Volume xx (*200y*), *Number z, pp. 1–11*

Pose Controlled Physically-Based Motion

Raanan Fattal[†] Dani Lischinski[‡]

School of Computer Science and Engineering
The Hebrew University of Jerusalem

Abstract

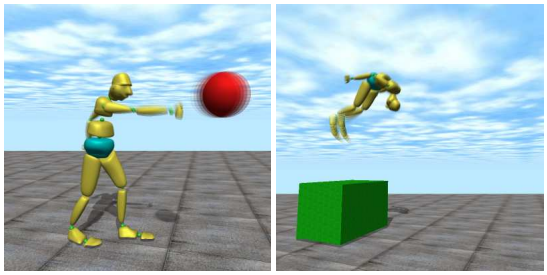
In this paper we describe a new method for generating and controlling physically-realistic motion of complex articulated characters. Our goal is to create motion from scratch, where the animator provides a small amount of input and gets in return a highly detailed and physically plausible motion. Our method relieves the animator from the burden of enforcing physical plausibility, but at the same time provides full control over the internal DOFs of the articulated character via a familiar interface. Control over the global DOFs is also provided by supporting kinematic constraints. Unconstrained portions of the motion are generated in real time, since the character is driven by joint torques generated by simple feedback controllers. Although kinematic constraints are satisfied using an iterative search (shooting), this process is typically inexpensive, since it only adjusts a few DOFs at a few time instances. The low expense of the optimization, combined with the ability to generate unconstrained motions in real time yields an efficient and practical tool, which is particularly attractive for high inertia motions with a relatively small number of kinematic constraints.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and RealismAnimation

1. Introduction

Generating and controlling physically realistic motion of complex articulated characters is a longstanding grand challenge in computer graphics. In today's animation houses such characters are typically animated by keyframing, which gives animators complete control over each degree of freedom at any point in time. However, enforcing physical plausibility by keyframing alone can be burdensome and tedious even for the most talented experts. Imagine a character punching an object; following the impact between the fist and the object, the reaction force propagates to the entire body of the character, resulting in a variety of subtle movements. Reproducing such effects is crucial for physical realism, but requires many keyframes and much fine tuning.

Physics-based animation tools, which attempt to relieve animators from the burden of manually emulating physics, have received much research attention during the past two decades. Such tools enforce physical realism automatically, via dynamic simulations or physics-based constraints. We review these tools in more detail in the next section. Unfortunately, some tools use expensive optimization and do not scale well to complex characters, while others limit the extent of expressive control that animators have over the resulting motion. Many recent tools also make use of previously captured or animated motions, which are not always available.



This paper presents a new physics-based animation tool.

† e-mail: raananf@cs.huji.ac.il
‡ e-mail: danix@cs.huji.ac.il

This tool enables animators to generate physically realistic motion inexpensively, while designing and controlling it via a familiar and intuitive interface, and without relying on motion libraries.

Our main goal was to allow animators to enjoy the benefits that physics-based simulations have to offer, without relinquishing their ability to manipulate individual degrees of freedom, as this ability is instrumental for fully expressive control.

In other words, we aim to leverage the strengths of both the animator and the computer: the animator should be free to focus on the style and the expressive aspects of the desired motion, while the computer takes care of the physics.

In our approach, each internal degree of freedom (DOF) of the articulated character is assigned a feedback controller. The animator specifies *control poses* for various DOFs at various times, and the feedback controllers continuously generate joint torques that propel each joint towards its next control pose. These torques are fed to a dynamics simulator, which integrates the equations of motion, updating the character at each time step.

Using this simple and inexpensive mechanism animators can generate real-time physically-realistic motion with keyframe-like control over the character's internal DOFs. Note, however, that while in keyframe animation the key frames are interpolated by the trajectories of the corresponding DOFs, our mechanism does not guarantee precise interpolation of control poses. Thus, the control poses should be viewed as handles for specifying and manipulating the motion, rather than as points that the motion curves must interpolate.

Our tool gives animators direct control over the internal DOFs of a character, while the six global DOFs of the character are determined implicitly, as they result from the interaction between the character and its environment via contact, friction, and gravity. For example, during free flight portions of the motion, the global DOFs are completely determined by the linear and angular velocities at the moment of take off. In cases where more direct control over the global DOFs is required, the animator may specify global kinematic constraints, and designate which control poses the system is allowed to modify in order to satisfy these constraints. A shooting strategy is then used to determine the required modifications for the designated poses.

In addition to using feedback controllers for generation of joint torques, we also use them to construct *autonomous objective controllers* (AOCs), responsible for achieving certain simple behaviors. For example, it is quite easy to rig a character to use its arm for keeping a tray horizontal while walking. This is another mechanism that could be used to relieve the animator from explicitly controlling various technical aspects of the motion, if the animator so chooses.

Another contribution of this work is the introduction of non-linear Proportional-Derivative controllers, which are demonstrated to be better suited towards generation of realistic joint motion.

Although the different building blocks of our approach have been used before in the context of physics-based simulation, the

contribution of this work is in providing a comprehensive solution for the problem of creating physically-based motion *from scratch*. This is achieved by integrating non-linear DOF-assigned feedback controllers, a shooting strategy for satisfying kinematic constraints, and autonomous controllers for higher level constraints.

In summary, the main advantages of our approach are:

- The animator is given full expressive control over the internal DOFs of the articulated character via a familiar interface, but is mostly relieved from the burden of enforcing physical plausibility.
- Motions where the global DOFs are not constrained are generated in real time, since they are driven by forces generated by simple feedback controllers. Although shooting is required to satisfy kinematic constraints, these optimizations are typically inexpensive since we only optimize over a small number of DOFs and over short time intervals.
- It is relatively easy to rig characters with controllers for maintaining various objectives.

Our experiments indicate that the proposed approach is best suited towards the generation of inertia-dominant motions, such as motion involving impact due to collisions, acrobatic manouvers, and slow motion sequences, which demand a high amount of physically plausible detail.

A limitation of our approach is that by exposing so much low level control to the animator, the resulting tool is more oriented towards experienced animators, who are able to leverage this kind of control for generation of expressive and lifelike motion. It is, however, not suitable for novice users or for generation of realistic motion based on minimal user input.

