Enabling real-time interaction with an electrophysiological cancer cell model

a BioTechMed-Graz Lab Rotation Report

of Peter Julius Waldert

supervised by **Prof. Christian Baumgartner** at the
Institute of Health Care Engineering with European Testing Center for Medical Devices,
Graz University of Technology, Austria,

29th of February, 2024.

1 Summary

Lung cancer is one of the most widespread pathologies worldwide and its mechanisms, specifically at the level of individual cells, are not well understood. We improve on the A549 electrophysiological cancer cell model introduced in Langthaler, Rienmüller et al. 2021; Langthaler, Zumpf et al. 2024, combining numerical methods with an efficient implementation to reduce simulation time and accuracy to a level where it is feasible for live interaction. More specifically, we were able to accelerate the simulation with adaptive timestepping and a highly efficient implementation in the Rust programming language, while we also managed to approach the corresponding inverse problem using a quadratic program, solving it within milliseconds. We introduce a visualisation approach of the entire model in the form of a live simulation dashboard available at `https://in-silico-cancer-cell.waldert.at/` running directly in the browser. The entire source code is freely available on GitHub and reusable through three different channels: the simulation interface (powered by compilation to WebAssembly), the Rust linkable library implementation and a Python package (simply run: `pip install in-silico-cancer-cell`). Our aim behind a distribution in this way is to make the topic and simulation as accessible as possible.

2 Introduction

Lung cancer is one of the most widespread pathologies worldwide and its mechanisms, specifically those of individual A549 cells, are not well understood. Computational techniques can help with a better understanding of the behaviour of these cancer cells. We work with the A549 model introduced in Langthaler, Rienmüller et al. 2021, together with a calcium channel extension introduced in Langthaler, Zumpf et al. 2024, reimplementing the model in the Rust programming language and performing a number of numerical optimizations such as adaptive timestepping. We also verify the model's performance in *Floating* mode, as compared to an individual simulation of the estimated number of channels. We introduce a visualisation approach of the entire model in the form of a live simulation dashboard[1]. The entire source code behind this simulation is freely available on GitHub[2], and reusable through three different channels: the simulation interface (powered by WebAssembly), the Rust linkable library implementation and a Python package. These three interfaces all originate from the same source code and our aim behind a distribution in this way is to make the simulation as accessible as possible. In order to find appropriate parameters for the model, an optimization procedure is performed. Multiple optimization approaches for the solution of the corresponding inverse problem (fitting model parameters to measurement data) are put in comparison. The measurements are obtained using a *Patch-Clamp System*, where one records the current through the membrane given a voltage protocol.

3 Methods

A cell's membrane consists of multiple ion channels, categorized into $M \in \mathbb{N}$ different types. Each ion channel is represented in one of $N_{s,k} \in \mathbb{N}$ states, which, in physical terms, is related to a positional configuration of a protein within the ion channel. Only some states can be observed directly, and its development only depends on the one previous state. Hence, we are working with a Hidden Markov Model (HMM). For many ion channel categories, their transition probabilities are voltage or ion-concentration dependent.

The whole cell current $I : T \to \mathbb{R}$ over time $t \in T \subset \mathbb{R}^+$ is then obtained as the sum of all individual channel contributions $I_k, k \in \{1, ..., M\}$ over $M \in \mathbb{N}$ channel types

$$I(t) := \sum_{k=1}^{M} N_k I_k(t) = \sum_{k=1}^{M} N_k g_k p_{o,k} \left(V(t) - E_k\right) , \quad (1)$$

where N_k is the number of channels of type $k \in \{1, ..., M\}$, g_k is the respective ion channel's conductivity, $p_{o,k} \in [0, 1]$ is the probability of observing the channel in a state where an ion current can flow ("open states"), $V : T \to \mathbb{R}$ is the

[1] `https://in-silico-cancer-cell.waldert.at/`
[2] `https://github.com/MrP01/InSilicoCancerCell`

voltage across the membrane and $E_k \in \mathbb{R}$ the reversal potential.

