Proceedings of the ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference
IDETC/CIE 2013
August 4-7, 2013, Portland, Oregon, USA

DETC2013-13239

CHRONO: A PARALLEL PHYSICS LIBRARY FOR RIGID-BODY, FLEXIBLE-BODY, AND FLUID DYNAMICS

Toby Heyn
Hammad Mazhar
Arman Pazouki
Daniel Melanz
Andrew Seidl
Justin Madsen
Aaron Bartholomew
Dan Negrut
Simulation Based Engineering Lab
Department of Mechanical Engineering
University of Wisconsin
Madison, WI, 53706
Email: heyn@wisc.edu

David Lamb
US Army TARDEC
Warren, MI 48397
Email: david.lamb@us.army.mil

Alessandro Tasora
Department of Industrial Engineering
University of Parma
V.G.Usberti 181/A, 43100, Parma, Italy
Email: tasora@ied.unipr.it

ABSTRACT

This contribution discusses a multi-physics simulation engine, called **Chrono**, that relies heavily on parallel computing. **Chrono** aims at simulating the dynamics of systems containing rigid bodies, flexible (compliant) bodies, and fluid-rigid body interaction. To this end, it relies on five modules: equation formulation (modeling), equation solution (simulation), collision detection support, domain decomposition for parallel computing, and post-processing analysis with emphasis on high quality rendering/visualization. For each component we point out how parallel CPU and/or GPU computing have been leveraged to allow for the simulation of applications with millions of degrees of freedom such as rover dynamics on granular terrain, fluid-structure interaction problems, or large-scale flexible body dynamics with friction and contact for applications in polymer analysis.

Introduction

About 50% of all traded products worldwide are in granular form. Grain, rice, sugar, coffee, cereal, salt, sand, drug pills and the constituents of a pill, animal feed pellets, and fertilizers are examples of granular material. Granular material and its storing/packing/motion come up in the design of a combine, the mobility of a Mars rover, avalanche dynamics, earthquakes, and the formation of asteroids and planets. Although granular problems are so pervasive, characterizing the dynamics of this medium in real-life applications remains an open problem. The problem of handling granular material becomes even more challenging when they form a suspension carried out in a flowing liquid. This contribution summarizes an effort aimed at addressing these and other questions such as: how is the shape/geometry of the bodies flowing in a liquid determine their time evolution? What happens when these floating bodies are long and flexible, which is effectively the case in polymer simulation?

To the best of our knowledge, there is no commercial or open source solution capable of characterizing through simulation the time evolution of such systems. These problems are governed by very large sets of ordinary differential equations and/or differential algebraic equations and/or partial differential equations. There are many open questions in the modeling stage; i.e., in the very process of formulating these equations. Moreover, there are numerous difficult questions in relation to their numerical solu-

tion. Attempts to answer these questions by this team have been summarized elsewhere [1, 2, 3]. In this contribution the emphasis is placed on discussing how parallel computing has allowed us to increase the size of the problems tackled by direct numerical simulation to levels that one decade ago would have seemed intractable. The paper is organized as follows: **Chrono**::Rigid presents implementation details related to the rigid body dynamics simulation engine; this is the most mature component of **Chrono** and and draws on both GPU and CPU parallel computing. Section **Chrono**::Flex gives a high level perspective on the **Chrono** component that formulates and solve the equations of motion associated with large collections of flexible bodies that can potentially interact through friction and contact. Section **Chrono**::Fluid outlines the framework that supports the simulation of fluid-solid interaction problems. Since some of the problems analyzed have millions of components, rendering high quality animations to present the time evolutions of these systems can be prohibitively long. Section **Chrono**::Render discusses a rendering pipeline that can leverage up to 320 instances of a commercial renderer to generate in parallel feature-rich images. Conclusions, directions of future work, and information about the availability of **Chrono** round up the paper.

