International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13 Number 23 (2018) pp.16204-16212

Dynamic Programming Method for Best Piecewise Linear Approximation for Vector Field of Nonlinear Boundary Value Problems on the Interval [0, 1]

Duggirala Meher Krishna
Student, Department of Electronics and Communication Engineering,
Gayatri Vidya Parishad College of Engineering (Autonomous), Madhurawada,
Visakhapatnam, Andhra Pradesh, India.

Duggirala Ravi
Professor, Department of Computer Science and Engineering,
Gayatri Vidya Parishad College of Engineering (Autonomous), Madhurawada,
Visakhapatnam, Andhra Pradesh, India.

Abstract
An important problem that arises in many engineering applications is the boundary value problem for ordinary differential equations. There have been many computational methods proposed for dealing with this problem. The convergence of the iterative schemes to a true solution, when one such exists, and their numerical stability are the central issues discussed in the literature. In this paper, we discuss a method for approximating the vector field, maintaining the boundary conditions and numerical stability. If a true solution exists, a subsequence of solutions convergent to one such can be produced, by finer discretization of the solution space.

Keywords: differential equations, boundary value problems, dynamic programming method.

Introduction

During the last few decades there has been a remarkable growth of interest in problems associated with solving linear and nonlinear ordinary differential equations satisfying boundary conditions. For many of the nonlinear boundary value problems that occur in engineering and applied sciences, it is difficult to obtain a solution analytically. For a nonlinear boundary value problem, the difficulty lies in establishing the existence of a solution mathematically, though in some cases multiple solutions exist. Approximation of the solution space of a given differential equation has gained importance as it speeds up or helps in solving the problem efficiently. These approximation methods can be put into two classes: (*i*) those in which a solution is approximated numerically at a number of discrete points of the domain, and (*ii*) those in which a solution is approximated by a finite number of terms of a sequence of functions. The approach in (*ii*) is called a weighted residual method. In most of the numerical methods described in the literature, we may have to add a sufficient number of some undetermined variables with implicitly assumed conditions at one end of the domain, and adjust the additional variables until the required conditions are satisfied at the other end to obtain the solution of the boundary value problem [3 – 5, 8, 11, 12 – 14], and further approximate the derivatives of the dependent variables with forward, backward or central difference operators defined at the grid points [10]. In the first part of a numerical scheme as just mentioned, the convergence may be very slow, and in the second part, convergence and stability of the particular difference scheme may depend on the selection of the approximations used for the derivatives involved in the differential equation and the boundary conditions. It may also be the case that the chosen difference method is not numerically stable, resulting in chaos phenomenon creeping into the iterative schemes, at places where matrix inversions are utilized, without the solver being explicitly aware of its entry. In some cases, suitable *regularization* and *relaxation* conditions, involving more variables, may have to be added to the constraints formulated in the previous steps. In the weighted residual methods, difference equations are generated using approximation methods with piecewise polynomial solutions [10]. Among the most popular and successful techniques for solving boundary value problems with nonlinearities is Galerkin procedure. In this approach, the solution of the ordinary differential equation is expressed as a linear combination of certain basis functions, and the coefficients of the basis functions are determined by requiring that the residual be orthogonal to each of the basis functions. The difficulty lies in the selection of basis

functions to obtain the desired solution, that can take care of the boundary conditions simultaneously. In recent times, the concept of piecewise linear approximation of the differential equation gained momentum [6, 8, 10, 11, 12]. The two point boundary value problems are approximated by piecewise linear ones which have analytical solutions and reduced to finding the slope of the solution at the left boundary, so that the boundary conditions at the right end of the interval are satisfied. This approach results in a complex system of non-linear algebraic equations. Some more recent and highly efficient algorithms [7] for solving these complex systems of differential and algebraic equations (DAE) can be used for computational purpose.

The motivation for the present work is the necessity of a new method that is applicable for most or all general continuous vector fields and general boundary conditions. The objective of our study is to find efficiently a solution, if exists, by an algorithm, such that the algorithm is able to detect and report when there is no solution, if an error term does not fall below a threshold, despite using various approximation schemes with several basis functions. We propose a new method based on dynamic programming for solving boundary value problems in one variable. The dynamic programming based formulation is adapted for obtaining an optimal approximation for the vector field for genera of two-point boundary value problem, which is usually formulated as an optimal control problem in the literature [2]. For improving an initial approximation, repeated application of the dynamic programming algorithm with refined discretization of the parameter space can be used. A modified Newton-Raphson method for improving an approximate solution along with updating the initial value is also discussed. These aspects are newly introduced in this work

.

Best Affine Approximation for Vector Fields of Two Point Boundary Value Problems

In this section, we consider the boundary value problem

$$\left.\begin{array}{l} \frac{dx}{dt} = \boldsymbol{f}(\boldsymbol{x}(t),\ t),\quad for\ \ 0 < t < 1, \\ \quad and\quad \beta\big(x(0),\ x(1)\big) = 0 \end{array}\right\} \qquad (1)$$

where $\boldsymbol{f} : \mathbf{R}^n \times [0,1] \to \mathbf{R}^n$ and $\beta : \mathbf{R}^n \times \mathbf{R}^n \to \mathbf{R}$ are continuous functions processing continuous derivatives of as high order as may be required, for some positive integer n, which is the dimension of the Euclidean space, and $\mathbf{R}$ is the set of real numbers. The vectors are written as columns of appropriate dimensions. The objective of obtaining a piecewise affine approximation is stated as follows:

1. Let, for some fixed integer $N \geq 2$, $0 = t_0 < t_1 < \cdots < t_N = 1$ be fixed node points of the time interval [0, 1].

2. Then, we obtain optimal values for the parameters of form $\pi_i = (A_i(t),\ \mathbf{b}_i(t),\ \theta_i)$, where $A_i(t)$ and $\mathbf{b}_i(t)$ are $n \times n$ and $n \times 1$ matrices, respectively, consisting of undetermined linear combination of some fixed basis functions defined on $[t_i,\ t_{i+1}]$ and θ_i is an $n \times 1$ column vector of undetermined constants, for $0 \leq i \leq N-1$, such that the curve $\mathbf{y}(t)$ satisfying

$$\left.\begin{array}{l} \frac{dy}{dt} = A_i(\boldsymbol{t})\,\boldsymbol{y}(t) + \mathbf{b}_i(t),\ for\ t_i < t < t_{i+1}\,, \\ with\ the\ initial\ condition\ \ \boldsymbol{y}(t_i) = \theta_i \end{array}\right\} \qquad (2)$$

for $0 \leq i \leq N-1$, subject to the boundary and continuity conditions

$$\left.\begin{array}{l} \beta(\theta_0,\ \theta_N) = 0\,.\ where\ \theta_N = \boldsymbol{y}(1^-)\ , \\ \boldsymbol{y}(t_i^-) = \theta_i\ ,\quad for\ \ 0 \leq i \leq N-1\,, \\ and\quad A_i(t_{i+1}^-)\,\theta_{i+1} + \mathbf{b}_i(t_{i+1}^-) = \\ \qquad A_{i+1}(t_{i+1}^+)\,\theta_{i+1} + \mathbf{b}_{i+1}(t_{i+1}^+)\ , \\ \qquad for\ \ 0 \leq i \leq N-2\,. \end{array}\right\} \qquad (3)$$

minimizes the error defined by

$$\boldsymbol{\mathcal{E}} = \sum_{i=0}^{N-1} \int_{t_i}^{t_{i+1}} \mathcal{G}\big(A_i(\boldsymbol{t})\boldsymbol{y}(t) + \mathbf{b}_i(t) - \mathbf{f}(\boldsymbol{y}(t),t)\big)\ \mathrm{d}t \qquad (4)$$

where the integrand $\mathcal{G}$ can be chosen to be any nonnegative continuous function defined on $\mathbf{R}^n$ and equal to 0 at the origin.

