Purpose

This document is meant to explain PST additions and alterations created to accommodate AGC. Code examples are taken from the AGC example file `d2a_AGC.m` which is a modified version of `d2a_dce.m`.

Area Definitions

To enable allow for AGC and the required area calculations, each bus in the `bus` array must be assigned to an area in the `area_def` array. An example `area_def` array is shown below.

```
%% area_def data format
% should contain same number of rows as bus array (i.e. all bus areas defined)
% col 1 bus number
% col 2 area number
area_def = [ ...
            1  1;
            2  1;
            3  1;
            4  1;
            10 1;
            11 2;
            12 2;
            13 2;
            14 2;
            20 1;
           101 1;
           110 2;
           120 2];
```

It should be noted that rows may not have to be in the same order as the bus array (untested). The `area_def` array is automatically placed into the global g structure.

```
>> g.area
ans =
    area_def: [13x2 double]
      n_area: 2
        area: [1x2 struct]
```

Each area currently contains values that may be relevant to AGC calculations. The `calcAreaVals` function is used to calculate and store such values. An example of what is stored in the `g.area.area(x)` structure is shown below.

```
>> g.area.area(2)
ans =
         number: 2
      areaBuses: [6x1 double]
         macBus: [2x1 double]
      macBusNdx: [3 4]
        loadBus: [4x1 double]
     loadBusNdx: [8 9 12 13]
         genBus: [2x1 double]
      genBusNdx: [6 7]
           totH: [1x4063 double]
           aveF: [1x4063 double]
         totGen: [1x4063 double]
        totLoad: []
            icA: [1x4063 double]
            icS: []
  exportLineNdx: [11 12]
  importLineNdx: []
       n_export: 2
       n_import: []
```

It should be noted that `icS` represents a placeholder for a scheduled interchange value and the `totLoad` is a field for collected total load. The collection of actual running load values may prove more complicated as the bus array does not seem to be updated every step, only a reduced Y matrix used in the `nc_load` function called from `i_simu`.

Line Monitoring

Power flow on a line must be calculated each step as AGC requires actual area interchange for the ACE calculation. The previously existing `line_pq` function performed this task, but was not fully incorporated into the simulation to allow calculation during execution. This minor oversight has been resolved and the `lmon_con` array is still used to define monitored lines as in previous version of PST. It should be noted that areas with AGC automatically track area interchange flows and do not need to be user defined.

```
%% Line Monitoring
% Each value corresponds to an array index in the line array.
% Complex current and power flow on the line will be calculated and logged during simulation

%lmon_con = [5, 6, 13]; % lines between bus 3 and 101, and line between 13 and 120

lmon_con = [3,10]; % lines to loads
```

Line monitoring data is collected in the `g.lmon` field of the global variable.

```
>> g.lmon
ans =
     lmon_con: [3 10]
       n_lmon: 2
    busFromTo: [2x2 double]
         line: [1x2 struct]
```

Each `g.lmon.line` entry contains the following fields and logged data:

```
>> g.lmon.line(2)
ans =
    busArrayNdx: 10
        FromBus: 13
          ToBus: 14
          iFrom: [1x4063 double]
            iTo: [1x4063 double]
          sFrom: [1x4063 double]
            sTo: [1x4063 double]
```

Weighted Average Frequency

An average weighted frequency is calculated for the total system and for each area if there are areas detected. The calculation involves a sum of system inertias that may change with generator trips. The current algorithm does not account for tripped generators, but was designed to incorporate this feature in the future.

In a system with N generators, M areas, and N_M generators in area M, the `calcAveF` function performs the following calculations for each area M:

$$H_{tot_M} = \sum_{i}^{N_M} MVA_{base_i} H_i$$

$$F_{ave_M} = \left(\sum_{i}^{N_M} Mach_{speed_i} MVA_{base_i} H_i \right) \frac{1}{H_{tot_M}}$$