Chrono::Rigid

Chrono::Rigid is a general-purpose simulation environment for three-dimensional problems with many rigid bodies [3]. This environment supports the simulation of very large systems common to those encountered in granular dynamics, where millions of objects can be interacting at any given time. Applications where this tool could be used include the simulation of tracked vehicles on granular terrain [4] or a rover operating on granular soil made up of discrete particles. In these applications granular terrain is modeled as a collection of millions of discrete bodies which interact through contact, friction and impact. Additionally these systems contain complex mechanisms composed of joints and rigid bodies. **Chrono**::Rigid was initially developed using the Differential Variational Inequality (DVI) formulation as an efficient way to deal with problems that contain many objects interacting through friction and contact - a typical bottleneck for other types of formulations [5,6]. **Chrono**::Rigid has since been extended to support the Discrete Element Method (DEM) formulation for solving granular dynamics problems with friction and contact [7, 8].

Using MPI for distributed Chrono::Rigid

This is an extension of the **Chrono** environment to leverage multicore compute clusters through the Message Passing Interface (MPI) [9]. Specifically, a domain decomposition approach has been implemented in which the simulation domain is divided into a number of sub-domains, each of which is mapped to a compute core for execution. Communication and synchronization occur at each time step of the simulation, as bodies may move between sub-domains. This framework currently uses the Discrete Element Method (DEM) formulation in which small interpenetrations between colliding rigid bodies are penalized based on a force model (see for example, [8]).

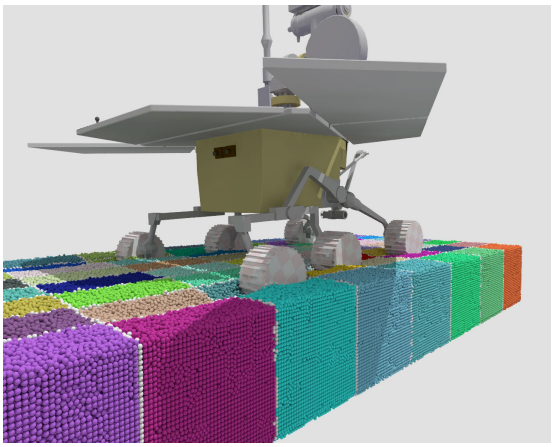

FIGURE 1: Snapshot of Mars Rover simulation with 2,016,000 terrain particles using 64 sub-domains. Bodies are colored by sub-domain, with shared bodies (those which span sub-domain boundaries) colored white.

To demonstrate the capabilities of this framework, a Mars Rover type vehicle has been simulated operating on discrete granular terrain. The vehicle is modeled with a chassis and six wheels driven with a constant angular velocity of π rad/sec. The granular terrain is represented by 2,016,000 spherical particles. The system is divided into 64 sub-domains and simulated on a single node with 4 x AMD Opteron 6274 2.2GHz 16 core processors. A snapshot of the simulation can be seen in Fig. 1.

CPU vs GPU Comparison For the CPU vs GPU comparison, simulations with increasing number of bodies were run. In each scenario, granular material was dropped into a box enclosed on all sides by rigidly fixed walls. Three seconds of this simulation were simulated, the total time taken for each simulation is presented in Fig. 2. The number of bodies simulated ranged from about 2300 for the smallest case to 16000 for the largest. It should be noted that for simulations smaller than 2000 bodies, the overhead associated with transferring data between the GPU and CPU is very large compared to the computational time. Also when solving such a small problem on the GPU, the large number of processing cores (480 in the case of the GTX 480) means that many cores will be idle and the full compute power of the GPU will not be utilized. Therefore for small problems it