We first consider a special case of (1), in which the boundary conditions are variables separable (i.e. of the form $\beta_0\big(x(0)\big) = \beta_1\big(x(1)\big) = 0$, where $\beta_0 : \mathbf{R}^n \to \mathbf{R}$ and $\beta_1 : \mathbf{R}^n \to \mathbf{R}$ are continuous functions. Later, we shall indicate the modifications needed to extend the method to the general case of boundary conditions described in (1).

Piecewise Affine Approximation for the Special Case

In this subsection, we consider the boundary value problem

$$\left.\begin{array}{l} \frac{dx}{dt} = \boldsymbol{f}(\boldsymbol{x}(t),\ t),\quad for\ \ 0 < t < 1, \\ \quad and\quad \beta\big(x(0)\big) = \beta\big(x(1)\big) = 0 \end{array}\right\} \qquad (5)$$

where $\boldsymbol{f}$, β_0 and β_1 are as just descried. The objective in the sequel is to find optimal values for the parameters of form $\pi_i = (A_i(t),\ \mathbf{b}_i(t),\ \theta_i)$, where $A_i(t)$ and $\mathbf{b}_i(t)$ are $n \times n$ and $n \times 1$ matrices, respectively, consisting of undetermined linear combination of some fixed basis functions defined on the interval $[t_i,\ t_{i+1}]$ and θ_i is an $n \times 1$ column vector of undetermined constants, for $0 \leq i \leq N-1$, for which the error functional in (4) is minimized, by the solution $\mathbf{y}(t)$ for (2), subject to the following boundary and continuity conditions:

$$\left.\begin{array}{l} \beta_0(\theta_0) = \beta_1(\theta_N) = 0\,.\ where\ \theta_N = \boldsymbol{y}(1^-)\ , \\ \boldsymbol{y}(t_i^-) = \theta_i\ ,\quad for\ \ 0 \leq i \leq N-1\,, \\ and\quad A_i(t_{i+1}^-)\,\theta_{i+1} + \mathbf{b}_i(t_{i+1}^-) = \\ \qquad A_{i+1}(t_{i+1}^+)\,\theta_{i+1} + \mathbf{b}_{i+1}(t_{i+1}^+)\ , \\ \qquad for\ \ 0 \leq i \leq N-2. \end{array}\right\} \qquad (6)$$

The solution for (2) in the interval $[t_i\,,\ t_{i+1}\,)$, for $0 \le i \le N-1$, is as follows :

$$\boldsymbol{y}(t) = e^{\left[\int_{t_i}^{t} A_i(v)\,\mathrm{d}v\right]}\theta_i + \int_{t_i}^{t} e^{\left[\int_{s}^{t} A_i(v)\,\mathrm{d}v\right]}\mathbf{b}_i(s)\,\mathrm{d}s \quad (7)$$

which is expressible explicitly in terms of the parameter values. Substituting the expression for $\boldsymbol{y}(t)$ in (4), the value of the error $\boldsymbol{E}$ incurred in the approximation can be found for any prescribed values of the parameters. If the parameter space is discretized, the values of $\boldsymbol{\mathcal{E}}$ can be tabulated for various values of the parameters, and, by the additive property of the error functional $\boldsymbol{\mathcal{E}}$, a dynamic programming method for obtaining the tables, quickly and efficiently, can be formulated as follows: let $\boldsymbol{E_j}$, be the associated j -step error functional satisfying $\boldsymbol{E_0} = \mathbf{0}$, and $\boldsymbol{E_j} = E_{\boldsymbol{j-1}} + \varepsilon_j$, for $1 \le j \le N$, where

$$\varepsilon_j = \int_{t_{j-1}}^{t_j} \mathcal{G}\left(A_{j-1}(\boldsymbol{t})\boldsymbol{y}(t) + \mathbf{b}_{j-1}(t) - \mathbf{f}(\boldsymbol{y}(t), t)\right)\mathrm{d}t \quad (8)$$

Thus $\boldsymbol{E}_N = \boldsymbol{\mathcal{E}}$, where $\boldsymbol{\mathcal{E}}$ is the error functional defined in (4). The functional $\boldsymbol{E_j}$ depends only on the parameters $(\pi_0,\ \pi_1,\ \ldots\ ,\ \pi_{j-1})$, for $1 \le j \le N$. For explicating this dependency, we sometimes write $\boldsymbol{E}_j(\pi_0,\ \pi_1,\ \ldots\ ,\ \pi_{j-1})$ for $\boldsymbol{E}_j$, and similarly $\varepsilon_j(\pi_{j-1})$ for ε_j. Now, let Ω_j be the set of parameters $(\pi_{j-1}\,,\ \pi_j)$ satisfying the following conditions:

$$\left.\begin{array}{c} \beta_0(\theta_0) = 0\,,\ if\ j = 1\,;\ \ \beta_1(\theta_N) = 0\,,\ if\ j = N\,, \\ where\ \ \theta_N = \boldsymbol{y}(1^-)\,; \\ \boldsymbol{y}(t_{j-1}^-) = \theta_{j-1}\,,\quad for\ 1 \le j \le N\,,\ \ and \\ A_{j-1}(t_j^-)\,\theta_{j-1} + \mathbf{b}_{j-1}(t_j^-) = A_j(t_j^+)\,\theta_j + \mathbf{b}_j(t_j^+), \\ for\ 1 \le j \le N-1. \end{array}\right\} \quad (9)$$

where, if $j = N$, the parameter π_N is interpreted to be simply θ_N , and $\boldsymbol{y}(t)$ in the interval $[t_{j-1}\,,\ t_j\,)$ is given in (7), taking $i = j-1$ there. Let

$$\mathbb{\Gamma}_j = \left\{(\pi_0\,,\ \pi_1\,,\ \ldots\,,\ \pi_j) : (\pi_{i-1}\,,\ \pi_i) \in \Omega_i,\ 1 \le i \le j\right\} \quad (10)$$

Now, we define a finite sequence of functions on the parameter space as follows: let $S_0\,(\pi_0) = 0$ and, for $1 \le j \le N$,

$$S_j(\pi_j) = \min_{\substack{(\pi_0,\ \pi_1,\ \ldots\ ,\ \pi_{j-1}):\\ (\pi_0,\ \pi_1,\ \ldots\ ,\ \pi_{j-1},\ \pi_j)\\ \in\ \mathbb{\Gamma}_j}} \left\{\boldsymbol{E}_j(\pi_0,\ \pi_1,\ \ldots\ ,\ \pi_{j-1})\right\} \quad (11)$$

and observe that for each fixed π_j, where $1 \le j \le N$, $S_j(\pi_j)$ satisfies the following recurrence relation:

$$S_j(\pi_j) = \min_{\substack{(\pi_{j-1}):\\ (\pi_{j-1,}\,\pi_j)\ \in\ \Omega_j}} \left\{\ S_{j-1}(\pi_{j-1}) + \varepsilon_j(\pi_{j-1})\ \right\} \quad (12)$$

