INSTITUTE FOR PARALLEL AND DISTRIBUTED SYSTEMS

SIMULATION TECHNOLOGY DEGREE COURSE

Master's Thesis

Submitted to the University of Stuttgart

Efficient Algorithms for Geodesic Shooting in Diffeomorphic Image Registration

Examiner
Prof. Dr. Miriam MEHL
Institute for Parallel and Distributed Systems

Supervisor
Dr. Ing. Andreas MANG
Department of Mathematics, University of Houston, USA

Submitted by

Author	Felix HUBER
SimTech-Nr.	58
Submission date	June 2019

Abstract

Diffeomorphic image registration is a common problem in medical image analysis. Here, one searches for a diffeomorphic deformation that maps one image (the moving or template image) onto another image (the fixed or reference image). We can formulate the search for such a map as a PDE constrained optimization problem. These types of problems are computationally expensive. This gives rise to the need for efficient algorithms.

After introducing the PDE constrained optimization problem, we derive the first and second order optimality conditions. We discretize the problem using a pseudo-spectral discretization in space and consider Heun's method and the semi-Lagrangian method for the time integration of the PDEs that appear in the optimality system. To solve this optimization problem, we consider an L-BFGS and an inexact Gauss-Newton-Krylov method. To reduce the cost of solving the linear system that arises in Newton-type methods, we investigate different preconditioners. They exploit the structure of the Hessian, and use algorithms to efficiently compute an approximation to its inverse. Further, we build the preconditioners on a coarse grid to further reduce computational costs.

The different methods are evaluated for two-dimensional image data (real and synthetic). We study the spectrum of the different building blocks that appear in the Hessian. It is demonstrated that low rank preconditioners are able to significantly reduce the number of iterations needed to solve the linear system in Newton-type optimizers. We then compare different optimization methods based on their overall performance. This includes the accuracy and time-to-solution. L-BFGS turns out to be the best method, in terms of runtime, if we solve solving for large gradient tolerances. If we are interested in computing accurate solutions with a small gradient norm, an inexact Gauss-Newton-Krylov optimizer with the regularization term as preconditioner performs best.

Acknowledgments

This work was supported by a fellowship within the FITweltweit program of the German Academic Exchange Service (DAAD). Without this scholarship this thesis would not have been possible.

Contents

Nomenclature

$\mathrm{ad}_v\, m$	$(\nabla v)m - (\nabla m)v$
$\mathrm{ad}_v^\dagger$	adjoint operator of ad_v. $\mathrm{ad}_v^\dagger m = (\nabla v)^{\mathrm{T}} m + (\nabla m)v + v\,\mathrm{div}(m)$
$L^\dagger$	adjoint of operator L
β	regularization parameter
$\mathrm{diag}_{i=1,\dots,n}(a_i)$	diagonal matrix $A \in \mathbb{R}^{n\times n}$ with $A_{i,i} = a_i$
$\mathrm{div}\, f = \nabla \cdot f$	divergence of a vector valued function, only in spatial dimensions
E	objective functional
E_h	discretized objective functional
$\mathcal{F}f$	Fourier transformation of f
$\mathrm{grad}\, f = \nabla f$	gradient of a function, only in spatial dimensions
H	Hessian
H_c	Hessian on a coarser discretization
H_{mis}	mismatch term of the Hessian $H = \beta H_{\text{reg}} + H_{\text{mis}}$
H_{reg}	regularization term of the Hessian $H = \beta H_{\text{reg}} + H_{\text{mis}}$
h_t	time step size
h_x	discretization width in each spatial dimension
I	identity map $\mathrm{I}(x) := x$ or matrix $\mathrm{I}x := x$
L^{-1}	inverse of operator L
K	inverse of L: $v = Km$
L	regularization operator or matrix in the discrete setting

M_{lr}^{-1}	preconditioner using a low-rank approxiation of the mismatch term
$M_{\mathrm{lr,c}}^{-1}$	M_{lr} preconditioner build on a coarser discretization
N_x	number of nodes in each spacial dimension
A^{T}	transpose of matrix A

BFGS	Broyden–Fletcher–Goldfarb–Shanno algorithm
CFL	Courant-Friedrichs-Lewy condition
CG	conjugate gradient method
FFT	Fast-Fourier-Transform
FN	(full) Newton's method
GD	gradient descent
GMRES	generalized minimal residual method
GN	Gauss-Newton
L-BFGS	Limited memory BFGS algorithm
LDDMM	large deformation diffeomorphic metric mapping
PCG	preconditioned conjugate gradient method
PDE	partial differential equation

1 Image Registration

Image registration is an *inverse problem*. It involves the process of finding a map ϕ that describes how a *template image* I_0 can be deformed to resemble a *reference image* I_1 in some sense [42]. This problem often arises in medical image analysis. Common examples include the registration of magnetic resonance imaging (MRI) scans onto a standardized template, normalizing time series of images, the reconstruction of a 3D model from multiple MRI scan slices or combining images from multiple measuring techniques [41, 42, 20, 49].

1.1 Overview

Given two images $I_0 : \Omega \to \mathbb{R}$ and $I_1 : \Omega \to \mathbb{R}$ we try to find a function $\phi : \Omega \to \Omega$ that maps the first image onto the second so that $I_0 \circ \phi^{-1} \approx I_1$. The image domain Ω is a compact subset of $\mathbb{R}^d$ for $d \in \{2,3\}$. Furthermore, we want to restrict admissible functions for ϕ to be diffeomorphisms. Using diffeomorphisms ensures that structures do not vanish, no folding is introduced and that neighborhood structures are preserved. In general, this problem is ill-posed [20]. The solution is not guaranteed to be unique as different mappings can yield the same or very similar results. Moreover, small perturbations in the images may lead to vastly different solutions. A strategy to overcome ill-posedness is to introduce a regularization model [19]. Consequently, the optimization problem consists of two term: a mismatch term and a regularizer. The mismatch term measures the fidelity of the deformed template image $I_0 \circ \phi^{-1}$ and the reference image I_1. The regularization term favors diffeomorphisms that are close to the identity in a predefined the metric [39].

It is not immediately clear, how one can minimize over the group of diffeomorphisms. One option is to introduce an artificial time variable $t \in [0,1]$ and use a time dependent velocity field v and obtain the associated diffeomorphism $\phi : \Omega \times [0,1] \to \Omega$ as the solution of the differential equation $\partial_t \phi(x,t) = v(\phi(x,t),t)$ where $\phi(x,0) = x$ [52, 53, 17, 39, 56]. Under certain smoothness requirements on v it follows that ϕ is a diffeomorphism [17, 46] and therefore admissible for the matching problem $(I_0 \circ \phi^{-1})(\cdot\,,1) \approx I_1(\cdot)$. This leaves us with a well posed problem [17].

Using v as a control variable, the problem can be reformulated so that we search for a smooth time dependent velocity field. Furthermore, it can be shown that the minimizing velocity field satisfies an Euler-Lagrange-Equation, which defines the evolution of v over time [39, 40]. This allows us to reduce the optimization space to the initial conditions of the Euler-Lagrange-Equation. In analogy to shooting a projectile into the air, where the trajectory is defined by the initial angle, velocity and physical laws, this method leads to a so called *geodesic shooting method*: The evolution of the velocity (and with it the diffeomorphism) is described by an initial velocity and the Euler-Lagrange-Equation.

Given the minimization functional we can now state an optimization problem that is constrained by the Euler-Lagrange-Equation. This problem can be solved using a gradient descent method or Newton's method. While Newton's methods should give better convergence rates, it also requires matrix-vector-products with the inverse of the Hessian of the minimization functional H^{-1}. Direct methods to invert H are in general not applicable as forming and and storing H is too expensive and not possible. Hence, we will consider iterative methods instead. To improve the convergence rate of the linear solver and reduce the computational effort needed in each Newton step, we are interested in efficient preconditioners.

1.2 Own Contribution and Thesis Outline

This work is based on the OCREG code [32, 33, 34] for diffeomorphic image registration. In this work the code has been extended by the geodesic shooting formulation for scalar and vector valued momentums, the corresponding gradients and Hessians for the optimization, a L-BFGS optimizer and different preconditioners for Newton's method.

Starting with chapter 2 we show how the PDE constrained minimization problem for geodesic shooting can be derived from a mismatch term and the requirement that the final mapping ϕ is a diffeomorphism.

In chapter 3 we reformulate the constrained minimization problem as an unconstrained minimization problem. We shortly discuss different optimization methods, which can be used and derive the necessary gradient and Hessian.

These expressions are then discretized in chapter 4. We discuss the spatial and temporal discretization. We also show how the PDEs arising in the gradient and Hessian expressions can be solved using Heun's method or a semi-Lagrangian scheme.

In chapter 5 we introduce a new preconditioner which can be used in the linear solver needed in Newton-type optimizers. This preconditioner is based on a low rank approximation of the Hessian and only requires matrix-vector-products during its setup. Hence, it is matrix free and does not require knowledge of individual matrix entries. Furthermore, it can be build on a coarser discretization.

We evaluate the efficiency of the preconditioner in chapter 6. We evaluate the preconditioner on its own based on its ability to reduce the costs of solving the linear Hessian system and based on its efficiency during the entire registration process. Furthermore, we compare the results of the Newton optimizers against L-BFGS optimizers.

1.3 Related Work

The image registration problem is often solved by minimizing a functional

$$E = E_{\text{reg}} + E_{\text{mis}}$$

which consists of a mismatch term E_{mis} and a regularization term E_{reg} [42]. The mismatch term defines how we measure the mismatch between two images. The regularization term usually enforces some type of smoothness on the solution to ensure well-posedness of the minimization problem [20]. Because a specific regularization term also implies what kind of solution is expected, a vast variety of different regularizers have been proposed for different problems [20, 42, 46].

In this work, we want to focus on registration problems with large deformations. Therefore, we use *large deformation diffeomorphic metric mapping* (LDDMM) methods [6]. To allow for large deformations, an artificial time variable $t \in [0,1]$ is introduced. The transformation of the image happens gradually over time. Early work goes back to [12], which models the transformation as an evolving fluid. Furthermore, we are interested in diffeomorphic maps, which ensures that the computed map ϕ is a bijections with a smooth inverse [3]. Diffeomorphic maps can be obtained by inverting for a time dependent velocity field subject to certain smoothness requirements [17] which leads to a PDE constrained optimization problem [6]. The minimization problem can be seen as a shortest path problem or path of minimal energy problem in the space of diffeomorphisms and defines a metric in this space [39, 6].

Geodesic shooting methods, such as [40, 54], only invert for an initial value instead of solving for a time dependent velocity field. The initial value combined with an additional PDE constraint on the minimization problem fully describe the evolution of the diffeomorphism over time.

Optimizations methods for the discretized LDDMM problem often involve first order methods such as the gradient descent method. LDDMM was initially presented in [6] to invert for a velocity field and is based on the theoretical results described in [52, 53]. Different gradient descent method for a shooting formulation has been presented in [54, 57]. Second order methods have been explored in [5, 24, 23, 32, 33, 35, 38, 34, 36], among others. [32] uses a preconditioned Newton-Krylov method to invert for a time dependent velocity field. [55] uses the Hessian matrix with respect to a vector valued initial momentum to estimate uncertainties in regularization parameters.

Because second order methods usually involve the solutions of a linear system, preconditioners can greatly improve the efficiency of such methods. Common choices for preconditioners include the inverse regularization operator [45, 32, 2, 37]. In [32] and [36] a two level preconditioner is proposed to invert for a velocity field. Further preconditioners have been discussed in [7, 48].

2 Geodesic Shooting

This introduction is mostly based on the work of [39, 40, 59], which provide a good overview of this topic. Another good introduction can be found in [58]. As mentioned before the regularization term favors diffeomorphisms close to the identity. This is done by penalizing the *geodesic length* of the path between the identity and the final diffeomorphism. Because the gradual evolution of the diffeomorphism is described by the velocity field v, the geodesic length is measured by integrating the velocity in a chosen norm $\|\cdot\|_V$ over the artificial time $t \in [0,1]$:

$$E_{\text{reg}}(\phi) := \int_0^1 \|v(\,\cdot\,,t)\|_V^2 \,\mathrm{d}t.$$

The norm of the velocity field is usually chosen to be a norm of a Sobolev space and defined as

$$\|v\|_V^2 := \langle v, Lv \rangle_{L^2}$$

with linear differential operator L. The differential operator has to be chosen so that the velocities are sufficiently smooth and thus give rise to a diffeomorphism [17, 46]. The required smoothness depends on the number of dimensions of the image domain. A common choice is $L = (\mathrm{I} - \Delta)^\gamma$ with $\gamma > 0$ [39, 55, 46]. Using the second Sobolev embedding theorem one can show that for two dimensional images $\gamma > 2$ is sufficient [46].

The shortest geodesic path between two diffeomorphisms satisfies the so-called *EPDiff equation*

$$0 = \partial_t m + \mathrm{ad}_v^\dagger m, \tag{2.1}$$

which is the Euler-Lagrange-equation for the corresponding variational problem for the *momentum* $m = Lv$. Solutions to this PDE are called *geodesics*.

It can be shown that the minimizer to the registration problem

$$v^* = \underset{v}{\operatorname{argmin}} \left(\frac{1}{2}\beta \int_0^1 \|v(\,\cdot\,,t)\|_V^2 \,\mathrm{d}t + E_{\text{mis}}(I_1, I(1)) \right)$$

subject to

$$0 = \partial_t I + v \cdot \nabla I$$

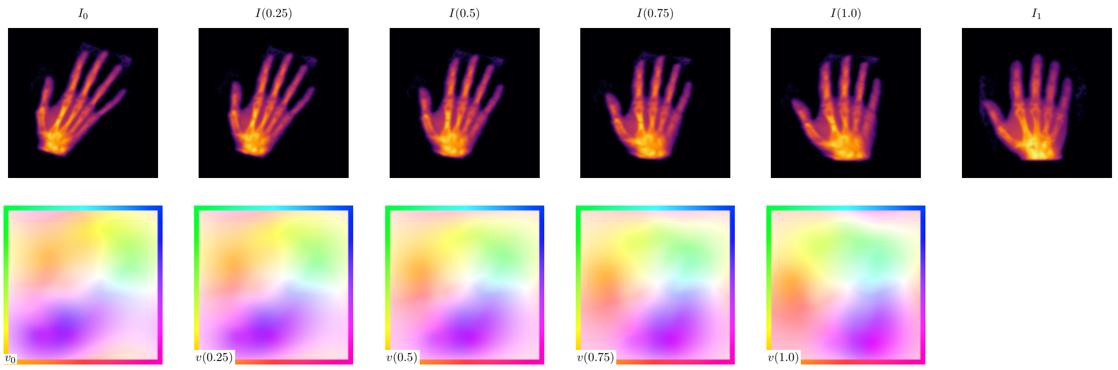

Figure 2.1: Evolution of the image (top) and the velocity field (bottom) over time.

is a geodesic and satisfies (2.1). This motivates the use of shooting methods. Because an initial velocity together with the EPDiff equation fully describes a geodesic, the minimization can be stated in terms of an initial velocity v_0 and the EPdiff (2.1). The minimization problem then reads

$$v_0^* = \underset{v_0}{\operatorname{argmin}} \left(\frac{1}{2}\beta \int_0^1 \|v(\cdot\,,t)\|_V^2 \,\mathrm{d}t + E_{\text{mis}}(I_1, I(1)) \right)$$

subject to

$$\begin{aligned} 0 &= \partial_t I + v \cdot \nabla I & I(0) &= I_0 \\ 0 &= \partial_t m + \mathrm{ad}_v^\dagger m & m(0) &= L^{-1} v_0 \\ 0 &= m - Lv. \end{aligned}$$

The second constraint is the EPDiff equation and the third constraint is the definition of the momentum. Together with $v = Km$, where $K := L^{-1}$, the EPDiff describes the evolution of the velocity along the geodesic path. Figure 2.1 shows an example for the evolution of the velocity field along a geodesic.

