CLAIRE: A DISTRIBUTED-MEMORY SOLVER FOR CONSTRAINED LARGE DEFORMATION DIFFEOMORPHIC IMAGE REGISTRATION*

ANDREAS MANG†, AMIR GHOLAMI‡, CHRISTOS DAVATZIKOS§, AND GEORGE BIROS¶

Abstract. We introduce CLAIRE, a distributed-memory algorithm and software for solving constrained large deformation diffeomorphic image registration problems in three dimensions. We invert for a stationary velocity field that parameterizes the deformation map. Our solver is based on a globalized, preconditioned, inexact reduced space Gauss–Newton–Krylov scheme.

We exploit state-of-the-art techniques in scientific computing to develop an effective solver that scales to thousand of distributed memory nodes on high-end clusters. Our improved, parallel implementation features parameter-, scale-, and grid-continuation schemes to speedup the computations and reduce the likelihood to get trapped in local minima. We also implement an improved preconditioner for the reduced space Hessian to speedup the convergence.

We test registration performance on synthetic and real data. We demonstrate registration accuracy on 16 neuroimaging datasets. We compare the performance of our scheme against different flavors of the `Demons` algorithm for diffeomorphic image registration. We study convergence of our preconditioner and our overall algorithm. We report scalability results on state-of-the-art supercomputing platforms. We demonstrate that we can solve registration problems for clinically relevant data sizes in two to four minutes on a standard compute node with 20 cores, attaining excellent data fidelity. With the present work we achieve a speedup of (on average) $5\times$ with a peak performance of up to $17\times$ compared to our former work.

Key words. diffeomorphic image registration; LDDMM; distributed-memory algorithm; Newton–Krylov method; KKT preconditioner; optimal control; PDE constrained optimization.

AMS subject classifications. 68U10, 49J20, 35Q93, 65K10, 65F08, 76D55.

1. Introduction. Deformable registration is a key technology in the medical imaging. It is about computing a map $\boldsymbol{y}$ that establishes a *meaningful* spatial correspondence between two (or more) images m_R (the *reference* (fixed) *image*) and m_T (the *template* (deformable or moving) *image*; image to be registered) of the same scene [45, 109]. Numerous approaches for formulating and solving image registration problems have appeared in the past; we refer to [45, 67, 109, 110, 131] for lucid overviews. Image registration is typically formulated as a variational optimization problem that consists of a data fidelity term and a Tikhonov regularization functional to over-come ill-posedness [43,45]. In many applications, a key concern is that $\boldsymbol{y}$ is a *diffeomorphism*, i.e., the map $\boldsymbol{y}$ is differentiable, a bijection, and has a differentiable inverse. A prominent strategy to ensure regularity of $\boldsymbol{y}$ is to introduce a pseudo-time variable t and invert for a smooth, time-dependent velocity field $\boldsymbol{v}$ that parameterizes the map $\boldsymbol{y}$ [17, 40, 107, 142]; existence of a diffeomorphism $\boldsymbol{y}$ can be guaranteed if $\boldsymbol{v}$ is adequately smooth [17, 31, 40, 137]. There exists a large body of literature of diffeomorphic registration parameterized by velocity fields $\boldsymbol{v}$ that, in many cases, focuses on theoretical considerations [40, 107, 145–147]. There is much less work on the design of efficient solvers; examples are [8, 9, 11, 13, 17, 36, 71, 116, 142, 148]. Most existing solvers use first order methods for numerical optimization and/or are based on heuristics that do not guarantee convergence. Due to computational costs, early termination results in compromised registration quality. Our intention in this work is to deploy an efficient solver for diffeomorphic image registration problems that (*i*) uses state-of-the art algorithms, (*ii*) is scalable to thousands of cores, (*iii*) requires minimal parameter tuning, (*iv*) and produces high-fidelity results with guaranteed regularity on a discrete level.

* This material is based upon work supported by NIH award 5R01NS042645-14; by NSF awards CCF-1817048 and CCF-1725743; by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Applied Mathematics program under award DE- SC0019393; by U.S. Air Force Office of Scientific Research award FA9550-17-1-0190; and by Simons Foundation award 586055. Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and do not necessarily reflect the views of the AFOSR, DOE, NIH, NSF, and Simons Foundation. Computing time on the Texas Advanced Computing Centers Stam- pede system was provided by an allocation from TACC and the NSF. This work was completed in part with resources provided by the Research Computing Data Core at the University of Houston.

†Department of Mathematics, University of Houston, Houston, Texas 77204-5008, USA, andreas@math.uh.edu

‡Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720-1770, USA, amir@accfft.org

§Center for Biomedical Image Computing and Analytics, Department of Radiology, University of Pennsylvania, Philadephia, PA 19104-2643, USA, christos.davatzikos@uphs.upenn.edu

¶Institute for Computational Engineering and Sciences, University of Texas at Austin, Austin, TX 78712-1229, USA, gbiros@acm.org

TABLE 1
Notation and symbols.

Symbol	Description	Acronym	Description
Ω	spatial domain; $\Omega := (0, 2\pi)^3 \subset \mathbf{R}^3$	CLAIRE	**c**onstrained **la**rge deformation diffeomorphic **i**mage **re**gistration [**?**]
$\boldsymbol{x}$	spatial coordinate; $\boldsymbol{x} := (x_1, x_2, x_3)^\mathsf{T} \in \mathbf{R}^3$	CFL	Courant–Friedrichs–Lewy (condition)
t	pseudo-time variable; $t \in [0, 1]$	CHEB(k)	Chebyshev (iteration) with fixed iteration number $k \in \mathbf{N}$ [56,61]
$m_R(\boldsymbol{x})$	reference image	FFT	fast Fourier transform
$m_T(\boldsymbol{x})$	template image (image to be registered)	GPL	GNU General Public License
$\boldsymbol{v}(\boldsymbol{x})$	stationary velocity field	HPC	high performance computing
$\boldsymbol{y}(\boldsymbol{x})$	deformation map	KKT	Karush–Kuhn–Tucker
$m(\boldsymbol{x}, t)$	state variable (transported intensities)	LDDMM	large deformation diffeomorphic metric mapping [17]
$m_1(\boldsymbol{x})$	final state; $m_1(\boldsymbol{x}) := m(\boldsymbol{x}, t = 1)$	matvec	matrix vector product
$\lambda(\boldsymbol{x}, t)$	adjoint variable	MPI	Message Passing Interface
$\tilde{m}(\boldsymbol{x}, t)$	incremental state variable	PETSc	Portable Extensible Toolkit for Scientific Computation [14,15]
$\tilde{\lambda}(\boldsymbol{x}, t)$	incremental state variable	PCG	preconditioned conjugate gradient (method) [73]
$\mathcal{L}$	Lagrangian functional	PCG(ϵ)	PCG method with relative tolerance $\epsilon \in (0, 1)$
$\boldsymbol{g}$	(reduced) gradient	RK2	2nd order Runge–Kutta method
$\mathcal{H}$	(reduced) Hessian operator	(S)DDEM	(symmetric) diffeomorphic demons [140,142]
∂_i	partial derivative with respect x_i	(S)LDDEM	(symmetric) log-domain diffeomorphic demons [141]
∇	gradient operator; $\nabla := (\partial_1, \partial_2, \partial_3)^\mathsf{T}$	TAO	Toolkit for Advanced Optimization [111]
$\nabla\cdot$	divergence operator		
Δ	Laplacian operator (vectorial and scalar)		
$\boldsymbol{n}_x$	number of grid points; $\boldsymbol{n}_x = (n_1, n_2, n_3)^\mathsf{T}$		
n_t	number of cells in temporal grid		
n	number of unknowns; $n = 3 \cdot \prod_{i=1}^{3} n_i$		

1.1. Outline of the Method. We use an optimal control formulation. The task is to find a smooth velocity field $\boldsymbol{v}$ (the ''control variable'') such that the distance between two images (or densities) is minimized, subject to a regularization norm for $\boldsymbol{v}$ and a deformation model given by a hyperbolic PDE constraint. More precisely, given two functions $m_R(\boldsymbol{x})$ (*reference image*) and $m_T(\boldsymbol{x})$ (*template image*) compactly supported on an open set $\Omega \subset \mathbf{R}^3$ with boundary $\partial\Omega$, we solve for a *stationary* velocity field $\boldsymbol{v}(\boldsymbol{x})$ as follows:

$$\underset{\boldsymbol{v}, m}{\text{minimize}} \quad \frac{1}{2} \int_\Omega (m_1(\boldsymbol{x}) - m_R(\boldsymbol{x}))^2 \, \mathrm{d}\boldsymbol{x} + \mathcal{S}(\boldsymbol{v}) \tag{1a}$$

$$\text{subject to} \quad \partial_t m + \boldsymbol{v} \cdot \nabla m = 0 \quad \text{in } \Omega \times (0, 1]$$
$$m = m_T \quad \text{in } \Omega \times \{0\} \tag{1b}$$

with periodic boundary conditions on $\partial\Omega$. Here, $m(\boldsymbol{x}, t)$ (the ''state variable'') corresponds to the transported intensities of $m_T(\boldsymbol{x})$ subject to the velocity field $\boldsymbol{v}(\boldsymbol{x})$; in our formulation, $m_1(\boldsymbol{x}) := m(\boldsymbol{x}, t = 1)$—i.e., the solution of (1b)—is equivalent to $m_T(\boldsymbol{y}(\boldsymbol{x}))$ for all $\boldsymbol{x}$ in Ω. The first part of the functional in (1a) measures the discrepancy between m_1 and m_R. The regularization functional $\mathcal{S}$ is a Sobolev norm that, if chosen appropriately, ensures that $\boldsymbol{v}$ gives rise to a diffeomorphism $\boldsymbol{y}$ [17,40,69,137]. We augment the formulation in (1) by constraints on the divergence of $\boldsymbol{v}$ to control volume change (see also [31,97]). Details on our formulation can be found in §2.1.

Problem (1) is ill-posed and involves ill-conditioned operators. We use the method of Lagrange multipliers to solve the constrained optimization problem (1). Our solver is based on an *optimize-then-discretize* approach; we first derive the optimality conditions and then discretize in space using a pseudospectral discretization with a Fourier basis. We use a globalized, inexact, preconditioned Gauss–Newton–Krylov method to solve for the first order optimality conditions [96]. The hyperbolic transport equations that appear in our formulation are integrated using a semi-Lagrangian method [44, 132]. Our solver uses distributed-memory parallelism and can be scaled up to thousands of cores [52,97].

1.2. Contributions. With this work we follow up on our former work on constrained diffeomorphic image registration [96–99]. We focus on registration performance, implementation aspects and the deployment of our solver, and introduce additional algorithmic improvements. Our contributions are:

- We implement an improved preconditioner for the reduced space Hessian (originally described in [98] for the two-dimensional case). We empirically evaluate several variants of this preconditioner.
- We introduce options for a grid, scale, and parameter continuation to our three-dimensional solver.
- We evaluate registration quality and compare our new, improved solver against different variants of

the diffeomorphic `Demons` algorithm [141,142].
- We study strong scaling performance of our improved solver.
- We make our software termed `CLAIRE` [**?**] (which stands for **c**onstrained **la**rge deformation diffeomorphic **i**mage **re**gistration) available under GPL license. The code can be downloaded here:

  https://github.com/andreasmang/claire.

  The URL for the deployment page is https://andreasmang.github.io/claire.

1.3. Limitations and Unresolved Issues. Several limitations and unresolved issues remain: • We assume similar intensity statistics for the reference image m_R and the template image m_T. This is a common assumption in many deformable image registration algorithms [17,69,90,112,143]. To enable the registration of images with a more complicated intensity relationship, more involved distance measures need to be considered [109,131]. • Our formulation is not symmetric, i.e., not invariant to a permutation of the reference and template image. The extension of our scheme to the symmetric case is mathematically straightforward [10,94,141] but its efficient implementation is nontrivial. This will be the subject of future work. • We invert for a stationary velocity field $\boldsymbol{v}(\boldsymbol{x})$ (i.e., the velocity does not change in time). Stationary paths on the manifold of diffeomorphisms are the group exponentials (i.e., one-parameter subgroups that do not depend on any metric); they do not cover the entire space of diffeomorphisms. The definition of a metric may be desirable in certain applications [17,105,148] and, in general, requires non-stationary velocities. Developing an effective, parallel solver for nonstationary $\boldsymbol{v}$ requires more work.

1.4. Related Work. With this work we follow up on our prior work on constrained diffeomorphic image registration [96,97,99,101,102]. We release `CLAIRE`, a software package for velocity-based diffeomorphic image registration. For excellent reviews on image registration see [67,109,131]. In diffeomorphic registration, we formally require that $\det \nabla \boldsymbol{y}$ does not vanish or change sign. An intuitive approach to safeguard against non-diffeomorphic $\boldsymbol{y}$ is to add hard and/or soft constraints on $\det \nabla \boldsymbol{y}$ to the variational problem [30,65,118,123]. An alternative strategy is to introduce a pseudo-time variable t and invert for a smooth velocity field $\boldsymbol{v}$ that parameterizes $\boldsymbol{y}$ [17,40,107,142]; existence of a diffeomorphism $\boldsymbol{y}$ can be guaranteed if $\boldsymbol{v}$ is adequately smooth [17,31,40,137]. Our approach falls into this category. We use a PDE-constrained optimal control formulation; we refer to [21,27,59,75,93] for insight into theory and algorithmic developments in optimal control. In general, the solver has to be tailored to the structure of the control problem, which is dominated by the PDE constraints; examples for elliptic, parabolic, and hyperbolic PDEs can be found in [1,22], [2,51,103,133], and [19,26,72,90,144], respectively. In our formulation, the PDE constraint is—in its simplest form—a hyperbolic transport equation. Our formulation has been introduced in [69,96,97]. A prototype implementation of our solver has been described in [96] and improved in [98]. We have extend our original solver [96] to the 3D setting in [52,99]. The focus in [52,99] is the scalability of our solver on HPC platforms. In [122] we presented an integrated formulation for registration and biophysical tumor growth simulations that has been successfully applied to segmentation of neuroimaging data [53,102].

Optimal control formulations that are related to ours have been described in [26,31,69,72,90,91,143]. Related formulations for optimal mass transport are described in [19,64,101,138]. Our work differs from optimal mass transport in that intensities are constant along the characteristics (i.e., mass is not preserved). Our formulation shares numerous characteristics with traditional optical flow formulations [77,81,120]. The key difference is that we treat the transport equation for the image intensities as a hard constraint. PDE-constrained formulations for optical flow, which are equivalent to our formulation, are described in [6,16,26,31]. Our work is closely related to the LDDMM approach [10,11,17,40,137,145], which builds upon the pioneering work in [33]. LDDMM uses a non-stationary velocity but there exist variants that use stationary $\boldsymbol{v}$ [7,8,71,94,95,142]; they are more efficient. If we are only interested in registering two images, stationary $\boldsymbol{v}$ produce good results. Another strategy to reduce the size of the search space is geodesic shooting [9,106,143,145,149]; the control variable is an initial momentum/velocity at $t = 0$.

Among the most popular, publicly available packages for diffeomorphic registration are `Demons` [141, 142], `ANTs` [11], `PyCA` [**?**], `deformetrica` [**?**,**?**,**?**,**?**] and `DARTEL` [8]. Other popular packages for deformable registration are `IRTK` [119], `elastix` [86], `NiftyReg` [108], and `FAIR` [110]. The latter are, with the exception of `FAIR`, based on (low dimensional) parametric deformation models. Unlike existing approaches, `CLAIRE` features explicit control on the determinant of the deformation gradient; we introduce hard constraints on the divergence of $\boldsymbol{v}$. Our formulation was originally proposed in [97]; a similar approach is described

in [26]. Other works that consider divergence-free $\boldsymbol{v}$ have been described in [31,74,104,120,121].

There exist few works on effective numerical methods. Despite the fact that first order methods for optimization have poor convergence rates for non-linear, ill-posed problems, most works, with the exception of ours [96–101] and [9,19,70,72,129,142], use first order gradient descent type approaches. We use a globalized Newton–Krylov method, instead. For these methods to be efficient, it is critical to design an effective preconditioner. (We refer to [18] for an overview on preconditioning of saddle point problems.) Preconditioners for problems similar to ours can be found in [19,72,129]. Another critical component is the PDE-solver. In our case, the expensive PDE operators are hyperbolic transport equations. Several strategies to efficiently solve these equations have been considered in the past [17,19,26,31,69,96,97,99,101,102,116,129]. We use a semi-Lagrangian scheme [17,31,99,102].

Another key feature of `CLAIRE` is that it can be executed in parallel [52,99]. Examples for parallel solvers for PDE-constrained optimization problems can be found in [3,4,20,21,23–25,128]. We refer to [42,46,125,127] for surveys on parallel algorithms for image registration. Implementations, such as `Demons` [141,142], `ANTs` [11], or `elastix` [86], which are largely based on kernels implemented in the `ITK` package [79], exploit multi-threading for parallelism. GPU implementations of different variants of map-based, low-dimensional parametric approaches are described in [108,124,126]. A GPU implementation of a map-based non-parametric approach is described in [87]. GPU implementations with formulations that are similar to ours are described in [**?**,**?**,62,63,130,138,139]. The work that is most closely related to ours, is [62,63,139]. In [62,63] a (multi-)GPU implementation of the approach described in [80] is presented. The work in [139] discusses a GPU implementation of `DARTEL` [8].

What sets our work apart are the numerics and our distributed-memory implementation: We use high-order numerical methods (second order time integration, cubic interpolation, and spectral differentiation). The linear solvers and the Gauss–Newton optimizer are built on top of `PETSc` [15] and `TAO` [111]. Our solver uses MPI for parallelism and has been deployed to HPC systems [52,97]. This allows us to target applications of unprecedented scale (such as CLARITY imaging [136]) without posing the need to downsample the data [88]. We will see that we can solve problems with 3 221 225 472 unknowns in 2 min on 22 compute nodes (256 MPI tasks) and in less than 5 s if we use 342 compute nodes (4096 MPI tasks). Exploiting parallelism also allows us to deliver run times that approach real-time capabilities.

1.5. Outline. We present our approach for large deformation diffeomorphic image registration in §2, which comprises the formulation of the problem (see §2.1), a formal presentation of the optimality conditions (see §2.2), and a discussion of the numerical implementation (see §2.3). Numerical experiments are reported in §4. We conclude with §6. This publication is accompanied by supplementary material. There, we report additional, more detailed results.

2. Methods. In what follows, we describe the main building blocks of our formulation, our solver, and its implementation, and introduce new features that distinguish this work from our former work [52, 96–101].