Then system total values are calculated as

$$H_{tot} = \sum_{i}^{M} H_{tot_M}$$

$$F_{ave} = \left(\sum_{i}^{M} F_{ave_M} \right) \frac{1}{M}$$

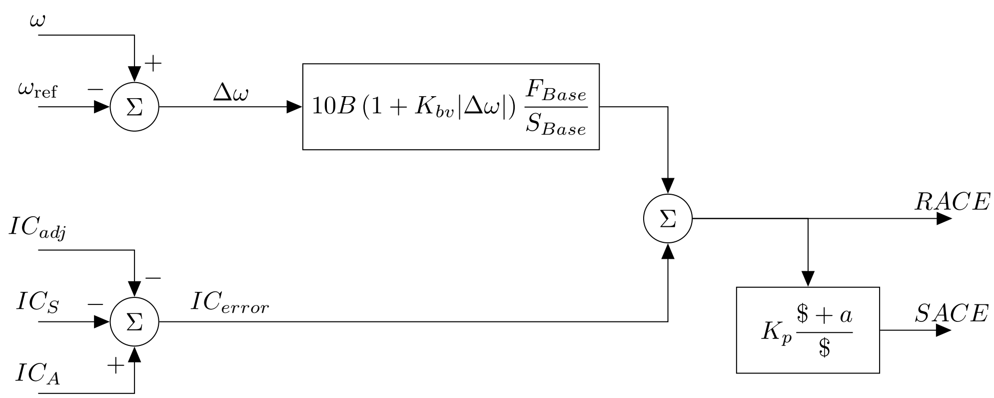
If $M == 0$, `calcAveF` performs

$$H_{tot} = \sum_{i}^{N} MVA_{base_i} H_i$$

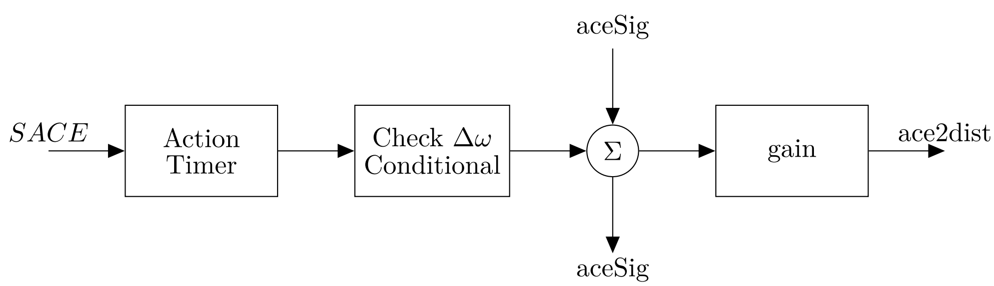
$$F_{ave} = \left(\sum_{i}^{N} Mach_{speed_i} MVA_{base_i} H_i \right) \frac{1}{H_{tot}}$$

Automatic Generation Control

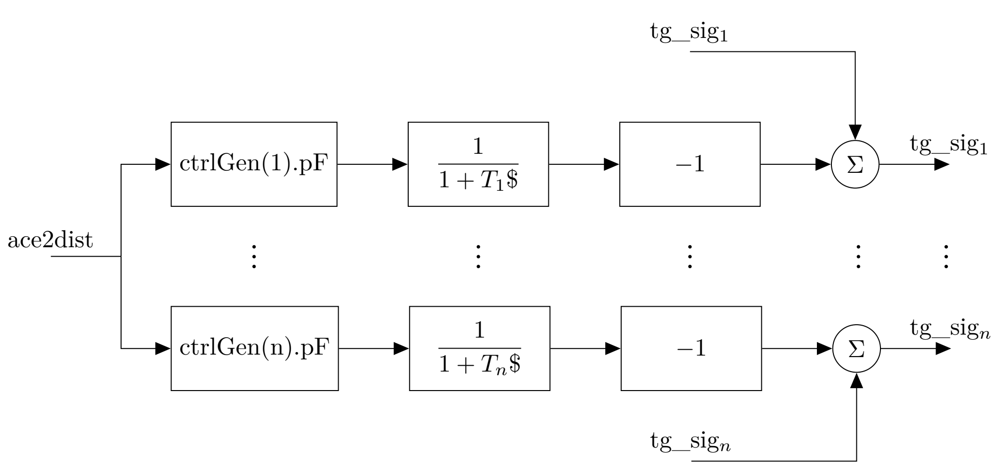
The AGC complete process is shown as the three following block diagrams. RACE and SACE are calculated using PU values assuming B is a positive non-PU value with units of $MW/0.1Hz$ If K_{bv} is not zero, the resulting $RACE$ is not the industry standard $RACE$ value. The scheduled interchange may be altered by the user via a `mAGC_sig` file that controls the behavior of the IC_{adj} input.



