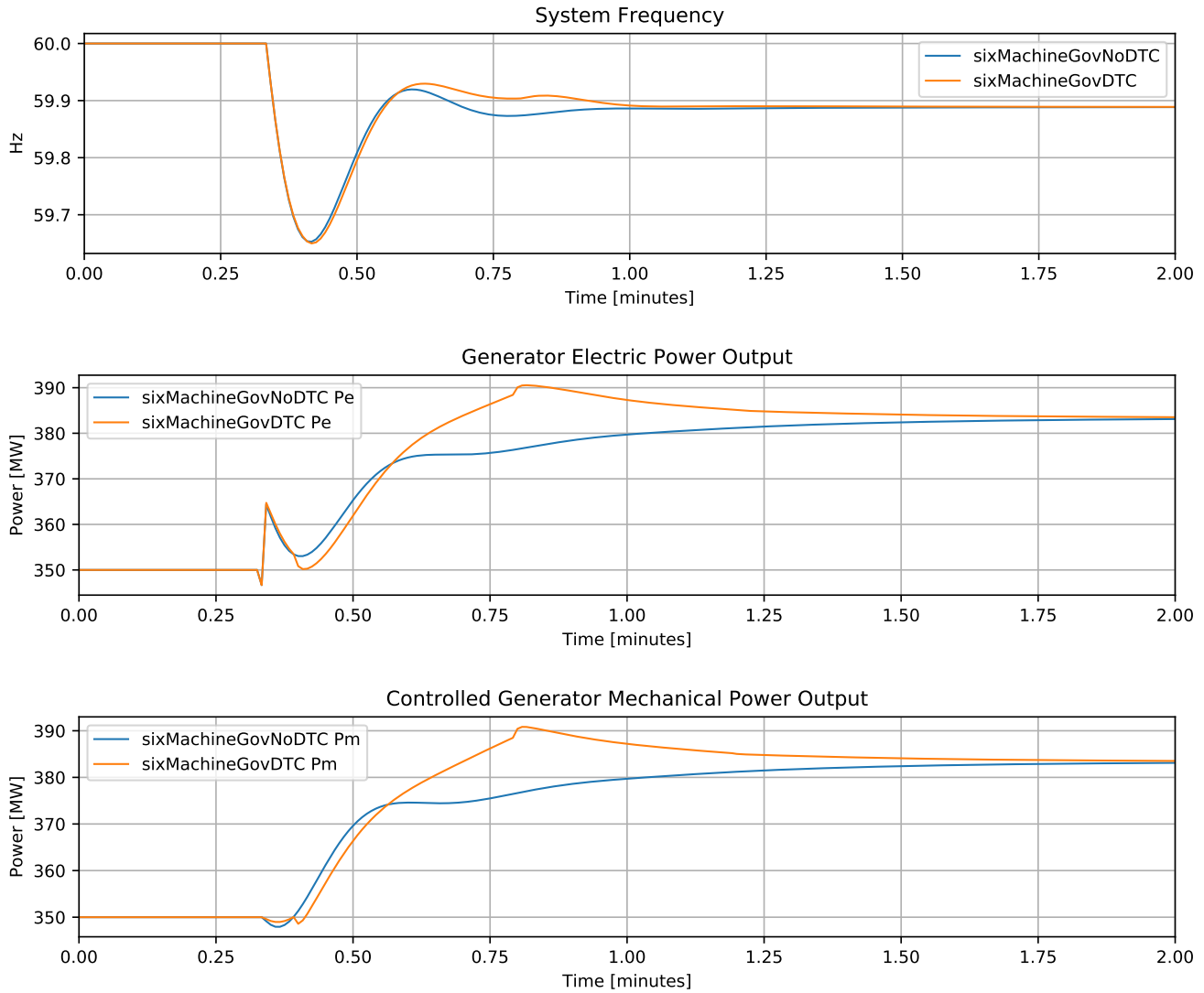


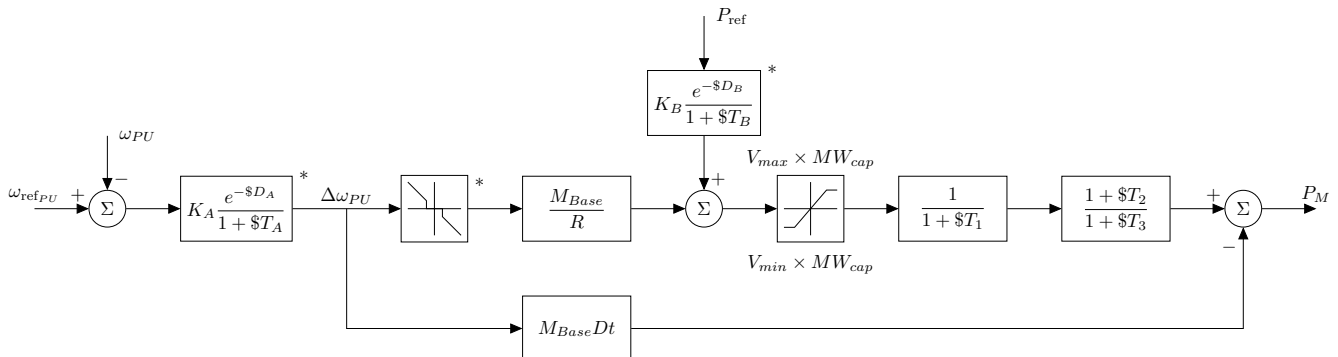
Initial Definite Time Controller (DTC) BPA Result Summary

Using a six machine system with 0.5 second time step, simulation results show that the method of stepping P_{ref} while also gaining input $\Delta\omega_{PU}$ produces similar generator output action provided by BPA as an 'undesirable' response. Remaining differences believed to be caused by differences in model size, time constants, and actual 'feed-forward' action.



Modified Governor Model

Input ω block moved so that gain would be more logical.



Code Example and Explanation Code used to define system step, governor delay, and DTC action is provided below. In practice, this code is user defined in the simulation .ltd.py file.

An ungoverned generator on bus 5 has mechanical power stepped down 100 MW at t=20 to simulate the tripping of a generator. A governor delay block is used to gain the ω input by 0.5. DTC action occurs every 24 seconds (so that first action is near frequency nadir) and sets

$$P_{ref} = P_{ref0} + \frac{\Delta\omega}{R} M_{base} * 0.5.$$

```

1  # Perturbances
2  mirror.sysPerturbances = [
3      'gen 5 : step Pm 20 -100 rel', # Step no-gov generator down
4  ]
5
6  # Delay block used as delta_w gain
7  mirror.govDelay = {
