Optimization of the performance of a voltage measuring station using genetic algorithm

Cite as: Rev. Sci. Instrum. 92, 064704 (2021); doi: 10.1063/5.0044438
Submitted: 16 January 2021 • Accepted: 21 May 2021 •
Published Online: 4 June 2021

Mehbub A. K. Nooruddin,[a] Subhasish Roy, and Nilotpal Bhandary

AFFILIATIONS

Department of Physics, Visva-Bharati, Santiniketan PIN 731235, India

[a] Author to whom correspondence should be addressed: mehbub.aknooruddin@gmail.com

ABSTRACT

This article describes the use of the genetic algorithm (GA) for the optimization of the performance of a voltage measuring station. This procedure can be used for the instruments for which regular and systematic calibration is required for better performance. Here, a GA-based correction module has been used to internally store a few previous measured-value and measured-time pairs and to use these values to correct the current readings. Sanity-checking runs against zero and standard voltages from time to time and auto-calibrates itself. The genetic algorithm libraries have been developed along with the correction module. This system can also be considered as a test bed for the study of self-calibration algorithms.

I. INTRODUCTION

High-precision voltage measurement is very important for most experiments. High-precision voltmeters are commonly available in the market, and in most cases, these are very expensive. These voltmeters need systematic and regular calibration. Self-calibration algorithms, such as Artificial Neural Network (ANN) algorithm and Genetic Algorithm (GA), can be used to calibrate such systems for better accuracy. The GA has already been used for instrument designing and optimization of neutron scattering instruments,[1] design and optimization of a higher harmonic cavity,[2] multilevel converters,[3] optical zooming in cellphone cameras,[4] calibration of a spectrometer,[5] digital power metering,[6] image reconstruction,[7] blood cell recognition machine,[8] and analog fault detection.[9] It has also been used for a causality study of the magnetic permeability-frequency[10] function of NiZn. Recently, GAs are becoming a very important tool for the optimization of the nonlinear beam-dynamics in accelerator physics.[11,12]

Here, we have described the development of a GA-based correction module to auto-calibrate a low-resolution ($\sim 3\frac{1}{2}$ digits) and low-accurate homemade voltage measuring station (VMS)[13] to achieve better resolution. The VMS is basically a dual-slope integrator, and integrating instruments of these kinds are slow but offer a significant advantage of averaging and have data smoothing with a reduction in contribution from high-frequency noise. However, polarization of the integrator capacitor forms the largest source of error, effectively reducing the achievable resolution and accuracy of the instrument. Unlike errors due to offsets in the active components, which are relatively constant and hence can be calibrated out, polarization errors are history-dependent. The values and times of the previous readings affect the current readings in ways that also depend on the humidity and temperature of the surroundings. The polarization effects of the capacitor cause electret-like behavior, wherein a voltage builds up across a previously charged capacitor even after momentarily shorting the terminals. This is a common problem with integrating A/D converters. Better integrating capacitors would improve the performance,[14] but in this work, the idea is to check whether the GA-based correction module leads to genuine improvements and an ordinary capacitor would test the efficacy of the method more thoroughly.

Here, we have used a daemon-based program, which internally stores a few previously measured-value and measured-time pairs. The idea is to use these values to estimate polarization residues and hence correct the current readings. Since evolutionary algorithms are self-training, temperature and humidity effects will be incorporated automatically as the daemon executes random sanity-checking runs against zero and reference voltages from time to time and uses the variable voltage-time data to calibrate itself in a self-consistent manner. The GAs allow for "hill climbing," which can help in locating global minima in complex function spaces with local minima, which would tend to trap the iterations if standard derivative-based fitting methods were used. The initial values would also influence the convergence. Here, the form of the fit function is open to be changed if necessary. However, if non-analytic functions such as absolute values are to be incorporated, derivative-based methods would fail. The GA-based method, however, will always lead to better values irrespective of the precise nature of the function chosen. Hence, this approach leaves ample scope for improvements as necessary without necessitating a complete re-write. The GA is computationally slow, but since the daemon will keep running with "checkpointing" (saving the latest set of coefficients to the disk) at power-down, the effect will be cumulative and computation speed will be fast.

In this paper, we have reported the correction module that can be used to achieve better resolution and the system that can be considered as a test bench for the study of self-calibration algorithms.

II. THE INSTRUMENT

The instrument (VMS) is basically an A/D converter, which works using the principle of dual-slope integration. It has been designed for such experiments where concurrent measurements of different parameters (in terms of voltages) are required. It has 16 input lines (channels), which can be connected to 16 different measurement units. The VMS is controlled through a microcontroller (AVR Atmel Atmega128) that is connected with an Ethernet controller (ENC28J60) with a pre-set MAC address that allows us to control and operate the VMS over a network.

