grg-mpdata Documentation

Release 0.1.1

Author

Dec 13, 2020

Contents

1 Introduction **3**
- 1.1 Overview . . . 3
- 1.2 Installation . . . 3
- 1.3 Testing . . . 3
- 1.4 Compatibility . . . 3

2 grg-mpdata package **5**
- 2.1 grg_mpdata.io module . . . 5
- 2.2 grg_mpdata.cmd module . . . 6
- 2.3 grg_mpdata.exception module . . . 6
- 2.4 grg_mpdata.struct module . . . 7
- 2.5 Module contents . . . 13

3 Indices and tables **15**

Python Module Index **17**

Index **19**

Contents:

CHAPTER 1

Introduction

1.1 Overview

grg-mpdata is a minimalist python package to support the reading and writing of Matpower network data files.

The primary entry point of the library is *grg_mpdata.io* module, which contains the methods for data input and output.

1.2 Installation

Simply run:

```
pip install grg-mpdata
```

1.3 Testing

grg-mpdata is designed to be a library that supports other software. It is not immediately useful from the terminal. However, you can test the parsing functionality from the command line with:

```
python -m grg_mpdata.io <path to Matpower case file>
```

If this command is successful, you will see a simplified plain text version of the network data printed to the terminal.

1.4 Compatibility

The following features of the Matpower 5.1 specification are not replicated in this library,

1. The following fields are not supported, 'A', 'l', 'u', 'H', 'Cw', 'N', 'fparm', 'z0', 'zl', 'zu'

2. The matrix columns are not extensible

CHAPTER 2

grg-mpdata package

2.1 grg_mpdata.io module

functions for reading and writing matpower data files

`grg_mpdata.io.`**`build_cli_parser`**`()`

`grg_mpdata.io.`**`main`**`(`*`args`*`)`

reads a matpower case file from a command line arguments and prints the parsed file to stdout

Parameters **`args`** – an argparse data structure

`grg_mpdata.io.`**`parse_mp_case_file`**`(`*`mpFileName`*`)`

opens the given path and parses it as matpower data

Parameters **`mpFileName`** (*`str`*) – path to the a matpower data file

Returns a grg_mpdata case

Return type *Case*

`grg_mpdata.io.`**`parse_mp_case_lines`**`(`*`mpLines`*`)`

parses a list of strings as matpower data

Parameters **`mpLines`** (*`list`*) – the list of matpower data strings

Returns a grg_mpdata case

Return type *Case*

`grg_mpdata.io.`**`parse_mp_case_str`**`(`*`mpString`*`)`

parses a given string as matpower data

Parameters **`mpString`** (*`str`*) – a matpower data file as a string

Returns a grg_mpdata case

Return type *Case*

`grg_mpdata.io.write_mp_case_file(output_file_location, case)`
writes a matpower case file

Parameters

- **output_file_location** (*str*) – the path of the file to write
- **case** (Case) – the data structure to write out

2.2 grg_mpdata.cmd module

functions for analyzing and transforming matpower data files

`grg_mpdata.cmd.build_cmd_parser()`

`grg_mpdata.cmd.compare_component_lists(list_1, list_2, comp_name, index_name='index')`
compares two lists and prints the differences to stdout. Objects in the lists are assumed to have an identification attribute.

Parameters

- **list_1** (*list*) – the first list
- **list_2** (*list*) – the second list
- **comp_name** (*string*) – the name of components being compared
- **index_name** (*string*) – the name of the object identification attribute

Returns (int): returns the number of items that differed in the two lists

`grg_mpdata.cmd.diff(case_1, case_2)`
Compares two *grg_mpdata.struct.Case* objects and prints the differences to stdout.

Parameters

- **case_1** – the first Matpower case
- **case_2** – the second Matpower case

Returns (int): returns the number of items that differed in the two cases

`grg_mpdata.cmd.eq(case_1, case_2)`

`grg_mpdata.cmd.main(args)`
reads a matpower case files and processes them based on command line arguments.