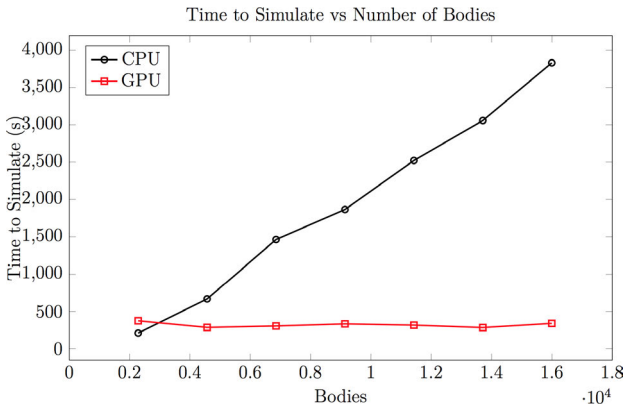

FIGURE 2: Scaling of the CPU vs GPU for different amounts of bodies

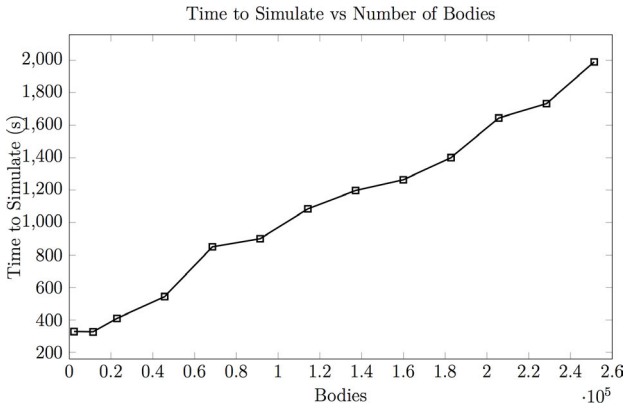

FIGURE 3: Scaling of the GPU

is recommended that the CPU algorithms be used.

Scaling Analysis For an identical simulation setup to that presented in section , larger amounts of bodies were dropped into the box. Note that for the previous comparison between the CPU and GPU the maximum number of bodies simulated was approximately 16000, for this analysis the maximum number of bodies simulated was an order of magnitude higher at around 250000. Fig. 3 shows that the GPU algorithms scale linearly with respect to the number of bodies and subsequently, the number of contacts as more bodies will result in more contacts. For small numbers of bodies, from 2000 to 10000, the total simulation time for the GPU is relatively flat. This is because the overhead associated with transferring memory from the CPU to the GPU is higher than the time taken for computations. Once the number of bodies gets higher than 20000 the total simulation time begins to increase.

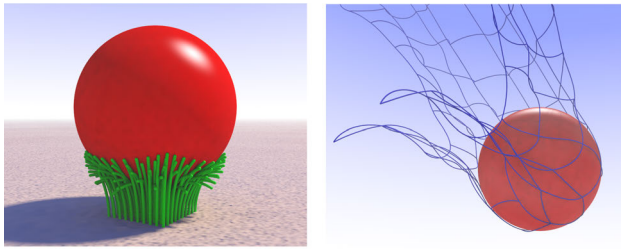

FIGURE 4: Two models with friction and contact using **Chrono**::Flex beam elements: a ball sitting on grass-like beams and a ball hitting a net.

Chrono::Flex

The **Chrono**::Flex software is a general-purpose simulator for three dimensional flexible multi-body problems and provides a suite of flexible body support. The features included in this module are multiple element types, the ability to connect these elements with a variety of bilateral constraints, multiple solvers, and contact with friction. Additionally, **Chrono**::Flex leverages the GPU to accelerate the solution of meaningful engineering problems. This implementation uses gradient deficient ANCF beam elements [10,11] to model slender beams, one of the several flexible elements in **Chrono**::Flex. Shown in Figure 4.

Scaling Analysis Several different net models, similar to the one shown in Figure 4, were simulated using an increasing number of elements with the intent to gauge the efficiency of the implementation. Several instances of the parallel implementation were run for varying net sizes on an Intel Nehalem Xeon E5520 2.26 GHz processor with an (i) NVIDIA GTX 680 graphics card, (ii) NVIDIA Tesla 2070 (Fermi Class), and (iii) NVIDIA Tesla K20 (Kepler Class). The time for each case was recorded and plotted in Figure 5. The speedup obtained with Tesla K20 can be attributed to the fact that this card is a generation Kepler GPU, which currently is the most recent NVIDIA architecture available. The slowest card, Tesla 2070 is a card belonging to the previous generation architecture, code name Fermi, which is two years old. The number of scalar processors on K20 is 2,496; on Tesla 2070 one has only 448 scalar processors, albeit clocked at a higher frequency: 1150 MHz, as opposed to 705 MHz for K20. The manufacturer advertises a double precision theoretical peak Flop/s rate of 1.17 TFlop/s for K20 and 0.515 TFlop/s for Tesla 2070. The GTX 680, is a Kepler card but without the double precision capabilities of K20. There are a many reasons why the relative Tesla K20 to Tesla 2070 speedup is not at around two, as suggested by the peak Flop/s rate ratio. First, the cards almost never operate at peak rate. Second, the simulation is memory intensive, which means that a lot of data is moved back and forth between the SMs and GPU global memory. While the bandwidth of Tesla 2070 is 166 GB/s, it is only modestly higher; i.e., 168

