

ENGSCI 700A/B

Research Compendium

Connor McDowall

*cmcd398
530913386*

November 3, 2020

Declaration of Contribution

I proposed this project. I am the sole contributor.

Contents of Package

Contents

1	Important Project Documents	3
1.1	Data	3
1.2	Documents	3
1.3	Source Files	3
1.4	Additional	3
2	Website	4
2.1	Jekyll, Markdown, Ruby, GitHub Pages	4
2.2	W3 Schools Adaptation	4
3	Project Log Book	6
4	GOCPI Documentation	12
5	Programming	32
5.1	GOCPI NZ Energy Systems Example	32
5.2	GOCPI Module	47
5.2.1	Navigation	47
5.2.2	Energysystems	48
5.2.3	CreateCases	90
5.2.4	Forecasting	105
5.2.5	Optimisation	108
6	Development Scripting	114
6.1	GOCPI Data Cases	114
6.2	GOCPI Energy Balances	116
6.3	GOCPI Geographies	123
6.4	GOCPI Inputs	125
6.5	GOCPI Model Import	127
6.6	GOCPI Optimisation	129
7	OseMOSYS	136
7.1	Model File	136
7.2	Data File	145
7.3	Linear Programme File	353
8	Bibliography	354

List of Figures

1	GOCPI Website V1 Cover Page	4
---	---------------------------------------	---

2	GOCPI Website V1 Project Information	4
3	GOCPI Website V1 Project Access	5
4	GOCPI Website V1 Project Contributors	5
5	GOCPI Website V1 Project Documents	5

1 Important Project Documents

All files related to the project are stored within a GitHub Repository.

This repository is accessible here and summarised as follows:

1.1 Data

- Energy Balances from the International Energy Agency (IEA) csv files to create energy balances for user defined energy systems. Source: International Energy Agency
- Australian and New Zealand Energy Statistics to build a New Zealand and Australian reference energy systems. Source: MBIE and The Australian Government.
- Sets of geographies for partitioning and creating regions to model user defined energy systems.
- Images for website design.
- Input data sources for TIMES and OseMOSYS modelling and developing user defined energy systems.
- GNU Mathprog, GAMS and Pyomo versions of OseMOSYS forked from the OseMOSYS GitHub Repository.
- The TIMES model and twelve demo versions of varying complexity.

1.2 Documents

- Notes related to the project.
- Presentations related to the project.
- Research Compendium, Project Report and all images, figures and sources used in these documents.
- Academic journals, articles and technical reports related to the literature review.

1.3 Source Files

- GOCPI package for creating, formulating, forecasting and solving user-defined energy systems.
- Processing and development scripts used to design the GOCPI package.
- The prototype webpage adapted from a W3 school template.
- Git, IBM, Microsoft Visual Studio Code configurations/formatting related files.

1.4 Additional

- Build files for producing GOCPI documentation using sphinx.

2 Website

2.1 Jekyll, Markdown, Ruby, GitHub Pages

The website was improved using Jekyll, Ruby and Markdown technologies. The website includes links to documentation required to build systems and software required. The repository used to build the website is separate from the GOCPI repository. **You can access the website here**

2.2 W3 Schools Adaptation

This iteration was adapted from an W3 schools template to create a prototype website for the project. This prototype describes key information related the project.

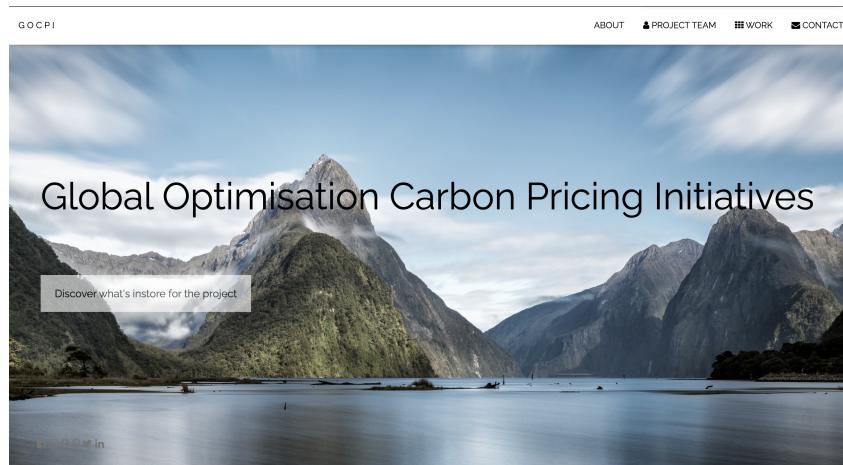


Figure 1: GOCPI Website V1 Cover Page

Model	Purpose	Output	System
The model adopts The Integrated Market EFORM System (TIMES) Model to model the entire energy system for a given geography	Communicate the urgency of addressing climate change while informing carbon pricing policy and sustainable investment strategies	Geography specific services demand profiles and energy system recommendations	The model is written in GAMS with supporting functionality provided by Python, HTML, CSS and JavaScript

Figure 2: GOCPI Website V1 Project Information

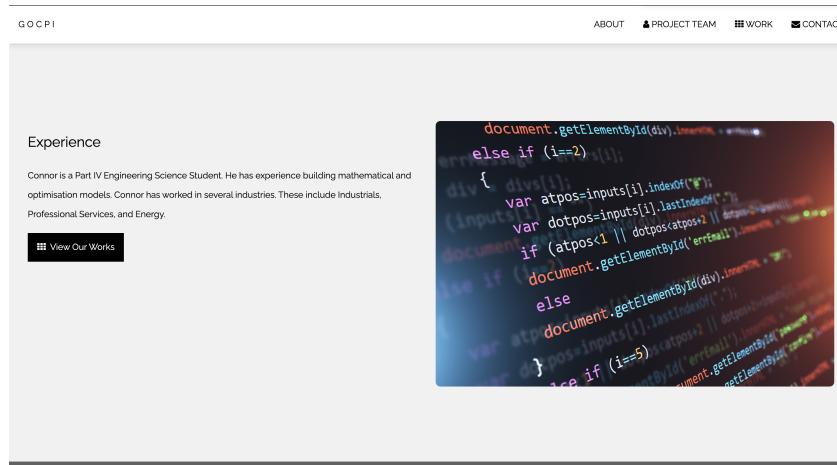


Figure 3: GOCPI Website V1 Project Access

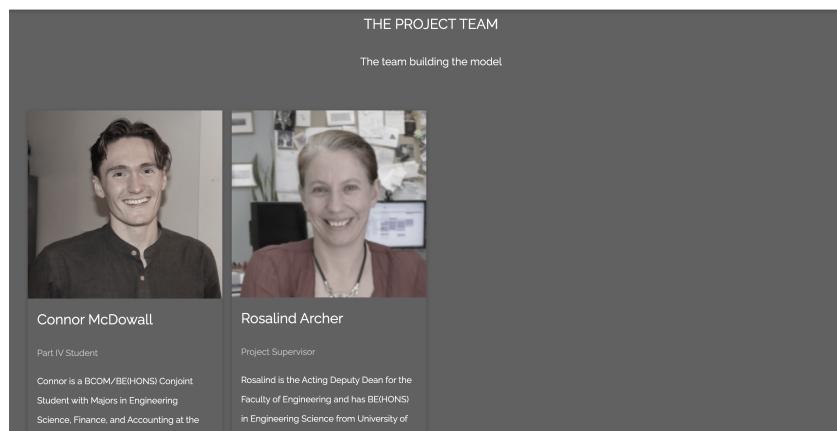


Figure 4: GOCPI Website V1 Project Contributors

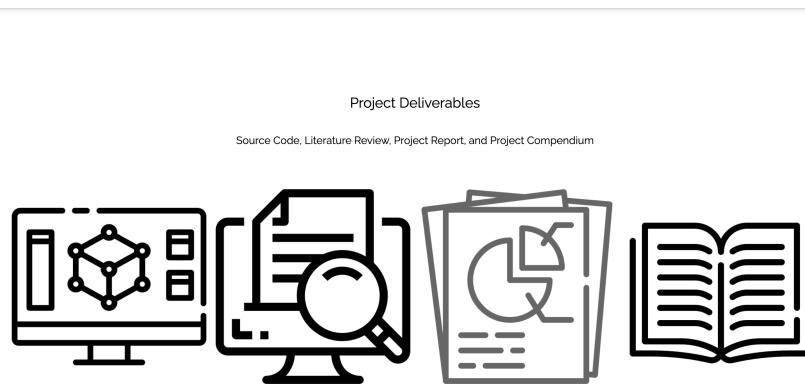


Figure 5: GOCPI Website V1 Project Documents

3 Project Log Book

Disclaimer: Contributions to the Project Log Book grew inconsistent toward the later stages of the project.

January - February

- Began scoping energy related project during experience in the Commercial team at ExxonMobil Australia
- Emailed and Meet with Rosalind
- Decided to look at Carbon Pricing Initiatives to inform reinvestment and carbon pricing initiatives
- Rosalind tasked with investigating GAMS

March 1st - May 30th

- Coronavirus was classified a worldwide pandemic
- New Zealand was sent into lockdown
- Researched 30+ Academic reports, articles, websites for Literature Review
- Wrote 10 page Literature Review
- Scoped the project
- Submitted Mid-Semester Literature Review on May 5th
- Installed GAMS on my local device
- Began researching the construction of an energy system with Excel, VEDA FE, GAMS, VEDA BE, Python
- Created GOCPI Geographies.gyp script to combined cities, countries and continents while providing granularity to the modelling process
- Created GOCPI.html as a project display for selling the project
- Ran into a series of installation and usage issues with VEDA and GAMS
- Requested VM to work from home
- Installed VMware and GAMS on FlexIT systems
- Faced GAMS Licensing issues on FlexIT

May 31st 2020

1. Installed Microsoft Remote Desktop and FortiClient VPN to access UoA Virtual Machine
2. Set up Virtual Machine

June 1st 2020

1. Installed VEDA FE and VEDA FE on Virtual Machine
2. Downloaded 12 Demo Models to build my TIMES Model

June 3rd

1. Begun testing the Model the Demo Models

June 4th - June 10th

1. Meeting with Rosalind. Discussed set up and action points moving forward.
2. Showed VEDA-FE. Four assessments were discussed.
3. Continued researching how to use VEDA

June 11th - Approximately 4 hours

1. Meeting with Rosalind at 10:30am via Zoom
2. Discussed action points moving forward.
3. Continued to adapt excel spreadsheets for Excel Data.
4. There is still an issue with GAMS Installation (Check with Tony. He knows a guy)
5. VEDA FE creates the necessary DD files. Continue to work through the DEMO Models to understand GAMS.

June 16th - July 1st

- No Progress - Study Break and Exams for ACCTG 371, FINANCE 362 and EN-GSCI 711

July 2nd

- Last meeting in Rosalind's corner office. Discussed online exams, Chegg, cheating and project next steps.
- Agreed to adapt spreadsheets for user input and use BP's World Energy Outlook Statistics to determine production, conversion and consumption rates.

July 3rd

- Began adapting Demo 12 model for custom inputs
- Began using the openpyxl python library to manipulate excel (GOCPI Input.gyp)

July 4rd

- Continue to work on openpyxl adaptation with xls and xlsx excel sheets

July 6th

- Created a proper file directory for managing the project
- Continued to adapt GOCPI Inputs.gyp to scale across multiple sheets
- Adapted GOCPI.html, GOCPI Inputs.gyp and GOCPI Geographies to work after rearranging the geographies
- Nearly had a heart attack as I was led to believe issues with Github and Git meant I deleted my entire project
- Recovered entire project and reports

July 7th

- Worked on file manipulation in Google Drive via Google Cloud APIs
- Discovered IEA Energy Balances on stats.OECD.org via Uni library databases
- Found 20GB csv on Energy Balances data
- Processed 20GB csv to create two 80MB csv for 2017 energy balance data using Microsoft Access

July 7th

- Developed and resolved issues relating to git and Github
- Developed processing methods for Energy Balance statistics using pandas pivot table function

July 17th

- Meeting with Kiti (NZ TIMES Energy Modeler)
- Discuss constraints associated TIMES and GAMS modelling
- Introduced to OseMOSYS (Open Source, Energy Modelling Tool)
- Introduced to MBIE,EECA (<https://www.eeca.govt.nz/>)
- Agreed to explore OseMOSYS and alternative datasources to build an alternative product.
- Agreed to keep Kiti updated on project process moving forward.

July 18th

- Downloaded MBIE Energy
- Research OseMOSYS energy modelling Approach
- Downloaded OseMOSYS energy modelling tools
- Tested Pyomo, GNU and GAMS approaches. GNU optimised using glpsol in conda environment. Progress works well.

- Decision: Move away from TIMES/GAMS modelling to using Osemosys.
- Began Scripting Sheet to generate model input text file

July 19th

- Created excel spreadsheet to store OseMOSYS energy model inputs
- Began adapting sets, parameters, variables, equations and constraints to excel template.
- Researched more about OseMOSYS

July 20th

- Continued to adapt 200+ lines of model code in the excel templates

July 21st

- Learned to create custom python packages.
- Began working on adjustable sets

July 22nd

-

July 23rd

- Productive meeting with Rosalind, showed model output. (Rosalind said progress was really exciting)

July 24th

- Continued creating a custom package for the GCOPI module.

July 25th

- Started GOCPI module to create scalable data files

July 26th

- Continued to adapt GOCPI custom package to create scalable data files (Completed)

July 27th

- Edited report headings and created a structure for the Research Report.

July 28th

- Investigated CPLEX Solvers
- Registered for the IBM Academic Initiative
- Downloaded and Installed IBM ILOG CPLEX Optimizer Studio
- Installed cplex and docplex Python APIs from the IBM ILOG CPLEX Optimizer Studio
- Added create model file model to GOCPI

July 30th - August 9th

- Spent a day fixing git commit and push issues
- Installed GIT LFS and the functionality of .gitignore to prevent the committing .mp4 and .lp files
- Installed yapf in Microsoft Visual Studio Code to enable PEP-8 Autoformatting
- Wrote 4.5 pages for the technical, mainly focusing on the setup of Python, Anaconda, CPLEX, Git, GitHub, folder structure suggested by Wilson et al and the OseMOSYS methodology.
- Submitted the 4-6 page technical report.
- Created presentation structure

August 10th

- Drafted and submitted four slide summary for presentation.
- Recorded and submitted 5 minute presentation

August 12th

- Lockdown and Became Ill
- Went and got COVID-19 Testing (Stood in Queue for 4.5 hours)

August 13th

- Very productive meeting with Rosalind
- Discussed project process, presentation and mid-year technical report
- Continuing doing what I am doing.
- Continued developing NZ Example
- Abandoned developing the NZ Example as faced severe limitations
- Continued developing the Navigation, Forecasting, Energysystems and CreateCases modules.

September 2nd - September 30th

- IBM Cloud Installation and Application.
- Discussed project process, presentation and mid-year technical report
- Investigated adopting DOCPLEX optimisation technologies.
- Discovered limitations in the IBM Decision Optimisation service. This was no longer viable as imported to IBM Watson Machine Learning service.
- Began exploring the implementation of the IBM Watson Machine Learning service to engage with this pipeline.
- Developed the optimisation module to use

October 1st - October 29th

- Systems week interfered with the construction of the report.
- Wrote the report
- Edited the report
- Reviewed the report
- Had three productive meetings with my supervisor about the report.

October 30th

- Submitted the final report

4 GOCPI Documentation

The GOCPI classes and modules are described in the following document. This document was generated from docstrings using sphinx.

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

Table of contents	i
List of figures	i
List of tables	i
1 GOCPI package	1
1.1 Submodules	1
1.2 GOCPI.CreateCases module	1
1.3 GOCPI.Energysystems module	9
1.4 GOCPI.Forecasting module	10
1.5 GOCPI.Navigation module	11
1.6 GOCPI.Optimisation module	11
1.7 Module contents	13
Python Module Index	14
Index	15

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 Exogenous Emissions

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

```
set_availability_factor(availability_matrix)
```

Sets the availability factors

Parameters *availability_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*) – Dictionary 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 debt 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 preference shares for each region
- **preference_dividends** (*dict*) – Dictionary of preference dividends for each region
- **market_risk_coefficient** (*dict*) – Dictionary of market risk coefficients

Returns Numpy array of discount rates

Return type [int, array]

set_emission(*emissions*)

Sets the case's 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 stor-
            age 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 Technolo-
            gies

```

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 storage 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 parameters

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 parameter

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 total 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_limits)
```

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_limits)
```

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_limits)
```

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 timeslices 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 systems 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 initialised 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 = Combined 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

GOCPI, [13](#)

GOCPI.CreateCases, [1](#)

GOCPI.Energysystems, [9](#)

GOCPI.Forecasting, [10](#)

GOCPI.Navigation, [11](#)

GOCPI.Optimisation, [11](#)

Index

C

calculate_cagr_forecasts()
(*GOCPI.Forecasting.Forecasting method*), 10
calculate_constant_average_growth_rate()
(*GOCPI.Forecasting.Forecasting method*), 10
create_data_file()
(*GOCPI.Energysystems.Energy_Systems method*), 9
create_linear_programme_file()
(*GOCPI.Navigation.Navigation method*), 11
create_linear_programme_file()
(*GOCPI.Optimisation.Optimisation method*), 11
create_model_file()
(*GOCPI.Energysystems.Energy_Systems method*), 9
CreateCases (*class in GOCPI.CreateCases*), 1

E

energy_balance_base()
(*GOCPI.Forecasting.Forecasting method*), 10

Energy_Systems (*class in GOCPI.Energysystems*), 9

F

Find_File (*GOCPI.Navigation.Navigation attribute*), 11
Find_File() (*GOCPI.Navigation.Navigation method*), 11

Forecasting (*class in GOCPI.Forecasting*), 10

G

GOCPI
module, 13
GOCPI.CreateCases
module, 1
GOCPI.Energysystems
module, 9
GOCPI.Forecasting
module, 10
GOCPI.Navigation
module, 11

GOCPI.Optimisation
module, 11

L
load_datacase()
(*GOCPI.Energysystems.Energy_Systems method*), 9

M

module
GOCPI, 13
GOCPI.CreateCases, 1
GOCPI.Energysystems, 9
GOCPI.Forecasting, 10
GOCPI.Navigation, 11
GOCPI.Optimisation, 11

N

Navigation (*class in GOCPI.Navigation*), 11

O

Optimisation (*class in GOCPI.Optimisation*), 11

R

reset() (*GOCPI.Optimisation.Optimisation method*), 11
run_cplex_local()
(*GOCPI.Optimisation.Optimisation method*), 12
run_ibm_wml_do()
(*GOCPI.Optimisation.Optimisation method*), 12

S

set_accumulated_annual_demand()
(*GOCPI.CreateCases.CreateCases method*), 1
set_accumulated_fuel()
(*GOCPI.CreateCases.CreateCases method*), 1
set_annual_emission_limit()
(*GOCPI.CreateCases.CreateCases method*), 1
set_annual_exogenous_emission()
(*GOCPI.CreateCases.CreateCases method*), 1

```

set_availability_factor()
    (GOCPI.CreateCases.CreateCases
     method), 1
set_availability_technology()
    (GOCPI.CreateCases.CreateCases
     method), 1
set_capacity_factor()
    (GOCPI.CreateCases.CreateCases
     method), 1
set_capacity_of_one_technology_unit()
    (GOCPI.CreateCases.CreateCases
     method), 2
set_capacity_technology()
    (GOCPI.CreateCases.CreateCases
     method), 2
set_capacity_to_activity_unit()
    (GOCPI.CreateCases.CreateCases
     method), 2
set_capital_cost()
    (GOCPI.CreateCases.CreateCases
     method), 2
set_capital_cost_storage()
    (GOCPI.CreateCases.CreateCases
     method), 2
set_conversion_ld()
    (GOCPI.CreateCases.CreateCases
     method), 2
set_conversion_lh()
    (GOCPI.CreateCases.CreateCases
     method), 2
set_conversion_ls()
    (GOCPI.CreateCases.CreateCases
     method), 2
set_daily_time_bracket()
    (GOCPI.CreateCases.CreateCases
     method), 3
set_day_split()
    (GOCPI.CreateCases.CreateCases
     method), 3
set_days_in_day_type()
    (GOCPI.CreateCases.CreateCases
     method), 3
set_daytype() (GOCPI.CreateCases.CreateCases
     method), 3
set_depreciation_method()
    (GOCPI.CreateCases.CreateCases
     method), 3
set_discount_rate()
    (GOCPI.CreateCases.CreateCases
     method), 3
set_emission()
    (GOCPI.CreateCases.CreateCases
     method), 4
set_emission_activity_ratio()
    (GOCPI.CreateCases.CreateCases
     method), 4
set_emissions_penalty()
    (GOCPI.CreateCases.CreateCases
     method), 4
set_fixed_cost()
    (GOCPI.CreateCases.CreateCases
     method), 4
set_fuel() (GOCPI.CreateCases.CreateCases
     method), 4
set_input_activity_ratio()
    (GOCPI.CreateCases.CreateCases
     method), 4
set_min_storage_charge()
    (GOCPI.CreateCases.CreateCases
     method), 4
set_mode_of_operation()
    (GOCPI.CreateCases.CreateCases
     method), 5
set_model_period_emission_limit()
    (GOCPI.CreateCases.CreateCases
     method), 5
set_model_period_exogenous_emission()
    (GOCPI.CreateCases.CreateCases
     method), 5
set_operational_life()
    (GOCPI.CreateCases.CreateCases
     method), 5
set_operational_life_storage()
    (GOCPI.CreateCases.CreateCases
     method), 5
set_output_activity_ratio()
    (GOCPI.CreateCases.CreateCases
     method), 5
set_re_min_production_target()
    (GOCPI.CreateCases.CreateCases
     method), 5
set_re_tag_fuel()
    (GOCPI.CreateCases.CreateCases
     method), 5
set_re_tag_technology()
    (GOCPI.CreateCases.CreateCases
     method), 5

```

```

set_region() (GOCPI.CreateCases.CreateCases
    method), 5
set_reserve_margin()
    (GOCPI.CreateCases.CreateCases
        method), 6
set_reserve_margin_tag_fuel()
    (GOCPI.CreateCases.CreateCases
        method), 6
set_reserve_margin_tag_technology()
    (GOCPI.CreateCases.CreateCases
        method), 6
set_residual_capacity()
    (GOCPI.CreateCases.CreateCases
        method), 6
set_residual_storage_capacity()
    (GOCPI.CreateCases.CreateCases
        method), 6
set_season() (GOCPI.CreateCases.CreateCases
    method), 6
set_specified_annual_demand()
    (GOCPI.CreateCases.CreateCases
        method), 6
set_specified_demand_profile()
    (GOCPI.CreateCases.CreateCases
        method), 6
set_specified_fuel()
    (GOCPI.CreateCases.CreateCases
        method), 7
set_storage() (GOCPI.CreateCases.CreateCases
    method), 7
set_storage_level_start()
    (GOCPI.CreateCases.CreateCases
        method), 7
set_storage_max_charge_rate()
    (GOCPI.CreateCases.CreateCases
        method), 7
set_storage_max_discharge_rate()
    (GOCPI.CreateCases.CreateCases
        method), 7
set_technology()
    (GOCPI.CreateCases.CreateCases
        method), 7
set_technology_from_storage()
    (GOCPI.CreateCases.CreateCases
        method), 7
set_technology_to_storage()
    (GOCPI.CreateCases.CreateCases
        method), 7
set_timeslice()
    (GOCPI.CreateCases.CreateCases
        method), 7
set_total_annual_max_capacity()
    (GOCPI.CreateCases.CreateCases
        method), 7
set_total_annual_min_capacity()
    (GOCPI.CreateCases.CreateCases
        method), 7
set_total_technology_annual_activity_lower_limit
    (GOCPI.CreateCases.CreateCases
        method), 8
set_total_technology_annual_activity_upper_limit
    (GOCPI.CreateCases.CreateCases
        method), 8
set_total_technology_period_activity_lower_limit
    (GOCPI.CreateCases.CreateCases
        method), 8
set_total_technology_period_activity_upper_limit
    (GOCPI.CreateCases.CreateCases
        method), 8
set_trade_route()
    (GOCPI.CreateCases.CreateCases
        method), 8
set_variable_cost()
    (GOCPI.CreateCases.CreateCases
        method), 8
set_year() (GOCPI.CreateCases.CreateCases
    method), 8
set_year_split()
    (GOCPI.CreateCases.CreateCases
        method), 8

```

U

```

use_bash_shell()
    (GOCPI.Optimisation.Optimisation
        method), 13

```

5 Programming

These section contains all programming scripts for the project.

5.1 GOCPI NZ Energy Systems Example

The GOCPI NZ Example file is the Python file

```
1 # GOCPI_NZ_Example.gyp is an exemplar script in how to build a
2 # data case for the Model
3
4 #
5 ##########
6 # This is a major input script for creating data files.
7 #
8 ##########
9 # Import all necessary python packages
10 import numpy as np
11 import pandas as pd
12 import matplotlib.pyplot as plt
13 import scipy as sc
14 import sklearn as skl
15 import csv as csv
16 import openpyxl
17 import pathlib
18 import os
19 from pathlib import Path
20 from openpyxl import load_workbook
21 import GOCPI as GF
22 import cplex as cp
23 import docplex as dp
24
25 # Sets sets (All must be one word)
26 # Creates a New Zealand Energy System Scenario using the CreateCases
27 # Module
28 nz_energy_system = GF.CreateCases()
29
30 # Set Definitions
31 #
32 ##########
33 #
34 ##########
35 # Defines the forecast period
36 nz_energy_system.set_year(2020, 2030, 1)
37
38 # Defines the regions
39 REGION = ['NEWZEALAND', 'AUSTRALIA']
40 nz_energy_system.set_region(REGION)
41
42 # Defines the Emissions
43 EMISSION = ['CO2', 'NOX', 'CO', 'METHANE']
44 nz_energy_system.set_emission(EMISSION)
45
```

```

42 # Technology
43 #
44 # ##########
45 # Defines the technology set (MBIE Energy Statistics Energy Supply and
46 # Demand -
47 Production = [
48     'Indigenous_Production', 'Imports', 'Exports', 'Stock_Change',
49     'International_Transport',
50 ]
51 Conversion = [
52     'Electricity_Generation', 'Cogeneration', 'Fuel_Production',
53     'Other_Transformation', 'Losses_and_Own_Use'
54 ]
55 Non_Energy = ['Non_Energy_Use']
56 Consumption = [
57     'Agriculture', 'Forestry_and_Logging', 'Fishing', 'Mining',
58     'Food_Processing', 'Textiles', 'Wood_Pulp_Paper_and_Printing', ,
59     'Chemicals',
60     'Non_Metallic_Minerals', 'Basic_Metals',
61     'Mechanical_Electrical_Equipment', 'Building_and_Construction',
62     'Unallocated', 'Commercial', 'Transport', 'Residential'
63 ]
64 Statistical_Differences = ['Statistical_Differences']
65 TECHNOLOGY_ALL = [
66     Production, Conversion, Non_Energy, Consumption,
67     Statistical_Differences
68 ]
69 TECHNOLOGY = []
70 for tech in TECHNOLOGY_ALL:
71     for i in range(0, len(tech), 1):
72         TECHNOLOGY.append(tech[i])
73
74 # Sets the technology set
75 nz_energy_system.set_technology(TECHNOLOGY)
76
77 # Sets capacity technologies for energy production
78 CAPACITY_TECHNOLOGY = Conversion
79 CONSUMPTION_TECHNOLOGY = Consumption
80 nz_energy_system.set_capacity_technology(TECHNOLOGY)
81 nz_energy_system.set_availability_technology(TECHNOLOGY)
82 # Sets the Conversion Sets
83
84 # ##########
85 # Calculates Energy Balances Base Year
86 #
87 # ##########
88 # Sets names for the energy balance sheets
89 NZ_energy_balances = GF.Forecasting()
90 root_energy_balance = pathlib.Path(

```

```

89     '/Users/connor/Google Drive/Documents/University/Courses/2020/
90     ENGSCI 700A&B/GOCPI/data/Energy Balances'
91 )
92 IEA_World_Energy_Balances_A2K = 'IEAWorldEnergyBalances2017A-K.csv'
93 IEA_World_Energy_Balances_L2Z = 'IEAWorldEnergyBalances2017L-Z.csv'
94 create_excel_spreadsheet = True
95 output_file = "Geo EB.xlsx"
96
97 # Creates the geography dataframe
98 outputs = NZ_energy_balances.energy_balance_base(
99     root_energy_balance, IEA_World_Energy_Balances_A2K,
100    IEA_World_Energy_Balances_L2Z, create_excel_spreadsheet,
101    output_file)
102
103 #
104 ##########
105 # Calculates Fuels
106 #
107 #########
108 # Defines the fuel set (MBIE Energy Statistics Energy Supply and Demand
109 # - Gross PJ (Higher Heating Value))
110 Coal = ['Bituminous', 'Sub_Bituminous', 'Lignite']
111 Oil = [
112     'Crude_Feedstocks_NGL', 'LPG', 'Petrol', 'Diesel', 'Fuel_Oil',
113     'Aviation_Fuel_and_Kerosine', 'Oil_Other'
114 ]
115 Natural_Gas = ['Natural_Gas']
116 Renewables = [
117     'Hydro', 'Geothermal', 'Solar', 'Wind', 'Liquid_Biofuels', 'Biogas',
118     'Wood'
119 ]
119 Electricity = ['Electricity']
120 Waste_Heat = ['Waste_Heat']
121
122 FUEL_ALL = [Coal, Oil, Natural_Gas, Renewables, Electricity, Waste_Heat
123     ]
124 FUEL = []
125 for fuel_type in FUEL_ALL:
126     for i in range(0, len(fuel_type), 1):
127         FUEL.append(fuel_type[i])
128
129 # Sets Specified Fuels
130 SPECIFIED_FUEL_ALL = [
131     Coal, Oil, Natural_Gas, Renewables, Electricity, Waste_Heat
132 ]
133 SPECIFIED_FUEL = []
134 for fuel_type in SPECIFIED_FUEL_ALL:
135     for i in range(0, len(fuel_type), 1):
136         SPECIFIED_FUEL.append(fuel_type[i])
137
138 # Sets Accumulated Fuels
139 ACCUMULATED_FUEL_ALL = [
140     Coal, Oil, Natural_Gas, Renewables, Electricity, Waste_Heat
141 ]
142 ACCUMULATED_FUEL = []
143 for fuel_type in ACCUMULATED_FUEL_ALL:

```

```
138     for i in range(0, len(fuel_type), 1):
139         ACCUMULATED_FUEL.append(fuel_type[i])
140
141 # Sets the total fuels
142 nz_energy_system.set_fuel(FUEL)
143 nz_energy_system.set_specified_fuel(FUEL)
144 nz_energy_system.set_accumulated_fuel(FUEL)
145 #
146 ##########
147 # Continues defining sets
148 #
149 ##########
150 # Defines timeslices
151 TIMESLICE = [
152     'DAY_SUMMER', 'NIGHT_SUMMER', 'DAY_WINTER', 'NIGHT_WINTER',
153     'DAY_INTERMEDIATE', 'NIGHT_INTERMEDIATE'
154 ]
155 nz_energy_system.set_timeslice(TIMESLICE)
156
157 # Defines Modes of Operation
158 nz_energy_system.set_mode_of_operation(1)
159
160 # Defines the storage set
161 STORAGE = ['DAM']
162 nz_energy_system.set_storage(STORAGE)
163
164 # Defines the datatype (numbers represent different datatypes)
165 # 1 = Weekday (Mon - Fri), 2 = Weekend (Sat & Sun)
166 nz_energy_system.set_datatype(2)
167
168 # Defines the seasons
169 # (Three seasons (Summer (1), Winter (2) and Intermediate (3)))
170 nz_energy_system.set_season(3)
171
172 # Defines the dailytimebracket (Number of distinct periods in a day)
173 # 4 = Morning (6hrs), Afternoon (6hrs), Evening (6hrs), Night (6hrs)
174 nz_energy_system.set_daily_time_bracket(4)
175
176 # Defines Global Parameters
177 #
178 ##########
179 # Defines the YearSplit parameter
180 # Creates Dictionary for number of days
181 days = {
182     'January': 31,
183     'February': 28,
184     'March': 31,
185     'April': 30,
186     'May': 31,
187     'June': 30,
188     'July': 31,
189     'August': 31,
```

```

188     'September': 30,
189     'October': 31,
190     'November': 30,
191     'December': 31
192 }
193
194 # Combines summer, winter and intermediate nights
195 days_summer = days['January'] + days['February'] + days['December']
196 days_winter = days['June'] + days['July'] + days['August']
197 days_intermediate = days['April'] + days['May'] + days['March'] + days[
198     'September'] + days['October'] + days['November']
199 days_total = days_summer + days_winter + days_intermediate
200
201 # Creates fractions and stores values in a dictionary
202 day_summer = (0.5 * days_summer / days_total)
203 night_summer = (0.5 * days_summer / days_total)
204 day_winter = (0.5 * days_winter / days_total)
205 night_winter = (0.5 * days_winter / days_total)
206 day_intermediate = (0.5 * days_intermediate / days_total)
207 night_intermediate = (0.5 * days_intermediate / days_total)
208
209 # Dictionaries
210 splits = {
211     'DAY_SUMMER': day_summer,
212     'NIGHT_SUMMER': night_summer,
213     'DAY_WINTER': day_winter,
214     'NIGHT_WINTER': night_winter,
215     'DAY_INTERMEDIATE': day_intermediate,
216     'NIGHT_INTERMEDIATE': night_intermediate
217 }
218 # Creates the YearSplit parameter 2D Matrix
219 nz_energy_system.set_year_split(TIMESLICE, nz_energy_system.year,
220     splits)
221
222 # Imports S&P NZX:50 and S&P ASX:200 Indices Arrays to calculate market
223 # returns
224 root = '/Users/connor/Google Drive/Documents/University/Courses/2020/
225     ENGSCI 700A&B/GOCPI/data/Inputs/GOCPI_OseMOSYS'
226 file_root = Path(root)
227 file_spreadsheet = 'Returns.xls'
228 location = GF.Navigation(file_root, file_spreadsheet)
229 market_returns = location.Find_File()
230 nz_df = pd.read_excel(market_returns, sheet_name='NZ')
231 aus_df = pd.read_excel(market_returns, sheet_name='AUS')
232 nz_index = nz_df[['Monthly_Returns']].to_numpy()
233 aus_index = aus_df[['Monthly_Returns']].to_numpy()
234
235 # Defines the Dictionaries required for Region. All regions should have
236 # the same names
237 # Creates a dictionary of market indices
238 market_index = {'NEWZEALAND': nz_index, 'AUSTRALIA': aus_index}
239 # Treasury Equity Balances as at 2019
240 # (Australia has negative equity, New Zealand has $139746000000)
241 # However, Governments do not have market equity so should be zero for
242 # both
243 equity = {'NEWZEALAND': 0, 'AUSTRALIA': 0}
244 # Treasury Debt Balance as at 2019
245 debt = {'NEWZEALAND': 110477000000, 'AUSTRALIA': 619219000000}

```

```

241 # Treasury Finance Cost(Interest Expenses on Debt as at 2019
242 cost_of_debt_pre_tax = {'NEWZEALAND': 4059000000, 'AUSTRALIA':
243     1708800000}
244 # Preference Equity (None for governments)
245 preference_equity = {'NEWZEALAND': 0, 'AUSTRALIA': 0}
246 market_value_preference_shares = {'NEWZEALAND': 1, 'AUSTRALIA': 1}
247 # (Set to zero if none otherwise you get an error)
248 preference_dividends = {'NEWZEALAND': 0, 'AUSTRALIA': 0}
249 # Calculated from 10 Year Treasury Bonds (10 Year Average)
250 risk_free_rate = {'NEWZEALAND': 0.0360, 'AUSTRALIA': 0.0335}
251 # Company Tax Rates
252 effective_tax_rate = {'NEWZEALAND': 0.28, 'AUSTRALIA': 0.30}
253 # Beta for region modelled
254 market_risk_coefficient = {'NEWZEALAND': 0, 'AUSTRALIA': 0}
255
256 # Sets the discount rates
257 nz_energy_system.set_discount_rate(equity, debt, market_index,
258                                     cost_of_debt_pre_tax, risk_free_rate
259                                     ,
260                                     effective_tax_rate,
261                                     preference_equity,
262                                     market_value_preference_shares,
263                                     preference_dividends,
264                                     market_risk_coefficient)
265
266 # Creates Dictionary of day splits (assumes constant accross years)
267 # Preserve the order of the split.
268 hour_split = {"1": 6, "2": 6, "3": 6, "4": 6}
269 num_days = 365
270 num_hours = 24
271 nz_energy_system.set_day_split(nz_energy_system.dailytimebracket,
272                                 nz_energy_system.year, hour_split,
273                                 num_days,
274                                 num_hours)
275
276 # Sets a dictionary to match the timeslice with season
277 link_ls = {
278     "DAY_SUMMER": "1",
279     "NIGHT_SUMMER": "1",
280     "DAY_WINTER": "2",
281     "NIGHT_WINTER": "2",
282     "DAY_INTERMEDIATE": "3",
283     "NIGHT_INTERMEDIATE": "3"
284 }
285 nz_energy_system.set_conversion_ls(nz_energy_system.timeslice,
286                                     nz_energy_system.season, link_ls)
287 # Sets a dictionary to match the timeslice with daytype
288 # Daytypes: 1 = Weekday (Mon - Fri), 2 = Weekend (Sat & Sun)
289 # Order must be preserved
290 link_ld = {
291     "DAY_SUMMER": np.ones((1, 2)),
292     "NIGHT_SUMMER": np.ones((1, 2)),
293     "DAY_WINTER": np.ones((1, 2)),
294     "NIGHT_WINTER": np.ones((1, 2)),
295     "DAY_INTERMEDIATE": np.ones((1, 2)),
296     "NIGHT_INTERMEDIATE": np.ones((1, 2))
297 }
298 nz_energy_system.set_conversion_ld(nz_energy_system.timeslice,
299                                     link_ld)

```

```

295                                         nz_energy_system.daytype, link_ld)
296 # Sets a dictionary to match the timeslice with daytype
297 # 1). Morning (6hrs), 2).Afternoon (6hrs), 3).Evening (6hrs), 4).Night
298 # (6hrs)
299 # Order must be preserved in the arrays
300 link_lh = {
301     "DAY_SUMMER": np.array([1, 1, 0, 0]),
302     "NIGHT_SUMMER": np.array([0, 0, 1, 1]),
303     "DAY_WINTER": np.array([1, 1, 0, 0]),
304     "NIGHT_WINTER": np.array([0, 0, 1, 1]),
305     "DAY_INTERMEDIATE": np.array([1, 1, 0, 0]),
306     "NIGHT_INTERMEDIATE": np.array([0, 0, 1, 1])
307 }
308 override_conversionlh = None
309 # Sets the Conversionlh parameter
310
311 nz_energy_system.set_conversion_lh(nz_energy_system.timeslice,
312                                     nz_energy_system.dailytimebracket,
313                                     link_lh,
314                                     override_conversionlh)
315 # Creates season dictionary for daytypes (Assumed to be the same each
316 # year)
317 link_dtdt = {
318     "1": np.array([5, 2]),
319     "2": np.array([5, 2]),
320     "3": np.array([5, 2])
321 }
322 override_dtdt = None
323 # Sets the DaysInDayType parameter
324 nz_energy_system.set_days_in_day_type(nz_energy_system.season,
325                                         nz_energy_system.daytype,
326                                         nz_energy_system.year, link_dtdt,
327                                         override_dtdt)
328
329 # Creates trade relationships using an 2D numpy array
330 # Must [NEWZEALAND, AUSTRALIA],[NEWZEALAND, AUSTRALIA]
331 # Hypothetically, you can model any trade relationship for any fuel in
332 # any year
333 # FUELS = As above
334 # YEAR = 2020 - 2030 (11)
335 trade = np.zeros((len(nz_energy_system.region), len(nz_energy_system.
336                   region),
337                         len(nz_energy_system.fuel), len(nz_energy_system.year
338                         )))
339 trade_all_fuels = np.array([[0, 1], [1, 0]])
340 for i in range(0, len(nz_energy_system.fuel), 1):
341     for j in range(0, len(nz_energy_system.year), 1):
342         trade[:, :, i, j] = trade_all_fuels
343 nz_energy_system.set_trade_route(trade)
344
345 # Creates depreciation methods dictionary
346 depreciation_methods = {"NEWZEALAND": 2, "AUSTRALIA": 2}
347 override_depreciation = None
348 nz_energy_system.set_depreciation_method(nz_energy_system.region,
349                                         depreciation_methods,
350                                         override_depreciation)
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900

```

```

346 #
347 # ##### Initialisation and Definition of demand parameters (Including
348 # forecasting)
349 #
350 # Sets dictionaries to calculate CAGR for Fuels Forecasts
351 nz_cagr_fuels = {}
352 aus_cagr_fuels = {}
353 cagr_dictionaries_regions = [nz_cagr_fuels, aus_cagr_fuels]
354 # Initialises cagr parameters
355 nz_start_year_fuels = {}
356 nz_end_year_fuels = {}
357 nz_start_value_fuels = {}
358 nz_end_value_fuels = {}
359 aus_start_year_fuels = {}
360 aus_end_year_fuels = {}
361 aus_start_value_fuels = {}
362 aus_end_value_fuels = {}
363 nz_cagr_dictionaries_parameters = [
364     nz_start_year_fuels, nz_end_year_fuels, nz_start_value_fuels,
365     nz_end_value_fuels
366 ]
367 aus_cagr_dictionaries_parameters = [
368     aus_start_year_fuels, aus_end_year_fuels, aus_start_value_fuels,
369     aus_end_value_fuels
370 ]
371 # Populates regional dictionaries with new entry, all fuel types with
372 # default cagr values
373 for region_fuels in cagr_dictionaries_regions:
374     for i in range(0, len(nz_energy_system.fuel), 1):
375         region_fuels[nz_energy_system.fuel[i]] = 0.05
376 # Populates regional dictionaries with new entry, all fuel types with
377 # default values
378 for parameters in nz_cagr_dictionaries_parameters:
379     for i in range(0, len(nz_energy_system.fuel), 1):
380         region_fuels[nz_energy_system.fuel[i]] = 1
381 for parameters in nz_cagr_dictionaries_parameters:
382     for i in range(0, len(nz_energy_system.fuel), 1):
383         region_fuels[nz_energy_system.fuel[i]] = 1
384 # Loads demand data to the parameter dictionaries (Energy units are in
385 # PJ's)
386 # New Zealand
387 nz_start_years = np.zeros(len(nz_energy_system.fuel))
388 nz_start_years[:] = 2010
389 nz_end_years = np.zeros(len(nz_energy_system.fuel))
390 nz_end_years[:] = 2018
391 nz_start_values = np.array([
392     7.23, 13.24, 4.19, 0, 7.11, 110.43, 106.09, 7.11, 14.62, 0, 60.29,
393     0, 9.21,
394     0.35, 0, 0, 0.33, 55.89, 146.49, 0
395 ])

```

```

395 nz_end_values = np.zeros(len(nz_energy_system.fuel))
396 nz_end_values = np.array([
397     3.07, 16.26, 5.14, 0, 8.71, 113.22, 138.79, 5.82, 16.23, 0, 73.97,
398     0, 8.03,
399     0.36, 0, 0, 0.33, 56.61, 142.87, 0
400 ])
401 # Australia
402 aus_start_years = np.zeros(len(nz_energy_system.fuel))
403 aus_start_years[:] = 2017
404 aus_end_years = np.zeros(len(nz_energy_system.fuel))
405 aus_end_years[:] = 2018
406 aus_start_values = np.array([
407     104.9, 9.0, 0.5, 2.3, 72.4, 847.9724, 1038.76619, 42.39862,
408     190.79379, 0.0,
409     0.0, 0.0, 0, 15.7, 0.0, 8.4, 94.7, 79.2, 821.8, 0
410 ])
411 aus_end_values = np.zeros(len(nz_energy_system.fuel))
412 aus_end_values = np.array([
413     104.445, 8.737, 0.38, 2.019, 67.499, 904.7584, 1108.32904,
414     45.23792,
415     135.71376, 0.35788, 942.965, 0, 0, 16.56, 0, 8.642, 83.592, 76.81,
416     835.439,
417     0
418 ])
419 # Assign values to the dictionary
420 for i in range(0, len(nz_energy_system.fuel), 1):
421     aus_start_year_fuels[nz_energy_system.fuel[i]] = aus_start_years[i]
422     aus_end_year_fuels[nz_energy_system.fuel[i]] = aus_end_years[i]
423     aus_start_value_fuels[nz_energy_system.fuel[i]] = aus_start_values[i]
424     aus_end_value_fuels[nz_energy_system.fuel[i]] = aus_end_values[i]
425     nz_start_year_fuels[nz_energy_system.fuel[i]] = nz_start_years[i]
426     nz_end_year_fuels[nz_energy_system.fuel[i]] = nz_end_years[i]
427     nz_start_value_fuels[nz_energy_system.fuel[i]] = nz_start_values[i]
428     nz_end_value_fuels[nz_energy_system.fuel[i]] = nz_end_values[i]
429
430 print("nz_start_year_fuels", nz_start_year_fuels)
431 print("nz_end_year_fuels", nz_end_year_fuels)
432 print("nz_start_value_fuels", nz_start_value_fuels)
433 print("nz_end_value_fuels", nz_end_value_fuels)
434
435 print("aus_start_year_fuels", aus_start_year_fuels)
436 print("aus_end_year_fuels", aus_end_year_fuels)
437 print("aus_start_value_fuels", aus_start_value_fuels)
438 print("aus_end_value_fuels", aus_end_value_fuels)
439
440 # Calculates the cagr dictionary
441 forecasting_functions = GF.Forecasting()
442 for fuel in nz_cagr_fuels:
443     nz_cagr_fuels[
444         fuel] = forecasting_functions.
445         calculate_constant_average_growth_rate(
446             nz_start_year_fuels[fuel], nz_end_year_fuels[fuel],
447             nz_start_value_fuels[fuel], nz_end_value_fuels[fuel])
448
449 for fuel in aus_cagr_fuels:
450     aus_cagr_fuels[
451         fuel] = forecasting_functions.
452         calculate_constant_average_growth_rate(
453             aus_start_year_fuels[fuel], aus_end_year_fuels[fuel],
454             aus_start_value_fuels[fuel], aus_end_value_fuels[fuel])
455

```

```

446             aus_start_value_fuels[fuel], aus_end_value_fuels[fuel])
447
448 # Calculates NZ CAGR forecasts
449 nz_fuel_forecast = forecasting_functions.calculate_cagr_forecasts(
450     nz_cagr_fuels, nz_end_value_fuels, nz_energy_system.fuel,
451     nz_energy_system.year)
452
453 # Calculates AUS CAGR forecasts
454 aus_fuel_forecast = forecasting_functions.calculate_cagr_forecasts(
455     aus_cagr_fuels, aus_end_value_fuels, nz_energy_system.fuel,
456     nz_energy_system.year)
457
458 fuel_forecasts = [nz_fuel_forecast, aus_fuel_forecast]
459
460 # Creates the forecast 3D array
461 forecast = np.zeros((len(nz_energy_system.region), len(nz_energy_system.
462     .fuel),
463                         len(nz_energy_system.year)))
464
465 # Sets the forecast 3D array with CAGR forecast values
466 for i in range(0, len(fuel_forecasts), 1):
467     forecast[i, :, :] = fuel_forecasts[i]
468
469 # Sets the Specified Demand Profiles
470 # nz_energy_system.set_specified_annual_demand(forecast[:, 0:-1, :])
471 nz_energy_system.set_specified_annual_demand(forecast[:, :, :])
472 # Sets the Accumulated Demand Profiles (Hack to make sure 3D Array)
473 acc_forecast = np.zeros(
474     (len(nz_energy_system.region), len(nz_energy_system.
475         accumulated_fuel),
476         len(nz_energy_system.year)))
477 acc_forecast[:, 0, :] = forecast[:, -1, :]
478
479 # Make adjustments to the accumulated fuel forecasts
480 nz_energy_system.set_accumulated_annual_demand(forecast[:, :, :])
481 # Sets linear profile for timeslices (In this example, it is assumed
482 #     the fuel is consumed uniformly in time splits)
483 linear_profile = splits
484 override = None
485
486 # Sets the Specified Demand Profiles
487 nz_energy_system.set_specified_demand_profile(
488     nz_energy_system.SpecifiedAnnualDemand, nz_energy_system.region,
489     nz_energy_system.specified_fuel, nz_energy_system.year,
490     nz_energy_system.timeslice, linear_profile, override)
491
492 # Sets the Capacity to Activity Factors (Assume conversion of GW to PJ)
493 nz_capacity_to_activity = {}
494 aus_capacity_to_activity = {}
495 for tech in nz_energy_system.capacity_technology:
496     nz_capacity_to_activity[tech] = 31.536
497     aus_capacity_to_activity[tech] = 31.536
498
499 capacity_dictionaries = [nz_capacity_to_activity,
500                           aus_capacity_to_activity]
501 # Sets the CapacityToActivity Function
502 override = None
503 nz_energy_system.set_capacity_to_activity_unit(
504     nz_energy_system.region, nz_energy_system.capacity_technology,

