

An Open WECC Model Set for Grid Code Studies in PSCAD, Built in FORTRAN

Building the WECC REPC_C, REEC_C and REGC_C models as FORTRAN code components in PSCAD, testing each one on its own, then joining them into a 50 MW battery plant on a 33 kV network and running it through a GB Grid Code test programme.

Reference	CS-002
Date	October 2026
Author	Steve Sommerville
Website	stevesommerville.com

Executive Summary

This project is a mix of experimenting with AI to see how far I can push it with PSCAD, and also to address an ongoing issue about the lack of WECC renewable models for PSCAD. Over about two weeks of sessions I had Claude build all three levels of a set of WECC models for a typical energy storage plant (the plant controller, the inverter electrical control and the converter) as FORTRAN code components, written directly into the PSCAD project files with no block diagrams. Each one was tested on its own against a simple test rig and a series of step responses, before moving onto the next module. Ultimately The three were then joined together as a 50 MW / 60 MVA aggregated battery plant on a 33 kV network I built by hand; this was then run through a seventeen case GB Grid Code test suite. The frequency responses land on the Grid Code formulas and the plant rides through all the voltage dips with active power held.

The testing picked up a few interesting things along the way: a post-fault hold that reverses the reactive injection if you implement the published diagram literally, a default parameter that fails the GB 85% voltage dip for 180 s requirement, a weak-grid LVRT oscillation (reproduced, explained and fixed), an initialisation bug, and three interface problems that only showed up once the models were connected to a real EMT network. None of the problems were in the main control maths. Everything was run by Claude interfacing directly to PSCAD through MCP servers and python scripts.

Contents

This table of contents will update automatically in Word.

1. Why Build It.....	2
2. How It Was Built.....	3
3. REPC_C Plant Controller	5
4. REEC_C Electrical Control.....	8
5. REGC_C Converter and the Coupled Model.....	12
6. The Overall Plant Model.....	16
7. What the Models Are Good For.....	22
8. The Process and Who Did What	22
9. Engineering Takeaways.....	23
10. References	23
Appendix A: Models and Test Records.....	24

1. Motivation

The WECC second generation generic renewable models are the standard way of representing inverter based plant in most planning studies. They are split into three blocks. The plant controller (REPC) regulates power, reactive power or voltage at the point of interconnection and sends set-points down to the inverters. The electrical control (REEC) turns those set-points into active and reactive current orders and handles the voltage dip logic and current limits. The generator/converter model (REGC) turns the current orders into what the network actually sees. Every RMS package has them built in, and a lot of what the industry thinks it knows about how IBR plant behaves has come from them.

PSCAD doesn't have them. If you want to do EMT work on a generic plant you either use an OEM model (compiled, encrypted and usually tied to one firmware version) or you build something yourself from library blocks, which for models this size is slow, tedious and easy to get wrong. What I wanted was an open set of WECC compatible models in PSCAD, written as code that can be read and checked, set up from the published WECC parameter lists, and good enough to use for Grid Code type studies on a realistic network.

There were three questions behind it:

- Can the whole WECC control stack be built in PSCAD as FORTRAN code components, without drawing a single block diagram, and be shown to be correct to a standard I would accept on a real project?
- Once the levels are joined together on a proper EMT network, do they behave well enough to run a GB Grid Code test programme, and what do they tell you about the models along the way?

- How much of this can the AI do on its own? Writing the code, coming up with the test plans, running the simulations and finding and fixing the bugs, with me setting the direction and checking the results rather than driving the mouse.

The individual models were written up as Technical Notes 004 (REPC_C) and 005 (REEC_C and REGC_C). This case study pulls the whole thing together: how each level was built and tested, how they were joined up, how the complete plant performs against the Grid Code, and what I learned about the models and the process.

2. How It Was Built

2.1 The platform and interface

The platform is PSCAD 5.0.2 with the Manitoba Hydro Python automation library, connected to Claude through an MCP server. PSCAD runs as normal on the desktop and Claude connects to it in the background. It can load projects, change any parameter on any component, build, run, and read back the build messages and the output files. The one thing the automation library can't do in this version is create components, so the models were written directly into the project XML files. Each WECC model is a single user-defined component, and its behaviour is a page of fixed-form FORTRAN in the component's script. The ports, parameter dialog and graphics are all written as XML alongside it. In PSCAD it looks like any other component, a box with pins and a parameter sheet, but there is no block diagram underneath it, just code.

As a human I like building models from transfer functions and blocks. It can be a fun process, but it is slow, and bug-hunting through a few hundred connected blocks can be very frustrating. Writing the model directly in code is quicker and a lot easier to check. A WECC block diagram translates fairly directly into state equations, and FORTRAN has been around long enough that its rules are well known. FORTRAN is also quite a few years older than me and I am not a fan of coding, so this is exactly the sort of job I wanted the AI to do.

```
! ----- voltage dip flag and post-dip hold timers
      IDIP = 0
      IF (STORF(NS+21) .GT. 0.5) THEN
        IDIP = 1
        IF (($VT .GE. $VDIP+$VHYS) .AND.
&         ($VT .LE. $VUP-$VHYS)) IDIP = 0
      ELSE
        IF (($VT .LT. $VDIP) .OR. ($VT .GT. $VUP)) IDIP = 1
      ENDIF
      ...
      IFRZ = IDIP
      IF (STORF(NS+15) .GT. 0.0) IFRZ = 1
```

Figure 1. A few lines of the REEC_C component as it sits in PSCAD.

2.2 Testing and validation

Each model was developed in its own small PSCAD project, with the component, a simple stand-in for whatever it would eventually be connected to, and some signal sources to drive it. One of the nice surprises came early on. Instead of chaining together lots of library step, ramp and timer blocks (which is what I would have done), Claude wrote its own small sequencer component. Each one has five levels and four switching times in a single dialog, and later there was a version with a ramp rate for frequency ramps. This is what makes the whole approach scriptable, because each test case is just a handful of numbers in a few component dialogs.

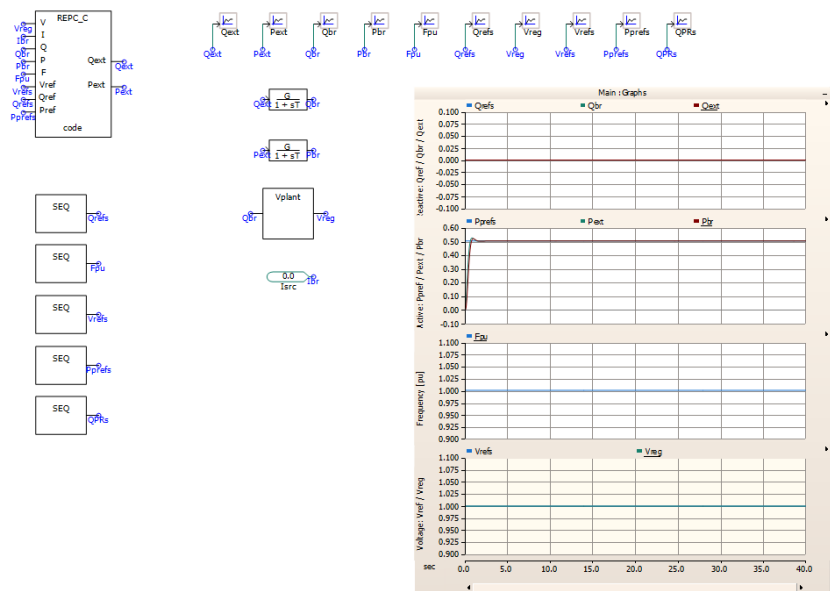


Figure 2. The REPC_C test harness in PSCAD.

2.3 Module Testing

Each level went through the same three stages of testing before it was joined to the next one:

- Calibration against hand calculations. Step changes in the set-points the way an operator would dispatch the plant: P to rating in both directions, MVar and power factor steps, and voltage set-points. The settled value at every step is checked against an operating point worked out by hand or by formula before the run.
- A replica check. For the plant controller Claude wrote a separate Python version of the block diagram and fed the recorded PSCAD inputs through it. If the FORTRAN does exactly what the diagram says, the two agree to machine precision.
- Event testing aimed at the switching logic: voltage dip entry and exit, post-fault holds, freezes, current limits, rate limits, and the PLL freeze and limits. This is where nearly all of the problems were found.

Figure 3 shows the order the work was done in and the stand-in each level was tested against. Each stand-in was good enough to develop against, but each one also hid something that only showed up at the next step.

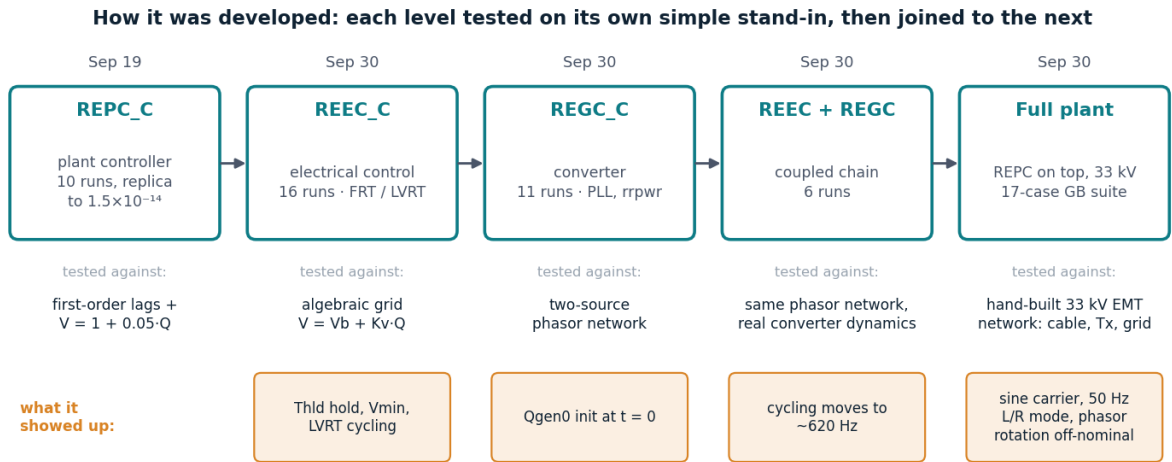


Figure 3. Model Development process

2.4 Running the tests by script

All of the testing was run by script. For each test case Claude sets the parameter values in the running PSCAD project, builds and runs it, reads the raw output files and saves a copy of the run straight away. This includes the full time series, the channel list, and the settings that produced it. PSCAD overwrites its output files every time it runs, so saving each run is important, and every figure in this report can be redrawn from those saved runs without opening PSCAD again. EMT with a fixed time step is deterministic, so re-running a saved case gives exactly the same answer, and when we checked this the difference was zero. That makes the whole process auditable, which is what you need for compliance work.

