Towards Real-time FE for surgical applications

Vukasin Strbac, Jos Vander Sloten, Nele Famaey
Biomechanics Section, Department of Mechanical Engineering KU Leuven, Belgium
Email: vule.strbac@mech.kuleuven.be

***Abstract*—As the Finite Element(FE) method is becoming more pervasive in scientific and engineering endeavours the application space of the method increases as well. This work provides indication that real-time surgical application is also possible. The challenge is reducing the solution times of this, generally computationally expensive method, such that FE on patient organs under the influence of surgical instruments can provide valuable information to the interventionalist during surgery.**

We present an implementation of a Total Lagrangian Explicit Dynamic Finite Element algorithm implemented on General Purpose Graphics Processing Units(GPGPUs) that suggests the fulfillment of this constraint. The usability of the method is demonstrated by conducting a broad assay on ranging model sizes and GPGPUs and comparing to an industry-proven FE code Abaqus.

I. Introduction

Modern surgical theaters today are capable of providing electromagnetic or optical position tracking systems that can be used to locate the position of instruments under the control of the surgeon. Likewise, current imaging techniques are capable of producing highly detailed geometries of patient specific anatomy. By combining these data with an appropriate material model of the organ or organs under consideration all boundary condition requirements for FE in surgery are met. Providing near-instantaneous response on the state of the tissue under mechanical load (pressing, pulling, cutting...) is instrumental in ensuring patient safety, thus creating the need for a real-time FE system.

Successful real-time implementations of nonlinear finite elements have been sparse in publications, mostly likely due to the limited computational power of hardware and the high arithmetic intensity of conventional FE algorithms. The Total Lagrangian Explicit Dynamic(TLED) [1] tackles this problem in part by using the total lagrangian spatial discretization scheme. It ensures that shape function derivatives do not need to be recomputed every time step, thus additionally reducing the number of operations. By using explicit time integration (the central difference method), the nonlinearities are handled in a simple way, provided a small enough time step is chosen. Another benefit of explicit integration is that per-element computations can be done somewhat independently, adding individual contributions gradually to form the global force matrix. This feature along with the per-node fully independent computations lends itself well to parallelization. The explicit nature of the solution also facilitates the tracking of one or mode variables throughout the solution, e.g. the accumulation of damage [2].

Nvidia CUDA [3] technology enables us to use the massively parallel architecture of modern GPUs to conduct general-purpose floating-point computations in an accessible and simplified way. By distributing the computations over the cores of the card, most calculations of the same time step can be done at the same time, providing significant speed-up to the algorithm.

In section II, we present details on the algorithm and implementation, hardware and the boundary conditions of the simulation, followed by results (III) and the conclusions (IV) on the topic.

II. Materials and methods

A. Mechanical model

For the purposes of validating the accuracy of the proposed method we solve a uniaxial compression problem on a cube of uniform size and increasing element density. The material employed is a hyperelastic Neo-Hookean material(ref. eq.1, 2), often used as an estimate of the behavior of most soft tissues.

$$\Psi = \frac{\mu}{2}(\overline{I}_1 - 3) + \frac{\kappa}{2}(J-1)^2, \tag{1}$$

where μ and κ are material constants - shear and bulk modulus, respectively. $\overline{I}_1$ is the first invariant of the right Cauchy-Green deformation tensor $\mathbf{C}$. The second Piola-Kirchoff stress tensor $\mathbf{S}$ can be derived from the SEDF as:

$$\mathbf{S} = 2\frac{\partial \Psi}{\partial \mathbf{C}} = \mu J^{-\frac{2}{3}}(\mathbf{I} - \frac{tr(\mathbf{C})}{3}\mathbf{C}^{-1}) + \kappa J(J-1)\mathbf{C}^{-1}. \tag{2}$$

The elements used in the simulations are under-integrated hexahedra, which offer good results for a relatively low number of computing operations. No hourglassing preventive algorithms have been implemented. A smooth loading curve up to 20% strain was used to minimize inertial effects and enabling us to study the phenomena under quasi-static conditions.

The simple boundary conditions in our simulations prescribe no displacements in the axial direction (Z) for the bottom nodes and enforces displacements along the same axis for the top nodes, all other degrees of freedom are non-constrained.

The critical time step was computed by using linear theory [4] and proving sufficient to maintain the convergence.

Each model has identical material properties and was loaded in a time span of 5 seconds. The critical time step c size was determined by using the following formula.

$$L_e = \frac{V_e}{A_e}; c = sqrt\frac{\lambda + 2\mu}{\rho}; \Delta t = \frac{L_e}{c} \tag{3}$$

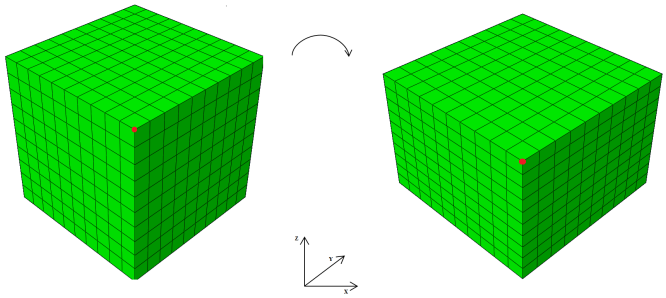

Fig. 1. Initial and deformed geometry and a reference point on a 10x10x10 model

Where L_e is the characteristic length, computed using the smallest element area in the model A_e and it's volume V_e, ρ is the mass density of the element. The 5 second loading time is therefore distributed along around 6000 to more than 50000 steps in different models.

