

Article

Not peer-reviewed version

---

# ION-Sim: A Novel Open-Source Simulation Framework for Intraoperative Neurophysiological Monitoring

---

[Rosmary Blanco](#) \* and [Riccardo Budai](#)

Posted Date: 16 April 2026

doi: 10.20944/preprints202604.1082.v1

Keywords: medical education; simulation; intraoperative neurophysiological monitoring (IONM); computational neuroscience; somatosensory evoked potentials (SSEP); Motor evoked potentials (MEP); visual evoked potential (VEP); compound action potential (CAP); cortico-cortical evoked potentials (CCEP)



Preprints.org is a free multidisciplinary platform providing preprint service that is dedicated to making early versions of research outputs permanently available and citable. Preprints posted at Preprints.org appear in Web of Science, Crossref, Google Scholar, Scilit, Europe PMC.

Copyright: This open access article is published under a [Creative Commons CC BY 4.0 license](#), which permit the free download, distribution, and reuse, provided that the author and preprint are cited in any reuse.

Disclaimer/Publisher's Note: The statements, opinions, and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions, or products referred to in the content.

Article

# ION-Sim: A Novel Open-Source Simulation Framework for Intraoperative Neurophysiological Monitoring

Rosmary Blanco <sup>1</sup> and Riccardo Budai <sup>2,\*</sup>

<sup>1</sup> Sano Centre for Computational Medicine - Krakow, PL / Amsterdam UMC location University of Amsterdam, Department of Radiology and Nuclear Medicine / Biomedical Engineering and Physics

<sup>2</sup> University Hospital S.M. della Misericordia - Udine, IT

\* Correspondence: riccardo.budai@gmail.com

## Abstract

The educational pathway for expertise in Intraoperative Neurophysiological Monitoring (IONM) is complex and lengthy, requiring a solid foundation in neuroscience, neurophysiology, and neuroanatomy. It also demands direct familiarity with a broad range of neurosurgical scenarios, including supratentorial, infratentorial, and spinal procedures, gained through exposure to at least ten distinct surgical approaches. Intraoperative neurophysiology must be tailored to each patient's preoperative assessments. It relies on a variety of methods to collect, analyse, and report neurophysiological signals that are relevant to the surgical procedure. Despite its importance, there remains a substantial shortage of training tools designed to support realistic practice and skill development. To address this gap, we developed a comprehensive framework (ION-Sim) that integrates all laboratory testing modalities and adapts them to the operating room environment. ION\_sim supports the simulation and analysis of spontaneous EEG and EMG activity, a wide range of evoked potentials, and intraoperative stimulus-response testing protocols. The framework provides a unified environment for practising, testing, and validating the core neurophysiological procedures employed during neurosurgical interventions. In addition, it incorporates a robust data-management architecture, maintaining a database with system setups, user profiles, educational performance metrics, and automatically generating reports. This structure enables the longitudinal tracking of objective skill acquisition and facilitates standardised assessments of trainee progress. ION\_Sim is distributed both as a ready-to-use application, suitable for direct integration into teaching and training programmes, and as a modular scientific library. Through its dedicated APIs, users can design customised configurations, create novel simulation scenarios, and extend the platform to support additional research or educational objectives. It is available upon request for educational purposes, open-source and released under the GNU General Public License, ensuring transparency, reproducibility, and long-term accessibility for the scientific and clinical communities.

**Keywords:** medical education; simulation; intraoperative neurophysiological monitoring (IONM); computational neuroscience; somatosensory evoked potentials (SSEP); Motor evoked potentials (MEP); visual evoked potential (VEP); compound action potential (CAP); cortico-cortical evoked potentials (CCEP)

---

## 1. Introduction

Training in clinical neurophysiology is inherently complex, necessitating the acquisition of practical skills through the execution of specific tests in clinical settings. Traditionally, this proficiency is attained during supervised rotations at accredited training centers, such as those affiliated with the Italian Society of Clinical Neurophysiology and its dedicated Intraoperative Neurophysiology section. However, direct learning in the operating room is not always feasible due to organisational

constraints; furthermore, it occurs in a high-pressure environment that carries inherent risks. The learning process is further challenged by the heterogeneity of clinical scenarios. Although standardized monitoring protocols exist, patient-specific adaptations frequently render the educational experience unpredictable. Mastering this field requires a comprehensive understanding of neurophysiology, pharmacology—particularly anesthesiology—and cranial and spinal surgical techniques.

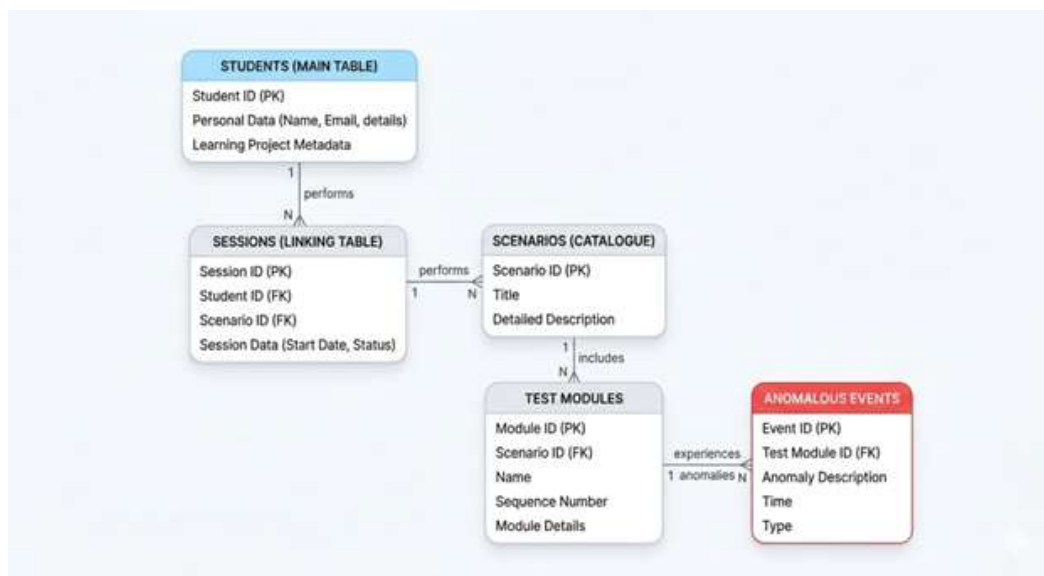
Currently, intraoperative monitoring simulations are largely restricted to proprietary tools bundled with specific commercial equipment. While a substantial body of literature [1–3] details operating room methodologies, these theoretical resources lack practical interactivity. Moreover, existing open-source and virtual reality simulators [Appendix.B] focus almost exclusively on surgical techniques, omitting neurophysiological integration. To address these educational gaps, this project introduces interactive simulations encompassing both individual neurophysiological tests and comprehensive operative scenarios. Specifically, in this paper, we present ION-Sim: a novel, open-source, and freely accessible framework designed to simulate Intraoperative Neurophysiological Monitoring (IONM) signals and clinical scenarios for educational purposes. Herein, we detail its architecture and functionalities, demonstrating its practical utility through representative case studies.

## 2. Materials and Methods

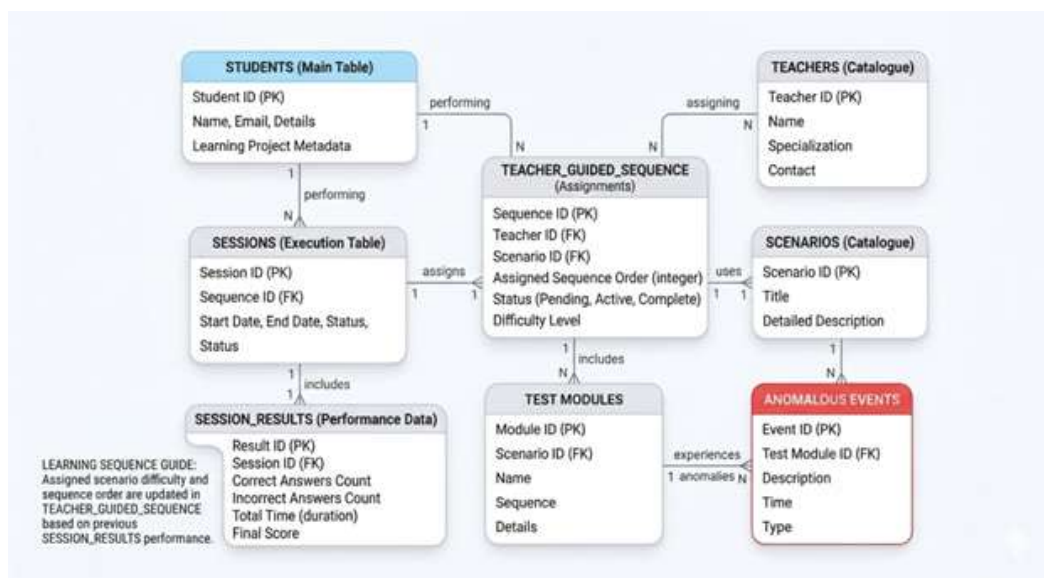
The architecture of ION-Sim is centered around a core database module designed to manage user authentication, track learning and work sessions, and archive performance metrics collected during simulations. Upon authentication, the system records learning outcomes—specifically, user responses to interactive queries—and correlates them with the active session. Ultimately, a comprehensive report is generated to consolidate all data derived from the specific simulation tests.

The pedagogical workflow begins with interactive tutorials that explain the features of the selected module, followed by active engagement with the simulation itself. Users can choose between a self-directed, in-depth study of individual modules or a guided learning pathway. The self-directed approach allows students to explore all available neurophysiological investigation methodologies, which serve as “baseline” data for surgical interventions and are essential for navigating complex simulation scenarios. Conversely, the guided learning pathway progresses from a foundational understanding of isolated modules to their integration within specific intraoperative scenarios. For example, a scenario for a supratentorial intervention near motor areas integrates multiple modules: electroencephalography (EEG), electrocorticography (ECoG), motor evoked potentials (MEPs) for direct cortical stimulation, somatosensory evoked potentials (SEPs) for central sulcus localisation, and free-running electromyography (EMG).