A. The hardware

Figure 1 shows the schematic of the 16-channel VMS. It has two parts: one for channel selection and the other for A/D conversion. Each channel is selected with the help of a 4-to-16 demultiplexer/decoder (LM 74154) followed by 16 fast comparators (4× LM 339) driving 16 CMOS switches (4× CD 4066). The four control lines of the demultiplexer are connected to four pins of PORT-D of the AVR. Depending on the pattern output of PORT-D, one of the comparators that are connected to the output lines of the demultiplexer will be high. The outputs of the comparators are connected to the control pins of the 16 switches (SW1–SW16). The output of the 16 switches is shorted together and is connected to the input of an inverting adder. The input lines of the 16 switches act as the 16 selectable inputs of the VMS.

Here, the A/D conversion is done via dual-slope integration. A CMOS switch (SW19) controlled by the AVR has been used to short the capacitor (paper capacitor) to the ground potential at the beginning of each measurement. The voltage to be measured is level-shifted by using the inverting adder to make it always positive, which allows the apparatus to work in the positive and in the negative voltage domains. At first, the shifted voltage is applied to the input of the integrator through the switch SW18 (controlled by the AVR) for a fixed time (using the fixed delay of the AVR) that ramps to a value that is proportional to the input voltage. Next, a fixed negative reference voltage (LM 399) (scaled down to −2 V) is applied to the input of the integrator through the switch SW17, and the time required for the integrator output to ramp back to zero is measured[13] using a 16-bit counter of the AVR. The time for the ramp back is now proportional to the original shifted input voltage. The scheme of the conversion is shown in Fig. 2. The zero crossing is sensed via a zero-sensing comparator followed by a transistor in the open-collector configuration with a 5 V pull-up resistor, which can be read directly through an input port of the AVR.

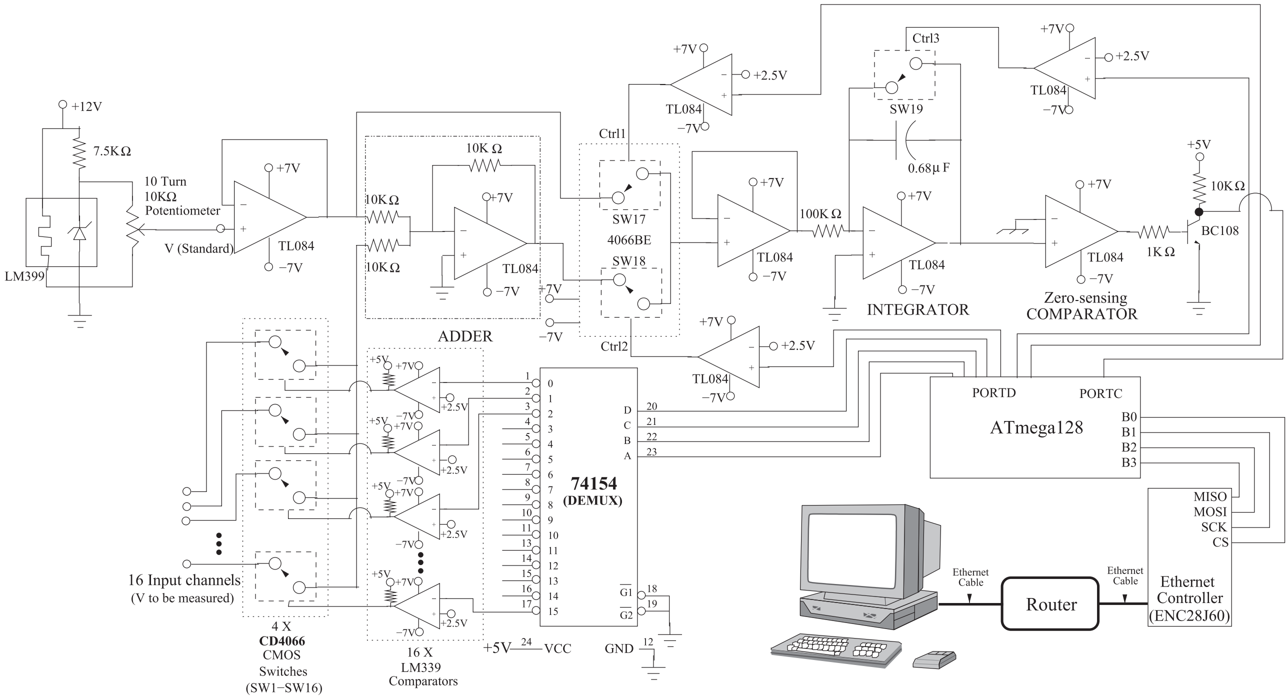

FIG. 1. Schematic of the 16-channel VMS.

