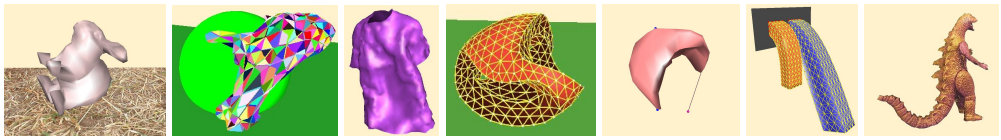
Efficient, physically plausible finite elements

Matthieu Nesme[1,2], Yohan Payan[2] and François Faure[1]

[1]GRAVIR/IMAG-INRIA, [2]TIMC/IMAG - Grenoble, France



Abstract

This paper discusses FEM-based simulations of soft bodies in terms of speed and robustness. To be physically plausible, three fundamental laws must be respected: rotational invariance, Newton's law and Euler's law. We show that precomputed strain-displacement matrices generate nonphysical torques which can lead to visual artifacts. We then derive the fastest FEM-based method meeting our criteria of plausibility and robustness and discuss their limitations.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer graphics]: Physically based modeling I.3.7 [Computer graphics]: Animation

1. Introduction and related work

Since Terzopoulos simulated viscoelastic deformable bodies [TPBF87], physically-based animation has its place in the field of computer graphics. Making real-time simulators with the objective of visual "realism" is now a central problem. Indeed, the graphics community now seeks to extend its modeling tools towards mathematical methods closer to the mechanics of continuous medium (from "realism" to "precision"). Because of the need for speed, the first interactive methods were based on precomputed matrix inversion [CDC*96, JP99]. Unfortunately these models are only valid for small displacements, this is why a non-linear computation of the deformations is used in [DDCB01, PDA03]. Recently proposed methods favor a new approach based on the decomposition of the displacement of each element into a rigid motion and a pure deformation tractable linearly in the local frame [MDM*02, EKS03, HS04, MG04], an idea first introduced in [TF88].

In this paper, we investigate how a FEM-based simulator can fulfill three major plausibility criteria: invariance to rotation, Newton's law on linear acceleration and Euler's law on angular acceleration. We show that it is necessary to recompute the strain-displacement matrix of each element at each time step to avoid ghost torques. Based on this, we propose a new implementation of tetrahedron-based FEM using an efficient computation and storage of the stiffness matrices and compare it with the current state-of-the-art methods.

2. Physically plausible linear FEM

We consider the standard finite element method (FEM) used to simulate tetrahedrized viscoelastic solids. Background can be found in standard texts [ZC67]. The force applied by the deformed element to its sampling points is given by

$$f = B^T \sigma = B^T D \varepsilon = B^T DBu = B^T DB(x - x^0) \qquad (1)$$

where σ is the stress, ε the strain, u the displacement, x and x^0 the current and the initial positions, B is the strain-displacement matrix and D is the stress-strain matrix. This article is limited to linear elastic material, but more complex relations can be used between σ and ε.

2.1. Rotational invariance

The linear equation (1) is insensitive to translations but inaccurate for large rotations of the elements. This results in so-called "ghost forces" which make the element artificially inflate. A possible approach to solve this problem is to use non-linear Green's strain tensor which is rotationally invariant. However this tensor is not able to linearly relate deformation to displacement except asymptotically for small displacements. An alternative approach was proposed by Müller *et al* [MDM*02] who decompose the displacement in a rigid rotation combined with a deformation. The net force calculation becomes $f = R^T B^T DB(Rx - x^0)$, where matrix R, which encodes the rotation of a local frame with respect to its initial orientation, is updated at each frame.

Three edges are used to compute the 3×3 transformation

matrix: $J = \begin{bmatrix} e_1^0 & e_2^0 & e_3^0 \end{bmatrix}^{-1} [e_1 \;\; e_2 \;\; e_3]$, with respect to the initial state, where the e_i^0 are the initial edge vectors and the e_i are the current ones. Matrix J is then decomposed in order to extract separately a rigid rotation R applied to the element and a deformation E as shown *fig. 1*. This decomposition is not unique and several approaches can be considered.