2.1. Formulation. Given two images—the *reference image* $m_R(\boldsymbol{x})$ and the *template image* $m_T(\boldsymbol{x})$—compactly supported on $\Omega = (0, 2\pi)^3 \subset \mathbf{R}^3$, with boundary $\partial\Omega$ and closure $\bar{\Omega}$, our aim is to compute a *plausible* deformation map $\boldsymbol{y}(\boldsymbol{x})$ such that $\forall \boldsymbol{x} \in \Omega : m_R(\boldsymbol{x}) \approx m_T(\boldsymbol{y}(\boldsymbol{x}))$ [45,109,110]. We consider a map $\boldsymbol{y}$ to be plausible if it is a diffeomorphism, i.e., an invertible map, which is continuously differentiable (a C^1-function) and maps Ω onto itself. In our formulation, we do not directly invert for $\boldsymbol{y}$; we introduce a pseudo-time variable $t \in [0,1]$ and invert for a *stationary* velocity $\boldsymbol{v}(\boldsymbol{x})$, instead. In particular, we solve for $\boldsymbol{v}(\boldsymbol{x})$ and a mass source map $w(\boldsymbol{x})$ as follows [97]:

$$\text{(2a)} \qquad \underset{m,\boldsymbol{v},w}{\text{minimize}} \quad \frac{1}{2}\int_\Omega (m_1 - m_R)^2 \,\mathrm{d}\boldsymbol{x} + \frac{\beta_v}{2}\langle \mathcal{B}[\boldsymbol{v}], \mathcal{B}[\boldsymbol{v}]\rangle_{L^2(\Omega)^s} + \frac{\beta_w}{2}\int_\Omega \nabla w \cdot \nabla w + w^2 \,\mathrm{d}\boldsymbol{x}$$

$$\text{subject to} \quad \partial_t m + \nabla m \cdot \boldsymbol{v} = 0 \quad \text{in } \Omega \times (0,1]$$

$$\text{(2b)} \qquad\qquad m = m_T \quad \text{in } \Omega \times \{0\}$$

$$\nabla \cdot \boldsymbol{v} = w \quad \text{in } \Omega$$

with periodic boundary conditions on $\partial\Omega$, and $s > 0$, $\beta_v > 0$, $\beta_w > 0$. The state variable $m(\boldsymbol{x}, t)$ in (2b) represents the transported intensities of m_T subjected to the velocity field $\boldsymbol{v}$; the solution of the first

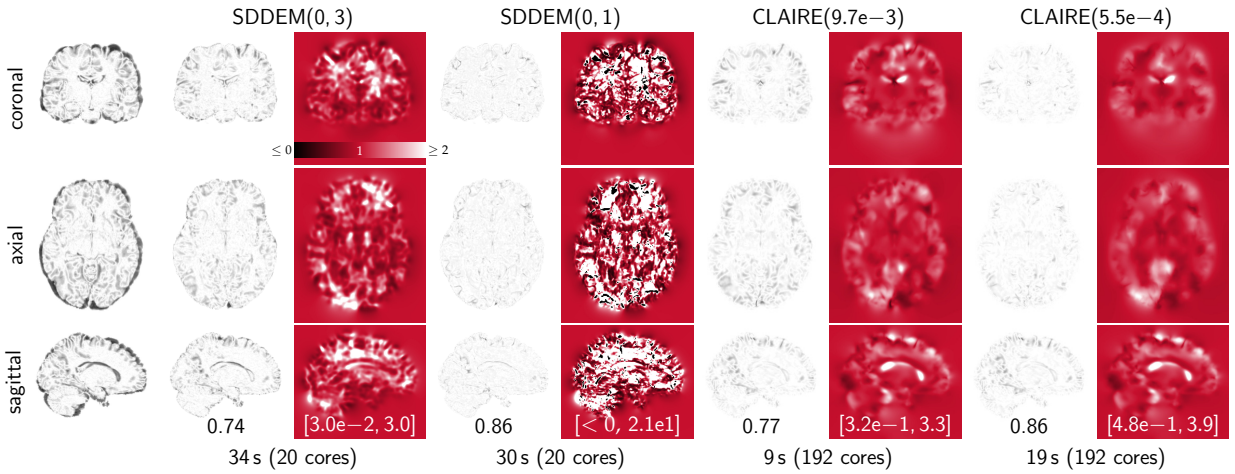

FIG. 1. *We compare results for* `CLAIRE` *and the diffeomorphic* `Demons` *algorithm. We consider the first two volumes of the* `NIREP` *dataset. We report results for the symmetric diffeomorphic* `Demons` *algorithm (SDDEM) with regularization parameters (σ_d, σ_u) determined by an exhaustive search. We report results for* `CLAIRE` *for different choices for the regularization parameter for the velocity ($\beta_v = 3.70\text{e}{-3}$ and $\beta_v = 5.50\text{e}{-4}$; determined by a binary search). We show the original mismatch on the left. For each variant of the considered algorithms we show the mismatch after registration and a map for the determinant of the deformation gradient. We report values of the Dice score of the union of all available gray matter labels below the mismatch. We also report the extremal values for the determinant of the deformation gradient. We execute the* `Demons` *algorithm on one node of the CACDS's Opuntia server (Intel ten-core Xeon E5-2680v2 at 2.80 GHz with 64 GB memory; 2 sockets for a total of 20 cores; [35]) using 20 threads. We use a grid continuation scheme with 15, 10, and 5 iterations per level, respectively. If we execute* `CLAIRE` *on the same system, the runtime is 103 s and 202 s, respectively. If we increase the number of iterations of SDDEM to 150, 100, 50 per level, we obtain a dice score of 0.75 and 0.86 with a runtime of 322 s and 297 s, respectively. The results for* `CLAIRE` *are for 16 nodes with 12 MPI tasks per node on TACC's Lonestar 5 system (2-socket Xeon E5-2690 v3 (Haswell) with 12 cores/socket, 64 GB memory per node; [135]). We execute* `CLAIRE` *at full resolution using a parameter continuation scheme in β_v. Detailed results for these runs can be found in the supplementary material, in particular Tab. SM7, Tab. SM8, and Tab. SM12.*

equation in (2b), i.e., $m_1(\boldsymbol{x}) := m(\boldsymbol{x}, t = 1)$, is equivalent to $m_T(\boldsymbol{y}(\boldsymbol{x}))$, where $\boldsymbol{y}$ is the Eulerian (or pullback) map. We use a squared L^2-distance to measure the proximity between m_1 and m_R. The parameters $\beta_v > 0$ and $\beta_w > 0$ control the contribution of the regularization norms for $\boldsymbol{v}$ and w. The constraint on the divergence of $\boldsymbol{v}$ in (2b) allows us to control the compressibility of $\boldsymbol{y}$. If we set w in (2b) to zero $\boldsymbol{y}$ is incompressible, i.e., $\forall \boldsymbol{x} \in \Omega : \det \nabla \boldsymbol{y}(\boldsymbol{x}) = 1$ up to numerical accuracy [60]. By introducing a nonzero mass-source map w, we can relax this model to near-incompressible diffeomorphisms $\boldsymbol{y}$; the regularization on w in (2a) acts like a penalty on the divergence of $\boldsymbol{v}$; we use an H^1-norm.

Our solver supports different Sobolev (semi-)norms to regularize $\boldsymbol{v}$. The choice of the differential operator $\mathcal{B}$ not only depends on application requirements but is also critical from a theoretical point of view; an adequate choice guarantees existence and uniqueness of an optimal solution of the control problem [16, 17, 26, 31, 90] (subject to the smoothness properties of the images). We use an H^1-seminorm, i.e., $\mathcal{B} = \nabla$, if we consider the incompressibility constraint. If we neglect the incompressibility constraint, we use $\mathcal{B} = -\Delta$.

2.2. Optimality Condition and Newton Step. We use the method of Lagrange multipliers [93] to turn the constrained problem (2) into an unconstrained one; neglecting boundary conditions, the Lagrangian functional is given by

(3)
$$\mathcal{L}[m, \lambda, p, w, \boldsymbol{v}] := \frac{1}{2}\int_\Omega (m_1 - m_R)^2 \,\mathrm{d}\boldsymbol{x} + \frac{\beta_v}{2}\langle \mathcal{B}[\boldsymbol{v}], \mathcal{B}[\boldsymbol{v}]\rangle_{L^2(\Omega)^s} + \frac{\beta_w}{2}\int_\Omega \nabla w \cdot \nabla w + w^2 \,\mathrm{d}\boldsymbol{x}$$
$$+ \int_0^1 \langle \partial_t m + \nabla m \cdot \boldsymbol{v}, \lambda\rangle_{L^2(\Omega)} \,\mathrm{d}t + \langle m(t=0) - m_T, \lambda(t=0)\rangle_{L^2(\Omega)} - \langle \nabla \cdot \boldsymbol{v} - w, p\rangle_{L^2(\Omega)},$$

with Lagrange multipliers $\lambda : \bar{\Omega} \times [0, 1] \to \mathbf{R}$ for the transport equation (2b), and $p : \bar{\Omega} \to \mathbf{R}$ for the incompressibility constraint (2b). Formally, we have to compute variations of $\mathcal{L}$ with respect to the state, adjoint, and control variables. We will only consider a reduced form (after eliminating the incompressibil-

ity constraint) of the optimality system—a system of nonlinear PDEs for m, λ, and $\boldsymbol{v}$. Details on how we formally arrive at this reduced from can be found in [96,97].

We use a reduced-space approach, in which we eliminate the state and adjoint variables, and iterate in the inversion parameter space. The evaluation of the reduced gradient $\boldsymbol{g}$ (the first variation of the Lagrangian $\mathcal{L}$ in (3) with respect to $\boldsymbol{v}$) for a candidate $\boldsymbol{v}$ requires several steps. We first solve the transport equation (2b) with initial condition $m(\boldsymbol{x}, t=0) = m_T(\boldsymbol{x})$ forward in time to obtain the state variable $m(\boldsymbol{x}, t)$. Given m we then compute the adjoint variable $\lambda(\boldsymbol{x}, t)$ by solving the adjoint equation

$$-\partial_t \lambda - \nabla \cdot \lambda \boldsymbol{v} = 0 \qquad \text{in } \Omega \times [0,1), \tag{4a}$$

$$\lambda = m_R - m \qquad \text{in } \Omega \times \{1\}, \tag{4b}$$

with periodic boundary conditions on $\partial\Omega$ backward in time. Once we have the adjoint and state fields, we can evaluate the expression for the reduced gradient

$$\boldsymbol{g}(\boldsymbol{v}) := \beta_v \mathcal{A}[\boldsymbol{v}] + \mathcal{K}\big[\int_0^1 \lambda \nabla m \, \mathrm{d}t\big]. \tag{5}$$

The differential operator $\mathcal{A} = \mathcal{B}^*\mathcal{B}$ in (5) corresponds to the first variation of the regularization norm for $\boldsymbol{v}$ in (3), resulting in an elliptic ($\mathcal{A} = -\Delta$), biharmonic ($\mathcal{A} = \Delta^2$) , or triharmonic ($\mathcal{A} = \Delta^3$) control equation for $\boldsymbol{v}$, respectively. The operator $\mathcal{K}$ projects $\boldsymbol{v}$ onto the space of incompressible or near-incompressible velocity fields; we have $\mathcal{K} := \boldsymbol{I} - \nabla(\beta_v(\beta_w(-\Delta + \mathrm{id}))^{-1} + \mathrm{id})^{-1}\Delta^{-1}\nabla\cdot$ and $\mathcal{K} := \mathrm{id} - \nabla\Delta^{-1}\nabla\cdot$ for the incompressible case. If we neglect the incompressibility constraint (2b), $\mathcal{K}$ in (5) is an identity operator. The dependence of m and λ on $\boldsymbol{v}$ is "hidden" in the transport and continuity equations (2b) and (4a), respectively.

The first order optimality condition (control or decision equation) requires that $\boldsymbol{g}(\boldsymbol{v}^\star) = \boldsymbol{0}$ for an admissible solution $\boldsymbol{v}^\star$ to (2). Most available registration packages use gradient descent type optimization schemes to find an optimal point [17, 69, 143]. Newton-type methods are expected to yield better convergence rates [28, 113]. However, if they are implemented naively, they can become computationally prohibitive. The expressions associated with the Newton step of our control problem are formally obtained by computing second variations of the Lagrangian in (3). In *full space methods* we find the Newton updates (i.e., the search direction) for the state, adjoint, and control variables of our control problem simultaneously. That is, we iterate on all variables at once. In *reduced space methods* we only iterate on the control variable $\boldsymbol{v}$. Reduced space methods can be obtained from the full space KKT system by block elimination [23–25,117]. The associated reduced space Newton system for the incremental control variable $\tilde{\boldsymbol{v}}$ (search direction) is given by

$$\mathcal{H}\tilde{\boldsymbol{v}} = -\boldsymbol{g}, \tag{6}$$

where $\boldsymbol{g}$ is the reduced gradient in (5). The expression for the (reduced space) Hessian matvec (action of $\mathcal{H}$ on a vector $\tilde{\boldsymbol{v}}$) in (6) is given by

$$\mathcal{H}[\tilde{\boldsymbol{v}}](\boldsymbol{v}) := \underbrace{\beta_v \mathcal{A}[\tilde{\boldsymbol{v}}]}_{=:\mathcal{H}_{\text{reg}}[\tilde{v}]} + \underbrace{\mathcal{K}\big[\int_0^1 \tilde{\lambda}\nabla m + \lambda \nabla \tilde{m} \, \mathrm{d}t\big]}_{=:\mathcal{H}_{\text{data}}[\tilde{v}]} = \mathcal{H}_{\text{reg}}[\tilde{v}] + \mathcal{H}_{\text{data}}[\tilde{v}]. \tag{7}$$

We use the notation $\mathcal{H}[\tilde{\boldsymbol{v}}](\boldsymbol{v})$ to indicate that the Hessian matvec in (7) is a function of $\boldsymbol{v}$ through a set of PDEs for $m(\boldsymbol{x}, t)$, $\tilde{m}(\boldsymbol{x}, t)$, $\lambda(\boldsymbol{x}, t)$, and $\tilde{\lambda}(\boldsymbol{x}, t)$. The space-time fields m and λ are found during the evaluation of (2a) and (5) for a candidate $\boldsymbol{v}$ as described above. What is missing to be able to evaluate (7) are the fields $\tilde{m}(\boldsymbol{x}, t)$ and $\tilde{\lambda}(\boldsymbol{x}, t)$. Given $m(\boldsymbol{x}, t)$, $\boldsymbol{v}(\boldsymbol{x})$, and $\tilde{\boldsymbol{v}}(\boldsymbol{x})$, we find $\tilde{m}(\boldsymbol{x}, t)$ by solving

$$\partial_t \tilde{m} + \nabla \tilde{m} \cdot \boldsymbol{v} + \nabla m \cdot \tilde{\boldsymbol{v}} = 0 \qquad \text{in } \Omega \times (0,1], \tag{8a}$$

$$\tilde{m} = 0 \qquad \text{in } \Omega \times \{0\}, \tag{8b}$$

forward in time. Now, given $\tilde{m}(\boldsymbol{x}, t=1)$, $\lambda(\boldsymbol{x}, t)$, $\boldsymbol{v}(\boldsymbol{x})$, and $\tilde{\boldsymbol{v}}(\boldsymbol{x})$ we solve

$$-\partial_t \tilde{\lambda} - \nabla \cdot (\tilde{\lambda}\boldsymbol{v} + \lambda \tilde{\boldsymbol{v}}) = 0 \qquad \text{in } \Omega \times [0,1), \tag{9a}$$

$$\tilde{\lambda} = -\tilde{m} \qquad \text{in } \Omega \times \{1\}, \tag{9b}$$

for $\tilde{\lambda}(\boldsymbol{x}, t)$ backward in time.

2.3. Numerics. In the following, we describe our distributed-memory solver for 3D diffeomorphic image registration problems and provide additional information for our software package termed CLAIRE.

2.3.1. Discretization. We discretize in space on a regular grid $\Omega^h \in \mathbf{R}^{3,n_1,n_2,n_3}$ with grid points $\mathbf{x}_k := 2\pi \boldsymbol{k} \oslash \boldsymbol{n}_x$, $\boldsymbol{k} = (k_1,k_2,k_3)^\mathsf{T} \in \mathbf{N}^3$, $-n_i/2+1 \leq k_i \leq n_i/2$, $i = 1,2,3$, $\boldsymbol{n}_x := (n_1,n_2,n_3)^\mathsf{T} \in \mathbf{N}^3$ and periodic boundary conditions; $\oslash$ denotes the Hadamard division. In the continuum, we model images as compactly supported (periodic), smooth functions. We apply Gaussian smoothing (in the spectral domain) with a bandwidth of $\boldsymbol{h}_x = (h_1,h_2,h_3)^\mathsf{T} \in \mathbf{R}^3$ and mollify the discrete data to meet these requirements. We rescale the images to an intensity range of $[0,1] \subset \mathbf{R}$ prior to registration. We use a trapezoidal rule for numerical quadrature and a spectral projection scheme for all spatial operations. The mapping between spectral and spatial domain is done using forward and inverse FFTs [50]. All spatial derivatives are computed in the spectral domain; we first take the FFT, then apply the appropriate weights to the spectral coefficients, and then take the inverse FFT. This scheme allows us to efficiently and accurately apply differential operators and their inverses. Consequently, the main cost of our scheme is the solution of the transport equations (2b), (4a), (8a), and (9a), and not the inversion of differential (e.g., elliptic or biharmonic) operators. We use a nodal discretization in time, which results in n_t+1 space-time fields we need to solve for. We use fully explicit, unconditionally stable semi-Lagrangian scheme [44,132] to solve the transport equations that appear in our formulation ((2b), (4a), (8a), and (9a)). This allows us to keep n_t small (we found empirically that $n_t = 4$ yields a good compromise between runtime and numerical accuracy). The time integration steps in our semi-Lagrangian scheme is implemented using a fully explicit 2nd order Runge–Kutta scheme. Interpolations are carried out using third-degree polynomials. Details for our semi-Lagrangian scheme can be found in [52,98,99].

2.3.2. Newton–Krylov Solver. A prototype implementation of our Newton–Krylov solver is described in [96,98]. We have already mentioned in §2.2 that we use a reduced space method. That is, we only iterate on the reduced space for the control variable $\mathbf{v} \in \mathbf{R}^n$, $n = 3n_1n_2n_3$. We globalize our method using an Armijo linesearch, resulting in the iterative scheme

$$\mathbf{v}_{k+1} = \mathbf{v}_k + \alpha_k \tilde{\mathbf{v}}_k, \qquad \mathbf{H}_k \tilde{\mathbf{v}}_k = -\mathbf{g}_k, \tag{10}$$

with iteration index k, step length $\alpha_k \geq 0$, iterate $\mathbf{v}_k \in \mathbf{R}^n$, search direction $\tilde{\mathbf{v}}_k \in \mathbf{R}^n$, reduced gradient $\mathbf{g}_k \in \mathbf{R}^n$ (see (5) for the continuous equivalent), and reduced space Hessian $\mathbf{H}_k \in \mathbf{R}^{n,n}$, where

$$\mathbf{H}_k = \mathbf{H}_{\text{reg}} + \mathbf{H}_{\text{data},k}$$

(see (7) for an expression for the Hessian matvec in the continuous setting). We refer to the steps for updating $\mathbf{v}_k$ as outer iterations and the steps for computing the search direction $\tilde{\mathbf{v}}_k$ as inner iterations.

Algorithm 2.1 Inexact Newton–Krylov method (outer iterations). We use the relative norm of the reduced gradient (with tolerance $\epsilon_{\text{opt}} > 0$ as stopping criterion.

