SAMS, 1995, Vol. 17, pp. 229–239
Reprints available directly from the publisher
Photocopying permitted by license only

EVALUATION OF MANIPULATOR DIRECT DYNAMICS USING CUSTOMIZED RUNGE-KUTTA METHODS

J. A. TENREIRO MACHADO and ALEXANDRA M. S. F. GALHANO

Faculty of Engineering of the University of Porto, Department of Electrical and Computer Engineering, 4099 Porto Codex, Portugal

(Received June 26, 1994)

This paper studies the efficiency of customized Runge-Kutta methods on the evaluation of the direct dynamics of robot manipulators. Customizing consists not only on the elimination of unnecessary and redundant calculations but also on the development of efficient integration methods. In this perspective, we investigate numerical algorithms with high order expansion and integration with adaptive step size based on the embedded estimation of the local truncation error.

KEY WORDS Numerical algorithms, robot manipulators, mechanical systems.

1. INTRODUCTION

Robotic manipulators are mechanical systems composed by several links interconnected through linear or rotational joints. The dynamics of these systems follows the laws of classical mechanics and, therefore, it can be studied on the basis of a well known, established, theoretical paradigm. Nevertheless, in practice, the derivation of the dynamic equations and its computer subsequent evaluation pose stringent problems. Due to these reasons, research is on progress in order to develop new methods for the study of the dynamic phenomena.

The dynamics can assume the differential or the integral forms which consist in the so-called inverse and direct descriptions. For the inverse dynamics we may mention the studies of Hollerbach [1] and Luh, Walker and Paul [2] which developed recursive numerical algorithms based on the Lagrangian and Newton-Euler formalisms, respectively. The comparison of the corresponding computational efficiency reveals that, although being both methods superior to the standard Uicker-Kahn algorithm [3, 4], the Newton-Euler scheme is the best. Nevertheless, Silver [5] demonstrated that such algorithms are, merely, alternative forms of the dynamics. By other words, the proposed numerical methods are different algorithms describing the same phenomena and, consequently, leading to the same results. An alternative strategy is the replacement of the computer calculation by memory evaluations. Raibert [6, 7] proposed the use of memory look-up tables while Albus developed an associative memory scheme [8, 9]. This second scheme was further investigated by Hirzinger [10] and Miller [11]; however, memory-based methods require huge computer memories and are, still, far from overcoming a research stage. Horak [12] studied a hybrid method consisting on the evaluation of the dynamics using symbolic equations for the three degrees of freedom (dof) of the arm and adopting a numerical recursive calculation for the three dof of the wrist. The hybrid method proved to be more efficient than the Newton-Euler scheme and, furthermore, suggested that symbolic-based calculations rather than numerical recursive computations could be a strategy towards better algorithms. Nevertheless, the hand held derivation of the expressions corresponding to the dynamics of a six dof manipulator is impractical and requires the adoption of symbolic manipulating packages. Leu and Hemati [13], Koplic and Leu [14], Faessler [15] and Neuman and Murray [16] developed computer-based procedures and demonstrated that the evaluation of the dynamics described through symbolic equations could achieve higher sampling frequencies. The last stage of performance optimization was attained by Neuman and Mürray [17, 18] using the 'customized computing' philosophy. This strategy corresponds to the elimination of all the unnecessary calculations such as additions of zero, multiplications by zero or one and the simplification of trigonometric identities. The results showed that both the Newton-Euler and the symbolic calculations were optimized. Moreover, the experiments revealed that for simple two or three dof manipulators a symbolic algorithm was the best approach while for more complex cases, that is, for robots with five or six dof, the customized Newton-Euler method was the most efficient scheme. Two alternative strategies, that were also studied, are the construction of mechanical structures that lead to simple dynamic equations [19–21] and the reduction of the complexity of the equations eliminating terms of small amplitude [22, 23]. In the first case, the simplification may be difficult or may lead to performance limitations. In the second case, we may get modelling errors that affect negatively model-based controllers [24].

