Th A10 11

Seismic Modeling with Half Precision Floating Point Numbers

G. Fabien-Ouellet* (Polytechnique Montréal, Stanford University), R.G. Clapp (Polytechnique Montréal)

Summary

Half precision floating point numbers is becoming increasingly supported by new processors, often with a significant throughput gain over single precision operations. In this article, we investigate whether half precision is suitable for finite difference based seismic modeling in the context of imaging and inversion. By scaling the finite difference expression of the isotropic elastic wave equation, we manage to obtain a stable solution despite the very narrow dynamic range of the half-precision format. We present a CUDA implementation of this code, which, on most recent GPUs, is nearly twice as fast and uses half the memory of the equivalent single precision version. The error on seismograms caused by the reduced precision is shown to correspond to a fraction of a percent of the total seismic energy, and is mostly incoherent with seismic phases. Thus, half precision modeling could accelerate full waveform inversion or migration with a negligible impact on the quality of their output.

Introduction

The recent boom of machine learning applications has positioned this industry as a major driving force behind processors and hardware evolution. Because training neural network can be done in low precision arithmetic, new accelerators now support the half precision floating point (FP16) format (Zuras et al., 2008). For example, Nvidia Tesla P100 and V100 can use half precision to achieve twice the throughput of single precision (Foley and Danskin, 2017). Even larger accelerations are possible with the use of Tensorcores, processing units specialized for matrix multiplication. This shift in favor of FP16 is industry-wide and includes nearly all major manufacturers like Intel, Nvidia and AMD.

To cope with the ever-increasing computational cost of seismic inversion and imaging, numerical algorithms used for seismic modeling must be able to benefit from hardware evolution. This has been the case with the shift to general-purpose processing on graphics processing units, which have accelerated significantly the finite-difference time-domain and the spectral element methods (Micikevicius, 2009; Komatitsch et al., 2010). Likewise, can seismic modeling and inversion benefit from this new trend of FP16 computing? In this paper, we try to address this question. We first propose a strategy to adapt the velocity-stress elastic wave equation to the narrow dynamic range of FP16. Then, we examine the accuracy and the performance of our CUDA implementation on the most recent Nvidia GPUs.

Method

Let's consider the isotropic elastic wave equation in the velocity-stress formulation:

$$\partial_t v_i - \frac{1}{\rho}\partial_j \sigma_{ij} = 0, \tag{1a}$$

$$\partial_t \sigma_{ij} - (M-2\mu)\partial_k v_k \delta_{ij} - \mu(\partial_j v_i + \partial_i v_j) = s_{ij}. \tag{1b}$$

Here, $v_i(\boldsymbol{x},t)$ are the particle velocity in direction i, $\sigma_{ij}(\boldsymbol{x},t)$ are the stress components, $s_{ij}(\boldsymbol{x},t)$ are the source terms and $\rho(\boldsymbol{x})$, $M(\boldsymbol{x})$ and $\mu(\boldsymbol{x})$ are the density, the P-wave modulus and the shear wave modulus. The velocity-stress formulation can be solved efficiently by the time-domain finite-difference method on a staggered grid (Virieux, 1986). For the sake of simplicity, we will discuss the 1D wave equation, which exhibits the main issues of using half precision for seismic modeling. Later numerical results will be performed in 2D and 3D, for which the explicit finite-difference solutions are given by Virieux (1986) and Graves (1996), respectively. In 1D, we have:

$$v_x^{t+1/2} = v_x^{t-1/2} + \Delta t \rho_x^{-1} D^{+} \sigma^t_{x+1/2}, \tag{2a}$$

$$\sigma^{t+1}_{x+1/2} = \sigma^t_{x+1/2} + \Delta t M_{x+1/2} D^{-} v_x^{t+1/2} + \Delta t s^t_{x+1/2}. \tag{2b}$$

The grids are staggered in both space and time. The superscripts t and subscripts x locate the position of a grid point on the time and space axis, respectively. Staggered grid points are indicated by non-integer indices, e.g. $x + 1/2$. Finally, we have left the spatial difference operators $D^{\pm}$ arbitrary for brevity.

Naively converting every variable from single to half precision in the modeling algorithm simply cannot work, because of the narrow dynamic range of FP16. To illustrate this point, consider the terms $\Delta t \rho^{-1}$ and $\Delta t M$ in Eq. 2, and take $\rho = 2000\ kg/m^3$, $V_p = 3500\ m/s$ and $\Delta t = 10^{-4}\ s$. Then, $\Delta t M = 2450000$, which is much larger than 65504, the maximum FP16 value, and $\Delta t \rho^{-1} = 5 \times 10^{-8}$, which is much smaller than the smallest normal number 6.1×10^{-5}. For this reason, it is impossible to directly work in FP16.

