Real-time Control of Physically Based Simulations using Gentle Forces

Jernej Barbič * Jovan Popović

Massachusetts Institute of Technology

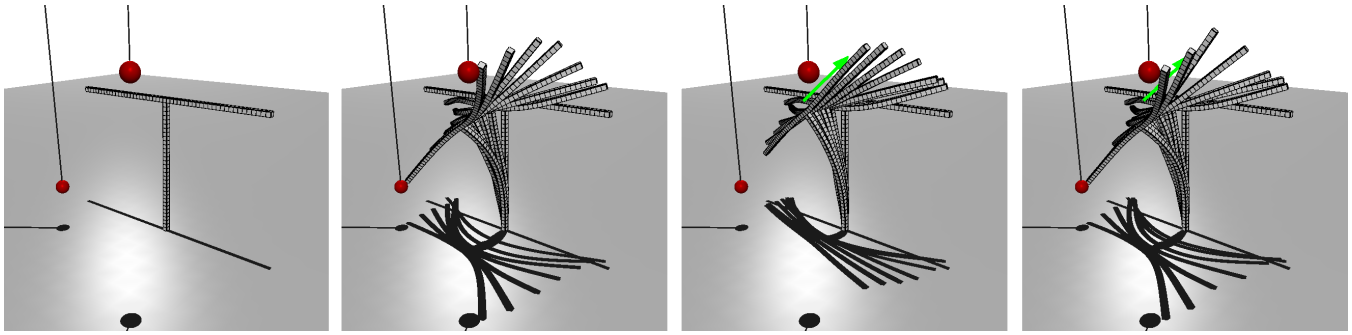

Figure 1: Real-time control ensures fixed simulation outcome regardless of runtime user forces: *First: the rest configuration of the "T"-shape structure and the two target balls. Second: reference motion from an external simulator; the two ends of the "T" impact the two balls. Third: user-perturbed real-time simulation, without control. The two ends miss the target. Forth: controlled user-perturbed real-time simulation, with gentle control forces, tracks the reference motion and successfully impacts the target. The perturbation force load (green arrow; applied 1/5 through the simulation, only in the third and fourth motion) pushes the "T" in the opposite direction of motion.*

Abstract

Recent advances have brought real-time physically based simulation within reach, but simulations are still difficult to control in real time. We present interactive simulations of passive systems such as deformable solids or fluids that are not only fast, but also directable: they follow given input trajectories while simultaneously reacting to user input and other unexpected disturbances. We achieve such directability using a real-time controller that runs in tandem with a real-time physically based simulation. To avoid stiff and over-controlled systems where the natural dynamics are overpowered, the injection of control forces has to be minimized. This search for gentle forces can be made tractable in real-time by linearizing the system dynamics around the input trajectory, and then using a time-varying linear quadratic regulator to build the controller. We show examples of controlled complex deformable solids and fluids, demonstrating that our approach generates a requested fixed outcome for reasonable user inputs, while simultaneously providing runtime motion variety.

CR Categories: I.6.8 [Simulation and Modeling]: Types of Simulation—Animation, I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling, I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual Reality

Keywords: control, real-time simulation, model reduction, deformations, fluids

*e-mail: barbic@csail.mit.edu, jovan@csail.mit.edu

1 Introduction

Fast physically based simulation of passive systems such as deformable models or fluids is a well-researched area in computer graphics. General optimal control of such systems, however, is computationally more demanding than mere forward simulation and is not tractable in real time for complex models. In this paper, we present physically based simulations which are not only fast, but also directable in real-time. We give a real-time controller which directs our nonlinear simulations to follow a given input trajectory while simultaneously reacting to user input and other unexpected disturbances. Such a controller enables *directable interaction*, i.e., interaction that can satisfy certain goals despite the fact that the particular perturbations (such as user forces, stochastic forces or numerical error) are not known at design time. For example, the deformable T-shape in Figure 1 impacts the two pendulum balls yet its motion is not fixed because it simultaneously responds to applied user forces; likewise the bee lands on the controlled flower despite user forces or stochastic wind forces (Figure 4). Many techniques exist that can generate animations offline, satisfying, say, a sparse set of keyframes. However, once authored, these animations are fixed. At runtime, one cannot modify the motion and still preserve desired outcomes; one would have to go back to the authoring phase and re-design the motion. In contrast to existing offline techniques, our method therefore enables interaction with *specific*, rather than merely physical outcomes. It can be used in computer games to provide animations that require both variety (so that no play is the same), and scripting (so that the story progresses).

Such a controller also makes it possible to replicate the look and feel of detailed physically based simulations under computational budget constraints. For example, film production has almost unlimited budget because a final frame needs to be created only once. Interactive systems (such as games), in turn, only have a limited amount of computation time available to produce the next simulation frame. Even in identical environments, one often cannot replicate the look and feel of an offline simulation within an interactive system. Once proper parameters and initial conditions are painstakingly devised for one simulation, their effect should be replicated in every sim-

ilar simulation, but it is often not, due to (for example) varying timesteps and different mesh resolutions. Our method can create real-time simulations that preserve target salient features of detailed simulations generated using a lot of human or computational effort. For example, in our fluid example in Figure 2, we first generated an offline fluid simulation where an immersed leaf follows a curve "S". The controller can then steer interactive simulations so that the leaf trajectory is preserved. Furthermore, an arbitrary number of simulations with similar leaf trajectories can be generated, either through runtime user input, or by sampling random external force perturbations. In training systems, our work could be used to replace the action of a human participant with a controller. For example, say that a surgeon needs to train a procedure which requires two surgeons manipulating a certain tissue in a collaborative effort. One could then replace one surgeon with our controller, allowing the single trainee to perfect just his part of the task.

Real-time controlled physically based simulation is not possible without real-time forward simulation. In our work, we achieve fast forward simulation using model reduction, a popular theme in many scientific disciplines [Li and Bai 2005]. We use model reduction because it gives very compact low-dimensional state spaces where real-time control is tractable. Reduced simulations, which approximate full simulations, can forward-simulate deformable solids and fluids at real-time rates. The underlying general principle of model reduction is to replace the full simulation state (say, with several thousand degrees of freedom) with a low-dimensional reduced state (with only several tens of DOFs), relate the two via a projection matrix, and approximate the full equations of motion with a projection onto the low-dimensional space.

The core of our method is a real-time controller which runs in tandem with a real-time physically based simulation. Our controller is a tracking controller: its objective is to steer the runtime simulation toward a given pre-simulated system trajectory. In the absence of any runtime perturbations, the controller simply plays back the external forces which generated the tracked trajectory (the *feedforward* control), which in turn replays the tracked trajectory. In the presence of runtime disturbances, however, the controller applies control forces (the *feedback* control) which guide the simulation toward the pre-simulated trajectory. Our feedback forces are a solution of an optimization problem. They minimize an objective function that combines the tracking error and amount of injected control for the remainder of the tracked trajectory. For a given level of tracking error, our controller produces smaller forces than previous strategies (such as PD control), hence we call our feedback forces "gentle". These gentle control forces can be computed quickly. They are a linear function in the current (position and/or velocity) deviation from the tracked trajectory, with time-varying gains. The gain matrices incorporate system's natural dynamics and are generated automatically using a quick precomputation step.

2 Related work

Physically based simulations of geometrically detailed models in general do not run at real time rates, but real-time simulations are possible with various approximations. Each approximation inevitably creates a different simulation making it difficult to enforce consistent outcome. Even in offline simulations, fictional discretization-dependent forces cause simulations on one discretization to differ from simulations on another. Our control can counter such forces to ensure user-prescribed outcomes while preserving plausible interaction.

Real-Time Simulation: We achieve real-time forward simulation using model reduction; in particular, we follow [Barbič and James 2005] for deformable solids and [Treuille et al. 2006] for fluids. Both of these methods simulate nonlinear systems, supporting large deformations of solids and chaotic fluid dynamics. We use empirical simulation bases, obtained by applying Principal Component Analysis (PCA) to data computed using a full offline simulator. Model reduction can also employ pre-determined global bases, such as low-order polynomial or super-quadric deformation fields [Witkin and Welch 1990; Metaxas and Terzopoulos 1992]. In computer graphics, real-time forward simulations have also been achieved using multi-resolution methods [Debunne et al. 2001; Capell et al. 2002; Grinspun et al. 2002], or by driving detailed rendering meshes with coarse simulations [Faloutsos et al. 1997; Müller and Gross 2004]. Unfortunately, real-time simulations seldom match the spatial and temporal behavior of more detailed simulations.

