34th Annual International Conference of the IEEE EMBS
San Diego, California USA, 28 August - 1 September, 2012

Integration of a Multigrid ODE Solver into an Open Medical Simulation Framework

Xunlei Wu[1], Jianhua Yao[2], Andinet Enquobahrie[3], Huai-Ping Lee[3], and Michel A. Audette[4]

Abstract— **In this paper, we present the implementation of a Multigrid ODE solver in SOFA framework. By combining the stability advantage of coarse meshes and the transient detail preserving virtue of fine meshes, Multigrid ODE solver computes more efficiently than classic ODE solvers based on a single level discretization. With the ever wider adoption of the SOFA framework in many surgical simulation projects, introducing this Multigrid ODE solver into SOFA's pool of ODE solvers shall benefit the entire community. This contribution potentially has broad ramifications in the surgical simulation research community, given that in a single-resolution system, a constitutively realistic interactive tissue response, which presupposes large elements, is in direct conflict with the need to represent clinically relevant critical tissues in the simulation, which are typically be comprised of small elements.**

I. Introduction

The broad objective of medical simulation is to exploit computers in order to synthesize the tissue response to a simulated therapy. More specifically, surgery simulation synthesizes tissue response to simulated surgical tools, such as a mechanical response to virtual cutting or manipulation. The simulated tissue behavior is a trade-off between material fidelity and computational efficiency, which can be thought of as a spectrum. The first pole of this spectrum coincides with predictive simulation, which consists of highly accurate off-line computations used by expert surgeons. The opposite pole can be termed interactive simulation, and offers a means of training surgical residents, in order to improve their skill without risk to a real patient, by way of a haptic device. Increasingly, the research community has sought to bridge the conflicting requirements inherent in the two poles, leading to the development of new efficiencies. These efficiencies are based on either hardware acceleration, typically *graphical processor units* (GPUs) [1], or on novel algorithms, such as constitutively and geometrically nonlinear elasticity models that leverage preprocessing based on reference coordinates: the *Total Lagrangian Explicit Dynamics* (TLED) [2] and *Multiplicative Jacobian Energy Decomposition* (MJED) [3].

The development of the Simulation Open Framework Architecture (SOFA) [4] for Medical Simulation has not only greatly enhanced the research community's ability to develop new algorithms more quickly, but also serves as as an efficient prototyping tool [5]. SOFA has included a handful of ODE solvers already, such as *explicit Runge-Kutta*, and *implicit Euler*, to name two. All of these solvers can process complex couplings of objects each modeled with a single and different discretization, however none of them is able to exploit multiple resolutions of the same object. Despite the modular and multi-modality mapping features in SOFA which can improve visual detail by pairing a fine-resolution visual model with a coarse mechanical model, single-level based explicit or implicit ODE solvers simply cannot retain the integration stability and transient dynamic details at the same time.

A Multigrid ODE solver [6] has been proposed to accelerate the time integration of tissue deformation with interactive medical simulation application in focus. By combining the stability advantage of coarse meshes and the transient detail-preserving virtue of fine meshes, the Multigrid ODE solver provides more efficient computation than classic ODE solvers coupled with a single mesh representation. This paper describes the implementation details of such Multigrid ODE solver (*MultiGridSolver*) into the current SOFA public release, as well as several enhancements over the original solver proposed in [6].

II. Methodology

A. Broad Objective: a SOFA Multigrid ODE Solver class

Our goal is to realize a *MultiGridSolver* in SOFA as an ODE solver class. This implementation is consistent with the architecture of SOFA, which exploits separate models representing the deformable anatomy, which are specialized for the following aspects of the simulation:

- *Visualization*: typically a relatively dense triangular surface model.
- *Continuum mechanics*: usually a tetrahedral mesh model that is coarser than the visualization model.
- *Collision detection*: a number of schemes are supported, including a representation of an organ in terms of coarse spheres or a coarsely triangulated surface.

In current modular design of SOFA, an ODE solver accesses *MechanicalState* (container of state vectors) of the associated *MechanicalObject* at the current time stamp, aggregates all the forces applied to the deformable object, and then evolves the state to the next time stamp. It is important to understand that all existing SOFA's ODE solvers do not compute the forces, e.g. collision forces, body forces. They merely aggregate them as already computed by other Visitor instances invoked in the animation loop. Before demonstrating our implementation, we present background information to facilitate the understanding

[1]: X. Wu is currently employed at SAS, Cary, NC. His email address is xunlei.wu@sas.com.

