GEOPHYSICS, VOL. 84, NO. 1 (JANUARY-FEBRUARY 2019); P. F1–F15, 12 FIGS., 7 TABLES.
10.1190/GEO2017-0624.1

Cu*Q*-RTM: A CUDA-based code package for stable and efficient *Q*-compensated reverse time migration

Yufeng Wang[1], Hui Zhou[1], Xuebin Zhao[1], Qingchen Zhang[2], Poru Zhao[3], Xiance Yu[3], and Yangkang Chen[4]

ABSTRACT

Reverse time migration (RTM) in attenuating media should take absorption and dispersion effects into consideration. The latest proposed viscoacoustic wave equation with decoupled fractional Laplacians facilitates separate amplitude compensation and phase correction in Q-compensated RTM (Q-RTM). However, intensive computation and enormous storage requirements of Q-RTM prevent it from being extended into practical application, especially for large-scale 2D or 3D cases. The emerging graphics processing unit (GPU) computing technology, built around a scalable array of multithreaded streaming multiprocessors, presents an opportunity for greatly accelerating Q-RTM by appropriately exploiting GPUs architectural characteristics. We have developed the cu$Q$-RTM, a CUDA-based code package that implements Q-RTM based on a set of stable and efficient strategies, such as streamed CUDA fast Fourier transform, checkpointing-assisted time-reversal reconstruction, and adaptive stabilization. The cu$Q$-RTM code package can run in a multilevel parallelism fashion, either synchronously or asynchronously, to take advantages of all the CPUs and GPUs available, while maintaining impressively good stability and flexibility. We mainly outline the architecture of the cu$Q$-RTM code package and some program optimization schemes. The speedup ratio on a single GeForce GTX760 GPU card relative to a single core of Intel Core i5-4460 CPU can reach greater than 80 in a large-scale simulation. The strong scaling property of multi-GPU parallelism is demonstrated by performing Q-RTM on a Marmousi model with one to six GPU(s) involved. Finally, we further verified the feasibility and efficiency of the cu$Q$-RTM on a field data set.

INTRODUCTION

Seismic wave absorption and dispersion, resulting from the presence of intrinsic anelasticity in subsurface media, has been considered one of the most important factors degrading the quality of seismogram and decreasing the resolution of migrated images, which affects the reliability of seismic interpretation (Wang and Guo, 2004; Carcione, 2007; Wang et al., 2018b). Many models have been proposed to characterize this frequency-dependent attenuation, which can be roughly classified into two categories: mechanical models and mathematical models. The former includes standard spring-pot models such as the Maxwell body, Kelvin-Voigt model, standard linear solid model (Carcione, 2007; Mainardi, 2010), their generalizations such as the generalized Maxwell body and generalized Zener body (Moczo and Kristek, 2005; Cao and Yin, 2014), and their fractional extensions such as the fractional Kelvin model and fractional Zener model (Rossikhin and Shitikova, 2010; Näsholm and Holm, 2013). In general, the attenuation of seismic waves appears to be adequately modeled by a power law (Strick, 1967; Szabo, 1994, 1995) or linear dependence on frequency over a finite bandwidth (a special case of power-law attenuation with a power of one) (McDonal et al., 1958; Futterman, 1962; Kjartansson, 1979). Therefore, another

Peer-reviewed code related to this paper can be found at http://software.seg.org/2018/0001.
Manuscript received by the Editor 21 September 2017; revised manuscript received 10 September 2018; published ahead of production 12 October 2018; published online 21 December 2018.

[1]China University of Petroleum-Beijing, State Key Lab of Petroleum Resources and Prospecting, Key Lab of Geophysical Exploration of CNPC, Changping 102249, Beijing, China. E-mail: hellowangyf@163.com; huizhou@cup.edu.cn; cupzhaoxuebin@qq.com.
[2]Chinese Academy of Geosciences, Center for Computational and Exploration Geophysics, State Key Laboratory of Geodesy and Earth's Dynamics, Institute of Geodesy and Geophysics, Xudong Road, Wuhan 430077, China. E-mail: qczh@whigg.ac.cn.
[3]China National Petroleum Corporation, Beijing, China. E-mail: zpru@hotmail.com; xiance_yu@hotmail.com.
[4]Zhejiang University, School of Earth Sciences, Hangzhou 310027, China. E-mail: chenyk2018@126.com.

category of models is established based on the assumption of power-law attenuation, which includes the Kolsky-Futterman model (Kolsky, 1956; Futterman, 1962), the power-law model, and Kjartansson's constant-Q model (Kjartansson, 1979).

The attenuation models mentioned above are designed for mathematically characterizing frequency-dependent attenuation effects of subsurface media and further paving the way to mitigate these effects during seismic wave propagation. Early attempts to compensate for the Q effect (attenuation and dispersion effects) are mostly conducted in the framework of one-way wave-equation migration (Dai and West, 1994; Mittet et al., 1995; Wang and Guo, 2004; Mittet, 2007; Zhang et al., 2012). In recent years, Q-compensated reverse time migration (Q-RTM) has received increasing attention from the geophysical community (Causse and Ursin, 2000; Zhang et al., 2010; Zhu et al., 2014; Guo et al., 2016; Li et al., 2016; Sun et al., 2016; Wang et al., 2018c), which generalizes acoustic RTM by considering viscoacoustic propagation and compensating amplitude loss and phase distortion during source and receiver wavefield extrapolation. Zhu and Harris (2014) propose a novel viscoacoustic wave equation with decoupled fractional Laplacians (DFLs), which separately dominate amplitude attenuation and phase dispersion, and they further apply this viscoacoustic wave equation in RTM so as to improve the resolution and quality of the image (Zhu et al., 2014). This decoupled viscoacoustic wave equation is attractive for Q-RTM due to its flexibility for separate amplitude compensation and phase correction, which can be achieved by reversing the absorption proportionality coefficient in sign while leaving the equivalent dispersion parameter unchanged (Treeby et al., 2010; Zhu et al., 2014).

Although the basic paradigm of Q-RTM has been well-established in recent years, there are still some problems and limitations in the process of the implementation, i.e., intensive computation, huge storage requirements and frequent disk I/O, and the difficult issue of stability. The emerging graphics processing unit (GPU) computing technology, built around a scalable array of multithreaded streaming multiprocessors, presents an opportunity for accelerating Q-RTM much further by appropriately exploiting the GPUs architectural characteristics (Farquhar et al., 2016; Tan et al., 2016). As a booming technology, GPU computing technology has been widely applied into seismic modeling (Micikevicius, 2009; Zhang and Gao, 2014), imaging (Foltinek et al., 2009; Zhang et al., 2009; Liu et al., 2012, 2013; Yang et al., 2014), and inversion (Shin et al., 2014; Yang et al., 2015). In this paper, we present a CUDA-based code package called cu$Q$-RTM, which aims to tackle these problems so as to achieve an efficient, storage saving, and stable Q-RTM. Next, we will briefly introduce how cu$Q$-RTM is designed to deal with these challenges and then outline the architecture of the cu$Q$-RTM code package.

Specifically, to avoid intensive computation, we implement Q-RTM in a multilevel parallel (MLP) fashion, either synchronously or asynchronously, to take advantage of all the CPUs and GPUs available. In the framework of cu$Q$-RTM, the basic forward and backward modeling modules, based on viscoacoustic wave equations with DFLs, are efficiently simulated using the Fourier pseudospectral method (PSM) (Carcione, 2010; Zhu and Harris, 2014; Chen et al., 2016). Discrete Fourier transforms (DFTs) of complex wavefields are the most time-consuming parts of these modules. Fortunately, DFTs can be efficiently computed by calling the CUDA fast Fourier transform (CUFFT) library API (Guide, 2013), which provides a simple interface for computing parallel FFTs on the GPU, and a simple configuration mechanism called a plan that completely specifies the optimal plan of execution. The use of a CUFFT standard library brings two obvious benefits: The configuration mechanism allows us to create the plans once and execute the plans multiple times (at every time step of the iteration) without recalculation of the configuration. Every CUFFT plan can be associated with a specified CUDA stream. Streaming the CUFFT execution allows for potential overlap between transforms and memory copies and provides a balanced calculation load on each card of the GPUs. CUFFT library functions can only be executed on the device and called from the host, so we have to split our customized kernel functions into k-space components and x-space components. From the brief codes in Appendix A, we can clearly figure out how these components are interconnected.

Apart from the issue of intensive computation, extensive data storage and burdensome disk I/O are two other bottlenecks for conventional RTM, especially for CUDA-based RTM, which demands frequent memory copying between the host and the device (Liu et al., 2013; Yang et al., 2014). In the past three decades, several wavefield reconstruction strategies have been developed to reach a reasonable compromise between the computer memory requirement and computational complexity, for example, reverse propagation coupled with effective boundary saving (Yang et al., 2014), the optimal checkpointing scheme (Griewank and Walther, 2000; Symes, 2007), and their combinations such as the time-reversal checkpointing (TRC) method (Anderson et al., 2012) and the checkpointing-assisted reverse-forward simulation (CARFS) method (Yang et al., 2016). Yang et al. (2016) propose a novel viscoacoustic wavefield reconstruction algorithm referred to as CARFS, which is implemented by monitoring the energy errors of the reconstruction and taking it as a criterion to decide whether forward simulation or reverse simulation will be performed at the next time step. Wang et al. (2017b) propose a robust viscoacoustic wavefield reconstruction scheme using TRC and k-space filtering (KSF). In this hybrid scheme, TRC serves as a time-domain regularization to eliminate accumulating errors by replacing the reconstructed wavefield with the stored wavefield at checkpoints, whereas KSF further suppresses high-wavenumber artifacts introduced during time-reversal reconstruction. In the cu$Q$-RTM package, we adopt the checkpointing-assisted time-reversal reconstruction (CATRC) scheme to reconstruct source wavefields, which combines the efficiency of reverse propagation and the stability of checkpointing. Unlike CARFS, our proposed CATRC scheme keeps the reconstruction errors within an acceptable range by imposing low-pass filtering on the time-reversal reconstructed wavefield so as to maintain a fixed recomputation ratio of two.

Finally, amplitude compensation in Q-RTM suffers from numerical instability because it boosts the high-frequency noise arising from the high-frequency noise in seismic data and the numerical errors from the finite machine precision (Wang, 2009; Zhu et al., 2014; Yang et al., 2016; Zhao et al., 2017). Therefore, stabilization needs to be introduced either in the frequency or wavenumber domain (Kalimeris and Scherzer, 2012; Ammari et al., 2013). Because the forward and backward modeling modules are simulated by PSM in cu$Q$-RTM, it is more natural to conduct stabilization in the wavenumber domain. In some literature concerning Q compensation, high-frequency noises are suppressed by using a low-pass Tukey filter with its cutoff frequency identified by the noise level of measured data (Treeby et al., 2010; Zhu et al., 2014; Li et al., 2016). However, conventional time-invariant filtering fails to adapt with the Q distribution and compensation depth (traveltime). Wang et al.

