

---

# **ElecSim Documentation**

*Release 0.1*

**Alexander J. M. Kell**

**Sep 16, 2020**



---

## Contents:

---

<b>1</b>	<b>elecsim</b>	<b>1</b>
1.1	elecsim package	1
1.1.1	Subpackages	1
1.1.1.1	elecsim.agents package	1
1.1.1.1.1	Subpackages	1
1.1.1.1.1.1	elecsim.agents.demand package	1
1.1.1.1.1.2	Submodules	1
1.1.1.1.1.3	elecsim.agents.demand.demand module	1
1.1.1.1.1.4	Module contents	2
1.1.1.1.1.5	elecsim.agents.generation_company package	2
1.1.1.1.1.6	Subpackages	2
1.1.1.1.1.7	elecsim.agents.generation_company.invest package	2
1.1.1.1.1.8	Submodules	2
1.1.1.1.1.9	elecsim.agents.generation_company.invest.investment module	2
1.1.1.1.1.10	Module contents	2
1.1.1.1.1.11	Submodules	2
1.1.1.1.1.12	elecsim.agents.generation_company.gen_co module	2
1.1.1.1.1.13	Module contents	3
1.1.1.1.2	Module contents	3
1.1.1.2	elecsim.data_manipulation package	3
1.1.1.2.1	Subpackages	3
1.1.1.2.1.1	elecsim.data_manipulation.data_modifications package	3
1.1.1.2.1.2	Submodules	3
1.1.1.2.1.3	elecsim.data_manipulation.data_modifications.extrapolation_interpolate module	3
1.1.1.2.1.4	elecsim.data_manipulation.data_modifications.inverse_transform_sampling module	3
1.1.1.2.1.5	elecsim.data_manipulation.data_modifications.linear_regression module	3
1.1.1.2.1.6	elecsim.data_manipulation.data_modifications.renewable_learning_rate module	3
1.1.1.2.1.7	elecsim.data_manipulation.data_modifications.value_estimations module	4
1.1.1.2.1.8	Module contents	4
1.1.1.2.2	Submodules	4
1.1.1.2.3	elecsim.data_manipulation.import_power_plant_dat module	4

1.1.1.2.4	elecsim.data_manipulation.uk_plant_db_manipulation module . . . . .	4
1.1.1.2.5	Module contents . . . . .	4
1.1.1.3	elecsim.market package . . . . .	4
1.1.1.3.1	Subpackages . . . . .	4
1.1.1.3.1.1	elecsim.market.electricity package . . . . .	4
1.1.1.3.1.2	Submodules . . . . .	4
1.1.1.3.1.3	elecsim.market.electricity.bid module . . . . .	4
1.1.1.3.1.4	elecsim.market.electricity.power_exchange module . . . . .	5
1.1.1.3.1.5	Module contents . . . . .	5
1.1.1.3.2	Module contents . . . . .	5
1.1.1.4	elecsim.mesa_addons package . . . . .	5
1.1.1.4.1	Submodules . . . . .	5
1.1.1.4.2	elecsim.mesa_addons.scheduler_addon module . . . . .	5
1.1.1.4.3	Module contents . . . . .	5
1.1.1.5	elecsim.model package . . . . .	5
1.1.1.5.1	Submodules . . . . .	5
1.1.1.5.2	elecsim.model.world module . . . . .	5
1.1.1.5.3	Module contents . . . . .	7
1.1.1.6	elecsim.plants package . . . . .	7
1.1.1.6.1	Subpackages . . . . .	7
1.1.1.6.1.1	elecsim.plants.fuel package . . . . .	7
1.1.1.6.1.2	Submodules . . . . .	7
1.1.1.6.1.3	elecsim.plants.fuel.fuel module . . . . .	7
1.1.1.6.1.4	Module contents . . . . .	7
1.1.1.6.1.5	elecsim.plants.plant_costs package . . . . .	7
1.1.1.6.1.6	Subpackages . . . . .	7
1.1.1.6.1.7	elecsim.plants.plant_costs.estimate_costs package . . . . .	7
1.1.1.6.1.8	Subpackages . . . . .	7
1.1.1.6.1.9	elecsim.plants.plant_costs.estimate_costs.estimate_modern_power_plant_costs package . . . . .	7
1.1.1.6.1.10	Submodules . . . . .	7
1.1.1.6.1.11	elecsim.plants.plant_costs.estimate_costs.estimate_modern_power_plant_costs.predict_m module . . . . .	7
1.1.1.6.1.12	Module contents . . . . .	8
1.1.1.6.1.13	elecsim.plants.plant_costs.estimate_costs.estimate_old_plant_cost_params package . . . . .	8
1.1.1.6.1.14	Subpackages . . . . .	8
1.1.1.6.1.15	elecsim.plants.plant_costs.estimate_costs.estimate_old_plant_cost_params.fuel_plant_cal package . . . . .	8
1.1.1.6.1.16	Submodules . . . . .	8
1.1.1.6.1.17	elecsim.plants.plant_costs.estimate_costs.estimate_old_plant_cost_params.fuel_plant_cal module . . . . .	8
1.1.1.6.1.18	Module contents . . . . .	8
1.1.1.6.1.19	elecsim.plants.plant_costs.estimate_costs.estimate_old_plant_cost_params.non_fuel_plant package . . . . .	9
1.1.1.6.1.20	Submodules . . . . .	9
1.1.1.6.1.21	elecsim.plants.plant_costs.estimate_costs.estimate_old_plant_cost_params.non_fuel_plant module . . . . .	9
1.1.1.6.1.22	Module contents . . . . .	9
1.1.1.6.1.23	Submodules . . . . .	9
1.1.1.6.1.24	elecsim.plants.plant_costs.estimate_costs.estimate_old_plant_cost_params.old_plant_para module . . . . .	9
1.1.1.6.1.25	Module contents . . . . .	10
1.1.1.6.1.26	Submodules . . . . .	10

1.1.1.6.1.27	elecsim.plants.plant_costs.estimate_costs.estimate_costs module .	10
1.1.1.6.1.28	Module contents . . . . .	11
1.1.1.6.1.29	Module contents . . . . .	11
1.1.1.6.1.30	elecsim.plants.plant_type package . . . . .	11
1.1.1.6.1.31	Submodules . . . . .	11
1.1.1.6.1.32	elecsim.plants.plant_type.fuel_plant module . . . . .	11
1.1.1.6.1.33	elecsim.plants.plant_type.non_fuel_plant module . . . . .	12
1.1.1.6.1.34	elecsim.plants.plant_type.power_plant module . . . . .	12
1.1.1.6.1.35	Module contents . . . . .	13
1.1.1.6.2	Submodules . . . . .	13
1.1.1.6.3	elecsim.plants.plant_registry module . . . . .	13
1.1.1.6.4	Module contents . . . . .	13
1.1.1.7	elecsim.role package . . . . .	13
1.1.1.7.1	Subpackages . . . . .	13
1.1.1.7.1.1	elecsim.role.investment package . . . . .	13
1.1.1.7.1.2	Submodules . . . . .	13
1.1.1.7.1.3	elecsim.role.investment.calculate_npv module . . . . .	13
1.1.1.7.1.4	elecsim.role.investment.predict_load_duration_prices module . . . . .	14
1.1.1.7.1.5	Module contents . . . . .	15
1.1.1.7.1.6	elecsim.role.market package . . . . .	15
1.1.1.7.1.7	Submodules . . . . .	15
1.1.1.7.1.8	elecsim.role.market.latest_market_data module . . . . .	15
1.1.1.7.1.9	elecsim.role.market.world_plant_capacity module . . . . .	15
1.1.1.7.1.10	Module contents . . . . .	15
1.1.1.7.1.11	elecsim.role.plants package . . . . .	17
1.1.1.7.1.12	Subpackages . . . . .	17
1.1.1.7.1.13	elecsim.role.plants.costs package . . . . .	17
1.1.1.7.1.14	Submodules . . . . .	17
1.1.1.7.1.15	elecsim.role.plants.costs.fuel_plant_cost_calculations module . . . . .	17
1.1.1.7.1.16	elecsim.role.plants.costs.non_fuel_cost_calculations module . . . . .	19
1.1.1.7.1.17	elecsim.role.plants.costs.plant_cost_calculation module . . . . .	20
1.1.1.7.1.18	Module contents . . . . .	20
1.1.1.7.1.19	Module contents . . . . .	21
1.1.1.7.2	Module contents . . . . .	21
1.1.1.8	elecsim.scenario package . . . . .	21
1.1.1.8.1	Submodules . . . . .	21
1.1.1.8.2	elecsim.scenario.scenario_data module . . . . .	21
1.1.1.8.3	Module contents . . . . .	21
1.1.1.9	elecsim.server package . . . . .	21
1.1.1.9.1	Module contents . . . . .	21
1.1.2	Module contents . . . . .	21

**2 Indices and tables** **23**

**Python Module Index** **25**

**Index** **27**



## 1.1 elecsim package

### 1.1.1 Subpackages

#### 1.1.1.1 elecsim.agents package

##### 1.1.1.1.1 Subpackages

###### 1.1.1.1.1.1 elecsim.agents.demand package

###### 1.1.1.1.1.2 Submodules

###### 1.1.1.1.1.3 elecsim.agents.demand.demand module

**class** `elecsim.agents.demand.demand.Demand`(*model*, *unique\_id*, *segment\_hours=None*, *segment\_consumption=None*)

Bases: `mesa.agent.Agent`

**step**()

A single step of the agent.

