Quasi-Dynamic Splines

Adrien Theetten [*] INRIA-Futurs
Laurent Grisoni [†] LIFL
Christian Duriez [‡] INRIA-Futurs
Xavier Merlhiot [§] CEA LIST

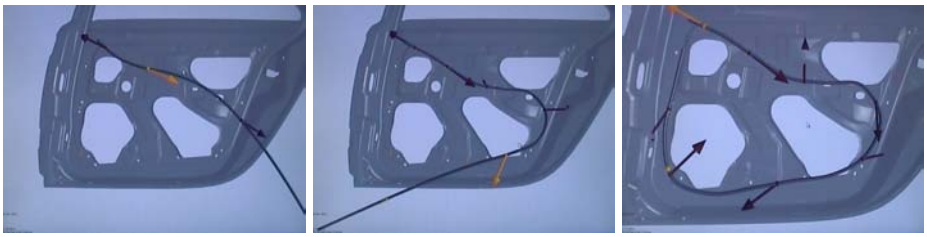

Figure 1: Several steps of cable positioning on a cardoor. This quasi-dynamic simulation runs at real-time.

Abstract

In this work, we propose interactive and physically based animation of one-dimensional deformable models using geometrically exact energy formulation. The proposed mechanical model has a high level of accuracy: it is based on continuous spline support and continuum mechanics media equations. We also detail a new efficient solving scheme, that can automatically switch between dynamic and static during simulation. With this scheme, we want to raise inconsistencies that could show up when human user is interacting with a physical simulation. We finally present a practical example in which the proposed model provides high-quality interaction.

CR Categories: I.6 [Simulation and Modeling]— [J.6]: Computer-Aided Design

Keywords: dynamic and quasi-static splines, real-time simulation

1 Introduction

Interacting with virtual deformable objects always need compromise between accuracy and efficiency. In most applications, simulation of one-dimensional objects necessitates both of these properties: numerical coarseness often leads to unrealistic geometrical configurations, while excessive computational cost drastically diminishes the interaction quality, as well as physical behavior perception. In many industrial cases, numerous applications need such accurate, and fast simulated object: e.g., virtual cable positioning (in car R&D industry), surgical threads for interactive medical simulation. Dynamic splines were introduced by Qin and Terzopoulos [Qin and Terzopoulos 1996]. They proposed for such simulations a formal framework where it is actually possible to provide *geometrically exact* (hence non-linear) formulation of energies making accurate simulation possible [Theetten et al. 2006].

For most known simulation methods of deformable objects, one has to choose between *quasi-static* and *dynamic* models before simulation. Most of previous works give no or very simple arguments for this choice, this is actually not easy to do in the general case. Static models are classically considered as being faster to solve than dynamic models. Yet, in some cases, dynamic simulation, using explicit integration can be of the same computation cost. On the other hand, dynamic models are supposed to be more accurate because they provide dynamic transitions between rest states. These transitions can be false if the simulation timestep is not adapted to the propagation of deformation wave (although such error is most often difficult to perceive with interactive simulations). That is the reason why we propose a new scheme that can switch between static and dynamic on-line. Decision/adaptation is autonomous and the rules are based on two consistency criterions: the non-singularity of the mechanical system and the temporal coherency between simulation and interaction. The decision rules provide an estimation of the need for dynamic simulation. The temporal consistency consideration is close to the *real-time* assumption (i.e. computation cost for one simulation step remains lower than the timestep for dyncamic simulation). Yet, we intend our model to be used on very general frameworks, not necessarily based on real-time operating systems. In that context, guaranteeing rigorous real-time dynamic simulation is so far pure illusion. When not suitable, or too expensive for the considered time step, quasi-static simulation is used as an alternative.

The proposed model is hence able to automatically adapt to the considered simulation context, and provides user with the feeling that simulation is always dynamic. Because of this flexibility, we call the resulting model *Quasi-Dynamic Splines*. In the present article, we identify the following scientific contributions: first, a high performance and real-time version of the *Geometrically Exact Dynamic Spline* (GEDS) model, introduced in [Theetten et al. 2006]. The presented algorithm complexity is a linear function depending on the control point amount. It is based on a linearly implicit integration scheme where the stiffness matrix is *analytically* computed. Second, an efficient solving scheme, that can *automatically switch between dynamic and quasi-static equilibrium* and guarantees, to some extent, real-time simulation. This implies two points: to handle the practical switch and to provide a satisfactory set of deterministic rules that prevent simulation from excessive numerical inertia in the decision process.

[*]e-mail: adrien.theetten@lifl.fr
[†]e-mail:laurent.grisoni@lifl.fr
[‡]e-mail:christian.duriez@inria.fr
[§]e-mail:xavier.merlhiot@cea.fr

SPM 2007, Beijing, China, June 04 – 06, 2007.
© 2007 ACM 978-1-59593-666-0/07/0006 $5.00

The remainder of the paper is organized as follows: next section describes recent related works on dynamic splines and real-time simulation. Section 3 provides a short overview of the geometrically exact dynamic spline, that we use as a basis for the main result of this article. Section 4 describes a solving scheme that is convenient to static and dynamic simulation. Finally, section 5 presents a dynamic switch between dynamic and quasi-static simulations, that preserves mechanical accuracy, and significantly improves computation efficiency.

