Journal of Southeast University (English Edition) Vol. 25, No. 3, pp. 340 – 345 Sept. 2009 ISSN 1003—7985

General dynamic model for educational assembling-type robot and its fast simulation algorithm

Gao Haitao Zhang Zhisheng Cao Jie Shi Jinfei

(School of Mechanical Engineering, Southeast University, Nanjing 211189, China)

Abstract: To realize automatic modeling and dynamic simulation of the educational assembling-type robot with open structure, a general dynamic model for the educational assembling-type robot and a fast simulation algorithm are put forward. First, the educational robot system is abstracted to a multibody system and a general dynamic model of the educational robot is constructed by the Newton-Euler method. Then the dynamic model is simplified by a combination of components with fixed connections according to the structural characteristics of the educational robot. Secondly, in order to obtain a high efficiency simulation algorithm, based on the sparse matrix technique, the augmentation algorithm and the direct projective constraint stabilization algorithm are improved. Finally, a numerical example is given. The results show that the model and the fast algorithm are valid and effective. This study lays a dynamic foundation for realizing the simulation platform of the educational robot.

Key words: educational robot; dynamic model; sparse matrix; augmentation algorithm; constraint stabilization

The educational assembling-type robot is a special type of robot used in the field of education for training students' creativity. It has numerous basic mechanical and control components with simple connecting modes. By freely assembling these components, students are able to set up robots with different structures and functions to accomplish some tasks. As a result, their creativity will be trained and enhanced.

In recent years, the educational robot has attracted increasing attention, and it has been explored and studied by many companies and researchers[1-4]. However, these existing products have many disadvantages such as high price, poor related software, low degree of standardization, and difficult maintenance. In order to overcome the above shortcomings and enrich product software, a simulation platform of the educational robot with visual design, simulation and animation is innovatively proposed based on robot simulation technology and modern manufacturing technology. This platform not only avoids the disadvantages of the existing educational robots, but also provides a new environment for creative design. In this paper, a general dynamic model of the educational robot and its fast simulation algorithm are studied in order to realize the above simulation platform.

Received 2008-12-18.

Biographies: Gao Haitao(1979—), male, graduate; Zhang Zhisheng (corresponding author), male, doctor, professor, oldbc@seu.edu.cn.

Foundation items: Hexa-Type Elites Peak Program of Jiangsu Province (No. 2008144), Qing Lan Project of Jiangsu Province, Fund for Excellent Young Teachers of Southeast University.

Citation: Gao Haitao, Zhang Zhisheng, Cao Jie, et al. General dynamic model for educational assembling-type robot and its fast simulation algorithm[J]. Journal of Southeast University (English Edition), 2009, 25(3): 340 – 345.

1 Establishment of the General Model

The educational assembling-type robot can be freely assembled according to students' imagination, so it has an unfixable topological structure. In order to search for a general modeling method, the robot with N components is abstracted to a multibody system whose topological structure is shown in Fig. 1. The reference coordinates of the robot system are first established and the centroid of the body is used as the center of body reference coordinates.

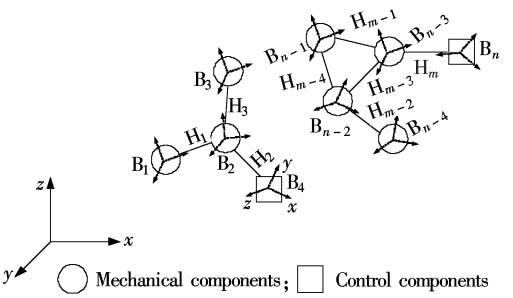

Fig. 1 Topological structure of the educational robot

The generalized coordinates of the body are Cartesian coordinates for position and Euler parameters for orientation, i.e.,

$$\boldsymbol{q}_i = [\boldsymbol{r}_i \quad \boldsymbol{p}_i] \qquad i = 1, 2, \ldots, N$$

where $\boldsymbol{r}_i = \{x_i, y_i, z_i\}$ and $\boldsymbol{p}_i = \{e_{i0}, e_{i1}, e_{i2}, e_{i3}\}$; x_i, y_i, z_i and e_{i0}, e_{i1}, e_{i2}, e_{i3} are the Cartesian coordinates and Euler parameters of each body, respectively. For each body, Newton-Euler dynamic equations with a matrix form are established[5] as

$$\begin{bmatrix} \boldsymbol{m}_i & \boldsymbol{0} \\ \boldsymbol{0} & 4\boldsymbol{G}_i\boldsymbol{J}_i\boldsymbol{G}_i \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{r}}_i \\ \ddot{\boldsymbol{p}}_i \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}_i^{\mathrm{a}} \\ \boldsymbol{L}_i^{\mathrm{a}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{F}_i^{\mathrm{c}} \\ \boldsymbol{L}_i^{\mathrm{c}} \end{bmatrix} - \begin{bmatrix} \boldsymbol{0} \\ 8\dot{\boldsymbol{G}}_i^{\mathrm{T}}\boldsymbol{J}_i\boldsymbol{G}_i\dot{\boldsymbol{p}}_i \end{bmatrix} \qquad i = 1, 2, \ldots, N \tag{1}$$