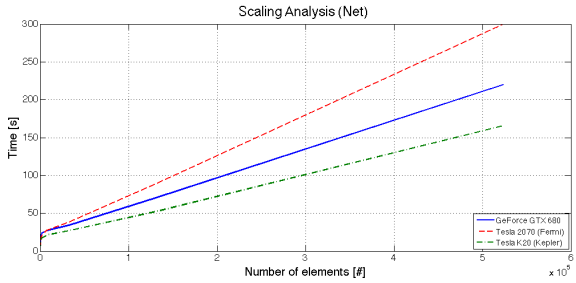

FIGURE 5: The computational time for net models of several sizes were recorded using several different GPUs. The time it took to simulate 10 time steps of the simulation was plotted as a function of the number of beams required to create the net. The scaling analysis went up to 0.5 million ANCF elements.

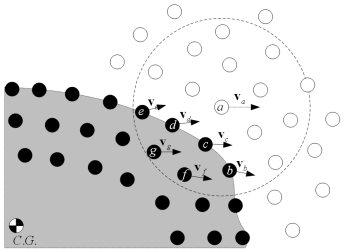

FIGURE 6: Coupling of the fluid and solid phases. BCE and fluid markers are represented by black and white circles, respectively.

GB/s, on the Tesla K20.

Chrono::Fluid

Chrono::Fluid is a library which provides the fluid dynamics simulation and works in conjunction with rigid body dynamics for the simulation of Fluid-Solid Interaction (FSI) problems. The simulation engine is developed in a Lagrangian framework using Smoothed Particle Hydrodynamics (SPH). This allows a straightforward coupling with solid phase and captures the domain deformation without any re-meshing requirement. The whole physical domain, including fluid and solid phases, is modeled as a set of moving markers carrying their volume of interaction, which are short-range symmetric functions, throughout the flow.

An ideal fluid-solid coupling enforces the impenetrability and no-slip conditions on the rigid body surface. This is achieved in **Chrono**::Fluid using Boundary Condition Enforcing (BCE) markers attached to the rigid body surface (see Fig. 6). The inclusion of the BCE markers in continuity and momentum equations trivially enforces the two-way coupling of the fluid and solid phases.

The short-range interaction of the markers requires the assembly of neighbors list which is carried out through the proximity

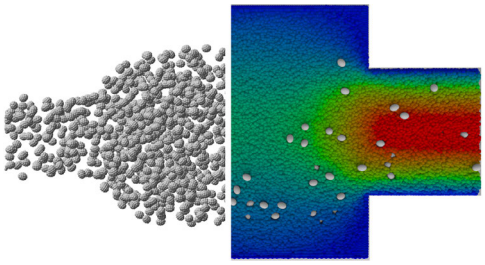

FIGURE 7: Simulation of rigid bodies inside a fluid flow: Rigid ellipsoids are shown in the left image while the fluid's velocity contours and rigid ellipsoids at the mid-section of the channel are shown in the right image.