2 Previous works

For most unconstrained Computer Aided Design applications, splines are probably the most classical tool for 1D objects. Dynamic splines were introduced by Qin and Terzopoulos [Qin and Terzopoulos 1996]. Some other one-dimensional models exist, from most computationally efficient to most numerically accurate. Cosserat models [Pai 2002],[Wakamatsu and Hirai 2004],[Grégoire and Schömer 2006] and Super-Helices [Bertails et al. 2006]. The reader can refer to [Theetten et al. 2006] for a detailed discussion on wire models.

As far as we are concerned, the dynamic spline model still remains the best choice for efficient and accurate simulation. The model of Qin and Terzopoulos [Qin and Terzopoulos 1996], Dynamic Non-Uniform Rational B-Splines (DNURBS), first combined spline representation with physics laws, using a lagrangian form of newton equation. Nocent and Rémion [Nocent and Rémion 2001] defined the *Dynamic Material Splines* (DMS), a full Lagrange-based simulation framework for splines. They considered spline control points as the degrees of freedom of the underlying continuous object. Lenoir [Lenoir et al. 2004b] introduced a curvature energy formulation for DMS that was not geometrically exact, but providing real-time manipulations, as well as adaptive simulations[Lenoir et al. 2005]. Using a background in mechanics consisting of elasticity and plasticity theories, Theetten & *al* proposed a deformable model for one-dimensional objects: *Geometrically Exact Dynamic Splines* (GEDS) [Theetten et al. 2006]. This approach addresses reversible and irreversible deformations, like stretching, twisting and bending, and can even detect fracture. This model provides both accurate mechanical simulation as well as quick calculation. However, using the forces with an adaptive Implicit Euler scheme still lacks of efficiency to provide real-time for all geometrical and material configurations, even if remains most of the time interactive.

Baraff used a linearly Implicit Euler integration to simulate cloth with large steps [Baraff and Witkin 1998], involving forces and force derivatives. We adopt the same method but we rely here on an energy formulation that encompasses all aspects of the one-dimensional object: stretching, twisting and bending.

Dynamic effects are not always required to provide a realistic simulation. Inertia and viscoelasticy can be neglected to obtain a static system, when the frequency of excitation is lower than roughly one-third of the structure's natural frequency. This assumption is very conventional, quasi-static simulations are much more convenient and cheap to do. Not only there are very few dynamic models of one-dimensional elements except for spline based-on simulation, but also there is no contribution, to our knowledge, on a dynamic scheduler that can switch between quasi-static and dynamic simulation. However, in the real-time community, Cortes & *al* [Cortes et al. 2005] presented a quasi-static approach where a number of schedules and switching points are prepared at design-time, so that at run-time the quasi-static scheduler only has to select, depending on the actual execution times, one of the precomputed schedules. Moreover, Redon [Redon et al. 2005] & *al* proposed an adaptive algorithm for computing forward dynamics of articulated bodies, which can automatically simplify the dynamics of a multi-body system, based on the desired number of degrees of freedom and the location of external forces and active joint forces. The simplification allows them to achieve up to two orders of magnitude performance improvement. This is a piece of evidence that adaptive solving methods of simulation can really improve performance.

3 Geometrically Exact Dynamic Splines

In this section, we present the core elements of GEDS model: the Lagrange equations of kinetic and potential energies and the equivalent matrix system in both static and dynamic cases.

3.1 GEDS formulation

A one-dimensional element is entirely described by a field that can be decomposed in two fields: a position field $\mathbf{r} = (x,y,z)$, which determinates neutral fiber f position, and a rotation field θ, which provides the roll of the cross-section. These fields are described by a set of polynomial spline curves: $\mathbf{q} = (\mathbf{r},\theta) = (x,y,z,\theta)$. Each resulting spline is given by $\mathbf{q}(u) = \sum_{i=1}^{n} b_i(u)\mathbf{q_i}$, where b_i are the i^{th} spline basis functions of the control points $\mathbf{q_i}$, and u is between 0 and ℓ, the length of the neutral fiber. Arc length is denoted by s. The derivative of control point $\mathbf{q}$, position $\mathbf{r}$ and roll θ with respect to u are denoted by $\mathbf{q}'$,$\mathbf{r}'$ and θ' respectively. The displacement elements ds and du are interrelated by $ds = \|r'\|du$. We also assume that the one-dimensional cross-section is constant along the spline. Since control points completely define the position of the spline and the orientation of the cross-sections, they can be considered as the degrees of freedom of the system and used in the Lagrange equations (1).

The *Lagrange equations* involve the *kinetic energy* T and the *potential energy* U of the system. The kinetic energy is the energy of motion whereas the potential energy is the stored energy of position possessed by an object. F is the sum of external forces. Assuming the mass distribution to be homogeneously distributed between the n degrees of freedom $\mathbf{q_i}$ of the object,

$$\forall i \in \{1,...,n\}, \frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\mathbf{q}}_\mathbf{i}}\right) = \mathbf{F}_i - \frac{\partial U}{\partial \mathbf{q_i}} \qquad (1)$$

Several parameters and relations characterize a material; all the involved quantities are described below to solve the Lagrange equations.