`elecsim.agents.demand.demand.logger` = `<Logger elecsim.agents.demand.demand (WARNING)>`  
demand.py: Agent which simulates the demand of the UK

#### 1.1.1.1.1.4 Module contents

#### 1.1.1.1.1.5 elecsim.agents.generation\_company package

#### 1.1.1.1.1.6 Subpackages

#### 1.1.1.1.1.7 elecsim.agents.generation\_company.invest package

#### 1.1.1.1.1.8 Submodules

#### 1.1.1.1.1.9 elecsim.agents.generation\_company.invest.investment module

#### 1.1.1.1.1.10 Module contents

#### 1.1.1.1.1.11 Submodules

#### 1.1.1.1.1.12 elecsim.agents.generation\_company.gen\_co module

**class** elecsim.agents.generation\_company.gen\_co.**GenCo** (*unique\_id, model, name, difference\_in\_discount\_rate, look\_back\_period, plants=None, money=5000000, rl\_bidding=False*)

Bases: mesa.agent.Agent

**calculate\_bids** (*segment\_hour, predict=False, actions=None*)

Function to generate the bids for each of the power plants owned by the generating company. The bids submitted are the fixed costs divided by lifetime of plant plus yearly variable costs plus a 10% margin :param segment\_hour: Number of hours in which the current segment is required :param segment\_demand: Electricity consumption required for the specified number of hours :return: Bids returned for the available plants at the specified segment hour

**create\_bid**

**delete\_old\_bids** ()

**dismantle\_old\_plants** ()

Remove plants that are past their lifetime agent from plant list

**forecast\_attribute\_price** (*fuel\_type*)

**forecast\_demand\_change** ()

**invest** ()

**invest\_RL** (*action*)

**purchase\_coal** ()

**purchase\_fuel** ()

**purchase\_gas** ()

**settle\_accounts** ()

**step** ()

A single step of the agent.

```
elecsim.agents.generation_company.gen_co.get_renewable_availability
```

```
elecsim.agents.generation_company.gen_co.logger = <Logger elecsim.agents.generation_company
gen_co.py: Agent which represents a generation company
```

#### 1.1.1.1.1.13 Module contents

#### 1.1.1.1.2 Module contents

#### 1.1.1.2 elecsim.data\_manipulation package

##### 1.1.1.2.1 Subpackages

##### 1.1.1.2.1.1 elecsim.data\_manipulation.data\_modifications package

##### 1.1.1.2.1.2 Submodules

##### 1.1.1.2.1.3 elecsim.data\_manipulation.data\_modifications.extrapolation\_interpolate module

```
class elecsim.data_manipulation.data_modifications.extrapolation_interpolate.ExtrapolateInt
```

Bases: object

```
min_max_extrapolate (point)
```

##### 1.1.1.2.1.4 elecsim.data\_manipulation.data\_modifications.inverse\_transform\_sampling module

```
elecsim.data_manipulation.data_modifications.inverse_transform_sampling.sample_from_custom
```

##### 1.1.1.2.1.5 elecsim.data\_manipulation.data\_modifications.linear\_regression module

```
elecsim.data_manipulation.data_modifications.linear_regression.linear_regression
```

```
elecsim.data_manipulation.data_modifications.linear_regression.logger = <Logger elecsim.dat
```

File name: linear\_regression Date created: 29/12/2018 Feature: #Enter feature description here

##### 1.1.1.2.1.6 elecsim.data\_manipulation.data\_modifications.renewable\_learning\_rate module

```
elecsim.data_manipulation.data_modifications.renewable_learning_rate.future_renewable_ener
```

```
elecsim.data_manipulation.data_modifications.renewable_learning_rate.logger = <Logger elec
```

File name: renewable\_learning\_rate Date created: 21/12/2018 Feature: # Contains the functionality for implementing a learning rate for renewable plants to simulate a decrease in prices

### 1.1.1.2.1.7 `elecsim.data_manipulation.data_modifications.value_estimations` module

`elecsim.data_manipulation.data_modifications.value_estimations.closest_row` (*dataframe*,  
*col-*  
*umn*,  
*value*)

Function which takes a dataframe and returns the row that is closest to the specified value of the specified column. :param dataframe: Dataframe object :param column: String which matches to a column in the dataframe in which you would like to find the closest value of. :param value: Value to find the closest row to. :return: Returns row that is closest to the value of the selected column of the dataframe

### 1.1.1.2.1.8 Module contents

#### 1.1.1.2.2 Submodules

#### 1.1.1.2.3 `elecsim.data_manipulation.import_power_plant_dat` module

#### 1.1.1.2.4 `elecsim.data_manipulation.uk_plant_db_manipulation` module

#### 1.1.1.2.5 Module contents

### 1.1.1.3 `elecsim.market` package

#### 1.1.1.3.1 Subpackages

#### 1.1.1.3.1.1 `elecsim.market.electricity` package

#### 1.1.1.3.1.2 Submodules

#### 1.1.1.3.1.3 `elecsim.market.electricity.bid` module

**class** `elecsim.market.electricity.bid.Bid` (*gen\_co*, *plant*, *segment\_hours*, *capacity\_bid*,  
*price\_per\_mwh*, *year\_of\_bid*, *rl\_bid=False*)

Bases: `object`

**accept\_bid** (*segment\_hour*)

Function to be called when bid is accepted :return: None

**partially\_accept\_bid** (*segment\_hour*, *demand\_fulfilled*)

Function to be called when bid is partially accepted :param demand\_fulfilled: :return: None

**reject\_bid** (*segment\_hour*)

Function to be called when bid is rejected :return: None

**to\_dict** ()

`elecsim.market.electricity.bid.logger` = `<Logger elecsim.market.electricity.bid (WARNING)>`

Bid.py: A class which holds information for each bid

#### 1.1.1.3.1.4 elecsim.market.electricity.power\_exchange module

#### 1.1.1.3.1.5 Module contents

File name: `__init__.py` Date created: 17/12/2018 Feature: #Enter feature description here

#### 1.1.1.3.2 Module contents

#### 1.1.1.4 elecsim.mesa\_addons package

##### 1.1.1.4.1 Submodules

##### 1.1.1.4.2 elecsim.mesa\_addons.scheduler\_addon module

**class** `elecsim.mesa_addons.scheduler_addon.OrderedActivation` (*model*)

Bases: `mesa.time.BaseScheduler`

A scheduler which activates each agent in the order that they are added to the scheduler

Assumes that all agents have a `step(model)` method.

**step** ()

Executes the step of all agents, one at a time.

