REALISTIC ANIMATION USING MUSCULOTENDON SKELETAL DYNAMICS AND SUBOPTIMAL CONTROL

Jihun Park, Donald Fussell, Marcus Pandy,
and James C. Browne

Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712-1188

TR-92-26 April 1992

TR 92-26

Realistic Animation using Musculotendon Skeletal Dynamics and Suboptimal Control

Jihun Park*, Donnald Fussell* Marcus Pandy**, and James C. Browne*
Department of Computer Sciences*
Department of Kinesiology and Health Education**
The University of Texas at Austin
Austin, TX 78712

April 30, 1992

Abstract

This paper presents a new methodology for model and control of the motion of an human body for the purposes of animation. The approach is to use a parameter optimization method for forward dynamic simulation to get a best set of control, and to use musculotendon skeletal model for the human body to model as close as possible. Skeletal and musculotendon dynamics enables us to do the human body animation more accurate than ever because the muscle force depends on geometry of a human as well as differential kinematic parameters. Our control for a musculotendon skeletal dynamic model is to have a moderate number of control nodes and linearly interpolates among this set of nodes. Then we use the control values from the linear interpolation to actuate each musculotendon actuators at the body. We then find a best set of control by using a parameter optimization method. The method is formulated in detail and is demonstrated by an animation of human jumping.

1 Introduction

It is well known that producing realistic animated motion of natural systems requires modeling the dynamics which determine key aspects of that motion [8, 52, 53]. Ideally, an animation system would allow the animator to specify only the start and the goal configurations of the moving system being modeled. Direct specification of the kinematic features of the motion itself is not desirable, since it is essentially not easy to achieve realistic motion in this way. Instead, a dynamic model of the natural system is built which determines the kinematics. It is desirable that such models be specified in terms of parameters which allow effective and intuitive control by the animator of the motion.

Dynamic models of articulated bodies typically have linear or tree- structured topologies. In the absence of environmental constraints, these are open-loop mechanisms – that is they can be modeled as links connected by joints, where the links at the end of the structure have a free unconstrained end. The presence of constraints, such as contact with a surface, cannot always be modeled as an open-loop mechanism since the end links are constrained by the forces invoked by contact with the surface. In this case a closed-loop model of the system or object is needed. Application and removal of the constraints changes the numbers of degrees of freedom in the model. We use a single general dynamic model to simulate all types of motions. The model is a free-base, open- loop kinematic chain which simulates a variety of mechanisms by including the corresponding force terms.

A human body is complicated mechanically, and is very different from the robotic mechanisms. Most people use robotic mechanisms for their human body animation, we believe, because of the complexity of musculotendon actuators. But by simple muscle actuators, we can get more realistic results than simple joint torque actuation like a robot. Because musculotendon actuator is a linear actuator, the perpendicular distance from the center of a joint to the muscle changes the resulting magnitude of joint torque heavily. So if a joint is fully extended, the joint can not generate much *active* torque, even though the muscle is fully activated, compared to the torque of a joint which is half extended. The major difference is due to *moment arm*. (We will use the term *moment arm* different from usual meaning. We mean moment arm the perpendicular distance from the center of joint to the line where musculotendon actuator generates forces.) The moment arm of fully extended joint is near zero. So the geometry of human body affects the magnitude of torque that can be generated by a human joint. So the magnitude is not a fixed value.[32] Also we do not need to worry about spring/damper coefficients which is difficult to select for simple joint torque animation. Because musculotendon dynamics is a first order differential equation which includes nonlinear spring and dampers, the argument of spring/damper coefficients goes away completely. Also because the muscle activation dynamics is nonlinear, this guarantees that the resulting control is nonlinear that there is no jerky motion compared to the robotic motion. We believe that the musculotendon model promises to give a good result even if it were used as only kinematic constraints for the human body motion.

Also in terms of skeletal dynamics, there are dynamic discontinuities. For human jumping, the degrees of freedom(DOF) changes from three to four and to six. When the heel is on the ground it has three DOF. If the heel is off the ground while the toe is on the ground it has four DOF. When the body is on the air, DOF changes to six. (This is completely different from Luxo Jr.[57])

Motion control is one of the most difficult problems in dynamic modeling. It is usually impossible to construct exact models, and it is not simple to measure exact forces and torques. Therefore, a great deal of attention has recently been devoted to heuristic methods for control of joints [53, 51, 52]. Much of this research has been aimed at achieving faithful representations of motion in a computationally effective way. Some controllers have been designed[48, 34] by analyzing the natural motions of animals and using forward dynamic simulation. Another approach is to discretize continuous time dynamic systems to create parameter optimization problems. This approach is a variant of finite difference method in solving two point boundary value problem[57]. Inverse dynamics methods have been used to calculate joint torques given kinematic motion planning [42, 22, 23]. *Pure* inverse dynamic methods typically lack reality because the motion has been pre-planned. There are also papers which use optimal control techniques [41, 7, 40]. Forward dynamic simulation usually gives the best quality motions, especially when the controller used is exact. The classic example is dropping of a pinpointed chain. The controller typically used is a constant(zero) function in terms of time. This approach is more like simulation than animation because it is difficult to do motion planning due to the nature of the simulation. The controller for this simulation does not have any motion planning capability. Sometimes exact motion planning is not readily obtainable. Exact implies satisfaction of every constraint including the final time constraint specified by the animator. Let us assume, for example, that we wish to animate a *creature* with a given start configuration and goal configuration and a specified time for attaining the goal configuration. It is not easy to satisfy the final time constraint through controller based animation, although it is straightforward to satisfy geometric constraints with controller based animation. This paper introduces a new approach for development of a motion controller which is applying parameter optimization method for forward dynamics problem. We call this method suboptimal control. The basis of the method is to approximate optimal control with a moderate number(say twenty) of control nodes and to thus convert the problem into a parameter optimization problem. We use

forward dynamic simulation to calculate the performance indexes and the constraint functions. Also this paper uses a musculotendon skeletal dynamic model and muscle activation dynamics to do more accurate animation of a human body.

2 Related Work

In this section we review the major animation techniques for articulated rigid bodies. These methods fall into two categories: those based on forward dynamics [51, 34, 48] and those using inverse dynamics [42, 22, 23]. Inverse dynamics-based methods also need forward dynamics for simulation. Optimal control based techniques [41, 7, 40] and the sub-optimal control technique presented in this paper use forward dynamics. The *Spacetime constraints* method [57] is based upon inverse dynamics.