2. Related Work

Attempts to use physics for realistic animation of articulated bodies date back to 1985 [AG85, WB85]. Since motion in the real world is the result of forces acting on objects which have shape and mass, it was reasoned that a motion will appear realistic if it is generated according to the physical principles of dynamics. It was soon realized, however, that controlling such simulations is a challenge in its own right.

In their groundbreaking work on *spacetime constraints*, Witkin and Kass [WK88] demonstrated that controlled realistic motion of articulated characters may be generated by solving a constrained optimization problem over the entire

time interval of interest. The animator specifies various kinematic and contact constraints, and the optimization process solves for the minimal energy time varying forces, which result in a motion that satisfies these constraints. Spacetime constraints is a powerful paradigm, which has produced some very compelling results. However, despite several various enhancements [Coh92, LGC94, RGBC96] it does not scale well with the number of DOFs, making it impractical for complex articulated characters, such as humans and animals. Fang and Pollard [FP03] improve the scalability of this paradigm by proposing an efficient method for computing derivatives of a broad range of constraints.

Popović and Witkin [PW99] used the spacetime constraints paradigm to generate realistic motions by transforming existing motions into new ones that satisfy various animator-specified constraints. In this approach the optimization problem is made tractable by reducing the number of DOFs (using a simplified character) and by starting from a good initial guess (the original motion). Safanova *et al.* [SHP03] also perform optimization in a low-dimensional space. In their method, the dimensionality is reduced by representing the motion as a linear combination of six to ten basis vectors extracted from examples of similar motions. Sulejmanpašić and Popović [SP05] describe a method for making adjustments in performed ballistic motions by first fitting a physically consistent motion.

While some very realistic motions were produced using such techniques, it should be noted that all of these techniques are most effective when motions similar to the desired one are already available. This is not the case when designing new fictionary characters (e.g., monsters or aliens), or when the desired motion is unusual (e.g., a dangerous stunt). In addition, example-based motion synthesis methods are typically limited in their ability to generate new poses, and the animator's control over the resulting motion is often limited to high-level constraints. In contrast, our approach does not make any use of existing motion data, and we believe it is particularly well-suited for experimenting with new characters and new motions.

Liu and Popović [LP02] describe a method for generating high-energy motions from animation sketches. Rather than fully simulating physics they produce realistic motion by enforcing a small set of linear and angular momentum constraints. However, this approach has not yet been extended to other types of motion, such as walking.

Another paradigm utilizes simulated feedback controllers for motion generation. Such controllers generate actuator forces as a function of the current state of the articulated body so as to carry out a particular task [RH91, vF93, vKF94]. A variety of controllers have been described that successfully generate specific human motions, such as walking, running, vaulting, cycling, etc. [HWBO95, LvF96]. Some non-human characters have also been animated using controllers [LvF00, Tv98]. Approaches that attempt to automate controller synthesis [HP97] and their composition [FvT01] have begun to emerge. However, designing a controller for a particular high-level task is generally still a difficult process, requiring a considerable amount of fine tuning and precise synchronization. Furthermore, most controller-based systems do not provide the animator with a sufficient degree of expressive control over the resulting motion.

In this work we also make extensive use of feedback controllers. However, unlike in most of the works mentioned above, our controllers do not encapsulate high-level tasks, such as "take a step forward" or "stand up". Instead, each controller is simply responsible for propelling an individual joint towards the animator specified control pose. Thus, in our approach the controllers do not interfere with the animator's ability to control the motion, but simply serve as a mechanism for translating the animator's intent into the joint torques required by the dynamics simulator.

Kokkevis *et al.* [KMB96] describe a physics-based system for guided animation of articulated figures. The system is provided with a set of animated or captured kinematic trajectories for some DOFs, and uses feedback controllers to actuate the corresponding joints and sites to follow these trajectories. Joint actuators produce internal torques, similarly to muscles, but site actuators apply external forces, like strings pulling a marionette. We also use controllers to propel joints towards desired angles, but resolve kinematic constraints in a more physically consistent manner, using a shooting strategy. Zordan and Hodgins [ZH02] also use feedback controllers to make an articulated figure closely track captured motions. This results in a detailed realistic motion, which is able to respond and interact with the environment. More recently, [ZMCF05] utilize feedback controllers to control the motion of a character falling in between two motion capture sequences. In [YCP03], captured data is transformed to a simulated motion in order to allow plausible modifications. The propelling forces are found by solving the inverse dynamics, and are later used to alter linear feedback controllers to better match empirical observations in muscle biomechanics. Unlike these methods, our approach requires neither captured motion, nor complete DOF trajectories as input. In contrast, we generate motion from scratch and control it using a sparse set of poses and kinematic constraints.

Also related is the work by Popović *et al.* [PSE*00] on interactive control of rigid body simulations. In their approach the animator is able to manipulate the trajectories of flying rigid bodies directly, while the system solves for the corresponding initial conditions using shooting. We adopt a similar solution to handle kinematic constraints for self-actuated articulated characters.

Endorphin, a commercial dynamic motion synthesis software [Nat] also enables animators to control dynamically simulated articulated characters. In particular, the animator may specify "active poses", which to the best of our understanding are similar to our control poses. The inner workings

of this system are not publically available, however, judging from the provided demos, this system does not appear to automatically satisfy kinematic constraints. The very existence of this product supports the usefulness of our approach towards the generation and control of motion.

3. Overview

Physically-based simulations of articulated rigid body structures are generated by numerically integrating the equations of motion. At each time step a dynamics simulator is provided with a set of joint torques and some external forces and computes the updated positions and velocities of the articulated bodies. In our current implementation the dynamics simulation is carried out by the Open Dynamics Engine library [Smi04], which is fast, and has built-in handling of constraints, collisions, and friction.

In our system the joint torques required by the dynamics simulator are generated by feedback controllers (FBCs), associated with each angular DOF in the articulated character. To animate the character, the animator specifies a set of control poses (target angles for various joints at various times), and the FBCs determine the proper actuating torques, which are fed to the dynamics simulator at each time step. These actuating torques are determined by the difference between the current and the target state of each joint. Thus, although each controller operates autonomously, their actions are implicitly coupled by the feedback they get at each time step.

