = Param(self.rate_reaction_idx, initialize={
           "R1": ("Sol","biomass"),
           "Rcoal": ("Sol","coal"),
           "RCH4": ("Vap","CH4"),
        },
        within=Any)

        dh_rxn_dict = {"R1": -2749556.40, # @ w=9%, h=6% ==> ncv=-2749556.40
                       "Rcoal": -284675.1254, #[J/molCoal]
                       "RCH4": -802125 #based on 50 Mj/kg LHV
                       }

        self.dh_rxn = Var(self.rate_reaction_idx,
                          initialize = dh_rxn_dict,
                          domain=Reals,
                          doc="Heat of reaction")
        self.dh_rxn.fix()

    @classmethod
    def define_metadata(cls, obj):
        obj.add_default_units({'time': pyunits.s,
                               'length': pyunits.m,
                               'mass': pyunits.kg,
                               'amount': pyunits.mol,
                               'temperature': pyunits.K})


class ReactionBlock(ReactionBlockBase):
    """
    This Class contains methods which should be applied to Reaction Blocks as a
    whole, rather than individual elements of indexed Reaction Blocks.
    """
    def initialize(blk, outlvl=0, **kwargs):
        '''
        Initialization routine for reaction package.
```

```
        Keyword Arguments:
            outlvl : sets output level of initialization routine
                     * 0 = no output (default)
                     * 1 = report after each step
        Returns:
            None
        '''
        if outlvl > 0:
            _log.info('{} Initialization Complete.'.format(blk.name))


@declare_process_block_class("BMReactionBlock", block_class=ReactionBlock)
class BMReactionBlockData(ReactionBlockDataBase):
    def build(self):
        """
        Callable method for Block construction
        """
        super(BMReactionBlockData, self).build()

        # Heat of reaction - no _ref as this is the actual property
        add_object_reference(
                self,
                "dh_rxn",
                self.config.parameters.dh_rxn)

    def get_reaction_rate_basis(b):
        return MaterialFlowBasis.molar
```

Appendix B3. Black Liquor and Methane Reaction Package

```
"""
reaction package for the combustion of black liquor and methane
"""
from pyomo.environ import Expression

# Import Python libraries
import logging

# Import Pyomo libraries
from pyomo.environ import (Constraint,
                           exp,
                           Set,
                           Var,
                           Param,
                           units as pyunits,
                           Reals,
                           Any)

# Import IDAES cores
from idaes.core import (declare_process_block_class,
                        MaterialFlowBasis,
                        ReactionParameterBlock,
                        ReactionBlockDataBase,
                        ReactionBlockBase)
from idaes.core.util.constants import Constants as const
from idaes.core.util.misc import add_object_reference

# Set up logger
_log = logging.getLogger(__name__)


@declare_process_block_class("MultiCombReactionParameterBlock")
class MultiCombReactionParameterData(ReactionParameterBlock):
    """
    Property Parameter Block Class
    Contains parameters and indexing sets associated with properties for
    superheated steam.
    """

    def build(self):
        '''
        Callable method for Block construction.
        '''
        super(MultiCombReactionParameterData, self).build()

        self._reaction_block_class = BMReactionBlock

        # List of valid phases in property package
```

```
        self.phase_list = Set(initialize=['Vap', 'Sol'])

        # Component list - a list of component identifiers
        self.component_list = Set(initialize=['H2O',
                                              'CO2',
                                              'O2',
                                              'N2',
                                              'BL',
                                              'uncombustible',
                                              'CH4'
                                              ])

        # Reaction Index
        self.rate_reaction_idx = Set(initialize=["Rbl","RCH4"])
        self.uncombs_set = Set(initialize=["Rbl",])

        self.reaction_set = Set(initialize=[("Rbl", "Vap", "H2O"),
                                            ("Rbl", "Vap", "CO2"),
                                            ("Rbl", "Vap", "O2"),
                                            ("Rbl", "Liq", "BL"),
                                            ("Rbl", "Vap", "N2"),
                                            ("Rbl", "Sol", "uncombustible"),
                                            ("Rbl", "Vap", "CH4"),

                                            ("RCH4", "Vap", "H2O"),
                                            ("RCH4", "Vap", "CO2"),
                                            ("RCH4", "Vap", "O2"),
                                            ("RCH4", "Liq", "BL"),
                                            ("RCH4", "Vap", "N2"),
                                            ("RCH4", "Sol", "uncombustible"),
                                            ("RCH4", "Vap", "CH4"),
                                            ])

        # Reaction Stoichiometry
        self.rate_reaction_stoichiometry = Var(self.reaction_set, initialize={
                                        ("Rbl", "Vap", "H2O"): 1-0.7-(0.875*0.3), # carryover water subtract(H2O
consumed based on Co2 emit.)
                                         ("Rbl", "Vap", "CO2"): 0.3, #based on assumed black liquor emissions
factor of 95.3 kgCO2/GJ https://naturvardsverket.diva-
portal.org/smash/get/diva2:1546963/FULLTEXT01.pdf
                                            ("Rbl", "Vap", "O2"): -0.3+(0.15*0.875), #0.3 based on Co2 emit.
add(Oxygen consumed/supplied for H2O)
                                            ("Rbl", "Liq", "BL"): -1,
                                            ("Rbl", "Vap", "N2"): 0,
                                           ("Rbl", "Sol", "uncombustible"): 0.143745, #goal seek mass balance with
other stoichs
                                            ("Rbl", "Vap", "CH4"): 0,

                                            ("RCH4", "Vap", "H2O"): 2,
                                            ("RCH4", "Vap", "CO2"): 1,
```

```
                                ("RCH4", "Vap", "O2"): -2,
                                ("RCH4", "Liq", "BL"): 0,
                                ("RCH4", "Vap", "N2"): 0,
                                ("RCH4", "Sol", "uncombustible"): 0,
                                ("RCH4", "Vap", "CH4"): -1,
                                })
        self.rate_reaction_stoichiometry.fix()

        # self.reactant_list=Set(initialize=["biomass","O2",'CH4'])
        self.limit_reactant_dict = Param(self.rate_reaction_idx, initialize={
           "Rbl": "BL",
           "RCH4": "CH4",
        },
        within=Any)

        dh_rxn_dict = {"Rbl": -135150,
                       "RCH4": -802125
                       }

        self.dh_rxn = Var(self.rate_reaction_idx,
                          initialize = dh_rxn_dict,
                          domain=Reals,
                          doc="Heat of reaction",
                          units=pyunits.J/pyunits.mol)
        self.dh_rxn.fix()


    @classmethod
    def define_metadata(cls, obj):
        obj.add_default_units({'time': pyunits.s,
                               'length': pyunits.m,
                               'mass': pyunits.kg,
                               'amount': pyunits.mol,
                               'temperature': pyunits.K})


class ReactionBlock(ReactionBlockBase):
    """
    This Class contains methods which should be applied to Reaction Blocks as a
    whole, rather than individual elements of indexed Reaction Blocks.
    """
    def initialize(blk, outlvl=0, **kwargs):
        '''
        Initialization routine for reaction package.
        Keyword Arguments:
            outlvl : sets output level of initialization routine
                    * 0 = no output (default)
                    * 1 = report after each step
        Returns:
            None
```

```
        '''
        if outlvl > 0:
            _log.info('{} Initialization Complete.'.format(blk.name))


@declare_process_block_class("BMReactionBlock", block_class=ReactionBlock)
class BMReactionBlockData(ReactionBlockDataBase):
    def build(self):
        """
        Callable method for Block construction
        """
        super(BMReactionBlockData, self).build()

        # Heat of reaction - no _ref as this is the actual property
        add_object_reference(
                self,
                "dh_rxn",
                self.config.parameters.dh_rxn)

    def get_reaction_rate_basis(b):
        return MaterialFlowBasis.molar
```

Appendix B4. Verification Model (see Section 2.3.3)

```
# Verification test for biomass combustion stoichiometry

from idaes.core.util.model_statistics import degrees_of_freedom
import idaes.logger as idaeslog

from pyomo.environ import ConcreteModel, SolverFactory, TransformationFactory, value
from pyomo.network import Arc
from idaes.core import FlowsheetBlock
from idaes.models.properties.modular_properties import GenericParameterBlock
from bm_comb_properties import configuration
from bm_comb_rp import BMCombRxnParameterBlock
from idaes.models.unit_models import StoichiometricReactor,Mixer

m = ConcreteModel()
m.fs = FlowsheetBlock(dynamic=False)

m.fs.properties = GenericParameterBlock(**configuration)
m.fs.reaction = BMCombRxnParameterBlock(property_package=m.fs.properties)
m.fs.react = StoichiometricReactor(
    property_package = m.fs.properties,
    reaction_package = m.fs.reaction,
    has_heat_of_reaction=True,
    has_heat_transfer=True,
    has_pressure_change=False
)


M_bm = 100 # [g/s]
FAratio = 6.5
mw_air = 28.96 #[g/mol]
M_air = M_bm*FAratio
N_air = M_air/mw_air

ash_wt=0.02
w_bm = 0.09
mw_bm=configuration["components"]["biomass"]["parameter_data"]["mw"][0]
mw_ash=configuration["components"]["uncombustible"]["parameter_data"]["mw"][0]

N_bm = (M_bm/mw_bm)
N_ash = ash_wt*(1-w_bm)*M_bm/mw_ash
stoich_ash = N_ash/N_bm
N_total = N_bm + N_air

#adjusting biomass combustion stoichiometry for incombustible content:
new_co2 = 6*11/(11+stoich_ash)
new_h2o = 5*11/(11+stoich_ash)
```

```
m.fs.react.config.reaction_package.rate_reaction_stoichiometry["Rbiomass", "Sol",
"uncombustible"].fix(stoich_ash)
m.fs.react.config.reaction_package.rate_reaction_stoichiometry["Rbiomass", "Vap",
"CO2"].fix(new_co2)
m.fs.react.config.reaction_package.rate_reaction_stoichiometry["Rbiomass", "Vap",
"H2O"].fix(new_h2o)

#mole_frac_comp spec
m.fs.react.inlet.mole_frac_comp[0,"O2"].fix(N_air*0.21/N_total)
m.fs.react.inlet.mole_frac_comp[0,"N2"].fix(N_air*0.79/N_total)
m.fs.react.inlet.mole_frac_comp[0,"CH4"].fix(1e-20)
m.fs.react.inlet.mole_frac_comp[0,"CO2"].fix(1e-20)
m.fs.react.inlet.mole_frac_comp[0,"H2O"].fix(1e-20)
m.fs.react.inlet.mole_frac_comp[0,"biomass"].fix(N_bm/N_total)
m.fs.react.inlet.mole_frac_comp[0,"uncombustible"].fix(1e-20)
m.fs.react.inlet.flow_mol.fix(N_total)
m.fs.react.inlet.temperature.fix(400)
m.fs.react.inlet.pressure.fix(101325)

m.fs.react.rate_reaction_extent[0,"RCH4"].fix(0)
m.fs.react.rate_reaction_extent[0,"Rbiomass"].fix(N_bm)

# m.fs.react.heat_duty[0].fix(0)
m.fs.react.outlet.temperature.fix(300+273.15)

print(degrees_of_freedom(m))
assert degrees_of_freedom(m) == 0
m.fs.react.initialize(outlvl=idaeslog.INFO)
solver=SolverFactory("ipopt")
status=solver.solve(m,tee=True)
m.fs.react.report()
# m.fs.react.display()

# print(value(m.fs.react.rate_reaction_extent))
print(m.fs.reaction.rate_reaction_stoichiometry["Rbiomass", "Vap", "O2"])
print(value(m.fs.properties.config.components["CO2"]["parameter_data"]["mw"][0]))
```

Appendix B5. Shomate Parameter Validation (see Section 2.3.4)

```
import numpy as np

#stoich_reactor_test
from pyomo.environ import (
    ConcreteModel,
    SolverFactory,
    value,
    units as pyunits
)
```

```
#Todo add the four other unit operations
from idaes.models.unit_models import (Pump)
from idaes.core.util.model_statistics import degrees_of_freedom
from idaes.core import FlowsheetBlock
# Import idaes logger to set output levels
import idaes.logger as idaeslog
from idaes.models.properties.modular_properties import GenericParameterBlock
# from  biomass_comb_pp import configuration


# Import Pyomo units
from pyomo.environ import units as pyunits

# Import IDAES cores
from idaes.core import VaporPhase, Component
import idaes.logger as idaeslog

from idaes.models.properties.modular_properties.state_definitions import FTPx
from idaes.models.properties.modular_properties.eos.ideal import Ideal

from idaes.models.properties.modular_properties.pure import NIST
from idaes.core import PhaseType as PT


# Set up logger
_log = idaeslog.getLogger(__name__)


#modular property package of just NIST ideal gases
configuration = {
    "include_enthalpy_of_formation":(False),
    # Specifying components
    "components": {
        "CH4": {
            "type": Component,
            "elemental_composition": {"C": 1, "H": 4},
            "enth_mol_ig_comp": NIST,
            "cp_mol_ig_comp": NIST,
            'valid_phase_types': PT.vaporPhase,
            "parameter_data": {
                "mw": (16.0425, pyunits.g / pyunits.mol),  # [4]
                "pressure_crit": (46.1e5, pyunits.Pa),  # [[4]
                "temperature_crit": (190.6, pyunits.K),  # [4]
                "cp_mol_ig_comp_coeff": { #valid range 298 K - 1300 K
                    "A": (-0.703029 , pyunits.J / pyunits.mol / pyunits.K),  # [4]
                    "B": (108.4773, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-1),
                    "C": (-42.52157, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-2),
                    "D": (5.862788, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-3),
                    "E": (0.678565, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**2),
                    "F": (-76.84376, pyunits.kJ / pyunits.mol),
                    "G": (158.7163, pyunits.J / pyunits.mol /pyunits.K),
```