As illustrated in the database schema (Figure 1), predefined scenarios specify the types of tests to be performed and include concise clinical descriptions. Each scenario is associated with multiple test modules, within which specific conditions—defined as “anomalies”—are implemented. During the initial learning phase, the system automatically selects and sequentially presents these anomalies with equal weighting. For targeted review, users can initiate new sessions focused on specific anomaly types. The framework supports progressive learning by unlocking increasingly complex scenarios and anomalies over time. Users who achieve an “Advanced” designation can fully personalise their learning trajectory. Additionally, the system includes a Tutor role, enabling educators to design new test conditions and custom anomalies based on real-world clinical cases (Figure 2).



**Figure 1.** Database Structure: Students (STUDENTS) participate in multiple SESSIONS, each linked to a predefined SCENARIO from the central catalogue. Each scenario contains one or more TEST MODULES, with ANOMALOUS EVENTS recorded per module as discrete or recurring patterns.



**Figure 2.** Teacher Role. The associative entity TEACHER\_GUIDED\_SEQUENCE connects students, teachers, and scenarios, tracking sequence status and difficulty. Teachers (TEACHERS) assign sequences to students and monitor performance across sessions. They can also create or customize scenarios by selecting test modules and associated anomalies, enabling tailored learning paths for specific organisational or specialisation needs.

### 3. System architecture

Building on this framework, the simulation system is designed around a centralised database repository (Figure 1). Functionally, it comprises a core Data Abstraction Layer (DAL) for database interaction (Figure 2), a suite of specialised simulation modules (Appendix A)—each corresponding to a distinct physiological model—and a Learning Manager designed to orchestrate the pedagogical progression.

Each simulation module follows a standard architectural pattern for interface and workflow, while integrating sub-components tailored to its specific neurophysiological scenario. Upon selection from the database, simulation instances initialise automatically. A global control mechanism

facilitates the dynamic injection of “anomalies” (pre-configured fault conditions retrieved directly from the repository, Figure 1).

Users can operate in two modes: autonomous, allowing full control over all functions, or guided, managed by the Learning Manager. In guided mode, users diagnose presented anomalies by selecting the appropriate resolution from a set of alternatives displayed via the Graphical User Interface (GUI) (Figure 3). To support theoretical understanding, each module is coupled with a comprehensive knowledge base, accessible at all times through the help function, containing extended documentation on the simulation model and related neurophysiological concepts.

### 3.1. Modules

The primary objective of the simulation framework is to facilitate experiential learning and validate user competency within domains that require extensive practice in complex, real-world operational scenarios. The framework relies on a central database that orchestrates the entire workflow and interacts with specialized modules for simulation execution, anomaly injection, response verification, and reporting.

Upon initialisation with a baseline knowledge base, users are assigned one of three roles: *Entry*, *Advanced*, and *Tutor*. Progression from *Entry* to *Advanced* is contingent on performance metrics achieved during the initial training phase, thereby granting the user access to more sophisticated simulation functionalities. The *Tutor* role, assigned administratively, provides full privileges to modify pre-configured base scenarios or design novel ones including defining specific scenario characteristics, injecting associated anomalies, and configuring the corresponding correct answers and distractors (Figure 2).

Finally, the framework is released as an open-source scientific library hosted on GitHub. It exposes a robust set of Application Programming Interfaces (APIs) designed for seamless integration within Python or C++ development environments. The ION-Sim system home screen is shown in Figure 3.

### 3.2. Physiological Modules

The Physiological Modules suite comprises multiple components, each simulating specific neurophysiological activation patterns and monitoring modalities. Each module integrates an anomaly management system and a dedicated Help interface detailing specific intraoperative monitoring (IOM) procedures. These include:

- CAP (Compound Action Potential) Simulation Module: simulates the stimulation of the afferent peripheral nerve trunk, with sensory fibre responses recorded at various distances from the stimulation point (e.g., wrist, elbow, axilla, Erb's point) (Figure 9)
- CMAP (Compound Motor Action Potential) Simulation Module: simulates the stimulation of the efferent peripheral nerve trunks, recording muscle responses across different segments and distances.
- SEP (Somatosensory Evoked Potential) Simulation Module: involves peripheral nerve stimulation with recording montages at Erb's point, the CV6 level, and scalp electrodes over the contralateral S1 and M1 sites. Key controllable parameters include the Stimulation Side (Right, Left), Block Average (sweep count), Overlay Percentage, Noise Amplitude ( $\mu\text{V}$ ), and Gain Control (Figure 7)
- VEP (Visual Evoked Potential) Simulation Module: simulates retinal flash stimulation with scalp recording over ipsilateral and contralateral occipital sites. Key controllable parameters include the Stimulation Mode (Right, Left, Bilateral), the Noise Level and the number of sweeps per block (update) (Figure 8)
- BAEP (Brainstem Auditory Evoked Potential) Simulation Module: utilizes click-based acoustic stimulation with contralateral masking noise, recorded via scalp electrodes using a bilateral mastoid montage. The module features automatic identification of latency and amplitude peaks for principal wave components (Waves I, III, V). Configurable parameters include: Stimulation

Side (Right, Left, Alternating), Stimulus Polarity (Rarefaction, Condensation, Alternating), Stimulus Frequency (Hz), Block Average (sweep count), Overlay Percentage, Noise Amplitude ( $\mu\text{V}$ ), and Gain Control. Figure 6

- MEPcb (Motor Evoked Potential - Cranio-Bulbar) Simulation Module: simulates transcranial scalp stimulation using a pulse train, recording from muscles innervated by cranial nerves (facial, trigeminal, glossopharyngeal, hypoglossal, spinal accessory). It supports up to 8 channels for CMAP recording and features a raster plot to track response variations over time, offering automatic marker localization for peak latency and amplitude. User-adjustable parameters include stimulation pulse count, amplitude (%), inter-stimulus interval, and noise levels. Both stack plot and raster display scales are fully customizable Figure 4
- MEPas (Motor Evoked Potential - Upper Limbs) Simulation Module: models muscle activation in the upper limbs following transcranial electrical pulse train stimulation, specifically targeting the 1st DI (first dorsal interosseous), ECD (extensor communis digitorum), and BB (biceps brachii) muscles. This module inherits the control interface and parameter definitions of the MEPcb module, presenting anomalies governed by predefined configuration schemas Figure 4
- MEPai (Motor Evoked Potential - Lower Limbs) Simulation Module: models muscle activation in the lower limbs following transcranial electrical pulse train stimulation, targeting the FHB (flexor hallucis brevis), TA (tibialis anterior), and VL (vastus lateralis) muscles. Similar to the upper limb module, it utilizes the standard MEP control interface and presents anomalies based on predefined configuration schemas Figure 4
- D-Wave (Direct Wave) Simulation Module: simulates corticospinal tract activation recorded directly from the spinal cord. While stimulation remains transcranial, recording is performed at two distinct spinal sites: proximal and distal to the surgical intervention level. The module incorporates standard stimulation controls and features a specialized “anomaly detection” interface, which manages introduced faults and queries the user for diagnostic responses Figure 5
- DCS (Direct Cortical Stimulation) Simulation Module: models diverse responses secondary to direct cortical stimulation, including motor responses, disruption of language function (speech arrest/aphasia), and alterations in cognitive processing (in progress).
- CCEP (Cortico-Cortical Evoked Potentials) Simulation Module: simulates cortical or subcortical stimulation with the recording of evoked responses at specific cortical distances, aimed at mapping the structural connectivity of tracts and cortical areas (in progress).
- EEG (Electroencephalography) Simulation Module: displays resting electrical activity and physiological variations associated with eyes open/closed states and motor activity (e.g., mu-rhythm desynchronization) and incorporates physiological cyclical variations derived from real-world clinical recordings. Figure 10
- ECoG (Electrocorticography) Simulation Module: simulates spontaneous corticographic activity, incorporating physiological cyclical variations derived from real-world clinical recordings Figure 11
- EMG (Electromyography) Simulation Module: models spontaneous electromyographic activity, including fluctuations and voluntary contraction phenomena, typical of awake patient procedures (e.g., awake craniotomy) and different patterns of spontaneous discharges Figure 12
- ANESTHESIA Simulation Module: displays patient metadata (age, sex, weight, height, BMI, LBM), monitors vital signs ( $\text{SpO}_2$ , HR, R-R interval, SDNN, RMSSD, NIBP, MAP, temperature, RPM), and manages drug infusion via target-controlled infusion (TCI) for Sevoflurane, Propofol, Remifentanyl, and Ketamine Figure 13.

Each module implements a comprehensive mathematical and biophysical model governing signal generation at the peripheral, spinal, cortical, and muscular levels. Furthermore, this framework accounts for signal propagation dynamics along afferent and efferent pathways, alongside the simulation of cellular activation within specific neuronal nuclei of the spinal cord, thalamus, and corresponding cortical areas.

### 3.3. Optimisation of Evoked Response Updates

A critical performance metric in intraoperative monitoring is the temporal resolution required to acquire a statistically valid response. The system must minimise the latency between a physiological event and its visualisation to promptly assess whether the signal remains aligned with the baseline or has deviated significantly in amplitude or latency.

To accelerate update rates—particularly for low-amplitude afferent evoked potentials (e.g., SEP, BAEP, VEP) where the signal-to-noise ratio (SNR) is inherently low—the system employs a weighted moving average (or overlapping technique) Figure 7.

**Signal Averaging Principles:** The identification of a variation within a biological signal requires the summation of a sufficiently large ensemble of homogeneous responses (sweeps). This process relies on the principle that time-locked physiological signals are additive, whereas stochastic background noise (such as EEG or EMG artifacts) attenuates by a factor proportional to the square root of the number of trials.

While employing a weighted moving average (WMA) ensures a more rapid refresh rate for the visualized waveform, it concomitantly imposes a systemic limitation known as the ‘smoothing effect.’ By averaging incoming raw sweeps—which might reflect a newly onset acute anomaly—against a significant percentage of pre-existing data, the system inherently dampens the magnitude of sudden physiological fluctuations. As a result, the complete visualization of an abrupt signal attenuation is hindered by a mathematical lag. The true extent of the pathological event becomes fully apparent only when the continuous influx of anomalous sweeps gradually overrides the weight of the ‘healthy’ sweeps lingering in the computational buffer.