```

```

500     capacity_dictionaries, override)
501 print(nz_energy_system.capacity_technology)
502 print(nz_energy_system.CapacityToActivityUnit)
503
504 # Sets capacity factor matrix to operate in every timeslice (Assumes
#      operate 0.8 of the time).
505 capacity_factors = np.zeros(
506     (len(nz_energy_system.region), len(nz_energy_system.
507         capacity_technology),
508         len(nz_energy_system.timeslice), len(nz_energy_system.year)))
509 capacity_factors[:, :, :, :] = 0.8
510
511 nz_energy_system.set_capacity_factor(capacity_factors)
512
513 # Set availability factors
514 availability_factors = np.zeros((len(nz_energy_system.region),
515                                     len(nz_energy_system.
516                                         availability_technology),
517                                         len(nz_energy_system.year)))
518
519 availability_factors[:, :, :] = 1
520 nz_energy_system.set_availability_factor(availability_factors)
521
522 # Sets up operational life
523 #
524 # print(nz_energy_system.YearSplit)
525 # print(nz_energy_system.DiscountRate)
526 # print(nz_energy_system.DaySplit)
527 # print(nz_energy_system.ConversionId)
528 # print(nz_energy_system.Conversionls)
529 # print(nz_energy_system.Conversionlh)
530 # print(nz_energy_system.TradeRoute)
531 # print(nz_energy_system.DaysInDayType)
532 # print(nz_energy_system.DepreciationMethod)
533
534 # Initialises yet to be written parameters to check progress / load
#      Parameters (Delete later)
535 ly = len(nz_energy_system.year)
536 lr = len(nz_energy_system.region)
537 le = len(nz_energy_system.emission)
538 lt = len(nz_energy_system.technology)
539 lf = len(nz_energy_system.fuel)
540 ll = len(nz_energy_system.timeslice)
541 lm = len(nz_energy_system.mode_of_operation)
542 ls = len(nz_energy_system.storage)
543 lld = len(nz_energy_system.daytype)
544 lls = len(nz_energy_system.season)
545 llh = len(nz_energy_system.dailytimebracket)
546
547 #nz_energy_system.YearSplit = np.ones((ll, ly))
548 #nz_energy_system.DiscountRate = np.ones((lr))
549 #nz_energy_system.DaySplit = np.ones((llh, ly))
550 #nz_energy_system.Conversionls = np.ones((ll, lls))
551 #nz_energy_system.ConversionId = np.ones((ll, lld))
552 #nz_energy_system.Conversionlh = np.ones((ll, llh))
553 #nz_energy_system.DaysInDayType = np.ones((lls, lld, ly))

```

```

554 #nz_energy_system.TradeRoute = np.ones((lr, lr, lf, ly))
555 #nz_energy_system.DepreciationMethod = np.ones((lr))
556 #nz_energy_system.SpecifiedAnnualDemand = np.ones((lr, lf, ly))
557 #nz_energy_system.SpecifiedDemandProfile = np.ones((lr, lf, ll, ly))
558 #nz_energy_system.AccumulatedAnnualDemand = np.ones((lr, lf, ly))
559 #nz_energy_system.CapacityToActivityUnit = np.ones((lr, lt))
560 #nz_energy_system.CapacityFactor = np.ones((lr, lt, ll, ly))
561 #nz_energy_system.AvailabilityFactor = np.ones((lr, lt, ly))
562 nz_energy_system.OperationalLife = np.ones((lr, lt))
563 nz_energy_system.ResidualCapacity = np.ones((lr, lt, ly))
564 nz_energy_system.InputActivityRatio = np.ones((lr, lt, lf, lm, ly))
565 nz_energy_system.OutputActivityRatio = np.ones((lr, lt, lf, lm, ly))
566 nz_energy_system.CapitalCost = np.ones((lr, lt, ly))
567 nz_energy_system.VariableCost = np.ones((lr, lt, lm, ly))
568 nz_energy_system.FixedCost = np.ones((lr, lt, ly))
569 nz_energy_system.TechnologyToStorage = np.ones((lr, lt, ls, lm))
570 nz_energy_system.TechnologyFromStorage = np.ones((lr, lt, ls, lm))
571 nz_energy_system.StorageLevelStart = np.ones((lr, ls))
572 nz_energy_system.StorageMaxChargeRate = np.ones((lr, ls))
573 nz_energy_system.StorageMaxDischargeRate = np.ones((lr, ls))
574 nz_energy_system.MinStorageCharge = np.ones((lr, ls, ly))
575 nz_energy_system.OperationalLifeStorage = np.ones((lr, ls))
576 nz_energy_system.CapitalCostStorage = np.ones((lr, ls, ly))
577 nz_energy_system.ResidualStorageCapacity = np.ones((lr, ls, ly))
578 nz_energy_system.CapacityOfOneTechnologyUnit = np.ones((lr, lt, ly))
579 nz_energy_system.TotalAnnualMaxCapacity = np.ones((lr, lt, ly))
580 nz_energy_system.TotalAnnualMinCapacity = np.ones((lr, lt, ly))
581 nz_energy_system.TotalAnnualMaxCapacityInvestment = np.ones((lr, lt, ly
    ))
582 nz_energy_system.TotalAnnualMinCapacityInvestment = np.ones((lr, lt, ly
    ))
583 nz_energy_system.TotalTechnologyAnnualActivityLowerLimit = np.ones(
    (lr, lt, ly))
584 nz_energy_system.TotalTechnologyAnnualActivityUpperLimit = np.ones(
    (lr, lt, ly))
585 nz_energy_system.TotalTechnologyModelPeriodActivityUpperLimit = np.ones(
    (
    lr, lt))
586 nz_energy_system.TotalTechnologyModelPeriodActivityLowerLimit = np.ones(
    (
    lr, lt))
587 nz_energy_system.ReserveMarginTagTechnology = np.ones((lr, lt, ly))
588 nz_energy_system.ReserveMarginTagFuel = np.ones((lr, lf, ly))
589 nz_energy_system.ReserveMargin = np.ones((lr, ly))
590 nz_energy_system.RETagTechnology = np.ones((lr, lt, ly))
591 nz_energy_system.RETagFuel = np.ones((lr, lf, ly))
592 nz_energy_system.REMinProductionTarget = np.ones((lr, ly))
593 nz_energy_system.EmissionActivityRatio = np.ones((lr, lt, le, lm, ly))
594 nz_energy_system.EmissionsPenalty = np.ones((lr, le, ly))
595 nz_energy_system.AnnualExogenousEmission = np.ones((lr, le, ly))
596 nz_energy_system.AnnualEmissionLimit = np.ones((lr, le, ly))
597 nz_energy_system.ModelPeriodExogenousEmission = np.ones((lr, le))
598 nz_energy_system.ModelPeriodEmissionLimit = np.ones((lr, le))
599
600
601
602
603
604 # Sets the case (Toggle depending on the data set you choose to use)
605 case = nz_energy_system
606
607 # Initialises the energy system

```

```

608 system = GF.Energy_Systems(
609     nz_energy_system.year, nz_energy_system.region, nz_energy_system.
610     emission,
611     nz_energy_system.technology, nz_energy_system.capacity_technology,
612     nz_energy_system.availability_technology, nz_energy_system.fuel,
613     nz_energy_system.specified_fuel, nz_energy_system.accumulated_fuel,
614     nz_energy_system.timeslice, nz_energy_system.mode_of_operation,
615     nz_energy_system.storage, nz_energy_system.daytype,
616     nz_energy_system.season, nz_energy_system.dailytimebracket)
617
617 # Loads the datacase to the system
618 system.load_datacase(case, system)
619
620 # Sets up location information
621 data_txt = 'GOCPI_NZ_Example_Data.txt'
622 model_source_file = 'GOCPI_OseMOSYS_Structure.xlsx'
623 root = '/Users/connor/Google Drive/Documents/University/Courses/2020/
624     ENGSCI 700A&B/GOCPI/data/Inputs/GOCPI_OseMOSYS'
624 data_roots = Path(root)
625 data_location_1 = os.path.join(data_roots, data_txt)
626
627 # Sets the default parameters
628 default_parameters = {
629     'YearSplit': 1,
630     'DiscountRate': 0.05,
631     'DaySplit': 1,
632     'Conversionls': 1,
633     'Conversionld': 1,
634     'Conversionlh': 1,
635     'DaysInDayType': 1,
636     'TradeRoute': 1,
637     'DepreciationMethod': 2,
638     'SpecifiedAnnualDemand': 1,
639     'SpecifiedDemandProfile': 1,
640     'AccumulatedAnnualDemand': 1,
641     'CapacityToActivityUnit': 1,
642     'CapacityFactor': 1,
643     'AvailabilityFactor': 1,
644     'OperationalLife': 1,
645     'ResidualCapacity': 1,
646     'InputActivityRatio': 1,
647     'OutputActivityRatio': 1,
648     'CapitalCost': 1,
649     'VariableCost': 1,
650     'FixedCost': 1,
651     'TechnologyToStorage': 1,
652     'TechnologyFromStorage': 1,
653     'StorageLevelStart': 1,
654     'StorageMaxChargeRate': 1,
655     'StorageMaxDischargeRate': 1,
656     'MinStorageCharge': 1,
657     'OperationalLifeStorage': 1,
658     'CapitalCostStorage': 1,
659     'ResidualStorageCapacity': 1,
660     'CapacityOfOneTechnologyUnit': 1,
661     'TotalAnnualMaxCapacity': 99999,
662     'TotalAnnualMinCapacity': 1,
663     'TotalAnnualMaxCapacityInvestment': 999999,

```

```

664     'TotalAnnualMinCapacityInvestment': 0,
665     'TotalTechnologyAnnualActivityLowerLimit': 0,
666     'TotalTechnologyAnnualActivityUpperLimit': 999999,
667     'TotalTechnologyModelPeriodActivityUpperLimit': 999999,
668     'TotalTechnologyModelPeriodActivityLowerLimit': 0,
669     'ReserveMarginTagTechnology': 1,
670     'ReserveMarginTagFuel': 1,
671     'ReserveMargin': 1,
672     'RETagTechnology': 1,
673     'RETagFuel': 1,
674     'REMinProductionTarget': 1,
675     'EmissionActivityRatio': 1,
676     'EmissionsPenalty': 1,
677     'AnnualExogenousEmission': 1,
678     'AnnualEmissionLimit': 1,
679     'ModelPeriodExogenousEmission': 1,
680     'ModelPeriodEmissionLimit': 1
681 }
682
683 # Sets the default toggles (To only use defaults)
684 toggle_defaults = {
685     'YearSplit': False,
686     'DiscountRate': False,
687     'DaySplit': False,
688     'Conversionls': False,
689     'Conversionld': False,
690     'Conversionlh': False,
691     'DaysInDayType': False,
692     'TradeRoute': False,
693     'DepreciationMethod': False,
694     'SpecifiedAnnualDemand': False,
695     'SpecifiedDemandProfile': False,
696     'AccumulatedAnnualDemand': False,
697     'CapacityToActivityUnit': False,
698     'CapacityFactor': False,
699     'AvailabilityFactor': False,
700     'OperationalLife': False,
701     'ResidualCapacity': False,
702     'InputActivityRatio': False,
703     'OutputActivityRatio': False,
704     'CapitalCost': False,
705     'VariableCost': False,
706     'FixedCost': False,
707     'TechnologyToStorage': False,
708     'TechnologyFromStorage': False,
709     'StorageLevelStart': False,
710     'StorageMaxChargeRate': False,
711     'StorageMaxDischargeRate': False,
712     'MinStorageCharge': False,
713     'OperationalLifeStorage': False,
714     'CapitalCostStorage': False,
715     'ResidualStorageCapacity': False,
716     'CapacityOfOneTechnologyUnit': False,
717     'TotalAnnualMaxCapacity': False,
718     'TotalAnnualMinCapacity': False,
719     'TotalAnnualMaxCapacityInvestment': False,
720     'TotalAnnualMinCapacityInvestment': False,
721     'TotalTechnologyAnnualActivityLowerLimit': False,

```

```

722     'TotalTechnologyAnnualActivityUpperLimit': False,
723     'TotalTechnologyModelPeriodActivityUpperLimit': False,
724     'TotalTechnologyModelPeriodActivityLowerLimit': False,
725     'ReserveMarginTagTechnology': False,
726     'ReserveMarginTagFuel': False,
727     'ReserveMargin': False,
728     'RETagTechnology': False,
729     'RETagFuel': False,
730     'REMinProductionTarget': False,
731     'EmissionActivityRatio': False,
732     'EmissionsPenalty': False,
733     'AnnualExogenousEmission': False,
734     'AnnualEmissionLimit': False,
735     'ModelPeriodExogenousEmission': False,
736     'ModelPeriodEmissionLimit': False
737 }
738 # Sets the default toggles (To only use defaults)
739 # toggle_defaults = {
740 #     'YearSplit': False,
741 #     'DiscountRate': False,
742 #     'DaySplit': False,
743 #     'Conversionsl': False,
744 #     'Conversionld': True,
745 #     'Conversionlh': True,
746 #     'DaysInDayType': True,
747 #     'TradeRoute': True,
748 #     'DepreciationMethod': True,
749 #     'SpecifiedAnnualDemand': True,
750 #     'SpecifiedDemandProfile': True,
751 #     'AccumulatedAnnualDemand': True,
752 #     'CapacityToActivityUnit': True,
753 #     'CapacityFactor': True,
754 #     'AvailabilityFactor': True,
755 #     'OperationalLife': True,
756 #     'ResidualCapacity': True,
757 #     'InputActivityRatio': True,
758 #     'OutputActivityRatio': True,
759 #     'CapitalCost': True,
760 #     'VariableCost': True,
761 #     'FixedCost': True,
762 #     'TechnologyToStorage': True,
763 #     'TechnologyFromStorage': True,
764 #     'StorageLevelStart': True,
765 #     'StorageMaxChargeRate': True,
766 #     'StorageMaxDischargeRate': True,
767 #     'MinStorageCharge': True,
768 #     'OperationalLifeStorage': True,
769 #     'CapitalCostStorage': True,
770 #     'ResidualStorageCapacity': True,
771 #     'CapacityOfOneTechnologyUnit': True,
772 #     'TotalAnnualMaxCapacity': True,
773 #     'TotalAnnualMinCapacity': True,
774 #     'TotalAnnualMaxCapacityInvestment': True,
775 #     'TotalAnnualMinCapacityInvestment': True,
776 #     'TotalTechnologyAnnualActivityLowerLimit': True,
777 #     'TotalTechnologyAnnualActivityUpperLimit': True,
778 #     'TotalTechnologyModelPeriodActivityUpperLimit': True,
779 #     'TotalTechnologyModelPeriodActivityLowerLimit': True,

```

```
780 # 'ReserveMarginTagTechnology': True,
781 # 'ReserveMarginTagFuel': True,
782 # 'ReserveMargin': True,
783 # 'RETagTechnology': True,
784 # 'RETagFuel': True,
785 # 'REMinProductionTarget': True,
786 # 'EmissionActivityRatio': False,
787 # 'EmissionsPenalty': False,
788 # 'AnnualExogenousEmission': False,
789 # 'AnnualEmissionLimit': False,
790 # 'ModelPeriodExogenousEmission': False,
791 # 'ModelPeriodEmissionLimit': False
792 # }
793
794 # Create the Data File
795 system.create_data_file(data_location_1, default_parameters,
    toggle_defaults)
796
797 # Create the Model File
798 system.create_model_file(root, model_source_file)
```

The GOCPI NZ Energy Systems Example is the processing script for designing NZ and AUS Energy Systems

5.2 GOCPI Module

5.2.1 Navigation

The module to provide navigation functionalities to access files in directories.

```
1 import os
2
3
4 class Navigation:
5     """ Navigation is a class for navigating, manipulating and editing
6     data in the GOCPI model.
7
8     Attributes:
9         Find_File(string) representing a string to the file path
10
11    TODO: Fill out all functions below
12
13    """
14
15    def __init__(self, target_root, target_file):
16        """ Initialises the navigation functions
17
18        Args:
19            target_root (str): Base directory to search from
20            target_file (str): Name of file to search for
21
22        """
23
24        self.target_root = target_root
25        self.target_file = target_file
26
27    def Find_File(self):
28        """ Find_File searches for a target file, from a base directory
29        , to construct
30            a target directory.
31
```

```

27
28     Returns:
29         str: File path for file
30     """
31
32     for root, dirs, files in os.walk(self.target_root):
33         for name in files:
34             if name == self.target_file:
35                 f = os.path.abspath(os.path.join(root, name))
36     return f
37
38     def create_linear_programme_file(self, directory, data_file,
39                                     model_file,
40                                     output_file):
41         """ Creates the model file through executing model system
42         commands
43             (Work in Progress)
44
45         Args:
46             directory (str): Name of directory to put data into
47             data_file (str): Name of energy system data file
48             model_file (str): Name of energy system model file
49             output_file (str): Name of output linear programme
50         """
51         # Change the working directory
52         os.chdir(directory)
53         # Load the custom anaconda environment
54         # This assumes the conda environment has already been
55         # initialised.
56         os.system('conda activate osemosys')
57         # Execute the file structure to create the linear programming
58         # file
59         # (glpsol -m GOCPI_OSeMOSYS_Model.txt -d GOCPI_NZ_Example_Data.
60         # txt --wlp GOCPI_NZ_Example.lpx)
61         command = 'glpsol -m ' + data_file + ' -d ' + model_file + '--'
62         wlp ' + output_file
63         os.system(command)
64
65

```

5.2.2 Energysystems

The module to load in existing energy systems to create model and data files.

```

1 import os
2 import numpy as np
3 import pandas as pd
4
5
6 class Energy_Systems:
7     """ A class of methods to initialise energy sytems and create the
8     data/model files needed for optimisation.
9     """
10
11     def __init__(self, year, region, emission, technology,
12                  capacity_technology,
13                  availability_technology, fuel, specified_fuel,
14                  accumulated_fuel, timeslice, mode_of_operation,
15                  storage,
16                  datatype, season, dailytimebracket):
17
18

```

```

13     """ Function to create complete energy system set to prepare
14     datafile, as per the established model.
15
16     Args:
17         year (list): List of years
18         region (list): List of regions
19         emission (list): List of emissions
20         technology (list): List of technologies
21         capacity_technology (list): List of technologies
22         availability_technology (list): List of technologies
23         fuel (list): List of fuels
24         specified_fuel (list): List of fuels
25         accumulated_fuel (list): List of fuels
26         timeslice (list): List of timeslices
27         mode_of_operation (list): List of modes of operation
28         storage (list): List of storage
29         daytype (list): List of daytypes
30         season (list): List of seasons
31         dailytimebracket (list): List of dailytimebrackets
32
33     """
34
35     self.year = year
36     self.region = region
37     self.emission = emission
38     self.technology = technology
39     self.capacity_technology = capacity_technology
40     self.availability_technology = availability_technology
41     self.fuel = fuel
42     self.specified_fuel = specified_fuel
43     self.accumulated_fuel = accumulated_fuel
44     self.timeslice = timeslice
45     self.mode_of_operation = mode_of_operation
46     self.storage = storage
47     self.daytype = daytype
48     self.season = season
49     self.dailytimebracket = dailytimebracket
50
51     ly = len(self.year)
52     lr = len(self.region)
53     le = len(self.emission)
54     lt = len(self.technology)
55     lct = len(self.capacity_technology)
56     lat = len(self.availability_technology)
57     lf = len(self.fuel)
58     lsf = len(self.specified_fuel)
59     laf = len(self.accumulated_fuel)
60     ll = len(self.timeslice)
61     lm = len(self.mode_of_operation)
62     ls = len(self.storage)
63     lld = len(self.daytype)
64     lls = len(self.season)
65     llh = len(self.dailytimebracket)
66
67     self.ly = ly
68     self.lr = lr
69     self.le = le

```

```

70     self.lf = lf
71     self.lsf = lsf
72     self.laf = laf
73     self.ll = ll
74     self.lm = lm
75     self.ls = ls
76     self.lld = lld
77     self.lls = lls
78     self.llh = llh
79
80     self.YearSplit = np.ones((ll, ly))
81     self.DiscountRate = np.ones((lr))
82     self.DaySplit = np.ones((llh, ly))
83     self.Conversionls = np.ones((ll, lls))
84     self.Conversionld = np.ones((ll, lld))
85     self.Conversionlh = np.ones((ll, llh))
86     self.DaysInDayType = np.ones((lls, lld, ly))
87     self.TradeRoute = np.ones((lr, lr, lf, ly))
88     self.DepreciationMethod = np.ones((lr))
89     self.SpecifiedAnnualDemand = np.ones((lr, lsf, ly))
90     self.SpecifiedDemandProfile = np.ones((lr, lsf, ll, ly))
91     self.AccumulatedAnnualDemand = np.ones((lr, laf, ly))
92     self.CapacityToActivityUnit = np.ones((lr, lct))
93     self.CapacityFactor = np.ones((lr, lct, ll, ly))
94     self.AvailabilityFactor = np.ones((lr, lat, ly))
95     self.OperationalLife = np.ones((lr, lct))
96     self.ResidualCapacity = np.ones((lr, lt, ly))
97     self.InputActivityRatio = np.ones((lr, lt, lf, lm, ly))
98     self.OutputActivityRatio = np.ones((lr, lt, lf, lm, ly))
99     self.CapitalCost = np.ones((lr, lt, ly))
100    self.VariableCost = np.ones((lr, lt, lm, ly))
101    self.FixedCost = np.ones((lr, lt, ly))
102    self.TechnologyToStorage = np.ones((lr, lt, ls, lm))
103    self.TechnologyFromStorage = np.ones((lr, lt, ls, lm))
104    self.StorageLevelStart = np.ones((lr, ls))
105    self.StorageMaxChargeRate = np.ones((lr, ls))
106    self.StorageMaxDischargeRate = np.ones((lr, ls))
107    self.MinStorageCharge = np.ones((lr, ls, ly))
108    self.OperationalLifeStorage = np.ones((lr, ls))
109    self.CapitalCostStorage = np.ones((lr, ls, ly))
110    self.ResidualStorageCapacity = np.ones((lr, ls, ly))
111    self.CapacityOfOneTechnologyUnit = np.ones((lr, lt, ly))
112    self.TotalAnnualMaxCapacity = np.ones((lr, lt, ly))
113    self.TotalAnnualMinCapacity = np.ones((lr, lt, ly))
114    self.TotalAnnualMaxCapacityInvestment = np.ones((lr, lt, ly))
115    self.TotalAnnualMinCapacityInvestment = np.ones((lr, lt, ly))
116    self.TotalTechnologyAnnualActivityLowerLimit = np.ones((lr, lt,
ly))
117    self.TotalTechnologyAnnualActivityUpperLimit = np.ones((lr, lt,
ly))
118    self.TotalTechnologyModelPeriodActivityUpperLimit = np.ones((lr
, lt))
119    self.TotalTechnologyModelPeriodActivityLowerLimit = np.ones((lr
, lt))
120    self.ReserveMarginTagTechnology = np.ones((lr, lt, ly))
121    self.ReserveMarginTagFuel = np.ones((lr, lf, ly))
122    self.ReserveMargin = np.ones((lr, ly))
123    self.RETagTechnology = np.ones((lr, lt, ly))

```

```

124     self.RETagFuel = np.ones((lr, lf, ly))
125     self.REMinProductionTarget = np.ones((lr, ly))
126     self.EmissionActivityRatio = np.ones((lr, lt, le, lm, ly))
127     self.EmissionsPenalty = np.ones((lr, le, ly))
128     self.AnnualExogenousEmission = np.ones((lr, le, ly))
129     self.AnnualEmissionLimit = np.ones((lr, le, ly))
130     self.ModelPeriodExogenousEmission = np.ones((lr, le))
131     self.ModelPeriodEmissionLimit = np.ones((lr, le))
132
133     def load_datacase(self, case, system):
134         """ Loads the data case to a correct configured and intialized
135         energy system
136
137         Args:
138             case (object): Energy system datacase
139             system (object): Initialised energy system
140
141         Returns:
142             system_case (dict): Returns the updated dictionary
143         """
144
145         # Loads the sets to the energy system
146         system.year = case.year
147         system.region = case.region
148         system.emission = case.emission
149         system.capacity_technology = case.capacity_technology
150         system.availability_technology = case.availability_technology
151         system.technology = case.technology
152         system.fuel = case.fuel
153         system.specified_fuel = case.specified_fuel
154         system.accumulated_fuel = case.accumulated_fuel
155         system.timeslice = case.timeslice
156         system.mode_of_operation = case.mode_of_operation
157         system.storage = case.storage
158         system.daytype = case.daytype
159         system.season = case.season
160         system.dailytimebracket = case.dailytimebracket
161
162         # Loads the parameters to the energy system
163         system.YearSplit = case.YearSplit
164         system.DiscountRate = case.DiscountRate
165         system.DaySplit = case.DaySplit
166         system.Conversionsls = case.Conversionsls
167         system.Conversionld = case.Conversionld
168         system.Conversionlh = case.Conversionlh
169         system.DaysInDayType = case.DaysInDayType
170         system.TradeRoute = case.TradeRoute
171         system.DepreciationMethod = case.DepreciationMethod
172         system.SpecifiedAnnualDemand = case.SpecifiedAnnualDemand
173         system.SpecifiedDemandProfile = case.SpecifiedDemandProfile
174         system.AccumulatedAnnualDemand = case.AccumulatedAnnualDemand
175         system.CapacityToActivityUnit = case.CapacityToActivityUnit
176         system.CapacityFactor = case.CapacityFactor
177         system.AvailabilityFactor = case.AvailabilityFactor
178         system.OperationalLife = case.OperationalLife
179         system.ResidualCapacity = case.ResidualCapacity
180         system.InputActivityRatio = case.InputActivityRatio
181         system.OutputActivityRatio = case.OutputActivityRatio
182         system.CapitalCost = case.CapitalCost
183         system.VariableCost = case.VariableCost

```

```

181     system.FixedCost = case.FixedCost
182     system.TechnologyToStorage = case.TechnologyToStorage
183     system.TechnologyFromStorage = case.TechnologyFromStorage
184     system.StorageLevelStart = case.StorageLevelStart
185     system.StorageMaxChargeRate = case.StorageMaxChargeRate
186     system.StorageMaxDischargeRate = case.StorageMaxDischargeRate
187     system.MinStorageCharge = case.MinStorageCharge
188     system.OperationalLifeStorage = case.OperationalLifeStorage
189     system.CapitalCostStorage = case.CapitalCostStorage
190     system.ResidualStorageCapacity = case.ResidualStorageCapacity
191     system.CapacityOfOneTechnologyUnit = case.
192     CapacityOfOneTechnologyUnit
193         system.TotalAnnualMaxCapacity = case.TotalAnnualMaxCapacity
194         system.TotalAnnualMinCapacity = case.TotalAnnualMinCapacity
195         system.TotalAnnualMaxCapacityInvestment = case.
196         TotalAnnualMaxCapacityInvestment
197             system.TotalAnnualMinCapacityInvestment = case.
198         TotalAnnualMinCapacityInvestment
199             system.TotalTechnologyAnnualActivityLowerLimit = case.
200         TotalTechnologyAnnualActivityLowerLimit
201             system.TotalTechnologyAnnualActivityUpperLimit = case.
202         TotalTechnologyAnnualActivityUpperLimit
203             system.TotalTechnologyModelPeriodActivityUpperLimit = case.
204         TotalTechnologyModelPeriodActivityUpperLimit
205             system.TotalTechnologyModelPeriodActivityLowerLimit = case.
206         TotalTechnologyModelPeriodActivityLowerLimit
207             system.ReserveMarginTagTechnology = case.
208         ReserveMarginTagTechnology
209             system.ReserveMarginTagFuel = case.ReserveMarginTagFuel
210             system.ReserveMargin = case.ReserveMargin
211             system.RETagTechnology = case.RETagTechnology
212             system.RETagFuel = case.RETagFuel
213             system.REMinProductionTarget = case.REMinProductionTarget
214             system.EmissionActivityRatio = case.EmissionActivityRatio
215             system.EmissionsPenalty = case.EmissionsPenalty
216             system.AnnualExogenousEmission = case.AnnualExogenousEmission
217             system.AnnualEmissionLimit = case.AnnualEmissionLimit
218             system.ModelPeriodExogenousEmission = case.
219         ModelPeriodExogenousEmission
220             system.ModelPeriodEmissionLimit = case.ModelPeriodEmissionLimit
221
222     def create_model_file(self, root, file):
223         """ Creates the model file necessary for the project to run
224
225         Args:
226             root (str): File path of root to start the search from
227                 file ([type]): File path of model file
228             """
229
230         # Finds the file
231         # data = Find_File(data_file,model_file)
232         model_location = os.path.join(root, file)
233         df = pd.read_excel(model_location, sheet_name='Model')
234         # Creates a new dataframe based on the variables on the Include
235         column values
236         df_Include = df[df.Include == 'Yes']
237         df_model = df_Include[['Name']].copy()
238
239         # Creates a file location and write the model to a text file

```

```

229     model_txt = 'GOCPI_OseMOSYS_Model.txt'
230     model_location = os.path.join(root, model_txt)
231
232     # Saves the user defined model to a text file
233     np.savetxt(model_location, df_model.values, fmt='%s')
234
235     def create_data_file(self, file_location, defaults_dictionary,
236                         toggle_defaults):
237         """ Creates the osemosys datafile
238
239         Args:
240             file_location (str): String of directory to save data file
241             defaults_dictionary (dict): Dictionary setting the default
242             values for parameters
243             toggle_defaults (Bool): Boolean (True/False to only print
244             the default functions
245
246             """
247             # Opens the file for write the data
248             with open(file_location, 'w') as f:
249                 # Sets up the preamble for the data file
250                 f.write('# GOCPI Energy System Data File\n')
251                 f.write(
252                     '# Insert instructions when the file is running
properly\n')
253                 f.write('#\n')
254                 # Sets
255                 f.write('# Sets\n#\n')
256                 # year
257                 set_string = ' '.join(self.year)
258                 f.write('set YEAR\t:=\t{0};\n'.format(set_string))
259                 # region
260                 set_string = ' '.join(self.region)
261                 f.write('set REGION\t:=\t{0};\n'.format(set_string))
262                 # emission
263                 set_string = ' '.join(self.emission)
264                 f.write('set EMISSION\t:=\t{0};\n'.format(set_string))
265                 # technology
266                 set_string = ' '.join(self.technology)
267                 f.write('set TECHNOLOGY\t:=\t{0};\n'.format(set_string))
268                 # fuel
269                 set_string = ' '.join(self.fuel)
270                 f.write('set FUEL\t:=\t{0};\n'.format(set_string))
271                 # timeslice
272                 set_string = ' '.join(self.timeslice)
273                 f.write('set TIMESLICE\t:=\t{0};\n'.format(set_string))
274                 # mode_of_operation
275                 set_string = ' '.join(self.mode_of_operation)
276                 f.write('set MODE_OF_OPERATION\t:=\t{0};\n'.format(
277                     set_string))
278                 # storage
279                 set_string = ' '.join(self.storage)
280                 f.write('set STORAGE\t:=\t{0};\n'.format(set_string))
281                 # daytype
282                 set_string = ' '.join(self.daytype)
283                 f.write('set DAYTYPE\t:=\t{0};\n'.format(set_string))
284                 # season
285                 set_string = ' '.join(self.season)
286                 f.write('set SEASON\t:=\t{0};\n'.format(set_string))

```

```

283     # dailytimebracket
284     set_string = ' '.join(self.dailytimebracket)
285     f.write('set DAILYTIMEBRACKET\t:=\t{0};\n'.format(
286         set_string))
287     f.write('#\n')
288     # Parameters
289
290     # YearSplit = np.zeros((ll,ly))
291     param = 'YearSplit'
292     f.write('#\n')
293     columns = self.year
294     column_string = ' '.join(columns)
295     # Writes index specific parameter values to the text files
296     if toggle_defaults[param] == True:
297         f.write("param\t{0}\tdefault\t{1}:=\t{2}:=\n".format(
298             param, defaults_dictionary[param], column_string))
299         # Converts maxtrix rows to list
300         array = np.array(self.timeslice)
301         array = array.T
302         lt = array.tolist()
303         # Creates 2D matrix for this value
304         mat = self.YearSplit[:, :]
305         # Converts combined matrix to list and combines lists
306         matlist = mat.tolist()
307         #Combines the two lists
308         combined_list = list(zip(lt, matlist))
309         for line in combined_list:
310             combinedflat = ''.join(str(line))
311             combinedflat = combinedflat.replace('[', '')
312             combinedflat = combinedflat.replace(']', '')
313             combinedflat = combinedflat.replace('"', '')
314             combinedflat = combinedflat.replace(',', '')
315             combinedflat = combinedflat.replace('(', '')
316             combinedflat = combinedflat.replace(')', '')
317             f.write("{0}\n".format(combinedflat))
318         else:
319             f.write("param\t{0}\tdefault\t{1}:=\n".format(
320                 param, defaults_dictionary[param]))
321             f.write(';\n')
322
323     # DiscountRate = np.zeros((lr))
324     param = 'DiscountRate'
325     f.write('#\n')
326     if toggle_defaults[param] == True:
327         f.write("param\t{0}\tdefault\t{1}:=\t{2}:=\n".format(
328             param, defaults_dictionary[param], column_string))
329         # Converts maxtrix rows to list
330         array = np.array(self.region)
331         array = array.T
332         lt = array.tolist()
333         # Creates 2D matrix for this value
334         mat = self.DiscountRate[:, :]
335         # Converts combined matrix to list and combines lists
336         matlist = mat.tolist()
337         #Combines the two lists
338         combined_list = list(zip(lt, matlist))
339         # Writes index specific parameter values to the text
files

```

```

339         for line in combined_list:
340             combinedflat = ''.join(str(line))
341             combinedflat = combinedflat.replace('[', '')
342             combinedflat = combinedflat.replace(']', '')
343             combinedflat = combinedflat.replace('"', '')
344             combinedflat = combinedflat.replace("'", '')
345             combinedflat = combinedflat.replace("(", '')
346             combinedflat = combinedflat.replace(")", '')
347             f.write("{}\n".format(combinedflat))
348     else:
349         f.write("param\t{}\tdefault\t{}:=\n".format(
350             param, defaults_dictionary[param]))
351     f.write(';\n')
352
353     # DaySplit = np.zeros((llh,ly))
354     param = 'DaySplit'
355     f.write('#\n')
356     columns = self.year
357     column_string = ' '.join(columns)
358     # Writes index specific parameter values to the text files
359     if toggle_defaults[param] == True:
360         f.write("param\t{}\tdefault\t{}:=\t{}:=\n".format(
361             param, defaults_dictionary[param], column_string))
362         # Converts maxtrix rows to list
363         array = np.array(self.dailytimebracket)
364         array = array.T
365         lt = array.tolist()
366         # Creates 2D matrix for this value
367         mat = self.DaySplit[:, :]
368         # Converts combined matrix to list and combines lists
369         matlist = mat.tolist()
370         #Combines the two lists
371         combined_list = list(zip(lt, matlist))
372         # Writes index specific parameter values to the text
373         files
374             f.write("param\t{}\t{}:=\n".format(param,
375             column_string))
376             for line in combined_list:
377                 combinedflat = ''.join(str(line))
378                 combinedflat = combinedflat.replace('[', '')
379                 combinedflat = combinedflat.replace(']', '')
380                 combinedflat = combinedflat.replace('"', '')
381                 combinedflat = combinedflat.replace("'", '')
382                 combinedflat = combinedflat.replace("(", '')
383                 combinedflat = combinedflat.replace(")", '')
384                 f.write("{}\n".format(combinedflat))
385     else:
386         f.write("param\t{}\tdefault\t{}:=\n".format(
387             param, defaults_dictionary[param]))
388     f.write(';\n')
389
390     # Conversionls = np.zeros((ll,ls))
391     param = 'Conversionls' # Change this line
392     f.write('#\n')
393     columns = self.season # Change this line
394     column_string = ' '.join(columns)
395     # Writes index specific parameter values to the text files
396     if toggle_defaults[param] == True:

```

```

395         f.write("param\t{0}\tdefault\t{1}:\t{2}:=\n".format(
396             param, defaults_dictionary[param], column_string))
397     # Converts maxtrix rows to list
398     array = np.array(self.timeslice) # Change this line
399     array = array.T
400     lt = array.tolist()
401     # Creates 2D matrix for this value
402     mat = self.Conversions[:, :] # Change this line
403     # Converts combined matrix to list and combines lists
404     matlist = mat.tolist()
405     #Combines the two lists
406     combined_list = list(zip(lt, matlist))
407     for line in combined_list:
408         combinedflat = ''.join(str(line))
409         combinedflat = combinedflat.replace('[', '')
410         combinedflat = combinedflat.replace(']', '')
411         combinedflat = combinedflat.replace('"', '')
412         combinedflat = combinedflat.replace(',', '')
413         combinedflat = combinedflat.replace('(', '')
414         combinedflat = combinedflat.replace(')', '')
415         f.write("{}\n".format(combinedflat))
416     else:
417         f.write("param\t{0}\tdefault\t{1}:=\n".format(
418             param, defaults_dictionary[param]))
419     f.write(';\n')
420
421     # ConversionId = np.zeros((ll,lld))
422     param = 'ConversionId' # Change this line
423     f.write('#\n')
424     columns = self.dtype # Change this line
425     column_string = ' '.join(columns)
426     if toggle_defaults[param] == True:
427         f.write("param\t{0}\tdefault\t{1}:\t{2}:=\n".format(
428             param, defaults_dictionary[param], column_string))
429     # Converts maxtrix rows to list
430     array = np.array(self.timeslice) # Change this line
431     array = array.T
432     lt = array.tolist()
433     # Creates 2D matrix for this value
434     mat = self.ConversionId[:, :] # Change this line
435     # Converts combined matrix to list and combines lists
436     matlist = mat.tolist()
437     #Combines the two lists
438     combined_list = list(zip(lt, matlist))
439     for line in combined_list:
440         combinedflat = ''.join(str(line))
441         combinedflat = combinedflat.replace('[', '')
442         combinedflat = combinedflat.replace(']', '')
443         combinedflat = combinedflat.replace('"', '')
444         combinedflat = combinedflat.replace(',', '')
445         combinedflat = combinedflat.replace('(', '')
446         combinedflat = combinedflat.replace(')', '')
447         f.write("{}\n".format(combinedflat))
448     else:
449         f.write("param\t{0}\tdefault\t{1}:=\n".format(
450             param, defaults_dictionary[param]))
451     f.write(';\n')
452

```

```

453     # Conversionlh = np.zeros((ll, llh))
454     param = 'Conversionlh' # Change this line
455     f.write('#\n')
456     columns = self.dailytimebracket # Change this line
457     column_string = ' '.join(columns)
458     if toggle_defaults[param] == True:
459         f.write("param\t{0}\tdefault\t{1}:=\n".format(
460             param, defaults_dictionary[param], column_string))
461         # Converts maxtrix rows to list
462         array = np.array(self.timeslice) # Change this line
463         array = array.T
464         lt = array.tolist()
465         # Creates 2D matrix for this value
466         mat = self.Conversionlh[:, :] # Change this line
467         # Converts combined matrix to list and combines lists
468         matlist = mat.tolist()
469         #Combines the two lists
470         combined_list = list(zip(lt, matlist))
471         for line in combined_list:
472             combinedflat = ''.join(str(line))
473             combinedflat = combinedflat.replace('[', '')
474             combinedflat = combinedflat.replace(']', '')
475             combinedflat = combinedflat.replace('"', '')
476             combinedflat = combinedflat.replace(',', '')
477             combinedflat = combinedflat.replace('(', '')
478             combinedflat = combinedflat.replace(')', '')
479             f.write("{0}\n".format(combinedflat))
480         else:
481             f.write("param\t{0}\tdefault\t{1}:=\n".format(
482                 param, defaults_dictionary[param]))
483             f.write(';\n')
484
485         # DaysInDayType = np.zeros((lls, lld, ly))
486         #Writes new line character at parameter metadata to the
text file
487         param = 'DaysInDayType'
488         f.write('#\n')
489         f.write("param\t{0}\tdefault\t{1}:=\n".format(
490             param, defaults_dictionary[param]))
491         if toggle_defaults[param] == True:
492             # Writes parameter values to the text files
493             for k in range(self.ly):
494                 # Sets index value for format string
495                 y = self.year[k]
496                 # Converts matrix columns to strings columns to
strings
497                 columns = self.daytype
498                 column_string = ' '.join(columns)
499                 # Converts maxtrix rows to list
500                 array = np.array(self.season)
501                 array = array.T
502                 lt = array.tolist()
503                 # Creates 2D matrix for this value
504                 mat = self.DaysInDayType[:, :, k]
505                 # Converts combined matrix to list and combines
lists
506                 matlist = mat.tolist()
507                 #Combines the two lists

```

```

508                     combined_list = list(zip(lt, matlist))
509                     # Writes index specific parameter values to the
510                     text files
511                     f.write("\t[*,{0}]:\t{1}\t:=\n".format(y,
512                     column_string))
513                     for line in combined_list:
514                         combinedflat = ''.join(str(line))
515                         combinedflat = combinedflat.replace('[', '')
516                         combinedflat = combinedflat.replace(']', '')
517                         combinedflat = combinedflat.replace('"', '')
518                         combinedflat = combinedflat.replace(',', '')
519                         combinedflat = combinedflat.replace('(', '')
520                         combinedflat = combinedflat.replace(')', '')
521                         f.write("{0}\n".format(combinedflat))
522                     f.write(';\n')
523
524                     # TradeRoute = np.zeros((lr,lr,lf,ly))
525                     param = 'TradeRoute' # Change this line
526                     f.write('#\n')
527                     f.write("param\t{0}\tdefault\t{1}:=\n".format(
528                         param, defaults_dictionary[param]))
529                     if toggle_defaults[param] == True:
530                         # Writes parameter values to the text files
531                         for j in range(self.lf):
532                             # Sets index value for format string
533                             fl = self.fuel[j]
534                             for k in range(self.ly):
535                                 # Sets index value for format string
536                                 y = self.year[k]
537                                 # Converts matrix columns to strings columns to
538                                 strings
539                                 columns = self.region
540                                 column_string = ', '.join(columns)
541                                 # Converts maxtrix rows to list
542                                 array = np.array(self.region)
543                                 array = array.T
544                                 lt = array.tolist()
545                                 # Creates 2D matrix for this value
546                                 mat = self.TradeRoute[:, :, j, k]
547                                 # Converts combined matrix to list and combines
548                                 lists
549                                 matlist = mat.tolist()
550                                 #Combines the two lists
551                                 combined_list = list(zip(lt, matlist))
552                                 # Writes index specific parameter values to the
553                                 text files
554                                 f.write("\t[*,{0},{1}]:\t{2}\t:=\n".format(
555                                     fl, y, column_string))
556                                 for line in combined_list:
557                                     combinedflat = ''.join(str(line))
558                                     combinedflat = combinedflat.replace('[', '')
559                                     combinedflat = combinedflat.replace(']', '')
560                                     combinedflat = combinedflat.replace('"', '')
561                                     combinedflat = combinedflat.replace(',', '')
562                                     combinedflat = combinedflat.replace('(', '')
563                                     combinedflat = combinedflat.replace(')', '')
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
272
```

```

557                     combinedflat = combinedflat.replace("(", " ")
558                     combinedflat = combinedflat.replace(")", " ")
559                     f.write("{0}\n".format(combinedflat))
560 f.write(';\n')
561
562 # DepreciationMethod = np.zeros((lr))
563 param = 'DepreciationMethod'
564 f.write('#\n')
565 f.write("param\t{0}\tdefault\t{1}:=\n".format(
566     param, defaults_dictionary[param]))
567 if toggle_defaults[param] == True:
568     # Converts maxtrix rows to list
569     array = np.array(self.region)
570     array = array.T
571     lt = array.tolist()
572     # Creates 2D matrix for this value
573     mat = self.DepreciationMethod[:, :]
574     # Converts combined matrix to list and combines lists
575     matlist = mat.tolist()
576     #Combines the two lists
577     combined_list = list(zip(lt, matlist))
578     # Writes index specific parameter values to the text
579 files
580     for line in combined_list:
581         combinedflat = ','.join(str(line))
582         combinedflat = combinedflat.replace('[', ' ')
583         combinedflat = combinedflat.replace(']', ' ')
584         combinedflat = combinedflat.replace('"', ' ')
585         combinedflat = combinedflat.replace(',', ' ')
586         combinedflat = combinedflat.replace('(', ' ')
587         combinedflat = combinedflat.replace(')', ' ')
588         f.write("{0}\n".format(combinedflat))
589 f.write(';\n')
590
591 # SpecifiedAnnualDemand = np.zeros((lr, lsf, ly))
592 param = 'SpecifiedAnnualDemand'
593 f.write('#\n')
594 f.write("param\t{0}\tdefault\t{1}:=\n".format(
595     param, defaults_dictionary[param]))
596 if toggle_defaults[param] == True:
597     # Writes parameter values to the text files
598     for k in range(self.ly):
599         # Sets index value for format string
600         y = self.year[k]
601         # Converts matrix columns to strings columns to
602 strings
603             columns = self.specified_fuel
604             column_string = ', '.join(columns)
605             # Converts maxtrix rows to list
606             array = np.array(self.region)
607             array = array.T
608             lt = array.tolist()
609             # Creates 2D matrix for this value
610             mat = self.SpecifiedAnnualDemand[:, :, k]
611             # Converts combined matrix to list and combines
612 lists

```

```

610             matlist = mat.tolist()
611             #Combines the two lists
612             combined_list = list(zip(lt, matlist))
613             # Writes index specific parameter values to the
614             text files
615             f.write("\t[*,{0}]:\t{1}\t:=\n".format(y,
616             column_string))
617             for line in combined_list:
618                 combinedflat = ','.join(str(line))
619                 combinedflat = combinedflat.replace('[', '')
620                 combinedflat = combinedflat.replace(']', '')
621                 combinedflat = combinedflat.replace('"', '')
622                 combinedflat = combinedflat.replace("'", '')
623                 combinedflat = combinedflat.replace("(", '')
624                 combinedflat = combinedflat.replace(")", '')
625                 f.write("{0}\n".format(combinedflat))
626
627             f.write(';\n')
628
629             # SpecifiedDemandProfile = np.zeros((lr,lf,ll,ly))
630             param = 'SpecifiedDemandProfile' # Change this line
631             f.write('#\n')
632             f.write("param\t{0}\tdefault\t{1}:=\n".format(
633                 param, defaults_dictionary[param]))
634             if toggle_defaults[param] == True:
635                 # Writes parameter values to the text files
636                 for j in range(self.ll):
637                     # Sets index value for format string
638                     x = self.timeslice[j]
639                     for k in range(self.ly):
640                         # Sets index value for format string
641                         y = self.year[k]
642                         # Converts matrix columns to strings columns to
643                         # strings
644                         columns = self.specified_fuel
645                         column_string = ', '.join(columns)
646                         # Converts maxtrix rows to list
647                         array = np.array(self.region)
648                         array = array.T
649                         lt = array.tolist()
650                         # Creates 2D matrix for this value
651                         mat = self.SpecifiedDemandProfile[:, :, j, k]
652                         # Converts combined matrix to list and combines
653                         lists
654                         matlist = mat.tolist()
655                         #Combines the two lists
656                         combined_list = list(zip(lt, matlist))
657                         # Writes index specific parameter values to the
658                         text files
659                         f.write("\t[*,{0},{1}]:\t{2}\t:=\n".format(
660                             x, y, column_string))
661                         for line in combined_list:
662                             combinedflat = ','.join(str(line))
663                             combinedflat = combinedflat.replace('[', '')
664                             combinedflat = combinedflat.replace(']', '')
665                             combinedflat = combinedflat.replace('"', '')
666                             combinedflat = combinedflat.replace("'", '')
667
668
669

