Spreadsheets in Education (eJSiE)

Volume 10 | Issue 1 Article 1

6-1-2017

A Spreadsheet Implementation of the Hodgkin-Huxley Model for Action Potentials in Neurons

Florian Henning Geyer
Theodor-Heuss-Gymnasium Goettingen, florian.h.geyer@gmail.com

A Spreadsheet Implementation of the Hodgkin- Huxley Model for Action Potentials in Neurons

Abstract
A Microsoft Excel workbook is presented that enables students to explore the Hodgkin-Huxley Model for action potentials in neurons. The model is implemented by a system of four coupled nonlinear ordinary differential equations and requires a numerical solution method. The method used in the workbook is the fourth-order Runge-Kutta method. The workbook provides a user interface to control the simulations by a variety of parameters and to display the model results. All related data is available to allow the addition of charts to the workbook. The workbook is designed to challenge the students' understanding of the Hodgkin-Huxley Model and to encourage them to create their own simulations that can be checked in a real lab course.

Keywords
Hodgkin- Huxley Model, Neurophysiology, Excel Spreadsheet, Systems of Ordinary Differential Equations, Fourth- Order Runge- Kutta Methods

1. Introduction

The Hodgkin-Huxley Model for action potentials in neurons [1,2] is one of the fundamental models of neuroscience. It simulates not only the form, time course and other features of action potentials with high accuracy, but also many other electrical properties of a neuron. While the experimental exploration of action potentials is standard in many neuroscience or biology education lab classes, it is also helpful for students to have a closer look at the mathematical modelling of action potentials in order to acquire a deeper understanding of the underlying processes. Working with the model, students can design and try many more different situations than in a real lab, just by changing some model parameters and starting the model calculation.

To do so, a versatile, easy-to-use modelling environment is required that offers modification of modelling parameters and/or external quantities, to quickly re-calculate the model and graphically display the results. Students should have access to all data, formulas and programming code related to the model calculation, but it should also be possible to study the model behaviour without reading or editing programming code.

Since the Hodgkin- Huxley Model has become a well- established part of the education in life sciences, publications describing tools or programs [3-5] and also online simulations [6] for calculating the model are available. In most cases these are based on 3GL or 4GL programming languages and compiled modules, and the program codes are not readable. Commercial applications may be expensive and won't offer any access to their code for IP reasons [7]. Even if codes are published, students interested in running them and/or modifying the model or its numerous parameters need a language-specific programming environment and programming skills. Brown [3] summarizes the advantages of Excel for student education and explains how to set up an Excel template capable of calculating the model [4]. His approach does not emphasize that the model consists of a set of four ODEs and seems to be based on the Euler method to solve differential equations without stating this explicitly.

This paper presents an MS Excel workbook that has been designed to be fully accessible and understandable for students who do not want to dig in program code. With a minimum knowledge of how to handle Excel in- cell formulas, and a "Cockpit" that makes systematic variation of all parameters easy, the model and its implementation can already be deeply explored, manipulated or extended. Students who want to go further may add or modify VBA code. MS Excel is, as part of Microsoft Office, a widespread and popular application and easily available to all students that offers all features needed.

The workbook comprises an implementation of the Hodgkin- Huxley Model, the fourth-order Runge- Kutta method to solve the ordinary differential equations (ODE) related to the model, and a user interface for running the model and displaying results in a number of predefined or user- made charts. The workbook is running on MS Excel 2010 for Windows, MS Excel 2013 for Mac OS and probably other versions that have not been checked by the author. Only standard Excel and VBA libraries are required.

The workbook extends Brown's approach [3,4] by leaving all model-specific formulas and parameters in-cell, but implementing the fourth-order Runge- Kutta method in VBA Code [8,9]. This method proves to be stable and precise in this context and speeds up the simulation. The VBA module implements the fourth- order Runge-Kutta method in a generic way and can be re- used in a completely different context. The usage of a VBA function that interprets in-cell formulas and calls the Runge-Kutta routine makes this possible.

Charts can be easily rescaled by using a VBA macro. For comparison, the Hodgkin-Huxley equations are placed right above the data tables that comprise the respective function values.

2. The Hodgkin- Huxley Model

It is not intended here to present a thorough discussion and derivation of the model which is much better explained in modern textbooks of neuroscience [2]. In this paper it is assumed that the reader is familiar with the related neuroscientific context, such as electrical properties of neurons, action potentials and their origin, etc. According terms will be used here without further explanation.