Within the simulation, we sample the state and current at discrete time points $T_{\text{meas}} \subset T$, for example

$$T_{\text{meas}} := \left\{ t_n := \sum_{i=0}^{n} (\Delta t)_i \;\middle|\; n \in \mathbb{N}_0 \;\middle|\; n < N_t \right\}$$

for N_t measurements with step size $(\Delta t)_n$, which may be chosen equally large for all $n \in \{0, ..., N_t - 1\}$. We adapt this time interval $(\Delta t)_n \in \mathbb{R}^+$ per simulation step based on a state change heuristic, cf. Section 3.1.

At each time step,

$$\boldsymbol{s}_{k,n+1} = H_k \left(V(t_n), \boldsymbol{C}(t_n), t_n \right) \boldsymbol{s}_{k,n} \tag{2}$$

where $\boldsymbol{s}_{k,n} \in [0,1]^{N_{s,k}}$ is the state vector of ion channel type k at the n-th time step, $H_k\left(V, \boldsymbol{C}, t_n\right) \in [0,1]^{N_{s,k} \times N_{s,k}}$ the transition matrix for type k with $\sum_{j=1}^{N_{s,k}} \{H_k\}_{i,j} = 1 \;\forall i$, $V(t_n)$ the voltage across the membrane at time t_n and $\boldsymbol{C}(t_n) \in \mathbb{R}^4$ the concentrations of Kalium, Calcium, Sodium and Chlorine at time t_n. We initialize the simulation at $t_0 = 0$ with $\boldsymbol{s}_{k,0} = (1, 0, ..., 0)^T$ for all k.

Given the state $\boldsymbol{s}_k$, current measurements are then simply

$$\boldsymbol{I} := \left(I(t_0), I(t_1), ..., I(t_{N_t - 1}) \right)^T \in \mathbb{R}^{N_t} ,$$

with I as stated above in Equation (1) and

$$p_{o,k} = \sum_{j \in \mathcal{S}_{o,k}} \{\boldsymbol{s}_{n,k}\}_j ,$$

where $\mathcal{S}_{o,k}$ is the set of all states contributing to the ion channel current, the "open states".

3.1 Adaptive Timestepping

In order to accelerate the simulation in areas where there is little change to the dynamics, we choose an adaptive step size based on

$$(\Delta t)_{n+1} = (\Delta t)_n \left(\frac{\Delta^{\text{tol}}}{\sum_{k=1}^{M} N_k \left\| \boldsymbol{s}_{k,n+1} - \boldsymbol{s}_{k,n} \right\|_2} \right)^{1/2} , \tag{3}$$

for all n, where $\Delta^{\text{tol}} \in \mathbb{R}^+$ is a measure for the allowed state change in between steps. When the state changes too quickly in between time steps, the above heuristic will decrease $(\Delta t)_{n+1}$ and vice-versa. In principle, it would be feasible to apply the adaptive timestepping to each ion channel type individually, however this would make current sampling and data synchronization between channels hard to realize, considering the ion concentration dependence of H_k. Within this paper, we set $\Delta^{\text{tol}} = 2 \cdot 10^{-7}$.

3.2 Inverse Problem

When regarding the cell model as a whole, the number of ion channels N_k per type k may be put into a configuration vector $\boldsymbol{N} := (N_1, ..., N_M)^T \in \mathbb{N}_0^M$ and the total simulated current I sampled at measurement points T_{meas} can be expressed as a matrix-vector product

$$\boldsymbol{I} = \sum_{k=1}^{M} N_k \boldsymbol{I_k} = R\boldsymbol{N} , \tag{4}$$

where $R \in \mathbb{R}^{N_t \times M}$ is the matrix of all current measurements per channel type.

Given the individual ion channel type models' parameters, which we know from literature (cf. **??**), the question that remains is how many channels there are of each type to fit the measurements. This problem can be solved using a number of optimization approaches.