1: initial guess $\mathbf{v}_0 \leftarrow \mathbf{0}$, $k \leftarrow 0$
2: $\mathbf{m}_0 \leftarrow$ solve state equation in (2b) forward in time given $\mathbf{v}_0$
3: $j_0 \leftarrow$ evaluate objective functional (2a) given $\mathbf{m}_0$ and $\mathbf{v}_0$
4: $\boldsymbol{\lambda}_0 \leftarrow$ solve adjoint equation (4a) backward in time given $\mathbf{v}_0$ and $\mathbf{m}_0$
5: $\mathbf{g}_0 \leftarrow$ evaluate reduced gradient (5) given $\mathbf{m}_0$, $\boldsymbol{\lambda}_0$ and $\mathbf{v}_0$
6: **while** $\|\mathbf{g}_k\|_2^2 > \|\mathbf{g}_0\|_2^2 \epsilon_{\text{opt}}$ **do**
7: $\quad \tilde{\mathbf{v}}_k \leftarrow$ solve $\mathbf{H}_k \tilde{\mathbf{v}}_k = -\mathbf{g}_k$ given $\mathbf{m}_k$, $\boldsymbol{\lambda}_k$, $\mathbf{v}_k$, and $\mathbf{g}_k$ (see Alg. 2.2)
8: $\quad \alpha_k \leftarrow$ perform line search on $\tilde{\mathbf{v}}_k$ subject to Armijo condition
9: $\quad \mathbf{v}_{k+1} \leftarrow \mathbf{v}_k + \alpha_k \tilde{\mathbf{v}}_k$
10: $\quad \mathbf{m}_{k+1} \leftarrow$ solve state equation (2b) forward in time given $\mathbf{v}_{k+1}$
11: $\quad j_{k+1} \leftarrow$ evaluate (2a) given $\mathbf{m}_{k+1}$ and $\mathbf{v}_{k+1}$
12: $\quad \boldsymbol{\lambda}_{k+1} \leftarrow$ solve adjoint equation (4a) backward in time given $\mathbf{v}_{k+1}$ and $\mathbf{m}_{k+1}$
13: $\quad \mathbf{g}_{k+1} \leftarrow$ evaluate (5) given $\mathbf{m}_{k+1}$, $\boldsymbol{\lambda}_{k+1}$ and $\mathbf{v}_{k+1}$
14: $\quad k \leftarrow k+1$
15: **end while**

Algorithm 2.2 Newton step (inner iterations). We illustrate the solution of the reduced KKT system (6) using a PCG method at a given outer iteration $k \in \mathbf{N}$. We use a superlinear forcing sequence to compute the tolerance η_k for the PCG method (inexact solve).

1: input: $\mathbf{m}_k$, $\boldsymbol{\lambda}_k$, $\mathbf{v}_k$, $\mathbf{g}_k$, $\mathbf{g}_0$
2: set $\epsilon_H \leftarrow \min(0.5, (\|\mathbf{g}_k\|_2/\|\mathbf{g}_0\|_2)^{1/2})$, $\tilde{\mathbf{v}}_0 \leftarrow \mathbf{0}$, $\mathbf{r}_0 \leftarrow -\mathbf{g}_k$
3: $\mathbf{z}_0 \leftarrow$ apply preconditioner $\mathbf{M}^{-1}$ to $\mathbf{r}_0$
4: $\mathbf{s}_0 \leftarrow \mathbf{z}_0$, $\iota \leftarrow 0$
5: **while** $\iota < n$ **do**
6: $\quad \tilde{\mathbf{m}}_\iota \leftarrow$ solve incremental state equation (8) forward in time given $\mathbf{m}_k$, $\mathbf{v}_k$ and $\tilde{\mathbf{v}}_\iota$
7: $\quad \tilde{\boldsymbol{\lambda}}_\iota \leftarrow$ solve incremental adjoint equation (9) backward in time given $\boldsymbol{\lambda}_k$, $\mathbf{v}_k$, $\tilde{\mathbf{m}}_\iota$ and $\tilde{\mathbf{v}}_\iota$
8: $\quad \tilde{\mathbf{s}}_\iota \leftarrow$ apply $\mathbf{H}_\iota$ to $\boldsymbol{s}_\iota$ given $\boldsymbol{\lambda}_k$, $\mathbf{m}_k$, $\tilde{\mathbf{m}}_\iota$ and $\tilde{\boldsymbol{\lambda}}_\iota$ (Hessian matvec; see (7))
9: $\quad \kappa_\iota \leftarrow \langle \mathbf{r}_\iota, \mathbf{z}_\iota \rangle / \langle \mathbf{s}_\iota, \tilde{\mathbf{s}}_\iota \rangle$, $\quad \tilde{\mathbf{v}}_{\iota+1} \leftarrow \tilde{\mathbf{v}}_\iota + \kappa_\iota \mathbf{s}_\iota$, $\quad \mathbf{r}_{\iota+1} \leftarrow \mathbf{r}_\iota - \kappa_\iota \tilde{\mathbf{s}}_\iota$
10: $\quad$ **if** $\|\mathbf{r}_{\iota+1}\|_2 < \epsilon_H$ **break**
11: $\quad \mathbf{z}_{\iota+1} \leftarrow$ apply preconditioner $\mathbf{M}^{-1}$ to $\mathbf{r}_{\iota+1}$
12: $\quad \mu_\iota \leftarrow \langle \mathbf{z}_{\iota+1}, \mathbf{r}_{\iota+1} \rangle / \langle \mathbf{z}_\iota, \mathbf{r}_\iota \rangle$, $\quad \mathbf{s}_{\iota+1} \leftarrow \mathbf{z}_{\iota+1} + \mu_\iota \mathbf{s}_\iota$, $\quad \iota \leftarrow \iota + 1$
13: **end while**
14: output: $\tilde{\mathbf{v}}_k \leftarrow \tilde{\mathbf{v}}_{\iota+1}$

In what follows, we drop the dependence on the (outer) iteration index k for notational convenience. The data term $\mathbf{H}_{\text{data}}$ of the reduced space Hessian $\mathbf{H}$ in (10) involves inverses of the state and adjoint operators (a consequence of the block elimination in reduced space methods; see §2.2). This makes $\mathbf{H}$ a non-local, dense operator that is too large to be computed and stored. (We have seen in §2.2 that each matvec given by (7) requires the solution of (8) forward in time and (9) backward in time; see also lines 6 and 7 in Alg. 2.2). So, to form $\mathbf{H}$ we require a total of $2n$ PDE solves per outer iteration k.) Consequently, direct methods to solve the linear system in (10) are not applicable. We use iterative, matrix-free Krylov subspace methods instead. They only require an expression for the action of $\mathbf{H}$ on a vector, which is precisely what we are given in (7). We use a PCG method [73] under the assumption that $\mathbf{H}$ is a symmetric, positive (semi)definite operator. To reduce computational costs, we do not solve the linear system in (10) exactly. Instead, we use a tolerance $\epsilon_H > 0$ that is proportional to the norm of $\mathbf{g}$ (see lines 2 and 10 in Alg. 2.2; details can be found in [39, 41] and [113, p. 166ff]). We summarize the steps for the outer and inner iterations of our Newton–Krylov solver in Alg. 2.1 and Alg. 2.2, respectively.

Since we are solving a non-convex problem it is not guaranteed that the Hessian $\mathbf{H}$ is positive definite. As a remedy, we use a Gauss–Newton approximation $\tilde{\mathbf{H}}$ to $\mathbf{H}$; doing so guarantees that $\tilde{\mathbf{H}} \succeq 0$ far away from the (local) optimum. This corresponds to dropping all terms in (7) and (9) that involve the adjoint variable λ. We expect the rate of convergence of our solver to drop from quadratic to superlinear. As λ tends to zero (i.e., the mismatch goes to zero), we recover quadratic convergence. We terminate the inversion if the ℓ^2-norm of the gradient in (5) is reduced by a factor of $\epsilon_g > 0$, i.e., if $\|\mathbf{g}_k\|_2^2 < \epsilon_{\text{opt}} \|\mathbf{g}_0\|_2^2$, where $\mathbf{g}_k \in \mathbf{R}^n$ is the gradient at (outer) iteration $k \in \mathbf{N}_0$ and $\mathbf{g}_0$ is the gradient for the initial guess $\mathbf{v}_0 = \mathbf{0}$ (see line 6 in Alg. 2.1). In most of our experiments, we use $\epsilon_g = 5\text{e}{-2}$. We also provide an option to set a lower bound for the absolute ℓ^2-norm of the gradient (the default value is $1\text{e}{-6}$). `CLAIRE` also features other stopping criteria discussed in [54, 96, 110] (not considered in this work).

2.3.3. Preconditioners for Reduced Space Hessian. We have seen that we need to solve two PDEs every time $\mathbf{H}$ is applied to a vector. These PDE solves are the most expensive part of our solver. Consequently, we have to keep the number of Hessian matvecs small for our solver to be efficient. This necessitates the design of an effective preconditioner $\mathbf{M}$. The speed of convergence of the linear solver used to compute the search direction $\tilde{\mathbf{v}}$ in (10) depends on the distance of $\mathbf{M}^{-1}\mathbf{H}$ from identity; ideally, the spectrum of $\mathbf{M}^{-1}\mathbf{H}$ is clustered around one. We cannot form and store $\mathbf{H}$ (too expensive). Moreover, we know that large eigenvalues of $\mathbf{H}$ are associated with smooth eigenvectors [96]. Consequently, standard preconditioners for linear systems are not applicable. In our former work, we have considered two matrix-free preconditioners. Our first preconditioner is based on the (exact) inverse of the regularization operator $\mathbf{H}_{\text{reg}}$; the regularization preconditioned Hessian is given by $\boldsymbol{I} + \mathbf{H}_{\text{reg}}^{-1}\mathbf{H}_{\text{data}}$. This is a common choice in PDE-constrained optimization problems [5, 29]. We used this preconditioner in [96, 97, 99–101].

REMARK 1. *$\mathbf{H}_{reg}$ is a discrete representation of the regularization operator. The computational costs for inverting*

and applying this operator are negligible (two FFTs and a diagonal scaling). Notice, that the operator $\mathbf{H}_{\text{reg}}$ *is singular if we consider a seminorm as regularization model in* (2)*. We bypass this problem by setting the zero singular values of the regularization operator to one before computing the inverse.*

The second preconditioner uses an inexact inverse of a coarse grid approximation to the Hessian $\mathbf{H}$. This preconditioner was proposed and tested in [98] for the 2D case. A similar preconditioner has been developed in [2, 22]. It is based on the conceptual idea that we can decompose the reduced Hessian $\mathbf{H}$ into two operators $\mathbf{H}_L$ and $\mathbf{H}_H$ that act on the high and low frequency parts of a given vector $\tilde{\mathbf{v}}$, respectively [2,22,55,82–84]. We denote the operators that project on the low and high frequency subspaces by $\mathbf{F}_L : \mathbf{R}^n \to \mathbf{R}^n$ and $\mathbf{F}_H : \mathbf{R}^n \to \mathbf{R}^n$, respectively. With $\mathbf{F}_H + \mathbf{F}_L = \mathbf{I}$, we have

$$\mathbf{H}\mathbf{e}_k = (\mathbf{F}_H + \mathbf{F}_L)\mathbf{H}(\mathbf{F}_H + \mathbf{F}_L)\mathbf{e}_k = \mathbf{F}_H\mathbf{H}\mathbf{F}_H\mathbf{e}_k + \mathbf{F}_L\mathbf{H}\mathbf{F}_L\mathbf{e}_k,$$

under the assumption that the unit vector $\mathbf{e}_k \in \mathbf{R}^n$, $(\mathbf{e}_k)_i = 1$ if $k = i$ and $(\mathbf{e}_k)_i = 0$ for $i \neq k$, $i,k = 1,\dots,n$, is an eigenvector of $\mathbf{H}$ so that $(\mathbf{F}_L\mathbf{H}\mathbf{F}_H)\mathbf{e}_k = (\mathbf{F}_H\mathbf{H}\mathbf{F}_L)\mathbf{e}_k = \mathbf{0}$. We note that this assumption will not hold in general. However, since we are only interested in developing a preconditioner, an approximate decomposition of the solution of the reduced space system is acceptable. Using this approximation we can represent the solution of $\mathbf{H}\tilde{\mathbf{v}} = -\mathbf{g}$ as $\tilde{\mathbf{v}} = \tilde{\mathbf{v}}_L + \tilde{\mathbf{v}}_H$, where $\tilde{\mathbf{v}}_L$ and $\tilde{\mathbf{v}}_H$ are found by solving

$$\mathbf{H}_L\tilde{\mathbf{v}}_L = (\mathbf{F}_L\mathbf{H}\mathbf{F}_L)\tilde{\mathbf{v}}_L = -\mathbf{F}_L\mathbf{g} \quad \text{and} \quad \mathbf{H}_H\tilde{\mathbf{v}}_H = (\mathbf{F}_H\mathbf{H}\mathbf{F}_H)\tilde{\mathbf{v}}_H = -\mathbf{F}_H\mathbf{g},$$

respectively.

We discuss how we use this decomposition to design a preconditioner, next. Let $\mathbf{s} \in \mathbf{R}^n$ denote the vector we apply our preconditioner to. Since we use an approximation of the inverse of $\mathbf{H}$, we have to design a scheme for approximately solving $\mathbf{H}\mathbf{u} = \mathbf{s}$. We find the smooth part of $\mathbf{u}$ by (iteratively) solving

$$\bar{\mathbf{H}}\bar{\mathbf{u}}_L = \mathbf{Q}_R\mathbf{F}_L\mathbf{s}, \tag{11}$$

where $\bar{\mathbf{H}} \in \mathbf{R}^{c,c}$ and $\bar{\mathbf{u}}_L \in \mathbf{R}^c$ represent coarse grid approximations of $\mathbf{H}_L$ and $\mathbf{u}_L$, respectively, and $\mathbf{Q}_R : \mathbf{R}^n \to \mathbf{R}^c$ is a restriction operator. We do not iterate on the oscilatory components of $\mathbf{s}$ (i.e., we replace $\mathbf{H}_H$ by $\mathbf{I}$). The solution $\mathbf{u}$ of $\mathbf{H}\mathbf{u} = \mathbf{s}$ is given by $\mathbf{u} = \mathbf{u}_L + \mathbf{u}_H \approx \mathbf{Q}_P\mathbf{F}_L\bar{\mathbf{u}}_L + \mathbf{F}_H\mathbf{s}$, where $\bar{\mathbf{u}}_L \approx \bar{\mathbf{H}}^{-1}\mathbf{Q}_R\mathbf{F}_L\mathbf{s}$ and $\mathbf{Q}_P : \mathbf{R}^c \to \mathbf{R}^n$ is a prolongation operator. We use spectral restriction and prolongation operators. The projection operators $\mathbf{F}_L$ and $\mathbf{F}_H$ are implemented as cut-off filters in the frequency domain.

An important aspect of our approach is that we do not apply our two-level preconditioner to the original Hessian $\mathbf{H}$. Since we can invert $\mathbf{H}_{\text{reg}} \succeq 0$ explicitly, we consider the (symmetric) regularization split-preconditioned system $(\mathbf{I} + \mathbf{H}_{\text{reg}}^{-1/2}\mathbf{H}_{\text{data}}\mathbf{H}_{\text{reg}}^{-1/2})\mathbf{w} = -\mathbf{H}_{\text{reg}}^{-1/2}\mathbf{g}$ instead, where $\mathbf{w} := \mathbf{H}_{\text{reg}}^{1/2}\tilde{\mathbf{v}}$. Notice, that the inverse of $\mathbf{H}_{\text{reg}}$ acts as a smoother. This allows us to get away with not treating high-frequency errors in our scheme. Our approach can be interpreted as an approximate two-level multigrid V-cycle with an explicit (algebraic) smoother given by $\mathbf{H}_{\text{reg}}^{-1/2}$.

The final questions are how to discretize and solve (11). We can use a Galerkin and a direct (non-Galerkin) discretization to implement the coarse grid operator $\bar{\mathbf{H}}$. Using the fact that $\mathbf{Q}_R$ and $\mathbf{Q}_P$ are adjoint operators, the Galerkin discretization is formally given by $\bar{\mathbf{H}} = \mathbf{Q}_R\mathbf{H}\mathbf{Q}_P$ [58, p. 75]. The drawback of using a Galerkin operator is that every matvec requires the solution of the incremental forward and adjoint equations on the fine grid. This is different if we directly discretize the matvec on the coarse grid. To save computational costs, we opt for this approach. For the iterative solver to approximately invert $\bar{\mathbf{H}}$ we have tested several variants, all of which are available in `CLAIRE`. We can use a nested PCG. This requires a tolerance $\epsilon_M > 0$ for the nested solver for the preconditioner that is only a fraction of the tolerance used to solve for the Newton step $\tilde{\mathbf{v}}$ on the fine grid, i.e., $\epsilon_M = \kappa\epsilon_H$ with $\kappa \in (0,1)$. This is due to the fact that Krylov subspace methods are nonlinear operators. We refer to this solver as PCG(κ). Another possibility is to use use a semi-iterative Chebyshev method [61] with a predefined number of iterations $k > 0$; this results in a fixed linear operator for a particular choice of eigenvalue bounds [56]. The eigenvalue bounds can be estimated using a Lanczos method. We refer to this strategy as CHEB(k). If we would like to use PCG with a fixed number of iterations as a nested solver, we can also replace the solver for the Newton step with a flexible Krylov subspace method [12, 115]. We observed that the performance deteriorated significantly as we reduce the regularization parameter. We disregard this approach.

3. Implementation and Software Aspects. We make CLAIRE [**?**] available under GPL license. CLAIRE is written in C/C++ and implements data parallelism via MPI. The source code can be downloaded from the github repository

https://github.com/andreasmang/claire.

The URL for the deployment page of CLAIRE is https://andreasmang.github.io/claire. Here, one can find a detailed documentation as well as use cases for CLAIRE. In what follows, we (*i*) describe implementation aspects, (*ii*) list features implemented in CLAIRE, and (*iii*) provide information relevant to potential users of CLAIRE. It is important to note that we will not be able to cover all implementation aspects, and we are continuously making improvements to our software. We refer to the README files in the code repository and the deployment page for updates and detailed information on how to compile, execute, and run CLAIRE on various systems.

3.1. Executables. CLAIRE has two main executables, `claire` and `clairetools`. The registration solver can be executed with the `claire` executable. The `clairetools` executable serves as a post-processing tool that allows users to, e.g., compute deformation measures (examples include the deformation map $\boldsymbol{y}$, the determinant of the deformation gradient, or a RAVENS map), or transport images or label maps for the evaluation of registration performance. We will keep adding features to these executables in future releases. Both executables provide a help message that briefly explains to users how control their behavior and how to set parameters. To access this help message, the user can simply execute the binaries without any parameters or add a `-help` flag to the executable (i.e., for instance execute `claire -help` from the command line window). The main output of `claire` is the computed velocity field. These fields can subsequently be used within `clairetools` to compute additional outputs. We explain the most common options for both executables in greater detail in the README files in the repository.

3.2. External Dependencies and IO. CLAIRE depends on four software packages. We use the PETSc library [14, 15] for linear algebra, and PETSc's TAO package [14, 111] for numerical optimization (TAO is included in PETSc). We use the AccFFT package [49,50]—a parallel, open-source FFT library for CPU/GPU architectures developed in our group, to apply spectral operators. AccFFT requires FFTW [47, 48]. We use niftilib [38] for IO. As such, CLAIRE currently supports IO of (uncompressed and compressed in gzip format) files in nifti-1 (`*.nii` or `*.nii.gz`) and Analyze 7.5 (`*.hdr` and `*.img`/`*.img.gz`) format. The default output format of CLAIRE is in `*.nii.gz`. We optionally support the PnetCDF format (`*.nc`) [92,114] for IO in parallel. The revision and version numbers for these libraries used in our experiments can be found in the references.

3.3. Compilation and Installation. Our solver supports single and double precision. (The precision is handed down by the PETSc library.) Our current software uses `make` for compilation. We provide scripts in the repository to download and compile the external libraries mentioned above using default settings that have worked most consistently on the systems we have executed CLAIRE on. Switches for controlling the precision are provided in the `makefile`. The user needs to compile PETSc and FFTW in single precision to be able to run CLAIRE in single precision. We have compiled, tested, and executed CLAIRE on HPC systems at TACC [135] (Stampede, Stampede 2, Lonestar 5, and Maverick), at HLRS (Hazelhen/CRAY XC40) [76] and at CACDS [35] (Opuntia and Sabine). Specifications of some of these systems can be found in §4.1. While we recommend the execution of CLAIRE on multi-core systems (to reduce the runtime), it is not a prerequisite to have access to HPC systems. CLAIRE has been successfully executed on personal computers and local compute servers with no inter-node communication. Large-scale systems are only required to significantly reduce the runtime or when considering large-scale applications (image sizes of 512^3 and beyond). We provide additional help for compilation and installation of CLAIRE in the repository.