Since the one-dimensional object is specified by position and rotation, its kinetic energy T comprises translation energy and rotational energy. Translation energy corresponds to the displacement of control points and rotation energy is due to the motion of cross-sections around the neutral axis. We define the inertia matrix, denoted $\mathbb{J}$, which is the same everywhere along the spline, since diameter is constant. *Spline kinetic energy* T is then formulated by the following integration along the beam:

$$T = \frac{1}{2}\int_0^L \frac{d\mathbf{q}}{dt}^t \mathbb{J}\frac{d\mathbf{q}}{dt}ds, \;\; \mathbb{J} = \begin{pmatrix} \mu & 0 & 0 & 0 \\ 0 & \mu & 0 & 0 \\ 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & I_o \end{pmatrix} \qquad (2)$$

μ corresponds to linear density, I_o to the polar momentum of inertia and ^{t} denotes a transpose.

Potential energy U is composed of gravitation energy and strain energy and refers to some elasticity and beam theory. We define

a strain vector ε, which is composed of three scalar components: stretching strain ε_s, twisting strain ε_t and bending strain ε_b. From these considerations, we provide a Hooke matrix $\mathbb{H}$, which derives from Hooke's law. Assuming cross-section is circular and its diameter D constant, *Spline strain energy* U thus yields:

$$U = \frac{1}{2}\int_0^L \varepsilon^t \mathbb{H}\varepsilon ds,\ \ \mathbb{H} = \frac{D^2\pi}{4}\begin{pmatrix} E & 0 & 0 \\ 0 & \frac{GD^2}{8} & 0 \\ 0 & 0 & \frac{ED^2}{16} \end{pmatrix} \qquad (3)$$

where E is the *Young modulus* and G is the *shear modulus* For more details, please see [Courbon 1980]).

3.2 Dynamic and quasi-static systems

This subsection explains how, from the above equations, to evaluate practically the linear systems to be solved in both quasi-static and dynamic cases. We consider here kinetic and potential energies (eq. 2, 3)with the Lagrange equations (eq. 1). Replacing $\mathbf{q}$ by the expression given in equation 3.1 in a similar way as described by Nocent and Rémion [Nocent and Rémion 2001], yields: $\frac{d}{dt}\frac{\partial T}{\partial \dot{\mathbf{q}}_i} = \sum_{j=1}^n \mathbb{J}\int_0^L (b_i(s)b_j(s))ds\frac{d^2\mathbf{q}_j}{dt^2}$. Considering $\mathbb{J}\int_0^L (b_i(s)b_j(s))ds$ and $\frac{d^2\mathbf{q}_j}{dt^2}$ as matrices $\mathbb{M}$ and vector A components $\mathbb{M}_{i,j}$ and $\mathbf{A}_j$ respectively, this equation yields: $\frac{d}{dt}\frac{\partial T}{\partial \dot{\mathbf{q}}_\mathbf{i}} = \sum_{j=1}^n \mathbb{M}_{i,j}\mathbf{A}_j$. Considering all degrees of freedom, this sum or Kinetic term can consequently be written as a matrix-vector product: $\mathbb{M}\mathbf{A}$.

The derivatives of strain energies with respect to generalized coordinates compose the right term of the Lagrange equations 1: $P^i = -\frac{\partial U}{\partial \mathbf{q}_i} = -\frac{1}{2}\int_0^L \frac{\partial \varepsilon^t \mathbb{H}\varepsilon}{\partial \mathbf{q}_i} ds$; they are homogeneous to three generalized forces : stretching force P_s, twisting force P_t and bending force P_b. Their calculation is detailled in appendix A. Considering all degrees of freedom, the right term can consequently be written as a sum of external force vector and generalized potential force vector: $\mathbf{F}+\mathbf{P}$.

The Lagrange equations (eq. 1) result in: $\mathbb{M}\mathbf{A} = \mathbf{F}+\mathbf{P}$. This system is to be solved when dynamic simulation is needed. Static equilibrium does not involve kinetic energy. In the later case, just consider the right term of the system: $0 = \mathbf{F}+\mathbf{P}$.

3.3 Accurate dynamic splines

In this subsection, we provide a geometrically exact formulation of stretching, twisting and bending strains ε_s, ε_t and ε_b, in large rotations (or large displacements). Such terms allow accurate results. We lay stress on the deformation model of our object: it is materially linear elastic (small strains) but geometrically exact (large rotations). The stretching strain ε_s is defined by $\varepsilon_s = 1 - \|\mathbf{r}'\|$. The twisting strain ε_t comprises two scalar parts: *geometrical or Frenet twisting* τ and *roll* θ. It yields: $\varepsilon_t = \theta' + \tau = \theta' + \frac{\mathbf{r}'\times\mathbf{r}''\cdot\mathbf{r}'''}{\|\mathbf{r}'\times\mathbf{r}''\|^2}$. The bending strain ε_b is equal to the scalar Frenet curvature k: $\varepsilon_b = k = \frac{\|\mathbf{r}'\times\mathbf{r}''\|}{\|\mathbf{r}'\|^3}$. Accuracy of GEDS is well demonstrated in [Theetten et al. 2006]. In this paper, the focus is laid on efficiency.

