NTNU
Det skapende universitet

Kombinasjonen av eksplisitt og implisitt løser for simulering av den elektriske aktiviteten i hjertet.

Martin Kaarby

Master i fysikk og matematikk
Oppgaven levert: Juni 2007
Hovedveileder: Brynjulf Owren, MATH
Biveileder(e): Anne Kværnø, MATH

Norges teknisk-naturvitenskapelige universitet
Institutt for matematiske fag

Oppgavetekst

I denne oppgaven vil vi se på en ny metode for å simulere den elektriske aktiviteten i hjertet. Arbeidet vil være fokusert på utvikling, analyse og testing av den numeriske metoden. Winslow-modellen vil alltid bli brukt som referanse, slik at den numeriske metoden vil bli optimalisert med hensyn på disse ligningene. Vi ønsker å finne ut om det er hensiktsmessig å løse systemet med en kombinasjon av en eksplisitt og implisitt løser, der vi bruker den eksplisitte løseren i den transiente fasen, og implisitt på resten av løsningen.

Oppgaven gitt: 17. januar 2007
Hovedveileder: Brynjulf Owren, MATH

Abstract

Simulating an ECG signal on a computer could be a great help when trying to understand the relationship between the observed ECG signal and the heart condition. In order to get a realistic simulation, a good mathematical model was suggested by Winslow et al. in 1999, called the Winslow model. This model consists of a set of 31 ODE's (Ordinary Differential Equation)s which describe these electrochemical reactions in a single heart cell. The cost of doing one single call to the Winslow ODE system is so big that a measure of efficiency is almost entirely based on the number of such calls. It is therefore of importance to reduce the number of calls to the ODE system when solving the problem. A closer look at the solution of the Winslow model unveils that it has an initial transient phase where an explicit method is more efficient than an implicit. The idea is therefore to implement a method that is explicit in the beginning, and switches to implicit as the problem becomes stiff. This method has proved to increase the efficiency by decreasing the number of calls to the ODE system with about 25%, while the accuracy is maintained.

Preface

This is my master thesis at the Norwegian University of Science and Technology, Department of Mathematical Sciences, spring 2007. In this thesis I study the Winslow model, which is a system of ordinary differential equations describing the electrochemical reactions in a heart cell. It is a study of how explicit and implicit methods can be combined in order to solve the model as efficient as possible. All results and plots are obtained by computing in MATLAB 7.0.

I want to thank my supervisors Brynjulf Owren and Anne Kværnø for their ability to always see things in a new light and come up with brilliant new ideas when I have been stuck.

Contents

Part I.

Introduction

In this thesis we will solve numerically the set of ordinary differential equations (ODE) suggested by Winslow et al in [4]. This is a system of 31 ODE's describing the electrochemical reactions in a heart cell and is called the Winslow model. The Winslow model is only a part of a larger mathematical model for simulating an ECG signal, but when simulating, most of the time is spent solving the ODE system. So, in order to speed up the simulations it is necessary to find a numerical method that solves the ODE system efficiently.

First we will describe, in Chapter 2, some basic physiology of the heart and how a mathematical model can be derived from this.

In Chapter 3 we build up a theoretical base that will help the reader for the rest of the thesis. This involves some classic Runge-Kutta theory and an explanation of the concept stiffness. Further, we give a short description of the Runge-Kutta solvers to be used throughout this thesis. We then discuss some stiffness detection techniques with explicit Runge-Kutta methods and show how they work in practice.

Chapter 4 is dedicated to step size strategies. Here, we will derive a step size strategy to be used in the explicit solvers when solving the Winslow equations.

At last, in Chapter 5, we wish to take advantage of the results of the previous chapters and combine an explicit solver with an implicit. This involves using both stiffness detection from Chapter 3, and our new step size strategy developed in Chapter 4. The performances of the new combined methods are then compared to the classic solvers.

1. A mathematical model of the heart

1.1. Physiology

The heart is truly one of our most vital organs. Its main task is to pump blood around in the body. It can be separated into four different valves, called the left/right atrium and left/right ventricles.

The human circulatory system is a two-part system whose purpose is to bring oxygen to all the tissues of the body. When the heart contracts it pushes the blood out into two major loops or cycles. In the systemic loop, the blood circulates into the body's systems, bringing oxygen to all its organs, structures and tissues collecting carbon dioxide waste. In the pulmonary loop, the blood circulates to and from the lungs, to release the carbon dioxide and pick up new oxygen. The systemic cycle is controlled by the left side of the heart, the pulmonary cycle by the right side of the heart. See [1] for more.

One of the most common ways to study the heart is by measuring the electric activity in it. This can be measured on the body surface, and the measurement is called an electrocardiogram (ECG). An ECG provides a doctor with a lot of information about the condition of the heart.

By simulating the electrical activity of the heart, the goal is to achieve a better understanding of the relationship between the heart condition and the ECG signal.

The electric signal is created in the heart and conducted[1] through the heart tissue and the surrounding torso. In the mathematical model to be presented here, the exterior of the heart is treated as a passive conductive medium with a given conductance, while in the interior the electric activity is modelled by the bidomain model (Winslow model), proposed by Winslow et al [4] in 1999.

[1] Electrical conductivity is a measure of a material's ability to conduct/lead an electric current

1.2. The bidomain model

To better understand the bidomain model, it is useful to know the basic physiology of the heart muscle tissue and the cells. A cell is delimited by the cell membrane, whose task is to control the flow of substances into and out of the cell. These substances pass the membrane through channels formed by the so called gating proteins. These channels are specialized so that only a narrow choice of substances can pass through a given channel.

The contraction of the heart is triggered by a change of electric potential over the cell membrane. In the bidomain model we distinguish between the intracellular and the extracellular potential, denoted by u_i and u_e. This leads us to the trans membrane potential $v = u_i - u_e$, describing the potential difference in the two domains. At about $-90mV$ this potential is called the resting potential, otherwise it is called an action potential.

The contraction is started in a point in the heart called the sinoatrial node. The special property of this particular point, is that the cells there trigger before all the other cells in a $healthy$ heart, causing a chain reaction of triggering cells. The direction of this reaction, and thus the direction of the electric field, is determined by the orientation of the muscle fibres because the conduction of the electric signal is fastest along the muscle fibres.

To be able to start the contraction of a cell, the cell has to depolarize. This means that the trans membrane potential becomes less negative. The depolarization is mainly caused by $Na^+ - ions$ passing through the $Na^+ - channel$, giving a higher concentration of positive ions in the cell. When the potential reaches approximately $-40mV$, the contraction is triggered. See [1] to learn more about the human heart and action potentials.

The bidomain model proposed by Winslow et al describe how this trans membrane potential changes over time, using 33 state variables to describe the complex dynamics of the cells. By doing a few modifications to the system, it can be reduced to involving 31 state variables. Each variable is given by a differential equation, which leads to a system of 31 ordinary differential equations to be solved. See Appendix for the equations.

1.3. The full mathematical model

The ODE system to be considered in this thesis, is only a small part of a full mathematical model for simulating an ECG on the body surface. The derivation of this global model is explained in detail in [10].

To give the reader an idea of what part of the full model we are working on, we will now give a quick motivation for the global set of equations.

If we denote by E the electrical field in the body and assume that it remains constant in time, Maxwell's equations says that $\nabla \times E = 0$. This vector field can then be written

in terms of a scalar potential function u_T,

$$E = -\nabla u_T \tag{1.1}$$

If in addition Ohm's law applies in the tissues of the body, $J_T = -M_T E = M_T \nabla u_T$ express the current. M_T is the electric conductivity. By using conservation laws and doing standard mathematic operations, it is possible to derive the following set of equations [10, Paper I, p.11-18].

$$\begin{array}{cc}
\frac{\partial s}{\partial t} = F(t, s(t,x), v(t,x); x) & x \in H \\
\nabla \cdot (M_i \nabla v) + \nabla \cdot (M_i \nabla u_e) = C_m \chi \frac{\partial v}{\partial t} + \chi I_{ion}(v, s) & x \in H \\
\nabla \cdot (M_i \nabla v) + \nabla \cdot ((M_i + M_e) \nabla u_e) = 0 & x \in H \\
\nabla \cdot (M_T \nabla u_T) = 0 & x \in T \\
n \cdot (M_i \nabla v) = 0 & x \in \partial H \\
u_e = u_T & x \in \partial H \\
n \cdot ((M_i + Me) \nabla u_e = n \cdot (M_T \nabla u_T) & x \in \partial H \\
n \cdot (M_T \nabla u_T) = 0 & x \in \partial T
\end{array}$$

As we see, the model is a combination of both ODE's and PDE's. The Winslow model is the set of equations describing the state variables seen in the first equation. Although solving $\frac{\partial s}{\partial t}$ looks like a small part of the problem, computer experiments have measured that up to 90% of the total time is spent solving this particular system. In Figure 1.1 the action potential v for a heart cell is plotted.

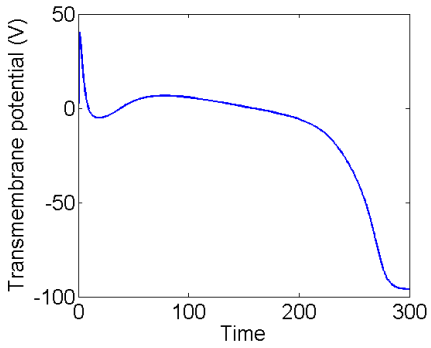


Figure 1.1.: The action potential.

Part II.

Theory, Methods and Numerical Examples

2. Runge-Kutta methods and Stiffness detection

2.1. Solving an ODE numerically

2.1.1. Runge-Kutta methods

Consider the differential equation

$$y'(t) = f(t, y(t)) \quad y(t_0) = y_0 \tag{2.1}$$

A numerical method tries to approximate a solution to the problem by advancing stepwise from a given point (t_0, y_0). At every time step $t_1, t_2, ..., t_m$ it finds approximations $y_1, y_2, ..., y_m$ to the real solution $y(t_1), y(t_2), ..., y(t_m)$, as explained in [8].

A general one-step method can be written as

$$y_{n+1} = y_n + h\Phi(y_n, t_n; h) \tag{2.2}$$

where h is the step size and Φ is a compact notation of the increment function. For example

$$\Phi(y_m, t_m; h) = f(t_m, y_m)$$

is the well known Eulers method.

The method is called explicit as long as Φ does not depend on y_{n+1}, and explicit otherwise.

In hope to get a good approximation to the real solution, the error done in each time step has to stay bounded in some way. The error done in each time step is called the local truncation error, written as

$$r_{n+1} := y(t_{n+1}) - h\Phi(y(t_n), t_n; h). \tag{2.3}$$

The residual error is said to be consistent if it satisfies

$$\tfrac{1}{h} r_{n+1} \to 0 \quad as \quad h \to 0,$$

and the method is convergent if

$$y_n - y(t_n) \to 0 \quad as \quad h \to 0.$$

In this thesis we look at methods that belong to the Runge-Kutta family. In short, this is a class of methods that splits the interval $\{t_n, t_{n+1}\}$ into shorter intervals

$\{t_n + c_1 h, t_n + c_2 h, ..., t_n + c_s h\}$, and finds an approximation to $y(t_{n+1})$ by forming a linear combination of the approximated values in the shorter intervals, see Figure 2.1. A general s-stage one step Runge-Kutta method can be written as

$$\begin{aligned} g_i &= y_n + h \sum_{j=1}^{s} a_{ij} f(t_0 + c_j h, g_j), \quad i = 1, ..., s \\ y_{n+1} &= h \sum_{j=1}^{s} b_j f(t_0 + c_j h, g_j), \end{aligned}$$

where the weights c_i, a_{ij} and b_i are given in a table called a Butcher-Tableau.

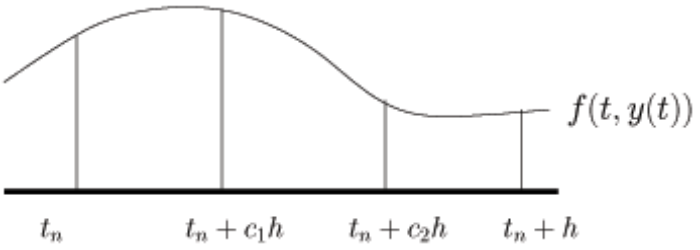


Figure 2.1.: A graphical representation of Runge-Kutta method

c_1	a_{11}	a_{12}	$\cdots$	a_{1s}
c_2	a_{21}	a_{22}	$\cdots$	a_{2s}
$\vdots$	$\vdots$			$\vdots$
c_s	a_{s1}	a_{s2}	$\cdots$	a_{ss}
	b_1	b_2	$\cdots$	b_s

Table 2.1.: The general form of a Butcher-tableau. If $A = \{a\}_{ij}$ is lower triangular, the method is explicit.