Note that the animator is only allowed to specify control poses for the angular DOFs of the character. The global translational DOFs of the character are controlled implicitly: they are completely determined by integrating the equations of motion while accounting for the interaction between the character and its surrounding environment. For example, in order to make a character walk the animator specifies the relative poses that the character should go through in the course of a gait, but it is the contact and the friction between the character's feet and the ground which cause the character to actually move forward.

Thus, in our approach, the animator defines and controls motions by instructing the character *how* to perform them, rather than by keyframing the global state of the character. This way of animating resembles the *straight ahead action* paradigm sometimes used in traditional animation [Las87]. In many cases this way of controlling the motion is sufficient. But there are also other cases, where the animator needs more precise global kinematic control over the resulting motion (specifying footprints, the height of a jump, etc.). To handle these cases we allow the animator to specify global kinematic constraints, and employ a shooting strategy that modifies certain key poses in order to satisfy these constraints.

In the following sections we describe our FBCs and their use in more detail. We then describe our shooting based approach for satisfying global kinematic constraints, and proceed to discuss some examples of autonomous objective controllers.

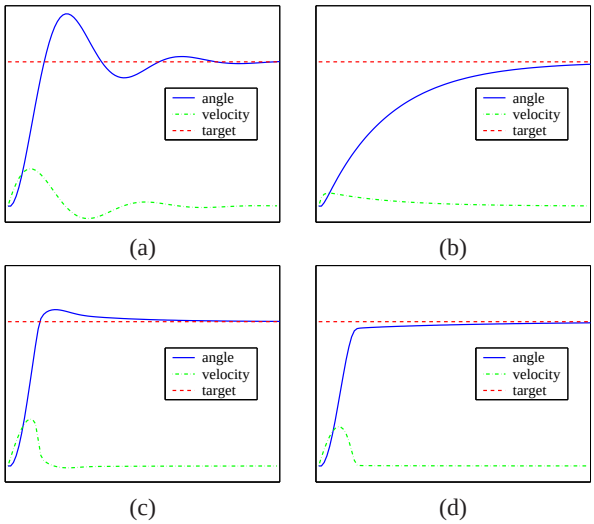


Figure 1: *Angular trajectories and corresponding velocities vs. time generated by differently damped PD controllers. In all plots* $k_s = 0.15$*. (a) low damping,* $k_d = 0.25$ *(b) high damping,* $k_d = 1.5$*. (c) non-linear low damping,* $k_d = 1.5$*; (b) non-linear high damping,* $k_d = 3$*.*

4. PD Feedback Control

Feedback controllers are a standard component in many engineering applications. Such controllers serve to achieve a specified target state, *setpoint*, in a system given its current state. For example, airplane autopilot systems use a FBC that keeps the aircraft horizontal by adjusting the elevators, based on the current orientation measured by a gyroscope.

As mentioned earlier, we use FBCs to determine the joint torques required for propelling the body towards the specified control poses. Consider first a simple setting: a single joint with a single DOF. Given the current joint angle $\theta(t)$ and a target angle θ^*, the torque specified by a *proportional derivative* (PD) feedback controller is given by

$$\tau = k_s \theta_e - k_d \dot{\theta}. \tag{1}$$

Here $\theta_e = \theta^* - \theta$ is the error between the desired and the current state of the joint, k_s is the stiffness coefficient, and k_d is the damping coefficient. The torque τ is then fed into the Euler equation of motion,

$$I\ddot{\theta} = \tau, \tag{2}$$

where I is the moment of inertia.

We found that ordinary PD controllers produce motion

with somewhat undesirable characteristics for our purposes, as demonstrated in Figure 1(a–b): controllers with a small amount of damping cause the system to vibrate around θ^*, failing to imitate a biomechanical joint. The vibrations may be eliminated by increasing damping, but this prevents the formation of inertia, which is also undesirable. In order to understand why this is so, let us rewrite Equations (1) and (2) as

$$\ddot{\theta} = \frac{k_d}{I}(\frac{k_s}{k_d}\theta_e - \dot{\theta}).$$

The resulting equation describes the convergence of $\dot{\theta}$ towards $\frac{k_s}{k_d}\theta_e$ at the rate of k_d/I. For large damping coefficients k_d this means that the velocity $\dot{\theta}$ quickly approaches a small value that changes linearly with the error θ_e. This results in an overly uniform motion. Taking these observations into account we use non-linear PD controllers, in which the damping term is reduced when the system is far away from its target state, thus allowing momentum to build up:

$$\tau = k_s\theta_e - k_d f(\theta_e)\dot{\theta}. \quad (3)$$

The damping reduction function f is defined as follows:

$$f(\theta_e) = \begin{cases} 1 & \text{if } |\theta_e| < \lambda \\ |\frac{\lambda}{\theta_e}|^\alpha & \text{otherwise} \end{cases}$$

Here λ is the characteristic angular error above which we want to reduce the damping force and α determines the rate of reduction. In all of the examples in this paper we use $\lambda = 1$ degree and $\alpha = 1$. The effect of this change results in a less uniform motion, as shown in Figure 1(c–d). For example, the maximal velocity in Figure 1c is three times higher than that in Figure 1b. As demonstrated by the plots (c,d) the amount of overshoot may be regulated by the damping coefficient. In some cases, a small amount of overshoot makes the motion appear more natural and interesting. Different non-linear feedback controllers have also been proposed by Matarić *et al.* [MZW99], where torque drops off at high errors, and by Neff and Fiume [NF02] who decouple stiffness control and position control and account for the effects of external forces.

5. Animating with FBCs

Similarly to standard keyframed CG animation, our system requires the animator to provide it with a set of control poses. Each control pose is a pair (θ_i^{*k}, t^k), where θ_i^{*k} is the angle of the i-th DOF at time t^k. Such poses may be specified using any standard (forward or inverse kinematics) user interface. Each DOF has a dedicated FBC that requires a target function $\theta_i^*(t)$ to guide it. A sample-and-hold approach where $\theta_i^*(t) = \theta_i^{*k}$ for some time interval around t^k causes the controller to produce abrupt transitions between poses: joints quickly move from one pose to the next, and then remain almost motionless until it's time to move again. Thus, to achieve more continuous transitions from one control pose to the next, and for better control over the timing at which each pose is matched, we provide the controllers with a continuous target function defined by linear interpolation between successive control poses.

