SOLVING THE FISHER/KPP EVOLUTION ON THE GLOBE

Written by

AMÉLIE JUSTINE LOHER

Supervised by

PROF. DR. RALF HIPTMAIR

BACHELOR THESIS

EIDGENÖSSISCHE TECHNISCHE HOCHSCHULE ETH ZURICH
Department of Mathematics

MAY 2020

ABSTRACT

The Fisher equation was originally intended to model the propagation of advantageous genes. We use this reaction-diffusion equation equipped with non-local boundary conditions to model the propagation of individuals across geographical maps. These boundary condition account for a non-local "diffusion" across channels. The results represent the dispersal of the Homo Sapiens across the globe. The discretisation is based on finite element methods. Moreover, the Strang splitting method is employed to cope with the diffusion and reaction term separately.

TABLE OF CONTENTS

ACKNOWLEDGEMENT

The author expresses her distinct gratifications to Prof. Dr. Hiptmair for the initiation of this thesis and for his continuous guidance. Throughout the development of this project Prof. Dr. Hiptmair has provided helpful advice, supportive suggestions and encouraging thoughts.

Chapter One

Introduction

1.1 Origins of the Fisher-KPP Equation

In 1937, two articles were published concerning the spatial propagation of advantageous genes in a population. The author of the first article was Fisher [4], who studied the propagation of dominant genes in one spatial dimension. He modelled the proportion of the population located at point x at time t possessing the dominant gene with $u(t,x)$, and thus there holds $0 \leq u \leq 1$. To take into account natural selection, he used Verhulst's logistics equation

$$\frac{\partial u}{\partial t} = \lambda u(1-u), \tag{1.1}$$

for $\lambda \in \mathbb{R}_{>0}$. Moreover, Fisher modelled the diffusion of the offspring of an individual with a favourable gene in a neighbourhood of x by adding a diffusion term,

$$\frac{\partial u}{\partial t} = \lambda u(1-u) + c\frac{\partial^2 u}{\partial x^2}. \tag{1.2}$$

Fisher then showed the existence of infinitely many travelling wave solutions for (1.2), that is solutions of the form $u(t,x) = u(x-vt)$ satisfying

$$0 \leq u(t,x) \leq 1, \quad \lim_{x\to-\infty} u(t,x) = 0, \quad \lim_{x\to\infty} u(t,x) = 1, \tag{1.3}$$

provided that $v \geq 2\sqrt{\lambda c}$. Thus, travelling waves of constant speed v along the x-axis connect the steady state $u = 1$ of the advantageous gene with the state $u = 0$ lacking any such gene.

The second article was published by Kolmogorov, Petrovsky and Piskunov [8]. Similarly, it concerned the propagation of dominant genes. Based on Mendelian genetics, they considered

$$\frac{\partial u}{\partial t} = f(u) + c\frac{\partial^2 u}{\partial x^2}, \tag{1.4}$$

where u represents again the density of favourable genes, and where $f(u)$ satisfies

$$f(0) = 0 = f(1), \quad f'(0) > 0, \tag{1.5}$$

and for $0 < u < 1$,

$$f(u) > 0, \qquad f'(u) < f'(0). \tag{1.6}$$

Assuming that the initial condition satisfies

$$0 \leq u(0,x) \leq 1, \tag{1.7}$$

as well as

$$\begin{aligned} u(0,x) &= 0 \qquad \forall x < x_1, \\ u(0,x) &= 1 \qquad \forall x > x_2 \geq x_1, \end{aligned} \tag{1.8}$$

the authors proved that the gene propagates at constant speed $v = 2\sqrt{f'(0)c}$. Furthermore, under these assumptions on the initial condition, (1.4) admits a unique solution which fulfils $0 < u(t,x) < 1$ for all $(t,x) \in \mathbb{R} \times \mathbb{R}_{\geq 0}$, and which converges towards a wave profile propagating at finite speed v. Thereof, we may conclude that the initial gradient governs the speed of propagation. It has been shown that for compactly supported initial datum, wave fronts evolve to solutions of minimal speed of propagation [10].

Let me emphasise that in case of the Fisher's equation (1.2), we have

$$f(u) = \lambda u(1-u). \tag{1.9}$$

Notice that this implies

$$f'(u) = \lambda(1-2u). \tag{1.10}$$

Therefore we realise that for $\lambda \in \mathbb{R}_{>0}$,

$$f(0) = 0 = f(1), \quad f'(0) = \lambda > 0, \tag{1.11}$$

and for $0 < u < 1$,

$$f(u) > 0, \quad f'(u) = \lambda(1-2u) < \lambda = f'(0). \tag{1.12}$$

Hence Fisher's equation satisfies the conditions (1.5), (1.6). If we now further suppose that (1.7) and (1.8) are fulfilled, there holds according to Kolmogorov, Petrovsky and Piskunov [8] that there is a unique solution propagating at constant speed

$$v = 2\sqrt{f'(0)c} = 2\sqrt{\lambda c}. \tag{1.13}$$

Consequently, the solutions propagate at finite speed.

On the other hand, we realise that in the case $\lambda = 0$, Fisher's equation (1.2) coincides with the heat equation, admitting infinite speed of propagation [1]. However, note that in this case the assumption (1.5) is violated.

In conclusion, in our case with constant or piecewise constant values for the diffusion coefficient c, we anticipate solutions with finite speed of propagation. The succeeding numerical experiments meet this expectation. In all simulations performed below aiming at the exhibition of travelling wave solutions originating from compactly supported initial datum on bounded domains, we observe numerically a finite speed of propagation, see section 5.1.

In two and three spatial dimensions, propagating wave fronts may evolve to some stationary states. Albeit these need not be unique, they usually possess some kind of symmetry. Under the assumption that a stationary wave propagates along the x-axis, travelling wave solutions read in the two-dimensional case for some $\mathbf{x} = (x_1, x_2) \in \mathbb{R}^2$,

$$u(t, \mathbf{x}) = u(x_1 - Vt, x_2), \tag{1.14}$$

where V is the velocity of the wave. In the two-dimensional case with two stationary states, we may expect a reflection symmetry for the wave front with respect to its propagation direction [2].

1.2 Earlier works on Population Dynamics

In 2017, W. P. Petersen et al. published a paper [14] on the numerical simulation of Fisher's equation on geographical maps based on a finite difference scheme, employing a Godunov-type splitting. Following their main goal to model population dynamics, we discuss in the sequel another approach based on finite element methods, where we employ the Strang splitting method.

Our interests are drawn to study the evolution of our population density over space and time. Since we want to model the dispersal of the Homo Sapiens on the globe, our time horizon starts 200000 years ago, and ranges up to 15000 years ago. Moreover, our domain Ω represents the earth's surface. We model the humans' evolution with the Fisher's equation, a special case of the Kolmogorov-Petrovsky-Piscounov equation.

Chapter Two

Mathematical Derivation

2.1 Problem Formulation

Let $\Omega \subset \mathbb{R}^2$ be a bounded domain, let $T \in \mathbb{R}_{\geq 0}$ be the final time. We consider a continuous function $u : [0,T] \times \Omega \to \mathbb{R}$, representing the population density.

Assuming that $u(t) \in C^2(\overline{\Omega})$, we have

$$\frac{\partial u}{\partial t} = \operatorname{div}(\ c(t,\mathbf{x})\ \mathbf{grad}\ u) + \lambda\ (1 - \frac{u}{K(t,\mathbf{x})})\ u \qquad \text{in } [0,T] \times \Omega \tag{2.1}$$

where $c : [0,T]\ \times \Omega \to \mathbb{R}$ denotes the diffusion coefficient, $\lambda \in \mathbb{R}_{>0}$ is a constant growth factor, and $K : [0,T] \times \Omega \to \mathbb{R}$ describes the carrying capacity.

Note that our domain Ω need not be connected. Indeed accounting for the various islands over the globe, it is certain not to be. Thus it is of great importance to impose boundary conditions describing properly the dispersal over straits, whenever two land territories are close enough such that humans crossed the sea. Therefore we establish non-local boundary conditions.

Let us consider a point on the boundary of Ω, $\mathbf{x} \in \partial\Omega$. Let $L \in \mathbb{N}$ denote the number of connected components of our domain Ω. The boundary of Ω consists of all the curves enclosing a single connected component. Thus we may number these curves Γ_i for $i \in 1, ..., L$ such that $\bigcup_{i=1}^{L} \Gamma_i = \partial\Omega$, where $\Gamma_i \cap \Gamma_j = \emptyset$ for $i \neq j$. For any such $\mathbf{x}$, we then have that $\mathbf{x} \in \Gamma_i$ for some $i \in 1, ..., L$.

To account for the dispersal over the sea, we proceed as follows: let $\mathbf{j} : [0,T] \times \Omega \to \Omega$ model the flow of individuals in our domain. By Fourier's Law, we find that

$$\mathbf{j}(t,\mathbf{x}) = -c(t,\mathbf{x})\ \mathbf{grad}\ u \qquad \text{in } \ \Omega. \tag{2.2}$$

At the boundary, we impose the non-local flux condition

$$\mathbf{j}(t,\mathbf{x}) \cdot \mathbf{n}(\mathbf{x}) = \int_{\partial\Omega} g(\mathbf{x},\mathbf{y})\ (u(t,\mathbf{x}) - u(t,\mathbf{y}))\ \mathrm{dS}(\mathbf{y}), \tag{2.3}$$

where $\mathbf{n} \in \mathbb{R}^2$ denotes the outer normal vector, and where $g : \partial\Omega \times \partial\Omega \to \mathbb{R}$ is a kernel function decaying with increasing distance between $\mathbf{x}$ and $\mathbf{y}$. The intuition behind (2.3) is that with a larger distance between two coastlines the difficulty of crossing the sea increases, and thus the probability of the actual happening of such traversal diminishes. Moreover, the gain of population on the point of arrival at $\partial\Omega$ evidently implies the loss of population on Γ_i. Note that by taking all of the boundary $\partial\Omega$ into consideration as possible arrival sites, we include the possibility of travelling along the coastline of departure Γ_i. Combining equation (2.2) with (2.3), we obtain

$$- c(t,\mathbf{x})\ \mathbf{grad}\ u \cdot \mathbf{n}(\mathbf{x}) = \int_{\partial\Omega} g(\mathbf{x},\mathbf{y})\ (u(t,\mathbf{x}) - u(t,\mathbf{y}))\ \mathrm{dS}(\mathbf{y}). \tag{2.4}$$

Equation (2.4) connote the non-local boundary condition for our model.

Finally, starting with an initial population denoted by $u_0 : \Omega \to \mathbb{R}$, we obtain the following model:

$$\begin{cases} \dfrac{\partial u}{\partial t} = \mathrm{div}(\ c(t,\mathbf{x})\ \mathbf{grad}\ u) + \lambda\ (1 - \dfrac{u}{K(t,\mathbf{x})})\ u & \text{in } \Omega \\ -c(t,\mathbf{x})\ \mathbf{grad}\ u \cdot \mathbf{n}(\mathbf{x}) = \displaystyle\int_{\partial\Omega} g(\mathbf{x},\mathbf{y})\ (u(t,\mathbf{x}) - u(t,\mathbf{y}))\ \mathrm{dS}(\mathbf{y}) & \text{on } \partial\Omega \\ u(0,\mathbf{x}) = u_0(\mathbf{x}) & \text{in } \Omega \end{cases} \tag{2.5}$$

2.2 Variational Formulation

In the sequel we derive the weak formulation for (2.5). Let $v \in C_c^\infty(\overline{\Omega})$ be a testing function. Integrating (2.1) over Ω yields

$$\begin{aligned} \int_\Omega \frac{\partial u}{\partial t}(t,\mathbf{x})\ v(\mathbf{x})\ \mathrm{d}\mathbf{x} &= \int_\Omega \mathrm{div}(\ c(t,\mathbf{x})\ \mathbf{grad}\ u(t,\mathbf{x})\ v(\mathbf{x})\ \mathrm{d}\mathbf{x} \\ &\qquad + \int_\Omega \lambda\ (1 - \frac{u(t,\mathbf{x})}{K(t,\mathbf{x})})\ u(t,\mathbf{x})\ v(\mathbf{x})\ \mathrm{d}\mathbf{x} \\ &= -\int_\Omega c(t,\mathbf{x})\ \mathbf{grad}\ u(t,\mathbf{x}) \cdot \mathbf{grad}\ v(\mathbf{x})\ \mathrm{d}\mathbf{x} \\ &\quad + \int_{\partial\Omega} c(t,\mathbf{x})\ \mathbf{grad}\ u(t,\mathbf{x}) \cdot \mathbf{n}(\mathbf{x})\ v(\mathbf{x})\ \mathrm{dS}(\mathbf{x}) + \int_\Omega \lambda\ (1 - \frac{u(t,\mathbf{x})}{K(t,\mathbf{x})})\ u(t,\mathbf{x})\ v(\mathbf{x})\ \mathrm{d}\mathbf{x} \\ &= -\int_\Omega c(t,\mathbf{x})\ \mathbf{grad}\ u(t,\mathbf{x}) \cdot \mathbf{grad}\ v(\mathbf{x})\ \mathrm{d}\mathbf{x} \\ &\qquad - \int_{\partial\Omega}\int_{\partial\Omega} g(\mathbf{x},\mathbf{y})\ (u(t,\mathbf{x}) - u(t,\mathbf{y}))\ v(\mathbf{x})\ \mathrm{dS}(\mathbf{y})\mathrm{dS}(\mathbf{x}) \\ &\qquad + \int_\Omega \lambda\ (1 - \frac{u(t,\mathbf{x})}{K(t,\mathbf{x})})\ u(t,\mathbf{x})\ v(\mathbf{x})\ \mathrm{d}\mathbf{x} \end{aligned}$$

where we have used Green's first formula [7] and the non-local boundary conditions of equation (2.5). Choosing suitable function spaces leads us to the following variational formulation:

We seek $[0,T] \ni t \to u(t) \in H^1(\Omega)$ satisfying

$$\begin{aligned}\int_\Omega \frac{\partial u}{\partial t}(t,\mathbf{x})\ v(\mathbf{x})\ \mathrm{d}\mathbf{x} + \int_\Omega c(t,\mathbf{x})\ \mathbf{grad}\ u(t,\mathbf{x}) \cdot \mathbf{grad}\ v(\mathbf{x})\ \mathrm{d}\mathbf{x} \\ + \int_{\partial\Omega}\int_{\partial\Omega} g(\mathbf{x},\mathbf{y})\ (u(t,\mathbf{x}) - u(t,\mathbf{y}))\ v(\mathbf{x})\ \mathrm{dS}(\mathbf{y})\mathrm{dS}(\mathbf{x}) \\ - \int_\Omega \lambda\ (1 - \frac{u(t,\mathbf{x})}{K(t,\mathbf{x})})\ u(t,\mathbf{x})\ v(\mathbf{x})\ \mathrm{d}\mathbf{x} = 0, \quad \forall v \in H^1(\Omega)\end{aligned} \tag{2.6}$$

with $u(0,\mathbf{x}) = u_0(\mathbf{x})$ for $\mathbf{x} \in \Omega$.

