Interactive Spacetime Constraints: Wiggly Splines

Michael Kass John Anderson
Pixar Animation Studios

Abstract

The Spacetime Constraints formulation attempts to marry the realism of physical simulation with the controllability of keyframe animation, but the resulting nonlinear optimization problems are generally extremely complicated and slow to solve. Here we explore the range of Spacetime Constraints problems that give rise to quadratic optimization functions solvable with linear systems of equations. We find that they generalize traditional splines to encompass oscillatory solutions. These problems can be solved at full frame rates, giving animators a keyframe animation tool with built in knowledge of a physical model. In addition to the splines themselves, we also introduce a new analysis method to extract oscillatory behavior from physical simulations in a way that can be connected naturally to the splines. It turns out that in order to have sufficient control of the frequency response of splines, we solve the Spacetime Constraints problems over the domain of complex numbers. As a consequence, our solutions have an imaginary part in addition to the real part. The imaginary part defines a phase angle that we show is very useful for controlling and generalizing oscillatory behavior whether extracted from simulation data or authored by hand.

CR Categories: I.3.7 [Computer Graphics]: Three Dimensional Graphics and Realism—Animation; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically-Based Modeling G.1.6 [Numerical Analysis]: Optimization

Keywords: Spacetime Constraints, Splines

1 Introduction

In traditional computer graphics animation, the motion of an object or character in a scene is specified by a set of splines which describe the way its parameters change over time. This technique provides excellent control of the motion since the splines simply approximate or interpolate a series of key values or knots carefully crafted by an animator. Unfortunately, the traditional technique provides little help to an animator in producing physically realistic motion. Any physical realism in the motion comes not from the mathematics of the splines themselves, but from the talent, skill, insight and sweat of the animator [Lasseter 1987].

When greater physical realism is desired, or when the required animator time would be too expensive, direct physical simulation of the underlying physics is sometimes used to create computer graphics motion. While physical simulation can create compelling realism of motion, it is often very difficult to control, and simulations can produce surprising or unintended results. Determining the proper applied forces and physical parameters necessary to achieve a particular desired effect is generally a formidable challenge.

The central problem in ordinary simulation is that simulation equations march forward in time from initial conditions to a final result. Forces lead to accelerations which are integrated to get velocities and positions over time. If the result at some point in time is not what is desired, it is usually very difficult to determine how the forces and parameters before hand have to be changed to bring the motion closer to the goal.

In 1988, Witkin and Kass [Witkin 1988] developed a technique called *Spacetime Constraints* to try to marry the physical realism of simulation with the controllability and predictability of traditional splines. Their idea was to have animators continue to specify key values of parameters over time, but interpolate the motion in the most physically realistic way possible subject to these constraints. Instead of demanding that Newton's second law, $f = ma$ be satisfied all the time as in traditional simulation, Spacetime Constraints minimizes the deviation from the second law while guaranteeing the interpolation constraints set by the animators are met.

Although the approach of Spacetime Constraints has the potential to provide the elusive combination of realism, predictability and control sought after in computer graphics production, it has thus far remained the realm of academic research for at least two reasons. Foremost among the difficulties in using Spacetime Constraints is the complexity involved. The full equations of motion for even the simple planar Luxo Lamp model used in the original Spacetime Constraints paper are tremendously complicated and required an elaborate symbolic math-based compiler to evaluate. Even if the equations of motion of interesting character models can be managed, a second obstacle is the execution time of the required nonlinear optimization. Animators are loath to use any technique which slows an animation system down from interactive speeds, and it will be many years before full character-level Spacetime Constraint problems can be solved fast enough with existing formulations.

A variety of researchers have applied the Spacetime Constraints approach to specific domains, used different optimization methods, tried accelerate it, or addressed other weaknesses in the original formulation (eg. [Cohen 1992], [Ngo 1993], [Liu 1994], [Witkin 1995], [Rose 1996], [Gleicher 1997], [Popović 1999], [Popović 2000], [Fang 2003], [Treuille 2003], [Safonova 2004]). Good summaries of the specific contributions and evolution of the ideas can be found in [Fang 2003] and [Safonova 2004]. Here we take a very different approach. Instead of looking for an optimization expression which will capture the full physics of the underlying problem, we choose to examine how far we can go using only optimizations that can be computed rapidly enough to be used in place of traditional splines. By contrast with the recent work of Fang and Pollard [Fang 2003], for example, who report clever optimizations to reduce running time down to a few minutes, we seek to simplify the problems more radically to the point where we can achieve true interactive update rates of 30 frames per second – thousands of times faster.

In particular, we limit ourselves only to formulations which give rise to simultaneous linear equations. While this restriction, of course, is severe, it still allows us to address a wide range of oscillatory phenomena while retaining the interactive performance and predictability of traditional splines. Since they arise from linear or linearized Spacetime Constraints problems, we refer to the solu-

tions as *Linear Spacetime Constraint Splines* (LSC splines), or even "Wiggly Splines" because of their penchant for oscillation. These splines have a collection of attractive features:

- They generalize traditional splines.
- They break the time symmetry of traditional splines, incorporating damping.
- They accurately characterize real physical systems near equilibrium.
- They can be computed in constant time per frame with banded linear systems.
- They can be chosen to have unconditional stability.

In addition, if the LSC splines are allowed to be complex valued, then their solution yields not only the physically meaningful real part, but also an imaginary part, and hence a phase angle. The existence of the phase angle allows a single animation curve to control a collection of different animation parameters related by different phase lags.