(2018c) develop an adaptive stabilization for Q-RTM by analytically deriving k-space Green's functions for the constant-Q wave equation with DFLs and its compensated equation, in which the stabilization factor can be explicitly identified by the specified gain limit according to an empirical formula. In our provided package, we use the proposed adaptive stabilization method to deal with numerical instability in Q-RTM, which exhibits superior properties of time variance and Q dependence over conventional low-pass filtering.

In this paper, we present an open-source code package cu$Q$-RTM, which overcomes several problems commonly existing in conventional Q-RTM such as intensive computation, data storage, and numerical stability, by adopting stable and efficient strategies such as streamed CUFFT, CATRC, and adaptive stabilization. The general architecture of the cu$Q$-RTM code package consists of memory manipulation, modules, kernels, and MLP. Each component plays an indispensable role in GPU-CPU cooperative computing. We further demonstrate the validity and efficiency of cu$Q$-RTM with synthetic and field examples.

OVERVIEW OF *Q*-RTM

In this section, we first review the general principle of Q-RTM in the framework of the viscoacoustic wave equation with DFLs, which includes viscoacoustic propagation, compensation, and imaging. Besides that, two stable and efficient strategies, CATRC and adaptive stabilization, are also provided to improve the computation and stability performance of Q-RTM.

Propagation, amplitude compensation, and imaging

The viscoacoustic wave equation with DFLs was first proposed by Zhu and Harris (2014) to characterize frequency-dependent attenuation and dispersion separately, which can be written as

$$\begin{cases}\frac{1}{c^2}\frac{\partial^2 p}{\partial t^2}(\mathbf{x},t)-\eta(-\nabla^2)^{\gamma+1}p(\mathbf{x},t)-\tau\frac{\partial}{\partial t}(-\nabla^2)^{\gamma+\frac{1}{2}}p(\mathbf{x},t)=\delta(\mathbf{x}_s)f(t),\\ p(\mathbf{x},t)=\frac{\partial p}{\partial t}(\mathbf{x},t)=0, \quad \mathbf{x}\in\Omega,t<0.\end{cases} \tag{1}$$

where Ω is a bounded domain in d-dimensional space $\mathbb{R}^d$, $\mathbf{x}_s$ denotes the source position, and $f(t)$ is the point source signature enforced at $\mathbf{x}_s$. The dimensionless parameter $\gamma = \arctan(1/\pi Q)$ ranges within $[0, 1/2)$, and $c^2 = c_0^2 \cos^2(\pi\gamma/2)$, where c_0 is the velocity model defined at the reference frequency ω_0. The proportionality coefficients of two fractional Laplacians, separately representing dispersion and absorption, are given by $\eta = -c_0^{2\gamma}\omega_0^{-2\gamma} \cos(\pi\gamma)$ and $\tau = -c_0^{2\gamma-1}\omega_0^{-2\gamma} \sin(\pi\gamma)$. Equation 1 seems to be attractive for Q-RTM owing to its flexibility for separately compensating amplitude loss and correcting phase distortion. Treeby et al. (2010) and Zhu et al. (2014) state that attenuation compensation based on this equation can be achieved by reversing the absorption proportionality coefficient in sign but leaving the equivalent dispersion parameter unchanged. Our latest work has analytically proved that Green's function of equation 1 is exponentially decreasing, whereas reversing the absorption proportionality coefficient in sign signifies replacing the Green's function with a phase-conjugated Green's function that is exponentially increasing (Wang et al., 2017c, 2018c).

The novel paradigm of Q-RTM first proposed by Zhu et al. (2014), where the source wavefield $p_s(\mathbf{x}, t)$ and receiver wavefield $p_r(\mathbf{x}, t)$ are compensated during forward extrapolation and time-reversal extrapolation simultaneously, coupled with a zero-lag crosscorrelation imaging condition, has proven to be a promising approach for generating high-resolution images and high-fidelity amplitude reflectors. Following the spirit of Treeby et al. (2010) and Zhu et al. (2014), the Q-compensated source wavefield $p_s(\mathbf{x}, t)$ is the solution of the following equation:

$$\begin{cases}\frac{1}{c^2}\frac{\partial^2 p_s}{\partial t^2}(\mathbf{x},t)-\eta(-\nabla^2)^{\gamma+1}p_s(\mathbf{x},t)+\tau\frac{\partial}{\partial t}(-\nabla^2)^{\gamma+\frac{1}{2}}p_s(\mathbf{x},t)=\delta(\mathbf{x}_s)f(t),\\ p_s(\mathbf{x},t)=\frac{\partial p_s}{\partial t}(\mathbf{x},t)=0, \quad \mathbf{x}\in\Omega,t<0,\end{cases} \tag{2}$$

and the Q-compensated receiver wavefield $p_r(\mathbf{x}, t)$ satisfies the following equation:

$$\begin{cases}\frac{1}{c^2}\frac{\partial^2 p_r}{\partial t^2}(\mathbf{x},t)-\eta(-\nabla^2)^{\gamma+1}p_r(\mathbf{x},t)+\tau\frac{\partial}{\partial t}(-\nabla^2)^{\gamma+\frac{1}{2}}p_r(\mathbf{x},t)=\delta(\mathbf{x}_r)g(\mathbf{x},T-t),\\ g(\mathbf{x},t)=p(\mathbf{x},t), \quad \mathbf{x}\in\mathbf{x}_r,t\in[0,T],\end{cases} \tag{3}$$

where $\mathbf{x}_r$ denotes the receiver positions and $g(\mathbf{x}, t)$ stands for the recorded seismic data at $\mathbf{x}_r$, which are reversed in time and enforced as the Dirichlet boundary condition. Finally, we realize Q-RTM via the following zero-lag crosscorrelation imaging condition:

$$I(\mathbf{x}) = \int_0^T p_s(\mathbf{x}, t)p_r(\mathbf{x}, t)dt. \tag{4}$$

However, an inevitable issue imposed by the crosscorrelation algorithm is that the forward wavefields need to be accessible at every time step (Anderson et al., 2012; Yang et al., 2016). Saving all forward wavefields requires tremendous memory and frequent disk I/O, which makes it impractical for large-scale 2D or 3D RTM (Symes, 2007; Tan and Huang, 2014), especially for CUDA-based RTM, which demands data transfer between the host and the device (Yang et al., 2014).

CATRC

To relieve extensive data storage and burdensome disk I/O and thus reach a reasonable compromise between the computer memory requirement and computational complexity, we propose an efficient wavefield reconstruction strategy called CATRC, which combines the efficiency of reverse propagation and the stability of checkpointing. Therefore, source wavefields used in the imaging condition in equation 4 can be well-reconstructed during time-reversal simulation. Here, we denote the reconstructed wavefields as $q(\mathbf{x}, t)$, which is the solution of

$$\begin{cases}\frac{1}{c^2}\frac{\partial^2 q}{\partial t^2}(\mathbf{x},t)-\eta(-\nabla^2)^{\gamma+1}q(\mathbf{x},t)-\tau\frac{\partial}{\partial t}(-\nabla^2)^{\gamma+\frac{1}{2}}q(\mathbf{x},t)=\delta(\mathbf{x}_{\partial\Omega})h(\mathbf{x},T-t),\\ h(\mathbf{x},t)=p_s(\mathbf{x},t), \quad \mathbf{x}\in\partial\Omega,t\in[0,T],\\ q(\mathbf{x},t)=p_s(\mathbf{x},t), \quad t\in\{C_i|i\in[0,N_c-1]\}\bigcup\{T,T-\Delta t\},\end{cases} \tag{5}$$

where $\partial\Omega$ is the boundary of space Ω; $h(\mathbf{x}, t)$ are the forward wavefields distributed on $\partial\Omega$, which are reversed in time and enforced as the Dirichlet boundary condition for source wavefield reconstruction; C_i denotes a checkpoint; and N_c is the number of checkpoints. According to equation 5, the implementation of CATRC can be briefly summarized as two processes. First, we compute forward wavefield $p_s(\mathbf{x}, t)$ by solving the compensated viscoacoustic wave equation in equation 2, and we save the forward wavefield at the outermost layer boundary of the simulation domain at every time step. At the same time, we also save the complete forward wavefield $p_s(\mathbf{x}, t)$ at the predefined checkpoints $(t \in C_i, i = 0, \ldots, N_c - 1)$ and the last two time steps. The checkpoints can be equally distributed or logarithmically distributed (Griewank and Walther, 2000; Symes, 2007). Then, we compute the backward wavefield $q(\mathbf{x}, t)$ in reverse time (from $t = T$ to $t = 0$) by solving the reconstructed viscoacoustic wave equation (equation 5) and replacing the calculated backward wavefield $q(\mathbf{x}, t)$ with the recorded forward wavefield $p_s(\mathbf{x}, t)$ at checkpoints $(t \in C_i, i = N_c - 1, \ldots, 0)$.

It is remarkable that reconstruction in equation 5 is a mathematically stable process, given that the source wavefield is compensated, whereas the reconstructed wavefield from the boundary is attenuated. However, this stable reconstruction still suffers from insufficient accuracy due to the fact that we use PSM to solve equation 5 with only the recorded forward wavefield at the outermost layer boundary of the simulation domain at every time step. This mismatch of simulation accuracy inevitably degrades the performance of the wavefield reconstruction. Fortunately, the TRC scheme acts as a time-domain regularization that eliminates accumulating errors by replacing the reconstructed wavefield with the stored wavefield at checkpoints (Wang et al., 2017b).

Adaptive stabilization

Mathematically speaking, the compensated viscoacoustic wave equations 2 and 3 are severely ill posed due to the presence of the compensating term $+\tau\partial_t(-\nabla^2)^{\gamma+1/2}p(\mathbf{x}, t)$. That is to say, amplitude compensation is a nonstationary process with energy exponentially amplified over the traveltime, which boosts high-frequency ambient noise and can even result in numerical instability. In the package, we apply an adaptive stabilization scheme for Q-RTM to suppress unwanted high-frequency artifacts, which is discussed in our previous work (Wang et al., 2018c). Here, we briefly summarize the process.