```
            "H": (-74.87310, pyunits.kJ / pyunits.mol),
        },
    },
},

"O2": {
    "type": Component,
    "elemental_composition": {"O":2},
    "enth_mol_ig_comp": NIST,
    "cp_mol_ig_comp": NIST,
    'valid_phase_types': PT.vaporPhase,
    "parameter_data": {
        "mw": (31.9988, pyunits.g / pyunits.mol),  # [4]
        "pressure_crit": (50.43e5, pyunits.Pa),  # [8]
        "temperature_crit": (154.58, pyunits.K),  # [8]
        "cp_mol_ig_comp_coeff": { # valid range 100 K - 700 K
            "A": (31.32234  , pyunits.J / pyunits.mol / pyunits.K),  # [4]
            "B": (-20.23531, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-1),
            "C": (57.86644, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-2),
            "D": (-36.50624, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-3),
            "E": (-0.007374, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**2),
            "F": (-8.903471, pyunits.kJ / pyunits.mol),
            "G": (246.7945, pyunits.J / pyunits.mol /pyunits.K),
            "H": (0, pyunits.kJ / pyunits.mol)
            #valid range 700 K - 2000 K
            # "A": (30.03235    , pyunits.J / pyunits.mol / pyunits.K),  # [4] #valid range  700-2000
            # "B": (8.772972, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-1),
            # "C": (-3.988133, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-2),
            # "D": (0.788313, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-3),
            # "E": (-0.741599, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**2),
            # "F": (-11.32468, pyunits.kJ / pyunits.mol),
            # "G": (236.1663, pyunits.J / pyunits.mol /pyunits.K),
            # "H": (0, pyunits.kJ / pyunits.mol)
        },
        "enth_mol_form_vap_comp_ref": (0, pyunits.kJ / pyunits.mol),

    },
},


"CO2": {
    "type": Component,
    "elemental_composition": {"C":1,"O":2, },
    "enth_mol_ig_comp": NIST,
    "cp_mol_ig_comp": NIST,
    'valid_phase_types': PT.vaporPhase,
    "parameter_data": {
        "mw": (44.0095, pyunits.g / pyunits.mol),  # [4]
        "pressure_crit": (73.80e5, pyunits.Pa),  # [[6]
        "temperature_crit": (304.18, pyunits.K),  # [6]
```

```
        "cp_mol_ig_comp_coeff": { #valid range 298 K - 1200 K
            "A": (24.99735  , pyunits.J / pyunits.mol / pyunits.K),  # [4]
            "B": (55.18696, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-1),
            "C": (-33.69137, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-2),
            "D": (7.948387, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-3),
            "E": (-0.136638, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**2),
            "F": (-403.6075, pyunits.kJ / pyunits.mol),
            "G": (228.2431, pyunits.J / pyunits.mol /pyunits.K),
            "H": (-393.5224, pyunits.kJ / pyunits.mol),
        },

    },
},
"H2O": {
    "type": Component,
    "elemental_composition":{"H":2,"O":1},
    "enth_mol_ig_comp": NIST,
    "cp_mol_ig_comp": NIST,
    'valid_phase_types': PT.vaporPhase,
    "parameter_data": {
        "mw": (18.0153e-3, pyunits.kg / pyunits.mol),
        "cp_mol_ig_comp_coeff": { #valid range 500 K- 1700 K
            "A": (30.09200,pyunits.J / pyunits.mol / pyunits.K,),  # [4]
            "B": (6.832514,pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-1,),
            "C": (6.793435,pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-2,),
            "D": (-2.534480,pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-3,),
            "E": (0.082139,pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**2,),
            "F": (-250.8810, pyunits.kJ / pyunits.mol),
            "G": (223.3967, pyunits.J / pyunits.mol / pyunits.K),
            "H": (-241.8264, pyunits.kJ / pyunits.mol),
        },
    },
},

"N2": {
    "type": Component,
    "elemental_composition": {"N":2},
    "enth_mol_ig_comp": NIST,
    "cp_mol_ig_comp": NIST,
    'valid_phase_types': PT.vaporPhase,
    "parameter_data": {
        "mw": (28.0134, pyunits.g / pyunits.mol),  # [4]
        "pressure_crit": (33.978e5, pyunits.Pa),  # [[7]
        "temperature_crit": (126.19, pyunits.K),  # [7]
        # "cp_mol_ig_comp_coeff": { #valid range 100 K to 500 K
        #     "A": (28.98641        , pyunits.J / pyunits.mol / pyunits.K),  # [4]
        #     "B": (1.853978, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-1),
        #     "C": (-9.647459   , pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-2),
        #     "D": (16.63537    , pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-3),
        #     "E": (0.000117, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**2),
```

```
            #     "F": (-8.671914, pyunits.kJ / pyunits.mol),
            #     "G": (226.4168, pyunits.J / pyunits.mol /pyunits.K),
            #     "H": (0, pyunits.kJ / pyunits.mol)
            # },
            "cp_mol_ig_comp_coeff": { #valid range 500 K to 2000 K
                "A": (19.50583  , pyunits.J / pyunits.mol / pyunits.K),  # [4]
                "B": (19.88705, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-1),
                "C": (-8.598535 , pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-2),
                "D": (1.369784  , pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**-3),
                "E": (0.527601, pyunits.J * pyunits.mol**-1 * pyunits.K**-1 * pyunits.kiloK**2),
                "F": (-4.935202, pyunits.kJ / pyunits.mol),
                "G": (212.3900, pyunits.J / pyunits.mol /pyunits.K),
                "H": (0, pyunits.kJ / pyunits.mol)
            },
        },
    },

},

# Specifying phases
"phases": {
    "Vap": {"type": VaporPhase, "equation_of_state": Ideal},#Pv=nT

},
# Set base units of measurement
"base_units": {
    "time": pyunits.s,
    "length": pyunits.m,
    "mass": pyunits.kg,
    "amount": pyunits.mol,
    "temperature": pyunits.K,
},
# Specifying state definition
"state_definition": FTPx,
"state_bounds": {
    "flow_mol": (0, 100, 1000, pyunits.mol / pyunits.s),
    "temperature": (273.15, 300, 2500, pyunits.K),
    "pressure": (5e3, 1e5, 1e6, pyunits.Pa),
},
"pressure_ref": (1e5, pyunits.Pa),
"temperature_ref": (300, pyunits.K),


}


m = ConcreteModel()
m.fs = FlowsheetBlock(dynamic=False)

m.fs.methane = GenericParameterBlock(**configuration)
```

```
m.fs.react = Pump(
    property_package = m.fs.methane,
)


m.fs.react.inlet.mole_frac_comp[0,"O2"].fix(1e-20)
m.fs.react.inlet.mole_frac_comp[0,"N2"].fix(1e-20)
m.fs.react.inlet.mole_frac_comp[0,"CO2"].fix(1e-20)
m.fs.react.inlet.mole_frac_comp[0,"H2O"].fix(1e-20)
m.fs.react.inlet.mole_frac_comp[0,"CH4"].fix(1e-20)
m.fs.react.inlet.temperature.fix(1000)
m.fs.react.inlet.pressure.fix(1e5)
m.fs.react.inlet.flow_mol.fix(10)
m.fs.react.deltaP.fix(30000)
m.fs.react.efficiency_pump.fix(0.8)


print(degrees_of_freedom(m))
assert degrees_of_freedom(m) == 0
m.fs.react.initialize(outlvl=idaeslog.INFO)
solver=SolverFactory("ipopt")
status=solver.solve(m,tee=True)

temps = list(range(300,2001,100))
temps.insert(0,298.15)
Cps = np.zeros((19,5)) #(row,col)
Cpstr = ""
compounds = ["O2","N2","CO2","H2O","CH4"]

for i in range(5):
    for j in range(19):
        m.fs.react.inlet.mole_frac_comp[0,compounds[i]].fix(1)
        m.fs.react.inlet.mole_frac_comp[0,compounds[i-1]].fix(1e-20)
        m.fs.react.inlet.temperature.fix(temps[j])
        solver=SolverFactory("ipopt")
        status=solver.solve(m,tee=True)
        Cps[j][i] = value(m.fs.react.control_volume.properties_in[0].cp_mol)


for j in range(19):
    for i in range(5):
        Cpstr+=str("{:.3f}".format(Cps[j][i]))
        Cpstr+=","
    Cpstr+="\n"

print(Cps)
print(Cpstr)
```

Appendix B6. Custom Combustion Unit Model

```
#custom_combustion_reactor

# Import Pyomo libraries
from pyomo.environ import Reference, Var, Param, units as pyunits, value
from pyomo.common.config import ConfigBlock, ConfigValue, In, Bool

# Import IDAES cores
from idaes.core import (
    ControlVolume0DBlock,
    declare_process_block_class,
    MaterialBalanceType,
    EnergyBalanceType,
    MomentumBalanceType,
    UnitModelBlockData,
    useDefault,
)
from idaes.core.util.config import (
    is_physical_parameter_block,
    is_reaction_parameter_block,
)

from biomass_combustion_rp import BMCombReactionParameterBlock

@declare_process_block_class("MultiCombReactor")
class MultiCombReactorData(UnitModelBlockData):
    CONFIG = UnitModelBlockData.CONFIG()

    CONFIG.declare(
        "material_balance_type",
        ConfigValue(
            default=MaterialBalanceType.useDefault,
            domain=In(MaterialBalanceType),
            description="Material balance construction flag",
            doc="""Indicates what type of mass balance should be constructed,"""
        ),
    )
    CONFIG.declare(
        "energy_balance_type",
        ConfigValue(
            default=EnergyBalanceType.useDefault,
            domain=In(EnergyBalanceType),
            description="Energy balance construction flag",
            doc="""Indicates what type of energy balance should be constructed,""",
        ),
    )
```

```
CONFIG.declare(
    "momentum_balance_type",
    ConfigValue(
        default=MomentumBalanceType.pressureTotal,
        domain=In(MomentumBalanceType),
        description="Momentum balance construction flag",
        doc="""Indicates what type of momentum balance should be constructed""",
    ),
)
CONFIG.declare(
    "has_heat_of_reaction",
    ConfigValue(
        default=False,
        domain=Bool,
        description="Heat of reaction term construction flag",
        doc="""Indicates whether terms for heat of reaction terms should be constructed,""",
    ),
)
CONFIG.declare(
    "has_heat_transfer",
    ConfigValue(
        default=False,
        domain=Bool,
        description="Heat transfer term construction flag",
        doc="""Indicates whether terms for heat transfer should be constructed,""",
    ),
)
CONFIG.declare(
    "has_pressure_change",
    ConfigValue(
        default=False,
        domain=Bool,
        description="Pressure change term construction flag",
        doc="""Indicates whether terms for pressure change should be constructed,""",
    ),
)
CONFIG.declare(
    "property_package",
    ConfigValue(
        default=useDefault,
        domain=is_physical_parameter_block,
        description="Property package to use for control volume",
        doc="""Property parameter object used to define property calculations,""",
    ),
)
CONFIG.declare(
    "property_package_args",
    ConfigBlock(
        implicit=True,
        description="Arguments to use for constructing property packages",
```

```
            doc="""A ConfigBlock with arguments to be passed to a property block(s)""",
        ),
    )

    def build(self):
        # Call UnitModel.build to setup dynamics
        super(MultiCombReactorData, self).build()
                                                                self.reaction_package          =
BMCombReactionParameterBlock(property_package=self.config.property_package)

        self.control_volume = ControlVolume0DBlock(
            dynamic=self.config.dynamic,
            property_package=self.config.property_package,
            property_package_args=self.config.property_package_args,
            reaction_package=self.reaction_package,
        )

        self.control_volume.add_state_blocks(has_phase_equilibrium=False)

        self.control_volume.add_reaction_blocks(has_equilibrium=False)

        self.control_volume.add_material_balances(
            balance_type=self.config.material_balance_type, has_rate_reactions=True
        )

        self.control_volume.add_energy_balances(
            balance_type=self.config.energy_balance_type,
            has_heat_transfer=self.config.has_heat_transfer,
            has_heat_of_reaction=self.config.has_heat_of_reaction,
        )

        self.control_volume.add_momentum_balances(
            balance_type=self.config.momentum_balance_type,
            has_pressure_change=self.config.has_pressure_change,
        )

        # Set references to balance terms at unit level
        if (
            self.config.has_heat_transfer is True
            and self.config.energy_balance_type != EnergyBalanceType.none
        ):
            self.heat_duty = Reference(self.control_volume.heat[:])
        if (
            self.config.has_pressure_change is True
            and self.config.momentum_balance_type != MomentumBalanceType.none
        ):
            self.deltaP = Reference(self.control_volume.deltaP[:])

        # Add Ports
        self.add_inlet_port()
```

```
        self.add_outlet_port()

        #abbreviations
        mw = self.config.property_package.config.components
        mw_air = 28.97
        O2_compound = "O2"
        N2_compound = "N2"

        # Add Custom Variables

        #casing heat loss variables
        self.ohtc = Var(initialize=250, units=pyunits.J/pyunits.m**2/pyunits.K/pyunits.s)
        self.surface_area = Var(initialize=0.02, units=pyunits.m**2, doc="casing outer surface area")
        self.surface_temp = Var(initialize=55+273.12, units=pyunits.K, doc="outer skin temperature of
boiler")

        # biomass-specific NCV variables
        self.hcon=Var(initialize=0.06) #concentration of hydrogen as a percentage of weight, h=6%
        self.wcon=Var(initialize=0.09) #water content of fuel as percentage of weight
        self.gcv=Param(initialize=20.2, units=pyunits.MJ/pyunits.kg, doc="gross calorific value")

