Adaptive resolution of 1D mechanical B-spline

Julien Lenoir, Laurent Grisoni, Philippe Meseure, Christophe Chaillou

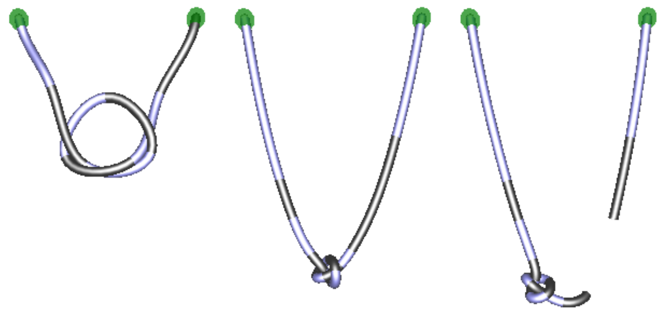
Adaptive resolution of 1D mechanical B-spline

Julien Lenoir[*]
CIMIT, The SimGroup

Laurent Grisoni[†]
ALCOVE, INRIA Futurs, LIFL

Philippe Meseure[‡]
SIC, University of Poitiers

Christophe Chaillou[§]
ALCOVE, INRIA Futurs, LIFL



(a) Fixed resolution (b) Adaptive resolution (c) Cutting

Figure 1: Knot tying and cutting

Abstract

This article presents an adaptive approach to B-spline curve physical simulation. We combine geometric refinement and coarsening techniques with an appropriate continuous mechanical model. We thus deal with the (temporal and geometric) continuity issues implied when mechanical adaptive resolution is used. To achieve real-time local adaptation of spline curves, some criteria and optimizations are shown. Among application examples, real-time knot tying is presented, and curve cutting is also pointed out as a nice side-effect of the adaptive resolution animation framework.

CR Categories: I.3.5 [Computing Methodologies]: COMPUTER GRAPHICS—Computational Geometry and Object Modeling: Splines I.3.7 [Computing Methodologies]: COMPUTER GRAPHICS—Three-Dimensional Graphics and Realism: Animation I.6.0 [Computing Methodologies]: SIMULATION AND MODELING—General I.6.8 [Computing Methodologies]: SIMULATION AND MODELING—Types of Simulation: Continuous

Keywords: Adaptive resolution, Real-time simulation, Mechanical 1D model

1 Introduction

Real thready objects have a spatial masses distribution but mostly located along a skeleton. Usually, thready model is defined as a 1D model representing the skeleton of the real object, showing most common behavior of the real object. Real time simulation implies to limit the computation time of the model. But a model also has to have a certain number of degrees of freedom to be able to exhibit complex behavior. One way to deal with both limitations is to use an adaptive model that changes locally and dynamically its own resolution.

In this paper, we propose a new approach to simulate adaptive B-splines with respect to physical properties and continuity during the simulation. Our main idea is to benefit from the huge literature concerning spline's subdivisions [Finkelstein and Salesin 1994] and adapt selected methods to mechanical purposes. Our main problem is thus to properly define a mechanical model which can naturally endure both refinement and coarsening operations. We will see that this implies to define the physical properties such as mass, damping or elasticity in a continuous way. Deformations of the model are not the main issue of this paper so that we only

[*]e-mail: Julien.Lenoir@lifl.fr
[†]e-mail: Laurent.Grisoni@lifl.fr
[‡]e-mail: philippe.meseure@sic.sp2mi.univ-poitiers.fr
[§]e-mail: Christophe.Chaillou@lifl.fr

deal with the stretching deformation introducing a physical structure into the model. A Typical problem of adaptive 1D model is to tie a knot. We show as a result that our proposition permits to handle such a problem. We also point out that our technique can be used to dynamically cut a curve at any location.

This paper is organized in the following way. After the presentation of related work in section (2), we expose the spline subdivision technique and the physical simulation of splines in section (3). In section (4), we show how we combine these two techniques to get a mechanical adaptive simulation of curves. Section (5) shows results and some applications of our work before concluding the paper.

2 Related Work

Adaptive techniques are useful for 1D physical model in case of extreme deformations like knot tightening. This special manipulation of knot tying has been studied this last years. For example, [Wang et al. 2005] propose a discrete model based on point masses linked by a set of springs providing stretching, bending and twisting deformations. Unfortunately, the model is not adaptive, so that a random tying needs a very strong discretised thread. Another way to deal with knot tying is to use a non physics based model, like [Brown et al. 2004]. Their model is defined by a set of points moving by "follow the leader" rules. Unfortunately, this technique does not permit to take into account all physical behavior like [Wang et al. 2005] pointed out. Moreover, this model is not adaptive so that it is also subjected to a strong discretisation.

The mechanical multi-resolution technique has been studied for a few years to overcome the cost of computing the behaviour of a huge number of points of a physical model. A coarse model is progressively refined in order to increase the number of points only where a high precision is needed, generally where interactions occur. Two main kinds of approaches have been proposed depending on whether an ideal continuous model exists or not.

