cant reduction in convergence time. Further studies are needed to determine optimal reorganization strategies.

References

[1] S. Lakshmivarahan and M. A. L. Thathachar, "Absolutely expedient learning algorithms for stochastic automata," *IEEE Trans. on Syst., Man, Cybern.*, vol. SMC-3, pp. 281–286, 1973.

[2] G. J. McMurtry and K. S. Fu, "A variable structure automaton used as a multimodel searching technique," *IEEE Trans. Automat. Contr.*, vol. AC-11, pp. 379–387, 1966.

[3] B. T. Mitchell and D. I. Kountanis, "Optimal hierarchical learning automata structures," Dept. Computer Science, Western Michigan Univ. Tech. Rep., 1982.

[4] K. S. Narendra and S. Lakshmivarahan, "Learning automata– A critique," Cybernetics Informat. Sci., vol. 1, pp. 53–66, 1978.

[5] K. S. Narendra and M. A. L. Thathachar, "Learning automata—A survey," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-4, pp. 323–334, 1974.

[6] I. J. Shapiro and K. S. Narendra, "Use of stochastic automata for parameter self-optimization with multimodel performance criteria," *IEEE Trans. Syst., Sci., Cybern.*, vol. SCC-5, pp. 352–360, 1969.

[7] M. A. L. Thathachar and K. R. Ramakrishnan, "A hierarchical system of learning automata," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-11, pp. 236–241, 1981.

[8] M. L. Tsetlin, "On the behavior of finite automata in random media," *Avtomatika i Telemethanika*, vol. 22, pp. 1345–1354, 1961, *Automation and Remote Control*, vol. 22, pp. 1210–1219, 1961.

Tabular Control of Balance in a Dynamic Legged System

MARC H. RAIBERT, MEMBER, IEEE, AND
FRANCIS C. WIMBERLY

Abstract—**A method for control of dynamic systems that uses a large table of precomputed data is described. The method was developed in the context of controlling legged systems that balance as they run. In particular, a place to put each foot must be chosen by the control system in order to control tipping and forward running velocity. The tabular approach takes advantage of the very regular, cyclic character of legged behavior in order to partition the system state variables into two sets: one set that varies in a relatively fixed way on each cycle, and another set that varies freely from cycle to cycle. Only this second subset of the state variables determines the size of the table. Stored data were computed by numerically simulating a dynamic model of the legged system as it progressed through the stance portion of the running cycle. Repeated simulations were used to characterize the system for different landing conditions. Because the size of the table may be prohibitively large for some problems, polynomial surfaces were used to approximate the tabular data. Good approximations to the tabular data were obtained with just a few dozen terms. The feasibility of using the tabular and polynomial methods to control balance and forward running velocity in a planar system with one leg is shown by simulations.**

Introduction

Control algorithms that use well-organized tabular data offer the promise of providing good control for systems with complicated dynamics. Tabular techniques are powerful because they use the results of arbitrarily complicated calculations for control, but the time penalty of actually doing the calculation is incurred off-line. Therefore tabular methods typically involve very simple run-time computations that execute with high speed. In a comparison of techniques for computing robot manipulator dynamics, Hollerbach showed that a tabular method required the fewest run-time operations when applied to a manipulator with fewer than nine joints [1].

Another advantage of tabular control methods is that tables make it easy to implement simple forms of learning and adaptation. A tabular controller typically performs a very simple computation on the state variables to determine appropriate control values. The computation is based on a representation of the dynamics of the system to be controlled. Because the computations are simple, it is usually easy to determine values for the coefficients of the computation, provided the form of the control computation is already known. Learning occurs when values for the coefficients of the computation are determined from data obtained by observing the behavior of the system to be controlled [2]–[5].

The main problem with tabular control methods is that the size of the tables grows exponentially with the number of state variables and control inputs needed to characterize the dynamic system [6]. This problem has been attacked in a number of ways. Albus used a hashing function that mapped tables of astronomical size into the available memory of his computer in order to control a robot manipulator [2], [3]. His hashing functions were designed to use knowledge of the manipulator's dynamics in order to minimize hashing collisions. Hashing worked in that case because the controller, rather than having the potential of producing all possible motions, dealt only with the subset of manipulator motions that had been learned. Raibert and Horn reduced the size of the tables needed to control a manipulator by striking a balance between computation and tabularization [7]. They found that for most manipulators with n joints, an $n-1$ dimensional *configuration space* table would be sufficient. Simons *et al.* reduced the size of the tables they use to do manipulator force control by finding an optimum quantization of the state inputs [8].

In this correspondence we describe a tabular controller that maintains balance and regulates forward running speed in a locomotion system that hops on one leg. The task of finding a useful table of moderate size is accomplished, not by manipulating the form of the table, but by partitioning the problem into parts that can be solved separately. Once the problem is partitioned, the table deals with only a subset of the state variables. We use the stereotyped cyclic motion of the legged system to find a simple partitioning. We also show that multivariate polynomials of low degree can effectively approximate the tabular data. The polynomial requires substantially fewer data than the table, but somewhat more run-time computation. Data are presented that show the results of using both methods to control the one-legged hopping system in simulation.