4 Making GEDS efficient

The model described in the previous section is mechanically accurate. However, it suffers, in regard of high-level interaction quality, from the main drawback that the global numerical solving algorithm is not efficient enough. An adaptive/iterative implicit scheme does not converge quickly due to the cost of the equation non-linearities, whereas an explicit scheme lacks of stability especially for high stiffness. The compromise of a linearly implicit Euler scheme has proved to be an efficient and fast convergent numerical method to simulate deformable objects like cloths [Baraff and Witkin 1998], while keeping the benefits of non-linearities. It consists in derivating the forces with respect to the degrees of freedom to linearly approximate the force of the following time step, as a curve is approximated localy by its tangent. This method is accurate enough to solve a large time step in only one iteration.

4.1 Linearization of the generalized force P

Considering t the current time step and $t+\Delta t$ the following time step, the sampled Lagrange equations give the following expression: $\mathbb{M}\ddot{\mathbf{q}}(t+\Delta t) = \mathbf{F}+\mathbf{P}(\mathbf{q}(t+\Delta t))$. This non-linear system is computationally too expensive to be solved in an iterative scheme (Newton-Raphson or conjugate gradient method) in a garanteed real-time simulation context. We propose to use a linearized expression of the generalized force $P(\mathbf{q}(t+\Delta t))$. Denoting $\mathbb{K}(\mathbf{q}(t))$ the hessian matrix of the system, the system now yields:

$$\mathbb{M}\ddot{\mathbf{q}}(t+\Delta t) + \mathbb{K}(\mathbf{q}(t))\Delta\mathbf{q} = \mathbf{F}+\mathbf{P}(\mathbf{q}(t)) \qquad (4)$$

where $\Delta\mathbf{q} = \mathbf{q}(t+\Delta t) - \mathbf{q}(t)$.

4.2 Linearly Implicit Euler integration scheme

To solve the resulting system, we use a classical Implicit Euler integration scheme; this results in a linearly Implicit Euler integration:

$$\left(\frac{\mathbb{M}}{\Delta t^2} + \mathbb{K}(\mathbf{q}(t))\right)\Delta\mathbf{q}(t) = \frac{\mathbb{M}}{\Delta t}\mathbf{V}(t) + \mathbf{F} + \mathbf{P}(\mathbf{q}(t)) \qquad (5)$$

where $\mathbf{V}(t) = \dot{\mathbf{q}}(t)$ is the velocity vector of the spline. The corresponding static system yields:

$$\mathbb{K}\Delta\mathbf{q} = \mathbf{F}+\mathbf{P} \qquad (6)$$

The generalized force $\mathbf{P}$ and the stiffness matrix $\mathbb{K}$ expressions are detailled in appendix A. Note that if Δt tends to infinity, then equation 5 tends to equation 6, that is, the dynamic system tends to a static system. In our practical test, and without loss of generality, we use 3^{th} order splines, like Catmull-Rom splines. Basis functions involve 4 parametric segments.

4.3 Constrained dynamic splines

The spline-based model is continuous, that is, mechanically defined everywhere along the one-dimensional object. A force may consequently be applied everywhere, but interacting with the manipulated object remains quite difficult. This is the reason why we use Lagrangian multipliers: they allow us to set the position or the direction of any point of the one-dimensional object. For further information on how to constrain a spline with Lagrangian mulitpliers, see [Lenoir et al. 2004a].

4.4 Solving the resulting system

The matrix $\mathbb{K}$ is positive definite banded and symmetric. Unfortunately, the Lagrange multipliers turn the positive definite system

into an indefinite system. So we solve the system with a LU decomposition for banded matrix. Moreover, the step computation time cost is constant for both dynamic and quasi-static systems. Note that x, y, z and θ generalized force components are linked and therefore can not be solved separatly, contrary to [Lenoir et al. 2004a]. However, an assemblage of splines connected in various locations will obviously grow the bandwidth of the stiffness matrix, the worst case being connecting the two ends of the spline. The solution time may then quickly start to dominate the simulation. A sparse matrix solving algorithm should then be used. To further reduce the error and improve stability, we may use a Newton's Steepest Descent Method [Evtushenko et al.]. So far, we simply use a displacement step ceiling. This ceiling avoid large displacements that would make the model leaves its linearization domain.

4.5 Overall algorithm

For the sake of clarity, algorithm 1 provides a global overview of the simulation algorithm of a GEDS, excluding the following described quasi-dynamic decision scheme.

Algorithm 1 Physically-based spline algorithm

```
Initialization
while simulation do
  for all spline samples do
    Compute strain forces:P_s, P_t, P_b
    Add external forces and insert Lagrange multipliers: F, L
    Compute stiffness matrix K
    if Dynamic System then
      Add control point velocities M V/Δt and inertia M/Δt^2
    end if
    LU solving for band matrix: get Δq_i, update q_i
  end for
end while
```

5 Quasi-dynamic consistent simulation

The aim of this paper is to provide precise and interactive mechanical simulation. However, a compromise between accuracy and efficiency has most of the time to be balanced. A way of increasing efficiency is to simplify physics, but we are looking for accurate results. Another way is to consider static models instead of dynamic ones. Static models are classically considered as being faster to solve than dynamic models. On the other hand, dynamic models are supposed to be more accurate because they provide dynamic transitions between rest states and they are even necessary to describe some physical behavior like oscillations. In this paper, we propose to assemble accuracy and efficiency when possible during simulation, by switching the simulation from dynamic to quasi-static or from quasi-static to dynamic. In this section, we present our method, and we detail two different heuristics to engage the switch.