where $\boldsymbol{m}_i$ is the constant diagonal mass matrix; $\boldsymbol{J}_i$ is the constant diagonal inertia matrix; $\boldsymbol{F}_i^{\mathrm{a}}$ and $\boldsymbol{L}_i^{\mathrm{a}}$ are the vectors of applied force and applied torque, respectively; $\boldsymbol{F}_i^{\mathrm{c}}$ and $\boldsymbol{L}_i^{\mathrm{c}}$ are the vectors of constraint force and constraint torque, respectively.

$$\boldsymbol{G}_i = \begin{bmatrix} -e_1 & e_0 & e_3 & -e_2 \\ -e_2 & -e_3 & e_0 & e_1 \\ -e_3 & e_2 & -e_1 & e_0 \end{bmatrix}_i$$

Let N_j denote the number of joints in the educational robot model and d_k represent the number of constraint equations induced by joint k. Without loss of generality, these

constraints are considered to be holonomic. Let $d = \sum_{k=1}^{N_j} d_k$, and then kinematic constraints can be written as

$$\boldsymbol{\phi}(\boldsymbol{r}, \boldsymbol{p}, t) = [\boldsymbol{\phi}_1 \quad \boldsymbol{\phi}_2 \quad \cdots \quad \boldsymbol{\phi}_d]^{\mathrm{T}} = \boldsymbol{0} \tag{2}$$

Introducing d Lagrange multipliers λ, according to the ideal constraint conditions[2], we can obtain

$$\boldsymbol{F}^{\mathrm{c}} = -\boldsymbol{\phi}_r^{\mathrm{T}}\lambda, \quad \boldsymbol{L}^{\mathrm{c}} = -\boldsymbol{\phi}_p^{\mathrm{T}}\lambda$$

where $\boldsymbol{\phi}_r, \boldsymbol{\phi}_p$ are corresponding Jacobian matrices. Then the dynamic equation of the whole system with the Lagrange multiplier is obtained as

$$\begin{bmatrix} \boldsymbol{m} & \boldsymbol{0} \\ \boldsymbol{0} & 4\boldsymbol{G}^{\mathrm{T}}\boldsymbol{J}\boldsymbol{G} \end{bmatrix}\begin{bmatrix} \ddot{\boldsymbol{r}} \\ \ddot{\boldsymbol{p}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\phi}_r^{\mathrm{T}} \\ \boldsymbol{\phi}_p^{\mathrm{T}} \end{bmatrix}\boldsymbol{\lambda} = \begin{bmatrix} \boldsymbol{F}^{\mathrm{a}} \\ \boldsymbol{L}^{\mathrm{a}} \end{bmatrix} - \begin{bmatrix} \boldsymbol{0} \\ 8\dot{\boldsymbol{G}}^{\mathrm{T}}\boldsymbol{J}\boldsymbol{G}\dot{\boldsymbol{p}} \end{bmatrix} \tag{3}$$

Eqs. (2) and (3) are used as general models of the educational robot. The simplified form of Eq. (3) is written as

$$\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{\phi}_q^{\mathrm{T}}\boldsymbol{\lambda} = \boldsymbol{Q} \tag{4}$$

2 Simplification of General Model

The established dynamics model should be as simple as possible for simulation speed. In the educational robot, there are some components with fixed connections which have no relative motions, so they can be considered as one body. In this way, the dynamic model can be simplified by reducing the number of components. In regard to N_{f} components connected by fixed connections, the total mass m_{c} after combination is the accumulation of the mass of each body, i. e. ,

$$m_{\mathrm{c}} = \sum_{i=1}^{N_{\mathrm{f}}} m_i$$

The centroid coordinates are computed by

$$x_{\mathrm{c}} = \frac{\sum_{i=1}^{N_{\mathrm{f}}} m_i x_i}{m_{\mathrm{c}}}, \quad y_{\mathrm{c}} = \frac{\sum_{i=1}^{N_{\mathrm{f}}} m_i y_i}{m_{\mathrm{c}}}, \quad z_{\mathrm{c}} = \frac{\sum_{i=1}^{N_{\mathrm{f}}} m_i z_i}{m_{\mathrm{c}}}$$