The Problem

During hopping in place, placement of the foot on each step determines how a legged system will balance and it influences the system's translational velocity. Consider the planar one-legged system shown in Fig. 1. It has a rigid body, a springy leg, a hip driven by an actuator that provides torque τ, and a small foot. In [9] this system is described in detail and its equations of motion are presented. If the foot is placed to the left, then the system will tip and accelerate to the right. If the foot is placed to the right, then the vice versa. If the foot is placed directly under the body, then the system will neither tip nor accelerate. A corresponding set of rules applies when the system is travelling with a forward velocity. For each forward velocity there is a forward position for

Manuscript received April 11, 1983; revised September 1983. This work was supported in part by the Systems Science Office of the Defense Advanced Research Projects Agency under contract MDA 903-81-C-0130, and by a grant from the System Development Foundation.

The authors are with the Department of Computer Science and The Robotics Institute, Carnegie-Mellon University, Schenley Park, Pittsburgh, PA, 15213.

0018-9472/84/0300-0334$01.00 ©1984 IEEE

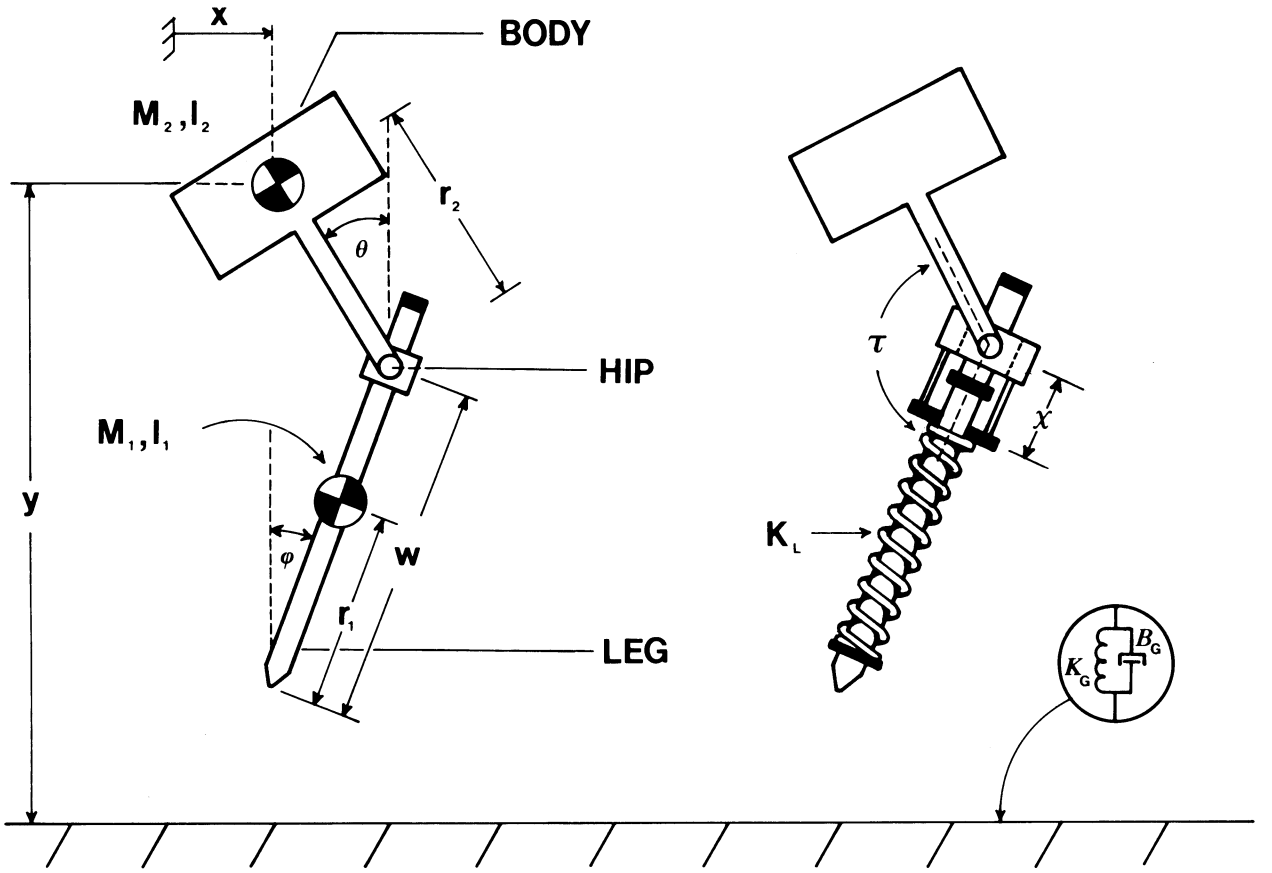

Fig. 1. Schematic diagram of planar one-legged system used to test tabular control. The body and leg, each having mass and moment of inertia, are connected by a hinge joint at which an actuator generates torque τ. The leg consists of a spring in series with a position actuator of length χ. Center of mass of the leg is located a distance r_1 from the lower tip of the leg, which is the foot. The body is represented by a rigid mass, with the center of mass located a distance r_2 above the hip. The angle of the leg φ, determines the relative position of the foot with respect to the system's center of gravity. Rhythmic activation of the leg actuator causes the system to leave the ground periodically in a hopping motion. Motion of the entire system is restricted to the plane. For more details and equations of motion, see [9].

the foot that will neither tip the system, nor change the rate of forward travel.

The effects of foot placement are important because position of the foot with respect to the body can be directly controlled by torquing the hip during flight, because the foot cannot be moved once placed, and because the foot's position strongly affects balance. For the present problem we think of the foot's position when the system first touches the ground, not as a state variable, but as a control input.