The first methods do not rely on a continuous model and describe only how the resolution of a discrete model can be refined or coarsened. This process is applied depending on the needed spatial or temporal frequencies of an interaction [Luciani et al. 1995]. Such approaches have focused on spring/mass nets [Hutchinson et al. 1996; Ganovelli et al. 1999]. They consist in adding or removing points of the net, and connecting them with the points of the lower resolution. However, springs have drawbacks. Indeed, splitting a spring into several implies higher elasticity constants (and a less stable integration of the dynamics equations). In practice, a spring is usually split at its middle, and the elasticity constants are doubled. This technique keeps the resolution away from an exact adaptation, leading to an iterative process. Moreover, the distribution of masses between the implied points is not straightforward. To avoid such distribution, Eberhardt et al. propose to use virtual particles [Etzmuss et al. 2000]. [Phillips et al. 2002] used a similar model to simulate knot tying. When the resolution changes, new nodes are inserted or removed and the total mass and moment are kept. But, as springs are placed on the nodes, the configuration changes during a changing resolution as does the behaviour.

The second family of approaches relies on a continuous "ideal" model which is discretised at different resolutions. Finite-element or finite-difference methods are used [Astley and Hayward 1997; Wu et al. 2001; Debunne et al. 2001]. Compared to the previous family, these methods guarantee that the physical laws do not depend on the discretisation level, since they refer to the continuous model. Nevertheless, the finer the discretisation is, the more precise the behaviour becomes (levels appear as low-band filters). To allow the use of different levels depending on where the interaction occurs, the different resolution levels must be linked. Thus, these are defined by splitting elements or using higher order basis functions [Grinspun et al. 2002], which allow to control geometric continuity between elements of different levels. Debunne et al. proposed a method to link independently-built discretisation levels together [Debunne et al. 2001]. Another way to build the different discretisation levels is to use multi-resolution Free-Form Deformation [Nocent et al. 2001; Capell et al. 2002]. One of the major concerns of all these methods is to ensure continuity when a switch occurs between levels. What is more, all of these techniques usually involve some pre-computation on regular, pre-set, subdivisions of the initial model. Providing point insertion at any location on the object usually involves dense refinement around point location, and only provides approximate adaptivity. In other words, the adaptation is surely local, but not sufficiently localised.

Our approach falls into the second family. Yet, in our model, we do not have to deal with the continuity problems between elements of different levels or during a level switching. The main idea is to rely on a continuous object which is discretised by a variable number of degrees of freedom. All the physical degrees of freedom are only based on the geometry. By using methods which guarantee the geometric invariance when adding or removing degrees of freedom, we naturally ensure the temporal continuity of the shape and its behaviour. As well, no pre-computation is needed, and as is shown in this article, B-spline multi-resolution straightforwardly provides insertion at any location. It can be pointed out that our technique is not a multi-resolution one as it does not deal with multiple resolutions at the same time. Our model is based on one adaptive resolution.

3 B-spline curve physical simulation

In this section we first define the notation used in the remainder of this article, and recall some of the B-spline's properties used to adapt the resolution. We then describe classical process for B-splines physical animation, and the general framework we use.

3.1 B-spline curve definition and properties

A B-spline curve is defined using classical spline definition:

$$C(s) = \sum_{i=0}^{N} q_i b_{i,d}(s) \tag{1}$$

In the above equation, q_i are the *control points* of the curve C. The functions $b_{i,d}$ are piecewise polynomials of degree d, defined using a *knot vector*, sorted list of scalar values:

$$T = (s_0, s_1, \ldots, s_M) \text{ with } s_0 \leq s_1 \leq \ldots \leq s_M$$

From such a knot vector, classical Cox-DeBoor recursive definition[Cox 1972; de Boor 1972] is used either to evaluate, or symbolically calculate, B-spline basis functions.

B-splines have many numerical and geometrical properties [Farin 1990], among which exact knot insertion [Prautzsch 1984; Schumaker 1981]. Precisely speaking, we consider a first B-spline representation space, constructed using the known vector $U = (u_0, \cdots, u_n)$: the generated basis functions of degree d, are written here $F^0_{j,d}$. We also consider a second B-spline space constructed using some knot vector $W = (w_0, \cdots, w_{n+k})$ created from U inserting k knots, that is, we suppose that there exists some injection σ, that maps $\{0, \cdots, n\}$ into $\{0, \cdots, n+k\}$, such that:

$$\forall i \in \{0, \cdots, n\}, \left\{ \begin{array}{l} i \leq \sigma(i) \\ u_i = w_{\sigma(i)} \end{array} \right. \tag{2}$$

Basis functions generated from W of degree d are written $F^1_{i,d}$. Theoretical knot insertion properties corresponds to the existence of coefficients $\beta^d_{i,j}$ such that:

$$\forall j \in \{0, \cdots, n-d\}, \forall s, F^0_{j,d}(s) = \sum_{i=0}^{n+k-d} \beta^d_{i,j} F^1_{i,d}(s) \tag{3}$$

The Oslo algorithm defines $\beta^d_{i,j}$ coefficients, using recursive formula [Prautzsch 1984]:

$$\beta^0_{i,j} = \left\{ \begin{array}{ll} 1 & \text{if } u_i \leq w_j < u_{i+1} \\ 0 & \text{otherwise} \end{array} \right.$$
$$\text{and, for } r = 0, \cdots, k-1: \tag{4}$$
$$\beta^{r+1}_{i,j} = \frac{w_{j+r} - u_i}{w_{i+r} - u_i} \beta^r_{i,j} + \frac{u_{i+r+1} - w_{j+r}}{u_{i+r+1} - u_{i+1}} \beta^r_{i+1,j}$$

Such an insertion process will be later in this article used in its global matrix form:

$$F^0 = \beta^T F^1 \tag{5}$$

where F^0 is the vector composed of the functions $F^0_{j,d}$, F^1 the vector composed of the functions $F^1_{i,d}$, and β^T the transpose of matrix β, matrix composed of the coefficients $\beta^d_{i,j}$, evaluated using eq. (3). It is to note that, in order to make notation more readable, indexes representing spline degree d will be removed from notation in the remainder of the article (i.e $b_{i,d}$ becomes b_i, etc...), as the spline degree is always considered as constant. Among other things classical consequences of knot insertion property, a curve, represented by a control point sequence q^0 on the knot vector U, will be represented on the knot vector W using control point sequence $q^1 = \beta q^0$. It is important to understand that knot insertion automatically entails control point insertion, and that the first cannot be achieved without the second.