As noted earlier, the DOF trajectories generated by the FBC driven physics simulation do not, in general, interpolate the control poses. Thus, these poses should be regarded as control points, which are approximated by the motion curves generated by the simulation, rather than points that lie strictly on the curves. The accuracy of the approximation depends on several factors, such as the difference between two successive control poses, the amount of time available for the transition, and the FBC parameters. Allowing these small discrepancies between the control poses and the actual motion curves enables our approach to avoid exhaustive optimization. The overall timing is controlled by the key poses; however, fast transitions between key poses may show a small lag. In general, we found that such discrepancies, resulting from the physical consistency, does not interfere with one's ability to obtain the desired motion. Furthermore, it is well known among animators [Com01, Lan01] that precise keyframe interpolation sometimes results in a mechanical/robotic looking motion. In fact, animators often use *key offsetting* (adding small time shifts) to loosen the motion in the cleanup phase of the animation process.

One should also keep in mind that in a physically consistent environment, the animator must maintain some consistency between the rates at which transitions between control poses occur and the physical competence of the character, such as the masses and lengths of the various parts, and the torque coefficients k_s and k_d for each DOF. For example, a weak (low k_s) joint cannot be expected to perform a large change in its angle over a brief period of time. Thus, before animating an articulated character with our approach, the character must be properly rigged, taking into account the tasks it will be expected to perform. For each of the motions shown in this paper the coefficients remained constant throughout the motion (except for the jumping motion example described in Section 6). Overall, we found the rigging process to be fairly intuitive, with most DOFs being assigned the same coefficients.

As pointed out earlier, the mechanism described so far does not provide explicit control over the global orientation of the character. For example, a character may find itself in a physically unstable pose, and fall because of gravity. Real creatures maintain their balance by exerting a set of delicately coordinated forces and torques by different muscles throughout their bodies. We cannot expect animators to achieve the same effect by adjusting key poses. Instead, similarly to van de Panne and Lamouret [vL95], we introduce external *stabilizing torques* that act on the root of the character's hierarchy to keep it stably balanced. This is done by assigning an FBC to each of the three rotational DOFs of the

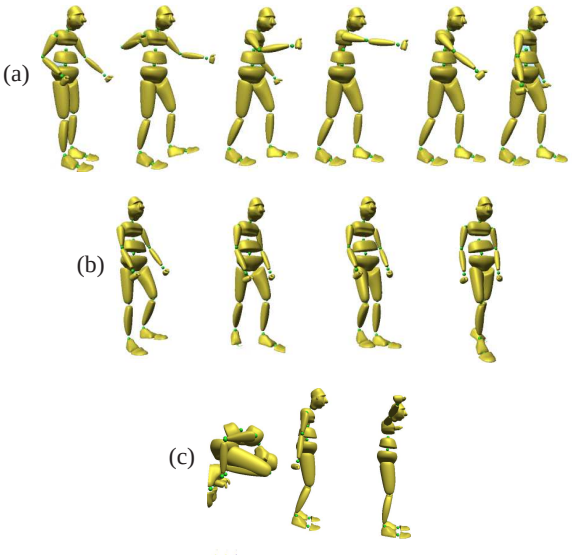


Figure 2: *Control poses used for: (a) punching motion; (b) walk cycle; (c) jump motion.*

root, and providing them with the target angles defined by the control poses (again using linear interpolation).

Even when stabilizing torques are in use, the animator should still make sure that the poses she prescribes for the character look natural and balanced, otherwise the fact that balance is maintained will appear unrealistic. The same is also true in the context of traditional keyframe animation. We found that so long as the character performs on the ground the physical realism of the resulting motion is not compromised by introducing the stabilizing torques. However, in situations where the character is supposed to fall, or during free flight motion, the stabilizing torques should be turned off.

Using our tool we produced several motions of a human-like character with 37 DOFs. It should be noted that all of the motions in this paper were animated by one of the authors, who has no prior animation experience. In the first animation, the character throws a punch at a ball in front of it. This motion required specifying between two and nine DOFs at six different times (see Figure 2a and the accompanying video). The control poses instruct the character to swing at the ball, and then return to its rest position. The collision between the fist and the ball and the resulting secondary motion due to the inertia and the impact are obtained automatically from the physical simulation, without the need to explicitly specify them (see Figure 3a and the video). To illustrate the added value of the physics-based simulation, the video also contains the motion produced by standard (cubic) interpolation of the control poses. As expected, the physics-based motion exhibits much richer, more convincing dynamics, involving the entire body. In this example, the use of the stabilizing torques results in a slightly exaggerated swaying motion after the punch. The motion was simulated in real-time on a 2.4GHz Pentium 4 machine.

In another experiment we generated a walk cycle for the character. Our intention here was to generate a motion of a caricature-like nature. The walk cycle required specifying ten DOFs at only four equally spaced times during the cycle, as shown in Figure 2b and the accompanying video. Walks of arbitrary lengths were generated in real time by repeating the walk cycle control poses as necessary. Although repeating the cycle in this manner results in a periodic sequence of control poses, the resulting walking motion does not repeat itself exactly, and appears much richer and more physically plausible than what could have been obtained with ordinary interpolation from the same set of poses (also shown in the video). For example, although no control poses we set for the shoulders, the arms swing in an anti-symmetric fashion.

Our third example demonstrates the ability of the animator to control motions where interaction and collisions with the environment play an important role. The character receives a strong blow to its right shoulder, causing it to fall. Without any control poses the dynamic simulation produces a physically-realistic fall, and the character lands on its face. To make the fall appear more natural, we first made adjustments to 14 DOFs to make the character attempt to absorb the fall with its arms and avoid getting hit in the head. Next, a similar number of adjustments were made to cause the character to roll on its back after the fall. The resulting three motions were simulated in real time, and are shown in the video. This example clearly shows the advantages of combining a physics-based simulation with low-level DOF control.

6. Matching Kinematic Constraints

The approach described so far provides the animator with good control over the poses of the articulated character in the course of its motion, but does not provide explicit control over the translational DOFs of the root. For example, the animator may cause a character to jump by "instructing" the character to first squash into a pre-jump pose and then quickly transition into a stretched release pose. However, it would clearly be difficult for the animator to specify the squashed and stretched poses accurately enough to make sure that the jump achieves a specific desired height, or make the character perform a somersault in the air.

