

Getting Started with FA $\mu$ ST Toolbox  
"Flexible Approximate Multi-Layer Sparse Transforms"



Adrien Leman; Nicolas Bellot

November 3, 2016

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Installation on Unix platform</b>	<b>5</b>
2.1	Required components . . . . .	6
2.2	Download Faust Package & launch terminal . . . . .	7
2.3	Basic Build & Installation using Makefile . . . . .	7
2.3.1	Install with administrator privilege . . . . .	7
2.3.2	Install without administrator privilege . . . . .	7
2.4	Basic Build & Install using Code Block IDE . . . . .	7
2.4.1	Install with administrator privilege . . . . .	8
2.4.2	Install without administrator privilege . . . . .	8
2.5	Basic Build & Install using Xcode IDE (for MAC OS) . . . . .	8
2.5.1	Install with administrator privilege . . . . .	9
2.5.2	Install without administrator privilege . . . . .	9
2.6	Custom - Advanced Installation . . . . .	10
<b>3</b>	<b>Installation on Windows platform</b>	<b>12</b>
3.1	Required components . . . . .	13
3.2	Download Faust Package . . . . .	14
3.3	Install using GCC-MinGW compiler . . . . .	14
3.3.1	Install GCC from MinGW . . . . .	14
3.3.2	Basic Build & Install using the command prompt . . . . .	15
3.3.3	Basic build & Install using Code::Blocks IDE . . . . .	16
3.4	Install using Microsoft Visual compiler . . . . .	17
3.4.1	Install Microsoft Visual compiler . . . . .	17
3.4.2	Basic Build & Install using Visual Studio IDE . . . . .	17
3.4.3	Basic Build & Install using terminal . . . . .	19
3.5	Custom - Advanced Installation . . . . .	20
<b>4</b>	<b>QuickStart</b>	<b>23</b>
4.1	Configure Matlab path . . . . .	23
4.2	Use a faust from a saved one . . . . .	23
4.3	Construct a Faust from a given matrix . . . . .	25
4.4	Construct a Faust from its factor . . . . .	26
<b>5</b>	<b>Example</b>	<b>27</b>
5.1	Brain Sources Localization . . . . .	27
<b>6</b>	<b>Annexes</b>	<b>29</b>
6.1	Required packages . . . . .	29
6.2	Compatibility between MATLAB and gcc compiler . . . . .	29
6.3	Further information about Build & Install process . . . . .	29

6.4	Required packages on Windows platform . . . . .	30
6.5	Matlab and processor architecture . . . . .	30
6.6	Add environment variable on Windows platform . . . . .	30
6.7	FAUST Install on MAC OS X platform, using Xcode IDE with terminal command	31
6.8	Optional FAuST installation using GPU process . . . . .	32

# Chapter 1

## Introduction

**Presentation:** FA $\mu$ ST is a C++ toolbox, useful to decompose a given dense matrix into a product of sparse matrices in order to reduce its computational complexity (both for storage and manipulation). In Figure 1.1, the matrix  $\mathbf{A}$  represents the dense matrix and  $\mathbf{S}_j$  correspond to the sparse matrices as  $\mathbf{A} = \prod_{j=1}^J \mathbf{S}_j$ .

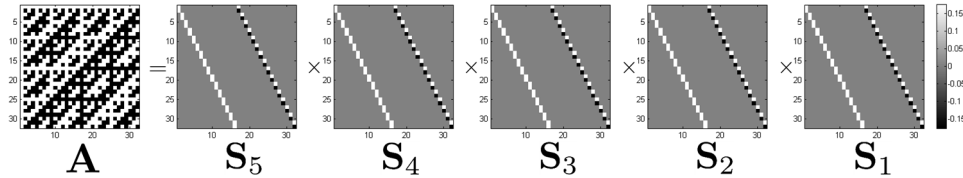
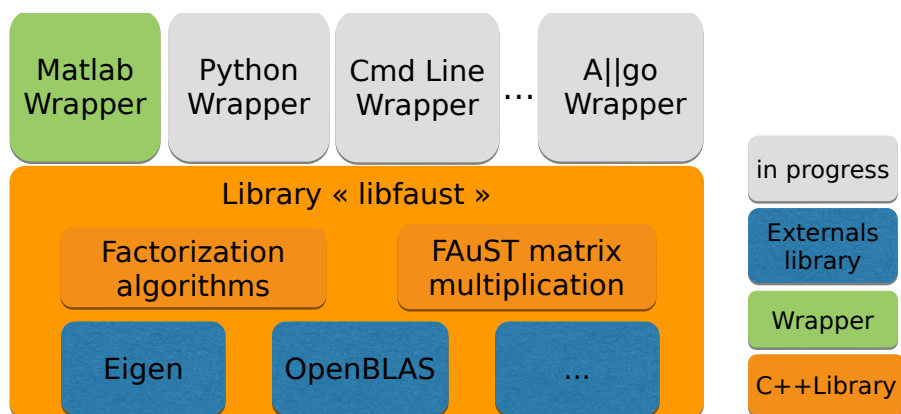


Figure 1.1: Brief presentation of FA $\mu$ ST

FA $\mu$ ST can be used to speed up iterative algorithms commonly used for solving high dimensional linear inverse problems. The algorithms implemented in the toolbox are described in details by Le Magoarou.[1]. The FA $\mu$ ST toolbox is delivered with a Matlab wrapper. For more information on the FAuST project, please visit the website of the project: <http://faust.gforge.inria.fr>.

**License:** Copyright (2016) Luc Le Magoarou, Remi Gribonval INRIA Rennes, FRANCE  
The FAuST Toolbox is distributed under the terms of the GNU Affero General Public License. This program is free software: you can redistribute it and/or modify it under the terms of the GNU Affero General Public License as published by the Free Software Foundation. This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Affero General Public License for more details. You should have received a copy of the GNU Affero General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

**FA $\mu$ ST Structure:** The Figure 1.2 presents an brief structure of the FA $\mu$ ST toolbox. The principal C++ library called "libfaust" includes two components: the "Factorization algorithms" is used to generate a FA $\mu$ ST core from a dense matrix and the "FA $\mu$ ST matrix multiplication" provides the linear operator to manipulate your data. It can be employ using different types of wrappers but in the second version of FA $\mu$ ST package (Version 2.0), only Matlab wrapper is provided.



**Figure 1.2:** Brief structure of FAuST version 2.0.0

**Valid Installation: Platforms, Compiler and IDE** The Figure 1.3 summarizes the tested configurations of the installation of the FA $\mu$ ST toolbox following the type of platform (Linux, Mac or Windows), compiler, version of Matlab and the type of Integrate Development Environment (IDE).

Choose among this Figure 1.3 the adapted platform and IDE following your system and refer to the corresponding Install Chapter. Authors suggest the installation using the GCC or Clang compiler directly from a command terminal because it is more simple and requires fewer externals components.

The installation on Windows Platform using Visual Studio IDE has been tested but there is compilation problems depending of the version of your Windows and your Visual Studio. **This install configuration is not guaranteed** but you can try and report any problem and/or suggestion on install-list of FA $\mu$ ST project on <http://lists.gforge.inria.fr/pipermail/faust-install/>.

Configuration Platform	Compiler and Matlab	Integrated Development Environment IDE	
LINUX	GCC (Version < v5.0) Matlab (v > 2013)	Code ::Blocks (with GCC)	
MAC OS X	Clang Matlab (v > 2013)	Code ::Blocks (with Clang)	Xcode
WINDOWS	GCC (MinGW Version 4.9 & 5.3) Matlab (v > 2013)	Code ::Blocks (with GCC)	Visual Studio

**Figure 1.3:** Checked Installation: Platforms and IDE

**Organization:** Chapter 2 explains how to install the library FA $\mu$ ST for UNIX platform and Chapter 3 corresponds to the Windows installation. Chapter 4 shows quickly how to use this library and finally an example is given Chapter 5.