It is to note that in the cubic case, that is, without any loss of generality about the presented results, the B-spline example case we used, the insertion of a single knot value only involves two non-zero values, i.e. in case of knot at index i, $\beta_{i,i}$ and $\beta_{i,i-1} = 1 - \beta_{i,i}$. In that specific case, the new control points are given by the simple relation:

$$q^1_i = \beta_{i,i} q^0_i + (1 - \beta_{i,i}) q^0_{i-1}$$

With such an insertion property, B-splines functions are said to be *refinable*, which is the basic property for wavelet based multi-resolution availability [Finkelstein and Salesin 1994; Kazinnik and Elber 1997]. As B-spline functions are local, insertion process is guaranteed to be in linear complexity in term of spline degree, and independent of the number of control points. The question to know which points can be removed, or, on the contrary, to calculate parametric values where knot should be inserted, typically involves studying the analytic structure of $C(t)$, in order to isolate maximal and minimal curvature points. Many different tools exist for B-splines in regard of high-curvature point isolation: interval analysis [Snyder 1992; Hart et al. 1998; Berchtold et al. 2000], symbolic root-isolation [Elber and Cohen 1993], wavelet decomposition using semi-orthogonal or biorthogonal B-spline multi-resolution analysis [Kazinnik and Elber 1997]. Interval analysis provides quite an efficient way for fast partitioning of curve into parts of different interests in regard of curvature [Berchtold et al. 2000]. Symbolic root finding is also efficient in spline context: B-spline basis functions are piecewise polynomials, most of the time of accessible degree. Wavelets provide a good theoretical framework for curve analysis. Yet, they are quite computationally expensive, and hardly affordable in a high-performance framework.

From a theoretical point of view, several results and techniques exist for B-spline knot removal [Eck and Hadenfeld 1995; Tiller 1992]. B-spline curve allow for exact knot removal when such operation can be inverted, i.e. shape is not modified and re-insertion of the removed knot provides exactly the original curve configuration. Since the Oslo algorithm moves existing control points, it is natural to also expect modification of neighbour control points during knot removal. [Eck and Hadenfeld 1995; Tiller 1992] provide a general algorithm for B-spline knot removal. We detail here the case we used for our tests, i.e. simple cubic case, that is simple application of these algorithms. The removal process must be the inverse function of the insertion (see eq. (3)). We consider for the explanation a spline defined by a set of control points $q^\star$ and a knot vector $t^\star$. We also consider a refined version of this curve, defined using control point q and knot vector t, that only has one inserted point at abscissa $t_{j+d} + \delta.(t_{j+d+1} - t_{j+d})$, with $0 \leq \delta \leq 1$.

The insertion algorithm gives the following equations:

$$\begin{cases} \forall k < j, \;\; q_k = q_k^\star \\ q_j = \beta_{j,j} q_j^\star + (1 - \beta_{j,j}) q_{j-1}^\star \\ q_{j+1} = \beta_{j+1,j+1} q_{j+1}^\star + (1 - \beta_{j+1,j+1}) q_j^\star \\ q_{j+2} = \beta_{j+2,j+2} q_{j+2}^\star + (1 - \beta_{j+2,j+2}) q_{j+1}^\star \\ \forall k > j+2, \;\; q_k = q_{k-1}^\star \end{cases} \tag{6}$$

During knot removal, q_{j+1} is suppressed and points q_j and q_{j+2} points are respectively changed into points $q_j^\star$ and $q_{j+1}^\star$. Equation (6) easily provides:

$$\begin{cases} q_j^\star = \dfrac{q_j - (1 - \beta_j) q_{j-1}^\star}{\beta_j} \\ q_{j+1}^\star = \dfrac{q_{j+2} - \beta_{j+2} q_{j+2}^\star}{1 - \beta_{j+2}} \end{cases} \tag{7}$$

3.2 Global animation process

The following algorithm presents the general algorithm used to simulate the material B-spline curve.

```
MECHANICAL LOOP()
1  while 1
2    do
3       COLLISIONSDETECTION()
4       COLLISIONSRESPONSE()
5       for (All objects o)
6          do o->COMPUTEACCELERATION()
7       NUMERICALINTEGRATION()
8       UPDATEDATA()
```

Functions about collisions handling (i.e. collisionsDetection() and collisionsResponse()) contain very classical algorithms and are not within the scope of this section, as they will no further be discussed in the remainder of the article. We refer the reader to [van den Bergen 2003] for possible techniques. For numericalIntegration() function, many techniques are also available in literature [Witkin and Baraff 1997]. The main idea of this function is to determine the new positions and velocities of the spline's control points from their acceleration. Each model just has to compute its accelerations and update its data at the end of the mechanical iteration. Only content of computeAcceleration() function is discussed in this section.

