

Getting Started with FA μ ST Toolbox
"Flexible Approximate Multi-Layer Sparse Transforms"



Adrien Leman; Nicolas Bellot

October 13, 2016

Contents

1	Introduction	2
2	Installation on Linux and MAC OS X platform	3
2.1	Required components	3
2.2	Basic Build & Installation	4
2.2.1	Install with administrator privilege	4
2.2.2	Install without administrator privilege	4
2.3	Basic Build & Install using Code Block IDE	5
2.3.1	Install with administrator privilege	5
2.3.2	Install without administrator privilege	5
2.4	Basic Build & Install using Xcode IDE (for MAC OS)	6
2.4.1	Install with administrator privilege	6
2.4.2	Install without administrator privilege	7
2.5	Custom - Advanced Installation	7
3	Installation on Windows platform	9
3.1	Required components	9
3.2	Install using GCC-MinGW compiler	10
3.2.1	Install GCC from MinGW	10
3.2.2	Basic Build & Install using the command prompt	11
3.2.3	Basic build & Install using Code::Blocks IDE	12
3.3	Install using Microsoft Visual compiler	14
3.3.1	Install Microsoft Visual compiler	14
3.3.2	Basic Build & Install using Visual Studio IDE	14
3.3.3	Basic Build & Install using terminal	15
3.4	Custom - Advanced Installation	16
4	QuickStart	19
4.1	Use a faust from a saved one	19
4.2	Other ways to build a Faust	20
4.2.1	build a Faust from a matrix and parameters	21
5	Example	23
5.1	Brain Sources Localization	23
6	Annexes	24
6.1	Required packages	24
6.2	Compatibility between MATLAB and compiler gcc	24
6.3	Further information about Build & Install process	25
6.4	Required packages on Windows platform	25
6.5	Add environment variable on Windows platform	25

Chapter 1

Introduction

Presentation: FA μ 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 $\mathbf{A} = \prod_{j=1}^J \mathbf{S}_j$.

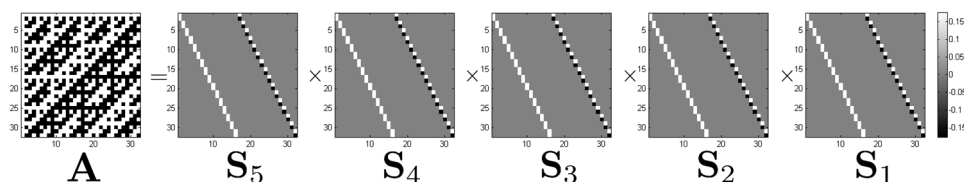


Figure 1.1: presentation

FA μ 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 μ 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/>.

Organization: Chapter 2 explains how to install the library FA μ 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 Linux and MAC OS X platform

The FA μ 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 project FA μ ST 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 μ ST tools on Unix platform (both Linux and Mac OS).

Firstly, please ensure that the **prerequisites components** listed in Section 2.1 are installed. 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 2.2.
- Basic installation using **the IDE "Code::Blocks"**, refer to Section 2.3.
- Basic installation using **the IDE "Xcode"** only for MAC OS X platform, refer to Section 2.4.

Finally in Section 2.5, the configure options available to build the FA μ ST toolbox are described to propose an Custom - Advanced installation. For example, the optional configuration can be used to modify the install directory path, or to build in debug mode.

2.1 Required components

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

- **Install CMake.** 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** (<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).

2.2 Basic Build & Installation

When prerequisites listed in precedent section 2.1 are checked, the FA μ ST installation can be done. If you are administrator of your machine (root access), follow instructions given in Section 2.2.1. Otherwise, for local installation, refer to Section 2.2.2.

2.2.1 Install with administrator privilege

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

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

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

2.2.2 Install without administrator privilege

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

```

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

```

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

2.3 Basic Build & Install using Code Block IDE

When prerequisites listed in precedent section 2.1 are checked, the FA μ ST installation can be done. This Build & Install section requires that you have the IDE Code::Blocks installed on your system. 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

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

```

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

```

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

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

2.3.2 Install without administrator privilege

- **Download** the FA μ ST package on the website : <http://faust.gforge.inria.fr/>
- **Unzip** the FA μ ST package into your FA μ ST directory.
- **Open** a command terminal

- **Set the current directory** to your FA μ ST directory (NOTE: do not use any special character in your FA μ ST directory path, for example the character μ)
- 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 μ 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.

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

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

FA μ ST install using the IDE Xcode concerns only MAC OS X environment.

