Efficient Synthesis of Physically Valid Human Motion for Computer Animation

by

Anthony C. Fang

B. Sc. (Hons), National University of Singapore, 1994

Sc. M., Brown University, 1999

A dissertation submitted in partial fulfillment of the
requirements for the Degree of Doctor of Philosophy
in the Department of Computer Science at Brown University

Providence, Rhode Island

May 2003

This dissertation by Anthony C. Fang is accepted in its present form by the Department of Computer Science as satisfying the dissertation requirement for the degree of Doctor of Philosophy.

Date 5/2/03 — Nancy S Pollard

Nancy S. Pollard, Director

Recommended to the Graduate Council

Date 4/18/03 — [illegible]

John F. Hughes, Reader

Date 5/2/03 — [illegible]

Jessica K. Hodgins, Reader
(Robotics Institute, Carnegie Mellon University)

Approved by the Graduate Council

Date 5/5/03 — Karen Newman

Karen Newman, Ph.D
Dean of the Graduate School

Acknowledgements

With thanks to my advisor, Nancy Pollard, whose guidance and encouragement made the completion of this research possible. Much thanks to John Hughes for guiding me through the early years of my doctoral studies, and to Jessica Hodgins for providing helpful comments on the thesis.

Contents

List of Figures

Chapter 1

Introduction

1.1 Elements of Physical Validity

Digital character animation is a powerful storytelling medium. Whether it is a simple sequence of frames or an epic animated adventure, the plausibility of an animation derives in no small part from its relevance to the viewer's physical existence. This includes having attributes such as conformance to physical laws of motion, that bodies do not in general penetrate each other, and not violating structural limits of the character. Such qualities in a motion are *objective* in a sense that they can be described through rigorous mathematical definitions. On the other hand, the credibility of the character's portrayal draws from its ability to mimic its real life counterparts. Characteristics such as behavioral responses, expressiveness, or the varied manners in which it carries out tasks, such as conserving energy expenditure, impart a certain lifelikeness to the animated character. These are *subjective* attributes and therefore less amenable to mathematical treatment. Put together, the perception of both objective and subjective aspects mark the qualities of a realistic character animation.

It is often helpful to distinguish between the two — the objective qualities set the basis for physical validity while the subjective qualities make the character's movement appear real. In many animation systems, such as keyframe systems, the burden of getting both right lies with the animator, who through repeated adjustments and trials, aims to do justice in both respects. This assures the animator of absolute control over the creative process, at the expense of having to visually arbitrate intricate physical laws without an explicit physical model.

The nature of physical validity is well-understood and achieving it in an animation ought to admit some degree of automation. One could, for instance, envision an animation system where the animator interactively designs a character's motion within the space of physically valid motion — as the animator makes changes to the motion, the system responds with changes that preserve physical correctness. These corrections however should not excessively change a motion that is being interactively refined. Neither should the animator's ability to make changes be stifled, for otherwise such a tool would be frustrating to use. Isolating the elements of physical validity provides the basis for controlled changes in such an animation system.

1.2 Elements of Style

Physical validity should be contrasted with the notion of physical optimality, the latter being a concept often associated with physically-based character animation. Validity implies that the motion is consistent with the laws of motion, while optimality suggests that in addition to being valid, it is optimal with respect to some measurable dynamic quantity. In the present literature, a commonly used quantity is some approximated measure of energy expenditure.

Physical optimality demands a particular style. That a certain style is present in all motion is obviously not true — low energy consumption may be appropriate for modeling, say, distance running, but in other instances character style, expressiveness, or performance plays an equally important part. To an animation artist, telling the story almost always overrides concerns of energy consumption. In a dance, rhythmic flow or elegance is desirable. In gymnastics or platform diving, strict and clean movements score high marks.

But measures of style are inherently subjective in nature, and perhaps best chosen by the animator for a given animation. Computationally, stylistic measures range from cheap, to expensive, to presently uncomputable. For instance, it has been hypothesized that the characteristic movements of personalities can be modeled as a stylistic measure. Obviously this is not possible with the present body of knowledge in human behavior, but imagine the possiblities if we were able to do that! Not only would we be able to populate virtual worlds with realistic moving characters, but the characters could also have individualized personalities — a seeming incarnation of oneself in the virtual world.

In a limited way, we are already progressing towards modeling personalities. In motion capture, the movements of subjects are recorded and bestowed upon virtual characters. Often the resolution of the capture is fine enough to capture the distinguishing characteristics of individuals. If we were to create a different but similar motion using a motion capture as a reference, then the style we desire is one which mimics the performer.

The insight is that a substantial part in animating characters involve matters of style, of which the "minimal-energy-style" is one, the "performer-style" is another, and other stylistic measures can certainly be envisioned. In any case, while the stylistic aspects of the motion may vary. the need for physical validity in motions intended to appear realistic is invariant and omnipresent.

1.3 Optimization Synthesis

Traditional animation methods provide great control to the artist. but do not provide any tools for automatically creating realistic motion. Dynamic simulations on the other hand generate physically correct motion, but do not provide sufficient control for an artist or scientist to create desired motion. Creating realistic motion of articulated characters with high-level specifications. keyframing motion with sparse and few key poses, or freestyle directing of virtual human character movements. are among the coveted goals of digital character animation.

One appealing vision in animation is that the animator should be able to create motion with a coarse description of a motion — and that the resulting motion should be physically correct yet optimal in some way. This is the fundamental ideology of motion synthesis methods that employ nonlinear optimization techniques as a means to construct physically optimal motion. Such methods accept a coarse description of goals, such as key poses or kinematic events. and formulate a constrained optimization problem such that the optimal solution is one which has the desired behavior.

The optimization approach to animation has proven useful for editing human motion capture data or for creating short segments of entirely new motions for simple characters. Current systems can adapt motion capture data to a wide variety of realistic and cartoon characters. cause characters to avoid obstacles that were not present when the original motion was performed. and seamlessly string together motions captured from different actors in different studios. However. the ambitious goal of being able to synthesize, as oppose to modify. humanlike motion in a high-level manner using physically based optimization remains to be fulfilled.

1.4 Thesis Synopsis

Optimization is a promising approach to motion synthesis. However. computational costs remain a major stumbling block in extending present physically based optimization approaches to full-scale humanlike articulated characters. Complexity analysis of the optimization cycle reveals that in present optimization approaches, per-iteration algorithmic cost is quadratic in the degrees-of-freedom of the articulated character. The proof of this cost is trivial — the density of the first-derivative matrix of the dynamics equations is of quadratic order. consequently any gradient-based optimizer is subjected to this quadratic lower-bound. This quadratic computational effort limits the viability of this approch for synthesizing the kind of motion we would like to see in human character animation. Despite advances in hardware speeds, the feasibility of the optimization approach to synthesizing humanlike motion still severely challenges today's machine. For this approach to become viable, the theoretical computational cost of physically based optimization must be addressed.

Various methods have been proposed to address this problem. These methods include ignoring or approximating physics, or approximating the dynamics of the character by reducing its degrees of freedom. The common result is that physical validity is compromised and that the motion synthesized via these approximations does not guarantee physical correctness.

We propose to view the problem from a different angle. We first observe that in present physically based optimization approaches, the physics constraints and the objective function are intimately coupled — the dynamics equations of motion set up a number of physics constraints as a function of joint torques, while joint torques are in turn used in the objective function for minimizing energy expenditure. We then note that when this dual-function of joint torques is taken apart, physical validity can be achieved without explicitly computing terms involving joint torques. This is done within the context of the Newton-Euler dynamics formulation by discriminating force quantities into two sets: joint actuation forces, and external forces. Algorithmically, we note that efficient recursive algorithms can be formulated to compute and differentiate external forces on the system in linear-time. We subsequently formulate physics constraints as a function of external forces and, via the chain rule of partial derivatives, compute all first derivatives of physics constraints in time linear in the degrees of freedom of the articulated character.

The main attractiveness of this approach is that it sets forth the basis in which physical validity can be achieved in linear-time, leaving open the task of designing objective functions. Several objective functions have been proposed in the past, some of which include proximity to reference motion, smooth acceleration profiles, or sum square torques which approximates energy expenditure. Often, objective functions have to be designed within the context of the application. For instance, when adapting motion capture data, it is desirable that the end result retain characteristics of the original data. And in applications such as motion synthesis, retaining a good measure of control over the resulting motion is important. In all cases, however, physical validity is a desirable factor.

This thesis describes the synthesis of physically valid humanlike motion within the framework of constrained optimization. We use humanlike figures, but the methods apply to any articulated structures that may be approximated with rigid-body linkages. This includes most robotic mechanisms or other artificial computer-created models, such as the now famous Luxo lamp.

1.4.1 Thesis Outline

This thesis is organized as follows.

- Chapter 2 provides an overview to trajectory optimization systems and sets up the nomenclature for the rest of the document.
- Chapter 3 reviews works where optimization plays a fundamental role in motion synthesis and sets the background to our work.
- Chapter 4 describes a scheme for enforcing physical validity in motion optimization.
- Chapter 5 describes an optimal time formulation for computing first-derivatives of physics constraints.
- Chapter 6 extends the linearity to the numerical optimization using a typical nonlinear optimization model;
- Chapter 7 illustrates some results which can be generated using our approach.
- Chapter 8 summarizes the results of this thesis, the shortcomings of our approach, and discusses future directions.

Chapter 2

Overview of Trajectory Optimization of Human Motion

2.1 Human modeling

In theory, the formulation of biomechanical problems would require a knowledge of the motion of each individual particle within the body. The forces between these particles, together with those exerted by muscles and the outside world, would then have to be accounted for. This however is not a practical proposition — even if knowledge of particle properties was available, the computational toll of achieving this level of detail would be insurmountable, and the volume of results obtained would be less than comprehensible.

As a compromise, the human body is usually modeled as a number of *linkages*, where each linkage is assumed to have constant length and inertial properties. In addition, linkages are connected at frictionless *joints*, where the linkages articulate about a constant center of rotation. This structure is represented as a tree topology, where the linkage at the root of the tree is called its *base*. Such a representation is known as the articulated body model, an instance of which is known as an *articulated character*.

The complexity of any calculations on the articulated model is thus tied to its number of linkages. Obviously, the higher the number of linkages, the closer it is to modeling the biomechanical characteristics of an actual person. Various levels of approximations may be obtained by grouping parts of the body into linkages. For instance, the spine of the human body consists of 24 movable vertebrae, which, as a coarse approximation, is often clustered into two or three linkages.

The inertial characteristics of the linkages may be borrowed from past studies in anthropometry.

A particularly useful reference is the statistical studies of [53, Chapter 6], who expressed mass of linkages as a percentage of body weight, moment of inertia as a percentage of linkage length, and linkage lengths as a percentage of stature. Thus, given height and weight, an instantiation of an articulated character representing a typical male or female is obtained. The articulated human characters used in this work are adapted from this source.

2.2 Generalized degrees of freedom

Each joint is endowed with a number of degrees of freedom in which it may provide relative movements to its adjoining linkages. The collective degrees of freedom of all joints in the articulated character are referred to as its *generalized degrees of freedom* and are specified by its *generalized coordinates* denoted by a vector q:

$$q = [q_1, ..., q_D]^T \tag{2.1}$$

where D is the total number of degrees of freedom of the character. Given the vector q, the *pose* of the character is fully specified.

The *state* of the articulated character consists of its pose and its movement to first order approximations. The first order motion of linkages is given by the generalized velocities, denoted:

$$\dot{q} = [\dot{q}_1, ..., \dot{q}_D]^T \tag{2.2}$$

Given a state, $q, \dot{q}$, the instantaneous positions and velocities of all linkages of the character are fully specified.

Newton's laws of motion assert that a body in motion tends to move at constant velocity, unless acted on by intervening forces. To study this effect, quantities for forces and rates of change of velocities are introduced. The rate of change of generalized velocities is its acceleration, given by:

$$\ddot{q} = [\ddot{q}_1, ..., \ddot{q}_D]^T \tag{2.3}$$

In articulated characters capable of actuating forces via muscle strengths, the actuated forces acting about the joints are represented as a *generalized force* vector, denoted:

$$\tau = [\tau_1, ..., \tau_D]^T \tag{2.4}$$

In addition, a number of external forces, such as gravity or contact forces from the foot, may be acting on the character. Let $\underline{f}_k$ be a set of linear forces applied at the points $\underline{p}_k$ located on any linkage of the character. The summed effect of this set of forces may be transformed to generalized coordinate space as follows:

$$ext_i = \sum_k (\frac{\partial \underline{p}_k}{\partial q_i})^T \underline{f}_k \tag{2.5}$$

where $\frac{\partial \underline{p}_k}{\partial q_i}$ is the rate of change of $\underline{p}_k$ with respect to q, commonly known as the ***kinematic Jacobian*** of $\underline{p}_k$. This relation is based on the principle of virtual work and is commonly known as the Jacobian transpose rule.

2.3 Trajectories over an interval

We will often be studying the motion of an articulated character over a real interval, $[t_0, t_1]$. The generalized coordinates are therefore real functions over time, denoted by $q(t) \in \Re^D$, as are the generalized forces, $\tau(t) \in \Re^D$ and $ext(t) \in \Re^D$. We shall term these time-varying functions the ***trajectories*** of the system. In particular, where trajectories are part of the unknowns in an optimization problem, we shall refer to it as the ***trajectory optimization problem.*** In the calculus of variations, such a problem is also known as a ***variational*** problem.

$q(t)$ is assumed to be twice-continuously differentiable, thus $\dot{q}(t)$ and $\ddot{q}(t)$ exist at all t. No differentiability requirement is placed on $\tau(t)$ throughout this document. For compactness, the time parameterization will be dropped when discussing instantaneous computations, such as in chapters 4 and 5.

2.4 Discretization of trajectories

When trajectories over a continuous domain are unknown, the resulting formulation represents an infinite-dimensional problem, which may be converted to a finite-dimensional problem by approximating each trajectory as a linear combination of a finite set of ***basis functions.*** The basis functions are denote by $\phi_k(t)$, such that:

$$q_i(t) = \sum_{k=1}^{K} c_{i,k}\phi_k(t) \tag{2.6}$$

where K denotes the number of basis functions per trajectory. For a given trajectory i, the set of $c_{i,k}$'s for all k is commonly known as its ***control points.*** The basis functions are assumed twice-continuously differentiable such that:

$$\dot{q}_i(t) = \sum_{k=1}^{K} c_{i,k}\dot{\phi}_k(t) \quad \text{and} \quad \ddot{q}_i(t) = \sum_{k=1}^{K} c_{i,k}\ddot{\phi}_k(t) \tag{2.7}$$

In addition, the basis functions are said to have *local* support if at any time t, only a constant number of $\phi_k(t)$ are non-zero.

The choice of basis functions directly affects the bandwidth of the derivative matrices associated with the objective and constraint functions. If all basis functions have strictly local support, as is the case for cubic B-spline bases, the derivative matrices have a block sparsity pattern which may be exploited in the solution method. If hierarchical bases, such as wavelet bases [47] or hierarchical B-spline bases [41], are used, the derivative matrices are less sparse but retain sparsity patterns that may be exploited. In our system, we employ cubic B-splines throughout to represent our motion trajectories.

2.5 Dynamics

Given the state of an articulated figure, Newton's laws of motion guide its evolution over time. The laws of motion are mathematically stated as a set of second order ordinary differential equations known as the system's equations of motion.

There are several ways to derive the equations of motion of an articulated system, including the Lagrangian formulation, Newton-Euler formulation, and Kane's method. The derivations can be found in literature for robotics and multibody mechanics. In principle all approaches to stating the equations of motion are equivalent; after all, they describe the same physical model.

The equations of motion relate two dependent quantities: accelerations and forces — forces accelerate the bodies in a deterministic manner, and conversely, accelerations of the bodies must be driven by some set of consistent forces. The former relation is commonly known as *forward dynamics*, where given a set of forces the accelerations are computed. The latter relation is known as *inverse dynamics*, where given accelerations a set of consistent forces is derived. Without reference to any particular formulation, we may state them in the following generic form:

$$\tau = invdyn(q, \dot{q}, \ddot{q}, ext) \tag{2.8}$$

$$\ddot{q} = fwddyn(q, \dot{q}, \tau, ext) \tag{2.9}$$

The equations may then be paraphrased in implicit form:

$$\begin{aligned} dyn(q, \dot{q}, \ddot{q}, \tau, ext) &= 0 \qquad (2.10) \\ &= invdyn(q, \dot{q}, \ddot{q}, ext) - \tau \\ &= fwddyn(q, \dot{q}, \tau, ext) - \ddot{q} \end{aligned}$$

Depending on a given problem and its unknowns, any one of the above three forms may be appropriate. In particular, in physically based optimization of articulated characters, three different formulations arise from these equations of motion.

2.6 Nonlinear optimization

The canonical form of an optimization problem may be written as:

$$\min_{\mathbf{x}\in\Re^n} f(\mathbf{x}) \in \Re \text{ subject to } C_j(\mathbf{x}) = 0 \ , \ j = 1,...m \tag{2.11}$$

The scalar function f is known as the ***objective function*** and the C_j's are known as the ***constraints***. A solution $\mathbf{x}^*$ satisfying all C_j's is said to be ***feasible***, and if it further minimizes the value of f, it is said to be ***optimal***.

Any of a number of numerical gradient-based solvers may used to solve (2.11). A popular choice is the sequential quadratic programming (SQP) method, which will be further described in chapter 6 as a model for our computational analysis.

At every iteration, the solver requires the values and first-derivatives of f and the C_j's, evaluated at a given $\mathbf{x}$. The first-derivatives of all constraints, organized into a $m \times n$ matrix is known as the ***Jacobian*** of the constraints. For some solution methods, such as the SQP model we will use later, the second-derivative of f, an $n \times n$ matrix known as the ***Hessian***, is also required.

2.6.1 Computational overheads

Numerical optimization is an iterative process. At every iteration, two steps dominate the computational effort. The first involves providing the solver with the first derivatives of the objective function and the constraints. The second involves solving a set of linear equations to obtain a descent direction. The asymptotic per-iteration order of complexity of a nonlinear optimization process is thus the sum of these two steps. An analysis of both steps is important — though computing derivatives of nonlinear functions is expensive, the overhead of the optimization algorithm becomes significant as the number of variables becomes large. An analysis of each step for our approach will be provided in the chapters to follow.

2.6.2 Optimization Criteria

The objective function evaluates the quality of a motion given an instance of the variables representing the motion. Mathematically, the criteria for a function to qualify as a candidate objective function is modest:

$$h(\mathbf{x}) : \Re^n \rightarrow \Re \ , \ \ \exists f_l \in \Re \ \text{ such that } \ h(\mathbf{x}) \geq f_l \ \ \forall \mathbf{x} \in \Re^n \tag{2.12}$$

that is, the objective function h is a scalar function (usually non-negative) and is bounded from below. For variational problems, a metric which evaluates a segment of the variational search space is usually written as an integral over the interval:

$$h(\mathbf{x}) = \int_{t=t_s}^{t_f} h(t, \mathbf{x}) \ dt \tag{2.13}$$

where t is the variational parameterization of the search space, usually time, and $[t_s, t_f]$ is the interval where the metric is applied. Many of these integrals are not easily expressible in closed form, thus a numerical quadrature is used to approximate the integral:

$$h(\mathbf{x}) = \sum_{k=1}^{K} w_k h(t_k, \mathbf{x}) \ \ , \ \ t_k \in [t_s, t_f] \tag{2.14}$$

where t_k and w_k are the samples and the weights of the quadrature respectively. Depending on the nature of h, any one of several quadrature methods [58], such as trapezoidal or Gaussian, may be used.

Several metrics have been proposed in the past in the context of motion optimization. One traditional approach is to use the sum-squared-torques integral to produce a motion that approximately minimizes energy expenditure:

$$h(x) = \int_{t=t_s}^{t_f} \tau^2(x, t) \ dt \tag{2.15}$$

This function is expensive because computing its gradient requires $O(D^2)$ work [43, 48].

An objective function that we have used to synthesize the examples in this thesis (See Chapter 7) is to minimize the integral of sum-squared weighted joint accelerations:

$$h(x) = \int_{t=t_s}^{t_f} \left(\sum_{i=1}^{D} w_i \ddot{q}_i^2(x, t) \right) \ dt \tag{2.16}$$

where w_i is aggregate mass subtended at joint i with respect to the effective root. For example, the weight for the left-knee during a left-legged support is the entire body mass minus the left lower-leg. Parameters $\ddot{q}_i$ do not include translational or rotational acceleration of the character root. Note that the analytical Hessian for this objective function is constant, symmetric, positive definite, and band-diagonal.

Where a reference motion $q_R(t)$ is available, a simple objective function with low cost is to simply minimize the distance from the reference motion:

$$h(x) = \int_{t=t_s}^{t_f} (q(x,t) - q_R(x,t))^2 \, dt \tag{2.17}$$

This objective function is similar to the one used in Gleicher [18].

Other objective functions include an integral of sum-squared contact forces:

$$h(x) = \int_{t=t_s}^{t_f} f_c^2(t) \, dt \tag{2.18}$$

The Jacobian of this function is computable in linear-time (See Chapter 5). Gaits generated using this function have a certain 'tip-toe' quality to them, as the function minimizes the amount of reaction forces derived from the contacts.