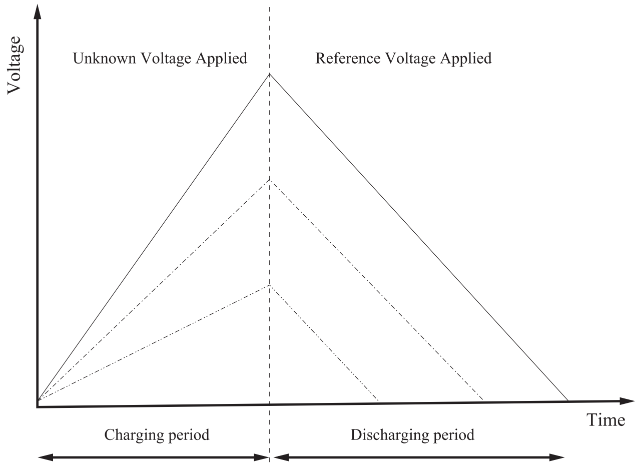

FIG. 2. Wave form of the output of the integrator.

If $V_{reference}$ and $V_{unknown}$ are the reference and unknown voltages (voltage to be measured) and $t_{reference}$ and $t_{unknown}$ are the corresponding times, then

$$V_{unknown} = \frac{t_{unknown}}{t_{reference}} \times V_{reference}.$$

B. The software

The software that is used here has been written in C++ with the Message Passing Interface (MPI) protocol to pass messages among the different modules. Since the VMS has 16 channels, we have incorporated "**daemon**" programs to avoid the conflict of request from different client programs.

1. The microcontroller daemon

The full sequence of instructions and operations is controlled by using this daemon program (**avrd**), which runs on a computer in the background. It waits to receive requests from client programs by opening a port through MPI. Whenever the daemon program receives any request from the clients within its "local network," it first checks whether the microcontroller is in the ***available*** mode or not. If the microcontroller is not available (i.e., busy in doing other measurements), it holds the request in **queue** and waits until the microcontroller is free. If the microcontroller is in the ***available*** mode, the daemon accepts the request from the clients and sets the ***unavailable*** flag first. It selects the particular channel as requested by the client, does the measurement by sequentially connecting the *unknown* and *reference* voltages, and sends back the measured voltage to the client. After each measurement, the daemon shorts the capacitor of the integrator to the ground potential through a CMOS switch (SW19 in Fig. 1), sets the ***available*** flag, and waits for the next request.