$RACE$ and $SACE$ are calculated every time step, however distribution of $SACE$ is determined by the `startTime` and `actionTime` variables. Assuming action, the conditional $\Delta\omega$ logic is processed before adjusting the `aceSig` value which is then gained to become `ace2dist`.



The `ace2dist` value is distributed to all controlled generators associated with the AGC model according to their respective participation factor `pF`. Each `ctrlGen` has a unique low pass filter that processes the signal that is then gained by -1 and added to the existing associated governor `tg_sig` value.



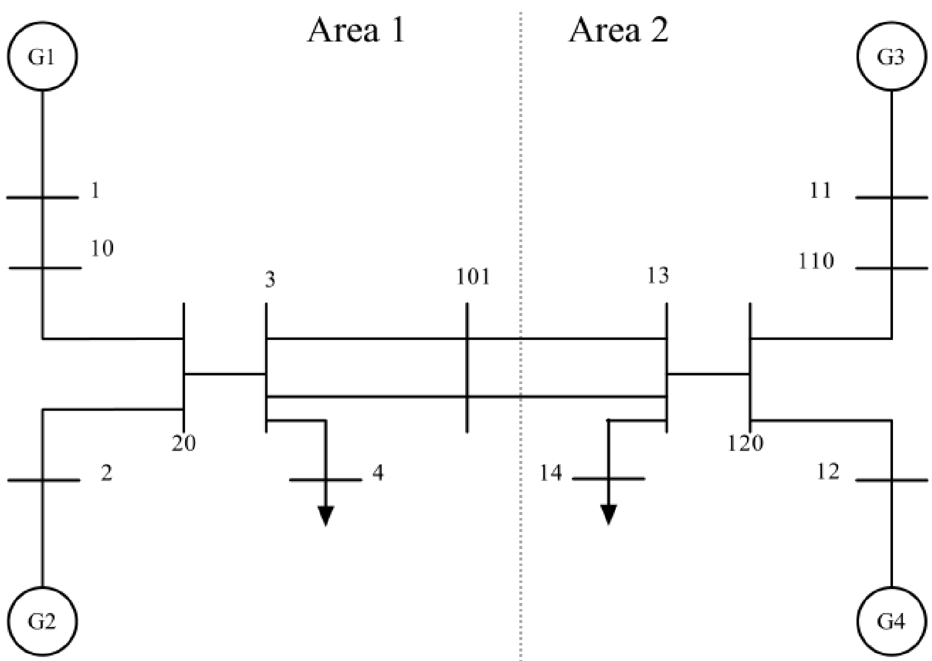
Example AGC settings

The following AGC settings are not entirely realistic, but useful in demonstrating the required user definitions and functionality of the AGC model.

```
%% AGC definition

%{
Experimental model definition akin to Trudnowski experimental code.
Each agc(x) has following fields:
    area        - Area number / controlled area
    startTime   - Time of first AGC signal to send
    actionTime  - Interval of time between all following AGC signals
    gain        - Gain of output ACE signal
    Btype       - Fixed frequency bias type (abs, percent of max capacity...)
        0 - absolute - Input B value is set as Frequency bias (positive MW/0.1Hz)
        1 - percent of max area capacity
    B           - Fixed frequency bias Value
    Kbv         - Varaible frequency bias gain used to gain B as B(1+kBv*abs(delta_w))
    condAce     - Conditional ACE flag
        0 - Conditional ACE not considered
        1 - ace2dist updated only if sign matches delta_w (i.e. in area event)

    (PI Filter Values)
    Kp          - Proportional gain
    a           - ratio between integral and proportional gain (placement of zero)

    ctrlGen_con - Controlled generator information (see format note below)
%}

agc(1).area = 1;
agc(1).startTime = 25;
agc(1).actionTime = 15;
agc(1).gain = 2; % gain of output signal
agc(1).Btype = 1; % per max area capacity
agc(1).B = 1;
agc(1).Kbv = 0; % no variable bias
agc(1).condAce = 0; % conditional ACE
agc(1).Kp = 0.04;
agc(1).a = 0.001;
agc(1).ctrlGen_con = [ ...
    % ctrlGen_con Format:
    %col 1 gen bus
    %col 2 participation Factor
    %col 3 low pass filter time constant [seconds]
    1, 0.75, 15;
    2, 0.25, 2;
    ];
```