Minimizing contact jerks (derivative of force) can be achieved by taking the squares of the first-order velocities of contact forces at each key-frame:

$$h(x) = \sum_{i=1}^{m-1} (f_c(t_i) - f_c(t_{i+1}))^2 \tag{2.19}$$

2.7 Trajectory optimization formulations

Let the set of variables in a trajectory optimization problem be denoted by $\mathbf{x} \in \Re^n$. When trajectories are unknown, $\mathbf{x}$ consists of control points representing the generalized coordinate and/or generalized force trajectories. The set of unknowns placed in $\mathbf{x}$ differs with approaches to the trajectory optimization problem, each of which then calls for selecting one of the three forms of the equations of motion (Equations 2.8–2.10).

The three different approaches to trajectory optimization, stated using their most commonly referred to names, are spacetime constraints (SC), optimal control (OC), and physically based motion optimization (MO).

2.7.1 Spacetime constraints (SC)

Introduced by Witkin and Kass [73], a method dubbed "spacetime constraints" casts motion synthesis as a trajectory optimization problem where both forces and motion trajectories are posed as unknowns. Forces consist of time-varying torques produced by muscles simulated with springs at the joints, and time-varying external forces whenever the character is in contact with the environment.

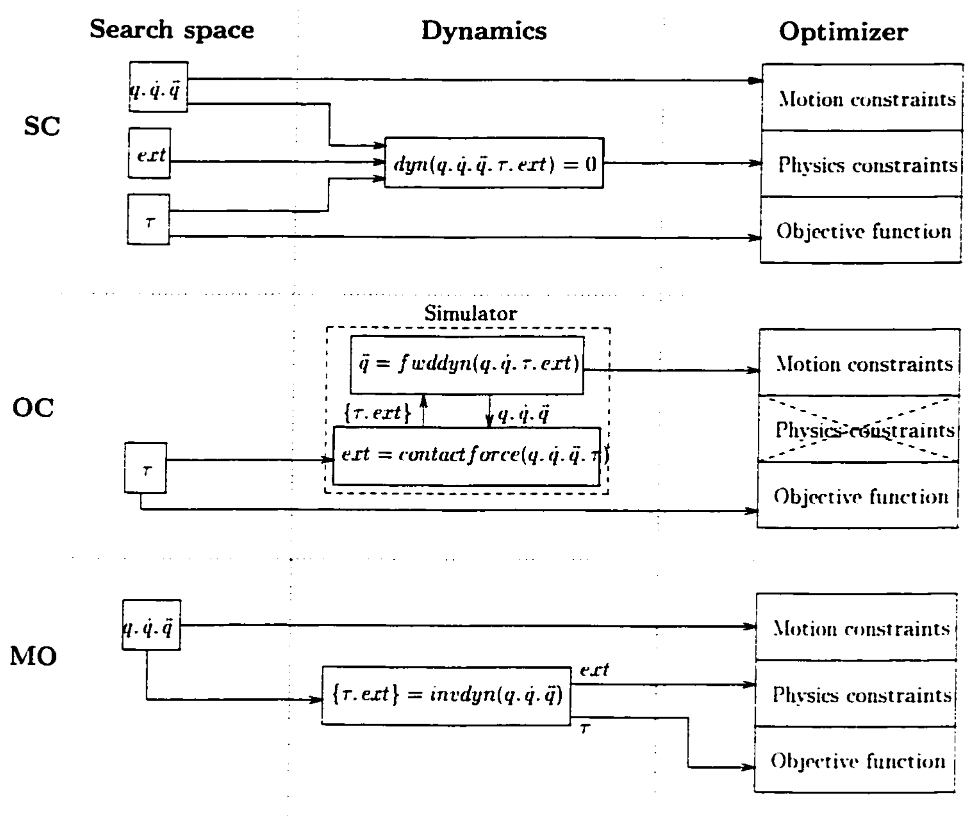

Figure 2.1: Block diagram of the three trajectory optimization set ups. (Top) Spacetime constraints; (Middle) Optimal control; (Bottom) Physically based motion optimization

The implicit form of the equations of motion (Equation 2.10) is then imposed as constraints to enforce physical correctness at a finite set of instances across the motion. In addition, motion constraints, such as initial pose and final pose are introduced to form a boundary-value constrained optimization problem. An objective function which minimizes expenditure of muscular forces is used such that the solver seeks the physically correct solution that at the same time minimizes energy expenditure.

Figure 2.1 (Top) shows the diagrammatic set up of the spacetime constraints method. The diagram relates the main parts of the system — the search variables, dynamics equations, physics constraints, motion constraints, and the objective function.

2.7.2 Optimal control (OC)

Optimal control is the classic methodology in robotics and engineering for solving variational problems involving time-varying control functions. Forces, generalized as "control signals", are of primary importance, because the resulting control signals are meant to be applied to actual mechanisms. The aim of the optimization problem is to seek a suitable set of control signals which is minimal in some pre-defined sense.

The search space in optimal control consists of the control signals, $\tau(t)$: neither motion trajectories nor external forces are explicitly represented. The equations of motion are then used in forward mode (Equation 2.9) — given an initial motion state, a trial control signal $\tau(t)$ is applied and the derivatives of the motion integrated to construct the motion trajectories.

Figure 2.1 (Middle) shows a typical optimal control set up. Central to its formulation is a simulator which performs the integration required to reconstruct the motion trajectories. No physics constraints are required because physics is already an integral part of the simulator.

2.7.3 Physically based motion optimization (MO)

Kinematic based motion optimization is an active research area in animation, much due to the advent of sophisticated motion capture devices. Physically based motion optimization is an extension of such systems to account for physical validity or energy optimality.

In animation, motion is of primary concern while computation of dynamics serves as a means to ensure plausibility and realism. When motion but not force trajectories are represented, forces on the sytem may be derived using inverse dynamics (Equation 2.8). Forces may then be constrained in a way such that motion and forces are consistent with each other, thus assuring physical validity. Joint torques may further be part of the objective function to produce motions that require minimal energy expenditures.

Figure 2.1 (Bottom) illustrates the set up of a typical system. Forces computed using inverse dynamics are discriminated into forces due to muscle actuations, τ, and forces due to the environment, ext. Physical validity is enforced by imposing constraints on ext. This framework is the one adopted in this present proposal, and details will be presented in the chapters to follow.

2.7.4 Enforcing physics over time

In both SC and MO, a finite number of instantaneous constraints are placed over the motion to constrain physics correctness. To avoid over-constraining the problem, the number of constraints m must be less than the number of variables n such that the Jacobian of the constraints is a rectangular matrix, wider than it is tall. For the purpose of this document, we shall assume that given a motion trajectory represented by K basis functions (per degree of freedom), K instantaneous physics constraints are placed over the motion.

2.7.5 Comparison of approaches

Figure 2.7.5 tabulates the main features in which the three approaches differ with respect to the following:

Search space. For consistency with the OC approach, the degrees of freedom D refers to the actuated degrees of freedom of the articulated character, hence its total degrees of freedom inclusive of its global position and orientation is $D+6$. In SC, joint torques and normal forces at each contact point are represented explicitly. Assuming all trajectories use the same discretization, and that C contact points are maintained throughout the motion, a total of $K(2D+C+6)$ control points is required in the SC approach. In the OC approach, only joint torques are represented, hence its search space is most compact and consists of simply KD control points. In the MO approach, the search space consists of the motion curves of all degrees of freedom, hence $K(D+6)$ control points is required.

Physics constraints. In the OC approach, physics constraints are not required. Instead, motion is generated via physically based forward simulation, which also requires a non-trivial procedure to compute contact forces. In SC, $D+6$ Newtonian constraints are required at each frame where correct physics is enforced, resulting in a quadratic size Jacobian with quadratic number of non-zero entries. The optimal time to compute the Jacobian is therefore quadratic in the degrees of freedom. As it turns out, and is pointed out by several authors such as [43, 61], this quadratic density of the physics Jacobian also determines the quadratic lower-bound complexity of the SC approach. In MO, a constant number of constraints is required per physics-enforced frame, its Jacobian is therefore linear in the degrees of freedom and, as we will later show, can be computed optimally in linear time.

Kinematic constraints. Kinematic constraints are functions defined over the motion trajectories $q(t)$. We assume K such constraints distributed over the K basis functions. In OC, $q(t)$ is generated by integration, hence any function at t depends on all basis control points affecting $q(t)$ in the interval $[0, t]$. The Jacobian of the kinematic constraints is therefore quadratically dense. In SC and MO, kinematic constraints are easier to evaluate since $q(t)$ is represented explicitly. When each basis function have only local support, each kinematic constraint depends on $O(D)$ number of control points. In particular, for cubic B-splines, each kinematic constraint depends on at most $4D$ control points. The number of non-zeros in the Jacobian of K constraints therefore has $O(KD)$ non-zero entries.

	SC	OC	MO
Search space:			
Per basis function	$\tau \in \Re^D$ $q \in \Re^{D+6}$ $ext \in \Re^C$ $total = \Re^{2D+C+6}$	$\tau \in \Re^D$	$q \in \Re^{D+6}$
K basis functions	$K \times (2D+C+6)$	$K \times D$	$K \times (D+6)$
Physics constraints:			
Eqns of motion	$dyn(q, \tau, ext) = 0$	$q = \int fwddyn(\tau, ext, q_0)$	$\{\tau, ext\} = invdyn(q)$
Constraints per frame	$D+6$ constraint equations on the degrees of freedom	Physics is simulated; No constraints required	≤ 6 constraint equations on ext.
Jacobian per frame	$(D+6) \times (2D+C+6)$, Dense	Not applicable	$6 \times D$. Dense
Jacobian for K frames	$K(D+6) \times K(2D+C+6)$, $O(KD^2)$ non-zeros	Not applicable	$6K \times K(D+6)$, $O(KD)$ non-zeros
Contact forces, ext	ext is part of search space	Contact force computations required	From inverse dynamics
Kinematic constraints:			
Motion constraints	Direct constraints on search variables.	Indirect: forward integration required.	Direct constraints on search variables
Jacobian sparsity for K frames	Block diagonal; maximum $O(D)$ non-zeros per row	Lower triangular matrix, $O(KD^2)$ non-zero entries	Block diagonal, maximum $O(D)$ non-zeros per row
Objective functions:			
$\sum_{i=1}^{D} \tau_i^2$	$O(D)$ Evaluation, Jacobian, and Hessian		$O(D)$ Evaluation, $O(D^2)$ Jacobian, Numerical Hessian

Figure 2.2: Comparison of features in the three trajectory optimization approaches.

Chapter 3

Literature

This chapter reviews the background literature of this thesis in the areas of motor control, dynamics simulation, and optimization based animation systems, with a focus on their computational models related to the framework of trajectory optimization.

3.1 Motor Control

Studies and experiments in motor control and biomechanics provide insight into the basis of human movements. As motion of human subjects is analyzed, computational models are often proposed to suitably fit these experimental results. These models provide the basis for predicting future behavior, and are frequently adapted for use in the synthesis of human movements.

Empirical studies in motor control show that out of the infinite number of ways that one could make a particular movement, we tend to pick one that is smooth [51]. For example, experiments in human point-to-point arm movements [6] show that when a subject is asked to reach towards a stationary target, the subjects tend to move their hand along a straight path with a single-peaked, bell-shaped velocity profile [1, 66, 67]. Studies show that this feature is independent of the hand's initial and final position, or perturbations in body position or orientation [24]. That this feature is invariant in typical goal-oriented motion is well-known. And it has been suggested that such invariance in observed motion provides hints as to the internal representation of movements in the central nervous system [6].

A number of theories have been proposed to model this phenomenon. The predominant thinking is that the smooth, stereotyped trajectories made by our motor system are specially chosen to optimize a cost that is integrated over the movement. Some of the cost functions found in present literature are as follows.

Minimum-jerk (Cartesian-space). The minimum-jerk model was originally proposed by Hogan [30] for a one-joint movements, and subsequently refined by Flash and Hogan [66] for multi-joint movements. The model states that the cost to be minimized is the first derivative of the Cartesian hand acceleration, or "jerk". For planar movements, the cost function is:

$$C_J = \frac{1}{2}\int_0^T (\frac{d^3x}{dt^3})^2 + (\frac{d^3y}{dt^3})^2 \ dt$$

where T is the duration of the movement and (x, y) is the hand's position at time t. Note that C_J is based entirely on the kinematics of the hand.

The trajectories predicted by the minimum-jerk model are straight lines with bell-shaped velocity profiles. in agreement with trajectories observed in experiments. Further. since dynamics are not accounted for, the trajectories are invariant under rotation and translation of start and end points of the movement so long as the start and end points are within the workspace of the hand. This latter point is also in agreement with empirical observations.

The minimum-jerk model predicts the trajectory of the hand in Cartesian space. The translation of the hand trajectory to joint motion or motor torques must be resolved by a separate control process. Examples of such processes include inverse modeling [50. 37], supervised learning [33]. feedback-error-learning [34], or virtual trajectory control [30]. Feedback mechanisms are employed in the latter two models such that errors in the actual trajectory translate into changes in the motor command. With the two-step plan and execute model. the minimum-jerk trajectory only serves as a reference trajectory, and adherence to this trajectory is subject to parameters of the control phase. such as tolerance of deviations or stiffness. The actual hand trajectory produced as a result of the control process is often curved.

The advantage of this two-step process is the distinction of a planning phase. which encompasses visual and cognition elements, and an execution phase, which determines the control trajectories and muscle activations. The two process are intricately dependent. leading to the feedback loop incorporated into many of the control models.

Although the minimum-jerk model suggests planning in Cartesian space. similar paths can be obtained via interpolation in joint space. In [31], a scheme was proposed to represent the start and end points as joint angles obtained through inverse kinematics. The joint trajectory is subsequently obtained through interpolation of these angles. The advantage of this approach is that the control process is simplified. requiring relatively simple proportional-derivative controllers that follow the prescribed joint trajectories.

Minimum-torque-change (Joint-space). An alternative approach to trajectory formation. the minimum-torque-change model. also known as the maximum-efficiency model. has been proposed [67]. It differs from the minimum jerk model in that the reference trajectories are dependent on the dynamics of the arm. The cost function is of the following form:

$$C_T = \frac{1}{2}\int_0^T (\frac{d\tau}{dt})^2 \ dt$$

where τ is the joint torque vector at time t. C_J and C_T are closely related because acceleration is proportional to torque at zero speed [67].

Minimum-torque-change, like minimum-jerk, predicts bell-shaped hand velocity profiles. both in agreement with experimental observations [67]. However, unlike the straight reference trajectory in the minimum-jerk model, the trajectory prescribed by the minimum-torque-change model is curved. This suggests that whereas the curved path in the minimum-jerk model is a result of the control process. this curvature is a direct result of the minimum-torque-change model. Also. since the primary output of the minimum-torque-change model is torque trajectories. the two-step distinction of the minimum-jerk model is eliminated. and a separate control resolution process is not necessary.

Minimum-variance (Cartesian-space). The minimum-variance cost function [24] takes into account the noisy and imprecise nature of neural control signals. It hypothesizes that goal-directed movements are engineered to maximize precision of the task, and that in presence of noise. the precision of a task may be only weakly dependent on the finer underlying motor control signals. The cost function that is minimized is the statistical variance of the eye or arm position across repeated movements summed over the period.

It was shown via simulations that the minimum-variance model also predicts the smoothness of movements; the bell-shaped velocity profile of the hand trajectory is preserved. The minimum-variance theory suggests that this smoothness is a by-product of a more fundamental goal of the motor system — one which is task and position based, rather than the more abstract notion of torque derivatives or acceleration derivatives.

There appears to be no principled explanation why the central nervous system should have evolved to optimize the quantities stated in each of these models. other than that these models predict smooth trajectories. The advantage of smoothness of movement itself remains unexplained. Furthermore, how the central nervous system could estimate complex quantities. such as jerk or torque change, and then integrate them over the duration of a trajectory is also unknown. Thus. the different models could only serve to fit typical behavior seen in empirical data of natural movements. but not to explain them.

At this time. the debate over the various cost functions remain unresolved. especially between kinematic and dynamic based cost functions [75]. Recent studies suggest that for arm movements. humans are unlikely to embark upon complex computations or estimations of dynamics quantities and that planning is essentially performed in kinematic space with Cartesian trajectories in accordance with the minimum-jerk model [76]. Evidence from perturbed visual feedback studies [76. 15] and force field studies [63, 38] also support the two-phase plan and execute strategy.

On the other hand, studies of more complex movements. for instance arm movements in presence of obstacles. suggests that knowledge of the dynamics of the arm is inherent in planning. In one study [75]. it was shown that in arm movements. subjects tend to select their movement paths so as to ensure that their closest point of approach to the obstacle is on an axis where the arm is most inertially stable. In another study [62]. interactions between the subject and its environment. such as movement constraints, is shown to have a significant effect on movement kinematics not predicted by kinematic based models.

Perhaps a more robust model lies in some middle ground. The minimum-jerk model and the minimum-torque-change model represent two extremes in evaluating a candidate trajectory. The former focuses entirely on its hand trajectory, deferring concerns of joints and movement dynamics. while the latter focuses entirely on the dynamics of movement. the hand trajectory being a by-product. The disparity of the two models is also apparent in terms of computational expense — the linear cartesian trajectory in the minimum-jerk model is easy to compute. whereas the derivative of joint torques require the invocation of a full multibody dynamics differential engine.

This treatise, in a way, bridges this disparity of cost functions. In the context of motion synthesis. purely kinematic methods produce motion which is unrealistic due to physical incorrectness. while methods which demand physical optimality are not only expensive but overly typify the style of the motion. The exposition of the middle ground, physical validity, is found to be a viable means to produce physically realistic animations. and may well contribute to the repertoire of cost functions in the area of biomechanic and motor control studies.

3.2 Dynamic Simulation and Control Systems

Dynamic simulation of animated characters is inspired by robotics research. The seminal work of Marc Raibert [59] showed in the late 1980's that robotic running could be accomplished using a few simple control laws which were assumed to be decoupled for design purposes. Raibert showed how to calculate, on an event-by-event basis, the control effort required for the next step of a run based on the results of the previous step, using simple physics-based rules.

Recently, Honda announced the success of a secret project to build a humanoid robot [23, 32]. The Honda robot is capable of walking, balance, and climbing stairs. Controllers for these activities are constructed around trajectory recordings and executed with moderately high stiffness. Real-time sensors provide feedback to suitably modulate the motion so as to maintain balance according to the Zero Moment Point (ZMP) control law [71]. The result is a remarkably life-like and convincing demonstration of the state-of-the-art in robotic locomotion.

The design of control systems for characters with interesting complexity has proved difficult. Designing control systems manually is labor intensive and requires that the animator not only have extensive knowledge about the behavior but also the ability to encode this knowledge in the form of a control algorithm. Although controllers have been successfully designed for various human motions such as walking, running, leaping, diving, vaulting, cycling, and gymnastics etc. [3, 72, 29, 39, 79, 77, 28, 25, 26, 60, 78, 8], characters equipped with a wide-ranging repertoire of motor skills currently remain an unachieved goal. To this end, Faloutsos et al. [13] proposed an approach to harness the collective effort of controller-based animators by unifying the framework of controller designs, where controllers designed to perform specific tasks are identified with a set of "pre-conditions" specifying the stipulated conditions under which the controller may be triggered. The hope is for the common framework to facilitate the exchange and composition of control algorithms.

In view of the effort required in designing control systems, researchers have looked into ways to automatically generate motion controllers [52, 70, 69, 2]. However, the algorithms, based mainly on high-dimensional state space searches, have been demonstrated mostly for simple articulations in 2D, and the extension of these algorithms to complex humanlike 3D characters appears to be computationally formidable.

Generating appealing motion is the central problem in animation. Dynamic simulation guarantees physical realism and offers a potential solution to this problem. Controller based approaches reveal the underlying physics, planning, and control and they therefore serve as a basis for more general solutions. It is less clear how to extend the role of dynamics simulation to traditional animation. Aside from passive body motions, dynamic simulation of self-actuated characters has not yet been adopted as a mainstream animation tool. The main issues are control over the simulation process and the tedium of specifying intended movements as control algorithms. Some of these issues are addressed for passive body motions [56]; it remains to be seen if similar strategies can be formulated for self-actuated characters.

There are fundamental differences between the requirements of computer animation and robotics-style dynamic simulation. In robotics, forces and torques, generalized as "control signals", are of primary concern and motion is a by-product. The fundamental purpose of dynamic simulations is to put to trial a set of forces actuated by a prototype control system. Since such control signals are eventually meant to be applied to actual robotic mechanisms, the quality of the control signals, such as smoothness and amplitude, must be acceptable and within physical limits of the robot.

On the other hand, motion is of primary concern in animation. Consideration of dynamics serves only as a means to ensure plausibility and realism: forces and joint torques are not usually presented in the final product. As such, concerns about the quality of the control signals are secondary. But when borrowing simulation as a means to animate characters, the quality of control signals cannot be compromised since the simulator is supposed to faithfully respond to its controller (doing otherwise would compromise the physical correctness of the animation) and bad input signals translate to failed simulations, incompletion of the intended task, or simply bad animations, all despite being perfectly physically correct!