Once a cycle of stepping activity has been established, the problem of controlling balance and forward velocity is one of choosing a place to put the foot on each cycle that will take the system to the desired state. We are primarily concerned with the forward velocity and the attitude of the body. More specifically, the control task is to find a position for the foot before *touchdown*, the moment there is contact between the foot and the ground, so as to minimize state errors at *lift-off*, the moment the foot next leaves the ground. The state errors of interest are those in forward velocity $\dot{x}$, body angle θ, and body angular rate $\dot{\theta}$. Assume that during flight the leg angle will be adjusted by a linear servo of the form:

$$\tau(t) = K_P(\varphi - \varphi_d) + K_V(\dot{\varphi}) \tag{1}$$

where

φ_d Desired leg angle.
K_P, K_V Feedback gains.

Further assume that during stance the angle between the leg and the body, $(\varphi - \theta)$, is held constant by a linear servo similar to (1). We designate the value of a variable at touchdown by the subscript TD, and the value of a variable at lift-off by subscript LO. The problem to be solved is, given the state at touchdown $\boldsymbol{X}_{\mathrm{TD}}$, find φ_{TD} to minimize

$$\mathrm{PI} = Q_1(\dot{x}_{\mathrm{LO}} - \dot{x}_d)^2 + Q_2(\theta_{\mathrm{LO}} - \theta_d)^2 + Q_3(\dot{\theta}_{\mathrm{LO}} - \dot{\theta}_d)^2 \tag{2}$$

where

Q_1, Q_2, Q_3 Weights.
$\dot{x}_d, \theta_d, \dot{\theta}_d$ Desired values for forward velocity, body angle, and body angle rate.

Tabular Method

In order to minimize (2), a relationship Γ is needed that relates the state of the system at lift-off to the state at touchdown:

$$\boldsymbol{x}_{\mathrm{LO}} = \Gamma(\boldsymbol{x}_{\mathrm{TD}}) \tag{3}$$

where

$$\boldsymbol{x}_{\mathrm{LO}} = [\dot{x}_{\mathrm{LO}}, \theta_{\mathrm{LO}}, \dot{\theta}_{\mathrm{LO}}]$$
$$\boldsymbol{x}_{\mathrm{TD}} = [x, \dot{x}, y, \dot{y}, \theta, \dot{\theta}, w, \dot{w}, \varphi, \dot{\varphi}].$$

In general, behavior of the system during stance, the period between touchdown and lift-off, is influenced by the entire state vector at touchdown. The regular nature of the stepping cycle permits us to partition the state variables into two groups, those that vary from one stepping cycle to the next, and those that do

not. We assume that the values of y, $\dot{y}$, w, and $\dot{w}$ vary along the same trajectory from one cycle to the next. The values of these variables are important to the relationship expressed in (3), but their effect is nearly constant from hop to hop. Therefore, these variables need not appear as independent variables in (3), which can be expressed as a function of a subset of the state variables

$$x_{LO} = \Gamma(x'_{TD}) \tag{4}$$

where:

$$x'_{TD} = [\dot{x}_{TD}, \theta_{TD}, \dot{\theta}_{TD}, \varphi_{TD}].$$

We call the vector $x_{TD} = [\dot{x}_{TD}, \theta_{TD}, \dot{\theta}_{TD}]$ the touchdown state vector, $x'_{TD} = [\dot{x}_{TD}, \theta_{TD}, \dot{\theta}_{TD}, \varphi_{TD}]$ the augmented state vector, and $x_{LO} = [\dot{x}_{LO}, \theta_{LO}, \dot{\theta}_{LO}]$ the lift-off state vector. We define a vector field Λ, such that there is a dimension of Λ that corresponds to each component of x'_{TD}, and for each point in Λ there is a unique value of x_{LO}.

For this problem we think of φ as a control input, since it can be changed during flight at will, and the remaining components of x_{TD} as state. For the general problem there are i control inputs and n state variables.

The vector field Λ is approximated by a multidimensional table. One dimension of the table corresponds to each dimension of Λ, and all dimensions are quantized to M levels. For n variables and i control inputs, each quantized to M values, there are M^{n+i} hyperregions in the table, each storing an n vector. M must be chosen to quantize the table finely enough to capture the variations in x_{LO}. We have used such a table to control the planar hopper in simulation. Forward velocity $\dot{x}$, body angle θ, and body angular rate $\dot{\theta}$ are the state variables used to address the table, $n = 3$. Leg angle φ is a control input, $i = 1$. These variables index a four-dimensional space. Each dimension of the memory is quantized to nine levels, $M = 9$, requiring that $nM^{i+n} = 19\,683$ values be stored. To compensate for such a coarse quantization, the function that accesses the tabular data, $T(x'_{TD})$, performs a linear interpolation among the 2^{n+i} stored values that bound the desired value.

Tabular data were obtained by simulating a large set of locomotion cycles, with systematically varied initial conditions. For these simulations, the angle between leg and body was held constant during stance by the servo in (1), just as it would be when controlled. In order to minimize (2) the table was searched along a path determined by varying φ through its entire range, with $x = x_{TD}$. The details of the search are given in Appendix A. The leg was moved to the minimizing value of φ before touchdown.