To see how the recurrence relation in (12) is obtained, the following sequence of steps is considered:

$$S_j(\pi_j) = \min_{\substack{(\pi_0,\ \pi_1,\ \ldots\ ,\ \pi_{j-1}):\\ (\pi_0,\ \pi_1,\ \ldots\ ,\ \pi_{j-1},\ \pi_j)\\ \in\ \mathbb{\Gamma}_j}} \left\{\boldsymbol{E}_j(\pi_0,\ \pi_1,\ \ldots\ ,\ \pi_{j-1})\right\}$$

$$= \min_{\substack{(\pi_0,\ \pi_1,\ \ldots\ ,\ \pi_{j-1}):\\ (\pi_0,\ \pi_1,\ \ldots\ ,\ \pi_{j-1},\ \pi_j)\\ \in\ \mathbb{\Gamma}_j}} \left\{\boldsymbol{E}_{j-1}(\pi_0,\ \pi_1,\ \ldots\ ,\ \pi_{j-2}) + \varepsilon_j(\pi_{j-1})\ \right\}$$

$$= \min_{\substack{(\pi_{j-1}):\\ (\pi_{j-1,}\,\pi_j)\ \in\ \Omega_j}} \left\{ \min_{\substack{(\pi_{j-1}):\\ (\pi_{j-1,}\,\pi_j)\ \in\ \Omega_j}} \left\{\ \varepsilon_j(\pi_{j-1})\ \right\} + \min_{\substack{(\pi_0,\ \pi_1,\ \ldots\ ,\ \pi_{j-2})\\ (\pi_0,\ \pi_1,\ \ldots\ ,\ \pi_{j-1})\ \in\ \mathbb{\Gamma}_{j-1}}} \left\{\boldsymbol{E}_j(\pi_0,\ \pi_1,\ \ldots\ ,\ \pi_{j-1})\right\} \right\}$$

$$= \min_{\substack{(\pi_{j-1}):\\ (\pi_{j-1,}\,\pi_j)\ \in\ \Omega_j}} \left\{\ \varepsilon_j(\pi_{j-1}) + S_{j-1}(\pi_{j-1})\right\}$$

which proves (12). The value of

$$\min_{\substack{(\pi_N):\\ (\pi_{N-1,}\,\pi_N)\ \in\ \Omega_N}} \left\{\ S_N(\pi_N)\ \right\} = \min_{\substack{(\theta_N):\\ \beta(\theta_N) = 0}} \left\{\ S_N(\theta_N)\ \right\}$$

is the minimum value of $\boldsymbol{E}_N$ over the parameter space constrained by the boundary and continuity conditions (6). Now, we describe a tabulation procedure for computation of optimal parameters, consisting of two passes: in the forward pass, the tables are constructed, saving optimal parameter vales for backtracking, and in the backward pass, the optimal parameter values are recovered.

Forward Pass (Tabulation of Parameters)

1. Initially, let $S_0(\pi_0) = 0$, for all discrete values of π_0
2. Now, compute , compute $S_j(\pi_j) = \min_{\substack{(\pi_{j-1}):\\ (\pi_{j-1,}\,\pi_j)\ \in\ \Omega_j}} \left\{\ S_{j-1}(\pi_0\,,\ \pi_j) + \varepsilon_j(\pi_{j-1})\ \right\}$ and set $P_{j-1}(\pi_j) = \{\pi_{j-1} : (\pi_{j-1}\,,\ \pi_j) \in \Omega_j\ \ and\ \ S_{j-1}(\pi_0\,,\ \pi_{j-1}) + \varepsilon_j(\pi_{j-1}) = S_j(\pi_0\,,\ \pi_j)\}$, for $1 \le j \le N$.

At the end of the tabulation procedure, we obtain a table for $S_N(\pi_N)$, for various values of π_N (which is θ_N). An optimal sequence of values for the parameters is found by tracing backward, as described next.

Backward Pass (Recovery of Optimal Parameters)

1. Find $\pi_N^* = \min_{\substack{(\pi_N):\\ (\pi_{N-1,}\,\pi_N)\ \in\ \Omega_N}} \left\{\ S_N(\pi_N)\ \right\}$

2. Now, chose π_{j-1}^* to be an element from the set $P_{j-1}(\pi_j^*)$, for $j = N,\ N-1,\ \ldots,\ 1$.

An optimal parameter sequence is $(\pi_0^*,\ \pi_1^*,\ \ldots,\ \pi_N^*)$, with respect to the error functional in (4), subject to the boundary and continuity conditions in (6).

PiecewiseAffine Approximation for the General Case

In this subsection, we describe a generalization of the method of the previous subsection to the general case of boundary conditions in (3). Let Ω_j be the set of parameters (π_{j-1}, π_j) satisfying the following conditions:

$$\left.\begin{array}{c} \exists\, \xi : \beta(\theta_0,\ \xi) = 0\ ,\ \ if\ j = 1\,; \\ \exists\, \xi : \beta(\xi,\ \theta_N) = 0\ ,\ \ if\ j = N\,, \\ where\ \ \theta_N = \boldsymbol{y}(1^-)\,; \\ \boldsymbol{y}(t_{j-1}^-) = \theta_{j-1}\,,\ \ for\ 1 \le j \le N\,,\ and \\ A_{j-1}(t_j^-)\ \theta_{j-1} + \mathbf{b}_{j-1}(t_j^-) = A_j(t_j^+)\,\theta_j + \mathbf{b}_j(t_j^+), \\ for\ 1 \le j \le N-1\,. \end{array}\right\} \quad (13)$$

where, if $j = N$, the parameter π_N is simply θ_N, and $\boldsymbol{y}(t)$ in the interval $[t_{j-1},\ t_j)$ is given in (7), taking $i = j-1$ there, and let $\mathbb{\Gamma}_j$ be as defined in (10), but with Ω_j as given in (13), for $1 \le j \le N$. Now, the parameter space constrained by (3) is given by $\mathbb{\Gamma}_N \cap \boldsymbol{\Delta}_N$, where $\boldsymbol{\Delta}_N = \{(\pi_0,\ \pi_1,\ \ldots,\ \pi_N) : \beta(\theta_0,\ \theta_N) = 0\}$. However, the j-step objective function value S_j depends on $(\pi_0,\ \pi_j)$, for $1 \le j \le N$ as follows : let $S_1(\pi_0, \pi_1) = \varepsilon_1(\pi_0)$, for $(\pi_0, \pi_1) \in \Omega_1$, and, for $2 \le j \le N$, let

$$S_j(\pi_0, \pi_j) = \min_{\substack{(\pi_1,\ \ldots,\ \pi_{j-1}):\\ (\pi_1,\ \pi_1,\ \ldots,\ \pi_{j-1}\ \pi_j)\\ \in\ \mathbb{\Gamma}_j}} \{\boldsymbol{E}_j(\pi_0,\ \pi_1,\ \ldots,\ \pi_{j-1})\}$$

$$= \min_{\substack{(\pi_{j-1}):\\ (\pi_{j-1},\ \pi_j)\ \in\ \Omega_j}} \{\ S_{j-1}(\pi_0, \pi_j) + \varepsilon_j(\pi_{j-1})\ \} \quad (14)$$

so that $\min_{\substack{(\pi_N):\\ \beta(\theta_N) = 0}} \{\ S_N(\theta_N)\ \} =$

$$\min_{\substack{(\pi_0, \pi_1,\ \ldots,\ \pi_{N-1}):\\ (\pi_0, \pi_1,\ \ldots,\ \pi_{N-1},\ \pi_N)\ \in\ \mathbb{\Gamma}_N \cap \boldsymbol{\Delta}_N}} \{\ \boldsymbol{E}_N(\pi_0, \pi_1,\ \ldots,\ \pi_{N-1})\ \}$$

Forward Pass (Tabulation of Parameters)

1. Initially, compute $S_1(\pi_0, \pi_1) = \varepsilon_1(\pi_0)$, for various values of $(\pi_0, \pi_1) \in \Omega_1$, and set $P_0(\pi_0, \pi_1) = \{\ \pi_0 : (\pi_0, \pi_1) \in \Omega_1\ \}$

2. Now, compute $S_j(\pi_0, \pi_j) =$

$$\min_{\substack{(\pi_{j-1}):\\ (\pi_{j-1},\ \pi_j)\ \in\ \Omega_j}} \{\ S_{j-1}(\pi_0, \pi_j) + \varepsilon_j(\pi_{j-1})\ \},$$

and set

$$P_{j-1}(\pi_0, \pi_j) = \{\pi_{j-1} : (\pi_{j-1}, \pi_j) \in \Omega_j\ \ and\ \ S_{j-1}(\pi_0, \pi_{j-1}) + \varepsilon_j(\pi_{j-1}) = S_j(\pi_0, \pi_j)\},$$

for $2 \le j \le N$.