There were a few quirks along the way that are worth noting:

- PSCAD works from a copy of the project held in memory. Any change to the XML file means the project has to be unloaded without saving, otherwise the old copy overwrites the new file. In practice that meant me clicking Unload in PSCAD whenever Claude asked.
- When a build fails PSCAD pops up a dialog box that blocks the automation connection. Claude wrote a small script to find the window and click OK, and then read the build messages.
- The project file format has a few traps. Component IDs have to fit in a 32-bit integer, duplicate IDs break the connections without any warning, FORTRAN lines must stop at column 72, a parameter name that starts with the name of another one gets corrupted, and a local variable can't have the same name as any signal in the project. All of these are now checked automatically by the scripts before a file is loaded.
- Numbered FORTRAN labels (DO 100 ... 100 CONTINUE) clash as soon as a component is used twice, because PSCAD copies the code once for each instance. Only unlabelled DO/ENDDO loops can be used.

3. REPC_C Plant Controller

3.1 Function

The REPC_C is the plant level controller. It measures voltage, current, power and frequency at the point of interconnection and sends a reactive power command (Qext) and an active power command (Pext) to the inverters. It can run in reactive power, power factor or voltage control mode, with droop or line drop compensation, and it also includes the frequency response with its deadbands, droops and active power PI controller.

I gave Claude the published REPC_C block diagram and the WECC guidance notes and it built REPC_C_F. It has ten ports (V, I, Q, P, f and the three references in, Qext and Pext out) and 59 parameters using the WECC names, each with a plain description in the dialog. It includes all three reference modes, droop and line drop compensation, the error deadband and limits, the reactive PI with anti-windup limits, the lead-lag and rate limit, the frequency measurement with its filter, deadbands and up and down droops, the plant P reference with its rate limits, the feedforward and feedback options, the active power PI and output lag, and the voltage freeze. I left out the switched shunt logic, the auxiliary inputs and the per-unit base conversion, as none of them are needed for a single aggregated plant.

3.2 Tuning

In the test harness the loop is closed through simple 0.1 s lags to represent the inverters, and a voltage plant of $V = 1 + 0.05 \cdot Q$. Claude tuned the PI controllers by pole cancellation (putting the PI zero on the main lag and letting the filters set the damping), and every loop met the targets we use at work (rise time under 1 s, overshoot under 5% and settling under 2 s) first time.

Table 1. REPC_C loop tuning against the test harness.

Loop	Gains	Measured response
Reactive / power factor	Kp 0.65, Ki 6.5	rise 0.32 s, overshoot 3.0%, settle 0.60 s
Active power	Kpg 0.33, Kig 3.3	rise 0.79 s, overshoot 4.6%, settle 1.21 s
Voltage (4% droop)	Kp 6.0, Ki 60	rise 0.56 s, overshoot 2.4%, settle 0.67 s

The voltage loop is the one to watch. Its gain depends on the grid stiffness plus the droop (here $K_v + K_c = 0.09$), so the voltage control gains end up about ten times higher than the reactive power gains. This came up again on the full plant (Section 6), where the gains that gave a nice quick MVar loop gave a voltage loop that took 18 seconds to settle.

3.3 Testing

Claude came up with the test plan and checked it with me before running anything. I asked for every function to be tested, and the approach was to dispatch the controller the way an operator would, with step changes across the full range of each control mode in both directions, and then deliberately push every limit, rate limit and deadband at least once.

Table 2. REPC_C test series and results.

Series	Commands	Result
P set-point	1.0 → 0.5 → 0 → -0.5 → -1.0 pu	every step settles on target to about 10^{-13} , including through zero and when charging
Q set-point	unity, 0.98 and 0.95 pf each way at P = 0.5	$Q = P \cdot \tan \phi = \pm 0.102$ and ± 0.164 pu exactly
Power factor	the same steps as a live Q/P ratio	same settled values; the ratio set-point was made a live signal for this
Voltage, 4% droop	+2%, +4%, -2%, -4%	$\pm 2\%$ settles at $Q = \Delta V / (K_v + K_c) = \pm 0.222$; $\pm 4\%$ holds at the ± 0.44 limit
Limits	Q demand ± 0.6 pu with a 0.2 pu/s rate limit	held at ± 0.44 , ramp measured at 0.2000 pu/s, no windup
Frequency while charging	f = 0.99 and 1.01 pu at P = -0.5	-0.312 and -0.688 pu = $20 \times (0.01 - 0.0006)$, exact

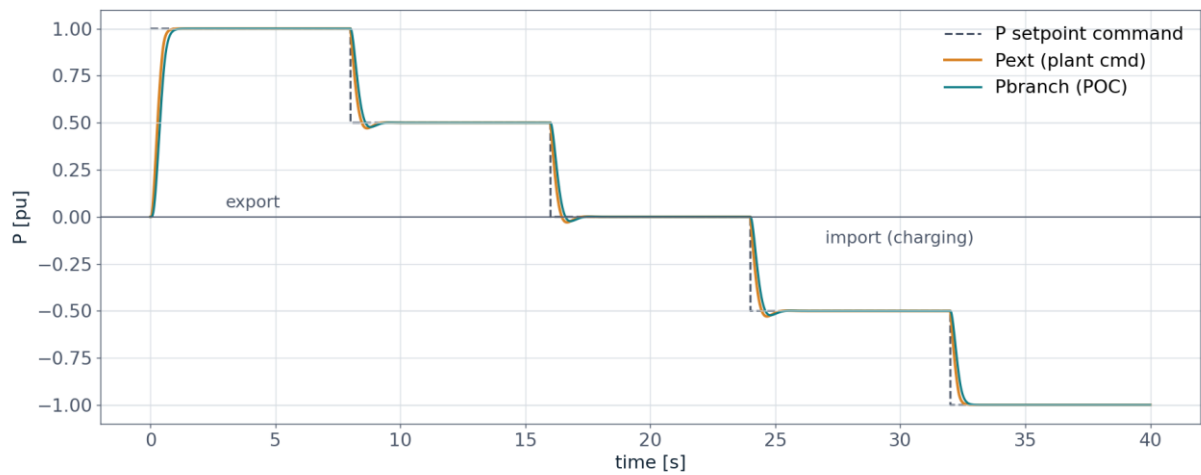


Figure 4. REPC_C active power steps from full export to full import. Every step settles on target to about 10^{-13} pu.

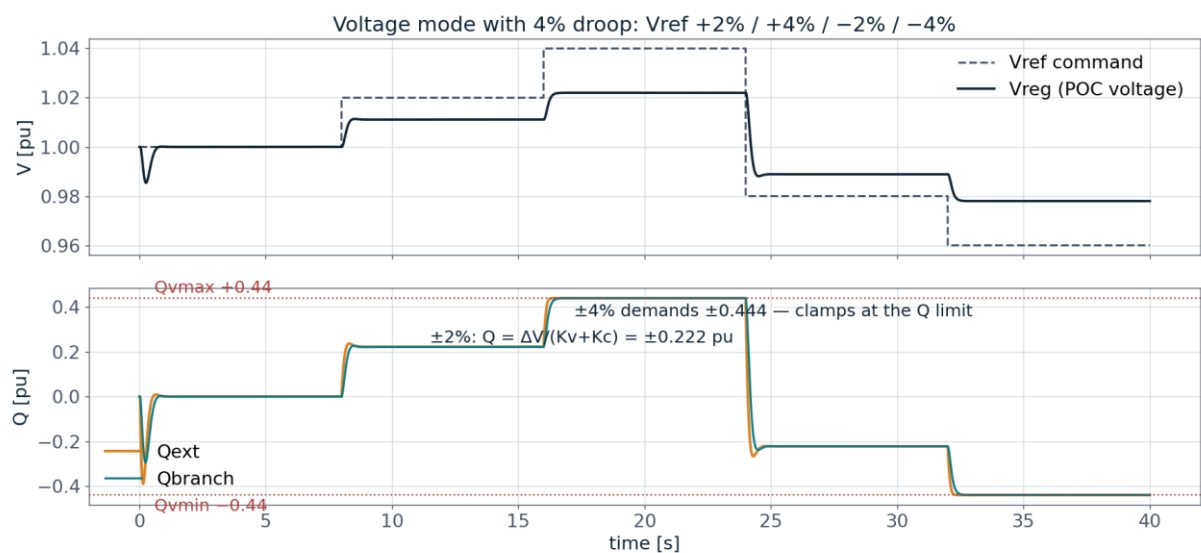


Figure 5. Voltage control with 4% droop. The $\pm 2\%$ steps settle at ± 0.222 pu. The $\pm 4\%$ steps ask for ± 0.444 pu, just over the ± 0.44 capability, so the output sits on the limit and recovers cleanly with no windup.

3.4 Validating

The part I didn't ask for, and which surprised me, was that Claude decided to build its own model of the controller in Python and check the PSCAD model against it. The recorded inputs from every run were fed through the Python version and the outputs compared, which showed tight alignment. The one subtle point was the order PSCAD calculates things in. Signals that are calculated after the controller in each time step (the plant feedbacks) arrive one time step old, while the reference signals arrive in the same step. If you get that wrong you see a small error and if you get it right the error drops by ten orders of magnitude. At that point you can be confident the FORTRAN does what the diagram says.

3.5 Conclusion

- Once the controller was built it worked first time. Most of the effort went into the tools: the ID limits, duplicate IDs, the dialog box problem, and integer parameters needing integer limits in their definition or the automation library rejects them.
- Voltage control gains need tuning for each site, because the loop gain is set by the grid and not the controller.

- With 4% droop, a $\pm 4\%$ voltage step lands exactly on a ± 0.44 pu capability, which makes it a handy limit test.

4. REEC_C Electrical Control

4.1 Function

The REEC_C represents the control system inside each inverter. It takes the P and Q (or voltage) commands from the plant controller and the measured terminal values, and works out the active and reactive current orders for the converter. Most of the interesting behaviour is in this model: the reactive power control options (power factor, local voltage, or a Q and V PI cascade), the voltage dip logic that puts the inverter into fault ride through and injects reactive current, the voltage dependent current limits with P or Q priority, the post-fault holds, and in the C version a battery state of charge (SOC) model that blocks charging and discharging at its limits.

