Interactive Spacetime Control of Deformable Objects

Klaus Hildebrandt Christian Schulz Christoph von Tycowicz Konrad Polthier
Freie Universität Berlin

Abstract

Creating motions of objects or characters that are physically plausible and follow an animator's intent is a key task in computer animation. The *spacetime constraints* paradigm is a valuable approach to this problem, but it suffers from high computational costs. Based on spacetime constraints, we propose a framework for controlling the motion of deformable objects that offers interactive response times. This is achieved by a model reduction of the underlying variational problem, which combines dimension reduction, multipoint linearization, and decoupling of ODEs. After a preprocess, the cost for creating or editing a motion is reduced to solving a number of one-dimensional spacetime problems, whose solutions are the *wiggly splines* introduced by Kass and Anderson [2008]. We achieve interactive response times through a new fast and robust numerical scheme for solving the one-dimensional problems that is based on a closed-form representation of the wiggly splines.

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;

Keywords: computer animation, spacetime constraints, physical-based animation, model reduction, control, wiggly splines

Links: DL PDF

1 Introduction

In traditional computer animation, the motions of objects or characters are typically generated from keyframes that specify values for all of the object's or character's degrees of freedom at a sparse set of points in time. A continuous motion is obtained by fitting splines through the keyframes. This technique is attractive since it offers an adequate amount of control over the motion at a low computational cost. One drawback for this technique is that it offers little help to an animator who wants to create physically plausible motions. Moreover, splines are designed to produce functions with high fairness (e.g. functions with few extrema and inflection points), whereas the motion of objects or characters is often oscillatory.

Physical simulation can produce realistic motions, but it is a delicate task to explicitly determine forces and physical quantities that produce a motion that matches an animator's intentions. This is aggravated by the fact that physical simulations are integrated forward in time, which means that small changes at some point in time can have a large impact on the state of the system at a later time. Control over a simulation can be achieved by computing optimal physical trajectories that are solutions of a variational spacetime problem. Such techniques calculate acting forces that minimize an objective functional while guaranteeing that the resulting motion satisfies prescribed spacetime constraints, *e.g.* interpolates a set of keyframes. Resulting forces are optimally distributed over the whole animation and show effects like squash-and-stretch, timing, or anticipation that are desired in animation. However, the computational cost for obtaining these results is that of solving a spacetime optimization problem. To date, recent methods, even those that use dimension reduction techniques, still require at least several minutes to solve the optimization problem for an interesting motion of an object or a character.

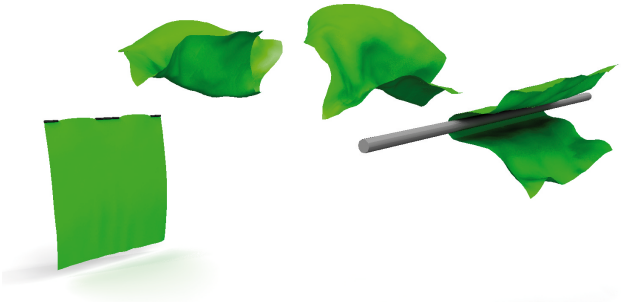

Figure 1: *A cloth animation that exhibits physical behavior but is controlled by keyframes is shown. (See also the attached video.)*

In this paper, we propose a technique for generating motions of deformable objects that can be controlled by spacetime constraints like keyframes, velocities, and forces. The main features of our scheme are that it achieves interactive response times and produces explicit time-continuous parametrizations of the motion. The scheme is based on a variational problem that determines optimal trajectories through the minimization of an acting force with respect to a spacetime L^2-norm. To lower the cost of the optimization, we combine different model reduction techniques and shift computational effort to a preprocess. First, we propose a simple yet effective way to construct a reduced space to which we restrict the variational problem. Secondly, we develop a multipoint linearization technique for the equations of motion. Using the fact that the desired trajectory interpolates the keyframes at specified points in time, we linearize the equations of motion around every keyframe and use each of the linear equations for a time interval around the corresponding point in time. Thirdly, we fully decouple the linearized spacetime constraint problems in the reduced space.

After decoupling the equations, only one-dimensional variational problems need to be solved. Such one-dimensional problems were considered by Kass and Anderson [2008]. Their structure is similar to the variational problem satisfied by cubic B-splines; it even includes B-splines as a special case. Thus the solutions can be seen as an extension of B-splines that in addition to smooth interpolants can also produce oscillating functions. Kass and Anderson named them *wiggly splines*. For computation they use a finite difference scheme.

ACM Reference Format
Hildebrandt, K., Schulz, C., von Tycowicz, C., Polthier, K. 2012. Interactive Spacetime Control of Deformable Objects. *ACM Trans. Graph. 31* 4, Article 71 (July 2012), 8 pages. DOI = 10.1145/2185520.2185567 http://doi.acm.org/10.1145/2185520.2185567.

At the heart of our technique is an efficient algorithm for the computation of wiggly splines based on a closed-form representation of the wiggly splines. We show that, as B-splines are composed of polynomials, wiggly splines can be assembled from functions that describe damped and driven oscillations. The explicit form of the functions depends on physical parameters. We distinguish six cases listed in the appendix. To compute the explicit representation of a wiggly spline, our algorithm solves a linear system with a band structure and of small size, which depends on the number of keyframes. As a result, the computation is robust (even for extreme parameter values) and produces $\sim 10\,k$ wiggly splines per second on a custom laptop (with an i7 quad-core 2.2 GHz CPU) even without using parallelization.

2 Related Work

Variational problems with spacetime constraints were introduced to computer animation by Witkin and Kass [1988]. Their example of a jump of the lamp *Luxo* nicely demonstrates the benefits of this approach. Since then the *spacetime constraints* paradigm has stimulated much research, see [Fang and Pollard 2003; Safonova et al. 2004] for a detailed summary. Based on spacetime constraints, techniques for controlling simulations of various types of physical systems including human motions [Gleicher 1997; Fang and Pollard 2003; Safonova et al. 2004; Chai and Hodgins 2007], ropes and strings [Barzel 1997], rigid body motions [Popović et al. 2003], fluids [Treuille et al. 2003; McNamara et al. 2004], particle systems [Wojtan et al. 2006], and elastic solids [Barbič et al. 2009] have been proposed. The resulting optimization problems are typically solved with gradient-based approaches or Newton methods. Local-to-global strategies, called *spacetime windowing*, were developed to speed up the solvers by Cohen [1992] and by Treuille et al. [2003]. A particular problem is the calculation of the derivatives of the objective functional. Automatic and symbolic differentiation has been used by many researchers, *e.g.* [Witkin and Kass 1988; Fang and Pollard 2003; Safonova et al. 2004], and spacetime constraints serve as one of the main target applications for the development of algorithms for automatic differentiation in graphics [Guenter 2007]. Explicit derivatives for different physical systems can be obtained through the adjoint method, see [McNamara et al. 2004; Wojtan et al. 2006; Barbič et al. 2009].