2.1 Pure Forward Dynamics

This technique models physical systems with differential equations and numerically integrates the differential equations in terms of time. Forward dynamics usually gives the most realistic physical simulation results if the system model is well formulated. The following is a general differential equation for forward dynamics.

$$\begin{aligned} \ddot{\vec{\phi}} \;=\; & [I^*_{\phi\phi}]^{-1}(\vec{T}^I_\phi - \dot{\vec{\phi}}^T[P^*_{\phi\phi\phi}]\dot{\vec{\phi}} \\ & +\text{external force/moments} \\ & +\text{gravitational force} \\ & +\text{damping force + spring force}). \end{aligned} \tag{1}$$

where $[I^*_{\phi\phi}]$ is a generalized mass term, $\vec{\phi}$ is a vector of generalized joint angles, $\vec{T}^I_\phi$ is a vector of input joint torques, and $[P^*_{\phi\phi\phi}]$ is a centripetal and coriolis term, $\dot{\vec{\phi}}$ is a joint velocity vector, and $\ddot{\vec{\phi}}$ is a joint acceleration vector. *Generalized joint angles* include all degrees of freedom (displacements and rotation angles) of a free base articulated rigid body. For example of our human jumping model,

$$\vec{\phi} = \begin{bmatrix} x & y & \phi_1 & \phi_2 & \phi_3 & \phi_4 \end{bmatrix}^T \tag{2}$$

A simple example of forward dynamic simulation is dropping a pin-pointed chain [42]. We use the term simulation because there is no motion planning in a dropping chain problem. Previous work on animation of articulated rigid bodies [34, 48] implemented controllers which were based upon the analysis of the natural motion of the bodies in motion. It is not always easy, however, to synchronize motions when there are several different kinds of animated figures. Let us assume that there are two different figures which are trying to reach the same position within the same amount of time. Because their controllers are different, their motion cannot be easily synchronized.

An algorithm for pure forward dynamic simulation is given as follows.
(a). Model the articulated rigid body kinematically.
(b). Determine initial configuration.
(c). Calculate $[I^*_{\phi\phi}]$ and $[P^*_{\phi\phi\phi}]$ and other kinematic differential relations.
(d). Get $\vec{T}^I_\phi$ from the controller.

(e). Do numerical integration for Δt.

$$\dot{\vec{\phi}} = \dot{\phi}(t) + \int_t^{t+\Delta t} \ddot{\vec{\phi}} dt \tag{3}$$

$$\vec{\phi} = \phi(t) + \int_t^{t+\Delta t} \dot{\vec{\phi}} dt \tag{4}$$

where $\ddot{\vec{\phi}}$ is calculated by equation (1). We keep $\vec{\phi}$, in terms of time, as resulting data for display. Usually Δt (small time amount) should be very small if wild motion is involved.
(f). Use current $\vec{\phi}$ and $\dot{\vec{\phi}}$ for next calculation of $[I^*_{\phi\phi}]$, $[P^*_{\phi\phi\phi}]$, and $\ddot{\vec{\phi}}$ etc. Update time and go to (c).

2.2 Pure Inverse Dynamics

Computation of the exact input joint torques needed to achieve a desired motion is one of the major difficulties in the forward dynamics approach. That is to say it is not easy to make a motion controller. In the inverse dynamics approach we do motion planning (usually in Cartesian space) and calculate the needed joint torques to achieve the specified motion. Forward dynamic simulation is then done with the calculated joint torque values because it is possible that there are another subject involved in an animation that it can cause external forces/torques. In this approach all the kinematic data is determined by the animator. Without very careful (and difficult to achieve) kinematic specifications, this approach may lack reality, especially if the motions are vigorous. Let $\vec{u}$ be an *animation space* [42] in which the animator specifies motion. The major difficulty in motion control using inverse dynamics arises where the dimension of $\vec{\phi}$ is greater than that of $\vec{u}$. This leaves unconstrained degrees of freedom in joint space, which must be dealt with in order to solve for the dynamic parameters. Robotics formulations typically use Lagrangian multiplier methods [37, 4] to eliminate redundant dimensions. This approach may not be adequate if there is much redundancy in the joint space [42] because the animator cannot actively control the configuration of the articulated body. The shape of the body sometimes becomes distorted in order to satisfy the kinematic constraints which the animator has assigned. One approach to overcome this problem was proposed in [42]. This technique uses *extended* generalized coordinate space for the animation space. *Extended* means that we allow any variable, not only the position of the joint angle of the body, in our motion control. Motion planning is then also done in the extended space as well as Cartesian space. This method extends the animation space by adding more kinematic constraints to control the shape of the body. These constraints can be actively controlled by the animator, minimizing the number of artificial constraints arising from using Lagrangian multipliers. The extension of animation space can reduce the redundancy to a manageable degree so that the shape of the body may still be maintained in the presence of this low redundancy.

An algorithm for pure inverse dynamics technique is as follows.
(a) Plan motion in $\vec{u}, \dot{\vec{u}}, \ddot{\vec{u}}$.
(b) Calculate $[I^*_{\phi\phi}]$ and $[P^*_{\phi\phi\phi}]$ these are functions in terms of current $\vec{\phi}$. Also calculate other kinematic relations.
(c) Calculate

$$\dot{\vec{\phi}} = [G^{\phi}_{u}]\dot{\vec{u}} \tag{5}$$

where $[G^{\phi}_{u}]$ is inverse of a Jacobian matrix. We need to use Lagrangian multiplier method if the dimension of $\vec{\phi}$ is greater than $\vec{u}$.

(d) Calculate

$$\ddot{\vec{\phi}} = [G_u^{\phi}]\ddot{\vec{u}} + \dot{\vec{u}}^T[H_{uu}^{\phi}]\dot{\vec{u}} \quad (6)$$

where

$$\left[H_{uu}^{\phi}\right] = -[G_u^{\phi}]^T([G_u^{\phi}] \otimes [H_{\phi\phi}^{u}])[G_u^{\phi}] \quad (7)$$

Here $[H_{\phi\phi}^{u}]$ is Hessian matrix and $\otimes$ is explained at [42].
(e) Calculate torque by inverse dynamics,

$$\begin{aligned} \vec{T}_{\phi}^{I} = {} & [I_{\phi\phi}^{*}]\ddot{\vec{\phi}} + \dot{\vec{\phi}}^T[P_{\phi\phi\phi}^{*}]\dot{\vec{\phi}} \\ & -\text{external forces/moments} \\ & -\text{gravitational forces} \\ & -\text{damping/spring load} \end{aligned} \quad (8)$$

(f) Do forward dynamic simulation, as explained in section 2.1, for Δ t (usually 0.0004 second) with the calculated $\vec{T}_{\phi}$ as the input joint torque. With the new $\vec{\phi}, \dot{\vec{\phi}}$ go to (a).

2.3 Spacetime Constraints

This method is a variant of finite difference method of two point boundary value problem. This is based on inverse dynamics and the application of non- linear programming. A continuous time system is discretized [57], and approximate velocity and acceleration values at the discrete time units are calculated by finite difference. These velocities and accelerations are used to calculate joint torque values at these time steps. A performance index for optimization as well as the constraint functions are evaluated using these values and a search direction is decided by calculating the gradients of performance index and constraint functions. This computation leads to a new set of configurations for the rigid body which is better than the previous one in terms of the selected performance index. The process is repeated until a desired level of the performance index is reached(where Kuhn-Tucker condition is satisfied). This method eliminates forward dynamic simulation and therefore can trade off physical world simulation accuracy for computation speed. A major defect of this approach is that the resulting control can be far different from real forward dynamic simulation. Accurate simulation of the real world requires the use of a large number of variables, which means that the non-linear program solver may require considerable computational effort.