Since the inertia matrix of each body is referenced to its own reference coordinates, the total inertia matrix is not the simple sum of each inertia matrix. The inertia matrix should be transformed to a common reference coordinate and then be computed. For this reason, we take the centroid of the combination body as an origin to build a reference coordinate parallel to the main reference coordinate. ρ_{ci} is defined as a position relative to the centroid of the combination body and $\boldsymbol{A}_{ci}$ as the relative direction cosine matrix($i \in N_{\mathrm{f}}$) . Inertia matrix $\boldsymbol{J}_{ci}$ referenced to the reference coordinate of the combination body can be obtained by the following steps:

Step 1 $\boldsymbol{J}_i' = \boldsymbol{A}_{ci}\boldsymbol{J}_i\boldsymbol{A}_{ci}$;

Step 2 $\boldsymbol{J}_{ci} = \boldsymbol{J}_i' + (\boldsymbol{\rho}_{ci}^{\mathrm{T}}\boldsymbol{\rho}_{ci}\boldsymbol{E} - \boldsymbol{\rho}_{ci}\boldsymbol{\rho}_{ci}^{\mathrm{T}})\boldsymbol{m}_i$.

These steps are repeated until the inertia matrix of every body is transformed into a common reference coordinate. Then the total inertia matrix $\boldsymbol{J}_{\mathrm{c}}$ of the combination is obtained by accumulation as follows:

$$\boldsymbol{J}_{\mathrm{c}} = \sum_{i=1}^{N_{\mathrm{f}}} \boldsymbol{J}_{ci}$$

Usually, the obtained inertia matrix is not the main inertia matrix, so the principal axis of inertia and the main inertia matrix are recalculated by solving the eigenvalue equation of the inertia matrix[2].

3 Fast Simulation Algorithm for Educational Robot

3.1 Traditional augmentation algorithm

To solve the above differential algebraic equations (2) and (4), various numerical algorithms have been developed in recent years[6]. Among them, the augmentation algorithm is so simple and easy to program that high usage frequency is obtained in computer simulation. In the traditional augmentation algorithm, the Lagrange multiplier is usually solved by the following equation:

$$\boldsymbol{\phi}_q\boldsymbol{M}^{-1}\boldsymbol{\phi}_q^{\mathrm{T}}\lambda = (\boldsymbol{\phi}_q\boldsymbol{M}^{-1}\boldsymbol{Q} - (\boldsymbol{\phi}_q\dot{\boldsymbol{q}})_q\dot{\boldsymbol{q}} - 2\boldsymbol{\phi}_{qt}\dot{\boldsymbol{q}}) \tag{5}$$

After obtaining the Lagrange multiplier, dynamic equations are transformed into the easily solved ordinary differential equations:

$$\begin{bmatrix} \dot{\boldsymbol{q}} \\ \dot{\boldsymbol{\nu}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\nu} \\ \boldsymbol{M}^{-1}(\boldsymbol{Q} - \boldsymbol{\phi}_q^{\mathrm{T}}\boldsymbol{\lambda}) \end{bmatrix} \tag{6}$$

Eqs. (5) and (6) are used for obtaining the state of multibody in dynamic simulation.

3.2 Sparse resolution of augmentation algorithm

Although the traditional augmentation algorithm is widely applied in dynamic simulations, it has the disadvantages of low computational efficiency and large constraint violation. So it is essential to improve the traditional algorithm to satisfy the demands of computational speed and accuracy for educational robot simulation. Then, solving equations of the Lagrange multiplier in the augmentation algorithm can be modified into the following equation[7]:

$$\begin{bmatrix} \boldsymbol{M} & \boldsymbol{\phi}_q^{\mathrm{T}} \\ \boldsymbol{\phi}_q & \boldsymbol{0} \end{bmatrix}\begin{bmatrix} \ddot{\boldsymbol{q}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{\tau} \end{bmatrix} \tag{7}$$

where $\boldsymbol{\tau} = -(\boldsymbol{\phi}_q\dot{\boldsymbol{q}})_q\dot{\boldsymbol{q}} - 2\boldsymbol{\phi}_{qi}\dot{\boldsymbol{q}}$. In contrast with Eq. (5), the improved equation, whose coefficient matrix has advantages of high degree of sparse and low effects of topology structure, can be solved by the sparse matrix technique.

In order to solve Eq. (7) by the sparse matrix technique, first, the coefficient matrix is reordered for computational efficiency. It is known that a sparse matrix has one-to-one correspondence with the undirected graph, so the ordering algorithm of the undirected graph can be used to reorder the corresponding sparse matrix. A modified reverse Cuthill-McKee algorithm(RCM) [8] is applied in this paper. As an example, the coefficient matrix of Eq. (7) has the following form:

$$
\begin{bmatrix}
\boldsymbol{m}_1 & & & & & & & & & \boldsymbol{\phi}_{11}^{r\mathrm{T}} & & \boldsymbol{\phi}_{31}^{r\mathrm{T}} & & & & & \\
& \boldsymbol{m}_2 & & & & & & & & \boldsymbol{\phi}_{12}^{r\mathrm{T}} & \boldsymbol{\phi}_{22}^{r\mathrm{T}} & & & & & \boldsymbol{0} & \\
& & \boldsymbol{m}_3 & & & & & & & & \boldsymbol{\phi}_{23}^{r\mathrm{T}} & & & & & & \\
& & & \boldsymbol{m}_4 & & & & & & & & \boldsymbol{\phi}_{34}^{r\mathrm{T}} & \boldsymbol{\phi}_{44}^{r\mathrm{T}} & & & & \\
& & & & \boldsymbol{m}_5 & & & & & & & & \boldsymbol{\phi}_{45}^{r\mathrm{T}} & & & & \\
& & & & & \boldsymbol{J}_1 & & & & & & & & \boldsymbol{\phi}_{11}^{p\mathrm{T}} & & \boldsymbol{\phi}_{31}^{p\mathrm{T}} & \\
& & & & & & \boldsymbol{J}_2 & & & \boldsymbol{0} & & & & \boldsymbol{\phi}_{12}^{p\mathrm{T}} & \boldsymbol{\phi}_{22}^{p\mathrm{T}} & & \\
& & & & & & & \boldsymbol{J}_3 & & & & & & & \boldsymbol{\phi}_{23}^{p\mathrm{T}} & & \\
& & & & & & & & \boldsymbol{J}_4 & & & & & & \boldsymbol{\phi}_{34}^{p\mathrm{T}} & \boldsymbol{\phi}_{44}^{p\mathrm{T}} & \\
& & & & & & & & & \boldsymbol{J}_5 & & & & & & \boldsymbol{\phi}_{45}^{p\mathrm{T}} \\
\boldsymbol{\phi}_{11}^{r} & \boldsymbol{\phi}_{12}^{r} & & & & & & & & \boldsymbol{0} & & & & & & \\
& \boldsymbol{\phi}_{22}^{r} & \boldsymbol{\phi}_{23}^{r} & & & & & \boldsymbol{0} & & & \boldsymbol{0} & & & & & & \\
\boldsymbol{\phi}_{31}^{r} & & & \boldsymbol{\phi}_{34}^{r} & & & & & & & & \boldsymbol{0} & & & & & \\
& & & \boldsymbol{\phi}_{44}^{r} & \boldsymbol{\phi}_{45}^{r} & & & & & & & & \boldsymbol{0} & & & & \\
& & & & & \boldsymbol{\phi}_{11}^{p} & \boldsymbol{\phi}_{12}^{p} & & & & & & & \boldsymbol{0} & & & \\
& & & & & & \boldsymbol{\phi}_{22}^{p} & \boldsymbol{\phi}_{23}^{p} & & & & & & & \boldsymbol{0} & & \\
& \boldsymbol{0} & & & & \boldsymbol{\phi}_{31}^{p} & & & \boldsymbol{\phi}_{34}^{p} & & & & & & & \boldsymbol{0} & \\
& & & & & & & & \boldsymbol{\phi}_{44}^{p} & \boldsymbol{\phi}_{45}^{p} & & & & & & & \boldsymbol{0}
\end{bmatrix}
$$

After reordering by the modified RCM algorithm, we obtain the following matrix:

