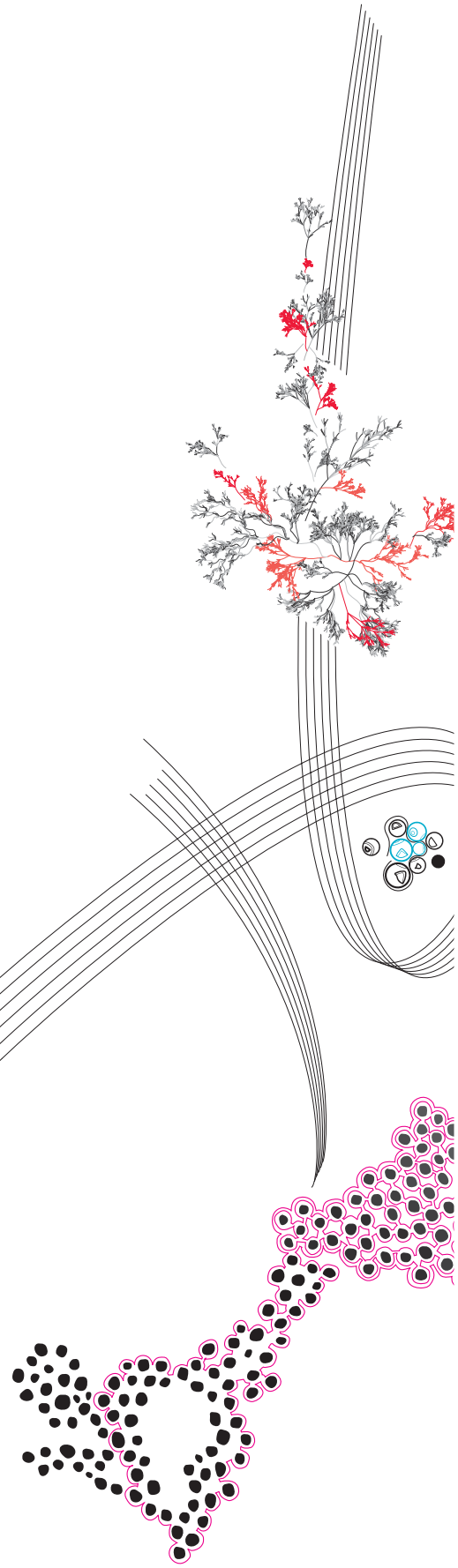

BSc Thesis Applied Mathematics

Model order reduction on FitzHugh-Nagumo model

Fleur van Alphen

Supervisor: Kathrin Smetana

June, 2020

Department of Applied Mathematics
Faculty of Electrical Engineering,
Mathematics and Computer Science

UNIVERSITY OF TWENTE.

Preface

I would like to thank Kathrin Smetana for her support in this thesis. She helped me to find a project on model order reduction with a model that was in my field of interest. She always took the time to check the steps I had taken, and helped me find the next steps in this research. Also, I would like to thank Miranda van der Bend for checking the spelling and grammar of this report, and I would like to thank Ruben de Baaij, Fay van Alphen and Lilian Spijker for their support.

Enschede, June, 2020.

Model order reduction on FitzHugh-Nagumo model

Fleur van Alphen*

June, 2020

Abstract

A combination of proper orthogonal decomposition (POD) and the Galerkin projection is used as a dimension reduction method, and is applied to the FitzHugh-Nagumo model. This dimension reduction method is shown to be effective on the linear part of the FitzHugh-Nagumo model, in the sense that far fewer variables are present, but the complexity of evaluating the nonlinear term remains that of the original problem. By applying this model order reduction to the FitzHugh-Nagumo model, the model can become more relevant due to the improved rapidity of approximating the results, without losing too much accuracy. The full order results are obtained by discretizing the model with an Implicit-Explicit scheme. The reduced order model is obtained by applying POD and a Galerkin projection to the full order results.

Keywords: Model order reduction, IMEX, POD, Galerkin projection, FitzHugh-Nagumo model

1 Introduction

Differential equations play a prominent role in many fields, such as engineering and biology. The solutions to differential equations often can not be found analytically, so numerical methods are needed to approximate the solutions. However, these differential equations can become very complex. Therefore it takes a lot of time and computer capacity to solve them. It is important that mathematical models are fast and precise in order to be relevant. Model order reduction can be used to this end [16].

Model order reduction is a method that reduces the computational complexity and computational time of big dynamical systems. An approximation to the original model is computed by reducing the model's state-space dimensions. This approximation has a much lower dimension but can nearly produce the same input/output characteristics [5]. In this thesis, model order reduction is applied to the FitzHugh-Nagumo model, a system of two high dimensional nonlinear ordinary differential equations. The FitzHugh-Nagumo model is a simplification of the Hodgkin-Huxley model, the nowadays considered classical model for nerve signal propagation [17]. We must use a very accurate and high dimensional discretization leading to a large system. To reduce the order of the system, model order reduction can be used [5]. The reduction of the FitzHugh-Nagumo system can for instance be used to create a precise personalized medication for patients.

Research has been done to model order reduction. 'Reduced Basis methods for Partial Differential Equations' [18] provides a basic mathematical introduction to reduced basis

*Email: f.i.m.vanalphen@student.utwente.nl

methods. Here, the theoretical properties, implementations aspects and errors are analyzed. Also in [19] is looked at reduced basis approximations and error estimation methods for rapid and reliable evaluation. It focuses on linear output functions and linear elliptic coercive partial differential equations, but the methods can be applied more generally. In [9] is looked at elliptic coercive problems and at time dependent cases with corresponding reduced basis formulations.

In this thesis, we discretize the FitzHugh-Nagumo model using the Implicit-Explicit Euler (IMEX) method [2]. We apply proper orthogonal decomposition (POD) to the solutions in order to reduce the dimensions of the system and use the Galerkin projection to create a new system of equations. The combination of POD and the Galerkin projection is a popular approach for constructing reduced-order models [5].

To obtain results, we have changed one variable of the FitzHugh-Nagumo model to be able to model the problem. We have shown that it is possible to reduce the order of the FitzHugh-Nagumo model using POD and a Galerkin projection without losing too much accuracy. However, a lot of computation time is still needed due to the nonlinear term.

To improve the dimension reduction efficiency of POD further, we advise using an emperical interpolation method (EIM), for example the discrete empirical interpolation method (DEIM). The idea of EIM is to approximate a given nonlinear valued function by a function that is rapidly computable [22, 3]. The EIM has sucessfully been applied to several nonlinear problems [4]. The effectiveness of the DEIM is already demonstrated in far more complex applications, for example the Hodgkin-Huxley model of realistic spiking neurons [11]. DEIM is also applied to the Fitzhugh-Nagumo model, and is demonstrated to be a promising approach to overcome the deficiencies of POD and to further reduce the dimension for time dependent and/or nonlinear Partial differential equations like the FitzHugh-Nagumo model [5].

In chapter 2 of this paper, we discuss methods for time discretization and we give an explanation of our choice of the IMEX method. In chapter 3 we discretize the FitzHugh-Nagumo model. The model order reduction is explained and an overview of steps is given in chapter 4. In appendix A we discretize the heat equation to check if all steps of the discretization of the FitzHugh-Nagumo model are correct. We present the results of our model in chapter 5, i.e. the solution of the heat equation, the solution of the FitzHugh-Nagumo model and the solution of the reduced order model of the FitzHugh-Nagumo model. The conclusions, limitations and further research possibilities are stated in chapter 6.

2 Time discretization schemes

Differential equations are used to describe the dynamics of a system. When an analytic solution to the differential equation can not be found, discretization to model the system is often used. Different methods can be used to discretize a differential equation. In this thesis, we have used the IMEX method, a combination of the Forward Euler and the Backward Euler method. In this section, we will give a description of the forward Euler method and the backward Euler method and compare these two methods. We will also present the IMEX method.

2.1 Forward Euler method

The forward Euler method is an explicit method for time discretization. It is a method to solve differential equations. Let $u \in C^1([t_0, T])$ be a continuous, differentiable function with $t_0 \leq T \in \mathbb{R}$, $u_0 \in \mathbb{R}$ and $f : [t_0, T] \times \mathbb{R} \to \mathbb{R}$ [15, 21]. Given an initial value problem

$$\dot{u}(t) = f(t, u(t)) \qquad u(t_0) = u_0$$

with $t \in [t_0, T]$, we can calculate the derivative as

$$\dot{u}(t) = \lim_{h \to 0} \frac{u(t+h) - u(t)}{h}.$$

By choosing h sufficiently small, we can approximate this derivative numerically. Using this numerical approximation, we can approximate the differential equation $\dot{u}(t) = f(t, u(t))$ by

$$\frac{u(t+h) - u(t)}{h} \approx f(t, u(t)).$$

This can be rewritten as

$$u(t+h) \approx u(t) + hf(t, u(t)). \tag{1}$$

We have arrived at an equation where we can approximate a value of u at time $t+h$ if we know the value of u at time t.
Introducing a grid of time steps $t_0 < t_1 < ... < t_N$ with $t_{n+1} = t_n + h$ where $h > 0$ and defining $u_n = u(t_n)$, we can rewrite equation (3) as

$$u_{n+1} \approx u_n + hf(t_n, u_n).$$

This is called the forward Euler method [1].

