Available online at www.sciencedirect.com
ScienceDirect
Journal of Computational Physics 220 (2006) 6–18
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

Short note

Fast geodesics computation with the phase flow method

Lexing Ying *, Emmanuel J. Candès

Applied and Computational Mathematics, California Institute of Technology, Caltech, MC 217-50, Pasadena, CA 91125, United States

Received 7 February 2006; received in revised form 9 May 2006; accepted 31 July 2006
Available online 12 September 2006

Abstract

This paper introduces a novel approach for rapidly computing a very large number of geodesics on a smooth surface. The idea is to apply the recently developed phase flow method [L. Ying, E.J. Candès, The phase flow method, J. Comput. Phys., to appear], an efficient and accurate technique for constructing phase maps for nonlinear ordinary differential equations on invariant manifolds, which are here the unit tangent bundles of the surfaces under study. We show how to rapidly construct the whole geodesic flow map which then allows computing any geodesic by straightforward local interpolation, an operation with constant complexity. A few numerical experiments complement our study and demonstrate the effectiveness of our approach.

Keywords: Geodesics; Weighted geodesics; Geodesic flow; The phase flow method; Manifolds; Tangent bundles; Charts; Surface parameterization; Spline interpolation

1. Introduction

1.1. The problem

This paper introduces a new method for rapidly computing the geodesic flow on smooth and compact surfaces. Suppose that Q is a smooth surface, then the geodesics of Q obey certain types of differential equations, which may take the form

$$\frac{\mathrm{d}\boldsymbol{y}}{\mathrm{d}t} = F(\boldsymbol{y}), \quad \boldsymbol{y} = \boldsymbol{y}_0; \tag{1.1}$$

$\boldsymbol{y} := (\boldsymbol{x}, \boldsymbol{p})$ where $\boldsymbol{x}$ is a running point on the surface Q, and $\boldsymbol{p}$ is a point in the tangent space so that $\boldsymbol{p}(t)$ is the vector tangent to the geodesic at $\boldsymbol{x}(t)$. Standard methods for solving such ordinary differential equations (ODEs) are based on local ODE integration rules such as the various Runge–Kutta methods. Typically, one chooses a small step size τ and makes repeated use of the local integration rule. If one wishes to integrate the equations up to time

* Corresponding author. Tel.: +1 626 395 5760.
E-mail address: lexing@acm.caltech.edu (L. Ying).

0021-9991/$ - see front matter
doi:10.1016/j.jcp.2006.07.032

T, the accuracy is generally of the order of τ^α – α is called the order of the local integration rule – and the computational complexity is of the order of T/τ, i.e. proportional to the total number of steps.

In many problems of interest, one would like to trace a large number of geodesics. Expressed differently, one would like to integrate the system (1.1) for many different initial conditions. Examples arising from geometric modeling and computational geometry include mesh parameterization, the segmentation of a surface into several components, shape classification [6], and the interpolation of functions defined on surfaces [18]. The range of applications are of course not limited to computer graphics and spans many areas of science and engineering – especially computational physics and computational mechanics. Consider an arbitrary dynamical system with holonomic constraints. Then it is often the case that the particle trajectories are the geodesics of the smooth manifold defined by these constraints. In the theory of general relativity, the trajectory of a free particle is the spatial projection of a geodesic traced on the curved four-dimensional space–time manifold. In order to understand the underlying dynamics of these physical problems, one often calculates a large number of trajectories which are nothing else than geodesics.

An interesting application which involves computing a large number of geodesics and will be studied in this paper comes from the field of high-frequency wave propagation. Suppose that a smooth body is "illuminated" by an incoming planar wave with an arbitrary direction of propagation. We wish to compute the scattered wavefield. Now the geometric theory of diffraction [8] asserts that straight diffraction rays are emitted from the so-called *creeping rays*. A creeping ray is a geodesic curve on the scatterer which starts from a point on the shadow line and whose initial tangent is parallel to the orientation of the incoming planar wave. To compute the scattered field then, one needs to trace as many geodesics as there are points on the various shadow lines, see Section 2.4 for more details.

In many of these problems, standard methods tracing geodesic curves one by one may be computationally very expensive, and in this paper we introduce a fast and accurate method for computing the whole geodesic flow map over the surface. Our strategy is built upon the phase flow method [19], a newly established method for solving system (1.1) which we review next.

1.2. The phase flow method

Suppose that we are given the system of ordinary differential equation (1.1), where the vector field $\boldsymbol{F}$: $\mathbf{R}^d \to \mathbf{R}^d$ is assumed to be smooth. For a fixed time t, the map g_t: $\mathbf{R}^d \to \mathbf{R}^d$ defined by $g_t(\boldsymbol{y}_0) = \boldsymbol{y}(t, \boldsymbol{y}_0)$ is called the *phase map*, and the family $\{g_t,\ t \in \mathbf{R}\}$ of all phase maps – which forms a one parameter family of diffeomorphisms – is called the *phase flow*. A manifold $M \subset \mathbf{R}^d$ is said to be *invariant* if $g_t(M) \subset M$. In many situations, we are interested in the restriction of the phase flow on an invariant manifold.

We wish to compute the solutions $\boldsymbol{y}(T, \boldsymbol{y}_0)$ of the system (1.1) with many initial conditions $\boldsymbol{y}_0$. Rather than integrating the system one ray at a time, we integrate (1.1) for all the initial conditions at once. The approach consists of two steps:

- First, construct an approximation $\tilde{g}_T$ to the phase map g_T at time T.
- Second, for each $\boldsymbol{y}_0$, the solution $\boldsymbol{y}(T, \boldsymbol{y}_0)$ is calculated by simply evaluating $\tilde{g}_T(\boldsymbol{y}_0)$.

The main difficulty is in the construction of $\tilde{g}_T$. Specifically, we need (1) to construct $\tilde{g}_T$ efficiently and accurately and (2) to represent $\tilde{g}_T$ in a way allowing fast evaluation, which is equally important. This is exactly what the phase flow method achieves.

Algorithm 1 (The phase flow method [19]).

1. *Parameter selection*. Select a grid size $h > 0$, a time step $\tau > 0$, and an integer constant $S \geqslant 1$ such that $B = (T/\tau)^{1/S}$ is an integer power of 2.
2. *Discretization*. Select a uniform or quasi-uniform grid $M_h \subset M$ of size h.
3. *Burn-in*. Compute $\tilde{g}_\tau$. For $\boldsymbol{y}_0 \in M_h$, $\tilde{g}_\tau(\boldsymbol{y}_0)$ is calculated by applying the ODE integrator (single time step of length τ). Then construct a local interpolant based on these sampled values, and for $\boldsymbol{y}_0 \notin M_h$, define $\tilde{g}_\tau(\boldsymbol{y}_0)$ by evaluating the interpolant at $\boldsymbol{y}_0$.

4. *Loop*. For $k = 1, \ldots, S$, evaluate $\tilde{g}_{B^k\tau}$. For $\boldsymbol{y}_0 \in M_h$, $\tilde{g}_{B^k\tau}(\boldsymbol{y}_0) = (\tilde{g}_{B^{k-1}\tau})^{(B)}(\boldsymbol{y}_0)$ where $f^{(2)} = f \circ f$, $f^{(3)} = f \circ f \circ f$ and so on. Construct a local interpolant based on these sampled values, and for $\boldsymbol{y}_0 \notin M_h$, define $\tilde{g}_{B^k\tau}(\boldsymbol{y}_0)$ by evaluating the interpolant at $\boldsymbol{y}_0$.
5. *Terminate*. The algorithm terminates at $k = S$ since by definition $B^S\tau = T$ and hence $\tilde{g}_T = \tilde{g}_{B^S\tau}$. The approximate solution $\tilde{\boldsymbol{y}}(T, \boldsymbol{y}_0)$ is equal to $\tilde{g}_T(\boldsymbol{y}_0)$.

