Fast Geodesics Computation with the Phase Flow Method

Lexing Ying and Emmanuel J. Candès

Applied and Computational Mathematics
California Institute of Technology, Pasadena, CA 91125

January 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 [15], 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.

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.

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 about 1, 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 $1/\tau$, i.e. proportional to the total number of steps.

In many problems of interest, e.g. arising in geometric modeling or mesh generation, one would like to trace a large number of geodesics. We give a concrete example in high-frequency wave propagation. Suppose that a smooth body is "illuminated" by an incoming planar wave or by a superposition of such planar waves with possibly many different directions of propagation. We wish to compute the scattered wavefield. Now the geometric theory of diffraction [7] 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.

Standard methods tracing geodesic curves one by one may be computationally very expensive in such setups 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 [15], a newly established method for solving ODEs which we review next.

1.2 The phase flow method

Suppose that we are given the system of ordinary differential equations (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, which is what the phase flow method computes. The algorithm for computing an accurate approximation of $g_t(\cdot)$ up to time T is as follows:

Algorithm 1. [The phase flow method [15]]

1. *Parameter selection.* Select a grid size $h > 0$, a time step $\tau > 0$, and an integer constant $S \geq 1$ 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 an 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 an 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. In practice and for large times, 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 [15]]

1. Choose $T_0 = O(1)$ 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 [15] is that the phase flow method is provably accurate.

Theorem 1 ([15]). *Suppose that the ODE integrator is of order α and that the local interpolation scheme is of order $\beta \geq 2$ 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*

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

for some positive constant $C > 0$.

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

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

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

1.3 General strategy

To make things concrete, we suppose that Q is a two-dimensional surface embedded in $\mathbf{R}^3$ although everything extends to intermediate dimensional surfaces in n dimensions. Suppose the smooth surface 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 $\mathbf{R}^2$. A geodesic is a path on the surface Q which minimizes the arclength between any pair of points. 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

$$\frac{\mathrm{d}^2 x^i}{\mathrm{d}t^2} + \sum_{1 \leq j,k \leq 2} \Gamma^i_{jk} \frac{\mathrm{d}x^j}{\mathrm{d}t} \frac{\mathrm{d}x^k}{\mathrm{d}t} = 0, \quad i \in \{1, 2\}, \tag{1.4}$$

where Γ^i_{jk} is the so-called Christoffel symbol. (The geodesic equations (1.4) are "extrinsic" in the sense that they depend upon the choice of the local coordinate system. The reader may want to refer to [9] (Section 7.5) 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

Our approach is markedly different from earlier contributions on this subject. Indeed, nearly all previous work has focused on the problem of computing geodesic distances from a *single* source point on the surface. Several algorithms [3, 11, 14] have been developed to date by the computational geometry community for the case where the surface is represented by a triangle mesh. Other approaches may regard 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 [13] to triangle meshes, see [8]. Finally and although this is a different problem, we mention related work in [10] where the idea is to approximate the geodesic distance on surface with the Euclidean distance computed in a "band" around the surface, which enables the use of dynamic programming on Cartesian grids—Dijkstra's algorithm essentially.

In short, the literature on fast computation of geodesics is fairly limited. We are aware of recent work by Motamed and Runborg [12], however, extending the method in [5] to efficiently compute creeping rays arising in high-frequency scattering.

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 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*

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

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

Proof. A possible way to derive the geodesic equations would be to apply the variational principle for the Lagrangian, and then invoke the Lagrange multiplier theorem to derive the Euler-Lagrange equations, see Section 8.3 in [9]. Here we choose a more direct method and only sketch the argument. To show that the integral curve of (2.1) is a geodesic, it is sufficient to 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

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

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

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

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. □

Given a positive function $c(\boldsymbol{x})$, 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

$$\int_{\boldsymbol{x}_0}^{\boldsymbol{x}_1} \frac{\mathrm{d}s}{c(\boldsymbol{x})},$$

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*

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

where $\boldsymbol{n}(\boldsymbol{x}) = \frac{\nabla F}{|\nabla F|}$ 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