2.1.2. Stability

Because there is a lower limit to the step size on a computer, we can not in practice choose h arbitrary small. We therefore look for a property called weak stability. This is a property that tells how well the numerical solution approaches the real solution for finitely small step sizes.

If we apply our Runge-Kutta method to Dahlquist's test equation

$$y' = \lambda y \tag{2.4}$$

with solution $y = \exp(\lambda t) y_0$, we find that the approximated value after advancing one step can be written as

$$y_{n+1} = R(z) y_n.$$

$R(z)$ is called the stability function of the method, and is an approximation to $e^{(z)}$. If the method is explicit with s stages and of order p then $R(z)$ is of the form $1 + z +$

$z^2/2! + ... + z^p/p! + \alpha_{p+1}z^{p+1} + ... + \alpha_s z^s$, and if it is implicit

$$R(z) = \frac{P(z)}{Q(z)} = \frac{det(I - zA + z\mathbb{1}b^T)}{det(I - zA)}.$$

For y_n to remain bounded for all n, a natural condition would be that the stability function $R(z)$ satisfies $|R(z)| \leq 1$. The stability domain of the method is then the region

$$S = \{z \in \mathbb{C}; |R(z)| \leq 1\}.$$

2.1.3. Stiffness

Stiffness is a concept of more practical meaning rather than mathematical. It is easy to understand what stiffness is, but hard to find a precise mathematical definition of it. The first definition of stiffness was (Cutiss and Hirschfelder 1952): *stiff equations are equations where certain implicit methods, in particular BDF, perform better, usually tremendously better, than explicit ones.* Stiffness is related to the eigenvalues of the Jacobian $\partial f/\partial y$ to the ODE system. When the eigenvalues become large, an explicit solver has to reduce the step size significantly in order to obtain a stable solution. So when step size is restricted by stability rather than by accuracy, the system is stiff. Since implicit methods are stable in most of the left half plane, and hence stable for large eigenvalues, they become a natural choice of method for stiff systems.

Then why not always use implicit methods? This has to do with efficiency. To advance one step with an implicit solver is usually a heavier operation involving more computations than the corresponding step with an explicit solver. Unless the problem is stiff. Then, because of stability, the explicit solver is forced to take several smaller steps to obtain a stable solution. This makes the implicit scheme more efficient. So, when choosing solver, it is important to know the characteristics of the problem in order to choose the most efficient one.

2.2. The solvers used

In this thesis we will mainly be working with four different solvers. These are the two implicit methods ESDIRK54a and Radau5, plus the explicit DOPRI 5(4) and Fehlberg 4(5). We will now give a short description of these.

2.2.1. ESDIRK54a

The ESDIRK (Singly Diagonally Implicit Runge-Kutta with Explicit first stage) methods suggested by Anne Kværnø in [7] are a special case of SDIRK methods (see [3] for more on SDIRK). In ESDIRK, the first stage is explicit, meaning that $Y_1 = f(y_n)$, while all other stages are implicit. A general ESDIRK method is given by the Butcher-Tableau

$$
\begin{array}{c|ccccccccc}
0 & 0 & & & & & & & & \\
c_2 & \gamma & \gamma & & & & & & & \\
c_3 & a_{31} & a_{32} & \gamma & & & & & & \\
\vdots & \vdots & & & \ddots & & & & & \\
\vdots & \vdots & & & & \ddots & & & & \\
c_{s-2} & a_{s-2,1} & a_{s-2,2} & a_{s-2,3} & \cdots & \cdots & \gamma & & & \\
1 & a_{s-1,1} & a_{s-1,2} & a_{s-1,3} & \cdots & \cdots & a_{s-1,s-1} & \gamma & & \\
1 & a_{s,1} & a_{s,2} & a_{s,3} & \cdots & \cdots & a_{s,s-2} & a_{s,s-1} & \gamma &
\end{array}
$$

Table 2.2.: A general ESDIRK method, notice the explicit first stage.

The methods are constructed according to the following list of requirements:

1. Stiffly accuracy in both the advancing and the error estimating methods.
2. $|R(\infty)| = 0$, and $|\hat{R}(\infty)|$ as small as possible, at least less than one. This to reduce the influence of inconsistent numerical values from the last step.
3. A-stability, or at least A(α)-stability for both methods.
4. As high stage order as possible.

The full method is of order p and constructed such that the first $s-1$ stages define a method of order $p-1$. This means that if requirement (1) is satisfied for both methods, the error estimate is simply given by

$$err = Y_s - Y_{s-1},$$

and the solution can be advanced using both local extrapolation, meaning that $y_{n+1} = Y_s$, or we could use Fehlberg, meaning that $y_{n+1} = Y_{s-1}$. ESDIRK methods using local extrapolation are denoted by ESDIRK $p/p-1a$ while those using Fehlberg are denoted by ESDIRK$p/p-1b$.

The 7 stage ESDIRK54a that is implemented takes $y_{n+1} = Y_s$ and uses $\gamma = 0.26$. The full
Butcher-Tableau is listed in the appendix.

A major advantage of these methods is that the Jacobian only has to be computed at most once per time step. In addition, since the first stage is explicit and $y_{n+1} = Y_s$, we have already computed the value $Y_1 = f(y_n)$ when starting a new step, a so called FSAL (First Step As Last) method. This means that they not only can be very efficient, but they are also easy to implement.

2.2.2. Radau5

The Radau5 method is a 3 stage implicit Runge-Kutta method of order 5 with good stability properties. It is a collocation method based on the Radau quadrature formula. The abscissas in the collocation polynomial $c_1, ..., c_s$ are chosen to be the zeros of

$$\frac{d^s - 1}{dx^s - 1}\left(x^{s-1}(x-1)^s\right) \tag{2.5}$$

while the other coefficients (b_i, a_{ij}) satisfies

$$\begin{aligned} B(p): &\quad \textstyle\sum_{i=1}^{s} b_i c_i^{q-1} = \frac{1}{q} \quad q = 1, ..., p \\ C(\eta): &\quad \textstyle\sum_{j=1}^{s} a_{ij} c_j^{q-1} = \frac{c_i^q}{q} \quad i = 1, ..., s \quad q = 1, ..., \eta; \end{aligned}$$

When these assumptions are fulfilled, it is a method of order $2s-1$. As we see, the coefficients a_{ij} are found by imposing condition $C(s)$.

The Butcher-Tableau for the method that is used in this thesis, seen in Table 2.3.

$\frac{4-\sqrt{6}}{10}$	$\frac{88-7\sqrt{6}}{360}$	$\frac{296-169\sqrt{6}}{1800}$	$\frac{-2+3\sqrt{6}}{225}$
$\frac{4+\sqrt{6}}{10}$	$\frac{296-169\sqrt{6}}{1800}$	$\frac{88+7\sqrt{6}}{360}$	$\frac{-2-3\sqrt{6}}{225}$
1	$\frac{16-\sqrt{6}}{36}$	$\frac{16+\sqrt{6}}{36}$	$\frac{1}{9}$
	$\frac{16-\sqrt{6}}{36}$	$\frac{16+\sqrt{6}}{36}$	$\frac{1}{9}$

Table 2.3.: The Radau IIA method of order 5.

The stability function of the method is given by

$$R(z) = \frac{P(z)}{Q(z)} = \frac{1 + 2z/5 + z^2/20}{1 - 3z/5 + 3z^2/20 - z^3/60} \tag{2.6}$$

and the stability region is shown in Figure 2.2, taken from [2].

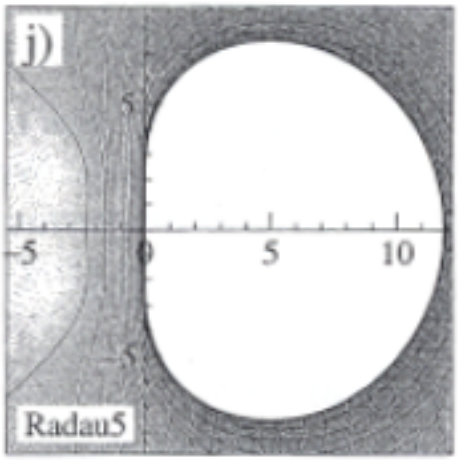


Figure 2.2.: Stability region for the Radau5 method used. Stability outside the white area.

2.2.3. DOPRI 5(4)

This is an explicit method constructed by Dormand and Prince (1980). Like ESDIRK54a, it is an FSAL (First Step As Last) method of order 5 with 6 stages. It uses local extrapolation for error control and step size calculation. This is the default scheme used in the built in MATLAB solver ODE45, and the methods coefficients are given in the Butcher-Tableau in Table 2.4. Its stability domain is plotted in Figure 2.2.3.

0							
$\frac{1}{5}$	$\frac{1}{5}$						
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$					
$\frac{4}{5}$	$\frac{44}{45}$	$-\frac{56}{15}$	$\frac{32}{9}$				
$\frac{8}{9}$	$\frac{19372}{6561}$	$-\frac{25360}{2187}$	$\frac{64448}{6561}$	$-\frac{212}{729}$			
1	$\frac{9017}{3168}$	$-\frac{355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$-\frac{5103}{18656}$		
1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	
y_1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	0
$\hat{y}_1$	$\frac{5179}{57600}$	0	$\frac{7571}{16695}$	$\frac{393}{640}$	$-\frac{92097}{339200}$	$\frac{187}{2100}$	$\frac{1}{40}$

Table 2.4.: DOPRI5(4)

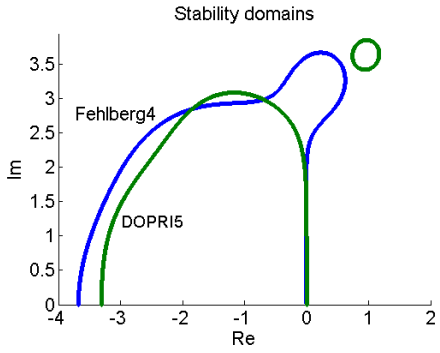


Figure 2.3.: The stability domains of the two explicit solvers DOPRI 5(4) and Fehlberg 4(5). We have stability inside the closed curves.

2.2.4. Fehlberg 4(5)

The Fehlberg 4(5) is an explicit method of order 4, which uses a stage of order 5 for error control and step size calculation. The coefficients of this method are given in the Butcher-Tableau in Table 2.5, and the stability domain are plotted together with the stability domain of DOPRI 5(4) in Figure 2.2.3.

0						
$\frac{1}{4}$	$\frac{1}{4}$					
$\frac{3}{8}$	$\frac{3}{32}$	$\frac{9}{32}$				
$\frac{12}{13}$	$\frac{1932}{2197}$	$\frac{7296}{2197}$				
1	$\frac{439}{216}$	-8	$\frac{3680}{513}$	$-\frac{845}{4104}$		
$\frac{1}{2}$	$-\frac{8}{27}$	2	$-\frac{3544}{2565}$	$\frac{1859}{4104}$	$-\frac{11}{40}$	
y_1	$\frac{25}{216}$	0	$\frac{1408}{2565}$	$\frac{2197}{4104}$	$-\frac{1}{5}$	0
$\hat{y}_1$	$\frac{16}{135}$	0	$\frac{6656}{12825}$	$\frac{28561}{56430}$	$-\frac{9}{50}$	$\frac{2}{55}$

Table 2.5.: Fehlberg 4(5)

2.3. Detecting stiffness

Although a problem is categorized as stiff, it does not necessarily mean that an implicit solver is the most efficient one in the whole interval. There may be areas of the solution where an ERK can be used. Often, stiff problems have an initial transient region where the solution varies quickly. Here, the step size (h_n) has to be small in order to obtain the expected accuracy, and consequently $h_n\lambda_i$ is inside the stability domain of the ERK method. Later the step size increases and $h_n\lambda_i$ falls outside the stability region. To avoid that the error becomes large, the solver now proceeds with a step size h_s such that $h_s\lambda_i$ is close to the stability boundary. We know that the Winslow model has such a transient phase, which makes it interesting to treat this part with an explicit solver, and switch to implicit when the problem gets too stiff. To determine when to switch, we need an indication that the problem is stiff. This is provided by the many existing stiffness detection techniques. We will now a few of these and show some numerical examples. The techniques are all explained in greater detail by Robertson in [9].

