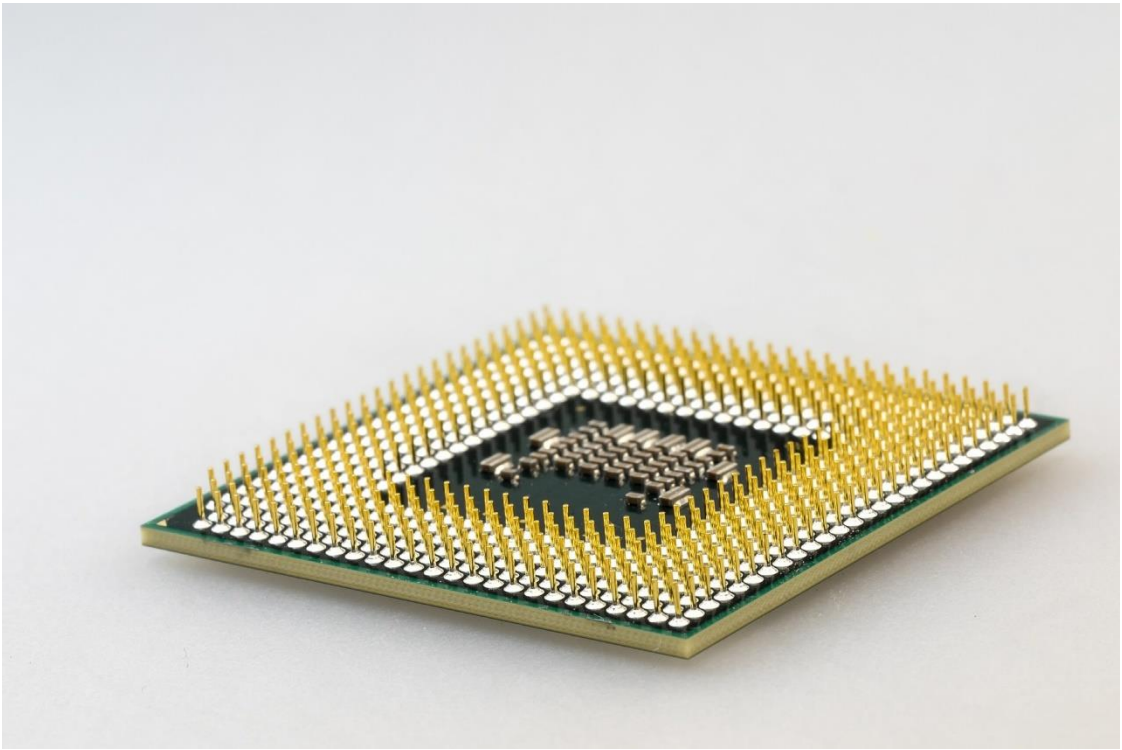
BlochSolver CPU version

User's Manual



Version 1.02, July 26, 2020

Contact: info_at_mrisimulations.com

Table of contents

1. Introduction

This manual is written for **BlochSolver-CPU**, an MRI simulation program that operates pulse sequences written in "Python Pulse Sequence Development Kit" (**PSDK**).

BlochSolver-CPU is an MRI simulation program that does not require a GPU (Graphics Processor Unit) and runs on the CPU (Central Processing Unit). **BlochSolver-CPU** works exactly the same as **BlochSolver-GPU** which has already been announced shown below.

BlochSolver-CPU uses all the cores of the CPU (Fig. 1), so the use of other applications is quite limited during operation. **BlochSolver-CPU** is several tens of times slower than **BlochSolver-GPU**, but **BlochSolver-CPU** can be used for various MR imaging simulations if the numbers of voxels and the number of subvoxels are properly selected.

BlochSolver-CPU requires a 64-bit CPU (manufactured by Intel or AMD which uses AVX2 instruction set) that operates under Windows 10. Please confirm the specification of your CPU before installation.

In the following sections, we will explain the installation method, test operation, preparations for simulation and image reconstruction, examples of simulation, and how to speed up the simulation.

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,*]

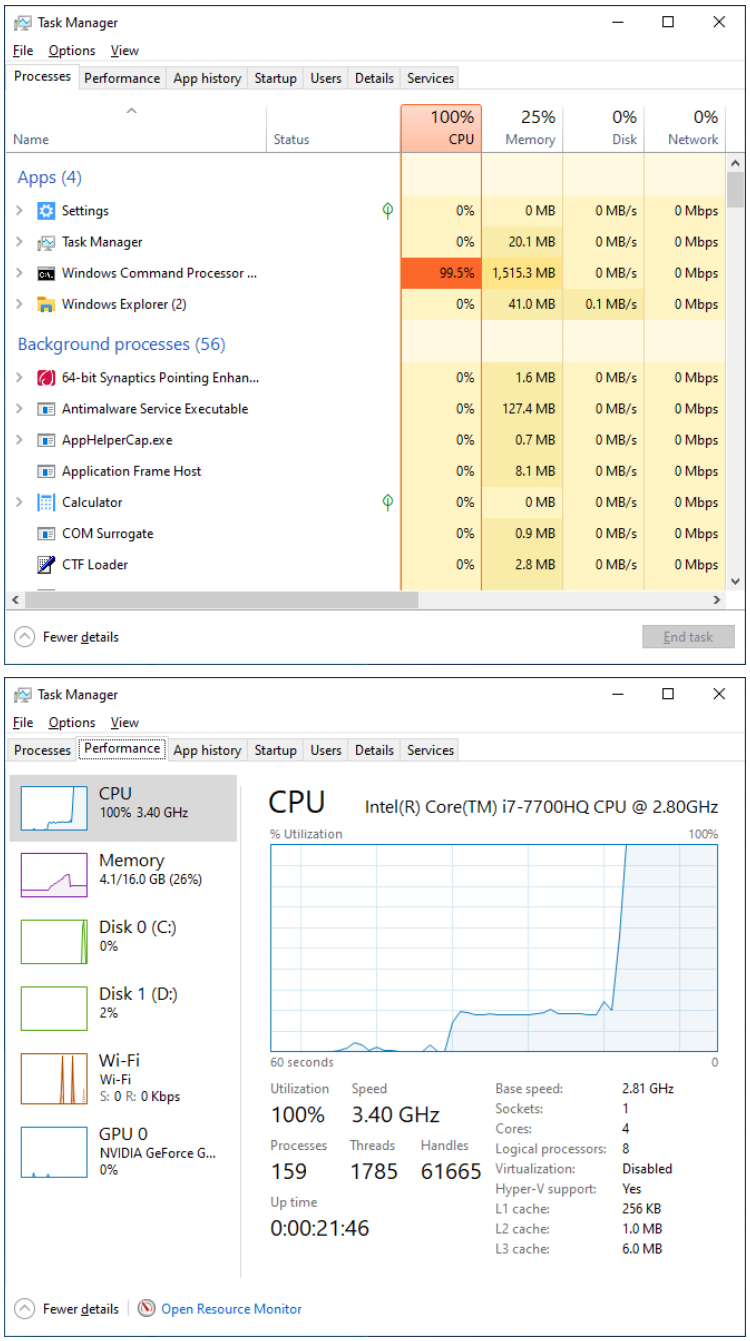
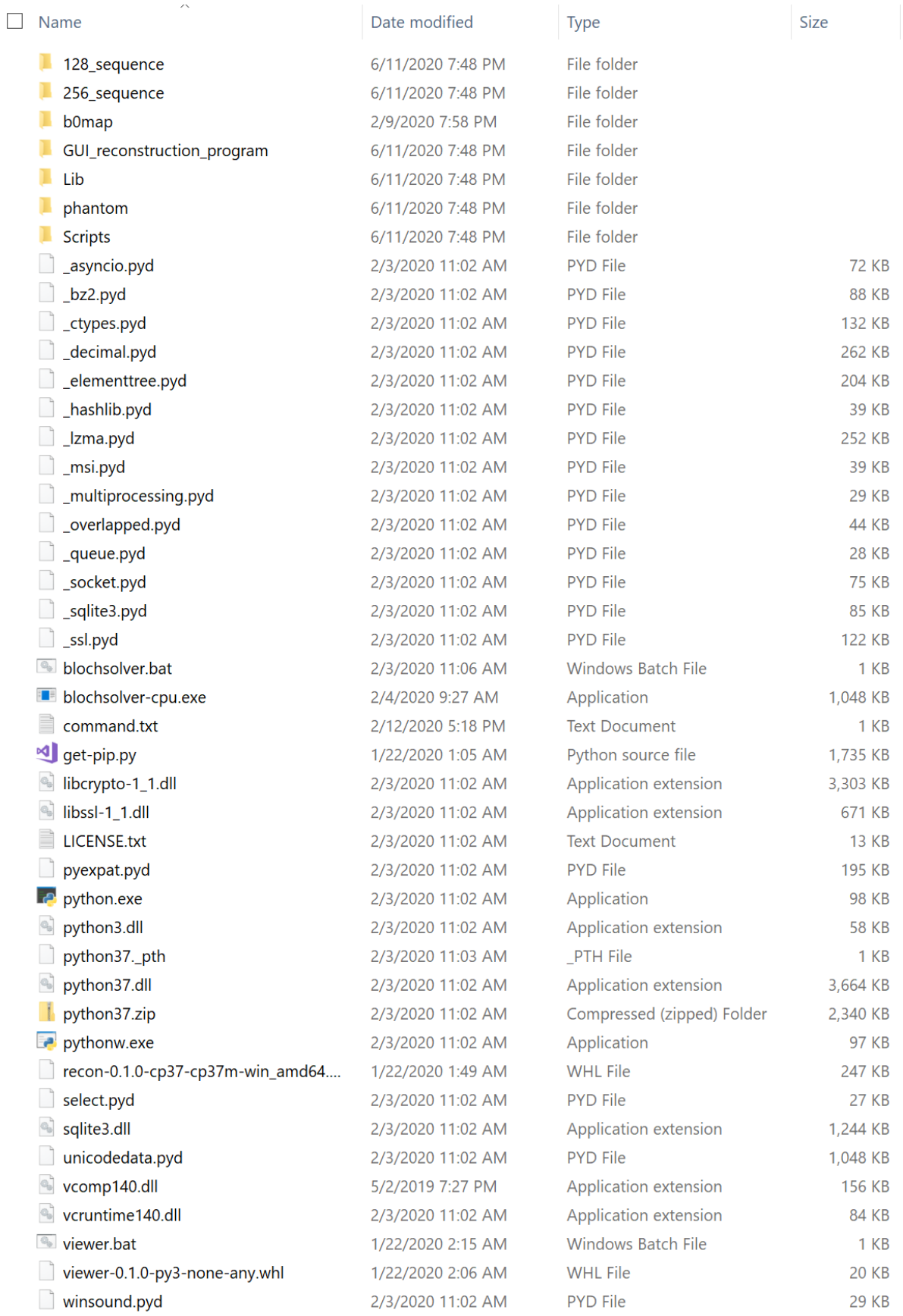


Fig.1 Task manager when BlochSolver-CPU is running. Upper: process, Lower: performance

2. Installation and operation test

Download a zip file (blochsolver-cpu.zip) and a user's manual (blochsolver-cpu_users_manual.pdf) from the download site. Unzip the zip file to a folder of your choice (e.g. desktop folder). It will take about a few minutes to unzip. The contents of the blochsolver-cpu folder are as follows.



Name	Date modified	Type	Size
128_sequence	6/11/2020 7:48 PM	File folder	
256_sequence	6/11/2020 7:48 PM	File folder	
b0map	2/9/2020 7:58 PM	File folder	
GUI_reconstruction_program	6/11/2020 7:48 PM	File folder	
Lib	6/11/2020 7:48 PM	File folder	
phantom	6/11/2020 7:48 PM	File folder	
Scripts	6/11/2020 7:48 PM	File folder	
_asyncio.pyd	2/3/2020 11:02 AM	PYD File	72 KB
_bz2.pyd	2/3/2020 11:02 AM	PYD File	88 KB
_ctypes.pyd	2/3/2020 11:02 AM	PYD File	132 KB
_decimal.pyd	2/3/2020 11:02 AM	PYD File	262 KB
_elementtree.pyd	2/3/2020 11:02 AM	PYD File	204 KB
_hashlib.pyd	2/3/2020 11:02 AM	PYD File	39 KB
_lzma.pyd	2/3/2020 11:02 AM	PYD File	252 KB
_msi.pyd	2/3/2020 11:02 AM	PYD File	39 KB
_multiprocessing.pyd	2/3/2020 11:02 AM	PYD File	29 KB
_overlapped.pyd	2/3/2020 11:02 AM	PYD File	44 KB
_queue.pyd	2/3/2020 11:02 AM	PYD File	28 KB
_socket.pyd	2/3/2020 11:02 AM	PYD File	75 KB
_sqlite3.pyd	2/3/2020 11:02 AM	PYD File	85 KB
_ssl.pyd	2/3/2020 11:02 AM	PYD File	122 KB
blochsolver.bat	2/3/2020 11:06 AM	Windows Batch File	1 KB
blochsolver-cpu.exe	2/4/2020 9:27 AM	Application	1,048 KB
command.txt	2/12/2020 5:18 PM	Text Document	1 KB
get-pip.py	1/22/2020 1:05 AM	Python source file	1,735 KB
libcrypto-1_1.dll	2/3/2020 11:02 AM	Application extension	3,303 KB
libssl-1_1.dll	2/3/2020 11:02 AM	Application extension	671 KB
LICENSE.txt	2/3/2020 11:02 AM	Text Document	13 KB
pyexpat.pyd	2/3/2020 11:02 AM	PYD File	195 KB
python.exe	2/3/2020 11:02 AM	Application	98 KB
python3.dll	2/3/2020 11:02 AM	Application extension	58 KB
python37._pth	2/3/2020 11:03 AM	_PTH File	1 KB
python37.dll	2/3/2020 11:02 AM	Application extension	3,664 KB
python37.zip	2/3/2020 11:02 AM	Compressed (zipped) Folder	2,340 KB
pythonw.exe	2/3/2020 11:02 AM	Application	97 KB
recon-0.1.0-cp37-cp37m-win_amd64....	1/22/2020 1:49 AM	WHL File	247 KB
select.pyd	2/3/2020 11:02 AM	PYD File	27 KB
sqlite3.dll	2/3/2020 11:02 AM	Application extension	1,244 KB
unicodedata.pyd	2/3/2020 11:02 AM	PYD File	1,048 KB
vcomp140.dll	5/2/2019 7:27 PM	Application extension	156 KB
vcruntime140.dll	2/3/2020 11:02 AM	Application extension	84 KB
viewer.bat	1/22/2020 2:15 AM	Windows Batch File	1 KB
viewer-0.1.0-py3-none-any.whl	1/22/2020 2:06 AM	WHL File	20 KB
winsound.pyd	2/3/2020 11:02 AM	PYD File	29 KB

Fig.2 File list of the blochsolver-cpu folder just after the installation. Display the file extension.

2.1 Contents of the blochsolver-cpu folder

Most of the files in this folder are used by the execution program (blochsolver-cpu.exe), but there are also files that the user directly uses. These files are as follows.

In the **128_sequence** folder, pulse sequences for 2D (128 × 128 pixels) and 3D (128 × 128 × 32 or 128 × 128 × 128 voxels) image acquisition are stored. These pulse sequences use 128 × 128 × 128 voxel or 256 × 256 × 256 voxel numerical phantoms.

In the **256_sequence** folder, pulse sequences for 2D (256 × 256 pixels) and 3D (256 × 256 × 32 or 256 × 256 × 256 voxels) image acquisition are stored. These pulse sequences use 256 × 256 × 256 voxel numerical phantoms.

In the **b0map** folder, b0map files that represent magnetic field inhomogeneity are stored. The magnetic field is expressed by the precession frequency of the protons in Hz. The file format is the same as that of the numerical phantom.

In the **GUI_reconstruction_program** folder, GUI programs for image reconstruction and display are stored.

In the **phantom** folder, three types of 256 × 256 × 256 voxel numerical phantoms and three types of 128 × 128 × 128 voxel numerical phantoms are stored. They are SlicePhantom, NISTphantom, and BrainPhantom, which are described in detail in the **blochsolver** website.

Just after the initial **test operation**, the **result** folder and **blochsolver.ini** file are automatically created. The result folder stores simulation results (MR signals). The blochsolver.ini file is an initial parameter file.

Users use three files: **blochsolver.bat**, **command.txt**, and **viewer.bat**. **blochsolver.bat** is a batch file that start the blochsolver.exe. **command.txt** is a text file to provide parameters to the batch file. **viewer.bat** is a batch file to perform image reconstruction and image display.

2.2 Initial test

Just after the installation, start the initial simulation program by clicking the **blochsolver.bat** file. The simulation program automatically starts as shown in Fig.3. During the simulation, the phase encoding step is output every time the phase encoding loop is updated. If this window is clicked during the calculation, the simulation is temporarily interrupted. To restart the simulation, press the enter key. When the simulation is completed, the calculated MR signal (complex signal) data is stored in a folder named by the date automatically created in the result folder.

