Analysis of noisy transient signals based on Gaussian process regression

I. Baglaeva, B. Iaparov, I. Zahradník, A. Zahradníková

Department of Cellular Cardiology, Institute of Experimental Endocrinology, Biomedical Research Center, Slovak Academy of Sciences, Bratislava, Slovakia.

Short title

Analysis of noisy transient signals

Abstract

Transient signals convey information on the state and function of the studied system. The estimation of characteristic parameters of transient signals is complicated by the noise in the records. The reduction of noise by filtering leads to certain signal distortion. Alternatively, the transient signal can be approximated by a mathematical function, if available. We introduce here the model-independent approximation method for general noisy transient signals based on the Gaussian process regression. The method was implemented in the software TransientAnalyzer, which detects transients in a record, finds their best approximation by the Gaussian process, constructs a surrogate spline function, and estimates specified signal parameters. The method and software were tested on a model of the calcium concentration transient, characterized by abrupt rise and slow decay, contaminated by various signal-to-noise ratios, and recorded at a low sampling frequency. Statistics of the analyzed data set revealed that the software estimates all signal parameters reliably. The error of estimation and the standard deviation of estimates were less than 10% for SNR ≥ 7. The software and its description are available on GitHub.

Manuscript contribution to the field

Transient signals convey information on the state and function of the studied system. However, the estimation of their characteristic parameters is complicated by the noise present in the recordings. Methods used for noise reduction have various disadvantages, such as distortion of the time course by filtering, the necessity of exact superposition of many transients for successful averaging, or the need for a functional description for data fitting. In this work, we present a general method for automatic analysis of noisy transients based on Gaussian process regression, and its implementation in Python. The method can reliably analyze data at SNR > 5, and even at SNR = 2 if the stimulus time for the transient can be provided.

Keywords

Data analysis, transient signal, noise, nonlinear regression, software

Introduction

Transient signals regulate many processes in biology. Their study is important for the understanding of underlying mechanisms. These signals are characterized by specific parameters, for instance, the peak amplitude, the time to peak, the duration at half maximum, or the rise and decay times, which can be read from the recorded traces if the signal-to-noise ratio is sufficiently large and the characteristics of interest are not degraded by the excess noise. In reality, the recorded signals are contaminated by excess noise that has to be suppressed before data analysis.

Currently available software products for transient signal analysis have various limitations. The software either works only with specific equipment (1), requires intensive manual input (2), or is written for a proprietary platform (3,4). Moreover, most published algorithms use linear interpolation of recorded data points and thus may report erroneous parameter values in the case of noisy data. The signal-to-noise ratio (SNR) in records of transient signals is very often below 10, that is, much smaller than acceptable for analysis by simple algorithms.

Several noise-suppression techniques are recently available. The signal averaging method is applicable only if the transients are stationary and have well-defined starting times or activating stimuli. In the absence of averaging, the use of band-pass filters (5) or an approximation by a known function (6,7) is often used. Application of a filter to the data record may be of some help, but often at the cost of inherent degradation of the signal time course. The approximation of signals by a related function depends on the quality of the mathematical model. Less exact models may introduce systematic errors. On the other hand, the models of transient signals with a complex structure or analysis of signals recorded under unstable conditions may need corrections in parameter settings disqualifying them from the automatic analysis.

In general, the shape of the transient can be very complex and variable. For the approximation of a general transient, we considered the Gaussian Process Regression (GPR) method (8), originally developed to approximate a multidimensional data set obtained from measurements of complex structures and arising from random processes. In the approximation problem of a two-dimensional dataset, we turned to the GPR as a nonparametric regression model, in which the model structure of a transient is not predetermined, e.g., by logistic or polynomial functions, but is constructed as the probability distribution over the functions defined by the kernel and conditioned by the given data set (8). In this work, we describe and test an automatic algorithm and software for the analysis of individual noisy transient signals, based on nonparametric regression (GPR), which allows the user to approximate transients with a different steepness of the rising and the decaying part of their time course. We show the ability of the software to automatically detect calcium transients recorded at SNR > 5, and semi-automatically down to SNR of 2. The characteristic parameters of detected transients were estimated reliably even at SNR as low as 2. The error of estimation and the standard deviation of estimates were less than 10% for SNR ≥ 7. The source code and the executable file on GitHub(https://github.com/IuliiaBaglaeva/TransientAnalyzerUI/releases/tag/Release) listed in Zenodo (doi:10.5281/zenodo.7039242) are freely available.