## Chapter 2

# Installation on Unix platform

The FA $\mu$ ST project is based on **C++ library** available for Linux, MAC OS X and Windows platforms. The proposed toolbox provides a Matlab wrapper. **CMake** has been chosen to build the FA $\mu$ ST project because it is an open-source, cross-platform family of tools designed to build, test and package software. This chapter presents the steps to install the FA $\mu$ ST tools on Unix platform (both Linux and Mac OS).

Firstly, please ensure that the **prerequisite components** listed in Section 2.1 are installed.

Secondly, refer to section 2.2 to download FA $\mu$ ST package and setting your terminal.

Then refer to the appropriate section following the use (or not) of an IDE (Integrated Development Environment):

- Basic installation using **the command line terminal**, refer to Section 2.3.
- Basic installation using **the IDE "Code::Blocks"**, refer to Section 2.4.
- Basic installation using **the IDE "Xcode"** only for MAC OS X platform, refer to Section 2.5.

Finally in Section 2.6, the custom options available to build the FA $\mu$ ST toolbox are described to propose Custom - Advanced installation. For example, the optional configuration can be used to modify the install directory path, or to build in debug mode.

```
1 > This kind of script represents the text command you must
2 > entered in an terminal command. For example :
3 > mkdir BUILD
```

```
1 when character '>' is missing in the beginning of each line,
2 it signifies that the message is an return text in your
3 current terminal.
```

```
1 >> This kind of script represents the text command you must
2 >> entered in a Matlab Command Window.
```

```
1 when character ">>" is missing in the beginning of each line,
2 it signifies that the message is a return text in your current
3 Matlab Command Window.
```

## 2.1 Required components

This Section lists the required components you must install before to begin the FA $\mu$ ST installation.

- **Install CMake.** The minimum version required is Cmake version 3.0.2. Visit the website <https://cmake.org/> to process to the installation.
- **Verify Cmake install** by typing in a command terminal :

```
1 > which cmake
```

The command terminal returns the path of your Cmake binary file like

```
1 /usr/bin/cmake
```

If not, add Cmake binary directory in the environment path. (in your `/.bashrc` file)

- **Install Matlab** (minimal version required is Matlab 2014a <https://fr.mathworks.com/downloads/>)
- **Verify Matlab install** by typing in a terminal the following command :

```
1 > which matlab
```

You must obtain the path of your matlab binary file like:

```
1 /usr/local/bin/matlab
```

If not, add `matlab` binary directory in your environment path (in your `/.bashrc` file).

- **Verify Matlab** has a MEX supported compiler by typing in a Matlab Command Window :

```
1 >> mex -setup C++
```

You must obtain this kind of message :

```
1 MEX configured to use <YOURCOMPILER> for C++ language
  compilation.
```

But if you have an error message of the style :

```
1 No supported compiler or SDK was found. For options, visit
2 http://www.mathworks.com/support/compilers/R<20XXx>/<ARCH>.html
```

Visit the webpage specified in the error message, in order to see the list of compiler supported by your Matlab version R<20XXx> on your platform <ARCH>.

## 2.2 Download Faust Package & launch terminal

When prerequisites listed in precedent section 2.1 are checked, you can get the package FA $\mu$ ST.

- **Download** the FA $\mu$ ST package on the website : <http://faust.gforge.inria.fr/>
- **Unzip** the FA $\mu$ ST package into your FA $\mu$ ST directory.
- **Open** a command terminal
- **Set the current directory** to your FA $\mu$ ST directory (NOTE: do not use any special character in your FA $\mu$ ST directory path, for example the character  $\mu$ )

## 2.3 Basic Build & Installation using Makefile

When you have done the step in section 2.2 (i.e download Faust package and launch the terminal in the right directory), the FA $\mu$ ST installation can start. If you are administrator of your machine (root access), follow instructions given in Section 2.3.1. Otherwise, for local installation, refer to Section 2.3.2.

### 2.3.1 Install with administrator privilege

In the terminal opened in Section 2.2, type the following commands :

```
1 > mkdir build
2 > cd build
3 > cmake ..
4 > make
5 > sudo make install # run with administrator privilege
```

FA $\mu$ ST Toolbox should be installed. Now, refer to Quick-Start Chapter 4 to check the install and to try FA $\mu$ ST toolbox.

For more detail about `cmake` ; `make` ; `make install` commands, refer to Section 6.3.

### 2.3.2 Install without administrator privilege

In the terminal opened in Section 2.2, type the following commands :

```
1 > mkdir build
2 > cd build
3 > cmake .. -DCMAKE_INSTALL_PREFIX="<Your/Install/Dir>"
4 > make
5 > make install
```

FA $\mu$ ST Toolbox should be installed. Now, refer to Quick-Start Chapter 4 to check the install and to try FA $\mu$ ST toolbox.

For more detail about `cmake` ; `make` ; `make install` commands, refer to Section 6.3.

## 2.4 Basic Build & Install using Code Block IDE

When you have done the step in section 2.2 (i.e download Faust package and launch the terminal in the right directory), the FA $\mu$ ST installation can start. If you are administrator of your machine (root access), follow instructions given in Section 2.4.1. Otherwise, for local installation, refer to Section 2.4.2.



### 2.4.1 Install with administrator privilege

- In the terminal opened in section 2.2, type the following commands :

```
1 > mkdir build
2 > cd build
3 > cmake .. -G "CodeBlocks - Unix Makefiles"
```

- Open the FA $\mu$ ST project from the file `./build/FAUST.cbp` with Code::Blocks IDE.
- In Code::Blocks IDE, select **ALL** target and build the project.
- Close Code::Blocks IDE
- Re-Open the FA $\mu$ ST project from the file `./build/FAUST.cbp` with Code::Blocks IDE **with administrator privilege**. For that, type in a terminal :

```
1 > sudo codeblocks
```

- In Code::Blocks IDE, select **install** target and build the project.

FA $\mu$ ST Toolbox should be installed. Now, refer to Quick-Start Chapter 4 to check the install and to try FA $\mu$ ST toolbox.

For more detail about `cmake` command, refer to Section 6.3.

### 2.4.2 Install without administrator privilege

- In the terminal opened in section 2.2, type the following commands :

```
1 > mkdir build
2 > cd build
3 > cmake .. -G "CodeBlocks - Unix Makefiles"
4 -DCMAKE_INSTALL_PREFIX="<Your/Install/Dir>"
```

- Open the FA $\mu$ ST project from the file `./build/FAUST.cbp` with Code::Blocks IDE.
- In Code::Blocks IDE, select **ALL** target and build the project.
- In Code::Blocks IDE, select **install** target and build the project.

FA $\mu$ ST Toolbox should be installed. Now, refer to Quick-Start Chapter 4 to check the install and to try FA $\mu$ ST toolbox.

For more detail about `cmake` command, refer to Section 6.3.

## 2.5 Basic Build & Install using Xcode IDE (for MAC OS)

FA $\mu$ ST install using the IDE Xcode concerns only MAC OS X environment.

When you have done the step in section 2.2 (i.e download Faust package and launch the terminal in the right directory), the FA $\mu$ ST installation can start. This Build & Install section requires that you have the IDE Xcode installed on your system. If you are administrator of your machine (root access), follow instructions given in Section 2.5.1. Otherwise, for local installation, refer to Section 2.5.2.

