Journal of Magnetic Resonance 281 (2017) 51–65

Contents lists available at ScienceDirect

Journal of Magnetic Resonance

journal homepage: www.elsevier.com/locate/jmr

BlochSolver: A GPU-optimized fast 3D MRI simulator for experimentally compatible pulse sequences

Ryoichi Kose [a], Katsumi Kose [b,*]

[a] MRTechnology Inc, 2-1-6 B5 Sengen, Tsukuba 3050047, Japan
[b] Institute of Applied Physics, University of Tsukuba, 1-1-1 Tennodai, Tsukuba 3058573, Japan

ARTICLE INFO

Article history:
Received 12 February 2017
Revised 15 May 2017
Accepted 15 May 2017
Available online 20 May 2017

Keywords:
Magnetic resonance imaging
Simulation
Graphic processor unit
MRI simulator

ABSTRACT

A magnetic resonance imaging (MRI) simulator, which reproduces MRI experiments using computers, has been developed using two graphic-processor-unit (GPU) boards (GTX 1080). The MRI simulator was developed to run according to pulse sequences used in experiments. Experiments and simulations were performed to demonstrate the usefulness of the MRI simulator for three types of pulse sequences, namely, three-dimensional (3D) gradient-echo, 3D radio-frequency spoiled gradient-echo, and gradient-echo multislice with practical matrix sizes. The results demonstrated that the calculation speed using two GPU boards was typically about 7 TFLOPS and about 14 times faster than the calculation speed using CPUs (two 18-core Xeons). We also found that MR images acquired by experiment could be reproduced using an appropriate number of subvoxels, and that 3D isotropic and two-dimensional multislice imaging experiments for practical matrix sizes could be simulated using the MRI simulator. Therefore, we concluded that such powerful MRI simulators are expected to become an indispensable tool for MRI research and development.

1. Introduction

Various magnetic resonance imaging (MRI) simulators, which reproduce MRI experiments using computers, have been proposed since the 1980s [1–15]. Among them, several MRI simulators were designed to run in parallel to accelerate the processing speeds [7,11–15]. However, mainly because of their inadequate processing speed, they have not been widely used in practical research and development areas such as pulse sequence development or image artifact analysis.

There are several problems with these proposed MRI simulators. The first is that it is difficult to simulate the large matrix (e.g., 256^3) images required for three dimensional (3D) or multislice image acquisitions. To overcome this problem, the use of PC clusters [7,11] and even a supercomputer [12] has been proposed, but these are not fully acceptable solutions for large matrix 3D MRI simulations because the processing speeds were not sufficient. In recent years, the rapid development of graphic processor units (GPUs) has made it possible to use highly parallel data processing (>1000 streams) in small laboratories and even for personal use [16]. Because MRI simulator calculations are well suited to highly parallel computing, MRI simulators utilizing GPU boards are expected to overcome the above problems [13,14]. In 2014, Xanthis et al. published a GPU-based MRI simulator (MRISIMUL) and demonstrated 31-228 times processing speed compared to CPU-based MRI simulations [13]. However, the comparison of the processing speed was relative one between the GPUs and the CPUs, and no absolute processing speed measured in FLOPS (floating operations per second), essential to numerical simulations, was presented.

A second problem is that it is difficult to reproduce a precise nuclear magnetic resonance (NMR) signal for continuous objects using a finite number of isochromats. In other words, if the nuclear magnetization of a voxel is calculated using a small number of isochromats, pseudo-echoes are frequently produced, resulting in observable image artifacts. To overcome this problem, a technique to use a large number of isochromats comparing to the number of voxels of the reconstructed image has been proposed. However, because in fast gradient echo sequences the coherence of the nuclear magnetization over successive excitation pulses makes sinusoidal modulation of the transverse magnetization within a voxel, division of a voxel into many subvoxels along the readout direction is a useful and straightforward approach to increasing the number of isochromats. A third problem is that it is difficult to compare experimental results with those of their simulations because there is no useful interface between them. One straightfor-

* Corresponding author.
E-mail address: kose@bk.tsukuba.ac.jp (K. Kose).

http://dx.doi.org/10.1016/j.jmr.2017.05.007
1090-7807/

ward solution to this problem is to develop the MRI simulator to run according to the experimentally used pulse sequences.

In this study, we developed a GPU-optimized MRI simulator for large matrix 3D images with easily adjustable numbers of subvoxels, measured the absolute processing speed in FLOPS, and quantitatively compared the processing speed with that of MRI simulations performed using the latest CPUs (two 18-core Xeons). Because our simulator was developed to run according to experimentally compatible pulse sequences, we demonstrated its usefulness by comparing MRI simulations with corresponding MRI experiments using identical pulse sequences.

2. Materials and methods

2.1. Formulation of the simulator

In this paper, the MRI simulation was formulated using the Bloch equation [17], and the molecular diffusion effect for a one-voxel object was calculated using the Bloch–Torrey equation [18].

The evolution of a nuclear magnetization vector $\overrightarrow{M} = (M_x, M_y, M_z)^T$ in a magnetic field $B_0(\overrightarrow{r}, t)$ is described in a rotating frame of reference using the Bloch equation as

$$\frac{d\overrightarrow{M}}{dt} = \gamma\overrightarrow{M} \times \overrightarrow{B_e} - \begin{pmatrix} M_x/T_2 \\ M_y/T_2 \\ (M_z - M_0)/T_1 \end{pmatrix}, \tag{1}$$

where γ, $\overrightarrow{B_e}$, M_0, T_1, and T_2 are the gyromagnetic ratio of the nucleus, effective magnetic field in the rotating frame of reference, longitudinal magnetization in the thermal equilibrium state, longitudinal relaxation time, and transverse relaxation time, respectively. $\overrightarrow{B_e}$ comprises the inhomogeneous magnetic field, the gradient magnetic fields, and the radio-frequency (RF) magnetic fields. It is expressed as

$$\begin{aligned}\overrightarrow{B_e}(x, y, z, t) = {} & \Delta B_z(x, y, z)\hat{z} + (G_x(t)x + G_y(t)y + G_z(t)z)\hat{z} \\ & + B_{1x}(t)\hat{x} + B_{1y}(t)\hat{y},\end{aligned} \tag{2}$$

where $\Delta B_z(x, y, z)$ is the z component of the inhomogeneous magnetic field, $G_x(t)$, $G_y(t)$, and $G_z(t)$ are the magnetic field gradients, and $B_{1x}(t)$ and $B_{1y}(t)$ are the two orthogonal components of the RF magnetic field in the rotating frame of reference.

When an RF pulse of arbitrary shape and phase is applied, the evolution of a magnetization vector can be approximated using the successive application of short square RF pulses of Δt duration. If a rotation vector (v_x, v_y, v_z) is defined as

$$(v_x, v_y, v_z) = \gamma\overrightarrow{B_e}\Delta t \tag{3}$$

and $$\theta = \sqrt{(v_x)^2 + (v_y)^2 + (v_z)^2}, \tag{4}$$

the rotation matrix **R** for the short square pulse can be written using Rodrigues' rotation formula as

$$\mathbf{R} = \frac{1}{\theta^2}\begin{pmatrix} v_x^2(1-\cos\theta) + \theta^2\cos\theta & v_x v_y(1-\cos\theta) - \theta v_z\sin\theta & v_x v_z(1-\cos\theta) + \theta v_y\sin\theta \\ v_x v_y(1-\cos\theta) + \theta v_z\sin\theta & v_y^2(1-\cos\theta) + \theta^2\cos\theta & v_y v_z(1-\cos\theta) - \theta v_x\sin\theta \\ v_x v_z(1-\cos\theta) - \theta v_y\sin\theta & v_y v_z(1-\cos\theta) + \theta v_x\sin\theta & v_z^2(1-\cos\theta) + \theta^2\cos\theta \end{pmatrix}. \tag{5}$$

Therefore, the evolution of the magnetization vector for an arbitrary RF pulse can be calculated using successive applications of the rotation matrix **R**.

When there is no RF pulse, the evolution of the magnetization vector can be written as

$$\begin{pmatrix} M_x(t+\Delta t) \\ M_y(t+\Delta t) \\ M_z(t+\Delta t) \end{pmatrix} = \begin{pmatrix} E_2\cos\varphi & -E_2\sin\varphi & 0 \\ E_2\sin\varphi & E_2\cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} M_x(t) \\ M_y(t) \\ M_z(t) \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 - E_1 \end{pmatrix} M_0, \tag{6}$$

where Δt is a time difference, $E_1 = \exp(-\Delta t/T_1)$, $E_2 = \exp(-\Delta t/T_2)$, and the angle

$$\varphi = \int_t^{t+\Delta t} (\overrightarrow{B_e})_z dt \tag{7}$$

To summarize the above formulation, our MRI simulation is categorized into two cases, namely with and without an RF pulse, and the large number of isochromats in the voxels are calculated according to a time chart of the pulse sequences. The NMR signal from the imaging object can be obtained by calculating the sum of the transverse components of the isochromats during the data-acquisition periods. The MR image can then be obtained through image reconstruction using a set of NMR signals obtained from the above calculation.

When molecular diffusion effects cannot be neglected, the Bloch equation should be replaced by the Bloch–Torrey equation [18] as

$$\frac{d\overrightarrow{M}}{dt} = \gamma\overrightarrow{M} \times \overrightarrow{B_e} - \begin{pmatrix} M_x/T_2 \\ M_y/T_2 \\ (M_z - M_0)/T_1 \end{pmatrix} + D\nabla^2\overrightarrow{M}, \tag{8}$$

where D is the isotropic diffusion coefficient. Although molecular diffusion effects were described using a more general formula in the original Bloch–Torrey equation, we use the simplest form to calculate signal attenuation in a voxel. The above equation was calculated using the following approximation for a voxel as

$$D\frac{\partial^2\overrightarrow{M}_i}{\partial x^2} = D\frac{\overrightarrow{M}_{i+1} + \overrightarrow{M}_{i-1} - 2\overrightarrow{M}_i}{(\Delta x)^2}, \tag{9}$$

where i is the index of the subvoxel in the voxel and Δx is the width of the subvoxel.

2.2. Computer system for the MRI simulation

In this study, we used two GPU boards (GTX 1080, nVIDIA, Santa Clara, USA) installed in a PC (CPU: Core i7-5960X, clock frequency: 3.0 GHz, RAM: 64 GB) running under the Windows 10 operating system (Microsoft, Seattle, USA). The GTX 1080 has 2560 single-precision CUDA (Compute Unified Device Architecture) cores, 8 GB RAM, and a 1733 MHz clock frequency. Peak performance for single-precision floating-point operations is 8873 GFLOPS at the clock frequency of 1733 MHz. To compare the processing speed with that of a CPU, we used a conventional workstation (CPU: Xeon E5-2699v3 × 2, 36 cores, clock frequency: 2.3 GHz, RAM: 64 GB)

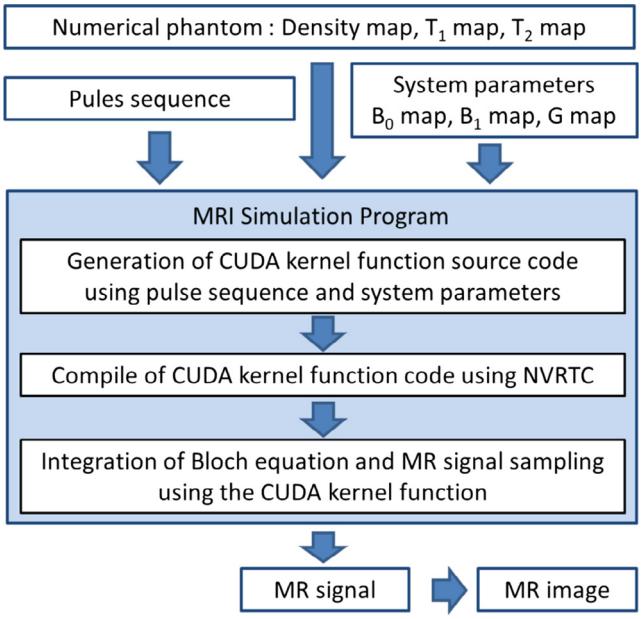

Fig. 1. Structure of BlochSolver. The input data are the numerical phantom, the pulse sequence, and the system parameters. CUDA kernel function code, generated automatically using the pulse sequence and the system parameters, is compiled to execution code for the GPU using NVRTC (NVIDIA runtime compilation library). The Bloch equation is integrated to calculate the NMR signal for the MR image.

running under the Windows 10 operating system. We used Visual C++ 2015 (Microsoft) and CUDA 8.0 (nVIDIA) for the development of the MRI simulator.