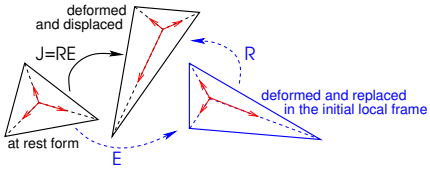


Figure 1: *An initial tetrahedron is deformed by the transformation J composed both a rigid motion R and the deformations are contained in E.*

Polar decomposition Etzmuß *et al* [EKS03], followed by other authors [HS04, MG04], presented a method based on the polar decomposition using eigenvalues and eigenvectors. The polar decomposition of a square matrix computes the nearest orthogonal frame to the given column axes [EKS03, MG04, HS04]. As such it provides the ideal decomposition of the displacement matrix J, giving the smallest deformations. The strain values can be derived as shown in the following formula.

$$\begin{aligned} J &= R_p.E_s \\ E_s &= R_p^{-1}J = \begin{bmatrix} 1+\varepsilon_{xx} & \varepsilon_{xy} & \varepsilon_{xz} \\ \varepsilon_{xy} & 1+\varepsilon_{yy} & \varepsilon_{yz} \\ \varepsilon_{xz} & \varepsilon_{yz} & 1+\varepsilon_{zz} \end{bmatrix} \end{aligned}$$

A related SVD-based approach has been used to handle element inversions [ITF04].

QR decomposition † The QR decomposition is an alternative to the polar approach. The first axis of the local frame is constrained to be aligned with the first column of J. Then the second axis is constrained to the plane spanned by the two first columns, an so on. We can compute it by performing a Gram-Schmidt orthogonalization, to guarantee that we obtain a right-handed frame. The strain can then be computed by projecting the columns of J to the axes of the local frame, or equivalently by the following decomposition:

$$\begin{aligned} J &= R_{qr}.E_t \\ E_t &= R_{qr}^{-1}J = \begin{bmatrix} 1+\varepsilon_{xx} & 2\varepsilon_{xy} & 2\varepsilon_{xz} \\ 0 & 1+\varepsilon_{yy} & 2\varepsilon_{yz} \\ 0 & 0 & 1+\varepsilon_{zz} \end{bmatrix} \end{aligned}$$

This decomposition is significantly faster than polar or SVD, however it depends on vertex ordering because all edges do not have the same influence, as illustrated in *fig. 2*. Consequently some ordering-dependent anisotropy is introduced, contrary to polar or SVD. Moreover, the evaluated strain is a bit higher. However, its computational efficiency can allow one to use more refined meshes.

† With our notations, Q corresponds to the rotation R_{qr} and R to E_t

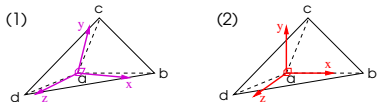


Figure 2: *the local frames. (1) Polar decomposition: single frame reflecting best the matter, nearest to the edges. (2) QR decomposition: the first axis is the first edge ab, the second axis is orthogonal to the first on plane (ab, ac), and the last axis is obtained by construction of an orthonormal frame.*

2.2. Newton's law

Newton's law on linear acceleration relates the acceleration of a system to the external forces applied to it: $\Sigma_j m_j \ddot{u}_j = \Sigma_j f_j^{ext}$ where f_j^{ext} is the net external force, m_j the masse, $\ddot{u}_j$ the acceleration applied to sampling point x_j. This law is true for a single particle, for an element as well as for the whole object. The violation of this law would allow an isolated (not submited to external forces) object to linearly accelerate.

We now show that Newton's law is necessarily satisfied by the construction of the strain-displacement matrix B, thanks to its property $\Sigma_j B_{ij} = 0$, for a row i. Indeed, for any uniform translation $\Delta u = [k...k]^T$, $k \in \mathbb{R}$ this property implies a null variation of the deformation $\Delta\varepsilon$: $\Delta\varepsilon_i = \Sigma_j B_{ij} \Delta u_j = 0$. Moreover, the net force generated by an arbitrary constraint vector σ is $\Sigma_j f_j = \Sigma_j \Sigma_i B_{ij}^T \sigma_i = \Sigma_i \sigma_i \Sigma_j B_{ij}^T = 0$. Note that the property is true even if B is obsolete due to a change of the shape of the element, even if it modifies the material, it does not create ghost forces. On the other hand, this property is not guaranted by [MDM*02], because it evaluates different local frame rotations for each node of a same element in processing one node after the other. Hence, methods processing one element after the other (presented in the previous section) are now prefered among the community.

