SIAM J. IMAGING SCIENCES
Vol. 2, No. 1, pp. 84–109

A Curvilinear Search Method for p-Harmonic Flows on Spheres*

Donald Goldfarb†, Zaiwen Wen†, and Wotao Yin‡

Abstract. The problem of finding p-harmonic flows arises in a wide range of applications including color image (chromaticity) denoising, micromagnetics, liquid crystal theory, and directional diffusion. In this paper, we propose an innovative curvilinear search method for minimizing p-harmonic energies over spheres. Starting from a flow (map) on the unit sphere, our method searches along a curve that lies on the sphere in a manner similar to that of a standard inexact line search descent method. We show that our method is globally convergent if the step length satisfies the Armijo–Wolfe conditions. Computational tests are presented to demonstrate the efficiency of the proposed method and a variant of it that uses Barzilai–Borwein steps.

Key words. energy minimization, p-harmonic maps, p-harmonic flows, finite difference, curvilinear search, global convergence, chromaticity denoising

AMS subject classifications. 58E20, 65N06, 65N12, 65M06, 68U10, 90C26, 90C30, 94A08

DOI. 10.1137/080726926

1. Introduction.

We introduced an algorithm in this paper to efficiently solve the p-harmonic flow problem, a functional minimization problem subject to a nonconvex constraint. This problem has applications in color image (chromaticity) denoising [12, 30, 39, 40], micromagnetics [27], liquid crystal theory [1, 16, 17, 29], directional diffusion [35], etc.

Let $|\cdot|$ denote the Euclidean norm, S^{N-1} denote the $(N-1)$-sphere in $\mathbb{R}^N$, i.e., $S^{N-1} := \{\mathbf{U} \in \mathbb{R}^N : |\mathbf{U}| = 1\}$, and Ω denote an open subset of $\mathbb{R}^M$, where $M \geq 1$ and $N \geq 2$. We propose, for the case $N = 3$ and $M = 2$, simple and very efficient methods for solving the minimization problem

$$\min \ \widehat{E}_p(\mathbf{U}) = \int_\Omega |\nabla \mathbf{U}(\mathbf{x})|_F^p \, d\mathbf{x}, \quad \text{subject to (s.t.) } \mathbf{U} \in W^{1,p}_{\mathbf{n}_0}(\Omega, S^{N-1}), \tag{1.1}$$

where $1 \leq p < \infty$;

$$W^{1,p}_{\mathbf{n}_0}(\Omega, S^{N-1}) := \{\mathbf{U} \in W^{1,p}(\Omega, \mathbb{R}^N) \mid \mathbf{U}(\mathbf{x}) \in S^{N-1} \text{ a.e.}; \mathbf{U}|_{\partial\Omega} = \mathbf{n}_0\}; \tag{1.2}$$

*Received by the editors June 12, 2008; accepted for publication (in revised form) September 30, 2008; published electronically January 21, 2009.
http://www.siam.org/journals/siims/2-1/72692.html
†Department of Industrial Engineering and Operations Research, Columbia University, New York, NY 10027 (goldfarb@columbia.edu, zw2109@columbia.edu). The research of these authors was supported in part by NSF grant DMS 06-06712, ONR grants N00014-03-0515 and N00014-8-1-1118, and DOE grants DE-FG01-02ER-25126 and DE-FG02-08-25856.
‡Department of Computational and Applied Mathematics, Rice University, Houston, TX 77005 (wotao.yin@rice.edu). This author's research was supported in part by NSF CAREER Award DMS-07-48839 and ONR grant N00014-08-1-1101.

$|\cdot|_F$ is the Frobenius norm, i.e., $|B|_F = \sqrt{\sum_{i,j} B_{i,j}^2}$; the operator ∇ denotes differentiation, i.e.,

$$\nabla \mathbf{U}(\mathbf{x}) = \begin{pmatrix} \frac{\partial \mathbf{U}_1(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial \mathbf{U}_1(\mathbf{x})}{\partial x_M} \\ & \cdots & \\ \frac{\partial \mathbf{U}_N(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial \mathbf{U}_N(\mathbf{x})}{\partial x_M} \end{pmatrix}$$

is the Jacobian matrix of $\mathbf{U}$ at $\mathbf{x}$; and $\mathbf{n}_0$ is a given function from the boundary $\partial\Omega$ to S^{N-1}. In what follows, we use the simpler notation $|\mathbf{U}| = 1$ to represent the constraint $\mathbf{U} \in W^{1,p}_{\mathbf{n}_0}(\Omega, S^{N-1})$. The mappings $\mathbf{U}$ that are stationary points of problem (1.1) are called *p-harmonic maps* and, in particular, *harmonic maps* for $p = 2$. The analytical properties of (1.1), including the existence, nonuniqueness, regularity, and singularities of minimizing harmonic maps, have been intensively studied [13, 14, 15, 18, 24, 25, 26, 28].

Several types of numerical approaches have been proposed for solving problem (1.1). The approach in [16, 17] solves the Euler–Lagrange equations for (1.1) iteratively, where at each step the spherical constraints are ignored at first and then the solution $\mathbf{V}$ is renormalized by setting $\mathbf{U} = \frac{\mathbf{V}}{|\mathbf{V}|}$. This renormalization approach is analyzed in [1], where it is shown that the energy is decreased after each renormalized step and a convergent algorithm is proposed. Related finite element methods are studied in [2, 3]. By modifying a discretization scheme for the heat flow equations corresponding to (1.1), constraint preserving finite element methods are developed in [7, 4]. A second approach [9, 10, 32] adds a penalty term to the objective function in (1.1) to penalize violation of the spherical constraint, i.e., it forms

$$\mathcal{P}_\epsilon(\mathbf{U}) = \int_\Omega |\nabla \mathbf{U}(\mathbf{x})|_F^p \, d\mathbf{x} + \frac{1}{\epsilon} \int_\Omega (|\mathbf{U}|^2 - 1)^2 \, d\mathbf{x},$$

and then solves a sequence of unconstrained minimization problems $\min \mathcal{P}_\epsilon(\mathbf{U})$ by letting $\epsilon \to 0$. This approach is also used to solve the minimization of the Ginzburg–Landau functional. A third approach [11, 41] is based on solving the unconstrained problem

$$\min_{\mathbf{U}} E_p(\mathbf{U}) = \int_\Omega \left| \nabla \left(\frac{\mathbf{U}}{|\mathbf{U}|} \right) \right|_F^p d\mathbf{x}, \quad \text{s.t. } \mathbf{U} \in W^{1,p}_{\mathbf{n}_0}(\Omega, \mathbb{R}^N). \tag{1.3}$$

A parameterization of the variable $\mathbf{U}$ is employed to derive a constraint preserving gradient descent method.

We propose here a method for solving (1.1) that is faster than either of the above three approaches. It generates a sequence $\{\mathbf{U}^n\}$ based on an updating formula that *preserves* $|\mathbf{U}^n| = 1$ *and does not involve renormalization*, and also uses advanced line search techniques. Every update from $\mathbf{U}^n$ to $\mathbf{U}^{n+1}$ is determined by a descent direction on the manifold $W^{1,p}_{\mathbf{n}_0}(\Omega, S^{N-1})$ and a step size; however, the updating formula preserves $|\mathbf{U}^{n+1}| = 1$ for any direction and step size. This important property allows us to directly apply classical optimization techniques developed for use in Euclidean spaces such as line search methods and Barzilai–Borwein step sizes to significantly accelerate convergence, resulting in a framework that we refer to as a *curvilinear method*. In addition, at least for $N \leq 3$, we show that the updating formula is simple and easy to compute. In contrast, the algorithms from [2, 3] require an additional fixed point method to compute the new trial point. The algorithm for S^1 mappings proposed in

[41] can be viewed as a special case of our method; however, for the S^2 case, the algorithm in [41] requires the solution of a nonlinear system of equations at each step to maintain the pointwise S^2 constraints.

This paper is organized as follows. In section 2.1, we derive a descent method that preserves $|\mathbf{U}| = 1$ from the Euler–Lagrange equations for (1.1). In section 2.2, we present an equivalent representation of the unconstrained objective functional in (1.3) for the case $N = 3$. We then introduce in section 2.3 a discrete counterpart of the descent method described in section 2.1. In section 2.4, we present a curvilinear search method applied to the discrete formulation and prove global convergence to a stationary point. In section 3 we describe how to incorporate Barzilai–Borwein steps into our curvilinear search framework. Finally a set of numerical results on both synthetic and real problems is presented in section 4 to demonstrate the efficiency of our algorithms.

2. Continuous and discrete descent methods.

2.1. A descent method that preserves $|\mathbf{U}^n| = 1$. We henceforth take N to be 3, unless otherwise specified. It is well known that the Euler–Lagrange equations (i.e., the first-order optimality conditions) for problems (1.1) and (1.3) are the set of coupled PDEs (assuming they are well defined)

$$\begin{cases} \nabla \widehat{E}_p(\mathbf{U}) := -\Delta_p \mathbf{U} - |\nabla \mathbf{U}|^p \, \mathbf{U} &= 0, \\ |\mathbf{U}| &= 1 \quad \text{a.e. on } \Omega_T, \\ \mathbf{U}|_{\partial\Omega} &= \mathbf{n}_0, \end{cases} \tag{2.1}$$

where $\nabla \widehat{E}_p(\mathbf{U})$ denotes the Fréchet derivative of $\widehat{E}_p(\mathbf{U})$ with respect to $\mathbf{U}$ and $\Delta_p \mathbf{U} := \nabla \cdot (|\nabla \mathbf{U}|^{p-2} \nabla \mathbf{U})$ and $\nabla\cdot$ is the divergence operator. From $|\mathbf{U}| = 1$ it follows that $(\nabla \mathbf{U})^\top \mathbf{U} = \mathbf{0}$; hence from this and the product rule for differentiation we obtain

$$0 = \nabla \cdot \left(|\nabla \mathbf{U}|^{p-2} \left(\nabla \mathbf{U}^\top \mathbf{U} \right) \right) = \nabla \cdot \left(\left(|\nabla \mathbf{U}|^{p-2} \nabla \mathbf{U} \right)^\top \mathbf{U} \right) = |\nabla \mathbf{U}|^p + \langle \mathbf{U}, \Delta_p \mathbf{U} \rangle,$$

which implies that $-|\nabla \mathbf{U}|^p = \langle \mathbf{U}, \Delta_p \mathbf{U} \rangle$ and

$$\nabla \widehat{E}_p(\mathbf{U}) = -|\nabla \mathbf{U}|^p \, \mathbf{U} - \Delta_p \mathbf{U} = \langle \mathbf{U}, \Delta_p \mathbf{U} \rangle \, \mathbf{U} - \langle \mathbf{U}, \mathbf{U} \rangle \Delta_p \mathbf{U} = \mathbf{U} \times (\mathbf{U} \times \Delta_p \mathbf{U}),$$

where the second equality follows from Lagrange's formula $\mathbf{a} \times (\mathbf{b} \times \mathbf{c}) = \langle \mathbf{a}, \mathbf{c} \rangle \mathbf{b} - \langle \mathbf{a}, \mathbf{b} \rangle \mathbf{c}$.

Given a current point $\mathbf{U}^n$ with $|\mathbf{U}^n| = 1$, the classical steepest descent method computes a new trial point $\mathbf{U}^{n+1}_{\mathrm{SD}}$ as

$$\mathbf{U}^n_{\mathrm{SD}}(\tau) = \mathbf{U}^n - \tau \nabla \widehat{E}_p(\mathbf{U}^n) = \mathbf{U}^n - \tau \mathbf{U}^n \times (\mathbf{U}^n \times \Delta_p \mathbf{U}^n), \tag{2.2}$$

where τ is a step size. In general, $\mathbf{U}^n_{\mathrm{SD}}(\tau)$ does not satisfy $|\mathbf{U}^n_{\mathrm{SD}}(\tau)| = 1$. Inspired by [41], we propose replacing the step direction

$$\mathbf{U}^n \times (\mathbf{U}^n \times \Delta_p \mathbf{U}^n)$$

by

$$\frac{\mathbf{U}^n(\tau) + \mathbf{U}^n}{2} \times (\mathbf{U}^n \times \Delta_p \mathbf{U}^n),$$

which yields the following method for computing a new trial point $\mathbf{U}^n(\tau)$:

$$\mathbf{U}^n(\tau) = \mathbf{U}^n - \tau \frac{\mathbf{U}^n(\tau) + \mathbf{U}^n}{2} \times (\mathbf{U}^n \times \Delta_p \mathbf{U}^n). \tag{2.3}$$

We note that (2.3) can be viewed as a Crank–Nicolson (C-N) type of discretization for p-harmonic heat flow $\partial \mathbf{U}/\partial \tau = -\nabla \widehat{E}_p(\mathbf{U})$. In a standard C-N method the term $\mathbf{U}^n \times \Delta_p \mathbf{U}^n$ would also be replaced by its average at the current and new points. Formula (2.3) gives a *simple* and *explicit* updating scheme for $\mathbf{U}^{n+1} := \mathbf{U}^n(\tau)$ with a step size τ as we show in Theorem 2.1 below. To obtain this result, we use the fact that the cross product in $\mathbb{R}^3$ can be expressed as a matrix-vector product. Specifically, if $\mathbf{a} = (a_1, a_2, a_3)^\top \in \mathbb{R}^3$ and $\mathbf{b} \in \mathbb{R}^3$, then $\mathbf{a} \times \mathbf{b} = \mathbf{a}^{(\times)}\mathbf{b}$, where the matrix $\mathbf{a}^{(\times)}$ is defined by

$$\mathbf{a}^{(\times)} = \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix}. \tag{2.4}$$

Using this result, we can obtain $\mathbf{U}^n(\tau)$ explicitly.