```

```

660             combinedflat = combinedflat.replace(", ", ',')
661             combinedflat = combinedflat.replace("( ", ',')
662             combinedflat = combinedflat.replace(") ", ',')
663             f.write("{0}\n".format(combinedflat))
664             f.write(';\n')
665
666     # AccumulatedAnnualDemand = np.zeros((lr,lf,ly))
667     param = 'AccumulatedAnnualDemand'
668     f.write('#\n')
669     f.write("param\t{0}\tdefault\t{1}:=\n".format(
670         param, defaults_dictionary[param]))
671     if toggle_defaults[param] == True:
672         # Writes parameter values to the text files
673         for k in range(self.ly):
674             # Sets index value for format string
675             y = self.year[k]
676             # Converts matrix columns to strings columns to
677             # strings
678             columns = self.accumulated_fuel
679             column_string = ', '.join(columns)
680             # Converts maxtrix rows to list
681             array = np.array(self.region)
682             array = array.T
683             lt = array.tolist()
684             # Creates 2D matrix for this value
685             mat = self.AccumulatedAnnualDemand[:, :, k]
686             # Converts combined matrix to list and combines
687             # lists
688             matlist = mat.tolist()
689             # Combines the two lists
690             combined_list = list(zip(lt, matlist))
691             # Writes index specific parameter values to the
692             # text files
693             f.write("\t[*,*,{0}]:\t{1}\t:=\n".format(y,
694                 column_string))
695             for line in combined_list:
696                 combinedflat = ''.join(str(line))
697                 combinedflat = combinedflat.replace('[', ',')
698                 combinedflat = combinedflat.replace(']', ',')
699                 combinedflat = combinedflat.replace('"', ',')
700                 combinedflat = combinedflat.replace("'", ',')
701                 combinedflat = combinedflat.replace("(", ',')
702                 combinedflat = combinedflat.replace(")", ',')
703                 f.write("{0}\n".format(combinedflat))
704                 f.write(';\n')
705
706     # CapacityToActivityUnit = np.zeros((lr,lt))
707     param = 'CapacityToActivityUnit' # Change this line
708     f.write('#\n')
709     columns = self.capacity_technology # Change this line
710     column_string = ', '.join(columns)
711     if toggle_defaults[param] == True:
712         f.write("param\t{0}\tdefault\t{1}:\t{2}:=\n".format(
713             param, defaults_dictionary[param], column_string))
714         # Converts maxtrix rows to list

```

```

711         array = np.array(self.region) # Change this line
712         array = array.T
713         lt = array.tolist()
714         # Creates 2D matrix for this value
715         mat = self.CapacityToActivityUnit[:, :] # Change this
716
717         line
718
719         # Converts combined matrix to list and combines lists
720         matlist = mat.tolist()
721         #Combines the two lists
722         combined_list = list(zip(lt, matlist))
723         for line in combined_list:
724             combinedflat = ','.join(str(line))
725             combinedflat = combinedflat.replace('[', '')
726             combinedflat = combinedflat.replace(']', '')
727             combinedflat = combinedflat.replace('"', '')
728             combinedflat = combinedflat.replace(',', '')
729             combinedflat = combinedflat.replace('(', '')
730             combinedflat = combinedflat.replace(')', '')
731             f.write("{}\n".format(combinedflat))
732
733
734         # CapacityFactor = np.zeros((lr,lt,ll,ly))
735         param = 'CapacityFactor' # Change this line
736         f.write('#\n')
737         f.write("param\t{0}\tdefault\t{1}:=\n".format(
738             param, defaults_dictionary[param]))
739         f.write(';\n')
740
741         # CapacityFactor = np.zeros((lr,lt,ll,ly))
742         param = 'CapacityFactor' # Change this line
743         f.write('#\n')
744         f.write("param\t{0}\tdefault\t{1}:=\n".format(
745             param, defaults_dictionary[param]))
746         if toggle_defaults[param] == True:
747             # Writes parameter values to the text files
748             for j in range(self.ll):
749                 # Sets index value for format string
750                 x = self.timeslice[j]
751                 for k in range(self.ly):
752                     # Sets index value for format string
753                     y = self.year[k]
754                     # Converts matrix columns to strings columns to
755                     # strings
756
757                     columns = self.capacity_technology
758                     column_string = ', '.join(columns)
759                     # Converts maxtrix rows to list
760                     array = np.array(self.region)
761                     array = array.T
762                     lt = array.tolist()
763                     # Creates 2D matrix for this value
764                     mat = self.CapacityFactor[:, :, j, k]
765                     # Converts combined matrix to list and combines
766                     lists
767
768                     matlist = mat.tolist()
769                     #Combines the two lists
770                     combined_list = list(zip(lt, matlist))
771                     # Writes index specific parameter values to the
772                     # text files
773
774                     f.write("\t[*,*,{0},{1}]:\t{2}\t:=\n".format(
775                         x, y, column_string))
776                     for line in combined_list:
777                         combinedflat = ', '.join(str(line))

```

```

765             combinedflat = combinedflat.replace('[', '')
766             combinedflat = combinedflat.replace(']', '')
767             combinedflat = combinedflat.replace('"', '')
768             combinedflat = combinedflat.replace(',', '')
769             combinedflat = combinedflat.replace("(", '')
770             combinedflat = combinedflat.replace(")", '')
771         )
772         f.write("{}\n".format(combinedflat))
773     f.write(';\n')
774
775     # AvailabilityFactor = np.zeros((lr,lt,ly))
776     param = 'AvailabilityFactor'
777     f.write('#\n')
778     f.write("param\t{0}\tdefault\t{1}:=\n".format(
779         param, defaults_dictionary[param]))
780     if toggle_defaults[param] == True:
781         # Writes parameter values to the text files
782         for k in range(self.ly):
783             # Sets index value for format string
784             y = self.year[k]
785             # Converts matrix columns to strings columns to
786             # strings
787             columns = self.availability_technology
788             column_string = ', '.join(columns)
789             # Converts maxtrix rows to list
790             array = np.array(self.region)
791             array = array.T
792             lt = array.tolist()
793             # Creates 2D matrix for this value
794             mat = self.AvailabilityFactor[:, :, k]
795             # Converts combined matrix to list and combines
796             # lists
797             matlist = mat.tolist()
798             #Combines the two lists
799             combined_list = list(zip(lt, matlist))
800             # Writes index specific parameter values to the
801             # text files
802             f.write("\t[*,*,{0}]:\t{1}\t:=\n".format(y,
803                 column_string))
804             for line in combined_list:
805                 combinedflat = ''.join(str(line))
806                 combinedflat = combinedflat.replace('[', '')
807                 combinedflat = combinedflat.replace(']', '')
808                 combinedflat = combinedflat.replace('"', '')
809                 combinedflat = combinedflat.replace(',', '')
810                 combinedflat = combinedflat.replace("(", '')
811                 combinedflat = combinedflat.replace(")", '')
812             f.write("{}\n".format(combinedflat))
813         f.write(';\n')
814
815     # OperationalLife = np.zeros((lr,lt))
816     param = 'OperationalLife' # Change this line
817     f.write('#\n')

```

```

813     columns = self.technology # Change this line
814     column_string = ', '.join(columns)
815     if toggle_defaults[param] == True:
816         f.write("param\t{0}\tdefault\t{1}:=\n".format(
817             param, defaults_dictionary[param], column_string))
818     # Converts maxtrix rows to list
819     array = np.array(self.region) # Change this line
820     array = array.T
821     lt = array.tolist()
822     # Creates 2D matrix for this value
823     mat = self.OperationalLife[:, :] # Change this line
824     # Converts combined matrix to list and combines lists
825     matlist = mat.tolist()
826     #Combines the two lists
827     combined_list = list(zip(lt, matlist))
828     for line in combined_list:
829         combinedflat = ''.join(str(line))
830         combinedflat = combinedflat.replace('[', '')
831         combinedflat = combinedflat.replace(']', '')
832         combinedflat = combinedflat.replace('"', '')
833         combinedflat = combinedflat.replace("'", '')
834         combinedflat = combinedflat.replace('(', '')
835         combinedflat = combinedflat.replace(')', '')
836         f.write("{0}\n".format(combinedflat))
837     else:
838         f.write("param\t{0}\tdefault\t{1}:=\n".format(
839             param, defaults_dictionary[param]))
840     f.write(';\n')

841
842     # ResidualCapacity = np.zeros((lr,lt,ly))
843     param = 'ResidualCapacity'
844     f.write('#\n')
845     f.write("param\t{0}\tdefault\t{1}:=\n".format(
846             param, defaults_dictionary[param]))
847     if toggle_defaults[param] == True:
848         # Writes parameter values to the text files
849         for k in range(self.ly):
850             # Sets index value for format string
851             y = self.year[k]
852             # Converts matrix columns to strings columns to
853             strings
854                 columns = self.technology
855                 column_string = ', '.join(columns)
856                 # Converts maxtrix rows to list
857                 array = np.array(self.region)
858                 array = array.T
859                 lt = array.tolist()
860                 # Creates 2D matrix for this value
861                 mat = self.ResidualCapacity[:, :, k]
862                 # Converts combined matrix to list and combines
863                 lists
864                     matlist = mat.tolist()
865                     #Combines the two lists
866                     combined_list = list(zip(lt, matlist))
867                     # Writes index specific parameter values to the
868                     text files
869                         f.write("\t[*,*,{0}]:\t{1}\t:=\n".format(y,
870                                         column_string))

```

```

867         for line in combined_list:
868             combinedflat = ''.join(str(line))
869             combinedflat = combinedflat.replace('[', '')
870             combinedflat = combinedflat.replace(']', '')
871             combinedflat = combinedflat.replace('"', '')
872             combinedflat = combinedflat.replace("'", '')
873             combinedflat = combinedflat.replace("(", '')
874             combinedflat = combinedflat.replace(")", '')
875             f.write("{0}\n".format(combinedflat))
876             f.write(';\n')
877
878     # InputActivityRatio = np.zeros((lr,lt,lf,lm,ly))
879     param = 'InputActivityRatio' # Change this line
880     f.write('#\n')
881     f.write("param\t{0}\tdefault\t{1}:=\n".format(
882         param, defaults_dictionary[param]))
883     if toggle_defaults[param] == True:
884         # Writes parameter values to the text files
885         for i in range(self.lf): # Change loops if you need
886             # Sets index value for format string
887             x = self.fuel[i]
888             for j in range(self.lm):
889                 # Sets index value for format string
890                 y = self.mode_of_operation[j]
891                 for k in range(self.ly):
892                     # Sets index value for format string
893                     z = self.year[k]
894                     # Converts matrix columns to strings
895                     columns = self.technology
896                     column_string = ' '.join(columns)
897                     # Converts maxtrix rows to list
898                     array = np.array(self.region)
899                     array = array.T
900                     lt = array.tolist()
901                     # Creates 2D matrix for this value
902                     mat = self.InputActivityRatio[:, :, i, j, k]
903
904                     # Converts combined matrix to list and
905                     # combines lists
906                     matlist = mat.tolist()
907                     #Combines the two lists
908                     combined_list = list(zip(lt, matlist))
909                     # Writes index specific parameter values to
910                     # the text files
911                     f.write("\t[*,*,{0},{1},{2}]:\t{3}\t:=\n".
912                         format(
913                             x, y, z, column_string))
914                     for line in combined_list:
915                         combinedflat = ''.join(str(line))
916                         combinedflat = combinedflat.replace('[',
917                             ', ')
918                         combinedflat = combinedflat.replace(']',
919                             ', ')
920                         combinedflat = combinedflat.replace('"',
921                             ', ')
922                         combinedflat = combinedflat.replace("'", '',
923                             ', ')

```

```

916             combinedflat = combinedflat.replace("(")
917             combinedflat = combinedflat.replace(")")
918             f.write("{}\n".format(combinedflat))
919             f.write(';\n')
920
921             # OutputActivityRatio = np.zeros((lr,lt,lf,lm,ly))
922             param = 'OutputActivityRatio' # Change this line
923             f.write('#\n')
924             f.write("param\t{0}\tdefault\t{1}:=\n".format(
925                 param, defaults_dictionary[param]))
926             if toggle_defaults[param] == True:
927                 # Writes parameter values to the text files
928                 for i in range(self.lf): # Change loops if you need
929                     # Sets index value for format string
930                     x = self.fuel[i]
931                     for j in range(self.lm):
932                         # Sets index value for format string
933                         y = self.mode_of_operation[j]
934                         for k in range(self.ly):
935                             # Sets index value for format string
936                             z = self.year[k]
937                             # Converts matrix columns to strings
938                             columns = self.technology
939                             column_string = ', '.join(columns)
940                             # Converts maxtrix rows to list
941                             array = np.array(self.region)
942                             array = array.T
943                             lt = array.tolist()
944                             # Creates 2D matrix for this value
945                             mat = self.OutputActivityRatio[:, :, i, j,
946                               k]
947
948                             # Converts combined matrix to list and
949                             # combines lists
950                             matlist = mat.tolist()
951                             #Combines the two lists
952                             combined_list = list(zip(lt, matlist))
953                             # Writes index specific parameter values to
954                             # the text files
955                             f.write("\t[*,*,{0},{1},{2}]:\t{3}\t:=\n".
956                               format(
957                                 x, y, z, column_string))
958                             for line in combined_list:
959                                 combinedflat = ''.join(str(line))
960                                 combinedflat = combinedflat.replace('[',
961                                     ', ')
962                                 combinedflat = combinedflat.replace(']', '')
963                                 combinedflat = combinedflat.replace(';',
964                                     ', ')
965                                 combinedflat = combinedflat.replace(",",
966                                     ", ")
967                                 combinedflat = combinedflat.replace("(",
968                                     ")")
969                                 combinedflat = combinedflat.replace(")")
970

```

```

961                               f.write("{0}\n".format(combinedflat))
962     f.write(';\n')
963
964     # CapitalCost = np.zeros((lr,lt,ly))
965     param = 'CapitalCost'
966     f.write('#\n')
967     f.write("param\t{0}\tdefault\t{1}:=\n".format(
968         param, defaults_dictionary[param]))
969     if toggle_defaults[param] == True:
970         # Writes parameter values to the text files
971         for k in range(self.ly):
972             # Sets index value for format string
973             y = self.year[k]
974             # Converts matrix columns to strings columns to
975             strings
976             columns = self.technology
977             column_string = ', '.join(columns)
978             # Converts maxtrix rows to list
979             array = np.array(self.region)
980             array = array.T
981             lt = array.tolist()
982             # Creates 2D matrix for this value
983             mat = self.CapitalCost[:, :, k]
984             # Converts combined matrix to list and combines
985             lists
986             matlist = mat.tolist()
987             #Combines the two lists
988             combined_list = list(zip(lt, matlist))
989             # Writes index specific parameter values to the
990             text files
991             f.write("\t[*,*,{0}]:\t{1}\t:=\n".format(y,
992                 column_string))
993             for line in combined_list:
994                 combinedflat = ''.join(str(line))
995                 combinedflat = combinedflat.replace('[', '')
996                 combinedflat = combinedflat.replace(']', '')
997                 combinedflat = combinedflat.replace('"', '')
998                 combinedflat = combinedflat.replace(',', '')
999                 combinedflat = combinedflat.replace('(', '')
1000                combinedflat = combinedflat.replace(')', '')
1001                f.write("{0}\n".format(combinedflat))
1002
1003     # VariableCost = np.zeros((lr,lt,lm,ly))
1004     param = 'VariableCost' # Change this line
1005     f.write('#\n')
1006     f.write("param\t{0}\tdefault\t{1}:=\n".format(
1007         param, defaults_dictionary[param]))
1008     if toggle_defaults[param] == True:
1009         # Writes parameter values to the text files
1010         for j in range(self.lm):
1011             # Sets index value for format string
1012             x = self.mode_of_operation[j]
1013             for k in range(self.ly):
1014                 # Sets index value for format string
1015                 y = self.year[k]
1016                 # Converts matrix columns to strings columns to
1017                 strings

```

```

1014             columns = self.technology
1015             column_string = ' '.join(columns)
1016             # Converts maxtrix rows to list
1017             array = np.array(self.region)
1018             array = array.T
1019             lt = array.tolist()
1020             # Creates 2D matrix for this value
1021             mat = self.VariableCost[:, :, j, k]
1022             # Converts combined matrix to list and combines
1023             lists
1024             matlist = mat.tolist()
1025             #Combines the two lists
1026             combined_list = list(zip(lt, matlist))
1027             # Writes index specific parameter values to the
1028             # text files
1029             f.write("\t[*,*,{0},{1}]:\t{2}\t:=\n".format(
1030                 x, y, column_string))
1031             for line in combined_list:
1032                 combinedflat = ''.join(str(line))
1033                 combinedflat = combinedflat.replace('[', '')
1034                 combinedflat = combinedflat.replace(']', '')
1035                 combinedflat = combinedflat.replace('"', '')
1036                 combinedflat = combinedflat.replace(',', '')
1037                 combinedflat = combinedflat.replace("(", '')
1038                 combinedflat = combinedflat.replace(")", '')
1039                 f.write("{0}\n".format(combinedflat))
1040             f.write(';\n')
1041
1042             # FixedCost = np.zeros((lr,lt,ly))
1043             param = 'FixedCost'
1044             f.write('#\n')
1045             f.write("param\t{0}\tdefault\t{1}:=\n".format(
1046                 param, defaults_dictionary[param]))
1047             if toggle_defaults[param] == True:
1048                 # Writes parameter values to the text files
1049                 for k in range(self.ly):
1050                     # Sets index value for format string
1051                     y = self.year[k]
1052                     # Converts matrix columns to strings columns to
1053                     strings
1054                     columns = self.technology
1055                     column_string = ' '.join(columns)
1056                     # Converts maxtrix rows to list
1057                     array = np.array(self.region)
1058                     array = array.T
1059                     lt = array.tolist()
1060                     # Creates 2D matrix for this value
1061                     mat = self.FixedCost[:, :, k]
1062                     # Converts combined matrix to list and combines
1063                     lists
1064                     matlist = mat.tolist()
1065                     #Combines the two lists

```

```

1062                     combined_list = list(zip(lt, matlist))
1063                     # Writes index specific parameter values to the
1064                     text files
1065                     f.write("\t[*,{0}]:\t{1}\t:=\n".format(y,
1066                                         column_string))
1067                     for line in combined_list:
1068                         combinedflat = ''.join(str(line))
1069                         combinedflat = combinedflat.replace('[', ' ')
1070                         combinedflat = combinedflat.replace(']', ' ')
1071                         combinedflat = combinedflat.replace('"', ' ')
1072                         combinedflat = combinedflat.replace(',', ' ')
1073                         combinedflat = combinedflat.replace('(', ' ')
1074                         combinedflat = combinedflat.replace(')', ' ')
1075                         f.write("{0}\n".format(combinedflat))
1076                     f.write(';\n')
1077
1078                     # TechnologyToStorage = np.zeros((lr,lt,ls,lm))
1079                     param = 'TechnologyToStorage' # Change this line
1080                     f.write('#\n')
1081                     f.write("param\t{0}\tdefault\t{1}:=\n".format(
1082                         param, defaults_dictionary[param]))
1083                     if toggle_defaults[param] == True:
1084                         # Writes parameter values to the text files
1085                         for j in range(self.ls):
1086                             # Sets index value for format string
1087                             x = self.storage[j]
1088                             for k in range(self.lm):
1089                                 # Sets index value for format string
1090                                 y = self.mode_of_operation[k]
1091                                 # Converts matrix columns to strings columns to
1092                                 strings
1093                                     columns = self.technology
1094                                     column_string = ', '.join(columns)
1095                                     # Converts maxtrix rows to list
1096                                     array = np.array(self.region)
1097                                     array = array.T
1098                                     lt = array.tolist()
1099                                     # Creates 2D matrix for this value
1100                                     mat = self.TechnologyToStorage[:, :, j, k]
1101                                     # Converts combined matrix to list and combines
1102                                     lists
1103                                     matlist = mat.tolist()
1104                                     #Combines the two lists
1105                                     combined_list = list(zip(lt, matlist))
1106                                     # Writes index specific parameter values to the
1107                                     text files
1108                                     f.write("\t[*,{0},{1}]:\t{2}\t:=\n".format(
1109                                         x, y, column_string))
1110                                     for line in combined_list:
1111                                         combinedflat = ''.join(str(line))
1112                                         combinedflat = combinedflat.replace('[', ' ')
1113                                         combinedflat = combinedflat.replace(']', ' ')
1114                                         combinedflat = combinedflat.replace('"', ' ')
1115                                         combinedflat = combinedflat.replace(',', ' ')
1116                                         combinedflat = combinedflat.replace('(', ' ')
1117                                         combinedflat = combinedflat.replace(')', ' ')
1118                                         combinedflat = combinedflat.replace(' ', ' ')
1119                                         combinedflat = combinedflat.replace(' ', ' ')
1120                                         combinedflat = combinedflat.replace(' ', ' ')

```

```

1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
    combinedflat = combinedflat.replace("(", " ")
    combinedflat = combinedflat.replace(")", " ")
    f.write("{0}\n".format(combinedflat))
f.write(';\n')

# TechnologyFromStorage = np.zeros((lr,lt,ls,lm))
param = 'TechnologyFromStorage' # Change this line
f.write('#\n')
f.write("param\t{0}\tdefault\t{1}:=\n".format(
    param, defaults_dictionary[param]))
if toggle_defaults[param] == True:
    # Writes parameter values to the text files
    for j in range(self.ls):
        # Sets index value for format string
        x = self.storage[j]
        for k in range(self.lm):
            # Sets index value for format string
            y = self.mode_of_operation[k]
            # Converts matrix columns to strings columns to
            strings
            columns = self.technology
            column_string = ', '.join(columns)
            # Converts maxtrix rows to list
            array = np.array(self.region)
            array = array.T
            lt = array.tolist()
            # Creates 2D matrix for this value
            mat = self.TechnologyFromStorage[:, :, j, k]
            # 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("\t[*,*,{0},{1}]:\t{2}\t:=\n".format(
                x, y, 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))
f.write(';\n')

# StorageLevelStart = np.zeros((lr,ls))
param = 'StorageLevelStart' # Change this line

```

```

1158     f.write('#\n')
1159     columns = self.storage # Change this line
1160     column_string = ' '.join(columns)
1161     if toggle_defaults[param] == True:
1162         f.write("param\t{0}\tdefault\t{1}:\t{2}:=\n".format(
1163             param, defaults_dictionary[param], column_string))
1164     # Converts maxtrix rows to list
1165     array = np.array(self.region) # Change this line
1166     array = array.T
1167     lt = array.tolist()
1168     # Creates 2D matrix for this value
1169     mat = self.StorageLevelStart[:, :] # Change this line
1170     # Converts combined matrix to list and combines lists
1171     matlist = mat.tolist()
1172     #Combines the two lists
1173     combined_list = list(zip(lt, matlist))
1174     for line in combined_list:
1175         combinedflat = ''.join(str(line))
1176         combinedflat = combinedflat.replace('[', '')
1177         combinedflat = combinedflat.replace(']', '')
1178         combinedflat = combinedflat.replace('"', '')
1179         combinedflat = combinedflat.replace(',', '')
1180         combinedflat = combinedflat.replace("(", '')
1181         combinedflat = combinedflat.replace(")", '')
1182         f.write("{}\n".format(combinedflat))
1183     else:
1184         f.write("param\t{0}\tdefault\t{1}:=\n".format(
1185             param, defaults_dictionary[param]))
1186     f.write(';\n')
1187
1188     # StorageMaxChargeRate = np.zeros((lr,ls))
1189     param = 'StorageMaxChargeRate' # Change this line
1190     f.write('#\n')
1191     columns = self.storage # Change this line
1192     column_string = ' '.join(columns)
1193     if toggle_defaults[param] == True:
1194         f.write("param\t{0}\tdefault\t{1}:\t{2}:=\n".format(
1195             param, defaults_dictionary[param], column_string))
1196     # Converts maxtrix rows to list
1197     array = np.array(self.region) # Change this line
1198     array = array.T
1199     lt = array.tolist()
1200     # Creates 2D matrix for this value
1201     mat = self.StorageMaxChargeRate[:, :] # Change this
1202     line
1203     # Converts combined matrix to list and combines lists
1204     matlist = mat.tolist()
1205     #Combines the two lists
1206     combined_list = list(zip(lt, matlist))
1207     for line in combined_list:
1208         combinedflat = ''.join(str(line))
1209         combinedflat = combinedflat.replace('[', '')
1210         combinedflat = combinedflat.replace(']', '')
1211         combinedflat = combinedflat.replace('"', '')
1212         combinedflat = combinedflat.replace(',', '')
1213         combinedflat = combinedflat.replace("(", '')
1214         combinedflat = combinedflat.replace(")", '')
1215         f.write("{}\n".format(combinedflat))

```

```

1215     else:
1216         f.write("param\t{0}\tdefault\t{1}:=\n".format(
1217             param, defaults_dictionary[param]))
1218         f.write(';\n')
1219
1220     # StorageMaxDischargeRate = np.zeros((lr,ls))
1221     param = 'StorageMaxDischargeRate' # Change this line
1222     f.write('#\n')
1223     columns = self.storage # Change this line
1224     column_string = ' '.join(columns)
1225     if toggle_defaults[param] == True:
1226         f.write("param\t{0}\tdefault\t{1}:\t{2}:=\n".format(
1227             param, defaults_dictionary[param], column_string))
1228     # Converts maxtrix rows to list
1229     array = np.array(self.region) # Change this line
1230     array = array.T
1231     lt = array.tolist()
1232     # Creates 2D matrix for this value
1233     mat = self.StorageMaxDischargeRate[:, :] # Change this
1234     line
1235     # Converts combined matrix to list and combines lists
1236     matlist = mat.tolist()
1237     #Combines the two lists
1238     combined_list = list(zip(lt, matlist))
1239     # Writes index specific parameter values to the text
1240     files
1241         f.write("param\t{0}\t:{1}:=\n".format(param,
1242         column_string))
1243         for line in combined_list:
1244             combinedflat = ' '.join(str(line))
1245             combinedflat = combinedflat.replace('[', '')
1246             combinedflat = combinedflat.replace(']', '')
1247             combinedflat = combinedflat.replace('"', '')
1248             combinedflat = combinedflat.replace("'", '')
1249             combinedflat = combinedflat.replace("(", '')
1250             combinedflat = combinedflat.replace(")", '')
1251             f.write("{0}\n".format(combinedflat))
1252     f.write(';\n')
1253
1254     # MinStorageCharge = np.zeros((lr,ls,ly))
1255     param = 'MinStorageCharge'
1256     f.write('#\n')
1257     f.write("param\t{0}\tdefault\t{1}:=\n".format(
1258         param, defaults_dictionary[param]))
1259     if toggle_defaults[param] == True:
1260         # Writes parameter values to the text files
1261         for k in range(self.ly):
1262             # Sets index value for format string
1263             y = self.year[k]
1264             # Converts matrix columns to strings columns to
1265             strings
1266                 columns = self.storage
1267                 column_string = ' '.join(columns)
1268                 # Converts maxtrix rows to list
1269                 array = np.array(self.region)

```

```

1269         array = array.T
1270         lt = array.tolist()
1271         # Creates 2D matrix for this value
1272         mat = self.MinStorageCharge[:, :, k]
1273         # Converts combined matrix to list and combines
1274         lists
1275         matlist = mat.tolist()
1276         # Combines the two lists
1277         combined_list = list(zip(lt, matlist))
1278         # Writes index specific parameter values to the
1279         text files
1280         f.write("\t[*,*,{0}]:\t{1}\t:=\n".format(y,
1281                                         column_string))
1282         for line in combined_list:
1283             combinedflat = ''.join(str(line))
1284             combinedflat = combinedflat.replace('[', '')
1285             combinedflat = combinedflat.replace(']', '')
1286             combinedflat = combinedflat.replace('"', '')
1287             combinedflat = combinedflat.replace(',', '')
1288             combinedflat = combinedflat.replace('(', '')
1289             combinedflat = combinedflat.replace(')', '')
1290             f.write("{0}\n".format(combinedflat))
1291         f.write(';\n')
1292
1293         # OperationalLifeStorage = np.zeros((lr,ls))
1294         param = 'OperationalLifeStorage' # Change this line
1295         f.write('#\n')
1296         columns = self.storage # Change this line
1297         column_string = ' '.join(columns)
1298         if toggle_defaults[param] == True:
1299             f.write("param\t{0}\tdefault\t{1}:\t{2}:=\n".format(
1300                 param, defaults_dictionary[param], column_string))
1301             # Converts maxtrix rows to list
1302             array = np.array(self.region) # Change this line
1303             array = array.T
1304             lt = array.tolist()
1305             # Creates 2D matrix for this value
1306             mat = self.OperationalLifeStorage[:, :] # Change this
1307             line
1308             # Converts combined matrix to list and combines lists
1309             matlist = mat.tolist()
1310             # Combines the two lists
1311             combined_list = list(zip(lt, matlist))
1312             # Writes index specific parameter values to the text
1313             files
1314             f.write("param\t{0}\t:{1}:=\n".format(param,
1315                                         column_string))
1316             for line in combined_list:
1317                 combinedflat = ''.join(str(line))
1318                 combinedflat = combinedflat.replace('[', '')
1319                 combinedflat = combinedflat.replace(']', '')
1320                 combinedflat = combinedflat.replace('"', '')
1321                 combinedflat = combinedflat.replace(',', '')
1322                 combinedflat = combinedflat.replace('(', '')
1323                 combinedflat = combinedflat.replace(')', '')
1324                 f.write("{0}\n".format(combinedflat))
1325         else:
1326             f.write("param\t{0}\tdefault\t{1}:=\n".format(

```

```

1321                 param, defaults_dictionary[param]))
1322             f.write(';\n')
1323
1324         # CapitalCostStorage = np.zeros((lr,ls,ly))
1325         param = 'CapitalCostStorage'
1326         f.write('#\n')
1327         f.write("param\t{0}\tdefault\t{1}:=\n".format(
1328             param, defaults_dictionary[param]))
1329     if toggle_defaults[param] == True:
1330         # Writes parameter values to the text files
1331         for k in range(self.ly):
1332             # Sets index value for format string
1333             y = self.year[k]
1334             # Converts matrix columns to strings columns to
1335             strings
1336             columns = self.storage
1337             column_string = ', '.join(columns)
1338             # Converts maxtrix rows to list
1339             array = np.array(self.region)
1340             array = array.T
1341             lt = array.tolist()
1342             # Creates 2D matrix for this value
1343             mat = self.CapitalCostStorage[:, :, k]
1344             # Converts combined matrix to list and combines
1345             lists
1346             matlist = mat.tolist()
1347             #Combines the two lists
1348             combined_list = list(zip(lt, matlist))
1349             # Writes index specific parameter values to the
1350             text files
1351             f.write("\t[*,*,{0}]:\t{1}\t:=\n".format(y,
1352                 column_string))
1353             for line in combined_list:
1354                 combinedflat = ''.join(str(line))
1355                 combinedflat = combinedflat.replace('[', '')
1356                 combinedflat = combinedflat.replace(']', '')
1357                 combinedflat = combinedflat.replace('"', '')
1358                 combinedflat = combinedflat.replace(',', '')
1359                 combinedflat = combinedflat.replace('(', '')
1360                 combinedflat = combinedflat.replace(')', '')
1361                 f.write("{0}\n".format(combinedflat))
1362             f.write(';\n')
1363
1364         # ResidualStorageCapacity = np.zeros((lr,ls,ly))
1365         param = 'ResidualStorageCapacity'
1366         f.write('#\n')
1367         f.write("param\t{0}\tdefault\t{1}:=\n".format(
1368             param, defaults_dictionary[param]))
1369     if toggle_defaults[param] == True:
1370         # Writes parameter values to the text files
1371         for k in range(self.ly):
1372             # Sets index value for format string
1373             y = self.year[k]
1374             # Converts matrix columns to strings columns to
1375             strings
1376             columns = self.storage
1377             column_string = ', '.join(columns)
1378             # Converts maxtrix rows to list

```

```

1374         array = np.array(self.region)
1375         array = array.T
1376         lt = array.tolist()
1377         # Creates 2D matrix for this value
1378         mat = self.ResidualStorageCapacity[:, :, k]
1379         # Converts combined matrix to list and combines
1380         lists
1381
1382         matlist = mat.tolist()
1383         # Combines the two lists
1384         combined_list = list(zip(lt, matlist))
1385         # Writes index specific parameter values to the
1386         # text files
1387         f.write("\t[*,{0}]:\t{1}\t:=\n".format(y,
1388
1389         column_string))
1390
1391         for line in combined_list:
1392             combinedflat = ''.join(str(line))
1393             combinedflat = combinedflat.replace('[', '')
1394             combinedflat = combinedflat.replace(']', '')
1395             combinedflat = combinedflat.replace('"', '')
1396             combinedflat = combinedflat.replace(',', '')
1397             combinedflat = combinedflat.replace('(', '')
1398             combinedflat = combinedflat.replace(')', '')
1399             f.write("{0}\n".format(combinedflat))
1400
1401             f.write(';\n')
1402
1403             # CapacityOfOneTechnologyUnit = np.zeros((lr,lt,ly))
1404             param = 'CapacityOfOneTechnologyUnit'
1405             f.write('#\n')
1406             f.write("param\t{0}\tdefault\t{1}:=\n".format(
1407                 param, defaults_dictionary[param]))
1408             if toggle_defaults[param] == True:
1409                 # Writes parameter values to the text files
1410                 for k in range(self.ly):
1411                     # Sets index value for format string
1412                     y = self.year[k]
1413                     # Converts matrix columns to strings columns to
1414                     strings
1415
1416                     columns = self.technology
1417                     column_string = ''.join(columns)
1418                     # Converts maxtrix rows to list
1419                     array = np.array(self.region)
1420                     array = array.T
1421                     lt = array.tolist()
1422                     # Creates 2D matrix for this value
1423                     mat = self.CapacityOfOneTechnologyUnit[:, :, k]
1424                     # Converts combined matrix to list and combines
1425                     lists
1426
1427                     matlist = mat.tolist()
1428                     # Combines the two lists
1429                     combined_list = list(zip(lt, matlist))
1430                     # Writes index specific parameter values to the
1431                     # text files
1432                     f.write("\t[*,{0}]:\t{1}\t:=\n".format(y,
1433
1434                     column_string))
1435                     for line in combined_list:
1436                         combinedflat = ''.join(str(line))
1437                         combinedflat = combinedflat.replace('[', '')
1438                         combinedflat = combinedflat.replace(']', '')

```

```

1425             combinedflat = combinedflat.replace(" ", ',')
1426             combinedflat = combinedflat.replace(", ", ',')
1427             combinedflat = combinedflat.replace("( ", ',')
1428             combinedflat = combinedflat.replace(") ", ',')
1429             f.write("{0}\n".format(combinedflat))
1430             f.write(';\n')
1431
1432     # TotalAnnualMaxCapacity = np.zeros((lr,lt,ly))
1433     param = 'TotalAnnualMaxCapacity'
1434     f.write('#\n')
1435     f.write("param\t{0}\tdefault\t{1}:=\n".format(
1436         param, defaults_dictionary[param]))
1437     if toggle_defaults[param] == True:
1438         # Writes parameter values to the text files
1439         for k in range(self.ly):
1440             # Sets index value for format string
1441             y = self.year[k]
1442             # Converts matrix columns to strings columns to
1443             # strings
1444             columns = self.technology
1445             column_string = ', '.join(columns)
1446             # Converts maxtrix rows to list
1447             array = np.array(self.region)
1448             array = array.T
1449             lt = array.tolist()
1450             # Creates 2D matrix for this value
1451             mat = self.TotalAnnualMaxCapacity[:, :, k]
1452             # Converts combined matrix to list and combines
1453             # lists
1454             matlist = mat.tolist()
1455             #Combines the two lists
1456             combined_list = list(zip(lt, matlist))
1457             # Writes index specific parameter values to the
1458             # text files
1459             f.write("\t[*,*,{0}]:\t{1}\t:=\n".format(y,
1460                 column_string))
1461             for line in combined_list:
1462                 combinedflat = ''.join(str(line))
1463                 combinedflat = combinedflat.replace('[', ',')
1464                 combinedflat = combinedflat.replace(']', ',')
1465                 combinedflat = combinedflat.replace('"",', ',')
1466                 combinedflat = combinedflat.replace(",\"", ',')
1467                 combinedflat = combinedflat.replace("(\"", ',')
1468                 combinedflat = combinedflat.replace(")\\"", ',')
1469                 f.write("{0}\n".format(combinedflat))
1470                 f.write(';\n')
1471
1472     # TotalAnnualMinCapacity = np.zeros((lr,lt,ly))
1473     param = 'TotalAnnualMinCapacity'
1474     f.write('#\n')
1475     f.write("param\t{0}\tdefault\t{1}:=\n".format(
1476         param, defaults_dictionary[param]))
1477     if toggle_defaults[param] == True:
1478         # Writes parameter values to the text files
1479         for k in range(self.ly):
1480             # Sets index value for format string
1481             y = self.year[k]

```

```

1478             # Converts matrix columns to strings columns to
1479             strings
1480             columns = self.technology
1481             column_string = ', '.join(columns)
1482             # Converts maxtrix rows to list
1483             array = np.array(self.region)
1484             array = array.T
1485             lt = array.tolist()
1486             # Creates 2D matrix for this value
1487             mat = self.TotalAnnualMinCapacity[:, :, k]
1488             # Converts combined matrix to list and combines
1489             lists
1490             matlist = mat.tolist()
1491             #Combines the two lists
1492             combined_list = list(zip(lt, matlist))
1493             # Writes index specific parameter values to the
1494             text files
1495             f.write("\t[*,*,{0}]:\t{1}\t:=\n".format(y,
1496             column_string))
1497             for line in combined_list:
1498                 combinedflat = ''.join(str(line))
1499                 combinedflat = combinedflat.replace('[', '')
1500                 combinedflat = combinedflat.replace(']', '')
1501                 combinedflat = combinedflat.replace('"', '')
1502                 combinedflat = combinedflat.replace(',', '')
1503                 combinedflat = combinedflat.replace('(', '')
1504                 combinedflat = combinedflat.replace(')', '')
1505                 f.write("{0}\n".format(combinedflat))
1506             f.write(';\n')
1507
1508             # TotalAnnualMaxCapacityInvestment = np.zeros((lr,lt,ly))
1509             param = 'TotalAnnualMaxCapacityInvestment'
1510             f.write('#\n')
1511             f.write("param\t{0}\tdefault\t{1}:=\n".format(
1512                 param, defaults_dictionary[param]))
1513             if toggle_defaults[param] == True:
1514                 # Writes parameter values to the text files
1515                 for k in range(self.ly):
1516                     # Sets index value for format string
1517                     y = self.year[k]
1518                     # Converts matrix columns to strings columns to
1519                     strings
1520                     columns = self.technology
1521                     column_string = ', '.join(columns)
1522                     # Converts maxtrix rows to list
1523                     array = np.array(self.region)
1524                     array = array.T
1525                     lt = array.tolist()
1526                     # Creates 2D matrix for this value
1527                     mat = self.TotalAnnualMaxCapacityInvestment[:, :, k
1528 ]
1529                     # Converts combined matrix to list and combines
1530                     lists
1531                     matlist = mat.tolist()
1532                     #Combines the two lists
1533                     combined_list = list(zip(lt, matlist))
1534                     # Writes index specific parameter values to the
1535                     text files

```

```

1528             f.write("\t[*,*,{0}]:\t{1}\t:=\n".format(y,
1529                 column_string))
1530                 for line in combined_list:
1531                     combinedflat = ''.join(str(line))
1532                     combinedflat = combinedflat.replace('[', ' ')
1533                     combinedflat = combinedflat.replace(']', ' ')
1534                     combinedflat = combinedflat.replace('"', ' ')
1535                     combinedflat = combinedflat.replace(',', ' ')
1536                     combinedflat = combinedflat.replace('(', ' ')
1537                     combinedflat = combinedflat.replace(')', ' ')
1538                     f.write("{0}\n".format(combinedflat))
1539             f.write(';\n')
1540
1541             # TotalAnnualMinCapacityInvestment = np.zeros((lr,lt,ly))
1542             param = 'TotalAnnualMinCapacityInvestment'
1543             f.write('#\n')
1544             f.write("param\t{0}\tdefault\t{1}:=\n".format(
1545                 param, defaults_dictionary[param]))
1546             if toggle_defaults[param] == True:
1547                 # Writes parameter values to the text files
1548                 for k in range(self.ly):
1549                     # Sets index value for format string
1550                     y = self.year[k]
1551                     # Converts matrix columns to strings columns to
1552                     strings
1553                     columns = self.technology
1554                     column_string = ' '.join(columns)
1555                     # Converts maxtrix rows to list
1556                     array = np.array(self.region)
1557                     array = array.T
1558                     lt = array.tolist()
1559                     # Creates 2D matrix for this value
1560                     mat = self.TotalAnnualMinCapacityInvestment[:, :, k]
1561
1562                     # Converts combined matrix to list and combines
1563                     lists
1564                     matlist = mat.tolist()
1565                     #Combines the two lists
1566                     combined_list = list(zip(lt, matlist))
1567                     # Writes index specific parameter values to the
1568                     text files
1569                     f.write("\t[*,*,{0}]:\t{1}\t:=\n".format(y,
1570                         column_string))
1571                     for line in combined_list:
1572                         combinedflat = ''.join(str(line))
1573                         combinedflat = combinedflat.replace('[', ' ')
1574                         combinedflat = combinedflat.replace(']', ' ')
1575                         combinedflat = combinedflat.replace('"', ' ')
1576                         combinedflat = combinedflat.replace(',', ' ')
1577                         combinedflat = combinedflat.replace('(', ' ')
1578                         combinedflat = combinedflat.replace(')', ' ')
1579                         f.write("{0}\n".format(combinedflat))
1580             f.write(';\n')
1581
1582             # TotalTechnologyAnnualActivityLowerLimit= np.zeros((lr,lt,
1583             ly))
1584             param = 'TotalTechnologyAnnualActivityLowerLimit'
1585             f.write('#\n')

```

```

1579         f.write("param\t{0}\tdefault\t{1}:=\n".format(
1580             param, defaults_dictionary[param]))
1581     if toggle_defaults[param] == True:
1582         # Writes parameter values to the text files
1583         for k in range(self.ly):
1584             # Sets index value for format string
1585             y = self.year[k]
1586             # Converts matrix columns to strings columns to
1587             # strings
1588             columns = self.technology
1589             column_string = ', '.join(columns)
1590             # Converts maxtrix rows to list
1591             array = np.array(self.region)
1592             array = array.T
1593             lt = array.tolist()
1594             # Creates 2D matrix for this value
1595             mat = self.TotalTechnologyAnnualActivityLowerLimit
1596             [ :, :, k]
1597             # Converts combined matrix to list and combines
1598             lists
1599             matlist = mat.tolist()
1600             #Combines the two lists
1601             combined_list = list(zip(lt, matlist))
1602             # Writes index specific parameter values to the
1603             # text files
1604             f.write("\t[*,*,{0}]:\t{1}\t:=\n".format(y,
1605                 column_string))
1606             for line in combined_list:
1607                 combinedflat = ', '.join(str(line))
1608                 combinedflat = combinedflat.replace('[', '')
1609                 combinedflat = combinedflat.replace(']', '')
1610                 combinedflat = combinedflat.replace('"', '')
1611                 combinedflat = combinedflat.replace(',', '')
1612                 combinedflat = combinedflat.replace('(', '')
1613                 combinedflat = combinedflat.replace(')', '')
1614                 f.write("{0}\n".format(combinedflat))
1615             f.write(';\n')
1616
1617             # TotalTechnologyAnnualActivityUpperLimit = np.zeros((lr,lt
1618             ,ly))
1619             param = 'TotalTechnologyAnnualActivityUpperLimit'
1620             f.write('#\n')
1621             f.write("param\t{0}\tdefault\t{1}:=\n".format(
1622                 param, defaults_dictionary[param]))
1623             if toggle_defaults[param] == True:
1624                 # Writes parameter values to the text files
1625                 for k in range(self.ly):
1626                     # Sets index value for format string
1627                     y = self.year[k]
1628                     # Converts matrix columns to strings columns to
1629                     strings
1630                     columns = self.technology
1631                     column_string = ', '.join(columns)
1632                     # Converts maxtrix rows to list
1633                     array = np.array(self.region)
1634                     array = array.T
1635                     lt = array.tolist()
1636                     # Creates 2D matrix for this value

```

```

1630                         mat = self.TotalTechnologyAnnualActivityUpperLimit
1631                         [:, :, k]
1632                         # Converts combined matrix to list and combines
1633                         lists
1634                         matlist = mat.tolist()
1635                         #Combines the two lists
1636                         combined_list = list(zip(lt, matlist))
1637                         # Writes index specific parameter values to the
1638                         text files
1639                         f.write("\t[*,{0}]:\t{1}\t:={\n".format(y,
1640                                         column_string))
1641                         for line in combined_list:
1642                             combinedflat = ''.join(str(line))
1643                             combinedflat = combinedflat.replace('[', '')
1644                             combinedflat = combinedflat.replace(']', '')
1645                             combinedflat = combinedflat.replace('"', '')
1646                             combinedflat = combinedflat.replace(',', '')
1647                             combinedflat = combinedflat.replace('(', '')
1648                             combinedflat = combinedflat.replace(')', '')
1649                             f.write("{0}\n".format(combinedflat))
1650                         f.write(';\n')
1651
1652                         # TotalTechnologyModelPeriodActivityUpperLimit = np.zeros((
1653                         lr, lt))
1654                         param = 'TotalTechnologyModelPeriodActivityUpperLimit' ##
1655                         Change this line
1656                         f.write('#\n')
1657                         columns = self.technology # Change this line
1658                         column_string = ' '.join(columns)
1659                         if toggle_defaults[param] == True:
1660                             # Writes index specific parameter values to the text
1661                             files
1662                             f.write("param\t{0}\tdefault\t{1}:\t{2}:=\n".format(
1663                                 param, defaults_dictionary[param], column_string))
1664                             # Converts maxtrix rows to list
1665                             array = np.array(self.region) # Change this line
1666                             array = array.T
1667                             lt = array.tolist()
1668                             # Creates 2D matrix for this value
1669                             mat = self.TotalTechnologyModelPeriodActivityUpperLimit
1670                             [:, :] # Change this line
1671                             # Converts combined matrix to list and combines lists
1672                             matlist = mat.tolist()
1673                             #Combines the two lists
1674                             combined_list = list(zip(lt, matlist))
1675                             for line in combined_list:
1676                                 combinedflat = ''.join(str(line))
1677                                 combinedflat = combinedflat.replace('[', '')
1678                                 combinedflat = combinedflat.replace(']', '')
1679                                 combinedflat = combinedflat.replace('"', '')
1680                                 combinedflat = combinedflat.replace(',', '')
1681                                 combinedflat = combinedflat.replace('(', '')
1682                                 combinedflat = combinedflat.replace(')', '')
1683                                 f.write("{0}\n".format(combinedflat))
1684                         else:
1685                             f.write("param\t{0}\tdefault\t{1}:=\n".format(
1686                                 param, defaults_dictionary[param]))
1687                         f.write(';\n')

```

```

1680
1681     # TotalTechnologyModelPeriodActivityLowerLimit = np.zeros((
1682     lr,lt))
1683     param = 'TotalTechnologyModelPeriodActivityLowerLimit' # 
Change this line
1684     f.write('#\n')
1685     columns = self.technology # Change this line
1686     column_string = ' '.join(columns)
1687     if toggle_defaults[param] == True:
1688         # Writes index specific parameter values to the text
files
1689     f.write("param\t{0}\tdefault\t{1}:=\t{2}:=\n".format(
1690         param, defaults_dictionary[param], column_string))
1691     # Converts maxtrix rows to list
1692     array = np.array(self.region) # Change this line
1693     array = array.T
1694     lt = array.tolist()
1695     # Creates 2D matrix for this value
mat = self.TotalTechnologyModelPeriodActivityLowerLimit
1696     [:, :] # Change this line
1697     # Converts combined matrix to list and combines lists
1698     matlist = mat.tolist()
#Combines the two lists
1699     combined_list = list(zip(lt, matlist))
1700     for line in combined_list:
1701         combinedflat = ''.join(str(line))
1702         combinedflat = combinedflat.replace('[', '')
1703         combinedflat = combinedflat.replace(']', '')
1704         combinedflat = combinedflat.replace('"', '')
1705         combinedflat = combinedflat.replace(',', '')
1706         combinedflat = combinedflat.replace('(', '')
1707         combinedflat = combinedflat.replace(')', '')
1708         f.write("{0}\n".format(combinedflat))
1709     else:
1710         f.write("param\t{0}\tdefault\t{1}:=\t{2}:=\n".format(
1711             param, defaults_dictionary[param]))
1712     f.write(';\n')

1713
1714     # ReserveMarginTagTechnology = np.zeros((lr,lt,ly))
1715     param = 'ReserveMarginTagTechnology'
1716     f.write('#\n')
1717     f.write("param\t{0}\tdefault\t{1}:=\t{2}:=\n".format(
1718         param, defaults_dictionary[param]))
1719     if toggle_defaults[param] == True:
1720         # Writes parameter values to the text files
1721         for k in range(self.ly):
1722             # Sets index value for format string
1723             y = self.year[k]
1724             # Converts matrix columns to strings columns to
strings
1725             columns = self.technology
1726             column_string = ' '.join(columns)
1727             # Converts maxtrix rows to list
1728             array = np.array(self.region)
1729             array = array.T
1730             lt = array.tolist()
1731             # Creates 2D matrix for this value
mat = self.ReserveMarginTagTechnology[:, :, k]
1732