[2]: J. Yao is currently employed at RAD&IS at NIH. His email address is jyao@cc.nih.gov.

[3]: A. Enquobahrie and H.P. Lee are employed by Kitware,. Their email addresses are andinet.enqu@kitware.com and huaiping.lee@kitware.com.

[4]: M.A. Audette is with the Dept. of Modeling, Simulation and Visualization Engineering, Old Dominion University, Norfolk, VA, 23529.

of the differences between *MultiGridSolver* and existing single-level SOFA ODE solvers.

B. MultiGridSolver's V-cycle

In constrast to single-level solvers included in SOFA, *MultiGridSolver* needs to access the states of several *MechanicalObject* instances. These *MechanicalObjects*, whose discrete mesh representations have different resolutions, are arranged in a sorted fashion. *MechanicalObject*$^{(0)}$ at level 0 represents the finest mesh whereas *MechanicalObject*$^{(M-1)}$ has the coarsest mesh for an M-level *MultiGridSolver* scheme. *MechanicalObject*$^{(0)}$ is used to change the appearance of the deformable anatomy's corresponding *VisualModel* and to respond to the anatomy's *CollisionModel*. The Multigrid solver passes information among multiple discretizations in a *V-cycle* fashion demonstrated in the right half of Figure 1.

Figure 1. Multigrid ODE Solver's RSV *V-cycle* in SOFA animation loop

When integrating $X_n^{(0)}$, *MechanicalState* of *MechanicalObject*$^{(0)}$, from time stamp n to $n+1$, the displacement information in $X_n^{(0)}$ and both the external and induced internal forces will propagate to and from the coarser levels, i.e. *MechanicalObject*$^{(i)}$ where $i \in [1, M-1]$.

MultiGridSolver's Restrict-State-and-Variation (RSV) scheme is chosen here to be implemented into SOFA due to its stability and convergence performance in comparison with a number of competing approaches [6].

In the beginning of each RSV *V-cycle*, *MultiGridSolver* first maps the state $X_n^{(i-1)}$ of each degree-of-freedom (DOF) of *MechanicalObject*$^{(i-1)}$to the next coarser level's state $X_n^{(i)}$ by either *restricting* $X_n^{(i-1)}$ directly or *updating* $X_n^{(i)}$ with the restricted version of $V_n^{(i-1)} \equiv X_n^{(i-1)} - X_{n-1}^{(i-1)}$, the *variation* of the corresponding DOF at time stamp n and level $i-1$. Please refer to [6] for the details on the switching between state mapping and variation mapping. Once the coarsest level is reached, $X_n^{(M-1)}$ can be thought as the ***restricted version*** of $X_n^{(0)}$. RSV then computes and aggregates the forces on $X_n^{(i)}$ before ODE integration to time stamp $n+1$.

The second stage of the RSV scheme interpolates the coarse mesh state to fine meshes. In a manner similar to the restriction switching mechanism, each DOF's $X_n^{(i-1)}$ is either an ***interpolation*** of $X_{n+1}^{(i)}$ or updated with the interpolation of $V_{n+1}^{(i)}$. Finally $X_n^{(0)}$ becomes the interpolated version of $X_{n+1}^{(M-1)}$ and is integrated to $X_{n+1}^{(0)}$ by the end of each *MultiGridSolver* loop. Apart from the standard interface of existing SOFA ODE solver class, every *V-cycle* requires 3 more components:

1. The ***state*** at both time stamps *n* and *n-1* of a series of *MechanicalObject*$^{(i)}$.
2. The ***mapping*** between adjacent-level Mechanical-Object meshes.
3. The capability to trigger the computation of ***external forces***, e.g. penalty forces, due to collision at every level.

The remaining subsections will illustrate how our implementation incorporates the above extra information into SOFA.

C. Accessing multiple MechanicalObject states

In our implementation, *MultiGridSolver* embeds several SOFA *Node*'s. Each *Node* contains a Mechani- calObject as well as MeshLoader, MeshTopology, FEMForceField, UniformMass, ODESolver, and elements needed by CollisionPipeline. Then *MechanicalObject*$^{(0)}$ and CollisionPipeline$^{(0)}$ are referenced in the main SOFA tree, which also contains the deformable object's visual model, e.g. *OglModel*, and of course, the ODESolver, i.e. *MultiGridSolver* . Figure 2 illustrates the multiple *Node* instances embedded in *MultiGridSolver* as well as the references of top level of *MechanicalObject* and CollisionPipeline in the SOFA main tree. At every step, SOFA's main animation visitor will invoke *MultiGridSolver*::solve(), where the sequential restriction and interpolation will be invoked. Since each *Node* in *MultiGridSolver* contains ODESolver as well, the time integration of each *MechanicalObject*$^{(i)}$ will be performed in *MultiGridSolver::solve()* as well.

