Querying Databases of Trajectories of Differential Equations II: Index Functions

Robert Grossman*
University of Illinois at Chicago

June, 1990

Abstract Suppose that a large number of parameterized trajectories γ of a dynamical system evolving in $\mathbf{R}^N$ are stored in a database. Let $\eta \subset \mathbf{R}^N$ denote a parameterized path in Euclidean space, and let $||\cdot||$ denote a norm on the space of paths. In this paper, we define data structures and indices for trajectories and give algorithms to answer queries of the following forms:
Query 1. Given a path η, determine whether η occurs as a subtrajectory of any trajectory γ from the database. If so, return the trajectory; otherwise, return null.
Query 2. Given a path η, return the trajectory γ from the database which minimizes the norm

$$||\eta - \gamma||.$$

1 Queries about trajectories

Suppose that a large number of parameterized trajectories γ of a dynamical system evolving in $\mathbf{R}^N$ are stored in a database. Let $\eta \subset \mathbf{R}^N$ denote a parameterized path in Euclidean space, and let $||\cdot||$ denote a norm on the space of paths to be specified later. In this paper, we define a data structure and indices to represent trajectories of dynamical systems and sketch algorithms to answer queries of the following forms:
Query 1. Given a path η, determine whether η occurs as a subtrajectory of any trajectory γ from the database. If so, return the trajectory; otherwise, return null.
Query 2. Given a path η, return the trajectory γ from the database which minimizes the norm

$$||\eta - \gamma||.$$

The paper is a successor to [2], which describes the data structure to store trajectories which is used here.

Efficient algorithms to answer these type of queries should prove useful for a number of applications. As an example, consider the path-planning problem for a robotic arm. Suppose that a large number of feasible trajectories of the robotic arm have been stored in a database. Let η be the desired path of the arm. It is not necessary that η itself be a feasible trajectory. Query 2 would return the feasible trajectory γ of the arm which is closest to the desired path η.

As another example, consider a database containing control trajectories for an aircraft. Assume that those trajectories which enter into an unstable control regime somewhere along their flight path are tagged.

*This research is supported in part by grant NAG2-513 from NASA and by grant DMS-8904740 from the National Science Foundation and by the Laboratory for Advanced Computing. Address: Department of Mathematics, Statistics, and Computer Science, Mail Code 249, University of Illinois at Chicago, Box 4348, Chicago, IL 60680, (312) 413-2164, grossman@uicbert.eecs.uic.edu.

Let η denote a measured portion of the flight path. Then Query 1 would return the nearest full control trajectory in the database, which includes information about the stability of the trajectory. More generally, one could imagine retrieving from the database those stable trajectories which avoid a given obstacle, such as a turbulent region of space. In other words, the query could be used as part of a supervisory control system and be viewed as a means of extracting qualitative or summary information about the control system.

The data structure we use to represent trajectories is closely related to hashing methods for curves that have been used in computer vision; see [7] and [8]. A related means of extracting qualitative information from dynamical systems is described in [1]. We are concerned in this paper with data structures and indexing for object oriented databases consisting of trajectories. For general methods of indexing in object oriented databases see [3], [4], [5], and [6].

In Section 2 we review the relevant facts about trajectories of differential equations and define different data structures to store trajectories. In Section 3, we show how these data structures can be used to answer the queries above. Section 4 contains some concluding remarks.

2 Paths, trees and vector fields

In this section, we describe a data structure for paths following [2]. The point of view is to assume that the path arises from a trajectory of a differential equation and to base the data structure upon the initial value problem for the differential equation.

We begin by recalling some basic facts and definitions about trajectories of differential equations. Let $D_\mu = \partial/\partial x_\mu$. A *vector field*

$$E = \sum_{\mu=1}^{N} a^\mu D_\mu$$

on $\mathbf{R}^N$ is determined by specifying N functions

$$a_\mu : \mathbf{R}^N \longrightarrow \mathbf{R}.$$

We also denote the vector field by E_a. A parameterized path

$$\gamma : [t^0, t^1] \subset \mathbf{R} \longrightarrow \mathbf{R}^N$$

is called a *trajectory* of the dynamical system

$$\dot{x}(t) = E_a(x(t)) \tag{1}$$

in case it is the unique solution of the initial value problem

$$\dot{x}(t) = E_a(x(t)), \qquad x(t^0) = \gamma(t^0). \tag{2}$$

We define the *vector field/reference point representation* or *VEFREP* of a path η to be the pair (E, R), consisting of a vector field E and a reference or initial point R, where the trajectory is the solution of the initial value problem

$$\dot{x}(t) = E(x(t)), \qquad x(t^0) = R.$$

Note that this representation is not unique. Indeed, several different vector fields could have a given spatial curve as a trajectory, while any point along the spatial curve could serve as the initial value.