The evaluation of the direct dynamics was investigated by Walker and Orin [25] that compared four methods of computing the joint accelerations using the Newton-Euler scheme knowing the joint positions, velocities and force/torques. The methods take advantage of the symmetry of the inertial matrix in order to get a faster computation. More recently, Neuman and Tourassis [26, 27] and Lee and Tsay [28] addressed the problem of integrating numerically the ordinary differential equations (ode's) that describe the inverse dynamics. Being mechanical systems, robot manipulators must obey the principle of the conservation of energy and, eventually, the principle of the conservation of momentum. Therefore, a numerical method based on such principles is a natural candidate for the integration of the dynamic ode's. Nevertheless, this method is compared with standard algorithms on the basis of indexes consisting on the residuals on energy and momentum that are not the most adequate. In fact, for robotic applications, we need to study the computational efforts *versus* the trajectory error that results for a given algorithm. In this line of thought, this paper addresses the efficiency of numerical methods for ode's on the context of the direct dynamics of robot manipulators [29, 30]. In section two, we begin by formulating the problem of the direct dynamics. Then, in a second stage, we analyse the integration procedure using the Taylor series expansion of the solution. Based on this preliminary analysis, in section three we compare the performances of Runge-Kutta methods with fixed and adaptive step sizes of integration. Finally, in section four, the main conclusions are drawn.

2. DIRECT DYNAMICS OF ROBOT MANIPULATORS

The dynamics of mechanical manipulators consists on a set of phenomena that follow the laws of classical mechanics. For a n dof manipulator the dynamics may be expressed in the form of a differential relation–the inverse dynamics–of the type:

$$\{\mathbf{q}(t), \dot{\mathbf{q}}(t), \ddot{\mathbf{q}}(t)\} \rightarrow \{\mathbf{T}(t)\} \tag{1}$$

where t is time, $\mathbf{q}$, $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$ are the n-vectors of joint positions, velocities and accelerations and T is the n-vector of joint torques. On the other hand, the dynamics may, also, take the form of an integral relation–the direct dynamics–of the type.

$$\{\mathbf{T}(t), \dot{\mathbf{q}}(0), \mathbf{q}(0)\} \rightarrow \{\ddot{\mathbf{q}}(t), \dot{\mathbf{q}}(t), \mathbf{q}(t)\} \tag{2}$$

Unfortunately, the inverse dynamics imposes an high computational burden that makes difficult its real-time calculation [31]. As the direct dynamics stems from the inverse model (1) we have also, for this case, similar difficulties.

2.1. *The Problem Formulation*

For a n dof manipulator the inverse dynamics (1), when expressed in the symbolic form, leads to a set of matrix ode's:

$$\mathbf{T} = \mathbf{J}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) \tag{3}$$

where $\mathbf{J}(\mathbf{q})$ is the intertial matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ represents the Coriolis/centripetal torques and $\mathbf{G}(\mathbf{q})$ are the gravitational torques. The direct dynamics (2) corresponds to the integration of the previous equation, that is to:

$$\ddot{\mathbf{q}} = \mathbf{J}(\mathbf{q})^{-1}[\mathbf{T} - \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) - \mathbf{G}(\mathbf{q})] \tag{4a}$$

$$\dot{\mathbf{q}} = \int \ddot{\mathbf{q}}\, \mathrm{d}t \tag{4b}$$

$$\mathbf{q} = \int \dot{\mathbf{q}}\, \mathrm{d}t \tag{4c}$$

Equation (4) is highly non-linear and, therefore, its integration requires the adoption of a numerical method. However, there are several techniques and, consequently, their relative performance must be investigated. Moreover, in the context of robotics our main points of interest are the computational efficiency and the error that results using an approximate numerical solution.