2.2 Backward Euler method

The backward Euler method is an implicit method for time discretization. It is also used to solve differential equations, but has other stability properties than the forward Euler method. We will discuss the backward Euler method since it is part of the IMEX method that we use to discretize our model. We define $u(t), u_0$ and $f(t, u(t)$ as in chapter 2.1. For $t \in [t_0, T]$, the initial value problem

$$\dot{u}(t) = f(t, u(t)) \quad u(t_0) = u_0$$

is given. We know the derivative can be calculated as

$$\dot{u}(t+h) = \lim_{h \to 0} \frac{u(t+h) - u(t)}{h}. \tag{2}$$

Notice that this derivative is slightly different from the derivative in the forward Euler method. By choosing h sufficiently small, we can approximate this derivative numerically. Using equation (2), we approximate the differential equation $\dot{y}(t+h) = f(t+h, y(t+h))$ by

$$\frac{u(t+h) - u(t)}{h} \approx f(t+h, u(t+h)).$$

This can be rewritten as

$$u(t+h) \approx u(t) + hf(t+h, u(t+h)). \tag{3}$$

We now introduce a grid of time steps $t_0 \leq t_1 \leq ...t_N$ with $t_{n+1} = t_n + h$. When we define $u_n = u(t_n)$, we can rewrite equation (3) as

$$u_{n+1} \approx u_n + hf(t_{n+1}, u_{n+1})$$

This method is called the backward Euler method. Notice that the function f is calculated at a different time step in the forward Euler method than in the backward Euler method. With this equation it is possible to estimate a value of u at time $t+h$ if the value of u at time t is known. However, it is much more difficult to solve this equation than the equation of the forward Euler method. This is because the input for f, u_{n+1} is not known yet when we want to calculate u_{n+1}. If f is nonlinear, we have to find the roots at each time step to find a solution [24].

2.3 Comparing Forward and Backward Euler method

As mentioned before, the forward Euler method is easier to implement than the backward Euler method as we do not have to solve a (non)linear system of equations in every time step. The forward Euler method is also faster and takes less memory. However, the biggest advantage of the backward Euler method is that it has greater stability properties. This means that smaller time steps have to be chosen for the forward Euler method in order to result a stable solution [23].

This difference in stability is especially noticeable for stiff problems. These kind of problems are reasonably handled by the backward Euler method, where the forward Euler method fails because prohibitevely small time steps have to be used. The stiffness causes that small variations at time t result in very big variations at time $t+1$. This difference is also shown in Fig. 1.

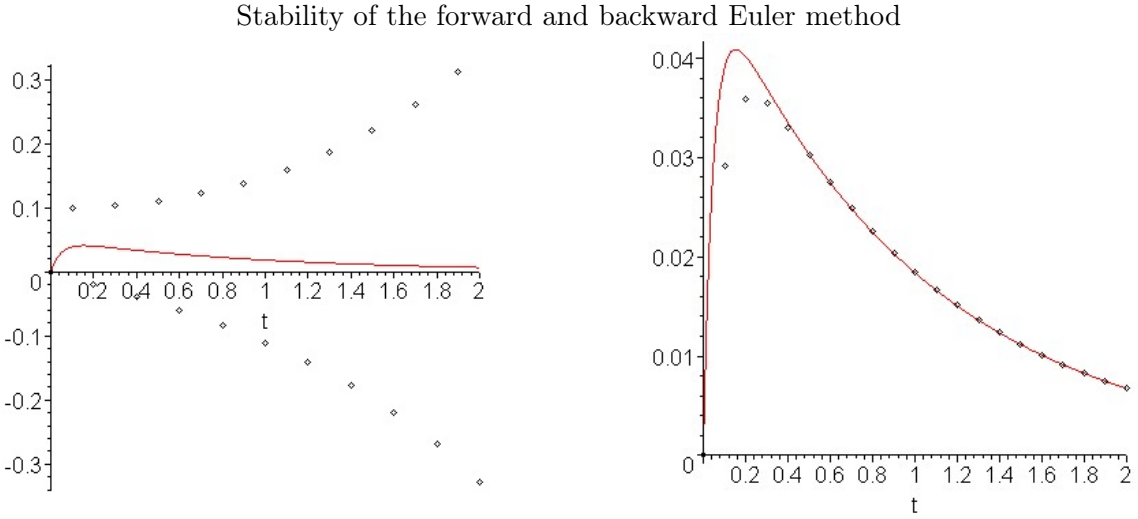

Figure 1: Left: forward Euler method applied to $u'(t) = -21u(t) + e^{-t}$; Right: backward Euler method applied to $u'(t) = -21u(t) + e^{-t}$ [10].

The deviating results of the forward Euler method can be explained using the vector field. To calculate the value $u(t+1)$, the derivative of $u(t)$ is used. In this example, the vector field above the solution points almost vertically downwards, and the vector field below the solution points almost vertically upwards. This causes the zigzagging motion [10].

2.4 IMEX

The IMEX method is a combination of the forward Euler and the backward Euler method. It uses the stability of the backward Euler method, and combines it with the speed of the forward Euler method. It is particularly appealing for our purposes, as we can then discretize the 'stiff' part of the FitzHugh-Nagumo model implicitly, and the computationally demanding nonlinear part explicitly.

We define $u \in C^1[t_0, T]$ to be a continuous differentiable function, $p : \mathbb{R} \to \mathbb{R}$ and $q : \mathbb{R} \to \mathbb{R}$. Consider the differential equation

$$\frac{du}{dt} = p(u) + q(u) \tag{4}$$

When we discretize this in an explicit way with the forward Euler scheme, we obtain

$$\frac{u^{n+1} - u^n}{h} = p(u^n) + q(u^n) \tag{5}$$

and when we discretize this in an implicit way with the backward Euler scheme, we obtain

$$\frac{u^{n+1} - u^n}{h} = p(u^{n+1}) + q(u^{n+1}). \tag{6}$$

An IMEX scheme is obtained for instance by applying a forward Euler method to $p(u)$ and a mix of forward and backward Euler method in $q(u)$. This results in:

$$\frac{u^{n+1} - u^n}{h} = p(u^n) + (1-\gamma)q(u^n) + \gamma q(u^{n+1}) \tag{7}$$

with $0 \leq \gamma \leq 1$.
Choosing $\gamma = 0.5$ results in a scheme called the Crank-Nicolson scheme. In this thesis we will choose $\gamma = 1$, which simplifies the equation to

$$\frac{u^{n+1} - u^n}{h} = p(u^n) + q(u^{n+1}). \tag{8}$$

This scheme is also known as a semi-implicit backward differentiation formula scheme. Usually when this scheme is applied, the nonlinear term is defined as $p(u)$ and is thus treated explicitly and the linear term is defined as $q(u)$ and is treated implicitly. This choice combines the stability of the linear term caused by the implicit method, and the low computational costs of the explicit scheme for the nonlinear term [2, 13].

3 Discretization of the FitzHugh-Nagumo model with Finite differences.

In order to find solutions to the FitzHugh-Nagumo model, it is important to discretize the model. The FitzHugh-Nagumo model is a simplified model of activation and deactivation

dynamics in a spiking neuron. It is given by the equations

$$\begin{cases} \epsilon \frac{\delta v(x,t)}{\delta t} = \epsilon^2 \frac{\delta^2 v(x,t)}{\delta x^2} + v(x,t)(v(x,t)-a)(1-v(x,t)) - w(x,t) + c, \\ \frac{\delta w(x,t)}{\delta t} = bv(x,t) - \gamma w(x,t) + c \end{cases} \tag{9}$$

with $a = 0.1, b = 0.5, \gamma = 2$, c $= 0.05$ and $\epsilon = 0.015$.

If we define $f(v) = v(v-a)(1-v)$, we can rewrite this problem as

$$\begin{cases} \epsilon \frac{\delta v}{\delta t} = \epsilon^2 \frac{\delta^2 v}{\delta x^2} + f(v) - w + c, \\ \frac{\delta w}{\delta t} = bv - \gamma w + c. \end{cases} \tag{10}$$

The problem is restricted to the boundary conditions

$$\begin{cases} v(x,0) = 0 \\ w(x,0) = 0 \\ \frac{\delta v}{\delta x}(0,t) = -i_0(t) \\ \frac{\delta v}{\delta x}(L,t) = 0 \end{cases} \tag{11}$$

with $L = 1$ and $i_0(t) = 50000t^3 exp(-15t)$ [5].

