Struct Multidisc Optim
DOI 10.1007/s00158-008-0247-2

RESEARCH PAPER

# Optimization-based motion prediction of mechanical systems: sensitivity analysis

**Yujiang Xiang · Jasbir S. Arora · Karim Abdel-Malek**

Received: 11 June 2007 / Revised: 17 October 2007 / Accepted: 6 February 2008


**Abstract** In this study, we derive sensitivity equations for the problem of optimization-based motion prediction of a mechanical system using the inverse recursive Lagrangian formulation. The simulation and sensitivity formulations are based on Denavit–Hartenberg transformation matrices. External forces and moments are taken into account in the formulation. The sensitivity information is needed in the optimization-based simulation process. The proposed formulation is demonstrated by calculating sensitivities for the optimal time trajectory planning problem of a two-link manipulator. In addition, sensitivities obtained using the proposed algorithm are compared to those obtained using the closed-form equations of motion. The two sensitivities match quite closely. The lifting motion of the two-link manipulator with external loads is also optimized by using the algorithm developed in this paper. More complex applications of the proposed formulation to digital human motion prediction are presented elsewhere.

**Keywords** Recursive Lagrangian equations · Sensitivities · Optimization · Predictive dynamics

## 1 Introduction

Dynamic motion prediction is a challenging problem because the equations of motion are nonlinear and cannot be solved in a closed form. Robust nonlinear programming algorithms (NLP) have been developed and applied to solve the motion prediction problem. Inverse dynamics is usually adopted in the optimization formulation so that integration of the equations of motion is avoided. Sensitivity of the nonlinear dynamics equations with respect to the state variables is needed in the optimization process to solve the problem efficiently and accurately. Development of sensitivity expressions and their implementation can be tedious. Therefore, more efficient and easy-to-implement algorithms need to be developed.

General dynamics equations have been extensively studied in recent decades in terms of computational efficiency. Uicker (1965) set up the Lagrangian equation for serial chains by using the Denavit–Hartenberg (DH) transformation matrices. This yields a convenient and compact description of the manipulator equations of motion. However, computation of the torques from equations of motion is of the order $O(n^4)$, where $n$ is the number of degrees of freedom (DOF) for the system. Hollerbach (1980) developed recursive Lagrangian dynamics equations that require order $O(n)$ calculations. Toogood (1989) presented an efficient symbolic generator for the recursive Lagrangian inverse dynamics formulation. Furthermore, the recursive Newton–Euler algorithm provides another efficient way to compute torques (Orin et al. 1979; Armstrong 1979; Luh et al. 1980) that also requires order $O(n)$ calculations. Featherstone (1987) used a concise spatial notation to express recursive Newton–Euler equations. Park et al. (1995) derived recursive dynamics formulation based on geometric concepts of Lie groups.

Recently, dynamic motion planning of digital humans has been solved using optimization which requires sensitivity analysis of dynamic equations. It is

Y. Xiang (✉) · J. S. Arora · K. Abdel-Malek
VSR/CCAD/College of Engineering,
The University of Iowa, Iowa City, IA 52242, USA
e-mail: yujxiang@engineering.uiowa.edu

computationally expensive to obtain these sensitivities, especially for a large DOFs model. Researchers have developed algorithms to obtain accurate sensitivity to reduce the computational cost of solving motion optimization problems. Hsieh and Arora (1984) studied both direct differentiation and adjoint variable methods of design sensitivity analysis of mechanical systems to deal with the point-wise constraints. Serban and Haug (1998) presented analytical kinematic and kinetic derivatives required in multibody analysis. Derivative computations were developed in the Cartesian coordinate formulation with Euler parameters. Based on Lie group and Lie algebra theory, Sohl and Bobrow (2001) developed sensitivity algorithm of recursive Newton–Euler equations for branched or tree-topology systems. Kim et al. (1999) presented a class of Newton-type algorithms to analytically compute both the first and second derivatives of the dynamics equations with respect to arbitrary joint variables.

The historical development of multibody system dynamics and its engineering applications were reviewed by Schiehlen (1997), in which various formalisms of multibody dynamics were discussed. Eberhard and Schiehlen (2006) overviewed some fundamentals in multibody dynamics, recursive algorithms and methods for dynamical analysis. Recursive formulation has been shown to be efficient for mechanical systems with large DOFs or articulated topology structures (Bae and Haug 1986; Rein 1995; Anderson and Hsu 2002; Rodríguez et al. 2004). Bae et al. (2001) presented a design sensitivity analysis method based on direct differentiation and generalized recursive formulas. The equations of motion were transformed from Cartesian coordinate system into the relative coordinate system by using the velocity transformations. Anderson and Hsu (2002) developed a fully recursive method to facilitate first-order sensitivity analysis in optimal design problems for multibody systems. The dynamic analysis algorithm was based on the velocity space projection method. An alternative method to calculate sensitivities of a dynamic system is the automatic differentiation procedure (Eberhard and Bischof 1999; Barthelemy and Hall 1995), which can reduce the effort in their implementation. However, automatic differentiation is a pure syntactical tool, and so it is difficult to obtain any insight knowledge about the problem structure within the algorithm (Anderson and Hsu 2002).

Optimization-based motion prediction is a robust methodology to solve trajectory planning for a redundant mechanical system. Such an optimization problem is usually a large scale sparse nonlinear programming (NLP) problem. Accurate sensitivity is a key factor to efficiently achieve an optimal solution. Although finite difference approach can be used to approximate gradients, the computational expense becomes more serious as the number of variables increases (i.e., the number of degrees of freedom). In addition, accuracy of the derivatives can affect convergence of the optimization process, thus leading to further computational expense. Snyman and Berner (1999) optimized a three-link revolute-joint planar manipulator. Input energy and average torque requirement were minimized subject to the joint and the torque limits. Gradient of the objective function was calculated using finite differences. Anderson and Pandy (2001) presented a model with 23 DOFs and 54 muscles for normal symmetric walking on level ground. The objective function was metabolic energy per unit distance, and the parallel computation techniques were used to evaluate gradients by finite differences. Lo et al. (2002) presented a general framework for human motion prediction incorporating inverse recursive Newton–Euler equations with analytical gradients. Although different algorithms for sensitivity of dynamic equation have been studied, limited work is found for inverse recursive Lagrangian formulation with sensitivity for general motion planning problem. By using $4 \times 4$ transformation matrices (DH method), recursive Lagrangian formulation is more efficient and convenient to implement compared to recursive Newton–Euler formulation (Hollerbach 1980). In addition, the Lagrangian formulation is the energy concept for the equations of motion typically defined in the joint space. This results in more convenient calculation for joint torques, especially for a skeletal model, compared to the Newton–Euler formulation which is based on the Cartesian coordinates.

In this study, the problem of motion prediction of a multibody system is formulated as an optimization problem. The recursive Lagrangian dynamics and sensitivity formulation for the system are presented. Implementation aspects of sensitivity calculations for the complex articulated mechanisms are addressed. The developed formulation can systematically treat open-loop, closed loop and branched mechanical systems. In addition the sensitivity analysis needed for the optimization process is easier to implement. The formulation is based on DH transformation matrices and external forces and moments at any point of the mechanism are included in the recursive formulation. The proposed formulation is used to solve an optimal time trajectory planning problem for a two-link manipulator. Initial and final conditions are given. Total travel time is minimized subject to the joint torque limits. The optimal solution is verified with the solution available in the literature. In addition, solution with the proposed formulation is compared to the one with the

closed-form equations of motion. The two solutions match quite closely. The lifting motion of the two-link manipulator with external loads is also optimized by using the algorithm developed in this paper and the results are verified by using commercial multibody dynamics software ADAMS$^{TM}$. Also the algorithm has been used to simulate walking and running of digital human models having many degrees of freedom. These results are presented in separate publications (Xiang et al. 2007; Chung et al. 2007).

## 2 Denavit–Hartenberg method

The Denavit–Hartenberg (DH) method relates the position of a point in one coordinate system to another by using transformation matrices (Denavit and Hartenberg 1955). Since the method is a key component of the present formulation, it is briefly summarized here. For a serial kinematics chain, $\mathbf{q} \in R^n$ is defined as the vector of $n$-generalized coordinates, the joint angles. The position vector of a point of interest in the Cartesian space can be written in terms of the joint angles as $\mathbf{X} = \mathbf{X}(\mathbf{q})$, where $\mathbf{X}(\mathbf{q})$ can be obtained from the multiplication of the $4 \times 4$ homogeneous transformation matrices ${}^{i-1}\mathbf{T}_i$ defined by the DH method as:

$$ {}^{i-1}\mathbf{T}_i = \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1} $$

which relates coordinate frames $i$ and $i-1$, represented by four parameters $\theta_i$, $d_i$, $\alpha_i$, and $a_i$, as shown in Fig. 1.

A revolute or prismatic joint is modeled as a one-DOF motion $q_i$ along local $z_{i-1}$ axis. The joint motion is incorporated in the transformation matrix ${}^{i-1}\mathbf{T}_i$ as in (2):

$$ q_i = \begin{cases} \theta_i & \text{for a revolute joint} \\ d_i & \text{for a prismatic joint} \end{cases} \tag{2} $$

Let us define the augmented $4 \times 1$ vectors ${}^0\mathbf{r}_j$ and $\mathbf{r}_j$ using the global Cartesian vector $\mathbf{X}(\mathbf{q})$ and the local Cartesian vector $\mathbf{X}_j$ as:

$$ {}^0\mathbf{r}_j = \begin{bmatrix} \mathbf{X}(\mathbf{q}) \\ 1 \end{bmatrix}, \mathbf{r}_j = \begin{bmatrix} \mathbf{X}_j \\ 1 \end{bmatrix} \tag{3} $$

where $\mathbf{X}_j$ is the position of the point with respect to the $j$th coordinate system. Using these vectors, ${}^0\mathbf{r}_j$ can be related to $\mathbf{r}_j$ (i.e., the global Cartesian vector $\mathbf{X}(\mathbf{q})$ can be expressed in terms of the local Cartesian vector $\mathbf{X}_j$) as:

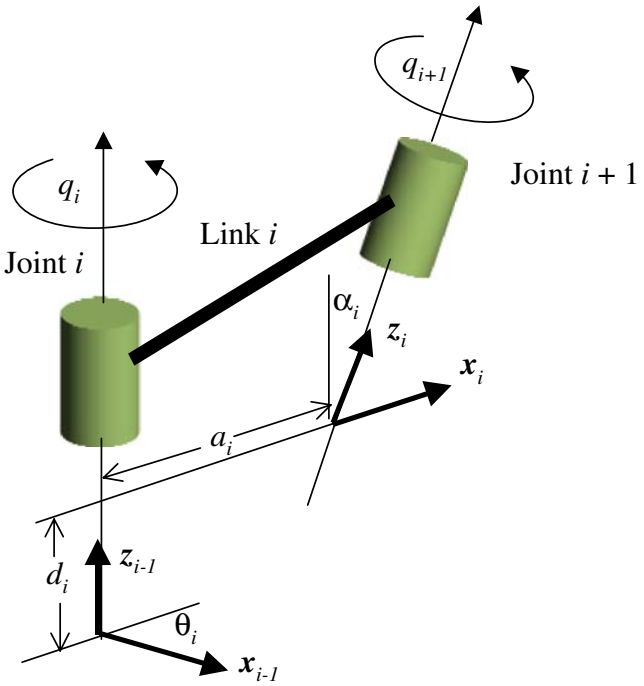


**Fig. 1** Joint coordinate systems

$$ {}^0\mathbf{r}_j = {}^0\mathbf{T}_j(\mathbf{q})\ \mathbf{r}_j \tag{4} $$

where

$$ {}^0\mathbf{T}_j(\mathbf{q}) = \prod_{i=1}^{j} {}^{i-1}\mathbf{T}_i = {}^0\mathbf{T}_1(q_1)\ {}^1\mathbf{T}_2(q_2) \cdots {}^{j-1}\mathbf{T}_j(q_j) \tag{5} $$

## 3 Regular Lagrangian equations

### 3.1 Formulation of regular Lagrangian equation

The regular form of the Lagrangian equation can be written in vector–matrix form (Fu et al. 1987):

$$ \boldsymbol{\tau} = \mathbf{M}(\mathbf{q})\,\ddot{\mathbf{q}} + V(\mathbf{q}, \dot{\mathbf{q}}) + \sum_i \mathbf{J}_i^T m_i \mathbf{g} + \mathbf{J}_s^T \mathbf{f}_s \tag{6} $$

$$ M_{ik}(\mathbf{q}) = \sum_{j=\max(i,k)}^{n} Tr\left( \frac{\partial {}^0\mathbf{T}_j(\mathbf{q})}{\partial q_k} \mathbf{I}_j \left( \frac{\partial {}^0\mathbf{T}_j(\mathbf{q})}{\partial q_i} \right)^T \right) \quad i,k = 1,2,\ldots,n \tag{7} $$

$$ \mathbf{V}_i(\mathbf{q}, \dot{\mathbf{q}}) = \sum_{k=1}^{n} \sum_{m=1}^{n} \sum_{j=\max(i,k,m)}^{n} Tr\left( \frac{\partial^2 {}^0\mathbf{T}_j(\mathbf{q})}{\partial q_k \partial q_m} \mathbf{I}_j \left( \frac{\partial {}^0\mathbf{T}_j(\mathbf{q})}{\partial q_i} \right)^T \right) \times \dot{q}_k \dot{q}_m \quad i,k,m = 1,2,\ldots,n \tag{8} $$

$$\mathbf{J}_i = \frac{\partial^0 \mathbf{T}_i}{\partial q_i} \overline{r_i} \tag{9}$$

$$\mathbf{J}_s = \frac{\partial^0 \mathbf{T}_s}{\partial q_s} \overline{r_s} \tag{10}$$

where $\mathbf{J}_i$ is the Jacobian matrix for link $i$ and $\overline{r_i}$ is the position of the center of mass of link $i$ with respect to the $i$th coordinate system; $\mathbf{I}_j$ is the augmented inertia matrix for link $j$; $\mathbf{J}_s$ is the Jacobian matrix for the external load $\mathbf{f}_s$ and $\overline{r_s}$ is the position where external load applied in the $s$th coordinate system. $Tr[\cdots]$ denotes the trace operation and $(\cdots)^T$ denotes the matrix transpose.

### 3.2 Sensitivity analysis

We note from (6)–(10) that the regular Lagrangian equations are coupled, nonlinear, and second-order differential equations. $\mathbf{M}(\mathbf{q})$ is an $n \times n$ matrix, and $\mathbf{V}(\mathbf{q}, \dot{\mathbf{q}})$ is an $n \times 1$ vector. Each term involves summation and state variables. Direct sensitivity analysis gives the $n \times n$ sensitivity matrix as:

$$\frac{\partial \boldsymbol{\tau}}{\partial \mathbf{q}} = \frac{\partial \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}}}{\partial \mathbf{q}} + \frac{\partial \mathbf{V}(\mathbf{q}, \dot{\mathbf{q}})}{\partial \mathbf{q}} + \frac{\partial \sum_i \mathbf{J}_i^T m_i \mathbf{g}}{\partial \mathbf{q}} + \frac{\partial \mathbf{J}_s^T \mathbf{f}_s}{\partial \mathbf{q}} \tag{11}$$

## 4 Recursive Lagrangian equations

### 4.1 Forward recursive kinematics

We can define $4 \times 4$ matrices $\mathbf{A}_j$, $\mathbf{B}_j$, $\mathbf{C}_j$ as recursive position, velocity, and acceleration transformation matrices, respectively, for the $j$th joint. Given the link transformation matrix ($\mathbf{T}_j$) and the kinematics state variables for each joint ($q_j$, $\dot{q}_j$, and $\ddot{q}_j$), we have for $j = 1$ to $n$:

$$\mathbf{A}_j = \mathbf{T}_1 \mathbf{T}_2 \mathbf{T}_3 \cdots \mathbf{T}_j = \mathbf{A}_{j-1} \mathbf{T}_j \tag{12}$$

$$\mathbf{B}_j = \dot{\mathbf{A}}_j = \mathbf{B}_{j-1} \mathbf{T}_j + \mathbf{A}_{j-1} \frac{\partial \mathbf{T}_j}{\partial q_j} \dot{q}_j \tag{13}$$

$$\mathbf{C}_j = \dot{\mathbf{B}}_j = \ddot{\mathbf{A}}_j = \mathbf{C}_{j-1} \mathbf{T}_j + 2\mathbf{B}_{j-1} \frac{\partial \mathbf{T}_j}{\partial q_j} \dot{q}_j + \mathbf{A}_{j-1} \frac{\partial^2 \mathbf{T}_j}{\partial q_j^2} \dot{q}_j^2 + \mathbf{A}_{j-1} \frac{\partial \mathbf{T}_j}{\partial q_j} \ddot{q}_j \tag{14}$$

where $\mathbf{A}_0 = \mathbf{1}$ (identity matrix) and $\mathbf{B}_0 = \mathbf{C}_0 = \mathbf{0}$. For the sake of simplicity, $\mathbf{T}_j$ is used to represent ${}^{j-1}\mathbf{T}_j$.

After obtaining all the transformation matrices $\mathbf{A}_j$, $\mathbf{B}_j$, $\mathbf{C}_j$, the global position, velocity, and acceleration of a point in Cartesian coordinates can be calculated as:

$${}^0\mathbf{r}_j = \mathbf{A}_j \mathbf{r}_j;\ {}^0\dot{\mathbf{r}}_j = \mathbf{B}_j \mathbf{r}_j;\ {}^0\ddot{\mathbf{r}}_j = \mathbf{C}_j \mathbf{r}_j \tag{15}$$

where $\mathbf{r}_j$ contains the augmented local coordinates of the point in $j$th coordinate system.

### 4.2 Backward recursive dynamics

Based on forward recursive kinematics, the backward recursion for dynamic analysis is accomplished by defining $4 \times 4$ transformation matrix $\mathbf{D}_i$ and $4 \times 1$ transformation vectors $\mathbf{E}_i$, $\mathbf{F}_i$, and $\mathbf{G}_i$ as follows.

Given the mass and inertia properties of each link, and the external force $\mathbf{f}_k^T = \begin{bmatrix} {}^k f_x & {}^k f_y & {}^k f_z & 0 \end{bmatrix}$ and the moment $\mathbf{h}_k^T = \begin{bmatrix} {}^k h_x & {}^k h_y & {}^k h_z & 0 \end{bmatrix}$ for the link $k$ defined in the global coordinate system, then the joint actuation torques $\tau_i$ are computed for $i = n$ to 1 as (Hollerbach 1980):

$$\tau_i = tr\left[\frac{\partial \mathbf{A}_i}{\partial q_i} \mathbf{D}_i\right] - \mathbf{g}^T \frac{\partial \mathbf{A}_i}{\partial q_i} \mathbf{E}_i - \mathbf{f}_k^T \frac{\partial \mathbf{A}_i}{\partial q_i} \mathbf{F}_i - \mathbf{G}_i^T \mathbf{A}_{i-1} \mathbf{z}_0 \tag{16}$$

where:

$$\mathbf{D}_i = \mathbf{I}_i \mathbf{C}_i^T + \mathbf{T}_{i+1} \mathbf{D}_{i+1} \tag{17}$$

$$\mathbf{E}_i = m_i\, {}^i\mathbf{r}_i + \mathbf{T}_{i+1} \mathbf{E}_{i+1} \tag{18}$$

$$\mathbf{F}_i = {}^k\mathbf{r}_f \delta_{ik} + \mathbf{T}_{i+1} \mathbf{F}_{i+1} \tag{19}$$

$$\mathbf{G}_i = \mathbf{h}_k \delta_{ik} + \mathbf{G}_{i+1} \tag{20}$$

with $\mathbf{D}_{n+1} = \mathbf{0}$ and $\mathbf{E}_{n+1} = \mathbf{F}_{n+1} = \mathbf{G}_{n+1} = \mathbf{0}$; $\mathbf{I}_i$ is the inertia matrix for link $i$; $m_i$ is the mass of link $i$; $\mathbf{g}$ is the gravity vector; ${}^i\mathbf{r}_i$ is the location of center of mass of link $i$ in the local frame $i$; ${}^k\mathbf{r}_f$ is position of the external force in the local frame $k$; $\mathbf{z}_0 = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}^T$ for a revolute joint and $\mathbf{z}_0 = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T$ for a prismatic joint. $\delta_{ik}$ is Kronecker delta.

The first term in torque expression is the inertia and Coriolis torque, the second term denotes the torque of force due to gravity, the third term is the torque due to external force, and the fourth term represents the torque due to external moment.

### 4.3 Sensitivity analysis

The derivatives, $\frac{\partial \tau_i}{\partial q_k}$, $\frac{\partial \tau_i}{\partial \dot{q}_k}$, $\frac{\partial \tau_i}{\partial \ddot{q}_k}$ ($i$ = 1 to $n$; $k$ = 1 to $n$), can be evaluated for a mechanical system in a recursive way using the foregoing recursive Lagrangian dynamics formulation.