We derive a k-space Green's function of equation 1 by enforcing a point source at time $t = t_0$ and $\mathbf{x} = \mathbf{x}_s$. The time-space harmonic Green's function $G(\mathbf{k}, \omega)$ is the solution of the following Helmholtz equation:

$$\left(\frac{\omega^2}{c^2} + \eta|\mathbf{k}|^{2\gamma+2} + i\omega\tau|\mathbf{k}|^{2\gamma+1}\right) G(\mathbf{k}, \omega) = \frac{1}{(2\pi)^{d+1}} e^{-i\omega t_0} e^{i\mathbf{k}\mathbf{x}_s}. \tag{6}$$

Solving for Green's function $G(\mathbf{k}, \omega)$, applying $d + 1$ dimensional inverse Fourier transformation, and then integrating the kernel function with respect to ω based on Cauchy's residue theorem, we have the following time-domain attenuated Green's function:

$$G_{\mathrm{att}}(\mathbf{x}, t) = \frac{c^2}{(2\pi)^d} \int_{\mathbb{C}^d} \frac{\sin(\xi_1 t) e^{-\xi_2 t}}{\xi_1} d\mathbf{k}. \tag{7}$$

The compensated Green's function can be obtained by reversing the absorption-related term τ in sign but leaving the other term η unchanged:

$$G_{\mathrm{comp}}(\mathbf{x}, t) = \frac{c^2}{(2\pi)^d} \int_{\mathbb{C}^d} \frac{\sin(\xi_1 t) e^{\xi_2 t}}{\xi_1} d\mathbf{k}, \tag{8}$$

where G_{att} and G_{comp} represent the attenuated and compensated Green's functions, respectively. These two Green's functions lay the foundation for designing an adaptive stabilization operator. Inspired by stabilization in inverse Q filtering (Wang, 2002, 2006; Irving and Knight, 2003), we proposed a similar adaptive stabilization for Q-RTM, which can be defined as

$$\Lambda(\mathbf{k}, t) = \frac{\beta(\mathbf{k}, t)}{\beta^2(\mathbf{k}, t) + \sigma^2} = \frac{e^{\xi_2 t}}{1 + \sigma^2 e^{2\xi_2 t}}, \tag{9}$$

where the amplitude-attenuated operator $\beta(\mathbf{k}, t) = e^{-\xi_2 t}$. The final form of our proposed stabilization operator can be given by

$$s(\mathbf{k}, l\Delta t) = \begin{cases} \frac{1}{1+\sigma^2 e^{2\xi_2(\mathbf{k})\Delta t}}, & l = 1, \\ \frac{1+\sigma^2 e^{2\xi_2(l-1)\Delta t}}{1+\sigma^2 e^{2\xi_2 l\Delta t}}, & l = 2, 3, \ldots, n. \end{cases} \tag{10}$$

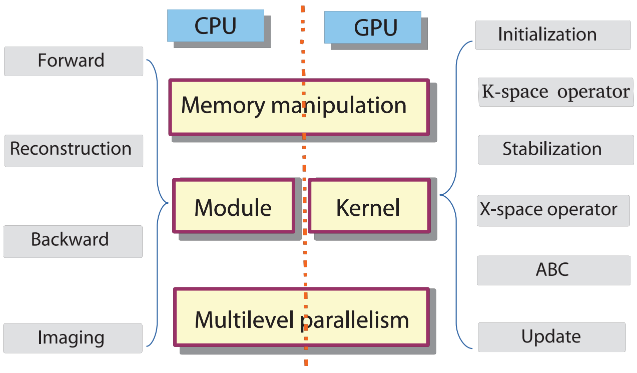

Figure 1. The architecture of the cuQ-RTM code package.

ARCHITECTURE OF THE CUQ-RTM CODE PACKAGE

In this section, we outline the architecture of cu$Q$-RTM code package and some program optimization schemes. From an overall perspective, this package can be roughly separated into four components: memory manipulation, modules, kernels, and MLP. As shown in Figure 1, each component plays an indispensable role in GPU-CPU cooperative computing. The following is a brief description of each component and how it interacts with the others.

Memory manipulation

The CUDA programming model assumes that the host and the device maintain their own separate memory spaces in DRAM, referred to as host memory and device memory, respectively. Before we introduce details about the architecture of the cu$Q$-RTM code package, we need to clarify the variable definition and figure out which variables need to be transferred between host memory and device memory. Table 1 presents some important variables allocated on the host and device, which fall into three memory types: pageable host memory, page-locked host memory, and global device memory. The philosophy of choosing host variable types is that variables to be frequently copied between host and device, such as *seismogram_rms* and *image_cor*, are allocated in the page-locked host memory, whereas the rest of the host variables are allocated as regular pageable host memory. Because copies between the page-locked host memory and the device memory can be performed concurrently with kernel execution, the data transfer can be overlapped during kernel execution leading to a more efficient streaming execution on cluster nodes with multiple GPUs.

The struct MultiGPU contains page-locked host variables and global device variables (with *d* as a prefix) on every stream. From this struct variable, we can estimate the total device memory usage before execution and ensure that the memory usage will not exceed the memory limit. CUDA threads (kernel functions) execute on a physically separate device (GPUs), whereas the rest of the C program executes on the host (CPUs). Therefore, a program manages the global memory accessible to kernels through calls to the CUDA runtime such as device memory allocation `cuda_Device_malloc(...)`, deallocation `cuda_Device_free(...)`, and initialization `cuda_Host_initialization(...)` as well as data transfer between host and device memory.

Kernel

Kernels, the most basic unit of cu*Q*-RTM to accomplish a series of specific tasks such as variable initialization and applying an absorbing boundary condition (ABC), can further be integrated into a fully functional module. Different variables are initialized with distinct kernels `cuda_kernel_initialization(...)`, `cuda_kernel_initialization_images(...)`, and `cuda_kernel_initialization_Finals(...)` based on their scope in modules. Wavefield variables are updated by the `cuda_kernel_update(...)`.

In the framework of cu*Q*-RTM, the basic forward and backward modules based on viscoacoustic wave equation with DFLs are efficiently simulated by PSM coupled with the CUFFT library. However, CUFFT library functions can only be executed on the device and called from the host, so we split our customized kernel functions into a k-space component and x-space component. The Fourier transform function `cufftExecC2C(..., CUFFT_FORWARD)` and inverse Fourier transform function `cufftExecC2C(..., CUFFT_INVERSE)` serve as the link between the x-space operator `cuda_kernel_visco_PSM_2d_forward_x_space(...)` and the k-space operator `cuda_kernel_visco_PSM_2d_forward_k_space(...)`. ABCs for these modeling operators are conducted by the multiple transmitting formula (MTF) `cuda_kernel_MTF_2nd(...)` (Liao et al., 1984).

In the cu*Q*-RTM package, we adopt the CATRC scheme to reconstruct source wavefields, which combines the efficiency of reverse propagation and the stability of checkpointing. Kernel functions `cuda_kernel_checkpoints_Out(...)` and `cuda_kernel_checkpoints_In(...)` are designed to record and fetch forward wavefields at predefined checkpoints. Furthermore, wavefields on the outermost layer boundary of the simulation domain at each time step are also recorded in global device variables `d_borders_up`, `d_borders_bottom`, `d_borders_left`, and `d_borders_right`. The total memory storage for 2D reconstruction can be estimated as

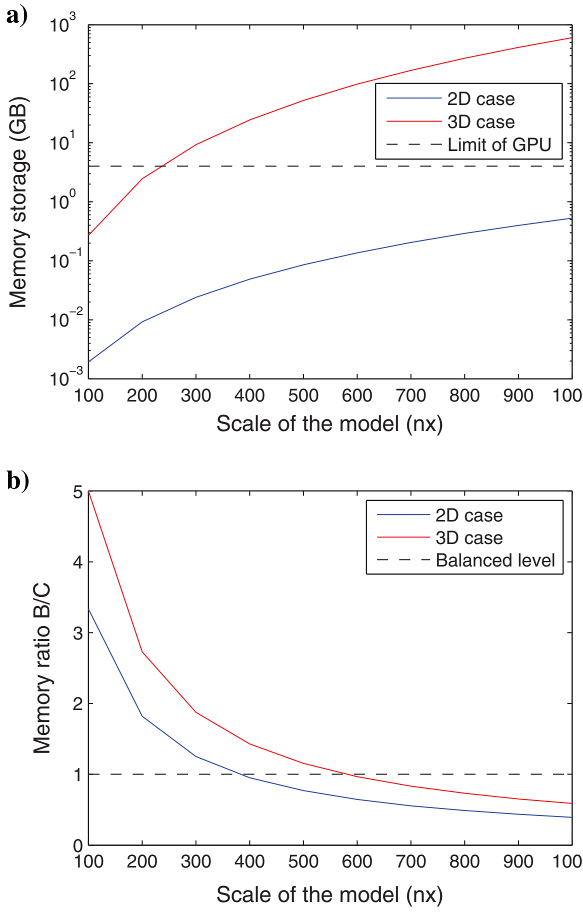

Figure 2. (a) Memory storage and (b) memory ratio of the boundary wavefield to the checkpointing wavefield (B/C) for the 2D and 3D cases.

Table 1. Some important variables allocated on the host and device.

	Memory type	Allocation and free	Variables
Host	Pageable	`malloc() free()`	*ricker, vp, Qp, Gamma, t_cp, kfilter, kstabilization Final_image_cor, Final_image_cor*

	Page locked	`cudaMallocHost()`
		`cudaFreeHost()`
			u0, u1, u2, seismogram_obs, seismogram_rms, image_cor, image_nor

Device	Global	`cudaMalloc()`
		`cudaFree()`
			d_ricker, d_vp, d_Gamma, d_t_cp, d_u_cp, d_u0, d_u1, d_u2, d_seimogram_rms, d_image_cor, d_image_nor d_uk, d_Lap_uk, d_amp_uk, d_pha_uk, d_borders_up, d_u2_final0, d_u2_final1

$$\mathrm{storage}_{\mathrm{2D}}[\mathrm{GB}] \approx \frac{2(nx+nz)nt+(N_c+2)(nx\times nz)}{1024^3/4}, \quad (11)$$

and for the 3D case,

$$\mathrm{storage}_{\mathrm{3D}}[\mathrm{GB}] \approx \frac{2(nx\times ny+nx\times nz+ny\times nz)nt+(N_c+2)(nx\times ny\times nz)}{1024^3/4}, \quad (12)$$

where nx, ny, nz, and nt are the spatial and temporal grid sizes. Figure 2a shows memory storage against the scale of the model (where we denote the size of the simulation as $nx = ny = nz = 0.1$ $nt = 10Nc$) for the 2D and 3D cases. Figure 2b presents the memory ratio between boundary savings and checkpointing savings for 2D and 3D cases. Such a large amount of memory storage is unacceptable for the 3D case, so we have to output the boundary savings to the disk and then read the borders by memory copying between the host and the device memory.

We develop an adaptive stabilization method to deal with numerical instability in Q-RTM, which exhibits superior properties of time variance and Q dependence over conventional low-pass filtering. The adaptive stabilization scheme `cuda_kernel_AdaSta(...)` and the low-pass filtering scheme `cuda_kernel_filter2d(...)` are provided in our package. Users can choose either of these two stabilizing methods to suppress high-frequency noise as they like.

