

Impulse Measurement Software Coded by Artificial Intelligence

Stefan Fink and Sven Lautensack, Institute for Technical Physics, Karlsruhe Institute of Technology (KIT) - Campus Nord, Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen, Germany
E-Mail: fink@kit.edu

Abstract

Impulse voltage measurement is an important feature of modern high voltage technology. In large scale laboratories, special hardware and software measurement solutions are used for this purpose, whose development requires considerable resources. An alternative approach is presented with a USB oscilloscope and a low-cost mini-PC as hardware platform. The C and Python program FiLiPulse for a standard lightning impulse voltage measuring system is generated by artificial intelligence under the guidance of the measurement system operator. The operator has no relevant knowledge of programming. Parameter calculation of FiLiPulse is validated with impulse waveforms from an IEC standard. The program stores measured raw data and calculated fitted and filtered data for each shot in a single file and writes one common report of the calculated parameters of all the full standard lightning impulses within one directory.

1 Motivation

Standard lightning impulse testing is an important issue for verification of sufficient dielectric strength of high voltage power apparatus. The IEC standard [1] comprises a procedure for computer aided calculation of the standardized curve parameters. Manual calculation of the impulse parameters from waveforms in graphical formats is also included but this procedure would lead to less accurate results. Furthermore, for series tests a manual procedure is more time consuming. However, special impulse measurement hardware and software is often unaffordable for small scale labs. In case of the Cryogenic High Voltage Lab of the KIT-ITEP, oscilloscopes for manual reading of front time and time to half value according to the withdrawn standard [2] are still in use. Battery powered traditional oscilloscopes reduce ground loops. However, the used battery powered oscilloscopes have only few 1000 acquisition samples which is disadvantageous for standard lightning impulse tests with long acquisition duration, e.g. for breakdown tests with liquid nitrogen with potential time to chopping above 100 μs (Figure 1). The resulting long sample intervals make parameter determination imprecise.

Recent developments offer the opportunity for an affordable automatic impulse measurement solution for replacing the manual oscilloscope reading concerning parameter determination. The first development is the progress of USB-oscilloscopes for combinations of input voltage, samples, bandwidth, and vertical resolution which make the use of such devices feasible for high voltage applications in agreement with the instrument standard [3]. The second recent benefit is the availability of PCs at low cost. Hardware that is no more suitable for last Windows versions can be operated with current Linux because support for Linux software development is provided by several USB-oscilloscope manufacturers. The third development is the progress of artificial intelligence (AI) which offers the opportunity to let code special software where the included software for USB oscilloscopes is laborious for impulse volt-

age measurement. The present paper describes the experience of a project which has the target to develop a standard lightning impulse measurement program by an operator with negligible previous knowledge of programming.

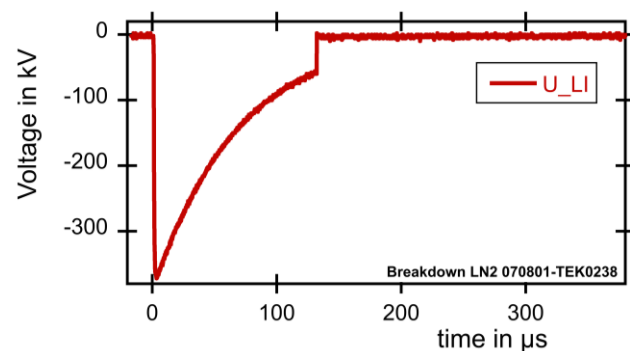


Figure 1 370 kV test for liquid nitrogen with time to chopping of 131 μs (hemisphere to plane setup, gap 13 mm, 0.108 MPa abs.).

2 Materials and methods

This chapter describes the resources and methods used to develop the AI coded program.

2.1 Hardware

A high voltage construction kit was used for standard lightning impulse generation. High voltage dividers Hilo-Test HVT 120 and 240 RCR were used for high voltage measurement and connected via N cables to a USB oscilloscope TiePie HS5-530XMS located in a metal box and supplied by a battery. It offers 14 bit vertical resolution in combination with a sampling rate of 100 MSa/s which is useful in case of future chopped wave measurement. The maximum input voltage of the oscilloscope is ± 80 V and the 3 dB bandwidth is 250 MHz. The signal from the USB oscilloscope was transferred via fiber optic transmission to the USB port of a Mini PC Intel NUC 5i3RYK with processor i3 5010U, 8 GB RAM (DDR3) and a disk with 224 GB.

