Redesign of the JavaFX Charts Library
in View of Real-Time Visualisation of Scientific Data

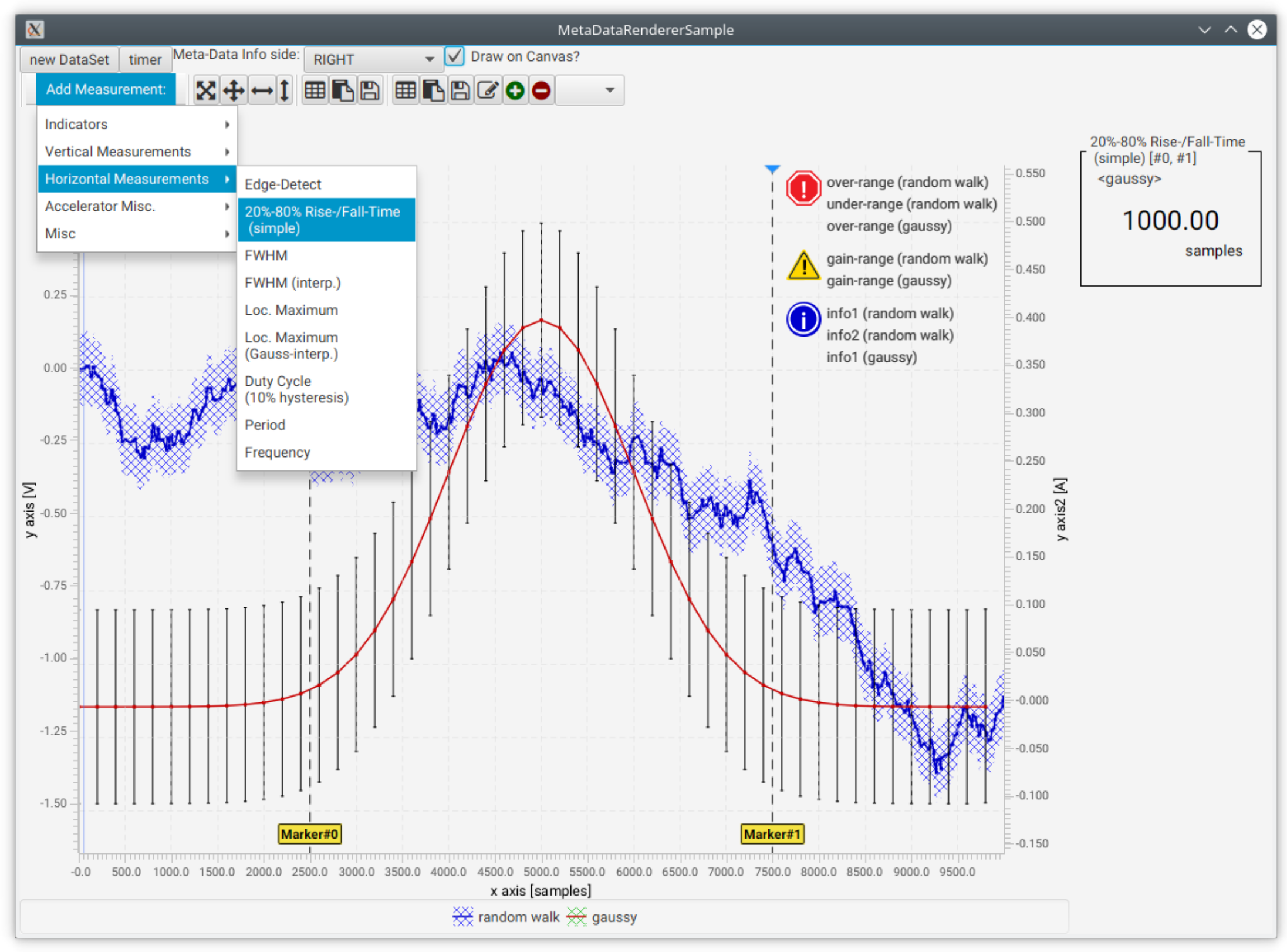
Ralph J. Steinhagen, Harald Bräuning, Alexander Krimm, Timo Milosic (GSI Helmholtzzentrum, Darmstadt)

The accurate graphical representation of accelerator- or beam-based parameters is crucial for commissioning and operation in any modern accelerator. Charts are one of the most visible but at the same time often underappreciated accelerator control system components even though these are crucial for easing and improving a quick intuitive understanding of complex or large quantities of data, which in turn is used to efficiently control, troubleshoot or improve the accelerator performance
Based on earlier designs the new ChartFx scientific charting library has been developed that preserves the feature-rich and extensible functionality of established earlier Swing-based libraries (JDataViewer) while addressing the performance bottlenecks and API issues of the JavaFX implementation. The library has been optimized for real-time data visualization at 25 Hz for data sets with a few 10 thousand up to 5 million data points common in digital signal processing applications.
Relying on modular open interfaces, the library allows the exchange of the underlying technology if necessary, while easing its use by casual developers and allowing more-inclined developers to modify, add or extend missing functionalities. A performance and functionality comparison with other existing charting libraries is provided.