Because $\|v\,(\cdot\,,t)\|_V$ is constant along a geodesic [54], the minimization functional can be simplified into

$$v_0^* = \underset{v_0}{\operatorname{argmin}} \left(\frac{1}{2}\beta \|v_0\|_V^2 + E_{\text{mis}}(I_1, I(1)) \right).$$

For the sake of simplicity, we will use a L^2-mismatch term $\frac{1}{2}\|I(1) - I_1\|_{L^2}^2$ throughout this work. Reformulating the expression in terms of an initial momentum $m_0 := Lv_0$ leads to a constrained minimization problem.

Problem 2.1. Geodesic Shooting for Vector-Valued Momentum

$$m_0^* = \underset{m_0}{\operatorname{argmin}}\, E(m_0) \tag{2.2}$$

with

$$E(m_0) = \frac{1}{2}\beta \langle m_0, K m_0 \rangle_{L^2} + \frac{1}{2} \| I(1) - I_1 \|_{L^2}^2 \tag{2.3}$$

subject to

$$\begin{aligned} 0 &= \partial_t I + v \cdot \nabla I & I(0) &= I_0 \\ 0 &= \partial_t m + \mathrm{ad}_v^\dagger m & m(0) &= m_0 \\ 0 &= m - Lv. \end{aligned} \tag{2.4}$$

Furthermore, it is shown in [6, Th. 2.1] that a minimizing $m(t)$ is parallel to the image gradient $\nabla I(t)$. Therefore, the minimization problem can also be stated in terms of a scalar valued momentum p [54].

Problem 2.2. Geodesic Shooting for Scalar-Valued Momentum

$$p_0^* = \underset{p_0}{\operatorname{argmin}}\, E(p_0) \tag{2.5}$$

with

$$E(p_0) = \frac{1}{2}\beta \langle p_0 \nabla I_0, K(p_0 \nabla I_0) \rangle_{L^2} + \frac{1}{2} \| I(1) - I_1 \|_{L^2}^2 \tag{2.6}$$

subject to

$$\begin{aligned} 0 &= \partial_t I + v \cdot \nabla I & I(0) &= I_0 \\ 0 &= \partial_t p + \nabla \cdot (pv) & p(0) &= p_0 \\ 0 &= Lv - p\nabla I. \end{aligned} \tag{2.7}$$

3 Optimization

In the previous chapter we derived the minimization problem. As a closed form solution to the problem is hard to find, we use iterative methods for minimization.

These methods depend on the gradient and the Hessian of the minimization problem with respect to the initial momentum. After giving a brief summary of different optimization methods, the following sections derive expressions for the gradient and the Hessian for both, a scalar and a vector valued initial momentum.

3.1 Iterative Optimization Methods

In the following sections we will see that the minimization problems 2.1 and 2.2 can be rewritten as an unconstrained minimization problem. Therefore, we briefly discuss different optimization methods for unconstrained optimization and how we use them in the optimization algorithm. For a more detailed explanation we refer to [44].

In the following we assume a minimization problem for $f(x)$ with a sufficiently smooth function $f : \mathbb{R}^n \to \mathbb{R}$.

3.1.1 Gradient Descent

Gradient descent is a simple first order method, which only depends on the gradient of the minimization problem. In each iteration the current iterate x_i is updated into the direction of the negative gradient:

$$x_{i+1} = x_i - \alpha_i \nabla f(x_i),$$

with a step size $\alpha_i > 0$ [44]. Even though the negative gradient is a descend direction, the step length is generally not known. Therefore, an additional line search is performed, which searches for a good α_i using additional function (and gradient) evaluations. As a simple line search that ensures that each iteration reduces the minimization functional f, we use *Armijo backtracking* [44, Ch. 3].

3.1.2 Newton's Method

Newton's method is a second order method, which also requires the Hessian matrix H. Based on a second order approximation, the Newton update reads [44]

$$x_{i+1} = x_i - \alpha_i H^{-1} \nabla f(x_i).$$

In each step a linear system has to be solved. We use a preconditioned conjugate gradient method method (PCG) as discussed in section 4.3. Care has to be taken, as the direction $-H^{-1}\nabla f(x_i)$ is not guaranteed to be a descend direction.

When the current x_i is still far from the solution, it is often not necessary to solve the linear system to a high accuracy. This leads to *inexact Newton Methods*, where the tolerance for the residual norm decreases during the optimization process [44, Ch. 11].

In our experiments, we use an inexact Newton method with a *quadratic forcing sequence* [18]

$$tol_i \leq \min\left(\frac{1}{2}, \frac{\|\nabla f(x_i)\|_2}{\|\nabla f(x_0)\|_2}\right)$$

for the tolerance of the residual norm tol_i in the linear solver.

To avoid problems with negative definite (negative curvature) or indefinite Hessians, we use an inexact Gauss-Newton method in most of our experiments.

3.1.3 Gauss-Newton

The Gauss-Newton method is a quasi-Newton method for minimization functionals of the form $f(m_0) = 1/2 \sum_i r_i^2(m_0)$ [44]. The Hessian of such a functional reads

$$H_f = \sum_i \left(\nabla r_i \cdot (\nabla r_i)^{\mathrm{T}} + r_i H_{r_i}\right),$$

where H_{r_1} is the Hessian matrix of r_i. The Gauss-Newton approximation drops the second term containing the second order derivatives. In contrast to Newtons method, it can be shown that under certain constraints, the Gauss-Newton update is a descend direction [44].

To drop the second Hessian term in our formulation, we drop all term that multiply with the residual r_i. As $\zeta(1) = I_1 - I(1)$ is the residual, this corresponds to setting $\zeta = 0$ in the incremental adjoint equations which we derive in section 3.4. This causes the Hessian approximation to be positive semi-definite.

In our experiments we use an inexact Gauss-Newton method, where the tolerance for the residual norm tol_i follows a *super-linear forcing sequence*

$$tol_i \leq \min\left(\frac{1}{2}, \sqrt{\frac{\|\nabla f(x_i)\|_2}{\|\nabla f(x_0)\|_2}}\right).$$

3.1.4 BFGS and L-BFGS

BFGS and the memory limited version L-BFGS [43] quasi-Newton methods and perform a Newton update using approximations to H or H^{-1}. Therefore, better convergence than gradient descent can be expected [44]. Compared to Newton's method, the approximate Hessian is much cheaper to compute, as it only needs gradient evaluations from previous iterations.

We use the *two-loop algorithm* [43, 44, Alg. 7.4] to approximate H^{-1} together with Armijo backtracking as a line search method. To ensure the positive definiteness of the approximation we skip updates that do not satisfy the curvature condition. During our experiments, no updates had to be skipped.

3.2 Optimize-then-Discretize

To solve the optimization problem we follow a *optimize-then-discretize* approach. This means that we use the continuous formulation of the optimization functional and first derive the corresponding gradient and Hessian in a continuous setting. We then discretize each term separately. This is in contrast to an *discretize-then-optimize* approach where the optimization functional is discretized first, and the gradient and Hessian are derived from the discretized minimization functional. A more detailed discussion can be found in [21, 26]. An optimize-then-discretize implementations of LDDMM can be found in [32]. A discretize-then-optimize implementations is described in [38].

3.3 Gradient of the Optimization Functional

The constrained minimization problem for the geodesic shooting formulation (2.2) and (2.5) represents a *optimal control problem* [26]. In such a setting one searches for an optimal control variable that minimizes a functional under the constraints of dynamical systems – the so called *state equations*. In the geodesic shooting problem the state equations define how the diffeomorphism evolves over time (by giving an evolution of m or p and therefore v) and how the diffeomorphism acts on the initial image (by stating the evolution of I).

The PDE constraints can be solved forward in time, which allows us to evaluate the objective functional for different initial momentums. Therefore, the most simple way to compute a gradient of the optimization problem is to use finite differences by evaluating the functional for different initial momentums. However, this implies that the constraints have to be solved very often if the initial momentum contains a large number of unknowns. This is usually the case for image data. A computationally cheaper approach is the so called *adjoint method* [9, 26, Ch. 1.6]. It often allows to compute a gradient at much lower cost, independent of the number of unknowns.

For the adjoint method we first turn the PDE constrained problem into an unconstrained problem, using the method of Lagrangian multipliers [26, Ch. 1.6]. Based on the resulting Lagrangian we can compute the gradient with respect to the initial momentum. For this, we first have to solve the so-called *adjoint equation*. This is often of similar cost to solving the state equations. The adjoint equations usually depend on the solution of the state equation. Each gradient evaluation includes solving the state and adjoint equations. For more details we refer to [26].

The resulting gradient can then be used in optimization methods like gradient descent.

3.3.1 Vector Valued Momentum

We now derive the gradient for the vector valued momentum following the previous outline. Based on the constrained minimization problem (2.2) the Lagrangian $\mathcal{L}$ reads

$$\begin{aligned}\mathcal{L}(m_0,I,m,v,\zeta,\mu,\nu) := {} & \frac{1}{2}\beta\left\langle m_0, Km_0\right\rangle_{L^2} + \frac{1}{2}\left\|I(1)-I_1\right\|_{L^2}^2 \\ & + \int_0^1 \left\langle \zeta, \partial_t I + v\cdot\nabla I\right\rangle_{L^2} \mathrm{d}t + \left\langle \zeta(0), I(0)-I_0\right\rangle_{L^2} \\ & + \int_0^1 \left\langle \mu, \partial_t m + \mathrm{ad}_v^\dagger m\right\rangle_{L^2} \mathrm{d}t + \left\langle \mu(0), m(0)-m_0\right\rangle_{L^2} \\ & + \int_0^1 \left\langle \nu, m - Lv\right\rangle_{L^2} \mathrm{d}t,\end{aligned} \tag{3.1}$$

where ζ, μ and ν are the Lagrangian multipliers for the state variables I, m and v, respectively. In this setting the Lagrangian multipliers are also called *dual* or *adjoint variables*.

The next step is to compute the gradient of the Lagrangian, because a minimizer of the original problem is also minimizes the Lagrangian. The gradient can be found by computing directional derivatives for each control, state and adjoint variable. Taking the derivative of the Lagrangian with respect to m_0 in the direction $\tilde{m}_0$ gives

$$\partial_{m_0}\mathcal{L}[\tilde{m}_0] = \left\langle \beta K m_0 - \mu(0), \tilde{m}_0\right\rangle_{L^2},$$

where we can read off the gradient

$$\nabla_{m_0}\mathcal{L} = \beta K m_0 - \mu(0). \tag{3.2}$$

For a given m_0, Km_0 can be easily computed. The expensive part is finding μ at $t=0$. To evaluate $\mu(0)$ we have to solve the adjoint equation. The variations with respecto to the state variables are

$$\begin{aligned}
\partial_I \mathcal{L}[\tilde{I}] &= \int_0^1 \langle -\partial_t \zeta - \nabla \cdot (\zeta v), \tilde{v} \rangle_{L^2} \,\mathrm{d}t + \left\langle \zeta(1) + I(1) - I_1, \tilde{I}(1) \right\rangle_{L^2} \\
\partial_m \mathcal{L}[\tilde{m}] &= \int_0^1 \langle -\partial_t \mu + \mathrm{ad}_v\, \mu + \nu, \tilde{v} \rangle_{L^2} \,\mathrm{d}t + \langle \mu(1), \tilde{m}(1) \rangle_{L^2} \\
\partial_v \mathcal{L}[\tilde{v}] &= \int_0^1 \left\langle -L\nu - \mathrm{ad}_\mu^\dagger m + \zeta \nabla I, \tilde{v} \right\rangle_{L^2} \mathrm{d}t
\end{aligned}$$

This leads to the adjoint equations in strong form

$$\begin{aligned}
0 &= -\partial_t \zeta - \nabla \cdot (\zeta v) \\
0 &= -\partial_t \mu + \mathrm{ad}_v\, \mu + \nu \\
0 &= -L\nu - \mathrm{ad}_\mu^\dagger m + \zeta \nabla I
\end{aligned} \tag{3.3}$$

with final conditions

$$\begin{aligned}
0 &= \zeta(1) + I(1) - I_1 \\
0 &= \mu(1).
\end{aligned}$$

The adjoint equations depend on I, m and v, which are obtained by solving the state equations for the current m_0.

Evaluating the gradient of the minimization functional consists of three steps:

1. The state equations (2.4) have to be solved forward in time.
2. The adjoint equations (3.3) have to be solved backward in time.
3. Given μ at $t = 0$, we can evaluate the gradient (3.2).

3.3.2 Scalar Valued Momentum

The gradient for the scalar valued momentum formulation can be derived similar to the vector valued momentum: Based on the optimization problem (2.5) we define the Lagrangian as

$$\begin{aligned}
\mathcal{L}(p_0, I, p, v, \zeta, \varrho, \nu) := \frac{1}{2}\beta \langle p_0 \nabla I_0, K(p_0 \nabla I_0) \rangle_{L^2} + \frac{1}{2} \| I(1) - I_1 \|_{L^2}^2 \\
+ \int_0^1 \langle \zeta, \partial_t I + v \cdot \nabla I \rangle_{L^2} \,\mathrm{d}t + \langle \zeta, I(0) - I_0 \rangle_{L^2} \\
+ \int_0^1 \langle \varrho, \partial_t p + \nabla \cdot (pv) \rangle_{L^2} \,\mathrm{d}t + \langle \varrho, p(0) - p_0 \rangle_{L^2} \\
+ \int_0^1 \langle \nu, Lv - p \nabla I \rangle_{L^2} \,\mathrm{d}t,
\end{aligned} \tag{3.4}$$

where ζ, ϱ and ν are the Lagrangian multipliers for I, p and v. As before, we refer to the Lagrangian multipliers as adjoint variables in this setting. Computing the first variation of the Lagrangian gives the adjoint equations

$$\begin{aligned}
0 &= -\partial_t \zeta + \nabla \cdot (p\nu - v\zeta) \\
0 &= -\partial_t \varrho - v \cdot \nabla \varrho - \nu \cdot \nabla I \\
0 &= L\nu - (p\nabla \varrho - \zeta \nabla I)
\end{aligned}$$

with the final conditions

$$\begin{aligned}
0 &= \zeta(1) + I(1) - I_1 \\
0 &= \varrho(1).
\end{aligned}$$

The equation for the gradient with respect to the initial momentum reads

$$\nabla_{p_0} \mathcal{L} = \beta \nabla I_0 \cdot K(p_0 \nabla I_0) - \varrho(0). \tag{3.5}$$

3.4 Hessian of the Optimization Functional

For higher order optimization methods such as Newton's method we need knowledge of the Hessian of the optimization problem. Building a full Hessian or its inverse is generally not feasible. If we use an iterative solver for the linear system that appears in Newton's method, all we require is an expression for the matrix-vector-products with the Hessian is sufficient.