Parameters **args** – an argparse data structure

2.3 grg_mpdata.exception module

a collection of all grg_mpdata exception classes

exception `grg_mpdata.exception.MPDataException`
Bases: `exceptions.Exception`

root class for all MPData Exceptions

exception grg_mpdata.exception.**MPDataParsingError**

Bases: *grg_mpdata.exception.MPDataException*

for errors that occur while attempting to parse a matpower data file

exception grg_mpdata.exception.**MPDataValidationError**

Bases: *grg_mpdata.exception.MPDataException*

for errors that occur while attempting to validate the correctness of a parsed matpower data file

exception grg_mpdata.exception.**MPDataWarning**

Bases: exceptions.Warning

root class for all MPData Warnings

2.4 grg_mpdata.struct module

data structures for encoding matpower data files

class grg_mpdata.struct.**Branch**(*index*, *f_bus*, *t_bus*, *br_r*, *br_x*, *br_b=0.0*, *rate_a=0.0*, *rate_b=0.0*, *rate_c=0.0*, *tap=0.0*, *shift=0.0*, *br_status=1*, *angmin=-360.0*, *angmax=360.0*, *pf=None*, *qf=None*, *pt=None*, *qt=None*, *mu_sf=None*, *mu_st=None*, *mu_angmin=None*, *mu_angmax=None*)

Bases: object

This data structure contains key power network branch parameters. If the value of tap or shift are non-zero, the branch is considered a transformer. Angle difference bound arguments have default values for backward compatibility with an older data specification.

Parameters

- **index** (*int*) – unique branch identifier
- **f_bus** (*int*) – the identifier of the from bus
- **t_bus** (*int*) – the identifier of the to bus
- **br_r** (*float*) – the branch resistance (p.u.)
- **br_x** (*float*) – the branch reactance (p.u.)
- **br_b** (*float*) – the total branch charging susceptance (p.u.)
- **rate_a** (*float*) – long term rating (MVA)
- **rate_b** (*float*) – short term rating (MVA)
- **rate_c** (*float*) – emergency rating (MVA)
- **tap** (*float*) – transformer off nominal turn ratio (p.u.)
- **shift** (*float*) – positive delay transformer phase shift (degrees)
- **br_status** (*int*) – branch status (in service = 1, out of service = 0)
- **angmin** (*float*) – phase angle difference lower bound (degrees)
- **angmax** (*float*) – phase angle difference upper bound (degrees)
- **pf** (*float, optional*) – from bus active power flow (MW)
- **qf** (*float, optional*) – from bus reactive power flow (MVAr)
- **pt** (*float, optional*) – to bus active power flow (MW)

- **qt** (*float, optional*) – to bus reactive power flow (MVAr)
- **mu_sf** (*float, optional*) – KKT multiplier on from bus long term rating limit (u/MVA)
- **mu_st** (*float, optional*) – KKT multiplier on to bus long term rating limit (u/MVA)
- **mu_angmin** (*float, optional*) – KKT multiplier on phase angle difference lower bound (u/degree)
- **mu_angmax** (*float, optional*) – KKT multiplier on phase angle difference upper bound (u/degree)

to_matpower()

Returns: a Matpower encoding of this data structure as a string

validate()

Checks that this data structure conforms to the Matpower data specification

class grg_mpdata.struct.**Bus**(*bus_i*, *bus_type*, *pd*, *qd*, *gs*, *bs*, *area*, *vm*, *va*, *base_kv*, *zone*, *vmax*, *vmin*, *lam_p=None*, *lam_q=None*, *mu_vmax=None*, *mu_vmin=None*)

Bases: object

This data structure contains key power network bus parameters.

