The Leap-Frog Algorithm and Optimal Control: Background and Demonstration

C. Yalçın Kaya
School of Mathematics
University of South Australia
The Levels, S.A. 5095 Australia
y.kaya@unisa.edu.au

J. Lyle Noakes
Department of Mathematics
University of Western Australia
Nedlands, W.A. 6907 Australia
lyle@maths.uwa.edu.au

Abstract

Pontryagin's Maximum Principle gives the necessary conditions for optimality of the behaviour of a control system, which requires the solution of a two-point boundary-value problem (TPBVP). TPBVPs are in general very difficult to solve, even if they occur as a result of a variational problem.
An algorithm recently developed by J.L. Noakes, called the Leap-Frog Algorithm, can find geodesics globally in a connected complete Riemannian manifold. A geodesic problem is a special type of TPBVP, but an optimal control problem results in a much more general TPBVP, which is much more difficult to solve.
In this work, a direct and intuitive implementation of the algorithm for general nonlinear systems with unbounded controls has been discussed. Prior to this discussion, we give an overview of the solution techniques for TPBVPs arising in optimal control. The motivation that led to the Leap-Frog Algorithm is also emphasized.

Key words: Geodesics, Optimal control, Nonlinear systems, Computations.

1 Introduction and Background

The optimal control of a dynamical system represented by ordinary differential equations usually involves the solution of a two-point boundary value problem (TPBVP). However, the task of solving a TPBVP is in general very difficult; the available solution techniques usually face serious convergence difficulties because of the lack of a good initial guess. Noakes [11] has recently developed a global algorithm for finding a geodesic joining two given points on a Riemannian manifold. The algorithm can be viewed as a solution method for a special type of TPBVP. It is simple to implement, and works well in practice, with no need for an initial guess. The algorithm will be called the *Leap-Frog Algorithm* because of the nature of its geometry.

It should be noted that an optimal control problem is much more general than finding a geodesic. Nevertheless recent numerical experiments suggest that a generalization of the algorithm to the case of optimal control may be possible. Within this context, a direct and intuitive implementation of the algorithm for a class of optimal control problems will be discussed, along with an example application.

In this introductory section, the optimal control problem is described, and the resulting control TPBVP is derived. An overview of the shooting methods for general TPBVPs is also given. As a followup, some of the general approaches to TPBVPs arising in optimal

control are also briefly discussed. The Lane and Riesenfeld (L&R) algorithm is a useful tool in the area of computer graphics and is a motivation to our proposed procedures. The L&R algorithm is introduced in Section 2. An extended L&R algorithm using geeodesics in more general spaces is also described. In Section 3, a practical description of the Leap-Frog Algorithm is given. Finally, in Sections 4 and 5, implementation of this algorithm in optimal control is discussed and an example of a specific application is presented.

Application of the Leap-Frog Algorithm to optimal control was first announced in Kaya & Noakes [4] in the form of a brief note. This work is a more detailed and extended account of that announcement. A theoretical investigation of the Leap-Frog Algorithm involving systems in the plane which are linear in control has been given by Kaya & Noakes [5].

1.1 The optimal control problem

A general control system is given by

$$\dot{x}(t) = f(x(t), u(t)) \tag{1.1}$$

where the state vector $x \in \mathbb{R}^n$, the control vector $u \in \mathbb{R}^m$, and time $t \in \mathbb{R}^+$. The vector field f in $\mathbb{R}^n$ is assumed to be smooth.

In this work, the states and the control inputs are assumed to be unconstrained.