### 3.4. Simulation of Spontaneous “Free-Running” Activity

The generation of spontaneous “free-running” bioelectrical activity—encompassing electroencephalography (EEG), electrocorticography (ECoG), and electromyography (EMG)—relies on the reproduction of empirical datasets derived from validated clinical recordings Figure 10 and Figure 11. These signals are rendered using a seamless cyclic buffering technique to ensure continuous, artifact-free monitoring streams. Modality-specific anomalies are dynamically superimposed onto this baseline activity. The injection of these pathological patterns adheres to the same temporal logic and triggering criteria (e.g., onset latency, stochastic recurrence) previously defined for evoked responses.

### 3.5. Anomaly Simulation Module

The training progresses to the systematic identification of anomalies embedded within specific simulation contexts. Structurally, the Anomaly Simulation Module comprises discrete components designed to inject targeted distortions into the baseline signals generated by the Physiological Simulation Module, thereby producing pathological output waveforms. Here, the student’s competencies are empirically evaluated based on their learned skills for detecting and interpreting diverse categories of simulated signal perturbations.

The system database provides these scenarios—ranging from single modalities to complex multimodal combinations—within the Scenario (catalog) table Figure 2, which currently defines 18 distinct configurations. The architecture supports the development of additional experimental setups in the Scenario (catalog) table, however new integrations require rigorous validation to prevent stimulus artifact overlap. The system enforces strict temporal management regarding multimodal evoked responses, ensuring that the epoch of a new stimulus does not encroach upon the acquisition window currently occupied by another active stimulation modality.

Anomalies available for simulation are indexed within the Scenario anomaly table and mapped to specific pre-configured scenarios. The injection of these anomalies is governed by configurable parameters, including onset latency and recurrence patterns, which may be defined as singular events or periodic occurrences with fixed or randomised intervals.

The simulated anomalies are classified according to the following taxonomy:

**Amplitude Anomalies:** Percentage decrements in signal magnitude that approach or exceed established clinical warning and alarm thresholds (e.g., a 50% reduction from baseline).

**Latency Anomalies:** Prolongation of signal latency relative to baseline, indicative of slowed neural conduction velocity, reaching or surpassing predefined temporal warning limits (e.g., a 10% increase).

**Signal Quality Anomalies:** Contamination by stochastic or deterministic noise arising from diverse perioperative sources. These include electrocautery interference, suction artifacts, stimulus artifact spillover, surgical manoeuvres, and mechanical disturbances (e.g., high-speed drills or irrigation systems).

**Signal Morphology Anomalies:** Qualitative alterations in waveform configuration and spectral content. Key features include temporal dispersion, increased polyphasia, and loss of component distinctness, typically reflecting desynchronized neural volley activation.

**Technical Anomalies:** Equipment-related failures, such as recording electrode dysfunction (e.g., dislodgement or high impedance) or stimulation system faults affecting one or more modalities.

**Systemic Anomalies:** Global physiological fluctuations affecting signal integrity, induced by pharmacological agents (e.g., bolus anaesthetic administration), hemodynamic instability (e.g., hypotension), or acute oxygen desaturation.

### 3.6. Learning Module

The Learning Module orchestrates the injection of anomalies into the physiological simulators and manages the subsequent anomaly interpretation tasks. This assessment is facilitated through a dedicated interface utilising a multiple-choice paradigm, consistently presenting one correct response alongside four distractors. To comprehensively assess user proficiency, the module evaluates several key performance indicators:

**Response latency:** The time elapsed between the onset of the anomaly and the user's initial action.

**Diagnostic accuracy:** The correctness of the selected response.

**Attempt frequency:** The number of iterations required to identify the correct solution.

**Resource utilisation:** The time spent consulting the integrated "help" documentation.

Furthermore, the Learning Module features an adaptive difficulty mechanism that progressively scales task complexity based on the user's accuracy and response immediacy. The module continuously evaluates operational behavior to foster sustained engagement and optimize the learning trajectory. Upon session completion, the framework generates a comprehensive analytical report and a final report of the user's learning progress.

## 4. Software Implementation and Availability

The simulation framework is implemented in Python 3.12, chosen for its extensive ecosystem of scientific computing libraries and its robust cross-platform compatibility.

**Technology Stack:** The architecture leverages a modular stack of specialized libraries.

**Computational Backend:** NumPy and SciPy constitute the core engine for signal processing, matrix operations, and the mathematical modeling of physiological responses.

**Graphical User Interface (GUI):** The frontend is developed using PySide6 (the official Python binding for the Qt framework), ensuring a responsive and native look-and-feel across different operating systems.

**Real-Time Visualization:** High-performance signal rendering—essential for simulating continuous EEG/EMG streams and evoked potentials—is managed by pyqtgraph, which is optimized for fast data plotting.

**User Interface Design:** the UI is engineered to streamline user interactions by abstracting the underlying database complexities. By automating session management and pre-loading scenario

configurations, the system minimises the setup burden, allowing the user's focus to the educational task.

**Licensing and Availability:** to foster collaboration and accessibility within the scientific community, the source code is released under the GNU General Public License v3 (GPLv3). The complete repository, including documentation and installation instructions, is publicly hosted on GitHub [insert link here].

#### 4. Discussion

The landscape of simulation tools for Intraoperative Neuromonitoring (IONM) can be broadly categorised into commercial proprietary solutions, research-oriented platforms, and procedure-specific training systems:

**Commercial Proprietary Solutions:** Leading biomedical device manufacturers, such as Inomed [30], Cadwell [25,26], and Neurosoft [31], provide simulation features primarily designed to familiarize users with their specific software interfaces and intraoperative settings. Among these, only Neurosoft explicitly lists a hardware-based "Patient Simulator" as a distinct accessory. However, these tools are generally closed-source and tethered to specific proprietary hardware, limiting their accessibility for broad academic or generalized training purposes.

**Research and Biomechanical Platforms:** In the open-source domain, OpenSim [32] represents a powerful standard for musculoskeletal modeling and movement analysis. While highly effective for biomechanics, its focus does not extend to the generation of specific electrophysiological signals (e.g., SSEP or MEP) required for neuromonitoring interpretation. Similarly, The Virtual Brain (TVB) [12] serves as a robust platform for simulating large-scale brain network dynamics. While it is an invaluable tool for neuroscience research, it is not designed to simulate discrete, real-time clinical scenarios for intraoperative educational applications.

**Procedure-Specific and Hybrid Systems:** Academic literature describes targeted implementations, such as a "Simulation system for intraoperative neuromonitoring" developed specifically for mastoidectomy training. This system utilises magnetic tracking to detect the position of a probe on a physical model, generating synthetic EMG signals based on proximity to a "virtual nerve." While this validates the clinical necessity and academic interest in such tools, it represents a hybrid solution (hardware-dependent) tailored to a single surgical procedure, rather than a generalised, software-only framework.

**Surgical Procedural Simulators:** Finally, platforms like SurgeonsLab [33] and InSimo [34] offer high-fidelity neurosurgical simulations. However, these tools predominantly focus on manual procedural skills, anatomical navigation, and haptic feedback, rather than on the analysis and interpretation of neurophysiological traces, which remains a distinct cognitive skill set. To contextualise the architecture of ION-Sim, it is essential to distinguish between a traditional Simulator and a Digital Twin, as their objectives and operational modes differ significantly.

A **Simulator** is primarily designed to study the behaviour of a system under hypothetical conditions using mathematical or logical models. Its connection to physical reality is often absent or limited, relying on static or pre-defined input data (historical or design data). Consequently, the output is a predictive test result that does not influence a real physical asset. It serves as a powerful tool for design, "what-if" scenario testing, and offline training, operating autonomously even before the physical object exists. Conversely, a **Digital Twin** [35] aims to dynamically replicate the state, condition, and behavior of a specific, existing physical asset. It requires a bidirectional, essential connection with physical reality, typically established via IoT sensors. A Digital Twin consumes real-time live data to update its internal state automatically, reflecting the exact condition of its physical counterpart. Its primary applications include real-time monitoring, predictive maintenance, and operational optimization across the asset's lifecycle.

While ION-Sim is natively a simulation framework, its architecture is designed to support Digital Twin-like interactions through the integration of the Lab Streaming Layer (LSL) protocol. This capability transforms the framework from a closed, static system into an open, interoperable

environment capable of bidirectional real-time data exchange. In the proposed architecture, the interaction with the “real world” (or external software agents) is managed through a low-latency multicast network:

**Outbound Stream:** ION-Sim generates neurophysiological data and distributes it via LSL, acting as a “virtual patient”.

**External Processing (The “Twin” Loop):** A connected external application captures this stream. This application can act as a dynamic “Anomaly Engine” or a feedback controller. Instead of using static parameters, it can process the incoming data and calculate perturbations or anomalies (randomised or logic-based) in real-time.

**Inbound Stream (Feedback):** The external application returns the modified parameters, or event triggers back to ION-Sim via a secondary LSL stream.

**Dynamic Update:** ION-Sim receives this control stream and instantaneously updates the ongoing simulation state. This closed-loop mechanism allows ION-Sim to function not just as a standalone trainer, but as a more complex, interconnected setup.

**Future Directions: Hybrid Simulation with Sensorized Phantoms:** Looking forward, this bidirectional LSL architecture lays the foundation for hybrid simulation paradigms. The framework is designed to be interfaced with physical sensorized phantoms (e.g., a mannequin equipped with cardio-respiratory and neurophysiological sensors). The phantom would provide afferent physical inputs (e.g., physical manipulation or surgical interventions by the trainee), which ION-Sim would process to generate efferent, realistic signal responses, effectively realizing a specialized Digital Twin for surgical training.

## 5. Conclusions

In this work, we presented ION-Sim, the first comprehensive, open-source framework specifically dedicated to the simulation of Intraoperative Neurophysiological Monitoring (IONM). By decoupling simulation logic from proprietary medical hardware, ION-Sim addresses a critical educational gap in the field, offering a scalable solution to the limitations of traditional operating room apprenticeship.

Through the integration of rigorous biophysical modelling, realistic signal processing technique and a structured pedagogical architecture, the framework successfully replicates the complexity of intraoperative decision-making. The ability to simulate a wide spectrum of modalities and to inject dynamic clinical anomalies allows trainees to develop and validate their diagnostic skills in a risk-free environment.