B. Total Lagrangian Explicit Dynamic algorithm

The system to be solved is:

$$\mathbf{M}\ddot{\mathbf{u}} + c_d\mathbf{M}\dot{\mathbf{u}} + \mathbf{P}(\mathbf{u}) = \mathbf{R}, \tag{4}$$

where $\mathbf{M}$ is the diagonalized mass matrix, c_d is the damping coefficient, $\mathbf{u}$ are the nodal displacements, $\mathbf{P}$ are the internal forces and $\mathbf{R}$ are the external forces. [1], [5], [6], [4], for example, provide additional detail on the topic. A pseudo-code of the solution of the system in equation 4 is provided below.

Pre-computation phase:

1) Shape function derivatives in global coordinates from isoparamatric coordinates and positions, using the Jacobian matrix:

$$^0\mathbf{J}_e = \frac{\partial(x,y,z)}{\partial(\xi,\eta,\zeta)} = \nabla\mathbf{N}_{Iso}\mathbf{p}_e \tag{5}$$

$$\nabla\mathbf{N}_{Glo,e} = {}^0\mathbf{J}_e^{-1}\nabla\mathbf{N}_{Iso} \tag{6}$$

2) Construct strain-displacement matrix: $\mathbf{B}_e$
3) Compute element volumes and diagonalized mass matrix $\mathbf{M} = M_e\mathbf{I}$
4) Central difference method coefficients:

$$a = -\frac{2(\Delta t^2)}{2t\Delta t M_e}; \qquad b = \frac{1+(2-q\Delta t)}{2+q\Delta t}; \tag{7}$$

$$c = -\frac{2-q\Delta t}{2+q\Delta t}; \qquad q = \frac{2(1-{C_r}^2)}{\Delta t(1+C_r)} \tag{8}$$

Iteration phase:

4) Deformation gradient:

$$^t\mathbf{F}_e = \mathbf{I} + \nabla\mathbf{N}^t\mathbf{u}_e \tag{9}$$

5) Cauchy-Green strain tensor, Jacobian determinant:

$$^t\mathbf{C}_e = {}^t\mathbf{F}_e^{T\,t}\mathbf{F}_e\ ,\ {}^tJ_e = det(^t\mathbf{F}_e) \tag{10}$$

6) Second Piola-Kirchoff stress:

$$^t\mathbf{S}_e = 2\frac{\partial^t\mathbf{\Psi}_e}{\partial^t\mathbf{C}_e} \tag{11}$$

7) Force contribution:

$$^t\mathbf{P}_e = 8det(^0J_e)\mathbf{B}_e^{T\,t}\mathbf{S}_e \tag{12}$$

8) Summation of elemental contributions into the global force vector:

$$^t\mathbf{P} = \sum {}^t\mathbf{P}_e \tag{13}$$

9) New displacements:

$$^t\mathbf{u} = a\,{}^t\mathbf{P} + b\,{}^t\mathbf{u} + c\,{}^{t-\Delta t}\mathbf{u} \tag{14}$$

10) Displacement loading for next step:

$$^{t+\Delta t}\mathbf{u}_{BC} = {}^{t+\Delta t}\mathbf{d}_{BC} \tag{15}$$

In the above notation the left superscript denotes the time step, the right subscript denotes element scope. C_r is the convergence rate of the central difference scheme, and set close to 1, as relaxation effects were not considered and a small enough time step was used.

In light of increasing the speed of the algorithm, pre-computing all possible values is essential. Due to the spatial discretization scheme chosen, the shape function derivatives do not change accross the time steps, therefore we compute them here. The central difference method coefficients are also pre-computed as they too do not change throughout.

Single point Gaussian integration is used to obtain element nodal forces from element stress in step 7, which was in turn computed from the right Cauchy-Green tensor and deformation gradient in steps 4-6. Depending on the material model used, the stress computation in step 7 can vary. In our implementation, the Neo-Hookean material with and without damage are used, according to equation 2. After the computation of the global force vector in step 8, the nodal displacements are calculated using the central difference scheme. Finally, new displacement boundary conditions are assigned to the boundary nodes, to prepare for the next step. For a more detailed discussion on TLED, the reader is referred to [1].

C. Hardware and implementation

The pseudocode described in section II-B is implemented using Nvidia Compute Unified Device Architecture (CUDA). The GPGPU cards used for this purpose are an Nvidia Tesla C2075, K20c and GTX680 capable of running thousands of threads concurrently. For a detailed overview of CUDA, the reader is referred to [3].