2.3.1. Constant step size

This is maybe the simplest way to detect stiffness and is based on the fact that when a problem becomes stiff, the step size is restricted by stability rather than accuracy. The eigenvalues of a stiff problem are usually constant or slow varying. This means that the step size has to be small and almost kept constant in order to keep $h_n\lambda_i$ within the stability region. Based on this, a good indication of stiffness is when the step size drops significantly and the explicit solver proceeds with close to constant step size over a number of steps. This is a very easy way of testing stiffness, but as we shall see, it can be unreliable.

2.3.2. Low-Order Comparison Formula

The idea behind this technique is to take a step with both the basic formula and a comparison formula when attempting to advance the solution. The comparison formula is of lower order (order 1 or 2) but has better stability properties. This means that the comparison formula is supposed to yield a larger error estimate than the basic formula when the problem is non-stiff, but in the stiff case the situation is different. Now, since the comparison formula has a larger stability region than the basic formula we expect that the comparison formula yields the smallest error estimate. In each step the error estimate is

$$||e_n|| \approx \theta TOL \tag{2.7}$$

where $\theta \in [0, 1]$ is called the pessimist factor.[1] If the error estimate $\hat{e}_n$ of the comparison formula satisfies $||\hat{e}_n|| \leq ||e_n||$, we have an indication that the problem may be stiff.

For the two explicit solvers in this thesis we have used different comparison formulas, one for each solver.

[1] The pessimist factor is added to try to avoid a large number of step rejections. Usually $\theta \in [0.8, 0.9]$

DOPRI 5(4)

A closer look at the Butcher-tableau of this method, Table 2.4, unveils that an Euler step of length $h_n c_2$ is taken in each step of the basic formula.

$$y_{n+1}^E = y_n + h_n c_2 f_1$$

If we combine this formula with the result of the Heuns method of order 2,

$$y_{n+1}^H = y_n + h_n c_2 \left(\frac{1}{2}(f_1 + f_2) \right)$$

we get the following estimate for the local error

$$le_n^E = y_{n+1}^H - y_{n+1}^E = (f_2 - f_1) h_n c_2 / 2. \tag{2.8}$$

Since this method has a larger stability domain than DOPRI 5(4) scaled down by $\frac{1}{5}$, as shown in Figure
2.4, we expect to get a smaller local error when the step size is chosen for stability reasons. Thus, the test returns a flag indicating stiffness when

$$||le_n^E|| \leq ||le_n||$$

where le_n is the estimated local error in the basic DORPI 5(4) formula.

Fehlberg 4(5)

For this specific method Shampine and Hiebert have developed a comparison formula pair of order 1 and 2 that uses the full step size h, instead of $h/5$ as in DOPRI 5(4). Another advantage is that the stability domain of this comparison formula is uniformly larger than that for the basic formula, see Figure 2.5. The coefficients of the new pair of comparison formula are

$$\begin{aligned}(b_1, ..., b_6) &= (0.084227, -0.163140, 0.761013, 0.405846, -0.131970, 0.044024)\\ (\hat{b}_1, ..., \hat{b}_6) &= (0.139682, -0.198633, 0.724442, 0.428953, -0.141485, 0.047041)\end{aligned}$$

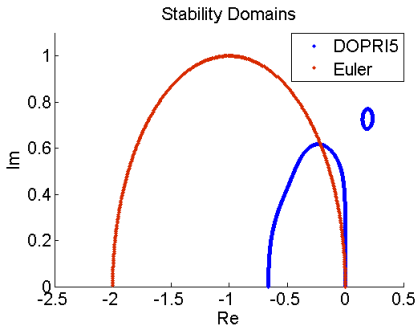


Figure 2.4.: Stability regions for the comparison formula and the fifth order formula DOPRI5 scaled 1/5

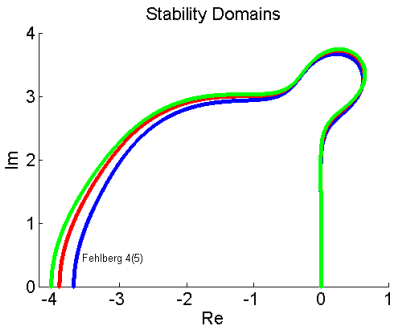


Figure 2.5.: Stability regions for both the basic and the comparison formula for the Fehlberg scheme

2.3.3. Lipschitz Constant Estimation

As mentioned earlier, stiffness has a close connection with the eigenvalues to the Jacobian of the ODE system. A new approach to detecting stiffness could therefore be to estimate the eigenvalues in some way. One obvious solution is to calculate the Jacobian and its corresponding eigenvalues numerically in every step, and observe when the spectral radius [2] $\rho(J)$ becomes large. In an implicit solver this can be done without too much of extra cost since we already need the Jacobian when advancing one step. Explicit solvers are different. Here there is no information about the Jacobian, so we have to find a clever way of estimating $\rho(J)$. A good approximation has turned out to be the Lipschitz constant estimate.

Assume that we are solving the initial value problem $y' = f(x, y), \quad a < x < b, \quad y(a) \quad given.$ If f satisfies a Lipschitz condition with respect to its second argument, i.e

$$||f(x, u) - f(x, v)|| \leq L||u - v||, \tag{2.9}$$

we have the condition

$$\rho(J) \leq ||J|| \leq L, \tag{2.10}$$

[2] The spectral radius is defined to be the modulus of the largest eigenvalue of the Jacobian

where L is called the Lipschitz constant. If we could find a good estimate to the lower bound of the Lipschitz constant, we could hope that

$$L_{est} \approx \rho(J),$$

and if this estimate is big, it could be an indication that the problem may be stiff.

Since $c_6 = c_7 = 1$ in DOPRI 5(4) the approximation

$$L_{est} = \frac{||k_7 - k_6||}{||Y_7 - Y_6||}, \tag{2.11}$$

is a possible way of estimating the Lipschitz constant. Here, we have used the notation $k_i = f(x_n + c_i h_n, Y_i)$. From (2.9) it follows that this is a valid lower bound on L.

For the Fehlberg 4(5) pair, Robertson [9] suggests

$$L_{est} = \frac{||f(y_{n+1}) - f(y_{n+1} + Ke_{n+1})||}{||Ke_{n+1}||}. \tag{2.12}$$

Where K is chosen to satisfy $||Ke_{n+1}|| = \sqrt{u}||y_{n+1}||$ and u is the computer precision. A disadvantage of this estimate is that it requires one extra function call, which is exactly what we are trying to avoid.

2.3.4. Other Stiffness Detection Methods

There are several other ways of detecting stiffness than the techniques just discussed which we are not going to explain in detail. With some extra cost, more reliable stiffness detection methods can be applied. In [9], Robertson goes through several other stiffness detection techniques. In addition, Ekeland, Owren and Øines have in [6] (1998) developed a stiffness detection scheme for explicit Runge-Kutta methods that uses a Krylow space approximation to estimate the eigenvalues of the Jacobian to the ODE system.

2.3.5. Numerical Examples

In this section we want to show how the different stiffness detection techniques discussed work. The techniques are tested on both the Van der Pol equation (with $\epsilon = 1e^{-6}$) and the Winslow model, see Appendix for the respective ODE systems.

Constant step size

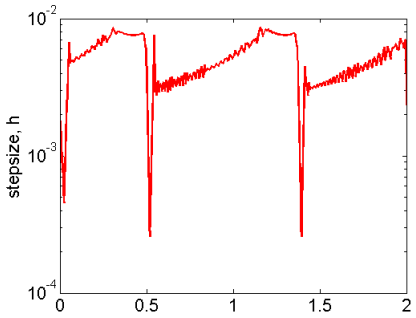


Figure 2.6.: Step size plot for the Van der Pol oscillator. We have used the Fehlberg 4(5) solver at $TOL = 1e^{-4}$

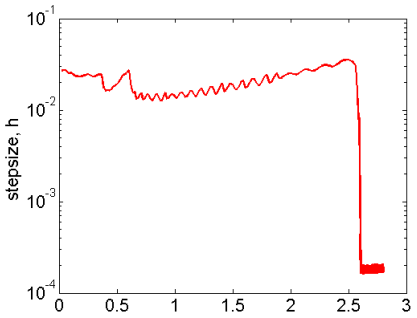


Figure 2.7.: Step size chosen by Fehlberg 4(5) when solving the Winslow model. $TOL = 1e^{-4}$

From Figures 2.6 and 2.7 we can see the unreliability of this stiffness detection technique. For the Van der Pol equation it is hard to conclude that the problem is stiff based on the step size. Here, the step size seems to vary throughout the whole interval giving no clear indication of stiffness. On the other hand, in Figure 2.7, we note that the step size drops dramatically at $t \approx 2.6$msec indicating that the problem has become stiff.

Although this technique seems to be unreliable it unveils the stiffness in the Winslow model. Since the Winslow model is the ODE-system of interest we will keep this technique in mind for later use.

Low-Order Comparison

After some testing on the Winslow model we experienced that if this test succeeded one or two consecutive times, indicating stiffness, it could still fail, but if it succeeded 5 consecutive times it never failed no more. From this, a flag is set indicating stiffness when the test has succeeded 5 times in a row when applied to the Winslow model. In Tables 2.6 and 2.7 we have summed up some stiffness detection statistics for the two different solvers, t_{stiff} is the time where the flag is set to stiff and N_{stiff} is the corresponding step number.

TOL	t_{stiff}	N_{stiff}
$1e^{-4}$	1.081	57
$1e^{-5}$	2.605	161
$1e^{-6}$	2.624	287
$1e^{-7}$	2.638	404
$1e^{-8}$	2.651	555

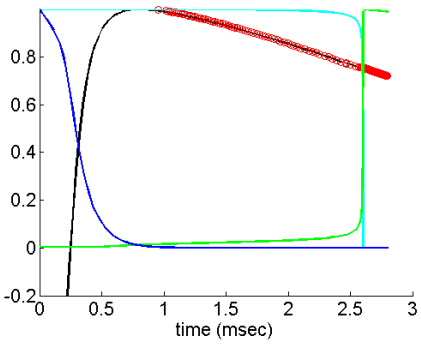
Table 2.6.: DOPRI 5(4)

TOL	t_{stiff}	N_{stiff}
$1e^{-4}$	0.887	48
$1e^{-5}$	2.605	167
$1e^{-6}$	2.619	281
$1e^{-7}$	2.633	409
$1e^{-8}$	2.646	576

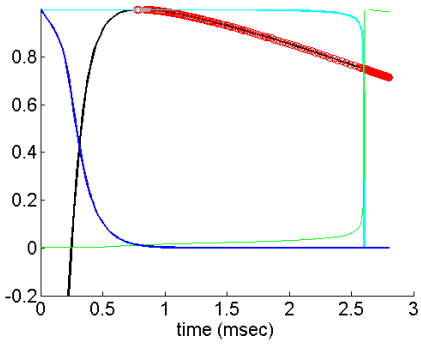
Table 2.7.: Fehlberg 4(5)

From the tables we see that the two methods are almost equal, with the comparison formula for Fehlberg 4(5) detects stiffness just a little later (in time) than for the DOPRI 5(4) detection. In Figure 2.8 we have plotted the solutions of the Winslow model and the points where stiffness is detected. The solutions are scaled so that they all are in the interval $[-1, 1]$.

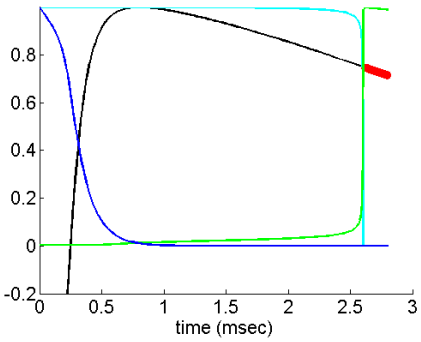
We see that at about t = 2.6msec some of the components in the solution changes dramatically. After this point the problem seems to be very stiff, and as we soon shall see, this is reflected in the Lipschitz constant estimation.



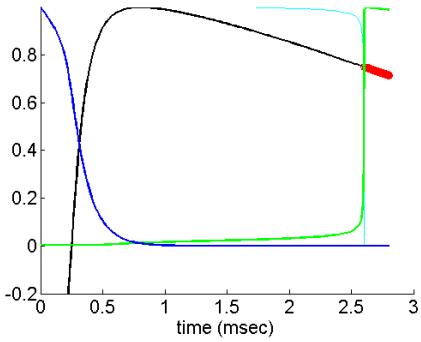
(a) solutions of the Winslow model. DOPRI 5(4) with $TOL = 1e^{-4}$.