3.1 Finding the recursive equations

We will compute the solutions of the FitzHugh-Nagumo problem on fixed points on a bounded rectangular domain, $x \in [0, L], t \in [0, T]$. We choose uniformly distributed grid points such that the difference in $x-$direction is h and in the $t-$direction is τ. Thus, we can write

$$0 = x_0 \leq x_1 \leq ... \leq x_M = L \qquad x_i = ih \text{ where } h = \frac{L}{M} \tag{12}$$

$$0 = t_0 \leq t_1 \leq t_2 \cdots \leq t_N = T \qquad t_i = i\tau \text{ where } \tau = \frac{T}{N} \tag{13}$$

We will define u_j^n as the value of u at space step j and time step n, which can be calculated by $u(jh, n\tau)$. When calculating the derivatives at a certain point in this grid numerically, we can use the formulas

$$\begin{cases} \frac{\delta^2 u_i^n}{\delta x^2} = \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{h^2} \\ \frac{\delta u_i^n}{\delta t} = \frac{u_i^{n+1} - u_i^n}{\tau} \end{cases} . \tag{14}$$

We apply the semi-implicit backward differentiation formula scheme of equation (8) with $p(v) = f(v) - w + c$ and $q(v) = \epsilon^2 \frac{dv}{dt}$, to the first equation of (10). The IMEX scheme is useful here, since $f(v)$ is a computationally demanding nonlinear term that is easie modelled with an explicit method, while applying the implicit method to $\epsilon^2 \frac{dv}{dt}$ makes sure the approximation is accurate. When using this IMEX scheme and approximating the derivatives as in equations (14) we obtain:

$$\epsilon \frac{\delta v}{\delta t} = \epsilon^2 \frac{\delta^2 v}{\delta x^2} + f(v) - w + c, \tag{15}$$

$$\epsilon \frac{v_i^{n+1} - v_i^n}{\tau} = \epsilon^2 \frac{v_{i+1}^{n+1} - 2v_i^{n+1} + v_{i-1}^{n+1}}{h^2} + f(v_i^n) - w_i^n + c \tag{16}$$

where i is the space step and i runs from 1 to $M-1$ and n is the time step that runs from 0 to N. Multiplying this by τ and setting $\lambda = \frac{\tau}{h^2}$ gives:

$$\epsilon v_i^{n+1} - \epsilon v_i^n = \lambda\epsilon^2(v_{i+1}^{n+1} - 2v_i^{n+1} + v_{i-1}^{n+1}) + \tau f(v_i^n) - \tau w_i^n + \tau c$$

Bringing all terms with index $n+1$ to one side, gives

$$(\epsilon + 2\lambda\epsilon^2)v_i^{n+1} - \lambda\epsilon^2 v_{i+1}^{n+1} - \lambda\epsilon^2 v_{i-1}^{n+1} = \epsilon v_i^n + \tau f(v_i^n) - \tau w_i^n + c\tau$$

The second equation of (10) can be written in a numerical way

$$\frac{\delta w}{\delta t} = bv - \gamma w + c$$
$$\frac{w_i^{n+1} - w_i^n}{\tau} = bv_i^n - \gamma w_i^n + c$$

Multiplying by τ gives

$$w_i^{n+1} - w_i^n = \tau(bv_i^n - \gamma w_i^n + c) \tag{17}$$

Bringing again all terms with $n+1$ to one side, yields:

$$w_i^{n+1} = w_i^n + \tau b v_i^n - \tau\gamma w_i^n + \tau c$$

3.1.1 Boundary condition at x=0

Now, we can use in the boundary condition $v_x(0,t) = -i_0(t)$ by saying $\frac{v_1^n - v_0^n}{h} = -i_0(n\tau)$. We can fill it into equation (15) with $i = 1$. This gives

$$\epsilon\frac{v_1^{n+1} - v_1^n}{\tau} = \epsilon^2\frac{v_2^{n+1} - 2v_1^{n+1} + v_0^{n+1}}{h^2} + f(v_1^n) - w_1^n + c \tag{18}$$

$$\epsilon\frac{v_1^{n+1} - v_1^n}{\tau} = \epsilon^2\frac{\frac{v_2^{n+1} - v_1^{n+1}}{h} - \frac{v_1^{n+1} - v_0^{n+1}}{h}}{h} + f(v_1^n) - w_1^n + c \tag{19}$$

$$\epsilon\frac{v_1^{n+1} - v_1^n}{\tau} = \epsilon^2\frac{\frac{v_2^{n+1} - v_1^{n+1}}{h} + i_0(n\tau + \tau)}{h} + f(v_1^n) - w_1^n + c \tag{20}$$

$$\epsilon(v_1^{n+1} - v_1^n) = \epsilon^2\lambda(v_2^{n+1} - v_1^{n+1}) + \frac{\tau}{h}i_0(\tau n + \tau) + \tau f(v_1^n) - \tau w_1^n + \tau c \tag{21}$$

$$(\epsilon + \epsilon^2\lambda)v_1^{n+1} - \epsilon^2\lambda v_2^{n+1} = \epsilon v_1^n + \frac{\tau}{h}i_0(\tau n + \tau) + \tau f(v_1^n) - \tau w_1^n + \tau c \tag{22}$$

3.1.2 Boundary condition at x = L

The boundary condition $v_x(L,t) = 0$ can be written as $\frac{v_M - v_{M-1}}{h} = 0$ Filling this in into equation (15) with $i = M-1$ gives:

$$\epsilon\frac{v_{M-1}^{n+1} - v_{M-1}^n}{\tau} = \epsilon^2\frac{v_M^{n+1} - 2v_{M-1}^{n+1} + v_{M-2}^{n+1}}{h^2} + f(v_{M-1}^n) - w_{M-1}^n + c \tag{23}$$

$$\epsilon\frac{v_{M-1}^{n+1} - v_{M-1}^n}{\tau} = \epsilon^2\frac{\frac{v_M^{n+1} - v_{M-1}^{n+1}}{h} - \frac{v_{M-1}^{n+1} - v_{M-2}^{n+1}}{h}}{h} + f(v_{M-1}^n) - w_{M-1}^n + c \tag{24}$$

$$\epsilon\frac{v_{M-1}^{n+1} - v_{M-1}^n}{\tau} = \epsilon^2\frac{0 - \frac{v_{M-1}^{n+1} - v_{M-2}^{n+1}}{h}}{h} + f(v_{M-1}^n) - w_{M-1}^n + c \tag{25}$$

$$\epsilon(v_{M-1}^{n+1} - v_{M-1}^n) = -\epsilon^2\lambda(v_{M-1}^{n+1} - v_{M-2}^{n+1}) + \tau f(v_{M-1}^n) - \tau w_{M-1}^n + \tau c \tag{26}$$

$$(\epsilon + \epsilon^2\lambda)v_{M-1}^{n+1} - \epsilon^2\lambda v_{M-2}^{n+1} = \epsilon v_{M-1}^n + \tau f(v_1^n) - \tau w_{M-1}^n + \tau c \tag{27}$$

3.1.3 Overview recursive equations

Combining the recursive equation for $1 < i < M-1$ and the boundary conditions, gives the recursive equations

$$\begin{cases} (\epsilon + \epsilon^2\lambda)v_1^{n+1} - \epsilon^2\lambda v_2^{n+1} = \epsilon v_1^n + \frac{\tau}{h} i_0(\tau n + \tau) + \tau f(v_1^n) - \tau w_1^n + \tau c \\ -\lambda\epsilon^2 v_{i-1}^{n+1} + (\epsilon + 2\lambda\epsilon^2)v_i^{n+1} - \lambda\epsilon^2 v_{i+1}^{n+1} = \epsilon v_i^n + \tau f(v_i^n) - \tau w_i^n + \tau c \quad \text{for } 1 < i < M-1 \\ (\epsilon + \epsilon^2\lambda)v_{M-1}^{n+1} - \epsilon^2\lambda v_{M-2}^{n+1} = \epsilon v_{M-1}^n + \tau f(v_1^n) - \tau w_{M-1}^n + \tau c \\ w_i^{n+1} = (1 - \tau\gamma)w_i^n + \tau b v_i^n + \tau c \end{cases} \tag{28}$$

where $\lambda = \tau/h^2$ and $f(v) = v(v-a)(1-v)$.

3.2 Creating an equation of matrices

We can write the system of equations (28) in matrix form. The structure of this equation is:

$$AY^{n+1} = BY^n + C + \tau F(Y^n) + I(n) \tag{29}$$

The vector $Y^n = (v_1^n, \ldots, v_{M-1}^n, , w_1^n, \ldots, w_{M-1}^n)^T$. A is the $[2M-2] \times [2M-2]$ matrix $\begin{bmatrix} A_1 & 0 \\ 0 & I \end{bmatrix}$ with

$$A_1 = \begin{bmatrix} \epsilon + \epsilon^2\lambda & -\epsilon^2\lambda & 0 & 0 & \ldots & 0 & 0 & 0 & 0 \\ -\epsilon^2\lambda & s & -\epsilon^2\lambda & 0 & \ldots & 0 & 0 & 0 & 0 \\ 0 & -\epsilon^2\lambda & s & -\epsilon^2\lambda & \ldots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & -\epsilon^2\lambda & s & -\epsilon^2\lambda & 0 \\ 0 & 0 & 0 & 0 & \ldots & 0 & -\epsilon^2\lambda & s & -\epsilon^2\lambda \\ 0 & 0 & 0 & 0 & \ldots & 0 & 0 & -\epsilon^2\lambda & \epsilon + \epsilon^2\lambda \end{bmatrix} \tag{30}$$

, where $s = (\epsilon + 2\lambda\epsilon^2)$, 0 is the zero-matrix of $[M-1] \times [M-1]$, and I the identity matrix of $[M-1] \times [M-1]$.