In the following we derive this expression for the matrix-vector-products. This expression can then be used in combination with a Krylov subspace method [47]. We arrive at a so called *Newton-Krylov* optimization algorithm [28] for the geodesic shooting problem. In general, we assume sufficient smoothness of the optimization problem, so that the Hessian exists and is symmetric.

3.4.1 Vector Valued Momentum

Similar to the procedure for the gradient evaluation in section 3.3 we start with the Lagrangian

$$\begin{aligned}
\mathcal{L} = & \langle \beta K m_0 - \mu(0), \tilde{\mu}(0)\rangle_{L^2} \\
& + \int_0^1 \left\langle \partial_t I + v\cdot\nabla I, \tilde{\zeta}\right\rangle_{L^2} \mathrm{d}t + \left\langle I(0) - I_0, \tilde{\zeta}(0)\right\rangle_{L^2} \\
& + \int_0^1 \left\langle \partial_t m + \mathrm{ad}_v^\dagger m, \tilde{\mu}\right\rangle_{L^2} \mathrm{d}t + \langle m(0) - m_0, \tilde{\mu}(0)\rangle_{L^2} \\
& + \int_0^1 \langle m - Lv, \tilde{\nu}\rangle_{L^2} \mathrm{d}t \\
& + \int_0^1 \langle -\partial_t \mu + \mathrm{ad}_v \mu + \nu, \tilde{m}\rangle_{L^2} \mathrm{d}t + \langle \mu(1), \tilde{m}(0)\rangle_{L^2} \\
& + \int_0^1 \left\langle -\partial_t\zeta - \nabla\cdot(\zeta v), \tilde{I}\right\rangle_{L^2} \mathrm{d}t + \left\langle \zeta(1) + I(1) - I_1, \tilde{I}(1)\right\rangle_{L^2} \\
& + \int_0^1 \left\langle -L\nu - \mathrm{ad}_\mu^\dagger m + \zeta\nabla I, \tilde{v}\right\rangle_{L^2} \mathrm{d}t.
\end{aligned}$$

It contains the gradient, the state equations, the adjoint equations and the corresponding boundary conditions in their weak form. Computing the directional derivatives with respect to I, m, v, ζ, μ and ν gives a new set of equations. Based on the initial and final conditions these equations can be split in the *incremental state equations*

$$\begin{aligned}
0 &= \partial_t \tilde{I} + v\cdot\nabla\tilde{I} + \tilde{v}\cdot\nabla I & 0 &= \tilde{I}(0) \\
0 &= \partial_t \tilde{m} + \mathrm{ad}_v^\dagger \tilde{m} + \mathrm{ad}_{\tilde{v}}^\dagger m & 0 &= \tilde{m}(0) - \tilde{m}_0 \\
0 &= \tilde{m} - L\tilde{v}
\end{aligned}$$

and the *incremental adjoint equations*

$$\begin{aligned}
0 &= -\partial_t\tilde{\zeta} - \nabla\cdot\left(\tilde{\zeta} v + \zeta\tilde{v}\right) & 0 &= \tilde{\zeta}(1) + \tilde{I}(1) \\
0 &= -\partial\tilde{\mu} + \mathrm{ad}_v\tilde{\mu} + \mathrm{ad}_{\tilde{v}}\mu + \tilde{\nu} & 0 &= \tilde{\mu}(1) \\
0 &= -L\tilde{\nu} + \zeta\nabla\tilde{I} + \tilde{\zeta}\nabla I - \mathrm{ad}_\mu^\dagger\tilde{m} - \mathrm{ad}_{\tilde{\mu}}^\dagger m.
\end{aligned}$$

The matrix-vector-product of $\tilde{m}_0$ with the Hessian H then reads

$$H\tilde{m}_0 = K\tilde{m}_0 - \tilde{\mu}(0).$$

The Hessian matrix-vector-product consists of a regularization term $K\tilde{m}_0$ and a mismatch term $-\tilde{\mu}(0)$.

Every matrix-vector-product requires the solution of the incremental state and incremental adjoint equation to evaluate the mismatch term. This is similar to the evaluation of the gradient of the minimization functional. The solution of the incremental equations depends on the solution of the state and adjoint equation.

3.4.2 Scalar Valued Momentum

The derivation of the incremental equations for the scalar valued momentum is analogues to the vector valued case: The Lagrangian

$$\begin{aligned}
L = & \langle \beta \nabla I_0 \cdot K(p_0 \nabla I_0) - \varrho(0), \tilde{p}(0) \rangle_{L^2} \\
& + \int_0^1 \left\langle I \partial_t I + v \cdot \nabla I, \tilde{\zeta} \right\rangle_{L^2} \mathrm{d}t + \left\langle I(0) - I_0, \tilde{\zeta}(0) \right\rangle_{L^2} \\
& + \int_0^1 \langle \partial_t p + \nabla \cdot (pv), \tilde{\varrho} \rangle_{L^2} \mathrm{d}t + \langle p(0) - p_0, \tilde{\varrho}(0) \rangle_{L^2} \\
& + \int_0^1 \langle Lv - p \nabla I, \tilde{\nu} \rangle_{L^2} \mathrm{d}t \\
& + \int_0^1 \left\langle -\partial_t \zeta + \nabla \cdot (p\nu - v\zeta), \tilde{I} \right\rangle_{L^2} \mathrm{d}t + \left\langle \zeta(1) + I(1) - I_1, \tilde{I}(1) \right\rangle_{L^2} \\
& + \int_0^1 \langle -\partial_t \varrho - v \cdot \nabla \varrho - \nu \cdot \nabla I, \tilde{p} \rangle_{L^2} \mathrm{d}t + \langle \varrho(1), \tilde{p}(1) \rangle_{L^2} \\
& + \int_0^1 \langle L\nu - (p \nabla \varrho - \zeta \nabla I), \tilde{v} \rangle_{L^2} \mathrm{d}t
\end{aligned}$$

leads to the incremental state equation

$$\begin{aligned}
0 &= \partial_t \tilde{I} + v \cdot \nabla \tilde{I} + \tilde{v} \cdot \nabla I & 0 &= \tilde{I}(0) \\
0 &= \partial_t \tilde{p} + \nabla \cdot (v\tilde{p} + \tilde{v}p) & 0 &= \tilde{p}(0) - \tilde{p}_0 \\
0 &= L\tilde{v} - \left(p \nabla \tilde{I} + \tilde{p} \nabla I \right)
\end{aligned}$$

and the incremental adjoint equation

$$\begin{aligned}
0 &= -\partial_t \tilde{\zeta} - \nabla \cdot \left(v\tilde{\zeta} + \tilde{v}\zeta - p\tilde{\nu} - \tilde{p}\nu \right) & 0 &= \tilde{\zeta}(1) + \tilde{I}(1) \\
0 &= -\partial_t \tilde{\varrho} - v \cdot \nabla \tilde{\varrho} - \tilde{v} \cdot \nabla \varrho - \nabla I \cdot \tilde{\nu} - \nabla \tilde{I} \cdot \nu & 0 &= \tilde{\varrho}(1) \\
0 &= L\tilde{\nu} - (p \nabla \tilde{\varrho} + \tilde{p} \nabla \varrho - \zeta \nabla \tilde{I} - \tilde{\zeta} \nabla I).
\end{aligned}$$

Again, the resulting Hessian matrix-vector-product given by

$$H\tilde{p}_0 = \beta \nabla I_0 \cdot K(\tilde{p}_0 \nabla I_0) - \tilde{\varrho}(0)$$

consists of a regularization term $\beta \nabla I_0 \cdot K(\tilde{p}_0 \nabla I_0)$ and a mismatch term $-\tilde{\varrho}(0)$.

3.5 Summary

Given an initial momentum, the objective functional $E(m_0)$ or $E(p_0)$ can be evaluated by solving the forward problem consisting of the state equations. This is done by integrating the state equations forward in time. To evaluate the gradient of the objective functional we use the adjoint method, which requires analytical expressions for the gradient. To evaluate the gradient we need to solve the adjoint equations which have to be solved backward in time. If we consider second order (Newton-type) methods for optimization we have to solve a linear system. For this, we use a reduced space matrix-free method. Each applications of the Hessian to a vector requires the solution of the incremental state and incremental adjoint equations. The incremental state equations have to be solved forward in time and the the incremental adjoint equations have to be solved backward in time.

With functional evaluations, the gradient and Hessian matrix-vector-products (and algorithms to approximately compute its inverse) we have all the necessary ingredients in order to run an optimization method to find an approximate solution to the geodesic shooting problem. By now, all these expressions are still in a continuous setting. Following the optimize-then-discretize approach, the next chapter will discuss the discretization of the individual expressions.

4 Discretization

Following the optimize-then-discretize approach, we now discuss the discretized expressions for the objective functional, it's gradient and Hessian. We discuss the spatial and temporal discretization of the state, adjoint and incremental equations, which have to be solved in order to obtain the derivatives of the optimization problem. For the evolution of the PDEs we either use Heun's method or a semi-Lagrangian method.

4.1 Spatial Discretization

In space the problem is discretized using an equidistant grid defined on $\Omega = [-\pi, \pi]^2$ with periodic boundary conditions, and the same number of nodes N_x along each spatial dimension. We denote the resulting mesh width as h_x.

All spatial differential operators are discretized using a pseudo-spectral approach [8]. They are computed in the Fourier domain based on the fact that derivatives can there be computed efficiently and accurately by point-wise multiplications. This allows a straightforward application of inverse differential operators. For example, applying the inverse regularization operator $(\mathrm{I} - \Delta)^{-\gamma}$ to a function f becomes a pointwise multiplication in the Fourier domain

$$(\mathrm{I} - \Delta)^{-\gamma} f = \mathcal{F}^{-1}\mathcal{F}[(\mathrm{I} - \Delta)^{-\gamma} f(\cdot)] = \mathcal{F}^{-1}\left[(1 + \|\cdot\|_2^2)^{-\gamma}\mathcal{F}[f](\cdot)\right],$$

even for a general $\gamma \in \mathbb{R}$. The mapping to and from the Fourier domain in the discrete setting is done using the Fast-Fourier-Transform (FFT) [15] and its inverse, respectively. This scheme has a high accuracy and in general displays little numerical diffusion if combined with adequate methods for numerical time integration [34].

4.1.1 Discretization of the Optimization Functional

Up to now we avoided discussing the choice of a discrete vector space. For the optimization functional we have to choose a discrete scalar product that corresponds to the L^2-scalar product in the continuous setting. Instead of using the usual Euclidean scalar product in $\mathbb{R}^d$, we use a *scaled* Euclidean scalar product

$$\langle \cdot\,, \cdot \rangle_h := h \langle \cdot\,, \cdot \rangle_2$$

with its induced norm

$$\|x\|_h := \sqrt{\langle x, x\rangle_h}.$$

Here, h is the mesh width of the spatial discretization. This norm is an approximation to the L^2-scalar product using a trapezoidal rule. Hence, using this norm makes error measurements independent of the discretization and makes it easier to compare results that use different spatial resolutions.

Using the scaled Euclidean norm, the discretized optimization functional $E_h : \mathbb{R}^{N_x^d} \to \mathbb{R}$ reads

$$E_h(m_0) = \frac{1}{2}\beta \|Km_0\|_h + \frac{1}{2}\|I(1) - I_1\|_h$$

for vector-valued momentum and

$$E_h(p_0) = \frac{1}{2}\beta \langle p_0 \nabla I_0, K(p_0 \nabla I_0)\rangle_h + \frac{1}{2}\|I(1) - I_1\|_h$$

for the scalar-valued momentum, respectively.

The choice of the discrete vector space also has an effect on the scaling of the gradient of the objective function. Using a Taylor expansion for a function f

$$\begin{aligned} f(x+d) &= f(x) + \left\langle \nabla^{(h)} f(x), d \right\rangle_h + \mathcal{O}\left(\|d\|_h^2\right) \\ &= f(x) + \left\langle \nabla^{(2)} f(x), d \right\rangle_2 + \mathcal{O}\left(\|d\|_h^2\right) \end{aligned}$$

shows that the gradient with respect to the h-scalar product $\nabla^{(h)} f(x)$ only differs by a factor of h from the gradient with respect to the 2-scalar product $\nabla^{(2)} f(x)$. That is $h\nabla^{(h)} f(x) = \nabla^{(2)} f(x)$. An analogous argument holds for the Hessian.

4.1.2 Grid Transfer Operations – Prolongation and Restriction

To use coarse grid techniques we define restriction and prolongation operators in the Fourier domain by truncating coefficients corresponding to high frequencies or padding the frequency spectrum with zeros, respectively. Using the FFT followed by the inverse FFT for a truncated spectrum adds an additional scaling factor. This scaling needs to be accounted for in a coarse grid approximation. The inverse scaling factor has to be used for the prolongation.

4.2 Temporal Discretization

The artificial time we introduced is discretized into time steps of size h_t. For the time evolution of the PDEs we either use Heun's method [11, Ch. 232] or a backward semi-Lagrangian method [50, 27, Ch. 3.3.3].

4.2.1 Heun's Method

Heun's method is an explicit, second order Runge-Kutta method [11] with a single intermediate step. Given a differential equation $\partial_t y(x,t) = f(y(x,t),t)$, Heun's method approximates the time integral for each time step by a trapezoidal rule

$$\begin{aligned}\int_{t_i}^{t_{i+1}} f(y(x,t),t)\,\mathrm{d}t &\approx \frac{h_t}{2}\left(f(y(x,t_i),t_i) + f(y(x,t_{i+1}),t_{i+1})\right) \\ &\approx \frac{h_t}{2}\left(f(y(x,t_i),t_i) + f(y(x,t_i) + h_t f(y(x,t_i),t_i),t_{i+1})\right),\end{aligned}$$

where the function evaluations at the new time step are approximated using an explicit Euler step. Hence, using $y_i := y(\,\cdot\,,t_i)$ and $f_i := f(\,\cdot\,,t_i)$, Heun's Method for a time step yields

$$y_{i+1} = y_i + \frac{h_t}{2}\left(f_i(y_i) + f_{i+1}(y_i + h_t f_i(y_i))\right).$$

The combination of Heun's method in time and a spectral discretization in space results in a small numerical diffusion [32]. Another advantage is the absence of intermediate function evaluations between the time steps. Therefore, we avoid temporal interpolations between time steps in the adjoint and incremental equations. These PDEs depend on the earlier solution of the other equations and only use function evaluations at full time steps, which are already known.

The size of the time steps is limited by the Courant–Friedrichs–Lewy (CFL) condition [16].

4.2.2 Semi-Lagrangian Method

Heun's method uses Eulerian coordinates, which means it describes how a value at a fixed point in space changes over time. In contrast, the semi-Lagrangian method uses Lagrangian coordinates, which follow the trajectory of particles defined by a given velocity field [50, 31, 2.13]. This formulation describes how a value associated with such a particle changes over time. As the semi-Lagrangian method is unconditionally stable for transport equations [50, 27] and therefore not constrained by the CFL condition, it allows to use much larger time steps compared to Heun's method. To simplify the notation we use subscripts $f_i(\,\cdot\,) = f(\,\cdot\,,t_i)$ to specify values at certain time steps.

To measure the change of a value along a trajectory one considers the *Lagrangian derivative*

$$\mathrm{d}_t^{(v)} := \partial_t + v \cdot \nabla$$

based on a velocity field v.