Let $u, v \in H^1(\Omega)$. Let us introduce bilinear functionals $m, a, b : H^1(\Omega) \times H^1(\Omega) \to \mathbb{R}$ with

$$m(\frac{\partial u}{\partial t}(t), v) := \int_\Omega \frac{\partial u}{\partial t}(t,\mathbf{x})\ v(\mathbf{x})\ \mathrm{d}\mathbf{x} = \frac{\partial}{\partial t}\int_\Omega u(t,\mathbf{x})\ v(\mathbf{x})\ \mathrm{d}\mathbf{x} = \frac{\partial}{\partial t} m(u(t), v), \tag{2.7}$$

$$a(u(t), v) := \int_\Omega c(t,\mathbf{x})\ \mathbf{grad}\ u(t,\mathbf{x}) \cdot \mathbf{grad}\ v(\mathbf{x})\ \mathrm{d}\mathbf{x}, \tag{2.8}$$

$$b(u(t), v) := \int_{\partial\Omega}\int_{\partial\Omega} g(\mathbf{x},\mathbf{y})\ (u(t,\mathbf{x}) - u(t,\mathbf{y}))\ v(\mathbf{x})\ \mathrm{dS}(\mathbf{y})\mathrm{dS}(\mathbf{x}), \tag{2.9}$$

and finally for the non-linear term, we write $r : H^1(\Omega) \times H^1(\Omega) \to \mathbb{R}$, a mapping linear in its second argument, defined as

$$r(u(t); v) := \int_\Omega \lambda\ (1 - \frac{u(t,\mathbf{x})}{K(t,\mathbf{x})})\ u(t,\mathbf{x})\ v(\mathbf{x})\ \mathrm{d}\mathbf{x}. \tag{2.10}$$

With these notations, we arrive at the following formulation equivalent to equation (2.6): we seek $[0,T] \ni t \to u(t) \in H^1(\Omega)$ satisfying

$$\begin{cases} \dfrac{\partial}{\partial t} m(u(t), v) + a(u(t), v) + b(u(t), v) - r(u(t); v) = 0, \quad \forall v \in H^1(\Omega) \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad u(0) = u_0 \quad \in H^1(\Omega) \end{cases} \tag{2.11}$$

Chapter Three

Discretisation: Method of Lines

3.1 Galerkin Discretisation

We pursue a method of lines approach [7], that is we first proceed with a spatial semi-discretisation for equation (2.11).

The spatial Galerkin semi-discretisation relies on linear Lagrangian finite elements on a triangular mesh $\mathcal{M}$ of Ω with the lowest-order Lagrangian finite element space $\mathcal{S}_1^0(\mathcal{M})$ using a polygonal approximation of the boundary. In a first step, we replace our infinite-dimensional Hilbert space $H^1(\Omega) =: V_0$ with a finite dimensional subspace $V_{0,h} \subset V_0$. Thus, we seek $[0,T] \ni t \to u_h(t) \in V_{0,h}$ satisfying

$$\begin{cases} \dfrac{\partial}{\partial t} m(u_h(t), v) + a(u_h(t), v) + b(u_h(t), v) - r(u_h(t); v) = 0, \quad \forall v \in V_{0,h} \\ u_h(0) = u_{0,h} \quad \in V_{0,h} \end{cases} \tag{3.1}$$

where $u_{0,h} \in V_{0,h}$ represents the projection of $u_0 \in V_0$ onto $V_{0,h} = \mathcal{S}_1^0(\mathcal{M})$.

As a second step, we choose an ordered basis $\mathcal{B}_h = \{b_h^1, ..., b_h^N\}$, where $N := \dim(V_{0,h})$, such that $V_{0,h} = \text{span}\{\mathcal{B}_h\}$. Then we may expand $u_h, v_h \in V_{0,h}$ in terms of this basis functions as $u_h(t) = \sum_{i=1}^N \mu_i(t) b_h^i$ and $v_h = \sum_{j=1}^N \nu_j b_h^j$, where $\mu_i, \nu_j \in \mathbb{R}$ for $i, j \in \{1, ..., N\}$. Inserting the corresponding expansions into equation (3.1) gives

$$\begin{aligned} \frac{\partial}{\partial t} m(\sum_{i=1}^N \mu_i(t) b_h^i, \sum_{j=1}^N \nu_j b_h^j) + a(\sum_{i=1}^N \mu_i(t) b_h^i, \sum_{j=1}^N \nu_j b_h^j) + b(\sum_{i=1}^N \mu_i(t) b_h^i, \sum_{j=1}^N \nu_j b_h^j) \\ - r(\sum_{i=1}^N \mu_i(t) b_h^i; \sum_{j=1}^N \nu_j b_h^j) = 0, \qquad \forall \nu_j \in \mathbb{R},\ j \in \{1, ..., N\}. \end{aligned}$$

Equivalently,

$$\sum_{i=1}^N \sum_{j=1}^N \nu_j \ (\ \frac{\partial}{\partial t} m(\mu_i(t) b_h^i, b_h^j) + a(\mu_i(t) b_h^i, b_h^j) + b(\mu_i(t) b_h^i, b_h^j) - r(\sum_{l=1}^N \mu_l(t) b_h^l; b_h^j) \ ) = 0,$$
$$\forall \nu_j \in \mathbb{R}, \quad j \in \{1, ..., N\}.$$

Using Lemma 2.2.2.3 in [7], we find

$$\sum_{i=1}^{N}\Big(\frac{\partial \mu_i}{\partial t}(t) m(b_h^i, b_h^j) + \mu_i(t) a(b_h^i, b_h^j) + \mu_i(t) b(b_h^i, b_h^j) - r\Big(\sum_{l=1}^{N} \mu_l(t) b_h^l; b_h^j\Big)\Big) = 0, \quad j \in \{1, ..., N\}.$$

Thus we seek $[0, T] \ni t \to \mu_i(t) \in \mathbb{R}, i \in \{1, ..., N\}$ satisfying

$$\begin{cases} \displaystyle\sum_{i=1}^{N}\Big(\frac{\partial \mu_i}{\partial t}(t) m(b_h^i, b_h^j) + \mu_i(t) a(b_h^i, b_h^j) + \mu_i(t) b(b_h^i, b_h^j) - r\Big(\sum_{l=1}^{N} \mu_l(t) b_h^l; b_h^j\Big)\Big) = 0, \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad j \in \{1, ..., N\} \\ \displaystyle\sum_{i=1}^{N} \mu_i(0) b_h^i = u_{0,h} \qquad \in V_{0,h} \end{cases}$$

Let us now introduce suitable Galerkin matrices $\mathbf{M}$, $\mathbf{A}$, $\mathbf{B} \in \mathbb{R}^{N,N}$, where

$$\mathbf{M} := [m(b_h^i, b_h^j)]_{i,j=1}^{N} \tag{3.2}$$

$$\mathbf{A} := [a(b_h^i, b_h^j)]_{i,j=1}^{N} \tag{3.3}$$

$$\mathbf{B} := [b(b_h^i, b_h^j)]_{i,j=1}^{N} \tag{3.4}$$

These matrices will be assembled from the entries of the local element matrices on the reference triangle, see section 4.2.2 below. Let $\boldsymbol{\mu} := (\mu_1, \mu_2, ..., \mu_N)^T \in \mathbb{R}^N$. For the non-linear term write $\boldsymbol{\rho} : [0, T] \times \mathbb{R}^N \to \mathbb{R}^N$ with

$$\boldsymbol{\rho}(t, \boldsymbol{\mu}(t)) = [r(\sum_{l=1}^{N} \mu_l(t) b_h^l; b_h^j)]_{j=1}^{N}. \tag{3.5}$$

We then seek $[0, T] \ni t \to \boldsymbol{\mu}(t) \in \mathbb{R}^N$ satisfying

$$\begin{cases} \mathbf{M}\{\dfrac{\partial \boldsymbol{\mu}}{\partial t}(t)\} + \mathbf{A}\boldsymbol{\mu}(t) + \mathbf{B}\boldsymbol{\mu}(t) - \boldsymbol{\rho}(t, \boldsymbol{\mu}(t)) = 0 \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \boldsymbol{\mu}(0) = \boldsymbol{\mu}_0 \end{cases} \tag{3.6}$$

where $\boldsymbol{\mu}_0$ is the coefficient vector for the projection of u_0 onto $V_{0,h}$. We can recover the discrete solution with $u_h = \sum_{i=1}^{N} \mu_i(t) b_h^i$.

Let me remark that by means of suitable numerical quadrature it is often possible to obtain diagonal mass matrices, a trick called mass lumping [7]. In our case, we will make use of mass lumping (see Remark 6.3.4.18 in [7]) based on the composite trapezoidal rule, (2.3.3.8) in [7]. This allows the efficient computation of the inverse of the mass matrix $\mathbf{M}$ in (3.6).

We have thus obtained a semi-discrete evolution problem for the coefficient vector $\boldsymbol{\mu} \in \mathbb{R}^N$. We acquire the full discretisation for equation (3.6) by employing a suitable time stepping method.

3.2 Strang Splitting Method

We apply the Strang splitting method [6] to solve the ordinary differential equation (3.6).

To this end, first note that since the bilinear form m in (2.7) is symmetric and positive definite, we have that $\mathbf{M}$ is invertible [7]. Hence we may rewrite (3.6) as

$$\begin{cases} \dfrac{\partial \boldsymbol{\mu}}{\partial t}(t) = f(t, \boldsymbol{\mu}) + g(t, \boldsymbol{\mu}) \\ \qquad\qquad\qquad \boldsymbol{\mu}(0) = \boldsymbol{\mu}_0 \end{cases} \tag{3.7}$$

where $f(t, \boldsymbol{\mu}) := -\mathbf{M}^{-1}(\mathbf{A} + \mathbf{B})\ \boldsymbol{\mu}(t)$, and $g(t, \boldsymbol{\mu}) := \mathbf{M}^{-1}\boldsymbol{\rho}(t, \boldsymbol{\mu}(t))$. Splitting equation (3.7) into two parts, $\frac{\partial \boldsymbol{\mu}}{\partial t}(t) = f(t, \boldsymbol{\mu})$ and $\frac{\partial \boldsymbol{\mu}}{\partial t}(t) = g(t, \boldsymbol{\mu})$, allows to solve each part separately.