Methods

The methods describe the detection of the transients in the record, approximation of individual transients based on Gaussian Process Regression (GPR), determination of parameters characterizing individual transients, and statistical analysis of the evaluation of detection results at a range of signal-to-noise ratios.

The software TransientAnalyzer was written in Python version 3.7. The inputs are NumPy (9) arrays of recorded time points and signal amplitudes. The TransientAnalyzer performs sequentially the detection of transients in the noisy records, the best approximation of individual transients with a smooth continuous function, and calculations of characteristic parameter values of the smoothed transients.

The software TransientAnalyzer is presented and deposited as a Graphical User Interface (GUI) application (https://github.com/IuliiaBaglaeva/TransientAnalyzerUI) and as a Python module for developers in GitHub (https://github.com/IuliiaBaglaeva/TransientAnalyzer) together with a listing on Zenodo with a Digital Object Identifier (DOI) 10.5281/zenodo.7039242 so that it is uniquely citable.

Detection of transients

The detection of a transient in the record is based on finding the rapid rise of the signal using the difference $D(t)$ between the recorded noisy original trace and the smoothed original trace calculated by a moving average (boxcar) filter at each time point (Figure 1). For the demonstrated example of calcium transients, the differences show a sharp maximum at the onset of a transient. The time between two maxima demarcates the duration of a transient, that is, the first point and the last point of a transient.

The correct number of detected transients and their starting points are influenced by the proper setting of the window size and the prominence parameters of the moving average filter. The window size specifies the number of averaged points and the prominence specifies the amplitude threshold of detected peaks relative to the surrounding baseline in the record (10). The improper prominence setting may influence the number of detected peaks, especially at low SNR. The improper window size may affect the correct start point estimate, depending on the kinetics of the transient and the sampling frequency. Thus both parameters of the filter should be set properly for the specific type of signals. The software allows finding the correct number of transients and their approximate start time values interactively. The interval between two consecutive start points determines the time interval to be approximated by the GPR. If the real start values are known, e.g., from the timing of the excitation stimuli, the automated detection may be overridden by supplying the file with the list of stimulus time values, which will demarcate the points of a transient used for approximation by the GPR. The stimuli time points can be used to estimate the delay between the stimulus and the onset of the transient.

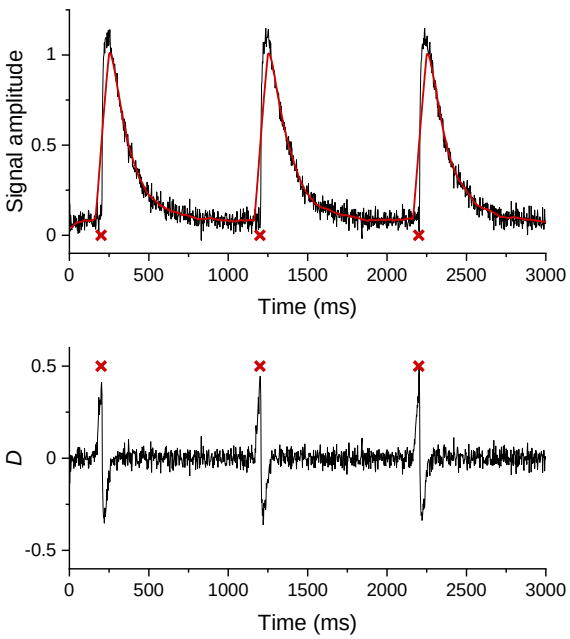

Figure 1. The detection of transients and determination of their start points. Top: The simulated record of transients with a signal-to-noise ratio of 10 (black), and its moving average-filtered trace (red) at a window size of 20 points and prominence of 1. Bottom: The difference (*D*) between the recorded and the filtered traces. Red crosses indicate the times of maximal difference and thus the approximate starting time.