```

```

1733             # Converts combined matrix to list and combines
1734             lists
1735             matlist = mat.tolist()
1736             #Combines the two lists
1737             combined_list = list(zip(lt, matlist))
1738             # Writes index specific parameter values to the
1739             text files
1740             f.write("\t[*,*,{0}]:\t{1}\t:=\n".format(y,
1741               column_string))
1742             for line in combined_list:
1743                 combinedflat = ''.join(str(line))
1744                 combinedflat = combinedflat.replace('[', ' ')
1745                 combinedflat = combinedflat.replace(']', ' ')
1746                 combinedflat = combinedflat.replace('"', ' ')
1747                 combinedflat = combinedflat.replace(',', ' ')
1748                 combinedflat = combinedflat.replace('(', ' ')
1749                 combinedflat = combinedflat.replace(')', ' ')
1750                 f.write("{0}\n".format(combinedflat))
1751             f.write(';\n')
1752
1753             # ReserveMarginTagFuel = np.zeros((lr,lf,ly))
1754             param = 'ReserveMarginTagFuel'
1755             f.write('#\n')
1756             f.write("param\t{0}\tdefault\t{1}:=\n".format(
1757               param, defaults_dictionary[param]))
1758             if toggle_defaults[param] == True:
1759                 # Writes parameter values to the text files
1760                 for k in range(self.ly):
1761                     # Sets index value for format string
1762                     y = self.year[k]
1763                     # Converts matrix columns to strings columns to
1764                     strings
1765                     columns = self.fuel
1766                     column_string = ' '.join(columns)
1767                     # Converts maxtrix rows to list
1768                     array = np.array(self.region)
1769                     array = array.T
1770                     lt = array.tolist()
1771                     # Creates 2D matrix for this value
1772                     mat = self.ReserveMarginTagFuel[:, :, k]
1773                     # Converts combined matrix to list and combines
1774                     lists
1775                     matlist = mat.tolist()
1776                     #Combines the two lists
1777                     combined_list = list(zip(lt, matlist))
1778                     # Writes index specific parameter values to the
1779                     text files
1780                     f.write("\t[*,*,{0}]:\t{1}\t:=\n".format(y,
1781                       column_string))
1782                     for line in combined_list:
1783                         combinedflat = ''.join(str(line))
1784                         combinedflat = combinedflat.replace('[', ' ')
1785                         combinedflat = combinedflat.replace(']', ' ')
1786                         combinedflat = combinedflat.replace('"', ' ')
1787                         combinedflat = combinedflat.replace(',', ' ')
1788                         combinedflat = combinedflat.replace('(', ' ')
1789                         combinedflat = combinedflat.replace(')', ' ')
1790                         f.write("{0}\n".format(combinedflat))

```

```

1784     f.write(';\n')
1785
1786     # ReserveMargin = np.zeros((lr,ly))
1787     param = 'ReserveMargin' # Change this line
1788     f.write('#\n')
1789     columns = self.year # Change this line
1790     column_string = ' '.join(columns)
1791     if toggle_defaults[param] == True:
1792         # Writes index specific parameter values to the text
1793         files
1794             f.write("param\t{0}\tdefault\t{1}:=\n".format(
1795                 param, defaults_dictionary[param], column_string))
1796             # Converts maxtrix rows to list
1797             array = np.array(self.region) # Change this line
1798             array = array.T
1799             lt = array.tolist()
1800             # Creates 2D matrix for this value
1801             mat = self.ReserveMargin[:, :] # Change this line
1802             # Converts combined matrix to list and combines lists
1803             matlist = mat.tolist()
1804             #Combines the two lists
1805             combined_list = list(zip(lt, matlist))
1806             for line in combined_list:
1807                 combinedflat = ''.join(str(line))
1808                 combinedflat = combinedflat.replace('[', '')
1809                 combinedflat = combinedflat.replace(']', '')
1810                 combinedflat = combinedflat.replace('"', '')
1811                 combinedflat = combinedflat.replace("'", '')
1812                 combinedflat = combinedflat.replace("(", '')
1813                 combinedflat = combinedflat.replace(")", '')
1814                 f.write("{0}\n".format(combinedflat))
1815             else:
1816                 f.write("param\t{0}\tdefault\t{1}:=\n".format(
1817                     param, defaults_dictionary[param]))
1818             f.write(';\n')
1819
1820     # RETagTechnology = np.zeros((lr,lt,ly))
1821     param = 'RETagTechnology'
1822     f.write('#\n')
1823     f.write("param\t{0}\tdefault\t{1}:=\n".format(
1824         param, defaults_dictionary[param]))
1825     if toggle_defaults[param] == True:
1826         # Writes parameter values to the text files
1827         for k in range(self.ly):
1828             # Sets index value for format string
1829             y = self.year[k]
1830             # Converts matrix columns to strings columns to
1831             strings
1832                 columns = self.technology
1833                 column_string = ' '.join(columns)
1834                 # Converts maxtrix rows to list
1835                 array = np.array(self.region)
1836                 array = array.T
1837                 lt = array.tolist()
1838                 # Creates 2D matrix for this value
1839                 mat = self.RETagTechnology[:, :, k]
1840                 # Converts combined matrix to list and combines
1841                 lists

```

```

1839         matlist = mat.tolist()
1840         #Combines the two lists
1841         combined_list = list(zip(lt, matlist))
1842         # Writes index specific parameter values to the
1843         text files
1844         f.write("\t[*,{0}]:\t{1}\t:=\n".format(y,
1845             column_string))
1846         for line in combined_list:
1847             combinedflat = ''.join(str(line))
1848             combinedflat = combinedflat.replace('[', '')
1849             combinedflat = combinedflat.replace(']', '')
1850             combinedflat = combinedflat.replace('"', '')
1851             combinedflat = combinedflat.replace(',', '')
1852             combinedflat = combinedflat.replace('(', '')
1853             combinedflat = combinedflat.replace(')', '')
1854             f.write("{0}\n".format(combinedflat))
1855             f.write(';\n')
1856
1857         # RETagFuel = np.zeros((lr,lf,ly))
1858         param = 'RETagFuel'
1859         f.write('#\n')
1860         f.write("param\t{0}\tdefault\t{1}:=\n".format(
1861             param, defaults_dictionary[param]))
1862         if toggle_defaults[param] == True:
1863             # Writes parameter values to the text files
1864             for k in range(self.ly):
1865                 # Sets index value for format string
1866                 y = self.year[k]
1867                 # Converts matrix columns to strings columns to
1868                 # strings
1869                 columns = self.fuel
1870                 column_string = ', '.join(columns)
1871                 # Converts maxtrix rows to list
1872                 array = np.array(self.region)
1873                 array = array.T
1874                 lt = array.tolist()
1875                 # Creates 2D matrix for this value
1876                 mat = self.RETagFuel[:, :, k]
1877                 # Converts combined matrix to list and combines
1878                 lists
1879                 matlist = mat.tolist()
1880                 #Combines the two lists
1881                 combined_list = list(zip(lt, matlist))
1882                 # Writes index specific parameter values to the
1883                 text files
1884                 f.write("\t[*,{0}]:\t{1}\t:=\n".format(y,
1885                     column_string))
1886                 for line in combined_list:
1887                     combinedflat = ''.join(str(line))
1888                     combinedflat = combinedflat.replace('[', '')
1889                     combinedflat = combinedflat.replace(']', '')
1890                     combinedflat = combinedflat.replace('"', '')
1891                     combinedflat = combinedflat.replace(',', '')
1892                     combinedflat = combinedflat.replace('(', '')
1893                     combinedflat = combinedflat.replace(')', '')
1894                     f.write("{0}\n".format(combinedflat))
1895                     f.write(';\n')
1896

```

```

1891     # REMinProductionTarget = np.zeros((lr,ly))
1892     param = 'REMinProductionTarget' # Change this line
1893     f.write('#\n')
1894     columns = self.year # Change this line
1895     column_string = ' '.join(columns)
1896     if toggle_defaults[param] == True:
1897         # Writes index specific parameter values to the text
1898         files
1899             f.write("param\t{0}\tdefault\t{1}:=\t{2}:=\n".format(
1900                 param, defaults_dictionary[param], column_string))
1901             # Converts maxtrix rows to list
1902             array = np.array(self.region) # Change this line
1903             array = array.T
1904             lt = array.tolist()
1905             # Creates 2D matrix for this value
1906             mat = self.REMinProductionTarget[:, :] # Change this
1907             line
1908                 # Converts combined matrix to list and combines lists
1909                 matlist = mat.tolist()
1910                 #Combines the two lists
1911                 combined_list = list(zip(lt, matlist))
1912                 for line in combined_list:
1913                     combinedflat = ','.join(str(line))
1914                     combinedflat = combinedflat.replace('[', '')
1915                     combinedflat = combinedflat.replace(']', '')
1916                     combinedflat = combinedflat.replace('"', '')
1917                     combinedflat = combinedflat.replace("'", '')
1918                     combinedflat = combinedflat.replace("(", '')
1919                     combinedflat = combinedflat.replace(")", '')
1920                     f.write("{0}\n".format(combinedflat))
1921             else:
1922                 f.write("param\t{0}\tdefault\t{1}:=\n".format(
1923                     param, defaults_dictionary[param]))
1924             f.write(';\n')
1925
1926             # EmissionActivityRatio = np.zeros((lr,lt,le,lm,ly))
1927             #Writes new line character at parameter metadata to the
1928             text file
1929                 param = 'EmissionActivityRatio' # Change this line
1930                 f.write('#\n')
1931                 f.write("param\t{0}\tdefault\t{1}:=\n".format(
1932                     param, defaults_dictionary[param]))
1933                 if toggle_defaults[param] == True:
1934                     # Writes parameter values to the text files
1935                     for i in range(self.le): # Change loops if you need
1936                         # Sets index value for format string
1937                         emission = self.emission[i]
1938                         for j in range(self.lm):
1939                             # Sets index value for format string
1940                             MOO = self.mode_of_operation[j]
1941                             for k in range(self.ly):
1942                                 # Sets index value for format string
1943                                 y = self.year[k]
1944                                 # Converts matrix columns to strings
1945
1946             columns to strings
1947                 columns = self.technology
1948                 column_string = ' '.join(columns)
1949                 # Converts maxtrix rows to list

```

```

1945             array = np.array(self.region)
1946             array = array.T
1947             lt = array.tolist()
1948             # Creates 2D matrix for this value
1949             mat = self.EmissionActivityRatio[:, :, i, j
1950
1951             , k]
1952
1953             # Converts combined matrix to list and
1954             # combines lists
1955
1956             matlist = mat.tolist()
1957             # Combines the two lists
1958             combined_list = list(zip(lt, matlist))
1959             # Writes index specific parameter values to
1960             # the text files
1961             f.write("\t[*,{0},{1},{2}]:\t{3}\t:=\n".
1962             format(
1963                 emission, MOO, y, column_string))
1964             for line in combined_list:
1965                 combinedflat = ''.join(str(line))
1966                 combinedflat = combinedflat.replace('[',
1967                 ', ')
1968                 combinedflat = combinedflat.replace(']',
1969                 ', ')
1970                 combinedflat = combinedflat.replace('"',
1971                 ', ')
1972                 combinedflat = combinedflat.replace(',', '')
1973                 combinedflat = combinedflat.replace("((",
1974                 combinedflat = combinedflat.replace("))"
1975                 f.write("{0}\n".format(combinedflat))
1976                 f.write(';\\n')
1977
1978             # EmissionsPenalty = np.zeros((lr,le,ly))
1979             param = 'EmissionsPenalty'
1980             f.write('#\\n')
1981             f.write("param\t{0}\tdefault\t{1}:=\n".format(
1982                 param, defaults_dictionary[param]))
1983             if toggle_defaults[param] == True:
1984                 # Writes parameter values to the text files
1985                 for k in range(self.ly):
1986                     # Sets index value for format string
1987                     y = self.year[k]
1988                     # Converts matrix columns to strings columns to
1989                     strings
1990                     columns = self.emission
1991                     column_string = ', '.join(columns)
1992                     # Converts maxtrix rows to list
1993                     array = np.array(self.region)
1994                     array = array.T
1995                     lt = array.tolist()
1996                     # Creates 2D matrix for this value
1997                     mat = self.EmissionsPenalty[:, :, k]
1998                     # Converts combined matrix to list and combines
1999                     lists
2000
2001                     matlist = mat.tolist()
2002                     # Combines the two lists
2003                     combined_list = list(zip(lt, matlist))

```

```

1991                         # Writes index specific parameter values to the
1992     text files
1993     f.write("\t[*,{0}]:\t{1}\t:=\n".format(y,
1994         column_string))
1995     for line in combined_list:
1996         combinedflat = ''.join(str(line))
1997         combinedflat = combinedflat.replace('[', ' ')
1998         combinedflat = combinedflat.replace(']', ' ')
1999         combinedflat = combinedflat.replace('"', ' ')
2000         combinedflat = combinedflat.replace(',', ' ')
2001         combinedflat = combinedflat.replace('(', ' ')
2002         combinedflat = combinedflat.replace(')', ' ')
2003         f.write("{0}\n".format(combinedflat))
2004     f.write(';\n')
2005
2006     # AnnualExogenousEmission = np.zeros((lr,le,ly))
2007     param = 'AnnualExogenousEmission'
2008     f.write('#\n')
2009     f.write("param\t{0}\tdefault\t{1}:=\n".format(
2010         param, defaults_dictionary[param]))
2011     if toggle_defaults[param] == True:
2012         # Writes parameter values to the text files
2013         for k in range(self.ly):
2014             # Sets index value for format string
2015             y = self.year[k]
2016             # Converts matrix columns to strings columns to
2017             # strings
2018             columns = self.emission
2019             column_string = ' '.join(columns)
2020             # Converts maxtrix rows to list
2021             array = np.array(self.region)
2022             array = array.T
2023             lt = array.tolist()
2024             # Creates 2D matrix for this value
2025             mat = self.AnnualExogenousEmission[:, :, k]
2026             # Converts combined matrix to list and combines
2027             lists
2028             matlist = mat.tolist()
2029             #Combines the two lists
2030             combined_list = list(zip(lt, matlist))
2031             # Writes index specific parameter values to the
2032             text files
2033             f.write("\t[*,{0}]:\t{1}\t:=\n".format(y,
2034                 column_string))
2035             for line in combined_list:
2036                 combinedflat = ''.join(str(line))
2037                 combinedflat = combinedflat.replace('[', ' ')
2038                 combinedflat = combinedflat.replace(']', ' ')
2039                 combinedflat = combinedflat.replace('"', ' ')
2040                 combinedflat = combinedflat.replace(',', ' ')
2041                 combinedflat = combinedflat.replace('(', ' ')
2042                 combinedflat = combinedflat.replace(')', ' ')
2043                 f.write("{0}\n".format(combinedflat))
2044             f.write(';\n')
2045
2046     # AnnualEmissionLimit = np.zeros((lr,le,ly))
2047     param = 'AnnualEmissionLimit'
2048     f.write('#\n')

```

```

2043         f.write("param\t{0}\tdefault\t{1}:=\n".format(
2044             param, defaults_dictionary[param]))
2045     if toggle_defaults[param] == True:
2046         # Writes parameter values to the text files
2047         for k in range(self.ly):
2048             # Sets index value for format string
2049             y = self.year[k]
2050             # Converts matrix columns to strings columns to
2051             # strings
2052             columns = self.emission
2053             column_string = ' '.join(columns)
2054             # Converts maxtrix rows to list
2055             array = np.array(self.region)
2056             array = array.T
2057             lt = array.tolist()
2058             # Creates 2D matrix for this value
2059             mat = self.AnnualExogenousEmission[:, :, k]
2060             # Converts combined matrix to list and combines
2061             # lists
2062             matlist = mat.tolist()
2063             #Combines the two lists
2064             combined_list = list(zip(lt, matlist))
2065             # Writes index specific parameter values to the
2066             # text files
2067             f.write("\t[*,{0}]:\t{1}\t:=\n".format(y,
2068                 column_string))
2069             for line in combined_list:
2070                 combinedflat = ''.join(str(line))
2071                 combinedflat = combinedflat.replace('[', '')
2072                 combinedflat = combinedflat.replace(']', '')
2073                 combinedflat = combinedflat.replace('"', '')
2074                 combinedflat = combinedflat.replace(',', '')
2075                 combinedflat = combinedflat.replace('(', '')
2076                 combinedflat = combinedflat.replace(')', '')
2077                 f.write("{0}\n".format(combinedflat))
2078             f.write(';\n')
2079
2080             # ModelPeriodExogenousEmission = np.zeros((lr,le))
2081             param = 'ModelPeriodExogenousEmission' # Change this line
2082             f.write('#\n')
2083             columns = self.emission # Change this line
2084             column_string = ' '.join(columns)
2085             if toggle_defaults[param] == True:
2086                 # Writes index specific parameter values to the text
2087                 # files
2088                 f.write("param\t{0}\tdefault\t{1}:\t{2}:=\n".format(
2089                     param, defaults_dictionary[param], column_string))
2090                 # Converts maxtrix rows to list
2091                 array = np.array(self.region) # Change this line
2092                 array = array.T
2093                 lt = array.tolist()
2094                 # Creates 2D matrix for this value
2095                 mat = self.ModelPeriodExogenousEmission[:, :] # Change
2096                 # this line
2097                 # Converts combined matrix to list and combines lists
2098                 matlist = mat.tolist()
2099                 #Combines the two lists
2100                 combined_list = list(zip(lt, matlist))

```

```

2095             # Writes index specific parameter values to the text
2096     files
2097         f.write("param\t{0}\t:{1}:=\n".format(param,
2098             column_string))
2099         for line in combined_list:
2100             combinedflat = ','.join(str(line))
2101             combinedflat = combinedflat.replace('[', '')
2102             combinedflat = combinedflat.replace(']', '')
2103             combinedflat = combinedflat.replace('"', '')
2104             combinedflat = combinedflat.replace(',', '')
2105             combinedflat = combinedflat.replace('(', '')
2106             combinedflat = combinedflat.replace(')', '')
2107             f.write("{0}\n".format(combinedflat))
2108     else:
2109         f.write("param\t{0}\tdefault\t{1}:=\n".format(
2110             param, defaults_dictionary[param]))
2111         f.write(';\n')
2112
2113         # ModelPeriodEmissionLimit = np.zeros((lr,le))
2114         param = 'ModelPeriodEmissionLimit' # Change this line
2115         f.write('#\n')
2116         columns = self.emission # Change this line
2117         column_string = ', '.join(columns)
2118         if toggle_defaults[param] == True:
2119             # Writes index specific parameter values to the text
2120             files
2121                 f.write("param\t{0}\tdefault\t{1}:\t{2}:=\n".format(
2122                     param, defaults_dictionary[param], column_string))
2123                 # Converts maxtrix rows to list
2124                 array = np.array(self.region) # Change this line
2125                 array = array.T
2126                 lt = array.tolist()
2127                 # Creates 2D matrix for this value
2128                 mat = self.ModelPeriodEmissionLimit[:, :] # Change
2129
2130                 this line
2131                 # Converts combined matrix to list and combines lists
2132                 matlist = mat.tolist()
2133                 #Combines the two lists
2134                 combined_list = list(zip(lt, matlist))
2135                 # Writes index specific parameter values to the text
2136             files
2137                 f.write("param\t{0}\t:{1}:=\n".format(param,
2138                     column_string))
2139                 for line in combined_list:
2140                     combinedflat = ','.join(str(line))
2141                     combinedflat = combinedflat.replace('[', '')
2142                     combinedflat = combinedflat.replace(']', '')
2143                     combinedflat = combinedflat.replace('"', '')
2144                     combinedflat = combinedflat.replace(',', '')
2145                     combinedflat = combinedflat.replace('(', '')
2146                     combinedflat = combinedflat.replace(')', '')
2147                     f.write("{0}\n".format(combinedflat))
2148     else:
2149         f.write("param\t{0}\tdefault\t{1}:=\n".format(
2150             param, defaults_dictionary[param]))
2151         f.write(';\n')
2152         f.write('end;\n')
2153         f.write('#')

```

2147 return

5.2.3 CreateCases

The module to create user-defined energy systems.

```

1 import os
2 import numpy as np
3 import pandas as pd
4
5
6 class CreateCases:
7     """ A class of methods to create user-defined energy system
8     """
9     def __init__(self):
10         """ Sets the parameters and sets for the energy case
11         """
12         # Sets (placeholders for setting values)
13         self.year = None
14         self.region = None
15         self.emission = None
16         self.technology = None
17         self.capacity_technology = None
18         self.availability_technology = None
19         self.fuel = None
20         self.specified_fuel = None
21         self.accumulated_fuel = None
22         self.timeslice = None
23         self.mode_of_operation = None
24         self.storage = None
25         self.daytype = None
26         self.season = None
27         self.dailytimebracket = None
28
29         # Parameters
30         self.Conversionls = None
31         self.Conversionld = None
32         self.Conversionlh = None
33         self.DaysInDayType = None
34         self.TradeRoute = None
35         self.DepreciationMethod = None
36         self.SpecifiedAnnualDemand = None
37         self.SpecifiedDemandProfile = None
38         self.AccumulatedAnnualDemand = None
39         self.CapacityToActivityUnit = None
40         self.CapacityFactor = None
41         self.AvailabilityFactor = None
42         self.OperationalLife = None
43         self.ResidualCapacity = None
44         self.InputActivityRatio = None
45         self.OutputActivityRatio = None
46         self.CapitalCost = None
47         self.VariableCost = None
48         self.FixedCost = None
49         self.TechnologyToStorage = None
50         self.TechnologyFromStorage = None
51         self.StorageLevelStart = None
52         self.StorageMaxChargeRate = None

```

```

53         self.StorageMaxDischargeRate = None
54         self.MinStorageCharge = None
55         self.OperationalLifeStorage = None
56         self.CapitalCostStorage = None
57         self.ResidualStorageCapacity = None
58         self.CapacityOfOneTechnologyUnit = None
59         self.TotalAnnualMaxCapacity = None
60         self.TotalAnnualMinCapacity = None
61         self.TotalAnnualMaxCapacityInvestment = None
62         self.TotalAnnualMinCapacityInvestment = None
63         self.TotalTechnologyAnnualActivityLowerLimit = None
64         self.TotalTechnologyAnnualActivityUpperLimit = None
65         self.TotalTechnologyModelPeriodActivityUpperLimit = None
66         self.TotalTechnologyModelPeriodActivityLowerLimit = None
67         self.ReserveMarginTagTechnology = None
68         self.ReserveMarginTagFuel = None
69         self.ReserveMargin = None
70         self.RETagTechnology = None
71         self.RETagFuel = None
72         self.REMinProductionTarget = None
73         self.EmissionActivityRatio = None
74         self.EmissionsPenalty = None
75         self.AnnualExogenousEmission = None
76         self.AnnualEmissionLimit = None
77         self.ModelPeriodExogenousEmission = None
78         self.ModelPeriodEmissionLimit = None
79
80     def set_year(self, start_year, end_year, interval):
81         """ Sets a list of forecast years
82
83         Args:
84             start_year (int): Starting year for forecasting (Less than
85             end_year)
86             end_year (int): Ending year for forecasting (Greater than
87             start_year)
88             interval (int): Gap for forecasting period
89             """
90
91         # Sets year array for new value
92         year = []
93         count = start_year
94         while count <= end_year:
95             year.append(str(count))
96             count = count + interval
97         self.year = year
98
99     def set_region(self, regions):
100        """ Sets the datacase's regions analysis
101
102        Args:
103            regions (list): list of regions
104            """
105        self.region = regions
106
107    def set_emission(self, emissions):
108        """Sets the cases emission types
109
110        Args:
111            emissions (List): list of emission types

```

```
109      """
110      self.emission = emissions
111
112  def set_technology(self, technology):
113      """ Sets the cases technology type
114
115      Args:
116          technology (list): List of technologies
117      """
118      self.technology = technology
119
120  def set_capacity_technology(self, capacity_technology):
121      """ Sets the cases capacity_technology type
122
123      Args:
124          capacity_technology (list): List of technologies
125      """
126      self.capacity_technology = capacity_technology
127
128  def set_availability_technology(self, availability_technology):
129      """ Sets the cases availability_technology type
130
131      Args:
132          availability_technology (list): List of technologies
133      """
134      self.availability_technology = availability_technology
135
136  def set_fuel(self, fuel):
137      """ Sets the case's fuel types
138
139      Args:
140          fuel (list): list of fuels
141      """
142      self.fuel = fuel
143
144  def set_specified_fuel(self, specified_fuel):
145      """ Sets the case's specified fuel types
146
147      Args:
148          specified_fuel (list): list of specified fuels
149      """
150      self.specified_fuel = specified_fuel
151
152  def set_accumulated_fuel(self, accumulated_fuel):
153      """ Sets the case's accumulated fuel types
154
155      Args:
156          specified_fuel (list): list of specified fuels
157      """
158      self.accumulated_fuel = accumulated_fuel
159
160  def set_timeslice(self, timeslice):
161      """ Set of timeslices
162
163      Args:
164          timeslice (list): list of timeslices
165      """
166      self.timeslice = timeslice
```

```
167
168     def set_mode_of_operation(self, num_modes_of_operation):
169         """ Create the number of modes of operation (n = 1, ..., num_modes_of_operation)
170
171         Args:
172             num_modes_of_operation (int): Number of modes of operation
173         """
174         # Create set of mode_of_operation
175         mode_of_operation = []
176         count = 1
177         while count <= num_modes_of_operation:
178             mode_of_operation.append(str(count))
179             count = count + 1
180         self.mode_of_operation = mode_of_operation
181
182     def set_storage(self, storage):
183         """ Sets storage set of the datacase
184
185         Args:
186             storage (list): list of storage types
187         """
188         self.storage = storage
189
190     def set_daytype(self, num_daytypes):
191         """ Sets the daytypes for the energy case
192
193         Args:
194             num_daytypes (int): Number of daytypes
195         """
196         # Create set of daytypes
197         daytype = []
198         count = 1
199         while count <= num_daytypes:
200             daytype.append(str(count))
201             count = count + 1
202         self.daytype = daytype
203
204     def set_season(self, num_seasons):
205         """ Creates set of seasons
206
207         Args:
208             num_seasons (int): Number of seasons
209         """
210         # Create set of seasons
211         season = []
212         count = 1
213         while count <= num_seasons:
214             season.append(str(count))
215             count = count + 1
216         self.season = season
217
218     def set_daily_time_bracket(self, num_dailymetimebrackets):
219         """ Creates set of daily time brackets
220
221         Args:
222             dailymetimebracket (int): Number of daily time brackets
223         """
```



```

269     Returns:
270         [int, array]: Numpy array of discount rates
271     """
272     # Creates empty dictionaries to stored values
273     annualised_returns = {}
274     cost_of_equity = {}
275     cost_of_debt = {}
276     cost_of_preference_equity = {}
277     WACC = {}
278     discount_rates = []
279     # Calculates
280     for region in market_index:
281         # Calculates annualised returns for each regions market
index
282         annualised_rate_of_return = (np.power(
283             (1 + ((market_index[region][-1] - market_index[region]
284             ][0]) /
285                 market_index[region][0])),
286             (12 / len(market_index[region]))) - 1)
287         annualised_returns[region] = annualised_rate_of_return
288         # Calculates cost of equity
289         cost_of_equity[region] = (
290             risk_free_rate[region] + (market_risk_coefficient[
291             region]) *
292                 (annualised_returns[region] - risk_free_rate[region]))
293         # Calculates cost of debt
294         cost_of_debt[region] = (cost_of_debt_pre_tax[region] / debt
295             [region]
296                 ) * (1 - effective_tax_rate[region])
297
298         # Calculates cost of preference equity
299         cost_of_preference_equity[region] = preference_dividends[
300             region] / market_value_preference_shares[region]
301         # Calculates WACC
302         WACC[region] = (
303             cost_of_equity[region] *
304                 (equity[region] /
305                     (equity[region] + debt[region] + preference_equity[
306             region])) +
307                 cost_of_debt[region] *
308                     (debt[region] /
309                         (equity[region] + debt[region] + preference_equity[
310             region])) +
311                 cost_of_preference_equity[region] *
312                     (preference_equity[region] /
313                         (equity[region] + debt[region] + preference_equity[
314             region])))
315         # Sets discount rates for each region
316         discount_rates.append(WACC[region])
317         # Set discount array
318         self.DiscountRate = np.asarray(discount_rates)

319     def set_day_split(self, daily_time_bracket, years, hour_split,
320         num_days,
321             num_hours):
322         """ Sets the day split parameter
323
324         Args:
325

```

```

318         daily_time_bracket (list): List of daily time brackets
319         years (list): List of years
320         hour_split (dict): Dictionary of hours in a daily time
321         bracket
322             num_days (int): Number of days in a year
323             num_hours (int): Number of hours in a day
324             """
325             # Initialises the DaySplit Array
326             DaySplit = np.ones((len(daily_time_bracket), len(years)))
327             index = 0
328             for split in daily_time_bracket:
329                 DaySplit[index, :] = hour_split[split] / (num_days *
330                 num_hours)
331                 index = index + 1
332             self.DaySplit = DaySplit
333
334     def set_conversion_ls(self, timeslice, season, link):
335         """ Sets the conversions parameter
336
337         Args:
338             timeslice (list): List of timeslices
339             season (list): List of seasons
340             link (dict): Dictionary describing the connection between
341             timeslices and seasons
342             """
343             Conversionsls = np.zeros((len(timeslice), len(season)))
344             for i in range(0, len(timeslice), 1):
345                 for j in range(0, len(season), 1):
346                     if link[timeslice[i]] == season[j]:
347                         Conversionsls[i, j] = 1
348
349             self.Conversionsls = Conversionsls
350
351     def set_conversion_ld(self, timeslice, daytype, link):
352         """ Sets the Conversionld parameter
353
354         Args:
355             timeslice (list): List of timeslices
356             daytype (list): List of daytypes
357             link (dict): Dictionary describing the connection between
358             timeslices and daytypes
359             """
360             Conversionld = np.zeros((len(timeslice), len(daytype)))
361             for i in range(0, len(timeslice), 1):
362                 Conversionld[i, :] = link[timeslice[i]]
363
364             self.Conversionld = Conversionld
365
366     def set_conversion_lh(self, timeslice, dailytimebracket, link,
367     override):
368         """ Sets the Conversionlh parameter
369
370         Args:
371             timeslice (list): List of timeslices
372             dailytimebracket (list): List of dailytimebracket
373             link (dict): Dictionary describing the connection between
374             timeslices and dailytimebrackets

```

```

369         override (int, array): Override if want to manually put in
370         the array
371         """
372         if override == None:
373             Conversionlh = np.zeros((len(timeslice), len(
374             dailytimebracket)))
375             for i in range(0, len(timeslice), 1):
376                 Conversionlh[i, :] = link[timeslice[i]]
377             self.Conversionlh = Conversionlh
378         else:
379             self.Conversionlh = override
380
381     def set_days_in_day_type(self, season, daytype, year, link,
382     override):
383         """ Sets the DaysInDayType parameter
384
385         Args:
386             season (list): List of seasons
387             daytype (list): List of daytypes
388             year (list): List of years
389             link (dict): Dictionary relating seasons to daytypes
390             override (int, array): Override if want to manually put in
391             the array
392             """
393             if override == None:
394                 DaysInDayType = np.zeros((len(season), len(daytype), len(
395                 year)))
396                 for i in range(0, len(season), 1):
397                     for j in range(0, len(year), 1):
398                         DaysInDayType[i, :, j] = link[season[i]]
399                     self.DaysInDayType = DaysInDayType
400             else:
401                 self.DaysInDayType = override
402
403     def set_trade_route(self, trade):
404         """ Sets the TradeRoute parameter between regions
405             (Assume it is the same across fuels and years)
406
407         Args:
408             trade (int ,array): 4D array representing trade
409             relationships
410                                         between regions, fuels and years. You
411                                         must model this manually.
412             """
413             self.TradeRoute = trade
414
415     def set_depreciation_method(self, region, methods, override):
416         """ Sets DepreciationMethod
417             (1 = Sinking Fund Depreciation, 2 = Straightline
418             Depreciation)
419
420         Args:
421             region (list): List of regions
422             override (int, array): Manual array for setting
423             depreciation methods
424             methods (dict): Dictionary assigning methods to regions
425             """
426

```

```

419         if override == None:
420             depreciation_method = np.ones((len(region)))
421             for i in range(0, len(region), 1):
422                 depreciation_method[i] = methods[region[i]]
423             self.DepreciationMethod = depreciation_method
424         else:
425             self.DepreciationMethod = override
426
427     def set_specified_annual_demand(self, specified_forecast):
428         """ Sets the annual demand for fuels per region over the
429         forecast period (Must be accurate)
430
431         Args:
432             forecast (float, array): The forecast array of size (len(
433             region), len(fuel), len(year))
434             """
435             self.SpecifiedAnnualDemand = specified_forecast
436
437     def set_specified_demand_profile(self, specified_annual_demand,
438         region,
439                     fuel, year, timeslice, profile,
440         override):
441         """ Sets the specified annual demand profiles using the
442         specified annual demand.
443
444         Args:
445             specified_annual_demand (float, array): Specified annual
446             demand profiles
447             region (list): List of regions
448             fuel (list): List of fuels
449             year (list): List of years
450             timeslice (list): List of timeslices
451             profile (Dict): Dictionary of fuel allocations to
452             timeslices
453             override (float, array): Manual override for the specified
454             annual demand profiles.
455             """
456             # Initialises the linear array
457             demand_profile = np.zeros(
458                 (len(region), len(fuel), len(timeslice), len(year)))
459             if override == None:
460                 # Calculates the demand profile
461                 for place in region:
462                     for fuel_type in fuel:
463                         for time in timeslice:
464                             for year_num in year:
465                                 region_index = region.index(place)
466                                 fuel_index = fuel.index(fuel_type)
467                                 timeslice_index = timeslice.index(time)
468                                 year_index = year.index(year_num)
469                                 demand_profile[region_index, fuel_index,
470                                     timeslice_index,
471                                     year_index] = profile[time]
472
473             self.SpecifiedDemandProfile = demand_profile
474         else:
475             self.SpecifiedDemandProfile = override
476
477

```

```

469     def set_accumulated_annual_demand(self, accumulated_forecast):
470         """ Sets the accumulated annual demand for fuels per region
471             over the forecast period.
472             This function relies on a similar forecasting methodology
473             as set_specific_demand.
474             Fuels set in this function cannot be defined in
475             set_specific_demand.
476
477             Args:
478                 accumulated_forecast (float, array): The forecast array of
479                 size (len(region),len(fuel),len(year))
480                 """
481             self.AccumulatedAnnualDemand = accumulated_forecast
482
483     def set_capacity_to_activity_unit(self, region, technology,
484                                         capacity_dictionaries, override):
485         """ Sets the capacity to activity parameter
486
487             Args:
488                 region (list): List of regions
489                 technology (list): List of technologies
490                 capacity_dictionaries (list): List of dictionaries to
491                 assign value
492                 override (float, array) = 2D Array to assign override
493                 values
494                 """
495             if override == None:
496                 cap_to_act = np.zeros((len(region), len(technology)))
497                 for i in range(0, len(capacity_dictionaries), 1):
498                     for j in range(0, len(technology), 1):
499                         cap_to_act[i, j] = capacity_dictionaries[i][
500                             technology[j]]
501                 self.CapacityToActivityUnit = cap_to_act
502             else:
503                 self.CapacityToActivityUnit = override
504
505     def set_capacity_factor(self, factor_matrix):
506         """ Sets capacity factors for conversion technologies.
507
508             Args:
509                 factor_matrix (float, array); Capacity Factors
510                 """
511             self.CapacityFactor = factor_matrix
512
513     def set_availability_factor(self, availability_matrix):
514         """ Sets the availability factors
515
516             Args:
517                 availability_matrix (float, array): Matrix describing
518                 availability factors for given technologies
519                 """
520             self.AvailabilityFactor = availability_matrix
521
522     def set_operational_life(self, operational_lives):
523         """ Sets operational life
524
525             Args:
526                 operational_lives (list):

```

```
519     """
520     self.OperationalLife = operational_lives
521
522     def set_residual_capacity(self, residential_capacities):
523         """ Set residual capacity
524
525         Args:
526             residential_capacities (float, array): residual capacities
527             parameter
528         """
529         self.ResidualCapacity = residential_capacities
530
531     def set_input_activity_ratio(self, input_activity_ratios):
532         """ Sets input activity ratios
533
534         Args:
535             input_activity_ratios (float, array): Sets the input
536             activity ratio
537         """
538         self.InputActivityRatio = input_activity_ratios
539
540     def set_output_activity_ratio(self, output_activity_ratios):
541         """ Sets output activity ratio
542
543         Args:
544             output_activity_ratios (float, array): output activity
545             ratio parameters
546         """
547
548     def set_capital_cost(self, capital_costs):
549         """ Sets capital costs
550
551         Args:
552             capital_costs (float, array): capital cost paramters
553         """
554         self.CapitalCost = capital_costs
555
556     def set_variable_cost(self, variable_costs):
557         """ Sets variable costs
558
559         Args:
560             variable_costs (float, array): variable costs parameters
561         """
562         self.VariableCost = variable_costs
563
564     def set_fixed_cost(self, fixed_costs):
565         """ Set fixed costs
566
567         Args:
568             fixed_costs (float, array): fixed cost parameters
569         """
570         self.FixedCost = fixed_costs
571
572     def set_technology_to_storage(self, technology_to_storage):
573         """ Sets the technology to storage parameter
574
575         Args:
```

```
573         technology_to_storage (float, array): technology to storage
574     parameter
575     """
576     self.TechnologyToStorage = technology_to_storage
577
578     def set_technology_from_storage(self, technology_from_storage):
579         """ Sets technology from storage binary paramter
580
581         Args:
582             technology_from_storage (float, array): technology from
583             storage parameter
584             """
585             self.TechnologyFromStorage = technology_from_storage
586
587     def set_min_storage_charge(self, minimum_storage_charges):
588         """ Sets the minimum storage charges
589
590         Args:
591             minimum_storage_charges (float, array): minimum storage
592             parameters
593             """
594             self.MinStorageCharge = minimum_storage_charges
595
596     def set_operational_life_storage(self, operational_life_storage):
597         """ Sets the operational life storage
598
599         Args:
600             operational_life_storage (float, array): operational life
601             storage parameters
602             """
603             self.OperationalLifeStorage = operational_life_storage
604
605     def set_capital_cost_storage(self, capital_cost_storage):
606         """ Sets the capital costs of using storage technologies
607
608         Args:
609             capital_cost_storage (float, array): capital cost of
610             storage technologies
611             """
612             self.CapitalCostStorage = capital_cost_storage
613
614     def set_storage_level_start(self, storage_level_start):
615         """ Sets the storage level starting point
616
617         Args:
618             storage_level_start (float, array): storage starting level
619             """
620             self.StorageLevelStart = storage_level_start
621
622     def set_storage_max_charge_rate(self,
623         storage_max_level_charge_rates):
624         """ Sets the storgae max charge rate
625
626         Args:
627             storage_max_level_charge_rates (float, array): Storage max
628             level charge rates
629             """
630             self.StorageMaxChargeRate = storage_max_level_charge_rates
```

```
624
625     def set_storage_max_discharge_rate(self ,
626
627         storage_max_level_discharge_rates):
628             """ Sets storage technologies maximum discharge rates
629
630             Args:
631                 storage_max_level_discharge_rates (float , array): Discharge
632                 rates for storage paramters
633                 """
634
635             self.StorageMaxDischargeRate =
636             storage_max_level_discharge_rates
637
638     def set_residual_storage_capacity(self , residual_storage_capacities
639     ):
640         """ Sets residual storage capacities
641
642             Args:
643                 residual_storage_capacities (float , array): residual
644                 storage capacities
645                 """
646
647             self.ResidualStorageCapacity = residual_storage_capacities
648
649     def set_capacity_of_one_technology_unit(self ,
650
651         capacity_of_one_technology_unit):
652             """ Set the capacity of one technology units for all
653                 technologies
654
655             Args:
656                 capacity_of_one_technology_unit (float , array): capacities
657                 for one technology units
658                 """
659
660             self.CapacityOfOneTechnologyUnit =
661             capacity_of_one_technology_unit
662
663     def set_total_annual_max_capacity(self , total_annual_max_capacities
664     ):
665         """ Sets the total annual maximum capacities
666
667             Args:
668                 total_annual_max_capacities (float , array): Total Annual
669                 Max Capacities
670                 """
671
672             self.TotalAnnualMaxCapacity = total_annual_max_capacities
673
674     def set_total_annual_min_capacity(self , total_annual_min_capacities
675     ):
676         """ Sets the totoal annual minimum capacities
677
678             Args:
679                 total_annual_min_capacities (float , array): Total Annual
680                 Min Capacities
681                 """
682
683             self.TotalAnnualMinCapacity = total_annual_min_capacities
684
685     def set_total_technology_annual_activity_lower_limit(
686                 self , total_technology_activity_lower_limits):
```

```

669     """ Sets the Total Technology Activity Lower Limits
670
671     Args:
672         total_technology_activity_lower_limits (float, array):
673             Technology Activity Lower Limits
674             """
675
676     def set_total_technology_annual_activity_upper_limit(
677         self, total_technology_annual_activity_upper_limits):
678         """ Sets the Total Technology Activity Upper Limits
679
680         Args:
681             total_technology_annual_activity_upper_limits (float, array)
682         ): Technology Activity Upper Limits
683             """
684
685         self.TotalTechnologyAnnualActivityUpperLimit =
686             total_technology_annual_activity_upper_limits
687
688     def set_total_technology_period_activity_upper_limit(
689         self, total_technology_period_activity_upper_limits):
690         """ Sets Total Technology Period Activity Upper Limits
691
692         Args:
693             total_technology_period_activity_upper_limits (float, array)
694         ): Total Technology Period Activity Upper Limit
695             """
696
697         self.TotalTechnologyModelPeriodActivityUpperLimit =
698             total_technology_period_activity_upper_limits
699
700     def set_total_technology_period_activity_lower_limit(
701         self, total_technology_period_activity_lower_limits):
702         """Sets Total Technology Period Activity Lower Limits
703
704         Args:
705             total_technology_period_activity_lower_limits ([type]):
706             Total Technology Period Activity Lower Limit
707             """
708
709         self.TotalTechnologyModelPeriodActivityLowerLimit =
710             total_technology_period_activity_lower_limits
711
712     def set_reserve_margin_tag_technology(self,
713
714         reserve_margin_tag_technologies):
715             """ Sets Reserve Margin Tag Technology
716
717             Args:
718                 reserve_margin_tag_technologies (float, array): Reserve
719                 Margin Tag Technologies
720                 """
721
722             self.ReserveMarginTagTechnology =
723                 reserve_margin_tag_technologies
724
725     def set_reserve_margin_tag_fuel(self, reserve_margin_fuel_tags):
726         """ Sets the reserve margin tag fuels
727
728             Args:

```

```
716             reserve_margin_fuel_tags (float, array): Sets the reserve
margin tag fuel parameters
717             """
718             self.ReserveMarginTagFuel = reserve_margin_fuel_tags
719
720     def set_reserve_margin(self, reserve_margins):
721         """ Sets reserve margins
722
723         Args:
724             reserve_margins (float, array): Reserve Margins
725             """
726             self.ReserveMargin = reserve_margins
727
728     def set_re_tag_technology(self, re_tag_technologies):
729         """ Sets RE Tag Technology
730
731         Args:
732             re_tag_technologies (float, array): RE Tag Technologies
733             """
734             self.RETagTechnology = re_tag_technologies
735
736     def set_re_tag_fuel(self, re_tag_fuels):
737         """ Sets RE Tag Fuels
738
739         Args:
740             re_tag_fuels (float, array): RE Tag Fuels
741             """
742             self.RETagFuel = re_tag_fuels
743
744     def set_re_min_production_target(self, re_min_production_targets):
745         """ Sets Renewable Energy Minimum Production Targets
746
747         Args:
748             re_min_production_targets (float, array): Renewable Energy
Minimum Production Targets
749             """
750             self.REMinProductionTarget = re_min_production_targets
751
752     def set_emission_activity_ratio(self, emission_activity_ratios):
753         """ Sets Emission Activity Ratios
754
755         Args:
756             emission_activity_ratios ([float, array]): Emission Activity
Ratios
757             """
758             self.EmissionActivityRatio = emission_activity_ratios
759
760     def set_emissions_penalty(self, emissions_penalties):
761         """ Sets Emissions Penalties
762
763         Args:
764             emissions_penalties (float, penalties): Emissions Penalties
765             """
766             self.EmissionsPenalty = emissions_penalties
767
768     def set_annual_exogenous_emission(self, annual_exogenous_emission):
769         """ Sets Annual Exogenous Emissions
770             """
```

```

771     Args:
772         annual_exogenous_emission (float, array): Annual Exogenous
773         Emissions
774         """
775         self.AnnualExogenousEmission = annual_exogenous_emission
776
776     def set_annual_emission_limit(self, annual_emission_limits):
777         """ Sets Annual Emission Limits
778
779         Args:
780             annual_emission_limits (float, array): Annual Emission
781             Limits
782             """
783             self.AnnualEmissionLimit = annual_emission_limits
784
784     def set_model_period_exogenous_emission(self,
785
785         model_period_exogenous_emissions):
786         """ Sets Model Period Exogenous Emissions
787
788         Args:
789             model_period_exogenous_emissions (float, array): Model
790             Period Exogenous Emissions
791             """
792             self.ModelPeriodExogenousEmission =
793             model_period_exogenous_emissions
794
793     def set_model_period_emission_limit(self,
794         model_period_emission_limits):
795         """ Sets Model Period Emission Limits
796
796         Args:
797             model_period_emission_limits (float, array): Model Period
798             Emission Limits
799             """
800             self.ModelPeriodEmissionLimit = model_period_emission_limits

```

5.2.4 Forecasting

The module to forecast energy and finance-related values.

```

1 import os
2 import numpy as np
3 import pandas as pd
4 import inquirer as iq
5
6
7 class Forecasting:
8     def __init__(self):
9         """Initialises the forecasting class
10         """
11         self.forecasts = None
12
13     def energy_balance_base(self, root, IEA_World_Energy_Balances_1,
14                             IEA_World_Energy_Balances_2,
15                             create_excel_spreadsheet, output_file):
16         """ Creates the baseline energy balance for forecasting
17

```

```

18     Args:
19         root (path): Path to provide access to all the files
20             IEA_World_Energy_Balances_1 (str): File name for Energy
21             Balance A to K
22             IEA_World_Energy_Balances_2 ([type]): File name for Energy
23             Balance L to Z
24             create_excel_spreadsheet (boolean): True/false on whether
25             to create a spreadsheet
26             output_file (str): Name of output energy balance
27             spreadsheet
28
29     Returns:
30         (dict): Dictionary of energy balances and unique lists (Use
31             these key words to access: Energy Balances, Fuel, Geography,
32             Technology)
33         """
34
35         IEAWEBAK = root / IEA_World_Energy_Balances_1
36         IEAWEBLZ = root / IEA_World_Energy_Balances_2
37
38         # Creates dataframes from IEA World Energy Statistics and
39         # Balances CSVs from Stats.OECD.org in the OECDLibrary
40         # Note the data is from #https://stats.oecd.org/ and #https://
41         # www-oecd-ilibrary-org.ezproxy.auckland.ac.nz/
42         column_headers = [
43             'ID', 'Unit', 'Geo_Code', 'Geo_Description', 'Prod_Code',
44             'Prod_Description', 'Flow_Code', 'Flow_Description', 'Year',
45             ,
46             'Value(TJ)'
47         ]
48         f1 = open(IEAWEBAK, 'r')
49         df_A = pd.read_csv(f1, header=None)
50         df_A.columns = column_headers
51         df_A.info(verbose=True)
52         f2 = open(IEAWEBLZ, 'r')
53         df_B = pd.read_csv(f2, header=None)
54         df_B.columns = column_headers
55         df_B.info(verbose=True)
56         frames = [df_A, df_B]
57         df = pd.concat(frames)
58         df.info(verbose=True)
59
60         # Closes the files
61         f1.close()
62         f2.close()
63
64         # Finds the unique items in each list of the energy balance
65         sheets
66         unique_fuel = df.Prod_Description.unique()
67         unique_geography = df.Geo_Description.unique()
68         unique_technology = df.Flow_Description.unique()
69         print(unique_geography)
70
71         # Asks for a user to select a geography using the inquirer
72         function
73         selected_geo = input(
74             "Please enter the geography you wish to extract energy
75             balances: "
76         )