Module

Much like the kernel layer insulates the user from the programming details of a series of specific tasks, the module layer insulates the user from the implementation details of the module of Q-RTM, which contains forward extrapolation, wavefield reconstruction, backward extrapolation, and imaging. Each of them is made up of several kernel functions and streams. A stream is defined by creating a stream object using `cudaStreamCreate(...)` and specifying it as the stream parameter to a sequence of kernel launches and host-device memory copies. Streams are released by calling `cudaStreamDestroy(...)`, which waits for all preceding commands in the given stream to complete before destroying the stream and returning control to the host thread. The forward module `cuda_visco_PSM_2d_forward(...)` in cu$Q$-RTM is designed in a splitting fashion and is called by the main function to conduct forward wavefield extrapolation. Because wavefield reconstruction, backward extrapolation, and imaging are conducted during time-reversal simulation, these three modules can be merged into one module `cuda_visco_PSM_2d_backward(...)`. The forward and backward modules are presented by brief codes in Appendix A.

Multilevel parallelism

The code package described above is implemented for multiple NVIDIA GPUs using message passing interface (MPI), C, and CUDA in an MLP fashion. The execution is divided into two components. The first is responsible for the coarse-grained parallelization between nodes of the clusters, which is parallelized using MPI. The second performs calculations within each GPU using CUDA.

Algorithm 1. Shot distribution for uniform clusters.

```
Input: The number of shots $ns$, nodes $np$, GPUs per node $ng$.
Output: Index of each shot $is$.
$nsid = ns/(np \times ng)$;
$modsr = ns\%(np \times ng)$;
$prcs = modsr/ng$;
if $myid < prcs$ then
    $eachsid = nsid + 1$;
    $offset = myid \times (nsid + 1) \times ng$;
else
    $eachsid = nsid$;
    $offset = prcs \times (nsid + 1) \times ng + (myid - prcs) \times nsid \times ng$;
end if
for $iss = 0 \cdots eachsid - 1$ do
    $offsets = offset + iss \times ng$;
    for $i = 0 \cdots ng - 1$ do
        $is = \mathrm{offsets} + i$;
    end for
end for
```

Algorithm 2. Shot distribution for nonuniform clusters.

```
Input: The number of shots $ns$, nodes $np$, GPUs per node $ng^{(id)}$.
Output: Index of each shot $is$.
$nsid = ns/\sum_{id=0}^{np-1} ng^{(id)}$;
$modsr = ns\% \sum_{id=0}^{np-1} ng^{(id)}$;
for $j = 1 \cdots np - 1$ do
    $prcs = modsr/\sum_{id=0}^{np-1-j} ng^{(id)}$
    if $prcs > 0$ then
        $break$;
    end if
end for
if $myid < np - j$ then
    $eachsid = nsid + prcs$;
    $offset = (nsid + prcs) \times \sum_{id=0}^{myid} ng^{(id)}$;
else
    $eachsid = nsid$;
    $offset = (nsid + prcs) \times \sum_{id=0}^{np-1-j} ng^{(id)} + nsid \times \sum_{id=np-j}^{myid} ng^{(id)}$;
end if
for $iss = 0 \cdots eachsid - 1$ do
    $offsets = offset + iss \times ng^{(myid)}$;
    for $i = 0 \cdots ng^{(myid)} - 1$ do
        $is = \mathrm{offsets} + i$;
    end for
end for
```

One of the most important issues arising when working with a hybrid MPI/CUDA code is the proper mapping of MPI processes and threads to GPUs and nodes. Thus, we can evenly distribute all shots among every node and every GPU, while being aware of the precise index of each shot during simulation. In the package, we provide two distributing schemes to allow the framework to run on uniform clusters (i.e., each node with the same number of integrated GPUs) and on nonuniform clusters (i.e., a mixture of nodes with different numbers and types of integrated GPUs). Every MPI process (rank) `MPI_Comm_rank(comm,&myid)` inspects the configuration of the node being executed on, and all GPUs of each node are launched by streaming execution. Algorithms 1 and 2 provide shot distributing schemes to ensure load balancing on each node and device. All threads on the host are synchronized by `MPI_Barrier(comm)` before migrated images from all shots are reduced by `MPI_Allreduce(...)`, which further guarantees less thread blocking time.

EXAMPLES

In this section, we use synthetic and field examples to demonstrate the superior performance of our presented cu$Q$-RTM package over traditional CPU-based computational models in terms of efficiency, memory storage, and stability. All of the following examples are reproducible when the C, CUDA, MATLAB, and Madagascar platforms (Fomel et al., 2013) are available.

Viscoacoustic modeling on a layered model for speedup comparison

In the first synthetic example, we perform viscoacoustic modeling on a multiscale layered model with a single GeForce GTX760 GPU and a single core of Intel Core i5-4460 CPU for speedup comparison. As shown in Figure 3, the scale of these layered models varies from 128×128 to 2048×2048 grids. We record the mean runtime per time step of viscoacoustic modeling using a single CPU core and a single GPU at each model scale and their corresponding speedup ratio, which are presented in Table 2. CPU-based simulation is compiled by GNU C++ compiler (g++ 4.8.4) with FFTW 3.3.2. GPU-based simulation is compiled by CUDA C with the CUFFT library API. Figure 4 shows the mean runtime per time step and the corresponding speedup ratio against the model scale. It indicates that our presented cu$Q$-RTM package running on a single GPU card can nearly be 50–80 times faster than conventional CPU implementation with a single CPU core. Furthermore, simulation on a larger model scale tends to achieve a greater speedup ratio.

Q-RTM on the Marmousi model for strong scaling analysis

The second synthetic example presented here is CUDA-based Q-RTM for the Marmousi model. Figure 5a and 5b shows its velocity and Q models, which contains a high-attenuation zone with a low Q-value. The model has 234 nodes with a sampling interval of $dz = 10$ m in depth and 663 nodes with a sampling interval of $dx = 10$ m in the horizontal direction. In the observation system, 60 sources are distributed laterally with a shot interval $ds = 100$ m, and each shot has 301 double-sided receivers with a maximum offset of 1500 m. The point source is a Ricker wavelet with a dominant frequency $f_d = 20$ Hz. The synthetic seismic data are modeled by the PSM with time interval $dt = 0.001$ s, and the records last 2 s.

Figure 6 shows the migrated image using conventional RTM from acoustic data (Figure 6a) and viscoacoustic data without compensation (Figure 6b) and Q-RTM from viscoacoustic data (Figure 6c and 6d), respectively. The acoustic imaging result shown

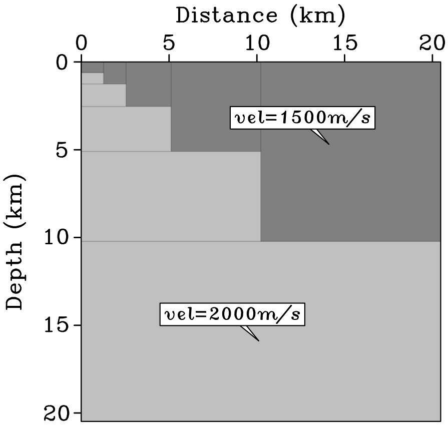

Figure 3. Velocity models for the multiscale layered model.

Table 2. The mean runtime per time step of viscoacoustic modeling using a single GTX760 GPU relative to a four-core Intel Core i5-4460 CPU and the corresponding speedup ratio against model scale.

Model scale (grids)	128×128	256×256	512×512	1024×1024	2048×2048
CPU runtime (ms)	9.7170	43.5925	101.3938	359.0682	1855.8382
GPU runtime (ms)	0.1839	0.8195	1.8262	6.3267	22.3263
Speedup ratio	52.8385	53.1940	55.5217	56.7544	83.1234

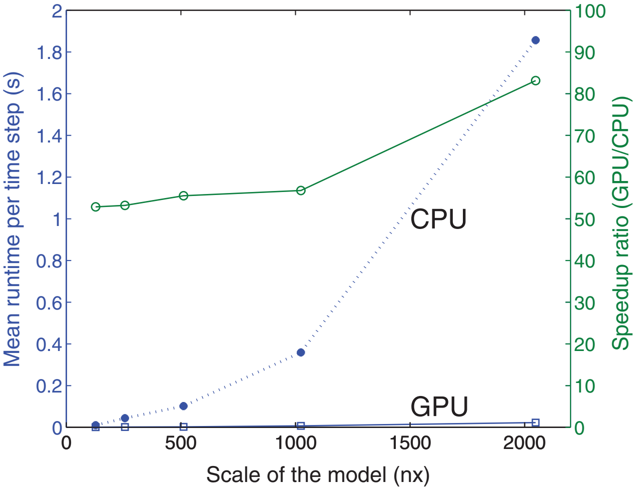

Figure 4. The mean runtime per time step of viscoacoustic modeling using a single GTX760 GPU relative to a four-core Intel Core i5-4460 CPU and the corresponding speedup ratio against the model scale.

in Figure 6a serves as a reference for comparison. Due to the presence of a high-attenuation zone, the imaging result of the structure beneath the high-attenuation zone shown in the blue frame in Figure 6b exhibits attenuated amplitudes and blurred structures. The attenuation also severely affects the migrated image of the anticlinal structure, shown in the green frame in Figure 6b below the unconformity. Figure 6c and 6d shows compensated images from Q-RTM using conventional low-pass filtering and our proposed adaptive stabilization scheme. The compensated images exhibit a clear anticlinal structure and recovered amplitudes compared with the noncompensated image. For another comparison, Figure 7 shows migrated seismic traces that are selected arbitrarily at three distances of 1500, 3600, and 5200 m from the imaging results shown in Figure 6. From these traces, we find that the compensated traces match well with the reference traces. This indicates that our developed cu$Q$-RTM package is capable of improving imaging quality.

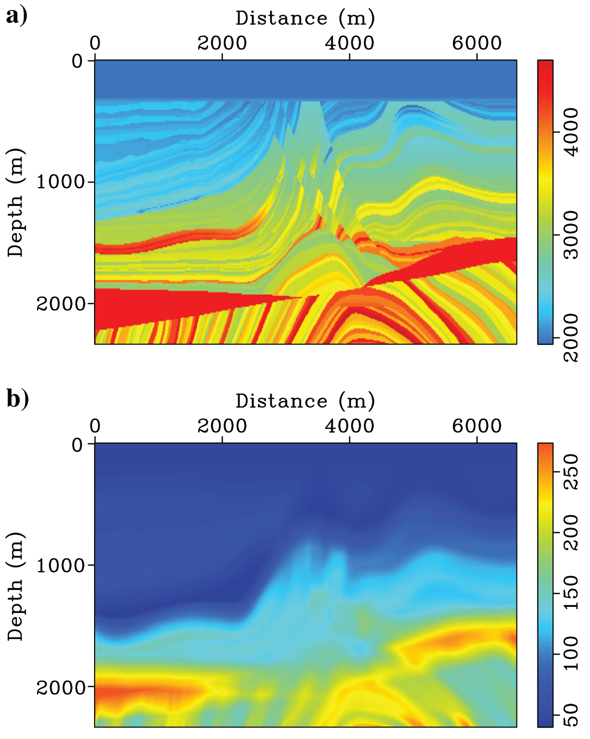

Figure 5. (a) Velocity and (b) Q of the Marmousi model.