Thus, it is necessary to augment our method with a mechanism capable of enforcing various kinematic constraints. This cannot be done with artificial external forces, since such forces will blatantly violate the conservation of linear momentum resulting in physically unrealistic motion. External torques, such as the stabilizing torques we use to balance the character, also violate the conservation of angular momentum, but if these torques are sufficiently small the resulting motion remains plausible.

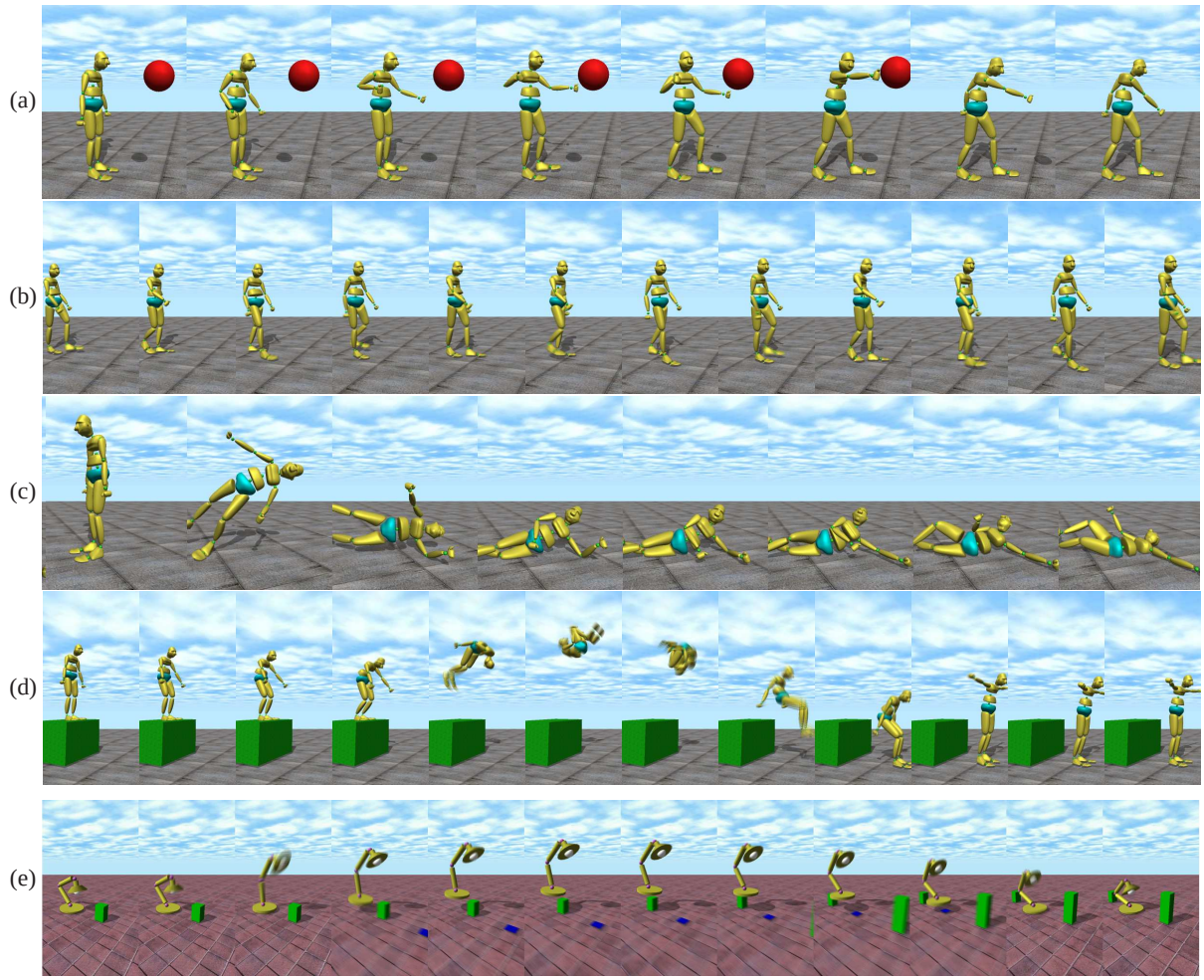


Figure 3: *Thumbnails from some of the motion sequences generated using our method. (a) Punching. (b) Walking. (c) Falling. (d) A jump with two somersaults. (e) A jumping lamp.*

We therefore turn to a shooting-based approach, which is able to provide animators with the desired kinematic control without compromising the physical realism of the resulting motion. Such an approach was successfully used by Popović *et al.* [PSE*00] for controlling rigid body simulations. Let us denote the state of the articulated character with the generalized state vector $\mathbf{q}$. The animator may specify a set of m kinematic constraints:

$$\mathbf{q}_{i_k}(t_k) = c_k \qquad k = 1, \ldots, m$$

meaning that the i_k-th DOF should attain the value of c_k at time t_k. The animator also specifies a set of n parameters $\mathbf{u}$, which the shooting method is allowed to change in order to satisfy the constraints. In our case these parameters are specific DOFs at specific times, designated by the animator. We require that $m \leq n$, which results in an underdetermined system (there are more DOFs to play with than constraints to satisfy).

Following the formalism in [PSE*00], we say that the dynamics simulator computes the simulation function $\mathscr{S}$, which gives the state $\mathbf{q}(t)$ of the animated character at every point in time as a function of the parameter vector $\mathbf{u}$:

$$\mathbf{q}(t) = \mathscr{S}(t, \mathbf{u}).$$

Denoting by $\mathbf{u}^*$ the original values assigned by the animator, we are looking for:

$$\arg\min_{\mathbf{u}} \|\mathbf{u} - \mathbf{u}^*\|_2 \tag{4}$$

subject to the constraints

$$\mathscr{S}_{i_k}(t_k, \mathbf{u}) = c_k \qquad k = 1, \ldots, m.$$

In other words, we are looking for the minimal change in the

parameters **u** that will cause the simulation to satisfy all of the constraints.

We solve this system of equations using a Newton-like iterative procedure, as described in more detail in Appendix A. The solution process involves estimating the Jacobian of $\mathscr{S}$ by integrating the motion equations and estimating derivatives with finite differences. We found finite differences to be sufficiently accurate for our purposes, since we do not allow collisions with discontinuous impulse.