To be able to animate the curve realistically, a physical simulation is applied to it, which aim at computing the successive positions of all the control points. To allow multi-resolution, it is desirable to exploit the continuous property inherent to the curve. So we avoid using a mass/spring model, but choose a Lagrangian approach instead. This method only requires the model to define a finite set of degrees of freedom and express its energies as a function of them. Our curve is based on equation (1), thus the degrees of freedom are the control point coordinates. The lagrangian equation of the system can be derived into a linear system [Lenoir et al. November 2002]:

$$\begin{pmatrix} M & 0 & 0 \\ 0 & M & 0 \\ 0 & 0 & M \end{pmatrix} \begin{pmatrix} \mathbf{A^x} \\ \mathbf{A^y} \\ \mathbf{A^z} \end{pmatrix} = \begin{pmatrix} \mathbf{B^x} \\ \mathbf{B^y} \\ \mathbf{B^z} \end{pmatrix} \tag{8}$$

with

$$M_g = \begin{pmatrix} M & 0 & 0 \\ 0 & M & 0 \\ 0 & 0 & M \end{pmatrix} \tag{9}$$

where $\mathbf{A}$ the degrees of freedom's accelerations, $\mathbf{B}$ a vector that sums the different contributions of all forces and internal energies, and finally M_g the generalised mass matrix (resulting from the kinetic energy term) defined via the sub-matrix M of size $n \times n$:

$$\forall (i,j) \in \{1..n\} \times \{1..n\}, \quad M_{ij} = m. \int_{\mathbb{R}} b_i(s).b_j(s)\,ds \tag{10}$$

with m the mass of the spline.

The resulting system size is $3n \times 3n$ where n is the number of control points. For reasonable value of n, the simulation can be done in real time thanks to a LU decomposition of the M matrix [Lenoir et al. November 2002].

All interaction forces are inserted into the system by the way of $\mathbf{B}$. Moreover, the gravity is also taken into account by this vector via the formula: $B_i^\alpha = B_i^\alpha + m.\mathbf{g}^\alpha \int_0^1 b_i(s)ds$ with $\alpha \in \{x,y,z\}$. Also an internal energy of deformation E, which quantifies the amount of energy needed to deform the body, must be added in the vector $\mathbf{B}$. This is done by computing the variation of this energy relatively to each degree of freedom: $B_i^\alpha = B_i^\alpha - \frac{\partial E}{\partial q_i^\alpha}$. In other words, this energy tends to make the model behave in a homogenous way.

This energy can be, for example, defined thanks to springs placed along the curve and giving it some elasticity in stretching or bending. Yet, such a discrete approach is not well suited to a changing resolution stage. A more convenient way is to define a continuous energy, which is based on the shape of the curve and not a finite set of points. This type of energy takes advantage of the natural model's continuity to be independent of the B-spline resolution. On a 1D model, three types of deformation are interesting: stretching, bending and twisting [Terzopoulos et al. july 1987]. As an example, the stretching deformation part is considered in this paper as it provides consistency to the model. The main idea is just to show how the adaptive process interacts with a deformation process.

3.3 Continuous stretching energy definition

Deformation energy is classically expressed by the way of a tensor product between a strain tensor and a stress tensor.

The strain tensor measures the deformation of the body while the stress tensor computes the generated forces.

In the case of linear elasticity, Nocent and Remion [Nocent and Remion 2001] propose the application of the Green-Lagrange strain tensor to the spline curve simulation for high deformation. By developing the equations, we obtain the resulting energy term:

$$E(t) = \frac{Y.r}{8} \int_{begin}^{end} (\gamma(s,t)^2 - 1)^2 . \frac{\partial l_0(s)}{\partial s} ds \tag{11}$$

with t the time, Y the Young modulus of the material, r the area of a curve's section (considered as constant), *begin* and *end* the valid boundaries of the parametric abscissae (for more readability, we suppose that this boundaries are respectively 0 and 1), l_0 the length of the curve in rest state and finally $\gamma(s,t) = \frac{\partial l(s,t)}{\partial s} / \frac{\partial l_0(s)}{\partial s}$ with $l(s,t)$ the current length of the curve (so that $\gamma(s,t)$ express the local dilatation factor in stretching).

To introduce this energy in the system, its first derivative with respect to the each degree of freedom q_i^α has to be computed [Nocent and Remion 2001]:

$$W_i^\alpha(t) = \frac{Y.r}{2} \sum_{m=1}^{n} q_m^\alpha(t) [B_{im} - \sum_{p,q=1}^{n} B_{impq}(q_p.q_q)] \tag{12}$$

where

$$\begin{aligned} B_{im} &= \int_0^1 b_i'(s) b_m'(s) / \frac{\partial l_0(s)}{\partial s} ds \\ B_{impq} &= \int_0^1 b_i'(s) b_m'(s) b_p'(s) b_q'(s) / \frac{\partial l_0(s)}{\partial s}^3 ds \end{aligned}$$

We state that such an energy can be used in a real-time context. Indeed, even if these terms cannot be formally computed, they can however be accurately evaluated since they are constant over time for a given spline configuration. Moreover, the locality of the spline model permits to eliminate some computation in equation (12). All sum terms in the equation are controlled and constant (linked to the locality parameter) boundaries. In consequence, the complexity of this continuous energy computation is $\mathcal{O}(n)$.

4 Adaptive model

Our main idea is to apply the subdivision principles detailed in section (3.1) to the spline animation of section (3.2). In this section, we are interested in how the model is altered when adding or removing a control point and when such operations must be applied. Global adaptive process is first described, and then energy factors update, as well as knot insertion/removal criteria, are discussed.