        # variables for each reaction in rate_reaction_idx list
        for r in self.reaction_package.rate_reaction_idx:
            p,l = self.reaction_package.limit_reactant_dict[r]
            setattr(self,f"conversion_{r}", Var(initialize=1,bounds=(0,1), units="dimensionless"))
            setattr(self,f"dh_rxn_{r}", Var(initialize=1000000, units=pyunits.J/pyunits.mol))
            setattr(self,f"excess_air_percent_{r}", Var(initialize = 15, units="dimensionless"))
            setattr(self,f"air_fuel_mass_ratio_{r}", Var(initialize=6, units="dimensionless"))
            setattr(self,f"mass_flow_kg_{l}", Var(initialize=1 ,doc="mass flow of fuel in kg/s"))
            setattr(self,f"mole_flow_{l}", Var(initialize=100, doc="fuel molar flow in mol/s"))
            setattr(self,f"mole_flow_air_{r}", Var(initialize=1000, doc="mole flow air in mol/s"))

        # ash mass content variable only for reactions with ash in uncombs_set list
        for u in self.reaction_package.uncombs_set:
            setattr(self,f"ash_mass_{u}",Var(initialize=0, units=pyunits.g/pyunits.g))

        # Add Custom Constraints

        @self.Constraint(self.reaction_package.rate_reaction_idx)
        def fuel_flow_conversion(b,r): #kg/s to mol/s
            p,l = b.reaction_package.limit_reactant_dict[r]
                                    return      getattr(b,      f"mass_flow_kg_{l}")      ==      getattr(b,
f"mole_flow_{l}")*mw[l]["parameter_data"]["mw"][0]/1000

        @self.Constraint(self.flowsheet().time,self.reaction_package.rate_reaction_idx)
        def mole_flow_link(b,t,r):
            p,l = b.reaction_package.limit_reactant_dict[r]
                                                return      getattr(b,      f"mole_flow_{l}")      ==
b.control_volume.properties_in[t].mole_frac_comp[l]*b.control_volume.properties_in[t].flow_mol
```

```
        @self.Constraint(self.reaction_package.rate_reaction_idx)
        def Air_Fuel_ratio(b,r):
            p,l = b.reaction_package.limit_reactant_dict[r]
                                    return      getattr(b,      f"air_fuel_mass_ratio_{r}")      ==      getattr(b,
f"mole_flow_air_{r}")*mw_air/(getattr(b, f"mass_flow_kg_{l}")*1000)

        @self.Constraint(self.reaction_package.rate_reaction_idx)
        def excess_air(b,r):
            p,l = b.reaction_package.limit_reactant_dict[r]
            O2_stoich = b.reaction_package.rate_reaction_stoichiometry[r,"Vap",O2_compound]
            fuel_stoich = b.reaction_package.rate_reaction_stoichiometry[r,p,l]
                            return     (1+getattr(b,     f"excess_air_percent_{r}")/100)*(-O2_stoich)/0.21/(-
fuel_stoich)*getattr(b, f"mole_flow_{l}") == getattr(b, f"mole_flow_air_{r}")

        @self.Constraint(self.flowsheet().time)
        def N2_flow_link(b,t):
                                    return      sum(getattr(b,      f"mole_flow_air_{r}")      for      r      in
b.reaction_package.rate_reaction_idx)*0.79                                                                  ==
b.control_volume.properties_in[t].mole_frac_comp[N2_compound]*b.control_volume.properties_i
n[t].flow_mol

        @self.Constraint(self.flowsheet().time)
        def O2_flow_link(b,t):
                                    return      sum(getattr(b,      f"mole_flow_air_{r}")      for      r      in
b.reaction_package.rate_reaction_idx)*0.21                                                                  ==
b.control_volume.properties_in[t].mole_frac_comp[O2_compound]*b.control_volume.properties_i
n[t].flow_mol

        @self.Constraint(self.flowsheet().time)
        def total_flow_link(b,t):
            l = b.reaction_package.limit_reactant_dict
            rxns = b.reaction_package.rate_reaction_idx
           return b.control_volume.properties_in[t].flow_mol == sum(getattr(b, f"mole_flow_{l[r][1]}")
for r in rxns)+sum(getattr(b, f"mole_flow_air_{r}") for r in rxns)


        @self.Constraint(self.reaction_package.uncombs_set)
        def ash_con(b,u):
            p,l = b.reaction_package.limit_reactant_dict[u]
            ash_perc = getattr(b,f"ash_mass_{u}")
            ashi = b.reaction_package.stoich_init[u,"Sol","ash"]
            fueli = b.reaction_package.stoich_init[u,p,l]
            mw_ash = mw["ash"]["parameter_data"]["mw"][0]
            mw_fuel = mw[l]["parameter_data"]["mw"][0]
            ash_perc_mol = ash_perc*mw_fuel/mw_ash
            added_mols_BM = (ash_perc_mol-ashi)*(-fueli)/(1-(ash_perc_mol-ashi))
            b.reaction_package.rate_reaction_stoichiometry[u,"Sol","ash"].unfix()
                                return      b.reaction_package.rate_reaction_stoichiometry[u,"Sol","ash"]==
added_mols_BM*(1+ashi)+ashi*(-fueli)
```

```
        @self.Constraint(self.reaction_package.uncombs_set)
        def ash_con_fuel(b,u):
            p,l = self.reaction_package.limit_reactant_dict[u]
            ash_perc = getattr(b,f"ash_mass_{u}")
             ashi = b.reaction_package.stoich_init[u,"Sol","ash"] #initial ash is assumed to be part of the
mass balance
            mw_ash = mw["ash"]["parameter_data"]["mw"][0]
            fueli = b.reaction_package.stoich_init[u,p,l]
            mw_fuel = mw[l]["parameter_data"]["mw"][0]
            ash_perc_mol = ash_perc*mw_fuel/mw_ash
            added_mols_BM = (ash_perc_mol-ashi)*(-fueli)/(1-(ash_perc_mol-ashi))
            b.reaction_package.rate_reaction_stoichiometry[u,p,l].unfix()
            return b.reaction_package.rate_reaction_stoichiometry[u,p,l] == -(added_mols_BM)+fueli

        @self.Constraint(self.reaction_package.rate_reaction_idx)
        def dh_rxn_link(b,r):
            b.reaction_package.dh_rxn[r].unfix()
            return getattr(b,f"dh_rxn_{r}") == b.reaction_package.dh_rxn[r]

        #dedicated NCV constaint for Biomass
        @self.Constraint()
        def ncv_eqn(b):
            ash_perc = getattr(b,f"ash_mass_R1")
            mw_ash = mw["ash"]["parameter_data"]["mw"][0]
            ashi = b.reaction_package.stoich_init["R1","Sol","ash"]
            fueli = b.reaction_package.stoich_init["R1","Sol","biomass"]
            mw_fuel = mw["biomass"]["parameter_data"]["mw"][0]
            ash_perc_mol = ash_perc*mw_fuel/mw_ash
            added_mols_BM = (ash_perc_mol-ashi)*(-fueli)/(1-(ash_perc_mol-ashi))
            return b.dh_rxn_R1 == (
                    -(b.gcv*(1-b.wcon)-2.447*b.wcon-2.447*b.hcon*9.01*(1-b.wcon))*162.1394*1000*(-
fueli)/(added_mols_BM-fueli))

        @self.Constraint(self.flowsheet().time,)
        def casing_heat_loss(b,t):
            return b.heat_duty[t] == (
            b.ohtc*b.surface_area*(-b.outlet.temperature[0]+b.surface_temp)
            )

        @self.Constraint(self.flowsheet().time,self.reaction_package.rate_reaction_idx)
        def conversion_performance_eqn(b, t, r):
            p,l = self.reaction_package.limit_reactant_dict[r]
            return getattr(b, f"conversion_{r}") == (
            b.control_volume.rate_reaction_extent[t,r]
             /(b.control_volume.properties_in[t].mole_frac_comp[l]
             *b.control_volume.properties_in[t].flow_mol
            ))

    #variables displayed in terminal unit model report
    def _get_performance_contents(self, time_point=0):
```

```
var_dict = {
    "Biomass Water Content": self.wcon,
    "Biomass H2 Content": self.hcon,
    }
for r in self.reaction_package.rate_reaction_idx:
    var_dict["Conversion %s"%(r)] = getattr(self,f"conversion_{r}")
    var_dict["dh_rxn %s"%(r)] = getattr(self, f"dh_rxn_{r}")
for u in self.reaction_package.uncombs_set:
    var_dict["Ash content %s"%(u)] = getattr(self,f"ash_mass_{u}")
if hasattr(self, "heat_duty"):
    var_dict["Heat Duty"] = self.heat_duty[time_point]
if hasattr(self, "deltaP"):
    var_dict["Pressure Change"] = self.deltaP[time_point]
return {"vars": var_dict}
```

Appendix B7. Multi Steady State Superheater Boiler System Model

```
"""
Combustion Boiler Model with steam superheater.
Modelled by adiabatic combustion reactor sending hot flue to boiler HX and superheater HX in
counter-current to boiler water stream.

                                        BlowDownWater
                                            ^
                                            |
                                    +-------[Phase Separation]
                                    |       |
                                    |     BoilerWater
                                    |       ^
                                    |       |
Fuel+Air --->[Reactor]<|---> Hot Flue ---> [Superheater] ---> [Boiler] ---> Stack Flue
             |              |       ^
             |              \/      |
             \/           Superheated  BoilerFeedWater
             Ash            Steam

"""

__author__ = "David Dickson"

#Importing required pyomo and idaes components

import numpy as np
from pyomo.environ import (
   ConcreteModel,
   Expression,
   SolverFactory,
   TransformationFactory,
   value,
   units as pyunits
)
from pyomo.network import Arc, SequentialDecomposition

#Todo add the four other unit operations
from idaes.models.unit_models import (
HeatExchanger,
Separator,
)
from idaes.models_extra.power_generation.unit_models.helm.phase_separator import
HelmPhaseSeparator

from idaes.core.util.model_statistics import degrees_of_freedom
from idaes.core import FlowsheetBlock
# Import idaes logger to set output levels
import idaes.logger as idaeslog
```

```
from  idaes.models.properties.modular_properties import GenericParameterBlock
from  biomass_comb_pp import configuration

#helmholtz import for water
from idaes.models.properties.general_helmholtz import (
        HelmholtzParameterBlock,
        AmountBasis,
        PhaseType,
    )
from idaes.models.unit_models.separator import SplittingType
# import custom_combustion_reactor

import matplotlib.pyplot as plt

from custom_combustion_reactor import MultiCombReactor

m = ConcreteModel()

m.fs = FlowsheetBlock(dynamic=False)

m.fs.biomass_properties = GenericParameterBlock(**configuration)

m.fs.steam_properties = HelmholtzParameterBlock(
        pure_component="h2o", amount_basis=AmountBasis.MOLE,
        phase_presentation=PhaseType.LG,
        # state_vars=StateVars.TPX
    )

m.fs.fire_side = MultiCombReactor(
    property_package = m.fs.biomass_properties,
    # reaction_package = m.fs.reaction_params,
    has_heat_of_reaction=True,
    has_heat_transfer=True,
    has_pressure_change=False,
)

m.fs.superheater = HeatExchanger(
    delta_temperature_callback=delta_temperature_amtd_callback,
    hot_side_name="shell",
    cold_side_name="tube",
    shell={"property_package": m.fs.biomass_properties},
    tube={"property_package": m.fs.steam_properties}
)

m.fs.boiler_hx = HeatExchanger(
    delta_temperature_callback=delta_temperature_amtd_callback,
    hot_side_name="shell",
    cold_side_name="tube",
    shell={"property_package": m.fs.biomass_properties},
    tube={"property_package": m.fs.steam_properties}
```

```
)

m.fs.ash_sep = Separator(
    property_package = m.fs.biomass_properties,
    split_basis = SplittingType.phaseFlow,
    outlet_list = ["flue", "ash"]
)

m.fs.bdw_sep = HelmPhaseSeparator(
    property_package = m.fs.steam_properties,
)

m.fs.s01 = Arc(source=m.fs.fire_side.outlet,destination=m.fs.ash_sep.inlet)
m.fs.s02 = Arc(source=m.fs.ash_sep.flue,destination=m.fs.superheater.shell_inlet)
m.fs.s03 = Arc(source=m.fs.superheater.shell_outlet,destination=m.fs.boiler_hx.shell_inlet)
m.fs.s04 = Arc(source=m.fs.boiler_hx.tube_outlet,destination=m.fs.bdw_sep.inlet)
m.fs.s05 = Arc(source=m.fs.bdw_sep.vap_outlet,destination=m.fs.superheater.tube_inlet)
TransformationFactory("network.expand_arcs").apply_to(m)


m.fs.fire_side.conversion_R1.fix(1)
m.fs.fire_side.hcon.fix(0.07)
m.fs.fire_side.wcon.fix(0.1)
m.fs.fire_side.ohtc.fix(100)
m.fs.fire_side.surface_area.fix(0.1)
m.fs.fire_side.surface_temp.fix(60)
m.fs.fire_side.ash_mass_R1.fix(0.0)

m.fs.fire_side.conversion_Rcoal.fix(1)
m.fs.fire_side.ash_mass_Rcoal.fix(0.0)
m.fs.fire_side.dh_rxn_Rcoal.fix(-284675.1254)