Furthermore, the release of the software under the GNU General Public License fosters transparency and collaboration, inviting the scientific community to extend the library with new physiological models and clinical scenarios. Ultimately, ION-Sim aims to democratize access to high-quality neurophysiological training and standardized assessment metrics, thereby contributing to improved patient safety in neurosurgery. The GitHub repository: [https://github.com/riccardo-budai/ION\\_Simula](https://github.com/riccardo-budai/ION_Simula)

**Author Contributions:** “Conceptualization, Budai R.; methodology, Budai R.; software, Budai R. and Blanco R.; formal analysis, Budai R.; investigation, Budai R.; resources, Blanco R.; data curation, Blanco R.; writing—original draft preparation, Blanco R.; writing—review and editing, Blanco R. and Budai R.; visualization and supervision, Budai R. All authors have read and agreed to the published version of the manuscript.”.

**Funding:** This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

**Institutional Review Board Statement:** “Not applicable”.

**Informed Consent Statement:** “Not applicable.”.:

**Data Availability Statement:** No new data were created or analyzed in this study. Data sharing is not applicable to this article.

**Acknowledgments:** During the preparation of this manuscript, the author(s) used Gemini 3.1, for the purposes of translation of text. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

**Conflicts of Interest:** The authors declare no conflicts of interest related to the publication of this article. No external funding was received for the preparation of this manuscript.

## Abbreviations

The following abbreviations are used in this manuscript:

|       |                                      |
|-------|--------------------------------------|
| EEG   | Electro Encephalography              |
| ECoG  | Electro Corticography                |
| MEP   | Motor Evoked Potential               |
| SEP   | Somatosensory Evoked Potential       |
| VEP   | Visual Evoked Potential              |
| Dwave | D wave                               |
| CAP   | Compound Action Potential            |
| CMAP  | Compound Motor Action Potential      |
| SFAP  | Single Fiber Action Potential        |
| MUAP  | Motor Unit Action Potential          |
| BAEP  | Brain Stem Acoustic Evoked Potential |

## Appendix A



**Figure 3.** ION-Sim system home screen.

### Appendix A.1 Module MEP

The Python code implements a MEP (Motor Evoked Potentials) Simulator designed for Intraoperative Monitoring (IOM) training. It features a sophisticated “Raster View” and “Stacked View” to visualize multi-channel motor responses across different clinical scenarios, such as upper limbs, lower limbs, or cranial nerves.

The simulation methodology focuses on generating realistic CMAP (Compound Muscle Action Potential) waveforms that respond dynamically to stimulation parameters and clinical anomalies. Each muscle response is synthesized mathematically to reflect its physiological properties:

- **Dual Gaussian Modeling:** A single MEP response is created by summing two Gaussian peaks— a primary negative wave and a secondary “polyphasic” shift. This mimics the complex morphology of real muscle potentials better than a single peak.

- **Muscle-Specific Parameters:** The simulator uses a global dictionary (ALL\_MUSCLE\_PARAMS) to assign unique latencies, amplitudes, and widths to each muscle. For example, the Deltoideus has a short latency (8 ms), while the FPA (foot) has a long latency (37 ms), reflecting the distance from the motor cortex.

In clinical tcMEP (transcranial MEP), a single pulse is often insufficient to elicit a response due to anesthesia. The code simulates this through:

- **Train-of-Pulses Facilitation:** The simulator applies a facilitation\_map where the final amplitude scale is determined by the number of pulses in the stimulator “train” (1 to 6 pulses).
- **Stimulation Artifacts:** It generates a series of high-frequency stimulation artifacts at the beginning of the trace, corresponding to the “Train Pulses” and “ISI” (Inter-Stimulus Interval) settings, providing a realistic visual reference for the moment of stimulation.

The simulator provides two complementary ways to analyze the signals:

- **Stacked View:** Displays the current MEP for each muscle vertically offset, allowing for easy identification of morphology and peak markers (onset and peak latency).
- **Raster History:** Implements a “waterfall” or scrolling history where the last 15 traces for each muscle are displayed. This is critical in IOM to detect subtle trends or sudden signal loss over time

The code integrates an AnomalyInjector system:

- **Physical Signal Modification:** When an anomaly is active, the injector modifies the physical data in real-time (e.g., causing a 50% drop in amplitude or a latency shift).
- **Decision-Making Assessment:** Once the user detects a change, the UI enables an “Anomaly Response” panel. The user must select the correct clinical action from a list of choices.



Figure 4. Mep module screen.

**AI Tutor Integration:** The user’s response is logged into a database and analyzed by an AI Agent. The agent provides personalized feedback and adjusts the simulation’s difficulty based on the user’s performance.



Figure 5. Dwave module screen.

#### Appendix A.2 Module BAEP

The Python code implements a BAEP (Brainstem Auditory Evoked Potential) Simulator BAEPdesigned for intraoperative monitoring (IOM) training. The application uses a multithreaded architecture where a BaepWorker handles mathematical signal generation while the BaepSimulator manages the GUI and real-time visualization.

The simulation synthesizes the BAEP signal using mathematical modeling to replicate the typical waveform complex (Waves I–V) and the technical challenges of clinical recording. The core of the BAEP complex is modeled by summing multiple Gaussian peaks.

Each wave (I, II, III, IV, V) is defined by specific latencies, amplitudes, and widths (standard deviation):

- Gaussian Model: The worker uses the formula  $Amp \cdot e^{-2\sigma^2(t-Lat)^2}$  to generate smooth, physiologically realistic peaks.
- Anatomical Differentiation: The simulator distinguishes between two recording channels:
  - Channel 1 (Ipsilateral, A1-Cz): Contains the full complex (Waves I, II, III, IV, and V).
  - Channel 2 (Contralateral, A2-Cz): Only simulates Waves III, IV, and V, with Wave III reduced in amplitude to 30%, reflecting the far-field nature of the contralateral recording.

The simulator replicates three clinical stimulation polarities that affect the morphology of the early response:

- Rarefaction/Condensation: These introduce a small latency shift (0.08 ms) and control the polarity of the Cochlear Microphonic (CM).
- Alternating: The worker alternates polarity every sweep; when these are averaged, the phase-inverted CM is cancelled out, which is a standard clinical technique to isolate the neural response.
- CM Generation: The CM is modeled as a sine wave that decays exponentially over time.



Figure 6. BAEP module screen.

To mimic the process of extracting the tiny BAEP signal (often  $< 0.5 \mu\text{V}$ ) from background noise:

- Noise Injection: Every individual sweep is combined with random Gaussian noise.
- Block Averaging: Signals are processed in “blocks” (e.g., 200 sweeps).
- Overlay Method: Instead of resetting to zero after every block, the system can retain a percentage of the previous block’s data (e.g., 75% overlay). This creates a smooth “moving average” effect in the history (Stack Plot), allowing trainees to see signals evolve as noise is reduced.

The system includes an automated analysis engine using `scipy.signal.find_peaks` to identify Waves I, III, and V. It searches for peaks within specific temporal windows (e.g., 5.0–6.5 ms for Wave V) and automatically calculates inter-peak latencies (I-III and I-V), which are critical for diagnostic purposes in IOM.

### Appendix A.3 Module SEP

The Python code implements a SEP (Somatosensory Evoked Potentials) Simulator. It simulates the afferent sensory pathway from the median nerve at the wrist to the cervical spine (CV7) and the primary somatosensory cortex (S1). The application utilizes a multithreaded architecture where a CAPWorker performs the heavy mathematical calculations, and a SepSimulator (PySide6-based) handles the real-time visualization. The simulation methodology is highly sophisticated, combining biophysical modeling for peripheral signals with real-world data interpolation for cortical responses.

The Compound Action Potential (CAP) at various distances (wrist, elbow, armpit, Erb’s point) is simulated using a bottom-up biophysical approach:

- Single Fiber Modeling: Each individual nerve fiber’s action potential (SFAP) is modeled as a biphasic pulse.
- Fiber Dispersion: The simulator generates a population of fibers (default 100) with varying Conduction Velocities (CV) ranging from 20 to 65 m/s.
- Temporal Summation: The CAP at a specific distance  $d$  is the sum of SFAPs, where each fiber’s delay is calculated as  $\text{Delay} = \text{Velocity} \times \text{Distance}$ . As the distance increases, the individual fiber potentials “spread out” (dispersion), accurately mimicking nerve physiology.

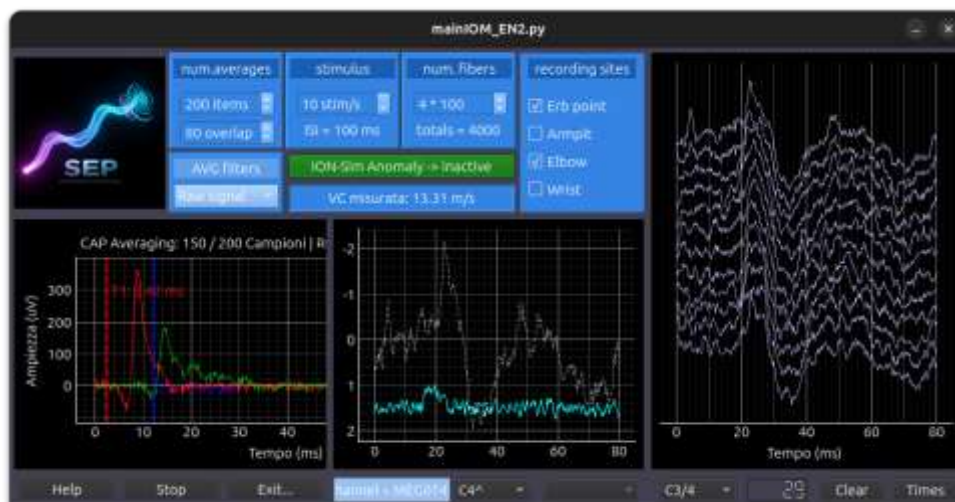


Figure 7. SEP module screen.

The cervical spinal response (Field Potential at CV6) is simulated using a Dipole Field Model:

- Static Field Potential: It calculates the potential using a Source-Sink dipole model with a fixed conductivity ( $\sigma$ ).
- Dynamic Current Pulse: The temporal evolution is governed by a current pulse function that models the rise and decay of the signal as the volley passes the cervical electrode.

Unlike the peripheral signals, the S1 Cortical response is derived from real human MEG (Magnetoencephalography) data:

- MNE Integration: The simulator loads a pre-recorded dataset from the “mne” library.
- Dynamic Interpolation: It interpolates and resamples the real MEG data to match the simulator’s specific time vector. Users can switch between different MEG channels (e.g., C3, C4) to see how the cortical SEP changes based on electrode location.