The Hodgkin- Huxley Model consists of four coupled non-linear first order ODEs (1)-(4). Eight more functions (5)-(12) are involved to fully define the model. Calibration of the model is provided by 7 parameters defining the static membrane properties (c_M, $\overline{g_{Na}}$, $\overline{g_K}$, $\overline{g_L}$, E_{Na}, E_K, E_L) and 18 parameters defining the membrane dynamics ($\propto_{x,0}$, $\beta_{x,0}$, $U_{\propto x}$, $U_{\beta x}$, $K_{\propto x}$, $K_{\beta x} - x \in \{n, m, h\}$). Extensive quantities are given per unit area.

$$j_{ext}(t) = g_{Na}(U_M) \cdot (U_M - E_{Na}) + g_K(U_M) \cdot (U_M - E_K) + \bar{g}_L \cdot (U_M - E_L) + c_M \frac{dU_M(t)}{dt} \quad (1)$$

$$\frac{dn}{dt} = \alpha_n(U_M) \cdot [1 - n(t)] - \beta_n(U_M) \cdot n(t) \quad (2)$$

$$\frac{dm}{dt} = \alpha_m(U_M) \cdot [1 - m(t)] - \beta_m(U_M) \cdot m(t) \quad (3)$$

$$\frac{dh}{dt} = \alpha_h(U_M) \cdot [1 - h(t)] - \beta_h(U_M) \cdot h(t) \quad (4)$$

$$g_K(U_M) = \bar{g}_K \cdot n^4(U_M) \quad (5)$$

$$g_{Na}(U_M) = \bar{g}_{Na} \cdot m^3(U_M) \cdot h(U_M) \quad (6)$$

$$\alpha_n(U_M) = \frac{\alpha_{n_0} \cdot (U_M - U_{\alpha_n})}{1 - e^{\frac{-(U_M - U_{\alpha_n})}{K_{\alpha_n}}}} \quad (7)$$

$$\beta_n(U_M) = \beta_{n_0} \cdot e^{\frac{-(U_M - U_{\beta_n})}{K_{\beta_n}}} \quad (8)$$

$$\alpha_m(U_M) = \frac{\alpha_{m_0} \cdot (U_M - U_{\alpha_m})}{1 - e^{\frac{-(U_M - U_{\alpha_m})}{K_{\alpha_m}}}} \quad (9)$$

$$\beta_m(U_M) = \beta_{m_0} \cdot e^{\frac{-(U_M - U_{\beta_m})}{K_{\beta_m}}} \quad (10)$$

$$\alpha_h(U_M) = \alpha_{h_0} \cdot e^{\frac{-(U_M - U_{\alpha_h})}{K_{\alpha_h}}} \quad (11)$$

$$\beta_h(U_M) = \frac{\beta_{h_0}}{1+e^{\frac{-(U_M - U_{\beta_h})}{K_{\beta_h}}}} \quad (12)$$

The first equation relates the change rate of the membrane potential U_M to ionic K^+ and Na^+ current densities, a leakage current density, and a current density j_{ext}, externally injected into the neuron. j_{ext} simulates a stimulus originating from other neurons. c_M is the capacity per unit area of the cell membrane. The currents are driven by the differences between the membrane potential U_M and the individual Nernst potentials [2] E_{Na}, E_K, and E_L established across the semi- permeable membrane. While the electrical conductance $\overline{g_L}$ of the leakage current is assumed constant, g_{Na} and g_K strongly depend on U_M. This dependence accounts for the characteristics of K^+ and Na^+ ion channels located in the cell membrane. Activation (opening) and inactivation (closing) of these channels is modelled by the functions n, m, and h, defined by ODEs (2) – (4). These functions vary between 0 and 1, yielding the fraction of ion channels per unit area being activated (n for K^+ channels, m for Na^+ channels) or inactivated (h for Na^+ channels). g_{Na} and g_K are given by (6) and (7) with $\overline{g_{Na}}$ and $\overline{g_K}$ the maximum conductances per unit area when all channels are activated. The rate functions $\propto_x (U_M)$ and $\beta_x(U_M)$ model the dynamics of ion channel activation and inactivation processes. Not yet activated channels open with a rate $\propto_x (U_M)$, and activated channels close with a rate $\beta_x(U_M)$. In case of the inactivation function h, $\propto_x (U_M)$ and $\beta_x(U_M)$ change their open/close roles.

1.1. Model Calibration

The Excel workbook uses the calibration of the model given by Hodgkin and Huxley in their original paper for the squid neuron [1]. Table 1 shows the respective values with short descriptions and their named ranges in the workbook.

Important conventions are: the extracellular electrical potential is set to zero, which means that the intracellular potential is negative (-65mV for the squid investigated by Hodgkin and Huxley), inward currents of positive ions and outward currents of negative ions are positive, outward currents of positive ions are negative.

Table 1:
Model calibration values and related named ranges in the Excel workbook. All values are taken from [1]. The value E_L is selected such that $j_{Total} = 0$ in the resting state of the neuron. The c_M value differs from Hodgkin and Huxley because the model is more stable using a value of two.