Approximation of the transient

In our case, the data set was represented by *N* points of simulated calcium signal (t_i, y_i).

The Gaussian Process Regression (GPR) is a type of Bayesian Regression (8) with the prior distribution being a multivariate Gaussian distribution with an arbitrary mean (usually set to 0) and a total N-dimensional covariance matrix K with the elements

$$K_{ij} = K(t_i, t_j) = k(t_i, t_j) + \sigma^2 \delta_{ij}, \qquad \text{Eq. 1}$$

where $k(t_i, t_j)$ is the covariance matrix, σ is the noise amplitude and δ_{ij} is the Kronecker delta. In the Gaussian process, the covariance matrix plays a crucial role as it defines the relationship between the points and hence the behavior of the process behind the data. Since the process is not stationary and the analyzed transients may display a positive or negative skewness, the kernel was extended to consider the fast rise and slow decay of the signal. Mathematically, it is defined as a variant of the Gibbs kernel (11):

$$k(t_i, t_j) = A\sqrt{\frac{2 l_i l_j}{l_i^2 + l_j^2}} \exp\left(-\frac{(t_i - t_j)^2}{l_i^2 + l_j^2}\right), \quad l_i = l(t_i) = B t_i + C, \qquad \text{Eq. 2}$$

where the parameters *A*, *B*, and *C* define the amplitude and temporal properties of the kernel. The parameters σ (Eq. 1) and *A*, *B*, *C* (Eq. 2) are estimated by maximization of the log marginal likelihood ((8) Eq. 2.30) using the limited-memory Broyden, Fletcher, Goldfarb, and Shanno bound algorithm (12). The GPR model of the simulated transient was built using GPFlow, a Tensorflow-based Python library (13).

Because of the variable length scale of the kernel, the GPR tends to overfit the data (i.e., to follow the noise) right at the start of the transient (small values of *i* in Eq. 2), especially at small SNR. This problem was solved by the automatic detection of those data points among the first 8 datapoints of the transient that were below the preceding baseline level and their exclusion from the approximation process.

The characteristics of a transient

The GPR determines the best approximation of the transient at each time point of the original record. It should be noted that the time points may be sampled at irregular intervals. The time resolution of approximated transients can be increased by finding an appropriate continuous function. To this end, the GPR predictions are sampled at time intervals shorter than those of the input data. In the demonstrated cases, the GPR time interval was set to 100 µs while the recorded sampling period was 2.5 ms. For a further increase in resolution, a surrogate continuous cubic spline model based on the GPR predictions was used for the sake of computational efficiency. This allows the TransientAnalyzer to determine the values of characteristic parameters at increased amplitude and time resolution.

The start time of the transient origin, t_0, is determined as the intersection between the surrogate spline function of the Gaussian process approximating the analyzed transient, and the baseline estimated as the value of the Gaussian process (GP) describing the preceding transient at its endpoint. For the first transient in the record, the baseline is calculated as an average of the last 80 data points before its starting point. The intersection was found using the Gradient descent algorithm (14). The parameter evaluation is illustrated in Figure 2. The amplitude is defined as the peak value of the surrogate spline function describing the transient relative to the baseline. The rise time and the decay time are calculated as the time between the specified percentage values of the peak amplitude. The duration is determined as the difference between the times at which the transient signal is equal to a specified percentage of its peak amplitude (the default value is 50%, giving the full width at half maximum (FDHM)).

The kinetic parameters were determined using the bisection method (14), while the peak amplitude, t_0, and time-to-peak (TTP) were determined using the L-BFGS-B method (13).