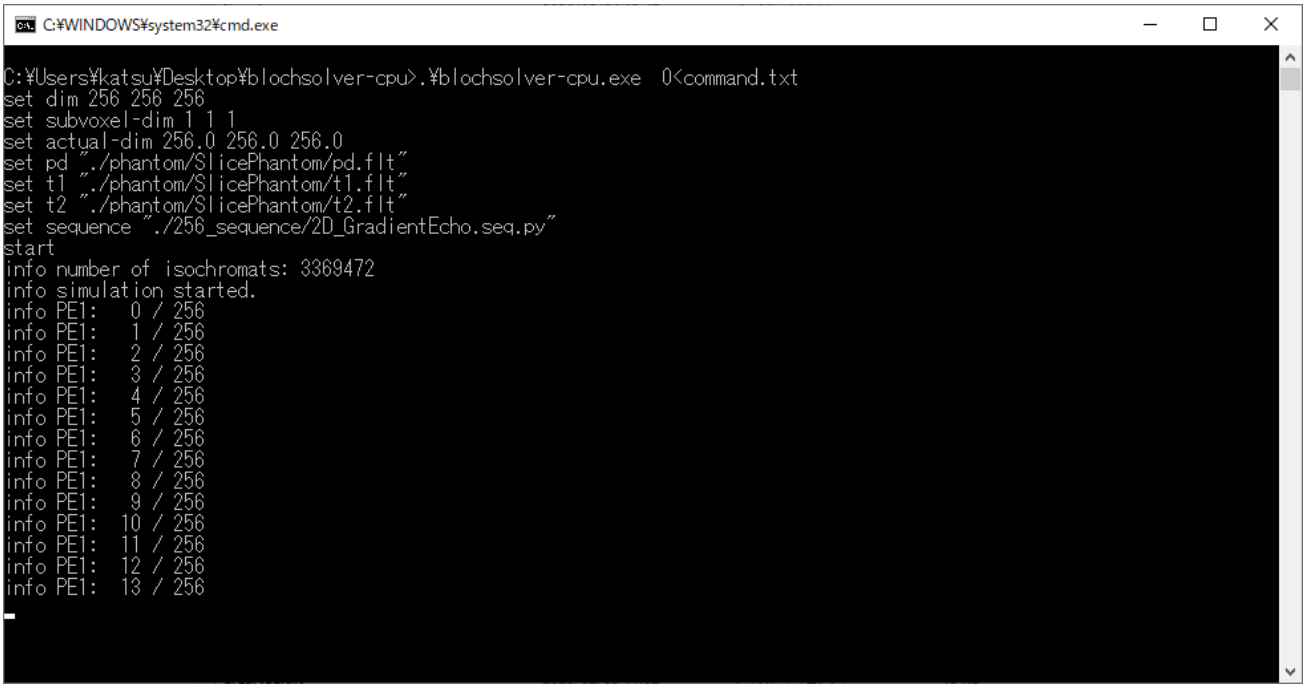


Fig.3 Command window for the blochsolver-cpu. 13th phase encoding step has just finished.

When the simulation is completed, close the window by the enter key. To reconstruct the MR image, start the program **Reconstruction_and_display_for_256x256_Cartesian.exe** in the **256** folder in the GUI_reconstruction_program. After the reconstruction program starts, select the “File” option in the main menu. If the MR signal file is properly selected, the MR image is automatically reconstructed and the following MR images are displayed.

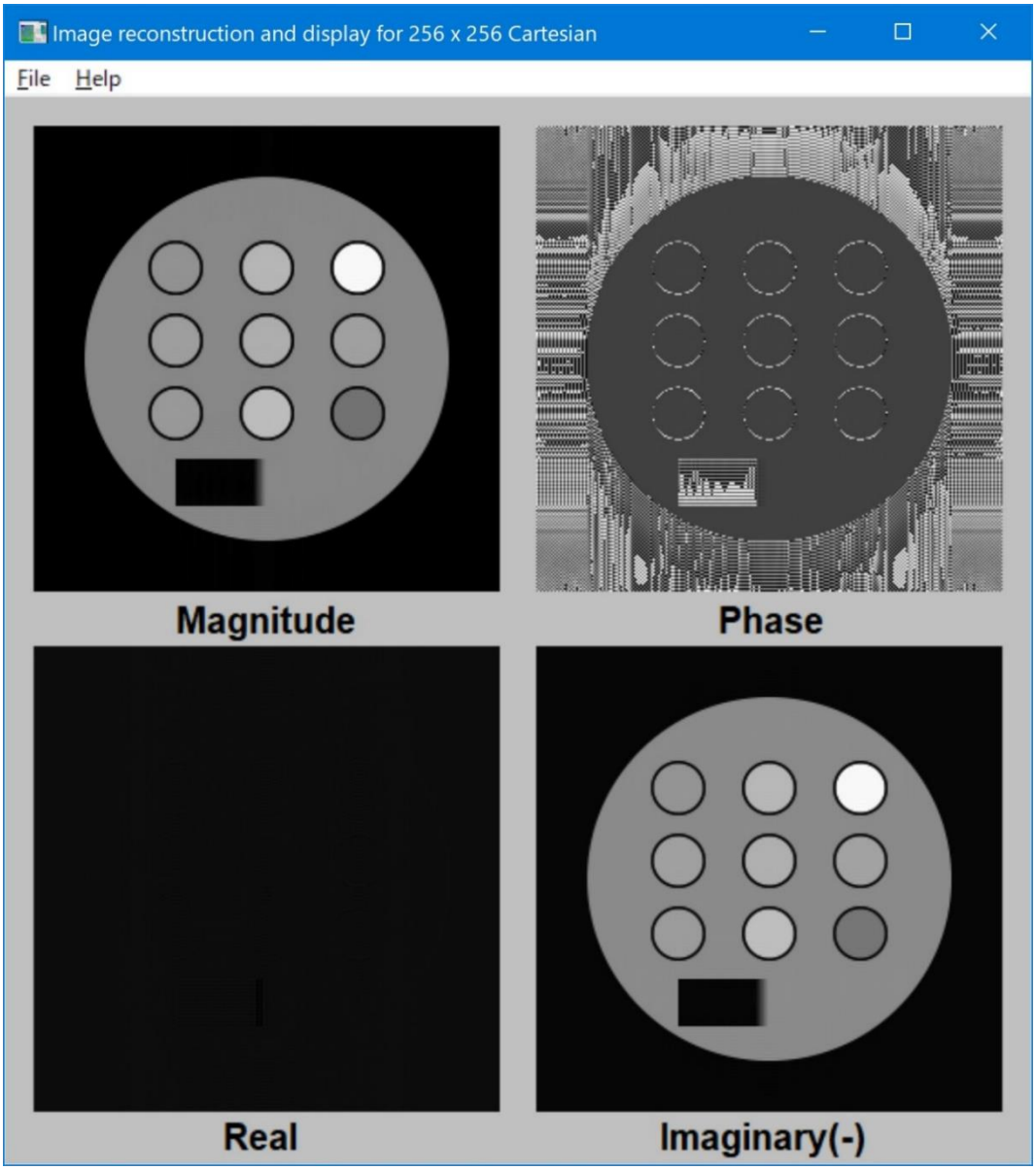


Fig.4 Reconstructed image. Magnitude, phase, real, and imaginary images are shown.
The magnetization refocuses along y in the rotating frame because the left-hand frame is used.

This image was obtained using the SlicePhantom (256 × 256 × 256 voxels) and a gradient echo sequence for a 256 × 256 pixel image. No artifact is observed even if the numbers of the **subvoxels** are 1, 1, and 1 in the x, y, and z directions because an inhomogeneous magnetic field is not applied.

Please use **Visual Studio Code** for editing the source code of the pulse sequence as shown below.

```python
from psdk import *
import numpy as np

gamma = 42.57747892 # [MHz/T]
TR = 20.0e+3 # [us]
TE = 6.0e+3 # [us]
NR = 256 # Number of readout points
NPE1 = 256 # Number of 1st phase encoding
fov = [256.0, 256.0, 256.0] # [mm]
dwell_time = 10.0 # [us]
slice_width = 5.0 # [mm]
gx_value = 1e+6 / (dwell_time * gamma * fov[0]) # [mT/m]
gy_value = 2e+6 / (dwell_time * gamma * fov[1]) * NPE1 / NR # [mT/m]
gz_value = 1.25 / (slice_width * 1.0e-3) / gamma # [mT/m]
gx_rt = 300.0 # [us] gx rise time
gy_rt = 300.0 # [us] gy rise time
gz_rt = 300.0 # [us] gz rise time
ex_pulse_width = 3200.0 # [us]
ex_pulse_flip_angle = 30.0 # [degree]

def sinc_with_hamming(flip_angle, pulse_width, points, *, min = -2.0 * np.pi, max = 2.0 * np.pi):
    x0 = np.arange(min, max, (max - min) / points)
    x1 = x0 + (max - min) / points
    y = (np.sinc(x0 / np.pi) + np.sinc(x1 / np.pi)) * 0.5 * np.hamming(points)
    return flip_angle * y * points / (y.sum() * pulse_width * 360.0e-6 * gamma)

with Sequence('2D GradientEcho'):

    with Block('Excitation', ex_pulse_width + 2.0*gz_rt):
        GZ(0.0, gz_value, gz_rt)
        RF(gz_rt, sinc_with_hamming(ex_pulse_flip_angle, ex_pulse_width, 160), ex_pulse_width / 160)
        GZ(ex_pulse_width + gz_rt, 0.0, gz_rt)

    with Block('PhaseEncoding', NR // 2 * dwell_time + gx_rt * 2.5):
        GX(0.0, -gx_value, gx_rt)
        GY(0.0, ([gy_value * (i - NPE1 // 2) / NPE1 for i in range(NPE1)], ['PE1']), gy_rt)
        GY(NR // 2 * dwell_time, 0.0, gy_rt)
        GX(NR // 2 * dwell_time + gx_rt * 0.5, gx_value, gx_rt * 2.0)
        GZ(0.0, -gz_value, gz_rt * 0.6)
        GZ(ex_pulse_width * 0.5 + 150.0, 0.0, gz_rt * 0.6)

    with Block('Readout', NR * dwell_time):
        AD(0.0, NR, dwell_time)

    with Block('Rewinding', NR // 2 * dwell_time + gx_rt):
        GY(0.0, ([gy_value * (NPE1 // 2 - i) / NPE1 for i in range(NPE1)], ['PE1']), gy_rt)
        GX(NR // 2 * dwell_time - gx_rt * 0.5, 0.0, gx_rt)
        GY(NR // 2 * dwell_time, 0.0, gy_rt)

    with Main():
        with Loop('PE1', NPE1):
            BlockRef('Excitation')
            WaitUntil(TE + ex_pulse_width * 0.5 + gz_rt - NR // 2 * 2 * dwell_time - gx_rt * 2.5)
            BlockRef('PhaseEncoding')
            BlockRef('Readout')
            BlockRef('Rewinding')
            WaitUntil(TR)
```

Gradient echo sequence written in PSDK. See website for detail.

The concept of the **subvoxel** is shown in Fig.5. Subvoxels are made by dividing a voxel into equal intervals in the x, y, and z directions. **One isochromat is placed in one subvoxel**. Subvoxels are required in the following cases in the Bloch image simulation.

(1) Phase variation by the static magnetic field inhomogeneity over a voxel is larger than about $\pi/2$ (*). (*) this value depends on the homogeneity of the image intensity you want in the simulated image. The smaller the better.

(2) To reproduce the slice profile by a selective excitation pulse. Roughly speaking, about 50 subvoxels are required to describe the slice profile exactly. For example, when the size of the voxel along the direction perpendicular to the slicing plane is 1 mm and the slice thickness is 5 mm, 4-8 subvoxels are required (: depends on the shape of the selective pulse).

(3) T_1, T_2 << TR or T_1, T_2 << echo spacing (FSE). Spin echo, stimulated echo, and higher order echo affect the intensity of the gradient echo or the spin echo.

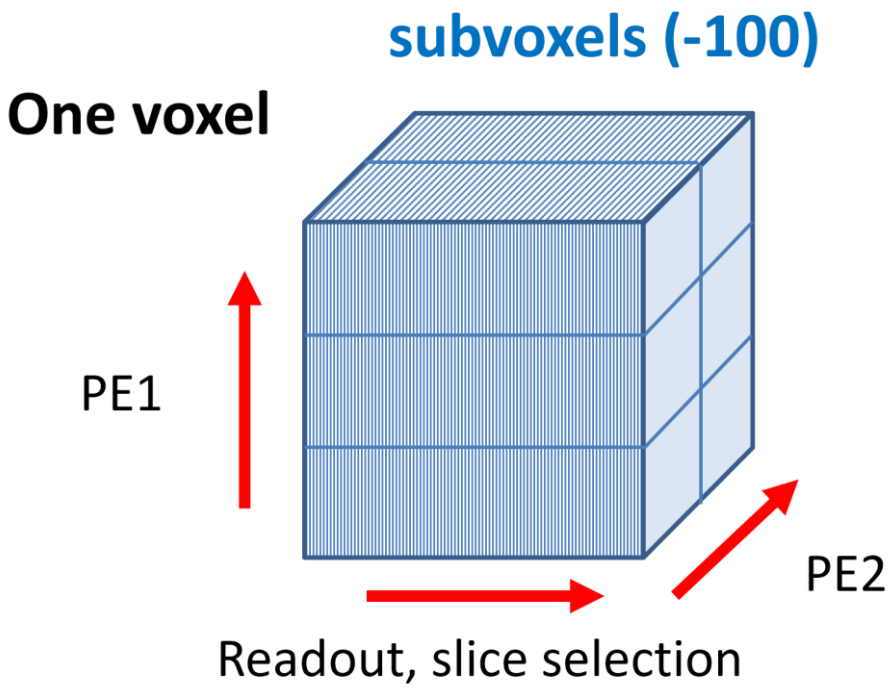


Fig.5 Concept of subvoxel. **One isochromat is placed in one subvoxel**.

3. Preparation for Bloch image simulation and image reconstruction

3.1 Preparation for Bloch image simulation

As described in the previous section, BlochSolver is started by blochsolver.bat. As shown in Fig.6, blochsolver(.exe) starts after parameters entered from **command.txt**.

```
1    blochsolver < command.txt
2    pause
```

Fig.6 Content of blochsolver.bat

The initial setting of command.txt is shown as follows:

```
1    set dim 256 256 256
2    set subvoxel-dim 1 1 1
3    set actual-dim 256.0 256.0 256.0
4    set pd "./phantom/SlicePhantom/pd.flt"
5    set t1 "./phantom/SlicePhantom/t1.flt"
6    set t2 "./phantom/SlicePhantom/t2.flt"
7    set sequence "./256_sequence/2D_GradientEcho.seq.py"
8    start
9    var PREFIX "./result/${DATE}/${SEQUENCE_NAME}-${SERIAL_ID}"
10   save complex "${PREFIX}-complex.flt"
11   quit
```

Fig.7 Command.txt (initial setting)

```
1    set dim 256 256 256
```

Set matrix size of the numerical phantom 256 × 256 × 256 (x, y, and z).

```
2    set subvoxel-dim 1 1 1
```

Set the numbers of subvoxels 1 × 1 × 1 (x, y, and z).

```
3    set actual-dim 256.0 256.0 256.0
```

Set the physical size of the phantom 256.0 mm × 256.0 mm × 256.0 mm (x, y, and z).

```
4    set pd "./phantom/SlicePhantom/pd.flt"
```

Set the file name of the proton density map file.

```
5    set t1 "./phantom/SlicePhantom/t1.flt"
```

Set the file name of the T_1 map file.

```
6    set t2 "./phantom/SlicePhantom/t2.flt"
```

Set the file name of the T_2 map file.

```
7    set sequence "./256_sequence/2D_GradientEcho.seq.py"
```

Set the sequence file name.

```
8    start
```

Start blochsolver.exe.

```
9    var PREFIX "./result/${DATE}/${SEQUENCE_NAME}-${SERIAL_ID}"
```

Set PREFIX variable.

```
10    save complex "${PREFIX}-complex.flt"
```

Save the simulation result (complex) as the defined file name.

```
11    quit
```

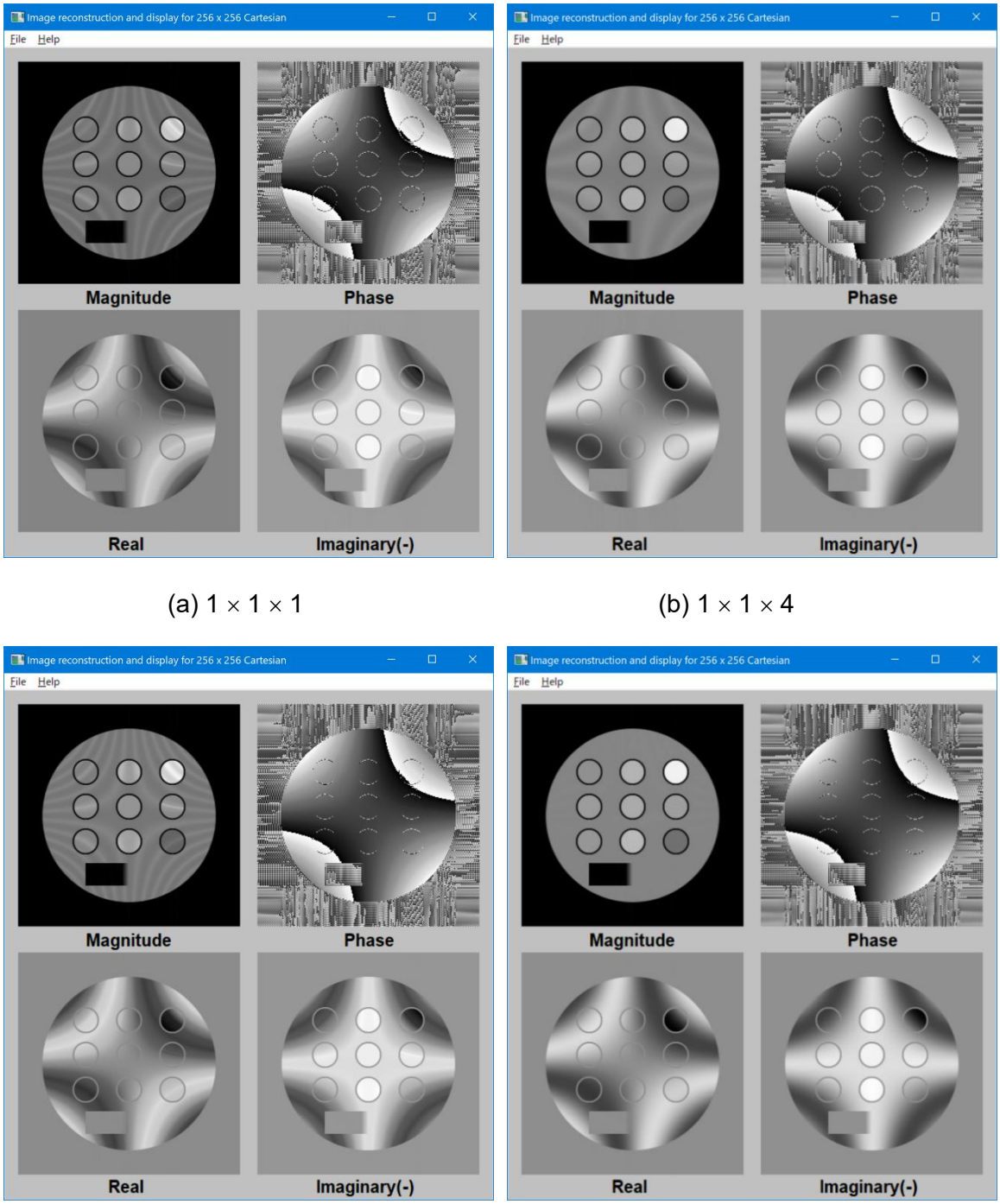
Complete the program.