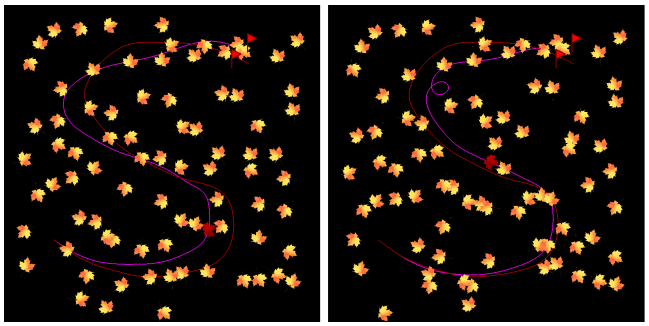

Figure 2: Controlled simulation with runtime variety: *the red leaf in this fluid simulation is controlled to follow the S curve, while random external forces (Perlin noise; sampled continuously in time) provide different perturbations for each simulation run. The other leaves are only advected by the fluid (not controlled). Two separate simulation runs shown.*

Fictitious Forces: Numerical damping is present in cloth simulations or particle systems, especially with implicit integrators at large timesteps. This has prompted the use of explicit integrators in industry applications [Kačić-Alesić et al. 2003] where artificial forces can be purposefully added and removed. Likewise, energy dissipation in semi-Lagrangian fluid advection schemes [Stam 1999] is often a mix of user-prescribed damping and uncontrollable numerical viscosity, especially at large time steps common in computer graphics. Excessive diffusion has been combated by artificially re-introducing vorticity into the flow [Fedkiw et al. 2001], using FLuid-Implicit-Particle (FLIP) methods [Zhu and Bridson 2005], by circulation-preserving discrete operators [Elcott et al. 2007], or higher-order advection techniques [Selle et al. 2008]. With model reduction, numerical diffusion causes a mismatch between full and reduced simulations. Even though reduced fluids can be made energy preserving [Treuille et al. 2006], numerical viscosity in full simulations does not necessarily correspond to any particular viscosity level. In our work, we inject small fictional forces in order to steer reduced simulations toward full simulation trajectories, allowing us to preserve features of full simulations such as trajectories of particles immersed into the fluid.

Control Forces: Nonlinear dynamics presents a key challenge for controlling real-time systems. One approach is to negate nonlinearities with inverse dynamics [Isaacs and Cohen 1987]. Although this simplifies control, inverse-dynamics will use arbitrary forces to overpower (cancel) the system natural dynamics, resulting in over-controlled (stiff) interaction even for naturally compliant materials.

These problems can be reduced with the constrained-Lagrangian formulation of inverse dynamics [Barzel and Barr 1988; Bergou et al. 2007]. This approach expresses control tasks as constraints and maintains them with fictional constraint forces, acting in directions normal to the constraint surface. However, forces of arbitrary magnitude can be applied to remove any disturbance components that violate the constraint. Hence, naturally compliant materials will feel stiff for some interactions and compliant for others.

Compliant control can be accomplished with linear feedback such as proportional-derivative control that has led to some of the most striking simulations of human and animal motion [Hodgins et al. 1995; Wooten and Hodgins 2000; Faloutsos et al. 2001; Yin et al. 2007]. Similar approaches have also been used to design force fields to control fluids [Fattal and Lischinski 2004; Rasmussen et al. 2004; Shi and Yu 2005; Thürey et al. 2006] and elastic deformations [Sifakis et al. 2005; Capell et al. 2005]. However, all of these approaches have depended on manual tuning of control gains. Desired outcome is difficult to enforce, the gains must be retuned with every change in simulation parameters, and the control efforts are not minimized. Tuning can be improved with heuristics [Hodgins and Pollard 1997; Zordan and Hodgins 2002; Yin et al. 2003; Zordan et al. 2005] or exhaustive search [Tedrake 2004; Sharon and van de Panne 2005; Sok et al. 2007]. However, it is not known how to extend these heuristics to new simulation domains or to state spaces with tens of degrees of freedom. We further compare our work to PD control in Section 6.

Optimal control minimizes the injection of control forces using the knowledge of the natural dynamics of the system. It has been used extensively for automatic generation of human motion [Popović and Witkin 1999; Fang and Pollard 2003; Safonova et al. 2004; Sulejmanpasić and Popović 2005; Liu et al. 2005]. The benefit of similar strategies has been demonstrated for control of rigid-body simulations [Popović et al. 2003] and fluids [McNamara et al. 2004]. However, all of these approaches are either too slow for real-time control or only generate feed-forward control, which is quickly invalidated by any user input. In this paper, we demonstrate real-time near-optimal *feedback* control. Although the general formulation of our problem is well known as a Linear Quadratic Regulator (LQR) [Stengel 1994], its practical applications in computer animation have been limited to offline control of simple physical systems [Brotman and Netravali 1988], and character animation [da Silva et al. 2008]. Another unique aspect of our study is the exploration of time-*varying* approximations whereas LQR control is more commonly applied to stabilize linear time-*invariant* approximations. With these approximations, LQR control, which is optimal for linear systems, will not ensure that we inject the smallest possible control forces, but it will reduce force levels to enable plausible real-time controlled interaction.

3 Background: Full and reduced simulation

Full simulations are simulations without reduction; they can in general (assuming linear control) be expressed as the following (high-dimensional) system of ODEs:

$$\dot{q} = F(q,t) + \overline{B}u. \tag{1}$$

Here, $q \in \mathbb{R}^n$ is the state vector (n will typically be at least several thousands), $F(q,t) \in \mathbb{R}^n$ is some (nonlinear) function specifying the system's internal dynamics, $\overline{B} \in \mathbb{R}^{n \times m}$ is a constant *control matrix*, and $u \in \mathbb{R}^m$ is the *control vector*. We demonstrate our method using two particular forms of Equation 1:

1. Navier-Stokes equations of incompressible fluid (smoke) inside a 2D rectangular domain, discretized on a MAC Eulerian grid, with free-slip boundary conditions. These assumptions lead to a first-order ODE for MAC grid velocities (our state vector q) [Fedkiw et al. 2001].

2. Geometrically nonlinear FEM solid deformable simulations supporting large deformation dynamics [Capell et al. 2002]. State vector q consists of displacements and velocities of the vertices of a 3D simulation mesh. Equations of motion are second-order. The inclusion of velocities into the state allows us to write them in the first-order form of Equation 1.

Neither of these equations uses explicit time dependency (we have $F = F(q)$); this is a common assumption and could be relaxed easily.

Reduced simulations are obtained by projecting Equation 1 onto a r-dimensional *subspace*, spanned by columns of some basis matrix $U \in \mathbb{R}^{n \times r}$ (r is in the 24-64 range for our simulations). The full state is then approximated as $q = Uz$, where $z \in \mathbb{R}^r$ is the *reduced state*. The resulting low-dimensional system of ODEs

$$\dot{z} = \tilde{F}(z,t) + Bw, \quad \text{for} \quad \tilde{F}(z,t) = U^T F(Uz,t), \tag{2}$$

approximates the high-dimensional system provided that the true solution states q are well-captured by the chosen basis U. Here, $B \in \mathbb{R}^{r,s}$ is a constant matrix, and $w \in \mathbb{R}^s$ is the *reduced control vector* (usually $s \leq r$).

4 Real-time control

Our runtime simulations are reduced simulations with control; they are obtained by time-stepping Equation 2, with the controller computing $w(t)$ in real-time. Our controller runs entirely in the low-dimensional space. Its objective is to make the reduced state $z(t)$ track a given *reduced* trajectory $z^{\text{ref}}(t)$. We call this tracked low-dimensional trajectory the *reference trajectory*. The reference trajectory is some given sequence of states $\{z_i^{\text{ref}}\}_i$, corresponding to times $\{t_i\}_i = i\Delta t$ along the time axis (Δt is the timestep size). At runtime, the user will apply disturbances to the system, which, in the absence of control, will cause the reduced system to veer off the tracked trajectory. The goal of our control is to minimize this deviation while still providing a plausible physical response to the runtime interaction forces.

The reference trajectory is accompanied with the *feed-forward control* (also called *reference control*); this is the low-dimensional control (we denote it by $w^{\text{ff}}(t)$) that reproduces the reference trajectory (in a reduced simulation) in the absence of runtime disturbances:

$$\frac{d}{dt} z^{\text{ref}}(t) = \tilde{F}\left(z^{\text{ref}}(t), t\right) + B w^{\text{ff}}(t). \tag{3}$$

It is not always possible to obtain $w^{\text{ff}}(t)$ that satisfies Equation 3 exactly. Our controller is able to track the reference trajectory even if $w^{\text{ff}}(t)$ is only approximate. The control term of Equation 2 can now be decomposed as

$$w(t) = w^{\text{ff}}(t) + w^{\text{fb}}(t) + w^{\text{ext}}(t), \tag{4}$$

where $w^{\text{fb}}(t)$ is the control applied by our real-time controller, and $w^{\text{ext}}(t)$ are external disturbances, such as user-applied interaction forces, stochastic forces, or numerical simulation error.

