Computing the Geodesic Interpolating Spline

Mills, Anna and Marsland, Stephen and Shardlow, Tony

2006

MIMS EPrint: **2006.4**

Manchester Institute for Mathematical Sciences
School of Mathematics

The University of Manchester

Reports available from: http://eprints.maths.manchester.ac.uk/
And by contacting: The MIMS Secretary
School of Mathematics
The University of Manchester
Manchester, M13 9PL, UK

ISSN 1749-9097

Computing the Geodesic Interpolating Spline

Anna Mills* Stephen Marsland† Tony Shardlow‡

January 5, 2006

Abstract

We examine non-rigid image registration by knotpoint matching. We consider registering two images, each with a set of knotpoints marked, where one of the images is to be registered to the other by a nonlinear warp so that the knotpoints on the *template* image are exactly aligned with the corresponding knotpoints on the *reference* image.

We explore two approaches for computing the registration by the Geodesic Interpolating Spline. First, we describe a method which exploits the structure of the problem in order to permit efficient optimization and second, we outline an approach using the framework of classical mechanics.

1 Introduction and Formulation of problem

Image registration is an important problem, arising, for instance, in medical image analysis and in cartography. The principle of image registration is to transform one image, the *template* image, to increase the similarity with a second *reference* image by rigid and non-rigid changes of the coordinate system. In this paper, we concentrate on the non-rigid registration problem, using knotpoints on the images to identify common features to be aligned by the registration. A comprehensive survey of registration methods is given in [7]. We develop methods for computation of the Geodesic Interpolating Spline (GIS) as described by Marsland and Twining in [6]. The GIS was developed from the thin-plate spline [3] to provide a diffeomorphic mapping between images so that no folds or tears are introduced to the images and no information is lost from the image being mapped.

The GIS uses, instead of one large step as in the thin-plate spline, a succession of small diffeomorphic steps to warp the image. To achieve this, a dependence on time is introduced [3]. The warp is characterized by a vector field $\mathbf{v}(\mathbf{x},t)$, selected to minimize a measure of deformation of the image. Then we define a warp, Φ so that $\Phi(\mathbf{P}) = \mathbf{Q} = \mathbf{x}(1)$ and $\mathbf{x}$ is the solution of

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{v}(t,\mathbf{x}(t)), \qquad 0 \le t \le 1, \qquad \mathbf{x}(0) = \mathbf{P}, \tag{1}$$

where $\mathbf{P}$ and $\mathbf{Q}$ are points on, respectively the template and reference images to be aligned.

We work with images in $\mathbb{R}^d$ where knotpoints $\mathbf{P}_i$ on the template image and $\mathbf{Q}_i$ on the reference image are given for $i = 1, \ldots, n_c$. We formulate this precisely as a minimization problem as follows. Minimize

$$l(\mathbf{q}_i(t), \mathbf{v}(t,\mathbf{x})) = \frac{1}{2}\int_0^1 \int_B \|L\mathbf{v}(t,\mathbf{x})\|^2_{\mathbb{R}^d} \mathrm{d}\mathbf{x}\,\mathrm{d}t, \tag{2}$$

over deformation fields, $\mathbf{v}(t,\mathbf{x}) \in \mathbb{R}^d$ and paths, $\mathbf{q}_i(t) \in B$ for $i = 1, \ldots, n_c$, where $B \subset \mathbb{R}^d$ is the domain of the image and L is a constant-coefficient, differential operator. We have constraints

$$\frac{\mathrm{d}\mathbf{q}_i}{\mathrm{d}t} = \mathbf{v}(t,\mathbf{q}_i(t)), \qquad 0 \le t \le 1, \tag{3a}$$

*School of Mathematics, The University of Manchester, Sackville Street, Manchester, M60 1QD, UK. amills@maths.man.ac.uk

†Institute of Information Sciences and Technology Massey University Private Bag 11222 Palmerston North New Zealand s.r.marsland@massey.ac.nz

‡School of Mathematics, The University of Manchester, Sackville Street, Manchester, M60 1QD, UK. shardlow@maths.man.ac.uk

and

$$\mathbf{q}_i(0) = \mathbf{P}_i, \quad \text{and} \quad \mathbf{q}_i(1) = \mathbf{Q}_i, \qquad\qquad i = 1, \ldots, n_c. \tag{3b}$$

In our case an appropriate choice for the operator, L, is to have $L = \nabla^2$ with zero Dirichlet and Neumann boundary conditions on B imposed, as this approximates the *Willmore energy* which quantifies the "bending energy" of the warp. One alternative approach would be to use a regularizer based on elasticity as discussed in [7]. We choose B to be the unit ball so we can have explicit Green's functions. This is an appropriate choice for B in analyzing brain images since brain images scale naturally to fit in the unit ball.

If we have large displacements of the knotpoints, a thin-plate spline mapping may not give a diffeomorphism. The techniques of the thin-plate spline, the clamped-plate spline and the Geodesic Interpolating Spline are all applied to the same problem in Figure 1. The clamped-plate spline is equivalent to the thin-plate spline with boundary conditions on B. Notice the folding in the upper right-hand quadrant of the mapping calculated by the clamped-plate technique. The GIS gives a diffeomorphic mapping so there is no folding.

