Real-time Seismic Wave Modeling and Visualization

Aaron Maynard[*] Minglun Gong[†]

Memorial University

ABSTRACT

Current rising energy demand/supply ratio trends are forecast to continue into the foreseeable future. As a consequence, effective and efficient seismic surveying and processing techniques are required now more than ever. Unfortunately those outside of organizations with the resources to develop their own processing and/or visualization packages, or license proprietary third-party ones, must make due with text-based processing systems driven by shell scripts. While these systems perform many of the important algorithms in the seismic processing pipeline with adequate accuracy, the slow performance and limited 3-D processing support along with the learning curve associated with the command-line interface leaves much to be desired in terms of usability .

To address these concerns, an interactive seismic simulation and visualization system has been designed and developed as a solution to enable geophysicists to process and manipulate seismic data in a more natural and intuitive manner. Seismic event simulations are accelerated on the graphics processing unit (GPU) to framerates suitable for real-time visualization, which permits fine control over simulation and visual parameters, allowing them to be modified by the user while the simulation is running. This allows the geophysicist to view the results of any modifications instantaneously, without having to re-start the simulation and generate a pre-computed video file as in prior systems. This paper will focus on the algorithms used to implement this system as well as the resulting improvements in geophysicist work-flow and data gathering efficiency that result from employing real-time simulation techniques; preliminary performance benchmarks will be given along with a brief conclusion.

1 INTRODUCTION

Mechanical waves originate when a sudden stress is applied to a suitable medium, causing a lack of equilibrium which is propagated from the source of impulse[3]. For seismic waves, the propagation medium is the Earths crust. Thus the characteristics of seismic waves, such as amplitude, frequency, and wave velocity; are determined by the laws of physics governing waves, and the composition of the Earths sub-surface. Inversely, if details of the seismic event are known, information regarding certain features of the subsurface can be deduced.

For over two decades[11], the leading open-source seismic processing toolkit has been Seismic Unix (*SU*). Developed at the Center for Wave Phenomena-Colorado School of Mines, SU has been adopted by universities as a research tool and by small to medium-sized companies to process the results of their seismic surveys. While it is robust in the sense that it is highly configurable and supports many seismic algorithms, it is also limited in several areas. SU runs in terminal mode only, meaning that any non-trivial processing sequence must be configured through shell-scripts; a feature that is useful to expert users but sharply increases the learning curve for geoscientists and others who may not have a programming background. SU has no 3D processing support, and although its results can be viewed as animations or images (as appropriate for the algorithm), the results must be pre-computed so animations are non-interactive and limited to a small 2D slice of the domain. This can be alleviated somewhat through clustering[10] as many of the algorithms are highly parallelizable. The fact remains that SU and many commercial packages[1] do not feature real-time processing concurrent with interactive visualization.

The aim is to not only demonstrate the aptness of GPUs for seismic processing and visualization, but also to improve the efficiency of the geophysicists workflow through its easy-to-learn user interface, specifically in the case of forward wave propagation modeling. Forward modeling is used for velocity model verification; that is, the spatial-domain models derived from raw field data, which have been subject to post-processing and depth-conversion[4]. The methods used to generate these models are highly sensitive to noise and random pertubations introduced by imperfections in the data recording device (geophone), thus the models are verified by simulating the seismic event that generated the data and comparing the simulated and field results. If the results are well-correlated then the model is considered to fit the data; if not, adjustments must be made based on the level of deviation from the measured field data. For geophysicists working on land or marine surveys, the cycle of data-gathering and model generation/verification can quickly become time-consuming and expensive for sufficiently complex structures; which is exasperated by the fact that processing resources are not typically located on-site, meaning a slight error in geophone array orientation, etc., can lead to numerous delays as data must be transported physically or accross slow satelite connections to the processing center for interpretation. These issues are addressed by using standard, consumer-grade programmable GPUs to accelerate the modeling pipeline; and providing an intuitive user interface which allows the geophysicist to simultaneously verify, visualize, and interpret the candidate model on-site with hardware compact enough to fit into laptop-sized forms, thus saving considerable time and associated financial resources.