*4.3.1 Kinematics sensitivity analysis*

For a given point, sensitivity of position, velocity, and acceleration with respect to state variables relates to transformation matrices **A**, **B**, and **C**.

$$\frac{\partial \mathbf{A}_i}{\partial q_k} = \begin{cases} \frac{\partial \mathbf{A}_{i-1}}{\partial q_k}\mathbf{T}_i & (k < i) \\ \mathbf{A}_{i-1}\frac{\partial \mathbf{T}_i}{\partial q_k} & (k = i) \\ \mathbf{0} & (k > i) \end{cases} \tag{21}$$

$$\frac{\partial \mathbf{B}_i}{\partial q_k} = \begin{cases} \frac{\partial \mathbf{B}_{i-1}}{\partial q_k}\mathbf{T}_i + \frac{\partial \mathbf{A}_{i-1}}{\partial q_k}\frac{\partial \mathbf{T}_i}{\partial q_i}\dot{q}_i & (k < i) \\ \mathbf{B}_{i-1}\frac{\partial \mathbf{T}_i}{\partial q_k} + \mathbf{A}_{i-1}\frac{\partial^2 \mathbf{T}_i}{\partial q_k^2}\dot{q}_i & (k = i) \\ \mathbf{0} & (k > i) \end{cases} \tag{22}$$

$$\frac{\partial \mathbf{B}_i}{\partial \dot{q}_k} = \begin{cases} \frac{\partial \mathbf{B}_{i-1}}{\partial \dot{q}_k}\mathbf{T}_i & (k < i) \\ \mathbf{A}_{i-1}\frac{\partial \mathbf{T}_i}{\partial q_k} & (k = i) \\ \mathbf{0} & (k > i) \end{cases} \tag{23}$$

$$\frac{\partial \mathbf{C}_i}{\partial q_k} = \begin{cases} \frac{\partial \mathbf{C}_{i-1}}{\partial q_k}\mathbf{T}_i + 2\frac{\partial \mathbf{B}_{i-1}}{\partial q_k}\frac{\partial \mathbf{T}_i}{\partial q_i}\dot{q}_i + \frac{\partial \mathbf{A}_{i-1}}{\partial q_k}\frac{\partial^2 \mathbf{T}_i}{\partial q_i^2}\dot{q}_i^2 \\ \qquad + \frac{\partial \mathbf{A}_{i-1}}{\partial q_k}\frac{\partial \mathbf{T}_i}{\partial q_i}\ddot{q}_i & (k < i) \\ \mathbf{C}_{i-1}\frac{\partial \mathbf{T}_i}{\partial q_k} + 2\mathbf{B}_{i-1}\frac{\partial^2 \mathbf{T}_i}{\partial q_k^2}\dot{q}_i + \mathbf{A}_{i-1}\frac{\partial^3 \mathbf{T}_i}{\partial q_k^3}\dot{q}_i^2 \\ \qquad + \mathbf{A}_{i-1}\frac{\partial^2 \mathbf{T}_i}{\partial q_k^2}\ddot{q}_i & (k = i) \\ \mathbf{0} & (k > i) \end{cases} \tag{24}$$

$$\frac{\partial \mathbf{C}_i}{\partial \dot{q}_k} = \begin{cases} \frac{\partial \mathbf{C}_{i-1}}{\partial \dot{q}_k}\mathbf{T}_i + 2\frac{\partial \mathbf{B}_{i-1}}{\partial \dot{q}_k}\frac{\partial \mathbf{T}_i}{\partial q_i}\dot{q}_i & (k < i) \\ 2\mathbf{B}_{i-1}\frac{\partial \mathbf{T}_i}{\partial q_k} + 2\mathbf{A}_{i-1}\frac{\partial^2 \mathbf{T}_i}{\partial q_k^2}\dot{q}_i & (k = i) \\ \mathbf{0} & (k > i) \end{cases} \tag{25}$$

$$\frac{\partial \mathbf{C}_i}{\partial \ddot{q}_k} = \begin{cases} \frac{\partial \mathbf{C}_{i-1}}{\partial \ddot{q}_k}\mathbf{T}_i & (k < i) \\ \mathbf{A}_{i-1}\frac{\partial \mathbf{T}_i}{\partial q_k} & (k = i) \\ \mathbf{0} & (k > i) \end{cases} \tag{26}$$

The forward recursive kinematics sensitivity equations are implemented as follows:

Input state variables $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}$ and initial condition $\mathbf{A}[0] = \mathbf{1}, \mathbf{B}[0] = \mathbf{0}, \mathbf{C}[0] = \mathbf{0}$

**Do** $i$=1, $n$

(1) Obtain $q_i, \dot{q}_i, \ddot{q}_i$

(2) Calculate and store $\mathbf{T}_i(q_i), \partial \mathbf{T}_i(q_i)/\partial q_i, \partial^2 \mathbf{T}_i(q_i)/\partial q_i^2, \partial^3 \mathbf{T}_i(q_i)/\partial q_i^3$

(3) Calculate and store $\mathbf{A}_i$

(4) Calculate and store $\mathbf{B}_i$

(5) Calculate and store $\mathbf{C}_i$

**Do** $j = 1, i$

(a) Calculate and store $\partial \mathbf{A}_i/\partial q_j$

(b) Calculate and store $\partial \mathbf{B}_i/\partial q_j, \partial \mathbf{B}_i/\partial \dot{q}_j$

(c) Calculate and store $\partial \mathbf{C}_i/\partial q_j, \partial \mathbf{C}_i/\partial \dot{q}_j, \partial \mathbf{C}_i/\partial \ddot{q}_j$

**End Do**

**End Do**

*4.3.2 Dynamics sensitivity analysis*

Sensitivity of the joint torque with respect to the state variables involves **D**, **E**, **F**, and **G**, which correspond to inertia and Coriolis, gravity, external force, and external moment, respectively.

$$\frac{\partial \mathbf{D}_i}{\partial q_k} = \begin{cases} \mathbf{I}_i\frac{\partial \mathbf{C}_i^T}{\partial q_k} + \mathbf{T}_{i+1}\frac{\partial \mathbf{D}_{i+1}}{\partial q_k} & (k \le i) \\ \frac{\partial \mathbf{T}_{i+1}}{\partial q_k}\mathbf{D}_{i+1} + \mathbf{T}_{i+1}\frac{\partial \mathbf{D}_{i+1}}{\partial q_k} & (k = i+1) \\ \mathbf{T}_{i+1}\frac{\partial \mathbf{D}_{i+1}}{\partial q_k} & (k > i+1) \end{cases} \tag{27}$$

$$\frac{\partial \mathbf{D}_i}{\partial \dot{q}_k} = \begin{cases} \mathbf{I}_i\frac{\partial \mathbf{C}_i^T}{\partial \dot{q}_k} + \mathbf{T}_{i+1}\frac{\partial \mathbf{D}_{i+1}}{\partial \dot{q}_k} & (k \le i) \\ \mathbf{T}_{i+1}\frac{\partial \mathbf{D}_{i+1}}{\partial \dot{q}_k} & (k > i) \end{cases} \tag{28}$$

$$\frac{\partial \mathbf{D}_i}{\partial \ddot{q}_k} = \begin{cases} \mathbf{I}_i\frac{\partial \mathbf{C}_i^T}{\partial \ddot{q}_k} + \mathbf{T}_{i+1}\frac{\partial \mathbf{D}_{i+1}}{\partial \ddot{q}_k} & (k \le i) \\ \mathbf{T}_{i+1}\frac{\partial \mathbf{D}_{i+1}}{\partial \ddot{q}_k} & (k > i) \end{cases} \tag{29}$$

$$\frac{\partial \mathbf{E}_i}{\partial q_k} = \begin{cases} \mathbf{0} & (k \le i) \\ \frac{\partial \mathbf{T}_{i+1}}{\partial q_k}\mathbf{E}_{i+1} + \mathbf{T}_{i+1}\frac{\partial \mathbf{E}_{i+1}}{\partial q_k} & (k = i+1) \\ \mathbf{T}_{i+1}\frac{\partial \mathbf{E}_{i+1}}{\partial q_k} & (k > i+1) \end{cases} \tag{30}$$

$$\frac{\partial \mathbf{F}_i}{\partial q_k} = \begin{cases} \mathbf{0} & (k \le i) \\ \frac{\partial \mathbf{T}_{i+1}}{\partial q_k}\mathbf{F}_{i+1} + \mathbf{T}_{i+1}\frac{\partial \mathbf{F}_{i+1}}{\partial q_k} & (k = i+1) \\ \mathbf{T}_{i+1}\frac{\partial \mathbf{F}_{i+1}}{\partial q_k} & (k > i+1) \end{cases} \tag{31}$$

$$\frac{\partial \mathbf{G}_i}{\partial q_k} = \mathbf{0} \tag{32}$$

$$\frac{\partial \tau_i}{\partial q_k} = \begin{cases} tr\left(\frac{\partial^2 \mathbf{A}_i}{\partial q_i \partial q_k}\mathbf{D}_i + \frac{\partial \mathbf{A}_i}{\partial q_i}\frac{\partial \mathbf{D}_i}{\partial q_k}\right) - \mathbf{g}^T\frac{\partial^2 \mathbf{A}_i}{\partial q_i \partial q_k}\mathbf{E}_i \\ \quad - \mathbf{f}^T\frac{\partial^2 \mathbf{A}_i}{\partial q_i \partial q_k}\mathbf{F}_i - \mathbf{G}_i^T\frac{\partial \mathbf{A}_{i-1}}{\partial q_k}\mathbf{z}_0 & (k \le i) \\ tr\left(\frac{\partial \mathbf{A}_i}{\partial q_i}\frac{\partial \mathbf{D}_i}{\partial q_k}\right) - \mathbf{g}^T\frac{\partial \mathbf{A}_i}{\partial q_i}\frac{\partial \mathbf{E}_i}{\partial q_k} - \mathbf{f}^T\frac{\partial \mathbf{A}_i}{\partial q_i}\frac{\partial \mathbf{F}_i}{\partial q_k} & (k > i) \end{cases} \tag{33}$$

$$\frac{\partial \tau_i}{\partial \dot{q}_k} = tr\left(\frac{\partial \mathbf{A}_i}{\partial q_i}\frac{\partial \mathbf{D}_i}{\partial \dot{q}_k}\right) \tag{34}$$

$$\frac{\partial \tau_i}{\partial \ddot{q}_k} = tr\left(\frac{\partial \mathbf{A}_i}{\partial q_i}\frac{\partial \mathbf{D}_i}{\partial \ddot{q}_k}\right) \tag{35}$$

Thus, the gradients of torque with respect to state variables are obtained through (33) to (35).

The backward recursive dynamics sensitivity equations are implemented as follows:

---

Input state variables $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}$ and final condition $\mathbf{D}[n+1] = \mathbf{0}, \mathbf{E}[n+1] = \mathbf{0}, \mathbf{F}[n+1] = \mathbf{0}, \mathbf{G}[n+1] = \mathbf{0}$ and using the stored information calculated from forward recursive sensitivity algorithm

**Do** $i = n, 1$

(1) Calculate and store $\mathbf{D}_i, \mathbf{E}_i, \mathbf{F}_i, \mathbf{G}_i$

(2) Calculate and store $\tau_i$

**Do** $j = 1, i$

Calculate and store $\partial^2 \mathbf{A}_i / \partial q_i \partial q_j$

**End Do**

**Do** $k = 1, n$

(a) Calculate and store $\partial \mathbf{D}_i / \partial q_k$, $\partial \mathbf{D}_i / \partial \dot{q}_k$, $\partial \mathbf{D}_i / \partial \ddot{q}_k$

(b) Calculate and store $\partial \mathbf{E}_i / \partial q_k$

(c) Calculate and store $\partial \mathbf{F}_i / \partial q_k$

(d) Calculate and store $\partial \tau_i / \partial q_k$, $\partial \tau_i / \partial \dot{q}_k$, $\partial \tau_i / \partial \ddot{q}_k$

**End Do**

**End Do**

---

### 4.4 Comparison of sensitivity analyses

Direct sensitivity analyses from the regular Lagrangian equations are challenging and inefficient because all terms are coupled together. For a large-DOF mechanism, this process is almost impossible. The computation cost of regular Lagrangian equations is of order $O(n^4)$, and sensitivity cost is even larger. Moreover, to deal with closed-loop or branched chains, the regular Lagrangian equations of the system must be rewritten, and sensitivity analysis has to go through each term in the coefficient matrix.