Let us consider the inverse dynamics equation (8). This equation holds throughout the entire simulation. The conversion from a continuous time system to a discrete time system is done using a set of state variable vectors, $\vec{\phi}_i, 0 \leq i \leq n$. For example, $\vec{\phi}_0$ represents the initial configuration , $\vec{\phi}_n$ represents the goal configuration, and $\vec{\phi}_i$ represents the (i+1)-th configuration of the articulated rigid body. The approximation of velocity and acceleration is done(not uniquely) as

$$\dot{\vec{\phi}}_i = \frac{\vec{\phi}_{i+1} - \vec{\phi}_{i-1}}{2h} \quad (9)$$

$$\ddot{\vec{\phi}}_i = \frac{\vec{\phi}_{i+1} - 2\vec{\phi}_i + \vec{\phi}_{i-1}}{h^2} \quad (10)$$

where i is an index for the discretized time system(index for a control node) and h is a time interval between the adjacent configurations. The resulting equation becomes

$$\vec{T}_{\phi_i}^{I} = [I_{\phi_i\phi_i}^{*}]\ddot{\vec{\phi}}_i + \dot{\vec{\phi}}_i^T[P_{\phi_i\phi_i\phi_i}^{*}]\dot{\vec{\phi}}_i$$

$$\begin{array}{l} -\text{external forces/moments} \\ -\text{gravitational forces} \\ -\text{damping/spring load} \end{array} \tag{11}$$

where $0 \leq i \leq n$.

In this method, it is important that the performance index include the relationships among the $(n+1)$ nonlinear equations specifying discrete configurations at the $(n+1)$ time steps. Otherwise, discretization may lead to independent constraints for each time step. This method is simple because motion planning is done in joint coordinates, not in Cartesian coordinates. It is simpler than the inverse dynamics method because it does not need to cope with a high degree of redundancy. It is less computationally intensive than pure forward dynamics simulation where the numerical integrations are expensive. The simplicity of this algorithm comes from joint space motion planning. We specify the start and the goal configurations of the body and *nonlinearly* interpolate them.

The rough algorithm for animation by nonlinear programming is as follows. Please note that there are many other ways to do the same task. Here we do not calculate the second derivatives of the performance index and constraints, contrary to [57], because the nonlinear program packages we use $(GRG2, VF02AD)$ do not need them. Please note that in this control, control variable is the same as state variable. Other optimal control(including suboptimal control presented here) usually have different set of control variables. This is due to the fact that the control is directly in joint space.

(a). Determine the number of control nodes. The number of control variables depend on the number of control nodes. Generalized joint angles at each *discrete* time instance, $\vec{\phi}_i, 0 \leq i \leq n$, become control variables.
(b) Initialize control variables.
(c) Calculate $\dot{\vec{\phi}}_i$, $\ddot{\vec{\phi}}_i$ using equation (9) and (10). (Apply finite difference method.)
(d) Calculate $[I^*_{\phi_i\phi_i}]$ and $[P^*_{\phi_i\phi_i\phi_i}]$.
(e) Calculate joint torques, by equation (11), and performance index J. J should include joint torque values as variables. Also calculate constraints if there are any.
(f) If Kuhn-Tucker conditions are satisfied, then exit. Otherwise continue.
(g) For $i = 1$ to $number - of - control - variables$ do
Perturb $i - th$ variable of control variables, while all other variables fixed. Compute first derivative of constraints and performance index.
(h) Determine a new set of control variables in which the new set gives a better value of performance index than that of the old one. Go to (c).

2.4 Other Related Work

There has also been some work which does not fall neatly into any of the categories previously discussed. The optimal control method has been applied by [40, 7, 41]. But usually optimal control method is very difficult to implement, or it is almost impossible to solve analytically if the problem is not simple. [40] used a shooting method to compute a best set of control for a human jumping *simulation* which takes really long time. They simulate up to before leaving the ground and then calculate the kinetic energy as a performance index. The most common approaches model a creature as an articulated body consisting of joints connected by links, including springs and dampers [53, 52]. Miller [36] has used a specialized spring and damper body for modeling the sliding motion of snakes and worms.

Lagrangian [8], Gibbs-Appell [51], Armstrong's Newton-Euler [3] and Featherstone's [34] formulations of forward dynamics have been applied to such systems. Iterative approaches, particularly the latter two, appear to provide the computational efficiency among existing simulation methods for these systems.

Issacs and Cohen [22] describe a system defining geometric constraints on the joints and specification of either accelerations or torques on the joints for open-loop systems. They further have developed a method [23] based on D'Alembert's principle of virtual work which implements inverse dynamic solutions for both open-loop and closed-loop systems. Zhao and Badler [60] present an iterative method to solve inverse kinematics with a given set of constraints, while Lee, et al. [32] define an iterative method to find a path satisfying both kinematic and dynamic constraints.

3 Simple Example of Suboptimal Control

Parameter optimization methods can be used in forward dynamic animation as well as in inverse dynamics methods. This approach is based upon the concept that any control history can be parameterized using a set of control nodes from which the control function is reconstructed by linear interpolation. The optimal control problem is, in this way, converted to a parameter optimization problem which we call suboptimal control. To conceptually convert the optimal control problem into a parameter optimization problem, the performance index and the control and state variable constraints must be converted into point conditions.

Consider a car moving along a trail. Mass of the car is m, and the engine force is $f(t)$. Starting

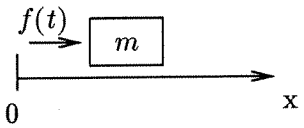


Figure 1: Example of a Car Moving along a Trail

from zero position, the driver wants to drive the car for 2 seconds, thus the final time is fixed. Also the car can be accelerated and decelerated. What we are interested is to find an optimal trajectory for a car so that we could display it. The performance index is set as follows.

$$J = \int_0^2 f^2(t)dt \tag{12}$$

Also there are two constraints such that final position is 1 meter away from the start position, and final velocity is zero.