Consider the following mappings for fixed $t \in [0, T]$:

$$\boldsymbol{\Phi}_f^t : \begin{cases} \mathbb{R}^N \to \mathbb{R}^N \\ \boldsymbol{\mu}_0 \to \boldsymbol{\mu}(t) \end{cases} \tag{3.8}$$

and

$$\boldsymbol{\Phi}_g^t : \begin{cases} \mathbb{R}^N \to \mathbb{R}^N \\ \boldsymbol{\mu}_0 \to \boldsymbol{\mu}(t) \end{cases} \tag{3.9}$$

corresponding to $\frac{\partial \boldsymbol{\mu}}{\partial t}(t) = f(t, \boldsymbol{\mu})$ and $\frac{\partial \boldsymbol{\mu}}{\partial t}(t) = g(t, \boldsymbol{\mu})$, respectively, with appropriate initial condition $\boldsymbol{\mu}(0) = \boldsymbol{\mu}_0$. These are well-defined mappings of the state space into itself, see [6]. We can now define evolution operators for each part:

$$\boldsymbol{\Phi}_f : \begin{cases} [0, T] \times \mathbb{R}^N \to \mathbb{R}^N \\ (t, \boldsymbol{\mu}_0) \to \boldsymbol{\Phi}_f^t(\boldsymbol{\mu}_0) := \boldsymbol{\mu}(t) \end{cases} \tag{3.10}$$

$$\boldsymbol{\Phi}_g : \begin{cases} [0, T] \times \mathbb{R}^N \to \mathbb{R}^N \\ (t, \boldsymbol{\mu}_0) \to \boldsymbol{\Phi}_g^t(\boldsymbol{\mu}_0) := \boldsymbol{\mu}(t) \end{cases} \tag{3.11}$$

Then we find that $\frac{\partial \boldsymbol{\Phi}_f}{\partial t}(t, \boldsymbol{\mu}) = f(t, \boldsymbol{\Phi}_f^t(\boldsymbol{\mu}))$, as well as $\frac{\partial \boldsymbol{\Phi}_g}{\partial t}(t, \boldsymbol{\mu}) = g(t, \boldsymbol{\Phi}_g^t(\boldsymbol{\mu}))$.

We proceed by introducing an equidistant temporal mesh with nodes $0 = t_0 < t_1 < ... < t_{M-1} < t_M = T,\ M \in \mathbb{N}$. Let $\tau := t_1 - t_0$ be the step size. This allows to discretise the mappings (3.8), (3.9), as well as the evolution operators (3.10) and (3.11). In particular we define the discrete mapping

$$\boldsymbol{\Psi}^\tau := \boldsymbol{\Phi}_f^{\tau/2} \circ \boldsymbol{\Phi}_g^\tau \circ \boldsymbol{\Phi}_f^{\tau/2}. \tag{3.12}$$

This constitutes the Strang splitting method.

3.2.1 Diffusion Term

Let us first consider the pure diffusion problem, represented by $\frac{\partial \boldsymbol{\mu}}{\partial t}(t) = f(t, \boldsymbol{\mu})$ with initial condition $\boldsymbol{\mu}(0) = \boldsymbol{\mu}_0$. We compute a sequence $(\boldsymbol{\mu}^{(k)})_{k=0}^{M}$ of approximations $\boldsymbol{\mu}^{(k)} \approx \boldsymbol{\mu}(t_k)$ for $k \in \{1, ..., M\}$ on the nodes of the temporal mesh, according to

$$\begin{aligned} \boldsymbol{\mu}^{(0)} &:= \boldsymbol{\mu}_0 \\ \boldsymbol{\mu}^{(k)} &:= \boldsymbol{\Psi}_f^{t_k}(\boldsymbol{\mu}^{(k-1)}), \ k \in \{1, ..., M\} \end{aligned} \tag{3.13}$$

where $\boldsymbol{\Psi}_f^{t_k}$ for $k \in 1, ..., M$ is the discretised mapping (3.8).

We require an L(π)-stable implicit 2-stage Runge-Kutta method for the diffusion term. We choose the SDIRK-2 method, which is described by the following Butcher Tableau:

$$\begin{array}{c|c} \mathbf{c} & \mathfrak{A} \\ \hline & \mathbf{b}^T \end{array} \quad \hat{=} \quad \begin{array}{c|cc} \xi & \xi & 0 \\ 1 & 1-\xi & \xi \\ \hline & 1-\xi & \xi \end{array}$$

where $\xi := 1 - \frac{1}{2}\sqrt{2}$. Indeed, the SDIRK-2 method is L(π)-stable according to Definition 6.2.7.48 in [7]. I reprint this definition below.

Definition 1: L(π)-stability A Runge-Kutta single-step method satisfying $\lim_{j\to\infty}(S(z))^j y_0 = 0, \ \forall y_0 \in \mathbb{R}, \forall z \in \mathbb{R}_{<0}$ is called L(π)-stable, if its stability function $S(z)$ satisfies

$$\begin{aligned} &(i) \ \ |S(z)| < 1, \quad \forall z < 0, \text{ and} \\ &(ii) \ \lim_{z\to-\infty} S(z) = 0. \end{aligned}$$

Claim 1: *The SDIRK-2 method is L(π)-stable.*

Proof. To verify Claim 1, we inspect the stability function $S(z)$ for the SDIRK-2 method. The stability function is derived by applying the method to the scalar, linear test equation

$$\dot{y}(t) = \lambda y(t), \quad \lambda \in \mathbb{R}.$$

The solution can be written as

$$y_{k+1} = S(z) y_k$$

where $S : \mathbb{C} \to \mathbb{R}$ is the stability function for the method, and $z := \lambda\tau$ with step size τ. In our case, we find for the increments $k_1, k_2 \in \mathbb{R}$

$$\begin{aligned} k_1 &= \lambda(y_k + \tau\xi k_1), \\ k_2 &= \lambda(y_k + \tau(1-\xi)k_1 + \xi k_2). \end{aligned}$$

Thus, we obtain

$$k_1 = \frac{\lambda}{1-\lambda\tau\xi} y_k,$$
$$k_2 = \frac{\lambda(1+\tau(1-\xi)\lambda)}{(1-\lambda\tau\xi)^2} y_k.$$

Furthermore, we find for the solution

$$\begin{aligned} y_{k+1} &= y_k + \tau(1-\xi)k_1 + \tau\xi k_2 \\ &= y_k + \tau(1-\xi)\frac{\lambda}{1-\lambda\tau\xi} y_k + \tau\xi\frac{\lambda(1+\tau(1-\xi)\lambda)}{(1-\lambda\tau\xi)^2} y_k \\ &= (1 + \frac{z(1-\xi)}{1-\xi z} + \frac{\xi z + \xi z^2(1-\xi)}{(1-\xi z)^2}) y_k \\ &= \underbrace{\frac{1+(1-2\xi)z}{(1-\xi z)^2}}_{=: S(z)} y_k. \end{aligned}$$

That is, we find

$$S(z) = \frac{1+(1-2\xi)z}{1-2\xi z+\xi^2 z^2}, \quad z \in \mathbb{C}^- := \{z \ \in \mathbb{C} | \Re(z) < 0\}. \tag{3.14}$$

Clearly, we see that (ii) in Def. 1 is satisfied. Moreover, to verify (i), we consider S on the imaginary axis.

$$|S(iy)|^2 = \frac{|1+(1-2\xi)iy|^2}{|(1-\xi iy)|^4} = \frac{1-(1-2\xi)^2y^2}{(1+(\xi y)^2)^2} = \frac{1-2\xi^2y^2}{1+2\xi^2y^2+\xi^4y^4} \leq 1, \quad y \in \mathbb{R},$$

where we have used for the last equality the definition of ξ. Since the only pole ($z = \frac{1}{\xi}$) of the stability function lies in the positive half plane of $\mathbb{C}$, the function S is holomorphic in the negative half plane. Furthermore, S is bounded by 1 on the imaginary axis, which coincides with the boundary of the negative half plane. By the maximum principle for holomorphic functions [11], S is bounded on the entire negative half plane by 1. Thus (i) in Def. 1 is satisfied, and therewith, the SDIRK-2 method is L(π)-stable. □

We now consider the SDIRK-2 method for the diffusive term. Following the Butcher Tableau, we compute $\boldsymbol{\mu}^{(k)}$ from $\boldsymbol{\mu}^{(k-1)}$ for $k \in \{1, ..., M\}$ by solving a linear system of equations.

We seek $\mathbf{k}_1, \mathbf{k}_2 \in \mathbb{R}^N$ such that

$$\begin{aligned} \mathbf{M}\mathbf{k}_1 + \tau\xi(\mathbf{A}+\mathbf{B})\mathbf{k}_1 &= -(\mathbf{A}+\mathbf{B})\boldsymbol{\mu}^{(k-1)} \\ \mathbf{M}\mathbf{k}_2 + \tau(1-\xi)(\mathbf{A}+\mathbf{B})\mathbf{k}_1 + \tau\xi(\mathbf{A}+\mathbf{B})\mathbf{k}_2 &= -(\mathbf{A}+\mathbf{B})\boldsymbol{\mu}^{(k-1)} \end{aligned} \tag{3.15}$$

Then we can recover the solution with

$$\boldsymbol{\mu}^{(k)} = \boldsymbol{\Psi}_f^\tau(\boldsymbol{\mu}^{(k-1)}) = \boldsymbol{\mu}^{(k-1)} + \tau(1-\xi)\mathbf{k}_1 + \tau\xi\mathbf{k}_2. \tag{3.16}$$

3.2.2 Reaction Term

For the reaction term, being represented by $\frac{\partial \boldsymbol{\mu}}{\partial t}(t) = g(t, \boldsymbol{\mu})$ with initial condition $\boldsymbol{\mu}(0) = \boldsymbol{\mu}_0$, we can implement the exact evolution operator, for the analytical solution of this ordinary differential equation is known. Thus the discrete evolution operator coincides with the exact evolution.

To derive the analytical solution of the ordinary differential equation arising from the reaction term, we solve the *purely local* logistics ordinary differential equation

$$\begin{cases} \dfrac{\partial u}{\partial t} = \lambda \left(1 - \dfrac{u}{K(t, \mathbf{x})}\right) u & \text{in } [T_0, T_1] \times \Omega, \\ u(T_0, \mathbf{x}) = u_0(\mathbf{x}) & \text{in } \Omega, \end{cases} \tag{3.17}$$

where $\mathbb{R}_{\geq 0} \ni T_0 < T_1 < \infty$. Note that λ is constant. Moreover, we may assume that the carrying capacity is constant over the time interval $[T_0, T_1]$, since in the end, we let our solution evolve over several time intervals, in each of which the carrying capacity is constant in *time*.

We first divide (3.17) by $(1 - \frac{u}{K})\, u$, where in the sequel we simply write K for $K(t, \mathbf{x}) = K(\mathbf{x}),\ \forall (t, \mathbf{x})$ in $[T_0, T_1] \times \Omega$. Then using differentials for the partial derivatives we obtain

$$\frac{1}{(1 - \frac{u}{K})\, u} \frac{\mathrm{d}u}{\mathrm{d}t} = \lambda. \tag{3.18}$$

Equivalently, we can write

$$\frac{1 - \frac{u}{K} + \frac{u}{K}}{(1 - \frac{u}{K})\, u} \frac{\mathrm{d}u}{\mathrm{d}t} = (\frac{1}{u} + \frac{1}{K - u}) \frac{\mathrm{d}u}{\mathrm{d}t} = \lambda. \tag{3.19}$$

Hence we obtain

$$(\frac{1}{u} + \frac{1}{K - u}) \mathrm{d}u = \lambda \mathrm{d}t, \tag{3.20}$$

which yields by integration

$$\ln(u) - \ln(K - u) = \lambda t + C_0, \tag{3.21}$$

with integration constant $C_0 \in \mathbb{R}$. Thus we find

$$\frac{u}{K - u} = C e^{\lambda t}, \tag{3.22}$$

with $C = e^{C_0}$. Finally, by properly rearranging we have

$$u = \frac{C K e^{\lambda t}}{1 + C e^{\lambda t}}. \tag{3.23}$$

We determine the constant C with the initial condition,

$$u = \frac{u_0 K e^{\lambda t}}{K + u_0(e^{\lambda t} - e^{\lambda T_0})} = \frac{K}{1 + (K u_0^{-1} - e^{\lambda T_0})e^{-\lambda t}}. \tag{3.24}$$

Using (3.24) for fixed $t \in [0, T]$, the mapping for the evolution operator reads

$$\text{For every component of } \boldsymbol{\mu}: \quad \boldsymbol{\Phi}_g^t(\boldsymbol{\mu}) = \frac{K}{1 + (K \boldsymbol{\mu}_0^{-1} - 1)e^{-\lambda t}}, \tag{3.25}$$

where $K = K(t, \mathbf{x})$ is the space dependent carrying capacity, constant over some finite time horizon. Let me remark that we may partition the total time interval $[0, T]$ into several time slots $[T_{j-1}, T_j]$, $j \in \{1, ..., J\}$, $J \in \mathbb{N}$ such that $T_{j-1} < T_j$, $j \in \{1, ..., J\}$, $T_0 = 0$ and $T_J = T$. Then K shall be constant in time in each time slot $[T_{j-1}, T_j]$ for $j \in \{1, ..., J\}$. Note that the locality of (3.17) in the finite element setting is preserved due to the employed mass lumping, see section 3.1 above.

Chapter Four

Implementation

In this chapter, I outline details of the implementation. In particular, note that I have made use of the finite element library LehrFEM++ [9]. I oriented the choice of parameters on a general notion of the settlements of the Homo Sapiens, ranging from 200000 years ago up to 15000 years ago, see [17]. Please note that information from pre-historic times is sparse, impeding highly accurate parameters.

4.1 Mesh Generation

First of all, I have generated a triangulation for our domain Ω, which, in our case of population dispersal, represents the earth's surface. The succeeding data flow diagram, figure 4.1, summarises the procedure of generating a mesh file for our domain Ω.