**NOTE:** For a command line install using Xcode IDE, refer to Annex 6.7.

### 2.5.1 Install with administrator privilege

- In the terminal opened in section 2.2, type the following commands :

```
1 > mkdir build
2 > cd build
3 > cmake .. -G "Xcode"
```

- Open the FA $\mu$ ST project from the file `./build/FAUST.xcodeproj` with Xcode IDE.
- In Xcode IDE, select **ALL** target and build the project.
- Close Xcode IDE
- Re-Open the FA $\mu$ ST project from the file `./build/FAUST.xcodeproj` with Xcode IDE **with administrator privilege**. For that, type in a terminal:

```
1 > sudo Xcode
```

- In Xcode IDE, select **install** target and build the project.

FA $\mu$ ST Toolbox should be installed. Now, refer to Quick-Start Chapter 4 to check the install and to try FA $\mu$ ST toolbox.

For more detail about `cmake` command, refer to Section 6.3.

### 2.5.2 Install without administrator privilege

- In the terminal opened in section 2.2, type the following commands :

```
1 > mkdir build
2 > cd build
3 > cmake .. -G "Xcode"
4 -DCMAKE_INSTALL_PREFIX="<Your/Install/Dir>"
```

- Open the FA $\mu$ ST project from the file `./build/FAUST.xcodeproj` with Xcode IDE.
- In Xcode IDE, select **ALL** target and build the project.
- In Xcode IDE, select **install** target and build the project.

FA $\mu$ ST Toolbox should be installed. Now, refer to Quick-Start Chapter 4 to check the install and to try FA $\mu$ ST toolbox.

For more detail about `cmake` command, refer to Section 6.3.

## 2.6 Custom - Advanced Installation

The project FA $\mu$ ST can be configured with optional parameters, for example if you want to install FA $\mu$ ST library in a different folder or to enable the parallel computing using multithread capacities provided by the OS. This build system can be parametrized using the Cmake Graphical User Interface, or the Cmake command line tools.

The Cmake Graphical User Interface `ccmake` allows you to select option input. Set your current directory to your build directory and type in a terminal :

```
1 > ccmake ..
```

The Cmake GUI appears in the console (see fig. 2.1).

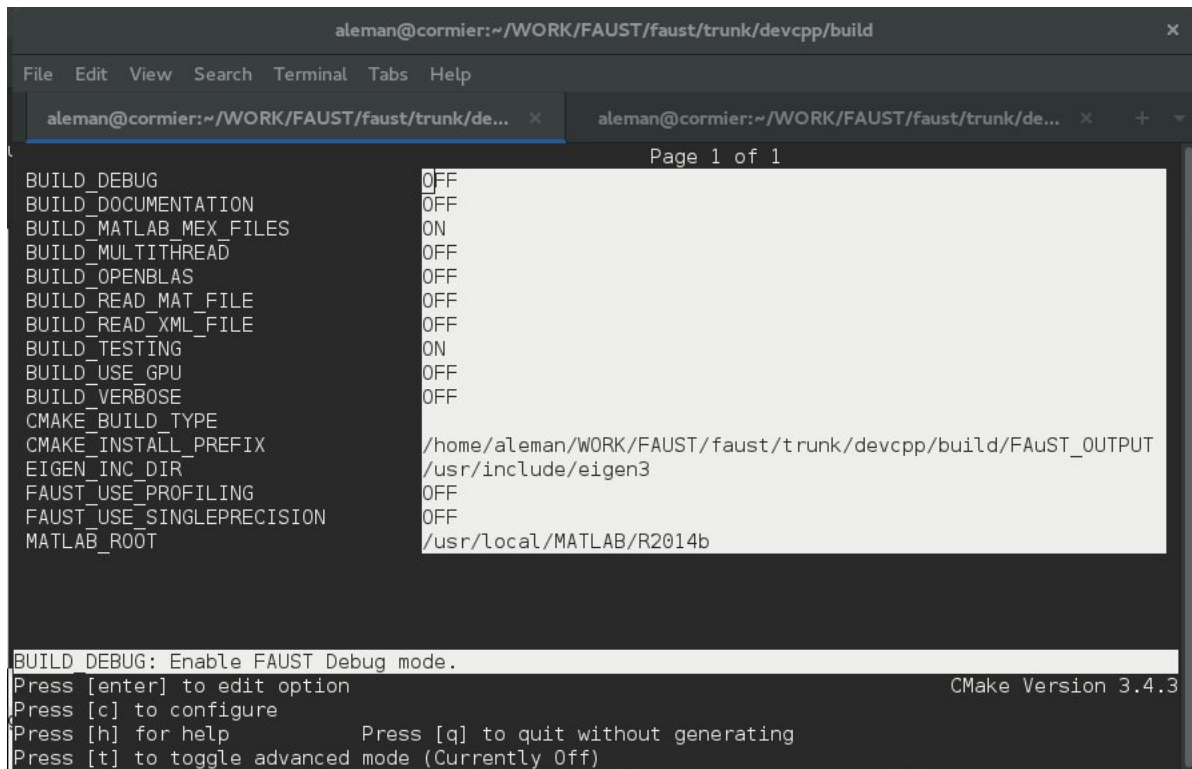


Figure 2.1: ccmake GUI

When scrolling on a value and pressing [enter], this value can be edited, the black underlaid row displays some information about the option and required path to create the build system. In the case of an option press [enter] to toggle the ON/OFF values. You can edit option by pressing [enter]. For example, press [enter] to edit option `CMAKE_INSTALL_PREFIX` to modify the install directory.

After choosing options for the build and setting the required fields, press [c] to configure. The configuration of the build system is checked again by Cmake, at the end of this check if the build settings are correct, you can press [g] in order to generate the build system.

Instead the ccmake GUI, an other possibility to configure and generate the project is to use the command line cmake which can take the option input. Here is the list of available options: `cmake .. -D<BUILD_NAME>=<value>`

- `CMAKE_INSTALL_PREFIX` : Install directory
- `BUILD_TESTING` : Enable the ctest option (default value is ON)

- `BUILD_DOCUMENTATION` : Generating the doxygen documentation (default value is OFF)
- `BUILD_MULTITHREAD` : Enable multithread with OpenMP Multithreading (default value is OFF)
- `BUILD_VERBOSE` : Enable verbose option when compile (-v) (default value is OFF)
- `BUILD_DEBUG` : Enable FA $\mu$ ST Debug mode (default value is OFF )
- `BUILD_USE_GPU` : Using both CPU and GPU process (default value is OFF) (refer to Annex 6.8 for installation and more detail)
- `BUILD_MATLAB_MEX_FILES` : Enable building Matlab MEX files (default value is ON)
- `BUILD_OPENBLAS` : Using openBLAS for matrix and vector computations (default value is OFF )

Following the selected option, the cmake installer automatically checks the dependent component (library OpenBlas, eigen, matio, libxml2).

## Chapter 3

# Installation on Windows platform

The FA $\mu$ ST project is based on **C++ library** available for Linux, MAC OS X and Windows platforms. The proposed toolbox provides a Matlab wrapper. **CMake** has been chosen to build the FA $\mu$ ST project because it is an open-source, cross-platform family of tools designed to build, test and package software. This chapter presents the steps to install the FA $\mu$ ST tools on Windows platform.

- Firstly, please ensure that the **prerequisite components** listed in Section 3.1 are installed.
- Secondly, **choose your preferred C++ Compiler** between GCC from MinGW "Minimalist GNU for Windows"(Section 3.3) and the C++ Compiler from Microsoft Visual Studio (Section 3.4).
- Then, **process to Basic installation** of FA $\mu$ ST tool by following instructions given in the appropriate section, depending to the kind of compiler (MinGW or Microsoft Visual) and the use (or not) of an IDE (Integrated Development Environment).