2.2 Software and human resource

Linux openSUSE 16.0 with KDE plasma version 6.4.2 as graphical user interface (GUI) was selected. AI recommends Python (version 3.13.13) as programming software and for C programming gcc (15.2.0). AI software provided by the employer was Microsoft Copilot M365, GPT-OSS 120B, GPT 5.1, GPT 5.4, and Qwen 3.5 397B A17B.

Instructor for the AI is a high voltage lab operator with negligible prior programming knowledge.

2.3 Method

Due to a lack of prior knowledge, the software generation methodology itself had to undergo a development, depending on intermediate results or even on the specific AI and its interface. Initially, there was only the idea to have a program for measuring and recording standard lightning impulses according to [1], faster than within 1 minute per impulse and without intensive operator care. A next step was the discussion with AI (M365) if this makes sense, about the software needed, the file structure etc. The program was divided in modules. A document (spec.) was written for each module starting with specification of purpose and abilities (i.e. what should this module do), input and output. In an iterative discussion this document did also collect the information what coding should consider in detail. The first module communicating with the oscilloscope required downloading of headers, libraries, and application programming interface (API) descriptions. The used AI environment did not allow to read web pages and to download APIs directly. C coding of the module was started at a certain point of progress. Then an iterative process with compiling, testing, uploading libraries and information, debugging, module ability revising etc. was started which also affected the revision of the spec. Each module was tested on its own or after running the preceding module if it required input information. After the AI proposed several integration approaches for the modules, the operator selected an orchestrator to compose them into a program.

Testing a complete run of all modules with a reasonable result required operation of a high voltage circuit because no low voltage generator is available at this lab which reproduces the waveform of a standard lightning impulse.

3 Results

Two kinds of results are considered. The first part describes the final program version in detail under the aspect of what was achieved by the AI prompted by the operator and how it was tested. The second part examines the hypothesis that an operator can let produce AI the complete code without having programming experience. This part also reports major obstacles to obtain successful coding by AI.

3.1 Program description and tests

Some unsuccessful trials were initially performed as well as coding a complete executable program which does not comply with [1]. The following description covers only the second program with the title FiLiPulse.

The Project directory FiLiPulse contains several directories. The docs directory covers the specifications for the orchestrator and the modules and the descriptions of the standard [1] and the APIs. Optional files ReadMe.txt and Copyright.txt can be stored in this directory, too. The config directory contains configuration files with information shared between different modules e.g., the setting of the oscilloscope. The output directory keeps output information e.g. data of the last impulse measurement. C-Code is stored in the src directory and Python and bash scripts are stored in the script directory. Compiled C-code files are stored in the directory with the name build. Communication between program and operator is done via console in order to avoid compatibility problems for PCs with other Linux GUIs.

The workflow for the modules M1 up to M6 can be seen in Figure 2.



Figure 2 FiLiPulse workflow. After module M6, the operator can either continue with the same setup (M4) or change the setup (M3). Program exits are not shown in the workflow for clarity.

Module 1 shows the FiLiPulse banner with purpose information. Module 2 scans for the USB oscilloscope and detects if the communication between oscilloscope and PC is fine. Module 3 prompts the user for the connected high voltage divider, the data storage path and the data directory name. Then the predefined setup is displayed. This setup features a measurement duration of 200 μ s with pre-trigger ratio of 0.1 (i.e. 20 μ s), the vertical resolution is 14 bit, the selected sampling rate is 100 MSa / s, DC coupling, polarity dependent direction edge trigger with trigger level of 12.5% of the selected range and a trigger hysteresis of 5%. Afterwards, the user is prompted for polarity and a maximum absolute voltage guess. The appropriate voltage range setting is selected based on this voltage guess.

Module 4 loads the persisted setup stored by module 3, applies all setup parameters to the oscilloscope, arms the device for a single-shot acquisition, waits for a trigger (or forced trigger or quit), writes the acquired block to an atomic file (impulse_lastshot.txt), and copies this file to the data storage directory.