```

```

64
65      # Creates a pivot table to display the data in the way similar
66      # to the Energy Balance Sheet (cols = Energy Product, rows = Energy
67      # Flows)
68      energy_balance_pivot_table = pd.pivot_table(
69          df,
70          index=['Geo_Description', 'Flow_Description'],
71          # Converted values to PJ
72          values=['Value(TJ)'],
73          columns=['Prod_Description'],
74          aggfunc=[np.sum],
75          fill_value=0)
76      # Filters to the geography the user has selected
77      Input_String = 'Geo_Description == ["' + selected_geo + '"]'
78      geography_energy_balance_pivot_table =
79      energy_balance_pivot_table.query(
80          Input_String)
81
82      if create_excel_spreadsheet == True:
83          # Write the filtered pivot table to an excel file
84          writer = pd.ExcelWriter(root / output_file)
85          geography_energy_balance_pivot_table.to_excel(writer,
86          selected_geo)
87          writer.save()
88
89      # Returns the unique lists and filtered pivot table as a
90      # dataframe
91      return {
92          "Energy Balances": geography_energy_balance_pivot_table,
93          "Fuel": unique_fuel,
94          "Geography": unique_geography,
95          "Technology": unique_technology
96      }
97
98
99      def calculate_constant_average_growth_rate(self, start_year,
100                                                 end_year,
101                                                 start_value, end_value):
102
103         """ Calculates the constant average growth rate (CAGR)
104
105         Args:
106             start_year (int): Starting year
107             end_year (int): Ending year
108             start_value (int): Initial value
109             end_value (int): Final value
110
111         Returns:
112             cagr: Constant average growth rate (1+ decimal)
113
114             """
115
116         if start_value == 0 or (end_year - start_year) == 0:
117             cagr = 1
118         else:
119             cagr = np.power((end_value / start_value),
120                           (1 / (end_year - start_year)))
121
122         return cagr
123
124
125         def calculate_cagr_forecasts(self, cagr_dictionary,
126                                     base_year_dictionary,
127                                     fuel, year):
128
129
130
131
132
133
134

```

```

115     """ Forecasts base year fuels by a constant average growth rate
116     for a forecast period
117
118     Args:
119         cagr_dictionary (Dict): Dictionary of constant average
120         growth rates per fuel
121         base_year_dictionary ([type]): Dictionary of base year fuel
122         consumption in energy types
123         fuel (list): List of Fuels
124         year (list): List of forecast years
125
126     Returns:
127         [float, array]: 2D Array of demand forecasts per fuel
128     """
129
130     # Initialises the size of the array
131     forecast = np.ones((len(fuel), len(year)))
132
133     # Set the first forecast as the base year
134     for i in range(0, len(fuel), 1):
135         forecast[i, 0] = base_year_dictionary[fuel[i]]
136
137     # Calculates the forecasting
138     for i in range(0, len(fuel), 1):
139         for j in range(1, len(year), 1):
140             forecast[i, j] = forecast[i, j - 1] * cagr_dictionary[
fuel[j]]
141
142     return forecast

```

5.2.5 Optimisation

The module to solve energy systems either locally or remotely using IBM technologies.

```

1 ######
2 # Optimisation contains the Optimisation Class to use CPLEX
3 #####
4
5 # Import python modules
6 import os
7 import cplex as cp
8 import docplex as dp
9 import subprocess as sp
10 from ibm_watson_machine_learning import APIClient
11 import tarfile as tf
12 import time
13
14
15 # Begin class breakdown
16 class Optimisation:
17     """ Prepares and runs optimisation with IBM ILOG CPLEX Optimisation
18     Studio
19     """
20     def __init__(self):
21         """ Initialise the optimisation class
22         """
23
24     def use_bash_shell(self, command):

```

```

24     """ Execute bash commands in python scripts
25
26     Args:
27         command (str): Command to execute
28     """
29     # Execute the demand
30     sp.Popen(['/bin/bash', '-c', command])
31
32     def create_linear_programme_file(self, directory, data_file,
33                                     model_file,
34                                     output_file):
35         """ Creates the model file through executing model system
36         commands
37
38         Args:
39             directory (str): Name of directory to put data into
40             data_file (str): Name of energy system data file
41             model_file (str): Name of energy system model file
42             output_file (str): Name of output linear programme
43             """
44         # Change the working directory
45         os.chdir(directory)
46         # Load the custom anaconda environment
47         # This assumes the conda environment has already been
48         initialised.
49         os.system('conda activate osemosys')
50         # Execute the file structure to create the linear programming
51         file
52         # (glpsol -m GOCPI_Model.txt -d GOCPI_Data.txt --wlp GOCPI.lpx
53         command = 'glpsol -m ' + data_file + ' -d ' + model_file + '--'
54         wlp ' + output_file
55         os.system(command)
56
57     def run_cplex_local(self, model_file):
58         """ This function runs cplex on the local device if the energy
59         system
60             is of a small enough scale
61
62             Args:
63                 model_file (str): Path of model file
64
65             Returns:
66                 [int]: Objective value
67             """
68         # Creates the model structure
69         model = cp.Cplex()
70         # Produces the results stream and log streams
71         output = model.set_results_stream(None)
72         output = model.set_log_stream(None)
73         # Write the energy system model to Cplex
74         model.read(model_file)
75         # Solve the model using the version of Cplex installed on the
76         local
77         # device (IBM ILOG CPLEX Optimisation Studio)
78         model.solve()
79         # Return the value of the objective function
80         objective_value = model.solution.get_objective_value()
81         return objective_value

```

```

75
76     def run_ibm_wml_do(self, apikey, url, deployment_space_name,
77                         cloud_object_storage_credential,
78                         service_instance_id,
79                         deployment_space_exists, data_assets_exist,
80                         data_asset_dictionary, model_name, model_type,
81                         model_runtime_uid, model_tar_file, num_nodes,
82                         deployment_exists, payload_input_data_id,
83                         payload_input_data_file, payload_output_data_id)
84     :
85
86         """ This function enables the user to solve python-based
87         optimisation models. The legacy offering
88         to solve optimisation models on IBM cloud was using
89         the docplex python api to run Cplex on DOcloud.
90         As of September 2020, the DOcloud
91         was discontinued with Decision Optimisation
92         functionalities imported to IBM's Watson Machine
93         Learning Service. The new process requires the
94         energy system model to be written in python. This
95         project saw the implementation of the osemosys
96         modelling methodology in GNU Mathprog written into
97         LP Files. IBM Decision Optimisation in cannot deploy
98         models in LP File formats to get jobs. Therefore,
99         this function is for future work in converting the
100        entire energy system modelling tool to python-based only.
101        This is well-documented the report in the Future Work
102        Section. Note: You must have access to IBM Watson Studio
103        and Cloud Products through the IBM Academic Initiative or
104        Similar.
105
106    Args:
107        apikey (str): API key from user's IBM Cloud Account
108        url ([type]): URL for the server the user is using for the
109        IBM services
110        deployment_space_name (str): Name of the deployment space
111        cloud_object_storage_credential (str): Credential for the
112        cloud object storage asset
113        service_instance_id (str): Service instance id for the
114        service being used (IBM WML)
115        deployment_space_exists (boolean): True/False if the
116        deployment space already exists
117        data_assets_exist (boolean): True/False if the data assets
118        (e.g. input data stored on cloud)
119        data_asset_dictionary (dict): A dictionary of data assets
120        to stored on IBM cloud
121        model_name (str): Name of the model
122        model_type (str): Name of the model
123        model_runtime_uid (str): Runtime ID for the model
124        model_tar_file (tar): Tar file containing the python model
125        num_nodes (int): Number of nodes the model is run off.
126        deployment_exists (boolean): True/False if the deployment
127        already exists
128        payload_input_data_id (str): Name of input data
129        payload_input_data_file (dataframe): Input data file in the
130        form of a dataframe
131        payload_output_data_id (str): Name of output data file
132
133    """

```

```

123     # Creates the Watson Machine learning Credentials
124     api_wml_credentials = {
125         # IBM Cloud User Account Access Code
126         "apikey": apikey,
127         # Url to code repository
128         "url": url
129     }
130     # Initials the client credentials
131     client = APIClient(api_wml_credentials)
132
133     # Create a deployment space on the IBM Cloud Service
134     space_metadata = {
135         # Configures deployment space name
136         client.spaces.ConfigurationMetaNames.NAME:
137             deployment_space_name,
138         # Configures deployment space description
139         client.spaces.ConfigurationMetaNames.DESCRIPTION:
140             deployment_space_name + ' Deployment for energy systems
models',
141         # Configures deployment space storage location
142         client.spaces.ConfigurationMetaNames.STORAGE: {
143             "type": "bmcos_object_storage",
144             "resource_crn": cloud_object_storage_credential
145         },
146         # Configures deployment
147         client.spaces.ConfigurationMetaNames.COMPUTE: {
148             "name": "existing_instance_id",
149             "crn": service_instance_id
150         }
151     }
152     # Bypasses the creation of the deployment space if is already
exists
153     if deployment_space_exists == True:
154         client.spaces.list()
155         # Asks user to input the Space ID of the Input Space
156         space_id = input('Please input the Space ID: ')
157     else:
158         # Stores the newly created space in the repositories spaces
list
159         space = client.spaces.store(meta_props=space_metadata)
160         space_id = client.spaces.get_id(space)
161
162     # Sets the client space
163     client.set.default_space(space_id)
164
165     # Creates input and output data assets if they don't exist
166     if data_assets_exist == False:
167         # Loop through dictionary of data assets to create
168         for key in data_asset_dictionary:
169             client.data_assets.create(key, data_asset_dictionary[
key])
170
171         # Creates software mane and specification for the deployment
172         client.software_specifications.list()
173         software_name = input("Please Input Software Name: ")
174         software_spec_uid = client.software_specifications.
get_uid_by_name(
175             software_name)

```

```

176
177     # Creates the model deployment
178     model_metadata = {
179         client.repository.ModelMetaNames.NAME: model_name,
180         client.repository.ModelMetaNames.DESCRIPTION: model_name +
181         'Model',
182         client.repository.ModelMetaNames.TYPE: model_type,
183         client.repository.ModelMetaNames.RUNTIME_UID:
184         model_runtime_uid,
185         client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:
186         software_spec_uid
187     }
188     # Creates the energy model details
189     model_details = client.repository.store_model(
190         model=model_tar_file, meta_props=model_metadata)
191     # Creates model uid
192     model_uid = client.repository.get_model_uid(model_details)
193
194     # Creates a deployment
195     meta_props = {
196         client.deployments.ConfigurationMetaNames.NAME:
197             "Deployment " + str(num_nodes),
198         client.deployments.ConfigurationMetaNames.DESCRIPTION:
199             "Deployment " + str(num_nodes),
200         # client.deployments.ConfigurationMetaNames.HARDWARE_SPEC:
201         client.deployments.ConfigurationMetaNames.BATCH: {},
202         client.deployments.ConfigurationMetaNames.COMPUTE: {
203             'name': 'S',
204             'nodes': num_nodes
205         }
206     }
207
208     # Tests if deployment already exists
209     if deployment_exists == True:
210         client.deployments.list()
211         deployment_uid = input('Please input the Deployment UID: ')
212     else:
213         deployment_details = client.deployments.create(
214             model_uid, meta_props=meta_props)
215         deployment_uid = client.deployments.get_uid(
216         deployment_details)
217
218     # Creates a payload for the solver to solve (Note: Amend based
219     # on the model you are solving)
220     payload = [
221         client.deployments.DecisionOptimizationMetaNames.INPUT_DATA
222         : [
223             {
224                 "id": payload_input_data_id,
225                 "values": [
226                     payload_input_data_file
227                 ],
228                 client.deployments.DecisionOptimizationMetaNames.
229                 OUTPUT_DATA: [
230                     {
231                         "id": payload_output_data_id
232                     }
233                 ]
234             }
235         ]
236     }

```

```
228     # Creates a new job using the deployment and payload
229     job_details = client.deployments.create_job(deployment_uid,
230     payload)
231     job_uid = client.deployments.get_job_uid(job_details)
232
233     # Print the status of the job until competition
234     while job_details['entity']['decision_optimization']['status'][
235         'state'] not in ['completed', 'failed', 'canceled']:
236         print(job_details['entity']['decision_optimization'][
237             'status'][
238                 'state'] + '...')
239         job_details = client.deployments.get_job_details(job_uid)
240         time.sleep(5)
241         print(job_details['entity']['decision_optimization'][
242             'status'][
243                 'state']))
244
245     # Reset tarfile function (Source: IBM Watson Machine Learning)
246     def reset(self, tarinfo):
247         """ Resets the tarfile information when creating tar files
248             This is to input into the filter when using tar.add()
249
250             Args:
251                 tarinfo (Object): Tar Object containing an ID of 0 and the
252                 root as the name
253
254             Returns:
255                 tarinfo (Object): Tar Object containing an ID of 0 and the
256                 root as the name
257
258                 """
259                 tarinfo.uid = tarinfo.gid = 0
260                 tarinfo.uname = tarinfo.gname = "root"
261                 return tarinfo
```

6 Development Scripting

The scripts within this section were used to design the GOCPI modules needed for the project.

6.1 GOCPI Data Cases

This script helped set the structure to build the model and data files for energy systems.

```

1 # GOCPI_Data_Cases is a methodology to import scenario data
2 # across multiple files. These are the
3 # sets and parameters for the Energy System Optimisation Model.
4 # A python script was chosen over other storage methods (e.g. excel)
5 # as values can be stored in matrices and many values are configured
6 # differently
7
8 # Import useful python packages
9 # Git repository
10 # https://github.com/CMCD1996/GOCPI.git
11 # Make more changes from the pull request
12 import numpy as np
13 import pandas as pd
14 import matplotlib.pyplot as plt
15 import scipy as sc
16 import sklearn as skl
17 import csv as csv
18 import openpyxl
19 import pathlib
20 import os
21 from pathlib import Path
22 from openpyxl import load_workbook
23 import GOCPI as GF
24 import cplex as cp
25 import docplex as dp
26
27 # Creates sets for the demo model
28 YEAR = [
29     '1990', '1991', '1992', '1993', '1994', '1995', '1996', '1997', '1998',
30     '1999', '2000', '2001', '2002', '2003', '2004', '2005', '2006', '2007',
31     '2008', '2009', '2010'
32 ]
33 REGION = ['NEWZEALAND', 'AUSTRALIA']
34 EMISSION = ['CO2', 'NOX', 'CO', 'METHANE']
35 TECHNOLOGY = [
36     'E01', 'E21', 'E31', 'E51', 'E70', 'IMPDSDL1', 'IMP GSL1', 'IMPHC01',
37     'IMPOIL1', 'IMPURN1', 'RHE', 'RHO', 'RL1', 'SRE', 'TXD', 'TXE', 'TXG',
38     'RIV', 'RHu', 'RLu', 'TXu'
39 ]
40 FUEL = [
41     'CSV', 'DSL', 'ELC', 'GSL', 'HCO', 'HYD', 'LTH', 'OIL', 'URN', 'RH',
42     'RL',
43     'TX'
44 ]
```

```

43 TIMESLICE = [
44     'INTERMEDIATE_DAY', 'INTERMEDIATE_NIGHT', 'SUMMER_DAY', ,
45     SUMMER_NIGHT',
46     'WINTER_DAY', 'WINTER_NIGHT'
47 ]
48 MODE_OF_OPERATION = ['1', '2']
49 STORAGE = ['DAM']
50 DAYTYPE = ['1', '2', '3']
51 SEASON = [
52     '1', '2', '3', '4'
53 ] # Must be denoted in numbers to match constraints in model (1:
      Summer, 2: Autumn, 3: Winter, 4): Spring)
54 DAILYTIMEBRACKET = ['1', '2', '3']
55
56 # Sets
57 sets = [
58     YEAR, REGION, EMISSION, TECHNOLOGY, FUEL, TIMESLICE,
59     MODE_OF_OPERATION,
60     STORAGE, DAYTYPE, SEASON, DAILYTIMEBRACKET
61 ]
62
63 # Create the energy system with sets and initialised parameters. The
   parameter have the necessary parameters
64 Demo = GF.Energy_Systems(YEAR, REGION, EMISSION, TECHNOLOGY, FUEL,
65                           TIMESLICE,
66                           MODE_OF_OPERATION, STORAGE, DAYTYPE, SEASON,
67                           DAILYTIMEBRACKET)
68
69 # This user must now initialise the parameters as they choose to
   configure the energy system for the optimisation model.
70 # This is incredibly important. The user must understand the
   configuration of the energy system to do this! Consult the
71 # User manual to build this optimisation.
72
73 # End of user defined inputs in this script
74
75 # Sets the txtfile saved locations
76 data_txt = 'GOCPI_OseMOSYS_Data.txt'
77 model_source_file = 'GOCPI_OseMOSYS_Structure.xlsx'
78 root = '/Users/connor/Google Drive/Documents/University/Courses/2020/
    ENGSCI 700A&B/GOCPI/data/Inputs/GOCPI_OseMOSYS'
79 data_roots = Path(root)
80 data_location_1 = os.path.join(data_roots, data_txt)
81
82 # Dictionary of default parameters for creating a model file
83 default_parameters = {
84     'YearSplit': 1,
85     'DiscountRate': 1,
86     'DaySplit': 1,
87     'Conversionls': 1,
88     'Conversionld': 1,
89     'Conversionlh': 1,
90     'DaysInDayType': 1,
91     'TradeRoute': 1,
92     'DepreciationMethod': 1,
93     'SpecifiedAnnualDemand': 1,
94     'SpecifiedDemandProfile': 1,
95     'AccumulatedAnnualDemand': 1,

```

```

93     'CapacityToActivityUnit': 1,
94     'CapacityFactor': 1,
95     'AvailabilityFactor': 1,
96     'OperationalLife': 1,
97     'ResidualCapacity': 1,
98     'InputActivityRatio': 1,
99     'OutputActivityRatio': 1,
100    'CapitalCost': 1,
101    'VariableCost': 1,
102    'FixedCost': 1,
103    'TechnologyToStorage': 1,
104    'TechnologyFromStorage': 1,
105    'StorageLevelStart': 1,
106    'StorageMaxChargeRate': 1,
107    'StorageMaxDischargeRate': 1,
108    'MinStorageCharge': 1,
109    'OperationalLifeStorage': 1,
110    'CapitalCostStorage': 1,
111    'ResidualStorageCapacity': 1,
112    'CapacityOfOneTechnologyUnit': 1,
113    'TotalAnnualMaxCapacity': 1,
114    'TotalAnnualMinCapacity': 1,
115    'TotalAnnualMaxCapacityInvestment': 1,
116    'TotalAnnualMinCapacityInvestment': 1,
117    'TotalTechnologyAnnualActivityLowerLimit': 1,
118    'TotalTechnologyAnnualActivityUpperLimit': 1,
119    'TotalTechnologyModelPeriodActivityUpperLimit': 1,
120    'TotalTechnologyModelPeriodActivityLowerLimit': 1,
121    'ReserveMarginTagTechnology': 1,
122    'ReserveMarginTagFuel': 1,
123    'ReserveMargin': 1,
124    'RETagTechnology': 1,
125    'RETagFuel': 1,
126    'REMinProductionTarget': 1,
127    'EmissionActivityRatio': 1,
128    'EmissionsPenalty': 1,
129    'AnnualExogenousEmission': 1,
130    'AnnualEmissionLimit': 1,
131    'ModelPeriodExogenousEmission': 1,
132    'ModelPeriodEmissionLimit': 1
133 }
134
135 # Create the Data File
136 Demo.create_data_file(data_location_1, default_parameters)
137
138 # Cereate the Model File
139 Demo.create_model_file(root, model_source_file)
140
141 # Convert created model and data files into a Linear Problem file (lp)
142 # Test the formatting

```

6.2 GOCPI Energy Balances

This script helped extract energy balances from the International Energy Agency's World Energy Balances.

```

1 # GOCPI_EB prepares the energy balance across time for certain
2   geographies
3 # This script was adapted into the GOCPI Module
4
5 # Import useful python packages
6 # Git repository
7 # https://github.com/CMCD1996/GOCPI.git
8 # Make more changes from the pull request
9 import numpy as np
10 import pandas as pd
11 import matplotlib.pyplot as plt
12 import scipy as sc
13 import sklearn as skl
14 import csv as csv
15 import openpyxl as pyxl
16 import pathlib
17 import os
18 import pydrive
19
20 # Very Important Step: Sets directory root for file operations.
21 source_root = pathlib.Path(
22     '/Users/connor/Google Drive/Documents/University/Courses/2020/
23       ENGSCI 700A&B/GOCPI/data/Energy Balances'
24 )
25
26 # Load in the EnergyBalance.csv file found from the University of
27   Auckland SourceOECD Database.
28 # This csv contains the energy balances around the world
29
30 # Important Step: Sets the Energy Balances Folder ID in my personal
31   google drive
32 folderID = '1PCUMeT8c9dJE1ES8JDg62w2rKMAS0xSW' # Energy Balance
33
34 # Loads in appropriate pydrive functions for access
35 from pydrive.auth import GoogleAuth
36 from pydrive.drive import GoogleDrive
37
38 # # Creates the authorisation to access the google drive
39 # gauth = GoogleAuth()
40 # gauth.LocalWebserverAuth() # Gains authorisation using the
41   clients_secrets.json file in the src directory
42 # # Creates a google drive object to handle files
43 # drive = GoogleDrive(gauth)
44
45 # # Tests the access to the google drive and finds all
46   IEAEnergyBalances.csv File IDs in the EnergyBalances Directory
47 # file_list = []
48 # title_list = []
49 # files = drive.ListFile({'q': "'1MD5ewAKAy2McqyCfjivwavj278giRvmR' in
50   parents and trashed=false"}).GetList()
51 # for filex in files:
52 #     print(filex['id'])
53 #     print(filex['title'])
54 #     file_list.append(filex['id']) # IEAEnergyBalance.csvs File IDs
55 #     title_list.append(filex['title']) # IEAEnergyBalance.csvs Titles
56
57 # Gets the links to the two files wanted

```

```

51 # Links for IEA Energy Balance for 2018 (A-K) and IEA Energy Balance
52 # for 2017 (L-Z)
53 IEAWEBAK = source_root / 'IEAWorldEnergyBalances2017A-K.csv'
54 IEAWEBLZ = source_root / 'IEAWorldEnergyBalances2017L-Z.csv'
55
56 # Creates dataframes from IEA World Energy Statistics and Balances CSVs
57 # from Stats.OECD.org in the OECDLibrary
58 # Note the data is from #https://stats.oecd.org/ and #https://www.oecd-
59 # -library-org.ezproxy.auckland.ac.nz/
60 column_headers = [
61     'ID', 'Unit', 'Geo_Code', 'Geo_Description', 'Prod_Code',
62     'Prod_Description', 'Flow_Code', 'Flow_Description', 'Year', 'Value
63     (TJ)',
64 ]
65 f1 = open(IEAWEBAK, 'r')
66 df_A = pd.read_csv(f1, header=None)
67 df_A.columns = column_headers
68 df_A.info(verbose=True)
69 f2 = open(IEAWEBLZ, 'r')
70 df_B = pd.read_csv(f2, header=None)
71 df_B.columns = column_headers
72 df_B.info(verbose=True)
73 frames = [df_A, df_B]
74 df = pd.concat(frames)
75 df.info(verbose=True)
76
77 # Closes the files
78 f1.close()
79 f2.close()
80
81 # Find the unique items in each list of the energy balance sheets
82 uv_prod = df.Prod_Description.unique()
83 uv_geo = df.Geo_Description.unique()
84 uv_flow = df.Flow_Description.unique()
85
86 # Establishes the rows and columns for the EnergyBalance.xlsm
87 # spreadsheet
88 # Note: Most likely in the calculation, Other will be a sink so Total
89 # Energy Supply - Conversion Losses = Total Energy Consumed
90 # Rows (Energy uses)
91 Primary = ['Domestic Supply', 'Imports', 'Exports', 'Total Primary
92 Supply']
93 Conversion = [
94     'Energy Sector Consumption', 'Electricity Plants', 'Heat Plants',
95     'Petroluem Refineries', 'Total Conversion'
96 ]
97 Consumption = [
98     'Residential', 'Commercial', 'Industry', 'Agriculture', 'Transport',
99     ,
100     'Other', 'Non Energy', 'Bunkers', 'Total Final Consumption'
101 ]
102
103 # Primary (To complete)
104 DomesticSupply = ['Production']
105 Imports = ['Imports']
106 Exports = ['Exports']
107 TotalPrimarySupply = ['Total primary energy supply']

```

```

101 # Conversion (To complete)
102 Energy_Sector_Consumption = [,,]
103 Electricity_Plants = [,,]
104 Heat_Plants = [,,]
105 Petroleum_Refineries = [,,]
106 Total_Conversion = [,,]
107
108 # Consumption (To complete)
109 Residential = [,,]
110 Commercial = [,,]
111 Industry = [,,]
112 Agriculture = [,,]
113 Transport = [,,]
114 Other = ['Stock changes', 'Transfers', 'Statistical differences']
115 Non_Energy = [,,]
116 Bunkers = ['International marine bunkers', 'International aviation
    bunkers']
117 Total_Final_Consumption = [,,]
118
119 # Energy Flows
120 Energy_Flows = [
121     'Production', 'Imports', 'Exports', 'International marine bunkers',
122     'International aviation bunkers', 'Stock changes',
123     'Total primary energy supply', 'Transfers', 'Statistical
    differences',
124     'Transformation processes', 'Main activity producer electricity
    plants',
125     'Autoproducer electricity plants', 'Main activity producer CHP
    plants',
126     'Autoproducer CHP plants', 'Main activity producer heat plants',
127     'Autoproducer heat plants', 'Heat pumps', 'Electric boilers',
128     'Chemical heat for electricity production', 'Blast furnaces', 'Gas
    works',
129     'Coke ovens', 'Patent fuel plants', 'BKB/peat briquette plants',
130     'Oil refineries', 'Petrochemical plants', 'Coal liquefaction plants
    ',
131     'Gas-to-liquids (GTL) plants', 'For blended natural gas',
132     'Charcoal production plants', 'Non-specified (transformation)',
133     'Energy industry own use', 'Coal mines', 'Oil and gas extraction',
134     'Gasification plants for biogases',
135     'Liquefaction (LNG) / regasification plants',
136     '"Own use in electricity, CHP and heat plants"', 'Pumped storage
    plants',
137     'Nuclear industry', 'Non-specified (energy)', 'Losses',
138     'Total final consumption', 'Industry', 'Mining and quarrying',
139     'Construction', 'Manufacturing', 'Iron and steel',
140     'Chemical and petrochemical', 'Non-ferrous metals',
141     'Non-metallic minerals', 'Transport equipment', 'Machinery',
142     'Food and tobacco', '"Paper, pulp and printing"', 'Wood and wood
    products',
143     'Textile and leather', 'Industry not elsewhere specified', '
    Transport',
144     'World aviation bunkers', 'Domestic aviation', 'Road', 'Rail',
145     'Pipeline transport', 'World marine bunkers', 'Domestic navigation'
    ,
146     'Non-specified (transport)', 'Residential',
147     'Commercial and public services', 'Agriculture/forestry', 'Fishing'
    ,

```

```

148     'Final consumption not elsewhere specified', 'Non-energy use',
149     'Non-energy use industry/transformation/energy',
150     'Memo: Non-energy use in industry', 'Memo: Non-energy use in
construction',
151     'Memo: Non-energy use in mining and quarrying',
152     'Memo: Non-energy use in iron and steel',
153     'Memo: Non-energy use in chemical/petrochemical',
154     'Memo: Non-energy use in non-ferrous metals',
155     'Memo: Non-energy use in non-metallic minerals',
156     'Memo: Non-energy use in transport equipment',
157     'Memo: Non-energy use in machinery',
158     'Memo: Non-energy use in food/beverages/tobacco',
159     'Memo: Non-energy use in paper/pulp and printing',
160     'Memo: Non-energy use in wood and wood products',
161     'Memo: Non-energy use in textiles and leather',
162     'Memo: Non-energy use in industry not elsewhere specified',
163     'Non-energy use in transport', 'Non-energy use in other',
164     'Electricity output (GWh)',
165     'Electricity output (GWh)-main activity producer electricity plants
',
166     'Electricity output (GWh)-autoproducer electricity plants',
167     'Electricity output (GWh)-main activity producer CHP plants',
168     'Electricity output (GWh)-autoproducer CHP plants',
169     'Heat output', 'Heat output-main activity producer CHP plants',
170     'Heat output-autoproducer CHP plants',
171     'Heat output-main activity producer heat plants',
172     'Heat output-autoproducer heat plants'
173 ]
174
175 # Columns (Energy Types)
176 Energy = [
177     'Solid Fuels', 'Natural Gas', 'Crude Oil', 'Diesel Oil', 'Kerosene',
178     , 'LPG',
179     'Motor Spirit', 'Naphtha', 'Heavy Fuel Oil', 'Other Petroleum
Products',
180     'Nuclear Energy', 'Biomass', 'Hydro power', 'Wind energy', 'Solar
Energy',
181     'Industrial Wastes', 'Derived Heat', 'Electricity', 'Total'
182 ]
183 Solid_Fuels = [',']
184 Natural_Gas = [',']
185 Crude_Oil = [',']
186 Diesel_Oil = [',']
187 Kerosene = [',']
188 LPG = [',']
189 Motor_Spirit = [',']
190 Naphtha = [',']
191 Heavy_Fuel_Oil = [',']
192 Other_Petroleum_Products = [',']
193 Nuclear_Energy = [',']
194 Biomass = [',']
195 Hydro_power = [',']
196 Wind_Energy = [',']
197 Solar_Energy = [',']
198 Industrial_Wastes = [',']
199 Derived_Heat = [',']
200 Electricity = [',']

```

```

201 Total = [ , ]
202
203 Energy_Types = [
204     'Hard coal (if no detail)', 'Brown coal (if no detail)', ,
205     'Anthracite',
206     'Coking coal', 'Other bituminous coal', 'Sub-bituminous coal', ,
207     'Lignite',
208     'Patent fuel', 'Coke oven coke', 'Gas coke', 'Coal tar',
209     'BKB', 'Gas works gas', 'Coke oven gas', 'Blast furnace gas',
210     'Other recovered gases', 'Peat', 'Peat products',
211     'Oil shale and oil sands', 'Natural gas',
212     'Crude/NGL/feedstocks (if no detail)', 'Crude oil', 'Natural gas
liquids',
213     'Refinery feedstocks', 'Additives/blending components',
214     'Other hydrocarbons', 'Refinery gas', 'Ethane',
215     'Liquefied petroleum gases (LPG)', 'Motor gasoline excl. biofuels',
216     'Aviation gasoline', 'Gasoline type jet fuel',
217     'Kerosene type jet fuel excl. biofuels', 'Other kerosene',
218     'Gas/diesel oil excl. biofuels', 'Fuel oil', 'Naphtha',
219     'White spirit & SBP', 'Lubricants', 'Bitumen', 'Paraffin waxes',
220     'Petroleum coke', 'Other oil products', 'Industrial waste',
221     'Municipal waste (renewable)', 'Municipal waste (non-renewable)',
222     'Primary solid biofuels', 'Biogases'
223     'Biogasoline'
224     'Biodiesels', 'Bio jet kerosene', 'Other liquid biofuels',
225     'Non-specified primary biofuels and waste', 'Charcoal',
226     'Elec/heat output from non-specified manufactured gases',
227     'Heat output from non-specified combustible fuels', 'Nuclear', ,
228     'Hydro',
229     'Geothermal', 'Solar photovoltaics', 'Solar thermal',
230     '"Tide, wave and ocean"', 'Wind', 'Other sources', 'Electricity', ,
231     'Heat',
232     'Total', 'Memo: Renewables'
233 ]
234
235 # Creates a pivot table to display the data in the way similar to the
236 # Energy Balance Sheet (cols = Energy Product, rows = Energy Flows)
237 EBPT = pd.pivot_table(df,
238                         index=['Geo_Description', 'Flow_Description'],
239                         values=['Value(TJ)'],
240                         columns=['Prod_Description'],
241                         aggfunc=[np.sum],
242                         fill_value=0)
243 # Filters to the geography the user has selected
244 Selected_Geo = uv_geo[0] # Update once turned into a custom function
245 Input_String = 'Geo_Description == ["' + Selected_Geo + '"]'
246 EBPTG = EBPT.query(Input_String)
247
248 # Write the filtered pivot table to an excel file
249 writer = pd.ExcelWriter(source_root / "Geo EB.xlsx")
250 EBPTG.to_excel(writer, Selected_Geo)
251 writer.save()
252
253 # Keep assigning this variables
254 # Energy_Flows = [
255 #     , , ,
256 #     , , ,
257

```

```

252 #   'Autoproducer electricity plants', 'Main activity producer CHP
253 #   plants',
254 #   'Autoproducer CHP plants', 'Main activity producer heat plants',
255 #   'Autoproducer heat plants', 'Heat pumps', 'Electric boilers',
256 #   'Chemical heat for electricity production', 'Blast furnaces', 'Gas
257 #   works',
258 #   'Coke ovens', 'Patent fuel plants', 'BKB/peat briquette plants',
259 #   'Oil refineries', 'Petrochemical plants', 'Coal liquefaction plants',
260 #   'Gas-to-liquids (GTL) plants', 'For blended natural gas',
261 #   'Charcoal production plants', 'Non-specified (transformation)',
262 #   'Energy industry own use', 'Coal mines', 'Oil and gas extraction',
263 #   'Gasification plants for biogases',
264 #   'Liquefaction (LNG) / regasification plants',
265 #   '"Own use in electricity, CHP and heat plants"', 'Pumped storage
266 #   plants',
267 #   'Nuclear industry', 'Non-specified (energy)', 'Losses',
268 #   'Total final consumption', 'Industry', 'Mining and quarrying',
269 #   'Construction', 'Manufacturing', 'Iron and steel',
270 #   'Chemical and petrochemical', 'Non-ferrous metals', 'Non-metallic
271 #   minerals',
272 #   'Transport equipment', 'Machinery', 'Food and tobacco',
273 #   '"Paper, pulp and printing"', 'Wood and wood products',
274 #   'Textile and leather', 'Industry not elsewhere specified', 'Transport',
275 #   'World aviation bunkers', 'Domestic aviation', 'Road', 'Rail',
276 #   'Pipeline transport', 'World marine bunkers', 'Domestic navigation',
277 #   'Non-specified (transport)', 'Residential',
278 #   'Commercial and public services', 'Agriculture/forestry', 'Fishing',
279 #   'Final consumption not elsewhere specified', 'Non-energy use',
280 #   'Non-energy use industry/transformation/energy',
281 #   'Memo: Non-energy use in industry', 'Memo: Non-energy use in
282 #   construction',
283 #   'Memo: Non-energy use in mining and quarrying',
284 #   'Memo: Non-energy use in iron and steel',
285 #   'Memo: Non-energy use in chemical/petrochemical',
286 #   'Memo: Non-energy use in non-ferrous metals',
287 #   'Memo: Non-energy use in non-metallic minerals',
288 #   'Memo: Non-energy use in transport equipment',
289 #   'Memo: Non-energy use in machinery',
290 #   'Memo: Non-energy use in food/beverages/tobacco',
291 #   'Memo: Non-energy use in paper/pulp and printing',
292 #   'Memo: Non-energy use in wood and wood products',
293 #   'Memo: Non-energy use in textiles and leather',
294 #   'Memo: Non-energy use in industry not elsewhere specified',
295 #   'Non-energy use in transport', 'Non-energy use in other',
296 #   'Electricity output (GWh)',
297 #   'Electricity output (GWh)-main activity producer electricity plants
298 #   ',
299 #   'Electricity output (GWh)-autoproducer electricity plants',
300 #   'Electricity output (GWh)-main activity producer CHP plants',
301 #   'Electricity output (GWh)-autoproducer CHP plants' 'Heat output',
302 #   'Heat output-main activity producer CHP plants',
303 #   'Heat output-autoproducer CHP plants',
304 #   'Heat output-main activity producer heat plants',
305 #   'Heat output-autoproducer heat plants']

# Energy_Types = ['Hard coal (if no detail)', 'Brown coal (if no detail
# )', 'Anthracite'],

```

```

302 #   'Coking coal', 'Other bituminous coal', 'Sub-bituminous coal', '
303 #   'Lignite',
304 #   'Patent fuel', 'Coke oven coke', 'Gas coke', 'Coal tar' 'BKB',
305 #   'Gas works gas', 'Coke oven gas', 'Blast furnace gas',
306 #   'Other recovered gases', 'Peat', 'Peat products', 'Oil shale and oil
307 #   sands',
308 #   'Natural gas', 'Crude/NGL/feedstocks (if no detail)', 'Crude oil',
309 #   'Natural gas liquids', 'Refinery feedstocks',
310 #   'Additives/blending components', 'Other hydrocarbons', 'Refinery gas
311 #   ',
312 #   'Ethane', 'Liquefied petroleum gases (LPG)',
313 #   'Motor gasoline excl. biofuels', 'Aviation gasoline',
314 #   'Gasoline type jet fuel', 'Kerosene type jet fuel excl. biofuels',
315 #   'Other kerosene', 'Gas/diesel oil excl. biofuels', 'Fuel oil', '
316 #   'Naphtha',
317 #   'White spirit & SBP', 'Lubricants', 'Bitumen', 'Paraffin waxes',
318 #   'Petroleum coke', 'Other oil products', 'Industrial waste',
319 #   'Municipal waste (renewable)', 'Municipal waste (non-renewable)',
320 #   'Primary solid biofuels', 'Biogases' 'Biogasoline' 'Biodiesels',
321 #   'Bio jet kerosene', 'Other liquid biofuels',
322 #   'Non-specified primary biofuels and waste', 'Charcoal',
323 #   'Elec/heat output from non-specified manufactured gases',
324 #   'Heat output from non-specified combustible fuels', 'Nuclear', '
325 #   'Hydro',
326 #   'Geothermal', 'Solar photovoltaics', 'Solar thermal',
327 #   '"Tide, wave and ocean"', 'Wind', 'Other sources', 'Electricity', '
328 #   'Heat',
329 #   'Total', 'Memo: Renewables']

```

6.3 GOCPI Geographies

This script helped create the geographical subsets for modelling energy regions.

```

1 # GOCPI_Geographies Structures the geographies into Countries, Cities
2 # and Continents
3
4 # Import useful python packages
5 # Git repository
6 # https://github.com/CMCD1996/GOCPI.git
7 # Make more changes from the pull request
8 import numpy as np
9 import pandas as pd
10 import matplotlib.pyplot as plt
11 import scipy as sc
12 import sklearn as skl
13 import csv as csv
14 import pathlib
15 import os
16
17 # Very Important Step: Sets directory root for file operations.
18 source_root = pathlib.Path('/Users/connor/Google Drive/Documents/
19 # University/Courses/2020/ENGSCI 700A&B/GOCPI/data/Geographies')
20
21 # Finds the relevant files needed to create csvs with the relationships
22 # between cities, countries and continents
23 # Finds a file within a function
24 def Find_File(target_root,target_file):

```

```

22     for root, dirs, files in os.walk(target_root):
23         for name in files:
24             if name == target_file:
25                 f = os.path.abspath(os.path.join(root, name))
26         return f
27
28 # Find the necessary files for the geography conversions
29 f1 = Find_File(source_root, "Country and Continent.txt")
30 f2 = Find_File(source_root, "countries.csv")
31 f3 = Find_File(source_root, "Cities.csv")
32 f4 = Find_File(source_root, "geography_set.csv")
33
34 # Creates python list for geographies, starting with countries
35 # Creates an empty list
36 countries = []
37
38 # Creates a geography set
39 geography_set = [['AFRICA'],
40                   ['ASIA'],
41                   ['EUROPE'],
42                   ['NORTH AMERICA'],
43                   ['OCEANIA'],
44                   ['SOUTH AMERICA']]
45 continents = ['AFRICA', 'ASIA', 'EUROPE', 'NORTH AMERICA', 'OCEANIA', 'SOUTH
46   AMERICA']
47
48 # Sets up a for loop to append countries to the continents in the
49   geography sets
50 file = open(f1, 'r')
51 for line in file:
52     string = line.split('\n')
53     string = string[0].split(',')
54     countries.append(string[1].upper())
55     for i in range(0,6,1):
56         if string[0].upper() == geography_set[i][0]:
57             geography_set[i].append(string[1].upper())
58
59 # This code block is to inform count
60 with open(f2, 'w') as file:
61     writer = csv.writer(file, delimiter = ',')
62     writer.writerow(countries)
63 file.close()
64
65 # Creates array of world cities
66 data = pd.read_csv(f3)
67 cities_df = pd.concat([data['city'], data['country'], data['population']],
68   axis = 1)
69 cities_df['continent'] = ""
70 cities_df.dropna(inplace = True)
71
72 # Capitalises country and city names
73 cities_df['country'] = cities_df['country'].str.upper()
74 cities_df['city'] = cities_df['city'].str.upper()
75
76 # Places the continent required in the row
77 for index, row in cities_df.iterrows():
78     for i in range(0,6,1):
79         for j in range(0, len(geography_set[i]), 1):

```

```

77         if geography_set[i][j] == row['country']:
78             cities_df.at[index, 'continent'] = geography_set[i][0]
79
80 # Saves dataframe as new CSV
81 cities_df.to_csv(f4, index=False)

```

6.4 GOCPI Inputs

This script helped update values in the Excel spreadsheet when developing a standardised modelling process for the TIMES Methodology.

```

1 # GOCPI_Inputs is a file processing script. This script prepare the
2     spreadsheets for VEDA processing to
3 be feed into the GAMS Optimisation
4 # This script was adpated into the GOCPI module
5
6 # Import useful python packages
7 # Git repository
8 # https://github.com/CMCD1996/GOCPI.git
9 # Make more changes from the pull request
10 import numpy as np
11 import pandas as pd
12 import matplotlib.pyplot as plt
13 import scipy as sc
14 import sklearn as skl
15 import csv as csv
16 import openpyxl
17 import pathlib
18 import os
19 from pathlib import Path
20 from openpyxl import load_workbook
21
22 # Very Important Step: Set directory root for file operations.
23 source_root = Path(
24     '/Users/connor/Google Drive/Documents/University/Courses/2020/
25 ENGSCI 700A&B/GOCPI/data/Inputs'
26 )
27
28 # Finds a file within a function
29 def Find_File(target_root, target_file):
30     for root, dirs, files in os.walk(target_root):
31         for name in files:
32             if name == target_file:
33                 f = os.path.abspath(os.path.join(root, name))
34     return f
35
36
37 # Defines custom functions necessary for excel script processing
38 # Set_Values updates single cell inputs for the VEDA spreadsheet
39 def Set_Values(source_root, source_file, source_sheet, source_range,
40                 updated_value, destination_file):
41     from openpyxl import load_workbook
42     from pathlib import Path
43     # Finds the source file from the assign root directory
44     f = Find_File(source_root, source_file)

```

```

45     # Performs the workbook manipulation and updates values
46     workbook = load_workbook(filename=f)
47     sheet = workbook[source_sheet]
48     defined_range = workbook.defined_names[source_range]
49     split_string = defined_range.attr_text.split('$')
50     address = split_string[1] + split_string[2]
51     sheet[str(address)].value = updated_value
52     # Finds the destination file
53     f = Find_File(source_root, destination_file)
54     # Saves the updated file
55     workbook.save(filename=f)
56
57
58 # Initialises all variables in the System Settings. These are created
59 # in arrays and interated through via for loops.
60
61 # Book Region_Maps (Number of Regions Base Sheet Mechanics relate to,
62 # This will be expanded upon depending on the sets of geographies to be
63 # included.
64 # We will begin with two regions (Based off TIMES Demo Model 12)s
65 REG1 = 'REG1' # Ideally two selected Regions (New Zealand)
66 REG2 = 'REG2' # (Australia)
67
68 # Timeslices
69 SZN1 = "S" # Summer
70 SZN2 = "W" # Winter
71 DN1 = "D" # Day
72 DN2 = "N" # Night
73
74 # Time Periods
75 StartYear = 2030
76
77 # ActivePDef
78 # This variable defines how split up the forecast period into smaller
# time intervals
79 # Pdef-1 is a two period definition (1 Year then 2 years for a total of
# 3 years)
80 # Pdef-5 is a 5 period definition of 1,2,5,5,5 year periods
# respectively.
81 # Pdef-11 is an 11 period definition of 1,2,5,5,5,5,5,5,5,5,5 year
# periods respectively.
82 # Pdef-1,5 and 11 are the only available options at the moment.
83 APDEF = "Pdef-11"
84
85
86 # Import Settings have been left unchanged in the SysSettings Sheet. See
# the import
87 # settings for a proper definition
88
89 # Interpolation and Extrapolation Defaults are unchanged as well. See
# details in the
90 # System settings spreadsheet if you want to make changes.
91
92 # Constants for the modelling process in the modelling sheet
93 # (TFM_INS)
94 GDY = StartYear # Discount Year
95 Discount = 0.05 # Discount Rate (This discount rate will change
# depending on the region in question
96 # Figure out how to vary dicount rates depending on financial inputs)

```

```

94
95 # Fraction of year for season and day/night level (Should change
# depending on the geography)
96 # Determine how to make these changes after you get a baseline model
# running
97 REG_Num_Sum_Days = 175
98 REG_Num_Days = 365
99 REG_Num_Win_Days = (REG_Num_Days - REG_Num_Sum_Days)
100 Frac_REG_Num_Sum_Days = REG_Num_Sum_Days / REG_Num_Days
101 Frac_REG_Num_Win_Days = REG_Num_Win_Days / REG_Num_Days
102
103 Sum_Hours_Per_Day = 12.5
104 Win_Hours_Per_Day = 11.5
105 Hours_Per_Day = 24
106
107 Frac_Sum_Hours_Per_Day = Sum_Hours_Per_Day / (Hours_Per_Day)
108 Frac_Win_Hours_Per_Day = Sum_Hours_Per_Day / (Hours_Per_Day)
109 Frac_Sum_Hours_Per_Night = (1 - Frac_Sum_Hours_Per_Day)
110 Frac_Win_Hours_Per_Night = (1 - Frac_Win_Hours_Per_Day)
111
112 SD_YRFR = Frac_REG_Num_Sum_Days * Frac_Sum_Hours_Per_Day
113 SN_YRFR = Frac_REG_Num_Sum_Days * Frac_Sum_Hours_Per_Night
114 WD_YRFR = Frac_REG_Num_Win_Days * Frac_Win_Hours_Per_Day
115 WN_YRFR = Frac_REG_Num_Win_Days * Frac_Win_Hours_Per_Night
116
117 # Currency for investment decisions underpinning the model
118 CUR = "MEuro05"
119
120 # Default Units (Review and come back to this commodity part of the
# model)
121 # Explicitly Commodity Groups are not required in the modelling process
# at this stage.
122
123 # Creates a function to update the spreadsheet relative to those feed
# in to the function
124 # Inserts the various cells in python as required
125 # Imports the various functions needed for the file
126
127 source_file = "SysSettings.xlsx"
128 source_sheet = "TimePeriods"
129 source_range = "StartYear"
130 updated_value = StartYear
131 destination_file = "SysSettings.xls"
132
133 # Update the StartYear
134 Set_Values(source_root, source_file, source_sheet, source_range,
# updated_value,
# destination_file)
135

```

6.5 GOCPI Model Import

This script helped import OseMOSYS models.

```

1 # GOCPI_Model_Import is a file processing script. This script prepares
# the text files from an Excel spreadsheet
2 # for the user defined energy systems model.
3 #
##########

```

```
4 # This script was adapted into the GOCPI module.
5 #
6 ######
7 # Import useful python packages
8 # Git repository
9 # https://github.com/CMCD1996/GOCPI.git
10 # Make more changes from the pull request
11 import numpy as np
12 import pandas as pd
13 import matplotlib.pyplot as plt
14 import scipy as sc
15 import sklearn as skl
16 import csv as csv
17 import openpyxl
18 import pathlib
19 import os
20 from pathlib import Path
21 from openpyxl import load_workbook
22
23 # Import custom functions for navigation
24 import GOCPI as GF
25 # Import data case for the model
26
27 # Beginning of scripting
28 # Very Important Step: Set directory root for file operations.
29 root = '/Users/connor/Google Drive/Documents/University/Courses/2020/
    ENGSCI 700A&B/GOCPI/data/Inputs/GOCPI_OseMOSYS'
30 model_roots = Path(root)
31
32 # sets strings as excel file names for the model and parameter data.
33 model_file = 'GOCPI_OseMOSYS_Structure.xlsx'
34 data_file = 'GOCPI_OseMOSYS_Structure.xlsx'
35
36 # Finds the files necessary to create pandas dataframes.
37 # model = Find_File(model_roots, model_file)
38 Location = GF.Navigation(model_roots, model_file)
39 model = Location.Find_File()
40 print(model)
41
42 # data = Find_File(data_file, model_file)
43 df = pd.read_excel(model, sheet_name='Model')
44 # Creates a new dataframe based on the variables on the Include column
        values
45 df_Include = df[df.Include == 'Yes']
46 df_model = df_Include[['Name']].copy()
47
48 # Creates a file location and write the model to a text file
49 model_txt = 'GOCPI_OseMOSYS_Model.txt'
50 model_location = os.path.join(model_roots, model_txt)
51
52 # Saves the user defined model to a text file
53 np.savetxt(model_location, df_model.values, fmt='%s')
54
55 # Creates array of parameters from select sets and functions
56 df_Include = df[df.Include == 'Yes']
```

```

57 df_target_sets = df_Include[df.Type == "Sets"]
58 df_sets = df_target_sets[['Name']].copy()
59 df_target_parameters = df_Include[df.Type == "Parameters"]
60 df_parameters = df_target_parameters[['Name']].copy()
61
62 # Import the scenario with all sets and