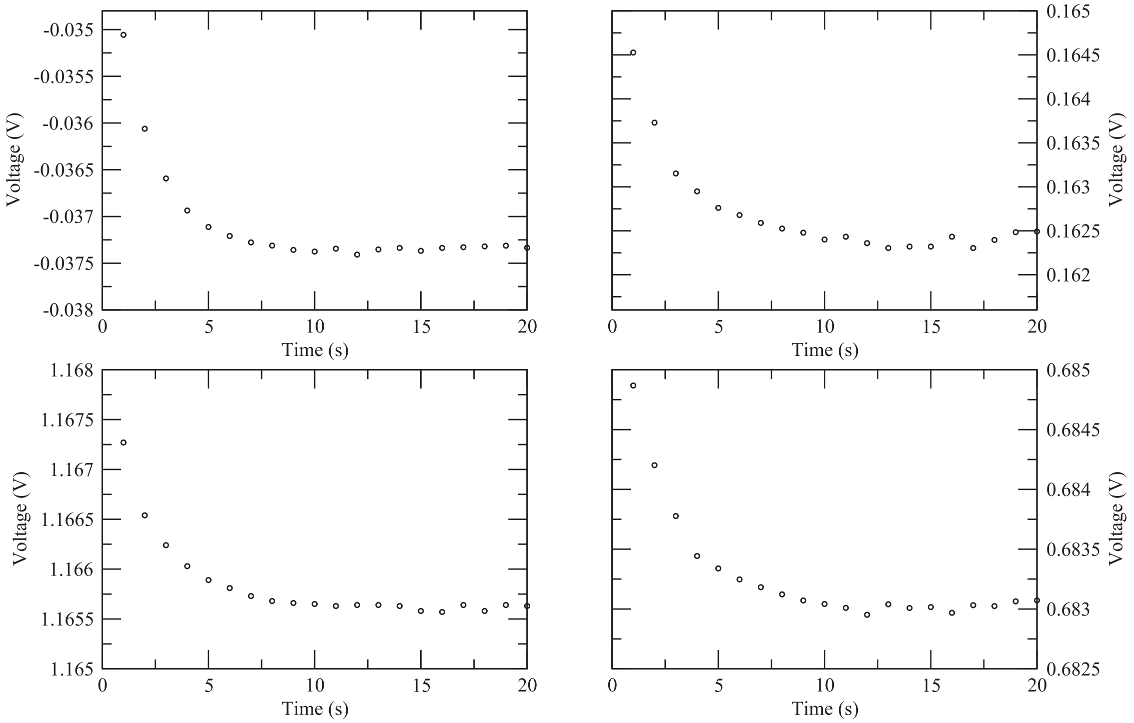

FIG. 3. Curves showing the capacitor discharging effect for different standard voltages. Each of the measurements was obtained by the repeated measurement of 1.840 58 V and a standard voltage with increasing time intervals. Top-left: 0 V. Top-right: 0.208 03 V. Bottom-left: 1.240 65 V. Bottom-right: 0.742 21 V.

III. PERFORMANCE

To test the performance of the instrument, we have tried to calibrate it by using a known standard voltage of 1.840 58 V measured by using the *Keithley Electrometer 6517B*. After calibrating the instrument, when we tested the stability of the measurement, we noted that if a small voltage was measured just after the measurement of a large voltage, the measured value showed an error. After repeated observations, we found that the value of the measured voltage depends on the time between two successive measurements. It is also seen that the measured value changes according to the humidity and temperature of the surroundings. We have tried to correlate the effects of temperature and humidity on the measured value throughout one year but failed to find any definite relation. As the output of the integrator of our measuring station is shorted to ground through a CMOS switch after each measurement, the effect of the one measurement is expected not to affect the next. However, it is seen that the current measurement is affected by the previous measurement. To investigate this effect, we have taken different datasets where the measurements for each set were carried out by successively measuring a large value followed by a comparatively small value for different time intervals. We fixed the larger value at 1.840 58 V and the other values as 0, 0.208 03, 0.742 21, and 1.240 65 V. These voltages are taken by scaling down LM 336 and LM 385.

Figure 3 shows the plots for different datasets where we can see that the nature of the curves initially looks like capacitor discharging, and they flatten out after a time interval of ~ 13 s. It is clearly seen from the graphs that the polarization effect of the capacitor plays a crucial role in determining the accuracy of the measured voltage. To calibrate such curves, we need a linear voltage shift and an exponential decay rate. We have noted that the coefficients of the exponential part and the linear shift are not constant for all the datasets. The exponential part depends on the temperature and humidity of the surroundings, whereas the part of the linear shift depends on the amplitude of the unknown voltage. Therefore, to optimize the results in such a situation, we have introduced a Genetic Algorithm (GA) component in our software.

A. Genetic algorithm correction module

Genetic Algorithms (GAs) are a method of optimization based on search procedures inspired by natural evolution.[15,16] Instead of using the "canonical" genetic algorithm scheme, we have used the ***modified genetic algorithm*** here.

We start with arrays of randomly generated bits (an unsigned char with size 20), which form ***chromosomes***. A "C++" class, ***necklace***, has been developed for this purpose. It allows for logical bit manipulations on this entire chain, such as setting and resetting bits, flipping bits, and extracting values from user-settable bit areas.

A class ***agar*** handles the genetic algorithm part. It allows for single-point crossover at an arbitrary point between two chromosomes. Two kinds of mutations are provided: one is a flip of arbitrary bits with a small probability and the other is a complete randomization of a full chromosome with a much smaller probability.

Goodness is calculated using ranks rather than with absolute fitness values. This makes the surface shallow in the beginning, which, in turn, reduces the evolutionary pressure and allows for more experimentation without killing off the individuals too fast. A Metropolis algorithm is used to select two individuals to mate using the goodness values. The two individuals to be eliminated are also selected through a Metropolis algorithm by using the badness values. This second Metropolis step is often dispensed within the standard genitor algorithms, choosing simply the two worst individuals to be eliminated.