Traditional splines have become very powerful tools in the hands of skilled animators, but their behavior is only very tenuously related to the underlying physics of motion. They exhibit perfect time symmetry, for example, while real motion is decidedly asymetric in time. LSC splines offer a generalization retaining all the power of traditional splines, while adding an inherent ability to produce common animation phenomena like overshoot due to follow through, passive damping, and motion which is fundamentally oscillatory. A large part of achieving a high degree of realism in character animation is getting the secondary motion to look right, and oscillatory phenomena such as these are the dominant form of secondary motion.

The rest of the paper is structured as follows. In section 2, we show that traditional piecewise cubic splines fall out as a special case solution of a Spacetime Constraints problem with no applied force. In section 3, we point out that individual muscles, and general systems near equilibrium give rise to systems like damped mass-spring oscillators with linear restoring forces. We show that the corresponding Spacetime Constraints problem can be solved with a banded linear system. In section 4, we provide the view from Digital Signal Processing, which allows us to accurately characterize the stability and resonances of the discrete system. In section 5, we show that we can broaden the resonance of LSC splines by using a complex-valued solution. This not only makes the physically-meaningful real part of the curve easier to manipulate, but also provides additional opportunities to use the phase of the animation curve to control additional animation parameters. In section 6, we show how to add in the effects of external forces, and then in section 7 we show how the full technique can be applied both to procedural animation models, and to models derived from simulation data.

2 Generalizing Traditional Splines

In the Spacetime Constraints formalism, we begin with a physical system, add constraints provided by the animator, and then, subject to these constraints, minimize an objective function which penalizes non-physical and inefficient motion. Let us first consider passive systems where efficiency is not an issue. In this case, the objective function will simply be a measure of the departure of the motion from passive physics.

A physically accurate simulation will follow Newton's second law of motion: $f = ma$. Consider the simple example of a point mass moving in one dimension with position given by $x(t)$. We expect the acceleration $a = \ddot{x}$ to be equal to the known force f divided by the mass m. If we observe the mass point accelerating in a way inconsistent with the known forces, then we can explain the observed motion by positing a mysterious "jet engine force" $J(t)$ acting on the mass. The most physically realistic motion will be one where the jet engine forces are as small as possible.

$$f + J = ma = m\ddot{x} \tag{1}$$

$$J = m\ddot{x} - f \tag{2}$$

Without loss of generality, we can choose units so m = 1, and then we have

$$J = \ddot{x} - f \tag{3}$$

In order to minimize the force of the jet engine, we need a norm to rank different possible jet engine functions $J(t)$, so we can choose among different possible paths of motion $x(t)$. There are a variety of choices here including penalizing the average power output of the jet engine or the amount of work it puts out, but in order to keep the optimization problem linear, we follow [Witkin 1988] and minimize the L^2 norm of the force.

$$E = \int J^2 dt \tag{4}$$

Clearly, if $E = 0$, then J must be uniformly zero, and the motion is completely physical with no jet engine force at all.

In general, we will be concerned with situations where the known applied force f is a linear combination of the motion x and its derivatives, but for the moment, consider what happens when the known applied force f is zero. Then the jet engine force in equation 3 becomes $J = \ddot{x}$ and the function to be minimized is:

$$E = \int \ddot{x}^2 dt. \tag{5}$$

Equation 5 is the minimum principle from which traditional cubic interpolatory splines are derived [Bartels 1987]. It is well known and easy to prove using calculus of variation that the minimization of equation 5 gives rise to piecewise cubic polynomial solutions, and these are widely used throughout computer graphics. Generalizing the continuous solution to situations where the applied force f is a linear differential operator on x, however, would lead to piecewise exponentials.

Our motivation in computing the minimum is to construct animation curves. Typically, animation curves need only be evaluated at discrete integer frame times and perhaps shutter-close times for motion blur. The existence of a continuous differentiable function is generally not required. As a result, we find it easier to express the minimization problem in the discrete domain and solve it there, although there may be applications for which the true piecewise continuous exponential solution may be preferable.

Discretizing equation 5 with standard finite differences leads to the following function to be minimized:

$$E_s = (1/h)^2 \sum_{i=1}^{n} (x_{i-1} - 2x_i + x_{i+1})^2 \tag{6}$$

where h is the time separation between samples. The minimum will occur when the gradient vanishes

$$\frac{\partial E_s}{\partial x_i} = 0 \tag{7}$$

For each i away from the boundaries, three terms of the sum in equation 6 lead to non-zero derivatives, and they combine so that

equation 7 leads to a banded linear system with bandwidth 5. Equations of this form can be solved using standard techniques [Golub 1983] [Press 1986] in time proportional to n, the number of samples of x_i.

2.1 Constraints

In order to make the solution of equation 7 interesting, we need to add some constraints. The primary constraint, of course, is the ability to set a particular value at a particular time – an interpolation constraint. Setting up interpolation constraints with the finite difference formalism is not at all difficult. Suppose we want to establish the constraint that $x_j = v$ for some j. All we have to do is drop x_j from equation 6 and replace it with the constant value v. The resulting linear system will have one fewer equation.

In addition to setting values, animators are used to establishing tangent constraints. This is also easy to handle in the finite difference formulation. Let g be the desired slope at sample i. Then we can add a penalty term to the optimization function as follows:

$$E = E_s + E_t \tag{8}$$

$$E_t = c_t((1/h)(x_{i+1} - x_i) - g)^2 \tag{9}$$

where c_t is a constant that sets the strength of the tangent penalty.