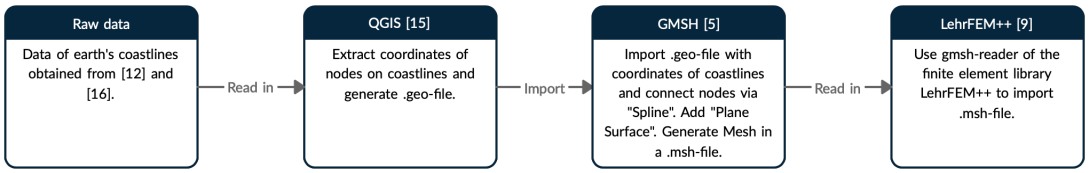


Figure 4.1 Data flow diagram depicting the main steps to generate a triangulation of the earth's surface.

4.1.1 Raw data

In a first step, I obtained data for the earth's coastlines from a geographical database, maintained by the University of Hawai'i together with a governmental geosciences lab, the national oceanic and atmospheric administration [16]. The database includes a high-resolution data set of the global shorelines. Moreover, I made use of a data base of land and sea-floor elevations [12]. This was needed to extract the contour lines of the coastlines.

4.1.2 QGIS

Subsequently, I imported this data into the geographic information system QGIS [15] via `Layer` → `Layer hinzufügen` → `Vektorlayer hinzufügen`. I selected the path to the data set with full resolution (*GSHHS_f_L1.shp*) from [16]. Furthermore, I added the data base of land and sea-floor elevations (*ETOPO1_Ice_g_gmt4.grd*) from [12] via `Layer` → `Layer hinzufügen` → `Rasterlayer hinzufügen`. I then extracted the contour of the coastlines, by selecting `Raster` → `Extraktion` → `Kontur`. Then there is the need of specifying the correct input layer (*ETOPO1_Ice_g_gmt4.grd*) and there is the option to save the output into a file. Also, I had to specify an additional command line argument, `-fl "-200"`. This yields the extracted coastlines. Note that I moved the coastlines of the continents Europe, Africa and Australia to the left of America via `Bearbeiten` → `Objekt(e) verschieben`. Finally, I selected all coastlines and merged them via `Bearbeiten` → `Gewählte Objekte verschmelzen`. Therewith, I have now extracted the coordinates from the nodes of the shorelines.

Note that QGIS allows to install a plug-in to generate *.geo*-files from given input data. This is the main reason why I have been working with QGIS. With this plug-in I was able to select `Erweiterungen` → `Gmsh` → `Generate a Gmsh geometry file`. I now selected the output file of the extracted shoreline, and then via `Generate geometry file`, I obtained a *.geo*-file with the coordinates of the nodes on all coastlines.

4.1.3 Gmsh

Consequently, I imported the resulting *.geo*-file into Gmsh [5]. It lists all point coordinates from nodes on the coastline. I connect these nodes by directly modifying the *.geo*-file: I add `Spline(IL+i) = {...}` where I list in curly brackets all the nodes to be connected. Note that `IL` stands for the splines, whereas `i` simply denotes the numbering of all splines. Then I add a line loop `Line Loop(ILL+i) = {IL+i}`, where `ILL` represents the line loops. Finally, I add the surfaces, `Plane Surface(IS+i) = {ILL+i}` with `IS` standing for the plane surfaces. Note that large seas inside a surface can be explicitly denoted as "holes" in the surface, by listing the corresponding line loop number denoting the surface of the "hole" behind the number of the line loop representing the surrounding surface. Furthermore, it is crucial to define the mesh size by adding these two succeeding lines on top of the *.geo*-file,

```
Mesh.CharacteristicLengthMin = 2;
Mesh.CharacteristicLengthMax = 2;
```

for otherwise, the resulting triangulation may be too coarse to properly respect the boundaries.

Now I can open this modified *.geo*-file in Gmsh and select `Mesh` → `2D`. This generates a two-dimensional mesh, stored in a *.msh*-file. Note that the number of point coordinates on the shoreline amounts to 340125. Therein are included all islands of reasonable size.

4.2 LehrFEM++

As a next step, I realised the population dispersal over the triangulated surface based on the mathematical model. The implementation heavily relies on the finite element library LehrFEM++ [9].

4.2.1 Input-Output Reader

In this library, there is a built-in reader to gather the information for the spatial mesh. This reader extracts all necessary information from the *.msh*-file. The number of degrees of freedoms in the final implementation amounts to 15463. Clearly, this implies that not every node on the boundary is considered in the final mesh. This could be remedied by refining the mesh size.

4.2.2 Galerkin Matrices

In the sequel, I discuss the detailed implementation of the Galerkin matrices defined in equations (3.2), (3.3), (3.4).

For the matrix $\mathbf{M} \in \mathbb{R}^{N,N}$ defined in (3.2), we have

$$\mathbf{M} := [m(b_h^i, b_h^j)]_{i,j=1}^N = [\int_\Omega b_h^i(t,\mathbf{x})\ b_h^j(\mathbf{x})\ \mathrm{d}\mathbf{x}\ ]_{i,j=1}^N. \tag{4.1}$$

This is the standard mass matrix (Remark 2.2.2.5 in [7]), and thus, I used the LehrFEM++ function **ReactionDiffusionElementMatrixProvider** for its assembly.

Furthermore, for the matrix $\mathbf{A} \in \mathbb{R}^{N,N}$ defined in (3.3), we find

$$\mathbf{A} := [a(b_h^i, b_h^j)]_{i,j=1}^N = [\int_\Omega c(t,\mathbf{x})\ \mathbf{grad}\ b_h^i(t,\mathbf{x}) \cdot \mathbf{grad}\ b_h^j(\mathbf{x})\ \mathrm{d}\mathbf{x}\ ]_{i,j=1}^N, \tag{4.2}$$

with diffusion coefficient $c : [0,T] \times \Omega \to \mathbb{R}$. Again, we are left with a standard Galerkin matrix, the stiffness matrix (Remark 2.2.2.5 in [7]). Thus I again made use of **ReactionDiffusionElementMatrixProvider** for assembling $\mathbf{A}$.

Finally, for $\mathbf{B} \in \mathbb{R}^{N,N}$ the discussion will be more elaborate. By definition, we obtain

$$\mathbf{B} := [b(b_h^i, b_h^j)]_{i,j=1}^N = [\int_{\partial\Omega}\int_{\partial\Omega} g(\mathbf{x},\mathbf{y})\ (b_h^i(t,\mathbf{x}) - b_h^i(t,\mathbf{y}))\ b_h^j(\mathbf{x})\ \ \mathrm{dS}(\mathbf{y})\mathrm{dS}(\mathbf{x})\ ]_{i,j=1}^N. \tag{4.3}$$

We proceed by considering the two terms separately. For $i, j \in \{1, ..., N\}$ we have

$$\begin{aligned}
\mathbf{B}_{i,j} &= \int_{\partial\Omega}\int_{\partial\Omega} g(\mathbf{x},\mathbf{y})\ b_h^i(t,\mathbf{x})\ b_h^j(\mathbf{x})\ \ \mathrm{dS}(\mathbf{y})\mathrm{dS}(\mathbf{x}) \\
&\quad - \int_{\partial\Omega}\int_{\partial\Omega} g(\mathbf{x},\mathbf{y})\ b_h^i(t,\mathbf{y})\ b_h^j(\mathbf{x})\ \ \mathrm{dS}(\mathbf{y})\mathrm{dS}(\mathbf{x}) \\
&= \int_{\partial\Omega}\int_{\partial\Omega} g(\mathbf{x},\mathbf{y})\ \mathrm{dS}(\mathbf{y})\ b_h^i(t,\mathbf{x})\ b_h^j(\mathbf{x})\ \mathrm{dS}(\mathbf{x}) \\
&\quad - \int_{\partial\Omega}\int_{\partial\Omega} g(\mathbf{x},\mathbf{y})\ b_h^i(t,\mathbf{y})\ \mathrm{dS}(\mathbf{y})\ b_h^j(\mathbf{x})\ \mathrm{dS}(\mathbf{x})
\end{aligned}$$

For the first integral, let us introduce the function $h_1 : \partial\Omega \to \mathbb{R}$, with

$$h_1(\mathbf{x}) = \int_{\partial\Omega} g(\mathbf{x},\mathbf{y})\ \mathrm{dS}(\mathbf{y}). \tag{4.4}$$

For the second term, we introduce $h_2^i : [0,T] \times \partial\Omega \to \mathbb{R},\ i \in \{1, ..., N\}$, satisfying

$$h_2^i(t,\mathbf{x}) = \int_{\partial\Omega} g(\mathbf{x},\mathbf{y})\ b_h^i(t,\mathbf{y})\ \mathrm{dS}(\mathbf{y}), \quad i \in \{1, ..., N\}. \tag{4.5}$$

Then we obtain for $i, j \in \{1, ..., N\}$,

$$\begin{aligned}
\mathbf{B}_{i,j} &= \int_{\partial\Omega} \underbrace{\int_{\partial\Omega} g(\mathbf{x},\mathbf{y})\ \mathrm{dS}(\mathbf{y})}_{=:\ h_1(\mathbf{x})}\ b_h^i(t,\mathbf{x})\ b_h^j(\mathbf{x})\ \mathrm{dS}(\mathbf{x}) \\
&\quad - \int_{\partial\Omega} \underbrace{\int_{\partial\Omega} g(\mathbf{x},\mathbf{y})\ b_h^i(t,\mathbf{y})\ \mathrm{dS}(\mathbf{y})}_{=:\ h_2^i(t,\mathbf{x})}\ b_h^j(\mathbf{x})\ \mathrm{dS}(\mathbf{x}) \\
&= \int_{\partial\Omega} h_1(\mathbf{x})\ b_h^i(t,\mathbf{x})\ b_h^j(\mathbf{x})\ \mathrm{dS}(\mathbf{x}) - \int_{\partial\Omega} h_2^i(t,\mathbf{x})\ b_h^j(\mathbf{x})\ \mathrm{dS}(\mathbf{x})
\end{aligned}$$

Now we observe that the first term corresponds to a standard mass matrix on the boundary, i.e. a mass edge matrix (Example 2.7.4.37 in [7]), with function handle h_1. Thus the implementation of the **MassEdgeMatrixProvider** in LehrFEM++ suits the required form. The implementation of h_1 requires the quadrature over $\partial\Omega$. We use the **EdgeMidpointRule** of third order for the concerned triangles.

The second integral demands a slightly more intricate treatment. We observe that it is of the form of a load vector (Remark 2.4.6.7 in [7]) on the boundary with function handle $h_2^i,\ i \in \{1, ..., N\}$. Hence we may use LehrFEM++'s **ScalarLoadEdgeVectorProvider**. What concerns the function handle $h_2^i,\ i \in \{1, ..., N\}$, we identify again the form of a load vector, where, for fixed $\mathbf{x} \in \partial\Omega$, we have the coefficient g depending on $\mathbf{y} \in \partial\Omega$. This implies the use of **ScalarLoadEdgeVectorProvider** one more time.

The system of equations (3.15) was solved with the sparse matrix solver from the header-only library Eigen [3].

4.2.3 Visualisation

Finally, I used the VTK-writer class from LehrFEM++, to write the results into a VTK-file, which in turn I imported into ParaView [13] in order to visualise the results.

Chapter Five

Model Problem

In this chapter, I discuss the simulation of equation (2.5) on simplified domains, with the aim to exhibit nice travelling wave solution as well as analysing the convergence of our numerical scheme. In the first section, we focus on travelling wave fronts.

5.1 Travelling Wave Solutions

Let me refer to section 1.1 for a discussion on travelling wave solutions. The exhibition of travelling wave fronts is taking place on three model domains.

Our first model domain is a ball of radius 0.5 around the point $\mathbf{y} = \binom{0.5}{0.5}$,

$$\tilde{\Omega}_1 := B_{0.5}(\binom{0.5}{0.5}) = \{\mathbf{x} \in \mathbb{R}^2 | \|\mathbf{x} - \binom{0.5}{0.5}\| < 0.5\}. \tag{5.1}$$

The mesh $\mathcal{M}_1$ for $\tilde{\Omega}_1$ is depicted in figure 5.1. We set the diffusion coefficient c and the

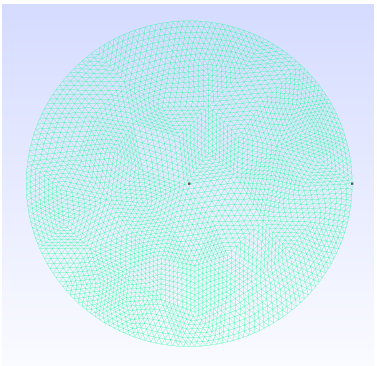
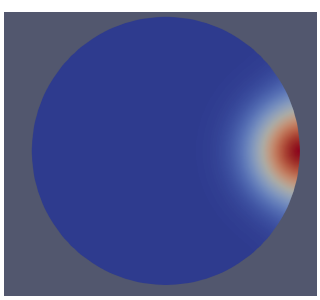


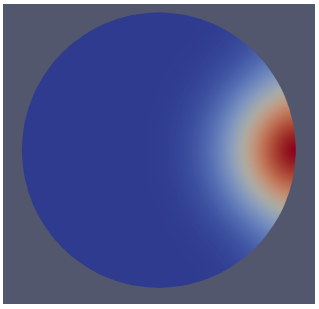
Figure 5.1 Triangular mesh $\tilde{\mathcal{M}}_1$ of $\tilde{\Omega}_1$ generated with Gmsh [5].

growth factor λ to be constant. We choose a constant carrying capacity K. Furthermore, we consider homogeneous Neumann boundary conditions.