##### 1.1.1.4.3 Module contents

#### 1.1.1.5 elecsim.model package

##### 1.1.1.5.1 Submodules

##### 1.1.1.5.2 elecsim.model.world module

**class** `elecsim.model.world.World` (*initialization\_year*, *scenario\_file=*`None`, *fitting\_params=*`None`, *long\_term\_fitting\_params=*`None`, *future\_price\_uncertainty\_m=*`None`, *future\_price\_uncertainty\_c=*`None`, *carbon\_price\_scenario=*`None`, *demand\_change=*`None`, *demand\_distribution=*`None`, *number\_of\_steps=*`32`, *total\_demand=*`None`, *number\_of\_agents=*`None`, *market\_time\_splices=*`1`, *data\_folder=*`None`, *time\_run=*`False`, *nuclear\_subsidy=*`None`, *highest\_demand=*`None`, *log\_level=*`'warning'`, *client\_rl=*`None`, *distribution\_name=*`None`, *dropbox=*`None`, *gencos\_rl=*`[]`, *write\_data\_to\_file=*`True`, *rl\_port\_number=*`9920`)

Bases: `mesa.model.Model`

Model for the electricity landscape world

**clear\_all\_bids** ()

**create\_data\_loggers** (*data\_folder*)

**delete\_old\_bids** ()

**dismantle\_old\_plants** ()

Remove plants that are past their lifetime agent from each agent from their plant list

**dismantle\_unprofitable\_plants** ()

**filter\_plants\_with\_no\_income** (*plants*)

**static get\_accepted\_bid\_capacity** (*model, plant\_type*)

**static get\_accepted\_bid\_capacity\_per\_segment\_hour** (*model*)

**static get\_accepted\_bids** (*gencos, plant\_type=None*)

**static get\_all\_plants** (*model*)

**static get\_capacity\_of\_plants** (*model, plant\_type*)

**static get\_carbon\_emitted** (*model*)

**static get\_current\_carbon\_tax** (*model*)

**static get\_electricity\_cost** (*model*)

**static get\_genco\_wealth** (*model*)

**get\_gencos** ()

**get\_profitable\_plants** (*plants*)

**get\_running\_plants** (*plants*)

**initialize\_gencos** (*financial\_data, plant\_data, gencos\_rl*)

**Creates generation company agents based on financial data and power plants owned. Estimates cost parameters of each power plant if data not for power plant not available.**

#### Parameters

- **financial\_data** – Data containing information about generation company’s financial status
- **plant\_data** – Data containing information about generation company’s plants owned, start year and name.

**operate\_constructed\_plants** (*minimum\_operation\_year=2018*)

**override\_carbon\_scenario** (*carbon\_price\_scenario*)

**override\_demand\_change** (*demand\_change*)

**override\_highest\_demand** (*highest\_demand*)

**override\_total\_demand** (*total\_demand, number\_of\_agents=None*)

**overwrite\_scenario\_file** (*scenario\_file*)

**set\_log\_level** (*log\_level*)

**settle\_gencos\_financials** ()

**step** (*carbon\_price=None*)

Advance model by one step

**stratify\_data** (*demand*)

**write\_scenario\_data** ()

**write\_timing\_results** (*end, time\_elapased*)

```
elecsim.model.world.logger = <Logger elecsim.model.world (WARNING)>
    Model.py: Model for the electricity landscape world
```

### 1.1.1.5.3 Module contents

### 1.1.1.6 elecsim.plants package

#### 1.1.1.6.1 Subpackages

##### 1.1.1.6.1.1 elecsim.plants.fuel package

##### 1.1.1.6.1.2 Submodules

##### 1.1.1.6.1.3 elecsim.plants.fuel.fuel module

fuel.py: Class which specifies the properties of a fuel

```
class elecsim.plants.fuel.fuel.Fuel (fuel_type, fuel_price, energy_density, co2_density,
                                     mwh_to_co2e_conversion_factor=0.0)
```

Bases: object

##### 1.1.1.6.1.4 Module contents

File name: `__init__.py` Date created: 25/11/2018 Feature: #Enter feature description here

##### 1.1.1.6.1.5 elecsim.plants.plant\_costs package

##### 1.1.1.6.1.6 Subpackages

##### 1.1.1.6.1.7 elecsim.plants.plant\_costs.estimate\_costs package

##### 1.1.1.6.1.8 Subpackages

##### 1.1.1.6.1.9 elecsim.plants.plant\_costs.estimate\_costs.estimate\_modern\_power\_plant\_costs package

##### 1.1.1.6.1.10 Submodules

##### 1.1.1.6.1.11 elecsim.plants.plant\_costs.estimate\_costs.estimate\_modern\_power\_plant\_costs.predict\_modern\_plant\_costs module

```
class elecsim.plants.plant_costs.estimate_costs.estimate_modern_power_plant_costs.predict_modern_plant_costs
```

Bases: object

**check\_plant\_exists** (*parameters\_of\_plant*)

Function which checks that there is data for specified power plant in the modern costs database. :param parameters\_of\_plant: Dictionary of plant which contains all of the values of the estimated power plant.

**static check\_pre\_dev\_spend** (*parameters*)

**parameter\_estimation** ()

Function which estimates costs of power plant based on capacity, plant\_type and start year. Use of linear interpolation for plants of capacity that fall within known range of costing variables. If plant capacity to be calculated falls out of range of known values, the highest or lowest capacity value is chosen. :return (dictstr): Returns dictionary containing variables for PowerPlant cost

### 1.1.1.6.1.12 Module contents

File name: `__init__.py` Date created: 25/11/2018 Feature: #Enter feature description here

### 1.1.1.6.1.13 `elecsim.plants.plant_costs.estimate_costs.estimate_old_plant_cost_params` package

#### 1.1.1.6.1.14 Subpackages

### 1.1.1.6.1.15 `elecsim.plants.plant_costs.estimate_costs.estimate_old_plant_cost_params.fuel_plant_calculations` package

#### 1.1.1.6.1.16 Submodules

### 1.1.1.6.1.17 `elecsim.plants.plant_costs.estimate_costs.estimate_old_plant_cost_params.fuel_plant_calculations.fuel_plant_costs` module

**class** `elecsim.plants.plant_costs.estimate_costs.estimate_old_plant_cost_params.fuel_plant_calculations.fuel_plant_costs`

Bases: `elecsim.plants.plant_costs.estimate_costs.estimate_old_plant_cost_params.estimate_old_plant_param_calc.OldPlantCosts`

**estimate\_cost\_parameters** ()

Function which estimates the parameters of the non-modern fuel power plant using the LCOE value for the relevant year. :return: Returns dictionary of the updated parameters for the power plant.

**estimate\_modern\_parameters** ()

**historic\_fuel\_price** = Unnamed: 0 Fuel Year value 0 0 Coal 1990 0.006353 1 1 Oil 1990 0.006353

### 1.1.1.6.1.18 Module contents

File name: `__init__.py` Date created: 04/12/2018 Feature: #Enter feature description here

**1.1.1.6.1.19** `elecsim.plants.plant_costs.estimate_costs.estimate_old_plant_cost_params.non_fuel_plant_calculati`  
package**1.1.1.6.1.20** Submodules**1.1.1.6.1.21** `elecsim.plants.plant_costs.estimate_costs.estimate_old_plant_cost_params.non_fuel_plant_calculati`  
module**class** `elecsim.plants.plant_costs.estimate_costs.estimate_old_plant_cost_params.non_fuel_pl`Bases: `elecsim.plants.plant_costs.estimate_costs.estimate_old_plant_cost_params.estimate_old_plant_param_calc.OldPlantCosts`**get\_cost\_parameters** ()

Function which estimates the parameters of the non-modern power plant using the LCOE value for the relevant year. :return: Returns dictionary of the updated parameters for the power plant.

**1.1.1.6.1.22** Module contentsFile name: `__init__.py` Date created: 04/12/2018 Feature: #Enter feature description here**1.1.1.6.1.23** Submodules**1.1.1.6.1.24** `elecsim.plants.plant_costs.estimate_costs.estimate_old_plant_cost_params.old_plant_param_calc`  
module**class** `elecsim.plants.plant_costs.estimate_costs.estimate_old_plant_cost_params.old_plant_p`Bases: `object`Class which takes LCOE values and `plant_type` of power plants from retrospective database and predicts more detailed cost parameters using the same proportions as the BEIS Power Plant Cost Database. Specifically uses 2018 power plants from BEIS Power Plant Cost Database.**find\_smallest\_year\_available** ()

Method which takes the modern cost BEIS database of power plants, and finds the earliest year that data for specified power plant type exists. For example, only returns data on Coal power plants from 2025 as only this data is provided in the BEIS datafile :return: Int containing smallest year available.

**get\_params\_to\_scale** ()**hist\_costs** = Unnamed: 0 Technology Discount\_rate capacity\_range Year lcoe 0 0 CCGT 0.