2.3. Euler's law

Euler's law relates the angular acceleration of a system to the net torque applied to it: $\Sigma_j u_j \times m_j \ddot{u}_j = \Sigma_j u_j \times f_j^{ext}$. The violation of this law would allow an isolated object to angularly accelerate. We now show that if matrix B is not up-to-date then Euler's law is not necessarily satisfied.

To respect Euler's law, let us show that the following property, true by construction of B, must be verified: $\Sigma_j x_j \times B_{ij}^T = 0$. Indeed, a pure rotation ω generates a variation of the displacements $\Delta u_j = \omega \times x_j$ but must not generate a variation of the deformation. This implies that $\Delta\varepsilon_i = \Sigma_j B_{ij} \omega \times x_j = 0$ for any ω, thus $\Sigma_j x_j \times B_{ij}^T = 0$. In the same way, let us check that the net torque due to an arbitrary constraint σ is null: $\Sigma_j x_j \times f_j = \Sigma_j x_j \times \Sigma_i B_{ij}^T \sigma_i = \Sigma_i \sigma_i \Sigma_j x_j \times B_{ij}^T = 0$. The property is no more guaranted when B is obsolete due to a change of shape because the original x_j are replaced by new values. Computing forces with initial strain-displacement matrices amounts at computing $f_{rest \to deformed}$ whereas $f_{deformed \to rest}$ is sought. Consequently, it is necessary to recompute each matrix B element's at each time step to avoid artificial torques. An example of artificial torque is given in *fig. 3*. Note, however, that multiplying matrix B with a scalar uniformly scales the net torque, and thus modifies the material, but does not induce artificial torques.

3. Efficient implementation

Our method updates strain-displacement matrices at each time step to avoid ghost torques. Deformations computation is based on the QR decomposition, because this decomposition simplifies a lot the calculations of strain-displacement matrices and becomes significantly faster. About the dynamic resolution, we show that the assembly is not inevitably the best approach in the case of interactive simulations.

3.1. Strain-displacement matrix computation

The strain-displacement matrix B is a 6×12 matrix straightfowardly deduced from shape functions, factored by $1/6V$ where V is the volume of the element. The classic computation of B takes 72 multiplications and 60 additions [BNC96], and the computation of $\Delta f = R^T B^T DB\Delta u$ using this matrix takes 6660 multiplications and 2760 additions. Since we are using QR decomposition, a lot of vertex coordinates are null in the local frame. The calculation of the strain-displacement matrix is thus greatly simplified. It is possible to recompute it at each time step using only 14 multiplications and 5 additions, and perform an optimized computation of Δf using 4554 multiplications and 1707 additions. For a tetrahedron (a,b,c,d), the coefficients of the shape functions $N_i = \alpha_i + \beta_i x + \gamma_i y + \delta_i z$ are, in the local frame:

$$\beta_a = -y_c z_d$$
$$\gamma_a = (x_c z_d) - (x_b z_d)$$
$$\delta_a = y_c x_d - x_c y_d + x_b y_d - x_b y_c$$

$$\begin{array}{lll} \beta_b = y_c z_d & \beta_c = 0 & \beta_d = 0 \\ \gamma_b = x_c z_d & \gamma_c = z_d x_b & \gamma_d = 0 \\ \delta_b = y_c x_d - x_c y_d & \delta_c = -y_d x_b & \delta_d = -x_b y_c \end{array}$$

It is shown in section 2.3 that it remains physically plausible when multiplied by a scalar. We can exploit this opportunity to use each element's initial volume instead of recomputing it. The advantages are a faster computation and robustness when large deformations result in flat elements with null volume.

3.2. Time integration

To dynamically interact with a FEM-based system, we solve a second order differential equation, globally, on all the elements vertices: $\mathbf{M\ddot{u}} + \mathbf{C\dot{u}} + \mathbf{Ku} = \mathbf{f}$ [‡], where matrix $\mathbf{M}$ models mass, $\mathbf{C}$ models damping ($\mathbf{C} = \alpha\mathbf{K} + \beta\mathbf{Id}$ is a popular approximation) and $\mathbf{K}$ models stiffness for all the vertices. The global matrices can be computed by summing up the contributions of each element to its vertices. This operation is called the assembly. Baraff [BW98] has shown how to solve this differential equation efficiently even in the case of stiff material. A modified conjugate gradient algorithm is used to iteratively solve a sparse linear equation system

‡ Bold letters denote global matrices and vectors, as opposed to single elements.