We treat the low-dimensional basis U, the reference trajectory, and feed-forward control as inputs to our algorithm. Our objective is to build a controller which can track (replay with variety) given offline data in real-time; generation of this data itself is not the subject of our work. For our examples, we generated our input using an external simulator which can timestep Equation 1 offline, potentially followed by appropriate reduced simulations. We describe this experimental setup in Results (Section 5).

4.1 Controller objective function

Given the reference trajectory $\{z_i^{\text{ref}}\}_i$, feed-forward control $\{w_i^{\text{ff}}\}_i$, the current runtime timestep index i and current state $z = z(t_i)$, the goal of our controller is to minimize

$$\frac{1}{2}\big(z(t_{\text{final}}) - z^{\text{ref}}(t_{\text{final}})\big)^T Q(t_{\text{final}})\big(z(t_{\text{final}}) - z^{\text{ref}}(t_{\text{final}})\big) + \tag{5}$$
$$\frac{1}{2}\int_{t=t_i}^{t=t_{\text{final}}} \Big(\big(z(t) - z^{\text{ref}}(t)\big)^T Q(t)\big(z(t) - z^{\text{ref}}(t)\big) + w^{\text{fb}}(t)^T R(t) w^{\text{fb}}(t)\Big) dt$$

over all possible feed-back control values $w_i^{\text{fb}}, w_{i+1}^{\text{fb}}, \ldots, w_{T-1}^{\text{fb}}$. The first term penalizes the deviation from the reference state at time $t = t_{\text{final}} = T\Delta t$, the first integral term penalizes the tracking error from the current time to the end of the reference trajectory, and the second integral term penalizes using control. Here, $Q(t) \in \mathbb{R}^{r\times r}$ and $R(t) \in \mathbb{R}^{s\times s}$ are arbitrary (potentially time-varying) position error and control cost matrices, respectively. These two matrices determine the trade-off between tracking the reference trajectory tightly and exerting control. The matrix $Q(t_{\text{final}})$ is the *final cost*. It enables one to boost the importance of meeting the reference state at the *last* frame along the reference trajectory. For example, by setting $Q(t) = 0$ for all t in the integral term, and $Q(t_{\text{final}}) \neq 0$, one obtains a controller that ensures the final condition is met closely, without regard on the intermediate tracking error . Note that Equation 5 explicitly singles out the tracking error at $t = t_{\text{final}}$. However, using a proper time-varying position cost $Q = Q(t)$, one can easily emphasize (or single out using a δ-function) other important times along the trajectory. Also, by time-varying R, one can shift the balance between tracking and control along the time axis. In our examples, we set Q and R to multiples of the identity matrix:

$$Q(t) = \alpha_Q I_{r\times r}, \quad Q(t_{\text{final}}) = \alpha_{Q_{\text{final}}} I_{r\times r}, \quad R = \alpha_R I_{s\times s}, \tag{6}$$

where $\alpha_Q, \alpha_{Q_{\text{final}}}$ and α_R are appropriate scalar parameters. Note that scaling these parameters with an arbitrary constant only rescales the objective function, so there are essentially only two independent parameters. Furthermore, we often set one of $\alpha_Q, \alpha_{Q_{\text{final}}}$ to zero. In this case, animators can tune a single intuitive scalar parameter (ratio of position vs control cost), allowing them to directly set the tradeoff between tracking error and amount of control.

4.2 The controller

Both full and reduced dynamical systems from Equations 1 and 2 are nonlinear, causing the minimization from Equation 5 to have no closed-form solutions. Assuming perturbations are known ahead of time, the solution can be obtained offline using standard techniques such as space-time optimization. Such approaches lead to nonlinear optimizations with many degrees of freedom and are orders of magnitude too slow for an interactive system.

There is, however, a way to simplify the problem to make it tractable in real-time: if one linearizes the ODEs of Equation 2 *around the reference trajectory*, then the minimization problem from Equation 5 has an exact analytical solution. Linearization around the reference trajectory means that the system of ODEs of Equation 2, at time t, is linearized around the reference state at time t, yielding a time-varying *linear* system of ODEs. Linearization is performed by introducing the *state error* variable $(\Delta z)(t) = z(t) - z^{\text{ref}}(t)$, rewriting Equation 2 in terms Δz, taking into account Equation 3, and applying the approximation

$$\tilde{F}\Big(z^{\text{ref}}(t) + \Delta z, t\Big) \approx \tilde{F}\Big(z^{\text{ref}}(t), t\Big) + \frac{\partial \tilde{F}}{\partial z}_{|z=z^{\text{ref}}(t)} \Delta z. \tag{7}$$

This gives a *linear* ODE for Δz, controlled only by w^{fb} :

$$\frac{d}{dt}\Delta z = \frac{\partial \tilde{F}}{\partial z}_{|z=z^{\text{ref}}(t)} \Delta z \; + \; B w^{\text{fb}}. \tag{8}$$

If the minimization of Equation 5 is performed with respect to the dynamical system of Equation 8, the optimal control policy can be shown to be *linear* in the current state error Δz. The controller that executes this policy is called a *linear quadratic regulator* [Stengel 1994]. Optimal control at timestep i when the current state error is Δz, equals $w^{\text{fb}} = K_i \Delta z$, where $K_i \in \mathbb{R}^{s\times r}$ is a constant *gain matrix* that depends only on the timestep index i, and not on Δz.

Given a reference trajectory, the feed-forward control, and the cost matrices, the matrices K_i can be precomputed efficiently by solving a Riccati differential equation, with an initial condition at the last timestep, backwards in time. Table 1 gives LQR computation statistics. Computation details are given in Appendix A.

	r	T	computation time	space for $\{K_i\}_i$
T-shape	24	501	17.5 sec	4.4 Mb
flower	24	1281	46.9 sec	11.2 Mb
dinosaur	30	181	12.5 sec	2.5 Mb
fluid	64	90	8.4 sec	3.1 Mb
leaves	64	200	15.2 sec	6.5 Mb

Table 1: LQR precomputation statistics: *All statistics are totals for the entire sequence $K_i, i = 0, \ldots, T-1$. Space complexity (given in double precision) is rsT floating point numbers, and time complexity is $O\big((f + r^3)T\big)$, where f is the complexity of a reduced forward-simulation timestep: $f = O(r^3)$ for fluids, $f = O(r^4)$ for geometrically nonlinear deformable solids.*

4.3 Deformable solids

We use the reduced deformable model obtained by applying (POD-style) model reduction to geometrically nonlinear FEM deformable models [Barbič and James 2005]. These models support large deformations and dynamics. The reduced model is simply a projection of the standard nonlinear second-order FEM deformable model [Capell et al. 2002]:

$$\ddot{p} + \tilde{D}(p)\dot{p} + \tilde{R}(p) = \tilde{f}_{\text{ext}}(t), \tag{9}$$

where $p \in \mathbb{R}^r$ are the reduced deformations, $\tilde{D}(p) \in \mathbb{R}^{r\times r}$ is the reduced Rayleigh damping matrix, $\tilde{R}(p) \in \mathbb{R}^r$ are the reduced internal forces, and $\tilde{f}_{\text{ext}}(t) \in \mathbb{R}^r$ are the reduced external forces. The full deformation vector, consisting of displacements of the vertices of the FEM simulation mesh, is given by $Up(t)$, where $U \in \mathbb{R}^{n\times r}$ is a subspace basis matrix. We rewrite Equation 9 into first-order form by introducing the state vector $z = [p^T, \dot{p}^T]^T \in \mathbb{R}^{2r}$; the resulting $\tilde{F}(z)$ and $B \in \mathbb{R}^{2r\times r}$ are given in Appendix B. Equation 8 now reads

$$\frac{d}{dt}\Delta z = A(t)\Delta z + B w^{\text{fb}}, \tag{10}$$

where the matrix $A(t) \in \mathbb{R}^{2r\times 2r}$ is given in Appendix B. In order to build the gain matrices K_i (Appendix A) it is necessary to evaluate $A(t_i)$, for all timesteps i. This computation has equal asymptotic cost as forward reduced simulation and can re-use much of the same code. We are only injecting control through reduced deformations, not reduced velocities (upper $r \times r$ block of B is zero). This is a common choice with second-order dynamical systems where external forces are the natural place to add control. In principle, one could inject control also at the velocity level, however, the results would then generally look less physical.