#### 1.1.1.6.1.25 Module contents

#### 1.1.1.6.1.26 Submodules

#### 1.1.1.6.1.27 `elecsim.plants.plant_costs.estimate_costs.estimate_costs` module

`elecsim.plants.plant_costs.estimate_costs.estimate_costs.create_power_plant`

Functionality that estimates the cost of a power plant based solely on year of construction, type of plant and capacity. :param name: Name of power plant. :param start\_date: Starting year of power plant :param simplified\_type: Type of power plant :param capacity: Capacity of power plant :return: Power plant object initialized with relevant variables.

`elecsim.plants.plant_costs.estimate_costs.estimate_costs.create_power_plant_group` (*name*,  
*start\_date*,  
*sim-*  
*pli-*  
*fied\_type*,  
*ca-*  
*pac-*  
*ity*,  
*num-*  
*ber\_of\_plant*)

`elecsim.plants.plant_costs.estimate_costs.estimate_costs.logger` = `<Logger elecsim.plants.plant_costs.estimate_costs>`

File name: `_select_cost_estimator` Date created: 01/12/2018 Feature: # Functionality to estimate costs based on year. If year is past 2018 then use modern data from BEIS file.

# If data is historic, then predict from historic LCOE values, maintaining same ratios from 2018.

#### 1.1.1.6.1.28 Module contents

#### 1.1.1.6.1.29 Module contents

#### 1.1.1.6.1.30 `elecsim.plants.plant_type` package

#### 1.1.1.6.1.31 Submodules

#### 1.1.1.6.1.32 `elecsim.plants.plant_type.fuel_plant` module

**class** `elecsim.plants.plant_type.fuel_plant.FuelPlant` (*name, plant\_type, capacity\_mw, construction\_year, average\_load\_factor, efficiency, pre\_dev\_period, construction\_period, operating\_period, pre\_dev\_spend\_years, construction\_spend\_years, pre\_dev\_cost\_per\_mw, construction\_cost\_per\_mw, infrastructure, fixed\_o\_and\_m\_per\_mw, variable\_o\_and\_m\_per\_mwh, insurance\_cost\_per\_mw, connection\_cost\_per\_mw*)

Bases: `elecsim.plants.plant_type.power_plant.PowerPlant`

**calculate\_lcoe** (*discount\_rate*)

**short\_run\_marginal\_cost** (*model, genco, fuel\_price=None, co2\_price=None*)

`elecsim.plants.plant_type.fuel_plant.logger` = `<Logger elecsim.plants.plant_type.fuel_plant.fuel_plant.py>`: Child class of power plant which contains functions for a power plant which consumes fuel.

#### 1.1.1.6.1.33 `elecsim.plants.plant_type.non_fuel_plant` module

```
class elecsim.plants.plant_type.non_fuel_plant.NonFuelPlant (name, plant_type,
construction_year,
capacity_mw, average_load_factor,
pre_dev_period,
construction_period,
operating_period,
pre_dev_spend_years,
construction_spend_years,
pre_dev_cost_per_mw,
construction_cost_per_mw,
infrastructure,
fixed_o_and_m_per_mw,
variable_o_and_m_per_mwh,
insurance_cost_per_mw,
connection_cost_per_mw,
efficiency)
```

Bases: `elecsim.plants.plant_type.power_plant.PowerPlant`

`calculate_lcoe` (*discount\_rate*)

`short_run_marginal_cost`

#### 1.1.1.6.1.34 `elecsim.plants.plant_type.power_plant` module

`power_plant.py`: Class which represents a Power Plant

```
class elecsim.plants.plant_type.power_plant.PowerPlant (name, plant_type, capacity_mw,
construction_year,
average_load_factor,
pre_dev_period,
construction_period,
operating_period,
pre_dev_spend_years,
construction_spend_years,
pre_dev_cost_per_mw,
construction_cost_per_mw,
infrastructure,
fixed_o_and_m_per_mw,
variable_o_and_m_per_mwh,
insurance_cost_per_mw,
connection_cost_per_mw)
```

Bases: `abc.ABC`

`check_if_operating_in_certain_year` (*current\_year, year\_difference\_from\_model\_year*)

`delete_old_plant_bids` ()

```

get_fixed_annual_payments ()
get_upfront_costs ()
get_year_of_operation ()
short_run_marginal_cost (model, genco, fuel_price, co2_price)

```

#### 1.1.1.6.1.35 Module contents

File name: `__init__.py` Date created: 25/11/2018 Feature: #Enter feature description here

#### 1.1.1.6.2 Submodules

##### 1.1.1.6.3 `elecsim.plants.plant_registry` module

```

class elecsim.plants.plant_registry.PlantRegistry (plant_type)
    Bases: object

    check_if_fuel_required ()
        Takes a plant_type type and returns a boolean specifying whether the power plant uses fuel or not. :param
        plant_type: Type of plant :return: Boolean specifying whether plant requires plant_type or not

    plant_type_to_plant_object ()

```

#### 1.1.1.6.4 Module contents

#### 1.1.1.7 `elecsim.role` package

##### 1.1.1.7.1 Subpackages

##### 1.1.1.7.1.1 `elecsim.role.investment` package

##### 1.1.1.7.1.2 Submodules

##### 1.1.1.7.1.3 `elecsim.role.investment.calculate_npv` module

```

class elecsim.role.investment.calculate_npv.CalculateNPV (model,
                                                         difference_in_discount_rate,
                                                         look_back_years,
                                                         price_curve_parameters=None)

    Bases: object

    calculate_npv (plant_type, plant_size)
    compare_npv ()
    fit_price_curve ()
    get_positive_npv_plants ()
    get_positive_npv_plants_list ()

```

```
elecsim.role.investment.calculate_npv.get_most_profitable_plants_by_npv(model,  
                                                                    dif-  
                                                                    fer-  
                                                                    ence_in_discount_rate,  
                                                                    look_back_period)
```

```
elecsim.role.investment.calculate_npv.get_yearly_payment(power_plant,  
                                                         interest_rate,  
                                                         down_payment_perc)
```

```
elecsim.role.investment.calculate_npv.logger = <Logger elecsim.role.investment.calculate_npv  
File name: calculate_npv Date created: 04/01/2019 Feature: # Contains functionality to assess options of investment and return lowest NPV for decision to be made.
```

```
elecsim.role.investment.calculate_npv.select_yearly_payback_payment_for_year(power_plant,  
                                                                              in-  
                                                                              ter-  
                                                                              est,  
                                                                              down-  
                                                                              pay-  
                                                                              ment_percentage,  
                                                                              model)
```

#### 1.1.1.7.1.4 elecsim.role.investment.predict\_load\_duration\_prices module

```
class elecsim.role.investment.predict_load_duration_prices.PredictPriceDurationCurve(model,  
                                                                                   look_ba
```

Bases: object

```
fit_linear_price_duration_curve(year_segment_consumption_predicted,  
                                year_segment_hours)
```

```
fit_linear_price_duration_curve_from_projections(year_segment_consumption_predicted,  
                                                  year_segment_hours)
```

```
market_price_curve_prediction(year_segment_consumption_predicted,  
                               year_segment_hours)
```

```
predict_price_duration_curve()
```

```
elecsim.role.investment.predict_load_duration_prices.estimate_lost_load_price(predicted_price_d
```

```
elecsim.role.investment.predict_load_duration_prices.get_price_duration_curve(model,  
                                                                              look_back_period)
```

```
elecsim.role.investment.predict_load_duration_prices.logger = <Logger elecsim.role.investment  
File name: predict_load_duration_prices Date created: 11/01/2019 Feature: # Predict Load Duration Prices
```

### 1.1.1.7.1.5 Module contents

### 1.1.1.7.1.6 elecsim.role.market package

### 1.1.1.7.1.7 Submodules

### 1.1.1.7.1.8 elecsim.role.market.latest\_market\_data module

**class** `elecsim.role.market.latest_market_data.LatestMarketData` (*model*)

Bases: `object`

**agent\_forecast\_value** (*value\_required, years\_to\_look\_back, years\_to\_look\_forward=None, demand\_linear=False*)

**fit\_exponential\_function** (*regression, years\_to\_look\_back, years\_to\_look\_forward*)

**get\_RL\_investment\_observations** ()

**get\_predicted\_marginal\_cost** (*power\_plant, look\_back\_years*)

`elecsim.role.market.latest_market_data.logger` = `<Logger elecsim.role.market.latest_market_data>`

File name: `current_market_data` Date created: 29/12/2018 Feature: # Data which collates market information for generator companies when making decisions about investments.