To solve this problem, we propose to rescale Eq. 2 so that that each term is normalized around 1. To do so, we define two scaling factors, one for the material parameters and one for the sources:

$$e_v = -\log_2(\Delta t \max(M)), \tag{3a}$$

$$e_s = -\log_2(\Delta t \max(s)). \tag{3b}$$

In the code, we perform the scaling before computations by modifying the material parameters and by defining normalized sources:

$$a_x = 2^{-e_v} \Delta t \rho_x^{-1}, \tag{4a}$$
$$b_{x+1/2} = 2^{e_v} \Delta t M_{x+1/2}, \tag{4b}$$
$$s'^{t}_{x+1/2} = 2^{e_s} \Delta t s^{t}_{x+1/2}. \tag{4c}$$

We can then solve a transformed version of Eq. 2:

$$v'^{t+1/2}_{x} = v'^{t-1/2}_{x} + a_x D^+ \sigma'^{t}_{x+1/2}, \tag{5a}$$
$$\sigma'^{t+1}_{x+1/2} = \sigma'^{t}_{x+1/2} + b_{x+1/2} D^- v'^{t+1/2}_{x} + s'^{t}_{x+1/2}. \tag{5b}$$

The primes indicate that variables are scaled. Using the linearity of the wave equation in respect to sources, the original solution can be recovered simply by rescaling the velocities and the stresses: $v = 2^{-e_s} v'$, $\sigma = 2^{-e_s - e_v} \sigma'$.

Observe that the scaled parameters a and b now have magnitudes commensurate to the dynamic range of half precision ($a = 0.1225$ and $b = 1$ for the same values as before). Scaling the sources further guarantees that σ' and v' will stay within the range of half precision. The scaling approach is directly applicable in 2D and 3D for the elastic equation (Eq. 1). In that case, a scaled parameter for μ, $c = 2^{e_v} \Delta t \mu$ is also required.

Results

We implemented the scaled modeling algorithm in CUDA, allowing computations on Nvidia GPUs. The code is a ported version of SeisCL's kernels (Fabien-Ouellet et al., 2017), translated from OpenCL to CUDA. We first benchmark the performance of our code and then test the accuracy of the solution.

Performance

Conversion from FP16 to FP32 leads directly to a reduction of memory usage by a half. This is quite appealing on its own. However, the reduced precision of FP16 over FP32 is only acceptable if it also comes with a significant reduction in runtime. We here quantify the gain in performance by comparing three different versions of our code: 1–the baseline FP32 version, 2–the FP16 IO version, which reads half values from global memory to shared memory, converts them to float, computes the update, converts back to half and writes the updated values to global memory, and 3–the FP16 COMP version, which performs every operation in half precision (storage and arithmetic). Apart from the float type, the three versions of the GPU kernel are identical.

We measured runtimes for square and cubic models (2D and 3D modeling), with edges between 2048 and 7744 every 64 points in 2D, and edges between 256 and 544 every 8 points in 3D. For each model size, 3 shots with 1000 time steps were computed. This measure was performed for different Nvidia GPU models: the Tesla K40 and the Tesla M40 (both supporting FP16 storage only) and the Tesla P100 and the Tesla V100 (both supporting full FP16 arithmetic). Note that we do not use the Tesla V100 Tensorcores in this work. The average acceleration of all model sizes along their standard deviation are presented in Fig. 1. The acceleration is defined as the ratio between runtime in FP32 and runtime for the two FP16 versions. The acceleration of FP16 IO is between 1.7x up to over 2x, both in 2D and 3D. In all cases, this is very near the expected 2x speedup attainable when both FP16 storage and arithmetic are enabled. On the P100 and V100, which support FP16 arithmetic, the added gain is between 5 and 15 %. This highlights the fact that finite-difference kernels are highly bandwidth limited, meaning that the time to read and write from/to GPU global memory is much larger than the time required by arithmetic operations. FP16 computation is certainly useful, however older GPU models without this feature also greatly benefit from using FP16.