Signal Processing and Learning Integration:

- Averaging Engine: To mimic clinical practice, the simulator uses a FIFO (First-In-First-Out) buffer to perform signal averaging. This reduces injected Gaussian noise to reveal the evoked response.
- Waterfall Plotting: The system generates “Waterfall” views, which are historical stacks of averaged blocks, allowing users to monitor signal stability over time.
- Anomaly Injection and AI: The system includes an AnomalyInjector that can physically alter the signals (e.g., adding latency or decreasing amplitude) based on a JSON configuration. This is used for Learning Assessment, where the user must identify the problem and select the correct clinical action, which is then analyzed by an AI Agent for feedback.

#### Appendix A.4 Module VEP

The Python code implements a VEP (Visual Evoked Potentials) Simulator designed for Intraoperative Monitoring (IOM) training. The application uses PySide6 for the user interface and pyqtgraph for high-speed signal rendering, simulating a two-channel recording from the left and right occipital hemispheres. The simulation employs a mathematical modeling approach to synthesize the typical waveforms recorded from the visual cortex in response to light stimuli.

Waveform Synthesis via Gaussian Summation. The core morphology of the VEP signal is generated by summing multiple Gaussian peaks. Each significant component of the VEP complex is defined by its latency, amplitude, and width (standard deviation):

- N75: A negative peak at ~75 ms.

- P100: The most clinically significant positive peak at ~100 ms.
- N145: A negative peak at ~145 ms.

The mathematical formula [29] used for each wave component is:  $f(t) = \text{Amp} \cdot e^{-2\sigma^2(t-\text{Lat})^2}$  This ensures a smooth, physiologically realistic curve rather than sharp, unnatural transitions.

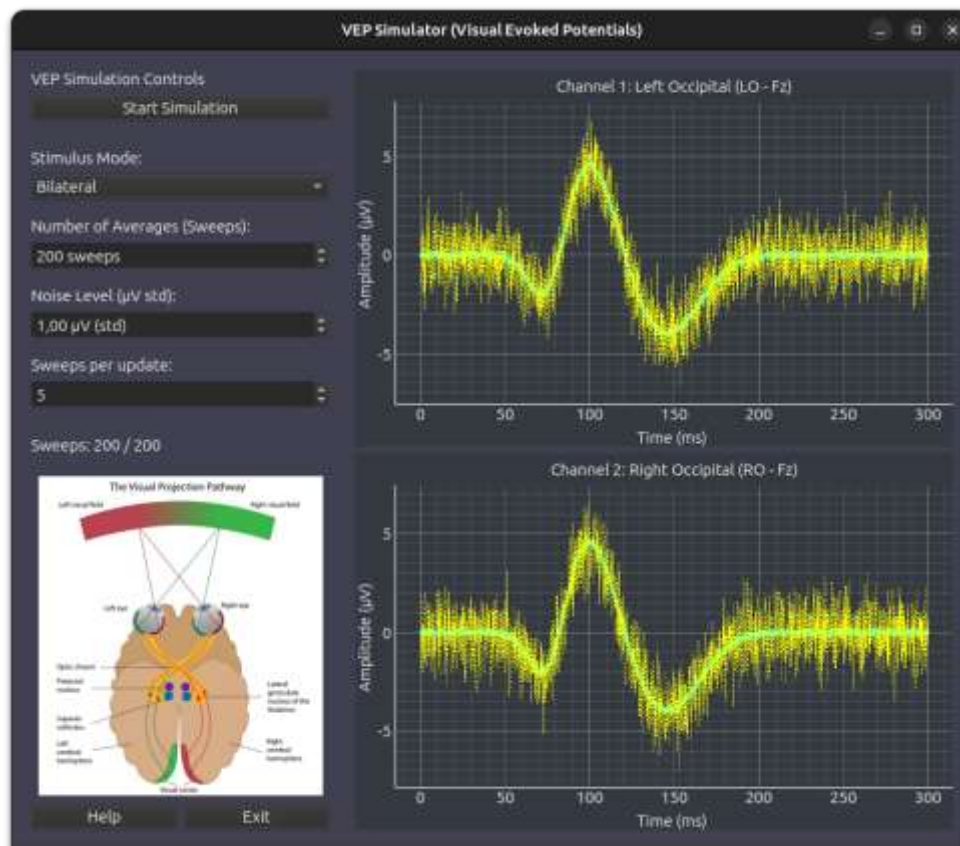


Figure 8. VEP module screen.

Stimulus Modality and Hemispheric Distribution. The simulator accounts for the anatomical decussation of the visual pathway. It allows for three stimulation modes that dynamically scale the amplitudes across the two channels:

- Bilateral: Full signal strength (100%) is applied to both the Left (Ch1) and Right (Ch2) Occipital channels.
- Monolateral (Left/Right): To simulate the partial crossing of fibers at the optic chiasm, the simulator applies full amplitude (1.0) to the contralateral hemisphere and a reduced scale factor (0.6) to the ipsilateral hemisphere.

Real-Time Averaging and Noise Injection. To replicate clinical recording conditions where the VEP signal is buried in background EEG noise:

- Gaussian Noise: Every “sweep” (individual stimulus response) is injected with random Gaussian noise based on a user-adjustable standard deviation.
- Moving Average: The simulator calculates a continuous mean across sweeps. As the sweep count increases toward the target (e.g., 200 sweeps), the random noise cancels out, and the stable VEP waveform emerges from the baseline.
- Dynamic Simulation Speed: Users can adjust “Sweeps per update” to accelerate the averaging process for training purposes.

Educational and Anomaly Features. The code includes a framework for Learning Management. It can load JSON-based anomalies to modify signal parameters in real-time—for example, increasing the latency of the P100 wave to simulate clinical conditions like optic neuritis.

### Appendix A.5 Module CAP

The Python code implements a Multi-Distance CAP (Compound Action Potential) Simulator designed for Intraoperative Monitoring (IOM) training. It simulates the propagation of electrical signals [29] along a nerve trunk (such as the dorsal columns) and visualizes the recording at multiple anatomical sites. The application is built using a multithreaded architecture: a CAPWorker class handles the intensive mathematical modeling in a background thread, while the MultiDistanceCAPSimulator class manages the graphical user interface (GUI). The simulation methodology is based on a bottom-up biophysical model that synthesizes the global nerve response from individual fiber contributions [36,37].

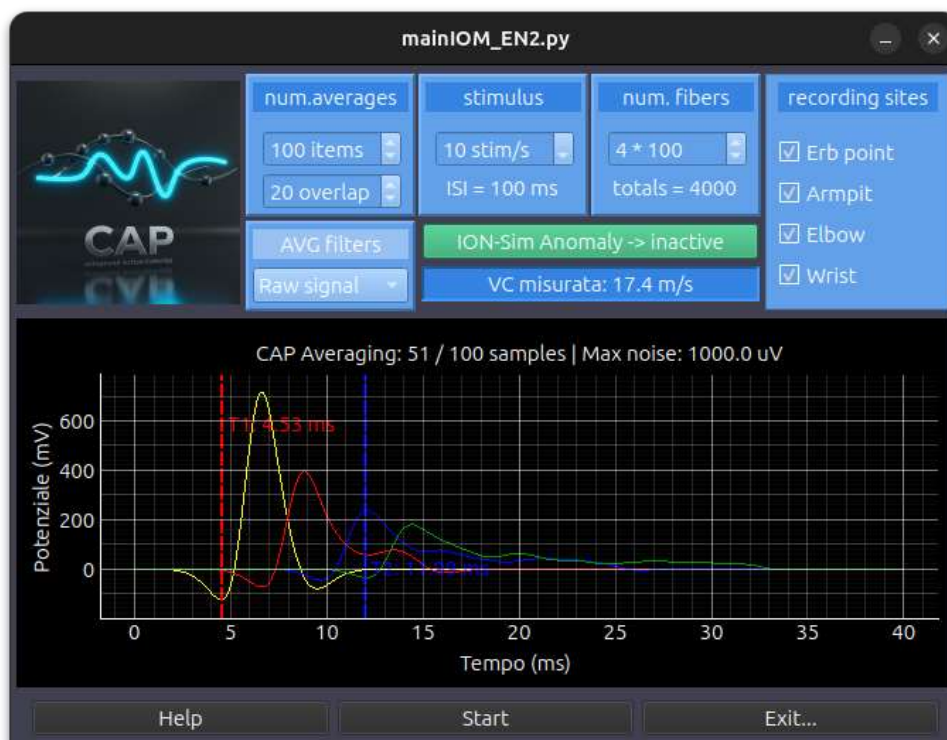


Figure 9. CAP module screen.

Individual Fiber Modeling (SFAP). The “building block” of the simulation is the Single Fiber Action Potential (SFAP). The code provides two mathematical models for this:

- **Bipolar Gaussian Model:** Combines two Gaussian functions (one positive and one negative) to create the characteristic biphasic shape of an extracellular recording.
- **Ricker Wavelet Model:** An alternative mathematical model (often called the “Mexican Hat”) used to simulate the SFAP pulse.

Nerve Trunk Simulation (Temporal Dispersion). To simulate a compound potential, the code models a population of nerve fibers (default 100 to 100,000 fibers):

- **Conduction Velocity (VC) Distribution:** Each fiber is assigned a velocity within a specific range (typically 20.0 to 65.0 m/s).
- **Spatial Propagation:** The potential for a specific distance (d) is calculated by applying a time delay to each fiber ( $\text{Delay} = VCd$ ).
- **Compound Summation:** The CAP displayed on the screen is the linear summation of all individual SFAPs. This naturally simulates temporal dispersion: as the recording site moves further from the stimulus, the signal becomes wider and lower in amplitude because faster and slower fibers arrive at increasingly different times.