4.1 Global animation algorithm description

The adaptive aspects are treated in the mechanical process just after the numerical integration stage in order not to disturb the numerical integration which calls the acceleration computation of all objects:

```
ADAPTIVE MECHANICAL LOOP()
1  while 1
2     do
3        ...
4        NUMERICALINTEGRATION()
5        ADAPTATION()
6        UPDATEDATA()
```

the function adaptation() is organised as followed:

```
ADAPTATION()
 1  while mustWeInsertPoint()
 2     do insert knot and control point
 3        modify control points positions and velocities
 4        update matrices and vectors
 5        update energy structure
 6
 7  while mustWeRemovePoint()
 8     do remove knot
 9        modify control points positions and velocities
10        remove control point
11        update matrices and vectors
12        update energy structure
```

After each numerical integration step, adaptive criteria (see section (4.3)) are tested on the curve: curve resolution is changed accordingly, and corresponding terms (especially regarding mass matrix M, and energy expression) are recalculated.

The generalised mass matrix M (cf. equation (10)) is modified as well as its LU decomposition. As B-spline have locality property, most of the coefficient stay unchanged, and matrix M is still a band structure of width $2d+1$. The multiresolution process, for one knot insertion/removal, affects $d+2$ basis functions and for each of them, $2d+1$ basis functions have a non null intersection support with it. Using the fact that M is symmetrical, knot insertion/removal involves $(d+2)(2d+1)$ re-computation of terms within matrix M, and the cost of such an operation is thus independent from the spline curve complexity. Yet, LU decomposition of this matrix needs to be recomputed. We presently do not update the LU decomposition but compute it completely. In practice, it is not a real drawback, since the number of points is rather small and adaptation provides a good means to keep this number small. However, if an application should need a high number of degrees of freedom, an update of the LU decomposition should be proposed. Besides, it is important to understand that the computation of a new mass matrix does not imply a physical change of the model, since it is always related to the same continuous mass distribution along the curve.

The gravitation is also re-computed using relation $B_i^\alpha = m.g^\alpha . \int_0^1 b_i(s) ds$. Again, thanks to the B-spline locality, only the $d+2$ basis functions affected by knot insertion/removal are recomputed by this formula. This update is done once again in $\mathcal{O}(1)$ in regard of number of control points. Deformation energy must be also re-computed. As this term deals with the behavior of the model, we focus on this computation in the next section.

4.2 Adaptation of the stretching energy

The insertion at a specific abscissa affects the basis functions whose supports contain this abscissa. For spline degree equal to d, this affects $d+2$ basis functions.

Changing resolution affects the stretching energy (cf equation (12)) via the terms B_{im} and B_{impq}. In these two terms, the factor $\frac{\partial l_0(s)}{\partial s}$ only depends on the original spline resolution. As a result, only the basis functions first derivative affect the two terms during a resolution changes (either knot insertion, or removal) stage. The two addressed terms are updated slightly differently:

- As the expression of B_{im} is symmetrical in regard of indexes i and m, we only store B_{im} for i and m such that $i \geq m$. For a given value of i, there are $2d+1$ consecutive values of m for which B_{im} is non-null, because of the spline locality property. Using the commutativity property, useful value for m only involves $d+1$ values. And, as there are $d+2$ basis functions affected by the insertion, the total number of elements to be recomputed is $(d+2)(d+1)$. As a result, the update cost of terms B_{im} is independent from spline complexity for a given knot insertion/removal.

- Again, as the expression of B_{impq} is symmetrical in regard of indexes i, m, p and q, we only store B_{impq} for i, m, p and q such that $i \geq m \geq p \geq q$. The recomputed elements on knot insertion/removal are these which have at least one index (between i, m, p and q) matching an affected basis functions's index. Once again, it is easy to see that B-spline locality entails modification cost that is only dependent on the spline degree that is used. In order to minimize coefficients' storage cost, we describe here the indexing technique we used, that stores exactly the needed coefficients. Indeed, B_{impq} coefficients are included in a collection that collects many null values, and heavily symmetrical (all permutations of indexes i, m, p, q provide the same value): storing such coefficients with a n^4 sized array would be wasteful. The choice we made is to use the following value indexing process: we first use, for indexing B_{impq} (with $i \geq m \geq p \geq q$), the integers q, $p-q$, $m-p$, $i-m$. This, because of spline locality, provides, for non-null B_{impq} values, a set of four integers that belong to $\{0,\cdots,n\} \times \{0,\cdots,d\}^3$, and that can quite easily be enumerated: as a result, coefficients are stored in an array, which size is linear in regard of the number of control points.

Since our continuous stretching energy only depends on the shape of the curve and since the insertion or removal of a point does not imply any shape alteration (see section (3.1)), we are assured that deformations are continuous relatively to the time.

4.3 Adaptation criteria

There are two different cases when point should be inserted on the spline curve during interactive manipulation:

- The first case is purely geometric. High curvature points on B-splines curve naturally involve mechanical limitation due to spline resolution. As a result, an insertion is needed. [Grisoni and Marchal 2003] presents a curvature evaluator adapted to high-performance, that we need here. The i-th spline segment is assigned the following value:

$$c_i = \frac{1}{2}\left(\frac{\overrightarrow{q_{i-1}q_i}.\overrightarrow{q_iq_{i+1}}}{\|\overrightarrow{q_{i-1}q_i}\|\|\overrightarrow{q_iq_{i+1}}\|}+1\right) \quad (13)$$

  Such an expression practically appears to provide fast criterion for B-spline segment maximal curvature evaluation: each spline segment connection can be, when needed, associated with c_i that measures the local control point *disturbance*. The curvature criterion of equation (13) is used for isolating high-curvature segments: a knot value (hence, a control point on the spline curve) is inserted at the middle of the i-th segment if and only if c_i and c_{i+1} are greater than some pre-set threshold value.