The strong scaling plot shows how the execution time decreases with an increasing number of computing resources. During large-scale imaging, the proportion of computational time spent to simulate wave propagation mandates that the solver must be efficient and scale well. In this regard, 60 shots of Q-RTM are evenly distributed among every GPU card with the number of GPUs (Tesla K10) varying from one to six. We record scheduling runtime and computational runtime during every test and present them in Table 3. Figure 8 shows the results of a strong scaling test of cu$Q$-RTM on the Marmousi model. It demonstrates that very close to ideal efficiency can be achieved with a balanced load on each GPU. Thus, the code package exhibits excellent scalability and can be run with almost ideal code performance, in part because communications are almost entirely overlapped with calculations.

Q-RTM on field data for feasibility demonstration

The field data example shown in Figures 9, 10, and 11 aims to further verify the feasibility and applicability of the proposed package. Velocity and Q models are presented in Figure 9, which were obtained by migration velocity analysis (Sava and Vlad, 2008) and Q tomography (Shen and Zhu, 2015). The size of the model is 8.0×15.6 km with the spatial interval $dx = dz = 10$ m. There are 77 shots horizontally distributed on the surface of the model. We perform $nt = 10,000$ time steps for each shot with the temporal interval $dt = 0.0005$ s. To eliminate the diffraction artifacts from long offset, we set the stacking aperture of 2.0 km around the shot. Figure 10 shows the migrated image using conventional RTM without compensation (Figure 10a) and Q-RTM from real data (Figure 10b), respectively. For a clearer comparison, we display the magnified seismic images in Figure 11 corresponding to the marked area from Figure 10. From Figure 10a and 10b, we conclude that the Q-compensated image using our proposed package exhibits sharper reflections and more balanced amplitude.

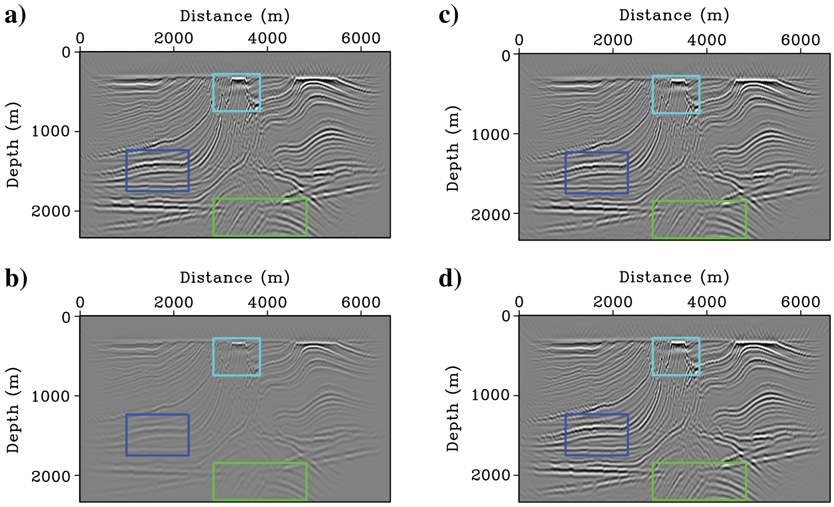

Figure 6. Migrated images of the Marmousi model using (a) conventional RTM from acoustic data, (b) conventional RTM from viscoacoustic media without compensation, (c) Q-RTM using low-pass filtering and (d) Q-RTM using adaptive stabilization scheme.

DISCUSSION

An open-source code package cu$Q$-RTM presented in this paper is designed for efficient and stable viscoacoustic imaging in attenuating media. The package uses streamed CUFFT, CATRC scheme, and adaptive stabilization to pertinently tackle some well-known problems in viscoacoustic imaging such as intensive computations, large storage requirements and frequent disk I/O, and instability. Each of these issues has been discussed in the literature, notably an efficient implementation of 3D FFTs across multiple-GPU systems (Czechowski et al., 2012; Nandapalan et al., 2012), memory-saving reconstruction schemes (Anderson et al., 2012; Yang et al., 2016; Wang et al., 2017b), and stabilized compensation strategies

(Zhu et al., 2014; Sun and Zhu, 2015; Wang et al., 2017a). Our proposed package aims at using a set of the state-of-art strategies to achieve an efficient, storage saving, and stable Q-RTM. Here, we discuss the superior performance of the CATRC scheme and adaptive stabilization over conventional methods. Unlike the conventional effective boundary-saving strategy using finite-differences (FDs) ($2Lc + 1$ grid points FD stencil), which requires Lc layers of boundary wavefields at each time step, our proposed CATRC scheme stores the outermost layers of boundary wavefields at each time step plus states at the checkpoints and the last two time steps to the reconstructed source wavefield for performing crosscorrelating imaging condition, without much loss of precision. Figure 12 shows source snapshots, reconstructed snapshots, and their differences from the Marmousi model at two propagation times. It demonstrates that the reconstructed wavefields from CATRC are accurate enough for Q-RTM.

Because this paper mainly focuses on Q-RTM and its GPU implementation, we do not pay much attention to the numerical simulation of the spatially varying fractional power of Laplacian operators. There are effective proposed schemes to handle spatial variable-order fractional Laplacians (Sun et al., 2014; Chen et al., 2016; Li et al., 2016; Wang et al., 2018a, 2018c). Chen et al. (2016) propose two efficient methods to calculate spatial variable-order fractional Laplacians, i.e., the Taylor-expansion approximation scheme and the low-rank decomposition scheme. Wang et al. (2018a) extend the Taylor-expansion approximation scheme to the viscoelastic case. All of these methods come at the expense of computational efficiency. In our code package, the averaged strategy (Zhu et al., 2014) is used to achieve a quick solution. Improving the accuracy of the code package will be our future work.

Another critical issue is the constructed architecture and parallelism strategy of the cu$Q$-RTM code package. The architecture of the cu$Q$-RTM code package has been discussed in detail, and it can be separated into four components: memory manipulation, kernel, module, and MLP. Task-oriented kernels form several fully functional modules, and these modules are further integrated into a complete process of Q-RTM. The package contains 2D seismic imaging schemes on compensated and noncompensated frames with adaptive stabilization and low-pass filtering strategies. We can execute Q-RTM in a flexible manner by choosing a series of flags responsible for switching among different scenarios without any code modifications. In this sense, cu$Q$-RTM provides a general GPU-based framework to ensure a time- and memory-efficient implementation. The proposed cu$Q$-RTM is implemented in an MLP manner to take advantage of all of the CPUs and GPUs available, while maintaining impressively good stability and flexibility. Whether for a single shot test or a complete simulation, with only a single machine containing seven Nvidia GPU cards, cu$Q$-RTM consistently provides speedup factors approaching or exceeding 50 times compared with conventional CPU-only implementations. Our package is particularly well-suited to Q-RTM in which multiple shots are run on clusters with multiple GPUs per node.

Shared memory is expected to be much faster than global memory, which enables direct GPU-to-GPU transfers (Jaros et al., 2012; Nandapalan et al., 2012). Many researchers proposed shared memory strategy for GPU parallel computing to improve computational efficiency (Micikevicius, 2009; Jaros et al., 2012; Liu et al., 2012, 2013; Nandapalan et al., 2012). In our cu$Q$-RTM code package, we adopt streamed CUFFT to improve computational efficiency with no domain decomposition and GPU-to-GPU transfers involved. For this reason, we do not take the shared memory strategy into considersation, which might be considered as an improvment in a future version.

Apart from outlining the architecture of the cu$Q$-RTM code package and underlining some program optimization schemes, we also provide speedup analysis and a strong scaling test on synthetic

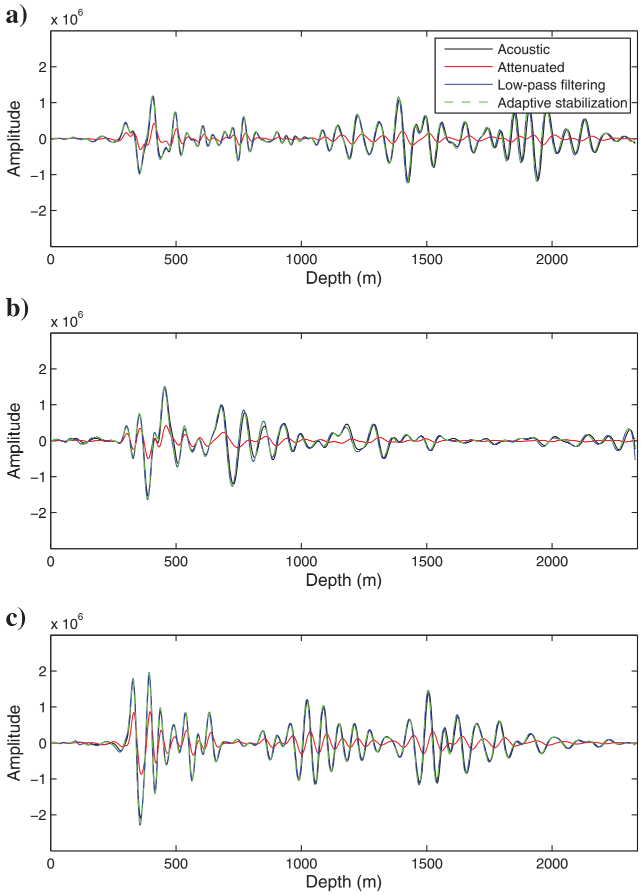

Figure 7. Migrated seismic traces selected at three distances of (a) 1500, (b) 3600, and (c) 5200 m from migration results shown in Figure 6.

Table 3. The runtime of cu$Q$-RTM using multiple tesla K10 GPUs and the corresponding speedup ratio against the number of GPUs.

The number of GPUs	1	2	3	4	5	6
Manipulational runtime (s)	7.62	10.07	10.52	11.02	11.41	11.92
Computational runtime (s)	2639.29	1329.40	889.82	672.78	539.21	449.97
Total runtime (s)	2646.91	1339.50	900.34	683.80	550.62	461.89
Speedup ratio	1.0000	1.9761	2.9399	3.8709	4.8071	5.7306

Note: The model has 234 nodes in depth and 663 nodes in the horizontal direction.

models. With only a single Nvidia GPU card, our presented cu*Q*-RTM code package can be 50–80 times faster than the state-of-art distributed CPU implementation running on a single CPU core. We also find that GPU-based simulation on a larger model scale tends to reach a higher speedup ratio compared with that of small-scale simulation. Objectively speaking, an abusolute speedup ratio without considering the hardware that we used is not really a "fair" comparison. In this paper, we test our package on a GeForce GTX760 GPU and compare it with the conventional CPU implementation running on a single core of Intel Core i5-4460 CPU. If a more modern CPU system or a better GPU card is used, the speedup ratio would be much lower or higher than what we claimed in this paper.