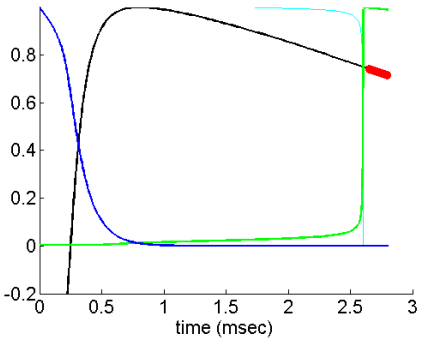
(b) solutions of the Winslow model. Fehlberg 4(5) with $TOL = 1e^{-4}$.



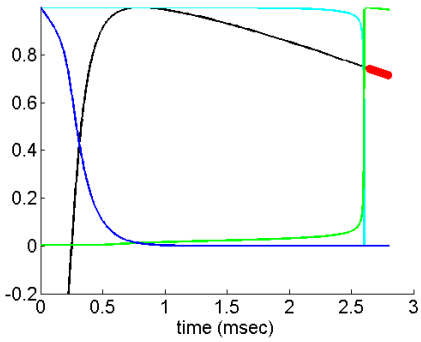
(c) solutions of the Winslow model. DOPRI 5(4) with $TOL = 1e^{-6}$.



(d) solutions of the Winslow model. Fehlberg 4(5) with $TOL = 1e^{-6}$.



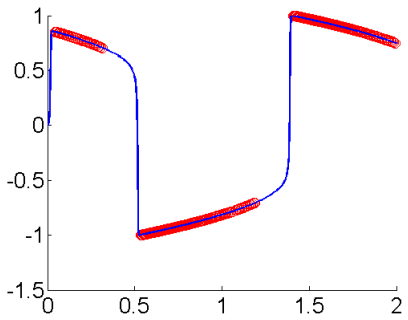
(e) solutions of the Winslow model. DOPRI 5(4) with $TOL = 1e^{-8}$.



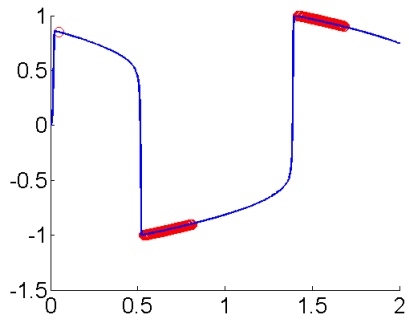
(f) solutions of the Winslow model. Fehlberg 4(5) with $TOL = 1e^{-8}$.

Figure 2.8.: Solutions of the Winslow model. The red dashes indicates where the stiffness test succeeds.

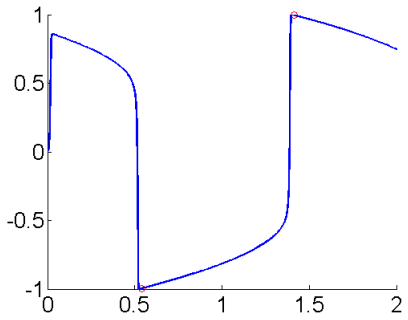
In the next example we want to show the influence that the tolerance has on the stiffness detection. Since the stiffness test should fail as long as the accuracy controls the step size, it is obvious to expect that when the tolerance is strict it should fail more often. For the solver, the problem appears to be less stiff as the tolerance get stricter. This is illustrated in Figure 2.9 we have the results after applying the low-order comparison technique to the Van der Pol equation at different tolerances. We see that, at strict tolerances, the test never indicates stiffness.



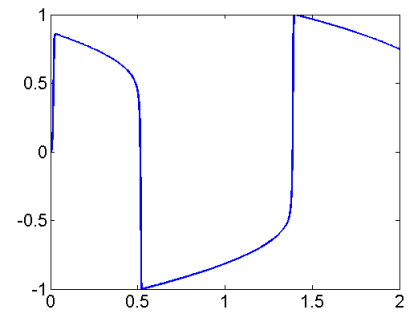
(a) stiffness detection on Van der Pol with $TOL = 1e^{-4}$.



(b) stiffness detection on Van der Pol with $TOL = 1e^{-6}$.



(c) stiffness detection on Van der Pol with $TOL = 1e^{-7}$.



(d) stiffness detection on Van der Pol with $TOL = 1e^{-8}$.

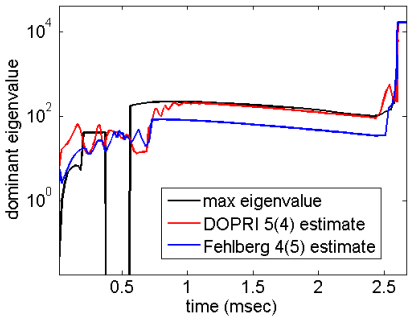
Figure 2.9.: Plot of the first solution component of the Van der Pol equation and where stiffness is indicated. At strict tolerances no stiffness is found.

Lipschitz Constant Estimation

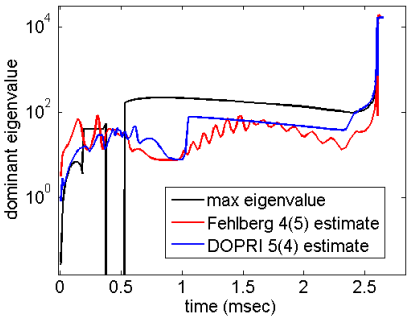
To compare the Lipschitz estimates with the dominant eigenvalue, λ_{max}, we recorded the values at several times t. The eigenvalues of the Jacobian were computed by MATLAB to find $|\lambda_{max}|$. Table 2.8 and Figure 2.10 sums up the results. Here, L_{est}^F is the estimate obtained by (2.12) and L_{est}^D is calculated using (2.11). The estimates are not very good in the less stiff region, but after $t = 2.60$ they both give good approximations. We notice here how fast the maximum eigenvalue is increased at this point.

t	L_{est}^F	L_{est}^D	$\|\lambda_{max}\|$
0.1	7.74	32.20	40.58
0.5	27.7	31.53	154.52
1.0	7.67	7.3	214.23
2.4	100.67	16.9	96.80
2.6	9149.61	810.27	865.59
2.61	16690.12	19440.8	16831.3
2.63	16870.14	16720.21	16810.62

Table 2.8.: Estimates to the Lipschitz constant with Fehlberg 4(5) and DOPRI 5(4), compared to the maximum eigenvalue of the Jacobi matrix, computed numerically in MATLAB. $TOL = 1e^{-5}$.



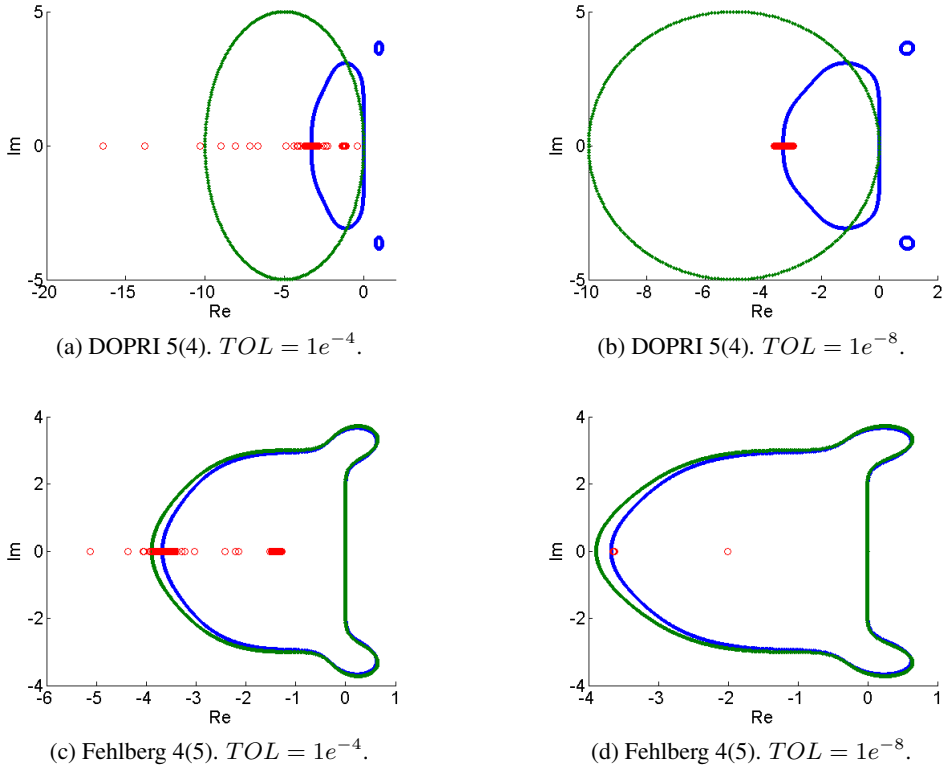
(a) $TOL = 1e^{-5}$.



(b) $TOL = 1e^{-7}$.

Figure 2.10.: Lipschitz constant estimation in the Winslow model and comparison to the dominant eigenvalue.

In the next figure, Figure 2.11, we have tried to illustrate that when the problem is stiff and the step size is chosen for stability reasons, then $h_n\lambda_i$ is close to the stability boundary. This is done by plotting the values for $-L_{est}h$ when the low-order comparison formula indicate stiffness. At high tolerances we see that the values are not necessarily close to the stability bound when stiffness is detected. This makes it a rather unreliable test for stiffness. Luckily the problem is not to detect stiffness, but rather to find out when it is best to advance with an implicit solver instead of an explicit. The estimate of the Lipschitz constant may not be the best stiffness detector, but it may give some useful information about the Jacobian that can be used to find the appropriate time to switch solver from explicit to implicit. Later we are going to see that we are more interested in the order of L_{est}, rather than the exact value.



(a) DOPRI 5(4). $TOL = 1e^{-4}$.

(b) DOPRI 5(4). $TOL = 1e^{-8}$.

(c) Fehlberg 4(5). $TOL = 1e^{-4}$.

(d) Fehlberg 4(5). $TOL = 1e^{-8}$.

Figure 2.11.: $L_{est}h$ when the low-order comparison formula technique indicates stiffness.

3. Step size strategies

3.1. A new step size strategy for the explicit Runge-Kutta methods

Based on the observation that when the step size is chosen for stability reasons in an explicit
Runge-Kutta solver, usually the step sizes are rapidly oscillating and many step rejections occur, a new step size strategy was developed by George Hall [5] (1994). In particular, Hall derives a strategy for the DOPRI 5(4) solver. It uses estimates of the dominant eigenvalue to detect stiffness and to compute a new step size. It also utilizes information about the stability region. The estimates are computed as follows.

Assume, as usual, that we are solving the initial value problem

$$y'(x) = f(x, y(x)), y(0) = y_0 \tag{3.1}$$

with an s stage explicit Runge-Kutta method

$$\begin{aligned} k_i &= f(x_n + c_i h_n, y_n + h_n \sum a_{ij} k_j{}_{j=1}^{i-1}) \quad i = 1, ..., s, \\ y_{n+1} &= y_n + h_n \sum b_i k_i{}_{i=1}^{s}. \end{aligned}$$

When this method is applied to the test equation $y' = Ay$ the k_i-values takes the form

$$h_n k_i = h_n A y_n + \delta_2 (h_n A)^2 + \cdots + \delta_i (h_n A)^i \quad i = 1, ..., s.$$

The idea is now to find weights β_{ij} such that a set of vectors d_i satisfies

$$d_i := h_n \sum \beta_{ij} k_j{}_{j=1}^{i} = (h_n A)^i y_n.$$

If A has a single real dominant eigenvalue λ we have

$$d_{i+1} \approx h_n \lambda d_i.$$

We are interested in an approximation to $h_n\lambda$, and after solving for $h_n\lambda$ in the above equation, we can define the estimates

$$t_i := d_i^T d_{i+1} / (d_i^T d_i)$$

to be the desired approximations. The strategy also uses the quantities

$$r_i := d_{i+1}^T d_{i+1} / (d_i^T d_i).$$

We now have the 5 estimates t_2, t_3, t_4, t_5, t_6 for $h_n\lambda$. In addition we have the quantities r_2, r_3, r_4, r_5, r_6 as approximations to $|h_n\lambda|^2$. These values can be used to indicate that $h_n\lambda$ is near the boundary of the stability region.