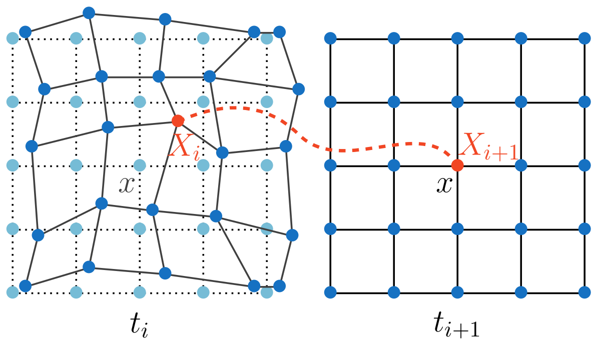

Figure 4.1: Illustration of the semi-Lagrangian method for a velocity field that transports towards the lower left corner. Step one: All particles start on a regular grid at t_{i+1} (on the right) with $X(x,t_{i+1}) = x$. They follow the characteristic backwards in time to t_i yielding a deformed grid (dark blue on the left). Step two: By interpolating the values at the previous time step t_i (light blue on the left) in in $X(x,t_i)$, we assign a value to each particle. Integrating the right hand side along the characteristic forward in time accounts for changes to these values between the time steps. The result are the values on a regular grid at the new time step.

Given a differential equation in Lagrangian coordinates X

$$\mathrm{d}_t^{(v)} y(X,t) = f(y(X,t),t),$$

the semi-Lagrangian method works in two stages. These are illustrated in figure 4.1. In a first step, we follow each particle backward in time to find out where it originates from. For this, a transport equation has to be solved so that the trajectories of the particles are known. In the second step, we compute how much the value associated with a particle changed. Therefore, the right hand side of the equation has to be integrated in time to compute the change of the transported quantity along the trajectory:

1. First, we have to compute the trajectory $X(x,t)$ of the particles. We do this backward in time. We initialize the trajectories at the new time step using the nodes of a regular grid. Using the velocity field, we solve the transport equation

$$\begin{aligned} \partial_t X(x,t) &= -v(X(x,t),t) \quad \text{for } t \in [t_i, t_{i+1}] \\ X(x,1) &= x \end{aligned}$$

   for X, which describes the trajectory based on the final position of each particle. A particle in position x at t_{i+1} originates from $X(x,t_i)$. We use one step of Heun's method to compute the trajectory.

2. Once the trajectory is known, we integrate the right hand side of the equation along the trajectory. Again, we use a single step of Heun's method . Therefore, f has to

be evaluated at the starting point and endpoint of each trajectory. Evaluating f at the new time step is straight forward, as we ensured that each trajectory ends on a grid point. However, the starting points of the trajectory are generally between grid points and have to interpolated. Because linear interpolation can lead to high numeric diffusion, we use cubic splines to interpolate the values, which is less diffusive [50].

The steps above work well when the velocity v is known, which is the case for the adjoint and both incremental equations. However, while integrating the state equation, the velocity at the next time step is still unknown. Therefore, we use a second order extrapolation

$$v_{i+1}(x) = 2v_i(x) - v_{i-1}(x) + \mathcal{O}\left(h_t^2\right)$$

to approximate v_{i+1} in the trajectory computation [50].

When the semi-Lagrangian method is applied, we have to keep in mind that the state equations are solved forward in time, and the adjoint equations are solved backward in time. This has an effect on the trajectory computation and the integration of the right hand side along the trajectory:

Heun's method to compute the trajectory for the state and incremental state equation backward in time reads

$$X_i = x - \frac{h_t}{2}\left(-v_{i+1}(x) - v_i\left(x + h_t v_{i+1}(x)\right)\right),$$

where all evaluations of v_i have to be interpolated. As the adjoint and incremental adjoint equations are solved backward in time, the trajectory is computed by Heun's method forward in time

$$X_{i+1} = x + \frac{h_t}{2}\left(-v_{i+1}(x) - v_i\left(x - h_t v_{i+1}(x)\right)\right).$$

Here, evaluations of v_{i+1} have to be interpolated, while the evaluations of v_i are aligned to grid nodes.

As the trajectories only depend on the velocity field they only have to be recomputed once the velocity or the momentum changes.

During the application of Heun's method to integrate the remaining right hand side of the equations, values at one time step have to be interpolated using the result from the trajectory. For the state and incremental state equations, which are solved forward in time, values at t_i have to be interpolated based on the starting point of the trajectory $X_i(x)$:

$$y_{i+1}(x) = y_i(X_i(x)) + \frac{h_t}{2}\left(f_i(X_i(x)) + f_{i+1}(x)\right)$$

For the adjoint and incremental adjoint equations, which are solved backward in time, values at t_{i+1} have to be interpolated based on $X_{i+1}(x)$.

4.3 Krylov Subspace Solver

For Newton's method we have to solve the linear Hessian system

$$Hx = -\nabla_{m_0} E \quad \text{or} \quad Hx = -\nabla_{p_0} E$$

for an update x. While the incremental equations allow to compute matrix-vector multiplications with the Hessian, individual entries of the Hessian are not directly known. Therefore, we use Krylov subspace methods like the (preconditioned) Conjungate Grandient method (PCG) or the Generalized Minimal Residual method (GMRES) to solve the system using only Hessian-vector multiplication. This leads to a so-called *Newton-Krylov method* for the minimization problem [28]. To reduce the number of iterations in each Newton step, we will design preconditioners. This improves the convergence of the linear solver. In chapter 5 we discuss different preconditioners and analyze their effectiveness in chapter 6.

4.4 Summary

Spatial operators are computed in the Fourier domain which allows to compute inverse differential operators and has a high accuracy. We prefer explicit over implicit time integrators as they are cheaper to compute and do not introduce another implicitly given problem. To reduce the cost of the time integration we use the semi-Lagrangian method which allows to increase the time step size.

5 Preconditioners

The most expensive component in each optimization step is the integration of the PDEs. Especially the repeated solution of the incremental equations to evaluate Hessian matrix-vector-products in the Krylov subspace solver for Newton-type optimization can take a significant amount of time.

The central idea of preconditioning is that the rate of convergence (and thus the required number of iterations) usually depends on the condition number of the matrix of the linear system [47]. By preconditioning the linear system we can improve the condition number of the matrix and in turn reduce the number of iterations. By that we reduce the number of matrix-vector-products with the Hessian.

The main challenge for the presented geodesic shooting problem is its matrix-free setting. While matrix-vector-products with the Hessian can be evaluated using the incremental equations, the individual matrix entries remain unknown. This makes most standard preconditioners such as Jacobi, Gauss-Seidl and incomplete LU preconditioners [47] difficult to apply. Even though it is possible to compute individual columns of the Hessian matrix by evaluating the matrix-vector-product with canonical unit vectors, building the entire Hessian matrix is generally unfeasible for large image dimensions, due to both, memory requirements and computational costs.

Instead, we will follow different approaches, which only use Hessian matrix-vector-products and the structure of the Hessian

$$H = \beta H_{\text{reg}} + H_{\text{mis}} \in \mathbb{R}^{N_x^d \times N_x^d},$$

which consists of a regularization term βH_{reg} and a mismatch term H_{mis}. The additive structure of the Hessian was mentioned earlier in section 3.4.

5.1 Inverse Regularization Term

The regularization term of the Hessian for the vector valued momentum $H_{\text{reg}} = K$ can be inverted analytically as $H_{\text{reg}}^{-1} = L$. The inverse can be used as a preconditioner for the CG method in each Newton step. Instead of solving the linear system

$$(\beta K + H_{\text{mis}})x = -\nabla_{m_0} E$$

for the Newton update x using the CG method, we use the preconditioned CG [25, 47, 44] method with $\beta^{-1}L$ as preconditioner. This is equivalent to solving the system

$$\begin{aligned}(\mathrm{I}+\frac{1}{\beta}L^{1/2}H_{\text{mis}}L^{1/2})y &= -\frac{1}{\sqrt{\beta}}L^{1/2}\nabla_{m_0}E \\ x &= \frac{1}{\sqrt{\beta}}L^{1/2}y\end{aligned} \tag{5.1}$$

which is left and right preconditioned with $(\beta H_{\text{reg}})^{-1/2} = \beta^{-1/2}L^{1/2}$. The expression $L^{1/2}$ exists for our choice of L as it is a symmetric positive definite operator and can be computed in Fourier space as mentioned in section 4.1.

5.2 Coarse Grid Preconditioner

The motivation for the coarse grid preconditioner from [34] is, that inverting the Hessian for a coarser discretization is significantly cheaper than inverting the Hessian on a finer discretization. The idea is to split the momentum into low and high frequency Fourier modes. We assume that the Hessian in the Fourier basis

$$\hat{H} = \begin{pmatrix}\hat{H}_f & \hat{H}_x \\ \hat{H}_x & \hat{H}_c\end{pmatrix}$$

mainly consists of two blocks $\hat{H}_f$ and $\hat{H}_c$, which map high to high and low to low frequencies and negligible blocks $\hat{H}_x$, which mix the frequency bands. We can then represent the low frequent Fourier modes on a coarser grid and solve for them. Hence we use the preconditioner

$$\hat{M}_c^{-1} := \begin{pmatrix}\mathrm{I} & \\ & \hat{H}_c^{-1}\end{pmatrix}$$

in the Fourier basis to improve the condition number by inverting the high frequency components of the Hessian. Using the restriction operator R, the prolongation P and the low-pass filter F the preconditioner then looks like

$$M_c^{-1} := (\mathrm{I}-F) + FPH_c^{-1}RF.$$

In our experiments we split the Fourier spectrum in half and use the operators

$$R = \begin{pmatrix}0 & \mathrm{I}\end{pmatrix}, \quad P = \begin{pmatrix}0 \\ \mathrm{I}\end{pmatrix} \quad \text{and} \quad F = \begin{pmatrix}0 & \\ & \mathrm{I}\end{pmatrix}.$$

As before, in each Newton step, the “outer” linear system with H is solved for a Newton update with a Krylov subspace method.

To evaluate matrix-vector-products with the preconditioner H_c^{-1} a second "inner" linear solver is used as in [34]. For this "inner" solver we use a PCG method with a stricter tolerance. We apply this preconditioner not to the original Hessian, but to the L-preconditioned Hessian.

5.3 Low Rank Approximation

This method is based on the preconditioned Hessian from (5.1)

$$\tilde{H} := \mathrm{I} + \frac{1}{\beta} \underbrace{L^{1/2} H_{\text{mis}} L^{1/2}}_{=:\tilde{H}_{\text{mis}}},$$

which results from applying the regularization operator as a preconditioner. A common observation is that the spectrum of the preconditioned mismatch term decays quickly [45, 55]. The general idea is to replace the mismatch term with a low rank approximation and invert the result analytically.

A similar approach has been taken in [55, 10, 45] in the context of statistical inverse problems and the sampling from probability distributions. As in [45], we use a truncated eigenvalue decomposition of the mismatch term and the Woodbury matrix identity [44, A.28] to obtain an approximation to the inverse preconditioned Hessian:

We begin with an eigenvalue decomposition for the preconditioned mismatch term $\tilde{H}_{\text{mis}} := L^{1/2} H_{\text{mis}} L^{1/2}$ into a diagonal matrix Λ, consisting of the eigenvalues $\Lambda_{ii} = \lambda_i \geq \lambda_{i+1}$, and the corresponding eigenvector matrix V

$$\tilde{H}_{\text{mis}} = V \Lambda V.$$

The inverse of the preconditioned Hessian can be written as

$$\tilde{H}^{-1} = (\mathrm{I} + \frac{1}{\beta}\tilde{H}_{\text{mis}})^{-1} = (\mathrm{I} + \frac{1}{\beta} V \Lambda V^{\mathrm{T}})^{-1} = \mathrm{I} - V(\beta\Lambda^{-1} + V^{\mathrm{T}} V)^{-1} V^{\mathrm{T}}$$

using the Woodbury matrix identity. As the eigenvectors are normalized and pairwise orthogonal this can be further simplified into

$$\tilde{H}^{-1} = \mathrm{I} - V(\underbrace{\beta\Lambda^{-1} + \mathrm{I}}_{\operatorname{diag}_i(\beta\lambda_i^{-1}+1)})^{-1} V^{\mathrm{T}}) = \mathrm{I} - V \operatorname*{diag}_i \left(\frac{\lambda_i}{\lambda_i + \beta}\right) V^{\mathrm{T}}.$$

In general, computing a complete eigenvalue decomposition of $\tilde{H}_{\text{mis}}$ is too expensive. However, if we compute a truncated eigenvalue decomposition of $\tilde{H}_{\text{mis}}$ using only the p largest eigenpairs we get an approximation to $\tilde{H}^{-1}$

$$M_{\text{lr}}^{-1} := \mathrm{I} - V_p \operatorname*{diag}_{i=1,\ldots,p} \left(\frac{\lambda_i}{\lambda_i + \beta}\right) V_p^{\mathrm{T}},$$

which can be used as preconditioner. The rectangular matrix $V_p \in \mathbb{R}^{N_x^d \times p}$ contains the eigenvectors corresponding to the first p largest eigenvalues. A nice property of this preconditioner is that the decomposition can be computed independent of the regularization parameter β. Because all nonzero entries in the diagonal matrix are positive and smaller then one for $\beta > 0$, the final preconditioner is positive definite.

This preconditioner depends on a matrix-free algorithm to compute the truncated eigenvalue decomposition of $\tilde{H}_{\text{mis}}$. In section 5.5 we will present randomized algorithms, which can be used to efficiently estimate the eigenpairs.

5.4 Low Rank Approximation on Coarse Grid

The ideas of building a low rank approximation can be combined with the idea to build a preconditioner on a coarse grid. For this we assume that the eigenvectors corresponding to the largest eigenvalues are smooth and can be represented on a coarser grid. The procedure is the same as in section 5.3. The only difference is that the truncated eigenvalue decomposition is performed on a coarser grid. Because the resulting approximation is defined on a coarser grid, it has to be transfered back to the original finer grid to be used as a preconditioner. Hence, applying the approximation to a vector, first restricts the vector to the coarse grid, applies the low rank approximation and prolongs the result back to the fine grid. This is equivalent to prolonging the eigenvectors to the finer grid and renormalizing them. The resulting preconditioner reads

$$M_{\text{lr,c}}^{-1} := \mathrm{I} - P V_p \operatorname*{diag}_{i=1,\dots,p}\left(\frac{\lambda_i}{\lambda_i + \beta}\right) V_p^{\mathrm{T}} R,$$

where P is the prolongation and R is the restriction as defined in section 4.1.2.

5.5 Randomized Low Rank Approximation

The previous preconditioners rely on an efficient matrix-free method to compute an eigenvalue decomposition. For this we use a randomized eigenvalue decomposition, which provides a stable algorithm that is easy to implement [22]. For a detailed discussion of randomized decompositions, their properties and a comparison to other approaches we refer to [22]. In the following we give a short summary.

We start with a singular value estimation for a matrix $H \in \mathbb{R}^{n \times n}$, which consists of two steps:

1. We approximate the range of the matrix H. This is done by applying H to p random vectors and building a basis $Q \in \mathbb{R}^{n \times p}$ of the result. The matrix H can then be

approximated in the smaller basis as $H \approx QQ^{\mathrm{T}}H$.
Here, $Q^{\mathrm{T}}H \in \mathbb{R}^{p\times n}$ can be understood as the application of H, and the projection into the smaller basis. Multiplying the result by Q transforms back into the original basis. The result is therefore the matrix H, restricted to the subspace of a smaller basis.