There is a rank, 20% from the top, which is designated as *Caesar's wife*. No mutations are allowed above this rank, the individuals being considered too valuable to be unsettled. They can mate all

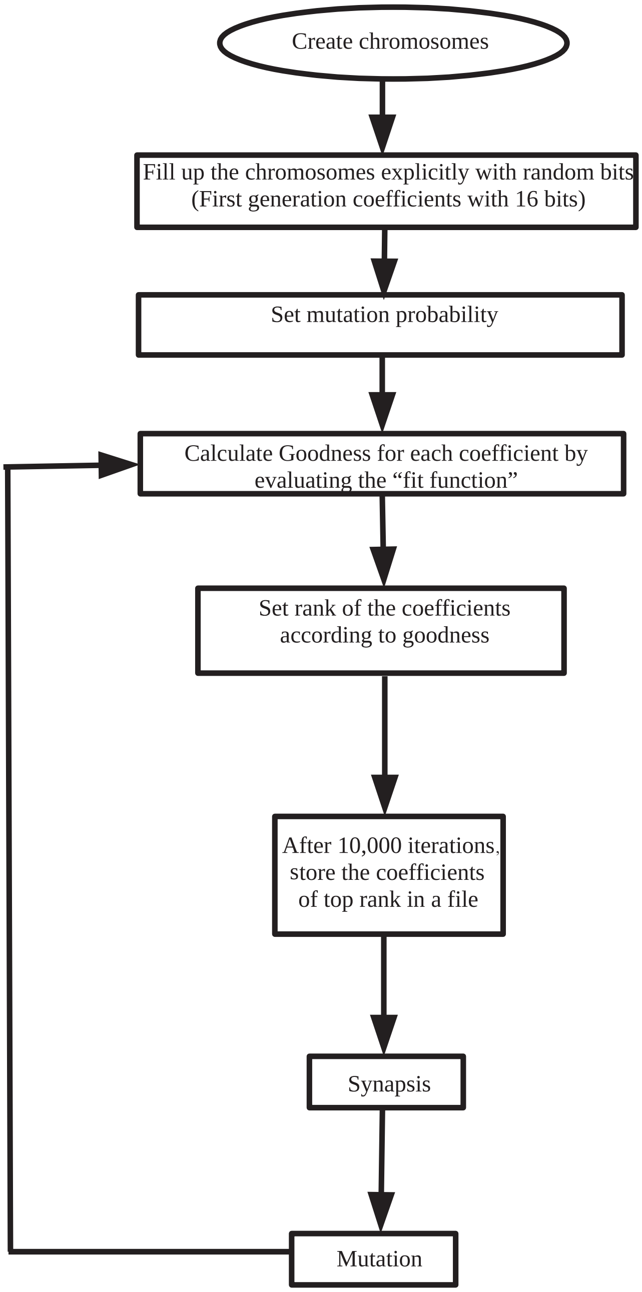
Create chromosomes
Fill up the chromosomes explicitly with random bits
(First generation coefficients with 16 bits)
Set mutation probability
Calculate Goodness for each coefficient by evaluating the "fit function"
Set rank of the coefficients according to goodness
After 10,000 iterations, store the coefficients of top rank in a file
Synapsis
Mutation

FIG. 4. Flow chart of the genetic algorithm program.

through the ranks. However, better individuals can always push up from below, relegating the previous good individuals to lower ranks.

In order to speed up the process, cluster computation using the Message Passing Interface (MPI) has been used, the goodness being calculated by helper programs running on other nodes. Figure 4 shows the flow chart of the genetic algorithm.

We have embedded the GA correction module in our system. In order to make it run hassle-free, we have introduced another *daemon* program, which is a correction module (**meterd**), which runs in the background and keeps a vigil on the whole process. The **meterd** program handles all the requests coming from the user (client) end. Depending on the availability of **avrd**, it transfers messages to **avrd** through MPI. The working principle of the correction module has been discussed with the help of a *data flow diagram* in Fig. 5, and the sequence of steps is as follows.

- Four of the channels of the voltage measuring station are connected (permanently) to four different standard voltages, V_S (0, 0.5, 1.0 V) and V_H (1.84 V).
- A *correction-module helper program* has been used to automatically trigger **meterd** to measure standard voltages. For this purpose, first, a higher voltage (V_H) is measured, and then, the value of the standard voltage (V_S) is recorded after a time interval (t_i), and this process is repeated for different t_i (1, 2, . . ., 20 s) for each V_S. For each case, the measured values are stored in a data file named "value.dat."
- A cluster of *compute goodness helper programs* reads the measured values from the data file and calculates the goodness of each set of coefficients (chromosomes) by using a **fit function**. The **fit function** chosen here is as follows:

$$V_{standard} = V_{measured}(t_1) \times A_0 + A_1 + A_2 \times \exp\left(-\frac{t_1 - t_0}{A_3}\right),$$

where $V_{standard}$ is the standard voltage, $V_{measured}(t_1)$ is the current measured voltage, A_0, A_1, A_2, and A_3 are the coefficients, t_1 is the current time of measurement, and t_2 is the previous time of measurement.