### 1.1.1.7.1.9 elecsim.role.market.world\_plant\_capacity module

**class** `elecsim.role.market.world_plant_capacity.WorldPlantCapacity` (*model*)

Bases: `object`

**get\_current\_total\_plant\_capacity** ()

**get\_power\_plants\_running\_in\_current\_year** ()

**get\_power\_plants\_running\_in\_reference\_year** (*reference\_year*)

**get\_reference\_year\_total\_capacity** (*reference\_year*)

**get\_renewable\_by\_type** (*type*)

**get\_renewable\_plants** ()

`elecsim.role.market.world_plant_capacity.logger` = `<Logger elecsim.role.market.world_plant_capacity>`

File name: `plant_capacity` Date created: 30/12/2018 Feature: # Functionality that aggregates all of the power plants in operation.

### 1.1.1.7.1.10 Module contents

File name: `__init__.py` Date created: 30/12/2018 Feature: #Enter feature description here



#### 1.1.1.7.1.11 `elecsim.role.plants` package

#### 1.1.1.7.1.12 Subpackages

#### 1.1.1.7.1.13 `elecsim.role.plants.costs` package

#### 1.1.1.7.1.14 Submodules

#### 1.1.1.7.1.15 `elecsim.role.plants.costs.fuel_plant_cost_calculations` module

**class** `elecsim.role.plants.costs.fuel_plant_cost_calculations.FuelPlantCostCalculations` (*plant*

*ca-*  
*pac-*  
*ity\_m*  
*con-*  
*struc*  
*tion\_*  
*av-*  
*er-*  
*age\_*  
*ef-*  
*fi-*  
*cienc*  
*pre\_c*  
*con-*  
*struc*  
*tion\_*  
*op-*  
*er-*  
*at-*  
*ing\_p*  
*pre\_c*  
*con-*  
*struc*  
*tion\_*  
*pre\_c*  
*con-*  
*struc*  
*tion\_*  
*in-*  
*fras-*  
*truc-*  
*ture,*  
*fixed*  
*vari-*  
*able\_*  
*in-*  
*sur-*  
*ance*  
*con-*  
*nec-*  
*tion\_*

Bases: `elecsim.role.plants.costs.plant_cost_calculation.`

*PlantCostCalculations*

**calculate\_lcoe** (*discount\_rate*)

Function which calculates the levelised cost of electricity for this power plant instance at a specified discount rate.

**Parameters** **discount\_rate** – The discount rate that is used for the calculation of the levelised cost of electricity.

**Returns** Returns LCOE value for power plant

**calculate\_short\_run\_marginal\_cost** (*model, genco, fuel\_price=None, co2\_price=None*)

Calculates the short run marginal cost for a fuel power plant ;param model: Model containing information such as current year ;param genco: Generation company object that requires short run marginal cost. Used to use genco price of fuel. :return: returns marginal cost to burn 1MWh of fuel.

**calculate\_total\_costs** (*estimated\_fuel\_prices=None, estimated\_carbon\_prices=None*)

**get\_BEIS\_carbon\_price** ()

**get\_co2\_price** (*co2\_price, model*)

**get\_fuel\_price** (*fuel\_price, year\_number, modifier*)

`elecsim.role.plants.costs.fuel_plant_cost_calculations.calculate_year_carbon_price()`

`elecsim.role.plants.costs.fuel_plant_cost_calculations.get_carbon_cost_in_year`

`elecsim.role.plants.costs.fuel_plant_cost_calculations.logger = <Logger elecsim.role.plants`

File name: fuel\_lcoe\_calculation Date created: 18/12/2018 Feature: #Enter feature description here

1.1.1.7.1.16 `elecsim.role.plants.costs.non_fuel_cost_calculations` module

```

class elecsim.role.plants.costs.non_fuel_cost_calculations.NonFuelCostCalculation (construction
    ca-
    pac-
    ity_mw,
    av-
    er-
    age_load_fa
    pre_dev_per
    con-
    struc-
    tion_period,
    op-
    er-
    at-
    ing_period,
    pre_dev_spe
    con-
    struc-
    tion_spend_
    pre_dev_cos
    con-
    struc-
    tion_cost_pe
    in-
    fras-
    truc-
    ture,
    fixed_o_and
    vari-
    able_o_and
    in-
    sur-
    ance_cost_p
    con-
    nec-
    tion_cost_pe
    ef-
    fi-
    ciency)

```

Bases: `elecsim.role.plants.costs.plant_cost_calculation.PlantCostCalculations`

**calculate\_lcoe** (*discount\_rate*)

Function which calculates the levelised cost of electricity for this power plant instance :return: Returns LCOE value for power plant

**calculate\_short\_run\_marginal\_cost** (*model*)

**calculate\_total\_costs** ()

#### 1.1.1.7.1.17 `elecsim.role.plants.costs.plant_cost_calculation` module

```
class elecsim.role.plants.costs.plant_cost_calculation.PlantCostCalculations(capacity_mw,  
con-  
struc-  
tion_year,  
av-  
er-  
age_load_factor,  
pre_dev_period,  
con-  
struc-  
tion_period,  
op-  
er-  
at-  
ing_period,  
pre_dev_spend_year,  
con-  
struc-  
tion_spend_years,  
pre_dev_cost_per_mw,  
con-  
struc-  
tion_cost_per_mw,  
in-  
fras-  
truc-  
ture,  
fixed_o_and_m_per_mw,  
vari-  
able_o_and_m_per_mw,  
in-  
sur-  
ance_cost_per_mw,  
con-  
nec-  
tion_cost_per_mw)
```

Bases: `object`

**calculate\_yearly\_outflow**()

**infrastructure**

**total\_income**(*expected\_sale\_price*)

```
elecsim.role.plants.costs.plant_cost_calculation.logger = <Logger elecsim.role.plants.costs.plant_cost_calculation>  
File name: calculate_lcoe Date created: 18/12/2018 Feature: #Enter feature description here
```

#### 1.1.1.7.1.18 Module contents

File name: `__init__.py` Date created: 30/12/2018 Feature: #Enter feature description here

#### 1.1.1.7.1.19 Module contents

File name: `__init__.py` Date created: 30/12/2018 Feature: #Enter feature description here

#### 1.1.1.7.2 Module contents

### 1.1.1.8 elecsim.scenario package

#### 1.1.1.8.1 Submodules

#### 1.1.1.8.2 elecsim.scenario.scenario\_data module

```
elecsim.scenario.scenario_data.concatenate_carbon_price()
```

#### 1.1.1.8.3 Module contents

File name: `__init__.py` Date created: 25/11/2018 Feature: #Enter feature description here

### 1.1.1.9 elecsim.server package

#### 1.1.1.9.1 Module contents

## 1.1.2 Module contents



## CHAPTER 2

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



---

## Python Module Index

---

### e

elecsim, 21

elecsim.agents, 3

elecsim.agents.demand, 2

elecsim.agents.demand.demand, 1

elecsim.agents.generation\_company, 3

elecsim.agents.generation\_company.gen\_co,  
2

elecsim.agents.generation\_company.invest,  
2

elecsim.agents.generation\_company.invest.investment,  
2

elecsim.data\_manipulation, 4

elecsim.data\_manipulation.data\_modifications,  
4

elecsim.data\_manipulation.data\_modifications.extrapolation\_interpolate,  
3

elecsim.data\_manipulation.data\_modifications.inverse\_transform\_sampling,  
3

elecsim.data\_manipulation.data\_modifications.linear\_regression,  
3

elecsim.data\_manipulation.data\_modifications.renewable\_learning\_rate,  
3

elecsim.data\_manipulation.data\_modifications.value\_estimations,  
4

elecsim.market, 5

elecsim.market.electricity, 5

elecsim.market.electricity.bid, 4

elecsim.mesa\_addons, 5

elecsim.mesa\_addons.scheduler\_addon, 5

elecsim.model, 7

elecsim.model.world, 5

elecsim.plants, 13

elecsim.plants.fuel, 7

elecsim.plants.fuel.fuel, 7

elecsim.plants.plant\_costs, 11

elecsim.plants.plant\_costs.estimate\_costs,  
11

elecsim.plants.plant\_costs.estimate\_costs.estimate\_costs.estimate\_costs,  
10