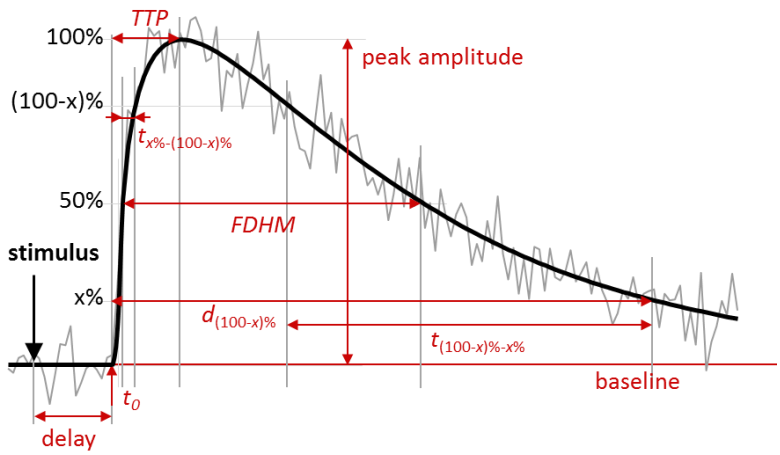

Figure 2. Characteristic parameters of a transient signal. The time dependence of the transient is characterized by specific parameters: the delay, the peak amplitude, the time to peak (TTP), the full duration at half-maximum (FDHM), the rise time ($t_{x\%-(100-x)\%}$), the decay time ($t_{(100-x)\%-x\%}$), and the transient duration ($d_{(100-x)\%}$) from the start to 100-*x*% of the peak amplitude. Noisy line - an example of the analyzed transient signal. Solid line - the surrogate spline function for the best GPR approximation of the transient.

Statistics

Statistical analysis was performed in Origin (OriginLab, Ver. 2022b). The significance of differences between the estimates and the true values of transient characteristics was determined using the one-sample t-test and $p = 0.05$ as a criterion.

Results and Discussion

Generally, the transients are likely to have various kinetics so that the rise and decay times can differ by an order of magnitude or more. In this case, any linear time-invariant (LTI) filter, for example, the Savitzky-Golay filter (5), fails to smooth the transient correctly. This may either overfit the signal and leave a significant noise level if the window size is small or cutoff frequency is high, or underfit and kinetically distort the fast dynamics of transients at large window sizes or small cutoff frequencies). In addition, LTI filters significantly shift the onset and the peak time of a transient (see Figure 1 as an example. Such signal distortions may be unacceptable for more precise measurements.

The GPR method with the commonly used stationary kernels such as Squared Exponential or Matern also fails to approximate the complex transients correctly, similarly to LTI filters, because of a common length scale for the whole signal. In the GPR we solved this problem by changing the length scale with time in the Gibbs kernel (the linear term in Eq. 2) and thus account for the dynamics of the transient. This kernel extension allows a better approximation of both, the short and the long time scale processes present in the signal.

The performance of the TransientAnalyzer was tested on simulated calcium transients generated by a model of human cardiomyocyte electrophysiology (15). The simulations were conducted in the OpenCOR software (16) using the CVODE numerical solver with a time step of 0.01 ms and recorded at a sampling frequency of 400 Hz. A sequence of 100 identical calcium transients, separated by 1 000 ms start-to-start, was generated and combined with Gaussian noise simulated using NumPy (9) for SNR in the range of 2 to 100. The noise level σ was defined as

$$\sigma = \frac{A}{3\,SNR}, \qquad \text{Eq. 3}$$

where *A* is the peak amplitude of the simulated transient relative to the baseline value.

The TransientAnalyzer could approximate the detected transients well even at very low SNR values (Figure 3); nevertheless, irregular artifacts could be present in the approximated traces, especially when the recorded signal was close to the baseline.