The following class of optimal control problems will be considered.

$$\mathrm{P} : \begin{cases} \min\limits_{u} \int_0^1 L(x(t), u(t))\, dt \\ \text{subject to} \quad \dot{x}(t) = f(x(t), u(t)) \\ \text{such that} \quad x(0) = x_0, \quad x(1) = x_f \end{cases}$$

For simplicity the time t will be dropped from the arguments in subsequent references to this system. The dynamic constraint above is adjoined to the performance index to get the augmented performance index.

$$\min_u \int_0^1 [L(x, u) + \lambda^T (\dot{x} - f(x, u))]\, dt$$

where the adjoint vector $\lambda \in \mathbb{R}^n$ is called the costate vector, and $(\cdot)^T$ denotes the transpose. A Hamiltonian $H : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^m \longrightarrow \mathbb{R}$ is defined as

$$H(x, \lambda, u) = L(x, u) + \lambda^T f(x, u) \tag{1.2}$$

Through the calculus of variations (Kirk [7]), and also by the Pontryagin Maximum Principle (Pontryagin [12]), the following necessary conditions for optimality are obtained.

$$\dot{x} = \frac{\partial H}{\partial \lambda} \tag{1.3}$$

$$\dot{\lambda} = -\frac{\partial H}{\partial x} \tag{1.4}$$

$$\frac{\partial H}{\partial u} = 0 \tag{1.5}$$

It is assumed that an optimal control $u = u^o$ can be found explicitly using equation (1.5); in other words, one can find a feedback law

$$u^o = k(x, \lambda) \tag{1.6}$$

where $k : \mathbb{R}^n \times \mathbb{R}^n \longrightarrow \mathbb{R}^m$ is continuous in x and λ. Substitution of equation (1.6) into equations (1.3) and (1.4) leads to the following equations, which together are called here the *control TPBVP*.

$$\mathrm{P}_c : \begin{cases} \dot{x} = f(x, k(x, \lambda)) \\ \dot{\lambda} = -\left(\frac{\partial f}{\partial x}(x, k(x, u))\right) \lambda \\ \text{with} \quad x(0) = x_0, \quad x(1) = x_f \end{cases}$$

where $x(0) = x_0$ and $x(1) = x_f$ represent the point boundary conditions.

Solving problem (P_c) is in general very difficult. To emphasize this difficulty and locate problem (P_c) in a general framework, some of the most commonly cited methods of solving general TPBVPs will be reviewed in Section 1.2. In Section 1.3 a quick overview of some computational methods used in optimal control will also be provided.

1.2 Methods for general TPBVPs

A general TPBVP can be given as

$$\mathrm{P}_2 : \begin{cases} \dot{y}(t) = f(y, t) \\ r(y, 0) = 0 \quad \text{and} \quad s(y, 1) = 0 \end{cases}$$

where $y \in \mathbb{R}^{n'}$, $r : \mathbb{R}^{n'} \times \mathbb{R} \longrightarrow \mathbb{R}^{m'}$ and $s : \mathbb{R}^{n'} \times \mathbb{R} \longrightarrow \mathbb{R}^{n'-m'}$. In the above problem, r and s represent the boundary conditions at $t = 0$ and $t = 1$, respectively. Note that in the control TPBVP, $y = (x, \lambda)^T$. Then $y \in \mathbb{R}^{2n}$. In the case of problem (P_c), the boundary conditions are also given in a simpler fashion, i.e., for $i = 1, \ldots, n$,

$$y_i(0) - y_{i0} = 0 \qquad y_i(1) - y_{if} = 0 \ .$$

General methods to solve problem (P_2) can be found in Keller [6], Stoer & Bulirsch [14], Roberts & Shipman [13]. Of these methods, the simple and multiple shooting methods are the most common ones. Their superiority over the others is demonstrated by Stoer & Bulirsch [14] especially for some ill-posed linear boundary-value problem.