2.2. *Solution Through Taylor Series Expansion*

The Taylor series expansion of the solution is the first technique that appears when thinking on the approximate solution of an ode. The solution $q_i(t)$ $(i = 1, \ldots, n)$ of (4) can be expanded in a Taylor series about a point $t = t_r$ as:

$$q_i(t) = \sum_{j=0}^{p} c_{ij}(t_r) \cdot (t - t_r)^j + R_{p+1} \tag{5}$$

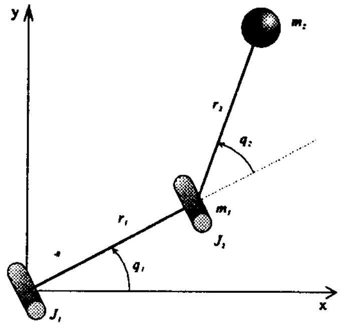

Figure 1 The RR manipulator.

where p is the order of the series approximation, c_{ij} are the polynomial coefficients and R_{p+1} is the $(p+1)$-th local truncation error (LTE). Taking the first and second derivatives of the truncated series (5) and substituting in (4) we can find c_{ij} $(i = 1, \ldots, n, j = 0, \ldots, p)$ through the method of undetermined coefficients. In this line of thought, in the sequel we perform several experiments, considering as our prototype manipulator of the RR structure (Fig. 1) with dynamics:

$$\mathbf{J}(\mathbf{q}) = \begin{bmatrix} (m_1 + m_2)r_1^2 + m_2 r_2^2 & m_2 r_2^2 + m_2 r_1 r_2 C_2 \\ + 2m_2 r_1 r_2 C_2 + J_1 & \\ m_2 r_2^2 + m_2 r_1 r_2 C_2 & m_2 r_2^2 + J_2 \end{bmatrix} \tag{6a}$$

$$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} -m_2 r_1 r_2 S_2 \dot{q}_2^2 - 2m_2 r_1 r_2 S_2 \dot{q}_1 \dot{q}_2 \\ m_2 r_1 r_2 S_2 \dot{q}_1^2 \end{bmatrix} \tag{6b}$$

$$\mathbf{G}(\mathbf{q}) = \begin{bmatrix} g(m_1 r_1 C_1 + m_2 r_1 C_1 + m_2 r_2 C_{12}) \\ g m_2 r_2 C_{12} \end{bmatrix} \tag{6c}$$

where $C_i = \cos(q_i)$, $C_{ij} = \cos(q_i + q_j)$ and $S_i = \sin(q_i)$. We start by studying the efficiency of the *customized* Taylor series for several p-th order approximations. The customizing of the computation consists on the simplification of all the redundant or unnecessary calculations and the elimination of the loops of the algorithm through the direct expansion of the corresponding code. In the experiments we decided to include an heuristic alternative of the Taylor method, using both p-th order polynomials for $\mathbf{q}(t)$ and $\dot{\mathbf{q}}(t)$ instead of the p-th and $(p-1)$-th order analytical approximations. With this strategy we can study the influence of the truncation error of the velocity.

Figure 2 shows the maximum error *versus* the computation time required for a given trajectory. The chart reveals that:

—The higher series order approximation the lower the trajectory error for a given fixed number of steps of integration

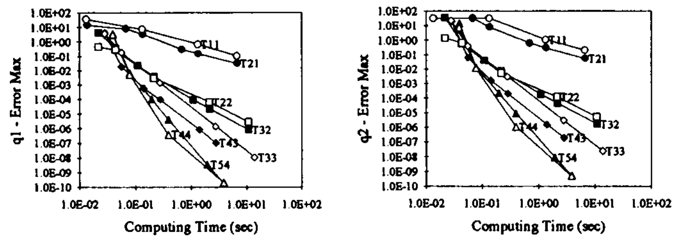

Figure 2 Maximum trajectory error *versus* required computing time using Taylor expansion with i-th and j-th order polynomials for position and velocity (T_{ij}).

—The 'investment' on higher order expansions gives better results than the adoption of an higher number of integration steps
—The heuristic method gives a limited improvement over the standard algorithm and leads to more complex characteristics.

In conclusion, the higher the order of the series expansion the better performances we get. Nevertheless, the derivation of the symbolic formulae for the c_{ij} becomes very complex for higher values of p and n. Therefore, solutions based on alternative methods are required and that is the subject of the next section.