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

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

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

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

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

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 $\Phi_\alpha : M_\alpha \to (-\delta, 1+\delta)^2 \times \mathbf{S}^1$ by

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

Since ϕ_α is one to one and $\boldsymbol{p}$ is non-degenerate, the map Φ_α is always one to one and smooth. For completeness, the object $T\phi_\alpha(\boldsymbol{x})$ is the linear tangent map of ϕ_α at $\boldsymbol{x}$. For the nonspecialist,

$T\phi_\alpha(\boldsymbol{x})$ is a linear mapping from the tangent space T_xQ_α 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 $\gamma'(0) \in T_xQ_\alpha$ (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 $h \leq \delta/2$. 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 $\bigcup_\alpha \Phi_\alpha^{-1} G_h$.

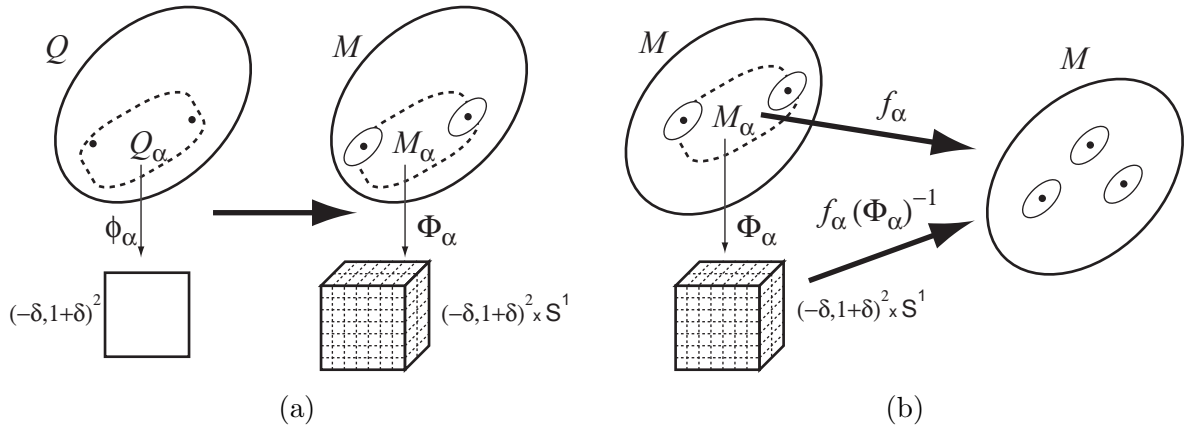


Figure 1: (a) Schematic representation of the parameterization of the surface Q and of its unit tangent bundle T^1Q. The discrete grid on M_α is obtained by lifting the lattice $G_h \subset (-\delta, 1+\delta)^2 \times \mathbf{S}^1$. (b) Construction of the local interpolant.

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,

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

is a smooth map between Euclidean domains, see Figure 1 (b). 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 gridpoints 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 $h \leq \delta/2$, 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 [6] 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 $M = T^1Q$ to itself, we impose an extra projection step which snaps a point $(\boldsymbol{x}, \boldsymbol{p}) \in R^6$ close to M back onto M. In a first step, we perform a Newton-type iteration

$$\boldsymbol{x} \leftarrow \boldsymbol{x} - \frac{F(\boldsymbol{x})}{|\nabla F(\boldsymbol{x})|^2} \nabla F(\boldsymbol{x})$$

until $|F(\boldsymbol{x})| \leq \varepsilon$, where ε is some prescribed accuracy parameter. Since $(\boldsymbol{x}, \boldsymbol{p})$ is always close to M, it usually takes only 3 to 4 iterations even when ε is as small as 10^{-9}. In a second step, we then project $\boldsymbol{p}$ onto the plane orthogonal to $\nabla F(\boldsymbol{x})$ (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