Parameters

- **bus_i** (*int*) – unique bus identifier
- **bus_type** (*int*) – PQ = 1, PV = 2, reference = 3, disconnected = 4
- **pd** (*float*) – active power demand (MW)
- **qd** (*float*) – reactive power demand (MVAr)
- **gs** (*float*) – shunt conductance (MW at 1.0 volts p.u.)
- **bs** (*float*) – shunt susceptance (MVAr at 1.0 volts p.u.)
- **area** (*int*) – area identifier
- **vm** (*float*) – voltage magnitude (volts p.u.)
- **va** (*float*) – voltage angle (degrees)
- **base_kv** (*float*) – base voltage (kilovolts)
- **zone** (*int*) – loss zone
- **vmax** (*float*) – voltage magnitude upper bound (volts p.u.)
- **vmin** (*float*) – voltage magnitude lower bound (volts p.u.)
- **lam_p** (*float, optional*) – Lagrange multiplier on active power KCL (u/MW)
- **lam_q** (*float, optional*) – Lagrange multiplier on reactive power KCL (u/MVAr)
- **mu_vmax** (*float, optional*) – KKT multiplier on voltage upper bound (u/volts p.u.)
- **mu_vmin** (*float, optional*) – KKT multiplier on voltage lower bound (u/volts p.u.)

to_matpower()

Returns: a Matpower encoding of this data structure as a string

validate()

Checks that this data structure conforms to the Matpower data specification

class grg_mpdata.struct.**BusName**(*index*, *name*)

Bases: object

This data structure contains bus name parameters.

Parameters

- **index** (*int*) – unique identifier for this bus name
- **name** (*str*) – a bus name

to_matpower()

Returns: a Matpower encoding of this data structure as a string

validate()

Checks that this data structure conforms to the Matpower data specification

class grg_mpdata.struct.**Case**(*name=None*, *version=None*, *baseMVA=None*, *bus=None*, *gen=None*, *branch=None*, *gencost=None*, *dcline=None*, *dclinecost=None*, *busname=None*)

Bases: object

This data structure contains lists of all the key components in a power network. All arguments have default values to allow clear error messages to be generated in the validation method. At this time, only Matpower case version 2 is supported.

Parameters

- **name** (*str*) – textual name of the test case. Must be a valid matlab function identifier
- **version** (*str*) – indicates the version of the test case
- **baseMVA** (*float*) – the network MVA base value (MVA)
- **bus** (*list of Bus*) – network buses
- **gen** (*list of Generator*) – network generators
- **branch** (*list of Branch*) – network branches
- **gencost** (*list of GeneratorCost, optional*) – generator cost models
- **dcline** (*list of DCLine, optional*) – network DC lines
- **dclinecost** (*list of DCLineCost, optional*) – DC line cost models
- **busname** (*list of BusName, optional*) – string names of items in bus list

remove_status_zero()

to_matpower()

Returns: a Matpower encoding of this data structure as a string

validate()

Checks that this data structure conforms to the Matpower data specification.

class grg_mpdata.struct.**DCLine**(*index*, *f_bus*, *t_bus*, *br_status*, *pf*, *pt*, *qf*, *qt*, *vf*, *vt*, *pmin*, *pmax*, *qminf*, *qmaxf*, *qmint*, *qmaxt*, *loss0*, *loss1*, *mu_pmin=None*, *mu_pmax=None*, *mu_qminf=None*, *mu_qmaxf=None*, *mu_qmint=None*, *mu_qmaxt=None*)

Bases: object

This data structure contains key power network dc line parameters.