Theorem 2.1. *Let $N = 3$. For any vector $\mathbf{H^n}(\mathbf{x}) \in \mathbb{R}^3$, the solution of the update formula*

$$\mathbf{U}^n(\tau) = \mathbf{U}^n - \tau \frac{\mathbf{U}^n(\tau) + \mathbf{U}^n}{2} \times \mathbf{H^n} \tag{2.5}$$

with respect to $\mathbf{U}^n(\tau)$ is

$$\mathbf{U}^n(\tau) = \left(I - \frac{1}{2}(\mathbf{H}^n)^{(\times)}\right)^{-1} \left(I + \frac{1}{2}(\mathbf{H}^n)^{(\times)}\right) \mathbf{U}^n, \tag{2.6}$$

where I is the identity matrix in $\mathbb{R}^{3\times 3}$ and $(\mathbf{H}^n)^{(\times)}$ is the matrix of the form (2.4) corresponding to $\mathbf{H}^n$. In addition, (2.6) satisfies

$$|\mathbf{U}^n(\tau)| = |\mathbf{U}^n|. \tag{2.7}$$

Consequently, if $|\mathbf{U}^0| = 1$, it follows that $\{\mathbf{U}^n\}$ satisfies $|\mathbf{U}^n| = 1$ for all $n > 0$.

Proof. 1. Rearranging (2.5), we get

$$\mathbf{U}^n(\tau) - \mathbf{U}^n + \frac{\tau}{2}\mathbf{U}^n(\tau) \times \mathbf{H}^n + \frac{\tau}{2}\mathbf{U}^n \times \mathbf{H}^n = 0.$$

Using the matrix-vector form of the cross product (2.4) and the fact that $\mathbf{a} \times \mathbf{b} = -\mathbf{b} \times \mathbf{a}$ and collecting common terms, we obtain

$$\left(I - \frac{\tau}{2}(\mathbf{H}^n)^{(\times)}\right) \mathbf{U}^n(\tau) = \left(I + \frac{\tau}{2}(\mathbf{H}^n)^{(\times)}\right) \mathbf{U}^n. \tag{2.8}$$

Since $(\mathbf{H}^n)^{(\times)}$ is skew symmetric, the matrix $\left(I - \frac{\tau}{2}(\mathbf{H}^n)^{(\times)}\right)$ is nonsingular for any $\mathbf{H}^n$; in fact, $\det\left(I - \frac{\tau}{2}(\mathbf{H}^n)^{(\times)}\right) = 1 + \frac{1}{4}\tau^2\|\mathbf{H}^n\|^2$. Therefore, the system of equations (2.8) is solvable and we obtain (2.6).

2. Taking the inner product on both sides of (2.5) with $\frac{\mathbf{U}^n(\tau)+\mathbf{U}^n}{2}$ and rearranging terms gives

$$\begin{aligned} 0 &= \left\langle \mathbf{U}^n(\tau) - \mathbf{U}^n, \frac{\mathbf{U}^n(\tau)+\mathbf{U}^n}{2} \right\rangle + \left\langle \tau \frac{\mathbf{U}^n(\tau)+\mathbf{U}^n}{2} \times \mathbf{H}^{\mathbf{n}}, \frac{\mathbf{U}^n(\tau)+\mathbf{U}^n}{2} \right\rangle \\ &= \frac{1}{2}\left(|\mathbf{U}^n(\tau)|^2 - |\mathbf{U}^n|^2 \right), \end{aligned}$$

where we have used the fact that the second term in the first equation is equal to zero because of the following properties of the cross product: $\langle \mathbf{a} \times \mathbf{b}, \mathbf{c} \rangle = -\langle \mathbf{b}, \mathbf{a} \times \mathbf{c} \rangle$ and $\mathbf{a} \times \mathbf{a} = \mathbf{0}$ for any $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{R}^3$. This result also follows from (2.6) and the fact that $B^\top B = I$ if $B = (I-K)^{-1}(I+K)$ and K is skew symmetric. ∎

Theorem 2.2. *Let* $\mathbf{U}_{\mathrm{SD}}^{n+1} = \mathbf{U}_{\mathrm{SD}}^n(\tau)$ *and* $\mathbf{U}^{n+1} = \mathbf{U}^n(\tau)$ *be defined by* (2.2) *and* (2.3), *respectively. The step* $\mathbf{U}^{n+1} - \mathbf{U}^n$ *is a descent step if the steepest descent step* $\mathbf{U}_{\mathrm{SD}}^{n+1} - \mathbf{U}^n$ *with the same step size* τ *is a descent step.*

Proof. It is easy to verify that

$$\mathbf{U}^{n+1} - \mathbf{U}_{\mathrm{SD}}^{n+1} = \frac{\tau}{2}(\mathbf{H}^n)^{(\times)}\left(\mathbf{U}^{n+1} - \mathbf{U}^n\right),$$

where $\mathbf{H}^n = \mathbf{U}^n \times \Delta_p \mathbf{U}^n$. Since $(\mathbf{H}^n)^{(\times)}$ is skew symmetric,

$$\left(\mathbf{U}^{n+1} - \mathbf{U}^n\right)^\top \left(\mathbf{U}^{n+1} - \mathbf{U}_{\mathrm{SD}}^{n+1}\right) = 0.$$

Hence

$$\begin{aligned} \left(\mathbf{U}^{n+1} - \mathbf{U}^n\right)^\top \left(\mathbf{U}_{\mathrm{SD}}^{n+1} - \mathbf{U}^n\right) &= \left(\mathbf{U}^{n+1} - \mathbf{U}^n\right)^\top \left(\mathbf{U}_{\mathrm{SD}}^{n+1} - \mathbf{U}^{n+1} + \mathbf{U}^{n+1} - \mathbf{U}^n\right) \\ &= |\mathbf{U}^{n+1} - \mathbf{U}^n|^2 > 0, \end{aligned}$$

since $\mathbf{U}^{n+1} \neq \mathbf{U}^n$ if $\mathbf{U}_{\mathrm{SD}}^{n+1} \neq \mathbf{U}^n$. ∎

In [7] the authors take $\mathbf{H}^{\mathbf{n}}$ as $\frac{\mathbf{U}^{n+1}+\mathbf{U}^n}{2} \times \Delta_p \mathbf{U}^{n+1}$ in a finite element discretization for the gradient flow equation and use a fixed point method to compute $\mathbf{U}^{n+1}$. Similar results are also presented in [6, 5]

Remark 2.3. The updating formula (2.3) can be applied to solve the general minimization problem

$$\min \mathcal{F}(\mathbf{U}), \text{ s.t. } \mathbf{U} \in S^2. \tag{2.9}$$

From the theory of Riemannian manifolds, the Euler–Lagrange equations for problem (2.9), given by the covariant derivative, are

$$\Pi_{\mathbf{U}}(\nabla \mathcal{F}) = \nabla \mathcal{F} - \langle \mathbf{U}, \nabla \mathcal{F} \rangle \mathbf{U} = 0, \quad \mathbf{U} \in S^2, \tag{2.10}$$

where $\Pi_{\mathbf{U}}$ is the orthogonal projection from $T_{\mathbf{U}}\mathbb{R}^3$ onto the tangent space $T_{\mathbf{U}}S^2$ [12, 31]. From the Lagrange formula $\mathbf{a} \times (\mathbf{b} \times \mathbf{c}) = \langle \mathbf{a}, \mathbf{c} \rangle \mathbf{b} - \langle \mathbf{a}, \mathbf{b} \rangle \mathbf{c}$, we obtain

$$\Pi_{\mathbf{U}}(\nabla \mathcal{F}) = -\mathbf{U} \times (\mathbf{U} \times \nabla \mathcal{F}), \quad \mathbf{U} \in S^2,$$

which implies that an updating formula similar to (2.3) can be used to solve (2.9).

2.2. An equivalent objective functional. In this subsection, we present an objective functional equivalent to (1.3) for introducing the discretization scheme in section 2.3.

By expanding the integral $|\nabla(\frac{\mathbf{U}}{|\mathbf{U}|})|_F^p$ in (1.3) for $\mathbf{U} = (u, v, w) \in \mathbb{R}^3$, we obtain

(2.11)
$$\min_{\mathbf{U}\in W_{\mathbf{n}_0}^{1,p}(\Omega,\mathbb{R}^N)} E_p(\mathbf{U}) = \int_\Omega \left(\left| \frac{u\nabla v - v\nabla u}{u^2+v^2+w^2} \right|^2 + \left| \frac{u\nabla w - w\nabla u}{u^2+v^2+w^2} \right|^2 + \left| \frac{v\nabla w - w\nabla v}{u^2+v^2+w^2} \right|^2 \right)^{p/2} \mathrm{d}\mathbf{x},$$

which is equivalent to problems (1.1) and (1.3).

Although the minimization problems (1.1) and (1.3) (and, hence, (2.11) for $N = 3$) have the same infimum, there are advantages to solving (1.3) (hence, (2.11)) instead of the equivalent problem (1.1). In (1.1) the explicit constraints $|\mathbf{U}| = 1$ are all nonconvex and thus difficult to handle numerically; in contrast, problem (1.3) or (2.11) is unconstrained, and one can directly apply nonlinear optimization methods to it. For (2.11), we give below a finite difference discretization which preserves the results in Theorems 2.1 and 2.2 and for which we derive a globally convergent and efficient iterative algorithm. Furthermore, we do not need to renormalize $\mathbf{U}$, either at every iteration or when our algorithm terminates (at a steady state).

Formula (2.11) can be extended to higher dimensional spaces, i.e., $N \geq 3$, by expanding the integral $|\nabla(\frac{\mathbf{U}}{|\mathbf{U}|})|_F^p$ explicitly. Also let us remark that (2.11) for $N = 2$, i.e., when $\mathbf{U} = (u, v)$ maps to S^1, can be simplified. Needless to say, all formulas and results developed below for $N = 3$ can also be simplified and applied to $N = 2$.

Remark 2.4. For p-harmonic mappings into S^1, we simply take $w = 0$; hence problem (1.3) becomes

$$\min E_p(\mathbf{U}) = \int_\Omega \left| \frac{u\nabla v - v\nabla u}{u^2+v^2} \right|^p \mathrm{d}\mathbf{x},$$

which has a gradient flow governed by

$$u_t + v\mathbf{H} = 0, \quad v_t - u\mathbf{H} = 0, \tag{2.12}$$

where $\mathbf{H} = u\nabla\cdot(|\nabla\mathbf{U}|^{p-2}\nabla v) - v\nabla\cdot(|\nabla\mathbf{U}|^{p-2}\nabla u)$.

Remark 2.5. For $N = 2$, i.e., mappings $\mathbf{U} = (u, v)$ into S^1, Vese and Osher [41] parametrize u and v by (r, θ) using $u = r\cos(\theta)$ and $v = r\sin(\theta)$. It can be shown that $|\nabla(\frac{\mathbf{U}}{|\mathbf{U}|})|_F^p \equiv |\nabla\theta|^p$. The authors calculate the gradient flow of the functional $\int_\Omega |\nabla\theta|^p \,\mathrm{d}\mathbf{x}$ with respect to θ. Using the transformation $\theta = \tan^{-1}\left(\frac{v}{u}\right)$ and the facts that $u^2 + v^2 = 1$ and $\nabla\theta = \frac{u\nabla v - v\nabla u}{u^2+v^2}$, they reformulate the gradient flow in (u, v) as

$$\begin{aligned} &uu_t + vv_t = 0, \\ &\frac{uv_t - vu_t}{u^2+v^2} = \nabla\cdot\left(\left| \frac{u\nabla v - v\nabla u}{u^2+v^2} \right|^{p-2} \frac{u\nabla v - v\nabla u}{u^2+v^2} \right), \end{aligned}$$

which they then further simplify into a set of equations similar to (2.12). From this, an updating scheme similar to (2.3) is proposed. Our approach generalizes theirs for p-harmonic maps into S^2.

In the next subsection we take a "discretize-then-optimize" approach, where the objective functional (2.11) is first discretized to obtain a standard nonlinear programming problem

in a finite dimensional space and then one solves this problem by a nonlinear programming algorithm. In an "optimize-then-discretize" approach, the updating formula (2.3), instead of the objective functional (2.11), is the starting point of the discretization. Given a specific discretization of (2.3), it is not easy to figure out the exact (discretized) objective function that the discretized version of (2.3) minimizes, so it is not clear how to specify certain algorithmic steps such as a line search and a convergence test.

Updating formula (2.3) and Theorem 2.1 provide guidelines for designing a discretized energy function for problem (2.11) (i.e., problem (1.3)). If the gradient of the discretized energy function at the point $\mathbf{U}^n$ can be represented in the form of $\mathbf{U}^n \times \mathbf{H}^n$ for some quantity $\mathbf{H}^n \in \mathbb{R}^3$, an updating scheme similar to (2.3) can be derived and the pointwise constraint $|\mathbf{U}^n| = 1$ can be maintained at all of the iterations.

2.3. A finite difference discretization scheme. For simplicity, we assume that the domain Ω is a rectangle and discretize it as the grid:

$$\Omega_h \stackrel{def}{=} \{(x_i, y_i) \ | x_i = ih_x, y_j = jh_y, i = 0, 1, \ldots, m;\ j = 0, 1, \ldots, n\}, \tag{2.13}$$

where h_x, h_y are the grid widths in the x and the y directions, respectively. We let $u_{i,j} = u(x_i, y_i)$, $v_{i,j} = v(x_i, y_i)$ and $w_{i,j} = w(x_i, y_i)$ at each grid point (x_i, y_i). A key aspect of our discretization is the use of the mean operators $\zeta_x u$ and $\zeta_y u$, which are defined as

$$\zeta_x u_{i,j} = \frac{u_{i,j} + u_{i-1,j}}{2}, \quad \zeta_y u_{i,j} = \frac{u_{i,j} + u_{i,j-1}}{2},$$

with respect to the x direction and y direction, respectively. To match the mean operators $\zeta_x u$ and $\zeta_y u$, the backward finite difference operators $\delta_x u$ and $\delta_y u$, which are defined as

$$\delta_x u_{i,j} = \frac{u_{i,j} - u_{i-1,j}}{h_x}, \quad \delta_y u_{i,j} = \frac{u_{i,j} - u_{i,j-1}}{h_y},$$

are used to approximate the partial derivatives ∇u. Similar mean operators and finite difference operators are defined for v and w. As shown in the following lemma, the property $(\nabla \mathbf{U})^\top \mathbf{U} = \mathbf{0}$ for $|\mathbf{U}| = 1$ holds for the $\mathbf{U}$ and $\nabla \mathbf{U}$ given by the means and finite differences defined above, respectively. This result plays an important role in the derivation of the Euler–Lagrange equations and the cross-product form of the gradient.