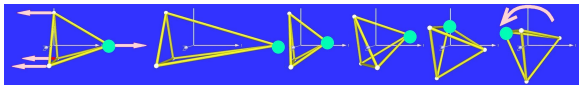


Figure 3: *A set of forces with null net torque is applied to an equilateral tetrahedron. If the strain-displacement matrix is not updated, then the tetrahedron starts to rotate.*

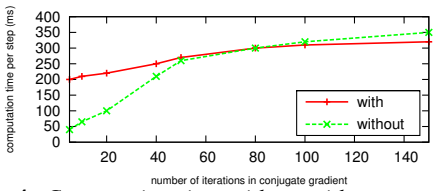


Figure 4: *Computation time with or without assembly, on 3430 tetrahedra and 512 particles.*

modeling a constrained elastic system. The main computational task consists in evaluating $\Delta\mathbf{f} = [\frac{\partial \mathbf{F}}{\partial \mathbf{u}}]\Delta\mathbf{u}$, where matrix $[\frac{\partial \mathbf{F}}{\partial \mathbf{u}}]$ is the stiffness matrix $\mathbf{K} = \mathbf{B}^T\mathbf{DB}$. When the stiffness matrix is precomputed, the calculation of the net forces ("right part" of the integration) can be optimized by precomputing $f^0 = R^T K x^0$, with $f = R^T K(Rx - x^0) = R^T KRx - f^0$ as shown in [MG04]. In this case, the computation of f and $\Delta f = R^T KR\Delta u$ use the same product by the stiffness matrix $R^{-1}KR$, so it is interesting to assembly all the individual stiffness matrices of all the elements, to limit the calculation by a single force computation by vertex. But we have shown in the section 2.3 that updating the stiffness matrix is mandatory. In this case, f^0 can't be precomputed, so we need two different products: one by $R^T KR$ for Δf and another by $R^T K$ for f. In pratice, it is more efficient not to build an assembled stiffness matrix ; its heavy construction could be amortized by a lighter computation of the conjugate gradient iterations, but in the case of interactive animations, the number of iterations is generally too small. It is preferable to store separately R, B and D and to process each element independently. For each element, we first compute $R\Delta u$, then $BR\Delta u$ until $R^T B^T DR\Delta u$.

Figure 4 shows that to amortize the cost of the assembly, 50 iterations minimum are necessary in this example, which is really too big in the case of an interactive simulation.

4. Discussion and results

4.1. Robustness

Large displacements or user manipulations sometimes result in degenerate configurations such as flat or inverted elements. Such cases are not properly speaking physical, but it is important to be able to face them to guarantee the stability of the simulator. The polar decomposition applied to an inverted element computes a left-handed local frame. The element tends to recover its initial shape in this frame, converging to a reversed shape. This can be solved by flipping the sign of an axis, but this requires the computation of the determinant to detect a change of sign resulting from the inversion. Irving *et al* [ITF04] propose a very elegant, but more expensive, solution to this problem, based on a SVD decomposition. They always compute the smallest inversion among

Figure 5: *A rabbit with null Young's modulus is crushed onto the ground. By increasing its stiffness, it regains its initial shape.*

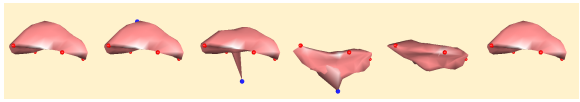


Figure 6: *A liver is fixed in four points (red balls). A user imposes a violent displacement (blue ball) which reverses elements. The system remains stable, and recovers its initial shape when released.*

all the possible combinations.
Using the QR decomposition method, the inversion of an element is freely detected and automatically modeled as the crossing of the (a,b,c) plane by vertex d, the only one not used in the construction of the local frame. If the inversion actually occurs at this fourth vertex, then the reaction is plausible, but in case of another vertex inversion, the element regains its rest form by a large rotation. On a single tetrahedron, this can lead to a non-intuitive behavior. In a complex model, most elements are not inverted and behave correctly and we have not detected visible artifacts. In all cases, the tetrahedra do not break and they recover their initial shape, as illustrated in *fig. 5 and 6*.