Parameters

- **index** (*int*) – unique dc line identifier

- **f_bus** (*int*) – the identifier of the from bus
- **t_bus** (*int*) – the identifier of the to bus
- **br_status** (*int*) – dc line status (in service = 1, out of service = 0)
- **pf** (*float*) – from bus active power flow (MW)
- **pt** (*float*) – to bus active power flow (MW)
- **qf** (*float*) – from bus reactive power flow (MVAr)
- **qt** (*float*) – to bus reactive power flow (MVAr)
- **vf** (*float*) – from bus voltage magnitude setpoint (volts p.u.)
- **vt** (*float*) – to bus voltage magnitude setpoint (volts p.u.)
- **pmin** (*float*) – active power flow lower bound (MW), from bus if >= 0, to bus if < 0
- **pmax** (*float*) – active power flow upper bound (MW), from bus if >= 0, to bus if < 0
- **qminf** (*float*) – from bus reactive power flow lower bound (MVAr)
- **qmaxf** (*float*) – from bus reactive power flow upper bound (MVAr)
- **qmint** (*float*) – to bus reactive power flow lower bound (MVAr)
- **qmaxt** (*float*) – to bus reactive power flow upper bound (MVAr)
- **loss0** (*float*) – constant term in from bus active power loss function (MW)
- **loss1** (*float*) – linear term in from bus active power loss function (scalar)
- **mu_pmin** (*float, optional*) – KKT multiplier on from bus active power lower bound (u/MW)
- **mu_pmax** (*float, optional*) – KKT multiplier on from bus active power upper bound (u/MW)
- **mu_qminf** (*float, optional*) – KKT multiplier on from bus reactive power lower bound (u/MVAr)
- **mu_qmaxf** (*float, optional*) – KKT multiplier on from bus reactive power upper bound (u/MVAr)
- **mu_qmint** (*float, optional*) – KKT multiplier on to bus reactive power lower bound (u/MVAr)
- **mu_qmaxt** (*float, optional*) – KKT multiplier on to bus reactive power upper bound (u/MVAr)

to_matpower()

Returns: a Matpower encoding of this data structure as a string

validate()

Checks that this data structure conforms to the Matpower data specification

class grg_mpdata.struct.**DCLineCost**(*index*, *model*, *startup=0*, *shutdown=0*, *ncost=0*, *cost=[]*)

Bases: *grg_mpdata.struct.MatpowerCost*

This data structure contains key power generator cost model parameters. Note that the generator cost identifier (i.e. index) is used to link the cost model to a particular generator

The piecewise linear model $f(x)$ is defined by, the coordinates $(\boldsymbol{cost}_0, \boldsymbol{cost}_1)$, $(\boldsymbol{cost}_2, \boldsymbol{cost}_3)$, ... , $(\boldsymbol{cost}_{2\boldsymbol{ncost}-1}, \boldsymbol{cost}_{2\boldsymbol{ncost}})$ of the end/break-points of the piecewise linear cost.

The polynomial cost model is defined as, $f(x) = \sum_{i \in 1..\boldsymbol{ncost}} \boldsymbol{cost}_{i-1} x^{\boldsymbol{ncost}-i}$.

Parameters

- **index** (*int*) – unique generator cost identifier
- **model** (*int*) – generator cost model (piecewise linear = 1, polynomial = 2)
- **startup** (*float*) – startup costs (US Dollars)
- **shutdown** (*float*) – shutdown costs (US Dollars)
- **ncost** (*int*) – number of data points or cost coefficients
- **cost** (*list of float*) – the list of data points or cost coefficients (US Dollars/hour), if a polynomial model it should have ncost values, if a piecewise linear model it should have 2*ncost values

class grg_mpdata.struct.**Generator**(*index*, *gen_bus*, *pg*, *qg*, *qmax*, *qmin*, *vg*, *mbase*, *gen_status*, *pmax*, *pmin*, *pc1=0*, *pc2=0*, *qc1min=0*, *qc1max=0*, *qc2min=0*, *qc2max=0*, *ramp_agc=0*, *ramp_10=0*, *ramp_30=0*, *ramp_q=0*, *apf=0*, *mu_pmax=None*, *mu_pmin=None*, *mu_qmax=None*, *mu_qmin=None*)

Bases: object

This data structure contains key power generator parameters. Some arguments have default values of 0 for backward compatibility with an older data specification.