elecsim.plants.plant\_costs.estimate\_costs.estimate\_costs.estimate\_costs.estimate\_costs,  
8

elecsim.plants.plant\_costs.estimate\_costs.estimate\_costs.estimate\_costs.estimate\_costs.estimate\_costs,  
7

elecsim.plants.plant\_costs.estimate\_costs.estimate\_costs.estimate\_costs.estimate\_costs.estimate\_costs.estimate\_costs,  
10

elecsim.plants.plant\_costs.estimate\_costs.estimate\_costs.estimate\_costs.estimate\_costs.estimate\_costs.estimate\_costs.estimate\_costs,  
8

elecsim.plants.plant\_costs.estimate\_costs.estimate\_costs.estimate\_costs.estimate\_costs.estimate\_costs.estimate\_costs.estimate\_costs.estimate\_costs,  
8

elecsim.plants.plant\_costs.estimate\_costs.estimate\_costs.estimate\_costs.estimate\_costs.estimate\_costs.estimate\_costs.estimate\_costs.estimate\_costs.estimate\_costs,  
9

elecsim.plants.plant\_costs.estimate\_costs.estimate\_costs.estimate\_costs.estimate\_costs.estimate\_costs.estimate\_costs.estimate\_costs.estimate\_costs.estimate\_costs.estimate\_costs,  
9

elecsim.plants.plant\_costs.estimate\_costs.estimate\_costs.estimate\_costs.estimate\_costs.estimate\_costs.estimate\_costs.estimate\_costs.estimate\_costs.estimate\_costs.estimate\_costs.extrapolation\_interpolate,  
9

elecsim.plants.plant\_registry, 13

elecsim.plants.plant\_type, 13

elecsim.plants.plant\_type.fuel\_plant,  
11

elecsim.plants.plant\_type.non\_fuel\_plant,  
12

elecsim.plants.plant\_type.power\_plant,  
12

elecsim.role, 21

elecsim.role.investment, 15

elecsim.role.investment.calculate\_npv,  
13

elecsim.role.investment.predict\_load\_duration\_price,  
14

elecsim.role.market, 15

elecsim.role.market.latest\_market\_data,  
15

elecsim.role.market.world\_plant\_capacity,  
15

elecsim.role.plants, 21

elecsim.role.plants.costs, 20

elecsim.role.plants.costs.fuel\_plant\_cost\_calculator,  
17

`elecsim.role.plants.costs.non_fuel_cost_calculations,`  
    [19](#)  
`elecsim.role.plants.costs.plant_cost_calculation,`  
    [20](#)  
`elecsim.scenario,` [21](#)  
`elecsim.scenario.scenario_data,` [21](#)

**A**

`accept_bid()` (*elecsim.market.electricity.bid.Bid* method), 4  
`agent_forecast_value()` (*elecsim.role.market.latest\_market\_data.LatestMarketData* method), 15  
`calculate_total_costs()` (*elecsim.role.plants.costs.non\_fuel\_cost\_calculations.NonFuelCostCalculations* method), 19  
`calculate_year_carbon_price()` (*in module elecsim.role.plants.costs.fuel\_plant\_cost\_calculations*), 18

**B**

`Bid` (class in *elecsim.market.electricity.bid*), 4  
`calculate_yearly_outflow()` (*elecsim.role.plants.costs.plant\_cost\_calculation.PlantCostCalculation* method), 20

**C**

`CalculateNPV` (class in *elecsim.role.investment.calculate\_npv*), 13  
`calculate_bids()` (*elecsim.sim.agents.generation\_company.gen\_co.GenCompany* method), 2  
`calculate_lcoe()` (*elecsim.sim.plants.plant\_type.fuel\_plant.FuelPlant* method), 11  
`calculate_lcoe()` (*elecsim.sim.plants.plant\_type.non\_fuel\_plant.NonFuelPlant* method), 12  
`calculate_lcoe()` (*elecsim.sim.role.plants.costs.fuel\_plant\_cost\_calculations.FuelPlantCostCalculations* method), 18  
`calculate_lcoe()` (*elecsim.sim.role.plants.costs.non\_fuel\_cost\_calculations.NonFuelCostCalculations* method), 19  
`calculate_npv()` (*elecsim.sim.role.investment.calculate\_npv.CalculateNPV* method), 13  
`calculate_short_run_marginal_cost()` (*elecsim.role.plants.costs.fuel\_plant\_cost\_calculations.FuelPlantCostCalculations* method), 18  
`calculate_short_run_marginal_cost()` (*elecsim.role.plants.costs.non\_fuel\_cost\_calculations.NonFuelCostCalculations* method), 19  
`calculate_total_costs()` (*elecsim.sim.role.plants.costs.fuel\_plant\_cost\_calculations.FuelPlantCostCalculations* method), 18  
`check_if_fuel_required()` (*elecsim.sim.plants.plant\_registry.PlantRegistry* method), 13  
`check_if_operating_in_certain_year()` (*elecsim.sim.plants.plant\_type.power\_plant.PowerPlant* method), 12  
`check_plant_exists()` (*elecsim.sim.plants.plant\_costs.estimate\_costs.estimate\_modern\_power\_plant* method), 7  
`check_pre_dev_spend()` (*elecsim.sim.plants.plant\_costs.estimate\_costs.estimate\_modern\_power\_plant* static method), 8  
`clear_all_bids()` (*elecsim.model.world.World* method), 5  
`closest_row()` (*in module elecsim.data\_manipulation.data\_modifications.value\_estimations*), 4  
`compare_npv()` (*elecsim.sim.role.investment.calculate\_npv.CalculateNPV* method), 13  
`concatenate_carbon_price()` (*in module elecsim.sim.scenario.scenario\_data*), 21  
`create_bid()` (*elecsim.sim.agents.generation\_company.gen\_co.GenCompany* attribute), 2  
`create_data_loggers()` (*elecsim.sim.model.world.World* method), 5  
`create_power_plant` (*in module elecsim.sim.plants.plant\_costs.estimate\_costs.estimate\_costs*),

10  
 create\_power\_plant\_group() (in module *elec-  
 sim.plants.plant\_costs.estimate\_costs.estimate\_costs*),  
 10

**D**

delete\_old\_bids() (*elec-  
 sim.agents.generation\_company.gen\_co.GenCo*  
*method*), 2

delete\_old\_bids() (*elecsim.model.world.World*  
*method*), 5

delete\_old\_plant\_bids() (*elec-  
 sim.plants.plant\_type.power\_plant.PowerPlant*  
*method*), 12

Demand (class in *elecsim.agents.demand.demand*), 1

dismantle\_old\_plants() (*elec-  
 sim.agents.generation\_company.gen\_co.GenCo*  
*method*), 2

dismantle\_old\_plants() (*elec-  
 sim.model.world.World* *method*), 5

dismantle\_unprofitable\_plants() (*elec-  
 sim.model.world.World* *method*), 6

**E**

elecsim (module), 21

elecsim.agents (module), 3

elecsim.agents.demand (module), 2

elecsim.agents.demand.demand (module), 1

elecsim.agents.generation\_company (mod-  
 ule), 3

elecsim.agents.generation\_company.gen\_co  
 (module), 2

elecsim.agents.generation\_company.invest  
 (module), 2

elecsim.agents.generation\_company.invest  
 (module), 2

elecsim.data\_manipulation (module), 4

elecsim.data\_manipulation.data\_modifications  
 (module), 4

elecsim.data\_manipulation.data\_modifications.  
 extrapolation\_interpolate  
 (module), 3

elecsim.data\_manipulation.data\_modifications.  
 inverse\_transform\_sampling  
 (module), 3

elecsim.data\_manipulation.data\_modifications.  
 linear\_regression  
 (module), 3

elecsim.data\_manipulation.data\_modifications.  
 renewable\_learning\_rate  
 (module), 3

elecsim.data\_manipulation.data\_modifications.  
 value\_estimations  
 (module), 4