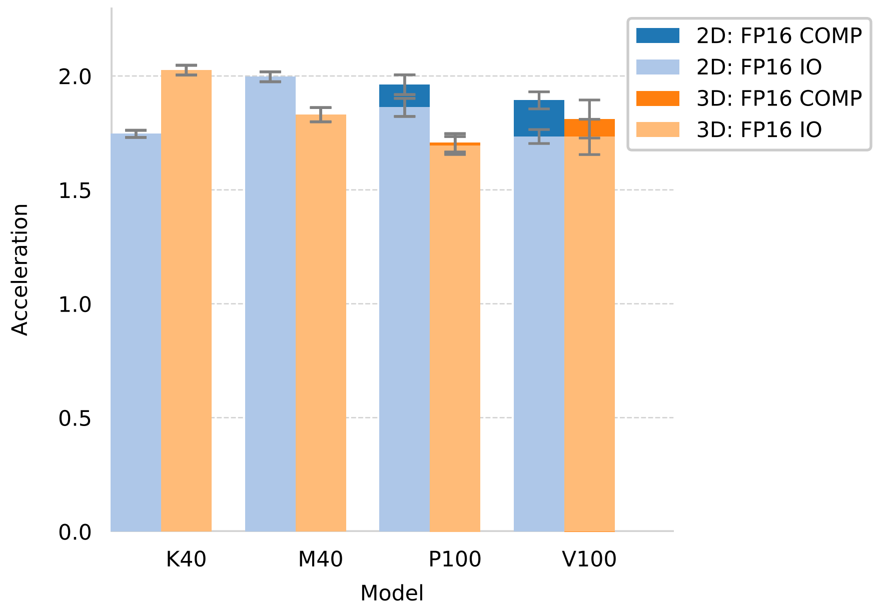

Figure 1 *Acceleration obtained by using half precision over single precision for different GPU models. Uncertainty bars indicate the standard deviation observed over all model sizes. Note that the Tesla K40 and M40 do not support FP16 arithmetic, which is why only the FP16 IO version results are reported.*

Accuracy

The scaling strategy addresses the limited dynamic range of FP16, but rounding errors still lead to a dramatic reduction of significant digits. How accurate is the finite-difference solution then? To answer this question, we modeled a seismic shot for the 2004 BP velocity benchmark (Billette and Brandsberg-Dahl, 2005). As a baseline, we use the FP32 solution computed with fourth order Taylor finite-difference coefficients (Fig. 2b). Traces were clipped at 0.5 % of the maximum value of the gather to highlight faint reflections and refractions, which are of interest. Fig. 2c-d show the error between the FP32 shot gather and the FP16 IO and the FP16 COMP outputs, respectively. To highlight the structure of the numerical error, we used the clipping of the FP32 gathers divided by 50. The error obtained with the FP16 IO version show two different components: a background non-coherent noise with an amplitude that decreases in time, and a coherent noise correlated with the main arrivals contained in the shot gathers. The coherent noise is concerning, but has a very low amplitude, amounting to only 0.035 % of the total energy contained in the shot gathers. The error of the FP16 COMP version is shown in Fig. 2d. The coherent noise is more energetic this time, and can be clearly seen for refracted events at large offsets. Error remains very low though, representing 0.030 % of the total energy. Because of the very small amplitude of the noise in both versions of the code, FWI or migration should not be impacted too much by the reduced accuracy of the FP16 solution.

Conclusions

Our results show that it is possible to use half precision floating point numbers to solve the isotropic elastic wave equation by the time-domain finite-difference method. A simple test on the BP benchmark model indicates that the solution should be accurate enough for FWI or migration, which is confirmed by further analysis (not shown herein). Advantages of using FP16 are a reduction by a factor of 2 of memory usage, and an acceleration by a factor between 1.7 and 2 of the code execution on recent GPUs. The acceleration is mainly caused by the twofold increase of the number of grid elements per cycle that can be accessed in memory. This means a significant acceleration can be attained even with older GPU models that only support FP16 reading and writing from/to memory. Finally, the possibility of solving

