grg-mp2grg Documentation

Release 0.1.1

Carleton Coffrin

Aug 18, 2019

Contents:

1 Introduction **1**
- 1.1 Overview . . . 1
- 1.2 Installation . . . 1
- 1.3 Testing . . . 1

2 grg-mp2grg package **3**
- 2.1 grg_mp2grg.io module . . . 3
- 2.2 grg_mp2grg.exception module . . . 4
- 2.3 grg_mp2grg.struct module . . . 4
- 2.4 Module contents . . . 6

3 Indices and tables **7**

Python Module Index **9**

Index **11**

CHAPTER 1

Introduction

1.1 Overview

grg-mp2grg is a python package for translating Matpower and GRG network data files.

The primary entry point of the library is *grg_mp2grg.io* module, which contains the methods for bi-directional translation.

1.2 Installation

Simply run:

```
pip install grg-mp2grg
```

1.3 Testing

grg-mp2grg 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_mp2grg.io <path to Matpower or GRG case file>
```

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

CHAPTER 2

grg-mp2grg package

2.1 grg_mp2grg.io module

functions for reading and writing matpower data files

grg_mp2grg.io.**build_cli_parser**()

grg_mp2grg.io.**build_gen_cost_mp**(*index*, *grg_cost_model*, *base_mva*, *float_precision*)

grg_mp2grg.io.**build_gen_cost_mp_default**(*index*, *model_type*, *degree*)

grg_mp2grg.io.**build_gen_cost_mp_losses**(*index*, *model_type*, *degree*)

grg_mp2grg.io.**build_mp_case**(*grg_data*, *mapping_ids=None*, *add_gen_costs=False*, *add_bus_names=False*)

grg_mp2grg.io.**currents_to_mvas**(*currents*, *from_bus*, *to_bus*)

grg_mp2grg.io.**main**(*args*)
reads a matpower or grg case file and processes it based on command line arguments.

Args: args: an argparse data structure

grg_mp2grg.io.**parse_grg_case_file**(*grg_file_name*)
opens the given path and parses it as json data

Args: grg_file_name(str): path to the a json data file

Returns: Dict: a dictionary case

grg_mp2grg.io.**parse_mp_case_file**(*mpFileName*)
opens the given path and parses it as matpower data

Args: mpFileName(str): path to the a matpower data file

Returns: Case: a mpdata case

grg_mp2grg.io.**parse_mp_case_lines**(*mpLines*)
parses a list of strings as matpower data

Args: mpLines(list): the list of matpower data strings

Returns: Case: a grg_mp2grg case

grg_mp2grg.io.**print_err**()
print(value, ..., sep=' ', end='n', file=sys.stdout)

Prints the values to a stream, or to sys.stdout by default. Optional keyword arguments: file: a file-like object (stream); defaults to the current sys.stdout. sep: string inserted between values, default a space. end: string appended after the last value, default a newline.

grg_mp2grg.io.**test_idempotent**(*input_data_file*)

grg_mp2grg.io.**write_json_case_file**(*output_file_location*, *case*)
writes a grg data json file

Args: output_file_location (str): the path of the file to write case (Case): the data structure to write out

2.2 grg_mp2grg.exception module

a collection of all grg_mp2grg exception classes

exception grg_mp2grg.exception.**MP2GRGWarning**
Bases: exceptions.Warning

root class for all MP2GRG Warnings

2.3 grg_mp2grg.struct module

extensions to data structures for encoding matpower data files to support grg data encoding

class grg_mp2grg.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: grg_mpdata.struct.Branch

get_grg_operations(*lookup*)

get_grg_setpoint(*base_mva*)
Returns: a grg data power flow set point as a dictionary

get_grg_status()
Returns: a grg data status assignment as a dictionary

get_grg_tap_changer_setpoint(*lookup*)

is_transformer()

to_grg_line(*lookup*, *base_mva*, *omit_subtype=False*)
Returns: a grg data line name and data as a dictionary

class grg_mp2grg.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: grg_mpdata.struct.Bus

get_grg_bus_setpoint(*lookup*)
Returns: a grg data voltage set point as a dictionary

get_grg_load_setpoint(*lookup*, *base_mva*)

get_grg_status()
Returns: a grg data status assignment as a dictionary

has_load()

has_shunt()

to_grg_bus(*lookup*, *omit_subtype=False*)
Returns: a grg data bus name and data as a dictionary

to_grg_load(*lookup*, *base_mva*, *omit_subtype=False*)
Returns: a grg data load name and data as a dictionary