5.1 Dynamic-static switch

From the matrix systems point of view, switching from static equations (6) to dynamic equations ((5) consists in adding the mass contribution $\frac{\mathbb{M}}{\Delta t^2}$ on matrix and the inertia contribution $\frac{\mathbb{M}}{\Delta t}\mathbf{V}(t)$ on the operand.

When the switch is performed from static to dynamic, the initial conditions (position and velocity) must be given. It means that, even during the static simulation, a velocity is evaluated. This evaluation is performed using the computation time t_c between two successive equilibrium states: $\mathbf{V} = \Delta\mathbf{q}/t_c$.

In the static case, the matrix may be singular: the stiffness matrix $\mathbb{K}$, on his own, is singular but the lagrangian constraints $\mathbb{L}$ can help the definition. On the opposite, the mass matrix $\mathbb{M}$ is always defined. Then, the first heuristic for the decision of a switch is based on the system singularity.

The use of dynamic differential equation and their integration introduce the notions of time step Δt. In interactive simulations, for the temporal consistency we need to make the computation time between two steps as close as possible to Δt. Moreover, this time step also depends on the mechanical eigenfrequency of the structure, and is related to the length of the element by the deformation propagation. Thus, the second heuristic is based on the ability for the dynamic simulation to respect temporal consistency.

5.2 Non singularity as heuristic rule

Let's consider the unconstrained static equation 6. From the consideration of overall equilibrium of the structure, the system contains six dependent equations corresponding to the six rigid-body degrees of freedom. This dependence will render the matrix $\mathbb{K}$ singular. Our first heuristic considers that it is not possible to perform a static computation unless rigid-body degrees of freedom are constraint by Lagrange multipliers which removes the singularity. A singularity test on the static system matrix is quite expensive, so we propose a simplified heuristic based on the Degrees of Freedom (DOFs) constrained by Lagrange multipliers. We consider that rigid-body motion is removed when:

- Four DOFs of a point (x, y, z, θ) and its local tangent are constraints ($= 6$ DOFs)
- Four DOFs of two different points of the spline, at least, are constraints ($= 2 \times 4$ DOFs)

5.3 Temporal consistency as heuristic rule

To integrate differential equations from mechanical dynamics, we need a numerical integration scheme and a time step. The choice of this time step is not easy. Indeed, by its mechanical properties (mass, damping, stiffness), a structure have eigenmodes of deformation that propagates in time. Wu *et al.* [Wu et al. 2001] proposed a formula to obtain the time step limit Δt_{crit} in order to have stable behavior in explicit schemes: $\Delta t_{crit} = h\sqrt{\frac{2\rho}{E}}$, where h is related to the size of the elements, ρ is the material density and E the Young modulus.

Implicit time integration has the advantage of being unconditionally stable. However, it does not mean that we can chose any time step. A deformation eigenfrequency of our object, w, according to Shannon's theorem, will be sampled only if: $\frac{2\pi}{\Delta t} \geq w$. Otherwise, this frequency will be filtered by the implicit scheme. Moreover, the deformation eigenfrequencies are changing according to state of the constraints that are applied on the spline. The more constraints are numerous, the more eigenfrequencies are high and the dynamical system is likely to be over damped if Δt is not adapted. We propose a simplified rule to change Δt during the simulation of the dynamic spline: Δt_0 is evaluated at the beginning of the simulation when no constraint are applied on the cable. Then, at each step, we evaluate the maximum length of the spline without any constraint l_{free}. The ratio between this length and the initial length l_{init} is used to balance the value of the time step: $\Delta t = \frac{l_{free}}{l_{init}}\Delta t_0$.

We recall that our target is to perform interactive physical simulation, so the time passed between two steps needs to be close to Δt. However, it is not possible to guaranty that the computation time t_c will never exceed Δt. So we propose to measure t_c during simulation in order to anticipate the moment when temporal consistency will not be held. This anticipation rule is based on t_c measured in previous step. If it exceeds 95% of the supposed Δt for this step, the simulation is switch to static. In order to avoid a permanent switch between static and dynamic models, we propose to create a hysteresis by introducing an smaller value for switching from static to dynamic: for instance, we used $0.9\Delta t$ (see Algorithm 2).

Algorithm 2 Quasi-dynamic algorithm

```
t_c = 0.9Δt_0;
Dynamic System = true;
while Simulation do
    beginTimeMeasure();
    Δt = (l_free / l_init) Δt_0;
    if !Dynamic System then
        V = Δq / t_c
    end if
    if singularityTest() then
        Dynamic System = true;
    else
        if t_c > 0.95Δt then
            Dynamic System = false;
        else if t_c < 0.9Δt then
            Dynamic System = true;
        end if
    end if
    Compute mechanics () {see Algorithm 1}
    t_c = elapsedTimeMeasured()
    sleep(Δt − t_c);
end while
```

5.4 Discussion

These two heuristic rules are fully compatible and somehow go to the same direction: when the dynamic spline model is under-constrained, the matrix system size is smaller so the computation time is better and the eigenfrequencies are lower so the time step can be quite large. All parameters make a dynamic simulation possible and the singularity criterion will enforce it. On the contrary, when the model is over-constrained, the singularity criterion allows a static simulation which seems to be more adapted: computation time is longer and eigenfrequencies are higher, so the time step is small. In section 4.4, we mentioned that the robustness of the method was reinforced by a displacement ceiling. This ceiling ensures that the future configuration of the dynamic spline is not too far from the current configuration, which would distort the linearized model. As the ceiling limits the displacement for one computation step (it is somehow a speed limit), it can be seen as an artificial damping. However, it means that the equilibrium state can be reach in more than only one step even if in static theory, there is no transition states before two equilibrium states. But, in practice, we find convenient to use these transitions to feed the animation because it helps a smooth switch from dynamic to static simulation.