Parameter	Equation	Description	Named Range	Value	Unit
$c_{M,}$	(1)	membrane capacity per unit area	_C	2	µF/cm²
$\bar{g}_{Na}$	(6)	maximum electrical conductance of the membrane per unit area for Na+ ions, all channels activated	g_Na	120	mS/cm²
$\bar{g}_K$	(5)	dito for K^+ ions	g_K	36	mS/cm²
$\bar{g}_L$	(1)	electrical conductance per unit area for the leakage currents	g_L	0.3	mS/cm²
E_{Na}	(1)	Nernst potential of Na^+ ions	E_Na	55	mV
E_K	(1)	Nernst potential of K^+ ions	E_K	-77	mV
E_L	(1)	Nernst potential of leakage current carriers	E_L	-54.5574	mV
$\propto_{n_0}$	(7)	parameters of rate functions $\propto_n (U_M)$ and $\beta_n(U_M)$	alpha_n	0.01	1/ms
β_{n_0}	(8)		beta_n	0.125	1/ms
$U_{\propto_n}$	(7)		U_an	-55	mV
U_{β_n}	(8)		U_bn	-65	mV
$K_{\propto_n}$	(7)		K_an	10	mV
K_{β_n}	(8)		K_bn	80	mV
$\propto_{m_0}$	(9)	parameters of rate functions $\propto_m (U_M)$ and $\beta_m(U_M)$	alpha_m	0.1	1/ms
β_{m_0}	(10)		beta_m	4	1/ms
$U_{\propto_m}$	(9)		U_am	-40	mV
U_{β_m}	(10)		U_bm	-65	mV
$K_{\propto_m}$	(9)		K_am	10	mV
K_{β_m}	(10)		K_bm	18	mV
$\propto_{h_0}$	(11)	parameters of rate functions $\propto_h (U_M)$ and $\beta_h(U_M)$	alpha_h	0.7	1/ms
β_{h_0}	(12)		beta_h	1	1/ms
$U_{\propto_h}$	(11)		U_am	-65	mV
U_{β_h}	(12)		U_bm	-35	mV
$K_{\propto_h}$	(11)		K_am	20	mV
K_{β_h}	(12)		K_bm	10	mV

2. Implementation and Usage

The Excel workbook consists of two sheets and two visual basic modules. Parameters of the equations (1) to (12) are given named ranges. The students do not need to deal with the VBA modules to be fully able to use the model, but the code is readable and extendable, e.g. to solve more than four coupled ODE or a system of 2nd order ODE, or to refine the model.

1.2. Cockpit Sheet

The cockpit sheet (Figure 1) allows entering all values required for calibrating the model and triggering action potentials by single or multiple external rectangular current pulses.

