Animating Oscillatory Motion With Overlap: Wiggly Splines

Michael Kass John Anderson
Pixar Animation Studios

Abstract

Oscillatory motion is ubiquitous in computer graphics, yet existing animation techniques are ill-suited to its authoring. We introduce a new type of spline for this purpose, known as a "Wiggly Spline." The spline generalizes traditional piecewise cubics when its resonance and damping are set to zero, but creates oscillatory animation when its resonance and damping are changed. The spline provides a combination of direct manipulation and physical realism. To create overlapped and propagating motion, we generate phase shifts of the Wiggly Spline, and use these to control appropriate degrees of freedom in a model. The phase shifts can be created directly by procedural techniques or through a paint-like interface. A further option is to derive the phase shifts statistically by analyzing a time-series of a simulation. In this case, the Wiggly Spline makes it possible to canonicalize a simulation, generalize it by providing frequency and damping controls and control it through direct manipulation.

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

Oscillatory motion is very common in computer graphics, occurring in virtually every computer animated feature film. Its ubiquity arises from the fact that most physical systems tend to oscillate when displaced from equilibrium. Typical examples of importance in film production include the motion of insect antennae, car suspensions and the swinging of ropes or animal tails. Unfortunately, traditional animation tools are poorly suited to these types of motions, adversely impacting both the artistic quality and the cost of animating oscillatory phenomena.

For decades, computer graphics scientists worked hard to get oscillations out of splines, and rightly so. Ordinary interpolants in use in other fields were prone to undesired oscillations and were deemed unacceptable in computer graphics. As a result, a variety of spline types were developed to combat oscillations and still provide adequate control to animators. While these spline types are well-suited to many types of motion, their use is extremely labor-intensive for motion which is fundamentally oscillatory. Large numbers of knots are required to create oscillations, and moving them appropriately in concert to craft oscillatory motion can be both awkward and time-consuming.

In recent years, physical simulation has been used with some success to animate oscillatory phenomena, but it too suffers from severe limitations. Chief among them is that the relationship between the parameters of a simulation and the actual behavior can be very complicated. As a result, technical specialists are often needed, and even they may be unable to come up with parameters that give a director the desired effect.

It may seem that one could set up the parameters for a physical simulation in a production environment when building characters and environments, and then use passive motion that simply reflected physical reality. However, even simple cases of secondary oscillatory animation like the jiggling of a character's belly, which we explore in section 6.2 and the associated video, involve non-trivial issues of story-telling, timing, cinematography and character expression. The way a belly moves informs the viewer of the character's moment by moment energy, strength and tension. Beyond that, anything that moves in film is liable to draw the viewer's attention. So animators and directors demand absolute control over what is moving, how it moves and for exactly how long. If a character's belly appears momentarily in profile, its jiggle may be excessively evident and need to be toned down. If an emotional moment demands stillness, the jiggle must come to a stop. All this is difficult with traditional simulation.Yet simulation still has much to offer. What we seek is a method that combines the physical reality of simulation with the directability and control of traditional splines.

Our solution begins with "Wiggly Splines," a generalization of the traditional animation splines that allow *intentional* oscillations. Wiggly Splines re-create traditional spline behavior when their resonant frequency and damping are set to zero, but create oscillatory animation when their resonance and damping are set to be representative of a an oscillatory physical system. The Splines provide a familiar interactive interface supporting direct manipulation while at the same time embedding the physical realism of an oscillatory differential equation into the spline itself. They provide the mix we seek between realism and control.

With Wiggly Splines, we can address not only individual degrees of freedom in a model that oscillate, but also the more broadly important problem of oscillatory motion that involves propagating phenomena like waves. With a phenomenon like belly jiggle, the nature of this propagation will determine whether the belly looks stiff, "sloshy" or somewhere in-between.

In particular, Wiggly Splines allow us to address a key issue known to animators by the term "overlap." The term generally refers to the fact that different degrees of freedom of any given model need to accelerate and decelerate at different times. If all the degrees of freedom vary synchronously in phase, the result is motion that usually looks very stiff or "computery." Breaking up the synchrony to the point that propagation effects are visible is vital to achieving an acceptably natural appearance. While this notion is central to animators' thinking, to the best of our knowledge, it has not until now received a mathematical characterization.

In order to address the issue of overlap, we argue that at least in the realm of oscillatory motion, and perhaps more broadly than that, overlap can be mathematically characterized through the notion of

ACM Reference Format
Kass, M., Anderson, J. 2008. Animating Oscillatory Motion With Overlap: Wiggly Splines. *ACM Trans. Graph. 27*, 3, Article 28 (August 2008), 8 pages. DOI = 10.1145/1360612.1360627 http://doi.acm.org/10.1145/1360612.1360627.

phase relationships of coupled oscillators. For ordinary splines, these phase relationships are not easily available. However, with Wiggly Splines, if one solves the optimization over complex numbers rather than real ones, there is a simple, automatic and natural assignment of phase angle to each point on the spline.