In contrast, the recursive formulation is convenient for computing sensitivities. Dynamic equations of the system are set up in a recursive way. Sensitivity information about a joint involves two adjacent joints. The computation cost is therefore reduced to the order of $O(n)$. In addition, the recursive algorithm is suitable for computer implementation. For a system with small DOF, the total computational time with different formulations of equations of motion may not be too different. However, for a model with a large number of DOF, the number of calculations can be significantly different. This can have a significant impact on the efficiency of the entire optimization process. It has been concluded in the literature that the recursive formulation is the most suitable for large scale mechanical system (Hollerbach 1980; Luh et al. 1980; Toogood 1989; Bae et al. 2001). By introducing the parent link and child link concepts, extension of the algorithm to branched chains is straightforward. The parent link transfers forward kinematics (**A**, **B**, **C**) to each child link, and every child link passes dynamics (**D**, **E**, **F**, **G**) back to the parent link, as shown in Fig. 2 (for clarity, only **C** and **D** matrices are shown in the figure).

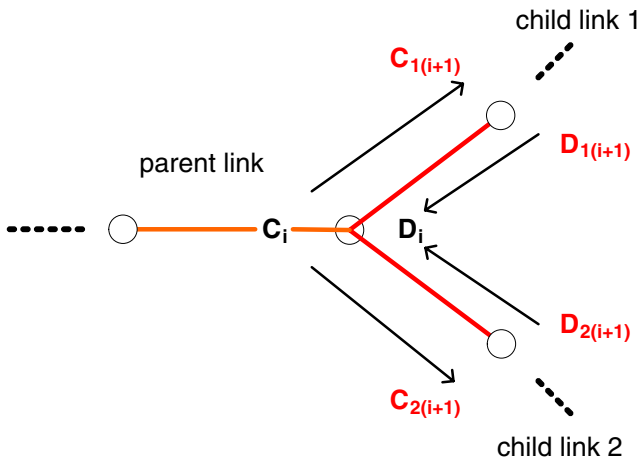


**Fig. 2** Branched chains system

Open-loop algorithm:

$$\begin{cases} \mathbf{A}\,[0] = \mathbf{1};\ \mathbf{B}\,[0] = \mathbf{0};\ \mathbf{C}\,[0] = \mathbf{0} \\ \mathbf{D}\,[n+1] = \mathbf{0};\ \mathbf{E}\,[n+1] = \mathbf{0};\ \mathbf{F}\,[n+1] = \mathbf{0};\ \mathbf{G}\,[n+1] = \mathbf{0} \end{cases} \tag{36}$$

The basis is always fixed in the inertial coordinate and moving basis can be dealt with by introducing more degrees of freedom (virtual joint with zero mass and inertia) which relate moving basis to the fixed origin in inertial coordinate.

Branched-chain algorithm:

For forward kinematics, the child branch receives kinematics of the parent branch; for backward dynamics, the parent branch accumulates dynamics of the child branches as follows:

$$\begin{cases} \mathbf{A}_{\text{child}}\,[0] = \mathbf{A}_{\text{parent}}\,[n_p] \\ \mathbf{B}_{\text{child}}\,[0] = \mathbf{B}_{\text{parent}}\,[n_p] \\ \mathbf{C}_{\text{child}}\,[0] = \mathbf{C}_{\text{parent}}\,[n_p] \end{cases} \tag{37a}$$

$$\begin{cases} \mathbf{D}_{\text{parent}}\,[n_p+1] = \sum_{i=1}^{m} \mathbf{D}_{\text{child}(i)}\,[1] \\ \mathbf{E}_{\text{parent}}\,[n_p+1] = \sum_{i=1}^{m} \mathbf{E}_{\text{child}(i)}\,[1] \\ \mathbf{F}_{\text{parent}}\,[n_p+1] = \sum_{i=1}^{m} \mathbf{F}_{\text{child}(i)}\,[1] \\ \mathbf{G}_{\text{parent}}\,[n_p+1] = \sum_{i=1}^{m} \mathbf{G}_{\text{child}(i)}\,[1] \end{cases} \tag{37b}$$

where $n_p$ is the number of degrees of freedom in parent branch and $m$ is the number of child branches connected to the parent branch; $\mathbf{A}_{\text{child}}\,[0]$ stores the parent branch kinematics and $\mathbf{D}_{\text{parent}}\,[n_p+1]$ stores the child branch dynamics as shown in Fig. 3.

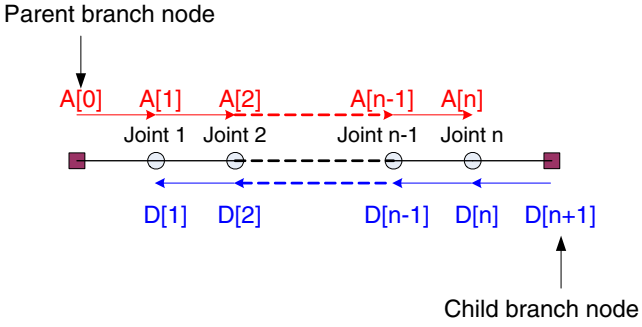


**Fig. 3** Connectivity between parent branch and child branch

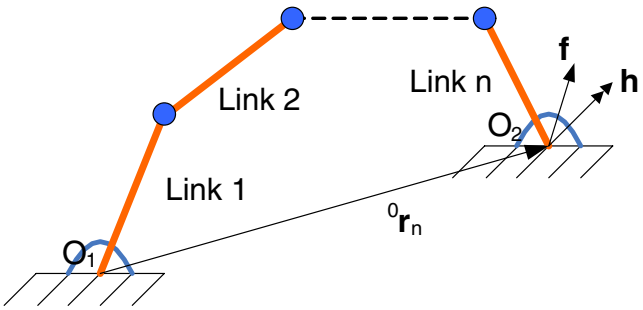


**Fig. 4** Closed-loop system

The derived algorithm can also be applied to closed-loop system by introducing cut-joint with external forces and position constraint as follows:

$$\begin{cases} \mathbf{A}\,[0] = \mathbf{1};\ \mathbf{B}\,[0] = \mathbf{0};\ \mathbf{C}\,[0] = \mathbf{0} \\ \mathbf{D}\,[n+1] = \mathbf{0};\ \mathbf{E}\,[n+1] = \mathbf{0};\ \mathbf{F}\,[n+1] = \mathbf{0};\ \mathbf{G}\,[n+1] = \mathbf{0} \end{cases} \tag{38a}$$

$$\begin{cases} g\left({}^0\mathbf{r}_n\right) \le 0 \ \textit{where}\ {}^0\mathbf{r}_n = \mathbf{A}\,[n]\,\mathbf{r}_n \\ \mathbf{f}_n^T = \left[{}^n f_x\ {}^n f_y\ {}^n f_z\ 0\right] \\ \mathbf{h}_n^T = \left[{}^n h_x\ {}^n h_y\ {}^n h_z\ 0\right] \end{cases} \tag{38b}$$

where ${}^0\mathbf{r}_n$ is the Cartesian coordinates of the end-effector and $g\,(\cdots)$is the constraint on it; $\mathbf{f}_n$ and $\mathbf{h}_n$ are external force and moment acting at the end-effector as depicted in Fig. 4.

## 5 Joint profile discretization

A joint profile $q\ (t)$ is parameterized by using uniform cubic $B$-splines as follows:

$$q\,(\mathbf{t}, \mathbf{P}) = \sum_{i=1}^{m} B_i\,(\mathbf{t})\ p_i \qquad 0 \le t \le T \tag{39}$$

where $B_i\,(\mathbf{t})$ are the basis functions, $\mathbf{t} = \{t_0, \ldots, t_s\}$ is the knot vector, and $\mathbf{P} = \{p_1, \ldots, p_m\}$ is the control point vector. With this representation, the control points become the optimization variables (also called the design variables). $B$-spline interpolation has many important properties, such as continuity, differentiability, and local control. These properties, especially differentiability and local control, make $B$-splines competent to represent joint angle trajectories, which require smoothness and flexibility (Wang et al. 2007).

The $B$-spline basis functions are uniquely determined by knot vector $\mathbf{t}$, which is evenly spaced on the time interval $[0\ T]$ with time step $\Delta t$, as shown in (40):

$$t_{i+1} = t_i + \Delta t, \qquad \Delta t = \frac{T}{s}, \qquad i = 0, \ldots, s-1 \tag{40}$$

where $s$ is the number of discretized segments.

$q$, $\dot{q}$, and $\ddot{q}$ calculated from (39) are functions of $\mathbf{t}$ and $\mathbf{P}$; therefore torque $\tau = \tau(\mathbf{t}, \mathbf{P})$ is an explicit function of the knot vector and control points from the equation of motion. Thus, the derivatives of a torque $\tau$ with respect to the control points and knot points can be computed using the chain rule as:

$$\frac{\partial \tau}{\partial p_i} = \frac{\partial \tau}{\partial q}\frac{\partial q}{\partial p_i} + \frac{\partial \tau}{\partial \dot{q}}\frac{\partial \dot{q}}{\partial p_i} + \frac{\partial \tau}{\partial \ddot{q}}\frac{\partial \ddot{q}}{\partial p_i} \tag{41}$$

$$\frac{\partial \tau}{\partial t_i} = \frac{\partial \tau}{\partial q}\frac{\partial q}{\partial t_i} + \frac{\partial \tau}{\partial \dot{q}}\frac{\partial \dot{q}}{\partial t_i} + \frac{\partial \tau}{\partial \ddot{q}}\frac{\partial \ddot{q}}{\partial t_i} \tag{42}$$

## 6 Numerical examples

### 6.1 Description of the problem

The manipulator under study consists of two links whose lengths are $L_1$ and $L_2$ and moments of inertia are $I_1$ and $I_2$ as shown in Fig. 5. $q_1$ and $q_2$ are the relative joint angles that are controlled by the joint actuator torques $\tau_1$ and $\tau_2$. The two links are considered rigid, and the relative joint angles are selected as independent generalized coordinates. The manipulator is assumed to lie in the vertical plane. Actuator torques drive the mechanism from the initial position ($q_1$ (0), $q_2$ (0)) to the final position ($q_1$ ($T$), $q_2$ ($T$)) in the time interval $T$. In addition, the manipulator is at rest at the initial and final points. The data for the manipulator are given as shown in Table 1.