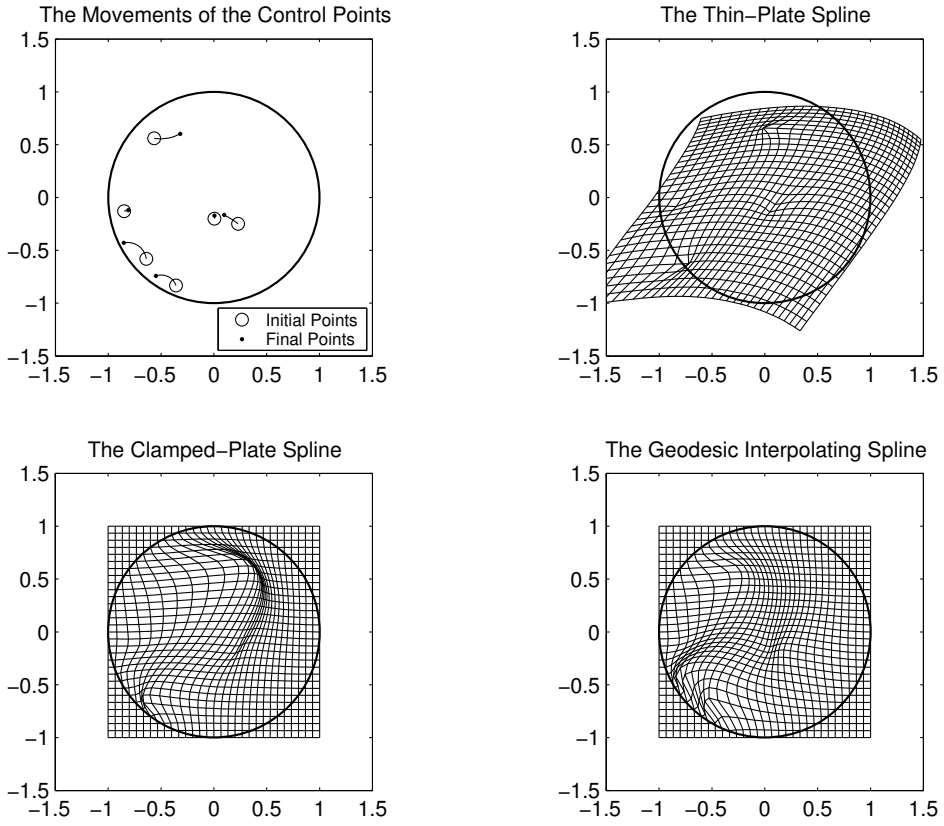

Figure 1: A system of 6 control points subjected to displacements, and the corresponding thin-plate, clamped-plate and geodesic interpolating splines

Using techniques in [4], it can be shown that we can represent the minimizing vector field as

$$\mathbf{v}(t, \mathbf{x}) = \sum_{i=1}^{n_c} \boldsymbol{\alpha}_i(t) G(\mathbf{q}_i(t), \mathbf{x}),$$

where $\boldsymbol{\alpha}_i(t), \mathbf{q}_i(t)$ are respectively multipliers and knotpoint positions at time t for a set of n_c knotpoints and where $G(\mathbf{x}, \mathbf{y})$ is the Green's function derived by Boggio [2]. We will do numerical experiments with the 2-dimensional Green's function

$$G(\mathbf{x}, \mathbf{y}) = -|\mathbf{x} - \mathbf{y}|^2 \ln |\mathbf{x} - \mathbf{y}|^2$$

for the biharmonic operator L with zero Dirichlet and Neumann boundary conditions on B. In this way, we can derive the optimisation problem:

$$\min \int_0^1 \frac{1}{2} \sum_{i,j=1}^{n_c} \boldsymbol{\alpha}_i^\top \boldsymbol{\alpha}_j G(\mathbf{q}_i, \mathbf{q}_j) \mathrm{d}t \tag{4a}$$

such that

$$\frac{\mathrm{d}\mathbf{q}_i}{\mathrm{d}t} = \sum_{j=1}^{n_c} \boldsymbol{\alpha}_j G(\mathbf{q}_i, \mathbf{q}_j), \qquad \mathbf{q}_i(0) = \mathbf{P}_i \qquad i = 1, \ldots, n_c, \tag{4b}$$

$$\text{and } \mathbf{q}_i(1) = \mathbf{Q}_i, \tag{4c}$$

where (4b) gives the velocity constraint.

We explore two methods for the minimization. In Section 2, we examine the structure of the discretized version of (4), and use an optimization method exploiting this structure. In Section 3, we reformulate the problem in a Hamiltonian framework to compute the GIS. In Section 4, we test the methods on brain images with knotpoints marked by clinicians. In Section 5, we summarize our results.

2 Numerical Optimization Exploiting Partial Separability

To find the minimizer of (4), we use numerical optimization techniques. First, we discretize in time to achieve a finite dimensional system, by applying forward Euler to the velocity constraint and the rectangle rule to the objective function. This yields a constrained optimization problem with a clear structure. The numerical optimization package Galahad uses a concept called group partial separability to express the dependence between different variables and make the structure of the resulting Hessian matrices clear and usable by linear algebra routines. This can markedly improve the performance on large scale problems. The discretization of the GIS is partially separable, because of the dependence on time, and we exploit this in our Galahad implementation.

We move to a discretized version of the problem using time step $\Delta t = 1/N$. We use $\boldsymbol{\alpha}_i^n \approx \boldsymbol{\alpha}_i(n\Delta t), n = 0, \ldots, N-1$ and $\mathbf{q}_i^n \approx \mathbf{q}_i(n\Delta t), n = 0, \ldots, N$ to give the problem in the following form.
Minimize

$$l\{\boldsymbol{\alpha}_i^n, \mathbf{q}_i^n\} = \frac{1}{2} \sum_{i,j=1}^{n_c} \sum_{n=0}^{N-1} \boldsymbol{\alpha}_i^{n\top} \boldsymbol{\alpha}_j^n G(\mathbf{q}_i^n, \mathbf{q}_j^n) \tag{5}$$

over $\boldsymbol{\alpha}_i^n, \mathbf{q}_i^n \;\; i = 1, \ldots, n_c$ such that

$$N(\mathbf{q}_i^{n+1} - \mathbf{q}_i^n) = \sum_{j=1}^{n_c} \boldsymbol{\alpha}_j^n G(\mathbf{q}_i^n, \mathbf{q}_j^n), \qquad n = 0, \ldots, N-1, \; i = 1, \ldots, n_c \tag{6a}$$

with conditions

$$\mathbf{q}_i^0 = \mathbf{P}_i, \;\; \text{and} \;\; \mathbf{q}_i^N = \mathbf{Q}_i \qquad\qquad i = 1, \ldots, n_c, \tag{6b}$$

where $G(\mathbf{x}, \mathbf{y})$ is the clamped-plate biharmonic Green's function.

We can take advantage of the partial separable structure of the problem using the routine Lancelot B from the optimization suite, *Galahad* [5]. Lancelot B is a Fortran 90 optimization routine for minimizing a group partial separable, constrained problem. Our problem is group partial separable since every partial separable problem is naturally group partial separable. As detailed in [5], Lancelot B uses an iterative method. Outer iterations form an augmented Lagrangian merit functions, and inner iterations minimise a quadratic model of the merit function. In Lancelot B, the objective function and constraints can be described in terms of a set of group functions and element functions, where each element function involves only a small subset of the minimization variables. Specifically, Lancelot B solves problems with objective functions of the form (where we omit terms not relevant to our problem)

$$f(\mathbf{x}) = \sum_{i \in \Gamma_0} w_i^g g_i \left(\sum_{j \in \mathcal{E}_i} w_{ij}^e e_j(\mathbf{x}_j^e) \right), \qquad \mathbf{x} = (x_1, x_2, \ldots, x_n)^\top. \tag{7}$$

In the above, Γ_0 is a set of indices of group functions g_i, $\mathcal{E}_i$ is a set of nonlinear element functions e_j, and w_i^g and w_{ij}^e describe the element and group weight parameters. The constraints must be of the form

$$c_i(\mathbf{x}) = w_i^g g_i \left(\sum_{j \in \mathcal{E}_i} w_{ij}^e e_j(\mathbf{x}_j^e) + \mathbf{a}_i^N \mathbf{x} \right) = 0, \tag{8}$$

for i in the set of indices of constraints Γ_c.