Signal Conditioning and Averaging. Because the CAP is often buried in noise, the simulator replicates clinical recording techniques:

- Gaussian Noise Injection: Random noise is added to the raw signal to simulate background interference.
- FIFO Averaging: The simulator uses a First-In-First-Out (FIFO) buffer system (default 100 averages) to perform signal averaging, which improves the signal-to-noise ratio over time.
- Digital Filtering: Users can apply various filters in real-time, including Moving Average (Smooth), Spline Interpolation, and Butterworth Bandpass filters (e.g., 5-3000 Hz or 10-1500 Hz).  
Interactive Clinical Features.
- Velocity Measurement: The GUI includes interactive vertical cursors. By placing cursors on the peaks of signals from different distances (e.g., Wrist at 7cm and Elbow at 20cm), the software automatically calculates the measured Conduction Velocity in m/s.
- Anomaly Injection: The code integrates an AnomalyInjector that can physically alter the signal in real-time based on a JSON configuration. This is used for training scenarios where a student must identify clinical changes (such as ischemia or compression) and choose the correct corrective action from a dropdown menu.

#### Appendix A.6 Module CMAP

The provided Python code implements a simulation framework for CMAP (Compound Muscle Action Potential) and MEP (Motor Evoked Potential) signals, which are critical components in Intraoperative Neuromonitoring (IOM) for assessing the integrity of motor pathways. The simulation is built on the principle of Summation of Motor Unit Action Potentials (MUAPs). Rather than using a single static wave, the code synthesizes a global muscle response by modeling hundreds of individual motor units and summing their electrical contributions.

Individual MUAP Waveform Modeling. The “building block” of the signal is the single MUAP:

- Triphasic Morphology: The `single_muap_waveform` function models a triphasic (Positive-Negative-Positive) wave. This is achieved by combining three Gaussian-like phases (P1, N1, and P2) with specific amplitudes and time offsets.
- Stochastic Variation: For every motor unit in the simulation, the code randomly varies the amplitude (simulating different motor unit sizes) and the duration/sigma (simulating the physical characteristics of the muscle fibers).

Conduction Velocity and Temporal Dispersion. A core feature of the CMAP simulation is how it handles the propagation of the signal from the nerve to the muscle:

- Conduction Velocity (VC) Distribution: Each motor unit is assigned a specific velocity based on a normal distribution (e.g., mean of 60 m/s with a standard deviation).
- Propagation Delay: The software calculates a unique delay for each unit based on the anatomical distance ( $\text{Delay} = \text{Velocity} \times \text{Distance}$ ).
- Temporal Dispersion: Because different fibers conduct at different speeds, the individual MUAPs arrive at the recording electrode at slightly different times. This causes the resulting CMAP to “spread out” and lower in amplitude as the distance increases, accurately reflecting nerve physiology.

CMAP vs. MEP Simulation Logic. The code differentiates between a peripheral nerve response (CMAP) and a central nervous system response (MEP):

- CMAP Methodology: Primarily focuses on peripheral conduction velocity dispersion and a fixed synaptic/junctional delay (e.g., 2.3 ms) to model the time it takes for the signal to cross the neuromuscular junction.
- MEP Methodology: Introduces the concept of Central Jitter (`tccm_spread_ms`). Unlike the synchronized stimulation of a peripheral nerve, a cortical stimulation (tcMEP) results in desynchronized firing of the corticospinal tract. The code models this by adding a “jitter” or

temporal spread to each MUAP, which creates the complex, multi-peaked morphology typical of clinical MEPs.

#### Appendix A.7 Module EEG

The Python code implements an EEG (Electroencephalography) and Auxiliar Signal Simulator designed for real-time neurophysiological monitoring and processing. The application uses PySide6 for the interface and pyqtgraph for high-performance signal rendering, integrating the MNE-Python library for specialized EEG data handling. The simulation methodology is based on a template-based streaming model combined with dynamic artifact injection and real-time spectral analysis. The system acts as a “virtual amplifier” by streaming real EEG data from clinical files:

- MNE CNT Loading: The simulator loads Neuroscan .cnt files using the `mne.io.read_raw_cnt` module.
- Real-time Resampling: If the original file’s sampling frequency differs from the target (default 512 Hz), the code performs on-the-fly resampling.
- Microvolt Scaling: Data is scaled by a factor of  $-1 \times 10^6$  to ensure signals are displayed in standard microvolts ( $\mu\text{V}$ ).

As the data is “played back” in chunks (default 100ms intervals), several processing layers are applied:

- Average Referencing: The spatial mean of all 16 EEG channels is subtracted from each channel to minimize common-mode noise.
- Frequency Filtering: The worker applies a bandpass filter (typically 1.6–35 Hz for EEG and 1.0–25 Hz for ECG) to remove slow drifts and high-frequency muscle interference.

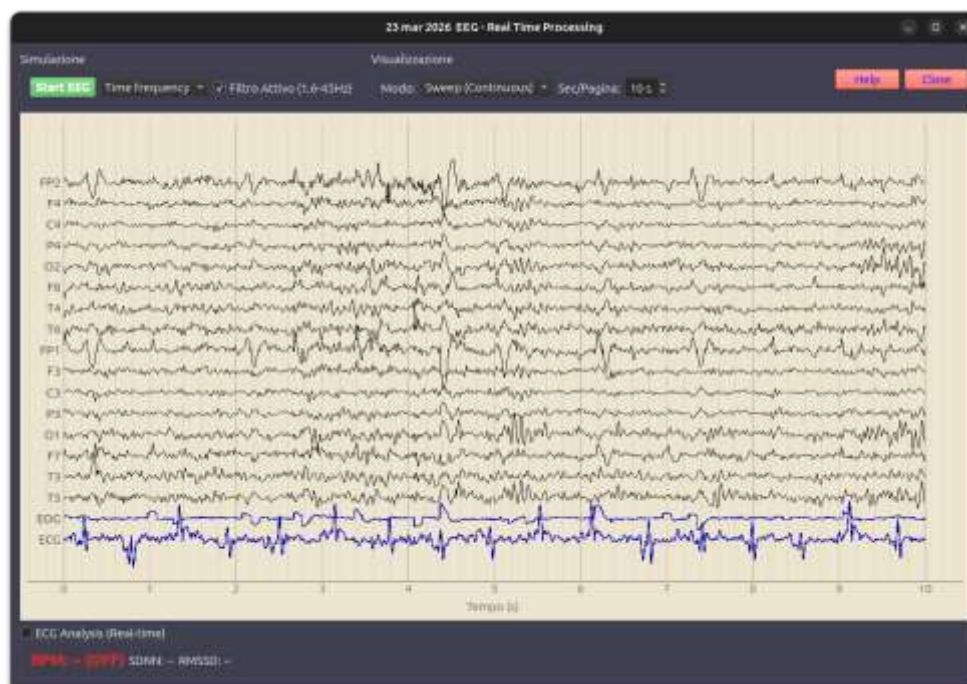


Figure 10. EEG module screen.

To replicate the clinical environment, the `add_eeg_artifacts_dynamic` function overlays synthetic noise onto the clean EEG data based on user GUI controls:

- EOG Blinks: The simulator injects vertical eye movement artifacts into the frontal channels (FP1, FP2) by scaling the loaded EOG template.

- EMG Noise: High-frequency Gaussian noise is added to all channels to simulate patient tension or muscle movement.

The simulation extends beyond simple time-domain viewing through advanced background workers:

- Block-based PSD: Every 2 seconds, the worker triggers a Power Spectral Density (PSD) calculation using the Welch method to display the frequency distribution (Delta, Theta, Alpha, Beta bands).
- Spectral Whitening: A specific feature allows for  $1/f$  “whitening” to flatten the power spectrum, making higher frequency oscillations like Alpha and Beta more visible against the dominant lower frequencies.
- ECG R-Peak Detection: If enabled, the system runs the `analyze_r_peaks` utility on the ECG channel, calculating Heart Rate (BPM) and Heart Rate Variability (SDNN, RMSSD) in real-time.

Visualization Features: Sweep vs. Page Mode: The GUI supports “Sweep” (continuous scrolling) or “Page” (static screen updates) visualization modes.

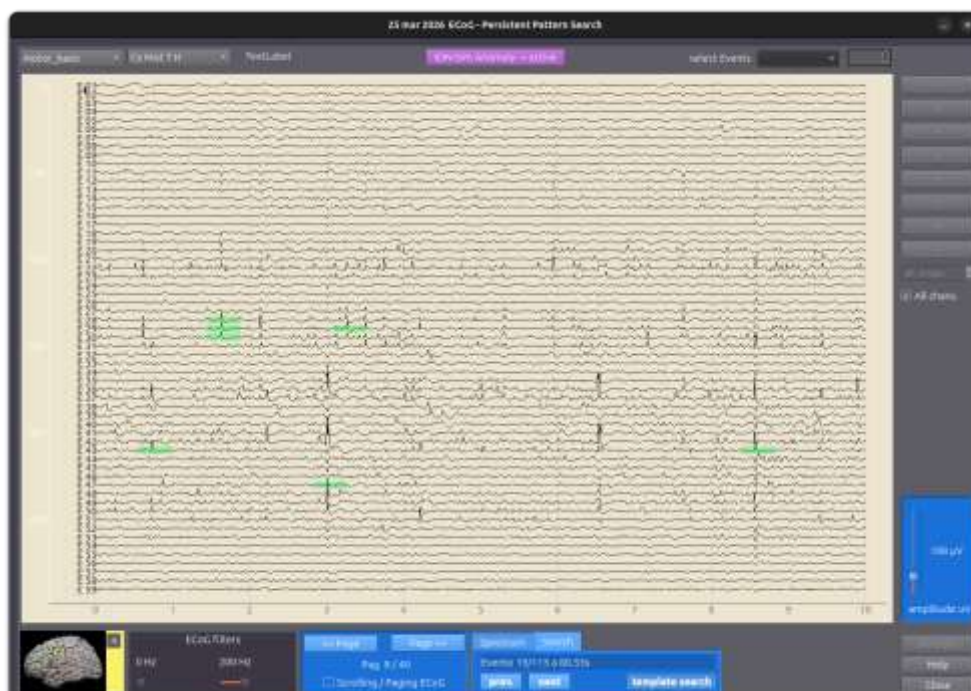
Spectra Window: A separate window provides specialized views for Channels PSD or Time-Frequency Representations (TFR), including an interactive cursor to measure specific decibel (dB) values at targeted frequencies

#### Appendix A.8 Module ECoG

The simulation does not generate purely synthetic mathematical data (unlike simpler simulators); instead, it employs a hybrid playback and artifact-injection model.