REEC_C_F includes all of this: the PFlag, VFlag and QFlag options, the dip logic with Kqv injection, the current limits with priority (each limit uses the other current from the previous time step, which is how the spec works), the Thld and Thld2 holds and the SOC model. The test harness is deliberately simple. The grid is represented as $V = V_b + K_v \cdot Q$ with 20 ms current lags, so every settled operating point can be worked out by hand by solving a quadratic for the injection against the grid.

4.2 Dispatch and Testing

The dispatch tests went the same way as for the REPC. The Q and power factor steps and the local voltage control all settled exactly, to around 10^{-12} . The battery test used a very small 40 second battery so the SOC limits would act during the run. A full discharge command hit the SOC floor at 7.5 s and the power was forced to zero and held there. Charging refilled it, the upper limit blocked further charging, and discharging released it again, which is exactly what the diagram says should happen. The SOC also matched a hand calculation.

Then came the GB events: 140 ms faults at 10% retained voltage (three phase), plus positive sequence equivalents of the unbalanced faults, and the LVRT tests at 30% for 384 ms, 50% for 710 ms, 80% for 2.5 s and the sustained 85% for 180 s. In the deep dip the reactive injection hits its $I_{qh1} = 1.05$ pu ceiling and the active current is reduced to exactly $\sqrt{(I_{max}^2 - I_{qh1}^2)}$. Across all the events the measured injection sits on the Kqv line, and more than twenty settled operating points matched the hand calculations, usually to four decimal places.

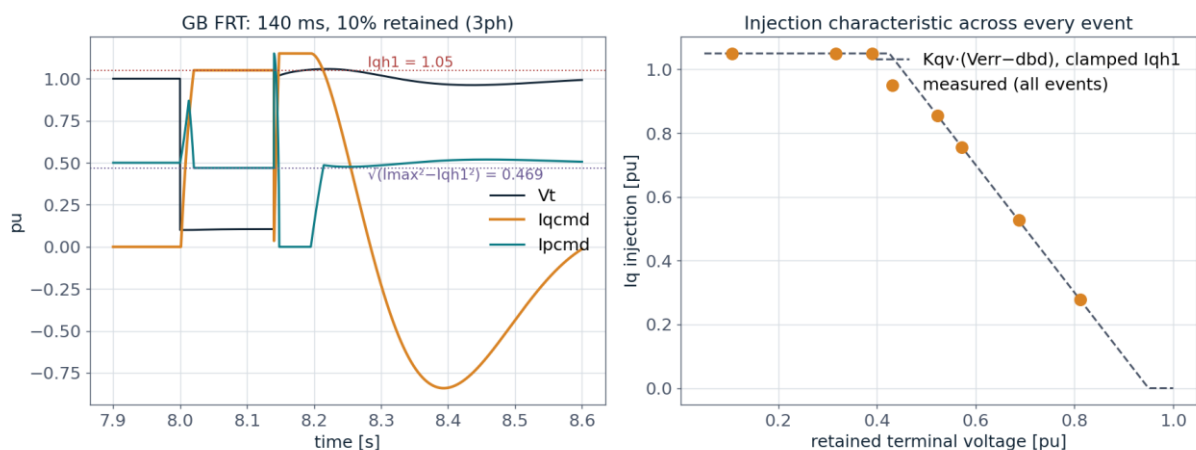


Figure 6. GB FRT through REEC_C.

One point that applies from here on is that the WECC models are positive sequence only. There is no negative sequence injection, which the GB requirements for unbalanced faults expect from real plant. This isn't a bug, it is a limitation of the model, and I come back to it in Section 7.

4.3 Finding 1: the post-fault hold reversed the injection

The first version held the reactive injection for the T_{hld} time after the dip cleared, as shown on the diagram, but the freeze on the controller states was tied to the dip flag only, which is also how the diagram reads. The result was that when the dip cleared, the voltage PI controller started up again while the injection was still being held, wound up against it, and 300 ms after the fault cleared the reactive current command had reversed to -0.39 pu. The plant was absorbing reactive power just when the network was trying to recover. The sensible way to read the diagram is that the whole ride through state (the freeze and the injection) is held for T_{hld} , and that is how it was fixed. This is a gap in the spec rather than a coding error, and I suspect it is part of the reason different RMS packages disagree on how REEC behaves after a fault.

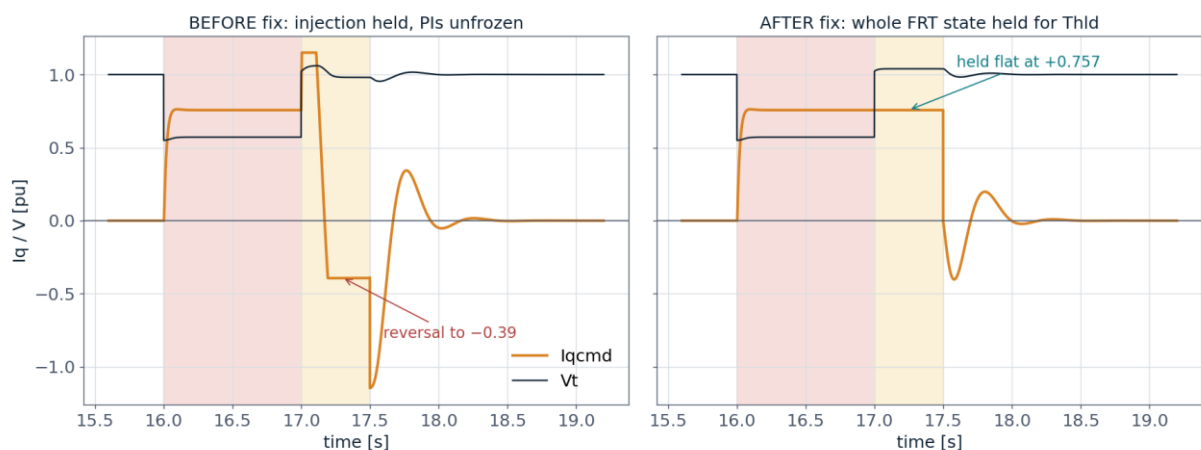


Figure 7. The same fault before and after the hold fix.

4.4 Finding 2: V_{min} and the sustained 85% voltage dip

At 85% retained voltage, which is the GB 180 second case, the ride through logic never operates because the dip threshold is 0.85 and the voltage sits right on it. What decides the result is V_{min} . Despite its name, V_{min} is not a ride through setting. It is the lowest voltage the reactive power controller is allowed to aim for. With the default setting of 0.90, the controller won't accept the lower voltage and pushes in about 1.1 pu of reactive current to try and pull the terminal voltage back up to 0.90. The current limit then cuts the active power from 0.50 to 0.27 pu for the whole of the dip. If V_{min} is set below the dip voltage the plant rides through at full output.

The 85% sustained sag, run twice, the only difference is the controller's internal voltage-command floor ('Vmin', not an FRT setting)

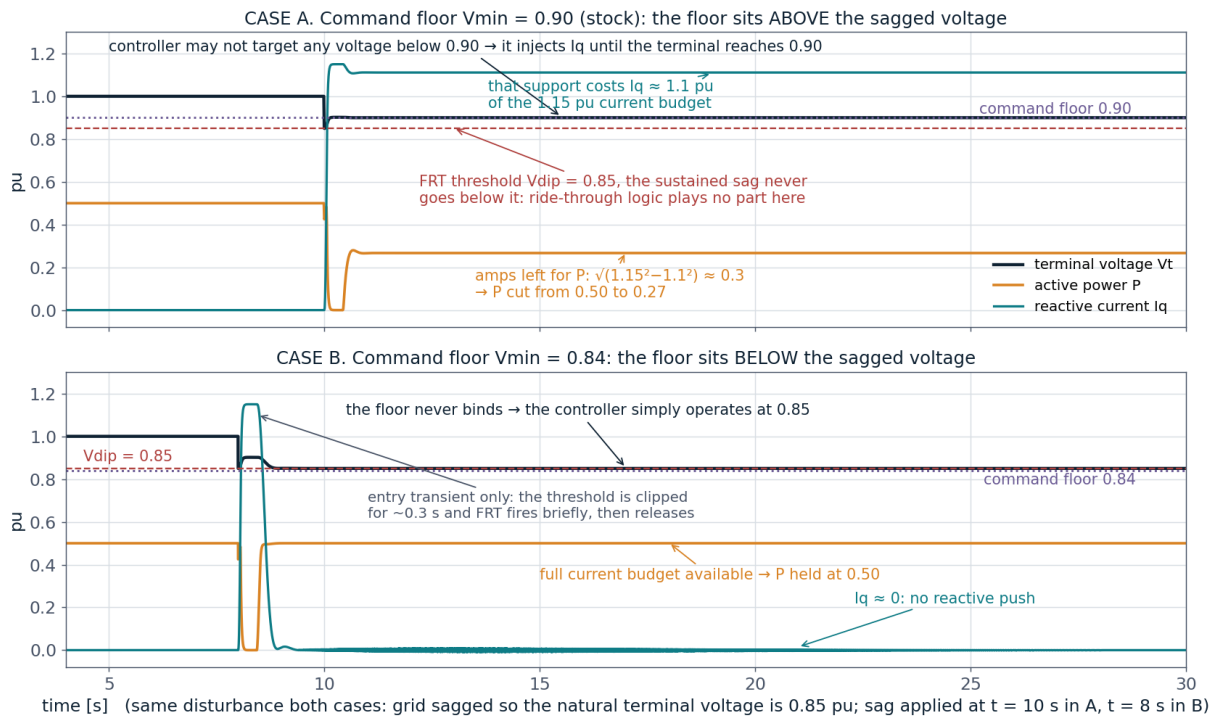


Figure 8. The sustained 85% dip run twice.

This is something I have come across a number of times with OEM models in the day job, and it has caused a fair bit of head scratching, as it is not always clear what all the parameters actually do. With the default value the model fails the GB requirement, and the parameter name gives no warning of it.

4.5 Finding 3: LVRT oscillation on a weak grid

This is a known, but uncommon, problem on weak grids, and Claude found it without being asked. In fixed Q or power factor mode the dip flag gives the plant two possible operating points: the terminal voltage with the reactive injection and the terminal voltage without it. If the grid voltage is at a level where one of these is below the 0.85 threshold and the other is above it, the plant can't settle in either. It enters the dip, injects reactive current, gets pushed back above the threshold, stops injecting and drops back in again. The WECC spec has no hysteresis or time delay on the flag, so there is nothing to stop it cycling.