1 Why Java?

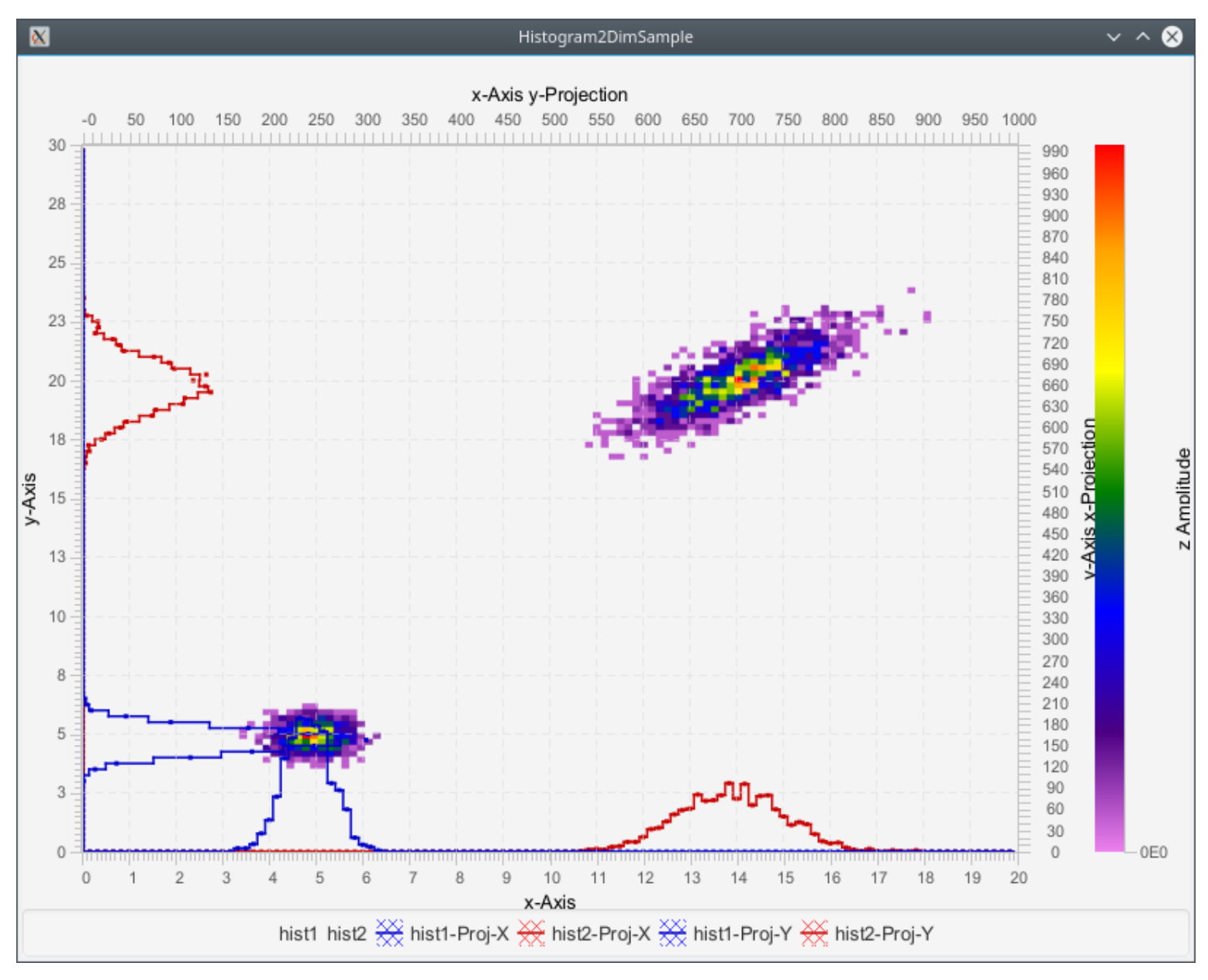
- Java adopted for FAIR's accelerator settings supply and other control sub-systems
- possibility of reusing server code on the client side made use of Swing UI framework favourable … albeit no native Swing-based charting
  → development of JDataViewer in 2002
- JavaFX replaces aging Swing by 2026 (stagnating API, bug-fixes only, ...)
- JavaFX provides Charts but with limited functionality and lacking performance

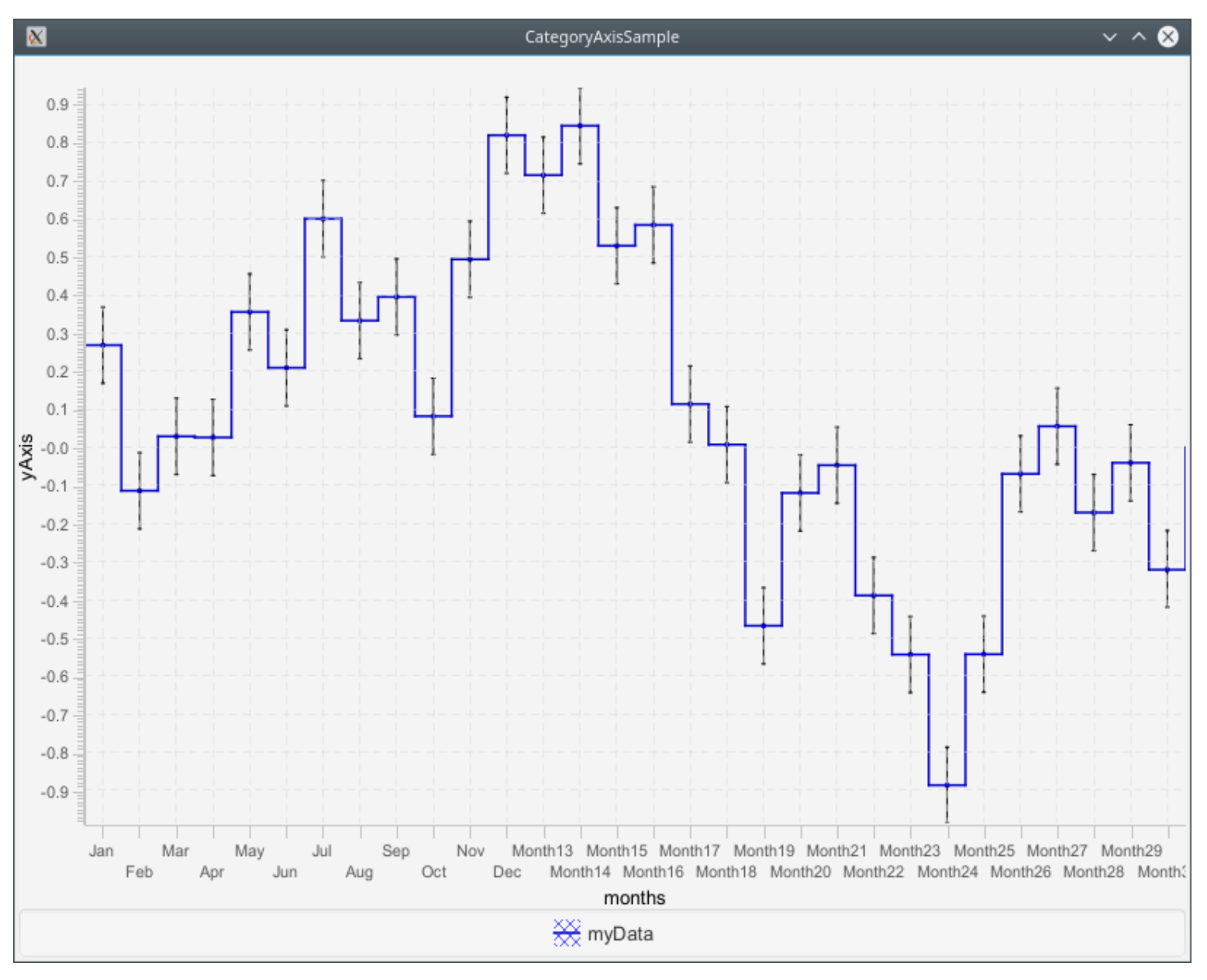
2 Why Re-Design of a new JavaFX Charting Library:

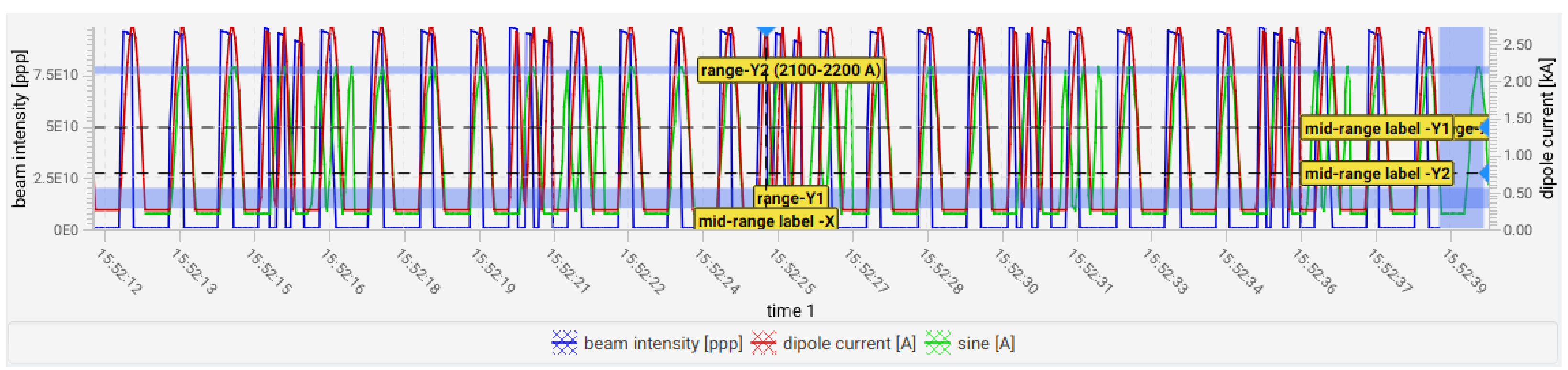


Worthwhile to re-engineer because of missing JavaFX's Chart features:

- performance (use of slow Scenegraph rather than Scene)
- extendability & long-term maintenance (JavaFX's frequent use of 'final' keyword)
- error-bars/-surfaces
- multiple-axes
- data-zooming
- plugin-interactors/editing
- open data container
- Lin/Log/User-def axes
- missing interaction with common math routines (signal processing etc.)