Model reduction is an established technique in solid mechanics that can be used to accelerate simulations of elastic solids, cf. [Nickell 1976; Idelsohn and Cardona 1985; Krysl et al. 2001]. In graphics Pentland and Williams [1989] pioneered work in this area by using modal analysis to automatically generate reduced spaces. Based on an efficient representation of the forces in the reduced space, Barbič and James [2005] obtained real-time rates for forward simulation of elastic solids. In addition, they extended the automatic generation of subspaces to include modal derivatives, which helps to improve the approximation of large deformations. Treuille et al. [2006] and Wicke et al. [2009] used reduced spaces constructed from eigenmodes for fluid dynamics. In geometry processing, reduced spaces were used for interactive modeling of triangular meshes by Huang et al. [2006] and Hildebrandt et al. [2011]. Kim and James [2009] used model reduction to speed up the calculation of large simulations for animation, by skipping full steps if the reduced step satisfies an accuracy condition. The reduced model is not built in a preprocess, but online as the simulation progresses. Reduced spaces have also been used for spacetime constraints. For human motions, Safonova et al. [2004] and Sulejmanpašić and Popović [2005] constructed reduced spaces from motion capture data. Barbič et al. [2009] constructed reduced spaces for deformable objects automatically from the keyframes by using vibration modes and tangents of a *deformation curve* that is fitted to the keyframes. This technique for constructing reduced spaces is similar to the way we construct reduced spaces. However, our method allows working with larger reduced spaces then theirs.

Related to spacetime constraints is also geometric interpolation between two or more shapes. The variational problem for geodesics in shape spaces has a comparable complexity and solvers need to deal with varying curves in a shape spaces, as well. Efficient multiresolution solvers for computing geodesics in shape spaces were proposed by Kilian et al. [2007] and by Wirth et al. [2009].

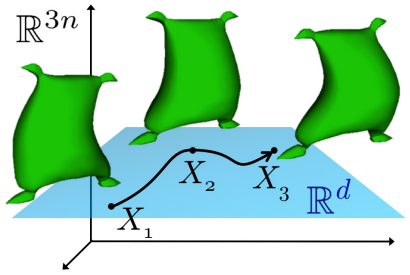

Figure 2: *An illustration of our scheme. The animator generates keyframes with a surface modeler. A reduced space (blue plane) is generated from the keyframes. Our method constructs a motion that interpolates the keyframes by solving a reduced spacetime optimization problem.*

3 Dynamics of Deformable Objects

There are different theories that model the behavior of elastic objects and various ways to discretize them. We keep the presentation of our approach general so that it covers a broad class of discrete deformable objects; the specific setting we use for our experiments is treated in Section 5. We consider a time-continuous motion of a discrete deformable object, represented by a vector valued function $x : t \mapsto \mathbb{R}^{3n}$, whose dynamics are described by a system of second order ODE's of the form

$$M\,\ddot{x}(t) + D(\dot{x}(t)) + G(x(t)) = F(t), \tag{1}$$

where M is a mass matrix and G,D, and F represent the inner forces, the damping, and the outer forces that act on x. The function x may represent the motion of vertices of a triangle or tet mesh, the nodes of a finite element mesh, or the control points of a subdivision surface or solid. Equation (1) covers discretizations with finite elements, finite differences, simple spring systems, as well as geometrically motivated discrete systems, cf. [Terzopoulos et al. 1987; Pentland and Williams 1989; Shabana 1997; Baraff and Witkin 1998; Barbič and James 2005; Chao et al. 2010].

To approximate G around a state $\hat{x}$, we expand

$$G(\hat{x} + u) \approx \hat{G} + K\,u, \tag{2}$$

where $u = x - \hat{x}$, $\hat{G} = G(\hat{x})$, and $K = \frac{\partial}{\partial u} G_{|\hat{x}}$ is the stiffness matrix. Furthermore, we use Rayleigh damping, i.e. we assume that D has the form

$$D(\dot{x}(t)) = (\alpha\,M + \beta\,K)\dot{x}(t).$$

With these approximations the equations of motion take the form

$$M\,\ddot{u}(t) + (\alpha\,M + \beta\,K)\dot{u}(t) + K\,u(t) + \hat{G} = F(t). \tag{3}$$

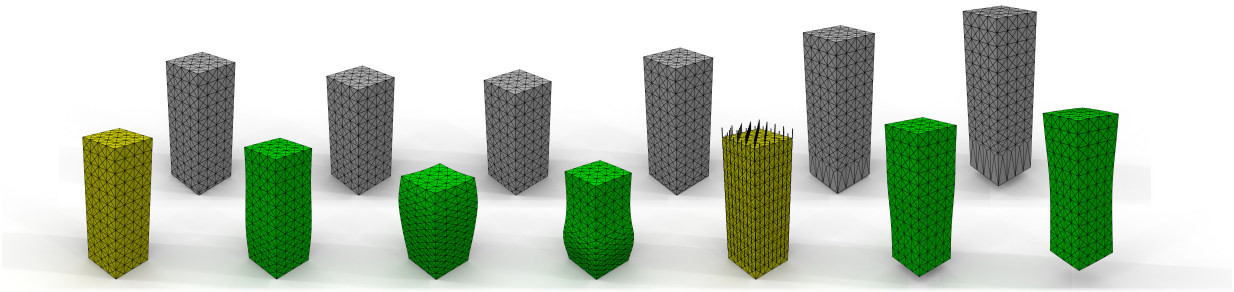

Figure 3: *Snapshots from an animation of a jumping block (green) that interpolates keyframes (yellow) and velocities at the keyframes (black arrows). Before it actually jumps, the block deforms in order to achieve the prescribed veocity at the second keyframe. For comparison, results of interpolation with cubic B-splines (gray) are shown. (Animations are shown in the attached video.)*

The matrix M is symmetric and positive definite. Furthermore, we assume that K is symmetric, which is the case for most models of elasticity considered in graphics (*e.g.* for hyperelastic materials). However, K need not be positive definite, since $\hat{x}$ may not be a rest state of the system.