```

6.6 GOCPI Optimisation

This script helped incorporate IBM optimisation technologies into the package.

```

1 #
# ##########
2 # GOCPI_Optimsation runs the optimisation through docplex
3 #
# #########
4
5 # Imports the necessary python modules
6 import numpy as np
7 import pandas as pd
8 import matplotlib.pyplot as plt
9 import scipy as sc
10 import sklearn as skl
11 import csv as csv
12 import openpyxl
13 import pathlib
14 import os
15 from pathlib import Path
16 from openpyxl import load_workbook
17 import GOCPI as GF
18 import cplex as cp
19 import subprocess as sb
20 import docplex.mp as dpmp
21 import tarfile as tf
22 from ibm_watson_machine_learning import APIClient
23
24 #
# #########
25 # Processing
26 #
# #########
27 # Initialise Optimisation Class
28 energy_system_optimisation = GF.Optimisation()
29 # Use Cplex on the IBM Cloud to create the optimisation techniques.
30 # Create APIClient to use your cloud platform
31 # API Key: (Bxhv-kuLYXfle61GiFIbR_uM7n_LA00u4X-RrMcgztE0) - IBM Cloud
# Access
32 apikey = "Bxhv-kuLYXfle61GiFIbR_uM7n_LA00u4X-RrMcgztE0"
33 url = "https://us-south.ml.cloud.ibm.com"
34 directory = '/Users/connor/Google Drive/Documents/University/Courses
# /2020/ENGSCI 700A&B/GOCPI/data/Inputs/GOCPI_OseMOSYS/'
35 data = 'GOCPI_Data.txt',
36 model = 'GOCPI_Model.txt'

```

```

37 output = 'GOCPI.lp'
38 payload_input = directory + output
39 payload_output = directory + "GOCPI.csv"
40 results = directory + "GOCPI_Output.txt"
41 tar_file = directory + "GOCPI.tar.gz"
42 csv = "GOCPI.csv"
43 csv_file = directory + csv
44 lp_file = directory + output
45 space_exists = True
46 deployment_exists = True
47 create_data_assets = False
48 string = 'glpsol -m ' + data + ' -d ' + model + '--wlp ' + output
49
50 api_wml_credentials = {
51     "apikey": apikey, # User Account API
52     #"instance_id":
53     #'2dc64ea2-6be8-43d0-b217-ec2a5743e8c9', # Watson Machine Learning
54     "url": url
55 }
56
57 # Initialises client credentials
58 client = APIClient(api_wml_credentials)
59
60 # Create a deployment space and set it
61 space_name = 'gocpi_deployment_space'
62 cos_resource_crn = 'crn:v1:bluemix:public:cloud-object-storage:global:a
   /09d7320da1734f7e84aaedf597c37111:83e6751a-cefc-49ce-93de-4
   fbaee7e52af::'
63 instance_crn = 'crn:v1:bluemix:public:pm-20:us-south:a/09
   d7320da1734f7e84aaedf597c37111:2dc64ea2-6be8-43d0-b217-ec2a5743e8c9
   ::'
64
65 metadata = {
66     client.spaces.ConfigurationMetaNames.NAME: space_name,
67     client.spaces.ConfigurationMetaNames.DESCRIPTION:
68     space_name + ' for Deployment',
69     client.spaces.ConfigurationMetaNames.STORAGE: {
70         "type": "bmcos_object_storage",
71         "resource_crn": cos_resource_crn
72     },
73     client.spaces.ConfigurationMetaNames.COMPUTE: {
74         "name": "existing_instance_id",
75         "crn": instance_crn
76     }
77 }
78 # Set the default spaces based on the outcomes
79 if space_exists == True:
80     client.spaces.list()
81     space_id = input('Please input the Space ID: ')
82 else:
83     space = client.spaces.store(meta_props=metadata)
84     space_id = client.spaces.get_id(space)
85
86 # Set the client space
87 client.set.default_space(space_id)
88
89 # Create input and output data assets
90 if create_data_assets == True:

```

```

91     client.data_assets.create('GOCPI_Energy_System_Lp_File', lp_file)
92     client.data_assets.create('GOCPI_Energy_System_CSV_File', csv_file)
93
94
95 # Deploy model files
96 # Get location of model deployment
97 # Create tar file for model deployment
98 # Reset tarfile function (Source: IBM Watson Machine Learning)
99 def reset(tarinfo):
100     tarinfo.uid = tarinfo.gid = 0
101     tarinfo.uname = tarinfo.gname = "root"
102     return tarinfo
103
104
105 # Create the tar file
106 tar = tf.open(tar_file, "w:gz")
107 tar.add(lp_file, arcname="GOCPI.lp", filter=reset)
108 tar.close()
109
110 # List deployments using python API
111 print(client.deployments.list())
112
113 # Get the list of software available
114 client.software_specifications.list()
115 software_name = input("Please Input Software Name: ")
116 software_spec_uid = client.software_specifications.get_uid_by_name(
117     software_name)
118
119 # Create the model deployment using the created arc file
120 energy_system_model_metadata = {
121     client.repository.ModelMetaNames.NAME: "Energy System",
122     client.repository.ModelMetaNames.DESCRIPTION: "Model for Energy
System",
123     client.repository.ModelMetaNames.TYPE: "do-cplex_12.10",
124     client.repository.ModelMetaNames.RUNTIME_UID: "do_12.10",
125     client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:
software_spec_uid
126 }
127
128 energy_system_model_details = client.repository.store_model(
129     model=tar_file, meta_props=energy_system_model_metadata)
130
131 energy_system_model_uid = client.repository.get_model_uid(
132     energy_system_model_details)
133
134 # Create deployment
135 n_nodes = 1
136
137 meta_props = {
138     client.deployments.ConfigurationMetaNames.NAME:
139     "Energy System Deployment " + str(n_nodes),
140     client.deployments.ConfigurationMetaNames.DESCRIPTION:
141     "Energy System",
142     # client.deployments.ConfigurationMetaNames.HARDWARE_SPEC:
143     client.deployments.ConfigurationMetaNames.BATCH: {},
144     client.deployments.ConfigurationMetaNames.COMPUTE: {
145         'name': 'S',
146         'nodes': n_nodes

```

```

147 }
148 }
149
150 # Test if deployment already exists
151 if deployment_exists == True:
152     client.deployments.list()
153     deployment_uid = input('Please input the Deployment UID: ')
154 else:
155     deployment_details = client.deployments.create(
156         energy_system_model_uid,
157         meta_props=meta_props)
158     deployment_uid = client.deployments.get_uid(deployment_details)
159
160 # Designs Payload for deployment
161 # solve_payload = {
162 #     client.deployments.DecisionOptimizationMetaNames.SOLVE_PARAMETERS
163 #     : {
164 #         'oaas.logAttachmentName': 'log.txt',
165 #         'oaas.logTailEnabled': 'true',
166 #         'oaas.resultsFormat': 'JSON'
167 #     },
168 #     client.deployments.DecisionOptimizationMetaNames.
169 INPUT_DATA_REFERENCES: [
170 #         'id':
171 #         'GOCPI.lp',
172 #         'type':
173 #         's3',
174 #         'connection': {
175 #             'endpoint_url':
176 #             COS_ENDPOINT,
177 #             'access_key_id':
178 #             cos_resource_crn['cos_hmac_keys']["access_key_id"],
179 #             'secret_access_key':
180 #             cos_resource_crn['cos_hmac_keys']["secret_access_key"]
181 #         },
182 #         'location': {
183 #             'bucket': COS_BUCKET,
184 #             'path': lp_file
185 #         }
186 #     ],
187 #     client.deployments.DecisionOptimizationMetaNames.
188 OUTPUT_DATA_REFERENCES: [
189 #         'id':
190 #         'solution.json',
191 #         'type':
192 #         's3',
193 #         'connection': {
194 #             'endpoint_url':
195 #             url,
196 #             'access_key_id':
197 #             cos_resource_crn['cos_hmac_keys']["access_key_id"],
198 #             'secret_access_key':
199 #             cos_resource_crn['cos_hmac_keys']["secret_access_key"]
200 #         },
201 #         'location': {
202 #             'bucket': COS_BUCKET,
203 #             'path': 'solution.json'
204 #         }
205 #     ]
206 # 
```

```

200 #         }
201 #     }, {
202 #       'id':
203 #       'log.txt',
204 #       'type':
205 #       's3',
206 #       'connection': {
207 #         'endpoint_url':
208 #         url,
209 #         'access_key_id':
210 #         cos_credentials['cos_hmac_keys']["access_key_id"],
211 #         'secret_access_key':
212 #         cos_credentials['cos_hmac_keys']["secret_access_key"]
213 #       },
214 #       'location': {
215 #         'bucket': COS_BUCKET,
216 #         'path': 'log.txt'
217 #       }
218 #     }]
219 #   }
220 energy_system_payload = {
221     client.deployments.DecisionOptimizationMetaNames.SOLVE_PARAMETERS:
222     {
223       "oaas.logTailEnabled": "true"
224     }
225     # client.deployments.DecisionOptimizationMetaNames.INPUT_DATA: [
226     #   "id": lp_file
227     # ],
228     # client.deployments.DecisionOptimizationMetaNames.OUTPUT_DATA: [
229     #   "id":
230     #   csv_file
231     # ]
232   }
233   # job_details = client.deployments.create_job(deployment_uid,
234   #                                               solve_payload)
235   # job_uid = client.deployments.get_job_uid(job_details)
236
237   # Create jobs for the deployment
238   # job_details = client.deployments.create_job(deployment_uid,
239   #                                               energy_system_payload)
240   # job_uid = client.deployments.get_job_uid(job_details)
241
242   # Run job using deployment
243
244   # Deletes deployment
245
246   # # Find deployment pace ID
247   # def guid_from_space_name(client, space_name):
248   #   instance_details = client.service_instance.get_details()
249   #   space = client.spaces.get_details()
250   #   return (next(item for item in space['resources']
251   #               if item['entity']["name"] == space_name) ['metadata'
252   #               ]['guid'])
253
254   # Set the default client space
255   # instance_details = client.service_instance.get_instance_id()

```

```

255 # client.set.default_space()
256 # client.set.default_project()
257 # print(instance_details)
258 # client.set.default_space()
259 # client.set.default_project()
260 # # Create a data asset to the IBM Cloud
261
262 # files = {
263 #     'Energy Balances 1':
264 #         '/Users/connor/Google Drive/Documents/University/Courses/2020/
265 #             ENGSCI 700A&B/GOCPI/data/Energy Balances/IEAWorldEnergyBalances2017A
266 #                 -K.csv',
267 #     'Energy Balances 2':
268 #         '/Users/connor/Google Drive/Documents/University/Courses/2020/
269 #             ENGSCI 700A&B/GOCPI/data/Energy Balances/IEAWorldEnergyBalances2017L
270 #                 -Z.csv'
271 # }
272
273 # data_assets_to_create = ['Energy Balances 1', 'Energy Balances 2']
274 # created_assets = {}
275 # for assets in data_assets_to_create:
276 #     asset_details = client.data_assets.create(name="
277 #         Energy_System_Test",
278 #                                         file_path=files[assets
279 #                                         ])
280 #     created_assets[assets] = asset_details
281
282 # Get information of assets
283
284 Optimise = GF.Optimisation()
285 directory = '/Users/connor/Google Drive/Documents/University/Courses
286             /2020/ENGSCI 700A&B/GOCPI/data/Inputs/GOCPI_OseMOSYS/'
287 data = 'GOCPI_Data.txt'
288 model = 'GOCPI_Model.txt'
289 output = 'GOCPI.lp'
290 results = directory + "GOCPI_Output.txt"
291 string = 'glpsol -m ' + data + ' -d ' + model + '--wlp ' + output
292 # os.chdir(directory)
293 # os.system('conda init bash')
294 # os.system('conda activate osemosys')
295
296 # Solve locally using Cplex
297 energy_system_cplex = cp.Cplex()
298 # Read in the model file
299 output = energy_system_cplex.set_results_stream(None)
300 output = energy_system_cplex.set_log_stream(None)
301 # Write the loaded model to the energy system
302 energy_system_cplex.read(lp_file)
303 # Solve the model
304 energy_system_cplex.solve()
305 # Returns the objective value
306 objective_value = energy_system_cplex.solution.get_objective_value()
307 values = energy_system_cplex.solution.get_values()
308 print(np.size(values))
309
310 # Creates a prints model outputs
311 with cp.Cplex() as c, open(results, "w") as f:
312     output = c.set_results_stream(f)

```

```
306     output.write("GOCPI Example")
307
308 # Creates Docplex example
309 # energy_system_docplex = docplex.cp.model.CpoModel(name="GOCPI_Docplex
310 energy_system_docplex_lp = dpmp.model_reader.ModelReader.read(
311     lp_file, model_name='GOCPI_Docplex_Lp')
312
313 mdl = energy_system_docplex_lp.solve(url=url, api=apikey)
314 # print('mdl', mdl)
315 # return_code = sb.call("conda init bash", shell=True)
316 # return_code = sb.call("conda activate osemosys", shell=True)
317 # os.system('conda activate osemosys')
318 # Optimise.create_linear_programme_file(directory, data, model, output)
```

7 OseMOSYS

This section displays the text files formulated to create the lp file. These are formulated using Python-based processing scripts and the GOCPI Energysystems module.

7.1 Model File

```

1 set YEAR;
2 set TECHNOLOGY;
3 set TIMESLICE;
4 set FUEL;
5 set EMISSION;
6 set MODE_OF_OPERATION;
7 set REGION;
8 set SEASON;
9 set DAYTYPE;
10 set DAILYTIMEBRACKET;
11 set STORAGE;
12 param YearSplit{l in TIMESLICE,y in YEAR};
13 param DiscountRate{r in REGION};
14 param DaySplit{lh in DAILYTIMEBRACKET,y in YEAR};
15 param Conversionls{l in TIMESLICE,ls in SEASON};
16 param Conversionld{l in TIMESLICE,ld in DAYTYPE};
17 param Conversionlh{l in TIMESLICE,lh in DAILYTIMEBRACKET};
18 param DaysInDayType{ls in SEASON ,ld in DAYTYPE,y in YEAR};
19 param TradeRoute{r in REGION,rr in REGION,f in FUEL,y in YEAR};
20 param DepreciationMethod{r in REGION};
21 param SpecifiedAnnualDemand{r in REGION,f in FUEL,y in YEAR};
22 param SpecifiedDemandProfile{r in REGION,f in FUEL,l in TIMESLICE,y in
   YEAR};
23 param AccumulatedAnnualDemand{r in REGION,f in FUEL,y in YEAR};
24 param CapacityToActivityUnit{r in REGION,t in TECHNOLOGY};
25 param CapacityFactor{r in REGION,t in TECHNOLOGY,l in TIMESLICE,y in
   YEAR};
26 param AvailabilityFactor{r in REGION,t in TECHNOLOGY,y in YEAR};
27 param OperationalLife{r in REGION,t in TECHNOLOGY};
28 param ResidualCapacity{r in REGION,t in TECHNOLOGY,y in YEAR};
29 param InputActivityRatio{r in REGION,t in TECHNOLOGY,f in FUEL,m in
   MODE_OF_OPERATION,y in YEAR};
30 param OutputActivityRatio{r in REGION,t in TECHNOLOGY,f in FUEL,m in
   MODE_OF_OPERATION,y in YEAR};
31 param CapitalCost{r in REGION,t in TECHNOLOGY,y in YEAR};
32 param VariableCost{r in REGION,t in TECHNOLOGY,m in MODE_OF_OPERATION,y
   in YEAR};
33 param FixedCost{r in REGION,t in TECHNOLOGY,y in YEAR};
34 param TechnologyToStorage{r in REGION,t in TECHNOLOGY,s in STORAGE,m in
   MODE_OF_OPERATION};
35 param TechnologyFromStorage{r in REGION,t in TECHNOLOGY,s in STORAGE,m
   in MODE_OF_OPERATION};
36 param StorageLevelStart{r in REGION,s in STORAGE};
37 param StorageMaxChargeRate{r in REGION,s in STORAGE};
38 param StorageMaxDischargeRate{r in REGION,s in STORAGE};
39 param MinStorageCharge{r in REGION,s in STORAGE,y in YEAR};
40 param OperationalLifeStorage{r in REGION, s in STORAGE};
41 param CapitalCostStorage{r in REGION,s in STORAGE,y in YEAR};
42 param ResidualStorageCapacity{r in REGION,s in STORAGE,y in YEAR};

```

```

43 param CapacityOfOneTechnologyUnit{r in REGION,t in TECHNOLOGY,y in YEAR
    };
44 param TotalAnnualMaxCapacity{r in REGION,t in TECHNOLOGY,y in YEAR};
45 param TotalAnnualMinCapacity{r in REGION,t in TECHNOLOGY,y in YEAR};
46 param TotalAnnualMaxCapacityInvestment{r in REGION,t in TECHNOLOGY,y in
    YEAR};
47 param TotalAnnualMinCapacityInvestment{r in REGION,t in TECHNOLOGY,y in
    YEAR};
48 param TotalTechnologyAnnualActivityUpperLimit{r in REGION,t in
    TECHNOLOGY,y in YEAR};
49 param TotalTechnologyAnnualActivityLowerLimit{r in REGION,t in
    TECHNOLOGY,y in YEAR};
50 param TotalTechnologyModelPeriodActivityUpperLimit{r in REGION,t in
    TECHNOLOGY};
51 param TotalTechnologyModelPeriodActivityLowerLimit{r in REGION,t in
    TECHNOLOGY};
52 param ReserveMarginTagTechnology{r in REGION,t in TECHNOLOGY,y in YEAR
    };
53 param ReserveMarginTagFuel{r in REGION,f in FUEL,y in YEAR};
54 param ReserveMargin{r in REGION,y in YEAR};
55 param RETagTechnology{r in REGION,t in TECHNOLOGY,y in YEAR};
56 param RETagFuel{r in REGION,f in FUEL,y in YEAR};
57 param REMinProductionTarget{r in REGION,y in YEAR};
58 param EmissionActivityRatio{r in REGION,t in TECHNOLOGY,e in EMISSION,m
    in MODE_OF_OPERATION,y in YEAR};
59 param EmissionsPenalty{r in REGION,e in EMISSION,y in YEAR};
60 param AnnualExogenousEmission{r in REGION,e in EMISSION,y in YEAR};
61 param AnnualEmissionLimit{r in REGION,e in EMISSION,y in YEAR};
62 param ModelPeriodExogenousEmission{r in REGION,e in EMISSION};
63 param ModelPeriodEmissionLimit{r in REGION,e in EMISSION};
64 var RateOfDemand{r in REGION,l in TIMESLICE,f in FUEL,y in YEAR} >=0;
65 var Demand{r in REGION,l in TIMESLICE,f in FUEL,y in YEAR}>=0;
66 var RateOfStorageCharge{r in REGION,s in STORAGE,ls in SEASON,ld in
    DAYTYPE,lh in DAILYTIMEBRACKET,y in YEAR};
67 var RateOfStorageDischarge{r in REGION,s in STORAGE,ls in SEASON,ld in
    DAYTYPE,lh in DAILYTIMEBRACKET,y in YEAR};
68 var NetChargeWithinYear{r in REGION,s in STORAGE,ls in SEASON,ld in
    DAYTYPE,lh in DAILYTIMEBRACKET,y in YEAR};
69 var NetChargeWithinDay{r in REGION,s in STORAGE,ls in SEASON,ld in
    DAYTYPE,lh in DAILYTIMEBRACKET,y in YEAR};
70 var StorageLevelYearStart{r in REGION,s in STORAGE,y in YEAR}>=0;
71 var StorageLevelYearFinish{r in REGION,s in STORAGE,y in YEAR}>=0;
72 var StorageLevelSeasonStart{r in REGION,s in STORAGE,ls in SEASON,y in
    YEAR}>=0;
73 var StorageLevelDayTypeStart{r in REGION,s in STORAGE,ls in SEASON,ld
    in DAYTYPE,y in YEAR}>=0;
74 var StorageLevelDayTypeFinish{r in REGION,s in STORAGE,ls in SEASON,ld
    in DAYTYPE,y in YEAR}>=0;
75 var StorageLowerLimit{r in REGION,s in STORAGE,y in YEAR}>=0;
76 var StorageUpperLimit{r in REGION,s in STORAGE,y in YEAR}>=0;
77 var AccumulatedNewStorageCapacity{r in REGION,s in STORAGE,y in YEAR
    }>=0;
78 var NewStorageCapacity{r in REGION,s in STORAGE,y in YEAR}>=0;
79 var CapitalInvestmentStorage{r in REGION,s in STORAGE,y in YEAR}>=0;
80 var DiscountedCapitalInvestmentStorage{r in REGION,s in STORAGE,y in
    YEAR}>=0;
81 var SalvageValueStorage{r in REGION,s in STORAGE,y in YEAR}>=0;

```

```

82 var DiscountedSalvageValueStorage{r in REGION,s in STORAGE,y in YEAR
83   }>=0;
84 var TotalDiscountedStorageCost{r in REGION,s in STORAGE,y in YEAR}>=0;
85 var NumberOfNewTechnologyUnits{r in REGION,t in TECHNOLOGY,y in YEAR
86   }>=0, integer;
87 var NewCapacity{r in REGION,t in TECHNOLOGY,y in YEAR}>=0;
88 var AccumulatedNewCapacity{r in REGION,t in TECHNOLOGY,y in YEAR}>=0;
89 var TotalCapacityAnnual{r in REGION,t in TECHNOLOGY,y in YEAR}>=0;
90 var RateOfActivity{r in REGION,l in TIMESLICE,t in TECHNOLOGY,m in
91   MODE_OF_OPERATION,y in YEAR} >=0;
92 var RateOfTotalActivity{r in REGION,t in TECHNOLOGY,l in TIMESLICE,y in
93   YEAR} >=0;
94 var TotalTechnologyAnnualActivity{r in REGION,t in TECHNOLOGY,y in YEAR
95   } >=0;
96 var TotalAnnualTechnologyActivityByMode{r in REGION,t in TECHNOLOGY,m
97   in MODE_OF_OPERATION,y in YEAR} >=0;
98 var TotalTechnologyModelPeriodActivity{r in REGION,t in TECHNOLOGY};
99 var RateOfProductionByTechnologyByMode{r in REGION,l in TIMESLICE,t in
100  TECHNOLOGY,m in MODE_OF_OPERATION,f in FUEL,y in YEAR}>=0;
101 var RateOfProductionByTechnology{r in REGION,l in TIMESLICE,t in
102  TECHNOLOGY,f in FUEL,y in YEAR} >=0;
103 var ProductionByTechnology{r in REGION,l in TIMESLICE,t in TECHNOLOGY,f
104  in FUEL,y in YEAR} >=0;
105 var ProductionByTechnologyAnnual{r in REGION,t in TECHNOLOGY,f in FUEL,
106  y in YEAR} >=0;
107 var RateOfProduction{r in REGION,l in TIMESLICE,f in FUEL,y in YEAR
108   }>=0;
109 var Production{r in REGION,l in TIMESLICE,f in FUEL,y in YEAR} >=0;
110 var RateOfUseByTechnologyByMode{r in REGION,l in TIMESLICE,t in
111  TECHNOLOGY,m in MODE_OF_OPERATION,f in FUEL,y in YEAR} >=0;
112 var RateOfUseByTechnology{r in REGION,l in TIMESLICE,t in TECHNOLOGY,f
113  in FUEL,y in YEAR} >=0;
114 var UseByTechnologyAnnual{r in REGION,t in TECHNOLOGY,f in FUEL,y in
115  YEAR} >=0;
116 var RateOfUse{r in REGION,l in TIMESLICE,f in FUEL,y in YEAR}>=0;
117 var Use{r in REGION,l in TIMESLICE,f in FUEL,y in YEAR} >=0;
118 var Trade{r in REGION,rr in REGION,l in TIMESLICE,f in FUEL,y in YEAR};
119 var TradeAnnual{r in REGION,rr in REGION,f in FUEL,y in YEAR};
120 var ProductionAnnual{r in REGION,f in FUEL,y in YEAR}>=0;
121 var UseAnnual{r in REGION,f in FUEL,y in YEAR}>=0;
122 var CapitalInvestment{r in REGION,t in TECHNOLOGY,y in YEAR}>=0;
123 var DiscountedCapitalInvestment{r in REGION,t in TECHNOLOGY,y in YEAR}
124   >=0;
125 var SalvageValue{r in REGION,t in TECHNOLOGY,y in YEAR} >=0;
126 var DiscountedSalvageValue{r in REGION,t in TECHNOLOGY,y in YEAR} >=0;
127 var OperatingCost{r in REGION,t in TECHNOLOGY,y in YEAR}>=0;
128 var DiscountedOperatingCost{r in REGION,t in TECHNOLOGY,y in YEAR}>=0;
129 var AnnualVariableOperatingCost{r in REGION,t in TECHNOLOGY,y in YEAR}
130   >=0;
131 var AnnualFixedOperatingCost{r in REGION,t in TECHNOLOGY,y in YEAR}
132   >=0;
133 var TotalDiscountedCostByTechnology{r in REGION,t in TECHNOLOGY,y in
134  YEAR} >=0;
135 var TotalDiscountedCost{r in REGION,y in YEAR}>=0;
136 var ModelPeriodCostByRegion{r in REGION}>=0;
137 var TotalCapacityInReserveMargin{r in REGION,y in YEAR}>=0;

```

```

121 var DemandNeedingReserveMargin{r in REGION,l in TIMESLICE,y in YEAR}
122   >=0;
123 var TotalREProductionAnnual{r in REGION,y in YEAR};
124 var RETotalProductionOfTargetFuelAnnual{r in REGION,y in YEAR};
125 var AnnualTechnologyEmissionByMode{r in REGION,t in TECHNOLOGY,e in
126   EMISSION,m in MODE_OF_OPERATION,y in YEAR} >=0;
127 var AnnualTechnologyEmission{r in REGION,t in TECHNOLOGY,e in EMISSION,
128   y in YEAR} >=0;
129 var AnnualTechnologyEmissionPenaltyByEmission{r in REGION,t in
130   TECHNOLOGY,e in EMISSION,y in YEAR} >=0;
131 var AnnualTechnologyEmissionsPenalty{r in REGION,t in TECHNOLOGY,y in
132   YEAR} >=0;
133 var DiscountedTechnologyEmissionsPenalty{r in REGION,t in TECHNOLOGY,y
134   in YEAR} >=0;
135 var AnnualEmissions{r in REGION,e in EMISSION,y in YEAR} >=0;
136 var ModelPeriodEmissions{r in REGION,e in EMISSION} >=0;
137 minimize cost: sum{r in REGION, y in YEAR} TotalDiscountedCost[r,y];
138 s.t. EQ_SpecifiedDemand{r in REGION, l in TIMESLICE, f in FUEL, y in
139   YEAR}: SpecifiedAnnualDemand[r,f,y]*SpecifiedDemandProfile[r,f,l,y]
140   / YearSplit[l,y]=RateOfDemand[r,l,f,y];
141 s.t. CAa1_TotalNewCapacity{r in REGION, t in TECHNOLOGY, y in YEAR}:
142   AccumulatedNewCapacity[r,t,y] = sum{yy in YEAR: y-yy <
143     OperationalLife[r,t] && y-yy>=0} NewCapacity[r,t,yy];
144 s.t. CAa2_TotalAnnualCapacity{r in REGION, t in TECHNOLOGY, y in YEAR}:
145   AccumulatedNewCapacity[r,t,y]+ ResidualCapacity[r,t,y] =
146   TotalCapacityAnnual[r,t,y];
147 s.t. CAa3_TotalActivityOfEachTechnology{r in REGION, t in TECHNOLOGY, l
148   in TIMESLICE, y in YEAR}: sum{m in MODE_OF_OPERATION}
149   RateOfActivity[r,l,t,m,y] = RateOfTotalActivity[r,t,l,y];
150 s.t. CAa4_Constraint_Capacity{r in REGION, l in TIMESLICE, t in
151   TECHNOLOGY, y in YEAR}: RateOfTotalActivity[r,t,l,y] <=
152   TotalCapacityAnnual[r,t,y] * CapacityFactor[r,t,l,y]*
153   CapacityToActivityUnit[r,t];
154 s.t. CAa5_TotalNewCapacity{r in REGION, t in TECHNOLOGY, y in YEAR}:
155   CapacityOfOneTechnologyUnit[r,t,y]<>0: CapacityOfOneTechnologyUnit[
156     r,t,y]*NumberOfNewTechnologyUnits[r,t,y] = NewCapacity[r,t,y];
157 s.t. CAB1_PlannedMaintenance{r in REGION, t in TECHNOLOGY, y in YEAR}:
158   sum{l in TIMESLICE} RateOfTotalActivity[r,t,l,y]*YearSplit[l,y] <=
159   sum{l in TIMESLICE} (TotalCapacityAnnual[r,t,y]*CapacityFactor[r,t,l
160   ,y]*YearSplit[l,y])* AvailabilityFactor[r,t,y]*
161   CapacityToActivityUnit[r,t];
162 s.t. EBa1_RateOfFuelProduction1{r in REGION, l in TIMESLICE, f in FUEL,
163   t in TECHNOLOGY, m in MODE_OF_OPERATION, y in YEAR}:
164   OutputActivityRatio[r,t,f,m,y] <>0: RateOfActivity[r,l,t,m,y]*
165   OutputActivityRatio[r,t,f,m,y] = RateOfProductionByTechnologyByMode
166   [r,l,t,m,f,y];
167 s.t. EBa2_RateOfFuelProduction2{r in REGION, l in TIMESLICE, f in FUEL,
168   t in TECHNOLOGY, y in YEAR}: sum{m in MODE_OF_OPERATION:
169   OutputActivityRatio[r,t,f,m,y] <>0}
170   RateOfProductionByTechnologyByMode[r,l,t,m,f,y] =
171   RateOfProductionByTechnology[r,l,t,f,y];
172 s.t. EBa3_RateOfFuelProduction3{r in REGION, l in TIMESLICE, f in FUEL,
173   y in YEAR}: sum{t in TECHNOLOGY} RateOfProductionByTechnology[r,l,t
174   ,f,y] = RateOfProduction[r,l,f,y];
175 s.t. EBa4_RateOfFuelUse1{r in REGION, l in TIMESLICE, f in FUEL, t in
176   TECHNOLOGY, m in MODE_OF_OPERATION, y in YEAR: InputActivityRatio[r,
177   t,f,m,y]<>0}: RateOfActivity[r,l,t,m,y]*InputActivityRatio[r,t,f,m,y
178   ] = 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,
161 ld, lh, y]) * YearSplit[l, y] * Conversionls[l, ls] * Conversionld[l, ld]
* Conversionlh[l, lh] = NetChargeWithinYear[r,s,ls,ld,lh,y];
s.t. S4_NetChargeWithinDay{r in REGION, s in STORAGE, ls in SEASON, ld
in DAYTYPE, lh in DAILYTIMEBRACKET, y in YEAR}: (RateOfStorageCharge
[162 [r,s,ls,ld,lh,y] - RateOfStorageDischarge[r,s,ls,ld,lh,y]) *
DaySplit[lh,y] = NetChargeWithinDay[r,s,ls,ld,lh,y];
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{ls in SEASON} min(ls) 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{ld in DAYTYPE}
min(ld) 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{ls in
SEASON} max(ls) && ld = max{ld in DAYTYPE} max(ld) then
StorageLevelYearFinish[r,s,y] else if ld = max{ld in DAYTYPE} max(
ld) 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{lh in DAILYTIMEBRACKET:lh-lh>0} NetChargeWithinDay[r,s
,ls,ld,lh,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{lh in DAILYTIMEBRACKET:lh-lh>0} NetChargeWithinDay[r,s,ls,
ld,lh,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{ld in DAYTYPE} min(
ld) then (StorageLevelDayTypeStart[r,s,ls,ld,y]-sum{lh in
DAILYTIMEBRACKET:lh-lh<0} NetChargeWithinDay[r,s,ls,ld-1,lh,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}: 0 <= if ld > min{ld in DAYTYPE} min(
ld) then (StorageLevelDayTypeStart[r,s,ls,ld,y]-sum{lh in
DAILYTIMEBRACKET:lh-lh<0} NetChargeWithinDay[r,s,ls,ld-1,lh,y])-_
StorageUpperLimit[r,s,y];

```

```

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;
171 s.t.
  SC3_LowerLimit_EndOfDayTimeBracketOfLastInstanceOfDayTypeInLastWeekConstraint
  {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];
172 s.t.
  SC3_UpperLimit_EndOfDayTimeBracketOfLastInstanceOfDayTypeInLastWeekConstraint
  {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;
173 s.t.
  SC4_LowerLimit_BeginningOfDayTimeBracketOfFirstInstanceOfDayTypeInLastWeekCons
  {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];
174 s.t.
  SC4_UpperLimit_BeginningOfDayTimeBracketOfFirstInstanceOfDayTypeInLastWeekCons
  {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;
175 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];
176 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];
177 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];
178 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];
179 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];
180 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];
181 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];
182 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];
183 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)))}: 0 = SalvageValueStorage[r,s,y];

```

```

] -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_TotalModelHorizonTechnologyActivityLowerLimit{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];
215 s.t. RE4_EnergyConstraint{r in REGION, y in YEAR}:
    REMinProductionTarget[r,y]*RETotalProductionOfTargetFuelAnnual[r,y]
    <= TotalREProductionAnnual[r,y];
216 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];
217 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];
218 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];
219 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];
220 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];
221 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];
222 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];
223 s.t. E7_EmissionsAccounting2{r in REGION, e in EMISSION}: sum{y in YEAR
    } AnnualEmissions[r,e,y] = ModelPeriodEmissions[r,e]-
    ModelPeriodExogenousEmission[r,e];
224 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];
225 s.t. E9_ModelPeriodEmissionsLimit{r in REGION, e in EMISSION}:
    ModelPeriodEmissions[r,e] <= ModelPeriodEmissionLimit[r,e];
226 solve;
227 end;

```

7.2 Data File

This data file is created from a partially complete NZ/AUS Energy system. The file shows the complexity of energy modelling and creating user-defined energy systems. The parameters to be modelled are mostly denoted by binary values.

```

1 # GOCPI Energy System Data File
2 # Insert instructions when the file is running properly
3 #
4 # Sets
5 #
6 set YEAR := 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000
    2001 2002 2003 2004 2005 2006 2007 2008 2009 2010;
7 set REGION := NEWZEALAND AUSTRALIA;
8 set EMISSION := CO2 NOX CO METHANE;

```

```

9 set TECHNOLOGY := E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
   IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu;
10 set FUEL := CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX;
11 set TIMESLICE := INTERMEDIATE_DAY INTERMEDIATE_NIGHT SUMMER_DAY
   SUMMER_NIGHT WINTER_DAY WINTER_NIGHT;
12 set MODE_OF_OPERATION := 1 2;
13 set STORAGE := DAM;
14 set DAYTYPE := 1 2 3;
15 set SEASON := 1 2 3 4;
16 set DAILYTIMEBRACKET := 1 2 3;
17 #
18 #
19 param YearSplit :1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000
   2001 2002 2003 2004 2005 2006 2007 2008 2009 2010:=
20 INTERMEDIATE_DAY 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
   1.0 1.0 1.0 1.0 1.0 1.0 1.0
21 INTERMEDIATE_NIGHT 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
   1.0 1.0 1.0 1.0 1.0 1.0 1.0
22 SUMMER_DAY 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
   1.0 1.0 1.0 1.0 1.0 1.0 1.0
23 SUMMER_NIGHT 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
   1.0 1.0 1.0 1.0 1.0 1.0 1.0
24 WINTER_DAY 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
   1.0 1.0 1.0 1.0 1.0 1.0 1.0
25 WINTER_NIGHT 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
   1.0 1.0 1.0 1.0 1.0 1.0 1.0
26 ;
27 #
28 param DiscountRate default 1:-
29 NEWZEALAND 1.0
30 AUSTRALIA 1.0
31 ;
32 #
33 param DaySplit :1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000
   2001 2002 2003 2004 2005 2006 2007 2008 2009 2010:=
34 1 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
   1.0 1.0 1.0 1.0
35 2 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
   1.0 1.0 1.0 1.0
36 3 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
   1.0 1.0 1.0 1.0
37 ;
38 #
39 param Conversionls :1 2 3 4:-
40 INTERMEDIATE_DAY 1.0 1.0 1.0 1.0
41 INTERMEDIATE_NIGHT 1.0 1.0 1.0 1.0
42 SUMMER_DAY 1.0 1.0 1.0 1.0
43 SUMMER_NIGHT 1.0 1.0 1.0 1.0
44 WINTER_DAY 1.0 1.0 1.0 1.0
45 WINTER_NIGHT 1.0 1.0 1.0 1.0
46 ;
47 #
48 param Conversionld :1 2 3:-
49 INTERMEDIATE_DAY 1.0 1.0 1.0
50 INTERMEDIATE_NIGHT 1.0 1.0 1.0
51 SUMMER_DAY 1.0 1.0 1.0
52 SUMMER_NIGHT 1.0 1.0 1.0
53 WINTER_DAY 1.0 1.0 1.0