2.3. Implementation of the simulation program

Fig. 1 shows the structure of the MRI simulation program developed in this study. The CUDA kernel function source codes for the MRI simulation performed in the GPU boards are "automatically" generated from the pulse sequence file, numerical phantoms, and system parameters. Subsequently, the execution codes for the GPU were generated using the NVIDIA runtime compilation library (NVRTC). This automatic execution code generation mechanism enabled our MRI simulator to be compatible with any experimentally used pulse sequences. We named this program "BlochSolver."

Fig. 2 shows the graphical user interface (GUI) of BlochSolver developed for the input of numerical phantoms, system parameters, and pulse sequences. The most significant input for the dialog boxes shown in the upper right of Fig. 2 is the matrix size of the numerical phantom. A voxel is further divided into subvoxels, the number of which can be arbitrarily changed in the x, y, and z directions. The matrix size and the maximum number of subvoxels are not restricted, provided the associated data matrices (described later) can be stored in the GPU memory (16 GB). Although the matrix size of the numerical phantom is independent of that of the acquired image, it is convenient to use an identical matrix size and adjust the number of subvoxels to describe the spatial changes of the numerical phantom within the voxel.

The numerical phantom consists of a nuclear (proton) density map, a T_1 map, and a T_2 map. These three maps are input as three data files described in terms of single-precision floating-point numbers. We do not include a diffusion coefficient map because the diffusion effect drastically reduces the calculation speed. We also omit a resonance frequency map because we use a static magnetic field map of the system parameters instead. The matrix size of the numerical phantoms is multiplied by the number of subvoxels and the values in the matrices are calculated using linear interpolation of the original values.

The system parameters comprise a static magnetic field map (B_0), a transmission RF field map (B_1 transmission), a reception or sensitivity RF field map (B_1 reception), and field gradient maps (Gx, Gy, and Gz). These six system parameter maps are input as six data files described using single-precision floating-point numbers with the same matrix size as that of the numerical phantom. The matrix size of the system parameter maps is multiplied by the number of subvoxels and the values in the matrices are calculated using linear interpolation of the original values.

As described above, BlochSolver requires 12 3D data matrices comprising three numerical phantoms, six system parameters, and three magnetization components for the imaging object. Therefore, if a large image matrix (e.g., $256 \times 512 \times 1024$, including subvoxels) is used, the total data size may exceed the GPU memory size (16 GB). We have addressed this problem by using the following technique. Because the voxel for which the proton density is zero does not contribute to the NMR signal, the 3D image arrays can be rearranged into one-dimensional arrays that exclude such voxels. In this way, the memory size and processing time for BlochSolver are significantly reduced. However, the size of the 3D data matrices sometimes exceeds the GPU memory size (16 GB) even if this memory and calculation time saving technique is used. In such case, the MRI simulation was performed by partitioning the 3D data using the main memory of the PC (128 GB or more).

Fig. 3 shows an example of a pulse sequence used in this study. The pulse sequence comprises two parts. The first part represents global parameters such as TR, NX, and NR (see Fig. 3), which control the overall behavior of the sequence. The second part represents the time series of the events, such as the RF pulse, gradient amplitude, and start of data acquisition [19]. The format of this sequence was originally developed for a pulse programmer using a digital signal-processor board with 100-ns resolution. It has been used for many compact MRI systems developed in our laboratory [20–24]. The same format has also been used for a 128-bit parallel-communication pulse-programmer using a PC with a 1-μs resolution [25], which was used in the experiments in this study. The sequence format includes many user-defined events such as the RF pulse shape, RF pulse phase, and phase encoding tables. BlochSolver reads this sequence file and performs the MRI simulation according to the time series described in the file.

Fig. 4 shows signal acquisition windows that display real, imaginary, and absolute value NMR signals in real time during the MRI simulation. This signal display window is very useful for the detection of any pseudo-echo signals and for adjusting the timing of the pulse sequence.

The most time consuming or time critical part of the MRI simulation was summation of all the transverse components of the magnetization vectors for NMR signal calculation performed at every sampling point. The evolution of the spatially distributed magnetization vectors were calculated highly in parallel in the streaming multiprocessors (SM) of the GPU. However, the fast parallel summation of the transvers components of the isochromats is not compatible with the fast parallel calculation of the evolution of the isochromats because the memory size for the whole of the isochromats is very large (e.g. $256^3 \times 4 \times 3$ bytes ~200 MB) and stored in the global memory (several hundreds of the clock time for the memory access) of the GPU. In this study we developed a special calculation technique that enabled both fast parallel calculation of the isochromat evolution and fast parallel summation of the transverse components of the isochromats within the register file (256 kB for each SM, no delay time for the memory access).

2.4. MRI system configuration

We used a home-built MRI system using a 1.5 T superconducting magnet. The superconducting magnet (JMTB–1.5/280/SSE,

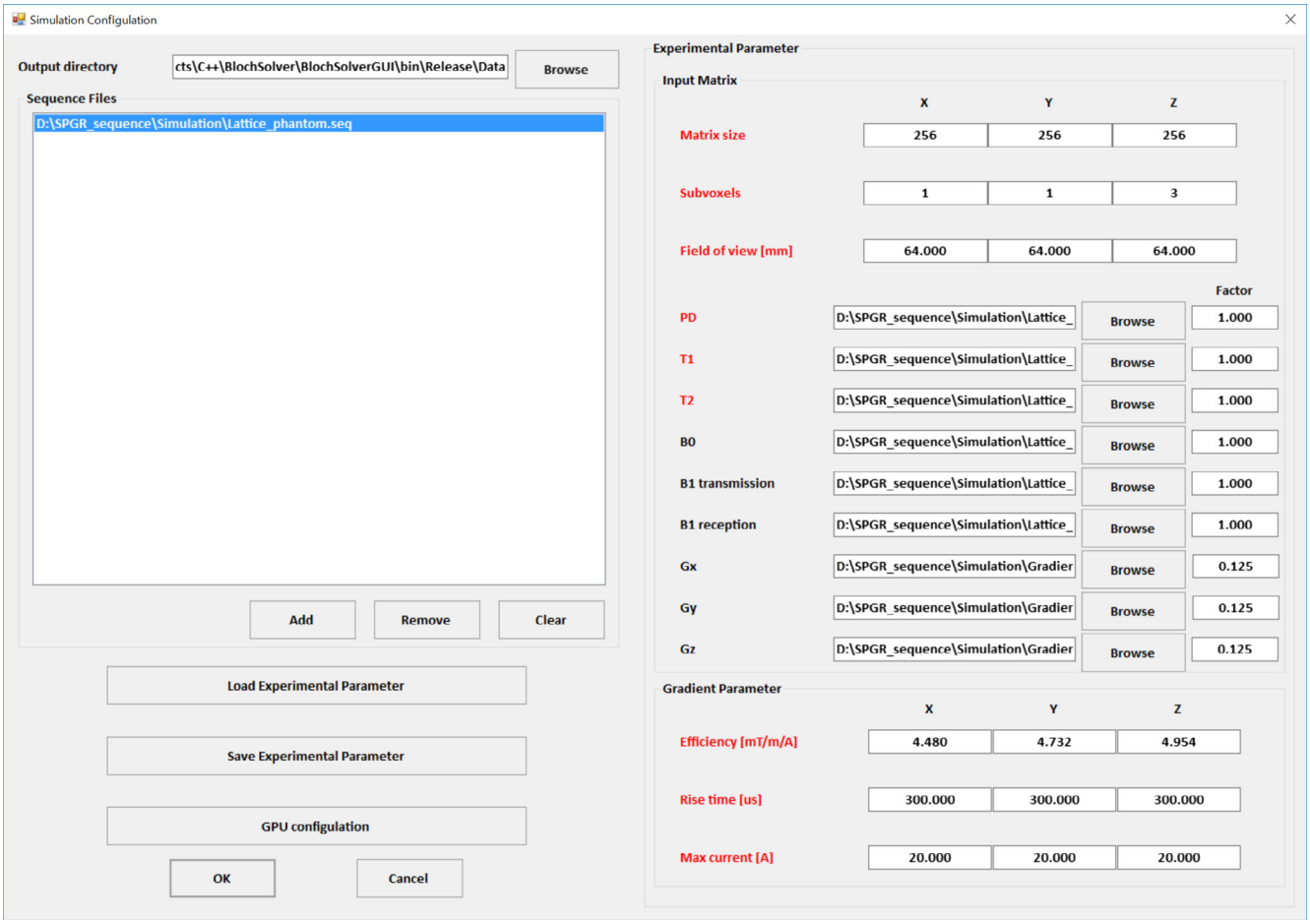

Fig. 2. Graphical user interface (GUI) for the parameter inputs of BlochSolver. The dialog boxes shown on the right side of the GUI refer to the matrix size of the numerical phantom, the numbers of subvoxels, field of view, proton density, T_1 map, T_2 map, B_0 map, B_1 transmission map, B_1 sensitivity map, maps representing gradient field nonlinearity, and gradient coil parameters. Dialog boxes written in red are mandatory and those in black are optional. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

JASTEC, Kobe, Japan) has a horizontal room temperature bore (diameter = 280 mm, length = 521 mm) and a 160-mm-diameter spherical homogeneous region (<50 ppm). We used two home-built gradient coil sets with 220- and 87-mm-diameter inner bores. The efficiencies of the gradient coils in the large-bore gradient coil set were 0.787, 0.862, and 0.638 mT/m/A for the Gx, Gy, and Gz coils, respectively. The efficiencies of the gradient coils in the small bore gradient coil set were 4.48, 4.73, and 4.95 mT/m/A for the Gx, Gy, and Gz coils, respectively. For the small bore gradient coil set, a second-order shim coil set with a 122-mm-diameter inner bore was used.

We used two home-built RF coils: a solenoid coil of 130-mm diameter and 120-mm length and a linear-drive 8-leg birdcage coil of 64-mm diameter and 64-mm length. The MRI console was constructed using a digital transceiver (DTRX-6, MRTechnology, Tsukuba, Japan). The data-acquisition software was Samper 7D (MRTechnology, Tsukuba, Japan) running under the Windows 10 operating system.