The method relies on three components which are application dependent, namely, the ODE integration rule, the local interpolation scheme, and the selection of the discrete grid M_h, all of which will be fully specified in concrete examples a little later.

The phase flow essentially exploits two important points. The first is the continuous dependence of the solution at time T on the initial condition, i.e. for each t, $g_t(\boldsymbol{y}_0)$ is a smooth function of $\boldsymbol{y}_0$. This enables the accurate approximation of g_T from its values on the grid M_h. The second is the group structure of the phase flow $\{g_t, t \in \mathbf{R}\}$, $g_{t+t'} = g_t \circ g_{t'}$, which holds since (1.1) is an autonomous system. The group property allows a systematic reuse of earlier computations – much like the repeated squaring argument which computes large powers of a matrix. The obvious advantage is that one can of course construct large-time phase maps with just a few iterations.

In practice, when the time T is large, g_T may become quite oscillatory while remaining smooth. The version below is usually more efficient and practical for large times.

Algorithm 2 (*The phase flow method*: *modified version* [19]).

1. Choose $T_0 = \mathrm{O}(1)$ such that g_{T_0} remains non-oscillatory and pick h so that the grid is sufficiently dense to approximate g_{T_0} accurately. Assume that $T = mT_0$, where m is an integer.
2. Construct $\tilde{g}_{T_0}$ using Algorithm 1.
3. For any $\boldsymbol{y}_0$, define $\tilde{g}_T(\boldsymbol{y}_0)$ by $\tilde{g}_T(\boldsymbol{y}_0) = (\tilde{g}_{T_0})^{(m)}(\boldsymbol{y}_0)$.

The main theoretical result in [19] is that the phase flow method is provably accurate.

Theorem 1 (cf. [19]). *Suppose that the ODE integrator is of order α and that the local interpolation scheme is of order $\beta \geqslant 2$ for sufficiently smooth functions. We shall also assume that the linear interpolation rule has h-independent L^∞ norm on continuous functions. Define the approximation error at time t by*

$$\varepsilon_t = \max_{\boldsymbol{b} \in M} |g_t(\boldsymbol{b}) - \tilde{g}_t(\boldsymbol{b})|. \tag{1.2}$$

Algorithms 1 and 2 enjoy the following properties:

(i) *The approximation error obeys*

$$\varepsilon_T \leqslant C \cdot (\tau^\alpha + h^\beta) \tag{1.3}$$

for some positive constant $C > 0$.

(ii) *The complexity is $\mathrm{O}(\tau^{-1/S} \cdot h^{-d(M)})$ where $d(M)$ is the dimension of M.*

(iii) *For each $y \in M$, $\tilde{g}_T(\boldsymbol{y})$ can be computed in $\mathrm{O}(1)$ operations.*

(iv) *For any intermediate time $t = m\tau \leqslant T$ where m is an integer, one can evaluate $\tilde{g}_t(\boldsymbol{y})$ for each $\boldsymbol{y} \in M$ in $\mathrm{O}(\log(1/\tau))$ operations.*

1.3. General strategy

To make things concrete, we suppose that Q is a two-dimensional orientable surface embedded in $\mathbf{R}^3$ although everything extends to low dimensional surfaces in n dimensions. Suppose the smooth surface Q is parameterized by an atlas $\{(Q_\alpha, \phi_\alpha) : \alpha \in I\}$ (a family of charts) where the collection of open sets Q_α covers Q, and each ϕ_α maps Q_α into $\mathbf{R}^2$. Given two vector fields X and Y defined in a neighborhood of a point $q \in Q$, introduce the *covariant derivative* $\nabla_X Y$ at q defined by

$$\nabla_X Y|_q = D_X Y|_q - \langle D_X Y|_q, \boldsymbol{n}\rangle \boldsymbol{n}, \tag{1.4}$$

where $D_X Y$ is the directional derivative of Y in the direction of X and $\boldsymbol{n}$ is the unit vector normal to the surface at the point q. The covariant derivative is the tangential component of the directional derivative and lies in the tangent plane of Q at q. A curve $\gamma(t)$ is called a *geodesic* on Q if $\nabla_{\gamma'}\gamma' = 0$ for all t in the domain of definition. Before we move on, we would like to bring up an important point: although the definition of the covariant derivative requires X and Y to be defined in a neighborhood of q, γ' is only defined along the curve $\gamma(t)$. However, we can justify the definition by smoothly extending γ' in a neighborhood of the curve $\gamma(t)$ in an arbitrary fashion, and proving that the quantity $\nabla_{\gamma'}\gamma'$ does not depend upon this extension.

In terms of local coordinates x^i, $i = 1, 2$, in each chart, a geodesic is a solution of the system of nonlinear ordinary differential equations

$$\frac{\mathrm{d}^2 x^i}{\mathrm{d}t^2} + \sum_{1\leqslant j,k\leqslant 2} \Gamma^i_{jk} \frac{\mathrm{d}x^j}{\mathrm{d}t}\frac{\mathrm{d}x^k}{\mathrm{d}t} = 0, \quad i \in \{1, 2\}, \tag{1.5}$$

where Γ^i_{jk} is the so-called Christoffel symbol. The geodesic equations (1.5) are "extrinsic" in the sense that they depend upon the choice of the local coordinate system. We point the reader to [10] (Chapter 4) for their derivation and for the definition of the Christoffel symbol.

The second-order geodesic equations may also be formulated as a first order ODE system defined on the tangent bundle TQ. The phase flow defined by this first order ODE system is often called the *geodesic flow*. An obvious invariant manifold is the whole tangent bundle TQ. However, since the geodesic flow preserves the length of a tangent vector, the unit tangent bundle T^1Q, which contains all the unit-normed tangent vectors is a manifold of smaller dimension and is also invariant. (The behavior of non-unitary tangent vectors can be obtained by rescaling the time variable t.) All of this is detailed in Section 2, where we will choose to work with an "intrinsic" first order ODE system which is conceptually simpler and which we will derive explicitly.

We then follow Algorithm 2 to construct the geodesic flow map g_{T_0} on the invariant manifold T^1Q where T_0 is O(1). The discretization of the invariant manifold $M = T^1Q$ and the local interpolation scheme are both defined with the help of the atlas $\{(Q_\alpha, \phi_\alpha)\}$ so that the interpolation grid is a standard Cartesian grid. The details of the construction are given in Section 2.2. Once g_{T_0} is available, the computation of any geodesic curve is obtained by repeatedly applying g_{T_0} – with each application having O(1) computational complexity.

1.4. Related work and contributions

As emphasized earlier, geodesic computations arise in many applications and various solutions have been proposed in the literature. A frequently discussed approach consists in representing the surface with a piecewise linear triangle mesh. The problem of computing geodesics is now reduced to that of tracing straight paths on a triangle mesh. Several algorithms [3,13,17] have been developed to date by the computational geometry community in this setup. Another popular solution regards the geodesic distance as the viscosity solution of the eikonal equation defined on the surface. This observation allows the extension of the fast marching method [16] to triangle meshes, see [9]. An extension is described in [12] where the idea is to approximate the geodesic distance on the surface with the Euclidean distance computed in a "band" around the surface, which enables the use of fast marching method on Cartesian grids.