```

```
54 WINTER_NIGHT 1.0 1.0 1.0
55 ;
56 #
57 param Conversionlh :1 2 3:=
58 INTERMEDIATE_DAY 1.0 1.0 1.0
59 INTERMEDIATE_NIGHT 1.0 1.0 1.0
60 SUMMER_DAY 1.0 1.0 1.0
61 SUMMER_NIGHT 1.0 1.0 1.0
62 WINTER_DAY 1.0 1.0 1.0
63 WINTER_NIGHT 1.0 1.0 1.0
64 ;
65 #
66 param DaysInDayType default 1:=
67   [*,* ,1990]: 1 2 3 := 
68   1 1.0 1.0 1.0
69   2 1.0 1.0 1.0
70   3 1.0 1.0 1.0
71   4 1.0 1.0 1.0
72   [*,* ,1991]: 1 2 3 := 
73   1 1.0 1.0 1.0
74   2 1.0 1.0 1.0
75   3 1.0 1.0 1.0
76   4 1.0 1.0 1.0
77   [*,* ,1992]: 1 2 3 := 
78   1 1.0 1.0 1.0
79   2 1.0 1.0 1.0
80   3 1.0 1.0 1.0
81   4 1.0 1.0 1.0
82   [*,* ,1993]: 1 2 3 := 
83   1 1.0 1.0 1.0
84   2 1.0 1.0 1.0
85   3 1.0 1.0 1.0
86   4 1.0 1.0 1.0
87   [*,* ,1994]: 1 2 3 := 
88   1 1.0 1.0 1.0
89   2 1.0 1.0 1.0
90   3 1.0 1.0 1.0
91   4 1.0 1.0 1.0
92   [*,* ,1995]: 1 2 3 := 
93   1 1.0 1.0 1.0
94   2 1.0 1.0 1.0
95   3 1.0 1.0 1.0
96   4 1.0 1.0 1.0
97   [*,* ,1996]: 1 2 3 := 
98   1 1.0 1.0 1.0
99   2 1.0 1.0 1.0
100  3 1.0 1.0 1.0
101  4 1.0 1.0 1.0
102  [*,* ,1997]: 1 2 3 := 
103  1 1.0 1.0 1.0
104  2 1.0 1.0 1.0
105  3 1.0 1.0 1.0
106  4 1.0 1.0 1.0
107  [*,* ,1998]: 1 2 3 := 
108  1 1.0 1.0 1.0
109  2 1.0 1.0 1.0
110  3 1.0 1.0 1.0
111  4 1.0 1.0 1.0
```

```
112 [*,* ,1999]: 1 2 3 :=  
113 1 1.0 1.0 1.0  
114 2 1.0 1.0 1.0  
115 3 1.0 1.0 1.0  
116 4 1.0 1.0 1.0  
117 [*,* ,2000]: 1 2 3 :=  
118 1 1.0 1.0 1.0  
119 2 1.0 1.0 1.0  
120 3 1.0 1.0 1.0  
121 4 1.0 1.0 1.0  
122 [*,* ,2001]: 1 2 3 :=  
123 1 1.0 1.0 1.0  
124 2 1.0 1.0 1.0  
125 3 1.0 1.0 1.0  
126 4 1.0 1.0 1.0  
127 [*,* ,2002]: 1 2 3 :=  
128 1 1.0 1.0 1.0  
129 2 1.0 1.0 1.0  
130 3 1.0 1.0 1.0  
131 4 1.0 1.0 1.0  
132 [*,* ,2003]: 1 2 3 :=  
133 1 1.0 1.0 1.0  
134 2 1.0 1.0 1.0  
135 3 1.0 1.0 1.0  
136 4 1.0 1.0 1.0  
137 [*,* ,2004]: 1 2 3 :=  
138 1 1.0 1.0 1.0  
139 2 1.0 1.0 1.0  
140 3 1.0 1.0 1.0  
141 4 1.0 1.0 1.0  
142 [*,* ,2005]: 1 2 3 :=  
143 1 1.0 1.0 1.0  
144 2 1.0 1.0 1.0  
145 3 1.0 1.0 1.0  
146 4 1.0 1.0 1.0  
147 [*,* ,2006]: 1 2 3 :=  
148 1 1.0 1.0 1.0  
149 2 1.0 1.0 1.0  
150 3 1.0 1.0 1.0  
151 4 1.0 1.0 1.0  
152 [*,* ,2007]: 1 2 3 :=  
153 1 1.0 1.0 1.0  
154 2 1.0 1.0 1.0  
155 3 1.0 1.0 1.0  
156 4 1.0 1.0 1.0  
157 [*,* ,2008]: 1 2 3 :=  
158 1 1.0 1.0 1.0  
159 2 1.0 1.0 1.0  
160 3 1.0 1.0 1.0  
161 4 1.0 1.0 1.0  
162 [*,* ,2009]: 1 2 3 :=  
163 1 1.0 1.0 1.0  
164 2 1.0 1.0 1.0  
165 3 1.0 1.0 1.0  
166 4 1.0 1.0 1.0  
167 [*,* ,2010]: 1 2 3 :=  
168 1 1.0 1.0 1.0  
169 2 1.0 1.0 1.0
```

```
170 3 1.0 1.0 1.0
171 4 1.0 1.0 1.0
172 ;
173 #
174 param TradeRoute default 1:=
175 [*,* ,CSV ,1990]: NEWZEALAND AUSTRALIA :=
176 NEWZEALAND 1.0 1.0
177 AUSTRALIA 1.0 1.0
178 [*,* ,CSV ,1991]: NEWZEALAND AUSTRALIA :=
179 NEWZEALAND 1.0 1.0
180 AUSTRALIA 1.0 1.0
181 [*,* ,CSV ,1992]: NEWZEALAND AUSTRALIA :=
182 NEWZEALAND 1.0 1.0
183 AUSTRALIA 1.0 1.0
184 [*,* ,CSV ,1993]: NEWZEALAND AUSTRALIA :=
185 NEWZEALAND 1.0 1.0
186 AUSTRALIA 1.0 1.0
187 [*,* ,CSV ,1994]: NEWZEALAND AUSTRALIA :=
188 NEWZEALAND 1.0 1.0
189 AUSTRALIA 1.0 1.0
190 [*,* ,CSV ,1995]: NEWZEALAND AUSTRALIA :=
191 NEWZEALAND 1.0 1.0
192 AUSTRALIA 1.0 1.0
193 [*,* ,CSV ,1996]: NEWZEALAND AUSTRALIA :=
194 NEWZEALAND 1.0 1.0
195 AUSTRALIA 1.0 1.0
196 [*,* ,CSV ,1997]: NEWZEALAND AUSTRALIA :=
197 NEWZEALAND 1.0 1.0
198 AUSTRALIA 1.0 1.0
199 [*,* ,CSV ,1998]: NEWZEALAND AUSTRALIA :=
200 NEWZEALAND 1.0 1.0
201 AUSTRALIA 1.0 1.0
202 [*,* ,CSV ,1999]: NEWZEALAND AUSTRALIA :=
203 NEWZEALAND 1.0 1.0
204 AUSTRALIA 1.0 1.0
205 [*,* ,CSV ,2000]: NEWZEALAND AUSTRALIA :=
206 NEWZEALAND 1.0 1.0
207 AUSTRALIA 1.0 1.0
208 [*,* ,CSV ,2001]: NEWZEALAND AUSTRALIA :=
209 NEWZEALAND 1.0 1.0
210 AUSTRALIA 1.0 1.0
211 [*,* ,CSV ,2002]: NEWZEALAND AUSTRALIA :=
212 NEWZEALAND 1.0 1.0
213 AUSTRALIA 1.0 1.0
214 [*,* ,CSV ,2003]: NEWZEALAND AUSTRALIA :=
215 NEWZEALAND 1.0 1.0
216 AUSTRALIA 1.0 1.0
217 [*,* ,CSV ,2004]: NEWZEALAND AUSTRALIA :=
218 NEWZEALAND 1.0 1.0
219 AUSTRALIA 1.0 1.0
220 [*,* ,CSV ,2005]: NEWZEALAND AUSTRALIA :=
221 NEWZEALAND 1.0 1.0
222 AUSTRALIA 1.0 1.0
223 [*,* ,CSV ,2006]: NEWZEALAND AUSTRALIA :=
224 NEWZEALAND 1.0 1.0
225 AUSTRALIA 1.0 1.0
226 [*,* ,CSV ,2007]: NEWZEALAND AUSTRALIA :=
227 NEWZEALAND 1.0 1.0
```

```
228 AUSTRALIA 1.0 1.0
229 [*,* ,CSV ,2008]: NEWZEALAND AUSTRALIA := 
230 NEWZEALAND 1.0 1.0
231 AUSTRALIA 1.0 1.0
232 [*,* ,CSV ,2009]: NEWZEALAND AUSTRALIA := 
233 NEWZEALAND 1.0 1.0
234 AUSTRALIA 1.0 1.0
235 [*,* ,CSV ,2010]: NEWZEALAND AUSTRALIA := 
236 NEWZEALAND 1.0 1.0
237 AUSTRALIA 1.0 1.0
238 [*,* ,DSL ,1990]: NEWZEALAND AUSTRALIA := 
239 NEWZEALAND 1.0 1.0
240 AUSTRALIA 1.0 1.0
241 [*,* ,DSL ,1991]: NEWZEALAND AUSTRALIA := 
242 NEWZEALAND 1.0 1.0
243 AUSTRALIA 1.0 1.0
244 [*,* ,DSL ,1992]: NEWZEALAND AUSTRALIA := 
245 NEWZEALAND 1.0 1.0
246 AUSTRALIA 1.0 1.0
247 [*,* ,DSL ,1993]: NEWZEALAND AUSTRALIA := 
248 NEWZEALAND 1.0 1.0
249 AUSTRALIA 1.0 1.0
250 [*,* ,DSL ,1994]: NEWZEALAND AUSTRALIA := 
251 NEWZEALAND 1.0 1.0
252 AUSTRALIA 1.0 1.0
253 [*,* ,DSL ,1995]: NEWZEALAND AUSTRALIA := 
254 NEWZEALAND 1.0 1.0
255 AUSTRALIA 1.0 1.0
256 [*,* ,DSL ,1996]: NEWZEALAND AUSTRALIA := 
257 NEWZEALAND 1.0 1.0
258 AUSTRALIA 1.0 1.0
259 [*,* ,DSL ,1997]: NEWZEALAND AUSTRALIA := 
260 NEWZEALAND 1.0 1.0
261 AUSTRALIA 1.0 1.0
262 [*,* ,DSL ,1998]: NEWZEALAND AUSTRALIA := 
263 NEWZEALAND 1.0 1.0
264 AUSTRALIA 1.0 1.0
265 [*,* ,DSL ,1999]: NEWZEALAND AUSTRALIA := 
266 NEWZEALAND 1.0 1.0
267 AUSTRALIA 1.0 1.0
268 [*,* ,DSL ,2000]: NEWZEALAND AUSTRALIA := 
269 NEWZEALAND 1.0 1.0
270 AUSTRALIA 1.0 1.0
271 [*,* ,DSL ,2001]: NEWZEALAND AUSTRALIA := 
272 NEWZEALAND 1.0 1.0
273 AUSTRALIA 1.0 1.0
274 [*,* ,DSL ,2002]: NEWZEALAND AUSTRALIA := 
275 NEWZEALAND 1.0 1.0
276 AUSTRALIA 1.0 1.0
277 [*,* ,DSL ,2003]: NEWZEALAND AUSTRALIA := 
278 NEWZEALAND 1.0 1.0
279 AUSTRALIA 1.0 1.0
280 [*,* ,DSL ,2004]: NEWZEALAND AUSTRALIA := 
281 NEWZEALAND 1.0 1.0
282 AUSTRALIA 1.0 1.0
283 [*,* ,DSL ,2005]: NEWZEALAND AUSTRALIA := 
284 NEWZEALAND 1.0 1.0
285 AUSTRALIA 1.0 1.0
```

```
286 [*,* ,DSL ,2006]: NEWZEALAND AUSTRALIA  :=  
287 NEWZEALAND 1.0 1.0  
288 AUSTRALIA 1.0 1.0  
289 [*,* ,DSL ,2007]: NEWZEALAND AUSTRALIA  :=  
290 NEWZEALAND 1.0 1.0  
291 AUSTRALIA 1.0 1.0  
292 [*,* ,DSL ,2008]: NEWZEALAND AUSTRALIA  :=  
293 NEWZEALAND 1.0 1.0  
294 AUSTRALIA 1.0 1.0  
295 [*,* ,DSL ,2009]: NEWZEALAND AUSTRALIA  :=  
296 NEWZEALAND 1.0 1.0  
297 AUSTRALIA 1.0 1.0  
298 [*,* ,DSL ,2010]: NEWZEALAND AUSTRALIA  :=  
299 NEWZEALAND 1.0 1.0  
300 AUSTRALIA 1.0 1.0  
301 [*,* ,ELC ,1990]: NEWZEALAND AUSTRALIA  :=  
302 NEWZEALAND 1.0 1.0  
303 AUSTRALIA 1.0 1.0  
304 [*,* ,ELC ,1991]: NEWZEALAND AUSTRALIA  :=  
305 NEWZEALAND 1.0 1.0  
306 AUSTRALIA 1.0 1.0  
307 [*,* ,ELC ,1992]: NEWZEALAND AUSTRALIA  :=  
308 NEWZEALAND 1.0 1.0  
309 AUSTRALIA 1.0 1.0  
310 [*,* ,ELC ,1993]: NEWZEALAND AUSTRALIA  :=  
311 NEWZEALAND 1.0 1.0  
312 AUSTRALIA 1.0 1.0  
313 [*,* ,ELC ,1994]: NEWZEALAND AUSTRALIA  :=  
314 NEWZEALAND 1.0 1.0  
315 AUSTRALIA 1.0 1.0  
316 [*,* ,ELC ,1995]: NEWZEALAND AUSTRALIA  :=  
317 NEWZEALAND 1.0 1.0  
318 AUSTRALIA 1.0 1.0  
319 [*,* ,ELC ,1996]: NEWZEALAND AUSTRALIA  :=  
320 NEWZEALAND 1.0 1.0  
321 AUSTRALIA 1.0 1.0  
322 [*,* ,ELC ,1997]: NEWZEALAND AUSTRALIA  :=  
323 NEWZEALAND 1.0 1.0  
324 AUSTRALIA 1.0 1.0  
325 [*,* ,ELC ,1998]: NEWZEALAND AUSTRALIA  :=  
326 NEWZEALAND 1.0 1.0  
327 AUSTRALIA 1.0 1.0  
328 [*,* ,ELC ,1999]: NEWZEALAND AUSTRALIA  :=  
329 NEWZEALAND 1.0 1.0  
330 AUSTRALIA 1.0 1.0  
331 [*,* ,ELC ,2000]: NEWZEALAND AUSTRALIA  :=  
332 NEWZEALAND 1.0 1.0  
333 AUSTRALIA 1.0 1.0  
334 [*,* ,ELC ,2001]: NEWZEALAND AUSTRALIA  :=  
335 NEWZEALAND 1.0 1.0  
336 AUSTRALIA 1.0 1.0  
337 [*,* ,ELC ,2002]: NEWZEALAND AUSTRALIA  :=  
338 NEWZEALAND 1.0 1.0  
339 AUSTRALIA 1.0 1.0  
340 [*,* ,ELC ,2003]: NEWZEALAND AUSTRALIA  :=  
341 NEWZEALAND 1.0 1.0  
342 AUSTRALIA 1.0 1.0  
343 [*,* ,ELC ,2004]: NEWZEALAND AUSTRALIA  :=
```

```
344 NEWZEALAND 1.0 1.0
345 AUSTRALIA 1.0 1.0
346 [*,* ,ELC ,2005]: NEWZEALAND AUSTRALIA := 
347 NEWZEALAND 1.0 1.0
348 AUSTRALIA 1.0 1.0
349 [*,* ,ELC ,2006]: NEWZEALAND AUSTRALIA := 
350 NEWZEALAND 1.0 1.0
351 AUSTRALIA 1.0 1.0
352 [*,* ,ELC ,2007]: NEWZEALAND AUSTRALIA := 
353 NEWZEALAND 1.0 1.0
354 AUSTRALIA 1.0 1.0
355 [*,* ,ELC ,2008]: NEWZEALAND AUSTRALIA := 
356 NEWZEALAND 1.0 1.0
357 AUSTRALIA 1.0 1.0
358 [*,* ,ELC ,2009]: NEWZEALAND AUSTRALIA := 
359 NEWZEALAND 1.0 1.0
360 AUSTRALIA 1.0 1.0
361 [*,* ,ELC ,2010]: NEWZEALAND AUSTRALIA := 
362 NEWZEALAND 1.0 1.0
363 AUSTRALIA 1.0 1.0
364 [*,* ,GSL ,1990]: NEWZEALAND AUSTRALIA := 
365 NEWZEALAND 1.0 1.0
366 AUSTRALIA 1.0 1.0
367 [*,* ,GSL ,1991]: NEWZEALAND AUSTRALIA := 
368 NEWZEALAND 1.0 1.0
369 AUSTRALIA 1.0 1.0
370 [*,* ,GSL ,1992]: NEWZEALAND AUSTRALIA := 
371 NEWZEALAND 1.0 1.0
372 AUSTRALIA 1.0 1.0
373 [*,* ,GSL ,1993]: NEWZEALAND AUSTRALIA := 
374 NEWZEALAND 1.0 1.0
375 AUSTRALIA 1.0 1.0
376 [*,* ,GSL ,1994]: NEWZEALAND AUSTRALIA := 
377 NEWZEALAND 1.0 1.0
378 AUSTRALIA 1.0 1.0
379 [*,* ,GSL ,1995]: NEWZEALAND AUSTRALIA := 
380 NEWZEALAND 1.0 1.0
381 AUSTRALIA 1.0 1.0
382 [*,* ,GSL ,1996]: NEWZEALAND AUSTRALIA := 
383 NEWZEALAND 1.0 1.0
384 AUSTRALIA 1.0 1.0
385 [*,* ,GSL ,1997]: NEWZEALAND AUSTRALIA := 
386 NEWZEALAND 1.0 1.0
387 AUSTRALIA 1.0 1.0
388 [*,* ,GSL ,1998]: NEWZEALAND AUSTRALIA := 
389 NEWZEALAND 1.0 1.0
390 AUSTRALIA 1.0 1.0
391 [*,* ,GSL ,1999]: NEWZEALAND AUSTRALIA := 
392 NEWZEALAND 1.0 1.0
393 AUSTRALIA 1.0 1.0
394 [*,* ,GSL ,2000]: NEWZEALAND AUSTRALIA := 
395 NEWZEALAND 1.0 1.0
396 AUSTRALIA 1.0 1.0
397 [*,* ,GSL ,2001]: NEWZEALAND AUSTRALIA := 
398 NEWZEALAND 1.0 1.0
399 AUSTRALIA 1.0 1.0
400 [*,* ,GSL ,2002]: NEWZEALAND AUSTRALIA := 
401 NEWZEALAND 1.0 1.0
```

```
402 AUSTRALIA 1.0 1.0
403 [*,* ,GSL ,2003]: NEWZEALAND AUSTRALIA := 
404 NEWZEALAND 1.0 1.0
405 AUSTRALIA 1.0 1.0
406 [*,* ,GSL ,2004]: NEWZEALAND AUSTRALIA := 
407 NEWZEALAND 1.0 1.0
408 AUSTRALIA 1.0 1.0
409 [*,* ,GSL ,2005]: NEWZEALAND AUSTRALIA := 
410 NEWZEALAND 1.0 1.0
411 AUSTRALIA 1.0 1.0
412 [*,* ,GSL ,2006]: NEWZEALAND AUSTRALIA := 
413 NEWZEALAND 1.0 1.0
414 AUSTRALIA 1.0 1.0
415 [*,* ,GSL ,2007]: NEWZEALAND AUSTRALIA := 
416 NEWZEALAND 1.0 1.0
417 AUSTRALIA 1.0 1.0
418 [*,* ,GSL ,2008]: NEWZEALAND AUSTRALIA := 
419 NEWZEALAND 1.0 1.0
420 AUSTRALIA 1.0 1.0
421 [*,* ,GSL ,2009]: NEWZEALAND AUSTRALIA := 
422 NEWZEALAND 1.0 1.0
423 AUSTRALIA 1.0 1.0
424 [*,* ,GSL ,2010]: NEWZEALAND AUSTRALIA := 
425 NEWZEALAND 1.0 1.0
426 AUSTRALIA 1.0 1.0
427 [*,* ,HCO ,1990]: NEWZEALAND AUSTRALIA := 
428 NEWZEALAND 1.0 1.0
429 AUSTRALIA 1.0 1.0
430 [*,* ,HCO ,1991]: NEWZEALAND AUSTRALIA := 
431 NEWZEALAND 1.0 1.0
432 AUSTRALIA 1.0 1.0
433 [*,* ,HCO ,1992]: NEWZEALAND AUSTRALIA := 
434 NEWZEALAND 1.0 1.0
435 AUSTRALIA 1.0 1.0
436 [*,* ,HCO ,1993]: NEWZEALAND AUSTRALIA := 
437 NEWZEALAND 1.0 1.0
438 AUSTRALIA 1.0 1.0
439 [*,* ,HCO ,1994]: NEWZEALAND AUSTRALIA := 
440 NEWZEALAND 1.0 1.0
441 AUSTRALIA 1.0 1.0
442 [*,* ,HCO ,1995]: NEWZEALAND AUSTRALIA := 
443 NEWZEALAND 1.0 1.0
444 AUSTRALIA 1.0 1.0
445 [*,* ,HCO ,1996]: NEWZEALAND AUSTRALIA := 
446 NEWZEALAND 1.0 1.0
447 AUSTRALIA 1.0 1.0
448 [*,* ,HCO ,1997]: NEWZEALAND AUSTRALIA := 
449 NEWZEALAND 1.0 1.0
450 AUSTRALIA 1.0 1.0
451 [*,* ,HCO ,1998]: NEWZEALAND AUSTRALIA := 
452 NEWZEALAND 1.0 1.0
453 AUSTRALIA 1.0 1.0
454 [*,* ,HCO ,1999]: NEWZEALAND AUSTRALIA := 
455 NEWZEALAND 1.0 1.0
456 AUSTRALIA 1.0 1.0
457 [*,* ,HCO ,2000]: NEWZEALAND AUSTRALIA := 
458 NEWZEALAND 1.0 1.0
459 AUSTRALIA 1.0 1.0
```

```
460 [*,* ,HCO ,2001]: NEWZEALAND AUSTRALIA :=  
461 NEWZEALAND 1.0 1.0  
462 AUSTRALIA 1.0 1.0  
463 [*,* ,HCO ,2002]: NEWZEALAND AUSTRALIA :=  
464 NEWZEALAND 1.0 1.0  
465 AUSTRALIA 1.0 1.0  
466 [*,* ,HCO ,2003]: NEWZEALAND AUSTRALIA :=  
467 NEWZEALAND 1.0 1.0  
468 AUSTRALIA 1.0 1.0  
469 [*,* ,HCO ,2004]: NEWZEALAND AUSTRALIA :=  
470 NEWZEALAND 1.0 1.0  
471 AUSTRALIA 1.0 1.0  
472 [*,* ,HCO ,2005]: NEWZEALAND AUSTRALIA :=  
473 NEWZEALAND 1.0 1.0  
474 AUSTRALIA 1.0 1.0  
475 [*,* ,HCO ,2006]: NEWZEALAND AUSTRALIA :=  
476 NEWZEALAND 1.0 1.0  
477 AUSTRALIA 1.0 1.0  
478 [*,* ,HCO ,2007]: NEWZEALAND AUSTRALIA :=  
479 NEWZEALAND 1.0 1.0  
480 AUSTRALIA 1.0 1.0  
481 [*,* ,HCO ,2008]: NEWZEALAND AUSTRALIA :=  
482 NEWZEALAND 1.0 1.0  
483 AUSTRALIA 1.0 1.0  
484 [*,* ,HCO ,2009]: NEWZEALAND AUSTRALIA :=  
485 NEWZEALAND 1.0 1.0  
486 AUSTRALIA 1.0 1.0  
487 [*,* ,HCO ,2010]: NEWZEALAND AUSTRALIA :=  
488 NEWZEALAND 1.0 1.0  
489 AUSTRALIA 1.0 1.0  
490 [*,* ,HYD ,1990]: NEWZEALAND AUSTRALIA :=  
491 NEWZEALAND 1.0 1.0  
492 AUSTRALIA 1.0 1.0  
493 [*,* ,HYD ,1991]: NEWZEALAND AUSTRALIA :=  
494 NEWZEALAND 1.0 1.0  
495 AUSTRALIA 1.0 1.0  
496 [*,* ,HYD ,1992]: NEWZEALAND AUSTRALIA :=  
497 NEWZEALAND 1.0 1.0  
498 AUSTRALIA 1.0 1.0  
499 [*,* ,HYD ,1993]: NEWZEALAND AUSTRALIA :=  
500 NEWZEALAND 1.0 1.0  
501 AUSTRALIA 1.0 1.0  
502 [*,* ,HYD ,1994]: NEWZEALAND AUSTRALIA :=  
503 NEWZEALAND 1.0 1.0  
504 AUSTRALIA 1.0 1.0  
505 [*,* ,HYD ,1995]: NEWZEALAND AUSTRALIA :=  
506 NEWZEALAND 1.0 1.0  
507 AUSTRALIA 1.0 1.0  
508 [*,* ,HYD ,1996]: NEWZEALAND AUSTRALIA :=  
509 NEWZEALAND 1.0 1.0  
510 AUSTRALIA 1.0 1.0  
511 [*,* ,HYD ,1997]: NEWZEALAND AUSTRALIA :=  
512 NEWZEALAND 1.0 1.0  
513 AUSTRALIA 1.0 1.0  
514 [*,* ,HYD ,1998]: NEWZEALAND AUSTRALIA :=  
515 NEWZEALAND 1.0 1.0  
516 AUSTRALIA 1.0 1.0  
517 [*,* ,HYD ,1999]: NEWZEALAND AUSTRALIA :=
```

```
518 NEWZEALAND 1.0 1.0
519 AUSTRALIA 1.0 1.0
520 [*,* ,HYD ,2000]: NEWZEALAND AUSTRALIA := 
521 NEWZEALAND 1.0 1.0
522 AUSTRALIA 1.0 1.0
523 [*,* ,HYD ,2001]: NEWZEALAND AUSTRALIA := 
524 NEWZEALAND 1.0 1.0
525 AUSTRALIA 1.0 1.0
526 [*,* ,HYD ,2002]: NEWZEALAND AUSTRALIA := 
527 NEWZEALAND 1.0 1.0
528 AUSTRALIA 1.0 1.0
529 [*,* ,HYD ,2003]: NEWZEALAND AUSTRALIA := 
530 NEWZEALAND 1.0 1.0
531 AUSTRALIA 1.0 1.0
532 [*,* ,HYD ,2004]: NEWZEALAND AUSTRALIA := 
533 NEWZEALAND 1.0 1.0
534 AUSTRALIA 1.0 1.0
535 [*,* ,HYD ,2005]: NEWZEALAND AUSTRALIA := 
536 NEWZEALAND 1.0 1.0
537 AUSTRALIA 1.0 1.0
538 [*,* ,HYD ,2006]: NEWZEALAND AUSTRALIA := 
539 NEWZEALAND 1.0 1.0
540 AUSTRALIA 1.0 1.0
541 [*,* ,HYD ,2007]: NEWZEALAND AUSTRALIA := 
542 NEWZEALAND 1.0 1.0
543 AUSTRALIA 1.0 1.0
544 [*,* ,HYD ,2008]: NEWZEALAND AUSTRALIA := 
545 NEWZEALAND 1.0 1.0
546 AUSTRALIA 1.0 1.0
547 [*,* ,HYD ,2009]: NEWZEALAND AUSTRALIA := 
548 NEWZEALAND 1.0 1.0
549 AUSTRALIA 1.0 1.0
550 [*,* ,HYD ,2010]: NEWZEALAND AUSTRALIA := 
551 NEWZEALAND 1.0 1.0
552 AUSTRALIA 1.0 1.0
553 [*,* ,LTH ,1990]: NEWZEALAND AUSTRALIA := 
554 NEWZEALAND 1.0 1.0
555 AUSTRALIA 1.0 1.0
556 [*,* ,LTH ,1991]: NEWZEALAND AUSTRALIA := 
557 NEWZEALAND 1.0 1.0
558 AUSTRALIA 1.0 1.0
559 [*,* ,LTH ,1992]: NEWZEALAND AUSTRALIA := 
560 NEWZEALAND 1.0 1.0
561 AUSTRALIA 1.0 1.0
562 [*,* ,LTH ,1993]: NEWZEALAND AUSTRALIA := 
563 NEWZEALAND 1.0 1.0
564 AUSTRALIA 1.0 1.0
565 [*,* ,LTH ,1994]: NEWZEALAND AUSTRALIA := 
566 NEWZEALAND 1.0 1.0
567 AUSTRALIA 1.0 1.0
568 [*,* ,LTH ,1995]: NEWZEALAND AUSTRALIA := 
569 NEWZEALAND 1.0 1.0
570 AUSTRALIA 1.0 1.0
571 [*,* ,LTH ,1996]: NEWZEALAND AUSTRALIA := 
572 NEWZEALAND 1.0 1.0
573 AUSTRALIA 1.0 1.0
574 [*,* ,LTH ,1997]: NEWZEALAND AUSTRALIA := 
575 NEWZEALAND 1.0 1.0
```

```
576 AUSTRALIA 1.0 1.0
577 [*,* ,LTH ,1998]: NEWZEALAND AUSTRALIA := 
578 NEWZEALAND 1.0 1.0
579 AUSTRALIA 1.0 1.0
580 [*,* ,LTH ,1999]: NEWZEALAND AUSTRALIA := 
581 NEWZEALAND 1.0 1.0
582 AUSTRALIA 1.0 1.0
583 [*,* ,LTH ,2000]: NEWZEALAND AUSTRALIA := 
584 NEWZEALAND 1.0 1.0
585 AUSTRALIA 1.0 1.0
586 [*,* ,LTH ,2001]: NEWZEALAND AUSTRALIA := 
587 NEWZEALAND 1.0 1.0
588 AUSTRALIA 1.0 1.0
589 [*,* ,LTH ,2002]: NEWZEALAND AUSTRALIA := 
590 NEWZEALAND 1.0 1.0
591 AUSTRALIA 1.0 1.0
592 [*,* ,LTH ,2003]: NEWZEALAND AUSTRALIA := 
593 NEWZEALAND 1.0 1.0
594 AUSTRALIA 1.0 1.0
595 [*,* ,LTH ,2004]: NEWZEALAND AUSTRALIA := 
596 NEWZEALAND 1.0 1.0
597 AUSTRALIA 1.0 1.0
598 [*,* ,LTH ,2005]: NEWZEALAND AUSTRALIA := 
599 NEWZEALAND 1.0 1.0
600 AUSTRALIA 1.0 1.0
601 [*,* ,LTH ,2006]: NEWZEALAND AUSTRALIA := 
602 NEWZEALAND 1.0 1.0
603 AUSTRALIA 1.0 1.0
604 [*,* ,LTH ,2007]: NEWZEALAND AUSTRALIA := 
605 NEWZEALAND 1.0 1.0
606 AUSTRALIA 1.0 1.0
607 [*,* ,LTH ,2008]: NEWZEALAND AUSTRALIA := 
608 NEWZEALAND 1.0 1.0
609 AUSTRALIA 1.0 1.0
610 [*,* ,LTH ,2009]: NEWZEALAND AUSTRALIA := 
611 NEWZEALAND 1.0 1.0
612 AUSTRALIA 1.0 1.0
613 [*,* ,LTH ,2010]: NEWZEALAND AUSTRALIA := 
614 NEWZEALAND 1.0 1.0
615 AUSTRALIA 1.0 1.0
616 [*,* ,OIL ,1990]: NEWZEALAND AUSTRALIA := 
617 NEWZEALAND 1.0 1.0
618 AUSTRALIA 1.0 1.0
619 [*,* ,OIL ,1991]: NEWZEALAND AUSTRALIA := 
620 NEWZEALAND 1.0 1.0
621 AUSTRALIA 1.0 1.0
622 [*,* ,OIL ,1992]: NEWZEALAND AUSTRALIA := 
623 NEWZEALAND 1.0 1.0
624 AUSTRALIA 1.0 1.0
625 [*,* ,OIL ,1993]: NEWZEALAND AUSTRALIA := 
626 NEWZEALAND 1.0 1.0
627 AUSTRALIA 1.0 1.0
628 [*,* ,OIL ,1994]: NEWZEALAND AUSTRALIA := 
629 NEWZEALAND 1.0 1.0
630 AUSTRALIA 1.0 1.0
631 [*,* ,OIL ,1995]: NEWZEALAND AUSTRALIA := 
632 NEWZEALAND 1.0 1.0
633 AUSTRALIA 1.0 1.0
```

```
634 [*,* ,OIL ,1996]: NEWZEALAND AUSTRALIA :=  
635 NEWZEALAND 1.0 1.0  
636 AUSTRALIA 1.0 1.0  
637 [*,* ,OIL ,1997]: NEWZEALAND AUSTRALIA :=  
638 NEWZEALAND 1.0 1.0  
639 AUSTRALIA 1.0 1.0  
640 [*,* ,OIL ,1998]: NEWZEALAND AUSTRALIA :=  
641 NEWZEALAND 1.0 1.0  
642 AUSTRALIA 1.0 1.0  
643 [*,* ,OIL ,1999]: NEWZEALAND AUSTRALIA :=  
644 NEWZEALAND 1.0 1.0  
645 AUSTRALIA 1.0 1.0  
646 [*,* ,OIL ,2000]: NEWZEALAND AUSTRALIA :=  
647 NEWZEALAND 1.0 1.0  
648 AUSTRALIA 1.0 1.0  
649 [*,* ,OIL ,2001]: NEWZEALAND AUSTRALIA :=  
650 NEWZEALAND 1.0 1.0  
651 AUSTRALIA 1.0 1.0  
652 [*,* ,OIL ,2002]: NEWZEALAND AUSTRALIA :=  
653 NEWZEALAND 1.0 1.0  
654 AUSTRALIA 1.0 1.0  
655 [*,* ,OIL ,2003]: NEWZEALAND AUSTRALIA :=  
656 NEWZEALAND 1.0 1.0  
657 AUSTRALIA 1.0 1.0  
658 [*,* ,OIL ,2004]: NEWZEALAND AUSTRALIA :=  
659 NEWZEALAND 1.0 1.0  
660 AUSTRALIA 1.0 1.0  
661 [*,* ,OIL ,2005]: NEWZEALAND AUSTRALIA :=  
662 NEWZEALAND 1.0 1.0  
663 AUSTRALIA 1.0 1.0  
664 [*,* ,OIL ,2006]: NEWZEALAND AUSTRALIA :=  
665 NEWZEALAND 1.0 1.0  
666 AUSTRALIA 1.0 1.0  
667 [*,* ,OIL ,2007]: NEWZEALAND AUSTRALIA :=  
668 NEWZEALAND 1.0 1.0  
669 AUSTRALIA 1.0 1.0  
670 [*,* ,OIL ,2008]: NEWZEALAND AUSTRALIA :=  
671 NEWZEALAND 1.0 1.0  
672 AUSTRALIA 1.0 1.0  
673 [*,* ,OIL ,2009]: NEWZEALAND AUSTRALIA :=  
674 NEWZEALAND 1.0 1.0  
675 AUSTRALIA 1.0 1.0  
676 [*,* ,OIL ,2010]: NEWZEALAND AUSTRALIA :=  
677 NEWZEALAND 1.0 1.0  
678 AUSTRALIA 1.0 1.0  
679 [*,* ,URN ,1990]: NEWZEALAND AUSTRALIA :=  
680 NEWZEALAND 1.0 1.0  
681 AUSTRALIA 1.0 1.0  
682 [*,* ,URN ,1991]: NEWZEALAND AUSTRALIA :=  
683 NEWZEALAND 1.0 1.0  
684 AUSTRALIA 1.0 1.0  
685 [*,* ,URN ,1992]: NEWZEALAND AUSTRALIA :=  
686 NEWZEALAND 1.0 1.0  
687 AUSTRALIA 1.0 1.0  
688 [*,* ,URN ,1993]: NEWZEALAND AUSTRALIA :=  
689 NEWZEALAND 1.0 1.0  
690 AUSTRALIA 1.0 1.0  
691 [*,* ,URN ,1994]: NEWZEALAND AUSTRALIA :=
```

```
692 NEWZEALAND 1.0 1.0
693 AUSTRALIA 1.0 1.0
694 [*,* ,URN ,1995]: NEWZEALAND AUSTRALIA := 
695 NEWZEALAND 1.0 1.0
696 AUSTRALIA 1.0 1.0
697 [*,* ,URN ,1996]: NEWZEALAND AUSTRALIA := 
698 NEWZEALAND 1.0 1.0
699 AUSTRALIA 1.0 1.0
700 [*,* ,URN ,1997]: NEWZEALAND AUSTRALIA := 
701 NEWZEALAND 1.0 1.0
702 AUSTRALIA 1.0 1.0
703 [*,* ,URN ,1998]: NEWZEALAND AUSTRALIA := 
704 NEWZEALAND 1.0 1.0
705 AUSTRALIA 1.0 1.0
706 [*,* ,URN ,1999]: NEWZEALAND AUSTRALIA := 
707 NEWZEALAND 1.0 1.0
708 AUSTRALIA 1.0 1.0
709 [*,* ,URN ,2000]: NEWZEALAND AUSTRALIA := 
710 NEWZEALAND 1.0 1.0
711 AUSTRALIA 1.0 1.0
712 [*,* ,URN ,2001]: NEWZEALAND AUSTRALIA := 
713 NEWZEALAND 1.0 1.0
714 AUSTRALIA 1.0 1.0
715 [*,* ,URN ,2002]: NEWZEALAND AUSTRALIA := 
716 NEWZEALAND 1.0 1.0
717 AUSTRALIA 1.0 1.0
718 [*,* ,URN ,2003]: NEWZEALAND AUSTRALIA := 
719 NEWZEALAND 1.0 1.0
720 AUSTRALIA 1.0 1.0
721 [*,* ,URN ,2004]: NEWZEALAND AUSTRALIA := 
722 NEWZEALAND 1.0 1.0
723 AUSTRALIA 1.0 1.0
724 [*,* ,URN ,2005]: NEWZEALAND AUSTRALIA := 
725 NEWZEALAND 1.0 1.0
726 AUSTRALIA 1.0 1.0
727 [*,* ,URN ,2006]: NEWZEALAND AUSTRALIA := 
728 NEWZEALAND 1.0 1.0
729 AUSTRALIA 1.0 1.0
730 [*,* ,URN ,2007]: NEWZEALAND AUSTRALIA := 
731 NEWZEALAND 1.0 1.0
732 AUSTRALIA 1.0 1.0
733 [*,* ,URN ,2008]: NEWZEALAND AUSTRALIA := 
734 NEWZEALAND 1.0 1.0
735 AUSTRALIA 1.0 1.0
736 [*,* ,URN ,2009]: NEWZEALAND AUSTRALIA := 
737 NEWZEALAND 1.0 1.0
738 AUSTRALIA 1.0 1.0
739 [*,* ,URN ,2010]: NEWZEALAND AUSTRALIA := 
740 NEWZEALAND 1.0 1.0
741 AUSTRALIA 1.0 1.0
742 [*,* ,RH ,1990]: NEWZEALAND AUSTRALIA := 
743 NEWZEALAND 1.0 1.0
744 AUSTRALIA 1.0 1.0
745 [*,* ,RH ,1991]: NEWZEALAND AUSTRALIA := 
746 NEWZEALAND 1.0 1.0
747 AUSTRALIA 1.0 1.0
748 [*,* ,RH ,1992]: NEWZEALAND AUSTRALIA := 
749 NEWZEALAND 1.0 1.0
```

```
750 AUSTRALIA 1.0 1.0
751 [*,* ,RH ,1993]: NEWZEALAND AUSTRALIA := 
752 NEWZEALAND 1.0 1.0
753 AUSTRALIA 1.0 1.0
754 [*,* ,RH ,1994]: NEWZEALAND AUSTRALIA := 
755 NEWZEALAND 1.0 1.0
756 AUSTRALIA 1.0 1.0
757 [*,* ,RH ,1995]: NEWZEALAND AUSTRALIA := 
758 NEWZEALAND 1.0 1.0
759 AUSTRALIA 1.0 1.0
760 [*,* ,RH ,1996]: NEWZEALAND AUSTRALIA := 
761 NEWZEALAND 1.0 1.0
762 AUSTRALIA 1.0 1.0
763 [*,* ,RH ,1997]: NEWZEALAND AUSTRALIA := 
764 NEWZEALAND 1.0 1.0
765 AUSTRALIA 1.0 1.0
766 [*,* ,RH ,1998]: NEWZEALAND AUSTRALIA := 
767 NEWZEALAND 1.0 1.0
768 AUSTRALIA 1.0 1.0
769 [*,* ,RH ,1999]: NEWZEALAND AUSTRALIA := 
770 NEWZEALAND 1.0 1.0
771 AUSTRALIA 1.0 1.0
772 [*,* ,RH ,2000]: NEWZEALAND AUSTRALIA := 
773 NEWZEALAND 1.0 1.0
774 AUSTRALIA 1.0 1.0
775 [*,* ,RH ,2001]: NEWZEALAND AUSTRALIA := 
776 NEWZEALAND 1.0 1.0
777 AUSTRALIA 1.0 1.0
778 [*,* ,RH ,2002]: NEWZEALAND AUSTRALIA := 
779 NEWZEALAND 1.0 1.0
780 AUSTRALIA 1.0 1.0
781 [*,* ,RH ,2003]: NEWZEALAND AUSTRALIA := 
782 NEWZEALAND 1.0 1.0
783 AUSTRALIA 1.0 1.0
784 [*,* ,RH ,2004]: NEWZEALAND AUSTRALIA := 
785 NEWZEALAND 1.0 1.0
786 AUSTRALIA 1.0 1.0
787 [*,* ,RH ,2005]: NEWZEALAND AUSTRALIA := 
788 NEWZEALAND 1.0 1.0
789 AUSTRALIA 1.0 1.0
790 [*,* ,RH ,2006]: NEWZEALAND AUSTRALIA := 
791 NEWZEALAND 1.0 1.0
792 AUSTRALIA 1.0 1.0
793 [*,* ,RH ,2007]: NEWZEALAND AUSTRALIA := 
794 NEWZEALAND 1.0 1.0
795 AUSTRALIA 1.0 1.0
796 [*,* ,RH ,2008]: NEWZEALAND AUSTRALIA := 
797 NEWZEALAND 1.0 1.0
798 AUSTRALIA 1.0 1.0
799 [*,* ,RH ,2009]: NEWZEALAND AUSTRALIA := 
800 NEWZEALAND 1.0 1.0
801 AUSTRALIA 1.0 1.0
802 [*,* ,RH ,2010]: NEWZEALAND AUSTRALIA := 
803 NEWZEALAND 1.0 1.0
804 AUSTRALIA 1.0 1.0
805 [*,* ,RL ,1990]: NEWZEALAND AUSTRALIA := 
806 NEWZEALAND 1.0 1.0
807 AUSTRALIA 1.0 1.0
```

```
808 [*,* ,RL ,1991]: NEWZEALAND AUSTRALIA :=  
809 NEWZEALAND 1.0 1.0  
810 AUSTRALIA 1.0 1.0  
811 [*,* ,RL ,1992]: NEWZEALAND AUSTRALIA :=  
812 NEWZEALAND 1.0 1.0  
813 AUSTRALIA 1.0 1.0  
814 [*,* ,RL ,1993]: NEWZEALAND AUSTRALIA :=  
815 NEWZEALAND 1.0 1.0  
816 AUSTRALIA 1.0 1.0  
817 [*,* ,RL ,1994]: NEWZEALAND AUSTRALIA :=  
818 NEWZEALAND 1.0 1.0  
819 AUSTRALIA 1.0 1.0  
820 [*,* ,RL ,1995]: NEWZEALAND AUSTRALIA :=  
821 NEWZEALAND 1.0 1.0  
822 AUSTRALIA 1.0 1.0  
823 [*,* ,RL ,1996]: NEWZEALAND AUSTRALIA :=  
824 NEWZEALAND 1.0 1.0  
825 AUSTRALIA 1.0 1.0  
826 [*,* ,RL ,1997]: NEWZEALAND AUSTRALIA :=  
827 NEWZEALAND 1.0 1.0  
828 AUSTRALIA 1.0 1.0  
829 [*,* ,RL ,1998]: NEWZEALAND AUSTRALIA :=  
830 NEWZEALAND 1.0 1.0  
831 AUSTRALIA 1.0 1.0  
832 [*,* ,RL ,1999]: NEWZEALAND AUSTRALIA :=  
833 NEWZEALAND 1.0 1.0  
834 AUSTRALIA 1.0 1.0  
835 [*,* ,RL ,2000]: NEWZEALAND AUSTRALIA :=  
836 NEWZEALAND 1.0 1.0  
837 AUSTRALIA 1.0 1.0  
838 [*,* ,RL ,2001]: NEWZEALAND AUSTRALIA :=  
839 NEWZEALAND 1.0 1.0  
840 AUSTRALIA 1.0 1.0  
841 [*,* ,RL ,2002]: NEWZEALAND AUSTRALIA :=  
842 NEWZEALAND 1.0 1.0  
843 AUSTRALIA 1.0 1.0  
844 [*,* ,RL ,2003]: NEWZEALAND AUSTRALIA :=  
845 NEWZEALAND 1.0 1.0  
846 AUSTRALIA 1.0 1.0  
847 [*,* ,RL ,2004]: NEWZEALAND AUSTRALIA :=  
848 NEWZEALAND 1.0 1.0  
849 AUSTRALIA 1.0 1.0  
850 [*,* ,RL ,2005]: NEWZEALAND AUSTRALIA :=  
851 NEWZEALAND 1.0 1.0  
852 AUSTRALIA 1.0 1.0  
853 [*,* ,RL ,2006]: NEWZEALAND AUSTRALIA :=  
854 NEWZEALAND 1.0 1.0  
855 AUSTRALIA 1.0 1.0  
856 [*,* ,RL ,2007]: NEWZEALAND AUSTRALIA :=  
857 NEWZEALAND 1.0 1.0  
858 AUSTRALIA 1.0 1.0  
859 [*,* ,RL ,2008]: NEWZEALAND AUSTRALIA :=  
860 NEWZEALAND 1.0 1.0  
861 AUSTRALIA 1.0 1.0  
862 [*,* ,RL ,2009]: NEWZEALAND AUSTRALIA :=  
863 NEWZEALAND 1.0 1.0  
864 AUSTRALIA 1.0 1.0  
865 [*,* ,RL ,2010]: NEWZEALAND AUSTRALIA :=
```

```
866 NEWZEALAND 1.0 1.0
867 AUSTRALIA 1.0 1.0
868 [*,* ,TX ,1990]: NEWZEALAND AUSTRALIA := 
869 NEWZEALAND 1.0 1.0
870 AUSTRALIA 1.0 1.0
871 [*,* ,TX ,1991]: NEWZEALAND AUSTRALIA := 
872 NEWZEALAND 1.0 1.0
873 AUSTRALIA 1.0 1.0
874 [*,* ,TX ,1992]: NEWZEALAND AUSTRALIA := 
875 NEWZEALAND 1.0 1.0
876 AUSTRALIA 1.0 1.0
877 [*,* ,TX ,1993]: NEWZEALAND AUSTRALIA := 
878 NEWZEALAND 1.0 1.0
879 AUSTRALIA 1.0 1.0
880 [*,* ,TX ,1994]: NEWZEALAND AUSTRALIA := 
881 NEWZEALAND 1.0 1.0
882 AUSTRALIA 1.0 1.0
883 [*,* ,TX ,1995]: NEWZEALAND AUSTRALIA := 
884 NEWZEALAND 1.0 1.0
885 AUSTRALIA 1.0 1.0
886 [*,* ,TX ,1996]: NEWZEALAND AUSTRALIA := 
887 NEWZEALAND 1.0 1.0
888 AUSTRALIA 1.0 1.0
889 [*,* ,TX ,1997]: NEWZEALAND AUSTRALIA := 
890 NEWZEALAND 1.0 1.0
891 AUSTRALIA 1.0 1.0
892 [*,* ,TX ,1998]: NEWZEALAND AUSTRALIA := 
893 NEWZEALAND 1.0 1.0
894 AUSTRALIA 1.0 1.0
895 [*,* ,TX ,1999]: NEWZEALAND AUSTRALIA := 
896 NEWZEALAND 1.0 1.0
897 AUSTRALIA 1.0 1.0
898 [*,* ,TX ,2000]: NEWZEALAND AUSTRALIA := 
899 NEWZEALAND 1.0 1.0
900 AUSTRALIA 1.0 1.0
901 [*,* ,TX ,2001]: NEWZEALAND AUSTRALIA := 
902 NEWZEALAND 1.0 1.0
903 AUSTRALIA 1.0 1.0
904 [*,* ,TX ,2002]: NEWZEALAND AUSTRALIA := 
905 NEWZEALAND 1.0 1.0
906 AUSTRALIA 1.0 1.0
907 [*,* ,TX ,2003]: NEWZEALAND AUSTRALIA := 
908 NEWZEALAND 1.0 1.0
909 AUSTRALIA 1.0 1.0
910 [*,* ,TX ,2004]: NEWZEALAND AUSTRALIA := 
911 NEWZEALAND 1.0 1.0
912 AUSTRALIA 1.0 1.0
913 [*,* ,TX ,2005]: NEWZEALAND AUSTRALIA := 
914 NEWZEALAND 1.0 1.0
915 AUSTRALIA 1.0 1.0
916 [*,* ,TX ,2006]: NEWZEALAND AUSTRALIA := 
917 NEWZEALAND 1.0 1.0
918 AUSTRALIA 1.0 1.0
919 [*,* ,TX ,2007]: NEWZEALAND AUSTRALIA := 
920 NEWZEALAND 1.0 1.0
921 AUSTRALIA 1.0 1.0
922 [*,* ,TX ,2008]: NEWZEALAND AUSTRALIA := 
923 NEWZEALAND 1.0 1.0
```

```

924 AUSTRALIA 1.0 1.0
925 [*,* ,TX,2009]: NEWZEALAND AUSTRALIA := 
926 NEWZEALAND 1.0 1.0
927 AUSTRALIA 1.0 1.0
928 [*,* ,TX,2010]: NEWZEALAND AUSTRALIA := 
929 NEWZEALAND 1.0 1.0
930 AUSTRALIA 1.0 1.0
931 ;
932 #
933 param DepreciationMethod default 1:=
934 NEWZEALAND 1.0
935 AUSTRALIA 1.0
936 ;
937 #
938 param SpecifiedAnnualDemand default 1:=
939 [*,* ,1990]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX := 
940 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
941 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
942 [*,* ,1991]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX := 
943 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
944 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
945 [*,* ,1992]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX := 
946 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
947 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
948 [*,* ,1993]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX := 
949 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
950 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
951 [*,* ,1994]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX := 
952 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
953 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
954 [*,* ,1995]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX := 
955 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
956 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
957 [*,* ,1996]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX := 
958 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
959 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
960 [*,* ,1997]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX := 
961 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
962 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
963 [*,* ,1998]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX := 
964 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
965 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
966 [*,* ,1999]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX := 
967 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
968 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
969 [*,* ,2000]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX := 
970 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
971 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
972 [*,* ,2001]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX := 
973 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
974 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
975 [*,* ,2002]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX := 
976 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
977 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
978 [*,* ,2003]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX := 
979 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
980 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
981 [*,* ,2004]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX := 

```

```

982 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
983 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
984 [*,* ,2005]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX := 
985 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
986 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
987 [*,* ,2006]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX := 
988 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
989 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
990 [*,* ,2007]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX := 
991 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
992 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
993 [*,* ,2008]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX := 
994 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
995 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
996 [*,* ,2009]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX := 
997 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
998 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
999 [*,* ,2010]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX := 
1000 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1001 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1002 ;
1003 #
1004 param SpecifiedDemandProfile default 1:=
1005 [*,* ,INTERMEDIATE_DAY ,1990]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH
    RL TX := 
1006 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1007 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1008 [*,* ,INTERMEDIATE_DAY ,1991]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH
    RL TX := 
1009 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1010 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1011 [*,* ,INTERMEDIATE_DAY ,1992]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH
    RL TX := 
1012 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1013 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1014 [*,* ,INTERMEDIATE_DAY ,1993]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH
    RL TX := 
1015 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1016 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1017 [*,* ,INTERMEDIATE_DAY ,1994]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH
    RL TX := 
1018 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1019 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1020 [*,* ,INTERMEDIATE_DAY ,1995]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH
    RL TX := 
1021 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1022 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1023 [*,* ,INTERMEDIATE_DAY ,1996]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH
    RL TX := 
1024 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1025 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1026 [*,* ,INTERMEDIATE_DAY ,1997]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH
    RL TX := 
1027 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1028 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1029 [*,* ,INTERMEDIATE_DAY ,1998]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH
    RL TX := 
1030 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0

```

```

1031 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1032 [*,* ,INTERMEDIATE_DAY ,1999]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH
      RL TX :=
1033 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1034 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1035 [*,* ,INTERMEDIATE_DAY ,2000]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH
      RL TX :=
1036 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1037 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1038 [*,* ,INTERMEDIATE_DAY ,2001]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH
      RL TX :=
1039 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1040 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1041 [*,* ,INTERMEDIATE_DAY ,2002]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH
      RL TX :=
1042 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1043 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1044 [*,* ,INTERMEDIATE_DAY ,2003]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH
      RL TX :=
1045 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1046 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1047 [*,* ,INTERMEDIATE_DAY ,2004]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH
      RL TX :=
1048 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1049 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1050 [*,* ,INTERMEDIATE_DAY ,2005]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH
      RL TX :=
1051 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1052 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1053 [*,* ,INTERMEDIATE_DAY ,2006]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH
      RL TX :=
1054 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1055 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1056 [*,* ,INTERMEDIATE_DAY ,2007]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH
      RL TX :=
1057 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1058 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1059 [*,* ,INTERMEDIATE_DAY ,2008]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH
      RL TX :=
1060 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1061 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1062 [*,* ,INTERMEDIATE_DAY ,2009]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH
      RL TX :=
1063 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1064 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1065 [*,* ,INTERMEDIATE_DAY ,2010]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH
      RL TX :=
1066 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1067 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1068 [*,* ,INTERMEDIATE_NIGHT ,1990]: CSV DSL ELC GSL HCO HYD LTH OIL URN
      RH RL TX :=
1069 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1070 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1071 [*,* ,INTERMEDIATE_NIGHT ,1991]: CSV DSL ELC GSL HCO HYD LTH OIL URN
      RH RL TX :=
1072 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1073 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0

```

```

1074 [*,* ,INTERMEDIATE_NIGHT ,1992]: CSV DSL ELC GSL HCO HYD LTH OIL URN
    RH RL TX :=
1075 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1076 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1077 [*,* ,INTERMEDIATE_NIGHT ,1993]: CSV DSL ELC GSL HCO HYD LTH OIL URN
    RH RL TX :=
1078 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1079 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1080 [*,* ,INTERMEDIATE_NIGHT ,1994]: CSV DSL ELC GSL HCO HYD LTH OIL URN
    RH RL TX :=
1081 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1082 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1083 [*,* ,INTERMEDIATE_NIGHT ,1995]: CSV DSL ELC GSL HCO HYD LTH OIL URN
    RH RL TX :=
1084 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1085 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1086 [*,* ,INTERMEDIATE_NIGHT ,1996]: CSV DSL ELC GSL HCO HYD LTH OIL URN
    RH RL TX :=
1087 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1088 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1089 [*,* ,INTERMEDIATE_NIGHT ,1997]: CSV DSL ELC GSL HCO HYD LTH OIL URN
    RH RL TX :=
1090 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1091 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1092 [*,* ,INTERMEDIATE_NIGHT ,1998]: CSV DSL ELC GSL HCO HYD LTH OIL URN
    RH RL TX :=
1093 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1094 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1095 [*,* ,INTERMEDIATE_NIGHT ,1999]: CSV DSL ELC GSL HCO HYD LTH OIL URN
    RH RL TX :=
1096 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1097 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1098 [*,* ,INTERMEDIATE_NIGHT ,2000]: CSV DSL ELC GSL HCO HYD LTH OIL URN
    RH RL TX :=
1099 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1100 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1101 [*,* ,INTERMEDIATE_NIGHT ,2001]: CSV DSL ELC GSL HCO HYD LTH OIL URN
    RH RL TX :=
1102 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1103 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1104 [*,* ,INTERMEDIATE_NIGHT ,2002]: CSV DSL ELC GSL HCO HYD LTH OIL URN
    RH RL TX :=
1105 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1106 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1107 [*,* ,INTERMEDIATE_NIGHT ,2003]: CSV DSL ELC GSL HCO HYD LTH OIL URN
    RH RL TX :=
1108 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1109 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1110 [*,* ,INTERMEDIATE_NIGHT ,2004]: CSV DSL ELC GSL HCO HYD LTH OIL URN
    RH RL TX :=
1111 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1112 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1113 [*,* ,INTERMEDIATE_NIGHT ,2005]: CSV DSL ELC GSL HCO HYD LTH OIL URN
    RH RL TX :=
1114 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1115 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1116 [*,* ,INTERMEDIATE_NIGHT ,2006]: CSV DSL ELC GSL HCO HYD LTH OIL URN
    RH RL TX :=

```

```

1117 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1118 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1119 [*,* ,INTERMEDIATE_NIGHT ,2007]: CSV DSL ELC GSL HCO HYD LTH OIL URN
    RH RL TX :=
1120 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1121 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1122 [*,* ,INTERMEDIATE_NIGHT ,2008]: CSV DSL ELC GSL HCO HYD LTH OIL URN
    RH RL TX :=
1123 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1124 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1125 [*,* ,INTERMEDIATE_NIGHT ,2009]: CSV DSL ELC GSL HCO HYD LTH OIL URN
    RH RL TX :=
1126 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1127 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1128 [*,* ,INTERMEDIATE_NIGHT ,2010]: CSV DSL ELC GSL HCO HYD LTH OIL URN
    RH RL TX :=
1129 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1130 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1131 [*,* ,SUMMER_DAY ,1990]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
    :=
1132 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1133 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1134 [*,* ,SUMMER_DAY ,1991]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
    :=
1135 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1136 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1137 [*,* ,SUMMER_DAY ,1992]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
    :=
1138 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1139 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1140 [*,* ,SUMMER_DAY ,1993]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
    :=
1141 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1142 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1143 [*,* ,SUMMER_DAY ,1994]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
    :=
1144 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1145 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1146 [*,* ,SUMMER_DAY ,1995]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
    :=
1147 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1148 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1149 [*,* ,SUMMER_DAY ,1996]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
    :=
1150 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1151 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1152 [*,* ,SUMMER_DAY ,1997]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
    :=
1153 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1154 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1155 [*,* ,SUMMER_DAY ,1998]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
    :=
1156 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1157 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1158 [*,* ,SUMMER_DAY ,1999]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
    :=
1159 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1160 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0

