Fast and Accurate Explicit Integration Scheme for Biologically Detailed Neuron Simulations

MARGARITA SASS-TISOVSKAYA

KTH Computer Science and Communication

Master of Science Thesis
Stockholm, Sweden 2006
Fast and Accurate Explicit Integration Scheme for Biologically Detailed Neuron Simulations

MARGARITA SASS-TISOVSKAYA

Master’s Thesis in Numerical Analysis (20 credits) at the Scientific Computing International Master Program Royal Institute of Technology year 2006
Supervisor at CSC was Alexander Kozlov
Examiner was Axel Ruhe

TRITA-CSC-E 2006:055
ISRN-KTH/CSC/E--06/055--SE
ISSN-1653-5715

Royal Institute of Technology
School of Computer Science and Communication
KTH CSC
SE-100 44 Stockholm, Sweden
URL: www.csc.kth.se
Abstract

Efficiency of numerical integration methods for a biologically detailed neuron model is examined. Because of the wide range of time constants, neuron equations are considered stiff. Stiffness is a property of a system in which the time step required to maintain stability for an explicit method is much smaller than the time step required to achieve a desired level of accuracy. Traditional explicit numerical methods optimised for non-stiff problems can be quite slow for the task. On the other hand, using implicit integration schemes designed to deal with extreme stiffness can be inefficient too. We investigate performance of an explicit Runge-Kutta Chebyshev (RKC) integration scheme, which is stabilized adaptively through a series of additional stages, for solving neuron equations. The method is compared to the Trapezoidal method (TR), a method currently implemented in several public domain neuron simulators. A comparison of the two numerical integration methods addresses: scalability with respect to CPU time and accuracy.
Acknowledgments

First and foremost I would like to thank my supervisor Alexander Kozlov for his constant support and guidance through my thesis. Special thanks to Lennart Edsberg, for being an excellent coordinator of the International Master programme, along with numerous help in practical details.

Finally, I want to thank my parents and my sister for their infinite belief in my success. Last, but not least, I thank my husband Alexei for his moral support, rewarding discussions and advice.
Contents

1 Introduction 1
   1.1 Neuromodeling Background .............................. 1
   1.2 Formulation of the problem ............................. 3

2 Numerical Preliminaries 5
   2.1 Basic concepts .......................................... 5
   2.2 Stiff problems ......................................... 6

3 Model description and numerical methods 9
   3.1 Test model ............................................... 9
   3.2 The difference schemes ................................. 14

4 Numerical Results 17
   4.1 Numerical solution ..................................... 17
   4.2 Efficiency for the mildly stiff and stiff neuron models 18

5 Conclusions 25

Bibliography 27
Chapter 1

Introduction

Neuroscience is a collection of disciplines unified by a concern for the function of the brain. Experimental approaches in neural science vary from analysis of molecular and cellular mechanisms in nerve cells to behavioral and psychological studies of whole organisms. Theoretical tools include mathematical and computational modeling approaches that have proved useful in other areas of science. Experimental questions include issues related to biophysical and neurochemical mechanisms within single nerve cells, functional neural circuits consisting of small numbers of neurons, the behavior of large systems of neurons, and the relationship between the activity of elements of the nervous system and the behavior of organisms.

1.1 Neuromodeling Background

Neuron is a nerve cell that sends and receives electrical signals. Many highly specialized types of neurons exist:

- Afferent neurons convey information from tissues and organs into the central nervous system.
- Efferent neurons transmit signals from the central nervous system to the effector cells.
- Interneurons connect neurons within the central nervous system.

and these differ widely in appearance. Characteristically, neurons are highly complex in shape. Typically neurons consist of soma, dendrites and axon, see Figure (1.1(a)).

Soma is the relatively large central part of the cell. The word "soma" comes from a Greek word meaning "body" and the soma of a neuron is often called the "cell body". Since there are many different specialized types of neurons, the size of the soma can range from about 5 micrometers to over 1 millimeter for some of the largest neurons of invertebrates.

Dendrite is a branching arbor of cellular extensions. The term "dendrite" comes from the Greek word dendron, meaning "tree". Each neuron has very many dendrites with a profuse dendritic branches which conduct the electrical stimulation received from other cells to the soma. These structures form the main information receiving network for the neuron.

Axon is a fine, cable-like projection which may extend tens, hundreds, or even tens of thousands of times the diameter of the soma in length. This is the structure which carries nerve signals away from the neuron. Each neuron has only one axon, but this axon may undergo extensive branching and thereby enable communication with many target cells.
CHAPTER 1. INTRODUCTION

(a) An example of a typical neuron consisting of a soma, dendrites and an axon

(b) A simplified compartmental model

Figure 1.1: General neuron structure and its simplified compartment model.

To simulate the neurons the following family of methods are usually employed:

- The *detailed compartmental model* is a standard approach where we divide the neuron into a finite number of interconnected compartments. In Fig. (1.1(b)) we show a simplified model in which the neuron is divided into several dendrite compartments, a soma and an axon. Each compartment and its interactions with the neighbors is then modeled with the appropriate differential equations describing an equivalent electrical circuit.

- The *equivalent cylinder model* can be regarded as a simplification of the *detailed compartmental model*, and it is used for those problems, when it is adequate to model neurons with a smaller number of non-equipotential compartments. In other words, under certain conditions, a complicated structure of the neuron can be approximated by a much simpler linear dendrite model. Unfortunately, as the complexity of the neuron increases, it is usually necessary to expand the model, thus reverting to the detailed compartment model.

- In the cases when the computer resources are limited, the *single compartment model* is used. Namely, the whole neuron is described by a very small number of compartments. On the other hand, for some particular problems only the *single compartment model* is needed since it manage to capture the essential behavior of the problem. However, one must always be aware that there are many local "computations" that are skipped by this model.

An equivalent electrical circuit is used to model the behavior of each separate compartment. This arises from the fact that the neuronal membranes have been demonstrated to behave as simple electrical circuits with some capacitance, resistance and the voltage sources. These model parameters define the so-called passive properties that are responsible for the way that electrical impulses are transmitted along the dendritic tree. It is usual practice to begin all single cell modeling efforts with a consideration of passive cellular properties of the cell. These properties form the
basis for the usually more interesting behavior that arises from the active properties provided by different voltage or ligand-dependent conductances.