Examining our objective function, we see that there is a natural division into N groups, each group being given by, for the n^{th} group

$$w_i^g g_i \left(\sum_{j \in \mathcal{E}_n} w_{nj}^e e_j(\mathbf{x}_j^e) \right) = \frac{1}{2} \sum_{i,j=1}^{n_c} (\boldsymbol{\alpha}_i^n G(\mathbf{q}_i^n, \mathbf{q}_j^n)) \boldsymbol{\alpha}_j^n, \tag{9}$$

each of which contains n_c^2 nonlinear element functions. In this notation, $\mathbf{x}_j^e$ is the vector containing the optimization variables $(\mathbf{q}_i^n, \boldsymbol{\alpha}_i^n), i = 1, \ldots, n_c$. Similarly, the $2n_cN$ velocity constraints in Equation (6a) can be characterized by $2n_cN$ groups each comprising n_c nonlinear elements and one linear element. We require that the start and end points of each control point path coincide with the landmarks on, respectively, the floating and reference images. This gives $4n_c$ constraints of the form

$$0 = (\mathbf{q}_i^1 - \mathbf{P}_i)w, \qquad 0 = (\mathbf{q}_i^{N+1} - \mathbf{Q}_i)w, \qquad i = 1, \ldots, n_c. \tag{10}$$

Each dimension of these $2n_c$ constraints consists of one linear element and one constant. Hence we have $4n_c$ groups characterizing the landmark constraints, each group being weighted by some $w \gg 1$. We have $N + 2n_cN + 4n_c$ groups characterizing the problem and within these groups, we have in total $3n_c^2N$ nonlinear elements.

The group partial separability of the problem introduces sparsity to the Hessian. We can see the block diagonal sparsity structure of the Hessian for the augmented Lagrangian function for our problem in Figure 2 where the Hessian for a problem involving 5 time steps and 8 knotpoints is shown. The Hessian was calculated using a numerical scheme on the augmented Lagrangian function as used in Lancelot B, given by

$$\mathcal{L}_A(\mathbf{x}, \boldsymbol{\lambda}; \mu) = f(\mathbf{x}) - \sum_{i \in \Gamma_c} \lambda_i c_i(\mathbf{x}) + \frac{1}{2\mu} \sum_{i \in \Gamma_c} c_i^2(\mathbf{x}), \tag{11}$$

where f is the constrained objective function, μ is the penalty parameter, Γ_c is the set of indices of the equality constraints, and λ_i for $i \in \Gamma_c$ are Lagrangian multipliers. Lancelot B uses Newton methods which require Hessian approximations. The routine exploits the group partial separability of the problem to make the calculation and storage of the Hessian approximations more efficient.

The Hessian of the augmented Lagrangian has structure

$$\begin{bmatrix} \mathcal{B} & \mathcal{A}^\top \\ \mathcal{A} & 0 \end{bmatrix}, \tag{12}$$

where $\mathcal{B}$ is the $\partial^2/\partial \mathbf{x}^2$ derivative, the matrix $\mathcal{A}$ is the $\partial^2/\partial \mathbf{x} \partial \boldsymbol{\lambda}$ derivative, and the zero block occurs since $\boldsymbol{\lambda}$ only appears in the augmented Lagrangian as order one. This structure is clearly visible in Figure 2. There are 5 time blocks on the main diagonal, each with coupling to the adjacent time blocks. There is a banded off-diagonal structure, the two bands of $\mathcal{A}$ being due to the constraints being divided into x-dimension and y-dimension constraints.

3 Classical Mechanics Approach

We present a novel formulation of the Geodesic Interpolating Spline for image registration as a problem in Hamiltonian dynamics. The Geodesic Interpolating Spline problem is given by the minimization problem (4). In the language of classical mechanics, we are minimising the following Lagrangian

$$L(\mathbf{q}, \dot{\mathbf{q}}) = \frac{1}{2} \sum_{i,j=1}^{n_c} \boldsymbol{\alpha}_i^\top \boldsymbol{\alpha}_j G(\mathbf{q}_i, \mathbf{q}_j), \tag{13}$$

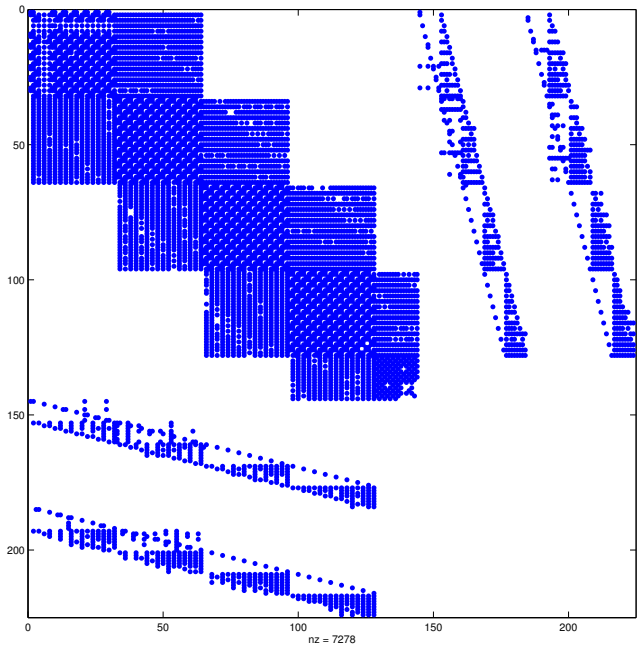

Figure 2: The Sparsity Structure of the Hessian of the Augmented Lagrangian Function for a Problem with 5 Time Steps and 8 Knotpoints

where the positions $\mathbf{q} = (\mathbf{q}_1, \ldots, \mathbf{q}_{n_c})$ and velocities $\dot{\mathbf{q}} = (\frac{\mathrm{d}\mathbf{q}_1}{\mathrm{d}t}, \ldots, \frac{\mathrm{d}\mathbf{q}_{n_c}}{\mathrm{d}t})$ are defined by (4b). Following Arnold [1], we can convert to a Hamiltonian formulation by computing the Legendre transform of the Lagrangian function as a function of the velocity, $\dot{\mathbf{q}}$: the Hamiltonian

$$H(\mathbf{p}, \mathbf{q}) = \mathbf{p}^\top \dot{\mathbf{q}} - L(\mathbf{q}, \dot{\mathbf{q}}), \tag{14}$$

where $\mathbf{p} = \partial L / \partial \mathbf{q}$ is the generalized momentum. The total differential of the Hamiltonian is

$$\mathrm{d}H = \frac{\partial H}{\partial \mathbf{p}} \mathrm{d}\mathbf{p} + \frac{\partial H}{\partial \mathbf{q}} \mathrm{d}\mathbf{q}, \tag{15}$$

where we abbreviate $H(\boldsymbol{\alpha}, \mathbf{q})$ to H. Examining the total differential of the right-hand side of (14) we have

$$\mathrm{d}H = \dot{\mathbf{q}} \mathrm{d}\mathbf{p} - \frac{\partial L}{\partial \mathbf{q}} \mathrm{d}\mathbf{q}. \tag{16}$$