4.4 Fluids

We also demonstrate our results using real-time 2D reduced fluids [Treuille et al. 2006], which are obtained by applying proper (POD-style) model order reduction to standard Eulerian grid fluid equations encountered in computer graphics. We use a standard MAC grid with velocities at edge centers and pressures at cell centers. The reduced equations of motion are

$$\dot{z} = \hat{A}(z)z + \nu\hat{D}z + w, \tag{11}$$

where z are the reduced velocities, $\hat{A}(z)$ is the reduced advection matrix, $\hat{D}$ is the reduced diffusion matrix, $\nu \geq 0$ is viscosity, and w are the reduced external forces (including control). The unreduced fluid velocities are approximated as $Uz(t)$, where U is a velocity basis matrix. Note that divergence-free pressure projection is not necessary for reduced simulations, as columns of U already are divergence-free by basis construction [Treuille et al. 2006]. For full simulations, we use a standard semi-Lagrangian advection scheme, and we performed pressure projection with a direct sparse solver.

The reduced fluid can be controlled with a procedure equivalent to the one described with reduced deformations: first identify $\tilde{F}$ and its gradients, then evaluate them along the reference trajectory to build a LQR controller. This allows one to steer the reduced velocities to that of the reference trajectory. While this tracking worked well in our experiments, we usually augment the reduced state by immersing one or a few particles into the fluid. Particles provide more visual output to the user than using only the velocities; control of fluids with particle forces has been embraced by the community [Rasmussen et al. 2004; Thürey et al. 2006]. We control the particles to their reference trajectories in addition to controlling the reduced fluid velocities. In particular, for a reduced fluid with N immersed particles, we augment the reduced state to $\hat{z} = [p_1^T, \ldots, p_N^T, z^T]^T \in \mathbb{R}^{2N+r}$, where $p_i \in \mathbb{R}^2$ is the current position of particle i. The equations of motion of particles are then

$$\frac{d}{dt}p = \left[\Psi\big(p_1(t)\big)^T, \ldots, \Psi\big(p_N(t)\big)^T\right]^T z(t), \tag{12}$$

where we have assembled $p = [p_1^T, \ldots, p_N^T]^T \in \mathbb{R}^{2N}$, and where $\Psi(x) \in \mathbb{R}^{2\times r}$ are the fluid velocity modes, evaluated at the world-coordinate location x in the fluid. These modes $\Psi(x)$ are obtained by interpolating MAC edge basis velocities of U to arbitrary locations x inside the fluid domain. We obtain particle reference trajectories either using a reduced simulation, or by immersing particles into a full simulation. Equation 8 takes the form

$$\frac{d}{dt}\begin{pmatrix}\Delta p\\ \Delta z\end{pmatrix} = \begin{pmatrix}Y(t) & W(t)\\ 0 & A(t)\end{pmatrix}\begin{pmatrix}\Delta p\\ \Delta z\end{pmatrix} + \begin{pmatrix}0\\ B\end{pmatrix}w^{\text{fb}} \tag{13}$$

where $w^{\text{fb}} \in \mathbb{R}^r$ only injects reduced velocity control, and matrices A, Y, W and B are given in Appendix C. Note that particles' positions are now coupled with reduced velocities and that one obtains pure reduced velocity control for $N = 0$.

Our position cost matrix is a diagonal matrix with entries $\lambda_P, \ldots, \lambda_P, \lambda_Q, \ldots, \lambda_Q$, where λ_P and λ_Q are the costs of errors in particle position and reduced state, respectively. In practice, we usually set λ_P orders of magnitude higher to λ_Q so that control puts more emphasis on steering the particles rather than velocities.

5 Results

Our controller requires a subspace basis, a reference trajectory and feed-forward control, which we generated using an external simulator (by time-stepping Equation 1 offline). The simulator produces one or more *full simulation trajectories*; each is a sequence $\{q_i\}_i$ of high-dimensional states q_i, together with the high-dimensional sequence of controls (external forces) $u_i = u(t_i)$ that generated $\{q_i\}_i$. The choice of this control is not the focus of our work; we applied a few initial impulses and selected an interesting trajectory to track (the *center-line trajectory*) using trial and error.

5.1 Basis extraction

Once the tracking trajectory has been selected, it is necessary to establish a low-dimensional basis for model reduction. The basic requirement is that the basis must capture the center-line trajectory well. We do so by applying Principal Component Analysis (PCA) to properly selected simulation data, which includes the center-line trajectory and enrichment data (see below). We apply PCA in the standard way: we assemble $\{q_i\}_i$ into one (large) matrix, column i containing q_i, and use SVD (or incremental SVD [James and Fatahalian 2003]) to extract a low-dimensional space.

5.2 Basis enrichment

Our reduced simulations should be able to express a rich set of perturbations to the center-line trajectory, establishing runtime simulation variety. It is not sufficient simply to PCA the centerline trajectory; this usually gives over-specific bases which do not generalize beyond the tracked trajectory. Instead, we enrich the centerline trajectory before PCA. We do so by running full simulations with control (external forces) randomly perturbed. We sampled N (typically ~ 50) locations $\overline{t_i}$ along the time-axis, using time-stratified random sampling. We then ran N full simulation runs, with each of the runs applying a single random perturbation, at time $\overline{t_i}$.

With fluid simulations, our perturbations are wind fields with randomized magnitude and width. Their location is either randomized with bias toward high-velocity regions at time $\overline{t_i}$, or, when we immerse particles in the fluid, simply selected to be a particle's position in the center-line simulation at time $\overline{t_i}$. With solid deformable simulations, we either randomly perturb initial velocities of the deformable object, or we sample random force impulses on a subset of mesh vertices, at times $\overline{t_i}$.

The perturbations need to be sufficiently large to establish variety, but not too large, so that a low-dimensional basis can still capture them adequately. The different sampling runs should sample a *tunnel* around the center-line trajectory, rather than completely new trajectories (see Figure 3, Red). In our experiments, the magnitude of perturbation forces was about 5-20% of the center-line control.

5.3 Reference trajectory and feed-forward control

We must now convert the selected full center-line trajectory into a low-dimensional pair (reference trajectory, feed-forward control), to be tracked by our controller. This can be done by projecting the control used to generate the full center-line simulation into the subspace, obtaining $w_i^{\text{ff}} = U^T\overline{B}u_i$, for all timesteps $i = 0, \ldots, T-1$. Next, we simulate the *reduced* system using control $\{w_i^{\text{ff}}\}_i$, producing the reference trajectory $\{z_i^{\text{ref}}\}_i$.

It is sometimes difficult to make such reference trajectories match the projection of the full simulation into the subspace. For example, full simulations in graphics usually advect with semi-Lagrangian backtracking [Stam 1999] which introduces numerical dissipation. Reduced simulations, however, preserve energy (or the chosen viscosity level) [Treuille et al. 2006] much more closely, resulting in a visual mismatch between full and reduced simulations. In such cases, our controller can be used to make the reduced simulation match the full simulation by setting the reference trajectory to the

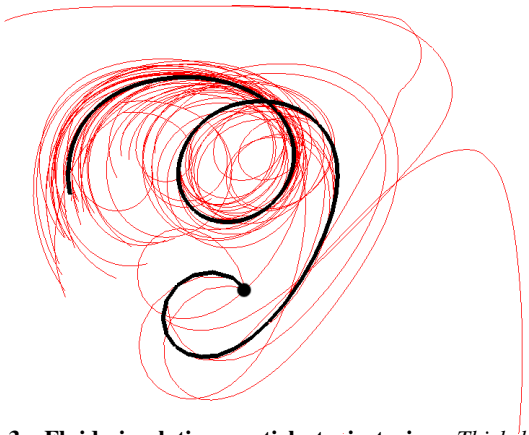

Figure 3: Fluid simulation particle trajectories: *Thick black line: full simulation "center-line" trajectory ($\nu = 0$; semi-Lagrangian advection). Red thin lines: full simulation basis enrichment trajectories. Bottom 1/4 of the simulation box is not shown. All trajectories correspond to a particle initially located at the center of the simulation box (denoted by the black circle) and integrated using a Runge-Kutta second-order integrator. Same initial external force, followed by control-free motion. 256x256 MAC grid.*

projection of the full center-line trajectory: $z_i = U^T q_i$. The feed-forward control is set to $w_i^{\text{ff}} = U^T \overline{B} u_i$. Note that such feed-forward control and reference trajectory do not necessarily match. However, in our experiments, controllers were robust and tolerated moderate mismatches. For example, in a fluid simulation where the motion was excited by a single initial impulse, the controller easily recovered to the reference trajectory even if feed-forward control was set to a constant zero vector (Figure 5, bottom).