1.2 Formulation of the problem

In this thesis we consider a simplified model neuron from the spinal cord of lamprey. This neuron consists only of the soma and dendrites attached to the soma in a line. The equations describing this model in the simplest case were first developed by Hodgkin and Huxley [1]. Conservation of current at the node on the inside of the cell gives

\[ C \frac{dV}{dt} = I_{\text{stim}} + G_{\text{leak}}(E_{\text{leak}} - V) + G_{\text{Na}} m^3 h (E_{\text{Na}} - V) + G_{\text{K}} n^4 (E_{\text{K}} - V) + G_{\text{core}} (U_1 - V) \]  (1.1)

That is, the current through the capacitance is the externally applied current minus the currents through the ion channels. In the equation above, \( C \) represents the membrane capacitance, \( G_{\text{Na}} \) is the sodium conductance, \( G_{\text{K}} \) stands for the potassium conductance, \( G_{\text{leak}} \) is the leakage conductance and \( G_{\text{core}} \) is the coupling conductance between the membrane and the first dendrite compartment. The membrane potential \( V \) is the potential inside the cell minus the potential outside and there can be a current \( I_{\text{stim}} \) injected into the cell from the other parts of the cell. Consistent with the usual convention, currents are positive in the inward direction for positively charged ions. To model the gating behavior of the sodium and potassium channels we use the following equations for the rate functions \( m, h \) and \( n \):

\[
\frac{dm}{dt} = (1 - m) \lambda_m(V) - m \beta_m(V), \\
\frac{dh}{dt} = (1 - h) \lambda_h(V) - h \beta_h(V), \\
\frac{dn}{dt} = (1 - n) \lambda_n(V) - m \beta_n(V),
\]  (1.2)

provided in [1]. In eq. (1.2) the sodium (Na) channel has two sets of gates, activation gates, represented by \( m \), and inactivation gates, represented by \( h \). The activation gates open and the inactivation gates close when the membrane depolarizes. The potassium (K) channel has only a single activation variable \( n \). The potential on the \( i \)-th dendrite compartment \( U_i \) is given by

\[ C \frac{dU_i}{dt} = G_{\text{leak}D_i}(E_{\text{leak}} - U_i) + G_{\text{core}D_i}(U_{i+1} - U_i) + G_{\text{core}D_{i-1}}(U_{i-1} - U_i) \quad \text{for} \quad i = 1, \ldots, N_{\text{dend}}. \]  (1.3)

where the neighboring dendrites \( D_i \) and \( D_{i+1} \) are electrically coupled by conductance \( G_{\text{core}D_i} \) and \( G_{\text{leak}D_i} \) is the leakage conductance on \( D_i \), see Chapter 3 for more details on the model.

This model (eq.’s (1.1), (1.2) and (1.3)) is relatively complex, and so provide a realistic and a challenging system to be solved numerically. Moreover, the model with increasing \( N_{\text{dend}} \) leads to a class of problems, known as stiff systems, see Chapter 2 for details. As a rule of thumb in such case implicit schemes are used for the numerical computations. Although, in general, one time step of the implicit scheme is more expensive (one has to solve a nonlinear problem) than that of the explicit one, implicit schemes are often preferable. The reason is that the magnitude of the time step for the implicit scheme is not constrained by the stability considerations, as it is for the explicit scheme when dealing with the stiff problems.
In this thesis, we consider an alternative approach. An explicit Runge-Kutta Chebyshev method (RKC) that possess an appropriately extended region of the absolute stability is applied to solving typical neuron equations. For the derivation and stability analysis of this stabilized method we need to assume that the eigenvalues of the problem ((1.1), (1.2) and (1.3))) lie in a long narrow strip along the negative axis of the complex plane, which is the case for our model.

The advantage of the RKC method, when compared to the implicit or semi-implicit methods, is that they do not require the expensive solution of complicated nonlinear system. Additionally, this method can be easily applied to large problems with only a low memory demand. It is the purpose of this thesis to apply RKC method to our model problem (eq.’s (1.1), (1.2) and (1.3)) and numerically investigate the behavior of RKC scheme for different regions of stiffness. We also numerically compare RKC to the most commonly used schemes for our model: the implicit Trapezoidal (TR) and explicit Euler methods.

To measure the efficiency of the methods we compare the CPU-time spent to achieve a particular error threshold by all methods. These tests were performed for different choices of those parameters, that affect the magnitude of the stiffness of the model. Those are the number of the dendrites and the conductance between the compartments ($G_{coreD_i}$).

This paper is organized in the following way. In Chapter 2 we give the numerical preliminaries, discuss such notions as error, stability and stiffness. In Chapter 3 we give a detailed description of our test model, present the commonly used methods and introduce the explicit Runge-Kutta Chebyshev (RKC) method. The numerical results and their analysis is given in Chapter 4, followed by the conclusion and the ideas for the future work.
Chapter 2

Numerical Preliminaries

In this chapter we discuss some of the basic concepts associated with the numerical methods. We give a brief review of such notions as stability, consistency and convergence. We also give a short discussion on stiffness for linear and nonlinear problems.

2.1 Basic concepts

The numerical methods for ODE’s involve two kinds of errors. The first one is usually called a round-off error, and is associated with the finite nature of the computers. That is the finite precision representation of all numbers in the machine. The second one, referred as truncation or discretization error, comes from the finite approximations of the infinite limiting processes that define derivatives and integrals.

In order to analyze these errors, three concepts were introduced: stability, consistency and convergence. The most fundamental is convergence which means that the difference between the numerical solution and the analytical solution can be made as small as required. Thus demonstrating convergence means showing that the numerical solution deviates from the exact solution by a term which goes to zero as the discretization is refined.

As the name implies, consistency of a numerical method ensures that the numerical solution solves the discrete problem that is the same as desired continuous problem. This usually amounts to determine whether the difference equations, when applied to the analytical solution to the continuous problem, produce only a small truncation error, that goes to zero as the discretization is refined. This definition of the consistency seems very similar to the definition of the convergence. However, a method can be consistent but not convergent. This is due to the fact that the consistency only demands that the exact solution satisfy the difference approximation of the continuous problem with the decreasing truncation error. Convergence demands that the numerical and the exact solutions can be made to differ by an arbitrary small amount. Thus convergence is a more restrictive concept.

Stability is the concept that fills the gap between consistency and convergence. We say that the numerical method is stable if the solution to the difference equations remains bounded as the discretization of the problem is refined. In the case when the consistent numerical method is not stable, the numerical solution grow without bound even when the analytical solution to the continuous problem might actually be quite small. It is clear, that the numerical method of this type is not convergent. Finally, we merely state one of the most remarkable results in the analysis of the difference methods for the differential equations, the Lax Equivalence Theorem, which states that a finite difference method for linear ODE’s is convergent if and
only if it is both consistent and stable. Because of this elegant relationship between these three concepts for linear problems, numerical methods for nonlinear problems also discuss consistency and the stability in the context of establishing numerical convergence [8].