The matrix B is the matrix $\begin{bmatrix} B_1 & B_2 \\ B_3 & B_4 \end{bmatrix}$ with B_1 the diagonal matrix with entries ϵ, B_2 the diagonal matrix with entries $-\tau$, B_3 the diagonal matrix with entries τb and B_4 the diagonal matrix with entries $1 - \tau\gamma$.

C is a column vector of length $[2M-2]$ with entries τc. $F(Y^n)$ is the nonlinear column vector of length $[2M+2]$ with entries $[f(v_1^n), f(v_2^n), \ldots, f(v_{M-2}^n), f(v_{M-1}^n), 0, 0, \ldots, 0)]$ and $I(n)$ is the column vector $[\frac{\tau}{h} i_0(\tau * n + \tau), 0, 0, \ldots, 0]$

4 Model order reduction

When solving the FitzHugh-Nagumo model numerically, it is necessary to use extremely small time steps in order for the solution to be stable. Because of this restriction, it takes a lot of time and storage capacity of the computer to compute solutions for a large amount of time. To lower the computational complexity of the problem, the order of the model can be reduced. In this section, we will create a reduced model of the FitzHugh-Nagumo

model using the proper orthogonal decomposition (POD). To compute a reduced order model, we compute the solutions of the system of equation (29). As a solution we get a matrix $Y_{sol} \in \mathbb{R}^{N \times 2M}$. Matrix Y_{sol} exists out of the solutions for $v(x,t)$ and $w(x,t)$, V_{sol} and W_{sol} respectively such that $Y_{sol} = \begin{pmatrix} V_{sol} \\ W_{sol} \end{pmatrix}$ where $V_{sol}, W_{sol} \in \mathbb{N} \times \mathbb{M}$.

4.1 General outline of model order reduction with POD

Proper orthogonal decomposition is a method that can be used to decrease the order of a model. The general idea of the POD is to approximate the solution vector $y \in \mathbb{R}^{m \times n}$ by the product of a matrix $\Phi_r \in \mathbb{R}^{m \times r}$ and a column vector $a_r^n \in \mathbb{R}^{1 \times r}$

$$y^n \approx \Phi_r a_r^n.$$

Here r is a lot smaller than m. In the columns of Φ_r are vectors that are coefficients of the finite difference approximations of the solution for certain time steps.

One can expect to obtain a good approximation of a solution for the FitzHugh-Nagumo equations by using Φ_r with only a few columns. This is because the solution of the FitzHugh-Nagumo model does not change a lot in space over time. Suppose the solution is completely constant in space over the full time. Naturally, the solution can then be approximated by one single column vector. When there is more variation over time, more column vectors are needed to have a good approximation of the solution.
The following steps are needed in the process of model order reduction with help of POD:

1. Compute solutions y^n for many points in time $n = 1, 2, \ldots, N$
2. Collect the solutions in columns of matrix $Y_{sol} = [y^1, y^2, \ldots, y^N]$
3. Perform a singular value decomposition: $Y_{sol} = U \Sigma Z^T$
4. Let Φ_r be a new matrix consisting of the first r columns of matrix U of the singular value decomposition $Y_{sol} = U \Sigma Z^T$
5. Assume $y^n \approx \Phi_r a(t)$ for some $a(t)$
6. Solve the differential equation for $a(t)$ and compute $y_a^n pprox = \Phi_r a(t)$

How solutions are obtained for the FitzHugh-Nagumo model is described in section (3) of this report. The rest of the steps of the POD method is described in this section.

4.2 Singular value decomposition

We obtained the matrix Y_{sol} consisting of the solutions $Y_{sol} = \begin{pmatrix} v_1 & v_2 & \ldots & v_N \\ w_1 & w_2 & \ldots & w_N \end{pmatrix}$, where v_n is the column vector with solution of $v(x,t)$ at time step n, and w_n is the column vector with solutions of $w(x,t)$ at time step n. The matrix Y_{sol} can be written as a product of three new matrices. We do this using the singular value decomposition (SVD) that is described in theorem 1 [21].

Theorem 1: The singular value decomposition (SVD).
Let $A \in \mathbb{R}^{m \times n}$ and let A^T be its transpose. Then A can be factorized as

$$A = U\Sigma Z^T$$

where

1. $U \in \mathbb{R}^{m \times m}$ is an orthogonal matrix
2. $\Sigma \in \mathbb{R}^{m \times n}$ is a diagonal matrix with entries $\sigma_i \geq 0$ and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_q$, $q = \min(m, n)$
3. $Z \in \mathbb{R}^{n \times n}$ is an orthogonal matrix.

Here, $\sigma_1, \ldots, \sigma_q$ are the singular values, which are the square roots of the eigenvalues of $A^T A$ and AA^T, neglecting the additional $|m - n|$ zero eigenvalues of $A^T A$ if $n > m$ or AA^T if $m > n$.

One nice property of the singular value decomposition $A = U\Sigma Z^T$ is that the eigenvectors of AA^T form the columns of U, and the eigenvectors of $A^T A$ form the columns of Z [7]. We therefore have to find the eigenvalues and corresponding eigenvectors of $Y_{sol}Y_{sol}^T$ and $Y_{sol}^T Y_{sol}$ to find the singular value decomposition of Y_{sol}. Eigenvalues of a matrix A are defined as the values $\lambda_1, \lambda_2, \ldots, \lambda_n$ for which holds that $Av_i = \lambda_i v_i$. Here v_i is the eigenvector corresponding to λi. Unfortunately, no algorithm is able to find all the eigenvalues of a matrix in a finite number of operations when the matrix is bigger than 2×2 [21]. A method called the generalized Schur (QR) method can be used to estimate eigenvalues. Using this method we find approximations of the eigenvalues of $Y_{sol}Y_{sol}^T$ and $Y_{sol}^T Y_{sol}$ and thus the approximation of the singular values of Y_{sol} [14].

To find an eigenvector v_i corresponding to an eigenvalue λ_i of $Y_{sol}Y_{sol}^T$, we have to approximate

$$(Y_{sol}Y_{sol}^T - \lambda_i I)v_i = 0.$$

Doing this for all eigenvalues, we find all sets of positive eigenvalues with eigenvectors $[\lambda_1, v_1], [\lambda_2, v_2], \ldots, [\lambda_q, v_q]$ of $Y_{sol}Y_{sol}^T$, with $\lambda_1 \leq \lambda_2 \leq \ldots, \leq \lambda_q$. Similarly, all sets of positive eigenvalues with eigenvectors $[\lambda_1, v_1^*], [\lambda_2, v_2^*], \ldots, [\lambda_q, v_q^*]$ of $Y_{sol}^T Y_{sol}$ can be found. It can easily be proven that the nonzero eigenvalues of $A^T A$ are equal to the eigenvalues of AA^T [6].

The matrix U is constructed as the $m \times m$ matrix $[v_1, v_2, \ldots, v_m]$, the matrix Σ is an $m \times n$ diagonal matrix with entries $\sqrt{\sigma_1}, \sqrt{\sigma_2}, \ldots, \sqrt{\sigma_q}$, $q = \min(m, n)$ and the matrix Z is the $n \times n$ matrix $[v_1^*, v_2^*, \ldots, v_n^*]$ such that

$$Y_{sol} = U\Sigma Z^T [7].$$

In the SVD, U contains the spatial structures and Z contains the time dependent structures [20]. Intuitively can be said that the matrix Σ determines the importance of each corresponding vector in U and W: a higher value of σ means the corresponding vector is more important. Since Σ is a diagonal matrix with decreasing entries, we can say that for a certain r, σ_i is significant for $i \leq r$ and σ_i is not significant for $i > r$. The Eckhart-Young Theorem can be used to show that this intuition is correct [21].

Theorem 2: The Eckart-Young Theorem.
Let $A \in \mathbb{R}^{m \times n}$. If $k < r = \text{rank}(A)$ and

$$A_k = \sum_{i=1}^{k} \sigma_i u_i z_i^T \tag{31}$$

Then

$$\min_{\text{rank}(B)=k} ||A - B||_2 = ||A - A_k||_2 = \sigma_{k+1}. \tag{32}$$

Here, $A = U\Sigma Z^T$ with σ_i the i^{th} diagonal entry of Σ, u_i is the i^{th} column of U and z_i the i^{th} column of Z.

In other words, the best approximation of a matrix A with rank k is A_k as defined in equation (31), and the norm of the difference between A and its approximation is σ_{k+1}.
It can be noticed that equation (31) is nothing else than

$$A_k = U_k \Sigma_k Z_k^T$$

where U_k is the matrix of the first k columns of U, Σ_k is the diagonal matrix with entries $\sigma_1, \sigma_1, \ldots, \sigma_k$ and Z_k is the matrix of the first k columns of Z. Thus, when the singular value decomposition of matrix A is known, it is easy to compute the best approximation of A of rank k: A_k [21].