$$
\begin{bmatrix}
\boldsymbol{m}_5 & \boldsymbol{\phi}_{45}^{r\mathrm{T}} & & & & & & & & & & & & & & & \\
\boldsymbol{\phi}_{45}^{r} & \boldsymbol{0} & \boldsymbol{\phi}_{44}^{r} & & & & & & & & & & & & & & \\
& \boldsymbol{\phi}_{44}^{r\mathrm{T}} & \boldsymbol{m}_4 & \boldsymbol{\phi}_{34}^{r\mathrm{T}} & & & & & & & & & & & & & \\
& & \boldsymbol{\phi}_{34}^{r} & \boldsymbol{0} & \boldsymbol{\phi}_{31}^{r} & & & & & & & & & & & & \\
& & & \boldsymbol{\phi}_{31}^{r\mathrm{T}} & \boldsymbol{m}_1 & \boldsymbol{\phi}_{11}^{r\mathrm{T}} & & & & & & & & & & & \\
& & & & \boldsymbol{\phi}_{11}^{r} & \boldsymbol{0} & \boldsymbol{\phi}_{12}^{r} & & & & & \boldsymbol{0} & & & & & \\
& & & & & \boldsymbol{\phi}_{12}^{r\mathrm{T}} & \boldsymbol{m}_2 & \boldsymbol{\phi}_{22}^{r\mathrm{T}} & & & & & & & & & \\
& & & & & & \boldsymbol{\phi}_{22}^{r} & \boldsymbol{0} & \boldsymbol{\phi}_{23}^{r} & & & & & & & & \\
& & & & & & & \boldsymbol{\phi}_{23}^{r\mathrm{T}} & \boldsymbol{m}_3 & & & & & & & & \\
& & & & & & & & & \boldsymbol{J}_5 & \boldsymbol{\phi}_{45}^{p\mathrm{T}} & & & & & & \\
& & & & & & & & & \boldsymbol{\phi}_{45}^{p} & \boldsymbol{0} & \boldsymbol{\phi}_{44}^{p} & & & & & \\
& & & & & & & & & & \boldsymbol{\phi}_{44}^{p\mathrm{T}} & \boldsymbol{J}_4 & \boldsymbol{\phi}_{34}^{p\mathrm{T}} & & & & \\
& & & & & & & & & & & \boldsymbol{\phi}_{34}^{p} & \boldsymbol{0} & \boldsymbol{\phi}_{31}^{p} & & & \\
& & & & & \boldsymbol{0} & & & & & & & \boldsymbol{\phi}_{31}^{p\mathrm{T}} & \boldsymbol{J}_1 & \boldsymbol{\phi}_{11}^{p\mathrm{T}} & & \\
& & & & & & & & & & & & \boldsymbol{\phi}_{11}^{p} & \boldsymbol{0} & \boldsymbol{\phi}_{12}^{p} & & \\
& & & & & & & & & & & & & \boldsymbol{\phi}_{12}^{p\mathrm{T}} & \boldsymbol{J}_2 & \boldsymbol{\phi}_{22}^{p\mathrm{T}} & \\
& & & & & & & & & & & & & & \boldsymbol{\phi}_{22}^{p} & \boldsymbol{0} & \boldsymbol{\phi}_{23}^{p} \\
& & & & & & & & & & & & & & & \boldsymbol{\phi}_{23}^{p\mathrm{T}} & \boldsymbol{J}_3
\end{bmatrix}
$$

Secondly, $\boldsymbol{LDL}^{\mathrm{T}}$ symbol decomposition is applied to the matrix after reordering. The symbol decomposition is considered as a virtual decomposition, because it only records the positions of filling elements and elimination elements. By this decomposition, the position of elements is confirmed and the optimal data structure for $\boldsymbol{LDL}^{\mathrm{T}}$ decomposition is built.

Finally, the numerical resolution of Eq. (7) is carried out, which includes numerical $\boldsymbol{LDL}^{\mathrm{T}}$ decomposition of the coefficient matrix and the back-substitution solution. Numerical decomposition is a numerical computation of specific elements according to the data structure after symbol decomposition, and then back substitution is applied to yield the resolution of the equation.

The whole process of solving Eq. (7) can be summarized as follows:

Step 1 Establish a corresponding undirected graph of the coefficient matrix, $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$.

Step 2 Order graph $\boldsymbol{G}$ by the improved RCM algorithm and confirm the optimal matrix structure.

Step 3 Apply symbol $\boldsymbol{LDL}^{\mathrm{T}}$ decomposition to the matrix after reordering. According to the obtained matrix structure, take out the major operated rows/major columns and the eliminated rows/columns, and then use the correspond-

ing elements of the major rows/major columns to eliminate elements of eliminated rows/columns. The positions of modified elements are confirmed and recorded in turn.

Step 4 Repeat step 3 until decomposition ends.

Step 5 Do numerical $\boldsymbol{LDL}^{\mathrm{T}}$ decomposition.

Step 6 Solve the equation by back substitution.

In the dynamic simulation, Eq. (7) needs to be solved for λ at each time step, which occupies a large number of computations. However, with regard to the educational robot without variable constraints, the structure of the coefficient matrix of Eq. (7) is unchangeable; consequently, steps 1 to 4 can be put outside the simulation cycle as simulation preprocessing, while steps 5 and 6 are put into the simulation cycle for computation. In this way, the process of solving λ only becomes a series of arithmetic operations in the simulation cycle and thus computational efficiency is improved.

3.3 Sparse resolution of direct constraint stabilization

Constraint violations inevitably exist in the augmentation algorithm and they must be stabilized for accuracy requirements of simulation. In general, the indirect methods, especially Baumgarte's stabilization method, are used to stabilize violations[9] although they have disadvantages of low accuracy and difficult selection of stabilization parameters. Compared with indirect methods, the direct violation correction methods have higher computer accuracy but lower computational efficiency. If the computation cost of the direct methods can be decreased, the direct methods should have better simulation effect. In this paper, direct projective stabilization equations are constructed and solved by the optimal data structure in preprocessing of simulations so as to save the computational cost.