The availability of phase angle with Wiggly Splines opens up some interesting possibilities. For example, a single animation curve can control multiple degrees of freedom in a model related by phase or amplitude shifts. These phase and amplitude relationships can be authored procedurally or specified directly using a paint or similar interface. In addition, we show that the phase and amplitude relationships can be extracted from simulation data using a statistical time-series analysis. The time-series method is powerful because it allows us to take a single simulation, extract a canonical form, and then generalize and control it with the Wiggly Spline.

The rest of the paper is structured as follows. In section 2, we discuss previous work and introduce the basic approach. In section 3, we derive the basic Wiggly Spline by applying spacetime constraints to a damped mass-spring oscillator and show that it can be computed efficiently. 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 extend our analysis to complex-valued solutions. The complex solution allows us not only to broaden the resonance of the Wiggly Spline, but also to make explicit its phase information. Finally, in section 6 we show how the phase information can be used to create overlapped motion for procedural models, hand-weighted models, and models derived from simulation data.

2 Previous Work

There have been a variety of animation systems over the years that have used filtering to introduce oscillatory effects in animation. Notably [Litwinowicz 1991] and most recently [Wang 2006] describe animation systems with filters that produce some degree of "anticipation" and "follow-through" for a given set of key-framed values. [Unuma et al. 1995]and [Bruderlin and Williams 1995] have also proposed signal processing based approaches to processing animation. While these systems are suggestive of the value of filtering, they fail to provide interpolation of keys. In these systems, altering filter parameters usually changes the values of the animation curves at all the key frames. In the case of production animation for film, this is largely unacceptable. Animators, having selected values at key frames, are loathe to see their carefully crafted poses modified by filters. Filtering effects are valuable, but we seek a technique that allows these kinds of effects to be introduced while maintaining the direct manipulation and interpolation of traditional animation splines.

When oscillatory motion involves moderate deformations, it is well known that good approximations to physical simulation can often be achieved by combining the effects of a small number of oscillatory modes. Modal dynamics was introduced to the graphics community by [Pentland and Williams 1989] who describe a system based on deformable superquadrics. James and Pai [James 2002] showed how to extend the method to general finite-element models and compute the result in real-time with graphics hardware. While these efforts address the type of motion we consider, for our purposes they suffer from two key limitations. First and foremost, like all traditional forward simulation methods, they fail to provide the kind of control we require. As presented, there is no way for an animator to craft a series of poses at different times with these modal techniques and know that the dynamics will accurately recreate them. In addition, these methods provide only the original simulation parameters as methods for changing the motion. By contrast, we seek to abstract the full complexity of the simulation parameters into a much smaller number of simple controls that make sense to an animator.

In 1988, [Witkin and Kass 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 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. This formalism makes it possible to combine interpolation constraints with modal dynamics, and acts as the foundation of our approach.

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 and Marks 1993], [Liu et al. 1994], [Witkin and Popović 1995], [Rose et al. 1996], [Gleicher 1997], [Popović and Witkin 1999], [Popović et al. 1999], [Fang and Pollard 2003], [Treuille et al. 2003], [Safonova et al. 2004]). Good summaries of the specific contributions and evolution of the ideas can be found in [Fang and Pollard 2003] and [Safonova et al. 2004]. It is also possible to use approaches specialized for particular classes of objects. [Barzel 1997] has shown the utility of simple quasi-dynamic analogs for ropes and springs.

Our interest in oscillatory phenomena and overlapped motion leads us to a unique twist on the traditional spacetime formulation. The relationships of coupled oscillators are often best expressed by their relative phase, and phase is most conveniently encoded by complex numbers. As a result, we investigate spacetime optimizations for the first time over the realm of complex-valued functions. We find that this significantly extends the suitability of spacetime methods for modal dynamics. Since our spacetime solutions have associated phase information, they allow us to control propagation effects simply by scaling phase differences.

3 Spacetime Mass and Spring

The widespread applicability of modal dynamics in engineering arises from the fact that the equations of motion for physical systems displaced slightly from equilibrium become particularly simple when viewed in special modal coordinates. In these coordinates, under the common assumption of proportional (Rayleigh) damping, the small displacement equations of motion are actually the same as those for a single mass and spring [Goldstein 1980]. If we can control the motion of a single mass and spring, then using modal coordinates, we can control very complex motion.

To apply the Spacetime Constraints formalism, we take our physical system with a single mass and spring, add constraints provided by the animator, and subject to these constraints, minimize an objective function which penalizes non-physical and inefficient motion. A physically accurate simulation will follow Newton's second law of motion: $f = ma$. Instead of integrating this differential equation, however, we follow [Witkin 1988] and minimize the L^2 norm of the deviation from correct physics.

$$E = \int (f - ma)^2 dt \tag{1}$$

Let our point mass move in one dimension with position given by

$x(t)$. In a standard damped spring, the applied force is given by:

$$f = -kx - \gamma\dot{x} + f_{ext} \tag{2}$$

where f_{ext} represents a specified external driving force applied to the spring. Plugging this expression for the force into equation 1 yields the optimization function

$$E = \int (\ddot{x} + \gamma\dot{x} + kx - f_{ext})^2 dt. \tag{3}$$

where we have chosen units in which the mass of the particle is one.

Consider what happens when the spring constant k and the damping γ are zero. Then without external forcing, the function to be minimized reduces to:

$$E = \int \ddot{x}^2 dt. \tag{4}$$

Equation 4 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 4 gives rise to piecewise cubic polynomial solutions which are widely used throughout computer graphics.