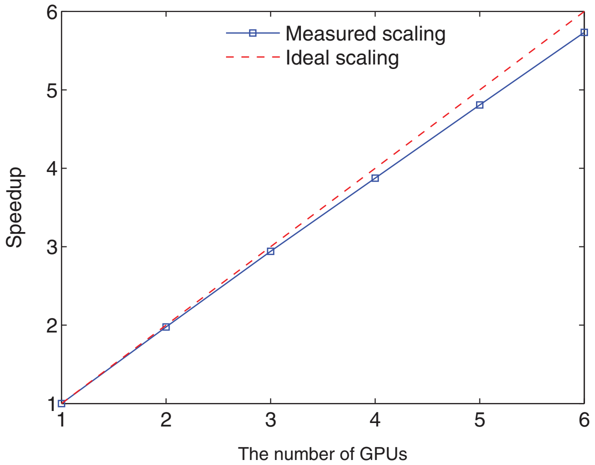

Figure 8. Strong scaling for cu*Q*-RTM on the Marmousi model using multiple Tesla K10 GPUs. Speedup ratios are plotted against the number of GPUs. The model has 234 nodes in depth and 663 nodes in the horizontal direction.

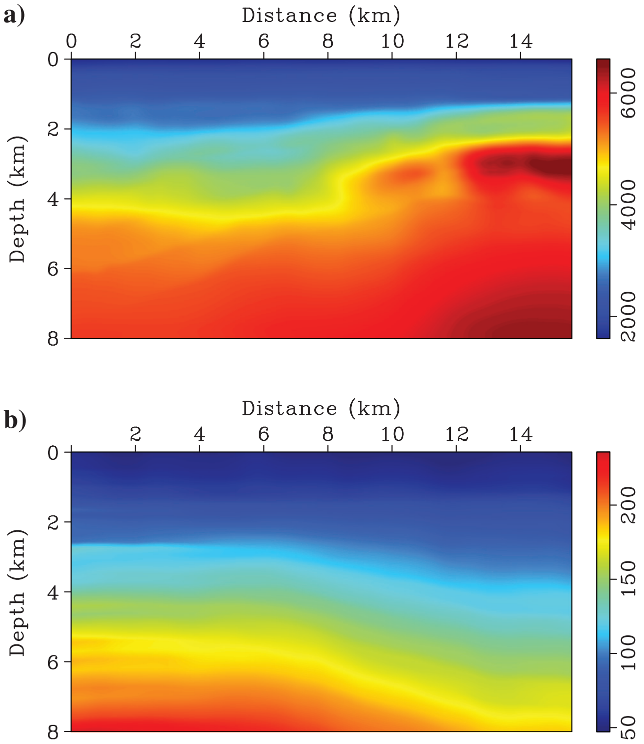

Figure 9. (a) Velocity and (b) *Q* models for field data.

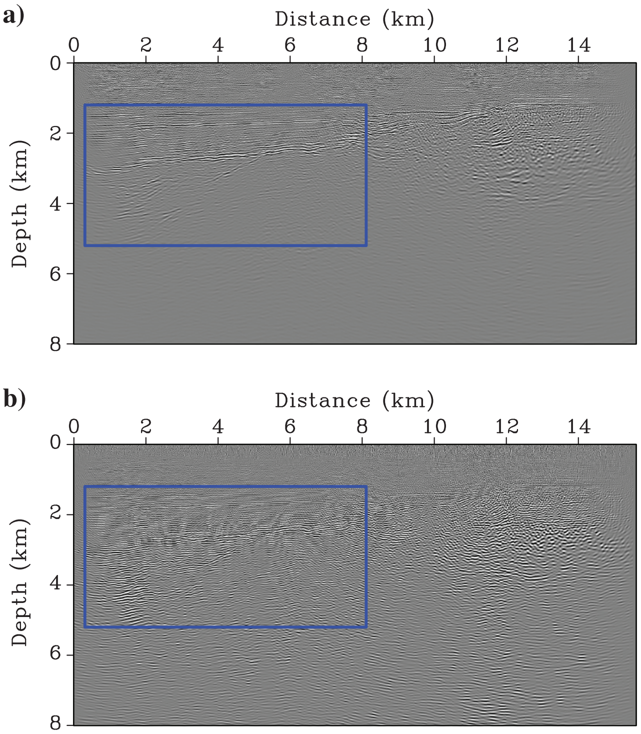

Figure 10. Migrated images of the field data using (a) conventional RTM from viscoacoustic media without compensation and (b) *Q*-RTM.

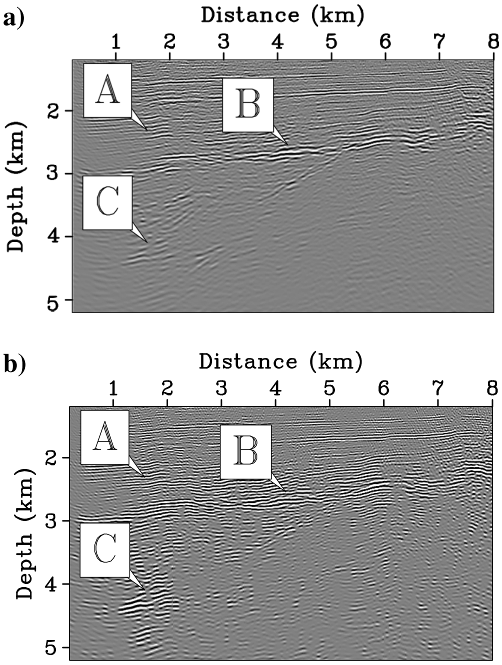

Figure 11. Magnified view of the images shown in the boxs in (a) Figure 10a and (b) Figure 10b.

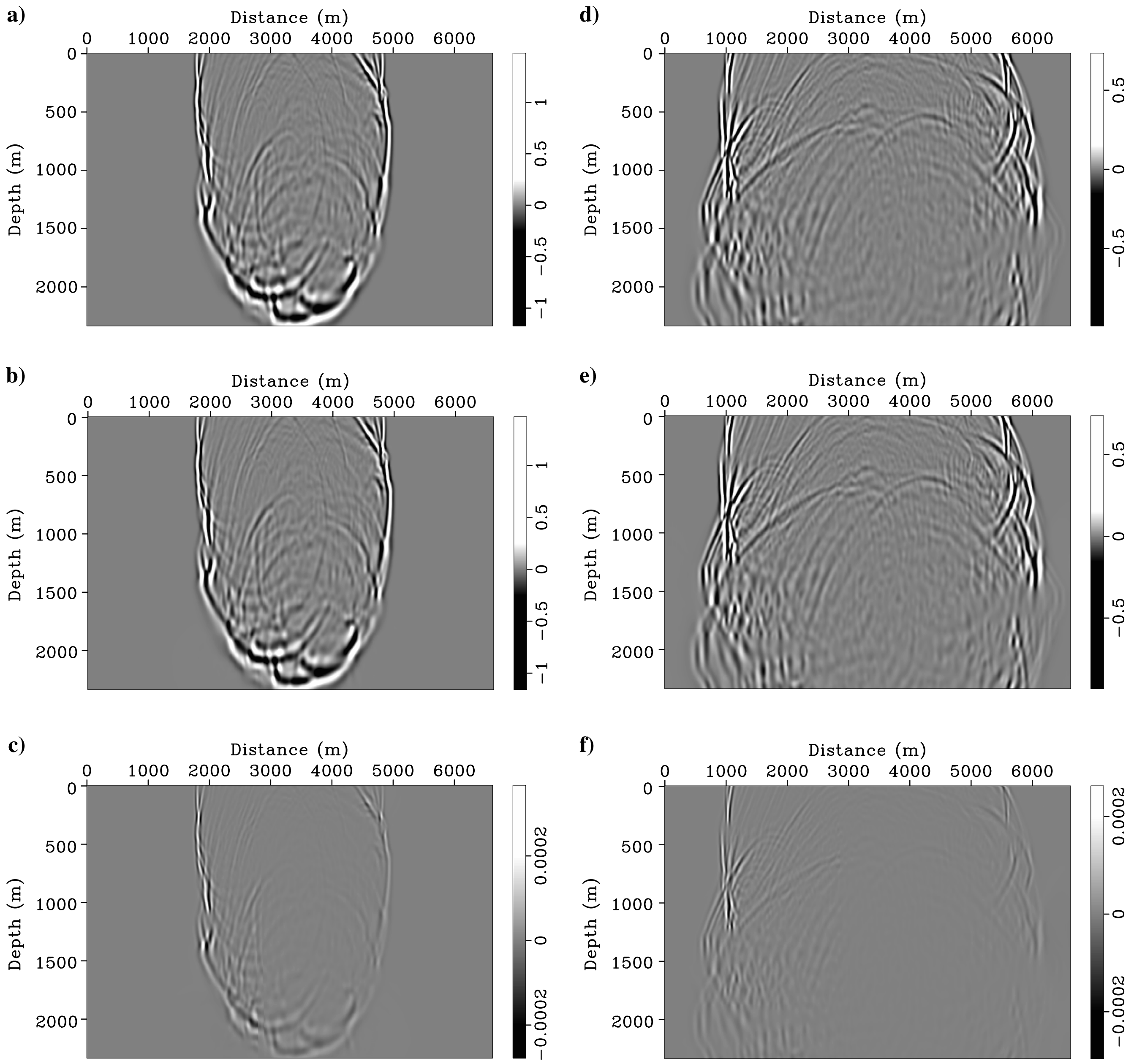

Figure 12. Forward snapshots, reconstructed snapshots, and their differences from Marmousi model (see Figure 5) at two propagation times: (a-c) $t = 0.82$ s and (d-f) $t = 1.22$ s.

Regarding the scaling test on multi-GPUs, our provided code package exhibits excellent scalability and can be run with almost ideal code performance in part because communications are almost entirely overlapped with calculations. Our package's architecture is designed to mimic how a geophysicist writes down a seismic processing module such as modeling, imaging, and inversion. By using the streamed CUFFT, the most time-consuming part of the pseudospectral simulation can be computed synchronously on each device, which improves performance and lends itself naturally to the future implementation of more complicated (*Q*-compensated) LSRTM and FWI (Yang et al., 2015; Jaros et al., 2016a) on the GPU. Future work may also generalize to the 3D case and incorporate a more efficient reconstruction scheme, whereas a further investigation of alternative or improved 2D and 3D FFTs techniques across multiple GPUs (Jaros et al., 2012, 2016b) may also prove worthwhile.