Parameters

- **index** (*int*) – unique generator identifier
- **gen_bus** (*int*) – the identifier of the bus that this generator is connected to
- **pg** (*float*) – active power output (MW)
- **qg** (*float*) – reactive power output (MVAr)
- **qmax** (*float*) – reactive power output upper bound (MVAr)
- **qmin** (*float*) – reactive power output lower bound (MVAr)
- **vg** (*float*) – voltage magnitude setpoint (volts p.u.)
- **mbase** (*float*) – machine mva base (MVA)
- **gen_status** (*int*) – generator status (in service > 0, out of service <= 0)
- **pmax** (*float*) – active power output upper bound (MW)
- **pmin** (*float*) – active power output lower bound (MW)
- **pc1** (*float*) – PQ capability curve, active power lower bound (MW)
- **pc2** (*float*) – PQ capability curve, active power upper bound (MW)
- **qc1min** (*float*) – reactive power output lower bound, at PC1 (MVAr)
- **qc1max** (*float*) – reactive power output upper bound, at PC1 (MVAr)
- **qc2min** (*float*) – reactive power output lower bound, at PC2 (MVAr)
- **qc2max** (*float*) – reactive power output upper bound, at PC2 (MVAr)
- **ramp_agc** (*float*) – AGC ramp rate (MW/min)
- **ramp_10** (*float*) – ramp rate for 10 minute reserves (MW)
- **ramp_30** (*float*) – ramp rate for 30 minute reserves (MW)
- **ramp_q** (*float*) – ramp rate for reactive power (MVAr/min)

- **apf** (*float*) – area participation factor
- **mu_pmax** (*float, optional*) – KKT multiplier on active power output upper bound (u/MW)
- **mu_pmin** (*float, optional*) – KKT multiplier on active power output lower bound (u/MW)
- **mu_qmax** (*float, optional*) – KKT multiplier on reactive power output upper bound (u/MVAr)
- **mu_qmin** (*float, optional*) – KKT multiplier on reactive power output lower bound (u/MVAr)

to_matpower()

Returns: a Matpower encoding of this data structure as a string

validate()

Checks that this data structure conforms to the Matpower data specification

class grg_mpdata.struct.**GeneratorCost**(*index, model, startup=0, shutdown=0, ncost=0, cost=[]*)

Bases: *grg_mpdata.struct.MatpowerCost*

This data structure contains key power generator cost model parameters. Note that the generator cost identifier (i.e. index) is used to link the cost model to a particular generator

The piecewise linear model $f(x)$ is defined by, the coordinates $(\boldsymbol{cost}_0, \boldsymbol{cost}_1)$, $(\boldsymbol{cost}_2, \boldsymbol{cost}_3)$, ... , $(\boldsymbol{cost}_{2\boldsymbol{ncost}-1}, \boldsymbol{cost}_{2\boldsymbol{ncost}})$ of the end/break-points of the piecewise linear cost.

The polynomial cost model is defined as, $f(x) = \sum_{i \in 1..\boldsymbol{ncost}} \boldsymbol{cost}_{i-1} x^{\boldsymbol{ncost}-i}$.

Parameters

- **index** (*int*) – unique generator cost identifier
- **model** (*int*) – generator cost model (piecewise linear = 1, polynomial = 2)
- **startup** (*float*) – startup costs (US Dollars)
- **shutdown** (*float*) – shutdown costs (US Dollars)
- **ncost** (*int*) – number of data points or cost coefficients
- **cost** (*list of float*) – the list of data points or cost coefficients (US Dollars/hour), if a polynomial model it should have ncost values, if a piecewise linear model it should have 2*ncost values

class grg_mpdata.struct.**MatpowerCost**(*index, model, startup=0, shutdown=0, ncost=0, cost=[]*)

Bases: object

This data structure contains key power generator cost model parameters. Note that the generator cost identifier (i.e. index) is used to link the cost model to a particular generator

The piecewise linear model $f(x)$ is defined by, the coordinates $(\boldsymbol{cost}_0, \boldsymbol{cost}_1)$, $(\boldsymbol{cost}_2, \boldsymbol{cost}_3)$, ... , $(\boldsymbol{cost}_{2\boldsymbol{ncost}-1}, \boldsymbol{cost}_{2\boldsymbol{ncost}})$ of the end/break-points of the piecewise linear cost.