3.1 Reduced spaces

In most of our experiments, we used a reduced shape space instead of the full space $\mathbb{R}^{3n}$. Our reasons for doing so are threefold. Firstly, the accurate representation of a geometry usually needs high dimensional discrete representation whereas animations often require only a fraction of the degrees of freedom, cf. [Kim and James 2009]. Secondly, the computational cost decreases dramatically. Thirdly, reduced spaces cut off high frequency modes and thereby lower the stiffness of the optimization problem which, in turn, increases the numerical robustness of the resulting method.

We consider a d-dimension reduced space that is a subspace of $\mathbb{R}^{3n}$ and is given by a set of d vectors that form a basis of the space. There are various possible ways to construct such a basis. Our construction uses the keyframes and modes of oscillation around the keyframes. At this point, we keep the presentation independent of the particular choice of a basis and postpone the discussion of our construction of such a basis to Section 5. Let U be a $3n\times$d-matrix whose columns are the basis vectors. Then, U maps coordinates q in the reduced space to the u-coordinates in $\mathbb{R}^{3n}$

$$u = Uq.$$

The reduced mass and stiffness matrices are

$$\bar{M} = U^T M U \quad \text{and} \quad \bar{K} = U^T K U, \tag{4}$$

and the equations of motion in the reduced space are

$$\bar{M}\,\ddot{q}(t) + (\alpha\,\bar{M} + \beta\,\bar{K})\dot{q}(t) + \bar{K}\,q(t) + U^T\hat{G} = U^T F(t). \tag{5}$$

This last equation follows from multiplying (3) with U^T and using (4). For our purposes it is convenient to choose a specific basis in the reduced space, namely a basis in which the reduced mass and stiffness matrices are diagonal matrices. This can be achieved by solving the (low-dimensional) generalized eigenvalue problem

$$\bar{K}\,\phi_i = \lambda_i \bar{M}\phi_i. \tag{6}$$

We assemble the eigenvectors ϕ_i to form the rows of a matrix Φ and denote coordinates with respect to the basis $\{\phi_i\}$ by ω. The transformations from ω to the q- and the u-coordinates are

$$q = \Phi\omega \quad \text{and} \quad u = \tilde{U}\omega, \tag{7}$$

where $\tilde{U} = U\Phi$, and the representation of the mass and the stiffness matrix in the ω-coordinates is

$$\mathbb{1} = \Phi^T\bar{M}\Phi \quad \text{and} \quad \Lambda = \Phi^T\bar{K}\Phi, \tag{8}$$

where Λ is the diagonal matrix whose diagonal entries are the eigenvalues λ_i. Then, the reduced equations of motion take the form

$$\ddot{\omega}(t) + (\alpha\,\mathbb{1} + \beta\,\Lambda)\dot{\omega}(t) + \Lambda\,\omega(t) + g = \tilde{U}^T F(t), \tag{9}$$

where $g = \tilde{U}^T\hat{G}$. The advantage of using the ω-coordinates is that (9) is a system of d independent ODEs, whereas (5) is a coupled system.

4 Interactive Spacetime Control

In this section, we describe the basis of our approach: a model reduction of the spacetime constraint problem for deformable objects. In Section 4.1, we formulate a linearized spacetime constraint problem in the reduced space and show that in the ω-coordinates it decouples to d one-dimensional spacetime problems. In Section 4.2, we derive a scheme for the computation of the exact solution of the one-dimensional problem, which amounts to solving a low-dimensional linear system with a band structure. Section 4.3 describes how we construct the final motion by blending a number of solutions of linearized problems. In Section 4.4 we give an outline of the algorithm and divide it into a preprocess and an interactive phase in which the animation is edited.

The spacetime problem, we consider, is to generate optimal motions that assume prescribed poses $X_0, X_1, ..., X_m$ at times $t_0, t_1, ..., t_m$, where a motion is optimal if the forces required to generate the motion have minimal spacetime L^2-norm (integral over the squared norm of the forces over all points of the deformable object and the time interval) among all competitors. For uniqueness of the solution, we additionally impose boundary conditions on the motion. For example, we fix the initial and final velocities $\dot{x}(t_0) = Y_0$ and $\dot{x}(t_m) = Y_m$. Alternatively, interpolation of second derivatives at the boundary and circular motions are possible.

4.1 Linearized spacetime problem

Let us assume that the reduced space contains all keyframes X_k and the vectors Y_0 and Y_m. If this is not the case, they can simply be added to the reduced basis. To restrict the spacetime problem to a reduced space, we allow only forces that are contained in the reduced space, and to linearize the problem, we assume that the dynamics are described by linearized equations of motion. Then,

the reduced problem can be formulated in the ω-coordinates. The components Ω_k of the keyframes in this basis satisfy $\tilde{U}\,\Omega_k = X_k - \hat{x}$, which follows from (7). Multiplying the equation with $\tilde{U}^T M$ and using (8) shows that this is equivalent to

$$\Omega_k = \tilde{U}^T M (X_k - \hat{x}).$$

Similarly, the vectors Y_0 and Y_m transform by

$$\Psi_0 = \tilde{U}^T M\, Y_0 \quad \text{and} \quad \Psi_m = \tilde{U}^T M\, Y_m.$$

Problem 1 (linearized spacetime problem) *In the space of admissible motions ω find*

$$\arg\min_{\omega} \int_{t_0}^{t_m} \|\ddot{\omega}(t) + (\alpha\,\mathbb{1} + \beta\,\Lambda)\dot{\omega}(t) + \Lambda\,\omega(t) + g\|^2 \; dt, \quad (10)$$

where ω is admissible if $\omega \in C^2([t_0, t_m], \mathbb{R}^d)$ and

$$\omega(t_k) = \Omega_k \quad \forall\, k \in \{0, 1, ..., m\},$$
$$\dot{\omega}(t_0) = \Psi_0, \quad \textit{and} \quad \dot{\omega}(t_m) = \Psi_m$$

holds.