2. In the approximate basis we can then use traditional decomposition algorithms for $Q^{\mathrm{T}}H$. This is possible as $Q^{\mathrm{T}}H$ is a much smaller rectangular matrix where the number of rows is the same as the dimension p of the smaller basis. With the singular value decomposition $Q^{\mathrm{T}}H =: U\Lambda V^{\mathrm{T}}$ the original matrix can be approximated as

$$H \approx QU\Lambda V^{\mathrm{T}},$$

with $U \in \mathbb{R}^{p\times p}$, $\Lambda \in \mathbb{R}^{p\times n}$ and $V \in \mathbb{R}^{n\times n}$. This is a singular value decomposition for $QQ^{\mathrm{T}}H$ [22].

Under the assumption that the matrix H is symmetric and positive definite, the first p columns of QU are eigenvectors of H with the corresponding eigenvalues on the diagonal of Λ. Remember that the Gauss-Newton approximation ensures that the Hessian is positive definite, which makes it possible to use this algorithm.

5.6 Summary

While a good preconditioner can greatly improve the convergence of the iterative solver, this improvement does not come for free. One has to weight the accuracy of the preconditioner and therefore the reduced number of iterations against the costs of building and applying the preconditioner.

As the most expensive part during the registration process are the different PDE solves, using the inverse regularization term L as preconditioner is a very cheap. As we will see in chapter 6, this allows us to improve the condition number.

The performance of the coarse grid preconditioner mainly depends on the cost of inverting the coarse Hessian. This has to be done in every iteration of the Krylov subspace method when the preconditioner is applied.

In contrast, the low-rank preconditioners will be set up once per Newton iteration and reused for the entire solve. To reduce the cost of the preconditioner, we also have the option to build it on a coarser grid, where matrix-vector-products with the Hessian are much cheaper to compute.

Because the preconditioners build on top of the already preconditioned Hessian system $\tilde{H}$, we have to think about how both preconditioners can be combined. Especially for the PCG method we have to ensure that the combined preconditioner is still symmetric. For this reason, we consider the double preconditioned Hessian system where we first use L and then apply a second preconditioner M_2^{-1} for the preconditioned system. Based on the Cholesky decomposition of the second preconditioner $M_2^{-1} = NN^{\mathrm{T}}$ we get

$$N^{\mathrm{T}} L^{1/2} H L^{1/2} N x = N^{\mathrm{T}} L^{1/2} b$$

where we used a split left and right preconditioning [47]. Comparing this with the preconditioner pattern for the CG method [47] we can read off the combined preconditioner

$$M^{-1} := \left(L^{1/2} N\right)\left(N^{\mathrm{T}} L^{1/2}\right) = L^{1/2} M_2^{-1} L^{1/2},$$

which can be used in a standard PCG algorithm.

6 Numerical Results

For our numerical experiments we use two different registration problems. The first problem registers two x-ray scans of *hands*. The scans are from [42, 4]. While there exists no perfect correspondence, the mismatch term of the registration problem can become very small. The second problem registers two *brain* scans from [1, 13, 30, 29, 14]. Because the brains differ in image intensity, the mismatch term will never become very small.

For all experiments, we start with an initial momentum that is zero everywhere and use the regularization term $\beta_1 \frac{1}{2} \|m_0\|_v^2 = \beta_1 \frac{1}{2} \langle m_0, K m_0 \rangle_{L^2}$ with $K^{-1} = L = (\mathrm{I} - \beta_2 \Delta)^2$. We use the following values for the regularization parameter β unless noted otherwise:

hands:	$\beta_1 = 2^{-3}$	$\beta_2 = 0.02$
brains:	$\beta_1 = 2^{-1}$	$\beta_2 = 0.03$

Figures 6.1 and 6.2 shows the result for the hand and brain registration problem, respectively.

6.1 Performance Metrics

We first discuss how the efficiency of different optimization methods and preconditioners can be compared.

For both, we are interested in methods that are independent of the image resolution. This means that an optimization method has to decrease the norm of the gradient by a fixed factor using a similar number of iterations, regardless of the image resolution. The solver for the preconditioned system has to decrease the norm of the residual by a certain factor using a similar number of iterations.

6.1.1 Optimization Methods

To compare different optimization methods it is not sufficient to compare the number of iterations. Each step of Newton's method is much more expensive than it is for gradient descent or L-BFGS, as it involves the solution of a linear system. When comparing

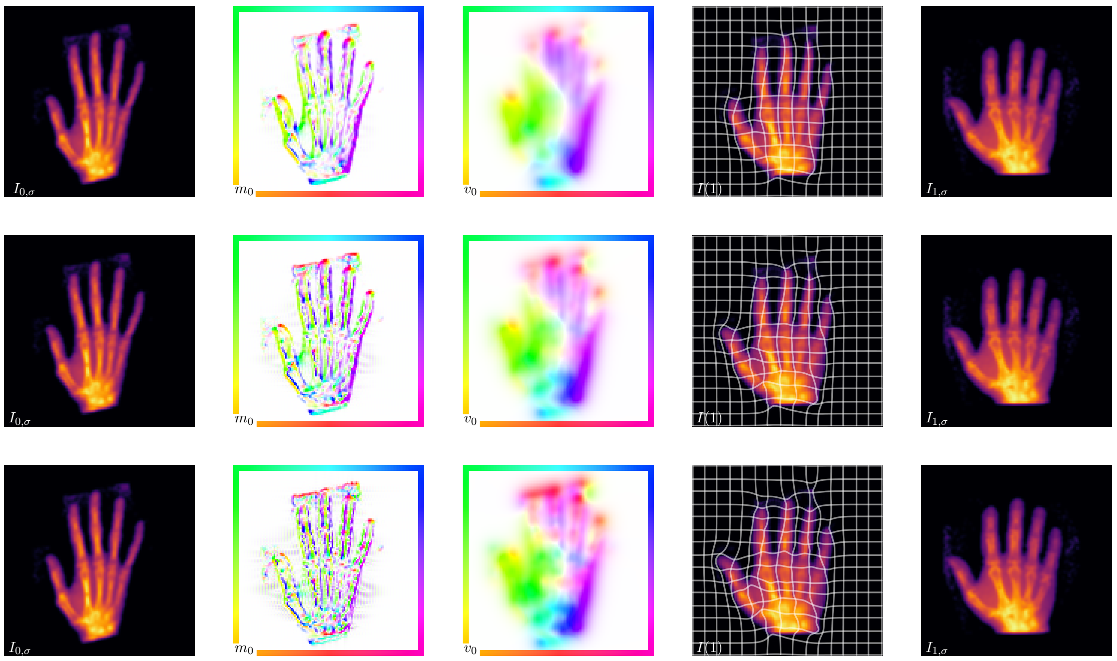

Figure 6.1: Registered hands for regularization parameters $\beta = 2^{-1}$, $\beta = 2^{-3}$ and $\beta = 2^{-5}$. I_0 is on the left and I_1 on the right. In between are the initial momentum, the initial velocity field and the final image $I(1)$. The grid on $I(1)$ illustrates the captured deformation of a regular grid on I_0. The biggest difference can be seen at the thumb and the ring finger. With decreasing regularization the thumb in $I(1)$ rotates more to the left to better resemble the pose in I_1. The point over the ring finger in I_0 does not have a correspondence in I_1. Lower regularization reduces the area covered by the point to reduce the mismatch term. Note the oscillating Gibbs artifacts in the initial momentum for $\beta = 2^{-5}$. This is caused by the spectral spatial discretization and shows that more regularization might be needed.

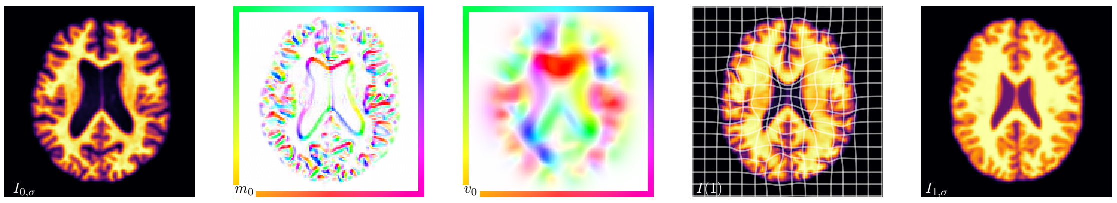

Figure 6.2: Registered brains.

Newton's methods with different preconditioners, the number of Newton iterations should be similar, even though they can vary a lot in computational complexity if one preconditioner is more effective than another.

Because the solution of the state, adjoint and incremental equations are by far the most expensive components during the registration process, we compare the number of PDE solves. This not only includes the gradient and Hessian evaluations, but also functional evaluations during the line search and the costs of setting up or applying the preconditioner. A drawback of this approach is that the higher complexity of the incremental equations is not taken into account. Also, solving on a coarser grid is generally less expensive. Hence, we additionally measure the runtime of the different methods.

6.1.2 Preconditioners

To compare different preconditioners for the linear Hessian system in Newton's method we have to ensure that all tests are performed under the same conditions. An efficient preconditioner is most important close to the solution, where we have to solve the linear Hessian system to a high accuracy. Hence, we compare the convergence of the PCG method with different preconditioners for an initial momentum that is close to the minimizer of the functional. Because the minimizer is not known we first set up a synthetic registration problem that is derived from the real registration problem:

Using a real registration problem we first solve for an initial momentum that is close to the minimizer and compute $\tilde{I}_1 := I(1)$. Then, we consider the registration problem between I_0 and $\tilde{I}_1$, for which we know the initial momentum that maps the images onto each other. We refer to this setup as being *at the solution* (of the synthetic registration problem) as it perfectly maps the images onto each other. While this initial momentum is not the minimizer of the new minimization functional, it should be close enough to the minimizer for a small regularization term. Therefore, we use this new regularization problem to test the preconditioners. We perform a single Newton step using the computed initial momentum as starting point and compare the convergence of the PCG method for different preconditioners.

For the low rank approximation based preconditioners we also want to know how well a preconditioner built in one iteration generalizes to other iterations. If a preconditioner can be reused in other Newton iterations, it does not have to be recomputed in every iteration and the costs of setting it up could be amortized by slacking the preconditioner, or building it incrementally during consecutive iterations.

To evaluate the efficiency of a preconditioner we also have to take the additional cost of applying the preconditioner into account.

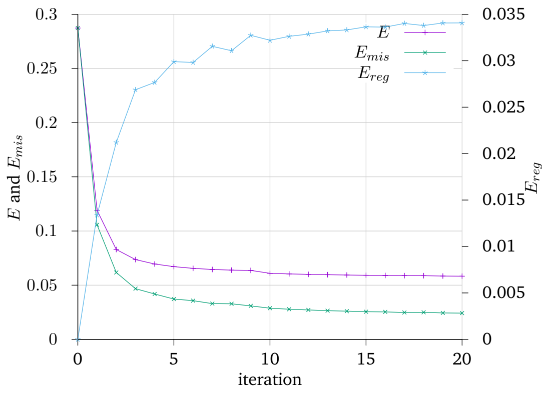

Figure 6.3: Trend of objective functional E, regularization term and mismatch term during the minimization process. The regularization term uses the right y-axis.

6.2 General Convergence

Before we discuss the efficiency of different preconditioners for Newton's method, we start with a general overview of the convergence during the image registration optimization process. Therefore, we consider the regularization and mismatch term of the minimization functional along with its gradient and how they change during the minimization.

In general, the value of the minimization functional is expected to decrease, because each optimization step tries to reduce its value. The image mismatch term should also decay initially and level out at some value, when the optimization process starts to get closer to the optimum. In general, we cannot expect the mismatch term to converge to zero, as there might not exist a mapping that perfectly maps the two images onto each other: One can think of two versions of otherwise identical images of an object where noise has been added. In this case, the mismatch term will converge to a nonzero value. Initially, the regularization term is zero, because we initialize the optimization with an initial momentum of zero, which will map I_0 onto itself. During the optimization, we expect the regularization term to increase and then level out once the optimization gets closer to the optimum. Figure 6.3 shows an example of this behavior.

Even though the terms of minimization function level out quickly for all optimization methods, the norm of the gradient still decreases further to zero. However, it does not have to decrease monotonically as can be seen for the L-BFGS and gradient descent method in figure (6.4). While Armijo line search ensures that the minimization functional is reduced in each step, this does not imply that the norm of the gradient is reduced in each step.

The Gauss-Newton method converges much faster and needs less iterations compared to gradient descent and L-BFGS. One can argue, that the Gauss-Newton method uses a better approximation to the inverse of the Hessian than the L-BFGS approximation.

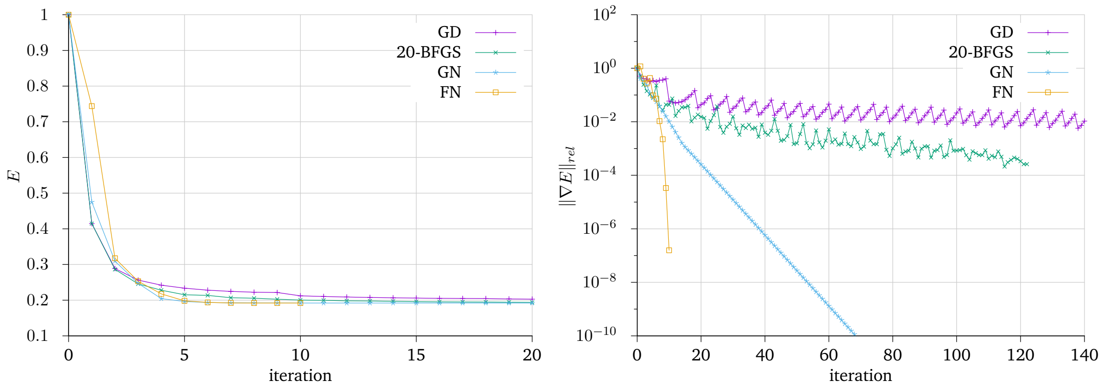

Figure 6.4: Value of the minimization functional (left) and norm of the gradient (right) after each iteration for the hand 64×64 image while solving for a vector valued momentum.

Under the assumption that the local second order approximation to the minimization functional holds, the higher quality of the Gauss-Newton approximation can be seen in the fact, that L-BFGS method terminates after about 120 iterations with a line search failure. In contrast, the Gauss-Newton method reaches a much smaller value for the gradient norm without performing any line search.

Nonetheless, we have to keep in mind that each Newton step is significantly more expensive because we have to solve a linear system. This involves many solutions of the incremental equations in the Hessian matrix-vector-products. In section 6.6 we compare the different optimization algorithms in more detail, based on the computational costs.

The classical (full) Newton method converges much faster than the Gauss-Newton method. However, using Newton's method does not always converge to a minimum, especially when the Hessian becomes negative definite or indefinite. This problem does not arise with the Gauss-Newton approximation, as it ensures the positive definiteness of the Hessian approximation. In the following experiments we therefore focus on the Gauss-Newton method.