Note that with just interpolation constraints, the solution of the minimization problem in equation 6 produces a spline with global support. In fact, it converges to the same curve as an interpolating C^2 cubic, because it minimizes the same energy. If tangent constraints are introduced at the interpolation constraints, however, the effective support of the spline becomes more and more local as the penalty constant c_t is increased. If c_t is large enough, and if the finite difference sampling is adequate, the global optimization will produce a result indistinguishable from locally supported piecewise cubics such as Bezier or Catmull-Rom splines (depending on the computation used to set the tangent constraints).

Interestingly, when the optimization is done numerically with finite differences, the constant c_t allows us to choose on a knot by knot basis whether we want to maximize smoothness by allowing global support, strictly enforce local support, or do something in between. It provides some additional freedom difficult to achieve with the usual analytic spline formulations.

The chief advantage of the usual analytic splines over the method described here is that strict local support allows values of the analytic splines in an interval to be calculated from a fixed number of neighboring knots. If one is willing to forego this property, the finite difference splines described here still require only constant time per sample to calculate, More importantly, by allowing the applied force to be non-zero, they allow a wide range of interesting new behavior.

3 Mass-Spring Oscillator

3.1 Why the Damped Mass and Spring?

We choose to investigate Spacetime Constraint solutions of linear differential equations for two main reason. First, they very commonly approximate physical situations of interest. Second, they are very tractable, both from the point of view of computing the result, and characterizing the expected behavior.

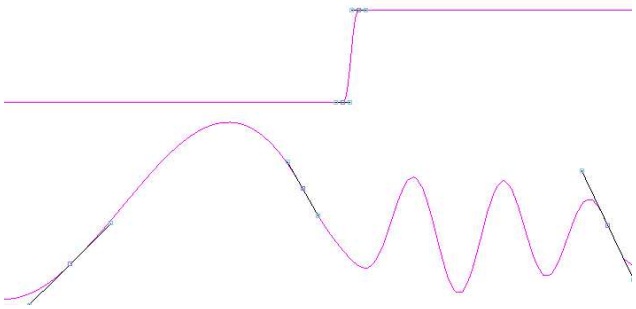

Figure 1: Top curve: Resonant frequency. The resonance begins at zero, and then changes abruptly to a non-zero value. Bottom curve: Linear Spacetime Spline with three interpolation and tangent constraints.

Linear differential equations arise commonly for physical systems displaced only slightly from equilibrium. Consider any mechanical system whose potential energy is given by a function $V(q_1, \ldots, q_n)$ of some generalized coordinates. Since the displacement is understood to be small, we can expand the energy function around the equilibrium using a Taylor series. Let us introduce new coordinates η_i to denote the displacement of the system from its equilibrium position $(q_{01}, \ldots, q_{0n})$.

$$q_i = q_{0i} + \eta_i \tag{10}$$

Potential energy functions have an arbitrary zero, so without loss of generality, we can shift the zero to coincide with the equilibrium position Q_0, making the first term of the Taylor series vanish. The second term will also vanish because the gradient is zero at the equilibrium. As a consequence, the first non-zero term of the expansion will be due to the second derivatives of the energy.

$$V(q_1, \ldots, q_n) \approx \frac{1}{2} \sum_{i,j} \left(\frac{\partial^2 V}{\partial q_i \partial q_j} \right)_{q_0} \eta_i \eta_j \tag{11}$$

Quadratic energies such as these imply linear forces since $f = -\nabla E$. The resulting systems will exhibit oscillatory solutions [Goldstein 1980]. If we switch from using the ηi to normal coordinates (see [Goldstein 1980] for details – in simple cases the normal coordinates are just the eigenvectors of $\partial^2 V / \partial q_i \partial q_j$), then the resulting equations correspond to the motion of damped springs in each of the normal coordinates.

The fact that any physical system, displaced slightly from equilibrium and viewed in the proper coordinate system, will exhibit the passive physics of a damped spring suggests that this type of behavior is fundamental and universal and that we would benefit from incorporating it into our animation interpolation schemes.

There is an additional reason for considering the damped spring model to be fundamental in animation. While there are certainly important nonlinearities in muscle behavior, the linear "Active State Muscle Model" [McMahon 1984] has been a very effective reference for the mechanical properties of muscles. Essentially, it consists of a set of linear elements which create a movable equilibrium with an adjustable tension around that equilibrium. The passive behavior of such a system is given by a damped spring oscillator forced by a changing equilibrium.

3.2 Applying Spacetime Constraints

If we return to the Spacetime Constraints problem of equations 3 and 4, but introduce the physical model of a damped spring, then the

applied force, instead of dropping out, becomes key to the behavior of the system. In a standard damped spring, the applied force is given by:

$$f = -kx - \gamma\dot{x} \tag{12}$$

Plugging this expression for the force into equation 4 yields the optimization function

$$E = \int (\ddot{x} + \gamma\dot{x} + kx)^2 dt. \tag{13}$$

Proceeding as before by substituting finite differences for the derivatives, we get

$$E_s = \frac{1}{h^2} \sum_{i=1}^{n} \left(m(x_{i-1} - 2x_i + x_{i+1}) + h\gamma(x_{i+1} - x_i) + h^2 x_i \right)^2 \tag{14}$$

While this expression is a good deal more complicated than the case of the ordinary spline, it is still a quadratic objective function. As a result, equation 7 once again gives rise to a banded set of simultaneous linear equations with a bandwidth of five.

We have embedded our solution to equation 14 into a traditional spline editor, using the original interface for interpolation and tangent constraints to modify the optimization problem as described in section 2. In our implementation, we can achieve over 30fps update rates, on a personal computer, and the LSC spline computation takes less execution time than the time required to draw the result. Figure 1 shows an example of an LSC spline being used. The upper curve is a traditional spline used to control the spring constant, and the lower curve is a Linear Spacetime Spline. On the left side of the figure, the spring constant and damping are zero, so the bottom curve behaves like a familiar piecewise cubic spline. In the middle, the upper curve, representing the spring constant, abruptly increases and shifts the resonance. As a consequence, on the right side, the curve oscillates between the second and third interpolation constraints. Note that the curve has a series of inflection points between the second and third constraints, and is clearly not representable on that span as a single cubic.