As described in [PSE*00], the computation of the Jacobian may be done efficiently by decomposing the simulation function $\mathscr{S}$ into analytically differentiable functions, and using the chain rule to numerically compose the Jacobian matrix. However, since we are dealing with articulated and self-actuating characters, rather than single and passive rigid bodies, even free-flight motion is not analytically differentiable.

There are still two cases where we can utilize analytical expressions to speed up optimization for the case of free flight: (i) when the kinematic constraints do not involve the angular components of the motion, and (ii) when the internal pose of the articulated character remains fixed after a certain point in time. In both these cases numerical differentiation is only performed until take-off (case (i)) or until the pose becomes fixed (case (ii)), since from these points in time the body may be regarded as rigid.

We used our kinematic constraints mechanism to make our character jump and perform a double somersault. In this experiment we specified 14 control poses at two different times (one for squash and one for stretch). These control poses were then adjusted by the system using shooting to produce a ballistic trajectory with the specified maximum height, landing distance, and initial angular velocity. Symmetry was utilized to reduce the number of optimization parameters to only eight. We also made the character fold in mid-air using control poses, and specified the landing poses (see Figures 2c and 3d and the video). In order to absorb the momentum during landing the stiffness of all controllers was reduced, while the damping was increased. The stabilizing torques were turned on once the character started landing. Note that the small bounce during landing was generated automatically by the simulation without animator involvement. The entire shooting process converged in four iterations and took a total of 34 seconds.

We could not resist experimenting with a jumping Luxo-like lamp character with 14 DOFs. We made the lamp perform a series of consecutive jumps, each with a constraint on the height of the trajectory and a landing distance. To achieve these constraints the system was allowed to adjust 4 DOFs at two times before each jump. In this case, the kinematic constraints involve only the linear components of the motion. Consequently, the trajectory of the center of mass can be predicted analytically, just like in a simple rigid body case. We take advantage of this and the shooting method ran simulations only between the squash and the stretch poses. The global orientation of the lamp was controlled by the stabilizing force, which did not appear to detract from the realism of the resulting motion. The entire shooting process converged in six iteration and took a total of 2.5 seconds.

In the walking motion demonstrated in the previous section, the animator was able to control the internal DOFs of the walking character, but had no direct control over its translational DOFs. Such control is provided by the kinematic constraints described in this section. For example, the animator may specify the footprints for the right foot of the character, and in this manner define a walking motion that follows a particular path. To satisfy these constraints the system was allowed to modify three DOFs in each walk cycle. Each constrained footprint required three shooting iterations (1.6 seconds per constraint).

Combining feedback controllers with occasional optimization over a few animator-designated optimization parameters offers an efficient tool for motions where kinematic constraints are relatively sparse. This is the case, since the torques applied to most of the character's DOFs most of the time are generated at virtually no cost by the FBCs driven by the animator specified control poses. The few optimization adjusted poses in conjunction with the FBCs guided by them can be viewed as a low dimensional parameterization of the space of torque functions, which we search in order to satisfy the kinematic constraints.

7. Autonomous Objective Controllers

As stated earlier, our goal is to allow animators to focus on controlling the expressive aspects of the character's motion, while relieving them, as much as possible, from the burdensome task of maintaining the physical realism of the motion. In this section we describe another mechanism that helps us realize this goal: *autonomous objective controllers* (AOCs).

As a motivating example, consider a waiter making his way from the kitchen to a table in a crowded restaurant, with a tray full of dishes held on one hand. In this case, we'd like the animator to focus on the style and path of the waiter's walk without worrying about keeping the tray level. Another example is a character trying to stand still or walk in a straight line despite a strong gusty wind, or some other external force, which pushes him in various directions.

In situations such as these, real characters perform many small and rapid yet precise reactive actions with multiple muscles operating in accord. Reproducing such actions using traditional keyframing is a highly complicated and tedious task, but even in our approach many densely spaced and finely tuned control poses might be necessary in order to make the dynamics simulation respond correctly. Thus, it is reasonable to assume that in such situations animators would prefer to control such behaviors using somewhat higher level control mechanisms.

Feedback controllers are a universal solution for these types of problems, and we therefore incorporate them here as well: each higher level task is assigned a new controller. The controller is provided with a set of inputs (sensors) and a set of joints designated by the animator for achieving a particular objective. This controller replaces the ones responsible for matching key poses at those joint. Many objectives may be expressed by a simple relationship between the inputs and the required joint actions, so one could imagine a simple GUI that animators could use to construct such AOCs. Of course, there are also more complicated tasks, which require more sophisticated programming. Such programming could be carried out by suitably skilled technical directors as part of the character rigging stage.

We applied AOCs to the example of the waiter carrying a tray with a glass on it. Two controllers were allocated to keep the glass from falling off the tray. Each of these controllers is given as input one of the two angles that determine the direction of the tray's normal. The setpoint of each of these controllers is zero (i.e., the normal points up). Each controller attempts to achieve its setpoint by applying torques on two joint DOFs: one controller affects one of the angles of the shoulder joint and the angle at the elbow, while the other affects another shoulder angle and the wrist angle. The resulting motion is shown in the accompanying video.

Note that in such cases the animator has parametric control over the manner in which the tasks are carried out. By experimenting with the stiffness and damping coefficients of the controllers the animator can determine how much the character is allowed to deviate from the objective, and the speed of its responses. Also, when more than one DOF is affected by a controller, it is possible to assign a different weight to each DOF.

Different rigging objectives can also be expressed via feedback controllers. This includes such things as rigging a character to perform certain motions in a symmetrical (or anti-symmetrical) manner, etc.

8. Conclusion

We have described a new tool for controlling physically-based simulations of articulated characters. Our tool presents the user with a familiar keyframe-like interface for designing different motions, while enjoying the benefits of a physically-consistent simulation environment.

Our approach relieves the animator from investing efforts into the purely physical aspects of the motion, and concentrating on the expressive aspects instead. Thus, it is most effective for animations where the laws of mechanics play an important role, such as inertial motions, collisions, etc. Since our approach is geared towards generating motion from scratch, it is attractive for situations where motion captured data is unavailable (imaginary creatures, dangerous stunts, etc.) Due to the simplicity of our approach and the resemblance between control poses and standard keyframes, we believe that our method could be easily integrated into today's commercial animation tools.