**Numerical Methods for ODE’s**

The theory for the numerical solution of ODEs is very well established, and the rigorous analysis of many classes of numerical methods has an extensive literature, see, for example, [4], [6]. We will distinguish numerical methods for ODEs based on a property of ODEs themselves, known as stiffness. We do not intend to give a precise definition of the notion here. The term was first used by Curtiss and Hirschfelder in 1952 [3], but it has been refined several times. It is perhaps true that a precise definition of stiffness is not crucial for practical purposes.

For now we just state that stiffness measures the difficulty of solving an ODE numerically, in much the same way as the condition number of a matrix measures the difficulty of numerically solving associated system of linear equations. Stiff systems are characterized by very different time scale. Nonstiff systems can be solved by explicit methods which are relatively simple, while stiff systems require more complex implicit methods. Moreover, one will typically observe that the explicit scheme will need to use an extremely small time steps in order to compute a stable solution to the stiff problem. The natural question is then: Will the increase in stepsize obtained with an appropriate implicit scheme compensate for the additional cost involved per step? In other words, is the implicit scheme cheaper per unit time than the explicit scheme?

Methods can also be classified by how the accuracy depends on the step size, $\Delta t$, usually expressed in “big O” notation. For example, a method is $O((\Delta t)^2)$ accurate if the solution differs from the exact solution to the ODE by an amount that goes to zero like $(\Delta t)^2$, as $\Delta t$ goes to zero. Thus if we half $\Delta t$ the error decreases by a factor of 4. Higher order methods are more accurate, but generally are more complicated to implement than lower order methods and require more work per time step. Thus a particular decision must be made weighing the numerical accuracy requirement versus the overall cost of both implementing and using a particular method.

A more precise analysis leads to a distinction between local and global truncation errors. The local truncation error is the error between the numerical solution and the exact solution after a single time step. Recalling the definition of consistency, it is the local truncation error that must go to zero as $\Delta t$ goes to zero for the method to be consistent. One can generally compute the local truncation error from the Taylor series expansion of the solution. A method is called $p$-order accurate if the local truncation error is $O((\Delta t)^p)$. The global truncation error is the difference between the computed solution and exact solution at a given time, $t = T = n\Delta t$. This is the error that must go to zero as $\Delta t$ goes to zero for a method to be convergent. In general, one cannot merely use the Taylor series to calculate the global truncation error explicitly but it can be shown that if the local truncation error is $O((\Delta t)^p)$ then the global truncation error of a convergent method will also be $O((\Delta t)^p)$.

### 2.2 Stiff problems

It is not an easy task to determine a priori whether the problem at hand is stiff or not. Usually, the degree of stiffness can be established by carefully analyzing the differential equation, taking into account the size of the tolerance to be used, the integration interval, the initial values and possibly other effects that will influence the stiffness.
2.2. STIFF PROBLEMS

Traditionally, a linear system of equations $Ax = b$ of size $n$ is said to be stiff if

$$\max_{1 \leq i \leq n} |Re(\lambda_i)| \gg \min_{1 \leq i \leq n} |Re(\lambda_i)|$$

(2.1)

with

$$Re(\lambda_i) < 0, \quad 1 \leq i \leq n$$

(2.2)

where $\lambda_i$ are the eigenvalues of the Jacobian of the system. Then, quantitatively the stiffness of the system $R$ is given by the ratio

$$R = \frac{\max_{1 \leq i \leq n} |Re(\lambda_i)|}{\min_{1 \leq i \leq n} |Re(\lambda_i)|}$$

(2.3)

By this definition, a stiff problem has a stable fixed point with eigenvalues of greatly different magnitudes. Unfortunately, this simple definition of stiffness is not valid for nonlinear systems. It is based on the linear model problem, and the eigenvalues $\lambda_i$ above pertain to a linear system. Moreover, one should note that the stiffness ratio is often not a good measure of stiffness even in linear systems, since if the minimum eigenvalue approaches zero, the problem has infinite stiffness ratio, but may not be stiff at all if the other eigenvalues are of moderate size.

We shall adopt a verbal definition of stiffness, valid for both linear and nonlinear problems, that is similar to Lambert’s [6]: “If a numerical method is forced to use, in a certain interval of integration, a step length which is excessively small in relation to the smoothness of the exact solution in that interval, then the problem is said to be stiff in that interval. In other words, if the size of the timestep is dictated by the boundary of the stability region of the numerical method in use, instead of the accuracy requirements, the problem is said to be stiff”.

Unlike the linear definition of stiffness, our definition allows a single equation, not just a system of equations, to be stiff. It also allows a problem to be stiff ‘in parts’: a nonlinear problem may start off non stiff and become stiff, or vice versa. It may even have alternating stiff and non stiff intervals.

We stress again that stiffness depends, in addition to the differential equation itself, on

- the accuracy criterion,
- the length of the interval of integration, and
- the region of absolute stability of the method.

Our multi-compartments model problem (eq.’s (1.1), (1.2) and (1.3)) is generally stiff, with the stiffness increasing with the number of compartments. For the practical reasons, we will use the spectral radius of the system

$$\rho(A) = \max_{1 \leq i \leq n} |\lambda_i|$$

as a measure of stiffness.
Chapter 3

Model description and numerical methods

In this chapter we introduce in detail the compartmental neuron model. We start with the description of the compartments that are used, continue to an equivalent electrical circuits and, finally, set up our system of equations. This is followed by the description of the most commonly used methods by neuronal modelers. Namely, explicit Euler method and implicit Trapezoidal method. Finally, we introduce an explicit stabilized second order Runge-Kutta-Chebyshev (RKC) method.

3.1 Test model

Compartment description

In the conceptual model, we do not take into account the detailed dendritic branching pattern of the different neurons. However, to allow for synaptic input at different distances from the soma, a simple representation of the dendritic tree is included. Like in many similar models, the neuron is represented by a set of electrically coupled compartments.

The first compartment represents the soma while the rest represent dendrites at various distances from the soma, see Figure 3.1. In principle, any number of compartments could be used in this model. The primary state variable in each compartment is the intracellular potential $V$, which is computed by calculating the ionic currents flowing into and out from the compartment. This includes the passive coupling of the compartments as well as active ion channels of different types.

![Simplified model](image-url)

Figure 3.1: Simplified model

9
Electrical circuit description

Each compartment has a membrane capacitance $C$ proportional to the membrane area, Figure 3.2.

\begin{equation}
\frac{dV}{dt} = (E_{\text{leak}} - V)G_{\text{leak}} + \sum_{i=1}^{N}(V_i - V)G_{\text{core}} + I
\end{equation}

where $i$ is the number of the compartment. Here, the conductance $G_{\text{leak}}$ models a passive leakage current through the membrane, $I$ is an external stimulus and current through the active ionic channels. This conductance is set so that an appropriate membrane time constant $C/G_{\text{leak}}$ is achieved. The equilibrium potential of this current, $E_{\text{leak}}$, is set close to the resting potential of the cell to be simulated, e.g. $-70\,\text{mV}$, [7].