4 Digital Signal Processing View

The continuous differential equation that results from equation 12 when $J = 0$ is stable for all positive values of k and γ. Unfortunately, once finite differences are substituted for continuous derivatives, stability is no longer guaranteed. In order to guarantee stability, we recast the equations in terms of digital signal processing.

Going back to equation 12 and substituting finite differences for the derivatives yields

$$f_i = -kx_i - \gamma(x_{i+1} - x_i)/h. \tag{15}$$

Combining this with equation 3 and using a finite difference for $\ddot{x}$ gives us

$$x_{i+1} - ax_i - bx_{i-1} = J_i \tag{16}$$

where a and b are functions of the spring constant k and the damping constant γ. In the language of Digital Signal Processing, equation 16 represents a second order recursive (IIR) filter where the input of the filter is the jet engine force computed by the optimization. We can now re-write the objective function in terms of a and b, and then use IIR filter design techniques to set a and b to values that guarantee stability. The re-written objective function is

$$E_s = \sum_{i=1}^{n} (x_{i+1} - ax_i - bx_{i-1})^2. \tag{17}$$

The response of the filter in equation 16 to the jet engine forcing is fully characterized by the system function of the filter, which is given by [Oppenheim 1975]

$$H(z) = \frac{1}{1 - az^{-1} - bz^{-2}}. \tag{18}$$

where z is a complex parameter. When evaluated on the unit circle, the system function gives the frequency response of the filter. If we factor the system function, then we will find two roots of the denominator polynomial

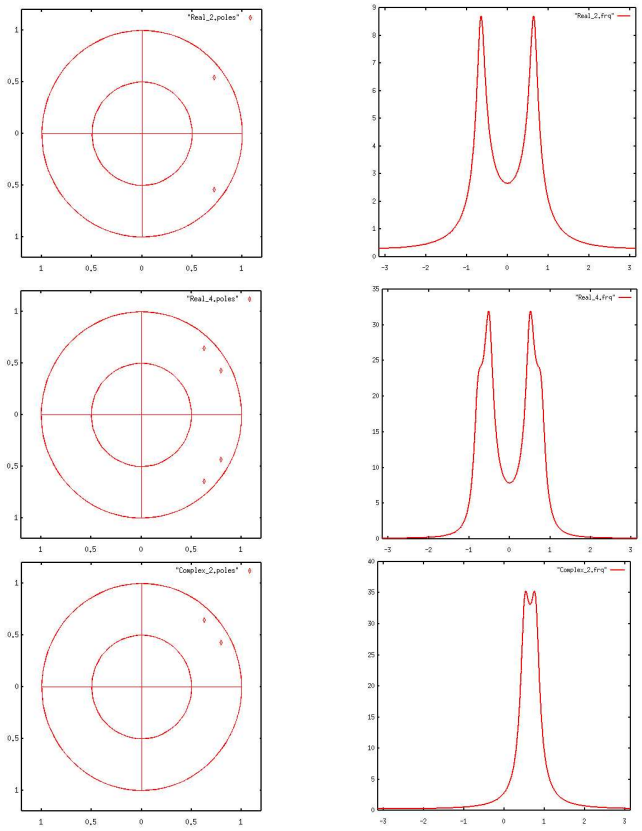

Figure 2: Left: pole locations. Right: frequency response. Top: Two pole real filter. Middle: Four pole real filter. Bottom: Two pole complex filter

$$H(z) = \frac{1}{(1 - \lambda_0 z^{-1})(1 - \lambda_1 z^{-1})} \tag{19}$$

These roots, λ_0 and λ_1 are known as the poles of the filter, and they provide the information required to control the frequency response and the stability of the filter. If both poles are inside the unit circle, the filter will be stable.

In practice we will generally design our system by chosing the poles to have the desired resonance and damping. Then we will compute a and b from λ_0 and λ_1 using the expressions:

$$a = \lambda_0 + \lambda_1 \tag{20}$$

$$b = -\lambda_0 \lambda_1. \tag{21}$$

In order for a and b to be real, the poles must either be real, resulting in a purely damped solution with no oscillation, or must be complex conjugates. This is an important restriction.

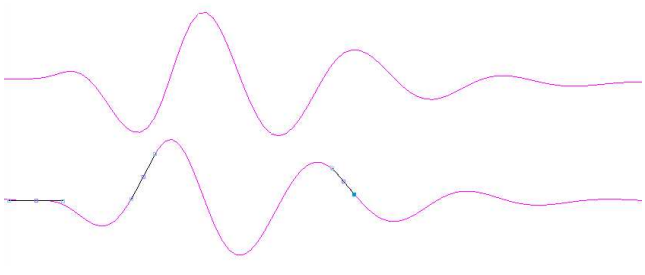

Figure 3: Complex-valued spline: The bottom curve shows the real part resulting from three interpolation and tangent constraints. The top curve is the computed imaginary part.

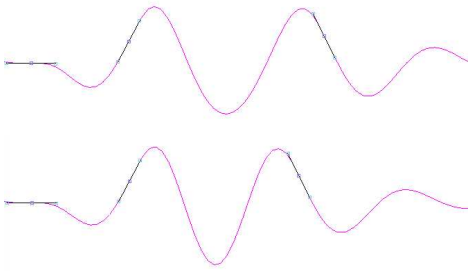

Figure 4: The complex spline is able to adjust its frequency as the interpolation constraints are moved.