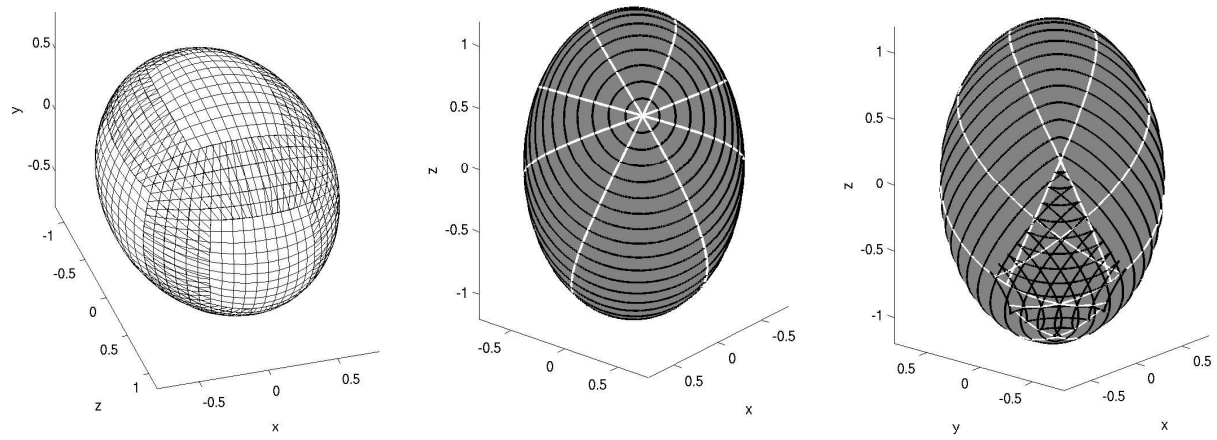
This section presents several numerical results. The proposed method is implemented in Matlab and all the computational results reported here were obtained on a desktop computer with a 2.6GHz CPU and 1GB of memory.

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}. To estimate the numerical error, we proceed as follows: we select N points $\{\boldsymbol{m}_i\}$ randomly from M; the "exact" solutions $g_{T_0}(\boldsymbol{m}_i)$ 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 $\pm z$. For every chart, the parameter domain $(-\delta, 1+\delta)$ is discretized with a Cartesian grid of size 16×16, see Figure 2(a) 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 seconds and the error is less than 10^{-6}. We then use $\tilde{g}_{T_0}$ to rapidly compute the geodesics starting from a single point, see Figures 2(b) and (c). The black curves are solutions at time $T = kT_0$ for integer values of k. The black curves are resolved with 1024 samples each.



(a) (b) (c)

Figure 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).

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

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

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 seconds, and the accuracy is about 10^{-6}. Figure 3 displays two different families of geodesics.

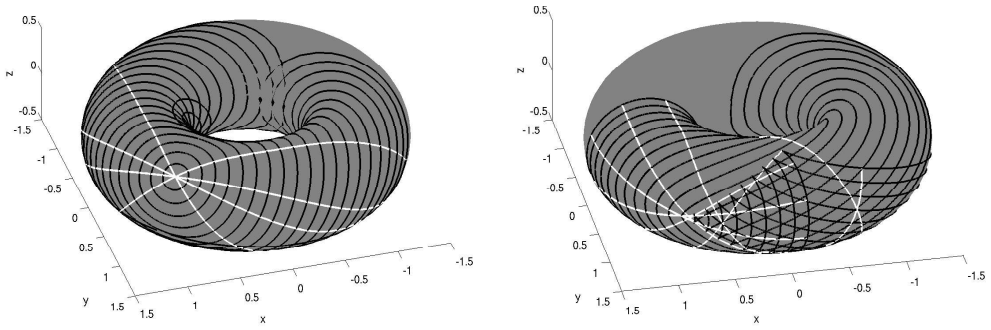
Example 3. In this example, we compute the weighted geodesics on a unit sphere. We choose the velocity field as $c(x,y,z)=1+x$. The parameterization and discretization used are here the same as those in Example 1. Figure 4 shows the computed geodesics starting from the north pole of the sphere.