For DOPRI 5(4) the β_{ij}-weights has been computed, giving 7 d-vectors.

$$
\begin{aligned}
d_1 &= h_n k_1 \\
d_2 &= 5h_n(k_2 - k_1) \\
d_3 &= 100h_n(2k_3 - 3k_2 + k_1)/9 \\
d_4 &= 25h_n(9k_4 - 64k_3 + 60k_2 - 5k_1)/36 \\
d_5 &= 25h_n(-2187k_5 + 4770k_4 - 14720k_3 + 12720k_2 - 583k_1)/2544 \\
d_6 &= h_n((550/7)k_6 - (18225/212)k_5 - (395000/3339)k_3 + \\
&\quad (500/3)k_2 - (1475/36)k_1) \\
d_7 &= h_n(600k_7 - (2750/7)k_6 + (820125/848)k_5 + (14375/8)k_4 + \\
&\quad (2210000/1113)k_3 - 1125k_2 - (11425/48)k_1)
\end{aligned}
$$

It is recommended to use the values t_6 and r_6 since they are expected to give the best approximations and in Figure 3.1 we see how the t_6/h_n-approximations fit with the dominant eigenvalue. We have here, also plotted the L^D_{est}, which can be used instead of computing the t_i-values.

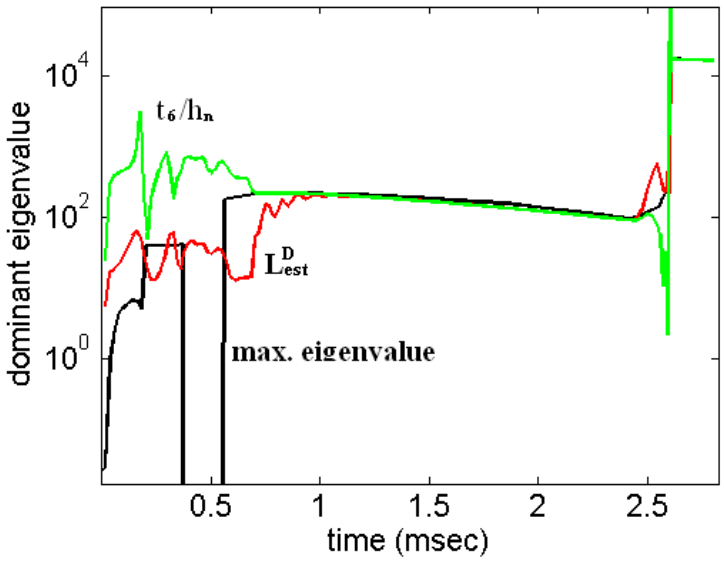


Figure 3.1.: Comparison between L^D_{est} and t_6/h_n.

In DOPRI 5(4) the old standard step size controller

$$h_{n+1} = \left(\frac{\theta TOL}{||err_n||} \right)^{(1/q)} h_n \tag{3.2}$$

is used, where q is the order of the method, θ is the pessimist factor (set to $\theta = 0.9$) and $||err_n||$ is the local error estimate calculated in each step. The step is rejected if $||err_n|| > TOL$, otherwise accepted.

For the new step size controller, Hall suggests the following strategy: Initially we start with the old controller (3.2). We store the 5 last step size changes and let $v_n = h_n - h_{n-1}$. Characterize the problem as stiff once the condition

$$\sum_{i=0}^{4} |v_{n-i}| > 2 \sum_{i=0}^{4} v_{n-i}$$

is satisfied 5 times. This test checks if the step size is oscillating.

Further we use the information about the stability region and the interception of this region with the negative real axis. For DOPRI 5(4) this is at $H_L = -3.3066$. When we have a real dominant eigenvalue we expect the values t_i and r_i to be close to H_L and $|H_L|^2$ respectively. So, after detecting oscillating step sizes we change to the step size strategy

$$h_{n+1} = \frac{3.3}{\gamma} \left(\frac{\theta TOL}{||err_n||} \right)^{(1/q)} h_n$$

if

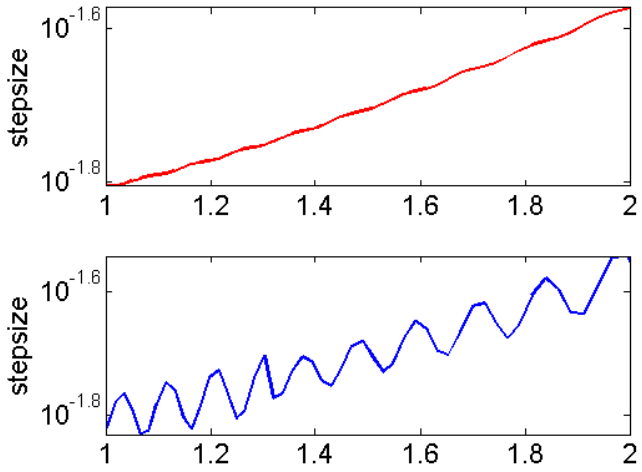
$$4 < r^6 < 25 \quad and \quad -1 > t^6 > -5,$$

where $\gamma = \sqrt{r^6}$.

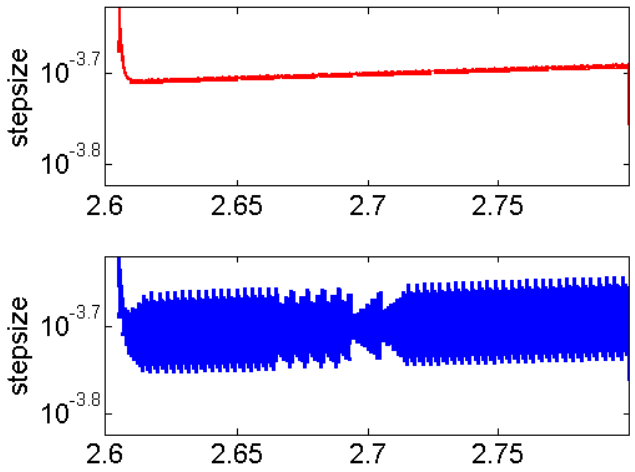
If any of these conditions are not met or after a step rejection, we go back to the old strategy.

3.2. Numerical computations

In Figure 3.2 we have plotted the chosen step sizes with both the new and the old strategy at some different regions of the solution. The new strategy clearly seems to have a positive effect on the oscillating step sizes and in Table 3.1 we see that the number of step rejections are also decreased.



(a) Step sizes. Upper: New strategy, Lower: Old strategy.



(b) Step sizes. Upper: New strategy, Lower: Old strategy.

Figure 3.2.: Comparison of the new and the old step size strategies at two different regions of the solution. Notice how the step size is smoothed in the new strategy.

We saw from Figure 3.1 that the estimate given in (2.11) seemed to be a slightly sharper estimate to the dominant eigenvalue than the estimate obtained by calculating the t_i-values. We therefore have implemented the new step size strategy based on this estimate as well. In Table 3.1 some statistics on the different step size strategies are given.

Here, FEVS are the number of function calls and REJECTED are the number of rejected steps. DOPRI 5(4) and Fehlberg 4(5) use the old step size strategy, while SDOPRI 5(4) and LDOPRI 5(4) are using the new strategy. The difference between S- and LDOPRI 5(4) is that LDOPRI 5(4) is implemented with the L_{est}^{D}-estimate (2.11) while SDOPRI 5(4) utilizes the t_i-values.

TOL	*Solver*	*FEVS*	*REJECTED*
1e-4	DOPRI 5(4)	7268	88
	SDOPRI 5(4)	6908	27
	LDOPRI 5(4)	6914	28
	Fehlberg 4(5)	7800	81
1e-5	DOPRI 5(4)	7160	57
	SDOPRI 5(4)	6992	28
	LDOPRI 5(4)	7022	33
	Fehlberg 4(5)	7872	77
1e-6	DOPRI 5(4)	7382	66
	SDOPRI 5(4)	7196	35
	LDOPRI 5(4)	7256	45
	Fehlberg 4(5)	8082	80
1e-7	DOPRI 5(4)	7676	66
	SDOPRI 5(4)	7574	48
	LDOPRI 5(4)	7610	55
	Fehlberg 4(5)	8520	99
1e-8	DOPRI 5(4)	8138	59
	SDOPRI 5(4)	8048	43
	LDOPRI 5(4)	8126	57
	Fehlberg 4(5)	9084	102

Table 3.1.: Comparison between the different step size strategies regarding function calls and rejected steps.

From the table it seems that the SDOPRI 5(4) solver leads to both less function calls and less rejected steps.

Figure 3.3 shows the number of rejected steps as a function of the tolerance. It is clear that SDOPRI 5(4) should be the favored solver, so from now on, DOPRI 5(4) is implemented with the step size strategy suggested by George Hall [5].

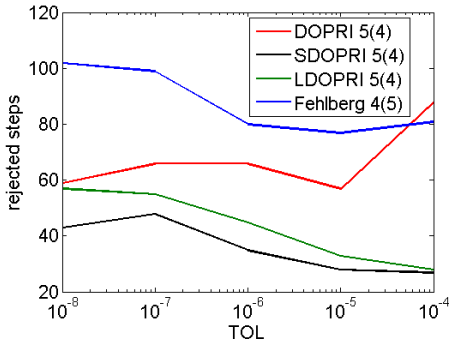


Figure 3.3.: The number of step rejections as a function of the tolerance obtained with the different step size strategies.

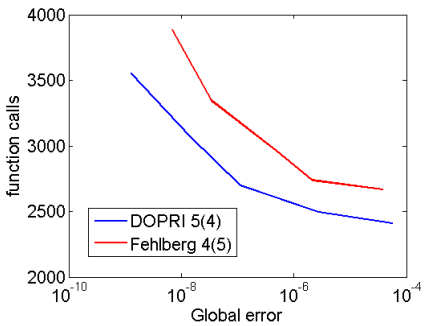
3.3. Comparison of DOPRI 5(4) and Fehlberg 4(5)

In the next chapter we want to put things together and make a solver that combines an explicit with an implicit. This will hopefully improve the efficiency when solving the Winslow model. In the last chapters we have looked at the two explicit solvers DOPRI 5(4) and Fehlberg 4(5). We have done stiffness testing and Lipschitz estimation with both solvers but never compared them with respect to accuracy. It is now time to agree on which explicit solver that is the most suited to combine with the implicit methods. In our case, this is determined by the number of function calls and the accuracy.

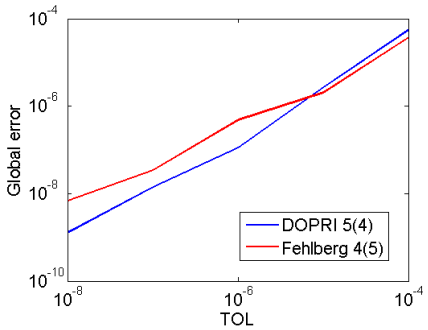
TOL	*Solver*	*FEVS*	*Global err.*
1e-4	DOPRI 5(4)	2414	$5.676e^{-5}$
	Fehlberg 4(5)	2670	$3.8496e^{-5}$
1e-5	DOPRI 5(4)	2498	$2.76e^{-6}$
	Fehlberg 4(5)	2742	$2.096e^{-6}$
1e-6	DOPRI 5(4)	2702	$1.133e^{-7}$
	Fehlberg 4(5)	2964	$4.959e^{-7}$
1e-7	DOPRI 5(4)	3074	$1.439e^{-8}$
	Fehlberg 4(5)	3348	$3.495e^{-8}$
1e-8	DOPRI 5(4)	3554	$1.294e^{-9}$
	Fehlberg 4(5)	3888	$6.919e^{-9}$

Table 3.2.: The number of function calls and the global error when solving the Winslow equation in the interval $t \in [0, 2.65]$.

From Table 3.2 and Figure 3.4 we conclude that DOPRI 5(4) should be the chosen explicit solver. It is both more efficient and accurate than the Fehlberg 4(5) scheme. Another advantage by choosing this solver, is that it has the same order as the implicit ones.



(a) Number of function calls as function of the global error



(b) Global error as function of the tolerance.

Figure 3.4.: Comparison of the two explicit solvers.

Part III.

Putting Everything Together

4. Putting everything together

In the previous chapters we have discussed stiffness detection and step size strategies in explicit Runge-Kutta methods. We have only been studying the initial transient phase of the solution, trying to improve the efficiency in this particular region. It is now time to move on, put it all together, and see whether it is of any use to involve explicit solvers or if it is best not to. Our goal is, as always, to reduce the number of calls to the ODE system. It is also of importance to investigate how the global error is changed.

If we start off with an explicit solver, we need to know when to switch to implicit. That is, we need to find a point that minimizes the number of function calls. To find this point is not necessarily a trivial task, but we are left with two different options. One option is to let the solver figure out itself when it is time to let the implicit solver take over. This will involve using stiffness detection, Lipschitz constant estimation etc. The other option is, since we are solving one specific problem, to explicit tell the solver when to switch. Then the implicit solver takes over at this point regardless of what the explicit solver finds out. The second option is of course the easiest one, but it is not a very dynamic approach, and we shall see that it is very difficult to determine the optimal point.

