Power Injection

October 21, 2023

1 Power Injection

This example is to elaborate how to get the bus injected power in ANDES, as answer for Discussion #471.

```
[ ]: import numpy as np
     import pandas as pd

     import andes

     import datetime
```

```
[ ]: print(f"Last run: {datetime.datetime.now()}\nANDES version: {andes.
       ↪__version__}")
```

```
Last run: 2023-10-21 21:51:03.716875
ANDES version: 1.8.10.post16+g596662e5
```

```
[ ]: andes.config_logger(stream_level=20)
```

Here we use IEEE 14-bus case as an example.

Load the case, run power flow, and initializ the TDS.

```
[ ]: sa = andes.load(andes.get_case("ieee14/ieee14_full.xlsx"),
                     no_output=True, setup=True)

     sa.PFlow.run()

     _ = sa.TDS.init()
```

```
Working directory: "/Users/jinningwang/Documents/work/ibrs/src/notes"
> Loaded config from file "/Users/jinningwang/.andes/andes.rc"
> Loaded config from file "/Users/jinningwang/.andes/andes.rc"
> Loaded generated Python code in "/Users/jinningwang/.andes/pycode".
Parsing input file
"/Users/jinningwang/Documents/work/andes/andes/cases/ieee14/ieee14_full.xlsx"…
Input file parsed in 0.1095 seconds.
System internal structure set up in 0.0220 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.
  Each island has a slack bus correctly defined and enabled.

-> Power flow calculation
           Numba: Off
   Sparse solver: KLU
 Solution method: NR method
Power flow initialized in 0.0041 seconds.
0: |F(x)| = 0.5605182134
1: |F(x)| = 0.006202200332
2: |F(x)| = 5.819382825e-06
3: |F(x)| = 6.957087684e-12
Converged in 4 iterations in 0.0032 seconds.
Initialization for dynamics completed in 0.0320 seconds.
Initialization was successful.
```

Initialize variables to store bus injected power from `Line`, `SynGen`, and `Load`, respectively.

[ ]:
```
p_inj_line = np.zeros(sa.Bus.n)
p_inj_syg = np.zeros(sa.Bus.n)
p_inj_load = np.zeros(sa.Bus.n)
```

In this case we only have `SynGen` in dynamic generators. If `RenGen` occurs, similar method can be applied to include it.

[ ]:
```
syg_idx = []
for mdl in sa.SynGen.models.values():
    syg_idx += mdl.idx.v

syg_bus = sa.SynGen.get(src='bus', attr='v', idx=syg_idx)

load_idx = []
for mdl in sa.StaticLoad.models.values():
    load_idx += mdl.idx.v

load_bus = sa.StaticLoad.get(src='bus', attr='v', idx=load_idx)
```

Here, a for loop is used to iterate through all buses in the system.

Note that this is only for demonstration purpose, and can be inefficient for large cases.

In model `Line`, attribute `a1` and `a2` are `ExtAlgeb` objects that will be summed to target `Bus` variable `a` for active power calculation. The attribute `e` of `ExtAlgeb` is the injected value.

Similarly, for a model connected to a bus such as `SynGen` or `Load`, there is usually an `ExtAlgeb` named `a` that is connected to the bus variable `a`.

[ ]:
```
for bus_idx in sa.Bus.idx.v:
    # get the location of bus device
    bus_loc = sa.Bus.idx2uid(bus_idx)

    # find the Line idx given "from bus"
    # NOTE: method `find_idx` returns incomplete idx if multiple matches occur
    inj_line_idx = []
    for line_idx in sa.Line.idx.v:
        if sa.Line.get(src='bus1', attr='v', idx=line_idx) == bus_idx:
            inj_line_idx.append(line_idx)
    line_loc = sa.Line.idx2uid(inj_line_idx)
    line_e = sa.Line.get(src='a1', attr='e', idx=inj_line_idx)
    p_inj_line[bus_loc] += line_e.sum()

    # similar, find the Line idx given "to bus"
    inj_line_idx = []
    for line_idx in sa.Line.idx.v:
        if sa.Line.get(src='bus2', attr='v', idx=line_idx) == bus_idx:
            inj_line_idx.append(line_idx)
    line_loc = sa.Line.idx2uid(inj_line_idx)
    line_e = sa.Line.get(src='a2', attr='e', idx=inj_line_idx)
    p_inj_line[bus_loc] += line_e.sum()

    # get Dynamic Generator idx given "bus"
    inj_syg_idx = []
    for syg in syg_idx:
        if sa.SynGen.get(src='bus', attr='v', idx=syg) == bus_idx:
            inj_syg_idx.append(syg)
    syg_e = sa.SynGen.get(src='a', attr='e', idx=inj_syg_idx)
    p_inj_syg[bus_loc] += syg_e.sum()

    # NOTE: If DynLoad occurs, similar method can be used
    inj_load_idx = []
    for load in load_idx:
        if sa.StaticLoad.get(src='bus', attr='v', idx=load) == bus_idx:
            inj_load_idx.append(load)
    load_e = sa.StaticLoad.get(src='a', attr='e', idx=inj_load_idx)
    p_inj_load[bus_loc] += load_e.sum()
```

In the last, the total bus injected power can be summed up from the three components.

Note that the positive direction is defined "out from the bus".

[ ]:
```
p_inj_bus = p_inj_line + p_inj_syg + p_inj_load

p_inj = pd.DataFrame({'Bus': sa.Bus.idx.v,
                      'Line': p_inj_line, 'SynGen': p_inj_syg,
                      'Load': p_inj_load, 'Total': p_inj_bus})

p_inj.round(4)
```

```
[ ]:     Bus    Line  SynGen   Load  Total
     0     1  0.8143 -0.8143  0.000   -0.0
     1     2  0.1830 -0.4000  0.217    0.0
     2     3 -0.1000 -0.4000  0.500   -0.0
     3     4 -0.4780  0.0000  0.478   -0.0
     4     5 -0.0760  0.0000  0.076    0.0
     5     6  0.1500 -0.3000  0.150   -0.0
     6     7 -0.0000  0.0000  0.000   -0.0
     7     8  0.3500 -0.3500  0.000    0.0
     8     9 -0.2950  0.0000  0.295    0.0
     9    10 -0.0900  0.0000  0.090   -0.0
     10   11 -0.0350  0.0000  0.035    0.0
     11   12 -0.0610  0.0000  0.061   -0.0
     12   13 -0.1350  0.0000  0.135   -0.0
     13   14 -0.2000  0.0000  0.200   -0.0
```