Shooting methods take their name from the situation in the TPBVP for a second-order differential equation with initial and final values of the solution prescribed (Roberts & Shipman [13]). Varying the initial slope gives rise to a set of profiles which suggest the trajectory of a projectile "shot" from an initial point. That initial slope is sought which results in the trajectory "hitting" the target; that is, the final value. Since $\dot{y}(0)$ is not known, the trajectory usually ends up with a discrepancy at $t = 1$, which is the difference between $y(1)$ and y_f. If the discrepancy is reasonably small, then Newton's method is applied to find the necessary $\dot{y}(0)$ so as to hit $y(1) = y_f$. Obviously, this works well if y_0 and y_f are close enough to each other, because in that case one obtains a good estimate of $\dot{y}(0)$ to start with. If, however, this is not the case, then either or both of the following difficulties may arise.

(i) a very good initial guess is required.
(ii) sensitivity to initial conditions or error accumulation in the solution of the initial value problems may cause trouble.

To cope with the above difficulties, simple shooting is sometimes replaced by multiple shooting. Then the interval $[0, 1]$ is subdivided into small subintervals whose initial data is simultaneously updated at each iteration. This alleviates chaotic effects but performance still depends on the quality of the initial guess made for each subinterval separately. As an added computational burden, the number of variables is substantially increased. As for simple shooting there is usually no guarantee of convergence.

1.3 Methods for control TPBVPs

Methods that are used for the solution of TPBVPs arising in optimal control usually incorporate some combination of

(a) Multiple shooting
(b) Collocation methods
(c) Continuation (or homotopy) methods

As multiple shooting is a very accurate method but does require a good initial guess, it is sometimes used in conjunction with some other method, for example, the method of collocation (von Stryk & Schlemmer [15]). The method of collocation is used to obtain an approximate solution, in terms of cubic splines for example, which is then fed into the the multiple shooting algorithm to achieve the required solution accurately. This two-fold procedure may eventually need long computational times.

The continuation (or homotopy) method is also a method used in conjunction with multiple shooting (Chen & Desrochers [2], Chen et al. [3]). Usually, the nonlinear system is partitioned into its linear and nonlinear parts, where nonlinear terms are multiplied by some continuation (or homotopy) parameters, whose values range between 0 and 1. These parameters are initially set to zero, in which case only the linear part of the system does take effect. After the solution is found easily for the linear part, the value of the parameters are increased gradually every time to find a sequence of solutions that converges to the real solution. This procedure however may take a long time before achieving the required solution, depending on the nonlinearity of the problem.

2 Approximations and Computer Graphics

The purpose of this section is to provide some motivation for the development of the Leap-Frog Algorithm.

2.1 The L&R algorithm

Lane and Riesenfeld [8] presented two fast subdivision algorithms for the evaluation of B-spline and Bernstein curves and surfaces, which are important concepts in computer graphics, in particular in CAD and CAM. These subdivision algorithms involve dividing a polygon that is formed by the data points into smaller polygons, called subpolygons, so that a repeated subdivision results in points in the subpolygons to converge to the required curve.

2.2 Geodesics

Consider the parameterized curve $\gamma : [0, 1] \longrightarrow N$, where N is a Riemannian manifold. The curve γ is said to be a geodesic if its velocity vector is constant (or parallel) with respect to the Riemannian structure. The interested reader is referred to Boothby [1] for this and the following definitions. A curve of minimum length (with respect to a Riemannian distance) joining two points y_0 and y_f in N is a geodesic—note however that a geodesic does not have to be of minimum length. In other words, if a curve y minimizes a functional

$$\int_0^1 Q(y, \dot{y})\, dt \tag{2.1}$$

where Q is a so-called Riemannian metric, then it is a geodesic. Note that the Riemannian metric Q is, by definition, positive definite and homogeneous of degree 2 in $\dot{y}$.

If N is the Euclidean space $\mathbb{R}^n$ then the geodesics are straight lines. If N is sphere, then great circles are geodesics. Note that in the case of a sphere the geodesics are closed curves in marked contrast to the straight lines in Euclidean space.