After n integration steps, a numerical solution of position $\tilde{\boldsymbol{q}}_n$ is obtained. Usually, there is a small error making the constraint $\boldsymbol{\phi}(\tilde{\boldsymbol{q}}_n) \neq \boldsymbol{0}$. Let the exact solution be $\boldsymbol{q}_n$ and the error be $\Delta\boldsymbol{q}_n$. To satisfy the position constraints, the numerical solution can be projected to the manifold defined as $\boldsymbol{\phi}(\boldsymbol{q}) = 0$ by solving the following minimization problem[10]:

$$\min_{\boldsymbol{q}_n \in \mathbf{R}^N} \| \boldsymbol{q}_n - \tilde{\boldsymbol{q}}_n \|_M \qquad \text{with } \boldsymbol{\phi}(\boldsymbol{q}_n) = \boldsymbol{0}$$

where $\| \boldsymbol{x} \|_M = \sqrt{\boldsymbol{x}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{x}}$ and $\boldsymbol{M}$ is a positive definite generalized mass matrix.

To solve the minimization problem, the constraints are coupled to the target function by the Lagrange parameter $\boldsymbol{\mu}$. It gives the necessary conditions for a minimum:

$$\boldsymbol{M}(\boldsymbol{q}_n - \tilde{\boldsymbol{q}}_n) + \boldsymbol{\phi}_q^{\mathrm{T}}(\boldsymbol{q}_n)\boldsymbol{\mu} = \boldsymbol{0} \tag{8a}$$

$$\boldsymbol{\phi}(\boldsymbol{q}_n) = \boldsymbol{0} \tag{8b}$$

The above equations can be linearized to produce

$$\begin{bmatrix} \boldsymbol{M} & \boldsymbol{\phi}_q^{\mathrm{T}}(\tilde{\boldsymbol{q}}_n) \\ \boldsymbol{\phi}_q(\tilde{\boldsymbol{q}}_n) & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{q}_n \\ \boldsymbol{\mu} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ -\boldsymbol{\phi}(\tilde{\boldsymbol{q}}_n) \end{bmatrix} \tag{9}$$

Likewise, the error in the velocity constraints can be corrected in a similar way by projection of the velocity $\tilde{\dot{\boldsymbol{q}}}_n$ onto the manifold defined by $\dot{\boldsymbol{\phi}}(\boldsymbol{q}_n) = \boldsymbol{0}$:

$$\min_{\dot{\boldsymbol{q}}_n \in \mathbf{R}^N} \| \dot{\boldsymbol{q}}_n - \tilde{\dot{\boldsymbol{q}}}_n \|_M \qquad \text{with } \boldsymbol{\phi}(\boldsymbol{q}_n)\dot{\boldsymbol{q}}_n = \boldsymbol{0}$$

This minimization problem is solved by the following linear equation:

$$\begin{bmatrix} \boldsymbol{M} & \boldsymbol{\phi}_q^{\mathrm{T}}(\boldsymbol{q}_n) \\ \boldsymbol{\phi}_q(\boldsymbol{q}_n) & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \Delta\dot{\boldsymbol{q}}_n \\ \boldsymbol{\eta} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ -\dot{\boldsymbol{\phi}}(\boldsymbol{q}_n, \tilde{\dot{\boldsymbol{q}}}_n) \end{bmatrix} \tag{10}$$

where $\boldsymbol{\eta}$ is the Lagrange parameter and $\Delta\dot{\boldsymbol{q}}_n$ is the error of velocity. Normally, Eq. (9) needs to be solved by a Newton-like iterative method. In order to improve computer efficiency, we need not use an iterative method in applications. So the limited number of Newton steps can only be performed to reduce the errors in the constraints. Practical experience shows that only one step of the Newton iteration for the projection of the position is sufficient for stabilization. Ref. [11] gives a rigorous proof that this strategy guarantees that the error in the constraints remains limited for arbitrary time intervals and the upper bound is

$$\max \| \boldsymbol{\phi}(\boldsymbol{q}_m, t_m) \| \leqslant Ch^3$$

where the constant C is independent of h and t. A comparison between an iterative projection method to avoid the drift-off and a noniterative projection method to avoid the drift-off is outlined in Fig. 2.

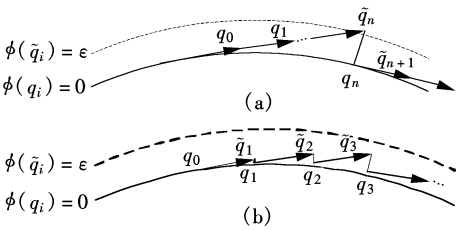

Fig. 2 Comparison of two stabilization methods. (a) Iterative projection method; (b) Noniterative projection method

Comparing Eqs. (9) and (10) with Eq. (7), we can find that they have the same coefficient matrix structure, so the sparse matrix computer technique is used for obtaining stabilization values again. Finally, the stabilized position and velocity are computed as

$$\boldsymbol{q}_n = \tilde{\boldsymbol{q}}_n + \Delta\boldsymbol{q}_n$$
$$\dot{\boldsymbol{q}}_n = \tilde{\dot{\boldsymbol{q}}}_n + \Delta\dot{\boldsymbol{q}}_n$$

3.4 Summary of simulation algorithm

The direct augmentation algorithm and projective constraint stabilization by the sparse matrix computer technique, together with the mature ODE integration method, constitute a complete simulation algorithm. The flowchart of the educational robot simulation is summarized in Fig. 3.