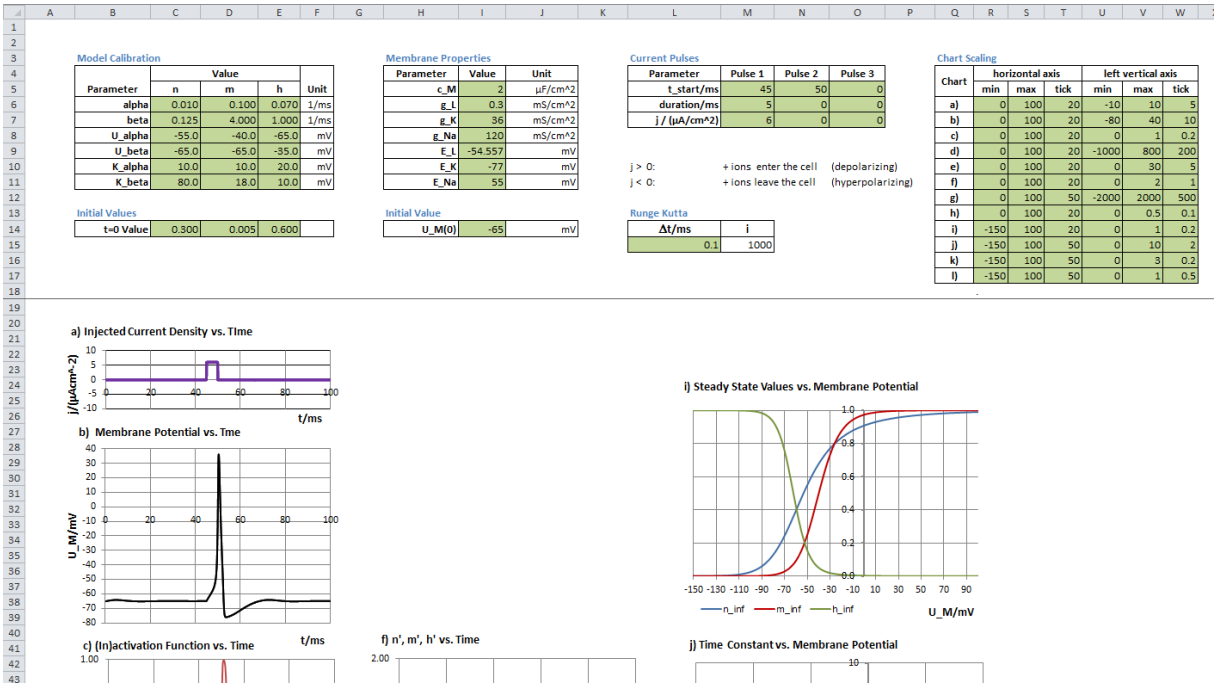


Figure 1: Screenshot of the top part of the Cockpit sheet. Not all charts are visible. The tables "Model Calibration" and "Membrane Properties" contain the calibration settings given in Table 2. Rectangular current pulses are set in table "Current Pulses". Initial values are set in row 14, the incremental step size of the simulation is set in cell L15. The "Chart Scaling" Table allows to set individual scales for each chart.

The solution of the system of ODE requires $t=0$ values for $U_M(t)$, $n(t)$, $m(t)$, and $h(t)$ which are also defined in the Cockpit sheet. Only green cells should be edited by the user. Up to three consecutive rectangular current pulses can be defined.

Each change in one of the green input fields triggers a recalculation of the model, and the calculation results are shown near real-time in a number of charts. Thus, the cockpit serves as a kind of neurophysiology lab. The student can systematically modify the model calibration and/or the external current pulses and watch the model neuron's response. In fact, it is possible to reproduce the experimental findings of Hodgkin's group with good quality. The students can even design and conduct more "experiments", examine the results and challenge their understanding of neuron behaviour –all that without the problems usually related to real experimental setups.

Table 2 lists all charts available in the Cockpit sheet.

Table 2:
List of Charts available in the Cockpit sheet.

Chart name	Horizontal axis	Vertical axis
a) Injected Current Density vs. Time	t in ms	current density j in μAcm^{-2}
b) Membrane Potential vs. Time	t in ms	U_M in mV
c) (In)activation Function vs. Time	t in ms	n(t), m(t), h(t)
d) Current Density vs. Time	t in ms	current density j in μAcm^{-2}
e) K-, Na- and Leak Conductance vs. Time	t in ms	g_K, g_{Na}, g_L in $mScm^{-2}$
f) n', m', h' vs Time	t in ms	n'(t), m'(t), h'(t)
g) Charge Density vs. Time	t in ms	q in $nAscm^{-2}$
h) n^4, m^3h vs. Time	t in ms	$n^4(t)$, $m^3h(t)$
i) Steady State Values vs. Membrane Potential	U_M in mV	$n_\infty(U_M)$, $m_\infty(U_M)$, $h_\infty(U_M)$
j) Time Constant vs. Membrane Potential	U_M in mV	$\tau_n(U_M)$, $\tau_m(U_M)$, $\tau_h(U_M)$

Scaling of the charts is supported by a table that feeds a macro, which sets the new scaling parameters upon each change in the table. Charts can be easily added to the scaling helper.

1.3. Data Tables Sheet

This is the place where the ODEs are implemented. All data calculated by the model and the respective cell formulas are accessible in this sheet. The student can create additional evaluation tables (and respective charts in the cockpit sheet) by referring to the tables in this sheet.

