Real-time Computer Simulation of Three Dimensional Elastostatics using the Finite Point Method

Kirana Kumara P, Ashitava Ghosal
Centre for Product Design and Manufacturing
Indian Institute of Science
Bangalore – 560 012, India
e-mail: kiranakumarap@gmail.com

Abstract—**Real-time simulation of deformable solids is essential for some applications such as biological organ simulations for surgical simulators. In this work, deformable solids are approximated to be linear elastic, and an easy and straight forward numerical technique, the Finite Point Method (FPM), is used to model three dimensional linear elastostatics. Graphics Processing Unit (GPU) is used to accelerate computations. Results show that the Finite Point Method, together with GPU, can compute three dimensional linear elastostatic responses of solids at rates suitable for real-time graphics, for solids represented by reasonable number of points.**

Keywords-real-time; simulation; elastostatics; solid; FPM

I. INTRODUCTION

Many virtual reality systems require the realistic and real-time graphical simulation of three dimensional deformable solids, e.g., a surgical simulator capable of simulating laparoscopic surgery needs a realistic simulation of liver. The quality of these simulations depend on how close the geometry of the solid is to the actual geometry, how closely the chosen material model represents the actual material, and how accurate is the numerical technique used for computations. Computations have to be made in real time and since, for real-time graphics, around 30 computations have to be completed in one second [1], real-time computations can take a maximum of only around 0.03 seconds to complete. One cannot compromise on speed but to achieve that speed one often has to compromise on the geometric description and/or material model and/or numerical technique. The following few paragraphs of this section give information on these approximations related to the present work. Of course, through the approximations, one effectively sacrifices a bit of realism for real-time performance.

Representing a three dimensional solid by a finite set of points belonging to the solid is a very simple way of representing the solid. This does not require any connectivity information and the representation allows the use of meshfree numerical techniques for discretization. In the present work, solids are represented by just points (or nodes).

As far as material models are concerned, physically based models such as those based on continuum mechanics are preferred for better realism [2]. Three dimensional linear elastostatics is the most basic of all three dimensional deformable solid idealizations. In many cases (e.g., simulating a machining process), this model alone may give acceptable accuracy since in many cases of modeling cutting, material to be cut is assumed linear elastic (e.g., [3]). But for some other applications (e.g., simulating a laparoscopic surgery), one has to at least consider geometric nonlinearity, as has been done in [4]. However, even in these cases, adopting the 3D linear elastostatic model is the first step towards realistic simulations, and many a times, this may provide a better realism when compared to a model which is not physically based. The 3D linear elastostatic model is used in the present work. Adopting this model has made it possible to perform the computations in real-time.

Many numerical techniques are available for solving three dimensional linear elastostatics of solids, the Finite Element Method (FEM) being the most popular. FEM needs a meshing of the solution domain and meshes need connectivity information. The connectivity creates bottlenecks while parallelizing computations, but parallel computing is required to achieve high performance demanded by real-time systems. Hence, meshfree methods [5, 6], which do not need any mesh, are more appropriate for real-time systems. Out of many meshfree methods found in the literatute, some are slow but very accurate (e.g., Element Free Galerkin Method (EFG), Reproducing Kernel Particle Method (RKPM)), others being faster but less accurate (e.g., Smoothed Particle Hydrodynamics (SPH), Finite Point Method (FPM)). The Finite Point Method (FPM) [7] is chosen for the present work because it is inherently fast. For many applications, including delicate applications like laparoscopic surgery simulation, accuracy is not too important, since humans cannot sense very small errors in real-time responses of solids [8]. The FPM has already been used to solve 3D elastostatics [7], but the present work shows that the method, accelerated with Graphics Processing Unit (GPU), is suitable for simulating real-time 3D linear elastostatics.