Next, we introduce magnetic field inhomogeneity to command.txt as shown in Fig.8. To introduce magnetic field inhomogeneity, set a file (b0map_xy.flt) with the same matrix size as the numerical phantom (256 × 256 × 256), which represents the spatial distribution of the static magnetic field by the precession frequency distribution (Hz) in the line 7. **factor** is a magnification factor for the number in the file and **offset** is a bias to the number in the file written in Hz.

```
1    set dim 256 256 256
2    set subvoxel-dim 2 1 4
3    set actual-dim 256.0 256.0 256.0
4    set pd "./phantom/SlicePhantom/pd.flt"
5    set t1 "./phantom/SlicePhantom/t1.flt"
6    set t2 "./phantom/SlicePhantom/t2.flt"
7    set b0 "./b0map/b0map_xy.flt" factor=4.0 offset=0.0
8    set sequence "./256_sequence/2D_GradientEcho.seq.py"
9    start
10   var PREFIX "./result/${DATE}/${SEQUENCE_NAME}-${SERIAL_ID}"
11   save complex "${PREFIX}-complex.flt"
12   quit
```

Fig.8 How to introduce magnetic field inhomogeneity

The simulated results are shown below. The captions below the figures show numbers of subvoxels (x, y, and z). Image artifacts (stripe pattern) decrease with increase of the number of subvoxels. This example clearly shows importance of subvoxels.



(a) 1 × 1 × 1 (b) 1 × 1 × 4

(c) 2 × 1 × 1 (d) 2 × 1 × 4

Fig.9 Simulation results for a 2D GRE sequence when a magnetic field inhomogeneity is present

3.2 Image reconstruction and display programs

We provide two kinds of image reconstruction and display programs. The first set of programs was developed for specialized pulse sequences stored in the **GUI_reconstruction_program folder**. The second one is the general-purpose program named **viewer.bat**.

The GUI_reconstruction_program folder includes two folders named **128** and **256**. In the 128 folder, image reconstruction and display programs for 128 × 128 or 128 × 128 × 128 voxels are stored (Fig. 10). In the 256 folder, image reconstruction and display programs for 256 × 256 or 256 × 256 × 256 voxels are stored (Fig.11). According to the name of the program, the MR signal should be entered to the program for image reconstruction. When the program is developed for multislice or 3D imaging, only the central cross-section is displayed on the window. All of the reconstructed image data are stored in the same folder as **abs.flt**, **phase.flt**, **real.flt**, and **imaginary.flt** when the GUI program is closed. These image data have a simple image format (no header) with single precision floating point number (32 bit). These images can be displayed viewer.bat or ImageJ (import as 32-bit Real).

Name	Type	Size
Reconstruction_and_display_for_128x128_2D_Cartesian.exe	Application	128 KB
Reconstruction_and_display_for_128x128_2D_EPI.exe	Application	129 KB
Reconstruction_and_display_for_128x128_FSE_PDW.exe	Application	128 KB
Reconstruction_and_display_for_128x128_FSE_T2W.exe	Application	128 KB
Reconstruction_and_display_for_128x128x16_2D_multiecho.exe	Application	128 KB
Reconstruction_and_display_for_128x128x24_2D_multislice.exe	Application	128 KB
Reconstruction_and_display_for_128x128x24_2D_FSE_PDW.exe	Application	129 KB
Reconstruction_and_display_for_128x128x24_2D_FSE_T2W.exe	Application	129 KB
Reconstruction_and_display_for_128x128x32_3D.exe	Application	129 KB
Reconstruction_and_display_for_128x128x128_3D.exe	Application	129 KB

Fig.10 Reconstruction and display programs for 128^2 or 128^3 voxel images

Name	Type	Size
Reconstruction_and_display_for_256x256_Cartesian.exe	Application	129 KB
Reconstruction_and_display_for_256x256_FSE_PDW.exe	Application	129 KB
Reconstruction_and_display_for_256x256_FSE_T2W.exe	Application	129 KB
Reconstruction_and_display_for_256x256x16_2D_multiecho.exe	Application	129 KB
Reconstruction_and_display_for_256x256x24_2D_FSE_PDW.exe	Application	129 KB
Reconstruction_and_display_for_256x256x24_2D_FSE_T2W.exe	Application	129 KB
Reconstruction_and_display_for_256x256x24_2D_multislice.exe	Application	129 KB
Reconstruction_and_display_for_256x256x32_3D.exe	Application	130 KB
Reconstruction_and_display_for_256x256x256_3D.exe	Application	129 KB

Fig.11 Reconstruction and display programs for 256^2 or 256^3 voxel images

Start window of **viewer.bat** is shown below.

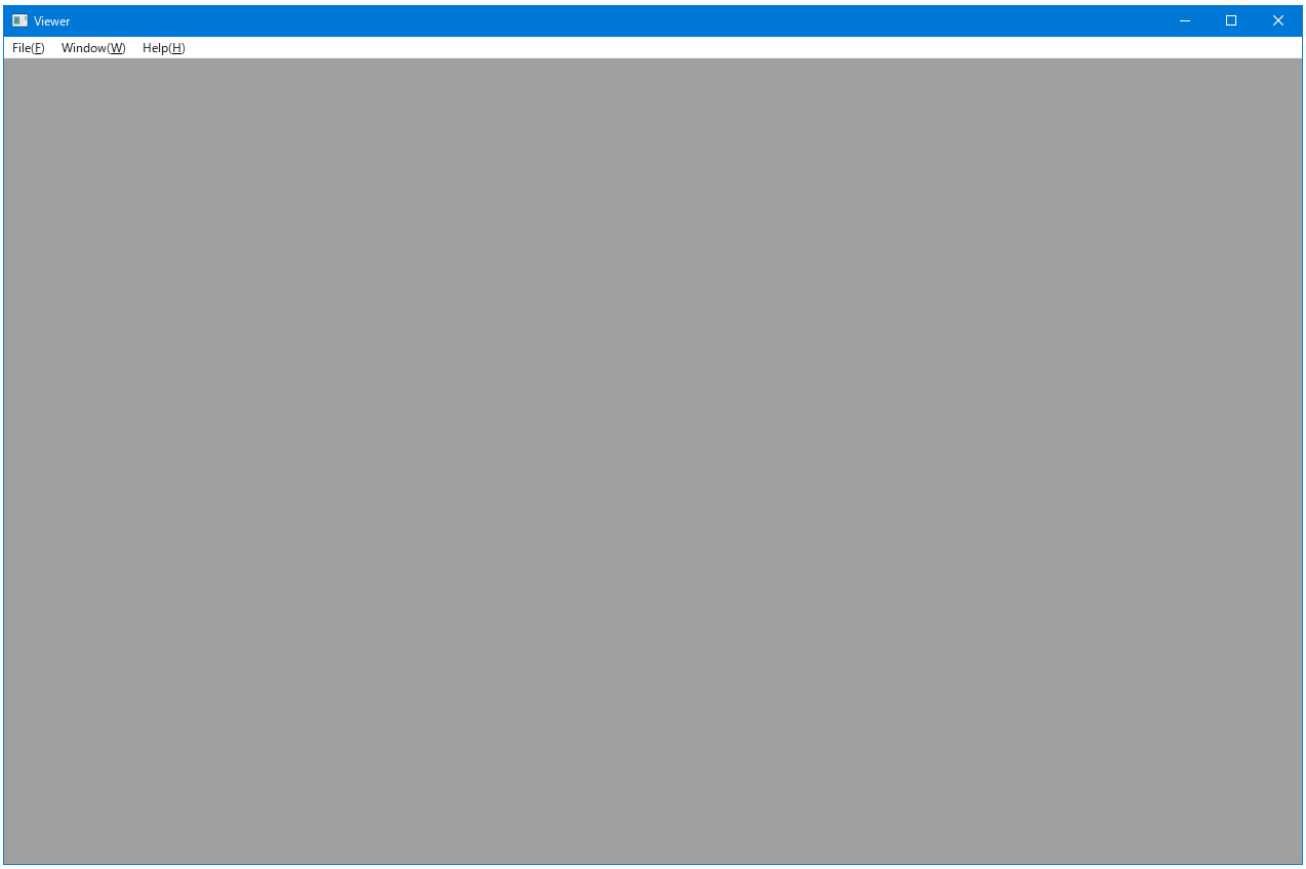


Fig.12 Start window of viewer.bat

When you press **File** menu and select **Open** item, the following dialog box will open.

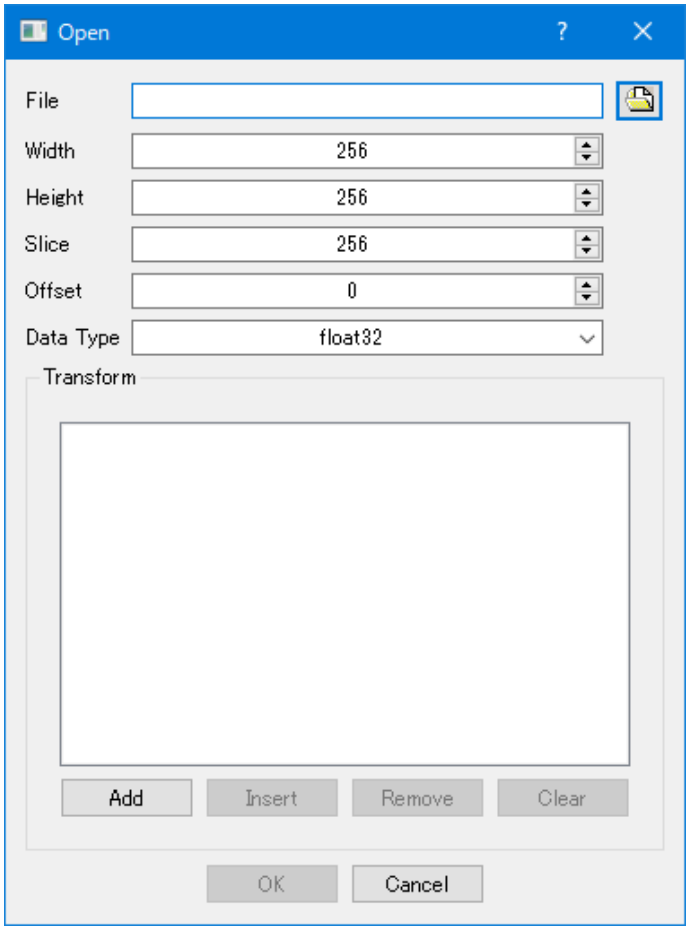


Fig.13 Fundamental dialog box for viewer.bat

To select a file for processing or display, press the **file icon** shown in the right of the text box named File. Then, **File open dialog box** will open, and select the file name you want to process.

In the text boxes named **Width**, **Height**, and **Slice**, enter the matrix size of the input data. The default values for the boxes are 256, 256, and 256, but any value is permitted (only limited by memory size). The name width, height, and slice correspond to the data array array[slice][height][width] in C language expression. When a 3D data set is displayed, the first display plane is a width × height plane, and the number of the plane displayed by the slice number is scrolled by the mouse roller.

In the text box labeled **Offset**, enter the number of bytes from the beginning of the file to the image data, and use it when there is an image header.

For the text box named **Data Type**, press the menu symbol and select data type from the dropdown menu as shown in Fig.14. The data type of the MR signal output by BlochSolver-CPU is **complex64** that consists of two single-precision floating-point numbers (float32) for the real and imaginary part numbers. For display of a numerical phantom, select float32.

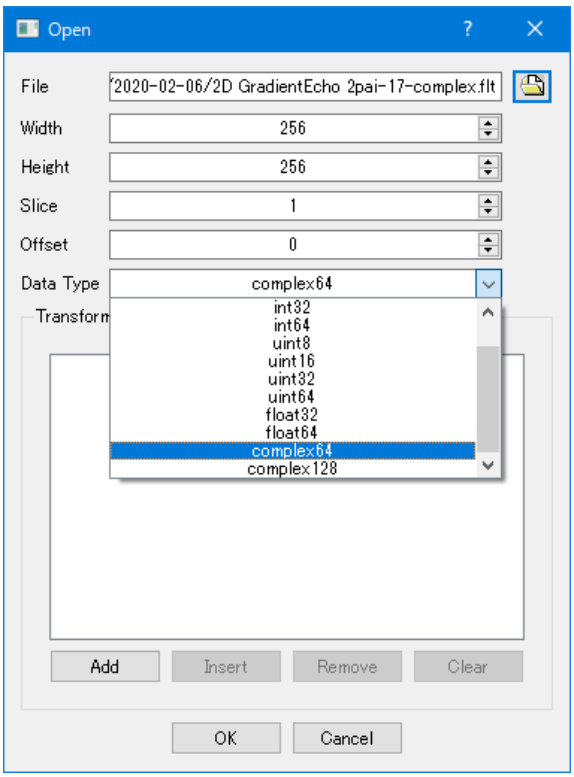


Fig.14 Selection of data type from the drop down menu

For the box indicated by **Transform** (or Processing), press the **Add** button to select the data processing methods for the input data. When the selection of the data processing is completed, press **OK** to execute the processing. Complex data processing can be performed by combining multiple processing items. In the next section, we introduce the processing method and the image reconstruction results for each MR signal data.

4. Simulation examples

Figure 15 shows the calculation time when the simulation was performed using the various pulse sequences. The hardware used was the CPU (Core i7 7700HQ (4 core)) and GPU (GTX 1070 (2560 core)) of the PC shown in Fig.16. Figure 17 shows calculation times of the CPU plotted against those of the GPU for 2D imaging. Figure 18 shows calculation times of the CPU for 256^2 pixel image plotted against those for 128^2 pixels.

Hardware		CPU (Core i7 7700HQ, 4 core)		GPU (GTX 1070, 2560 core)	
Sequence	pixel / subvoxel	128^2 / 128^3	256^2 / 256^3	128^2 / 128^3	256^2 / 256^3
2D Gradient Echo	1 × 1 × 1	15.8	220.7	0.58	7.2
2D Spin Echo	1 × 1 × 1	24.8	401.7	1.0	12.7
2D Inversion Recovery	1 × 1 × 1	50.1	783.1	2.0	24.0
2D PDW FSE	1 × 1 × 16	445.5	7082.7	13.7	207.6
2D T2W FSE	1 × 1 × 32	879.2	14148.0	26.1	403.0
2D multiple SE	1 × 1 × 4	887.9	14772.1	29.4	442.1
2D multislice GRE	1 × 1 × 4	1505.1	21002.0	41.8	631.8
2D multislice SE	1 × 1 × 4	2396.7	38066.0	77.6	1143.9
2D multislice PDW FSE	1 × 1 × 8	5256.0	-	163.1	2506.4
2D multislice T2W FSE	1 × 1 × 8	5304.9	-	162.6	2504.5
3D GRE (32 slice)	1 × 1 × 1	76.1	1735.5	4.2	58.5
3D GRE (128 / 256 slice)	1 × 1 × 1	300.8	13877.1	16.1	451.3
2D Gradient echo EPI	1 × 1 × 1	1.58	11.6	0.1	0.4

Fig.15 Calculation time (CPU vs GPU) measured in second.