Let us assume that we use three equally spaced(in terms of time so that each control node corresponds to time history 0, 1, and 2 seconds.), which are numbered 0, 1 and 2. Let use denote f_i to be force value at control node i, and x_i, $\dot{x}_i$, $\ddot{x}_i$ means position, velocity and acceleration of the car at control node i where $i = 1, 2, 3$. We want to linearly interpolate f_i values to get a continuous $f(t)$ function so that we could be able to do forward dynamic simulation using $f(t)$. We make f_0, f_1 and f_2 as control variables so that if we change its value, resulting $f(t)$ changes and simulation is affected. By linear interpolation, we can construct $f(t)$ such that

$$\begin{aligned} f(t) &= (f_1 - f_0)t + f_0, 0 \leq t \leq 1 \\ f(t) &= (f_2 - f_1)(t-1) + f_1, 1 \leq t \leq 2 \end{aligned}$$

Also we have dynamic equation such that $f = m\ddot{x}$ so

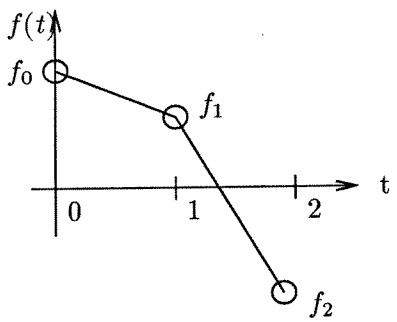


Figure 2: Example of a Linear Interpolation of $f(t)$

$$\begin{pmatrix} dx/dt \\ d\dot{x}/dt \end{pmatrix} = \begin{pmatrix} \dot{x} \\ \frac{f(t)}{m} \end{pmatrix} \tag{13}$$

By integrating $\ddot{x}$ once we get $\dot{x}$, and by twice we get x. This integration process corresponds to forward dynamic simulation. Here f_0, f_1 and f_2 becomes control variables and we initially guess these values. Then we do forward dynamic simulation(integrate) and get performance index, J, and constraint functions $c_1(t)$, $c_2(t)$ where

$$\begin{aligned} c_1(t) &= \dot{x}_2 - 0 \\ c_2(t) &= x_2 - 1 \end{aligned}$$

Then we perturb f_0, f_1 and f_2 (one at a time) by a small amount and do forward dynamic simulation(integrate) to calculate gradients of performance index and constraints. We repeat above process until Kuhn-Tucker condition is satisfied[33]. As a final result, we can get a time history of $x(t)$ so that we can display it for animation.

Please note that in the real optimal control, we have an infinite number of control nodes so that we can adjust $f(t)$ freely to give us a real optimal solution. But in our case, we have only three nodes and approximate the optimal $f(t)$. Although we do not have real optimal solution, the solution we get is pretty close. By using parameter optimization method, we can save huge amount of computation time.

4 Our approach - Suboptimal Control

Let us assume that we use (n+1) control nodes, ranging from 0 to n, and consider i-th state. The differential equation(system dynamic equation) $\dot{\vec{x}} = f(\vec{x}, t, \vec{u})$ holds everywhere. Here $\vec{x}$ denotes state variable(Same with x of previous example), $\vec{u}$ denotes a set of control.(The value computed by interpolating two adjacent control variables. Same with $f(t)$ of previous example.). Please note that the only difference from the previous example is final time. Here final time is not fixed.

Consider the parameter vector(vector of control variables)

$$\vec{z} = \begin{pmatrix} t_f & \vec{u}_0 & \vec{u}_1 & \dots & \vec{u}_n \end{pmatrix}^T \tag{14}$$

Initially, t_f and $\vec{u}_i$, $0 \le i \le n$, vectors(control variables) are given initially guessed values, and these $\vec{u}_i$s are used to calculate $\vec{u}(t)$. Here we have n time intervals (for simplicity assume that all control nodes are equally spaced)

$$\left[0, \frac{t_f}{n}\right), \quad \left[\frac{t_f}{n}, \frac{2t_f}{n}\right), \quad \dots, \quad \left[\frac{(n-1)t_f}{n}, t_f\right]$$

Here $\vec{u}_i$ represents the control nodes at time $\frac{it_f}{n}$. If time t belongs to $[\frac{(j-1)t_f}{n}, \frac{jt_f}{n}), 1 \leq j \leq n$, then $\vec{u}(t)$ is calculated by a linear interpolation(for simplicity) of $\vec{u}_{j-1}$ and $\vec{u}_j$. Also we convert the final time, t_f, into a parameter, such that $\tau = \frac{t}{t_f}$. In this way we fix the interval of integration $0 \leq \tau \leq 1$, and the system differential equation $\dot{\vec{x}} = f(\vec{x}, t, u)$ becomes

$$\frac{d\vec{x}}{d\tau} = t_f f(\vec{x}(\tau), t_f\tau, \vec{u}(\tau)). \tag{15}$$

This equation is integrated in terms of τ, $0 \leq \tau \leq 1$, using the prespecified boundary condition $\vec{x}_0$ such that $\vec{x}(0) = \vec{x}_0$, to compute the values of the states at the final time, $\vec{x}_f = \vec{x}(1)$. At the final time it should satisfy the system equality constraints

$$\Psi(t_f, \vec{x}_f) = 0, \tag{16}$$

and inequality constraints

$$\Theta(t_f, \vec{x}_f) \leq 0. \tag{17}$$

In summary, we replace the infinite number of control nodes in true optimal control by a modest number of control nodes and linear interpolation of adjacent nodes.

Let us take an example for the case of an human body which was used for our animation. Let performance index $J = t_f$ (for simplicity), and assume we have m degrees of freedom(in this case m is 6), l controls for a body, and $(n+1)$ control nodes(we used 21 control nodes so n is 20). Then the state variables become

$$\vec{x} = \begin{pmatrix} x & y & \phi_1 & \phi_2 & \phi_3 & \phi_4 \end{pmatrix}^T$$

the control variables become

$$\vec{z} = \begin{pmatrix} t_f & \vec{u}_0 & \vec{u}_1 & \dots & \vec{u}_n \end{pmatrix}^T \tag{18}$$

Because we worked on muscle model(not finished yet) as well as plain joint torque actuation model, control variable changes for each simulation. For plain joint torque actuation model, l is 4.

$$\vec{u}_i = \begin{pmatrix} T_{i1} & T_{i2} & T_{i3} & T_{i4} \end{pmatrix}^T \tag{19}$$

But for musculotendon actuation model, l is 8 because we used 8 muscles.

$$\vec{u}_i = \begin{pmatrix} w_{i1} & w_{i2} & w_{i3} & \cdots & w_{i8} \end{pmatrix}^T \tag{20}$$

and T_{ij}, $1 \leq j \leq 4$ represents the torque values for the (i+1)-th control nodes and the j-th body joint if we work on plain joint torque actuation model, or w_{ij}, $1 \leq j \leq 8$ represents the neural activation signal values to a j-th muscle for the (i+1)-th control nodes if we work on a musculotendon dynamic model. In our example of human jumping, we used six degrees of freedom, two for the tip of the foot and four for the body. (Figure 8) The system dynamics equations are

$$\begin{aligned} \frac{d\vec{\phi}}{d\tau} &= t_f\dot{\vec{\phi}} \\ \frac{d\dot{\vec{\phi}}}{d\tau} &= t_f[I^*_{\phi\phi}]^{-1}(\vec{T}^I_\phi - \dot{\vec{\phi}}^T[P^*_{\phi\phi\phi}]\dot{\vec{\phi}} \\ &\quad +\text{external force/moments} \\ &\quad +\text{gravitational force} \\ &\quad +\text{damping force + spring force)} \end{aligned} \tag{21}$$

A basic algorithm for suboptimal control is as follows.
(a). Determine the number of control nodes, number of constraints and stopping accuracy.
(b). Give initial guess for $\vec{z}$, final time and a set of torque or neural activation values for the control nodes.
(c). Specify initial conditions (initial configuration of the human body).
(d). Linearly interpolate control nodes to construct $\vec{u}(\tau)$. Calculate the performance index and constraints by forward dynamic simulation. (See section 2.1.)
(e). If Kuhn-Tucker conditions are satisfied, exit. Otherwise continue.
(f). For $i = 1$ to $(ln + 1)$ do
Perturb the i-th variable of $\vec{z}$, z_i with all other variables fixed. Compute the first derivative of the constraints and the performance index by forward dynamic simulation.
(g). Compute new $\vec{z}$ such that $J_{new} < J$, where J_{new} is a new value of the performance index. Go to (d).