This tabular controller was tested in simulation for a simple balance problem. The task was to return the one-legged system to a balanced posture, after starting with the body in an inclined position. Fig. 2 plots the body angle and horizontal position of the system for the test in which the system was dropped from a height of 0.3 m with an initial body angle error of 0.8 rad. Setpoints were $\dot{x}_d = 0$, $\theta_d = 0$, $\dot{\theta}_d = 0$. A vertical posture with no horizontal motion was attained in about 6 s. In this test no attempt was made to control horizontal position x.

The same algorithm was used to control forward velocity while the system traveled from one point to another. Fig. 3 shows data from the resulting translation in which position was controlled indirectly through rate control. Forward velocity was very low, but precisely controlled with no limit cycles like those caused by a linear controller used in previous experiments [10]. The low rate of forward travel was an artifact of the restricted motion of the hip during stance, as required by the simple foot placement algorithm. It is not an inherent attribute of the tabular control method.

In the example given here the control input, φ, was not explicitly varied during the interval between touchdown and lift-off. Hip angle was fixed during stance. In general it is not necessary that the control inputs be constant, only that they do not vary with more degrees of freedom than are represented in the table. This means that rich variations in the control signals are perfectly acceptable, provided that all variations are completely determined by the augmented state vector that is available when (3) is used.

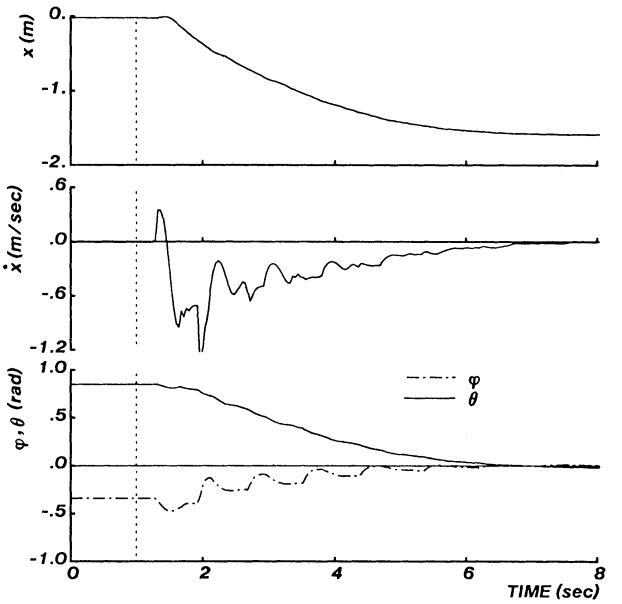

Fig. 2. Orientation of the body was corrected using tabular method. At $t = 0$ initial error in body attitude was 0.8 rad. At $t = 1$ the system was dropped from 0.3 m and hopping began (dotted line). State errors approached zero about 6 s later. Horizontal position was not controlled, so position changed without correction. ($Q_1 = 1.5$, $Q_2 = 5.0$, $Q_3 = 1.0$).

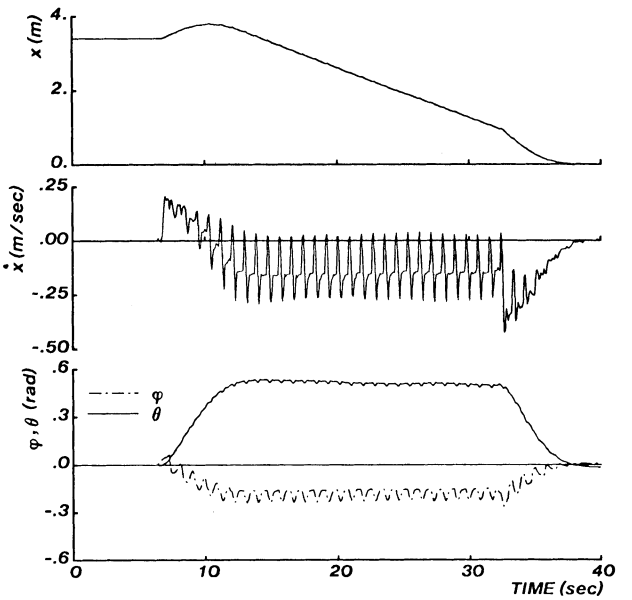

Fig. 3. Lateral step controlled by tabular method. ($Q_1 = 1.0$, $Q_2 = 5.0$, $Q_3 = 1.0$).

The table just described is used to evaluate (3). Control is actually accomplished by finding the value of φ_{TD} that minimizes (2) for given values of x_{TD} and x_d. Since only φ varies during this minimization, with x_{TD} and x_d fixed, only a one-dimensional search through the tabulated data is required to find the minimizing value.

A closed form minimization procedure is used within each quantized region of the tabular data, as described in Appendix I.

This procedure requires about $(n(M+1)-1)2^n + (M-1)(7n+1)$ multiplies for one control input, or about 408 multiplies per hop for the parameters used in this correspondence $n = 3$, $M = 9$.

In certain circumstances it is possible to create a table that need not be searched at run time to satisfy (2). This can be done when specific values for $\dot{x}_d$, θ_d, $\dot{\theta}_d$, Q_1, Q_2, and Q_3 are known at the time the table is created. For this special case control can proceed without a run-time minimization. We did not experiment with this idea.

Polynomial Approximation to Tabular Data

The data of the last section show that the tabular method can effectively control a nonlinear dynamic system with few state variables and control inputs. However, even when the problem is partitioned, the memory requirements for this approach become severe in larger applications. In this section we show that the tabular data can be approximated by polynomials in the state variables. The polynomials can have many fewer coefficients than entries in the original table at the expense of additional run-time computation.