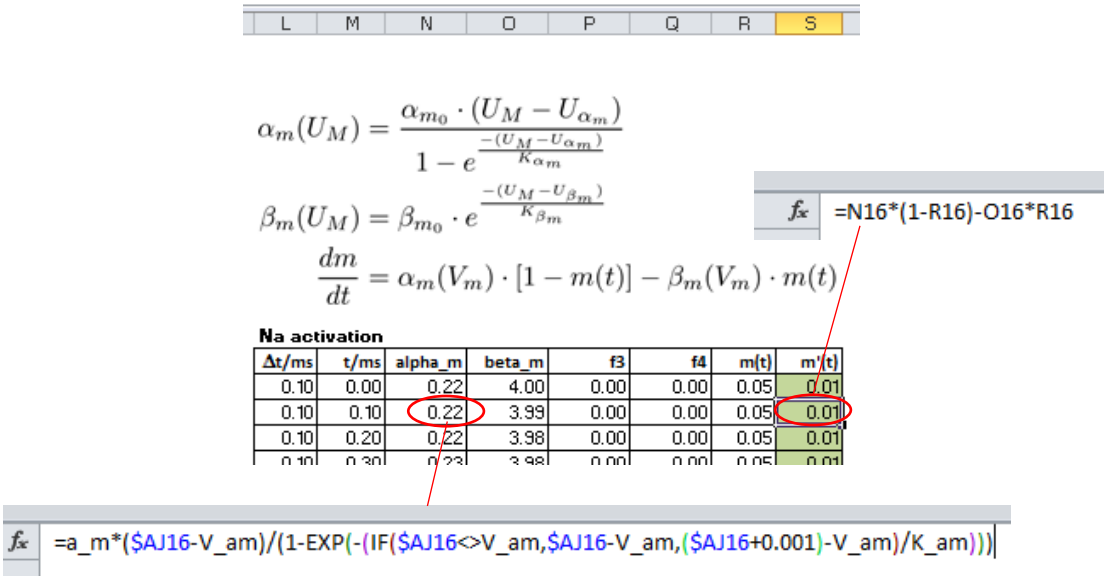


Figure 2:

Screenshot of the table used for solving equation (3). Columns f3 and f4 are not used but required for calling the function RK_4. Columns alpha_m and beta_m couple m(t) to eq. (1). Cell $AJ16 provides the membrane potential $U_M(0.1ms)$. The IF() statement avoids divergence of $\propto_m (U_M)$ in case of $U_M = U_{\propto m}$ without introducing a significant error.

The functions defining the ODE are entered as ordinary spreadsheet cell formulas into the green fields, using standard relative cell references. Each of the equations (1) - (4) has its own table range. Above each of the four tables the respective equations are attached such that the student can easily compare with the spreadsheet formulas (Figure 2).

1.4. Implementation of the Runge- Kutta method

The Runge- Kutta algorithm [8,9] is implemented in a few lines of Visual Basic for Applications (VBA) code. It is not necessary to incorporate the differential equations (1-4) into this code. Just one generic VBA code is used for all ODEs. The function `RK_4(…)` is used in the Data Tables sheet.

```
' RK_4
' fourth-order Runge- Kutta method for solving a system of up to four
' coupled and/or externally driven ODEs

' parameters are:
' h – incremental step size of independent variable
' t – independent variable
' f1, f2, f3, f4 – up to 4 function values involved in defining the ODE,
either
' providing coupling to another ODE or external input
' f – function that is to be calculated by the ODE f'(t) = g(t, f, f_i(t))
' g – Range parameter, referencing the cell that contains g(t, f, f_i(t)) as
' spreadsheet formula; formula is evaluated by the function CalcFormula4

' implementation follows https://wiki.zum.de/images/7/7b/Runge-Kutta.pdf
' prerequisite: all parameter values of RK_4 are taken from one
' Excel table; the table columns contain the parameters from left to right
' in the order they are provided in the function call

Public Function RK_4 _
( _
    ByVal h As Double, _
    ByVal t As Double, _
    ByVal f1 As Double, _
    ByVal f2 As Double, _
    ByVal f3 As Double, _
    ByVal f4 As Double, _
    ByVal f As Double, _
    ByRef g As Range _
) _
As Double

' local slopes used by the fourth order Runge- Kutta method
    Dim k_1 As Double
    Dim k_2 As Double
    Dim k_3 As Double
    Dim k_4 As Double

' approximations of f(t+h)
    Dim y_2 As Double
    Dim y_3 As Double
    Dim y_4 As Double
```

```
' first slope
    k_1 = g.Value
' second approximation (first approximation is f)
    y_2 = f + h / 2 * k_1
' second slope
    k_2 = CalcFormula4(t + h / 2, f1, f2, f3, f4, y_2, g)
' third approximation
    y_3 = f + h / 2 * k_2
' third slope
    k_3 = CalcFormula4(t + h / 2, f1, f2, f3, f4, y_3, g)
' fourth approximation
    y_4 = f + h * k_3
' fourth slope
    k_4 = CalcFormula4(t + h, f1, f2, f3, f4, y_4, g)
' result: f(t+h) as weighted average of slopes k_1 to k_4
    RK_4 = f + h / 6 * (k_1 + 2 * k_2 + 2 * k_3 + k_4)
End Function
```

Figure 3 shows the function call of `RK_4` in the Cockpit sheet.