In the improved algorithm, solving Lagrange multipliers requires a computational complexity of about $O(2m)$ at each step instead of about $O(m^3)$ in the traditional algorithm where m is the number of constraints. Moreover, projective stabilization totally requires a computational complexity of $2 \times O(2m)$ by the sparse matrix technique. This

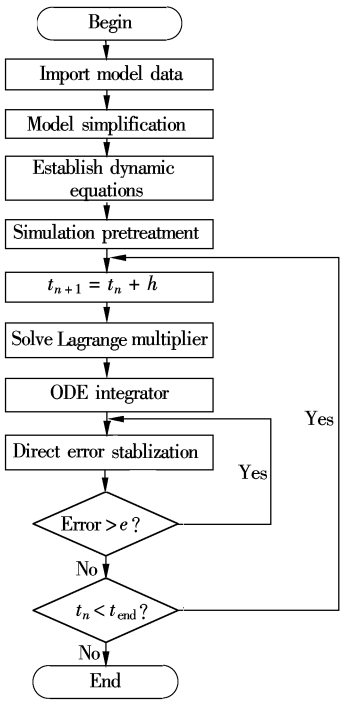

Fig. 3 Simulation flowchart

shows that the improved algorithm, whose computational complexity is linear with the number of constraints, is superior to the traditional algorithm in computational speed.

4 Examples of Dynamic Simulation

In order to verify the improved algorithm and its stabilization, as shown in Fig. 4, there is a manipulator with components. Parameters of each body are as follows: $l_1 = l_2 = l_3 = l_4 = l_7 = 0.2$ m, $l_8 = 0.08$ m, $l_9 = 0.02$ m, $l_5 = l_6 = 0.06$ m, $m_1 = m_2 = m_3 = m_4 = m_7 = 0.1$ kg, $m_5 = m_6 = 0.03$ kg, $m_9 = 0.01$ kg, $m_8 = 0.04$ kg. The inertia matrix of each body is computed according to shape, quality and body coordinates. The generalized coordinates of the system are defined as

$$\boldsymbol{q}_i = \{x_i, y_i, p_i\}^{\mathrm{T}} \qquad i = 1, 2, \ldots, 9$$

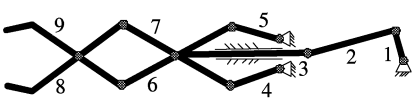

Fig. 4 Structural sketch of manipulator

Based on general modeling methods and the simplified method, the dynamic model of the manipulator is built. The traditional augmentation algorithm and the improved algorithm are programmed to simulate the manipulator in Matlab where the correction coefficients of Baumgarte's stabilization are set at $\alpha = \beta = 30$. An explicit Newton integration algorithm is used for the ODE integrator whose time step is fixed as $h = 10$ ms.

An initial condition of the simulation which satisfies position and velocity constraints is given, while applied torque in the crank is 1 N · mm. After simulation, the tip-position of the manipulator (x_9) is shown in Fig. 5; constraints violations of the position and the velocity of the two algorithms are contrasted in Figs. 6 and 7, respectively. In the computer whose CPU is 2.8 GHz and memory is 1 GB to simulate a cycle of 10^5 times, the CPU occupancy time of the two algorithms is shown in Tab. 1.