- The *compute goodness helper program* sends goodness of each coefficient to *compute goodness program,* which sets the rank of each coefficient and sends the rank to the *genetic main program.*
- The *genetic main program* writes the values of the coefficients corresponding to the top rank in a file "coefficient.dat."
- When any user (program) sends request to measure a voltage, the **meterd** program handles the request. It first measures the voltage through **avrd** and corrects the measured value by using the coefficients stored in the file "coefficient.dat." The corrected value is sent back to the user.

A *warm up* time is required for the system to train itself and to be ready for measurement. The GA correction module has been designed in such a way that it automatically updates the calibration coefficients by measuring the standard voltages in regular intervals.

To test the GA correction module, we have remeasured the previously measured standard voltage in the similar manner. Figure 6 shows the measured values and the exponential corrected values using the GA correction module. The total corrected value, which is the sum of the exponential part and the linear shift, is shown in Fig. 7.

We have tested the stability of measurement of the VMS by recording the data points of each of the standard voltages for 30 min. We have noted that the corrected measured values are accurate up to the fifth *decimal* place.

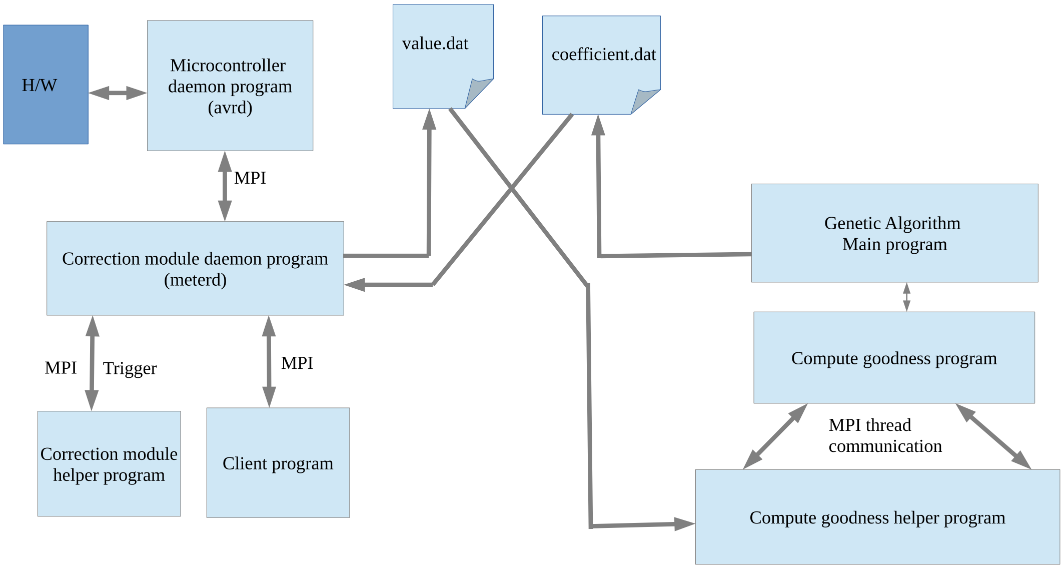
H/W
Microcontroller daemon program (avrd)
value.dat
coefficient.dat
MPI
Correction module daemon program (meterd)
Genetic Algorithm Main program
Compute goodness program
MPI
Trigger
MPI
MPI thread communication
Correction module helper program
Client program
Compute goodness helper program

FIG. 5. Data flow diagram of the correction module.