6.3 L-BFGS Convergence

The convergence of L-BFGS tends to improve the more updates are used in the inverse Hessian approximation. Only storing the last update already significantly reduces the iteration number compared to the gradient descent method. Figure 6.5 shows the effect of different memory sizes. The number of iterations is smaller when inverting for a vector-valued momentum compared to inverting for a scalar-valued momentum. For a larger memory, this difference becomes smaller. Considering the runtime, the scalar-valued

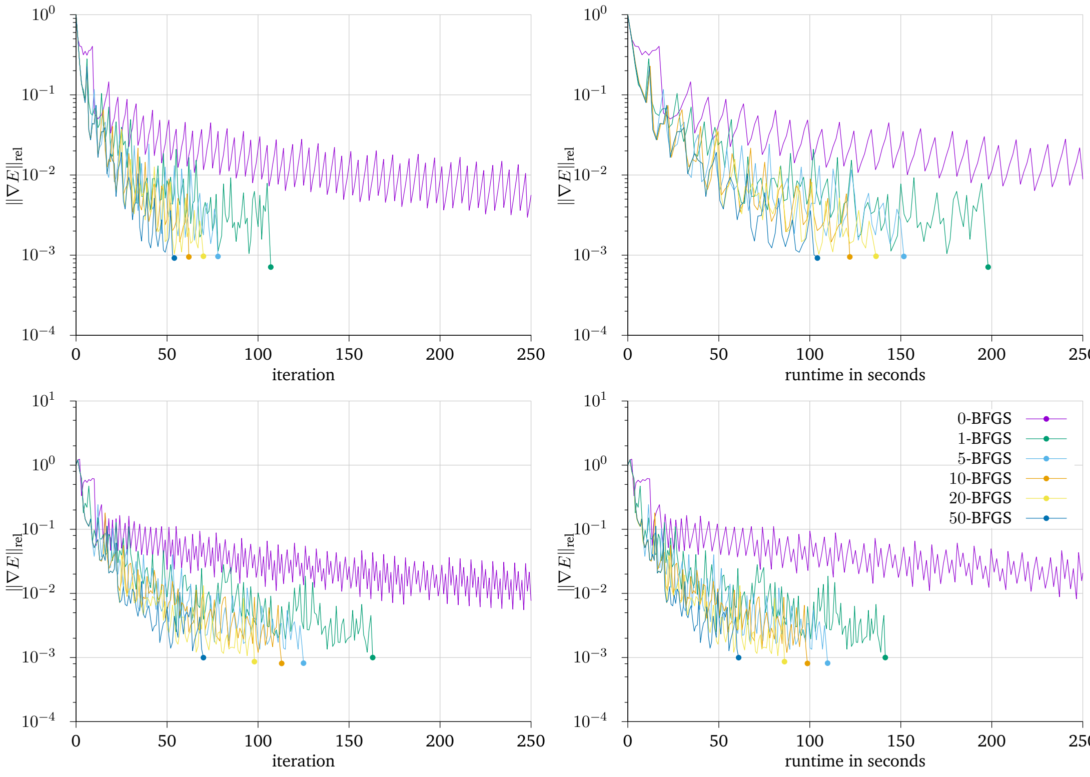

Figure 6.5: Convergence of L-BFGS for vector valued momentum (top) and scalar valued momentum (bottom).

momentum performs much better.

6.4 Spectral Properties of the Hessian and Low Rank Approximation

To use the low rank approximation from section 5.3 as an efficient preconditioner, we have to ensure that the spectrum of the mismatch term of the Hessian decays quickly, so that there exists a good low rank approximation.

Therefore, we compute an eigenvalue decomposition of the individual components of the Hessian to see, how fast the eigenvalues decay. As mentioned in section 6.1 we build the preconditioner at the solution and for an initial momentum of zero, to see how good the preconditioner works close to the minimizer and how much the spectral properties change during the optimization. Additionally, we compare the spectrum of the Hessian on a fine and coarse grid, to see if it is possible to build the low rank approximation on a coarser grid as suggested in section 5.4.

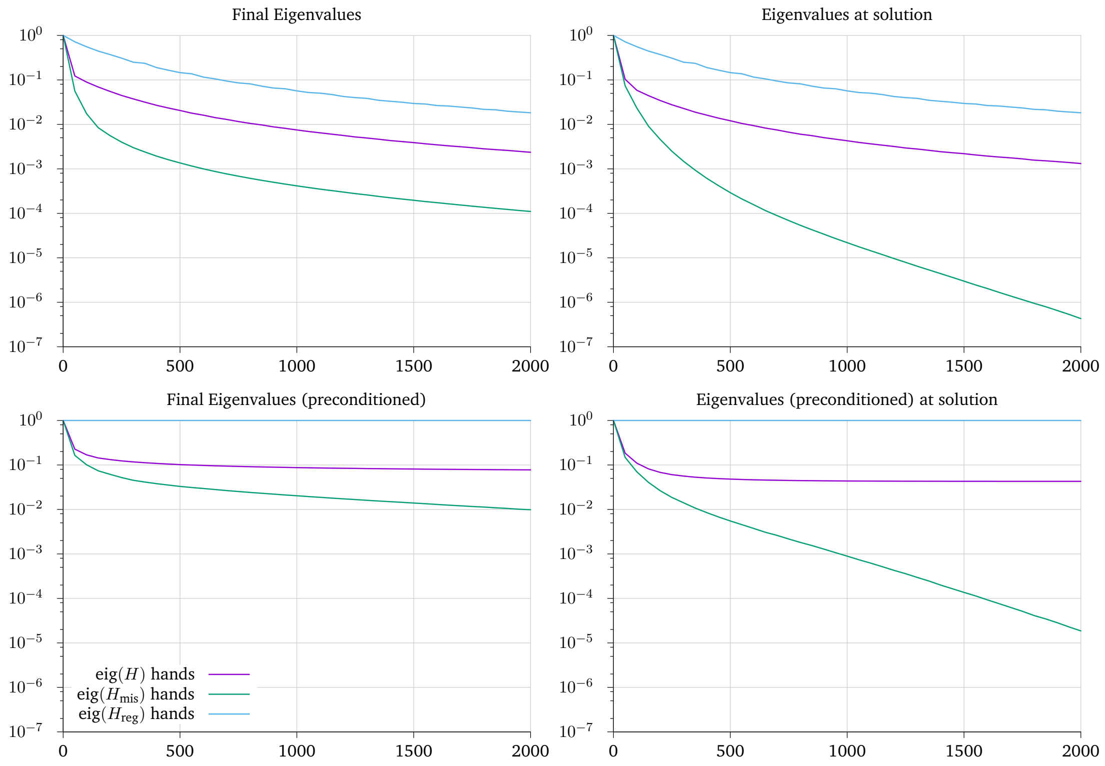

Figure 6.6: Largest 2000 eigenvalues for the Gauss-Newton Hessian approximation for the hand registration problem on a 64×64 grid for the vector valued momentum formulation. Left: Eigenvalues after convergence. Right: Eigenvalues at the solution, where the mismatch is zero.

6.4.1 Decay of Eigenvalues

To compute the eigenvalues of the Hessian, we first build the full Hessian using matrix-vector-products and then compute an eigenvalue decomposition of this Hessian matrix. Figure 6.6 shows the decay of the eigenvalues of the individual term of the Hessian for the hand registration problem. We see, that the mismatch term decays much faster than the regularization term and the eigenvalues of the full Hessian. Additionally, the eigenvalues of the mismatch term decay even faster at the solution. Therefore, we can expect a good low rank approximation for the mismatch term of the Hessian. This is important, because a good preconditioner is of special importance close to the minimizer, where the linear system for the Newton step has to be solved with a high accuracy and a good preconditioner can reduce the number of iterations significantly.

For the L-preconditioned system (5.1) the decay is very similar. The eigenvalues of the mismatch term decay a bit slower than in the unconditioned system. However, because the preconditioned Hessian is the sum of the scaled identity matrix βI and a mismatch

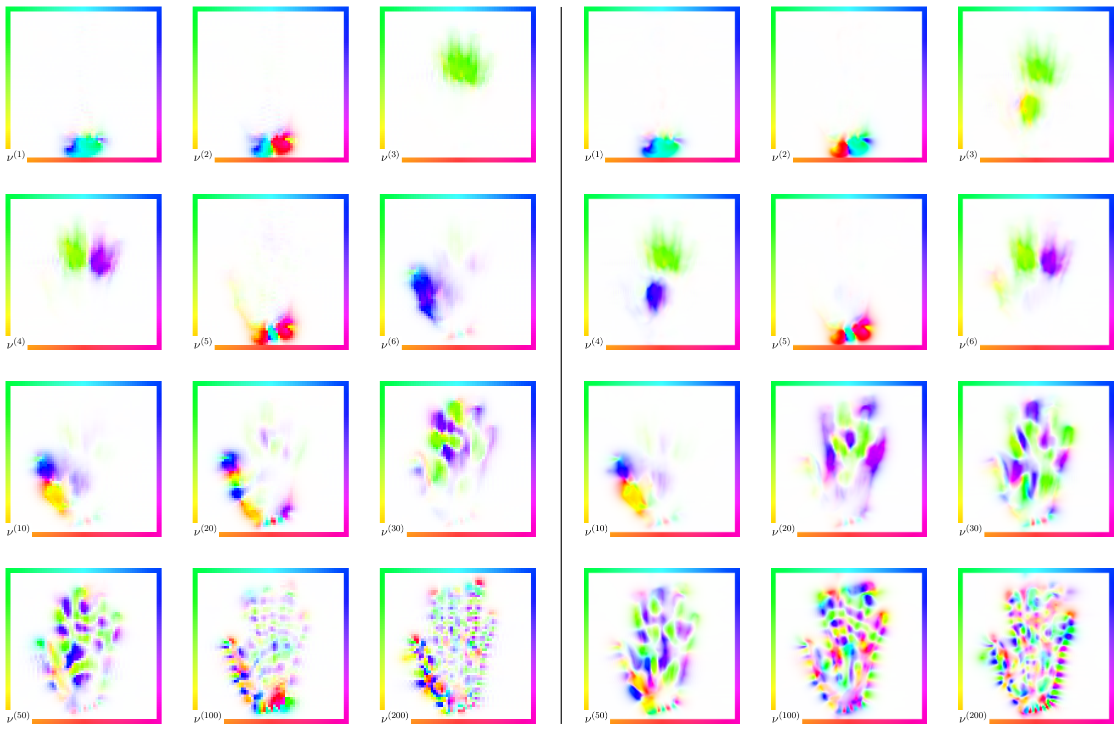

Figure 6.7: Computed eigenvectors $\nu^{(i)}$ of the mismatch term $\tilde{H}_{\text{mis}}$ at the solution of the hand registration problem on a coarse 64×64 (left) and fine 128×128 (right) grid. The eigenvalues corresponding to the largest eigenvalues are similar on both resolutions, even though the two images cannot represent the same level of detail.

term $\tilde{H}_{\text{mis}}$ with a fast decaying spectrum, the condition number of the L-preconditioned system is better than the original system.

Coarse Grid

To justify the eigenvalue decomposition on a coarser grid, we have to ensure that the largest eigenvalues of $\tilde{H}_{\text{mis}}$ and the corresponding eigenvectors are similar on both resolutions.

Figure 6.7 shows the eigenvectors for different resolutions. We see, that the eigenvectors corresponding to the largest eigenvalues are smoother than the eigenvectors for smaller eigenvalues and are similar on different resolutions. Also the eigenvalues have very similar values. The eigenvectors corresponding to smaller eigenvalues contain much higher frequencies and differ a lot. Using the eigenpairs from the coarse grid for the low rank approximation should therefore be possible, as long as the number of eigenvalues is not too large.

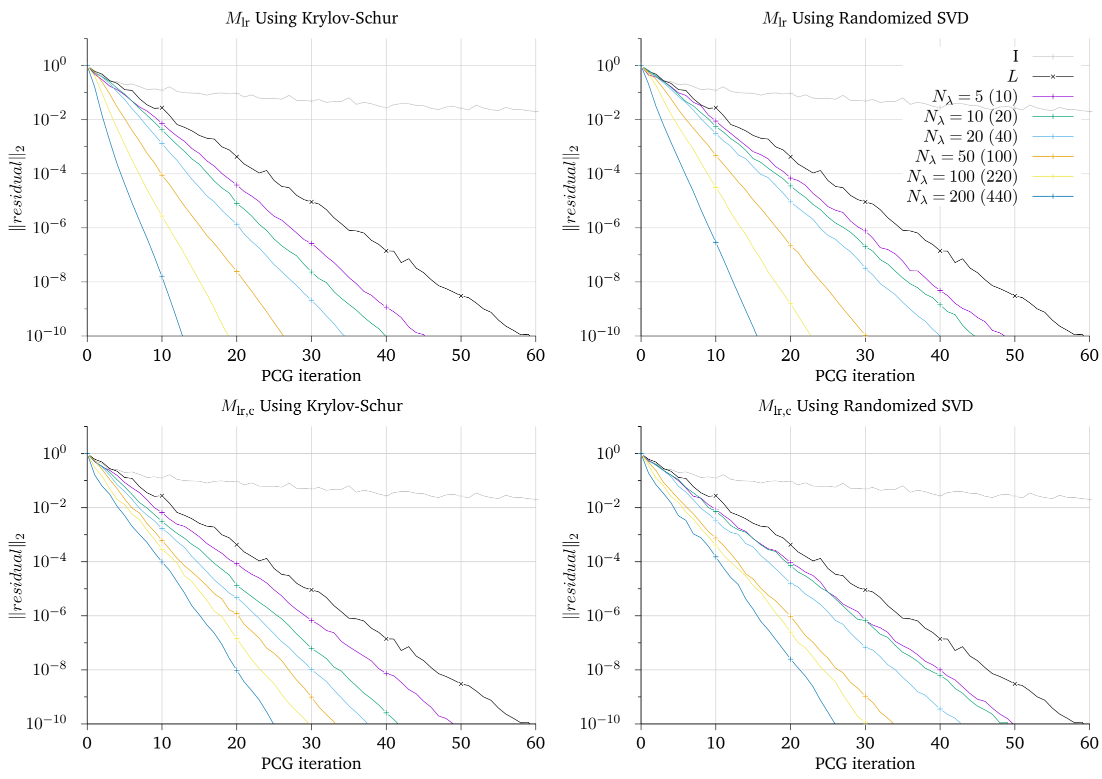

Figure 6.8: Convergence using different numbers of eigenvalues N_λ for M_{lr} (top) and $M_{\mathrm{lr,c}}$ (bottom). In each row, the preconditioner built with Krylov-Schur is on the left and the preconditioner built with the randomized algorithm is on the right. The number of Hessian vector multiplications needed to build the preconditioners are given in brackets.

Convergence at the Solution and Expansion Length

The quality of the preconditioner strongly depends on the rank of the low rank approximation. A higher rank, and thus a larger number of eigenvalues, should give better results, given that the eigenvalue composition is performed with a high accuracy. However, computing more eigenvalues also implies that the preconditioner becomes more expensive to set up because more matrix-vector-multiplications with the Hessian are needed. Computing the preconditioner only once with an initial momentum of zero and using it for the entire registration process did not work and reduced the convergence of the PCG solver significantly.

Since the inverse regularization operator as preconditioner is cheap to apply and does not need any setup, we use it as a base line for the effectiveness for the different preconditioners.