Two optimization problems are implemented. The first problem is the time-optimal trajectory-planning design without gravity and external forces. The reason to use this example is to validate the numerical results; solutions of this well-studied example are readily available in the literature (Dissanayake et al. 1991; Wang et al. 2005). The second problem is to optimize a lifting motion with a mixed performance criterion. Both gravity and external force are considered. Sensitivity results with the recursive algorithm and the closed-form formulation are numerically compared.

For the parameterized optimization, the travel time $T$ is evenly discretized into 20 segments with $\Delta t = T/20$. The number of control points for each joint angle is taken as 22 so that the total number of control points is 44. Initial estimates of design variables are obtained by linear interpolation between the initial and final joint angles. The optimization problems are solved using the SNOPT program which uses a sequential quadratic programming (SQP) algorithm (Gill et al. 2002).

**Table 1** Parameters for the two-link manipulator

| Parameters | |
|---|---|
| Length | $L_1 = L_2 = 0.4$ m |
| Mass | $m_1 = m_2 = 0.5$ kg |
| Inertia | $I_1 = I_2 = 0.1$ kgm$^2$ |
| Center of Mass of Link 1 | ${}^1r_1 = (-L_1/2, 0)$ |
| Center of Mass of Link 2 | ${}^2r_2 = (-L_2/2, 0)$ |
| Torque upper bound | $\tau^U = 10$ Nm |
| Torque lower bound | $\tau^L = -10$ Nm |

### 6.2 Recursive Lagrangian sensitivity equations for the two-link manipulator

DH parameters of the planar two-link manipulator are described in Fig. 6 and Table 2.

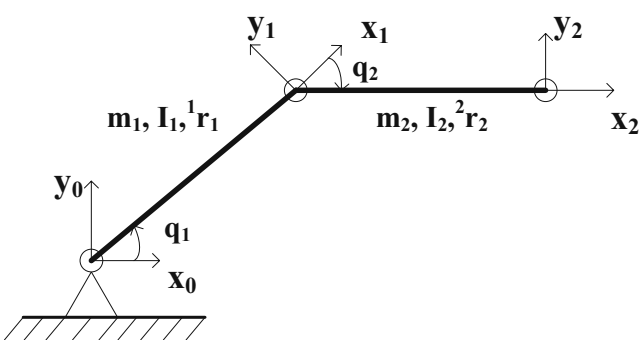


**Fig. 5** Two-link manipulator in vertical plane

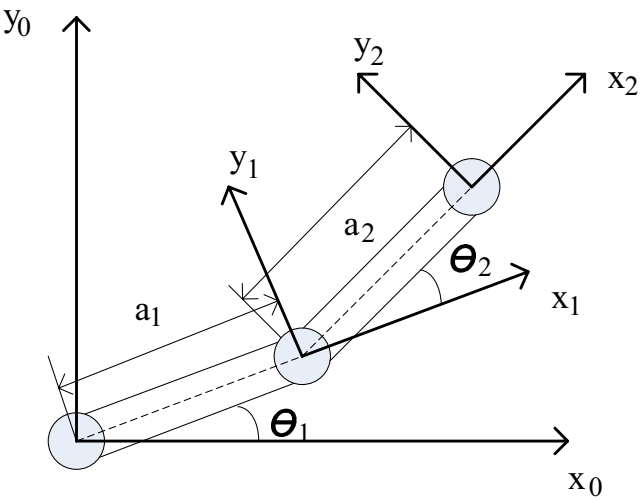


**Fig. 6** DH coordinates for the two-link manipulator

**Table 2** DH parameters for the two-link manipulator

| LINK | $\theta_i$ | $d_i$ | $a_i$ | $\alpha_i$ |
|---|---|---|---|---|
| 1 | $\theta_1$ | 0 | $L_1$ | 0 |
| 2 | $\theta_2$ | 0 | $L_2$ | 0 |

The recursive Lagrangian dynamics and sensitivity equations of the two-link rigid manipulator are implemented as follows:

Transformation matrices:

$$\mathbf{T}_1 = \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & L_1\cos\theta_1 \\ \sin\theta_1 & \cos\theta_1 & 0 & L_1\sin\theta_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix};$$
$$\mathbf{T}_2 = \begin{pmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & L_2\cos\theta_2 \\ \sin\theta_2 & \cos\theta_2 & 0 & L_2\sin\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{43}$$

Forward recursive kinematics:

$$\mathbf{A}_1 = \mathbf{T}_1; \qquad \mathbf{A}_2 = \mathbf{A}_1\mathbf{T}_2 \tag{44}$$

$$\mathbf{B}_1 = \frac{\partial \mathbf{T}_1}{\partial \theta_1}\dot{\theta}_1; \qquad \mathbf{B}_2 = \mathbf{B}_1\mathbf{T}_2 + \mathbf{A}_1\frac{\partial \mathbf{T}_2}{\partial \theta_2}\dot{\theta}_2 \tag{45}$$

$$\mathbf{C}_1 = \frac{\partial^2 \mathbf{T}_1}{\partial \theta_1^2}\dot{\theta}_1^2 + \frac{\partial \mathbf{T}_1}{\partial \theta_1}\ddot{\theta}_1;$$

$$\mathbf{C}_2 = \mathbf{C}_1\mathbf{T}_2 + 2\mathbf{B}_1\frac{\partial \mathbf{T}_2}{\partial \theta_2}\dot{\theta}_2 + \mathbf{A}_1\frac{\partial^2 \mathbf{T}_2}{\partial \theta_2^2}\dot{\theta}_2^2 + \mathbf{A}_1\frac{\partial \mathbf{T}_2}{\partial \theta_2}\ddot{\theta}_2 \tag{46}$$

Backward recursive dynamics:

$$\mathbf{I}_2 = \begin{pmatrix} I_2 + m_2(l_2 - L_2)^2 & 0 & 0 & m_2(l_2 - L_2) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ m_2(l_2 - L_2) & 0 & 0 & m_2 \end{pmatrix}; \tag{47a}$$

$$\mathbf{I}_1 = \begin{pmatrix} I_1 + m_1(l_1 - L_1)^2 & 0 & 0 & m_1(l_1 - L_1) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ m_1(l_1 - L_1) & 0 & 0 & m_1 \end{pmatrix} \tag{47b}$$

$$\mathbf{D}_2 = \mathbf{I}_2\mathbf{C}_2^T; \mathbf{D}_1 = \mathbf{I}_1\mathbf{C}_1^T + \mathbf{T}_2\mathbf{D}_2 \tag{48}$$

$$\mathbf{E}_2 = m_2\ (l_2 - L_2\ 0\ 0\ 1)^T;$$

$$\mathbf{E}_1 = m_1\ (l_1 - L_1\ 0\ 0\ 1)^T + \mathbf{T}_2\mathbf{E}_2 \tag{49}$$

$$\mathbf{F}_2 = (\,0\ 0\ 0\ 1\,)^T; \mathbf{F}_1 = \mathbf{T}_2\mathbf{F}_2 \tag{50}$$

$$\mathbf{g} = (0\ -g\ 0\ 0)^T;\ \mathbf{f}_2 = (0\ -f\ 0\ 0)^T \tag{51}$$

Torques:

$$\begin{aligned}\tau_1 &= tr\left[\frac{\partial \mathbf{A}_1}{\partial q_1}\mathbf{D}_1\right] - \mathbf{g}^T\frac{\partial \mathbf{A}_1}{\partial q_1}\mathbf{E}_1 - \mathbf{f}_2^T\frac{\partial \mathbf{A}_1}{\partial q_1}\mathbf{F}_1 \\ &= \left(I_1 + I_2 + m_1l_1^2 + m_2\left(L_1^2 + l_2^2 + 2L_1l_2\cos\theta_2\right)\right)\ddot{\theta}_1 \\ &\quad + \left(I_2 + m_2l_2^2 + m_2L_1l_2\cos\theta_2\right)\ddot{\theta}_2 - 2m_2L_1l_2\dot{\theta}_1\dot{\theta}_2\sin\theta_2 \\ &\quad - m_2L_1l_2\dot{\theta}_2^2\sin\theta_2 + m_2gl_2\cos(\theta_1 + \theta_2) \\ &\quad + m_1gl_1\cos\theta_1 + m_2gL_1\cos\theta_1 \\ &\quad + fL_2\cos(\theta_1 + \theta_2) + fL_1\cos\theta_1 \end{aligned} \tag{52}$$

$$\begin{aligned}\tau_2 &= tr\left[\frac{\partial \mathbf{A}_2}{\partial q_2}\mathbf{D}_2\right] - \mathbf{g}^T\frac{\partial \mathbf{A}_2}{\partial q_2}\mathbf{E}_2 - \mathbf{f}_2^T\frac{\partial \mathbf{A}_2}{\partial q_2}\mathbf{F}_2 \\ &= \left(I_2 + m_2l_2^2\right)\ddot{\theta}_2 + \left(I_2 + m_2l_2^2 + m_2L_1l_2\cos\theta_2\right)\ddot{\theta}_1 \\ &\quad + m_2L_1l_2\dot{\theta}_1^2\sin\theta_2 + m_2gl_2\cos(\theta_1 + \theta_2) \\ &\quad + fL_2\cos(\theta_1 + \theta_2) \end{aligned} \tag{53}$$