D. Mapping adjacent meshes

The meshes of *MechanicalObject*$^{(i)}$ and MechanicalObject$^{(i-1)}$ are mapped with each other through SOFA's MeshBarycentricMapperEngine during the initialization of *MultiGridSolver*, i.e. Multi- GridSolver::init(). Such mapping associates every vertex's displacement on one mesh to a weighted sum of all the corner vertices' displacement of an element in another mesh during *MultiGridSolver V-cycle*. Since the mapping has to be performed before the SOFA main animation loop starts, after each *MultiGridSolver Node* and its child objects are instantiated, *Node::init()* has to be called before calling *MultiGridSolver:: init()*. This approach is unconventional with respect to existing SOFA applications where *init()* method of all instantiated *Node*s and child objects are invoked implicitly through the root *Node's init()* by the end of main function. This is because all the objects in the existing SOFA applications can be traversed from the main tree and there is no cross-node coupling.

E. Resolving contacts within MultiGridSolver

The contact resolution requirement has proved to be the most difficult to resolve in the current SOFA architecture. A SOFA ODE Solver's solve() method only aggregates the already computed modular forces. The collision detection and the induced collision forces have been calculated by CollisionPipeline visitor prior to an ODE solver's solve() being called. Secondly, in the ascending portion of *MultiGridSolver V-cycle*, once $X_n^{(i-1)}$ is updated given $X_{n+1}^{(i)}$ and $V_{n+1}^{(i)}$, the contact between Mechanical- Object$^{(i-1)}$ and the rest of the contact bodies in a SOFA simulation has to be resolved. Users have to develop a custom AnimationLoop in order to support such interaction within an ODE solver class in the current SOFA release. The main idea behind our approach is to carefully arrange all the objects both in the main simulation tree and each *Node* embedded in *MultiGridSolver* illustrated in Figure 2.

Fundamentally, *Node*$^{(i)}$ has to contain objects embodying a collision pipeline in addition to objects describing a deformable object. Also, the references of other objects' collision models in the SOFA scene have to be registered in *Node*$^{(i)}$. Recall that *Node*$^{(i)}$ has its own (standard) ODE solver, e.g. (ex/implicit) Euler solver. The main simulation tree copies the same tree structure as *Node*$^{(0)}$ of *MultiGridSolver* except that the reference to ODESolver is replaced by an instance of Multi- GridSolver class. Note that the objects instantiated within *MultiGridSolver* child nodes are not duplicated but rather referenced. By customizing *MultiGrid-Solver::solve()* and accessing *CollisionPipeline*$^{(i)}$, *Node*$^{(i)}$ can now perform collision detection against current mechanical states, aggregate the collision response and other forces, and launch that level's

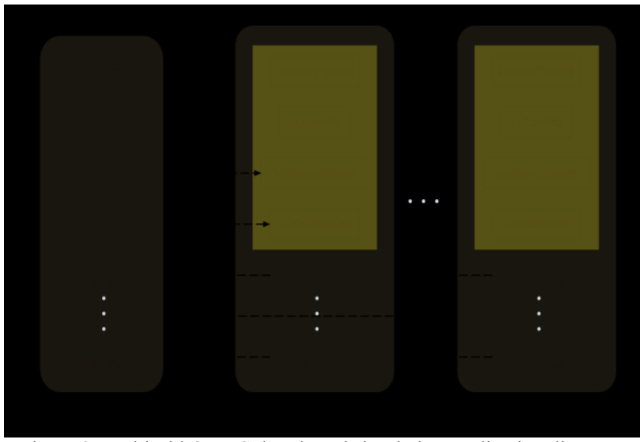
Figure 2. Multigrid ODE Solver based simulation application diagram.

ODESolver::solve() in order to update the state to the next time stamp. The advantages of this approach are that it offers the least intrusion into existing SOFA architecture and the most flexible. One disadvantage of this setup is one extra collision detection performed in the main animation loop before *MultiGridSolver::solve() is* invoked.