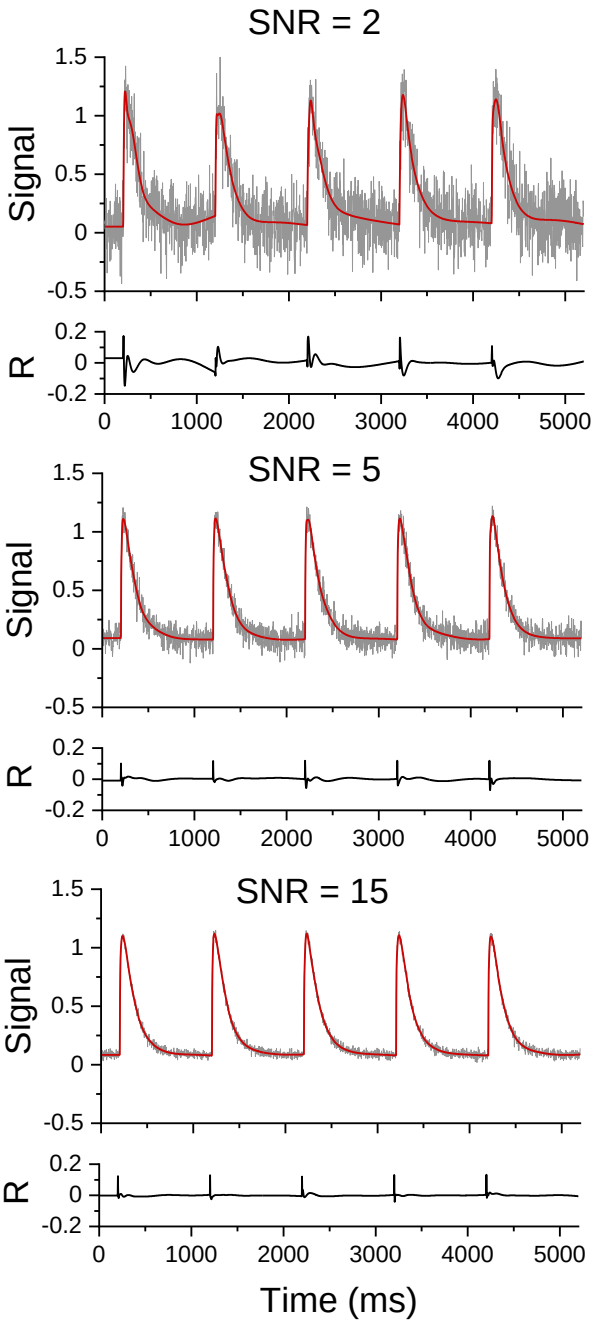

Figure 3. Approximations of transients by the Gaussian process regression. Signal (a.u.) - the examples of simulated noisy transient signals at the indicated SNR values with the best approximation curves (red lines). In the case of SNR=2, the time interval for the transient approximation was determined using the data file with stimulus times. R (a.u.) - residuals (the differences between the theoretical and the approximation functions).

At low values of the SNR (SNR < 5), the detection algorithm often identified false transient events even at optimized parameter settings of the filter. This problem could be avoided if the stimulus file was provided. For the tested signals, we used the start points of action potentials as the stimulus times for GPR analysis.

The parameter values estimated for individual transients were compared with their true values, calculated from the corresponding theoretical transients, by the statistics on 100 noisy transients for each SNR value (Figure 4). The estimates of individual parameters were very close to the true ones. The relative errors of the estimates were smaller than 10% at each SNR except for the delay and $t_{10\text{-}90\%}$ rise time at SNR < 7. In the cases of the delay, $t_{90\text{-}10\%}$, and $d_{90\%}$, the estimates differed by <1% from the true values but these differences were statistically significant ($p < 0.05$) even at high SNRs. Nevertheless, this difference was much below the temporal resolution of 2.5 ms. The standard

deviations were below 5% at all SNRs. Taken together, these results demonstrate the robustness of the GPR algorithm and its applicability for automatic parameter estimation of recorded transients even in the presence of high noise in the recorded signal.