3.4. Parallel Algorithms and Computational Kernels. The main computational kernels of CLAIRE are FFTs (spectral methods) and scattered data interpolation operations (semi-Lagrangian solver; see [52, 98, 99] for details). We use the AccFFT package [49, 50] to perform spectral operations (a software package developed by our group). This package dictates the data layout on multi-core systems: We partition the data based on a pencil decomposition for 3D FFTs [37,57]. Let $n_p = p_1 p_2$ denote the number of MPI tasks. Then each MPI task gets $(n_1/p_1) \times (n_2/p_2) \times n_3$ grid points. That is, we partition the domain $\boldsymbol{\Omega}$ of size $3 \times n_1 \times n_2 \times n_3$ along the x_1- and x_2-axis into subdomains $\boldsymbol{\Omega}_i$, $i = 1, \ldots, n_p$, of size $3 \times (n_1/p_1) \times (n_2/p_2) \times n_3$. There is no partitioning in time.

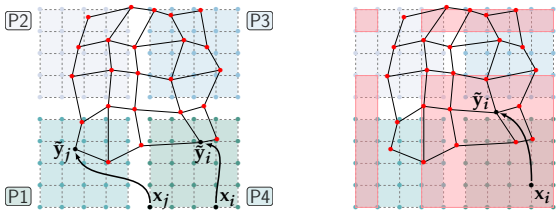

FIG. 2. *2D illustration of the data layout and the communication steps for the evaluation of the interpolation kernel. The original grid at timepoint t^{k+1} is distributed across $n_p = 4$ processors Pi, $i = 1,2,3,4$. To solve the transport problem using a semi-Lagrangian scheme, we have to trace a characteristic for each grid point $\mathbf{x}_l$ backward in time. This requires a scattered data interpolation step. The deformed configuration of the grid (i.e., the departure points) originally owned by P4 (red points) are displayed in overlay. We illustrate three scenarios: The departure point is located (i) on P4 (left; $\mathbf{x}_i \to \tilde{\mathbf{y}}_i$), (ii) on a different processor P1 (left; $\mathbf{x}_j \to \tilde{\mathbf{y}}_j$), and (iii) between processors P3 and P4 (right). For the first case, no communication is required. The second case requires the communication of $\tilde{\mathbf{y}}_j$ to P1, and the communication of the interpolation result back to P4. For the third case, we add a ghost layer with a size equal to the support of the interpolation kernel (4 grid points in our case) to each processor; the evaluation of the interpolation happens on the same processor (like in the first case). Notice that the communication of the departure points (for the forward and backward characteristics) needs to be performed only once per Newton iteration, since our velocity field is stationary.*

The scalability of the 3D FFT is well explored [37, 50, 57]. We refer to [50, 97] for scalability results for `AccFFT`. If we assume that the number of grid points n_i, $i = 1,2,3$, is equal along each spatial direction, i.e., $\tilde{n} = n_1 = n_2 = n_3$, each 3D FFT requires $\mathcal{O}(3\tilde{n}\log(\tilde{n})/2n_p)$ computations and $\mathcal{O}(2\sqrt{n_p}t_s + (2\tilde{n}^3/n_p)t_w)$ communications, where $t_s > 0$ is the startup time for the data transfer and $t_w > 0$ represents the per-word transfer time [57].

The parallel implementation of our interpolation kernel is introduced in [99] and improved in [52]. We use a tricubic interpolation model to evaluate off-grid points in our semi-Lagrangian scheme (see [98, 99] for a detailed description of our solver). The polynomial is implemented in Lagrange form. The evaluation of the interpolation kernel requires the computation of twelve basis polynomials. The local support of the cubic basis is 4^3 grid points. Overall, this results in a complexity of $\mathcal{O}(256\tilde{n}^3/n_p)$ computations. We have implemented a SIMD vectorization based on advanced vector extensions (AVX2) for Haswell architectures for the evaluation of the interpolation kernel (available for single precision only). Compared to our initial work in [99] our method is now bound by communication time instead of time spent in the interpolation. The communication costs are more difficult to estimate; they not only depend on the data layout but also on the characteristics obtained for a given velocity field. If a departure point is owned by the current processor, we require no communication. If the values for a departure point are owned by another processor/MPI task (the "*worker*"), we communicate the coordinates from the "*owner*" to the "*worker.*" We then evaluate the interpolation model on the "*worker*" and communicate the result back to the "*owner.*" This results in a communication cost of $4t_w$ per off-grid point not owned by a processor. To evaluate the interpolation model at off-grid points not owned by any MPI task (i.e., located in-between the subdomains Ω_i), we add a layer of four ghost points (scalar values to be interpolated; see Fig. 2, right). This results in an additional communication cost of $n_s(2n_3(n_1/p_1 + n_2/p_2)t_w) + 4t_s$ for each MPI task for the four face neighbors, where n_s is the size of layer for the ghost points (in our case four). The communication with the four corner neighbors can be combined with the messages of the edge neighbors, by appropriate ordering of the messages. Notice that the communication of the departure points (for the forward and backward characteristics) needs to be performed only once per Newton iteration, since our velocity field is stationary. We perform this communication when we evaluate the forward and the adjoint operators, i.e., during the evaluation of the objective functional and the reduced gradient.

3.5. Memory Requirements. In our most recent implementation, we have reduced the memory footprint for the Gauss–Newton approximation; we only store the time history of the state and incremental state variable. This is accomplished by evaluating the time integrals that appear in the reduced gradient in (5) and the Hessian matvec in (7) simultaneously with the time integration of the adjoint and incremental adjoint equations (4) and (9), respectively. With this we can reduce the memory pressure from $\mathcal{O}((2n_t + 8)n_1n_2n_3)$ (full Newton) to $\mathcal{O}((n_t + 7)n_1n_2n_3)$ (Gauss–Newton) for the gradient (see (5)) and from $\mathcal{O}((4n_t + 13)n_1n_2n_3)$ (full Newton) to $\mathcal{O}((n_t + 10)n_1n_2n_3)$ (Gauss–Newton) for the Hessian matvec (see (7)), respectively. Notice that we require $0.5\times$ the memory of the Hessian matvec, if we consider the

TABLE 2
Parameters available in `CLAIRE` *(there are more, but these are the critical ones).*

variable	meaning	suggested value	determined automatically
β_v	regularization parameter for v	—	yes
β_w	regularization parameter for w	1e−4	no
ϵ_g	relative tolerance for gradient	5e−2	no
n_t	number of time steps	4	no
ϵ_j	bound for $\det \nabla \boldsymbol{y}^{-1}$	0.25 (H^1-div) or 0.1 (H^2)	no

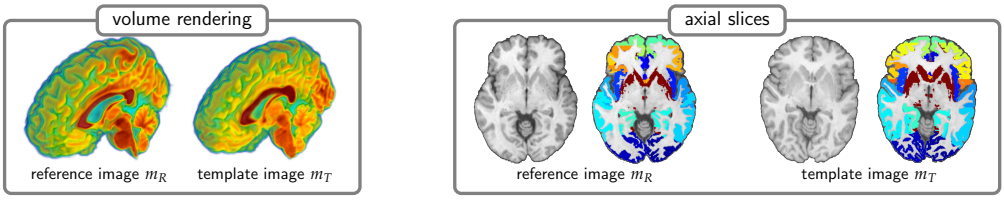

FIG. 3. *Illustration of exemplary datasets from the* `NIREP` *dataset [32]. Left: Volume rendering of an exemplary reference image $m_R(\boldsymbol{x})$ (dataset* `na01`*) and an exemplary template image $m_T(\boldsymbol{x})$ (dataset* `na03`*), respectively. Right: Axial slice for these datasets together with label maps associated with these data. Each color corresponds to one of the 32 individual anatomical gray matter regions that serve as a ground truth to evaluate the registration performance.*

two-level preconditioner. The spectral preconditioner does not add to the memory pressure.

3.6. Additional Software Features. We provide schemes for automatically selecting an adequate regularization parameter. This a topic of research by itself [66, 68]. Disregarding theoretical requirements [17, 40, 137], one in practice typically selects an adequate regularization norm based on application requirements. From a practical point of view we are interested in computing velocities for which the determinant of the deformation map does not change sign/is strictly positive for every point inside the domain. This guarantees that the transformation is locally diffeomorphic (subject to numerical accuracy). Consequently, we determine the regularization parameter β_v for the Sobolev norm for the velocity based on a binary search (this strategy was originally proposed in [96]; a similar strategy is described in [66]). We control the search based on a bound for the determinant of the deformation gradient. That is, we choose β_v so that the determinant of the deformation gradient is bounded below by ϵ_J and bounded above $1/\epsilon_J$, where $\epsilon_J \in (0, 1)$ is a user defined parameter. This search is expensive, since it requires a repeated solution of the inverse problem. (For each trial β_v we iterate until we meet the convergence criteria for our Newton solver and then use the obtained velocity as an initial guess for the next β_v.) We assume that, once we have found an adequate β_v, we can use this parameter for similar registration problems. Such cohort studies are quite typical in medical imaging.

`CLAIRE` features several well established schemes to accelerate the rate of convergence and reduce the likelihood to get trapped in local minima. The user can choose between (*i*) parameter continuation in β_v (starting with a default value of $\beta_v = 1$ we reduce β_v until we reach a user defined parameter $\beta_v^\star$; we found this scheme to perform best), (*ii*) grid continuation, i.e., a coarse-to-fine multi-resolution scheme with a smoothing of $\sigma = 1$ voxels (consequently, the standard deviation increases for coarser grids), and (*iii*) scale continuation using a scale-space representation of the image data (again, coarse-to-fine).

We summarize the critical parameters of `CLAIRE` in Tab. 2.

4. Experiments. We evaluate the registration accuracy for 16 segmented MRI brain volumes [32]. Details on the considered datasets can be found in §4.2. We showcase two exemplary datasets in Fig. 3. Notice, that these datasets have been rigidly preregistered. We directly apply our method to this data (without an additional affine preregistration step). The runs were executed on the CACDS's Opuntia server or on TACC's Lonestar 5 system. The specs of these systems can be found below. Notice that we accompany this document with *supplementary material* that provides more detailed results for some of the experiments conducted in this study.

4.1. Setup, Implementation, and Hardware. We execute the runs on CACDS's Opuntia system (Intel ten-core Xeon E5-2680v2 at 2.80 GHz with 64 GB memory; 2 sockets for a total of 20 cores [35]) and

TACC's Lonestar 5 system (2-socket Xeon E5-2690 v3 (Haswell) with 12 cores/socket, 64 GB memory per node [135]). Our code is written in C++ and uses MPI for parallelism. It is compiled with the default Intel compilers available on these systems at the time (Lonestar 5: Intel 16.0.1 and Cray MPICH 7.3.0; Opuntia: Intel PSXE 2016, Intel ICS 2016, and Intel MPI 5.1.1). We use `CLAIRE` commit `v0.07-131-gbb7619e` to perform the experiments. For the software packages/libraries used in combination with `CLAIRE` we refer to §3. The versions of the libraries used for our runs can be found in the references.

4.2. Real and Synthetic Data. We report results for the `NIREP` ("*Non-Rigid Image Registration Evaluation Project*") data [32]. This repository contains 16 rigidly aligned T1-weighted MRI brain datasets (`na01`–`na16`; image size: $256 \times 300 \times 256$ voxels) of different individuals. Each dataset comes with 32 labels of anatomical gray matter regions. (Additional information on the data sets, the imaging protocol, and the pre-processing can be found in [32].) We illustrate an exemplary dataset in Fig. 3. The initial Dice score (before registration) for the combined label map (i.e., the union of the 32 individual labels) is on average 5.18e−1 (mean) with a maximum of 5.62e−1 (dataset `na08`) and a minimum of 4.38e−1 (dataset `na14`). We generate the data for grids not corresponding to the original resolution based on a cubic interpolation scheme.

The scalability runs reported in §5.2 are based on synthetic test data. We use a template image $m_T(\boldsymbol{x}) = ((\sin x_1)(\sin x_1) + (\sin x_2)(\sin x_2) + (\sin x_3)(\sin x_3))/3$. The reference image $m_R(\boldsymbol{x})$ is computed by solving the forward problem for a predefined velocity field $\boldsymbol{v}^\star(\boldsymbol{x}) = (v_1^\star(\boldsymbol{x}), v_2^\star(\boldsymbol{x}), v_3^\star(\boldsymbol{x}))^\mathsf{T}$, where $v_1^\star(\boldsymbol{x}) = \sin x_3 \cos x_2 \sin x_2$, $v_2^\star(\boldsymbol{x}) = \sin x_1 \cos x_3 \sin x_3$, and $v_3^\star(\boldsymbol{x}) = \sin x_2 \cos x_1 \sin x_1$.

4.3. Convergence: Preconditioner. We study the performance of different variants of our preconditioner for the reduced space Hessian.

Setup. We solve the KKT system in (7) at a true solution $\boldsymbol{v}^\star$. This velocity $\boldsymbol{v}^\star$ is found by registering two neuroimaging data sets from the `NIREP` dataset (`na01` and `na02`). The images are downsampled to a resolution of $128 \times 150 \times 128$ (half the original resolution). We consider an H^1-div regularization model with $\beta_v = 1\text{e}{-2}$ and $\beta_w = 1\text{e}{-4}$ and an H^2 regularization model with $\beta_v = 1\text{e}{-4}$ with a tolerance $\epsilon_g = 1\text{e}{-2}$ to compute $\boldsymbol{v}^\star$. Once we have found $\boldsymbol{v}^\star$, we generate a synthetic reference image m_R by transporting the reference image using $\boldsymbol{v}^\star$. We use the velocity $\boldsymbol{v}^\star$ as an initial guess for our solver, and iteratively solve for the search direction $\tilde{\boldsymbol{v}}$ using different variants of our preconditioner. The number of time steps for the PDE solves is set to $n_t = 4$. We fix the tolerance for the (outer) PCG method to $\epsilon_H = 1\text{e}{-3}$. We consider an inexact Chebyshev semi-iterative method with a fixed number of $k \in \{5, 10, 20\}$ iterations (denoted by CHEB(k)) and a nested PCG method with a tolerance of $\epsilon_P = 1\text{e}{-1}\epsilon_H$ (denoted by PCG(1e−1)) for the iterative inversion of the preconditioner. Details can be found in §2. We compare these strategies to a spectral preconditioner (inverse of the regularization operator $\mathcal{A}$; used in [52, 99, 100]). We study the rate of convergence of the PCG solver for a vanishing regularization parameter β_v. We consider mesh sizes of $128 \times 150 \times 128$ and $256 \times 300 \times 256$. We execute `CLAIRE` on a single node of Opuntia with 20 MPI tasks.

Results. We display the trend of the residual with respect to the (outer) PCG iterations in Fig. 4 (H^2-seminorm for $\boldsymbol{v}$, i.e., $\mathcal{A} = -\Delta$, with $\beta_v \in \{1\text{e}{-2}, 5\text{e}{-3}, 1\text{e}{-3}, 5\text{e}{-4}, 1\text{e}{-4}\}$) and in Fig. 5 ($H^1$-div regularization model with an H^1-seminorm for $\boldsymbol{v}$, i.e., $\mathcal{A} = \nabla$ with penalty for $\nabla \cdot \boldsymbol{v}$, with $\beta_v \in \{1\text{e}{-1}, 5\text{e}{-2}, 1\text{e}{-2}, 5\text{e}{-3}\}$ and $\beta_w = 1\text{e}{-4}$), respectively. Detailed results for these runs can be found in Tab. SM1 and Tab. SM2 in the supplementary material.

Observations. The most important observations are:

- The PCG method converges significantly faster for the two-level preconditioner.
- The performance of all preconditioners considered in this study is *not* independent of the regularization parameter β_v. The workload increases significantly for vanishing regularity of the velocity $\boldsymbol{v}$ for all preconditioners. The plots in Fig. 4 and Fig. 5 imply that the convergence of the PCG method for the two-level preconditioner is less sensitive to (or even independent of) β_v. The work goes to the inversion of the reduced space Hessian on the coarse grid (cf. Tab. SM1 and Tab. SM2 in the supplementary material for details). If we further reduce the regularization parameter (below 1e−5 for the H^2-regularization model and below 1e−4 for the H^1-div regularization model) the performance of our preconditioners deteriorates further; the runtime becomes impractical for all preconditioners.
- The rate of convergence of the PCG method is (almost) independent of the mesh size for all preconditioners. We note that we apply a smoothing of $\sigma = 2$ along each spatial dimension so that the input image data is resolved on the coarse grid of size $128 \times 150 \times 128$. The same frequency content

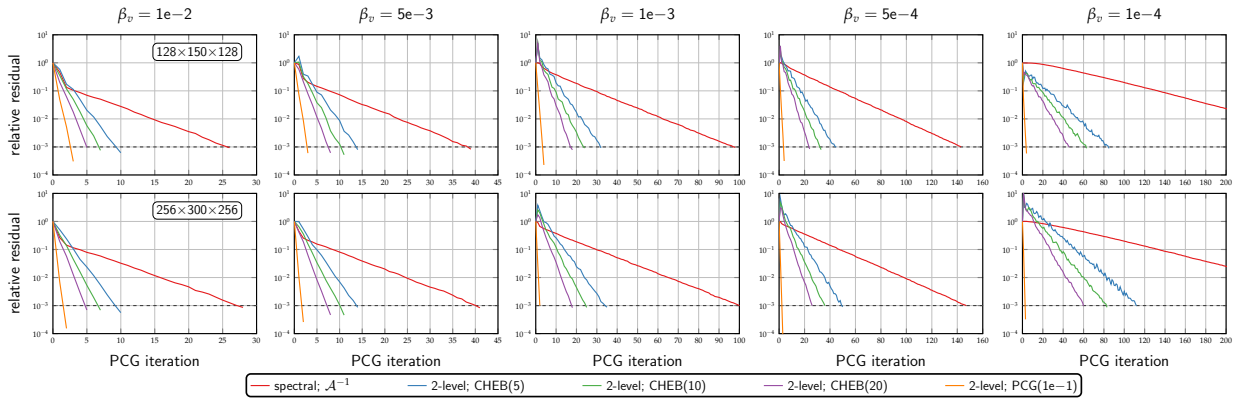

FIG. 4. *Convergence of Krylov solver for different variants of the preconditioner for the reduced space Hessian. We consider an H^2-seminorm as regularization model for the velocity. We report results for different regularization weights $\beta_v \in \{1e{-}2, 5e{-}3, 1e{-}3, 5e{-}4, 1e{-}4\}$. We report the trend of the relative residual for the outer Krylov method (PCG) versus the iteration count. We report results for the spectral preconditioner and the two-level preconditioner. We use different iterative algorithms to compute the action of the inverse of the preconditioner: CHEB(k) with $k \in \{5, 10, 20\}$ refers to a CHEB method with a fixed number of k iterations; PCG(1e−1) refers to a PCG method with a tolerance that is 0.1× smaller than the tolerance used for the outer PCG method.*

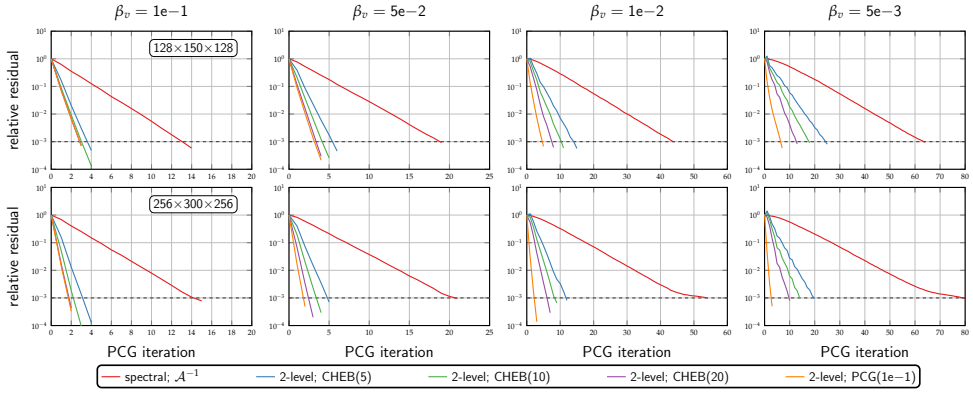

FIG. 5. *Convergence of Krylov solver for different variants of the preconditioner for the reduced space Hessian. We consider an H^1-div regularization model with an H^1-seminorm for the velocity. We report results for different regularization weights $\beta_v \in \{1e{-}1, 5e{-}2, 1e{-}2, 5e{-}3\}$. We set $\beta_w = 1e{-}4$. We report the trend of the relative residual for the outer Krylov method (PCG) versus the iteration count. We report results for the spectral preconditioner and the two-level preconditioner. We use different algorithms to compute the action of the inverse of the preconditioner: CHEB(k) with $k \in \{5, 10, 20\}$ refers to a CHEB method with a fixed number of k iterations; PCG(1e−1) refers to a PCG method with a tolerance that is 0.1× smaller than the tolerance used for the outer PCG method.*

is presented to the solver on the fine grid of size $256 \times 300 \times 256$.