- The second case is more mechanically involved. In practice, for a spline curve with a constant, coarse, resolution, it is not possible to plausibly replicate action one would have on real physical model, simply because of resolution limitations. For example, acting on an elastic model at any point would intuitively entail local deformation. Such a deformation on a coarse resolution curve is not possible to simulate, because local degrees of freedom density is not high enough. As a result, we chose to insert some point each time user interacts directly on the spline curve, the new knot value being inserted at the parameter value where the action is located.

The removal of a control point is more simple. In practice, the curvature criterion defined in equation (13) is once again used, but this time in order to detect low-curvature segment. A point is removed when local curvature is low enough. Precisely speaking, control point q_i is removed when c_i is below some pre-set threshold.

It can be pointed out that the insertion process can occur anywhere without any condition. But the removal process has to verify a geometric condition to be an exact algorithm[Eck and Hadenfeld 1995]. We approximate this condition by the low curvature criterion. By this way, we are able to tune the exactness of the removal process. For an exact process, points are removed only when the curve is straight, which happens essentially at an equilibrium state of the simulation.

5 Applications

All the presented tests have been performed on a PIV 2.4Ghz 512MB using fast implicit Euler integration scheme and a virtual time step fixed to 1ms. Times announced in the different sub-sections are the mechanical computation times and thus only take into account the mechanical computation (including the collision process and the numerical integration stage), independently from visualisation time.

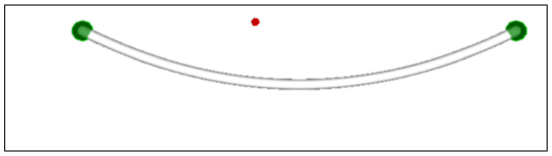
5.1 Application: B-spline curve cutting

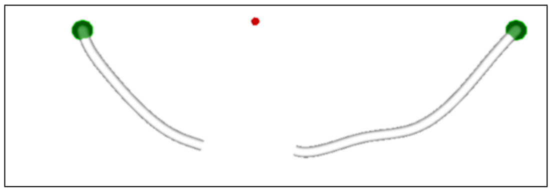
It is known that B-spline continuity at a given knot value equals the difference between polynomial degree and knot multiplicity. This explains, among other things, that cubic B-spline is the B-spline of lowest degree that allow C^2 continuity. It also has as a consequence that creating knot multiplicity equal to $d+1$ for a spline of polynomial degree d creates C^{-1} discontinuity, which means that the spline curve, even when defined as a whole, will practically exist as two separate pieces. As shown here, this property directly extends to adaptive physical B-spline simulation. Knot insertion is used to simulate curve cutting.

This operation is presented by figure (2) where an initial spline of 12 control points is left hanging from its extremities (figure (2(a))). Figure ((2(b))) shows the consequence, on the simulation, of multiple knots insertion at arbitrary parametric abscissa. Figure (2(c)) shows mechanical independence of the two created pieces, a manipulation on the first piece, does not affect the simulation of the second one. This simulation takes about 4ms for each simulation step at the beginning of the simulation (12 control points and 6 constraints to fix the two extremities). At the end, the simulation step takes about 8ms with 16 control points and still 6 constraints.

Figure (3) points out the possibility to create multiple independent pieces. A curve is simulated with 10 control points and 3 different cuttings are applied during the simulation. The computation time begins at 1ms and finish around 20-30ms depending on collisions (for 22 control points). Each piece interacts independently as shown in figure (3(d)).

We stress that this cutting procedure is only a direct application of the adaptive animation technique, and involves no other special treatment (such as re-meshing) unlike other cutting techniques. This simple example illustrates that the use of adaptive physically-based splines can be a powerful tool.


(a) Initial configuration


(b) Arbitrary cutting

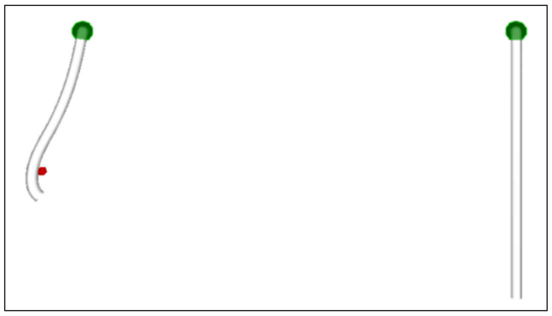

(c) Interaction on one piece

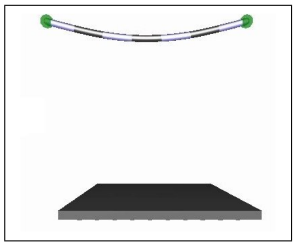
Figure 2: Example of curve cutting using mechanical B-spline multi-resolution.