#reactor feed stream
m.fs.fire_side.inlet.mole_frac_comp[0,"N2"].fix(0.70)
m.fs.fire_side.inlet.mole_frac_comp[0,"O2"].fix(0.2)
m.fs.fire_side.inlet.mole_frac_comp[0,"CO2"].fix(1e-20)
m.fs.fire_side.inlet.mole_frac_comp[0,"H2O"].fix(1e-20)
m.fs.fire_side.inlet.mole_frac_comp[0,"biomass"].fix(1e-20)
m.fs.fire_side.inlet.mole_frac_comp[0,"coal"].fix(0.1)
m.fs.fire_side.inlet.mole_frac_comp[0,"ash"].fix(1e-20)
m.fs.fire_side.inlet.temperature.fix(300)
m.fs.fire_side.inlet.pressure.fix(101325)
m.fs.fire_side.inlet.flow_mol.fix(40)

#specifying ash separation
m.fs.ash_sep.split_fraction[0,"ash","Sol"].fix(1)
m.fs.ash_sep.split_fraction[0,"ash","Vap"].fix(0)

steam_press=101325
```

```
m.fs.boiler_hx.tube_inlet.enth_mol.fix(m.fs.steam_properties.htpx(p=steam_press*pyunits.Pa,T=300*pyunits.K))
m.fs.boiler_hx.tube_outlet.enth_mol.fix(m.fs.steam_properties.htpx(p=steam_press*pyunits.Pa,x=0.95))
m.fs.superheater.tube_outlet.enth_mol.fix(m.fs.steam_properties.htpx(p=steam_press*pyunits.Pa,T=400*pyunits.K))
m.fs.superheater.overall_heat_transfer_coefficient[0].fix(100)
m.fs.boiler_hx.tube_inlet.flow_mol.fix(20)
m.fs.boiler_hx.tube_inlet.pressure.fix(steam_press)
m.fs.boiler_hx.overall_heat_transfer_coefficient[0].fix(100)


#initialization routine
seq = SequentialDecomposition()
seq.options.select_tear_method = "heuristic"
seq.options.tear_method = "Wegstein"
seq.options.iterLim = 3

G = seq.create_graph(m)
heuristic_tear_set = seq.tear_set_arcs(G, method="heuristic")
order = seq.calculation_order(G)

#for identifying tear stream:
""" for o in heuristic_tear_set: #fs.s03
    print(o.name) """

tear_guesses = {
    "mole_frac_comp": {
        (0, "N2"): 0.67,
        (0, "O2"): 0.22,
        (0, "CO2"): 0.06,
        (0, "H2O"): 0.05,
        (0, "biomass"): 1e-20,
        (0, "coal"): 1e-20,
        (0, "ash"): 1e-20,
    },
    "flow_mol": {0: 40},
    "temperature": {0: 1000},
    "pressure": {0: 101325},
}

seq.set_guesses_for(m.fs.boiler_hx.shell_inlet, tear_guesses)

def function(unit):
    unit.initialize(outlvl=idaeslog.INFO)
    # print(degrees_of_freedom(unit))

print(degrees_of_freedom(m))
assert degrees_of_freedom(m) == 0
```

```
seq.run(m, function)

# m.fs.boiler_hx.overall_heat_transfer_coefficient[0].unfix()
# m.fs.boiler_hx.shell_outlet.temperature.fix(400)
m.fs.boiler_hx.shell_outlet.temperature.fix(400)
m.fs.fire_side.ash_mass_R1.fix(0.01)
m.fs.fire_side.inlet.flow_mol.unfix()
m.fs.fire_side.ash_mass_Rcoal.fix(0.03)

print(degrees_of_freedom(m))
assert degrees_of_freedom(m) == 0
solver=SolverFactory("ipopt")
status=solver.solve(m,tee=True)

m.fs.boiler_eff = Expression( #must change to for-loop that somes all rxn extents/dh_rxn's
   expr = 100*(m.fs.superheater.heat_duty[0]+m.fs.boiler_hx.heat_duty[0])/
                                        (sum(m.fs.fire_side.control_volume.rate_reaction_extent[0,r]*-
m.fs.fire_side.reaction_package.dh_rxn[r]                    for                    r                    in
m.fs.fire_side.reaction_package.rate_reaction_idx))
)

m.fs.duty_to_steam = Expression(
      expr   =   (m.fs.superheater.heat_duty[0]+m.fs.boiler_hx.heat_duty[0])/1000/1000/1000*60*60
#GJ/h
)

m.fs.fire_side.report()
print(value(m.fs.boiler_eff))
print(value(m.fs.duty_to_steam))


steady_states = list(range(2,30,2))
flue_temps = [340,370,400, 430]

efficiencies = np.zeros((len(flue_temps),len(steady_states)))
steam_duties = np.zeros((len(flue_temps),len(steady_states)))

""" Create MSS For-Loop Starting Here: """
for p,n in enumerate(flue_temps):
   m.fs.boiler_hx.shell_outlet.temperature.fix(n)
   for j,i in enumerate(steady_states):

      m.fs.boiler_hx.tube_inlet.flow_mol.fix(i)
      #pre-solve [actual] re-specification
      # m.fs.boiler_hx.shell_outlet.temperature.fix(450)
      # m.fs.boiler_hx.area.fix(20)

      # solver=SolverFactory("ipopt")
      print(degrees_of_freedom(m))
      assert degrees_of_freedom(m) == 0
```

```
        status=solver.solve(m,tee=True)

        steam_duties[p][j] = value(m.fs.duty_to_steam)
        efficiencies[p][j] = value(m.fs.boiler_eff)
        print(f"{p}-{j}")

print(efficiencies)
print(steam_duties)


for i,j in enumerate(flue_temps):
    plt.plot(steam_duties[i],efficiencies[i],label=f"{(j-273.15):.2f} C ")
plt.ylim(70,100)
plt.xlabel('Steam Generation Duty [GJ/hr]')
plt.ylabel('Boiler System Efficiency [%]')
plt.legend()
plt.title(f"Coal Fuel")
plt.show()


#results
m.fs.fire_side.report()
m.fs.superheater.report()
m.fs.boiler_hx.report()

print(f"    Boiler Efficiency: {value(m.fs.boiler_eff):.2f}%")
```

Appendix B8. Black Liquor Case Study Flowsheet Model

```
#black liquor case study

import matplotlib.pyplot as plt

#stoich_reactor_test
from pyomo.environ import (
    Constraint,
    Var,
    ConcreteModel,
    Expression,
    SolverFactory,
    TransformationFactory,
    value,
    units as pyunits
)

#Todo add the four other unit operations
from idaes.models.unit_models import (
Mixer,
# StoichiometricReactor,
Heater,
Heater
)

from idaes.core.util.model_statistics import degrees_of_freedom
from idaes.core import FlowsheetBlock
import idaes.logger as idaeslog
from idaes.models.properties.modular_properties import GenericParameterBlock
from black_liquor_pp import configuration

#helmholtz import for water
from idaes.models.properties.general_helmholtz import (
        HelmholtzParameterBlock,
        AmountBasis,
        PhaseType,
    )

from bl_combustion_reactor import MultiCombReactor #has rxn pkg included

from pyomo.environ import Reference, Var, Param, units as pyunits, value

m = ConcreteModel()

m.fs = FlowsheetBlock(dynamic=False)

m.fs.bl_properties = GenericParameterBlock(**configuration)

m.fs.steam_properties = HelmholtzParameterBlock(
```

```
        pure_component="h2o", amount_basis=AmountBasis.MOLE,
        phase_presentation=PhaseType.LG,
    )

m.fs.R101 = MultiCombReactor(
    property_package = m.fs.bl_properties,
    # reaction_package = m.fs.reaction_params,
    has_heat_of_reaction=True,
    has_heat_transfer=True,
    has_pressure_change=False
)

m.fs.H101 = Heater(
    property_package = m.fs.steam_properties,
)

m.fs.mix = Mixer(
    property_package = m.fs.bl_properties,
    inlet_list = ["air","fuel"]
)

m.fs.s01 = Arc(source=m.fs.mix.outlet, destination=m.fs.R101.inlet)
TransformationFactory("network.expand_arcs").apply_to(m)


#case study inputs
#flows
f = {
    "air_flow": 976,
    "bl_flow": 998,
    "gas_flow": 0.1717
}


fuel_total = f["bl_flow"]+f["gas_flow"]


# air stream
m.fs.mix.air.mole_frac_comp[0,"N2"].fix(0.79)
m.fs.mix.air.mole_frac_comp[0,"O2"].fix(0.21)
m.fs.mix.air.mole_frac_comp[0,"CO2"].fix(1e-20)
m.fs.mix.air.mole_frac_comp[0,"H2O"].fix(1e-20)
m.fs.mix.air.mole_frac_comp[0,"BL"].fix(1e-20)
m.fs.mix.air.mole_frac_comp[0,"uncombustible"].fix(1e-20)
m.fs.mix.air.mole_frac_comp[0,"CH4"].fix(1e-20)
m.fs.mix.air.temperature.fix(33+273.15)
m.fs.mix.air.pressure.fix(101325)
m.fs.mix.air.flow_mol.fix(f["air_flow"])

# black liquor stream stream
m.fs.mix.fuel.mole_frac_comp[0,"N2"].fix(1e-20)
```

```
m.fs.mix.fuel.mole_frac_comp[0,"O2"].fix(1e-20)
m.fs.mix.fuel.mole_frac_comp[0,"CO2"].fix(1e-20)
m.fs.mix.fuel.mole_frac_comp[0,"H2O"].fix(1e-20)
m.fs.mix.fuel.mole_frac_comp[0,"BL"].fix(f["bl_flow"]/fuel_total)
m.fs.mix.fuel.mole_frac_comp[0,"uncombustible"].fix(1e-20)
m.fs.mix.fuel.mole_frac_comp[0,"CH4"].fix(f["gas_flow"]/fuel_total)
m.fs.mix.fuel.temperature.fix(123.6+273.15)
m.fs.mix.fuel.pressure.fix(101325)
m.fs.mix.fuel.flow_mol.fix(fuel_total)

m.fs.R101.conversion_Rbl.fix(1)
m.fs.R101.conversion_RCH4.fix(0)

m.fs.R101.dh_rxn_RCH4.fix(-802125)
m.fs.R101.dh_rxn_Rbl.fix(-135150)

m.fs.R101.outlet.temperature.fix(195.66+273.15)


m.fs.mix.initialize(outlvl=idaeslog.INFO)
m.fs.R101.initialize(outlvl=idaeslog.INFO)

m.fs.H101.inlet.flow_mol.fix(2177.69)
m.fs.H101.inlet.enth_mol.fix(m.fs.steam_properties.htpx(p=45*100*1000*pyunits.Pa,T=
(136.39+273.15)*pyunits.K))
m.fs.H101.inlet.pressure.fix(45*100*1000)

m.fs.H101.outlet.enth_mol.fix(m.fs.steam_properties.htpx(p=45*100*1000*pyunits.Pa,T
=(400+273.15)*pyunits.K))


m.fs.R101.heat_loss = Var(initialize=100000, units=pyunits.J/pyunits.s)


def heat_transfer_rule(b,t):
    return b.H101.heat_duty[0]+b.R101.heat_loss == -b.R101.heat_duty[0]

m.fs.heat_transfer = Constraint(
    m.fs.time,
    rule=heat_transfer_rule
)


print(degrees_of_freedom(m.fs.H101))
m.fs.H101.initialize(outlvl=idaeslog.INFO)


m.fs.boiler_efficiency = Expression(
                                                        expr                =
100*value(m.fs.H101.heat_duty[0])/sum(m.fs.R101.control_volume.rate_reaction_extent
```

```
[0,r]*(-m.fs.R101.reaction_package.dh_rxn[r])          for          r          in
m.fs.R101.reaction_package.rate_reaction_idx)
    )

solver=SolverFactory("ipopt")
status=solver.solve(m,tee=True)
case_fluetemp = value(m.fs.R101.outlet.temperature[0])
case_eff = value(m.fs.boiler_efficiency)

m.fs.R101.report()
print(case_eff)
print(case_fluetemp)
print(value(m.fs.R101.heat_loss))

m.fs.R101.heat_loss.fix(30604244.11776122) #solved for case study condition

m.fs.H101.inlet.flow_mol.unfix()

print(degrees_of_freedom(m))
assert degrees_of_freedom(m) == 0
temps = range(600-273,150,-10)
effs = [0]*len(temps)


for i,flue_temp in enumerate(temps):
    m.fs.R101.outlet.temperature[0].fix(flue_temp+273.15)

    solver=SolverFactory("ipopt")
    status=solver.solve(m,tee=True)

    m.fs.boiler_efficiency = Expression(
                                                          expr               =
100*value(m.fs.H101.heat_duty[0])/sum(m.fs.R101.control_volume.rate_reaction_extent
[0,r]*(-m.fs.R101.reaction_package.dh_rxn[r])          for          r          in
m.fs.R101.reaction_package.rate_reaction_idx)
    )
    effs[i] = value(m.fs.boiler_efficiency)


m.fs.R101.report()
m.fs.H101.report()
# m.fs.mix.report()
print(f"{value(m.fs.boiler_efficiency):.2f}%")
print(f"{(value(m.fs.R101.outlet.temperature[0])-273.15):.2f} C")
# m.fs.R101.display()

print(case_eff)
print(case_fluetemp)