Fig.16 Laptop gaming PC (OMEN by HP Laptop 17-an0xx) used for simulation

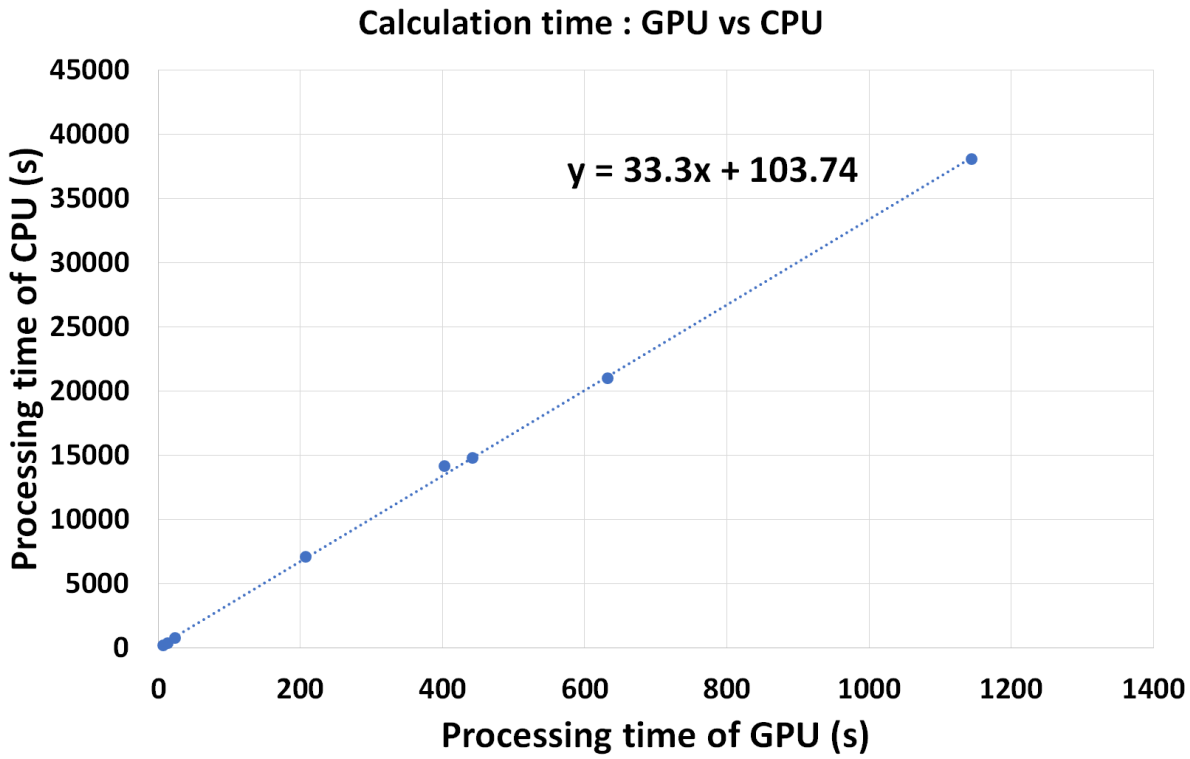


Fig.17 Calculation time of the CPU plotted against that of the GPU for 2D imaging

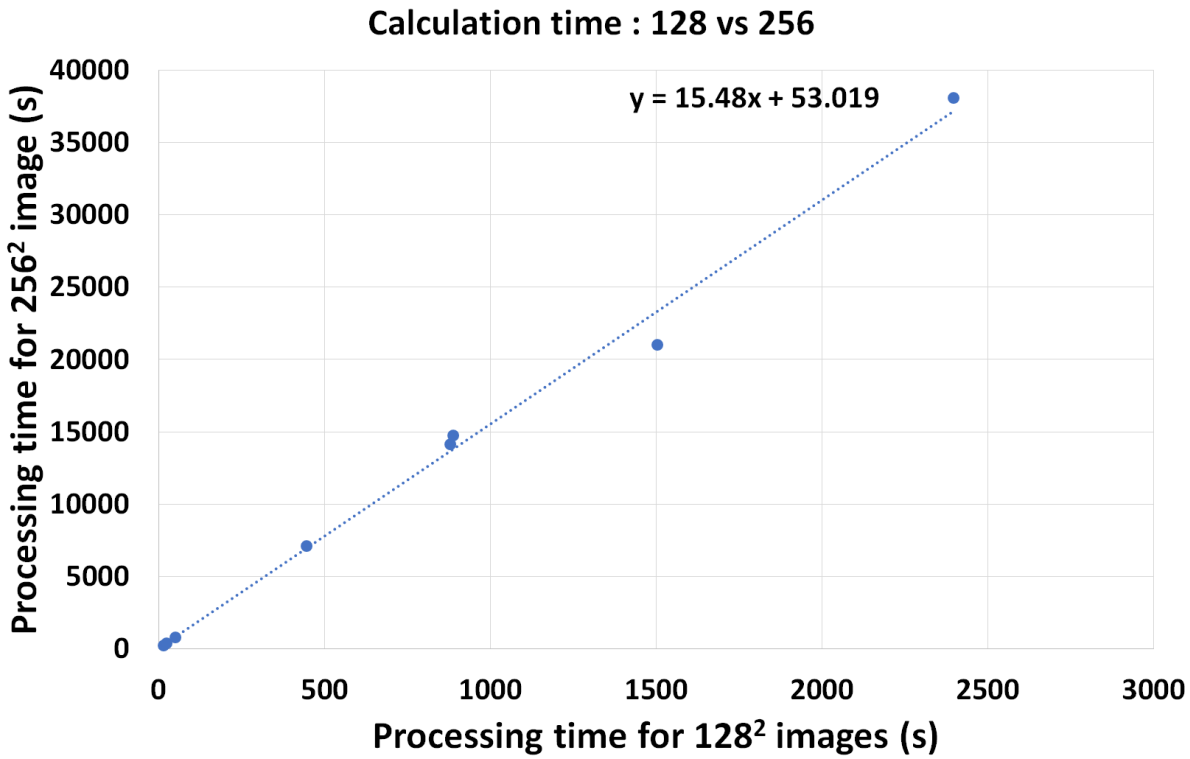


Fig.18 Calculation time of the CPU (128^2 pixels vs 256^2 pixels)

4-1 2D Gradient echo

To reconstruct of a 256 × 256 pixel image, operate in the following way. After selecting the file to be reconstructed in the dialog box shown in Fig.14, set the data size to 256, 256, 1 as shown in Fig.19 and select complex64 for the data type. Next, press the Add button and select FFT, and press configure as shown in Fig.20. In the configuration dialog box, checkout the Z box (do not FFT for the z axis) and press the OK button as shown in Fig.21. Then press the OK button in the bottom of the dialog box. The reconstructed result is displayed in the main window as shown in Fig.22

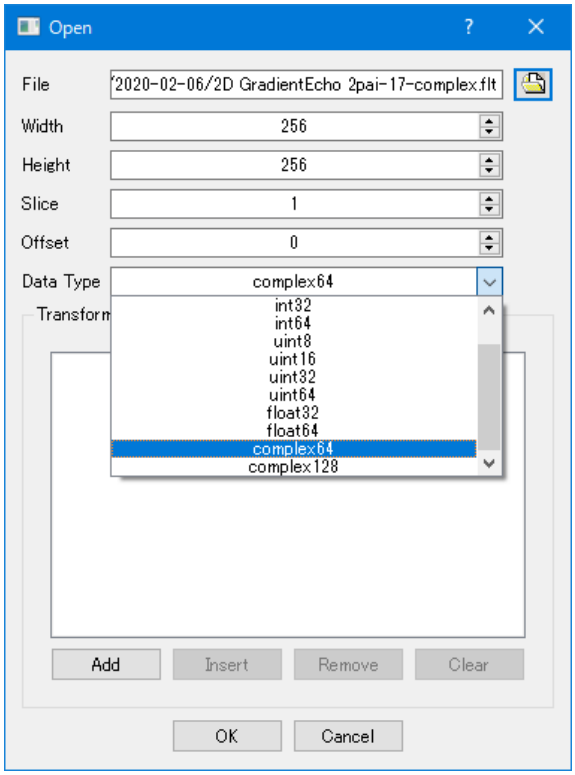


Fig.19 Selection of Data type

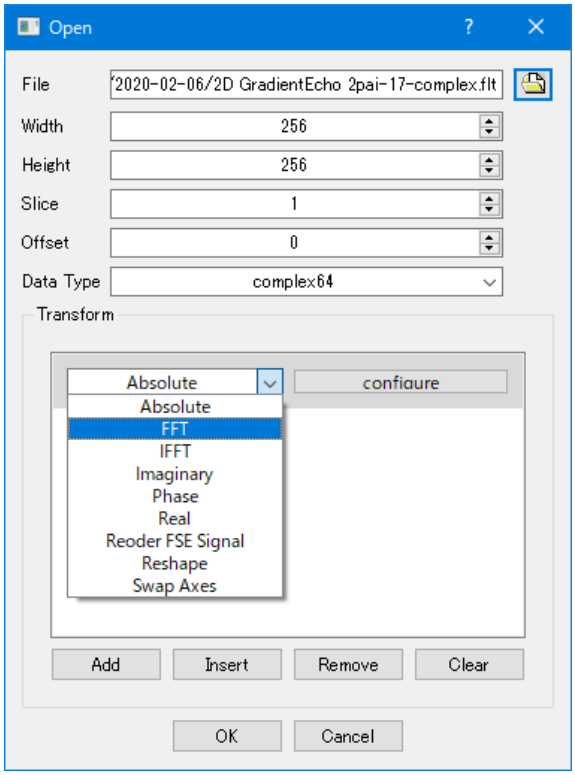


Fig.20 Selection of Processing

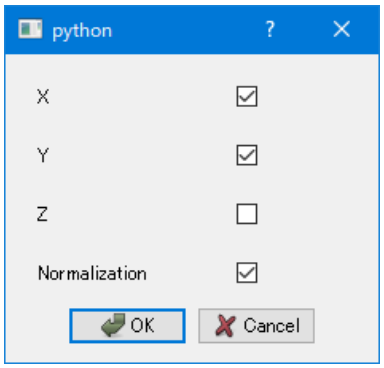


Fig.21 Configuration for FFT

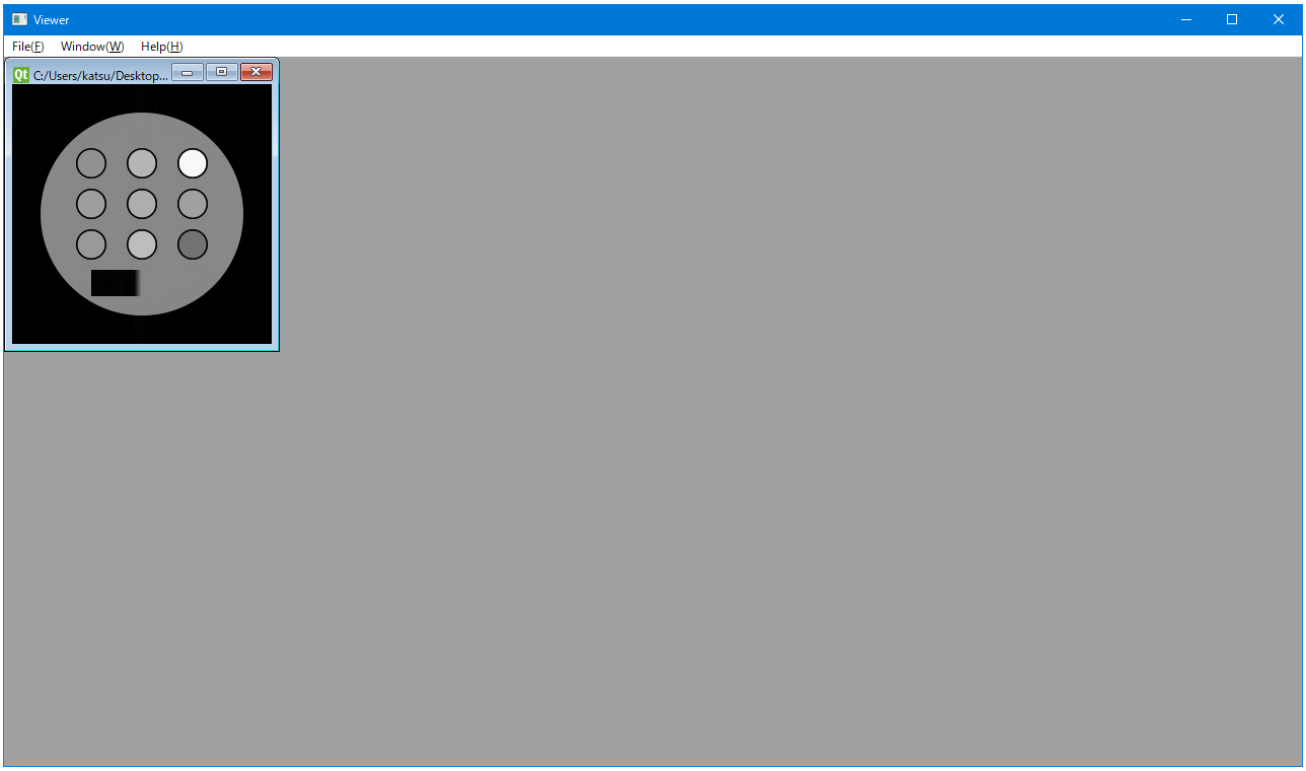


Fig.22 Reconstructed image displayed in the main window.

Figure 23 shows simulated gradient echo images for 128 × 128 and 256 × 256 pixel images.

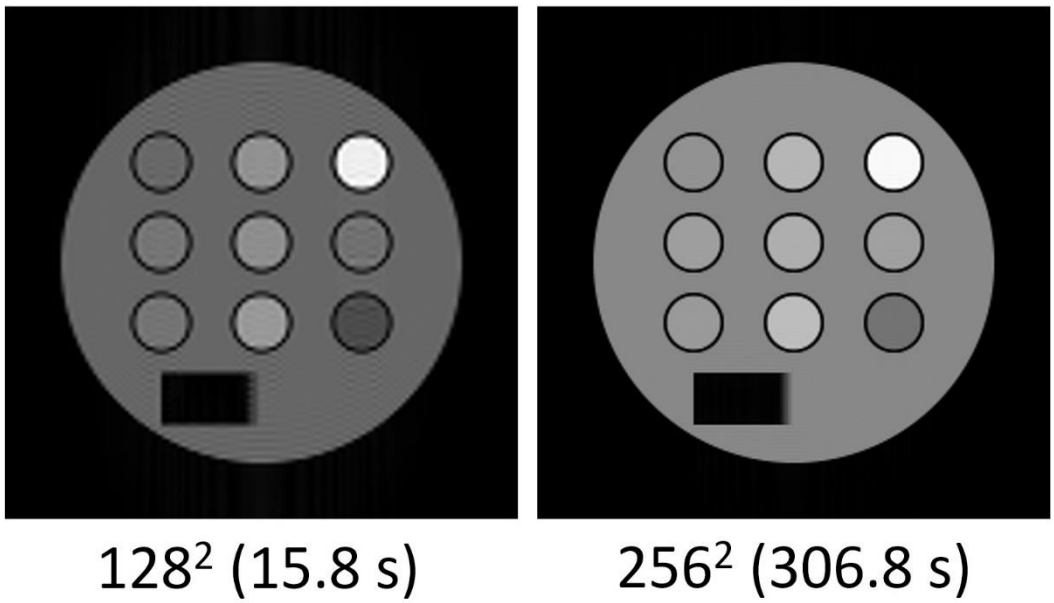


Fig.23 Simulated gradient echo images with 128^2 and 256^2 pixels

4-2 2D Spin Echo

The reconstruction method for the 2D SE method is same as that for the 2D GRE method.

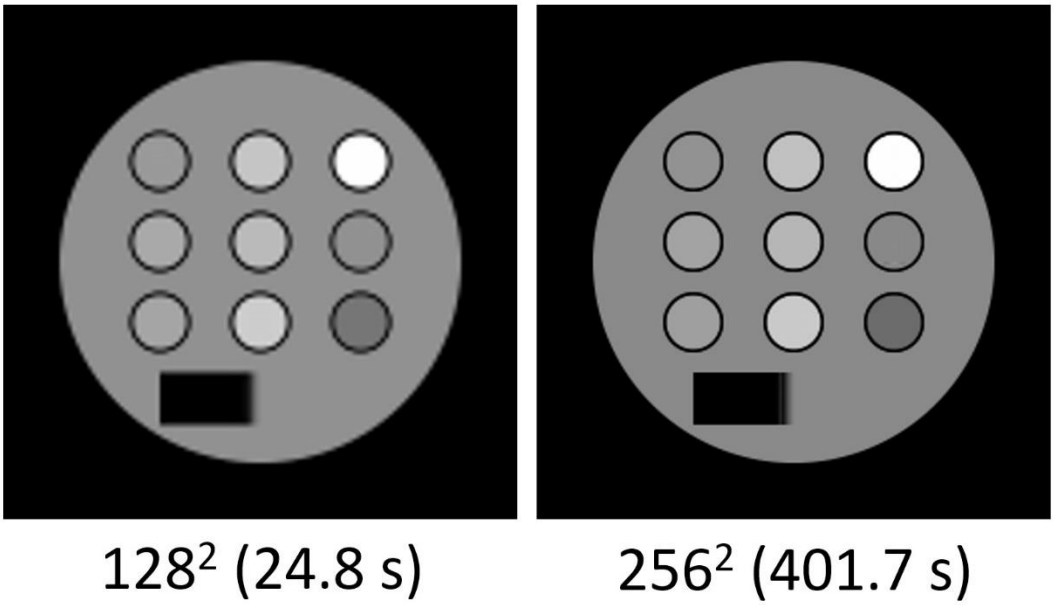


Fig.24 Simulated spin echo images with 128^2 and 256^2 pixels

4-3 2D Inversion Recovery

The reconstruction method for the 2D IR method is same as that for the 2D GRE method.