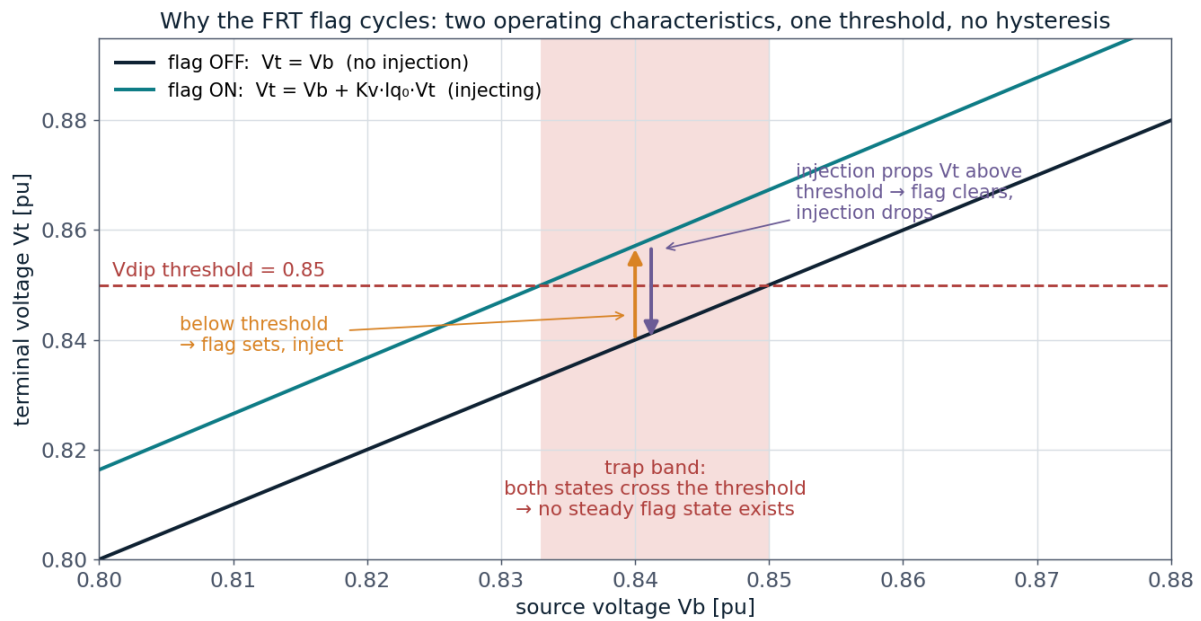


Figure 9. Why the flag oscillates.

The band of grid voltages where this happens can be predicted. It runs from $V_{dip} \cdot (1 - K_v \cdot I_{q0})$ up to V_{dip} , so its width depends on the grid stiffness K_v , which is why it is a weak grid problem. Claude ran a sweep of grid voltage across the band (with $K_v = 0.10$ the predicted edges are 0.833 and 0.850) and the edges matched exactly. Below the band the plant latches into the dip and gives steady support, above it there is no dip, and the cycling only happens inside the band.

The sweep also showed two other things. Inside the band the cycling settles into a pattern where the on/off ratio adjusts itself so that the average injection holds the terminal voltage at exactly 0.8500. The measured average matches $(V_{dip} - V_b) / (K_v \cdot V_{dip})$ to three decimal places, so the flag is effectively doing rough voltage regulation. The second thing is that the fix needs to be sized properly. Claude added an exit hysteresis parameter (V_{HYS}), so the plant enters the dip at V_{dip} but only leaves above $V_{dip} + V_{HYS}$. It defaults to zero so the published behaviour is unchanged. The hysteresis needs to be bigger than the voltage rise caused by the injection itself, $V_{HYS} > K_v \cdot K_{qv} \cdot (V_{ref0} - V_{dip} - dbd) \cdot V_{dip}$, which is about 0.017 here. If it is too small (0.008 was tried) it doesn't fix the problem, it just slows the cycling down to a sawtooth at about 9 Hz, which is the typical signature of flaky FRT seen on site. At 2% the flag latches on the first entry and the plant gives steady support until the dip clears.

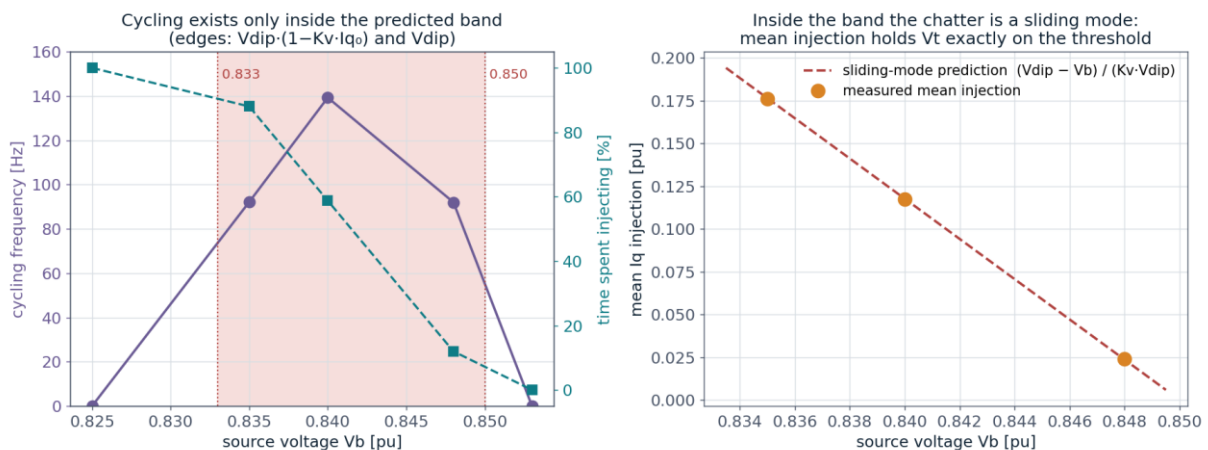


Figure 10. The band mapped by the sweep.

The cure and its sizing: exit hysteresis must exceed the injection’s voltage prop so $VHYS > K_v \cdot K_{qv} \cdot (V_{ref0} - V_{dip} - dbd) \cdot V_{dip} \approx 0.017$

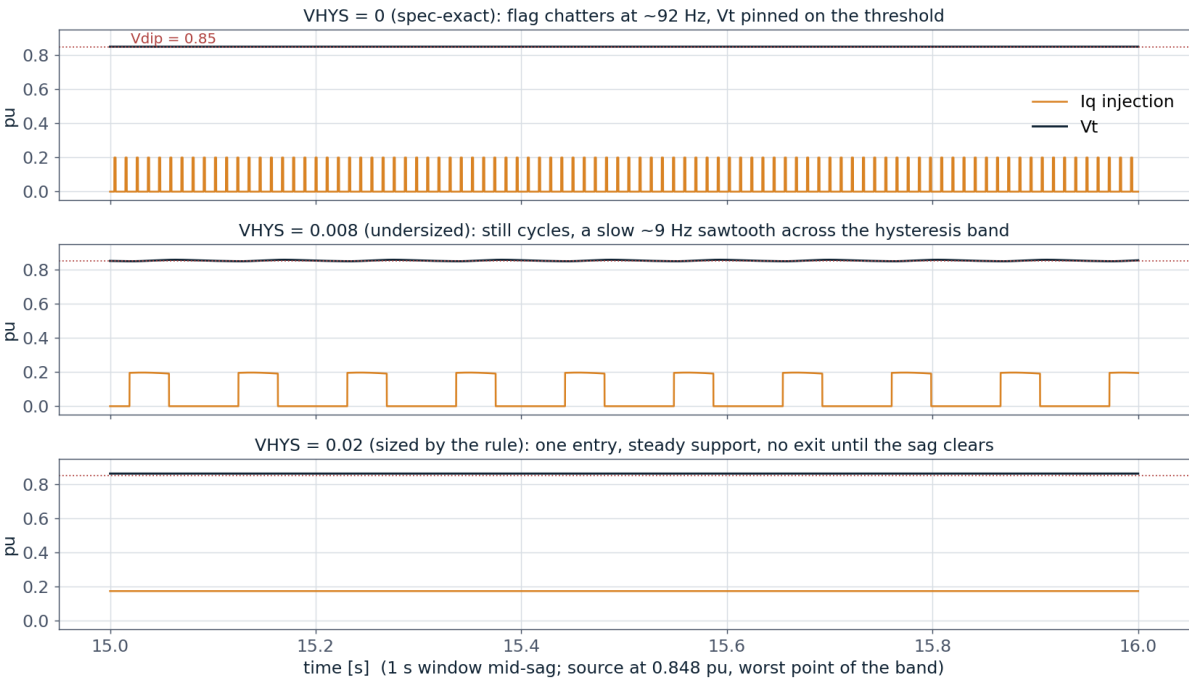


Figure 11. The fix and why it needs sizing.

In voltage control mode the oscillation never happens, because the fast PI controller adjusts the injection and holds the plant on the threshold. That probably explains why the plants that hunt in the field tend to be the ones running in fixed Q mode.

Table 3. REEC_C findings.

Finding	Type	Action
Thld held the injection but unfroze the PIs, reversing the post-fault injection	gap in the spec	freeze covers the dip and the hold
Vmin of 0.90 loses P in sustained dips	parameter setting	set Vmin below the deepest sustained dip
LVRT/HVRT cycling on weak grids in fixed Q mode	missing from the spec (no hysteresis)	VHYS added, off by default
Frozen PI states make the ride through result depend on history	built into the freeze	worth knowing for repeat tests
No negative sequence injection	model limitation	OEM model or an extension for unbalanced work

5. REGC_C Converter and the Coupled Model

5.1 Function

The REGC_C is the generator/converter model and the first level that connects to the network. Unlike the older REGC_A, which is a current source, REGC_C is a voltage source behind an impedance ($R_e + jX_e$). It takes the current orders from the REEC, passes them through inner current PI controllers, a ramp limiter and some lags, and sets the internal voltage on a reference frame from a phase locked loop (PLL). It is still a deliberately simple positive sequence model, and this is the level where OEM models start to differ from it, as it is where the hardware comes in.

REGC_C_F has the model's nine states: the current PI controllers (limited at $\pm I_{max}$), the voltage filter, the reactive current recovery rate limit, the active power ramp limit (in current or power depending on RateFlag), the voltage lags, and a PLL with a low voltage freeze and frequency limits. To test it on its own Claude wrote a second small component called NET. This is the converter voltage behind $R_e + jX_e$ connected to a grid source whose voltage and phase angle are driven by sequencers, so faults, voltage dips and phase jumps are all just numbers in a dialog.

