University of Auckland
Department of Engineering Science

Global Optimisation Carbon Pricing Initiative (GOCPI)

Author: Connor McDowall
Supervisor: Rosalind Archer

October 26, 2020

Abstract

Acknowledgements

Rosalind Archer

I am most thankful for Rosalind. She was a guiding light in a most turbulent time. 2019 and 2020 were incredibly tough years. Our meetings were a weekly highlight, giving a sense of normalcy during the depths of the COVID-19 Lockdowns. Often accompanied by her two corgis, we embarked into the unknown in building a Python-based open source energy modelling tool. Surprisingly, energy modelling is quite complex. Our discussions encompassed a diverse range of intellectually stimulating topics, from the phenomena of the West Texas Intermediate (WTI) future contracts going negative, to speculating on the transformation we will see in the education sector moving forward. Thank you for helping foster a growth mindset and build the confidence to not be afraid to fail.

Declaration of Contribution

I proposed this project. I am the sole contributor.

Contents

List of Figures

List of Tables

1 Introduction

The imminence of the energy transition is clear. The coronavirus pandemic accelerated a shift away from fossil fuels and adoption of Sustainable technologies to drive economic recovery. A thorough literature review 2 informs the need for rapid transformation before irreversible damage is done. Economic models predict unfavourable consequences if no swift action is taken. Financial concepts such as Return on Investment (ROI) and Net Present Value (NPV) analysis drive decisions related to sustainable investment. Sustainable technologies have seen significant cost reductions over the last decade creating positive prospects for investors. These developments improve the feasibility of the energy transition in a timely manner. However, there are educational disparities between policy makers, governments, industry groups and consumers. Major stakeholders know a transition must happen swiftly, but not how a successful transition will impact their energy system or the implementation details. Stakeholders have different needs and reside within different energy systems. They may address changes in their best interest without considering the other stakeholders equitably. It is important to make the transition as transparent as possible to deliver equitable outcomes for all stakeholders.

The opaque nature of the transition, in terms of the implementation for different regions, is caused by the sophistication and inaccessibility of energy modelling. Energy modelling involves overcoming several technical barriers. Large volumes of energy-related data is required to build accurate systems. Most of the time this data is proprietary and hard to access. Energy systems are expressed as mathematical models. They can be linear, non linear, integer or mixed integer programmes. These models can be difficult to formulate and hard to understand. The complexity of energy modelling leads to the need for powerful commercial solvers to solve the energy systems. Most commercial solvers are expensive and inaccessible to most users. The concepts underpinning energy systems are interdisciplinary, often encompassing finance, economics, engineering, statistics and mathematics. These issues and complexities make it difficult to equitably evaluate energy policy and investment. Most noticeably, it is hard to define an effective transition strategy which aligns with both the United Nations Sustainable Development Goals and the Paris Agreement.

My proposed solution is to develop an accessible, scalable energy system modelling tool to address these issues. The intention is remove the aforementioned sophistication and enable users to model their own energy systems. This will remove educational disparities to bring more stakeholders to the table when discussing how to best implement the energy transition. Hopefully, this will lead to more equitable outcomes before it is too late.

2 Literature Review

2.1 An Introduction to Energy

Energy built the world we live in. Industrial sectors use energy to power machining and production processes. Trade uses oil-based products to fuel shipping and logistics networks. The technology sector uses energy to supply power to data centres to facilitate internet services. The manufacturing sector use the chemical properties of oil and natural gas to produce glass, synthetic, and explosive products. In summary, energy has a role in every industry. Energy comes in many forms. The International Energy Agency (IEA) aggregates energy products into six key fuels: Coal, Gas, Oil, Renewables, Electricity, and Nuclear [1]. Coal, Gas, and Oil are non-renewable fossil fuels with high energy intensities and carbon emissions. Nuclear energy is a non-renewable resource as the uranium used in production is finite. Renewables utilise technologies to produce energy from infinitely available wind, solar, geothermal and hydro resources. These technologies emit low to no carbon emissions. Regardless, energy will continue to drive our way of life.

2.2 Energy, Emissions, and the Economy

Academics spent decades hypothesizing causal relationships between economic growth, energy consumption, and carbon dioxide (CO_2) emissions. One study explored this hypothesis by applying panel unit root tests, panel co-integration methods, panel causality tests, Fully Modified Ordinary Least Squares (FMOLS), and Dynamic Ordinary Least Squares (DOLS) estimation methods [23]. Another study explored both linear and non-linear causality using Granger Causality and Vector Autoregressive models [12]. Although these modeling techniques are not explored further in this literature review or project, there results show a relationship between economic growth, energy consumption, and carbon dioxide (CO_2) emissions. Subsequently, we must improve our understanding of the three variables and how their relationship informs energy strategy.

2.2.1 Energy

Global energy consumption increased annually over the last decade. As outlined in BP's Statistical Review of World Energy, world consumption grew from 11705.1 Million Tonnes Oil Equivalent (Mtoe) in 2008 to 13864.9 Mtoe in 2018. World consumption grew 2.9% per annum in 2018 and 1.5% per annum for the period 2007-2017. In 2018, 85% of global energy consumption came from Oil, Natural Gas, and Coal. Global oil consumption increased by 1.2%, global coal by 1.4%, global gas by 5.3%, global nuclear by 2.4%, global hydro-electricity by 3.1%, and global renewables by 14.5% [44]. The 14.5% growth in renewables from 2017 to 2018 highlights how renewables are increasingly competitive as emphasised in the 2020 Deloitte US Renewable Energy Outlook. Flat electricity load growth, declining costs, and the maturation of energy storage all contribute to decreasing costs and increasing competitiveness. Decentralised energy networks with remote renewable micro-grids bolster resilence from increasingly frequent adverse weather events. Collaborative efforts lead to newly created efficiencies and the nurturing of new ideas, investment, and leadership. The combination of stakeholders' demand for increased resiliency in power networks, and collorative efforts to foster innovation in renewables, will continue to drive renewable uptake [52].

The fallout from the 2008 financial crisis led to a decrease in global consumption. The COVID-19 pandemic is ravaging global oil markets with an approximate 1550 Mtoe (33% [51]) decrease in consumption. The fall in demand from COVID-19, and the inability to store more oil, will catalyse significant change. Governments are distributing stimulus packages in response to the pandemic. Consumption is likely to bounce back after the fallout from COVID-19. Stimulus and the disruption to the energy industry may change the way energy is produced and consumed moving forward.

2.2.2 Emissions

Greenhouse gas emissions (GHG) are adverse by-products from the production or consumption of fossil fuels. Carbon dioxide (CO_2) and Nitrous Oxide (N_2O) originate from combusting fossil fuels. Methane (CH_4) is emitted from fossil fuel production processes. GHGs act as insulators, slowing the rate energy leaves the earth's atmosphere by absorbing energy. Global Warming Potential (GWP) measures how much energy one tonne of gas will absorb relative to one tonne of CO_2. The GWP for CO_2 is 1 and remains in the atmosphere for thousands of years. The GWP for N_2O is 265-298 over 100 years. Lastly, the GWP for CH_4 is 28-36 over 100 years [4]. Carbon dioxide emissions from the consumption of oil, gas, and coal for combustion increased annually over the last decade. This subset of global emissions rose from 30336.7 Mtoe in 2008 to 33890.8 Mtoe in 2018, growing 11.72% over this period [44].

It is hard to argue anthropomorphic contributions to global emissions are not playing a role in climate change. The Inter-government Panel on Climate Change (IPCC) are an inter-government body of the United Nations (UN). The IPCC inform relevant parties on the scientific basis of risk-induced climate change. The IPCC also convey it's natural, economic, and political implications. The IPCC prepared a special report on Global Warming of 1.5 °C, based on an assessment of around 6,000 peer-reviewed publications. This report confirms climate change is affecting livelihoods and ecosystems worldwide. Approximately between 0.8 °C and 1.2 °C of global warming above pre-industrial levels is attributable to human activities. It is likely to reach 1.5°C between 2030 and 2052 if the current rate of increase continues. The warming from the pre-industrial era to the present will persist for centuries to millennia. Climate models are predicting increased mean temperatures in land and sea regions, hot extremes, heavy precipitation, drought, and precipitation deficits. Biodiversity and ecosystem impacts, such as species loss and extinction, will increase. Ocean temperatures, acidity, and oxygen scarcity will increase. Global warming will impose risks to health, livelihoods, food security, water supply, human security, and economic growth. All expected to increase unless severe action is taken [17]. Subsequently, emissions will continue to play an important role in our future.

2.2.3 Economy

The global economy grew substantially over the last century. Worldwide Gross Domestic Product (GDP) per capita (constant 2010 USD) grew from $3746 in 1960 to $10891 in 2018 [10]. Gross Domestic Product increased from $66.1 trillion (Current USD) in 2010 to $85.9 trillion in 2018 (Current USD). The World Bank segments GDP into four key segments: Agriculture, Industry, Manufacturing, and Services. Agriculture corresponds to International Standard Industrial Classification (ISIC) divisions 1-5 which cover forestry,

fishing, cultivating crops, and livestock production. Industry corresponds to ISIC divisions 15-37 covering mining, manufacturing, construction, electricity, water, and gas. Manufacturing corresponds to ISIC divisions 10-45 covering businesses which physically or chemically transform materials of components into new products. Services corresponds to ISIC divisions 50-99 covering wholesale trade, retail trade, transport, and government, financial, professional, and personal services. In 2018, Agriculture, Industry, Manufacturing, and Services made up 3%, 25%, 16%, and 65% of GDP respectively [11]. On inspection, energy related products are key inputs to producing segment outputs. Traditionally, production is a function of capital and labour. Energy should be included in this set with theoretical and time series analysis supporting this claim. Empirical and theoretical evidence suggests energy availability, coupled with energy use and output, plays a key role in enabling growth [48]. Unfortunately, the COVID-19 pandemic drew to a close this period of unprecedented growth as the economy is forecast to plunge into a deep recession.

It is clear action must be taken to reduce anthropomorphic emissions while meeting the needs the economy after reviewing energy, emissions, and the economy.

2.3 Current Political Action

2.3.1 Paris Agreement

Over the last few decades, there has been debate on anthropomorphic emissions causing climate change. During this period, the UN began the United Nation's Climate Change Regime to ratify global agreements to combat climate change. There have been three major phases to this regime. The first phase was negotiating, adopting, and ratifying the United Nations Framework Convention on Climate Change from 1990 to 1995. The second phase was the adoption of the Kyoto Protocol on the 11th of December 1997. Unfortunately, the protocol took until the 16th of February 2005 to be ratified due to a complex ratification process. The Kyoto Protocol operationalized the United Nation's Framework Convention on Climate Change. Industrialised countries were committed to reduce and limit GHG emissions in accordance with targets agreed on an individual basis. The protocol only bound developed countries as they were deemed mostly responsible for the state of emissions. The protocol was split into two commitment periods. The first set binding emission targets for 36 industrialised countries, and the European Union, to reduce 5% of emissions compared to 1990 levels. The second commitment was set by the adoption of the Doha Amendment on the 8th of December 2012. This commitment was set to start 2013, end in 2020, and reduce emissions by 18% from 1990 levels. Unfortunately, the second commitment period was not ratified. The protocol established three flexible market mechanisms; International Emissions Trading, Clean Development Mechanisms, and Joint Implementation. The protocol held parties accountable by establishing rigorous monitoring, review, verification, and compliance systems [18].

This protocol was a step in the right direction but there were some inherent issues. A complex ratification process limited uptake. Participating countries had disproportionate obligations to reduce emissions. The commitment periods did not have a long term outlook, contained different subsets of participants, and were not global in nature.

There was another long term co-operative action promoted under the UNFCCC, launched in the Bali Action Plan. Both the second commitment and the co-operative action were to conclude at the 2009 Copenhagen Conference. However, this conference ended in

disappointment as there wasn't enough time before the conference to resolve the issues in the regime.

At the conference, the Copenhagen Accord, political in nature, was agreed upon on the final night by a group of states including most major economies. This accord established a bottom up architecture where both targets and actions were set individually then reported internationally. In 2010, the Cancun Agreements incorporated the Copenhagen Accord into the UNFCCC regime. However, all progress to date did not look pass 2020.

This issue was resolved at the 2011 Durban Conference when the Durban Platform for Enhanced Action launched the negotiations which lead to the Paris Agreement. Negotiations continued for years, addressing how to develop an instrument suitable for all parties, and requested the submission of nationally determined contributions (NDCs)[14].

The 2015 United Nations' Climate Change Conference was held in Paris. After some exceptional negotiating from French Foreign Minister Laurant Fabius, the Paris Agreement was adopted on the 12th of December 2015, and ratified on the 4th of November 2016.

The Paris Agreement's main objective is to strengthen the global response to climate change by keeping the temperature rise this century below 2° above pre-industrial levels. The agreement also pursues efforts to limit the increase to 1.5°. The agreement addresses the following across 29 articles: long term temperature goals, global peaking targets, mitigation efforts through nationally determined contributions (NDCs), sink and reservoir use, market/non-market approaches, adaptations, loss/damage mitigations from the adverse effects of climate change, financial support, technology support, capacity-building support, education, reporting systems, and global stocktake procedures [19], [41]. Article six of the Paris Agreement outlines mitigations and NDCs which include the following carbon pricing initiatives: Emissions Trading Schemes and Carbon Taxes.

2.3.2 Carbon Pricing Initiatives

Emissions Trading Systems (ETS) facilitate emission reductions where cheapest. Polluters who find it easier to reduce emissions can sell emission allowances to polluters who struggle to lower emissions. There are two types of ETS. Firstly, Cap and Trade. Secondly, Baseline and Credit Systems. Cap and Trade sets an upper limit on emissions with emission credits either grandfathered or auctioned. Most emission credits under this system are grandfathered with proceeds captured by existing polluters. Most cap and trade systems are localised regionally or nationally. However, the European Union (EU) implemented the EU ETS to trade across the EU. Baseline and credit systems don't have a fixed level of emissions. Polluters can reduce emissions to earn credits to sell to other polluters who need to meet regulations [9].

Carbon Taxes are either implicit or explicit. An implicit tax is incorporated into the price of a GHG intensive product e.g. retail fuels. An explicit tax is a price per quantity of emissions produced e.g. \$10 per tonne of CO_2 equivalent.

Carbon pricing initiatives are increasingly recognized as instrumental to cost-effectively deliver the transition to low-carbon societies. The Organisation for Economic Co-operation and Development (OECD), International Monetary Fund (IMF) and the IPCC all recognise the need to strengthen these initiatives.

57 carbon pricing initiatives are either currently implemented, or scheduled for implementation, as at April 1st 2019. 28 ETFs are spread across national and subnational jurisdictions with 29 carbon taxes primarily implemented at a national level. These pricing initiatives only cover approximately 20% of Greenhouse Gas Emissions (11 GtCO_2e). There is variation in carbon prices from less than US\$1/tCO_2e to US\$127/tCO_2e. Unfortunately most carbon tax price increases are linked to inflation only. Governments raised more than US\$44 billion in carbon pricing revenue during 2018. There are determined price trajectories that can deliver on the Paris Agreement which increase each decade on a non-discounted basis. IPCC trajectories show the marginal cost of reducing GHG emissions. Other sources provide carbon price ranges which consider ambitious climate policy. When setting carbon prices, local, ethical and distribution factors must be considered.

Internal carbon pricing is increasing in popularity within the private sector. 1300 companies, including 100 Fortune 500 companies, disclosed the current use or intent to use internal carbon pricing. Traditionally, internal carbon pricing drove investment planning for mandatory carbon policies. Internal carbon pricing is transitioning towards informing long-term climate investment and risk strategies. Financial policy frameworks are reassessed to use carbon pricing to support sustainable growth. Internal carbon pricing is informing the construction of market indexes e.g. S&P Carbon Price Adjusted Index to inform market climate risk. Internal carbon pricing will continue to catalyse both the investment in sustainable technology and the divestment in fossil fuels. Although new carbon pricing initiatives are emerging with an increased emphasis on global collaboration, it is clear the global community is far from reaching the objectives set by the Paris Agreement. Only 5% of carbon pricing initiatives are currently priced to meet the Paris Agreement's temperature goals [31].

2.4 Opportunities and Challenges

2.4.1 Market-Based Approaches

Both carbon pricing initiatives are market-based and can be less obtrusive for industry than regulatory controls. The prevalence of emissions across global industries sees a market-based approach more equitable, evenly distributed across industries, and quicker to implement.

Arguably, a carbon tax is the most straightforward approach. A carbon tax could be imposed on all emitting inputs and outputs of production. The marginal cost of carbon would be covered by a tax rate. Tax credits from sequestration or other initiatives could be used to invest directly into alternative technologies. Tax rates could be adjusted to illicit the desired market response. A carbon tax would align with existing systems used to facilitate inland revenue functions in the desired jurisdiction.

The effectiveness of cap and trade systems is undermined by grandfathering credits, using offsets in lieu of meaningful emission reduction targets, and the challenges of setting baselines. If credits are grandfathered, there are no tax revenues to invest in alternative energy or technology. In a cap and trade system, there is uncertainty around the price of those reductions, and the subsequent effect these credits have on lowering emissions. Cap and trade initiatives provide benefit certainty as you can ascertain the environmental benefits from an imposed ceiling on emissions. However, there is no cost certainty which

is provided through a carbon tax.

2.4.2 Benefits and Costs to a Carbon Tax

A carbon tax is simpler as is set at a cost per tCO_2e produced. In contracts, a cap and trade is more complex due to negotiating baselines, grandfathering/auctioning processes, monitoring systems, international trading guidelines, and cost uncertainty prevention. A carbon tax more easily generates revenue to invest in reducing greenhouse gas emissions and supporting businesses adopting more sustainable processes. A cap and trade system generates revenues using credit auctioning but is less effective than a carbon tax if grandfathering credits exists. A carbon tax ensures cost certainty as the exact cost of emissions is quantified. In addition, there is a clear message to polluters with a carbon tax. Under a cap and trade system, there is more uncertainty around the impact polluters have as they can merely purchase more credits to increase emissions. In light of these positives, there are also negatives to a carbon tax. There is significant political opposition in proposing a new form of tax. The benefits of a carbon tax will need to be clearly communicated to all stakeholders. The benefits of a carbon tax are uncertain in how they enact reductions in emissions. Tax exemptions may reduce the effectiveness of the carbon tax if granted inequitably. For example, if a super major was given a significant tax exemption on the basis of an existing political relationship. A carbon tax is also difficult to co-ordinate with other participants [9]. In addition to these disadvantages, there are misconceptions leading to the opposition of carbon taxes. These misconceptions include, but are not limited to, the following: Taxes reducing welfare, and increasing unemployment, from lower levels of consumption and production. Taxes perceived to put incumbents' business models at risk. Subsequently, these incumbents lobby heavily prevent change. Taxes increasing the prices of goods and services consumers consume, creating opposing public opinions [49].

Addressing the disadvantages and misconceptions of a carbon tax would help implement carbon taxes across geographies to reduce greenhouse gas emissions.

2.4.3 Decarbonization, Investment, and Technology

The revenue generated from carbon taxes would need to be reinvested to benefit regional, national, and global communities. One opportunity is investing in sustainable technologies. Fortunately, this type of investment is increasing. Between 2010 and 2019, renewable technology drew $2.6 trillion in investment. In 2019, $282 billion of renewable capacity was financed worldwide. Wind technologies (onshore and offshore) and solar were financed $138 billion and $131 billion respectively. This financing success was attributable to falling costs and maturing technologies. Renewable technologies are now profitable. Costs associated with solar and wind technologies have fallen 85% and 49% respectively in the last 10 years [30]. In Australia, New Zealand, Canada, Europe, Japan, and the United States, sustainable investments reached assets of $30.7 trillion in early 2018, one-third of total investment. By 2025, renewables will be competitive with natural gas. New technology is emerging in the Oil and Gas Industry to decarbonise the industry. These technologies include but are not limited to: renewable power sources, electrification, vapor recovery units, carbon capture, carbon storage, and green hydrogen [13]. Carbon tax revenues could be used in the following ways: investment into sustainable technologies, replacing existing infrastructure to support sustainable technology

growth, reinvestment into participating businesses, or addressing the adverse affects of climate change. It is important to communicate the benefit of sustainable investment to all relevant stakeholders.

2.5 Energy Modeling

It is important to understand how energy demand and supply is modeled before discussing emission levels, carbon taxes, and reinvestment opportunities. Unsurprisingly, energy modeling is complex.

2.5.1 Global and Domestic Energy Scenarios

Different scenarios define modeling processes. The World Energy Council devised three energy transition scenarios describing plausible pathways for the global energy transition to follow. The scenarios look forward to 2060. There is an inflection point in 2040 to assess the success of the strategies underpinning the scenarios. The council leveraged expert member communities and annual surveys to devise these strategies. The three global scenarios tell different narratives relating to the progression of global primary energy demand, electrification,mobility improvements, energy efficiency, infrastructure innovation, investment, new technologies, political action, and Paris Agreement alignment. There are three global scenarios: Modern Jazz, Unfinished Symphony, and Hard Rock. Modern Jazz is a market driven scenario. Unfinished Symphony is a highly collaborative, policy driven scenario. Hard Rock is a minimally collaborative, internal policy driven scenario [47].