plt.plot(temps,effs)
```

```
plt.ylim(0,100)
plt.xlabel("Flue Stack Temperature [C]")
plt.ylabel("Boiler Efficiency [%]")
plt.title("Boiler Efficiency Curve Against Stack Temperature")
plt.plot([case_fluetemp-273.15],[case_eff],'ro',label='Case Study Conditions')
plt.legend(loc='center left')
plt.show()
```

Appendix C1. Original Literature Review

ENGEN582-25X

Honours Research and Development Project

Literature Review

Digitalization of Thermal Combustion Reaction Modelling for Steam Boiler Optimization

David Dickson

Ben Lincoln, James Carson

The University of Waikato

Table of Contents

1.0 Introduction

The aim of this literature review is to review the literature evaluation and development of digital tools for the modelling and optimisation of thermal combustion reactions in steam raising boiler systems. In section 2, the practical and geo-economical New Zealand context in which steam boilers are being used must first be scoped. Then in section 3, a significant evaluation of digitalization and digital tool technology in the space of process systems engineering proceeds with an aim to determine which current digital tool could be leveraged to be the basis of an advanced digital tool. Then in section 4, a review of the more detailed modelling of steam boilers regarding the chemical reactions and heat transfer is done. The findings show the available work that has been done that can support the specific modelling of fuel fired combustion steam boilers.

2.0 Process Heat Decarbonisation

2.1 Process Heat Decarbonisation Challenge

Energy intensive industries contribute significant CO_2 equivalent emissions to New Zealand's total emissions. 35% of New Zealand's energy is used for process heat with 60% of the process heat supplied by burning fossil fuels (*Accelerating the Decarbonisation of Process Heat*, 2021).

The Paris Agreement and New Zealand's Emissions Reduction Plan to target net zero carbon emissions (except biogenic methane) by 2050 has put pressure on energy intensive industries to optimise for decarbonised processes (Ministry of Business & Employment, 2022). The emissions reduction plan includes…

With these goals in place, there is a demand for process engineers to coordinate with the industrial sector to create effective energy solutions. This will require the development of tools and techniques which can enable the serviceable deployment of such solutions.

Efforts of decarbonisation contribute to the 7th United Nation Sustainable Development Goal that is 'Ensure access to affordable, reliable, sustainable and modern energy for all ("The 17 Sustainable Development Goals," 2015).

2.2 Process Systems Engineering and Process Optimization

As expounded by Pistikopoulos et al. (2021), "Process Systems Engineering (PSE) is the scientific discipline of integrating scales and components describing the behaviour of a physicochemical system, via mathematical modelling, data analytics, design, optimization and control." Useful techniques for process energy optimisation exist will be used by process system engineers to contribute to process decarbonisation. Some of these techniques include pinch analysis, robust optimisation, and stochastic optimisation (Deng et al., 2023; Kasivisvanathan et al., 2014). Such techniques will help process decarbonisation by optimizing for energy efficiency.

Digital tools for designing, modelling and simulating new or existing processes are essential to the practice of process engineering. For this reason, there are many tools that are standard to process engineering modelling such as Aspen Hysys and Ansys Fluent which are well-known tools for process simulation and CFD respectively (Ansys, 2024; Aspen Technology, 2025).

It is standard practice for process systems engineers to use modelling and simulation tools for process optimisation and heat decarbonisation. Applicable optimisation techniques such as those mentioned above require the modelling of the process and its relevant process variables to some suitable degree of form and accuracy. Yip, et al. (2004) demonstrates the effect of model fidelity on optimisation. Models can also allow investigation of alternative fuel sources which is a key approach required by NZ's emissions reduction plan.

A system of particular interest to NZ process heat decarbonisation are steam raising boiler systems with 68% of NZ process heat generated by boiler systems (Ministry of Business & Employment, 2017). Process engineering optimisation techniques applied with digital tools with a focus on steam boiler systems will be a necessary approach for significantly advancing process heat decarbonisation for NZ industries.

Digital Tools for Digitalization for Boiler System Optimisation

3.1 Digitalization for Process Engineering

Digitalization is the use of digital tools to create value for existing systems. This is different to digitization which encapsulates the conversion of analogue data into digital data, perhaps by sensors and measurements (Bountouri, 2017).

Digitization can still however be encapsulated by digitalization. Digital tools are the software applications that are used by the end user to add said value. Digital tools, and subsequently digitalization can bring value by leveraging the inherent advantages of the software paradigm. Software applications have the potential to be highly customizable for nearly any system, especially those with high data, computational requirements, and complexity. In this section digitalization will be considered in context with the application of boiler system optimisation.

Digitalization has become increasingly prevalent in process engineering due to its benefits and the advancement of digital technologies that enable innovation in this space. The development of digital tools is desirable for the range of advantages they offer. Digital tools increase user productivity by automating steps in a workflow. Digital tools can leverage computing power for computationally heavy tasks. They allow the opportunity for integration with the Internet of Things for more dynamic connectivity of information. Digitalization has already seen great adoption for the enhancement of the process systems engineering discipline. All cutting-edge tools and techniques including process simulators mentioned in section 1 are applied as digital tools.

Digitalization can be applied to many areas of industry, Pietrasik et al. (2024) elaborates on digital modelling as well as feedstock and energy management, describing how digitalization advancements have helped develop the respective areas over time till present and near future. Digital modelling and energy management are directly related to process modelling and optimisation. There is clearly opportunity for digitalization to be applied to process heat decarbonisation.

However, there is still technological process to be made as far as digital process simulation tools are concerned. As described by de Beer and Depew (2021), "Process simulators are typically poorly integrated into engineering workflows beyond the process world, and if so, with a single directional information flow." This highlights the insufficient connectivity to high level workflows, interconnected systems for control and analysis, and the general adaptability for free flow of information as key limitations in current digital process simulation tools.

These limitations can be addressed by the characteristics of Industry 4.0. As explained by Lasi et al. (2014), one of the key technological pushes for Industry 4.0 is digitalization that can combine smarter technologies in an integrated system. Digitalization is an important step for firmly establishing process systems engineering in the 4th industrial revolution.

3.2 Digital Twins and Industry 4.0 Relevance

A significant example of digitalization and Industry 4.0 seeing quick and recent development in the process systems engineering space are digital twins. Walmsley et al. (2024) describes adaptive digital twins and explains how they can greatly contribute to energy intensive industries by their interconnectivity between the digital and physical systems as well as enhancement by advanced computing technologies. Tancredi et al. (2024) showcases two case studies of how digital twins can be used in control applications for food processing systems.

Both articles highlight the significance of process model simulation to form the basis of the digital system as a point of comparison to the physical system.

Walmsley et al. (2024) only introduces the ADT concept with seven potential applications in energy-intensive industries but not practical implementation. Digitalized solutions remain to be developed for these seven applications in this article. Tancredi et al. (2024) has physical systems in each case study with a structured software approach and digital twins defined by mathematical model. Since neither of the digital twin mathematical models are complex systems with a large array of unit operations, the leverage of process model simulation programs for more complex systems is left unexplored. Such leverage of process model simulation could be implemented for adaptive digital twins in energy-intensive industries.

For a process modelling and simulation tool to be considered for leverage, it should be sufficiently flexible and adaptable to being enhanced by digitalization and connectivity with advanced technologies like cyber physical systems and the Internet of Things that are characteristic of Industry 4.0.

3.3 Tool Evaluation: Aspen Hysys vs. IDAES

In this section two digital tools capable of process modelling and simulation are compared for their suitability to digitalization by enhanced connectivity with advanced technology. The two process modelling and simulation tools considered are Aspen Hysys and IDAES-CMF (Institute for the Design of Advanced Energy Systems-Core Modelling Framework).

Aspen Hysys is made by Aspen Tech and is typically used in commercial applications for petrochemical related industrial processes (Aspen Technology, 2025). Aspen Hysys is a legacy software program which was initially released in 1996 by researchers at the University of Calgary (Gani et al., 2001). It has since been acquired by Aspen Tech and is continually developed to remain up to date with relevant focuses of process engineering. Recently, for example, the electrolyser block was recently added to Aspen Hysys and Aspen Plus to support the development of sustainable technologies (Aspen Technology, 2024). Aspen Tech sells Aspen Hysys as a commercial product and its software is not open source. This makes Aspen Hysys proprietary software that limits the available freedom to users for digitalization and extension of its base process simulation capabilities. If the capabilities of Aspen Hysys were to be leveraged for a digital tool, the inherent commercial cost for Aspen Hysys would ultimately limit the economic accessibility of the digital tool.

Additionally, because Hysys is a relatively older software program it was initially developed without the foresight of present-day technology paradigms. It therefore was not developed with the openness that is more desirable for modern digitalization efforts. Aspen Tech

Aspen hysys has options for bidirectional information flow (Aspen Technology, 2004a). Aspen Hysys has the capacity for automation which allows externally coded programs, coded for example with Microsoft Visual Basic for Applications; to automate tasks a user may execute in the Aspen Hysys environment. There is also capacity for information flow in the opposite direction

In this research by Zarafshani et al. (2024), Aspen Hysys is utilized by extracting model data and evaluating a Life Cycle Analysis using python code as illustrated in Figure 1. This implementation of data extraction does not dynamically communicate with the Aspen process flow model in a bidirectional flow of information.

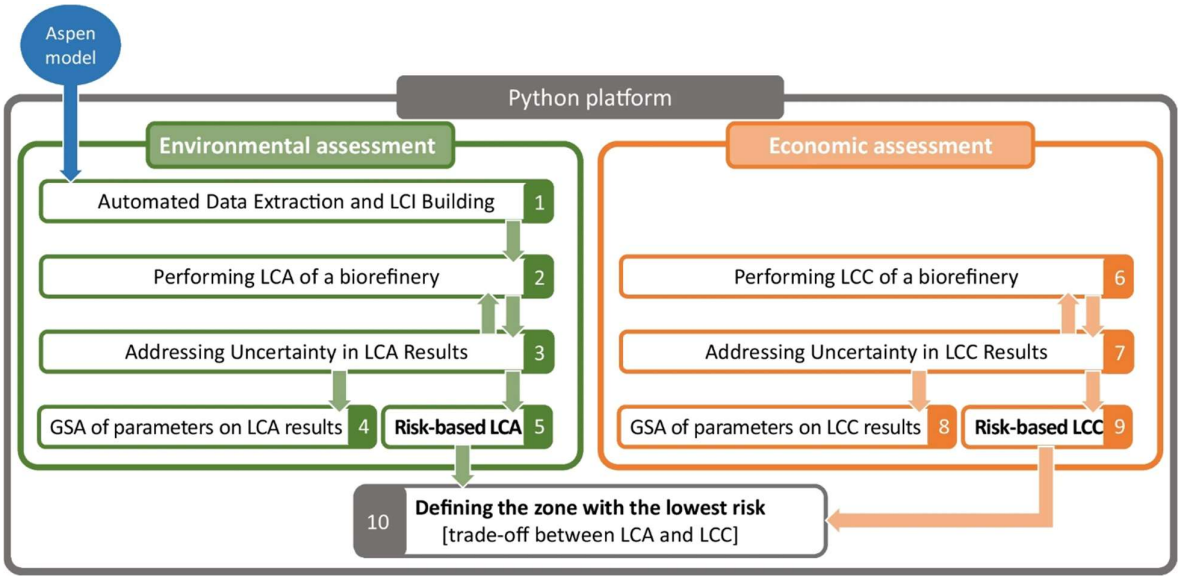

Figure 1. Principle of the SAB tool. Integration and information flow between blocks (Zarafshani et al., 2024).

Even with a bidirectional system the underlying mathematical model cannot be accessed. Approaches to interconnectivity with external programming cannot customize or export the mathematical model. Because of this, no attempt can be made to make optimized software designed for an integrated digitalized solution that can effectively leverage both the capabilities of Aspen Hysys' process model simulation and software programming.

Aspen Hysys uses a sequence modular approach to solving and simulating the mass and energy balance for a process flowsheet model. This means that each unit operation is solved sequentially with the output stream result of one becoming the input stream conditions of the next. The computational demand for this solving approach can become very high when there are many unit operations and loops.

Overall, Aspen Hysys possesses limitations which make the program less desirable to be transformed into an integrated digital tool.

IDAES-CMF is a framework for process flowsheet modelling and simulation that has been built off the Python programming language (Lee et al., 2021). It leverages the Pyomo algebraic modelling environment (Bynum, 2021). As such, IDAES can do equation-based solving. Equation-based can be more computationally efficient when process flow models are more complex and elaborate with many unit-processes.

IDAES and its libraries are open source and free to download and use by the public. This makes IDAES highly accessible both economically and technologically. Researchers from any level of economic standing can use the framework as a process modelling and simulation tool. IDAES has its own accessible libraries for unit models and property models. Since it is open source, these libraries can be used, modified, and shared to the community of IDAES users, researchers, and developers. The underlying framework is also accessible by the public, and outside developers can help by contributing to the ongoing development of IDAES. Since IDAES is built from Python code, all Python accessible libraries can be utilized to the extent the base IDAES capabilities.

IDAES is well documented with complete documentation on the IDAES website idaes-pse.readthedocs.io. The external development community also follow suit, for example, WaterTap has external libraries made to be compatible with the IDAES platform and has a website with complete documentation of the external libraries.