The central idea in this thesis is to exploit the relaxed requirement of animation by placing motion, not control signals, as the primary primitive. In treating motion as the primitive representation, intended outcomes of the motion can be effected directly and concisely. By introducing only sufficient dynamics to ensure physical validity, a bulk of computation associated with control signals may be disregarded, and an overall savings in computational cost may be reaped. This approach requires a change in paradigm and will be elaborated on in subsequent chapters.

3.3 Motion Editing

Motion editing techniques are useful for making corrections to a motion or for obtaining new motion from existing examples. This section reviews the existing techniques used for transforming motion data. A motion may be transformed in the frequency domain, where broad changes to a motion are possible using frequency filters, or in the spatial domain where localized changes to a motion are allowed.

3.3.1 Frequency domain

In signal processing parlance applied to motion, low frequencies contain general, gross motion patterns, whereas high frequencies contain detail, subtleties, and in the case of digitized motion, most of the noise. Motion capture capture systems provide one direct example of how changes to a motion in its frequency domain can be useful: data acquired in motion capture systems is often too noisy to be used directly. As such, most motion capture systems provide nonlinear impulse-noise removal filters as well as linear smoothing filters.

Among a library of other techniques, Bruderlin and Williams [9] applied the principles of multiresolution filtering and analysis to change the quality of a motion. Each joint motion is treated as a single dimensional signal and is transformed into the frequency domain using bandpass filters. As adjustments are made in each frequency band, effects such as smoothing of movements and increased twitching are demonstrated.

Unuma et al. [68] used Fourier analysis to interpolate and extrapolate motion data. Based on a frequency analysis of the joint angles, a basic 'walking' factor and a 'qualitative' factor like 'brisk' or 'fast' are extracted. These factors are then used to generate new movements by blending the motion in the frequency domain such that a walk can be changed continuously from normal to brisk walking.

Though abstracting the qualities of the motion in terms of its frequency bands is useful, the difficulty with modifying frequency elements of a motion is that the reciprocal changes in its spatial domain may violate important visual aspects. For instance, joint limits or non-penetration with the ground may be violated as a result of frequency changes. Another difficulty is that changes in frequency domain affect the entire motion. Such changes are hard to control and their effects are not always intuitive or desirable. For instance suppressing high frequency elements for the purpose of removing noise has its pitfalls because significant events in the motion, such as impulses, changes in contact configurations, or other intentionally fast movements also result in high frequency components. It is also intuitively wrong to treat the motion signals corresponding to joint degrees of freedom as independent entities. For instance, a rotation joint has three degrees of freedom and therefore three motion curves. However, in any parameterization of the rotations, the three degrees of freedom are inter-related. An operation such as filtering a particular frequency of a specific degree of freedom does not make much sense. Moreover, tasks are usually achieved by the combined effort of a number of degrees of freedom. For instance, a leg swing motion is often accompanied by a knee retraction. This implies that in the context of the overall motion, the degrees of freedom are not independent and therefore should not be processed as such.

3.3.2 Spatial filtering

Online applications impose a real-time challenge on motion editing techniques. Iterative optimization methods may not produce satisfactory solutions within interactive rates. And in the case of applications such as virtual characters or computer puppetry, lookahead data for analyzing a motion's context may be limited.

In the language of signal processing, motion streams that may not fulfill certain pre-ascertained conditions may be subjected to various correction filters. The simplest of such filters are kinematic filters, which for instance adjust the pose of a character on a frame by frame basis to maintain constraints such as keeping the stance foot planted on the ground, or ensuring that the character's center of mass projects to a point within its base of support [36]. Dynamic filters on the other hand can be used to maintain physical constraints, such as keeping torques withing limits, and keeping contact forces within the friction cone [80, 55].

3.3.3 Spatial domain

Most commercial character animation systems have the capability to adjust key frames while maintaining a number of contraints, such as foot position and balance. The process of maintaining geometric constraints while interactively changing a character's pose is often known as inverse kinematics [81]. The extension of such systems that allow instantaneous constrained changes to changes within the context of the motion is the subject of several motion editing approaches.

Bruderlin and Williams [9] described motion blending techniques to build complex motions from simple ones. The concept of a displacement map was also introduced to allow local changes to a motion while preserving its overall characteristics. The idea of the displacement map, which essentially represents the changes to a motion rather than the changed motion, turned out to be an important idea which was adopted in several subsequent motion editing papers.

Witkin and Popović [74] developed a method in which the motion data is warped between keyframe-like constraints. By using displacement maps in conjunction with sparse keyframe constraints, both timing and blending between motion clips can be altered by the animator.

Gleicher presented the motion editing problem as a constrained optimization problem [22]. As in the Spacetime Constraints apprach, the system considers a number of constraints over the entire duration of the motion. However, instead of posing physics laws as constraints and energy minimization as the optimization objective, the system sought to minimize changes to a reference motion subject to a set of geometric constraints. Gleicher demonstrated that with such a set up, real-time interactive performance could be achieved on a variety of motion editing tasks.

Lee and Shin [41] applied multi-level B-splines to solving the same problem by decomposing the motion in time such that each subproblem can be solved very efficiently. Using a divide-and-conquer strategy, the motion editing problem is first solved independently at each frame such that constraints are satisfied instantaneously at all frames. The solutions are then combined by a B-Spline curve fitting algorithm. Since each of the two steps may undo the work of the other, they are interleaved and iterated until the solution converges.

Recently, Gleicher compiled a taxonomy of motion editing techniques based on the differences in the distribution of constraints and the numerical solution methods [20. 21].

The optimization based approaches to motion editing, such as [22. 41] have close relation to the work present in this thesis and will be further discussed in the section 3.5 along with other optimization based animation techniques.

3.4 Motion Retargeting

When prescribing motion obtained from one subject to another. important qualities of the original performance ought to be preserved. This problem is known as motion retargeting.

The main issue in motion retargeting is the anthropometric difference between the source and target subjects. If the subjects have, say, different limb lengths. prescribing the same set of joint angles to both will result in different placement of their extremities. Were the hands placed on a table with feet on the ground in the original motion, the retargeted motion would either have dangling hands or floating feet. Concerns such as stride lengths in locomotion must be accounted for as well. Like motion editing, retargeting techniques need to modify the motion in its entirety since changes to any frame of the motion must be consistent with the continuity of the motion.

Where kinematic integrity of retargeted motion is of primary concern. the problem of motion retargeting may be cast as a constrained optimization problem. Gleicher [19] approached the motion retargeting problem by minimizing an objective function based on first-order differences between the original and retargeted motions, subject to a set of kinematic constraints. The source and target characters have identical structure but different body part lengths. The paper demonstrated the practicality of their technique in adapting motions such as walking, lifting. dancing. and climbing between characters with markedly different body dimensions. Lee and Shin [41] also demonstrated that their motion editing system using multi-level B-Splines system is capable of motion retargeting between characters of different body dimensions. Choi and Ko [10] used closed-loop inverse rate control to produce a retargeted motion online by minimizing end-effector path difference.

Motion retargeting systems need not employ optimization techniques. Using a combination of scaling and inverse kinematics, Baek et. al. [4] applied a posture-based retargeting technique in a VR-based seting and provided qualitative analysis of the retargeted motion.

In real-time performance puppetry for broadcast entertainment, Shin et al. [64] introduced an importance-based approach in which the relative importance of joint angles for freespace movements and end-effector position and orientation were evaluated based on the proximity of objects and on a priori notations in the performer's script. Whereas previously interactions with the environment were often imposed as hard constraints, the importance-based approach allows a compromise between tracking the reference motion and yielding to influences from the environment.

In robotics research, Pollard et al. [54] retargeted motion capture data of human subjects to the upper-body of a humanoid robot. A trajectory tracking control system drives the motion of the robot to match a motion capture sequence that has been appropriately scaled to fit its joint and velocity limits. Issues pertinent to motion retargeting, such as maintainance of multiple contacts, were not accounted for since the robot is rooted at the hips and the hands were not in contact. Had the robot hands been in contact with a static object, the retargeting issue would have been more complicated, perhaps requiring one of the motion retargeting solutions described above.

Though preserving contact constraints is important, retargeting is not solely a kinematic issue. Even when limb lengths are similar and contacts preserved, differences in body mass and inertia may result in physically unrealistic motion.

Hodgins and Pollard [26, 27] adapted a control system designed for one character to another character of different dynamic properties. Naively adapted control systems result in simulations that fail to run, fall short of achieving the intended task, or unable reach a steady state. Just as simple scaling of joint angles does not suffice for creating correct motion, simple scaling of control parameters is not adequate to transform one control system to another. In their two-stage algorithm, the control system parameters are first scaled based on differences in geometry, masses, and inertias. In the second stage, a subset of the control parameters is fine-tuned via a search process based on repeated evaluations of simulation trials. The resulting system is able to adapt control algorithms for running and cycling between characters with significantly different anthropometry.

Generating new animations with existing resources is an important concern in animation. Retargeting motion or adapting control systems provide means to populate virtual worlds with varied characters without devising each motion design from scratch. Previously accepted designs or readily available motions may be retrofited upon new characters as quick ways to obtain new motions.

Retargeting motion differs from adapting control systems in the primitives that are transformed but shares the idea of reusability. Motion retargeting transforms a segment of motion whereas adapting control systems transforms the engine that produces the motion; the applicability of each approach depend on the resources available.

Whereas adapting control systems inherently accounts for dynamic properties of characters, motion retargeting often works with motion whose dynamics are disregarded. Present motion retargeting systems do not account for physically based behavior. This approach allows much flexibility in creating interesting animations, for instance, adapting a striding man to a soda can [19]. It is yet unclear how dynamics can be suitably incorporated into a motion retargeting system.

especially when structural differences in subjects prohibit the performance of a prescribed motion in any physically correct way — an original motion may be compared with a new one via various difference measures, but what does it mean to compare differences in dynamics? How is one motion's dynamics similar to another in terms of physically based measures? Will accounting for dynamic properties in a motion retargeting system provide any real benefit? At this point, these remain open questions.

Optimization approaches to motion editing and retargeting are of much relevance to this thesis and will be discussed in greater detail in the next section within a computational context.

3.5 Optimization based Animation

Optimization is a general tool applicable to many areas where desirability and constraints can be mathematically encoded. In graphics, optimization has been shown to be useful in areas such as image rendering, geometric modeling, and other design and parametric simulation problems [49]. In animation, optimization has applications in motion synthesis, editing, planning, and other physically based techniques that involve solutions to local or global inverse problems. This section surveys works where optimization plays a fundamental role in creating animations.

3.5.1 Optimization of passive body motion

Optimization has been shown useful in constructing physically valid paths of passive bodies. Popović et al. [56] describe a system where the user interactively manipulates the motion of a rigid body while the system computes the new parameters that achieve the desired motion changes. The problem is cast as a second-order initial value problem — the optimization variables are the initial states of the bodies and include physical parameters such as mass, inertia, and elasticities. Given an initial state, a simulator numerically integrates the state derivatives to produce the motion path. In the context of our work, the most relevant advantages of this approach are that the parameter space is compact, and that the motion trajectory need not be explicity represented. Not having to represent a motion trajectory is useful, since representations such as splines inherently limit the frequency content in the resulting motion.

The nature of this work however differs from ours in several ways: Firstly, the method targets passive bodies, whereas we wish to animate self-actuated articulated characters, that is, characters capable of exerting forces or torques. Secondly, the simulator function is highly nonlinear even for single rigid bodies. This is because any instant along the motion path is dependent on the entire trajectory prior to it. For complex articulated bodies, these nonlinearities will be further compounded. As such, the use of simulator-type optimization, a technique closely related to the single-shooting methods in optimal control, is in general not feasible for animating human-like articulations. For these reasons, motion optimization for articulated characters is usually cast as a variational problem.

3.5.2 Optimization with physics

A variational optimization problem is one where the unknowns are functions. In animation, these functions are the motion or force trajectories. Witkin and Kass [73] used constrained optimization to solve a variational optimization problem and coined the term "spacetime constraints". A variety of animations involving a jumping Luxo lamp from simple descriptions including start pose, end pose, and a physically based objective function were demonstrated. The characters described in the paper have few degrees of freedom — a planar particle with two degrees of freedom, and Luxo with four — but the examples serves well as a proof-of-concept that constrained optimization has potential for creating physically plausible motion of self-actuated characters.

In the spacetime constraints formulation, the search space consists of both the joint motion and joint torque trajectories (contact forces are also in the search space). This representation is redundant in the sense that joint motion can be uniquely derived from joint torques and vice versa via the equations of motion. Though not explained in the paper, I suppose there are two reasons for creating a redundant search space: Firstly, if joint torques are represented explicitly, computing sum-squares of torques is easy and the analytical Hessian (second derivative) of this function is a constant matrix. Because their particular choice of the numerical solver requires the analytical Hessian of the objective function (but not the constraints), the Hessian and its trivial inversion are available for free. The second reason is that creating a larger search space allows errors to be better distributed, in a least-squares sense, for over-constrained cases. Hence difficulties in satisfying physics constraints may be compensated by both torques and motion.

Creating a larger variable space has its problems, however. If the optimization complexity is of quadratic order, doubling the search space increases its time and space complexity by a factor of four. The search also has to occur in a larger dimensional space. For simple characters this cost is less significant compared to, say, evaluating partial derivatives of the physics constraints. However, for high dimensional characters, this cost may make the problem less tractable.

Rose et al. [61] described a variant of spacetime constraints that generates short transitions between motion capture segments. The aim is to minimize joint torques over the transition period. They reported that physically based transitions look more natural than transitions interpolated with purely kinematic means. The degrees of freedom in their example is high — a 44 degree-of-freedom human character, and an efficient formulation of Newton-Euler inverse dynamics are described to compute joint torques in linear time, and their partial derivatives in quadratic time.

The paper has several aspects different from the Witkin and Kass's spacetime constraints: Firstly, the torque trajectory is not explicitly represented. This halves the dimensions of the search space, at the expense of a numerical Hessian instead of an analytical one. Secondly, a hybrid inverse kinematics and inverse dynamic optimization is used — contact limbs are positioned kinematically while the rest of the body motion is optimized with a sum-squares-torques objective function. For instance, in a transition that does not change foot contacts, e.g. from standing to swing a racquet, the lower limbs are positioned using inverse kinematics while the upper body motion is optimized. As a result, their nonlinear formulation for minimizing joint-torques is an unconstrained problem. Thirdly, the transitions are kept short, typically 5-10 basis functions per trajectory. Although their problem still has quadratic complexity bounds, these changes to the spacetime formulation made their optimization problem on a full-scale human model tractable.

When the number of basis functions is large (per motion curve) and when basis functions have only local influence, the nonlinear optimization problem posed by the spacetime constraints method may suffer problems due to ill-conditioning. This problem is well addressed by representing the motion using hierarchical bases, such as spline wavelets. Though a change in bases does not improve upon the quadratic order of complexity of the spacetime constraints method (in fact, time complexity to evaluate all partial derivatives increases by a logarithmic factor), it has been shown that the better conditioning provided by the hierarchical bases significantly improves convergence rates [43, 47]. The authors reported that roughly 30 cubic spline bases is considered large. Since we are not addressing problems related to length of motion in this work, we shall circumvent concerns related to ill-conditioning by limiting the length of motion we work with. As is advised by the authors, the length of motion throughout this work will be kept roughly within 30 spline bases per degree of freedom.

3.5.3 Optimization without physics

The ability to edit a pre-existing motion or to apply it to a different model is a necessary complement to motion capture systems. Editing or retargetting a motion while preserving motion constraints such as limb contacts can be formalized as a constrained optimization problem.

Interactivity is an important part of a motion editing system. In order to achieve interactive rates, several authors [18, 41] proposed to abandon the two most compute intensive parts of the spacetime constraints problem — computing joint torques and enforcing physical validity. For lack of a torque minimal objective, the objective function is reformulated with a proximity function that minimizes changes to a reference motion.

Since physical equations of motion are not considered, these techniques apply mostly to editing motion capture data, where any quality of physical plausibility must be inherited from the reference motion. Numerically, the objective functions are simple to evaluate and differentiate. In all cases, the analytical Hessian of the objective function is available for free. Constraints are kinematic constraints, such as limb contacts, and may be evaluated and differentiated very efficiently. It is clear that exceptionally good performance can be obtained when physics or energy optimality is left out of the optimization problem.

3.5.4 Optimization with approximate physics

One way to cope with complex characters is to abstract their articulation. Liu and Cohen [44] showed that in diving, the articulated character and its dynamics equations can be simplified by considering the various phases of the diving motion. Popović and Witkin [57] showed that optimization can be performed in reasonable time when a high dimensional humanlike character is simplified to degrees of freedom that are most essential to the task. Motion editing is performed by spacetime optimization on the reduced character, and the resulting motion is mapped back to the full character. This final mapping is under-determined as it maps a smaller set of data to a larger one and is guided by minimizing changes from a reference motion.

Simplifying an articulation reduces the computation effort by a constant factor. Popović and Witkin reported that a typical simplification factor is between 2 to 5. In terms of order of computation, simplifying a character does not change the quadratic complexity of the spacetime formulation. However, by drastically reducing the degrees of freedom of the dynamic model they showed that optimization times in order of minutes is possible.

Deciding which degrees of freedom are essential is not always trivial. In the paper, character simplification is done manually by a variety of means, such as fusing joints, combining limbs, or discarding degrees of freedom. Supposedly, this must be done with the task in mind. Given a reference motion, it might be possible to carry this out automatically, but to my knowledge it has not been done.

In the case of motion synthesis without reference data, the method has limited appeal, since without a reference motion it is unclear how the inverse mapping should be disambiguated. An optimization step to carry out this mapping may be as difficult as the original problem. Without reference motion, it may also be difficult to prevent the optimizer from straying to a solution that capitalizes on the reduced model's structural "handicap".

More recently, Liu and Popović [42] showed that highly dynamic motion may be synthesized when guided by momentum patterns derived from biomechanics studies: when a character is in flight, its angular momentum remains constant while its linear momentum decreases linearly; when the character is in contact, its linear and angular momenta follow a simple, smooth characteristic transition from one constant to another. By constraining the movements of the character to these patterns, they showed that given a small set of initial key frames, motions such as jumping, running, and handspring may be synthesized.

That motion may be synthesized from simple momentum patterns is an interesting observation. However, the laws of motion are dictated by second-order differential equations involving accelerations, forces, and torques. Without computing forces, dynamic constraints between the character and the environment, such as friction directions and limits, are difficult to enforce. Other motions where changes in angular momentum are less regular, such as a falling cat turning itself over (net momentum is zero), a kicking motion that has contact with the ground throughout the motion (net momentum is irregular), may be hard to synthesizing using momentum patterns.

3.5.5 Optimization without contact physics

A number of authors [61, 48] proposed to simplify the motion optimization problem by ignoring contact forces and friction. When no frictional forces are allowed, the ground reaction force is strictly normal to the contact plane — torques about the contact support are not permitted. This approach is equivalent to setting the friction coefficient of the Coloumb friction model to zero. Lo and Metaxas [48] demonstrated that examples of such motion include weight-lifting and push-ups. The spacetime constraints problem thus reduces to a torque-minimization problem and no explicit physics constraints are required.

Accurate modelling of contact forces is crucial, however, in motion where dynamics plays a dominant role and where reaction forces from the environment empower the figure to enact the dynamics. These include motion such as running, jumping, and all motion where frictional forces play a major part in either retarding or propelling the character's movements. In order to synthesize such motion, ground reaction forces must be properly accounted for.

3.5.6 The use of displacement maps

The use of displacement maps [9, 35] is a recurring theme in motion editing. A displacement map represents changes to an underlying motion. The map may use a representation different from the original motion, for instance, the motion may be closely spaced key frames while the displacements are B-splines. Gleicher used a displacement map to restrict frequency changes to a motion [18, 19], where usually only low frequency changes are permitted, the rationale being that high frequency information contains details characteristic of the motion and ought to be preserved. Lee combined quaternion algebra with exponential maps to achieve a similar effect [41].

Working with displacement maps is desirable in several ways. Not only does it allow controlled changes in content, in terms of nonlinear optimization it also helps in setting up a well-scaled nonlinear problem as displacements have a neutral value of zero. Further, an objective function that minimizes sum of changes to the underlying motion can be formulated in closed form. The Hessian of this function is also trivially available and its inverse is obtained for free. Unfortunately, displacement maps are not quite applicable where reference motion is not available. In particular, for motion synthesis, displacement maps do not provide much help.

In anticipation that our method has applications in motion editing systems, Chapter 6 shows that linear-time per iteration is achieved when displacement maps are used in conjunction with our method.

3.5.7 Summary of approaches

Figure (3.5.7) categorizes a sampling of the present body of work in motion optimization with respect to its ability to handle physics, and the cost of differentiating its functions.

The upper-left quadrant are approaches which do not incorporate physics into the optimization. We know that exceptionally good performance can be obtained when physics is left out of the optimization problem. Many of these techniques are therefore directly applicable in an interactive setting.