2.4 Creeping rays

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

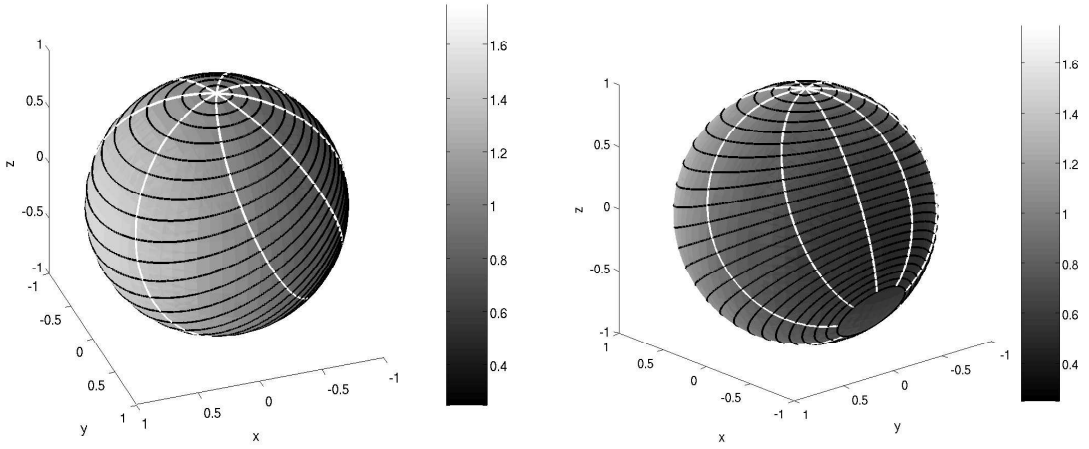
Example 4. The scatterer is here a "twisted" ellipsoid given by

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



(a) (b)

Figure 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.



(a) (b)

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

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 seconds. The computed map $\tilde{g}_{T_0}$ has accuracy about 10^{-5}.

Figure 5 plots the results for two different incident directions. The left 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 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.