IDAES, compared to Aspen Hysys, is a much younger software platform having been initially released in early 2023. As such it was built with the awareness of more advanced technological digitalization paradigms. Open-source software for example is much more popular now than it was 20 years ago. *OPEN SOURCE SECURITY AND RISK ANALYSIS REPORT* (2023) scanned over 1000 codebases in their audit and found the majority 76% of codebases to be open-source code. This report discusses the security concerns of open-source codebases which are their own set of challenges outside the scope of this literature review. Overall, a software tool contributing to sustainability development is better suited to having better access, proliferation, community, and technological advancement via the open-source paradigm.

From a software perspective, the integration potential that the IDAES framework has with Python makes it highly suitable for dynamic bidirectional flow of information to and from the process flow model which allows great connectivity advantage. This is illustrated in Figure 2.

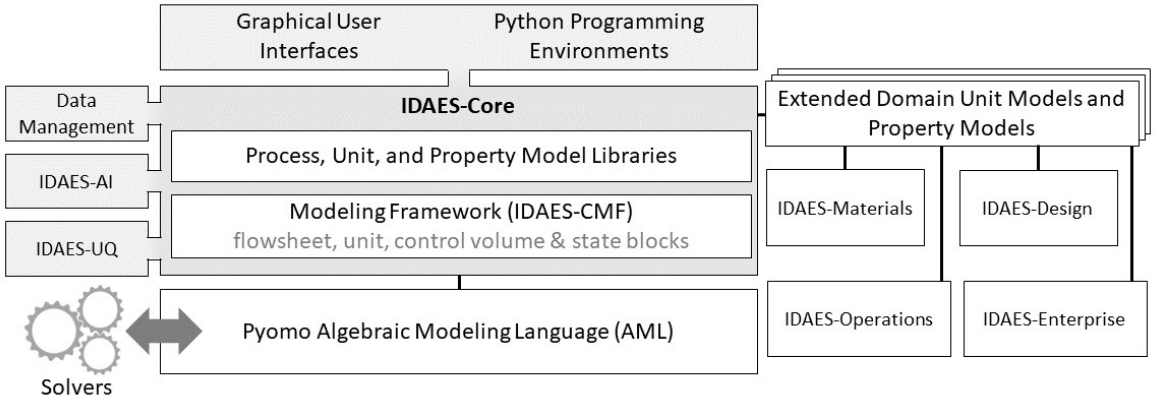

Figure 2 IDEAS-IP (integrated platform) and interconnected components (*Concepts — IDAES*, 2024).

IDAES is more promising as the basis for an accessible and adaptable digitalization solution for decarbonization of New Zealand Processes with a focus on the steam boiler system. This is evaluated by comparison with the commercial and well-known Aspen Hysys as summarized in Table 1.

Table 1. Digital tool comparison of Aspen Hysys and IDAES process model simulators.

Aspen Hysys	**IDAES**
Closed source	Open source
Sequential modular solver	Equation based algebraic solver
Initially released in 1996	Initially released in 2023
inaccessible mathematical model solver	accessible underlying pyomo algebraic mathematical model solver
Expensive to purchase	Free to download and use
Poor documentation	Good documentation

3.4 Suitability for Chemical Reaction and Boiler Combustion Modelling

Both process model simulators being compared can simulate the combustion and steam raising process of a steam boiler operation.

Aspen Hysys has the built-in ‘fired heater’ unit operation which can model with sufficient detail a steam raising boiler. The unit operation can readily function as a boiler in the Hysys flowsheet in a

wider process model. In the Hysys fired heater operation only the type of fuel, fuel ratio, and other related parameters are defined, leaving the specifics of the chemical reaction ignored in this model. Alternatively, Hysys can model chemical reactions by various reactor operations. This is useful for modelling a specific combustion reaction with greater kinetic detail that is uncaptured in the fired heater model. A combustion reactor, in combination with heat exchanger units could simulate a steam boiler.

IDAES also has unit models suitable for steam boiler modelling with the 'Boiler Fire Side Model' and 'Steam Heater Model'. Similarly to Aspen Hysys, these unit models are not concerned with the kinetic parameters of the combustion reaction. Though, IDAES also has unit models for reactors and reaction packages for specific chemical reactions. With reaction packages users can define their own chemical reactions with power law kinetic parameters in a variety of forms of equations. Much like in Aspen Hysys, IDAES can also incorporate other heat exchanger units to simulate a steam boiler.

Deng et al. (2023) notes a gap in the current literature being the combined optimisation of the chemical reaction network and the heat integrated water network. This highlights the need for a framework for modelling and optimising the energy network in conjunction with the chemical reaction network. This relationship is exemplified in the nexus that is combustion boilers for the generation and distribution of heat energy as steam by the combustion of fuels. IDAES's reaction modelling capabilities and its flexibility for integrated techniques such as optimisation make it suitable for addressing this gap in the literature as a digital tool. In this paper by Ghosh et al. (2024), IDAES is used to model a chemical process and optimise it for both minimised cost and CO_2 equivalent emissions. This example models its chemical reactions using reduced order kinetics, which makes this example relevant to the chemical reaction network considerations made by Deng et al. (2023). The paper highlights key advantages of equation-oriented modelling framework that is IDAES. The framework is scalable, customizable, and can reproducibly analyse systems for optimisation at varying conditions. For the case of the paper, these conditions vary by shale composition. The same advantages for system analysis and optimization are highly applicable to boiler system modelling and optimization. This solidifies the IDAES framework as an attractive basis for digitalization of steam boiler optimization for decarbonisation.

4.0 Kinetic modelling of combustion reactions

As established in section 1, steam boilers are the focal point of most energy intense industries New Zealand. Regarding decarbonization, of major interest is the combustion chemical reaction which drives energy generation in a steam boiler. This is because combustion ultimately contributes significant greenhouse gas emissions primarily in the form of CO_2 emissions. An example of this is demonstrated for methane fuel (see reaction 1 in this section). To sufficiently model the steam boiler system for the purposes of optimizing for decarbonization, it is necessary to model and simulate the thermal chemical combustion reaction.

4.1 Reaction Kinetic Modelling

The fundamentals of reaction kinetic modelling are based on the power law which has been used for hundreds of years for modelling chemical rates of reaction, originating with the law of mass action by Lund (1965). A simple example of the power law for a reaction rate equation including the temperature dependent term k and the concentration dependent term A in equation (1).

$$r_A = k[A]^n \quad (1)$$

In practice as demonstrated by the research referred to below, the kinetic parameters are obtained by comparing the equation to experimental results. The parameters are then fitted to the experimental results. Power law kinetic parameters can then be inputted into a process simulator to simulate the reaction with results comparable to reality.

Power law kinetics are still used in the 21st century. This research for example by Li et al. (2005) chooses a power law kinetic model for the catalysed high temperature water gas shift reactor. This is as compared to other kinetic models for accuracy to experimental industrial data. Power law kinetics are still useful for accurate reaction modelling.

Developments to the power law equation have yielded other kinetic models such as the Langmuir-Hinshelwood-Hougen-Watson (LHHW) kinetic model made to characterize catalyst adsorption of reaction components. LHHW kinetics use more terms or parameters but will more accurately model the kinetics of catalyst dependent reactions. For example, Akram et al. (2020) creates a power law model and multiple LHHW models for a catalysed reaction. Each model is compared to the experimental data. The study ultimately suggests use of one of the LHHW models due to lower error percent as compared to the experimental data. This highlights the superior accuracy of a more detailed kinetic model.

Research has been done to go beyond the simplifying assumptions of power law kinetic models by considering greater fundamental detail of the relevant mechanics. One approach is to consider first principles in a molecular dynamics simulation as done for example by Zhang et al. (2023) for determining kinetic models for methane-air combustion. Zhang et al. (2023) ultimately end up with a 30-step reduced first principle kinetic model which can be used for sufficiently accurate simulation of methane air combustion.

One of the applications of more detailed kinetics is simulation in Computational Fluid Dynamics (CFD). CFD is more concerned with the internal fluid flow behaviour and spatial composition evolution of the chemical reaction which is typically abstracted away in a single step in reactor model simulations. Zettervall et al. (2021) for example recommends a kinetic mechanism with 22 species and 104 reactions. This however is mechanistic detail that goes beyond the scope of application in modelling for a process simulator digital tool.

Overall, much work has been done on forming kinetic models of chemical reactions ranging from highly detailed to simplistic. To paraphrase Salvato et al. (2010), to perform simulations within the range of computationally practical time, kinetic models should be the proper compromise between accuracy and efficiency, i.e. between detail and simplicity. In process model simulations, simplicity and efficiency is preferred because of the complexity and relevance of a full process system beyond an individual reaction.

4.2 Kinetic Modelling for Process Simulation

The accuracy of kinetic models is still important to process simulation. For example, studies of the kinetic model impact on process simulation results have been done on methanol synthesis and methane reforming processes by Bisotti et al. (2022) and Quirino et al. (2021) respectively. The kinetic model impacts how well the process simulation behaviour can be reliably predicted as compared to known process data. Process system behaviour prediction is also relevant to steam boiler systems, so, the same reliable accuracy is important when considering fuel combustion kinetic models.

In consideration of reaction kinetics to process simulation, the focus circles back to the two process simulator examples compared in section 2. Those being Aspen Hysys and IDAES established to be capable of reaction modelling and simulation. The study by Bisotti et al. (2022) for example uses Aspen Hysys for process simulation prediction. A key point of import in the steam boiler operation is the energy balance. Enthalpy of reaction is largely defined by the extent or conversion of the reactions. The minimum specification, conversion, is used in the simplest reactor unit in Aspen Hysys, the ‘conversion reactor’, which only specifies the extent of reaction (Aspen Technology, 2004b).

The heat of reaction is subsequently directly calculated by thermodynamic enthalpy heat of reaction. A step above in complexity from the 'conversion reactor' are the ideal reactors that include continuously stirred reactor and plug flow reactor. These reactors add residence time and physical dimensions as a parameter. If the kinetic model is sufficiently accurate, the reactor unit operation can accurately model the extent of reaction with respect to residence time and the subsequent heat of reaction. There are a range of Aspen Hysys options for defining the reaction kinetic model. Models such as power law, LHHW, and equilibrium can be defined.

IDAES has options for reactors and reaction kinetic models. There are built in unit models like Aspen Hysys such as equilibrium reactor and plug flow reactor. Regarding kinetic modelling with IDAES, they can be made with the generic reaction package framework. This allows kinetic modelling like van 't Hoff equation for equilibrium constant or power law for reaction rate. Detailed rate forms could be made custom considering the flexibility of the IDAES platform. Ghosh et al. (2024) is an example of reaction kinetics being modelled with IDAES.

Overall, the comparison for suitability to reaction modelling is even, and the conclusions remain as was evaluated in section 2.

4.3 Combustion Considerations: Fuel and Flue

Considering the New Zealand energy context as explained in section 1, an initial target for reaction modelling are combustion reactions in context of significant steam boiler use. By the New Zealand emissions reduction plan (see section 1), fuel switching and process optimisation are points for advancement. A component process simulation digital tool interested in the emissions and energy behaviour must be able to simulate the relevant fuel combustion compositionally and energetically.

Methane sourced from natural gas makes up 28% of New Zealand energy use (Ministry of Business & Employment, 2024). It makes up a significant part of the energy mix and so is useful to model as the fuel source for a steam boiler.

Methane is an attractive fuel for fuel switching from a practical perspective for both emissions and resource sustainability reasons. Methane is considered a cleaner fuel as at emits less CO2 per unit of energy released compared to other more heavier hydrocarbon fuels. Methane fuel as it is sourced as natural gas is also compositionally cleaner as compared to other fuel sources. It is mainly composed of around 85% methane with small compositions of CO_2, N_2, C_2H_6, and other higher alkane hydrocarbons. The resulting flue gas therefore contains less diversity of hazardous or Greenhouse Gas (GHG) components. Other common fossil fuels by comparison like wood or coal have much more complex composition with greater variety in GHG components in the flue.

Methane, despite being a GHG when sourced as natural gas, has potential to being a renewable energy source. There are multiple processes for renewable methane generation including anaerobic digestion and power to methane that have been reviewed by Qian et al. (2025) and Ghaib and Ben-Fares (2018) respectively. Renewable process pathways like these make methane an increasingly desirable fuel and chemical feed stock source as the technology continues to advance.

Methane combustion can be simpler to simulate due to its physical properties. It is a non-polar real molecule so only requires the relatively simple Peng-Robinson Physical Property Package (Carlson, 1996). The reaction is also in the single vapour phase. Other common fuels like petrochemicals and biomass have greater inherent complexity in composition, physical properties, and reaction pathways, which make them more difficult to conveniently simulate accurately.

Methane combustion is a non-elementary reaction. As opposed to a single step kinetic model, it can instead be modelled as multiple steps of elementary reactions to simulate the relevant intermediary steps. Each elementary reaction can each be modelled with power law kinetics. For example, abstraction of the methane combustion reaction can allow a one-step or two-step global reaction with