The norm we use in (10) is simply the standard norm of $\mathbb{R}^d$. Since the mass matrix in the ω-coordinates is the identity matrix, this norm is a discrete L^2-norm for functions on the deformable object. The advantage of using the ω-coordinates is that the problem decouples, *i.e.* component functions $\omega_i(t)$ and $\omega_j(t)$ do not influence each other (for $i \neq j$). Hence, we only need to solve the one-dimensional problems

$$\arg\min_{\omega_i} \int_{t_0}^{t_m} \left(\ddot{\omega}_i(t) + 2\,\delta_i\dot{\omega}_i(t) + \lambda_i\,\omega_i(t) + g_i\right)^2 \, \mathrm{d}t \quad (11)$$

subject to

$$\omega_i(t_k) = (\Omega_k)_i \quad \forall\, k \in \{0, 1, ..., m\},$$
$$\dot{\omega}_i(t_0) = (\Psi_0)_i, \quad \text{and} \quad \dot{\omega}_i(t_m) = (\Psi_m)_i,$$

for all $i \in \{1, 2, ..., d\}$. Here we set $\delta_i = \frac{1}{2}(\alpha + \beta\,\lambda_i)$. Once a solution $\omega(t)$ in the reduced space is computed, the matrix $\tilde{U}$ maps $\omega(t)$ to a motion $u(t)$ in $\mathbb{R}^{3n}$, see also eq. (7).

Remark 1 *We would like to mention that Problem 1 can be motivated as an application of Gauß' principle of least constraint [Gauß 1829] to keyframe interpolation.*

4.2 Explicit wiggly splines

Kass and Anderson [2008] considered the one-dimensional spacetime constraint problem (11) and called the solutions *wiggly splines*. They used a finite difference method to solve the problem. Here we propose an alternative scheme to solve the problem based on a closed-form representation of the wiggly splines.

The Euler-Lagrange equation of the one-dimensional problem (11) is the fourth order ODE

$$\dddot{\omega}_i(t) + 2(\lambda_i - 2\delta_i^2)\,\ddot{\omega}_i(t) + \lambda_i^2\,\omega_i(t) + \lambda_i\,g_i = 0. \quad (12)$$

This can be verified by a lengthy but straightforward calculation. The solutions of this equation form a 4-dimensional (affine) vector space. The type of functions that are in this space depends on relations of λ_i and δ_i. We classify these different types into six

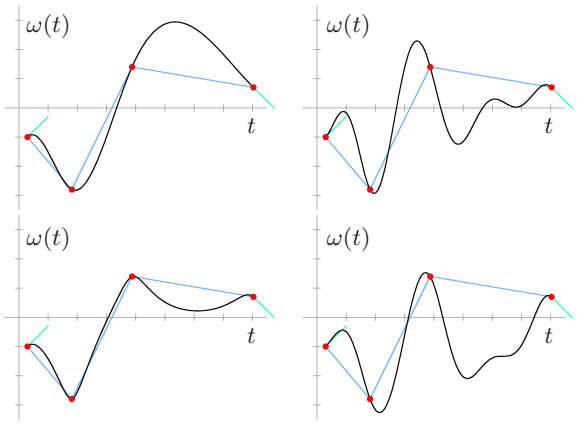

Figure 4: *Wiggly splines (black) with varying stiffness and damping parameters, interpolation points (red), polygon connecting the points (blue), and boundary conditions (turquois). The parameters are: $\lambda = 0, \delta = 0$ (top left), $\lambda = 5, \delta = 0$ (top right), $\lambda = 5, \delta = 5$ (bottom left), $\lambda = 5, \delta = 0$, and an additional constant gravitational force (bottom right).*

cases and explicitly list basis functions $\{b_i^1(t), b_i^2(t), b_i^3(t), b_i^4(t)\}$ that span the spaces of solutions for all cases in the appendix. The minimizer $\omega_i(t)$ of the 1-D spacetime problem is a function that is four times continuously differentiable and satisfies (12) within all intervals (t_k, t_{k+1}) and is twice continuously differentiable at the t_ks. It follows that the restriction of the minimizer to any interval $[t_{k-1}, t_k]$ is a combination of the basis functions. We denote by $\omega_{i,k}(t)$ the restriction of $\omega_i(t)$ to the interval $[t_{k-1}, t_k]$. Then, for every $k \in \{1, 2, ..., m\}$ there are four coefficients $w_{i,k}^1, w_{i,k}^2, w_{i,k}^3$, and $w_{i,k}^4$ such that

$$\omega_i(t)_{|[t_{k-1}, t_k]} = \omega_{i,k}(t) = \sum\nolimits_{l=1}^{4} w_{i,k}^l b_i^l(t) - c_i,$$

where $c_i = g_i / |\lambda_i|$ if $\lambda_i \neq 0$ and $c_i = 0$ if $\lambda_i = 0$. To obtain $\omega_i(t)$, we need to compute the $4m$ coefficients $w_{i,k}^l$ for all $k \in \{1, 2, ..., m\}$ and $l \in \{1, 2, 3, 4\}$. These are determined by the interpolation conditions

$$\omega_{i,k}(t_{k-1}) = (\Omega_{k-1})_i \quad \text{and} \quad \omega_{i,k}(t_k) = (\Omega_k)_i \quad (13)$$

for all $k \in \{1, 2, ..., m\}$, the regularity assumptions

$$\dot{\omega}_{i,k}(t_k) = \dot{\omega}_{i,k+1}(t_k) \quad \text{and} \quad \ddot{\omega}_{i,k}(t_k) = \ddot{\omega}_{i,k+1}(t_k) \quad (14)$$

for all $k \in \{1, 2, ..., m-1\}$, and the boundary conditions that specify the velocities at t_0 and t_m

$$\dot{\omega}_{i,1}(t_0) = (\Psi_0)_i \quad \text{and} \quad \dot{\omega}_{i,m}(t_m) = (\Psi_m)_i. \quad (15)$$

Together these $4\,m$ linear equations determine a unique spline $\omega_i(t)$. The conditions can be arranged to form a band matrix with bandwidth 8. Since m is typically small (*e.g.* $m = 10$), solving such a system requires only fractions of a ms; even on a custom laptop and without parallelization, one can compute 10 k wiggly splines within a second.