Countries use the global scenarios to inform their regional and national scenarios. New Zealand has followed this methodology through devising the Tui and Kea scenarios. Tui follows the narrative of a global community effort. New Zealand does not generally have a common view on what is important. Subsequently, the country adopts a wait and see approach with some protection provided to local businesses. New Zealand will focus firstly on economic prosperity and individually wellbeing by leveraging off comparative advantages. This is purely a commercial response. Kea forecasts the New Zealand economy cannot remain internationally competitive under current emission intensity trends. The country will take leadership in lowering emissions, choosing to undergo an early and aggressive economic transformation. New Zealand will act before the global economy at the expense of its own.

Both scenarios are underpinned by 19 critical uncertainties. These vary from external sources such as global stability, international fuel markets, urban sustainability, energy affordability, and the allocation of natural resources. Both scenarios consider different service demands e.g. Number of km travelled, population, GDP ($), forecast carbon prices ($/t$CO_2$e), carbon emissions (Mt/p.a), required investment ($), and commodity prices [20], [21]. These scenarios were feed into The Integrated MARKEL-EFORM System (TIMES) model to forecast their impact.

2.5.2 The Integrated MARKEL-EFORM System (TIMES) Model

The Integrated MARKEL-EFORM System (TIMES) model generator was developed as a component of the International Energy Authority's (IEA) Energy Technology Systems

Analysis Program (ETSAP). The IEA-ETSAP uses long term energy scenarios to conduct comprehensive environment and energy analyses [38]. Energy is modeled through combining complementary technical engineering and economic approaches. TIMES uses a technology rich bottom up architecture, using linear programming to produce least cost energy systems for medium to long term time horizons [39]. The TIMES model encompasses each step in the value chain to produce and supply energy to meet the demand for energy services by consumers. These include: primary resources, transformation processes, transportation methods and conversion processes. The supply side considers production methods and net exports. Energy is carried through to residential, commercial, agricultural, transport, and industrial sectors. The relationships between producers and consumers underpin the TIMES model. The nature of these relationships are mathematical, economic, and technological. TIMES considers technologies, commodities, flows, and scenarios constrained by policy decisions. Services demanded by consumers are the main inputs for the model. The model will make investment, supply, trade, and operating decisions when considering the inputs, constraints, and scenarios in the model. Consumer and producer surpluses are maximized from a mathematical perspective. The main outputs are energy system configurations at the lowest cost that meet end users' service demands. These include energy prices, flows, emission quantities, capacities, and costs [39], [25].

The TIMES model is useful for using energy scenarios to devise strategies for implementing carbon taxes.

2.5.3 General Algebraic Modeling System (GAMS)

GAMS is a high-level modeling system for optimization and mathematical programming. The system is suitable for large scale, complex modeling applications. The system gives you access to a diverse portfolio of solvers to solve linear, non-linear, and mixed integer optimization problems. GAMS is the ideal tool for modeling energy and carbon tax scenarios using TIMES Models [27].

2.5.4 Open Source Energy Modeling System (OseMOSYS)

The OseMOSYS project is an open source project to improve the accessibility of modeling energy systems. The modeling methodology is scalable from city to continental granularities. The approach is designed to require no upfront investment, little time commitment and provides a fast learning curve. This methodology is suitable for developer, modelers, academics and policy makers. The model structure includes pre-defined sets, parameters, objective function and constraints relating to energy systems. Python, GNU MathProg and GAMS versions are available for download in remote repositories on Github [43].

3 Project Scope and Research Objectives

The literature led to some interesting outcomes. Carbon taxes are an effective way to help meet the goals set by the Paris Agreement, reduce global emissions, and reinvest in both sustainable and decarbonizing technologies. However, there are significant obstacles stopping the implementation of carbon taxes. These obstacles include political opposition, effectiveness in reducing emissions, co-ordination with other local, regional, national and international carbon tax efforts, and the distribution of tax exemptions. Currently, the distribution and implementation of carbon pricing initiatives is inequitable as major polluters aren't enforcing these initiatives e.g the majority of the US and India. Producers who operate globally may pivot to produce and sell GHG intensive products elsewhere in the world if taxes are not correctly set and spread amongst geographies. In this context, geographies are defined as geographical areas where energy consumption and emission production occurs. Earth is the entire set of geographies. This set can be partitioned into continental, national, provincial, or regional geographies (subsets). Any carbon pricing initiative must consider the specific combination of energy factors for the associated geography. The aggregate outcome from all geographies should meet the objectives set by the Paris Agreement. The revenues generated should be distributed back to participants with commercial benefits communicated through key financial metrics relevant to that geography or business. Carbon pricing initiatives constrain energy systems. It is important to consider how carbon pricing initiatives affect the energy system in driving sustainable energy investment and policy. The overall research objective is:

Develop a Global Carbon Pricing Optimisation Model.

The model will be developed by the following process:

1. Understand the existing approach to forecasting services demand and develop a standardised forecasting process which aligns with emission targets.
2. Partition the global set of geographies into subsets with varying levels of granularity e.g. a continental subset with seven geographies (continents), a national subset with 195 geographies (sovereign states according to the United Nations) etc.
3. Forecast services demand for each geography in each subset.
4. Adapt an existing TIMES or OseMOSYS model to input standardised energy system parameters.
5. Run simulations using the adapted model to form distributions on model outputs. These distributions are to inform the negotiation of carbon pricing initiatives.
6. Develop and model investment strategies from subsequent key outputs including carbon prices.
7. Develop a model interface (UI and UX) to communicate optimal global carbon prices, performance metrics, and the carbon tax benefits to support co-ordinated efforts to negotiate and implement carbon pricing initiatives.

In conclusion, this project will attempt to address issues surrounding climate change. GOCPI will enable any user to design and model their own energy system to inform investment and policy decisions. The intention is to empower users to influence energy investment and policy decisions made by public and private parties.

4 Methodology and Implementation

4.1 Project organisation

GOCPI adopted Data Science best practice, as described by Wilson et al [54]. Although these practices are mostly reserved for data science projects, their principles are suitable for product development and version control. All data and results were saved regularly and reproducible. The retention of data in all forms received high levels of attention. Project files were synched continuously to Google Drive [28]. Git [37] was used to manage version control for GOCPI's source code, data, documentation and results. Git stores a complete history of versions using Git hashes. These hashes are strings unique to each state of the publicly available GOCPI repository[1]. Git hashes enabled the discretisation of GOCPI's development over time, enabling the accessibility and recollection of all previous states given a unique git hash. This functionality enabled reproducibility, error correction and the ability to revert to previous models.

4.1.1 Version Control

Git, hosted by GitHub, provided a comprehensive set of version control technologies. These technologies provided a range of benefits. Firstly, Git is excellent at providing and supporting collaborative functionalities. The master version of a project is accessible for all who have access to the repository. Each contributor could create custom copies of branches through pull requests on the master branch. Contributors could commit changes to custom branches and push these changes to the master branch through push requests. The product manager could review these push requests, approving suitable requests to integrate changes to the master branch. Collaborative efforts were possible with commit messages describing the contributions from each contributor. This project had one contributor. Git ensured the histories of code, work and authors are stored. The descriptive nature of the commit log ensured an accurate journal is kept.

4.1.2 Folder Structure

GOCPI maintained the file folder structure recommended in Wilson et al [54]. Project organisation was paramount as the modelling of energy systems involves integrating a range of optimisation models, data files and documents. Wilson et al's recommendations were appropriate as data science projects require similar organisational rigor. Subsequently, file management and structure was most efficient and comprehensive. **GOCPI** is the root directory of this project and contains several sub directories: bin, data, doc, src and results. The **bin** sub directory contained external scripts and compiled programmes related to the GOCPI project. The **data** sub directory contained all raw data associated with the project. This data included energy statistics, energy balance datasets, partitioned geographies, standardised optimisation models and TIMES modelling frameworks. The **doc** sub directory stored GOCPI's user guides, academic resources, research reports and project deliverables. The **results** sub directory contained the output from optimisation simulations and processed data to display on dashboards and websites to inform investment and policy decisions. The **src** sub directory stores the source code for preparing raw data, partitioning sets of geographies with varying granularities and the

[1] https://github.com/CMCD1996/GOCPI

GOCPI python package available to download using PyPI[2] and install using pip[3]. All files were continuously backed up using Google Drive.

4.1.3 Python

Python 3.7 was the primary coding language for the GOCPI project. GOCPI's objective is to enable any user to design and model their own energy system to inform investment and policy decisions. The intention is to empower users to discuss energy investment and policy decisions made by public and private parties. Additionally, GOCPI intends to reduce misinformation regarding energy policies and help assess the feasibility of meeting the International Energy Agency's Sustainable Development Scenario [3]. Python is omnipresent, widespread in software development. Python's language design makes the language highly productive and simple to use. Python can hand off computationally straining tasks to C/C++ and has first-class integration capabilities with these two languages. The language also has a very active and supportive community [40]. In addition, Python is the most popular coding language on the planet defined by the PYPL PopularitY of Programming Language Index. As at August 2020, Python had 31.59% of all language tutorial search instances on Google [46]. Python has many useful packages for creating the GOCPI package such as NumPy, Scikit-learn, os, csv and Pandas. Programming is quick due to Python's dynamic nature. The language is also open-source with no cost. Subsequently, Python was the best language to ensure the GOCPI model is accessible for many users to use and extend.

4.1.4 Package Management

The Anaconda package management platform for Python [6] was the chosen coding environment. Anaconda is a well defined, free platform with known versions of python packages such as matplotlib, numpy and pip. The use of this environment ensured both reproducibility and consistency across infrastructure. Although this project required no collaboration, the use of Anaconda will inform future developers on how to manage collaborative processes, especially for packages which are less well-maintained. Anaconda allows you to create custom environments which was necessary for creating scalable linear optimization problems to express energy systems. Pip is Python's default package manager and is included in the Anaconda package. Pip was used to install and update packages for python not available on Anaconda such as twine and the custom GOCPI package developed for this project.

4.1.5 Excel

It is important users are comfortable with using the GOCPI model. Energy modelling can be quite complex. The modelling process must be transparent to inform users how to build their own models. Excel is ubiquitous across academic and professional communities. Excel's omnipotence makes the software well-suited for describing the components of the GNU Mathprog energy system model. The **GOCPI OseMOSYS Structure.xlsx** file describes the sets, paramaters, constraints and objective function of a scalable energy system model. The User may toggle statement sets, parameters and constraints to adjust the complexity of the model. The model file was imported to a text file. However, data

[2]https://pypi.org/
[3]https://pypi.org/project/pip/

related to these energy systems was stored using Python dictionaries, lists and NumPy arrays. This Python formulation was later transcribed to a text file. Excel is best for two dimensional variables or data stored in Codd-Boyce relational databases [7]. The majority of parameters in energy systems were three or more dimensions. Therefore, Excel was not suitable to store these parameters. Python dictionaries, lists and NumPy arrays were preferred alternatives.

4.1.6 IBM ILOG CPLEX Optimization

The OseMOSYS methodology (see 4.5.1) translates energy systems into linear programming problems. A solver was required to optimise these user-defined energy systems. The IBM ILOG Optimization Studio [36], more commonly known as CPLEX, was chosen to be this solver. CPLEX solves very large linear programming problems using the Barrier Interior-point method [45] or primal/dual variants of the Simplex Method [15]. GOCPI's user-defined energy systems could be scaled up to model very large systems, creating large linear programming problems.

The IBM ILOG CPLEX Optimization Studio has an interface with the Python language based on a C programming interface. Subsequently, Python APIs were available to run the CPLEX solver when installed either locally or on a cloud service. The python packages are **cplex** and **docplex**. The cplex package contains classes for accessing CPLEX for the Python programming language. The Cplex class is the most important class in this package as provides methods for creating, modifying, querying, or solving optimisation problems. Docplex also enables the formulation of new linear programmes where one creates the model, defines the decision variables, sets the constraints and expresses the objective function. The user uses docplex to solve the linear programme on a local solver. Alternatively, the model can be solved on a private cloud using Decision Optimisation on Cloud service through the provision of a service url and personal API key. The CPLEX Python APIs were most attractive as provided the user with a powerful commercial solver in an accessible format.

There is a caveat to the use of the CPLEX solver. The IBM ILOG CPLEX Optimization Studio is commercial by nature and requires a license to use. Fortunately IBM have the IBM Academic Initiative [34], granting students access to commercial software for free. This commercial nature creates accessibility issues for users who are not enrolled at an academic institution or can afford to pay for the software. Accessibility issues caused by the need for commercial solvers must be addressed to enable the distribution of the GOCPI product.

4.1.7 IBM Watson Machine Learning Service

The IBM CPLEX Optimisation Cplex python API is suitable for smaller models that can be solved locally. As the model increases in complexity, the docplex Python API did enable the ability to solve larger linear programmes. Unfortunately, IBM phased out the docplex Python API by incorporating the Decision Optimisation on Cloud services into the IBM Watson Machine Learning cloud services [35]. This change occurred during September 2020. This service uses IBM Cloud to access assets through credentials, create model deployments in IBM's servers and execute jobs to solve models. The model deployments must be Python-based models with jobs specifying a payloads containing input data and output formats.

4.1.8 PyPI

PyPI[1] is the Python Package Index, a repository of software for the python programming language. This repository helps you find and install software developed by the Python community who have decided to share their work. The GOCPI package is distributed from this platform to enable as many as possible the ability to model their own energy systems to inform and question energy policy and investment. Enter command: **pip install GOCPI** in the terminal to install the package using pip package management software.

4.1.9 Code Style

The GOCPI project was developed as the GOCPI package. All development code is organised within this package. The PEP8 style for Python Code was the formatting style for development code [32]. All code was formatted with **yapf**, a formatter maintained by Google to format Python files. Standardised formatting is important as makes the code easy to read, helps optimise the code and promotes consistency. Docstrings and commenting were most important in documentation. A docstring is a Python inline comment. Each class and function has an unique docstring, a one sentence description of the function, inputs with data types and types of outputs. The Google style docstring was most appropriate because of it's readability, ease to write and consistency with the Google Style Guide. Additionally, automated documentation generators (**pdoc3, Sphinx** etc.) can parse this format to create documentation. This self-consistent code style facilitated best practise maintenance and enabled reproducibility.

4.1.10 Infrastructure

GOCPI creates scalable energy system optimisation models with complexity size dependent. Computations either took place locally on a 128 GB, four core Apple MacBook Pro or remotely using a cloud service.

4.2 Documentation

The GOCPI project is well documented to keep an accurate record of key design decisions. The commit history described in 4.1.1 was the most important form of document. Other explicit documentation methods were applied to supplement this commit history. These methods, in addition to in-code documentation, include project updates and meeting minutes nested within a project logbook.

4.2.1 Project updates

Project updates were recorded as itemized lists. Each item is a brief description of the work completed during that day, week or month. Items include, but are not limited to, completing GOCPI submodules, researching energy system statistics, building websites or writing sections of this research report. These updates were pivotal to exploring new options, monitoring progress and making decisions to drive forward development. For example, the decision to adopt the OseMOSYS methodology in favour of the TIMES

[1] https://pypi.org/

modelling methodology. Project updates were transcribed to the project logbook held in this project's research compendium.

4.2.2 Meeting minutes

Project meetings took place for half an hour once a week. These meetings included discussions on energy markets, modelling methodologies, project progress and key design decisions. The minutes from these meetings accompanies project updates in the project logbook nested within the research compendium.

4.3 Geographical Partitioning

The first stage of the GOCPI project was to come up with an exhaustive list of geographies. The user has the choice to either model the energy systems of these geographies individually or model a network of geographies connected by trade relationships. Geographies are grouped into three distinct sets. There is a continental set with six continents: Africa, North America, South America, Oceania, Asia and Africa. Antarctica was excluded from this set. The country set includes 194 United Nations member states. The elements in the city set number approximately 13800 unique cities from the 194 member states. The **GOCPI Geographies.gyp** script processes these unique sets to create a csv file detailing an aggregate geography set. This file contains 13800 geography combinations. Each combination includes a city's name, population, country and continent e.g. VANCOUVER,CANADA,2313328.0,NORTH AMERICA. A user may use this information to design energy systems for cities with similar populations, countries and/or continents. The user may also create combinations using cities or countries. The user may also build energy systems on a continent level. After the formation of an exhaustive list of geographies, the exploration and adaptation of energy system modelling methods began.

4.4 Energy Modelling: TIMES

The World Energy Council devised three energy scenarios to inform energy policy and investment. These are Hard Rock, Unfinished Symphony and Modern Jazz [47]. New Zealand devised two energy scenarios to inform national energy policy and investment. These are Tui and Kea [21]. TIMES modelling underpins the construction of these scenarios. Subsequently, I began the standardisation of a TIMES implementation to develop a method for building scalable energy systems.

4.4.1 TIMES: The Integrated Markel-Eform System

An implementation of TIMES is divided into a five stages as displayed in Figure 1:

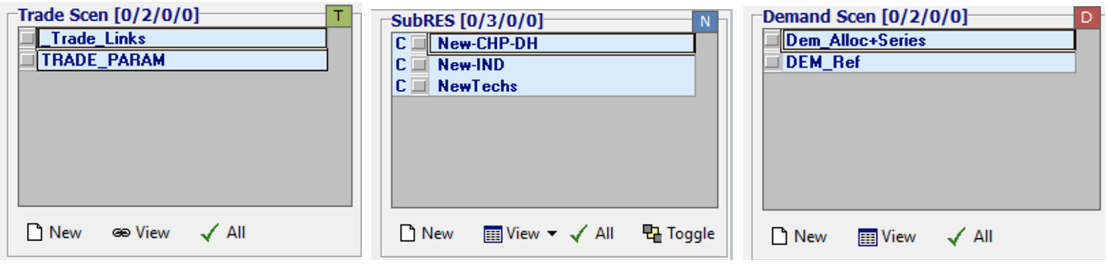
Figure 1: TIMES Implementation Process

The VErsatile data analyst (VEDA) [24] is a software package required for TIMES modelling. VEDA is an interface to design energy systems from Excel-based inputs and create the TIMES data and model text files for optimisation using GAMS. The installation of VEDA FE and VEDA BE required a windows operating system. A virtual private network (VPN) and Microsoft Remote Desktop was used to access a windows operating system via a virtual machine. GAMS Studio was installed on both the local Macintosh operating system and remote Windows operating system to perform the optimisation formulated in GAMS. VEDA is proprietary and requires a commercial license to use. Fortunately, there is a 60 day trial version available for download and use. After installing VEDA, GAMS and Microsoft Excel, I began following a TIMES user manual written by the IEA to build a standardised TIMES model. The user manual started with a very basic energy system, incrementing on the energy model using twelve demo models. I downloaded these twelve demo models to edit their associated Excel files.

The first stage was to adapt the existing Excel files so the user could design their own reference energy systems and sets of energy scenarios. Data and assumptions for reference energy systems and scenarios are primarily recorded in several tables across separate excel sheets and files. I made copies of these excel files and began writing functions to change the data stored in these sheets for a user to create new energy systems.

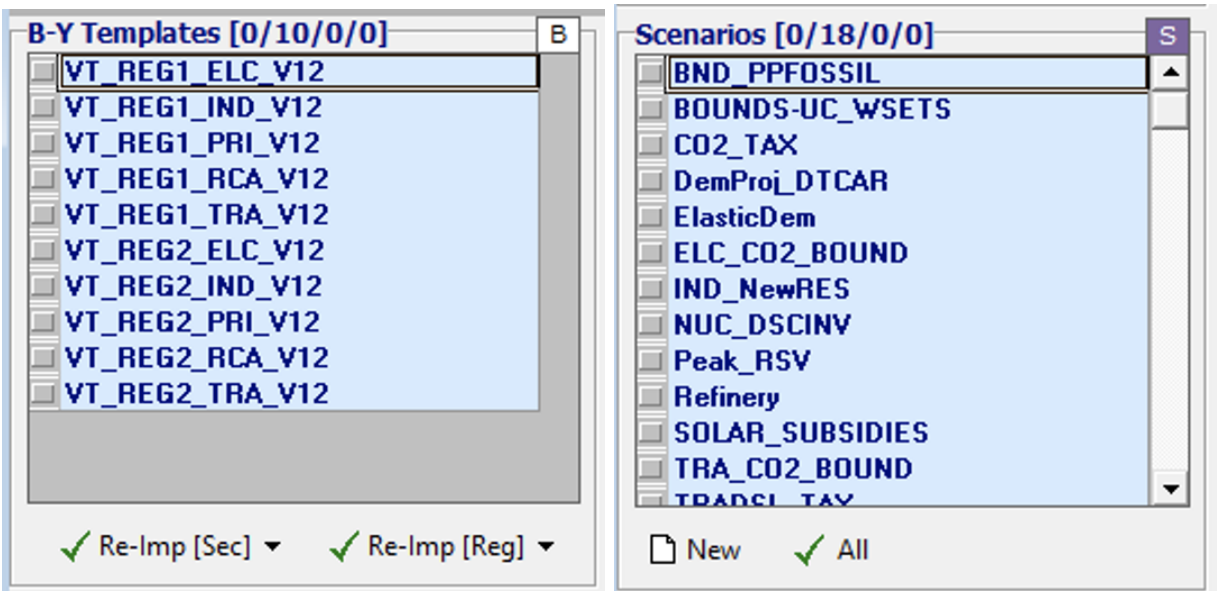
The second stage was to design energy systems using the VEDA interface. Figures 4 and 3 show the VEDA interface. I used the VEDA interface to combine the Excel files I adapted from the VEDA demo models. Initially, I built the base year scenario as shown in figure 4a. The inclusion of different energy scenarios (figure 4b) led to a more robust energy system. Examples include introducing the price elasticity of demand, solar technology subsidies and carbon dioxide taxes. New resources, technologies, trade relationships and demand profiles were added as shown in subfigures 3c, 3b and 3a. VEDA informs the user if the energy system is consistent with all data and assumptions correctly defined by the key shown in figure 2.