However, the formulation in Equation (4) also gives rise to a least-squares formulation, by projecting the measured current into the space of all individual channel currents. More specifically, we want to find

$$\boldsymbol{N}_{\text{opt}} = \underset{\boldsymbol{N} \in \mathbb{N}_0^M}{\arg\min} \frac{1}{2} \left\| R\boldsymbol{N} - \boldsymbol{I}_{\text{meas}} \right\|_2^2 , \tag{5}$$

with $\boldsymbol{I}_{\text{meas}} \in \mathbb{R}^{N_t}$ the experimentally measured current. The most important constraint here is that of integer non-negativity, $\boldsymbol{N}_{\text{opt}} \in \mathbb{N}_0^M$, which makes this problem hard to solve directly.

The unconstrained least-squares problem could be solved very efficiently using a QR-decomposition of the current basis $R = Q_b R_b$, its solution would be $\boldsymbol{N}_{opt} = \lfloor R_b^{-1} Q_b^* \boldsymbol{I}_{\text{meas}} \rceil \in \mathbb{Z}^M$.

3.3 Formulation as a Quadratic Program

Relaxing the integer condition on the solution, and letting $\boldsymbol{d} := \boldsymbol{I}_{\text{meas}}$ for brevity, we can reformulate Equation (5),

$$\boldsymbol{N}_{\text{opt}} \approx \underset{\boldsymbol{x} \in \mathbb{R}_+^M}{\arg\min} f(\boldsymbol{x}) = \underset{\boldsymbol{x} \in \mathbb{R}_+^M}{\arg\min} \, 1/2 \left\| R\boldsymbol{x} - \boldsymbol{d} \right\|_2^2 ,$$

with cost function $f : \mathbb{R}^M \to \mathbb{R}^+$, which we manipulate to

$$\begin{aligned} f(\boldsymbol{x}) &= 1/2 (R\boldsymbol{x} - \boldsymbol{d})^T (R\boldsymbol{x} - \boldsymbol{d}) \\ &= 1/2 \left(\boldsymbol{x}^T R^T R \boldsymbol{x} - \boldsymbol{x}^T R^T \boldsymbol{d} - \boldsymbol{d}^T R \boldsymbol{x} + \boldsymbol{d}^T \boldsymbol{d} \right) \\ &= 1/2 \left(\boldsymbol{x}^T P \boldsymbol{x} + \boldsymbol{x}^T \boldsymbol{q} + \boldsymbol{q}^T \boldsymbol{x} \right) \\ &= 1/2 \boldsymbol{x}^T P \boldsymbol{x} + \boldsymbol{q}^T \boldsymbol{x} \end{aligned}$$

where we let $P := R^T R \in \mathbb{R}^{M \times M}$ and $\boldsymbol{q} := -R^T \boldsymbol{d} \in \mathbb{R}^M$ and leave out the constant $\boldsymbol{d}^T \boldsymbol{d}$. We can express the non-negativity constraint $\boldsymbol{x} \geq \boldsymbol{0}$ as an equality constraint using a slack variable $\boldsymbol{s} \in \mathbb{R}_+^M$,

$$-\boldsymbol{x} + \boldsymbol{s} = \boldsymbol{0} \quad \Leftrightarrow \quad A\boldsymbol{x} + \boldsymbol{s} = \boldsymbol{b} ,$$

where we set $A := -\mathbb{1} \in \mathbb{R}^{M \times M}$ and $\boldsymbol{b} := \boldsymbol{0} \in \mathbb{R}^M$. This leaves us with a constrained *quadratic program*,

$$\min_{\boldsymbol{x} in \mathbb{R}^M} \frac{1}{2} \boldsymbol{x}^T P \boldsymbol{x} + \boldsymbol{q}^T \boldsymbol{x}, \tag{6}$$

$$s.t. \; A\boldsymbol{x} + \boldsymbol{s} = \boldsymbol{b} , \; \boldsymbol{s} \in \mathbb{R}_+^M . \tag{7}$$

We solve the quadratic problem in this exact form using Clarabel Goulart and Chen 2024. Note that in Clarabel notation, the slack variable is to be taken as an element of the nonnegativity cone.