II. THREE DIMENSIONAL LINEAR ELASTOSTATICS

The governing differential equations for 3D linear elastostatics are given by [9]:

$$(\lambda+\mu)\left\{\frac{\partial^2 u_x}{\partial x^2}+\frac{\partial^2 u_y}{\partial x \partial y}+\frac{\partial^2 u_z}{\partial x \partial z}\right\}+\mu\left\{\frac{\partial^2 u_x}{\partial x^2}+\frac{\partial^2 u_x}{\partial y^2}+\frac{\partial^2 u_x}{\partial z^2}\right\}=0 \tag{1}$$

$$(\lambda+\mu)\left\{\frac{\partial^2 u_y}{\partial y^2}+\frac{\partial^2 u_x}{\partial x \partial y}+\frac{\partial^2 u_z}{\partial y \partial z}\right\}+\mu\left\{\frac{\partial^2 u_y}{\partial x^2}+\frac{\partial^2 u_y}{\partial y^2}+\frac{\partial^2 u_y}{\partial z^2}\right\}=0 \tag{2}$$

$$(\lambda+\mu)\left\{\frac{\partial^2 u_z}{\partial z^2}+\frac{\partial^2 u_x}{\partial x \partial z}+\frac{\partial^2 u_y}{\partial y \partial z}\right\}+\mu\left\{\frac{\partial^2 u_z}{\partial x^2}+\frac{\partial^2 u_z}{\partial y^2}+\frac{\partial^2 u_z}{\partial z^2}\right\}=0 \tag{3}$$

where $\lambda=\frac{\nu E}{(1+\nu)(1-2\nu)}$ $\quad \mu = \frac{E}{2(1+\nu)}$

Here, u_x, u_y and u_z are the displacements along x, y and z directions; E is the Young's modulus and ν is the Poisson's ratio.

III. The Finite Point Method for Three Dimensional Linear elastostatics

A brief summary of the method that is explained in detail in [7], is given here. Displacements u_x, u_y and u_z in the governing equations are approximated by:

$$u(x) \cong \hat{u}(x) = \sum_{l=1}^{m} p_l(x)\alpha_l = \boldsymbol{p}(x)^T \boldsymbol{\alpha} \tag{4}$$

where $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, ...\alpha_m]^T$

Assuming the base interpolating functions to be quadratic, $\boldsymbol{p}$ is given by:

$$\boldsymbol{p} = [1, x, y, z, xy, yz, zx, x^2, y^2, z^2]^T$$

Hence, m=10 (total number of elements of $\boldsymbol{p}$).

To find **α**, a set of neighboring nodes (total number = n) is considered. In FPM, n should be greater than m. Function $u(x)$ is sampled at n points giving:

$$\boldsymbol{u}^h = \begin{Bmatrix} u_1^h \\ u_2^h \\ \vdots \\ u_n^h \end{Bmatrix} \cong \begin{Bmatrix} \hat{u}_1 \\ \hat{u}_2 \\ \vdots \\ \hat{u}_n \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{p}_1^T \\ \boldsymbol{p}_2^T \\ \vdots \\ \boldsymbol{p}_n^T \end{Bmatrix} \boldsymbol{\alpha} = \boldsymbol{C}\boldsymbol{\alpha} \tag{5}$$

Since, $n > m$, approximation cannot fit nodal values. This problem is overcome by determining the $u(x)$ values by minimizing the sum of the square distances, with respect to the **α** parameters, of the error at each point weighted with a function φ as:

$$J = \sum_{j=1}^{n} \varphi(x_j)(u_j^h - \hat{u}(x_j))^2 = \sum_{j=1}^{n} \varphi(x_j)(u_j^h - \boldsymbol{p}_j^T \boldsymbol{\alpha})^2 \tag{6}$$

The weight function φ is taken as:

$$\varphi(r) = \frac{\exp(-r^2/c^2) - \exp(-r_m{}^2/c^2)}{1-\exp(-r_m{}^2/c^2)}$$

where $c = 0.25\, r_{\max}$ $\quad r_m = 2r_{\max}$

$r_{\max}$= distance to the farthest node, out of n nodes

r = distance to nodes (contained in n)

Standard minimization of (6) gives **α**.