SESDIRK54a and SRadau5

The combined explicit/implicit solvers will in the following be called SESDIRK54a (SwithingESDIRK54a) and SRadau5 (SwitchingRadau5). They will both initially start with the explicit
DOPRI 5(4) solver just discussed, and later switch to the implicit ESDIRK54a and Radau5 respectively.

4.1. Finding out when to switch

We are now going to figure out when to switch, but first let us see what to improve. Table 4.1 shows some details about how the implicit solvers perform alone on the whole interval $t \in [0, 300]$.

TOL	*Solver*	*FEVS*	*JACCALC*	*Nsteps*
1e-4	ESDIRK54a	5146	98	99
	Radau5	4279	97	101
1e-5	ESDIRK54a	7504	141	142
	Radau5	5756	132	139
1e-6	ESDIRK54a	10529	209	209
	Radau5	7681	180	192
1e-7	ESDIRK54a	14294	296	296
	Radau5	10501	248	275
1e-8	ESDIRK54a	18891	296	296
	Radau5	14578	344	399

Table 4.1.: The number of function calls, Jacobi approximations and the number of successful steps when solving the Winslow model in the interval $t \in [0, 300]$ with the implicit solvers. These are the numbers that we want to improve by solving the initial transient phase with an explicit solver

4.1.1. Hard-coded switch

By experimenting, we start out by trying to find the optimal point to switch at each tolerance. This is almost an impossible task because the total number of function calls is very sensitive with respect to the switching point. We have given it a try, but as we shall see, the points found here are not the ones that minimize the number of function calls for all tolerances.

The routine takes in t_s as a parameter and switches immediately when t_n crosses this switching point t_s. That is when

$$t_n \geq t_s.$$

This means that the actual point where the solver switches can be a little later than the decided t_s depending on the step size, but in this way we avoid unnecessary function calls spent on hitting the switch point exactly.

The results obtained and the switching points t_s are all listed in Table 4.2. We see that all $t_s > 2.60$, except for SRadau5 at tolerance $1e^{-4}$. The column $Improvement$ shows the achieved improvements, in percentage, compared to the numbers in Table 4.1. The best result is at $TOL = 1e^{-5}$ where we have decreased the number of FEVS with 28% while at the other tolerances the improvements are about 10-20%, which is also a significant change.

TOL	*Solver*	t_s	*FEVS*	*Improvement*
1e-4	SESDIRK54a	2.61	4290	16.6%
	SRadau5	0.97	3852	9.9%
1e-5	SESDIRK54a	2.609	5403	28.0%
	SRadau5	2.6055	4579	20.4%
1e-6	SESDIRK54a	2.62	8517	19.1%
	SRadau5	2.62	6283	18.9%
1e-7	SESDIRK54a	2.606	11443	19.9%
	SRadau5	2.61	8036	23.5%
1e-8	SESDIRK54a	2.6535	16278	13.8%
	SRadau5	2.65	11748	19.4%

Table 4.2.: Switching times, number of function calls and the improved efficiency with respect to function calls obtained with the hard-coded switch.

4.1.2. Switch point determined by constant step size

We saw in Chapter 3.3 that the stiffness detection technique based on the step size is in general not very reliable, but for our specific problem it seemed to work. This makes it interesting for us to use this technique to determine where to switch. Another observation, seen in Figure 4.1, is that the step size, after the problem has become stiff, is equal for all tolerances in our problem. The reason for this is obviously because when the step size is chosen for stability reasons rather than accuracy, it becomes more or less independent of the specified tolerance.

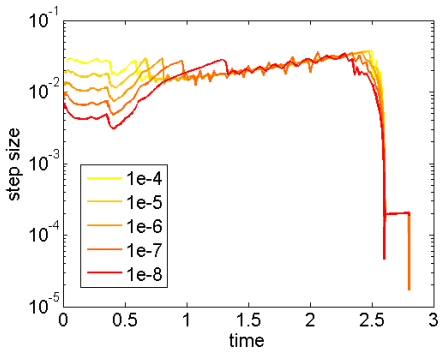


Figure 4.1.: Step size plot of the DOPRI 5(4) solver at the different tolerances

Since the step size seems to be equal in the stiff region at all tolerances, we hope that this can be used to find a general switch point. Figure 4.1 unveils that the step size drops dramatically and stays almost constant at about $t = 2.60$msecs. After taking a closer look, we find that the problem may be categorized as stiff when the step size drops below $2.5e^{-4}$. To be sure that we are in the stiff region, we don't switch before $h_n < 2.5e^{-4}$ has been satisfied 5 consecutive times. Table 4.3 shows how this switching technique works. Here, t_{switch} is the time when the solver switches from explicit to implicit.

TOL	*Solver*	t_{switch}	*FEVS*	*Improvement*
1e-4	SESDIRK54a	2.60578	4113	20.1%
	SRadau5	”	3733	12.7%
1e-5	SESDIRK54a	2.60322	5928	21.0%
	SRadau5	”	4787	16.8%
1e-6	SESDIRK54a	2.60253	8675	17.6%
	SRadau5	”	6276	18.3%
1e-7	SESDIRK54a	2.6019	12195	14.7%
	SRadau5	”	8465	19.4%
1e-8	SESDIRK54a	2.60134	16815	10.9%
	SRadau5	”	11625	20.2%

Table 4.3.: Switching times, number of function calls and the improved efficiency with respect to function calls obtained with the hard-coded switch.

We note that some of the numbers here are better than the ones obtained by the hard-

coded switch, which implies that the hard-coded switch has to be further tuned. Since this is a very time-consuming task and since the expected gain is not very large compared to the numbers found with the other switching techniques, we will not do this here.

4.1.3. Switch point determined by the Low-Order comparison stiffness detection

The idea is now to let the solver figure out itself when it is time to switch by using the Low-Order comparison stiffness detection. We implemented this by switching when the Low-Order comparison indicates stiffness g consecutive times.

As for the hard-coded switch, experiments shows that the number of function calls is very sensitive with respect to the number g. This makes it hard to find a general value for g. It is seen clearly from the oscillations in Figure 4.2 where we have plotted the number of function calls for all values of $g \in [1, 200]$. From g to $g + 1$ the number of function calls may differ by more than 200. In the figure we have minimum at $g = 79$.

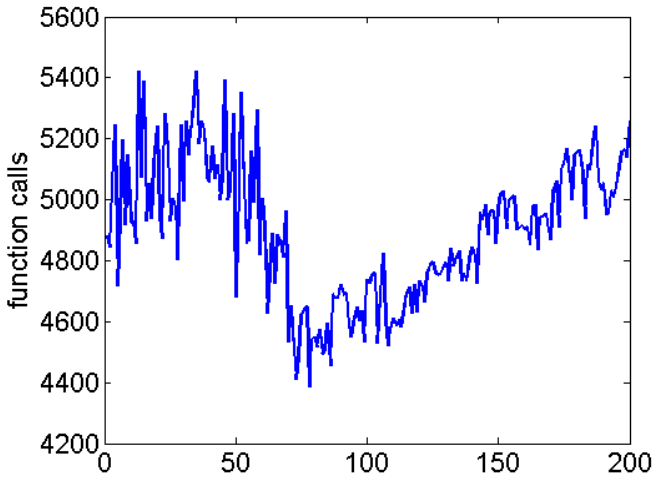


Figure 4.2.: Number of function calls as a function of the number g at $TOL = 1e^{-4}$

By making similar plots to Figure 4.2 for all tolerances we find the number g that minimizes the function calls. The results are listed in Table 4.4 together with the switching times. As we can see, the switching times are very close to the values found in the hard-coded switch.

The observation is that it seems reasonable to assume that $g < 20$ since all values, except SESDIRK54a at $TOL = 1e^{-4}$, of g is less than 20. In SRadau5 we can even say that a suitable value is $g \leq 5$.

TOL	*Solver*	*g*	*FEVS*	t_{switch}	*Improvement*
1e-4	SESDIRK54a	79	4386	2.61935	14.8%
	SRadau5	2	3858	0.971318	9.9%
1e-5	SESDIRK54a	8	5409	2.60915	28.0%
	SRadau5	5	4585	2.60553	20.3%
1e-6	SESDIRK54a	6	8524	2.62315	19.1%
	SRadau5	12	6367	2.62433	17.1%
1e-7	SESDIRK54a	19	11933	2.63942	16.5%
	SRadau5	3	8677	2.63628	17.4%
1e-8	SESDIRK54a	19	16284	2.65363	13.9%
	SRadau5	3	11749	2.65028	19.4%

Table 4.4.: The numbers g that minimizes the number of FEVS and the improvements obtained.

4.1.4. Switch point determined by the Lipschitz estimates

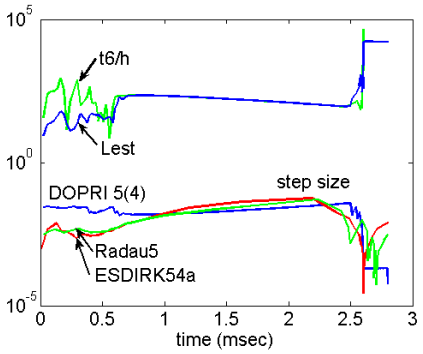
Again we want the solver to switch at a non-user specified point, but now based on the value of the Lipschitz estimate. We know from earlier (Chapter 2) that we have two ways of estimating the Lipschitz constant with DOPRI 5(4), namely (2.11) and t_6/h_n. We are going to look at both options.

With the Lipschitz estimate we hope to, in some way, be able to know something about what step size the implicit solver would have chosen. At every time step DOPRI 5(4) needs 6 function calls. In ESDIRK54a this number depends on the number of Newton-iterations at each stage. The method computes 6 stages and needs in each iteration one function call. In addition we have the computation of the Jacobian, which for the Winslow problem adds 31 function calls. So, if we assume that on average, the Newton iteration converges after two iterations, ESDIRK54a calls the ODE-system a total of $31 + 6 \cdot 2 = 43$ times when advancing one step. Radau5 needs 3 function calls for every Newton-iteration plus the 31 needed for calculating the Jacobian. If we again assume that Newton converges after two iterations, we get $31 + 3 \cdot 2 = 37$ function calls at every time step. This means that it is smart to switch when ESDIRK54a takes a step that is about 7 times larger than the step suggested by DOPRI 5(4). For Radau5 the step has to be about 6 times larger.

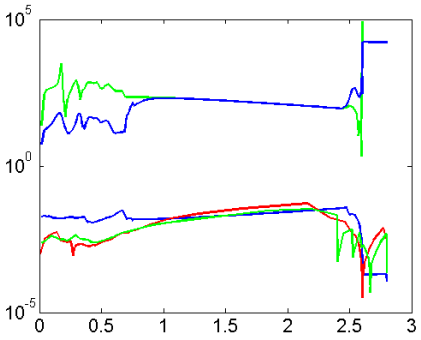
Since we know the step size strategy of ESDIRK54a, we can plot the step size suggested by ESDIRK54a and the Lipschitz estimates from DOPRI 5(4) and see if there is a dependency. To be sure that it is best to proceed with the implicit method, we scale down the step size in ESDIRK54a by 10 instead of 7 when compared to the step size suggested by DOPRI 5(4). The idea is to switch when

$$h_{ESDIRK54a} \geq 10 h_{DOPRI}.$$

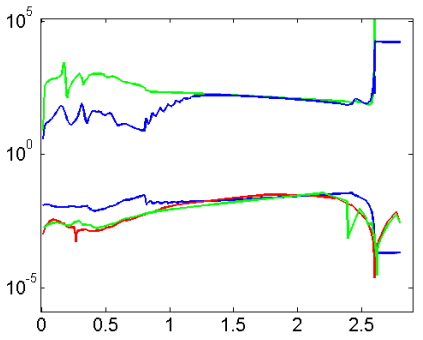
In Figure 4.3 we see how the step sizes behave and the corresponding Lipschitz estimates. We see that it is not easy to find any dependency between the Lipschitz estimate and the step size. At the higher tolerances ($1e^{-4}$ and $1e^{-5}$), it may be possible to switch when the Lipschitz estimate L_{est} satisfies $L_{est} > 100$, but $L_{est} > 10000$ seems to be a safer choice. This is also a choice that will make the solver switch at a point $t > 2.60$, which is suitable for our problem.