Lemma 2.6. *If $|U_{i,j}| = |(u_{i,j}, v_{i,j}, w_{i,j})| = 1$ for all i and j, then*

$$\zeta_x u_{i,j} \delta_x u_{i,j} + \zeta_x v_{i,j} \delta_x v_{i,j} + \zeta_x w_{i,j} \delta_x w_{i,j} = 0, \tag{2.14}$$

$$\zeta_y u_{i,j} \delta_y u_{i,j} + \zeta_y v_{i,j} \delta_y v_{i,j} + \zeta_y w_{i,j} \delta_y w_{i,j} = 0. \tag{2.15}$$

The proof is trivial.

Next, we define

$$(f_1^x)_{i,j} := (f_1^x)_{i,j}(u, v) = \frac{\zeta_x u_{i,j} \delta_x v_{i,j} - \zeta_x v_{i,j} \delta_x u_{i,j}}{S^x_{i,j}},$$

$$(f_1^y)_{i,j} := (f_1^y)_{i,j}(u, v) = \frac{\zeta_y u_{i,j} \delta_y v_{i,j} - \zeta_y v_{i,j} \delta_y u_{i,j}}{S^y_{i,j}},$$

where $S^x_{i,j} = (\zeta_x u_{i,j})^2 + (\zeta_x v_{i,j})^2 + (\zeta_x w_{i,j})^2$ and $S^y_{i,j} = (\zeta_y u_{i,j})^2 + (\zeta_y v_{i,j})^2 + (\zeta_y w_{i,j})^2$. Similarly, we define $(f^x_2)_{i,j}(u,w)$, $(f^y_2)_{i,j}(u,w)$, $(f^x_3)_{i,j}(v,w)$, and $(f^y_3)_{i,j}(v,w)$. The terms $(f^x_k)_{i,j}$ and $(f^y_k)_{i,j}$ for $k = 1,2,3$ are well defined if $S^x_{i,j} \neq 0$ or $S^y_{i,j} \neq 0$, and they are used to approximate the terms

$$\frac{u\nabla v - v\nabla u}{u^2+v^2+w^2}, \quad \frac{u\nabla w - w\nabla u}{u^2+v^2+w^2}, \quad \frac{v\nabla w - w\nabla v}{u^2+v^2+w^2}$$

in the x and y directions, respectively. Furthermore, we introduce

$$F_{i,j} = \left((f^x_1)^2_{i,j} + (f^y_1)^2_{i,j} + (f^x_2)^2_{i,j} + (f^y_2)^2_{i,j} + (f^x_3)^2_{i,j} + (f^y_3)^2_{i,j} + \xi\right)^{p/2},$$

where ξ ($\xi = 0$ if p is even and $\xi > 0$ if p is odd) is a small perturbation to avoid non-differentiability. Using the above definitions, the objective functional in (2.11) is discretized as

$$E_p(\mathbf{U}) = \sum_{i=1}^{m}\sum_{j=1}^{n} F_{i,j} \tag{2.16}$$

by abusing the notation and letting $\mathbf{U} = \{(u_{i,j}, v_{i,j}, w_{i,j})\}$.

We show below in Lemma 2.7, proved in Appendix B, that the partial derivatives of the discrete function $E_p(\mathbf{U})$ with respect to $u_{i,j}$, $v_{i,j}$, and $w_{i,j}$ can be represented in a cross-product form similar to their continuous counterparts (2.2).

Lemma 2.7. *Suppose $|\mathbf{U}_{i,j}| = 1$ for all i and j. The partial derivatives of $E_p(\mathbf{U})$ with respect to variables $\mathbf{U}_{i,j} = (u_{i,j}, v_{i,j}, w_{i,j})$ for each $i = 1,\dots,m-1$ and $j = 1,\dots,n-1$ are*

$$(\nabla E_p(\mathbf{U}))_{i,j} = \mathbf{U}_{i,j} \times \mathbf{H}_{i,j}, \tag{2.17}$$

where $\mathbf{H} = (\mathbf{H}_a, \mathbf{H}_b, \mathbf{H}_c)$ and

(2.18)
$$\begin{cases}
(\mathbf{H}_c)_{i,j} = +p\left(-F_{i,j}^{\frac{p-2}{p}}\left(\dfrac{(f^x_1)_{i,j}}{h_x S^x_{i,j}} + \dfrac{(f^y_1)_{i,j}}{h_y S^y_{i,j}}\right) + F_{i+1,j}^{\frac{p-2}{p}}\dfrac{(f^x_1)_{i+1,j}}{h_x S^x_{i+1,j}} + F_{i,j+1}^{\frac{p-2}{p}}\dfrac{(f^y_1)_{i,j+1}}{h_y S^y_{i,j+1}}\right),\\
(\mathbf{H}_b)_{i,j} = -p\left(-F_{i,j}^{\frac{p-2}{p}}\left(\dfrac{(f^x_2)_{i,j}}{h_x S^x_{i,j}} + \dfrac{(f^y_2)_{i,j}}{h_y S^y_{i,j}}\right) + F_{i+1,j}^{\frac{p-2}{p}}\dfrac{(f^x_2)_{i+1,j}}{h_x S^x_{i+1,j}} + F_{i,j+1}^{\frac{p-2}{p}}\dfrac{(f^y_2)_{i,j+1}}{h_y S^y_{i,j+1}}\right),\\
(\mathbf{H}_a)_{i,j} = +p\left(-F_{i,j}^{\frac{p-2}{p}}\left(\dfrac{(f^x_3)_{i,j}}{h_x S^x_{i,j}} + \dfrac{(f^y_3)_{i,j}}{h_y S^y_{i,j}}\right) + F_{i+1,j}^{\frac{p-2}{p}}\dfrac{(f^x_3)_{i+1,j}}{h_x S^x_{i+1,j}} + F_{i,j+1}^{\frac{p-2}{p}}\dfrac{(f^y_3)_{i,j+1}}{h_y S^y_{i,j+1}}\right).
\end{cases}$$

Remark 2.8. Forward finite differences or central finite differences can also be used, and they give similar results with appropriate mean operators.

2.4. A discrete descent method and curvilinear search algorithm. In this subsection, we describe a method for solving the unconstrained discretized problem (2.16), based on the special structure of the gradient (2.17). Our updating scheme is analogous to (2.3). Essentially, we define a curve from the current point $\mathbf{U}^n$ on the surface of the sphere S^2 and search along it for a new point $\mathbf{U}^{n+1}$.

The steepest descent method computes a new point $\mathbf{U}^n(\tau)$ by the formula

$$\mathbf{U}^n_{i,j}(\tau) = \mathbf{U}^n_{i,j} - \tau\ (\nabla E_p(\mathbf{U}^n))_{i,j}\,, \tag{2.19}$$

where $\nabla E_p(\mathbf{U}^n)$ is given by (2.17). Since in general $|\mathbf{U}^n_{i,j}(\tau)| \neq 1$, we modify (2.19) in the same way as in Theorem 2.1 and obtain

$$\mathbf{U}^n_{i,j}(\tau) = \mathbf{U}^n_{i,j} - \tau\,\frac{\mathbf{U}^n_{i,j}(\tau) + \mathbf{U}^n_{i,j}}{2} \times \mathbf{H}^n_{i,j}, \tag{2.20}$$

which is clearly linear in $\mathbf{U}^n_{i,j}(\tau)$. Below we derive the candidate point $\mathbf{U}^n(\tau)$ and its derivative with respect to τ and show that $|\mathbf{U}^n(\tau)| = |\mathbf{U}^n|$ for any τ similar to Theorem 2.1.

Theorem 2.9. *For any τ, the solution $\mathbf{U}^n_{i,j}(\tau)$ of the system of equations (2.20) with respect to $\mathbf{U}^n_{i,j}$ is*

$$\mathbf{U}^n_{i,j}(\tau) = \left(\mathcal{W}^{n-}_{i,j}\right)^{-1} \mathcal{W}^{n+}_{i,j}\mathbf{U}^n_{i,j}, \tag{2.21}$$

where $\mathcal{W}^{n+}_{i,j} = I + \frac{\tau}{2}(\mathbf{H}^n_{i,j})^{(\times)}$, $\mathcal{W}^{n-}_{i,j} = I - \frac{\tau}{2}(\mathbf{H}^n_{i,j})^{(\times)}$, I is the identity matrix in $\mathbb{R}^{3\times 3}$, $\mathbf{H}^n_{i,j}$ is given in (2.18), and $(\mathbf{H}^n_{i,j})^{(\times)}$ is defined by (2.4). Moreover, $|\mathbf{U}^n_{i,j}(\tau)| = |\mathbf{U}^n_{i,j}|$, and the derivative of $\mathbf{U}^n_{i,j}(\tau)$ with respect to τ is

$$(\mathbf{U}^n_{i,j})'(\tau) = \frac{d\mathbf{U}^n_{i,j}(\tau)}{d\tau} = \frac{1}{2}\left(\mathcal{W}^{n-}_{i,j}\right)^{-1} (\mathbf{H}^n_{i,j})^{(\times)}\left(\mathbf{U}^n_{i,j} + \mathbf{U}^n_{i,j}(\tau)\right). \tag{2.22}$$

Proof. Except for the derivation of (2.22), the proof is identical to the proof of Theorem 2.1 with $\mathbf{U}^n$, $\mathbf{U}^n(\tau)$, and $\mathbf{H}^n$ replaced by $\mathbf{U}^n_{i,j}$, $\mathbf{U}^n_{i,j}(\tau)$, and $\mathbf{H}^n_{i,j}$, respectively. To obtain (2.22) we differentiate both sides of (2.21) with respect to τ, which gives

$$-\frac{1}{2}(\mathbf{H}^n_{i,j})^{(\times)}\mathbf{U}^n_{i,j}(\tau) + \left(I - \frac{\tau}{2}(\mathbf{H}^n_{i,j})^{(\times)}\right)\frac{d\mathbf{U}^n_{i,j}(\tau)}{d\tau} = \frac{1}{2}(\mathbf{H}^n_{i,j})^{(\times)}\mathbf{U}^n_{i,j}. \qquad \blacksquare$$

The matrix form of the updating formula (2.21) should be expanded to avoid inverting the matrices $\mathcal{W}^{n-}_{i,j}$ at each grid point (i,j). To simplify the notation, we temporarily drop the subscripts (i,j) and superscript n. Let $(\mathbf{H}_a)_{i,j}, (\mathbf{H}_b)_{i,j}, (\mathbf{H}_c)_{i,j}$ be denoted by a, b, c, respectively, and recall that $\mathbf{U}^n = (u, v, w)$ and $\mathbf{U}^n(\tau) = (u(\tau), v(\tau), w(\tau))$. Then the solution given by (2.21) can be expressed explicitly as

$$u(\tau) = \frac{4u + 4\tau bw - 4\tau cv - \tau^2 b^2 u + \tau^2 a^2 u - \tau^2 c^2 u + 2\tau^2 abv + 2\tau^2 acw}{4 + \tau^2 c^2 + \tau^2 a^2 + \tau^2 b^2}, \tag{2.23}$$

$$v(\tau) = \frac{4v + 4\tau cu - 4\tau aw + \tau^2 b^2 v - \tau^2 a^2 v - \tau^2 c^2 v + 2\tau^2 cbw + 2\tau^2 abu}{4 + \tau^2 c^2 + \tau^2 a^2 + \tau^2 b^2}, \tag{2.24}$$

$$w(\tau) = \frac{4w + 4\tau av - 4\tau bu - \tau^2 b^2 w - \tau^2 a^2 w + \tau^2 c^2 w + 2\tau^2 bcv + 2\tau^2 acu}{4 + \tau^2 c^2 + \tau^2 a^2 + \tau^2 b^2}. \tag{2.25}$$

Since a direct verification is tedious, a symbolic "MATLAB script" is given in Appendix A.

Remark 2.10. A curve $\mu(\tau) : \mathbb{R} \to S^2$ is a geodesic on the unit sphere S^2 if and only if

$$\ddot{\mu} = \langle \ddot{\mu}, \mu \rangle \mu. \tag{2.26}$$

It can be verified that the curve $\mathbf{U}^n(\tau)$ defined by formula (2.21) is not a geodesic since it does not satisfy (2.26). To further our understanding, we compare the curves generated by formula (2.21) with the two curves generated by the traditional steepest descent method followed by a projection onto S^2, called the normalization method, and by the geodesic steepest descent method on S^2. Specifically, the curve generated by the normalization method is defined as

$$\widehat{\mathbf{U}}^n(\tau) = \frac{\mathbf{U}^n - \tau \nabla E_p(\mathbf{U}^n)}{|\mathbf{U}^n - \tau \nabla E_p(\mathbf{U}^n)|}. \tag{2.27}$$

The geodesic steepest descent method searches along the geodesic of S^2 in the negative gradient direction. We first make sure that the gradient $\nabla E_p(\mathbf{U}^n)$ lies on the tangent plane $T_{\mathbf{U}^n} S^2$ by computing $\mathbf{d} = \Pi_{\mathbf{U}^n}(\nabla E_p(\mathbf{U}^n))$, where $\Pi_{\mathbf{U}} \mathbf{V} = \mathbf{V} - \langle \mathbf{V}, \mathbf{U} \rangle \mathbf{U}$ is the orthogonal projection from the tangent space $T_{\mathbf{U}} \mathbb{R}^3$ onto the tangent space $T_{\mathbf{U}} S^2$. Then the curve along the geodesic of the unit sphere [12] is defined as

$$\widetilde{\mathbf{U}}^n(\tau) = \cos(\tau |\mathbf{d}|) \mathbf{U}^n + \sin(\tau |\mathbf{d}|) \frac{\mathbf{d}}{|\mathbf{d}|}. \tag{2.28}$$

To obtain some intuition, we study Example 4.1 with $p = 1$ from section 4 on numerical implementation. We choose $\mathbf{U}^n$ as the initial point $\mathbf{U}^0$ given by (4.1), choose a grid point (i, j), and then compute the points $\mathbf{U}^n(\tau_n)$, $\widehat{\mathbf{U}}^n(\tau_n)$, and $\widetilde{\mathbf{U}}^n(\tau_n)$ for $\tau_n = k/3$, $k = 1, \ldots, 30$, on the curves $\mathbf{U}^n(\tau)$, $\widehat{\mathbf{U}}^n(\tau)$, and $\widetilde{\mathbf{U}}^n(\tau)$ defined by formulas (2.21), (2.27), and (2.28), respectively, corresponding to that grid point. The left-hand side of Figure 1 is the result corresponding to the grid point $(11, 11)$, and the right-hand side of Figure 1 is the result corresponding to the grid point $(11, 15)$. These plots show that the three curves are different.