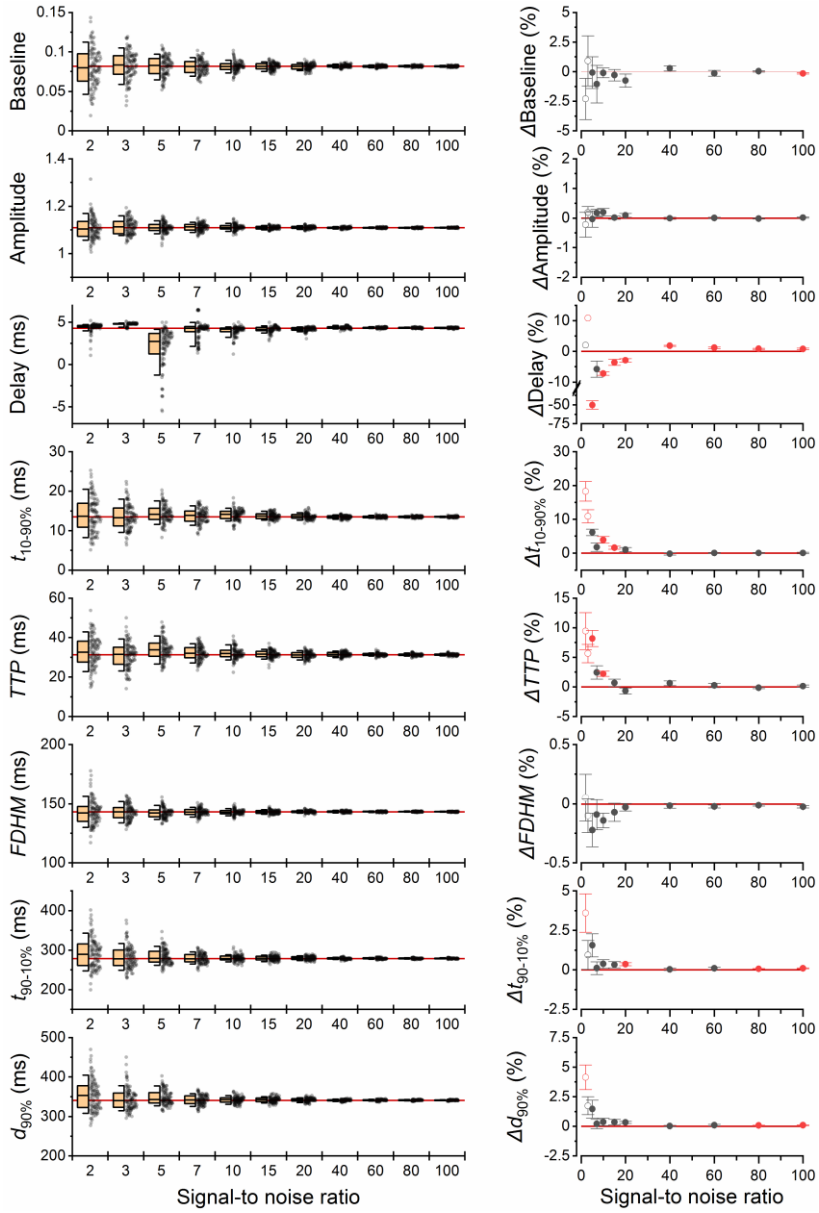

Figure 4. Statistical analysis of estimates of the characteristics of the transients by the GPR method. Left: Distribution of the parameter estimates for 100 individual transients at different SNR values. The parameter estimates (gray dots) are shown together with the half-box statistic (left to the data). The black line inside the half-box represents the median, whiskers show the 10% and 90% percentile, and the half-box shows the 25% and 75% percentile. Right: Relative errors of the parameter estimates (Δ%), calculated as the difference between the mean of estimates at the given SNR and the respective parameter value of the theoretical model. Mean relative errors significantly different from 0 at $p < 0.05$ (one-sample t-test) are indicated in red. The red lines indicate the parameter values of the theoretical model of the transient. For SNR < 5 (open circles), the analysis was performed only on transients detected using the stimulus file. At all other SNR values (filled circles), the results of statistics were independent of the transient detection method.