Calculus of variations can similarly be used to construct an analytic minimum for equation 3, but we find a finite difference approach to more convenient for our purposes. Discretizing with standard finite differences leads to the following function to be minimized:

$$E_s = \frac{1}{h^2} \sum_{i=1}^{n} \Big((x_{i-1} - 2x_i + x_{i+1}) + h\gamma(x_{i+1} - x_i) + h^2 x_i - f_{ext} \Big)^2 \tag{5}$$

where h is the time separation between samples. The objective function is a positive-definite quadratic form, so the minimum will occur when the gradient vanishes

$$\frac{\partial E_s}{\partial x_i} = 0 \tag{6}$$

For each i away from the boundaries, three terms of the sum in equation 5 lead to non-zero derivatives, and they combine so that equation 6 leads to a banded linear system with bandwidth 5. Equations of this form can be solved using standard techniques [Golub and Van Loan 1983] [Press et al. 1986] in time proportional to n, the number of samples of x_i.

In order to make the solution of equation 6 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 5 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. We can make the tangent constraint absolute by using the equation $x_{i+1} - x_i = g$ to eliminate a variable from the linear system. Alternatively, we can introduce an approximating tangent constraint by adding a penalty term to the optimization function as follows:

$$E = E_s + E_t \tag{7}$$

$$E_t = c_t((1/h)(x_{i+1} - x_i) - g)^2 \tag{8}$$

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

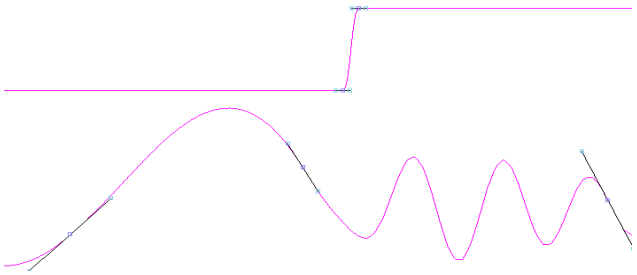

Figure 1: Top curve: Resonant frequency. The resonance begins at zero, and then changes abruptly to a non-zero value. Bottom curve: Wiggly Spline with three interpolation and tangent constraints.

Note that with just interpolation constraints, the solution of the minimization problem in equation 5 produces a spline with global support. If local support is important, strong tangent constraints will decouple the solution on one side from the solution on the other.

We have embedded our solution to equation 5 into a traditional spline editor. Our implementation achieves frame rates limited by drawing speed rather than the computation time of the Wiggly Spline itself. Figure 1 shows an example of a Wiggly Spline being used. The upper curve is a traditional spline used to control the spring constant, and the lower curve is the Wiggly Spline itself. 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.

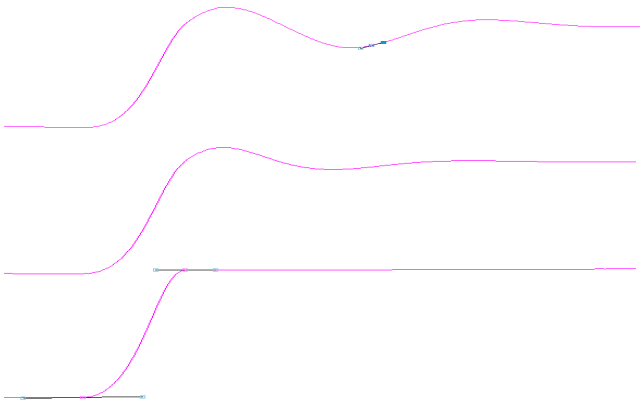

Figure 2: Bottom: Changing equilibrium value. Middle: Wiggly Spline with no constraints, showing passive response. Top: Wiggly Spline edited with an interpolation constraint.

Figure 2 illustrates the difference between our spacetime approach and traditional forward simulation. In this example, we have animated the equilibrium position of our spring to follow the curve at the bottom of figure 2. As with the method of [James and Pai 2002], the acceleration of the spring equilibrium causes a external force f_{ext} to be applied to the spring, and the middle curve of figure 2 shows the passive dynamics that results. This, however, is where the similarity to forward simulation ends. In forward simulation, the only way to modify the curve is by changing the simulation parameters, but the relationship between the simulation parameters

and the resulting curve can be very complex. Moreover, there is no guarantee that *any* settings of the parameters will yield the desired motion. Animators faced with such circumstances will frequently give up and accept an artistic compromise. With our system, by contrast, the animator can edit motion curves with direct manipulation. The top curve of figure 2 shows this in action. An animator has introduced a knot and tangent constraint to increase the amount of overshoot, and the Wiggly Spline faithfully interpolates the animator's new knot. If the animator wants absolute control of a piece of the curve, he or she can set the resonant frequency and damping of the Wiggly Spline to zero on the corresponding interval and edit that segment like any other traditional piecewise cubic. With Wiggly Splines, the animator gets the default behavior of forward simulation, but can edit the output with the absolute control and directability of traditional splines.