Further, the compartments are electrically coupled by conductances $G_{\text{core}}$ (Figure 3.2). The summation in the equation above is made over all the adjacent compartments connected to the compartment under study. In the case of a general dendritic tree structure, this would correspond to one compartment in the proximal direction and possibly several compartments connected to the distal end. $V_i$ is the potential in the connected compartment. This passive model thus forms a system of coupled ordinary first order differential equations, with one equation for each compartment.

Finally, the $I$ term models the currents entering through all the active ion channels into the compartment. How these currents are calculated will be described in the next section.

Action Potentials

When a nerve cell is depolarized from the resting membrane potential, different types of voltage gate ion channels will open, provided that the membrane potential reaches a threshold value. Such voltage dependent channels give rise to the action potential, during which sodium channels initially open to bring the membrane potential toward the equilibrium potential for $Na^+$ ions (around $+50\,\text{mV}$). With some delay, voltage depended potassium channels open, which pulls the membrane potential toward the equilibrium potential for $K^+$ ions (around $+90\,\text{mV}$) the sodium channels have the additional property of closing after a period of depolarization, i.e. they are inactivated, whereas the potassium channels are not, [1]. The potassium channels open slower than sodium channels and are not fast enough to disturb the fast depolarization of the spike. They are, however, faster than the inactivation of
sodium channels and therefore constitute the primary factor involved in the rapid re-polarization of the spike. A quantitative model for the sodium and potassium currents was presented by Hodgkin and Huxley as early as 1952, based on a description of the membrane of the squid giant axon. Since then, it has been used for neuronal membranes in general. In this thesis, we have used a modified version of the Hodgkin and Huxley model and adjusted the parameters to match the known characteristics of the real neurons being simulated.

The Hodgkin and Huxley model describes how the mean conductance through the sodium and potassium channels varies with membrane potential and time. It can, however, be interpreted on a single channel level as a two state gate which stochastically changes between an open and closed state.

Figure 3.3: By adding the active ion channels to the model, the electrical equivalent of the soma compartment is extended as in this Figure.

The assumption is that

$$I = I_{stim} + I_{Na} + I_K$$

where $I_{stim}$ is a current injected into the cell from an electrode or from other parts of the cell, $I_{Na}$ is a current through sodium channels and $I_K$ is the current through potassium channels for activation and inactivation. The current through sodium channels, entering the soma compartment, is computed as:

$$I_{Na} = (E_{Na} - V_{soma})G_{Na}m^3h$$

(3.2)

where $E_{Na}$ is the reversal potential for $Na^+$, $G_{Na}$ is the maximum sodium conductance through the membrane and $m$ and $h$ are the degrees of activation and inactivation of the $Na^+$ channels, respectively.

The activation of the $Na^+$ channels is described by a simple differential equation:

$$\frac{dm}{dt} = \alpha_m(1 - m) - \beta_m m$$

(3.3)

where $\alpha_m$ is the rate by which the channels switch from a closed to an open state and $\beta_m$ is the rate for reverse. These rate functions depend only membrane potential in the soma and are given by expressions:

$$\alpha_m = \frac{A(V_{soma} - B)}{1 - e^{(B - V_{soma})/C}} \quad \beta_m = \frac{A(B - V_{soma})}{1 - e^{(V_{soma} - B)/C}}$$
To be able to fit the model equations experimental data, some arbitrary exponents were introduced by Hodgkin and Huxley [1]. One possible explanation for need for such exponents is that the channel molecule actually has to perform several state transitions in order to reach an open state. Note that the parameters $A$, $B$ and $C$ are not necessarily the same for two expressions.

The inactivation of the $Na^+$ channels is described by a similar, but not identical, set of equations:

$$\frac{dh}{dt} = \alpha_h (1 - h) - \beta_h h$$  \hspace{1cm} (3.4)

where

$$\alpha_h = \frac{A(B - V_{soma})}{1 - e^{(V_{soma} - B)/C}} \quad \beta_h = \frac{A(B - V_{soma})}{1 + e^{(B - V_{soma})/C}}$$

The potassium channels are treated in a similar way, except that an inactivation is not included, resulting in:

$$I_K = (E_K - V_{soma})G_K n^4$$ \hspace{1cm} (3.5)

where $E_K$ is the reversal potential for $K^+$, $G_K$ is the potassium conductance and $n$ is the degree of activation of the $K^+$ channels, described by:

$$\frac{dn}{dt} = \alpha_n (1 - n) - \beta_n n$$ \hspace{1cm} (3.6)

with the rate functions $\alpha_n$, $\beta_n$ that are given:

$$\alpha_n = \frac{A(V_{soma} - B)}{1 - e^{(V_{soma} - B)/C}} \quad \beta_n = \frac{A(B - V_{soma})}{1 - e^{(V_{soma} - B)/C}}$$

See Table (3.1) for values of the parameters $A$, $B$, and $C$ for the ion channels used.

**Our model problem**

Summarizing, we arrive at the following model problem, represented by the electrical circuit drawn in Figure 3.4: This circuit is appropriate for simple membrane systems like the squid giant axon or other axonal membranes, where only two voltage-dependent channels are seen. In the model, $C$ represents the membrane capacitance, $G_{Na}$ is a sodium conductance, $G_{core}$ is the conductance between the compartments, $G_K$ is potassium conductance and $G_{leak}$ is a leakage conductance. The membrane capacitance $C$ is given by:

$$C = \frac{G_{soma} + G_{Na} + G_{K} + G_{leak}}{R_{Na}}$$

See Table (3.1) for values of the parameters $A$, $B$, and $C$ for the ion channels used.