We know that equations (15) and (16) must be equal so we see that

$$\frac{\partial H}{\partial \mathbf{p}} = \dot{\mathbf{q}}, \tag{17}$$

$$\frac{\partial H}{\partial \mathbf{q}} = -\frac{\partial L}{\partial \mathbf{q}}. \tag{18}$$

Lemma The generalized momentum

$$\frac{\partial L}{\partial \dot{\mathbf{q}}} \equiv \mathbf{p} = \boldsymbol{\alpha}. \tag{19}$$

Proof. Construct a matrix $\mathcal{A} \in \mathbb{R}^{n_c \times n_c}$ so that

$$\mathcal{A}_{ij} = G(\mathbf{q}_i, \mathbf{q}_j) \qquad i, j = 1, \ldots, n_c, \tag{20}$$

and a matrix $\mathcal{G} \in \mathbb{R}^{dn_c \times dn_c}$ as

$$\mathcal{G} = \mathcal{A} \otimes I,$$

(where I is the d-dimensional identity matrix). We can use equation (4b) and define a vector $\boldsymbol{\alpha} = [\boldsymbol{\alpha}_1^\top, \boldsymbol{\alpha}_2^\top, \ldots, \boldsymbol{\alpha}_{n_c}^\top]^\top$ to derive an equation for the velocity constraint in matrix form,

$$\frac{\mathrm{d}\mathbf{q}}{\mathrm{d}t} = \mathcal{G}\boldsymbol{\alpha}. \tag{21}$$

Similarly, we can express the Lagrangian (13) in matrix form as

$$L = \frac{1}{2}\boldsymbol{\alpha}^\top \mathcal{G}\boldsymbol{\alpha}.$$

We calculate

$$\frac{\partial L}{\partial \dot{\mathbf{q}}} = \frac{\partial L}{\partial \boldsymbol{\alpha}} \frac{\partial \boldsymbol{\alpha}}{\partial \dot{\mathbf{q}}}. \tag{22}$$

Examining the terms on the right hand side of equation (22),we have

$$\frac{\partial L}{\partial \boldsymbol{\alpha}} = \mathcal{G}\boldsymbol{\alpha},$$

and, from (21),

$$\frac{\partial \dot{\mathbf{q}}}{\partial \boldsymbol{\alpha}} = \mathcal{G}, \qquad \frac{\partial \boldsymbol{\alpha}}{\partial \dot{\mathbf{q}}} = \mathcal{G}^{-1},$$

where $\mathcal{G}$ is known to be positive definite, and hence invertible, since the energy in the problem is always positive for non-trivial mappings. Hence we see that the generalized momentum is given by,

$$\frac{\partial L}{\partial \dot{\mathbf{q}}} = \mathcal{G}^{-1}\mathcal{G}\boldsymbol{\alpha} = \boldsymbol{\alpha}.$$

□

We have the Euler-Lagrange equations governing the system [1]

$$\frac{\partial L}{\partial \mathbf{q}} - \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{\mathbf{q}}}\right) = 0. \tag{23}$$

Substituting the expression for generalized momentum given in equation (19) into the Euler-Lagrange equations (23) gives

$$\frac{\partial L}{\partial \mathbf{q}} = \frac{\mathrm{d}\boldsymbol{\alpha}}{\mathrm{d}t}.$$

Hence, with equation (18), we have the coupled system of Hamiltonian equations

$$\dot{\mathbf{q}} = \frac{\partial H}{\partial \boldsymbol{\alpha}}, \tag{24a}$$

$$\dot{\boldsymbol{\alpha}} = -\frac{\partial H}{\partial \mathbf{q}}. \tag{24b}$$

Now we examine how the Geodesic Interpolating Spline minimization problem relates to the Hamiltonian system. Equation (14) simplifies as follows. Substituting the first derivatives (21) into (17), the first term of the right hand side of (14) becomes

$$\boldsymbol{\alpha}^\top \dot{\mathbf{q}} = \boldsymbol{\alpha}^\top \boldsymbol{\alpha} \mathcal{G}. \tag{25}$$

Hence we have

$$H(\boldsymbol{\alpha}, \mathbf{q}) = \boldsymbol{\alpha}^\top \dot{\mathbf{q}} - L(\mathbf{q}, \dot{\mathbf{q}}) \tag{26}$$

$$= \sum_{i,j=1}^{n_c} \boldsymbol{\alpha}_i^\top \boldsymbol{\alpha}_j G(\mathbf{q}_i, \mathbf{q}_j) - \frac{1}{2}\sum_{i,j=1}^{n_c} \boldsymbol{\alpha}_i^\top \boldsymbol{\alpha}_j G(\mathbf{q}_i, \mathbf{q}_j) \tag{27}$$

$$= \frac{1}{2}\sum_{i,j=1}^{n_c} \boldsymbol{\alpha}_i^\top \boldsymbol{\alpha}_j G(\mathbf{q}_i, \mathbf{q}_j), \tag{28}$$

so that H is now a function of $\boldsymbol{\alpha}$ and $\mathbf{q}$.