Section 3.5 describes the custom options available to build the FA $\mu$ ST toolbox, to propose Custom - Advanced installation. For example, the optional configuration can be used to modify the install directory path, or to build in debug mode.

```
1 > This kind of script represents the text command you must
2 > entered in an terminal command. For example :
3 > mkdir BUILD
```

```
1 when character '>' is missing in the beginning of each line,
2 it signifies that the message is an return text in your
3 current terminal.
```

```
1 >> This kind of script represents the text command you must
2 >> entered in a Matlab Command Window.
```

```
1 when character ">>" is missing in the beginning of each line,
2 it signifies that the message is a return text in your current
3 Matlab Command Window.
```

## 3.1 Required components

The installation of the FA $\mu$ ST tool depends on other components to be installed in order to run properly.

- **Install CMake.** The minimum version required is Cmake version 3.0.2. Download Binary distributions correspond to your environment from <https://cmake.org/download/>.
- **Verify CMake install :** Open a terminal and type the following command:

```
1 > where cmake
```

You should obtain the path of your `cmake.exe` binary file. If not, please add the directory of your `cmake.exe` file in your environment variable. (to add an environment variable, follow instructions given in 6.6).

- **Install Matlab** (minimal version required is Matlab 2014a) (<https://fr.mathworks.com/downloads/>).
- **Verify Matlab install** by checking the `matlab.exe` binary file in the default directory: type in a terminal the following command :

```
1 > where matlab
```

You must obtain the path of your matlab binary file like:

```
1 C:\Program Files\MATLAB\<R2015b>\bin\matlab.exe
```

If not, please add the directory of your `matlab.exe` file in your environment variable. (to add an environment variable, follow instructions given in 6.6).

- **Verify Matlab** has a MEX supported compiler by typing in a Matlab Command Window :

```
1 >> mex -setup C++
```

You must obtain this kind of message :

```
1 MEX configured to use <YOURCOMPILER> for C++ language
  compilation.
```

But if you have an error message of the style :

```
1 No supported compiler or SDK was found. For options, visit
2 http://www.mathworks.com/support/compilers/R<20XXx>/<ARCH>.html
```

Visit the webpage specified in the error message, in order to see the list of compiler supported by your Matlab version R<20XXx> on your platform <ARCH>.

- **Install 7-zip tool** from <http://www.7-zip.org/>.
- **Verify 7-zip install** by typing in a terminal the following command :

```
1 > where 7z
```

You must obtain the path of your `7z.exe` binary file:

```
1 C:\Program Files\7-Zip\7z.exe
```

If not, add `7z.exe` directory in your environment path. (to add an environment variable, follow instructions given in 6.6).

## 3.2 Download Faust Package

When prerequisite components listed in precedent section 3.1 are checked, you can get the FA $\mu$ ST package.

- **Download** the FA $\mu$ ST package on the website : <http://faust.gforge.inria.fr/>
- **Unzip** the FA $\mu$ ST package into your FA $\mu$ ST directory.

```
1 > 7z x FAUST-Source-2.0.0.zip
```

## 3.3 Install using GCC-MinGW compiler

When prerequisite components listed in previous section 3.1 are checked, you must install **MinGW containing the C++ Compiler**. Then, the FA $\mu$ ST installation can be done using Command prompt, following instructions given in Section 3.3.2 or using CodeBlocks IDE, following instructions given in Section 3.3.3.

### 3.3.1 Install GCC from MinGW

- **Download MinGW** in <https://sourceforge.net/projects/mingw/files/latest/download?source=files>
- **Launch install file** and choose latest version of MinGW. Select the MinGW corresponding to gcc-g++ compiler. Then, select mingw32-make.exe in "All packages" menu.
- Add `gcc.exe` directory in your environment path. (to add an environment variable, follow instructions given in 6.6).
- **Verify MinGW install** by typing in a terminal the following command :

```
1 > where gcc
```

You must obtain the path of your `gcc.exe` binary file:

```
1 C:\mingw\mingw64\bin\gcc.exe
```

If not, add `gcc.exe` directory in your environment path. (to add an environment variable, follow instructions given in 6.6).

- **Verify make tool:** In a terminal command, type

```
1 > where make
```

You must obtain the path of your `make.exe` binary file. If not, please check if `make.exe` file is present in MINGW install directory. If not, you can copy and rename `mingw32-make.exe` to `make.exe`.

- From Matlab IDE, you must install MinGW version 4.9.2 using the **ADDON menu**. For more detail, please follow the instruction given in following link : [http://fr.mathworks.com/help/matlab/matlab\\_external/install-mingw-support-package.html](http://fr.mathworks.com/help/matlab/matlab_external/install-mingw-support-package.html). For that, you must have an id session for Mathwork (easy to create). Current this latest step, an environment variable called `MW_MINGW64_LOC` is automatically generated.

### 3.3.2 Basic Build & Install using the command prompt

The basic install described in this section requires the GCC compiler from MinGW (see precedent Section 3.3.1). If you have administrator privilege, follow instructions in Section 3.3.2.1, otherwise follow instructions given in Section 3.3.2.2.

#### 3.3.2.1 Install with administrator privilege

- Open a command terminal
- Set the current directory to your FA $\mu$ ST directory (NOTE: don't use any special character in your FA $\mu$ ST directory path, for example the character  $\mu$ )
- Type the following commands :

```
1 > mkdir build
2 > cd build
3 > cmake -G "MinGW Makefiles" ..
4 > make
```

- Open a command terminal **with administrator privilege** : right click on command prompt icon and select run as administrator.
- Set the current directory to your FA $\mu$ ST directory
- Type the following commands :

```
1 > make install
```

FA $\mu$ ST Toolbox should be installed. Now, refer to Quick-Start Chapter 4 to check the install and to try FA $\mu$ ST toolbox.



### 3.3.2.2 Install without administrator privilege

- Open a command terminal
- Set the current directory to your FA $\mu$ ST directory (NOTE: don't use any special character in your FA $\mu$ ST directory path, for example the character  $\mu$ )
- Type the following commands :

```
1 > mkdir build
2 > cd build
3 > cmake -G "MinGW Makefiles" ..
4         -DCMAKE_INSTALL_PREFIX="<Your/Install/Dir>"
5 > make
6 > make install
```

FA $\mu$ ST Toolbox should be installed. Now, refer to Quick-Start Chapter 4 to check the install and to try FA $\mu$ ST toolbox.

### 3.3.3 Basic build & Install using Code::Blocks IDE

The basic install described in this section requires the GCC compiler from MinGW (see precedent Section 3.3.1). If you have administrator privilege, follow instructions in Section 3.3.3.1, otherwise follow instructions given in Section 3.3.3.2.

You must select the gcc compiler in the IDE code::Blocks in the Settings menu -> Compiler... -> Selected Compiler

Then, the FA $\mu$ ST installation can be done.

#### 3.3.3.1 Install with administrator privilege

- Open a command terminal
- Set the current directory to your FA $\mu$ ST directory (NOTE: don't use any special character in your FA $\mu$ ST directory path, for example the character  $\mu$ )
- Type the following commands :

```
1 > mkdir build
2 > cd build
3 > cmake -G "CodeBlocks - MinGW Makefiles" ..
```

- Open the FA $\mu$ ST project from the file ./build/FAUST.cbp with Code::Blocks IDE.
- In Code::Blocks IDE, select ALL BUILD, select RELEASE mode and generate the project.
- Re-Open the FA $\mu$ ST project from the file ./build/FAUST.cbp with Code::Blocks IDE **with administrator privilege**. For that, right-click on the Code::Blocks icon and select "run as administrator".
- In Code::Blocks IDE, select INSTALL target, select RELEASE mode and build.

FA $\mu$ ST Toolbox should be installed. Now, refer to Quick-Start Chapter 4 to check the install and to try FA $\mu$ ST toolbox.