For a system with n state variables to be controlled, and i control inputs, n polynomials are constructed, each a function of $N = n + i$ state variables and control inputs. Each of the n polynomials minimizes the total square error for the variable it approximates across all data points in the table.

Let A be a matrix in which each row contains values of the N state variables and control inputs; let B be the matrix that contains future values (that is, at lift-off) of the state variables in corresponding rows. The matrices A and B then form a data structure for the table. Given a sequence of distinct terms of the form:

$$\langle x_1^{\alpha_{11}} x_2^{\alpha_{12}} x_3^{\alpha_{13}} x_4^{\alpha_{14}}, \cdots, x_1^{\alpha_{M1}} x_2^{\alpha_{M2}} x_3^{\alpha_{M3}} x_4^{\alpha_{M4}} \rangle \quad (5)$$

determine a row of the M-column matrix C by evaluating these terms at the values defined by the same row of A. Then

$$(C^T C X = C^T B) \quad (6)$$

is a linear system whose solution X contains, in each column, the coefficients of a least squares polynomial that estimates the values in the corresponding column of B [11]. The polynomial is determined by the choice of the exponents in the above sequence, many of which are set to zero.

Using such polynomials, (2) can be minimized in closed form. The details of the procedure are given in Appendix B. For the case of the one-legged machine, the components of x'_{TD}, namely $\dot{x}_{TD}$, θ_{TD}, $\dot{\theta}_{TD}$, and φ_{TD}, are the independent variables of the polynomials, and the components of x_{LO}, that is $\dot{x}_{LO}$, θ_{LO}, and $\dot{\theta}_{LO}$, are the variables to be approximated.

Several polynomials have been tested using the same simulation procedure described earlier in Fig. 2, as well as other similar tests. In each case, the resulting behavior was compared to that obtained with the tabular data. One measure of comparison is the area between the θ trajectory produced by the table and the trajectories produced by each of the polynomials. The trajectories used are shown in Fig. 4, and the areas are given in the following list:

- *24 term polynomial* consists of all odd terms of degree 3 or less. The body angle did not reach the setpoint after 20 s. The area between θ trajectories for table and for 24 term polynomial is 3.24 rad/s.
- *40 term polynomial* consists of all terms of degree 1 and 3, with 16 terms of degree 5. Behavior was similar to the table, with slightly less rapid convergence. The area between θ trajectories for table and for 40 term polynomial is 1.12 rad/s.
- *68 term polynomial* consists of all terms of degree 1, 3, and 5. Convergence is slightly faster than for the 40 term polynomial. The area between θ trajectories for table and for 68 term polynomial is 1.05 rad/s.
- *625 term polynomial* consists of all terms such that the exponents of each of the independent variables is less than or equal to 4. Behavior is very similar to the original tabulated data. The area between θ trajectories for table and for 625 term polynomial is 0.23 rad/s.

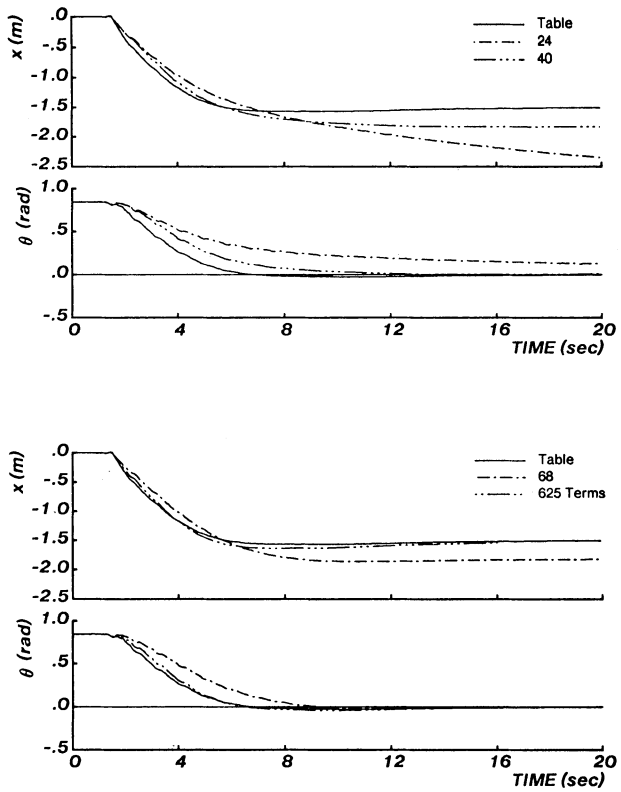

Fig. 4. Polynomial approximations are compared to tabular data in test that corrects orientation of body, using same procedure as in Fig. 2. The responses for the tabular data and for four different polynomial approximations are plotted. See text for description of how terms of polynomial were chosen.

The approximation error for each polynomial is listed in the following Table I.

The area measure given earlier allows a more realistic assessment of the polynomial than the global fit measure, since it takes into account which parts of the table are used for control and which are not.

To use the table we must search it at run-time to find the entry that minimizes (2). Specifically this is a search where $\dot{x}$, θ and $\dot{\theta}$ are fixed and φ may vary. The table is quantized with respect to φ at nine equally spaced points in the interval -1 rad $< \varphi < 1$ rad. In each subinterval we derive a linear interpolation formula for each element of x_{LO}. Since φ is the only free variable, we substitute for $\dot{x}$, θ, and $\dot{\theta}$ into (2), differentiate with respect to φ, set the result equal to zero, and solve for φ. Eight candidates are obtained. We select the one that minimizes globally. The details of this procedure are given in Appendix I.