The two most important factors in speeding up a program by using a GPGPU is replacing loops with parallel functions (kernels) and the pre-computation of all possible values for later use in the algorithm. Albeit, the former can be used only if iterations are independent from one another. For this

Parameter	Value
μ	1006.7e-6 MPa
κ	50e-3 MPa
λ	49.3e-3 MPa

TABLE I
MATERIAL PARAMETERS

reason we can parallelize over elements and nodes and not over time steps. The most suitable portions of code to implement in parallel are: obtaining the global force matrix (steps 4-8) using element-level granularity, updating of displacements (step 9) and loading for the next step (10), both using node-level granularity.

The granularity of parallelization in the computation of force contributions in the global force matrix assembly is one element, meaning that each thread run by the hardware computes the values of force for one element. The interconnectivity of elements mandates storage problems when summing force contributions on a node from surrounding elements in a parallel setting. A reduction scheme has been implemented for this reason that avoids the serialization of storage writes.

In the second kernel, we obtain per-node displacements using coefficients computed in the pre-computation phase. Using the central difference method we compute future displacements of nodes based on the global force matrix and the pre-computed coefficients in step 3. Here, the granularity is one node.

The last kernel executes the loading of the nodes under boundary conditions. Much like the previous kernel, it is executed per-node and it is a pleasingly parallel task due to no communication requirements between nodes.

An Abaqus finite element simulation code was used to validate the newly implemented code. Identical boundary conditions and smooth loading curve were used in both solution regimes, as well as the material parameters (I). The Abaqus jobs were run on a system with an Intel i7-3740QM processor and 4GB of RAM.

III. RESULTS

The results obtained through simulation in Abaqus/Explicit are regarded as benchmark. The accuracy of the newly implemented algorithm was evaluated by comparing the displacement vector in the XY plane of one top node at the corner. For all simulations the deviation from Abaqus solutions never exceeds 0.3%. The initial configuration and the loaded, converged configuration are symmetric with respect to all orthogonal planes and vertical edges remain vertical.

The results in Figure 2 show a speed up factor of TLED versus Abaqus on a per-time step basis. The total solution times speed up are proportional to these values and range from 1 second to 1h22m40s for Abaqus and from 0.2s to 50s for the parallel solver on the fastest K20c device.

IV. CONCLUSION

We have presented an integral part of a computer integrated system for monitoring of any mechanical stress-related

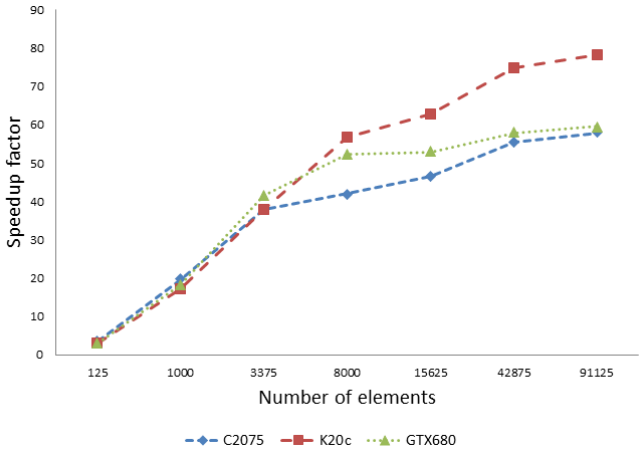

Fig. 2. Speed up factors versus Abaqus as a function of model element density

variable in patient tissues in an intraoperative setting. The nonlinear Finite Element algorithm employed was described in detail and results on a single loading scenario of a cube using a range of element densities and execution GPGPUs was presented.

The results show significant speedups and good accuracy in comparison to stable and reliable solutions. This work, therefore, encourages the prospect of the described technology soon becoming implemented in the modern surgical theater.

ACKNOWLEDGMENT

This research was supported by the EUs Seventh Framework Programme under GA No. 601021 (CASCADE). The authors would like to thank Prof. K.Miller and Dr. G.Joldes for considerable instruction and assistance with the details of the TLED algorithm.

REFERENCES

[1] K. Miller, G. Joldes, D. Lance, and A. Wittek, "Total lagrangian explicit dynamics finite element algorithm for computing soft tissue deformation," *Communications in Numerical Methods in Engineering*, 2007.

[2] N. Famaey, J. Vander Sloten, and E. Kuhl, "A three-constituent damage model for arterial clamping in computer-assisted surgery." *Biomech Model Mechanobiol*, Mar 2012. [Online]. Available: http://dx.doi.org/10.1007/s10237-012-0386-7

[3] Nvidia, *CUDA C Programming Guide 4.2*, Nvidia Corporation, 2012. [Online]. Available: http://developer.nvidia.com/nvidia-gpu-computing-documentation

[4] T. J. Hughes, *The Finite Element Method - Linear Static and Dynamic Finite element Analysis*. Dover publications, Inc., 1987.

[5] G. R. Joldes, A. Wittek, and K. Miller, "Suite of finite element algorithms for accurate computation of soft tissue deformation for surgical simulation." *Med Image Anal*, vol. 13, no. 6, pp. 912–919, Dec 2009. [Online]. Available: http://dx.doi.org/10.1016/j.media.2008.12.001

[6] T. Belytchko and T. J. Hughes, Eds., *Computational methods for transient analysis*, 1983.