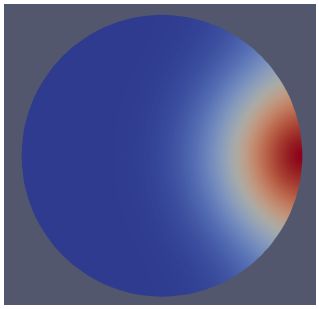
In a first step, the initial condition is represented by one point load with value 0.001 located at $\binom{1}{0.5}$. This setting yields clearly visible travelling wave fronts, originating from the initial condition, see figure 5.2.



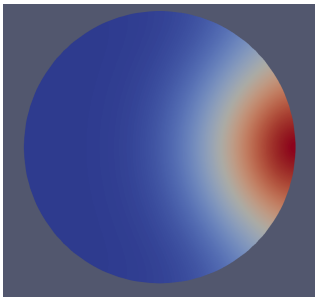
(a) Solution after $M = 100$ time steps.



(b) Solution after $M = 200$ time steps.



(c) Solution after $M = 300$ time steps.



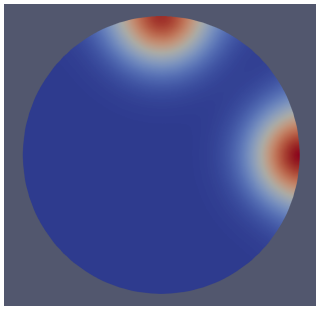
(d) Solution after $M = 400$ time steps.

Figure 5.2 Solution vector with initial condition represented by one point load visualised with ParaView [13] on mesh with $N = 3581$ degrees of freedom.

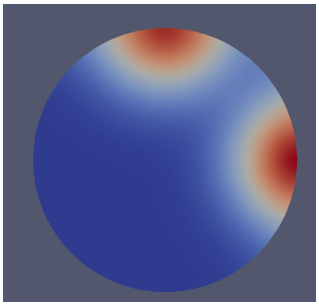
In a second step, we leave all parameters unchanged, except for the initial condition, which is now represented by an additional point load, the first being located at $\binom{1}{0.5}$, the second located at $\binom{0.5}{1}$, both of value 0.0005. In this case we obtain again nice travelling wave fronts, originating from both sources, and superimposing one another as time evolves, see figure 5.3. Note that the wave originates from a compactly supported initial datum.

In both cases, we observe a good approximation of a wave front evolving with a finite speed of propagation. Admittedly, we must behold that in the frame of the Strang splitting, the first half time step coincides with the solution of the heat equation. Since the heat equation admits infinite speed of propagation, we expect that after this first half time step, the solution immediately attains a value arbitrarily close to zero, but non-zero, on the whole domain. This can be noticed in figure 5.10, where we can read off the values of the solution vector on the domain. There is no area where the solution is zero; instead the areas not yet populated carry values very close to zero, namely $7.1 \cdot 10^{-6}$. The finite propagation speed arises due to the non-linearity. Indeed, we observe that the wave front propagates for further time steps with finite speed. Accordingly, the numerical results *approximate* the finite speed of propagation.

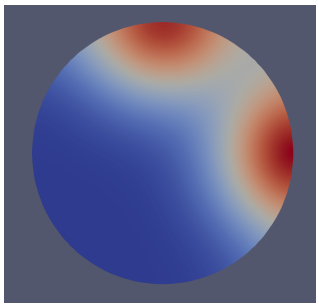
Secondly, we consider three circles, at a positive distance from each other. Concretely,



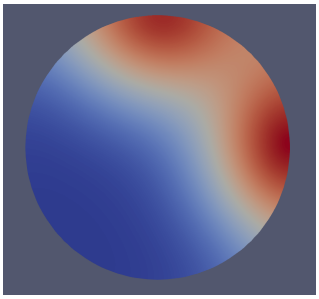
(a) Solution after $M = 100$ time steps.



(b) Solution after $M = 200$ time steps.



(c) Solution after $M = 300$ time steps.



(d) Solution after $M = 400$ time steps.

Figure 5.3 Solution vector with initial condition represented by two point loads visualised with ParaView [13] on mesh with $N = 3581$ degrees of freedom.

we set

$$\tilde{\Omega}_2 := B_{0.25}(\binom{0}{0}) \cup B_{0.5}(\binom{2}{0}) \cup B_{0.5}(\binom{0}{2}). \tag{5.2}$$

On this domain we aim to study the influence of the non-local boundary conditions (2.4) on the travelling wave solution. The mesh $\tilde{\mathcal{M}}_2$ is illustrated in figure 5.4. We set the parameters as follows: we choose a constant diffusion coefficient c and growth factor λ. We set our carrying capacity K constant as well. We start with an initial population on one circle, represented by one point load of 0.00008, such that the wave front originates from this source.

In a first step, we impose homogenous Neumann boundary conditions. This yields wave fronts evolving in a single circle only, see figure 5.5. Again, the wave fronts propagate with finite speed.

In a second step, we ascertain non-local boundary conditions (2.4). The kernel is chosen as

$$g_1(\mathbf{x}, \mathbf{y}) := \frac{1}{1 + \|\mathbf{x} - \mathbf{y}\|^2}. \tag{5.3}$$

Notice that (5.3) is a slowly decaying function kernel. Note also that in this case, the kernel attributes a non-trivial value on all of the boundary $\partial\Omega_2$. With this choice, we observe that all circles are hit by the wave fronts, see figure 5.6.

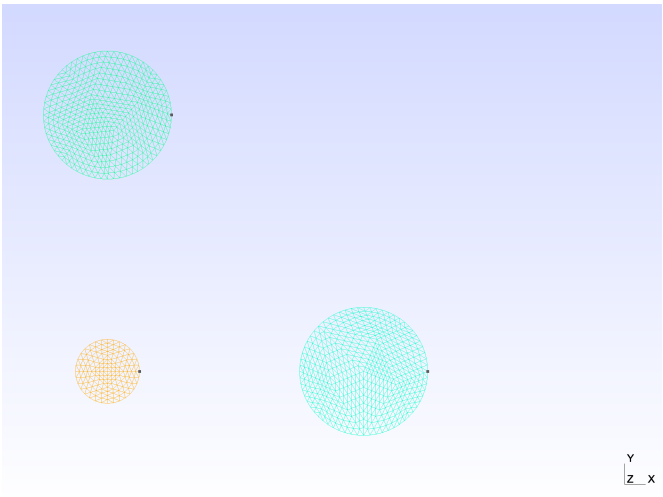


Figure 5.4 Triangular mesh $\tilde{\mathcal{M}}_2$ of $\tilde{\Omega}_2$ generated with Gmsh [5].

In a third step, we aim at further restricting the kernel. We set it to

$$g_2(\mathbf{x},\mathbf{y}) := \begin{cases} \frac{1}{1+\|\mathbf{x}-\mathbf{y}\|^2}, & \text{if } \|x-y\| \leq 1.75, \\ 0, & \text{elsewhise.} \end{cases} \tag{5.4}$$

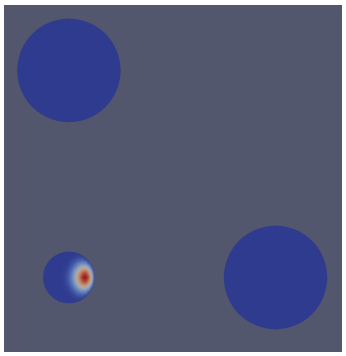
Let me recall the domain $\tilde{\Omega}_2$. Starting from the small ball $B_{0.25}(\binom{0}{0})$, we realise that the upper bound $\|x-y\| \leq 1.75$ in (5.4) does not reach all of $\partial\tilde{\Omega}_2$. However, we also note that an upper bound does not restrict the wave fronts from evolving along the boundaries, and thus, we still expect all of the boundary to be affected by the non-local boundary conditions. Indeed, the results in figure 5.7 qualitatively coincide with those in figure 5.6.

Now, in a final step, we may try to confine the influence of the kernel by adding a lower bound. Consider

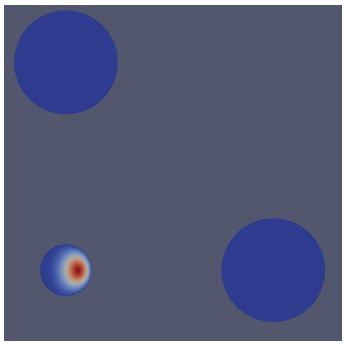
$$g_3(\mathbf{x},\mathbf{y}) := \begin{cases} \frac{1}{1+\|\mathbf{x}-\mathbf{y}\|^2}, & \text{if } 1.25 \leq \|x-y\| \leq 1.75, \\ 0, & \text{elsewhise.} \end{cases} \tag{5.5}$$

In compliance with Ω_2, we note that the upper bound assures that the two balls, $B_{0.5}(\binom{2}{0})$ and $B_{0.5}(\binom{0}{2})$, will be affected by the non-local boundary conditions. Nonetheless, the lower bound prevents the wave fronts to propagate along the boundaries, see figure 5.8.

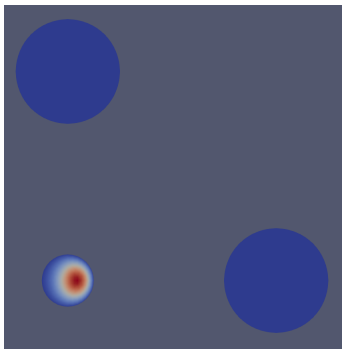
As a final domain, we set $\tilde{\Omega}_3$ to two arbitrarily shaped islands, whose boundaries are at a positive distance from each other. Figure 5.9 presents the triangular mesh $\tilde{\mathcal{M}}_3$ for $\tilde{\Omega}_3$ with 6474 degrees of freedom. In this case, the diffusion coefficient and the growth rate are chosen to be constant. The carrying capacity is constant as well. The initial condition is set to a point load having value 0.3 on the left lower corner of the right island. The decaying function kernel for the boundary conditions is chosen as in (5.3).



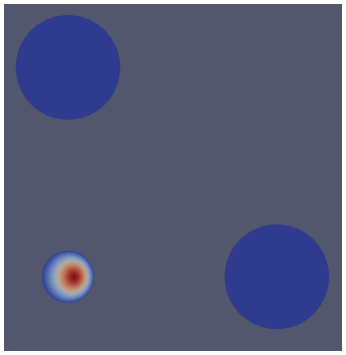
(a) Solution after $M = 100$ time steps.



(b) Solution after $M = 200$ time steps.

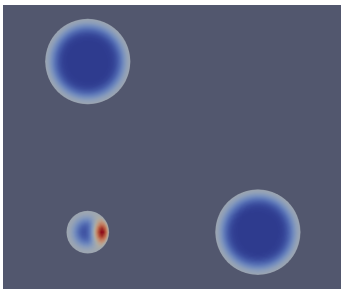


(c) Solution after $M = 300$ time steps.

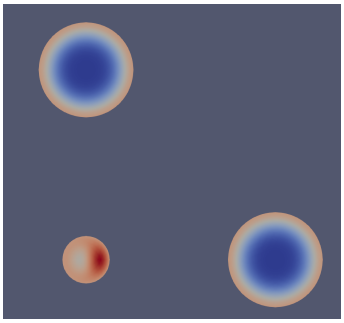


(d) Solution after $M = 400$ time steps.

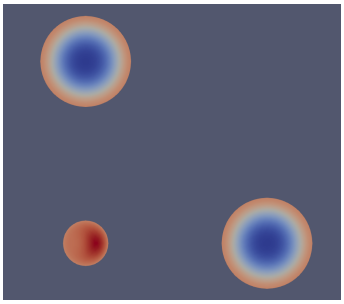
Figure 5.5 Solution vector with homogeneous Neumann boundary conditions visualised with ParaView [13] on $\tilde{\mathcal{M}}_2$.



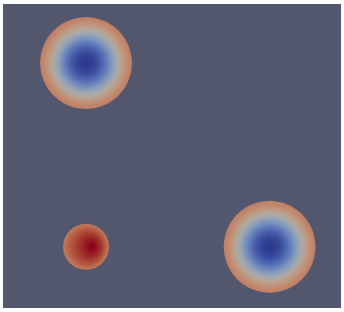
(a) Solution after $M = 100$ time steps.



(b) Solution after $M = 200$ time steps.

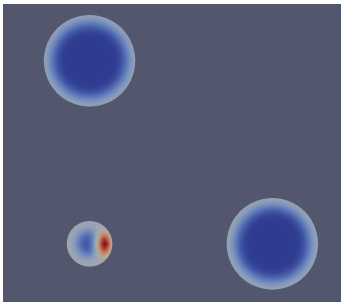


(c) Solution after $M = 300$ time steps.

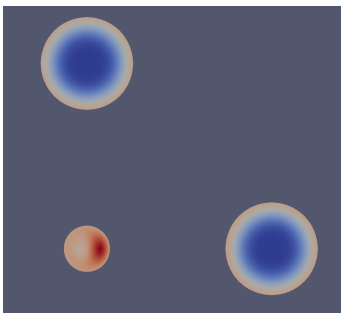


(d) Solution after $M = 400$ time steps.