Our approach is markedly different from these earlier contributions in several ways. The first difference concerns the surface representation. Although the piecewise linear triangle mesh is a powerful tool to represent surfaces, it only provides a second-order approximation of the surface under study (it may just be used to get a first-order approximation of the tangent plane for example). However, geodesic computations are governed by the curvature of the surface, and straight paths on triangle meshes are often poor approximations of the real geodesics. In contrast, our algorithm works directly on smooth surfaces instead and produces geodesic curves which are provably accurate. The second difference concerns the output of the computation. There are often more than one geodesics connecting two points on a smooth surface. In most of the aforementioned papers, the algorithm returns the geodesic with the shortest distance. Our

approach can return *all* the geodesics which have length less than T. This is essential in some important applications, which include the problem of creeping rays we will detail in the next section. The third and last difference concerns precomputations. Most of the existing techniques precompute the geodesic (distance) information from a single source point. This information obviously needs to be recomputed when one is interested in geodesics starting from a different source. Instead, we compute the whole phase map, which includes all the information for geodesics with arbitrary initial points and arbitrary initial orientations.

The recent work by Motamed and Runborg [14] on computing creeping rays in high-frequency scattering is similar in spirit to our approach on these three points. However, while our approach based on the phase flow method computes the geodesic map at a fixed time T, their Eulerian-type algorithm extends a different method in [5] to efficiently calculate an "exit" function which involves the phase maps at different times.

2. Fast geodesic flow computations

This section develops our approach for computing the geodesic flow on T^1Q, and we will assume that the smooth compact surface $Q \subset \mathbf{R}^3$ is the zero level set of a smooth function F: $\mathbf{R}^3 \to \mathbf{R}$, i.e. $Q = \{\boldsymbol{x}: F(\boldsymbol{x}) = 0\}$. This is a useful assumption for getting simple equations but as we have pointed out earlier, the method does not depend upon this assumption (we just need to be able to derive the equations of the geodesic flow).

2.1. The geodesic flow equations

Letting $T_{\boldsymbol{x}}Q$ be the tangent space of Q at a point $\boldsymbol{x}$, the geodesic curves obey the differential equations below.

Theorem 2. *Suppose $\boldsymbol{x}_0 \in Q$, $\boldsymbol{p}_0 \in T_{\boldsymbol{x}_0}Q$ and $|\boldsymbol{p}_0| = 1$. The geodesic with initial point $\boldsymbol{x}_0$ and tangent $\boldsymbol{p}_0$ is the integral curve of the system*

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \boldsymbol{p}, \quad \frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t} = -\frac{\langle \boldsymbol{p}, \nabla^2 F \boldsymbol{p}\rangle}{|\nabla F|^2}\nabla F, \tag{2.1}$$

and with initial conditions $\boldsymbol{x}(0) = \boldsymbol{x}_0$, $\boldsymbol{p}(0) = \boldsymbol{p}_0$.

Proof. Let us first check that, for each $t > 0$, the following three conditions hold: (i) $\mathrm{d}\boldsymbol{p}/\mathrm{d}t$ is parallel to ∇F; (ii) $\boldsymbol{p}$ is in the tangent space, i.e. $\langle \boldsymbol{p}, \nabla F\rangle = 0$; and (iii) $|\boldsymbol{p}| = 1$. The first condition follows from the second equation in (2.1). For (ii), note that the time derivative of $\langle \boldsymbol{p}, \nabla F\rangle$ obeys

$$\frac{\mathrm{d}}{\mathrm{d}t}\langle \boldsymbol{p}, \nabla F\rangle = \left\langle \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{p}, \nabla F \right\rangle + \left\langle \boldsymbol{p}, \nabla^2 F \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} \right\rangle = -\frac{\langle \boldsymbol{p}, \nabla^2 F \boldsymbol{p}\rangle}{|\nabla F|^2}\langle \nabla F, \nabla F\rangle + \langle \boldsymbol{p}, \nabla^2 F \boldsymbol{p}\rangle = 0.$$

At $t = 0$, $\langle \boldsymbol{p}(0), \nabla F(\boldsymbol{x}(0))\rangle = 0$ since $\boldsymbol{p}_0 \in T_{\boldsymbol{x}_0}Q$, which implies $\langle \boldsymbol{p}, \nabla F\rangle = 0$ for each $t > 0$. Finally, we compute

$$\frac{\mathrm{d}}{\mathrm{d}t}\langle \boldsymbol{p}, \boldsymbol{p}\rangle = 2\left\langle \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{p}, \boldsymbol{p} \right\rangle = -2\frac{\langle \boldsymbol{p}, \nabla^2 F \boldsymbol{p}\rangle}{|\nabla F|^2}\langle \nabla F, \boldsymbol{p}\rangle = 0,$$

where we have used (ii) in the last step. It follows from $|\boldsymbol{p}_0|^2 = 1$ that $\boldsymbol{p}$ obeys (iii) for all t's.

It follows from (ii) that

$$\frac{\mathrm{d}}{\mathrm{d}t}F(\boldsymbol{x}(t)) = \langle \boldsymbol{p}, \nabla F\rangle = 0.$$

Since $F(\boldsymbol{x}(0)) = 0$, $F(\boldsymbol{x}(t))$ is equal to zero for all t, which shows that the curve $\boldsymbol{x}(t)$ belongs to Q at all times. The third condition $|\boldsymbol{p}| = 1$ implies that $D_{\boldsymbol{p}}\boldsymbol{p} = \mathrm{d}\boldsymbol{p}/\mathrm{d}t$. It follows from the definition of a geodesic in Section 1.3 that we need to show $\nabla_{\boldsymbol{p}}\boldsymbol{p} = 0$ in order to show that $\boldsymbol{x}(t)$ is a geodesic curve. Since $\mathrm{d}\boldsymbol{p}/\mathrm{d}t$ is parallel to the normal direction ∇F and $\nabla_{\boldsymbol{p}}\boldsymbol{p}$ is the tangent component of $D_{\boldsymbol{p}}\boldsymbol{p} = \mathrm{d}\boldsymbol{p}/\mathrm{d}t$ then, by definition, $\nabla_{\boldsymbol{p}}\boldsymbol{p}$ is equal to zero. The proof is complete. □

It is a well known fact that the shortest path between two points on a smooth manifold is a geodesic (up to reparameterizations). Given a positive function $c(\boldsymbol{x})$ on Q, one can also define the *weighted geodesic* between two points $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$ as the curve on the surface Q minimizing $\int_{x_0}^{x_1} 1/c(\boldsymbol{x})\,\mathrm{d}s$ where s is the arclength parameterization.