computation. In our work Proximity computation leverages the algorithm provided in CUDA SDK [12], where the computation domain is divided into bins whose sizes are the same as the resolution length of the SPH kernel function. A hash value is assigned to each marker based on its location with respect to the bins. Markers are sorted based on their hash value. The sorted properties are stored in independent arrays to improve the memory access and cache coherency. To compute the forces on a marker, the lists of the possible interacting markers inside its bin and all 26 neighbor bins are called. The hash values of the bins are used to access the relevant segments of the sorted data. The proximity computation is followed by the force calculation step which evaluates the markers interaction according to SPH, for fluid-fluid interaction, or DEM, for solid-solid interaction. The total number of 6 GPU-based computation kernels are called afterward to update the kinematics of fluid and BCE markers as well as rigid bodies, independently. Since a rigid wall boundary is a particular instance of a rigid body (with zero or pre-defined velocity), it needs no specialized treatment. However, an extra kernel is required for periodic boundary conditions if they are included in the physics. The theoretical linear scalability of the simulation engine was tested and verified in the simulation of fluid-solid interaction problems composed of upto 2.5e6 markers and 3.0e4 rigid bodies. An NVUDIA GeForce GTX 480 GPU was used for the scaling analysis using which the simulation time of the aforementioned largest problem was about 0.6 seconds per time step. Figure 7 shows a snapshot of one of the simulations composed of about 1.5e3 rigid bodies in a flow.

Chrono::Render

Chrono::Render is a software package that offers simple tools for visualizing arbitrary scientific data with a Pixar RenderMan-compliant renderer. Additionally, **Chrono**::Render provides a rendering pipeline that is easily integrated into distributed and remote computing environments. The core design principle of **Chrono**::Render is to enable visualization as an auto-

FIGURE 8: **Chrono**::Render architecture.

mated post-processing service for simulation programs by means of abstracting away the complexities and expertise needed to produce high-quality graphics. Specifically, **Chrono**::Render is a highly-extendible and scriptable rendering framework that is composed of a hybrid of Python modules and compiled libraries. As seen in Figure 8, **Chrono**::Render combines simulation data, hierarchical-data specifications, and optional user-defined Python scripts to generate complex and appealing renders.

The choice to use RenderMan is motivated by the massive scope of data sets and the resulting memory-intensive processing managed by the renderer; REYES, RenderMan's underlying architecture divides surfaces in the scene into micropolygon grids, and requires only a small set of scene elements to be loaded in memory at a time. Scene data can be distributed into small-memory buckets among cores for concurrent rendering. The low memory-footprint and efficient parallel resource usage for the complex scenes makes it a suitable renderer for a general purpose visualization platform. Utilizing this capability to make something visually appealing is difficult without computer graphics expertise. Thus, the key principle of **Chrono**::Render is to make high-quality visualization available to individuals who otherwise lack the knowledge or resources to use comprehensive graphics applications or to understand the complexities of REYES. **Chrono**::Render accomplishes this abstraction by encapsulating complicated visual effects into a succint hierarchical data specification (XML or YAML).

This description is often enough to visualize most generic data, however, it cannot handle all arbitrary visualizations. To maintain generality we make use of Python scripts and wrappers to enable simplified procedural RenderMan Interface Bytestream generation. Most of the **Chrono**::Render Python modules wrap C++ functions and classes with the purpose of exploiting speed while still making use of the syntactical/type-free simplicity of Python. Figure 9 demonstrates a combination of YAML and Python scripts to procedurally generate a heightmap from comma separated data and use it to drive displacement (ie soil compression).

Chrono::Render is currently capable of intergrating with DRMAA and Torque/PBS distributed job managers right out of the box; additionally, **Chrono**::Render automatically configures jobs to maximize the resource usage of these distributed environments for multithreaded rendering. Members of the Wisconsin Applied Computing Center can use this capability re-

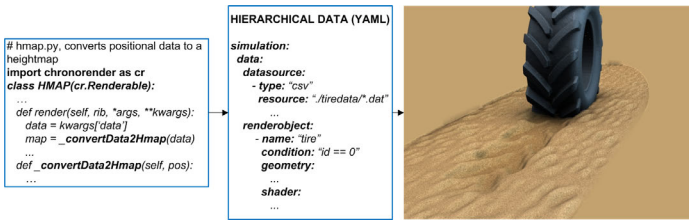

FIGURE 9: Data is tagged with a name which can be accessed later. In this case, the HMAP class implements a render method which converts positional data to a heightmap texture and uses it to drive a displacement shader"

motely as a service by leveraging 320 AMD cores on which **Chrono**::Render is currently deployed.