- The PCG method converges significantly faster if we consider an H^1-regularization model for $\boldsymbol{v}$. This is a direct consequence of fact that the condition number of the Hessian increases with the order of the regularization operator $\mathcal{A}$.
- The differences of the performance of the preconditioners are less pronounced for an H^1-div regularization model for $\boldsymbol{v}$ than for an H^2-regularization model. For an H^2 regularization model with $\beta_v = 1e{-}4$ we require more than 200 iterations for the spectral preconditioner.
- Considering runtime (not reported here), we obtain a speedup of up to 2.9 for the H^2-regularization model (see run #20 in Tab. SM1 in the supplementary material) and a speedup of up to 2.6 for the H^1-div regularization model (see run #40 in Tab. SM2 in the supplementary material). The coarser the grid, the less effective is the two-level preconditioner, especially for vanishing regularization parameters β_v. This is expected, since we cannot resolve high-frequency components of the fine level on the coarse level. Secondly, we do not use a proper (algorithmic) smoother in our scheme to reduce the high-frequency errors.
- The performance of the CHEB and the nested PCG method for iteratively inverting the reduced space

Hessian are similar. There are differences in terms of the mesh size. For a coarser grid ($128 \times 150 \times 128$) the CHEB seems to perform slightly better. For a grid size of $256 \times 300 \times 256$ the nested PCG method is slightly better.

Conclusions. (*i*) The two-level preconditioner is more effective than the spectral preconditioner. (*ii*) The nested PCG method is more effective than the CHEB method on a finer grid (and does not require a repeated estimation of the spectrum of the Hessian operator). (*iii*) The PCG method converges faster if we consider an H^1-div regularization model for $\boldsymbol{v}$. (*iv*) Designing a preconditioner that delivers a good performance for vanishing regularization parameters requires more work.

4.4. Convergence: Newton–Krylov Solver. We study the rate of convergence of our Newton–Krylov solver for the entire inversion. We consider the neuroimaging data described in §4.2. We report additional results for a synthetic test problem (ideal case) in the supplementary material.

Setup. We register the datasets `na02` through `na16` (template images) with `na01` (reference image). We execute the registration in full resolution ($256 \times 300 \times 256$; 58 982 400 unknowns). We consider and H^1-div regularization model (H^1-seminorm for $\boldsymbol{v}$ with $\beta_v = 1\text{e}{-2}$ and $\beta_w = 1\text{e}{-4}$; the parameters are chosen empirically). The number of Newton iterations is limited to 50 (not reached). The number of Krylov iterations is limited to 100 (not reached). We use a tolerance of $5\text{e}{-2}$ and $1\text{e}{-6}$ (the latter is not reached) for the relative reduction and the absolute ℓ^2-norm of the reduced gradient as a stopping criterion, respectively. We use $n_t = 4$ time steps for numerical time integration. We compare results obtained for the two-level preconditioner to results obtained using a spectral preconditioner (inverse of the regularization operator). We use a nested PCG method with a tolerance of $\epsilon_P = 0.1\epsilon_H$ for computing the action of the inverse of the two-level preconditioner. We do not perform any parameter, scale, or grid continuation. (We note that we observed that these continuation schemes are critical when performing runs for smaller regularization parameters.) We compare results obtained for single (32 bit) and double (64 bit) precision. We execute these runs on TACC's Lonestar 5 system (see §4.1 for specs).

Results. We show convergence plots for all datasets in Fig. 6. We plot the relative reduction of the mismatch (left column), the relative reduction of the gradient (middle column), and the relative reduction of the objective functional (right column) with respect to the Gauss–Newton iterations. The top row shows results for the spectral preconditioner; the other two rows show results for the two-level preconditioner (middle row: double precision; bottom row: single precision). The run time for the inversion is reported in the plot at the top right of Fig. 6. An exemplary trend for the residual of the PCG method per Gauss–Newton iteration is displayed at the bottom right of Fig. 6. These plots summarize results reported in the supplementary material; results for the spectral preconditioner are reported in Tab. SM3; results for the two-level preconditioner are reported in Tab. SM4 (double precision) and Tab. SM5 (double precision). We also report a comparison of the performance of our solver for single (32 bit) and double (64 bit) precision in Tab. SM6 for two exemplary images of the `NIREP` dataset.

Observations. The most important observations are:

- Switching from double to single precision does not affect the convergence of our solver (see Fig. 6; detailed results are reported in Tab. SM6 in the supplementary material).
- The two-level preconditioner executed with single precision yields a speedup of up to $6\times$ (with an average speedup of 4.4 ± 0.8) compared to our baseline method (spectral preconditioner executed in double precision) [52, 99] (see Fig. 6 top right). Switching from single to double precision yields a speedup of more than $2\times$ (detailed results are reported in Tab. SM6 in the supplementary material).
- The average runtime of our improved solver is $85\,\text{s} \pm 22\,\text{s}$ with a maximum of 140 s (see run #13 in Tab. SM5 in the supplementary material for details) and a minimum of 56 s (see run #7 in Tab. SM5 in the supplementary material for details).
- We obtain a very similar convergence behavior for the outer Gauss–Newton iterations for different variants of our solver (see Fig. 6). We can reduce the ℓ^2-norm of the gradient by $5\text{e}{-2}$ in 6 to 14 Gauss–Newton iterations (depending on the considered pair of images).
- The mismatch between the deformed template image and the reference image stagnates once we have reduced the gradient by more than one order of magnitude (for the considered regularization weight).
- We oversolve the reduced space KKT system if we consider a superlinear forcing sequence in combination with a nested PCG method (see Fig. 6 bottom right). This is different for synthetic data (we report exemplary results in the supplementary material).

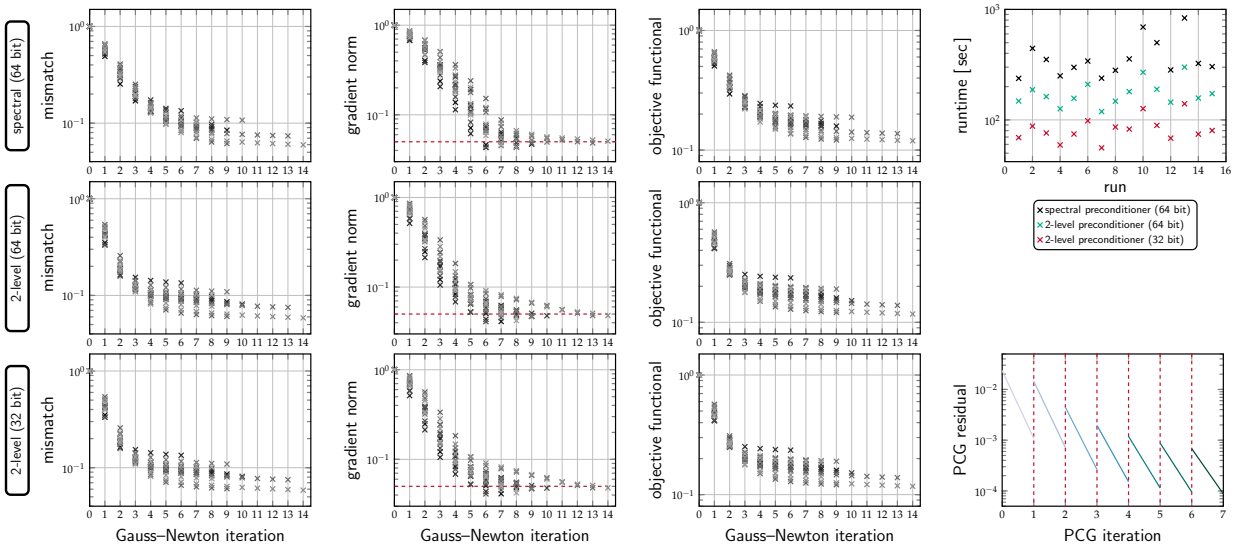

FIG. 6. *Convergence of* CLAIRE*'s Newton–Krylov solver for neuroimaging data for different realizations of the preconditioner. Top row: inverse regularization operator. Middle and bottom row: two-level preconditioner using PCG(*1e−1*) for double (*64 bit*; middle row) and single (*32 bit*; bottom row) precision, respectively. We report results for 15 multi-subject brain registration problems (*na02 *through* na16 *of the* NIREP *repository registered to* na01*). Each of these 15 registrations is plotted in a different shade of gray. We plot (from left to right) the relative reduction of (i) the mismatch (squared L^2-distance between the images to be registered), (ii) the reduced gradient, and (iii) the objective functional, with respect to the Gauss–Newton iterations. We use a relative change of the gradient of* 5e−2 *as a stopping criterion (dashed red line in second column). We also report the run time in seconds for each registration problem (right plot at top) and an exemplary plot of the reduction of the residual of the (outer) PCG solver per Newton iteration (right plot at bottom; the Newton iterations are separated by vertical dashed lines). The runs are performed on one node of TACC's Lonestar 5 system. The results reported here correspond to those in Tab. SM3, Tab. SM4, and Tab. SM5 in the supplementary material.*

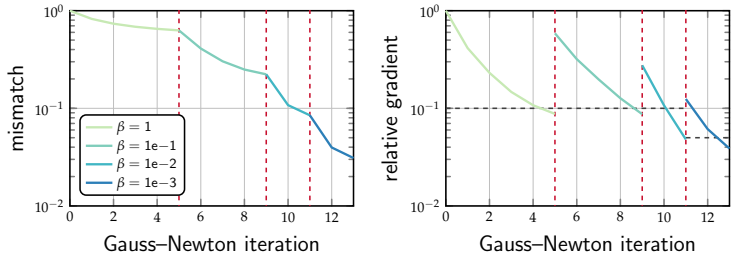

FIG. 7. *Convergence results for the parameter continuation scheme implemented in* CLAIRE*. We report results for the registration of* na03 *to* na01 *(run #8 in Tab. ??). We report the reduction of the mismatch (left) the reduced gradient (right) per level versus the number of Gauss–Newton iterations. The individual levels are separated by vertical, dashed lines. The horizontal dashed line in the right plot shows the tolerance for the relative reduction of the gradient for the inversion.*

Conclusions. (*i*) Our improved implementation of CLAIRE yields an overall speedup of 4× for real data if executed on a single resolution level. (*ii*) Executing CLAIRE in single precision does not deteriorate the performance of our solver (if we consider an H^1-regularization model for the velocity).

5. Time-to-Solution. We study the performance of CLAIRE. We note that the Demons algorithm requires between approximately 30 s (3 levels with 15, 10, and 5 iterations) and 3600 s (3 levels with 1500, 1000 and 500 iterations) until ''convergence'' on the same system (depending on the parameter choices; this includes a grid continuation scheme; see §5.1 for details).

REMARK 2. *Since we perform a fixed number of iterations for the* Demons *algorithm, the runtime only depends on the execution time of the operators. The regularization parameters control the support of the Gaussian smoothing operator; the larger the parameters, the longer the execution time. This is different for* CLAIRE*; large regularization parameters result in fast convergence and, hence, yield a short execution time. A simple strategy to obtain competitive results in terms of runtime would be to also execute* CLAIRE *for a fixed number of iterations. We prefer to use a tolerance for the relative reduction of the gradient, instead, since it yields consistent results across different datasets.*

Setup. We use the dataset na02 and na03 as a template images, and register them to na01 (reference image). We consider and H^1-div regularization model (H^1-seminorm for $\boldsymbol{v}$ with $\beta_v \in \{1e{-}2, 1e{-}3\}$ and

TABLE 3

We compare different schemes implemented in CLAIRE for stabilizing and accelerating the computations. We consider two datasets as a template image (na03 and na10). We use an H^1-div regularization model with $\beta_w = 1e{-}4$. We consider regularization parameters $\beta_v = 1e{-}2$ and $\beta_v = 1e{-}3$. We execute the inversion with a spectral preconditioner (double precision) to establish a baseline (run #1, run #6, run #11, and run #16; corresponds to the method presented in [99]). The remaining results are obtained with a two-level preconditioner using a nested PCG method with a tolerance of $1e{-}1\epsilon_H$ to compute the action of the inverse of the preconditioner. For each dataset and each choice of β_v we report results for (from top to bottom) for a two-level preconditioner without any accelerations and a parameter continuation scheme (PC) scheme. We report (from left to right) the number of Gauss–Newton iterations per level (#iter; the total number for the entire inversion is the sum), the number of Hessian matvecs per level (#matvecs; the total number for the entire inversion is the sum), the number of PDE solves (on the fine grid; #PDE), the relative reduction of the mismatch, the absolute ℓ^2-norm of the reduced gradient ($\|\mathbf{g}^\star\|_2$), and the relative ℓ^2-norm of the reduced gradient after convergence ($\|\mathbf{g}^\star\|_{rel}$). We also report the runtime (in seconds) as well as the speedup compared to our baseline method presented in [99].

		β_v		#iter	#matvecs	#PDE	mismatch	$\|\mathbf{g}^\star\|_2$	$\|\mathbf{g}^\star\|_{\text{rel}}$	runtime	speedup
#1	na03	1e−2	—	9	83	187	8.47e−2	4.63e−4	4.71e−2	6.05e+2	
#2			—	9	9	39	8.60e−2	4.65e−4	4.73e−2	1.22e+2	5.0
#3			PC	4,3,2	4,3,2	46	9.84e−2	8.66e−4	4.77e−2	9.33e+1	6.5
#4		1e−3	—	7	128	273	2.88e−2	3.97e−4	4.94e−2	8.97e+2	
#5			—	12	12	73	2.56e−2	3.72e−4	4.63e−2	7.17e+2	1.3
#6			PC	4,3,2,2	4,3,2,2	56	3.37e−2	8.25e−4	4.55e−2	1.61e+2	5.6
#7	na10	1e−2	—	7	52	121	9.67e−2	4.98e−4	4.91e−2	3.84e+2	
#8			—	7	7	31	9.62e−2	4.99e−4	4.92e−2	9.35e+1	4.1
#9			PC	3,3,2	3,3,2	42	1.10e−1	9.55e−4	4.98e−2	9.04e+1	4.2
#10		1e−3	—	7	134	285	3.17e−2	3.46e−4	4.24e−2	1.04e+3	
#11			—	8	16	51	3.11e−2	3.85e−4	4.73e−2	4.78e+2	2.2
#12			PC	3,3,2,2	3,3,2,3	54	3.78e−2	7.41e−4	3.86e−2	1.87e+2	5.6

$\beta_w = 1e{-}4$; these parameters are chosen empirically). The number of Newton iterations is limited to 50 (not reached). The number of Krylov iterations is limited to 100 (not reached). We use a tolerance of 5e−2 for the relative reduction of the ℓ^2-norm of the gradient and a tolerance of 1e−6 (not reached) for its ℓ^2-norm as a stopping criterion. We use $n_t = 4$ time steps for numerical time integration. We compare results obtained for the two-level preconditioner (runs executed in single precision) to results obtained using a spectral preconditioner (inverse of the regularization operator; runs executed in double precision; the baseline method described in [99]). We use a nested PCG method with a tolerance of $\epsilon_P = 1e{-}1\epsilon_H$ for computing the action of the inverse of the two-level preconditioner. We execute CLAIRE using a parameter continuation scheme. That is, we reduce the regularization weight be one order of magnitude (starting with $\beta_v = 1$) until we reach the target regularization parameter. We execute these runs on one node of the Opuntia system using 20 MPI tasks (see §4.1 for specs).

Results. We report the results in Tab. 3. We report the number of Gauss–Newton iterations, the number of Hessian matrix vector products (per level), the number of PDE solves (per level), the relative reduction of the mismatch, the ℓ^2-norm of the reduced gradient, the relative reduction of the ℓ^2-norm of the gradient, the run time, and the associated speedup compared to a full solve disregarding any acceleration schemes. We showcase the trend of the mismatch and the ℓ^2-norm of the gradient for different levels of the parameter continuation scheme in Fig. 7. We show exemplary registration results for the parameter continuation in Fig. 8 (for the registration of na10 to na01).

Observations. The most important observations are:

- The parameter continuation scheme in β_v yields a speedup between 4× and 6× (run #3, run #6, run #9, and run #12 in Tab. 3) even if we reduce the target regularization parameter from 1e−2 to 1e−3. The runtime range between 9.04e+1 s (run #9) and 1.87e+2 s (run #12) depending on problem and parameter selection.
- The results obtained for the different schemes are qualitatively and quantitatively very similar. We obtain similar values for the relative mismatch, e.g., between 1.10e−1 and 9.62e−2 for $\beta_v = 1e{-}2$ and between 3.78e−2 and 3.11e−2 for $\beta_v = 1e{-}3$ for the registration of na10 to na01.