Corollary 1. *Suppose* $\boldsymbol{x}_0 \in Q$, $\boldsymbol{p}_0 \in T_{\boldsymbol{x}_0}Q$ *and* $|\boldsymbol{p}_0| = 1$. *The weighted geodesic with initial value* $(\boldsymbol{x}_0, \boldsymbol{p}_0)$ *is the integral curve of*

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = c(\boldsymbol{x})\frac{\boldsymbol{p}}{|\boldsymbol{p}|}, \quad \frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t} = -(\nabla c - \langle \nabla c, \boldsymbol{n}\rangle \boldsymbol{n})|\boldsymbol{p}| - c\frac{\langle \boldsymbol{p}, \nabla^2 F \boldsymbol{p}\rangle}{|\nabla F|^2}\frac{\nabla F}{|\boldsymbol{p}|}, \tag{2.2}$$

where $\boldsymbol{n}(\boldsymbol{x}) = \frac{\nabla F(\boldsymbol{x})}{|\nabla F(\boldsymbol{x})|}$ *is the surface normal at* $\boldsymbol{x}$.

We briefly sketch why this is correct. Fermat's principle [2] asserts that a weighted geodesic curve is an integral curve of the Hamiltonian equations generated by the Hamiltonian $H(\boldsymbol{x}, \boldsymbol{p}) = c(\boldsymbol{x})|\boldsymbol{p}|$ with the additional constraint that $\boldsymbol{x} \in Q$. It then follows from the d'Alembert principle [1] that the Hamiltonian equations of $H(\boldsymbol{x}, \boldsymbol{p})$ are of the form

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = c(\boldsymbol{x})\frac{\boldsymbol{p}}{|\boldsymbol{p}|}, \quad \frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t} = -\nabla c(\boldsymbol{x})|\boldsymbol{p}| + k\boldsymbol{n}, \tag{2.3}$$

where k is a scalar function. The relationship $\langle \boldsymbol{p}, \nabla F\rangle = 0$ is then used to derive a formula for k. We have

$$\frac{\mathrm{d}}{\mathrm{d}t}\langle \boldsymbol{p}, \nabla F\rangle = \langle -\nabla c|\boldsymbol{p}| + k\boldsymbol{n}, \nabla F\rangle + \langle \boldsymbol{p}, \nabla^2 F \boldsymbol{p}\rangle \frac{c}{|p|} = 0,$$

and rearranging the terms, we obtain

$$k = (\nabla c \cdot \boldsymbol{n})|\boldsymbol{p}| - \frac{\langle \boldsymbol{p}, \nabla^2 F \boldsymbol{p}\rangle}{|\nabla F|}\frac{c}{|\boldsymbol{p}|}.$$

Substitution into (2.3) gives (2.2) as claimed. Note that with $\boldsymbol{u} = \boldsymbol{p}/|\boldsymbol{p}|$, we can also use the reduced Hamiltonian system

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = c(\boldsymbol{x})\boldsymbol{u}, \quad \frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} = -\nabla c + \langle \nabla c, \boldsymbol{n}\rangle \boldsymbol{n} + \langle \nabla c, \boldsymbol{u}\rangle \boldsymbol{u} - c\frac{\boldsymbol{u}\nabla^2 F \boldsymbol{u}}{|\nabla F|}\boldsymbol{n}. \tag{2.4}$$

This observation is useful because it effectively reduces the dimension of the invariant manifold.

2.2. Discretization, parameterization, and interpolation

We now apply the phase flow method (Algorithms 1 and 2) to construct the phase map g_T of the geodesic flow defined by (2.1) and (2.4). We work with the invariant, compact and smooth manifold $M = T^1Q$, where T^1Q is the *unit tangent bundle* of Q (the smoothness is inherited from that of Q):

$$T^1Q = \{(\boldsymbol{x}, \boldsymbol{p}) : \boldsymbol{x} \in Q, \langle \boldsymbol{p}, \nabla F(\boldsymbol{x})\rangle = 0, |\boldsymbol{p}| = 1\} \subset \mathbf{R}^6.$$

Note that M is a three-dimensional manifold.

As remarked earlier, we need to specify the discretization of M, the (local) interpolation scheme, and the ODE integration rule.

Discretization. We suppose Q is parameterized by an atlas $\{(Q_\alpha, \phi_\alpha)\}$ (a family of charts) where the collection of open sets Q_α covers Q, and each ϕ_α maps Q_α into $\mathbf{R}^2$. Assume without loss of generality that $\phi_\alpha(Q_\alpha) = (-\delta, 1+\delta)^2$ for some fixed constant $\delta > 0$, and that $Q = \bigcup_\alpha \phi_\alpha^{-1}([0,1]^2)$ (the convenience of this assumption will be clear when we will discuss the interpolation procedure). Our atlas induces a natural parameterization of M: put $M_\alpha := T^1Q_\alpha$ for short, and for each α, define Φ_α: $M_\alpha \to (-\delta, 1+\delta)^2 \times \mathbf{S}^1$ by

$$\Phi_\alpha(\boldsymbol{x}, \boldsymbol{p}) = \left(\phi_\alpha(\boldsymbol{x}), \frac{T\phi_\alpha(\boldsymbol{x}) \cdot \boldsymbol{p}}{|T\phi_\alpha(\boldsymbol{x}) \cdot \boldsymbol{p}|}\right), \quad \boldsymbol{x} \in Q_\alpha, \boldsymbol{p} \in T^1_{\boldsymbol{x}}Q_\alpha.$$

Since ϕ_α is one to one and $\boldsymbol{p}$ is non-degenerate, the map Φ_α is always one to one and smooth. $T\phi_\alpha(\boldsymbol{x})$, the tangent map of ϕ_α at $\boldsymbol{x}$, is a linear mapping from the tangent space $T_{\boldsymbol{x}}Q_\alpha$ into $\mathbf{R}^2$, which may be defined as follows: let $\gamma(t)$ be a curve on the surface Q_α with $\gamma(0) = \boldsymbol{x}$; then $\gamma'(0) \in T_{\boldsymbol{x}}Q_\alpha$ (and conversely, every tangent vector is the derivative of a curve) and we set $T\phi_\alpha(\boldsymbol{x}) \cdot \gamma'(0) = (\phi_\alpha \circ \gamma)'(0)$.

With this in place, we now introduce a Cartesian grid G_h on $(-\delta, 1+\delta)^2 \times \mathbf{S}^1$ with grid spacing $h \leqslant \delta/2$. For each α, the grid G_h may be lifted onto M_α by the inverse of Φ_α, and our discretization grid M_h is simply the set $\bigcup_\alpha \Phi_\alpha^{-1} G_h$.

Interpolation. Suppose that $f: M \to M$ is a smooth function – for us, the phase map. We wish to construct an interpolant from the values of f on the grid M_h. The key point here is that we shall actually construct the interpolant in the parameter space $(-\delta, 1+\delta)^2 \times \mathbf{S}^1$. Introduce $f_\alpha: M_\alpha \to M$, the restriction of f to M_α. Because Φ_α is a smooth map,

$$u_\alpha := f_\alpha \Phi_\alpha^{-1} : (-\delta, 1+\delta)^2 \times \mathbf{S}^1 \to M \subset \mathbf{R}^6$$

is a smooth map between Euclidean domains, see Fig. 1b. There is a one-to-one correspondence between the values of f on M_h and those of u_α on G_h, and so all we need is to construct, for each α, an interpolant for a smooth (although non-periodic) function defined on a Cartesian grid. This is accomplished by interpolation via tensor product cardinal B-splines. The endpoint conditions are specified as follows: first, since u_α is periodic with respect to $\mathbf{S}^1$, the periodicity condition is invoked in this variable; second, u_α is in general non-periodic with respect to $(0,1)^2$, and we use the not-a-knot condition [4] for these variables. We note that the construction of the spline interpolant requires inverting sparse matrices with a small diagonal band, an operation which has linear computational complexity in terms of the number of the grid points. From now on, we denote by $\tilde{u}_\alpha$ the interpolant constructed as above.