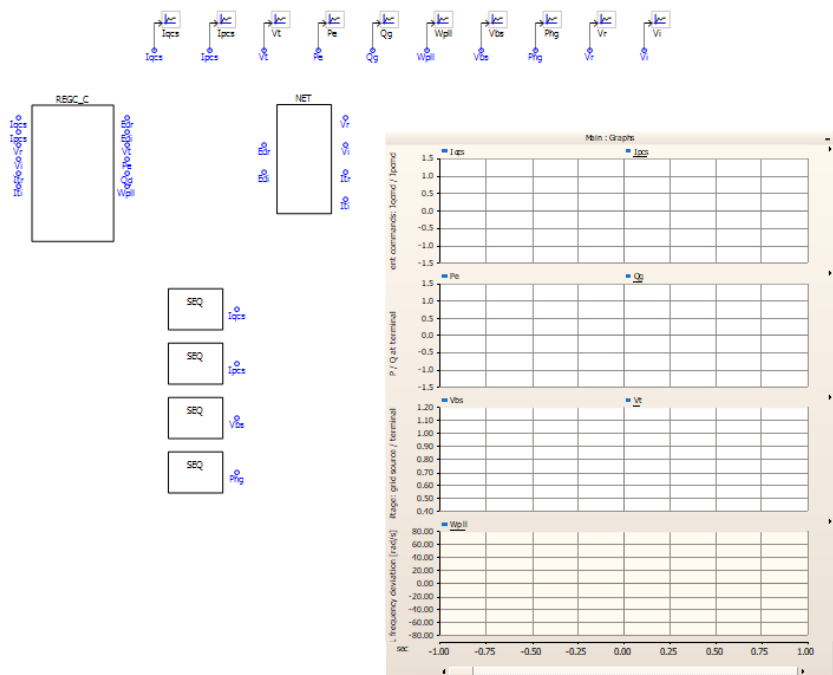


Figure 12. The REGC_C test bench in PSCAD

5.2 Testing

Table 4. REGC_C tests on its own.

Test	Result
Initial run, $I_p = 0.5$ into $0.005 + j0.05$	settles on the hand-calculated network solution: V_t 1.00219, P_e 0.50109, $Q = 0$
I_q steps 0, ± 0.3 , ± 0.6 and I_p steps ± 1.0 , ± 0.5	every step exact: $Q = V_t \cdot I_q$ and $P = V_t \cdot I_p$ to 5 decimal places
Active ramp limit 0.25 pu/s, command ± 1.0	all four ramp cases behave as the spec says
RateFlag = 1 at $V_t = 0.70$	ramp measured at exactly 0.25 pu/s of power, not current
Dip to 0.30 pu (below the PLL freeze)	PLL integrator freezes, proportional part keeps tracking, clean recovery
$\pm 30^\circ$ phase jumps; frequency limit of 10 rad/s	back in sync in about 0.25 s, peak 21.8 rad/s; limit holds at exactly 10.0
I_q command of 1.5 pu, both directions	held at exactly $\pm I_{max} = \pm 1.15$

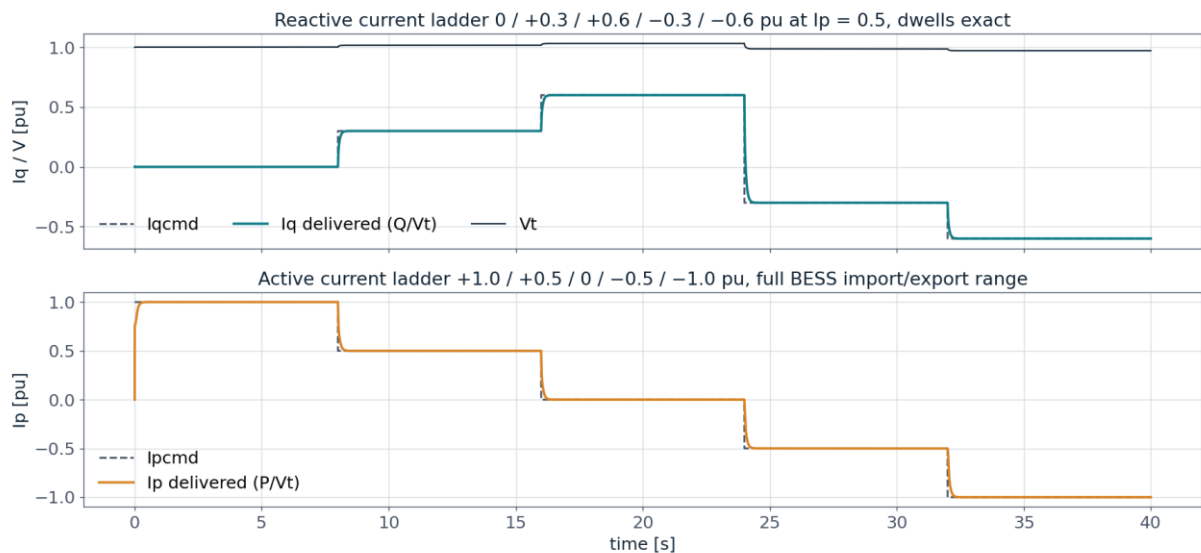


Figure 13. Current order tracking across the operating range: reactive current steps at half load, then the full battery active current range through zero. The voltage moves with Q exactly as the source impedance says it should.

Two of the results looked odd at first but turned out to be how the model is meant to work. The active ramp limit is not the same in each direction (it limits increases while positive and decreases below zero, and is free otherwise), and the way its lag is arranged allows a short step through zero. Also, in a deep dip with the PLL integrator frozen, the steady state needs a small voltage error, which shows up as a -0.005 pu Q offset. This is the model's real behaviour and not an error.

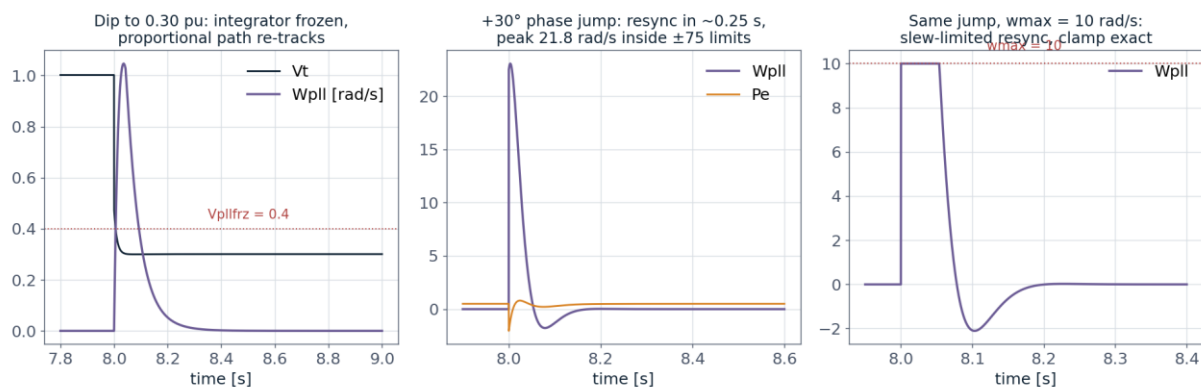


Figure 14. The PLL tests: integrator freeze in a deep dip, a 30° phase jump, and the frequency limit.

5.3 Finding 4: a recovery limit that never switched on

The REGC_C reactive current recovery limits only act depending on whether the reactive power was positive or negative at the start. The first version checked this at time zero, before the network part of the simulation had run, so the currents were zero and both limits stayed switched off for the whole run. The 0.8 s recovery ramp that the spec asks for simply didn't happen. Checking on the first proper time step fixed it, and the retest gave a slope of exactly 0.500 pu/s. It was a one line fix, but there is a general lesson in it: don't sample network values at time zero when initialising a WECC model, because the order PSCAD runs things in means they will be zero.

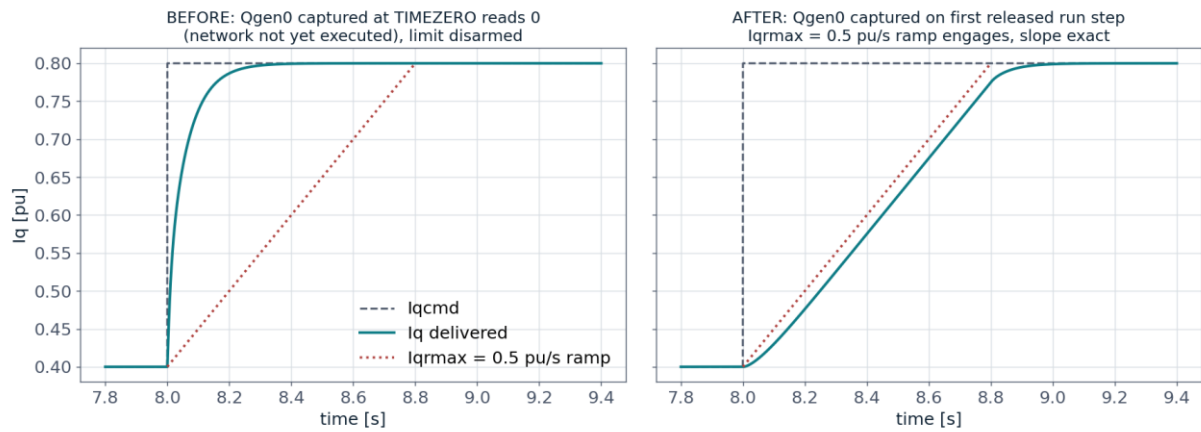


Figure 15. The recovery limit before and after the initialisation fix.

5.4 Joining REEC_C and REGC_C together

Connecting the two models takes five signals: the converter's terminal voltage, power and reactive power go into the REEC, and the REEC's two current orders go into the converter. That is exactly the boundary between the two WECC blocks. The joined project was built by copying the tested component definitions unchanged from the two separate projects, so nothing was re-typed or re-tuned. The first run gave $P_e = 0.50000$ exactly (the REEC divides its power order by the voltage and the converter multiplies it back), with the PLL settled and the SOC discharging at exactly the right rate.