On the other diagonal end are methods with physics constraints and energy optimality. These methods have orders of complexities quadratic in the degrees of freedom. As such, the methods are not expected to scale well with the complexity of the character.

At the top-right quadrant are methods where physics is approximated. Simplifying a character reduces the order of complexities by a constant factor, but does not assure physically correct results.

The present work fills the void where correct physics is accounted for, and where derivatives can be efficiently computed in linear time.

	Linear time metrics/constraints	Quadratic time metrics/constraints
Approx. or no physics	• Witkin/Popovic, SG95, "Motion warping" [74] • Gleicher, SG97, "Motion editing with spacetime" [18] • Gleicher, SG98, "Retargetting motion" [19] • Lee/Shin, SG99, "Hierarchical motion editing" [41] • Liu/Karen, SG02, "Complex character motion from simple animations" [42]	• Popovic/Witkin, SG99, "Physically-based motion transform" [57]
Full physics	• This present work.	• Witkin/Kass, SG88, "Spacetime constraints" [73] • Brotman/Netravali, SG88, "Motion interp. by optimal control" [7] • Cohen, SG92, "Interactive spacetime control" [11] • Liu/Gortler/Cohen, SG94, "Hierarchical spacetime control" [47] • Rose/Guenter/Bodenheimer/Cohen, SG96 "Efficient motion transitions" [61] • Lo/Metaxas, CA99, "Recursive dyn. & optimal control" [48]

Figure 3.1: Classification of motion optimization papers with respect to physics consideration and cost of metric.

Chapter 4

Physical Validity

This chapter describes an approach for enforcing physics in the framework of physically based motion optimization. Figure 2.1 (Bottom) illustrates the set up for this scheme. The search space consists of control points of the motion trajectories. A finite set of keyframes is chosen (section 2.7.4) and physical validity is imposed at these keyframes. At each keyframe t. the instantaneous trajectory — $q(t)$. $\dot{q}(t)$, and $\ddot{q}(t)$ — is obtained by evaluating the positions. velocities. and accelerations of the motion curves at t. Inverse dynamics is then used to compute the set of forces consistent with this instantaneous trajectory. The set of forces is discriminated into joint forces and external forces. Validity of a given set of external forces is determined by subjecting it to the feasible ranges of a Coloumb contact model. At all keyframes, constraint functions are set up to ensure that external forces do not exceed ranges permitted by the contact model. The following describes the details of this approach.

4.1 Newtonian equations of motion

In a trajectory optimization system, physics constraints are used to enforce Newton's laws of motion. The laws of motion are mathematically stated as a set of equations known as the equations of motion. The inputs to the equations are the positions, velocities, and accelerations of the degrees of freedom. as well as actuated torques and external forces on the system. For any given set of input to be physically valid, the equations of motion must be simultaneously satisfied.

There are several ways to derive the equations of motion of an articulated system. The foundation for all these techniques is laid by Newton's laws of motion. The two main formulations that emerge are the Lagrangian formulation and the Newton-Euler formulation. Both are equally efficient and lead to linear time computable recursive equations.

The Lagrangian formulation is often a simpler approach to begin with. Initial expressions are shorter and its closed form is possible for many simple articulations. It is therefore well-suited as a analytical technique. Indeed, classical texts often begin with the Lagrangian method before introducing others. The basis of the Lagrangian formulation are energy expressions, which when subjected to a series of differentiations give rise to the equations of motion. Because energy terms involve only first order terms, namely joint velocities, and symbolic differentiation can be performed by rote, quantities involving momenta, forces, or joint accelerations need not explicitly occur.

In contrast, the Newton-Euler approach is more structural and algorithmic in nature. Meaningful physical quantities such as momenta and forces are at the core of its formulation. As such the Newton-Euler approach provides an intuitive understanding of the mechanics by which the equations are derived. The approach centers on the influences of bodies among each other, and propagation of physical quantities forms much of its concern. The entire formulation is often stated as a set of recursive equations, which may be taken as a recipe for computing the equations of motion. Unlike the Lagrangian approach, no further differentiation is required to obtain the equations of motion.

Both the Lagrangian and Newton-Euler formulations have been previously used in context of trajectory optimization for animating articulated characters — Lagrangian formulation in [46, 43, 45, 47, 11, 57, 73] and Newton-Euler in [61, 48]. Our system uses the Newton-Euler approach [14]. Because of its algorithmic nature, exploiting structure to optimize our algorithms, especially when computing analytical derivatives, appears to be easier. Further constraining forces in Newton-Euler dynamics is more straight-forward since forces are readily available as part of its formulation.

The equations of motion serve two purposes: Firstly, they provide expressions for joint torques which may be used in a function for energy minimization. Secondly, they provide expressions for constraints on physical validity, which may be used for enforcing Newton's laws of motion. In principle these two issues may be considered separately; and doing so may be desirable when physical validity is wanted, but not energy optimality. In practice, the set of equations performs both functions simultaneously. Unfortunately the equations are coupled in a way that evaluating one without the other is non-trivial. Uncoupling them is the focus of this chapter.

4.2 Physical validity

Given the positions and velocities of an articulated system, the equations of motion associate two quantities: accelerations and forces. There are two variants of the equations, each computing one quantity from the other — equations for forward dynamics express accelerations as a function of forces, and inverse dynamics express forces as a function of accelerations. Mathematically they are equivalent — in linearized equational form it is akin to placing a matrix on the left or its inverse on the right. Inverse dynamics is the form that does not involve an inverse (note the naming irony) and therefore easier to compute: we shall focus on this.

There are two types of forces in a system: forces at the joints exerted by muscle actuation, and forces from the environment. Minimizing the former leads to energy expenditure optimality. Constraining the latter leads to physical validity. In terms of inverse dynamics, uncoupling energy optimality and physical validity means computing one type of forces without the other — in particular, computing external forces without computing joint forces.

Newton-Euler dynamics does not distinguish between the two types of forces — given positions, velocities, and accelerations, the equations compute all forces acting on the system. Known external forces, for instance gravitational forces, are compensated while the remaining are projected onto the system's generalized degrees of freedom. It therefore helps to separate the degrees of freedom into two sets: actuated and unactuated. This leads to the following proposition — for a physically valid state, unactuated forces must be consistent with contact configuration. In a sense, how the unactuated forces come about, for instance a larger force from the left foot or the right, does not matter as much as whether it is possible for the force to exist. But all forces must be accounted for. Given a contact configuration, there is a space of valid forces that may come from the environment. This leads to inequalities that constrain forces to within their valid ranges, as opposed to past spacetime formulations where external forces are explicitly represented as part of the optimization problem and physics constraints are of the equality type.

Unactuated degrees of freedom come from passive joints, that is, joints incapable of exerting forces in directions in which they are capable of motion. In robotics, such mechanisms are known as under-actuated robot manipulators. In humans, all joints are capable of actuating forces, with exception of the six degree of freedom base joint representing the global position and orientation of the body. Forces applied to the base joint must therefore be explainable by external forces which arise from the current contact configuration. We are interested in computing the 6-vector force at the base of the articulation and subsequently its partial derivatives in optimal time. We shall term this force on the base joint the aggregate force on the system.

4.2.1 Definition of terms

We will first briefly note some terms used in the following sub-sections. Detailed descriptions of these terms may be found in Appendices A and B.

D	Degrees of freedom of the articulated figure
X_i^j	Spatial transform from frame i to frame j
X_i^0	Spatial transform from frame i to world frame
s_i'	Joint axis of link i (frame i)
v_i	Velocity of link i (frame i)
a_i	Acceleration of link i (frame i)
I_i'	Spatial inertia of link i (frame i)
p_0	Aggregate momentum of articulated figure (world frame)
f_0	Aggregate force of articulated figure (world frame)
$\underline{c}$	Center of contact support (world frame)
$\underline{N}$	Normal of contact plane (world frame)
$\underline{T}_x, \underline{T}_y$	Tangent vectors spanning rectangular support region (world frame)
μ	Coefficient of friction

4.2.2 Computing the aggregate force

The aggregate force can be derived from the aggregate momentum of the system. Written in spatial notation (Appendix A), the aggregate momentum of an articulated chain comprised of D bodies is defined as the sum of momenta of all bodies in the articulation:

$$p_0 = \sum_{i=1}^{D} X_i^0 I_i' (\sum_{j=1}^{i} X_j^i s_j' \dot{q}_j) \tag{4.1}$$

which may be computed in recursive form as follows:

$$\begin{aligned} v_i &= X_{i-1}^i v_{i-1} + s_i' \dot{q}_i && (4.2) \\ p_i &= X_{i+1}^i p_{i+1} + I_i' v_i && (4.3) \end{aligned}$$

where p_0 is the aggregate momentum. The aggregate force is the time derivative of the aggregate momentum, computed as follows:

$$\begin{aligned} a_i &= X_{i-1}^i a_{i-1} + s_i' \ddot{q}_i + v_i \times s_i' \dot{q}_i && (4.4) \\ f_i &= X_{i+1}^i f_{i+1} + I_i' a_i + v_i \times I_i' v_i && (4.5) \end{aligned}$$

where f_0 is the aggregate force.

This particular formulation is not a convenient form for time optimal computation of the Jacobian

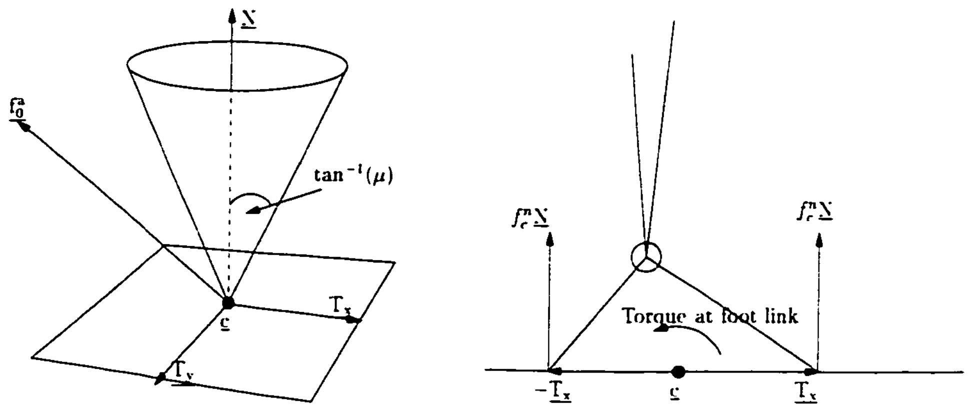

Figure 4.1: (Left) Coloumb contact model at center of support. (Right) Maximum torque exerted by normal force at edge of support.

of $\mathfrak{f}_0$. However, we shall defer this concern till the next section where we discuss linear-time computation of partial derivatives. Here we shall solely concern ourselves with what constitutes a valid aggregate force.

4.2.3 Constraining the aggregate force

There are two cases to consider: contact and flight.

Flight constraints. During flight, no forces (with exception of gravity) may be derived from the environment. Therefore $\mathfrak{f}_0$ must be zero:

$$C_{1-6}(t,x): \quad \mathfrak{f}_0(t) = 0 \quad t \in \text{flight} \tag{4.6}$$

which defines 6 equalities per constrained frame t at flight.

Contact constraints. During contact, the contact force vector is a linear combination of basis vectors of the contact support [55]. To simplify computations, we shall assume that the contact support is a oriented rectangular region spanned by two orthogonal tangent vectors $\underline{T}_x$ and $\underline{T}_y$. The unit normal of the plane is $\underline{N}$, and the center of support is $\underline{c}$ (Figure 4.1, left). A contact model based on Coulomb's friction model is constructed at $\underline{c}$, where the coefficient of friction is the constant μ. The contact configuration is part of the problem specification, hence $\underline{T}_x$, $\underline{T}_y$, $\underline{N}$, and $\underline{c}$ are known at all times.

The aggregate force f_0 consists of two parts, a linear force and a torque, organized as two 3-vectors, and is translated to c as follows:

$$\mathrm{f}_c = \begin{bmatrix} \underline{\mathrm{f}}_c^a \\ \underline{\mathrm{f}}_c^b \end{bmatrix} = \begin{bmatrix} \underline{\mathrm{f}}_0^a \\ \underline{\mathrm{f}}_0^b + \underline{c}^0 \times \underline{\mathrm{f}}_0^a \end{bmatrix} \tag{4.7}$$

where $\underline{c}^0$ is the world vector from the base of the articulation to $\underline{c}$.

The linear part is constrained within the friction cone as follows: Let the linear force projected onto the contact normal be:

$$f_c^n = \underline{N}^T \underline{\mathrm{f}}_c^a \tag{4.8}$$

The maximum angle subtended by the friction cone is given by $\tan^{-1}(\mu)$. The angle between the linear force and the normal is $\cos^{-1}(f_c^n/|\underline{\mathrm{f}}_c^a|)$, which is constrained as follows:

$$C_1(t,x) : 0 \leq \cos^{-1}(\frac{f_c^n}{|\underline{\mathrm{f}}_c^a|}) \leq \tan^{-1}(\mu) \tag{4.9}$$

Equivalently, the constraint without the inverse trigonometric functions is:

$$C_1(t,x) : 1 > \frac{f_c^n}{|\underline{\mathrm{f}}_c^a|} > \frac{1}{\sqrt{\mu^2+1}} \tag{4.10}$$

The maximum torque exertable along $\underline{T}_x$ or $\underline{T}_y$ is a function of the linear force projected onto the normal and the furthest distance on the support where this projected force may act. The maximum torque along the axis $\underline{T}_y$ is given by (Figure 4.1, right):

$$\begin{aligned} \hat{\underline{T}}_y^T(\underline{T}_x \times (f_c^n \underline{N})) &= f_c^n |\underline{T}_x| (\hat{\underline{T}}_y^T(\hat{\underline{T}}_x \times \underline{N})) \\ &= f_c^n |\underline{T}_x| (\hat{\underline{T}}_y^T \hat{\underline{T}}_y) \\ &= f_c^n |\underline{T}_x| \end{aligned} \tag{4.11}$$

where $\hat{\underline{T}}_x$ and $\hat{\underline{T}}_y$ are the normalized tangent vectors. The aggregate torque along the axis $\underline{T}_y$ is obtained by projecting $\underline{\mathrm{f}}_c^b$ onto $\hat{\underline{T}}_y$:

$$\hat{\underline{T}}_y^T \underline{\mathrm{f}}_c^b \tag{4.12}$$

The aggregate torques along the tangent plane is thus constrained as follows:

$$C_2(t,x) \; : \; -f_c^n |\underline{T}_x| \leq \hat{\underline{T}}_y^T \underline{\mathrm{f}}_c^b \leq f_c^n |\underline{T}_x| \tag{4.13}$$

$$C_3(t,x) \; : \; -f_c^n |\underline{T}_y| \leq \hat{\underline{T}}_x^T \underline{\mathrm{f}}_c^b \leq f_c^n |\underline{T}_y| \tag{4.14}$$

In addition two constraints on the magnitude of the twist force and the projected linear force are imposed.

$$C_4(t,x) \quad : \quad -f_c^n K_{twist} \leq \underline{N}^T \underline{f}_c^b \leq f_c^n K_{twist} \tag{4.15}$$

$$C_5(t,x) \quad : \quad 0 \leq f_c^n \leq K_{force} \tag{4.16}$$

where K_{twist} and K_{force} are constants for maximum twist and normal forces. There are a total of 5 double-sided inequality constraints per frame where contact physics is enforced.

4.3 Discussion

In past energy-optimal spacetime constraints papers, physical validity and energy optimality are often closely related — the constraining equations of motion involve torques, and torques are part of the objective function. The method of spacetime constraints presents a way where both physical validity and energy optimality are achieved. Unfortunately, the computational expense of doing is overwhelmingly daunting for all but the simplest of examples.

Physical validity is a more limited notion of physical optimality. A motion may be physically valid in the sense that Newtonian laws of motion are not defied. Yet, the motion may not be carried out in any energy optimal manner. It is further arguable whether energy optimality is justified in a character animation system, since energy optimal motion is not often intuitive or even desirable. For instance, Luxo's minimal energy pose is one where all links are aligned in an upright manner (adding springs to coerce this pose has side-effects on its dynamics). Humans have a similar energy-minimal stance. It will be rather annoying to the animator if the character insists on its energy minimal pose, requiring him to disperse pose constraints or spring stiffness to coax it away from the unintended pose. Notice that a crouching Luxo is physically valid, but in absence of springs of appropriate strengths not energy minimal.

Ideally we would like objective functions that act as "director level" controls that specify meaningful attributes such as walk in a graceful way, or "as if you are scared", or "like Charlie Chaplin". While such objectives were anticipated by [73], these qualities in a motion seem very hard to encode mathematically.

For computational reasons and the reasons above, we have treated physics validity and energy optimality independently — we have chosen physics validity over energy optimality on the basis that a motion should foremost be physically valid. With this choice, the lower bound per iteration of optimization is reduced from quadratic time to linear time, as will be shown in the following two chapters.

Chapter 5

Linear-time Derivatives

In optimizations that utilize gradient information, calculating derivatives often becomes a computational bottleneck. This is especially true when functions are highly nonlinear and derivative expressions involve a large number of terms.

Most non-trivial functions of an articulated body involve several degrees of freedom: in the worst case all degrees of freedom must be considered . A common example from robotics is a function that computes the world coordinates of an end joint in a serial manipulator. Evaluating this function requires $O(D)$ time, where D is the number of degrees of freedom, because the world position of the end joint is dependent on all joints leading up to it. It is also well-known however that its derivative, commonly known as the Jacobian, can also be computed in $O(D)$ time. This is done by accumulating information in a way such that no redundant computation is performed. The idea is similar to the principles of dynamic programming, where recursion, re-use of common expressions, and linear-time precomputations are instrumental to formulating efficiently computable functions.

Computing the Jacobian of a non-trivial function in time linear in the degrees of freedom of the body is algorithmically optimal — because the Jacobian contains $O(D)$ values, assembling the vector of values itself requires $O(D)$ time. In practice, linear time computability is important, because quadratic or higher orders of computation inherently limit the scalability of the system.

This section describes linear-time derivative computations of aggregate functions of an articulated body. In particular, we are most interested in the aggregate force arising from the Newtonian equations of motion as our physics constraints are formulated around this force.

5.1 Computing analytical Jacobians

Multibody functions are deterministic constructs of articulated body configurations. They are inherently procedural, traversing the articulation while building information toward the function results. Evaluating the functions is therefore of computational order linear in the degrees of freedom.

One could in essence differentiate these procedures by rote — traverse the articulation in the exactly same way, but, instead of function values, compute derivative values. This is what most automatic differentiators do. It results in $O(D^2)$ time Jacobian computations — each traversal obtains one partial derivative: there are D of them.

Some automatic differentiators accumulate both function values *and* derivatives during function evaluation. This is often done by tagging each intermediary computation with D partial derivatives. This also results in $O(D^2)$ Jacobian computation — because each computed value may depend on all D degrees of freedom, maintaining D values within a traversal of length D is a double "for" loop each indexing D values.

One observation is that in order for an automatic differentiator to provide $O(D)$ time derivatives to an $O(D)$ procedure, it must have the ability to remove one "for" loop. It must therefore analyse the given procedure, and infer a strategy for accumulating information in a way that $O(D)$ partial derivatives can be simultaneously computed in a single sweep. Local analysis does not suffice. In short, the automatic differentiator program must itself infer a dynamic program. It is perhaps too much of a burden on the differentiator, but it would be presumptuous to assume it is not possible.

In our system, analytical derivatives are computed and coded manually, at times with help from a symbolic package. This allows us to maintain full control over algorithmic efficiencies but more importantly, it allows us to provide provable orders of running times. It is important to note that, like the Newton-Euler equations of motion, the derivative expressions are fully general and need only be derived once — any tree-like articulated body may be constructed and the derivative expressions may be applied without modification; only the propagation order of the quantities changes with different articulations. Thus, anyone interested in replicating our results may directly use the equations as provided in the Appendices of this thesis.

As described in the previous chapter, aggregate functions are central to our definition of physical validity. Functions and constraints that are defined on aggregate functions are differentiated via the chain rule. Unlike differentiating aggregate functions which involve recursive expressions, this latter differentiation is relatively easy — since aggregate functions compute vectors of constant size (usually a spatial vector), any function defined over an aggregate function has parameters of fixed length. The Jacobian of such a function with respect to its parameters therefore has constant dimensions and thus can be evaluated and differentiated in constant time.

The following sections describe linear-time algorithms to compute the partial derivatives of functions defined on an articulated body, beginning with a simple kinematic Jacobian. leading to the Jacobian of the aggregate force.

5.2 Kinematics Jacobians

Efficient ways to compute kinematic Jacobians have been known for a long time. The kinematic Jacobian is used extensively in robot manipulators, and in animation its efficiency is crucial for interactive inverse kinematics systems.

A simple kinematic function is a function that computes the end location of a serial manipulator. Though simple, the example shows how one might err toward inefficient Jacobian computations.