5.4 Examples

We show examples of complex physically based simulations tracking precomputed data and reacting to user input. Table 2 gives runtime statistics of our controlled real-time reduced simulations.

	r	n	FS	control	FU	fps
T-shape	24	1032	0.8	0.02	95	160 Hz
flower	24	12705	0.8	0.02	2300	65 Hz
dinosaur	30	53178	1.4	0.03	22,000	55 Hz
fluid	64	131,584	3.0	0.07	78	80 Hz
leaves	64	131,584	3.0	0.07	78	145 Hz

Table 2: Runtime statistics: *Our control is computationally very inexpensive compared to reduced forward simulation. All timings are for a single timestep and are given in milliseconds.* FS=*reduced forward simulation cost (without control),* FU=*full forward simulation cost (no control). The number of simulation mesh vertices is* $n/3$. *Output graphical frame rate is* fps. *Rendering is accelerated on the GPU. Machine specs: Apple MacBook Pro, Mac OS X, 2.33 GHz Intel Core 2 Duo processor with 3 Gb memory, ATI Radeon X1600 graphics card with 256 Mb memory.*

Our first solid deformable example is a deformable T-shape structure (see Figure 1). In the reference trajectory, the two endpoints of the "T" strike two suspended balls. In a forward simulation with user-applied runtime forces, however, the "T" misses its target. Our real-time controller ensures that the "T" hits the two targets, despite user perturbations, with a minimal amount of artificially injected forces. Several trade-offs between control and tracking error are possible: tracking can be made stiff (overwhelming natural dynamics), final cost can be set high to enforce the final goal with less emphasis on the in-between trajectory, or control can be kept at a low level which better preserves system's inherent dynamics.

The flower example (see Figure 4) demonstrates robust real-time interaction in the presence of large disturbances. In this example, feed-forward control is non-zero throughout the motion: we manually scripted feed-forward forces that cause the flower's stem to swing far left, then far right, and then back to undeformed pose, thus making a complete cycle of motion. We also added stochastic feed-forward wind forces so that the leaves are in constant motion. It was difficult to script the forces so that the last frame cycle meets the initial frame. We only matched them approximately through trial and error, and then used our controller to fix the discrepancies, generating endless motion. To illustrate how our controller can ensure fixed outcomes, we added a non-controlled bee following a scripted trajectory in space-time. The bee only lands on the flower because our controller ensures that the landing spot is at the right place at the right time. Without control, user forces alter the motion of the flower, and the bee does not land on the flower.

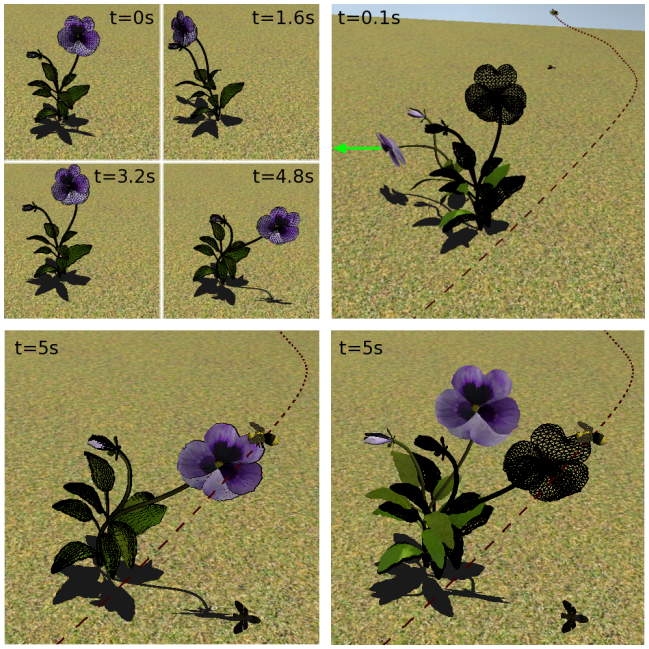

Figure 4: Real-time control enables simulations with predictable outcomes: *Top-left: four frames of the reference trajectory. Top-right: user applies a large perturbation. Bottom-left: controlled simulation; flower and bee meet in space-time. Bottom-right: uncontrolled simulation; flower and bee do not meet. Wireframe mesh gives the reference trajectory. Same user external forces in both cases (shown at Top-right).*

Our method can be used to create real-time simulations that preserve offline simulators properties, such as the trajectory of one or a few particles immersed in the fluid. This usually causes smoke in the vicinity of particles to be advected in a similar way to offline simulations. Figure 5 (top-left) shows the mismatch between the full and reduced trajectory of a particle. We can use our controller to cause the particles to follow their trajectories from full simulation. We selected three particles and performed the full forward (center-line) simulation to obtain their full simulation trajectories. Next, we set these trajectories as particle reference trajectories, and set the projection of the full simulation to the low-dimensional space as the reference velocity trajectory. We then ran real-time controlled sim-

ulations ($r = 64$), using the enrichment basis from Figure 3, with and without user disturbances, for several control cost values (see Figure 5). Without disturbances, the controller maintains both the reduced velocities and particles' positions close to the reference trajectories (Figure 5, top-left). With disturbances, the controller applies a compromise between state error and control (Figure 5, top-right). Cheaper controls causes closer trajectory matches, but disturbs the natural dynamics with higher levels of control forces.

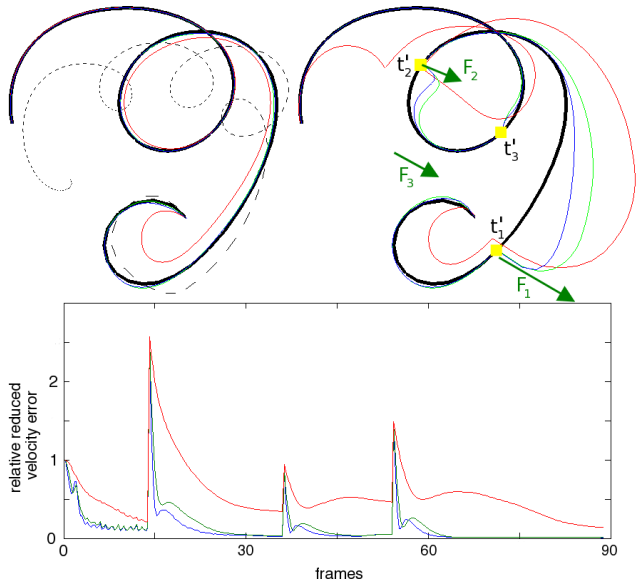

Figure 5: Particle trajectories in controlled real-time fluid simulations: *Top-Left: The trajectory of a particle in a 256x256 full simulation (thick solid), in an uncontrolled reduced simulation (thin dashed), and in controlled reduced simulations (thin solid) under three levels of LQR control: green and blue trajectories correspond to 100x, 100,000x cheaper control than the red trajectory, respectively. Numerical viscosity causes the full simulation to be less energetic than the uncontrolled reduced simulation rendering the two trajectories quite different. Top-Right: Full particle trajectory and projected fluid velocities are used as a reference for controlled simulations with three user perturbations (forces F_1, F_2, F_3 applied at times t'_1, t'_2, t'_3, indicated by green arrows; reference particle positions at perturbation times are indicated by yellow boxes). Bottom: relative velocity tracking error. We used zero feed-forward control in this example; in the initial part of the simulation the controller quickly increases the velocities to match the high initial reference velocities.Three particles were controlled simultaneously in this experiment; only one is shown in the top row diagrams for clarity.*

6 Comparison to PD control

In this section, we compare the LQR controller to the proportional-derivative (PD) controller. While PD has the advantage of simplicity (although a LQR implementation is not very complex either; see Appendix A), LQR produces visibly more natural motions for the same level of tracking error (see Figure 6). A PD controller (formulated here for reduced fluids with immersed particles) applies a force only by looking at the current state:

$$w_{\text{fb}}^{\text{PD}} = -k_Z \Delta z - k_P \Delta p - k_D (\Delta p)', \tag{14}$$

where k_Z, k_P and k_D are scalar PD gains. The LQR controller, in turn, picks control forces that minimize an objective function over the remaining part of the reference trajectory. Assuming a linear time-varying system (LTV), PD will therefore always give sub-optimal objective values. In our work, controllers drive a nonlinear simulation which is well-approximated by a LTV for moderate state errors Δz, causing LQR to use less control than PD, for same level of tracking error. The advantage of LQR is particularly pronounced when long-horizon planning is important, such as when control is expensive (gentle forces), or when final costs are high (both of which are relevant for computer animation). With inexpensive (stiff) control, the two controllers yield similar results, as any trajectory deviations are quickly removed by either controller.