Remark 2 (complex wiggly splines) *In this paper, we use only solutions of the 1-D spacetime constraint problem over the real numbers. However, the complex solutions are interesting as well; Kass and Anderson [2008] show great examples of how they can be used for designing and augmenting motions of characters. Our algorithm directly generalizes to the calculation of complex wiggly splines. The space of complex solutions of the Euler-Lagrange*

equation (12), in the generic case where $\delta_i \neq 0$ and $\delta_i^2 - \lambda_i \neq 0$, is spanned by the four complex functions

$$b_i^l(t) = e^{\left(\pm\delta_i \pm \sqrt{\delta_i^2 - \lambda_i}\right)t}. \tag{16}$$

Then, to determine a complex wiggly spline, the interpolation (13), continuity (14), and boundary constraints (15) need to be treated as complex equations.

The proposed computation of the wiggly splines has three major advantages over a finite difference scheme. The first is that the result is a wiggly spline in closed form, not just an approximation. The second is a significantly lower computational cost. Thirdly, we do not need to deal with stability issues caused by discretization. Roughly speaking, the problem with stability is that the continuous equation is stable, but once continuous derivatives are replaced by finite differences, stability is no longer guaranteed. Frequencies higher than those representable at the used sampling rate can destabilize the equation. For a throughout discussion of the stability problem, we refer to Kass and Anderson [2008].

4.3 Multipoint linearization and blending

A problem when using linearized equations of motion (3) is that artifacts may develop for larger deformations away from the point $\hat{x}$ (around which the inner forces were expanded). To counteract such problems, we use the fact that the desired motion interpolates the keyframes. We expand the inner forces around every keyframe X_j and get $(m+1)$ linearized equations of motion. Then, we solve the corresponding linearized spacetime problems to compute $(m+1)$ motions $x^j(t)$. From these solutions, we assemble a final motion $x(t)$ by blending over each interval $[t_j, t_{j+1}]$ the trajectories $x^j(t)$ and $x^{j+1}(t)$ with cubic splines. In our implementation, we used only the motion $x^j(t)$ in the interval $[t_j-(t_j-t_{j-1})/4, t_j+(t_{j+1}-t_j)/4]$ and blended in the intervals $[t_j + (t_{j+1} - t_j)/4, t_{j+1} - (t_{j+1} - t_j)/4]$.

A limitation of this approach is that the deformation from one keyframe to its successor should not be too large. However, since the points around which we linearize the equations of motion need not be interpolated by the motion, one could add *ghost meshes* between keyframes and linearize the equations around these extra meshes without forcing interpolation of these meshes. However, for our examples, we did not use such a technique.

4.4 Algorithm outline

We divide the computation into two parts: a preprocess and an interactive editing phase. The following is shifted to the preprocess

1. a basis of the reduced space is calculated and stored in the matrix U
2. for every keyframe, the mass and stiffness matrices M and K and their reduced counterparts $\bar{M}$ and $\bar{K}$ are set up
3. for every pair of reduced matrices $\bar{M}$ and $\bar{K}$ the eigenvalue problem (6) is fully solved and the matrices Φ and Λ as well as the vector g are set up

Then, the interactive phase begins. To compute the animation, we only need to compute the $(m+1)$ motions and blend them to get the final motion. This means that $(m+1) * d$ wiggly splines must be computed, which (for settings like $m = 10$ and $d = 100$) takes only fractions of a second. Editing operations to change

- physical parameters, like material stiffness or damping coefficients

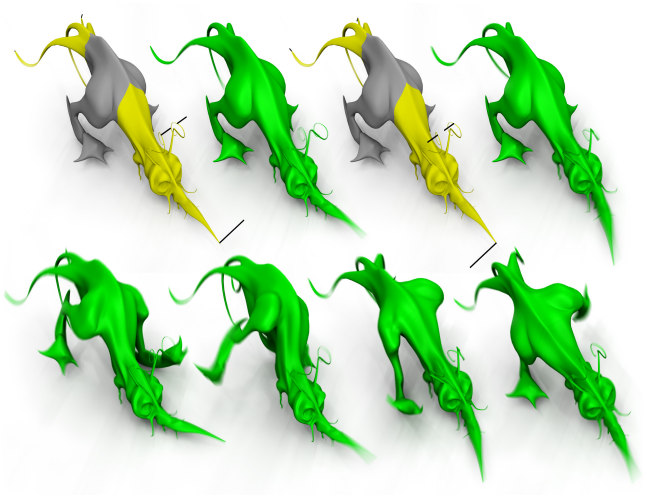

Figure 5: *Animation of a walking bug enriched with secondary motion. Top row shows only the secondary motion and the keyframes that generate the motion.*

- boundary conditions
- times of the keyframes
- positions in space of keyframes

require only a recomputation of the wiggly splines, hence take only fractions of a second, as well. Additional effort is required for modeling or adding a keyframe. However, we do not need to rerun the whole preprocess. Instead, we can keep the reduced space, which was computed in the preprocess, and add some basis vectors. Similarly, we only need to compute the stiffness matrix K and mass matrix M of the new (or modified) keyframe and we keep the (unreduced) matrices of all other keyframes. Only the operations in the reduced space (that is setting up the reduced matrices and solving the reduced eigenvalue problems) must be carried out for all keyframes. Since the reduced space is low-dimensional, these operations come at low computational costs. Alternatively to augmenting the reduced space, one can project the new (or modeled) keyframe into the reduced space. Such a strategy further reduces the computation effort but modifies the keyframe. A nice approach is to create or model new keyframes directly in the reduced space.

5 Our Implementation

Our technique can be used for different types of discrete elastic objects; for our implementation, we decided to use *Discrete Shells* [Grinspun et al. 2003; Bergou et al. 2007]. Discrete Shells are formulated for triangle meshes, which then represent the middle surface of an elastic shell. The potential energy of the inner forces splits into two parts, a flexural (or bending) and a membrane (or stretching) energy. Both parts can be computed from elementary geometric properties of the triangle mesh. For explicit formulae we refer to [Grinspun et al. 2003]. The vector $\hat{G}$ and the matrix K in (2) are the gradient and the Hessian of the potential energy at the point $\hat{x}$. To compute the derivatives, we used the automatic-differentiation library *ADOL-C*, see [Griewank et al. 1996]. In most of our experiments, we set both material parameters (weighting of the flexural and membrane energies) to the value 1, and, additionally, we scaled the meshes to a diameter of 10. The latter is important since the membrane and flexural energies scale differently. An advantage of using the Discrete Shells, compared to using elastic solids, is that we could directly use keyframes gener-