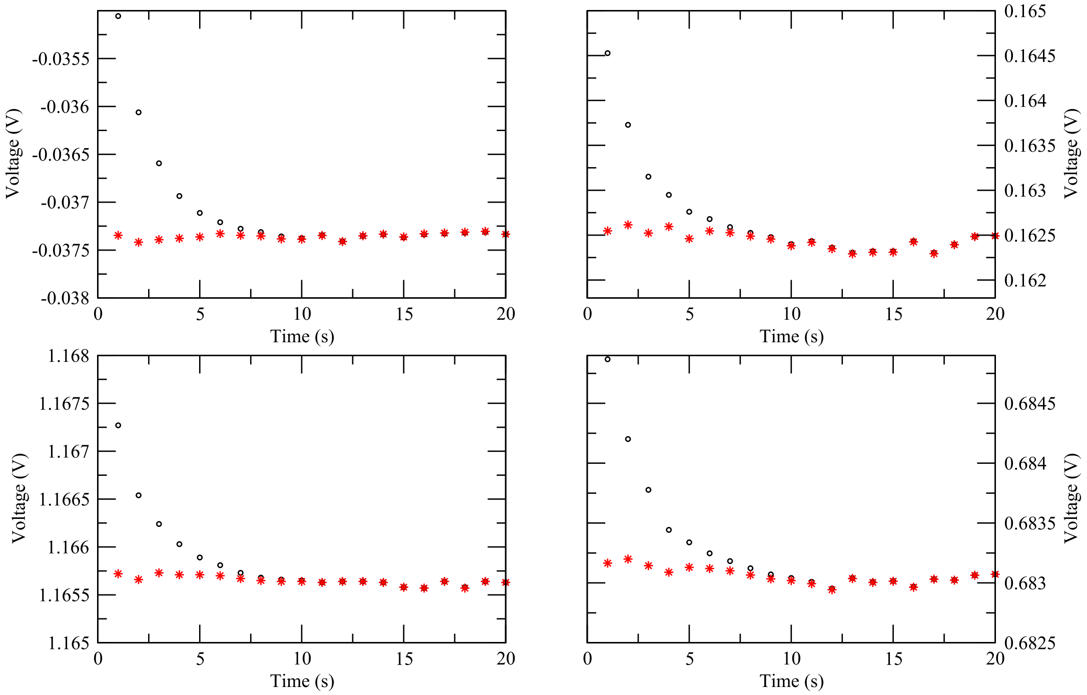

FIG. 6. Effect of the GA correction module on the exponential part of the measured values. For each set, the circles denote raw data points (without correction) and the stars denote exponential corrected data points. Top-left: 0 V. Top-right: 0.208 03 V. Bottom-left: 1.240 65 V. Bottom-right: 0.742 21 V.

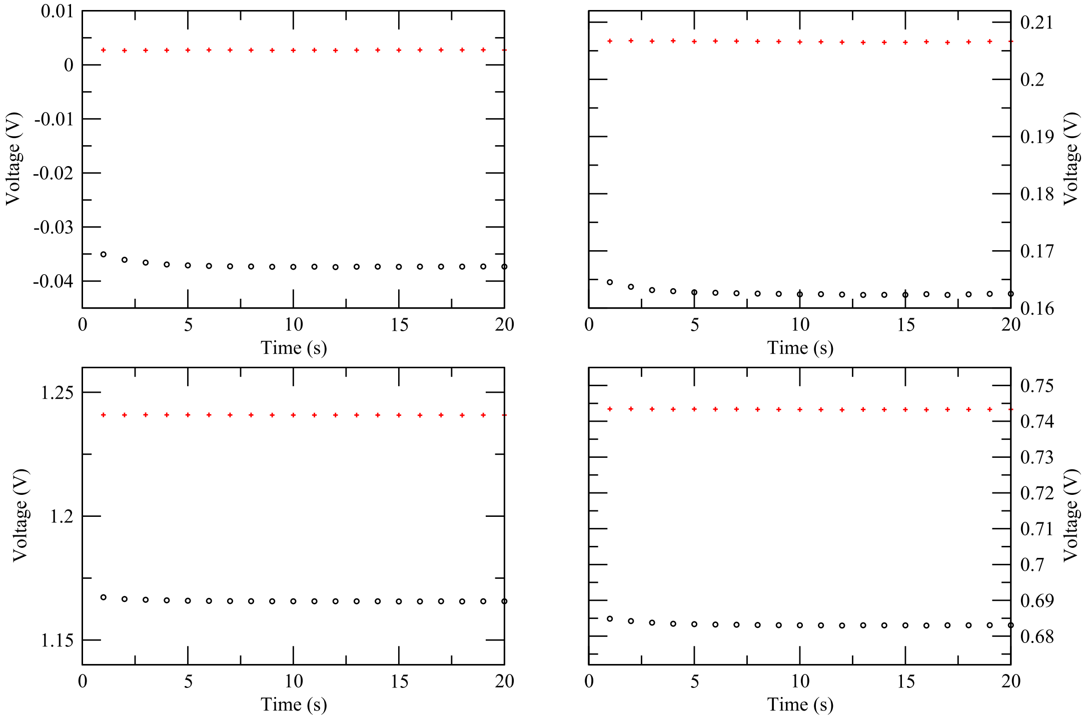

FIG. 7. Corrected values after using the GA correction module on the measured values. Here, both the exponential and linear corrections are considered. For each set, the circles denote raw data points (without correction) and the stars denote corrected data points. Top-left: 0 V. Top-right: 0.208 03 V. Bottom-left: 1.240 65 V. Bottom-right: 0.742 21 V.