Suppose now that we want to evaluate the value of f at a point $\boldsymbol{m} = (\boldsymbol{x}, \boldsymbol{p}) \in M$. We need to specify which $\tilde{u}_\alpha$ to use since $\boldsymbol{x}$ may be 'covered' by multiple charts Q_α. Although each $\tilde{u}_\alpha$ with $\boldsymbol{x} \in Q_\alpha$ will produce close results since each interpolant is constructed from the same values of f on the grid M_h, a careful selection of the chart α will nevertheless dramatically improve the accuracy. Our selection is guided by the following important observation: when the not-a-knot condition is used, the interpolation error at points which are at least two grid-points away from the boundary is considerably lower than that at points which are closer to the boundary. This is where our assumption becomes handy. Since $Q = \bigcup_\alpha \phi_\alpha^{-1}([0,1]^2)$ and $h \leqslant \delta/2$, we are guaranteed that for any $\boldsymbol{x} \in Q$, one can choose $\alpha(\boldsymbol{x})$ such that $\phi_{\alpha(\boldsymbol{x})}(\boldsymbol{x})$ is at least 'two grid points away' from the boundary. The value of $f(\boldsymbol{x})$ is then approximated in an obvious fashion, namely, by $\tilde{u}_{\alpha(\boldsymbol{x})}(\Phi_{\alpha(\boldsymbol{x})}(\boldsymbol{x}, \boldsymbol{p}))$.

ODE integration rule. We work with the 4th order Runge–Kutta method [7] as a local integration rule. Now, even though any integral curve of (2.1) remains on the surface Q the numerical solution will surely deviate from the surface because of the integration error. In order to ensure that the approximate phase map $g_t(\cdot)$ maps $M = T^1 Q$ to itself, we impose an extra projection step which snaps a point $(\boldsymbol{x}_0, \boldsymbol{p}_0) \in \mathbf{R}^6$ close to M back onto M. We could for instance find the point $\boldsymbol{x} \in Q$ which is closest to $\boldsymbol{x}_0$, i.e. the solution to

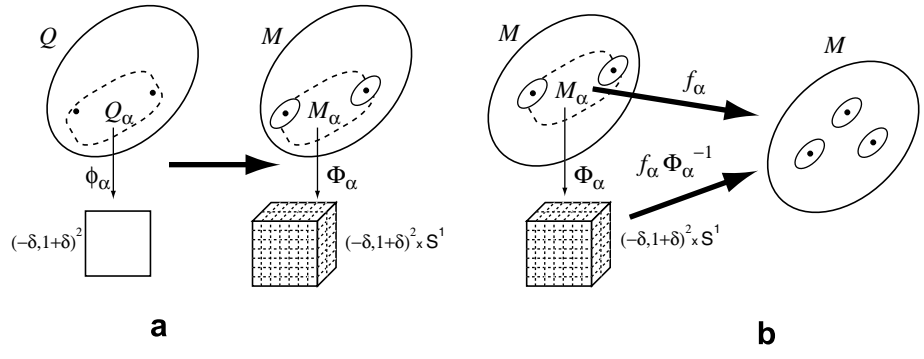

Fig. 1. (a) Schematic representation of the parameterization of the surface Q and of its unit tangent bundle $T^1 Q$. The discrete grid on M_α is obtained by lifting the lattice $G_h \subset (-\delta, 1+\delta)^2 \times \mathbf{S}^1$. (b) Construction of the local interpolant.

$$\min_{\{x:F(x)=0\}} |x - x_0|^2.$$

Standard iterative techniques for this are discussed in [15] (x_0 can naturally serve as the initial guess). Because x_0 is always very close to M, we find that the following simple algorithm works quite well: using $x = x_0$ as an initial guess, we apply the following two operations iteratively:

$$x \leftarrow x - \frac{F(x)}{|\nabla F(x)|} n(x),$$
$$x \leftarrow x_0 - \langle x_0 - x, n(x)\rangle n(x)$$

with $n(x) = \frac{\nabla F(x)}{|\nabla F(x)|}$. The first operation moves x closer to Q while the second one aligns $x - x_0$ with $n(x)$. We stop iterating when x obeys the two conditions below:

$$\frac{|F(x)|}{|\nabla F(x)|} \leqslant \varepsilon \quad \text{and} \quad \left|\left\langle \frac{x_0 - x}{|x_0 - x|}, n(x)\right\rangle\right| \leqslant \varepsilon,$$

where ε is a prescribed accuracy parameter. In practice, we choose ε to be comparable to the interpolation error, namely of size $O(h^\beta \tau)$. This allows us to assimilate the projection error with the interpolation error, which implies that the error analysis of the phase flow method remains valid as is. Since (x,p) is always close to M, it usually takes only 3–4 iterations even when ε is as small as 10^{-10}. In a second step, we then project p onto the plane orthogonal to $\nabla F(x)$ (and apply renormalization to keep a unit-length vector). This projection step is also invoked during the interpolation wherever the interpolant deviates from the invariant manifold M.

2.3. Numerical results

This section presents several numerical results. The proposed method is implemented in Matlab and all the computational results reported here were obtained on a desktop computer with a 2.6 GHz CPU and 1 GB of memory. It is expected that a careful implementation in C or Fortran would typically offer a significant improvement in terms of computation time and, therefore, we focus on the speedup over the standard ray tracing methods. We use Algorithm 2 (the modified version of the phase flow method) to construct the geodesic flow. In every example, we set $T_0 = 1/8$ and τ is chosen to be 2^{-10}. We use tensor product cubic splines to interpolate the phase map $g_t(\cdot)$. The accuracy parameter ε in the projection step is set to be 10^{-10}. To estimate the numerical error, we proceed as follows: we select N points $\{m_i\}$ randomly from M; the "exact" solutions

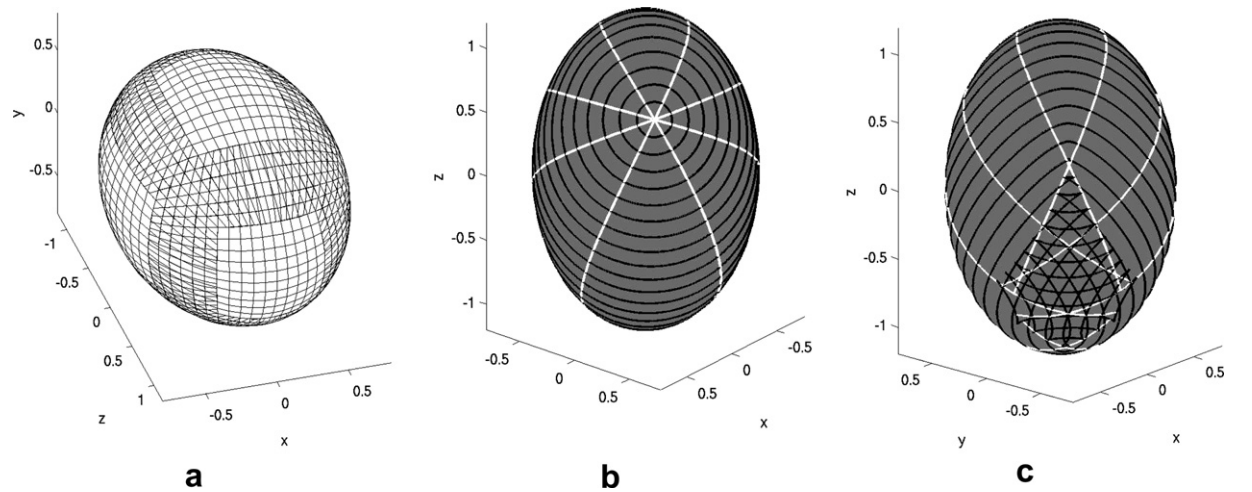

Fig. 2. Example 1: (a) discretization grid on Q, (b) and (c) geodesic flow; the white curves are the geodesics while the black curves are the points at a fixed distance from the initial point. Note the fish-tail pattern in (c).