Explicit gradients of torque with respect to state variables are derived as follows:

$$\begin{aligned}\frac{\partial \tau_1}{\partial \theta_1} &= tr\left(\frac{\partial^2 \mathbf{A}_1}{\partial q_1\partial q_1}\mathbf{D}_1 + \frac{\partial \mathbf{A}_1}{\partial q_1}\frac{\partial \mathbf{D}_1}{\partial q_1}\right) \\ &\quad - \mathbf{g}^T\frac{\partial^2 \mathbf{A}_1}{\partial q_1\partial q_1}\mathbf{E}_1 - \mathbf{f}_2^T\frac{\partial^2 \mathbf{A}_1}{\partial q_1\partial q_1}\mathbf{F}_1 \\ &= -m_2gl_2\sin(\theta_1 + \theta_2) - m_1gl_1\sin\theta_1 - m_2gL_1\sin\theta_1 \\ &\quad - fL_2\sin(\theta_1 + \theta_2) - fL_1\sin\theta_1 \end{aligned} \tag{54}$$

$$\begin{aligned}\frac{\partial \tau_1}{\partial \theta_2} &= tr\left(\frac{\partial \mathbf{A}_1}{\partial q_1}\frac{\partial \mathbf{D}_1}{\partial q_2}\right) - \mathbf{g}^T\frac{\partial \mathbf{A}_1}{\partial q_1}\frac{\partial \mathbf{E}_1}{\partial q_2} - \mathbf{f}_2^T\frac{\partial \mathbf{A}_1}{\partial q_1}\frac{\partial \mathbf{F}_1}{\partial q_2} \\ &= (-2m_2L_1l_2\sin\theta_2)\ddot{\theta}_1 + (-m_2L_1l_2\sin\theta_2)\ddot{\theta}_2 \\ &\quad - 2m_2L_1l_2\dot{\theta}_1\dot{\theta}_2\cos\theta_2 - m_2L_1l_2\dot{\theta}_2^2\cos\theta_2 \\ &\quad - m_2gl_2\sin(\theta_1 + \theta_2) - fL_2\sin(\theta_1 + \theta_2) \end{aligned} \tag{55}$$

$$\frac{\partial \tau_1}{\partial \dot{\theta}_1} = tr\left(\frac{\partial \mathbf{A}_1}{\partial q_1}\frac{\partial \mathbf{D}_1}{\partial \dot{\theta}_1}\right) = -2m_2L_1l_2\dot{\theta}_2\sin\theta_2 \tag{56}$$

$$\begin{aligned}\frac{\partial \tau_1}{\partial \dot{\theta}_2} &= tr\left(\frac{\partial \mathbf{A}_1}{\partial q_1}\frac{\partial \mathbf{D}_1}{\partial \dot{\theta}_2}\right) = -2m_2L_1l_2\dot{\theta}_1\sin\theta_2 \\ &\quad - 2m_2L_1l_2\dot{\theta}_2\sin\theta_2 \end{aligned} \tag{57}$$

$$\begin{aligned}\frac{\partial \tau_1}{\partial \ddot{\theta}_1} &= tr\left(\frac{\partial \mathbf{A}_1}{\partial q_1}\frac{\partial \mathbf{D}_1}{\partial \ddot{\theta}_1}\right) = I_1 + I_2 + m_1l_1^2 \\ &\quad + m_2\left(L_1^2 + l_2^2 + 2L_1l_2\cos\theta_2\right) \end{aligned} \tag{58}$$

$$\frac{\partial \tau_1}{\partial \ddot{\theta}_2} = tr\left(\frac{\partial \mathbf{A}_1}{\partial q_1}\frac{\partial \mathbf{D}_1}{\partial \ddot{\theta}_2}\right) = I_2 + m_2 l_2^2 + m_2 L_1 l_2 \cos\theta_2 \quad (59)$$

$$\begin{aligned}\frac{\partial \tau_2}{\partial \theta_1} &= tr\left(\frac{\partial^2 \mathbf{A}_2}{\partial q_2 \partial q_1}\mathbf{D}_2 + \frac{\partial \mathbf{A}_2}{\partial q_2}\frac{\partial \mathbf{D}_2}{\partial q_1}\right) \\ &\quad -\mathbf{g}^T\frac{\partial^2 \mathbf{A}_2}{\partial q_2 \partial q_1}\mathbf{E}_2 - \mathbf{f}_2^T\frac{\partial^2 \mathbf{A}_2}{\partial q_2 \partial q_1}\mathbf{F}_2 \\ &= -m_2 g l_2 \sin(\theta_1+\theta_2) - f L_2 \sin(\theta_1+\theta_2) \quad (60)\end{aligned}$$

$$\begin{aligned}\frac{\partial \tau_2}{\partial \theta_2} &= tr\left(\frac{\partial^2 \mathbf{A}_2}{\partial q_2 \partial q_2}\mathbf{D}_2 + \frac{\partial \mathbf{A}_2}{\partial q_2}\frac{\partial \mathbf{D}_2}{\partial q_2}\right) - \mathbf{g}^T\frac{\partial^2 \mathbf{A}_2}{\partial q_2 \partial q_2}\mathbf{E}_2 \\ &\quad -\mathbf{f}_2^T\frac{\partial^2 \mathbf{A}_2}{\partial q_2 \partial q_2}\mathbf{F}_2 \\ &= \left(-m_2 L_1 l_2 \sin\theta_2\right)\ddot{\theta}_1 + m_2 L_1 l_2 \dot{\theta}_1^2 \cos\theta_2 \\ &\quad -m_2 g l_2 \sin(\theta_1+\theta_2) - f L_2 \sin(\theta_1+\theta_2) \quad (61)\end{aligned}$$

$$\frac{\partial \tau_2}{\partial \dot{\theta}_1} = tr\left(\frac{\partial \mathbf{A}_2}{\partial q_2}\frac{\partial \mathbf{D}_2}{\partial \dot{\theta}_1}\right) = 2m_2 L_1 l_2 \dot{\theta}_1 \sin\theta_2 \quad (62)$$

$$\frac{\partial \tau_2}{\partial \dot{\theta}_2} = tr\left(\frac{\partial \mathbf{A}_2}{\partial q_2}\frac{\partial \mathbf{D}_2}{\partial \dot{\theta}_2}\right) = 0 \quad (63)$$

$$\frac{\partial \tau_2}{\partial \ddot{\theta}_1} = tr\left(\frac{\partial \mathbf{A}_2}{\partial q_2}\frac{\partial \mathbf{D}_2}{\partial \ddot{\theta}_1}\right) = I_2 + m_2 l_2^2 + m_2 L_1 l_2 \cos\theta_2 \quad (64)$$

$$\frac{\partial \tau_2}{\partial \ddot{\theta}_2} = tr\left(\frac{\partial \mathbf{A}_2}{\partial q_2}\frac{\partial \mathbf{D}_2}{\partial \ddot{\theta}_2}\right) = I_2 + m_2 l_2^2 \quad (65)$$

The foregoing recursive sensitivity equations are verified with Appendix 1 which contains details for the equations of motion in the closed form and their sensitivity expressions.

### 6.3 Example 1: time-optimal trajectory planning for a two-link manipulator

Time-optimal trajectory planning for the two-link manipulator is carried out by using the recursive Lagrangian algorithm. The objective is to minimize total travel time $T$ subjected to boundary conditions and torque limits. The same problem has been examined by Dissanayake et al. (1991) and Wang et al. (2005) using the closed-form equation of motion without the gravity effects.

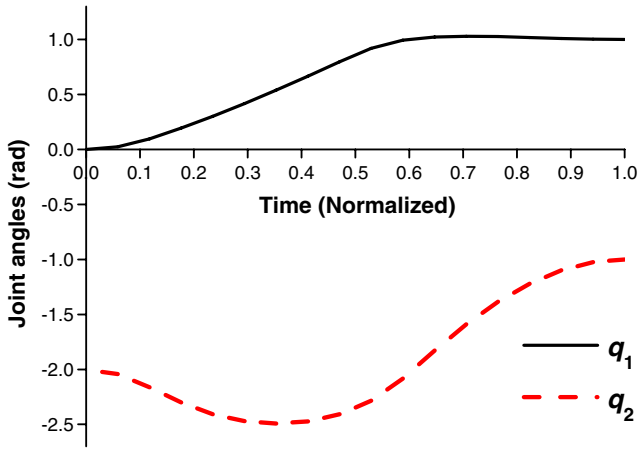


**Fig. 7** Joint angle profiles, Example 1

The parameter optimization problem can then be stated mathematically as follows: to compute design variables $\boldsymbol{x}$, which are control points $\mathbf{P}$ and total travel time $T$, and to minimize $T$ subject to the constraints on boundary conditions and torque limits as described in (66):

$$\begin{aligned}&Min.\ T(\boldsymbol{x}) \\ &St.\quad q_1(0) = 0.0,\ q_2(0) = -2.0 \\ &\qquad q_1(T) = 1.0,\ q_2(T) = -1.0 \\ &\qquad \dot{q}_1(0) = \dot{q}_2(0) = \dot{q}_1(T) = \dot{q}_2(T) = 0.0 \\ &\qquad -10 \le \boldsymbol{\tau}(\mathbf{x}) \le 10\end{aligned} \quad (66)$$

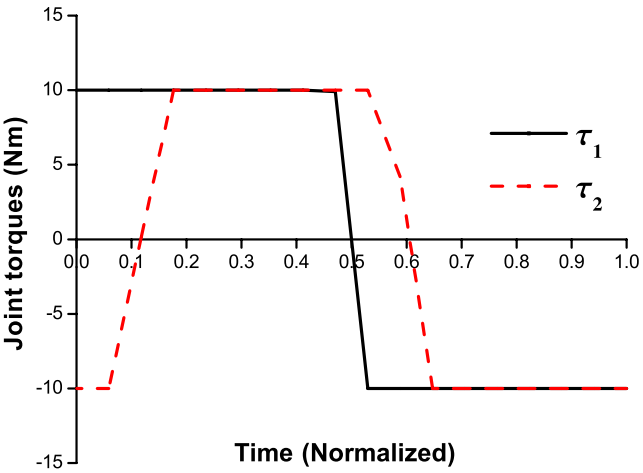


**Fig. 8** Joint torque profiles, Example 1

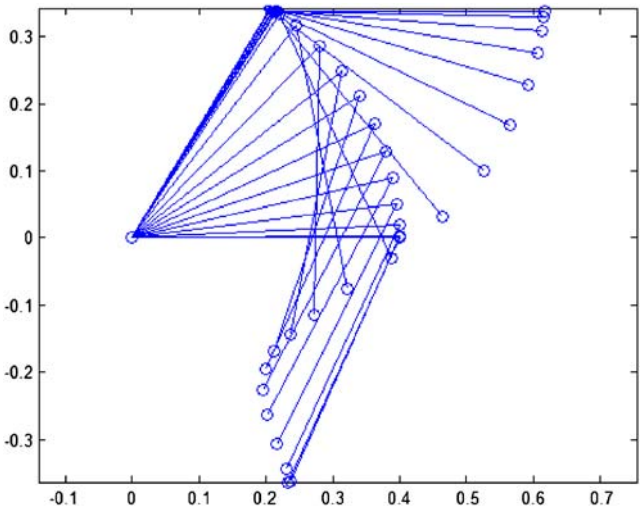


**Fig. 9** Motion trajectory, Example 1

where $T$ is the final time that needs to be minimized. Gravity effects are neglected in this time-optimal design problem, i.e., $g = 0$.

The optimal joint profiles are shown in Fig. 7, and joint torque profiles are as depicted in Fig. 8. The manipulator's trajectory is illustrated in Fig. 9.

The minimum travel time is $T = 0.3934$ s, which is similar to that reported in the literature (0.3945 s by Furukawa (2002); 0.394 s by Wang et al. (2005)). The optimality and feasibility tolerances are both set to the default value $\varepsilon = 10^{-6}$ in SNOPT. Different starting points were tried for the optimization and they all converge to the same optimal solution, implying a global solution for the problem. The convergence history of the cost function (the total travel time T) is plotted in Fig. 10. It is noted that the SQP algorithm in SNOPT worked quite well and optimal solutions were obtained in 1.07 CPU seconds on a 1.40 GHz PC.