LQR and PD controllers are functionally equivalent: in both cases, feedback force is a product of some (time-varying) gain matrix and the current state deviation from the reference trajectory. LQR results can always be replicated with a PD controller that simply employs the time-varying LQR gain matrices $\{K_i\}_i$. The key difference between LQR and PD is that with PD, one needs to find such time-varying feedback gains *manually*, whereas LQR computes them automatically, and in seconds. For example, suppose one needs to control a system where interim trajectory deviation is not important, but where certain final conditions must be imposed. The PD gains of Equation 14 are constant in time. For quality PD tracking, one would need to tune time-varying versions of PD parameters (a large number of parameters), whereas LQR computes the time-varying gains automatically. Also, PD gains only indirectly correspond to the desired goal. With LQR, one tunes parameters that directly control the trade-off between tracking error and control. Typically, one tunes a single parameter: the ratio of position (or final) cost vs control cost. It is easier to reach a given goal by increasing its weight in the objective function than by adjusting the PD gains, much like we prefer using spline control points to the coefficients of the cubic polynomial. And, because of the optimality for the linearized time-varied system, LQR gains will be difficult to outperform by manual tuning. In our experience, LQR parameter tuning was easy and intuitive, and fast due to short LQR precomputation times. Such tuning is virtually impossible with previously proposed offline optimal control strategies, where even computing a trajectory under a single set of parameters can take hours.

The PD controller has the advantage of being able to start tracking instantly, whereas a LQR controller needs to do a short precomputation (8.4 - 46.9 sec in our examples). However, both controllers require a trajectory to track, which typically requires (non-instantaneous) presimulation. Also note that the linearizations employed by LQR could be performed with respect to a sparser set of "keyframe" shapes. Another positive aspect of PD is that it can be used without reduction. However, 256 x 256 unreduced 2D simulations with PD control are about 26x slower than reduced controlled simulations with LQR (Table 2). Our LQR controller generalizes to 3D fluids where timing differences will be even greater. The difference is also large with deformable simulations (dinosaur: 15,700x).

Deformable LQR vs PD experiment: In this experiment, we tracked a precomputed solid deformable dinosaur (rooted to ground) simulation, with zero position costs and non-zero final costs ($\alpha_Q = 0$, $\alpha_{Q_{\text{final}}} > 0$). We applied an initial force perturbation, and then tried to match LQR performance with PD. When PD gains (time-constant) were tuned such that PD dispensed equal effort as LQR, the PD simulation missed the target by a substantial margin. When PD gains were tuned so that PD simulation achieved the same final error as LQR, PD dispensed 6.3x the effort of LQR.

Fluid LQR vs PD experiment: Given a particular choice of position costs, control costs, and user disturbances, one can run the corresponding LQR controller and record the resulting particle tra-

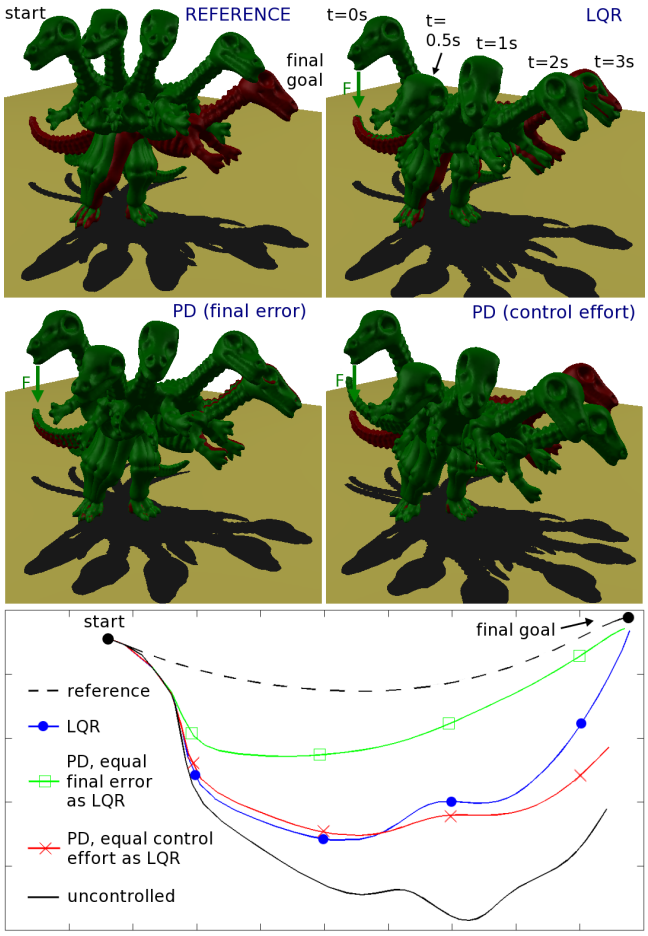

Figure 6: PD either invests more effort, or misses the target. *Top-Left: Reference trajectory. Top-Right: LQR simulation with a user perturbation. Middle-Left: PD simulation tuned to equal final error as LQR. Middle-Right: PD simulation tuned to equal control effort as LQR. Red frame is the final reference frame. Position cost was zero, final cost was non-zero, and control cost was set to a high value compared to final cost. Bottom: bird's eye view on the trajectory of a vertex at the top of dinosaur's head. PD that matched LQR in terms of final error consumed 6.3x the effort of LQR, with a very visible difference: LQR dynamics is richer and more compliant than PD (compare, e.g., to the uncontrolled curve), yet both LQR and PD reach the same final goal. Same spatial unit on both axes.*

jectories and the total amount of injected control. We selected one such set of parameters (with expensive control cost), and then attempted to match the LQR tracking performance using P and PD controllers. We did so by exploring PD gains k_Z, k_P, k_D (Equation 14) such that the total amount of injected PD control matched the LQR level. Note that LQR has two parameters in this case (1. ratio of control cost vs position cost, 2. particle position cost), P has two (k_Z, k_P), and PD has three (k_Z, k_P, k_D) parameters. We chose not to compare to a PID controller because PID has four parameters. The PD tuning was systematic, exploring all three gain parameter dimensions, 100 parameter values total, until we could no longer improve performance. We plot the results in Figure 7. We found that, for a fixed control budget, LQR is able to track the reference trajectory more closely than PD. Two of the plotted trajectories correspond to a P controller, which has the same number of parameters as LQR, but is suboptimal both to PD and LQR. The P controller, limited by the control budget, always failed to counter the first user applied force, causing the particle to swirl to the right. The other two curves show our best PD result, which is suboptimal to LQR, and another typical PD result. Note how in the last phase of motion PD can only asymptotically approach the reference trajectory (due to the limited budget), whereas LQR matches it almost exactly.

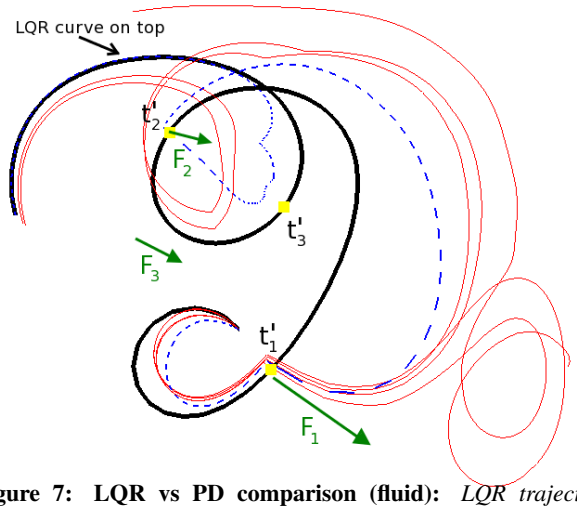

Figure 7: LQR vs PD comparison (fluid): *LQR trajectory matches the reference trajectory more closely than any of the PD trajectories. The thick solid black line gives the reference trajectory, the dashed blue line is the LQR trajectory, and the solid red lines are PD trajectories. Same setup as in Figure 5. Control cost was set to a high value, position cost was non-zero, and final cost was set to zero. Same amount of control for all LQR and PD curves.*

7 Conclusion

We presented physically based simulations that are directable in real-time. Our controller injects gentle forces that keep the system trajectories close to desired input trajectories. This makes it possible to generate simulations both with predictable outcome and variety. Real-time control is a natural complement to offline control which can only compute strategies needed for some complex outcome, but cannot provide simulation variety. As opposed to PD gains, the LQR cost parameters directly control the trade-off between tracking and amount of control. We successfully used the final cost to put emphasis only on matching the last simulation frame.