Figure 2: VEDA Energy System Consistency Key



(a) Trade Relationships (b) New Resources (c) Demand Proliles

Figure 3: VEDA Technology, Demand and Trade Scenarios



(a) Energy System Base Year (b) Energy Scenarios

Figure 4: VEDA Energy System Base Year and Scenarios

After designing the energy system using the VEDA interface, a case manager was used to generate a TIMES text file and run the optimisation. The model file is written in GAMS and integrated into the VEDA interface. The case manager allows you to select different scenarios and produce new model files.

The third stage was to execute the GAMS optimisation and produce outputs to be fed into VEDA BE. The forth stage is for VEDA BE to convert the GAMS output into Excel-based data files. In the fifth stage, data visualisation would enable the user to take the Excel outputs and make informed decisions related to energy policy and investment.

Unfortunately, I faced severe limitations at stage 3 of the TIMES implementation. Several factors made the TIMES implementation difficult:

1. The remote nature of the project made software integration between Flex IT, local hardware and remote servers difficult.
2. VEDA and GAMS Studio's commercial nature creates accessibility and equity issues for those who don't have the resources or connections to access the software. This contradicts the project's purpose.
3. The complexity of TIMES inhibits reproducibility and creates technical barriers for users not too familiar with energy modelling.

These reasons led to the abandonment of the TIMES implementation in favour of an alternative method. An alternative methodology would trade the complexity and sophistication of TIMES modelling for usability and accessibility. After consultation with my project supervisor and one of New Zealand's Energy Council modelers, the OseMOSYS methodology was recommended. Subsequently, I began developing a scalable open source energy system using the OseMOSYS methodology.

4.5 Energy Modelling: OseMOSYS

4.5.1 Overview

The OseMOSYS methodology was the next best alternative to TIMES for modelling large scale energy systems. TIMES can model useful complexities such as stochastic processes where OseMOSYS cannot. However, OseMOSYS proved to be the better of the two methods as more closely aligned with the overall scope and purpose of the project. The aforementioned tradeoffs are justified given the usability and availability of OseMOSYS.

There is an online community who make contributions to the development of OseMOSYS. Their mission is to make energy modelling more transparent and accessible. This online community is developing three versions of OseMOSYS accompanied with exemplar energy systems. The resources created by this online community can be found on Github [53]. A version of OseMOSYS was available in GAMS, GNU Mathprog or Python (Pyomo). Each version was at different stages of development. The GNU Mathprog version was chosen as was most complete of the three versions and included two energy system examples. The model and data files were downloaded as text files and stored in the project's working directory.

4.5.2 Implementation

It is not practical to explain the entire model given the scale of OseMOSYS. Therefore, the provision of a high level outline follows. Please refer to section 9.1 to view the complete model. The OseMOSYS model contains 11 sets, 52 parameters, 67 variables, 1 objective function and 94 constraints. A summary of the model's components follow:

Component	Factors
Sets	Year, Technology, Timeslice, Fuel, Emission, Mode of Operation, Region, Season, Day Type, Daily Time Bracket, Storage
Parameters	Trade, Activities, Demand Profiles, Costs, Storage Rates, Conversion Rates, Emission Limits etc.
Variables	Demand, Storage Rates, Production, Emission Penalties, Operating Costs, Technology Use etc.
Constraints	Energy Balances, Production, Charging Discharging, Storage Costs, Capacities, Investments Renewable Energy Requirements, Emissions Accounting etc.

Table 1: OseMOSYS Components

The sets are the primary components to the model. It is important they are explained to provide greater context.

1. **Year**: Represents the timeframe of the model containing all years to be considered.
2. **Technology**: Represents any element in the energy system that changes a commodity from one form to another, uses it or supplies it. The modeler is free to

interpret the technology. It may be a single technology e.g. a power plant or an aggregation of many technologies e.g. millions of solar panels.

3. **Timeslice**: Represents the times resolution of the model through time splits.
4. **Fuel**: Represents any energy vector, proxies or energy service flowing through technologies. Depending on the required analysis, fuels may be individual groups, aggregate groups or artificially separated.
5. **Emission**: Represents emissions potentially derived from operating technologies.
6. **Mode of Operation**: Represents the number of modes a technology may have. If a technology can have various input and output fuels, and can choose any linear combination of the inputs and outputs, this may be expressed as a mode of operation. E.g. a CHP plant may produce heat in one mode and electricity in another.
7. **Region**: Represents the regions to model as explained in section 4.3. Supply-demand energy balances for all energy vectors, including trade with other regions, are ensured.
8. **Season**: Represents the number of seasons accounted for e.g. Summer, Autumn , Winter and Spring. These are expressed as successive integers e.g. Summer = 1, Autumn = 2.
9. **Day Type**: Represents the number of days types accounted for e.g. Weekend, Weekday. These are expressed as successive integers e.g. Weekend = 1, Weekday = 2.
10. **Daily Time Bracket**: Represents the number of splits for a day e.g. night, morning, afternoon, evening. These are expressed as successive integers e.g. night = 1, morning = 2, afternoon = 3, evening = 4.
11. **Storage**: Represents storage facilities/capabilities used in the model.

$$\textbf{Min:} \sum_{r} \sum_{y} \text{TotalDiscountedCost}_{r,y}$$

Figure 5: OseMOSYS Objective Function

Combinations of sets are used to define the parameters and variables. These parameters and variables are used to set constraints. The objective function uses the constraints to find the optimal variables. The objective function is the minimisation of total discounted cost in each region across all years specified in the forecast horizon (Figure 5). Total discounted cost is the sum of the total discounted costs by technology over all the technologies.

4.5.3 Integration

An lp-format file was required to solve OseMOSYS implementations with a solver. These files are written in GNU Mathprog, a language intended for describing linear mathematical programming models [50]. GNU Mathprog is a subset of the Algebraic Mathematical Programming Language (AMPL) [5] and consists of user-defined sets of statements and data blocks. The custom anaconda environment (**osemosys**) was created to facilitate

this model formulation. This environment contains the libcxx package which provides a standard library of C++ functionalities. The GNU Linear Programming Kit (GLPK) sits within this library. GLPK uses GNU Mathprog to solve large scale linear programming (LP), mixed integer programming (MIP), and other related problems. GLPK is called using the **glpsol** command in the osemosys conda environment:

glpsol -m model.txt -d data.txt –wlp GOCPI.lp

The aforementioned glpsol command converts a data and model text file to an lp-file. The command also solves the problem if the energy system is a reasonable size. It is likely glpsol would not be able to solve larger scale energy system. Therefore, a commercial solver would be needed. The installation of the GLPK was tested using the Utopian example data and model files with the aforementioned glpsol function. The necessary lp-format file was generated for the users chosen solver. This was paramount before disembarking on developing a standardised energy systems development method to generate both text and data files from user-defined energy systems.

4.5.4 Storage

Energy modelling and OseMOSYS can be quite complex. It is important this complexity is conveyed to the user so they understand how to build their own energy systems. Subsequently, an Excel-based template was created to describe and store all model components. The **GOCPI_OseMOSYS_Structure.xlsx** contains all the sets, parameters, variables and constraints needed for an OseMOSYS model. The spreadsheet also includes descriptions of all model components and the ability to toggle their inclusion in the user defined model. The user may customise the structure of their energy model by making adjustments to the model's components in the GOCPI_OseMOSYS_Structure Excel sheet. Constraints may be added or removed and the objective function changed to consider other factors e.g. the minimise emissions penalties or maximise the uptake of renewable technologies. The base model came from the model text file, written in GNU Mathprog, stored in the OseMOSYS Github Repository. The template was adjusted to ensure constraint, variable and parameter indices were consistent with the demo data files.

4.6 Building Energy Systems

After the correct installation and initialisation of the OseMOSYS structure, it was appropriate to begin developing a methodology for building energy systems. The first step in this process was to understand the components and structure of the Utopia.txt example available from the OseMOSYS Github repository.

4.6.1 Utopia Example

The Utopia example models a 'Utopian' energy system. The system is developed by the OseMOSYS community to function as a baseline model and inform the construction of user-defined energy systems. The deconstruction of this system was paramount to understanding how to produce a standardised approach to building energy systems described in section 4.9.3. The Utopian example is a self sufficient energy system with no trade relationships. The system models twelve fuels processed in 21 technologies over a 21 year

forecast period. The model tracks two types of emissions, two modes of operation for some technologies and one storage technology. The utopian data file benchmarked the structure needed to generate compatible energy system data files.

4.6.2 NZ Example

The construction of a NZ energy system was undertaken after understanding the structure of the Utopian example. The purpose of the NZ energy system example was to demonstrate the practicality and usability of this model. The NZ energy system models the energy systems of New Zealand, Australia and a bi-directional trade relationship between the two nations. After disembarking on the development of a user-defined energy system, the limitations inhibiting the formation of a complete system were made apparent. These are briefly described as follows:

1. **Data Availability**: Publicly available and accurate data is hard to find when creating energy systems. This project was able to access publicly available from government agencies and global reporting entities. However, data is recorded differently depending on the region. This creates issues with building consistency between modeled regions. Many underlying assumptions must be made to build energy systems. These assumptions may change depending on the region. It is recommended there is access to consistent data source for all regions the user intends to model.
2. **Time Constraints**: The construction of an energy system is very time consuming. The NZ Energy system between Australia and New Zealand required the collection and scripting of a lot of data. This example was only between two regions. The model can support the maximum number of region combinations the IBM Watson Machine Learning Decision Optimisation service is capable of solving. The construction of a feasible and realistic energy system could be a Part IV project in itself.

These limitations led to the abandonment of developing a complete energy system. The NZ example is partially complete to show proof of concept. Section 6.2.2 explains next steps in finishing this example.

4.6.3 Energy System Concepts

The construction of energy systems requires a firm grasp on several concepts. These concepts are either related to general financial and energy theory, or specific to the OseMOSYS methodology. Each parameter in the model formulation relates to an important financial or energy concept in the energy system. The scope of this report prevents the inclusion of detailed descriptions for all integral components. Subsequently, the most important concepts are addressed in this section.

4.6.4 Reference Energy System

The **Reference Energy System (RES)** describes the network flows amongst the energy production, conversion and consumption of different fuels from different technologies. Essentially, it maps all relevant technologies to be involved in the energy system. Each region in the aggregate energy system has it's own energy system.

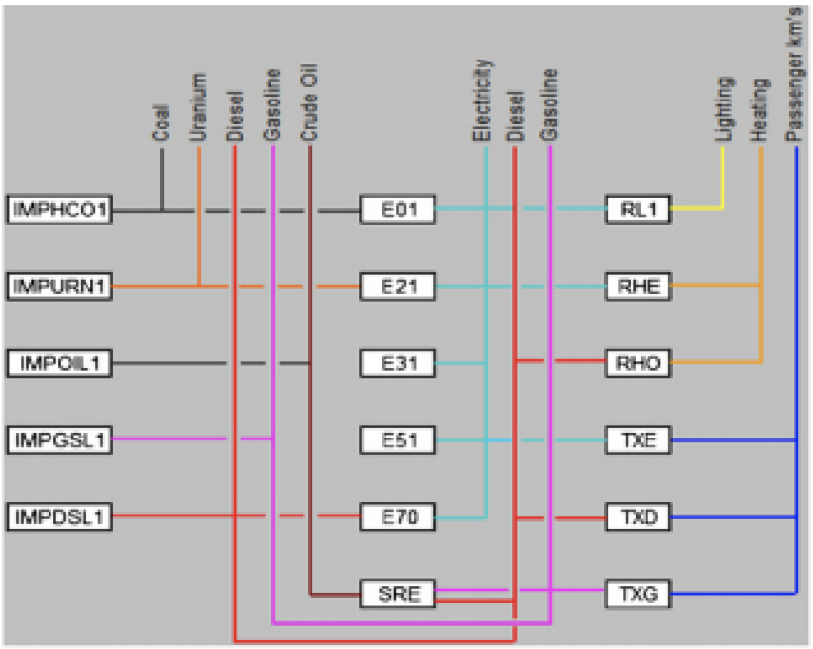


Figure 6: Utopian Reference Energy System

The Utopian energy example shown in figure 6 is the a very simple mapping of technologies. A realistic example is more complex. The NZ Reference Energy System was designed using energy balance data available from the Ministry of Business, Innovation and Employment (MBIE) ([16]). Table 2 describes the technologies mapped in the New Zealand RES. The fuels within the RES are described in table 3. The Australian RES was derived from the Department of Industry, Science, Energy and Resources (ISER) energy data set ([22]). The Australian dataset was adapted to match the NZ RES as RES between regions must follow the same format. Simplifying assumptions were required to ensure RES consistency. Subsequently, these assumptions create limitations in building large energy systems as it may be difficult to create consistent systems with the available data. Figure 7 shows the RES mapping of the NZ energy system for this project. In comparison to the Utopian RES, the mapping of technologies for the NZ RES is more complex.

Categories	Technologies
Production	Indigenous Production, Imports, Exports, Stock Change, International Transport
Conversion	Electricity Generation, Cogeneration, Fuel Production, Other Transformation, Losses and Own Use
Non Energy Uses	Non Energy
Consumption	Agriculture, Forestry and Logging, Fishing, Mining, Food Processing, Textiles, Wood Pulp Paper and Printing, Chemicals, Non Metallic Minerals, Basic Metals, Mechanical Electrical Equippment, Building and Construction, Unallocated, Commercial, Transport, Residential

Table 2: NZ Reference Energy System Technologies

Fuel Type	Fuels
Coal	Bituminous, Sub Bitumious, Lignite,
Oil	Crude Feedstocks NGL, LPG, Petrol, Diesel, Fuel Oil, Aviation Fuel and Kerosine, Oil Other
Natural Gas	Natural Gas
Renewables	Hydro, Geothermal, Solar, Wind, Liquid Biofuels, Biogas, Wood
Electricity	Electricity
Waste Heat	Waste Heat

Table 3: NZ Reference Energy System Fuels

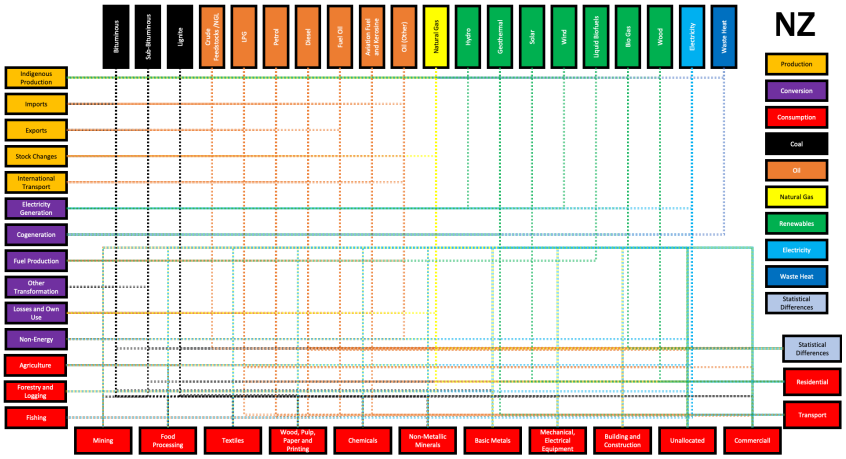


Figure 7: NZ Reference Energy System Mapping

4.6.5 Energy Balances

Energy balances are the base year energy production, conversion and consumption for a reference energy system. The energy balance is forecast for each interval in the energy system's time horizon. The NZ and Australia energy balances for the NZ example RES are derived from MBIE and ISER data respectively. Additionally, GOCPI includes a functionality to create energy balances from a large dataset published by International Energy Agency (IEA) ([2]). A constant average growth rate (CAGR) forecasting methodology was the baseline forecasting methodology for this project. This is a simple forecasting method. The energy balances between 2010 and 2018 were used to derive the CAGR using the formula as shown in figure 8.

$$\text{CAGR} = \frac{\text{Value}_{\text{Final}}}{\text{Value}_{\text{Initial}}}^{\frac{1}{(\text{Number of Years})}} - 1$$

Figure 8: CAGR Formula

The number of parameters to estimate made forecasting difficult. The use of CAGR to forecast values is not very realistic as you cannot assume values will grow or fall at a constant rate. Stochasticity is a large factor in energy production, conversion and consumption. OseMOSYS does not account for stochasticity. Therefore, randomness is not addressed. There are no degrees of freedom

for estimated parameters. For example, renewable energy production is stochastic. Weather patterns are increasing volatile so energy production will need to address this volatility for accuracy. It is recommended future developments include Machine Learning (ML) and Autoregressive Integrated Moving Average Models (ARIMA) forecasting methods. An ARIMA model is a times series forecasting method aiming to describe autocorrelations in the data to account for stochasticity ([33]). ML methods train models to make energy balance predictions using factors not obviously considered.

4.6.6 Trade Relationships

The trade relationships describe the exchange of energy-related resources between the modeled regions in the energy system.

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Figure 9: NZ/AUS Free Trade Matrix

These relationships are modeled using a four dimensional matrix. The matrix is indexed as follows: [region, region, fuel, year]. The first position corresponds to the regions where trade originates from. The second position corresponds to the regions which receive trade from the region indexed in the first position. The third position is the fuel being traded. The fourth position is the year the trade takes place. Values are binary to set trade between two regions for fuels and regions (1 = Trade, 0 = No Trade). The NZ energy system example assumes free trade exist between both NZ and Australia for all fuel types every year.

4.6.7 Discount Rate

The **discount rate** for a region is one of the most important concepts as informs for discounted costs and therefore the objective function. The function **def set_discount_rate** calculates the discount rate for each region. This function is most important as the discount rate underpins the total value of discounted cash flows and the objective function. The discount rate is calculated using the weighted average cost of capital (WACC) method. The WACC, cost of equity, cost of preference equity and pre-tax cost of debt are calculated as follows:

$$WACC = \frac{D}{E+D+K} \times r_d \times (1-t) + \frac{E}{D+E+K} \times r_e + \frac{K}{D+E+K} \times r_k \quad (1)$$

$$r_e = r_f + \beta \times (r_m - r_f) \quad (2)$$

$$r_d = \frac{i}{D} \quad (3)$$

$$r_k = \frac{Div}{MV_{ps}} \quad (4)$$

D = Book value of debt ($m)
r_f = Risk free rates (%)
K = Market value of preferred equity ($m)
r_d = Pre-tax cost of debt (%)
r_e = Cost of ordinary equity (%)
Div = Preference Dividends ($)
MV_{ps} = Market value of preference shares ($)
r_k = Cost of preferred equity (%)
t = Effective tax rate (%)
i = Cost of borrowings (%)
β = Market risk co-efficient (%)
r_m = Market return (%)
E = Market value of equity ($m)

The regions used in the energy system example are New Zealand and Australia. Both are public entities in this case. The model may be used for both private and public entities. It is assumed the financial performance and position for each region, reported in the the New Zealand and Australian Government's financial reports, are suitable for providing the financials necessary for calculating discount rates. Cost of borrowings (i), Equity (E), Debt (D) and Preference Equity (K) are reported in each region's annual financial reports (New Zealand [55] and Australia [29]). It is assumed both have no preference shareholders, do not distribute preference dividends (Div), list preference shares (MV_{ps}) or have preference equity (K). The effective tax rates (t) are each region's company tax rate. This example assumes New Zealand's market value of equity (E) is zero as not publically traded and Australia's negative net work implies market equity (E) is zero. Risk free rates (r_f) were derived from average 10 Year Government bond yields using data from each region's respective Reserve Bank (Reserve Bank of New Zealand [42] and Reserve Bank of Australia [8]). The monthly market returns for each region are annualised to calculate the market return (r_m). Each region has a market index which serves as a proxy for the market. New Zealand has the NZX 50 and Australia has the ASX 200. Historical returns related to these proxies were downloaded from yahoo finance [26]. Monthly returns are annualised as follows:

$$\text{Annualised Monthly Return} = ((1 + \frac{\text{Index}_{\text{Ending}} - \text{Index}_{\text{Beginning}}}{\text{Index}_{\text{Beginning}}})^{(\frac{12}{\text{Number of Months}})}) - 1 \tag{5}$$

The cost of ordinary equity is calculated using the Capital Assets Pricing Model (CAPM). Both New Zealand and Australia are assumed to have market risk co-efficients (β) of zero as their fiscal policy is market independent. This methodology is appropriate for both public and private entities.