A geodesic solves the second-order ODEs

$$\ddot{y}_i = -\sum_{j,k=1}^{n} \Gamma^i_{jk}(y)\, \dot{y}_j\, \dot{y}_k \tag{2.2}$$

for $k = 1, \ldots, n$, with $y_i(0)$ and $y_i(1)$ given. The nonlinear functions Γ^i_{jk} of y, which are referred to as the Christoffel symbols in differential geometry, depend on the coordinate charts. Note that the right-hand side of equation (2.2) is homogeneous of degree 2 in $\dot{y}$.

2.3 Nonlinear corner-cutting

Using the observation that, in the L&R algorithm, lines in the polygons are the shortest paths joining two given points in the Euclidean space, Noakes [9] gives a Riemannian generalization of a special case of the L&R algorithm. This generalization is called the *nonlinear corner-cutting algorithm*, where the line segments in the L&R algorithm are replaced by geodesic segments, and Euclidean space by a Riemannian manifold. In this case the limit of the sequence $\{q_i\}$ of points obtained by the ith subdivision is shown to be a C^∞ curve.

Noakes [9, 10]motivated a global algorithm for finding geodesics which is to be presented in the following section.

3 The Leap-Frog Algorithm

Let N be a C^∞ path-connected Riemannian n-manifold where $n \geq 1$ is finite. If the initial and final points y_0 and y_f in N are nearby, then Γ^i_{jk} in equation (2.2) can be written down explicitly, and then one needs to solve the TPBVP given in (2.2) in order to find a geodesic between y_0 and y_f. As pointed out in Section 1.2, the simple shooting method is well-suited to such a problem with nearby boundary points.

When y_0 and y_f are not nearby, simple shooting method does not work well. Then usually multiple shooting as described before is used, however it is also not free from troubles. The global algorithm presented by Noakes [11] is very simple to implement, and works well in practice, with no need for an initial guess. This global algorithm, as its geometric nature suggests, will be called the *Leap-Frog Algorithm*.

The Leap-Frog Algorithm resembles multiple shooting in that the interval $[0, 1]$ is subdivided and geodesics are found separately over each subinterval. So the algorithm is not expected to be troubled by the nonlinear dynamics either. The most important differences between the Leap-Frog Algorithm and multiple shooting are

- Each step of the Leap-Frog Algorithm updates only n real variables at a time.
- The curves of the Leap-Frog Algorithm satisfy both boundary conditions at every step of the iteration.

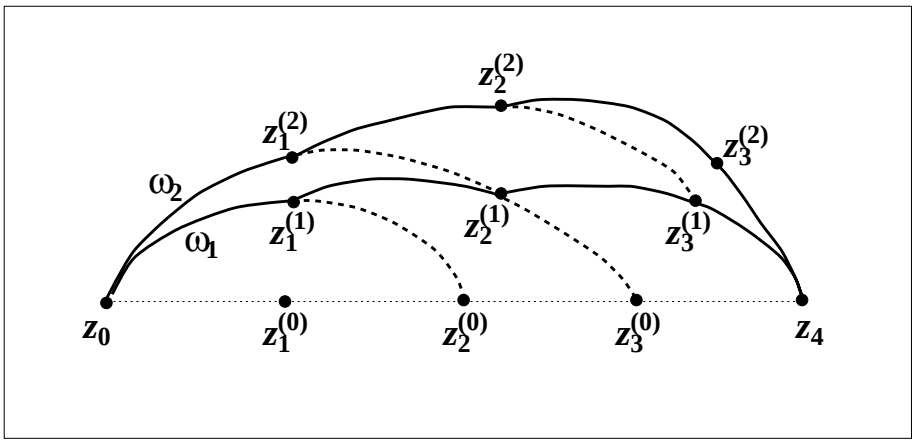

Figure 1: The Leap-Frog Algorithm

- The Leap-Frog Algorithm almost always converges, without the need for an initial guess (Noakes [11]). Usually there is no need to search for convergent subsequences. The entire sequence of approximations is proved to converge under fairly general conditions.

A description of the algorithm is given below.