3. ANALYSIS OF CUSTOMIZED COMPUTATIONAL METHODS FOR THE DIRECT DYNAMICS

The literature presents a plethora of numerical methods for integrating ode's. For a practical case, and given a set of initial conditions, there is no simple criteria for the selection of the most appropriate method to implement. Important issues such as computational load, convergence and trajectory error have intricate relationships and, in practice, it is difficult to know in advance the real performances of a numerical method. In fact, given an ode and a set of initial conditions the best strategy is to experiment the algorithm and, *a posteriori*, to investigate it from the point of view of the performance criteria. Having these ideas in mind, we decided to compare the computational efficiency *versus* the maximum trajectory error, for several numerical methods when applied to the RR manipulator. Furthermore, in order to restrict the number of schemes under study, we consider the class of Runge-Kutta (RK) methods [32, 33].

3.1. *Runge-Kutta Methods with Fixed Step*

The RK methods with fixed step size [32–35] are popular algorithms for integrating ode's. The general s-stage explicit RK method for the system of m first-order ode's:

$$y' = f(t, y), \quad y(t_0) = y_0, \quad t \in [t_0, t_a] \tag{7}$$

follows the algorithm

$$t_{r+1} = t_r + h \tag{8a}$$

$$y_{r+1} = y_r + h \sum_{i=1}^{s} b_i \mathbf{k}_i \tag{8b}$$

$$\mathbf{k}_i = \mathbf{f}\left(t_r + c_i h, y_r + h \sum_{j=1}^{i-1} a_{ij} \mathbf{k}_j\right), \quad i = 1, 2, \ldots, s \tag{8c}$$

where h is the step of integration and a_{ij}, b_i and c_i are coefficients to be defined for each method and displayed in the so-called Butcher array:

$$\begin{array}{c|ccccc} 0 & & & & & \\ c_2 & a_{21} & & & & \\ c_3 & a_{31} & a_{32} & & & \\ \vdots & \vdots & \vdots & \vdots & & \\ c_s & a_{s1} & a_{s2} & \cdots & a_{s,s-1} & \\ \hline & b_1 & b_2 & \cdots & b_{s-1} & b_s \end{array} \tag{9}$$

Table 1 shows the number s of stages of required calculations for a given RK method of order i (RKi) for each first-order ode.

For the robot dynamics (6) the customized RK methods lead to the chart of Figure 3. The experiments were performed with robot parameters and initial conditions similar to those adopted in the previous section. It must be highlighted that it is not valid to extrapolate the curves to different numerical situations; nevertheless, it is common practice to generalize the conclusions of a particular chart to the whole class of ode's to which it corresponds. The results of Figure 3 reveal that the Taylor

Table 1 Computational load of Runge-Kutta methods for a first-order ode.

Method		*Number of stages of of calculation*
Fixed step	*Adaptive step*	*s*
RK 1		1
RK 2	RKB 1(2)	2
RK 3	RKF 1(2), RKF 2(3)	3
RK 4	RKB 2(4)	4
	RKF 3(4), RKB 3(4)	5
RK 5	RKF 4(5)	6
RK 6	RKB 4(5), RKDP 5(4)	7
	RKF 5(6), RKV 5(6)	8
RK 7	RKB 5(6)	9
	RKV 6(7)	10
RK 8		11
	RKF 7(8), RKV 7(8), RKDP 8(7)	13
RK 10		17

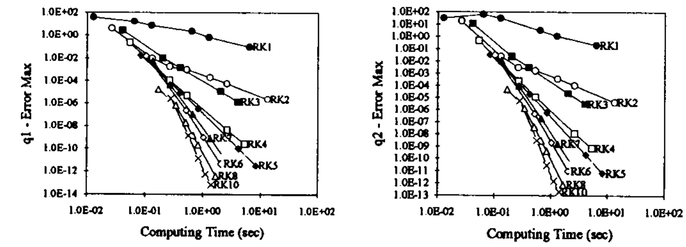

Figure 3 Maximum trajectory error *versus* required computing time using the Runge-Kutta method of order *i* (RK*i*).

series expansion and the RK method have similar properties; however, the RK method does not require the laborious derivation of a formula for each particular case. We must also mention that there are several alternative sets of coefficients for each different order of approximation of the RK method. The experiments showed that, for the same order of approximation of the RK method, alternative sets of numerical coefficients produce minor differences on the final chart.