Conclusions. (*i*) Introducing the parameter continuation stabilizes the computations (similar results can be observed for grid and scale continuation schemes; not reported here). While the speedup for the preconditioner deteriorates as we reduce β_v (see, e.g., run #2 and run #5 in Tab. 3), we can observe a speedup of about 5× for the parameter continuation scheme irrespective of β_v. We note that for small regularization parameters it is critical to execute CLAIRE using a parameter continuation scheme. That is, for certain problems we observed a stagnation in the reduction of the gradient if CLAIRE is executed

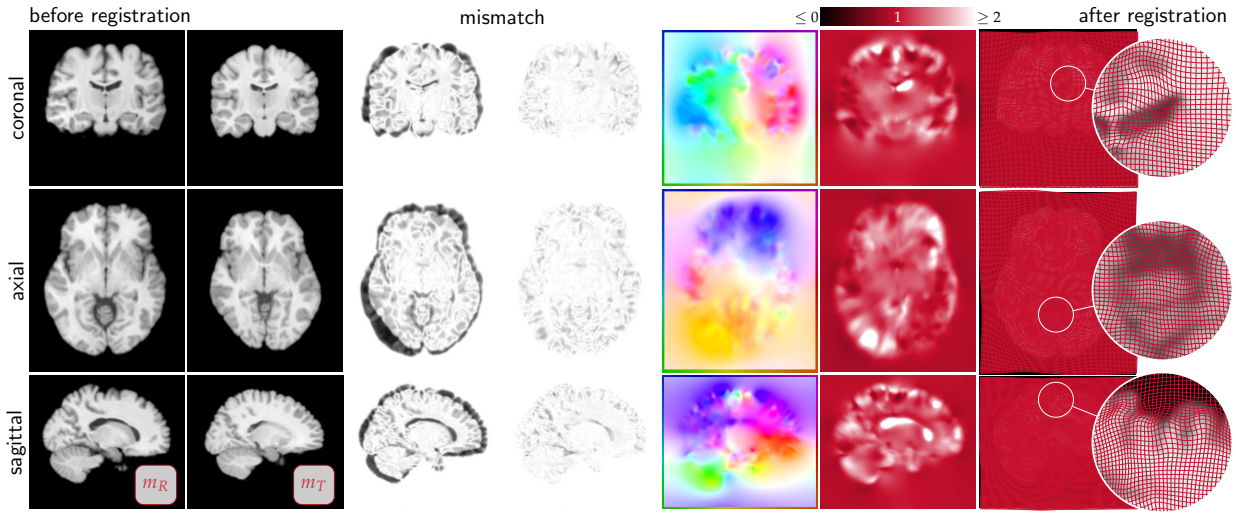

FIG. 8. *Exemplary results for the parameter continuation scheme implemented in* CLAIRE. *We consider the datasets* na10 *(template image) to* na01 *(reference image). We show (from top to bottom) coronal, axial and sagittal slices. The three columns on the left show the original data (left: reference image m_R; middle: template image m_T; right: mismatch between m_R and m_T before registration). The four columns on the right show results for the parameter continuation scheme (run #9 in Tab. 3; from left to right: mismatch between m_R and m_1 (after registration); a map of the orientation of $\boldsymbol{v}$; a map of the determinant of the deformation gradient (the color bar is shown at the top); and a deformed grid illustrating the in plane components of $\boldsymbol{y}$).*

without a parameter continuation scheme for small regularization parameters. We attribute this behavior to the accumulation of numerical errors in our scheme. This observation requires further exploration. (*ii*) Depending on the desired mismatch and regularity requirements, we achieve a runtime that is almost competitive with the Demons algorithm using the same system (i.e., the same number of cores). We expect to obtain an additional speedup by combining grid and parameter continuation. Designing an effective algorithm that combines these two approaches requires more work.

5.1. Registration Quality. We study registration accuracy for multi-subject image registration problems based on the NIREP dataset (see §4.2). We compare results for our method to different variants of the diffeomorphic Demons algorithm.

Setup. We consider the entire NIREP data repository. We register the dataset na02 through na16 (template images) to na01 (reference image). The data has been rigidly preregistered [32]. We do not perform an additional affine preregistration step. Each dataset comes with a label map that contains 32 labels (ground truth segmentations) identifying distinct gray matter regions (see Fig. 3 for an example). We quantify registration accuracy based on the Dice coefficient (the optimal value is one) for these labels after registration. For ease of presentation we limit the evaluation to the union of the 32 labels (we report results for the individual 32 labels for CLAIRE in Fig. SM2 of the supplementary material). We assess the regularity of the computed deformation map based on the extremal values for the determinant of the deformation gradient. The analysis is limited to the foreground of the reference image (i.e., the area occupied by brain, identified by thresholding using a threshold of 0.05). We compare the performance of our method against different variants of the diffeomorphic Demons algorithm. We execute all runs on one node of the Opuntia system using 20 MPI tasks (see §4.1 for specs).

- *Demons:* We consider (*non-*)*symmetric diffeomorphic* ((**S**)**DDEM**; *diffeomorphic update rule*) [140, 142], and the (*non-*)*symmetric log-domain diffeomorphic* Demons algorithm ((**S**)**LDDDEM**; (*symmetric*) *log-domain update rule*) [141]. We have tested different settings for these methods (see below). We limit our study to the default parameters suggested in the literature, online resources, and the manual of the software. We use the code available at [78]. We compile in release mode, with the -O3 option. The code has been linked against ITK version 4.9.1 [79, 85]. Notice that the implementation uses multi-threading based on pthreads to speed up the computations. We use the default setting, which corresponds to the number of threads being equal to the number of cores of the system. We use the *symmetrized force* for the symmetric strategies. We consider the *gradient of the deformed template* as a force for the non-

symmetric strategies. We use a nearest-neighbor interpolation model to transform the label maps. We perform various runs to identify adequate parameters. For the first set of runs we use a three-level grid continuation scheme with 15, 10, and 5 iterations per level (the default), respectively. We estimate an optimal combination of regularization parameters $\sigma_u \geq 0$, $\sigma_d \geq 0$, and $\sigma_v \geq 0$ based on an exhaustive search. This search is limited to the datasets `na01` (reference image) and `na02` (template image). We define the optimal regularization parameter to be the one that yields the highest Dice score subject to the map $\boldsymbol{y}$ being diffeomorphic. We note that accurately computing $\det \nabla \boldsymbol{y}$ is challenging. The values reported in this study have to be considered with the numerical accuracy in mind. For `Demons` we report the values generated by the software. We refine this parameter search by increasing the number of iterations per level by a factor of 2, 5, 10, and 100 to make sure that we have ''converged'' to an ''optimal'' solution. We apply the best variants identified by this exhaustive search to the entire `NIREP` data.

- *`CLAIRE`*: We consider an H^1-div regularization model (H^1-seminorm for $\boldsymbol{v}$, i.e., $\mathcal{A} = -\Delta$, with an additional penalty for $\nabla \cdot \boldsymbol{v}$). We set the regularization weight for the penalty for the divergence of $\boldsymbol{v}$ to $\beta_w = 1\text{e}{-4}$. To select an adequate regularization parameter β_v, we use a binary search. We set the bounds for the determinant of the deformation gradient to 0.25 and 0.30, respectively. We set the number of time steps of the SL scheme to $n_t = 4$. The number of maximal iterations is set to 50 (not reached). The number of Krylov iterations is limited to 100 (not reached). We use a tolerance of $5\text{e}{-2}$ and $1\text{e}{-6}$ for the relative and absolute reduction of the reduced gradient as a stopping criterion. We use $n_t = 4$ time steps for numerical integration. We run the registration on full resolution and (based on the experiments in §5) use a parameter continuation scheme in β_v to solve the registration problem. Probing for an optimal regularization parameter is expensive. We limit this estimation to the datasets `na01` (reference image) and `na02` (template image), assuming that we can estimate an adequate parameter for a particular application based on a subset of images. We execute `CLAIRE` on the remaining images using the identified parameters. We compute $\det \nabla \boldsymbol{y}$ directly from $\boldsymbol{v}$ by solving a transport equation (see [96, 100] for details). We transport the label maps to generate results that are consistent with the values reported for the determinant of the deformation map. This requires an additional smoothing (standard deviation: one voxel) and thresholding (threshold: 0.5) step.

Results. We illustrate the search for an optimal regularization weight for `CLAIRE` in Fig. 9. We showcase an exemplary result for the rate of convergence of SDDEM and `CLAIRE` in Fig. 10 (the software is executed at full image resolution). We summarize exemplary registration results for all datasets in Fig. 11. Here, D1, D2, D3, C1, and C2 correspond to different variants of the `Demons` algorithm and `CLAIRE`. C1 corresponds to `CLAIRE` with regularization weight of $9.72\text{e}{-3}$ ($\epsilon_J = 0.3$) and C2 to `CLAIRE` with a regularization weight of $5.50\text{e}{-4}$ ($\epsilon_J = 0.25$). The first `Demons` variant D1 is SDDEM with $(\sigma_u, \sigma_d) = (0, 3.5)$ (smooth setting). It yields results that are competitive with `CLAIRE` in terms of the determinant of the deformation gradient. The second variant D2 is SDDEM with $(\sigma_u, \sigma_d) = (0, 3.0)$, which gave us the best result (highest attainable Dice score with the determinant of the deformation gradient not changing sign for the training data `na01` and `na02`). The third variant D3 is SDDEM with $(\sigma_u, \sigma_d) = (0, 1.0)$ (aggressive setting). We achieve results that are competitive with `CLAIRE` in terms of the Dice score. We execute the `Demons` algorithm with a three-level grid continuation scheme with 150, 100, and 50 iterations per level, respectively.

We refer the interested reader to the supplementary material for more detailed results for these runs and an additional insight into the parameter search we have conducted to identify the best variant of the `Demons` algorithm. Detailed results for the `CLAIRE` variant C1 are reported in Tab. SM7. Detailed results for the `CLAIRE` variant C2 are reported in Tab. SM8. For `CLAIRE`, we report Dice coefficients for the individual 32 gray matter labels in Fig. SM2. Results for probing for adequate regularization parameters σ_u, σ_d, and σ_v for different variants of the `Demons` algorithm are reported in Tab. SM9 and Tab. SM10 (exhaustive search). Building up on these results we extend this search by additionally increasing the iteration count. These results are reported in Tab. SM11. We determined that SDDEM gives us the best results in terms of the Dice coefficient. Detailed results for the variants D1, D2, and D3 can be found in Tab. SM12.

Observations. The most important observations are:

- `CLAIRE` yields a smaller mismatch/higher Dice coefficient with a better control of the determinant of the deformation gradient (see Fig. 11). We obtain an average Dice coefficient of $8.38\text{e}{-1}$ with $(\min, \max) = (4.14\text{e}{-1}, 1.11\text{e}{+1})$ as extremal values for the determinant of the deformation gradient

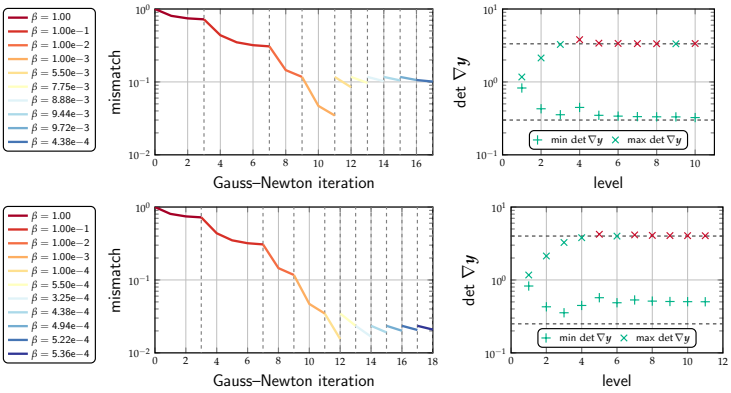

FIG. 9. *Estimation of the regularization parameter β_v. We use an H^1-div regularization model with $\beta_w = 1e{-}4$. We show the trend of the mismatch with respect to the Gauss–Newton iterations (left column) and the trend of the extremal values of the determinant of the deformation gradient with respect to the continuation level (right column). The top block shows results for a bound of 0.3 for* $\min \det \nabla \boldsymbol{y}$. *The bound for the bottom row is 0.25. These bounds are illustrated as dashed gray lines in the plots on the right. Here, we show (per continuation level) the trend of* $\max \det \nabla \boldsymbol{y}$ *(marker: ×) and* $\min \det \nabla \boldsymbol{y}$ *(marker: +). If the bounds are violated, we display the marker in red. We separate the continuation levels with a vertical gray line in the plots for the mismatch; the color of the line corresponds to a particular regularization weight (see legend).*

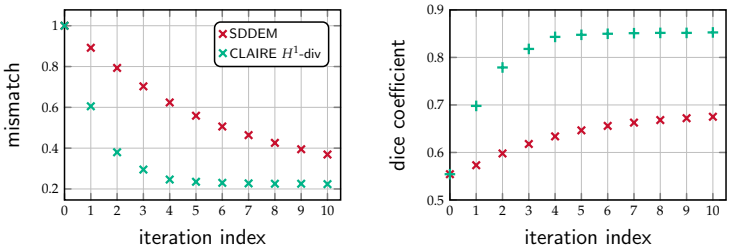

FIG. 10. *Convergence results for* CLAIRE *and SDDEM. We report the trend of the mismatch (left) and the Dice coefficient (right) versus the outer iterations. For* CLAIRE*, we solve this problem more accurately than in the other runs on the real data to show the asymptotic behaviour of our solver. We do not perform any grid, scale, or parameter continuation for both methods. We consider the datasets* na01 *(reference image) and* na02 *(template image).*

(on average). The Dice score for the best variant of the Demons algorithm, SDDEM, is 8.42e−1. To attain this score we have to commit to non-diffeomorphic deformation maps (as judged by the values for the determinant of the deformation gradient reported by the Demons software). An extension of CLAIRE, which we did not consider in this work, is to enable a monitor for the determinant of the deformation gradient that increases the regularization parameter if we hit the bound we used to estimate β_v. This would prevent the outliers we observe in this study, without having to probe for a new regularization parameter for each individual datasets.

- For CLAIRE, the average runtime (across all registrations) is 1.08e+2 s and 2.43e+2 s for $\beta_v = 9.72e{-}3$ and $\beta_v = 5.50e{-}4$, respectively. This is between 1.5× and 5× slower than the Demons algorithm if we execute Demons using 15, 10, and 5 iterations per level. Notice, that Demons is executed for a fixed number of iterations. The runs reported here use 10× more iterations per level (which slightly improves the performance of Demons; we refer the interested reader to Tab. SM11 in the supplementary material for details). This increases the runtime of the Demons algorithm by roughly a factor of 10. CLAIRE uses a relative tolerance for the gradient as a stopping criterion. Moreover, Demons uses a grid continuation scheme. We execute these runs on the fine resolution, and perform a parameter continuation instead (since we observed it is more stable for vanishing β_v; see §5).
- On the fine grid (single-level registration), CLAIRE converges significantly faster than the Demons algorithm. We reach a Dice score of more than 0.8 for CLAIRE after only three Gauss–Newton iterations (see Fig. 10).

Conclusions. With CLAIRE we achieve (*i*) a computational performance that is close to that of the Demons algorithm (1.5× to 5× slower for the fastest setting we used for Demons) with (*ii*) a registration quality that is superior (higher Dice coefficient with a better behaved determinant of the deformation gradient).

5.2. Scalability. We study strong scaling of our improved implementation of CLAIRE for up to 3 221 225 472 unknowns for a synthetic test problem consisting of smooth trigonometric functions (see §4.2).

Setup. We consider grid sizes 128^3, 256^3, 512^3, and 1024^3. We use an H^1-div regularization model with $\beta_w = 1e{-}3$ and $\beta_w = 1e{-}4$. We use the two-level preconditioner with a nested PCG method with a tolerance of $1e{-}1\epsilon_H$ to compute the action of the inverse of the preconditioner. We set the tolerance for the stopping condition for the relative reduction of the reduced gradient to 1e−2 (with an absolute

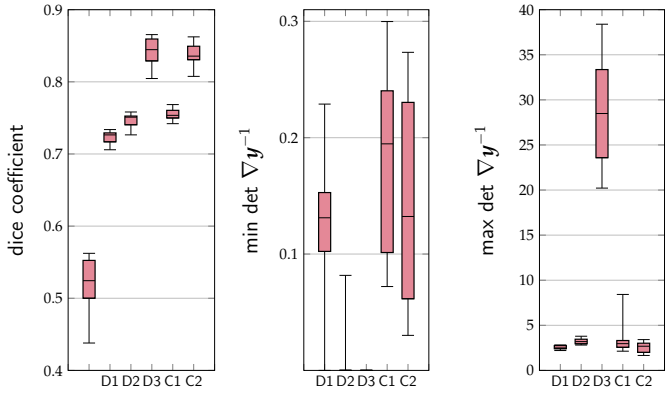

FIG. 11. *Registration results for the* NIREP *data. We consider three variants of the diffeomorphic* Demons *algorithm: D1 corresponds to SDDEM with* $(\sigma_u, \sigma_d) = (0, 3.5)$*, D2 to SDDEM with* $(\sigma_u, \sigma_d) = (0, 3.0)$*, and D3 to SDDEM with* $(\sigma_u, \sigma_d) = (0, 1.0)$*. These choices are based on an exhaustive search (we refer the interested reader to the supplementary material for details). For* CLAIRE *we use two different choices of the regularization parameter for the* H^1*-div regularization model (C1 corresponds to* CLAIRE *with* $\beta_v = 9.72\text{e}{-3}$ *and C2 to* CLAIRE *with* $\beta_v = 5.50\text{e}{-4}$*; these parameters are determined via a binary search (see Fig. 9)). We report results for the entire* NIREP *dataset. The plot on the left shows the Dice coefficient (on the very left, we also provide a box plot for the Dice coefficient before registration). This coefficient is computed for the union of all gray matter labels (to simplify the analysis). The middle and right box plot show the extremal values for the determinant of the deformation gradient.*

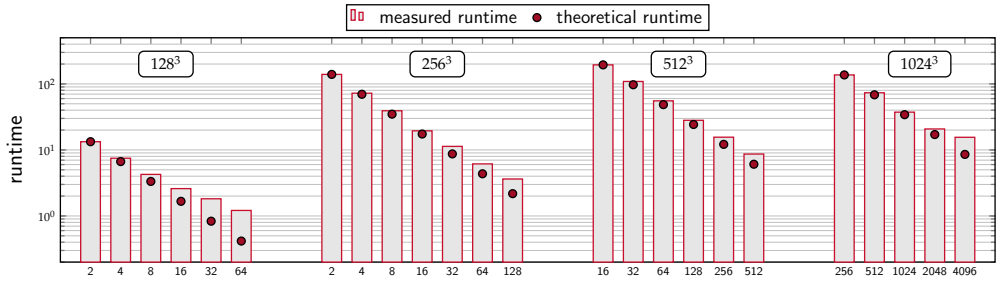

FIG. 12. *Strong scaling results for a synthetic test problem on TACC's Lonestar 5 system (see §4.1 for specs). We use 12 MPI tasks per node. We report the runtime (time-to-solution) for the entire inversion (in seconds). Our Newton–Krylov solver converges in three iterations (with three Hessian matvecs and a total of 15 PDE solves on the fine level). We consider grid sizes* 128^3, 256^3, 512^3, *and* 1024^3 *(from left to right). The largest run uses 4096 MPI tasks on 342 compute nodes (we solve for* 3 221 225 472 *unknowns).*

tolerance of 1e−6 (not reached)). We execute the runs on TACC's Lonestar 5 system (see §4.1 for specs).

Results. We report strong scaling results for CLAIRE in Fig. 12. We report the time-to-solution and compare it to the runtime we expect theoretically. We report detailed results, which form the basis of the run times reported in Fig. 12, in Tab. 4. Here, we report the execution time of the FFT and the interpolation kernels on the coarse (two-level preconditioner) and fine grid, the runtime of our solver (time-to-solution), and the strong scaling efficiency of our improved implementation of CLAIRE. We refer to [52, 99] more detailed results on the scalability of our original implementation of CLAIRE

Observations. The most important observations are:

- We obtain a good strong scaling efficiency that is at the order of 60%.
- The strong scaling results are in accordance with the performance reported in [52, 97]. The key difference is that the scalability of our new solver is dominated by the coarse grid discretization within the preconditioner. That is, we do not observe the scalability reported in [52, 97] if we execute CLAIRE with the same amount of resources for a given resolution of the data. However, if we compare the scalability results reported in [52] with a resolution that matches the coarse grid in the preconditioner, we can observe a similar strong scaling efficiency.
- We can solve clinically relevant problems in about 2 s if we execute CLAIRE with 256 MPI tasks (see run #14 in Tab. 4).
- We can solve problems with up to 3 221 225 472 unknowns in less then 5 s with 4096 MPI tasks on 342 compute nodes on TACC's Lonestar 5 system (see run #25 in Tab. 4). The solver converges in 1.37e+2 s if we execute the run on 22 nodes with 256 MPI tasks.