5 Complex Valued Spline

In actual practice, although the Spacetime Constraint mass-spring oscillator will dutifully interpolate positional and tangent constraints, it can be difficult to coax into a desired curve. The main difficulty is that its resonance is very sharp. As a result, the spline is over-eager to oscillate at precisely the resonant frequency. If the constraints are moved in time so that all the time intervals are increased in size by a small percentage, then instead of the solution stretching slightly, the frequency of oscillation will remain unchanged, and the curve will manage to satisfy the constraints by adding an additional low-frequency component.

In a general Spacetime Constraints setting, one could allow the resonant frequency to be a parameter of the optimization, so altering the timing of the constraints would tend to change the frequency of oscillation. Unfortunately, all the obvious ways of allowing the resonant frequency to be computed by the optimization result in strongly nonlinear Spacetime Constraints problems. Within the domain of linear Spacetime Constraints problems, the way to mitigate this effect is to widen the resonance of the linear filter, turning it into as close to an ideal bandpass filter as possible with the passband centered around a frequency chosen by the animator.

The second order mass-spring system provides limited degrees of freedom for shaping the frequency response because the two poles are required to be complex conjugates in order to ensure that the filter coefficients are real. The top of figure 2 shows a representative sample pair of conjugate pole locations, and the corresponding of the frequency response. Note that the resonance is very sharp. At the same time, there is significant response at zero frequency where the influence of the two conjugate poles interact.

The usual way to gain additional degrees of freedom to shape the frequency response of a filter is to make it higher order, for example, by adding poles. The middle row of figure 2 shows a representative example of what can be accomplished with this technique. It shows a fourth-order filter with pairs of poles centered around the previous poles and exhibits the resulting frequency response. The resonance has been broadened, but the response at zero frequency is still unacceptably high.

An alternative solution to the problem is to allow the filter coefficients to be complex. Then the poles need not be conjugate, and a very different frequency response results. The bottom row of figure 2 shows a representative example of this technique. The pass band has been widened while keeping the response at zero frequency much lower.

In principle, it is possible to achieve the same result with a fourth-order real filter as the second-order complex filter, if you can avoid exciting the oscillatory modes represented by the conjugate poles. In practice, this is very difficult, particularly in the optimization setting, where constraints may be specified at arbitrary times. As a consequence, the kind of behaviour we seek is much more easily achieved with a complex-valued spline than with a real one.

The computation of the complex-valued spline begins just as before. The animator selects the resonant frequency and degree of damping desired, which together determine a point on the complex plane. The complex poles λ_0 and λ_1 are then generated by rotating the point clockwise and counterclockwise by an amount that determines the bandwidth of the filter. From λ_0 and λ_1, we compute the coefficients a and b of our recursive filter. This time, however, the coefficients a and b and the resulting spline will all be complex. The new objective function will be

$$E_s = \sum_{i=1}^{n} |x_{i+1} - ax_i - bx_{i-1}|^2 . \tag{22}$$

where the sum of squares in equation 17 has been replaced with a sum of squared complex magnitudes. The number of degrees of freedom in the optimization has been doubled, since each value of x has both a real and an imaginary part. The condition for the minimum is that the partial derivative of the objective function E with respect to both the real part and the imaginary part of each x_i must vanish.

Figure 4 shows the real (below) and imaginary (above) parts of the complex-valued LSC spline solution. All the interpolation constraints are applied to the real part, and for most purposes, this is the only part of interest. The existence of the imaginary part, however, does open up some interesting possibilities as will be seen in section 7.

Figure 4 shows that the widening of the resonance with the complex LSC spline is effective. Moving the third interpolation constraint from its position in the bottom curve to its position in the top curve causes the curve to stretch as expected, instead of finding some other, less intuitive, way of meeting the constraints.

6 External Forcing

In section 3, we assumed that the only force on the mass-spring system was due to the passive dynamics. In the case of character animation, we are generally interested a combination of passive dynamics along with active muscle action due to intent.

If the equilibrium position of the spring is movable and given by the function $y(t)$, then we have

$$a = \ddot{x} + \ddot{y} \tag{23}$$

$$j = \ddot{x} + \ddot{y} - f \tag{24}$$

Making this change to j results in the following refinement to equation 22.

$$E_s = \sum_{i=1}^{n} |x_{i+1} - ax_i - bx_{i-1} + \ddot{y}|^2 \tag{25}$$

Figure 5 shows the use of the changing equilibrium position. The bottom curve shows an animated curve for the function $y(t)$ created with traditional splines. The middle curve shows the LSC solution. No interpolation constraints were applied, so this represents the pure forward dynamics that could be computed using an ordinary differential equation (ODE). The top curve shows the fundamental difference between the LSC spline and an ODE solution. An interpolation constraint has been added to the middle of the curve, illustrating that unlike ODE solutions, the LSC spline is editable at any point.

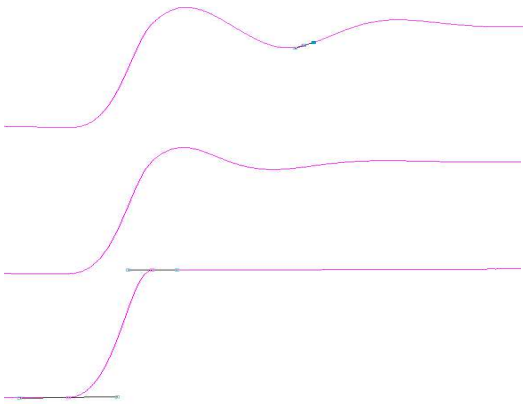

Figure 5: Bottom: Changing equilibrium value. Middle: LSC spline with no constraints, showing passive response. Top: LSC spline edited with an interpolation constraint.