It is well known that the steepest descent method with a fixed step size may not converge. By choosing the step size wisely, we can guarantee convergence and even accelerate the speed of convergence without greatly increasing the computational cost at each iteration. One approach is to minimize the objective value $E_p(\mathbf{U})$ along the curve $\mathbf{U}^n(\tau)$ with respect to τ, i.e., to obtain an optimal τ^* by solving

$$\min_{\tau} E_p(\mathbf{U}^n(\tau)).$$

Since finding a global minimizer τ^* is computationally expensive, one is usually satisfied with an approximate minimizer such as a τ_n satisfying the Armijo–Wolfe conditions [34, 38, 22]

$$E_p(\mathbf{U}^n(\tau_n)) \le E_p(\mathbf{U}^n(0)) + \rho_1 \tau_n E_p'(\mathbf{U}^n(0)), \tag{2.29a}$$

$$E_p'(\mathbf{U}^n(\tau_n)) \ge \rho_2 E_p'(\mathbf{U}^n(0)), \tag{2.29b}$$

where $E_p'(\mathbf{U}^n(0))$ and $E_p'(\mathbf{U}^n(\tau_n))$ are the derivatives of $E_p(\mathbf{U}^n(\tau))$ with respect to τ at $\tau = 0$ and $\tau = \tau_n$, respectively, and $0 < \rho_1 < \rho_2 < 1$ are two parameters. To select a step size τ_n to satisfy the Armijo–Wolfe conditions (2.29a) and (2.29b), we refer to Algorithms 3.2 and 3.3 in

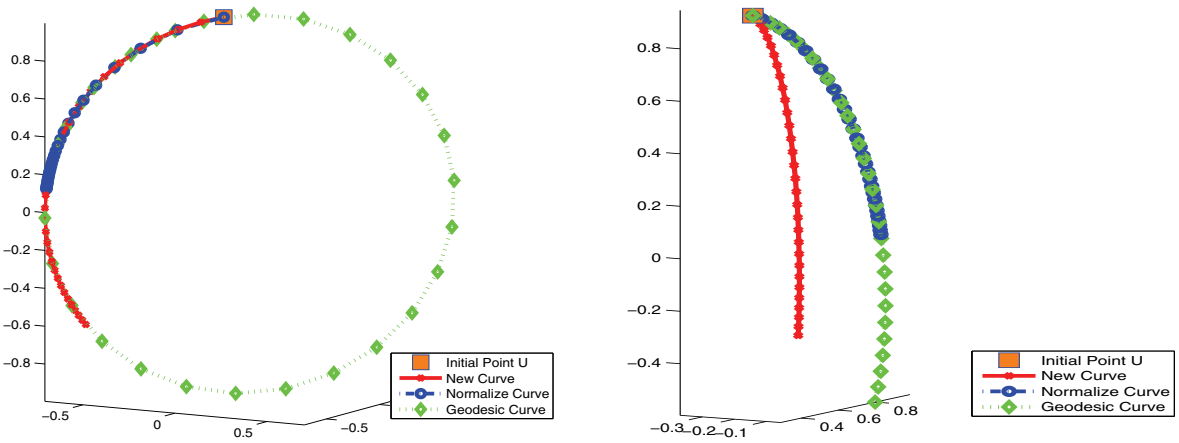


Figure 1. *Comparison of updating formulas: (left) the grid point* $(11, 11)$*; (right) the grid point* $(11, 15)$*. The solid curve marked by* $*$ *was generated by* (2.21)*, the dash-dot curve marked by* $\circ$ *was generated by the normalization method, and the dashed curve marked by* $\diamond$ *was generated by the geodesic steepest descent method.*

Algorithm 1. A gradient descent method with curvilinear search.

STEP 0: **Initialization**. Given an initial point $\mathbf{U}^0$ such that $|\mathbf{U}^0| = 1$. Set $n = 0$, $\epsilon \geq 0$, and $0 < \rho_1 < \rho_2 < 1$.

STEP 1: **Compute the step size** τ_n. Call line search along the path $\mathbf{U}^n(\tau)$ defined by (2.21) to obtain a step size τ_n that satisfies the Armijo–Wolfe conditions (2.29a) and (2.29b).

STEP 2: **Update**. Set the new trial point $\mathbf{U}^{n+1} = \mathbf{U}^n(\tau_n)$. If $\|\nabla E_p(\mathbf{U}^{n+1})\| \leq \epsilon$, then **STOP**. Set $n = n + 1$; goto STEP 1.

[34], which are based on interpolation and bisection. For a more detailed description of these kinds of strategies, see, for example, [33]. To summarize, we describe the curvilinear search approach in Algorithm 1.

Using the chain rule, the derivative of $E_p(\mathbf{U}^n(\tau))$ with respect to τ is

$$E_p'(\mathbf{U}^n(\tau)) = [\nabla E_p(\mathbf{U}^n(\tau))]^\top (\mathbf{U}^n)'(\tau), \tag{2.30}$$

where the partial derivatives $(\mathbf{U}^n)'(\tau)$ are given by (2.22). Using (2.22) we have

$$(\mathbf{U}_{i,j}^n)'(0) = (\mathbf{H}_{i,j}^n) \times \mathbf{U}_{i,j}^n = -(\nabla E_p(\mathbf{U}^n))_{i,j} \tag{2.31}$$

or

$$E_p'(\mathbf{U}^n(0)) = -\|\nabla E_p(\mathbf{U}^n)\|^2 \leq 0. \tag{2.32}$$

Moreover, (2.31) shows that the negative gradient is the direction of the trajectory $\mathbf{U}^n(\tau)$ at $\tau = 0$.

Since $E_p(\mathbf{U}(\tau))$ is continuously differentiable and bounded from below, it is not difficult to prove that there exists a τ_n satisfying the Armijo–Wolfe conditions (2.29a) and (2.29b).

Therefore, every iteration of Algorithm 1 is well defined. Formally, we have the following lemma.

Lemma 2.11 (see [34, Lemma 3.1]). *If $0 < \rho_1 < \rho_2 < 1$ and $E_p'(\mathbf{U}^n(0)) < 0$, there exist intervals of step lengths satisfying the Armijo–Wolfe conditions* (2.29a) *and* (2.29b).

To prove the global convergence of Algorithm 1, let us define the level set

$$\mathcal{L} = \{\mathbf{U} \,|\, E_p(\mathbf{U}) \le E_p(\mathbf{U}^0),\ |\mathbf{U}^0_{i,j}| = 1,\ |\mathbf{U}_{i,j}| = 1\}, \tag{2.33}$$

which is a compact set. Starting from a point $\mathbf{U}^0$ with $|\mathbf{U}^0_{i,j}| = 1$, the sequence $\{\mathbf{U}^n\}$ generated by Algorithm 1 stays in the level set $\mathcal{L}$ since $E_p(\mathbf{U}^n)$ is decreasing and $\mathbf{U}^n$ always satisfies the pointwise constraints $|\mathbf{U}^n_{i,j}| = 1$. We first show in Lemma 2.12 that $\lim_{n\in K} \|\mathbf{U}^n(\tau_n) - \mathbf{U}^n(0)\| = 0$ and $\lim_{n\in K} \|(\mathbf{U}^n)'(\tau_n) - (\mathbf{U}^n)'(0)\| = 0$ if $\tau_n \to 0$ for any subsequence $\{\mathbf{U}^n\}_{n\in K}$ generated by Algorithm 1.

Lemma 2.12. *Suppose that $\{\mathbf{U}^n\}_{n\in K}$ is an infinite subsequence generated by Algorithm 1. Then the sequence $\{\mathbf{U}^n(\tau_n)\}_{n\in K}$ defined by formula* (2.21) *satisfies $\lim_{n\in K} \|\mathbf{U}^n(\tau_n) - \mathbf{U}^n(0)\| = 0$ and $\lim_{n\in K} \|(\mathbf{U}^n)'(\tau_n) - (\mathbf{U}^n)'(0)\| = 0$ if $\lim_{n\in K} \tau_n = 0$.*

Proof. 1. It follows from (2.21) that

$$\mathbf{U}^n_{i,j}(\tau) - \mathbf{U}^n_{i,j}(0) = \left[\left(\mathcal{W}^{n-}_{i,j}\right)^{-1} \mathcal{W}^{n+}_{i,j} - I\right] \mathbf{U}^n_{i,j} = \widehat{\mathcal{W}}^n_{i,j} \mathbf{U}^n_{i,j},$$

and hence that

$$\left|\mathbf{U}^n_{i,j}(\tau) - \mathbf{U}^n_{i,j}(0)\right| \le \left\|\widehat{\mathcal{W}}^n_{i,j}\right\|_2 |\mathbf{U}^n_{i,j}| = \kappa^n_{i,j},$$

where $\kappa^n_{i,j} = \frac{2\tau_n |\mathbf{H}^n_{i,j}|}{\sqrt{4+\tau_n^2 |\mathbf{H}^n_{i,j}|^2}}$ is the 2-norm of the matrix $\widehat{\mathcal{W}}^n_{i,j}$ (see Appendix A for a procedure for determining this norm). Therefore, we obtain

$$\|\mathbf{U}^n(\tau_n) - \mathbf{U}^n(0)\|^2 = \sum_{i,j} |\mathbf{U}^n_{i,j}(\tau_n) - \mathbf{U}^n_{i,j}(0)|^2 \le \sum_{i,j} \left(\kappa^n_{i,j}\right)^2.$$

Since $\{\mathbf{H}^n_{i,j}\}_{n\in K}$ are continuous functions with respect to the variables $\mathbf{U}^n$, which stay in the compact level set $\mathcal{L}$, it follows that the sequences $\{\mathbf{H}^n_{i,j}\}_{n\in K}$ are uniformly bounded for all i and j. This, together with the fact that $\lim_{n\in K} \tau_n = 0$, gives $\lim_{n\in K} (\kappa^n_{i,j})^2 = 0$ and hence the result $\lim_{n\in K} \|\mathbf{U}^n(\tau_n) - \mathbf{U}^n(0)\| = 0$.

2. It follows from (2.21) and (2.22) in Theorem 2.9 that

$$\begin{aligned}
(\mathbf{U}^n_{i,j})'(\tau) - (\mathbf{U}^n_{i,j})'(0) &= \frac{1}{2}\left(\mathcal{W}^{n-}_{i,j}\right)^{-1} (\mathbf{H}^n_{i,j})^{(\times)} \left(\mathbf{U}^n_{i,j} + \mathbf{U}^n_{i,j}(\tau)\right) - (\mathbf{H}^n_{i,j})^{(\times)} \mathbf{U}^n_{i,j} \\
&= \left[\frac{1}{2}\left(\mathcal{W}^{n-}_{i,j}\right)^{-1} (\mathbf{H}^n_{i,j})^{(\times)} \left(I + \left(\mathcal{W}^{n-}_{i,j}\right)^{-1} \mathcal{W}^{n+}_{i,j}\right) - (\mathbf{H}^n_{i,j})^{(\times)}\right] \mathbf{U}^n_{i,j} \\
&= \widetilde{\mathcal{W}}^n_{i,j} \mathbf{U}^n_{i,j},
\end{aligned}$$

where the 2-norm of the matrix $\widetilde{\mathcal{W}}^n_{i,j}$ is

$$\widetilde{\kappa}^n_{i,j} = \sqrt{\frac{\tau_n^4 |\mathbf{H}^n_{i,j}|^6 + 16\tau_n^2 |\mathbf{H}^n_{i,j}|^4}{\tau_n^4 |\mathbf{H}^n_{i,j}|^4 + 8\tau_n^2 |\mathbf{H}^n_{i,j}|^2 + 16}}.$$

Therefore, a proof similar to that in part 1 gives $\lim_{n\in K}\|(\mathbf{U}^n)'(\tau_n)-(\mathbf{U}^n)'(0)\|=0$. ∎

We now study the convergence properties of the sequence $\{\mathbf{U}^n\}$ generated by Algorithm 1. Since $\{\mathbf{U}^n\}$ stays in the compact level set $\mathcal{L}$, there exists at least one accumulation point. Moreover, the following result shows that the sequence of gradient $\{\nabla E_p(\mathbf{U}^n)\}$ converges to zero.