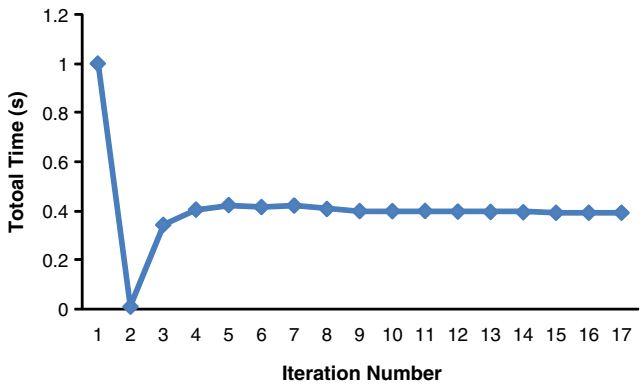


**Fig. 10** Iteration history for the cost function, Example 1

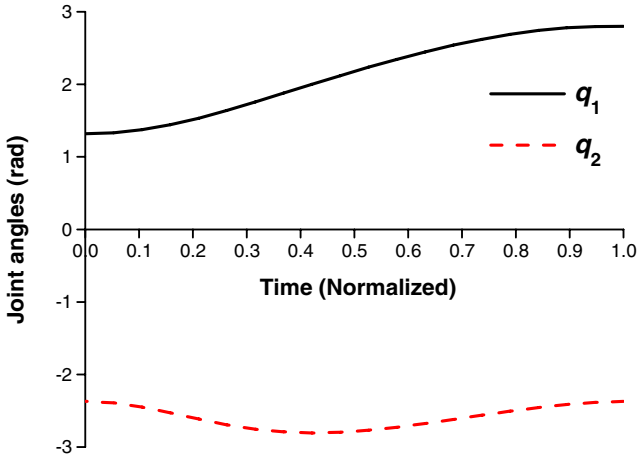


**Fig. 11** Joint angle profile with external load, Example 2

### 6.4 Example 2: mixed optimization of a two-link manipulator with an external load

The lifting motion of the two-link manipulator is studied with a mixed performance criterion, as shown in (67):

$$\int_{t=0}^{T} \left( (1-u) + u \sum_{i=1}^{n} \tau_i^2 \right) dt \quad u \in [0, 1] \tag{67}$$

where $u$ is a specified constant and $T$ is the total travel time. The second term, torque square is related to energy consumption. When $u$ goes to zero, the mixed performance criterion becomes the time-optimal criterion. An appropriate $u$ avoids harsh functioning of the actuator torques (Bessonnet and Lallemand 1990). This example problem with external loads is new that has not been solved in the literature before.

The physical parameters are listed in Table 1. The constant vertical external load $f = 10$ N acting at the

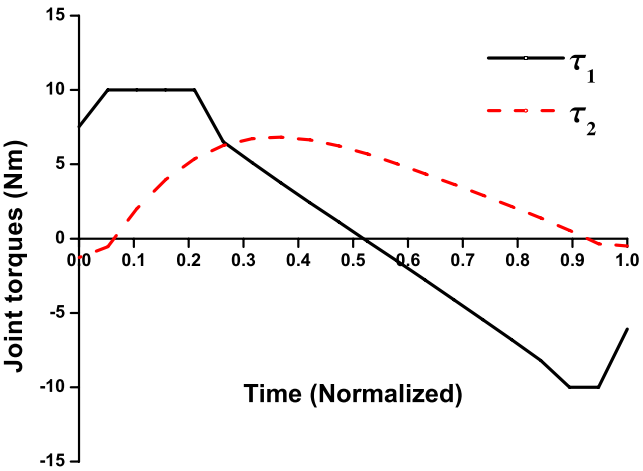


**Fig. 12** Joint torque profile with external load, Example 2

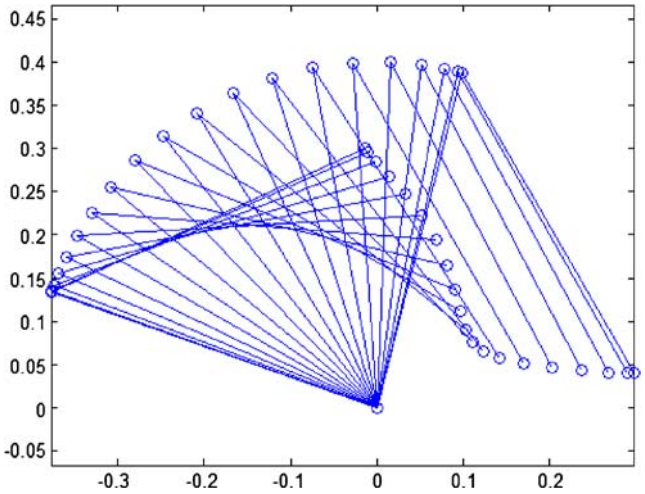


**Fig. 13** Optimal motion trajectory with external load, Example 2

end-effecter and the gravity effect $g = 9.8062$ m/s$^2$ are considered in this case. The optimization formulation is given as follows:

$$
\begin{aligned}
&Min. \int_{t=0}^{T} \left( (1-u) + u \sum_{i=1}^{n} \tau_i^2 (\boldsymbol{x}) \right) dt \qquad u \in [0,1] \\
&St. \quad q_1(0) = 1.32,\ q_2(0) = -2.37 \\
&\qquad q_1(T) = 2.80,\ q_2(T) = -2.37 \\
&\qquad \dot{q}_1(0) = \dot{q}_2(0) = \dot{q}_1(T) = \dot{q}_2(T) = 0.0 \\
&\qquad -10 \le \boldsymbol{\tau}(\boldsymbol{x}) \le 10 \qquad (68)
\end{aligned}
$$

where $\boldsymbol{x}$ is a vector of control points $\mathbf{P}$ (joint profile) and total time $T$: $\boldsymbol{x} = [\mathbf{P}, T]$. $u = 0.01$. Starting points are obtained by linear interpolation between the initial and final joint angles.

Optimal travel time for the mixed optimization problem is $T = 0.520$ s. Figure 11 shows joint profiles, and Fig. 12 shows joint torque profiles. The optimality and feasibility tolerances are both set to the default value $\varepsilon = 10^{-6}$ in SNOPT and the optimal solutions were obtained in 2.44 CPU seconds on a 1.40 GHz PC. To verify the optimal solution, the problem was also solved using commercial multibody dynamics software ADAMS$^{TM}$. The optimized joint torques as shown in Fig. 12 were treated as inputs and the equations of motion were automatically generated and integrated to obtain the response. The two motion trajectories matched quite well. Figure 13 shows the optimal trajectory for the problem.

Sensitivity results with the recursive algorithm and the closed-form formulation at the optimal design are compared in Fig. 14. The sensitivity obtained from the two algorithms match quite closely.

## 7 Concluding remarks

In this paper, the recursive Lagrangian equation and sensitivity algorithm were presented, and the recursive algorithm was validated with the closed-form equations of motion. External forces and moments were systematically included in the recursive formulation.

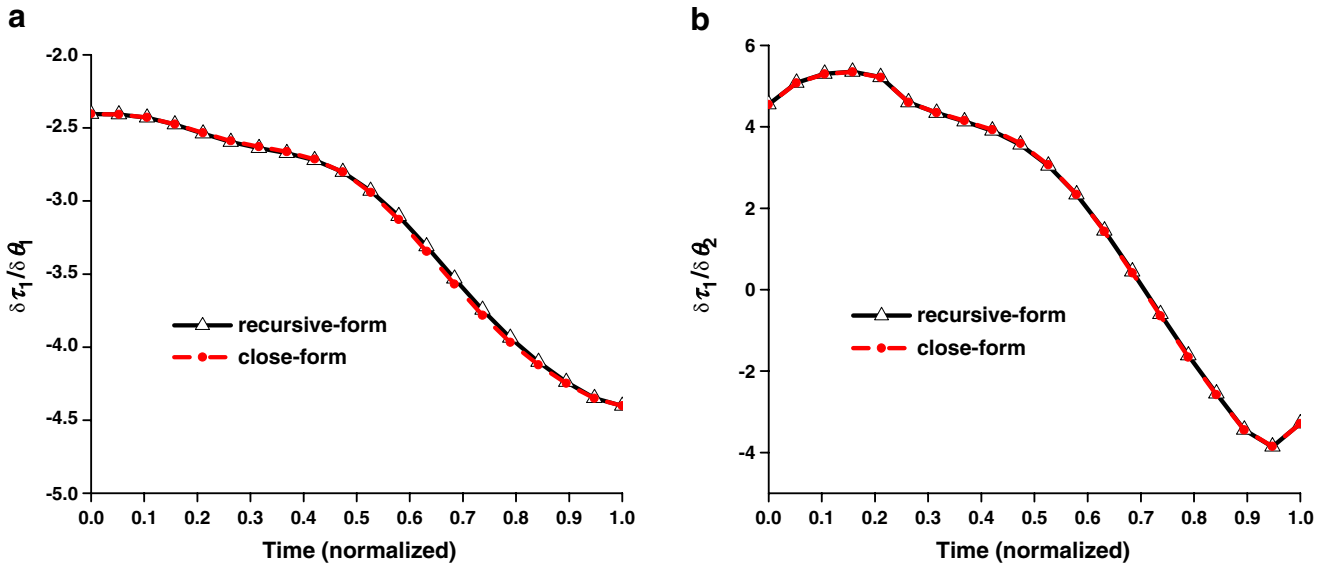


**Fig. 14** Sensitivity comparison at the optimal design

Although recursive Newton–Euler also gives linear computational complexity with respect to the number of DOFs, recursive Lagrangian formulation is more straightforward to calculate joint torques since the internal force calculation is avoided. The time-optimal trajectory planning and the lifting motion problems of a two-link manipulator were solved for motion prediction. The motion planning was formulated as a parameter optimization problem by using inverse dynamics. The control points of the $B$-spline representation for the joint angle profiles and total travel time were selected as design variables, and the equations of motion were considered in the optimization formulation. Optimal solutions were successfully obtained with the proposed sensitivity scheme. The simulation results suggest that more complicated motions such as walking and running, or those associated with a large-DOF digital human models, can be solved with the proposed sensitivity and optimization scheme. The results of these complex applications are presented in separate papers (Xiang et al. 2007; Chung et al. 2007) which use a 3D articulated human skeleton model with 55 DOFs.