7 Animating with LSC Splines

7.1 Controlling Procedural Models

While we introduced the imaginary part of our LSC spline x in order to craft the desired frequency response, it has additional value for controlling animation. If we represent x in polar imaginary coordinates using the identity

$$x = re^{i\theta} \tag{26}$$

then in addition to the real part of x which the animator manipulates directly, we know *theta* which defines a phase angle at each point along the curve. Very commonly, during oscillatory motions, there are degrees of freedom that oscillate with different phase relationships to the driving motion. Knowing θ with our complex-valued spline, we can derive those very easily. Let ϕ be the desired phase difference from our controlling oscillation x. We can derive a family of phase-shifted curves x_ϕ using the formula

$$x_\phi = \Re(re^{i(\phi+\theta)}) = \Re(e^{i\phi}x) \tag{27}$$

where $\Re$ denotes the real part of a complex number.

Figure 6 shows a procedural model of a tail being animated in this way. The original LSC spline x is used to drive the first bend angle at the base of the tail. Scaled and phase-shifted versions of the spline are used to drive all the other bend angles. Even with just a few animation knots, it is possible to get very interesting and lively animation out of models like this. The accompanying mpeg movie shows an example of an animation curve and the resulting the real-time motion of the tail.

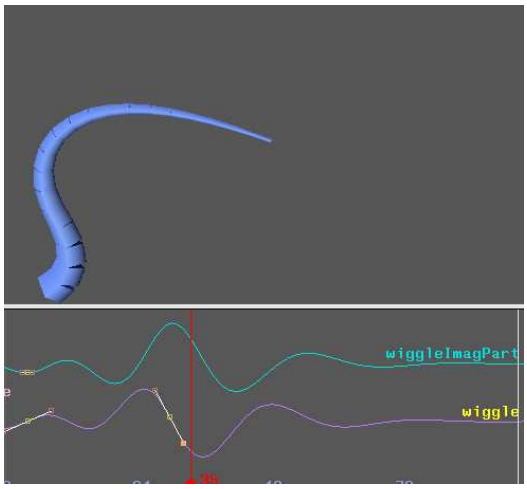

Figure 6: Animation of a procedural tail

7.2 Controlling Simulations

In section 3.1, we noted the universality of mass-spring systems for describing the oscillations of general physical systems displaced slightly from their equilibria. In the original given coordinates, the motion of a system around its equilibrium may appear to be extremely complicated, but if the energy is approximately quadratic, there exist a set of normal coordinates that allow for a much simpler description. The normal coordinates, which are simply vectors of weights for the parameters used to pose a model, provide a set of preferred directions in the parameter space. When viewed in these preferred directions, physically-accurate unforced motion reduces to damped simple harmonic motion. LSC splines, of course, are designed to handle just this type of situation. If we want to apply LSC splines to control and animate these oscillations, the chief remaining obstacle is to compute the normal coordinates.

One way to compute the normal coordinates for a physical system is to construct differential quadratic approximations for both the potential and kinetic energy of a system and employ "simultaneous diagonalization of two quadratic forms" [Goldstein 1980]. The resulting eigenvectors provide the required coordinates. For our purposes, however, the method has two limitations. First of all, quadratic approximations based on energy derivatives are only accurate for small displacements, while animators tend to make frequent use of large displacements. Second of all, if we directly compute the normal coordinates for a system with a large number of degrees of freedom, we will get a correspondingly large number of normal coordinates, and it is not clear how to pick the relevant ones, present them to animators, or control their blending.

For animation purposes, we choose to start by trying to characterize a small set of oscillations that are meaningful to animators. For example, with the character in figure 7, we develop canonical up-down and left-right oscillations. The idea is to create physical conditions that excite the desired oscillations and then discover the normal coordinates by analyzing the time series of simulation data. Since we excite oscillations with large displacements, the analysis will discover the best linear approximation to a large displacement,

rather than a small one. Having discovered the proper normal coordinates with our analysis, we can then control their oscillations with an LSC spline. In effect, the LSC spline makes it possible to begin with a canonical oscillation discovered through simulation and then generalize it, adjusting the frequency and damping as desired for cinematic effect.

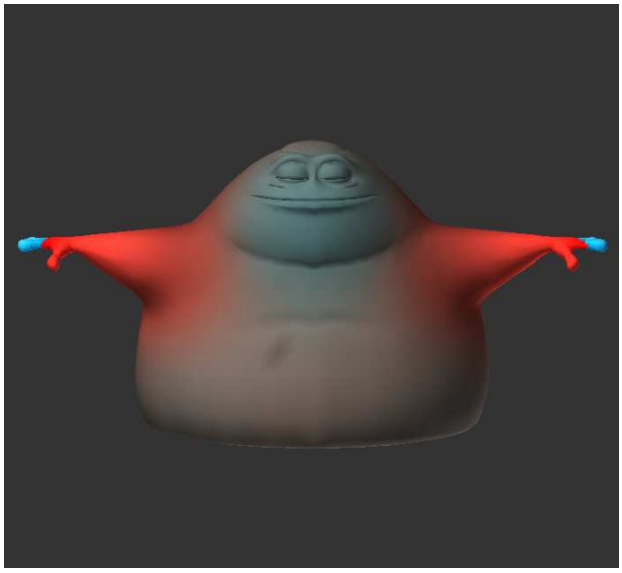

Figure 7: Deformable character with phase map for up-down oscillations. The phase of the normal coordinate is encoded into the color of each point.