Fig. 5 shows the real MRI phantoms used for the MRI experiments. Numerical phantoms were made to simulate the real MRI phantoms using C++ programs. The matrix sizes for the numerical phantoms corresponding to Fig. 5(a)–(d) were $256 \times 256 \times 16$, $512 \times 512 \times 64$, $256 \times 256 \times 32$, and $256 \times 256 \times 512$, respectively.

When a cylindrical object (radius = R) with homogeneous magnetic susceptibility μ_i is placed in a uniform material with homogeneous magnetic susceptibility μ_e such that the cylindrical axis is perpendicular to a homogeneous magnetic field B_0 (parallel to z), the component of the induced magnetic field B_z collinear with the external magnetic field is

$$B_z = B_0\left(1 - \frac{\mu_e - \mu_i}{\mu_e + \mu_i} R^2 \frac{z^2 - x^2}{r^4}\right), \tag{10}$$

where $r^2 = x^2 + z^2$ [26]. When the phantom shown in Fig. 5(a) was placed such that the cylindrical axis was perpendicular to the static magnetic field, similar magnetic field distribution was produced. However, because the susceptibilities of the $CuSO_4$ water solution, air, and the plastic bottle were unknown, we treated the coefficient $(\mu_e - \mu_i)R^2/(\mu_e + \mu_i)$ as one adjustable parameter in the MRI simulation.

Similarly, when a spherical object (radius = R) with homogeneous magnetic susceptibility μ_i is placed in a uniform material with homogeneous magnetic susceptibility μ_e and a uniform magnetic field B_0 (parallel to z) is applied, the component of the induced magnetic field B_z collinear with the external magnetic field is

$$B_z = B_0\left(1 - \frac{\mu_e - \mu_i}{2\mu_e + \mu_i} R^3 \frac{2z^2 - x^2 - y^2}{r^5}\right), \tag{11}$$

where $r^2 = x^2 + y^2 + z^2$ [26]. When the phantom shown in Fig. 5(b) was placed in a static magnetic field, a similar magnetic field

```
:TR 20
:NX 1
:NR 1024
:N1 8192
:N2 1
:S1 128
:S2 128
:DW 5
:DU 100
:NI 1
:LK 0

00.000.050.0 PH 0000<-v5[C:¥MRT¥sequence¥kose¥Gradient_echo¥RF_phase117.txt]
00.000.100.0 RF 000F=[C:¥MRT¥sequence¥Kose¥RF_pulse¥hard_30deg_40usec.txt]
00.000.180.0 GX 5000
00.000.280.0 GY 8000<-v5[C:¥MRT¥Sequence¥Kose¥Gradient_echo¥gy_encode.txt]
00.000.380.0 GZ 8000<-v5[C:¥MRT¥Sequence¥Kose¥Gradient_echo¥gz_encode.txt]
00.001.273.0 GY 8000
00.001.329.0 GZ 8000
00.002.170.0 GX 9A37
00.003.540.0 AD 0000
00.008.960.0 GY 8000<-v5[C:¥MRT¥Sequence¥Kose¥Gradient_echo¥gy_rewind.txt]
00.009.060.0 GZ 8000<-v5[C:¥MRT¥Sequence¥Kose¥Gradient_echo¥gz_rewind.txt]
00.009.953.0 GY 8000
00.010.009.0 GZ 8000
00.011.420.0 GX 8000
```

Fig. 3. Example of a pulse sequence text file describing an RF spoiled gradient-echo sequence (SPGR). The first 11 rows represents control parameters for the whole pulse sequence: repetition time (TR), number of signal averages (NX), number of sampling points (NR), number of the first encoding step (N1), number of second encoding step (N2), step size for the first encoding gradient (S1), step size for the second encoding gradient (S2), dwell time for signal sampling (DW), number of dummy scans before data acquisition (DU), number of images for acquisition (NI), and NMR lock mode (NK). The 14 rows behind the first 11 rows represent the time series for the pulse sequence in units of 100 ns. The first column shows the sequence event time written in the unit of second, millisecond, microsecond, and 100 ns. The second column shows kinds of sequence events such as PH (transmitter RF pulse phase), RF (shape of the RF pulse), GX, GY, and GZ (gradient coil currents), and AD (acquisition start). <-v5 shows references to external files that describe tables of RF phase values and gradient currents.

distribution was produced. Again, because the susceptibility of the $CuSO_4$ water solution, air, and the plastic sphere were unknown, we treated the coefficient $(\mu_e - \mu_i)R^3/(2\mu_e + \mu_i)$ as one adjustable parameter in the MRI simulation.

2.5. B_1 field distribution in the birdcage coil

When a birdcage coil is driven linearly, the direction of the RF magnetic field (B_1) is constant. The distribution of B_1 can be approximated by using eight linear currents on the cylindrical surface of the birdcage coil (diameter = 64 mm) placed parallel to the cylindrical axis [27]. Fig. 6 shows a 54-mm diameter B_1 map calculated within the cylindrical region. This B_1 map was used for MRI simulation of the multislice imaging.

2.6. Comparison of GPU and CPU processing speeds

Processing speeds of the GPUs (GTX 1080 × 2) and CPUs (Xeon E5-2699v3 × 2) for the MRI simulation were compared using an identical MRI simulation; the matrix size was 256 × 256 × 32, the number of subvoxels was 65, and the pulse sequence was the 3D RF spoiled gradient echo (SPGR) sequence (TR = 20 ms, TE = 6 ms) [28].

The CPU programs were optimized using the OpenMP parallel computing technique for the two 18-core Xeon CPUs, AVX (advanced vector extension) 2 SIMD (single instruction multiple data) parallel operations (256-bit parallel operations), implementation of a fast customized routine of trigonometric and exponential functions for the AVX 2, and local-loop size optimization for the 32 kbyte L1 cashe memory.

To compare the processing speeds between BlochSolver and the GPU-MRI simulator previously reported [13], a typical 2D single slice gradient-echo MRI simulation was performed using the identical number of isochromats (150 × 150 × 60).

3. Results

3.1. Susceptibility phantom

Fig. 7 shows cross-sectional images obtained by simulations and experiments of the phantom shown in Fig. 5(a). Fig. 7(a)–(c) show central cross-sections selected from the 3D image datasets obtained by the simulation using the 256 × 256 × 16-voxel numerical phantom and 1 × 1, 2 × 2, and 4 × 4 in-plane subvoxels. The calculation times for the 3D image datasets were 2.28, 4.81, and 15.20 s, respectively. Fig. 7(d) and (e) are the difference images between Fig. 7(a) and (b), and between Fig. 7(b) and (c), respectively. Fig. 7(f) is the central cross-section acquired with the experiment. These images show that the intravoxel phase dispersion caused by the inhomogeneity of the magnetic field can be described by using an appropriate number of subvoxels.

Fig. 7 shows cross-sectional images obtained by simulations and experiments of the phantom shown in Fig. 5(b). Fig. 8(a)–(c) show typical cross-sections selected from the 3D image datasets acquired with the gradient-echo sequence (TR = 200 ms, TE = 48 ms, image matrix = 512 × 512 × 64). Fig. 8(d)–(f) show corresponding cross-sections selected from the 3D image datasets obtained by the simulation, using the 512 × 512 × 64-voxel numerical phantom and 4 × 4 × 4 subvoxels. The total matrix size including the subvoxels was 2048 × 2048 × 256 (= 1,073,731,824 or 1 Gigavoxels). Because the memory capacity required for this

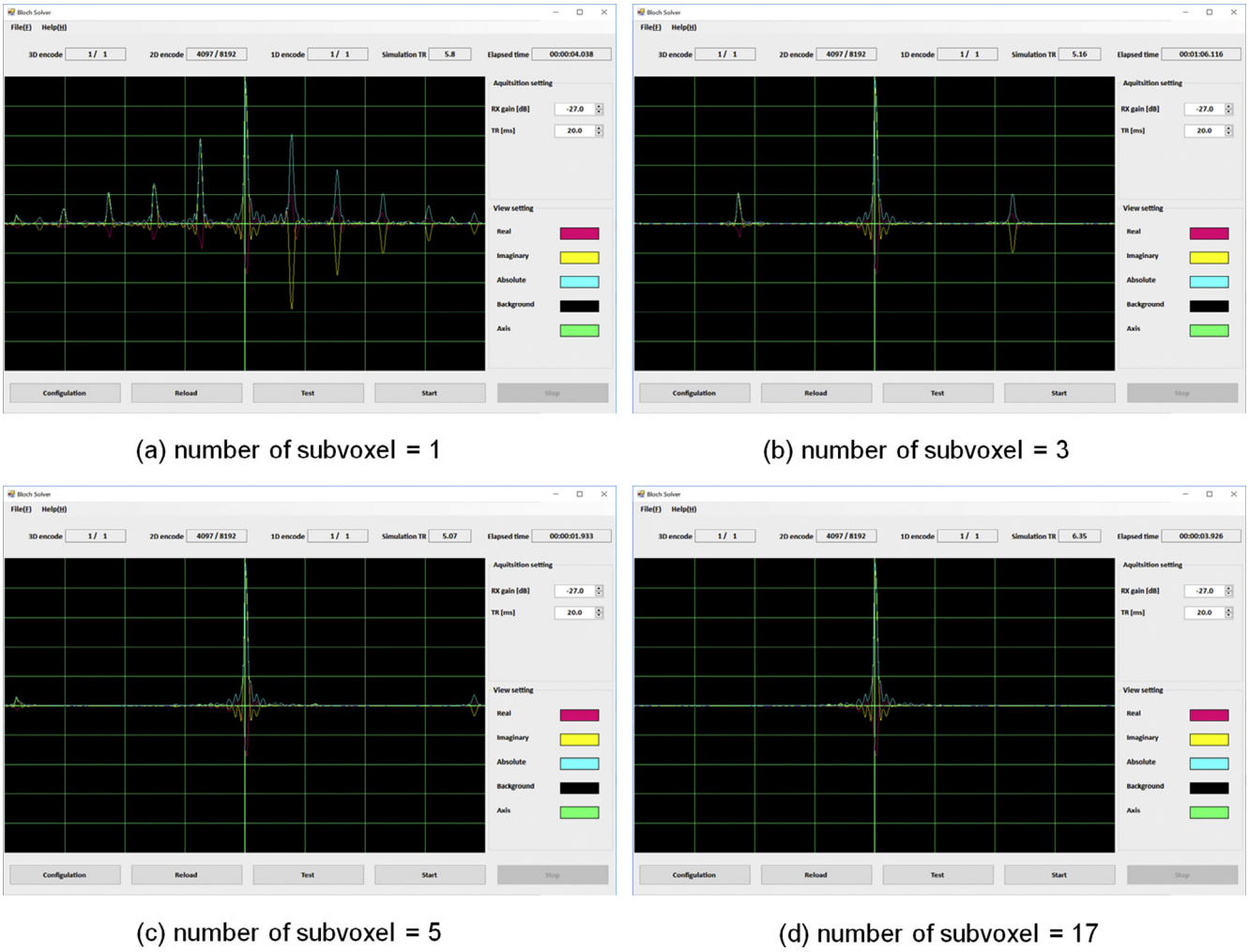

Fig. 4. Signal display windows of BlochSolver. BlochSolver displays the NMR signal computed by the GPU in real time. The phase encoding steps, calculation TR, and elapsed time are also displayed in real time. (a)–(d) are displaying NMR signals passing through the center of the k-space when the RF spoiled gradient echo sequence (TR = 20 ms, TE = 6 ms) was applied to the phantom shown in Fig. 5(c) with the number of subvoxels = 1, 3, 5, and 17. When the number of subvoxels was insufficient (1–5), pseudo-echoes were observed but disappeared if the number of subvoxels was sufficient (=17). The real, imaginary, and absolute NMR signals are displayed with different colors.

calculation would exceed the memory size of the two GPUs (16 GB), even if the memory and calculation time saving technique described in Section 2.3 were used, the numerical phantom was divided into 64 parts. NMR signals for the 64 parts were calculated separately with BlochSolver and then combined to synthesize the overall MRI signal. The calculation time for the 3D image dataset was 3.83 h. By comparing the upper and lower images in Fig. 8, it can be seen that the signal change caused by the magnetic field distribution near the sphere was clearly reproduced, although there was considerable static magnetic field inhomogeneity.