The integer solution can then be obtained from rounding,

$$\boldsymbol{N}_{\text{opt}} = \lfloor \boldsymbol{x} \rceil \in \mathbb{N}_0^M \, .$$

3.4 Implementation and Usage

The core simulation is implemented in the Rust programming language Matsakis and Klock 2014. Each channel is

```python
# pip install in-silico-cancer-cell
import in_silico_cancer_cell
```

3.5 Live Simulation

The live

4 Results

4.1 Adaptive Timestepping

With the standard forward iteration approach, the simulation takes 412 million steps to simulate over all voltage protocols and cell cycle phases, while the adaptive timestepping only requires 9 million steps for the same configuration. While keeping the same level of accuracy when matching with the experimental data, our adaptive timestepping method is therefore 45 times faster than the standard approach, on average.

4.2 Inverse Problem

We compare different optimization approaches based on their runtime and root mean square error,

$$\Delta = \sqrt{{}^1\!/_{N_t} \sum_{j=1}^{N_t} (I_j - I_{\text{meas},j})^2} = \frac{\|\boldsymbol{I} - \boldsymbol{I}_{\text{meas}}\|_2}{\sqrt{N_t}} \, .$$

5 Outlook

Numerically, the stability of the simulation varies greatly with the time step state change tolerance Δ^{tol}, this could be improved using a higher-order integration scheme. Regarding the simulation dashboard, there are still many adjustments that could improve and enable further usage perspectives.