CONCLUSION

In this paper, we have presented an open-source code package cu*Q*-RTM, equipped with a set of the state-of-art strategies such as streamed CUFFT, the CATRC scheme, and adaptive stabilization, to achieve an efficient and robust *Q*-RTM. The architecture of the cu*Q*-RTM code package is composed of four components: memory manipulation, kernel, module, and MLP. Task-oriented kernels are consolidated into several fully functional modules, which are further integrated into the complete process of *Q*-RTM. The package is implemented in an MLP manner to take advantages of all the CPUs and GPUs available, while maintaining impressively good stability and flexibility. We have demonstrated the effectiveness and applicability of our developed package by performing Q-RTM on synthetic and field data. Either synthetic or field migrated images with Q compensation exhibit sharper reflections and more balanced amplitude. Furthermore, speedup tests via viscoacoustic modeling on layered models indicate that our presented cu$Q$-RTM can be 50–80 times faster compared with conventional CPU-based implementation with only a single GPU card. The strong scaling analysis of Q-RTM across multiple GPUs demonstrates the excellent scalability of the package.

ACKNOWLEDGMENTS

We acknowledge J. Dellinger, J. Shragge, and the two anonymous reviewers for their critical comments that significantly improved the manuscript. We thank J. Fang and H. Chen for inspiring discussions about CUDA programming. We are grateful to the developers of the Madagascar software package for providing codes for testing the algorithms and preparing the figures. This work was financially supported in part by the National Natural Science Foundation of China (41174119), National Science and Technology Program (2016ZX05010001), Major Project of the China National Petroleum Corporation (2016A-3302), and Scientific Research and Development of CNPC (2017D-5006-16). Y. Wang is financially supported by the China Scholarship Council (CSC).

DATA AND MATERIALS AVAILABILITY

The Marmousi model used in this package is available for download from the Madagascar MainPage: http://www.ahay.org/data/marm2. The “living” package is available from GitHub at https://github.com/Geophysics-OpenSource/cuQRTM.

```
extern “C”
void cuda_visco_PSM_2d_forward(...)
{
 Multistream plans[GPU_N];
 for(i=0;i<GPU_N;i++)
 {
  cudaSetDevice(i);
  cudaStreamCreate(&plans[i].stream); // create stream for CUFFT
  cufftSetStream(plan[i].PLAN_FORWARD,plans[i].stream);
  cufftSetStream(plan[i].PLAN_BACKWARD,plans[i].stream);
  // copy memory from the host to the device
  cudaMemcpyAsync(...,cudaMemcpyHostToDevice,plans[i].stream);
  // wavefield initialization
  cuda_kernel_initialization<<<dimGrid,dimBlock,0,plans[i].stream>>>(...);
 }
 for(it=0;it<nt;it++) // forward modeling
 {
  for(i=0;i<GPU_N;i++)
  {
   cudaSetDevice(i);
   cufftExecC2C(...,CUFFT_FORWARD); // CUFFT_FORWARD
   // k_space forward modeling
   cuda_kernel_visco_PSM_2d_forward_k_space<<<...>>>(...);
   cufftExecC2C(...,CUFFT_INVERSE); // CUFFT_INVERSE
   // x_space forward modeling
   cuda_kernel_visco_PSM_2d_forward_x_space<<<...>>>(...);
   cuda_kernel_MTF_2nd<<<...>>>(...); // MTF ABC
   cuda_kernel_checkpoints_Out<<<...>>>(...); // record checkpoints
   cuda_kernel_update<<<...>>>(...); // wavefield unpdate
  }
 }
 for(i=0;i<GPU_N;i++)
 {
   cudaSetDevice(i);
   // output seismograms
   cudaMemcpyAsync(...,cudaMemcpyDeviceToHost,plans[i].stream);
   cudaDeviceSynchronize(); // synchronize
   cudaStreamDestroy(plans[i].stream); // destroy stream for CUFFT
 }
}
extern “C”
void cuda_visco_PSM_2d_backward(...)
{
 Multistream plans[GPU_N];
 for(i=0;i<GPU_N;i++)
 {
  cudaSetDevice(i);
  cudaStreamCreate(&plans[i].stream); // create stream for CUFFT
  cufftSetStream(plan[i].PLAN_FORWARD,plans[i].stream);
  cufftSetStream(plan[i].PLAN_BACKWARD,plans[i].stream);
  // copy memory from the host to the device
  cudaMemcpyAsync(...,cudaMemcpyHostToDevice,plans[i].stream);
  // wavefield and images initialization
  cuda_kernel_initialization<<<dimGrid,dimBlock,0,plans[i].stream>>>(...);
 }
 for(it=nt−3;it>=0;it−−)
 {
  for(i=0;i<GPU_N;i++)
  {
   cudaSetDevice(i);
   if(Sto_Rec==0) // wavefield reconstruction
   {
    cufftExecC2C(...,CUFFT_FORWARD); // CUFFT_FORWARD
    // k_space reconstruction
    cuda_kernel_visco_PSM_2d_reconstruction_k_space<<<...>>>(...);
    cufftExecC2C(...,CUFFT_INVERSE); // CUFFT_INVERSE
    cuda_kernel_AdaSta<<<...>>>(...); // stabilization
    // x_space reconstruction
    cuda_kernel_visco_PSM_2d_reconstruction_x_space<<<...>>>(...);
    cuda_kernel_MTF_2nd<<<...>>>(...); // MTF ABC
    cuda_kernel_checkpoints_In<<<...>>>(...); // read checkpoints
    cuda_kernel_update<<<...>>>(...); // wavefield unpdate
   }
   cufftExecC2C(...,CUFFT_FORWARD); // CUFFT_FORWARD
   // k_space backward modeling
   cuda_kernel_visco_PSM_2d_backward_k_space<<<...>>>(...);
   cufftExecC2C(...,CUFFT_INVERSE); // CUFFT_INVERSE
   cuda_kernel_AdaSta<<<...>>>(...); // stabilization
   // x_space backward modeling
   cuda_kernel_visco_PSM_2d_backward_x_space<<<...>>>(...);
   cuda_kernel_MTF_2nd<<<...>>>(...); // MTF ABC
   cuda_kernel_image<<<...>>>(...); // imaging
   cuda_kernel_update<<<...>>>(...); // wavefield unpdate
  }
 }
 for(i=0;i<GPU_N;i++)
 {
  cudaSetDevice(i);
  cudaMemcpyAsync(...,cudaMemcpyDeviceToHost,plans[i].stream);
  cudaDeviceSynchronize(); // synchronize
  cudaStreamDestroy(plans[i].stream); // destroy stream for CUFFT
 }
}
```

APPENDIX A

BRIEF CODES FOR FORWARD AND BACKWARD MODULES

REFERENCES

Ammari, H., E. Bretin, J. Garnier, and A. Wahab, 2013, Time reversal algorithms in viscoelastic media: European Journal of Applied Mathematics, **24**, 565–600, doi: 10.1017/S0956792513000107.

Anderson, J. E., L. Tan, and D. Wang, 2012, Time-reversal checkpointing methods for RTM and FWI: Geophysics, **77**, no. 4, S93–S103, doi: 10.1190/geo2011-0114.1.

Cao, D., and X. Yin, 2014, Equivalence relations of generalized rheological models for viscoelastic seismic-wave modeling: Bulletin of the Seismological Society of America, **104**, 260–268, doi: 10.1785/0120130158.

Carcione, J. M., 2007, Wave fields in real media: Wave propagation in anisotropic, anelastic, porous and electromagnetic media: Elsevier.

Carcione, J. M., 2010, A generalization of the Fourier pseudospectral method: Geophysics, **75**, no. 6, A53–A56, doi: 10.1190/1.3509472.

Causse, E., and B. Ursin, 2000, Viscoacoustic reverse-time migration: Journal of Seismic Exploration, **9**, 165–183.

Chen, H., H. Zhou, Q. Li, and Y. Wang, 2016, Two efficient modeling schemes for fractional Laplacian viscoacoustic wave equation: Geophysics, **81**, no. 5, T233–T249, doi: 10.1190/geo2015-0660.1.

Czechowski, K., C. Battaglino, C. McClanahan, K. Iyer, P.-K. Yeung, and R. Vuduc, 2012, On the communication complexity of 3D FFTs and its implications for exascale: Proceedings of the 26th ACM International Conference on Supercomputing, 205–214.

Dai, N., and G. F. West, 1994, Inverse Q migration: 64th Annual International Meeting, SEG, Expanded Abstracts, 1418–1421, doi: 10.1190/1.1822799.

Farquhar, M. E., T. J. Moroney, Q. Yang, and I. W. Turner, 2016, GPU accelerated algorithms for computing matrix function vector products with applications to exponential integrators and fractional diffusion: SIAM Journal on Scientific Computing, **38**, C127–C149, doi: 10.1137/15M1021672.

Foltinek, D., D. Eaton, J. Mahovsky, P. Moghaddam, and R. McGarry, 2009, Industrial scale reverse time migration on GPU hardware: 79th Annual International Meeting, SEG, Expanded Abstracts, 2789–2793, doi: 10.1190/1.3255428.

Fomel, S., P. Sava, I. Vlad, Y. Liu, and V. Bashkardin, 2013, Madagascar: Open-source software project for multidimensional data analysis and reproducible computational experiments: Journal of Open Research Software, **1**, e8, doi: 10.5334/jors.ag.

Futterman, W. I., 1962, Dispersive body waves: Journal of Geophysical Research, **67**, 5279–5291, doi: 10.1029/JZ067i013p05279.

Griewank, A., and A. Walther, 2000, Algorithm 799: Revolve: An implementation of checkpointing for the reverse or adjoint mode of computational differentiation: ACM Transactions on Mathematical Software, **26**, 19–45, doi: 10.1145/347837.347846.

Guide, D., 2013, CUDA C programming guide: NVIDIA.

Guo, P., G. A. McMechan, and H. Guan, 2016, Comparison of two viscoacoustic propagators for Q-compensated reverse time migration: Geophysics, **81**, no. 5, S281–S297, doi: 10.1190/geo2015-0557.1.

Irving, J. D., and R. J. Knight, 2003, Removal of wavelet dispersion from ground-penetrating radar data: Geophysics, **68**, 960–970, doi: 10.1190/1.1581068.

Jaros, J., A. P. Rendell, and B. E. Treeby, 2016a, Full-wave nonlinear ultrasound simulation on distributed clusters with applications in high-intensity focused ultrasound: The International Journal of High Performance Computing Applications, **30**, 137–155, doi: 10.1177/1094342015581024.

Jaros, J., B. E. Treeby, and A. P. Rendell, 2012, Use of multiple GPUs on shared memory multiprocessors for ultrasound propagation simulations: Proceedings of the 10th Australasian Symposium on Parallel and Distributed Computing 127, Australian Computer Society, Inc., 43–52.

Jaros, J., F. Vaverka, and B. E. Treeby, 2016b, Spectral domain decomposition using local Fourier basis: Application to ultrasound simulation on a cluster of GPUs: Supercomputing Frontiers and Innovations, **3**, 40–55.