5 Musculotendon and Its Activation Dynamics

This section is due to [38, 58]. Human body is a mechanically redundant actuator system. We have more muscles than DOF (degrees of freedom). Suboptimal control method allows for the incorporation of muscle dynamics because it check all possible combination of neural signals and do forward dynamic simulation. Then as a result it finds out a best set of control as well as its corresponding motion. In terms of [57]'s method, we need to solve inverse dynamics problem. But the problem becomes a little complicated because we need to distribute joint torques among larger number of muscles. (Also this process needs a big assumption that the muscle force is time independent.) In this process we need another parameter optimization method. But the assumption that the muscle force is time independent is wrong that the resulting force may have pretty steep slope which is not desirable. In this sense, forward dynamic simulation is better for musculotendon model. Figure 3 shows a lower extremity of human body with eight muscles used for our animation.

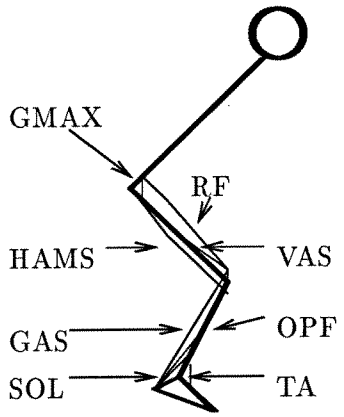


Figure 3: Schematic representation of Musculoskeletal model for the human jumping. Symbols appearing in the diagram are : soleus(SOL), gastrocnemius(GAS), other plantarflexors(OPF), tibialis anterior(TA), vasti(VAS), rectus femoris(RF), hamstrings(HAMS), and gluteus maximus(GMAX)

Figure 4 shows a control flow to control these muscles to generate the motion we desire.

Activation dynamics may have less importance compared to musculotendon dynamics or joint geometry. But it becomes important if we model an old person. Because the latency time for old people is longer and the activation rate is smaller than that of young people, the resulting

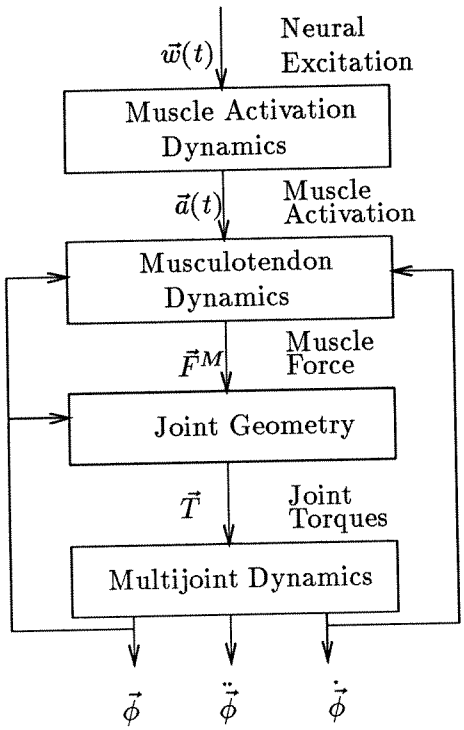


Figure 4: Control Flow of Musculotendon Human Body Model

motion is affected a lot. That is why old person respond slower than young person. Also more importantly, the activation dynamics eliminates robotic jerky motion because $a(t)$ is continuous and nonlinear. So the resulting control for the musculotendon model is a nonlinear control. The activation dynamic equation used in the simulation is as follows.

$$\frac{da(t)}{dt} = \frac{1}{\tau_{act}}[1 - a(t)]\,w(t) + \frac{1}{\tau_{deact}}[w(t) - 1]\,a(t)$$

where τ_{act} is activation rise time, τ_{deact} is activation decay time, $a(t)$ is activation signal, $w(t)$ is neural signal. Also Figure 5 shows activation signal $a(t)$ given neural signal $w(t)$ and other

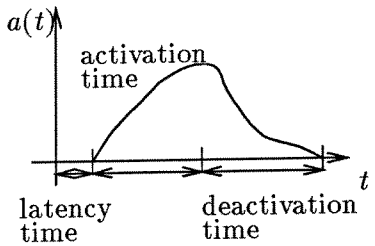


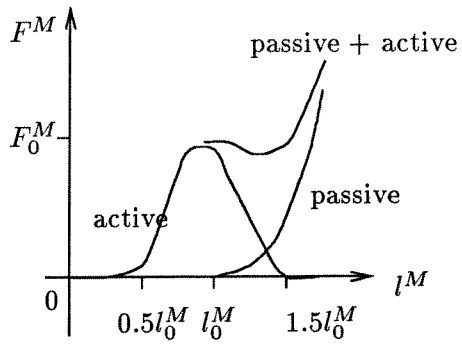
Figure 5: Muscle Activation and Deactivation

parameters related with time.

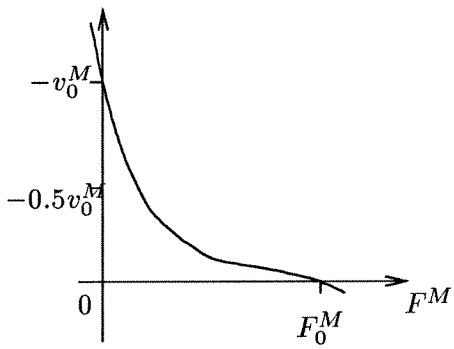
Muscle and tendon dynamic equation was derived from the experiments. [58] is a good reference for the musculotendon dynamics. Before going on, we want to explain the meaning of symbols. F means force, l means length. Superscript M means muscle, T means tendon, MT means musculotendon. So l^{MT} means a length of musculotendon actuator.

A. V. Hill observed that the following properties of muscle. Most of muscle model is based on his model. Also most muscle model of animals have pretty similar characteristics that we can use for almost all animals by changing several parameters such as peak isometric muscle force(F_0^M) that can be generated by a muscle at a length l_0^M, muscle pennation angle(α, muscle and tendon make an angle at the junction. This angle differs by muscle.), tendon slack length, stiffness of musculotendon elastic components etc.