Kinematic Jacobian of end-effector. Suppose we are interested in computing the world coordinates of the end joint of a serial chain of D one degree-of-freedom links. each joint distanced from its parent joint by the vector $\underline{r}'_i$. One could proceed as follows:

1. Compute the world rotation matrix at each joint:

$$\begin{aligned} \underline{R}^0_1 &= \underline{R}'_1(q_1) && (5.1)\\ \underline{R}^0_i &= \underline{R}^0_{i-1}\underline{R}'_i(q_i) \quad \text{for } i = 2,..,D && (5.2) \end{aligned}$$

2. Rotate each $\underline{r}'_i$ to world space:

$$\underline{r}_i = \underline{R}^0_i \underline{r}'_i \tag{5.3}$$

3. Sum up all the rotated vectors:

$$\underline{p} = \sum_{i=1}^{D} \underline{r}_i \tag{5.4}$$

The position of the end joint in world coordinates is $\underline{p}$ and its Jacobian is the $3 \times D$ matrix whose columns are $\partial\underline{p}/\partial q_i$. Though it serves its purpose, problems arise when computing the Jacobian using this representation — the matrix $\underline{R}^0_i$ depends on $q_1, q_2, ..., q_i$, therefore the partial derivative $\partial\underline{p}/\partial q_i$ has i affected terms. The total number of affected terms of all partials is $D(D-1)/2$. which leads to an $O(D^2)$ time Jacobian computation.

To overcome this problem, consider the following routine:

1. Compute the vector from each joint to the end joint in its local frame:

$$\underline{r}_D = \underline{r}'_D \tag{5.5}$$

$$\underline{r}_i \quad = \quad \underline{R}'_{i+1}\underline{r}_{i+1} + \underline{r}'_i \qquad \text{for } i = D-1,\dots,0 \tag{5.6}$$

2. The position of the end joint is simply:

$$\underline{p} = \underline{r}_0 \tag{5.7}$$

The Jacobian can be obtained as follows:

$$\frac{\partial \underline{p}}{\partial q_i} = \underline{R}_i^0(\underline{s}_i \times \underline{r}_i) \tag{5.8}$$

for $i = 1..D$, where $\underline{s}_i$ is the joint axis of joint i. Since each partial derivative is computed in constant time, the full Jacobian is obtained in linear time.

This simple example highlights a few points: Firstly, the recursive formulation has much effect on derivative computations, even though numerically they compute the same values with the same effort. Secondly, intermediate values computed during evaluation are often helpful in derivative expressions. Lastly, parameters preceding a joint should preferably not affect aggregate values computed at and beyond the joint; reversing the order of traversal often helps in achieving this goal.

5.3 Dynamics Jacobians

Differentiating aggregates of dynamics equations in linear time presents more difficulties than differentiating kinematic equations in linear time. There are two main reasons for this: Firstly, dynamics equations contain more cross product terms, such as terms representing coriolis forces. Because of the non-associative nature of cross products, factorizing these terms into aggregate forms to facilitate differentiation is more difficult. Secondly, the inertia tensor present in dynamics equations complicates computing aggregate terms because inertia tensors are matrices and matrix products are non-commutative. The following sections highlight some of these problems and how we overcame them, starting with the momentum equation of a serial articulation, whose time derivative leads to the force equation.

5.3.1 Linear Time Momentum Jacobian

Efficiently computing $\partial \mathfrak{f}_0/\partial q$, the force Jacobian, requires efficiently computing $\partial \mathfrak{p}_0/\partial q$, the momentum Jacobian, because aggregate force $\mathfrak{f}_0$ is the time derivative of aggregate momentum $\mathfrak{p}_0$. We begin with a discussion of the momentum equations and present an argument that the momentum

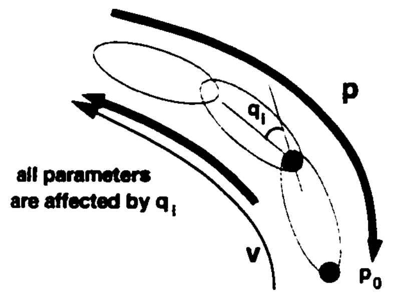

Figure 5.1: The effect of parameter q_i is propagated up the tree with velocities v and back down the tree with momentum terms p. Computing $\partial p_0/\partial q_i$ requires $O(D)$ time and results in an $O(D^2)$ algorithm for computing the momentum Jacobian.

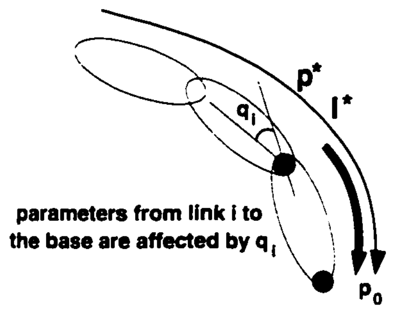

Figure 5.2: The effect of rewriting the recursion is to limit the effect of q_i to parameters collected at joints between i and 0. Terms required for the momentum Jacobian are accumulated in a single pass from leaf to base, and the momentum Jacobian can be computed in linear time.

Jacobian can be computed in linear time. Section 5.3.2 extends this linear time result to the force Jacobian, the quantity required to compute derivatives of physics constraints.

The usual way to compute aggregate momentum is to formulate the following recursion:

$$\mathrm{v}_i = \mathrm{X}^i_{i-1}\mathrm{v}_{i-1} + \mathrm{s}'_i\dot{q}_i \quad (5.9)$$

$$\mathrm{p}_i = \mathrm{X}^i_{i+1}\mathrm{p}_{i+1} + \mathrm{I}'_i\mathrm{v}_i \quad (5.10)$$

where p_0 is the desired result.

Velocities v_i are propagated from base to leaf, and momentum p_i is propagated from leaf to base. Figure 5.1 shows this process. Parameter q_i appears in the coordinate transforms X^i_{i+1} and X^{i+1}_i, and so every v_j for $j > i$ depends on q_i, and every p_j for $j \geq 0$ depends on q_i. Unrolling the recursion to collect terms for $\partial \mathrm{p}_0/\partial q_i$ requires $O(D)$ time. There are D terms q_i, and this approach will lead to an $O(D^2)$ computation for the momentum Jacobian. There is no clever way to simplify the calculation by aggregating terms when it is presented in this form.

We observe that rewriting the recursion solves this dilemma:

$$\begin{aligned}
I_i^* &= X_{i+1}^i I_{i+1}^* X_i^{i+1} + I_i' && (5.11)\\
p_i^* &= X_{i+1}^i p_{i+1}^* + I_i^* v_i' && (5.12)\\
p_0 &= p_0^* && (5.13)
\end{aligned}$$

The key thing to notice here is that p_i^* is expressed as a function of v_i', which is a local variable at link i. As a result, only propagation from leaf to base is required, and each parameter q_j does not affect terms computed for joints $j+1$ and beyond (Figure 5.2). Also note that p_i^* is in general not equal to p_i if $i \neq 0$. A term superscripted with an asterix should be treated only as an intermediary quantity, unless its subscript is zero in which case it is the desired aggregated result.

A linear time expression for the momentum Jacobian can be derived in a straightforward manner based on this form of the recursion. The results of this calculation are presented in Appendix B. Note that we are not simplifying or changing the outcome of the dynamics computation, only changing the order in which terms are computed. Aggregate momentum p_0 and the momentum Jacobian are exactly the same in both formulations.

5.3.2 Linear Time Force Jacobian

In a traditional inverse dynamics formulation, force is represented by taking the time derivative of Equations 5.9 and 5.10 to obtain:

$$\begin{aligned}
a_i &= X_{i-1}^i a_{i-1} + s_i'\ddot{q}_i + v_i \hat{\times} s_i'\dot{q}_i && (5.14)\\
f_i &= X_{i+1}^i f_{i+1} + I_i' a_i + v_i \hat{\times} I_i' v_i && (5.15)
\end{aligned}$$

where the symbol $\hat{\times}$ is the cross product operator for spatial vectors (Appendix A). As with momentum, this form results in an expression for the force Jacobian that requires $O(D^2)$ time to compute. For fast computation, we instead take the time derivative of Equation 5.13, which results in

$$f_i^* = X_{i+1}^i f_{i+1}^* + v_i' \hat{\times} p_i^* + I_i^* a_i' + \dot{I}_i^* v_i' \qquad (5.16)$$

This equation has the properties we are looking for. Velocity v_i' and acceleration a_i' are local to link i, and terms are propagated from leaf to base only. Note that as is the case of aggregate momentum, f_i^* is in general different from the actual joint force f_i if $i \neq 0$.

Differentiating Equation 5.16 while accumulating the coefficients of derivative elements results in

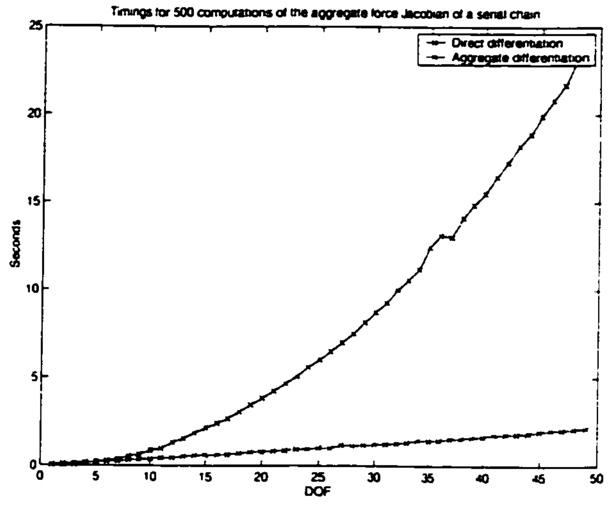

Figure 5.3: Timing of 500 computations of the Jacobian of the aggregate force by direct differentiation and by the linear-time method.

the simplified form as given in the Appendix B. Each partial derivative of the aggregate force with respect to joint positions, velocities, and accelerations may be obtained in constant-time, and hence the full Jacobian obtained in linear-time.

5.4 Benchmarking

Figure 5.3 shows timing results in computation of all partial derivatives of the aggregate force via the proposed method and by direct differentiation of the Newton-Euler equations of motion. Numerically the partial derivatives are identical. The articulated model is a serial chain ranging from 3 to 50 links. As expected, the proposed method is linear in the degrees of freedom, while direct differentiation shows quadratic growth. It is also observed that despite overheads in computing aggregate intermediate terms, the linear-time method shows a computational advantage with as few as 5 degrees of freedom.

5.5 Discussion

The ability to compute Jacobians in linear time depends to a great extent on the recursive formulation of the function. Unlike writing expressions and coding procedures where derivatives are never needed, functions where derivatives are needed must be written with the derivative computation in mind. Given the nature of the articulated figure — that joint angles influence motion in a direction away from the base — a choice such as forward or back propagation has much effect on the complexity of the derivative computations.

The size of the Jacobian sets a lower bound on computation time. If a function is a D-valued vector function of a D degree-of-freedom articulation, its Jacobian matrix has dimensions $D \times D$ — the optimal time for computing a $D \times D$ matrix is trivially $O(D^2)$. This implies that whenever non-trivial functions that compute D values are used in an optimization system. such as previous spacetime formulations where D Newtonian constraints are enforced per constrained frame. the system will have an inevitable lower bound of $O(D^2)$. Our proposed formulation enforces a constant number of constraints per constrained frame, which makes linear-time Jacobian computations possible.

The preceding sections illustrates partial derivatives of recursive equations of aggregate momentum and force with respect to the joint parameter q_i. In general we would also need partial derivatives with respect to $\dot{q}_i$ for the momentum equation, and in addition $\ddot{q}_i$ for the force equation. These partials. however. are relatively easy to obtain as they mostly involve scalar multiplications with readily available terms. The full expressions for the aggregate momentum and force Jacobians are provided in Appendix B.

This chapter addressed one of the two compute-intensive parts of a trajectory optimization system, namely the computation of partial derivatives. It is shown that partial derivatives of physics constraints can be computed in linear time. The other computationally intensive part concerns solving linear systems in sequence within the numerical optimizer and will be discussed next. The next section will describe how the linearity we have achieved here extends to the numerical solution process.

Chapter 6

Computational Complexity

Nonlinear optimization is a non-deterministic, iterative process. Without prior knowledge of the number of iterations, it is not possible to provide a complete complexity analysis of the entire solution process. Convergence studies that lead to claims of linear- or superlinear-convergence are usually only applicable for restricted classes of problems. Often convexity is assumed. In our case, convexity conditions do not hold and therefore no claim on the rate of convergence is possible. The best we can do is to analyse per-iteration complexities, and provide empirical results on overall convergence.

In practice the number of iterations depends on several factors such as: initial guess, nonlinearity, conditioning, and singularity of the equations; scaling of variables and functions; and sometimes even a little luck [20]. Here I shall only analyse per-iteration complexity without strong claims on convergence properties.

At every optimization iteration, two steps dominate the computational effort. The first is the computation of derivatives, and the second, solving a set of sparse linear equations to obtain a descent direction. An analysis of both is important — though function derivatives are expensive, the overhead within the optimizer becomes significant as the number of variables becomes large.

In order to discuss optimization complexity, it is necessary to introduce a suitable optimization model, even though in practice one might opt to use an optimization package to avoid the intricacies of coding a nonlinear optimizer. For the purpose of complexity analysis, I will use the projected Lagrangian SQP algorithm described in [17, pages 237–241], and is also used in [73].

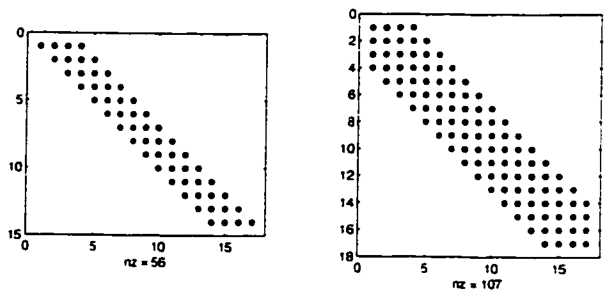

Figure 6.1: The Hessian matrix (right) of a least-squares function (left) of joint accelerations has a symmetric band diagonal structure.

6.1 Assumptions

To apply the SQP model to be described, one further assumption has to be made. We will assume that the objection function is a simple function of the optimization variables. By simple we mean that its Hessian is a sparse matrix whose linear system can be solved in linear time. One function that fits this criterion is the least-squares of a linear function where each term depends on a constant number of variables. Two such examples will be given here.

Displacement maps. As mentioned in chapter 1, the displacement map is a useful representation for controlling changes to a pre-existing motion. The squares integral of the displacement map is a simple function that can be formulated as follows:

$$f(\mathbf{x}) = \mathbf{x}^T M \mathbf{x} \tag{6.1}$$

where $\mathbf{x}$ is the set of control points of the displacement map, and M is a diagonal matrix of appropriate weights. This objective function is used in [18] for kinematic motion optimization and a similar one for exponential maps can be found in [41]. The Hessian of f is simply M whose inverse is a diagonal matrix of reciprocal elements in M.

Joint accelerations. The vector of joint accelerations is a linear function of the control points:

$$\ddot{\mathrm{q}}(x, t) = \ddot{\phi}(t)\mathbf{x} \tag{6.2}$$

where ϕ is the matrix of cubic B-spline basis evaluated at time t. Each row in ϕ has at most four non-zero entries (Figure 6.1, left). The integral of squared joint accelerations can be expressed as:

$$f(x) = (\ddot{\phi}\mathbf{x})^T \ddot{\phi}\mathbf{x} = \mathbf{x}^T(\ddot{\phi}^T \ddot{\phi})\mathbf{x} \tag{6.3}$$

The bracketed term is the Hessian of f and has a symmetric band diagonal structure with a maximum bandwidth of four (Figure 6.1, right). The maximum bandwidth of a band diagonal matrix is the number of non-zero diagonals adjacent to the main diagonal. A linear system involving a symmetric band diagonal matrix can be solved in linear time [58, 65]. This objective function is used in [44] for animating a human diver.

6.2 Problem formulation

Optimizing a motion involves minimizing an objective function subject to a set of constraints. In spacetime constraints, physically based constraints are enforced at a finite number of frames. To obtain plausible physics across the motion, the number of physics-enforced frames should be at least as dense as the discretization of the motion — a sparser distribution allows too much freedom for error between physics-enforced frames. A denser distribution of physics constraints on the other hand may overly constrain the system such that no solution may be found.

With K basis functions and D motion curves, one for each degree of freedom, the size of the search space is KD, which we denote by the vector $\mathbf{x}$. Our method requires a maximum of 6 physics constraint equations per physics constrained frame. For a trajectory of K basis functions, K physics constraints are imposed on the motion. The total number of physics constraints is hence $6K$. For the sake of conciseness, we shall let $n = KD$ and $m = 6K$. The Jacobian of the constraints is therefore an $m \times n$ matrix, and the Hessian of the objective function is a $n \times n$ matrix.

6.3 Sequential quadratic programming

The most general form of the nonlinear programming problem can be expressed as follows:

$$\min_{\mathbf{x} \in \Re^n} \quad f(\mathbf{x}) \tag{6.4}$$

subject to

$$C_j(\mathbf{x}) \quad \geq \quad 0, \quad 1 \leq j \leq l \tag{6.5}$$

$$C_j(\mathbf{x}) \; = \; 0, \quad l+1 \leq j \leq m \tag{6.6}$$

Since inequality constraints can be mapped to equality constraints via active sets, we consider the following equivalent form:

$$\min_{\mathbf{x}\in\Re^n} \; f(\mathbf{x}) \tag{6.7}$$

subject to

$$C_j(\mathbf{x}) \; = \; 0, \quad 1 \leq j \leq m \tag{6.8}$$

Here, f and C_j are twice-continuously differentiable scalar functions, possibly nonlinear.

One common approach to solving this nonlinear optimization problem is Sequential Quadratic Programming (SQP). The method of SQP, as its name implies, casts (6.8) into a sequence of subproblems, where each subproblem is solved using a locally quadratic model. In this particular variant of SQP, a second-order Newton-Raphson step in f is first computed, followed by a first-order Newton-Raphson step in the C_i's. The two steps are then combined by projecting the first onto the null space of the second. The method therefore requires the first and second derivatives of the objective function, and the Jacobian of the constraints.

The Jacobian of the constraints is an $m \times n$ matrix where the jth column holds the partial derivative of C_i with respect to the variable x_j.

$$J_{ij} = \frac{\partial C_i}{\partial x_j} \tag{6.9}$$

The first derivative of the objective function, $\frac{\partial f}{\partial x_i}$, is an $n \times 1$ vector, obtained by differentiating the scalar function f with respect to each variable. The second derivative of f, known as the Hessian, is an $n \times n$ matrix expressed as:

$$H_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j} \tag{6.10}$$

The SQP step is obtained by solving two linear systems in sequence. The first solves for a step $\widehat{\Delta x_j}$ that minimizes a second-order approximation to f without regard to the constraints:

$$-\frac{\partial f}{\partial x_i} = H_{ij}\widehat{\Delta x_j} \tag{6.11}$$

The second yields a step $\widetilde{\Delta x_j}$ that drives linear approximations to the C_i's simultaneously to zero.

and at the same time projects the optimization step $\widehat{\Delta x}$ onto the null space of the constraint Jacobian:

$$-C_i = J_{ij}(\widehat{\Delta x} + \widetilde{\Delta x}) \tag{6.12}$$

The final update to the variables is:

$$\Delta x = (\widehat{\Delta x} + \widetilde{\Delta x}) \tag{6.13}$$

The algorithm repeats the steps and reaches a fixed point when $C_i = 0$ and when any further decrease in f requires violating the constraints.

The main computational cost in this method lies in providing SQP with the various derivatives, and solving the two linear systems (6.11) and (6.12). The derivative matrices resulting from our problem are typically sparse, hence direct methods that invert matrices in $O(n^3)$ time are usually not necessary. The following describes how each linear system may be solved.

Solving linear system (6.11). The objective function f is assumed to be a simple function of the variables whose Hessian is invertible in linear time. If f is not simple, for instance when f contains terms which are functions of the kinematics or dynamics of an articulated body, quasi-Newton SQP methods which approximate the Hessian matrix of f are needed. Analysis of computational complexity must then also account for the various updates to the Hessian. To retain the SQP model presently described, we shall only consider simple objective functions.

For simple functions, the Hessian matrix has a sparse structure whose linear system can be solved in linear time either by Gaussian elimination or by methods which take advantage of its special structure [58, 65]. Solving (6.11) thus involves computing

$$\widehat{\Delta x_j} = -H_{ij}^{-1}\frac{\partial f}{\partial x_i} \tag{6.14}$$

and can be computed in linear time.

Solving linear system (6.12). A solution to (6.12) involves inverting the Jacobian of the non-linear physics constraints. As described earlier (section 6.2), given K basis functions over a B-spline motion, we enforce K physics constrained frames, each frame having at most 6 physics constraints.

The Jacobian matrix J_{ij} is therefore a non-square $6K \times KD$ sparse matrix. Due to the local influence of B-spline basis functions, each constraint is affected by at most $4D$ control points. Thus, J_{ij} has a sparse structure where each row has at most $4D$ non-zero entries. Matrix-vector multiplication between J_{ij} and a n-vector is therefore $O(KD)$.