$g_{T_0}(\boldsymbol{m}_i)$ are computed with Matlab's adaptive ODE solver with a prescribed error equal to 10^{-9}; the numerical error is estimated by $\sqrt{\sum_{i=1}^{N}|g_{T_0}(\boldsymbol{m}_i)-\tilde{g}_{T_0}(\boldsymbol{m}_i)|^2/N}$. In all these examples, $N=200$.

Example 1. The surface Q is an ellipsoid given by

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}=1$$

with $a=1.2$ and $b=c=0.8$. The atlas consists of six charts, each one corresponding to one of the six principal axes: $\pm x$, $\pm y$ and $\pm z$.

In this example, the parameter domain $(-\delta,1+\delta)$ of each chart is discretized with a Cartesian grid of size 16×16, see Fig. 2a for a plot of the discretization grid on Q. We discretize $\mathbf{S}^1$, which parameterizes the unit tangent directions, with 64 equispaced points. Constructing $\tilde{g}_{T_0}$ takes about 100 s and the error is around 3×10^{-6}.

We then use $\tilde{g}_{T_0}$ to rapidly compute the geodesics starting from an arbitrary point on the surface until $T=3.125$, see Fig. 2b and c. The black curves are solutions at time kT_0 for $k=1,2,\ldots,25$. Each one is resolved with 1024 samples each. The computation of these geodesics take 0.8 s, offering a speedup factor of 50 over standard adaptive ODE solvers.

Table 1 reports the relationship between the discretization size, the time T_0 and the numerical error of the approximate phase map. As suspected, the numerical error increases when T_0 increases and as the mesh size gets coarser. Note that for a fixed mesh size, the error increases about linearly.

Table 1
The error of the approximate phase map for different choices of discretization grids and T_0

Discretization vs. T_0	0.03125	0.0625	0.125	0.25
(8, 32)	3.088e − 05	4.917e − 05	1.538e − 04	2.385e − 04
(16, 64)	8.473e − 07	1.271e − 06	2.860e − 06	6.195e − 06
(32, 128)	8.993e − 08	9.848e − 08	2.597e − 07	3.295e − 07
(48, 192)	1.421e − 08	2.452e − 08	7.980e − 08	1.390e − 07

The rows indicate the number of gridpoints used to sample the invariant manifold $M=T^1Q$, i.e. (16, 64) means that 16 points are used for each dimension of the chart domain and 64 points for the unit tangent direction. Each column corresponds to a different value of T_0.

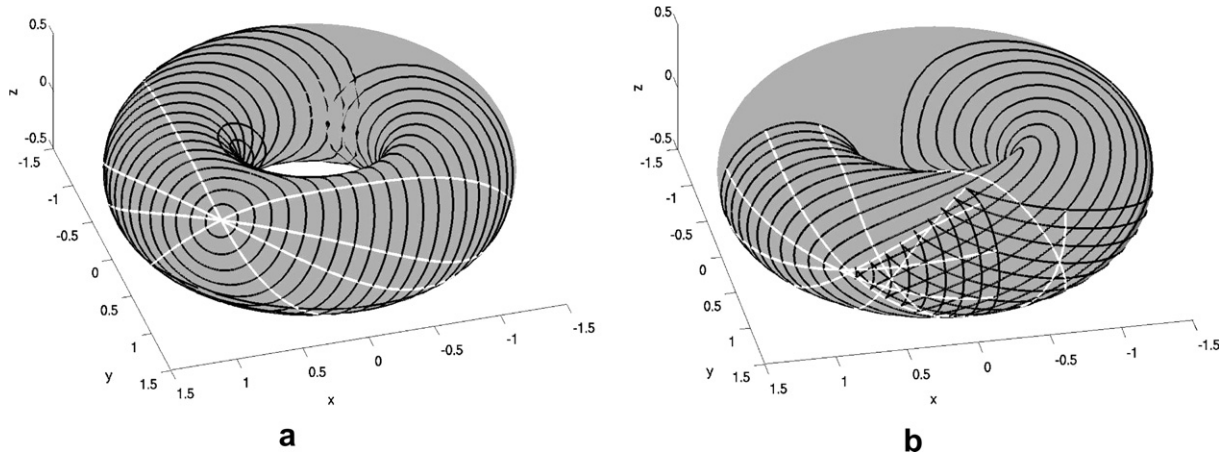

Fig. 3. Example 2. Two different families of geodesics. (a) Geodesics starting from a single point. (b) Geodesics starting from the minor circle of the torus.

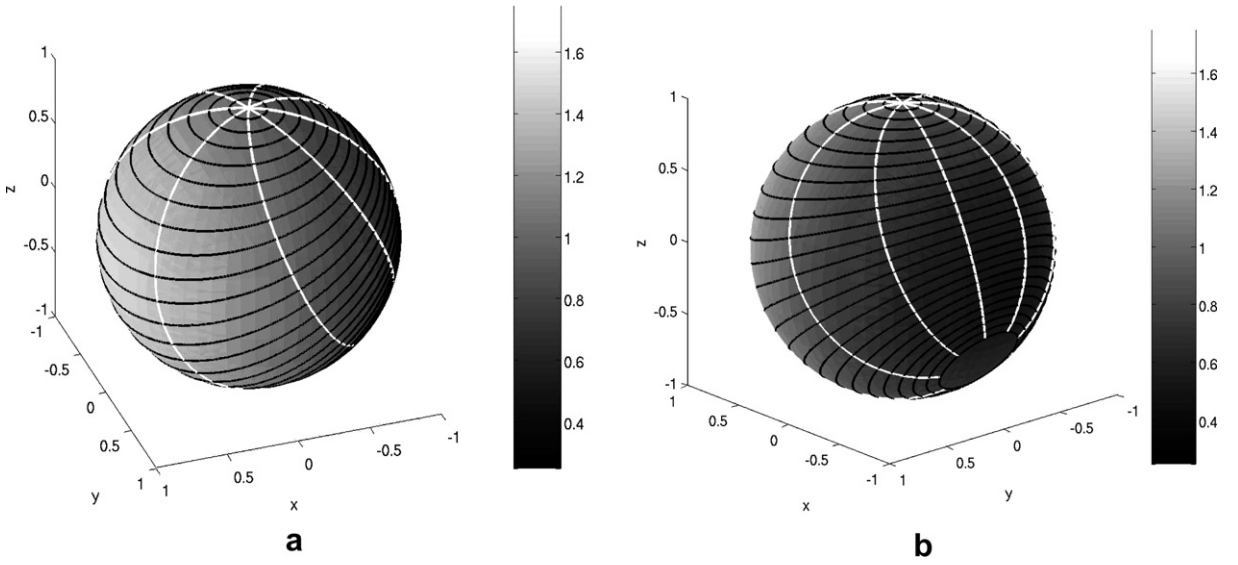

Fig. 4. Example 3. Weighted geodesics starting from the north pole. The velocity is here increasing with $\boldsymbol{x}$ since $c(x,y,z) = 1 + x$.