![Figure 3.4: Electrical scheme, corresponding to eq.s (3.7),(3.8) and (3.9).](image-url)
3.1. TEST MODEL

potential V is the potential inside the cell minus the potential outside and there can be a current \( I_{\text{stim}} \) injected into the cell from an electrode or from other parts of the cell. Consistent with the usual convention, currents are positive in the inward direction for positively charged ions. This leads us to the following system of ODE’s

\[
C \frac{dV}{dt} = I_{\text{stim}} + G_{\text{leak}}(E_{\text{leak}} - V) + G_{\text{Na}}m^3h(E_{\text{Na}} - V) \\
+ G_Kn^4(E_K - V) + G_{\text{core}}(U_1 - V) \tag{3.7}
\]

where

\[
\begin{align*}
\frac{dm}{dt} &= (1 - m)\lambda_m(V) - m\beta_m(V), \\
\frac{dh}{dt} &= (1 - h)\lambda_h(V) - h\beta_h(V), \\
\frac{dn}{dt} &= (1 - n)\lambda_n(V) - m\beta_n(V),
\end{align*} \tag{3.8}
\]

and

\[
C \frac{dU_i}{dt} = G_{\text{leakD}_i}(E_{\text{leak}} - U_i) + G_{\text{coreD}_i}(U_{i+1} - U_i) \\
+ G_{\text{coreD}_{i-1}}(U_{i-1} - U_i) \quad \text{for} \quad i = 1, ..., N_{\text{dend}} \tag{3.9}
\]

Here, (3.7) and (3.8) represent the soma, and (3.9) describes the potential on the \( i \)th dendrite compartment. In (3.7) the leakage and the coupling conductances are modeled by \( G_{\text{leakD}_i} = G_{\text{leak}}/i \) and \( G_{\text{coreD}_i} = G_{\text{core}}/i^2 \) correspondingly. The currents are given by

\[
I = I_{\text{stim}} + I_{\text{Na}} + I_K
\]

where \( I_{\text{stim}} \) is the current injected into the cell from electrode or from other other part of the cell, \( I_{\text{Na}} \) given by eq. (3.2) and \( I_K \) is described by (3.5). The rate functions \( m, n \) and \( h \) are given by (3.3), (3.6) and (3.4), respectively. Finally, the initial potentials of both soma and dendrites are zero \( (V(0) = U_i(0) = 0 \text{ for } i = 1...N) \) while rate function are \( m(0) = n(0) = 0 \) and \( h(0) = 1 \).

These equations give a model which is capable of producing action potentials with a realistic shape. The parameters used in the model are listed Tables 3.1 and 3.2. It is important to note that this system is a useful example for numerical com-

<table>
<thead>
<tr>
<th>( \alpha )</th>
<th>( Na^+ )</th>
<th>( K^+ )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A ) (mV^{-1} ms^{-1})</td>
<td>0.2</td>
<td>0.008</td>
</tr>
<tr>
<td>( B ) (mV)</td>
<td>-40</td>
<td>-40</td>
</tr>
<tr>
<td>( C ) (mV)</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \beta )</th>
<th>( Na^+ )</th>
<th>( K^+ )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A ) (mV^{-1} ms^{-1})</td>
<td>0.06</td>
<td>0.4 ms^{-1}</td>
</tr>
<tr>
<td>( B ) (mV)</td>
<td>-49</td>
<td>-36</td>
</tr>
<tr>
<td>( C ) (mV)</td>
<td>20</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 3.1: Parameters used in describing the ion channels.

putations. Namely, these equations are basic expressions of current conservation. Thus they can relatively easy accommodate other aspects of the neuronal models. As a result, this nonlinear system models a realistic phenomena and, thus, is a good test for the proposed numerical methods. The numerical methods that work well for this system should work equally well on other equations a neuromodeler may wish to explore.
### Passive Properties

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E_{\text{leak}}$</td>
<td>-70 mV</td>
</tr>
<tr>
<td>$G_{\text{leak}}$</td>
<td>0.003 $\mu$S</td>
</tr>
<tr>
<td>$C$</td>
<td>0.03 nF</td>
</tr>
<tr>
<td>$G_{\text{core}}$</td>
<td>0.04 $\mu$S</td>
</tr>
</tbody>
</table>

### Active Properties

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E_{Na}$</td>
<td>50 mV</td>
</tr>
<tr>
<td>$G_{Na}$</td>
<td>1.0 $\mu$S</td>
</tr>
<tr>
<td>$E_{K}$</td>
<td>-90 mV</td>
</tr>
<tr>
<td>$G_{K}$</td>
<td>0.2 $\mu$S</td>
</tr>
<tr>
<td>$I_{\text{stim}}$</td>
<td>100 $\mu$A/cm$^2$</td>
</tr>
</tbody>
</table>

Table 3.2: Parameters used in the system (3.7), (3.8) and (3.9), [7].

### 3.2 The difference schemes

Define the $N_{\text{dend}} + 4$ dimensional vector $Y = (V, m, h, n, U_1, \ldots U_{N_{\text{dend}}})$ and rewrite the system (3.7), (3.8) and (3.9) as a single vector differential equation

$$
\frac{dY}{dt} = F(Y, t) \tag{3.10}
$$

where $F(Y, t)$ is a vector-valued right hand side corresponding to the right hand side of (3.7), (3.8) and (3.9), and $N_{\text{dend}}$ is the total number of dendrite compartments.

Together, with the initial condition $Y_0 = (0, 0, 1, 0, 0, 0, \ldots)$ eq. (3.10) constitutes an initial value problem (IVP), the solution to which can be uniquely determined.

#### Explicit Euler and Implicit Trapezoidal methods

Denote $t^n = n\Delta t$ and $Y^n = Y(n\Delta t)$. Explicit Euler method or forward Euler method is the simplest and the easiest numerical method to integrate the system. This method is derived from the Taylor series expansion which is

$$
Y(t_n + \Delta t) = Y(t_n) + \Delta t \frac{dY}{dt} \bigg|_{t_n} + O(\Delta t^2)
$$

$$
Y^{n+1} = Y^n + \Delta t F(Y^n, t^n) \tag{3.11}
$$

This is an explicit one-step first order accurate numerical scheme. For stiff systems this method requires a very small time step due to the stability reasons. As a result, one has to do a lot of steps to reach the desired time [2].

As an alternative, one can use a first order implicit version or Backward Euler method

$$
Y^{n+1} = Y^n + \Delta t F(Y^{n+1}, t^{n+1})
$$

We also consider a second order Trapezoidal method which can be viewed as a mean value between Forward and Backward Euler methods

$$
Y^{n+1} = Y^n + \frac{\Delta t}{2} (F(Y^n, t^n) + F(Y^{n+1}, t^{n+1})) \tag{3.12}
$$

For implicit methods the time step is not constrained by the stability considerations but one is required to solve a nonlinear system on every step.

#### Runge-Kutta Chebyshev method

Historically, the principal goal when constructing Runge-Kutta schemes was to achieve the highest possible order of accuracy with a given number of stages $s$. Stabilized methods are different in that only a few stages are used to achieve a
usually low order whereas additional stages are exploited to increase the region of absolute stability, depending on the particular application.