4.3 Reducing the order of the model

By equation (32) of the Eckhart-Young theorem,

$$Y_{sol_r} = \sum_{i=1}^{r} \sigma_i u_i z_i^T$$

is the best approximation of rank r of the matrix Y_{sol}. The 2-norm error between Y_{sol_r} and Y_{sol} is σ_{r+1}. To reduce the model, we have to define a tolerance for the error due to the model order reduction. Once we have defined the tolerance, we can determine what is the smallest rank possible to reduce A to, for which

$$\frac{\sigma_{r+1}}{\sigma_1} \leq \text{tolerance}$$

holds. For the reduced solution Y_{sol_r}, the memory required is $r(m+n)$. The memory required for the original solution Y_{sol} is mn. Thus, if $r << min(m,n)$, the memory requirement is a lot smaller [8].

To apply the POD method, we put the first columns of U in a new matrix and call this matrix Φ_r. This matrix Φ_r has dimensions $r \times m$. To create a smaller system of equations, we use the approximation $Y^n \approx \Phi_r a^n$. We plug $Y^n \approx \Phi_r a^n$ into equation (29) and perform a Galerkin projection by multiplying the equations from the left by Φ_r^T. This gives us

$$\Phi_r^T A \Phi_r a^{n+1} = \Phi_r^T B \Phi_r a^n + \Phi_r^T C + \Phi_r^T F(\Phi_r a^n) + \Phi_r^T I^n.$$

We make the substitutions

$$\begin{cases} \Phi_r^T A \Phi_r := A_{red}, \\ \Phi_r^T B \Phi_r := B_{red}, \\ \Phi_r^T C := C_{red}, \\ \Phi_r^T F(\Phi_r a^n) := F_{red}^n(\Phi_r a^n), \\ \Phi_r^T I^n; = I_{red}^n \end{cases}$$

and we rewrite the equation as

$$A_{red} a^{n+1} = B_{red} a^n + C_{red} + \tau F_{red}^n + I_{red}^n \tag{33}$$

[12]. The reduced matrices are in general much smaller than the original matrices because Φ_r generally has only few columns. This makes the new system of equations faster to solve. The matrices A_{red}, B_{red} and C_{red} have to be calculated only once, and can be used for all time steps. The column I_{red}^n has to be calculated for every time step. However, remember that the vector I^n has only one nonzero entry, which makes it very fast to calculate I_{red}^n. The expensive part is the matrix $F_{red}^n(\Phi_r a^n)$. The input for this matrix has to be updated for every time step with the new results of a^n and to obtain F_{red} the input has to go into the formula $f(v) = v(v-a)(1-v)$. remember $a = 0.1$ is a constant in this formula.

Once equation (33) is solved for a^n, $n = 0, 1, \ldots, N$, the new approximation of Y_n, Y_{nsol} can be calculated using the approximation $Y_s^n ol \approx \Phi_r^n a^n$.

With this method of model order reduction, the computation time and the needed space are decreased. However, still a lot of computation time is needed due to the nonlinear term $F_{red}^n(\Phi_r a^n)$. To reduce the computation time even further, we want to apply model order reduction also for the nonlinear term. In several papers, this has been done successfully for other models using an empirical interpolation method (EIM) and a discrete empirical interpolation method (DEIM). Therefore, for further research we want to advise to apply the EIM or DEIM method described in [3] and in [5].

5 Results

In this section, we will describe the results found for the heat equation and the FitzHugh-Nagumo model. We will see the numerical approximation to the solutions of both systems and the error of the approximation of the solution of the heat equation. Also the model order reduction described in section 4 is applied to the FitzHugh-Nagumo model, and the results are shown.

5.1 Heat equation

We solved the heat equation of equation (34) numerically by implementing equation (39) in MATLAB. We solved the heat equation on grid points in the time domain $t \in (0, 1)$ and spatial domain $x = (0, 1)$. The grid points are equally spaced with $\delta t = \tau = 2.5 \cdot 10^{-5}$ and $\delta x = h = 10^{-2}$.

In Fig. 2, we can see the approximation of the solution of the heat equation computed from equation (A.2). To compute the error of the numerical solution compared to the analytic

solution, we use the relative L^2 error. This error is defined as

$$\sqrt{\frac{\int_0^L (u_{approx}(x) - u_{real}(x))^2 dx}{\int_0^L (u_{real}(x))^2 dx}}$$

where u_{approx} is a function that fits the numerical results, and u_{real} is the analytic solution of the heat equation. Fig. 3 shows this L^2 error, the numerical solution of Fig. 2 is compared to the analytic solution $v(x,t) = cos(\pi x)t$.

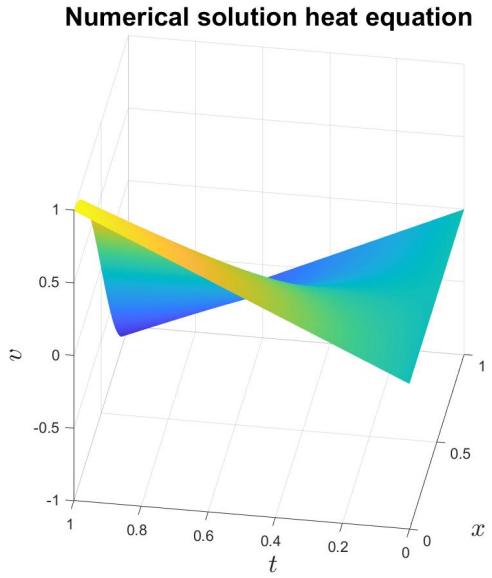

Figure 2: The numerical solution of equation (39) with $T = 1$ and $L = 1$

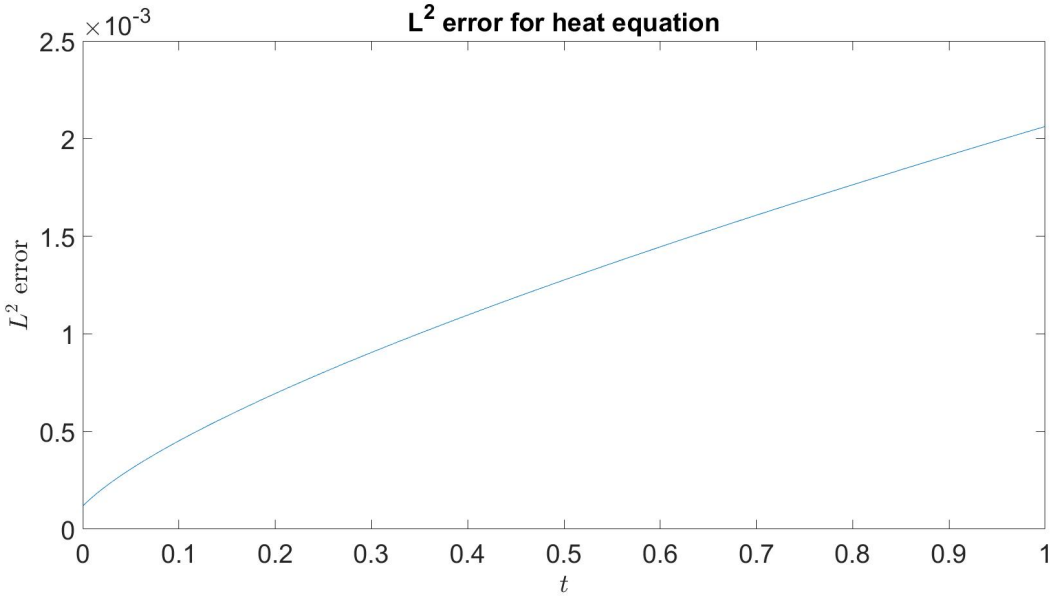

Figure 3: The L^2 error of the numerical approximation of the solution to the heat equation, compared to the analytic solution

In Fig. 3 can be seen that the relative L^2 error is of the order 10^{-3}. This means the numerical solution approximates the real solution good. We also see that the error increases when time increases. This can be explained by the dynamics of the heat equation. When we proceed in time, the rate of change over space of the solution is higher. This causes that more space steps are needed to have a good approximation to the solution. If not, the error increases.

5.2 FitzHugh-Nagumo model

To solve the FitzHugh-Nagumo equation numerically, we implemented equation (29) in MATLAB. We changed the value of ϵ to 1 so that the nonlinear part of the equation has less influence on the behavior of the solution, and the solution is more stable. Without this change of the value ϵ, it took a lot of time steps to have a stable solution as a result. We solved the FitzHugh-Nagumo equation on grid points in the time domain $t \in (0, 80)$ and spatial domain $x = (0, 1)$. The grid points are equally spaced with $\delta t = \tau = 8 \cdot 10^{-6}$ and $\delta x = h = 10^{-2}$.

In Fig. 4, we plotted the full order approximation of the solution for v and w of the FitzHugh Nagumo equation. We see that the solution for v as well as the solution for w have a peak when $t \in (0, 2)$, and that it remains almost constant when $t > 10$. In Fig. 5 we have a closer look at the peak in the solution for v and w. It is visible that the peak is highest at $x = 0$ and decreases when x increases. This peak is caused by the boundary condition $\frac{\delta v}{\delta x}(0, t) = -i_0(t)$. The peak is caused to be flattened at $x = 1$ because of the boundary condition $\frac{\delta v}{\delta x}(L, t) = 0$.