Module 5 acts as a hybrid orchestrator for the standard lightning impulse analysis according to [1]. It manages files and reporting in C, while delegating the numerical fitting and digital filtering to a Python script. The workflow

of this module starts with locating the newest raw data file and invokes Python to calculate the parameters namely maximum value of the test voltage (U_t), front time (T_1), time to half-value (T_2), and relative overshoot magnitude (β). The module writes the calculated parameters in the data file header and appends the reconstructed test voltage curve $U_t(t)$ as the 3rd column of the data file. Finally, it generates or updates the daily summary report and a fit protocol.

The Python script implements the core algorithm defined in [1] Annex B.3, D.1, and D.2 to extract standard lightning impulse parameters from the raw measurement data as follows (Figure 3):

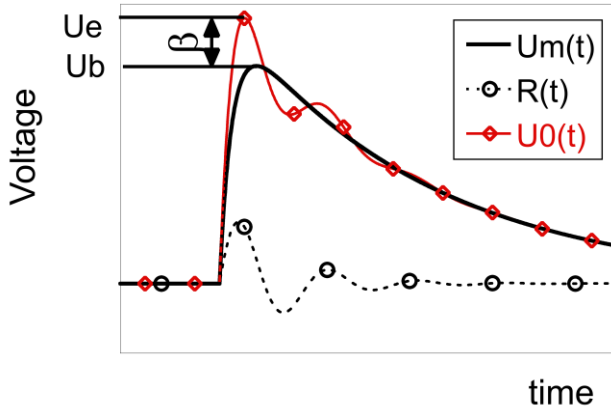


Figure 3 Offset compensated recorded curve $U_0(t)$, base curve $U_m(t)$, residual $R(t)$, and β .

1. Initial data processing:

The baseline offset of the measured data is calculated from the “zero” voltages of the first 15 μs and the offset-compensated curve $U_0(t)$ is generated. Extreme value (U_e) is detected and the fit time range is defined.

2. Non-linear fitting:

The double-exponential curve is fitted to the measurement data using the trust region reflective (trf) algorithm of scipy [4]. Normalized time and voltage scales are used to improve convergence. The iteration step number is reported in order to allow an estimation of the effort for the fit.

3. Waveform reconstruction and filter:

The base curve $U_m(t)$ is constructed based on the fit results for the total duration of 200 μs (which exceeds the time period selection of [1] annex B.3 h) in order to allow comparative display of the “ideal curve” with chopped raw data for large breakdown delay as discussed for Figure 1. Then the residual $R(t) = U_0(t) - U_m(t)$ was calculated. The zero-phase infinite impulse response (IIR) digital filter was applied to $R(t)$ to obtain $R_f(t)$. The test voltage curve $U_t(t)$ was obtained by adding $R_f(t)$ and $U_m(t)$.

4. Parameter calculation:

U_t and β are calculated based on voltage maxima. T_1 , T_2 were calculated by linear interpolation with the values at 30%, 90%, and 50% voltage according to [1].

Module 6 reads the data-storage directory from the configuration file and finds the newest measurement file that matches the file name pattern of the measurement day.

Then, module 6 extracts important specific record information from the file. It starts gnuplot (using Qt terminal for graphical output) and draws two vertically stacked sub-plots namely the complete 200 μs waveform and a 7 μs zoom with the front and the peak. The file name, trigger time and parameter information are displayed. Finally, it prompts the user for how to continue, i.e. proceed testing with same settings, continue with setting changes (e.g. change polarity) or to quit.

The orchestrator script for the work flow of the modules is written by the AI, too. The user starts the program in the console by calling this orchestrator script. The orchestrator begins with module 1. The next module with a number higher by one was called by the orchestrator after termination of the module without errors, each. In case a fault happens the program aborts with an error message. In case the final module 6 was successfully terminated (Figure 4) the orchestrator calls either the setup module 3 (for changing the setup) or it calls module 4 for acquiring with the same settings or it quits the program depending on user decision.

