

How to Install JSindo

Kiyoshi Yagi
kiyoshi.yagi@riken.jp

Theoretical Molecular Science Laboratory
RIKEN Pioneering Research Cluster

2018/04

1. Install Java

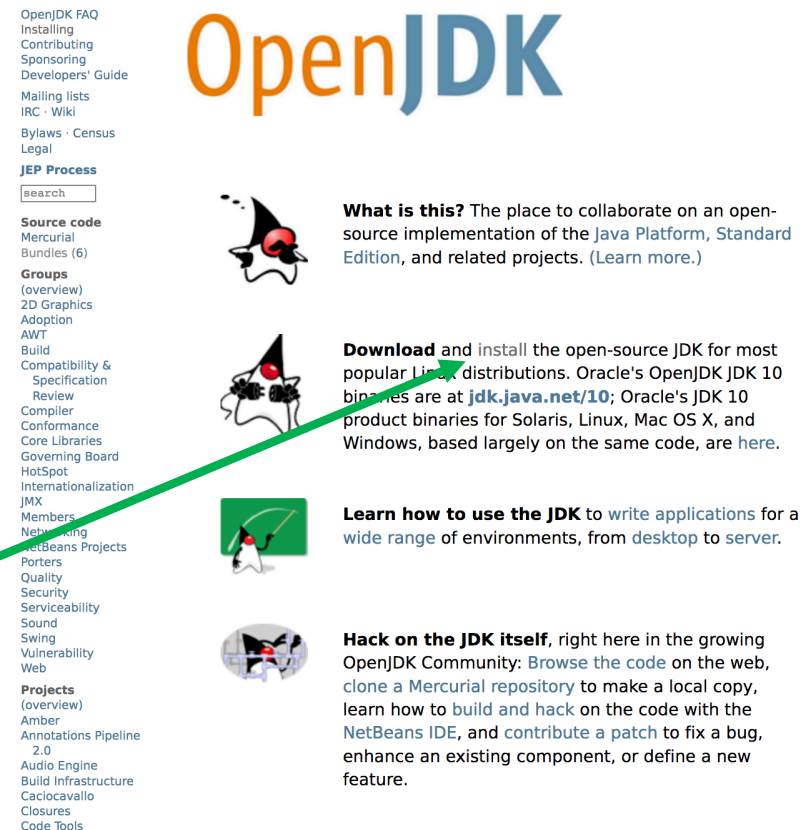
Install Java of oracle (left), Open JDK (right), or any other.

<https://www.java.com/en/>



Click here and follow the instructions.

<http://openjdk.java.net/>



Note: Although Java 9 and 10 have been released since 2017/09 and 2018/03, respectively, Java 8 (1.8.x) is still recommended for JSindo, because it uses Java3D which may or may not be compatible with the newer Java.

2. Setting up Java3D

JSindo uses Java3D for visualization. A stable version, 1.6.0, is available from JogAmp. Goto <http://jogamp.org>



Main Page

Welcome to the [JogAmp](#) wiki. It documents JOGL, JOCL and JOAL, the cross-platform bindings to the OpenGL, OpenCL, and OpenAL APIs.

click here

Getting Started

- [Downloading and installing](#)
- [Versioning and Releases](#)
- [Setting up a JogAmp project in your favorite IDE](#)
- [Source Code Repositories](#)
- [Tracking Reports](#)
- [Build and Test Server](#)

Community

- [Contribute to JogAmp](#)
- [Build JogAmp](#)
- [Maintainer and Contacts](#)
- [Report a bug](#)
 - [Bugzilla](#)
- [Ask a question in the forum](#)
- [JogAmp IRC](#)

Downloading and installing JOGL

Before you can build a project that uses JOGL [in your IDE](#) or [on the command line](#), you'll need to download and install the JOGL JAR files and native JARs or native library files (.dll/.so/.jnilib files).

You have a choice of JOGL versions to download. The [latest stable version](#) is the safest, but lags behind in features. The [latest automatic build](#) contains all checked-in code, but may be failing some tests.