The RES, energy balances, trade relationships and discount rate are a small subset of the energy systems concepts required to build a model. Please refer to the model sets and parameters in the **GOCPI_OseMOSYS_Structure** for the complete list of required energy system concepts required to build an energy system.

4.7 Optimisation

OseMOSYS is expressed as a linear programme requiring a commercial solver to solve the system. A few commercial solvers were considered.

4.7.1 IBM CPLEX

IBM has a suite of optimisation and analytics services accessible on IBM Cloud. There is an Academic Initiative to grant access to the suite of service's for free. The IBM ILOG CPLEX Optimisation Studio allows the user to build and solve problems locally. There are a number of Python APIs to access the IBM Cloud or the Optimisation Studio.

4.7.2 Gurobi

The Gurobi suite of optimisation products was considered as the default commercial solver. The Gurobi Optimizer can solve linear programmes and has the capability to use Python APIs to solve energy systems locally. Gurobi also offers the Gurobi Computer Server and Gurobi Cloud, suppported by Microsoft Azure and Amazon Web Services, to solve large optimisation problems remotely.

4.7.3 Optimisation Implementation

IBM's suite of services offered greater opportunities to develop the product further by setting up a pipeline to take advantage of IBM Cloud's full range of services.

Firstly, the IBM Academic Initiative grants the user access to a subset of the full sweet of IBM products. The user must represent, or be a member of, an academic institution to gain access to the initiative. The academic initiative allocates a IBMid linked to an academic email address. This IBMid enables access to IBM's services.

The CPLEX Optimisation Studio IDE was installed locally to enable the installation of both cplex and docplex Python APIs. The cplex API connects with the CPLEX Optimisation Studio IDE (10)to solve small scale optimisation problems locally. Alternatively, you can build your models in the IDE instead of using the Python API. The docplex API uses an api key to connect to a remote server to run the IBM Decision Optimisation service and solve larger optimisation problems in the cloud. Unfortunately, the IBM Decision Optimisation in Cloud service the docplex API connects to was discontinued in September 2020. The IBM Decision Optimisation in Cloud service was imported into the IBM Watson Machine Learning service in IBM Cloud. Subsequently, the project had to pivot to using the IBM Cloud services to solve larger energy systems.

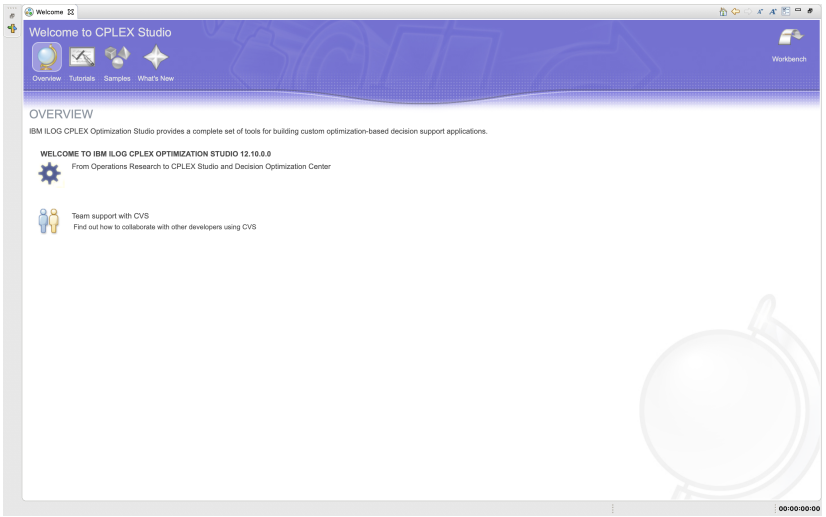


Figure 10: CPLEX Optimisation Studio IDE

An API key is required to use IBM Cloud services. This key grants access to your account and use of all licensed services. IBM Cloud has approximately 173 services to analyse

data and inform decisions. A subset of these services are displayed in figure 11.

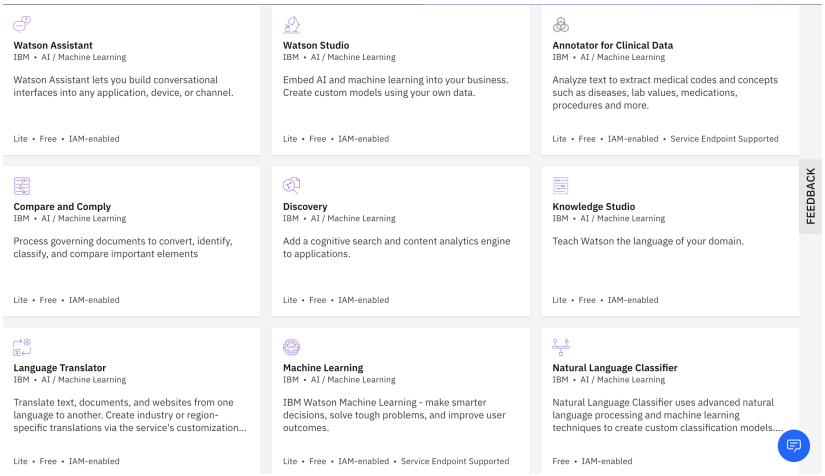


Figure 11: IBM Cloud Services

Services include AI/Machine Learning, Blockchain, Security, Database and Optimisation functionalities. The IBM Decision Optimisation service sits within the IBM Watson Machine Learning Service which is apart of IBM Watson Studio resource. A service or storage object sits within a resource. A new IBM Watson Studio resource was created to access the required services. After initialising the resource, the GOCPI project was created. This project contains two additional services and one deployment space. The services are the IBM Watson Machine Learning service to use the IBM Decision Optimisation service and a Cloud Storage Object (COS). The COS stores content related to the project. The deployment space is the location to evaluate, configure, deploy and monitor deployable assets. A deployment asset includes the input data and model formulation related to the project. Each component of a resource (service, storage object etc.) has a CRN and GUID tag. These identify the services when accessing them remotely using API's. Services are executed using the **ibm_watson_machine_learning** module and IBM Developer Tools VS Code extension.

The process to solve energy systems, using the IBM Decision Optimisation service, is as follows:

1. Create the IBM Watson Machine Learning, COS and Deployment Space on IBM Watson Studio.
2. Formulate the OseMOSYS model, in Python, for model deployment.
3. Create a model deployment to fill the deployment space (if one does not already exist).
4. Design a payload (input data and output structure).
5. Create a new job to solve the payload using the model deployment.
6. Push the job in the deployment space until the energy system either solved or deemed infeasible.

The docplex API enabled IBM Decision Optimisation to solve energy systems in a lp format. Therefore, the GNU Mathprog OseMOSYS formulation was acceptable using the docplex API. Unfortunately, the IBM Watson Machine Learning Decision Optimisation service required Python-based OseMOSYS model formulations. This limitation lead to the inability to solve large scale energy systems. A pipeline was created to solve Python-

based OseMOSYS formulations for subsequent iterations detailed in section 6.2. This pipeline is detailed in section 4.9.6.

4.8 Interface

It was important to make the tool accessible for those proficient in python. The model needed to easy way to access and distribute the GOCPI product. A website was the best tool to achieve this desired outcome. The website has one page dedicated to the project interface. The remaining pages relate to content out of project scope.

4.8.1 Jekyll, GitHub pages and Google Domain

Jekyll is a static site generator, simple in nature and written in Ruby. GitHub enables the free hosting of websites through GitHub pages. GitHub pages is powered by Jekyll, enabling the ability to build websites. The directory structure for the project website is shown in figure 12.

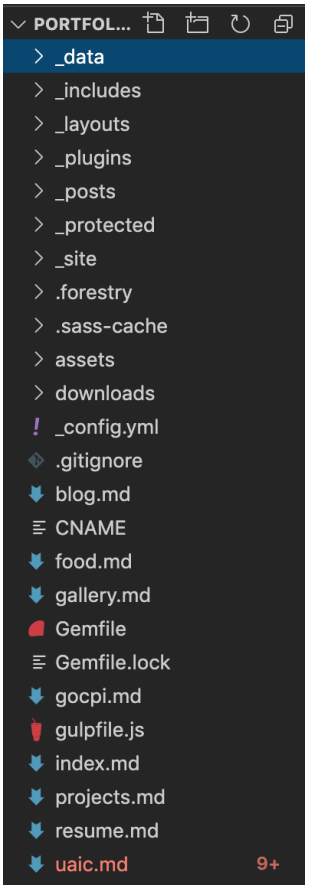


Figure 12: Website Directory

Jekyll is unique in terms of site generation. Ruby gems are used for package management. Ruby and bundler was installed on the local device to enable website development. Firstly, the GitHub pages functionality and a custom domain name were initialised for the GitHub repository. In the Gemfile shown in figure 12, ruby gem commands were added to control the website's theme, markdown parsing and GitHub pages adaptation. Content was transcribed in markdown files using text. These markdown files were converted into html scripts stored in the _site sub directory. An outline of the project was conveyed through an embedded video presentation and a hyperlinked powerpoint. Hyperlinked images take the user to this project report and the research compendium. At the base of the site, there are hyperlinks to download and install the software required to use the product. The images used for hyperlinking were stored in the assets sub directory. The hyperlinked reports are stored in the downloads sub directory. The structure of the html scripts generated from markdown files were stored in the _layouts sub directory. A custom css file to enable video embedding was added to a video embedding html script in the _includes folder to enable responsive sizing when switching between mobile and desktop displays. The minima theme, github pages and kramdown-parser-gfm gems were added to the Gemfile and implemented using the terminal command **bundle exec jekyll serve**. This command uses the bundler package and Gemfile to generate the html and css files required to build the website. The command also creates a local server to display the website. Committing and pushing changes to the GitHub repository generates updates for website content. Additionally, a google

domain name (https://connormcdowall.com) was purchased for the project. The chosen domain name was purchased as will be used for additional uses outside the scope of the project. Custom resource records (IP) were generated to create a new Domain Name System (DNS) record. This record translates the purchased domain to the newly set IP addresses in order to load the website on any internet browser.

4.9 Standardised Forecasting Methodology: GOCPI Package Development

4.9.1 PyPI and Directory Structure

The aforementioned distribution method PyPI (section 4.1.8) was most appropriate. The GOCPI project is listed as an active project. The package required a specific directory structure to publish new distributions. The **GOCPI** stores several files required for the package: A sub directory of the same name containing an initialisation file (**init.py**) and package modules. A distributions sub directory (**dist**) for new package versions. A Python file for setup information (**setup.py**) and a setup file for package configuration (**setup.cfg**). Lastly, there is a sub directory storing text files related to dependencies, package information and source links.

New distributions are stored in the **dist** displayed in figure 13. **Twine** is a package to upload new setup descriptions, configurations and code changes via the local terminal. PyPI enables the immediate release and availability of the GOCPI package. Users can continue to use the product as new versions are released.

The development of the GOCPI package used the following process:

1. Make additions, deletions or increments to package modules.
2. Update the initialisation file to reflect additions, deletions or increments.
3. Update package information in the **setup.py** file and increment package version number.
4. Create a new distribution (tar.gz file) using the following command: **python setup.py sdist**.
5. Upload the new distribution using the following command: **twine upload dist/GOCPI-X.tar.gz'** where X is the version number.
6. Download the new distribution using the following command: **pip install –upgrade GOCPI'**.

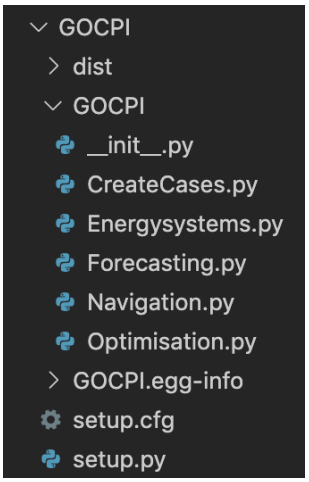


Figure 13: GOCPI File Structure

Over 80 distributions were released during this prototype's development. Five classes were developed to support the construction, forecasting and optimisation of user defined energy systems. These will be addressed in turn as contribute to the key functionalities of

the package. The docstrings related to all written functions may be found in the section 9.2.

4.9.2 Navigation Module

The navigation module is a class to help the user navigate their local directory. The most important function in this class is the **Find_File** function. The purpose of Find_File() is to enable the user to seek and find designated files in their modelling directory from a directory root. This root may be any node in the directory structure. The FindFile function is important for the other functions in this package.

4.9.3 EnergySystems Module

The EnergySystems module is a class to load, create model text files and create data text files from input user defined energy systems. There are three core modules to this class. **load_datacase()**, **create_model_file()** and **create_data_file()**. The EnergySystems class is initialized by taking the 11 sets, defined as lists, outlined in section 4.5.2 as inputs. The lengths of these sets are calculated using python's numpy module and used to initialise all 52 parameters related to an OseMOSYS energy model.

The load_datacase() function will load a user built energy system as outlined in section 4.6. The function simply replaces the EnergySystems initial sets and parameters with the ones defined by the user. The create_model_file() reads the GOCPI_OseMOSYS_Structure.xlsx explained in section 4.5.4 using the pandas module and writes the generated dataframe to a text file in the correct structure.

The create_data_file() function is the most extensive of the three functions in this class. Simplistically, the function write the python arrays and matrices for each set and parameter to one text file. Each set and parameter must have the correct syntax and structure to create a compatible data file written as a text file. Effectively, user-defined python-based multi-dimensional lists/arrays had to be converted into GNU Mathprog-structure data blocks. This GNU Mathprog conversion had to occur for all 11 sets and 52 parameters. Sets and parameters varied in size, ranging from one dimensional arrays/lists to five dimensional matrices. The variability of size and need for correct naming conventions created a considerable design challenge. This solution had to change based on the users inputs while still generating the correct GNU Mathprog structure. The solution was to design a conversion block for each set and parameter. Examples of set and conversion block are shown in listings 1 and 2 in the appendix section 9.0.1. Please refer to the Research Compendium for the complete set of 63 Python to GNU Mathprog Conversion Blocks in the create_data_file() function. The create_data_file() is approximately 2000 lines including commenting and formatting described in section 4.1.9.

4.9.4 CreateCases Module

The create cases module is the most comprehensive of the GOCPI package. This module sets the user-defined sets and parameters to an energy systems class. The class is initialised with an empty set of energy system sets and parameters. Each set and parameter has it's own function to set the correct values. The construction of an energy system occurs in a processing script. The development of the NZ energy system was facilitated in the (**GOCPI_NZ_Example.gyp**) script as explained in section 4.6.2. Please refer to

the **Research Compendium** for a complete set of the functions in **CreateCases** class and the **GOCPI_NZ_Example.gyp** Processing Script.

4.9.5 Forecasting Class

The forecasting class contains three modules. The **energy_balance_base()**, **calculate_constant_average_growth_rate()** and **calculate_cagr_forecast()**. A major issue in designing energy systems is the inconsistency of energy data across different regions. An energy system requires consistent reference energy systems across regions. The structure is the same with the flows and values different.

The energy_balance_base() module solves this issue. The International Energy Agency publishes a comprehensive dataset for energy balances in OECD countries. A process was developed to extract IEA's Energy balance for a target region. The 2017 IEA world energy balances were downloaded and stored in two csv files, approximately $1.35 million data points. The files are processed to extract a unique list of fuels, technologies and regions. These lists are stored as data frames. The user is prompted to select a target region to extract energy balances. The energy balances for the target region is stored as a pivot table and returns a dictionary with the unique list and pivot table. The NZ energy system example uses different data for proof of concept. Further developments should prioritise the International Energy Agency dataset to standardise the structure of energy balances for reference energy systems.

The calculate_constant_average_growth_rate() is a simple function to calculate the constant average growth rate (CAGR) for a time period. The formula is expressed in figure 8.

The calculate_cagr_forecast() uses the calculate_constant_average_growth_rate() function to forecast energy balances. Several dictionaries are passed into the function to forecast the energy balance for each interval in the time horizon.

4.9.6 Optimisation Class

The optimisation class has two core functions: **run_cplex_local()** and **run_ibm_wml_do()**. These two modules run IBM's commercial solvers, either locally or remotely, to minimise the total discounted cost for the energy system.

The run_cplex_local() module initialises a model, loads the energy system model file, solves the problem and returns the objective function value.

The run_ibm_wml_do() is more complex. Firstly, the function initialises IBM credentials to grant the user remote access to IBM Cloud services. After initialising credentials, a deployment space is created on the IBM Cloud service. If needed, data assets are created. After the creation of a deployment space, a deployment is either created or loaded from an existing set. Lastly, a model payload is sent as a new job request to the deployment to solve the problem using the IBM Watson Machine Learning Decision Optimisation service. This function is written to accommodate future developments of a Python-based OseMOSYS methodology.

The remaining functions are in development at this stage of the project.

5 Results

5.1 Outline

The intention of this project was to develop a global carbon pricing optimisation model. The project changed in scope slightly as limitations were discovered, leading to the need to pivot in new directions. Carbon taxes are imposed as constraints in the OseMOSYS methodology. This project successfully sets the foundation for building a Python-based open source energy modelling tool. The TIMES methodology explored was not suitable for building a scalable solution. OseMOSYS was a better alternative as the methodology is more intuitive while still capturing the complexity required for energy modelling. An energy modeller can access the OseMOSYS model in Excel to customise the model formulation to meet their needs. Geographies are partitioned to use regions and IEA world energy balances in order to construct reference energy systems. The user can devise processing scripts to design and formulate their own energy systems with Python dictionaries and matrices. The GOCPI package provides a suite of classes to build, solve and interpret energy systems. A user can design energy systems, navigate directories to access files, formulate energy systems, make forecasts and solve energy systems with IBM optimisation technologies using the package. The package is distributed using PyPI and can be installed using pip, Python's package management software. All project related information is accessible on a website which acts as an interface for the user. A pipeline was created to build a Python-based OseMOSYS model, continue to build the user-defined energy systems and use IBM Cloud technologies to inform analysis. The project set the foundation to create an accessible, scalable energy modelling tool in order to make informed decisions about energy investment and policy.

5.2 Deliverables

This project does not have a traditional set of results from experimentation. The project has set of deliverables informing a minimal viable product (MVP) and the process for further development. The following list describes the deliverables achieved in this project:

1. An adaptation of the OseMOSYS Methodology a user can customise to meet modelling needs.
2. A method to generate energy balances for regions from IEA world energy balances.
3. A processing script to build user defined energy systems.
4. A partially complete energy system, with a bi-lateral trade relationship between Australia and New Zealand, to show proof of concept.
5. The ability to solve energy system models locally using the IBM ILOG CPLEX Optimisation Studio and associated Python APIs.
6. A pipeline to use IBM Cloud services to solve energy system models using the IBM Decision Optimisation service.
7. A Python-based open source energy modelling package (GOCPI) distributable using PyPI. Use the following command to access: **pip install GOCPI**.
    (a) Navigation: Navigate directory structures to access key resources and data.

(b) EnergySystems: Convert energy systems into the necessary formats needed to solve the system.

(c) CreateCases: Build energy systems from Python arrays and dictionaries.

(d) Forecasting: Generate energy balances and forecast energy/financial information.

(e) Optimisation: Utilize IBM technologies to solve energy systems. These models may be solved locally or remotely.

8. A website to access key project resources and communicate information related to the project. **Visit the website here to view content.**

6 Discussion

6.1 Outcome

Here, we summarise the key findings from the project. Section 6.1.1 outlines the inaccessibility of the legacy approach to energy modelling. Section 6.1.2 identifies OseMOSYS is the best alternative to the legacy offering. Section 6.1.3 explains the GOCPI package developed as the main project deliverable. Section 6.1.4 identifies the construction of a process to correctly design energy systems. Section 6.2 outlines the improvements to make to the modelling approach and GOCPI package.

6.1.1 TIMES

TIMES is the legacy energy modelling approach for energy systems. After attempting to set up the structure required to create a scalable energy system, the approach was deemed not fit for purpose considering the scope of the project. Ultimately, reproducibility, integration and complexity issues led to the abandonment of this method.

6.1.2 OseMOSYS

The alternative energy modelling methodology OseMOSYS replaced TIMES. The OseMOSYS structure is similar to most linear programmes with sets, parameters, an objective function, constraints and variables defining an energy system. These models are easy to interpret with constraints and objective function customisable to meet modelling needs. The GNU Mathprog version of OseMOSYS was adapted to build energy systems from reference energy systems.

6.1.3 GOCPI: An open source, scalable energy modelling solution

The main objective of the project was to build a scalable modelling tool. This was achieved in the GOCPI python package. The code repository is version controlled and accessible using GitHub. The package is distributed on PyPI and downloadable using the pip package management software. There are five classes in the package to build and solve energy systems. All classes and modules are well documented for interpretability and inform future developments. The package interfaces with IBM Cloud technologies to solve energy systems.