A similar approach is used in applying the polynomial. Since $\dot{x}$, θ, and $\dot{\theta}$ are fixed during the search for φ, multivariate polynomials in four variables become simpler polynomials in one variable. For instance, consider the 68 term polynomials discussed above. The highest power of φ occurring in each of the three polynomials is φ^5. When computing a control signal, we evaluate six coefficients for each of the three polynomials; these are determined by the given values of $\dot{x}, \theta, \dot{\theta}$ as well as by the 68 original coefficients. The six term polynomials are algebraically substituted into (2), and the resulting polynomial of degree ten is differentiated with respect to φ. The result is a single polynomial

Table I

	Mean Square Error		
No. of Terms	$\dot{x}$	θ	$\dot{\theta}$
24	16.6	0.602	4.53
40	13.6	0.527	4.39
68	10.5	0.359	4.25
625	9.55	0.308	3.94

Table II

No. of Terms	Mults to find poly in φ	Mults to find roots	Total Mults
24	196	360	556
40	343	1035	1378
68	460	1035	1495
625	2822	875	3697
Table			408

of degree nine in φ. Its zeros are found by using Laguerre's method [12]. The PI is explicitly evaluated for each real zero in the interval $-1 < \varphi < 1$, and the smallest of these values determines the globally optimum φ. The details of this procedure are given in Appendix II.

This procedure for minimizing (2) does not require finding the zeros of a large polynomial in four variables. The 24 term polynomials require finding zeros of a single 5th degree polynomial in φ, while the 40, 68, and 625 term polynomials require finding roots of 9th, 9th, and 7th degree polynomials in φ, respectively.

The computational cost of using the approximating polynomials cannot be determined precisely for the general case, since the cost depends on which terms are included in the polynomials, and since an iterative method is used to find roots. In order to get a rough idea of the computational cost of approximating tabular data with polynomials, we measured the number of multiplies required to use each of the four polynomials mentioned earlier. The cost is broken into two parts: the number of multiplies needed to apply the methods of Appendix II in order to convert the original multivariate polynomials into polynomials in one variable φ, and the number of multiplies needed to find the roots of this polynomial using Laguerre's method. The rough operation counts shown in Table II suggest that polynomials with a few dozen terms may be used with only three or four times the run-time computing cost of the table. The 625 term polynomial requires almost ten times the run-time computing that the table requires.

Conclusion

Tables can be useful for control of nonlinear dynamic systems when simple inverse descriptions are not available, or when evaluation of such descriptions requires unacceptably large amounts of run-time computation. The limitation of using tabulated data for control is in the size of the tables needed. In this paper we approached this problem in two ways.

First we took advantage of the particular characteristics of legged locomotion in order to partition the state variables of the system into two sets; those that varied in a predictable, stereotyped manner, and those that varied freely. Only the freely varying subset contributed to the size of the table.

Second, multivariate polynomials were used to approximate the tabular data. This resulted in a substantial reduction of memory requirements at the expense of additional run-time computing requirements. Two of the polynomials we examined, those with 40 and 68 terms, gave good performance with low storage and run-time computing cost.

One way to interpret the use of tabular data in the present method is that they are used to make predictions about the future state of the system based on the present state. Specifically for the locomotion problem, the tabular data provide a prediction of state at lift-off based on an augmented state vector at touchdown. Such a prediction provides the same sort of information as would a forward integration of the system's equations of motion, but much faster and with less computation.

Appendix I. Algorithm that Minimizes Performance Index for Tabular Data

Given a state vector at touchdown $[\dot{x}, \theta, \dot{\theta}]$ it is required to find the value of φ that minimizes the performance index $\mathrm{PI}([\dot{x}_{\mathrm{LO}}, \theta_{\mathrm{LO}}, \dot{\theta}_{\mathrm{LO}}])$ where $[\dot{x}_{\mathrm{LO}}, \theta_{\mathrm{LO}}, \dot{\theta}_{\mathrm{LO}}] = \mathrm{T}([\varphi, \dot{x}, \theta, \dot{\theta}])$ is the state vector at next lift-off; T denotes the vector function that implements the table look-up with linear interpolation, and T_i is the ith component of T. Recall that

$$\mathrm{PI}([\dot{x}_{\mathrm{LO}}, \theta_{\mathrm{LO}}, \dot{\theta}_{\mathrm{LO}}]) = Q_1(\dot{x}_{\mathrm{LO}} - \dot{x}_d)^2 + Q_2(\theta_{\mathrm{LO}} - \theta_d)^2 + Q_3(\dot{\theta}_{\mathrm{LO}} - \dot{\theta}_d)^2 \quad (7)$$

where Q_1, Q_2 and Q_3 are weights and $\dot{x}_d$, θ_d, and $\dot{\theta}_d$ are desired values.