Consider a parameterized piecewise-geodesic curve $\omega : [0,1] \longrightarrow N$ from y_0 to y_f. If there occur j geodesic segments, then ω is determined by the $(j-1)$-tuple

$$(z_1, z_2, \ldots, z_{j-1})$$

of junctions of geodesic segments. Each z_i is adjusted as follows. Set $z_0 = y_0$ and $z_j = y_f$. Then moving z_i onto the minimal geodesic joining z_{i-1} and z_{i+1} achieves the largest possible decrease in length while keeping other variables fixed. For convenience, the point where z_i goes onto the minimal geodesic is taken to be the midpoint. The Leap-Frog Algorithm consists of iterating this procedure so that all z_i are moved infinitely often where $i = 1, \ldots, q-1$. This generates a sequence $\Omega = \{\omega_k : [0,1] \longrightarrow N : k \geq 1\}$ of piecewise geodesics whose lengths are decreasing.

The sequence $\Omega = \{\omega_k\}$ has a uniformly convergent subsequence. However, in many cases it is not necessary to go to the subsequences as Ω is guaranteed to converge to a minimal geodesic. The only case where it might be necessary to go to subsequences is where there are distinct geodesics of the same length, which are homotopic through curves from y_0 to y_f.

The Leap-Frog Algorithm is illustrated in Figure 1. The first set of points z_i, $i = 1, 2, 3$, is chosen arbitrarily. Two iterations are depicted in the figure.

4 Implications and an Example Application

Recall that if a curve minimizes (2.1), then it is a geodesic and it is represented by equation (2.2). Now, let $x = y$ and $\lambda = \dot{y}$. Then equation (2.2) can be rewritten in terms of the coordinates x_i and λ_i as

$$\mathrm{P}_g : \begin{cases} \dot{x}_i = \lambda_i \\ \dot{\lambda}_i = -\sum_{j,k=1}^{n} \Gamma^i_{jk}(x)\, \lambda_j\, \lambda_k \\ \text{with} \quad x(0) = y_0, \quad x(1) = y_f \end{cases}$$

Iteration	n_p	*Cost*
1	4	13615
2	4	10038
3	4	9689
4	4	9491
5	2	2786

Table 1: Iterations of the Leap-Frog Algorithm.

for $i = 1, \ldots, n$. However, it does not seem plausible to try and make immediate analogies between problems (P_g) and (P_c).

Despite the fact that theory is originally developed only for the case of solving the geodesic problem (P_g), recent numerical experiments suggest that the algorithm would also work for problem (P_c).

We consider a simulation example below.

$$\mathrm{P}: \begin{cases} \min\limits_u \frac{1}{2}\int_0^1 u^2\, dt \\ \quad \text{subject to} \quad \dot{x}_1 = x_2 \\ \qquad\qquad\quad\ \ \dot{x}_2 = x_3 + (1 - x_1^2)\, x_1 \\ \qquad\qquad\quad\ \ \dot{x}_3 = x_2 \sin x_2 + u \\ \text{such that} \quad x(0) = [-4,\ 0,\ 0]^T, \quad x(1) = [-2,\ 0,\ 0]^T. \end{cases}$$

The Hamiltonian is

$$H(x, \lambda, u) = \frac{1}{2}u^2 + \lambda_1\, x_2 + \lambda_2\, (x_3 + ((1 - x_1^2)\, x_1) + \lambda_3\, (x_2 \sin x_2 + u) \tag{4.1}$$

The necessary condition $\partial H/\partial u = 0$ gives the optimal control input as $u^o = -\lambda_3$. Then, problem (P_c) for this example becomes

$$\mathrm{P}_c: \begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = x_3 + (1 - x_1^2)\, x_1 \\ \dot{x}_3 = x_2 \sin x_2 - \lambda_3 \\ \dot{\lambda}_1 = -(1 - 3x_1^2)\, \lambda_2 \\ \dot{\lambda}_2 = -(\lambda_1 + \lambda_3(\sin x_2 + x_2 \cos x_2)) \\ \dot{\lambda}_3 = -\lambda_2 \\ \text{with} \quad x(0) = [-4,\ 0,\ 0]^T, \quad x(1) = [-2,\ 0,\ 0]^T. \end{cases}$$