Following is a muscle force-velocity characteristic.(Figure 6 (b)) Active muscle contains a damped element in series with the undamped elastic element. The damped element, if allowed to shorten, exerts a force which is smaller the faster the shortening. This is a kind of common sense that the human finger can move very fast but can not generate larger force. (Imagine a pianist playing.) But if he moves slow, then he can generate bigger force. In order to measure this characteristic, "quick release experiment" [56, 35, 26, 55, 54, 24] was done. The curve can be approximated by *arctan* function.



(a) Muscle force-length curve



(b) Muscle force-velocity curve

Figure 6: Muscle Force-Length curve and Force-Velocity Curve

Also force-length curve can be obtained from the experiment. (Figure 6 (a)) The passive properties of a muscle is easy to measure. We vary the force and measure the length of a muscle. The resulting curve can be modeled by exponential function. In order to measure the active properties, we need to activate the muscle and measure the length according to the force. The value we got contains both active and passive components. In order to calculate active part, we just need to subtract *passive* curve from the *passive* + *active* curve. According to the activation level, the magnitude changes. In terms of tendon, tendon is elastic.(We can consider it to be a linear spring.) So the tendon compliance is proportional to tendon slack length. Also tendon length affects

the musculotendon actuator. Because of so many different kinds of muscles, it is better to work on normalized version of musculotendon dynamics and then multiply the appropriate magnitude at the end of computation. Muscle force is a first order differential function.

$$\dot{F}^M = f(F^M, l^{MT}, \dot{l}^{MT}, a(t))$$

The musculotendon equation can be derived from muscle force-velocity curve, force-length curve and elastic tendon function.

The next computational segment in the control flow for a musculotendon dynamic model is joint geometry. We have muscle force which is acting linearly. In order to calculate joint torque actuated by musculotendon actuators, we need to calculate the perpendicular distance from the center of a joint to a line where muscle force is acting. Unfortunately some muscles span more than one joint. In terms of planar case we can easily apply trigonometric rule. But in terms of three dimensional case, it becomes very complicated to depend on the trigonometric rules. Moment arm of a muscle, r, can be defined as a time derivative of musculotendon length with respect to a joint angle, θ.

$$r = -\frac{d|l^{MT}|}{d\theta}$$

For the soleus, moment arm is computed to be

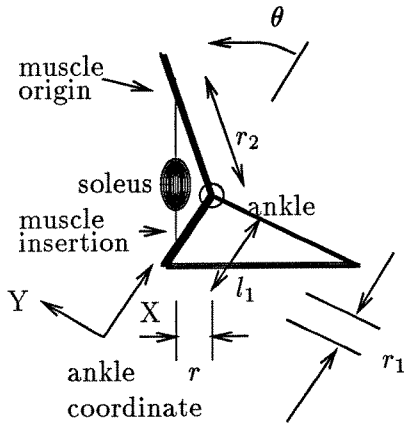


Figure 7: An example of calculating moment arm : for soleus

$$\begin{aligned} r_{soleus} &= \frac{(l_1 - r_1) r_2 sin\theta}{l^{MT}} \\ \text{where} \quad & l^{MT} = \sqrt{(l_1 - r_1 + r_2 cos\theta)^2 + (r_2 sin\theta)^2} \\ T_{soleus} &= r_{soleus} F^M_{soleus} \end{aligned}$$

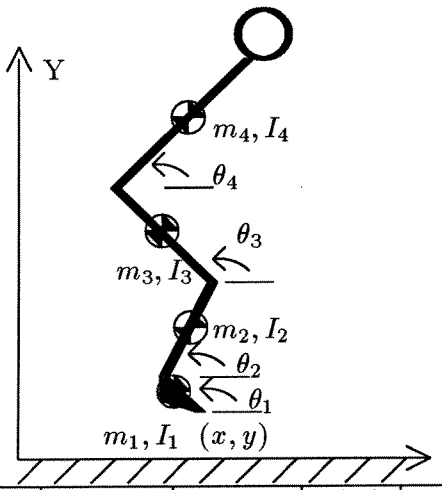
Here T_{soleus} is a ankle joint torque caused by soleus muscle force F^M_{soleus}.

Finally we got a set of joint torques which is actuated by a set of musculotendon actuators. Multijoint dynamics part is pretty similar to usual robotic dynamic equation computation, except that the joint torque is computed from the linear actuators rather than the rotary joint actuators. Because the current configuration affects joint geometry and the moment arm changes according to the current configuration, and musculotendon actuator length is determined by current joint angle, we need a feedback of joint angles as can be seen from the Figure 4. Also current joint angular velocity affects the velocity of linear musculotendon actuator so that we need a feedback of angular velocity.

Also our model is important in the sense that we do pretty accurate simulation which can predict the amount of forces acting on the muscles or foot. So that the result can be used for surgery or in making an artificial limb. In terms of suboptimal control, it is a best method ever known in terms of precise simulation to get a best set of motion control. Someone may try to measure exact EMG(Electro-Myo-Graphic) signal, but they can not measure signals for the muscles which lie deep inside the body. Our model select a performance index which approximates human motion strategy and guess the initial neural signal. As a result we may get a complete set of neural excitation signal for a human motion. (The argument here is based on the assumption that EMG signal is proportional to real neural signal. EMG signal can be measured by putting a special sensor at a muscle.)

6 Basic Dynamic Equations

We have developed/implemented a recursive algorithm of Generalized principle of D'Alembert's. But recursive Newton-Euler formulation [11] also can be easily modified to apply to the free base case. The equations are derived for the planar case, and we obtain two additional equations from the base reaction force relations. Collection of similar terms and simplification leads to equation (1). Figure 8 shows a schematic representation of the four segment model of a jumping human. Please note that the $[I^*_{\phi\phi}]$ matrix is positive definite and symmetric so that we can easily recover the form of equation (1). We have used linear spring and damper models for joint torsional muscles(this is just for plain joint torque actuation model), and the ground reaction force for heel is modeled by having a torsional exponential springs and dampers at toe. For complete simulation with six DOF model, we need horizontal and vertical stiff spring and dampers at the toe.



Segment	$m_i(Kg)$	$l_{c_i}(m)$	$l_i(m)$	$I_i(Kgm^2)$
Foot	2.2	0.14275	0.25	0.0155
Shank	7.5	0.274	0.435	0.065
Thigh	15.15	0.251	0.400	0.126
HAT	51.22	0.343	0.343	6.814

Figure 8: Schematic representation of the four-segment model of a jumping human. m_1, m_2, m_3, m_4 are the lumped masses of the foot, shank, thigh, and HAT(head, arms, and truck) respectively; I_1, I_2, I_3, I_4 are the moments of inertia of the foot, shank, thigh and HAT respectively.

7 Dynamic Simulation

This section only applies to plain joint actuation case; We are still developing a foot model for better animation. As you may see we have just 8 muscles. The foot muscles are missing. So the resulting animation is a kind of animation of disabled who is missing toes. The following is a result of simulation with joint torsional actuators. Generally the human jumping animation is really challenging task. The coordination of each muscle is very important that if a relatively small disturbance in a control signal will corrupt the jumping motion completely. You can imagine that it takes pretty long time for a human to be able to work. It takes longer to jump up.