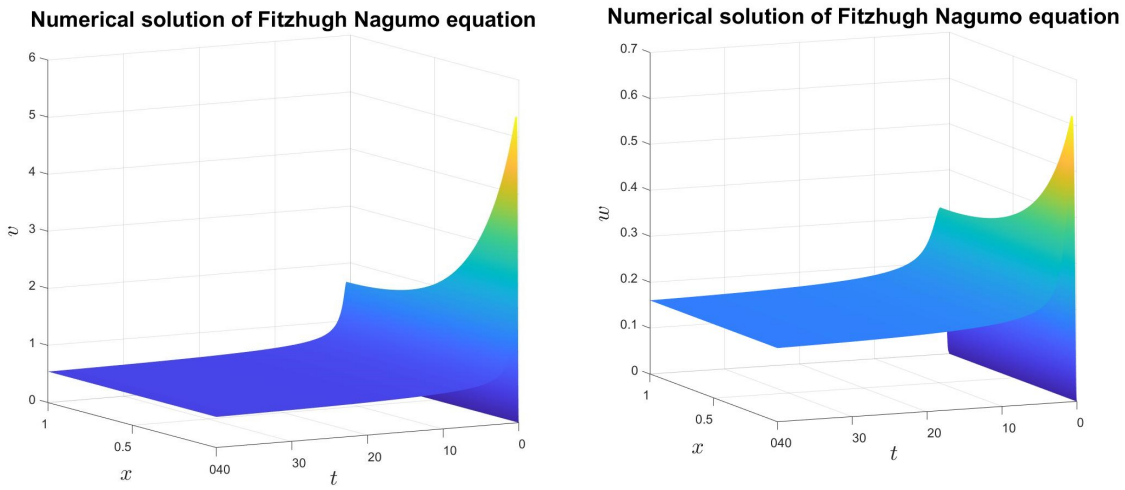

Figure 4: The numerical approximation of the solution for v (left) and w (right) of equation 10 for $t \in (0, 40)$ and $x \in (0, 1)$

Fig. 6 shows the fast decay of 100 singular values of the snapshot solutions for v and w. Small decaying eigenvalues indicate that for many values of t_x, the solution at $t = t_x$ is a linear combination of solutions at $t \neq t_x$. Since the solution shown in 4 remains almost constant over a large time interval, this is the case. Thus the fast decay of singular values can be expected.

Using the singular value decomposition, we computed a matrix Φ_r and a^n as described

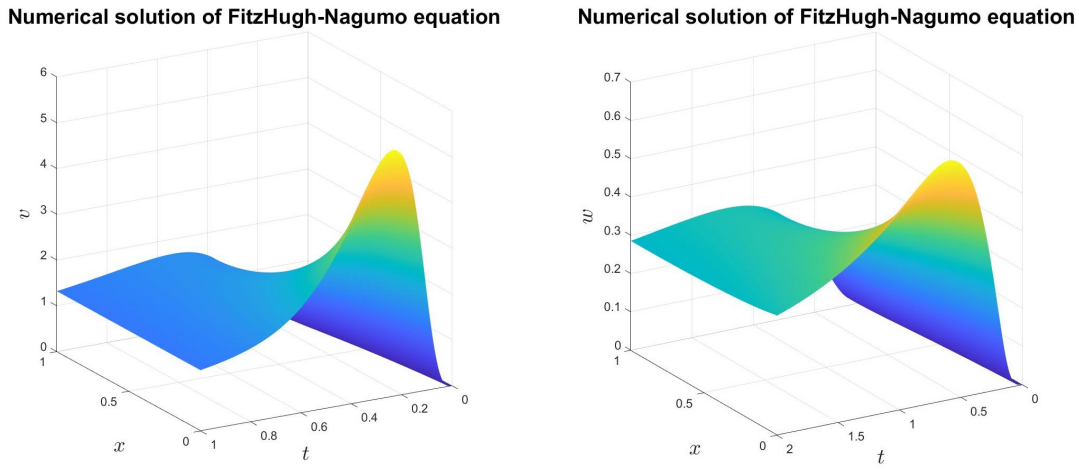

Figure 5: Left: The numerical approximation of the solution for v of equation 10 for $t \in (0,1)$ and $x \in (0,1)$. Right: The numerical approximation of the solution for w of equation 10 for $t \in (0,2)$ and $x \in (0,1)$.

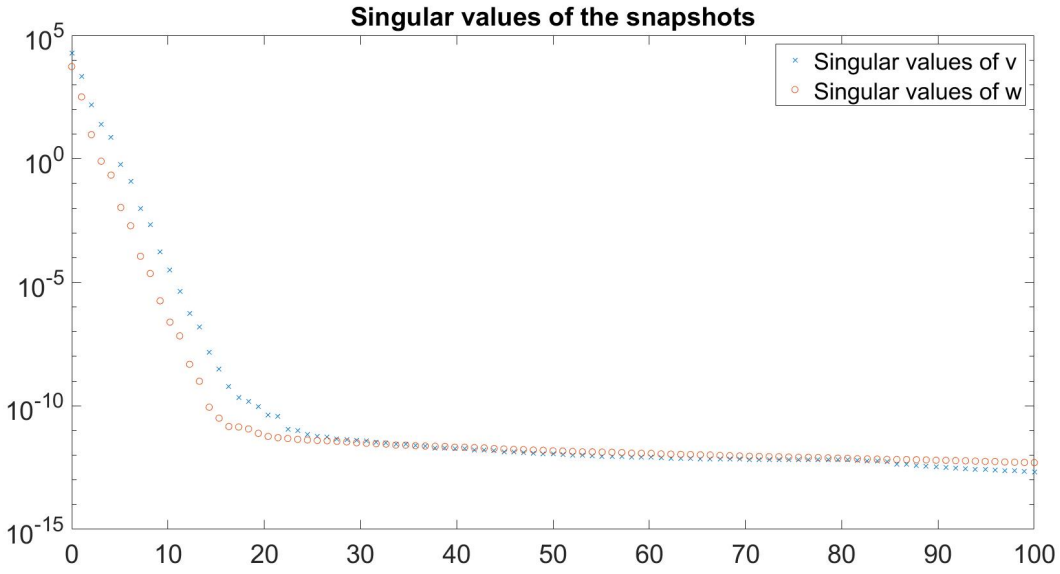

Figure 6: The singular values of 100 snapshot solutions for v and w from the full order system (29)

in section 4.3 for different values of r. We computed the L^2 error between the full order model and the reduced order model by the formula

$$L^2 = \sqrt{\frac{\int_0^L (v_{red}(x) - v_{full}(x))^2 dx}{\int_0^L (v_{full}(x))^2 dx}}.$$

Here, v_{full} is a function that fits the solution of v of the full order model and v_{red} is a function that fits the solution of v of the reduced order model. The described L^2 error between the solution obtained by $\Phi_r^n a^n$ and the obtained solution from Fig. 4, is shown in Fig. 7. The maximum of the L^2 error over time for each r is shown in Fig. 8. Since

the singular values shown in Fig. 6 decay fast, it is expected that only few columns r are needed to obtain a good approximation to the full order numerical solution.

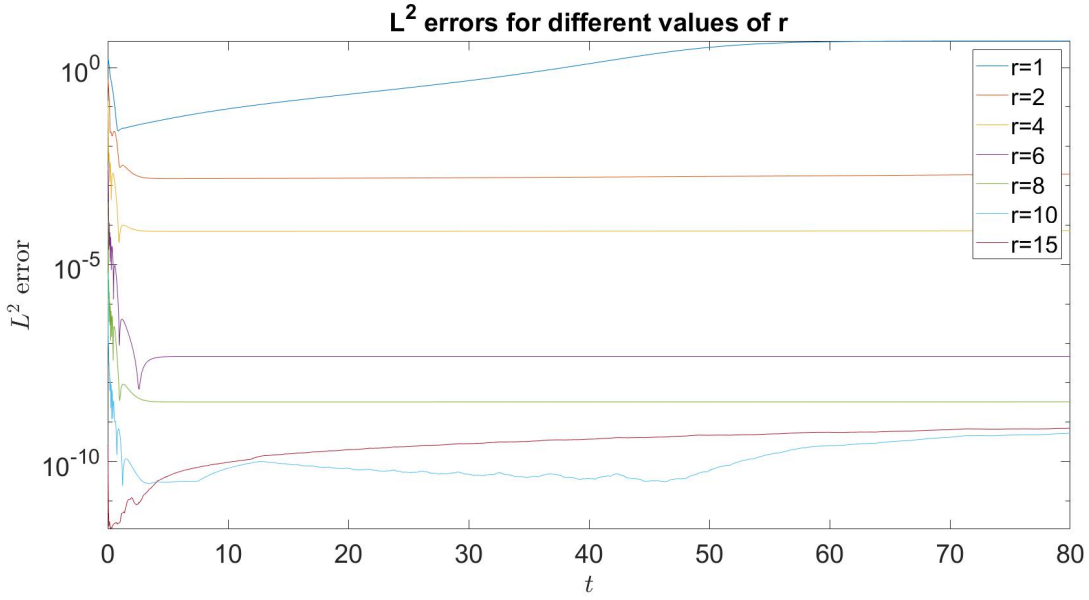

Figure 7: The L^2 error between the solution of the reduced order system $\Phi_r^n a^n$ and the full order solution shown in Fig. 4