3.2. RF spoiled gradient-echo sequences

Fig. 9 shows the variations in the central cross-section acquired using the SPGR sequence for the phase increment angle $\Phi = 117^\circ$ while the number of subvoxels in the readout direction was changed from 1 to 129. When the number of subvoxels was small, serious artifacts were observed. However, the artifacts decreased with the increase in the number of subvoxels and were absent altogether when it exceeded 65.

To evaluate the artifacts quantitatively, we calculated the mean error E_n for the SPGR images when $\Phi = 0^\circ$ (conventional gradient-echo) (TR = 20 and 2000 ms) and 117° (TR = 20 ms) as

$$E_n = \frac{\sum_{i,j,k}|I_n(i,j,k) - I_{257}(i,j,k)|}{\sum_{i,j,k}|I_{257}(i,j,k)|}, \tag{12}$$

where $I_n(i,j,k)$ is the image intensity of the SPGR images at the (i,j,k) index of the image matrix when the number of subvoxels is n ($1 \leqslant n \leqslant 129$).

Fig. 10 shows E_n plotted against n. This graph clearly shows that the mean error is drastically reduced with the increase of the number of subvoxels and depends on TR and Φ. When TR = 20 ms ($\ll T_2$) and Φ is 117°, the mean error is less than 0.1% for more than 65 subvoxels. When TR = 20 ms ($\ll T_2$) and $\Phi = 0^\circ$, the mean error is less than 0.1% for more than 17 subvoxels. When TR = 2000 ms ($\gg T_2$) and $\Phi = 0^\circ$, the mean error is less than 0.1% even for more than 5 subvoxels. This result clearly shows that a sufficient number of subvoxels is required to describe the spatial modulation of the magnetization in the voxels caused by the coherence of the transverse magnetization for short TR sequences [29,30].

Fig. 11 shows typical central cross-sectional images acquired with the SPGR sequences when the phase increment angle Φ was 0°, 30°, 117°, 120°, and 180°. The images shown in the upper and lower rows are obtained by the experiment and simulation (without molecular diffusion effect), respectively (the number of subvoxels = 65). Although the contrasts in the images acquired

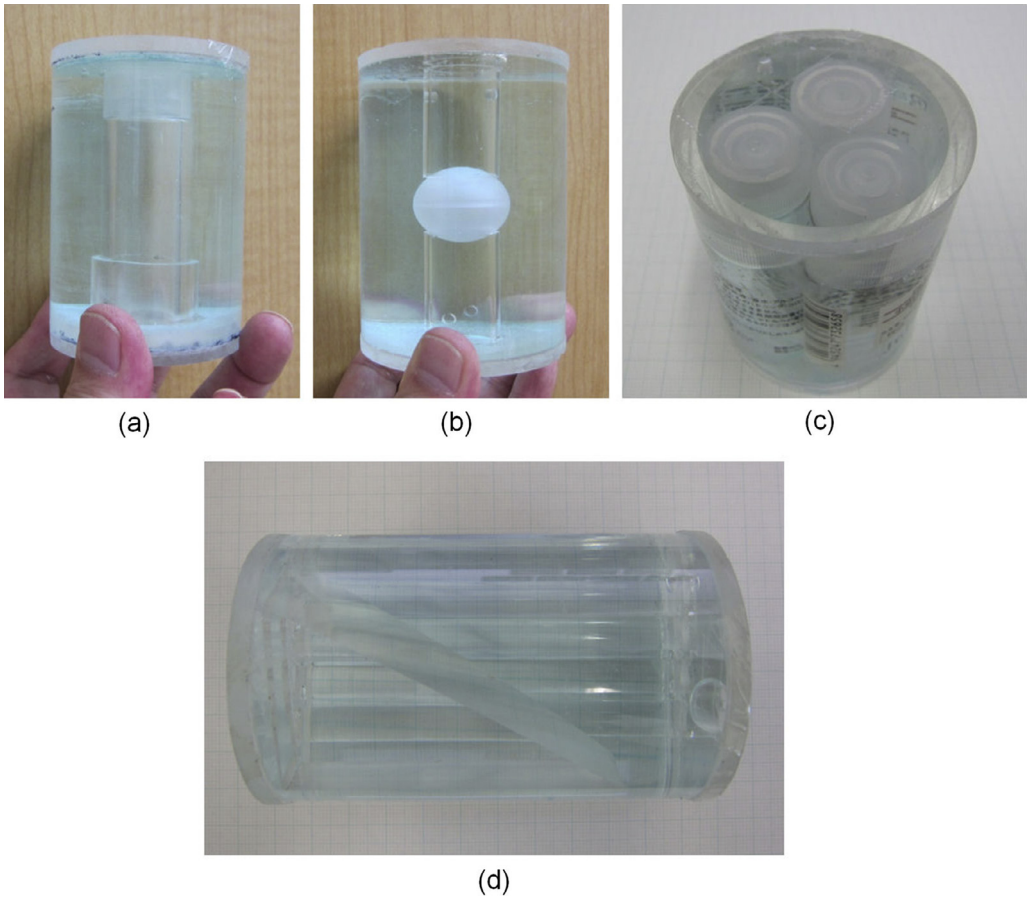

Fig. 5. Phantoms for the MRI experiments. (a and b) Susceptibility phantoms with an air-filled cylinder (diameter = 21 mm, length = 80 mm) and an air-filled sphere (diameter = 25.4 mm). The cylinder was fixed at the center of the container filled with $CuSO_4$ water solution ($T_1 \sim T_2 \sim 100$ ms). The sphere was supported with acrylic pipes. (c) Relaxation time phantom comprising three plastic bottles (OD = 21 mm, ID = 20 mm, length = 60 mm) filled with different density $CuSO_4$ water solution in a cylindrical container filled with baby oil ($T_1 \sim 186$ ms, $T_2 \sim 80$ ms). The T_1 ($\sim T_2$) values for the $CuSO_4$ water solution in the bottles were 114, 244, and 341 ms, respectively. (d) Multislice phantom. A right triangle acrylic block and eight round bars were fixed in a cylindrical acrylic container (OD = 58 mm, ID = 54 mm, length = 90 mm) filled with $CuSO_4$ water solution ($T_1 \sim T_2 \sim 120$ ms).

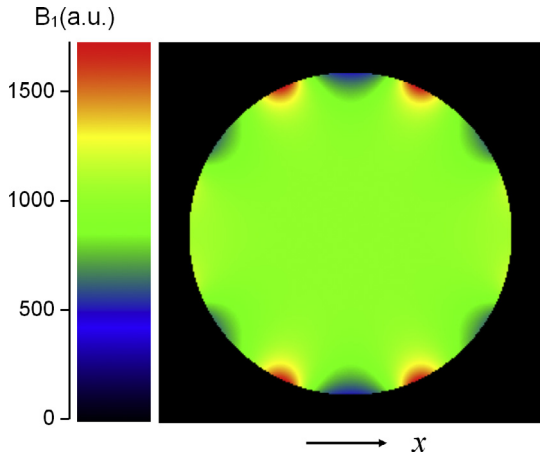

Fig. 6. Theoretically calculated B_1 map of the linear-drive birdcage coil. The direction of the RF magnetic field is parallel to the x direction. The field of view is $(64\ mm)^2$. Diameter of the birdcage coils is 64 mm. The diameter of the circle shown above is 54 mm.

by the experiment and by the simulation seem similar, they are not identical. In particular, the large image intensity difference among three bottles shown in the experimental images for $\Phi = 120$ and 180° significantly decreased in the simulated images. The difference would be caused by the molecular diffusion effect to be shown in Fig. 12.

Fig. 12(a) and (b) show image intensities averaged over all parts of the SPGR images acquired by the experiment and by the simulation plotted against the phase increment angle Φ. It took about 18 h to acquire the 3D ($256 \times 256 \times 32$ voxel) images in the experiments and about 17 h to calculate the 3D images for all angles from 0° to 359° by 1° increments. Although the peaks observed in the simulations were very sharp, those observed in the experiments were less sharp.

Fig. 12(c) shows the intensity variation for a voxel calculated using the Bloch–Torrey equation for the SPGR sequences. For the diffusion calculation, 128 subvoxels (2-μm width) in the signal readout direction for the voxel (0.25 mm width in the readout direction) were used. Diffusion coefficients of 3.16×10^{-3} and $8.6 \times 10^{-5}\ mm^2/s$ for the water and the baby oil were used in the simulation because the temperature of the phantom was raised to about 35 °C in the experiments. This graph reproduces well the image intensity acquired by the experiment, which demonstrates the importance of the diffusion effect on the image contrast for SPGR sequences [31].

3.3. Multislice imaging

Fig. 13(a) shows multislice images of the phantom shown in Fig. 5(d) acquired with the gradient echo multislice sequence.