Data Playback Engine:

- Template-Based Streaming: The simulator loads real ECoG recordings from `.mat` files (using `scipy.io`) and converts them into MNE `RawArray` objects.
- Worker-Timer Architecture: A dedicated `ECoGWorker` runs on a separate thread to prevent UI freezing. It streams data chunks at specific intervals (default 100ms) to mimic live acquisition from an amplifier.
- Circular Buffer Logic: The `_run_simulation_step` method manages a circular pointer over the loaded dataset, ensuring continuous data flow by wrapping back to the start once the file ends.



**Figure 11.** ECoG module screen.

Real-Time Signal Conditioning. As the data is streamed, several digital signal processing (DSP) steps are applied:

- Common Average Reference (CAR): For each chunk, the spatial mean across all electrodes is subtracted from each channel to remove global noise and artifacts common to all sensors.
- Bandpass Filtering: A dynamic Butterworth filter (applied via `scipy`) allows users to adjust low-cut (e.g., 2Hz) and high-cut (e.g., 300Hz) frequencies in real-time to focus on specific oscillations like Gamma or Ripple bands.

JSON-Driven Artifact & Anomaly Injection. A core feature of this simulator is its ability to inject clinical or technical anomalies dynamically:

- ArtifactGenerator Integration: The simulator uses an external `ArtifactGenerator` class that reads configurations from a JSON file.
- Dynamic Modification: Based on the JSON “scenario,” the worker modifies the raw ECoG signal in real-time. This can include:
  - Specific Channel Targeting: Anomalies can be applied to all channels or a subset.
  - Time-Locked Events: Using a continuous time vector, the generator can overlay sinusoidal noise, spikes, or baseline drifts.

Advanced Feature: Persistent Pattern Search. The “Persistent Pattern Search Edition” includes a methodology to find recurring neurophysiological events (such as interictal spikes or specific oscillations):

- ROI Selection: Users select a “Region of Interest” on the plot using a graphical UI element (`LinearRegionItem`).
- FFT-Based Cross-Correlation: To achieve a “100x speedup,” the code uses Fast Fourier Transforms (via the `fast_normalized_cross_correlation` utility) to scan the entire recording for matches that exceed a user-defined correlation threshold (e.g., 0.80).

Averaging Logic: Once matches are found, the system extracts those segments and computes an Event-Related Average, allowing the user to compare the “Model” pattern against the statistical average of all found hits.

#### *Appendix A.9 Module Electromyography Free Running*

Il modulo di simulazione EMG registrato con elettrodi ad ago subdermici, è simulato come white noise signal su cui sono sovrapposte le anomalie riscontrabili in uno scenario intraoperatorio: A-train (neurotonic), B-burst (rhythmic), B-spike (intermittent), C-train (complex). Il modulo ha una capacità di individuare automaticamente gli eventi in relazione ad un soglia di ampiezza regolabile.

Tutti gli eventi e le annotazioni associate sono registrati e salvati in una lista con l’orario e l’identificatore del muscolo che ha presentato quello specifico pattern.

#### *Appendix A.9 Module Anesthesia*

The Python code implements an Anesthesia Simulator tailored for Intraoperative Monitoring (IOM) contexts. It simulates the pharmacokinetics (PK) and pharmacodynamics (PD) of common anesthetic agents and their impact on vital signs like Heart Rate (HR), Blood Pressure (MAP), and Heart Rate Variability (HRV). The simulation methodology is based on three main pillars: Target Controlled Infusion (TCI) models, Pharmacodynamic interaction logic, and Stochastic vital sign generation.

The simulator calculates plasma concentrations ( $C_p$ ) and effect-site concentrations ( $C_e$ ) in real-time using established medical models:

- Propofol (Schnider Model) [38]: Uses a three-compartment model to calculate how Propofol distributes and is eliminated based on the patient’s age and weight.

- Remifentanyl (Minto Model) [39]: A specialized model for high-potency opioids, accounting for rapid onset and offset.
- Ketamine: Modeled as a sympathomimetic agent that counteracts the depressive effects of other drugs on the cardiovascular system.
- Sevoflurane: Simulates inhalation anesthesia measured in MAC (Minimum Alveolar Concentration).



Figure 12. EMG free running module screen.



Figure 13. Anesthesia module screen.

Pharmacodynamic (PD) Interaction Logic. The code simulates how these drugs interact to affect the “Patient State”:

- Synergy vs. Antagonism: It calculates a global depressive effect from Propofol, Remifentanyl, and Sevo. This combined effect reduces Heart Rate and MAP.
- Ketamine Offset: Ketamine is programmed with a “sympathomimetic” factor. As the Ketamine concentration increases, it mathematically offsets the bradycardia and hypotension caused by Propofol, raising HR and MAP back toward baseline.

Vital Signs Simulation Engine. The physiological signals are generated using a combination of deterministic and stochastic processes:

- ECG Waveform: A synthetic ECG signal is generated by concatenating a baseline with a “QRS complex” (a high-frequency triangle wave). The frequency of these complexes is tied directly to the calculated instantaneous Heart Rate.
- Respiratory Sinus Arrhythmia (RSA): The simulator introduces small, natural fluctuations in Heart Rate to mimic the interaction between breathing and the heart.
- HRV Metrics: The code calculates SDNN (Standard Deviation of NN intervals) and RMSSD in real-time. These metrics decrease as the depth of anesthesia increases, providing a neurophysiological indicator of autonomic nervous system depression.

Interactive Clinical Features

- MAP Calculation: The simulator approximates Mean Arterial Pressure (MAP) using the drug concentrations and a random walk component to simulate blood pressure stability.
- Real-time Controls: Users can adjust drug dosages via sliders, and the simulation immediately updates the decay curves ( $C_e$ ) and the resulting physiological waveforms.
- Visualization: It uses pyqtgraph to provide a “Vital Signs Monitor” view (ECG) alongside a “Pharmacokinetic” view (Concentration curves), bridging the gap between drug administration and physiological response.

## Appendix B

Simulation systems can be classified into three fundamental categories:

**Biophysical Simulators:** Tools that model biophysical principles using a bottom-up approach, starting from individual ion channels and single neurons. Their primary objective is to foster conceptual understanding of fundamental biological mechanisms.

**Clinical Waveform Generators:** Instruments engineered to replicate clinically recognizable signals and their pathological variants. These tools are specifically designed for training in diagnostic pattern recognition.

**Complex Systems and Research Platforms:** High-level frameworks for modeling large-scale neural networks and global brain dynamics. These platforms often necessitate advanced programming skills and are primarily suited for academic research or advanced postgraduate medical education.

**Neurosim 5:** A commercial software package specifically engineered for teaching cellular neurophysiology at both undergraduate and postgraduate levels. Its core philosophy centers on accessibility; it requires no prior programming knowledge, enabling students to conduct virtual experiments that would be unfeasible in a traditional laboratory setting [4]. The strength of the platform lies in its modular architecture, which deconstructs the complexities of neurophysiology into several distinct components:

**GOLDMAN Module:** Simulates the Goldman-Hodgkin-Katz equation, allowing students to manipulate ionic concentrations and membrane permeabilities to directly observe the generation of the resting potential. This module provides a tangible representation of the Nernst and Goldman equations, which are fundamental to all aspects of electrophysiology [5].

**HODGKIN-HUXLEY Module:** Implements the original action potential model. Students can apply stimuli in current-clamp or voltage-clamp modes, vary the temperature, administer virtual

pharmacological agents (such as tetrodotoxin to block sodium channels), and investigate phenomena such as refractory periods and accommodation [5].

**PASSIVE CONDUCTION Module:** Simulates the cable properties of an ideal axon or dendrite. This is crucial for grasping electronic conduction and signal propagation—key concepts for understanding temporal dispersion in evoked potentials and nerve conduction studies [6].

**NETWORK Module:** Facilitates the construction of small neuronal circuits. Students can link neurons via chemical or electrical synapses, exploring emergent phenomena such as lateral inhibition, oscillatory circuits, and synaptic integration. This module serves as a conceptual bridge toward understanding the network activity that underlies EEG signals [4].

Furthermore, the platform is supported by an extensive suite of tutorials, guided exercises, and classroom activities, offering a “turnkey” solution for educators [4]. Its longevity and continuous updates are well-documented in academic literature, consolidating its status as a benchmark pedagogical tool [7].

**MetaNeuron:** A freeware application designed for interactive educational sessions, focusing on core neurophysiological principles through simplified simulations [8]. Its interface facilitates real-time parameter manipulation, providing instantaneous feedback on neuronal responses and promoting an exploration-based learning paradigm. Its guided “lessons” cover topics parallel to those in Neurosim, including resting membrane potential (based on passive conductances and membrane time/space constants) and the action potential as modeled by the Hodgkin-Huxley equations [8]. Its open-access nature makes it an ideal solution for resource-constrained educational settings.

**SEREEGA (Simulating Event-Related EEG Activity):** An open-source MATLAB-based toolbox engineered for generating simulated EEG data, with a specific focus on event-related potentials [9]. Its primary objective is to provide researchers with a known ground truth to validate signal processing methods, source localization techniques, or Brain-Computer Interface (BCI) classification algorithms [10]. SEREEGA is an optimal tool for advanced curricula in biomedical engineering, computational neuroscience, or signal processing. It enables students to gain hands-on experience with the EEG forward problem: how dipolar source activations within the brain volume project onto scalp electrodes. Users can define components with specific spatial coordinates and orientations, assign various activation signals (such as Event-Related Potentials, ERP, or Event-Related Spectral Perturbations, ERSP), and observe the resulting EEG [9]. While the software supports diverse head models and offers high extensibility, it requires a MATLAB environment and familiarity with the EEGLAB toolbox [9].

**The Virtual Brain (TVB):** A high-performance open-source research platform designed to simulate large-scale, whole-brain network dynamics [11]. Unlike other simulators, TVB constructs personalized brain models integrated with an individual’s structural neuroimaging data (e.g., DTI/DSI tractography for white matter connectivity). Using this connectivity matrix, it simulates neuronal population activity to generate emergent neurophysiological signals, including EEG, MEG, and fMRI [11]. In a pedagogical context, TVB is best suited for postgraduate and research levels. It offers unparalleled insights into how brain anatomy constrains and shapes dynamic function, allowing for the study of emergent brain rhythms or the propagation of pathological activity, such as epileptic seizures [12]. The TVB EduPack, along with numerous online tutorials, provides educational use cases, interactive Jupyter notebooks, and video guides that lead users through the replication of published studies on epilepsy and Alzheimer’s disease [13].