At the end of the tabulation procedure, we obtain a table for $S_N(\pi_0, \pi_N)$, for various values of (π_0, π_N). An optimal sequence of values for the parameters is found by tracing backward, as described next.

Backward Pass (Recovery of Optimal Parameters)

1. Find $(\pi_0^*,\ \pi_N^*) = \min_{\substack{(\pi_0,\ \pi_N)\\ \beta\ (\theta_0,\ \theta_N) = 0}} \{\ S_N(\pi_0, \pi_N)\}$

2. Now, chose π_{j-1}^* to be an element from the set $P_{j-1}(\pi_0^*,\ \pi_j^*)$, for $j = N,\ N-1,\ \ldots,\ 1$.

An optimal parameter sequence is $(\pi_0^*,\ \pi_1^*,\ \ldots,\ \pi_N^*)$ with respect to the error functional in (4), subject to the boundary and continuity conditions in (3).

Improving an Initial Approximate Solution

In this section, we describe a method similar to Newton-Raphson method for improving an initial approximation for a boundary value problem. The method computes an improved solution for an initial value problem. The correction of the approximation consists of two parts: in the first part, a correction term with 0 initial value for improving the solution in the interior is obtained, and the second part finds an optimal correction term in the initial value, so that the updated solution satisfies the boundary conditions more accurately. Alternatively, it is also possible to achieve the same by either Picard's successive approximation (with suitable correction term in the initial value) or repeated application of the dynamic programming method described in the last section with finer discretization of the parameter space restricted to a tube-like set around the initial approximate solution.

Improving an Initial Approximate Solution with Initial Value Held Fixed

We assume that the vector field $\boldsymbol{f}$ is differentiable with continuous derivatives of upto as high an order (upto two) as necessary. Let $\boldsymbol{x}_0(t)$ be an initial approximate solution satisfying

$$\left.\begin{array}{c} \frac{d\boldsymbol{x}_0}{dt} \approx \boldsymbol{f}(\boldsymbol{x}_0(t),\ t)\,,\ \ for\ \ 0 < t < 1, \\ \text{with the initial condition}\ \ \boldsymbol{x}_0(0) = \theta_0 \end{array}\right\} \quad (16)$$

The objective is to formulate an efficient method for improving the approximation in (16) by successive iterations. Let, for $k = 0, 1, 2, \ldots$,

$$\left.\begin{array}{c} \boldsymbol{x}_k'(t) = \frac{\mathrm{d}\boldsymbol{x}_k(t)}{\mathrm{dt}} \approx \boldsymbol{f}(\boldsymbol{x}_k(t),\ t)\,,\ \ for\ \ 0 < t < 1, \\ \text{with the initial condition}\ \ \boldsymbol{x}_k(0) = \theta_0 \end{array}\right\} \quad (17)$$

During the iteration number $k+1$, the correction term $\boldsymbol{y}_k(t) = \boldsymbol{x}_{k+1}(t) - \boldsymbol{x}_k(t)$ must be found such that the following holds in the ideal case scenario :

$$\left.\begin{array}{c} \boldsymbol{x}'_{k+1}(t) \;=\; \frac{\mathrm{d}\boldsymbol{x_{k+1}}(t)}{\mathrm{dt}} \;=\; \boldsymbol{f}(\boldsymbol{x}_{k+1}(t),\ t), \\ for \quad 0 < t < 1, \\ \text{with the initial condition} \;\; \boldsymbol{x}_{k+1}(0) = \theta_0 \end{array}\right\} \quad (18)$$

Subtracting (17) from (18), we find that $\boldsymbol{y}_k(t)$ must satisfy

$$\left.\begin{array}{c} \boldsymbol{y}'_k(t) \;=\; \frac{\mathrm{d}\boldsymbol{y}_k(t)}{\mathrm{dt}} \;=\; \boldsymbol{f}(\boldsymbol{x}_{k+1}(t),\ t) - \boldsymbol{x}'_k(t)\,, \\ for \quad 0 < t < 1, \\ \text{with the initial condition} \;\; \boldsymbol{y}_k(0) = 0 \end{array}\right\} \quad (19)$$

Now, using the Taylor series approximation with respect to the first variable $\boldsymbol{x}$ upto the first order, for $\boldsymbol{f}(\boldsymbol{x}_{k+1}(t),\ t) = \boldsymbol{f}(\boldsymbol{x}_k(t) + \boldsymbol{y}_k(t),\ t)$, for each fixed t, we find that

$$\left.\begin{array}{l} \boldsymbol{f}(\boldsymbol{x}_{k+1}(t),\ t) \;\approx\; \boldsymbol{f}(\boldsymbol{x}_k(t),\ t) + C_k(\mathrm{t})\boldsymbol{y}_k(t)\,, \\ \text{where} \;\; C_k(\mathrm{t}) \;=\; \left[\ \boldsymbol{\partial_x} \boldsymbol{f}(\boldsymbol{x},\ s)\ \right]_{\substack{s=t \\ x\,=\,x_k(t)}} \end{array}\right\} \quad (20)$$

Writing $\boldsymbol{z}_k(t) = \boldsymbol{f}(\boldsymbol{x}_k(t),\ t) - \boldsymbol{x}'_k(t)$ and taking (20) as an equation, we can rewrite (19) as follows :

$$\left.\begin{array}{c} \boldsymbol{y}'_k(t) \;=\; \frac{\mathrm{d}\boldsymbol{y}_k(t)}{\mathrm{dt}} \;=\; C_k(\mathrm{t})\boldsymbol{y}_k(t) + \boldsymbol{z}_k(t)\,, \\ for \quad 0 < t < 1, \\ \text{with the initial condition} \;\; \boldsymbol{y}_k(0) = 0 \end{array}\right\} \quad (21)$$

The solution for (21) is as follows :

$$\boldsymbol{y}_k(t) \;=\; \int_0^t e^{\left[\int_s^t C_k(\mathrm{v})\,\mathrm{d}v\right]} \boldsymbol{z}_k(t)\ \ \mathrm{d}\,s \quad (22)$$

The iteration converges fast (at almost quadratic rate) provided the initial approximate solution $\boldsymbol{x}_0(t)$ in (16) is sufficiently close to the exact solution. If the matrix $C_k(\mathrm{t})$ is invertible with a bounded inverse in the interval [0, 1], uniformly for various possible choices of $\boldsymbol{x}_k(t)$ and t, then the correction term $\boldsymbol{y}_k(t)$ can also be taken as follows:

$$\boldsymbol{y}_k(t) \;=\; -\,C_k^{-1}(\mathrm{t})\,\boldsymbol{z}_k(t) \quad (23)$$

which is the well-known Newton method for solving for $\boldsymbol{x}$ from $\boldsymbol{f}(\boldsymbol{x},\ t) = 0$, for each $t \in [0,1]$. This method of computing from (23) also modifies the initial condition, with the result that $\boldsymbol{y}_k(0)$ so obtained from (23) may not be equal to 0. If the initial value is required to be updated independently, then we have to find $\boldsymbol{y}_k(t)$ from (21), with its solution given by (22). It may be observed that when either iteration with update terms defined in (22) and (23) converges, the following holds : $\lim_{k\to\infty} \boldsymbol{z}_k(t) = 0$, almost everywhere (a. e.) , for $t \in [0,1]$. Thus, if convergent, either iteration leads to the final solution satisfying $\mathbf{x}'(t) = (\mathbf{x}(t),\ t)$, where $\mathbf{x}'(t)$ is the derivative of $\mathbf{x}(t)$, a. e, for $t \in [0,1]$.