We have shown that the solutions, $\mathbf{q}_i(t)$ and $\boldsymbol{\alpha}_i(t)$ of (4) are solutions of the system of differential equations, (24). We consider the Hamiltonian equations (24) with given initial data

$$\begin{bmatrix} \mathbf{q}(0) \\ \boldsymbol{\alpha}(0) \end{bmatrix} = \begin{bmatrix} \mathbf{P} \\ \mathbf{A} \end{bmatrix} = \mathbf{Y},$$

where $\mathbf{P}$ is the vector of initial knotpoint positions in Equation (4b) and $\mathbf{A}$ is the initial vector of generalized momentum. We solve the nonlinear system of equations $\Phi(\mathbf{A};\mathbf{P}) = \mathbf{Q}$ for $\mathbf{A}$ as a shooting problem, where $\mathbf{Q}$ is the vector of final knotpoint positions in (4c) and $\Phi(\mathbf{A};\mathbf{P}) := \mathbf{q}(1)$, the position component of the solution of the following Hamiltonian system:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{q}_i = \sum_{j=1}^{n_c} \boldsymbol{\alpha}_j G(\mathbf{q}_i, \mathbf{q}_j) \tag{29a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\alpha}_i = -\sum_{j=1}^{n_c} \boldsymbol{\alpha}_i^\top \boldsymbol{\alpha}_j \frac{\partial}{\partial \mathbf{q}_j} G(\mathbf{q}_i, \mathbf{q}_j) \qquad i = 1, \ldots, n_c, \tag{29b}$$

with initial conditions

$$\begin{bmatrix} \mathbf{q}(0) \\ \boldsymbol{\alpha}(0) \end{bmatrix} = \begin{bmatrix} \mathbf{P} \\ \mathbf{A} \end{bmatrix} = \mathbf{Y}. \tag{29c}$$

3.1 Numerical Implementation

To solve the coupled system of differential equations (29), we discretize in time. We choose to discretize using the Forward Euler method. Experiments with symplectic methods have shown no advantage for this problem, principally because it is a boundary value problem where long time simulations are not of interest, and no suitable explicit symplectic integrators are available. Using the notation $\mathbf{q}_i^n \approx \mathbf{q}_i(n\Delta t)$, $\boldsymbol{\alpha}_i^n \approx \boldsymbol{\alpha}_i(n\Delta t),\ n = 0, \ldots, N,\ \Delta t = 1/N$, we have

$$\begin{bmatrix} \mathbf{q}^{n+1} \\ \boldsymbol{\alpha}^{n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{q}^{n} \\ \boldsymbol{\alpha}^{n} \end{bmatrix} + \Delta t \begin{bmatrix} \dfrac{H(\mathbf{q}^n, \boldsymbol{\alpha}^n)}{\partial \boldsymbol{\alpha}} \\ -\dfrac{H(\mathbf{q}^n, \boldsymbol{\alpha}^n)}{\partial \mathbf{q}} \end{bmatrix}, \tag{30}$$

with initial conditions

$$\begin{bmatrix} \mathbf{P} \\ \mathbf{A} \end{bmatrix} = \begin{bmatrix} \mathbf{q}^0 \\ \boldsymbol{\alpha}^0 \end{bmatrix} = \mathbf{Y}. \tag{31}$$

We wish to examine the variation with respect to the initial momentum, $\mathbf{A}$, in order to provide Jacobians for the nonlinear solver, so we need to approximate

$$\frac{\mathrm{d}\mathbf{q}(1)}{\mathrm{d}\mathbf{A}} \approx \frac{\mathrm{d}\mathbf{q}^N}{\mathrm{d}\boldsymbol{\alpha}^0}. \tag{32}$$

We can neglect the variation with respect to $\mathbf{q}^0$, the initial positions, since the initial positions, $\mathbf{P}$, remain fixed.

Using the Forward Euler scheme for some function $\mathbf{f}$, we have

$$\mathbf{X}^{n+1} = \mathbf{X}^n + \Delta t \mathbf{f}(\mathbf{X}^n) \tag{33}$$

with initial condition $\mathbf{X}^0 = \mathbf{Y}$. Let J^n denote the Jacobian $\frac{\mathrm{d}\mathbf{X}^n}{\mathrm{d}\mathbf{A}}$. Differentiating (33) with respect to the initial condition, we have

$$\frac{\mathrm{d}\mathbf{X}^{n+1}}{\mathrm{d}\mathbf{A}} = \frac{\mathrm{d}\mathbf{X}^n}{\mathrm{d}\mathbf{A}} + \Delta t \frac{\mathrm{d}\mathbf{f}(\mathbf{X}^n)}{\mathrm{d}\mathbf{X}^n}\frac{\mathrm{d}\mathbf{X}^n}{\mathrm{d}\mathbf{A}}, \qquad \frac{\mathrm{d}\mathbf{X}^0}{\mathrm{d}\mathbf{A}} = \begin{pmatrix} 0 \\ I \end{pmatrix}. \tag{34}$$

Then we can solve numerically a coupled system of equations

$$J^{n+1} = J^n + \Delta t \frac{\mathrm{d}\mathbf{f}(\mathbf{X}^n)}{\mathrm{d}\mathbf{X}^n} J^n \tag{35}$$

$$\mathbf{X}^{n+1} = \mathbf{X}^n + \Delta t \mathbf{f}(\mathbf{X}^n) \tag{36}$$

with initial conditions

$$J^0 = \begin{bmatrix} 0 \\ I \end{bmatrix}, \qquad \mathbf{X}^0 = Y. \tag{37}$$

In our problem, we have

$$\mathbf{f}(\mathbf{X}) = \begin{bmatrix} \frac{\partial H}{\partial \mathbf{q}} \\ -\frac{\partial H}{\partial \boldsymbol{\alpha}} \end{bmatrix}, \qquad \mathbf{X} = \begin{bmatrix} \mathbf{q} \\ \boldsymbol{\alpha} \end{bmatrix}$$

and so

$$\frac{\mathrm{d}\mathbf{f}(\mathbf{X}^n)}{\mathrm{d}\mathbf{X}^n} = \begin{bmatrix} \frac{\partial^2 H}{\partial \mathbf{q} \partial \boldsymbol{\alpha}} & \frac{\partial^2 H}{\partial \boldsymbol{\alpha}^2} \\ -\frac{\partial^2 H}{\partial \mathbf{q}^2} & -\frac{\partial^2 H}{\partial \boldsymbol{\alpha} \partial \mathbf{q}} \end{bmatrix}. \tag{38}$$