IV. Test Problems and Results

The test solid is a square prism which is 99 millimeters long and has (4 millimeters x 4 millimeters) cross section. Young's modulus is taken as 200000 N/mm^2 and the Poisson's ratio is taken as 0.33 (These values are the same as those for steel). In the present work, FPM is used to discretize the 3D domain itself; domain is not idealized as a 1D or 2D domain. It is a standard practice in such cases to have at least 4 points along each dimension, to properly describe the domain in 3D. Domain discretization for the present work uses (5 x 5) points to describe the cross section, and uses 100 points along the length direction. Thus the solid is represented by (5 x 5 x 100) = 2500 points. Even complicated geometries such as a liver may satisfactorily be described by 2500 points.

Now, three sets of boundary condition are applied on the solid, giving rise to three different problems to be solved. For the first problem (Problem 1), one end of the prism shaped solid is fixed and the other end is given a known lateral displacement of 5 *mm*. For the second problem (Problem 2), both the ends are fixed and a 5 *mm* lateral displacement is given to a point that is located at exactly half the length of the beam and that is also located on the neutral axis. As a third problem (Problem 3), one end is fixed and the other end is given an axial displacement of 5 *mm*. One can see that the thickness to length ratio for the solid considered is around 1/25, and hence, one can use the standard analytical formulae applicable to beams and bars [10] to get approximate analytical solutions for all the three problems considered, and one can compare the solutions from FPM with these analytical solutions. All the three test problems contain only Dirichlet boundary conditions, since this is the case with graphical simulations; even in simulations with haptic feedback, forces are calculated from displacement fields, not vice versa. In the present work, MATLAB [11] is used to develop FPM codes.

Fig. 1, Fig. 2 and Fig. 3 show the displacements along the neutral axis of the solid for Problem 1, Problem 2 and Problem 3 respectively. Vertical axes represent displacements. Horizontal axes represent length of the prism shaped solid. Horizontal and vertical axis has different scales. Numbers on the horizontal axis just represent numbers corresponding to the global degrees of freedoms and do not have any significance in figures.

For Problem 1 and Problem 2, solution is calculated using FPM for both n=50 (plotted with dotted lines) and n=250 (plotted with dashed lines). It can be observed that the solutions are almost the same irrespective of the value of n, thus confirming that n (number of neighbors) does not have much effect on solutions over a range. Analytical (or theoretical) solutions are also plotted (with solid lines) along with the solutions from FPM.

For Problem 3, analytical solution is just a straight line and has not been plotted (only solution from FPM is plotted using solid line).