Improving the Initial Value

Suppose that the initial value at $t = 0$ of the approximate correction term $\boldsymbol{y}_k(t)$ is $\boldsymbol{\eta}_k$. Then, the correction term is $\boldsymbol{y}_k(t\,;\,\boldsymbol{\eta}_k)$, where

$$\boldsymbol{y}_k(t\,;\,\boldsymbol{\eta}_k) \;=\; e^{\left[\int_0^t C_{k_i}(v)\,\mathrm{d}v\right]}\boldsymbol{\eta}_k \;+\; \int_0^t e^{\left[\int_s^t C_k(v)\,\mathrm{d}v\right]}\boldsymbol{z}_k(s)\,\mathrm{d}s \quad (24)$$

where $C_k(\mathrm{t})$ is as in (20) and $\boldsymbol{z_k}(t)$ as in the following line. The parameter $\boldsymbol{\eta}_k$ needs to be found based on an optimality criterion. The objective can be formulated as follows:

$$\boldsymbol{\eta}_k^* \;=\; \underset{(\boldsymbol{\eta})}{\operatorname{argmin}}\, \mathcal{F}(\boldsymbol{\eta})\,, \quad \text{where}$$

$$\mathcal{F}(\boldsymbol{\eta}) = \int_0^1 \left[\ \|\boldsymbol{y}_k(t\,;\,\boldsymbol{\eta})\|^2 + \left\|\boldsymbol{y}'_k(t\,;\,\boldsymbol{\eta})\right\|^2\ \right] \mathrm{d}t \quad (25)$$

If the initial approximation is sufficiently accurate at $t = 0$, then $\boldsymbol{\eta}_k^*$ must be small. Thus, we can expect that the minimum of $\mathcal{F}(\boldsymbol{\eta})$ in (25) is attained for $\boldsymbol{\eta} \approx \mathbf{0}$. The update value can be chosen to be proportional to the gradient of the gradient of $\mathcal{F}(\boldsymbol{\eta})$ at $\boldsymbol{\eta} = \mathbf{0}$. Specifically, we can choose $\boldsymbol{\eta}_k^* = -\,\boldsymbol{h}_k\,\boldsymbol{\nabla}\mathcal{F}(\mathbf{0})$, for a small positive number $\boldsymbol{h}_k$, resulting in an easy updation of the initial value. The multiplier $\boldsymbol{h}_k$ can be found by binary search method over an interval of the form $[0\,,\delta]$, for some positive constant δ. Alternately, it is also possible to find $\boldsymbol{\eta}_k^*$ by solving $\boldsymbol{\nabla}\mathcal{F}(\boldsymbol{\eta}) = \mathbf{0}$, by the iterative formula for $i = 0, 1, 2, \ldots$,

$$\boldsymbol{\gamma}_0 = 0\,, \;\; \boldsymbol{and} \;\; \boldsymbol{\gamma}_{i+1} = \boldsymbol{\gamma}_i - \mathcal{H}^{-1}(\boldsymbol{\gamma}_i)\,\boldsymbol{\nabla}\mathcal{F}(\boldsymbol{\gamma}_i) \quad (26)$$

where $\mathcal{H}(\boldsymbol{\eta})$ is the Hessian matrix of $\mathcal{F}(\boldsymbol{\eta})$, and $\boldsymbol{\eta}_k^*$ is chosen to be $\lim_{i\to\infty} \boldsymbol{\gamma}_i$. The method is fast and does not require a separate search for the multiplier constant as in the case of gradient descent method.

Combining Both : Improving and Initial Approximation by Successive Iterations

In this subsection, we briefly describe a gradient descent method for improving an initial solution of the boundary value problem (16). For this purpose, we assume that β is continuously differentiable having continuous derivatives upto Hessian. The update $\boldsymbol{y}_k(t)$ for the approximation $\boldsymbol{x}_k(t)$ in the $(k+1)$-th iteration is given in (24), with initial condition $\boldsymbol{y}_k(0) = \boldsymbol{\eta}_k$ and $\boldsymbol{y}_k(t)$ satisfying the differential equation in (21), for $0 < t < 1$. The initial value $\boldsymbol{\eta}_k$ of $\boldsymbol{y}_k(t)$ at $t = 0$ is determined by solving the boundary condition. Specifically, let $\boldsymbol{\rho}_k = \boldsymbol{x}_k(0)$ and $\boldsymbol{\sigma}_k = \boldsymbol{x}_k(1)$. The initial value $\boldsymbol{\rho}_{k+1} = \boldsymbol{x}_{k+1}(0)$ is $\boldsymbol{\rho}_k + \boldsymbol{\eta}_k$, and the final value $\boldsymbol{\sigma}_{k+1} = \boldsymbol{x}_{k+1}(1)$ is $\boldsymbol{x}_k(1) + \boldsymbol{y}_k(1)$. The updated boundary values $(\boldsymbol{\rho}_{k+1}\,,\ \boldsymbol{\sigma}_{k+1})$ in the $(k+1)$-th iteration must now satisfy, in the ideal case scenario, the following :

$$\beta\,(\boldsymbol{\rho_{k+1}}\,,\ \boldsymbol{\sigma}_{k+1}) \;=\; \mathbf{0} \quad (27)$$

Now, substituting $\boldsymbol{\rho}_{k+1} = \boldsymbol{\rho}_k + \boldsymbol{\eta}_k$ and $\boldsymbol{\sigma}_{k+1} = \boldsymbol{x}_k(1) + \boldsymbol{y}_k(1) = \boldsymbol{\sigma}_k + \boldsymbol{y}_k(1)$ in (27) with $\boldsymbol{y}_k(1)$ found from (24), taking $t = 1$ there , the following is obtained :

$$\beta\Big(\ \boldsymbol{\rho}_k + \boldsymbol{\eta}_k\ ,\ \boldsymbol{\sigma}_k + e^{\left[\int_0^t c_{k_i}(v)\,\mathrm{d}v\right]}\boldsymbol{\eta}_k + \int_0^t e^{\left[\int_s^t c_k(v)\,\mathrm{d}v\right]}\boldsymbol{z}_k(s)\,\mathrm{d}s\ \Big) = \mathbf{0} \quad (28)$$