The controller can correct various imperfections in input data, such as moderate timing mismatches between full and reduced simulations. The simulation can loop endlessly by rewinding the tracked frame to the beginning of motion; any mismatches between the final and first frames are smoothed out by the controller. At designated moments in time (or triggered by external events), our controller can change the tracked goal to, say, a closest state to the current state along the entire trajectory, and then continue tracking the subsequent part of the trajectory. This can be done simply by changing the current tracked time to that of the closest configuration.

Our control cooperates naturally with the underlying reduced model. While the reduced model cannot produce trajectories significantly different from data used to derive the basis, the controller keeps the system near the data center-line. As such, the reduced model is kept in the region where it naturally performs well.

The control forces are computed by linearizing the system dynamics around the reference trajectory; their accuracy reduces when the runtime trajectory deviates significantly from the reference. We are

unable to break our solid deformable examples. Our particles immersed in fluids, however, sometimes do not recover under very large perturbations. This happens when the linearization in the $Y(t)$ matrix in Equation 13 is no longer able to predict the fluid velocity modes at the current particle position. Such occurrences can be prevented by limiting the magnitude of user-applied forces. Very few methods in control theory of nonlinear systems can provide theoretical guarantees on robustness. We can give the following intuition: a deformable system linearized around the reference state at some time $t = t_0$ is of oscillatory nature; even under large deviations the linearized internal forces will generally point in the correct direction. With fluids, however, the system is more chaotic and linearizations only weakly describe the behavior of the system. Due to the chaotic nature of the Navier-Stokes equations, it is inherently more difficult to control fluids than deformable solids.

In this paper, we track one trajectory in one global basis. In the future, real-time control should be applied to track trajectories organized into a motion graph [Kovar et al. 2002].

Appendix

A Linear quadratic regulator

We follow the derivation in [Stengel 1994], which gives optimal control for a *continuous* linear time-varying system (LTV)

$$\dot{z} = A(t)z + Bw(t) \tag{15}$$

with constant control over each timestep. Denote $A_i = A(t_i)$, and let position and control cost matrices at timestep i be Q_i and R_i, respectively, for $i = 0, \ldots, T-1$ (final cost is Q_T). The exact solution to controlling the LTV involves integrals of matrix exponentials:

$$\Phi_i(\tau) = e^{A_i\tau}, \quad \Gamma_i(\tau) = A_i^{-1}(e^{A_i\tau} - I)B, \quad \hat{\Phi}_i = \Phi_i(\Delta t), \tag{16}$$

$$\hat{\Gamma}_i = \Gamma_i(\Delta t), \quad \hat{Q}_i = \int_0^{\Delta t} \Phi_i^T(\tau) Q_i \Phi_i(\tau) d\tau, \tag{17}$$

$$\hat{M}_i = \int_0^{\Delta t} \Phi_i(\tau) Q_i \Gamma_i(\tau) d\tau, \quad \hat{R}_i = \int_0^{\Delta t} \Big(\Gamma_i^T(\tau) Q_i \Gamma_i(\tau) + R_i\Big) d\tau. \tag{18}$$

We approximate these integrals using Simpson's rule (with 10 integration points; adaptive rule could be used instead). Matrix exponentials were computed using Expokit [Sidje 1998]. Gain matrices K_i are obtained by time-stepping a Riccati ODE backwards in time:

$$K_i = -\Big(\hat{R}_i + \hat{\Gamma}_i^T P_{i+1} \hat{\Gamma}_i\Big)^{-1} \Big(\hat{M}_i^T + \hat{\Gamma}_i^T P_{i+1} \hat{\Phi}_i\Big), \tag{19}$$

$$P_i = \hat{Q}_i + \hat{\Phi}_i^T P_{i+1} \hat{\Phi}_i + (\hat{M}_i + \hat{\Phi}_i^T P_{i+1} \hat{\Gamma}_i)^T K_i, \quad P_T = Q_T. \tag{20}$$

Riccati equation can be stiff; however, sufficiently small timesteps always resolved it well in our examples. LQR core, fluid-specific and solid-specific parts of our implementation consisted of 402, 783 and 671 lines of C++ code, respectively. This includes all control-related components, both precomputation and runtime.

B LQR details: Reduced deformations

We use tangential Rayleigh damping: $\tilde{D}(p) = \alpha I + \beta \tilde{K}(p)$, where α and β are scalar damping parameters, and $\tilde{K}(p) = \partial \tilde{R}/\partial p$ is the gradient of the reduced internal forces. Our reduced state vector takes the form $z = [p^T, \dot{p}^T]^T =: [z_1^T, z_2^T]^T \in \mathbb{R}^{2r}$. Equation 9 in first-order form then reads

$$\dot{z} = \tilde{F}(z) + Bw = \Big[z_2^T,\ \big(-\tilde{D}(z_1)z_2 - \tilde{R}(z_1)\big)^T\Big]^T + \Big[0, w^T\Big]^T. \tag{21}$$

Matrices $A(t)$ and B of Equation 10 are

$$A(t_i) = \begin{bmatrix} 0 & I \\ -\beta(\frac{\partial \tilde{K}}{\partial z_1} : z_2^i) - \tilde{K}(z_1^i) & -\tilde{D}(z_1^i) \end{bmatrix}, \qquad B = \begin{bmatrix} 0 \\ I \end{bmatrix}, \tag{22}$$

for $[z_1^{i\,T}, z_2^{i\,T}]^T := z_i^{\text{ref}}$. All partial derivatives are evaluated at z_i^{ref}. Colon notation $H : a$ denotes tensor-vector multiplication (contraction; see, e.g., [Barbič and James 2005]). In practice, we sometimes omitted the damping-related Hessian tensor term $\partial \tilde{K}/\partial z_1$.

C LQR details: Reduced fluids

The matrices from Equation 13 are

$$A(t) = \hat{A}(z^{\text{ref}}(t)) + \nu \hat{D} + \Big[\hat{A}^{(1)} z^{\text{ref}}(t), \ldots, \hat{A}^{(r)} z^{\text{ref}}(t)\Big] \in \mathbb{R}^{r \times r} \tag{23}$$

$$B = [0 \quad I]^T \in \mathbb{R}^{(2N+r)\times r} \tag{24}$$

$$W(t) = [\Psi(p_1^{\text{ref}}(t))^T, \ldots, \Psi(p_N^{\text{ref}}(t))^T]^T \in \mathbb{R}^{2N \times r}. \tag{25}$$

Here, $\hat{A}^{(j)}$ denotes the j-th column of $\hat{A}$. Matrix $Y(t)$ is a $2N \times 2N$ block-diagonal matrix with one 2×2 block Y_i for each particle i :

$$Y_i(t) = \Big[\frac{\partial \Psi}{\partial x}_{|(x,y)=p_i^{\text{ref}}(t)} : z^{\text{ref}}(t),\ \frac{\partial \Psi}{\partial y}_{|(x,y)=p_i^{\text{ref}}(t)} : z^{\text{ref}}(t)\Big]. \tag{26}$$

References

BARBIČ, J., AND JAMES, D. L. 2005. Real-Time Subspace Integration for St. Venant-Kirchhoff Deformable Models. *ACM Trans. on Graphics (SIGGRAPH 2005) 24*, 3, 982–990.

BARZEL, R., AND BARR, A. H. 1988. A modeling system based on dynamic constraints. In *Computer Graphics (Proc. of ACM SIGGRAPH 88)*, 179–188.

BERGOU, M., MATHUR, S., WARDETZKY, M., AND GRINSPUN, E. 2007. Tracks: Toward directable thin shells. *ACM Trans. on Graphics (SIGGRAPH 2007) 26*, 3, 50:1–50:10.

BROTMAN, L. S., AND NETRAVALI, A. N. 1988. Motion interpolation by optimal control. In *Computer Graphics (Proc. of ACM SIGGRAPH 88)*, 309–315.

CAPELL, S., GREEN, S., CURLESS, B., DUCHAMP, T., AND POPOVIĆ, Z. 2002. A Multiresolution Framework for Dynamic Deformations. In *Proc. of the Symp. on Computer Animation (SCA)*, 41–48.

CAPELL, S., BURKHART, M., CURLESS, B., DUCHAMP, T., AND POPOVIĆ, Z. 2005. Physically based rigging for deformable characters. In *Symp. on Computer Animation (SCA)*, 301–310.