6.1.4 Methodology

The project establishes a structure for generating energy systems in GNU Mathprog from Python dictionaries, matrices, Numpy arrays and lists. Energy system sets and parameters must be structured consistently to correctly design a system. The process converts one to five dimensional Python matrices into combinations of text blocks need to create lp formatted text files. This structure enables the user to model N combinations of regions in one energy system.

6.2 Outlook

6.2.1 Python Development

The OseMOSYS methodology must be adapted into a Python implementation to take advantage of IBM technologies and solve larger energy systems using the IBM Decision Optimisation service. There is a Python version available for distribution from the OseMOSYS GitHub repository. This version uses Pyomo, a Python-based, open-source optimisation modelling language. Pyomo was not originally chosen as larger energy systems could not be solved remotely on powerful commercial solvers e.g. IBM Cloud of Gurobi. A Python adaptation will be required moving forward.

6.2.2 NZ Example Completion

The energy system with a bi-lateral relationship between New Zealand and Australia is partially complete. Data inconsistency and inaccessibility made it difficult to construct an accurate representation of this energy system. Building this example could be a project in itself. It is important for proof of concept an example system is finished.

6.2.3 Develop Simulation Processes

Energy systems are stochastic in nature with production, conversion and consumption prone to variation e.g. renewable energy production. These uncertainties must be addressed by incorporating stochastic processes in energy modelling. This could be in the form of Autoregressive Integrated Moving Average models or Stochastic Dual Dynamic Programming (SDDP).

6.2.4 Develop Forecasting Methodologies

IBM Cloud has a range of forecasting services using machine learning techniques. These may be used to forecast financial and energy-relate information. Forecasts are currently estimated using constant average growth rates which are unrealistic. There may be unexplained factors to consider in estimating forecasts which more complex methodologies consider.

6.2.5 Develop Investment Strategies

The energy transition will require different investments depending on different stakeholders. A range of investment strategies need to be considered to drive equitable outcomes for the energy transition.

6.2.6 Develop Interactive Interface

It is important to be able to interpret the outputs from an energy system. The objective function, variables and constraints are to be displayed on the web-based user interface.

7 Summary and Conclusion

Energy modelling is inherently quite complex under existing methodologies. It doesn't have to be. The OseMOSYS methodology is a simple alternative that still captures many of the complexities underpinning energy modelling. The GNU Mathprog version of OseMOSYS was adapted to build a scalable method for building energy systems. Reference energy systems for OECD regions are easy to generate. A Python-based open-source energy modelling package gives the modeler the necessary tools to construct energy systems. A pipeline of optimisation and forecasting functionalities is set to further develop the product for functional use. The interface with IBM Cloud technologies enables powerful predictive possibilities. The energy modelers have access to project resources from the web-based interface. This project takes a big leap forward in making energy accessible and removing many of the technical barriers. Continuing to adopt the OseMOSYS methodology and develop the GOCPI package will help inform energy investment and policy. Overall, this initiative will drive equitable outcomes for the energy transition.

8 References

References

[1] International Electricity Agency. *Fuels and Technologies*. Available at https://www.iea.org/fuels-and-technologies (2020/04/21).

[2] International Energy Agency. *World Energy Balances and Statistics*. Available at https://www.iea.org/subscribe-to-data-services/world-energy-balances-and-statistics (2019/10/21).

[3] International Energy Agency. *World Energy Model Documentation*. Tech. rep. 2019.

[4] United States Environmental Protection Agency. *Overview of Greenhouse Gas Emissions*. Available at https://www.epa.gov/ghgemissions/overview-greenhouse-gases (2020/04/21).

[5] AMPL Optimization Inc. *AMPL*. Aug. 8, 2020. URL: https://ampl.com/.

[6] Anaconda, Inc. *Anaconda*. Version 2020.02. Aug. 8, 2020. URL: https://www.anaconda.com/.

[7] Marcelo Arenas. "Boyce-Codd Normal Form". In: *Encyclopedia of Database Systems*. Ed. by LING LIU and M. TAMER ÖZSU. Boston, MA: Springer US, 2009, pp. 264–265. ISBN: 978-0-387-39940-9. DOI: 10.1007/978-0-387-39940-9_1245. URL: https://doi.org/10.1007/978-0-387-39940-9_1245.

[8] Reserve Bank of Australia. *Zero Coupon Interest Rates - Analytical Series - 2009 to Current)*. Available at https://www.rba.gov.au/statistics/tables/ (2020/08/15).

[9] Reven S. Avi-Yonah and David M.Uhlmann. "Combating Global Climate Change: Why a Carbon Tax Is a Better Response to Global Warming Than Cap and Trade". In: *Stanford Environmental Law Journal* 28.3 (2009).

[10] The World Bank. *The World Bank Data*. Available at https://data.worldbank.org/indicator/NY.GDP.PCAP.KD (2020/04/21).

[11] The World Bank. *World Bank Indicators: Structure of Output*. Available at http://wdi.worldbank.org/table (2020/04/23).

[12] Marco R. Barassi and Nicola Spagnolo. "Linear and Non-linear Causality between CO2 Emissions and Economic Growth". In: *The Energy Journal* 33.3 (2012), pp. 23–38.

[13] Chantel Beck et al. "The Future is Now: How Oil and Gas Companies can Decarbonixe". McKinsey and Company: Oil and Gas Analysis. 2020.

[14] Daniel Bodansky. "The Paris Climate Change Agreement: A New Hope?" In: *American Journal of International Law* 110.2 (2016), pp. 288–319.

[15] Richard Bronson and Gabriel B. Costa. "4 - An Introduction to Optimization". In: *Matrix Methods (Third Edition)*. Ed. by Richard Bronson and Gabriel B. Costa. Third Edition. Boston: Academic Press, 2009, pp. 127–148. ISBN: 978-0-12-374427-2. DOI: https://doi.org/10.1016/B978-0-08-092225-6.50010-3. URL: http://www.sciencedirect.com/science/article/pii/B9780080922256500103.

[16] Ministry of Business Innovation and Employment. *Energy Statistics and Modelling*. Available at https://www.mbie.govt.nz/building-and-energy/energy-and-natural-resources/energy-statistics-and-modelling/ (2019/10/21).

[17] Intergovernmental Panel on Climate Change. *Global Warming of 1.5 C*. Tech. rep.

[18] United Nations Framework Convention on Climate Change. *What is the Kyoto Protocol?* Available at https://unfccc.int/kyoto_protocol (2020/04/23).

[19] United Nations Framework Convention on Climate Change. *What is the Paris Agreement?* Available at https://unfccc.int/process-and-meetings/the-paris-agreement/what-is-the-paris-agreement (2020/04/23).
[20] BusinessNZ Energy Council. *New Zealand Energy Scenarios - Navigating Energy Futures to 2050*. Tech. rep. 2015.
[21] BusinessNZ Energy Council. *New Zealand Energy Scenarios - Navigating Energy Futures to 2060*. Tech. rep. 2019.
[22] Energy Department of Industry Science and Australian Government Resources. *Energy Data*. Available at https://www.energy.gov.au/government-priorities/energy-data (2019/10/21).
[23] Chaido Dritsaki and Melina Dritsaki. "Causal Relationship between Energy Consumption, Economic Growth and CO2 Emissions: A Dynamic Panel Data Approach". In: *International Journal of Energy Economics and Policy* 4.2 (2014), pp. 125–136.
[24] Kanors EMR. *VErsatile Data Analyst*. Available at https://www.kanors-emr.org/ (2020/10/03).
[25] ETSAP. *Overview of TIMES Modelling Tool*. Available at https://iea-etsap.org/index.php/etsap-tools/model-generators/times (2020/04/26).
[26] Yahoo Finance. *ASX 200 and NZX 50 Historical Returns*. Available at https://nz.finance.yahoo.com/ (2019/10/21).
[27] GAMS. *An Introduction to GAMS*. Available at https://www.gams.com/products/introduction/ (2020/04/23).
[28] Google LLC. *Google Drive*. Aug. 5, 2020. URL: https://www.google.com/drive/.
[29] The Australian Government. *Financial Statements of the Government of New Zealand for the year ended 30 June 2019*. https://www.finance.gov.au/sites/default/files/2019-12/consolidated-financial-statements-201819.pdf. Dec. 2019.
[30] Bloomberg Green. *Renewable Investment*. Available at https://www.bloomberg.com/graphics/climate-change-data-green/investment.html (2020/04/23).
[31] World Bank Group. *State and Trends of Carbon Pricing 2019*. Tech. rep. 2019.
[32] Barry Warsaw Guido Van Rossum and Nick Coghlan. "PEP 8: Style Guide for Python Code". Python. org, 1565. 2001.
[33] Robin John Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. English. 2nd. Australia: OTexts, 2018.
[34] IBM. *IBM Academic Initiative)*. Available at https://www.ibm.com/academic/home (2020/08/09).
[35] IBM. *IBM Watson Machine Learning*. Available at https://www.ibm.com/cloud/machine-learning (2020/09/23).
[36] International Business Machines Corporation. *IBM ILOG CPLEX Optimization Studio*. Version 12.10. Aug. 9, 2020. URL: https://www.ibm.com/nz-en/products/ilog-cplex-optimization-studio.
[37] Linus Torvalds. *Git*. Version 2.28.0. Aug. 5, 2020. URL: https://git-scm.com/.
[38] Richard Loulou, Gary Goldstein, and Ken Noble. "Documentation for the MARKEL Family of Models". In: (2004).
[39] Richard Loulou et al. "Documentation for the TIMES Model - PART I 1–78". In: (2005).
[40] Medium. *Top 10 In-Demand programming languages to learn in 2020*. Available at https://towardsdatascience.com/top-10-in-demand-programming-languages-to-learn-in-2020-4462eb7d8d3e (2020/08/06).

[41] United Nations. *Paris Agreement*. Tech. rep. 2015.
[42] Reserve Bank of New Zealand. *Wholesale Interest Rates -B2*. Available at https://www.rbnz.govt.nz/statistics/b2 (2020/08/15).
[43] OSeMOSYS. *Open Source Energy Modelling System*. Available at http://www.osemosys.org/ (2020/08/02).
[44] BP plc. *BP Statistical Review of World Energy*. Tech. rep. 2019.
[45] Florian A. Potra and Stephen J. Wright. "Interior-point methods". In: *Journal of Computational and Applied Mathematics* 124.1 (2000). Numerical Analysis 2000. Vol. IV: Optimization and Nonlinear Equations, pp. 281–302. ISSN: 0377-0427. DOI: https://doi.org/10.1016/S0377-0427(00)00433-7. URL: http://www.sciencedirect.com/science/article/pii/S0377042700004337.
[46] PYPL. *PYPL PopularitY of Programming Language*. Available at http://pypl.github.io/PYPL.html (2020/08/06).
[47] World Energy Scenarios. *World Energy Scenarios 2019*. Tech. rep. 2019.
[48] David I. Stern. "The Role of Energy in Economic Growth". In: *SSRN Electronic Journal* (2010).
[49] T Sterner and G Kohlin. "Pricing Carbon: The Challenges". PhD thesis. 2015.
[50] Massachusetts Institute of Technology. *GNU Mathprog*. Available at http://web.mit.edu/lpsolve_v5525/doc/MathProg.htm (2020/08/08).
[51] Financial Times. *US Oil Price Back Below Zero After Historic Plunge*. Available at https://www.ft.com/content/26ea5ef9-0619-4e50-b605-58e36d3fc4d9 (2020/04/21).
[52] Deloitte US. *2020 Renewable Energy Industry Outlook*. Tech. rep. 2019.
[53] Various. *OseMOSYS Github Repository.* Available at https://github.com/OSeMOSYS/OSeMOSYS (2020/10/10).
[54] Greg Wilson et al. "Good Enough Practices in Scientific Computing". In: *PLOS Computational Biology* 13 (Aug. 2016). DOI: 10.1371/journal.pcbi.1005510.
[55] Treasury New Zealand. *Financial Statements of the Government of New Zealand for the year ended 30 June 2019*. https://treasury.govt.nz/sites/default/files/2019-10/fsgnz-2019.pdf. June 2019.

9 Appendices

9.0.1 Set and Parameter Conversion Blocks

```
# year
set_string = ' '.join(self.year)
f.write('set YEAR\t:=\t{0};\n'.format(set_string))
```

Listing 1: One Dimensional Set Python to GNU Mathprog Conversion Block

```

# DaySplit = np.zeros((llh,ly))
param = 'DaySplit'
f.write('#\n')
columns = self.year
column_string = ' '.join(columns)
# Writes index specific parameter values to the text files
if toggle_defaults[param] == True:
  f.write("param\t{0}\tdefault\t{1}:\t{2}:=\n".format(
    param, defaults_dictionary[param], column_string))
  # Converts maxtrix rows to list
  array = np.array(self.dailytimebracket)
  array = array.T
  lt = array.tolist()
  # Creates 2D matrix for this value
  mat = self.DaySplit[:, :]
  # Converts combined matrix to list and combines lists
  matlist = mat.tolist()
  #Combines the two lists
  combined_list = list(zip(lt, matlist))
  # Writes index specific parameter values to the text files
  f.write("param\t{0}\t:{1}:=\n".format(param, column_string))
  for line in combined_list:
    combinedflat = ''.join(str(line))
    combinedflat = combinedflat.replace('[', '')
    combinedflat = combinedflat.replace(']', '')
    combinedflat = combinedflat.replace("'", '')
    combinedflat = combinedflat.replace(",", '')
    combinedflat = combinedflat.replace("(", '')
    combinedflat = combinedflat.replace(")", '')
    f.write("{0}\n".format(combinedflat))
else:
  f.write("param\t{0}\tdefault\t{1}:=\n".format(
    param, defaults_dictionary[param]))
f.write(';\n')
```