For finding a best possible vector $\boldsymbol{\eta}_k^*$ satisfying (28) , we propose first a gradient descent method for minimization of β^2. Assuming that $\boldsymbol{\eta}_k$ is small, we can evaluate the gradient of β in (28) with respect to $\boldsymbol{\eta}_k$, at $\boldsymbol{\eta}_k = \mathbf{0}$. Thus, the update in the initial value, $\boldsymbol{\eta_k}$, is chosen by the following single step update formula :

$$\boldsymbol{\eta}_k = -\boldsymbol{h}_k\,\beta(\boldsymbol{\rho}\,,\ \boldsymbol{\sigma})\Big[\boldsymbol{\partial}_{\boldsymbol{\rho}}\ \beta(\boldsymbol{\rho}\,,\ \boldsymbol{\sigma}) + e^{\left[\int_0^t c_{k_i}(v)\,\mathrm{d}v\right]}\boldsymbol{\partial}_{\boldsymbol{\sigma}}\ \beta(\boldsymbol{\rho}\,,\boldsymbol{\sigma})\Big] \quad (29)$$

where $\beta(\boldsymbol{\rho}\ ,\ \boldsymbol{\sigma})$ and its partial derivatives $\boldsymbol{\partial}_{\boldsymbol{\rho}}\ \beta(\boldsymbol{\rho}\ ,\ \boldsymbol{\sigma})$ and $\boldsymbol{\partial}_{\boldsymbol{\sigma}}\ \beta(\boldsymbol{\rho}\ ,\ \boldsymbol{\sigma})$ are evaluated for $\boldsymbol{\rho} = \boldsymbol{\rho_k}$ and $\boldsymbol{\sigma} = \boldsymbol{\sigma}_k + \int_0^t e^{\left[\int_s^t c_k(v)\,\mathrm{d}v\right]}\boldsymbol{z}_k(s)\,\mathrm{d}s$. The parameter $\boldsymbol{h_k}$ is a small multiplier that can be found, for example, by binary search method over an interval of the form $[0\,,\delta]$, for some positive constant δ. It is important to constrain $\boldsymbol{h}_k$ to be close to 0, since $\boldsymbol{\eta}_k$ must be restricted such that the curve $\boldsymbol{y}_k(t)$ never leaves a tube-like set that can be determined for convergence of the Newton-Raphson method. Alternately, it is also possible to find $\boldsymbol{\eta}_k$ such that $\nabla_{\boldsymbol{\eta}}\,\beta = \mathbf{0}$, where β is as in (28) , with β and $\nabla_{\boldsymbol{\eta}}$ β evaluated at $\boldsymbol{\eta} = \mathbf{0}$. In this case, the update in $\boldsymbol{\eta}_k$ is found by the following formula :

$$\boldsymbol{\eta}_k = -\mathcal{H}^{-1}(\boldsymbol{\rho}\,,\ \boldsymbol{\sigma})\Big[\boldsymbol{\partial}_{\boldsymbol{\rho}}\ \beta(\boldsymbol{\rho}\,,\ \boldsymbol{\sigma}) + e^{\left[\int_0^t c_{k_i}(v)\,\mathrm{d}v\right]}\boldsymbol{\partial}_{\boldsymbol{\sigma}}\ \beta(\boldsymbol{\rho}\,,\boldsymbol{\sigma})\Big] \quad (30)$$

where $\mathcal{H}(\boldsymbol{\rho}\ ,\ \boldsymbol{\sigma})$ is the Hessian matrix of $\beta(\boldsymbol{\rho}\ ,\ \boldsymbol{\sigma})$, with respect to $\boldsymbol{\eta}$. The function $\beta(\boldsymbol{\rho}\,,\ \boldsymbol{\sigma})$, its partial derivatives $\boldsymbol{\partial}_{\boldsymbol{\rho}}\ \beta(\boldsymbol{\rho}\ ,\ \boldsymbol{\sigma})$ and $\boldsymbol{\partial}_{\boldsymbol{\sigma}}\ \beta(\boldsymbol{\rho}\,,\ \boldsymbol{\sigma})$ and the Hessian $\mathcal{H}(\boldsymbol{\rho}\ ,\ \boldsymbol{\sigma})$ occurring in (30) are evaluated for $\boldsymbol{\rho} = \boldsymbol{\rho}_k$ **a**nd $\boldsymbol{\sigma} = \boldsymbol{\sigma}_k + \int_0^t e^{\left[\int_s^t c_k(v)\,\mathrm{d}v\right]}\boldsymbol{z}_k(s)\,\mathrm{d}s$. The method is fast and converges to the true boundary values provided the initial approximation is sufficiently accurate. It may be expected that the initial value update formulated in (29) is more accurate than that formulated in (30), besides being easier. The updates formulated in (29) and (30) can be performed as a single step, since the time duration updates in (22) and (23), for the correction term $\boldsymbol{y}_k(t)$ with initial value 0, are performed only once per iteration. The same holds for the correction term $\boldsymbol{y}_k(t)$ with updated initial value $\boldsymbol{y}_k(0) = \boldsymbol{\eta}_k$, and the iteration starts all over again with $\boldsymbol{x}_{k+1}(t)$ subsequently. This approach of constructing successive approximations is justified, when we observe that the initial approximate solution produced by the dynamic programming method satisfies the boundary conditions as accurately as can be specified.

Summary of Boundary Value Problem and its Extension to Optimal Control Problem and Multidimensional Cases

In this section, we summarize the dynamic programming method, with an error function using uniform metric. The dynamic programming method can be extended to a vary large class of metrics. We bring out the essential characteristic that is needed for formulation of a dynamic programming based discrete optimization method.

A special formulation for the optimal control problem is given. The optimal control problem for hybrid systems, where the dynamics may involve abrupt transitions, with possibly a different vector field in the new state, is stated. The transition conditions from one discrete state to another can be expressed similar to the boundary and continuity conditions. The major goal in the hybrid system control theory is to design the additive control function that can run the system in a viable condition, possibly with loops occurring among the discrete state transitions. The set of discrete states is finite. The viability conditions are required to be able to run the system forever, which can become a closed loop system. In the open system interface specification, there are entry and exit states, which must be followed correctly. The overall cost can be expressed as the sum of the costs within each discrete state and the costs incurred during each discrete transition, assuming that the system is enabled to take those discrete or continuous transitions. The optimal control problem for a hybrid system is as follows: among several possible viable alternatives, that permit the system to run indefinitely forever, construct one which may involve external input (which is the control), such that the cost of its execution is minimum. Common applications of this abstraction can be found in the design, management and scheduling of multilevel and multistage logistics, inventory, priority service of a queueing network, traffic and transportation control systems.

Towards the end of the section, we briefly indicate how to extend the dynamic programming method for the partial differential equations defined in the interior of a simply connected compact, preferably convex, domain, with a regular boundary together with prescribed conditions on it that a solution must satisfy. It is assumed that the boundary is prescribed by a covering as in an atlas. Applications of this method for solving partial differential equations can be found in remote sensing, spectroscopy and tomography, in which regions of physical matter of different permeating, penetrating, reflexivity or resistivity properties, that can affect a flow field, are estimated by the modeling parameters, using measurements taken at the surface and some interior points, where the measurements at these interior points may be assumed or default values, and comparing the reconstruction with another model, which is assumed to be free from anomalies.