For any articulated structure with D greater than 6 (always the case for three dimensional models), J_{ij} is wider than it is tall and the linear system (6.12) is indefinite. Intuitively, this provides room for the optimizer to minimize its objective, while the Jacobian of the constraint matrix provides the necessary information for it to navigate away from undesirable areas of the domain. Mathematically, it implies that the null space of the constraints is non-zero.

Because J_{ij} is not a square matrix direct inversion is not possible — there exist an infinite number of solutions that can satisfy (6.12). The usual strategy is to compute a pseudo-inverse solution — a solution to under-constrained systems of equations which at the same time minimizes the norm of Δx.

To compute the pseudo-inverse solution, we use the conjugate gradient method. The conjugate gradient method solves a system of linear equations by iteratively minimizing its squared errors. In the case of (6.12), the conjugate gradient method computes:

$$\min_{\widetilde{\Delta x}} \quad | J_{ij}(\widehat{\Delta x} + \widetilde{\Delta x}) + C_i |^2 \tag{6.15}$$

One advantage of the conjugate gradient method is that it computes the solution of a linear system using only products of the matrix, J_{ij}, with a vector — an $O(KD)$ operation. The remaining question is the number of conjugate gradient iterations required and hence the number of times this matrix/vector multiplication is performed.

For a linear system of equations $Ax = b$, it is known that in exact arithmetic, the number of conjugate gradient iterations required is equal to the number of distinct eigenvalues of A. Since J_{ij} is non-square, the linear system is indefinite and its eigenvalues are not defined. Instead, we can examine the normal equations, $A^TAx = A^Tb$ [1]. In our case J_{ij} has the lesser row dimension of $6K$, therefore $J_{ij}^TJ_{ij}$ has at most $6K$ distinct eigenvalues [2]

It follows that the conjugate gradient solution to (6.15) requires at most $6K$ iterations. Because each conjugate gradient iteration is $O(KD)$, the linear system can be solved in $O(K^2D)$ time.

6.4 Per-iteration computational complexity

Two tasks dominate the computational costs per optimization cycle: computing derivatives and solving linear systems.

[1] $J_{ij}^TJ_{ij}$ should never be explicitly computed since it is an $O(K^2D^2)$ operation. Instead, either compute any matrix/vector multiplication in sequence, or use conjugate gradient methods specific for indefinite systems. The latter is further advisable, since $J_{ij}^TJ_{ij}$ has twice the condition number of J_{ij}. Details in [16].

[2] This is the case since for any matrix A, A^TA and AA^T has the same eigenvalues.

Computing derivatives. The cost for computing derivatives is bounded by the most expensive function in the system. In physically based motion optimization, these functions are the Newtonian equations of motion. In our formulation, the physics constraints can be evaluated and differentiated in linear time. When all other functions in the system can also be evaluated and differentiated in linear time, the cost for providing all derivatives to the optimizer is linear in the number of variables.

Solving linear systems. Two linear systems must be solved in the SQP algorithm. The first involves the Hessian of the objective function and can be solved in linear time using Gaussian elimination. The second linear system involves the Jacobian of the constraints and requires an iterative conjugate gradient solution. The maximum number of iterations is $6K$. Because each iteration is $O(KD)$, the cost of this step is $O(K^2D)$, K being the number frames where physics is enforced.

Per-iteration cost. The asymptotic per-iteration cost is the sum of the previous two costs: $O(KD) + O(K^2D) \approx O(K^2D)$. The asymptotic per-iteration cost in the our method is thus linear in the complexity of the articulated figure.

6.5 Discussion

The analysis shows that using this SQP model, enforcing physical validity is linear in the complexity of the figure and quadratic in the number of basis functions. This thesis does not address the quadratic factor with respect to number of basis functions. When the number of basis functions is large, a problem of a different nature sets in, namely ill-conditioning, which has been addressed by other authors, such as [47]. Numerically however, the outlook concerning this quadratic factor is optimistic. Firstly, since the conjugate gradient inner iteration computes the solution to an approximate subproblem, it is not necessary to compute an exact solution. Further, as Gleicher showed in motion optimization [20], the empirical costs of a well-conditioned conjugate gradient step could actually be roughly linear in the number of variables, suggesting a constant number of conjugate gradient iterations independent of the number of variables. In some nonlinear optimizers such as Lancelot [12], a hard limit (Lancelot's default is 200) is set on maximum number of conjugate gradient iterations, even when the number of variables is much larger.

The analysis also implies that theoretically, enforcing physical validity is no more expensive than kinematic based motion optimization. Whereas past spacetime formulations are bounded by quadratic cost per iteration, even when the articulated model is replaced with a simplified model, our method has linear cost per iteration. This suggest that enforcing valid physics in an interactive animation or editing system is possible, in the way that it has been demonstrated to be possible in kinematic-based motion editing systems. However, pragmatically speaking, our constraint equations are almost certain to be more nonlinear and therefore more difficult to satisfy. As a result, we would expect our system to require more SQP iterations, each taking smaller descents.

Chapter 7

Results

This chapter describes a sampling of the results. The articulated figure we adopted is a 22-DOF character — 19 rotational DOF, and 3 translational DOF. The anthropometry details of the character are obtained from [53, Chapter 6]. The height of the model is 1.65m and the weight is 65kg. We also tested the character with our algorithm in symmetric-planar mode. In planar mode, the movements of the rotational joints are constrained along the axial (body left-right) direction, represented as revolute joints whose rotation axis is along the Z-axis. The left and right limb movements are symmetric about the vertical cross-section of the body. The symmetric-planar character thus has 5 rotational degrees of freedom, and 2 translational degrees of freedom.

As far as possible, we provide minimal input to the optimizer. For example, no key frame or suggestions of pose are provided thoughout the results in this section (with exception of the final pose in the dismount example). The expected behavior of the animation, such as squash-and-stretch and swinging of limbs, emerges from the requirement of correct physics.

Although the initial guess is often an important part of any local optimization technique, we find that for many motions the initial guess need not be excessively close to the final solution. No special effort is made to construct good initial joint motions in all the animations produced here. The initial joint motions are flat B-splines curves representing a constant pose over time. This pose can trivially be set to something appropriate to the motion. For instance, we use an upright pose with arms extended upwards for a swing motion and extended downwards for a running motion. One possible explanation for why initial joint motion curves need not be overly contrived is that the local minimal motion is often one whose joint motion centers about some neutral pose. An approximation to this neutral pose as an initial motion is sufficient for the optimizer to descend toward the expected minima.

High-bar Dismount Setup Information	
DOF	7 (5 rotational, 2 translational)
2D/3D	2D
Number of variables	105
Implicit constraints	Hand contact during swing Feet contact during landing Swing time 0.9s, Flight time 0.6s, 0.8s
Explicit constraints	Initial COM velocity zero Fixed final pose, final joint velocity zero
Number of iterations	650
Time per iteration	0.04s
Total time	0.43min

Figure 7.1: Setup information for the swing example.

Simple kinematic interpolations of the character's global rotation and translation can be useful for constructing more meaningful initial motions. For the initial motions used to produce the animations here, the position of the body is kinematically interpolated to fit the constraints of the task. For instance, the body is placed to satisfy foot stance constraints during support phases, and translated through a projectile motion during flight between support stances. When no overall rotation of the body is expected in the motion, the initial global rotations are simply flat curves. In cases where an overall rotation is expected, such as the high-bar dismount routine, the global rotation is interpolated over the approximate span of the body rotation.

7.1 High-bar Dismount

The dismount animation is in 2D. The character is modeled in symmetric-planar mode. The goal is to have the character swing off a high-bar, perform a single backward somersault in the air, and land on both feet. The initial linear center-of-mass velocity is constrained to zero so that the character starts at the peak of the preparatory swing. The initial velocities of the joints however may be non-zero. The final joint velocities are constrained to zero. The final pose, timings, and landing distance are constants provided by the user.

Figure 7.1 details the setup and the numerical benchmarks for the dismount animation. Figure 7.2 shows the film-strip results of the animation.

The initial guess (Figure 7.2, Top) is a kinematic interpolation of a vertically hanging character to the final constrained pose. The body rotation of the final pose is set to 360 degrees such that the initial guess interpolates the body over one complete rotation. Note the the initial motion appears very unstable at landing — the character would fall over. The flight times are controlled by the user.

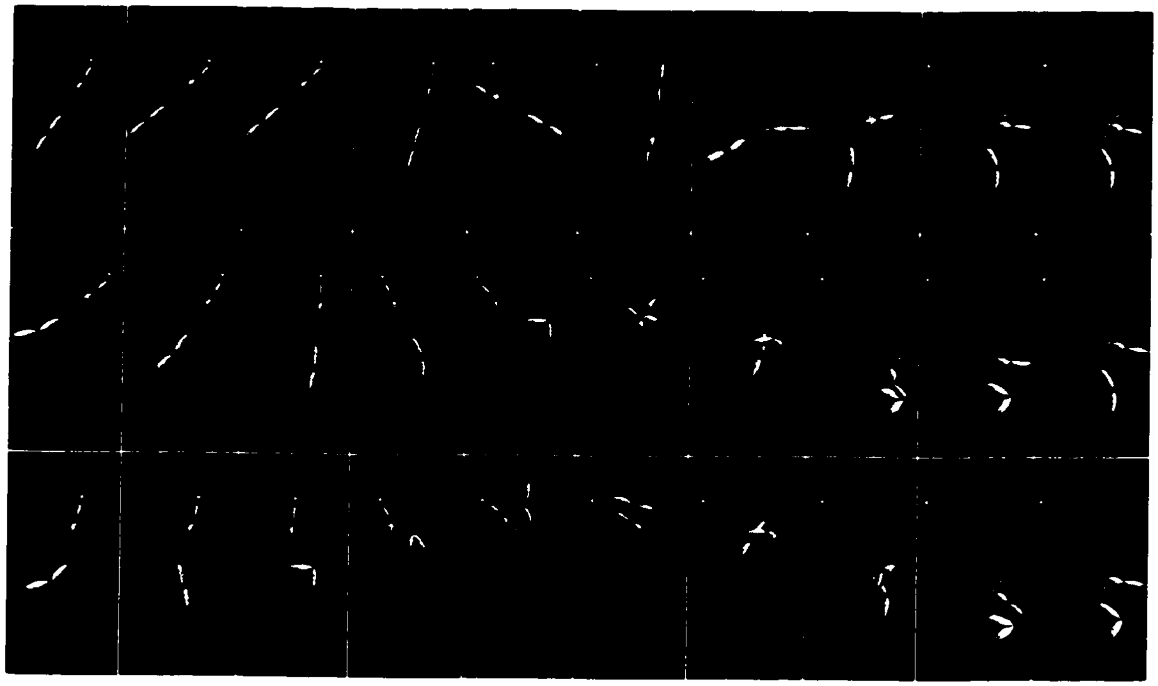

Figure 7.2: High-bar dismount with single back somersault. (Top) Initial guess. (Middle) 0.6s flight time. (Bottom) 0.8s flight time.

With a flight time of of 0.6s (Figure 7.2, Middle), the character performs a tightly tucked somersault and a fast whipping motion. The maximum height of the flip is well below the high-bar. With a flight time of 0.8s (Figure 7.2, Bottom), the flip is noticably more relaxed. The tuck is looser and the flight trajectory is higher, resulting in the maximum height of the flip exceeding the high-bar.

7.2 Monkey Bars

The monkey bars motion, and all subsequent animations, are in 3D using the 22-DOF character. The goal is for the character to traverse four sections of the monkey bars. The timings of the contact periods are fixed by the user.

Figure 7.3 details the setup for this example. Figure 7.4 shows the film-strip results of the animation.

The initial guess (Figure 7.4, Top) consists of the character, with arms extended upwards, translated over four sections of the monkey bars in this fixed pose. The period where the hand is in contact with the bar is set to 0.7s. Figure 7.4 (Middle, Bottom) shows the results of the optimization. Close examination of the resulting motion reveals that the lower arm intersects with the monkey bars as it moves from one contact to another. Ideally, the geometry of the environment

Monkey Bar Setup Information	
DOF	22 (19 rotational, 3 translational)
2D/3D	3D
Number of variables	532
Implicit constraints	Hand contact during support Support time 0.7s, Zero flight time
Explicit constraints	none
Number of iterations	1330
Time per iteration	0.11s
Total time	2.4min

Figure 7.3: Setup information for the monkey bar example.

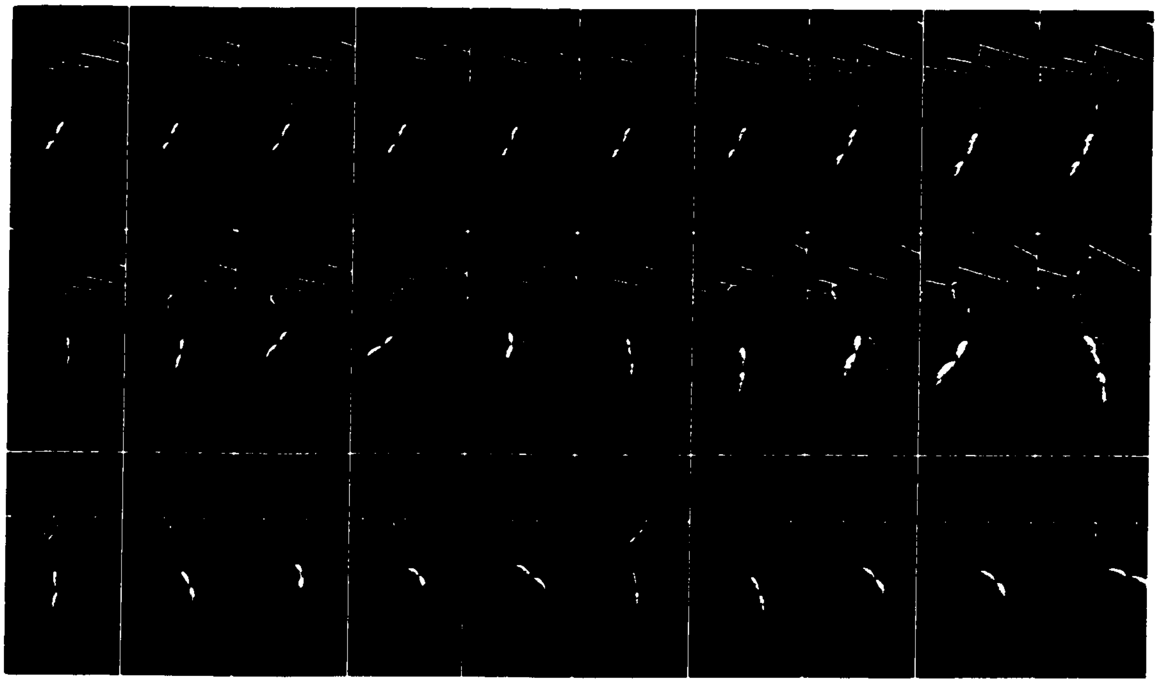

Figure 7.4: Traversing the monkey bars. (Top) Initial guess, (Middle) Front view, (Bottom) Side view.

Peg Running Setup Information	
DOF	22 (19 rotational, 3 translational)
2D/3D	3D
Number of variables	532
Implicit constraints	Foot contact during support Support time 0.35s, Flight time 0.4s
Explicit constraints	none
Number of iterations	2213
Time per iteration	0.11s
Total time	4.0min

Figure 7.5: Setup information for the peg running example.

Figure 7.6: Pegs leaping

should be factored into the optimization as well. This would have subdued the arm motion to adopt a lower swing in order to avoid the bar collision. However, for the purpose of this demonstration, the results of the optimization are shown verbatim from the physics optimization without further touch ups.

7.3 Leaping

Figure 7.5 shows the setup information for synthesis of the character leaping on vertical stilts. Figure 7.6 shows the film-strip for this animation.

The stilts have a height of 0.2m. Foot constraints are enforced such that the feet land on alternate stilts with user specified timings. The support phase and flight phase are set to 0.35s and 0.4s respectively. Note that the feet of the character consistently extend below the top of the stilts. Had this been a ground running motion, the character's feet would have penetrated the ground. To fix the ground penetration artifact, we enforced collision constraints on the swing leg such that the feet positions stay above the ground (Figure 7.7).

Figure 7.7: Ground leaping with collision constraints.

7.4 Space Spin

Figure 7.8 shows a human figure turning itself upside-down under zero-gravity environment. The aim is to move from a upright to an inverted pose without external contacts. Both angular and linear momentum must be conserved, thus each movement of a body part is counteracted by a movement in an opposite direction. In our approach, this is considered a flight phase under zero gravity, and physics constraints impose the condition that external forces on the system must be zero. In the motion the character swings his limbs in anticipation and curls about the hips to perform the spin.

7.5 Timing.

To empirically test the advantage of our method for fast derivative computation, we ran the peg example (bottom row of Figure 7.5) 5 times, each time with the identical setup except that a different technique was used to compute all required first derivatives. The differentiation techniques tested were:

- **Our method.** Analytical gradient computation using our approach.
- **Direct method.** Analytical gradient obtained by direct differentiation of the equations of motion.
- **NR1.** Numerical differentiation by ordinary forward differences.
- **NR2.** Numerical differentiation by central differences.
- **NR3.** Richardson-extrapolation of order 6.

Figure 7.9 summarizes the results. All timing information is for a 750 MHz Pentium 3 computer.

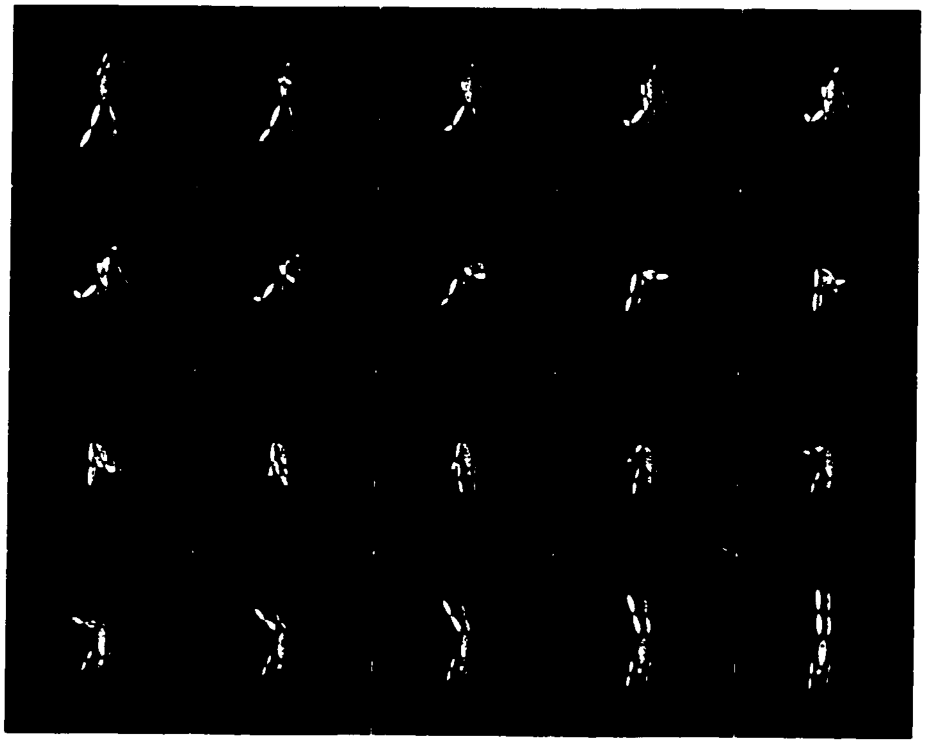

Figure 7.8: Rotating under zero gravity.

Technique	Time for one iteration	Average error
Our method	0.11s	0
Direct method	0.62s	0
NR1	0.97s	0.0010
NR2	1.92s	1.03e-06
NR3	5.73s	1.46e-08

Figure 7.9: Time required for one iteration of the peg example using a variety of differentiation techniques.

Chapter 8

Conclusion

8.1 Summary

The use of optimization as a means to create motion originated from the engineering and robotics community. While techniques designed for robotics mechanisms may be adopted wholesale to generate appealing animations, the fundamental requirements of computer animation and robotics are significantly different. Whereas animation is a medium for visual communication. the field of robots is concerned with the mechanics and construction of actual mechanisms. This dichotomy implies relaxed requirements for formulating techniques that are applicable to animation. And it is under these relaxed conditions, as described in Chapter 2, that this thesis brings forth a method for synthesizing plausible looking animations at a computational cost that is provably lower than similar methods of the past.

Central to this thesis is the notion of what a physically valid motion means. The validity of motion is described in Chapter 1, and then mathematically defined in Chapter 4. By isolating the elements of physical validity from elements of style. one can design motion or effect controlled changes within the space of physically valid motion. We reason that the omnipresence of physical validity implies that it is justified to do so.

Physically correct motion is guided by the principles of Newton's laws of motion with Euler's extension to multiply linked bodies. Unfortunately. the quantities in the Newton-Euler equations of motion are coupled in a way that isolating the quantities pertaining to physical validity is a non-trivial task — all quantities need to be computed even when a subset of these quantities are used. In Chapter 5, we solved this problem by rewriting the equations of motion in a manner which allows not only efficient evaluation. but also efficient propagation of differential quantities. Computationally.

when the Newton-Euler equations of motion are stripped down to the bare requirements of physical validity, it is shown that the associated quantities and their gradients can be obtained in time linear in the degrees of freedom of the character.