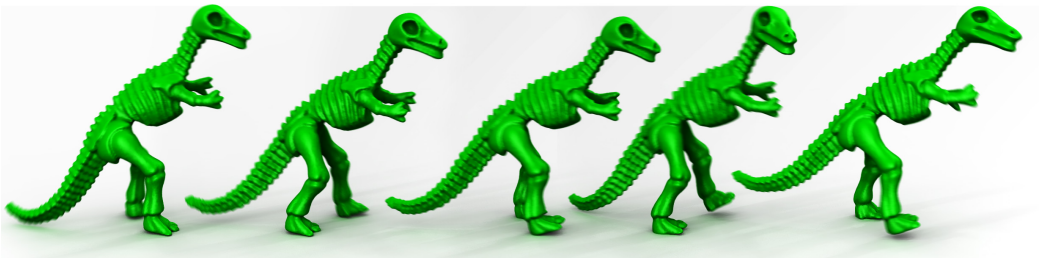

Figure 6: *Physically based soft-body walking dinosaur that interpolates animator-specified keyframes.*

ated with a surface modeler, thus avoiding the need to generate and model keyframes with tet meshes.

To construct the reduced space, we collect vectors in a set S and then use a SVD to obtain an orthonormal basis of the linear span of S. We store the basis in the matrix U. The set S consists of two parts: one collects the vectors pointing to the keyframes and the other contains 5-30 vibration modes of each of the keyframes. The computation of the vibration modes follows [Hildebrandt et al. 2012]. For this we set the keyframe to be the rest state of the discrete shells' potential energy and set up the Hessian K and the mass matrix. Then, we compute the modes corresponding to the lowest eigenfrequencies. These are solutions of the generalized sparse eigenvalue problem

$$K\,\varphi_i = \lambda_i\,M\,\varphi_i.$$

In our experiments, this construction produced good results. We did not observe the typical artifacts that one gets when working with reduced spaces constructed from vibration modes around only one state.

For small-size meshes, one can solve the full problem instead of using a reduced space. For example, we computed the motion of the block, see Figure 3, with 450 vertices in the full 1350 dimensional space. This demonstrates that we can work with larger spaces compared to methods that solve the non-linear problem in a reduced space, like [Barbič et al. 2009], and use 20-30 dimensional reduced spaces.

6 Experiments

The attached video shows five examples of motions produced with the proposed technique and Table 1 shows the size of the meshes and the reduced spaces as well as run times. In this section, a wiggly spline means a vector-valued spline that describes a motion and not a one-dimensional spline.

The first example, the jumping block, is composed of three wiggly splines that interpolate four keyframes. This divides the motion into three parts: preparing the jump, jumping, and landing. During the first and the last part, the base of the block is fixed to the ground with equality constraints. Each of the three wiggly splines interpolates two successive keyframes and satisfies boundary conditions, which prescribe first derivatives (velocities) at all vertices of both keyframes. The prescribed velocities at the first keyframe vanish, and all vertices of the second keyframe, except those at the base, have a the same upwards pointing velocity. At the second keyframe the first and second wiggly spline meet. Both splines interpolate the second keyframe and have the same velocities at this point. Therefore, the compound curve is differentiable at this keyframe. As a results, the motion anticipates the jump and the force required to accelerate the block is distributed over the time interval. All vertices of the third keyframe (including the base) have the same velocity pointing into the ground. This models a collision with ground. Since the base of the block is fixed in the third wiggly spline, the compound curve is continuous but not differentiable at the third keyframe. The effect is that there is no anticipation of the landing, the block collides with the ground. Since the block cannot bounce upwards, the collision causes a deformation of the block: a wave is traversing upwards the block. We show two motions of the jumping block with varying material stiffness (*i.e.*, membrane and flexural energy scaled up with the same parameter). For comparison, a motion generated with cubic B-splines that satisfy the same interpolation and boundary conditions is shown. Whereas our motions show physical behavior, like waves that are traveling up and down the block, the B-spline animation only varies the length of the block. Some stills of the animation are shown in Fig. 3. We want to note that since all four keyframes have the same geometry, this animation is produced from one linearized spacetime-constraint problem.

The second example, the running bug, shows a walking cycle that is enriched with secondary motion. The primary motion interpolates 12 keyframes that form a walking cycle, which we took from the book [Ritchie et al. 2005]. The circular wiggly spline used here does not require any boundary conditions and is fully determined by the interpolation conditions. To get secondary motion that does not need to interpolate the keyframes, we blended the walking cycle motion with a second motion. For this second motion, we used two keyframes and constrained a part of the models (gray area in Fig. 5) that has been carefully crafted and should not be affected by secondary motion. Then we computed a reduced space and added simple vector fields to the keyframes (black arrows). We used the projections of these vector fields into the reduced space as derivatives of the motion and forced the wiggly splines to interpolate them.

The third example, shows a piece of cloth that first hangs on a line, then is blown into the air, and finally gets stuck on a cylinder. The motion is assembled from three wiggly splines. The first and the last wiggly spline satisfy equality constraints that fix parts of the piece of cloth to the line respectively the cylinder. The keyframes were generated by taking snapshots from three different forward simulations: cloth hanging on a line, cloth blown in the air, and cloth stuck on a cylinder. The benefit of our approach is that we can combine these (otherwise unrelated) animations into one animation in which the cloth is flying through the air and hits the cylinder.

The forth example, shows a walking dinosaur (Fig. 6). The motion is composed of five wiggly splines, one for each step. Each step is modeled with three keyframes and the foot on the ground is fixed with equality constraints. The boundary conditions for each of the wiggly splines are finite differences of the keyframes. For comparison we show results of the non-linear method of Barbič et al. [2009]. To increase comparability, we used reduced spaces of approximately the same size. A difference of the methods is that their scheme does not interpolate, but approximates the keyframes.