Assume that $\dot{x}_A \leqslant \dot{x} \leqslant \dot{x}_B$, $\theta_A \leqslant \theta \leqslant \theta_B$, and $\dot{\theta}_A \leqslant \dot{\theta} \leqslant \dot{\theta}_B$ where the A and B subscripts indicate adjacent values stored at quantized locations in the table. Also, assume that: $\varphi_A \leqslant \varphi \leqslant \varphi_B$. Consider as an example the first term on the right side of (7), which can be written in terms of the tabulated data:

$$Q_1(\dot{x}_{\mathrm{LO}} - \dot{x}_d)^2 = Q_1\left[\frac{(\varphi - \varphi_A)\dot{x}_B - (\varphi_B - \varphi)\dot{x}_A}{(\varphi_B - \varphi_A)} - \dot{x}_d\right]^2 \quad (8)$$

where $\dot{x}_A = T_1[(\varphi_B, \dot{x}, \theta, \dot{\theta})]$ and $\dot{x}_B = T_1[(\varphi_B, \dot{x}, \theta, \dot{\theta})]$. In other words, once interpolation has been completed for the three state variables, two adjacent values in the table that bracket φ, φ_A and φ_B, are substituted into the linear interpolation formula. Equation (8) represents the linear interpolation in the table for the φ dimension.

Now the right side of (8) can be rewritten as

$$Q_1(\dot{x}_{\mathrm{LO}} - \dot{x}_d)^2 = Q_1(\varphi Q_A + Q_B)^2 \quad (9)$$

where

$$Q_A = \frac{\dot{x}_B - \dot{x}_A}{\varphi_B - \varphi_A} - \dot{x}_d \text{ and } Q_B = \frac{\dot{x}_A\varphi_B - x_B\varphi_A}{\varphi_B - \varphi_A} - \dot{x}_d.$$

An identical treatment of the other two terms of (7) yields

$$\mathrm{PI} = Q_1[Q_A\varphi + Q_B]^2 + Q_2[Q_C\varphi + Q_D]^2 + Q_3[Q_E\varphi + Q_F]^2. \quad (10)$$

Upon differentiation with respect to φ, setting the result equal to zero, and solving for φ we have a closed expression for that value of φ that minimizes the PI in the interval $\varphi_A \leqslant \varphi \leqslant \varphi_B$:

$$\varphi = -\frac{Q_1Q_AQ_B + Q_2Q_CQ_D + Q_3Q_EQ_F}{Q_1Q_A^2 + Q_2Q_C^2 + Q_3Q_E}. \quad (11)$$

To obtain the *global* minimum this computation is performed for each of the $M - 1$ subintervals determined by the quantization of φ.

Appendix II. Algorithm that Minimizes Performance Index for Polynomial

The general form of the K-term polynomials that approximate the tabulated data is

$$\begin{aligned}
\theta_{\mathrm{LO}} &= f_{1,1}\varphi^{\alpha_{11}}\theta^{\alpha_{12}}\dot{\theta}^{\alpha_{13}}\dot{x}^{\alpha_{14}} + \cdots + f_{1,K}\varphi^{\alpha_{K1}}\theta^{\alpha_{K2}}\dot{\theta}^{\alpha_{K3}}\dot{x}^{\alpha_{K4}} \\
\dot{\theta}_{\mathrm{LO}} &= f_{2,1}\varphi^{\alpha_{11}}\theta^{\alpha_{12}}\dot{\theta}^{\alpha_{13}}\dot{x}^{\alpha_{14}} + \cdots + f_{2,K}\varphi^{\alpha_{K1}}\theta^{\alpha_{K2}}\dot{\theta}^{\alpha_{K3}}\dot{x}^{\alpha_{K4}} \\
\dot{x}_{\mathrm{LO}} &= f_{3,1}\varphi^{\alpha_{11}}\theta^{\alpha_{12}}\dot{\theta}^{\alpha_{13}}\dot{x}^{\alpha_{14}} + \cdots + f_{3,K}\varphi^{\alpha_{K1}}\theta^{\alpha_{K2}}\dot{\theta}^{\alpha_{K3}}\dot{x}^{\alpha_{K4}}
\end{aligned} \quad (12)$$

where, again, the LO subscript denotes values at next lift-off and $[\theta, \dot{\theta}, \dot{x}]$ is a state vector at touchdown. Since φ is the only free variable, (12) can be recast as

$$\theta_{LO} = F_{1,1}\varphi^{\beta_0} + \cdots + F_{1,N}\varphi^{\beta_N}$$
$$\dot{\theta}_{LO} = F_{2,1}\varphi^{\beta_0} + \cdots + F_{2,N}\varphi^{\beta_N}$$
$$\dot{x}_{LO} = F_{3,1}\varphi^{\beta_0} + \cdots + F_{3,N}\varphi^{\beta_N} \quad (13)$$

where $\beta_0 \geqslant 0$ and β_N is the highest power of φ in (12). Substituting these equations into (7) yields a polynomial of degree $2\beta_N$ that expresses the PI as a function of φ:

$$PI = G_0 + \theta_d^2 + \dot{\theta}_d^2 + \dot{x}_d^2 + G_1\varphi^{2\beta_0} + \cdots + G_{2N}\varphi^{2\beta_N}. \quad (14)$$

The real zeros of the derivative of this polynomial are found using Laguerre's method. The global minimum is found by evaluating (13) for each zero and substituting the results into (7).

References

[1] J. M. Hollerbach, "A recursive formulation of Lagrangian manipulator dynamics," *IEEE Trans. Syst., Man, Cybern.*, vol. 10, pp. 730–736, 1980.

[2] J. S. Albus, "A new approach to manipulator control: the cerebullar model articulation controller (CMAC)," *J. Dynamic Systems, Measurement, and Control, Series G 97*, vol. 3, pp. 220–227, 1975.