When prerequisites listed in precedent section 2.1 are checked, the FA μ ST installation can be done. 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 2.4.1. Otherwise, for local installation, refer to Section 2.4.2.

2.4.1 Install with administrator privilege

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

```

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

```

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

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

2.4.2 Install without administrator privilege

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

```

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

```

- Open the FA μ 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.

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

2.5 Custom - Advanced Installation

The project FA μ ST can be configured with optional parameters, for example if you want to install FA μ 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. When using the `ccmake` command in your build directory, the Cmake GUI appears in the console (see fig. 2.1).

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.

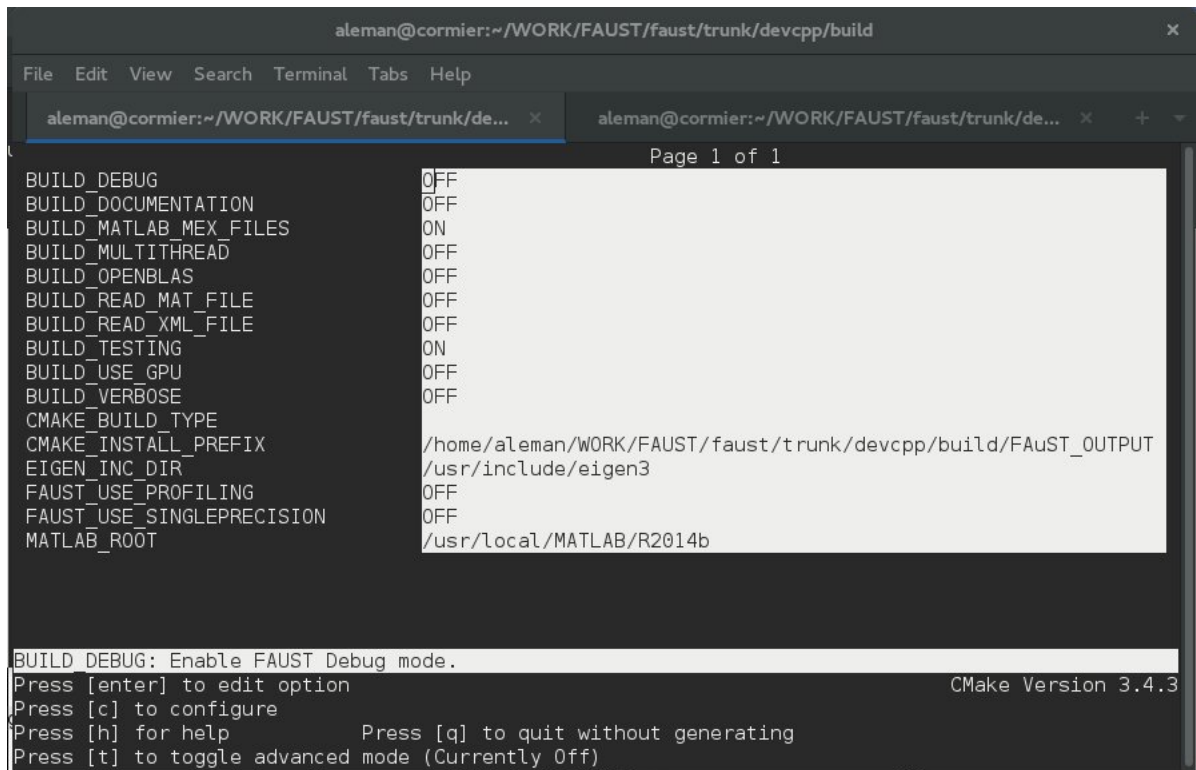


Figure 2.1: ccmake GUI

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 μ ST Debug mode (default value is OFF)
- `BUILD_USE_GPU` : Using both CPU and GPU process (default value is OFF)
- `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)
- `BUILD_READ_XML_FILE` : Using xml2 library to read xml files (default value is OFF)
- `BUILD_READ_MAT_FILE` : Using matio library to read mat files (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 μ 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 project FA μ ST 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 μ ST tools on Windows platform.

- Firstly, please ensure that the **prerequisites components** listed in Section 3.1 are installed.
- Secondly, **choose your preferred compiler** between GCC from MinGW "Minimalist GNU for Windows"(Section 3.2) or from Microsoft Visual Studio (Section 3.3).
- Then, **process to Basic installation** of FA μ 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 3.4 describes the configure options available to build the FA μ ST toolbox, to propose an Custom - Advanced installation. For example, the optional configuration can be used to modify the install directory path, or to build in debug mode.

3.1 Required components

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

- **Install CMake:** 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, verify that the CMAKE install directory is without any space character. Otherwise, please add the directory of your `cmake.exe` file in your environment variable. (to add an environment variable, follow instructions given in 6.5).