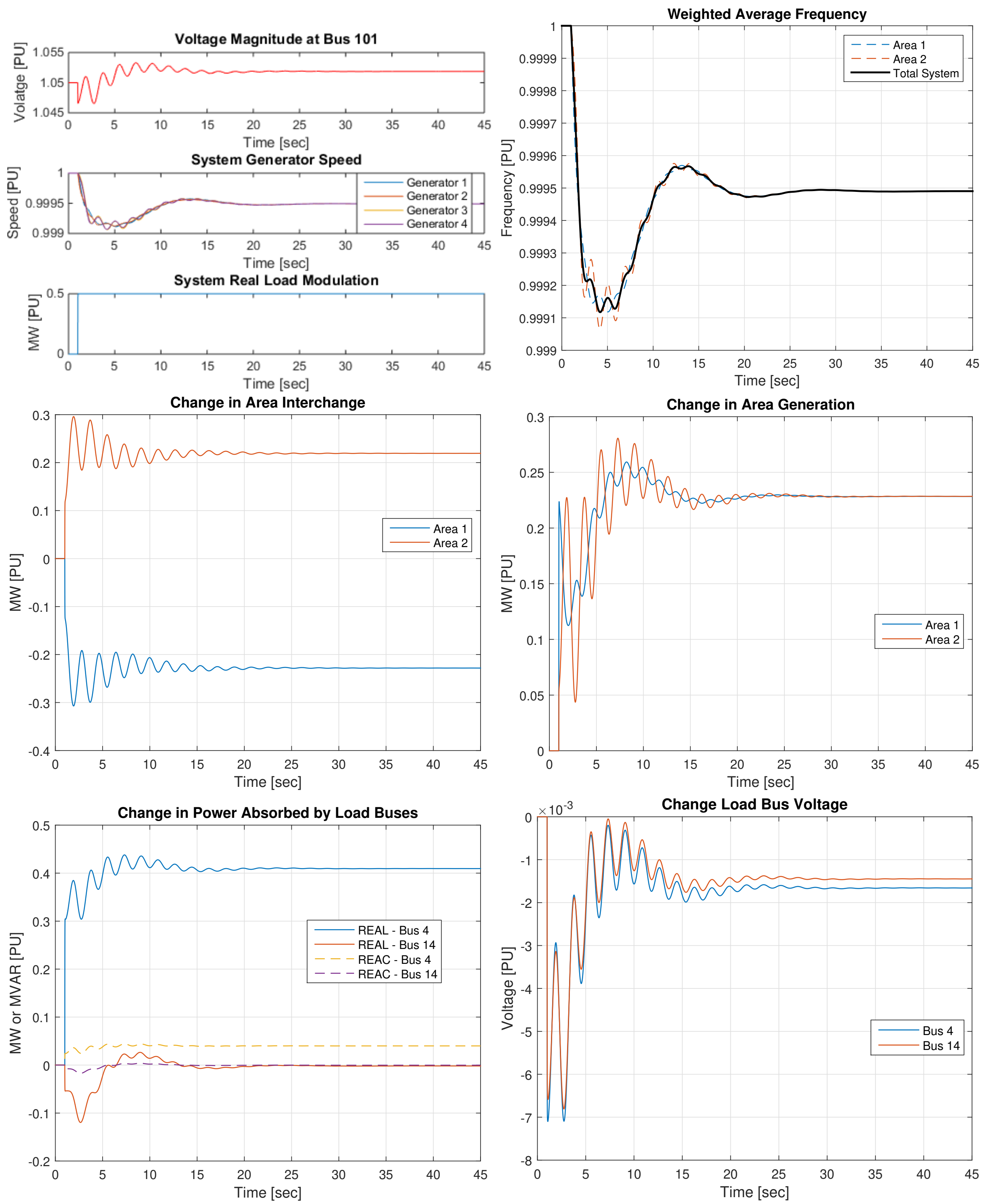
```
agc(2)=agc(1); % duplicate most settings from AGC 1 to AGC 2
agc(2).area = 2;
agc(2).ctrlGen_con = [...
%    col 1 gen bus
%    col 2 participation Factor
    %col 3 low pass filter time constant [seconds]
    11, 0.25, 10;
    12, 0.75, 5;
    ];
```