For more precise control of global motion objectives, our approach supports kinematic constraints, which are satisfied using shooting-based optimization. The animator provides a good initial guess and reduces the dimensionality of the search space by designating specific DOFs. Additionally, shooting is typically performed only over brief time intervals. The low expense of the optimization, combined with the ability of our system to generate unconstrained motions in real time yields an efficient tool, which is particularly attractive for motions where kinematic constraints are relatively sparse.

We believe that our method may also be effective for generating motions of autonomous characters in computer games, digital extras, etc. Such characters could be driven by sequences of control poses generated by relatively simple finite state machines.

In summary, we have demonstrated the ability of our tool to produce physically plausible motions, while providing elaborate control over the character. However, it is by no means an "instant animation machine". Talent and experience are still required in order to produce expressive and lifelike animations.

In the future we would like to further explore the utilization of simple controllers for satisfying additional kinematic constraints, and further reducing the need in optimization. Another important future research direction is to incorporate into our approach relevant research results from the biomechanics literature in order to enhance the ability of our tool to better imitate motion of real creatures.

References

[AG85] ARMSTRONG W. W., GREEN M.: The dynamics of articulated rigid bodies for purposes of animation. In *Proceedings of Graphics Interface '85* (1985), Wein M., Kidd E. M., (Eds.), Canadian Inf. Process. Soc., pp. 407–415.

[Coh92] COHEN M. F.: Interactive spacetime control for animation. *Computer Graphics (Proceedings of SIGGRAPH 92) 26*, 2 (July 1992), 293–302.

[Com01] COMET M. B.: Animation process — a case study. http://www.comet-cartoons.com/toons/3ddocs/animprocess/, 2001.

[FP03] FANG A. C., POLLARD N. S.: Efficient synthesis of physically valid human motion. *ACM Transactions on Graphics 22*, 3 (July 2003), 417–426.

[FvT01] FALOUTSOS P., VAN DE PANNE M., TERZOPOULOS D.: Composable controllers for physics-based character animation. In *Proceedings of SIGGRAPH 2001* (2001), Fiume E., (Ed.), Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, ACM Press, pp. 251–260.

[HP97] HODGINS J. K., POLLARD N. S.: Adapting simulated

behaviors for new characters. In *Proceedings of SIGGRAPH 97* (Aug. 1997), Whitted T., (Ed.), Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, Addison Wesley, pp. 153–162.

[HWBO95] HODGINS J. K., WOOTEN W. L., BROGAN D. C., O'BRIEN J. F.: Animating human athletics. In *Proceedings of SIGGRAPH 95* (Aug. 1995), Cook R., (Ed.), Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, ACM Press / Addison Wesley, pp. 71–78.

[KMB96] KOKKEVIS E., METAXAS D., BADLER N. I.: User-controlled physics-based animation for articulated figures. In *Computer Animation '96* (1996).

[Lan01] LANGO K.: Pose to pose animation: Organized keyframing and how it works. http://www.keithlango.com, 2001.

[Las87] LASSETER J.: Principles of traditional animation applied to 3D computer animation. *Computer Graphics (Proceedings of SIGGRAPH 87) 21*, 4 (July 1987), 35–44.

[LGC94] LIU Z., GORTLER S. J., COHEN M. F.: Hierarchical spacetime control. In *Proceedings of SIGGRAPH 94* (July 1994), Glassner A. S., (Ed.), Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, ACM Press / Addison-Wesley, pp. 35–42.

[LP02] LIU C. K., POPOVIĆ Z.: Synthesis of complex dynamic character motion from simple animations. *ACM Transactions on Graphics 21*, 3 (July 2002), 408–416.

[LvF96] LASZLO J. F., VAN DE PANNE M., FIUME E.: Limit cycle control and its application to the animation of balancing and walking. In *Proceedings of SIGGRAPH 96* (Aug. 1996), Rushmeier H., (Ed.), Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, Addison-Wesley, pp. 155–162.

[LvF00] LASZLO J., VAN DE PANNE M., FIUME E.: Interactive control for physically-based animation. In *Proceedings of SIGGGRAPH 2000* (2000), Akeley K., (Ed.), Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, ACM Press / Addison-Wesley, pp. 201–208.

[MZW99] MATARIĆ M. J., ZORDAN V. B., WILLIAMSON M. M.: Making complex articulated agents dance. *Autonomous Agents and Multi-Agent Systems 2*, 1 (Mar. 1999), 23–43.

[Nat] NATURALMOTION: Endorphin 2.0. http://www.naturalmotion.com.

[NF02] NEFF M., FIUME E.: Modeling tension and relaxation for computer animation. In *SCA '02: Proc. 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2002), ACM Press, pp. 81–88.

[PSE*00] POPOVIĆ J., SEITZ S. M., ERDMANN M., POPOVIĆ Z., WITKIN A.: Interactive manipulation of rigid body simulations. In *Proceedings of SIGGRAPH 2000* (2000), Akeley K., (Ed.), Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, ACM Press / Addison Wesley, pp. 209–218.

[PW99] POPOVIĆ Z., WITKIN A.: Physically based motion transformation. In *Proceedings of SIGGRAPH 99* (1999), Rockwood A., (Ed.), Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, ACM Press / Addison-Wesley, pp. 11–20.

[RGBC96] ROSE C. F., GUENTER B., BODENHEIMER B., COHEN M. F.: Efficient generation of motion transitions using spacetime constraints. In *Proceedings of SIGGRAPH 96* (Aug. 1996), Rushmeier H., (Ed.), Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, Addison-Wesley, pp. 147–154.

[RH91] RAIBERT M. H., HODGINS J. K.: Animation of dynamic legged locomotion. *Computer Graphics (Proceedings of SIGGRAPH 91) 25*, 4 (July 1991), 349–358.

[SHP03] SAFANOVA A., HODGINS J. K., POLLARD N. S.: Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Transactions on Graphics 23*, 3 (Aug. 2003), 512–519.

[Smi04] SMITH R.: *Open Dynamics Engine, v0.5*, 2004. http://ode.org.

[SP05] SULEJMANPAŠIĆ A., POPOVIĆ J.: Adaptation of performed ballistic motion. *ACM Transactions on Graphics 24*, 1 (Jan. 2005), 165–179.

