A Guide to Three-Dimensional Analytical Magnetic Resonance Imaging Phantom in the Fourier Domain

SECTION 1 INTRODUCTION

Welcome. I am excited that you are interested in this analytical phantom.

In the archived file STBBFolder.zip, you will find 9 items:

1. this guide,
2. the Software agreement,
3. JarFile folder,
4. JavaDoc folder,
5. JavaTest folder,
6. MathematicaTest folder,
7. MatlabTest folder,
8. IDLTest folder, and
9. src folder.

The JarFile folder contains a jar file called STBB.jar.

JavaDoc contains HTML files shows how the classes and methods are named.

JavaTest contains an example of using the 3D analytical phantom in the Java environment.

To facilitate investigators who are not familiar with Java but are comfortable with Mathematica, Matlab or IDL, I have included information on how to call Java classes from these commercial packages.

Finally, src folder contains the source codes written in Java programming language.

Here is a summary of the topics covered in this guide:

> Section 2 introduces the reference on the 3D analytical phantom in the Fourier domain.
>
> Section 3 provides the basics on installing Java and setting up Java environment and run a simple Java application, i.e. SampleTest.java in the JavaTest folder.
>
> Section 4 outlines the basics of JLink in Mathematica and how to call Java from Mathematica.
>
> Section 5 shows how to use Matlab to create Java classes.

Section 6 deals with IDL and shows how to call Java from IDL.

Many colleagues had worked with me in creating the interfaces between Java and other programming languages for another related project, namely, the nonlinear least squares tensor estimations[1]. I have adapted these interfaces to this work. If you need the software on tensor estimations, please send your request to me.

Sincere thanks go to Dr. Joelle E Sarlls, Dr. Evren Özarslan for helpful discussion and for carefully and critically reading the work related to the 3D phantom. I am very grateful to Dr. Peter J. Basser and Dr. Carlo Pierpaoli for their continued support.

Please feel free to contact me if you have any question related to the software. If you have any question related to IDL or Matlab, I may not be the best person to talk to but I will try to help. However, if your questions are related to Mathematica or Java, it is more likely that I will be able to answer them.

Good luck and best wishes.

Cheng Guan Koay
STBB/LIMB/NICHD
National Institutes of Health
Building 13, Room 3W16,

13 South Drive

Bethesda, MD 20892

Tel: (301) 435-9333, Fax: (301) 435-5035

guankoac@mail.nih.gov, cgkoay@uwalumni.com

07/07/2007

[1] Koay CG et al. A Unifying Theoretical and Algorithmic Framework for Least Squares Methods of Estimation in Diffusion Tensor Imaging. J Magn Reson 2006; 182: 115-125.

SECTION 2: REFERENCE

This is a quick guide to users interested in using the 3D analytical phantom in the Fourier domain (k-space) or in the image domain.

Readers may find the following article helpful:

(1) Koay CG, Sarlls JE, Özarslan E. Three Dimensional Analytical Magnetic Resonance Imaging Phantom in the Fourier Domain. Magn Reson Med. (Early view)

If you find this software useful in your research, please pass the message by word of mouth or citation so that other researchers may benefit from this software.

You can obtain the above articles on the STBB website:
http://dir2.nichd.nih.gov/nichd/stbb/publications.html

SECTION 3: Installing and Running Java

There are several ways to use and develop Java applications. We can use the basic text editors (i.e. vi, emacs, wordpad, notepad) to write java programs and run java applications using command lines or we can use Java Interactive Development Environment (IDE) such as Netbeans to write our codes, see www.netbeans.org. Personally, I find Netbeans' IDE to be very useful. For example, the IDE will do text completion. And, it is great!

If you plan to develop applications in Java Standard Edition (Java SE), you may want to use Java Development Kit (JDK) rather than Java Runtime Environment (JRE). You can download JDK or JRE from (last checked on 08/28/06):
http://java.sun.com/javase/downloads/index.jsp
Look for "JDK 5.0 Update 7 with Netbeans 5.0", "JDK 5.0 Update 8" or "Java Runtime Environment (JRE) Update 8" to download. You can find installation instructions on the same page.

Assume now that you have successfully installed the software. You can check your Java version by typing on the terminal:
"java –version"
This command works in both the Linux and Windows systems.
To be sure, we will assume you know where you stored the JDK, say "c:\java\j2sdk5.0\" (Linux: /home/java/j2sdk5.0/) so that we know that full path to java.exe, i.e. "c:\java\j2sdk5.0\bin\java.exe" (Linux: /home/java/j2sdk5.0/bin/java).

For simplicity, we will outline how to run the SampleTest.java using command line approach.

Step1: Go to the directory where SampleTest.java is located. As an example we shall use this fictitious directory path: "c:\STBBFolder\JavaTest\" for Windows and "/home/STBBFolder/JavaTest/" for Linux.