4 Digital Signal Processing View

The continuous differential equation that results from equation 2 is stable for all positive values of k and γ. Unfortunately, once finite differences are substituted for continuous derivatives, stability is no longer guaranteed. Frequencies higher than those representable at the given sampling rate can cause aliasing and make the equations diverge. Here we make use of results from digital signal processing to derive conditions that guarantee stability.

Discussion of the stability of spacetime constraint optimization is generally absent in the literature, perhaps because it seems hopeless to make any guarantees of convergence in the general non-linear case. Nonetheless, non-linear spacetime problems are usually solved by a sequence of linearized sub-problems, and the techniques we use here can provide stability conditions for the linearized sub-problems.

Let $J = f - ma$ represent the departure of the spacetime solution from accurate physics in the spacetime solution. J has units of force and can be regarded as the force that has to be applied to our particle to make the equations of physics hold.

Substituting finite differences for the derivatives in equation 2 yields

$$f_i = f_{ext} - kx_i - \gamma(x_{i+1} - x_i)/h. \tag{9}$$

Combining this with the definition of J and using a finite difference for $\ddot{x}$ gives us

$$x_{i+1} - ax_i - bx_{i-1} = J_i + f_{ext} \tag{10}$$

where a and b are functions of the spring constant k and the damping constant γ. In the language of Digital Signal Processing, equation 10 represents a second order recursive (IIR) filter where the input of the filter is the sum of the external force and the residual 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

$$E_s = \sum_{i=1}^{n} (x_{i+1} - ax_i - bx_{i-1})^2. \tag{11}$$

The response of the filter in equation 10 to the input is fully characterized by the system function of the filter, which is given by [Oppenheim 1975]

$$H(z) = \frac{1}{1 - az^{-1} - bz^{-2}}. \tag{12}$$

where z is a complex parameter. When evaluated on the unit circle, the system function gives the frequency response of the filter. If

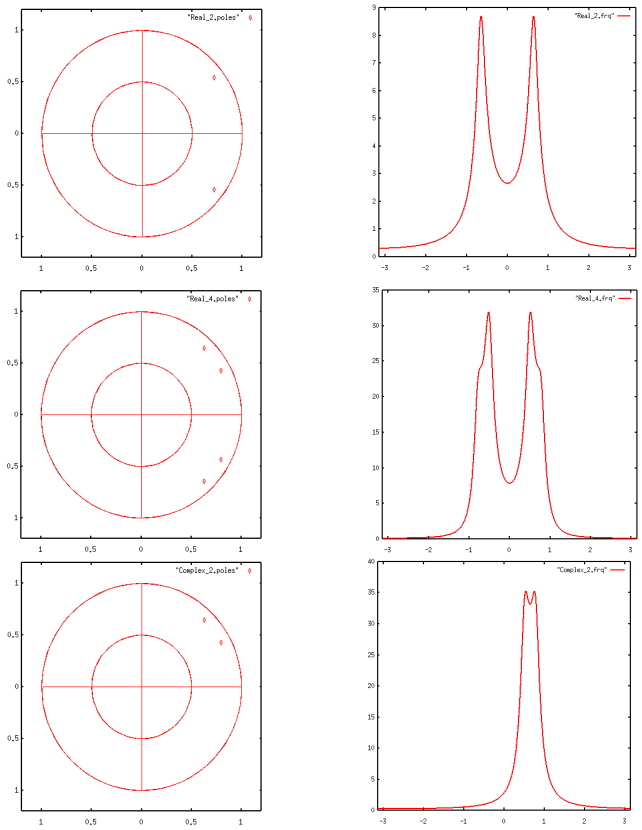

Figure 3: Left: pole locations. Right: frequency response. Top: Two pole real filter. Middle: Four pole real filter. Bottom: Two pole complex filter

we factor the system function, then we will find two roots of the denominator polynomial

$$H(z) = \frac{1}{(1 - \lambda_0 z^{-1})(1 - \lambda_1 z^{-1})} \tag{13}$$

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.

We find that a fixed angular separation of the poles works for our purposes. If a user has selected a center frequency of ω and a damping rate $d > 1$, we use poles

$$\lambda_0 = e^{-d + i(\omega + .05)} \tag{14}$$

$$\lambda_1 = e^{-d + i(\omega - .05)} \tag{15}$$

and compute a and b using

$$a = \lambda_0 + \lambda_1 \tag{16}$$

$$b = -\lambda_0 \lambda_1. \tag{17}$$

For a general non-linear spacetime problem, the analysis we provide above can be used for linearized sub-problems. In that case, step size control can be used at each iteration to bring all the poles within the unit circle.

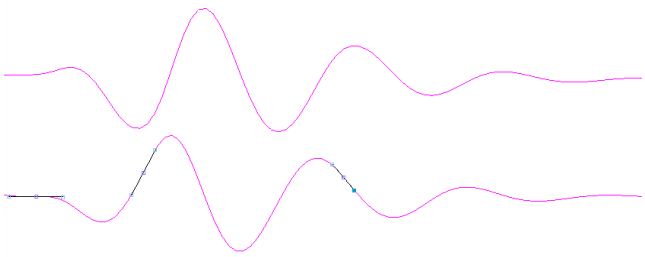

Figure 4: Complex-valued spline: The bottom curve shows the real part resulting from three interpolation and tangent constraints. The top curve is the computed imaginary part.