**LabVIEW Biomedical Toolkit:** An add-on for National Instruments’ LabVIEW graphical development environment, a standard in engineering for data acquisition and analysis [14]. Within this toolkit, the “Simulate EEG” Virtual Instrument (VI) offers a pragmatic and highly controllable approach to EEG signal simulation [15]. Through a Graphical User Interface (GUI), users can specify fundamental signal parameters, including output amplitude, sampling frequency, and the relative power of canonical frequency bands (delta, theta, alpha, beta) [15]. A key pedagogical feature is the

ability to introduce realistic artifacts; users can inject controlled-amplitude white noise and, crucially, 50 Hz or 60 Hz power-line interference to simulate real-world recording conditions [15].

**EMG Simulator by Stålberg Software:** Widely considered the benchmark tool for EMG education and research [16]. This standalone Windows application is independent of specific EMG hardware [16]. Its exceptional fidelity stems from its underlying biophysical model: the software generates Motor Unit Action Potentials (MUAPs) by summing individual muscle fiber action potentials, calculated according to current literature and biophysical principles [16]. Users maintain granular control over a wide range of physiological and recording parameters, including:

Fiber parameters: Diameter and jitter (variability in neuromuscular transmission).

Motor unit parameters: Fiber count, spatial distribution, and end-plate localization. Recording parameters: Electrode type (e.g., concentric) and precise intramuscular positioning [16].

This level of customization allows for an in-depth exploration of how MUAP morphology (amplitude, duration, phases) depends on these factors. However, its most powerful pedagogical feature is the capacity to explicitly simulate pathological conditions. Users can model scenarios for myopathic diseases (fiber loss), neurogenic disorders (reinnervation with increased fiber density), and myasthenic syndromes (increased jitter and blocking) [16]. This functionality is indispensable for training neurologists and clinical neurophysiologists, as it correlates pathophysiological findings with diagnostic EMG patterns in a controlled environment. The developer also provides the MNCS Simulator, designed to teach the principles of motor nerve conduction studies, including complex concepts such as conduction block and axonal loss [17]. **LabVIEW Biomedical Toolkit:** Analogous to its EEG counterpart, the toolkit includes a “Simulate EMG” VI. This instrument allows for the configuration of parameters such as contraction time, amplitude, noise levels, and frequency spectra [18]. Although less biophysically detailed than the Stålberg simulator, it provides a robust, engineering-oriented platform for understanding the fundamental properties of both surface and intramuscular EMG signals. To date, no standalone software equivalent to Neurosim or the Stålberg EMG Simulator has been identified that is specifically designed to teach the generation and clinical interpretation of SEPs, VEPs, BAEPs, or MEPs [19].

**ADInstruments LabChart:** This platform constitutes an integrated hardware (PowerLab) and software (LabChart) ecosystem for biological signal acquisition [48]. It utilizes various stimulators (electrical, visual, auditory) to evoke real potentials from human or animal subjects [20]. Its pedagogical value lies in its pre-configured experimental packages and sample data files. For instance, the “Visual Evoked Potential (VEP) – Pre-Lab Prep” module [21] enables students to analyze VEP data, teaching them to measure the latencies and amplitudes of key components (e.g., N75, P100, N145) and interpret their clinical significance, even in the absence of live recordings [22].

**iWorx LabScribe:** Similar to LabChart, iWorx provides an integrated hardware-software system tailored for educational laboratories [23]. The LabScribe software includes modules for Event-Related Potential (ERP) analysis and offers guided experiments covering auditory and visual reflexes, providing a framework for studying evoked potentials. While the platform includes an ECG Simulator and an Arrhythmia Generator—demonstrating significant simulation capabilities—these do not currently extend to neurophysiological evoked potentials [24].

**Cadwell Sierra Software:** This is a clinical-grade diagnostic system used in hospital settings for EMG, nerve conduction studies, and a comprehensive suite of evoked potentials (SSEP, VEP, AEP, MEP) [26]. Its application in education occurs primarily through hands-on clinical workshops where participants utilize the hardware to perform tests and analyze clinical cases [25]. It does not appear to feature a standalone educational “simulation mode” that can be operated without a patient or subject.

Scientific literature regarding **MEP simulation** focuses largely on the creation of statistical models to understand inter-trial variability—a research goal aimed at optimizing Transcranial Magnetic Stimulation (TMS) dosing [27]. These models, however, are not implemented in accessible educational software. Clinical systems, such as the Cadwell Sierra, provide protocols for MEP acquisition but lack dedicated simulation features for teaching purposes [28].

This “evoked potential simulation gap” creates a pedagogical dependency on expensive hardware ecosystems or advanced technical expertise (such as MATLAB proficiency for SEREEGA). This introduces a significant barrier to accessibility and equity for institutions lacking substantial laboratory budgets or dedicated technical staff. While foundational concepts can be taught using free or low-cost software, the practical instruction of evoked potentials risks becoming a privilege reserved for well-funded institutions capable of maintaining high-end data acquisition labs. Consequently, the development of a user-friendly, open-source evoked potential simulator— analogous to Neurosim— would represent a substantial contribution toward the global democratization of neurophysiological education.

## References

1. Loftus, C. M., Biller, J., & Baron, E. M. (Eds.). (2014). *Intraoperative Neuromonitoring*. New York, NY: McGraw-Hill Education Medical.
2. Deletis, V., Shils, J. L., Sala, F., & Seidel, K. (A cura di). (2020). *Neurophysiology in Neurosurgery: A Modern Approach* (2<sup>a</sup> ed.). Academic Press (Elsevier).
3. Verst, S. M., Barros, M. R., & Maldaun, M. V. C. (Eds.). (2022). *Intraoperative Monitoring: Neurophysiology and Surgical Approaches*. Springer Nature Switzerland AG.
4. <https://www.st-andrews.ac.uk/~wjh/neurosim/>
5. <https://apps.microsoft.com/detail/9nc7xnpkmszm?hl=en-US&gl=US>
6. <https://www.st-andrews.ac.uk/~wjh/neurosim/modules.html>
7. Heitler, B. Neurosim: Some Thoughts on Using Computer Simulation in Teaching Electrophysiology doi: 10.59390/JGIP5297
8. Newman, M. H., Newman, E. A. (2013) *MetaNeuron: A Free Neuron Simulation Program for Teaching Cellular Neurophysiology* J Undergrad Neurosci Educ 2013 Oct 15;12(1):A11–A17.
9. <https://github.com/lrkrol/SEREEGA>
10. <https://lrkrol.com/tools.html>
11. <https://ebrains.eu/data-tools-services/tools?filter=whole-brain-simulation&page=12>
12. <https://www.thevirtualbrain.org/tvb/zwei/neuroscience-simulation>
13. <https://wiki.ebrains.eu/bin/view/Collabs/documentation/tutorials/The%20Virtual%20Brain/>
14. <https://www.ni.com/en/support/downloads/tools-network/download.labview-biomedical-toolkit.html>
15. [https://www.ni.com/docs/en-US/bundle/labview-biomedical-toolkit-api-ref/page/lvbiomed/bio\\_sim\\_eeg.html](https://www.ni.com/docs/en-US/bundle/labview-biomedical-toolkit-api-ref/page/lvbiomed/bio_sim_eeg.html)
16. <https://stalbergsoftware.com/emgsimulators.aspx>
17. <https://stalbergsoftware.com/mnccsimulators.aspx>
18. [https://www.ni.com/docs/en-US/bundle/labview-biomedical-toolkit-api-ref/page/lvbiomed/bio\\_sim\\_emg.html](https://www.ni.com/docs/en-US/bundle/labview-biomedical-toolkit-api-ref/page/lvbiomed/bio_sim_emg.html)
19. <https://www.st-andrews.ac.uk/~wjh/neurosim/>
20. <https://www.adinstruments.com/research/human/neuro/stimulation>
21. <https://www.adinstruments.com/support/downloads/windows/visual-evoked-potential-vep-%E2%80%93-pre-lab-prep>
22. <https://www.physoc.org/magazine-articles/new-virtual-adventure-in-physiology-practicals/>
23. <https://iworx.com/labscribe/>
24. <https://iworx.com/products/psychological-physiology/pk-tr-psychological-physiology-roam-teaching-kit/>
25. <https://cadwell.education/biomedical-training/>
26. <https://www.cadwell.com/clinical-solutions/electrodiagnostics/evoked-potentials/>
27. Goetz, S. M., Madhi Alavi, S. M., Deng, Zhi-De, Peterchev, A. V. Statistical Model of Motor Evoked Potentials for Simulation of Transcranial Magnetic and Electric Stimulation <https://doi.org/10.1101/406777>
28. <https://github.com/lrkrol/SEREEGA>
29. Nunes, P. L., Nunes, M. D., Srinivasan R. *Multi-Scale Neural Sources of EEG: Genuine, Equivalent, and Representative. A Tutorial Review* Brain Topography (2019) 32:193–214 <https://doi.org/10.1007/s10548-019-00701-3>

30. <https://www.inomed.com/>
31. <https://neurosoft.com/en/catalog/iom>
32. <https://simtk.org/projects/opensim>
33. <https://surgeonslab.com/>
34. <https://www.insimo.com/>
35. <https://www.ibm.com/think/topics/digital-twin>
36. Contento, G., Budai R., *Characterisation of the Conduction Properties of Nerves with the Distribution of Fibre Conduction Velocities*, in Bioinstrumentation and Biosensors, ed. D.L. Wise, Marcel Dekker, Inc., New York, 1991
37. Contento, G., et al., *Conduction Velocity Distributions of Normal and Pathological Fibres in the same Nerve*, Proc. VIII Annual Conf. of the IEEE Engineering in Medicine and Biology Society, Fort Worth, 1986
38. Linassi, F., Zanatta P., Spano L., Burelli P., Farnia A., Carron M. *Schnider and Eleveld Models for Propofol Target-Controlled Infusion Anesthesia: A Clinical Comparison* Life (Basel) 2023 Oct 16;13(10):2065. doi: 10.3390/life13102065
39. Eleveld, D. J., ColinAnthony, P., Absalom, R., Struys M. M.R.F., *Target-controlled-infusion models for remifentanyl dosing consistent with approved recommendations* British Journal of Anaesthesia, 2020; 125, 483-491

**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.