Kalimeris, K., and O. Scherzer, 2012, Photoacoustic imaging in attenuating acoustic media based on strongly causal models: Mathematical Methods in the Applied Sciences, **36**, 2254–2264, doi: 10.1002/mma.v36.16.

Kjartansson, E., 1979, Constant Q-wave propagation and attenuation: Journal of Geophysical Research Solid Earth, **84**, 4737–4748, doi: 10.1029/JB084iB09p04737.

Kolsky, H., 1956, LXXI. The propagation of stress pulses in viscoelastic solids: Philosophical Magazine, **1**, 693–710, doi: 10.1080/14786435608238144.

Li, Q., H. Zhou, Q. Zhang, H. Chen, and S. Sheng, 2016, Efficient reverse time migration based on fractional Laplacian viscoacoustic wave equation: Geophysical Journal International, **204**, 488–504, doi: 10.1093/gji/ggv456.

Liao, Z., K. Huang, B. Yang, and Y. Yuan, 1984, A transmitting boundary for transient wave analyses: Science in China Series A-Mathematics, Physics, Astronomy and Technological Science, **27**, 1063–1076.

Liu, G., Y. Liu, L. Ren, and X. Meng, 2013, 3D seismic reverse time migration on GPGPU: Computers and Geosciences, **59**, 17–23, doi: 10.1016/j.cageo.2013.05.009.

Liu, H. W., H. Liu, X. L. Tong, and Q. Liu, 2012, A Fourier integral algorithm and its GPU/CPU collaborative implementation for one-way wave equation migration: Computers and Geosciences, **45**, 139–148, doi: 10.1016/j.cageo.2011.10.029.

Mainardi, F., 2010, Fractional calculus and waves in linear viscoelasticity: An introduction to mathematical models: World Scientific.

McDonal, F., F. Angona, R. Mills, R. Sengbush, R. Van Nostrand, and J. White, 1958, Attenuation of shear and compressional waves in Pierre shale: Geophysics, **23**, 421–439, doi: 10.1190/1.1438489.

Micikevicius, P., 2009, 3D finite difference computation on GPUs using CUDA: Proceedings of 2nd workshop on General Purpose Processing on Graphics Processing Units, ACM, 79–84.

Mittet, R., 2007, A simple design procedure for depth extrapolation operators that compensate for absorption and dispersion: Geophysics, **72**, no. 2, S105–S112, doi: 10.1190/1.2431637.

Mittet, R., R. Sollie, and K. Hokstad, 1995, Prestack depth migration with compensation for absorption and dispersion: Geophysics, **60**, 1485–1494, doi: 10.1190/1.1443882.

Moczo, P., and J. Kristek, 2005, On the rheological models used for time-domain methods of seismic wave propagation: Geophysical Research Letters, **32**, 5, doi: 10.1029/2004GL021598.

Nandapalan, N., J. Jaros, A. P. Rendell, and B. Treeby, 2012, Implementation of 3D FFTs across multiple GPUs in shared memory environments: 13th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT), IEEE, 167–172.

Näsholm, S., and S. Holm, 2013, On a fractional Zener elastic wave equation: Fractional Calculus and Applied Analysis, **16**, 26–50, doi: 10.2478/s13540-013-0003-1.

Rossikhin, Y. A., and M. V. Shitikova, 2010, Application of fractional calculus for dynamic problems of solid mechanics: Novel trends and recent results: Applied Mechanics Reviews, **63**, 010801, doi: 10.1115/1.4000563.

Sava, P., and I. Vlad, 2008, Numeric implementation of wave-equation migration velocity analysis operators: Geophysics, **73**, no. 5, VE145–VE159, doi: 10.1190/1.2953337.

Shen, Y., and T. Zhu, 2015, Image-based Q tomography using reverse time Q, doi: 10.1190/segam2015-5852526.1.

Shin, J., W. Ha, H. Jun, D.-J. Min, and C. Shin, 2014, 3D Laplace-domain full waveform inversion using a single GPU card: Computers and Geosciences, **67**, 1–13, doi: 10.1016/j.cageo.2014.02.006.

Strick, E., 1967, The determination of Q, dynamic viscosity and transient creep curves from wave propagation measurements: Geophysical Journal International, **13**, 197–218, doi: 10.1111/j.1365-246X.1967.tb02154.x.

Sun, J., S. Fomel, T. Zhu, and J. Hu, 2016, Q-compensated least-squares reverse time migration using low-rank one-step wave extrapolation: Geophysics, **81**, no. 4, S271–S279, doi: 10.1190/geo2015-0520.1.

Sun, J., and T. Zhu, 2015, Stable attenuation compensation in reverse-time migration: 85th Annual International Meeting, SEG, Expanded Abstracts, 3942–3947, doi: 10.1190/segam2015-5888552.1.

Sun, J., T. Zhu, and S. Fomel, 2014, Viscoacoustic modeling and imaging using low-rank approximation: Geophysics, **80**, no. 5, A103–A108, doi: 10.1190/geo2015-0083.1.

Symes, W. W., 2007, Reverse time migration with optimal checkpointing: Geophysics, **72**, no. 5, SM213–SM221, doi: 10.1190/1.2742686.

Szabo, T. L., 1994, Time domain wave equations for lossy media obeying a frequency power law: The Journal of the Acoustical Society of America, **96**, 491–500, doi: 10.1121/1.410434.

Szabo, T. L., 1995, Causal theories and data for acoustic attenuation obeying a frequency power law: The Journal of the Acoustical Society of America, **97**, 14–24, doi: 10.1121/1.412332.

Tan, S., and L. Huang, 2014, Reducing the computer memory requirement for 3D reverse time migration with a boundary-wavefield extrapolation method: Geophysics, **79**, no. 5, S185–S194, doi: 10.1190/geo2014-0075.1.

Tan, W., L. Cao, and L. Fong, 2016, Faster and cheaper: Parallelizing large-scale matrix factorization on GPUs: Proceedings of the 25th ACM International Symposium on High-Performance Parallel and Distributed Computing, 219–230.

Treeby, B. E., E. Z. Zhang, and B. T. Cox, 2010, Photoacoustic tomography in absorbing acoustic media using time reversal: Inverse Problems, **26**, 115003-20, doi: 10.1088/0266-5611/26/11/115003.

Wang, N., H. Zhou, H. Chen, M. Xia, S. Wang, J. Fang, and P. Sun, 2018a, A constant fractional-order viscoelastic wave equation and its numerical simulation scheme: Geophysics, **83**, no. 1, T39–T48, doi: 10.1190/geo2016-0609.1.

Wang, Y., 2002, A stable and efficient approach of inverse Q filtering: Geophysics, **67**, 657–663, doi: 10.1190/1.1468627.

Wang, Y., 2006, Inverse Q-filter for seismic resolution enhancement: Geophysics, **71**, no. 3, V51–V60, doi: 10.1190/1.2192912.

Wang, Y., 2009, Seismic inverse Q filtering: John Wiley and Sons.

Wang, Y., and J. Guo, 2004, Seismic migration with inverse Q filtering: Geophysical Research Letters, **31**, 163–183, doi: 10.1029/2003GL018323.

Wang, Y., X. Ma, H. Zhou, and Y. Chen, 2018b, L1-2 minimization for exact and stable seismic attenuation compensation: Geophysical Journal International, **213**, 1629–1646, doi: 10.1093/gji/ggy064.

Wang, Y., H. Zhou, H. Chen, and Y. Chen, 2018c, Adaptive stabilization for Q-compensated reverse time migration: Geophysics, **83**, no. 1, S15–S32, doi: 10.1190/geo2017-0244.1.

Wang, Y., H. Zhou, Q. Li, X. Zhao, X. Zhao, and Y. An, 2017a, Regularized Q-RTM using time-variant filtering in the k-space: 79th Annual International Conference and Exhibition, EAGE, Extended Abstracts, doi: 10.3997/2214-4609.201700676.

Wang, Y., H. Zhou, Q. Zhang, X. Zhao, Z. Zhou, and Y. An, 2017b, Wavefield reconstruction in attenuating media using time-reversal checkpointing and k-space filtering: 79th Annual International Conference and Exhibition, EAGE, Extended Abstracts, doi: 10.3997/2214-4609.201701152.

Wang, Y., H. Zhou, X. Zhao, M. Xia, and X. Cai, 2017c, The k-space Green's functions for decoupled constant-Q wave equation and its adjoint equation: 79th Annual International Conference and Exhibition, EAGE, Extended Abstracts, doi: 10.3997/2214-4609.201701153.

Yang, P., R. Brossier, L. Métivier, and J. Virieux, 2016, Wavefield reconstruction in attenuating media: A checkpointing-assisted reverse-forward simulation method: Geophysics, **81**, no. 6, R349–R362, doi: 10.1190/geo2016-0082.1.

Yang, P., J. Gao, and B. Wang, 2014, RTM using effective boundary saving: A staggered grid GPU implementation: Computers and Geosciences, **68**, 64–72, doi: 10.1016/j.cageo.2014.04.004.

Yang, P., J. Gao, and B. Wang, 2015, A graphics processing unit implementation of time-domain full-waveform inversion: Geophysics, **80**, no. 3, F31–F39, doi: 10.1190/geo2014-0283.1.

Zhang, G., and J. Gao, 2014, Time domain viscoelastic forward modeling on GPU: 84th Annual International Meeting, SEG, Expanded Abstracts, 3530–3535, doi: 10.1190/segam2014-0684.1.

Zhang, J., S. Wang, and Z. Yao, 2009, Accelerating 3D Fourier migration with graphics processing units: Geophysics, **74**, no. 6, WCA129–WCA139, doi: 10.1190/1.3223186.

Zhang, J., J. Wu, and X. Li, 2012, Compensation for absorption and dispersion in prestack migration: An effective Q approach: Geophysics, **78**, no. 1, S1–S14, doi: 10.1190/geo2012-0128.1.

Zhang, Y., P. Zhang, and H. Zhang, 2010, Compensating for visco-acoustic effects in reverse time migration: 80th Annual International Meeting, SEG, Expanded Abstracts, 3160–3164, doi: 10.1190/1.3513503.

Zhao, X., H. Zhou, Q. Li, and Y. Wang, 2017, A method to avoid the snapshots wave-fields storage in reverse time migration: 79th Annual International Conference and Exhibition, EAGE, Extended Abstracts, doi: 10.3997/2214-4609.201700679.

Zhu, T., and J. M. Harris, 2014, Modeling acoustic wave propagation in heterogeneous attenuating media using decoupled fractional Laplacians: Geophysics, **79**, no. 3, T105–T116, doi: 10.1190/geo2013-0245.1.

Zhu, T., J. M. Harris, and B. Biondi, 2014, Q-compensated reverse-time migration: Geophysics, **79**, no. 3, S77–S87, doi: 10.1190/geo2013-0344.1.