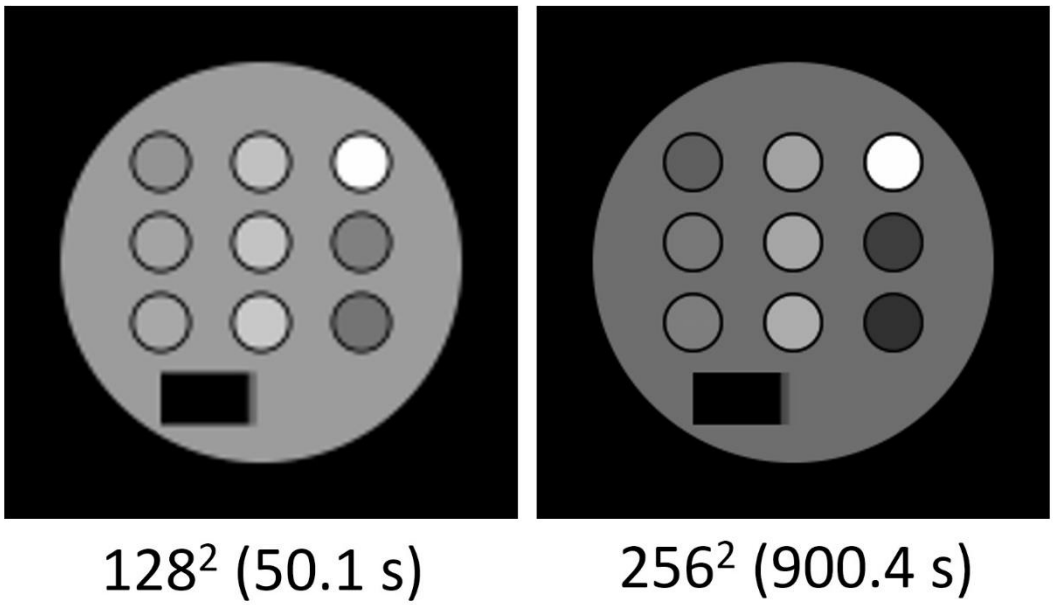


Fig.25 Simulated inversion recovery images with 128^2 and 256^2 pixels

4-4 Proton density weighted fast spin echo

For the reconstruction of the single slice PD weighted FSE image, after setting the parameters as in Fig.26, open the MR signal file.

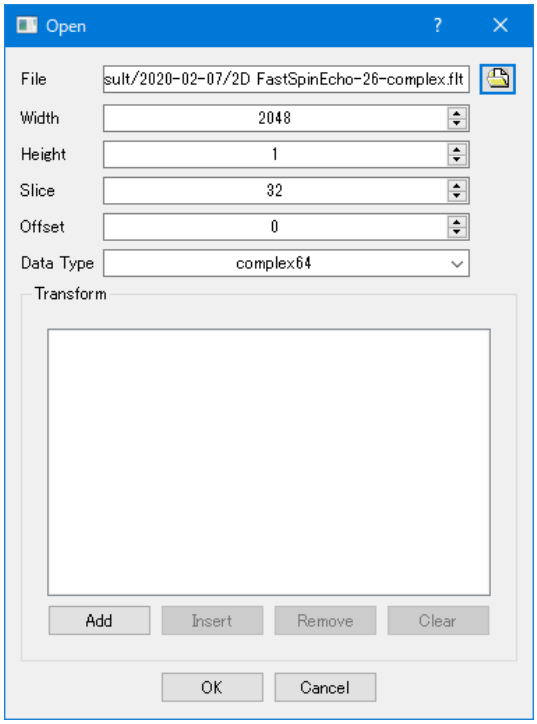


Fig.26 Dialog box to input single slice 2D PDW FSE data

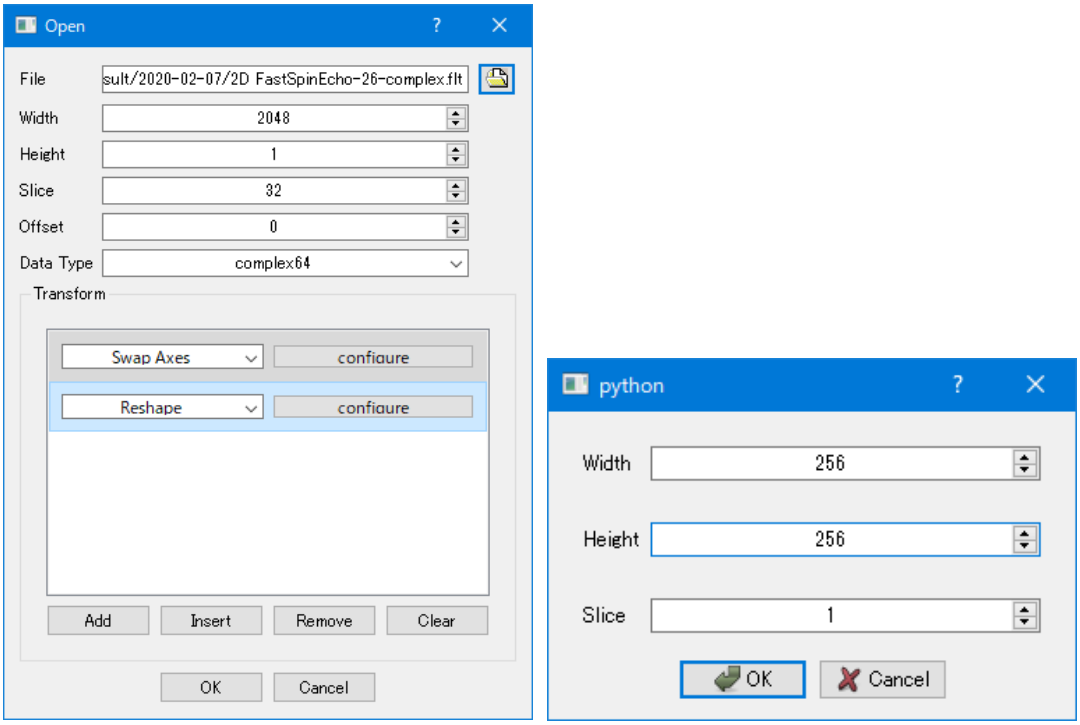


Fig.27 Add Swap axes and Reshape, and set the configuration box as above

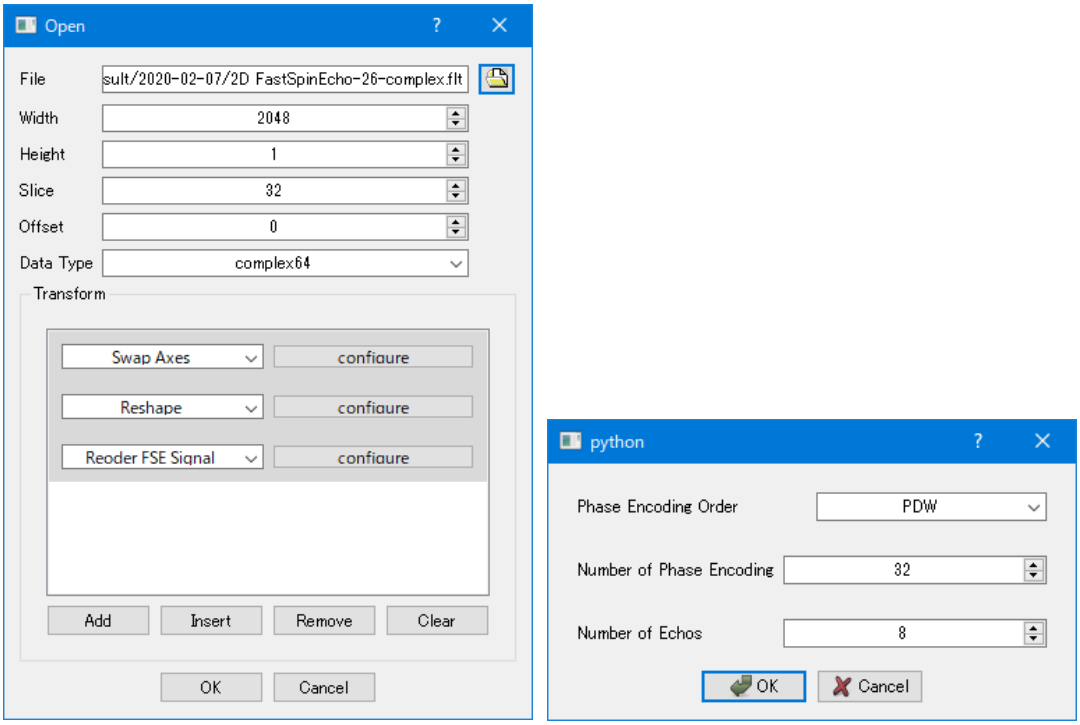


Fig.28 Add Reorder FSE signal and set the configuration box as above

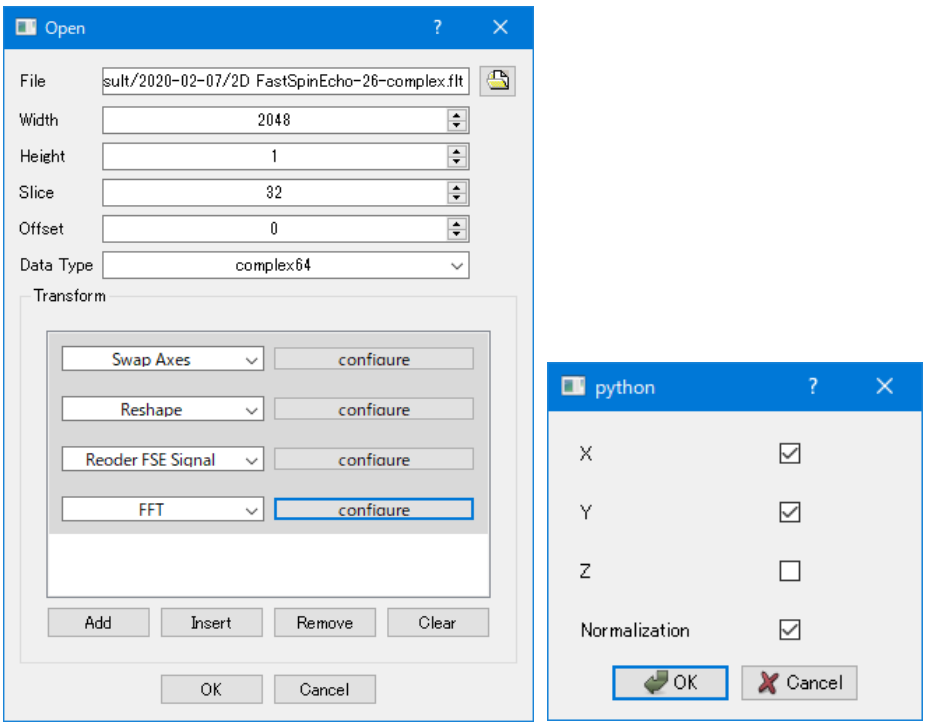


Fig.29 Add FFT and set the configuration box as above

After the above settings are made and the OK button is pressed, the reconstructed image is displayed on the window.

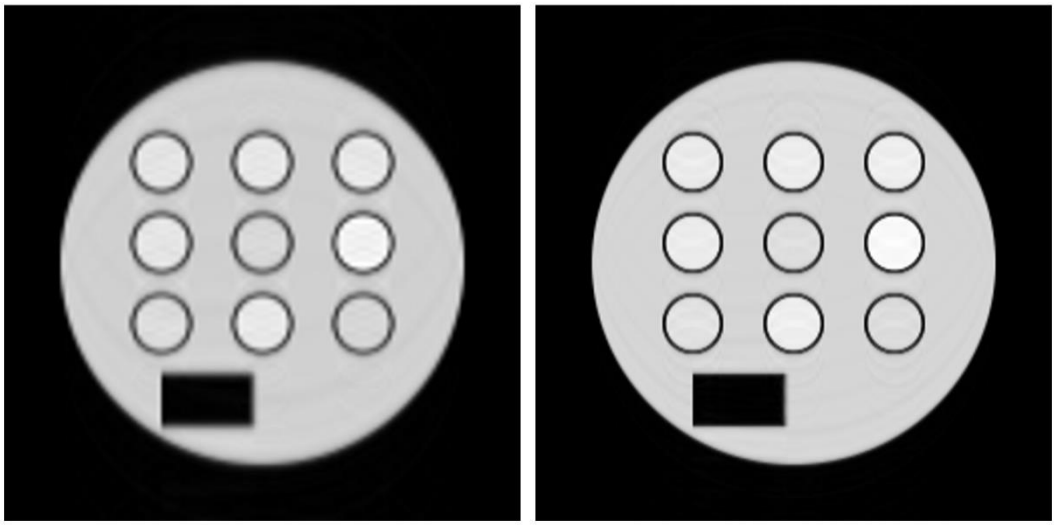


Fig.30 Proton density weighted FSE images with 128^2 and 256^2 pixels

When reconstructing an image with 128 × 128 pixels, set 2048 → 1024, 32 → 16 in Fig. 26, 128, 128, 1 in the configuration box in Fig.27, and 16, 8 in the configuration box in Fig. 28.

4-5 T2 weighted fast spin echo

For the reconstruction of the single slice T2 weighted FSE image, after setting the parameters as in Fig.26, open the MR signal file. Other operations are the same as in the case of PDW FSE, but T2W is selected for the configuration box of Reorder FSE signal in Fig. 31.

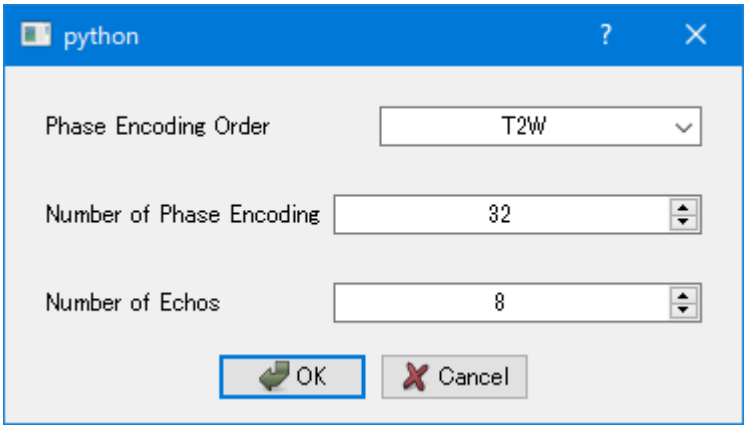


Fig.31 The configuration box for Reorder FSE signal

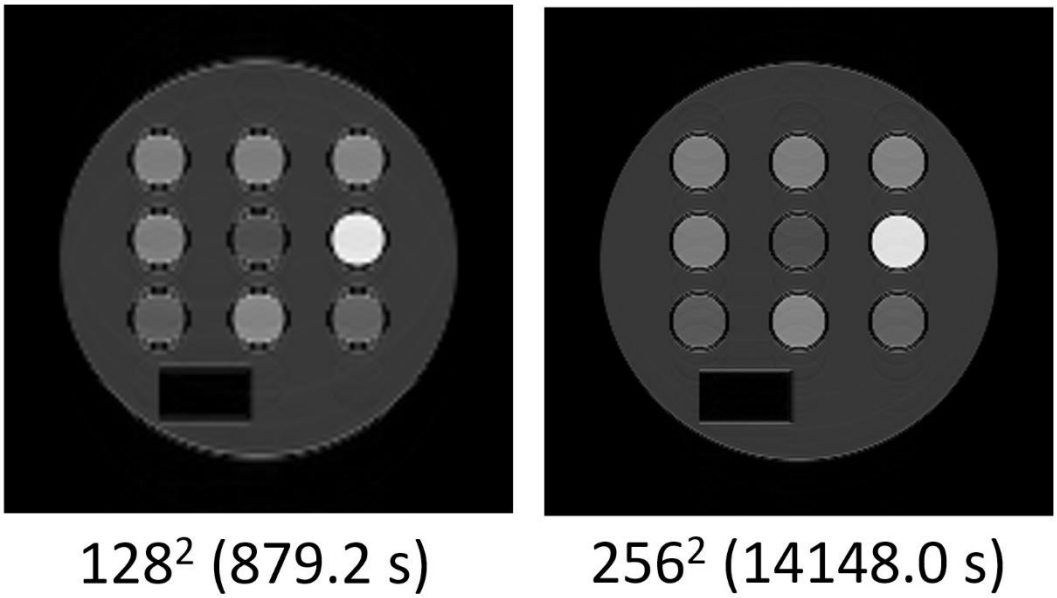


Fig.32 T2 weighted FSE images with 128^2 and 256^2 pixels

When reconstructing an image with 128 × 128 pixels, set 2048 → 1024, 32 → 16 in Fig. 26, 128, 128, 1 in the configuration box in Fig.27, and 16, 8 in the configuration box in Fig. 28.

4-6 Multiple spin echo

For the reconstruction of the multiple SE image, after setting the parameters as in Fig.33, open the MR signal file. For the configuration dialog box, push OK after setting the numbers as in Fig.33. Next, add FFT, checkout the z box in the configuration dialog box, and push OK as in Fig.34.