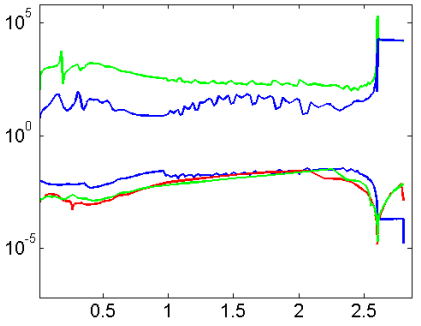
(a) $TOL = 1e^{-4}$



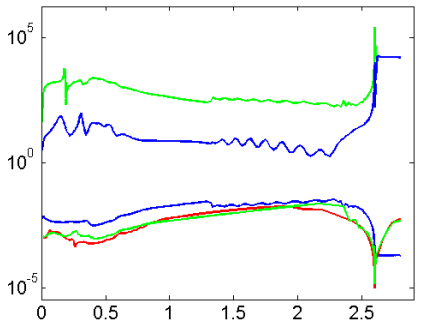
(b) $TOL = 1e^{-5}$



(c) $TOL = 1e^{-6}$



(d) $TOL = 1e^{-7}$



(e) $TOL = 1e^{-8}$

Figure 4.3.: Plots that show the different step sizes and the two estimates to the Lipschitz constant in DORPI 5(4). The step sizes of the implicit methods are scaled 1/10.

The routine is now implemented such that it switches once we know that the Lipschitz estimate is greater than 10 000, i.e when the condition $L_{est} > 10000$ has been satisfied 5 times. The estimate used is (2.11) since this has proved to be a sharper estimate and because it is available without any additional function calls. The results are presented in Table 4.5. Also here we find better results than the results obtained by the hard-coded switch.

TOL	*Solver*	*FEVS*	t_{switch}	*Improvement*
1e-4	SESDIRK54a	4112	2.60512	20.0%
	SRadau5	3700	”	13.5%
1e-5	SESDIRK54a	5730	2.60553	23.6%
	SRadau5	4579	”	20.4%
1e-6	SESDIRK54a	8494	2.60649	19.3%
	SRadau5	6125	”	20.2%
1e-7	SESDIRK54a	11663	2.60748	18.4%
	SRadau5	8117	”	22.7%
1e-8	SESDIRK54a	15988	2.6218	15.4%
	SRadau5	11102	”	23.8%

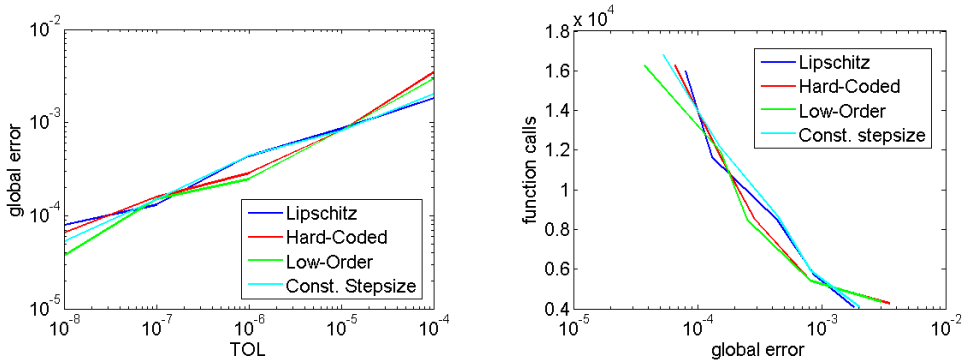
Table 4.5.: Number of function calls and the switching time t when $L_{est} > 10000$.

In general, this switching technique is best regarding function calls, but we want to compare the different techniques with respect to the global error as well before we can make a final decision. In Table 4.6 we have listed the global error for SESDIRK54a and SRadau5 with the four different switches, and we can see the global error is almost equal for all techniques. Figure 4.4 gives a visual representation of the comparisons.

Since all methods seem to be almost equal it is not crucial what choice of switch we do, but since the Lipschitz switch is independent of the tolerance, and usually leads to fewer function calls, we choose to continue with this one.

TOL	*Solver*	*Hard-Coded*	*Const. step size*	*Low-Order*	*Lipschitz*
1e-4	SESDIRK54a	$3.52e^{-3}$	$2.04e^{-3}$	$3.0e^{-3}$	$1.84e^{-3}$
	SRadau5	$8.1e^{-4}$	$4.78e^{-4}$	$8.1e^{-4}$	$4.48e^{-4}$
1e-5	SESDIRK54a	$8.3e^{-4}$	$8.21e^{-4}$	$8.3e^{-4}$	$8.67e^{-4}$
	SRadau5	$2.61e^{-4}$	$2.99e^{-4}$	$2.61e^{-4}$	$2.6e^{-4}$
1e-6	SESDIRK54a	$2.9e^{-4}$	$4.47e^{-4}$	$2.5e^{-4}$	$4.4e^{-4}$
	SRadau5	$3.78e^{-5}$	$7.12e^{-5}$	$4.17e^{-5}$	$7.4e^{-5}$
1e-7	SESDIRK54a	$1.6e^{-4}$	$1.52e^{-4}$	$1.53e^{-4}$	$1.32e^{-4}$
	SRadau5	$2.12e^{-5}$	$2.41e^{-5}$	$2.92e^{-6}$	$1.84e^{-5}$
1e-8	SESDIRK54a	$6.61e^{-5}$	$5.31e^{-5}$	$3.77e^{-5}$	$8.03e^{-5}$
	SRadau5	$3.70e^{-6}$	$6.51e^{-6}$	$3.41e^{-6}$	$7.59e^{-6}$

Table 4.6.: The global error with the different switching techniques.



(a) Error as function of the tolerance

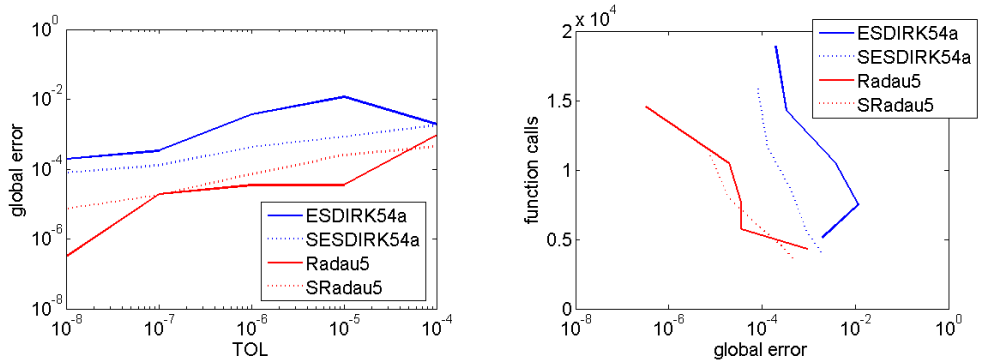
(b) Function calls as function of the global error

Figure 4.4.: Comparison of the different switching techniques discussed with respect to the global error and efficiency.

4.1.5. Global error and efficiency

We now have put everything together. We know which explicit solver to use, and we know where to switch. All we have left to do now is to sum up and see what the improvements are, if any.

The plots in Figure 4.5 compares the old, and purely implicit solvers, ESDIRK54a and Radau5 with the respective new combined explicit/implicit solvers SESDIRK54a and SRadau5. From the plots we can see how the performance of ESDIRK54a has been improved when combined with DOPRI 5(4). Both the accuracy and efficiency with respect to the error has been increased. In the case of Radau5, the improvements are less significant. Here it looks like the explicit/implicit combination has a negative effect on the accuracy, which of course increases the number of function calls with respect to the global error.



(a) Error as function of the tolerance

(b) Function calls as function of the global error

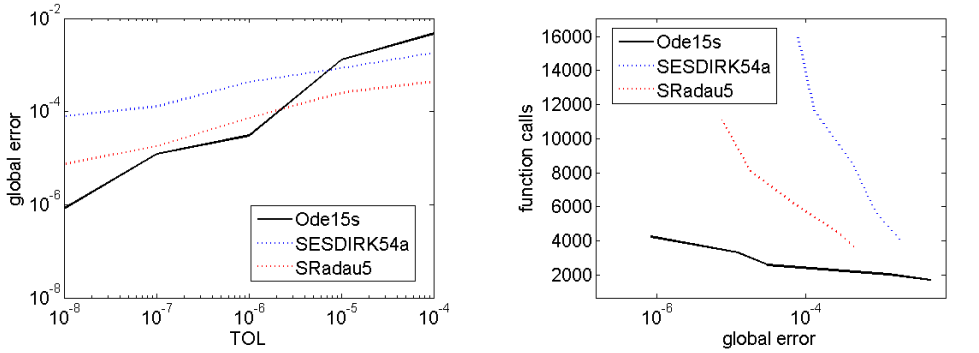
Figure 4.5.: Performance of the combined solvers compared to the respective implicit solvers.

Earlier studies have unveiled that the built in MATLAB multi step solver Ode15s solves the problem very efficiently, and serves as a goal to reach for. We are therefore going to compare the performance of our combined solvers to this method. In Table 4.7 we see how well this method solves the Winslow model, where *Time* is the total time spent solving the problem, measured in seconds.

TOL	$1e^{-4}$	$1e^{-5}$	$1e^{-6}$	$1e^{-7}$	$1e^{-8}$
FEVS	1730	2049	2595	3338	4266
Global error	$4.8e^{-3}$	$1.31e^{-3}$	$3.1e^{-5}$	$1.24e^{-5}$	$8.36e^{-7}$
Time	3.18s	3.74s	4.8s	6.28s	8.14s

Table 4.7.: Performance of the MATLAB solver Ode15s on the Winslow model.

The performance is also illustrated in Figure 4.6, where we have compared it to our combined methods. The Ode15s seems still to be invincible.



(a) Error as function of the tolerance

(b) Function calls as function of the global error

Figure 4.6.: Performance of the combined solvers compared to the MATLAB Ode15s solver.

The last thing we are going to look at is the efficiency regarding the actual time spent. This is because the aim is really to reduce the time, and not necessarily the number of function calls spent solving the problem. But since we know that there is a close relationship between the two when solving the Winslow model, we only hope to verify that the work done on reducing the number of function calls speeds up the solver as well. In Figure 4.7 we get this verification. The plot is almost equal to Figure 4.5(b) which indicates that there is a direct link between the number of function calls and the time spent.

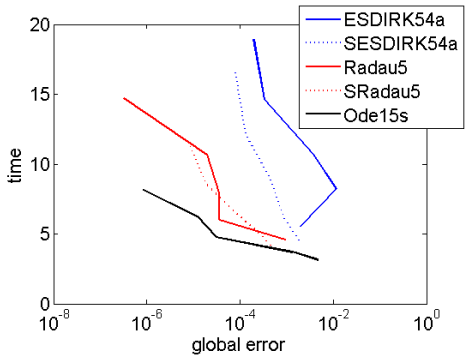


Figure 4.7.: Performance of the combined solvers compared to the MATLAB Ode15s solver.

5. Conclusion and further recommendations

In this thesis we studied the solution of the Winslow model. This is an ODE system of 31 equations, derived by Raimond L. Winslow et al. in [4]. Earlier studies have unveiled, that there is a direct link between the number of function calls and the time it takes to solve the problem. It is therefore important to find a solver that is cheap on function calls in order to improve the efficiency. ERK's (Explicit Runge-Kutta methods) have this property, but they don't work very well on stiff problems like the Winslow model. Regardless of this, because the Winslow model has an initial transient phase where explicit methods tends to be more efficient, the idea was to start with an explicit solver and switch to an implicit when the problem became too stiff for the ERK method. In order to be able to switch solver we needed a stiffness indicator in the explicit solver.

In Chapter 3 we discussed different stiffness detection techniques for the two ERK methods
DOPRI 5(4) and Fehlberg 4(5). Here, we found that both the Low-Order comparison and the Lipschitz constant estimation techniques are good stiffness indicators. The disadvantage with the Lipschitz estimate in the Fehlberg 4(5) method is that an extra function call is needed, which makes it not of interest in our situation.

Another way to decrease the number of function calls, is to have a sound step size strategy that leads to fewer rejected steps. In Chapter 4, a step size strategy was derived for the DOPRI 5(4). This strategy was developed particularly for stiff problems by George Hall in [5] . The goal was to reduce the number of function calls by smoothing out the step sizes. To do this, he makes use of the properties of the stability domain to DOPRI 5(4). The old step size controller

$$h_{n+1} = \left(\frac{\theta TOL}{||err_n||}\right)^{(1/q)} h_n$$

was changed to

$$h_{n+1} = \frac{3.3}{\gamma}\left(\frac{\theta TOL}{||err_n||}\right)^{(1/q)} h_n,$$

when we had oscillating step sizes. With this substitution, we ended up with less step rejections and a 5% decrease in function calls.