6 Results

The model described in this paper has been implemented in C++. We performed several tests using a 1.6GHz Intel Pentium M and a ATI X600 graphic card. We solve the resulting system by using the Intel Math Kernel Library (MKL), which includes highly optimized

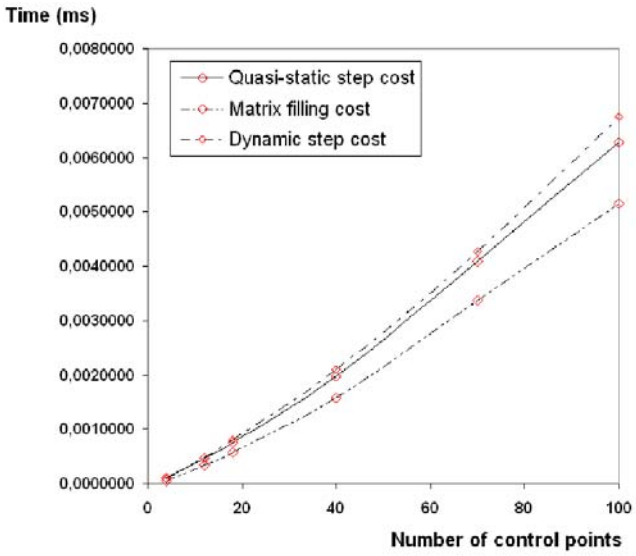

Figure 2: Computation time cost of a simulation step for a LU decomposition and solving, depending on the control point amount, on a simple Pentium M 1.6Ghz.

LAPACK routines for the platform we work on. The figure 2 sums up computations costs for a LU decomposition for band matrix. We can notice several interesting characteristics. First, our algorithm is fully linear time with the number of control point n. Second, quasi-static and dynamic systems are nearly as fast to compute and solve; it facilitates much the switching between them. Third, a major part of the step computation is dedicated to the matrix filling (80% with 20 or more control points). This task is consequently the bottleneck of our algorithm. This indeed justifies the very intensive care that has been taken in the present work about optimizing $\mathbb{K}$ and $\mathbf{F}$ computation. For the same number of degree of freedom, the resulting one-dimensional object simulation is 6 times faster than Gregoire's model [Grégoire and Schömer 2006]. Because of the non-linear geometry that is used, one has also to keep in mind that each degree of freedom (control point) has a greater power of representation than classical linear discretization, based on mass-springs.

We illustrate the potential benefit of our model on an application of virtual cable positioning on the inner structure of a car door. Several steps are shown in figure 1. The purpose is to test compatibility between planned fixing clip positions, and mechanical cable properties. Car engineers still need to build prototypes, since existing solutions are not accurate enough. Our model can prevent them from undergoing this fastidious and inevitable step. In this application, fixing clips are mechanically modeled as a set of lagrangian constraints. In our application, we consider simple fixed point constraints; if larger clips were needed, combination of fixed points, and fixed first derivative constraints would provide satisfactory results. During the interactive manipulation, when the cable meets a fixing clip, we create a point constraint. Solving these constraints gives the required forces or multipliers λ to maintain the global equilibrium of the cable. If a resulting force overwhelms the fixing clip strength in a determinated direction, the clip fails to keep the cable: the corresponding constraint is deleted. This real-time simulation of a 18 control point spline uses quasi-static and dynamic integration. It dynamically switches from dynamic to static when the cable is pulled at the end of the companion video. The time step of 1ms is computed in an average time of 0.8ms, with a decent framerate of 140 frames per second.

7 Conclusion and future works

We have proposed a deformable model for one-dimensional objects, which gathers geometrically exact accuracy and efficient computation. Our simulation method is linear time and automatically switch between dynamic and static representation according to the context. The addressed class of simulated objects is quite large, both in terms of geometry, and material, from surgical threads to steel cables. For taking best of available computation time, a natural (yet, still challenging research), would be to adapt computation using a dynamic adaptive repartition of control points. Adding the ability to handle contact and friction with the environment would also be needed for improving realism and simulation possibilities, in situations like suturing and assembling.

Acknowledgement

This work has been carried out within the framework of the INRIA Alcove project and is supported by the IRCICA and the CEA LIST.

References

BARAFF, D., AND WITKIN, A. 1998. Large steps in cloth simulation. In *Computer Graphics Proceedings, Annual Conference Series*.

BERTAILS, F., AUDOLY, B., CANI, M.-P., QUERLEUX, B., LEROY, F., AND LÉVÊQUE, J.-L. 2006. Super-helices for predicting the dynamics of natural hair. In *ACM Transactions on Graphics (Proceedings of the Siggraph Conference)*.