General Example Scenario Details

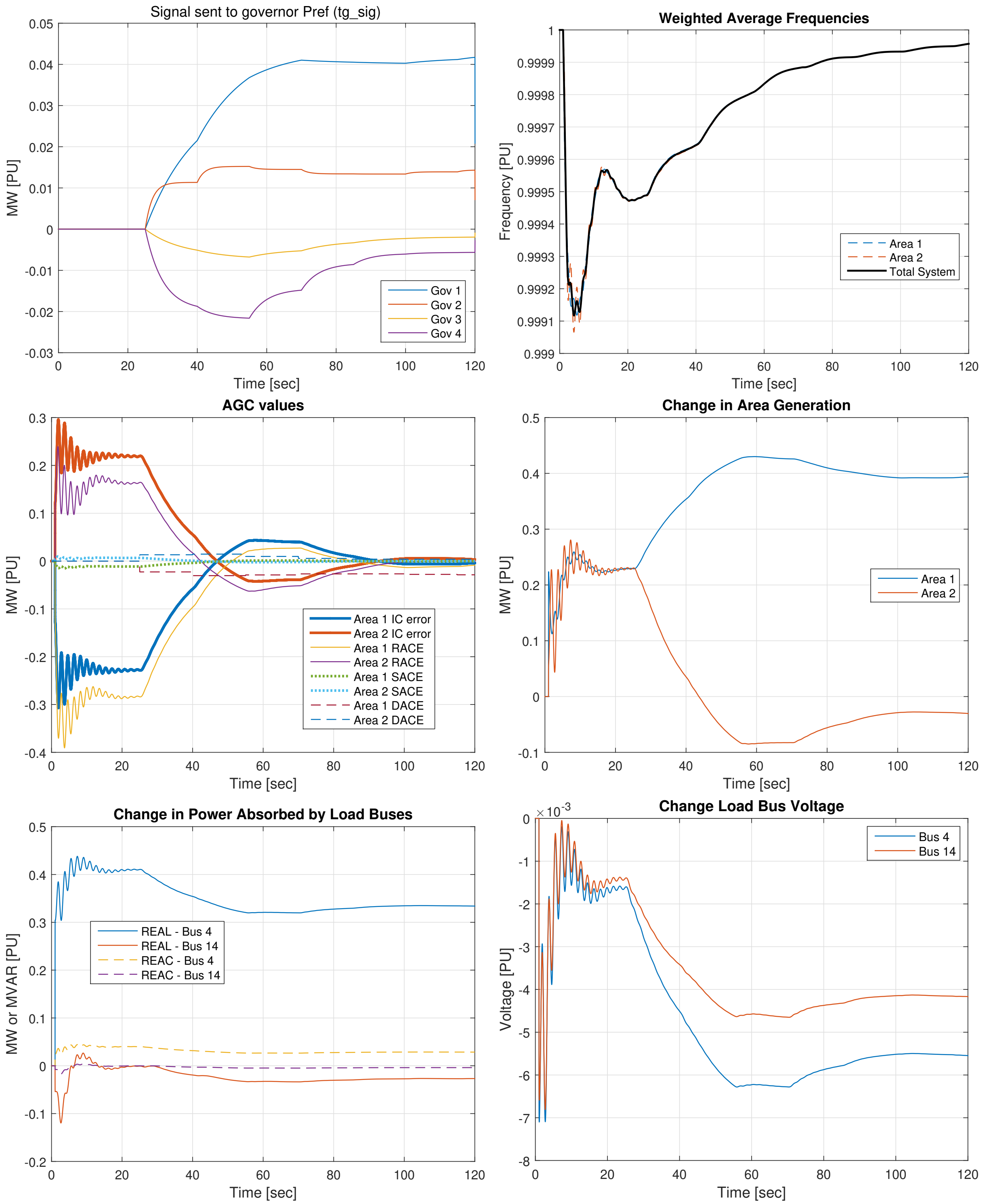


- Kundur 4 machine system packaged with PST
- Constant Z load model
- All machines, exciters, and govs identical
- PSS on gen 1 and 3
- SVC on bus 101
- Event: +50 MW (0.5 PU) step of load on bus 4 at t=1
- AGC model available in pstSETO only

Initial simulation results (No AGC)



Initial simulation results (with AGC)



Initial simulation results (with conditional AGC)