- **Install Matlab** (<https://fr.mathworks.com/downloads/>). You must install Matlab in the default directory like:

```
1 C:\Program Files\MATLAB
2 C:\Program Files (x86)\MATLAB
```

or install Matlab in a directory without any space character.

- **Verify Matlab install** by checking the `matlab.exe` binary file in the default directory, for example in

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

In the case of an installation in a directory without any space character, type in a terminal the following command :

```
1 > where matlab
```

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

```
1 C:\ProgramFiles\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 6.5).

- **Install 7-zip tool** from <http://www.7-zip.org/> in a install directory without any space character.
- **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\7-Zip\7z.exe
```

If not, verify that the 7-Zip install directory is without any space character. Otherwise, add `7z.exe` directory in your environment path. (to add an environment variable, follow instructions given in 6.5).

3.2 Install using GCC-MinGW compiler

When prerequisites listed in previous section 3.1 are checked, you must install **MinGW containing the C++ Compiler**. Then, the FA μ ST installation can be done using Command prompt, following instructions given in Section 3.2.2 or using CodeBlocks IDE, following instructions given in Section 3.2.3.

3.2.1 Install GCC from MinGW

- **Download MinGW** in <https://sourceforge.net/projects/mingw/files/latest/download?source=files>

- **Launch install file** and choose MinGW version 4.9.2 for mex tool compatibility. The install directory must be **without any space character**.
- Add `gcc.exe` directory in your environment path. (to add an environment variable, follow instructions given in 6.5).
- **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-w64\mingw64\bin\gcc.exe
```

If not, add `gcc.exe` directory in your environment path. (to add an environment variable, follow instructions given in 6.5).

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

```
1 > where make
```

You must obtain the path of your `make.exe` binary file. If not, verify that the MINGW install directory is without any space character. Otherwise, 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. For that, you must have a id session for Mathwork (easy to create). Current this latest step, an environment variable called `MW_MINGW64_LOC` is automatically generated.

3.2.2 Basic Build & Install using the command prompt

The basic install described in this section requires the GCC compiler from MinGW (see precedent Section 3.2.1). If you have administrator privilege, follow instructions in Section 3.2.2.1, otherwise follow instructions given in Section 3.2.2.2.

3.2.2.1 Install with administrator privilege

- **download the FA μ ST package** on the website <http://faust.gforge.inria.fr/>.
- **Unzip** the FA μ ST package into your FA μ ST directory.

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

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

```

1 > make install

```

3.2.2.2 Install without administrator privilege

- **download the FA μ ST package** on the website <http://faust.gforge.inria.fr/>.
- **Unzip** the FA μ ST package into your FA μ ST directory.

```

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

```

- Open a command terminal
- Set the current directory to your FA μ ST directory (NOTE: don't use any special character in your FAUST directory path, for example the character μ)
- 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

```

3.2.3 Basic build & Install using Code::Blocks IDE

The basic install described in this section requires the GCC compiler from MinGW (see precedent Section 3.2.1). If you have administrator privilege, follow instructions in Section 3.2.3.1, otherwise follow instructions given in Section 3.2.3.2.

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

Then, the FA μ ST installation can be done.

3.2.3.1 Install with administrator privilege

- **download the FA μ ST package** on the website <http://faust.gforge.inria.fr/>.
- **Unzip** the FA μ ST package into your FA μ ST directory.

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

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

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

- Open the FA μ ST project from the file ./build/FAUST.cbp with Code::Blocks IDE.
- In Code::Blocks IDE, select ALL target and build the project.
- Re-Open the FA μ 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 and build the project. For more detail about cmake commands, refer to Section 6.3.

3.2.3.2 Install without administrator privilege

- **download the FA μ ST package** on the website <http://faust.gforge.inria.fr/>.
- **Unzip** the FA μ ST package into your FA μ ST directory.

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

- Open a command terminal
- Set the current directory to your FA μ ST directory (NOTE: don't use any special character in your FAUST directory path, for example the character μ)
- 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 μ 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. For more detail about cmake commands, refer to Section 6.3.

3.3 Install using Microsoft Visual compiler

When prerequisites listed in section 3.1 are checked, you must install **Microsoft Visual Studio** containing the C++ Compiler (Section 3.3.1). Then, the FA μ ST installation can be done using Command prompt, following instructions given in Section 3.2.2 or using Visual Studio IDE, following instructions given in Section 3.2.3.

3.3.1 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 μ ST installation can be done with Visual Studio IDE, following instructions given in Section 3.3.2 or with a terminal (Section 3.3.3).