Let $S_{i,j}$ represent the full state of a physical simulation where the first index ranges over the n time samples and the second index ranges over the m different parameters that describes the instantaneous state of the model. We will refer to i as the temporal index and j as the spatial index of our state. Our hypothesis is that the motion can be represented reasonably well as the sum of a set of uncoupled oscillations in the normal coordinates. Let $N_{k,j}$ represent the k normal coordinate vectors. If $R_{k,i}$ represents the displacement over time of the kth normal coordinate, then the hypothesis that the data can be explained as the sum of independent oscillations in the normal coordinates can be written

$$S_{i,j} = \sum_k R_{k,i} N_{k,j}. \tag{28}$$

There is a well-established method to compute the best least-squares approximation to equation 28 from a time series, and it is known as principal components analysis (PCA). The least-squares best normal coordinates $N_{k,j}$ are simply the eigenvectors of the covariance matrix SS^T while the least-squares best temporal displacements of these coordinates $R_{k,i}$ are just the eigenvectors of the matrix $S^T S$.

A weakness of the standard PCA is that it is not very good at discovering propagating phenomena. The problem is that travelling waves fundamentally involve 2-dimensional eigenspaces with a single eigenvalue, and these are degenerate when working with real values. If PCA can identify the travelling wave, it will show up as a pair of sinusoidal eigenvectors 90 degrees out of phase with each other, but the absolute phase of the eigenvectors is arbitrary and will be determined primarily by noise.

Instead of looking through the eigenvectors and searching for real-valued pairs that represent propagating waves, a much better approach is use a modified version of PCA that does all the arithmetic over the complex domain and finds propagating phenomena directly. Developed by climate researchers in the early 1980s [Rasmusson 1981] [Anderson 1983], it has become known as Complex Principal Components Analysis or CPCA. The basic idea is to take the real data $S_{i,j}$ and extend it into a complex representation by using a Hilbert transform to add an imaginary part to each real-valued time series. Then the same mathematical analysis as standard PCA can be done over a complex data set and will reveal phase relationships within the time series.

Hilbert transforms have not been widely used in Computer Graphics, but their theory can be found in standard texts (e.g. [Oppenheim 1975]). Suppose we have a real-valued function of time $f(t)$ which is zero for all $t < 0$. Such functions are known as *causal* functions. If we would like a complex-valued function of time whose real part matches $f(t)$ and is well-behaved enough that its Taylor series converges everywhere, there is a single choice. The unique extension to a complex-valued analytic function is $f(t) + i\hat{f}(t)$ where $\hat{f}(t)$ is the Hilbert transform of $f(t)$. There are a variety of ways of computing Hilbert transforms. For a periodic signal, the Hilbert transform can be obtained by simply shifting all the Fourier components by 90 degrees. Where boundaries are involved, however, more care is required. Oppenheim [1975] goes into some depth about the issues involved.

If we take the original real data $S_{i,j}$ and added in an imaginary part given by its Hilbert transform, we get a complex time series $\tilde{S}_{i,j}$. The rest of the computation is the same as standard PCA except that all the arithmetic is complex. In the end, we get complex normal coordinates $\tilde{N}_{k,j}$ and displacements $\tilde{R}_{k,i}$.

We have used this method to discover normal coordinates for the character shown in figure 7. We began with a set of volumetric finite element simulations using the method of Irving et. al [2004], and for each simulation, extracted the complex normal coordinate with the largest eigenvalue. We designed each simulation to excite a particular kind of oscillation meaningful to animators, so the deformations controlled by the LSC splines would make sense to them. In the accompanying video, we show the results from using the two coordinates $N_{1,j}$ which corresponds to an up-down shake, and $N_{2,j}$ which corresponds to a side-to-side oscillation.

It is possible to use only the real parts of $\tilde{N}_{1,j}$ and $\tilde{N}_{2,j}$ and drive them directly with the real value of the LSC spline. As shown in the video, however, this results in very stiff-looking motion typical of low-end computer graphics because all the points move in perfect synchrony. To break up the stiffness, we need to introduce what animators refer to as overlap. Different parts of the character need to begin their motions at different times. The phase components of the normal coordinates provide just the right information to make this happen. For a single coordinate, the proper deformation using the phase information can be written as

$$D(j) = \Re(x\tilde{N}_{1,j}). \tag{29}$$

The video shows the results of using this method for both normal coordinates $\tilde{N}_{1,j}$ and $\tilde{N}_{2,j}$. The resulting motion is much more organic, losing the former harsh CG stiffness. Figure 7 shows a false color map of the phase for the vertical component of the up-down motion. The phase varies smoothly over the character, with the motion of the arms and hands being delayed significantly compared to the body. In effect, the amplitude of $\tilde{N}_{k,j}$ for each j lets us know how much the jth point moves when the character oscillates in the kth coordinate, and the phase of $\tilde{N}_{k,j}$ lets us know whether the jth point moves before or after the main oscillation and by how much.

The final example in the video shows a combination of oscillations along the two coordinates. We could have used different LSC splines for each coordinate, but that would risk creating oscillations

at nearby frequencies that would beat against each other in disturbing ways. Instead, we chose to use amplitude and phase shifts of a single LSC spline for the full deformation:

$$D(j) = \Re(A_1 e^{i\phi_1} x\tilde{N}_{1,j} + A_2 e^{i\phi_2} x\tilde{N}_{2,j}) \quad (30)$$

where A_1 and A_2 represent the relative amplitudes, and ϕ_1 and ϕ_2 the relative phases of the two oscillations. The example in the video was constructed using just nine knots, yet it shows surprising subtlety and complexity.