For more detail about cmake commands, refer to Section 6.3.

### 3.3.3.2 Install without administrator privilege

- Open a command terminal
- Set the current directory to your FA $\mu$ ST directory (NOTE: don't use any special character in your FA $\mu$ ST directory path, for example the character  $\mu$ )
- Type the following commands :

```
1 > mkdir build
2 > cd build
3 > cmake -G "CodeBlocks - MinGW Makefiles" ..
4         -DCMAKE_INSTALL_PREFIX="<Your/Install/Dir>"
```

- Open the FA $\mu$ ST project from the file `./build/FAUST.cbp` with Code::Blocks IDE.
- In Code::Blocks IDE, select ALL target, select RELEASE mode and build the project.
- In Code::Blocks IDE, select install target, select RELEASE mode and build the project.

FA $\mu$ ST Toolbox should be installed. Now, refer to Quick-Start Chapter 4 to check the install and to try FA $\mu$ ST toolbox.

For more detail about cmake commands, refer to Section 6.3.

## 3.4 Install using Microsoft Visual compiler

The installation on Windows Platform using Visual Studio IDE has been tested but there is compilation problems depending of the version of your Windows and your Visual Studio. **This install configuration is not guaranteed** but you can try and report any problem and/or suggestion on install-list of FA $\mu$ ST project on <http://lists.gforge.inria.fr/pipermail/faust-install/>.

When prerequisites listed in section 3.1 are checked, you must install **Microsoft Visual Studio** containing the C++ Compiler (Section 3.4.1). Then, the FA $\mu$ ST installation can be done using Visual Studio IDE, following instructions given in Section 3.4.2 or using Command prompt, following instructions given in Section 3.4.3.

### 3.4.1 Install Microsoft Visual compiler

- Download Microsoft Visual Studio Professional 2013 from <https://www.microsoft.com/en-US/download/details.aspx?id=44916>
- Install Microsoft Visual Studio Professional 2013

Then, the FA $\mu$ ST installation can be done using Visual Studio IDE, following instructions given in Section 3.4.2 or using Command prompt, following instructions given in Section 3.4.3.

### 3.4.2 Basic Build & Install using Visual Studio IDE

The basic install described in this section requires Microsoft Visual Studio (see precedent Section 3.4.1). If you have administrator privilege, follow instructions in Section 3.4.2.1, otherwise follow instructions given in Section 3.4.2.2.

### 3.4.2.1 Install with administrator privilege

- Open a command terminal
- Set the current directory to your FA $\mu$ ST directory (NOTE: don't use any special character in your FA $\mu$ ST directory path, for example the character  $\mu$ )
- Type the following commands :

```
1 > mkdir build
2 > cd build
3 > cmake ..
```

- Open the FA $\mu$ ST project from the file `./build/FAUST.sln` with Visual Studio IDE.
- In Visual Studio IDE, select ALL BUILD, select RELEASE mode, and generate the project three times.
- Close Visual Studio IDE
- Re-Open the FA $\mu$ ST project from the file `./build/FAUST.sln` with Visual Studio IDE with **administrator privilege**.
- In Visual Studio IDE, select INSTALL target, select RELEASE mode, and generate the project.

FA $\mu$ ST Toolbox should be installed. Now, refer to Quick-Start Chapter 4 to check the install and to try FA $\mu$ ST toolbox.

For more detail about cmake commands, refer to Section 6.3.

### 3.4.2.2 Install without administrator privilege

- Open a command terminal
- Set the current directory to your FA $\mu$ ST directory (NOTE: don't use any special character in your FA $\mu$ ST directory path, for example the character  $\mu$ )
- Type the following commands :

```
1 > mkdir build
2 > cd build
3 > cmake .. -DCMAKE_INSTALL_PREFIX="<Your/Install/Dir>"
```

- Open the FA $\mu$ ST project from the file `./build/FAUST.sln` with Visual Studio IDE.
- In Visual Studio IDE, select ALL target, select RELEASE mode and generate the project three times.
- In Visual Studio IDE, select install target, select RELEASE mode, and generate the project.

FA $\mu$ ST Toolbox should be installed. Now, refer to Quick-Start Chapter 4 to check the install and to try FA $\mu$ ST toolbox.

For more detail about cmake commands, refer to Section 6.3.

### 3.4.3 Basic Build & Install using terminal

The basic install described in this section requires the Microsoft Visual Studio compiler (see precedent Section 3.4.1). If you have administrator privilege, follow instructions in Section 3.4.3.1, otherwise follow instructions given in Section 3.4.3.2.

#### 3.4.3.1 Install with administrator privilege

In the case of **Microsoft Visual Studio 2013 compiler using the command terminal** :

- Open a command terminal
- Set the current directory to your FA $\mu$ ST directory (NOTE: don't use any special character in your FA $\mu$ ST directory path, for example the character  $\mu$ )
- Type the following commands :

```
1 > mkdir build
2 > cd build
3 > cmake ..
4 > cmake --build . --config "Release"
5 > cmake --build . --config "Release"
```

- Re-Open a command terminal **with administrator privilege**

```
1 > cmake --build . --config "Release" --target "install"
```

FA $\mu$ ST Toolbox should be installed. Now, refer to Quick-Start Chapter 4 to check the install and to try FA $\mu$ ST toolbox.

For more detail about cmake commands, refer to Section 6.3.

#### 3.4.3.2 Install without administrator privilege

In the case of **Microsoft Visual Studio 2013 compiler using the command terminal** :

- Open a command terminal
- Set the current directory to your FA $\mu$ ST directory (NOTE: don't use any special character in your FA $\mu$ ST directory path, for example the character  $\mu$ )
- Type the following commands :

```
1 > mkdir build
2 > cd build
3 > cmake .. -DCMAKE_INSTALL_PREFIX="<Your/Install/Dir>"
4 > cmake --build . --config "Release"
5 > cmake --build . --config "Release"
6 > cmake --build . --config "Release" --target "install"
```

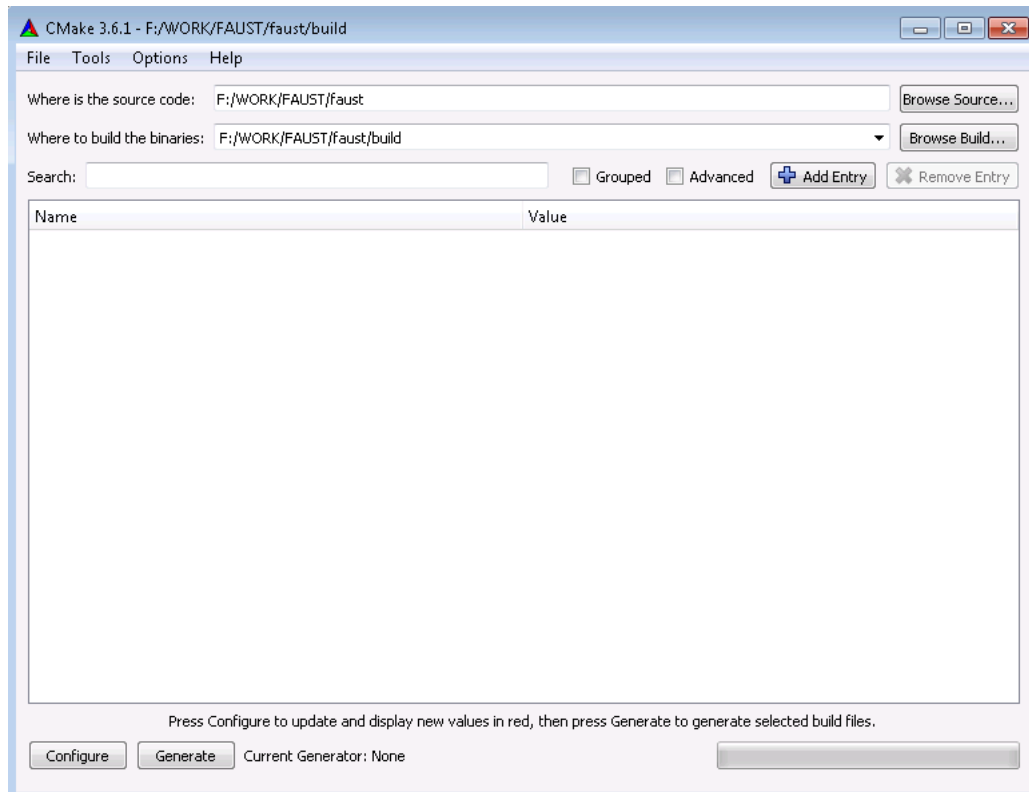
FA $\mu$ ST Toolbox should be installed. Now, refer to Quick-Start Chapter 4 to check the install and to try FA $\mu$ ST toolbox.