fewer kinetic parameters (see reactions 1 to 3) (Li et al., 2019). As mentioned above, a simpler kinetic model with fewer parameters makes for higher computational efficiency.

Combustion of methane one global step.

$$CH_4 + O_2 \rightarrow CO_2 + H_2O \quad (1)$$

Combustion of methane two global step.

$$CH_4 + \frac{3}{2}O_2 \rightarrow CO + 2H_2O \quad (2)$$

$$CO + \frac{1}{2}O_2 \rightarrow CO_2 \quad (3)$$

It is clearly feasible to simulate methane combustion as kinetic models and parameters are available, such as those by Li et al. (2019).

When it comes to the more difficult fuels such as coal, or wood, there are more non-idealised phenomena that significantly impact model accuracy. the ash content in the flue becomes more relevant to the steam boiler heat transfer modelling. This adds another layer of dynamic behaviour contributing to the system. Much research has gone into studying these behaviours.

Zevenhoven et al. (2003) applies a database of fuel behaviours with CFD as a tool for predicting the spatial ash deposit formation. The study finds importance of calculating the effective emissivity and absorptivity of the flue gas, as well as the particle suspension.

Rousseau et al. (2023) explores coal fired boilers by comparing one dimensional thermofluidic network modelling to site data. The study recommends a high particle load model for high ash fuels like coal, as it also has an applicable range for low particle loading scenarios. The study evaluates how different load models calculate the fly ash pressure-based absorption coefficient, leading to support of the conclusion that the high particle load model is more certain in this regard.

There are a lot of variables to consider when modelling flue ash radiative emittance behaviour. There are two approaches, one being models based on the physical processes that calculate the scattering and optical constants of ash. The other type are empirical models that have simple expressions but are limited by having narrow applicable ranges (Yao et al., 2020). The study notes a gap in availability of empirical models that account for a multitude of factors, highlighting importance of using emittance measurement data from a database to drive empirical models.

The efforts of the above studies highlight the added layer of complexity that ash particle behaviour brings to steam boiler modelling with the need of supporting empirical models with sufficient databases of measurements. Such types of data exist, for example by Jones et al. (2019), offering a compilation of high temperature emissivity data. Overall, the modelling of flue ash effects on heat transfer can be modelled, as it parallels reaction kinetics in that experimental data must be compared to validate parameters and models.

5.0 Conclusion

This literature review reviewed the literature and evaluated the technological readiness and available research scope for the development of digital tools for the modelling and optimisation of thermal combustion reactions in steam raising boiler systems.

In section 2, the practical and geo-economical New Zealand context in which steam boilers are being used was reviewed. New Zealand has significant motivations for process energy decarbonisation, with steam boilers being a significant contributor to energy generation and GHG emissions

In section 3, relevant advancements and recent technological paradigms were analysed. Digitalization and digital tool applications have already been utilised practiced for research purposes in the literature. More recent technological paradigms like open-source and Industry 4.0 show their prevalence in digital tools like IDAES. IDAES is determined to be suitable for the basis of an advanced digital tool for steam boiler combustion modelling, simulation, and optimisation due to technology paradigms enhancing its adaptability and flexibility.

Section 4 reviews the detailed modelling of steam boilers regarding the chemical reactions and heat transfer. Much research has gone into modelling and prediction simulating the relevant phenomena like the combustion reaction, ash deposition, and heat transfer. This includes specific modelling cases for steam boilers. The research done in these areas form the modelling foundation that can be applied to a digital tool using the digitalization capacity of the technology assessed in section 4.

In conclusion this literature review has evaluated that there is presently technological readiness for the digitalization of modelling and optimising combustion reactions for steam boilers. However, work remains to be done to implement an adaptable digitalization approach in this area of modelling and simulation.

References

The 17 Sustainable Development Goals. (2015). https://sdgs.un.org/goals

Accelerating the Decarbonisation of Process Heat. (2021). https://www.eeca.govt.nz/assets/EECA-Resources/Research-papers-guides/Accelerating-the-decarbonisation-of-Process-Heat.pdf

Akram, M. S., Aslam, R., AlHumaidan, F., & Usman, M. (2020). An exclusive kinetic model for the methylcyclohexane dehydrogenation over alumina-supported Pt catalysts. *International Journal of Chemical Kinetics*, *52*. https://doi.org/10.1002/kin.21360

Ansys, I. (2024). *Ansys Fluent*. Ansys. https://www.ansys.com/products/fluids/ansys-fluent

Aspen Technology, I. (2004a). Aspen HYSYS Customization Guide. https://sites.ualberta.ca/CMENG/che312/F06ChE416/HysysDocs/AspenHYSYSCustomizationGuide.pdf

Aspen Technology, I. (2004b). Aspen HYSYS User Guide. https://sites.ualberta.ca/CMENG/che312/F06ChE416/HysysDocs/AspenHYSYSUserGuide.pdf

Aspen Technology, I. (2024). *Get a Jump on Green Hydrogen Projects with Electrolyzer Models*. Aspen Technology, Inc. https://www.aspentech.com/en/resources/on-demand-webinars/get-a-jump-on-green-hydrogen-projects-with-electrolyzer-models

Aspen Technology, I. (2025). *Aspen HYSYS*. Aspen Tech. https://www.aspentech.com/en/products/engineering/aspen-hysys

Bisotti, F., Fedeli, M., Prifti, K., Galeazzi, A., Dell'Angelo, A., & Manenti, F. (2022). Impact of Kinetic Models on Methanol Synthesis Reactor Predictions: In Silico Assessment and Comparison with Industrial Data. *Industrial & Engineering Chemistry Research*, *61*(5), 2206-2226. https://doi.org/10.1021/acs.iecr.1c04476

Bountouri, L. (2017). 3 - Digitization. In L. Bountouri (Ed.), *Archives in the Digital Age* (pp. 29-36). Chandos Publishing. https://doi.org/https://doi.org/10.1016/B978-1-84334-777-4.00003-7

Bynum, M. L. (2021). *Pyomo — Optimization Modeling in Python* (Vol. 67). Springer. https://doi.org/10.1007/978-3-030-68928-5

Carlson, E. C. (1996). Don't Gamble with Physical Properties for Simulations. *Chemical Engineering Progress*, *92*(10), 35-46.

Concepts — IDAES. (2024). The IDAES Project. https://idaes-pse.readthedocs.io/en/stable/explanations/concepts.html

de Beer, J., & Depew, C. (2021). The role of process engineering in the digital transformation. *Computers & Chemical Engineering*, *154*, 107423. https://doi.org/https://doi.org/10.1016/j.compchemeng.2021.107423

Deng, J., Zhou, C., & Wang, J. (2023). Approaches and application of heat and water network integration in chemical process system engineering: A review. *Chemical Engineering and Processing - Process Intensification*, *183*, 109263. https://doi.org/https://doi.org/10.1016/j.cep.2022.109263

Gani, R., Rgensen, S. B. J., & Jorgensen, S. B. (2001). *European symposium on computer aided process engineering - 11*. Elsevier Science & Technology.

Ghaib, K., & Ben-Fares, F.-Z. (2018). Power-to-Methane: A state-of-the-art review. *Renewable and Sustainable Energy Reviews*, *81*, 433-446. https://doi.org/https://doi.org/10.1016/j.rser.2017.08.004

Ghosh, K., Salas, S. D., Garciadiego, A., Dunn, J. B., & Dowling, A. W. (2024). Multiscale Equation-Oriented Optimization Decreases the Carbon Intensity of Shale Gas to Liquid Fuel Processes. *ACS Sustainable Chemistry & Engineering*, *12*(28), 10351-10362. https://doi.org/10.1021/acssuschemeng.4c00933

Jones, J. M., Mason, P. E., & Williams, A. (2019). A compilation of data on the radiant emissivity of some materials at high temperatures. *Journal of the Energy Institute*, *92*(3), 523-534. https://doi.org/https://doi.org/10.1016/j.joei.2018.04.006

Kasivisvanathan, H., Ubando, A. T., Ng, D. K. S., & Tan, R. R. (2014). Robust Optimization for Process Synthesis and Design of Multifunctional Energy Systems with Uncertainties. *Industrial & Engineering Chemistry Research*, *53*(8), 3196-3209. https://doi.org/10.1021/ie401824j

Lasi, H., Fettke, P., Kemper, H.-G., Feld, T., & Hoffmann, M. (2014). Industry 4.0. *Business & Information Systems Engineering*, *6*(4), 239-242. https://doi.org/10.1007/s12599-014-0334-4

Lee, A., Ghouse, J. H., Eslick, J. C., Laird, C. D., Siirola, J. D., Zamarripa, M. A., Gunter, D., Shinn, J. H., Dowling, A. W., Bhattacharyya, D., Biegler, L. T., Burgard, A. P., & Miller, D. C. (2021). The IDAES process modeling framework and model library—Flexibility for process simulation and optimization. *Journal of Advanced Manufacturing and Processing*, *3*(3), e10095. https://doi.org/https://doi.org/10.1002/amp2.10095

Li, J., Bai, J., Yao, F., & Li, C. (2005). Kinetic study of high temperature water-gas shift reaction over LB type catalyst. *Chinese Journal of Catalysis*, *26*, 25-31.

Li, Z., Li, Y., & Lou, C. (2019, 2019/05/22). Kinetic Simulation of Methane Combustion Reaction: From Mechanism to Application. Applied Energy Symposium: MIT A+B, Boston, USA.

Lund, E. W. (1965). Guldberg and Waage and the law of mass action. *Journal of Chemical Education*, *42*(10), 548. https://doi.org/10.1021/ed042p548

Ministry of Business, I., & Employment. (2017). *Process Heat in New Zealand: Current State*. https://www.mbie.govt.nz/dmsdocument/152-process-heat-current-state-fact-sheet-pdf

Ministry of Business, I., & Employment. (2022). *New Zealand Emissions Reduction Plan*. https://www.mbie.govt.nz/building-and-energy/energy-and-natural-resources/low-emissions-economy/emissions-reduction-plan

Ministry of Business, I., & Employment. (2024). *Energy in New Zealand 2024*. https://www.mbie.govt.nz/building-and-energy/energy-and-natural-resources/energy-statistics-and-modelling/energy-publications-and-technical-papers/energy-in-new-zealand/energy-in-new-zealand-2024

OPEN SOURCE SECURITY AND RISK ANALYSIS REPORT. (2023). https://balwurk.com/wp-content/uploads/2023/06/rep-ossra-2023.pdf

Pietrasik, M., Wilbik, A., & Grefen, P. (2024). The enabling technologies for digitalization in the chemical process industry. *Digital Chemical Engineering*, *12*, 100161. https://doi.org/https://doi.org/10.1016/j.dche.2024.100161

Pistikopoulos, E. N., Barbosa-Povoa, A., Lee, J. H., Misener, R., Mitsos, A., Reklaitis, G. V., Venkatasubramanian, V., You, F., & Gani, R. (2021). Process systems engineering – The generation next? *Computers & Chemical Engineering*, *147*, 107252. https://doi.org/https://doi.org/10.1016/j.compchemeng.2021.107252

Qian, S., Chen, L., Xu, S., Zeng, C., Lian, X., Xia, Z., & Zou, J. (2025). Research on Methane-Rich Biogas Production Technology by Anaerobic Digestion Under Carbon Neutrality: A Review. *Sustainability*, *17*(4).

Quirino, P. P. S., Amaral, A., Pontes, K. V., Rossi, F., & Manenti, F. (2021). Impact of kinetic models in the prediction accuracy of an industrial steam methane reforming unit. *Computers & Chemical Engineering*, *152*, 107379. https://doi.org/https://doi.org/10.1016/j.compchemeng.2021.107379

Rousseau, P., Laubscher, R., & Rawlins, B. T. (2023). Heat Transfer Analysis Using Thermofluid Network Models for Industrial Biomass and Utility Scale Coal-Fired Boilers. *Energies*, *16*(4).

Salvato, L., Viggiano, A., Valorani, M., & Magi, V. (2010). On the simplified ethanol kinetics by means of a computational singular perturbation approach. *Thermal and environmental issues in energy systems*, *2*, 917-922.

Tancredi, G. P. C., Eleonora, B., & and Vignali, G. (2024). Digital twin-enabled process control in the food industry: proposal of a framework based on two case studies. *International Journal of Production Research*, *62*(12), 4331-4348. https://doi.org/10.1080/00207543.2023.2260495

Walmsley, T. G., Patros, P., Yu, W., Young, B. R., Burroughs, S., Apperley, M., Carson, J. K., Udugama, I. A., Aeowjaroenlap, H., Atkins, M. J., & Walmsley, M. R. W. (2024). Adaptive

digital twins for energy-intensive industries and their local communities. *Digital Chemical Engineering*, *10*, 100139. https://doi.org/https://doi.org/10.1016/j.dche.2024.100139

Yao, Y., Jiang, L., Zhang, M., Yang, H., Lü, J., He, W., & Zhou, Q. (2020). Research progress review on the emittance of the ash from coal combustion [Review]. *Meitan Xuebao/Journal of the China Coal Society*, *45*(3), 1170-1178. https://doi.org/10.13225/j.cnki.jccs.2019.0306

Zarafshani, H., Watjanatepin, P., Navare, K., Sauve, G., & Van Acker, K. (2024). SAB: An open-access Python-based integrated platform for fully automated emergent sustainability assessment of biorefineries. *The International Journal of Life Cycle Assessment*, *29*(4), 632-651. https://doi.org/10.1007/s11367-023-02271-w

Zettervall, N., Fureby, C., & Nilsson, E. J. K. (2021). Evaluation of Chemical Kinetic Mechanisms for Methane Combustion: A Review from a CFD Perspective. *Fuels*, *2*(2), 210-240.

Zevenhoven, M., Skrifvars, B. J., Yrjas, P., Backman, R., Mueller, C., & Hupa, M. (2003). Co-firing in FBC: A challenge for fuel characterization and modeling. Proceedings of the International Conference on Fluidized Bed Combustion,

Zhang, T., Shou, L., Chen, L., Wang, H., Long, Y., Wang, Z., & Chen, J. (2023). Detailed and reduced chemical kinetic model of CH4/air mixtures combustion based on high-precision first-principles molecular dynamics simulation. *Chemical Engineering Science*, *281*, 119159. https://doi.org/https://doi.org/10.1016/j.ces.2023.119159

Appendix C2. Estimation of Combustion Stoichiometry for Complex Fuels

Estimation of Combustion Stoichiometry for Complex Fuels

David Dickson[1, a *], Benjamin Lincoln[1] and Bert Downs[1]

[1]The university of Waikato, Hamilton, New Zealand

[a*]dd188@students.waikato.ac.nz

Abstract