All of the earlier results repeated through the converter. The dispatch is still exact to five decimal places, and P now stays steady through the Q steps because the converter's reference frame separates them. The 140 ms FRT holds the injection at 1.05 pu with the active current at 0.469, and the 85% dip settles at the V_{min} point with the terminal voltage held at 0.900 and the power at 0.500.

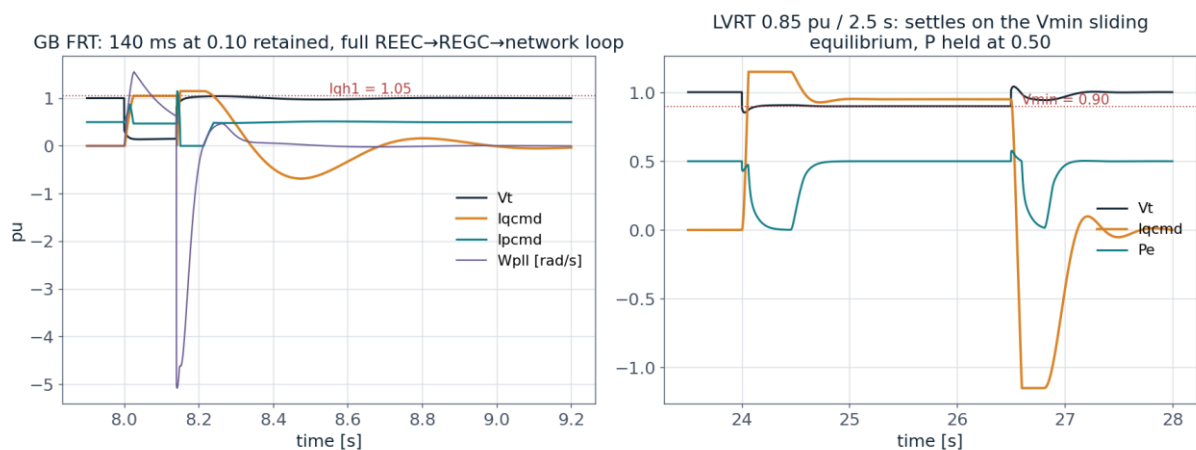


Figure 16. GB events through the joined REEC and REGC model

The one thing that changed was the LVRT oscillation. On a weak grid in fixed Q mode, the flag now cycles at about 620 Hz through the converter, compared with around 140 Hz against the simple grid, because the converter's fast inner loops tighten up the feedback. The VHYS fix, sized with the same rule, worked unchanged: one entry, steady support, a clean exit and no repeats over 20 seconds.

Weak grid ($X_g = 0.2$), fixed-Q mode ($Q_{\text{flag}} = 0$), source at 0.84 pu, just inside the trap band

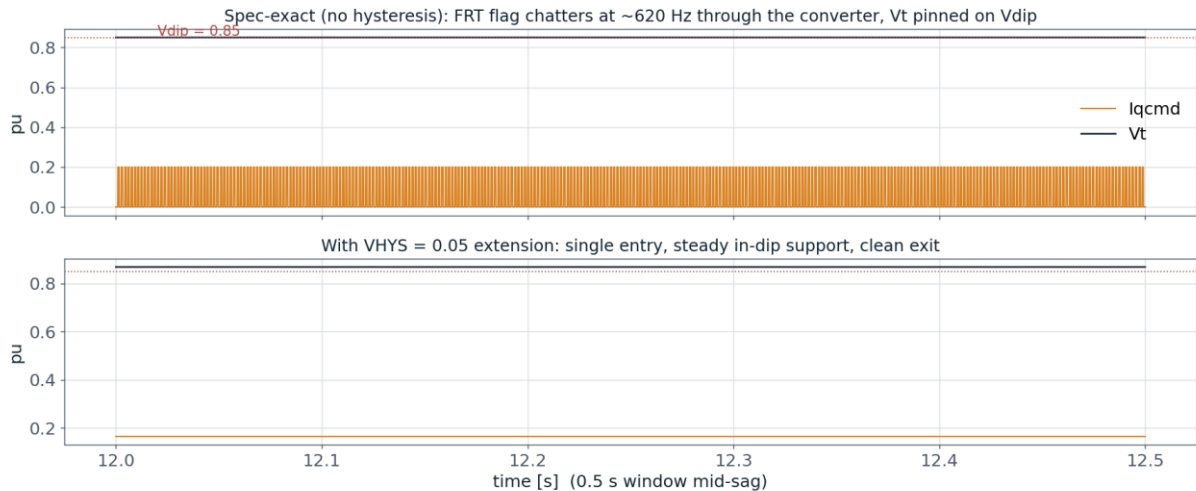


Figure 17. LVRT cycling through the converter (top) and the same case with exit hysteresis added (bottom).

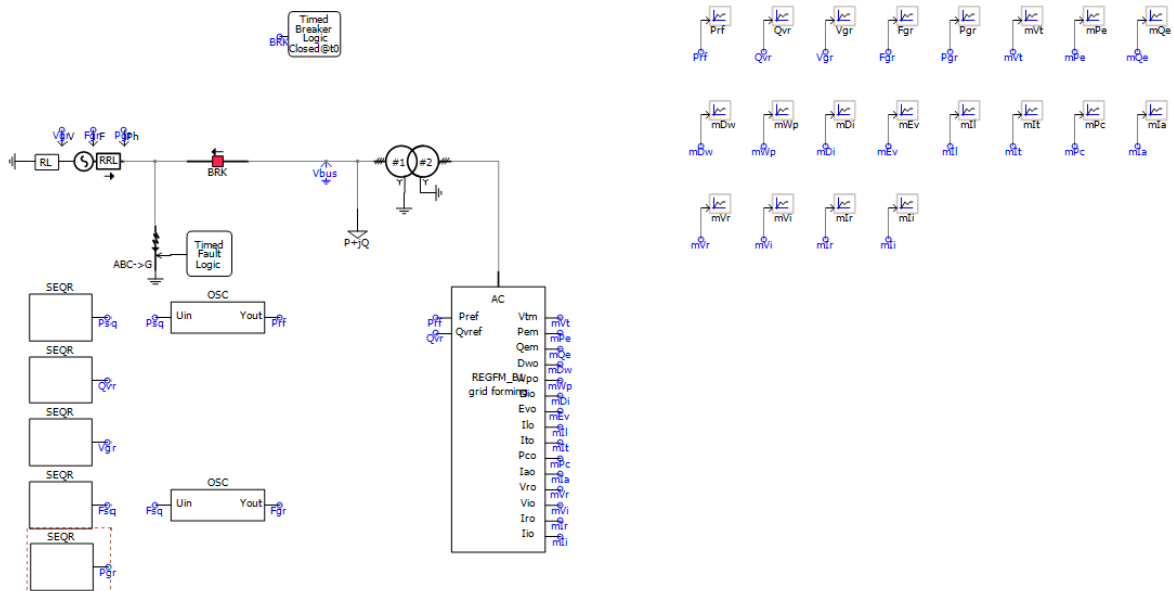
One setting worth flagging for anyone using these models: $V_{\text{Flag}} = 0$ in REEC_C is not fixed Q mode. It sends the reactive command into the fast voltage PI, which then regulates the terminal voltage to V_{min} and the Q output swings about. Fixed Q mode is $Q_{\text{Flag}} = 0$.

6. The Overall Plant Model

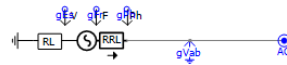
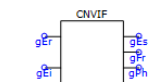
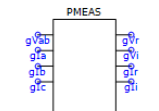
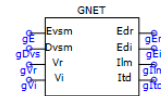
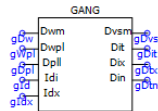
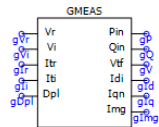
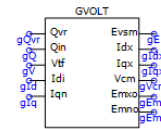
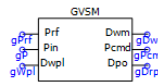
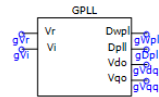
6.1 The network and the plant

For the full plant model I built a simple 33 kV system by hand in PSCAD and asked Claude to set the parameters. It has a 33 kV source with a 20 kA fault level and an X/R of 10, 5 km of single core 500 mm² copper cable (R 0.047, X 0.11 Ω/km and C 0.26 $\mu\text{F}/\text{km}$), a breaker, a 33/0.69 kV transformer with a tap changer and a small auxiliary load. The point of interconnection (POI) is the 33 kV bus at the plant end of the cable (HV_Bus2).

The plant uses an aggregated model, as recommended by WECC and NESO, so one 60 MVA inverter and one 60 MVA transformer represent the whole site. The Registered Capacity is 50 MW at 0.95 power factor lagging and leading (± 16.43 MVar), measured at the POI, which is where the plant controller takes its measurements. The integration was done in two steps: first the REEC and REGC on their own at 5 MVA, and then the 60 MVA aggregate with REPC_C controlling at the POI.



[PREF] 
 [QVREF] 



[VTM] 
 [PEM] 
 [QEM] 
 [DWO] 
 [WPO] 
 [DIO] 
 [EVO] 
 [ILO] 
 [ITO] 
 [PCO] 
 [IAO] 
 [VRO] 
 [VIO] 
 [IRO] 
 [IIO] 

Figure 20. The PSCAD nested sub-models

6.2 The converter interface

REGC_C is already a voltage source behind X_e , so there was no need for a separate converter model. The interface to the network is a standard PSCAD three phase source in external control mode, which follows the model's internal voltage every time step. A small component (CNVIF) converts the internal voltage into the source's magnitude, phase and frequency inputs, and a measurement component (PMEAS) converts the terminal voltages and currents back into the per unit values the controls expect. I wasn't keen on this at first, as having a second source sitting next to my grid source looked strange, but it is a fair representation of what REGC_C actually is, and once it was moved inside the inverter module (Section 6.5) it looks much more sensible.

6.3 Connecting to a real EMT network

Connecting the controls to a real EMT network showed up three problems that couldn't be seen on the simple test benches the models were developed on:

- Sine and cosine. The PSCAD source generates its waveform as a sine. If the measurements are converted to phasors using cosine instead, everything the controller does ends up 90° out. The model doesn't trip or go unstable. It settles at a completely wrong operating point with large circulating P and Q, while each controller still looks reasonable on its own. The fix was to use the sine reference and check it against the source zero crossings.
- Real network behaviour. The EMT network has a lightly damped 50 Hz mode from its inductance and resistance, which the simple test benches never had, and the source's external control input had a default 50 ms lag. The current PI controller had to be slowed down (from 0.5/20 to 0.2/5), the source lag was reduced to 1 ms, some damping was added through the source impedance angle, and the REEC controllers were retuned for the real grid stiffness (voltage PI from 200 to 60, Q PI from 0.25 to 0.1).