Figure 6.8 shows the convergence of different preconditioners at the solution. We see

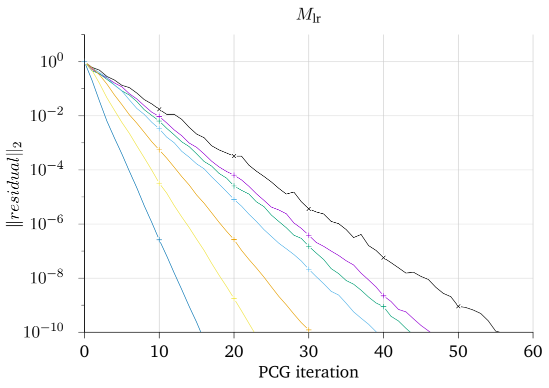

Figure 6.9: Convergence of low-rank preconditioners for the hand problem using different number of eigenvalues N_λ on a 265×265 grid that was prolonged from 128×128. Left: low-rank preconditioner M_{lr}. Right: low-rank preconditioner $M_{\text{lr,c}}$ on a coarse grid. The number of matrix-vector-products with the Hessian needed to build the preconditioners are given in brackets. All preconditioners use the randomized eigendecomposition method.

that a low rank preconditioner clearly improves the convergence. Furthermore, the accuracy of the preconditioner can be improved using a larger number of eigenpairs which improves the convergence.

The difference between different eigendecomposition algorithms is rather small. The preconditioner using the randomized decomposition performs slightly worse than the preconditioner using a Krylov-Schur algorithm [51] with a high accuracy. In particular we use Matlab's `eigs` function with a tolerance of 10^{-14} and the `IsFunctionSymmetric` flag. However, the randomized algorithm uses fewer Hessian matrix-vector-multiplications for a small number of eigenvalues to set up the preconditioner than the Krylov-Schur algorithm.

The convergence of the PCG method for the Hessian solver when the preconditioner is built on a coarse grid is similar but slightly worse. For a small number of eigenpairs the results do not differ a lot. With a larger number of eigenpairs the convergence is worse but still clearly improves the performance. However, compared to the convergence of the preconditioners that are set up on the original grid, the preconditioner is less effective. A reason for this is that the finer grid contains image details that are not present on the coarse grid, which causes the eigenvectors to differ. Especially eigenvectors containing high frequencies are different on different resolutions. However, setting up the preconditioner on a coarser grid is significantly less expensive, which can make up for the slower convergence.

If the image on the fine grid is a prolonged version of a coarser image, both images contain the same level of detail. In this case high frequency eigenvectors are much more

				matvecs				
precond	N_λ	N_x	method	fine	coarse	iters	reduction	red. / iter
I	-	128	-	-	-	2510	$9.8 \cdot 10^{-07}$	$9.9 \cdot 10^{-01}$
L	-	128	-	-	-	60	$5.6 \cdot 10^{-11}$	$6.7 \cdot 10^{-01}$
M_{lr}	10	128	eigs	66	-	40	$9.3 \cdot 10^{-11}$	$5.6 \cdot 10^{-01}$
M_{lr}	10	128	rsvd	20	-	45	$8.5 \cdot 10^{-11}$	$6.0 \cdot 10^{-01}$
$M_{\text{lr,c}}$	10	128	eigs	-	61	42	$7.8 \cdot 10^{-11}$	$5.7 \cdot 10^{-01}$
$M_{\text{lr,c}}$	10	128	rsvd	-	20	50	$4.0 \cdot 10^{-11}$	$6.2 \cdot 10^{-01}$
M_{lr}	10	$128^{(p)}$	eigs	63	-	38	$5.6 \cdot 10^{-11}$	$5.4 \cdot 10^{-01}$
M_{lr}	10	$128^{(p)}$	rsvd	20	-	42	$8.4 \cdot 10^{-11}$	$5.8 \cdot 10^{-01}$
$M_{\text{lr,c}}$	10	$128^{(p)}$	eigs	-	68	42	$6.8 \cdot 10^{-11}$	$5.7 \cdot 10^{-01}$
$M_{\text{lr,c}}$	10	$128^{(p)}$	rsvd	-	20	45	$6.7 \cdot 10^{-11}$	$5.9 \cdot 10^{-01}$
M_{lr}	100	128	eigs	251	-	19	$8.7 \cdot 10^{-11}$	$3.0 \cdot 10^{-01}$
M_{lr}	100	128	rsvd	220	-	23	$7.5 \cdot 10^{-11}$	$3.6 \cdot 10^{-01}$
$M_{\text{lr,c}}$	100	128	eigs	-	251	30	$8.0 \cdot 10^{-11}$	$4.6 \cdot 10^{-01}$
$M_{\text{lr,c}}$	100	128	rsvd	-	220	31	$6.0 \cdot 10^{-11}$	$4.7 \cdot 10^{-01}$
M_{lr}	100	$128^{(p)}$	eigs	251	-	17	$3.2 \cdot 10^{-11}$	$2.4 \cdot 10^{-01}$
M_{lr}	100	$128^{(p)}$	rsvd	220	-	20	$8.1 \cdot 10^{-11}$	$3.1 \cdot 10^{-01}$
$M_{\text{lr,c}}$	100	$128^{(p)}$	eigs	-	251	21	$6.0 \cdot 10^{-11}$	$3.3 \cdot 10^{-01}$
$M_{\text{lr,c}}$	100	$128^{(p)}$	rsvd	-	220	23	$7.9 \cdot 10^{-11}$	$3.6 \cdot 10^{-01}$

Table 6.1: Convergence of the PCG method at the solution with different preconditioners for the hand registration problem. The randomized algorithm is marked as "rsvd" and the Krylov-Schur algorithm is marked as "eigs". The "matvecs" denote the number of matrix-vector-products on a coarse and fine grid during the preconditioner setup to estimate N_λ eigenpairs. "iters" is the number of PCG iterations to reduce the initial residuum norm by the factor "reduction". "red. / iter" is the average reduction of the norm per iteration, computed by $\sqrt[\text{iters}]{\text{reduction}}$. Images that are prolonged from coarser data are marked with $^{(p)}$.

similar on the different resolutions. Figure 6.9 shows the convergence for prolonged data. The convergence for the preconditioner build on the coarse grid is overall better than for non-prolonged images and much closer to the convergence with the preconditioner that is built on the original grid. Compared to the measurements with non-prolonged images, this also means better convergence for a large number of eigenvalues.

In table 6.1 we give a detailed overview of the different preconditioners with different eigenvalue estimators, their performance, and the associated costs. Generally, the costs of setting up a preconditioner should not dominate the savings gained by a reduced number of PCG iteration. In section 6.5 we take a closer look at the costs of each preconditioner during the registration process.

6.4.2 Resolution Independence

The effectiveness of the preconditioners is independent of the resolution. The convergence of the inverse regularization preconditioner L and the low rank preconditioner is very similar on different image resolutions. This means, that the number of PCG iterations is roughly the same on all resolutions. This can be seen when we compare the convergence results observed in figure 6.8 with the convergence results reported in figure 6.9. Both experiments use the same setup, but in the latter experiment, the Hessian system is solved for a prolonged 256×256 image.

6.5 Preconditioner for Newton's Method

To evaluate the effectiveness of the preconditioners, we measure the cost of the whole image registration process. Figure 6.10 shows the convergence of the Gauss-Newton method with different preconditioners. The convergence measured in iterations remains almost unchanged for different preconditions and shows a first order convergence. The only exception is the unpreconditioned Gauss-Newton method, which does not converge further than 10^{-2}. This can be explained by the bad condition number of the Hessian, which can cause large errors in the search direction.

The number of PDE solves and the total runtime shows more difference between the methods. Most low-rank preconditioners perform worse than the L-preconditioner. Slacking the preconditioner for a certain number of iterations does reduce the costs of the preconditioners enough to justify the savings in the reduced number of PCG iterations. Only one preconditioner on the coarse grid is able to reduce the runtime slightly when the problem is solved for a very small gradient norm.

6.6 Comparison of L-BFGS and Newton Methods

To give a final evaluation of the different methods we now compare the performance of Newton's method combined with a preconditioner against the performance of L-BFGS. Figure 6.11 refines the results from section 6.2. While the Newton methods clearly need less iterations, the L-BFGS method has a better runtime, and requires less PDE solves. However, it terminates at a relative gradient norm of about 10^{-4} due to a line search failure. The Gauss-Newton method is able to solve to a much higher accuracy.

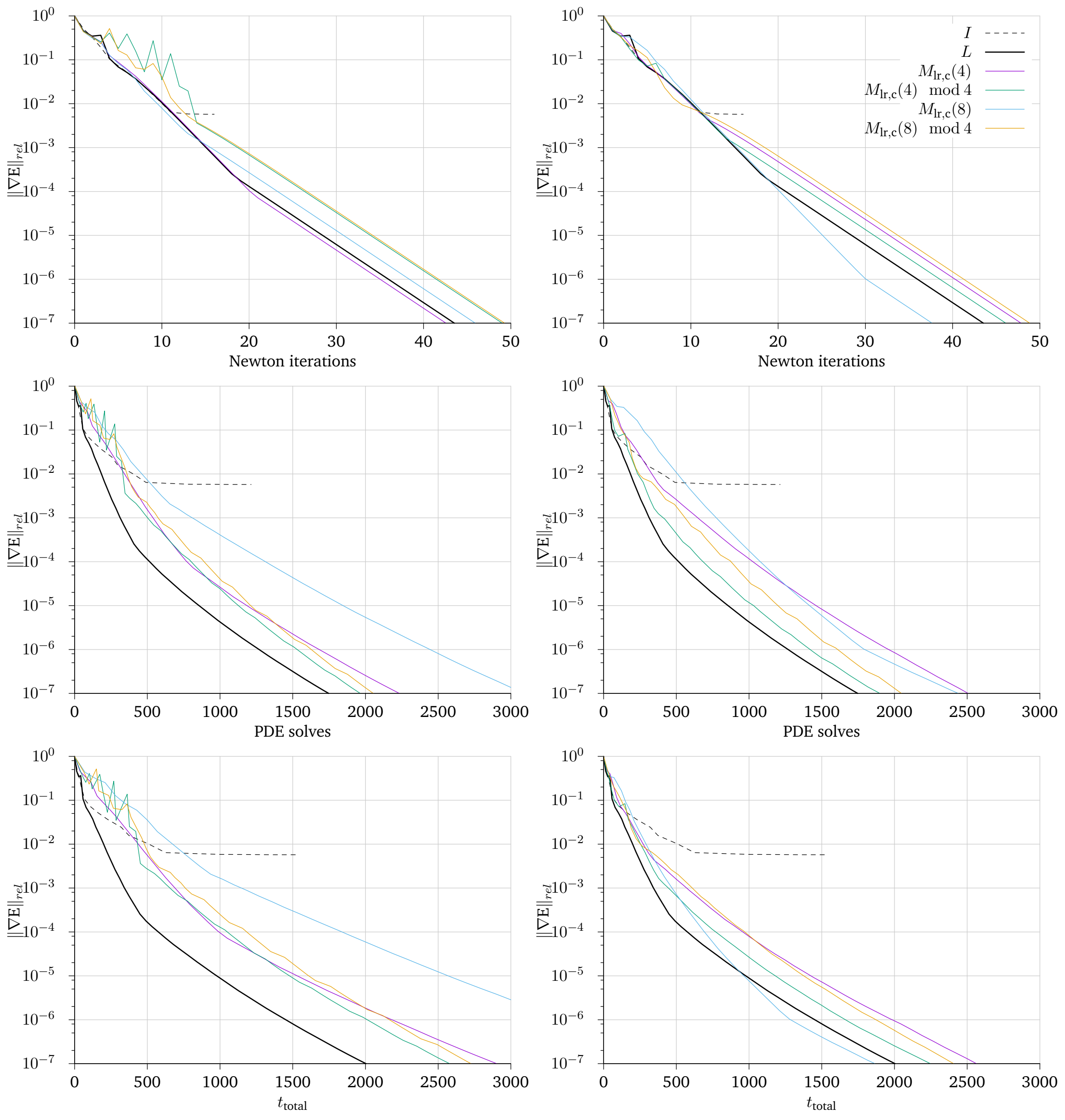

Figure 6.10: Convergence of Gauss-Newton with different low-rank preconditioners for the hand registration problem on a 64×64 grid.

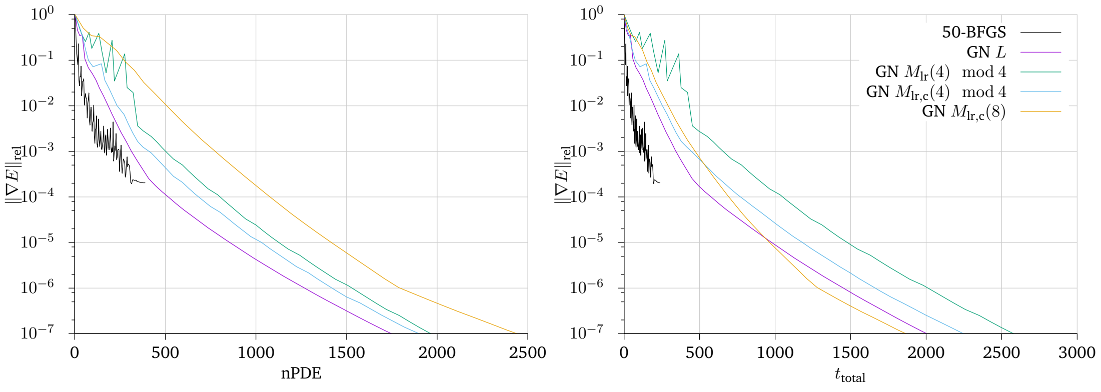

Figure 6.11: Comparison of L-BFGS and Newton's method with different preconditioners for the hand registration problem on a 64×64 grid.

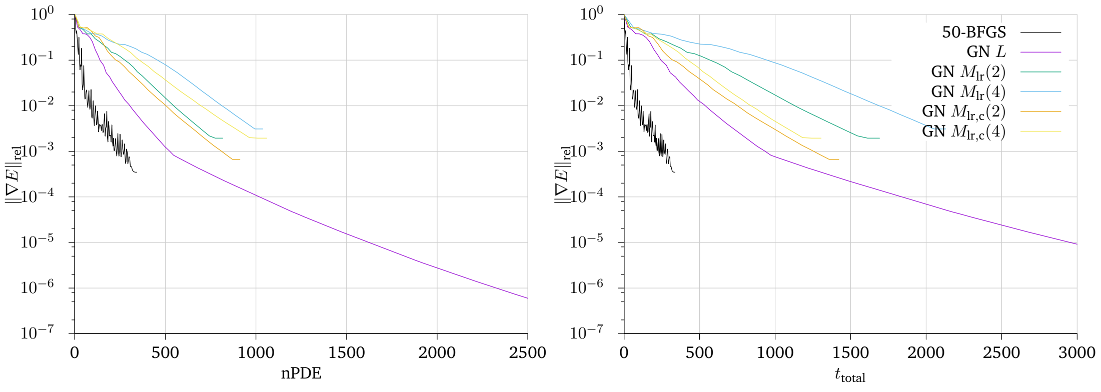

Figure 6.12: Comparison of L-BFGS and Newton's method with different preconditioners for the brain registration problem on a 64×64 grid.

Similar results are obtained for the brain registration problem in figure 6.12. However, in this experiment no low-rank preconditioned method was able to reduce the norm of the gradient by more than a factor of 10^{-3}.

6.7 Using the Semi-Lagrangian-Method

The semi-Lagrangian method allows to use much larger time steps than Heun's method. Therefore, the solution of the state, adjoint and incremental equations is much cheaper, as fewer time steps have to be computed. Compared to the results using Heun's method, the overall runtime is greatly reduced. Figure 6.13 shows that while the semi-Lagrangian scheme improves the runtime, it does not change the qualitative results from the previous section.