Conclusions and Future Work

This paper outlines **Chrono**, a simulation framework that relies on parallel computing to predict the dynamics of multi-physics systems. **Chrono** has been used to solve problems with million degrees of freedom that required the parallel formulation and solution of large systems of possible coupled ordinary, differential-algebraic, and partial differential equations. In its current form, **Chrono** typically leverages either GPU computing or CPU computing, and only rarely both of them at the same time. Addressing this limitation remains a direction of future work, along with an ongoing validation effort that is expected to further confirm the predictive capabilities of **Chrono**. Components of **Chrono** can be downloaded for non-commercial use at [13]. Animations of simulations run in **Chrono** are available online [14].

ACKNOWLEDGMENT

Financial support for the University of Wisconsin-Madison authors was provided in part by two awards: National Science Foundation 0840442 and Army Research Office W911NF-12-1-0395. Financial support for A.Tasora was provided in part by the Italian Ministry of Education under the PRIN grant 2007Z7K4ZB. We thank NVIDIA and AMD for sponsoring our research programs in the area of high performance computing.

REFERENCES

[1] Heyn, T., Anitescu, M., Tasora, A., and Negrut, D., 2013. "Using krylov subspace and spectral methods for solving complementarity problems in many-body contact dynamics simulation". *International Journal for Numerical Methods in Engineering*.

[2] Pazouki, A., and Negrut, D., 2013. "A numerical study of the effect of rigid body rotation, size, skewness, mutual distance, and collision on the radial distribution of suspensions in pipe flow". *under review, Langmuir*.

[3] Tasora, A., and Anitescu, M., 2011. "A matrix-free cone complementarity approach for solving large-scale, nonsmooth, rigid body dynamics". *Computer Methods in Applied Mechanics and Engineering,* ***200***(5-8), pp. 439–453.

[4] Heyn, T., 2009. "Simulation of Tracked Vehicles on Granular Terrain Leveraging GPUComputing". M.S. thesis, Department of Mechanical Engineering, University of Wisconsin–Madison, http://sbel.wisc.edu/documents/TobyHeynThesis_final.pdf.

[5] Anitescu, M., and Tasora, A., 2010. "An iterative approach for cone complementarity problems for nonsmooth dynamics". *Computational Optimization and Applications,* ***47***(2), pp. 207–235.

[6] Tasora, A., and Anitescu, M., 2010. "A convex complementarity approach for simulating large granular flows". *Journal of Computational and Nonlinear Dynamics,* ***5***(3), pp. 1–10.

[7] Cundall, P., 1971. "A computer model for simulating progressive large-scale movements in block rock mechanics". In Proceedings of the International Symposium on Rock Mechanics. Nancy, France.

[8] Cundall, P., and Strack, O., 1979. "A discrete element model for granular assemblies". *Geotechnique,* ***29***, pp. 47–65.

[9] Gropp, W., Lusk, E., and Skjellum, A., 1999. *Using MPI: Portable Parallel Programming with the Message-Passing Interface, Second Edition*. MIT Press.

[10] Berzeri, M., Campanelli, M., and Shabana, A. A., 2001. "Definition of the elastic forces in the finite-element absolute nodal coordinate formulation and the floating frame of reference formulation". *Multibody System Dynamics,* ***5***, pp. 21–54.

[11] von Dombrowski, S., 2002. "Analysis of large flexible body deformation in multibody systems using absolute coordinates". *Multibody System Dynamics,* ***8***, pp. 409–432. 10.1023/A:1021158911536.

[12] NVIDIA Corporation, 2012. NVIDIA CUDA Developer Zone. Available online at https://developer.nvidia.com/cuda-downloads.

[13] SBEL, 2010. Software at the Simulation-Based Engineering Laboratory, University of Wisconsin-Madison. http://sbel.wisc.edu/Software.

[14] SBEL, 2012. Multibody Dynamics Simulation Movies, University of Wisconsin-Madison. http://sbel.wisc.edu/Animations.