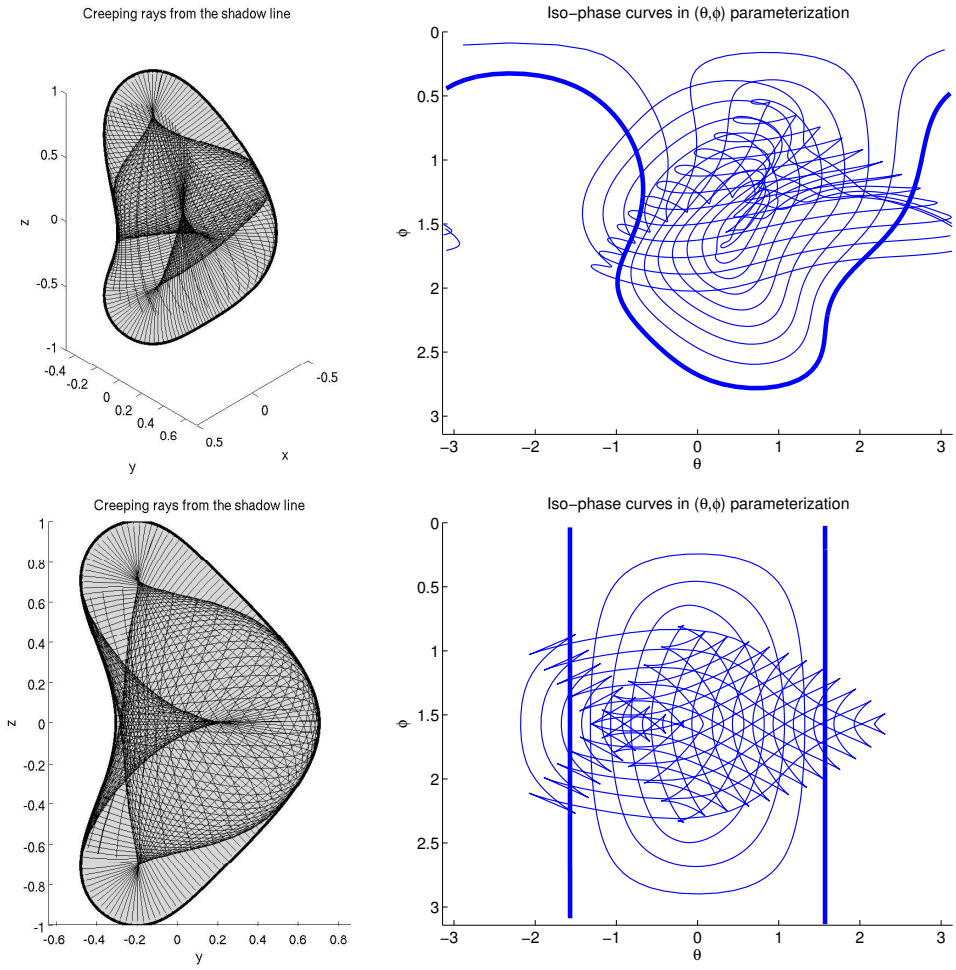


Figure 5: Example 4. Creeping rays on a "twisted torus" corresponding to two incident directions: $(1, 1, 1)$ (first row) and $(1, 0, 0)$ (second row). The left column shows the creeping rays on the surface of the scatterer. All the rays are generated from the shadow line (bold curve). The right column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

Example 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. Figure 6 plots the creeping rays starting from these two components separately.