A framework for estimating simplified, mass-conserving combustion stoichiometry for complex fuels was developed. It was intended for process modelling applications in the context of industrial decarbonisation. Biomass and subbituminous coal are complex with variable ash content depending on the source. This complicates the formulation of variable reaction stoichiometries. This study proposes a method that uses literature-sourced compositional data and arbitrary molecular weights.

Keywords. Stoichiometry, Biofuel, Ash, GHG Emissions.

1. Introduction

A framework for estimating simplified combustion stoichiometry for otherwise complex fuels is desired. This would contribute to advancing the accessible process modelling for combustion of compositionally complex fuels like biomass, black liquor, and coal. These fuels are worth modelling because they represent the relevant heating solution options amidst the global energy transition towards renewable sources. The stoichiometry is important to model as it quantifies the CO2 emissions that are essential to model for the decarbonisation incentives of global climate change.

In process modelling the stoichiometry encodes for a given reaction the ratio between reactants converting to products. A material inventory analysis could be done before and after a reaction to evaluate what this ratio should be. For complex fuels there is a high chance for inconsistencies, for example the variety of different types of biomasses. There is a benefit in being able to specify differing ash contents or to assess how different ash contents can impact behaviour of processes with such reactions.

Reactions should conserve mass; this leads to difficulties when one would like to adjust the stoichiometry to account for a different fuel ash content. The standard for balancing chemical reactions is to do an elemental balance of atoms which are necessarily conserved outside of nuclear reactions. But ash and complex fuel types are made up of a variety of elements of different weights, so a rigorous elemental balance is unreasonable for these compounds. One solution to the stoichiometry is to simply add more ash to the products stoichiometry to simulate increased ash content being produced from the fuel, but the mass balance then becomes false. A more nuanced approach algebraic approach should be used to conserve mass balance. The equations used should be algebraically linear for efficiency in a computational modelling implementation.

Various literature measurement data for compositional analyses of the various fuels this current paper is interested in. However, they are limited by not being compiled into a usable combustion stoichiometry framework for process reaction modelling. For the accurate stoichiometry estimation of complex impure fuels, multiple points of corroborating literature measurement data that impacts the stoichiometry should be compiled to characterise accurate combustion stoichiometry.

The approach developed in this study was applied to subbituminous coal and wood biomass fuel types.

2. Experimental procedure

Provide sufficient details to allow the work to be reproduced by an independent researcher. Methods that are already published should be summarized and indicated by a reference. If quoting directly

from a previously published method, use quotation marks and cite the source. Any modifications to existing methods should also be described.
This section can contain *Material and methods* and *Theory/calculation*. A Theory section should extend, not repeat, the background to the article already dealt with in the Introduction and lay the foundation for further work. In contrast, a Calculation section represents a practical development from a theoretical basis.

The literature data measurements gathered to deconvolve combustion stoichiometry were as follows:

- Compositional data
- Sub-element molar weight
- Emissions factor data

From these data measurements would be used to estimate stoichiometry parameters and decide molecular weights, largely for nominal mole to mass conversions. The molecular weight is arbitrary if it is used consistently as was done in this study.

Ash was considered as its own molecular component within coal and biomass. But it is not a molecule, so ash itself must be characterised by estimations based on literature. Compositional data compiled by AL-Kharabsheh et al. (2022b) was weighted then used to weight the molecular weight of constituents to determine a nominal molecular weight for ash. This was arbitrary but some molecular weight value should be decided for consistency throughout the methods of this study. The results are as shown in Table 1.

Table.2. Ash compositional estimation by literature measurements for molecular weight and heat capacity.

Component	Wood Ash Compositions from Literature[A]					Average	Cp@ 1000[K][B]	Weighted Cp	mw	Weighted mw
							[J/mol/K]	[J/mol/K]	[g/mol]	[g/mol]
SiO_2	48.96	25	55.52	2.7	29.1	32.3	69	2225.7	60.1	1938.1
Al_2O_3	11.24	0.76	3.11	1.3	10.3	5.3	125	667.8	144.9	774.3
CaO	11.59	35.6	9.92	61	35.6	30.7	54	1660.1	56.1	1723.9
Fe_2O_3	0.6	2.76	0.41	1.3	6.55	2.3	150	348.6	159.7	371.1
MgO	5.05	1.76	2.32	8.7	4.7	4.5	51	229.8	40.3	181.6
Sum:	77.44	65.88	71.28	75	86.25	75.2		5131.9		4989.0
Weighted:	1.03	0.88	0.95	1.00	1.15	1		68.27		66.370

Note.

[A] (AL-Kharabsheh et al., 2022)

[B] (Chase, 1998)

An ultimate analysis of subbituminous coal was estimated by data provided by Mares (2009) and is summarised in Table 2. This was used to estimate its arbitrary molecular weight. This data was also used as a reference point for conserving the elemental balance across the reaction.

Table.3. Ultimate analysis of subbituminous coal used to estimate molecular weight.

Component	mol%	mw [g/mol]	mw weighted
C	0.73	12.01	8.77
H	0.05	1.01	0.05
O	0.17	16.00	2.72
N	0.01	14.01	0.14
ash	0.03	66.37	1.99
Sum	0.99		11.86

Wood biomass was assumed to be pure cellulose with some percentage of ash. Wood is up to 50% cellulose on a dry basis with other dry content including lignin and hemicellulose (Dongre et al., 2024). All three have a similar elemental composition of Carbon, Hydrogen, and Oxygen, with an average standard deviation of these elemental compositions between the three molecules of 0.0556 as calculated by Table 3. Thus, the assumption that all dry matter in wood biomass was reasonable. This also fixed the arbitrary molecular weight to that of cellulose, 162.14 [g/mol].

Table 4. Comparing elemental composition of the common molecular wood constituents Lignin, Cellulose, and Hemicellulose.

		Elemental Composition			Elemental % Composition			Standard Deviation
Element	mw [g/mol]	Lignin	Cellulose	Hemi-cellulose	Lignin	Cellulose	Hemi-cellulose	
Carbon	12.01	31	6	1	40.79%	28.57%	25.00%	0.0828
Hydrogen	1.01	34	10	2	44.74%	47.62%	50.00%	0.0264
Oxygen	16.00	11	5	1	14.47%	23.81%	25.00%	0.0576
Sum		76	21	4			Avg. std. dev.:	0.0556

A literature emissions factor of 26.2 [ton-Carbon/TJ] for subbituminous coal was used as corroborating data to its combustion stoichiometry (Simmons, 2001). A literature emissions factor of 1.7 [kg-CO_2-e/kg] was not used as corroborating data, but instead as a validation reference point for the results of this study (Ministry for the Environment, New Zealand, 2024).

The combustion stoichiometry for biomass wood was nontrivial as it was assumed to cellulose with a known elemental composition that could easily fixed by an element balance. Coal instead used the ultimate compositional analysis in corroboration with the assumed emissions factor to fix CO_2

stoichiometry. All Hydrogen in coal was assumed to convert into H_2O, and all Nitrogen into N_2. The subsequent Oxygen reactant needed to fulfil the fixed CO_2 and H_2O products could then be fixed. These elemental balance relations are expressed in the following equations:

$$v_{N_2} = y_N/2 \quad (1)$$
$$v_{H_2O} = y_H/2 \quad (2)$$
$$v_{O_2} = -\left(y_H/4 + v_{CO_2}\right) + y_{O_2}/2 \quad (3)$$

Where:

v = reaction stoichiometry

y = ultimate analysis coal composition [mol%]

As the ultimate analysis included ash content, ash missing from the stoichiometry products would indicate a net mass imbalance. By converting the stoichiometry to a mass basis, the mass balance could resolve the required products ash stoichiometry. It is also retroactively resolved the nominal coal molecular weight. By this mixed-elemental balance approach, the nominal stoichiometry for coal were summarised in Table 4.

Table 5. Subbituminous coal stoichiometry by the methods of this study

Component	Stoichiometry	Mw [g/mol]
Coal	-1	11.86
O_2	-0.548	32.00
H_2O	0.025	18.01
CO_2	0.621	44.01
N_2	0.005	28.01
Ash	0.005362	66.37

Now there is the issue of adjusting ash content relative to the nominal stoichiometry into a new ash content. Taking into consideration the desired mol% ash content and initial relative stoichiometries of fuel and ash, the new ash and fuel stoichiometries can be correctly evaluated to the correct new ash content. This was achieved by the new equations developed in this study shown in Equations 4 and 5.

$$v_{ash,n} = \left(\left(\left(x_{ash} - v_{as\ ,i} mw_{as}\ \right)\left(-v_{fuel,i} mw_{fuel}\right)\frac{1}{1-\left(x_{as}\ - v_{as\ ,i} mw_{as}\ \right)}\right) + v_{as\ ,i} mw_{as}\ \right)/mw_{as} \quad (4)$$

$$v_{fuel,n} = -\left(\left(\left(x_{ash} - v_{as\ ,i} mw_{as}\ \right)\left(-v_{fuel,i} mw_{fuel}\right)\frac{1}{1-\left(x_{as}\ - v_{as\ ,i} mw_{as}\ \right)}\right) + v_{fuel,i} mw_{\text{fuel}}\right)/ mw_{\text{fuel}} \quad (5)$$

where:

$v_{ash,n}$ = new stoichiometry for ash [mol]

$v_{fuel,n}$ = new stoichiometry for fuel [mol]

$v_{as\ ,i}$ = initial stoichiometry for ash [mol]

$v_{fuel,i}$ = initial stoichiometry for fuel [mol]

x_{ash} = fuel ash content [mol%]

mw_{ash} = molecular weight of ash [g/mol]

mw_{fuel} = molecular weight of fuel [g/mol]

As biomass wood was nominally composed of only cellulose in its stoichiometry, these equations were applied to account for an ash content of 3 wt% to yield the summarised biomass stoichiometry in Table 5.

Table 6. Biomass wood combustion stoichiometry with ash.

Component	Stoichiometry	mw [g/mol]
Biomass	-1.0309	162.139
O2	-6	31.998
CO2	6	44.009
H2O	5	18.015
Ash	0.07556	66.37

3. Results

The ash-adjustment equations had to be robust to a variety of initial stoichiometries, molecular weights, and either increasing or decreasing from nominal ash content. This was tested for biomass wood combustion stoichiometry, and the results were illustrated in Figure 1. Notice that the added fuel and ash stoichiometries in ash basis (bottom) remain equivalent.

	initial	final (overwritten)		initial	final (overwritten)		initial	final (overwritten)	
mw_fuel (g/mol)	162.1394			162.1394			162.1394		
mw_ash (g/mol)	66.37		new mols:	66.37		new mols:	66.37		new mols:
-fuel (N_stoich)	1	167.154021	1.030928	1.1	181.387223	1.118712	1.1	177.787088	1.096508
ash (N_stoich)	0	5.014620619	0.075556	0.0002	3.047156687	0.045912	0.0005	-0.533066876	-0.00803
mass_ash%(of 1m BM)	0.03	0.03		0.03	0.01679918		0.03	-0.002998344	
mol_ash%(of 1N BM)	0.073288865	0.073288865		0.073288865	0.041039763		0.073288865	-0.007324841	
added mass_BM	5.014620619			3.033882687			-0.56625188		
equiv mass stoich ash	0	12.25052853		0.013274	7.443888547		0.033185	-1.302762218	
-fuel(m_stoich)	162.1394	27102.25261		178.35334	29410.01545		178.35334	28826.2918	
ash (m_stoich)	0	332.8203705		0.013274	202.2397893		0.033185	-35.37964854	
mass_ash%	0	0.012280174		7.44253E-05	0.006876562		0.000186063	-0.00122734	
	162.1394	167.154021		178.35334	181.387223		178.35334	177.787088	
added mass fuel	0	5.014621		0.013274	3.047157		0.033185	-0.533067	
added mass ash		5.014621			3.033883			-0.566252	

Figure 3. Demonstrating results of mass-conserving ash stoichiometry balancer.

4. Discussion

This technique is significant because it can generally be applicable to many complex fuels that have uncertain, but desire to be fixed, ash contents for the purpose of reaction modelling. The algebra can also be re used in a more computationally optimised code implementation.

Moisture content was not included, but fixing water content would be as trivial as re-specifying for water on either side of the stoichiometry so it was not a focus of this study.

The purpose of this study was not for chemical rigour, but for a practical approach to combustion stoichiometry for lean computational reactions.

5. Conclusions

This study developed an ash combustion stoichiometry framework for complex fuels. It estimated reaction stoichiometry properties by using corroborating literature data on subbituminous coal and wood biomass. Then a correlation was developed to make ash content flexible whilst conserving mass balance.

Acknowledgements

Thank you to my co-authors Benjamin Lincoln and Bert Downs for giving me the inspiration for pursuing the ash mass stoichiometry balance equations and guiding my intuitions.

References

AL-Kharabsheh, B. N., Arbili, M. M., Majdi, A., Ahmad, J., Deifalla, A. F., & Hakamy, A. (2022). A Review on Strength and Durability Properties of Wooden Ash Based Concrete. *Materials*, *15*(20), 7282. https://doi.org/10.3390/ma15207282

Chase, Jr., M. W. (1998). *NIST-JANAF Thermochemical Tables, Fourth Edition, Monograph 9 (Part I and Part II)*. American Chemical Society and American Institute of Physics for the National Institute of Standards and Technology. https://www.nist.gov/data/janaf-thermochemical-tables

Dongre, P., Nagardeolekar, A., Corbett, D., & Bujanovic, B. M. (2024). Chapter 1—Chemical aspects of the composite structure of wood and its recalcitrance to enzymatic hydrolysis. In D. Kumar, S.

Kumar, K. Rajendran, & R. C. Ray (Eds.), *Sustainable Biorefining of Woody Biomass to Biofuels and Biochemicals* (pp. 1–41). Woodhead Publishing. https://doi.org/10.1016/B978-0-323-91187-0.00012-6

Mares, T. E. (2009). *An investigation of the relationship between coal and gas properties in the Huntly coalfield, New Zealand*. University of Canterbury.

Ministry for the Environment, New Zealand. (2024). *Measuring emissions: A guide for organisations—2024 detailed guide* (No. ME 1829). Ministry for the Environment. https://environment.govt.nz/assets/publications/Measuring-Emissions-2024/Measuring-emissions_Detailed-guide_2024_ME1829.pdf

Simmons, T. (2001). CO2 EMISSIONS FROM STATIONARY COMBUSTION OF FOSSIL FUELS. In *Good Practice Guidance and Uncertainty Management in National Greenhouse Gas Inventories, Chapter 2 Energy*. International Panel on Climate Change. https://www.ipcc-nggip.iges.or.jp/public/gp/english/