to_grg_shunt(*lookup*, *base_mva*, *omit_subtype=False*)
Returns: a grg data shunt name and data as a dictionary

class grg_mp2grg.struct.**Case**(*name=None*, *version=None*, *baseMVA=None*, *bus=None*, *gen=None*, *branch=None*, *gencost=None*, *dcline=None*, *dclinecost=None*, *busname=None*)
Bases: grg_mpdata.struct.Case

to_grg(*omit_subtype=False*, *skip_validation=False*)
Returns: an encoding of this data structure as a grg data dictionary

class grg_mp2grg.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: grg_mpdata.struct.DCLine

get_grg_setpoint(*lookup*, *base_mva*)
Returns: a grg data power flow set point as a dictionary

get_grg_status()
Returns: a grg data status assignment as a dictionary

to_grg_dcline(*lookup*, *base_mva*, *omit_subtype=False*)
Returns: a grg data dc line name and data as a dictionary

class grg_mp2grg.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: grg_mpdata.struct.Generator

get_grg_setpoint(*lookup*, *base_mva*)
Returns: a grg data power output set point as a dictionary

get_grg_status()
Returns: a grg data status assignment as a dictionary

is_synchronous_condenser()

to_grg_generator(*lookup*, *base_mva*, *omit_subtype=False*)
Returns: a grg data gen name and data as a dictionary

class grg_mp2grg.struct.**GeneratorCost**(*index*, *model*, *startup=0*, *shutdown=0*, *ncost=0*, *cost=[]*)
Bases: grg_mpdata.struct.GeneratorCost

get_grg_cost_model(*lookup*, *gen_id*, *gen_count*, *base_mva*)
Returns: a grg data encoding of this data structure as a dictionary

2.4 Module contents

a package for converting matpower data files to grg data files

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

g

grg_mp2grg, 6
grg_mp2grg.exception, 4
grg_mp2grg.io, 3
grg_mp2grg.struct, 4

Index

B

Branch (*class in grg_mp2grg.struct*), 4
build_cli_parser() (*in module grg_mp2grg.io*), 3
build_gen_cost_mp() (*in module grg_mp2grg.io*), 3
build_gen_cost_mp_default() (*in module grg_mp2grg.io*), 3
build_gen_cost_mp_losses() (*in module grg_mp2grg.io*), 3
build_mp_case() (*in module grg_mp2grg.io*), 3
Bus (*class in grg_mp2grg.struct*), 4

C

Case (*class in grg_mp2grg.struct*), 5
currents_to_mvas() (*in module grg_mp2grg.io*), 3

D

DCLine (*class in grg_mp2grg.struct*), 5

G

Generator (*class in grg_mp2grg.struct*), 5
GeneratorCost (*class in grg_mp2grg.struct*), 5
get_grg_bus_setpoint() (*grg_mp2grg.struct.Bus method*), 4
get_grg_cost_model() (*grg_mp2grg.struct.GeneratorCost method*), 5
get_grg_load_setpoint() (*grg_mp2grg.struct.Bus method*), 5
get_grg_operations() (*grg_mp2grg.struct.Branch method*), 4
get_grg_setpoint() (*grg_mp2grg.struct.Branch method*), 4
get_grg_setpoint() (*grg_mp2grg.struct.DCLine method*), 5
get_grg_setpoint() (*grg_mp2grg.struct.Generator method*), 5
get_grg_status() (*grg_mp2grg.struct.Branch method*), 4
get_grg_status() (*grg_mp2grg.struct.Bus method*), 5
get_grg_status() (*grg_mp2grg.struct.DCLine method*), 5
get_grg_status() (*grg_mp2grg.struct.Generator method*), 5
get_grg_tap_changer_setpoint() (*grg_mp2grg.struct.Branch method*), 4
grg_mp2grg (*module*), 6
grg_mp2grg.exception (*module*), 4
grg_mp2grg.io (*module*), 3
grg_mp2grg.struct (*module*), 4

H

has_load() (*grg_mp2grg.struct.Bus method*), 5
has_shunt() (*grg_mp2grg.struct.Bus method*), 5

I

is_synchronous_condenser() (*grg_mp2grg.struct.Generator method*), 5
is_transformer() (*grg_mp2grg.struct.Branch method*), 4

M

main() (*in module grg_mp2grg.io*), 3
MP2GRGWarning, 4

P

parse_grg_case_file() (*in module grg_mp2grg.io*), 3
parse_mp_case_file() (*in module grg_mp2grg.io*), 3
parse_mp_case_lines() (*in module grg_mp2grg.io*), 3
print_err() (*in module grg_mp2grg.io*), 4

T

test_idempotent() (*in module grg_mp2grg.io*), 4

to_grg() (*grg_mp2grg.struct.Case method*), 5
to_grg_bus() (*grg_mp2grg.struct.Bus method*), 5
to_grg_dcline() (*grg_mp2grg.struct.DCLine method*), 5
to_grg_generator() (*grg_mp2grg.struct.Generator method*), 5
to_grg_line() (*grg_mp2grg.struct.Branch method*), 4
to_grg_load() (*grg_mp2grg.struct.Bus method*), 5
to_grg_shunt() (*grg_mp2grg.struct.Bus method*), 5

W

write_json_case_file() (*in module grg_mp2grg.io*), 4