We now give an algorithm whose input is a parameterized path

$$\eta : [t^0, t^1] \subset \mathbf{R} \longrightarrow \mathbf{R}^N,$$

and whose output is a labeled, rooted binary tree. We assume for convenience that $t^0 = 0$ and $t^1 = 1$; if not, we can reparameterize. We do not assume that η is a trajectory of the dynamical system (1). To define the tree, we first fix a tolerance $\epsilon > 0$. The tree we define is a subset of the complete rooted binary tree. There are 2^k children at height k from the root: number them left to right from 1 to 2^k. We assign two labels to the jth node v from the left at height k:

$$\kappa(v) = (1/2^k)\left(\eta(\frac{j}{2^k}) - \eta(\frac{j-1}{2^k})\right) \in \mathbf{R}^N$$

and

$$\theta(v) = \eta\left(\frac{j-1}{2^k}\right) \in \mathbf{R}^N.$$

We use the following stopping criterion to grow the tree. If a node has children v and v' with labels κ and κ', respectively, and if $||\kappa - \kappa'|| \leq \epsilon$, then the nodes v and v' are leaves. Here $||\cdot||$ denotes the Euclidean norm. We denote by $T(\eta)$ the tree that arises in this fashion. This tree has a simple interpretation: the θ labels represent points on the path η, while the κ labels represent approximate tangent vectors at those points. The tree is grown until the difference between two adjacent tangent vectors is uniformly small.

Using the tree $T(\eta)$, we now define a vector field $E(\eta)$. The vector field $E(\eta)$ is simply the vector field which interpolates the labels $(\theta(v), \eta(v))$

$$E(\eta)(\theta(v)) = \kappa(v), \tag{3}$$

for all leaves v in $T(\eta)$. Recall that $\theta(v)$ is the point on the curve η corresponding to the node v, and $\kappa(v)$ is the approximate tangent to the curve at that point.

For some applications, it is better to impose an upper bound on the degree of the interpolating functions. Let q denote this bound. In this case, we can define the vector field $E(\eta)$ by requiring that the coefficients b_j^μ minimize the quantity

$$\sum_{\text{leaves } v} ||E(\eta)(\theta(v)) - \kappa(v)||, \tag{4}$$

where the minimum is over vector fields with interpolating functions of degree less than or equal to q.

We conclude this section by defining a specific point $R(\eta)$, associated with a parametrized path

$$\eta : [t^0, t^1] \subset \mathbf{R} \longrightarrow \mathbf{R}^N.$$

Let $T(\eta)$ the corresponding tree and $E(\eta)$ the associated vector field. Consider the trajectory defined by the initial value problem

$$\dot{x}(t) = E_a(x(t)), \qquad x(0) = \eta(t^0).$$

Let γ denote this trajectory. In general, γ is only an approximation to the path η. Define $R(\eta) \in \mathbf{R}^N$ as follows: if the path γ and the unit sphere in $\mathbf{R}^N$ intersect precisely once, let $R(\eta)$ denote this intersection; if they intersect several times, let $R(\eta)$ denote the intersection which occurs first when the list of intersections is ordered in lexigraphical order; otherwise, let $R(\eta)$ denote the closest point between the unit sphere and the trajectory γ.

3 Query Algorithms

Let $\eta \subset \mathbf{R}^N$ denote a path. In this section, we define an index $I(\eta)$ that can be used for storing and accessing the path η. First, fix injective functions

$$h_n : \mathbf{R}^n \longrightarrow \{1, 2, \ldots\},$$

for each $n = 1, 2, \ldots$ Given a path η, first compute its VEFREP $(E(\eta), R(\eta))$ from its rectifying tree $T(\eta)$. Assume the tree $T(\eta)$ has K leaves. Next, view the coefficients of the vector field $E(\eta)$ as a $K \cdot N$ vector, so that the pair $(E(\eta), R(\eta))$ has $(K+1)N$ components. Then the *hash index* $H(\eta)$ associated with η is defined by

$$H(\eta) = h_{(K+1)N}(E(\eta), R(\eta)).$$

We can now assign indices $I(\eta)$ to trajectories sequentially: use the hash index $H(\eta)$ to determine whether the path η has an index assigned to it. If so, use that index; if not, use the next available index.