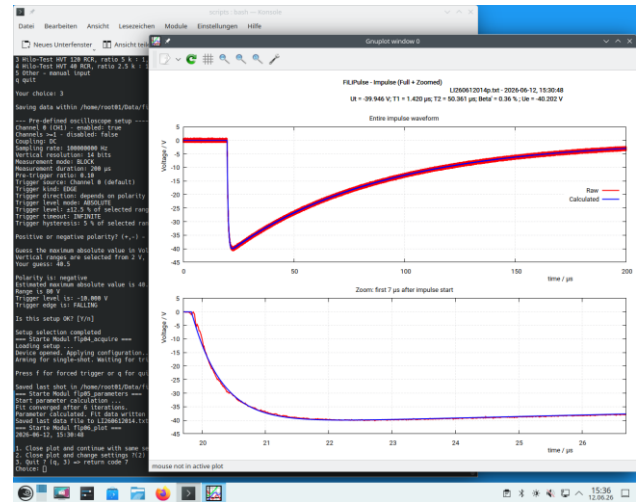


Figure 4 Screenshot of -200 kV impulse measured with FiLiPulse and display of U_t , T_1 , T_2 , β . The plots show full range of 200 μs (top) and zoomed view with rise and peak.

First parameter calculation verification was done with artificial self-generated data for full standard lightning impulse with negligible oscillations which is the intended use of this software. Then comparison with IEC generator [5] was done showing agreement within the specified tolerances for full standard lightning impulse even in case of oscillations (see chapter discussion for details). Hence the calculation mechanism is considered as validated.

High voltage tests were performed (Figure 5) with two high voltage dividers connected to an impulse generator. One TiePie oscilloscope was operated with FiLiPulse and one Tektronix oscilloscope was used for comparison on the other divider, each. Standard lightning impulse tests of both polarities were applied up to 200 kV. Agreement between FiLiPulse and manual oscilloscope reading was obtained. However, the comparability of the high voltage tests was limited because of differences of the high voltage

dividers and the accuracy of manual reading of the Tektronix device. Therefore, the parameter verification was mainly confirmed by the software comparison and the high voltage test was more intended as an overall and electromagnetic compatibility check.



Figure 5 High voltage test setup.

Hence, the total number of 41 high voltage impulses with successful measurements and no software malfunction was considered as sufficient to start with first normal high voltage operation. It takes 20 s to save the data for one impulse on the Tektronix oscilloscope. Instead, one complete impulse measurement with FiLiPulse including data saving and tentative visual control by the operator can be done in case of unchanged settings in less than 10 s which is well below the target time of 1 min. Hence, it can be concluded that the performed software and high voltage tests were successful for full standard lightning impulse.

3.2 Coding

In the year 2025, the user interface for the first used AI Microsoft Copilot M365 had been limited to a textbox with poor character recognition which caused confusion for initial coding trials. In contrast, guidance by AI about computer wordings, hardware requirements, file hierarchical structure, user permissions etc. and discussions about selection of program language, version control etc. worked well. During January 2026, the character recognition was improved and file exchange was unlocked. Furthermore, the AI seemed to make quality progress over a time of several weeks, although it still sometimes forgot important instructions within a single chat session. However, internal regulations for the communication between AI and user made coding difficult due to context length limitations and an imposed internet search. Finally, this AI was confused and established a program according to an outdated standard because it prioritized internet search results higher than the user instructions and spec. This first coded impulse measurement program was executable standalone including a filter depending on bandwidth of used high voltage divider. Therefore, this code resulted in a faster measurement compared to the similar used manual method. How-

ever, the standard lightning impulse parameter determination was only in agreement with the outdated method and not with [1]. Hence, one can summarize as result that the first coded program with AI failed.

Then, access to the AI GPT-OSS 120B with context length of 128 K was possible. Information exchange was only available by the chat text box. Figures must be described by the user and no internet search by this AI was possible. The most important improvement was that the AI tried to follow the user instructions and was not confused by internal regulations like a useless mandatory internet search. No outdated APIs received by the internet disturb the work flow. The disadvantage was that all information (e.g. current API details) must be transferred to the AI by the chat. Finally, the orchestrator code and the C-code for all modules were established by AI. However, the obtained parameter calculation was not satisfying and often the calculation failed. It turned out that this problem was caused around the code section around the fit. The fit was based on the Levenberg-Marquardt solver of the GNU Scientific Library (GSL) [6].