In this thesis we consider an example of stabilized Runge-Kutta Chebyshev method (RKC). For our system (3.10) the RKC method reads

\[ U_0 = Y_n, \]
\[ U_1 = U_0 + \mu_1 h F_0, \]
\[ U_j = (1 - \mu_j - \nu_j) U_0 + \mu_j U_{j-1} - \nu_j U_{j-2} + \bar{\mu}_j h F_{j-1} + \bar{\nu}_j h F_0 \]
\[ Y_{n+1} = U_s \]

where \( F_k \) denotes \( F(t_n + c_j h, U_k) \) and \( j = 2, ..., s \). All coefficients are available in analytical form for arbitrary \( s \geq 2 \). They are defined as follows. Let \( T_j \) be the Chebyshev polynomial of the first kind defined by the following three-term recursion

\[ T_j(x) = 2x T_{j-1}(x) - T_{j-2}(x), \quad j = 2, 3, ..., s, \]

with \( T_0(x) = 1, \ T_1(x) = x \). Then

\[ w_0 = 1 + \epsilon/s^2, \quad w_1 = \frac{T'_1(w_0)}{T''_1(w_0)}, \quad b_j = \frac{T''_j(w_0)}{(T'_j(w_0))^2}, \quad b_0 = b_1 = b_2 \]
\[ \bar{\mu}_1 = b_1 \omega_1, \quad \mu_j = \frac{2b_j \omega_j}{b_j - 1}, \quad \nu_j = -\frac{b_j}{b_j - 1}, \quad \bar{\nu}_j = -\frac{(1 - b_j - 1 T_{j-1}(\omega_0)) \bar{\mu}_j}{b_j - 1}, \quad (2 \leq j \leq s). \]

In (3.13) the stage \( F_j = F(t_n + c_j h, U_j) \) has \( c_j \)

\[ c_0 = 0, \quad c_1 = \frac{c_2}{4 \omega_0}, \quad c_j = \frac{T''_j(\omega_j)}{T''_j(\omega_0)}, \quad \bar{\mu}_j = \frac{\beta T''_j(\omega_j) - \beta T''_j(\omega_0)}{T'_j(\omega_0)}, \quad j \leq 2 \leq j \leq s - 1, \quad c_s = 1. \]

When applied to the scalar stability test equation \( Y' = \lambda Y \), we get at each stage a relation \( U_j = P_j(z) U_0 \) with \( z = h \lambda \) and \( P_j(z) \) a polynomial of degree \( j \) in \( z \) with \( P_1(z) \) as stability polynomial. Formula (3.13) has in fact been derived from a particular set of functions \( P_j(z) \) (0 ≤ \( j \) ≤ \( s \)) satisfying three design criteria:

(i) nearly optimal step-by-step stability of \( P_s(z) \) for parabolic problem

(ii) internal stability, i.e., controlled round-off accumulation in a single step for \( s \) large,

(iii) second-order consistency of \( P_j(c_j z) \) with respect to \( e^{c_j z} \) for \( j = 2, ..., s \)

Criterion (iii) automatically implies second-order consistency of all \( U_j \) (2 ≤ \( j \) ≤ \( s \)) at \( t = t_n + c_j h \) for general problems \( Y' = F(Y, t) \). The first-stage formula is necessarily first-order consistent being forward Euler with step size \( \bar{\mu}_1 h \).

The chosen functions \( P_j \) based on the Chebyshev polynomials \( T_j(x) \) are given by

\[ P_j(z) = a_j + b_j T_j(\omega_1 + \omega_1 z), \quad a_j = 1 - b_j T_j(\omega_0) \]

Using \( T'_j(1) = s^2, \ T''_j(1) = \frac{3}{5} s^2 (s^2 - 4) \) and \( T''_j(1) = \frac{3}{15} s^2 (s^2 - 4) \), the real stability boundary \( \beta(s) \) can be seen to satisfy

\[ \beta(s) \approx \frac{\omega_0 - 1}{T''_j(\omega_0)} \approx \frac{2}{3} \left(s^2 - 1\right)(1 - \frac{2}{15} \epsilon) \]

where the parameter \( \epsilon \) is called a damping parameter as it gives values of \( P_s(z) \) strictly less than one in the interior of the real stability interval \([-\beta(s), 0]\). Taking \( \epsilon = 2/13 \), we get approximately 0.33 ≤ \( P_s(z) \) ≤ 0.95 in most of the interior of the stability interval and a reduction in the \( \beta(s) \) of about 2% to \( \beta(s) \approx 0.65(s^2 - 1) \) compared to the undamped case \( (\epsilon = 0) \) [5]. Figures 3.5 and 3.6 illustrates the stability region with damping for \( s = 2 \) and \( s = 5 \), respectively. For larger values of \( s \) similar regions exist, extended to the left along the negative real line.
CHAPTER 3. MODEL DESCRIPTION AND NUMERICAL METHODS

Figure 3.5: Stability regions for Chebyshev polynomial $P_2$.

Figure 3.6: Stability regions for the Chebyshev polynomial $P_5$. 
Chapter 4

Numerical Results

In this chapter we present the numerical results for our problem. First, we show that RKC method can be successfully applied to our compartment model. We also numerically compare the explicit RKC method to implicit Trapezoidal method. That is we investigate the performance of the methods for different sets of parameters that affect our model: number of dendrite compartments $N_{dend}$, the coupling conductance $G_{core}$ and the some error tolerance $E_{tol}$. The first two parameters affect the stiffness of the model while the last one is the absolute error tolerance between the reference solution (obtained with the standard *Matlab* routine *ode23s* with stringent tolerance) and our approximate solution.

4.1 Numerical solution

Here, we present the numerical solution of our model problem obtained with the RKC method, see Figures 4.1(a) and 4.1(b).

![Figure 4.1: The intracellular potentials on soma and dendrite with $N_{dend} = 3$](image)

As we can see from the pictures above, the potential on soma exhibits almost periodic behavior, while the dendrite potentials are rapidly decaying. This is a well-known behavior that has been confirmed, for example, in [7]. To investigate the performance of our methods we consider two parameters that affect the most the stiffness of the model. Namely, the number of the dendritic compartments $N_{dend}$ and the coupling conductance $G_{core}$. Note, that we take spectral radius of our model problem as a qualitative measure of the stiffness, see Chapter 2 for the details.
Figure 4.2: The dependence of the spectral radius of the model problem on $N_{dend}$ and $G_{core}$.