CORTES, L. A., ELES, P., AND PENG, Z. 2005. Quasi-static scheduling for multiprocessor real-time systems with hard and soft tasks. In *International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'05)*, IEEE, 422–428.

COURBON, J., 1980. Théorie des poutres.

EVTUSHENKO, Y., MORETTI, A., AND ZHADAN, V. Newton's steepest descent method for linear programming.

GRÉGOIRE, M., AND SCHÖMER, E. 2006. Interactive simulation of one-dimensional flexible parts. In *Symposium on Solid and Physical Modeling*, ACM, 95–103.

LENOIR, J., GRISONI, L., MESEURE, P., RÉMION, Y., AND CHAILLOU, C. 2004. Smooth constraints for spline variational modeling. In *Graphite*, 58–64.

LENOIR, J., MESEURE, P., GRISONI, L., AND CHAILLOU, C. 2004. A suture model for surgical simulation. *2nd International Symposium on Medical Simulation (ISMS'04)* (june 17-18), 105–113.

LENOIR, J., GRISONI, L., MESEURE, P., AND CHAILLOU, C. 2005. Adaptive resolution of 1d mechanical B-spline. In *Graphite*, 395–403.

NOCENT, O., AND RÉMION, Y. 2001. Continuous deformation energy for dynamic material splines subject to finite displacements. In *Proceedings of the Eurographic workshop on Computer animation and simulation*, Springer-Verlag New York, Inc., New York, NY, USA, ACM, 88–97.

PAI, D. 2002. STRANDS: Interactive simulation of thin solids using cosserat models. *Computer Graphics Forum (Eurographics'02) 21-3*.

QIN, H., AND TERZOPOULOS, D. 1996. D-NURBS: A physics-based framework for geometric design. In *Transactions on Visualization and Computer Graphics*, vol. 2-1, IEEE, 85–96.

REDON, S., GALOPPO, N., AND LIN, M. C. 2005. Adaptive dynamics of articulated bodies. *ACM Transactions on Graphics (SIGGRAPH 2005) 24*, 3.

THEETTEN, A., GRISONI, L., ANDRIOT, C., AND BARSKY, B. 2006. Geometrically exact dynamic splines. Tech. rep., INRIA-Futurs.

WAKAMATSU, AND HIRAI. 2004. Static modeling of linear object deformation based on differential geometry. *The International Journal of Robotics Research. 23*, 293-311.

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*, 349358.

A Calculation of generalized force vector and stiffness matrix

A.1 Force

We recall here the expressions demonstrated in [Theetten et al. 2006] and simplify them considering splines are initially straigth and not twisted. Deformed initial states involve more complex expressions and limited developments. Furthermore, we introduce the following variables for compactness of the equations: $\mathtt{C} = \mathbf{r}' \times \mathbf{r}''$, $\mathtt{P}^i = \frac{\partial \mathtt{C}}{\partial \mathbf{r}_i}$, $\mathtt{R}^i = \mathtt{C} b_i''' - \mathtt{P}^i \times \mathbf{r}'''$, $\mathtt{G}^i = \mathtt{C} \times \mathtt{P}^i$, $\mathtt{T}^i = \mathtt{R}^i - 2\tau\mathtt{G}^i$, $\mathtt{F}^i = \frac{\mathtt{T}^i}{\|\mathtt{C}\|^2} - \frac{\theta' \mathbf{r}' b_i'}{\|\mathbf{r}'\|^2}$ and $\mathtt{B}^{i,j} = b_j'' b_i' - b_j' b_i''$. For the sake of optimization, they should be precomputed at each time step before generalized forces and stiffnesses. The stretching generalized force yields:

$$P_s^i(\mathbf{r}) = -\frac{\pi E D^2}{4} \int_0^L \left(1 - \frac{\|\mathbf{r}_0'\|}{\|\mathbf{r}'\|}\right) \mathbf{r}' b'_i ds$$

The twisting generalized force yields:

$$P_t^i(\mathbf{r}) = -\frac{\pi G D^4}{32} \int_0^L \varepsilon_t \mathtt{F}^i ds, P_t^i(\theta) = -\frac{\pi G D^4}{32} \int_0^L \varepsilon_t \frac{b_i{}'}{\|r'\|} ds$$

The bending generalized force yields:

$$P_b^i(\mathbf{r}) = -\frac{\pi E D^4}{64} \int_0^L \frac{\mathtt{G}^i}{\|\mathbf{r}'\|^6} - 3\frac{k^2 b'_i \mathbf{r}'}{\|\mathbf{r}'\|^2} ds$$

A.2 Stiffness

The force expressions are used to calculate the stiffness matrix $\mathbb{K}$, the hessian matrix of the strain energy U. We define the set $C = \{\alpha, \beta, \gamma, \theta\}$ as a permutation of the elements of the set $\{x, y, z, \theta\}$ Considering two elements c_1 and c_2 from C, an element of $\mathbb{K}$ results in the following expression: $K_{c_1,c_2}^{i,j}(\mathbf{r}, \theta) = -\frac{\partial P_\alpha^i(\mathbf{q}_{c_1})}{\partial c_2^j}$. As $\mathbb{K}$ is a symmetric matrix, its elements have the following property: $K_{c_2,c_1}^{j,i}(\mathbf{r}, \theta) = K_{c_1,c_2}^{i,j}(\mathbf{r}, \theta)$.