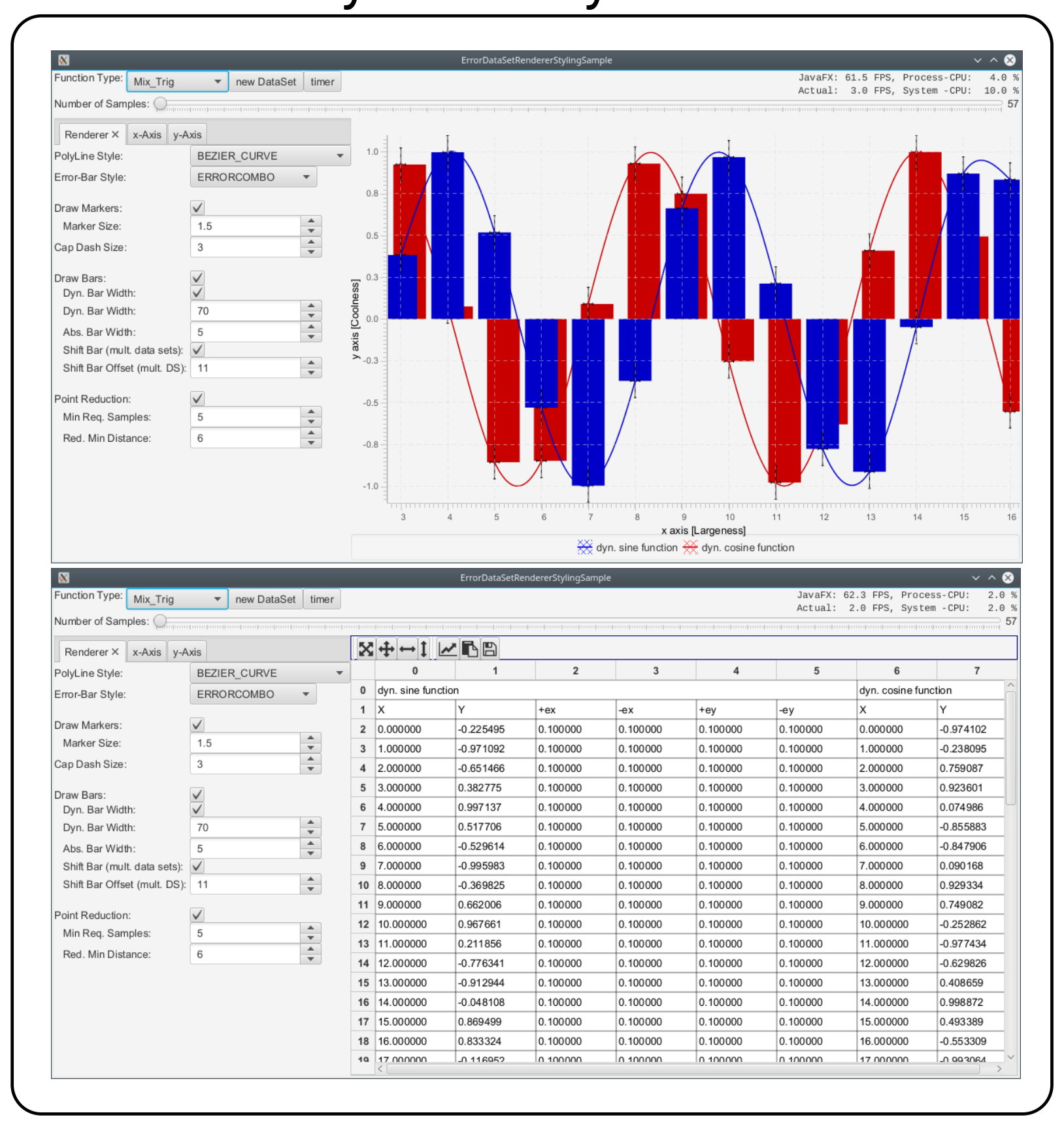


3 Interactors & Extendability

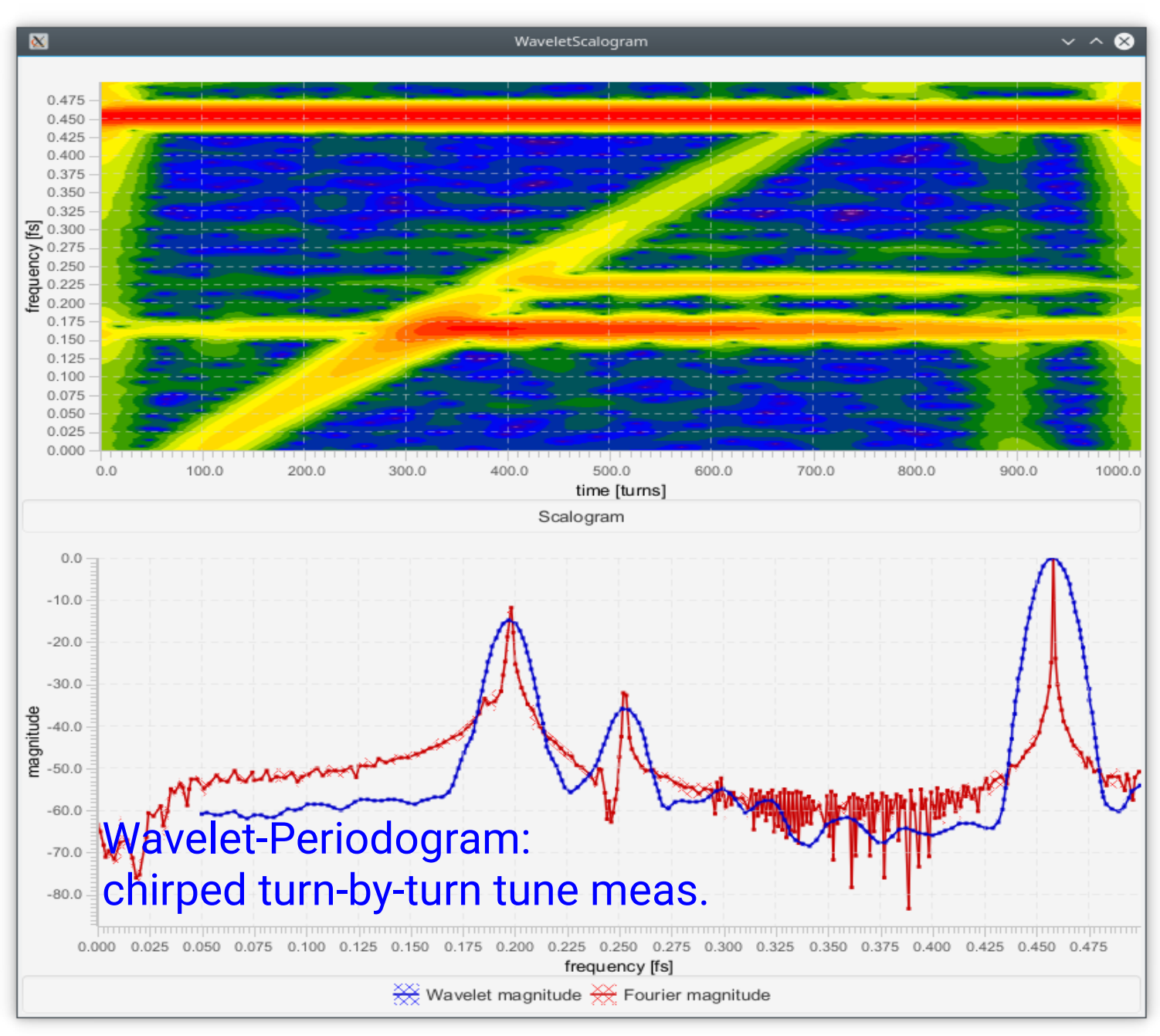
Strong focus on: common standards, maintainability, testability, reliability, use of interfaces and abstract classes

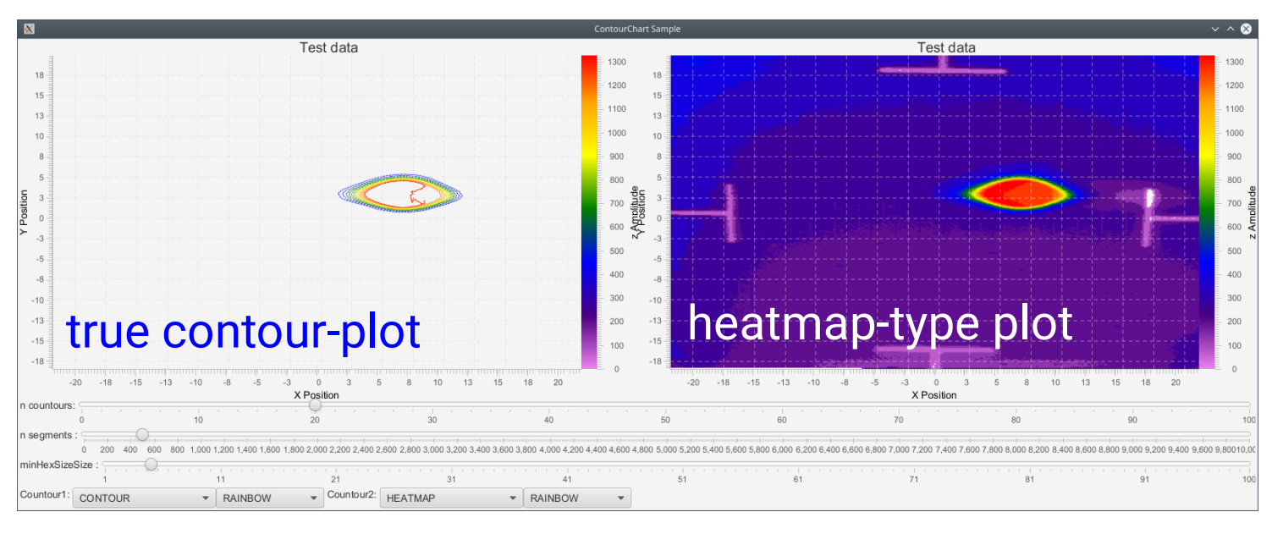
- DataSets, Error- and Meta-Data handling
- Signal-processing math library: FFTs, Wavelets, linear algebra, integration and differentiation, IIR- & FIR-type filters, non-linear $\chi 2$ -type fitting, ...
- Chart: euclidean, polar, 2D projections, ...
- Renderer: see examples
- ChartPlugin: data zoomer, panner, data value and range indicators, data point tool-tips, DataSet editing, table view, export to CSV and system clipboard
- parameter measurement such as rise-time, min, max, rms, etc. common in lab grade oscilloscope, spectrum analyzers, vector-network-analyzers, etc.