Theorem 2.13. *Let $\{\mathbf{U}^n\}$ be the full sequence generated by Algorithm 1. Then*

$$\lim_{n\to\infty}\|\nabla E_p(\mathbf{U}^n)\|=0. \tag{2.34}$$

Proof. For a proof by contradiction we suppose that (2.34) does not hold. Then there exist a constant $\epsilon>0$ and a infinite index set $K\subseteq\mathbb{N}$ such that

$$\|\nabla E_p(\mathbf{U}^k)\|>\epsilon \text{ for all } n\in K. \tag{2.35}$$

It follows from Lemma 2.11 that a step size that satisfies the Armijo–Wolfe conditions (2.29a) and (2.29b) is well defined for each iteration. Summing the inequalities (2.29a), we obtain that

$$\sum_{n=0}^{\infty}\rho_1\tau_n\|\nabla E_p(\mathbf{U}^n)\|^2\le E_p(\mathbf{U}^0)-\lim_{n\to\infty}E_p(\mathbf{U}^n), \tag{2.36}$$

where the limit exists because of the descent property of $E_p(\mathbf{U}^n)$ and the boundedness of $\mathcal{L}$. Hence, we have $\tau_n\|\nabla E_p(\mathbf{U}^n)\|^2\to 0$, which implies that $\tau_n\to 0$ for $n\in K$ because of (2.35). Therefore, from Lemma 2.12, we have

$$\lim_{n\in K}\|\mathbf{U}^n(\tau_n)-\mathbf{U}^n(0)\|=0 \text{ and } \lim_{n\in K}\|(\mathbf{U}^n)'(\tau_n)-(\mathbf{U}^n)'(0)\|=0. \tag{2.37}$$

Using relation (2.32) and the curvature condition (2.29b), we obtain that, for all $n\in K$,

$$(1-\rho_2)\|\nabla E_p(\mathbf{U}^n)\|^2=(\rho_2-1)E_p'(\mathbf{U}^n(0))\le E_p'(\mathbf{U}^n(\tau_n))-E_p'(\mathbf{U}^n(0)).$$

Using relation (2.30), we have

$$\begin{aligned}&E_p'(\mathbf{U}^n(\tau_n))-E_p'(\mathbf{U}^n(0))\\&=\nabla E_p(\mathbf{U}^n(\tau_n))^\top(\mathbf{U}^n)'(\tau_n)-\nabla E_p(\mathbf{U}^n(0))^\top(\mathbf{U}^n)'(0)\\&=\nabla E_p(\mathbf{U}^n(\tau_n))^\top[(\mathbf{U}^n)'(\tau_n)-(\mathbf{U}^n)'(0)]+[\nabla E_p(\mathbf{U}^n(\tau_n))-\nabla E_p(\mathbf{U}^n(0))]^\top(\mathbf{U}^n)'(0).\end{aligned}$$

Therefore, it follows from the Cauchy–Schwarz inequality that

$$\begin{aligned}(1-\rho_2)\|\nabla E_p(\mathbf{U}^n)\|^2\le{}&\|\nabla E_p(\mathbf{U}^n(\tau_n))\|\,\|(\mathbf{U}^n)'(\tau_n)-(\mathbf{U}^n)'(0)\|\\&+\|\nabla E_p(\mathbf{U}^n(\tau_n))-\nabla E_p(\mathbf{U}^n(0))\|\,\|(\mathbf{U}^n)'(0)\|.\end{aligned} \tag{2.38}$$

Since $\nabla E_p(\mathbf{U})$ is continuous on the compact set $\mathcal{L}$ and recalling (2.37), we have

$$\lim_{n\in K}\|\nabla E_p(\mathbf{U}^n(\tau_n))-\nabla E_p(\mathbf{U}^n(0))\|=0.$$

Furthermore, $\{\|\nabla E_p(\mathbf{U}^n)\|\}_{n\in K}$ is bounded, and so is $\|(\mathbf{U}^n)'(0)\|$ since by (2.31) $\|(\mathbf{U}^n)'(0)\|=\|\nabla E_p(\mathbf{U}^n)\|$. These facts together with (2.37) imply that the right-hand side in (2.38) converges to zero as $n\in K$ tends to ∞. This, in turn, implies that $\lim_{n\in K}\|\nabla E_p(\mathbf{U}^n)\|=0$, which contradicts (2.35). ∎

Remark 2.14. If a backtracking line search is used in Algorithm 1, results similar to Theorem 2.13 still hold.

3. Accelerating the curvilinear search method. In this section we apply so-called Barzilai–Borwein (BB) steps [8] for a significant acceleration of convergence. Calculating BB steps requires less computation per iteration than performing a line search, and using them often significantly reduces the required number of iterations in a gradient descent method.

The BB method for solving

$$\min_{x\in\mathbb{R}^N} f(x) \tag{3.1}$$

has the form $x^{n+1} = x^n - \alpha_n g^n$, where $g^n = \nabla f(x^n)$ and α_n is determined by the information obtained at the points x^{n-1} and x^n. Let

$$s^{n-1} = x^n - x^{n-1}, \quad y^{n-1} = g^n - g^{n-1}.$$

Barzilai and Borwein choose the step size α_n so that the matrix $D^n = \alpha_n I$, which can be viewed as an approximation to the Hessian of f at x^n, has the quasi-Newton property $D^n y^{n-1} = s^{n-1}$. This yields

$$\alpha_n^1 = \frac{(s^{n-1})^\top s^{n-1}}{(s^{n-1})^\top y^{n-1}} \tag{3.2}$$

or

$$\alpha_n^2 = \frac{(s^{n-1})^\top y^{n-1}}{(y^{n-1})^\top y^{n-1}}. \tag{3.3}$$

For the discretized problem (2.16), one can apply either one of the step sizes (3.2) or (3.3), or use (3.2) and (3.3) alternatively on odd/even steps. On the very first iteration of the algorithm, s^0 and y^0 do not exist so the BB steps α_1^1 and α_1^2 are not defined and a line search must be performed. Since the steepest descent method with inexact line search usually works well in early iterations, our curvilinear search algorithm with BB steps uses this approach. The result is Algorithm 2.

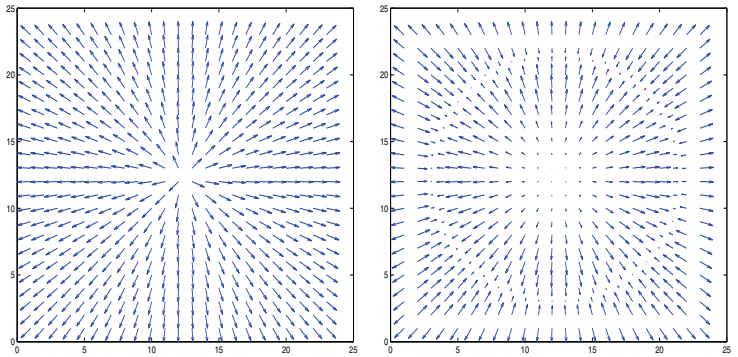
Algorithm 2. A curvilinear search method with BB steps.

STEP 0: **Initialization**. Given an initial point $\mathbf{U}^0$ such that $|\mathbf{U}^0| = 1$. Run Algorithm 1 for γ steps to return another initial solution $\mathbf{U}^\gamma$, where $\gamma \geq 2$ is a prescribed integer. Set $n = \gamma + 1$, $\epsilon \geq 0$.

STEP 1: **Compute the step size** τ_n. If n is odd, compute τ_n by rule (3.2); otherwise, compute τ_n by rule (3.3).

STEP 2: **Update**. Set the new trial point $\mathbf{U}^{n+1} = \mathbf{U}^n(\tau_n)$. If $\|\nabla E_p(\mathbf{U}^{n+1})\| \leq \epsilon$, then **STOP**. Set $n = n + 1$; goto STEP 1.

The BB method is a nonmonotone method since it does not decrease the objective value at every iteration. We note that there are many ways to improve the BB method, incorporating it into a globalization strategy while preserving its good features. For example, a nonmonotone line search strategy that guarantees global convergence when combined with the BB method is studied in [36]. An alternative step gradient method is proposed in [19]. For other references on the BB method, see, for example [20, 21] and the references therein.



(a) $\widetilde{\mathbf{U}}(\mathbf{x})$ (b) $\mathbf{U}^0(\mathbf{x})$

Figure 2. *Example* 4.1.

4. Numerical results. In this section, we demonstrate the effectiveness of our curvilinear search algorithms on two sets of test problems. We compared three algorithms: (1) the gradient flow method with a fixed step size, denoted by "fixed-step," (2) Algorithm 1 with Armijo–Wolfe line search, denoted by "curve-ls," and (3) Algorithm 2, denoted by "curve-BB." All codes were written in MATLAB (Release 7.3.0); the curvilinear search code is based upon the code "DCSRCH" [33] with an initial step size of 10^{-2} and parameters $\rho_1 = 10^{-4}$ and $\rho_2 = 0.9$ for Armijo–Wolfe conditions (2.29a) and (2.29b). In our first set of tests, we also compared the performance of these three algorithms against the state-of-the-art nonlinear programming software package "ipopt" (version 3.3) [42]. In our implementation, we omitted the term h_x, h_y in the objective function. All experiments were performed on a Dell Precision 670 workstation with an Intel Xeon 3.4GHZ CPU and 6GB of RAM.

In the following examples, our tests are on mappings into S^2.

Example 4.1 (see [2, 7, 37]). Let $\Omega = (-1, 1)^2$, and the initial solution $\mathbf{U}^0 : \Omega \to S^2$ for $\mathbf{x} \in \Omega$ be defined by

$$\mathbf{U}^0(\mathbf{x}) = \left(\frac{\mathbf{x}}{|\mathbf{x}|}\sin\phi(|\mathbf{x}|), \cos\phi(|\mathbf{x}|)\right), \text{ where } \phi(r) = \begin{cases} br^2 & \text{for } r \le 1, \\ b & \text{for } r \ge 1 \end{cases} \tag{4.1}$$

and $b = \frac{3\pi}{2}$. The Dirichlet boundary condition is taken as $\widetilde{\mathbf{U}}(\mathbf{x}) = (\frac{\mathbf{x}}{|\mathbf{x}|}, 0)$ on $\partial\Omega$. The grid spacing is set to $h = \sqrt{2}/2^4$. We chose two cases $p = 1$ and 2. We terminated all algorithms when the norm of the gradient was less than $\epsilon = 10^{-5}$ and limited the total number of iterations to 10000. For the "fixed-step" method, we set $\tau = 10^{-2}$ for $p = 1$ and $\tau = 5 * 10^{-4}$ for $p = 2$ to avoid a blow-up in the objective function. In Algorithm 2, the number of monotone curvilinear search iterations γ was set to 20.

We plot the first two components of $\widetilde{\mathbf{U}}(\mathbf{x})$ and the initial solution $\mathbf{U}^0(\mathbf{x})$ in Figure 2. It is easy to see that the initial solution $\mathbf{U}^0(\mathbf{x})$ is not close to the original data $\widetilde{\mathbf{U}}(\mathbf{x})$ by comparing their two quiver plots at various grid points.

First, let us consider the $p = 1$ case. A summary of the computational costs for all four methods is presented in Table 1 (please refer to the caption of this table for the detailed meaning of the statistics "CPU," "ITER," "NFE," "NGE," "NEE," "NJE," "NHE," "$E_p(\mathbf{U})$,"

Table 1

Computational summary for Example 4.1. "CPU" denotes CPU time measured in seconds (for "ipopt," the first number is CPU time in "ipopt" without the function, gradient, and Hessian evaluations, and the second number is the CPU time of these evaluations); "ITER" denotes the number of iterations; "NFE" denotes the total number of function evaluations; "NGE" denotes the total number of gradient evaluations; "NEE" denotes the total number of equality constraint evaluations; "NJE" denotes the total number of equality constraint Jacobian evaluations; "NHE" denotes the total number of Hessian evaluations; $E_p(\mathbf{U})$ and $\|\nabla E_p\|$ denote the objective function value and the norm of the gradient, respectively, when the algorithm terminates.

p	ipopt				
	CPU (sec.)	ITER	NFE/NGE/NEE/NJE/NHE	$E_p(\mathbf{U})$	$\|\nabla E_p\|$
1	6.018/4.501	128	188/129/196/129/128	7.40e+01	7.91e-05
2	5.088/4.910	158	159/159/159/159/158	1.28e+01	8.15e-06

p	fixed-step				
	CPU (sec.)	ITER	NFE	$E_p(\mathbf{U})$	$\|\nabla E_p\|$
1	8.086515	10000	10001	7.40e+01	2.79e-05
2	7.172569	10000	10001	1.79e+01	1.23e+00

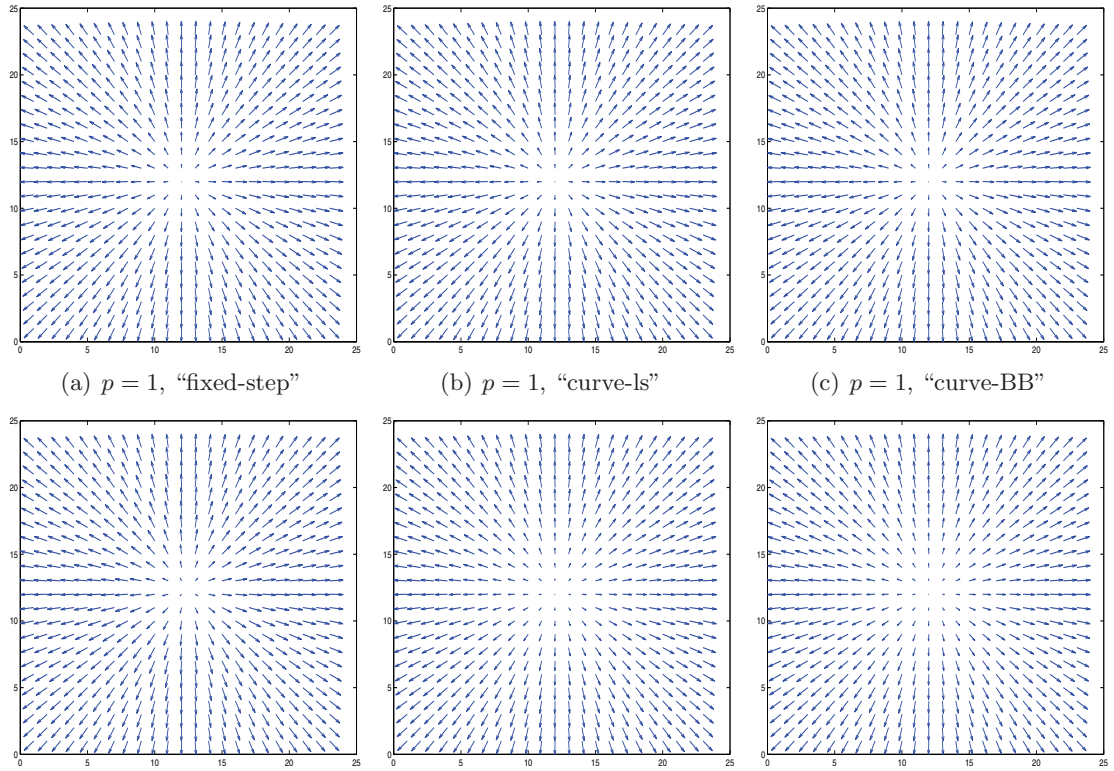
p	curve-ls				
	CPU (sec.)	ITER	NFE	$E_p(\mathbf{U})$	$\|\nabla E_p\|$
1	5.478214	3308	3998	7.40e+01	9.90e-06
2	1.763271	1085	1365	1.28e+01	9.92e-06

p	curve-BB				
	CPU (sec.)	ITER	NFE	$E_p(\mathbf{U})$	$\|\nabla E_p\|$
1	0.371542	331	334	7.40e+01	9.88e-06
2	0.177434	162	169	1.28e+01	9.73e-06