elecsim.market (module), 5

elecsim.market.electricity (module), 5

elecsim.market.electricity.bid (module), 4

elecsim.mesa\_addons (module), 5

elecsim.mesa\_addons.scheduler\_addon  
 (module), 5

elecsim.model (module), 7

elecsim.model.world (module), 5

elecsim.plants (module), 13

elecsim.plants.fuel (module), 7

elecsim.plants.fuel.fuel (module), 7

elecsim.plants.plant\_costs (module), 11

elecsim.plants.plant\_costs.estimate\_costs  
 (module), 11

elecsim.plants.plant\_costs.estimate\_costs.estimate  
 (module), 10

elecsim.plants.plant\_costs.estimate\_costs.estimate  
 (module), 8

elecsim.plants.plant\_costs.estimate\_costs.estimate  
 (module), 7

elecsim.plants.plant\_costs.estimate\_costs.estimate  
 (module), 10

elecsim.plants.plant\_costs.estimate\_costs.estimate  
 (module), 8

elecsim.plants.plant\_costs.estimate\_costs.estimate  
 (module), 8

elecsim.plants.plant\_costs.estimate\_costs.estimate  
 (module), 9

elecsim.plants.plant\_costs.estimate\_costs.estimate  
 (module), 9

elecsim.plants.plant\_costs.estimate\_costs.estimate  
 (module), 9

elecsim.plants.plant\_registry (module), 13

elecsim.plants.plant\_type (module), 13

elecsim.plants.plant\_type.fuel\_plant  
 (module), 11

elecsim.plants.plant\_type.non\_fuel\_plant  
 (module), 12

elecsim.plants.plant\_type.power\_plant  
 (module), 12

elecsim.role (module), 21

elecsim.role.investment (module), 15

elecsim.role.investment.calculate\_npv  
 (module), 13

elecsim.role.investment.predict\_load\_duration\_price  
 (module), 14

elecsim.role.investment.transform\_sampling  
 (module), 15

elecsim.role.market (module), 15

elecsim.role.market.latest\_market\_data  
 (module), 15

elecsim.role.market.world\_plant\_capacity  
 (module), 15

elecsim.role.plants (module), 21

elecsim.role.plants.costs (module), 20

elecsim.role.plants.costs.fuel\_plant\_cost\_calculat  
 (module), 17

elecsim.role.plants.costs.non\_fuel\_cost\_calculatio  
 (module), 19

elecsim.role.plants.costs.plant\_cost\_calculation

(*module*), 20

elecsim.scenario (*module*), 21

elecsim.scenario.scenario\_data (*module*), 21

estimate\_cost\_parameters() (*elec-sim.plants.plant\_costs.estimate\_costs.estimate\_old\_plant\_cost\_params.fuel\_plant\_calculations.fuel\_plants\_old\_params.FuelPlantCostCalculations* static method), 8

estimate\_lost\_load\_price() (*in module elec-sim.role.investment.predict\_load\_duration\_prices*), 14

estimate\_modern\_parameters() (*elec-sim.plants.plant\_costs.estimate\_costs.estimate\_old\_plant\_cost\_params.fuel\_plant\_calculations.fuel\_plants\_old\_params.FuelPlantCostCalculations* static method), 8

ExtrapolateInterpolate (*class in elec-sim.data\_manipulation.data\_modifications.extrapolation\_interpolation*), 3

## F

filter\_plants\_with\_no\_income() (*elec-sim.model.world.World* method), 6

find\_smallest\_year\_available() (*elec-sim.plants.plant\_costs.estimate\_costs.estimate\_old\_plant\_cost\_params.fuel\_plant\_calculations.fuel\_plants\_old\_params.FuelPlantCostCalculations* static method), 9

fit\_exponential\_function() (*elec-sim.role.market.latest\_market\_data.LatestMarketData* method), 15

fit\_linear\_price\_duration\_curve() (*elec-sim.role.investment.predict\_load\_duration\_prices.PredictPriceDurationCurve* method), 14

fit\_linear\_price\_duration\_curve\_from\_projections() (*elec-sim.role.investment.predict\_load\_duration\_prices.PredictPriceDurationCurve* method), 14

fit\_price\_curve() (*elec-sim.role.investment.calculate\_npv.CalculateNPV* method), 13

forecast\_attribute\_price() (*elec-sim.agents.generation\_company.gen\_co.GenCo* method), 2

forecast\_demand\_change() (*elec-sim.agents.generation\_company.gen\_co.GenCo* method), 2

Fuel (*class in elec-sim.plants.fuel.fuel*), 7

FuelOldPlantCosts (*class in elec-sim.plants.plant\_costs.estimate\_costs.estimate\_old\_plant\_cost\_params.fuel\_plant\_calculations.fuel\_plants\_old\_params.FuelPlantCostCalculations* static method), 8

FuelPlant (*class in elec-sim.plants.plant\_type.fuel\_plant*), 11

FuelPlantCostCalculations (*class in elec-sim.role.plants.costs.fuel\_plant\_cost\_calculations*), 17

future\_renewable\_energy\_costs() (*in module elec-sim.data\_manipulation.data\_modifications.renewable\_learning\_rate*), 3

## G

GenCo (*class in elec-sim.agents.generation\_company.gen\_co*), 2

get\_accepted\_bid\_capacity() (*elec-sim.model.world.World* static method), 6

get\_accepted\_bid\_capacity\_per\_segment\_hour() (*elec-sim.model.world.World* static method), 6

get\_accepted\_bids() (*elec-sim.model.world.World* static method), 6

get\_all\_plants() (*elec-sim.model.world.World* static method), 6

get\_BEIS\_carbon\_price() (*elec-sim.role.plants.costs.fuel\_plant\_cost\_calculations.FuelPlantCostCalculations* method), 18

get\_capacity\_of\_plants() (*elec-sim.model.world.World* static method), 6

get\_carbon\_cost\_in\_year (*in module elec-sim.role.plants.costs.fuel\_plant\_cost\_calculations*), 18

get\_carbon\_emitted() (*elec-sim.model.world.World* static method), 6

get\_co2\_price() (*elec-sim.role.plants.costs.fuel\_plant\_cost\_calculations.FuelPlantCostCalculations* method), 18

get\_cost\_parameters() (*elec-sim.plants.plant\_costs.estimate\_costs.estimate\_old\_plant\_cost\_params.fuel\_plant\_calculations.fuel\_plants\_old\_params.FuelPlantCostCalculations* static method), 8

get\_current\_carbon\_tax() (*elec-sim.model.world.World* static method), 6

get\_current\_total\_plant\_capacity() (*elec-sim.role.market.world\_plant\_capacity.WorldPlantCapacity* method), 15

get\_electricity\_cost() (*elec-sim.model.world.World* static method), 6

get\_fixed\_annual\_payments() (*elec-sim.plants.plant\_type.power\_plant.PowerPlant* method), 12

get\_fuel\_price() (*elec-sim.role.plants.costs.fuel\_plant\_cost\_calculations.FuelPlantCostCalculations* method), 18

get\_genco\_wealth() (*elec-sim.model.world.World* static method), 6

get\_generation() (*elec-sim.model.world.World* static method), 6

get\_most\_profitable\_plants\_by\_npv() (*in module elec-sim.role.investment.calculate\_npv*), 13

get\_params\_to\_scale() (*elec-sim.plants.plant\_costs.estimate\_costs.estimate\_old\_plant\_cost\_params.fuel\_plant\_calculations.fuel\_plants\_old\_params.FuelPlantCostCalculations* static method), 9

get\_positive\_npv\_plants() (*elec-sim.role.investment.calculate\_npv.CalculateNPV* method), 13