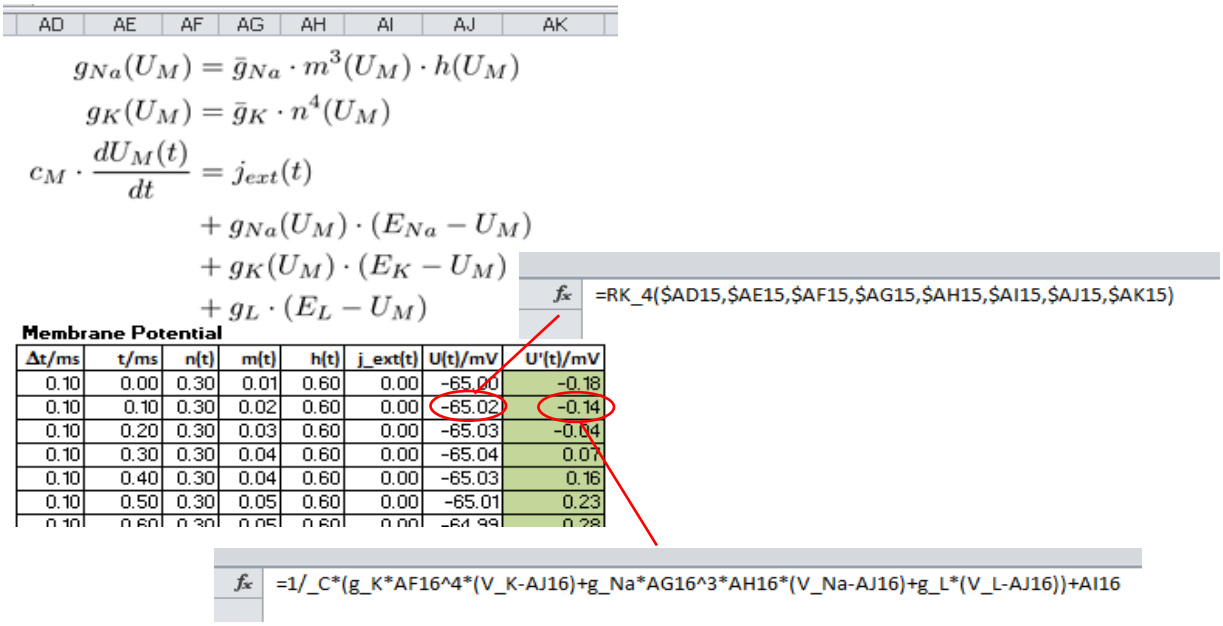


Figure 3:

Screenshot of the table used for solving equation (1), showing how RK_4 is called and the resulting value used for the next iteration.

A VBA function `CalcFormula4` reads the formulas as `String` values, transforms them to VBA compatible expressions and evaluates them. Using this function avoids implementing the equations (1)-(4) by VBA code.

```
' CalcFormula4
' this function extracts cell formulas as Strings, replaces the references
' to Excel cells by their actual values, and evaluates the resulting
' Strings to Double values

' parameters are
' t - time
' f1, f2, f3, f4 - up to 4 function values involved in
  defining the ODE
' f - function that defines the ODE f'(t) = g(t, f, f_i(t))
' g - Range parameter, referencing the cell that contains g(t, f, f_i(t))
```

```
Public Function CalcFormula4 _
(
      ByVal t As Double, _
      ByVal f1 As Double, _
      ByVal f2 As Double, _
      ByVal f3 As Double, _
      ByVal f4 As Double, _
      ByVal f As Double, _
      ByRef g As Range _
) As Double

      Dim strFormula As String

' read formula from Range g as String and remove the '='
      strFormula = Mid(g.FormulaR1C1, 2)

' replace references to t, fi, f by their actual values
' prerequisite: all parameter values of CalcFormula4 are taken from one
' Excel table; the table columns contain the parameters from left to right
' in the order they are provided in the function call

      strFormula = Replace(strFormula, "RC[-6]", CStr(t))
      strFormula = Replace(strFormula, "RC[-5]", CStr(f1))
      strFormula = Replace(strFormula, "RC[-4]", CStr(f2))
      strFormula = Replace(strFormula, "RC[-3]", CStr(f3))
      strFormula = Replace(strFormula, "RC[-2]", CStr(f4))
      strFormula = Replace(strFormula, "RC[-1]", CStr(f))

' evaluate the formula String and return the result

      CalcFormula4 = g.Worksheet.Evaluate(strFormula)
End Function
```

3. Example: Firing an Action Potential

Figure 4 shows the response of the model when stimulated by three different external current injections, each for 5ms. $j_{ext}=3\mu A/cm^2$ only causes a passive response of the membrane potential, $j=6\mu A/cm^2$ triggers an action potential which is only marginally higher for a ten times larger stimulus of $j=60nA/cm^2$.

Zoom-ins of the charts in Figure 5 allow to investigate more details and to relate the courses of the (in)activation functions and their derivatives, the current and charge densities, and the K^+ and Na^+ conductances to the action potential. Especially the different dynamics of the underlying processes are visible and can help to understand the "undershoot phase" of the action potential.