3.2. *Embedded Runge-Kutta Methods*

Embedded RK methods consist on the adaptive variation of the size of the step of integration based on the estimation of the LTE. This strategy requires the calculation of two expansions of different order. In this sense, a RK $p(q)$ method is the p-th order approximation of the solution using a q-th order estimation of the LTE. Nevertheless, the calculation of two different solution requires almost twice the computational effort. The embedding overcomes this problem by taking advantage of the freedom of choice in the coefficients of the RK method. With this strategy, the p-th and q-th order expansions have calculations in common and the total computational load is reduced.

A s-stage RK embedded method for the system of ode's (7) follows the algorithm:

$$t_{r+1} = t_r + h \tag{10a}$$

$$y_{r+1} = y_r + h \sum_{i=1}^{s} b_i \mathbf{k}_i \tag{10b}$$

$$\hat{y}_{r+1} = y_r + h \sum_{i=1}^{s} \hat{b}_i \mathbf{k}_i \tag{10c}$$

$$\mathbf{k}_i = \mathbf{f}\left(t_r + c_i h, y_r + h \sum_{j=1}^{i-1} a_{ij} \mathbf{k}_j\right), \quad i = 1, 2, \ldots, s \tag{10d}$$

where (10b) and (10c) are solution approximations of order p and q (usually $q = p - 1$ or $q = p + 1$), respectively. The coefficients for each method may be displayed in a Butcher array of the type:

$$\begin{array}{c|ccccc} 0 & & & & & \\ c_2 & a_{21} & & & & \\ c_3 & a_{31} & a_{32} & & & \\ \vdots & \vdots & \vdots & \vdots & & \\ c_s & a_{s1} & a_{s2} & \cdots & a_{s,s-1} & \\ \hline & b_1 & b_2 & \cdots & b_{s-1} & b_s \\ \hline & \hat{b}_1 & \hat{b}_2 & \cdots & \hat{b}_{s-1} & \hat{b}_s \end{array} \tag{11}$$

Table 1 shows the number s of stages of required calculations for a given RK embedded method of order $p(q)$ (RK$p(q)$). Given an integration stepsize $h, \mathbf{y}_{r+1}$and $\hat{\mathbf{y}}_{r+1}$ denote the approximations to $\mathbf{y}(t_r+h)$ generated by the methods of order p and q, respectively. Therefore, for each variable, the estimation of the local truncation error (LTE) is:

$$\mathrm{LTE}_{r+1} \equiv |y_{r+1} - \hat{y}_{r+1}| \cong |\alpha(t_r)h^{p+1} - \beta(t_r)h^{q+1}| \tag{12}$$

and, providing that h is sufficiently small it results that:

$$\mathrm{LTE}_{r+1} \cong \begin{cases} |\alpha(t_r)|h^{p+1} & q>p \\ |\beta(t_r)|h^{q+1} & q<p \end{cases} \tag{13}$$

With this estimation, the stepsize h_{TOL} required to make the LTE about a tolerance magnitude TOL is:

$$h_{\mathrm{TOL}} = \sigma h \left(\frac{\mathrm{TOL}}{\mathrm{LTE}}\right)^{1/\max(p,q)+1} \tag{14}$$

where σ is a safety factor to increase the likelihood that h_{TOL} will produce an LTE less than TOL. Furthermore, for this technique we have to take into account:

—a larger number of stages of calculations than those required by the fixed step RK methods
—the computational overhead represented by the calculation of the adaptive size scheme
—the non-ideal estimation of the LTE which leads to 'failed steps', that is, which leads to the repetition of the integration using smaller steps when the LTE estimation fails.

In order to investigate the embedded RK strategy we considered the set of coefficients proposed by Fehlberg [36], Verner [37], Butcher [32] and Dormand and Prince [38, 39] using $\sigma = 0.9$. As referred previously, after each step the algorithm verifies if LTE > TOL. If this occurs the integration step in considered 'failed' and a new step is tried according with (14).

Figures 4 to 7 show the results of the customized embedded RK methods for the RR manipulator.