The application of this method is shown in the movie, *Ruined Race*. *Ruined Race* has inchworms racing. A human figure appears in the animation and jumps forward to step on the worms. The first four inchworms were animated using an inverse dynamics method with extended generalized coordinate transformations. The extended generalized coordinate transformation method was used to actively constrain the motion and thus fix the shape of the inchworm. Lagrangian multiplier methods were used to control the remaining redundancies. The remaining worms were animated using the *Spacetime Constraints* method. The shape of the worm is interpolated between the start and goal configurations. This method is effective when there are no wild motions involved.

The human figure is simulated using suboptimal control. In the simulations we shift the number of degrees of freedom from four to six and six to four during the course of the motion. We keep four degrees of freedom when the body is touching the ground, but when it is in the air, six degrees of freedom are required. Whenever the tip of the foot reaches the ground, we shift from six degrees of freedom to four degrees of freedom. We shift from four degrees of freedom to six when the vertical force acting at the center of gravity of the body becomes greater than zero. This simulation with varying degrees of freedom was compared with a simulation in which six degrees of freedom were maintained throughout the entire simulation. The latter simulation was universally poor.

For the springs and dampers used in the model, we used both exponential and linear models at the bottom of the foot, linear springs and damper for joint muscles, and exponential spring and damper for the joint limit. Figure 9 shows several frames from the dynamic animation of the human jumping.

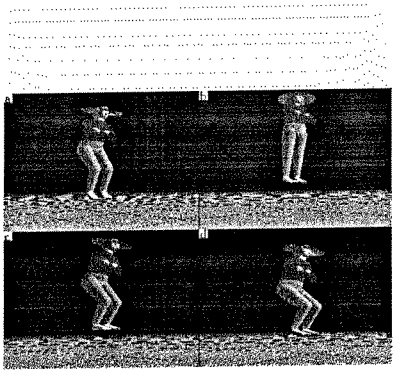


Figure 9: Dynamic Animation of Human Jumping

8 Conclusion

This paper introduces musculotendon actuators and its geometry to be used for realistic animation. Muscle is highly nonlinear spring and dampers and its activation dynamics affects the resulting motion. The importance of muscle dynamics was explained at the section 5. Also musculotendon activation dynamics affects the human motion. More importantly musculotendon is linear actuator so that the resulting joint torque is heavily dependent on the configuration; if moment arm is very small, the resulting joint torque becomes small however big the muscle force is. By applying our musculotendon skeletal dynamic model to the work of [32], we could get a better result because the constraints given by musculotendon skeletal model may be more accurate. The human data we use were achieved from several papers including[12]. Also we want to point out that we can not replace muscles by a rotary joint torque actuators because the joint torque which is calculated from a set of muscles is a unidirectional function.

This paper has also formulated and demonstrated a technique for controlling the motion of an articulated rigid body. The technique is based on a simplification of optimal control techniques in which the continuous control nodes are replaced by a finite number of control nodes and linear interpolation among the nodes. The performance index which we usually use is a nearly flat convex function. Therefore, the solution we obtain using suboptimal control is close to the real optimal value. Additionally, the suboptimal control technique can be used to define body control with the inclusion of a final time constraint in a straightforward way. Note that from an initial start configuration suboptimal control will test a variety of cases before determining the *best* controller. Our control variables(for a musculotendon model) is neurosignals ranging from 0 to 1, which is better for our parameter optimization. Also we want to stress that the suboptimal control is essential if we work on a musculotendon skeletal dynamic mode. Because this model is actuator redundant system that we can not do exact simulation.

The complexity of the algorithm depends upon the number of control variables used. We used 28 control nodes and the total number of variables used was 113 for a rotary joint actuator. We used the GRG2 [30] nonlinear program solver for the human jumping animation, and a C version of VF02AD [44, 45] for the *Spacetime Constraints* method for the worm race. (Also we are using for musculotendon model human jumping.) GRG2 is based on reduced gradient method and VF02AD is based on SQP. VF02AD is quicker in solving a problem. The dynamic equation was derived by hand. The time step used for integration was 0.0001 seconds.

Results of this work suggest that simple spring and damper models for human joints are not sufficient for quality animations. The real musculoskeletal dynamics [19, 17, 18, 28, 27, 40] is very different from the simple spring and damper model.

9 Acknowledgements

We thank Leon Lasdon for his help in the code GRG2, Jesse Driver and A. T. Campbell for their support in taking images. This work was also supported in part by a grant from IBM Research Initiation Grant Program to support Interdisciplinary Computer Science Research.

References

[1] R. Alexander. *Handbook of Physiology : The Nervous System*, chapter Mechanics of skeleton and tendon. American Physiology Society, 1981.

[2] Americana. *The Encylopedia Americana*, chapter Measuring Worm. Grolier Inc., 1990.

[3] W. Armstrong, M. Green, and R. Lake. Near-real-time control of human figure models. *IEEE Computer Graphics and Applications*, pages 52–61, June 1987.

[4] H. Asada and J. Slotin. *Robot Analysis and Control.* John Wiley and Sons, New York, NY, 1985.

[5] R. Brand, D. Pederson, and J. Friederich. The sensitivity of muscle force prediction to changes in physiologic cross sectional area. *J. Biomechanics*, 19(8):589–596, 1986.

[6] Britannica. *The New Encylopedia Britannica*, chapter Measuring Worm. Encylopedia Britannica Inc., 1990.

[7] L. Brotman and A. Netravali. Motion interpolation by optimal control. *Computer Graphics*, 22(4):309–315, August 1988.

[8] A. Bruderlin and T. Calvert. Goal-directed, dynamic animation of human walking. *Computer Graphics*, 23(3):233–242, July 1989.

[9] A. Bryson and Y. Ho. *Applied Optimal Control : optimization, estimation, and control.* Halsted Press, Washington D. C., 1975.

[10] D. Butler, E. Grood, F. Noyes, and R. Zernicke. Biomechanics of ligaments and tendons. *Exercise Sport Science Review*, 6, 1979.

[11] J. Craig. *Introduction to Robotics Mechanics and Control.* Addison-Wesley, Reading, MA, 1986.

[12] S. Delp, J. Loan, M. Hoy, F. Zajac, E. Topp, and J. Rosen. An interactive graphics-based model of the lower extremety to study orthopaedic surgical procedures. *IEEE Tr. on Biomedical Engineering*, 37(8), 1990.

[13] R. Freeman. *Discussions with Dr. Freeman.* 1990.

[14] R. Freeman and D. Tesar. Dynamic Modeling of Serial and Parallel Mechanisms / Robotic Systems : Part II - Applications. In *Trends and Developments in Mechanisms, Machines and Robotics, 20th Biennial Mechanisms Conference*, 1988.