The goal in Chapter 5 was to put everything together. Of the three different switching techniques discussed, the switch based on the Lipschitz estimate

$$L_{est} = \frac{||k_7 - k_6||}{||Y_7 - Y_6||},$$

and a switch based on the step size seemed to be the only switches that did not have to be modified for every tolerance. A comparison with respect to the global error and the efficiency, made us decide on the Lipschitz estimate.

When everything were put together, we saw that treating the initial transient phase with the
DOPRI 5(4) method had a very positive effect regarding both efficiency and accuracy. The number of total function calls was decreased by up to 25% while the accuracy was maintained. Because of the correspondence between function calls and the time to solve the problem, the efficiency with respect to time was improved correspondingly.

We have seen the importance of the number of function calls. The 3-stage Radau5 was more efficient than the 7-stage ESDIRK54a solver. Future work could therefore be to try to solve the problem with an implicit method of lower order. The lower order methods usually have fewer stages, which could lead to fewer function calls. It could also be of interest to track the stiffness of the system on the entire interval $t = [0, 300]$msec, and see if there are other areas where it is appropriate to apply an explicit method.

Another way of improving efficiency is to optimize the solvers regarding the Winslow model by tuning the different method parameters. Here, the possibilities are endless, and the optimal values have to be found by experimenting.

Part IV.

Appendix and Bibliography

Appendix

The Van der Pol oscillator

$$\begin{array}{rcll} y_1' & = & y_2 & , y_1(0) = 2 \\ y_2' & = & ((1-y_1^2)y_2 - y_1)/\epsilon & , \epsilon = 10^{-6} \quad , y_2(0) = 0 \end{array}$$

The Winslow ODE system of 31 equations

The transmembrane potential

$$\begin{aligned} \frac{\mathrm{d}V}{\mathrm{d}t} &= -(I_{Na} + I_{Ca} + I_{Ca,K} + I_{Kr} + I_{Ks} + I_{to} + I_{K1} + I_{Kp} \\ &\quad + I_{NaCa} + I_{NaK} + I_{p(Ca)} + I_{Ca,b} + I_{Na,b}) \end{aligned}$$

K^+- and Na^+- gate variables

$$\begin{array}{rcl} \frac{\mathrm{d}m}{\mathrm{d}t} & = & \alpha_m(1-m) - \beta_m m \\ \frac{\mathrm{d}h}{\mathrm{d}t} & = & \alpha_h(1-h) - \beta_h h \\ \frac{\mathrm{d}j}{\mathrm{d}t} & = & \alpha_j(1-j) - \beta_j j \\ \frac{\mathrm{d}X_{Kr}}{\mathrm{d}t} & = & \frac{X_{Kr}^{\infty} - X_{Kr}}{\tau_{X_{Kr}}} \\ \frac{\mathrm{d}X_{Ks}}{\mathrm{d}t} & = & \frac{X_{Ks}^{\infty} - X_{Ks}}{\tau_{X_{Ks}}} \\ \frac{\mathrm{d}X_{to}}{\mathrm{d}t} & = & \alpha_{X_{to}}(1-X_{to}) - \beta_{X_{to}} X_{to} \\ \frac{\mathrm{d}Y_{to}}{\mathrm{d}t} & = & \alpha_{Y_{to}}(1-Y_{to}) - \beta_{Y_{to}} Y_{to} \end{array}$$

Here all coefficients only depend on the transmembrane potential V.

RyR-channel

$$\begin{array}{rcl} \frac{\mathrm{d}P_{C_1}}{\mathrm{d}t} & = & -k_a^+ \left[Ca^{2+}\right]_{ss}^4 P_{C_1} + k_a^- P_{O_1} \\ \frac{\mathrm{d}P_{O_1}}{\mathrm{d}t} & = & -k_a^+ \left[Ca^{2+}\right]_{ss}^4 P_{C_1} - k_a^- P_{O_1} - k_b^+ \left[Ca^{2+}\right]_{ss}^3 P_{O_1} + k_b^- P_{O_2} - k_c^+ P_{O_1} + k_c^- P_{C_2} \\ \frac{\mathrm{d}P_{O_2}}{\mathrm{d}t} & = & -k_b^+ \left[Ca^{2+}\right]_{ss}^3 - k_b^- P_{O_2} \\ \frac{\mathrm{d}P_{C_2}}{\mathrm{d}t} & = & k_c^+ P_{O_1} - k_c^- P_{C_2} \end{array}$$

All the k-coefficients are constant. We will later see that the stiffness of the system is to a great extent attributed to the state variable P_{C_1}.

L-type Ca^{2+}-channel

$$
\begin{aligned}
\frac{\mathrm{d}C_0}{\mathrm{dt}} &= \beta C_1 + \omega C_{Ca0} - (4\alpha + \gamma)C_0 \\
\frac{\mathrm{d}C_1}{\mathrm{dt}} &= 4\alpha C_0 + 2\beta C_2 + \tfrac{\omega}{b} C_{Ca1} - (\beta + 3\alpha + \gamma a)C_1 \\
\frac{\mathrm{d}C_2}{\mathrm{dt}} &= 3\alpha C_1 + 3\beta C_3 + \tfrac{\omega}{b^2} C_{Ca2} - (2\beta + 2\alpha + \gamma a^2)C_2 \\
\frac{\mathrm{d}C_3}{\mathrm{dt}} &= 2\alpha C_2 + 4\beta C_4 + \tfrac{\omega}{b^3} C_{Ca3} - (3\beta + \alpha + \gamma a^3)C_3 \\
\frac{\mathrm{d}C_4}{\mathrm{dt}} &= \alpha C_3 + gO + \tfrac{\omega}{b^4} C_{Ca4} - (4\beta + f + \gamma\alpha^4)C_4 \\
\frac{\mathrm{d}O}{\mathrm{dt}} &= fC_4 - gO \\
\frac{\mathrm{d}C_{Ca0}}{\mathrm{dt}} &= \beta' C_{Ca1} + \gamma C_0 - (4\alpha' + \omega)C_{Ca0} \\
\frac{\mathrm{d}C_{Ca1}}{\mathrm{dt}} &= a\alpha' C_{Ca0} + 2\beta' C_{Ca2} + \gamma a C_1 - (\beta' + 3\alpha' + \tfrac{\omega}{b})C_{Ca1} \\
\frac{\mathrm{d}C_{Ca2}}{\mathrm{dt}} &= 3\alpha' C_{Ca1} + 3\beta' C_{Ca3} + \gamma a^2 C_2 - (2\beta' + 2\alpha' + \tfrac{\omega}{b^2})C_{Ca2} \\
\frac{\mathrm{d}C_{Ca3}}{\mathrm{dt}} &= 2\alpha' C_{Ca2} + 4\beta' C_{Ca4} + \gamma a^3 C_3 - (3\beta' + \alpha' + \tfrac{\omega}{b^3})C_{Ca3} \\
\frac{\mathrm{d}C_{Ca4}}{\mathrm{dt}} &= \alpha' C_{Ca3} + \gamma a^4 C_4 - (4\beta' + f' + \tfrac{\omega}{b^4})C_{Ca4} \\
\frac{\mathrm{d}y}{\mathrm{dt}} &= \tfrac{y_\infty - y}{\tau_y}
\end{aligned}
$$

Intracellular Ca^{2+} fluxes (slow buffers)

$$
\begin{aligned}
\frac{\mathrm{d}[HTRPNCa]}{\mathrm{dt}} &= k^+_{htrpn}\left[Ca^{2+}\right]_i([HTRPN]_{tot} - [HTRPNCa]) - k^-_{htrpn}[HTRPNCa] \\
\frac{\mathrm{d}[LTRPNCa]}{\mathrm{dt}} &= k^+_{ltrpn}\left[Ca^{2+}\right]_i([LTRPN]_{tot} - [LTRPNCa]) - k^-_{ltrpn}[LTRPNCa]
\end{aligned}
$$

Intracellular ionic concentrations

$$
\begin{aligned}
\frac{\mathrm{d}\left[K^+\right]}{\mathrm{dt}} &= -(I_{Kr} + I_{Ks} + I_{to} + I_{K1} + I_{Kp} + I_{Ca,K} - 2I_{NaK})\tfrac{A_{cap}C_{sc}}{V_{myo}F} \\
\frac{\mathrm{d}\left[Ca^{2+}\right]_i}{\mathrm{dt}} &= \beta_i\left(J_{xfer} - J_{up} - J_{trpn} - (I_{Ca,b} - 2I_{NaCa} + I_{p(Ca)})\tfrac{A_{cap}C_{sc}}{2V_{myo}F}\right) \\
\frac{\mathrm{d}\left[Ca^{2+}\right]_{ss}}{\mathrm{dt}} &= \beta_{ss}(J_{rel}\tfrac{V_{JSR}}{V_{ss}} - J_{xfer}\tfrac{V_{myo}}{V_{ss}} - I_{Ca}\tfrac{A_{cap}C_{sc}}{2V_{myo}F}) \\
\frac{\mathrm{d}\left[Ca^{2+}\right]_{JSR}}{\mathrm{dt}} &= \beta_{JSR}(J_{tr} - J_{rel}) \\
\frac{\mathrm{d}\left[Ca^{2+}\right]_{NSR}}{\mathrm{dt}} &= J_{up}\tfrac{V_{myo}}{V_{NSR}} - J_{tr}\tfrac{V_{JSR}}{V_{NSR}}
\end{aligned}
$$

ESDIRK54a - The coefficients in the method And the numerical values

0						
γ	γ					
a_{31}	a_{32}	γ				
a_{41}	a_{42}	a_{43}	γ			
a_{51}	a_{52}	a_{53}	a_{54}	γ		
a_{61}	a_{62}	a_{63}	a_{64}	a_{65}	γ	
a_{71}	a_{72}	a_{73}	a_{74}	a_{75}	a_{76}	γ

Table 5.1.: ESDIRK p/p-1

$$
\begin{aligned}
\gamma &= 0.26 \\
a_{31} &= 0.13 \\
a_{32} &= 0.84033320996790809; \\
a_{41} &= 0.22371961478320505; \\
a_{42} &= 0.47675532319799699; \\
a_{43} &= -0.06470895363112615; \\
a_{51} &= 0.16648564323248321; \\
a_{52} &= 0.10450018841591720; \\
a_{53} &= 0.03631482272098715; \\
a_{54} &= -0.13090704451073998; \\
a_{61} &= 0.13855640231268224; \\
a_{63} &= -0.04245337201752043; \\
a_{64} &= 0.02446657898003141; \\
a_{65} &= 0.61943039072480676; \\
a_{71} &= 0.13659751177640291; \\
a_{73} &= -0.05496908796538376; \\
a_{74} &= -0.04118626728321046; \\
a_{75} &= 0.62993304899016403; \\
a_{76} &= 0.06962479448202728;
\end{aligned}
$$

Bibliography

[1] J.E. Hall A.C. Guyton. *Textbook of Medical Physiology, 10th edition*. Number 10 in 10. W.B. Saunders Company, 2000.

[2] E.Hairer and G.Wanner. *Solving Ordinary Differential Equations II, Stiff and Differential-Algebraic Problems*. Number 3 in Springer Series. Springer-Verlag, 2002.

[3] S.P. Nørsett E.Hairer and G.Wanner. *Solving Ordinary Differential Equations I, Nonstiff Problems*. Number 4 in Springer Series. Springer-Verlag, 1987.

[4] Raimond L. Winslow et al. Mechanisms of Altered Excitation-Contraction Coupling in Canine Tachycardia-Induced Heart Failure, ii : Model Studies. *Circulation Research*, 1(1), 1999.

[5] George Hall. A New Stepsize Strategy for Runge-Kutta Codes. *University of Manchester, Department of Mathematics*, 1(14), 1994.

[6] Brynjulf Owren Kersti Ekeland and Eivor Øines. Stiffness Detection and Estimation of Dominant Spectra with Explicit Runge-Kutta Methods. *The Norwegian University of Science and Technology, The Department of Sciences*, 24(4), 1998.

[7] A. Kværnø. Singly Diagonally Implicit Runge-Kutta Methods with an Explicit First Stage. *The Norwegian University of Science and Technology, The Department of Sciences*, 10(6), 2002.

[8] B. Owren. Forelesningsnotater i TMA4215 Numerisk Matematikk. *The Norwegian University of Science and Technology, The Department of Sciences*, 2005.

[9] Brian Christopher Robertson. Detecting Stiffness with Explicit Runge-Kutta Formulas. *University of Toronto, Department of Computer Science*, 87(193), 1987.

[10] J. Sundnes. Numerical Methods for Simulating the Electrical Activity of the Heart. *University of Olso,Faculty of Mathematics and Natural Sciences*, 1(226), 2002.