Conclusions. With CLAIRE we deploy a solver that scales on HPC platforms. CLAIRE approaches run-times that represent a significant step towards providing "*real-time*" capabilities for clinically relevant problem sizes (inversion for ∼50 million unknowns in 2.34 s using 256 MPI tasks; see also [52, 99]). Further accelerations for the basis of our current work. CLAIRE can also be used to solve diffeomorphic image

TABLE 4

Scalability results for CLAIRE for a synthetic test problem. We report strong scaling results for up to 3 221 225 472 unknowns (grid sizes: 128^3, 256^3, 152^3, and 1024^3). We execute these runs on TACC's Lonestar 5 system (see §4.1 of the main manuscript for the specs). We consider an H^1-div regularization model with $\beta_v = 1e{-}3$ and $\beta_w = 1e{-}4$. We use a two-level preconditioner with a nested PCG method. We terminate the inversion if the gradient is reduced by 1e−2. We execute these runs in single precision. We use 12 MPI tasks per node. We report the execution time for the FFT and the interpolation (on the coarse and the fine grid; in seconds), the runtime of the solver (time-to-solution; in seconds), and the strong scaling efficiency.

grid	run	nodes	tasks	fine grid				coarse grid				runtime	efficiency
				FFT		interpolation		FFT		interpolation			
128^3	#1	1	2	4.25	(32%)	2.82	(21%)	2.21	(17%)	1.73	(13%)	1.33e+1	
	#2	1	4	2.45	(32%)	1.46	(19%)	1.29	(17%)	9.22e−1	(12%)	7.49	89%
	#3	1	8	1.35	(32%)	8.14e−1	(19%)	7.32e−1	(17%)	5.13e−1	(12%)	4.26	78%
	#4	2	16	7.39e−1	(28%)	5.69e−1	(22%)	4.37e−1	(17%)	3.11e−1	(12%)	2.59	64%
	#5	3	32	4.16e−1	(23%)	3.91e−1	(21%)	3.78e−1	(21%)	2.55e−1	(14%)	1.82	46%
	#6	6	64	3.12e−1	(26%)	3.45e−1	(28%)	1.52e−1	(13%)	1.22e−1	(10%)	1.21	34%
256^3	#7	1	2	5.55e+1	(40%)	2.77e+1	(20%)	2.08e+1	(15%)	1.47e+1	(11%)	1.39e+2	
	#8	1	4	2.70e+1	(37%)	1.41e+1	(19%)	1.18e+1	(16%)	7.59	(10%)	7.23e+1	96%
	#9	1	8	1.45e+1	(37%)	7.70	(20%)	6.30	(16%)	4.14	(11%)	3.92e+1	89%
	#10	2	16	6.87	(35%)	3.50	(18%)	3.41	(18%)	2.13	(11%)	1.95e+1	89%
	#11	3	32	4.06	(36%)	1.94	(17%)	2.01	(18%)	1.15	(10%)	1.13e+1	77%
	#12	6	64	2.15	(35%)	1.04	(17%)	1.05	(17%)	6.38e−1	(10%)	6.14	71%
	#13	11	128	1.20	(33%)	6.26e−1	(17%)	5.92e−1	(16%)	3.90e−1	(11%)	3.63	60%
	#14	22	256	7.08e−1	(30%)	4.38e−1	(18%)	3.34e−1	(14%)	2.58e−1	(11%)	2.34	47%
512^3	#15	2	16	8.01e+1	(41%)	3.26e+1	(17%)	3.39e+1	(17%)	1.85e+1	(10%)	1.94e+2	
	#16	3	32	4.52e+1	(41%)	1.79e+1	(16%)	1.94e+1	(18%)	9.86	(9%)	1.09e+2	89%
	#17	6	64	2.21e+1	(40%)	8.87	(16%)	1.03e+1	(19%)	5.08	(9%)	5.54e+1	88%
	#18	11	128	1.07e+1	(38%)	4.30	(15%)	5.59	(20%)	2.68	(10%)	2.81e+1	86%
	#19	22	256	5.70	(37%)	2.26	(15%)	3.16	(20%)	1.58	(10%)	1.56e+1	78%
	#20	43	512	3.00	(35%)	1.45	(17%)	1.40	(16%)	9.39e−1	(11%)	8.66	70%
1024^3	#21	22	256	5.69e+1	(42%)	2.16e+1	(16%)	2.68e+1	(20%)	1.14e+1	(8%)	1.37e+2	
	#22	43	512	2.85e+1	(39%)	1.06e+1	(14%)	1.70e+1	(23%)	6.42	(9%)	7.34e+1	93%
	#23	86	1024	1.45e+1	(39%)	5.23	(14%)	7.75	(21%)	3.29	(9%)	3.74e+1	92%
	#24	171	2048	7.22	(35%)	3.26	(16%)	4.13	(20%)	2.15	(10%)	2.08e+1	82%
	#25	342	4096	4.49	(28%)	2.30	(15%)	3.20	(21%)	1.76	(11%)	1.55e+1	55%

registration problems of unprecedented scale, something that is of interest for whole body imaging [89, 134] or experimental, high-resolution microscopic imaging [34, 88, 136]. The largest problem we have solved with our original implementation of CLAIRE is 25 769 803 776 unknowns (see [52]). To the best of our knowledge, CLAIRE is the only software for large deformation diffeomorphic registration with these capabilities.

6. Conclusions. With this publication we release CLAIRE, a memory-distributed algorithm for stationary velocity field large deformation diffeomorphic image registration in 3D. This work builds up on our former contributions on constrained large deformation diffeomorphic image registration [52, 96–100]. We have performed a detailed benchmark study of the performance of CLAIRE on synthetic and real data. We have studied the convergence for different schemes for preconditioning the reduced space Hessian in §4.3. We have examined the rate of convergence of our Gauss–Newton–Krylov solver in §4.4. We have reported results for different schemes available in CLAIRE in §5 to study the time-to-solution. We have compared the registration quality obtained with CLAIRE to different variants of the diffeomorphic Demons algorithm in §5.1. We have also reported strong scaling results for our improved memory-distributed solver on supercomputing platforms (see §5.2). We note that we accompany this work with supplementary material that provides a more detailed picture about the performance of our method. The most important conclusions are:

- CLAIRE delivers high-fidelity results with well-behaved deformations. Our results are in accordance with observations we have made for the two-dimensional case [97]. Our H^1-div formulation outperforms the diffeomorphic Demons algorithm in terms of data fidelity and deformation regularity (as judged by the higher dice score and more well-behaved extremal values for the determinant of the deformation gradient; see Fig. 11 in §5.1).
- Our Gauss–Newton–Krylov solver converges after only a few iterations to high-fidelity results. The rate of convergence of CLAIRE is significantly better than that of the Demons algorithm (if we run the code on a single resolution level; see Fig. 10 in §5.1).

- CLAIRE introduces different acceleration schemes. These schemes not only stabilize the computations but also lead to a reduction in runtime (see Tab. 3 in §5). CLAIRE delivers a speedup of 5× for the parameter continuation. We observed a speedup of up to 17× when considering a grid continuation scheme (results not reported here). We disregarded this scheme, because we observed a significant dependence of the performance on the regularity of the velocity. Combining the grid and parameter continuation scheme may yield an even better performance. Designing an effective schedule for a combined scheme remains subject to future work.
- Our two-level preconditioner is effective. We achieve the best performance if we compute the action of its inverse with a nested PCG method. This allows us to avoid a repeated estimation of spectral bounds of the reduced space Hessian operator, which is necessary if we consider a semi-iterative Chebyshev method. For real data, we achieve a moderate speedup of about 4× for the entire inversion compared to our prior work [97]. Moreover, we saw that the performance of our schemes for preconditioning the reduced space Hessian is *not* independent of the regularization parameter for the velocity. Designing a preconditioner that yields a good performance for vanishing regularity of the objects requires more work.
- CLAIRE delivers good scalability results. In this work, we showcase results for up to 3 221 225 472 unknowns on 342 compute nodes of TACC's Lonestar 5 system executed with 4096 MPI tasks. This demonstrates that we can tackle applications that require the registration of high-resolution imaging data such as, e.g., CLARITY imaging (a new optical imaging technique that delivers sub-micron resolution [34, 88, 136]). Further, we demonstrated that CLAIRE can deliver run-times that represent a significant step towards providing ''real-time'' capabilities for clinically relevant problem sizes (inversion for ∼50 million unknowns in about 2 s using 256 MPI tasks). To the best of our knowledge, CLAIRE is the only software with these capabilities. Further run time accelerations form the basis of our current work.

With this work we have identified several aspects of CLAIRE that need to be improved. The time-to-solution on a single workstation is not yet fully competitive with the diffeomorphic Demons algorithm. We are currently working on improvements to our computational kernels to further reduce the execution time of CLAIRE. In addition to algorithmic improvements, we are also actively working on a GPU implementation of CLAIRE. As we have mentioned in the limitations, CLAIRE does not support non-stationary velocities. The design of efficient numerical schemes for non-stationary (time dependent) velocities is something we will address in our future work. Moreover, we are currently adding support for new distance measures to enable multi-modal registration.

REFERENCES

[1] S. S. Adavani and G. Biros, *Fast algorithms for source identification problems with elliptic PDE constraints*, SIAM Journal on Imaging Sciences, 3 (2008), pp. 791–808. 3

[2] ———, *Multigrid algorithms for inverse problems with linear parabolic PDE constraints*, SIAM Journal on Scientific Computing, 31 (2008), pp. 369–397. 3, 9

[3] V. Akcelik, G. Biros, and O. Ghattas, *Parallel multiscale Gauss-Newton-Krylov methods for inverse wave propagation*, in Proc ACM/IEEE Conference on Supercomputing, 2002, pp. 1–15. 4

[4] V. Akcelik, G. Biros, O. Ghattas, J. Hill, D. Keyes, and B. van Bloemen Waanders, *Parallel algorithms for PDE constrained optimization*, vol. 20 of Parallel Processing for Scientific Computing, SIAM, Philadelphia, Pennsylvania, US, 2006, ch. 16, pp. 291–322. 4

[5] 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. 8

[6] R. Andreev, O. Scherzer, and W. Zulehner, *Simultaneous optical flow and source estimation: Space–time discretization and preconditioning*, Applied Numerical Mathematics, 96 (2015), pp. 72–81. 3

[7] V. Arsigny, O. Commowick, X. Pennec, and N. Ayache, *A Log-Euclidean framework for statistics on diffeomorphisms*, in Proc Medical Image Computing and Computer-Assisted Intervention, vol. LNCS 4190, 2006, pp. 924–931. 3

[8] J. Ashburner, *A fast diffeomorphic image registration algorithm*, NeuroImage, 38 (2007), pp. 95–113. 1, 3, 4

[9] J. Ashburner and K. J. Friston, *Diffeomorphic registration using geodesic shooting and Gauss-Newton optimisation*, NeuroImage, 55 (2011), pp. 954–967. 1, 3

[10] B. B. Avants, C. L. Epstein, M. Brossman, and J. C. Gee, *Symmetric diffeomorphic image registration with cross-correlation: Evaluating automated labeling of elderly and neurodegenerative brain*, Medical Image Analysis, 12 (2008), pp. 26–41. 3

[11] B. B. Avants, N. J. Tustison, G. Song, P. A. Cook, A. Klein, and J. C. Gee, *A reproducible evaluation of ANTs similarity metric performance in brain image registration*, NeuroImage, 54 (2011), pp. 2033–2044. 1, 3, 4

[12] O. Axelsson and P. S. Vassilevski, *A black box generalized conjugate gradient solver with inner iterations and variable step precon-*

ditioning, SIAM Journal on Matrix Analysis and its Applications, 12 (1991), pp. 625–644. 9

[13] R. Azencott, R. Glowinski, J. He, A. Jajoo, Y. Li, A. Martynenko, R. H. W. Hoppe, S. Benzekry, and S. H. Little, *Diffeomorphic matching and dynamic deformable surfaces in 3D medical imaging*, Computational Methods in Applied Mathematics, 10 (2019), pp. 235–274. 1

[14] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, D. A. May, L. C. McInnes, R. T. Mills, T. Munson, K. Rupp, Patrick, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang, *PETSc Webpage*. https://www.mcs.anl.gov/petsc. 2, 10

[15] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, and H. Zhang, *PETSc users manual*, Tech. Rep. ANL-95/11 - Revision 3.7, Argonne National Laboratory, 2016. 2, 4, 10

[16] V. Barbu and G. Marinoschi, *An optimal control approach to the optical flow problem*, Systems & Control Letters, 87 (2016), pp. 1–9. 3, 5

[17] 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. 1, 2, 3, 4, 5, 6, 11

[18] M. Benzi, G. H. Golub, and J. Liesen, *Numerical solution of saddle point problems*, Acta Numerica, 14 (2005), pp. 1–137. 4

[19] 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. 149–173. 3, 4

[20] L. T. Biegler, O. Ghattas, M. Heinkenschloss, D. Keyes, and B. van Bloemen Waanders, *Real-time PDE-constrained optimization*, SIAM, 2007. 4

[21] L. T. Biegler, O. Ghattas, M. Heinkenschloss, and B. van Bloemen Waanders, *Large-scale PDE-constrained optimization*, Springer, 2003. 3, 4

[22] G. Biros and G. Doğan, *A multilevel algorithm for inverse problems with elliptic PDE constraints*, Inverse Problems, 24 (2008). 3, 9

[23] G. Biros and O. Ghattas, *Parallel Newton-Krylov methods for PDE-constrained optimization*, in Proc ACM/IEEE Conference on Supercomputing, 1999, pp. 28–40. 4, 6

[24] ———, *Parallel Lagrange-Newton-Krylov-Schur methods for PDE-constrained optimization—Part I: The Krylov-Schur solver*, SIAM Journal on Scientific Computing, 27 (2005), pp. 687–713. 4, 6

[25] ———, *Parallel Lagrange-Newton-Krylov-Schur methods for PDE-constrained optimization—Part II: The Lagrange-Newton solver and its application to optimal control of steady viscous flows*, SIAM Journal on Scientific Computing, 27 (2005), pp. 714–739. 4, 6

[26] A. Borzì, K. Ito, and K. Kunisch, *Optimal control formulation for determining optical flow*, SIAM Journal on Scientific Computing, 24 (2002), pp. 818–847. 3, 4, 5

[27] A. Borzì and V. Schulz, *Computational optimization of systems governed by partial differential equations*, SIAM, Philadelphia, Pennsylvania, US, 2012. 3

[28] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004. 6

[29] 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. 8

[30] M. Burger, J. Modersitzki, and L. Ruthotto, *A hyperelastic regularization energy for image registration*, SIAM Journal on Scientific Computing, 35 (2013), pp. B132–B148. 3

[31] K. Chen and D. A. Lorenz, *Image sequence interpolation using optimal control*, Journal of Mathematical Imaging and Vision, 41 (2011), pp. 222–238. 1, 2, 3, 4, 5

[32] G. E. Christensen, X. Geng, J. G. Kuhl, J. Bruss, T. J. Grabowski, I. A. Pirwani, M. W. Vannier, J. S. Allen, and H. Damasio, *Introduction to the non-rigid image registration evaluation project*, in Proc Biomedical Image Registration, vol. LNCS 4057, 2006, pp. 128–135. 12, 13, 18

[33] G. E. Christensen, R. D. Rabbitt, and M. I. Miller, *Deformable templates using large deformation kinematics*, Image Processing, IEEE Transactions on, 5 (1996), pp. 1435–1447. 3

[34] K. Chung and K. Deisseroth, *CLARITY for mapping the nverous system*, Nature Methods, 10 (2013), pp. 508–513. 21, 23

[35] Core facility for Advanced Computing and Data Science, *CACDS Webpage*. https://www.uh.edu/cacds. 4, 10, 12

[36] W. R. Crum, C. Tanner, and D. J. Hawkes, *Anisotropic multi-scale fluid registration: Evaluation in magnetic resonance breast imaging*, Physics in Medicine and Biology, 50 (2005), pp. 5153–5174. 1

[37] K. Czechowski, C. Battaglino, C. McClanahan, K. Iyer, P.-K. Yeung, and R. Vuduc, *On the communication complexity of 3D FFTs and its implications for exascale*, in Proc ACM/IEEE Conference on Supercomputing, 2012, pp. 205–214. 10

[38] Data Format Working Group of the Neuroimaging Informatics Technology Initiative, *niftilib*. http://niftilib.sourceforge.net, 2019. 10

[39] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, *Inexact Newton methods*, SIAM Journal on Numerical Analysis, 19 (1982), pp. 400–408. 8

[40] P. Dupuis, U. Gernander, and M. I. Miller, *Variational problems on flows of diffeomorphisms for image matching*, Quarterly of Applied Mathematics, 56 (1998), pp. 587–600. 1, 2, 3, 11

[41] 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. 8

[42] A. Eklund, P. Dufort, D. Forsberg, and S. M. LaConte, *Medical image processing on the GPU–past, present and future*, Medical Image Analysis, 17 (2013), pp. 1073–1094. 4

[43] H. Engl, M. Hanke, and A. Neubauer, *Regularization of inverse problems*, Kluwer Academic Publishers, Dordrecht, NL, 1996. 1

[44] M. Falcone and R. Ferretti, *Convergence analysis for a class of high-order semi-Lagrangian advection schemes*, SIAM Journal on Numerical Analysis, 35 (1998), pp. 909–940. 2, 7

[45] B. Fischer and J. Modersitzki, *Ill-posed medicine – an introduction to image registration*, Inverse Problems, 24 (2008), pp. 1–16. 1, 5

[46] O. Fluck, C. Vetter, W. Wein, A. Kamen, B. Preim, and R. Westermann, *A survey of medical image registration on graphics hardware*, Computer Methods and Programs in Biomedicine, 104 (2011), pp. e45–e57. 4

[47] M. Frigo and S. G. Johnson, *FFTW Webpage*. http://www.fftw.org. 10

[48] M. Frigo and S. G. Johnson, *The design and implementation of FFTW3*, in Proc of the IEEE, vol. 93, 2005, pp. 216–231. 10

[49] A. Gholami and G. Biros, *AccFFT GitHub repository*. https://github.com/amirgholami/accfft. 10

[50] A. Gholami, J. Hill, D. Malhotra, and G. Biros, *AccFFT: A library for distributed-memory FFT on CPU and GPU architectures*, arXiv e-prints, (2016). https://arxiv.org/abs/1506.07933. 7, 10

[51] A. Gholami, A. Mang, and G. Biros, *An inverse problem formulation for parameter estimation of a reaction-diffusion model of low grade gliomas*, Journal of Mathematical Biology, 72 (2016), pp. 409–433. https://doi.org/10.1007/s00285-015-0888-x. 3

[52] A. Gholami, A. Mang, K. Scheufele, C. Davatzikos, M. Mehl, and G. Biros, *A framework for scalable biophysics-based image analysis*, in Proc ACM/IEEE Conference on Supercomputing, 2017, pp. 19:1–19:13. http://doi.acm.org/10.1145/3126908.3126930. 2, 3, 4, 5, 7, 10, 11, 13, 15, 21, 22