We calculate the entries of (38). We have

$$\begin{aligned} H(\boldsymbol{\alpha}, \mathbf{q}) &= \frac{1}{2} \sum_{i,j=1}^{n_c} \boldsymbol{\alpha}_i^\top \boldsymbol{\alpha}_j G(\mathbf{q}_i, \mathbf{q}_j) \\ &= \sum_{i<j} \boldsymbol{\alpha}_i^\top \boldsymbol{\alpha}_j G(\mathbf{q}_i, \mathbf{q}_j) + \frac{1}{2} \sum_{i=1}^{n_c} \boldsymbol{\alpha}_i^\top \boldsymbol{\alpha}_i G(\mathbf{q}_i, \mathbf{q}_i), \end{aligned}$$

where we use the notation $\sum_{i<j}$ to denote summation from 1 to n_c such that $i < j$. We obtain first order derivatives

$$\frac{\partial H}{\partial \boldsymbol{\alpha}_k} = \sum_{j=1}^{n_c} \boldsymbol{\alpha}_j G(\mathbf{q}_k, \mathbf{q}_j) \tag{39}$$

$$\frac{\partial H}{\partial \mathbf{q}_k} = \sum_{i \neq k} \boldsymbol{\alpha}_i^\top \boldsymbol{\alpha}_k \frac{\partial G(\mathbf{q}_i, \mathbf{q}_k)}{\partial \mathbf{q}_k} + \frac{1}{2} \boldsymbol{\alpha}_k^\top \boldsymbol{\alpha}_k \frac{\partial G(\mathbf{q}_k, \mathbf{q}_k)}{\partial \mathbf{q}_k} \tag{40}$$

Then we can calculate second derivatives

$$\frac{\partial^2 H}{\partial \boldsymbol{\alpha}_k \partial \boldsymbol{\alpha}_l} = G(\mathbf{q}_k, \mathbf{q}_l) I \tag{41}$$

$$\frac{\partial^2 H}{\partial \boldsymbol{\alpha}_k \partial \mathbf{q}_l} = \boldsymbol{\alpha}_l \frac{\partial G(\mathbf{q}_k, \mathbf{q}_l)}{\partial \mathbf{q}_l}, \tag{42}$$

$$\frac{\partial^2 H}{\partial \boldsymbol{\alpha}_k \partial \mathbf{q}_k} = \sum_{j=1}^{n_c} \boldsymbol{\alpha}_j \frac{\partial G(\mathbf{q}_k, \mathbf{q}_j)}{\partial \mathbf{q}_k}, \tag{43}$$

$$\frac{\partial^2 H}{\partial \mathbf{q}_k \partial \mathbf{q}_l} = \boldsymbol{\alpha}_k^\top \boldsymbol{\alpha}_l \frac{\partial^2 G(\mathbf{q}_k, \mathbf{q}_l)}{\partial \mathbf{q}_k \partial \mathbf{q}_l} \tag{44}$$

$$\frac{\partial^2 H}{\partial \mathbf{q}_k \partial \mathbf{q}_k} = \boldsymbol{\alpha}_k^\top \boldsymbol{\alpha}_k \frac{\partial^2 G(\mathbf{q}_k, \mathbf{q}_k)}{\partial \mathbf{q}_k \partial \mathbf{q}_k}, \tag{45}$$

(where I is the 2-dimensional identity matrix). The calculation of the Jacobian permits efficient solution of the nonlinear equation $\Phi(A; \mathbf{P}) = \mathbf{Q}$ using the NAG nonlinear solver `nag_nlin_sys_sol` [9].

4 Comparison of Techniques

4.1 Numerical Optimization Approach

The inner iterations of the optimization method of Lancelot B use a minimization of a quadratic model function for which an approximate minimum is found. This leads to a model reduction by solving one or more quadratic minimization problems, requiring a solution of a sequence of linear systems. The Lancelot optimization package allows a choice of linear solver from 12 available options [5]. Experimentation with the choice of linear solver for the problem shows that the choice of solver has a large effect on the performance of the optimization, the best choice of linear solver varying from problem to problem. The methods tested are described below.

CG - conjugate gradient method without preconditioning

DIAG - preconditioned gradient method with a diagonal preconditioner

EXP BAND - preconditioned conjugate gradient method with an expanding band incomplete Cholesky preconditioner

MUNKS - preconditioned conjugate gradient method with Munksgaard's preconditioner

SCHNABEL - preconditioned conjugate gradient method with a modified Cholesky preconditioner

GMPS - preconditioned conjugate gradient method with a modified Cholesky preconditioner

BAND - preconditioned conjugate gradient method with a band preconditioner, semi-bandwidth 7

LIN-MORE - preconditioned conjugate gradient method with an incomplete Cholesky factorization preconditioner

MFRONTAL - a multifrontal factorization method

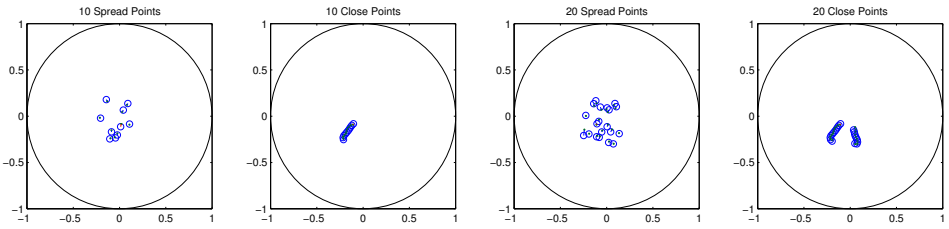

Figure 3: Test cases for testing the linear solvers showing $\mathbf{P}_i$ as circles and $\mathbf{Q}_i$ as points

The implementation was initially tested on four selected test cases, as illustrated in Figure 3, two comprising knotpoints taken from adjacent points in a data-set of 123 knotpoints hand-annotated on an MRI brain image, and two comprising points taken equally spaced throughout the data-set. The experiments used 10 time steps. It turns out that those involving adjacent points are harder to solve.

We see the results of the experiment in Table 1 and Table 2. We notice that for most methods, the problems with the points closer together in the domain are hardest to solve. We see that this is not always true, however. For instance, the diagonal method performs better with 10 close points than with 10 points spread through the domain. This is also true of the method using the Munksgaard preconditioner. The expanding band incomplete Cholesky preconditioner performs the best over the four test cases. It is clear, however, from these experiments that it is difficult to predict how an individual linear solver will perform on a particular problem. None of the linear solvers resulted in convergence for the case involving all of the 123 knotpoints.