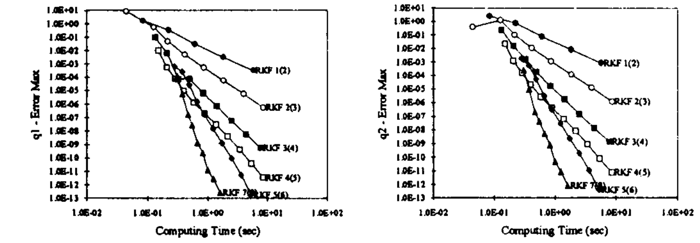

Figure 4 Maximum trajectory error *versus* required computing time using the Runge-Kutta-Fehlberg embedded method of order $p(q)$ [RKF $p(q)$].

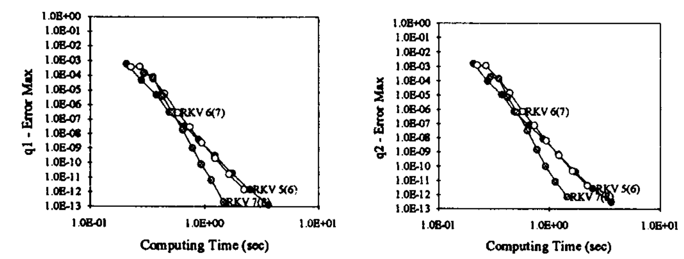

Figure 5 Maximum trajectory error *versus* required computing time using the Runge-Kutta-Verner embedded method of order $p(q)$ [RKV $p(q)$].

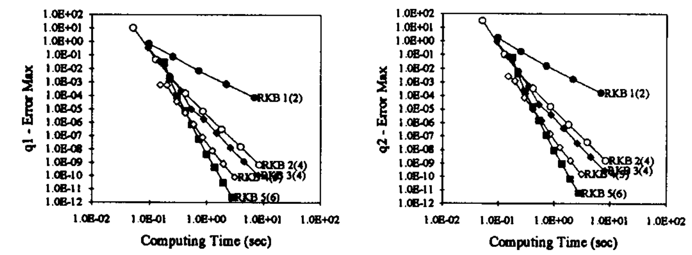

Figure 6 Maximum trajectory error *versus* required computing time using the Runge-Kutta-Butcher embedded method of order $p(q)$ [RKB $p(q)$].

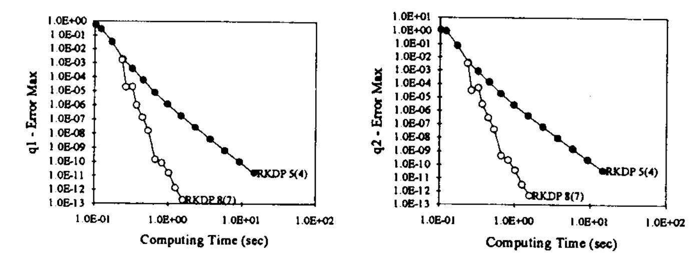

Figure 7 Maximum trajectory error *versus* required computing time using the Runge-Kutta-Dormand and Prince embedded method of order $p(q)$ [RKDP $p(q)$].

The charts reveal, once more, that the higher the order of the approximation the closer the numerical and ideal solutions. On the other hand, for a given total number of integration steps this technique is superior than the standard RK. Nevertheless, superimposing Figures 3 to 7 we verify that the benefits of the adaptive algorithm are lost, to a great extent, in the extra computation effort. Therefore, we may conclude that the requirement of an high precision may be attained through a massive computation using an huge number of steps and a simple algorithm or, otherwise, through a small number of steps using a method that is closer to the ideal solution, at the expense of an higher complexity.

4. CONCLUSIONS

The dynamics of robot manipulators models phenomena which obey the laws of classical mechanics. The dynamic model may be expressed in the differential and the integral forms that are the so-called inverse and direct dynamics. For the inverse dynamics, research lead to the concept of customizing computing as the most effective way of speeding-up calculations. This paper introduces the customized computing to the direct dynamics. In this case we verify that the elimination of unnecessary and redundant calculations is important. Nevertheless, in this context customizing goes deeper in the sense that the most effective integration methods are those that are closer to the solution. In this line of thought, high order solution expansions and variable step size of integration are key guidelines towards the development of customized direct dynamics algorithms.