Summary of Dynamic Programming Method for Boundary Value Problem

The method described in Section 2 can be applied with any nonnegative error functional $\boldsymbol{\mathcal{E}}$, for which the *j*-step error

functional $\boldsymbol{E_j}$ can be evaluated based on $\boldsymbol{E}_{j-1}$ and $\varepsilon_{\boldsymbol{j}}$, for for $2 \leq j \leq N$. In particular, for some function h defined on $[0, \infty) \times [0, \infty)$, if the following holds:

$$\boldsymbol{E}_j(\pi_0, \pi_1, \ldots, \pi_{j-1}) = h\left(\boldsymbol{E}_{j-1}(\pi_0, \pi_1, \ldots, \pi_{j-2}), \varepsilon_{\boldsymbol{j}}(\pi_{j-1})\right) \quad (31)$$

for $1 \leq j \leq N$, with specified $\boldsymbol{E_0}$, usually as 0, then the dynamic programming method can be still applied with the function g in stead of $+$. One of the important error functionals is with respect to the uniform metric, which takes the form

$$\boldsymbol{\mathcal{E}} = \max_{\substack{t_{i-1} \leq t \leq t_i \\ 0 \leq i \leq N-1}} \{ \| A_i(\boldsymbol{t})\boldsymbol{y}(t) + \mathbf{b}_i(t) - \mathbf{f}(\boldsymbol{y}(t), t) \| \} \quad (32)$$

The j-step error functional E_j for (32) is defined as follows: for $1 \leq j \leq N$,

$$\text{let } \varepsilon_j(\pi_{j-1}) = \max_{t_{j-1} \leq t \leq t_j} \left\{ \| A_{j-1}(\boldsymbol{t})\boldsymbol{y}(t) + \mathbf{b}_{j-1}(t) - \mathbf{f}(\boldsymbol{y}(t), t) \| \right\}$$

$$\text{and } \boldsymbol{E}_j(\pi_0, \pi_1, \ldots, \pi_{j-1}) = \max\{\boldsymbol{E}_{j-1}(\pi_0, \pi_1, \ldots, \pi_{j-2}), \varepsilon_{\boldsymbol{j}}(\pi_{j-1})\}$$

The error functional in (32) is best suited for obtaining an initial approximation, followed by the Newton-Raphson method of iterative improvement. With the error functional as defined in (32), the dynamic programming method can still be applied, repeatedly until the desired precision is achieved, using finer and finer discretization and restricting the search to only the tube-like set around the previously obtained optimal parameters. Decisions concerning which parameters (not necessarily optimal with respect to S_j) to retain in subsequent iterations can be made based on quantitative measures such as stiffness at the parameter value for identifying the tube set locally. A measure similar to stiffness is the spectrum of the matrix $C_k(\boldsymbol{t})$ defined in (20), which indicates as the iteration progresses which components move inward and which outward of the tube. The spectrum of $C_k(\boldsymbol{t})$ indicates the torsion and oscillatory properties of the solution. These measures can be evaluated at the parameter value from the vector field $\boldsymbol{f}$, without requiring complete or even part of the actual solution. Thus besides the tables for S_j, auxiliary tables containing information regarding stiffness or other measures can be used for choosing the parameter values in the dynamic programming algorithm. The usefulness of the auxiliary tables is especially significant when using larger discretization step-sizes. The auxiliary information can expedite the search method by eliminating unwanted parameter values and retaining only those parameter values that could actually produce the true solution, so that in the subsequent iterations, closer approximations (*i.e.*, with smaller error) are produced. The basic abstract model for the error functional in (31) can be recast in such a way that allowances for parameter dependences in the accumulating function h and dependence of the j-step cost function $\boldsymbol{E_j}$ on a future state to reach, which is described by the model parameter π_j, for $1 \leq j \leq N$, are explicated, as follows:

$$\boldsymbol{E}_j(\pi_0, \pi_1, \ldots, \pi_j) = h\left((\pi_0, \pi_1, \ldots, \pi_j), \boldsymbol{E}_{j-1}(\pi_0, \pi_1, \ldots, \pi_{j-1}), \varepsilon_{\boldsymbol{j}}(\pi_{j-1}, \pi_j)\right) \quad (33)$$

The additional model parameter sequence $(\pi_0, \pi_1, \ldots, \pi_j)$ in the arguments of the h-function in (33) allows the designer to take into consideration costs incurred due to lag or drag involved in the course from initial or past state up to until the current state, described by the model parameter π_{j-1} to reach the next state described by the model parameter π_j, for $1 \leq j \leq N$. In the cost function of a hybrid system, the single transition cost $\varepsilon_{\boldsymbol{j}}(\pi_{j-1})$ is replaced with $\varepsilon_{\boldsymbol{j}}(\pi_{j-1}, \pi_j)$, to signify that this cost may depend also on the future state π_j intended to reach from the current state π_{j-1}, for $1 \leq j \leq N$.

Extension for Optimal Control Problem

In this subsection, we describe a method to solve the optimal control problem, in which a control function $\mathbf{u}(t)$ that minimizes a cost function C, defined on the space of admissible or feasible control functions, must be found, subject to the following conditions :

$$\frac{\mathrm{dx}}{\mathrm{dt}} = \boldsymbol{f}(\mathrm{x}(t), t) + \mathrm{u}(t), \text{ for } 0 < t < 1, \quad \text{and} \quad \beta\big(x(0), x(1)\big) = 0$$

where $\boldsymbol{f}$ and β are as in Section 2.1. The cost functional C can be written as a sum of cost functionals for the intervals $[t_i, t_{i+1})$, for $0 \leq i \leq N-1$. In this situation, taking $\mathbf{u}(t) = A_i(\boldsymbol{t})\mathbf{x}(t) + \mathbf{b}_i(t) - \mathbf{f}(\mathbf{x}(t), t)$ on the interval $[t_i, t_{i+1})$, for $0 \leq i \leq N-1$, the dynamic programming method and improvements of approximations described in the previous section can be utilized to find an approximate optimal solution for the control function $\mathbf{u}(t)$.

The design of the control function $\mathbf{u}(t)$ turns out be more critical for hybrid systems, which combine the continuous and discrete state transitions [1]. For instance, in [15], the optimal control based formulation is used for estimation of sets reachable from an initial state. The boundary conditions in hybrid systems can encode the conditions required for maintenance and viability of the system. The control function can be modeled as a special input to the system, possibly involving human intervention, for maintenance of safety, while facilitating progress and viability of the system. The transition function $\boldsymbol{f}$ within a discrete state of a hybrid system may be determined by natural forces or engineered to meet a specific purpose. The cost of the control function can also vary with time and state, and the control function must adapt to particular situations, in which the system operates.

In hybrid systems, the time intervals are $0 = t_0 \leq t_1 \leq \ldots \leq t_N$, and the time instances t_0 and t_N correspond to entry into and exit from the system, respectively, for an open system interface, and initial and periodicity time instances, for a closed system interface. If $t_{j-1} = t_j$, for some index j, where $1 \leq j \leq N$, then a discrete state transition is understood to take effect at that instant. The

state variables consist of two components, which are one discrete and the other continuous states. Within a discrete state, the continuous dynamics are applicable to the continuous state, which are specific to the discrete state. There are transition guards enabling or maintaining a discrete transition, at some regions of the continuous states within a discrete state, for switching a discrete state transition, which is specified as part of the transition relation of a hybrid system. When a discrete transition is taken, the continuous state in the next discrete state is also specified as part of transition guard conditions, which may involve a cost for effecting the transition. It may be taken note of that the model parameter π_j contains the state information θ_j, which further contains the discrete state and continuous state components, for $1 \leq j \leq N$. The cost incurred for a transition from a system state described by the model parameter π_{j-1} to another state corresponding to the model parameter π_j is denoted by $\varepsilon_j(\pi_{j-1}, \pi_j)$, instead of just $\varepsilon_j(\pi_{j-1})$, and this cost is also additive, for the time interval specified by the end points t_{j-1} and t_j, but depends on the particular transition taken, assuming it to be enabled by the system. The overall system boundary conditions specified at time instances t_0 and t_N can be of either entry-exit form or viability form, satisfying periodicity conditions. It is convenient to specify end time instance of a course, or a run, of the system to be an indeterminate t_N, which is not restricted to be equal to the time instance 1. The number of time intervals N must be so chosen that a course of the system can traverse as many discrete states as needed by it during its transitions, and hence, it can vary with the course. There are additional constraints such as initialization, causality, fairness and replenishment conditions, which must be met, by a course, and that may hopefully reduce the otherwise infinitely many possible runs into a finitely many possible courses. In that case, the dynamic programming method followed by the continuous state updation can be utilized for finding an optimal control function, for a hybrid system. Although, we have just discussed the optimal control problem with the objective of minimizing a cost functional only, practical systems may prefer an objective that maximizes productivity of the system.