Listing 2: Two Dimensional Parameter Python to GNU Mathprog Conversion Block

9.1 OseMOSYS Model

The Open Source Energy Modelling System is a linear programme with sets, parameters, an objective function and constraints related to energy systems. The model is described below. Please refer to GOCPI OseMOSYS Structure.xlsx in the GitHub Repository for full descriptions of the sets, parameters and constraints.

```
set YEAR;
set TECHNOLOGY;
set TIMESLICE;
set FUEL;
set EMISSION;
set MODE_OF_OPERATION;
set REGION;
set SEASON;
set DAYTYPE;
set DAILYTIMEBRACKET;
set STORAGE;
param YearSplit{l in TIMESLICE,y in YEAR};
param DiscountRate{r in REGION};
param DaySplit{lh in DAILYTIMEBRACKET,y in YEAR};
param Conversionls{l in TIMESLICE,ls in SEASON};
param Conversionld{l in TIMESLICE,ld in DAYTYPE};
param Conversionlh{l in TIMESLICE,lh in DAILYTIMEBRACKET};
param DaysInDayType{ls in SEASON ,ld in DAYTYPE,y in YEAR};
param TradeRoute{r in REGION,rr in REGION,f in FUEL,y in YEAR};
param DepreciationMethod{r in REGION};
param SpecifiedAnnualDemand{r in REGION,f in FUEL,y in YEAR};
param SpecifiedDemandProfile{r in REGION,f in FUEL,l in TIMESLICE,y in
    YEAR};
param AccumulatedAnnualDemand{r in REGION,f in FUEL,y in YEAR};
param CapacityToActivityUnit{r in REGION,t in TECHNOLOGY};
param CapacityFactor{r in REGION,t in TECHNOLOGY,l in TIMESLICE,y in
    YEAR};
param AvailabilityFactor{r in REGION,t in TECHNOLOGY,y in YEAR};
param OperationalLife{r in REGION,t in TECHNOLOGY};
param ResidualCapacity{r in REGION,t in TECHNOLOGY,y in YEAR};
param InputActivityRatio{r in REGION,t in TECHNOLOGY,f in FUEL,m in
    MODE_OF_OPERATION,y in YEAR};
param OutputActivityRatio{r in REGION,t in TECHNOLOGY,f in FUEL,m in
    MODE_OF_OPERATION,y in YEAR};
param CapitalCost{r in REGION,t in TECHNOLOGY,y in YEAR};
param VariableCost{r in REGION,t in TECHNOLOGY,m in MODE_OF_OPERATION,y
     in YEAR};
param FixedCost{r in REGION,t in TECHNOLOGY,y in YEAR};
param TechnologyToStorage{r in REGION,t in TECHNOLOGY,s in STORAGE,m in
     MODE_OF_OPERATION};
param TechnologyFromStorage{r in REGION,t in TECHNOLOGY,s in STORAGE,m
    in MODE_OF_OPERATION};
param StorageLevelStart{r in REGION,s in STORAGE};
param StorageMaxChargeRate{r in REGION,s in STORAGE};
param StorageMaxDischargeRate{r in REGION,s in STORAGE};
param MinStorageCharge{r in REGION,s in STORAGE,y in YEAR};
param OperationalLifeStorage{r in REGION, s in STORAGE};
param CapitalCostStorage{r in REGION,s in STORAGE,y in YEAR};
param ResidualStorageCapacity{r in REGION,s in STORAGE,y in YEAR};
param CapacityOfOneTechnologyUnit{r in REGION,t in TECHNOLOGY,y in YEAR
    };
```

```
param TotalAnnualMaxCapacity{r in REGION,t in TECHNOLOGY,y in YEAR};
param TotalAnnualMinCapacity{r in REGION,t in TECHNOLOGY,y in YEAR};
param TotalAnnualMaxCapacityInvestment{r in REGION,t in TECHNOLOGY,y in
    YEAR};
param TotalAnnualMinCapacityInvestment{r in REGION,t in TECHNOLOGY,y in
    YEAR};
param TotalTechnologyAnnualActivityUpperLimit{r in REGION,t in
   TECHNOLOGY,y in YEAR};
param TotalTechnologyAnnualActivityLowerLimit{r in REGION,t in
   TECHNOLOGY,y in YEAR};
param TotalTechnologyModelPeriodActivityUpperLimit{r in REGION,t in
   TECHNOLOGY};
param TotalTechnologyModelPeriodActivityLowerLimit{r in REGION,t in
   TECHNOLOGY};
param ReserveMarginTagTechnology{r in REGION,t in TECHNOLOGY,y in YEAR
   };
param ReserveMarginTagFuel{r in REGION,f in FUEL,y in YEAR};
param ReserveMargin{r in REGION,y in YEAR};
param RETagTechnology{r in REGION,t in TECHNOLOGY,y in YEAR};
param RETagFuel{r in REGION,f in FUEL,y in YEAR};
param REMinProductionTarget{r in REGION,y in YEAR};
param EmissionActivityRatio{r in REGION,t in TECHNOLOGY,e in EMISSION,m
    in MODE_OF_OPERATION,y in YEAR};
param EmissionsPenalty{r in REGION,e in EMISSION,y in YEAR};
param AnnualExogenousEmission{r in REGION,e in EMISSION,y in YEAR};
param AnnualEmissionLimit{r in REGION,e in EMISSION,y in YEAR};
param ModelPeriodExogenousEmission{r in REGION,e in EMISSION};
param ModelPeriodEmissionLimit{r in REGION,e in EMISSION};
var RateOfDemand{r in REGION,l in TIMESLICE,f in FUEL,y in YEAR} >=0;
var Demand{r in REGION,l in TIMESLICE,f in FUEL,y in YEAR}>=0;
var RateOfStorageCharge{r in REGION,s in STORAGE,ls in SEASON,ld in
   DAYTYPE,lh in DAILYTIMEBRACKET,y in YEAR};
var RateOfStorageDischarge{r in REGION,s in STORAGE,ls in SEASON,ld in
   DAYTYPE,lh in DAILYTIMEBRACKET,y in YEAR};
var NetChargeWithinYear{r in REGION,s in STORAGE,ls in SEASON,ld in
   DAYTYPE,lh in DAILYTIMEBRACKET,y in YEAR};
var NetChargeWithinDay{r in REGION,s in STORAGE,ls in SEASON,ld in
   DAYTYPE,lh in DAILYTIMEBRACKET,y in YEAR};
var StorageLevelYearStart{r in REGION,s in STORAGE,y in YEAR}>=0;
var StorageLevelYearFinish{r in REGION,s in STORAGE,y in YEAR}>=0;
var StorageLevelSeasonStart{r in REGION,s in STORAGE,ls in SEASON,y in
   YEAR}>=0;
var StorageLevelDayTypeStart{r in REGION,s in STORAGE,ls in SEASON,ld
   in DAYTYPE,y in YEAR}>=0;
var StorageLevelDayTypeFinish{r in REGION,s in STORAGE,ls in SEASON,ld
   in DAYTYPE,y in YEAR}>=0;
var StorageLowerLimit{r in REGION,s in STORAGE,y in YEAR}>=0;
var StorageUpperLimit{r in REGION,s in STORAGE,y in YEAR}>=0;
var AccumulatedNewStorageCapacity{r in REGION,s in STORAGE,y in YEAR
   }>=0;
var NewStorageCapacity{r in REGION,s in STORAGE,y in YEAR}>=0;
var CapitalInvestmentStorage{r in REGION,s in STORAGE,y in YEAR}>=0;
var DiscountedCapitalInvestmentStorage{r in REGION,s in STORAGE,y in
   YEAR}>=0;
var SalvageValueStorage{r in REGION,s in STORAGE,y in YEAR}>=0;
var DiscountedSalvageValueStorage{r in REGION,s in STORAGE,y in YEAR
   }>=0;
var TotalDiscountedStorageCost{r in REGION,s in STORAGE,y in YEAR}>=0;
```

```
84 var NumberOfNewTechnologyUnits{r in REGION,t in TECHNOLOGY,y in YEAR
   }>=0, integer;
85 var NewCapacity{r in REGION,t in TECHNOLOGY,y in YEAR}>=0;
86 var AccumulatedNewCapacity{r in REGION,t in TECHNOLOGY,y in YEAR}>=0;
87 var TotalCapacityAnnual{r in REGION,t in TECHNOLOGY,y in YEAR}>=0;
88 var RateOfActivity{r in REGION,l in TIMESLICE,t in TECHNOLOGY,m in
   MODE_OF_OPERATION,y in YEAR} >=0;
89 var RateOfTotalActivity{r in REGION,t in TECHNOLOGY,l in TIMESLICE,y in
    YEAR} >=0;
90 var TotalTechnologyAnnualActivity{r in REGION,t in TECHNOLOGY,y in YEAR
   } >=0;
91 var TotalAnnualTechnologyActivityByMode{r in REGION,t in TECHNOLOGY,m
   in MODE_OF_OPERATION,y in YEAR} >=0;
92 var TotalTechnologyModelPeriodActivity{r in REGION,t in TECHNOLOGY};
93 var RateOfProductionByTechnologyByMode{r in REGION,l in TIMESLICE,t in
   TECHNOLOGY,m in MODE_OF_OPERATION,f in FUEL,y in YEAR}>=0;
94 var RateOfProductionByTechnology{r in REGION,l in TIMESLICE,t in
   TECHNOLOGY,f in FUEL,y in YEAR} >=0;
95 var ProductionByTechnology{r in REGION,l in TIMESLICE,t in TECHNOLOGY,f
    in FUEL,y in YEAR} >=0;
96 var ProductionByTechnologyAnnual{r in REGION,t in TECHNOLOGY,f in FUEL,
   y in YEAR} >=0;
97 var RateOfProduction{r in REGION,l in TIMESLICE,f in FUEL,y in YEAR
   }>=0;
98 var Production{r in REGION,l in TIMESLICE,f in FUEL,y in YEAR} >=0;
99 var RateOfUseByTechnologyByMode{r in REGION,l in TIMESLICE,t in
   TECHNOLOGY,m in MODE_OF_OPERATION,f in FUEL,y in YEAR} >=0;
100 var RateOfUseByTechnology{r in REGION,l in TIMESLICE,t in TECHNOLOGY,f
   in FUEL,y in YEAR} >=0;
101 var UseByTechnologyAnnual{r in REGION,t in TECHNOLOGY,f in FUEL,y in
   YEAR} >=0;
102 var UseByTechnology{r in REGION,l in TIMESLICE,t in TECHNOLOGY,f in
   FUEL,y in YEAR} >=0;
103 var RateOfUse{r in REGION,l in TIMESLICE,f in FUEL,y in YEAR}>=0;
104 var Use{r in REGION,l in TIMESLICE,f in FUEL,y in YEAR} >=0;
105 var Trade{r in REGION,rr in REGION,l in TIMESLICE,f in FUEL,y in YEAR};
106 var TradeAnnual{r in REGION,rr in REGION,f in FUEL,y in YEAR};
107 var ProductionAnnual{r in REGION,f in FUEL,y in YEAR} >=0;
108 var UseAnnual{r in REGION,f in FUEL,y in YEAR}>=0;
109 var CapitalInvestment{r in REGION,t in TECHNOLOGY,y in YEAR}>=0;
110 var DiscountedCapitalInvestment{r in REGION,t in TECHNOLOGY,y in YEAR}
   >=0;
111 var SalvageValue{r in REGION,t in TECHNOLOGY,y in YEAR} >=0;
112 var DiscountedSalvageValue{r in REGION,t in TECHNOLOGY,y in YEAR} >=0;
113 var OperatingCost{r in REGION,t in TECHNOLOGY,y in YEAR} >=0;
114 var DiscountedOperatingCost{r in REGION,t in TECHNOLOGY,y in YEAR} >=0;
115 var AnnualVariableOperatingCost{r in REGION,t in TECHNOLOGY,y in YEAR}
   >=0;
116 var AnnualFixedOperatingCost{r in REGION,t in TECHNOLOGY,y in YEAR}
   >=0;
117 var TotalDiscountedCostByTechnology{r in REGION,t in TECHNOLOGY,y in
   YEAR} >=0;
118 var TotalDiscountedCost{r in REGION,y in YEAR} >=0;
119 var ModelPeriodCostByRegion{r in REGION}>=0;
120 var TotalCapacityInReserveMargin{r in REGION,y in YEAR} >=0;
121 var DemandNeedingReserveMargin{r in REGION,l in TIMESLICE,y in YEAR}
   >=0;
122 var TotalREProductionAnnual{r in REGION,y in YEAR};
```

```
123 var RETotalProductionOfTargetFuelAnnual{r in REGION,y in YEAR};
124 var AnnualTechnologyEmissionByMode{r in REGION,t in TECHNOLOGY,e in
    EMISSION,m in MODE_OF_OPERATION,y in YEAR} >=0;
125 var AnnualTechnologyEmission{r in REGION,t in TECHNOLOGY,e in EMISSION,
    y in YEAR} >=0;
126 var AnnualTechnologyEmissionPenaltyByEmission{r in REGION,t in
    TECHNOLOGY,e in EMISSION,y in YEAR} >=0;
127 var AnnualTechnologyEmissionsPenalty{r in REGION,t in TECHNOLOGY,y in
    YEAR} >=0;
128 var DiscountedTechnologyEmissionsPenalty{r in REGION,t in TECHNOLOGY,y
    in YEAR} >=0;
129 var AnnualEmissions{r in REGION,e in EMISSION,y in YEAR} >=0;
130 var ModelPeriodEmissions{r in REGION,e in EMISSION} >=0;
131 minimize cost: sum{r in REGION, y in YEAR} TotalDiscountedCost[r,y];
132 s.t. EQ_SpecifiedDemand{r in REGION, l in TIMESLICE, f in FUEL, y in
    YEAR}: SpecifiedAnnualDemand[r,f,y]*SpecifiedDemandProfile[r,f,l,y]
    / YearSplit[l,y]=RateOfDemand[r,l,f,y];
133 s.t. CAa1_TotalNewCapacity{r in REGION, t in TECHNOLOGY, y in YEAR}:
    AccumulatedNewCapacity[r,t,y] = sum{yy in YEAR: y-yy <
    OperationalLife[r,t] && y-yy>=0} NewCapacity[r,t,yy];
134 s.t. CAa2_TotalAnnualCapacity{r in REGION, t in TECHNOLOGY, y in YEAR}:
     AccumulatedNewCapacity[r,t,y]+ ResidualCapacity[r,t,y] =
    TotalCapacityAnnual[r,t,y];
135 s.t. CAa3_TotalActivityOfEachTechnology{r in REGION, t in TECHNOLOGY, l
     in TIMESLICE, y in YEAR}: sum{m in MODE_OF_OPERATION}
    RateOfActivity[r,l,t,m,y] = RateOfTotalActivity[r,t,l,y];
136 s.t. CAa4_Constraint_Capacity{r in REGION, l in TIMESLICE, t in
    TECHNOLOGY, y in YEAR}: RateOfTotalActivity[r,t,l,y] <=
    TotalCapacityAnnual[r,t,y] * CapacityFactor[r,t,l,y]*
    CapacityToActivityUnit[r,t];
137 s.t. CAa5_TotalNewCapacity{r in REGION, t in TECHNOLOGY, y in YEAR:
    CapacityOfOneTechnologyUnit[r,t,y]<>0}: CapacityOfOneTechnologyUnit[
    r,t,y]*NumberOfNewTechnologyUnits[r,t,y] = NewCapacity[r,t,y];
138 s.t. CAb1_PlannedMaintenance{r in REGION, t in TECHNOLOGY, y in YEAR}:
    sum{l in TIMESLICE} RateOfTotalActivity[r,t,l,y]*YearSplit[l,y] <=
    sum{l in TIMESLICE} (TotalCapacityAnnual[r,t,y]*CapacityFactor[r,t,l
    ,y]*YearSplit[l,y])* AvailabilityFactor[r,t,y]*
    CapacityToActivityUnit[r,t];
139 s.t. EBa1_RateOfFuelProduction1{r in REGION, l in TIMESLICE, f in FUEL,
     t in TECHNOLOGY, m in MODE_OF_OPERATION, y in YEAR:
    OutputActivityRatio[r,t,f,m,y] <>0}: RateOfActivity[r,l,t,m,y]*
    OutputActivityRatio[r,t,f,m,y]  = RateOfProductionByTechnologyByMode
    [r,l,t,m,f,y];
140 s.t. EBa2_RateOfFuelProduction2{r in REGION, l in TIMESLICE, f in FUEL,
     t in TECHNOLOGY, y in YEAR}: sum{m in MODE_OF_OPERATION:
    OutputActivityRatio[r,t,f,m,y] <>0}
    RateOfProductionByTechnologyByMode[r,l,t,m,f,y] =
    RateOfProductionByTechnology[r,l,t,f,y];
141 s.t. EBa3_RateOfFuelProduction3{r in REGION, l in TIMESLICE, f in FUEL,
     y in YEAR}: sum{t in TECHNOLOGY} RateOfProductionByTechnology[r,l,t
    ,f,y]  =  RateOfProduction[r,l,f,y];
142 s.t. EBa4_RateOfFuelUse1{r in REGION, l in TIMESLICE, f in FUEL, t in
    TECHNOLOGY, m in MODE_OF_OPERATION, y in YEAR: InputActivityRatio[r,
    t,f,m,y]<>0}: RateOfActivity[r,l,t,m,y]*InputActivityRatio[r,t,f,m,y
    ]  = RateOfUseByTechnologyByMode[r,l,t,m,f,y];
143 s.t. EBa5_RateOfFuelUse2{r in REGION, l in TIMESLICE, f in FUEL, t in
    TECHNOLOGY, y in YEAR}: sum{m in MODE_OF_OPERATION:
    InputActivityRatio[r,t,f,m,y]<>0} RateOfUseByTechnologyByMode[r,l,t,
```

```
    m,f,y] = RateOfUseByTechnology[r,l,t,f,y];
144 s.t. EBa6_RateOfFuelUse3{r in REGION, l in TIMESLICE, f in FUEL, y in
    YEAR}: sum{t in TECHNOLOGY} RateOfUseByTechnology[r,l,t,f,y]  =
    RateOfUse[r,l,f,y];
145 s.t. EBa7_EnergyBalanceEachTS1{r in REGION, l in TIMESLICE, f in FUEL,
    y in YEAR}: RateOfProduction[r,l,f,y]*YearSplit[l,y] = Production[r,
    l,f,y];
146 s.t. EBa8_EnergyBalanceEachTS2{r in REGION, l in TIMESLICE, f in FUEL,
    y in YEAR}: RateOfUse[r,l,f,y]*YearSplit[l,y] = Use[r,l,f,y];
147 s.t. EBa9_EnergyBalanceEachTS3{r in REGION, l in TIMESLICE, f in FUEL,
    y in YEAR}: RateOfDemand[r,l,f,y]*YearSplit[l,y] = Demand[r,l,f,y];
148 s.t. EBa10_EnergyBalanceEachTS4{r in REGION, rr in REGION, l in
    TIMESLICE, f in FUEL, y in YEAR}: Trade[r,rr,l,f,y] = -Trade[rr,r,l,
    f,y];
149 s.t. EBa11_EnergyBalanceEachTS5{r in REGION, l in TIMESLICE, f in FUEL,
     y in YEAR}: Production[r,l,f,y] >= Demand[r,l,f,y] + Use[r,l,f,y] +
     sum{rr in REGION} Trade[r,rr,l,f,y]*TradeRoute[r,rr,f,y];
150 s.t. EBb1_EnergyBalanceEachYear1{r in REGION, f in FUEL, y in YEAR}:
    sum{l in TIMESLICE} Production[r,l,f,y] = ProductionAnnual[r,f,y];
151 s.t. EBb2_EnergyBalanceEachYear2{r in REGION, f in FUEL, y in YEAR}:
    sum{l in TIMESLICE} Use[r,l,f,y] = UseAnnual[r,f,y];
152 s.t. EBb3_EnergyBalanceEachYear3{r in REGION, rr in REGION, f in FUEL,
    y in YEAR}: sum{l in TIMESLICE} Trade[r,rr,l,f,y] = TradeAnnual[r,rr
    ,f,y];
153 s.t. EBb4_EnergyBalanceEachYear4{r in REGION, f in FUEL, y in YEAR}:
    ProductionAnnual[r,f,y] >= UseAnnual[r,f,y] + sum{rr in REGION}
    TradeAnnual[r,rr,f,y]*TradeRoute[r,rr,f,y] + AccumulatedAnnualDemand
    [r,f,y];
154 s.t. Acc1_FuelProductionByTechnology{r in REGION, l in TIMESLICE, t in
    TECHNOLOGY, f in FUEL, y in YEAR}: RateOfProductionByTechnology[r,l,
    t,f,y] * YearSplit[l,y] = ProductionByTechnology[r,l,t,f,y];
155 s.t. Acc2_FuelUseByTechnology{r in REGION, l in TIMESLICE, t in
    TECHNOLOGY, f in FUEL, y in YEAR}: RateOfUseByTechnology[r,l,t,f,y]
    * YearSplit[l,y] = UseByTechnology[r,l,t,f,y];
156 s.t. Acc3_AverageAnnualRateOfActivity{r in REGION, t in TECHNOLOGY, m
    in MODE_OF_OPERATION, y in YEAR}: sum{l in TIMESLICE} RateOfActivity
    [r,l,t,m,y]*YearSplit[l,y] = TotalAnnualTechnologyActivityByMode[r,t
    ,m,y];
157 s.t. Acc4_ModelPeriodCostByRegion{r in REGION}: sum{y in YEAR}
    TotalDiscountedCost[r,y] = ModelPeriodCostByRegion[r];
158 s.t. S1_RateOfStorageCharge{r in REGION, s in STORAGE, ls in SEASON, ld
     in DAYTYPE, lh in DAILYTIMEBRACKET, y in YEAR}: sum{t in TECHNOLOGY
    , m in MODE_OF_OPERATION, l in TIMESLICE:TechnologyToStorage[r,t,s,m
    ]>0} RateOfActivity[r,l,t,m,y] * TechnologyToStorage[r,t,s,m] *
    Conversionls[l,ls] * Conversionld[l,ld] * Conversionlh[l,lh] =
    RateOfStorageCharge[r,s,ls,ld,lh,y];
159 s.t. S2_RateOfStorageDischarge{r in REGION, s in STORAGE, ls in SEASON,
     ld in DAYTYPE, lh in DAILYTIMEBRACKET, y in YEAR}: sum{t in
    TECHNOLOGY, m in MODE_OF_OPERATION, l in TIMESLICE:
    TechnologyFromStorage[r,t,s,m]>0} RateOfActivity[r,l,t,m,y] *
    TechnologyFromStorage[r,t,s,m] * Conversionls[l,ls] * Conversionld[l
    ,ld] * Conversionlh[l,lh] = RateOfStorageDischarge[r,s,ls,ld,lh,y];
160 s.t. S3_NetChargeWithinYear{r in REGION, s in STORAGE, ls in SEASON, ld
     in DAYTYPE, lh in DAILYTIMEBRACKET, y in YEAR}: sum{l in TIMESLICE:
    Conversionls[l,ls]>0&&Conversionld[l,ld]>0&&Conversionlh[l,lh]>0}  (
    RateOfStorageCharge[r,s,ls,ld,lh,y] - RateOfStorageDischarge[r,s,ls,
    ld,lh,y]) * YearSplit[l,y] * Conversionls[l,ls] * Conversionld[l,ld]
     * Conversionlh[l,lh] = NetChargeWithinYear[r,s,ls,ld,lh,y];
```

```
161 s.t. S4_NetChargeWithinDay{r in REGION, s in STORAGE, ls in SEASON, ld
    in DAYTYPE, lh in DAILYTIMEBRACKET, y in YEAR}: (RateOfStorageCharge