For more detail about cmake commands, refer to Section 6.3.

### 3.5 Custom - Advanced Installation

The project FA $\mu$ ST can be configured with optional parameters, for example if you want to install FA $\mu$ ST library in a different folder or to enable the parallel computing using multithread capacities provided by the OS. This build system can be parametrized using the Cmake Graphical User Interface, or the Cmake command line tools. The Cmake Graphical User Interface allows you to select option input.

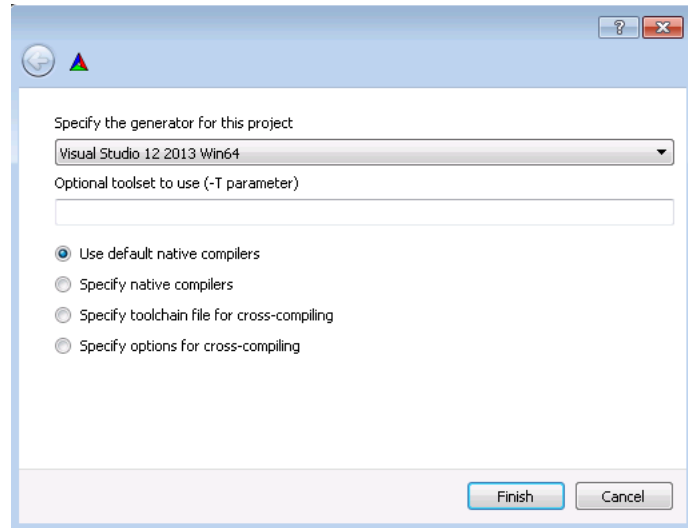
1. Open application `cmake-GUI.exe` from the program menu or from your cmake install binaries directory to launch the CMake configuration application:



**Figure 3.1:** cmake GUI

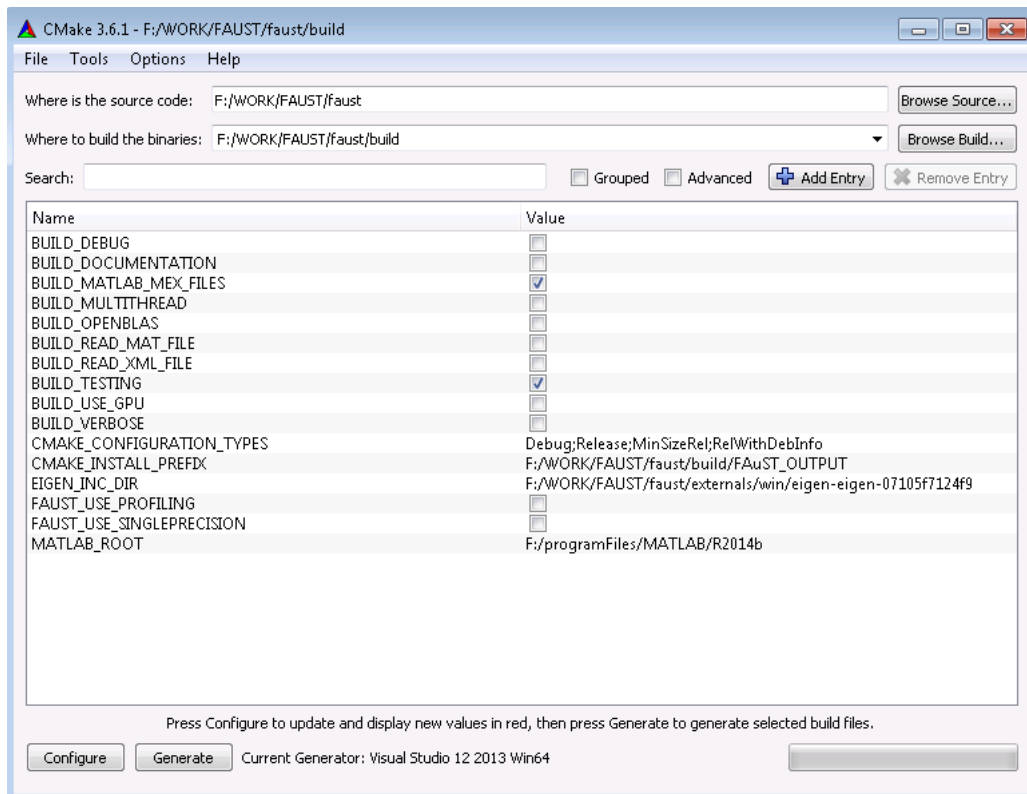
2. Set the "Where is the source code:" text box with the path of the directory where the source files are located (F:/WORK/FAUST/faust) and the "Where to build the binaries:" with the path of the directory where you want to build the library and executable files (F:/WORK/FAUST/faust/build). (see fig. 3.1).

When clicking for the first time on the [Configure] button, CMake will ask for the build tool you want to use. The build system type depends on the builder you want to use, in our case this is the Visual Studio X (X depending the version of Visual installed on the computer) chain tools. (see fig. 3.2).



**Figure 3.2:** cmake GUI

3. When pressing again the [Configure] button to configure the build system, CMake performs a list of tests to determine the system configuration and manage the build system. If the configuration is correct then no pop-up will appear during the tests and CMake finally shows the various options of the build underlaid in grey. In case of a configuration issue, a pop up window warns you about this issue indicating which test has failed, in this case the build option in the CMake application software will be underlaid in red. We will discuss in Section 3.5 what to do in such a case, but let us for the moment assume that everything ran smoothly. (see 3.3).



**Figure 3.3:** cmake GUI

Here is the list of available options:

- CMAKE\_INSTALL\_PREFIX : Install directory
- BUILD\_TESTING : Enable the ctest option (default value is ON)
- BUILD\_DOCUMENTATION : Generating the doxygen documentation (default value is OFF)
- BUILD\_MULTITHREAD : Enable multithread with OpenMP Multithreading (default value is OFF)
- BUILD\_VERBOSE : Enable verbose option when compile (-v) (default value is OFF)
- BUILD\_DEBUG : Enable FA $\mu$ ST Debug mode (default value is OFF )
- BUILD\_USE\_GPU : Using both CPU and GPU process ( default value is OFF) (refer to Annex [6.8](#) for installation and more detail)
- BUILD\_MATLAB\_MEX\_FILES : Enable building Matlab MEX files (default value is ON)
- BUILD\_OPENBLAS : Using openBLAS for matrix and vector computations (default value is OFF )

# Chapter 4

## QuickStart

A Matlab wrapper is delivered with the FA $\mu$ ST C++ library. It provides a user friendly new class of matrix **Faust** efficient for the multiplication with matlab built-in dense matrix class.