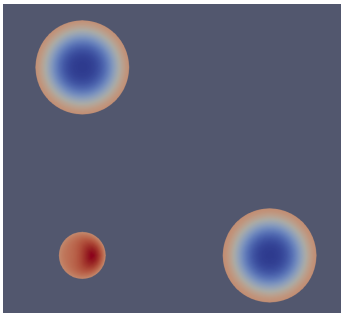
Figure 5.6 Solution vector with non-local boundary conditions using the kernel g_1 visualised with ParaView [13] on $\tilde{\mathcal{M}}_2$.



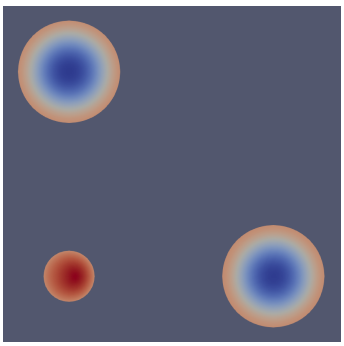
(a) Solution after $M = 100$ time steps.



(b) Solution after $M = 200$ time steps.

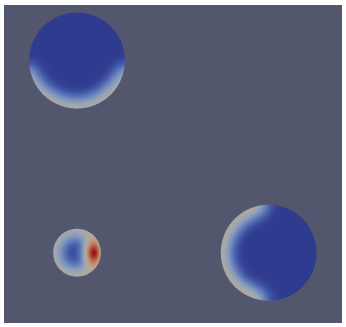


(c) Solution after $M = 300$ time steps.

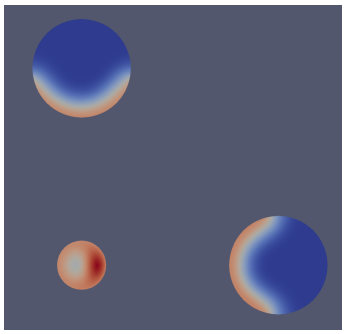


(d) Solution after $M = 400$ time steps.

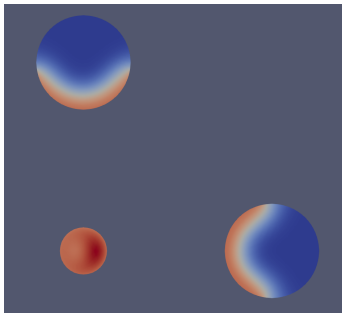
Figure 5.7 Solution vector with non-local boundary conditions using the kernel g_2 visualised with ParaView [13] on $\tilde{\mathcal{M}}_2$.



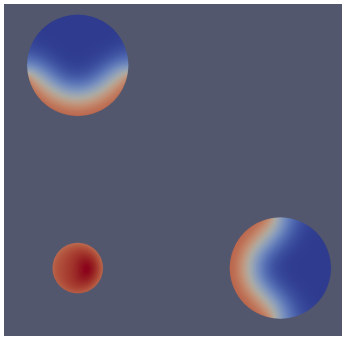
(a) Solution after $M = 100$ time steps.



(b) Solution after $M = 200$ time steps.



(c) Solution after $M = 300$ time steps.



(d) Solution after $M = 400$ time steps.

Figure 5.8 Solution vector with non-local boundary conditions using the kernel g_3 visualised with ParaView [13] on $\tilde{\mathcal{M}}_2$.

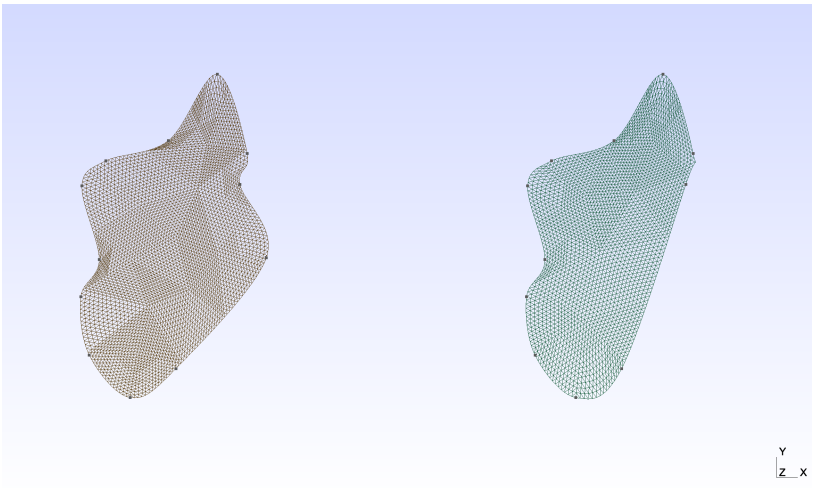


Figure 5.9 Triangular mesh $\tilde{\mathcal{M}}_3$ of $\tilde{\Omega}_3$ generated with Gmsh [5].

In this setting, we observe in figure 5.10 the travelling wave fronts propagating with finite speed on our model domain $\tilde{\Omega}_3$.

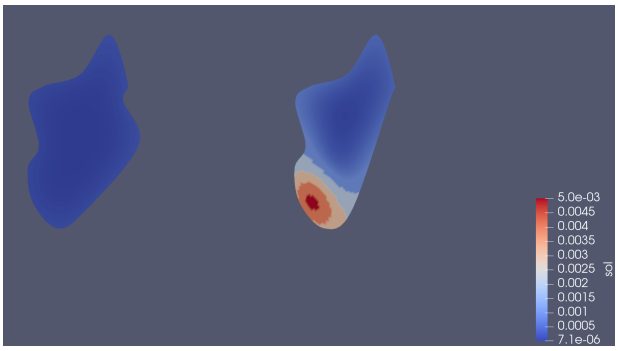
As a final experiment, we set the diffusion coefficient to a very small value, concretely $c = 0.0625$. Otherwise, we consider the same setting as above for figure 5.10; except for the growth factor λ, which we slightly decrease in order to balance the significant reduction of the diffusion coefficient. We have in mind to visualise the case where the non-local boundary conditions render the diffusion over the sea more attractive than the diffusion on land. Thus we would expect an accumulation of the solution on the boundaries $\partial\tilde{\Omega}_3$. Let me refer to figure 5.11 which meets our expectancy. We conceive the initial datum on one island, and otherwise a distribution on the coastlines of both islands rather than a rapid diffusion on land. Nevertheless, we also observe that there is no rapid growth along the coastlines. This can be explained due to the slight decrease in the growth factor λ.

Note that in all cases, the carrying capacity represents the limit value the solutions may attain. By performing further steps in the evolution process, it can be numerically verified that the wave profiles evolve towards the stationary state whose value is given by the carrying capacity.

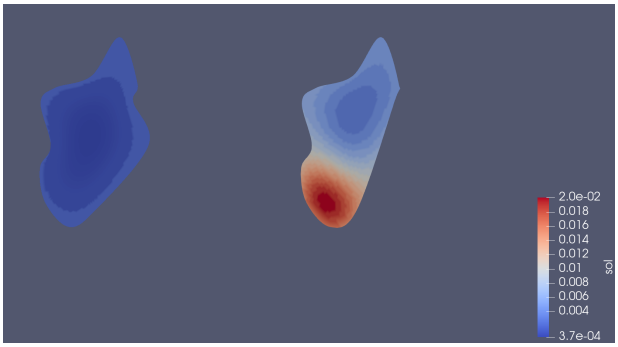
5.2 Convergence Analysis

This section now elucidates the convergence of the Strang splitting method applied to the initial boundary value problem (2.5). In the sequel, we consider the two arbitrarily shaped islands, $\tilde{\Omega}_3$, as in figure 5.9.

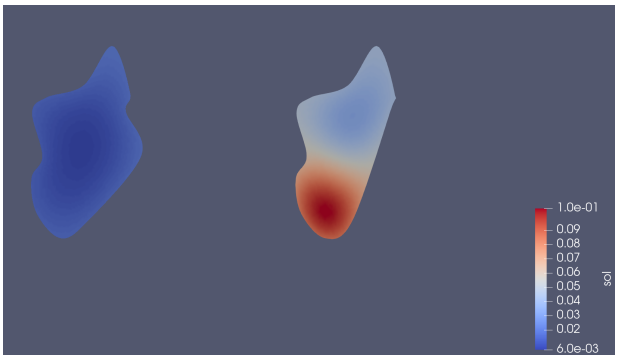
First, remark that it is hardly possible to compute an analytical solution for problem



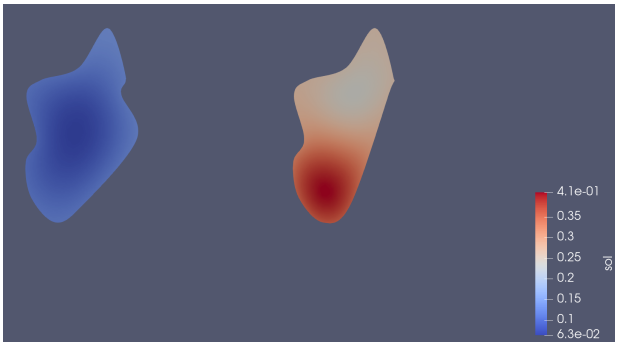
(a) Solution after $M = 120$ time steps.



(b) Solution after $M = 240$ time steps.

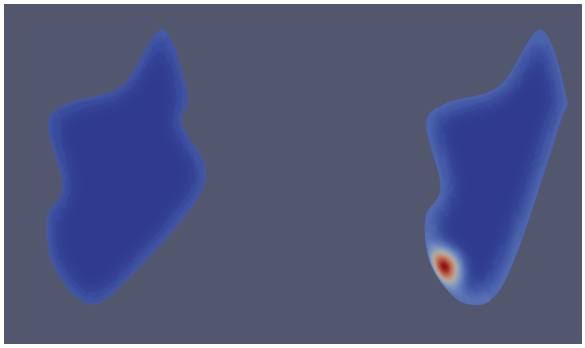


(c) Solution after $M = 360$ time steps.

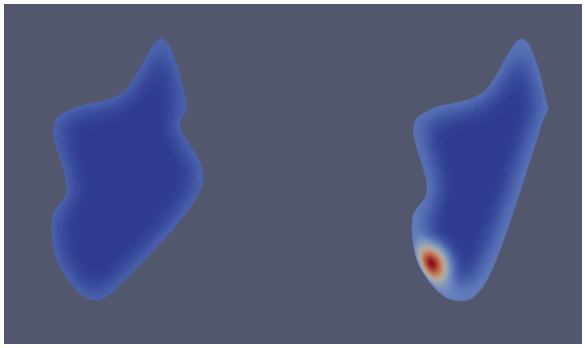


(d) Solution after $M = 480$ time steps.

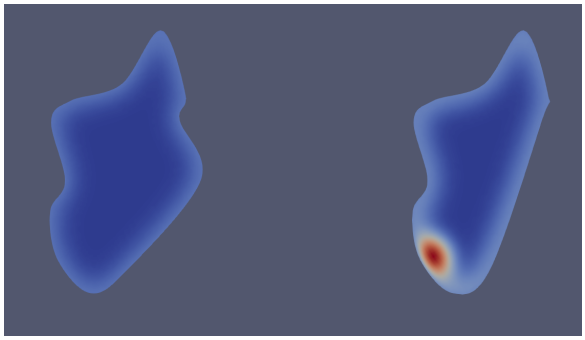
Figure 5.10 Solution vector visualised with ParaView [13] on mesh with $N = 6474$ degrees of freedom.



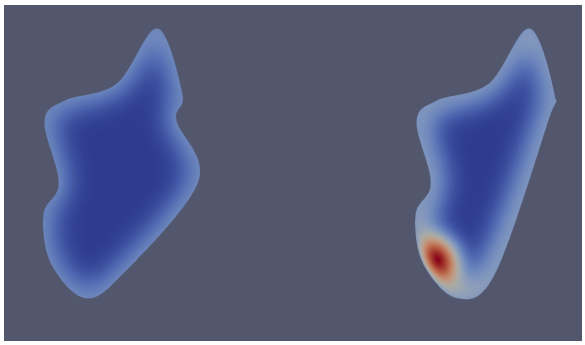
(a) Solution after $M = 360$ time steps.



(b) Solution after $M = 480$ time steps.



(c) Solution after $M = 600$ time steps.



(d) Solution after $M = 720$ time steps.

Figure 5.11 Solution vector with a minuscule diffusion coefficient visualised with ParaView [13] on mesh with $N = 6474$ degrees of freedom.

(2.5), not even in one spatial dimension, see [4]. Visually, I indeed do obtain the proclaimed *travelling wave solutions*, where I refer to figure 5.10. Now I aim at quantifying the convergence of our numerical scheme. To this end, I consider the initial boundary value problem (2.5) on the model domain $\tilde{\Omega}_3$.

Again, I choose a constant diffusion coefficient $c(t, \mathbf{x}) = c$ and a constant growth factor λ. Moreover, I choose the carrying capacity to be constant as well. As this capacity represents the maximal limit value for the population density, I expect the solution to attain this limit after a sufficiently long evolution. Furthermore, the same kernel g is chosen as in (5.5) with adequate bounds to account for the non-local boundary values. Finally, the initial population is again represented by a point load on one of the islands.