Model	#Verts	#S	Dim.	#K	t_p	t_u
Dinosaur	28098	5	5×18	11	505	0.6
Bug (primary)	4358	1	200	12	89	1.9
Bug (secondary)	4358	1	30	1	9	0.03
Cloth	1861	3	92,143,186	12	74	0.85
Block	450	3	3×1350	6	83	0.54
Flour Sack	1762	1	124	7	20	0.25

Table 1: *Performance measured on a desktop PC with an Intel Xeon 3.33 GHz CPU (single thread). From left to right: number of vertices, number of composite splines, dimension of reduced space, number of keyframes, time in seconds for the complete preprocess and for an update of the full animation.*

The original keyframes, they used are tet meshes that enclose the dinosaur; they used the triangle meshes only for rendering the animation. These tet meshes have 1.8 k vertices, hence are coarser then the triangle meshes, which have 28 k vertices. For our animation, we used the higher resolved triangle meshes. This explains why we need 500 s to construct the reduced space for the whole animation, while they need only 25 s. After the preprocess, our method needs 0.6 s to compute or update the animation, while their optimization takes 15 min.

The last example is a live demo. It shows how we generate an animation of a dancing flour sack. The motion is generated by a circular wiggly spline that interpolates eight keyframes (made from 4 poses). The video shows how times of keyframes are adjusted and material parameters are varied. This demonstrates the interactive response times that our technique offers. The live demo is followed by demo of a wiggly spline editor. It illustrates the effect of variations of the frequency, the damping coefficient, the interpolation constraints, and the boundary conditions as well as a constant (gravitational) force.

When creating keyframes with a surface modeler, it is preferable to use a physical-based modeler that takes the Discrete Shells energy into account. When using keyframes that were generated with other modelers (*e.g.* the bug and the dino models), we did an optimization routine to lower the Discrete Shells energy of the shapes. Explicitly, we minimize a weighted sum of the Discrete Shells energy and the least-squares distances of the vertices to their original position.

7 Conclusion

We presented a technique that implements the spacetime constraints paradigm for animations of deformable objects. It combines physical realism with control over the animation. The main novelty of the approach is that it offers interactive response times, hence allows for interactive adjustment of physical parameters or editing of control parameters, like times of keyframes. This is achieved by a model reduction of the underlying variational spacetime problem, which combines dimension reduction, a multipoint linearization strategy, and decoupling of the reduced equations of motion. After reduction, a number of one-dimensional variational problems need to be solved. The solutions of these problems are known as wiggly splines. A second main contribution of this paper is a fast and robust numerical scheme for computing wiggly splines that is based on an explicit representation of the wiggly splines. Without this efficient numerical scheme, our method would hardly be interactive. Our current implementation does not use parallelization or graphic cards. Still, the preprocess as well as the computation of the wiggly splines are highly parallelizable.

7.1 Limitations and challenges

In its current form, our scheme requires that every keyframe prescribes a pose for the whole object. We are working on an extension of our scheme that can handle keyframes that prescribe only the positions of some parts of the object (and leave the rest free). Furthermore, we want to extend the scheme such that motions that do not interpolate, but only approximate the keyframes can be produced. A challenge for our scheme (and for schemes that follow the spacetime constraint paradigm in general) is the integration of contact and collision handling. Currently, only limited handling of contact is possible. In our experiments, we prescribe incoming and outgoing velocities at a keyframe and fix parts on the object in space by equality constraints to model the contacts of the landing block, the walking dino (whose feet hit the ground), and the cloth hitting the cylinder. A interesting direction of future work is to develop counterparts of the presented approach for other types of physical systems, like fluids or particles systems, and for parametrized characters. Furthermore, we expect to see more application of the wiggly splines. We see them as a powerful alternative to traditional splines that offers oscillatory behavior.

Acknowledgements. We would like to thank Jernej Barbič and Kiaran Ritchie for sharing their animations and keyframes and the anonymous reviewers for their comments and suggestions. This work was supported by the DFG Research Center MATHEON "Mathematics for Key Technologies" in Berlin.

References

BARAFF, D., AND WITKIN, A. 1998. Large steps in cloth simulation. In *Proc. of ACM SIGGRAPH*, 43–54.

BARBIČ, J., AND JAMES, D. L. 2005. Real-time subspace integration for St. Venant-Kirchhoff deformable models. *ACM Trans. Graph. 24*, 3, 982–990.

BARBIČ, J., DA SILVA, M., AND POPOVIĆ, J. 2009. Deformable object animation using reduced optimal control. *ACM Trans. Graph. 28*, 53:1–53:9.

BARZEL, R. 1997. Faking dynamics of ropes and springs. *IEEE Comput. Graph. Appl. 17*, 31–39.

BERGOU, M., MATHUR, S., WARDETZKY, M., AND GRINSPUN, E. 2007. TRACKS: Toward Directable Thin Shells. *ACM Trans. Graph. 26*, 3.

CHAI, J., AND HODGINS, J. K. 2007. Constraint-based motion optimization using a statistical dynamic model. *ACM Trans. Graph. 26*.

CHAO, I., PINKALL, U., SANAN, P., AND SCHRÖDER, P. 2010. A simple geometric model for elastic deformations. *ACM Trans. Graph. 29*, 38:1–38:6.

COHEN, M. F. 1992. Interactive spacetime control for animation. *Proc of ACM SIGGRAPH 26*, 293–302.

FANG, A. C., AND POLLARD, N. S. 2003. Efficient synthesis of physically valid human motion. *ACM Trans. Graph. 22*, 417–426.

GAUSS, C. F. 1829. Über ein neues allgemeines Grundgesetz der Mechanik. *J. Reine Angew. Math. 4*, 232–235.

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

GRIEWANK, A., JUEDES, D., AND UTKE, J. 1996. Algorithm 755: ADOL-C: a package for the automatic differentiation of algorithms written in C/C++. *ACM Trans. Math. Softw. 22*, 2, 131–167.

GRINSPUN, E., HIRANI, A. N., DESBRUN, M., AND SCHRÖDER, P. 2003. Discrete shells. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 62–67.

GUENTER, B. 2007. Efficient symbolic differentiation for graphics applications. *ACM Trans. Graph. 26*.

HILDEBRANDT, K., SCHULZ, C., VON TYCOWICZ, C., AND POLTHIER, K. 2011. Interactive surface modeling using modal analysis. *ACM Trans. Graph. 30*, 119:1–119:11.

HILDEBRANDT, K., SCHULZ, C., VON TYCOWICZ, C., AND POLTHIER, K. 2012. Modal shape analysis beyond Laplacian. *Computer Aided Geometric Design 29*, 5, 204–218.

HUANG, J., SHI, X., LIU, X., ZHOU, K., WEI, L.-Y., TENG, S.-H., BAO, H., GUO, B., AND SHUM, H.-Y. 2006. Subspace gradient domain mesh deformation. *ACM Trans. Graph. 25*, 3, 1126–1134.