### 4.1 Configure Matlab path

In order to use matlab wrapper, follow the instructions :

- **Install** FA $\mu$ ST tool (see Chapter 2 in case of Unix install or Chapter 3 in case of Windows install)
- **Launch** Matlab.
- **Set the working directory** of the Matlab Command Window to your FA $\mu$ ST install directory (default install directory is /<HOMEDIR>/Documents/MATLAB/Faust)
- **Configure** the Matlab path by typing the following commands :

```
1 >> cd /<HOMEDIR>/Documents/MATLAB/Faust
2 >> setup_Faust
```

### 4.2 Use a faust from a saved one

Now, you can run `quick_start.m` script in the Matlab Command Window by typing :

```
1 >> quick_start
```

`quick_start.m` script is located in following path :

<HOMEDIR>/Documents/MATLAB/faust/demo/Quick\_start/quick\_start.m

In this script, first of all, a Faust of size 4000x5000 is loaded from a previous one that is saved into a matfile located in :

<HOMEDIR>/Documents/MATLAB/faust/demo/Quick\_start/faust\_quick\_start.mat

```
1 % loading a Faust A from saved-one
2 A=Faust('faust_quick_start.mat');
```



Secondly, a list of overloaded matlab function shows that a Faust is handled as a normal Matlab builtin matrix.

```
1  % get the size of the faust
2  [dim1,dim2] = size(A);
3
4  % transpose a faust
5  A_trans = A';
6
7  % multiplication by A
8  x1 = rand(dim2,1);
9  y1 = A*x1;
10
11 % multiplication by A'
12 x2 = rand(dim1,5);
13 y2 = A'*x2;
14
15
16 % get the 2-norm (spectral norm) of the faust A
17 norm_A = norm(A); % equivalent to norm(A,2);
18
19 % convert Faust to full matrix
20 A_full=full(A);
21
22
23 % get the coefficient i,j and slicing for reading purpose
24 coeff=A(3,4);
25 col_2=A(:,2);
26 submatrix_A=A(3:5,2:3);
27 submatrix_A=A(2:end,3:end-1);
28
29 % Warning : A(i,j)=3; will throw an error with this message :
30 %          'function not implemented for Faust class'
```

Finally, it performs a little time comparison between multiplication by a Faust or its full matrix equivalent. This is in order to illustrate the speed-up induced by the Faust. This speed-up should be around 30 (depending on your machine).

### 4.3 Construct a Faust from a given matrix

To see an example of building a Faust from a matrix, you can run `factorise_matrix.m` in the Matlab Command Window by typing :

```
1 >> factorise_matrix
```

`factorise_matrix.m` script is located in following path :

`<HOMEDIR>/Documents/MATLAB/faust/demo/Quick_start/factorise_matrix.m`

In this script, from a given matrix **A** of size 100x200 :

```
1 % number of row of the matrix
2 dim1 = 100;
3 % number of column of the matrix
4 dim2 = 200;
5 % matrix to factorise
6 A = rand(dim1,dim2);
```

We generate the parameters of the factorisation from :

- The dimension of **A** (**dim1** and **dim2**),
- **nb\_factor** the number of factor of the Faust,
- **rcg** the Rational Complexity Gain, which represents the theoretical memory gain and multiplication speed-up of the Faust compared to the initial matrix .

**WARNING :** A trade-off exists between the RCG/speed-up of the Faust and the data fidelity to the input matrix. The higher the RCG, the higher the error of the Faust relative to the input matrix

```
1 % Rational complexity Gain (theoretical speed-up) of the Faust
2 rcg = 100;
3 % number of factor of the Faust
4 nb_factor = 2;
5 %% generate parameters of the factorisation
6 params = generate_params(dim1,dim2,nb_factor,rcg);
```

Then we factorize the matrix **A** into a Faust **Faust\_A**

```
1 %% factorisation (create Faust from matrix A)
2 faust_A = faust_decompose(A,params);
```

And as for `quickstart.m`, we make some time comparison at the end.

## 4.4 Construct a Faust from its factor

To see an example of building a Faust from its factors, you can run `construct_Faust_from_factors.m` in the Matlab Command Window by typing :

```
1 >> construct_Faust_from_factors
```

This following example shows how to build a faust from a cell-array representing its factors.

```
1 % number of row of the Faust
2 dim1=300;
3 % number of column of the Faust
4 dim2=100;
5 % number of factor of the Faust
6 nb_factor=4;
7 % density of each factor
8 density_factor=0.1;
9
10 %cell-array representing its factors
11 factors = cell(1,nb_factor);
12
13 % 1st factor is a rectangular sparse matrix of density equal to
    density_factor
14 factors{1} = sprand(dim1,dim2,density_factor);
15
16 % all the others factor are square sparse matrix of density equal
    to density_factor
17 for i=2:nb_factor
18     factors{i} = sprand(dim2,dim2,density_factor);
19 end
20
21 %% construct the Faust
22 A_faust=Faust(factors);
23
24
25 % a multiplicative scalar can be taken into account to construct
    the Faust
26 lambda=2;
27
28 % B_faust=lambda*A_faust
29 B_faust=Faust(factors,lambda);
```

## Chapter 5

# Example

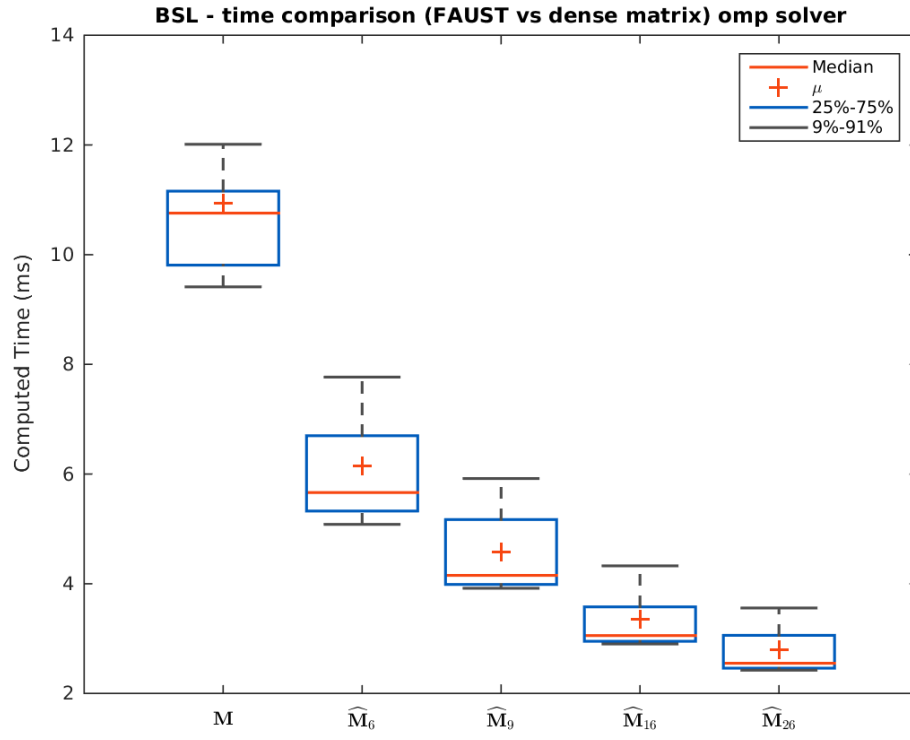
### 5.1 Brain Sources Localization

An experience of Brain Source Localization using several gain matrices including FA $\mu$ ST and several solvers is provided. After configuring the Matlab path (cf section 4.1), you just need to type in the Matlab Command Window:

```
1 >> BSL
2 >> Fig_BSL
```

BSL.m runs the experiment (see the script file :  
/<HOMEDIR>/Documents/MATLAB/Faust/demo/Brain\_source\_localization/BSL.m)  
Fig\_BSL.m displays the Figure 5.1. illustrating the speed-up using a FA $\mu$ ST. (see the script file:  
/<HOMEDIR>/Documents/MATLAB/Faust/demo/Brain\_source\_localization/Fig\_BSL.m)

This figure illustrates the speed-up with a FA $\mu$ ST ( $\mathbf{M}$  is the gain dense matrix and  $\widehat{\mathbf{M}}_6, \widehat{\mathbf{M}}_9, \widehat{\mathbf{M}}_{16}, \widehat{\mathbf{M}}_{26}$  are different FA $\mu$ ST representing  $\mathbf{M}$ ):



**Figure 5.1:** Performance of FAuST tool on Brain Source Localization experiment

# Chapter 6

## Annexes

### 6.1 Required packages

Here is a list of packages used in the FA $\mu$ ST project. The installation of this packages are automatically done. There are nothing to do. (see the source directory `"/externals"`).