"$\|\nabla E_p\|$"). From the table, the superiority of the "curve-BB" method is obvious, especially with respect to "ipopt." The "curve-BB" method took less CPU time and fewer function evaluations to achieve a point with similar accuracy than the "curve-ls" method, which is better than the "fixed-step" method. Although "ipopt" took fewer function evaluations than the "curve-BB" method, it consumed much more CPU time (6.018 seconds without including the time to compute function, gradient, and Hessian values) since "ipopt" is a second-order type of method and the computational cost of matrix factorization is very expensive. It is worth noting that the CPU time (4.501 seconds) for function, gradient, and Hessian evaluations in "ipopt" is not surprising since we used the AMPL [23] interface of "ipopt" to compute Example 4.1 and the gradient and the Hessian are evaluated through automatic differentiation in AMPL, which is very expensive. Hence, our effort to exploit the structure of the gradient is highly rewarded.

We depict the performance of the curvilinear search methods in Figures 3(a), 3(b), 3(c), 4(a), and 4(b). From Figure 3, all of the three methods recovered the solution quite well. Figure 4(a) shows the energy-versus-iteration histories. All three methods reduced the objective value quickly at early iterations. Then they all slowed down, yielding "L"-shaped curves. However, it is obvious that "curve-BB" converges faster than "curve-ls," which converges faster than "fixed-step." Figure 4(b) shows a portion of Figure 4(a) magnified. It shows that the algorithm "curve-BB" quickly converged in fewer than 100 iterations.

The recovered solution and the energy-versus-iteration histories for $p = 2$ are given in Figures 3(d), 3(e), 3(f), 4(c), and 4(d). These figures and Table 1 demonstrate that the



(a) $p = 1$, "fixed-step" (b) $p = 1$, "curve-ls" (c) $p = 1$, "curve-BB"

(d) $p = 2$, "fixed-step" (e) $p = 2$, "curve-ls" (f) $p = 2$, "curve-BB"

Figure 3. *Recovered solution for Example 4.1.*

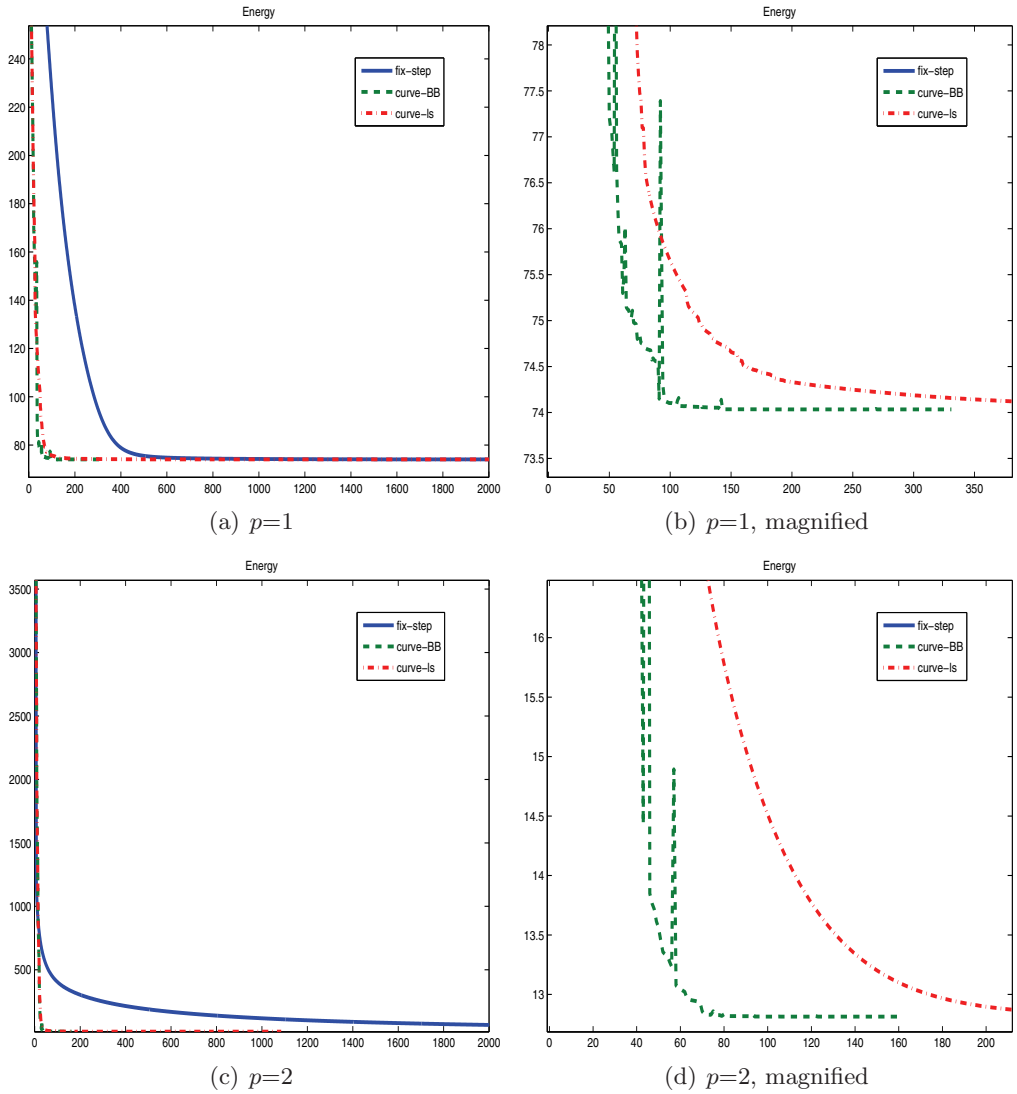
curvilinear search methods were effective and the BB step strategy significantly accelerated convergence, while the method "fixed-step" did not converge within 10000 iterations. The method "ipopt" provided a good solution, but it was far less efficient in terms of the CPU time.

Next, we consider the application of RGB color image denoising. This involves solving problem (1.1) with the Neumann boundary conditions $\frac{\partial \mathbf{U}}{\partial \vec{\mathbf{n}}}|_{\partial\Omega} = 0$, where $\vec{\mathbf{n}}$ denotes the exterior unit normal to $\partial\Omega$. Let $\mathbf{I} = (I_R, I_G, I_B) \in \mathbb{R}^3$ be an original color image, from which we extracted the intensity or brightness $|\mathbf{I}| = \sqrt{I_R^2 + I_G^2 + I_B^2}$ and the chromaticity

$$\mathbf{f} \stackrel{def}{=} \frac{\mathbf{I}}{|\mathbf{I}|} = \left(\frac{I_R}{\sqrt{I_R^2 + I_G^2 + I_B^2}}, \frac{I_G}{\sqrt{I_R^2 + I_G^2 + I_B^2}}, \frac{I_B}{\sqrt{I_R^2 + I_G^2 + I_B^2}} \right) \in S^2.$$

In our tests, noise was added to the image, but only to the chromaticity $\mathbf{f}$ so that the noisy chromaticity became $\mathbf{f}^{(0)} = \frac{\mathbf{f}+\beta\xi}{|\mathbf{f}+\beta\xi|}$, where $\xi \sim \text{Normal}(\mathbf{0}, \mathbf{1})$ and β was the noise level. We then applied Algorithms 1 and 2 and obtained an optimal restoration $\mathbf{f}^*$. Finally, using the original brightness $|\mathbf{I}|$, we assembled the new image $\mathbf{I}_{new} = \mathbf{f}^*|\mathbf{I}|$.

Example 4.2. In Figure 5, we depict three color RGB images: (a) "pepper" (resolution: 135×198), (b) "clown" (resolution: 200×320), and (c) "fabric" (resolution: 300×300)



(a) p=1 (b) p=1, magnified

(c) p=2 (d) p=2, magnified

Figure 4. *Energy versus iterations for Example 4.1.*

and their corresponding noisy versions (d), (e), and (f). The noise level β was set to 0.5. We studied two cases $p = 1$ and 2. For the fixed step size method, τ was set to 10^{-3}. Let $\xi = \sqrt{\beta}\|\mathbf{f} - \mathbf{f}^{(0)}\|_F$, the scaled error between the original image and the initial solution. We terminated the algorithm when the norm of the gradient was less than ϵ, where $\epsilon = 0.8\xi$ for $p = 1$ and $\epsilon = 0.2\sqrt{\xi}$ for $p = 2$. (The solution will be oversmoothed if ϵ is too small.) We also limited the total number of iterations to 500.

The denoised solutions of the images "pepper," "clown," and "fabric" are depicted in Figures 6, 7, and 8, respectively. They show that both the "curve-BB" method and the "curve-ls" method accurately recovered the image. Summaries of the computational performance for the three methods are presented in Tables 2, 3, and 4, respectively. We did not run "ipopt"

(a) "pepper" (original image) (b) "clown" (original image) (c) "fabric" (original image)

(d) "pepper" (initial solution) (e) "clown" (initial solution) (f) "fabric" (initial solution)

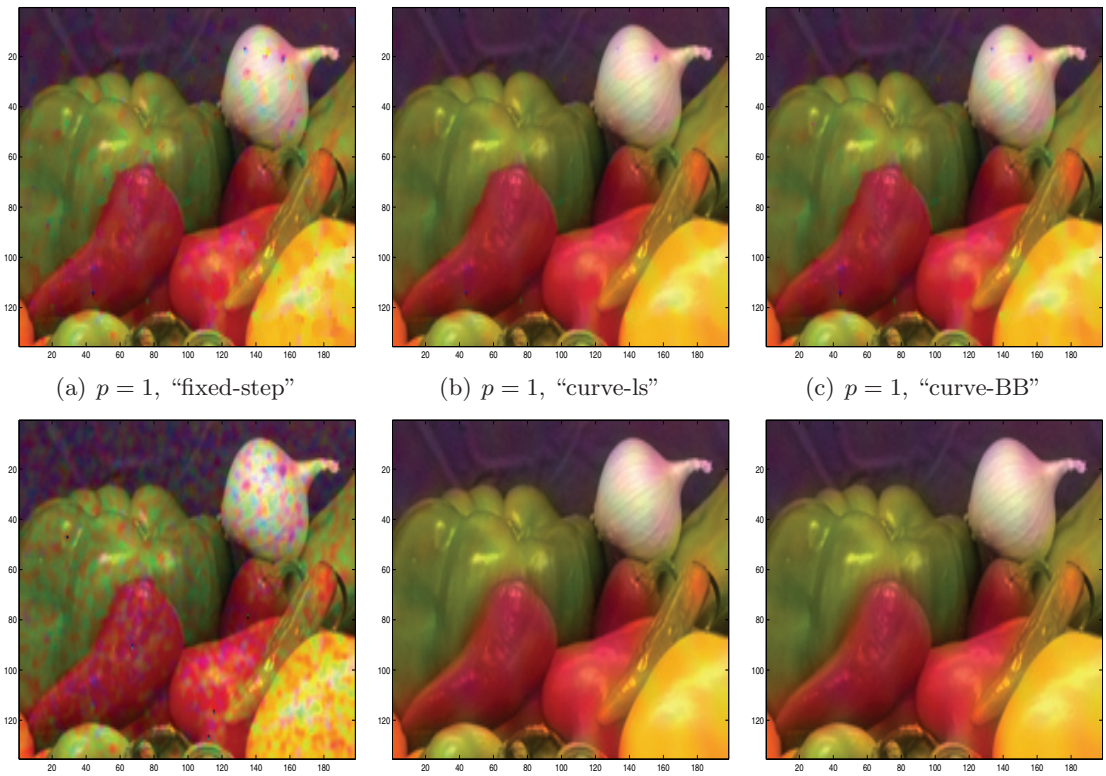
Figure 5. *Original images and initial solutions for Example 4.2.*

on these problems because it cannot solve such large problems efficiently (e.g., "fabric" has 270000 variables and 90000 nonlinear constraints).

5. Conclusion. In this paper, we present new gradient descent algorithms for the p-harmonic flow problem on spheres. The algorithms are based on a simple updating formula and a specialized finite difference scheme, which preserve the pointwise constraints $|\mathbf{U}| = 1$. One of the algorithms determines a step size by an inexact curvilinear search and is globally convergent. The other algorithm uses Barzilai–Borwein step sizes and is nonmonotonic. While not shown to converge in theory, the latter algorithm exhibits exceptional computational efficiency.