References

[1] John M. Hollerbach, A Recursive Lagrangian Formulation of Manipulator Dynamics and a Comparative Study of Dynamics Formulation Complexity, *IEEE Trans. Syst., Man, Cybern.*, **10**, pp. 730–736, Nov. (1980).
[2] J. Y. S. Luh, M. W. Walker and R. P. C. Paul, On-line Computational Scheme for Mechanical Manipulators, *ASME J. Dynamic Syst., Meas., Contr.*, **102**, pp. 69–76, June (1980).
[3] J. J. Uicker, On the Dynamic Analysis of Spatial Linkages Using 4 x 4 Matrices, Ph.D dissertation, Northwestern Univ., Aug. (1965).
[4] M. E. Kahn, The Near-Minimum-Time Control of Open-Loop Articulated Kinematic Chains, Standford Artificial Intelligence Project Memo. AIM-106, Dec. (1969).
[5] William M. Silver, On the Equivalence of Lagrangian and Newton-Euler Dynamics for Manipulators, *The Int. J. Robotics Research*, **1** (2), pp. 60–70, Summer (1982).
[6] Marc H. Raibert, *IEEE Conf. Decision Contr.*, Analytical Equations vs. Table Look-Up for Manipulation. A Unifying Concept, New Orleans La., pp. 576–579, Dec. (1977).
[7] M. H. Raibert, A Model for Sensorimotor Control and Learning, *Biological Cybernetics* **29**, pp. 29–36 (1978).
[8] J. S. Albus, A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC), *ASME J. Dynamic Syst. Meas. Contr.*, **97**, pp. 220–227, Sept. (1975).
[9] J. S. Albus, Data Storage in the Cerebellar Model Articulation Controller (CMAC), *ASME J. Dynamic Syst. Meas. Contr.*, **97**, pp. 228–233, Sept. (1975).
[10] Gerhard Hirzinger, Robot Systems Completely Based on Sensory Feedback *IEEE Trans. Ind. Electron.*, **33**, pp. 105–109, June (1986).
[11] W. Thomas Miller III, Sensor-Based Control of Robotic Manipulators Using General Learning Algorithms, *IEEE J. Robotics and Automation*, **3**, pp. 157–165, April (1987).
[12] D. T. Horak, A Simplified Modelling and Computational Scheme for Manipulator Dynamics, *ASME J. Dynamic Syst., Meas., Contr.*, **106**, pp. 350–353, Dec. (1984).

[13] M. C. Leu and N. Hemati, Automated Symbolic Derivation of Dynamic Equations of Motion for Robotic Manipulators, *ASME J. Dynamic Syst. Meas. Contr.*, **108**, pp. 172–179, Sept. (1986).

[14] J. Koplic and M. C. Leu, Computer Generation of Robot Dynamic Equations and the Related Issues, *J. of Robotic Systems*, **3** (3), pp. 301–319, Fall (1986).

[15] H. Faessler, Computer-Assisted Generation of Dynamic Equations for Multibody Systems, *The Int. J. Robotics Research*, **5**, n. 3, pp. 129–141, Fall, (1986).

[16] Charles P. Neuman and John J. Murray, The Complete Dynamic Model and Customized Algorithms of the Puma Robot, *IEEE Trans. Syst., Man, Cybern.*, **17**, pp. 635–644, July/Aug. (1987).

[17] Charles P. Neuman and John J. Murray, Customized Computational Robot Dynamics, *J. of Robotic Systems*, **4**, pp. 503–526, Aug (1987).

[18] Charles P. Neuman and John J. Murray, Symbolically Efficient Formulations for Computational Robot Dynamics, *J. of Robotic Systems*, **4** (6), pp. 743–769, Dec. (1987).

[19] D. C. Yang and S. W. Tzeng, Simplification and Linearization of Manipulator Dynamics by the Design of Inertia Distribution, *The Int. J. Robotics Research*, **5** (3), pp. 120–128, Fall (1986).

