Model documentation (Fan Tong’s model 20180503)

Fan Tong, May 4, 2018

**Abstract**

I wrote this documentation for my own model (which is out-of-synchronization from the GitHub branches for weeks). The idea of sharing this documentation is that we should have a similar doc for the GitHub model. Ideally, each major version/update of the GitHub model should be accompanied by such as documentation and/or a change log. In addition, the model should pass the diagnostic test scenarios and the optimization results for these scenarios should be saved.

Script.py

|  |  |
| --- | --- |
| What it does | The starting point of the code – run the whole coding package.  Run a set of scenarios.  Prepare coding assumptions.  Prepare model assumptions. |
| Input | The starting point of the code. |
| Output | Copy and paste this source code file to the output folder. |
| Functions called | Loop\_Model.py |
| Package used | numpy  shutil.copy2 |

Loop\_Model.py

|  |  |
| --- | --- |
| What it does | Defines/Houses loop\_model()  The outer layer of the model. The role is to set up loops and formulate inputs to call the core model, and then take care of the results. |
| Functions defined | loop\_model(): |
| Package used | os  csv  numpy  shutil.copy2  datetime |

Function:: loop\_model

|  |  |
| --- | --- |
| What it does | The outer layer of the model. The role is to set up loops and formulate inputs to call the core model, and then take care of the results. |
| Input | inputs - a dictionary variable sent from Script.py |
| Output (return variables) | output\_folder – the directory of the summary output folder. |
| Output (others) | Write key scenario information to console.  Save the complete scenario results to individually-named output folders.  Save the summary results to a csv file in the summary output folder.  Save the source code files to the summary output folder (ending with “SUMMARY”, different from the output folders created in core\_model()). |
| Functions called | core\_model()  post\_processing() |

Core\_Model.py

|  |  |
| --- | --- |
| What it does | Defines/Houses core\_model()  The actual core model. For a given set of inputs (data and technology assumptions), formulate the optimization problem, solve it, and save the optimization results. |
| Functions defined | core\_model(): |
| Package used | numpy  cvxpy  time |

Function:: core\_model

|  |  |
| --- | --- |
| What it does | Defines/Houses core\_model()  The actual core model. For a given set of inputs (data and technology assumptions), formulate the optimization problem, solve it, and save the optimization results. |
| Input | model\_inputs - a dictionary variable sent from loop\_model() |
| Output (return variables) | model\_results – a dictionary variable on optimization results, optimization status, and supporting information. |
| Output (other) | None |
| Functions called | None |

Dictionary variable::inputs

|  |
| --- |
| dictionary variable between Script.py (the starting script) and Loop\_Model.py (the outer layer of the core model).  defined in Scipt.py  19 technology assumption variables, 5 modeling control variables, 3+2 input data variables |
| # Programming assumptions  "DIRECTORY\_Project": DIRECTORY\_Project,  "DIRECTORY\_Input": DIRECTORY\_Input,  "DIRECTORY\_Output": DIRECTORY\_Output,  "PRINT\_FIGURE": PRINT\_FIGURE,  "DATA\_TYPE": DATA\_TYPE,  # Power generation technologies  "power\_tech\_index": power\_tech\_index,  "efficiency\_power": efficiency\_power,  "capital\_cost\_ngcc\_array": capital\_cost\_ngcc\_array,  "capital\_cost\_wind\_array": capital\_cost\_wind\_array,  "capital\_cost\_solar\_array": capital\_cost\_solar\_array,  "capital\_cost\_nuclear\_array": capital\_cost\_nuclear\_array,  # Fuel cost  "ng\_fuel\_cost\_array": ng\_fuel\_cost\_array,  "nuclear\_fuel\_cost": nuclear\_fuel\_cost,  "storage\_fuel\_cost": storage\_fuel\_cost,  # Energy storage technologies  "capital\_cost\_storage\_array": capital\_cost\_storage\_array,  "storage\_charging\_efficiency": storage\_charging\_efficiency,  # Unmet demand  "unmet\_demand\_cost\_array": unmet\_demand\_cost\_array,  "curtailment\_cost": curtailment\_cost,  # Other "default" assumptions  "variable\_OM\_cost\_power": variable\_OM\_cost\_power,  "variable\_OM\_cost\_storage": variable\_OM\_cost\_storage,  "fixed\_OM\_cost\_power": fixed\_OM\_cost\_power,  "fixed\_OM\_cost\_storage":fixed\_OM\_cost\_storage,  "capital\_charge\_rate\_power": capital\_charge\_rate\_power,  "capital\_charge\_rate\_storage": capital\_charge\_rate\_storage,  # Input data  "demand": demand,  "wind\_capacity\_factor": wind\_capacity\_factor,  "solar\_capacity\_factor":solar\_capacity\_factor,  # Dealing with input data  # "year\_simulation\_array":year\_simulation\_array,  "simulation\_start\_array": simulation\_start\_array,  "simulation\_end\_array": simulation\_end\_array, |

Dictionary variable:: model\_inputs

|  |
| --- |
| dictionary variable between Loop\_Model.py (the outer layer of the core model) and Core\_Model.py (the actual core model).  defined in Loop\_Model.py  # 8 assumption variables, 3 input data variables, 9 supporting variables |
| # Core assumptions  "power\_tech\_index": power\_tech\_index,  "fixed\_cost\_power": fixed\_cost\_power,  "fixed\_cost\_storage": fixed\_cost\_storage,  "variable\_cost\_power": variable\_cost\_power,  "variable\_cost\_storage": variable\_cost\_storage,  "storage\_charging\_efficiency": storage\_charging\_efficiency,  "unmet\_demand\_cost": unmet\_demand\_cost,  "curtailment\_cost": curtailment\_cost,  # Input data  "demand": demand,  "wind\_capacity\_factor": wind\_capacity\_factor,  "solar\_capacity\_factor": solar\_capacity\_factor,  # Supportive variables (used in post-processing)  "ng\_fuel\_cost": ng\_fuel\_cost,  "capital\_cost\_power": capital\_cost\_power,  "capital\_cost\_storage": capital\_cost\_storage,  "annualized\_capital\_cost\_power": annualized\_capital\_cost\_power,  "annualized\_capital\_cost\_storage": annualized\_capital\_cost\_storage,  "variable\_fuel\_cost\_power": variable\_fuel\_cost\_power,  "variable\_fuel\_cost\_storage": variable\_fuel\_cost\_storage,  "simulation\_start": simulation\_start,  "simulation\_end": simulation\_end, |

Dictionary variable:: model\_results

|  |
| --- |
| dictionary variable between Loop\_Model.py (the outer layer of the core model) and Core\_Model.py (the actual core model).  defined in Loop\_Model.py  8 core variables, and 4 supporting informative variables. |
| # Core optimization results (decision variables)  'dispatched\_power': dispatched\_power\_solution,  'capacity\_power': capacity\_power\_solution,  'dispatched\_storage\_discharge': dispatched\_storage\_discharge\_solution,  'dispatched\_storage\_charge': dispatched\_storage\_charge\_solution,  'unmet\_demand': unmet\_demand\_solution,  'dispatched\_curtailment': dispatched\_curtailment\_solution,  'capacity\_storage': capacity\_storage\_solution,  'storage\_energy\_soc': storage\_energy\_soc\_solution,  # Optimization status  'solution\_status': prob.status,  'optimum': prob.value,  # Supporting information (used in post-processing)  'run\_time': (simulation\_end\_timestamp - simulation\_start),  'constraints\_count': len(constraints), |