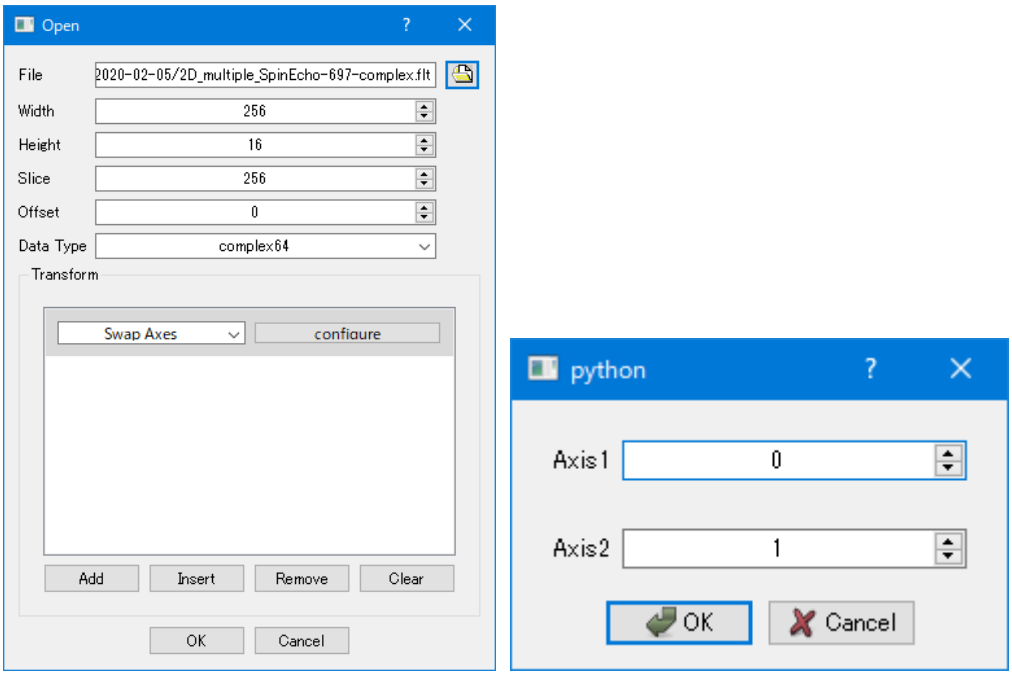


Fig.33 Dialog box to input a multiple SE signal file and the configuration dialog box for Swap axes

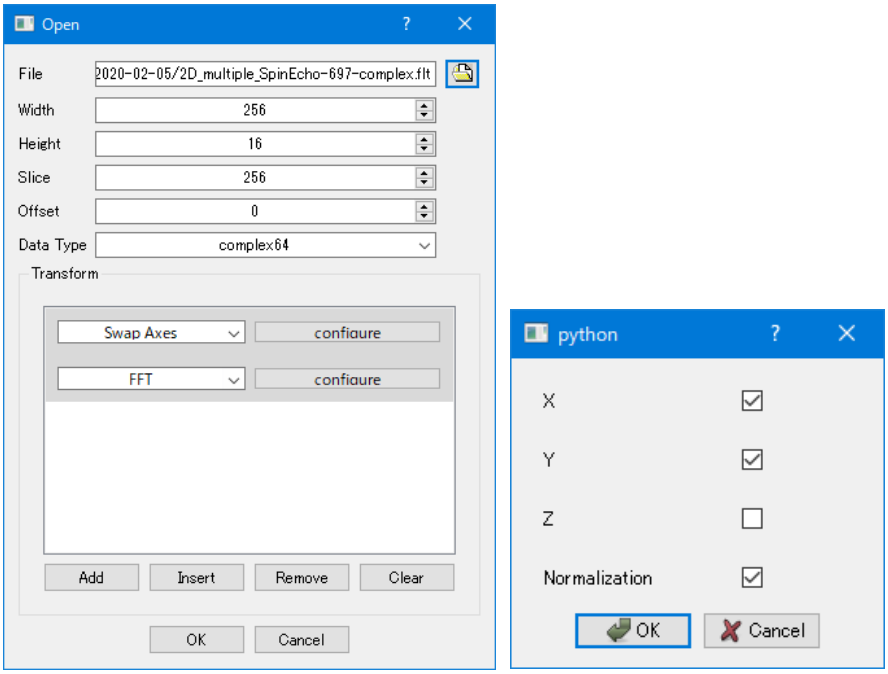


Fig.34 Add FFT and checkout z box in the configuration box of FFT

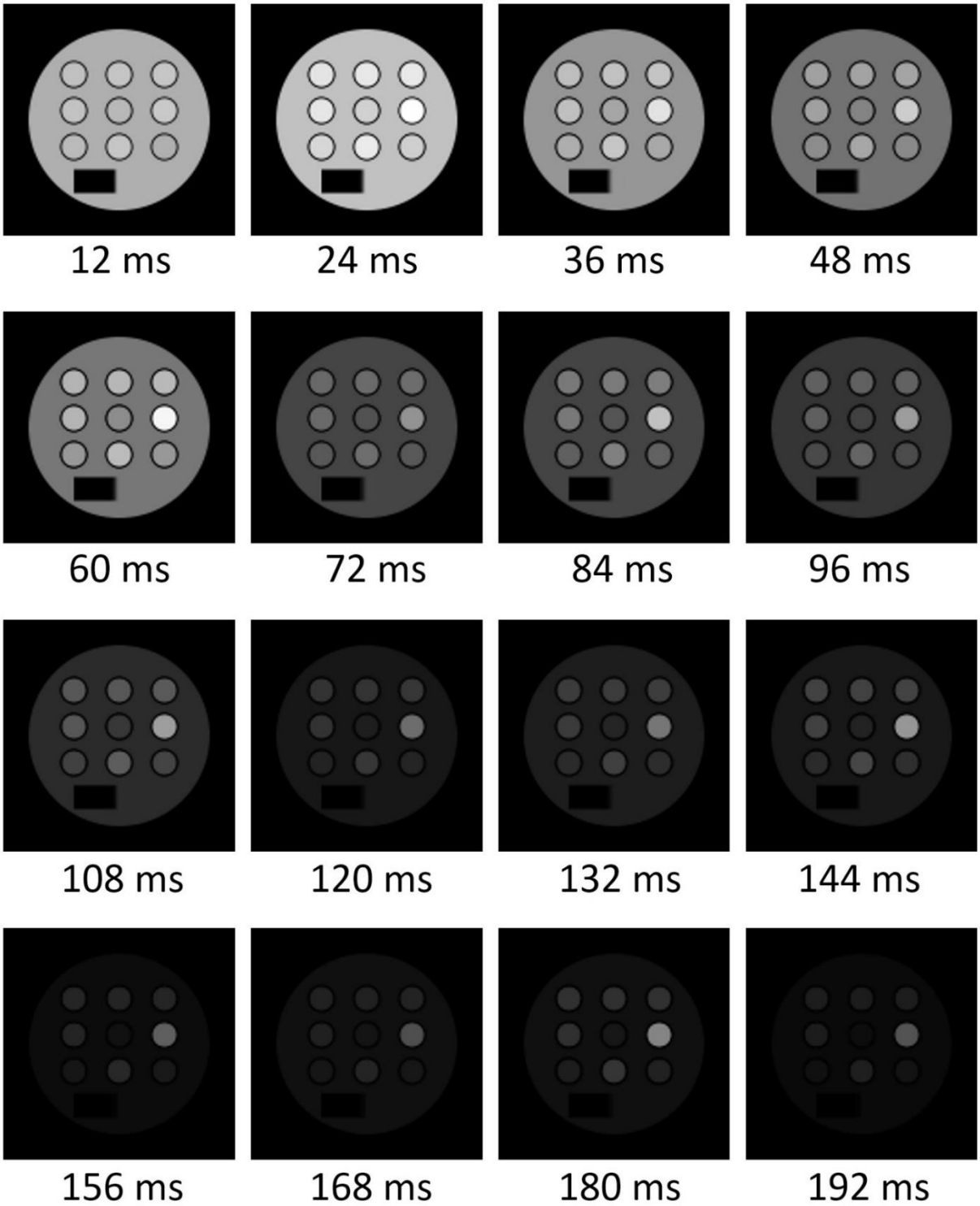


Fig.35 Multiple spin echo images with 128 × 128 pixels (887.9 s)

When reconstructing 128 × 128 pixel images, set 256 → 128 in Fig.33

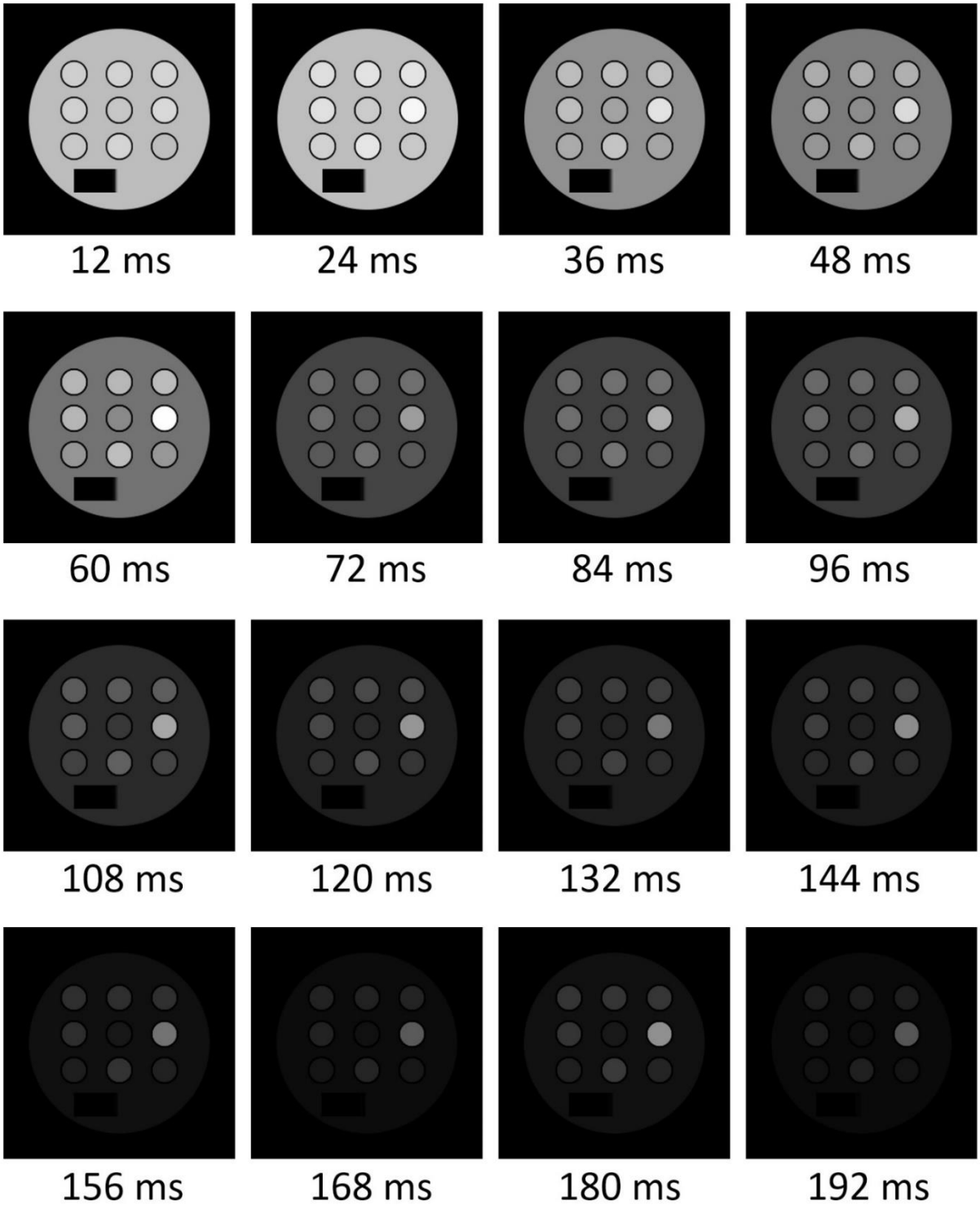


Fig.36 Multiple spin echo images with 256 × 256 pixels (14772.1 s)

4-7 Multislice gradient echo

The image reconstruction method is the same as in the case of multi-echo. However, change the number for height in Fig. 33 from 16 to 24 and perform the same operation.

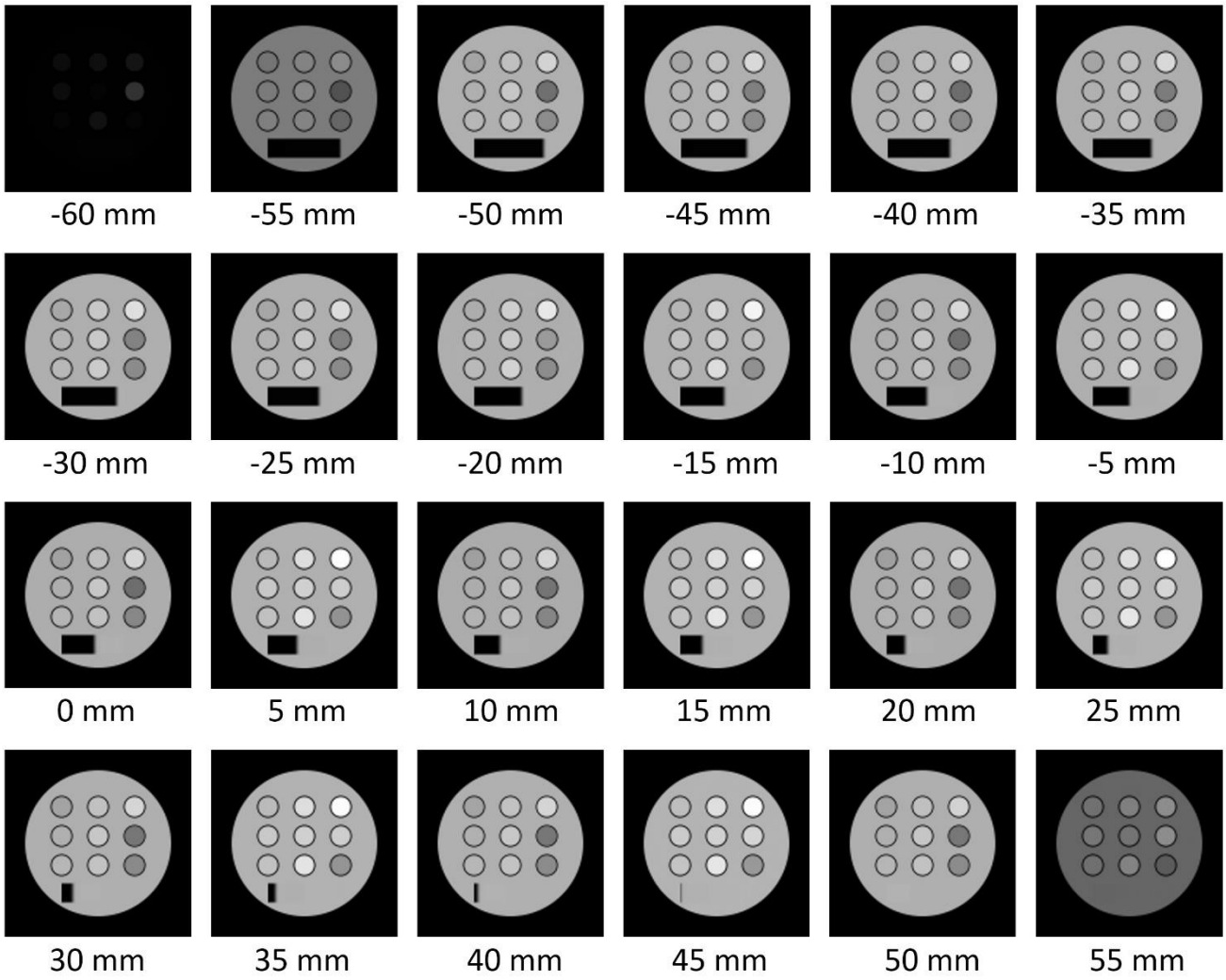


Fig.37 Multislice gradient echo images with 128 × 128 pixels (1505.1 s)

When reconstructing 128 × 128 pixel images, set 256 → 128 in the file input dialog box.