Our future work includes extending the proposed algorithmic framework to problems in higher dimensional manifolds and adapting classical optimization techniques such as Newton's method and conjugate directions within this framework.

Appendix A. Symbolic computational procedure. Since a direct verification of Theorem 2.9 is tedious, we give a symbolic computational procedure, Listing 1, written in the language of MATLAB. First, symbolic objects are declared to define the matrix $(\mathbf{H}^n_{i,j})^{(\times)}$, where a, b, c denote $(\mathbf{H}_a)_{i,j}$, $(\mathbf{H}_b)_{i,j}$, $(\mathbf{H}_c)_{i,j}$, respectively. Then "Wplus" and "Wminus" denote the matrices $\mathcal{W}^{n+}_{i,j}$ and $\mathcal{W}^{n-}_{i,j}$, respectively. Finally, "Ut" is computed as the solution of (2.3),



(a) $p = 1$, "fixed-step" (b) $p = 1$, "curve-ls" (c) $p = 1$, "curve-BB"

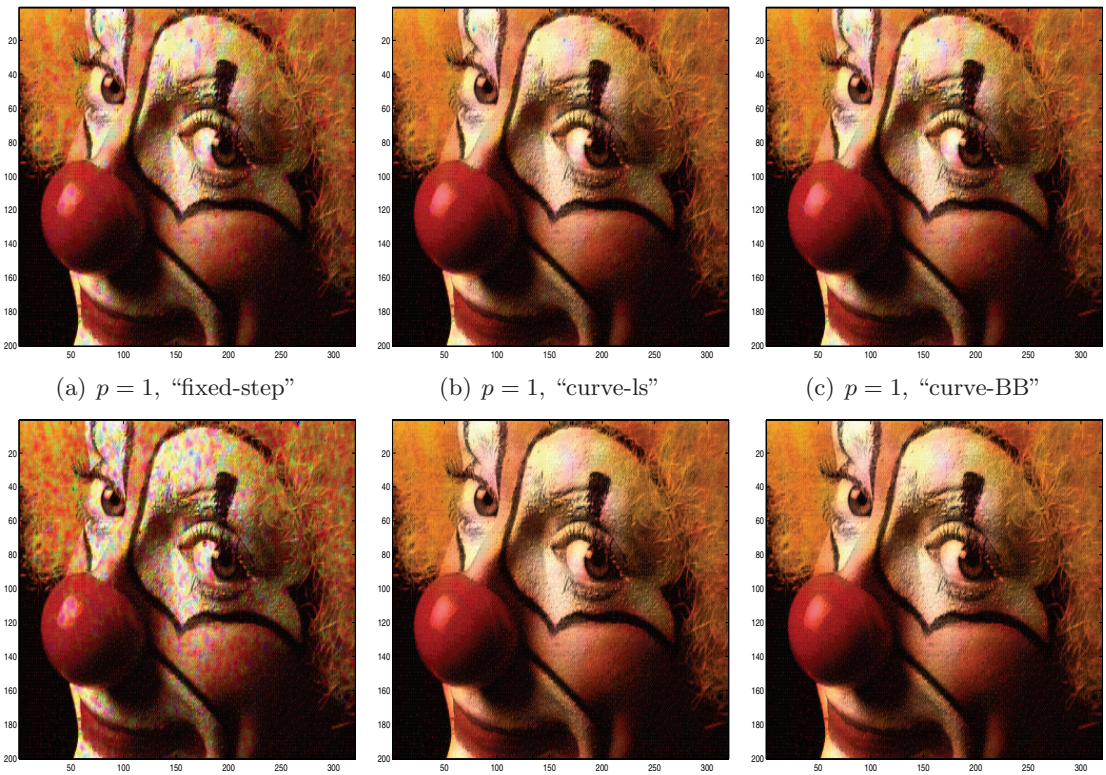
(d) $p = 2$, "fixed-step" (e) $p = 2$, "curve-ls" (f) $p = 2$, "curve-BB"

Figure 6. *Recovered solution for the image "pepper" in Example 4.2.*

Table 2

Computational summary for the image "pepper" in Example 4.2.

p	fixed-step				
	CPU (sec.)	ITER	NFE	$E_p(\mathbf{U})$	$\|\nabla E_p\|$
1	20.219459	500	501	1.52e+03	1.06e+02
2	11.688046	500	501	2.07e+07	1.34e+10
p	curve-ls				
	CPU (sec.)	ITER	NFE	$E_p(\mathbf{U})$	$\|\nabla E_p\|$
1	2.445225	55	61	7.25e+02	5.72e+01
2	1.918936	47	62	1.70e+01	1.74e+00
p	curve-BB				
	CPU (sec.)	ITER	NFE	$E_p(\mathbf{U})$	$\|\nabla E_p\|$
1	1.776940	38	42	9.02e+02	6.24e+01
2	1.556385	41	49	1.29e+01	1.44e+00



(a) $p = 1$, "fixed-step" (b) $p = 1$, "curve-ls" (c) $p = 1$, "curve-BB"

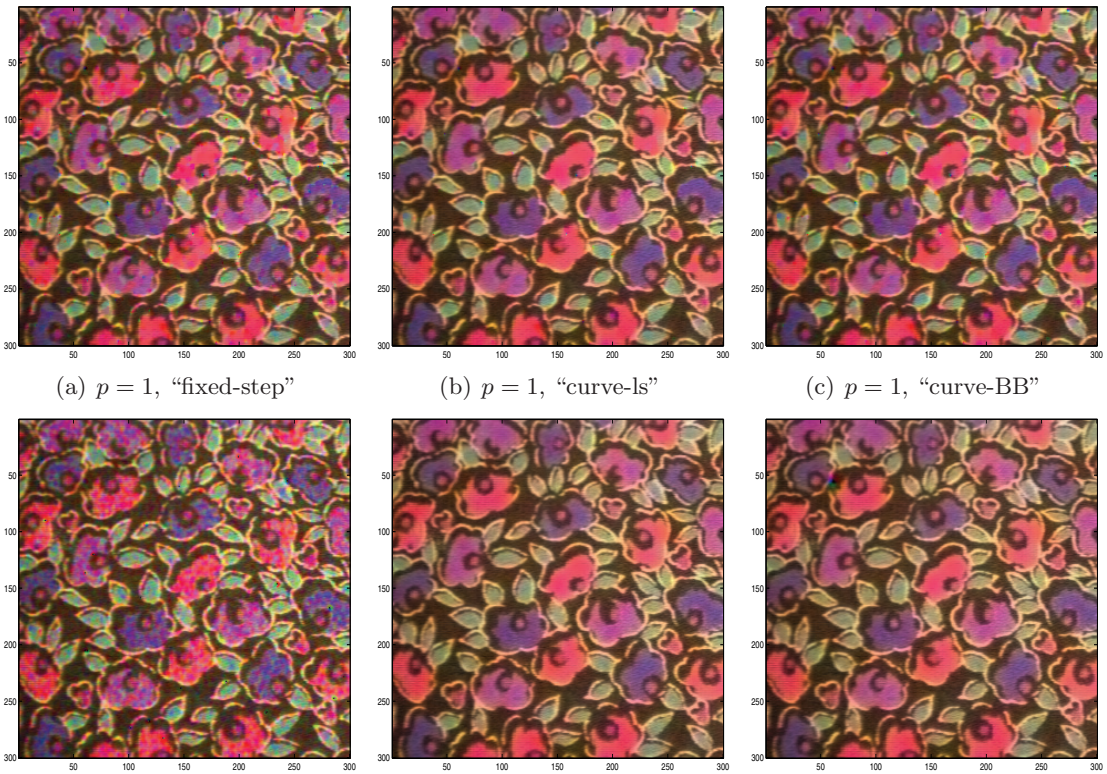
(d) $p = 2$, "fixed-step" (e) $p = 2$, "curve-ls" (f) $p = 2$, "curve-BB"

Figure 7. *Recovered solution for the image "clown" in Example 4.2.*

Table 3

Computational summary for the image "clown" in Example 4.2.

p	fixed-step				
	CPU (sec.)	ITER	NFE	$E_p(\mathbf{U})$	$\|\nabla E_p\|$
1	51.430550	500	501	3.88e+03	1.71e+02
2	30.585574	500	501	6.22e+07	2.39e+10
p	curve-ls				
	CPU (sec.)	ITER	NFE	$E_p(\mathbf{U})$	$\|\nabla E_p\|$
1	9.945352	59	70	1.51e+03	9.17e+01
2	7.227943	63	90	2.21e+01	2.17e+00
p	curve-BB				
	CPU (sec.)	ITER	NFE	$E_p(\mathbf{U})$	$\|\nabla E_p\|$
1	8.250743	50	59	1.95e+03	9.65e+01
2	5.067317	47	63	2.14e+01	2.18e+00



(a) $p = 1$, "fixed-step" (b) $p = 1$, "curve-ls" (c) $p = 1$, "curve-BB"

(d) $p = 2$, "fixed-step" (e) $p = 2$, "curve-ls" (f) $p = 2$, "curve-BB"

Figure 8. *Recovered solution for the image "fabric" in Example 4.2 with $p = 1$.*

Table 4
Computational summary for the image "fabric" in Example 4.2.

p	fixed-step				
	CPU (sec.)	ITER	NFE	$E_p(\mathbf{U})$	$\|\nabla E_p\|$
1	82.922588	500	501	5.34e+03	1.91e+02
2	64.345355	500	501	1.98e+08	6.59e+11
p	curve-ls				
	CPU (sec.)	ITER	NFE	$E_p(\mathbf{U})$	$\|\nabla E_p\|$
1	16.018488	64	70	2.15e+03	9.80e+01
2	19.517887	70	100	2.74e+01	2.20e+00
p	curve-BB				
	CPU (sec.)	ITER	NFE	$E_p(\mathbf{U})$	$\|\nabla E_p\|$
1	10.120239	40	44	3.01e+03	1.15e+02
2	18.441347	89	103	4.78e+01	2.06e+00

Listing 1
Verify Theorem 2.9.

```
% define symbolic objects
syms a b c u v w t
% define the matrix version of the cross product
H = [0 -c b; c 0 -a; -b a 0];
% define intermediate matrix
Wminus =  eye(3) - 0.5*t*H;
Wplus  =  eye(3) + 0.5*t*H;
% solve the system of linear equations
Ut = simplify(inv(Wminus)* Wplus * [u;v; w])
% compute the derivative of Ut with respect to t
simplify(diff(Ut,'t'))
% check the determinant of the matrix Wminus
det(Wminus)
% check the 2-norm of the matrix Lemma 2.11
hatW = inv(Wminus)* Wplus - eye(3);
simplify(eig(hatW.'*hatW))
tildeW = 0.5* inv(Wminus)*H*(eye(3) + inv(Wminus)* Wplus) - H;
simplify(eig(tildeW.'*tildeW))
```

where "eye" is the function that generates an identity matrix, and "inv" is the function that inverts a matrix. The function "simplify" helps us to get a simplified version of a function. We use the function "det" to check the determinant of the matrix $\mathcal{W}_{i,j}^{n-}$, so that we actually can invert it. Specifically, the command "det(Wminus)" returns $1 + 1/4 * t^2 * a^2 + 1/4 * t^2 * c^2 + 1/4 * t^2 * b^2$ which is $1 + \frac{1}{4}\tau^2\|\mathbf{H}_{i,j}^n\|^2$ since $\|\mathbf{H}_{i,j}^n\|^2 = a^2 + b^2 + c^2$. To check the 2-norm of the matrices needed in Lemma 2.12, we declare "hatW" and "tildeW" to denote the matrices $\widehat{\mathcal{W}}_{i,j}^n$ and $\widetilde{\mathcal{W}}_{i,j}^n$, respectively. Then the function "eig" is called to compute their eigenvalues. Finally, the returned results are further simplified by noting that $\|\mathbf{H}_{i,j}^n\|^2 = a^2 + b^2 + c^2$.

Appendix B. Proof of Lemma 2.7.