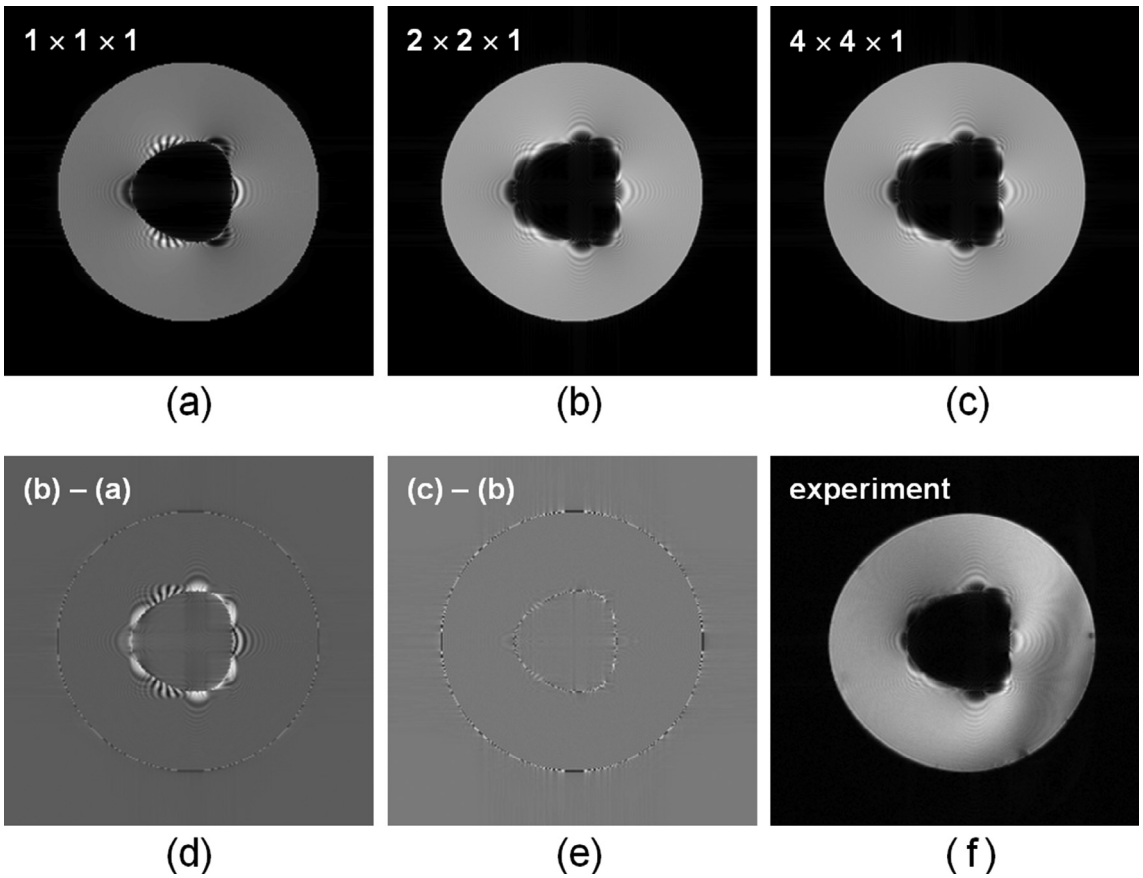

Fig. 7. Cross-sectional images of the air-filled cylindrical phantom. The matrix size of the numerical phantom was 256 × 256 × 16. (a) Simulation with 1 × 1 × 1 subvoxels. (b) Simulation with 2 × 2 × 1 subvoxels. (c) Simulation with 4 × 4 × 1 subvoxels. (d) Difference between (a) and (b). (e) Difference between (b) and (c). (f) Experimental image (image matrix: 256 × 256 × 16). The simulated and experimental images were acquired with a gradient echo sequence (TR = 200 ms, echo time = 48 ms, flip angle = 90°, field of view = [76.8 mm]^3, pixel bandwidth = 48.6 Hz). The execution times for (a), (b), and (c) were 2.28, 4.81, and 15.20 s, respectively.

Although inhomogeneous image intensity was observed in the image acquired with the selective excitation pulse at −40 kHz and −25 mm from the center of the magnet, homogeneous image intensity and the clear shape of the phantom were observed for other slices.

Fig. 13(b) and (c) shows multislice images of the numerical phantom calculated using BlochSolver. The numbers of subvoxels were 7 and 2 for the readout and phase encoding directions, respectively. The B_1 distribution shown in Fig. 6 was used for the calculation. In Fig. 13(b) and (c), the selective excitation pulses (sinc(x) with 1-ms duration) were approximated using 100 and 200 short square pulses, of width 10 and 5 μs, respectively. The calculation times for all multislice images were about 45 and 75 min, respectively. This result clearly shows that the calculation time for the RF excitations was about 30 and 60 min, respectively, because the time for the NMR signal calculation was identical for both calculations.

Fig. 14 shows selected multislice images acquired by the gradient-echo multislice simulation with the selective excitation pulse (sinc(x) with 1-ms duration) approximated with 100, 110, 115, and 120 short pulses. The numbers of subvoxels were 3 and 1 for the readout and phase encoding directions, respectively. The first, second, third, and fourth rows represent the slice positions whose central frequencies are −40, −32, +32, and +40 kHz, respectively. The ghost at +32 kHz disappears between 100 and 110 pulses and that at +40 kHz disappears between 110 and 115 pules. This result clearly shows that the approximation using 100 short square pulses was insufficient for precise RF excitation of the plane, but that using 115 pulses was sufficient for simulation of multislice imaging under this experimental condition. This result can be understood in the following way. During the selective excitation in the multislice sequence, isochromats in the slicing plane and the selective excitation RF pulse rotate about the z-axis in the same sense and in the same frequency such as +32 kHz. However, if the RF pulse was approximated or replaced by a short RF pulse with 10 μs duration, the phase of the RF pulse get behind that of the isochromats by 120 degree at most because the short RF pulse is stationary in the rotating frame and the rotation cycle of the isochromats is about 30 μs. This mechanism partly produces negative sense rotation (−32 kHz) of the isochromats through the nutation of the isochromats. This mechanism also works for the isochromats rotating in the negative frequency (−32 kHz) and the isochromats are partly excited by the short RF pulse. In this way, the ghost artifacts from the negatively symmetric position can be observed. On the other hand, if the selective excitation RF pulse was approximated or replaced by a short RF pulse with 9 μs duration, the phase of the RF pulse get behind that of the isochromats by 110 degree at most, the negative sense rotation is not produced and the ghosting artifacts are not observed.

3.4. Processing speed

Table 1 shows a comparison of processing speeds for the Xeon CPU and the GPU 1080 performed using the numerical phantom (matrix size: 256 × 256 × 32) corresponding to Fig. 5(c). Because voxels with zero-valued proton density were excluded from the

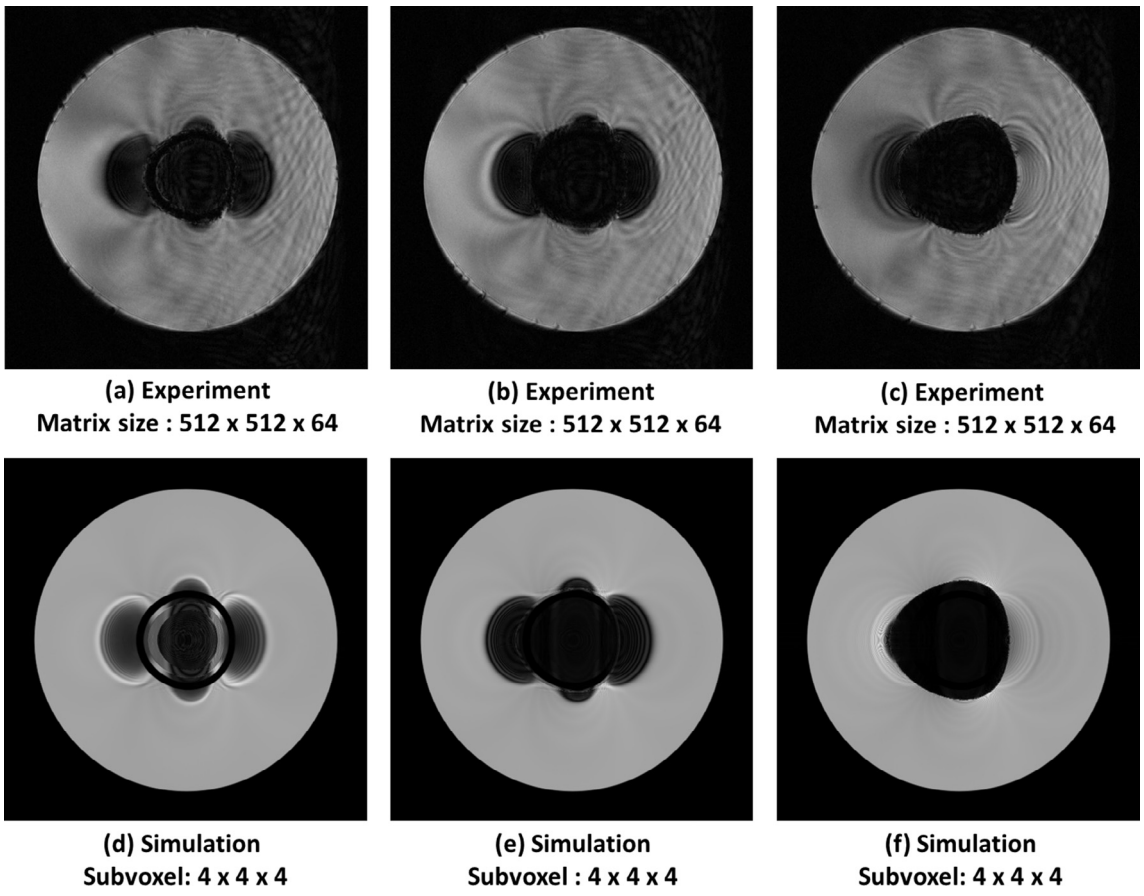

Fig. 8. Cross-sectional images of the spherical phantom. (a)–(c) experiments acquired with a gradient-cho sequence (TR = 200 ms, TE = 48 ms, image matrix = 512 × 512 × 64). (d)–(f) simulation with the matrix size of the numerical phantom = 512 × 512 × 64. The number of subvoxels was 4 × 4 × 4 in the x, y, and z directions. The simulated and experimental images were acquired with a gradient echo sequence (TR = 200 ms, echo time = 48 ms, flip angle = 90°, field of view = $[76.8\ \text{mm}]^3$, pixel bandwidth = 48.6 Hz). The execution time was 3.83 h.

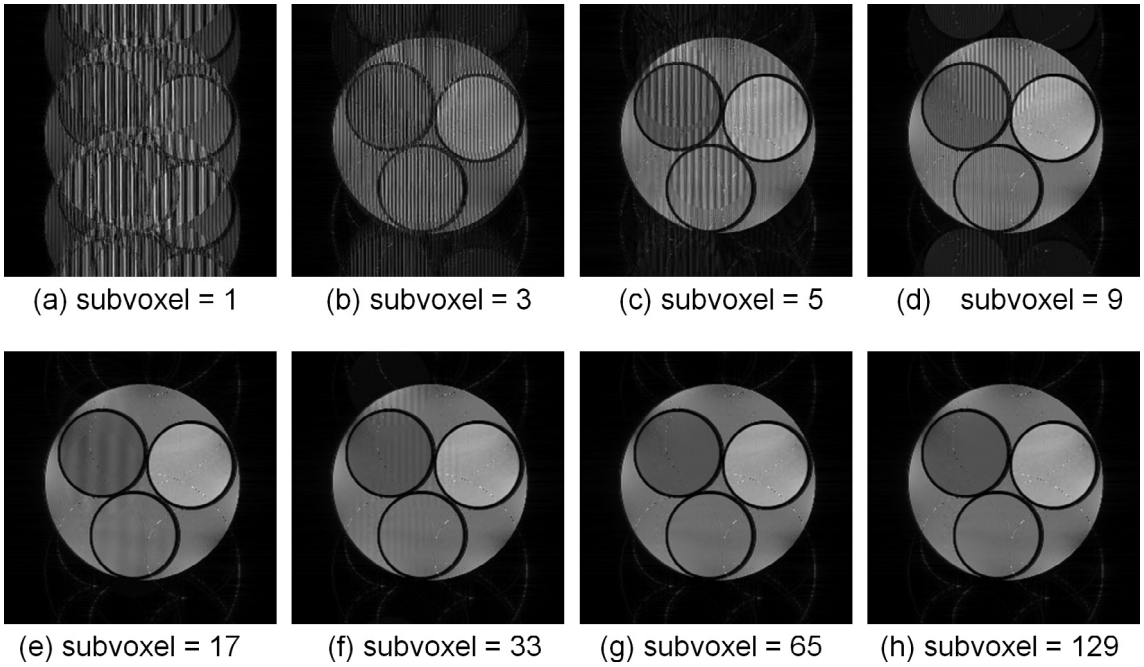

Fig. 9. Cross-sectional images selected from 3D image datasets of a numerical phantom simulated for the RF spoiled gradient echo sequence (TR/TE = 20 ms/6 ms, FA = 30°, image matrix = 256 × 256 × 32) using various numbers of subvoxels. The phase φ of the RF pulse was updated according to $\varphi = n(n+1)\Phi/2$ (n: order of the RF pulse, Φ: phase increment angle). When the number of subvoxels is larger than 65, the stripe artifacts cannot be seen. T_1 and T_2 for the $CuSO_4$ water solution in the three small bottles were 114, 244, and 351 ms, and those for the baby oil surrounding the bottles were 186 and 80 ms. The resonance frequency difference between the water and the oil was 220 Hz in the simulation. The execution times for (a)–(h) were 5.1, 9.6, 14.4, 23.8, 42.6, 80.4, 156.1, and 310.5 s, respectively.

calculation, the actual number of isochromats used for the calculation was 52,202,496 (= 256 × 256 × 32 × 65 × 0.382955...). The number 0.382955... is the ratio of the number of non-zero proton density voxels to the total number of the voxels in the image matrix.