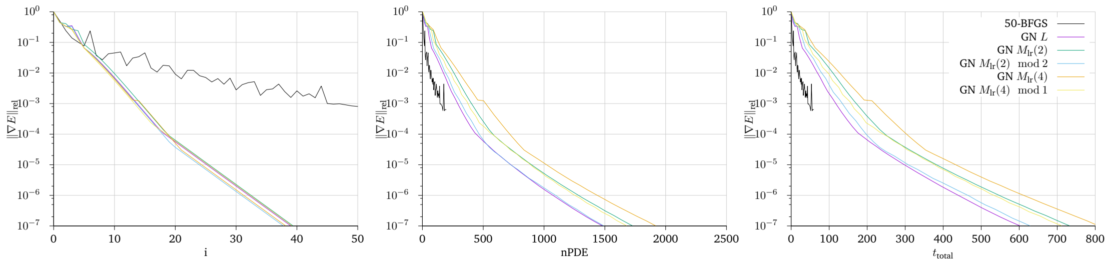

Figure 6.13: Convergence for different methods using the semi-Lagrangian method using 8 time steps for the time integration of the PDEs. Note that the scaling of the x-axis for the time measurement is different from figure 6.11.

			total PDEs		PDE solves					
method	iters	secs	fine	coarse	grad	LS	CG	PC	PC c	$\|\nabla E\|_{\text{rel}}$
GD	687	1327	2210	0	1376	832	0	0	0	$1.0 \cdot 10^{-03}$
1-BFGS	107	198	331	0	216	114	0	0	0	$7.1 \cdot 10^{-04}$
50-BFGS	54	104	166	0	110	55	0	0	0	$9.2 \cdot 10^{-04}$
GN	16	1531	1214	0	34	32	1140	0	0	$5.7 \cdot 10^{-03}$
GN L	16	375	342	0	34	15	292	0	0	$6.5 \cdot 10^{-04}$
GN M_{lr}	16	735	574	0	34	15	268	256	0	$6.8 \cdot 10^{-04}$
GN $M_{\text{lr,c}}$	16	501	306	512	34	15	256	0	512	$7.1 \cdot 10^{-04}$
GN $M'_{\text{lr,c}}$	19	577	429	160	40	18	370	0	160	$8.0 \cdot 10^{-04}$

Table 6.2: Convergence of different optimization methods. $M'_{\text{lr,c}}$ is a slacked low rank preconditioner on a coarse grid. Note that the gradient descent and Gauss-Newton method without preconditioner did not reach the tolerance of $\|\nabla E\|_{\text{rel}} \leq 10^{-3}$.

6.8 Summary

Table 6.2 shows a breakdown of the costs of different methods.

In our experiments, L-BFGS proved to be the most efficient algorithm for both, the scalar and vector-valued momentum formulation. With few changes, it is easily implemented based on a gradient descent method, and provides a significant improvement. However, the methods showed problems when the gradient norm becomes small.

The Gauss-Newton method uses much less Newton iterations than L-BFGS, however each step includes many PDE solves in the inner PCG method. Using the Gauss-Newton method without a preconditioner is very expensive, because many iterations are needed to solve the linear system.

The different preconditioners are able to reduce the number of inner iterations and greatly improve the performance of the Gauss-Newton method. Among the considered preconditioners, using the inverse regularization term of the Hessian proved to be the cheapest and most effective option. While low-rank preconditioners clearly improved the convergence of the inner PCG solver, the setup costs of the approximation are too high to justify the reduced PCG iteration number during the registration process. To reduce the costs of setting up the preconditioner we build it on a coarse grid. This works well with only a minor reduction of the PCG performance. Slacking the approximation for some iterations, reduces the costs of the preconditioner but the reduced quality of the preconditioner often leads to more PCG iterations and does not reduce the costs far enough to provide an advantage over the L-preconditioner.

Overall, L-BFGS turned out to be the most efficient method regardless of the time integrator. It only requires gradient information and is algorithmically much less complex than a preconditioned Gauss-Newton method. Nonetheless, latter are able to solve the inverse problem with higher accuracy which could not be achieved by L-BFGS.

7 Outlook

While the Gauss-Newton method ensures that each update reduces the minimization functional, its convergence is much slower than Newton's method. An alternative approach would be to run the (full) Newton's method and only fall back to Gauss-Newton if a negative- or indefinite matrix is detected.

To reduce the setup costs of the preconditioners, other ways have to be found to distribute the costs over multiple iterations. While slacking the preconditioner did not work well in our experiments, one can think about how the cost of the randomized singular value decomposition can be split over different iterations. One way could be to reuse the basis for multiple Newton iterations, and only recompute the eigenvalues. One could also examine if it is possible to use the existing matrix-vector-products, which are present in the PCG algorithm, to build a preconditioner.

Further work is needed to evaluate how the results for the low rank preconditioner differ in three dimensions. While they might require a larger number of eigenpairs, operations on a coarse grid are much cheaper relative to the finer resolution due to the increased dimensionality.

Bibliography

[1] *BrainWeb: Simulated brain database.* `http://www.bic.mni.mcgill.ca/brainweb/`.

[2] A. Alexanderian, N. Petra, G. Stadler, and O. Ghattas, *A fast and scalable method for a-optimal design of experiments for infinite-dimensional bayesian nonlinear inverse problems*, SIAM Journal on Scientific Computing, 38 (2016), pp. A243–A272.

[3] H. W. Alt, *Linear functional analysis: an application-oriented introduction*, Springer, 2016.

[4] Y. Amit, *A nonlinear variational problem for image matching*, SIAM Journal on Scientific Computing, 15 (1994), pp. 207–224.

[5] J. Ashburner and K. J. Friston, *Diffeomorphic registration using geodesic shooting and Gauss–Newton optimisation*, NeuroImage, 55 (2011), pp. 954–967.

[6] M. F. Beg, M. I. Miller, A. Trouvé, and L. Younes, *Computing large deformation metric mappings via geodesic flows of diffeomorphisms*, International Journal of Computer Vision, 61 (2005), pp. 139–157.

[7] M. Benzi, E. Haber, and L. Taralli, *A preconditioning technique for a class of pde-constrained optimization problems*, Advances in Computational Mathematics, 35 (2011), pp. A2494–A2523.

[8] J. P. Boyd, *Chebyshev and Fourier spectral methods*, Dover, Mineola, New York, US, 2000.

[9] A. M. Bradley, *PDE-constrained optimization and the adjoint method*, (2010).

[10] T. Bui-Thanh, O. Ghattas, J. Martin, and G. Stadler, *A computational framework for infinite-dimensional Bayesian inverse problems part I: The linearized case, with application to global seismic inversion*, SIAM Journal on Scientific Computing, 35 (2013), pp. A2494–A2523.

[11] J. C. Butcher and N. Goodwin, *Numerical methods for ordinary differential equations*, Wiley Online Library, 3rd ed., 2016.

[12] G. E. Christensen, R. D. Rabbitt, and M. I. Miller, *Deformable templates using large deformation kinematics*, IEEE Transactions on Image Processing, 5 (1996), pp. 1435–1447.

[13] C. A. Cocosco, V. Kollokian, R. K.-S. Kwan, G. B. Pike, and A. C. Evans, *BrainWeb: Online interface to a 3D MRI simulated brain database*, in NeuroImage, vol. 5, May 1997, p. S425.

[14] D. L. Collins, A. P. Zijdenbos, V. Kollokian, J. G. Sled, N. J. Kabani, C. J. Holmes, and A. C. Evans, *Design and construction of a realistic digital brain phantom*, IEEE Transactions on Medical Imaging, 17 (1998), pp. 463–468.

[15] J. W. Cooley and J. W. Tukey, *An algorithm for the machine calculation of complex fourier series*, Mathematics of Computation, 19 (1965), pp. 297–301.

[16] R. Courant, K. Friedrichs, and H. Lewy, *Über die partiellen Differenzengleichungen der mathematischen Physik*, Mathematische Annalen, 100 (1928), pp. 32–74.

[17] P. Dupuis, U. Grenander, and M. I. Miller, *Variational problems on flows of diffeomorphisms for image matching*, Quarterly of Applied Mathematics, 56 (1998), pp. 587–600.

[18] S. C. Eisentat and H. F. Walker, *Choosing the forcing terms in an inexact Newton method*, SIAM Journal on Scientific Computing, 17 (1996), pp. 16–32.

[19] H. W. Engl, M. Hanke, and A. Neubauer, *Regularization of inverse problems*, Kluwer Academic Publishers, 1996.

[20] B. Fischer and J. Modersitzki, *Ill-posed medicine—an introduction to image registration*, Inverse Problems, 24 (2008), p. 034008.

[21] M. D. Gunzburger, *Perspectives in flow control and optimization*, SIAM, Philadelphia, Pennsylvania, US, 2003.

[22] N. Halko, P.-G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM review, 53 (2011), pp. 217–288.

[23] M. Hernandez, *PDE-constrained LDDMM via geodesic shooting and inexact Gauss–Newton–Krylov optimization using the incremental adjoint Jacobi equations*, Physics in Medicine & Biology, 64 (2019), p. 025002.

[24] M. Hernandez and S. Olmos, *Gauss-newton optimization in diffeomorphic registration*, in Proc IEEE International Symposium on Biomedical Imaging, 2008, pp. 1083–1086.

[25] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, Journal of Research of the National Bureau of Standards, 49 (1952), pp. 409–436.

[26] M. Hinze, R. Pinnau, M. Ulbrich, and S. Ulbrich, *Optimization with PDE constraints*, vol. 23, Springer Science & Business Media, 2009.

[27] E. Kalnay, *Atmospheric modeling, data assimilation and predictability*, Cambridge University Press, 2003.

[28] D. A. KNOLL AND D. E. KEYES, *Jacobian-free Newton–Krylov methods: a survey of approaches and applications*, Journal of Computational Physics, 193 (2004), pp. 357–397.

[29] R. K.-S. KWAN, A. C. EVANS, AND G. B. PIKE, *An extensible MRI simulator for post-processing evaluation*, in Visualization in Biomedical Computing, vol. 1131 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1996, pp. 135–140.

[30] R.-S. KWAN, A. C. EVANS, AND G. B. PIKE, *MRI simulation-based evaluation of image-processing and classification methods*, IEEE Transactions on Medical Imaging, 18 (1999), pp. 1085–1097.

[31] R. J. LEVEQUE, *Finite Volume Methods for Hyperbolic Problems*, Cambridge Texts in Applied Mathematics, Cambridge University Press, 2002.

[32] A. MANG AND G. BIROS, *An inexact Newton–Krylov algorithm for constrained diffeomorphic image registration*, SIAM Journal on Imaging Sciences, 8 (2015), pp. 1030–1069.

[33] ——, *Constrained H^1-regularization schemes for diffeomorphic image registration*, SIAM Journal on Imaging Sciences, 9 (2016), pp. 1154–1194.

[34] ——, *A semi-Lagrangian two-level preconditioned Newton–Krylov solver for constrained diffeomorphic image registration*, SIAM Journal on Scientific Computing, 39 (2017), pp. B1064–B1101.

[35] A. MANG, A. GHOLAMI, AND G. BIROS, *Distributed-memory large-deformation diffeomorphic 3D image registration*, in Proc ACM/IEEE Conference on Supercomputing, no. 72, 2016.

[36] A. MANG, A. GHOLAMI, C. DAVATZIKOS, AND G. BIROS, *CLAIRE: A distributed-memory solver for constrained large deformation diffeomorphic image registration*, arXiv preprint arXiv:1808.04487, (2018).

[37] A. MANG, A. GHOLAMI, C. DAVATZIKOS, AND G. BIROS, *PDE-constrained optimization in medical image analysis*, Optimization and Engineering, 19 (2018), pp. 765–812.

[38] A. MANG AND L. RUTHOTTO, *A Lagrangian Gauss–Newton–Krylov solver for mass- and intensity-preserving diffeomorphic image registration*, SIAM Journal on Scientific Computing, 39 (2017), pp. B860–B885.

[39] M. I. MILLER, A. TROUVÉ, AND L. YOUNES, *On the metrics and Euler–Lagrange equations of computational anatomy*, Annual Review of Biomedical Engineering, 4 (2002), pp. 375–405.

[40] M. I. MILLER, A. TROUVÉ, AND L. YOUNES, *Geodesic shooting for computational anatomy*, Journal of Mathematical Imaging and Vision, 24 (2006), pp. 209–228.

[41] J. MODERSITZKI, *Numerical methods for image registration*, Oxford University Press, 2004.

[42] ——, *FAIR: Flexible algorithms for image registration*, vol. 6, SIAM, 2009.

[43] J. Nocedal, *Updating quasi-newton matrices with limited storage*, Mathematics of computation, 35 (1980), pp. 773–782.

[44] J. Nocedal and S. J. Wright, *Numerical optimization*, Springer, 2nd ed., 2006.

[45] N. Petra, J. Martin, G. Stadler, and O. Ghattas, *A computational framework for infinite-dimensional Bayesian inverse problems Part II: Stochastic Newton MCMC with application to ice sheet flow inverse problems*, SIAM Journal on Scientific Computing, 36 (2014), pp. A1525–A1555.

[46] T. Polzin, *Large Deformation Diffeomorphic Metric Mappings–Theory, Numerics, and Applications*, PhD thesis, University of Lübeck, 2018.

[47] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, 2nd ed., 2003.

[48] V. Simoncini, *Reduced order solution of structured linear systems arising in certain pde-constrained optimization problems*, Computational Optimization and Applications, 53 (2012), pp. 591–617.

[49] A. Sotiras, C. Davatzikos, and N. Paragios, *Deformable medical image registration: A survey*, IEEE Transactions on Medical Imaging, 32 (2013), pp. 1153–1190.

[50] A. Staniforth and J. Côté, *Semi-Lagrangian integration schemes for atmospheric models–a review*, Monthly Weather Review, 119 (1991), pp. 2206–2223.

[51] G. W. Stewart, *A Krylov–Schur algorithm for large eigenproblems*, SIAM Journal on Matrix Analysis and Applications, 23 (2002), pp. 601–614.

[52] A. Trouvé, *An infinite dimensional group approach for physics based models in pattern recognition*, tech. rep., Laboratoir d'Analyse Numerique CNRS URA, Universite Paris, 1995.

[53] ——, *Diffeomorphisms groups and pattern matching in image analysis*, International Journal of Computer Vision, 28 (1998), pp. 213–221.

[54] F.-X. Vialard, L. Risser, D. Rueckert, and C. J. Cotter, *Diffeomorphic 3D image registration via geodesic shooting using an efficient adjoint calculation*, International Journal of Computer Vision, 97 (2012), pp. 229–241.

[55] X. Yang and M. Niethammer, *Uncertainty quantification for LDDMM using low-rank Hessian approximation*, in Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, Springer, 2015, pp. 289–296.

[56] L. Younes, *Shapes and diffeomorphisms*, Springer Science & Business Media, 2010.

[57] M. Zhang and P. T. Fletcher, *Finite-dimensional Lie algebras for fast diffeomorphic image registration*, in Information Processing in Medical Imaging, vol. 25, 2015, pp. 249–260.

[58] ——, *Fast diffeomorphic image registration via fourier-approximated lie algebras*, International Journal of Computer Vision, 127 (2019), pp. 61–73.

[59] M. Zhang, N. Singh, and P. T. Fletcher, *Bayesian estimation of regularization and atlas building in diffeomorphic image registration*, in Proc Information Processing in Medical Imaging, Springer, 2013, pp. 37–48.

Declaration