[Tv98] TORKOS N., VAN DE PANNE M.: Footprint-based quadruped motion synthesis. In *Proceedings of Graphics Interface '98* (June 1998), pp. 151–160.

[vF93] VAN DE PANNE M., FIUME E.: Sensor-actuator networks. In *Proceedings of SIGGRAPH 93* (1993), Kajiya J. T., (Ed.), Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, ACM Press / Addison-Wesley, pp. 335–342.

[vKF94] VAN DE PANNE M., KIM R., FIUME E.: Virtual wind-up toys for animation. In *Proceedings of Graphics Interface '94* (Banff, Alberta, Canada, May 1994), Canadian Information Processing Society, pp. 208–215.

[vL95] VAN DE PANNE M., LAMOURET A.: Guided optimization for balanced locomotion. In *Computer Animation and Simulation '95* (1995), Terzolpoulos D., Thalmann D., (Eds.), Springer-Verlag Wien New York, pp. 165–177. Proceedings of the Eurographics Workshop in Maastricht, Netherlands, September 2–3, 1995.

[WB85] WILHELMS J., BARSKY B. A.: Using dynamic analysis to animate articulated bodies such as humans and robots. In *Proceedings of Graphics Interface '85* (1985), Wein M., Kidd E. M., (Eds.), Canadian Inf. Process. Soc., pp. 97–104.

[WK88] WITKIN A., KASS M.: Spacetime constraints. *Computer Graphics (Proceedings of SIGGRAPH 88) 22*, 4 (Aug. 1988), 159–168.

[YCP03] YIN K., CLINE M. B., PAI D. K.: Motion perturbation based on simple neuromotor control models. In *11th Pacific Conference on Computer Graphics and Applications (PG 2003), 8-10 October 2003, Canmore, Canada* (2003), IEEE Computer Society.

[ZH02] ZORDAN V. B., HODGINS J. K.: Motion capture-driven simulations that hit and react. In *SCA '02: Proc. 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2002), ACM Press, pp. 89–96.

[ZMCF05] ZORDAN V. B., MAJKOWSKA A., CHIU B., FAST M.: Dynamic response for motion capture animation. *ACM Transactions on Graphics. 24*, 3 (2005), 697–701.

Appendix A: Shooting

We derive an iterative procedure for solving equation (4) by relaxing the problem as follows,

$$\min_{\delta\mathbf{u}} \left\{ \sum_{j=1}^{n} \delta\mathbf{u}_j^2 + \alpha \sum_{k=1}^{m} (\mathbf{S}_k(\mathbf{u}) - c_k)^2 \right\} \tag{5}$$

such that

$$\forall 1 \le k \le m \qquad \mathbf{S}_k(\mathbf{u}) = c_k$$

where $\mathbf{S}_k(\mathbf{u}) = \mathscr{S}_{i_k}(t_k, \mathbf{u})$ is the i_k-th kinematic component given by the simulation at time t_k, $\mathbf{u}$ is the vector of n parameters associated with the control poses that the animator allows to change, and $\delta\mathbf{u} = \mathbf{u} - \mathbf{u}^*$ is the change vector. To guarantee solvability, we assume that $m \le n$.

We solve this equation iteratively by setting the initial guess to be $\delta\mathbf{u}^0 = 0$, and computing $\delta\mathbf{u}^{l+1}$ given $\delta\mathbf{u}^l$ using a Newton step,

$$\begin{aligned} \mathbf{c} &= \mathbf{S}(\mathbf{u}^{l+1}) \\ &\approx \mathbf{S}(\mathbf{u}^l) + \frac{\partial \mathbf{S}(\mathbf{u}^l)}{\partial \mathbf{u}} \cdot [\delta\mathbf{u}^{l+1} - \delta\mathbf{u}^l] \\ \frac{\partial \mathbf{S}}{\partial \mathbf{u}} \delta\mathbf{u}^{l+1} &\approx \mathbf{c} - \mathbf{S}(\mathbf{u}^l) + \frac{\partial \mathbf{S}}{\partial \mathbf{u}} \delta\mathbf{u}^l \end{aligned}$$

where $\mathbf{u}^l = \mathbf{u}^* + \delta\mathbf{u}^l$. Since this is an underdetermined system, we seek the solution with minimal magnitude, as defined by equation (5).

According to the Lagrange multipliers rule, this solution is given by

$$\frac{\partial \mathbf{S}}{\partial \mathbf{u}}^\top \lambda = \delta\mathbf{u}^{l+1} + \alpha \sum_{k=1}^{m} \frac{\partial \mathbf{S}_k}{\partial \mathbf{u}}^\top (\mathbf{S}_k(\mathbf{u}^l) - c_k).$$

Multiplying by $\frac{\partial \mathbf{S}}{\partial \mathbf{u}}$ yields

$$\begin{aligned} \frac{\partial \mathbf{S}}{\partial \mathbf{u}} \frac{\partial \mathbf{S}}{\partial \mathbf{u}}^\top \lambda &= \mathbf{c} - \mathbf{S}(\mathbf{u}^l) + \frac{\partial \mathbf{S}}{\partial \mathbf{u}} \delta\mathbf{u}^l + \\ &\qquad \alpha \sum_{k=1}^{m} \frac{\partial \mathbf{S}}{\partial \mathbf{u}} \frac{\partial \mathbf{S}_k}{\partial \mathbf{u}}^\top (\mathbf{S}_k(\mathbf{u}^l) - c_k) \end{aligned}$$

where λ is the Lagrange multipliers vector of dimension m.

Having solved for λ we can now update

$$\delta\mathbf{u}^{l+1} = \frac{\partial \mathbf{S}}{\partial \mathbf{u}}^\top \lambda + \alpha \sum_{k=1}^{m} \frac{\partial \mathbf{S}_k}{\partial \mathbf{u}}^\top (c_k - \mathbf{S}_k(\mathbf{u}^l))$$

The Jacobian $\frac{\partial \mathbf{S}}{\partial \mathbf{u}}$ is estimated using finite differences, which involves running the simulation $n+1$ times.

The parameter α relaxes the minimality requirement while the solver is still far from satisfying the constraints. Once the constraints are roughly satisfied the solver will reduce $\|\delta\mathbf{u}\|^2$.