The computation times of 2161.409 and 157.469 s were obtained for the CPU and the GPU, respectively. By dividing the number of operations by these computation times, we obtained computation speeds of 506.5 and 6952 GFLOPS for the CPU and the GPU, respectively. These computation speeds were 19.1% and 39.1% of the peak performances of 2650 and 17,746 GFLOPS for the CPU and the GPU, respectively.

In summary, the calculation speed for the GPU-based MRI simulator was about 14 times faster than that for the CPU-based simulator when using the same number of processor units (number of CPUs = number of GPUs).

4. Discussion

4.1. Processing speed

We discuss several aspects of the processing speed of BlochSolver achieved in this study.

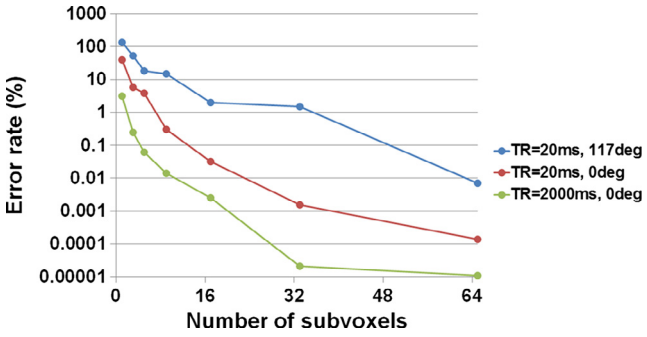

Fig. 10. Error rate plotted against the number of subvoxels for the RF spoiled gradient echo sequence with (a) TR = 20 ms and Φ = 117° (conventional SPGR) shown in blue, (b) TR = 20 ms and Φ = 0° (conventional gradient-echo sequence) shown in brown, and (c) TR = 2000 ms and Φ = 0° shown in light green. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

First, the calculation time for a typical 3D gradient-echo image of 256 × 256 × 16 image matrix with 16 subvoxels was 15.20 s. This processing speed corresponds to a computational TR of 3.7 ms, which is much less than the typical TR in MRI experiments. Therefore, this result suggests that MRI studies using BlochSolver will be much more efficient than experimental MRI studies, even when using large image matrices.

Next, the calculation times for multislice imaging for eleven 256 × 256 pixel gradient-echo images with 7 × 2 subvoxels were 45 and 75 min using 100 and 200 short square pulse approximations, respectively. Because multislice imaging is one of multipulse experiments and the slice profile is essential in multislice imaging, a large number of subvoxels (469,762,048) is required to obtain artifact-free images. To provide sufficient memory for the numerical phantom, system parameters, and components of the isochromats, about 24 GB memory ($\sim 4.7 \times 10^8 \times 4 \times 12$) was required, which exceeded the memory size of the GPU boards (16 GB). However, by using the memory and calculation time saving technique developed in this study, multislice simulation for this matrix size became possible.

Finally, the calculation speed of the GPUs was about 14 times that of the CPUs used in this study. Because GPUs and CPUs use identical semiconductor technology, the difference in processing speed between GPUs and the CPUs is unlikely to change in the near future.

4.2. Processing speed: Comparison with the earlier study

As described in the Introduction section, Xanthis et al. reported the first GPU-based MRI simulator in 2014 [13]. They reported 31–228 times processing speed advantage over the CPU-based MRI simulation. However, because developments in the CPU technology have made it possible to perform parallel processing such as AVX (advanced vector extension) 2 or 3 for the CPUs, the processing speeds of the CPUs approaching those of the GPUs. As a result, our present processing speed comparison between them became about 14 times with AVX 2 and 83 times without AVX 2.

To compare the processing speed between the previous and the present MRI simulators quantitatively, we performed a typical 2D single slice gradient-echo MRI simulation which was supposed to be identical to that performed in the previous study. The simulation condition is as follows; the number of isochromats: 1,350,000 (150 × 150 × 60), TR = 50 ms, selective excitation pulse:

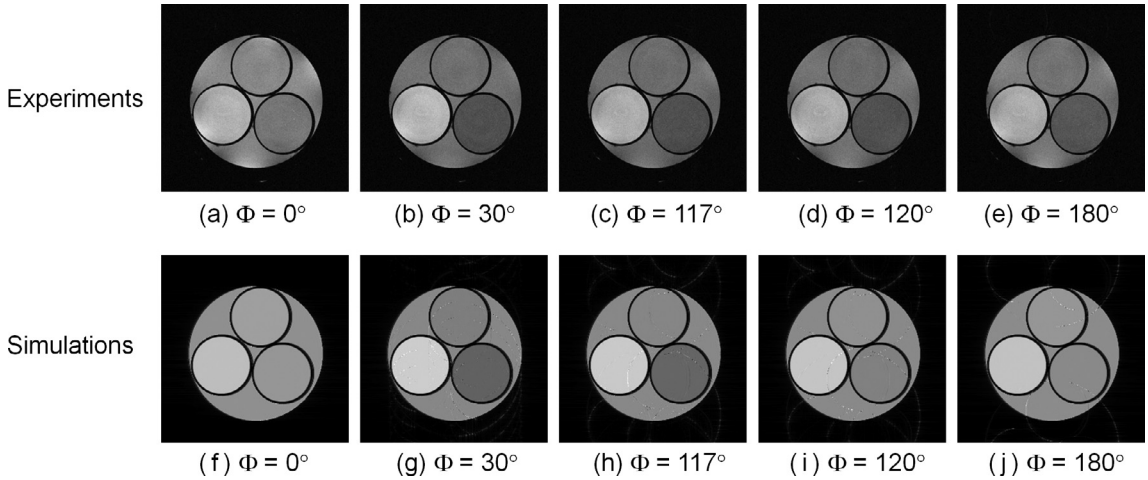

Fig. 11. Cross-sectional images selected from 3D image datasets acquired with the SPGR sequences (TR/TE = 20 ms/6 ms, FA = 30°, image matrix = 256 × 256 × 32, FOV = [64 mm]^3). Φ was the phase increment angle described in the text. (a)–(e) acquired by experiment at 1.5 T. (f)–(j) calculated with BlochSolver (without diffusion effects). The number of subvoxels was 65. The execution time for one SPGR image was about 156 s.

three lobe sinc pulse (300 short pulse approximation with 3 ms duration time), data-sampling interval: 10 μs, the number of the sampling points: 256, k-space matrix: 256 × 256. The calculation times for the previous and the present studies were 282 and 4.06 s, respectively. Because they used a single GPU board of C2070 (NVIDIA) with peak performance of 1.03 TFLOPS (single precision) and we used dual GPU boards of GTX 1080 with peak performance of 17.746 TFLOPS, the processing speed advantage due to the hardware was about 17.2. Therefore, we think that the processing speed advantage of about 70 times was achieved by about four times improvement of software implementation, which was probably due to the difference between the use of register file (present study) and shared memory (previous study) for NMR signal calculation.

4.3. Floating-point calculations: Single vs. Double

In many scientific and technical calculations, calculations are performed in terms of 64-bit double-precision floating-point operations. Calculation using current high-end CPUs is usually performed in double precision. In addition, for such CPUs, the difference in speed between double-precision and 32-bit single-precision floating-point operation is small. However, for GPUs, the time for a double-precision operation is significantly more than that for a single precision operation (by a factor of 2–32). In particular, for the inexpensive GPU board (GTX 1080) used in this research, the factor is actually 32 (single precision: 8.9 TFLOPS, double precision: 0.277 TFLOPS [emulation]).

On the other hand, because white noise caused by thermal noise is always present in experimental MR images, the intensity of observable MR image artifacts can be at least 0.1% different from equivalent artifact-free images. Therefore, there is no problem if the calculation accuracy for voxel values is within 0.1%. However, in the MRI signal calculation for 256^3 voxels, for example, the signal intensity is summation of more than about 2^{24} (~16,000,000) transverse magnetization components. This means that the cancellation of significant digits would occur if we were to use 32-bit floating-point representation. We therefore performed the calculation of evolution and the local summation of isochromats (actually 1024 isochromats) using single-precision floating-point operations and performed the total summation of these local signals in double precision. In this way, we could achieve both rapid calculations and adequate precision at the same time.

4.4. Subvoxel

Because the goal of an MRI simulator is to obtain an accurate MRI signal for a continuous object, a crucial issue is the discretization of the nuclear magnetization of the object. If the processing speed is sufficient, the simple method of finely dividing a voxel into many subvoxels and calculating with them will be effective. Therefore, we will discuss about the practical methods to determine the appropriate number of the subvoxels below. There are two typical cases where subvoxels are indispensable.

First, for the gradient echo sequences, static magnetic field inhomogeneity causes intravoxel phase dispersion, which causes a signal decrease or loss. In this case, the practical method to determine to the appropriate number of the subvoxels is to change the number of the subvoxels along x, y, and/or z directions and repeat the MRI simulations as shown in Fig. 7. If the difference between the images with different numbers of subvoxels becomes small or negligible, the appropriate number of the subvoxels can be determined.