The spectral radius is grows asymptotically as $O(G_{core})$ and $O(N_{dend}^2)$ as shown in Figures 4.2(a) and 4.2(b). At this point we should note, that to find the spectral radius for all of these cases we first had to calculate the solution (say with the help of ode23s) and plug it in the non-constant Jacobean of the system. To avoid that one could have used the fact that for our problem the spectral radius behaves like a periodic function, see Figure 4.3. Moreover, we can clearly see that with the increasing number of compartments $N_{dend}$ the oscillations in the spectral radius are damped and it behaves like a constant function for the chosen solution. Thus, it might be possible to estimate the spectral radius once and for all in the beginning of the calculations. But this goes out of the scope of the current thesis work.

4.2 Efficiency for the mildly stiff and stiff neuron models

In this section we numerically compare the implicit Trapezoidal and explicit RKC methods for three different test problem. To measure the efficiency of the methods
we compare the CPU-time spent to achieve a particular error threshold. In the first and second test problems we investigate the performance of our methods for different regimes of stiffness. In the third test problem we look at the CPU-time as a function of the relative tolerance in the error. All test were done in the Matlab environment.

Test Problem 1

Here, we look at the spent CPU time as a function of the absolute error in the solution. This test is performed for three different values of the dendrite compartments \( N_{dend} = 3, 50, 85 \). The rest of the parameters are held fixed. The results at \( t = 0.049 \) with the coupling conductance \( G_{core} = 4.4 \times 10^{-9} \) are reported in Tables 4.1,4.2 and 4.3 (Figures 4.4.4.5 and 4.6) correspondingly.

### Table 4.1: Results for RKC, TR and Euler methods for \( N_{dend} \approx 2.8 \times 10^4 \).

<table>
<thead>
<tr>
<th>RKC method</th>
<th>TR method</th>
<th>Euler method</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 8.01 \times 10^{-4} )</td>
<td>( 9.35 \times 10^{-4} )</td>
<td>( 1.6400 )</td>
</tr>
<tr>
<td>( 2.29 \times 10^{-4} )</td>
<td>( 2.35 \times 10^{-4} )</td>
<td>( 2.9700 )</td>
</tr>
<tr>
<td>( 9.46 \times 10^{-5} )</td>
<td>( 5.85 \times 10^{-5} )</td>
<td>( 5.5200 )</td>
</tr>
<tr>
<td>( 2.61 \times 10^{-5} )</td>
<td>( 1.45 \times 10^{-5} )</td>
<td>( 9.8400 )</td>
</tr>
<tr>
<td>( 4.41 \times 10^{-6} )</td>
<td>( 2.33 \times 10^{-6} )</td>
<td>( 20.9400 )</td>
</tr>
<tr>
<td>( 1.12 \times 10^{-6} )</td>
<td>( 5.84 \times 10^{-7} )</td>
<td>( 41.1700 )</td>
</tr>
<tr>
<td>( 2.81 \times 10^{-7} )</td>
<td>( 1.48 \times 10^{-7} )</td>
<td>( 80.6300 )</td>
</tr>
</tbody>
</table>

### Table 4.2: Results for RKC, TR methods for \( N_{dend} = 50 \) (\( \approx 2.4 \times 10^6 \)).

<table>
<thead>
<tr>
<th>RKC method</th>
<th>TR method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>CPU time</td>
</tr>
<tr>
<td>( 5.9247 \times 10^{-4} )</td>
<td>( 7.94000 )</td>
</tr>
<tr>
<td>( 1.5727 \times 10^{-4} )</td>
<td>( 10.7000 )</td>
</tr>
<tr>
<td>( 4.2249 \times 10^{-5} )</td>
<td>( 14.9000 )</td>
</tr>
<tr>
<td>( 1.1159 \times 10^{-5} )</td>
<td>( 21.3800 )</td>
</tr>
<tr>
<td>( 1.6939 \times 10^{-6} )</td>
<td>( 35.8899 )</td>
</tr>
<tr>
<td>( 2.6358 \times 10^{-7} )</td>
<td>( 50.7899 )</td>
</tr>
<tr>
<td>( 1.2829 \times 10^{-7} )</td>
<td>( 79.5700 )</td>
</tr>
</tbody>
</table>

### Table 4.3: Results for RKC, TR methods for \( N_{dend} = 85 \) (\( \approx 1.2 \times 10^7 \)).

<table>
<thead>
<tr>
<th>RKC method</th>
<th>TR method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>CPU time</td>
</tr>
<tr>
<td>( 1.6802 \times 10^{-4} )</td>
<td>( 0.174100 \times 10^2 )</td>
</tr>
<tr>
<td>( 1.5624 \times 10^{-4} )</td>
<td>( 0.306600 \times 10^2 )</td>
</tr>
<tr>
<td>( 4.1730 \times 10^{-5} )</td>
<td>( 0.430400 \times 10^2 )</td>
</tr>
<tr>
<td>( 1.1028 \times 10^{-5} )</td>
<td>( 0.601700 \times 10^2 )</td>
</tr>
<tr>
<td>( 1.8377 \times 10^{-6} )</td>
<td>( 0.966800 \times 10^2 )</td>
</tr>
<tr>
<td>( 4.7189 \times 10^{-7} )</td>
<td>( 1.351900 \times 10^2 )</td>
</tr>
<tr>
<td>( 1.9031 \times 10^{-8} )</td>
<td>( 3.228100 \times 10^2 )</td>
</tr>
</tbody>
</table>
The error displayed in the tables is the difference between the numerical solution and the reference solution computed with the help of \texttt{ode23s} from Matlab library with $10^{-11}$ tolerance. The CPU time is given in seconds.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure4.4}
\caption{A log-log CPU time versus error for TR method, Euler method and RKC method for model with three dendrites.}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure4.5}
\caption{A log-log CPU time versus error for TR method and RKC method for model with 50 dendrites.}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure4.6}
\caption{A log-log CPU time versus error for TR method and RKC method for model with soma 85 dendrites.}
\end{figure}
4.2. EFFICIENCY FOR THE MILDLY STIFF AND STIFF NEURON MODELS

As we can see from the results, the stiffness of the problem (spectral radius) increases with the number of compartments used. RKC method proved to be superior to Trapezoidal method for the mildly stiff problems. That is, for the problems where the number of the compartments is not too high ($N_{dend}$). At the same time, we observe that for the stiff problems, RKC is still a method of choice if one is interested in a high accuracy of the numerical solution.

Test Problem 2

In this test problem, we fix the number of dendrites $N_{dend} = 3$ and augment the coupling conductance $G_{core}$ thereby increasing the stiffness of the problem. As before, we look at the CPU time as a function of the absolute error in the solution. We consider the solution at $t = 0.049$ for two different values of the coupling conductance $G_{core} = 2.2 \times 10^{-5}$ and $4.4 \times 10^{-4}$, see Tables 4.4 and 4.5 (Figures 4.7 and 4.8), correspondingly.