4.2. Efficiency

The results given *fig. 7* use a time step of 0.4 ms and five iterations of the conjugate gradient solution are performed in the implicit integration. We can see the major interest of the QR decomposition applied to tetrahedra concerning the computation time, which is about 30% faster than the method using the polar decomposition when precomputation of B is disabled to ensure correctness basic mechanic laws. By increasing the number of iterations in the conjugate gradient, results remain similar, because the computation of Δf at each iteration is more efficient with QR decomposition than polar decomposition. With the display, our method runs at 30 frames per second for approximately 2000-3000 tetrahedra.

5. Conclusion

We have shown that physical plausibility requires the update of the strain-displacement matrix. We have proposed an efficient implementation for interactive applications. Rotational invariance and robustness of tetrahedra are more efficiently handled using the QR decomposition, while the polar decomposition is preferable to enforce isotropy. Finally, we have seen that the cost of stiffness assembly is difficult to amortize in case of interactive simulations using few iterations in implicit integration. In the near future, we would like to extend our model to hexahedral elements and compare the accuracy of such interactives methods.

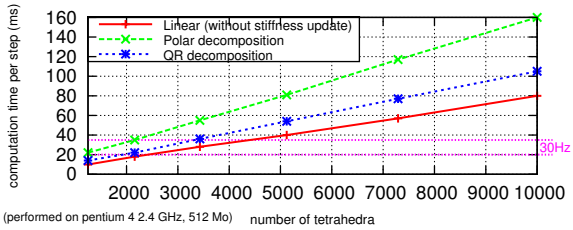


Figure 7: *Computation time per time step against number of tetrahedra (with stiffness update, without assembly).*

References

[BNC96] BRO-NIELSEN M., COTIN S.: Real-time volumetric deformable models for surgery simulation using finite elements and condensation. In *Proc Eurographics* (1996). 3

[BW98] BARAFF D., WITKIN A.: Large steps in cloth simulation. In *Proc SIGGRAPH* (1998). 3

[CDC*96] COTIN S., DELINGETTE H., CLEMENT J.-M., TASSETTI V., MARESCAUX J., AYACHE N.: Volumetric deformable models for simulation of laparoscopic surgery. In *Computer Assisted Radiology* (1996). 1

[DDCB01] DEBUNNE G., DESBRUN M., CANI M.-P., BARR A. H.: Dynamic real-time deformations using space and time adaptive sampling. In *Computer Graphics Proceedings* (2001). 1

[EKS03] ETZMUSS O., KECKEISEN M., STRASSER W.: A Fast Finite Element Solution for Cloth Modelling. *Proc Pacific Graphics* (2003). 1, 2

[HS04] HAUTH M., STRASSER W.: Corotational simulation of deformable solids. In *Proc WSCG* (2004). 1, 2

[ITF04] IRVING G., TERAN J., FEDKIW R.: Invertible finite elements for robust simulation of large deformation. In *Proc SCA* (2004). 2, 3

[JP99] JAMES D., PAI D.: Accurate real time deformable objects. In *Proc SIGGRAPH* (1999). 1

[MDM*02] MÜLLER M., DORSEY J., MCMILLAN L., JAGNOW R., CUTLER B.: Stable real-time deformations. In *Proc SCA* (2002). 1, 2

[MG04] MÜLLER M., GROSS M.: Interactive virtual materials. In *Proc Graphics Interface* (2004). 1, 2, 3

[PDA03] PICINBONO G., DELINGETTE H., AYACHE N.: Nonlinear anisotropic elasticity for real-time surgery simulation. *Graph. Models* (2003). 1

[TF88] TERZOPOULOS D., FLEISCHER K.: Modeling inelastic deformation : viscoelasticity, plasticity, fracture. In *Proc SIGGRAPH* (1988). 1

[TPBF87] TERZOPOULOS D., PLATT J., BARR A., FLEISCHER K.: Elastically deformable models. In *Proc SIGGRAPH* (1987). 1

[ZC67] ZIENKIEWICZ O. C., CHEUNG Y. K.: *The Finite Element Method in Structural and Continuum Mechanics*. McGraw-Hill Publ, 1967. 1