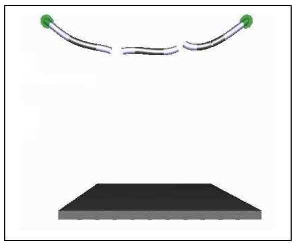
5.2 Application: knot tying

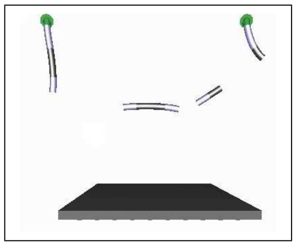
The second result demonstrates efficiency of adaptive B-spline physical simulation on the classical case of knot tying. The initial spline configuration is composed with a preformed knot and the two extremities are fixed. The spline is then animated using simple gravity, in order to let the knot tie under simple action of curve weight. Figure (4(a)) shows the result of such animation with a fixed resolution and figure (4(b)) shows the result with an adaptive resolution. Figure (4(c)) zoom in the knot to be able to see the different insertions in the knot area. On each left figure, spline segments color have been alternatively changed, so that knot repartition (and implicitly knot insertion/removal) could be visible on such pictures. On each right figure, control points are shown.

On figure (4(a)), one can easily see uniform sampling limits. The curve encounters a barrier due to the lack of control implied by the limited number of degrees of freedom. This spline requires 2.27ms computing time per mechanical iteration. For the knot to be tightened, one would need a much higher number of regularly distributed control points, in order to get a sufficient control wherever the knot is tightened on the curve. In practice, this leads to prohibitive computation time for real-time purposes (several seconds or even minutes per simulation step).

On figures (4(b)) and (4(c)), the same simulation is done


(a) Initial configuration


(b) Arbitrary cuttings


(c) 4 independent pieces

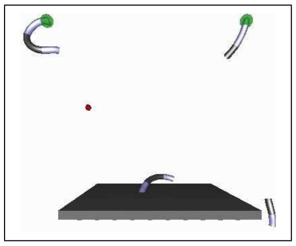

(d) Evolution of each pieces

Figure 3: Example of curve cutting in multiple piece.

using an adaptive resolution. One can see that both knot removal (see low curvature areas of the curve) and knot insertion have been used for providing the shown result. The non adaptive spline involves 11 control points, and the adaptive one involves, at the end of the animation, 16 points. A mechanical iteration takes about 9.89ms.

The modifications, involving by the adaptive process, affect the B-spline parametrization by the way of its knot vector. At the beginning of the simulation, this vector is:

$$T = \begin{array}{cccccccccc} \{ & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ & 10 & 11 & 12 & 13 & 14 & \} \end{array}$$

During the simulation, 9 control points are inserted (at the parametric abscissae 8.5, 5.5, 6.5, 7.5, 6.75, 7.75, 6.875, 7.25 and 7.625) followed by 4 control points removed (at the parametric abscissae 4, 5.5, 9 and 10). The removal process usually occurs when the simulation is stable and some spline segments are quite aligned. These adaptive modifications result in the following knot vector at the equilibrium state:

$$T = \begin{array}{ccccccccc} \{ & 0 & 1 & 2 & 3 & 5 & 6 & 6.5 & 6.75 \\ & 6.875 & 7 & 7.25 & 7.5 & 7.625 & 7.75 & 8 & 8.5 \\ & 11 & 12 & 13 & 14 & \} \end{array}$$

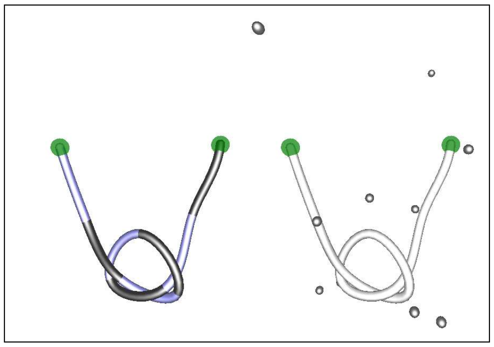
The spline knot vector shows the locality of the process. It presents large parametric segments on low curvature locations (before and after the knot) and smaller one on high curvature case (in the knot area).

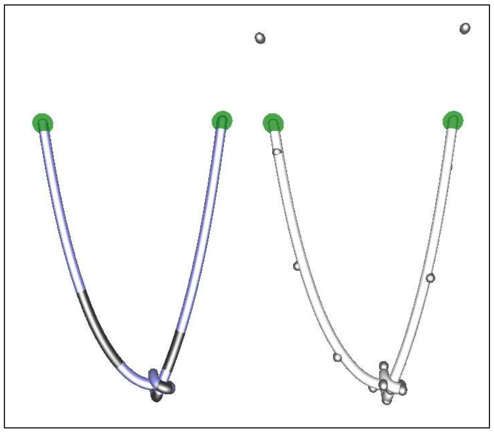
6 Conclusion and future work

We have presented in this paper an adaptive model of B-spline physical simulation with continuous deformation energy allowing a continuity in physical movement during the resolution changing. This method refines the model locally around any arbitrary point on the curve and allows a finer geometric adaptation, by the way of a higher number of degrees of freedom. This model presents interesting results for some applications like simulation of curve cutting (for example during a surgical simulation) or knot tying.

This work could be improved by studying new terms like bending and twisting (continuous) energies permitting to treat specific curves as surgical threads. Another related work is the treatment of self collision by robust methods as Lagrange multipliers instead of penalty methods thanks to dedicated constraints.

It is also interesting to point the fact that, in order to provide efficient adaptive physical simulation, both geometric and mechanical aspects need to be taken into account. However, the link between geometric adaptation and the need for a mechanical refinement is obviously less simple than one could expect at first, and would also deserve closer study.


(a) Knot at fixed resolution


(b) Knot with adaptive resolution

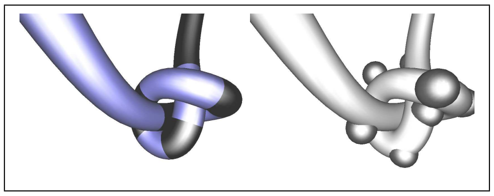

(c) Zoom in the knot with adaptive resolution

Figure 4: Example of knot tying.