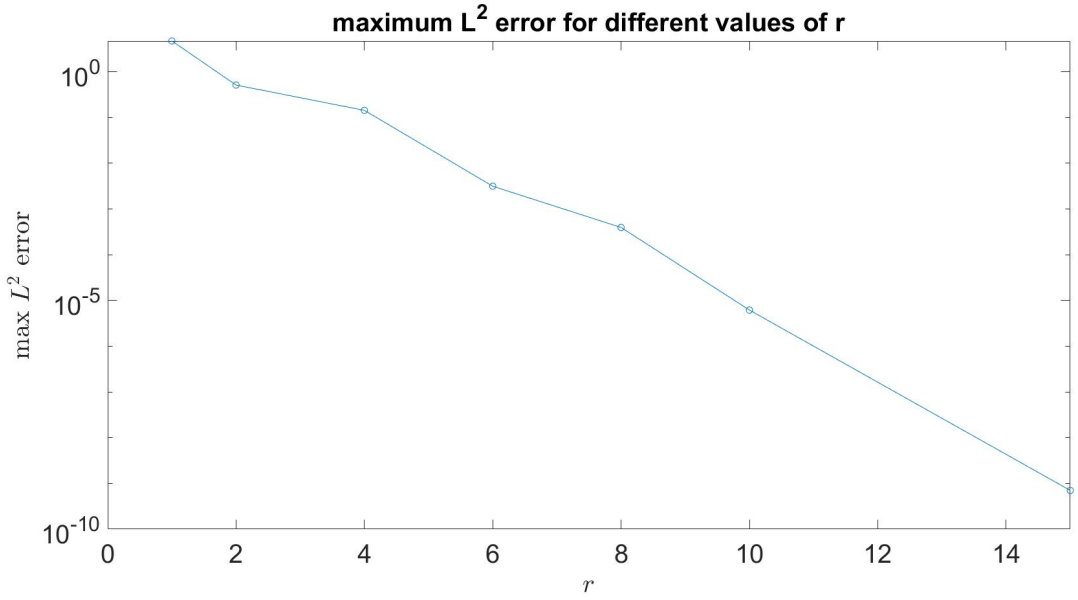

Figure 8: The maximum of the L^2 error of Fig. 7 for different ranks r

In Fig. 7 and Fig. 8, we can see that the error decreases when r increases. This is exactly what we would suspect to happen. However, we can see that the error when $r = 15$ is bigger than the error when $r = 10$ on a large interval. This can be explained by the machine precision. This precision has an accuracy of approximately 10^{-16}. Since the square root is taken to obtain the error, the accuracy is only about 10^{-8} and thus the L^2 error shown in Fig. 7 for $r = 15$ is effected by that.

In Fig. 7 we can also see that the error between the full order and reduced order solution always has a peak at the beginning. This can be expected and is explained by the peak in the solution shown in 4.

6 Conclusions

In this thesis we have discretized the FitzHugh-Nagumo model and found a numerical solution to this problem. This numerical solution is used to obtain a reduced order model that approximates the full order numerical solution. We did this using POD and the Galerkin projection. This method for model order reduction works for the FitzHugh-Nagumo model. In Fig. 8, it can be noted that the maximum error caused by POD decreases when the rank of the approximation increases. The L^2 error caused by the POD obtained with a rank $r = 10$ is only of the order 10^{-5}, which is insignificant compared to the error caused by the discretization. To choose the best rank r, we can approximate the error of the numerical solution compared to the analytical solution, and make sure the error caused by the POD is smaller than this. This way, the POD does not cause significant errors.

6.1 Limitations

Due to the non-linearity in the FitzHugh-Nagumo problem, a lot of time steps are needed to ensure stability of the numerical solution. This was not possible for a time period longer than $t \in [0, 1]$, due to lack of storage capacity. Besides, obtaining these results required a lot of computation time, which was not available. To solve this limitation, the variable ϵ is set to 1. This reduces the influence of the nonlinear part in the FitzHugh-Nagumo equation.

6.2 Future research

To obtain more relevant results, the same numerical solution as described in this report can be obtained for $\epsilon = 0.015$, but on a faster computer with more capacity.

In the reduced order model created in this report, there is still a nonlinear part $\Phi_r^T F(\Phi_r a^n)$, which has to calculated in each time step. This takes time to compute, and causes that the solution of the reduced order model still takes a large amount of time to compute. A method called the emperical interpolation method (EIM) [3, 4], and its discrete variant the discrete empirical interpolation method (DEIM) [5] can be used to improve the dimension reduction efficiency of POD with Galerkin projection. DEIM in combination wit POD is faster than only applying POD, and can provide a nearly optimal subspace approximation of this nonlinear term [5].

References

[1] M.L. Abell and J.P Braselton. *Introductory Differential Equations*. Academic Press, 2018.

[2] U.M. Ascher, S.J. Ruuth, and B.T.R. Wetton. Implicit-Explicit Methods for Time-Dependent Partial Differential Equations. *SIAM Journal on Numerical Analysis*, 32(3):797–823, 1995.

[3] M. Barrault, Y. Maday, N. C. Nguyen, and A. T. Patera. An 'Empirical Interpolation' Method: application to efficient Reduced-basis Discretization of Partial Differential Equations. *Comptes Rendus Mathematique*, 339(9):667–672, 2004.

[4] Y. Bourgault, M. Ethier, and V. G. Leblanc. Simulation of electrophysiological waves with an unstructured finite element method. *ESAIM: Mathematical Modelling and Numerical Analysis*, 37(4):649–661, 2003.

[5] S. Chaturantabut and D. C. Sorensen. Nonlinear Model Reduction via Discrete Empirical Interpolation. *SIAM Journal on Scientific Computing*, 32(5):2737–2764, 2010.

[6] S. Friedberg, A. Insel, and L. Spence. *Linear Algebra*. Pearson Education Limited, 2014.

[7] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press., 2013.

[8] M. Grepl and K. Veroy-Grepl. *Model Order Reduction Techniques SVD and POD*, 2014. Available at `https://www.igpm.rwth-aachen.de/Download/ss14/mor/ROM_L10_SS2014.pdf`.

[9] B. Haasdonk. Reduced Basis Methods for Parametrized PDEs — A Tutorial, Introduction for Stationary and Instationary Problems. In P. Benner, M. Ohlberger, A. Cohen, and K.E. Wilcox, editors, *Model Reduction and Approximation: Theory and Algorithms*, pages 65 – 136.

[10] D.W. Harde. *Topic 14.6: Stiff Differential Equations*. University of Waterloo, Department of Electrical and Computer Engineering, 2005.

[11] A. R. Kellems, S. Chaturantabut, D. C. Sorensen, and S. J. Cox. Morphologically accurate Reduced Order Modeling of Spiking Neurons. *Journal of computational neuroscience*, 28(3):477–94, 2010.

[12] N. Kutz. ROM introduction. Available at `https://www.youtube.com/watch?v=YtFuVwrZxC4`.

[13] M. Liu, W. Cao, and Z. Fan. Convergence and stability of the semi-implicit euler method for a linear stochastic differential delay equation. *Journal of Computational and Applied Mathematics*, 170(2):255–268, 2004.

[14] MathWorks Benelux. Eigenvalues and Eigenvectors: MATLAB eig. Available at `https://nl.mathworks.com/help/matlab/ref/eig.html`.

[15] A. Megretski. *Lecture 2 : Differential Equations As System Models 1*. Massachusetts Institute of Technology, 2003.

[16] K. S. Mohamed. *Machine Learning for Model Order Reduction*, volume 664. Springer, 2018.

[17] T. Peets and K. Tamm. Mathematics of Nerve Signals. In Arkadi Berezovski and Tarmo Soomere, editors, *Applied Wave Mathematics II: Selected Topics in Solids, Fluids, and Mathematical Methods and Complexity*, pages 207–238. Springer International Publishing, 2019.

[18] A. Quarteroni, A. Manzoni, and F. Negri. *Reduced Basis Methods for Partial Differential Equations: an introduction*, volume 92. Springer, 2016.

[19] G. Rozza, D.B.P. Huynh, and A.T. Patera. Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations: Application to transport and continuum mechanics. *Archives of Computational Methods in Engineering*, 15:229–275, 2008.

[20] P. J. Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28, 2010.

[21] K. Smetana. Scientific Computing Lecture Notes. 2019.

[22] T. Tonn. *Reduced-Basis Method (RBM) for Non-Affine Elliptic Parametrized PDEs*. PhD thesis, Universität Ulm, 2012.

[23] A. Wibisono. *Forward and Backward Euler*. Github, 2016.

[24] P. Wilson and H.A. Mantooth. *Model-based Engineering for Complex Electronic Systems*. Newnes, 2013.