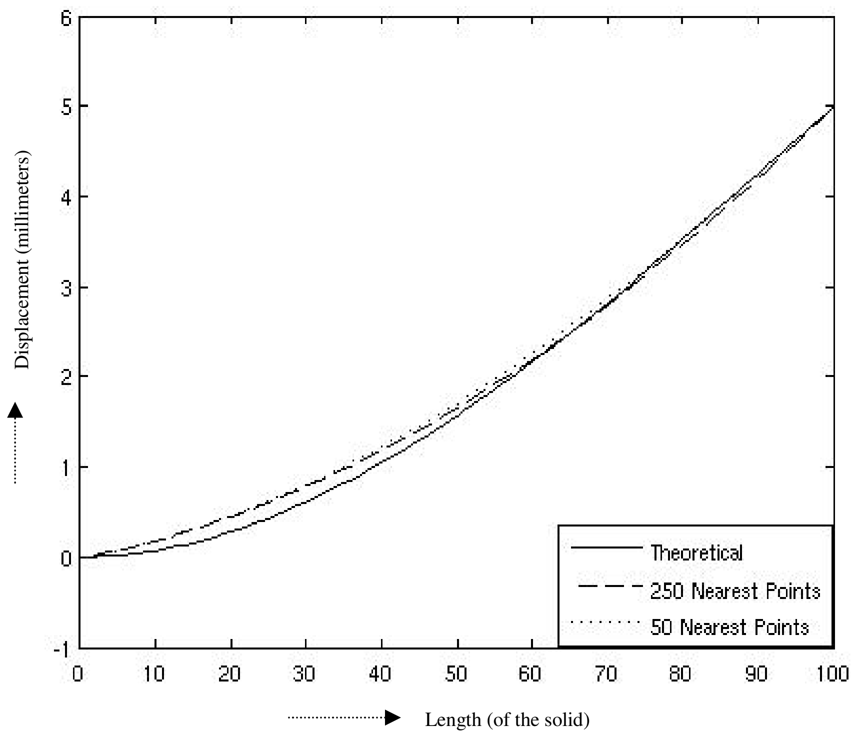

Figure 1. Displacement along neutral axis for Problem 1

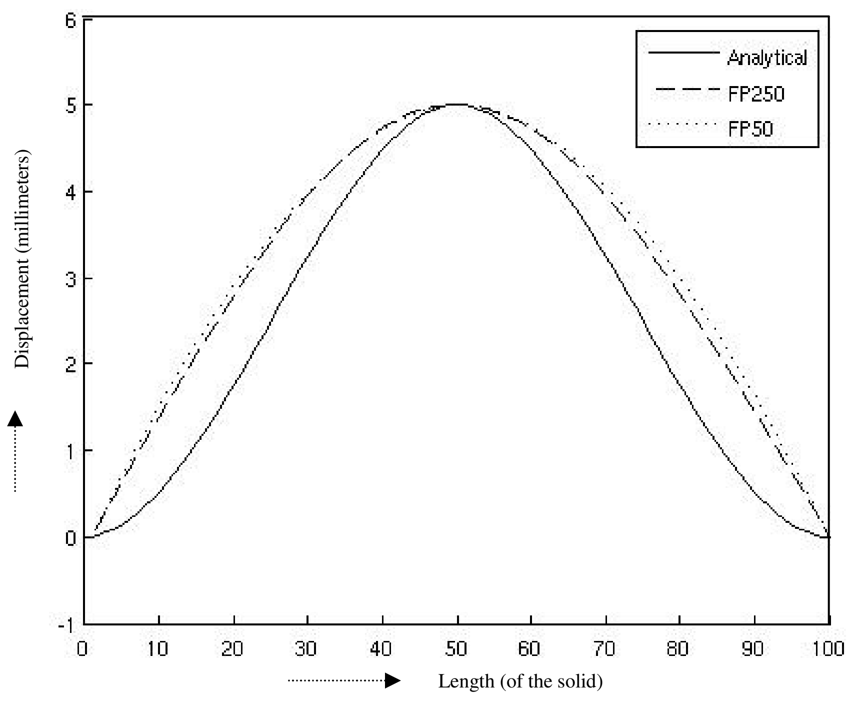

Figure 2. Displacement along neutral axis for Problem 2

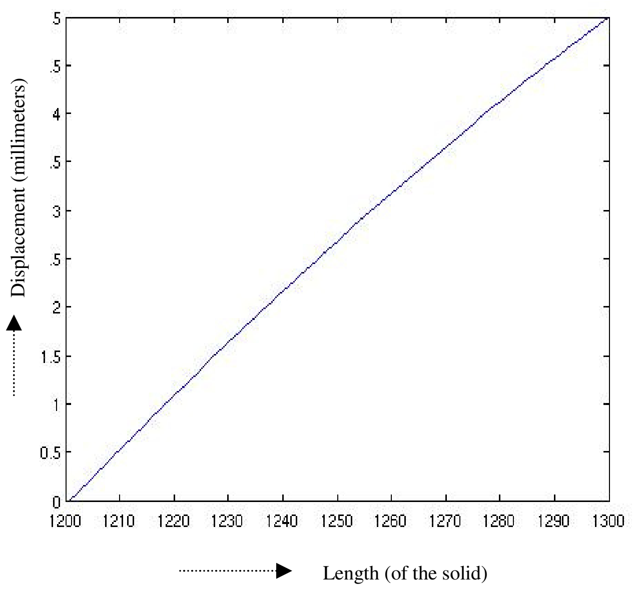

Figure 3. Displacement along neutral axis for Problem 3

From figures, analytical solutions and the solutions from FPM agree very well except in Fig. 2. The slight deviation in Fig. 2 is because, using FPM makes only the displacements at the two ends of the solid zero, while the analytical solution makes the slopes at the two ends also equal to zero.