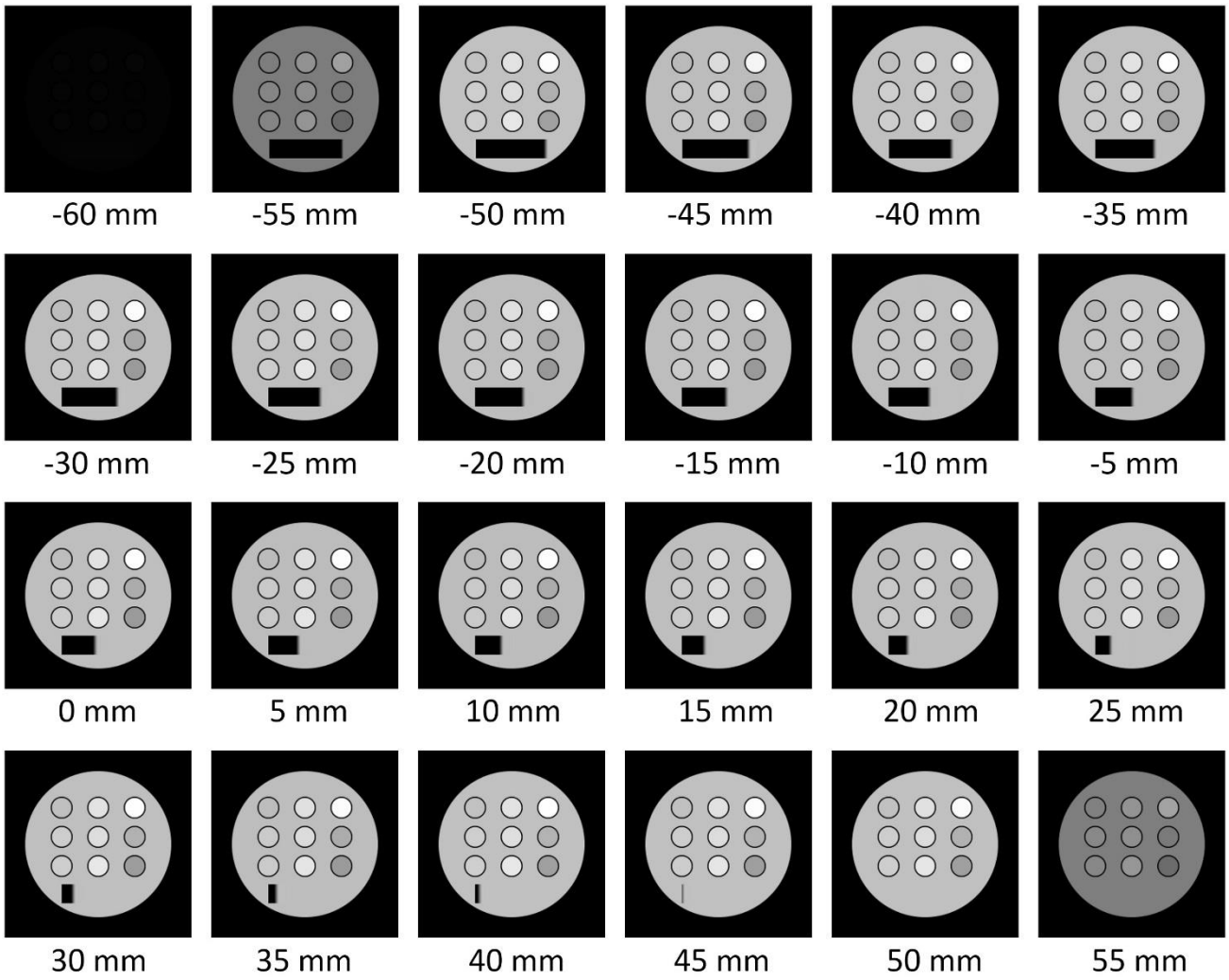


Fig.38 Multislice gradient echo images with 256 × 256 pixels (21002.0 s)

4-8 Multislice spin echo

The image reconstruction method is the same as that for the multi-slice gradient echo.

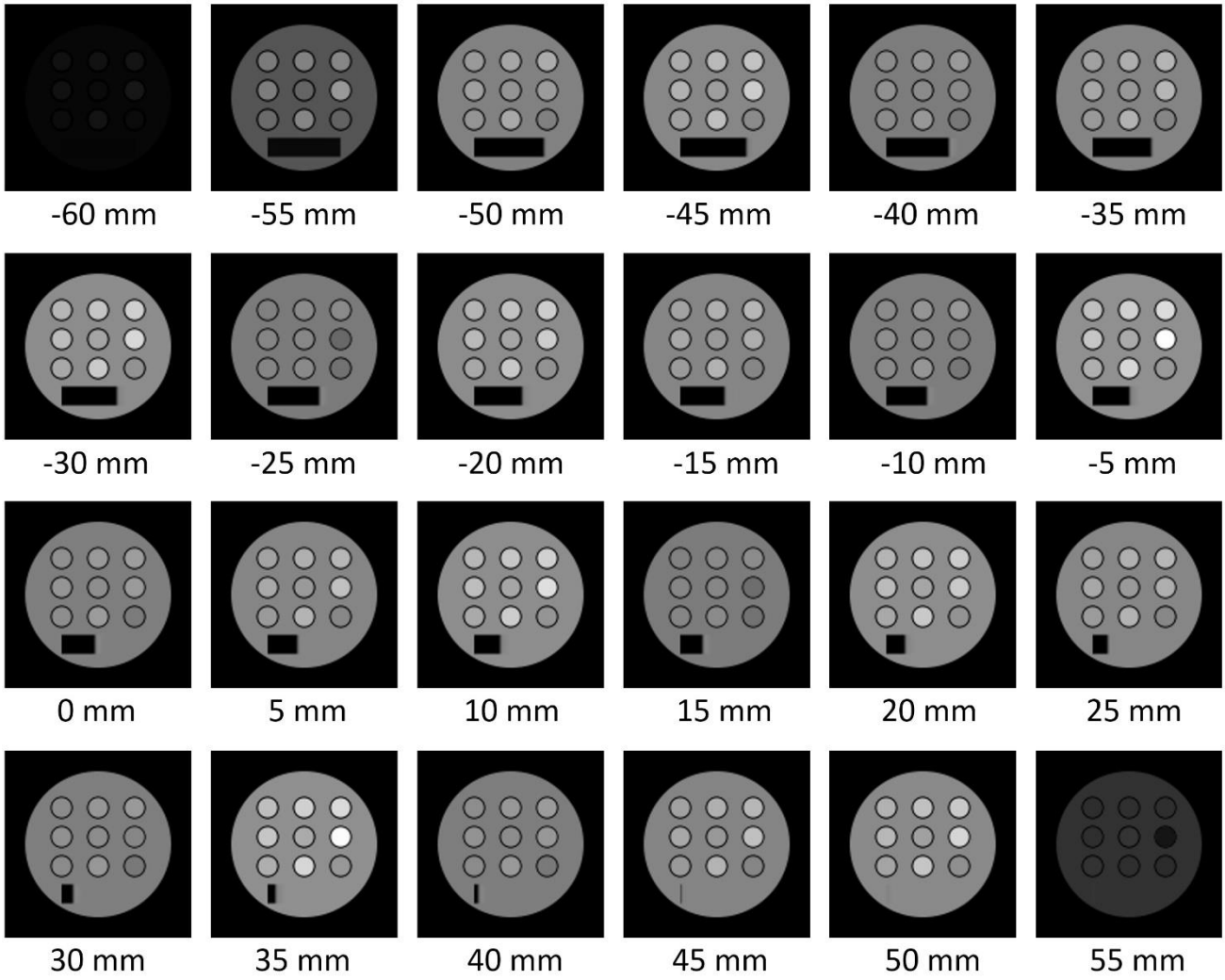


Fig.39 Multislice spin echo images with 128 × 128 pixels (2396.7 s)

When reconstructing 128 × 128 pixel images, set 256 → 128 in the file input dialog box.

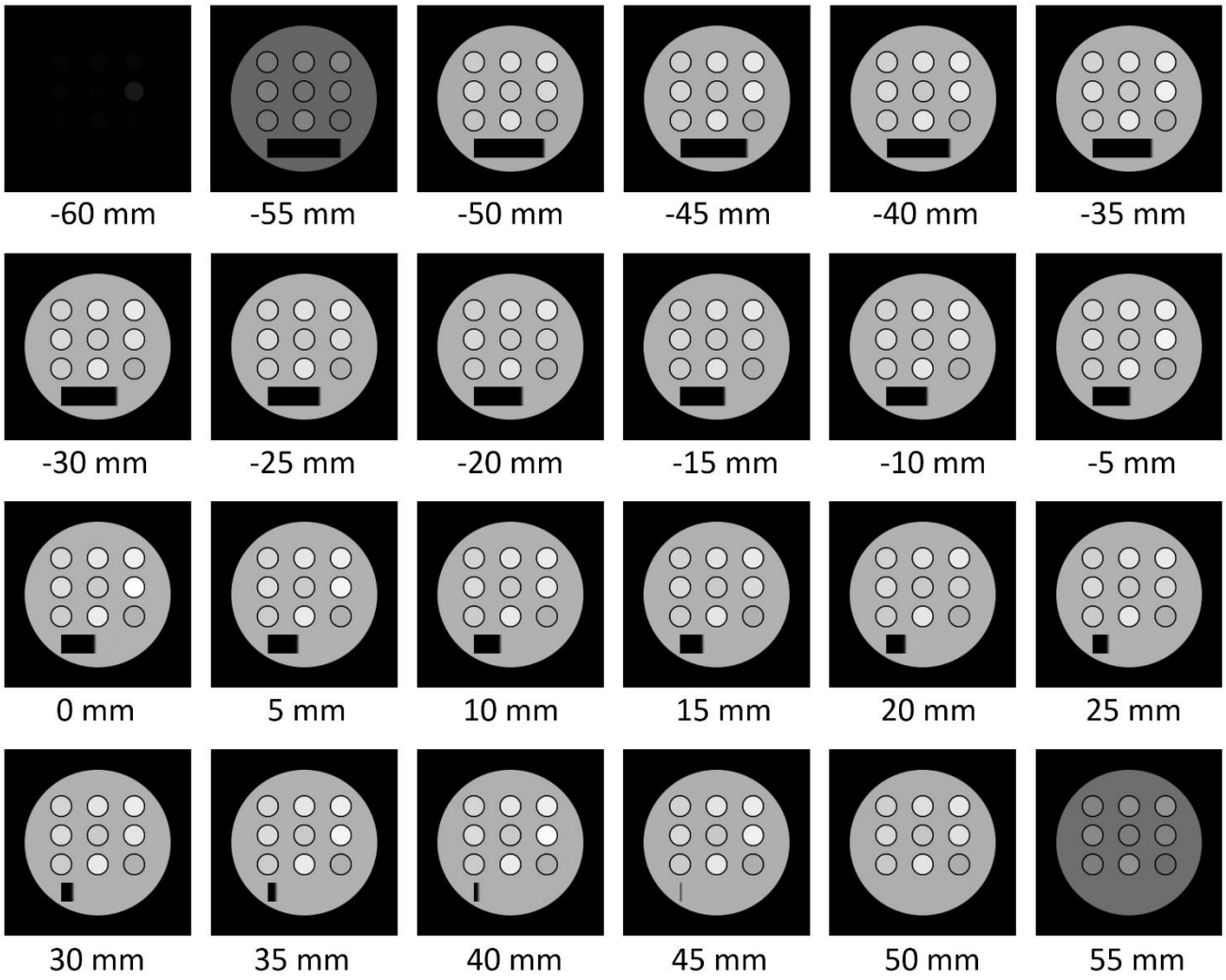


Fig.40 Multislice spin echo images with 256 × 256 pixels (38066.0s)

4-9 3D gradient echo (32 slice)

In the image reconstruction of the signal obtained by the simulation of the 3D gradient echo sequence, set the input dialog box as shown in Fig. 41 and press the OK button to perform reconstruction. FFT is performed in all x, y, and z directions as shown in the configuration box. For reconstruction of an image of 128 × 128 × 32 pixels, 256 → 128 in Fig. 41.

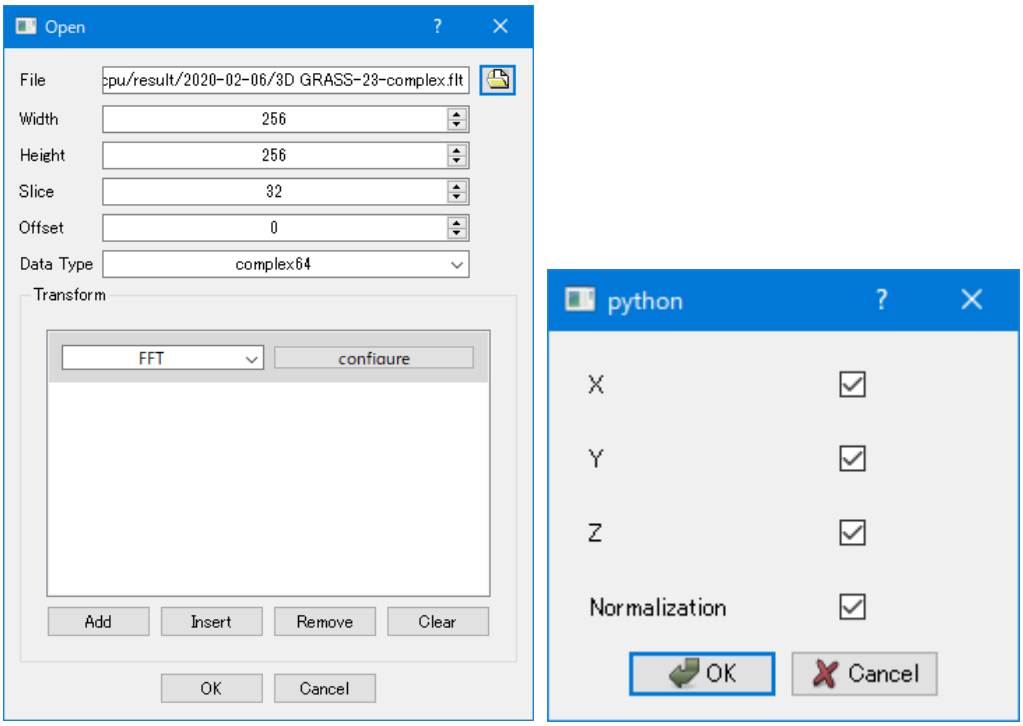


Fig.41 Dialog box for input the 3D gradient echo signal and the configuration box for FFT

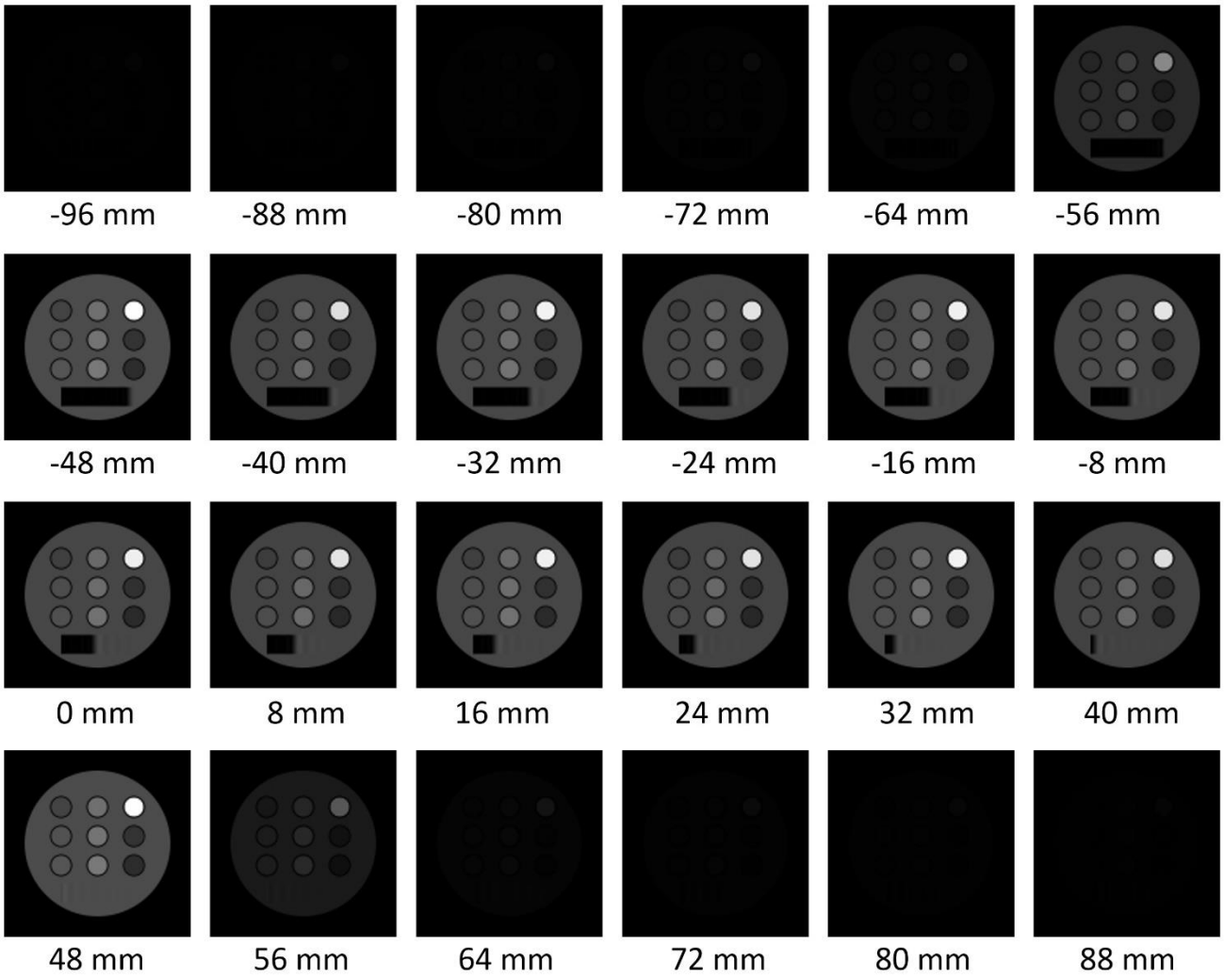


Fig.42 3D gradient echo image with 128 × 128 × 32 voxels (76.1 s)