DA SILVA, M., ABE, Y., AND POPOVIĆ, J. 2008. Interactive simulation of stylized human locomotion. *ACM Trans. on Graphics (SIGGRAPH 2008) 27*, 3, 82:1–82:10.

DEBUNNE, G., DESBRUN, M., CANI, M.-P., AND BARR, A. H. 2001. Dynamic Real-Time Deformations Using Space & Time Adaptive Sampling. In *Proc. of ACM SIGGRAPH 2001*, 31–36.

ELCOTT, S., TONG, Y., KANSO, E., SCHRDER, P., AND DESBRUN, M. 2007. Stable, circulation-preserving, simplicial fluids. *ACM Trans. on Graphics 26*, 1 (Jan.).

FALOUTSOS, P., VAN DE PANNE, M., AND TERZOPOULOS, D. 1997. Dynamic Free-Form Deformations for Animation Synthesis. *IEEE Trans. on Vis. and Comp. Graphics 3*, 3, 201–214.

FALOUTSOS, P., VAN DE PANNE, M., AND TERZOPOULOS, D. 2001. Composable controllers for physics-based character animation. In *Proc. of ACM SIGGRAPH 2001*, 251–260.

FANG, A. C., AND POLLARD, N. S. 2003. Efficient synthesis of physically valid human motion. *ACM Trans. on Graphics (SIGGRAPH 2003) 22*, 3, 417–426.

FATTAL, R., AND LISCHINSKI, D. 2004. Target-driven smoke animation. *ACM Trans. on Graphics (SIGGRAPH 2004) 23*, 3, 441–448.

FEDKIW, R., STAM, J., AND JENSEN, H. W. 2001. Visual Simulation of Smoke. In *Proc. of ACM SIGGRAPH 2001*, 15–22.

GRINSPUN, E., KRYSL, P., AND SCHRÖDER, P. 2002. CHARMS: A Simple Framework for Adaptive Simulation. *ACM Trans. on Graphics (SIGGRAPH 2002) 21*, 3, 281–290.

HODGINS, J. K., AND POLLARD, N. S. 1997. Adapting Simulated Behaviors For New Characters. In *Proc. of ACM SIGGRAPH 97*, 153–162.

HODGINS, J. K., WOOTEN, W. L., BROGAN, D. C., AND O'BRIEN, J. F. 1995. Animating human athletics. In *Proc. of ACM SIGGRAPH 95*, 71–78.

ISAACS, P. M., AND COHEN, M. F. 1987. Controlling dynamic simulation with kinematic constraints, behavior functions and inverse dynamics. In *Computer Graphics (Proc. of ACM SIGGRAPH 87)*, 215–224.

JAMES, D. L., AND FATAHALIAN, K. 2003. Precomputing interactive dynamic deformable scenes. *ACM Trans. on Graphics (SIGGRAPH 2003) 22*, 3, 879–887.

KAČIĆ-ALESIĆ, Z., NORDENSTAM, M., AND BULLOCK, D. 2003. A practical dynamics system. In *Symp. on Computer Animation (SCA)*, 7–16.

KOVAR, L., GLEICHER, M., AND PIGHIN, F. 2002. Motion Graphs. *ACM Trans. on Graphics (SIGGRAPH 2002) 21*, 3, 473–482.

LI, R.-C., AND BAI, Z. 2005. Structure preserving model reduction using a Krylov subspace projection formulation. *Comm. Math. Sci. 3*, 2, 179–199.

LIU, C. K., HERTZMANN, A., AND POPOVIĆ, Z. 2005. Learning physics-based motion style with nonlinear inverse optimization. *ACM Trans. on Graphics (SIGGRAPH 2005) 24*, 3, 1071–1081.

MCNAMARA, A., TREUILLE, A., POPOVIĆ, Z., AND STAM, J. 2004. Fluid control using the adjoint method. *ACM Trans. on Graphics (SIGGRAPH 2004) 23*, 3, 449–456.

METAXAS, D., AND TERZOPOULOS, D. 1992. Dynamic deformation of solid primitives with constraints. In *Computer Graphics (Proc. of ACM SIGGRAPH 92)*, 309–312.

MÜLLER, M., AND GROSS, M. 2004. Interactive Virtual Materials. In *Proc. of Graphics Interface 2004*, 239–246.

POPOVIĆ, Z., AND WITKIN, A. P. 1999. Physically based motion transformation. In *Proc. of ACM SIGGRAPH 99*, 11–20.

POPOVIĆ, J., SEITZ, S. M., AND ERDMANN, M. 2003. Motion sketching for control of rigid-body simulations. *ACM Trans. on Graphics 22*, 4 (Oct.), 1034–1054.

RASMUSSEN, N., ENRIGHT, D., NGUYEN, D., MARINO, S., SUMNER, N., GEIGER, W., HOON, S., AND FEDKIW, R. 2004. Directable photorealistic liquids. In *Symp. on Computer Animation (SCA)*, 193–202.

SAFONOVA, A., HODGINS, J., AND POLLARD, N. 2004. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Trans. on Graphics (SIGGRAPH 2004) 23*, 3, 514–521.

SELLE, A., FEDKIW, R., KIM, B., LIU, Y., AND ROSSIGNAC, J. 2008. An unconditionally stable MacCormack method. *J. of Scientific Computing 35*, 2-3, 350–371.

SHARON, D., AND VAN DE PANNE, M. 2005. Synthesis of controllers for sylized planar bipedal walking. In *International Conference on Robotics and Automation (ICRA)*, 2387–2392.

SHI, L., AND YU, Y. 2005. Controllable smoke animation with guiding objects. *ACM Trans. on Graphics 24*, 1 (Jan.), 140–164.

SIDJE, R. B. 1998. Expokit: A Software Package for Computing Matrix Exponentials. *ACM Trans. on Mathematical Software 24*, 1, 130–156. www.expokit.org.

SIFAKIS, E., NEVEROV, I., AND FEDKIW, R. 2005. Automatic determination of facial muscle activations from sparse motion capture marker data. *ACM Trans. on Graphics (SIGGRAPH 2005) 24*, 3 (Aug.), 417–425.

SOK, K. W., KIM, M., AND LEE, J. 2007. Simulating biped behaviors from human motion data. *ACM Trans. on Graphics (SIGGRAPH 2007) 26*, 3, 107:1–107:9.

STAM, J. 1999. Stable fluids. In *Proc. of ACM SIGGRAPH 99*, 121–128.

STENGEL, R. F. 1994. *Optimal Control and Estimation*. Dover Publications, New York.

SULEJMANPASIĆ, A., AND POPOVIĆ, J. 2005. Adaptation of performed ballistic motion. *ACM Trans. on Graphics 24*, 1 (Jan.), 165–179.

TEDRAKE, R. L. 2004. *Applied Optimal Control for Dynamically Stable Legged Locomotion*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA.

THÜREY, N., KEISER, R., PAULY, M., AND RÜDE, U. 2006. Detail-preserving fluid control. In *Symp. on Computer Animation (SCA)*, 7–15.

TREUILLE, A., LEWIS, A., AND POPOVIĆ, Z. 2006. Model reduction for real-time fluids. *ACM Trans. on Graphics (SIGGRAPH 2006) 25*, 3, 826–834.

WITKIN, A., AND WELCH, W. 1990. Fast animation and control of nonrigid structures. In *Computer Graphics (Proc. of ACM SIGGRAPH 90)*, 243–252.

WOOTEN, W. L., AND HODGINS, J. K. 2000. Simulating leaping, tumbling, landing and balancing humans. *International Conference on Robotics and Automation (ICRA)*, 656–662.

YIN, K., CLINE, M., AND PAI, D. K. 2003. Motion perturbation based on simple neuromotor control models. In *Pacific Conference on Computer Graphics and Applications (PG)*, 445–449.

YIN, K., LOKEN, K., AND VAN DE PANNE, M. 2007. SIMBICON: Simple biped locomotion control. *ACM Trans. on Graphics (SIGGRAPH 2007) 26*, 3, 105:1–105:10.

ZHU, Y., AND BRIDSON, R. 2005. Animating sand as a fluid. *ACM Trans. on Graphics (SIGGRAPH 2005) 24*, 3, 965–972.

ZORDAN, V. B., AND HODGINS, J. K. 2002. Motion capture-driven simulations that hit and react. In *Symp. on Computer Animation (SCA)*, 89–96.

ZORDAN, V. B., MAJKOWSKA, A., CHIU, B., AND FAST, M. 2005. Dynamic response for motion capture animation. *ACM Trans. on Graphics (SIGGRAPH 2005) 24*, 3, 697–701.