Coming to solution times, a GPU is used to accelerate computations. The GPU used is NVIDIA GeForce GTX 460. GPU is used from within MATLAB using GPUmat [12], a GPU toolbox for MATLAB. Table 1 shows the solution time with only CPU and with only GPU. Table also gives the time for data transfer from CPU to GPU and back from GPU to CPU. Solution times are the same for all three problems considered (all have exactly the same number of degrees of freedom).

One can see that with a GPU, one can get real-time performance (i.e., 50 computations per second, as against the required 30 computations per second). But one can observe that, if one considers the data transfer from CPU to GPU and vice versa, GPU needs more time than the CPU.

V. CONCLUDING REMARKS

Results show that, for applications which can manage with the 3D linear elastostatic idealization of solids, the Finite Point Method, accelerated with GPU, can simulate solids at real-time. The accuracy is seen to be satisfactory.

Present work attempts to simulate only real-time graphics (which needs around 30 computations per second). But for some applications, simulating haptic feedback (which needs around 1000 computations per second [1]) may also be desirable to make the simulations more realistic. Computational speed achieved in the present work is not sufficient for simulating real-time haptics. One can also observe that the time required to transfer the data from CPU to GPU and vice versa is a bottleneck while using GPUs, and in a real implementation, one has to be careful to implement a proper GPU algorithm which minimizes this transfer.

Future work aims to include geometric nonlinearity and nonlinear material models. Aim is to design and implement customized GPU algorithms which can simulate nonlinear solids at rates suitable for real-time haptics.

TABLE I. SOLUTION TIMES

Solution Time (seconds)		**Data Transfer Time (seconds)**
CPU only	***GPU only***	
0.066	0.020	0.194

REFERENCES

[1] OPENHAPTICS™ TOOLKIT version 2.0, PROGRAMMER'S GUIDE, SensAble technologies, 2005

[2] U. Meier, O. Lopez, C. Monserrat, M. C. Juan, M. Alcaniz, "Real-time deformable models for surgery simulation: a survey," Computer Methods and Programs in Biomedicine (2005) 77, 183-97

[3] Mohsen Mahvash and Vincent Hayward, "Haptic Rendering of Cutting: A Fracture Mechanics Approach," Vol. 2. No. 3., Haptics-e, 21-Nov-2001, http://www.haptics-e.org

[4] Yi-Je Lim, Suvranu De, "Real time simulation of nonlinear tissue response in virtual surgery using the point collocation-based method of finite spheres," Comput. Methods Appl. Mech. Engrg. 196 (2007) 3011–3024

[5] Shaofan Li and Wing Kam Liu, "Meshfree and particle methods and their applications," Appl Mech Rev vol 55, no 1, January 2002

[6] G R Liu and Y T Gu, An Introduction to Meshfree Methods and Their Programming, Springer,2005

[7] E. Onate, F. Perazzo, J. Miquel, "A finite point method for elasticity problems," Computers and Structures 79 (2001) 2151-2163

[8] Ottensmeyer M, Salisbury K, "In Vivo Data Acquisition Instrument for Solid Organ Mechanical Property Measurement," MICCAI: Medical Image Computing and Computer-Assisted Intervention 4th International Conference, Utrecht, The Netherlands, 2001, Springer-Verlag

[9] L S Srinath, Advanced Mechanics of Solids, Tata McGraw-Hill, Third Edition, 2009

[10] K. Lingaiah, Design Data Hand Book, McGraw Hill, 2ndEd., 2003

[11] The MATLAB website, http://www.mathworks.com/

[12] GPUmat: GPU toolbox for MATLAB, http://gp-you.org/