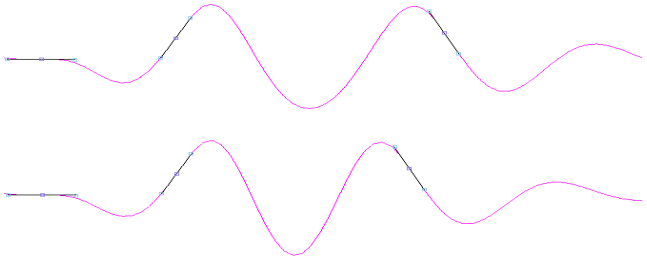

Figure 5: The complex spline is able to adjust its frequency as the interpolation constraints are moved.

5 Complex Valued Spline

The Wiggly Spline as described above has one remaining difficulty as an authoring tool. Its resonance is too sharp. The spline is overeager to oscillate at precisely the resonant frequency, and this can make it difficult to control.

What we expect when we pull two knots apart is that the spline curve in between will stretch up to some limit, and then when it passes the limit begin to introduce an additional cycle. With an excessively sharp resonance, this will not happen. Instead, the curve will continue to oscillate at exactly its resonant frequency, and it will satisfy the constraints by adding whatever low-frequency components are necessary. The behavior is extremely counter-intuitive for an animator. In order to fix this behavior, we need to widen the resonance, creating a relatively flat passband for the filter.

The second order mass-spring system provides very 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 3 shows a representative sample pair of conjugate pole locations, and the corresponding of the frequency response. Note the sharpness of the resonance. 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 3 shows a representative example of what can be accomplished with this technique. It depicts 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 can be obtained. The bottom row of figure 3 shows a representative example of this technique. The pass band has been widened while keeping the response at zero frequency much lower.

Figure 3 is indicative of a general issue. With real-valued signals and real-valued filters, it is not possible to distinguish between positive frequencies and negative ones, so the frequency response of a filter has to be symmetric around zero. Going to a complex representation makes this issue go away. More generally, with real-valued signals, there is a fundamental ambiguity of phase. The phase of a real-valued oscillatory signal exists only implicitly. Moving to complex-valued signals, by contrast, makes the phase both explicit and unambiguous. While the idea of having a complex-valued animation curve may seem odd at first, the advantages of an explicit phase representation have led us to embrace it.

Extending the Wiggly Spline to the complex domain turns out to be relatively simple. We begin as before by having the animator select the resonant frequency and damping. These 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 selects the bandwidth of the filter. In our experience, a rotation of about .05 radians in each direction has worked well. From λ_0 and λ_1, we can then 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

$$E_s = \sum_{i=1}^{n} |x_{i+1} - ax_i - bx_{i-1}|^2 . \tag{18}$$

where the sum of squares in equation 11 has been replaced with a sum of squared complex magnitudes. The number of degrees of freedom in the optimization has also 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 Wiggly Spline solution. All the interpolation constraints are applied to the real part, and the imaginary part is computed almost as a side effect. The animator can edit the real part, blissfully unaware that the imaginary part even exists. The presence of the imaginary part, however, makes the complex-valued Wiggly Spline a good deal more powerful than its real-valued counterpart.

Figure 5 shows that the widening of the resonance with the complex Wiggly 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 Animating with Wiggly Splines

6.1 Controlling Procedural Models

We introduced the complex-valued Wiggly Spline in part to craft a desired frequency response, but also to make the phase explicit. Here we show that the explicit phase information can help us create overlapped motion. Consider the value x of the complex Wiggly Spline at a given moment in time. We can represent it in polar imaginary coordinates using the identity

$$x = re^{i\theta} \tag{19}$$

As long as r does not vanish, the value of θ is well defined and easy to calculate from x. In all cases, whether r vanishes or not,

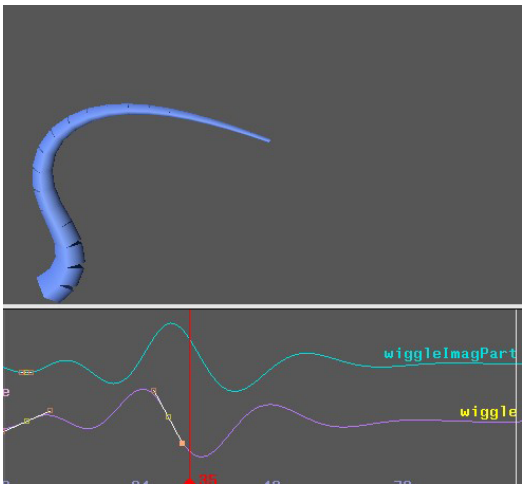

Figure 6: Animation of a procedural tail

it is easy to create phase-shifted animation curves from x. Since phase angles add under multiplication, all we have to do is multiply x by a unit-magnitude complex number with the appropriate phase to accomplish the shift. Let ϕ be the desired phase shift. We can compute a real-valued phase-shifted animation curve x_ϕ using the formula

$$x_\phi = \Re(re^{i(\phi+\theta)}) = \Re(e^{i\phi}x) \tag{20}$$

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

Figure 6 shows a procedural model of a tail being animated with this computation. The real part of the original Wiggly 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. The phase shifts increase along the length of the tail at a rate proportional to the width. The amplitudes are proportional to the width. Even with just a few animation knots, it is possible to get very interesting and lively animation out of models like this. The accompanying movie shows an example of an animation curve and the resulting real-time motion of the tail. The changing phase shifts along the length of the curve give rise to propagating and overlapped motion.