Look&Feel style-able by Code or CSS



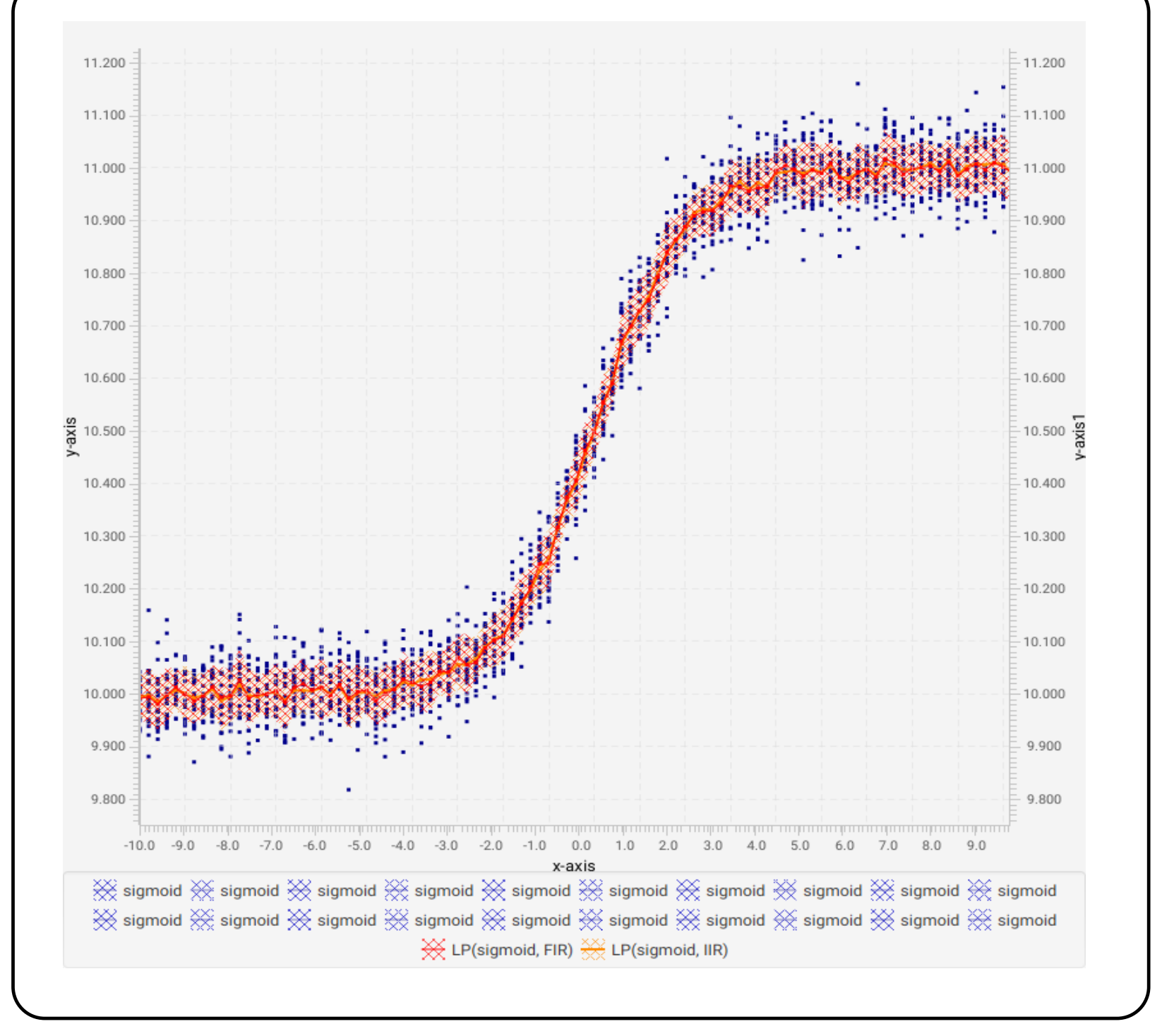
4 2D Projections of 3D Data



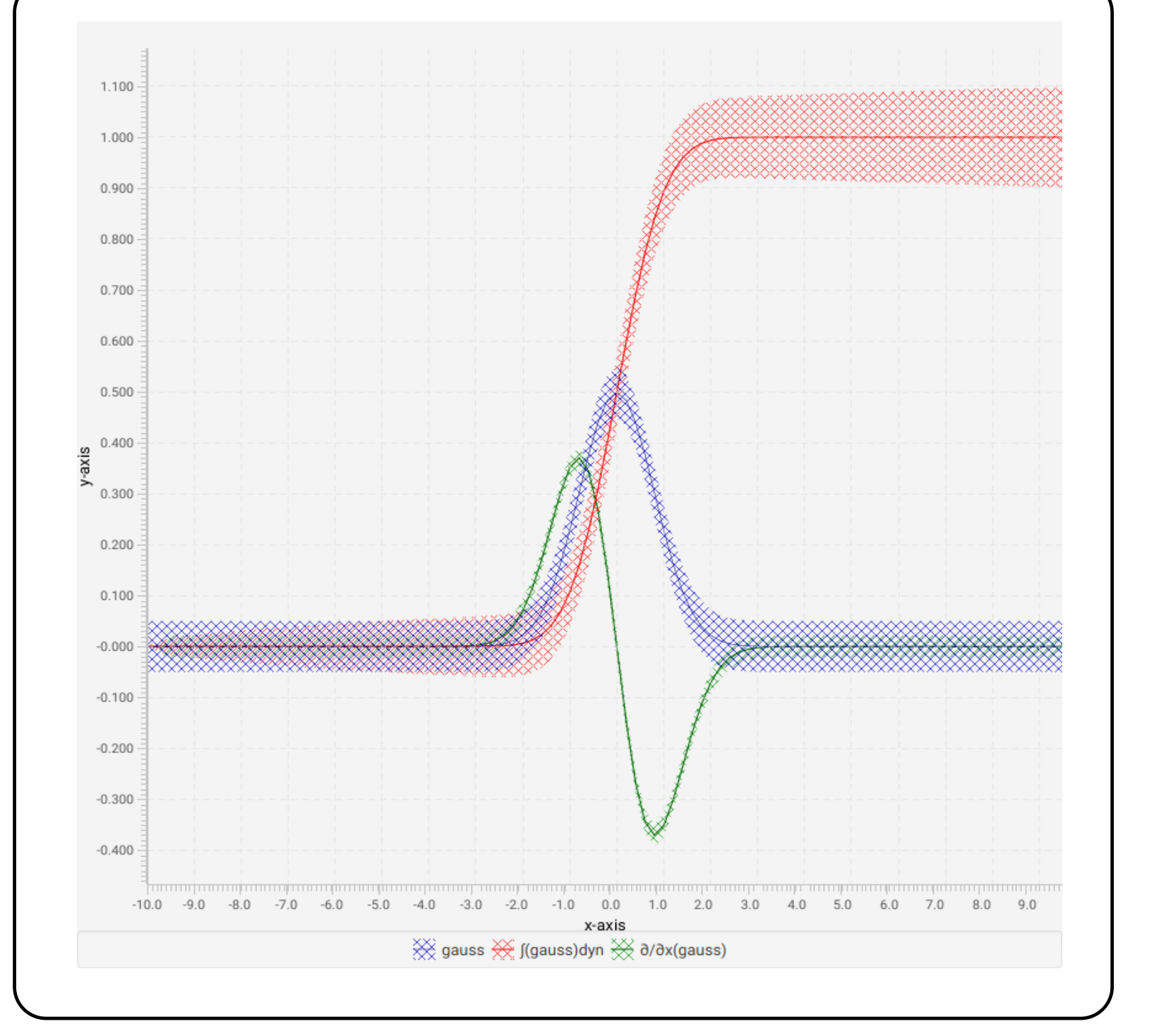


5 Math-Library operating directly on DataSet Interface

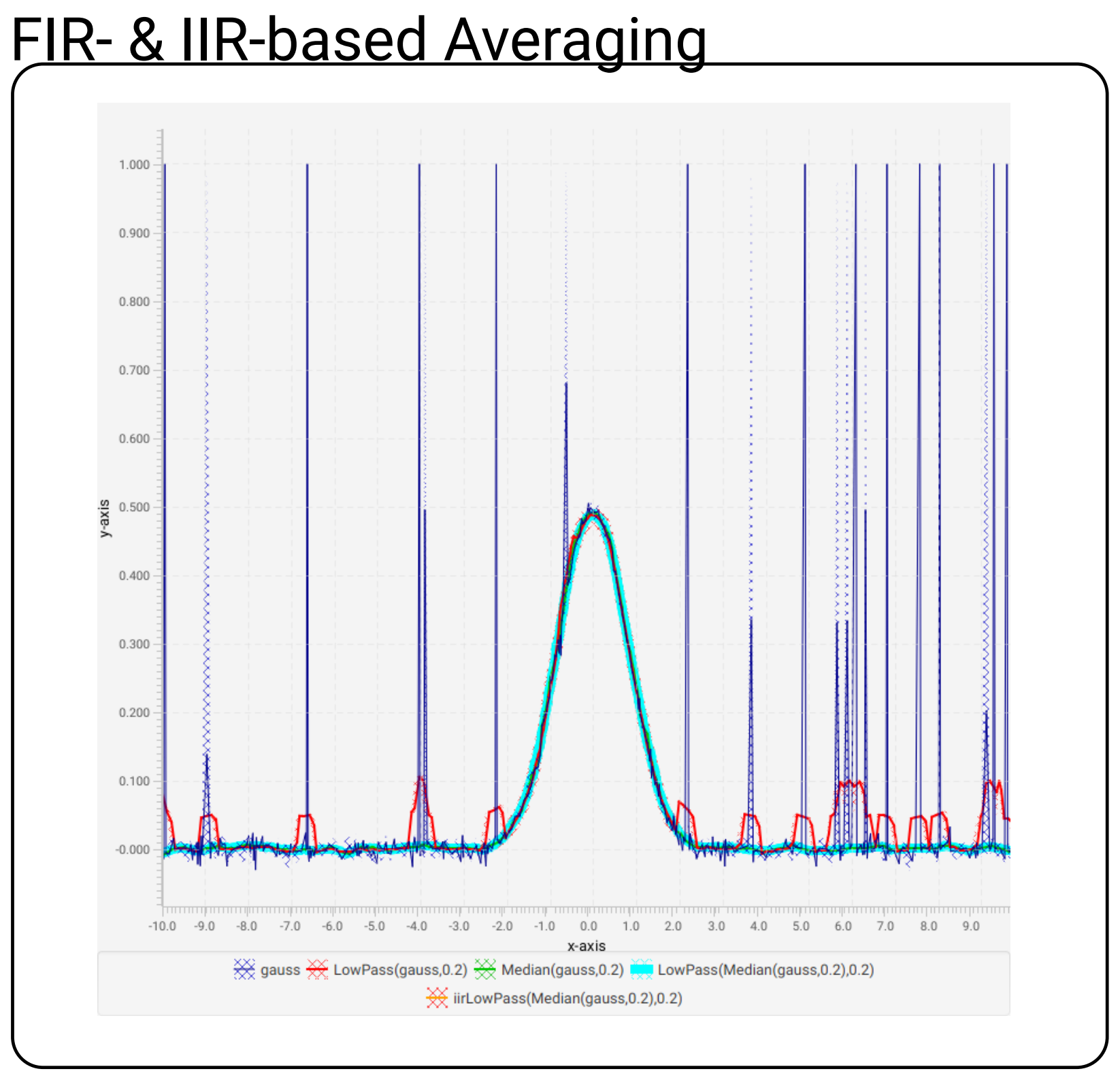
FIR- & IIR-based Averaging



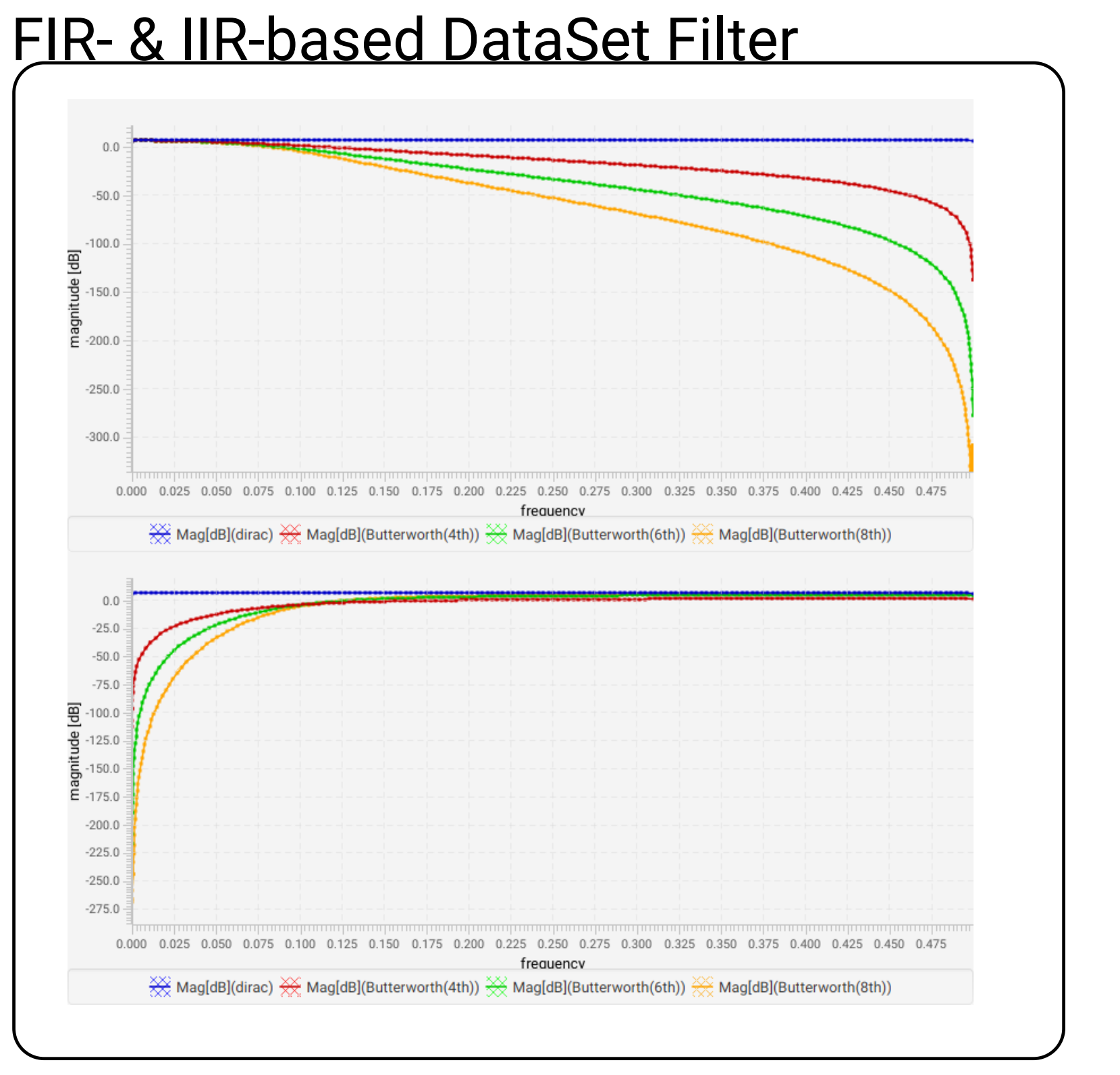
Differentiation & Integration



FIR- & IIR-based Averaging

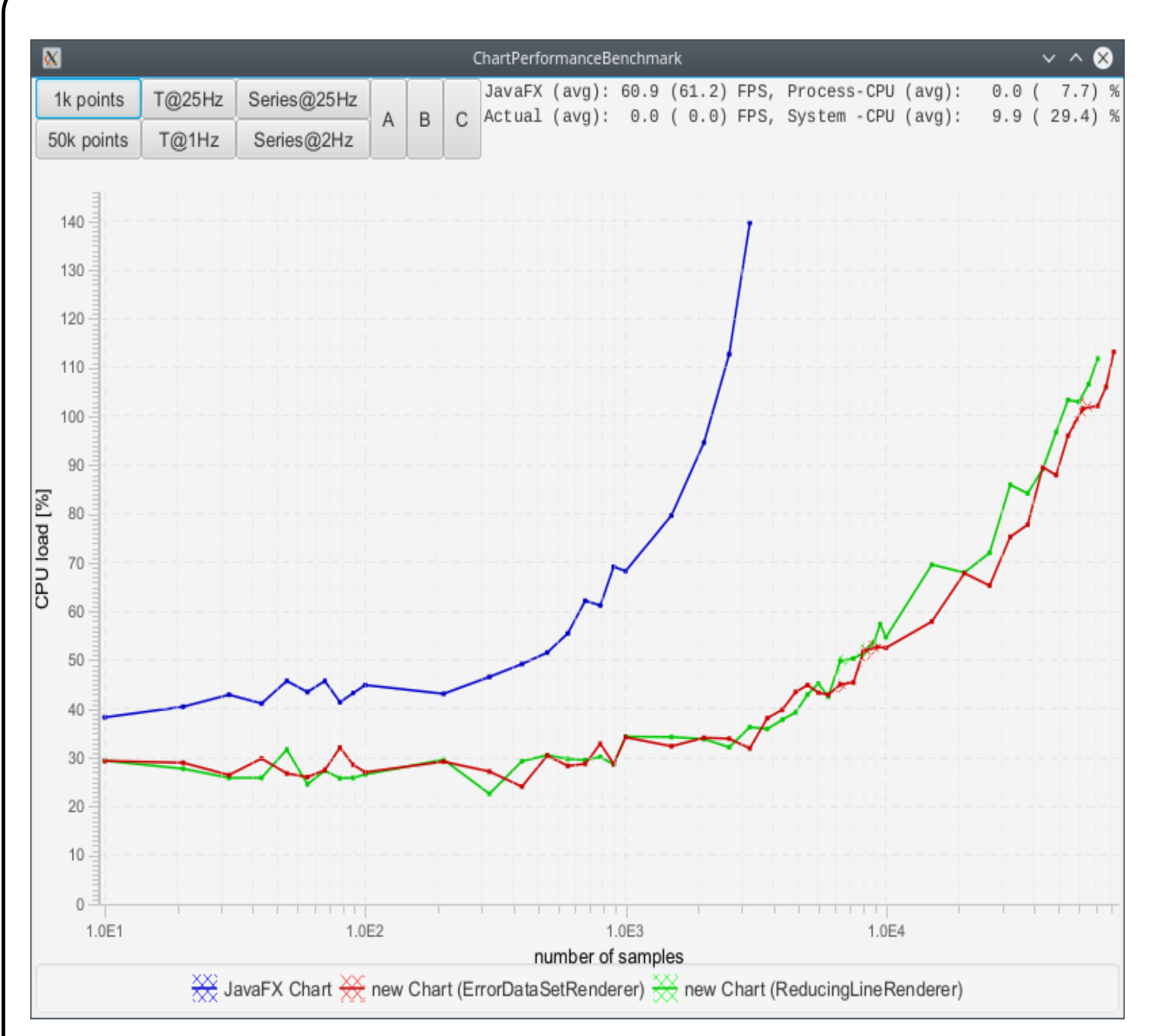


FIR- & IIR-based DataSet Filter