Capabilities

The TransientAnalyzer software provides a general solution to the common problem of the exact characterization of noisy transient signals. First, it detects the transients and finds their onset in a continuous record. Second, it approximates the detected transients one by one with a Gaussian process regression and creates the corresponding spline function. The output is a noise-free continuous function that allows an exact estimation of its characteristic parameters at a high amplitude and time resolution. The application is independent of the device and method of detection. As demonstrated, TransientAnalyzer reliably estimates the characteristic parameter values of transient changes in cellular calcium concentration. In the same way, it could be used to estimate parameters of cell shortening occurring in cultured or isolated myocytes, or various optical, electrical, mechanical, or chemical signals related to specific cell activities. The GPR-based approach is especially useful when the process that generates the transient is complex and not well understood. In such cases, reliable estimation of transient characteristics recorded in various experimental situations may help to discover the underlying mechanisms.

For convenience, the application is published at (https://github.com/IuliiaBaglaeva/TransientAnalyzerUI) as the Python package TransientAnalyzer, which can be installed by the generally available Package Installer for Python (17) using the command *pip install TransientAnalyzer*, and as a graphical user interface created using PyQt5 (18) and pyqtgraph (19). The core module of the TransientAnalyzer documented in (https://transientanalyzer.readthedocs.io/en/latest/) allows customization of the software by the user for a specific purpose.

Limitations

The detection part is based on the maximal difference between the recorded and filtered signals. It can be expected that large instances of noise could result in false detection of a transient or incorrect determination of its onset. On the other hand, small and slowly rising transients may be missed. This problem can be resolved by the trial-and-error method using the interactive design of the GUI. At SNR < 5, it was useful to specify the starting points of the transients using a stimulus file, to avoid instances of false transient detection.

It should be noted that the activation phase was the critical part of the tested signal. In the simulated transients with the rise time $t_{10\text{-}90\%}$ of 12.5 ms, the rising phase was sampled by only five data points. Despite this, the approximation software based on the GPR method performed robustly. Nevertheless, the high and low-frequency noise may cause substantial errors when the amplitude of the noise is comparable with that of the signal, as discussed in the Results section. From this point of view, the number of outliers was acceptable. The functionalities of TransientAnalyzer were not tested for SNR smaller than 2.

The GPR method is computationally intensive. The complexity caused by matrix inversion increases with the 3rd power of the number of data points. This may result in a somewhat prolonged analysis in the case of a large number of transients per record. For example, the 100 simulated calcium transients analyzed in the Results section (400 data points per transient) took approximately 3 minutes on a PC with 4 core 8 threads Intel® Core™ i7-4790. However, the GPR module uses GPFlow based on TensorFlow technology, which may accelerate substantially the learning and prediction of GPR by the use of a dedicated GPU. Alternatively, downsampling of records or omission of the steady part of the baseline can be used if individual transients contain more than 500 points to speed up calculations on less powerful computers.

Conclusions

The TransientAnalyzer is a self-standing computational tool designed for the automatic analysis of transient signals in records with a high noise level that provides reliable estimates of the characteristic parameters of detected transients. The software TransientAnalyzer including the core module is documented to allow its extension and customization for specific applications by a knowledgeable user of the Python system.

Contributions

IB, AZ, IZ, and BI designed research; IB and BI performed research; IB and BI contributed analytic tools; IB and AZ analyzed data; IB, BI, AZ, and IZ wrote the paper.

Declaration of Ineterest

The authors declare no competing interests.

Funding

The research was supported by the SAV-TUBITAK project JRP/2019/836/RyRinHeart, by the project VEGA 2/0182/21, the Slovak Research and Development Agency project APVV-21-0443, and by the Operational Programme Integrated Infrastructure for the project: Long-term strategic research of prevention, intervention, and mechanisms of obesity and its comorbidities, IMTS: 313011V344, co-financed by the European Regional Development Fund.