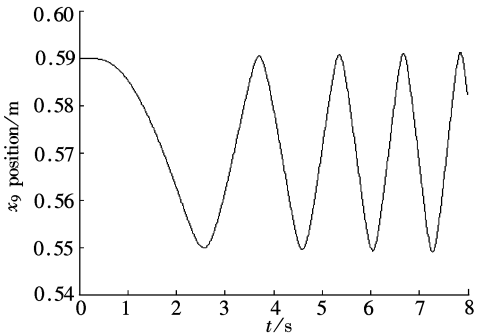

Fig. 5 Tip-position of manipulator

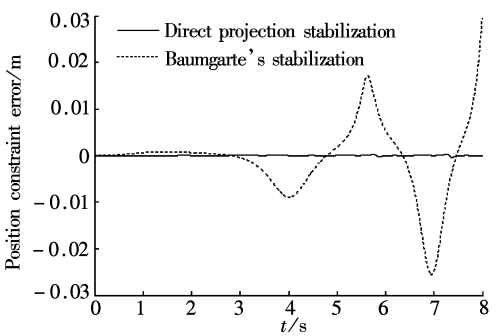

Fig. 6 Position constraint error of two algorithms

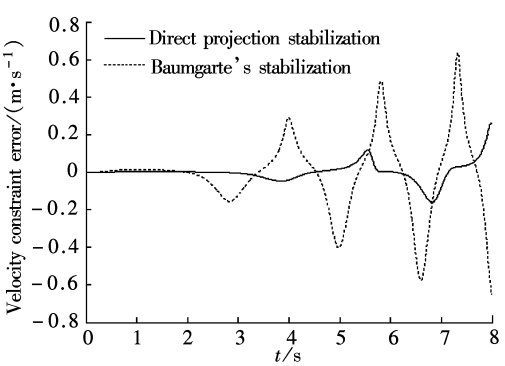

Fig. 7 Velocity constraint error of two algorithms

Tab. 1 CPU occupancy time of two algorithms

Algorithm	Step time/ms	Cycle number	Occupancy time/s
Traditional	1	10^5	16.870
Improved	1	10^5	15.014 7

As can be seen from Figs. 6 and 7, the improved algorithm is far higher in computation accuracy than the traditional algorithm. In CPU cost, the improved algorithm is also lower than the traditional algorithm as shown in Tab. 1. This shows that the improved algorithm is efficient.

5 Conclusion

In this paper, the general dynamic model of the educational assembling-type robot is established and simplified according to the structural characteristics of educational robots. Subsequently, a high efficiency simulation algorithm is given based on the sparse matrix technique. It is concluded that the improved algorithm has better accuracy and higher

speed compared with the traditional augmentation algorithm. As for future work, it will be interesting to conduct studies on the numerical algorithms of educational robots with contact and friction.

References

[1] Billard A. Robota: clever toy and educational tool [J]. *Robotics and Autonomous System*, 2003, **42**(3): 259 - 269.

[2] Michaud F, Clavet A, Lachiver G, et al. Robus—a mobile robotic platform for electrical and computer engineering education [J]. *IEEE Robotics & Automation Magazine*, 2003, **10**(3): 20 - 24.

[3] Mataric M J. Robotics education for all ages [C]//*The Proceedings of AAAI Spring Symposium on Accessible, Hands-on AI and Robotics Education*. Palo Alto, USA, 2004: 22 - 24.

[4] Fred G M. Circuits to control: learning engineering by designing LEGO robots [D]. Cambridge: Massachusetts Institute of Technology, 1994.

[5] Hong Jiazhen. *Computational dynamics of multibody systems* [M]. Beijing: Higher Education Press, 1999: 68 - 73. (in Chinese)

[6] Pan Zhenkuan, Zhao Weijia, Hong Jiazhen, et al. Numerical method on differential/algebra equation of multibody system dynamics [J]. *Advances in Mechanics*, 1996, **26**(1): 28 - 39. (in Chinese)

[7] Baraff David. Linear-time dynamics using Lagrange multipliers [C]//*Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*. New York, 1996, **30**: 137 - 146.

[8] Zhang Zhizhen, Li Shangjian, Li Zhigang. A new algorithm for reducing bandwidth of sparse matrix [J]. *Journal of Huazhong University of Science and Technology*: *Natural Science*, 1998, **26**(12): 43 - 45. (in Chinese)

[9] Baumgarte J. Stabilization of constraints and integrals of motion [J]. *Computer Methods in Applied Mechanics and Engineering*, 1972(1): 1 - 16.

[10] Hairer E, Wanner G. *Solving ordinary differential equations* Ⅱ: *stiff and differential-algebraic problems* [M]. 2nd ed. Berlin: Springer-Verlag, 1996.

[11] Burgermeister B, Arnold M, Esterl B. DAE time integration for real-time applications in multi-body dynamics [J]. *Journal of Applied Mathematics and Mechanics*, 2006, **86**(10): 759 - 771.

通用组合式教育机器人动力学模型及其快速仿真算法

高海涛　张志胜　曹　杰　史金飞

(东南大学机械工程学院,南京 211189)

摘要: 为实现具有开放结构的组合式教育机器人自动建模与动态仿真,建立了教育机器人的动力学模型及其仿真算法. 首先,将教育机器人抽象为多体系统,利用牛顿-欧拉法建立其通用的动力学模型,并结合教育机器人的结构特点,以构件绑定方法对机器人模型进行了简化. 其次,基于稀疏矩阵计算技术,对计算机仿真中常用的增广动力学求解算法和直接投影修正算法进行改进,以此作为教育机器人的仿真算法,提高仿真速度和精度. 最后,通过一个算例验证了模型和改进算法的有效性. 该研究为实现教育机器人仿真平台奠定了动力学基础.

关键词: 教育机器人;动力学模型;稀疏矩阵;增广算法;违约修正

中图分类号: TP242