A.2.1 Stretching

$$\begin{aligned} K_{s,(\alpha,\alpha)}^{i,j}(\mathbf{r}) &= \frac{\pi E D^2}{4} \int_0^L \left(1 + \frac{\mathbf{r}_\alpha'^2 \|\mathbf{r}_0'\|}{\|\mathbf{r}'\|^3} - \frac{\|\mathbf{r}_0'\|}{\|\mathbf{r}'\|}\right) b'_i b'_j ds \\ K_{s,(\alpha,\beta)}^{i,j}(\mathbf{r}) &= \frac{\pi E D^2}{4} \int_0^L \frac{\mathbf{r}'_\alpha \mathbf{r}'_\beta \|\mathbf{r}_0'\|}{\|\mathbf{r}'\|^3} b'_i b'_j ds \end{aligned}$$

A.2.2 Twisting

$$K_{t,(\alpha,\alpha)}^{i,j}(\mathbf{r}, \theta) = \frac{\pi G D^4}{32} \int_0^L \mathtt{F}_\alpha^j \mathtt{F}_\alpha^i + \varepsilon_t \left(-2\frac{\mathtt{R}_\alpha^i \mathtt{G}_\alpha^j + \mathtt{R}_\alpha^j \mathtt{G}_\alpha^i}{\|\mathtt{C}\|^4} + 8\tau\frac{\mathtt{G}_\alpha^i \mathtt{G}_\alpha^j}{\|\mathtt{C}\|^4}\right.$$

$$K_{t,(\theta,\theta)}^{i,j}(\mathbf{r}, \theta) = \frac{\pi G D^4}{32} \int_0^L \frac{b'_i b'_j}{\|\mathbf{r}'\|^2} ds \qquad \left. \ddots -2\tau\frac{\mathtt{P}_\beta^i \mathtt{P}_\beta^j + \mathtt{P}_\gamma^i \mathtt{P}_\gamma^j}{\|\mathtt{C}\|^2}\right) ds$$

$$K_{t,(\alpha,\theta)}^{i,j}(\mathbf{r}, \theta) = \frac{\pi G D^4}{32} \int_0^L \frac{b'_j}{\|\mathbf{r}'\|} \mathtt{F}_\alpha^i - \frac{\varepsilon_t b'_i b'_j \mathbf{r}'_\alpha}{\|\mathbf{r}'\|^3} ds$$

‘

$$K_{t,(\alpha,\beta)}^{i,j}(\mathbf{r}, \theta) = \frac{\pi G D^4}{32} \int_0^L \mathtt{F}_\beta^j \mathtt{F}_\alpha^i + \varepsilon_t \left(\frac{-\mathtt{P}_\gamma^j b_i''' + \mathtt{P}_\gamma^i b_j''' + \mathtt{B}^{i,j} r_\gamma'''}{\|\mathtt{C}\|^2}\right.$$

$$\left. -2\frac{\mathtt{R}_\alpha^i \mathtt{G}_\beta^j + \mathtt{R}_\beta^j \mathtt{G}_\alpha^i}{\|\mathtt{C}\|^4} + 8\tau\frac{\mathtt{G}_\alpha^i \mathtt{G}_\beta^j}{\|\mathtt{C}\|^4} + 2\tau\frac{\mathtt{P}_\alpha^j \mathtt{P}_\beta^i + \mathtt{C}_\gamma \mathtt{B}^{i,j}}{\|\mathtt{C}\|^2}\right) ds$$

A.2.3 Bending

$$K_{b,(\alpha,\alpha)}^{i,j}(\mathbf{r}) = \frac{\pi E D^4}{64} \int_0^L \frac{\mathtt{P}_\beta^i \mathtt{P}_\beta^j + \mathtt{P}_\gamma^i \mathtt{P}_\gamma^j}{\|\mathbf{r}'\|^6}$$

$$-6\frac{\left(\left(\mathtt{C} \times \mathtt{P}^i\right)_\alpha b'_j + \mathtt{G}_\alpha^j b'_i\right) \mathbf{r}'_\alpha}{\|\mathbf{r}'\|^8} + 3\frac{k^2 b'_i b'_j}{\|\mathbf{r}'^2\|} \left(8\frac{\mathbf{r}_\alpha'^2}{\|\mathbf{r}'\|^2} - 1\right) ds$$

$$K_{b,(\alpha,\beta)}^{i,j}(\mathbf{r}) = \frac{\pi E D^4}{64} \int_0^L -\frac{\mathtt{P}_\alpha^j \mathtt{P}_\beta^i + \mathtt{C}_\gamma \mathtt{B}^{i,j}}{\|\mathbf{r}'\|^6} - 6\frac{\mathtt{G}_\alpha^i \mathbf{r}'_\beta b'_j + \mathtt{G}_\beta^j \mathbf{r}'_\alpha b'_i}{\|\mathbf{r}'\|^8} + 24k^2 b'_i b'_j \frac{\mathbf{r}'_\alpha \mathbf{r}'_\beta}{\|\mathbf{r}'\|^4} ds$$