References

1. Ionoptix L. L. C. https://www.ionoptix.com/company/
2. Molecular Devices L. L. C. https://www.moleculardevices.com/products/axon-patch-clamp-system/acquisition-and-analysis-software/pclamp-software-suite
3. Janicek, R., M. Hotka, A. Zahradníková, A. Zahradníková, and I. Zahradník. 2013. Quantitative Analysis of Calcium Spikes in Noisy Fluorescent Background. *PLoS ONE*. 8(5):e64394-e64394 (https://doi.org/10.1371/journal.pone.0064394).
4. Psaras, Y., F. Margara, M. Cicconet, A. J. Sparrow, G. G. Repetti, M. Schmid, V. Steeples, J. A. L. Wilcox, A. Bueno-Orovio, C. S. Redwood, H. C. Watkins, P. Robinson, B. Rodriguez, J. G. Seidman, C. E. Seidman, and C. N. Toepfer. 2021. CalTrack: High-Throughput Automated Calcium Transient Analysis in Cardiomyocytes. *Circulation Research*. 129(2):326–341 (https://doi.org/10.1161/circresaha.121.318868).
5. Orfanidis, S. J. 1995. Introduction to signal processing. Prentice-Hall, Inc.
6. Lacampagne, A., C. W. Ward, M. G. Klein, and M. F. Schneider. 1999. Time Course of Individual Ca2+ Sparks in Frog Skeletal Muscle Recorded at High Time Resolution. *Journal of General Physiology*. 113(2):187–198-187–198 (https://doi.org/10.1085/jgp.113.2.187).
7. Zahradníková, A., E. Poláková, I. Zahradník, and A. Zahradníková. 2007. Kinetics of calcium spikes in rat cardiac myocytes. *The Journal of Physiology*. 578(3):677–691-677–691 (https://doi.org/10.1113/jphysiol.2006.117796).
8. Rasmussen, C. E. 2006. Gaussian processes for machine learning. MIT Press.
9. Harris, C. R., K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. 2020. Array programming with NumPy. *Nature*. 585(7825):357–362-357–362 (https://doi.org/10.1038/s41586-020-2649-2).

10. SciPy API Reference. https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.peak_prominences.html
11. Gibbs, M. N. (1998). Bayesian Gaussian processes for regression and classification. University of Cambridge.
12. Zhu, C., R. H. Byrd, P. Lu, and J. Nocedal. 1997. Algorithm 778: L-BFGS-B. *ACM Transactions on Mathematical Software*. 23(4):550–560-550–560 (https://doi.org/10.1145/279232.279236).
13. de G. Matthews, A. G., M. van der Wilk, T. Nickson, K. Fujii, A. Boukouvalas, P. León-Villagrá, Z. Ghahramani, and J. Hensman. 2017. GPflow: A Gaussian Process Library using TensorFlow. *Journal of Machine Learning Research*. 18(40):1–6-1–6, http://jmlr.org/papers/v18/16-537.html.
14. Boyd, S., S. P. Boyd, and L. Vandenberghe. 2004. Convex optimization. Cambridge university press.
15. Tomek, J., A. Bueno-Orovio, E. Passini, X. Zhou, A. Minchole, O. Britton, C. Bartolucci, S. Severi, A. Shrier, L. Virag, A. Varro, and B. Rodriguez. 2019. Development, calibration, and validation of a novel human ventricular myocyte model in health, disease, and drug block. *eLife*. 8 (https://doi.org/10.7554/elife.48890).
16. Garny, A., and P. J. Hunter. 2015. OpenCOR: a modular and interoperable approach to computational biology. *Frontiers in Physiology*. 6 (https://doi.org/10.3389/fphys.2015.00026).
17. Python Software Foundation. https://pypi.org/
18. Riverbank Computing Limited. https://www.riverbankcomputing.com/software/pyqt/
19. PyQtGraph. https://www.pyqtgraph.org/