I perform three different convergence analyses. Firstly, I control the error with the mesh size. Secondly, I govern the error in the time step sizes; and lastly, I consider the solution by regulating both, the mesh width and the time step size synchronously, see section 6.2.8 in [7].

5.2.1 Convergence Studies for Mesh Width

I generate an initial mesh on the model domain $\tilde{\Omega}_3$, having in this case 39 degrees of freedom, and refine it with LehrFEM++'s **Mesh Hierarchy** class in the namespace **refinement** [9]. The finest mesh has 6474 degrees of freedom. Since no analytic solution is available, I instead refer to the solution on the finest mesh, see figure 5.12. I let the time step size be constant (i.e. $\tau = 0.01$), small enough to measure the discrete error in the mesh size. In figure 5.12, the $L^2(\tilde{\Omega}_3)$-norm of the difference of the solution on the considered mesh and the solution on the finest mesh is computed; that is

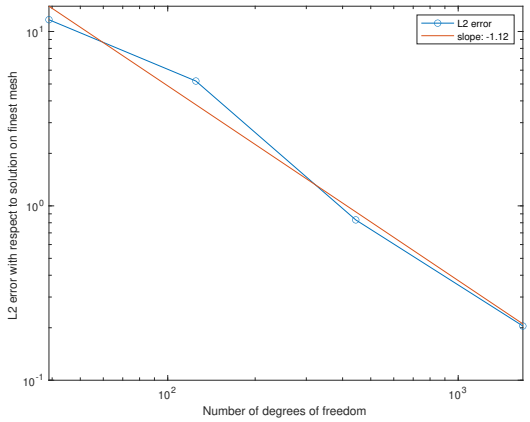
$$(e)_l := \sqrt{h(l)} \| u_{N(l)} - u_{N(5)} \|_{L^2(\tilde{\Omega}_3)}, \qquad l \in \{1, ..., 4\}, \tag{5.6}$$

where $u_{N(l)},\ l \in \{1, ..., 5\}$ represents the discrete solution vector on a mesh with $N(l)$ degrees of freedom after 100 time steps, and where $h \in \mathbb{R}$ denotes the mesh width. Note that $N(l)$ is given by

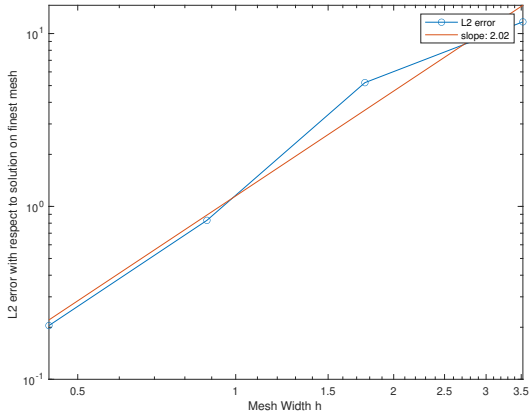
$$N(1) = 39, \quad N(2) = 125, \quad N(3) = 444, \quad N(4) = 1670, \quad N(5) = 6474.$$

Also notice that we have to interpolate the solution vectors in the finite element setting $u_{N(l)}, l \in \{1, ..., 4\}$ onto the finest mesh with $N(5) = 6474$ degrees of freedom such that the difference in (5.6) is well-defined. This is done with LehrFEM++'s **MeshFunctionFE**-, **MeshFunctionTransfer**- and **NodalProjection**-functionality [9]. Concretely, this means that we consider the solution vectors on the coarse meshes with $N(l), l \in \{1, ..., 4\}$ degrees of

freedom. For these we then extract the corresponding mesh functions using the **MeshFunctionFE**-functionality. The result is transferred onto the finest mesh via **MeshFunctionTransfer**. Finally, with the **NodalProjection**-functionality we obtain the solution vectors $u_{N(l)}, l \in \{1, ..., 4\}$ interpolated onto the finest mesh.



(a) $L^2(\tilde{\Omega}_3)$-norm against number of degrees of freedom N.



(b) $L^2(\tilde{\Omega}_3)$-norm against mesh size h.

Figure 5.12 $L^2(\tilde{\Omega}_3)$-norm of difference of final solution after 100 time steps on meshes with number of degrees of freedom $N \in \{39, 125, 444, 1670, 6474\}$ with respect to the final solution on the finest mesh with $N = 6474$ degrees of freedom.

We observe an algebraic convergence rate of 1.12 in terms of the number of degrees of freedom for the Strang splitting method, and thus an algebraic rate of convergence of 2.02 in terms of the mesh size. Therefore, the error behaves asymptotically as $\mathcal{O}(N) = \mathcal{O}(h^2)$, where h denotes the mesh width.

5.2.2 Convergence Studies for Time Step Sizes

In this section, I instead govern the time step sizes to analyse the incurred error. To this end I consider a sufficiently fine mesh $\mathcal{M}_3$ for $\tilde{\Omega}_3$ with 1670 degrees of freedom. I consider different number of time steps M, namely

$$M(1) = 10, \quad M(2) = 20, \quad M(3) = 40, \quad M(4) = 80, \quad M(5) = 160, \quad M(6) = 320.$$

The time step sizes $\tau(l) = \frac{T}{M(l)},\ l \in \{1, ..., 6\}$ for final time $T = 1$ are then correspondingly,

$$\tau(1) = 0.1, \quad \tau(2) = 0.05, \quad \tau(3) = 0.025,$$
$$\tau(4) = 0.00125, \quad \tau(5) = 0.000625, \quad \tau(6) = 0.0003125.$$

Again, having no analytical solution at hand, I instead refer to the solution computed with the smallest time step size $\tau(6) = 0.0003125$. I then control the $L^2(\tilde{\Omega}_3)$-norm of the difference of the solutions $u_{M(l)},\ l \in \{1, ..., 5\}$ and the solution with most time steps, $u_{M(6)}$. In figure 5.13, you observe

$$(e)_l := \sqrt{\tau(l)} \|u_{M(l)} - u_{M(6)}\|_{L^2(\tilde{\Omega}_3)}, \qquad l \in \{1, ..., 5\}, \tag{5.7}$$

plotted against the number of time steps $M(l),\ l \in \{1, ..., 5\}$. There is an algebraic rate of convergence of -1.7, and thus an asymptotic behaviour of $\mathcal{O}(M^{-1.7}) = \mathcal{O}(\tau^{1.7})$. Theoretically, we would expect a convergence rate of $\mathcal{O}(\tau^2)$, for the employed SDIRK method is of order 2, and for the reaction term we have applied an exact evolution operator.

5.2.3 Convergence Studies for Mesh Width linked to Time Step Sizes

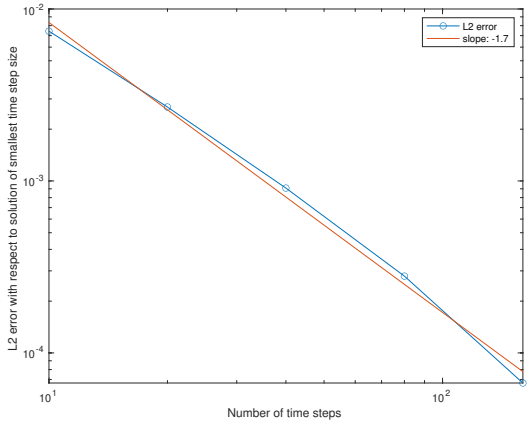
Finally, we link the mesh sizes to the time step sizes, and control the error norm incurred thereof.

In theory, the error norm behaves as $\mathcal{O}(h^2 + \tau^2)$. Thus, we impose a linear dependence of mesh width to time step size. Concretely, we consider different meshes refined as above for $N \in \{39, 125, 444, 1670, 6474\}$ degrees of freedom. We start with $M(1) = 100$ number of time steps. For $l \in \{2, ..., 5\}$ we set

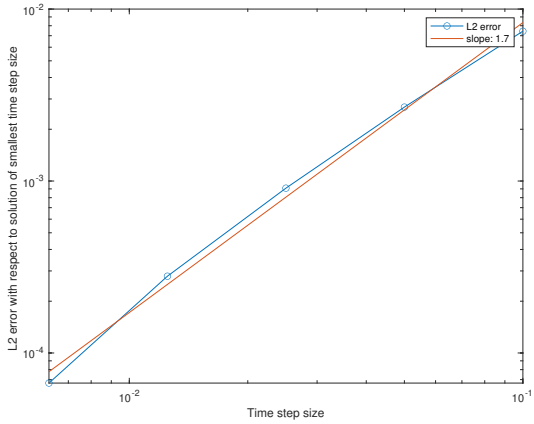
$$\tau(l) = \frac{\tau(1)}{h(1)} h(l), \tag{5.8}$$

where $h(l)$ is the mesh size for the mesh with $N(l)$ degrees of freedom. The corresponding number of time steps $M(l)$ are rounded to the next integer,

$$M(l) = \lceil \frac{T}{\tau(l)} \rceil, \quad l \in \{2, ..., 5\}, \tag{5.9}$$



(a) $L^2(\tilde{\Omega}_3)$-norm against number of time steps M.



(b) $L^2(\tilde{\Omega}_3)$-norm against time step size τ.

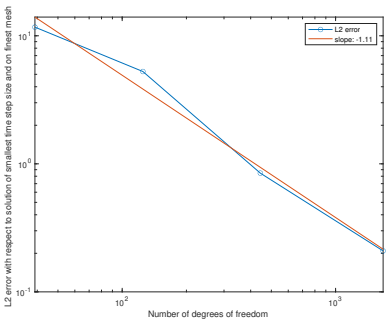
Figure 5.13 $L^2(\tilde{\Omega}_3)$-norm of difference of final solutions with number of time steps $M \in \{10, 20, 40, 80, 160, 320\}$ with respect to $M(6) = 320$ time steps on a mesh with $N = 1670$ degrees of freedom.

with final time $T = 1$.

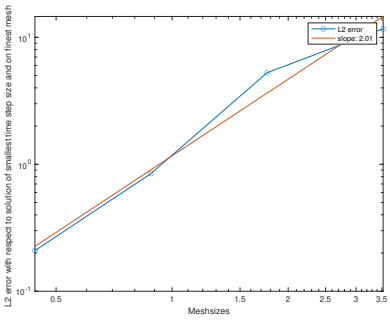
Let me refer to figure 5.14, which illustrates

$$(e)_l := \sqrt{h(l)} \| u_{N(l),M(l)} - u_{N(5),M(5)} \|_{L^2(\tilde{\Omega}_3)}, \qquad l \in \{1, ..., 4\}, \tag{5.10}$$

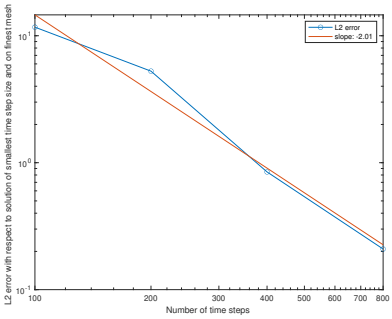
where $u_{N(5),M(5)}$ is the reference solution on the finest mesh. Again, let me point out that the solution on the coarser meshes $u_{N(l),M(l)}, l \in \{1, ..., 4\}$, were interpolated onto the finest mesh using LehrFEM++'s functionalities as above in section 5.2.1. We conceive the asymptotes $\mathcal{O}(N + M^2) = \mathcal{O}(h^2 + \tau^2)$, as predicted.



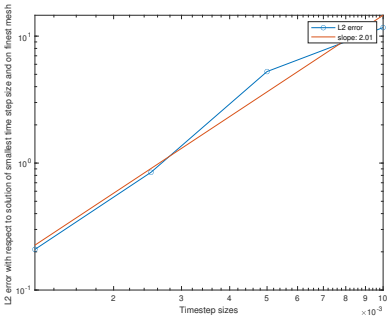
(a) $L^2(\tilde{\Omega}_3)$-norm against number of degrees of freedom N.



(b) $L^2(\tilde{\Omega}_3)$-norm against mesh sizes h.



(c) $L^2(\tilde{\Omega}_3)$-norm against number of time steps M.



(d) $L^2(\tilde{\Omega}_3)$-norm against time step size τ.

Figure 5.14 $L^2(\tilde{\Omega}_3)$-norm of difference of final solutions with number of time steps $M \in \{100, 200, 400, 800\}$ on meshes with $N \in \{39, 125, 444, 1670\}$ degrees of freedom with respect to $M(5) = 976$ time steps on a mesh with $N = 6474$ degrees of freedom.

Chapter Six

Results

In this chapter, I will provide the visualised results for the evolution of the population density from 200000 years ago up to 15000 years ago, according to the parameters as in the first approach in section 6.1.1. Let me refer to figure 6.1 for the visualisation of the triangular mesh $\mathcal{M}$ on our domain Ω.