2.5 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

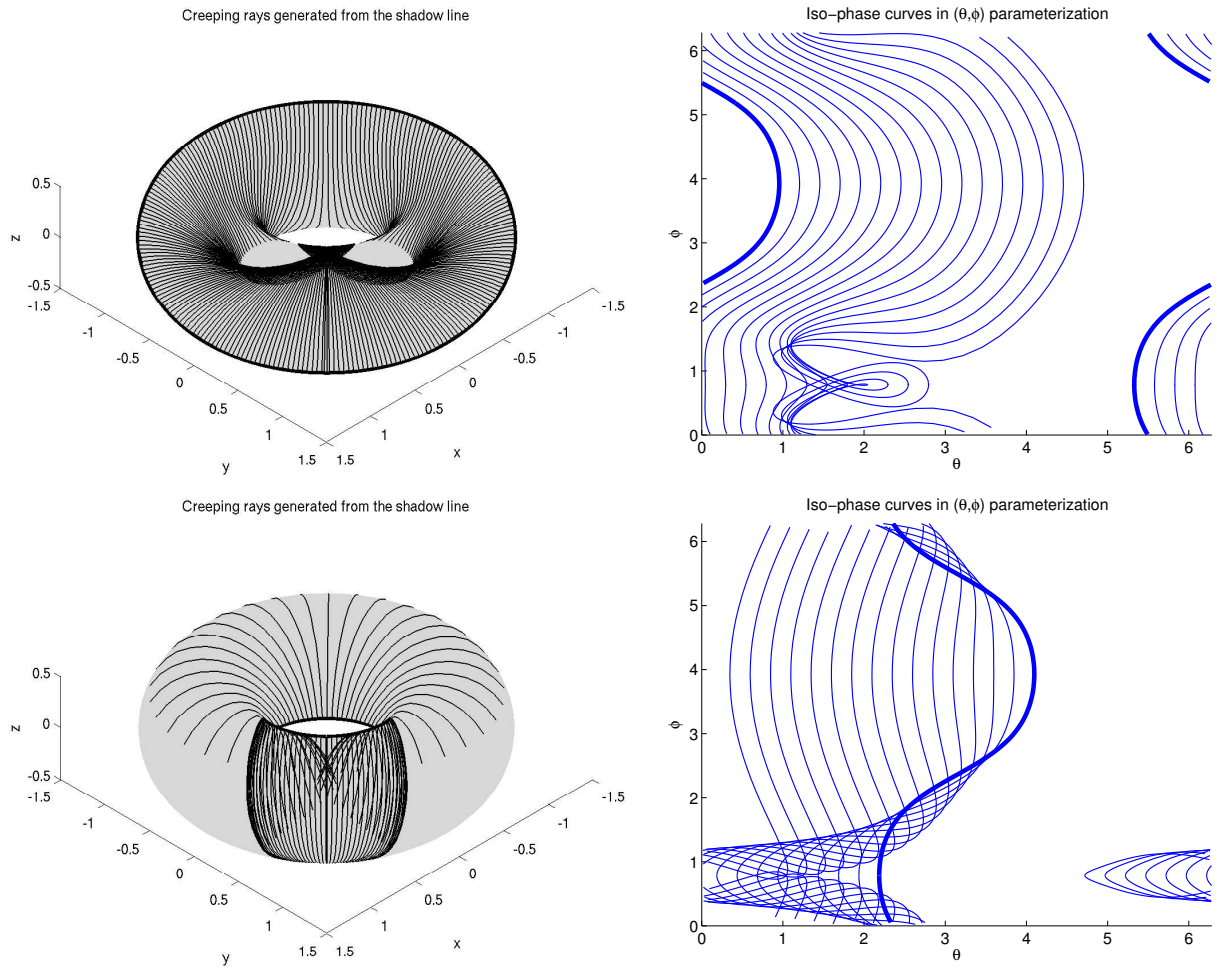


Figure 6: Example 5. Creeping rays on a torus corresponding to the incident direction $(1, 1, 1)$. The shadow line (bold) is composed of two disjoint curves. Each row is associated with a single shadow curve. The left figures show the creeping rays. The right figures show the iso-phase curves in the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

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.

References

[1] V. I. Arnold. *Mathematical methods of classical mechanics*. Springer-Verlag, New York, 1978. Translated from the Russian by K. Vogtmann and A. Weinstein, Graduate Texts in Mathematics, 60.

[2] M. Born and E. Wolf. *Principles of Optics*. Cambridge University Press, seventh edition, 1999.

[3] J. Chen and Y. Han. Shortest paths on a polyhedron. I. Computing shortest paths. *Internat. J. Comput. Geom. Appl.*, 6(2):127–144, 1996.

[4] C. de Boor. *A practical guide to splines*, volume 27 of *Applied Mathematical Sciences*. Springer-Verlag, New York, revised edition, 2001.

[5] S. Fomel and J. A. Sethian. Fast-phase space computation of multiple arrivals. *Proc. Natl. Acad. Sci. USA*, 99(11):7329–7334 (electronic), 2002.

[6] A. Iserles. *A first course in the numerical analysis of differential equations*. Cambridge Texts in Applied Mathematics. Cambridge University Press, Cambridge, 1996.

[7] J. B. Keller and R. M. Lewis. Asymptotic methods for partial differential equations: the reduced wave equation and Maxwell's equations. In *Surveys in applied mathematics, Vol. 1*, volume 1 of *Surveys Appl. Math.*, pages 1–82. Plenum, New York, 1995.

[8] R. Kimmel and J. A. Sethian. Computing geodesic paths on manifolds. *Proc. Natl. Acad. Sci. USA*, 95(15):8431–8435 (electronic), 1998.

[9] J. E. Marsden and T. S. Ratiu. *Introduction to mechanics and symmetry*, volume 17 of *Texts in Applied Mathematics*. Springer-Verlag, New York, 1994. A basic exposition of classical mechanical systems.

[10] F. Mémoli and G. Sapiro. Fast computation of weighted distance functions and geodesics on implicit hyper-surfaces. *J. Comput. Phys.*, 173(2):730–764, 2001.

[11] J. S. B. Mitchell, D. M. Mount, and C. H. Papadimitriou. The discrete geodesic problem. *SIAM J. Comput.*, 16(4):647–668, 1987.

[12] M. Motamed and O. Runborg. A fast phase space method for computing creeping rays. Submitted, 2005.

[13] J. A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proc. Nat. Acad. Sci. U.S.A.*, 93(4):1591–1595, 1996.

[14] V. Surazhsky, T. Surazhsky, D. Kirsanov, S. J. Gortler, and H. Hoppe. Fast exact and approximate geodesics on meshes. *ACM Trans. Graph.*, 24(3):553–560, 2005.

[15] L. Ying and E. J. Candès. The phase flow method. Technical report, California Institute of Technology, 2005. Submitted.