Further trials with the meanwhile available AI GPT 5 and Qwen 3.5 failed, too. After discussion with the AIs, it was revealed that all three AIs selected solvers from the same library. After intensive unsuccessful debugging trials, this way was aborted by the operator. The operator instructed the AI Qwen 3.5 with the context length of 256 K to write the mathematical calculations by Python code and modify the C-Code of module 5 for calling the Python script and for operating the other functions like module input and output.

The selected AI was able to produce the parameter calculations after some debug support (repair logical errors) by the operator. However, it turned out that instead of the Levenberg Marquardt algorithm the trust region reflective algorithm was included in the first successful operating code. Nevertheless, the second trial of AI coding resulted in an executable program which delivers parameters based on a least squares fit and an IIR filtering.

4 Discussion

During the initial guidance and discussion phase it was decided that the file structure shall follow the Filesystem Hierarchy Standard (FHS) [7]. Python seems to be considered as simpler compared to C regarding AI coded programming according to [8] [9]. However, comparison on energy consumption show potential higher efficiency for C (or C++) [10]. In addition, the AI was (as nearly always) very optimistic to solve the task with C, too. Therefore, the operator decided that C is used by the AI for the program code. Python is selected for support tasks outside the program with single or few use (e.g. for data comparison). The operator experience is that the code generation process with Python was a little bit faster than with C because of the omitted compilation process but the total difference was negligible because most time requirement had to do with logic fault detection which is in principle independent of the language itself.

AI generated coding is a fast-growing software programming method [11]. The fast improvement of the AI tools was even noticed within the few months of the performed project. The program was coded by AI based on operator prompts (or file exchange with Copilot). Nevertheless, a simple vibe coding strategy [12] by just prompting the error codes until the program works would have not resulted in a useful standard lightning impulse measurement program. Breaking logic cycles, supervise high voltage specific issues and helping with new ideas was essential for debugging and progress. Such a method may be assigned as human supported coding. In a more formal manner this can be extended to vise coding described in [11]. For programmers it was reported in [13] that while using AI improved the average completion time of the task, the improvement in efficiency was not significant in a study. By contrast, in the FiLiPulse project it should be noticed that the success would have been impossible without AI coding because the operator was not able to do this job. In addition, the operator acquired a lot of skills by collaborating with the coding AI.

Comparison of the FiLiPulse program with special high voltage measurement software [14] – [16] was not possible because no such software is available for the operator. However, an IEC standard [17] delivers a data generator (TDG Version 2.1.0) with parameter evaluation for comparison. According to the purpose of FiLiPulse only the 29 “LI” full lightning impulse data were generated (A1 up to A12, M1 up to M17). It is not possible to process IEC test files directly since FiLiPulse is dedicated to standard lightning impulse measurement with a predefined setup. Therefore, the Python script of module 5, which makes the parameter calculation, was used for a core functional duplicate but enhanced as stand-alone application. Further modifications were implemented in the stand-alone application, e.g. to accept other input file format and measurement durations. The fit and filter procedures remain the same. Fortunately, the FiLiPulse sampling rate of 100 MSa is also the same for the IEC test files. Four T2 parameter results (A7, M2, M3, M5) of FiLiPulse were outside the tolerance of 1%. The parameters of T1, Ut and beta’ were fine for all 29 files. None of the test impulses with incorrect T2 calculation fulfil the 1,2/50 standard lightning impulse definition. In contrast, all parameters are calculated correctly by FiLiPulse within the tolerances in case the waveform is a 1,2/50 standard lightning impulse. Hence, the calculation target was already reached successfully.

Nevertheless, two other scipy least squares fit algorithms [5] namely Levenberg Marquardt (lm) and dogleg with rectangular trust regions (dogbox) were compared with the implemented trf method. Limitations to avoid bound problems were added for the lm algorithm that prevent searching in impossible regions (e.g. negative times). The improved lm fit produces 8, dogbox more than 10 parameter values outside tolerance. Hence, trf algorithm was kept although lm algorithm is mentioned explicitly in [1].

The influence of the parameters and the percental deviation between FiLiPulse and IEC data were investigated.