[3] ——, "Data storage in the cerebellar model articulation controller (CMAC)." *J. Dynamic Systems, Measurement, and Control, Series G 97*, vol. 3, pp. 228–233, 1975.

[4] M. H. Raibert, "A model for sensorimotor control and learning," *Biological Cybernetics 29* (1978), 29–36.

[5] H. Miura, and I. Shimayama, "Computer control of an unstable mechanism," *J. Fac. Eng.*, vol. 17, pp. 12–13, 1980.

[6] M. H. Raibert, "Analytical equations vs. table look-up for manipulation: A unifying concept." in *Proc. IEEE Conf. on Decision and Control*, pp. 576–579, 1977.

[7] M. H. Raibert and B. K. P. Horn, "Manipulator control using the configuration space method," *The Industrial Robot 5*, vol. 2, pp. 69–73, 1978.

[8] J. Simons, H. Van Brussel, J. De Shutter, and J. Verhaert, "A self-learning automaton with variable resolution for high precision assembly by industrial robots." *IEEE Trans. Automat. Contr.* vol. AC-27, pp. 1109–1112, 1982.

[9] M. H. Raibert, "Hopping in legged systems—Modelling and simulation for the 2D one-legged case." *IEEE Trans. Syst., Man, Cybern.*, in press 1984.

[10] M. H. Raibert, H. B. Brown, Jr., M. Chepponis, E. Hastings, S. T. Shreve, and F. C. Wimberly, "Dynamically Stable Legged Locomotion," Robotics Inst., Carnegie-Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-81-9, 1981.

[11] G. W. Stewart, *Introduction to Matrix Computations*. New York, Academic, 1973.

[12] G. Dahlquist, A. Bjorck, N. Anderson, *Numerical Methods*. Englewood Cliffs, NJ: Prentice-Hall, 1974.

General Models for Simulating Population Growth in Insects Grouped Both by Stages and Age Classes

R. L. TUMMALA, LI DIANMO, AND D. L. HAYNES

***Abstract*—Pest models are typically required by modern pest management systems as the basis for decisionmaking [1], [2]. Many models have been developed in the past that are specific to a given pest population [3]. The authors have found from their past experience in this area that the structure of these models are similar and thus a general model structure can be identified. Based on this understanding, several general component models are developed that can be used to simulate the population growth of many insects of interest. It is the purpose of this correspondence to present these models and show their application to agricultural pests.**

Manuscript received April 11, 1983. This work was supported in part under EPA Grant no. R-803785.

R. L. Tummala is with the Department of Electrical Engineering and System Science, Michigan State University, East Lansing, MI 48824.

L. Dianmo is a Visiting Scholar from Beijing, China.

D. L. Haynes is with the Department of Entomology, Michigan State University, East Lansing, MI 48824.

I. Introduction

This correspondence is organized as follows: Section II deals with the general description of the pest dynamics. The description of the models are given in Section III. Section IV describes the computer implementation. Section V discusses an example illustrating the use of these models.

II. Pest Dynamics

For many agricultural insects, the activity starts in the spring and stops at the end of the summer. Adult insects emerge from their overwintering sites, feed, mate, lay eggs and die of natural causes in a few weeks. The eggs laid by the adults hatch in a few days and go through two other developmental stages called larva and pupa before becoming adults. The number of adults that emerge at the end of the developmental cycle is determined by the survival potentials of these immature stages (eggs, larvae, pupae). Usually these survivals depend on weather, food resources, and naturally occurring biological controls such as parasites, predators, etc. At the end of the summer, the available adults disperse to their overwintering sites. These adults do not lay eggs until the following spring. Furthermore, the number of adults returning in the next spring is dependent upon their overwintering survival potential. The number of generations that an insect can have varies from insect to insect.

In summary, the population dynamics of an insect is determined by 1) egg-laying capacity of the spring adults (fecundity rate); 2) survival and development of eggs, larvae, and pupae during the season; 3) overwintering survival of summer adults; and 4) effects due to natural enemies and other external control measures.

The models developed here consider the recognizable stages of the population, such as eggs, larvae, pupae, etc., temperature-dependent time lags (developmental rates), fecundity rates and density dependent survival rates. Furthermore, we incorporated the age structure within each stage by dividing each stage into ten age classes. Since certain biological predators or parasites key on the size of the insect's life stage, the incorporation of age structure within each life stage provides accurate estimates of predation and parasitism. Furthermore, the age-dependent crop damage caused by each life stage can be accurately estimated.

III. Model Development

If we assume that every insect has a finite number of identifiable stages, say, s, $(s \geqslant 1)$, then the number in stage i at time $(t + \Delta t)$ can be written as

$$N(t + \Delta t) = M(t)N(t) \quad (1)$$

where $N(t) = [N_1(t), N_2(t) \cdots N_s(t)]$ is called the population vector, and

$$M(t) = \left[m_{ij}(t)\right] \qquad i = 1,2,\cdots,s \qquad j = 1,2,\cdots,s$$

is the transition matrix whose elements are functions of temperature.

In the above formulation, we assume that the update interval Δt is chosen such that it is small compared to the smallest development time of all the stages. The result of this choice is that the number in stage i at $(t + \Delta t)$ only depends on the number in stage i and stage $(i - 1)$ at time t. A reasonable choice for many of the insects is one calendar day. The form of

0018-9472/84/0300-0339$01.00 ©1984 IEEE