ElecSim Documentation

Release 0.1

Alexander J. M. Kell

Sep 16, 2020

Contents:

CHAPTER 1

elecsim

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_energ**

elecsim.data_manipulation.data_modifications.renewable_learning_rate.**logger = <Logger elecs**
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*, *column*, *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_pla module

class elecsim.plants.plant_costs.estimate_costs.estimate_modern_power_plant_costs.predict_m

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.fu module

class elecsim.plants.plant_costs.estimate_costs.estimate_old_plant_cost_params.fuel_plant_c

Bases: *elecsim.plants.plant_costs.estimate_costs.estimate_old_plant_cost_params.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 (

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_calculatio 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_calculatio module

class elecsim.plants.plant_costs.estimate_costs.estimate_old_plant_cost_params.non_fuel_pla

Bases: *elecsim.plants.plant_costs.estimate_costs.estimate_old_plant_cost_params.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 contents

File 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_pa

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*, *simplified_type*, *capacity*, *number_of_plant*

elecsim.plants.plant_costs.estimate_costs.estimate_costs.**logger = <Logger elecsim.plants.pl**
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*, *difference_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_np**

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*, *interest*, *down_payment_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.investme**

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_**
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_**
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_capacity_mw*, *average_load_factor*, *pre_dev_period*, *construction_period*, *operating_period*, *pre_dev_spend_year*, *construction_spend_year*, *pre_dev_cost_per_mw*, *construction_cost_per_mw*, *infrastructure*, *fixed_o_and_m_per_mw*, *variable_o_and_m_per_mw*, *insurance_cost_per_mw*, *connection_cost_per_mw*, *efficiency*)

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*, *construction_year*, *average_load_factor*, *pre_dev_period*, *construction_period*, *operating_period*, *pre_dev_spend_yea*
construction_spend_years, *pre_dev_cost_per_m*
construction_cost_per_mw, *infrastructure*, *fixed_o_and_m_per*
*variable_o_and_m_per_*
insurance_cost_per_mw, *connection_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**

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

Index

D

E

F

G

H

I

L

M

N

O

P

R

S

T

W