6.2 Hand Crafting Modes

In section 3, we noted that if we use modal coordinates, the motion of complex systems near equilibrium can be described accurately by the preferred dynamics of a Wiggly Spline. There remains the question of how to create these coordinates and connect them to a Wiggly Spline. For maximum artistic control, there will be times when these oscillatory coordinates are best authored by hand.

Figure 7 illustrates an example where we have done just that to add belly jiggle to an animated character. For this example we hand specified amplitude $A(p)$ and phase $\phi(p)$ fields on the surface points of the character. The amplitude field has been constructed to have a zero value on the perimeter of the belly where the motion blends into the rigid character and increases to a maximum in the center of the soft region. The phase field is likewise constructed to be zero on the perimeter and smoothly increases to a maximum value in the softest part of the character. There are larger phase angles in the lower regions of the belly to represent their weaker connection to the frame.

Together, these two fields provide a complex number at each point on the belly that control the relationship of the belly-jiggle deformation to the driving wiggly spline. The amount d of belly-jiggle applied to each point p is then given by:

$$d(p) = \Re(rA'e^{i(\phi'+\theta)}) \tag{21}$$

where A' and ϕ' incorporate the effects of a scaling, s, of the imaginary component:

$$A' = |A\cos(\phi) + isA\sin(\phi)| \tag{22}$$

$$\phi' = \arg(A\cos(\phi) + isA\sin(\phi)) \tag{23}$$

In the accompanying video, we show examples of the animation with the phase scaling s of the belly jiggle set to 0, 1.0 and 2.3. At a value of zero, the belly jiggle has no true propagation effects, and looks somewhat stiff. As the scaling is increased to one, the motion looks increasingly fluid, due to the overlap. At a value of 2.3, the belly appears to be sloshing around. This scaling of the imaginary (out of phase) component provides an effective tool for controlling the apparent rigidity of part of a model.

It may seem that similar effects can be accomplished through time delays. They can, however, the delays would have to be changing with time. In this example, the time between foot plants varies by about 50% over the clip due to changes in both the character's speed and the length of his steps. The constant phase delay we used provides a consistent impression of the firmness of the character's belly. A constant time delay would not. Achieving a similar-quality result with time delays would require animating the delays with care to keep the belly jiggle synchronized to the footfalls.

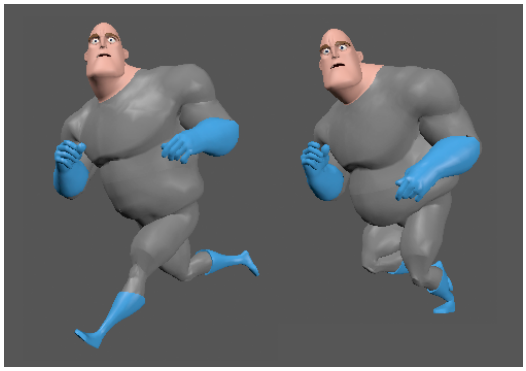

Figure 7: Two frames from a shot with Wiggly Spline belly jiggle.

6.3 Deriving Modes from Simulations

While hand-authored modal coordinates provide the greatest possible control, in some cases simulation can provide a more realistic result. [James and Pai 2002] for example illustrate modes that can be extracted directly as eigenvectors of the stiffness matrix of a finite-element simulation. These are certainly candidates for use with Wiggly Splines, although they suffer from two important limitations. The first is that the eigenvectors of the stiffness matrix are real-valued. A real-valued mode by itself cannot describe propagating motion – it will appear to oscillate in place, and look very stiff.

When finite-element models give rise to propagating phenomena, the modes usually come in orthogonal out-of-phase pairs. One will correspond to the real part and the other to the imaginary part of the complex-valued mode we seek. It is the complex modes that are capable of describing propagating phenomena. In principle, the real and imaginary parts of the complex eigenvector will share an eigenvalue, so it should be possible to join them automatically, but

numerical issues may make it difficult to bring these pairs of real modes together.

A second and more fundamental limitation of extracting modes directly from the stiffness matrix is that they are based on small-deflection approximations. In many cases we are concerned with large enough deformations that the small-deflection approximation is not very good. Instead, a much better result is obtained by creating the best linear approximation to the behavior of the system under large deflections. For these purposes, we propose a different technique.

To approximate the linear behavior of a non-linear finite-element simulation under large displacements, we suggest forcing the system into large displacements of interest and then look for statistical linear relationships in the resulting motion. From this we can extract appropriate modes. We have done this with a variety of CG character models including the character in figure 8.