STEVE SOMMERVILLE

- Off-nominal frequency. A 5 ms filter on the phasor measurement rotates the measured values by a few degrees when the frequency is away from 50 Hz. The converter uses that measured voltage to set its output, and at 52 Hz this cut the active power to about 70% of what was asked for. It only happened in one direction of frequency error, so the under frequency tests didn't show it. Reducing the filter to 1 ms fixed it.

With these fixed, the 5 MVA model dispatched P to ± 1.0 pu and Q to ± 0.44 pu exactly through the transformer, cable and grid. A 50 ms disconnection at the POI followed by an out of phase reclose was back in sync within 0.3 s.

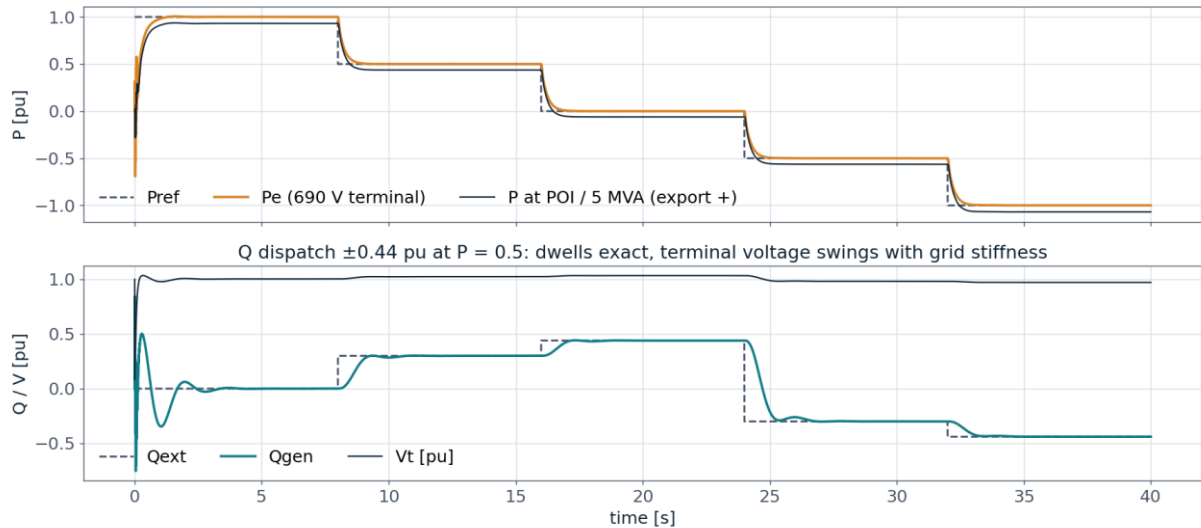


Figure 21. First runs on the real network: P and Q dispatch through the transformer, cable and 33 kV grid.

6.4 Adding the plant controller at the POI

Adding REPC_C on top needed a second measurement at the POI (the 33 kV bus voltage and the breaker currents). It also showed up an important point: the plant capability is defined at the POI, but the controller limits act at the inverter terminals.

- To deliver 50 MW at the POI, Pmax has to allow for the transformer and cable losses (0.8563 pu on 60 MVA). The same Pmax is what correctly limits the LFSM-U response to the Registered Capacity.
- The reactive power limits are not equal once they are referred through the transformer, because the transformer reactive losses reduce the export and add to the import. The limits end up as +0.336 and -0.212 pu.
- Only one reactive power integrator should be used. With both the REPC and REEC Q controllers active, the two integrators fight each other and converge very slowly. The REEC passes the plant controller's command straight through (QFlag = 0) and the REPC does the regulating.
- Different reactive power gains are needed for each mode. In voltage control the error is in volts rather than MVar, so it needs its own gains (5 and 15) compared with the Q and power factor modes (0.5 and 1.0). The active power gains were fixed and never changed during the whole test programme, which was a rule I set, as that is how a real plant should behave.
- Voltage control at the POI is the normal operating mode, as that is how National Grid runs plant. Q and power factor control are there for testing and commissioning.

There was also a sign convention to watch. REPC_C calculates its frequency error as the reference minus the measured value, so over frequency ends up in the lower deadband parameter. The GB deadbands (50.4 Hz and 49.5 Hz for LFSM) have to be entered the opposite way round to what you would expect, and nothing warns you if they are wrong.

6.5 The inverter module

To make the model easier to use and present, the inverter was then packaged as a PSCAD page module. REEC_C, REGC_C, the interface components, the controlled source and its ground are all inside, and the outside just has one three phase AC connection, three control inputs (Pref, Qext and the power factor angle) and some outputs for monitoring. The components were copied in unchanged and the first run matched the earlier results exactly. Two PSCAD quirks came up while doing this. The pins on a module symbol need lines drawn in the definition, otherwise they connect but don't show on the drawing. And the neutral connection on the three phase source moves when its zero sequence data is switched on, which had disconnected the ground on my grid source without me noticing. It turned out to be the cause of two floating terminal warnings I had been ignoring.

6.6 The GB Grid Code tests

With the plant controller controlling P and voltage at the POI, the full plant was put through seventeen scripted test cases in export mode (I decided to leave out the import mode repeats). The expected results were worked out from the Grid Code formulas before running anything, so each test is a check of the simulation against the calculation, not just a look at the plots.

Table 5. GB Grid Code tests on the full plant.

Test	Result at the POI
Calibration: MW, MVar, power factor and voltage	± 50.00 MW, ± 16.43 MVar, $Q = P \cdot \tan\phi$, V on set-point, all to 4 decimal places
LFSM-O at 100% RC, 50 to 52 Hz	50 to 34.00 MW, exactly $RC \cdot (f - 50.4)/(50 \cdot 0.1)$
LFSM-U at 80% RC, 50 to 48 Hz	40 to 49.98 MW, rises and stops at RC
FSM-O and FSM-U, 2 Hz ramps	limited at exactly $\pm 10\%$ of RC beyond ± 0.5 Hz
Voltage steps $\pm 2\%$ from 0 MVar and from 95% of capability	reaches the capability limits in about 2 s, P unaffected
Full lagging to full leading	whole range in about 3 s
FRT at 10% retained for 140 ms, from RC and full leading	full injection in the dip, back to dispatch within 2 s
LVRT at 30%, 50%, 80% and 85%	all ride through, P held at RC through both sustained dips

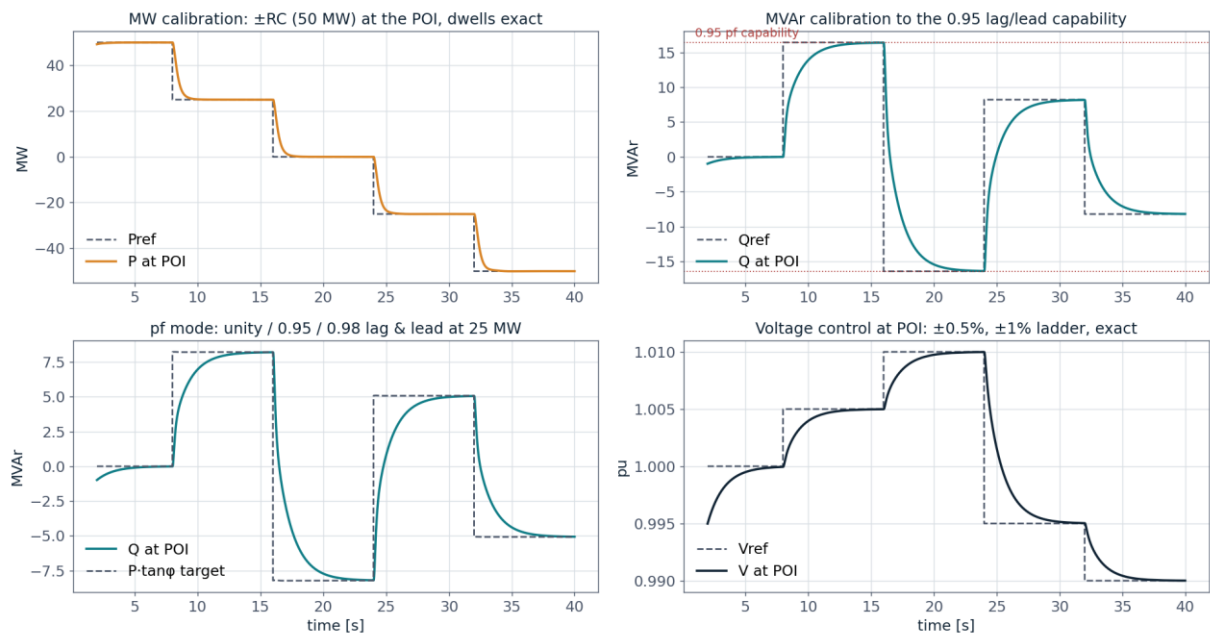


Figure 22. Plant calibration at the POI:

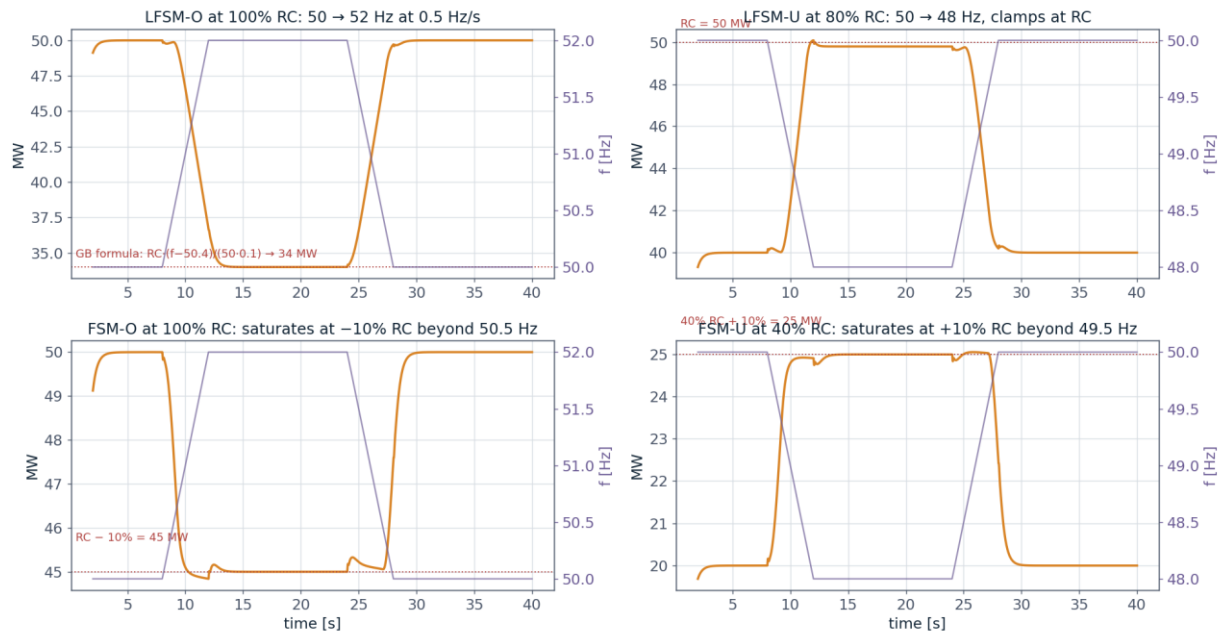


Figure 21. LFSM and FSM against the Grid Code formula values (red lines). The results sit on them.