IDELSOHN, S. R., AND CARDONA, A. 1985. A reduction method for nonlinear structural dynamic analysis. *Comput. Meth. Appl. Mech. Eng. 49*, 3, 253 – 279.

KASS, M., AND ANDERSON, J. 2008. Animating oscillatory motion with overlap: wiggly splines. *ACM Trans. Graph. 27*, 28:1–28:8.

KILIAN, M., MITRA, N. J., AND POTTMANN, H. 2007. Geometric modeling in shape space. *ACM Trans. Graph. 26*.

KIM, T., AND JAMES, D. L. 2009. Skipping steps in deformable simulation with online model reduction. *ACM Trans. Graph. 28*, 123:1–123:9.

KRYSL, P., LALL, S., AND MARSDEN, J. E. 2001. Dimensional model reduction in non-linear finite element dynamics of solids and structures. *Int. J. Numer. Meth. Eng. 51*, 479–504.

MCNAMARA, A., TREUILLE, A., POPOVIĆ, Z., AND STAM, J. 2004. Fluid control using the adjoint method. *ACM Trans. Graph. 23*, 449–456.

NICKELL, R. 1976. Nonlinear dynamics by mode superposition. *Comput. Meth. Appl. Mech. Eng. 7*, 1, 107 – 129.

PENTLAND, A., AND WILLIAMS, J. 1989. Good vibrations: modal dynamics for graphics and animation. *Proc. of ACM SIGGRAPH 23*, 207–214.

POPOVIĆ, J., SEITZ, S. M., AND ERDMANN, M. 2003. Motion sketching for control of rigid-body simulations. *ACM Trans. Graph. 22*, 1034–1054.

RITCHIE, K., CALLERY, J., AND BIRI, K. 2005. *The Art of Rigging*. CG Toolkit.

SAFONOVA, A., HODGINS, J. K., AND POLLARD, N. S. 2004. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Trans. Graph. 23*, 514–521.

SHABANA, A. 1997. *Theory of Vibration II: Vibration of Discrete and Continuous Systems*, 2nd ed. Springer Verlag.

SULEJMANPAŠIĆ, A., AND POPOVIĆ, J. 2005. Adaptation of performed ballistic motion. *ACM Trans. Graph. 24*, 165–179.

TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. 1987. Elastically deformable models. In *Proc. of ACM SIGGRAPH*, 205–214.

TREUILLE, A., MCNAMARA, A., POPOVIĆ, Z., AND STAM, J. 2003. Keyframe control of smoke simulations. *ACM Trans. Graph. 22*, 716–723.

TREUILLE, A., LEWIS, A., AND POPOVIĆ, Z. 2006. Model reduction for real-time fluids. *ACM Trans. Graph. 25*, 826–834.

WICKE, M., STANTON, M., AND TREUILLE, A. 2009. Modular bases for fluid dynamics. *ACM Trans. Graph. 28*, 39:1–39:8.

WIRTH, B., BAR, L., RUMPF, M., AND SAPIRO, G. 2009. Geodesics in shape space via variational time discretization. In *Proc. of the 7th Intern. Conf. on Energy Minimization Methods in Computer Vision and Pattern Recognition*, 288–302.

WITKIN, A., AND KASS, M. 1988. Spacetime constraints. *Proc. of ACM SIGGRAPH 22*, 159–168.

WOJTAN, C., MUCHA, P. J., AND TURK, G. 2006. Keyframe control of complex particle systems using the adjoint method. In *Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 15–23.

Appendix

In this appendix, we explicitly describe the basis functions spanning the (real) spaces of solutions of the Euler–Lagrange equation (12). We distinguish between the two main cases and we note four special cases. For brevity, we set $\eta_i = \sqrt{|\delta_i^2 - \lambda_i|}$, which is the frequency in the case of oscillation. In the first case, $\delta_i^2 - \lambda_i < 0$, the mesh oscillates in the direction of the mode ϕ_i (the square root in (16) is imaginary) and the basis functions are:

$$\begin{array}{ll} b_i^1(t) = e^{-\delta_i t}\cos(\eta_i t) & b_i^2(t) = e^{-\delta_i t}\sin(\eta_i t) \\ b_i^3(t) = e^{\delta_i t}\cos(\eta_i t) & b_i^4(t) = e^{\delta_i t}\sin(\eta_i t) \end{array}$$

In the second case, $\delta_i^2 - \lambda_i > 0$, the mesh exponentially decays or grows in the direction of ϕ_i and the basis functions are:

$$\begin{array}{ll} b_i^1(t) = e^{(-\delta_i+\eta_i)t} & b_i^2(t) = e^{(-\delta_i-\eta_i)t} \\ b_i^3(t) = e^{(\delta_i+\eta_i)t} & b_i^4(t) = e^{(\delta_i-\eta_i)t} \end{array}$$

The special cases are

1. $\delta_i = 0$ and $\lambda_i > 0$:

$$\begin{array}{ll} b_i^1(t) = \cos(\eta_i t) & b_i^2(t) = \sin(\eta_i t) \\ b_i^3(t) = t\,\cos(\eta_i t) & b_i^4(t) = t\,\sin(\eta_i t) \end{array}$$

2. $\delta_i = 0$ and $\lambda_i < 0$ or $\eta_i = 0$ and $\lambda_i > 0$:

$$\begin{array}{ll} b_i^1(t) = e^{-\sqrt{|\lambda_i|}t} & b_i^2(t) = e^{\sqrt{|\lambda_i|}t} \\ b_i^3(t) = t\,e^{-\sqrt{|\lambda_i|}t} & b_i^4(t) = t\,e^{\sqrt{|\lambda_i|}t} \end{array}$$

3. $\delta_i \neq 0$ and $\lambda_i = 0$:

$$\begin{array}{ll} b_i^1(t) = 1 & b_i^2(t) = t \\ b_i^3(t) = \frac{e^{-2\delta_i t}}{4\delta_i^2} & b_i^4(t) = \frac{e^{2\delta_i t}}{4\delta_i^2} \end{array}$$

4. $\delta_i = \lambda_i = 0$: in this case $\omega_i(t)$ is a B-spline

$$\begin{array}{ll} b_i^1(t) = 1 & b_i^2(t) = t \\ b_i^3(t) = t^2 & b_i^4(t) = t^3 \end{array}$$