P and M class PMU examples for MATLAB ® Simulink

Andrew J. Roscoe
Version 20131028
(28th October 2013)

RoscoePMU_Public_20131028.zip

This contains examples of P and M class PMUs based on [1] and [2]. The examples now support reporting at 50 or 60Hz, via an input parameter which sets both f0 (nominal frequency) and Fs, the reporting rate. Presently the M class example is limited in that Fs=f0, so the longer, more exotic filters provided by slower reporting rates such as Fs=10Hz are not available (yet). Examples with sample rates of 4kHz, 4.8kHz, 10kHz, 12.8kHz and 15.36kHz are provided, to cater for 80 and 256 samples-per-nominal-cycle at both 50 and 60Hz [3], plus the useful 10kHz sample rate. The examples are provided as 64-bit "mexw64" files. Examples of the algorithms are shown in a test environment which reproduces the 30-second test waveform from [2].

RoscoePMU_Public_20131028

Quick start

1. Unzip to a directory.
2. Double-click "*PMU_for_PublicUse_setup.m*" (MATLAB starts)
3. Run "*PMU_for_PublicUse_setup.m*" in MATLAB
4. Open "*PMU_for_PublicUse_P.slx*" in MATLAB/Simulink
5. Run "*PMU_for_PublicUse_P.slx*"
6. Ditto with "*PMU_for_PublicUse_M.slx*"

What's in this archive?

This directory contains compiled MATLAB® Simulink "S functions" for P and M class PMUs using adaptive filters. The P and M class algorithms are exactly those used in [1] (The hybrid P/M PMU referred to in [1] is not included). For further description of the algorithms and their evolution, please refer to [2] and [4], and previous papers referred to in [1, 2, 4].

How do these PMUs work?

These PMUs use adaptive filtering and so the rejection of harmonics and unbalance is virtually "ideal", no matter what the power system frequency is. The problem of "off-nominal frequency" is virtually eliminated in these PMUs, since the filter zeros always track the harmonics. The baseband signal is also always at almost exactly zero Hz, and so the issue of FIR filter flatness and calibration/compensation is much reduced compared to non-frequency-tracking implementations. In that respect, these algorithms perform much better than the "reference" algorithms from C37.118 [5] and other non-frequency-tracking implementations. The algorithms are C37.118.1 compliant in all respects *except* [1].

1. The M class ROCOF during steady-state testing (across the frequency range) when the signal amplitude is <<1pu, which is common to most PMUs due to ADC resolution.
2. M class Frequency and ROCOF during OOB testing.

Notably, the exceptions listed above will cease to be exceptions when the standard is adjusted in the near future, because all PMUs are failing these same tests (i.e. they are practically impossible to comply with).

Nominal frequency, frequency range and reporting rate Fs

These examples (compiled as Simulink "S functions") now have an input port which allows you to set frequency f0, and this same value will also be used as the reporting rate Fs. So, in these examples, there is a restriction that Fs=f0. The PMUs will track over a range of 0.7*f0 to 1.3*f0, with hardly any degradation in performance within those ranges. The M class device has filter lengths as described in [1] so that it complies with the F_S=50 & 60Hz dynamic tests. In these examples you cannot pick the other M-class choices with longer filters such as F_S=25Hz or F_S=10Hz. But, in [2], I showed how "frequency chirp" effects can be a big problem in those longer M-class filters if frequency moves dynamically. So, in my opinion the F_S=50Hz M-class PMU is a good choice anyway.

It is possible to change f0 (=Fs) dynamically with these PMUs. The algorithm will settle into the new reporting rate after a short time. These examples will start up in f0=Fs=50Hz configuration on the first sample frame, so even if you input f0=Fs=60Hz it may be a couple of cycles before you see reports at 1/60s intervals.

While developing the functionality for dynamic adjustment of f0 and Fs, I was really hoping to release examples of the M class devices with slower reporting rates. The logic to allow the dynamic adjustment of f0 and Fs is actually complete, so that you could, for example change from Fs=50 to Fs=10 in real time. However, for the class devices with Fs of 30Hz and below, I have uncovered some issues involving simultaneous compliance with the C37.118 out-of-band (OOB) and bandwidth test which I still need to resolve. This requires some subtle adjustment to the design of the longer M-class filters and calibration paths which I do not have time to address right now.

Sample rates

There are 5 versions of each P and M class PMU. The differences are only in the sample rate which they have been pre-compiled with. The options are 10kHz (as used in [1]), and also 4/4.8kHz and 12.8/15.36kHz since these are favoured sample rates representing 80 and 256 samples-per-nominal cycles at both 50 and 60Hz [3]. You can choose the appropriate block from the file "*RoscoePMUs.slx*" and drop it into your Simulink Simulation.