8      'delaygen2' : {
9          'genBus' : 2,
10         'genId' : '1', # optional
11         'wDelay' : (0, 0, .5),
12         'PrefDelay' : (0, 0)
13     },
14     #end of defined governor delays
15 }
16
17 # Definite Time Controller Definitions
18 mirror.DTCdict = {
19     'bpaTest' : {
20         'RefAgents' : {
21             'ra1' : 'mirror : f',
22             'ra2' : 'gen 2 1 : R',
23             'ra3' : 'gen 2 1 : Pref0',
24             'ra4' : 'gen 2 1 : Mbase',
25         }, # end Referenc Agents
26         'TarAgents' : {
27             'tar1' : 'gen 2 1 : Pref',
28         }, # end Target Agents
29         'Timers' : {
30             'set' : { # set Pref
31                 'logic' : "(ra1 > 0)", # should always eval as true
32                 'actTime' : 24, # seconds of true logic before act
33                 'act' : "tar1 = ra3 + (1-ra1)/(ra2) * ra4 *0.5 ", # step Pref
34             }, # end set
35             'reset' : { # not used in example
36                 'logic' : "0",
37                 'actTime' : 0, # seconds of true logic before act
38                 'act' : "0", # set any target On target = 0
39             }, # end reset
40             'hold' : 0, # minimum time between actions (not used in example)
41         }, # end timers
42     }, # end bpaTest
43 } # end DTCdict

```