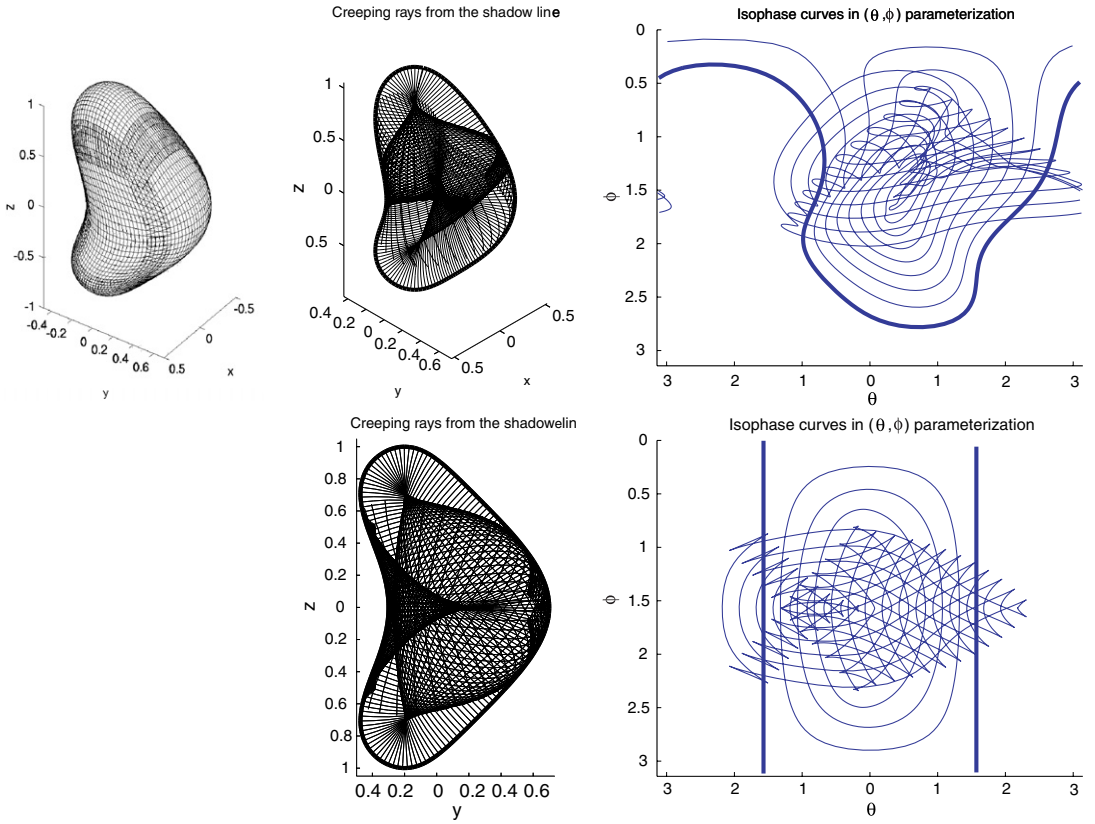

Fig. 5. Example 4. Creeping rays on a "twisted" ellipsoid corresponding to two incident directions: $(1,1,1)$ (first row) and $(1,0,0)$ (second row). The left plot shows the discretization grid on Q. The middle column shows the creeping rays on the surface of the scatterer. All the rays are generated from the shadow line (bold curve). The right column shows the iso-phase curves in the parametric domain. The bold curve is the shadow line.

Example 2. The surface is here a torus obeying the equation:

$$\left(\sqrt{x^2+y^2}-1\right)^2+z^2=0.5^2.$$

The atlas consists of a single chart with a parameterization given by

$$x=1+0.5\cos(\theta)\cos(\psi),\quad y=1+0.5\cos(\theta)\sin(\psi),\quad z=0.5\sin(\theta),$$

with $(\theta,\psi)\in[0,2\pi)^2$. We then use a grid of size 32×64 to discretize (θ,ψ) while the unit tangent direction (parameterized by $\mathbf{S}^1$) is sampled with 64 evenly distributed samples. The construction of the geodesic flow map up to T_0 takes about 30 s, and the accuracy is about 2×10^{-6}. Fig. 3 displays two different families of geodesics.

Example 3. In this example, we compute the weighted geodesics on a unit sphere. We choose the velocity field as $c(x,y,z)=1+x$. The parameterization and discretization used here are the same as those in Example 1. Fig. 4 shows the computed geodesics starting from the north pole of the sphere.