Step2: To compile the Java application, run the following command:
Windows:
"c:\java\j2sdk5.0\bin\javac –classpath .;c:\STBBFolder\JarFile\STBB.jar SampleTest.java"
Linux:
"/home/java/j2sdk5.0/bin/javac –classpath .:/home/STBBFolder/STBB.jar SampleTest.java"

These commands should return SampleTest.class.

Note the difference in notation when one specifies the classpath. Windows uses semi-colon and Linux uses colon.

Step2: To compile the Java application, run the following command:
Windows:
“c:\java\j2sdk5.0\bin\javac –classpath .;c:\STBBFolder\JarFile\STBB.jar SampleTest.java”
Linux:
“/home/java/j2sdk5.0/bin/javac –classpath .:/home/STBBFolder/STBB.jar SampleTest.java”

Step 3: Run the application by entering the following command:
Windows:
“c:\java\j2sdk5.0\bin\java –classpath .;c:\STBBFolder\JarFile\STBB.jar SampleTest”
Linux:
“/home/java/j2sdk5.0/bin/java –classpath .:/home/STBBFolder/STBB.jar SampleTest”

If you manage to run this simple application, you should see on the screen the following results:

You should see the following results printed on the terminal:

image domain signal at (-0.25,0.15,0.05) is 1.2
Fourier domain signal at (0.25,0.15,0.05) is 2.4834061312960225-0.00838733761651407I
image domain signals at {{-0.25,0.15,0.05},{0.5,0.35,0.5}} are { 1.2, 2.0};

Fourier domain signals at {{-0.25,0.15,0.05},{0.75,0.5,0.5}} are { 2.48331915292207-0.004916488124360764I, -0.22797836078897474-0.012638899072246745I};

NOTE: FourierDomainSignal returns an array of two elements the first and the second elements are the real part and the imaginary part of the complex value signal, respectively.

Hope this is enough to get you started. You can learn more by looking up the Java webpage: http://java.sun.com.

SECTION 4: Mathematica

NOTE: FourierDomainSignal returns an array of two elements the first and the second elements are the real part and the imaginary part of the complex value signal, respectively.

If you are using Mathematica 5.0, you should already have JLink installed. JLink is the interface for basic communication between Java and Mathematica.

SampleTest.nb is the notebook where the same example as in SampleTest.java is carried within the Mathematica notebook. It shows examples of initiating JLink, creating objects and calling methods associated with the objects. The explanations are provided in the notebook and SampleTest.nb is in the MathematicaTest folder.

If you manage to run SampleTest.nb, you should get the same results as in Section 3.

SECTION 5: Matlab

NOTE: FourierDomainSignal returns an array of two elements the first and the second elements are the real part and the imaginary part of the complex value signal, respectively.

Calling Java from Matlab is quite easy but you need version 7.0 and above. In the MatlabTest folder, you will find SampleTest.m. It contains all the necessary information to run the test program.

If you manage to run SampleTest.m, you should get the same results as in Section 3.

SECTION 6: IDL

NOTE: FourierDomainSignal returns an array of two elements the first and the second elements are the real part and the imaginary part of the complex value signal, respectively.

This is the most complicated one. You need to set up your bash shell environment or csh shell depending on your preference. I have not run the SampleTest.pro on Windows. Therefore, the focus is on the Linux platform.

All files related to IDL are located in the IDLTest folder.

Step 1:
If you are using bash shell:
Try this: Copy those lines beginning with the word “export” in bashrc.txt to your “.bashrc” file in your home directory and makes changes to the directory names in those lines where necessary.

If you are using csh shell:
Try this: Copy those lines beginning with the word “setenv” in cshrc.txt to your “.cshrc” file in your home directory and make changes to the directory names in those lines where necessary.

Step 2: Change the directory name in the idl.cfg file.

Step 3: Depending on which “shell” you like to use, type in “csh” or “bash” to create new shell environment. Then, run your idl program.

Step 4: Run SampleTest.pro

You should get the same results as in Section 3.

FINAL NOTE: FourierDomainSignal or ImageDomainSignal can be overloaded with array of arrays: namely:

```
FourierDomainSignal(
                    {{ kx1, ky1, kz1 },
                     { kx2, ky2, kz2},
                         .     .     .   ,
                     { kxN, kyN, kzN}
                    )
```

and

```
ImageDomainSignal(
                    {{ x1, y1, z1 },
                     { x2, y2, z2},
                         .     .     .   ,
                     { xN, yN, zN}
                    )
```

Note that the braces are the convention used in Java and Mathematica. For Matlab or IDL, the convention is different.

Matlab:

```
[ x1 y1 z1; …
  x2 y2 z2; …
   .   .   . ;…
  xN yN zN];
```

IDL:

```
p=[ [x1, y1, z1],$
  [x2, y2, z2],$
   .   .   .    ,$
  [xN, yN, zN]]
```

ImageDomainSignal(transpose(p))

IDL is atypical. It defines row and column differently from the mathematical convention. So, you need to transpose the matrix p.