F. Improvements over the original Multigrid algorithm

Along with our implementation based on SOFA, we have improved the original Multigrid solver in several areas:

1. Selecting different ODE solvers for different levels - The original Multigrid ODE solver deployed the same ODE solver at every level, e.g. explicit Euler. Despite its simplicity, it was less flexible to be customized. In the implementation presented in this paper, each level can have different ODE solvers. One example is applying explicit Euler from level *0* to *M-2* and implicit Euler at the coarsest level *M-1*. While the coarsest level achieves unconditional stability, the explicit methods in fine levels improve the transient details of the deformation.
2. Applying different time steps for different levels - In the original algorithm, an *M* level *MultiGridSolver* with step size t yields every level's *ODESolver* to have equal step size *t/M.* In conjunction with the previous improvement, the user now can assign arbitrary distributions of step sizes to different levels provided the partition of unity of these distribution factors. As an example, define $N^{(i)}$ as the count of elements at level i. Assuming every level uses a uniform mesh, the higher the count of elements, the smaller the elements get, and thus smaller step size is needed for the stability of an explicit ODE solver. The step size at level *i* is

$$t^{(i)} = \frac{1/N^{(i)}}{\sum(1/N^{(i)})} t \qquad (1)$$

The user can certainly deploy more elaborated measure of element size and stiffness than this naïve approach to control the step size distributions. In general, the finer mesh gets a smaller step size. If *ODESolver*$^{(M-1)}$ is implicit, the user can assign bigger portion of *t* to the coarsest level to further improve the stability of the hybrid *MultiGridSolver* who combines both explicit solvers on fine levels and implicit solvers on the coarsest level. Certainly this approach would reduce the transient details captured.

III. Results

We have tested our 4-level *MultiGridSolver* against a single level ODE solver in SOFA public release 1.0 beta4 on an Intel Core i7-2720QM laptop in Debug mode. Figure 3 is a screenshot of the application running in SOFA GUI where *OglModel* and *CollisionModel* are triangular mesh of a ball. The surface mesh is mapped to *MechanicalObject* who is based on multiple tetrahedron meshes of the same cube geometry. The ball drops under gravity and collides with bowl-shaped fixed floor.

The reference case applies a single level ODE solver on the finest mesh used by *MultiGridSolver*. The performance results are summarized in Table 1 and 2. When using *MultiGridSolver* as a pure explicit ODE solver, it is more efficient than single level based ODE solver. *MultiGridSolver* uses 50% amount of time of its single- level explicit Euler brother to simulate the scene to the same length of time. Although the hybrid version of *MultiGridSolver* cannot compete with single-level implicit Euler in terms of stability, it takes 50% less amount of time to finish each solve() call. In addition, the hybrid *MultiGridSolver* introduces less artificial damping as well because of the fine level's explicit ODE solvers. In the example recorded in Table 2, 50% of the total portion of time step assigned to level 0 to 2 computed from eq. (1) has been assigned to level 3. Namely, the coarsest level's implicit Euler solver simulates a step size of 17ms while level 0 to 2 explicit Euler solvers simulate a total step size of 6ms. By applying 17ms step size to Single-level

ODESolver case with implicit Euler, it will need (1000/17)(1/3.5)=16.807sec CPU time to finish 1sec of the simulated scene with comparable amount of artificial damping than the hybrid *MultiGridSolver* counterpart.

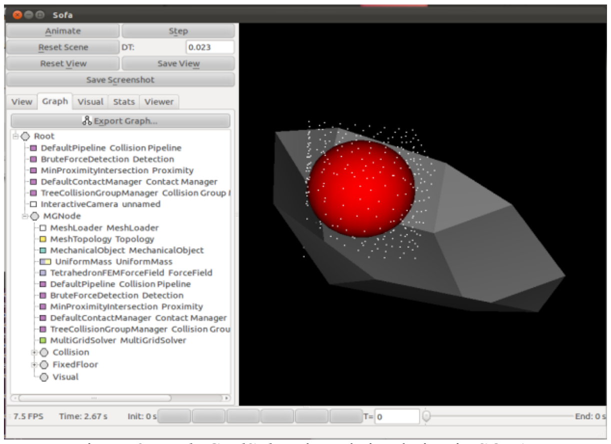
Sofa
Animate
Step
Reset Scene
DT:
0.023
Reset View
Save View
Save Screenshot
View
Graph
Visual
Stats
Viewer
Export Graph...
Root
DefaultPipeline Collision Pipeline
BruteForceDetection Detection
MinProximityIntersection Proximity
DefaultContactManager Contact Manager
TreeCollisionGroupManager Collision Group
InteractiveCamera unnamed
MGNode
MeshLoader MeshLoader
MeshTopology Topology
MechanicalObject MechanicalObject
UniformMass UniformMass
TetrahedronFEMForceField ForceField
DefaultPipeline Collision Pipeline
BruteForceDetection Detection
MinProximityIntersection Proximity
DefaultContactManager Contact Manager
TreeCollisionGroupManager Collision Grou
MultiGridSolver MultiGridSolver
Collision
FixedFloor
Visual
7.5 FPS
Time: 2.67 s
Init: 0 s
T= 0
End: 0 s