    [r,s,ls,ld,lh,y] - RateOfStorageDischarge[r,s,ls,ld,lh,y]) *
    DaySplit[lh,y] = NetChargeWithinDay[r,s,ls,ld,lh,y];
162 s.t. S5_and_S6_StorageLevelYearStart{r in REGION, s in STORAGE, y in
    YEAR}: if y = min{yy in YEAR} min(yy) then StorageLevelStart[r,s]
    else StorageLevelYearStart[r,s,y-1] + sum{ls in SEASON, ld in
    DAYTYPE, lh in DAILYTIMEBRACKET} NetChargeWithinYear[r,s,ls,ld,lh,y
    -1] = StorageLevelYearStart[r,s,y];
163 s.t. S7_and_S8_StorageLevelYearFinish{r in REGION, s in STORAGE, y in
    YEAR}: if y < max{yy in YEAR} max(yy) then StorageLevelYearStart[r,s
    ,y+1] else StorageLevelYearStart[r,s,y] + sum{ls in SEASON, ld in
    DAYTYPE, lh in DAILYTIMEBRACKET} NetChargeWithinYear[r,s,ls,ld,lh,y]
     = StorageLevelYearFinish[r,s,y];
164 s.t. S9_and_S10_StorageLevelSeasonStart{r in REGION, s in STORAGE, ls
    in SEASON, y in YEAR}: if ls = min{lsls in SEASON} min(lsls) then
    StorageLevelYearStart[r,s,y] else StorageLevelSeasonStart[r,s,ls-1,y
    ] + sum{ld in DAYTYPE, lh in DAILYTIMEBRACKET} NetChargeWithinYear[r
    ,s,ls-1,ld,lh,y] = StorageLevelSeasonStart[r,s,ls,y];
165 s.t. S11_and_S12_StorageLevelDayTypeStart{r in REGION, s in STORAGE, ls
     in SEASON, ld in DAYTYPE, y in YEAR}: if ld = min{ldld in DAYTYPE}
    min(ldld) then StorageLevelSeasonStart[r,s,ls,y] else
    StorageLevelDayTypeStart[r,s,ls,ld-1,y] + sum{lh in DAILYTIMEBRACKET
    } NetChargeWithinDay[r,s,ls,ld-1,lh,y] * DaysInDayType[ls,ld-1,y] =
    StorageLevelDayTypeStart[r,s,ls,ld,y];
166 s.t. S13_and_S14_and_S15_StorageLevelDayTypeFinish{r in REGION, s in
    STORAGE, ls in SEASON, ld in DAYTYPE, y in YEAR}:  if ls = max{lsls
    in SEASON} max(lsls) && ld = max{ldld in DAYTYPE} max(ldld) then
    StorageLevelYearFinish[r,s,y] else if ld = max{ldld in DAYTYPE} max(
    ldld) then StorageLevelSeasonStart[r,s,ls+1,y] else
    StorageLevelDayTypeFinish[r,s,ls,ld+1,y] - sum{lh in
    DAILYTIMEBRACKET} NetChargeWithinDay[r,s,ls,ld+1,lh,y] *
    DaysInDayType[ls,ld+1,y] = StorageLevelDayTypeFinish[r,s,ls,ld,y];
167 s.t.
    SC1_LowerLimit_BeginningOfDailyTimeBracketOfFirstInstanceOfDayTypeInFirstWeekCon
    {r in REGION, s in STORAGE, ls in SEASON, ld in DAYTYPE, lh in
    DAILYTIMEBRACKET, y in YEAR}: 0 <= (StorageLevelDayTypeStart[r,s,ls,
    ld,y]+sum{lhlh in DAILYTIMEBRACKET:lh-lhlh>0} NetChargeWithinDay[r,s
    ,ls,ld,lhlh,y])-StorageLowerLimit[r,s,y];
168 s.t.
    SC1_UpperLimit_BeginningOfDailyTimeBracketOfFirstInstanceOfDayTypeInFirstWeekCon
    {r in REGION, s in STORAGE, ls in SEASON, ld in DAYTYPE, lh in
    DAILYTIMEBRACKET, y in YEAR}: (StorageLevelDayTypeStart[r,s,ls,ld,y
    ]+sum{lhlh in DAILYTIMEBRACKET:lh-lhlh>0} NetChargeWithinDay[r,s,ls,
    ld,lhlh,y])-StorageUpperLimit[r,s,y] <= 0;
169 s.t.
    SC2_LowerLimit_EndOfDailyTimeBracketOfLastInstanceOfDayTypeInFirstWeekConstraint
    {r in REGION, s in STORAGE, ls in SEASON, ld in DAYTYPE, lh in
    DAILYTIMEBRACKET, y in YEAR}: 0 <= if ld > min{ldld in DAYTYPE} min(
    ldld) then (StorageLevelDayTypeStart[r,s,ls,ld,y]-sum{lhlh in
    DAILYTIMEBRACKET:lh-lhlh<0} NetChargeWithinDay[r,s,ls,ld-1,lhlh,y])-
    StorageLowerLimit[r,s,y];
170 s.t.
    SC2_UpperLimit_EndOfDailyTimeBracketOfLastInstanceOfDayTypeInFirstWeekConstraint
    {r in REGION, s in STORAGE, ls in SEASON, ld in DAYTYPE, lh in
    DAILYTIMEBRACKET, y in YEAR}: if ld > min{ldld in DAYTYPE} min(ldld)
     then (StorageLevelDayTypeStart[r,s,ls,ld,y]-sum{lhlh in
    DAILYTIMEBRACKET:lh-lhlh<0} NetChargeWithinDay[r,s,ls,ld-1,lhlh,y])-
```

```
    StorageUpperLimit[r,s,y] <= 0;
s.t.
    SC3_LowerLimit_EndOfDailyTimeBracketOfLastInstanceOfDayTypeInLastWeekConstraint
    {r in REGION, s in STORAGE, ls in SEASON, ld in DAYTYPE, lh in
    DAILYTIMEBRACKET, y in YEAR}: 0 <= (StorageLevelDayTypeFinish[r,s,ls
    ,ld,y] - sum{lhlh in DAILYTIMEBRACKET:lh-lhlh<0} NetChargeWithinDay[
    r,s,ls,ld,lhlh,y])-StorageLowerLimit[r,s,y];
s.t.
    SC3_UpperLimit_EndOfDailyTimeBracketOfLastInstanceOfDayTypeInLastWeekConstraint
    {r in REGION, s in STORAGE, ls in SEASON, ld in DAYTYPE, lh in
    DAILYTIMEBRACKET, y in YEAR}: (StorageLevelDayTypeFinish[r,s,ls,ld,y
    ] - sum{lhlh in DAILYTIMEBRACKET:lh-lhlh<0} NetChargeWithinDay[r,s,
    ls,ld,lhlh,y])-StorageUpperLimit[r,s,y] <= 0;
s.t.
    SC4_LowerLimit_BeginningOfDailyTimeBracketOfFirstInstanceOfDayTypeInLastWeekCons
    {r in REGION, s in STORAGE, ls in SEASON, ld in DAYTYPE, lh in
    DAILYTIMEBRACKET, y in YEAR}: 0 <= if ld > min{ldld in DAYTYPE} min(
    ldld) then (StorageLevelDayTypeFinish[r,s,ls,ld-1,y]+sum{lhlh in
    DAILYTIMEBRACKET:lh-lhlh>0} NetChargeWithinDay[r,s,ls,ld,lhlh,y])-
    StorageLowerLimit[r,s,y];
s.t.
    SC4_UpperLimit_BeginningOfDailyTimeBracketOfFirstInstanceOfDayTypeInLastWeekCons
    {r in REGION, s in STORAGE, ls in SEASON, ld in DAYTYPE, lh in
    DAILYTIMEBRACKET, y in YEAR}: if ld > min{ldld in DAYTYPE} min(ldld)
     then (StorageLevelDayTypeFinish[r,s,ls,ld-1,y]+sum{lhlh in
    DAILYTIMEBRACKET:lh-lhlh>0} NetChargeWithinDay[r,s,ls,ld,lhlh,y])-
    StorageUpperLimit[r,s,y] <= 0;
s.t. SC5_MaxChargeConstraint{r in REGION, s in STORAGE, ls in SEASON,
    ld in DAYTYPE, lh in DAILYTIMEBRACKET, y in YEAR}:
    RateOfStorageCharge[r,s,ls,ld,lh,y] <= StorageMaxChargeRate[r,s];
s.t. SC6_MaxDischargeConstraint{r in REGION, s in STORAGE, ls in SEASON
    , ld in DAYTYPE, lh in DAILYTIMEBRACKET, y in YEAR}:
    RateOfStorageDischarge[r,s,ls,ld,lh,y] <= StorageMaxDischargeRate[r,
    s];
s.t. SI1_StorageUpperLimit{r in REGION, s in STORAGE, y in YEAR}:
    AccumulatedNewStorageCapacity[r,s,y]+ResidualStorageCapacity[r,s,y]
    = StorageUpperLimit[r,s,y];
s.t. SI2_StorageLowerLimit{r in REGION, s in STORAGE, y in YEAR}:
    MinStorageCharge[r,s,y]*StorageUpperLimit[r,s,y] = StorageLowerLimit
    [r,s,y];
s.t. SI3_TotalNewStorage{r in REGION, s in STORAGE, y in YEAR}: sum{yy
    in YEAR: y-yy < OperationalLifeStorage[r,s] && y-yy>=0}
    NewStorageCapacity[r,s,yy]=AccumulatedNewStorageCapacity[r,s,y];
s.t. SI4_UndiscountedCapitalInvestmentStorage{r in REGION, s in STORAGE
    , y in YEAR}: CapitalCostStorage[r,s,y] * NewStorageCapacity[r,s,y]
    = CapitalInvestmentStorage[r,s,y];
s.t. SI5_DiscountingCapitalInvestmentStorage{r in REGION, s in STORAGE,
     y in YEAR}: CapitalInvestmentStorage[r,s,y]/((1+DiscountRate[r])^(y
    -min{yy in YEAR} min(yy))) = DiscountedCapitalInvestmentStorage[r,s,
    y];
s.t. SI6_SalvageValueStorageAtEndOfPeriod1{r in REGION, s in STORAGE, y
     in YEAR: (y+OperationalLifeStorage[r,s]-1) <= (max{yy in YEAR} max(
    yy))}: 0 = SalvageValueStorage[r,s,y];
s.t. SI7_SalvageValueStorageAtEndOfPeriod2{r in REGION, s in STORAGE, y
     in YEAR: (DepreciationMethod[r]=1 && (y+OperationalLifeStorage[r,s
    ]-1) > (max{yy in YEAR} max(yy)) && DiscountRate[r]=0) || (
    DepreciationMethod[r]=2 && (y+OperationalLifeStorage[r,s]-1) > (max{
    yy in YEAR} max(yy)))}: CapitalInvestmentStorage[r,s,y]*(1-(max{yy
```

```
    in YEAR} max(yy) - y+1)/OperationalLifeStorage[r,s]) =
    SalvageValueStorage[r,s,y];
184 s.t. SI8_SalvageValueStorageAtEndOfPeriod3{r in REGION, s in STORAGE, y
     in YEAR: DepreciationMethod[r]=1 && (y+OperationalLifeStorage[r,s
    ]-1) > (max{yy in YEAR} max(yy)) && DiscountRate[r]>0}:
    CapitalInvestmentStorage[r,s,y]*(1-(((1+DiscountRate[r])^(max{yy in
    YEAR} max(yy) - y+1)-1)/((1+DiscountRate[r])^OperationalLifeStorage[
    r,s]-1))) = SalvageValueStorage[r,s,y];
185 s.t. SI9_SalvageValueStorageDiscountedToStartYear{r in REGION, s in
    STORAGE, y in YEAR}: SalvageValueStorage[r,s,y]/((1+DiscountRate[r])
    ^(max{yy in YEAR} max(yy)-min{yy in YEAR} min(yy)+1)) =
    DiscountedSalvageValueStorage[r,s,y];
186 s.t. SI10_TotalDiscountedCostByStorage{r in REGION, s in STORAGE, y in
    YEAR}: DiscountedCapitalInvestmentStorage[r,s,y]-
    DiscountedSalvageValueStorage[r,s,y] = TotalDiscountedStorageCost[r,
    s,y];
187 s.t. CC1_UndiscountedCapitalInvestment{r in REGION, t in TECHNOLOGY, y
    in YEAR}: CapitalCost[r,t,y] * NewCapacity[r,t,y] =
    CapitalInvestment[r,t,y];
188 s.t. CC2_DiscountingCapitalInvestment{r in REGION, t in TECHNOLOGY, y
    in YEAR}: CapitalInvestment[r,t,y]/((1+DiscountRate[r])^(y-min{yy in
     YEAR} min(yy))) = DiscountedCapitalInvestment[r,t,y];
189 s.t. SV1_SalvageValueAtEndOfPeriod1{r in REGION, t in TECHNOLOGY, y in
    YEAR: DepreciationMethod[r]=1 && (y + OperationalLife[r,t]-1) > (max
    {yy in YEAR} max(yy)) && DiscountRate[r]>0}: SalvageValue[r,t,y] =
    CapitalCost[r,t,y]*NewCapacity[r,t,y]*(1-(((1+DiscountRate[r])^(max{
    yy in YEAR} max(yy) - y+1)-1)/((1+DiscountRate[r])^OperationalLife[r
    ,t]-1)));
190 s.t. SV2_SalvageValueAtEndOfPeriod2{r in REGION, t in TECHNOLOGY, y in
    YEAR: (DepreciationMethod[r]=1 && (y + OperationalLife[r,t]-1) > (
    max{yy in YEAR} max(yy)) && DiscountRate[r]=0) || (
    DepreciationMethod[r]=2 && (y + OperationalLife[r,t]-1) > (max{yy in
     YEAR} max(yy)))}: SalvageValue[r,t,y] = CapitalCost[r,t,y]*
    NewCapacity[r,t,y]*(1-(max{yy in YEAR} max(yy) - y+1)/
    OperationalLife[r,t]);
191 s.t. SV3_SalvageValueAtEndOfPeriod3{r in REGION, t in TECHNOLOGY, y in
    YEAR: (y + OperationalLife[r,t]-1) <= (max{yy in YEAR} max(yy))}:
    SalvageValue[r,t,y] = 0;
192 s.t. SV4_SalvageValueDiscountedToStartYear{r in REGION, t in TECHNOLOGY
    , y in YEAR}: DiscountedSalvageValue[r,t,y] = SalvageValue[r,t,y
    ]/((1+DiscountRate[r])^(1+max{yy in YEAR} max(yy)-min{yy in YEAR}
    min(yy)));
193 s.t. OC1_OperatingCostsVariable{r in REGION, t in TECHNOLOGY, l in
    TIMESLICE, y in YEAR}: sum{m in MODE_OF_OPERATION}
    TotalAnnualTechnologyActivityByMode[r,t,m,y]*VariableCost[r,t,m,y] =
     AnnualVariableOperatingCost[r,t,y];
194 s.t. OC2_OperatingCostsFixedAnnual{r in REGION, t in TECHNOLOGY, y in
    YEAR}: TotalCapacityAnnual[r,t,y]*FixedCost[r,t,y] =
    AnnualFixedOperatingCost[r,t,y];
195 s.t. OC3_OperatingCostsTotalAnnual{r in REGION, t in TECHNOLOGY, y in
    YEAR}: AnnualFixedOperatingCost[r,t,y]+AnnualVariableOperatingCost[r
    ,t,y] = OperatingCost[r,t,y];
196 s.t. OC4_DiscountedOperatingCostsTotalAnnual{r in REGION, t in
    TECHNOLOGY, y in YEAR}: OperatingCost[r,t,y]/((1+DiscountRate[r])^(y
    -min{yy in YEAR} min(yy)+0.5)) = DiscountedOperatingCost[r,t,y];
197 s.t. TDC1_TotalDiscountedCostByTechnology{r in REGION, t in TECHNOLOGY,
     y in YEAR}: DiscountedOperatingCost[r,t,y]+
    DiscountedCapitalInvestment[r,t,y]+
```

```
    DiscountedTechnologyEmissionsPenalty[r,t,y]-DiscountedSalvageValue[r
    ,t,y] = TotalDiscountedCostByTechnology[r,t,y];
198 s.t. TDC2_TotalDiscountedCost{r in REGION, y in YEAR}: sum{t in
    TECHNOLOGY} TotalDiscountedCostByTechnology[r,t,y]+sum{s in STORAGE}
     TotalDiscountedStorageCost[r,s,y] = TotalDiscountedCost[r,y];
199 s.t. TCC1_TotalAnnualMaxCapacityConstraint{r in REGION, t in TECHNOLOGY
    , y in YEAR}: TotalCapacityAnnual[r,t,y] <= TotalAnnualMaxCapacity[r
    ,t,y];
200 s.t. TCC2_TotalAnnualMinCapacityConstraint{r in REGION, t in TECHNOLOGY
    , y in YEAR: TotalAnnualMinCapacity[r,t,y]>0}: TotalCapacityAnnual[r
    ,t,y] >= TotalAnnualMinCapacity[r,t,y];
201 s.t. NCC1_TotalAnnualMaxNewCapacityConstraint{r in REGION, t in
    TECHNOLOGY, y in YEAR}: NewCapacity[r,t,y] <=
    TotalAnnualMaxCapacityInvestment[r,t,y];
202 s.t. NCC2_TotalAnnualMinNewCapacityConstraint{r in REGION, t in
    TECHNOLOGY, y in YEAR: TotalAnnualMinCapacityInvestment[r,t,y]>0}:
    NewCapacity[r,t,y] >= TotalAnnualMinCapacityInvestment[r,t,y];
203 s.t. AAC1_TotalAnnualTechnologyActivity{r in REGION, t in TECHNOLOGY, y
     in YEAR}: sum{l in TIMESLICE} RateOfTotalActivity[r,t,l,y]*
    YearSplit[l,y] = TotalTechnologyAnnualActivity[r,t,y];
204 s.t. AAC2_TotalAnnualTechnologyActivityUpperLimit{r in REGION, t in
    TECHNOLOGY, y in YEAR}: TotalTechnologyAnnualActivity[r,t,y] <=
    TotalTechnologyAnnualActivityUpperLimit[r,t,y] ;
205 s.t. AAC3_TotalAnnualTechnologyActivityLowerLimit{r in REGION, t in
    TECHNOLOGY, y in YEAR: TotalTechnologyAnnualActivityLowerLimit[r,t,y
    ]>0}: TotalTechnologyAnnualActivity[r,t,y] >=
    TotalTechnologyAnnualActivityLowerLimit[r,t,y];
206 s.t. TAC1_TotalModelHorizonTechnologyActivity{r in REGION, t in
    TECHNOLOGY}: sum{y in YEAR} TotalTechnologyAnnualActivity[r,t,y] =
    TotalTechnologyModelPeriodActivity[r,t];
207 s.t. TAC2_TotalModelHorizonTechnologyActivityUpperLimit{r in REGION, t
    in TECHNOLOGY: TotalTechnologyModelPeriodActivityUpperLimit[r,t]>0}:
     TotalTechnologyModelPeriodActivity[r,t] <=
    TotalTechnologyModelPeriodActivityUpperLimit[r,t] ;
208 s.t. TAC3_TotalModelHorizenTechnologyActivityLowerLimit{r in REGION, t
    in TECHNOLOGY: TotalTechnologyModelPeriodActivityLowerLimit[r,t]>0}:
     TotalTechnologyModelPeriodActivity[r,t] >=
    TotalTechnologyModelPeriodActivityLowerLimit[r,t] ;
209 s.t. RM1_ReserveMargin_TechnologiesIncluded_In_Activity_Units{r in
    REGION, l in TIMESLICE, y in YEAR}: sum {t in TECHNOLOGY}
    TotalCapacityAnnual[r,t,y] * ReserveMarginTagTechnology[r,t,y] *
    CapacityToActivityUnit[r,t]       =
    TotalCapacityInReserveMargin[r,y];
210 s.t. RM2_ReserveMargin_FuelsIncluded{r in REGION, l in TIMESLICE, y in
    YEAR}: sum {f in FUEL} RateOfProduction[r,l,f,y] *
    ReserveMarginTagFuel[r,f,y] = DemandNeedingReserveMargin[r,l,y];
211 s.t. RM3_ReserveMargin_Constraint{r in REGION, l in TIMESLICE, y in
    YEAR}: DemandNeedingReserveMargin[r,l,y] * ReserveMargin[r,y]<=
    TotalCapacityInReserveMargin[r,y];
212 s.t. RE1_FuelProductionByTechnologyAnnual{r in REGION, t in TECHNOLOGY,
     f in FUEL, y in YEAR}: sum{l in TIMESLICE} ProductionByTechnology[r
    ,l,t,f,y] = ProductionByTechnologyAnnual[r,t,f,y];
213 s.t. RE2_TechIncluded{r in REGION, y in YEAR}: sum{t in TECHNOLOGY, f
    in FUEL} ProductionByTechnologyAnnual[r,t,f,y]*RETagTechnology[r,t,y
    ] = TotalREProductionAnnual[r,y];
214 s.t. RE3_FuelIncluded{r in REGION, y in YEAR}: sum{l in TIMESLICE, f in
     FUEL} RateOfProduction[r,l,f,y]*YearSplit[l,y]*RETagFuel[r,f,y] =
    RETotalProductionOfTargetFuelAnnual[r,y];
```

```
s.t. RE4_EnergyConstraint{r in REGION, y in YEAR}:
   REMinProductionTarget[r,y]*RETotalProductionOfTargetFuelAnnual[r,y]
   <= TotalREProductionAnnual[r,y];
s.t. RE5_FuelUseByTechnologyAnnual{r in REGION, t in TECHNOLOGY, f in
   FUEL, y in YEAR}: sum{l in TIMESLICE} RateOfUseByTechnology[r,l,t,f,
   y]*YearSplit[l,y] = UseByTechnologyAnnual[r,t,f,y];
s.t. E1_AnnualEmissionProductionByMode{r in REGION, t in TECHNOLOGY, e
   in EMISSION, m in MODE_OF_OPERATION, y in YEAR}:
   EmissionActivityRatio[r,t,e,m,y]*TotalAnnualTechnologyActivityByMode
   [r,t,m,y]=AnnualTechnologyEmissionByMode[r,t,e,m,y];
s.t. E2_AnnualEmissionProduction{r in REGION, t in TECHNOLOGY, e in
   EMISSION, y in YEAR}: sum{m in MODE_OF_OPERATION}
   AnnualTechnologyEmissionByMode[r,t,e,m,y] = AnnualTechnologyEmission
   [r,t,e,y];
s.t. E3_EmissionsPenaltyByTechAndEmission{r in REGION, t in TECHNOLOGY,
    e in EMISSION, y in YEAR}: AnnualTechnologyEmission[r,t,e,y]*
   EmissionsPenalty[r,e,y] = AnnualTechnologyEmissionPenaltyByEmission[
   r,t,e,y];
s.t. E4_EmissionsPenaltyByTechnology{r in REGION, t in TECHNOLOGY, y in
    YEAR}: sum{e in EMISSION} AnnualTechnologyEmissionPenaltyByEmission
   [r,t,e,y] = AnnualTechnologyEmissionsPenalty[r,t,y];
s.t. E5_DiscountedEmissionsPenaltyByTechnology{r in REGION, t in
   TECHNOLOGY, y in YEAR}: AnnualTechnologyEmissionsPenalty[r,t,y]/((1+
   DiscountRate[r])^(y-min{yy in YEAR} min(yy)+0.5)) =
   DiscountedTechnologyEmissionsPenalty[r,t,y];
s.t. E6_EmissionsAccounting1{r in REGION, e in EMISSION, y in YEAR}:
   sum{t in TECHNOLOGY} AnnualTechnologyEmission[r,t,e,y] =
   AnnualEmissions[r,e,y];
s.t. E7_EmissionsAccounting2{r in REGION, e in EMISSION}: sum{y in YEAR
   } AnnualEmissions[r,e,y] = ModelPeriodEmissions[r,e]-
   ModelPeriodExogenousEmission[r,e];
s.t. E8_AnnualEmissionsLimit{r in REGION, e in EMISSION, y in YEAR}:
   AnnualEmissions[r,e,y]+AnnualExogenousEmission[r,e,y] <=
   AnnualEmissionLimit[r,e,y];
s.t. E9_ModelPeriodEmissionsLimit{r in REGION, e in EMISSION}:
   ModelPeriodEmissions[r,e] <= ModelPeriodEmissionLimit[r,e];
solve;
end;
```