Dependence between the deviation for T2 was found for T1. Some relationships are shown in Figure 6.

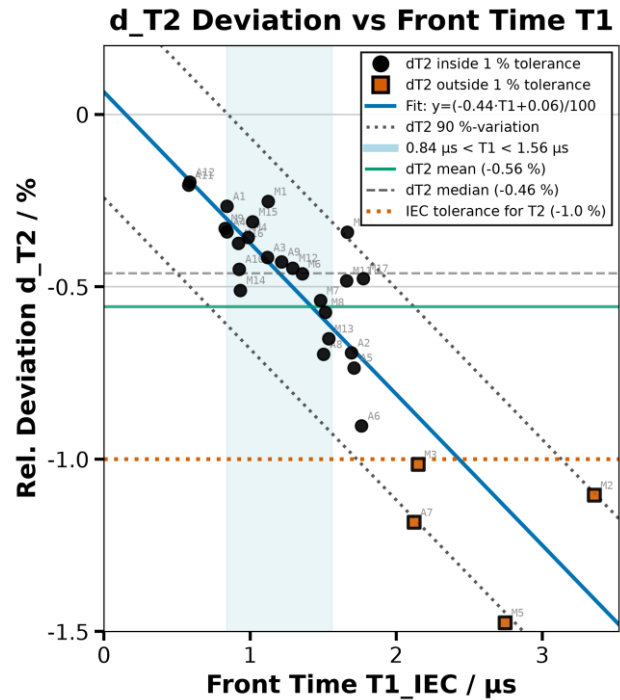


Figure 6 Percental deviation d_{T2} of calculated T2 by FiLiPulse from T2 of IEC standard vs. T1 of IEC standard.

All T2 deviations between FiLiPulse and IEC values are between -0.2% and -1.48%. 25 values are within the 1% tolerance for T2 (Figure 6, black circles) whereas four values are outside the 1% tolerance for T2 (red squares), as already mentioned. The influence of these four values outside the 1% tolerance causes a shift of the mean value away from the median. A T2 shift for +0.5% would bring all T2 values in the tolerance range of 1%. This may be considered as a not acceptable hack because the reason for the deviation is unknown. On the other hand, one can argue that it is acceptable to adjust such a simple low budget system against the high-tech solutions of IEC for 0.5% as one would do so with a low cost multimeter, too.

However, looking on the linear fit line in Figure 6, a value dependent adjustment would be preferred instead of the simple 0.5% addition. It should even be noticed that the four d_{T2} calculations outside the 1% tolerance are also the calculations with highest distance to the 1.56 μ s limit. Hence, in case of limiting the adjustment near the standard lightning impulses, it may be favorable to ignore the values for the regions with e.g. $T1 > 2.0 \mu$ s, where all the calculations outside the tolerance occurred and recalculate the data for the fit line again. This would lead to a better adjustment towards the IEC values which could be even done individually for a specific T2 range in case of special measurements. Manual reading of T2 with the Tektronix oscilloscope had been done only with steps of 0.5 μ s. In addition, adjusting T2 between 40 μ s and 60 μ s with the available resistors had been never critical with the used high voltage construction kit (in contrast to adjust T1).

Therefore, it was decided to keep the T1 dependent relationship for T2 in mind and presently let the FiLiPulse calculation for standard lightning impulse measurement in the already valid way as it is. The T2 value can be adjusted depending on the specific T1 data in case of future higher requirements for T2 accuracy.

High voltage tests were performed up to 200 kV in a screened experimental cabin. The screened divider cable is guided through a copper tube / additional copper screen to the screened measurement cabin and connected to the USB oscilloscope located in a grounded metal box. Such a setup reduces the influence of earth currents [17] resulting in a low disturbed measurement signal even at the USB oscilloscope input levels of few volts up to few 10 volts. It is not proven up to now, that the USB oscilloscope can work also under unscreened onsite condition with poor shielding.

A following project with AI coded software is foreseen for standard switching impulse measurement. This is interesting for the $d_{T2} = f(T1)$ function because of the higher values of T1 compared to the lightning impulse. Alternative fit software may be tried out not limited to C or Python in case of unacceptable results. As an alternative, the AI may even create its own code for a fit algorithm. Chopping handling according to [1] or automatic evaluation and archiving of thousands of files generated by traditional oscilloscopes in the past would also be an option for a project.