2 SEISMIC WAVE PROPAGATION

2.1 Modeling Algorithm

$$\frac{1}{C(x,y,z)^2}\frac{\partial^2\Psi}{\partial t^2} = \frac{\partial^2\Psi}{\partial x^2} + \frac{\partial^2\Psi}{\partial y^2} + \frac{\partial^2\Psi}{\partial z^2} + src(t) \qquad (1)$$

Equation1: Acoustic Wave Equation

The Earth is an effective propagator of several forms of wave energy, as it is both an elastic and acoustic medium. Geophysicists conducting seismic surveys are mainly concerned with acoustic waves, as their higher propagation velocity ensures they are recorded first by the geophone array[2]. Acoustic waves can be modeled with the second-order partial differential equation given in (1), where C is the wave velocity at a point in the domain of interest; *x, y, z,* and t are spatial and temporal coordinates respectively; src(t) is the seismic source function; and Ψ is the wave pressure field. Solving this equation accross a domain Ω will gradually disperse the wave energy introduced by src(t) over a given Δt.

Because computing exact derivatives is far too costly to be considered for an interactive application, a set of finite difference (FD) approximations are used to calculate the wave propagation vector; more specifically an order-8 in space and order-2 in time kernel. The domain is decomposed into a uniform mesh; then at every

[*]e-mail: thomasm@mun.ca

[†]e-mail: gong@mun.ca

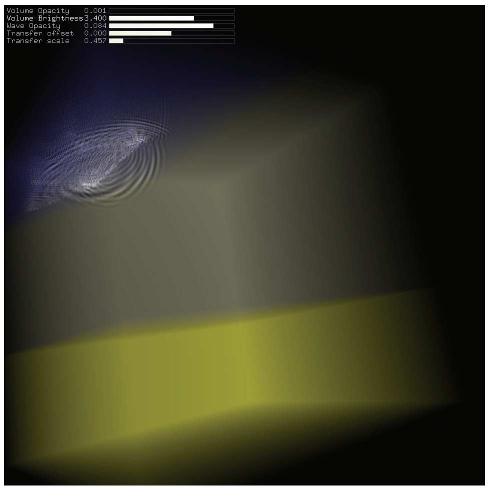
mesh element, the FD kernel is evaluated, which takes the half-order number of neighbouring elements in each spacial direction and the previous (in time) value of the current element, computes the differences between them, then sums the results modulated by a particular weight value, determined by the size of the kernel[6].

2.2 GPU Implementation

Seismic problems such as wave modeling are particularly suitable for GPU optimization, especially since the development of GPGPU oriented languages has eased the task of implementing parallel scientific algorithms on the GPU. The FD method is by its nature, a data-parallel algortihm in that each mesh element can be computed in any given order, which seems to fit nicely within the stream processor paradigm of the GPU. However, there does exist a memory dependency since each node needs to access its spatial and temporal neighbours. Temporal neighbours are handled by double-buffering the pressure field; i.e. the entire current and previous simulation frames are in memory at all times, once the value is read from the relevant node in the previous timeframe, it is no longer required and the results for the new frame are stored in its place, effectivly swapping the buffers at the end of each frame.

A slightly modified version of the method presented by Micikevicius[8] performs memory optimization for accessing the spatial neighbours. Each thread iterates along the z-axis, storing the current node in shared memory. Following a syncronization, neighbouring threads in the x-y grid can access each others node value through the low-latency shared memory, rather than having to read from global memory for each access. The modifications employed relate to integrating proper FD weights into the computation, and an implementation of energy-absorbing boundary conditions along the edges of the dataset (except for the top boundary, which represents the Earth's surface and should generate a reflection), optimized to execute only when required along the model boundary.