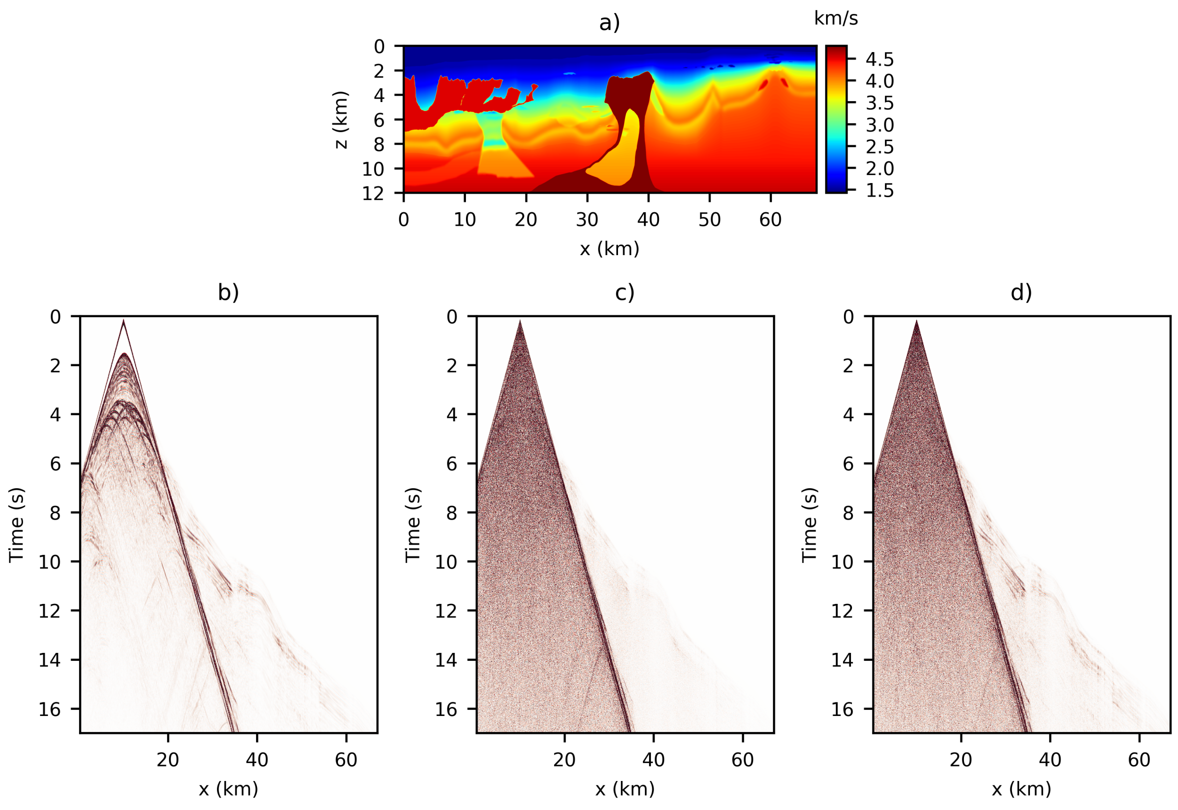

***Figure 2** Comparison between modeling in FP16 and FP32: the Vp velocity a), the FP32 shot gather b), the error of FP16 IO c) and of FP16 COMP d).*

the wave equation in reduced precision opens the way to benefit from more sophisticated accelerators that are now being produced for FP16 computations, like Nvidia Tensorcores.

References

Billette, F. and Brandsberg-Dahl, S. [2005] The 2004 BP velocity benchmark. In: *67th EAGE Conference & Exhibition*.

Fabien-Ouellet, G., Gloaguen, E. and Giroux, B. [2017] Time-domain seismic modeling in viscoelastic media for full waveform inversion on heterogeneous computing platforms with OpenCL. *Computers & Geosciences*, **100**, 142–155.

Foley, D. and Danskin, J. [2017] Ultra-Performance Pascal GPU and NVLink Interconnect. *IEEE Micro*, **37**(2), 7–17.

Graves, R.W. [1996] Simulating seismic wave propagation in 3D elastic media using staggered-grid finite differences. *Bulletin of the Seismological Society of America*, **86**(4), 1091–1106.

Komatitsch, D., Erlebacher, G., Göddeke, D. and Michéa, D. [2010] High-order finite-element seismic wave propagation modeling with MPI on a large GPU cluster. *Journal of Computational Physics*, **229**(20), 7692–7714.

Micikevicius, P. [2009] 3D finite difference computation on GPUs using CUDA. In: *Proceedings of 2nd workshop on general purpose processing on graphics processing units*. 79–84.

Virieux, J. [1986] P-SVwave propagation in heterogeneous media: Velocity-stress finite-difference method. *Geophysics*, **51**(4), 889–901.

Zuras, D., Cowlishaw, M., Aiken, A., Applegate, M., Bailey, D., Bass, S., Bhandarkar, D., Bhat, M., Bindel, D. and Boldo, S. [2008] IEEE standard for floating-point arithmetic. *IEEE Std 754-2008*, 1–70.