We now compute the gradient of the discrete energy function $E_p(\mathbf{U})$ explicitly. Since the variable $u_{i,j}$ appears only in the terms $F_{k,l}$, where (k,l) corresponds to the grid points $(i,j),(i+1,j)$, and $(i,j+1)$, the partial derivative $\frac{\partial E_p(\mathbf{U})}{\partial u_{i,j}}$ can be written as

$$\frac{\partial E_p(\mathbf{U})}{\partial u_{i,j}} = \frac{\partial F_{i,j}}{\partial u_{i,j}} + \frac{\partial F_{i+1,j}}{\partial u_{i,j}} + \frac{\partial F_{i,j+1}}{\partial u_{i,j}}, \tag{B.1}$$

where, by the chain rule of differentiation,

$$\begin{aligned}\frac{\partial F_{i,j}}{\partial u_{k,l}} = pF_{i,j}^{\frac{p-2}{p}} \Bigg(&(f_1^x)_{i,j}\frac{\partial (f_1^x)_{i,j}}{\partial u_{k,l}} + (f_2^x)_{i,j}\frac{\partial (f_2^x)_{i,j}}{\partial u_{k,l}} + (f_3^x)_{i,j}\frac{\partial (f_3^x)_{i,j}}{\partial u_{k,l}} \\ &+ (f_1^y)_{i,j}\frac{\partial (f_1^y)_{i,j}}{\partial u_{k,l}} + (f_2^y)_{i,j}\frac{\partial (f_2^y)_{i,j}}{\partial u_{k,l}} + (f_3^y)_{i,j}\frac{\partial (f_3^y)_{i,j}}{\partial u_{k,l}}\Bigg).\end{aligned}$$

Since $(f_1^x)_{i,j}$ is a function of the variables $u_{i,j}, u_{i-1,j}, v_{i,j}$, and $v_{i-1,j}$ at a particular grid point (i,j), it suffices to compute the partial derivatives of $(f_1^x)_{i,j}$ with respect to these variables;

all other components are zero. We compute $\frac{\partial (f_1^x)_{i,j}}{\partial u_{i,j}}$ explicitly as an illustration:

$$\frac{\partial (f_1^x)_{i,j}}{\partial u_{i,j}} = \frac{1}{(S_{i,j}^x)^2}\left[\left(\frac{\partial \zeta_x u_{i,j}}{\partial u_{i,j}}\delta_x v_{i,j} - \zeta_x v_{i,j}\frac{\partial \delta_x u_{i,j}}{\partial u_{i,j}}\right) S_{i,j}^x \right.$$
$$\left. - \left(\zeta_x u_{i,j}\delta_x v_{i,j} - \zeta_x v_{i,j}\delta_x u_{i,j}\right)\left(2\zeta_x u_{i,j}\frac{\partial \zeta_x u_{i,j}}{\partial u_{i,j}}\right)\right].$$

Using (2.14), we obtain

$$\begin{aligned}\left[\zeta_x u_{i,j}\delta_x v_{i,j} - \zeta_x v_{i,j}\delta_x u_{i,j}\right]\zeta_x u_{i,j} &= (\zeta_x u_{i,j})^2\delta_x v_{i,j} - \zeta_x v_{i,j}(-\zeta_x v_{i,j}\delta_x v_{i,j} - \zeta_x w_{i,j}\delta_x w_{i,j}) \\ &= \left(S_{i,j}^x\right)\delta_x v_{i,j} + \left(\zeta_x v_{i,j}\delta_x w_{i,j} - \zeta_x w_{i,j}\delta_x v_{i,j}\right)\zeta_x w_{i,j}.\end{aligned}$$

Together with $\frac{\partial \zeta_x u_{i,j}}{\partial u_{i,j}} = \frac{1}{2}$, we have

$$\frac{\partial (f_1^x)_{i,j}}{\partial u_{i,j}} = \frac{1}{(S_{i,j}^x)^2}\left[-\left(\frac{\partial \zeta_x u_{i,j}}{\partial u_{i,j}}\delta_x v_{i,j} + \zeta_x v_{i,j}\frac{\partial \delta_x u_{i,j}}{\partial u_{i,j}}\right) S_{i,j}^x \right.$$
$$\left. - \left(\zeta_x v_{i,j}\delta_x w_{i,j} - \zeta_x w_{i,j}\delta_x v_{i,j}\right)\zeta_x w_{i,j}\right].$$

It follows from the fact $\frac{\partial \zeta_x u_{i,j}}{\partial u_{i,j}}\delta_x v_{i,j} + \zeta_x v_{i,j}\frac{\partial \delta_x u_{i,j}}{\partial u_{i,j}} = \frac{1}{h_x}v_{i,j}$ that

$$\frac{\partial (f_1^x)_{i,j}}{\partial u_{i,j}} = -\frac{1}{h_x S_{i,j}^x}v_{i,j} - \frac{1}{S_{i,j}^x}(f_3^x)_{i,j}\zeta_x w_{i,j}. \tag{B.2}$$

Similarly, we obtain

$$\frac{\partial (f_2^x)_{i,j}}{\partial u_{i,j}} = -\frac{1}{h_x S_{i,j}^x}w_{i,j} + \frac{1}{S_{i,j}^x}(f_3^x)_{i,j}\zeta_x v_{i,j}. \tag{B.3}$$

Since only the term $S_{i,j}^x$ in $(f_3^x)_{i,j}$ is related to $u_{i,j}$, a direct calculation gives

$$\frac{\partial (f_3^x)_{i,j}}{\partial u_{i,j}} = -\frac{1}{S_{i,j}^x}(f_3^x)_{i,j}\zeta_x u_{i,j}. \tag{B.4}$$

Noting that $-(f_1^x)_{i,j}(f_3^x)_{i,j}\zeta_x w_{i,j} + (f_2^x)_{i,j}(f_3^x)_{i,j}\zeta_x v_{i,j} = (f_3^x)_{i,j}^2\zeta_x u_{i,j}$ and combining (B.2), (B.3), and (B.4), we obtain

$$(f_1^x)_{i,j}\frac{\partial (f_1^x)_{i,j}}{\partial u_{k,l}} + (f_2^x)_{i,j}\frac{\partial (f_2^x)_{i,j}}{\partial u_{k,l}} + (f_3^x)_{i,j}\frac{\partial (f_3^x)_{i,j}}{\partial u_{k,l}} = -\frac{1}{h_x S_{i,j}^x}\left((f_1^x)_{i,j}v_{i,j} + (f_2^x)_{i,j}w_{i,j}\right). \tag{B.5}$$

Similarly to (B.5), we obtain for the y direction that

$$(f_1^y)_{i,j}\frac{\partial (f_1^y)_{i,j}}{\partial u_{k,l}} + (f_2^y)_{i,j}\frac{\partial (f_2^y)_{i,j}}{\partial u_{k,l}} + (f_3^y)_{i,j}\frac{\partial (f_3^y)_{i,j}}{\partial u_{k,l}} = -\frac{1}{h_y S_{i,j}^y}\left((f_1^y)_{i,j}v_{i,j} + (f_2^y)_{i,j}w_{i,j}\right). \tag{B.6}$$

Finally, by symmetry of the variables and noting the relationship (B.1), we can write out the gradient explicitly as

$$\text{(B.7)}\qquad \begin{cases} \frac{\partial E_p(\mathbf{U})}{\partial u_{i,j}} = +v_{i,j}(\mathbf{H}_c)_{i,j} - w_{i,j}(\mathbf{H}_b)_{i,j}, \\ \frac{\partial E_p(\mathbf{U})}{\partial v_{i,j}} = -u_{i,j}(\mathbf{H}_c)_{i,j} + w_{i,j}(\mathbf{H}_a)_{i,j}, \\ \frac{\partial E_p(\mathbf{U})}{\partial w_{i,j}} = +u_{i,j}(\mathbf{H}_b)_{i,j} - v_{i,j}(\mathbf{H}_a)_{i,j}, \end{cases}$$

which leads to the cross-product expression in the statement of Lemma 2.7.

Acknowledgments. We would like to thank Stanley J. Osher (UCLA), Luminita A. Vese (UCLA), Sören Bartels (Rheinische Friedrich-Wilhelms-Universität, Bonn), and François Alouges (U-PSUD) for their helpful discussions and comments. We also would like to acknowledge the two anonymous referees for making helpful suggestions.

REFERENCES

[1] F. Alouges, *A new algorithm for computing liquid crystal stable configurations: The harmonic mapping case*, SIAM J. Numer. Anal., 34 (1997), pp. 1708–1726.
[2] J. W. Barrett, S. Bartels, X. Feng, and A. Prohl, *A convergent and constraint-preserving finite element method for the p-harmonic flow into spheres*, SIAM J. Numer. Anal., 45 (2007), pp. 905–927.
[3] S. Bartels, *Stability and convergence of finite-element approximation schemes for harmonic maps*, SIAM J. Numer. Anal., 43 (2005), pp. 220–238.
[4] S. Bartels and A. Prohl, *Convergence of an implicit finite element method for the Landau–Lifshitz–Gilbert equation*, SIAM J. Numer. Anal., 44 (2006), pp. 1405–1419.
[5] S. Bartels and A. Prohl, *Fully Practical, Constraint Preserving, Implicit Approximation of Harmonic Map Heat Flow into Spheres*, Technical report, Department of Mathematics, Humboldt-Universität zu Berlin, Berlin, 2006; available online from http://na.uni-tuebingen.de/preprints.shtml.
[6] S. Bartels and A. Prohl, *Stable discretization of scalar and constrained vectorial Perona-Malik equation*, Interfaces Free Bound., 9 (2007), pp. 431–453.
[7] S. Bartels and A. Prohl, *Convergence of an implicit, constraint preserving finite element discretization of p-harmonic heat flow into spheres*, Numer. Math., 109 (2008), pp. 489–507.
[8] J. Barzilai and J. M. Borwein, *Two-point step size gradient methods*, IMA J. Numer. Anal., 8 (1988), pp. 141–148.
[9] F. Bethuel, H. Brezis, and F. Hélein, *Limite singulière pour la minimisation de fonctionnelles du type Ginzburg-Landau*, C. R. Acad. Sci. Paris Sér. I Math., 314 (1992), pp. 891–895.
[10] F. Bethuel, H. Brezis, and F. Hélein, *Asymptotics for the minimization of a Ginzburg-Landau functional*, Calc. Var. Partial Differential Equations, 1 (1993), pp. 123–148.
[11] T. Cecil, S. Osher, and L. Vese, *Numerical methods for minimization problems constrained to S^1 and S^2*, J. Comput. Phys., 198 (2004), pp. 567–579.
[12] T. Chan and J. Shen, *Variational restoration of nonflat image features: Models and algorithms*, SIAM J. Appl. Math., 61 (2000), pp. 1338–1361.
[13] Y. Chen, *The weak solutions to the evolution problems of harmonic maps*, Math. Z., 201 (1989), pp. 69–74.
[14] Y. Chen and F.-H. Lin, *Remarks on approximate harmonic maps*, Comment. Math. Helv., 70 (1995), pp. 161–169.
[15] Y. Chen and M. Struwe, *Existence and partial regularity results for the heat flow for harmonic maps*, Math. Z., 201 (1989), pp. 83–103.
[16] R. Cohen, R. Hardt, D. Kinderlehrer, S. Y. Lin, and M. Luskin, *Minimum energy configurations for liquid crystals: Computational results*, in Theory and Applications of Liquid Crystals (Minneapolis, MN, 1985), IMA Vol. Math. Appl. 5, Springer, New York, 1987, pp. 99–121.
[17] R. Cohen, S. Y. Lin, and M. Luskin, *Relaxation and gradient methods for molecular orientation in liquid crystals*, Comput. Phys. Comm., 53 (1989), pp. 455–465.

[18] J.-M. Coron and R. Gulliver, *Minimizing p-harmonic maps into spheres*, J. Reine Angew. Math., 401 (1989), pp. 82–100.
[19] Y.-H. Dai, *Alternate step gradient method. Theory, methods and applications of optimization*, Optimization, 52 (2003), pp. 395–415.
[20] Y.-H. Dai and R. Fletcher, *Projected Barzilai-Borwein methods for large-scale box-constrained quadratic programming*, Numer. Math., 100 (2005), pp. 21–47.
[21] Y.-H. Dai and L.-Z. Liao, **R**-*linear convergence of the Barzilai and Borwein gradient method*, IMA J. Numer. Anal., 22 (2002), pp. 1–10.
[22] R. Fletcher, *Practical Methods of Optimization*, 2nd ed., A Wiley-Interscience Publication, John Wiley & Sons, Chichester, UK, 1987.
[23] R. Fourer, D. M. Gay, and B. W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*, 2nd ed., Duxbury Press/Brooks/Cole Publishing (Cengage Learning), Florence, KY, 2002.
[24] N. Hungerbühler, *Global weak solutions of the p-harmonic flow into homogeneous spaces*, Indiana Univ. Math. J., 45 (1996), pp. 275–288.
[25] N. Hungerbühler, *Non-uniqueness for the p-harmonic flow*, Canad. Math. Bull., 40 (1997), pp. 174–182.
[26] N. Hungerbühler, *Heat flow into spheres for a class of energies*, in Variational Problems in Riemannian Geometry, Progr. Nonlinear Differential Equations Appl. 59, Birkhäuser, Basel, 2004, pp. 45–65.
[27] M. Kružík and A. Prohl, *Recent developments in the modeling, analysis, and numerics of ferromagnetism*, SIAM Rev., 48 (2006), pp. 439–483.
[28] F.-H. Lin, *A remark on the map $x/|x|$*, C. R. Acad. Sci. Paris Sér. I Math., 305 (1987), pp. 529–531.
[29] S.-Y. Lin and M. Luskin, *Relaxation methods for liquid crystal problems*, SIAM J. Numer. Anal., 26 (1989), pp. 1310–1324.
[30] M. Lysaker, S. Osher, and X.-C. Tai, *Noise removal using smoothed normals and surface fitting*, IEEE Trans. Image Process., 13 (2004), pp. 1345–1357.
[31] F. Mémoli, G. Sapiro, and S. Osher, *Solving variational problems and partial differential equations mapping into general target manifolds*, J. Comput. Phys., 195 (2004), pp. 263–292.
[32] M. Misawa, *Approximation of p-harmonic maps by the penalized equation*, Nonlinear Anal., 47 (2001), pp. 1069–1080.
[33] J. J. Moré and D. J. Thuente, *Line search algorithms with guaranteed sufficient decrease*, ACM Trans. Math. Software, 20 (1994), pp. 286–307.
[34] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed., Springer Ser. Oper. Res. Fin. Engrg., Springer, New York, 2006.
[35] P. Perona, *Orientation diffusion*, IEEE Trans. Image Process., 79 (1998), pp. 12–49.
[36] M. Raydan, *The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem*, SIAM J. Optim., 7 (1997), pp. 26–33.
[37] M. Struwe, *Geometric evolution problems*, in Nonlinear Partial Differential Equations in Differential Geometry (Park City, UT, 1992), IAS/Park City Math. Ser. 2, Nonlinear Programming, 1996, pp. 257–339.
[38] W. Sun and Y.-X. Yuan, *Optimization theory and methods*, Springer Optimization and Its Applications 1, Springer, New York, 2006.
[39] B. Tang, G. Sapiro, and V. Caselles, *Diffusion of general data on non-flat manifolds via harmonic maps theory: The direction diffusion case*, Int. J. Comput. Vis., 36 (2000), pp. 149–161.
[40] B. Tang, G. Sapiro, and V. Caselles, *Color image enhancement via chromaticity diffusion*, IEEE Trans. Image Process., 10 (2001), pp. 701–707.
[41] L. A. Vese and S. J. Osher, *Numerical methods for p-harmonic flows and applications to image processing*, SIAM J. Numer. Anal., 40 (2002), pp. 2085–2104.
[42] A. Wächter and L. T. Biegler, *On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming*, Math. Program., 106 (2006), pp. 25–57.