- Library **Eigen** <http://eigen.tuxfamily.org>: C++ template library for linear algebra: matrices, vectors, numerical solvers, and related algorithms.
- Library **OpenBLAS** <http://www.openblas.net>: Optimized BLAS library based on Go-toBLAS2 1.13 BSD version.

### 6.2 Compatibility between MATLAB and gcc compiler

Adjust your version of GCC compiler in order to run the installation properly. The use of the `mex` function in Matlab requires that you have a third-party compiler installed on your system. The latest version of Matlab (2016a in our case) only supports up to GCC 4.7 (see <http://fr.mathworks.com/support/compilers/R2016a/index.html?sec=glxa64> for more detail).

To temporally change your version of `cc` and `cxx` compiler, you can modify the environment variable `CC` an `CXX`. For that, export your `CC` and `CXX` variables corresponding to `gcc` and `g++` binaries path :

- find your `gcc` and `g++` version path using `which` command in a terminal :

```
1 > which gcc
2 > which g++
```

- Open your `/.bashrc` file and save the return-path of `gcc` and `g++` like:

```
1 # export version of gcc
2 export CC=/usr/lib64/ccache/gcc
3 export CXX=/usr/lib64/ccache/g++
```

### 6.3 Further information about Build & Install process

When using the **cmake** command to generate the build system, **cmake** performs a list of tests to determine the system configuration and manage the build system. If the configuration is

correct then the build system is generated and written. In this case, the three last lines of the console log of **cmake** command should be:

```
1 -- Configuring done
2 -- Generating done
3 -- Build files have been written to: <YOUR/LOCAL/DIRECTORY/build>
```

The command **make** will compile the build files.

The command **sudo make install** will install the library and others components in the default directory:

/usr/local/lib/libfaust.a for the FA $\mu$ ST library,

~/Documents/MATLAB/faust/ for the wrapper matlab.

You must have administrator privilege because the library file **libfaust.a** is copied in an root path directory. If you do not have administrator privilege, you can realize a local install using **cmake** optional parameter **-DCMAKE\_INSTALL\_PREFIX=<Your/Install/Dir>**.

The **cmake** optional parameter **-DCMAKE\_INSTALL\_PREFIX=<Your/Install/Dir>** allows to install the binaries on the selected install directory.

The **cmake** optional parameter **-G "CodeBlocks - Unix Makefiles"** allows to generate the Code Blocks project and the Unix Makefiles.

The **cmake** optional parameter **-G "Xcode"** allows to generate the Xcode project.

## 6.4 Required packages on Windows platform

Here is a list of packages used in the FA $\mu$ ST project. Eigen and OpenBlas library are automatically installed : there are nothing to do (see the directory `./externals/win/`).

- **Eigen** is a C++ template library for linear algebra: matrices, vectors, numerical solvers, and related algorithms (see <http://eigen.tuxfamily.org>).
- **OpenBLAS** is an optimized BLAS library based on GotoBLAS2 1.13 BSD version. (see <http://www.openblas.net>). To install OpenBlas, refer to <https://github.com/xianyi/OpenBLAS/wiki/Installation-Guide>. You can directly download precompiled binary here <https://sourceforge.net/projects/openblas/files/v0.2.14/>

## 6.5 Matlab and processor architecture

If your processor architecture is 64 bit, Matlab must be installed in 64 bit. If your processor architecture is an 32 bit, Matlab must be installed in 32 bit. You can check your environment variable called `PROCESSOR_ARCHITECTURE` by typing :

```
1 > set
```

Check in the list the variable `PROCESSOR_ARCHITECTURE`.

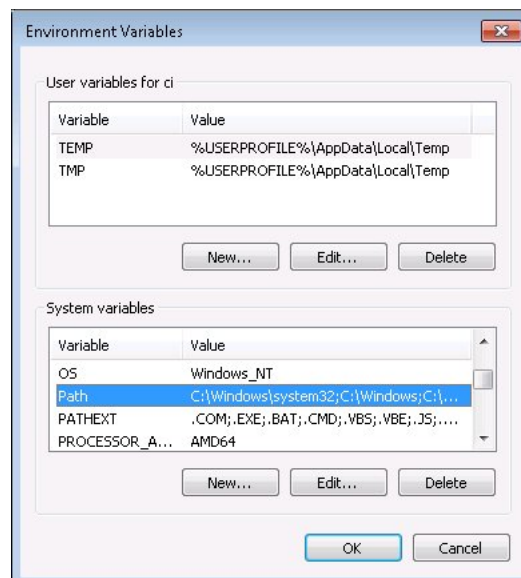
`PROCESSOR_ARCHITECTURE=AMD64` correspond to 64bit, `PROCESSOR_ARCHITECTURE=x86` correspond to 32 bit.

## 6.6 Add environment variable on Windows platform

Here is the steps to add an environment variable in Windows 7.

1. From the Desktop, right-click the Computer icon and select Properties. If you don't have a Computer icon on your desktop, click the Start button, right-click the Computer option in the Start menu, and select Properties.
2. Click the Advanced System Settings link in the left column.
3. In the System Properties window, click on the Advanced tab, then click the Environment Variables button near the bottom of that tab.
4. In the Environment Variables window (pictured below), highlight the Path variable in the "System variables" section and click the Edit button. Add or modify the path lines with the paths you want the computer to access. Each different directory is separated with a semicolon as shown below.

```
1 C:\Program Files;C:\Winnt;C:\Winnt\System32
```



## 6.7 FAUST Install on MAC OS X platform, using Xcode IDE with terminal command

You can generated the target using the terminal command `xcodebuild` :

```
1 > mkdir build
2 > cd build
3 > cmake .. -G "Xcode"
4 # list all target of the project
5 > xcodebuild -list -project FAUST.xcodeproj
6 # Build the targets
7 > xcodebuild -configuration "Release" -target "ALL_BUILD" build
8 # performs the "make install"
9 > xcodebuild -configuration "Release" -target "install" build
```



## 6.8 Optional FAuST installation using GPU process

The toolbox FA $\mu$ ST integrates optional Graphics Processing Unit (GPU) to improve its performances.

**Warning:** This optional GPU install has only be realized on an Linux machine. There is no guarantee that the installation and the use will be effective for every system.

- **Install** the CUDA Toolkit and the drivers for NVIDIA.
- **Verify install** of GPU tools by typing in a terminal :

```
1 > which nvcc
```

You must obtain the path of your `nvcc` compiler like

```
1 /usr/local/cuda-7.5/bin/nvcc
```

If not, add `nvcc` directory in your environment path (in your `/.bashrc` file).

# Bibliography

- [1] L. Le Magoarou and R. Gribonval. Flexible multilayer sparse approximations of matrices and applications. *IEEE Journal of Selected Topics in Signal Processing*, 10(4):688–700, June 2016.