The second case involves fast gradient-echo sequences and multislice imaging where coherence of the nuclear magnetization is essential [29]. In this case, if the number of subvoxels is sufficient to express the variation in nuclear magnetization within the voxel, i.e., the wave number in the signal-readout gradient direction, artifacts can be removed and the voxel intensity can be correctly reproduced, as shown in Fig. 9. The practical method to determine the appropriate number of the subvoxels is to use the data-acquisition window shown in Fig. 4. In this case, if the number of subvoxels is insufficient, pseudo-echoes are observed in the data-acquisition window. In such case, we can find the appropriate number of the subvoxels by repeating the simulation by increasing the number of the subvoxels as 1, 3, 5, 7, . . . and observing the NMR signal in the data-acquisition window. In some cases, pseudo-echoes are observed far from the center of the k-space. In such case, the number of subvoxels along the phase encoding direction should be increased and the MRI simulation should be repeated to obtain an artifact-free image. Anyway, it is difficult to predict gen-

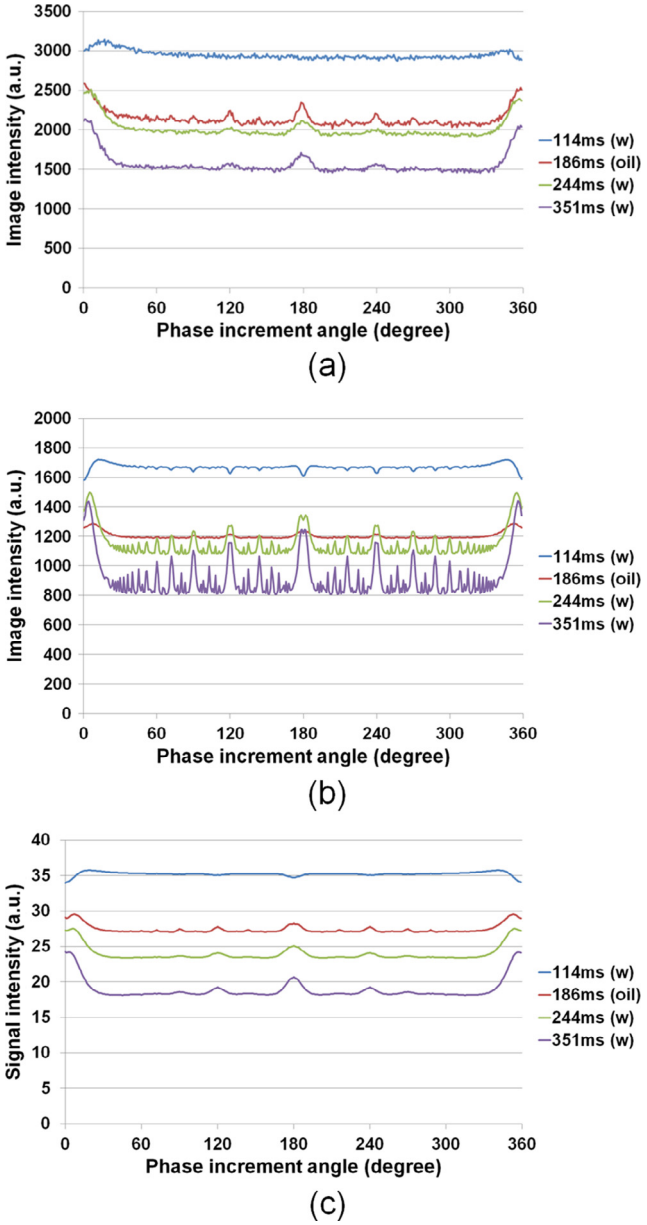

Fig. 12. Phase increment angle dependence of the image intensity in the relaxation phantom. (a) Experiments. (b) Simulations without molecular diffusion effects. (c) One voxel simulation with a molecular diffusion effect. The numbers written in the legends of the graphs are T_1 values.

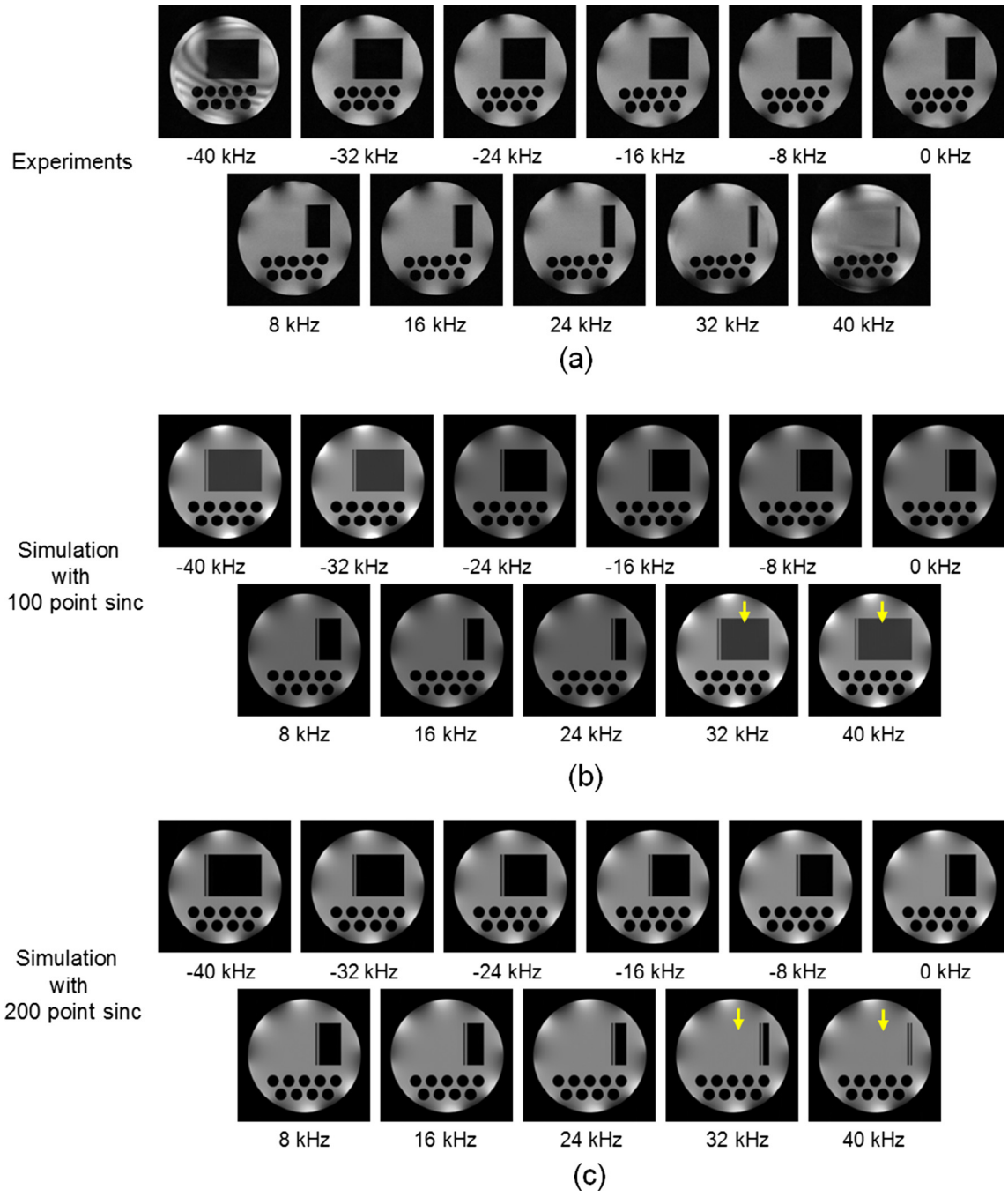

Fig. 13. Cross-sectional images acquired by (a) experiment, (b) simulation with a selective excitation pulse (a 100 short-square-pulse approximation to a sinc pulse), and (c) simulation with a selective excitation pulse (a 200 short-square-pulse approximation to a sinc pulse). The pulse sequence used to image the multislice phantom shown in Fig. 6(d) was a gradient-echo multislice sequence (TR = 1200 ms, TE = 6 ms, number of slices = 11, time between RF pulses = 100 ms, slice thickness = 5 mm, FA = 90°, FOV = [64 mm]^2, image matrix = 256 × 256). The ghost artifacts observed in (b) were not visible in (c), as shown by the arrows. The execution times for (b) and (c) were 45 and 75 min.

eration of pseudo-echoes, however, the extended phase graph (EPG) approach [30] may be useful for the artifact prediction.

As described above, being able to apply appropriate settings for subvoxels is an important and powerful aspect of MRI simulators, and with the speeding up of calculations in BlochSolver, it becomes simple and practical.

4.5. Sequence compatibility

In the imaging experiments and simulations described in this study, the experiments were conducted first, followed by the creation of numerical phantoms to simulate the real phantoms. Calculations were performed with BlochSolver using the pulse sequences used in the experiments. Improvements were added to BlochSolver so that the experimental results and the simulation results agreed with each other. In this way, by designing the same sequences as those used in the experiments, comparison between the simulations and the experiments became straightforward.

In this study, we used the pulse sequence format that we have been using for many years in our laboratory. Therefore, if we apply BlochSolver to clinical machines or other research-oriented MRI machines, it will be necessary to prepare translations from those

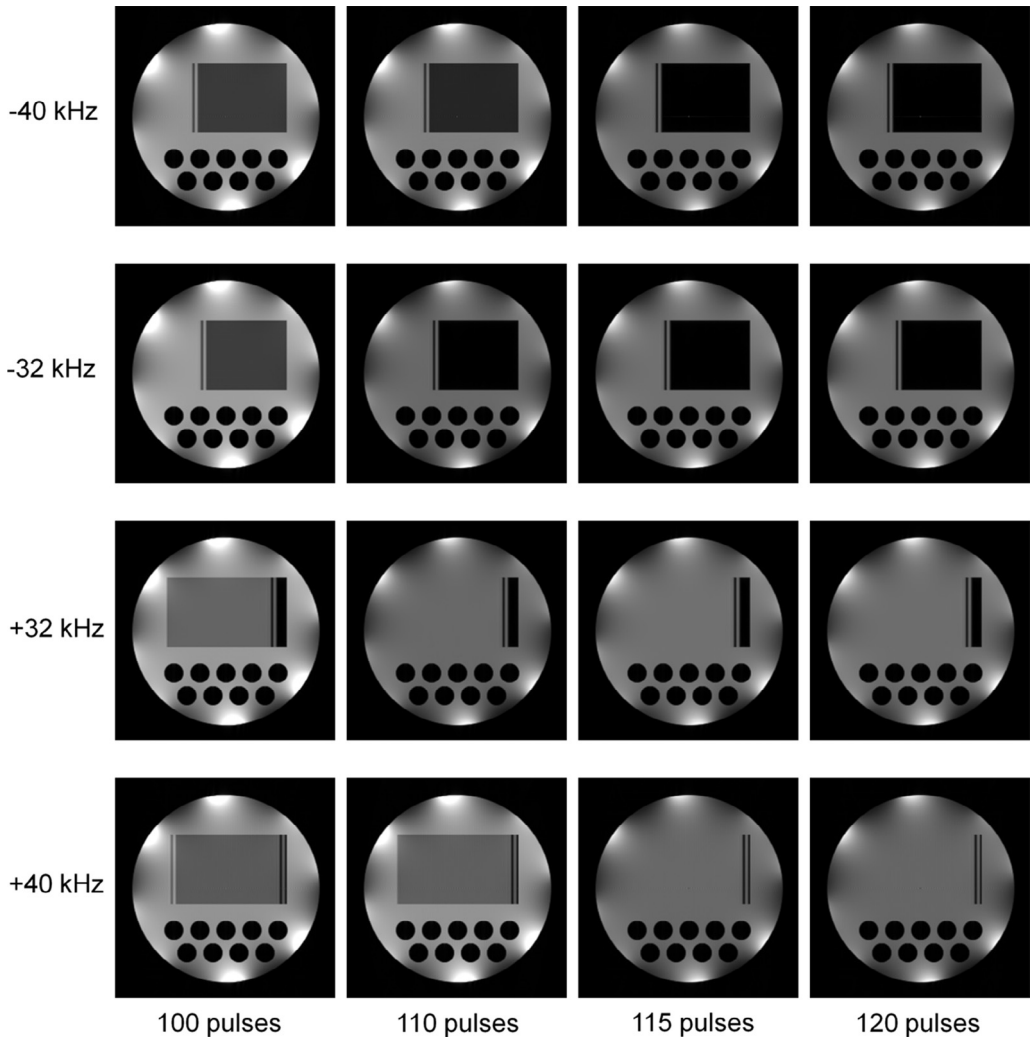

Fig. 14. Multislice images acquired by the gradient-echo multislice simulation with a selective excitation pulse (sinc(x) with 1-ms duration) approximated with 100, 110, 115, and 120 short pulses. The numbers of subvoxels were 3 and 1 for the readout and phase encoding directions, respectively. The first, second, third, and fourth rows represent the slice positions whose central frequencies are −40, −32, +32, and +40 kHz, respectively. The ghost at +32 kHz disappears between 100 and 110 pulses and that at +40 kHz disappears between 110 and 115 pules.