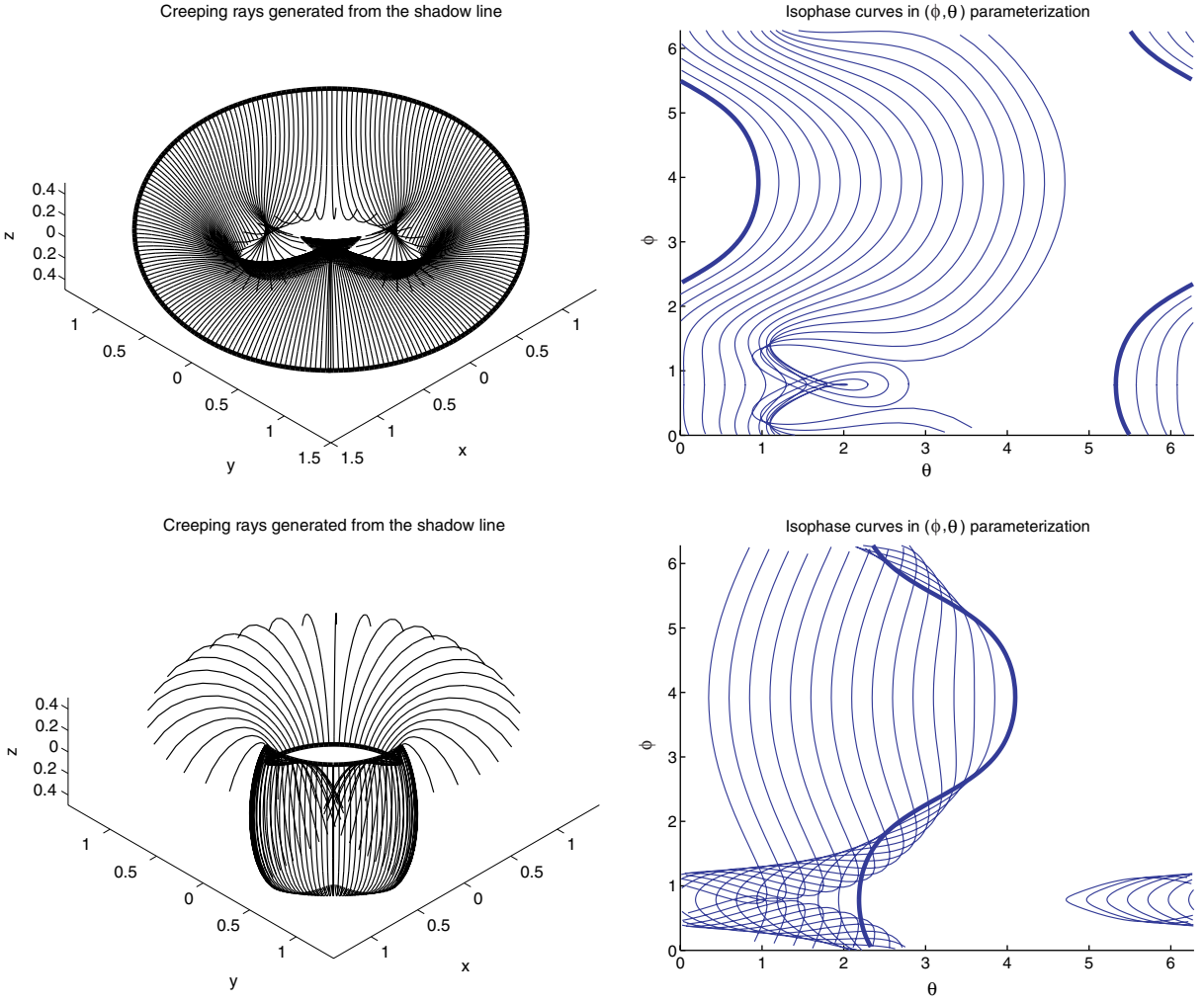

Fig. 6. Example 5. Creeping rays on a torus corresponding to the incident direction $(1,1,1)$. The shadow line (bold) is composed of two disjoint curves. Each row is associated with a single shadow curve. The left figures show the creeping rays. The right figures show the isophase curves in the (θ,ϕ) parametric domain (the bold curve is the shadow line).

2.4. Creeping rays

We finally apply our method to compute creeping rays on smooth scatterers and give two numerical examples.

Example 4. The scatterer is here a "twisted" ellipsoid given by

$$\frac{x^2}{a^2}+\frac{(y-d/c\cdot\cos(\pi z))^2}{b^2}+\frac{z^2}{c^2}=1,$$

where $a=b=1/2$, $c=1$ and $d=0.2$. The atlas here is essentially the same as that used in Example 1. For each chart, the parameter space $(-\delta,1+\delta)^2$ is discretized with a 24×24 grid, while the unit circle $\mathbf{S}^1$ is sampled with 96 equispaced points. The construction of the geodesic flow map up to time $T_0=1/8$ takes about 280 s. The computed map $\tilde{g}_{T_0}$ has accuracy about 10^{-5}.

Fig. 5 plots the results for two different incident directions. The middle column shows the creeping rays on the surface of the scatterer, while the right column displays several iso-phase curves, which simply are those points (taken from different creeping rays) at a fixed travel time away from the shadow line. These iso-phase curves are plotted with respect to the standard (θ,ϕ) (polar) coordinates used for parameterizing genus-0 surfaces.

Example 5. In this example, the scatterer is the torus used in Example 2. We plot the creeping rays associated with the incident direction $(1,1,1)$. Because this surface is nonconvex and has genus 1, the shadow line has two disconnected components. Fig. 6 plots the creeping rays starting from these two components separately.

3. Conclusion

We have shown how to use the phase flow method for computing geodesic flows on smooth and compact surfaces. In applications where one needs to trace many geodesics, our method is considerably superior than standard methods in terms of computational efficiency. One such application is the problem of computing creeping rays for which our method is especially well suited. We also demonstrated that the entire approach is numerically highly accurate.

We have presented the method in detail when the surface under study is the level set of a smooth function, and made clear that it extends to general setups. All we need is a parameterization of the surface and differential equations governing the dynamics of the geodesic flow. What is perhaps less clear is whether one could extend our approach to handle surfaces represented by triangle meshes or point clouds. Consider a triangle mesh for example. We could of course interpolate the mesh and trace geodesics on the smooth interpolated surface. If one insists, however, on tracing geodesics on the triangle mesh, the essential step towards extending our ideas would be the design of accurate local interpolation schemes which are as precise as possible on piecewise smooth objects [11].

Acknowledgments

E.C. is partially supported by a Department of Energy Grant DE-FG03-02ER25529 and by a National Science Foundation Grant CCF-0515362. L.Y. is supported by that same DOE grant and by a National Science Foundation Grant ITR ACI-0204932. The authors thank anonymous referees for their helpful comments and references.

References

[1] V.I. Arnold, Mathematical Methods of Classical Mechanics, Springer, New York, 1978, Transl. from the Russian by K. Vogtmann and A. Weinstein, Graduate Texts in Mathematics, vol. 60.
[2] M. Born, E. Wolf, Principles of Optics, seventh ed., Cambridge University Press, Cambridge, 1999.
[3] J. Chen, Y. Han, Shortest paths on a polyhedron. I. Computing shortest paths, Int. J. Comput. Geom. Appl. 6 (2) (1996) 127–144.
[4] C. de Boor, A Practical Guide to Splines, revised ed.Applied Mathematical Sciences, vol. 27, Springer, New York, 2001.

[5] S. Fomel, J.A. Sethian, Fast-phase space computation of multiple arrivals, Proc. Natl. Acad. Sci. USA 99 (11) (2002) 7329–7334 (electronic).
[6] M. Hilaga, Y. Shinagawa, T. Kohmura, T.L. Kunii, Topology matching for fully automatic similarity estimation of 3d shapes, in: SIGGRAPH'01: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, ACM Press, New York, NY, USA, 2001, pp. 203–212.
[7] A. Iserles, A First Course in the Numerical Analysis of Differential EquationsCambridge Texts in Applied Mathematics, Cambridge University Press, Cambridge, 1996.
[8] J.B. Keller, R.M. Lewis, Asymptotic methods for partial differential equations: the reduced wave equation and Maxwell's equations, in: Surveys in Applied MathematicsSurveys Appl. Math., vol. 1, Plenum Press, New York, 1995, pp. 1–82.
[9] R. Kimmel, J.A. Sethian, Computing geodesic paths on manifolds, Proc. Natl. Acad. Sci. USA 95 (15) (1998) 8431–8435 (electronic).
[10] W. Kühnel, Differential Geometry, American Mathematical Society, Providence, RI, 2002.
[11] J.-L. Mallet, Geomodeling, Oxford University Press, New York, NY, USA, 2002.
[12] F. Mémoli, G. Sapiro, Fast computation of weighted distance functions and geodesics on implicit hyper-surfaces, J. Comput. Phys. 173 (2) (2001) 730–764.
[13] J.S.B. Mitchell, D.M. Mount, C.H. Papadimitriou, The discrete geodesic problem, SIAM J. Comput. 16 (4) (1987) 647–668.
[14] M. Motamed, O. Runborg, A fast phase space method for computing creeping rays, J. Comput. Phys (to appear).
[15] J. Nocedal, S.J. Wright, Numerical OptimizationSpringer Series in Operations Research, Springer, New York, 1999.
[16] J.A. Sethian, A fast marching level set method for monotonically advancing fronts, Proc. Natl. Acad. Sci. USA 93 (4) (1996) 1591–1595.
[17] V. Surazhsky, T. Surazhsky, D. Kirsanov, S.J. Gortler, H. Hoppe, Fast exact and approximate geodesics on meshes, ACM Trans. Graph. 24 (3) (2005) 553–560.
[18] H. Wendland, Scattered Data ApproximationCambridge Monographs on Applied and Computational Mathematics, vol. 17, Cambridge University Press, Cambridge, 2005.
[19] L. Ying, E.J. Candès, The phase flow method, J. Comput. Phys. (to appear).