5 Conclusions

The FiLiPulse USB oscilloscope software is a remarkable progress for the KIT Cryogenic High Voltage Lab concerning full lightning impulse parameter measurement accuracy and automatic parameter recording compared to the manual reading of a traditional battery powered 9 bit oscilloscope with only 10 k samples. The workload for the operator is reduced to a minimum by the tailor-made software and the run time is negligible on the 11 year old mini PC (i3) which works fine with actual Linux.

The AI generated all the code and gave support for PC handling including file permission management, software project structure recommendation, version control and result evaluation (e.g. Figure 6 is set up by AI). Hence no prior programming knowledge is needed for establishing the program. However, the AI made sometimes nonsense or was confined in logical loops during troubleshooting. Hence, the software project would never be successful by just sending error messages back to the AI and wait until it works. Accuracy of parameter calculation was validated by comparison with 29 IEC lightning impulses of [5]. FiLiPulse trf algorithm instead of lm algorithm mentioned in [1] was used because of better agreement. For full standard lightning impulses, the calculated results are within the tolerance range of [5] for all parameters U_t , T1, T2, and β . No disturbance of FiLiPulse system occurred during high voltage tests. Full specified functionality has been achieved. Hence, the activity is considered as successful and performed with reasonable effort.

Standard switching impulse software coding by AI is foreseen as the next project.

6 Literature

- [1] High-voltage test techniques – Part 1: General terminology and test requirements, IEC standard 60060-1, 2025.
- [2] High-voltage test techniques – Part 1: General definitions and test requirement, IEC standard 60060-1, 1989.
- [3] Instruments and software used for measurements in high-voltage and high-current tests - Part 1: Requirements for instruments for impulse tests, IEC standard 61083-1, 2021.
- [4] The SciPy community: “least_squares”, SciPy documentation, [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.least_squares.html#scipy.optimize.least_squares. [Accessed June 05, 2026].
- [5] Instruments and software used for measurements in high-voltage and high-current tests - Part 2: Requirements for software for tests with impulse voltages and currents, IEC standard 61083-2, 2013.
- [6] The GSL Team: “Nonlinear Least-Squares Fitting,” GNU Scientific Library. [Online]. Available: <http://www.gnu.org/software/gsl/doc/html/nls.html>. [Accessed June 05, 2026].
- [7] LSB Workgroup of The Linux Foundation: “Filesystem Hierarchy Standard,” The Linux Foundation, [Online]. Available: <https://refspecs.linuxfoundation.org>. [Accessed: June 05, 2026].
- [8] Mukhopadhyay, C.: A Comprehensive Survey on Programming Languages for AI and ML. Zenodo. 2025. doi: 10.5281/zenodo.18021195.
- [9] Amiri, S.M.H: The Two Faces of Python: Bridging Simplicity and High Performance. Software Engineering Ejournal. 2026; 9(67):1–22. doi: 10.2139/SSRN.6329940.
- [10] Díaz-Álvarez J., García Arenas M., Sánchez-Venegas A.; Romero López G., de Vega F.F., Valdivieso P.A.C.: Hardware and Software Influence on EAs Power Consumption, Lecture Notes in Computer Science. 2026.
- [11] Faragó, David: From Vibe to Vise Coding: Addressing the AI-Generated Code Quality Crisis. Software-technik-Trends 45(3): 30-34, 2025.
- [12] Karpathy, A.: Vibe Coding, X, February 2, 2025, <https://x.com/karpathy/status/188619218480814938>.
- [13] Shen, J. H., & Tamkin, A.: How AI impacts skill formation. arXiv: 2601.20245, 2026 doi: 10.48550/arXiv.2601.20245.
- [14] AMOTronics, SATURN Studio II, “Saturn transient recorder“, Edition v1.07e, p. 6.
- [15] HIGHVOLT, HiRes, “HiRes transient recorder“, p. 3.
- [16] Strauss, WinTR-AS for windows, TRAS F2-E.
- [17] Kind, D., Feser, K.: High voltage test techniques, 2 nd edition, Newnes, 2001, pp. 119ff.