`get_positive_npv_plants_list()` (*elec- initialize\_gencos()* (*elecsim.model.world.World*  
*sim.role.investment.calculate\_npv.CalculateNPV* method), 6  
method), 13  
`get_power_plants_running_in_current_year()` *invest()* (*elecsim.agents.generation\_company.gen\_co.GenCo*  
(*elecsim.role.market.world\_plant\_capacity.WorldPlantCapacity*) (*elecsim.agents.generation\_company.gen\_co.GenCo*  
method), 15 method), 2  
`get_power_plants_running_in_reference_year()` *LatestMarketData* (class in *elec-*  
(*elecsim.role.market.world\_plant\_capacity.WorldPlantCapacity* LatestMarketData (class in *elec-*  
method), 15 *sim.role.market.latest\_market\_data*), 15  
`get_predicted_marginal_cost()` (*elec- linear\_regression* (in module *elec-*  
*sim.role.market.latest\_market\_data.LatestMarketData* *sim.data\_manipulation.data\_modifications.linear\_regression*),  
method), 15 3  
`get_price_duration_curve()` (in module *elec- logger* (in module *elecsim.agents.demand.demand*), 1  
*sim.role.investment.predict\_load\_duration\_prices*), 14 logger (in module *elec-*  
3  
`get_profitable_plants()` (*elec- sim.agents.generation\_company.gen\_co*),  
*sim.model.world.World* method), 6 3  
`get_reference_year_total_capacity()` logger (in module *elec-*  
(*elecsim.role.market.world\_plant\_capacity.WorldPlantCapacity* *sim.data\_manipulation.data\_modifications.linear\_regression*),  
method), 15 3  
`get_renewable_availability` (in module *elec- logger* (in module *elec-*  
*sim.agents.generation\_company.gen\_co*), 3 *sim.data\_manipulation.data\_modifications.renewable\_learning\_r*  
3  
`get_renewable_by_type()` (*elec- 3*  
*sim.role.market.world\_plant\_capacity.WorldPlantCapacity* logger (in module *elecsim.market.electricity.bid*), 4  
method), 15 logger (in module *elecsim.model.world*), 6  
`get_renewable_plants()` (*elec- logger* (in module *elec-*  
*sim.role.market.world\_plant\_capacity.WorldPlantCapacity* *sim.plants.plant\_costs.estimate\_costs.estimate\_costs*),  
method), 15 10  
`get_RL_investment_observations()` (*elec- logger* (in module *elec-*  
*sim.role.market.latest\_market\_data.LatestMarketData* *sim.plants.plant\_type.fuel\_plant*), 11  
method), 15 logger (in module *elec-*  
6  
`get_running_plants()` (*elec- sim.role.investment.calculate\_npv*), 14  
*sim.model.world.World* method), 6 logger (in module *elec-*  
14  
`get_upfront_costs()` (*elec- sim.role.investment.predict\_load\_duration\_prices*),  
*sim.plants.plant\_type.power\_plant.PowerPlant* method), 13 14  
`get_year_of_operation()` (*elec- logger* (in module *elec-*  
*sim.plants.plant\_type.power\_plant.PowerPlant* *sim.role.market.latest\_market\_data*), 15  
method), 13 logger (in module *elec-*  
15  
`get_yearly_payment()` (in module *elec- logger* (in module *elec-*  
*sim.role.investment.calculate\_npv*), 14 *sim.role.plants.costs.fuel\_plant\_cost\_calculations*),  
18  
**H** logger (in module *elec-*  
`hist_costs` (*elecsim.plants.plant\_costs.estimate\_costs.estimate\_old* *sim.plants.plant\_costs.plant\_cost\_calculations.OldPlantCosts*  
attribute), 9 20  
`historic_fuel_price` (*elec- M*  
*sim.plants.plant\_costs.estimate\_costs.estimate\_old* *plant\_cost\_params.fuel\_plant\_calculations.fuel\_plants\_old\_params.Fuel*  
attribute), 8 market\_price\_curve\_prediction() (*elec-*  
**I** *sim.role.investment.predict\_load\_duration\_prices.PredictPriceDu*  
method), 14  
`infrastructure` (*elec- min\_max\_extrapolate()* (*elec-*  
*sim.role.plants.costs.plant\_cost\_calculation.PlantCostCalculation* *sim.data\_manipulation.data\_modifications.extrapolation\_interpo*  
attribute), 20 method), 3

## N

NonFuelCostCalculation (class in *elec-sim.role.plants.costs.non\_fuel\_cost\_calculations*), 19

NonFuelOldPlantCosts (class in *elec-sim.plants.plant\_costs.estimate\_costs.estimate\_old\_plant\_costs\_params.non\_fuel\_plant\_calculations.non\_fuel\_plants\_old\_params*), 9

NonFuelPlant (class in *elec-sim.plants.plant\_type.non\_fuel\_plant*), 12

*elec-sim.agents.generation\_company.gen\_co.GenCo*.purchase\_coal() (method), 2

*elec-sim.agents.generation\_company.gen\_co.GenCo*.purchase\_fuel() (method), 2

*elec-sim.agents.generation\_company.gen\_co.GenCo*.purchase\_gas() (method), 2

## O

OldPlantCosts (class in *elec-sim.plants.plant\_costs.estimate\_costs.estimate\_old\_plant\_costs\_params.old\_plant\_param\_calc*), 9

operate\_constructed\_plants() (*elec-sim.model.world.World* method), 6

OrderedActivation (class in *elec-sim.mesa\_addons.scheduler\_addon*), 5

override\_carbon\_scenario() (*elec-sim.model.world.World* method), 6

override\_demand\_change() (*elec-sim.model.world.World* method), 6

override\_highest\_demand() (*elec-sim.model.world.World* method), 6

override\_total\_demand() (*elec-sim.model.world.World* method), 6

overwrite\_scenario\_file() (*elec-sim.model.world.World* method), 6

## P

parameter\_estimation() (*elec-sim.plants.plant\_costs.estimate\_costs.estimate\_modern\_power\_plant\_costs.predict\_modern\_plant\_costs.PredictModernPlantParameters* method), 8

partially\_accept\_bid() (*elec-sim.market.electricity.bid.Bid* method), 4

plant\_type\_to\_plant\_object() (*elec-sim.plants.plant\_registry.PlantRegistry* method), 13

PlantCostCalculations (class in *elec-sim.role.plants.costs.plant\_cost\_calculation*), 20

PlantRegistry (class in *elec-sim.plants.plant\_registry*), 13

PowerPlant (class in *elec-sim.plants.plant\_type.power\_plant*), 12

predict\_price\_duration\_curve() (*elec-sim.role.investment.predict\_load\_duration\_prices.PredictPriceDurationCurve* method), 14

PredictModernPlantParameters (class in *elec-sim.plants.plant\_costs.estimate\_costs.estimate\_modern\_power\_plant\_costs.predict\_modern\_plant\_costs.PredictModernPlantParameters*), 7

PredictPriceDurationCurve (class in *elec-sim.role.investment.predict\_load\_duration\_prices*), 14

*elec-sim.agents.generation\_company.gen\_co.GenCo*.purchase\_coal() (method), 2

*elec-sim.agents.generation\_company.gen\_co.GenCo*.purchase\_fuel() (method), 2

*elec-sim.agents.generation\_company.gen\_co.GenCo*.purchase\_gas() (method), 2

*elec-sim.market.electricity.bid.Bid*.reject\_bid() (method), 4

*elec-sim.model.world.World*.sample\_from\_custom\_distribution() (in module *elec-sim.data\_manipulation.data\_modifications.inverse\_transform\_sampler*), 3

*elec-sim.model.world.World*.select\_yearly\_payback\_payment\_for\_year() (in module *elec-sim.role.investment.calculate\_npv*), 14

*elec-sim.model.world.World*.set\_log\_level() (method), 6

*elec-sim.agents.generation\_company.gen\_co.GenCo*.settle\_accounts() (method), 2

*elec-sim.model.world.World*.settle\_gencos\_financials() (method), 6

*elec-sim.plants.plant\_type.non\_fuel\_plant.NonFuelPlant*.short\_run\_marginal\_cost (method), 12

*elec-sim.plants.plant\_type.fuel\_plant.FuelPlant*.short\_run\_marginal\_cost() (method), 11

*elec-sim.plants.plant\_type.power\_plant.PowerPlant*.short\_run\_marginal\_cost() (method), 13

*elec-sim.agents.demand.demand.Demand*.step() (method), 1

*elec-sim.agents.generation\_company.gen\_co.GenCo*.step() (method), 2

*elec-sim.mesa\_addons.scheduler\_addon.OrderedActivation*.step() (method), 5

*elec-sim.model.world.World*.step() (method), 6

*elec-sim.model.world.World*.stratify\_data() (method), 6

## T

*elec-sim.market.electricity.bid.Bid*.total\_income() (*elec-sim.role.plants.costs.plant\_cost\_calculation.PlantCostCalculations* method), 20

## W

World (*class in `elecsim.model.world`*), 5  
WorldPlantCapacity (*class in `elecsim.role.market.world_plant_capacity`*), 15  
write\_scenario\_data() (*`elecsim.model.world.World` method*), 6  
write\_timing\_results() (*`elecsim.model.world.World` method*), 6