Figure 3. *MultiGridSolver* based simulation in SOFA.

	MultiGridSolver	Single-level ODESolver
Total Element Count	2570+1428+950+529= 5477	2570
Fine Level ODESolver	Explicit Euler	Explicit Euler
Coarsest Level ODESolver	Explicit Euler	N/A
Maximum Stable Step Size	5ms	1ms
Average FPS	14	34
CPU Time to simulate 1 second	(1000/5)(1/14)=14.286 sec	(1000/1)(1/34)=29.412 sec

Table 1. Explicit *MultiGridSolver* performance

	MultiGridSolver	Single-level ODESolver
Total Element Count	2570+1428+950+529= 5477	2570
Fine Level ODESolver	Explicit Euler	Implicit Euler
Coarsest Level ODESolver	Implicit Euler	N/A
Maximum Stable Step Size	23ms	Any
Average FPS	7	3.5
CPU Time to simulate 1 second	(1000/23)(1/7)=6.211 sec	(1000/1000)(1/3.5)=0.286 sec

Table 2. Hybrid *MultiGridSolver* performance

IV. Discussion and Future Work

This paper has presented work on the integration of a Multigrid ODE solver into the open-source platform. Our results represent a proof-of-concept of the feasibility of representing an object at different resolutions and using these various levels to collaboratively and stably estimate deformations, within the open-source SOFA platform.

The recasting of surgery simulation, in terms of deformations applied to anatomical models of progressively finer resolution and smaller spatial extent, coinciding with the pathology, the surgical corridor to it, and neighboring critical tissues, will have profound impact on the simulation community. This approach is consistent surgical ontologies based on *neurosurgical approach* and *pathology* type, which naturally leads to smaller subvolumes of clinical interest, coinciding with medium- and fine-resolution meshes. Last, this approach leads to multi-GPU massive parallelization.

Last, our *MultiGridSolver* implementation integrates multiple *MechanicalObjects* during its setup. Since this is different from standard SOFA ODE Solver class interface, *MultiGridSolver* is not compatible with the current SOFA scripting. A script-based implementation of *MultiGridSolver*, in addition to the C++, is underway.

Acknowledgment

The authors would like to express heartfelt gratitude to Prof. François Fauré, Université Joseph Fourier, Grenoble, France and Dr. Jérémie Allard, INRIA, Lille, France for their invaluable suggestions and advices. This research was supported by NIH grant 1 R43 NS067742-01.

References

1. *High-speed nonlinear finite element analysis for surgical simulation using graphics processing units.* **Taylor Z, et al** 2008 May, IEEE Trans Med Imaging., pp. 27(5):650-63.
2. *Total Lagrangian explicit dynamics finite element algorithm for computing soft tissue deformation.* **K. Miller, et al.,** 2007, Comm. Numer. Meth. Eng., 23(2), pp121–134.
3. *Multiplicative Jacobian Energy Decomposition method for fast porous visco-hyperelastic soft tissue model.* **Marchesseau S et al.** Beijing : Springer, 2010. Med Image Comput Comput Assist Interv. pp. 13(Pt 1):235-42.
4. **SOFA.** SOFA, Simulation Open Framework Architecture. [Online] http://www.sofa-framework.org/home.
5. *SOFA - an Open Source Framework for Medical Simulation.* **J. Allard et al.** 2007. Medicine Meets Virtual Reality (MMVR'15). pp. pages 13-18.
6. *Multigrid Integration for Interactive Deformable Body Simulation.* **X. Wu, F. Tendick.** Cambridge, MA : Springer, 2004. Int. Symp. Med. Sim.. pages 92-104.