**Acknowledgements** This research is partially supported by a project from the US Army TACOM.

## Appendix 1: Closed-form equations of motion and sensitivity formulation

The closed-form Lagrangian equation of a two-link rigid manipulator is well-studied and can be written as follows (Fig. 15):

$$\begin{aligned}\tau_1 = &\left(I_1 + I_2 + m_1 l_1^2 + m_2\left(L_1^2 + l_2^2 + 2L_1 l_2 \cos\theta_2\right)\right)\ddot{\theta}_1 \\ &+ \left(I_2 + m_2 l_2^2 + m_2 L_1 l_2 \cos\theta_2\right)\ddot{\theta}_2 \\ &-2m_2 L_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \sin\theta_2 - m_2 L_1 l_2 \dot{\theta}_2^2 \sin\theta_2 \\ &+m_2 g l_2 \cos(\theta_1 + \theta_2) + m_1 g l_1 \cos\theta_1 \\ &+m_2 g L_1 \cos\theta_1 + f L_2 \cos(\theta_1 + \theta_2) + f L_1 \cos\theta_1 \end{aligned} \tag{69}$$

$$\begin{aligned}\tau_2 = &\left(I_2 + m_2 l_2^2\right)\ddot{\theta}_2 + \left(I_2 + m_2 l_2^2 + m_2 L_1 l_2 \cos\theta_2\right)\ddot{\theta}_1 \\ &+m_2 L_1 l_2 \dot{\theta}_1^2 \sin\theta_2 + m_2 g l_2 \cos(\theta_1 + \theta_2) \\ &+ f L_2 \cos(\theta_1 + \theta_2) \end{aligned} \tag{70}$$

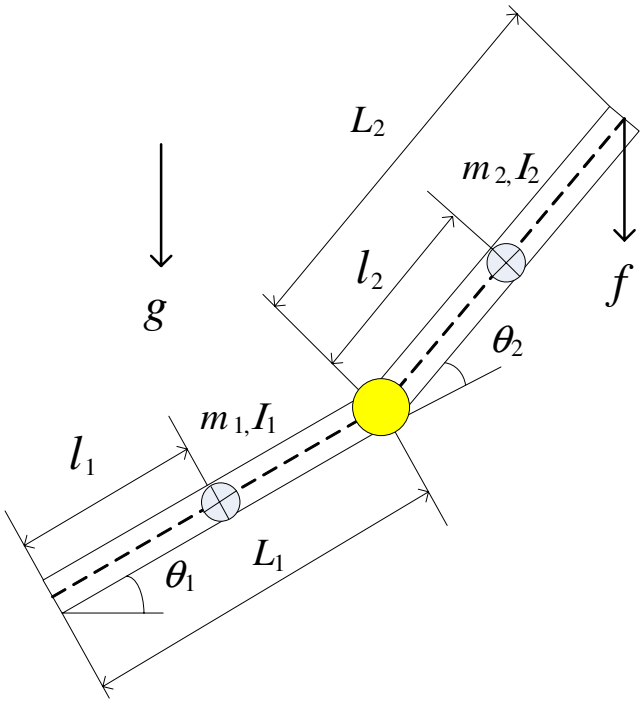


**Fig. 15** Two-link manipulator

Explicit gradients of torque with respect to state variables are derived as follows:

$$\begin{aligned}\frac{\partial \tau_1}{\partial \theta_1} = &-m_2 g l_2 \sin(\theta_1 + \theta_2) - m_1 g l_1 \sin\theta_1 - m_2 g L_1 \sin\theta_1 \\ &- f L_2 \sin(\theta_1 + \theta_2) - f L_1 \sin\theta_1 \end{aligned} \tag{71}$$

$$\begin{aligned}\frac{\partial \tau_1}{\partial \theta_2} = &\left(-2m_2 L_1 l_2 \sin\theta_2\right)\ddot{\theta}_1 + \left(-m_2 L_1 l_2 \sin\theta_2\right)\ddot{\theta}_2 \\ &-2m_2 L_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \cos\theta_2 - m_2 L_1 l_2 \dot{\theta}_2^2 \cos\theta_2 \\ &-m_2 g l_2 \sin(\theta_1 + \theta_2) - f L_2 \sin(\theta_1 + \theta_2) \end{aligned} \tag{72}$$

$$\frac{\partial \tau_1}{\partial \dot{\theta}_1} = -2m_2 L_1 l_2 \dot{\theta}_2 \sin\theta_2 \tag{73}$$

$$\frac{\partial \tau_1}{\partial \dot{\theta}_2} = -2m_2 L_1 l_2 \dot{\theta}_1 \sin\theta_2 - 2m_2 L_1 l_2 \dot{\theta}_2 \sin\theta_2 \tag{74}$$

$$\frac{\partial \tau_1}{\partial \ddot{\theta}_1} = I_1 + I_2 + m_1 l_1^2 + m_2\left(L_1^2 + l_2^2 + 2L_1 l_2 \cos\theta_2\right) \tag{75}$$

$$\frac{\partial \tau_1}{\partial \ddot{\theta}_2} = I_2 + m_2 l_2^2 + m_2 L_1 l_2 \cos\theta_2 \tag{76}$$

$$\frac{\partial \tau_2}{\partial \theta_1} = -m_2 g l_2 \sin(\theta_1 + \theta_2) - f L_2 \sin(\theta_1 + \theta_2) \tag{77}$$

$$\frac{\partial \tau_2}{\partial \theta_2} = (-m_2 L_1 l_2 \sin\theta_2)\,\ddot{\theta}_1 + m_2 L_1 l_2 \dot{\theta}_1^2 \cos\theta_2 - m_2 g l_2 \sin(\theta_1+\theta_2) - f L_2 \sin(\theta_1+\theta_2) \quad (78)$$

$$\frac{\partial \tau_2}{\partial \dot{\theta}_1} = 2 m_2 L_1 l_2 \dot{\theta}_1 \sin\theta_2 \quad (79)$$

$$\frac{\partial \tau_2}{\partial \dot{\theta}_2} = 0 \quad (80)$$

$$\frac{\partial \tau_2}{\partial \ddot{\theta}_1} = I_2 + m_2 l_2^2 + m_2 L_1 l_2 \cos\theta_2 \quad (81)$$

$$\frac{\partial \tau_2}{\partial \ddot{\theta}_2} = I_2 + m_2 l_2^2 \quad (82)$$

## References


Anderson K, Hsu Y (2002) Analytical fully-recursive sensitivity analysis for multibody dynamic chain systems. Multibody Syst Dyn 8:1–27

Anderson FC, Pandy MG (2001) Dynamic optimization of human walking. J Biomech Eng 123(5):381–390

Armstrong WM (1979) Recursive solution to the equations of motion of an N-link manipulator. In: Proceedings of the 5th world congress on theory of machines and mechanisms, vol. 2, pp 1343–1346

Bae DS, Haug EJ (1987) A recursive formulation for constrained mechanical system dynamics: part I. Open loop systems. Mechan Struct Mach 15(3):359–382

Bae DS, Cho H, Lee S, Moon W (2001) Recursive formulas for design sensitivity analysis of mechanical systems. Comput Methods Appl Mech Eng 190:3865–3879

Barthelemy JFM, Hall LE (1995) Automatic differentiation as a tool in engineering design. Struct Optim 9:76–82

Bessonnet G, Lallemand JP (1990) Optimal trajectories of robot arms minimizing constrained actuators and travelling time. IEEE Int Conf Robot Autom 1:112–117

Chung HJ, Xiang Y, Mathai A, Rahmatalla S, Kim J, Marler T, Beck S, Yang J, Arora JS, Abdel-Malek K, Obuseck J (2007) A robust formulation for prediction of human running. 2007 Digital human modeling for design and engineering symposium, Seattle, Washington, 16–18 June

Denavit J, Hartenberg RS (1955) A kinematic notation for lower-pair mechanisms based on matrices. J Appl Mech 77: 215–221

Dissanayake MWMG, Goh CJ, Phan-Thien N (1991) Time-optimal trajectories for robot manipulators. Robotica 9: 131–138

Eberhard P, Bischof C (1999) Automatic differentiation of numerical integration algorithms. Math Comput 68:717–731

Eberhard P, Schiehlen W (2006) Computational dynamics of multibody systems: history, formalisms, and applications. J Comput Nonlinear Dyn 1:1–12

Featherstone R (1987) Robot dynamics algorithms. Kluwer, Boston

Fu KS, Gonzalez RC, Lee CSG (1987) Robotics: control, sensing, vision, and intelligence. McGraw-Hill, New York

Furukawa T (2002) Time-subminimal trajectory planning for discrete non-linear systems. Eng Optim 34:219–243

Gill PE, Murray W, Saunders MA (2002) SNOPT: an SQP algorithm for large-scale constrained optimization. SIAM J Optim 12:979–1006

Hollerbach JM (1980) A recursive Lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity. IEEE Trans Syst Man Cybern 11(10):730–736

Hsieh CC, Arora JS (1984) Design sensitivity analysis and optimization of dynamic response. Comput Methods Appl Mech Eng 43(2):195–219

Kim JG, Baek JH, Park FC (1999) Newton-type algorithms for robot motion optimization. Proceedings of IEEE/RSJ international conference on intelligent robots and systems 3:1842–1847

Lo J, Huang G, Metaxas D (2002) Human motion planning based on recursive dynamics and optimal control techniques. Multibody Syst Dyn 8(4):433–458

Luh J, Walker M, Paul R (1980) On-line computational scheme for mechanical manipulators. J Dyn Syst Meas Control 102(2):69–76

Orin D, McGhee R, Vukobratovic M, Hartoch G (1979) Kinematic and kinetic analysis of open-chain linkages utilizing Newton–Euler methods. Math Biosci 43(1–2):107–130

Park FC, Bobrow JE, Ploen SR (1995) A lie group formulation of robot dynamics. Int J Robot Res 14(6):609–618

Rein U (1995) Efficient object-oriented programming of multibody dynamics formalisms. Computational dynamics in multibody systems. Kluwer, Dordrecht, The Netherlands, pp 37–47

Rodríguez JI, Jiménez JM, Funes FJ, Jalón JGD (2004) Recursive and residual algorithms for the efficient numerical integration of multi-body systems. Multibody Syst Dyn 11: 295–320

Schiehlen W (1997) Multibody system dynamics: roots and perspectives. Multibody Syst Dyn 1:149–188

Serban R, Haug EJ (1998) Kinematic and kinetic derivatives in multibody system analysis. Mech Struct Mach 26(2):145–173

Snyman JA, Berner DF (1999) A mathematical optimization methodology for the optimal design of a planar robotic manipulator. Int J Numer Methods Eng 44:535–550

Sohl GA, Bobrow JE (2001) A recursive multibody dynamics and sensitivity algorithm for branched kinematic chains. J Dyn Syst Meas Control 123:391–399

Toogood RW (1989) Efficient robot inverse and direct dynamics algorithms using micro-computer based symbolic generation. IEEE Int Conf Robot Autom 3:1827–1832

Uicker JJ (1965) On the dynamic analysis of spatial linkages using $4 \times 4$ matrices. Ph.D. thesis, Northwestern University, Evanston, IL, USA

Wang Q, Xiang Y-J, Kim H-J, Arora JS, Abdel-Malek K (2005) Alternative formulations for optimization-based digital human motion prediction. Paper 2005-01-2691, 2005 digital human modeling for design and engineering symposium, Iowa City, IA, 14–16 June

Wang Q, Xiang Y-J, Arora JS, Abdel-Malek K (2007) Alternative formulations for optimization-based human gait planning. 48th AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics and materials conference, Honolulu, Hawaii, 23–26 April

Xiang Y, Chung HJ, Mathai A, Rahmatalla S, Kim J, Marler T, Beck S, Yang J, Arora JS, Abdel-Malek K, Obuseck J (2007) Optimization-based dynamic human walking prediction. 2007 digital human modeling for design and engineering symposium, Seattle, Washington, 11–14 June