The polynomial cost model is defined as, $f(x) = \sum_{i \in 1..\boldsymbol{ncost}} \boldsymbol{cost}_{i-1} x^{\boldsymbol{ncost}-i}$.

Parameters

- **index** (*int*) – unique generator cost identifier
- **model** (*int*) – generator cost model (piecewise linear = 1, polynomial = 2)
- **startup** (*float*) – startup costs (US Dollars)

- **shutdown** (*float*) – shutdown costs (US Dollars)
- **ncost** (*int*) – number of data points or cost coefficients
- **cost** (*list of float*) – the list of data points or cost coefficients (US Dollars/hour), if a polynomial model it should have ncost values, if a piecewise linear model it should have 2*ncost values

to_matpower()

Returns: a Matpower encoding of this data structure as a string

validate()

Checks that this data structure conforms to the Matpower data specification

2.5 Module contents

a package for reading and writing of matpower data files

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

g

grg_mpdata, 13
grg_mpdata.cmd, 6
grg_mpdata.exception, 6
grg_mpdata.io, 5
grg_mpdata.struct, 7

Index

B

`Branch` (*class in grg_mpdata.struct*), 7
`build_cli_parser()` (*in module grg_mpdata.io*), 5
`build_cmd_parser()` (*in module grg_mpdata.cmd*), 6
`Bus` (*class in grg_mpdata.struct*), 8
`BusName` (*class in grg_mpdata.struct*), 8

C

`Case` (*class in grg_mpdata.struct*), 9
`compare_component_lists()` (*in module grg_mpdata.cmd*), 6

D

`DCLine` (*class in grg_mpdata.struct*), 9
`DCLineCost` (*class in grg_mpdata.struct*), 10
`diff()` (*in module grg_mpdata.cmd*), 6

E

`eq()` (*in module grg_mpdata.cmd*), 6

G

`Generator` (*class in grg_mpdata.struct*), 11
`GeneratorCost` (*class in grg_mpdata.struct*), 12
`grg_mpdata` (*module*), 13
`grg_mpdata.cmd` (*module*), 6
`grg_mpdata.exception` (*module*), 6
`grg_mpdata.io` (*module*), 5
`grg_mpdata.struct` (*module*), 7

M

`main()` (*in module grg_mpdata.cmd*), 6
`main()` (*in module grg_mpdata.io*), 5
`MatpowerCost` (*class in grg_mpdata.struct*), 12
`MPDataException`, 6
`MPDataParsingError`, 6
`MPDataValidationError`, 7
`MPDataWarning`, 7

P

`parse_mp_case_file()` (*in module grg_mpdata.io*), 5
`parse_mp_case_lines()` (*in module grg_mpdata.io*), 5
`parse_mp_case_str()` (*in module grg_mpdata.io*), 5

R

`remove_status_zero()` (*grg_mpdata.struct.Case method*), 9

T

`to_matpower()` (*grg_mpdata.struct.Branch method*), 8
`to_matpower()` (*grg_mpdata.struct.Bus method*), 8
`to_matpower()` (*grg_mpdata.struct.BusName method*), 9
`to_matpower()` (*grg_mpdata.struct.Case method*), 9
`to_matpower()` (*grg_mpdata.struct.DCLine method*), 10
`to_matpower()` (*grg_mpdata.struct.Generator method*), 12
`to_matpower()` (*grg_mpdata.struct.MatpowerCost method*), 13

V

`validate()` (*grg_mpdata.struct.Branch method*), 8
`validate()` (*grg_mpdata.struct.Bus method*), 8
`validate()` (*grg_mpdata.struct.BusName method*), 9
`validate()` (*grg_mpdata.struct.Case method*), 9
`validate()` (*grg_mpdata.struct.DCLine method*), 10
`validate()` (*grg_mpdata.struct.Generator method*), 12
`validate()` (*grg_mpdata.struct.MatpowerCost method*), 13

W

`write_mp_case_file()` (*in module grg_mpdata.io*), 5