P and M class PMU examples for MATLAB ® Simulink

Andrew J. Roscoe

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.mdl*" in MATLAB/Simulink
5. Run "*PMU_for_PublicUse_P.mdl*"
6. Ditto with "*PMU_for_PublicUse_M.mdl*"

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 [3], and previous papers referred to in [1-3]. Both 32 and 64-bit versions are given. Your MATLAB installation should pick the right one automatically.

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 [4] 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, all 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 and frequency range

These examples (compiled as Simulink "S functions") are all pre-configured for a nominal frequency of 50Hz, but they will track over a range of 40-70Hz and so can be used at 60Hz. The reporting rate for these S functions is fixed at 50Hz. The M class device has filter lengths as described in [1] so that it complies with the F_S=50Hz dynamic tests. F_S is fixed at 50Hz and so 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 the best choice anyway.

Sample rates

There are 3 versions of the P and M class PMUs. 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 4kHz and 12.8kHz since these are favoured sample rates under IEC61850. You can choose the appropriate block from the file "*RoscoePMUs.mdl*" and drop it into your Simulink Simulation.

Example applications

To show very simple examples of use, there are two model files "*PMU_for_PublicUse_P.mdl*" and "*PMU_for_PublicUse_M.mdl*". They are virtually identical and contain a very basic signal generator and PMU demonstration. 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 `SamplesPerCycleAtFnom` in "*PMU_for_PublicUse_setup.m*" and make sure it matches the block you insert from "*RoscoePMUs.mdl*".

Insertion into more complex environments

If you want to move the 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.

If you are assessing what the *errors* are (i.e. TVE, frequency error, ROCOF error) compared to a known applied signal, then you need to carefully account for the timestamp which the PMU issues, which will be "in the past" compared to the present *UTC_time*. In my C37 test environment, I have a piece of code which remembers the applied signal parameters in a rolling buffer, so that they can be "pulled back" from the timestamp and compared accurately. This type of code is not in the given example, for reasons of simplicity etc..

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) intervals following a UTC second rollover. Since the block is executing at 4, 10 or 12.8kHz, the data from the block still comes out at this rate, but it only changes on the 20ms 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 20ms using rate transition blocks if you are careful.

You can set the input flag to 0. If you do this, then ignore *"NewReportAvailable"*. The PMU will issue a new report every sample (at 4, 10 or 12.8kHz), 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 [4] 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.

Different configurations

It would be possible to compile slightly different versions, for instance with nominal frequency 60Hz and reporting rate 60Hz, or for sample rates other than 4, 10 or 12.8kHz. If this would be particularly helpful for your research, then please email me. Also, while the examples only provide the most basic PMU output information, the per-phase data is not given, only the positive-sequence summary. I could provide a model in the public domain with this functionality if there is a good enough reason. Please email to discuss.

Finally, these algorithms do not contain any special fault-ride-through optimisations yet. So, while [5] showed how you can make the performance better, I don’t consider that work finished and no such algorithmic “tweaks” are present in these algorithms yet. The determination of frequency and ROCOF is strictly as per [1].

Andrew J. Roscoe

Andrew.J.Roscoe@strath.ac.uk

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

14th August 2013

[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] 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.

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

[5] A. J. Roscoe, G. M. Burt, and G. Rietveld, "Improving frequency and ROCOF accuracy during faults, for P class Phasor Measurement Units," in *IEEE Applied Measurements in Power Systems (AMPS)*, Aachen, Germany, 2013.