A Heat equation

To be able to validate all steps taken and to verify the code written for the FitzHugh-Nagumo model, it is useful to repeat each step in a simplified model. For this model, a solution should be found by hand, such that we can check if everything is done in the right way. To create this simplified model, we assume $w = 0$ and neglect the nonlinear part $f(v)$ and we obtain the heat equation. This new model is given by the equation

$$\epsilon \frac{\delta v}{\delta t} = \epsilon^2 \frac{\delta^2 v}{\delta x^2} + f(x,t) \tag{34}$$

and is restricted to the boundary conditions

$$\begin{cases} v(x,0) = 0 \\ \frac{\delta v}{\delta x}(0,t) = 0 \\ \frac{\delta v}{\delta x}(L,t) = 0 \end{cases} . \tag{35}$$

A.1 Recursive equations

When applying the semi-implicit backward differentiation formula scheme of equation (8) with $p(v) = f(x,t)$ and $q(v) = \epsilon^2 \frac{d^2 v}{dx^2}$, to the heat equation, we get

$$\epsilon \frac{v_i^{n+1} - v_i^n}{\tau} = \epsilon^2 \frac{v_{i+1}^{n+1} - 2v_i^{n+1} + v_{i-1}^{n+1}}{h^2} + f(ih, n\tau). \tag{36}$$

Here, n is the time step and runs from 0 to N. The space step i runs from 1 to $M-1$. For $2 < i < M-2$, multiplying by τ and bringing all terms of time $n+1$ to the left side gives:

$$(\epsilon + 2\epsilon^2\lambda)v_i^{n+1} - \epsilon^2\lambda v_{i+1}^{n+1} - \epsilon^2\lambda v_{i-1}^{n+1} = \epsilon v_i^n + \tau f(ih, n\tau) \tag{37}$$

with $\lambda = \tau/h^2$

A.1.1 Boundary condition at x=0

The boundary condition $\frac{dv}{dx}(0,t) = 0$ can be written as $\frac{v_1 - v_0}{h} = 0$. Filling this in into equation (36) with $i = 1$ gives:

$$\begin{aligned}
\epsilon \frac{v_1^{n+1} - v_1^n}{\tau} &= \epsilon^2 \frac{v_2^{n+1} - 2v_1^{n+1} + v_0^{n+1}}{h^2} + f(h, n\tau) \\
\epsilon \frac{v_1^{n+1} - v_1^n}{\tau} &= \epsilon^2 \frac{\frac{v_2^{n+1} - v_1^{n+1}}{h} - \frac{v_1^{n+1} - v_0^{n+1}}{h}}{h} + f(h, n\tau) \\
\epsilon \frac{v_1^{n+1} - v_1^n}{\tau} &= \epsilon^2 \frac{\frac{v_2^{n+1} - v_1^{n+1}}{h} - 0}{h} + f(h, n\tau) \\
\epsilon \frac{v_1^{n+1} - v_1^n}{\tau} &= \epsilon^2 \frac{v_2^{n+1} - v_1^{n+1}}{h^2} + f(h, n\tau) \\
\epsilon(v_1^{n+1} - v_1^n) &= \epsilon^2\lambda(v_2^{n+1} - v_1^{n+1}) + \tau f(h, n\tau) \\
(\epsilon + \epsilon^2\lambda)v_1^{n+1} - \epsilon^2\lambda v_2^{n+1} &= \epsilon v_1^n + \tau f(h, n\tau)
\end{aligned}$$

Since $\frac{v_1^n - v_0^n}{h} = 0$, we know v_0^n is equal to v_1^n.

A.1.2 Boundary condition at x = L

The boundary condition $\frac{dv}{dx}(L,t) = 0$ can be written as $\frac{v_M - v_{M-1}}{h} = 0$. Filling this in into equation (36) with $i = M-1$ gives:

$$\begin{aligned}
\epsilon\frac{v_{M-1}^{n+1} - v_{M-1}^{n}}{\tau} &= \epsilon^2\frac{v_M^{n+1} - 2v_{M-1}^{n+1} + v_{M-2}^{n+1}}{h^2} + f((M-1)h, n\tau)\\
\epsilon\frac{v_{M-1}^{n+1} - v_{M-1}^{n}}{\tau} &= \epsilon^2\frac{\frac{v_M^{n+1} - v_{M-1}^{n+1}}{h} - \frac{v_{M-1}^{n+1} - v_{M-2}^{n+1}}{h}}{h} + f((M-1)h, n\tau)\\
\epsilon\frac{v_{M-1}^{n+1} - v_{M-1}^{n}}{\tau} &= \epsilon^2\frac{0 - \frac{v_{M-1}^{n+1} - v_{M-2}^{n+1}}{h}}{h} + f((M-1)h, n\tau)\\
\epsilon\frac{v_{M-1}^{n+1} - v_{M-1}^{n}}{\tau} &= -\epsilon^2\frac{v_{M-1}^{n+1} - v_{M-2}^{n+1}}{h^2} + f((M-1)h, n\tau)\\
\epsilon(v_{M-1}^{n+1} - v_{M-1}^{n}) &= -\epsilon^2\lambda(v_{M-1}^{n+1} - v_{M-2}^{n+1}) + \tau f((M-1)h, n\tau)\\
(\epsilon + \epsilon^2\lambda)v_{M-1}^{n+1} - \epsilon^2\lambda v_{M-2}^{n+1} &= \epsilon v_{M-1}^{n} + \tau f((M-1)h, n\tau)
\end{aligned}$$

Since $\frac{v_M^n - v_{M-1}^n}{h} = 0$, we know v_M^n is equal to v_{M-1}^n.

A.1.3 Overview recursive equations

To summarize, the recursive equations found are described by

$$\begin{cases}
(\epsilon + 2\epsilon^2\lambda)v_i^{n+1} - \epsilon^2\lambda v_{i+1}^{n+1} - \epsilon^2\lambda v_{i-1}^{n+1} = \epsilon v_i^n + \tau f(ih, n\tau) \quad \text{for } 1 < i < M-1\\
(\epsilon + \epsilon^2\lambda)v_1^{n+1} - \epsilon^2\lambda v_2^{n+1} = \epsilon v_1^n + \tau f(h, n\tau)\\
(\epsilon + \epsilon^2\lambda)v_{M-1}^{n+1} - \epsilon^2\lambda v_{M-2}^{n+1} = \epsilon v_{M-1}^n + \tau f((M-1)h, n\tau).
\end{cases} \tag{38}$$

From the boundary conditions, we know $v_0^n = v_1^n$ and $v_M^n = v_{M-1}^n$

A.2 Equations in matrix form

We can write equations (38) in matrix form. The structure of this equation is

$$AY^{n+1} = BY^n + C^n. \tag{39}$$

Here, the vector $Y^n = (v_1^n, v_1^n, ..., v_{M-1}^n,)^T$, the matrix A^n is a matrix of $[M-1] \times [M-1]$

$$A^n = \begin{bmatrix}
\epsilon + \epsilon^2\lambda & -\epsilon^2\lambda & 0 & 0 & \dots & 0 & 0 & 0 & 0\\
-\epsilon^2\lambda & s & -\epsilon^2\lambda & 0 & \dots & 0 & 0 & 0 & 0\\
0 & -\epsilon^2\lambda & s & -\epsilon^2\lambda & \dots & 0 & 0 & 0 & 0\\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots\\
0 & 0 & 0 & 0 & \dots & -\epsilon^2\lambda & s & -\epsilon^2\lambda & 0\\
0 & 0 & 0 & 0 & \dots & 0 & -\epsilon^2\lambda & s & -\epsilon^2\lambda\\
0 & 0 & 0 & 0 & \dots & 0 & 0 & -\epsilon^2\lambda & \epsilon + \epsilon^2\lambda
\end{bmatrix}$$

with $s = (\epsilon + 2\lambda\epsilon^2)$. The matrix B is the matrix

$$B = \begin{bmatrix} \epsilon & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & \epsilon & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & \epsilon & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \epsilon & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & \epsilon & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & \epsilon \end{bmatrix} \tag{40}$$

and C^n is a nonlinear column vector $[\tau f(h, n\tau), \tau f(2h, n\tau), \dots, \tau f((M-1)h, n\tau)]$.

A.2.1 Finding a solution by hand

To test if the recursive equations, matrices and MATLAB code of the heat equation are implemented correctly, we need to choose a solution $v(x,t)$ that fulfills the boundary conditions

$$\begin{cases} v(x,0) = 0 \\ \frac{\delta v}{\delta x}(0,t) = 0 \\ \frac{\delta v}{\delta x}(L,t) = 0 \quad \text{with } L \in \mathbb{N} \end{cases} \tag{41}$$

One function that fulfills all these conditions is $v(x,t) = cos(\pi x)t$. To find the corresponding function $f(x,t)$ to this solution, we fill $v(x,t) = cos(\pi x)t$ in into equation (34). This gives

$$\epsilon \frac{\delta v}{\delta t} = \epsilon^2 \frac{\delta^2 v}{\delta x^2} + f(x,t) \tag{42}$$

$$\iff \epsilon \cos(\pi x) = -\epsilon^2 \pi^2 \cos(\pi x)t + f(x,t) \tag{43}$$

$$\iff f(x,t) = \epsilon \cos(\pi x)(1 + \epsilon \pi^2 t). \tag{44}$$