References

ASTLEY, O., AND HAYWARD, V. 1997. Real-time finite elements simulation of general visco-elastic materials for haptic presentation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

BERCHTOLD, J., VOICULESCU, I., AND BOWYER, A. 2000. Interval arithmetic applied to multivariate bernstein-form polynomials. Tech. Report 31/98, School of Mechanical Engineering, Univ. of Bath, October.

BROWN, J., LATOMBE, J., AND MONTGOMERY, K. 2004. Real-time knot-tying simulation. *The Visual Computer 20*, 2-3 (May), 165–179.

CAPELL, S., GREEN, S., CURLESS, B., DUCHAMP, T., AND POPOVIC, Z. 2002. A multiresolution framework for dynamic deformations. In *ACM Symposium on Computer Animation*.

COX, M. G. 1972. The Numerical Evaluation pf B-Splines. In *J. Inst. Mathematics and Applications*, vol. 10, 134–149.

DE BOOR, C. 1972. On Calculating with B-Splines. *J. Approximation Theory 6*, 1 (July), 50–62.

DEBUNNE, G., DESBRUN, M., CANI, M., AND BARR, A. 2001. Dynamic real-time deformations using space and time adaptive sampling. In *SIGGRAPH'01 Conference Proceedings, Computer Graphics annual conference series*.

ECK, M., AND HADENFELD, J. 1995. Knot removal for B-spline curves. *Computer Aided Geometric Design 12*, 3, 259–282.

ELBER, G., AND COHEN, E. 1993. Second order surface analysis using hybrid symbolic and numeric operators. *ACM Transaction on Graphics 12*, 2 (April), 160–178.

ETZMUSS, O., EBERHARDT, B., AND HAUTH, M. 2000. Collision adaptive particle systems. In *Pacific Graphics'2000 Conference*.

FARIN, G. 1990. *Curves and Surfaces for Computer Aided Geometric Design*, inc. second ed. Academic Press.

FINKELSTEIN, A., AND SALESIN, D. H. 1994. Multiresolution curves. *Computer Graphics 28*, Annual Conference Series, 261–268.

GANOVELLI, F., CIGNONI, P., AND SCOPIGNO, R. 1999. Introducing multiresolution representation in deformable object modeling. In *Spring Conference on Computer Graphics*, 149–158.

GRINSPUN, E., KRYSL, P., AND SCHRÖDER, P. 2002. Charms: a simple framework for adaptive simulation. In *SIGGRAPH'02 Conference Proceedings, Computer Graphics annual conference series*, 281–290.

GRISONI, L., AND MARCHAL, D. 2003. High Performance Generalized Cylinder Visualization. In *Proc. Shape Modeling International '03*, IEEE Computer Society Press, 257–263.

HART, J. C., DURR, A., AND HARSH, D. 1998. Critical points of polynomial metaballs. *Implicit Surface'98 Proc.*, 69–76.

HUTCHINSON, D., PRESTON, M., AND HEWITT, T. 1996. Adaptive refinement for mass/spring simulations. In *EUROGRAPHICS Workshop on Animation and Simulation*, 31–45.

KAZINNIK, R., AND ELBER, G. 1997. Orthogonal decomposition of non-uniform bspline spaces using wavelets. *Eurographics'97 Proc.* (August), 27–38.

LENOIR, J., MESEURE, P., GRISONI, L., AND CHAILLOU, C. November 2002. Surgical thread simulation. *Modelling and Simulation for Computer-aided Medecine and Surgery*.

LUCIANI, A., HABIBI, A., AND MANZOTTI, E. 1995. A multiscale model of granular materials. In *Graphics Interface'95 Conference*, 136–146.

NOCENT, O., AND REMION, Y. 2001. Continuous deformation energy for dynamic material splines subject to finite displacements. *Eurographics CAS'2001*.

NOCENT, O., NOURRIT, J., AND RÉMION, Y. 2001. Toward mechanical level of detail for knitwear simulation. In *WSCG'01 Conference*, 252–259.

PHILLIPS, J., LADD, A., AND KAVRAKI, L. 2002. Simulated knot tying. In *International Conference on Robotics and Automation*, 841–846.

PRAUTZSCH, H. 1984. A short proof of the oslo algorithm. *Computer Aided Geometric Design 1*, 95–96.

SCHUMAKER, L. L. 1981. *Spline Functions: Basic Theory*. John Willey & Sons, New York.

SNYDER, J. 1992. Interval analysis for computer graphics. *Computer Graphics (SIGGRAPH Proc.) 26*, 2 (July), 121–130.

TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. july 1987. Elastically deformable models. *Computer Graphics*.

TILLER, W. 1992. Knot-removal algorithms for nurbs curves and surfaces. *CAD 24*, 8, 445–453.

VAN DEN BERGEN, G. 2003. *Collision Detection*. Morgan Kaufmann, November. ISBN:155860801X.

WANG, F., BURDET, E., DHANIK, A., POSTON, T., AND TEO, C. 2005. Dynamic thread for real-time knot-tying. In *World Haptic Conference*, 507–508.

WITKIN, A., AND BARAFF, D. 1997. Physically based modeling: Principles and practice. In *Siggraph '97 Course notes*.

WU, X., DOWNES, M., GOKETIN, T., AND TENDICK, F. 2001. Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes. *Computer Graphics Forum 20*, 3 (Sept.), 349–358.