[20] Kamal Youcef-Toumi and Haruhiko Asada, The Design of Open-Loop Manipulator Arms With Decoupled and Configuration-Invariant Inertia Tensors, *ASME J. Dynamic Syst., Meas., Contr.*, **109**, pp. 268–275, Sept. (1987).

[21] H. Kazerooni, Statically Balanced Direct Drive Robot Manipulator, Robotica, **7**, Part 2, pp. 143–149, April (1989).

[22] Richard P. Paul, Robot Manipulators, Mathematics, Programming and Control, Cambridge, MA: *Mass. Inst Tech.*, (1981).

[23] B. Armstrong, O. Khatib and J. Burdick, The Explicit Dynamic Model and Inertial Parameters of the PUMA 560 Arm, IEEE Int. Conf. on Robotics and Automation, San Francisco, CA (1986).

[24] Vassilios D. Tourassis, Principles and Design of Model-Based Robot Controllers, *Int. J. Control*, **47** (5), pp. 1267–1275 (1988).

[25] M. W. Walker and D. E. Orin, Efficient Dynamic Computer Simulation of Robotic Mechanisms, *ASME J. Dynamic Syst. Meas. Contr.*, **104**, pp. 205–211, Sept. (1982).

[26] Charles P. Neuman and Vassilios D. Tourassis, Discrete Dynamic Robot Models, *IEEE Trans. Syst., Man, Cybern.*, **15**, pp. 193–204, March/April (1985).

[27] Vassilios D. Tourassis and Charles P. Neuman, Inverse Dynamics Application of Discrete Robot Models, *IEEE Trans. Syst., Man, Cybern.*, **15**, pp. 798–803 Nov./Dec. (1985)

[28] Tsu-Tian Lee and Yuh-Feng Tsay, Analysis od Discrete Dynamic Robot Models, *IEEE J. Rob. Automat.*, **3**, pp. 583–590, Dec. (1987).

[29] J. A. Tenreiro Machado and Alexandra M. S. F. Galhano, Customized Direct Dynamics of Robot Manipulators, *IEEE Int. Conf. on Syst., Man, Cyber.*, Le Touquet, France, pp. 566–571, Oct. (1993).

[30] J. A. Tenreiro Machado and Alexandra M. S. F. Galhano, Customized Direct Dynamics of Mechanical Manipulators, IEEE 3rd Int. Workshop on Advanced Motion Control, Berkeley, California, March (1994).

[31] J. A. Tenreiro Machado and J. L. Martins de Carvalho, Microprocessor-Based Controllers for Robotic Manipulators, Microprocessors in Robotic and Manufacturing Systems, Kluwer Academic Publishers (1991).

[32] J. C Butcher, The Numerical Analysis of Ordinary Differential Equations, Runge-Kutta and General Linear Methods, Wiley (1987).

[33] E. Hairer, S. P. Nørsett and G. Wanner, Solving Ordinary Differential Equations I: Nonstiff Problems, Springer-Verlag (1987).

[34] G. J. Cooper and J. H. Verner, Some Explicit Runge-Kutta Methods of High Order, *SIAM J. Numer. Anal.*, **9**, pp. 389–405, Sept. (1972).

[35] E. Hairer, A Runge-Kutta Method of Order 10, *J. Inst. Maths Applics.*, **21**, pp. 47–59 (1978).

[36] Erwin Fehlberg, Classical Fifth-, Sixth- Seventh-, and Eighth-Order Runge-Kutta Formulas with Stepsize Control, NASA TR R-287 (1968).

[37] J. H. Verner, Explicit Runge-Kutta Methods with Estimates of the Local Truncation Error, *SIAM J Num. Anal.*, **15**, pp. 772–790, Aug. (1978).

[38] J. R. Dormand and P. J. Prince, A Family of Embedded Runge-Kutta Formulae, *J. of Comput. and Appl. Math.*, **6**, pp. 19–26 (1980).

[39] P. J. Prince and J. R. Dormand, High Order Embedded Runge-Kutta Formulae, *J. of Comput. and Appl. Math.*, **7**, pp. 67–75 (1981).