Fast Operational Filter (FOR)

Appedge, a French high-technologies company
specialized in Real-Time/Control.
Website: www.appedge.com – E-mail: info@appedge.com - Phone: 0033 1 47 82 95 05.

I. Introduction.

The Fast Operational Filter (**FOR**™ for ***Filtre Opérationnel Rapide***) designed by Appedge, is based on algebraic differentiator developed by l'INRIA/CNRS[1]. Operational differentiators are very easy to set and own a formidable efficiency compared with classic filtering.

II. Principle.

The aim is to rebuild a signal at each sample by passing to it a polynomial which extract $Y(t)$ and $\frac{dy}{dt}$, estimated in real-time.

III. Mathematical and Real-time sides.

Filters based on operational mathematics are computed in continuous time by some integrals rational fractions. The literature puts forward a theory for the design and use of these filters in continuous, discrete or differential time. By following it step-by-step, the numerical implementation of these filters requires a sizeable number operations and a high sampling time, then a significant time consuming. Thus, this implementation in Real-time constraint is difficult or impossible to use.

This handicap was pointed out by Appedge by developing an innovative integration methodology of these filters while meeting the CPU hardware constraints and minimizing with efficiency the number operations. In some cases, the number of operations is similar to the number of operations of classic filtering. This new methodology opens up important perspectives in the **on-line parameters estimate**. But the real benefit is the estimate of the derivatives in the development of PID[2] or Real-time Control. Thus, all the drawbacks use are vanished. Now, with the use of few Simulink™ blocks, it is possible to get an efficient filter.

IV. FOR Deployment.

The **FOR** can be implemented directly on Real-time boards in 16 bits or less depending on the range of the signal to process.

Appedge puts forward implementations:

- With Simulink™, which will be able to be generated on different Real-time targets,
- With FPGA, which will enable to replace classic filters in Integrated Circuits,
- Or on any simulator like OpenModelica, Amesim™, Dymola™ or any proprietary applications.

V. A certain and basic implementation.

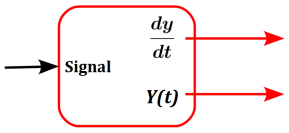
The **FOR** implementation is done without preliminary computation and is basic. Concretely, the filter owns setting parameters:

- Filter order,
- Filter sampling period.

Regarding the outputs, the filter generates the time derivative of the signal and the $Y(t)$ filtered signal.

The achievement of the **time derivative** is usually unknown without analytical solutions. In the same manner, the "Trial and Error" techniques of the numerical filters or the complex settings regarding the compromise between the signal dynamic and its noise are not the right way to compute this derivative.
The **FOR**, provided that the signal reconstitution is acceptable compared with its requirements, then, displays the time derivative.

Fig. 1: FOR Components.



FOR
APPEDGE V1.0

VI. CPU Load and Setting.

CPU Load depends on filtering expected but also the sample size available, for example on a transition time. **FOR** can apply to signals with low samplings (4 or 5 per transition time). In this instance, the **FOR** order must be chosen with a low value. In case of oversampling - more than one hundred dots on the transition time, the filter order could be increased and the sampling adjusted. Thus, CPU Load can range from three to thirty computing operations by using of ten to two memory cells per sample period.

[1] Frequency Analysis of algebraic differentiators: Francisco de Asís GARCÍA COLLADO, Brigitte D'ANDRÉA-NOVEL, Michel FLIESS, Hugues MOUNIER and GRETSI.
[2] Product sheet n°1: Generic PID Real-time (Online control without model).

VII. User benefits: FOR versus Classic Filters.

- A simple setting with efficient results immediately.
- The **FOR** filter insures the true time derivative because the signal is rebuilt from the valued derivative.
- The filter gets no delay in its optimum use. The filtered signal is smoother than classic filter while keeping the same dynamic of original signals. Thus, there is no need to adapt the filter to the computation to achieve.

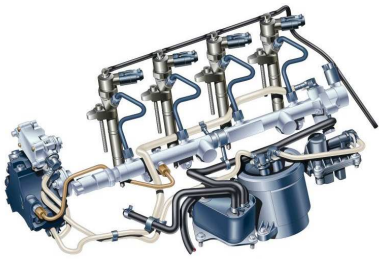


Fig. 2: Common Rail

VIII. Filtering Application and Common Rail.

The studied case is the filtering and the estimate of the Rail pressure derivative. In order to get a workable derivative with classic methods (Filter + Differentiator), it is necessary to select a dynamic filtering which crushes the signal.