Table 1
Comparison of CPU and GPU processing speeds for an identical MRI simulation. The number of operations includes only multiplications and summations for the precession and relaxation of the isochromats and does not include sine, cosine, and exponential function calls and global summations for NMR signal calculation. This value is calculated by multiplications of the number of isochromats (52,202,496), the number of k-space sampling points (256 × 256 × 32), and the number of operations (10) used for integration of the Bloch equation. The ratio of the number of sine, cosine, and exponential function calls to that of multiplications and summations is 3:40. The processing time for the sine, cosine, and exponential function calls depends on the architecture of the processor and the generation of the GPU. The ratio of the processing time for the special function calls to that for the whole operations was about 20% for GTX-1080. The peak performance means the speed of the arithmetic logic units published by the manufacturers. The computational efficiency is the ratio of the measured computation speed to the peak performance.

Device	Xeon E5–2699v3 × 2	GeForce GTX–1080 × 2	Ratio (GPU/CPU)
Number of isochromats	52,202,496		1
Number of operations ($\times 10^9$)	1,094,766		1
Computation time (s)	2161.409	157.469	0.07285
	12,999 (w/o AVX 2)		0.01211
Computation speed (GFLOPS)	506.5	6952	13.726
	84.2185 (w/o AVX 2)		82.55
Peak performance (GFLOPS)	2650	17,746	6.70
Computational efficiency	0.191	0.392	2.05
Optimization technique	OpenMP, AVX 2 Register optimization Customized sin, cos, exp. Loop size optimization	MRI signal calculation within the register file	

sequence formats to our sequence format, or to develop a new sequence-description language that is compatible with many pulse sequence formats.

4.6. Diffusion effect

As shown in Fig. 12, by introducing a molecular diffusion effect to the SPGR sequence, it was possible to reproduce the changes in the image intensity acquired by the experiment when the phase increment angle Φ was changed. However, in order to calculate the molecular diffusion effect, it was necessary to calculate second partial derivatives using adjacent subvoxels. This operation is not compatible with the high-speed calculation of image subvoxels using a GPU. Therefore, we consider it more practical to multiply the diffusional signal attenuation that is individually calculated in each voxel by the components of the isochromats in the voxel used for MRI signal calculation.

4.7. Chemical shift and T_2^*

The present GUI of BlochSolver shown in Fig. 2 cannot be used for objects with chemical shift distributions and T_2^* effects. For objects with various chemical shifts (e.g., water and fat), if required number of numerical phantoms, system parameters, and nuclear magnetization components are individually defined, calculations using BlochSolver can be performed for individual objects with different chemical shifts (resonance frequency). Although various approaches to the representation of T_2^* effects on MR images have been proposed [7,10], there has been no straightforward solution. We consider that the T_2^* effects could be modeled by distribution of objects with different resonant frequencies within the voxels and calculated with BlochSolver.

4.8. Future directions

In the development of BlochSolver, the primary purpose was to improve calculation speed for the NMR signal with constant signal readout gradients. In other words, it is currently unsuited to non-Cartesian sequences such as the spiral sequence where the gradient field strength changes during the signal readout periods. Currently, BlochSolver cannot be used for parallel transmission or for parallel imaging. Extension to these two directions is now underway.

In this study, BlochSolver used two GPU boards optimized for single-precision floating point operations. Because the GPU we used is inexpensive (<1000 USD), a large-scale parallel system could be constructed easily. In addition, with calculation speeds approximately doubling every 2 years, we consider that more realistic simulations (e.g. multislice fast spin-echo) could be performed more quickly in future if we were to adopt large-scale parallel computing systems in the future. In addition, applications of BlochSolver to motion and flow of the imaging objects will be an interesting extension of this study.

5. Conclusion

We have developed a GPU-optimized MRI simulator (BlochSolver) for experimentally compatible pulse sequences and performed three types of experiments and corresponding simulations. Our results demonstrate that the calculation speed of the GPU-based system was about 14 times faster than that of a CPU-based system when the same number of processing units is used. MR images acquired by experiment could be reproduced by using an appropriate number of subvoxels. We demonstrated that BlochSolver could be used for 3D MRI simulations that involved image matrices of practical sizes. In conclusion, we expect the MRI simulators to become powerful and indispensable tools for MRI research and development in near future.

Acknowledgments

We acknowledge Dr. Tomoyuki Haishi for supporting this work. This work was also supported by the Japan Science and Technology Agency.

References

[1] J. Bittoun, J. Taquin, M. Sauzade, A computer algorithm for the simulation of any nuclear magnetic resonance (NMR) imaging method, Magn. Reson. Imaging 2 (1984) 113–120.
[2] R.M. Summers, L. Axel, S. Israel, A computer simulation of nuclear magnetic resonance imaging, Magn. Reson. Med. 3 (1986) 363–376.
[3] P. Shkarin, R.G. Spencer, Direct simulation of spin echoes by summation of isochromats, Concepts Magn. Reson. A8 (1996) 253–268.
[4] P. Shkarin, R.G. Spencer, Time domain simulation for Fourier imaging by summation of isochromats, Int. J. Imaging Syst. Technol. 8 (1997) 419–426.
[5] R.K. Kwan, A.C. Evans, G.B. Pike, MRI simulation-based evaluation of image-processing and classification methods, IEEE Trans. Med. Imaging 18 (1999) 1085–1097.
[6] D.A. Yoder, Y. Zhao, C.B. Paschal, J.M. Fitzpatrick, MRI simulator with object-specific field map calculations, Magn. Reson. Imaging 22 (2004) 315–328.
[7] H. Benoit-Cattina, G. Collewet, B. Belaroussi, H. Saint-Jalmes, C. Odet, The SIMRI project: a versatile and interactive MRI simulator, J. Magn. Reson. 173 (2005) 97–115.
[8] T.H. Jochimsen, A. Schäfer, R. Bammer, M.E. Moseley, Efficient simulation of magnetic resonance imaging with Bloch-Torrey equations using intra-voxel magnetization gradients, J. Magn. Reson. 180 (2006) 29–38.
[9] J.C. Sharp, D. Yin, R.H. Bernhardt, Q. Deng, A.E. Procca, R.L. Tyson, K. Lo, B. Tomanek, The integration of real and virtual magnetic resonance imaging experiments in a single instrument, Rev. Sci. Instrum 80 (2009) 093709.
[10] P. Latta, M.L.H. Gruwel, V. Jellúš, B. Tomanek, Bloch simulations with intra-voxel spin dephasing, J. Magn. Reson. 203 (2010) 44–51.
[11] T. Stöcker, K. Vahedipour, D. Pflugfelder, N.J. Shah, High-performance computing MRI simulations, Magn. Reson. Med. 64 (2010) 186–193.
[12] K.G. Baum, G. Menezes, M. Helguera, Simulation of high-resolution magnetic resonance images on the IBM Blue Gene/L supercomputer using SIMRI, Int. J. Biomed. Imaging. 2011 (2011) Article ID 305968.
[13] C.G. Xanthis, I.E. Venetis, A.V. Chalkias, A.H. Aletras, MRISIMUL: a GPU-based parallel approach to MRI simulations, IEEE Trans. Med. Imaging 33 (2014) 607–616.
[14] C.G. Xanthis, I.E. Venetis, A.H. Aletras, High performance MRI simulations of motion on multi-GPU systems, J. Cardiovasc. Magn. Reson. 16 (2014) 48–62.
[15] Z. Cao, S. Oh, C.T. Sica, J.M. McGarrity, T. Horan, W. Luo, C.M. Collins, Bloch-based MRI system simulator considering realistic electromagnetic fields for calculation of signal, noise, and specific absorption rate, Magn. Reson. Med. 72 (2014) 237–247.
[16] J. Chen, M. Grossman, T. McKercher, Professional CUDA C Programming, John Wiley & Sons, Inc, Indianapolis, IN, USA, 2014.
[17] F. Bloch, Nuclear induction, Phys. Rev. 70 (1946) 460–473.
[18] H.C. Torrey, Bloch equations with diffusion terms, Phys. Rev. 104 (1956) 563–565.
[19] K. Kose, T. Haishi, Development of a flexible pulse programmer for MRI using a commercial digital signal processor board, in: P. Blümler, B. Blümich, R. Botto, E. Fukushima (Eds.), Spatially Resolved Magnetic Resonance, Wiley–VCH, 1998, pp. 703–709.
[20] T. Haishi, T. Uematsu, Y. Matsuda, K. Kose, Development of a 1.0 T MR microscope using a Nd–Fe–B permanent magnet, Magn. Reson. Imaging 19 (2001) 875–880.
[21] Y. Matsuda, S. Utsuzawa, T. Kurimoto, T. Haishi, Y. Yamazaki, K. Kose, I. Anno, M. Marutani, Super-parallel MR microscope, Magn. Reson. Med. 50 (2003) 183–189.
[22] K. Kose, Y. Matsuda, T. Kurimoto, S. Hashimoto, Y. Yamazaki, T. Haishi, S. Utsuzawa, H. Yoshioka, S. Okada, M. Aoki, T. Tsuzaki, Development of a compact MRI system for trabecular bone volume fraction measurements, Magn. Reson. Med. 52 (2004) 440–444.
[23] N. Iita, S. Handa, S. Tomiha, K. Kose, Development of a compact MRI for measurement of trabecular bone microstructure of the finger, Magn. Res. Med. 57 (2007) 272–277.
[24] S. Handa, S. Tomiha, T. Haishi, K. Kose, Development of a compact MRI system for trabecular bone microstructure measurements of the distal radius, Magn. Reson. Med. 58 (2007) 225–229.
[25] S. Hashimoto, K. Kose, T. Haishi, Development of a pulse programmer for magnetic resonance imaging using a personal computer and a high-speed digital input–output board, Rev. Sci. Instrum 83 (2012) 053702.
[26] K.M. Lüdeke, P. Röschmann, R. Tischler, Susceptibility artifact in NMR imaging, Magn. Reson. Imaging 3 (1985) 329–343.

[27] C.E. Hayes, W.A. Edelstein, J.F. Schenck, O.M. Mueller, M. Eash, An efficient, highly homogeneous radio frequency coil for whole-body NMR imaging at 1.5T, J. Magn. Reson. 63 (1985) 622–628.
[28] Y. Zur, M.L. Wood, L.J. Neuringer, Spoiling of transverse magnetization in steady-state sequences, Magn. Reson. Med. 21 (1991) 251–263.
[29] M. Weigel, Extended phase graphs: dephasing, RF pulses, and echoes - pure and simple, J. Magn. Reson. Imaging. 41 (2015) 266–295.
[30] S.J. Malik, A. Sbrizzi, H. Hoogduin, J.V. Hajnal, Equivalence of EPG and isochromat-based simulation of MR signals, Proc. Intl. Soc. Mag. Reson. Med. 24 (2016) 3196.
[31] V.L. Yarnykh, Optimal radiofrequency and gradient spoiling for improved accuracy of T_1 and B_1 measurements using fast steady state techniques, Magn. Reson. Med. 63 (2010) 1610–1626.