Example applications

To show examples of use, there are two model files "*PMU_for_PublicUse_P.slx*" and "*PMU_for_PublicUse_M.slx"*. They are virtually identical. Both contain the same 30-second test waveform generator from [2]. I have provided that signal generation code in a source-code format so that it is transparent. You could test other PMUs with it, too. It is not very tidy code and has evolved bit-by-bit without ever being cleaned up. The PMU source code is much, much neater and better controlled! If you run the examples at 10kHz, you can also use the included block (S function) "*PMU_HistoricalRecall_10kHz*" which allows a comparison of the PMU results to the actual generated waveform, so that TVE and other errors can be assessed. To run these examples, first execute the script "*PMU_for_PublicUse_setup.m*" and then run the model.

Changing sample rate

If you want to use a different sample rate, then adjust the setting of `SampleFreq` in "*PMU_for_PublicUse_setup.m*" and make sure it matches the block you insert from "*RoscoePMUs.slx*". You will need to remove the error comparison/calculation sections since the *PMU_HistoricalRecall_10kHz* block is only provided as a 10kHz example, and will throw an error if you try to run it at any other rate.

Insertion into more complex environments

If you want to move the PMU blocks to your own more complex models using high-fidelity power system simulations with small step times, then take extra care with rate transitions and analogue anti-aliasing filter implementations since these can insert simulation delays which you don't expect. Code sections in "*PMU_for_PublicUse_setup.m*" describe how to configure the input parameters *"HardwareAndAntiAlias_GainCorrectCoeffs"* and *"HardwareAndAntiAlias_PhaseCorrectCoeffs"*. If you set these up correctly, the PMU will compensate the amplitude and phase results for your (simulated) analogue hardware and (simulated) ADC sampling.

Creating report outputs at the full sample rate

There is a configurable input flag called *"Sample only when Timestamp crosses Fs intervals"*. Normally you should set this to 1 and the PMU will issue a new report only on the regular 20ms (50Hz) or 16.6667ms (60Hz) intervals (1/Fs) following a UTC second rollover. Since the block is executing at 4, 4.8, 10, 12.8 or 15.36kHz , the data from the block still comes out at this rate, but it only changes on the 1/Fs intervals when you set the flag to 1. There is also a *"NewReportAvailable"* output flag which goes true when a new report appears. This is the C37 compliant mode. You can downsample the outputs if you like to 1/Fs using rate transition blocks if you are careful.

You can alternatively set the input flag to 0. If you do this, then ignore *"NewReportAvailable"*. The PMU will issue a new report every sample, and the timestamps will be determined by the present UTC_time and the exact configurations of the internal filters. The timestamps will not necessarily increase at monotonic intervals, or match up exactly with the UTC_time second rollover. This mode is not C37 compliant, but it is very useful for assessing the dynamic response characteristics since the "equivalent time sampling approach" from [5] does not need to be used.

There is no execution time penalty for using this latter mode, since the PMU generates a new report internally at the full sample rate. You should find that the algorithms execute very quickly, even the M class, due to the architecture and also because they are precompiled into the linked libraries.

Andrew J. Roscoe

Andrew.J.Roscoe@strath.ac.uk

http://personal.strath.ac.uk/andrew.j.roscoe/

28th October 2013

Previous version history

RoscoePMU_Public_20130814.zip

This contains examples of P and M class PMUs based on [1] and [2]. The reporting rate is fixed at 50Hz, although the PMUs would measure at 50Hz or 60Hz (over the 40-70Hz range). Examples with sample rates of 4kHz, 10kHz and 12.8kHz are provided, in both 32-bit and 64-bit S function MEX files.

References

[1] A. J. Roscoe, "Exploring the relative performance of frequency-tracking and fixed-filter Phasor Measurement Unit algorithms under C37.118 test procedures, the effects of interharmonics, and initial attempts at merging P class response with M class filtering," *IEEE Transactions on Instrumentation and Measurement,* vol. 62, pp. 2140-2153, 2013.

[2] A. J. Roscoe, I. F. Abdulhadi, and G. M. Burt, "P and M Class Phasor Measurement Unit Algorithms using Adaptive Cascaded Filters," *IEEE Transactions on Power Delivery,* vol. 28, pp. 1447-1459, 2013.

[3] UCA International Users Group, "Implementation Guideline for Digital Interface to Instrument Transformers using IEC 61850-9-2," 2004. Available: http://iec61850.ucaiug.org/implementation%20guidelines/digif_spec_9-2le_r2-1_040707-cb.pdf, accessed Oct 2013.

[4] A. J. Roscoe, I. F. Abdulhadi, and G. M. Burt, "P-Class Phasor Measurement Unit Algorithms Using Adaptive Filtering to Enhance Accuracy at Off-Nominal Frequencies," in *IEEE SMFG 2011 Smart Measurements For Future Grids*, Bologna, Italy, 2011.

[5] IEEE, "IEEE Standard for Synchrophasor Measurements for Power Systems," C37.118.1-2011, 2011.