IV. CONCLUSION

We have reported a GA-based correction module and tested it on a simple voltage measuring station. The correction module has been incorporated to correct the measured voltages that may be affected by temperature and humidity of the surroundings. Here, daemon programs have been used to access 16 channels independently without conflict of request between clients. The VMS can measure voltages between −2 and 2 V. The components of the dual-slope integrator of the VMS are chosen so that a count of $FFFF_H$ corresponds to 2 V and 0000_H to −2 V. The resolution of 1-in-65 535, which is indicated by the counter value of $FFFF_H$, can be achieved; however, we could attain a resolution of ~ $5\frac{1}{2}$ digits.

ACKNOWLEDGMENTS

We gratefully acknowledge Dr. Arani Chakravarti of the Department of Physics, Visva-Bharati, for his continuous support in developing the system.

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

REFERENCES

[1] R. Hölzel, P. M. Bentley, and P. Fouquet, "Instrument design and optimization using genetic algorithms," Rev. Sci. Instrum. **77**, 105107 (2006).

[2] H. Feng, S. De Santis, K. Baptiste, W. Huang, C. Tang, and D. Li, "Proposed design and optimization of a higher harmonic cavity for ALS-U," Rev. Sci. Instrum. **91**, 014712 (2020).

[3] B. Ozpineci, L. M. Tolbert, and J. N. Chiasson, "Harmonic optimization of multilevel converters using genetic algorithms," IEEE Power Electron. Lett. **3**(3), 92 (2005).

[4] P. C.-P. Chao and S.-C. Wu, "Optimal design of magnetic zooming mechanism used in cameras of mobile phones via genetic algorithm," IEEE Trans. Magn. **43**(6), 2579 (2007).

[5] P. Sprzeczak and R. Z. Morawski, "Calibration of a spectrometer using a genetic algorithm," IEEE Trans. Instrum. Meas. **49**(2), 449 (2000).

[6] C.-H. Kung, M. J. Devaney, C.-M. Huang, and C.-M. Kung, "Fuzzy-based adaptive digital power metering using a genetic algorithm," IEEE Trans. Instrum. Meas. **47**(1), 183 (1998).

[7] M. Pastorino, A. Massa, and S. Caorsi, "A microwave inverse scattering technique for image reconstruction based on a genetic algorithm," IEEE Trans. Instrum. Meas. **49**(3), 573 (2000).

[8] S. Osowski, R. Siroic, T. Markiewicz, and K. Siwek, "Application of support vector machine and genetic algorithm for improved blood cell recognition," IEEE Trans. Instrum. Meas. **58**(7), 2159 (2009).

[9] Y. Tan, Y. He, C. Cui, and G. Qiu, "A novel method for analog fault diagnosis based on neural networks and genetic algorithms," IEEE Trans. Instrum. Meas. **57**(11), 2631 (2008).

[10] S. Boggi and W. G. Fano, "Numerical response and causality study of the magnetic permeability-frequency function of NiZn ferrites using genetic algorithm," J. Magn. Magn. Mater. **500**, 166305 (2020).

[11] A. Bacci, V. Petrillo, and M. Rossetti Conti, "GIOTTO: A genetic code for demanding beam-dynamics optimizations," in Proceedings of the 7th International Particle Accelerator Conference (IPAC 2016), Busan, Korea, 8–13 2016.

[12] A. Hofler, B. Terzić, M. Kramer, A. Zvezdin, V. Morozov, Y. Roblin, F. Lin, and C. Jarvis, "Innovative applications of genetic algorithms to problems in accelerator physics," Phys. Rev. Spec. Top.–Accel. Beams **16**, 010101 (2013).

[13] S. Roy, S. Sil, and A. Chakravarti, "A voltmeter with browser-based control: An inexpensive instrument," Indian J. Phys. **84**(3), 301–307 (2010).

[14] L. Michaeli, J. Šaliga, J. Buša, P. Dolinský, and I. Andráš, "Influence of the capacitor's dielectric absorption on the dual slope ADC," in 21st IMEKO TC4 International Symposium and 19th International Workshop on ADC Modelling and Testing Understanding the World through Electrical and Electronic Measurement, Budapest, Hungary, 7–9 September 2016.

[15] R. L. Haupt, *Practical Genetic Algorithms* (Wiley-Interscience Publication, 1998).

[16] J. S. Arora, "Genetic algorithms for optimum design," *Introduction to Optimum Design*, 3rd ed. (Academic Press, 2012), Chap. 16, pp. 643–655.