Solver	10 Spread	10 Close	20 Spread	20 Close	Total
CG	20	23	29	34	106
DIAG	222	162	474	675	1533
EXP BAND	13	14	29	18	74
MUNKS	65	28	46	46	185
SCHNABEL	25	25	106	64	220
GMPS	14	17	78	105	214
BAND	21	24	114	98	257
LIN-MORE	23	20	37	64	144
MFRONTAL	15	17	102	26	160

Table 1: Number of Iterations to Converge

Solver	10 Spread	10 Close	20 Spread	20 Close	Total
CG	5.78	4.86	54.04	71.07	135.74
DIAG	24.38	46.40	366.98	785.47	1223.23
EXP BAND	0.89	1.69	10.00	6.35	18.93
MUNKS	22.98	15.04	128.72	158.95	325.68
SCHNABEL	5.68	5.07	192.81	155.38	358.93
GMPS	1.34	2.01	62.35	96.76	162.46
BAND	5.55	8.70	220.94	227.10	462.28
LIN-MORE	4.53	4.62	49.26	89.20	147.61
MFRONTAL	1.42	2.09	76.07	21.00	100.58

Table 2: Time Taken to Converge (in seconds)

In order to understand this behaviour better, we explore the change in condition number of the interpolation matrix with respect to the minimum separation between knotpoints. First, we examine the interpolation matrix for the clamped-plate spline. Computing the clamped-plate spline requires solving a linear system involving an interpolation matrix. Our $n_c \times n_c$ interpolation matrix, G, is constructed of biharmonic Green's functions in the same manner as that in which we constructed matrix $\mathcal{A}$ in (20), namely

$$\mathrm{G}_{ij} = G(\mathbf{q}_i, \mathbf{q}_j),$$

where $G(\cdot, \cdot)$ denotes the d-dimensional biharmonic Green's function and $\mathbf{q}_i, \mathbf{q}_j$ are d-dimensional knotpoint position vectors. This interpolation matrix is also a key feature of the Hessian of the augmented Lagrangian, as discussed in Section 2.

In Figure 4, we see the change in condition number for the interpolation matrix for two parallel knotpoint paths. The separation of two knotpoint start and finish points were varied from 0.002 to 0.0001 in steps of 0.00001, the condition number of the interpolation matrix being calculated at each point. From the literature [8], [10], we expect the condition number in the 2-norm, $\kappa_2(\mathrm{G})$ of the interpolation matrix to vary with the minimum separation of the knotpoints in a manner bounded above by $\alpha \min_{i,j} \|\mathbf{q}_i - \mathbf{q}_j\|^{-\beta}$, some $\alpha, \beta > 0$. Accordingly, for comparison, we calculate and plot

$$\alpha \min_{i,j} \|\mathbf{q}_i - \mathbf{q}_j\|^{-\beta},$$

for each test set of knotpoints, where α and β are calculated using polynomial fitting in MATLAB. Where the minimum separation distances are varied over the interval [0.002,0.0001], we see that α is calculated to be 0.1232 and β is calculated to be 1.837. The corresponding function is plotted in Figure 4 against the actual condition numbers. We see close correspondence between the condition numbers and their predicted bounds. Thus the condition number of the Hessian in the Lancelot B optimisation is expected to be large when points are close together; we suspect the poor performance of the linear solvers in the test cases is due to ill conditioning and for knotpoints with small separation.

4.2 Classical Mechanics Approach

Experiments show that the Hamiltonian implementation can solve all of the test cases illustrated in Figure 3 in less than one second, whereas the Lancelot B implementation takes over 18 seconds to solve the test

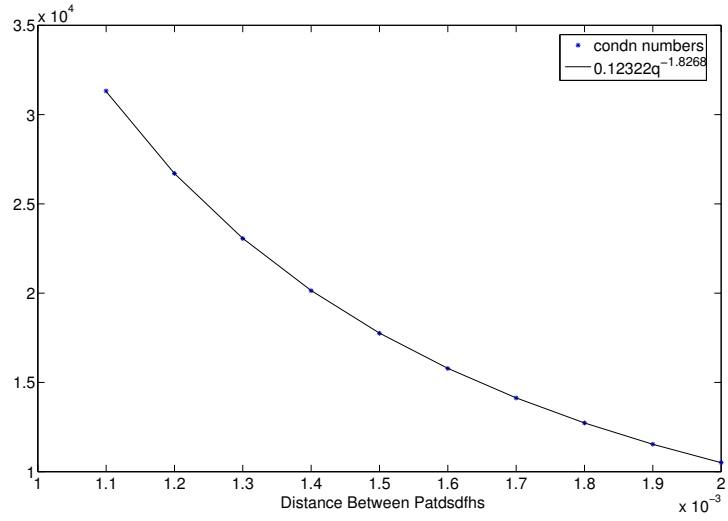

Figure 4: Condition Numbers for Knotpoints with Decreasing Separation

cases. An illustration of typical final knotpoint paths calculated by the classical mechanics approach is given in Figure 5 where paths for 62 points are shown. The hooked paths are typical of paths calculated by the Geodesic Interpolating Spline.

Experiments are presented which show the performance of the Hamiltonian method under increases in the number of knotpoints and in the number of time steps, and the effect of the inclusion of a user-supplied Jacobian. We see the effect of an increase in the number of time steps in Figure 6, comparing results using a numerical gradient with those using a true gradient. We see that the performance of the method using the true gradient is much superior to that using a numerical gradient, both in terms of function evaluations and of time. For these tests, the first 60 knot points of the 123 point set were used.

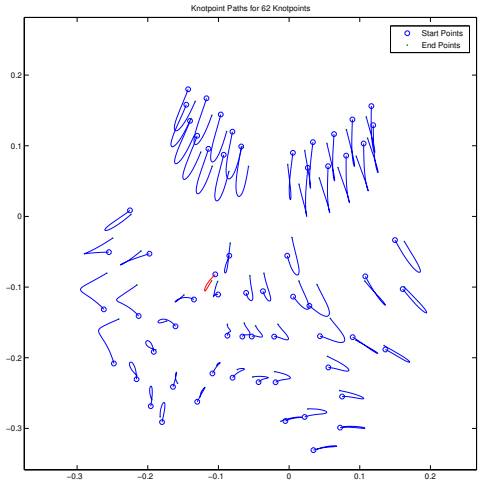

Figure 5: Knotpoint Paths for the Geodesic Interpolating Spline

In Figure 7, we see the effect on performance of the Hamiltonian implementation under an increase in the number of knotpoints, both with a numerical gradient and with a user-supplied analytic gradient, as described above. The knotpoints are taken as consecutive excerpts from the same set of 123 points as for the test sets above and 20 time steps are used. The speed of convergence of the method improves by a factor of approximately 4 when there is a user-supplied gradient. Notice that solving for the full 123 point