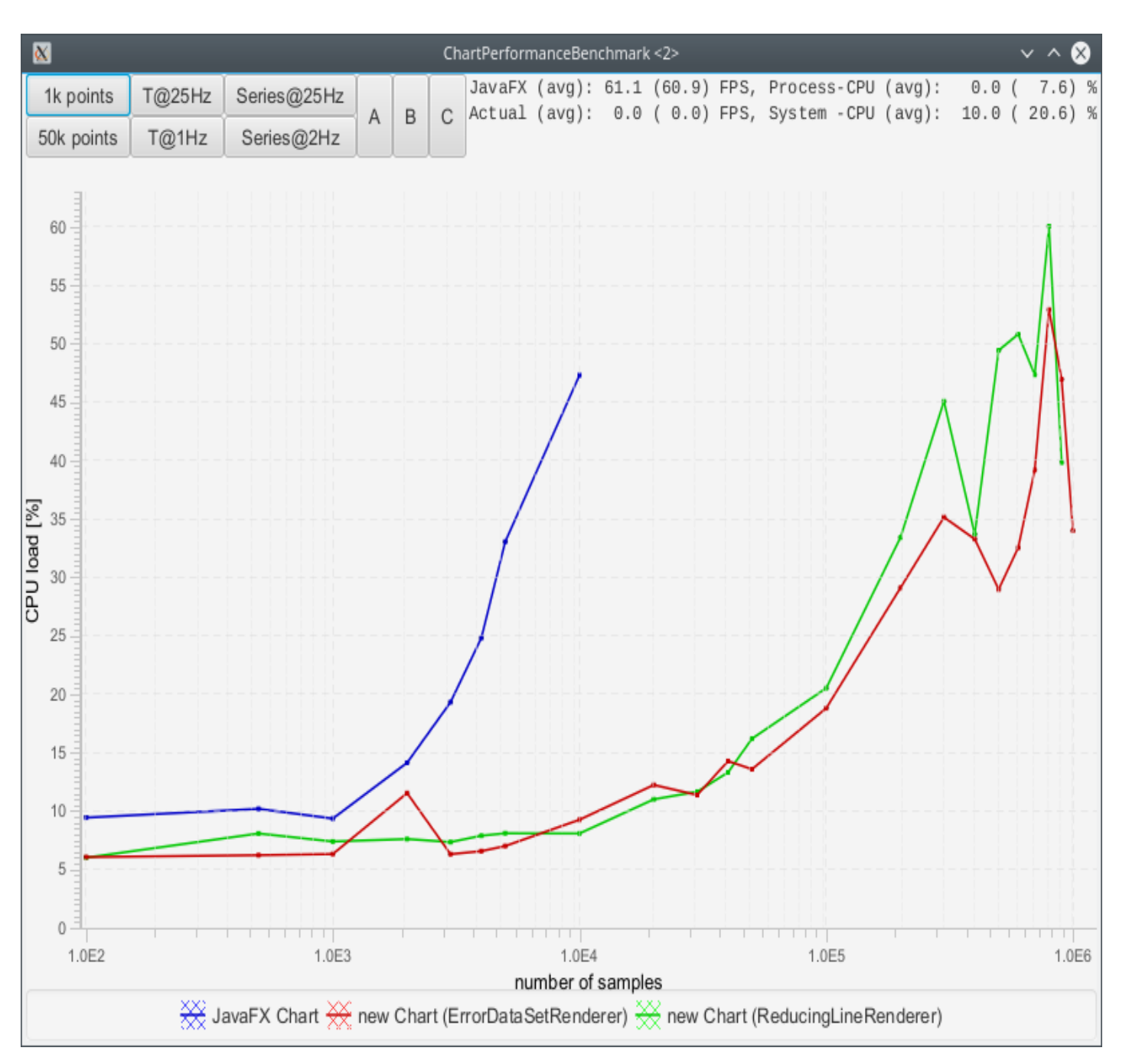


6 Chart Performance Comparison & Quo-Vadis

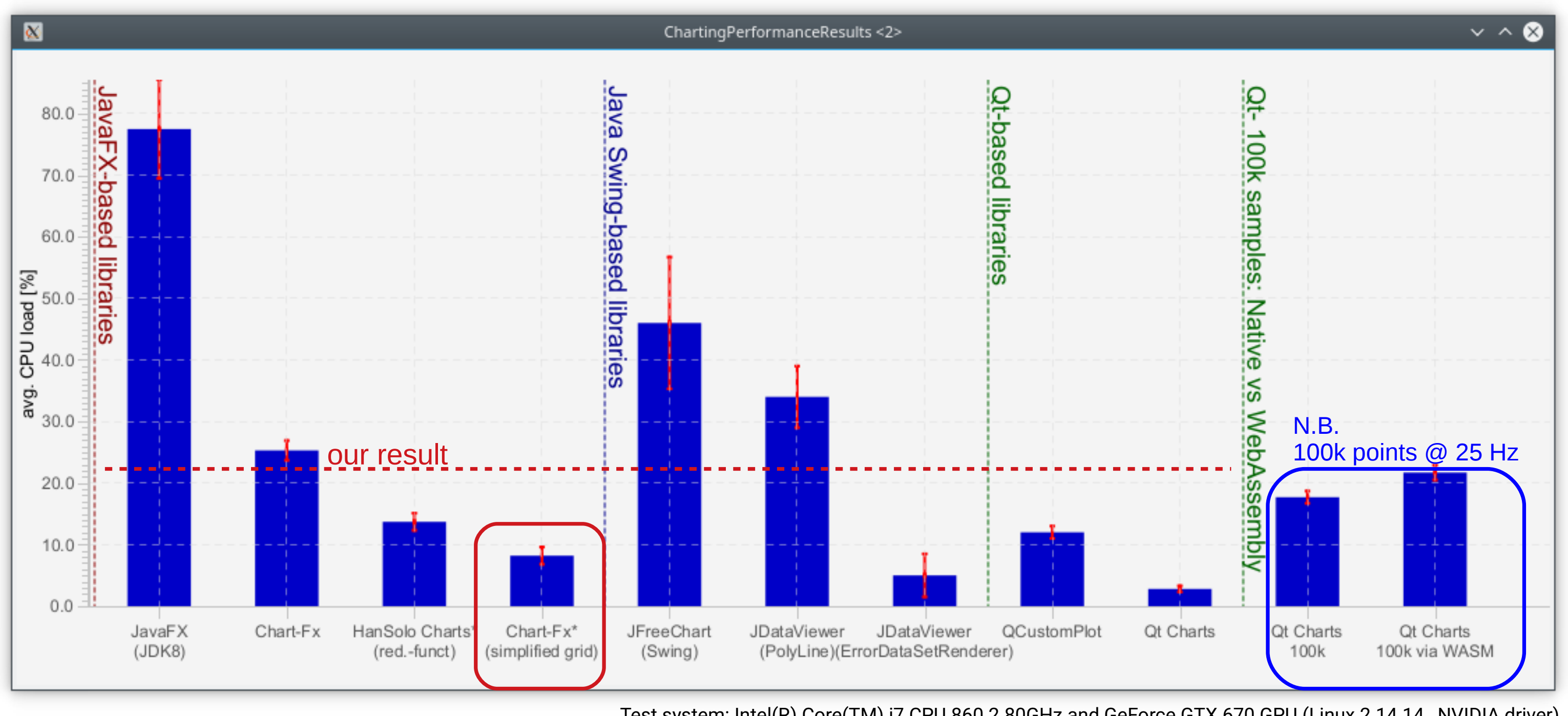
Life Update@ 25 Hz



Update of large DataSets@ 2 Hz



Performance Comparison with other libraries (750 points ↔ 30 seconds @25 Hz)



Test system: Intel(R) Core(TM) i7 CPU 860 2.80GHz and GeForce GTX 670 GPU (Linux 2.14.14, NVIDIA driver).

Conclusions

While starting out to improve the JDK's JavaFX Chart functionality and performance through initially extending, then gradually replacing bottle-necks, and eventually re-designing and replacing the original implementations, the resulting ChartFx library provides a substantially larger functionality and achieved an about two orders of magnitudes performance improvement.
Nevertheless, improved functionality aside, a direct performance comparison even for the best-case JavaFx scenario (static axes) with other non-JavaFX libraries demonstrated the raw JavaFX graphics performance – despite the redesign – being still behind the existing Java Swing-based JDataViewer and most noticeable the Qt Charts implementations. The library will be further maintained at GiHub and further used for existing and future JavaFX-based control room UIs at GSI.
Gained experience and interfaces will provide a starting point for a C++-based counter-part implementation using Qt or another suitable low-level charting library.

More Info:

https://github.com/GSI-CS-CO/chart-fx