The cost of an optimization iteration however includes not only providing the optimizer with the quantities that it requires but also the cost of the optimization algorithm. In Chapter 6, using a variant of a widely used nonlinear optimization algorithm, we show that the cost of a single iteration of an optimization problem posed with the constraints of physical validity can be accomplished in time linear in the degrees of freedom of the character.

Although theoretical per-iteration costs are important, the difficulty of the problem posed to any optimizer is paramount to its ability to converge. The difficulty of an optimization problem has several aspects, of which the topological irregularity of the functions due to nonlinearity is a major factor. We note that our set of physical validity equations is a subset of the complete Newton-Euler equations and therefore in terms of nonlinearity cannot be worse than similar techniques which employ complete physics. On the other hand, the fact that the gradient of the physical validity constraints may be obtained in linear time instead of quadratic time should hint that our constraint manifold is more regular than constraints associated with the full physics model. At this time, however, we can offer no formal proof to this statement; only the intuition.

Taken in the context of the present state-of-the-art in computer animation, this thesis bridges the gap between the techniques that do not consider physics, and techniques that require computationally expensive physics. The former techniques include all present interactive motion editing techniques, while the latter are predominantly offline techniques, including those from the field of optimal control. In terms of the theoretical order of complexity, the physical validity functions are no more expensive than the kinematic functions used in the interactive motion-editing systems. It goes to reason that the methods described in this thesis may provide a lead to incorporating physics optimization into mainstream character animation systems.

8.2 Shortcomings of Approach

Our approach is targeted at self-actuated articulated figures, such as humanlike models. Specifically, the time savings draw from the fact that there are more self-actuated joints than unactuated ones. The only unactuated degrees of freedom in a "healthy" human model are the 6 degrees of freedom that represent whole body movement — three linear and three rotational degrees of freedom, forming a 6-DOF joint at the base of the articulation tree.

The method can be extended to support articulations where there are more than one unactuated joint, or when a select set of joint torques are important to the motion and need to be constrained. This is done by applying the linear-time derivative computations to sub-trees of the articulation (eg. at the shoulder joint). Complexity-wise, the algorithmic order grows linearly with the number of joints torques whose derivative is required. When the derivatives of all joint torques are required (eg. a chain of passive rigid bodies), our method loses its computational advantage and we revert to a quadratic time complexity.

A direct consequence of this observation is that, unlike the original spacetime constraints method. our approach is not suitable for the synthesis of passive body motions, such as a falling serial chain of rigid bodies. Because our method assumes actuated internal joints. the natural whipping motion seen in a downward swing would be subdued. Simulation of passive body motion. however. is much more easily done and more efficient using forward simulation.

Relatively static motion. or actions where one expects minimal energy movements. is not suitable with this technique. An example is "stand for 5 seconds". The minimal energy solution is a static pose where limbs are relaxed and pointing down. A kinematic objective function with physical validity constraints results in some static pose near the initial guess whose projection of the center of mass is in the support area. If the minimal energy pose were to be expected. then energy-based optimization is more appropriate. On the other hand, if some guess of the desired motion is to be provided by an animator. our approach should provide results which are closer to the animator's expectation.

8.3 Future Work

The following lists areas which our contributions may lead to:

Physically based real-time motion editing. In the present state of the art. real-time editing of motion data that incorporates physics considerations has yet to be demonstrated. Even when a character is drastically abstracted, optimization of edits are still in the order of minutes. The order of computational savings proposed in this thesis may contribute towards realizing such a system.

One way to ensure that an interactively changed motion remains physically valid is to augment present kinematic editing systems, such as those described by Gleicher [18] or Lee and Shin [41]. with the constraints proposed in this thesis. Our method has the same theoretical complexity as interactive kinematic motion editing techniques (See Chapter 2). Although our examples are synthesized in the order of minutes. it is important to note that we start from initial guesses which are very far from the solution of the optimization. Often. in interactive motion editing. the differential changes to a motion are small, localized, and intermediate results of the optimization iterations are meaningful and rendered to the user. That means that fast iterations and the responsiveness of system are important in interactive systems. It remains to be seen whether the per-iteration efficiency that we have achieved, coupled with future hardware advances, would allow physically based interactive editing of the motion of complex characters.

Non-intrusive physically based keyframing. Animators often need absolute control over the creative process. Constructing an animation is often a refinement process, rather than the declarative one that spacetime methods require. Futher, it is arguable whether energy optimality is justified in a character animation system, since energy optimal motion is not often intuitive or even desirable. For instance, Luxo's minimal energy pose is one where all links are aligned in an upright manner. A similar stiff minimal energy pose is present in humans. It will be rather frustrating and intrusive if the character insists on energy minimal poses while the animator is still refining his work. Our system does not have this effect, since a crouching Luxo is equally physically valid. A system that non-intrusively interpolates keyframes while maintaining physical validity may be a useful tool for the animator.

One way that optimization can be non-intrusively introduced into an animation system is to indicate, via some visual cue, parts of the motion that are not physically correct and needs to be refined. For instance, the visual cue could be a color bar above the animation timeline — physically invalid periods could be painted red along the color bar and green otherwise. The animator may then choose to refine the problematic parts of the motion either manually or via optimization (the color bar may have widgets to control the width of the motion to be optimized). The measures of physically incorrectness are exactly the degree of violations of the constraints described in Chapter 4.

Physically based motion synthesis for virtual avatars. Online optimal control is already an enabling technology in several engineering and robotics areas. In animating virtual characters, we have the advantage that motion results need not be critically optimal — plausibility suffices. Given this relaxed requirement, a fast physically based synthesis algorithm could allow avatars to move and react in a realistic manner.

Online systems impose an additional time critical constraint on optimization systems — whereas offline optimization may take an inordinate amount of time, online systems must respond within a certain time quota. One way to apply our approach to online systems is to maintain a database of motion that evolves in realism over time. As a character chooses to perform some sequence of motion (eg. a sequence of footprints), a fast database lookup [40] is performed to find some suitable approximation of the motion (if no suitable approximation is found, we can start from scratch). This approximation may then be refined via optimization within the time quota, and the refined result may in turn be stored back into the database. The next time that this particular motion is recalled, we will have a better (in terms of physical validity) approximation to work with. The refinement process may be repeated such that as the database evolves, we have a larger collection of motions with progressively improved degrees of realism.

Stylistic metrics. Many typical motions are not energy optimal. In many instances, style, expressiveness, and performance play equally important roles. Completing the task or telling the story is almost always more important than its energy consumption. In a dance motion, for instance, elegance or charm is desirable. In gymnastics and platform diving, strict and clean movements score high marks. However, physical validity is still important. Stylistic metrics have been given little attention thus far. The results of this work allow experimentation with stylistic objective metrics while maintaining physical validity.

Accurate physics versus approximate physics. Our approach focuses on validating physics without simplifying the underlying articulation. However, the same computational savings may be reaped on a simplified model. When approximate physics is acceptable, the two techniques can be combined and should result in significant computation savings.

Equivalence with Lagrangian spacetime formulation. As a purely theoretical endeavor, it will be interesting to find an equivalence or duality between our Newton-Euler force-based physics constraints and the Lagrangian energy-based physics constraints adopted in past spacetime papers. Given that the equivalence of Lagrangian dynamics and Newton-Euler dynamics has been proven in the robotics, it should be possible to show an equivalence for spacetime physics constraints as well.

Appendix A

Spatial Notation

Spatial notation is a convenient representation for expressing equations of multibody systems. Just like 3-vectors is for representing differential motion of points, a 6-vector is the natural primitive for representing differential motion of rigid bodies. Most operators for points and vectors, such as transformations, cross and dot products, has correspondences in spatial notation. The object is to suitably define these operators, and in doing so be able to manipulate compact expressions for rigid body motions in ways analogous to the more familiar 3-vectors. This section summarizes the basic concepts and operators of spatial notations. Further details about spatial notations and its applications may be found in [14]; its use dates back to [5].

Conventions. Where subscripted, e.g. v'_i, the quantity is ascribed to link i; where primed, the quantity is relative to its parent link, otherwise it is an absolute quantity. All quantities, relative or absolute, are expressed at the subscripted frame. Where underscored, the term refers to a 3-vector quantity; where italicized, it is scalar. Capitalized quantities are 6×6 matrices unless underscored, in which case they are 3×3.

Spatial vectors. The vector primitive in spatial notation is a 6-vector of the following form:

$$\mathrm{s}'_i = \begin{bmatrix} \underline{\mathrm{s}}'_i \\ \underline{\mathrm{r}}'_i \times \underline{\mathrm{s}}'_i \end{bmatrix} \tag{A.1}$$

where the top 3-vector is translation invariant, and the bottom is dependent on the frame of reference.

Twists and wrenches. A spatial vector may be a twist or a wrench. A twist is a differential rigid body motion, whereas a wrench is a force (or impulse) applied to a rigid body. In a twist, the amount of rotation is translation invariant, therefore the top part of the twist vector refers to a rotational velocity about the frame of reference while the bottom part refers to a linear velocity. In a wrench, the linear force is translation invariant, therefore the top part refers to a linear force applied to the rigid body, while the bottom part refers to a torque about the frame of reference. Notice that the linear and rotation parts are switched in the twist and wrench representation. The purpose is so that we may define meaningful operators that apply to both types of spatial vectors. Twists and wrenches are not distinguished in notation, but its type should be clear from context. In terms of notation, spatial acceleration is a twist-type quantity; spatial momentum is a wrench-type quantity.

Joint axes. A 1-DOF rotational joint axis (or revolute joint) is a twist vector about its joint.

$$\mathrm{s}'_i = \begin{bmatrix} \underline{\mathrm{s}}'_i \\ 0 \end{bmatrix} \tag{A.2}$$

where $\underline{\mathrm{s}}'_i$ is its axis of rotation. A 1-DOF prismatic (translational) joint axis may be written,

$$\mathrm{s}'_i = \begin{bmatrix} 0 \\ \underline{\mathrm{s}}'_i \end{bmatrix} \tag{A.3}$$

where $\underline{\mathrm{s}}'_i$ is its axis of translation. Joints with more than one DOF can be represented as sequences of single DOF joints connected by massless bodies.

Link velocities and accelerations. If the single DOF joint i is parameterized by q_i, its relative velocity and acceleration are,

$$\mathrm{v}'_i = \dot{q}_i \mathrm{s}'_i \tag{A.4}$$

$$\mathrm{a}'_i = \ddot{q}_i \mathrm{s}'_i \tag{A.5}$$

Spatial cross products. The ordinary cross product between 3-vectors may be written in matrix form:

$$\underline{\mathrm{v}} \times \underline{\mathrm{w}} = \begin{bmatrix} \underline{\mathrm{v}}^x \\ \underline{\mathrm{v}}^y \\ \underline{\mathrm{v}}^z \end{bmatrix} \times \underline{\mathrm{w}} = \begin{bmatrix} 0 & -\underline{\mathrm{v}}^z & \underline{\mathrm{v}}^y \\ \underline{\mathrm{v}}^z & 0 & -\underline{\mathrm{v}}^x \\ -\underline{\mathrm{v}}^y & \underline{\mathrm{v}}^x & 0 \end{bmatrix} \underline{\mathrm{w}} = \underline{\tilde{\mathrm{v}}\mathrm{w}} \tag{A.6}$$

where $\underline{\tilde{v}}$ is known as the skew-symmetric matrix of $\underline{v}$. The spatial cross product between spatial vectors v and w is defined similarly:

$$\mathrm{v} \times \mathrm{w} = \begin{bmatrix} \underline{v}_a \\ \underline{v}_b \end{bmatrix} \times \mathrm{w} = \begin{bmatrix} \underline{\tilde{v}}_a & 0 \\ \underline{\tilde{v}}_b & \underline{\tilde{v}}_a \end{bmatrix} \mathrm{w} = \tilde{\mathrm{v}}\mathrm{w} \tag{A.7}$$

Rigid body transforms. Spatial transformations are in general 6 x 6 matrices. Rigid body transformations however have a special form, known as the Special Euclidean form, or $SE(3)$. The rigid body transformation that takes quantities from joint i to its parent frame may be written as:

$$\mathrm{X}_i^{i-1} = \begin{bmatrix} \underline{R}'_i & 0 \\ \underline{\tilde{r}}'_i \underline{R}'_i & \underline{R}'_i \end{bmatrix} \tag{A.8}$$

where $\underline{R}'_i$ is the 3×3 rotation matrix at joint i, and $\underline{r}'_i$ is the vector from its parent joint. Transforms to other joint frames may be composed using the usual matrix multiplication rules: the composited transform also lies in $SE(3)$.

Parameterized joint transforms — rotation. Let a revolute joint i be parameterized by q_i, then the sub-matrices in the spatial transform at joint i is parameterized as follows:

$$\mathrm{X}_i^{i-1}(\underline{s}'_i, q_i) = \begin{bmatrix} \underline{R}'_i(\underline{s}'_i, q_i) & 0 \\ \underline{\tilde{r}}'_i \underline{R}'_i(\underline{s}'_i, q_i) & \underline{R}'_i(\underline{s}'_i, q_i) \end{bmatrix} \tag{A.9}$$

where

$$\underline{R}'_i(\underline{s}'_i, q_i) = 2 \begin{bmatrix} \frac{1}{2} - yy - zz & xy - wz & xz + wy \\ xy + wz & \frac{1}{2} - xx - zz & yz - wx \\ xz - wy & yz + wx & \frac{1}{2} - xx - yy \end{bmatrix} \tag{A.10}$$

which is the matrix form of the quaternion:

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} \sin(\frac{q_i}{2})\underline{s}'^x_i \\ \sin(\frac{q_i}{2})\underline{s}'^y_i \\ \sin(\frac{q_i}{2})\underline{s}'^z_i \\ \cos(\frac{q_i}{2}) \end{bmatrix} \tag{A.11}$$

In general, the parameters associated with the transforms are dropped: it should however be clear from the sub- and super-scripts what the parameters are.

Parameterized joint transforms — translation. Let a prismatic joint i be parameterized by q_i. The spatial transform at joint i is:

$$X_i^{i-1} = \begin{bmatrix} 1 & 0 \\ \underline{r}_i' + q_i \underline{s}_i' & 1 \end{bmatrix} \tag{A.12}$$

Spatial inertias. Rigid body spatial inertias represents both body mass and rotational inertia. The inertia of a body about its center of mass may be written as:

$$I_i^c = \begin{bmatrix} 0 & m_i \underline{E} \\ \underline{I}_i^c & 0 \end{bmatrix} \tag{A.13}$$

where m_i is the mass of body i, $\underline{E}$ is the 3×3 identity matrix, and $\underline{I}_i^c$ is the symmetric 3×3 rotational inertia of body i about its center of mass expressed as:

$$\underline{I}_i^c = \begin{bmatrix} \underline{I}_i^{xx} & 0 & 0 \\ 0 & \underline{I}_i^{yy} & 0 \\ 0 & 0 & \underline{I}_i^{zz} \end{bmatrix} \tag{A.14}$$

where $[\underline{I}_i^{xx} \; \underline{I}_i^{yy} \; \underline{I}_i^{zz}]$ is known as the principle moments of inertia.

Transformation of spatial inertias. Though they appear and act as 6×6 matrices, spatial inertias are in fact second rank tensors, whose geometric properties resemble more that of a vector than a linear transform (vectors are first-rank tensors while scalars are zero-rank tensors). The important property of tensors is that they should be transformed in such a way as to preserve its geometrical or physical meaning. In our case, the inertia tensor of a body is defined at its center of mass. Intuitively, it should be transformed in a way such that the transformed inertia still acts at the center of mass location where it was defined. The spatial inertia of body i transformed to frame j is therefore:

$$I_j^i = X_i^j I_i X_j^i \tag{A.15}$$

Geometrically, an inertia/vector multiplication at frame j brings the given vector back to frame i, where the inertia acts, and returns the result to frame j. Expanding the product relates a translated inertia to its center of mass:

$$I_i = \begin{bmatrix} -m_i \tilde{\underline{c}}_i & m_i \underline{E} \\ \underline{I}_i^c + m_i \underline{c}_i \underline{c}_i^T - \underline{c}_i^T \underline{c}_i \underline{E} & m_i \tilde{\underline{c}}_i \end{bmatrix} \tag{A.16}$$

where $\underline{c}_i$ denotes the 3-vector from the reference frame to the center of mass. The lower-left quadrant remains a 3×3 symmetric matrix after transformation. The representation also suggests that a conservative way to manage the inertia matrix is to only store the unique elements in the inertia matrix. In general, we use the following form:

$$\mathrm{I}_i = \begin{bmatrix} -m_i\tilde{\underline{c}}_i & m_i\underline{E} \\ \begin{bmatrix} \underline{I}_i^{xx} & \underline{I}_i^{xy} & \underline{I}_i^{xz} \\ \underline{I}_i^{xy} & \underline{I}_i^{yy} & \underline{I}_i^{yz} \\ \underline{I}_i^{xz} & \underline{I}_i^{yz} & \underline{I}_i^{zz} \end{bmatrix} & m_i\tilde{\underline{c}}_i \end{bmatrix} \tag{A.17}$$

where 10 unique values are stored instead of 36 in the full 6×6 matrix. [also savings in matrix/vector multiply]

Transpose. The final basic operator is the transpose of a spatial vector defined as:

$$\mathrm{v}^T = \begin{bmatrix} \underline{v}_a \\ \underline{v}_b \end{bmatrix}^T = \begin{bmatrix} \underline{v}_b^T & \underline{v}_a^T \end{bmatrix} \tag{A.18}$$

where $\underline{v}_a^T$ denotes the ordinary 3-vector transpose. Note that the transpose swaps the order of its sub-parts. As such, transpose operations is only meaningful for dot products between a twist-type vector and a wrench-type vector.

Closure. Like 3-vectors, the space of spatial vectors is closed under vector addition, scalar multiplications, spatial transforms, cross products, and its linear combinations. The space of spatial transforms is closed under multiplication, and spatial inertias closed under addition and tensor transformations.

Time derivatives — Forward transform. In 3-space, let a 3-vector $\underline{v}_i$ be local to a frame rotating under $\underline{R}_i'(\underline{s}_i', q_i)$. The rotated vector is thus the product $\underline{R}_i'(\underline{s}_i', q_i)\underline{v}_i$. The rate of change of $\underline{v}_i$ is obtained by differentiating the product [derive this]:

$$\frac{d}{dt}\underline{R}_i'\underline{v}_i = \underline{R}_i'(\dot{q}_i\underline{s}_i' \times \underline{v}_i + \frac{d\underline{v}_i}{dt}) \tag{A.19}$$

where the parameters to $\underline{R}_i'$ is dropped for clarity. The time derivative of a spatial vector has the analogous form,

$$\frac{d}{dt}\mathrm{X}_i^{i-1}\mathrm{v}_i = \mathrm{X}_i^{i-1}(\dot{q}_i\mathrm{s}_i' \times \mathrm{v}_i + \frac{d\mathrm{v}_i}{dt}) \tag{A.20}$$

Time derivatives — Inverse transform. Let v_{i-1} be a spatial vector defined in the parent frame. The rate of change of the vector transformed to the ith frame, $X^i_{i-1}v_{i-1}$, is derived as follows:

$$\frac{d}{dt}X^i_{i-1}v_{i-1} = X^i_{i-1}\frac{dv_{i-1}}{dt} + (X^i_{i-1}v_{i-1}) \times \dot{q}_i s'_i \tag{A.21}$$

Time derivatives — spatial inertia. Let I_i be transformed to its parent frame as $I^i_{i-1} = X^{i-1}_i I_i X^i_{i-1}$. The time derivative of the transformed inertia has the following form:

$$\begin{aligned}
\frac{d}{dt}I^i_{i-1} &= \frac{d}{dt}(X^{i-1}_i I_i X^i_{i-1}) \\
&= X^{i-1}_i(\tilde{s}'_i\dot{q}_i I_i - I_i\tilde{s}'_i\dot{q}_i)X^i_{i-1} \\
&= X^{i-1}_i(\tilde{v}'_i I_i - I_i\tilde{v}'_i)X^i_{i-1} \\
&= X^{i-1}_i \begin{bmatrix} -m_i\dot{\tilde{\underline{c}}}_i & 0 \\ \begin{bmatrix} \dot{\underline{I}}^{xx}_i & \dot{\underline{I}}^{xy}_i & \dot{\underline{I}}^{xz}_i \\ \dot{\underline{I}}^{xy}_i & \dot{\underline{I}}^{yy}_i & \dot{\underline{I}}^{yz}_i \\ \dot{\underline{I}}^{xz}_i & \dot{\underline{I}}^{yz}_i & \dot{\underline{I}}^{zz}_i \end{bmatrix} & m_i\dot{\tilde{\underline{c}}}_i \end{bmatrix} X^i_{i-1}
\end{aligned} \tag{A.22}$$

Appendix B

Equations of Motion

Conventions. In addition to those mentioned in the previous appendix, we will assume in this section a serial chain comprising of L single-DOF links numbered $1, 2, \ldots, L$. The subscript zero is reserved for the quantities representing the entire multibody. Where superscripted with an asterix, e.g. I_i^*, the quantity represent aggregated information accumulated from L to i.