3 Model Visualization



```
Volume Opacity    0.001
Volume Brightness 3.400
Wave Opacity      0.084
Transfer offset   0.000
Transfer scale    0.457
```

Figure 1: Seismic wave visualization featuring refraction and reflection at velocity interface

The simulation is visualized using a volume rendering method. Volume rendering allows varying transparency or color to be encoded into a transfer function that highlights interesting portions of a dataset and also allows the rendering of these variations to proceed efficiently. The physical-approximation approach of volume rendering enables high-quality image results and permits easily mixing surface, volume, and other primitives in a scene, while accounting for many of their optical interactions. This allows the geophysicist or other interpreter to resolve features with greater detail than traditional rasterization[7]. To aid perception and interpretation of data, the user is given abillities to interactively adjust viewing location and orientation, extract and isolate volume slices along any axis, and to adjust the transfer function scale in order to clearly visualize data accross a dynamic range of scales.

4 Preliminary Results

The system has been benchmarked on several classes of machines. On an NVIDIA GTX 285 GPU, the simulator solved the FD equations at a rate of approx. 130 GFLOP/s while rendering at a framerate of approx 25 FPS. The system was also tested on a mobile class GPU, an NVIDIA GTX 260M, which due to a lower shader count and power saving design performed considerably worse; about 5.5 GFLOP/s for the simulation and between 3-5 FPS.

5 Conclusion

The development of programmable GPUs over the last decade and inherent increases in performance and flexibility has opened up a new opportunity to advance interactive user interfaces in ways that were previously unfeasable. It has provided the means by which the default rendering pipeline can be altered to implement the direct volume rendering algorithm; and also to offload intensive computations onto the parallel platform to acheive performance increases vs. serial and distributed implementations, and real-time visualization speeds. The system presented here can also be generalized in order to model other scientific phenomena approximated by FD, such as weather/climate modeling[5] and heat diffusion[9]; and, along with the appropriate transfer functions, can visualize them so that the improvement in information conveyal and user interaction applied here to the geosciences can be extended to these and other scientific domains.

References

[1] *Visualization Reference for the Oil and Gas Industry*. Gulf Publishing Company, 2006.

[2] R. P. Bording. *Wave equation difference engine*. PhD thesis, University of Tulsa, 1995.

[3] J.-P. Cordier. *Velocities in Reflection Seismology*. Kluwer Academic Publishers, 1985.

[4] E. L. Etris, N. J. Crabtree, J. Dewar, and S. Pickford. True depth conversion: More than a pretty picture. *Canadian Society of Exploration Geophysicists Recorder*, 26:11–22, 2001.

[5] J. W. Finch and J. H. C. Gash. Application of a simple finite difference model for estimating evaporation from open water. *Journal of Hydrology*, 255(1-4):253 – 259, 2002.

[6] B. Fornberg. Calculation of weights in finite difference formulas. *SIAM Rev*, 40:685–691, 1998.

[7] H. F. Gerd Marmitt and P. Slusallek. Interactive volume rendering with ray tracing. Eurographics 2006 STAR Report, 2006.

[8] P. Micikevicius. 3d finite difference computation on gpus using cuda. In *GPGPU-2: Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units*, pages 79–84, New York, NY, USA, 2009. ACM.

[9] A. S. Mokhtar, K. A. Abbas, M. M. H. M. Ahmad, S. M. Sapuan, A. Ashraf, M.A.Wan, and B. Jamilah. Explicit finite difference solution of heat transfer problems of fish packages in precooling. *American Journal of Applied Sciences*, 1(2):115–120, 2004.

[10] A. E. Murillo and J. Bell. Distributed seismic unix: a tool for seismic data processing. *Concurrency: Practice and Experience*, 11:169–187, 1999.

[11] S. U. Website. *http://www.cwp.mines.edu/cwpcodes*. Colorado School of Mines - Center For Wave Phenomena, 2010.