8 Discussion and Conclusions

Our purpose in limiting ourselves to Spacetime Constraints problems with linear solutions is not to suggest that they suffice for achieving the goal of producing physically-realistic yet controllable motion. On the contrary, nonlinear mechanisms will certainly play an important role. Nonetheless, nonlinear optimizations are frought with peril from the point of view of real-world use by animators. For the most part, animators expect that continuous changes to their inputs will produce similarly continuous changes in their results. This is nearly impossible to ensure with nonlinear optimizations, but almost guaranteed by linear systems. As a consequence, LSC splines may face many fewer barriers to adoption than alternatives, particularly given that they completely generalize familiar piecewise cubics.

We do not claim to have settled on the best way to control the LSC splines, but we view their full interactive speed as key to their value. Animators demand true real-time control, and only by putting a technology like this in their hands can we get the feedback necesary to refine the Spacetime Constraints solutions into a powerful and effective tool.

The application of digital signal processing mathematics to this domain appears to be very powerful and enlightening. First of all, it allows us to express stability criteria and describe the resonances of linear Spacetime Constraints problems with ease. Second, the idea of using complex-valued splines would not have occurred to us outside this context. Our experience using both hand-crafted procedural models and models derived from simulation data have convinced us of the value of complex splines. Using the LSC splines and the complex principal components analysis together, we are able to create a method that distills simulation data into a simplified model and then allows us to generalize it by controlling its frequency, damping and phase relationships to other motions. This makes it possible to "composite" pieces of simulations together in a way that has not before been possible.

Acknowledgements

Thanks to [] for creating some of the initial simulation data we used for analysis.

References

[Anderson 1983] John R. Anderson and Richard D. Rosen, "The Latitude-Height Structure of 40-50 Day Variations in Atmospheric Angular Momentum," Journal of the Atmospheric Sciences, 40 (6) 1983 pp. 1584–1591.

[Bartels 1987] Richard Bartels, John Beatty and Brian Barsky,*An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*, Morgan Kaufmann, 1987.

[Cohen 1992] Michael Cohen, "Interactive spacetime control for animation," SIGGRAPH 1992, pp.293–302.

[Fang 2003] Anthony Fang and Nancy Pollard, "Efficient Synthesis of Physically Valid Human Motion," SIGGRAPH 2003, pp. 417–426.

[Godunov 1987] S. Godunov and V. Ryabenkii, *Difference Schemes: An Introduction to the Underlying Theory*, Elsevier, 1987

[Golub 1983] Gene Golub and Charles Van Loan, *Matrix Computations*, Johns Hopkins University Press, 1983.

[Goldstein 1980] Herbert Goldstein, *Classical Mechanics, Second Edition*, Addison Wesley, 1980

[Gleicher 1997] Michael Gleicher, "Motion Editing with Spacetime Constraint" in *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, pp 139-148.

[Irving 2004] J. Irving, J. Teran and R. Fedkiw, "Invertible Finite Elements For Robust Simulation of Large Deformation," Eurographics/ACM SIGGRAPH Symposium on Computer Animation (2004), pp.131-140.

[Lasseter 1987] John Lasseter, "Principles of Traditional Animation Applied to 3D Computer Animation," SIGGRAPH 1987, pp. 35–44.

[Liu 1994] Z. Liu, S. Gortler, and Michael Cohen, "Hierarchical Spacetime Control," SIGGRAPH 1994, pp. 35–42.

[McMahon 1984] Thomas McMahon, '*Muscles, Reflexes, and Locomotion*, Princeton University Press, 1984.

[Ngo 1993] Tom Ngo and Joe Marks, "Spacetime Constraints Revisited," SIGGRAPH 1993, pp. 343–350

[Oppenheim 1975] Alan Oppenheim and Ronald Schafer, '*Digital Signal Processing*, Prentice Hall, 1975.

[Popović 1999] Zoran Popović and Andrew Witkin, "Physically Based Motion Transformation," SIGGRAPH 1999, pp. 11-20.

[Popović 2000] Jovan Popović, Steven Seitz, Michael Erdmann, Zoran Popovic and Andrew Witkin, "Interactive Manipulation of Rigid Body Simulations," SIGGRAPH 2000, pp. 209-218.

[Press 1986] William Press, Brian Flannery, Saul Teukolsky and William Vetterling, *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Press, 1986.

[Rasmusson 1981] E. M. Rasmusson, P. A. Arkin, W. Y. Chen and J. B. Jalickee, "Biennial Variations in Surface Temperature over the United States as Revealed by Singular Decomposition" Mon. Wea. Rev. **109**, 1981, pp. 587-598.

[Rose 1996] Charles Rose , Brian Guenter , Bobby Bodenheimer and Michael F. Cohen, "Efficient generation of motion transitions using spacetime constraints," SIGGRAPH 1996, pp.147–154.

[Treuille 2003] Adrien Treuille, Antoine McNamara, Zoran Popović and Jos Stam, "Keyframe Control of Smoke Simulations," SIGGRAPH 2003, pp. 716–723.

[Safonova 2004] Alla Safonova, Jessica Hodgins and Nancy Pollard, "Synthesizing Physically Realistic Human Motion in Low-Dimensional, Behavior-Specific Spaces", SIGGRAPH 2004 pp. 514–521.

[Witkin 1988] Andrew Witkin and Michael Kass, “Spacetime Constraints,” SIGGRAPH 1988, pp. 159–168.

[Witkin 1995] Andrew Witkin and Zoran Popovic, “Motion Warping,” SIGGRAPH 1995 pp. 105–108.