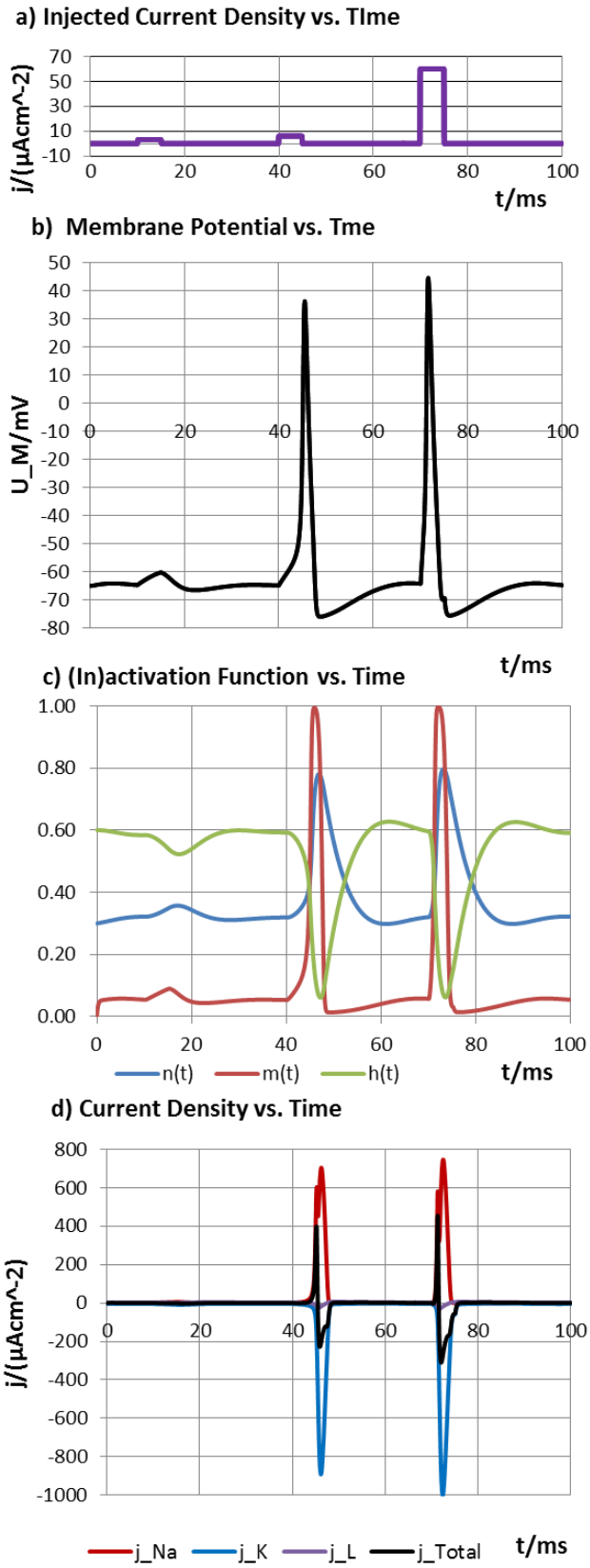
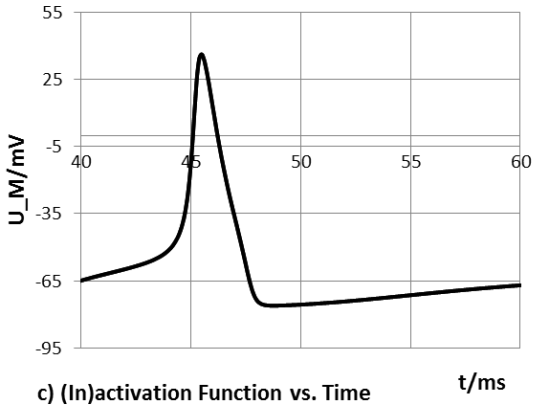


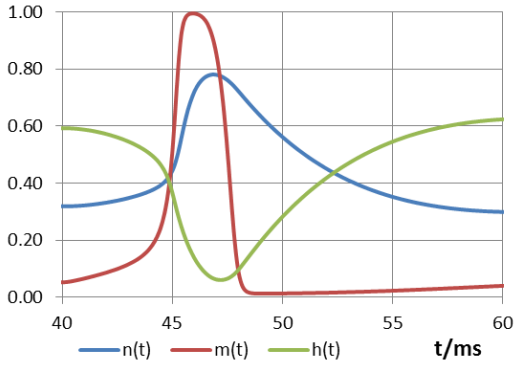
Figure 4:

Response of the model to three current density injections, each for 5ms. j=3µA/cm² only causes a passive response of the membrane potential, j=6µA/cm² triggers an action potential which is only marginally higher for a ten times larger stimulus of j=60µA/cm².