The problem (P_c) given above has been solved by using the Leap-Frog Algorithm. The results have been documented in Table 1. At the beginning, the junction points were taken on a straight line between $x(0)$ and $x(1)$. The number of partitions n_p was initially taken to be 2, but it was seen to fail. This is not unexpected, given the nonlinearity of the system. Note anyway that when $n_p = 2$, the algorithm is nothing but the simple shooting from $x(0)$ to $x(1)$. Application of the algorithm succeeded when n_p was set to 4. In the general computer program written, we consider values of n_p as powers of 2, for ease of bookkeeping. At each iteration $n_p = 2$ (the simple shooting) was tried first, but it did not work until the 5th iteration. The optimal cost was found to be 2786.

As a general strategy, one should start with the lowest possible n_p, because as n_p increases, the convergence becomes slower. One should then try to reduce n_p in further iterations to attain a faster convergence, until eventually getting to $n_p = 2$, the simple shooting case.

References

[1] Boothby, W.M. (1986). *An Introduction to Differentiable Manifolds and Riemannian Geometry.* Academic Press, London.

[2] Chen, Y., & Desrochers, A.A. (1993). Minimum-time control laws for robotic manipulators. *International Journal of Control*, **57**(1), 1-27.

[3] Chen, Y., & Huang, J., & Wen, J.T.Y. (1993). A continuation method for time-optimal control synthesis for robotic point-to-point motion. *Proceedings of the 32nd Conference on Decision and Control*, San Antonio, Texas, U.S.A., December 1993.

[4] Kaya, C.Y., & Noakes, J.L. (1997). Geodesics and an optimal control algorithm. *Proceedings of the 36th IEEE CDC*, San Diego, California, U.S.A., December 1997, pp. 4918-4919.

[5] Kaya, C.Y., & Noakes, J.L. (1998). The Leap-Frog Algorithm and optimal control: theoretical aspects. *Proceedings of ICOTA 98*, Perth, Australia.

[6] Keller, H.B. (1968). *Numerical Methods for Two-Point Boundary-Value Problems*, Blaisdell Publishing Co.

[7] Kirk, D.E. (1970). *Optimal Control Theory: An Introduction*, Prentice-Hall, New Jersey.

[8] Lane, J.M., & Riesenfeld R.F. (1980). A theoretical development for the computer generation and display of piecewise polynomial surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-2**(1), 35-46.

[9] Noakes, J.L. (1997a). Nonlinear Corner-Cutting. *Advances in Computational Mathematics* (to appear).

[10] Noakes, J.L. (1997b). Riemannian quadratics. *Proceedings of Chamonix* (to appear).

[11] Noakes, J.L. (1997c). A global algorithm for geodesics. *Journal of the Australian Mathematical Society, Series A* (to appear).

[12] Pontryagin, L.S., Boltyanskii V.G., Gamkrelidze, R.V., & Mishenko, E.F. (1962). *the Mathematical Theory of Optimal Processes*, Interscience, New York.

[13] Roberts, S.M., & Shipman, J.S. (1972). *Two-Point Boundary Value Problems: Shooting Methods*, American Elsevier, New York.

[14] Stoer, J., & Bulirsch, R. (1993). *Introduction to Numerical Analysis*, 2nd edition, Springer-Verlag, New York.

[15] von Stryk, O., & Schlemmer, M. (1994). Optimal control of the industrial robot Manutec r3. In: *Computational Optimal Control, Eds: Bulirsch, R., & Kraft, D., Birkhäuser Verlag, Basel.*