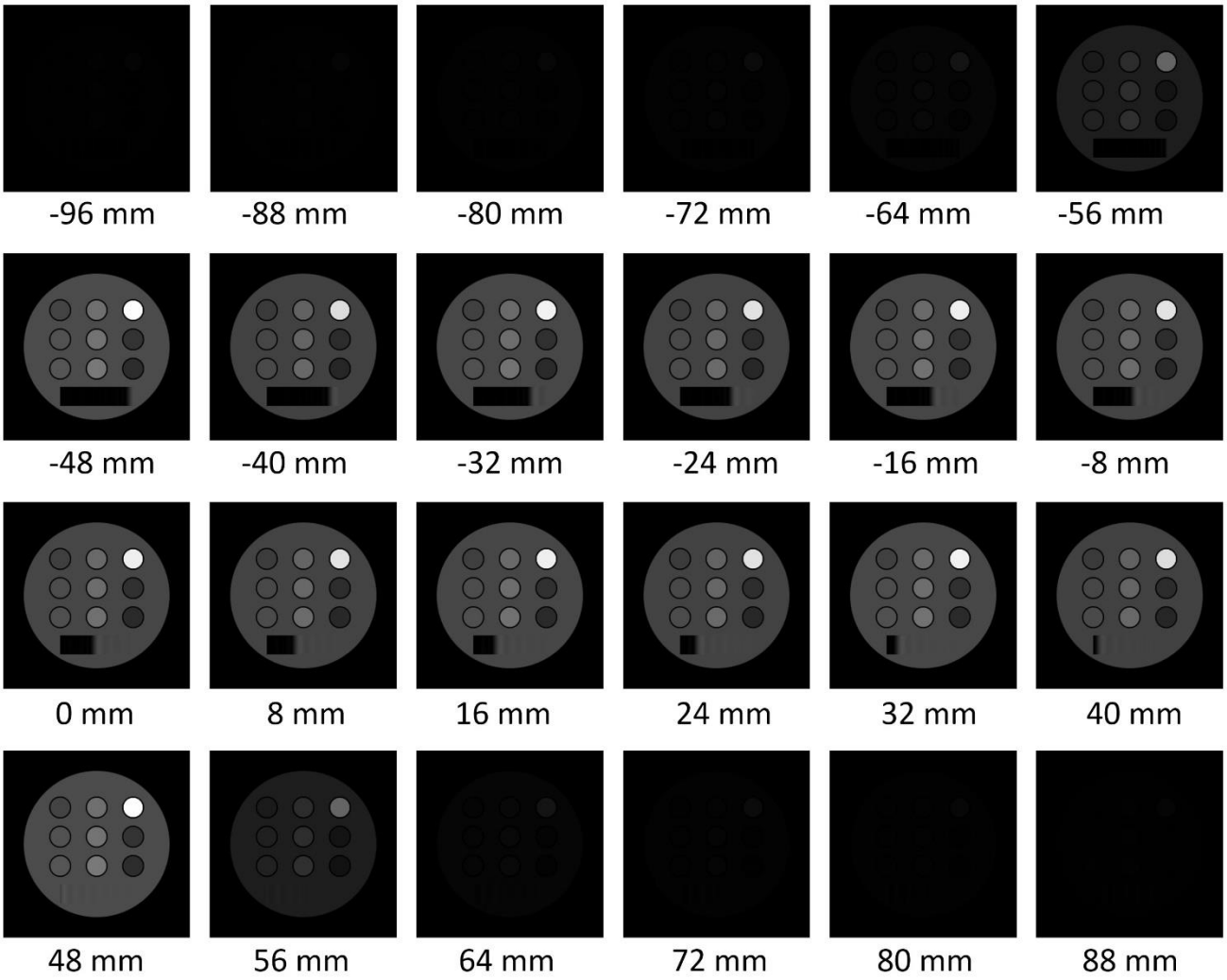


Fig.43 3D gradient echo image with 256 × 256 × 32 voxels (1735.5 s)

4-10 3D gradient echo (128/256 slices)

In the image reconstruction of the signal obtained by the simulation of the 3D gradient echo sequence, change 32 to 128 or 256 in the Slice text box in Fig. 41, and press the OK button. FFT is performed in all x, y, and z directions as shown in the configuration box.

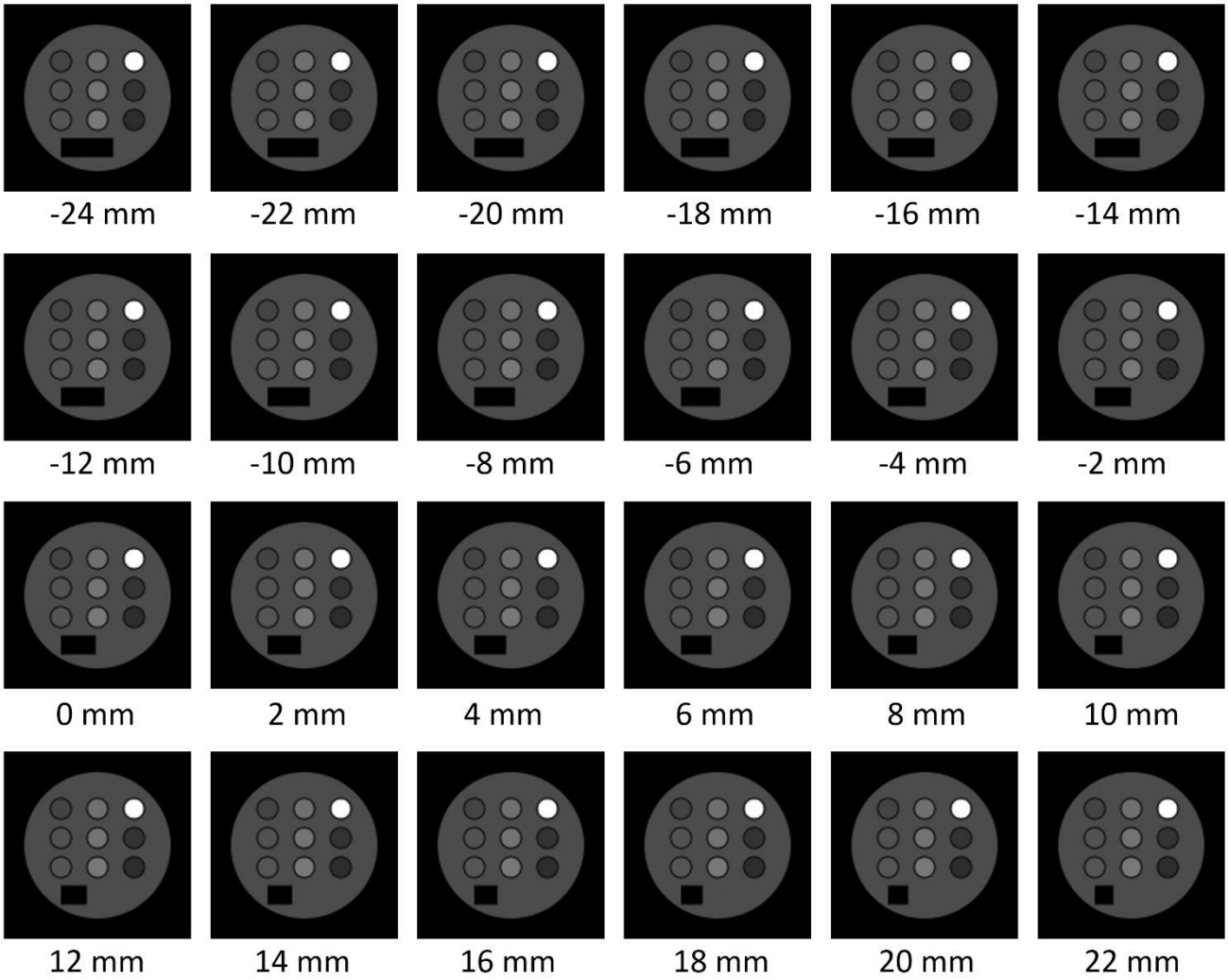


Fig.44 3D gradient echo with 128 × 128 × 128 voxels (300.8 s)

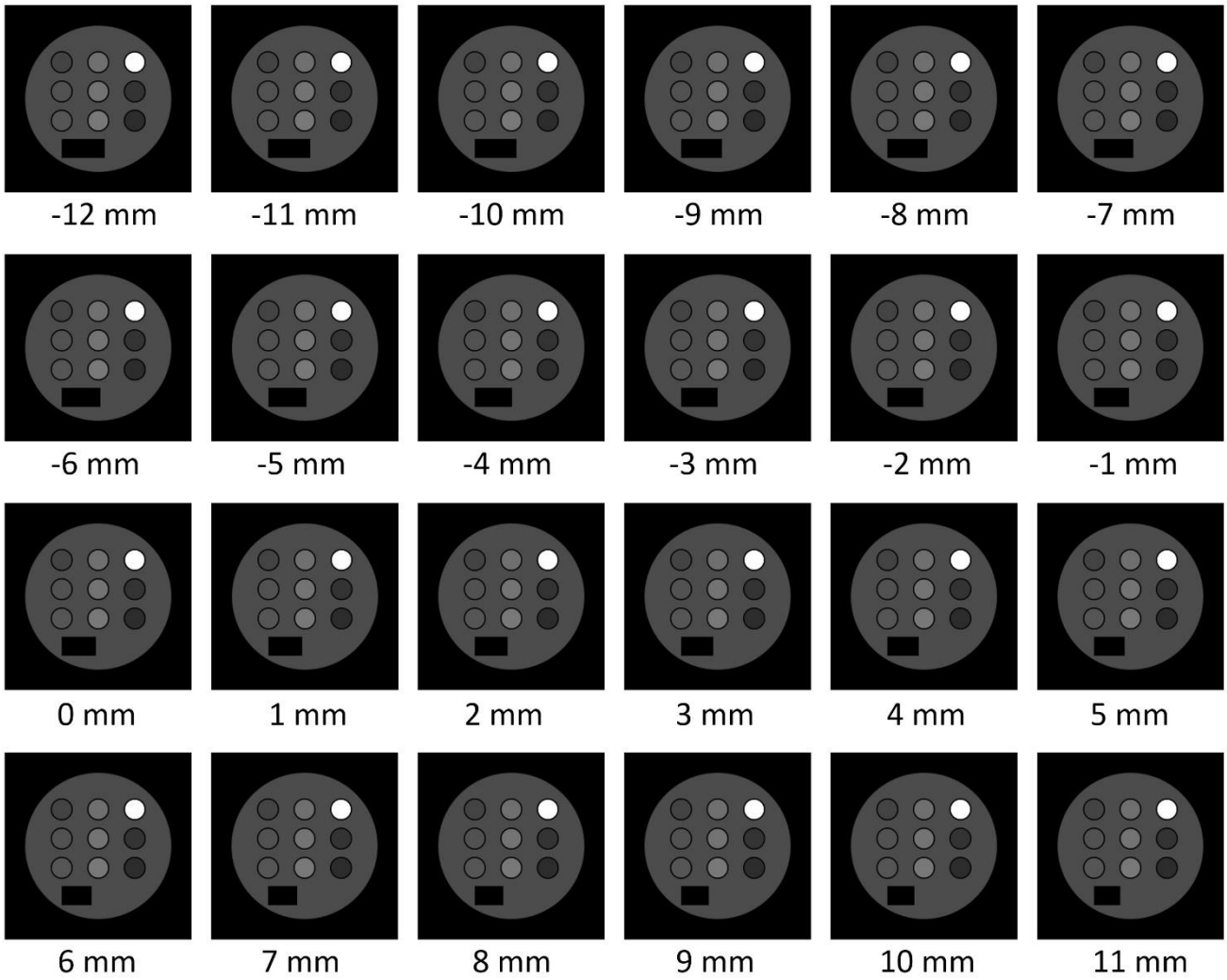


Fig.45 3D gradient echo image with 256 × 256 × 256 voxels (13877.1 s)

4-11 Echo planar imaging

Viewer.bat does not support image reconstruction of echo planar imaging, so use the GUI program. In the images in Fig. 46, the pulse sequence is the same, but the size of the numerical phantom is different: 128^3 and 256^3 voxels.

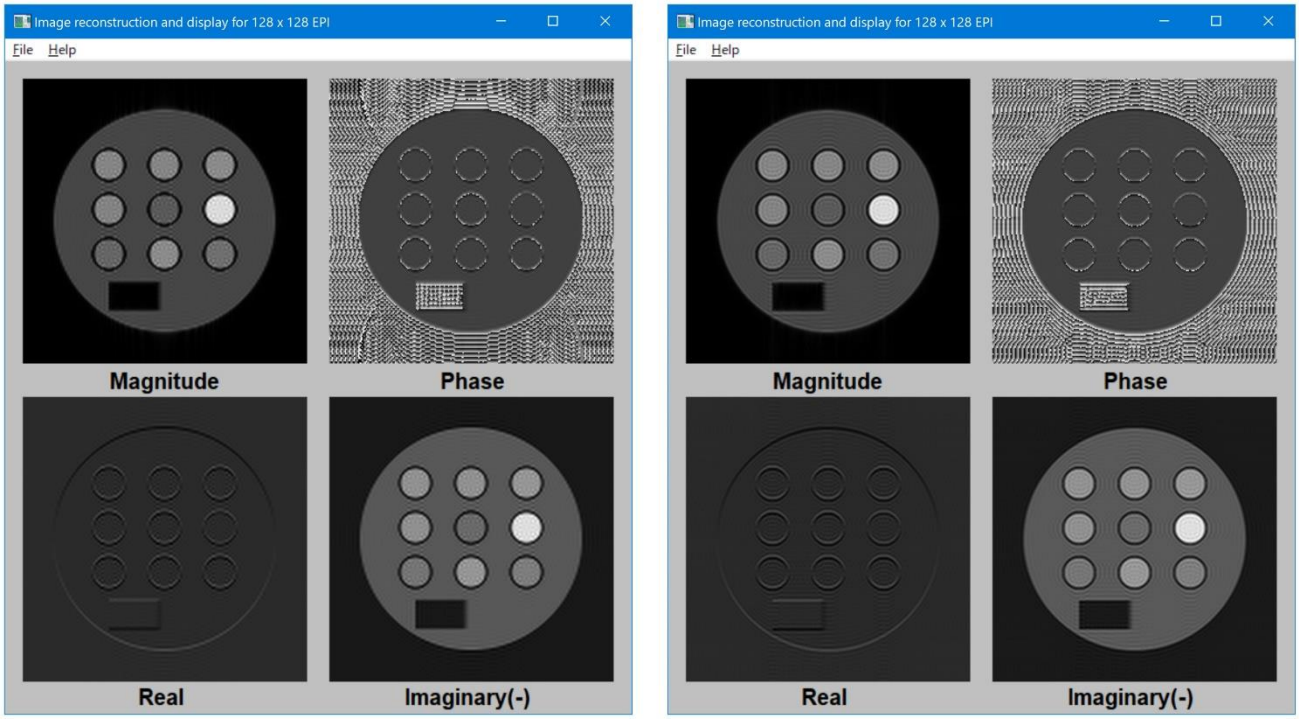


Fig.46 EPI images with a 128^3 voxel numerical phantom and a 256^3 voxel numerical phantom

5. How to speed up the Bloch simulation

In order to speed up the simulation, it is important to select the **number of image matrices** and the **number of subvoxels** in addition to the use of **GPU**.

5.1 Number of image matrix

Regarding the number of image matrices, there are two options: use an image with 128^2 to 128^3 pixels or select an image with 256^2 to 256^3 pixels. Of course, using an image with 64^2 to 64^3 pixels or an image with 512^2 to 512^3 pixels is also possible in the specifications, but it is not a realistic option (may be an option in special cases). It is also possible to use an image matrix size that is not a power of 2, but because the standard size of MR images is 256^2 to 512^2, which is not realistic.

As shown in Fig.18, the processing speed of 128^2 pixel images is about 15 times faster than that of 256^2 pixel images. It is convenient to use a 128^2 pixel image for sequence adjustment, parameter search, etc., and a 256^2 pixel image for final evaluation.

5.2 Number of subvoxels

It is necessary to set the number of subvoxels correctly in order to obtain images without artifacts. In the following cases multiple subvoxels are required.

(1) Magnetic field inhomogeneity is present

In the presence of static magnetic field inhomogeneity, the phase of nuclear magnetization may change significantly within a voxel, and in such cases the signal strength may drop or be lost. In order to correctly describe such a phenomenon, it is necessary to set multiple subvoxels in the voxel. The example shown in Fig.9 is a typical example. In the case of a single slice image, if two subvoxels in the signal readout direction, one in the phase encoding direction, and four in the slice thickness direction are set, artifacts can be eliminated.

(2) To describe the slice profile exactly

Depending on the type of selective excitation pulse used for the slice, when a selective excitation pulse is applied under the presence of the slicing gradient, the nuclear magnetization rotates multiple times in the voxel in the direction perpendicular to the slice plane. To describe the situation correctly, multiple subvoxels are required in that direction. In the pulse sequence we provided, a hamming windowed sinc pulse of $\pm 2\pi$ is used. In this case, it is empirically preferable that the number of subvoxels in the slice direction is 4 or more. When the gradient is applied in the slice thickness direction and repetitive RF pulses are applied in multi-slice imaging, it is necessary to accurately reproduce the slice profile, and thus more subvoxels are required. The most prominent example is the T2-weighted fast spin echo method, which requires 16 to 32 sub-voxels even with 8 echoes. In this case, description of stimulated echoes and higher order echoes using subvoxels are required.

(3) TR << T_2, Echo spacing << T_2 (multiple echo)

In the RF spoiled gradient echo sequence, when the transverse magnetization component generated by the past RF pulse remains in the voxel, a large number of subvoxels are required in the signal readout direction depending on the relationship between T2 and TR.

6.Concluding remarks

Together with the website, this manual was written so that MRI simulation can be performed while using this manual. However, information about BlochSolver may not be sufficient, so please feel free to contact us at the email address below.

info_at_mrisimulations.com (_at_ is @).

I will reply as soon as possible.