The most widely-used statistical analysis technique for extracting linear relationships from time-series data is Principle Component Analysis (PCA), but like the technique of computing modes directly from a stiffness matrix, it generates only real-valued eigenvectors. In order to capture complex modes corresponding to propagating motion, we choose a modified version of PCA that does all its arithmetic over the complex domain. Developed by climate researchers in the early 1980s [Rasmusson et al. 1981] [Anderson and Rosen 1983], it has become known as Complex Principal Components Analysis. An accessible review of these techniques appears in [Horel 1984].

The basic structure of the Complex PCA technique is to take the real data $S_{i,j}$ and extend it to a complex representation by using a finite impulse response Hilbert transformer to add an imaginary part to the real-valued time series for each point. Details on the design of Hilbert transformers can be found in [Oppenheim and Schafer 1975].

A standard PCA analysis is then performed on the complex data set. The resulting complex eigenvectors represent both the amplitude and phase propagation of the motions in the original data set. To the best of our knowledge, this represents the first use of Complex PCA in the field of computer graphics.

Many of our ideas about the use of complex PCA to provide an editable representation of simulations derive from our experience attempting to simulate the character presented in this section for the short film "Lifted". We found that using an editable phase-amplitude representation of the key simulation behavior was a much more effective way of achieving the desired look and feel of the character than trying to find a "sweet spot" in the simulation parameter space.

In order to demonstrate the use of the Complex PCA technique we have simulated an animated character by impulsively forcing it in the vertical direction using the physical volumetric model described in [Irving 2004]. The first complex mode of this simulation is shown in figure 8.

The animation of the character shown in the accompanying video is constructed following equation 21 where the complex mode appears as $A(p)$ and $\phi(p)$ and the excitation factors r and θ are given by the product of the Wiggly Spline and a complex coupling constant representing the relative phase and amplitude relationship between the wiggly spline and the mode excitation.

When only the real part of the Wiggly Spline is used to drive the mode, corresponding to $s = 0$, the motion is stiff and uninteresting, resembling the motion of a nutcracker. This is the result of a complete lack of overlap. When both the real and imaginary components of the Wiggly Spline are used, $s = 1$, the propagating behavior approximates the phase propagation of the training simulation, completely changing the impression of the motion. Instead of being rigid and "computery," it achieves a much more fluid appearance.

As with the previous example in section 6.2 it is also possible to exaggerate the fluidity of the motion by using scale factors: $s > 1$ which can yield sloshy effects that might be very difficult to simulate with physical models.

We performed an additional Complex SVD analysis of a second simulation, forced with side to side impulsive forcing. In the video these two modes have been combined, driven by the same Wiggly Spline but with two separate complex coupling coefficients to generate the hula dancing alien clip.

By extracting the complex mode from this character's simulation, we have characterized much of the feel of a reference simulation into a single vector. We can alter its frequency or its damping, and we can change the apparent stiffness of the material from which it is made by scaling the phase differences. We can vary all of these parameters continuously through time, and the Wiggly Spline will blend their effects together. With a collection of modes like this, we can essentially piece together generalized simulations like visual composites, mixing the realism of simulation with the expressive power of direct manipulation.

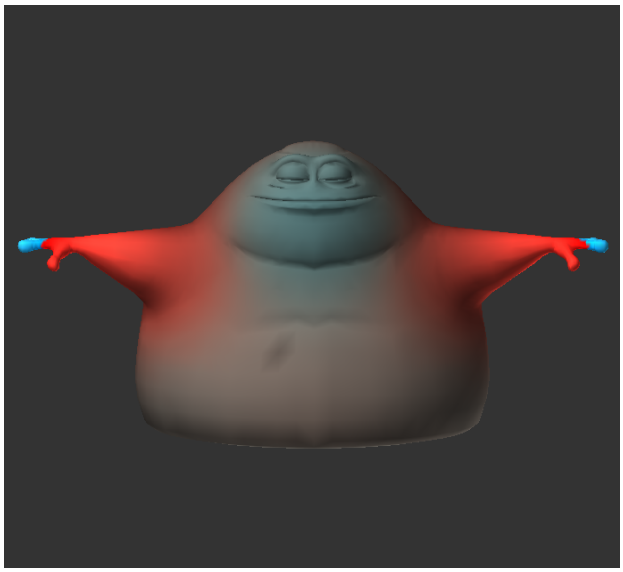

Figure 8: First complex (phase-amplitude) PCA mode of a simulation of a deformable character driven by up-down oscillations. For this figure the phase is encoded in the hue and the amplitude in the saturation.

7 Discussion and Conclusions

Animators have known for a long time that overlap is critical to creating pleasing animation. The term overlap, however, has never had a precise definition. In our examples, we show that the notion of overlap correlates very well with the degree of phase difference between the motion of different degrees of freedom in a model. We also show that Wiggly Splines provide direct control over these phase differences, allowing this animation application to "speak" the language of animators.

With their ability to control overlap, Wiggly Splines provide a practical animation method for oscillatory phenomena. They combine the strength of physical simulation with the high degree of control

present in traditional splines. We have used them successfully in a variety of tests and anticipate using them extensively in production. Yet we believe the value of this paper goes beyond Wiggly Splines themselves. In the course of developing the splines, we have introduced:

- The representation of animation curves with complex values.
- The characterization of "overlap" by phase differences.
- The use of DSP analysis to improve the stability of spacetime solution methods.
- The use of Complex PCA to extract propagating phenomena from graphics simulations.
- The approach of extracting a canonical motion from a simulation and then generalizing it by altering parameters such as frequency, damping and rate of propagation.