9.2 Project Modules

The classes and modules developed for this project are detailed in the documentation starting on the next page.

University of Auckland
Department of Engineering Science

Global Optimisation Carbon Pricing Initiative (GOCPI) Modules

Author: Connor McDowall
Supervisor: Rosalind Archer

Monday 26th October, 2020

Table of contents

List of figures

List of tables

1 GOCPI package

1.1 Submodules

1.2 GOCPI.CreateCases module

class GOCPI.CreateCases.CreateCases
Bases: object

A class of methods to create user-defined data cases

set_accumulated_annual_demand(*accumulated_forecast*)

Sets the accumulated annual demand for fuels per region over the forecast period.
This function relies on a similar forecasting methodology as set_specific_demand.
Fuels set in this function cannot be defined in set_specific_demand.

Parameters accumulated_forecast (*float , array*) – The forecast array of size (len(region),len(fuel),len(year))

set_accumulated_fuel(*accumulated_fuel*)
Sets the case's accumulated fuel types

Parameters specified_fuel (*list*) – list of specified fuels

set_annual_emission_limit(*annual_ emission_ limits*)
Sets Annual Emission Limits

Parameters annual_emission_limits (*float , array*) – Annual Emission Limits

set_annual_exogenous_emission(*annual_ exogenous_ emission*)
Sets Annual Exogneous Emissions

Parameters annual_exogenous_emission (*float , array*) – Annual Exogenous Emissions

set_availability_factor(*availablility_ matrix*)
Sets the availability factors

Parameters availablility_matrix (*float , array*) – Matrix describing availability factors for given technologies

set_availability_technology(*availability_ technology*)
Sets the cases availability_technology type

Parameters availability_technology (*list*) – List of technologies

set_capacity_factor(*factor_ matrix*)
Sets capacity factors for conversion technologies.

Parameters `factor_matrix` (*float , array*) –

`set_capacity_of_one_technology_unit`(*capacity_ of_ one_ technology_ unit*)
Set the capacity of one technology units for all technologies

Parameters `capacity_of_one_technology_unit` (*float , array*) – capacities for one technology units

`set_capacity_technology`(*capacity_ technology*)
Sets the cases capacity_technology type

Parameters `capacity_technology` (*list*) – List of technologies

`set_capacity_to_activity_unit`(*region*, *technology*, *capacity_ dictionaries*, *override*)
Sets the capacity to activity parameter

Parameters
- `region` (*list*) – List of regions
- `technology` (*list*) – List of technologies
- `capacity_dictionaries` (*list*) – List of dictionaries to assign value
- `override` (*float , array*) –

`set_capital_cost`(*capital_ costs*)
Sets capital costs

Parameters `capital_costs` (*float , array*) – capital cost paramters

`set_capital_cost_storage`(*capital_ cost_ storage*)
Sets the capital costs of using storage technologies

Parameters `capital_cost_storage` (*float , array*) – capital cost of storage technologies

`set_conversion_ld`(*timeslice*, *daytype*, *link*)
Sets the Conversionld parameter

Parameters
- `timeslice` (*list*) – List of timeslices
- `daytype` (*list*) – List of daytypes
- `link` (*dict*) – Dictionary describing the connection between timeslices and daytypes

`set_conversion_lh`(*timeslice*, *dailytimebracket*, *link*, *override*)
Sets the Conversionlh parameter

Parameters
- `timeslice` (*list*) – List of timeslices
- `dailytimebracket` (*list*) – List of dailytimebracket
- `link` (*dict*) – Dictionary describing the connection between timeslices and dailytimebrackets
- `override` (*int , array*) – Override if want to manually put in the array

`set_conversion_ls`(*timeslice*, *season*, *link*)
Sets the Conversionls parameter

Parameters

- `timeslice` (*list*) – List of timeslices
- `season` (*list*) – List of seasons
- `link` (*dict*) – Dictionary describing the connection between timeslices and seasons

`set_daily_time_bracket(`*num_dailytimebrackets*`)`

Creates set of daily time brackets

Parameters `dailytimebracket` (*int*) – [description]

`set_day_split(`*daily_time_bracket*, *years*, *hour_split*, *num_days*, *num_hours*`)`

Sets the day split parameter

Parameters

- `daily_time_bracket` (*list*) – List of daily time brackets
- `years` (*list*) – List of year
- `hour_split` (*dict*) – Dictonary of hours in a daily time bracket
- `num_days` (*int*) – Number of days in a year
- `num_hours` (*int*) – Number of hours in a day

`set_days_in_day_type(`*season*, *daytype*, *year*, *link*, *override*`)`

Sets the DaysInDayType parameter

Parameters

- `season` (*list*) – List of seasons
- `daytype` (*list*) – List of daytypes
- `year` (*list*) – List of years
- `link` (*dict*) – Dictionary relating seasons to daytypes
- `override` (*int , array*) – Override if want to manually put in the array

`set_daytype(`*num_daytypes*`)`

[summary]

Parameters `num_daytypes` (*int*) – Number of daytypes

`set_depreciation_method(`*region*, *methods*, *override*`)`

Sets DepreciationMethod (1 = Sinking Fund Depreciation, 2 = Straightline Depreciation)

Parameters

- `region` (*list*) – List of regions
- `override` (*int , array*) – Manual array for setting depreciation methods
- `methods` (*dict*) – Dictionary assigning methods to regions

`set_discount_rate(`*equity*, *debt*, *market_index*, *cost_of_debt_pre_tax*, *risk_free_rate*, *effective_tax_rate*, *preference_equity*, *market_value_preference_shares*, *preference_dividends*, *market_risk_coefficient*`)`

[summary]

Parameters

- `equity` (*dict*) – Dictionary of equity totals from treasury balance sheets
- `debt` (*dict*) – Dictionary of equity totals from treasury balance sheets
- `market_index` (*int , array*) – Regional monthly index returns (Arrays)
- `cost_of_debt_pre_tax` (*dict*) – Dictionary of pre-tax cost of debts calculated from treasury balance sheets
- `risk_free_rate` (*dict*) – Dictionary of risk free rates from 10 year swap rates for each region
- `effective_tax_rate` (*dict*) – Dictionary of company tax rates for each region
- `preference_equity` (*dict*) – Dictionary of preference equity for each region
- `market_value_preference_shares` (*dict*) – Dictionary of the market value of prefence shares for each region
- `preference_dividends` (*dict*) – Dictionary of prefence dividends for each region
- `market_risk_coefficient` (*dict*) – Dictionary of markey risk coefficients

Returns Numpy array of discount rates

Return type [int, array]

`set_emission(`*emissions*`)`

Sets the cases emission types

Parameters `emissions` (*List*) – list of emission types

`set_emission_activity_ratio(`*emission_activity_ratios*`)`

Sets Emission Activity Ratios

Parameters `emission_activity_ratios` (*[float , array*) – Emission Activity Ratios

`set_emissions_penalty(`*emissions_penalties*`)`

Sets Emissions Penalties

Parameters `emissions_penalties` (*float , penalties*) – Emissions Penalties

`set_fixed_cost(`*fixed_costs*`)`

Set fixed costs

Parameters `fixed_costs` (*float , array*) – fixed cost parameters

`set_fuel(`*fuel*`)`

Sets the case's fuel types

Parameters `fuel` (*list*) – list of fuels

`set_input_activity_ratio(`*input_activity_ratios*`)`

Sets input activity ratios

Parameters `input_activity_ratios` (*float , array*) – Sets the input activity ratio

`set_min_storage_charge`(*minimum_storage_charges*)
Sets the minimum storage charges

Parameters `minimum_storage_charges` (*float , array*) – minimum storage parameters

`set_mode_of_operation`(*num_modes_of_operation*)
Create the number of modes of operation (n = 1,. . . ,num_modes_of_operation)

Parameters `num_modes_of_operation` (*int*) –

`set_model_period_emission_limit`(*model_period_emission_limits*)
Sets Model Period Emission Limits

Parameters `model_period_emission_limits` (*float , array*) – Model Period Emission Limits

`set_model_period_exogenous_emission`(*model_period_exogenous_emissions*)
Sets Model Period Exogenous Emissions

Parameters `model_period_exogenous_emissions` (*float , array*) – Model Period Exogenous Emissions

`set_operational_life`(*operational_lives*)
Sets operational life

Parameters `operational_lives` (*list*) –

`set_operational_life_storage`(*operational_life_storage*)
Sets the operational life storage

Parameters `operational_life_storage` (*float , array*) – operational life storage parameters

`set_output_activity_ratio`(*output_activity_ratios*)
Sets output activity ratio

Parameters `output_activity_ratios` (*float , array*) – output activity ratio parameters

`set_re_min_production_target`(*re_min_production_targets*)
Sets Renewable Energy Minimum Production Targets

Parameters `re_min_production_targets` (*float , array*) – Renewable Energy Minimum Production Targets

`set_re_tag_fuel`(*re_tag_fuels*)
Sets RE Tag Fuels

Parameters `re_tag_fuels` (*float , array*) – RE Tag Fuels

`set_re_tag_technology`(*re_tag_technologies*)
Sets RE Tag Technology

Parameters `re_tag_technologies` (*float , array*) – RE Tag Technologies

set_region(*regions*)
Sets the datacase's regions analysis

Parameters regions (*list*) – list of regions

set_reserve_margin(*reserve_margins*)
Sets reserve margins

Parameters reserve_margins (*float, array*) – Reserve Margins

set_reserve_margin_tag_fuel(*reserve_margin_fuel_tags*)
Sets the reserve margin tag fuels

Parameters reserve_margin_fuel_tags (*float, array*) – Sets the reserve margin tag fuel parameters

set_reserve_margin_tag_technology(*reserve_margin_tag_technologies*)
Sets Reserve Margin Tag Technology

Parameters reserve_margin_tag_technologies (*float, array*) – Reserve Margin Tag Technologies

set_residual_capacity(*residential_capacities*)
Set residual capacity

Parameters residential_capacities (*float, array*) – residual capacities parameter

set_residual_storage_capacity(*residual_storage_capacities*)
Sets residual storage capacities

Parameters residual_storage_capacities (*float, array*) – residual storage capacities

set_season(*num_seasons*)
Creates set of seasons

Parameters num_seasons (*int*) – Number of seasons

set_specified_annual_demand(*specified_forecast*)
Sets the annual demand for fuels per region over the forecast period (Must be accurate)

Parameters forecast (*float, array*) – The forecast array of size (len(region),len(fuel),len(year))

set_specified_demand_profile(*specified_annual_demand*, *region*, *fuel*, *year*, *timeslice*, *profile*, *override*)
Sets the specified annual demand profiles using the specified annual demand.

Parameters

- specified_annual_demand (*float, array*) – Specified annual demand profiles
- region (*list*) – List of regions
- fuel (*list*) – List of fuels
- year (*list*) – List of years
- timeslice (*list*) – List of timeslices
- profile (*Dict*) – Dictionary of fuel allocations to timeslices

- `override` (*float , array*) – Manual override for the specified annual demand profiles.

`set_specified_fuel(`*specified_fuel*`)`
Sets the case's specified fuel types

Parameters `specified_fuel` (*list*) – list of specified fuels

`set_storage(`*storage*`)`
Sets storage set of the datacase

Parameters `storage` (*list*) – list of storage types

`set_storage_level_start(`*storage_level_start*`)`
Sets the storage level starting point

Parameters `storage_level_start` (*float , array*) – storage starting level

`set_storage_max_charge_rate(`*storage_max_level_charge_rates*`)`
Sets the storgae max charge rate

Parameters `storage_max_level_charge_rates` (*float , array*) – Storage max level charge rates

`set_storage_max_discharge_rate(`*storage_max_level_discharge_rates*`)`
Sets storage technologies maximum discharge rates

Parameters `storage_max_level_discharge_rates` (*float , array*) – Discharge rates for storage paramters

`set_technology(`*technology*`)`
Sets the cases technology type

Parameters `technology` (*list*) – List of technologies

`set_technology_from_storage(`*technology_from_storage*`)`
Sets technology from storage binary paramter

Parameters `technology_from_storage` (*float , array*) – technology from storage parameter

`set_technology_to_storage(`*technology_to_storage*`)`
Sets the technology to storage parameter

Parameters `technology_to_storage` (*float , array*) – technology to storage parameter

`set_timeslice(`*timeslice*`)`
Set of timeslices

Parameters `timeslice` (*list*) – list of timeslices

`set_total_annual_max_capacity(`*total_annual_max_capacities*`)`
Sets the total annual maximum capacities

Parameters `total_annual_max_capacities` (*float , array*) – Total Annual Max Capacities

set_total_annual_min_capacity(*total_annual_min_capacities*)
Sets the totoal annual minimum capacities

Parameters total_annual_min_capacities (*float , array*) – Total Annual Min Capacities

set_total_technology_annual_activity_lower_limit(*total_technology_activity_lower_limits*)
Sets the Total Technology Activity Lower Limits

Parameters total_technology_activity_lower_limits (*float , array*) – Technology Activity Lower Limits

set_total_technology_annual_activity_upper_limit(*total_technology_annual_activity_upper_*
Sets the Total Technology Activity Upper Limits

Parameters total_technology_annual_activity_upper_limits (*float , array*) – Technology Activity Upper Limits

set_total_technology_period_activity_lower_limit(*total_technology_period_activity_lower_*
Sets Total Technology Period Activity Lower Limits

Parameters total_technology_period_activity_lower_limits (*[type]*) – Total Technology Period Activity Lower Limit

set_total_technology_period_activity_upper_limit(*total_technology_period_activity_upper_*
Sets Total Technology Period Activity Upper Limits

Parameters total_technology_period_activity_upper_limits (*float , array*) – Total Technology Period Activity Upper Limit

set_trade_route(*trade*)

Sets the TradeRoute parameter between regions (Assume it is the same across fuels and years)

Parameters trade (*int , array*) – 4D array representing trade relationships between regions, fuels and years. You must model this manually.

set_variable_cost(*variable_costs*)
Sets variable costs

Parameters variable_costs (*float , array*) – variable costs parameters

set_year(*start_year*, *end_year*, *interval*)
Sets a list of forecast years

Parameters

- start_year (*int*) – Starting year for forecasting (Less than end_year)
- end_year (*int*) – Ending year for forecasting (Greater than start_year)
- interval (*int*) – Gap for forecasting period

set_year_split(*timeslices*, *years*, *splits*)

Creates 2D Numpy Array Parameter Splits. (Note: The index positions of timelices and splits must match)

Parameters

- `timeslices` (*list*) – List of timeslices
- `years` (*list*) – List of years
- `splits` (*dict*) – A dictionary linking yearsplits to timeslices

1.3 GOCPI.Energysystems module

`class GOCPI.Energysystems.Energy_Systems(`*year*, *region*, *emission*, *technology*, *capacity_technology*, *availability_technology*, *fuel*, *specified_fuel*, *accumulated_fuel*, *timeslice*, *mode_of_operation*, *storage*, *daytype*, *season*, *dailytimebracket*`)`

Bases: `object`

A class of methods to initialise energy sytems and create the data/model files needed for optimisation.

`create_data_file(`*file_location*, *defaults_dictionary*, *toggle_defaults*`)`

Creates the osemosys datafile

Parameters

- `file_location` (*str*) – String of directory to save data file
- `defaults_dictionary` (*dict*) – Dictionary setting the default values for parameters
- `toggle_defaults` (*Bool*) – Boolean (True/False to only print the default functions

`create_model_file(`*root*, *file*`)`

Creates the model file necessary for the project to run

Parameters `for the basic problem` (*Parameters*) –
Returns The loaded in parameters and sets

`load_datacase(`*case*, *system*`)`

Loads the data case to a correct configured and intialised energy system

(The load status dictionary must be compatible with the data_case and system_case)

Parameters

- `case` (*object*) – Energy system datacase
- `system` (*object*) – Initialised energy system
- `load_status` (*dict*) – Dictionary setting the required sets and parameters to load

Returns Returns the updated dictionary
Return type system_case (dict)

1.4 GOCPI.Forecasting module

class GOCPI.Forecasting.Forecasting

Bases: object

calculate_cagr_forecasts(*cagr_dictionary*, *base_year_dictionary*, *fuel*, *year*)

Forecasts base year fuels by a constant average growth rate for a forecast period

Parameters

- cagr_dictionary (*Dict*) – Dictionary of constant average growth rates per fuel
- base_year_dictionary (*[type]*) – Dictionary of base year fuel consumption in energy types
- fuel (*list*) – List of Fuels
- year (*list*) – List of forecast years

Returns 2D Array of demand forecasts per fuel

Return type [float, array]

calculate_constant_average_growth_rate(*start_year*, *end_year*, *start_value*, *end_value*)

Calculates the constant average growth rate (CAGR)

Parameters

- start_year (*int*) – Starting year
- end_year (*int*) – Ending year
- start_value (*int*) – Initial value
- end_value (*int*) – Final value

Returns Constant average growth rate (1+ decimal)

Return type cagr

energy_balance_base(*root*, *IEA_World_Energy_Balances_1*, *IEA_World_Energy_Balances_2*, *create_excel_spreadsheet*, *output_file*)

Creates the baseline energy balance for forecasting

Parameters

- root (*path*) – Path to provide access to all the files
- IEA_World_Energy_Balances_1 (*str*) – File name for Energy Balance A to K
- IEA_World_Energy_Balances_2 (*[type]*) – File name for Energy Balance L to Z
- create_excel_spreadsheet (*boolean*) – True/false on whether to create a spreadsheet
- output_file (*str*) – Name of output energy balance spreadsheet

Returns Dictionary of energy balances and unique lists (Use these key words to access: Energy Balances, Fuel, Geography, Technology)

Return type (dict)

1.5 GOCPI.Navigation module

class GOCPI.Navigation.Navigation(*target_root*, *target_file*)

Bases: object

Navigation is a class for navigating, manipulating and editing data in the GOCPI model.

Find_File

Type string

TODO: Fill out all functions below

Find_File()

Find_File searches for a target file, from a base directory, to construct a target directory.

Inputs: target_root = The base directory to search from (string). target_file = The name of the target file (string).

Outputs: f = Combinated target file location (string).

create_linear_programme_file(*directory*, *data_file*, *model_file*, *output_file*)

Creates the model file through executing model system commands

Parameters

- **directory** (*str*) – Name of directory to put data into
- **data_file** (*str*) – Name of energy system data file
- **model_file** (*str*) – Name of energy system model file
- **output_file** (*str*) – Name of output linear programme

1.6 GOCPI.Optimisation module

class GOCPI.Optimisation.Optimisation

Bases: object

Prepare and runs optimisation with IBM ILOG CPLEX Optimisation Studio

create_linear_programme_file(*directory*, *data_file*, *model_file*, *output_file*)

Creates the model file through executing model system commands

Parameters

- **directory** (*str*) – Name of directory to put data into
- **data_file** (*str*) – Name of energy system data file
- **model_file** (*str*) – Name of energy system model file
- **output_file** (*str*) – Name of output linear programme

reset(*tarinfo*)

Resets the tarfile information when creating tar files This is to input into the filter when using tar.add()

Parameters **tarinfo** (*Object*) – Tar Object containing an ID of 0 and the root as the name

Returns Tar Object containing an ID of 0 and the root as the name
Return type tarinfo (Object)

`run_cplex_local(`*model_file*`)`
This function runs cplex on the local device if the energy system is of a small enough complexity

`run_ibm_wml_do(`*apikey*, *url*, *deployment_space_name*, *cloud_object_storage_credential*, *service_instance_id*, *deployment_space_exists*, *data_assets_exist*, *data_asset_dictionary*, *model_name*, *model_type*, *model_runtime_uid*, *model_tar_file*, *num_nodes*, *deployment_exists*, *payload_input_data_id*, *payload_input_data_file*, *payload_output_data_id*`)`

This function enables the user to solve python-based optimisation models. The legacy offering to solve optimisation models on IBM cloud was using the docplex python api to run Cplex on DOcloud. As of September 2020, the DOcloud was discontinued with Decision Optimisation functionalities imported to IBM's Watson Machine Learning Service. The new process requires the energy system model to be written in python. This project saw the implementation of the osemosys modelling methodology in GNU Mathprog written into LP Files. IBM Decision Optimisation in cannot deploy models in LP File formats to get jobs. Therefore, this function is for future work in converting the entire energy system modelling tool to python-based only. This is well-documented the report in the Future Work Section. Note: You must have access to IBM Watson Studio and Cloud Products through the IBM Academic Initiative or Similar.

Parameters

- `apikey` (*str*) – API key from user's IBM Cloud Account
- `url` (*[type]*) – URL for the server the user is using for the IBM services
- `deployment_space_name` (*str*) – Name of the deployment space
- `cloud_object_storage_credential` (*str*) – Credential for the cloud object storage asset
- `service_instance_id` (*str*) – Service instance id for the service being used (IBM WML)
- `deployment_space_exists` (*boolean*) – True/False if the deployment space already exists
- `data_assets_exist` (*boolean*) – True/False if the data assets (e.g. input data stored on cloud)
- `data_asset_dictionary` (*dict*) – A dictionary of data assets to stored on IBM cloud
- `model_name` (*str*) – Name of the model
- `model_type` (*str*) – Name of the model
- `model_runtime_uid` (*str*) – Runtime ID for the model
- `model_tar_file` (*tar*) – Tar file containing the python model
- `num_nodes` (*int*) – Number of nodes the model is run off.
- `deployment_exists` (*boolean*) – True/False if the deployment already exists
- `payload_input_data_id` (*str*) – Name of input data

- `payload_input_data_file` (*dataframe*) – Input data file in the form of a dataframe
- `payload_output_data_id` (*str*) – Name of output data file

`use_bash_shell(`*command*`)`

Execute bash commands in python scripts

Parameters `command` (*str*) – Command to execute

1.7 Module contents

Indices and tables

- genindex
- modindex
- search

Python Module Index

g

Index

U