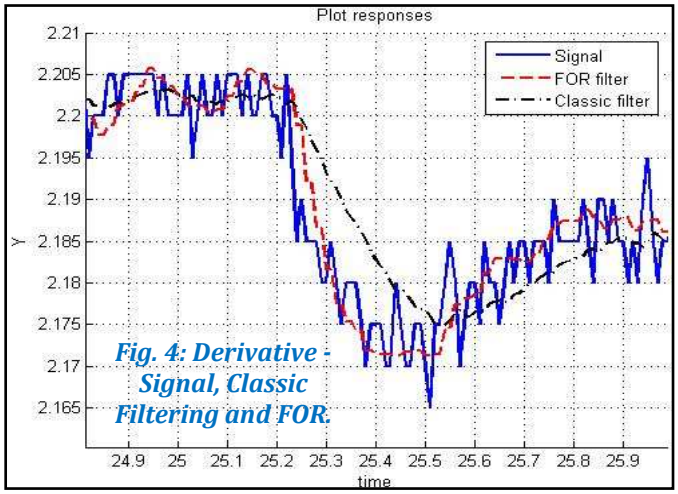


Fig. 4: Derivative - Signal, Classic Filtering and FOR.

This filtering is not workable for others computations (too much delays). In a control system, a Rail for example, there are usually much as filters as computation steps since each numerical filter needs to be adapted to the computation run downstream. Thus, it will significantly increase the CPU load. **FOR** enables to fix this point.

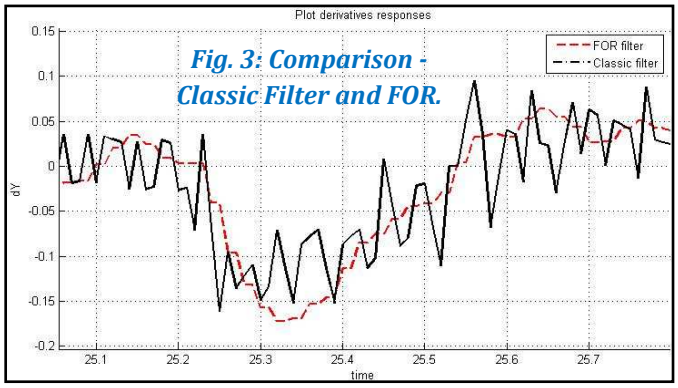


Fig. 3: Comparison - Classic Filter and FOR.

IX. Cylinder Balancing.

The aim is to characterize in less than thirty samples, a piston among the 2, 3, 4,... in order to adapt injection into the goal to balance the stresses on the crankshaft and to improve the combustion. The quality and the steadiness of acquired signals thank to **FOR**, enable effortlessly, to detect and diagnose the differences between cylinders without a complex algorithms implementation. Indeed, **FOR** just compares the derivative signal amplitude of each cylinder.

To compare, the numerical derivative (in yellow on Fig. 6) does not enable to reach the accuracy and the reproducibility expected: the signal is too chaotic on over the working range of the engine.

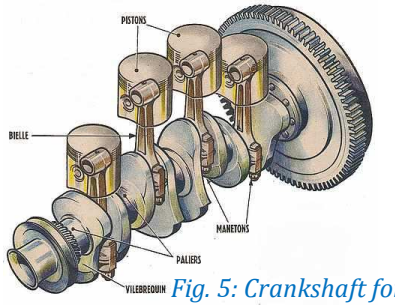


Fig. 5: Crankshaft for a 4-Cylinders engine

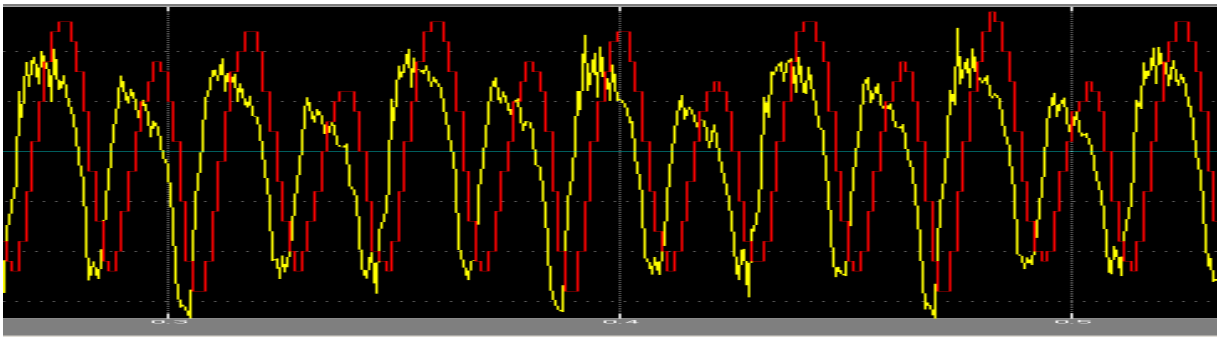


Fig. 6: Comparison for Cylinder Balancing Application - Numerical (yellow) and FOR (red).