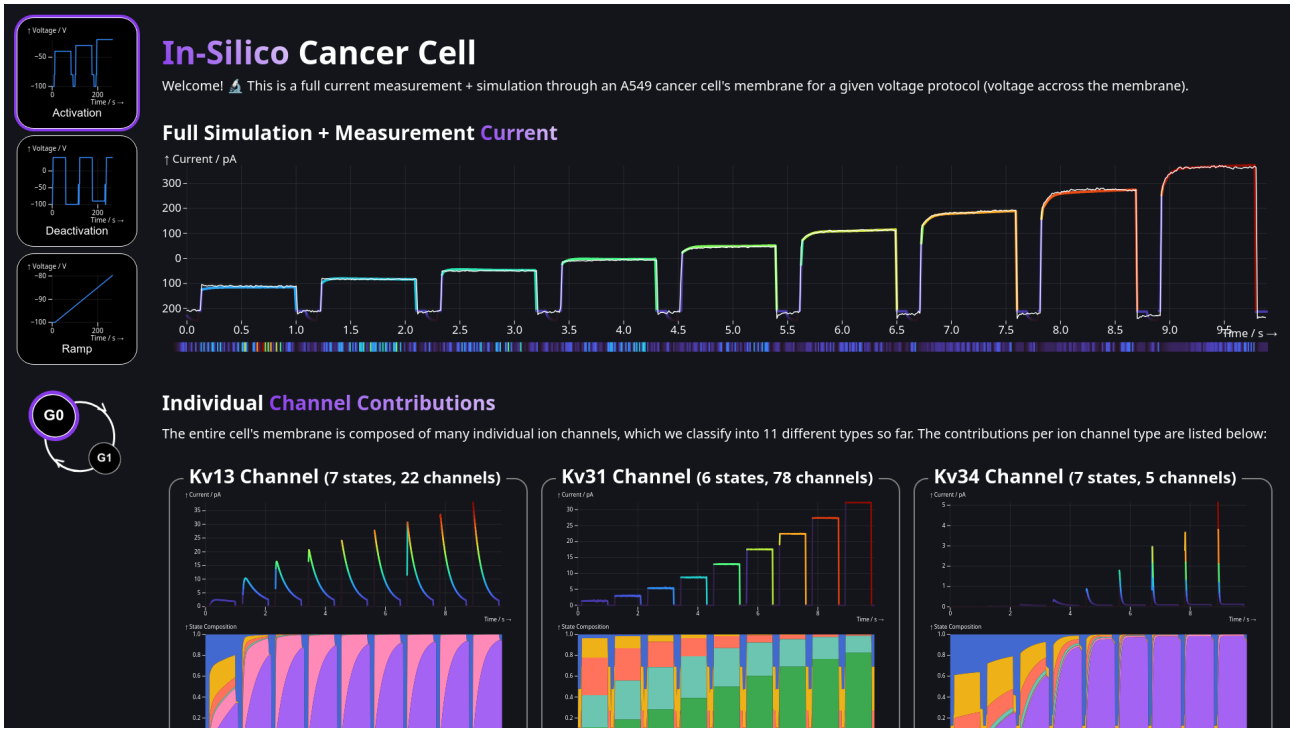

Figure 6: Screenshot of the live simulation dashboard, available here: https://in-silico-cancer-cell.waldert.at/. This is the graphical user interface to the simulation written in Rust. The full simulation, completing within around 500ms, runs live in the browser using Rust's bindings to WebAssembly. One can choose the voltage protocol and cell cycle phase (G0/G1) in the menu on the left, while the full current and individual channel currents and internal states can be seen on the right. The remaining ion channel types have been cut off on this screenshot.

Signatures

This project was carried out as part of a BioTechMed-Graz *Lab Rotation*. The aim of the programme is to give interested graduates a chance to broaden their scientific interests beyond the topic of their master's thesis.

(Peter Waldert) Place and Date

(Prof. Christian Baumgartner) Place and Date

References

Langthaler, Sonja, Theresa Rienmüller, Susanne Scheruebel, Brigitte Pelzmann, Niroj Shrestha, Klaus Zorn-Pauly, Wolfgang Schreibmayer, Andrew Koff and Christian Baumgartner (June 2021). 'A549 in-silico 1.0: A first computational model to simulate cell cycle dependent ion current modulation in the human lung adenocarcinoma'. In: *PLoS Comput. Biol.* 17.6, e1009091. DOI: 10.1371/journal.pcbi.1009091.

Langthaler, Sonja, Christian Zumpf, Theresa Rienmüller, Niroj Shrestha, Julia Fuchs, Rui Zhou, Brigitte Pelzmann, Klaus Zorn-Pauly, Eleonore Fröhlich, Seth H. Weinberg and Christian Baumgartner (May 2024). 'The bioelectric mechanisms of local calcium dynamics in cancer cell proliferation: an extension of the A549 in silico cell model'. In: *Front. Mol. Biosci.* 11, p. 1394398. DOI: 10.3389/fmolb.2024.1394398.

Goulart, Paul J. and Yuwen Chen (2024). *Clarabel: An interior-point solver for conic programs with quadratic objectives*. arXiv: 2405.12762 [math.OC].

Pusch, Michael, Raffaella Magrassi, Bernd Wollnik and Franco Conti (Aug. 1998). 'Activation and Inactivation of Homomeric KvLQT1 Potassium Channels'. In: *Biophys. J.* 75.2, pp. 785–792. ISSN: 0006-3495. DOI: 10.1016/S0006-3495(98)77568-X.

Matsakis, Nicholas D. and Felix S. Klock (Oct. 2014). 'The rust language'. In: *Ada. Lett.* 34.3, pp. 103–104. ISSN: 1094-3641. DOI: 10.1145/2692956.2663188.

Bro, Rasmus and Sijmen De Jong (Sept. 1997). 'A fast non-negativity-constrained least squares algorithm'. In: *J. Chemom.* 11.5, pp. 393–401. ISSN: 0886-9383. DOI: 10.1002/(SICI)1099-128X(199709/10)11:5<393::AID-CEM483>3.0.CO;2-L.