```

```

1161 [*,* ,SUMMER_DAY ,2000]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
1162      :=  

1163 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

1164 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

1165 [*,* ,SUMMER_DAY ,2001]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
1166      :=  

1167 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

1168 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

1169 [*,* ,SUMMER_DAY ,2002]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
1170      :=  

1171 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

1172 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

1173 [*,* ,SUMMER_DAY ,2003]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
1174      :=  

1175 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

1176 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

1177 [*,* ,SUMMER_DAY ,2004]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
1178      :=  

1179 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

1180 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

1181 [*,* ,SUMMER_DAY ,2005]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
1182      :=  

1183 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

1184 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

1185 [*,* ,SUMMER_DAY ,2006]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
1186      :=  

1187 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

1188 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

1189 [*,* ,SUMMER_DAY ,2007]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
1190      :=  

1191 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

1192 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

1193 [*,* ,SUMMER_DAY ,2008]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
1194      :=  

1195 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

1196 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

1197 [*,* ,SUMMER_NIGHT ,1990]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL
1198      TX :=  

1199 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

1200 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

1201 [*,* ,SUMMER_NIGHT ,1991]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL
1202      TX :=  

1203 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

1204 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

1205 [*,* ,SUMMER_NIGHT ,1992]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL
1206      TX :=  

1207 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

1208 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

1209 [*,* ,SUMMER_NIGHT ,1993]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL
1210      TX :=

```



```

1248 [*,* ,SUMMER_NIGHT ,2008]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL
1249 TX :=
1250 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1251 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1252 [*,* ,SUMMER_NIGHT ,2009]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL
1253 TX :=
1254 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1255 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1256 [*,* ,SUMMER_NIGHT ,2010]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL
1257 TX :=
1258 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1259 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1260 [*,* ,WINTER_DAY ,1990]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
1261 :=

1262 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1263 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1264 [*,* ,WINTER_DAY ,1991]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
1265 :=

1266 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1267 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1268 [*,* ,WINTER_DAY ,1992]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
1269 :=

1270 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1271 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1272 [*,* ,WINTER_DAY ,1993]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
1273 :=

1274 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1275 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1276 [*,* ,WINTER_DAY ,1994]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
1277 :=

1278 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1279 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1280 [*,* ,WINTER_DAY ,1995]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
1281 :=

1282 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1283 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1284 [*,* ,WINTER_DAY ,1996]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
1285 :=

1286 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1287 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1288 [*,* ,WINTER_DAY ,1997]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
1289 :=

1290 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1291 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1292 [*,* ,WINTER_DAY ,1998]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
1293 :=

1294 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1295 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1296 [*,* ,WINTER_DAY ,1999]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
1297 :=

1298 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1299 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1300 [*,* ,WINTER_DAY ,2000]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
1301 :=

1302 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1303 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1304 [*,* ,WINTER_DAY ,2001]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
1305 :=
```

```

1291 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1292 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1293 [*,* ,WINTER_DAY ,2002]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
   :=
1294 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1295 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1296 [*,* ,WINTER_DAY ,2003]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
   :=
1297 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1298 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1299 [*,* ,WINTER_DAY ,2004]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
   :=
1300 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1301 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1302 [*,* ,WINTER_DAY ,2005]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
   :=
1303 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1304 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1305 [*,* ,WINTER_DAY ,2006]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
   :=
1306 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1307 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1308 [*,* ,WINTER_DAY ,2007]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
   :=
1309 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1310 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1311 [*,* ,WINTER_DAY ,2008]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
   :=
1312 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1313 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1314 [*,* ,WINTER_DAY ,2009]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
   :=
1315 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1316 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1317 [*,* ,WINTER_DAY ,2010]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX
   :=
1318 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1319 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1320 [*,* ,WINTER_NIGHT ,1990]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL
   TX  :=
1321 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1322 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1323 [*,* ,WINTER_NIGHT ,1991]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL
   TX  :=
1324 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1325 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1326 [*,* ,WINTER_NIGHT ,1992]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL
   TX  :=
1327 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1328 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1329 [*,* ,WINTER_NIGHT ,1993]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL
   TX  :=
1330 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1331 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1332 [*,* ,WINTER_NIGHT ,1994]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL
   TX  :=
1333 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1334 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0

```

```

1335 [*,* ,WINTER_NIGHT ,1995]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL
    TX :=
1336 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1337 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1338 [*,* ,WINTER_NIGHT ,1996]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL
    TX :=
1339 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1340 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1341 [*,* ,WINTER_NIGHT ,1997]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL
    TX :=
1342 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1343 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1344 [*,* ,WINTER_NIGHT ,1998]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL
    TX :=
1345 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1346 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1347 [*,* ,WINTER_NIGHT ,1999]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL
    TX :=
1348 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1349 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1350 [*,* ,WINTER_NIGHT ,2000]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL
    TX :=
1351 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1352 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1353 [*,* ,WINTER_NIGHT ,2001]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL
    TX :=
1354 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1355 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1356 [*,* ,WINTER_NIGHT ,2002]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL
    TX :=
1357 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1358 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1359 [*,* ,WINTER_NIGHT ,2003]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL
    TX :=
1360 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1361 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1362 [*,* ,WINTER_NIGHT ,2004]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL
    TX :=
1363 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1364 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1365 [*,* ,WINTER_NIGHT ,2005]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL
    TX :=
1366 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1367 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1368 [*,* ,WINTER_NIGHT ,2006]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL
    TX :=
1369 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1370 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1371 [*,* ,WINTER_NIGHT ,2007]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL
    TX :=
1372 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1373 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1374 [*,* ,WINTER_NIGHT ,2008]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL
    TX :=
1375 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1376 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1377 [*,* ,WINTER_NIGHT ,2009]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL
    TX :=

```

```

1378 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1379 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1380 [*,* ,WINTER_NIGHT ,2010]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL
1381 TX :=  

1382 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1383 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1384 ;
1385 #
1386 param AccumulatedAnnualDemand default 1:-
1387 [*,* ,1990]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX :=  

1388 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1389 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1390 [*,* ,1991]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX :=  

1391 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1392 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1393 [*,* ,1992]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX :=  

1394 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1395 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1396 [*,* ,1993]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX :=  

1397 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1398 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1399 [*,* ,1994]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX :=  

1400 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1401 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1402 [*,* ,1995]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX :=  

1403 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1404 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1405 [*,* ,1996]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX :=  

1406 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1407 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1408 [*,* ,1997]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX :=  

1409 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1410 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1411 [*,* ,1998]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX :=  

1412 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1413 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1414 [*,* ,1999]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX :=  

1415 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1416 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1417 [*,* ,2000]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX :=  

1418 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1419 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1420 [*,* ,2001]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX :=  

1421 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1422 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1423 [*,* ,2002]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX :=  

1424 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1425 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1426 [*,* ,2003]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX :=  

1427 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1428 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1429 [*,* ,2004]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX :=  

1430 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1431 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1432 [*,* ,2005]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX :=  

1433 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1434 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1435 [*,* ,2006]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX :=
```

```

1435 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1436 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1437 [*,* ,2007]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX := 
1438 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1439 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1440 [*,* ,2008]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX := 
1441 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1442 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1443 [*,* ,2009]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX := 
1444 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1445 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1446 [*,* ,2010]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX := 
1447 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1448 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1449 ;
1450 #
1451 param CapacityToActivityUnit :E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1
1452     IMPHCO1 IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu
1453     := 
1454 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1455     1.0 1.0 1.0 1.0 1.0 1.0
1456 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1457     1.0 1.0 1.0 1.0 1.0 1.0
1458 ;
1459 #
1460 param CapacityFactor default 1:=
1461 [*,* ,INTERMEDIATE_DAY,1990]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1
1462     IMPHCO1 IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu
1463     := 
1464 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1465     1.0 1.0 1.0 1.0 1.0 1.0
1466 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1467     1.0 1.0 1.0 1.0 1.0 1.0
1468 [*,* ,INTERMEDIATE_DAY,1991]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1
1469     IMPHCO1 IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu
1470     := 
1471 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1472     1.0 1.0 1.0 1.0 1.0 1.0
1473 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1474     1.0 1.0 1.0 1.0 1.0 1.0
1475 [*,* ,INTERMEDIATE_DAY,1992]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1
1476     IMPHCO1 IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu
1477     := 
1478 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1479     1.0 1.0 1.0 1.0 1.0 1.0
1480 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1481     1.0 1.0 1.0 1.0 1.0 1.0
1482 [*,* ,INTERMEDIATE_DAY,1993]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1
1483     IMPHCO1 IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu
1484     := 
1485 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1486     1.0 1.0 1.0 1.0 1.0 1.0
1487 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1488     1.0 1.0 1.0 1.0 1.0 1.0
1489 [*,* ,INTERMEDIATE_DAY,1994]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1
1490     IMPHCO1 IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu
1491     := 

```



```

1520 [*,* ,INTERMEDIATE_NIGHT ,1990]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1
    IMPHCO1 IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu
    :=
1521 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
1522 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
1523 [*,* ,INTERMEDIATE_NIGHT ,1991]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1
    IMPHCO1 IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu
    :=
1524 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
1525 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
1526 [*,* ,INTERMEDIATE_NIGHT ,1992]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1
    IMPHCO1 IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu
    :=
1527 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
1528 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
1529 [*,* ,INTERMEDIATE_NIGHT ,1993]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1
    IMPHCO1 IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu
    :=
1530 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
1531 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
1532 [*,* ,INTERMEDIATE_NIGHT ,1994]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1
    IMPHCO1 IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu
    :=
1533 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
1534 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
1535 [*,* ,INTERMEDIATE_NIGHT ,1995]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1
    IMPHCO1 IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu
    :=
1536 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
1537 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
1538 [*,* ,INTERMEDIATE_NIGHT ,1996]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1
    IMPHCO1 IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu
    :=
1539 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
1540 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
1541 [*,* ,INTERMEDIATE_NIGHT ,1997]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1
    IMPHCO1 IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu
    :=
1542 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
1543 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
1544 [*,* ,INTERMEDIATE_NIGHT ,1998]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1
    IMPHCO1 IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu

```



```

1625 [*,* ,SUMMER_DAY ,2004]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
    IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
1626 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0
1627 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0
1628 [*,* ,SUMMER_DAY ,2005]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
    IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
1629 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0
1630 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0
1631 [*,* ,SUMMER_DAY ,2006]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
    IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
1632 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0
1633 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0
1634 [*,* ,SUMMER_DAY ,2007]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
    IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
1635 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0
1636 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0
1637 [*,* ,SUMMER_DAY ,2008]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
    IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
1638 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0
1639 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0
1640 [*,* ,SUMMER_DAY ,2009]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
    IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
1641 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0
1642 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0
1643 [*,* ,SUMMER_DAY ,2010]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
    IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
1644 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0
1645 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0
1646 [*,* ,SUMMER_NIGHT ,1990]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
    IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
1647 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0
1648 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0
1649 [*,* ,SUMMER_NIGHT ,1991]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
    IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
1650 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0
1651 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0
1652 [*,* ,SUMMER_NIGHT ,1992]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
    IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
1653 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0

```



```

1712 [*,* ,WINTER_DAY ,1991]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
     IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
1713 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
     1.0 1.0 1.0 1.0 1.0 1.0
1714 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
     1.0 1.0 1.0 1.0 1.0 1.0
1715 [*,* ,WINTER_DAY ,1992]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
     IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
1716 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
     1.0 1.0 1.0 1.0 1.0 1.0
1717 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
     1.0 1.0 1.0 1.0 1.0 1.0
1718 [*,* ,WINTER_DAY ,1993]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
     IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
1719 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
     1.0 1.0 1.0 1.0 1.0 1.0
1720 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
     1.0 1.0 1.0 1.0 1.0 1.0
1721 [*,* ,WINTER_DAY ,1994]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
     IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
1722 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
     1.0 1.0 1.0 1.0 1.0 1.0
1723 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
     1.0 1.0 1.0 1.0 1.0 1.0
1724 [*,* ,WINTER_DAY ,1995]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
     IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
1725 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
     1.0 1.0 1.0 1.0 1.0 1.0
1726 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
     1.0 1.0 1.0 1.0 1.0 1.0
1727 [*,* ,WINTER_DAY ,1996]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
     IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
1728 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
     1.0 1.0 1.0 1.0 1.0 1.0
1729 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
     1.0 1.0 1.0 1.0 1.0 1.0
1730 [*,* ,WINTER_DAY ,1997]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
     IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
1731 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
     1.0 1.0 1.0 1.0 1.0 1.0
1732 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
     1.0 1.0 1.0 1.0 1.0 1.0
1733 [*,* ,WINTER_DAY ,1998]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
     IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
1734 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
     1.0 1.0 1.0 1.0 1.0 1.0
1735 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
     1.0 1.0 1.0 1.0 1.0 1.0
1736 [*,* ,WINTER_DAY ,1999]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
     IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
1737 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
     1.0 1.0 1.0 1.0 1.0 1.0
1738 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
     1.0 1.0 1.0 1.0 1.0 1.0
1739 [*,* ,WINTER_DAY ,2000]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
     IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
1740 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
     1.0 1.0 1.0 1.0 1.0 1.0

```



```

1887 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1888 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1889 [*,* ,2007]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
    IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu := 
1890 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1891 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1892 [*,* ,2008]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
    IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu := 
1893 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1894 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1895 [*,* ,2009]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
    IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu := 
1896 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1897 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1898 [*,* ,2010]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
    IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu := 
1899 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1900 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1901 ;
1902 #
1903 param OperationalLife :E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
    IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu := 
1904 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1905 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1906 ;
1907 #
1908 param ResidualCapacity default 1:=
1909 [*,* ,1990]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
    IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu := 
1910 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1911 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1912 [*,* ,1991]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
    IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu := 
1913 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1914 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1915 [*,* ,1992]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
    IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu := 
1916 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1917 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0

```



```

3485 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
3486 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
3487 ;
3488 #
3489 param OutputActivityRatio default 1:=
3490 [*,* ,CSV ,1 ,1990]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
3491 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
3491 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
3492 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
3493 [*,* ,CSV ,1 ,1991]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
3493 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
3494 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
3495 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
3496 [*,* ,CSV ,1 ,1992]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
3496 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
3497 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
3498 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
3499 [*,* ,CSV ,1 ,1993]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
3499 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
3500 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
3501 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
3502 [*,* ,CSV ,1 ,1994]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
3502 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
3503 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
3504 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
3505 [*,* ,CSV ,1 ,1995]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
3505 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
3506 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
3507 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
3508 [*,* ,CSV ,1 ,1996]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
3508 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
3509 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
3510 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
3511 [*,* ,CSV ,1 ,1997]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
3511 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
3512 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
3513 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
3514 [*,* ,CSV ,1 ,1998]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
3514 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=

```



```

3979 [*,* ,GSL ,2 ,2006]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
    IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
3980 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
3981 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
3982 [*,* ,GSL ,2 ,2007]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
    IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
3983 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
3984 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
3985 [*,* ,GSL ,2 ,2008]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
    IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
3986 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
3987 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
3988 [*,* ,GSL ,2 ,2009]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
    IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
3989 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
3990 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
3991 [*,* ,GSL ,2 ,2010]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
    IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
3992 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
3993 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
3994 [*,* ,HCO ,1 ,1990]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
    IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
3995 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
3996 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
3997 [*,* ,HCO ,1 ,1991]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
    IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
3998 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
3999 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
4000 [*,* ,HCO ,1 ,1992]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
    IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
4001 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
4002 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
4003 [*,* ,HCO ,1 ,1993]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
    IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
4004 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
4005 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
4006 [*,* ,HCO ,1 ,1994]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
    IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
4007 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0

```



```

5053 [*,* ,2006]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5054 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5055 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5056 [*,* ,2007]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5057 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5058 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5059 [*,* ,2008]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5060 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5061 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5062 [*,* ,2009]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5063 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5064 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5065 [*,* ,2010]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5066 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5067 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5068 ;
5069 #
5070 param VariableCost default 1:=
5071 [*,* ,1,1990]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5072 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5073 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5074 [*,* ,1,1991]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5075 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5076 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5077 [*,* ,1,1992]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5078 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5079 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5080 [*,* ,1,1993]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5081 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5082 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0

```

```

5083 [*,* ,1,1994]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5084 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5085 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5086 [*,* ,1,1995]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5087 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5088 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5089 [*,* ,1,1996]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5090 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5091 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5092 [*,* ,1,1997]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5093 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5094 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5095 [*,* ,1,1998]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5096 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5097 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5098 [*,* ,1,1999]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5099 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5100 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5101 [*,* ,1,2000]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5102 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5103 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5104 [*,* ,1,2001]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5105 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5106 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5107 [*,* ,1,2002]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5108 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5109 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5110 [*,* ,1,2003]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5111 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0

```



```

5170 [*,* ,2,2002]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5171 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5172 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5173 [*,* ,2,2003]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5174 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5175 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5176 [*,* ,2,2004]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5177 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5178 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5179 [*,* ,2,2005]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5180 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5181 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5182 [*,* ,2,2006]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5183 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5184 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5185 [*,* ,2,2007]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5186 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5187 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5188 [*,* ,2,2008]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5189 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5190 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5191 [*,* ,2,2009]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5192 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5193 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5194 [*,* ,2,2010]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5195 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5196 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5197 ;
5198 #
5199 param FixedCost default 1:=

```



```

5258 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5259 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5260 [*,* ,2010]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5261 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5262 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5263 ;
5264 #
5265 param TechnologyToStorage default 1:=
5266 [*,* ,DAM,1]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5267 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5268 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5269 [*,* ,DAM,2]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5270 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5271 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5272 ;
5273 #
5274 param TechnologyFromStorage default 1:=
5275 [*,* ,DAM,1]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5276 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5277 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5278 [*,* ,DAM,2]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5279 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5280 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5281 ;
5282 #
5283 param StorageLevelStart :DAM:=
5284 NEWZEALAND 1.0
5285 AUSTRALIA 1.0
5286 ;
5287 #
5288 param StorageMaxChargeRate :DAM:=
5289 NEWZEALAND 1.0
5290 AUSTRALIA 1.0
5291 ;
5292 #
5293 param StorageMaxDischargeRate :DAM:=
5294 NEWZEALAND 1.0
5295 AUSTRALIA 1.0
5296 ;
5297 #
5298 param MinStorageCharge default 1:=

```

```
5299 [*,* ,1990]: DAM :=  
5300 NEWZEALAND 1.0  
5301 AUSTRALIA 1.0  
5302 [*,* ,1991]: DAM :=  
5303 NEWZEALAND 1.0  
5304 AUSTRALIA 1.0  
5305 [*,* ,1992]: DAM :=  
5306 NEWZEALAND 1.0  
5307 AUSTRALIA 1.0  
5308 [*,* ,1993]: DAM :=  
5309 NEWZEALAND 1.0  
5310 AUSTRALIA 1.0  
5311 [*,* ,1994]: DAM :=  
5312 NEWZEALAND 1.0  
5313 AUSTRALIA 1.0  
5314 [*,* ,1995]: DAM :=  
5315 NEWZEALAND 1.0  
5316 AUSTRALIA 1.0  
5317 [*,* ,1996]: DAM :=  
5318 NEWZEALAND 1.0  
5319 AUSTRALIA 1.0  
5320 [*,* ,1997]: DAM :=  
5321 NEWZEALAND 1.0  
5322 AUSTRALIA 1.0  
5323 [*,* ,1998]: DAM :=  
5324 NEWZEALAND 1.0  
5325 AUSTRALIA 1.0  
5326 [*,* ,1999]: DAM :=  
5327 NEWZEALAND 1.0  
5328 AUSTRALIA 1.0  
5329 [*,* ,2000]: DAM :=  
5330 NEWZEALAND 1.0  
5331 AUSTRALIA 1.0  
5332 [*,* ,2001]: DAM :=  
5333 NEWZEALAND 1.0  
5334 AUSTRALIA 1.0  
5335 [*,* ,2002]: DAM :=  
5336 NEWZEALAND 1.0  
5337 AUSTRALIA 1.0  
5338 [*,* ,2003]: DAM :=  
5339 NEWZEALAND 1.0  
5340 AUSTRALIA 1.0  
5341 [*,* ,2004]: DAM :=  
5342 NEWZEALAND 1.0  
5343 AUSTRALIA 1.0  
5344 [*,* ,2005]: DAM :=  
5345 NEWZEALAND 1.0  
5346 AUSTRALIA 1.0  
5347 [*,* ,2006]: DAM :=  
5348 NEWZEALAND 1.0  
5349 AUSTRALIA 1.0  
5350 [*,* ,2007]: DAM :=  
5351 NEWZEALAND 1.0  
5352 AUSTRALIA 1.0  
5353 [*,* ,2008]: DAM :=  
5354 NEWZEALAND 1.0  
5355 AUSTRALIA 1.0  
5356 [*,* ,2009]: DAM :=
```

```
5357 NEWZEALAND 1.0
5358 AUSTRALIA 1.0
5359 [* ,*,2010]: DAM := 
5360 NEWZEALAND 1.0
5361 AUSTRALIA 1.0
5362 ;
5363 #
5364 param OperationalLifeStorage :DAM:=
5365 NEWZEALAND 1.0
5366 AUSTRALIA 1.0
5367 ;
5368 #
5369 param CapitalCostStorage default 1:=
5370 [* ,*,1990]: DAM :=
5371 NEWZEALAND 1.0
5372 AUSTRALIA 1.0
5373 [* ,*,1991]: DAM :=
5374 NEWZEALAND 1.0
5375 AUSTRALIA 1.0
5376 [* ,*,1992]: DAM :=
5377 NEWZEALAND 1.0
5378 AUSTRALIA 1.0
5379 [* ,*,1993]: DAM :=
5380 NEWZEALAND 1.0
5381 AUSTRALIA 1.0
5382 [* ,*,1994]: DAM :=
5383 NEWZEALAND 1.0
5384 AUSTRALIA 1.0
5385 [* ,*,1995]: DAM :=
5386 NEWZEALAND 1.0
5387 AUSTRALIA 1.0
5388 [* ,*,1996]: DAM :=
5389 NEWZEALAND 1.0
5390 AUSTRALIA 1.0
5391 [* ,*,1997]: DAM :=
5392 NEWZEALAND 1.0
5393 AUSTRALIA 1.0
5394 [* ,*,1998]: DAM :=
5395 NEWZEALAND 1.0
5396 AUSTRALIA 1.0
5397 [* ,*,1999]: DAM :=
5398 NEWZEALAND 1.0
5399 AUSTRALIA 1.0
5400 [* ,*,2000]: DAM :=
5401 NEWZEALAND 1.0
5402 AUSTRALIA 1.0
5403 [* ,*,2001]: DAM :=
5404 NEWZEALAND 1.0
5405 AUSTRALIA 1.0
5406 [* ,*,2002]: DAM :=
5407 NEWZEALAND 1.0
5408 AUSTRALIA 1.0
5409 [* ,*,2003]: DAM :=
5410 NEWZEALAND 1.0
5411 AUSTRALIA 1.0
5412 [* ,*,2004]: DAM :=
5413 NEWZEALAND 1.0
5414 AUSTRALIA 1.0
```

```

5415 [*,* ,2005]: DAM :=
5416 NEWZEALAND 1.0
5417 AUSTRALIA 1.0
5418 [*,* ,2006]: DAM :=
5419 NEWZEALAND 1.0
5420 AUSTRALIA 1.0
5421 [*,* ,2007]: DAM :=
5422 NEWZEALAND 1.0
5423 AUSTRALIA 1.0
5424 [*,* ,2008]: DAM :=
5425 NEWZEALAND 1.0
5426 AUSTRALIA 1.0
5427 [*,* ,2009]: DAM :=
5428 NEWZEALAND 1.0
5429 AUSTRALIA 1.0
5430 [*,* ,2010]: DAM :=
5431 NEWZEALAND 1.0
5432 AUSTRALIA 1.0
5433 ;
5434 #
5435 param ResidualStorageCapacity default 1:-
5436 [*,* ,1990]: DAM :=
5437 NEWZEALAND 1.0
5438 AUSTRALIA 1.0
5439 [*,* ,1991]: DAM :=
5440 NEWZEALAND 1.0
5441 AUSTRALIA 1.0
5442 [*,* ,1992]: DAM :=
5443 NEWZEALAND 1.0
5444 AUSTRALIA 1.0
5445 [*,* ,1993]: DAM :=
5446 NEWZEALAND 1.0
5447 AUSTRALIA 1.0
5448 [*,* ,1994]: DAM :=
5449 NEWZEALAND 1.0
5450 AUSTRALIA 1.0
5451 [*,* ,1995]: DAM :=
5452 NEWZEALAND 1.0
5453 AUSTRALIA 1.0
5454 [*,* ,1996]: DAM :=
5455 NEWZEALAND 1.0
5456 AUSTRALIA 1.0
5457 [*,* ,1997]: DAM :=
5458 NEWZEALAND 1.0
5459 AUSTRALIA 1.0
5460 [*,* ,1998]: DAM :=
5461 NEWZEALAND 1.0
5462 AUSTRALIA 1.0
5463 [*,* ,1999]: DAM :=
5464 NEWZEALAND 1.0
5465 AUSTRALIA 1.0
5466 [*,* ,2000]: DAM :=
5467 NEWZEALAND 1.0
5468 AUSTRALIA 1.0
5469 [*,* ,2001]: DAM :=
5470 NEWZEALAND 1.0
5471 AUSTRALIA 1.0
5472 [*,* ,2002]: DAM :=

```

```

5473 NEWZEALAND 1.0
5474 AUSTRALIA 1.0
5475 [*,* ,2003]: DAM :=
5476 NEWZEALAND 1.0
5477 AUSTRALIA 1.0
5478 [*,* ,2004]: DAM :=
5479 NEWZEALAND 1.0
5480 AUSTRALIA 1.0
5481 [*,* ,2005]: DAM :=
5482 NEWZEALAND 1.0
5483 AUSTRALIA 1.0
5484 [*,* ,2006]: DAM :=
5485 NEWZEALAND 1.0
5486 AUSTRALIA 1.0
5487 [*,* ,2007]: DAM :=
5488 NEWZEALAND 1.0
5489 AUSTRALIA 1.0
5490 [*,* ,2008]: DAM :=
5491 NEWZEALAND 1.0
5492 AUSTRALIA 1.0
5493 [*,* ,2009]: DAM :=
5494 NEWZEALAND 1.0
5495 AUSTRALIA 1.0
5496 [*,* ,2010]: DAM :=
5497 NEWZEALAND 1.0
5498 AUSTRALIA 1.0
5499 ;
5500 #
5501 param CapacityOfOneTechnologyUnit default 1:-
5502 [*,* ,1990]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5503 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0
5504 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0
5505 [*,* ,1991]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5506 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0
5507 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0
5508 [*,* ,1992]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5509 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0
5510 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0
5511 [*,* ,1993]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5512 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0
5513 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0
5514 [*,* ,1994]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5515 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0

```



```

5604 [*,* ,2002]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5605 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5606 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5607 [*,* ,2003]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5608 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5609 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5610 [*,* ,2004]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5611 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5612 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5613 [*,* ,2005]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5614 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5615 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5616 [*,* ,2006]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5617 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5618 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5619 [*,* ,2007]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5620 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5621 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5622 [*,* ,2008]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5623 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5624 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5625 [*,* ,2009]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5626 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5627 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5628 [*,* ,2010]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5629 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5630 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5631 ;
5632 #
5633 param TotalAnnualMinCapacity default 1:=

```



```

5751 [*,* ,2007]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5752 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0
5753 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0
5754 [*,* ,2008]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5755 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0
5756 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0
5757 [*,* ,2009]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5758 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0
5759 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0
5760 [*,* ,2010]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5761 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0
5762 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0
5763 ;
5764 #
5765 param TotalAnnualMinCapacityInvestment default 1:=
5766 [*,* ,1990]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5767 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0
5768 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0
5769 [*,* ,1991]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5770 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0
5771 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0
5772 [*,* ,1992]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5773 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0
5774 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0
5775 [*,* ,1993]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5776 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0
5777 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0
5778 [*,* ,1994]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5779 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0
5780 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0

```

```

5781 [*,* ,1995]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
    IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5782 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
5783 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
5784 [*,* ,1996]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
    IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5785 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
5786 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
5787 [*,* ,1997]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
    IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5788 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
5789 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
5790 [*,* ,1998]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
    IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5791 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
5792 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
5793 [*,* ,1999]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
    IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5794 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
5795 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
5796 [*,* ,2000]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
    IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5797 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
5798 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
5799 [*,* ,2001]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
    IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5800 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
5801 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
5802 [*,* ,2002]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
    IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5803 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
5804 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
5805 [*,* ,2003]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
    IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5806 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
5807 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
5808 [*,* ,2004]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
    IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5809 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0

```



```

5869 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5870 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5871 [*,* ,2003]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5872 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5873 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5874 [*,* ,2004]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5875 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5876 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5877 [*,* ,2005]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5878 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5879 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5880 [*,* ,2006]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5881 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5882 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5883 [*,* ,2007]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5884 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5885 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5886 [*,* ,2008]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5887 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5888 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5889 [*,* ,2009]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5890 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5891 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5892 [*,* ,2010]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
5893 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5894 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      1.0 1.0 1.0 1.0 1.0 1.0
5895 ;
5896 #
5897 param TotalTechnologyAnnualActivityUpperLimit default 1:=
5898 [*,* ,1990]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
      IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=

```



```

6055 [*,* ,1995]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX :=  

6056 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6057 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6058 [*,* ,1996]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX :=  

6059 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6060 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6061 [*,* ,1997]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX :=  

6062 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6063 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6064 [*,* ,1998]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX :=  

6065 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6066 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6067 [*,* ,1999]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX :=  

6068 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6069 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6070 [*,* ,2000]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX :=  

6071 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6072 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6073 [*,* ,2001]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX :=  

6074 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6075 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6076 [*,* ,2002]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX :=  

6077 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6078 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6079 [*,* ,2003]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX :=  

6080 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6081 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6082 [*,* ,2004]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX :=  

6083 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6084 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6085 [*,* ,2005]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX :=  

6086 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6087 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6088 [*,* ,2006]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX :=  

6089 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6090 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6091 [*,* ,2007]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX :=  

6092 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6093 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6094 [*,* ,2008]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX :=  

6095 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6096 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6097 [*,* ,2009]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX :=  

6098 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6099 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6100 [*,* ,2010]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX :=  

6101 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6102 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6103 ;  

6104 #  

6105 param ReserveMargin :1990 1991 1992 1993 1994 1995 1996 1997 1998 1999  

       2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010:=  

6106 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

       1.0 1.0 1.0 1.0 1.0 1.0  

6107 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

       1.0 1.0 1.0 1.0 1.0 1.0  

6108 ;  

6109 #

```

```

6110 param RETagTechnology default 1:=
6111 [*,* ,1990]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
6112 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=  

6112 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6113 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6113 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6114 [*,* ,1991]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
6115 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=  

6115 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6116 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6116 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6117 [*,* ,1992]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
6118 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=  

6118 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6119 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6119 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6120 [*,* ,1993]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
6121 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=  

6121 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6122 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6122 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6123 [*,* ,1994]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
6124 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=  

6124 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6125 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6125 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6126 [*,* ,1995]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
6127 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=  

6127 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6128 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6128 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6129 [*,* ,1996]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
6130 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=  

6130 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6131 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6131 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6132 [*,* ,1997]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
6133 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=  

6133 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6134 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6134 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6135 [*,* ,1998]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
6136 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=  

6136 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6137 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6137 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0  

6138 [*,* ,1999]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
6138 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=

```



```

6220 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
6221 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
6222 [*,* ,2005]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX := 
6223 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
6224 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
6225 [*,* ,2006]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX := 
6226 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
6227 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
6228 [*,* ,2007]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX := 
6229 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
6230 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
6231 [*,* ,2008]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX := 
6232 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
6233 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
6234 [*,* ,2009]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX := 
6235 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
6236 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
6237 [*,* ,2010]: CSV DSL ELC GSL HCO HYD LTH OIL URN RH RL TX := 
6238 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
6239 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
6240 ;
6241 #
6242 param REMinProductionTarget :1990 1991 1992 1993 1994 1995 1996 1997
       1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010:=
6243 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
       1.0 1.0 1.0 1.0 1.0 1.0
6244 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
       1.0 1.0 1.0 1.0 1.0 1.0
6245 ;
6246 #
6247 param EmissionActivityRatio default 1:=
6248 [*,* ,C02 ,1,1990]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
       IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
6249 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
       1.0 1.0 1.0 1.0 1.0 1.0
6250 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
       1.0 1.0 1.0 1.0 1.0 1.0
6251 [*,* ,C02 ,1,1991]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
       IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
6252 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
       1.0 1.0 1.0 1.0 1.0 1.0
6253 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
       1.0 1.0 1.0 1.0 1.0 1.0
6254 [*,* ,C02 ,1,1992]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
       IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
6255 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
       1.0 1.0 1.0 1.0 1.0 1.0
6256 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
       1.0 1.0 1.0 1.0 1.0 1.0
6257 [*,* ,C02 ,1,1993]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
       IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
6258 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
       1.0 1.0 1.0 1.0 1.0 1.0
6259 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
       1.0 1.0 1.0 1.0 1.0 1.0
6260 [*,* ,C02 ,1,1994]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01 IMPOIL1
       IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=

```



```

6638 [*,* ,METHANE ,1 ,1994]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
    IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
6639 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
6640 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
6641 [*,* ,METHANE ,1 ,1995]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
    IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
6642 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
6643 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
6644 [*,* ,METHANE ,1 ,1996]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
    IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
6645 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
6646 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
6647 [*,* ,METHANE ,1 ,1997]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
    IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
6648 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
6649 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
6650 [*,* ,METHANE ,1 ,1998]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
    IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
6651 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
6652 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
6653 [*,* ,METHANE ,1 ,1999]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
    IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
6654 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
6655 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
6656 [*,* ,METHANE ,1 ,2000]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
    IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
6657 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
6658 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
6659 [*,* ,METHANE ,1 ,2001]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
    IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
6660 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
6661 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
6662 [*,* ,METHANE ,1 ,2002]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
    IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
6663 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
6664 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
6665 [*,* ,METHANE ,1 ,2003]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
    IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
6666 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0

```



```

6725 [*,* ,METHANE ,2 ,2002]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
    IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
6726 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
6727 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
6728 [*,* ,METHANE ,2 ,2003]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
    IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
6729 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
6730 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
6731 [*,* ,METHANE ,2 ,2004]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
    IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
6732 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
6733 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
6734 [*,* ,METHANE ,2 ,2005]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
    IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
6735 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
6736 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
6737 [*,* ,METHANE ,2 ,2006]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
    IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
6738 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
6739 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
6740 [*,* ,METHANE ,2 ,2007]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
    IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
6741 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
6742 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
6743 [*,* ,METHANE ,2 ,2008]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
    IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
6744 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
6745 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
6746 [*,* ,METHANE ,2 ,2009]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
    IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
6747 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
6748 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
6749 [*,* ,METHANE ,2 ,2010]: E01 E21 E31 E51 E70 IMPDSL1 IMPGSL1 IMPHC01
    IMPOIL1 IMPURN1 RHE RHO RL1 SRE TXD TXE TXG RIV RHu RLu TXu :=
6750 NEWZEALAND 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
6751 AUSTRALIA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0
6752 ;
6753 #
6754 param EmissionsPenalty default 1:=
6755 [*,* ,1990]: CO2 NOX CO METHANE  :=

```

```

6756 NEWZEALAND 1.0 1.0 1.0 1.0
6757 AUSTRALIA 1.0 1.0 1.0 1.0
6758 [*,* ,1991]: CO2 NOX CO METHANE := 
6759 NEWZEALAND 1.0 1.0 1.0 1.0
6760 AUSTRALIA 1.0 1.0 1.0 1.0
6761 [*,* ,1992]: CO2 NOX CO METHANE := 
6762 NEWZEALAND 1.0 1.0 1.0 1.0
6763 AUSTRALIA 1.0 1.0 1.0 1.0
6764 [*,* ,1993]: CO2 NOX CO METHANE := 
6765 NEWZEALAND 1.0 1.0 1.0 1.0
6766 AUSTRALIA 1.0 1.0 1.0 1.0
6767 [*,* ,1994]: CO2 NOX CO METHANE := 
6768 NEWZEALAND 1.0 1.0 1.0 1.0
6769 AUSTRALIA 1.0 1.0 1.0 1.0
6770 [*,* ,1995]: CO2 NOX CO METHANE := 
6771 NEWZEALAND 1.0 1.0 1.0 1.0
6772 AUSTRALIA 1.0 1.0 1.0 1.0
6773 [*,* ,1996]: CO2 NOX CO METHANE := 
6774 NEWZEALAND 1.0 1.0 1.0 1.0
6775 AUSTRALIA 1.0 1.0 1.0 1.0
6776 [*,* ,1997]: CO2 NOX CO METHANE := 
6777 NEWZEALAND 1.0 1.0 1.0 1.0
6778 AUSTRALIA 1.0 1.0 1.0 1.0
6779 [*,* ,1998]: CO2 NOX CO METHANE := 
6780 NEWZEALAND 1.0 1.0 1.0 1.0
6781 AUSTRALIA 1.0 1.0 1.0 1.0
6782 [*,* ,1999]: CO2 NOX CO METHANE := 
6783 NEWZEALAND 1.0 1.0 1.0 1.0
6784 AUSTRALIA 1.0 1.0 1.0 1.0
6785 [*,* ,2000]: CO2 NOX CO METHANE := 
6786 NEWZEALAND 1.0 1.0 1.0 1.0
6787 AUSTRALIA 1.0 1.0 1.0 1.0
6788 [*,* ,2001]: CO2 NOX CO METHANE := 
6789 NEWZEALAND 1.0 1.0 1.0 1.0
6790 AUSTRALIA 1.0 1.0 1.0 1.0
6791 [*,* ,2002]: CO2 NOX CO METHANE := 
6792 NEWZEALAND 1.0 1.0 1.0 1.0
6793 AUSTRALIA 1.0 1.0 1.0 1.0
6794 [*,* ,2003]: CO2 NOX CO METHANE := 
6795 NEWZEALAND 1.0 1.0 1.0 1.0
6796 AUSTRALIA 1.0 1.0 1.0 1.0
6797 [*,* ,2004]: CO2 NOX CO METHANE := 
6798 NEWZEALAND 1.0 1.0 1.0 1.0
6799 AUSTRALIA 1.0 1.0 1.0 1.0
6800 [*,* ,2005]: CO2 NOX CO METHANE := 
6801 NEWZEALAND 1.0 1.0 1.0 1.0
6802 AUSTRALIA 1.0 1.0 1.0 1.0
6803 [*,* ,2006]: CO2 NOX CO METHANE := 
6804 NEWZEALAND 1.0 1.0 1.0 1.0
6805 AUSTRALIA 1.0 1.0 1.0 1.0
6806 [*,* ,2007]: CO2 NOX CO METHANE := 
6807 NEWZEALAND 1.0 1.0 1.0 1.0
6808 AUSTRALIA 1.0 1.0 1.0 1.0
6809 [*,* ,2008]: CO2 NOX CO METHANE := 
6810 NEWZEALAND 1.0 1.0 1.0 1.0
6811 AUSTRALIA 1.0 1.0 1.0 1.0
6812 [*,* ,2009]: CO2 NOX CO METHANE := 
6813 NEWZEALAND 1.0 1.0 1.0 1.0

```

```

6814 AUSTRALIA 1.0 1.0 1.0 1.0
6815 [*,* ,2010]: CO2 NOX CO METHANE := 
6816 NEWZEALAND 1.0 1.0 1.0 1.0
6817 AUSTRALIA 1.0 1.0 1.0 1.0
6818 ;
6819 #
6820 param AnnualExogenousEmission default 1:=
6821 [*,* ,1990]: CO2 NOX CO METHANE := 
6822 NEWZEALAND 1.0 1.0 1.0 1.0
6823 AUSTRALIA 1.0 1.0 1.0 1.0
6824 [*,* ,1991]: CO2 NOX CO METHANE := 
6825 NEWZEALAND 1.0 1.0 1.0 1.0
6826 AUSTRALIA 1.0 1.0 1.0 1.0
6827 [*,* ,1992]: CO2 NOX CO METHANE := 
6828 NEWZEALAND 1.0 1.0 1.0 1.0
6829 AUSTRALIA 1.0 1.0 1.0 1.0
6830 [*,* ,1993]: CO2 NOX CO METHANE := 
6831 NEWZEALAND 1.0 1.0 1.0 1.0
6832 AUSTRALIA 1.0 1.0 1.0 1.0
6833 [*,* ,1994]: CO2 NOX CO METHANE := 
6834 NEWZEALAND 1.0 1.0 1.0 1.0
6835 AUSTRALIA 1.0 1.0 1.0 1.0
6836 [*,* ,1995]: CO2 NOX CO METHANE := 
6837 NEWZEALAND 1.0 1.0 1.0 1.0
6838 AUSTRALIA 1.0 1.0 1.0 1.0
6839 [*,* ,1996]: CO2 NOX CO METHANE := 
6840 NEWZEALAND 1.0 1.0 1.0 1.0
6841 AUSTRALIA 1.0 1.0 1.0 1.0
6842 [*,* ,1997]: CO2 NOX CO METHANE := 
6843 NEWZEALAND 1.0 1.0 1.0 1.0
6844 AUSTRALIA 1.0 1.0 1.0 1.0
6845 [*,* ,1998]: CO2 NOX CO METHANE := 
6846 NEWZEALAND 1.0 1.0 1.0 1.0
6847 AUSTRALIA 1.0 1.0 1.0 1.0
6848 [*,* ,1999]: CO2 NOX CO METHANE := 
6849 NEWZEALAND 1.0 1.0 1.0 1.0
6850 AUSTRALIA 1.0 1.0 1.0 1.0
6851 [*,* ,2000]: CO2 NOX CO METHANE := 
6852 NEWZEALAND 1.0 1.0 1.0 1.0
6853 AUSTRALIA 1.0 1.0 1.0 1.0
6854 [*,* ,2001]: CO2 NOX CO METHANE := 
6855 NEWZEALAND 1.0 1.0 1.0 1.0
6856 AUSTRALIA 1.0 1.0 1.0 1.0
6857 [*,* ,2002]: CO2 NOX CO METHANE := 
6858 NEWZEALAND 1.0 1.0 1.0 1.0
6859 AUSTRALIA 1.0 1.0 1.0 1.0
6860 [*,* ,2003]: CO2 NOX CO METHANE := 
6861 NEWZEALAND 1.0 1.0 1.0 1.0
6862 AUSTRALIA 1.0 1.0 1.0 1.0
6863 [*,* ,2004]: CO2 NOX CO METHANE := 
6864 NEWZEALAND 1.0 1.0 1.0 1.0
6865 AUSTRALIA 1.0 1.0 1.0 1.0
6866 [*,* ,2005]: CO2 NOX CO METHANE := 
6867 NEWZEALAND 1.0 1.0 1.0 1.0
6868 AUSTRALIA 1.0 1.0 1.0 1.0
6869 [*,* ,2006]: CO2 NOX CO METHANE := 
6870 NEWZEALAND 1.0 1.0 1.0 1.0
6871 AUSTRALIA 1.0 1.0 1.0 1.0

```

```

6872 [*,* ,2007]: CO2 NOX CO METHANE :=  

6873 NEWZEALAND 1.0 1.0 1.0 1.0  

6874 AUSTRALIA 1.0 1.0 1.0 1.0  

6875 [*,* ,2008]: CO2 NOX CO METHANE :=  

6876 NEWZEALAND 1.0 1.0 1.0 1.0  

6877 AUSTRALIA 1.0 1.0 1.0 1.0  

6878 [*,* ,2009]: CO2 NOX CO METHANE :=  

6879 NEWZEALAND 1.0 1.0 1.0 1.0  

6880 AUSTRALIA 1.0 1.0 1.0 1.0  

6881 [*,* ,2010]: CO2 NOX CO METHANE :=  

6882 NEWZEALAND 1.0 1.0 1.0 1.0  

6883 AUSTRALIA 1.0 1.0 1.0 1.0  

6884 ;  

6885 #  

6886 param AnnualEmissionLimit default 1:=  

6887 [*,* ,1990]: CO2 NOX CO METHANE :=  

6888 NEWZEALAND 1.0 1.0 1.0 1.0  

6889 AUSTRALIA 1.0 1.0 1.0 1.0  

6890 [*,* ,1991]: CO2 NOX CO METHANE :=  

6891 NEWZEALAND 1.0 1.0 1.0 1.0  

6892 AUSTRALIA 1.0 1.0 1.0 1.0  

6893 [*,* ,1992]: CO2 NOX CO METHANE :=  

6894 NEWZEALAND 1.0 1.0 1.0 1.0  

6895 AUSTRALIA 1.0 1.0 1.0 1.0  

6896 [*,* ,1993]: CO2 NOX CO METHANE :=  

6897 NEWZEALAND 1.0 1.0 1.0 1.0  

6898 AUSTRALIA 1.0 1.0 1.0 1.0  

6899 [*,* ,1994]: CO2 NOX CO METHANE :=  

6900 NEWZEALAND 1.0 1.0 1.0 1.0  

6901 AUSTRALIA 1.0 1.0 1.0 1.0  

6902 [*,* ,1995]: CO2 NOX CO METHANE :=  

6903 NEWZEALAND 1.0 1.0 1.0 1.0  

6904 AUSTRALIA 1.0 1.0 1.0 1.0  

6905 [*,* ,1996]: CO2 NOX CO METHANE :=  

6906 NEWZEALAND 1.0 1.0 1.0 1.0  

6907 AUSTRALIA 1.0 1.0 1.0 1.0  

6908 [*,* ,1997]: CO2 NOX CO METHANE :=  

6909 NEWZEALAND 1.0 1.0 1.0 1.0  

6910 AUSTRALIA 1.0 1.0 1.0 1.0  

6911 [*,* ,1998]: CO2 NOX CO METHANE :=  

6912 NEWZEALAND 1.0 1.0 1.0 1.0  

6913 AUSTRALIA 1.0 1.0 1.0 1.0  

6914 [*,* ,1999]: CO2 NOX CO METHANE :=  

6915 NEWZEALAND 1.0 1.0 1.0 1.0  

6916 AUSTRALIA 1.0 1.0 1.0 1.0  

6917 [*,* ,2000]: CO2 NOX CO METHANE :=  

6918 NEWZEALAND 1.0 1.0 1.0 1.0  

6919 AUSTRALIA 1.0 1.0 1.0 1.0  

6920 [*,* ,2001]: CO2 NOX CO METHANE :=  

6921 NEWZEALAND 1.0 1.0 1.0 1.0  

6922 AUSTRALIA 1.0 1.0 1.0 1.0  

6923 [*,* ,2002]: CO2 NOX CO METHANE :=  

6924 NEWZEALAND 1.0 1.0 1.0 1.0  

6925 AUSTRALIA 1.0 1.0 1.0 1.0  

6926 [*,* ,2003]: CO2 NOX CO METHANE :=  

6927 NEWZEALAND 1.0 1.0 1.0 1.0  

6928 AUSTRALIA 1.0 1.0 1.0 1.0  

6929 [*,* ,2004]: CO2 NOX CO METHANE :=

```

```

6930 NEWZEALAND 1.0 1.0 1.0 1.0
6931 AUSTRALIA 1.0 1.0 1.0 1.0
6932 [*,* ,2005]: CO2 NOX CO METHANE := 
6933 NEWZEALAND 1.0 1.0 1.0 1.0
6934 AUSTRALIA 1.0 1.0 1.0 1.0
6935 [*,* ,2006]: CO2 NOX CO METHANE := 
6936 NEWZEALAND 1.0 1.0 1.0 1.0
6937 AUSTRALIA 1.0 1.0 1.0 1.0
6938 [*,* ,2007]: CO2 NOX CO METHANE := 
6939 NEWZEALAND 1.0 1.0 1.0 1.0
6940 AUSTRALIA 1.0 1.0 1.0 1.0
6941 [*,* ,2008]: CO2 NOX CO METHANE := 
6942 NEWZEALAND 1.0 1.0 1.0 1.0
6943 AUSTRALIA 1.0 1.0 1.0 1.0
6944 [*,* ,2009]: CO2 NOX CO METHANE := 
6945 NEWZEALAND 1.0 1.0 1.0 1.0
6946 AUSTRALIA 1.0 1.0 1.0 1.0
6947 [*,* ,2010]: CO2 NOX CO METHANE := 
6948 NEWZEALAND 1.0 1.0 1.0 1.0
6949 AUSTRALIA 1.0 1.0 1.0 1.0
6950 ;
6951 #
6952 param ModelPeriodExogenousEmission :CO2 NOX CO METHANE:=
6953 NEWZEALAND 1.0 1.0 1.0 1.0
6954 AUSTRALIA 1.0 1.0 1.0 1.0
6955 ;
6956 #
6957 param ModelPeriodEmissionLimit :CO2 NOX CO METHANE:=
6958 NEWZEALAND 1.0 1.0 1.0 1.0
6959 AUSTRALIA 1.0 1.0 1.0 1.0
6960 ;
6961 end;
6962 #

```

7.3 Linear Programme File

This file is the lp formulation required to use CPLEX. The file is not included in this compendium as the Utopian example is nearly 1.2 million lines of code.

8 Bibliography