Contents [\[hide\]](#)

- 1 [Downloading the latest stable version](#)
 - 1.1 [Using the 7z jogamp-all-platforms archive](#)
- 2 [Downloading the latest aggregated autobuild](#)
- 3 [Downloading the latest automatic build](#)
 - 3.1 [Native JARs vs. native library files](#)
 - 3.2 [Unzipping the files](#)
- 4 [Maven](#)
- 5 [More information](#)

Downloading the latest stable version

Go to [this page](#) and download the all-in-one 7z archive file:

[jogamp-all-platforms.7z](#)

click here and download
jogamp-all-platforms.7z

Go back to the Main page and scroll down

Main Page

Welcome to the [JogAmp](#) wiki. It documents JOGL, JOCL and JOAL, the cross-platform bindings to the OpenGL, OpenCL, and OpenAL APIs.

⋮ ↓ Scroll down

Related Projects

Java3D

- Overview
- **Downloading and installing**
- Tutorial
- API Documentation
- FAQ

click here

Ji Gong

- Overview
- Motivation
- FAQ

Page **Discussion**

Downloading and installing Java3D

Downloading the latest stable version

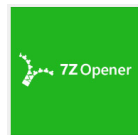
Go to [this page](#) and download the 7z archive file:

[jogamp-java3d.7z](#)

Do the same for JogAmp as it is indicated [here](#).

click here and download
jogamp-java3d.7z

Unarchive the two files you've just downloaded. 7z files can be unarchived using, for example, "7Z Opener" (for Win) or "The Unarchiver" (for Mac), which are freely available.



7Z Opener
Tiny Opener



The Unarchiver
MacPaw Inc.

You will find jar files in jogamp-all-platforms/jar and in jogamp-java3d. The following jar files are needed for JSindo:

```
jogamp-all-platforms/jar/  
  gluegen-rt.jar  
  gluegen.jar  
  gluegen-rt-natives-XXX.jar  
  jogl-all.jar  
  jogl-all-natives-XXX.jar
```

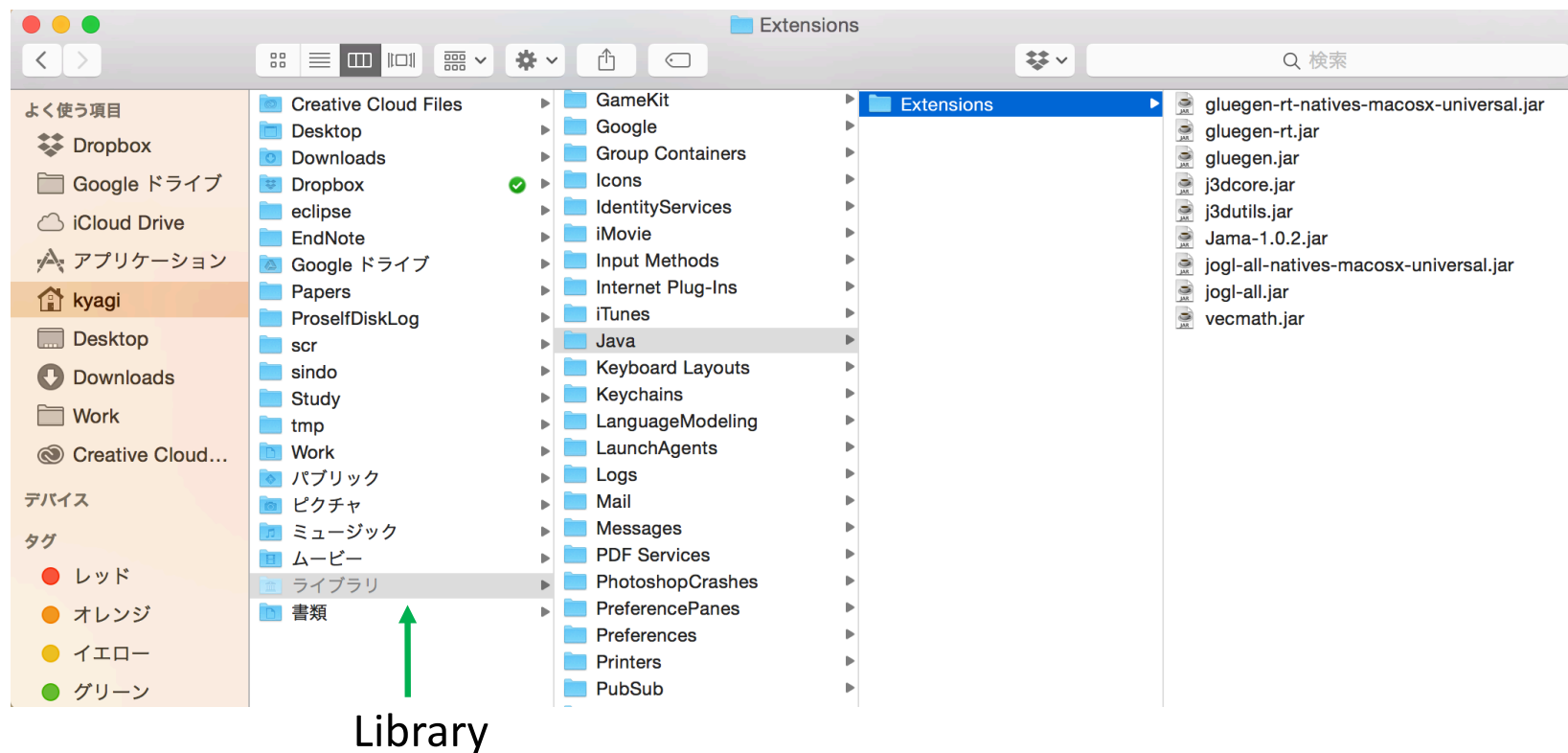
```
jogamp-java3d/  
  j3dcore.jar  
  j3dutils.jar  
  vecmath.jar
```