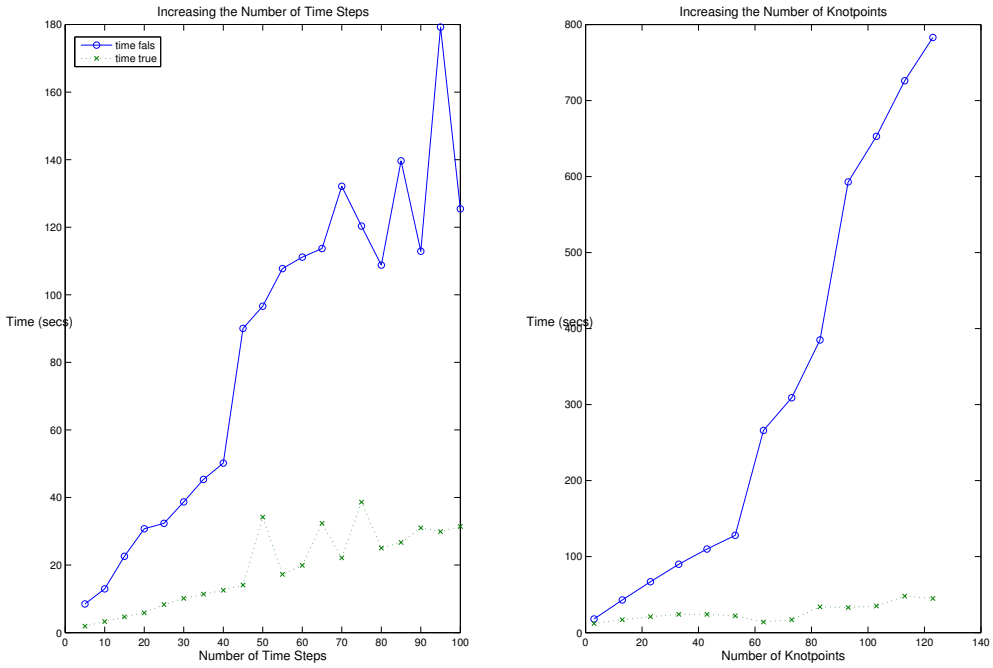

Figure 6: Increasing the number of time steps for the Hamiltonian method

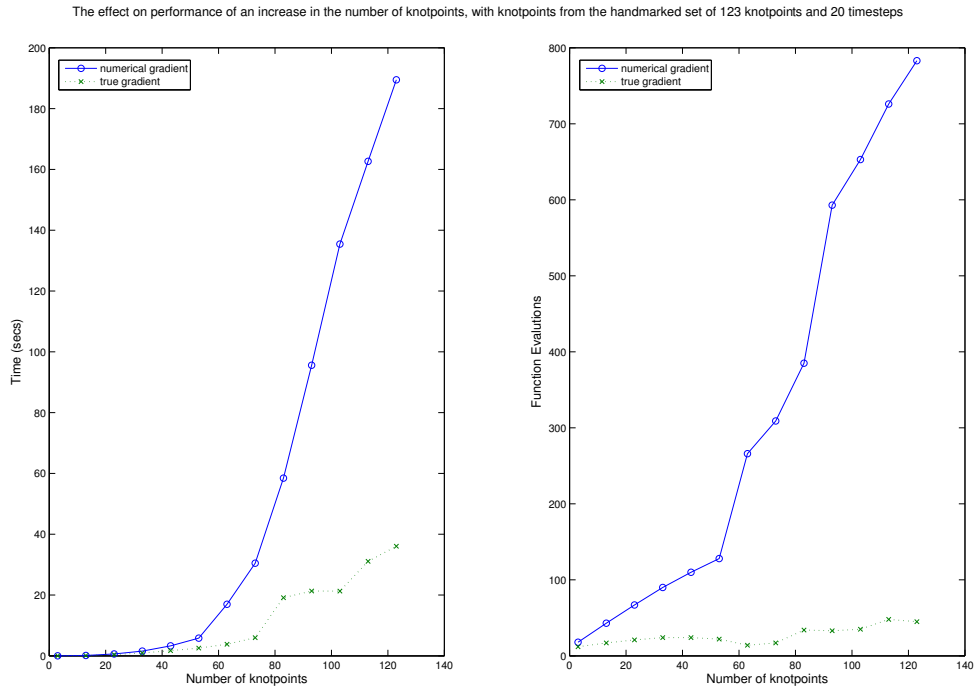

Figure 7: Increasing the number of knotpoints for the Hamiltonian method, $\Delta t = 1/20$

set takes less than 40 seconds using this method. The Hamiltonian method significantly outperforms all of the Lancelot B solvers (see Table 2) on the test problems, despite the results shown for the Hamiltonian method being calculated using twice as many time steps as those shown for the Lancelot B solvers.

5 Conclusions

The Lancelot B implementation was developed by the authors as an improvement to Marsland and Twining's original MATLAB method [6] for calculating the GIS and showed significant improvement over the MATLAB implementation. The Hamiltonian method shows an impressive improvement over both of these methods. We believe that the Lancelot B method shows disappointing performance due to the lack of a preconditioner suitable for the ill-conditioning of the interpolation matrix. The Hamiltonian method for computing the Geodesic Interpolating Spline dramatically outperforms the Lancelot B implementation over the test set of real data. It is clear that exact Jacobians should be supplied to the Hamiltonian implementation to give efficient performance. We see from the experiments carried out that, with exact second derivatives provided, the performance of the Hamiltonian method is superior to the performance of previous methods.

References

[1] V. I. Arnold, *Mathematical methods of classical mechanics*, vol. 60 of Graduate Texts in Mathematics, Springer Verlag, New York, 2 ed., 1989. 508 pages.

[2] T. Boggio, *Sulle funzioni di green d'ordine m.*, Circolo Matematico di Palermo, 20 (1905), pp. 97–135.

[3] V. Camion and L. Younes, *Geodesic interpolating splines*, in M.A.T. Figueiredo, J. Zerubia, A K. Jain ed., vol. 2134 of Energy Minimization Methods in Computer Vision and Pattern Recognition, Lecture notes in Computer Science, Springer-Verlag, 2001, pp. 513–527.

[4] W. Cheney and W. Light, *A Course in Approximation Theory*, Brooks/Cole, 1999.

[5] N. I. M. Gould, D. Orban, and P. L. Toint, *Galahad, a library of thread-safe FORTRAN 90 packages for large-scale nonlinear optimization*, ACM Trans. Math. Softw., 29 (2003), pp. 353–372.

[6] S. Marsland and C. Twining, *Measuring geodesic distances on the space of bounded diffeomorphisms*, in British Machine Vision Conference (BMVC), 2002.

[7] J. Modersitzki, *Numerical Methods for Image Registration*, Oxford University Press, New York, 2004.

[8] F. J. Narcowich and J. D. Ward, *Norms of inverses and condition numbers for matrices associated with scattered data*, J. Approx. Theory, 64 (1991), pp. 69–94.

[9] Numerical Algorithms Group, *NAG Manual*, http://www.nag.co.uk/numeric/FN/manual/.

[10] R. Schaback, *Error estimates and condition numbers for radial basis function interpolation*, Advances in Computational Mathematics, 3 (1995), pp. 251–264.