Extension for Boundary Value Problems for Partial Differential Equations Defined over a Simply Connected Compact Domain of the Euclidean Space

In this subsection, we describe a method to solve a boundary value problem of a partial differential equation, defined over a simply connected compact domain, which is a subset of $\mathbf{R}^d$, for some positive integer $d \geq 2$. For simplicity of description, the domain is further assumed to be convex, although this assumption is not always necessary. The boundary is assumed to be sufficiently regular and specified by smooth surfaces, parameterized as in an atlas, such as in the case of a sphere. The boundary is then propagated inward by, for example, computing the Euclidean distance from a point inside the domain. This approach is called boundary propagation or front propagation [9]. A change of coordinates, that is consistent with the description of the domain as consisting of concentric surfaces diffeomorphic to the initial boundary of the domain, is performed in the given partial differential equation. These surfaces are parametrically described in such a way that some geometric attribute remains constant for each surface, and hence they are called level sets of the propagating boundary. If the initial boundary is convex, with its unit normal pointing outward, then the boundary can be propagated inward by collecting points obtained by subtracting vectors normal to the boundary at the initial points on the boundary with small multipliers as absolute value from those points on the initial boundary, as in the gradient descent method, with varying multipliers. But this propagation may not eventually end in a single point, and, depending on the absolute values chosen as multipliers of the surface normal vectors, various shapes can be realized. In order to construct a set of concentric surfaces ending in a point, the Euclidean distance is computed from a point, which is termed as the center of the domain, to various points on the boundary, whose normal always points outwards, by assumption. A criterion for an interior point to be the center is stated as follows: let the distance of an interior point to the boundary be defined to be the maximum distance from it to any point on the boundary, and let the center be chosen to be the interior point that results in a minimum distance to the boundary among all interior points. This method of choosing the center is appropriate with compact convex regular boundaries and domains. Then, from the center, the distance to a point on the boundary is multiplied by a scale, as in the case of projective coordinate system, but the scale parameter is chosen to be a real number between 0 and 1. The level sets are the surfaces corresponding to the same scale. Now, appropriate dynamic programming tables are constructed that approximate a solution and its partial derivatives with respect to the changed coordinate system. In this case, the objective is to find a solution that agrees at the center of the domain, when approached from various directions, with minimum error.

Conclusions

In this paper, we have presented a dynamic programming based formulation for obtaining a best piecewise linear approximation for continuous vector fields with the solution constrained by arbitrary boundary conditions. The method attempts obtain a fast but reasonably accurate approximate solution, assuming discretized space. The parameters of the form $\pi_i = (A_i(t), \mathbf{b}_i(t), \theta_i)$, where $A_i(t)$ and $\mathbf{b}_i(t)$ contain undetermined linear combination of some fixed basis functions (such as polynomials) defined for approximation during the interval $[t_i, t_{i+1}]$, with undetermined initial value θ_i, for $0 \leq i \leq N-1$, are tabulated by the dynamic programming method, from which an optimal sequence of parameters is found by tracing backwards. The objective function for minimization of error is usually chosen to be the standard L^2-norm or can be chosen from among a large class of error functionals, including the uniform metric. The particular aspect of the error functional that allows a formulation of the dynamic programming method is usually a recurrence relation that an associated j-step functional satisfies. In the

proposed method, the *j*-step functionals satisfy a one-step recurrence relation. It is possible to formulate dynamic programming based methods also for error functionals satisfying more general recurrence relations, involving difficult parameter dependencies. A method for improving an initial approximation by successive iterations is presented, which can also update the initial and boundary values. The correction function in the interior of the interval is found by either gradient descent or Newton-Raphson method. The instant when to switch from the dynamic programming method to successive iterations method for improvement of the current solution can be determined based on the width of the convergence set for the Newton-Raphson method, when it is preferred instead of gradient descent method, for this purpose. It is also possible to consider taking a convex combination of formulated updations, for choosing the actual update. If any one of the updation formula always produces a more accurate solution than the other, then the convex combination degenerates into binary exclusive combination.

Possible extensions to formulation of solution methods for optimal control problem for hybrid systems and for boundary value problems of partial differential equations defined on compact simply connected convex domains are briefly described.

References

[1] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis and S. Yovine, "The algorithmic analysis of hybrid systems", *Theoretical Computer Science*, vol. 138(1), 1995, pp. 3-34

[2] Richard Bellman, and Robert E. Kalaba, "Dynamic programming and modern control theory", Academic Press, New York, 1965

[3] J. W. Daniel, "A road map of methods for approximating solutions of two point boundary value problems", in "Codes for boundary value problems in ordinary differential equations", Bart Childs, Melvin R. Scott, James W. Daniel, Eugene D. Denman, Paul Nelson (Eds.), *Proceedings of a Working Conference* May 1978, LNCS vol. 76, Springer Verlag, 1979.

[4] Bruce A. Finlayson, "The method of weighted residual and variational principles, with application in fluid mechanics, heat and mass transfer", *Mathematics in Science and Engineering*, vol. 87, Academic Press, 1972.

[5] Leslie Fox, "The numerical solution of two-point boundary problems in ordinary differential equations", Clarendon Press, 1957, reprinted by Dover Publications, 1990.

[6] C. M. García-López, and J. I. Ramos, "A piecewise-linearized method for ordinary differential equations: two-point boundary-value problems", International Journal for Numerical Methods in Fluids, vol. 22, Issue 11, June 1996, pp.1089-1102.

[7] Alan. C. Hindmarsh, Peter N. Brown, Keith E. Grant, Steven L. Lee, Radu Serban, Dan E. Shumaker and Carol S. Woodward, "SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers", *ACM Transactions on Mathematical Software*, Special issue on Advanced CompuTational Software (ACTS), Vol. 31, Issue 3, September 2005.

[8] Herbert B. Keller, "Numerical solution of two point boundary value problems", Dover Publications, 1992.

[9] R. Malladi, J. A. Sethian and B. C, Vemuri, "Shape modeling with front propagation: a level set approach", *IEEE Trans. on PAMI*, vol 17(2), 1995, pp. 158-175.

[10] C. P. Neuman and A. Sen, "Galerkin's procedure, quasilinearization, and nonlinear boundary-value problems", *Journal of Optimization Theory and Applications*, vol. 9(6), June 1972, pp. 433-437.

[11] L. S. Ramachandra and D. Roy, "A new method for nonlinear two-point boundary value problems in solid mechanics", Journal of Applied Mechanics, vol. 68(5), September 2001, pp. 776-786.

[12] M. R. Osborne, "On Shooting methods for boundary value problems", *J. Math. Anal. Appl.*, vol. 27(2), August 1969, pp. 417-433.

[13] J. I. Ramos, "Piecewise quasilinearization techniques for singular boundary-value problems", *Computer Physics Communications*, vol. 158(1), March 2004, pp. 12-25.

[14] Sanford M Roberts and Jerome S. Shipman, "Two-point boundary value problems: shooting methods", *Modern analytic and computational methods in science and mathematics*, 31, American Elsevier Pub. Co., First Edition, 1972..

[15] P. Varaiya, "Reach set computation using optimal control", *Proceedings of KIT Workshop*, Verimag, Grenoble, 1998.