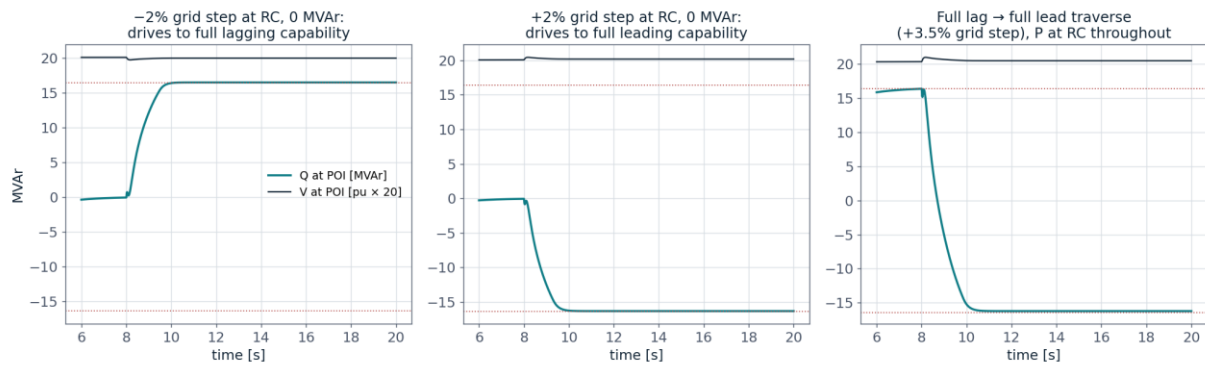


Figure 24. Voltage control at the POI:

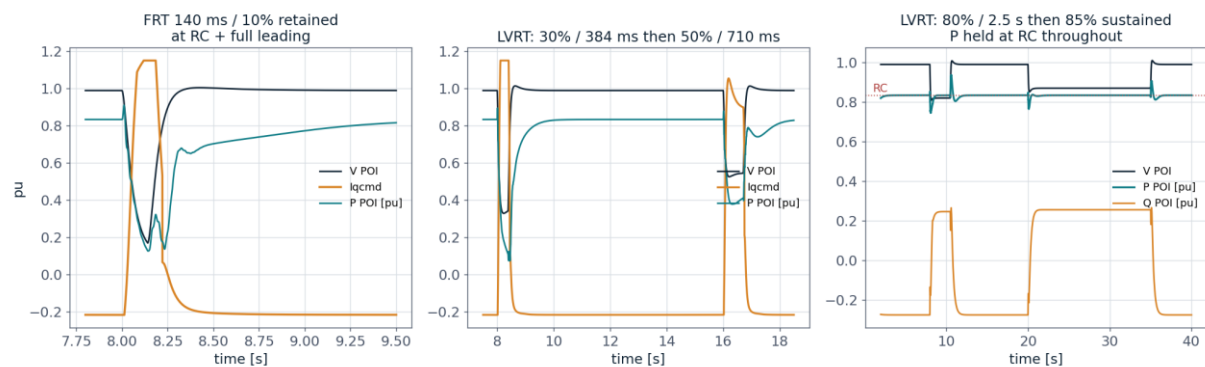


Figure 25. Ride through from the hardest starting point, full output and full leading:

The ride through tests were run in voltage control mode, and there was no LVRT cycling in any of them, which matches what the REEC_C tests showed in Section 4. None of the control models needed any changes to the code between being tested on their own and being used in the full plant.

7. What the Models Are Good For

The aim was a generic, open, WECC compatible model that can be used for Grid Code studies, so it is worth being clear about what it can and can't do.

The plant controller is a good representation of the real thing. Real plant controllers are slow, work on measured phasors and only see the positive sequence, which is what REPC_C does. The electrical control is a good representation for balanced events. The parts that differ between manufacturers are the parts where the spec is least clear (the FRT thresholds, holds and hysteresis), which is also where this work found its problems. The converter model is where the simplification really shows: positive sequence only, no switching detail and no DC side.

This means the model set is suitable for the balanced studies that make up most of a GB compliance programme (calibration, frequency response, voltage control, and balanced FRT and LVRT), run in EMT on a realistic network. That is where the interactions between the plant and the network show up, which an RMS model won't show. It is not a replacement for an OEM model for unbalanced faults, negative sequence injection, harmonics or the first cycle of a fault current. Because every input to the models is positive sequence, an unbalanced event will affect the accuracy of the upper levels too. As long as you are clear about these limits the models are very useful, but if you forget them they will still give you an answer, and it may well be wrong.

8. The Process and Who Did What

Claude wrote all the model code and the project files, wrote the scripts that build and check them, came up with the test plans (which I approved before they were run), ran all the simulations, found and fixed the problems, and in several cases did the analysis that turned a strange result into something useful, such as the oscillation band, the average injection and the rule for sizing the hysteresis. My part was building the network, setting the questions and the rules (voltage control as the normal mode, fixed P gains, keeping the number of files to a minimum), unloading projects when asked, and challenging the explanations. On two occasions Claude's first explanation of a finding didn't stand up when I questioned it, and the Vmin figure went through three versions before it showed clearly what was going on.

In terms of time, the three models and the full plant came together over about two weeks of sessions alongside the day job. The seventeen case Grid Code suite runs in under an hour without anyone watching it, every case is saved with its settings, and any of them can be re-run to give exactly the same result. On a real compliance job this makes thorough testing much more affordable.

Table 6. Findings across the whole exercise. None of them were in the main control maths.

Finding	Model	Type
Post-fault hold unfroze the PIs and reversed the injection	REEC_C	switching logic (gap in the spec)
Vmin of 0.90 fails the GB 85%/180 s case	REEC_C	parameter setting
LVRT cycling on weak grids; averaging effect; hysteresis sizing rule	REEC_C	switching logic (missing from the spec)
Recovery limits never switched on (sampled at time zero)	REGC_C	initialisation
VFlag = 0 is not fixed Q mode	REEC_C	settings
Sine/cosine measurement mismatch, 90° error	plant	interface
Source lag and 50 Hz network mode against the current loop	plant	interface
Measurement filter rotates phasors off-nominal, P reduced	plant	interface
POI capability needs loss allowance, unequal Q limits	plant	interface (POI vs terminal)

The main thing I would want other engineers to take from this is that every problem found, whether in the code, in the published models or in connecting them together, was in the initialisation, the switching logic, the settings or an interface. None of them were in the transfer functions. So if two packages give different results for the same WECC model, those are the places to look first: the freeze logic, the holds, the order of the limits, the initialisation and the measurement conventions, as those are the areas the spec doesn't fully define.

9. Engineering Takeaways

Engineering takeaway: the full WECC control stack can be built in PSCAD as open FORTRAN code, tested one level at a time against hand calculations, and taken through a GB Grid Code test programme on a real network. Scripted AI-driven testing found the problems that mattered, and they were all in the switching logic, settings and interfaces rather than the control maths.

- Build each level as code, check it against values you can work out by hand, and only then join it to the next one. A replica in a second language confirms the code matches the diagram.
- Dispatch tests check the maths, but event tests are what find the problems. Every finding that mattered came from a dip, a hold, a limit or an initialisation.
- Simple test benches hide things. Each step of joining the models together showed up something the previous test bench couldn't, which is why it is worth building the full plant on a real EMT network.
- Check Vmin against your sustained dip requirement, as the default of 0.90 fails the GB 85%/180 s case.
- LVRT oscillation on weak grids can be predicted and fixed, but the hysteresis has to be bigger than the voltage rise from the injection or it just slows the cycling down.
- Capability is defined at the POI but the controller limits act at the terminals, so allow for losses and expect unequal Q limits.
- These are positive sequence models. They are good for balanced Grid Code work in EMT, but not a replacement for OEM models for unbalanced faults, harmonics or first cycle fault currents.
- Running the tests by script makes thorough testing affordable, with every case saved and repeatable.

10. References

- WECC second generation generic renewable models (REPC, REEC and REGC): WECC REMWG specifications, and EPRI and PowerWorld model documentation.
- GB Grid Code (ECC and ECP): frequency response (LFSM and FSM), fault ride through and voltage dip requirements.
- PSCAD 5.0.2 and the mhi.pscad Python automation library (Manitoba Hydro International).
- Technical Notes TN-004 (REPC_C plant controller) and TN-005 (REEC_C and REGC_C storage system) at stevesommerville.com.
- Build and test slide decks for each part (REPC_C, REEC_C, REGC_C and the joined model, and the full plant Grid Code tests), in the project folders listed below.

Appendix A: Models and Test Records

Table 7. Everything is under OneDrive\PSCAD\. Each run is saved as raw data together with the settings that produced it.

Part	Model and project files	Test records
REPC_C	WECC REPC_C\WECC_REPC_C.pscx (component), REPC_TB4.pscx (test bench)	tests\ (10 runs and the replica check script)
REEC_C	WECC REEC_C\WECC_REEC_C.pscx	tests\ (16 runs plus the band sweeps)
REGC_C	WECC REGC_C\WECC_REGC_C.pscx	tests\ (11 runs)
Joined model	WECC Chain\WECC_REEC_REGC.pscx	tests\ (6 runs)
Full plant	WECC Chain\WECC_Overall.pscx (inverter module, REPC at the POI)	tests\ (17 Grid Code cases)
Scripts	WECC Scripts\ (generators, checks, test drivers, plots and slide decks)	