where **XXX** indicates your OS and CPU.

We will next describe how to install these files in each platform.

MacOSX: **XXX** = macosx-universal.

The easiest install for MacOSX is to copy the jar files to an extension folder. In MacOSX, it is set to ~/Library/Java/Extensions. Click Go menu of finder with option key pressed (~/Library is hidden) and choose Library. Create the folder if you don't have it yet, then copy the jar files in this folder.



Windows: **XXX** = windows-amd64 or windows-i586

i586 is for 32-bit and amd64 is for 64-bit. Let's first check if your Java is 32- or 64-bit. In the DOS prompt, type "java -version" and you will see a message like this:

```
>java -version
java version "1.8.0_45"
Java(TM) SE Runtime Environment (build 1.8.0_45-b14)
Java HotSpot(TM) 64-Bit Server VM (build 25.45-b02, mixed mode)
```

This is an example of 64-bit. If "64-Bit" is absent, then it's 32-bit. (It doesn't explicitly state "32-Bit", unfortunately.)

Now, we will copy the jar files to an extension folder, which are located in

32-bit	c:\Program Files(x86)\Java\jre1.x.x_xxx\lib\ext
64-bit	c:\Program Files\Java\jre1.x.x_xxx\lib\ext

Copy the following jar files in this folder,

```
gluegen-rt.jar
gluegen.jar
gluegen-rt-natives-windows-xxx.jar
jogl-all.jar
jogl-all-natives-windows-xxx.jar
```

```
j3dcore.jar
j3dutils.jar
vecmath.jar
```

LINUX: **XXX** = linux-amd64 or linux-i586

(* CPU = armv6 and armv6hf are for ARM cpus, which are used for phones.)

i586 is for 32-bit and amd64 is for 64-bit. Let's first check if your Java is 32- or 64-bit. In the terminal, type "java -version" and you will see a message like this:

```
>java -version
openjdk version "1.8.0_121"
OpenJDK Runtime Environment (build 1.8.0_121-b13)
OpenJDK 64-Bit Server VM (build 25.121-b13, mixed mode)
```

This is an example of 64-bit. If "64-Bit" is absent, then it's 32-bit.

Now, we add the jar files to an environment variable CLASSPATH. If you are in bash,

```
JOGAMP_JAR=${HOME}/lib/jogamp-all-platforms/jar
export CLASSPATH=${CLASSPATH}:${JOGAMP_JAR}/gluegen.jar
export CLASSPATH=${CLASSPATH}:${JOGAMP_JAR}/gluegen-rt.jar
export CLASSPATH=${CLASSPATH}:${JOGAMP_JAR}/jogl-all.jar
export CLASSPATH=${CLASSPATH}:${JOGAMP_JAR}/gluegen-rt-natives-linux-xxx.jar
export CLASSPATH=${CLASSPATH}:${JOGAMP_JAR}/jogl-all-natives-linux-xxx.jar

J3D_JAR=${HOME}/lib/jogamp-java3d
export CLASSPATH=${CLASSPATH}:${J3D_JAR}/j3dcore.jar
export CLASSPATH=${CLASSPATH}:${J3D_JAR}/j3dutils.jar
export CLASSPATH=${CLASSPATH}:${J3D_JAR}/vecmath.jar
```

where **xxx** = amd64 or i586 for 64-bit or 32-bit, respectively. It would be a good idea to add these lines in your .bashrc, so that they take effect upon login.