[53] A. Gholami, S. Subramanian, V. Shenoy, N. Himthani, X. Yue, S. Zhao, P. Jin, G. Biros, and K. Keutzer, *A novel domain adaptation framework for medical image segmentation*, in Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries, vol. LNCS 11384, 2019, pp. 289–298. 3

[54] P. E. Gill, W. Murray, and M. H. Wright, *Practical optimization*, Academic Press, Waltham, Massachusetts, US, 1981. 8

[55] L. Giraud, D. Ruiz, and A. Touhami, *A comparitive study of iterative solvers exploiting spectral information for SPD systems*, SIAM Journal on Scientific Computing, 27 (2006), pp. 1760–1786. 9

[56] G. H. Golub and R. S. Varga, *Chebyshev semi-iterative methods, successive overrelaxation iterative methods, and second order Richardson iterative methods*, Numerische Mathematik, 3 (1961), pp. 147–156. 2, 9

[57] A. Grama, A. Gupta, G. Karypis, and V. Kumar, *An Introduction to parallel computing: Design and analysis of algorithms*, Addison Wesley, second ed., 2003. 10, 11

[58] W. Griggs, V. E. Henson, and S. F. McCormick, *A multigrid tutorial*, SIAM, 2000. 9

[59] M. D. Gunzburger, *Perspectives in flow control and optimization*, SIAM, Philadelphia, Pennsylvania, US, 2003. 3

[60] M. E. Gurtin, *An introduction to continuum mechanics*, vol. 158 of Mathematics in Science and Engineering, Academic Press, 1981. 5

[61] M. Gutknecht and S. Röllin, *The Chebisyev iteration revisited*, Parallel Computing, 28 (2002), pp. 263–283. 2, 9

[62] L. Ha, J. Krüger, S. Joshi, and C. T. Silva, *Multiscale unbiased diffeomorphic atlas construction on multi-GPUs*, in CPU Computing Gems Emerald Edition, Elsevier Inc, 2011, ch. 48, pp. 771–791. 4

[63] L. K. Ha, J. Krüger, P. T. Fletcher, S. Joshi, and C. T. Silva, *Fast parallel unbiased diffeomorphic atlas construction on multi-graphics processing units*, in Proc Eurographics Conference on Parallel Grphics and Visualization, 2009, pp. 41–48. 4

[64] E. Haber and R. Horesh, *A multilevel method for the solution of time dependent optimal transport*, Numerical Mathematics: Theory, Methods and Applications, 8 (2015), pp. 97–111. 3

[65] E. Haber and J. Modersitzki, *Image registration with guaranteed displacement regularity*, International Journal of Computer Vision, 71 (2007), pp. 361–372. 3

[66] E. Haber and D. Oldenburg, *A GCV based method for nonlinear ill-posed problems*, Computational Geosciences, 4 (2000), pp. 41–63. 11, 12

[67] J. V. Hajnal, D. L. G. Hill, and D. J. Hawkes, eds., *Medical Image Registration*, CRC Press, Boca Raton, Florida, US, 2001. 1, 3

[68] P. C. Hansen, *Rank-deficient and discrete ill-posed problems*, SIAM Monographs on Mathematical Modeling and Computation, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, Philadelphia, Pennsylvania, US, 1998. 11

[69] G. L. Hart, C. Zach, and M. Niethammer, *An optimal control approach for deformable registration*, in Proc IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 9–16. 2, 3, 4, 6

[70] M. Hernandez, *Gauss-Newton inspired preconditioned optimization in large deformation diffeomorphic metric mapping*, Physics in Medicine and Biology, 59 (2014), pp. 6085–6115. 3

[71] M. Hernandez, M. N. Bossa, and S. Olmos, *Registration of anatomical images using paths of diffeomorphisms parameterized with stationary vector field flows*, International Journal of Computer Vision, 85 (2009), pp. 291–306. 1, 3

[72] R. Herzog, J. W. Pearson, and M. Stoll, *Fast iterative solvers for an optimal transport problem*, arXiv e-prints, (2018). . 3, 4

[73] 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. 2, 8

[74] J. Hinkle, P. T. Fletcher, B. Wang, B. Salter, and S. Joshi, *4D MAP image reconstruction incorporating organ motion*, in Proc Information Processing in Medical Imaging, no. LNCS 5636, 2009, pp. 676–687. 3

[75] M. Hinze, R. Pinnau, M. Ulbrich, and S. Ulbrich, *Optimization with PDE constraints*, Springer, Berlin, DE, 2009. 3

[76] Höchstleistungsrechenzentrum Stuttgart, *HLRS Webpage*. https://www.hlrs.de. 10

[77] B. K. P. Horn and B. G. Shunck, *Determining optical flow*, Artificial Intelligence, 17 (1981), pp. 185–203. 3

[78] H. Johnson and J. Matsui, *BRAINSia/LogSymmetricDemons GitHub repository*. https://github.com/BRAINSia/LogSymmetricDemons. 18

[79] H. J. Johnson, M. M. McCormick, and L. Ibáñez, *The ITK software guide: Design and functionality*, Kitware Inc., 2015. 4, 18

[80] S. Joshi, B. Davis, M. Jornier, and G. Gerig, *Unbiased diffeomorphic atlas construction for computational anatomy*, NeuroImage, 23 (2005), pp. S151–S160. 4

[81] E. M. Kalmoun, L. Garrido, and V. Caselles, *Line search multilevel optimization as computational methods for dense optical flow*, SIAM Journal on Imaging Sciences, 4 (2011), pp. 695–722. 3

[82] B. Kaltenbacher, *On the regularizing properties of a full multigrid method for ill-posed problems*, Inverse Problems, 17 (2001), pp. 767–788. 9

[83] ———, *V-cycle convergence of some multigrid methods for ill-posed problems*, Mathematics of Computation, 72 (2003), pp. 1711–1730. 9

[84] J. T. King, *On the construction of preconditioners by subspace decomposition*, Journal of Computational and Applied Mathematics, 29 (1990), pp. 195–205. 9

[85] Kitware, *Insight Segmentation and Registration Toolkit (ITK) Webpage*. https://itk.org. 18

[86] S. Klein, M. Staring, K. Murphy, M. A. Viergever, and J. P. W. Pluim, *ELASTIX: A tollbox for intensity-based medical image registration*, Medical Imaging, IEEE Transactions on, 29 (2010), pp. 196–205. 3, 4

[87] L. Koenig, J. Ruehaak, A. Derksen, and J. Lellmann, *A matrix-free approach to parallel and memory-efficient deformable image registration*, SIAM Journal on Scientific Computing, 40 (2018), pp. B858–B888. 4

[88] K. S. Kutten, N. Charon, M. I. Miller, J. T. Ratnanather, K. Deisseroth, L. Ye, and J. T. Vogelstein, *A diffeomorphic approach to multimodal registration with mutual information: Applications to CLARITY mouse brain images*, in Proc Medical Image Computing and Computer-Assisted Intervention, vol. LNCS 10433, 2017, pp. 275–282. 4, 21, 23

[89] F. E. Lecouvet, *Whole-body MR imaging: Musculoskeletal applications*, Radiology, 279 (2016), pp. 345–365. 21

[90] E. Lee and M. Gunzburger, *An optimal control formulation of an image registration problem*, Journal of Mathematical Imaging and Vision, 36 (2010), pp. 69–80. 3, 5

[91] ———, *Anaysis of finite element discretization of an optimal control formulation of the image registration problem*, SIAM Journal on Numerical Analysis, 49 (2011), pp. 1321–1349. 3

[92] J. Li, W. Liao, A. Choudhary, R. Ross, R. Thakur, W. Gropp, R. Latham, A. Siegel, B. Gallagher, and M. Zingale, *Parallel netCDF: A scientific high-performance I/O interface*, in Proc ACM/IEEE Conference on Supercomputing, 2003, p. 39. 10

[93] J. L. Lions, *Optimal control of systems governed by partial differential equations*, Springer, 1971. 3, 5

[94] M. Lorenzi, N. Ayache, G. B. Frisoni, and X. Pennec, *LCC-demons: a robust and accurate symmetric diffeomorphic registration algorithm*, NeuroImage, 81 (2013), pp. 470–483. 3

[95] M. Lorenzi and X. Pennec, *Geodesics, parallel transport and one-parameter subgroups for diffeomorphic image registration*, International Journal of Computer Vision, 105 (2013), pp. 111–127. 3

[96] 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. https://doi.org/10.1137/140984002. 2, 3, 4, 5, 7, 8, 12, 19, 22

[97] ———, *Constrained H^1-regularization schemes for diffeomorphic image registration*, SIAM Journal on Imaging Sciences, 9 (2016), pp. 1154–1194. https://doi.org/10.1137/15M1010919. 2, 3, 4, 5, 8, 10, 21, 22, 23

[98] ———, *A semi-Lagrangian two-level preconditioned Newton–Krylov solver for constrained diffeomorphic image registration*, SIAM Journal on Scientific Computing, 39 (2017), pp. B1064–B1101. https://doi.org/10.1137/16M1070475. 2, 3, 5, 7, 9, 10, 11, 22

[99] 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. https://doi.org/10.1109/SC.2016.71. 2, 3, 4, 5, 7, 8, 10, 11, 13, 15, 17, 21, 22

[100] A. Mang, A. Gholami, C. Davatzikos, and G. Biros, *PDE-constrained optimization in medical image analysis*, Optimization and Engineering, 19 (2018), pp. 765–812. https://doi.org/10.1007/s11081-018-9390-9. 3, 5, 8, 13, 19, 22

[101] 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. https://doi.org/10.1137/17M1114132. 3, 4, 5, 8

[102] A. Mang, S. Tharakan, A. Gholami, N. Nimthani, S. Subramanian, J. Levitt, M. Azmat, K. Scheufele, M. Mehl, C. Davatzikos, B. Barth, and G. Biros, *SIBIA-GlS: Scalable biophysics-based image analysis for glioma segmentation*, in Proc BraTS 2017 Workshop (MICCAI), 2017, pp. 197–204. 3, 4

[103] A. Mang, A. Toma, T. A. Schuetz, S. Becker, T. Eckey, C. Mohr, D. Petersen, and T. M. Buzug, *Biophysical modeling of brain tumor progression: From unconditionally stable explicit time integration to an inverse problem with parabolic PDE constraints for model calibration*, Medical Physics, 39 (2012), pp. 4444–4459. https://doi.org/10.1118/1.4722749. 3

[104] T. Mansi, X. Pennec, M. Sermesant, H. Delingette, and N. Ayache, *iLogDemons: A demons-based registration algorithm for tracking incompressible elastic biological tissues*, International Journal of Computer Vision, 92 (2011), pp. 92–111. 3

[105] M. I. Miller, *Computational anatomy: Shape, growth and atrophy comparison via diffeomorphisms*, NeuroImage, 23 (2004), pp. S19–S33. 3

[106] M. I. Miller, A. Trouvé, and L. Younes, *Geodesic shooting for computational anatomy*, Journal of Mathematical Imaging and Vision, 24 (2006), pp. 209–228. 3

[107] M. I. Miller and L. Younes, *Group actions, homeomorphism, and matching: A general framework*, International Journal of Computer Vision, 41 (2001), pp. 61–81. 1, 3

[108] M. Modat, G. R. Ridgway, Z. A. Taylor, M. Lehmann, J. Barnes, D. J. Hawkes, N. C. Fox, and S. Ourselin, *Fast free-form deformation using graphics processing units*, Computer Methods and Programs in Biomedicine, 98 (2010), pp. 278–284. 3, 4

[109] J. Modersitzki, *Numerical methods for image registration*, Oxford University Press, New York, 2004. 1, 3, 5

[110] ———, *FAIR: Flexible algorithms for image registration*, SIAM, Philadelphia, Pennsylvania, US, 2009. 1, 3, 5, 8

[111] T. Munson, J. Sarich, S. Wild, S. Benson, and L. C. McInnes, *TAO 3.7 users manual*, Argonne National Laboratory, Mathematics and Computer Science Division, 2017. 2, 4, 10

[112] O. Museyko, M. Stiglmayr, K. Klamroth, and G. Leugering, *On the application of the Monge-Kantorovich problem to image registration*, SIAM Journal on Imaging Sciences, 2 (2009), pp. 1068–1097. 3

[113] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, New York, New York, US, 2006. 6, 8

[114] Northwestern University and Argonne National Laboratory, *PnetCDF: A parallel I/O library for NetCDF file access*. https://trac.mcs.anl.gov/projects/parallel-netcdf. 10

[115] Y. Notay, *Flexible conjugate gradients*, SIAM Journal on Scientific Computing, 22 (2000), pp. 1444–1460. 9

[116] T. Polzin, M. Niethammer, M. P. Heinrich, H. Handels, and J. Modersitzki, *Memory efficient LDDMM for lung CT*, in Proc Medical Image Computing and Computer-Assisted Intervention, vol. LNCS 9902, 2016, pp. 28–36. 1, 4

[117] E. E. Prudencio, R. Byrd, and X.-C. Cai, *Parallel full space SQP Lagrange–Newton–Krylov-Schwarz algorithms for PDE-constrained optimization problems*, SIAM Journal on Scientific Computing, 27 (2006), pp. 1305–1328. 6

[118] T. Rohlfing, C. R. Maurer, D. A. Bluemke, and M. A. Jacobs, *Volume-preserving nonrigid registration of MR breast images using free-form deformation with an incompressibility constraint*, Medical Imaging, IEEE Transactions on, 22 (2003), pp. 730–741. 3

[119] D. Rueckert, L. I. Sonoda, C. Hayes, D. L. G. Hill, M. O. Leach, and D. J. Hawkes, *Non-rigid registration using free-form deformations: Application to breast MR images*, Medical Imaging, IEEE Transactions on, 18 (1999), pp. 712–721. 3

[120] P. Ruhnau and C. Schnörr, *Optical Stokes flow estimation: An imaging-based control approach*, Experiments in Fluids, 42 (2007), pp. 61–78. 3
[121] K. A. Saddi, C. Chefd'hotel, and F. Cheriet, *Large deformation registration of contrast-enhanced images with volume-preserving constraint*, in Proc SPIE Medical Imaging, vol. 6512, 2008, pp. 651203–1–651203–10. 3
[122] K. Scheufele, A. Mang, A. Gholami, C. Davatzikos, G. Biros, and M. Mehl, *Coupling brain-tumor biophysical models and diffeomorphic image registration*, Computer Methods in Applied Mechanics and Engineering, 347 (2019), pp. 533–567. https://doi.org/10.1016/j.cma.2018.12.008. 3
[123] M. Sdika, *A fast nonrigid image registration with constraints on the Jacobian using large scale constrained optimization*, Medical Imaging, IEEE Transactions on, 27 (2008), pp. 271–281. 3
[124] J. Shackleford, N. Kandasamy, and G. Sharp, *On developing B-spline registration algorithms for multi-core processors*, Physics in Medicine and Biology, 55 (2010), pp. 6329–6351. 4
[125] ———, *High performance deformable image registration algorithms for manycore processors*, Morgan Kaufmann, Waltham, Massachusetts, US, 2013. 4
[126] D. P. Shamonin, E. E. Bron, B. P. F. Lelieveldt, M. Smits, S. Klein, and M. Staring, *Fast parallel image registration on CPU and GPU for diagnostic classification of Alzheimer's disease*, Frontiers in Neuroinformatics, 7 (2014), pp. 1–15. 4
[127] R. Shams, P. Sadeghi, R. A. Kennedy, and R. I. Hartley, *A survey of medical image registration on multicore and the GPU*, Signal Processing Magazine, IEEE, 27 (2010), pp. 50–60. 4
[128] O. Shenk, M. Manguoglu, A. Sameh, M. Christen, and M. Sathe, *Parallel scalable PDE-constrained optimization: Antenna identification in hyperthermia cancer treatment planning*, Computer Science—Research and Development, 23 (2009), pp. 177–183. 4
[129] 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. 3, 4
[130] S. Sommer, *Accelerating multi-scale flows for LDDKBM diffeomorphic registration*, in Proc IEEE International Conference on Computer Visions Workshops, 2011, pp. 499–505. 4
[131] A. Sotiras, C. Davatzikos, and N. Paragios, *Deformable medical image registration: A survey*, Medical Imaging, IEEE Transactions on, 32 (2013), pp. 1153–1190. 1, 3
[132] A. Staniforth and J. Côté, *Semi-Lagrangian integration schemes for atmospheric models—A review*, Montly Weather Review, 119 (1991), pp. 2206–2223. 2, 7
[133] M. Stoll and T. Breiten, *A low-rank in time approach to PDE-contrained optimization*, SIAM Journal on Scientific Computing, 37 (2015), pp. B1–B29. 3
[134] D. L. Tarnoki, A. D. Tarnoki, A. Richter, K. Karlinger, V. Berczi, and D. Pickuth, *Clinical value of whole-body magnetic resonance imaging in health screening of general adult population*, Radiology and Oncology, 49 (2015), pp. 10–16. 21
[135] Texas Advanced Computing Center, *TACC Webpage*. https://www.tacc.utexas.edu. 4, 10, 12
[136] R. Tomer, L. Ye, B. Hsueh, and K. Deisseroth, *Advanced CLARITY for rapid and high-resolution imaging of intact tissues*, Nature protocols, 9 (2014), pp. 1682–1697. 4, 21, 23
[137] A. Trouvé, *Diffeomorphism groups and pattern matching in image analysis*, International Journal of Computer Vision, 28 (1998), pp. 213–221. 1, 2, 3, 11
[138] T. ur Rehman, E. Haber, G. Pryor, J. Melonakos, and A. Tannenbaum, *3D nonrigid registration via optimal mass transport on the GPU*, Medical Image Analysis, 13 (2009), pp. 931–940. 3, 4
[139] P. Valero-Lara, *Multi-GPU acceleration of DARTEL (early detection of Alzheimer)*, in Proc IEEE International Conference on Cluster Computing, 2014, pp. 346–354. 4
[140] T. Vercauteren, X. Pennec, A. Perchant, and N. Ayache, *Diffeomorphic demons using ITK's finite difference solver hierarchy*, The Insight Journal, 1926/510 (2007). http://hdl.handle.net/1926/510. 2, 18
[141] ———, *Symmetric log-domain diffeomorphic registration: A demons-based approach*, in Proc Medical Image Computing and Computer-Assisted Intervention, vol. LNCS 5241, 2008, pp. 754–761. 2, 3, 4, 18
[142] ———, *Diffeomorphic demons: Efficient non-parametric image registration*, NeuroImage, 45 (2009), pp. S61–S72. 1, 2, 3, 4, 18
[143] 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. 3, 6
[144] L. C. Wilcox, G. Stadler, T. Bui-Thanh, and O. Ghattas, *Discretely exact derivatives for hyperbolic PDE-constrained optimization problems discretized by the discontinuous Galerkin method*, Journal of Scientific Computing, 63 (2015), pp. 138–162. 3
[145] L. Younes, *Jacobi fields in groups of diffeomorphisms and applications*, Quarterly of Applied Mathematics, 650 (2007), pp. 113–134. 1, 3
[146] ———, *Shapes and diffeomorphisms*, Springer, 2010. 1
[147] L. Younes, F. Arrate, and M. I. Miller, *Evolutions equations in computational anatomy*, NeuroImage, 45 (2009), pp. S40–S50. 1
[148] M. Zhang and P. T. Fletcher, *Bayesian principal geodesic analysis for estimating intrinsic diffeomorphic image variability*, Medical Image Analysis, 25 (2015), pp. 37–44. 1, 3
[149] M. Zhang and P. T. Fletcher, *Finite-dimensional Lie algebras for fast diffeomorphic image registration*, in Proc Information Processing in Medical Imaging, vol. 24, 2015, pp. 249–259. 3