Individual Final Report

46045 Design and Implementation of Coordinated Distributed Energy Systems

Daniel Brasholt s214676

June 2024

The following sections will concern the fourth test, "Supervisor Controller Dropout".

Section 1 Test Case

The purpose of this was to observe the system response to one of the failure modes to which we set out to become (more) resilient. As written in the group report, all of the controller units, as well as the supervisor, was activated during the test. Wind production was also active. As such, all units were under observation, however specifically the system daemon was the object under investigation, as this was our way of implementing a system response to failure in the form of loss of a controller. The system daemon should be passive during normal modes of operation, only receiving supervisor heartbeats, and should only kick into action when failure occurs.

Our use case, as specified in the group report, concerns the hybrid power plant as a means of frequency regulation with compensation for the fluctuations normally associated with renewable energy production. Therefore, the system should not be disruptive in case of a single controller breaking down; if that were the case, a single operator mistake could be rather costly to the rest of the grid participants.

Because of these concerns, the main metrics with which the outcome is measured is the $RMSE$ from the expected frequency response during a supervisor controller dropout, as well as a more qualitative assessment of the system reaction time to a fault.

Section 2 Test Specification

This test begins with the system in a normal operating state. The individual unit controllers, as well as the supervisor controller, are started. The supervisor controller should then be given some amount of time (in this case 15-20 seconds) to calculate a PCC baseline based on the baseline load combined with median PV production. Once this has been established, regular frequency data should be published to the controller.

After the supervisor has had a chance to behave somewhat normally - in our run, 10 seconds after baseline establishing, the controller process is interrupted. The unit controllers are not directly investigated during this step, however their reaction to the lack of frequency data and split points should be apparent in the general system response. To stress the system further, the battery controller is also dropped after the supervisor has been ressurrected.

The output of the test, from which the $RMSE$ is calculated, is the deviance in the FCR activation from the FCR setpoint, in the span of the test.

Section 3 Test Results

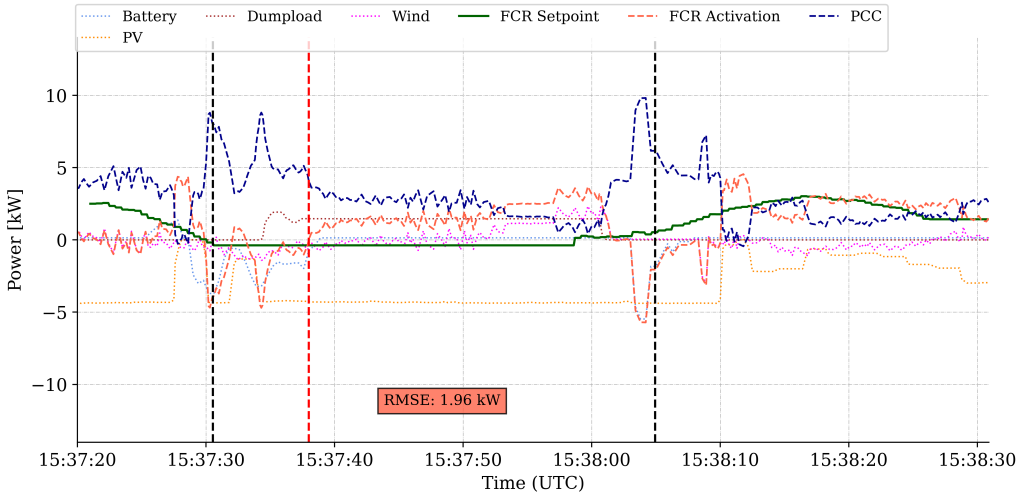

Figure 1: *Results from the dropout test*

The above fig. 1 shows the results from the test. The two dotted, horizontal, black lines specify the times at which controllers were dropped; first the supervisor, then the battery controller. The dotted red line is when the system daemon notices the absence of the supervisor, and revives it by spawning a new process.

From the $RMSE$ value, it seems that the system responds rather well to the supervisor being killed. Further inspection shows that the system shows signs of instability in the period immediately following the controllers being dropped, as should be expected, since it takes some time for the other components to mark the supervisor as deceased. However, it seems that the system stabilizes after a while with only minor fluctuations which could be from the wind turbine, since that is out of our control.

More results can be found in our GitLab repository. In the subfolder

`testing/final_tests_and_results/dropout/console_logs`

the complete logs from the three unit controllers, the supervisor, and the system daemon. It should be noted that when the supervisor is killed, the console output ends; this is since the system daemon, after a set timeout, revives the supervisor by spawning a new subprocess. This subprocess will then be a child of the system daemon, which itself is a child of a specific terminal. Therefore, the console output of the newly spawned supervisor is printed to the terminal containing the system daemon. This is the same for the battery controller. During this test, neither the PV- or load controllers were touched, so these have continuing output all throughout the test.

Section 4 Discussion and Outlook

To conclude our test of controller dropout, the system seems to handle a controller dropout reasonably well. This failure mode is a last-ditch effort in case of absolute emergency; only in extreme circumstances would it be expected for a controller to drop out spontaneously. It is, however, rather comforting to know that should it happen, the system seems to stabilize after a short while.

There is, however, room for improvement. The controllers seem to react violently in the first phases of a supervisor dropout. This may not be able to be mitigated, however it could be an area of interest.

As was also apparent from our other tests, the controllers seem to behave in a competing way, sometimes over-shadowing the other unit controllers. By phasing in controller response following e.g. a sigmoid curve, the system response could perhaps be modified to be a more gentle, gradual reaction.

It should also be noted that this way of spawning new processes only works by spawning said child process on the machine running the system daemon. This detail could of course be changed by changing the mechanism of spawning a controller, however it is a definite limitation of the current design. Due to time constraints, it was not further developed, as the principle of reviving controllers was more important than that controller running on a specific machine.

A next test, which could yield interesting results and conclusions, could be to observe how the system were to react in case of temporary controller dropouts, where the dropped controller came back to life. In this case, the system daemon would probably respond by creating another controller, which would then compete with the old one, however the exact result cannot be recognized without running the test. It could also be interesting to see how the system would act in case of these dropouts happening more frequently, or during periods of frequency volatility.