3. Download JAMA

JAMA is a linear algebra library for JAVA. We use it for matrix multiplications, diagonalization, and so on. It can be downloaded from,

<https://math.nist.gov/javanumerics/jama/>

JAMA : A Java Matrix Package

[\[Background \]](#) [\[The Package \]](#) [\[Request for Comments \]](#) [\[Authors \]](#) [\[Related Links & Libraries \]](#)

Background

JAMA is a basic linear algebra package for Java. It provides user-level classes for constructing and manipulating real, dense matrices. It is meant to provide sufficient functionality for routine problems, packaged in a way that is natural and understandable to non-experts. It is intended to serve as *the* standard matrix class for Java, and will be proposed as such to the [Java Grande Forum](#) and then to [Sun](#). A straightforward public-domain reference implementation has been developed by the [MathWorks](#) and [NIST](#) as a strawman for such a class. We are releasing this version in order to obtain public comment. There is no guarantee that future versions of JAMA will be compatible with this one.

⋮ ↓ Scroll down

The Package

Version 1.0.3 (November 9, 2012)

- [Documentation](#)
- [Example](#)
- Source [[Jama-1.0.3.zip](#)] [[Jama-1.0.3.tar.gz](#)]
- Jar file [[Jama-1.0.3.jar](#)]
- [ChangeLog](#)

→ click here and download a jarfile.

Then, copy the jarfile to the extension folder or add to CLASSPATH as before.

4. Test JSindo

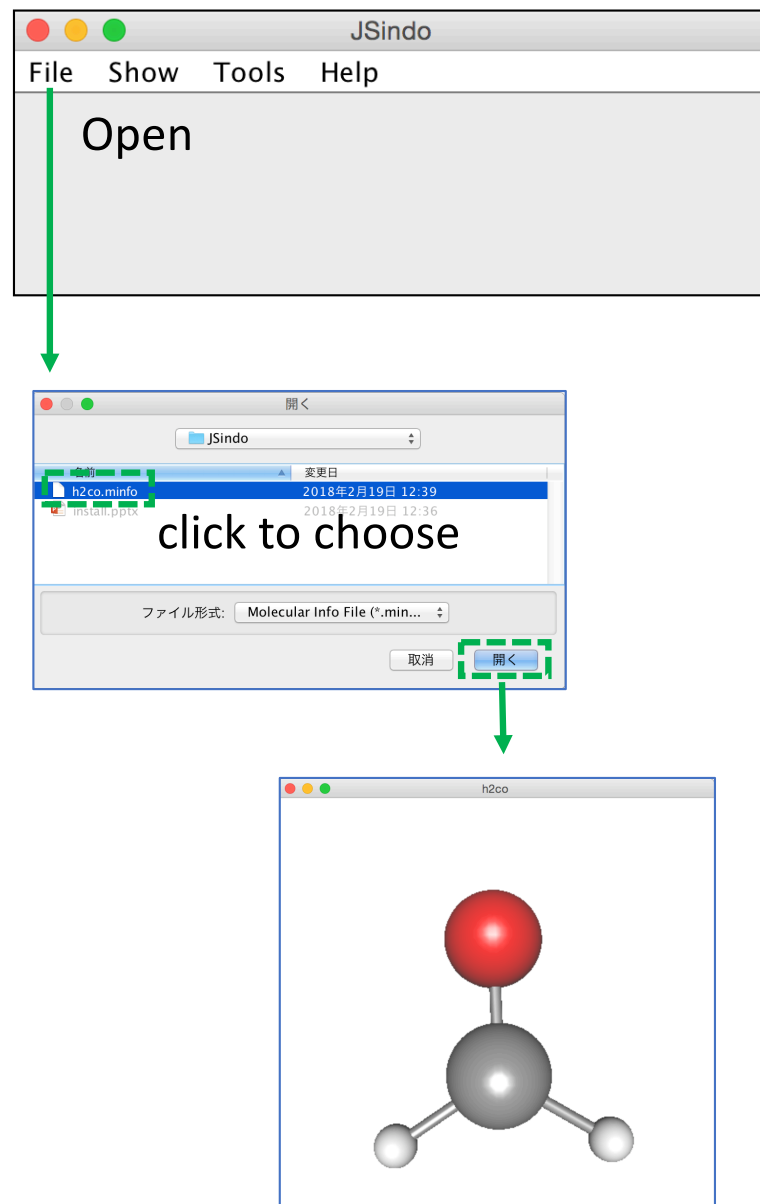
Now, double click JSindo-4.0_xxxxxx.jar. You should see a control panel of JSindo.