<table>
<thead>
<tr>
<th>RKC method</th>
<th>TR method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>CPU time</td>
</tr>
<tr>
<td>5.8957 \cdot 10^{-4}</td>
<td>2.740000</td>
</tr>
<tr>
<td>1.5784 \cdot 10^{-5}</td>
<td>3.880000</td>
</tr>
<tr>
<td>4.2387 \cdot 10^{-5}</td>
<td>5.519999</td>
</tr>
<tr>
<td>1.1276 \cdot 10^{-5}</td>
<td>7.519999</td>
</tr>
<tr>
<td>1.7579 \cdot 10^{-6}</td>
<td>12.19999</td>
</tr>
<tr>
<td>2.6400 \cdot 10^{-7}</td>
<td>20.01999</td>
</tr>
<tr>
<td>9.7553 \cdot 10^{-8}</td>
<td>22.41999</td>
</tr>
</tbody>
</table>

Table 4.4: Results for RKC and TR methods with $G_{core} = 2.2 \times 10^{-5}$ (the spectral radius $\rho \approx 2 \times 10^{6}$).

As expected, the method exhibit similar behavior to the previous test problem. Namely, the stiffness of the problem increases with the coupling conductance $G_{core}$.
CHAPTER 4. NUMERICAL RESULTS

<table>
<thead>
<tr>
<th>RKC method</th>
<th>TR method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>CPU time</td>
</tr>
<tr>
<td>5.8491 · 10^{-4}</td>
<td>12.1200</td>
</tr>
<tr>
<td>1.5604 · 10^{-4}</td>
<td>17.1000</td>
</tr>
<tr>
<td>4.1591 · 10^{-5}</td>
<td>24.1600</td>
</tr>
<tr>
<td>1.0953 · 10^{-5}</td>
<td>34.3900</td>
</tr>
<tr>
<td>1.8080 · 10^{-6}</td>
<td>53.3000</td>
</tr>
<tr>
<td>4.5710 · 10^{-7}</td>
<td>76.6100</td>
</tr>
<tr>
<td>1.1422 · 10^{-7}</td>
<td>109.420</td>
</tr>
</tbody>
</table>

Table 4.5: Results for RKC and TR methods for $G_{core} = 4.4 \times 10^{-4}$ (the spectral radius $\rho \approx 4 \times 10^7$).

Figure 4.8: A log-log plot of CPU time versus error for TR and RKC methods for the model with $G_{core} = 4.4 \times 10^{-4}$ (the spectral radius $\rho \approx 4 \times 10^7$).

and the RKC method is superior to the Trapezoidal method if one is interested in the accuracy of the solution.

Test Problem 3

Finally, we look at the CPU time as a function of the dendrite compartments for three different values of the absolute error $E_{tol} = 10^{-4}, 10^{-5}, 10^{-7}$. The rest of the parameters are fixed as in the Test Problem 1.
4.2. EFFICIENCY FOR THE MILDLY STIFF AND STIFF NEURON MODELS

<table>
<thead>
<tr>
<th>RKC method</th>
<th>TR method</th>
<th>Euler method</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_{dend}$</td>
<td>CPU time</td>
<td>$N_{dend}$</td>
</tr>
<tr>
<td>1</td>
<td>0.6499</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0.6900</td>
<td>10</td>
</tr>
<tr>
<td>20</td>
<td>1.0800</td>
<td>20</td>
</tr>
<tr>
<td>30</td>
<td>1.9000</td>
<td>30</td>
</tr>
<tr>
<td>40</td>
<td>3.2000</td>
<td>40</td>
</tr>
<tr>
<td>50</td>
<td>4.5000</td>
<td>50</td>
</tr>
</tbody>
</table>

Table 4.6: Results for RKC, TR and Euler methods for the value of $E_{tol} = 10^{-4}$.

<table>
<thead>
<tr>
<th>RKC method</th>
<th>TR method</th>
<th>Euler method</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_{dend}$</td>
<td>CPU time</td>
<td>$N_{dend}$</td>
</tr>
<tr>
<td>1</td>
<td>1.5300</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>1.5300</td>
<td>10</td>
</tr>
<tr>
<td>20</td>
<td>3.3800</td>
<td>20</td>
</tr>
<tr>
<td>30</td>
<td>6.6100</td>
<td>30</td>
</tr>
<tr>
<td>40</td>
<td>9.9799</td>
<td>40</td>
</tr>
<tr>
<td>50</td>
<td>14.990</td>
<td>50</td>
</tr>
</tbody>
</table>

Table 4.7: Results for RKC, TR and Euler methods for the value of $E_{tol} = 10^{-5}$.

<table>
<thead>
<tr>
<th>RKC method</th>
<th>TR method</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_{dend}$</td>
<td>CPU time</td>
</tr>
<tr>
<td>1</td>
<td>37.0100</td>
</tr>
<tr>
<td>10</td>
<td>38.6900</td>
</tr>
<tr>
<td>20</td>
<td>42.9900</td>
</tr>
<tr>
<td>30</td>
<td>46.8300</td>
</tr>
<tr>
<td>40</td>
<td>51.6000</td>
</tr>
<tr>
<td>50</td>
<td>90.4800</td>
</tr>
</tbody>
</table>

Table 4.8: Results for RKC and TR methods for the value of $E_{tol} = 10^{-7}$.

Figure 4.9: CPU time versus number of dendrites for Euler, Trapezoidal and RKC methods for $E_{tol} = 10^{-4}$. 
These results are consistent to those of the previous test problems. Namely, RKC is the method of choice with regards to the accuracy. Otherwise, for the problems with large number of compartments and low accuracy requirement on the solution, one should use the Trapezoidal method.
Chapter 5

Conclusions

Simulations of large sets of neurons require fast numerical methods. In large neural network simulations computational speed and stability prevail over accuracy for practical reasons which dictates the choice of a numerical solver. The explicit solvers are fast and can be used for non-stiff and usually simple neuron models. The implicit solvers are stable and should be used for highly stiff neuron models with high spatial resolution. While the vast majority of currently used neuron models are supposedly only mildly stiff, there is still no recommended solver for the "gray area".

In this thesis, we have successfully applied an explicit stabilized Runge-Kutta Chebyshev method to our model. To investigate the performance of the RKC scheme, we compare it to the Trapezoidal method, that is one of the most commonly used methods for our model. The numerical experiments show that for the problems with the large number of compartments or for the problems where the parameters were chosen such that the problem becomes vastly stiff, the Trapezoidal method performs slightly better than RKC. Though if one is interested in the highly accurate solution or in the problems that are only mildly stiff, RKC is the method of choice.
Bibliography