[15] R. Freeman and D. Tesar. Dynamic Modeling of Serial and Parallel Mechanisms / Robotic Systems : Part I - Methodology. In *Trends and Developments in Mechanisms, Machines and Robotics, 20th Biennial Mechanisms Conference*, 1988.

[16] M. Girard and A. Maciejewski. Computational modeling for the computer animation of legged figures. *Computer Graphics*, 19(3):263–270, July 1985.

[17] H. Hatze. A complete set of control equations for the human musculo-skeletal system. *Journal of Biomechanics*, 10:977–805, 1977.

[18] H. Hatze. A comprehensive model for human motion simulation and its application to the take-off phase of the long jump. *Journal of Biomechanics*, 14:135–142, 1977.

[19] H. Hatze. Neuromusculoskeletal control systems modeling - a critical survey of recent developments. *IEEE Tr. on Automatic Control*, 25(3), June 1980.

[20] M. Hoy, F. Zajac, and M. Gordon. A musculoskeletal model of the human lower extremity : The effect of muscle, tendon, and moment arm on the moment-angle relationship of musculotendon actuators at the hip, kee, and ankle. *J. Biomechanics*, 23(2):157–169, 1990.

[21] D. Hull. *Lecture Notes in Optimal Control I.* University of Texas at Austin, Austin, TX, 1991.

[22] P. Isaacs and M. Cohen. Controlling dynamic simulation with kinematic constraints, behavior functions and inverse dynamics. *Computer Graphics*, 21(4):215–224, July 1987.

[23] P. Isaacs and M. Cohen. Mixed methods for complex kinematic constraints in dynamic figure animation. *Visual Computer*, pages 296–305, April 1988.

[24] B. Jewell and D. Wilkie. An analysis of mechanical components in frog's striated muscle. *J. Physiology*, 143, 1958.

[25] T. Kane and D. Levinson. *Dynamics : Theory and Applications.* McGraw-Hill, New York, NY, 1985.

[26] B. Katz. The relation between force and speed in muscular contraction. *J. Physiology*, 96, 1939.

[27] G. Khang and F. Zajac. Paraplegic standing controlled by functional neuromuscular stimulation : Part ii - computer simulation studies. *IEEE Tr. on Biomedical Engineering*, 36(9):885–894, 1989.

[28] G. Khang and F. Zajac. Paraplegic standing controlled by functional neuromuscular stimulation : Part i - computer model and control-system design. *IEEE Tr. on Biomedical Engineering*, 36(9):873–884, 1989.

[29] G. Kinzel and L. Gutkowski. Joint models, degrees of freedom, and anatomical motion measurement. *ASME J. Biomechanical Engineering*, 105, 1983.

[30] L. Lasdon, R. Fox, and M. Ratner. Nonlinear Optimization using the Generalized Reduced Gradient Method. Department of Operations Research 325, Case Western Reserve University, October 1973.

[31] L. Lasdon and A. Waren. *GRG2 User's Guide*, 1989.

[32] P. Lee, S. Wei, J. Zhao, and N. Badler. Strength guided motion. *Computer Graphics*, 24(4):253–263, August 1990.

[33] D. Luenberger. *Linear and Nonlinear Programming.* Addison-Wesley, Reading Ma., 1984.

[34] M. McKenna and D. Zeltzer. Dynamic simulation of autonomous legged locomotion. *Computer Graphics*, 24(4):29–38, August 1990.

[35] T. McMahon. *Muscles, ,Reflexes, and Locomotion.* Princeton University Press, 1984.

[36] G. S. Miller. The motion dynamics of snakes and worms. *Computer Graphics*, 22(4):169–178, August 1988.

[37] D. N. Nenchev. Redundancy resolution through local optimization : A review. *Journal of Robotic Systems*, 6(6):769–799, 1989.

[38] M. Pandy. *Lectures in Musculoskeletal Biomechanics.* University of Texas, 1992.

[39] M. Pandy, F. Anderson, and D. Hull. A parameter optimization approach for the optimal control of large-scale musculoskeletal systems. *Submitted to the Journal of Biomechanical Engineering*, March 1991.

[40] M. Pandy, F. Zajac, E. Sim, and W. Levine. An optimal control model for maximum-height human jumping. *Journal of Biomechanics*, 23(12):1185–1198, 1988.

[41] M. Panne, E. Fiume, and Z. Vranesic. Reusable motion synthesis using state-space controllers. *Computer Graphics*, 24(4):225–234, August 1990.

[42] J. Park, D. Fussell, and J. C. Browne. Motion control using extended generalized coordinate transformations. *submitted for publication*, Apr. 1992.

[43] C. Phillips, J. Zhao, and N. Badler. Interactive real-time articulated figure manipulation using multiple kinematic constraints. *Computer Graphics*, 24(2):245–250, 1990.

[44] M. Powell. A Fast Algorithm for Nonlinearly Constrained Optimization calculations. In *Dundee Conference on Numerical Analysis*, 1978.

[45] M. Powell. Algorithms for nonlinear constraints that use lagrangian functions. *Mathematical Programming*, 14:224–248, 1978.

[46] M. Powell. *VF02AD User's Guide*, 1978.

[47] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes.* Cambridge University Press, Cambridge, England, 1986.

[48] M. Raibert and J. Hodgins. Animation of dynamic legged locomotion. *Computer Graphics*, 25(4):349–358, July 1991.

[49] J. Ritchie and D. Wilkie. The dynamics of muscular contraction. *J. Physiology*, 143, 1958.

[50] M. Thomas and D. Tesar. Dynamic modeling of serial manipulator arms. *Transactions of the ASME*, 104:218–228, Sep. 1982.

[51] J. Wilhelms. Using dynamic analysis for realistic animation of articulated bodies. *IEEE Computer Graphics and Applications*, 7(6):12–27, June 1987.

[52] J. Wilhelms. Dynamic animation : Interaction and control. *Visual Computer*, Dec. 1988.

[53] J. Wilhelms. *Making them Move*, chapter Dynamic Experiences. Morgan Kaufmann Publishers, 1991.

[54] D. Wilkie. Measurement of the series elastic component at various times during a single twitch. *J. Physiology*, 134, 1956.

[55] D. Wilkie. The mechanical properties of muscle. *British Medical Bulletin*, 12(3), 1956.

[56] D. Winter. *Biomechanics and Motor Control of Human Movement*, chapter Chap 6 : Muscle Mechanics.

[57] A. Witkin and M. Kass. Spacetime constraints. *Computer Graphics*, 22(4):159–168, August 1988.

[58] F. Zajac. Muscle and tendon : Properties, models, scaling, and application to biomechanics and motor control. *Critical Review in Biomedical Engineering*, 17(4), 1989.

[59] F. Zajac and M. Gordon. Determining muscle's force and action in multi-articular movement. *Exercise Sport Science Review*, 17, 1990.

[60] J. Zhao and N. Badler. Real Time Inverse Kinematics with Joint Limits and Spatial Constraints. MS-CIS 89-09, University of Pennsylvania, January 1989.