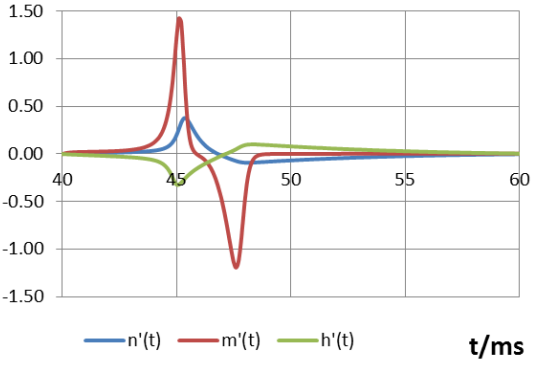
b) Membrane Potential vs. Tme



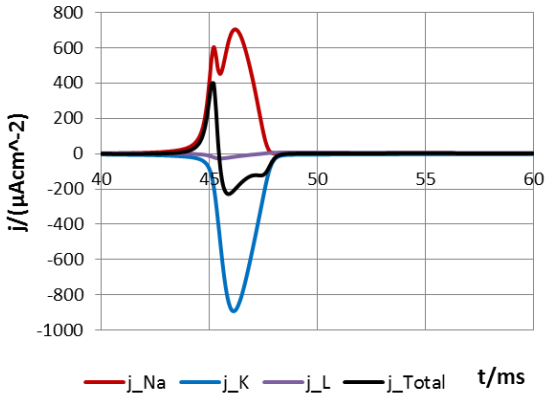
c) (In)activation Function vs. Time



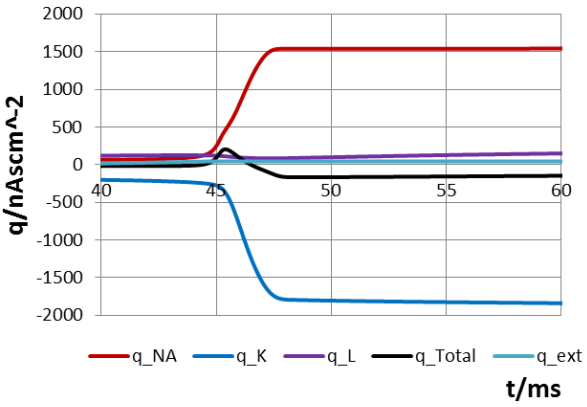
f) n', m', h' vs. Time



d) Current Density vs. Time



g) Charge Density vs. Time



e) K-, Na-, and Leak Conductance vs. Time

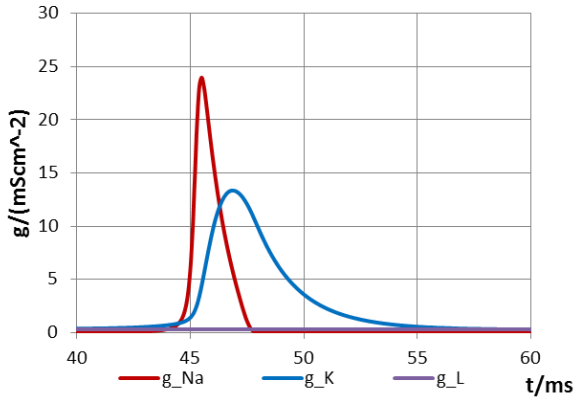


Figure 5:

Zoom-in of Figure 4, showing detailed traces of the (in)activation functions and their derivatives, current and charge densities and ion conductances per cm^2.

4. References

[1] Hodgkin, A. L. und A. F. Huxley. (1952a) *A Quantitative Description of Membrane Current and its Application to Conduction and Excitation in Nerve*. In: J. Physiol.117, p. 500–544.

[2] Purves, D. (2012) *Neuroscience* 5th Edition. Sinauer Associates, Inc.

[3] Brown, A. (1999) *A methodology for simulating biological systems using Microsoft Excel.* Comp. Methods Progr. Biomed. 58, p. 181-190

[4] Brown, A. (2000) *Simulation of axonal excitability using a Spreadsheet template created in Microsoft Excel.* Comp. Methods Progr. Biomed. 63, p. 47-54

[5] Kenyon, *J. How to Solve and Program the Hodgkin – Huxley Equations.* http.//www.imt.liu.se/edu/courses/TBMT01/pdf_files/HH_Kenyon.pdf (visited on 05/25/2017)

[6] https://neurowiki.case.edu/JSNeuroSim/simulations/simple_currentclamp_jqplot.html (visited on 05/25/2017)

[7] http://www.biosoft.com/w/neurosim.htm (visited on 05/25/2017).

[8] Erik Sheever. (2005-2015) *Fourth Order Runge- Kutta* http://lpsa.swarthmore.edu/NumInt/NumIntFourth.html (visited on 03/25/2017).

[9] Fendt, W. (2011) *Das klassische (vierstufige) Runge-Kutta-Verfahren*. https://wiki.zum.de/wiki/Datei:RungeKutta.pdf (visited on 03/03/2017).