Suppose that $\gamma_1, \ldots, \gamma_P$ are trajectories of the dynamical system

$$\dot{x}(t) = E_a(x(t)),$$

as a ranges over some parameter space. To each such trajectory γ, let

$$(E(\gamma), R(\gamma))$$

denote its VEFREP representation. Given a parameterized path η, Algorithm 1 below returns the trajectory γ from the database which contains a segment equal to the path η. If there is no such trajectory, null is returned.

Algorithm 1. The input is a parameterized path η, and the output is the trajectory γ from the database answering Query 1. Fix $\epsilon > 0$ and $q > 1$.

Step 1. This step is a precomputation. For each trajectory γ_i, $i = 1, \ldots, P$, compute its VEFREP representation $(E(\gamma_i), R(\gamma_i))$. This depends upon q and ϵ.

Step 2. Given a query path η, compute its rectifying tree $T(\eta)$. This depends upon ϵ. Using $T(\eta)$ and Equation 4, compute its VEFREP representation $(E(\eta), R(\eta))$. This depends upon q.

Step 3. Using the VEFREP $(E(\eta), R(\eta))$, compute the hash index $H(\eta)$. If there is an index in the row $H(\eta)$ of the index table, retrieve the VEFREP representation (E, R) corresponding to this index; otherwise, return null.

Step 4. If Step 3 yielded a VEFREP (E, R), return the trajectory γ which is the solution to the initial value problem

$$\dot{x}(t) = E(x(t)), \qquad x(0) = R;$$

otherwise, return null.

Theorem 3.1 *Assume that the database contains n trajectories. Algorithm 1 answers Query 1 in time* $O(1)$.

Easy modifications of Algorithm 1 can be used to answer Query 2 in time $O(n)$.

4 Conclusion

In this paper, we have described preliminary work concerned with queries of databases containing trajectories of differential equations. Trajectories of differential equations have many different representations. For the types of queries considered here, we have chosen to represent parameterized trajectories

$$\gamma : [t^0, t^1] \subset \mathbf{R} \longrightarrow \mathbf{R}^N$$

by a pair, consisting of a vector field E on $\mathbf{R}^N$ with polynomial coefficients and a point $R \in \mathbf{R}^N$ such that the trajectory is the solution of the initial value problem:

$$\dot{x}(t) = E(x(t)), \qquad x(t^0) = R.$$

We call this a VEFREP representation. Using the VEFREP representation, we have introduced an index $I(\gamma)$ and algorithms to answer queries which retrieve subtrajectories and close by trajectories of a given query trajectory.

References

[1] H. Abelson and G. J. Sussman, Dynamicists' Workbench I: "Automatic Preparation of Numerical Experiments," R. Grossman (editor), *Symbolic Computation: Applications to Scientific Computing*, SIAM, 1989.

[2] R. Grossman, "Querying Databases of Trajectories of Differential Equations I: Data Structures for Trajectories," to appear in *Proceedings of the 23rd Hawaii International Conference on Systems Sciences*, IEEE, 1990.

[3] W. Kim, J. Banerjee, H. T. Chou, J. F. G. Garze, and D. Woelk, "Composite Object Support in an Object Oriented Database System," *Proceedings of OOPSLA 1987.*

[4] W. Kim, K-C Kim, and A. Dale, "Indexing Techniques for Object-Oriented Databases," in *Object-Oriented Concepts, Databases, and Applications*, W. Kim and F. H. Lochovsky, editors, ACM, New York, 1989.

[5] D. Maier and J. Stein, "Indexing in an Object-Oriented DBMS," Proceedings of the 1986 International Workshop on Object-Oriented Database Systems, Pacific Grove, California, 1986.

[6] N. Paton and P. M. D. Gray, "Identification of Database Objects by Key," in *Advances in Object-Oriented Database Systems*, K. R. Dittrick, editor, Springer-Verlag, Berlin, 1988, pp. 280–285.

[7] J. T. Schwartz and M. Sharir, "Identification of Partially Obscured Objects in Two Dimensions by Matching of Noisy *Characteristic Curves*," International J. Robotics Research, 6 (1987), 29–44.

[8] H. Wolfson, "On Curve Matching," *Proceedings of Workshop on Computer Vision, Miami Beach*, IEEE, 1987, 307–310.