Newton-Euler equations of motion. The equations of motion of a serial multibody chain is compactly expressed in the following recursive inverse dynamics form using spatial notation:

$$\mathrm{v}_i = \mathrm{X}_{i-1}^i \mathrm{v}_{i-1} + \mathrm{s}_i' \dot{q}_i \tag{B.1}$$

$$\mathrm{a}_i = \mathrm{X}_{i-1}^i \mathrm{a}_{i-1} + \mathrm{s}_i' \ddot{q}_i + \mathrm{v}_i \times \mathrm{s}_i' \dot{q}_i \tag{B.2}$$

$$\mathrm{p}_i = \mathrm{X}_{i+1}^i \mathrm{p}_{i+1} + \mathrm{I}_i' \mathrm{v}_i \tag{B.3}$$

$$\mathrm{f}_i = \mathrm{X}_{i+1}^i \mathrm{f}_{i+1} + \mathrm{I}_i' \mathrm{a}_i + \mathrm{v}_i \times \mathrm{I}_i' \mathrm{v}_i \tag{B.4}$$

where the second and fourth equations are the time derivatives of the first and third equations respectively (see Appendix A). For a multibody rooted at its base joint, the following end conditions for simulating gravity is used:

$$\mathrm{a}_0 = -G\ , \quad \mathrm{v}_0 = \mathrm{p}_{L+1} = \mathrm{f}_{L+1} = 0 \tag{B.5}$$

Joint torques is obtained by projecting joint forces onto joint axes as follows:

$$\tau_i = \mathrm{s}_i'^T \mathrm{f}_i \tag{B.6}$$

The Newton-Euler equations propagates quantities in two directions: kinematic quantities are propagated from base to the leaf, and dynamic quantities from leaf to the base. In order to efficiently compute aggregate quantities and their derivatives, it is useful to rewrite the dynamic equations such that propagation only occurs in one direction, from leaf to base. Intuitively this uncouples the equations in a way that joint parameters preceding a given joint does not affect quantities computed at the joint. The following sections describe these aggregate equations.

Aggregate equations — momentum. Compute:

$$\begin{aligned} \mathrm{I}_i^* &= \mathrm{X}_{i+1}^i \mathrm{I}_{i+1}^* \mathrm{X}_i^{i+1} + \mathrm{I}_i' && \text{(B.7)} \\ \mathrm{p}_i^* &= \mathrm{X}_{i+1}^i \mathrm{p}_{i+1}^* + \mathrm{I}_i^* \mathrm{s}_i' q_i && \text{(B.8)} \end{aligned}$$

where body inertias and momenta propagate from the leaf to the base. I_0^* and p_0^* is the aggregate inertia and momentum of the whole body, and p_0^* is equal to p_i computed from the previous Newton-Euler recursive equations. However, p_i^* is in general not equal to p_i where $i \neq 0$ and should only be used as intermediary quantities in computing the aggregates.

Aggregate equations — force. Compute:

$$\begin{aligned} \dot{\mathrm{I}}_i^* &= \mathrm{X}_{i+1}^i \dot{\mathrm{I}}_{i+1}^* \mathrm{X}_i^{i+1} + (\bar{\mathrm{s}}_i' \mathrm{I}_i^* - \mathrm{I}_i^* \bar{\mathrm{s}}_i') \dot{q}_i && \text{(B.9)} \\ \mathrm{f}_i^* &= \mathrm{X}_{i+1}^i \mathrm{f}_{i+1}^* + \mathrm{s}_i' \dot{q}_i \times \mathrm{p}_i^* + \mathrm{I}_i^* \mathrm{s}_i' \ddot{q}_i + \dot{\mathrm{I}}_i^* \mathrm{s}_i' \dot{q}_i && \text{(B.10)} \end{aligned}$$

where f_0^* is the aggregate force on the body whole, equal to f_0 computed previously. As before, f_i^* is in general not equal to f_i where $i \neq 0$.

First derivatives – Aggregate equations. All partial derivatives are in frame i.

$$\begin{aligned} \frac{\partial \mathrm{I}_0^*}{\partial q_i} &= (\bar{\mathrm{s}}_i' \mathrm{I}_i^* - \mathrm{I}_i^* \bar{\mathrm{s}}_i') && \text{(B.11)} \\ \frac{\partial \mathrm{p}_0^*}{\partial \dot{q}_i} &= \mathrm{I}_i^* \mathrm{s}_i' && \text{(B.12)} \\ \frac{\partial \mathrm{p}_0^*}{\partial q_i} &= \frac{\partial \mathrm{I}_0^*}{\partial q_i} \mathrm{v}_{i-1}^i + \mathrm{s}_i' \times \mathrm{p}_i^* && \text{(B.13)} \\ \frac{\partial \mathrm{f}_0^*}{\partial \ddot{q}_i} &= \frac{\partial \mathrm{p}_0^*}{\partial \dot{q}_i} && \text{(B.14)} \\ \frac{\partial \mathrm{f}_0^*}{\partial \dot{q}_i} &= \frac{\partial \mathrm{p}_0^*}{\partial q_i} + \dot{\mathrm{I}}_i^* \mathrm{s}_i' + \mathrm{v}_i \times \frac{\partial \mathrm{p}_0^*}{\partial \dot{q}_i} && \text{(B.15)} \end{aligned}$$

$$
\begin{aligned}
\frac{\partial \mathrm{f}_0^*}{\partial q_i} &= \mathrm{s}_i' \hat{\times} \mathrm{f}_i^* \\
&+ \frac{\partial \mathrm{I}_0^*}{\partial q_i} \mathrm{a}_{i-1}^i \\
&+ \mathrm{v}_{i-1}^i \hat{\times} (\mathrm{s}_i' \hat{\times} \mathrm{p}_i^* + \frac{\partial \mathrm{I}_0^*}{\partial q_i} \mathrm{v}_{i-1}^i) \\
&+ (\tilde{\mathrm{s}}_i' \dot{\mathrm{I}}_i^* - \dot{\mathrm{I}}_i^* \tilde{\mathrm{s}}_i') \mathrm{v}_{i-1}^i
\end{aligned}
\tag{B.16}
$$

where

$$\mathrm{v}_{i-1}^i = (\mathrm{X}_{i-1}^i \mathrm{v}_{i-1}) \tag{B.17}$$

$$\mathrm{a}_{i-1}^i = (\mathrm{X}_{i-1}^i \mathrm{a}_{i-1}) \tag{B.18}$$

Bibliography

[1] C. Atkeson and J. Hollerbach. Kinematic features of unrestrained vertical arm movements. *Journal of Neuroscience*, 5:2318–2330, 1985.

[2] J. Auslander, A. Fukunaga, H. Partovi, J. Christensen, L. Hsu, P. Reiss, A. Shuman, J. Marks, and J. Ngo. Further experience with controller-based automatic motion synthesis for articulated figures. *ACM Transactions on Graphics*, 14(4):311–336, Oct. 1995.

[3] N. I. Badler, C. B. Phillips, and B. L. Webber. *Simulating Humans: Computer Graphics Animation and Control.* Oxford University Press, New York, 1993.

[4] S. Baek, S. Lee, and G. J. Kim. Motion evaluation for vr-based motion training. *EUROGRAPHICS 2001*, 20:3, 2001.

[5] R. S. Ball. *A Treatise on the Theory of Screws.* Cambridge Univ Press, 1900.

[6] N. A. Bernstein. *The co-ordination and regulation of movements.* Pergamon Press, Oxford, 1967.

[7] L. S. Brotman and A. N. Netravali. Motion interpolation by optimal control. In J. Dill, editor, *Computer Graphics (SIGGRAPH 88 Proceedings)*, volume 22, pages 309–315, Aug. 1988.

[8] A. Bruderlin and T. W. Calvert. Goal-directed, dynamic animation of human walking. In *Computer Graphics (SIGGRAPH 89 Proceedings)*, volume 23, pages 233–242, July 1989.

[9] A. Bruderlin and L. Williams. Motion signal processing. In *SIGGRAPH 95 Proceedings*, Annual Conference Series, pages 97–104. ACM SIGGRAPH, Addison Wesley, Aug. 1995.

[10] K. Choi and H. Ko. On-line motion retargeting. *Proceedings of the International Pacific Graphics*, pages 32–42, 1999.

[11] M. F. Cohen. Interactive spacetime control for animation. In E. E. Catmull, editor, *Computer Graphics (SIGGRAPH 92 Proceedings)*, volume 26, pages 293–302, July 1992.

[12] A. R. Conn, N. I. M. Gould, and P. L. Toint. *LANCELOT: a Fortran package for large-scale nonlinear optimization (Release A)*. Number 17 in Springer Series in Computational Mathematics. Springer Verlag, Heidelberg, Berlin, New York. 1992.

[13] P. Faloutsos, M. van de Panne, and D. Terzopoulos. Composable controllers for physics-based character animation. In *SIGGRAPH 01 Proceedings*, Annual Conference Series. ACM SIGGRAPH, ACM Press, Aug. 2001.

[14] R. Featherstone. *Robot Dynamics Algorithms*. Kluwer Academic Publishers. Boston. MA. 1987.

[15] J. Flanagan and A. Rao. Trajectory adaptation to a nonlinear visuo-motor transformation: Evidence of motion planning in visually perceived space. *Journal of Neurophysiology*. 5:2174–2178, 1995.

[16] R. Fletcher. Conjugate gradient methods for indefinite systems. In *Lecture Notes in Mathematics Volume 506*, pages 73–89, 1975.

[17] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Academic Press. New York. 1981.

[18] M. Gleicher. Motion editing with spacetime constraints. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, pages 139–148, Providence, RI. Apr. 1997.

[19] M. Gleicher. Retargetting motion to new characters. In *SIGGRAPH 98 Proceedings*. Annual Conference Series. ACM SIGGRAPH. ACM Press, Aug. 1998.

[20] M. Gleicher. Comparative analysis of constraint-based motion editing methods. In *International Workshop on Human Modeling and Animation*. June 28–29, 2000.

[21] M. Gleicher. Comparing constraint-based motion editing methods. *Graphical models*. 63(2):107–134, 2001.

[22] M. Gleicher. Motion path editing. In *Symposium on Interactive 3D Graphics*. pages 195–202. 2001.

[23] K. Harai, M. Hirose, Y. Haikawa, and T. Takenaka. The development of the honda humanoid robot. In *Proc. IEEE Intl. Conference on Robotics and Automation*. 1998.

[24] C. M. Harris and D. M. Wolpert. Signal-dependent noise determines motor planning. *Nature*. 394:780–784, 1989.

[25] J. K. Hodgins. Three-dimensional human running. In *Proc. IEEE Intl. Conference on Robotics and Automation*, 1996.

[26] J. K. Hodgins and N. S. Pollard. Adapting simulated behaviors for new characters. In *SIGGRAPH 97 Proceedings*, pages 153–162. ACM SIGGRAPH, Addison Wesley, 1997.

[27] J. K. Hodgins and N. S. Pollard. Adapting behaviors to new environments, characters, and tasks. In *Yale Workshop on Adaptive and Learning Systems*, 1998.

[28] J. K. Hodgins and M. H. Raibert. Biped gymnastics. *International Journal of Robotics Research*, 9(2):115–132, 1990.

[29] J. K. Hodgins, W. L. Wooten, D. C. Brogan, and J. F. O'Brien. Animating human athletics. In *SIGGRAPH 95 Proceedings*, Annual Conference Series, pages 71–78. ACM SIGGRAPH. Addison Wesley, Aug. 1995.

[30] N. Hogan. An organizing principle for a class of voluntary movements. *Journal of Neuroscience*, 4:2745–2754, 1984.

[31] J. Hollerbach and C. Atkeson. Deducing planning variables from experimental arm trajectories: pitfalls and possibilities. *Biological Cybernetics*, 56:279–292, 1987.

[32] Honda Motor Co. Ltd. http://world.honda.com/robot/.

[33] M. Jordan and D. Rumelhard. Forward models: Supervised learning with a distal teacher. *Cognitive Science*, 16:307–354, 1992.

[34] M. Kawato. Feedback-error-learning neural network for supervised learning. *Eckmiller, R., editor, Advanced neural computers*, pages 365–372, 1992.

[35] M. J. Kim, M. Kim, and S. Shin. A general construction scheme for unit quaternion curves with simple high order derivatives. In *Compute Graphics (Siggraph 1995 Proceedings)*, pages 369–376, 1995.

[36] H. Ko and N. I. Badler. Straight line walking animation based on kinematic generalization that preserves the original characteristics. In *Proceedings of Graphics Interface '93*, pages 9–16, Toronto, Ontario, Canada, May 1993. Canadian Information Processing Society.

[37] M. Kuperstein. Neural model of adaptive hand-eye coordination for single postures. *Science*, 239:1308–1311, 1988.

[38] J. Lackner and P. DiZio. Rapid adaptation to coriolis force perturbations of arm trajectory. *Journal of Neurophysiology*, 72:1:299–313, 1994.

[39] J. Laszlo, M. van de Panne, and E. Fiume. Limit cycle control and its application to the animation of balancing and walking. In *SIGGRAPH 96 Proceedings*, Annual Conference Series, pages 155–162. ACM SIGGRAPH, ACM Press, Aug. 1996.

[40] J. Lee, J. Chai, P. Reitsma, J. Hodgins, and N. Pollard. Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics (SIGGRAPH 2002)*, 21(3):491–500, 2002.

[41] J. Lee and S. Y. Shin. A hierarchical approach to interactive motion editing for human-like figures. In *SIGGRAPH 99 Proceedings*, Annual Conference Series, pages 39–48. ACM SIGGRAPH, ACM Press, Aug. 1999.

[42] K. Liu and Z. Popović. Synthesis of complex dynamic character motion from simple animations. *ACM Transactions on Graphics (SIGGRAPH 2002)*, 21(3):408–416, 2002.

[43] Z. Liu. Efficient animation techniques balancing both user control and physical realism. In *Ph.D. Thesis*. Princeton University, 1996.

[44] Z. Liu and M. Cohen. Decomposition of linked figure motion: Diving. In *5th Eurographics Workshop on Animation and Simulation*, 1994.

[45] Z. Liu and M. F. Cohen. An efficient symbolic interface to constraint based animation systems. In *6th Eurographics Workshop on Animation and Simulation*, 1995.

[46] Z. Liu and M. F. Cohen. Keyframe motion optimization by relaxing speed and timing. In *6th Eurographics Workshop on Animation and Simulation*, 1995.

[47] Z. Liu, S. J. Gortler, and M. F. Cohen. Hierarchical spacetime control. In *SIGGRAPH 94 Proceedings*, Annual Conference Series, pages 35–42. ACM SIGGRAPH, ACM Press, July 1994.

[48] J. Lo and D. Metaxas. Recursive dynamics and optimal control techniques for human motion planning. In *Proceedings of Computer Animation '99*, pages 220–234, 1999.

[49] J. Marks, M. Cohen, J. T. Ngo, S. Shieber, and J. Snyder. Optimization — an emerging tool in computer graphics. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 483–484. ACM Press, 1994.

[50] W. Miller. Sensor-based control of robotic manipulators using a general learning algorithm. *IEEE Journal of Robotics and Automation*, 3:157–165, 1987.

[51] W. Nelson. Physical principles for economies of skilled movements. *Biol. Cybern.*. 46:135–147. 1983.

[52] J. T. Ngo and J. Marks. Spacetime constraints revisited. In J. T. Kajiya. editor. *Computer Graphics (SIGGRAPH 93 Proceedings)*, volume 27, pages 343–350. Aug. 1993.

[53] S. Pheasant. *Bodyspace: Anthropometry, Ergonomics and Design.* Taylor and Francis. London and Philadelphia, 1986.

[54] N. S. Pollard, J. K. Hodgins, M. J. Riley, and C. G. Atkeson. Adapting human motion for the control of a humanoid robot. In *Proc. IEEE Intl. Conference on Robotics and Automation.* Washington, D.C., May 2002.

[55] N. S. Pollard and P. S. A. Reitsma. Animation of humanlike characters: Dynamic motion filtering with a physically plausible contact model. In *Yale Workshop on Adaptive and Learning Systems*, 2001.

[56] J. Popović. S. M. Seitz. M. Erdmann, Z. Popović, and A. Witkin. Interactive manipulation of rigid body simulations. In *SIGGRAPH 00 Proceedings*, 2000.

[57] Z. Popović and A. Witkin. Physically-based motion transformation. In *SIGGRAPH 99 Proceedings*, Annual Conference Series. ACM SIGGRAPH. ACM Press, Aug. 1999.

[58] W. H. Press. S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C.* Cambridge University Press, New York, 1992.

[59] M. H. Raibert. *Legged robots that balance.* MIT Press, 1986.

[60] M. H. Raibert and J. K. Hodgins. Animation of dynamic legged locomotion. In T. W. Sederberg. editor, *Computer Graphics (SIGGRAPH 91 Proceedings)*, volume 25, pages 349–358. July 1991.

[61] C. F. Rose, B. Guenter, B. Bodenheimer, and M. F. Cohen. Efficient generation of motion transitions using spacetime constraints. In *SIGGRAPH 96 Proceedings*, Annual Conference Series, pages 155–162. ACM SIGGRAPH. Addison Wesley, Aug. 1996.

[62] P. Sabes and M. Jordan. Obstacle avoidance and a perturbation sensitivity model for motor planning. *Journal of Neuroscience*, 7:7119–7128, 1997.

[63] R. Shadmehr and F. Mussa-Ivaldi. Adaptive representation of dynamics during learning of a motor task. *Journal of Neurophysiology*, 14:5:3208–3224. 1994.

[64] H. J. Shin, J. Lee, S. Y. Shin, and M. Gleicher. Computer puppetry: An importance-based approach. *ACM Transactions on Graphics*, 20(2):67–94, 2001.

[65] M. C. Suries. An algorithm with linear complexity for interactive, physically-based modeling of large protein. In *Compute Graphics (Siggraph 1992 Proceedings)*, pages 221–230, 1992.

[66] N. H. Tamar Flash. The co-ordination of arm movements: An experimentally confirmed mathematical model. *Journal of Neuroscience*, 5:1688–1703, 1985.

[67] Y. Uno, M. Kawato, and R. Suzuki. Formation and control of optimal trajectories in human multijoint arm movements: Minimum torque-change model. *Biol. Cybern.*, 61:81–101, 1989.

[68] M. Unuma, K. Anjyo, and R. Takeuchi. Fourier principles for emotion-based human figure animation. In *SIGGRAPH 95 Proceedings*, Annual Conference Series, pages 91–96. ACM SIGGRAPH, Addison Wesley, Aug. 1995.

[69] M. van de Panne and E. Fiume. Sensor-actuator networks. In J. T. Kajiya, editor, *Computer Graphics (SIGGRAPH 93 Proceedings)*, volume 27, pages 335–342, Aug. 1993.

[70] M. van de Panne, E. Fiume, and Z. Vranesic. Reusable motion synthesis using state-space controllers. In F. Baskett, editor, *Computer Graphics (SIGGRAPH 90 Proceedings)*, volume 24, pages 225–234, Aug. 1990.

[71] M. Vukobratović. On the stability of biped locomotion. *IEEE Trans. Biomedical Engineering*, 17(1):25–36, 1970.

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

[73] A. Witkin and M. Kass. Spacetime constraints. In J. Dill, editor, *Computer Graphics (SIGGRAPH 88 Proceedings)*, volume 22, pages 159–168, Aug. 1988.

[74] A. Witkin and Z. Popović. Motion warping. In *SIGGRAPH 95 Proceedings*, Annual Conference Series, pages 105–108. ACM SIGGRAPH, Addison Wesley, Aug. 1995.

[75] D. M. Wolpert. Computational approaches to motor control. *Trends in Cognitive Sciences*, 1:6:209–216, 1997.

[76] D. M. Wolpert, Z. Ghahramani, and M. Jordan. Are arm trajectories planned in kinematic or dynamic coordinates? an adaptation study. *Experimental Brain Research*, 103:460–470, 1995.

[77] W. Wooten. *Simulation of Leaping, Tumbling, Landing, and Balancing Humans.* PhD Thesis. Georgia Institute of Technology, 1999.

[78] W. L. Wooten and J. K. Hodgins. Animation of human diving. *Computer Graphics Forum*, 15(1):3–14, 1996.

[79] W. L. Wooten and J. K. Hodgins. Simulating leaping, tumbling, landing, and balancing humans. In *Proc. IEEE Intl. Conference on Robotics and Automation*, 2000.

[80] K. Yamane and Y. Nakamura. Dynamics filter - concept and implementation of on-line motion generator for human figures. In *Proc. IEEE Intl. Conference on Robotics and Automation*, 2000.

[81] J. Zhao and N. I. Badler. Inverse kinematics positioning using nonlinear programming for highly articulated figures. *ACM Transactions on Graphics*, 13(4):313–336, 1994.