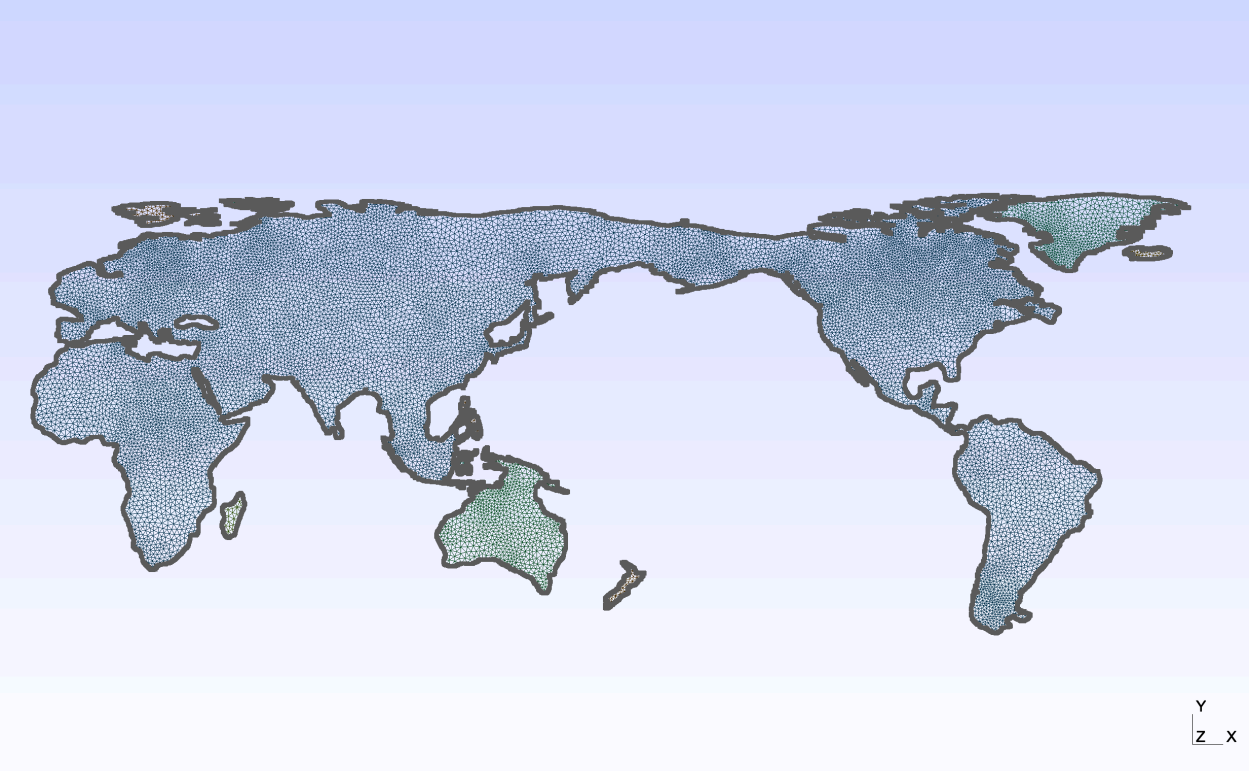


Figure 6.1 Triangular mesh $\mathcal{M}$ of Ω generated with Gmsh [5]. Number of degrees of freedoms: 15463.

6.1 Parameters

In this paragraph, I discuss the choice of parameters in the implementation.

6.1.1 First Approach

Firstly, I used a constant diffusion coefficient, neglecting the earth's topography. In order to account for high mountain chains, impeding the dispersal of population, the diffusion coefficient might be chosen piecewise constant only, see paragraph 6.1.2 on the second approach below.

Secondly, I chose a constant growth factor λ. Here again, one could choose a time-dependent growth factor, to respect differences in growth rates per epoch. However, this is more difficult to take into account, due to the sparsity of data for the concerned time epoch.

Thirdly, the carrying capacity plays an important role in my model. In my case, the carrying capacity relies on the dispersal of the Homo Sapiens according to [17]. Initially, I attributed on each degree of freedom a low carrying capacity. Then, for each epoch, I put a point load onto the surface coinciding with the location of the humans at this time. I let this point load diffuse over a certain area, such that *visually* the carrying capacity is augmented in the area where the humans assembled in this epoch according to [17]. As time evolves, I added up all the contribution of the carrying capacity, yielding a final carrying capacity for the last time steps. The values of the capacities range from 0 to 1.

Fourthly, for the non-local boundary value, I chose a decaying kernel depending on the distance between two coastlines. Let me refer to section 2.1. Specifically, the kernel $g : \partial\Omega \times \partial\Omega \to \mathbb{R}$ is chosen as

$$g(\mathbf{x}, \mathbf{y}) = \begin{cases} \frac{1}{\|\mathbf{x}-\mathbf{y}\|^2+1}, & \text{if } \ 5 \leq \|x-y\| \leq 30, \\ 0, & \text{elsewhise.} \end{cases} \tag{6.1}$$

Note that I have chosen lower and upper bounds for the decaying function kernel. The bounds are chosen in accordance to the domain Ω. I intended to model sea crossings of near-by boundaries, as well as travels along coastlines. Yet, it also makes sense to impose a lower bound, for it is reasonable to suggest that people prefer to travel on land to close-by sites on the same coast instead of directly taking the boat. However, note that this does not mean that traveling along the same coastline is suppressed. Indeed, all coastlines beyond the lower threshold can be reached.

And finally, the initial population is set to a small point load located in Eritrea 200000 years ago.

6.1.2 Second Approach

In a second approach, I modified the diffusion coefficient in such a way as to account for high mountain ranges on the earth's surface complicating the diffusion of the population. Therefore, instead of a constant diffusion coefficient, I now chose a piecewise constant coefficient depending on the coordinates $c(t, \mathbf{x}) = c(\mathbf{x}), \;\; t \in \; [0, T], \;\; \mathbf{x} \in \Omega$. The other parameters were left unchanged.

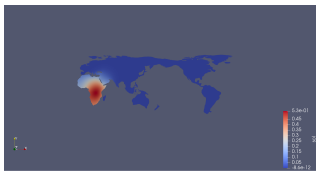
Due to the fact that the differences in the two approaches are visually negligible - you may only perceive a slight decrease in the dispersal over the mountain ranges - I restrict the provision of the result to figure 6.2, visualising the first approach.

6.2 Visualisation

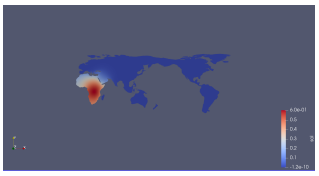
With the choice of parameters according to the first approach in section 6.1.1 above, we obtain the travelling wave fronts on the earth's surface in figure 6.2. Clearly, the results are of a qualitative nature due to sparse availability of data from pre-historic times.

(a) Population density 200000 years ago.

(b) Population density 150000 years ago.

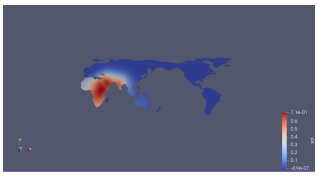


(c) Population density 130000 years ago.

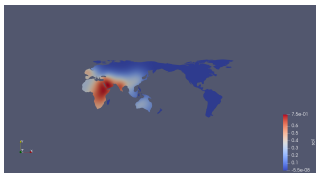


(d) Population density 100000 years ago.

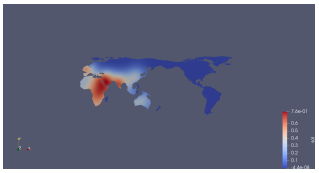
(e) Population density 70000 years ago.



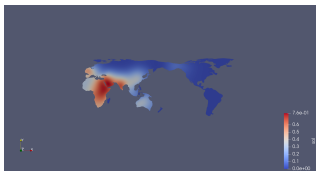
(f) Population density 65000 years ago.



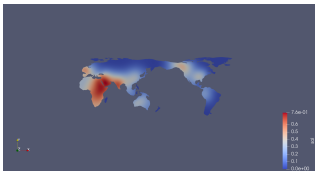
(g) Population density 50000 years ago.



(h) Population density 45000 years ago.



(i) Population density 25000 years ago.



(j) Population density 15000 years ago.

Figure 6.2 Solution vector visualised with ParaView [13] on mesh with $N = 15463$ degrees of freedom.

Chapter Seven

Conclusions

We have seen that the Fisher equation is fairly well suited to model population dynamics. Equipped with non-local boundary condition, we were able to work on a disconnected domain, where we have taken into account the dispersal over "straits".

The numerical scheme based on finite element methods yields a beautiful variational problem. The Strang splitting method is fairly well suited. For it allows to employ a time stepping scheme for the diffusion and the reaction term separately. Consequently, we were able to use an $\mathrm{L}(\pi)$-stable method of second order for the diffusion term, as well as the exact evolution operator for the reaction term, for the reactive part of Fisher equation coincides with Verhulst's logistics equation, whose solution is analytically computable.

The convergence studies assert an asymptotic behaviour of our numerical scheme as $\mathcal{O}(h^2 + \tau^2)$, where h is the mesh width and τ the time step size. This coincides with the theoretical prediction for the method we have employed. Moreover, we have also observed travelling wave fronts on several bounded domains, in each of which, given a compactly supported initial datum, the solutions propagate at finite speed.

The choice of parameter was set such that the result coincided visually with the desired output. Especially the carrying capacity maps were assembled in such a way as to adhere to the agreement of paleontologists' view on the dispersal of our ancestors. We even accomplished to account for the topology on the earth's surface, by choosing a piecewise constant diffusion coefficient.

To improve efficiency of our method, we could make use of Quad Trees for a more efficient assembly of the non-standard Galerkin matrix for the second part of the non-local boundary term (4.3). Apart from this, instead of the employment of the Strang splitting method, a semi-implicit method could be deployed, such as a Rosenbrock-Wanner method.

REFERENCES

[1] F. Andreu, V. Caselles, J. M. Mazon. A Fisher-Kolmogorov equation with finite speed of propagation. Journal of Differential Equations. Vol. 248. pp. 2528–2561. 2010.

[2] P. K. Brazhnik, J. T. Tyson. On traveling wave solutions of Fisher's equation in two spatial dimensions. SIAM J. Applied Mathematics. Vol. 60, No. 2, pp. 371–39. 1999.

[3] Eigen. C++ Template Library for Linear Algebra: Matrices, Vectors, Numerical Solvers, and related Algorithms. Version 3.3.7. http://eigen.tuxfamily.org/index.php?title=Main_Page.

[4] R. A. Fisher. The Wave of Advance of Advantageous Genes. Annals of Human Genetics, Volume 7, Issue 4. 1937.

[5] C. Geuzaine and J.-F. Remacle. Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. International Journal for Numerical Methods in Engineering 79(11), pp. 1309-1331, 2009.

[6] R. Hiptmair. Numerical Methods for Computational Science and Engineering. Lecture Notes, 2015. http://www.sam.math.ethz.ch/∼hiptmair/tmp/NumCSE/NumCSE15.pdf

[7] R. Hiptmair. Numerical Methods for Partial Differential Equations. Lecture Notes, 2020.

[8] A. N. Kolmogorov, I. G. Petrovskii, N. S. Piskunov. Étude de l'équation de la diffusion avec croissance de la quantité de matière et son application à un problème de biologie. Moskow Universal Bulletin Mathematics. Volume 1, Issue 1. 1937.

[9] LehrFEM++. A simple Finite Element Library for teaching. https://craffael.github.io/lehrfempp/getting_started.html.

[10] J. H. Merkin, D. J. Needham. Propagating reaction-diffusion waves in a simple isothermal quadratic autocatalytic chemical system. J. Engrg. Math., Vol. 23, pp. 343–356. 1989.

[11] E. C. Titchmarsh, The Theory of Functions, 1939, (Second edition). Oxford University Press. Chapter 5.

[12] Data Announcement 88-MGG-02, Digital relief of the Surface of the Earth. NOAA, National Geophysical Data Center, Boulder, Colorado, 1988.

[13] ParaView: An open-source, multi-platform data analysis and visualization application. https://www.paraview.org.

[14] W. P. Petersen, S. Callegari, G. R. Lake, N. Tkachenko, J. D. Weissmann, C. P. E. Zollikofer. A Stable Finite-Difference Scheme for Population Growth and Diffusion on a Map. PLoS ONE. Volume 12. Issue 1. 2017.

[15] QGIS. A Free and Open Source Geographic Information System. https://qgis.org/de/site/.

[16] P. Wessel, W. H. F. Smith. A Global Self-consistent, Hierarchical, High-resolution Geography Database. SOEST, University of Hawai'i, Honolulu, HI; NOAA Geosciences Lab, National Ocean Service, Silver Spring, MD. 2017. http://www.soest.hawaii.edu/pwessel/gshhg/

[17] Wikipedia, the free encyclopedia. Settlement of the Americas. https://en.wikipedia.org/wiki/Settlement_of_the_Americas. Last access on: 06.05.2020.

APPENDIX

The implementation can be accessed at *https://github.com/craffael/lehrfempp/tree/master/projects/FisherKPP*. The main simulation on the earth constitutes of the files *strangsplitting.h*, *strangsplitting.cc* and *strangsplitting_main.cc*.

Furthermore, all model problems discussed in Chapter 5 have a corresponding file named *modelproblem_*_main.cc*. Clearly, these rely on the previously mentioned files *strangsplitting.h* and *strangsplitting.cc* for the underlying numerical schemes.

Moreover, there are three files for the convergence analyses. These are implemented in the files *convergence_*_main.cc*. The plots are then generated with the Matlab scripts provided in the folder *convergence_scripts*.

Lastly, note that all underlying meshes used in the main simulation, in the model problems and in the convergence studies can be accessed in the folder *meshes*.