If you don't see the panel, review the installation of Java.

Let's open "sample/h2co.minfo", which comes with this document. It contains data of formaldehyde.

Click, File -> Open, choose "h2co.minfo", and click Open. If you see formaldehyde, you're done with the first step!

If this step fails, it is highly likely that Java3D has a problem. Double check if the jarfiles are set correctly, that is, copied to the extension folder or set to CLASSPATH.



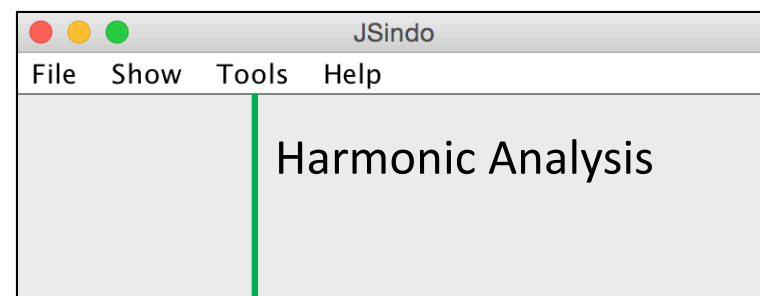
Finally, goto Tools -> Harmonic Analysis. This should create a panel of “Normal modes”.

If you don't see this panel, JAMA isn't working. Check if the jarfile of JAMA is set correctly.

If the panel appears, you're all set! Congradulations!

Check on “show vibrational coordinates”, and choose a mode you want to see. Vibrational motion will be indicated by arrows. You can “Invert the arrows” by a check box, and change the magnitude using a slider.

Thanks for using JSindo!
Enjoy!



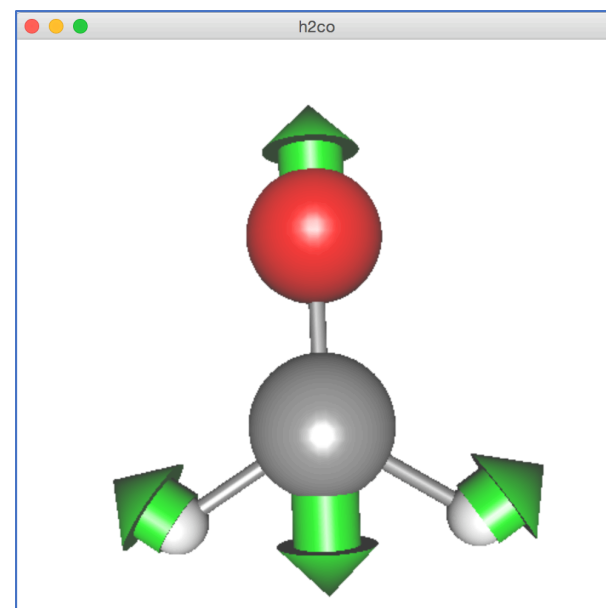
Normal modes (h2co)

Mode	Frequency (cm...	Reduced Mass (...)	Intensity (km m...
1	1196.9147	1.3615	7.0342
2	1266.7685	1.3335	9.3885
3	1540.1545	1.1550	10.7003
4	1752.9374	5.7700	67.7530
5	2973.6886	1.0439	66.6832
6	3047.6560	1.1221	88.4298

☐ Show vibrational coordinates.

☐ Invert the arrows.

Slider: []



FAQ

1. I want to use JSindo from a command line.

Use the following command:

```
>java -cp /path/to/JSindo-4.0_xxxxxx.jar JSindo
```

2. I cannot or don't want to copy jarfiles into an extension folder.

Specify all jarfiles using ":" as a separator,

```
>java -cp JSindo-4.0_xxxxxx.jar:Jama-1.0.3.jar:... JSindo
```

3. Is there an alternative to "-cp"? The command is too long.

The environment variable, CLASSPATH, does the same thing. Set CLASSPATH for all jarfiles,

```
> export CLASSPATH=${CLASSPATH}:/path/to/JSindo-4.0_xxxxxx.jar  
> export CLASSPATH=${CLASSPATH}:/path/to/Jama-1.0.3.jar  
...
```

Then, you can start simply by,

```
> java JSindo
```

I save the setting to ~/.bashrc, so that the CLASSPATH is configured upon login.