We hope these ideas will find applications beyond the scope of this paper.

Acknowledgements

We would like to thank Geoffrey Irving for his assistance with the Physbam simulations used to create the modes in section 6.3.

References

ANDERSON, J. R., AND ROSEN, R. D. 1983. The latitude-height structure of 40-50 day variations in atmospheric angular momentum. *Journal of the Atmospheric Sciences 40*, 6, 1584–1591.

BARTELS, R., BEATTY, J., AND BARSKY, B. 1987. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann.

BARZEL, R. 1997. Faking dynamics of ropes and springs. *IEEE Computer Graphics and Applications 17*, 3, 31–39.

BRUDERLIN, A., AND WILLIAMS, L. 1995. Motion signal processing. In *Proceedings of SIGGRAPH 2001*, 97–104.

COHEN, M. 1992. Interactive spacetime control for animation. In *Proceedings of SIGGRAPH 1992*, 293–302.

FANG, A., AND POLLARD, N. 2003. Efficient synthesis of physically valid human motion. In *Proceedings of SIGGRAPH 2003*, 417–426.

GLEICHER, M. 1997. Motion editing with spacetime constraints. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, 139–148.

GODUNOV, S., AND RYABENKII, V. 1987. *Difference Schemes: An Introduction to the Underlying Theory*. Elsevier.

GOLDSTEIN, H. 1980. *Classical Mechanics, Second Edition*. Addison Wesley.

GOLUB, G. H., AND VAN LOAN, C. F. 1983. *Matrix Computations*. Oxford Pressl.

HOREL, J. D. 1984. Complex principal component analysis: Theory and examples. *Journal of Climate and Applied Meteorology 23*, 12, 1660–1673.

IRVING, G., TERAN, J., AND FEDKIW, R. 2004. Invertible finite elements for robust simulation of large deformation. In *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, 131–140.

JAMES, D., AND PAI, D. 2002. Dyrt: Dynamic response textrues for real time deformation simulation with graphics hardware. In *Proceedings of SIGGRAPH 2002*, 582–585.

LASSETER, J. 1987. Principles of traditional animation applied to 3d computer animation. In *Proceedings of SIGGRAPH 1987*, 35–44.

LITWINOWICZ, P. 1991. Inkwell: A 2-d animation system. In *Proceedings of SIGGRAPH 1991*, 113–22.

LIU, Z., GORTLER, S., AND COHEN, M. 1994. Hierarchical spacetime control. In *Proceedings of SIGGRAPH 1994*, 35–42.

MCMAHON, T. 1984. *Muscles, Reflexes, and Locomotion*. Princeton University Press.

NGO, T., AND MARKS, J. 1993. Spacetime constraints revisited. In *Proceedings of SIGGRAPH 1993*, 343–350.

OPPENHEIM, A., AND SCHAFER, R. 1975. *Digital Signal Processing*. Prentice Hall.

PENTLAND, A., AND WILLIAMS, J. 1989. Good vibrations: model dynamics for graphics and animation. In *Proceedings of SIGGRAPH 1989*, 215–222.

POPOVIĆ, Z., AND WITKIN, A. 1999. Physically based motion transformation. In *Proceedings of SIGGRAPH 1999*, 11–20.

POPOVIĆ, J., SEITZ, S., ERDMANN, M., POPOVIĆ, Z., AND WITKIN, A. 1999. Physically based motion transformation. In *Proceedings of SIGGRAPH 1999*, 11–20.

PRESS, W., FLANNERY, B., TEUKOLSKY, S., AND VETTERLING, W. 1986. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press.

RASMUSSON, E., ARKIN, P., CHEN, W., AND JALICKEE, J. 1981. Biennial variations in surface temperature over the united states as revealed by singular decomposition. *Mon. Wea. Rev. 109*, 587–598.

ROSE, C., GUENTER, B., BODENHEIMER, B., AND COHEN, M. F. 1996. Efficient generation of motion transitions using spacetime constraints. In *Proceedings of SIGGRAPH 1996*, 147–154.

SAFONOVA, A., HODGINS, J., AND POLLARD, N. 2004. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. In *Proceedings of SIGGRAPH 2004*, 514–521.

TREUILLE, A., MCNAMARA, A., POPOVIĆ, Z., AND STAM, J. 2003. Keyframe control of smoke simulations. In *Proceedings of SIGGRAPH 2003*, 716–723.

UNUMA, M., ANJYO, K., AND TAKEUCHI, R. 1995. Fourier principles for emotion-based human figure animation. In *Proceedings of SIGGRAPH 1995*, 91–96.

WANG, J., STEVEN DRUCKER, M. A., AND COHEN, M. 2006. The cartoon animation filter. In *Proceedings of SIGGRAPH 2006*, 1169–1173.

WITKIN, A., AND KASS, M. 1988. Spacetime constraints. In *Proceedings of SIGGRAPH 1988*, 159–168.

WITKIN, A., AND POPOVIĆ, Z. 1995. Motion warping. In *Proceedings of SIGGRAPH 1995*, 105–108.