3.3.2 Basic Build & Install using Visual Studio IDE

The basic install described in this section requires Microsoft Visual Studio (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

- **download the FA μ ST package** on the website <http://faust.gforge.inria.fr/>.
- **Unzip** the FA μ ST package into your FA μ ST directory.

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

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

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

- Open the FA μ ST project from the file `./build/FAUST.sln` with Visual Studio IDE.
- In Visual Studio IDE, select ALL target and generate the project three times.
- Re-Open the FA μ ST project from the file `./build/FAUST.sln` with Visual Studio IDE with **administrator privilege**.
- In Visual Studio IDE, select install target and generate the project. For more detail about cmake commands, refer to Section 6.3.

3.3.2.2 Install without administrator privilege

- download the FA μ ST package on the website <http://faust.gforge.inria.fr/>.
- **Unzip** the FA μ ST package into your FA μ ST directory.

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

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

```
1 > mkdir build
2 > cd build
3 > cmake .. -DCMAKE\ _INSTALL\ _PREFIX="<Your/Install/Dir>"
```

- Open the FA μ ST project from the file `./build/FAUST.sln` with Visual Studio IDE.
- In Visual Studio IDE, select ALL target and generate the project three times.
- In Visual Studio IDE, select install target and generate the project. For more detail about cmake commands, refer to Section 6.3.

3.3.3 Basic Build & Install using terminal

The basic install described in this section requires the Microsoft Visual Studio compiler (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.

3.3.3.1 Install with administrator privilege

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

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

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

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


3.3.3.2 Install without administrator privilege

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

- Open a command terminal
- Set the current directory to your FA μ ST directory (NOTE: don't use any special character in your FAUST directory path, for example the character μ)
- 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"
```

3.4 Custom - Advanced Installation

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

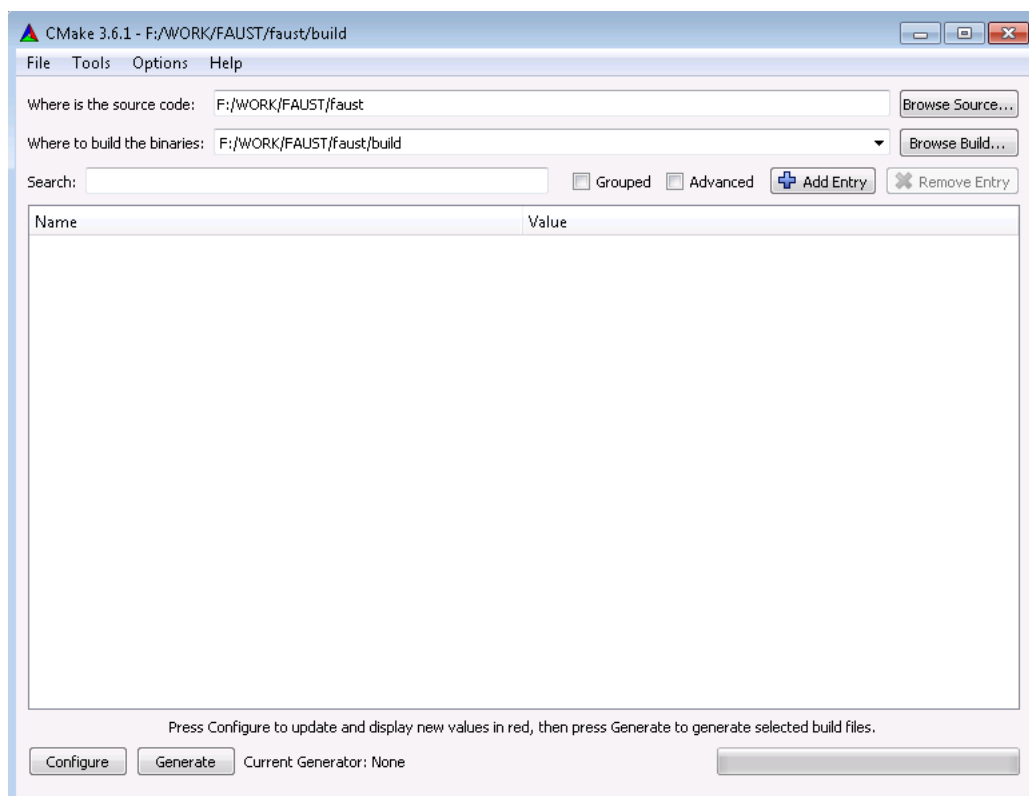


Figure 3.1: cmake GUI

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).

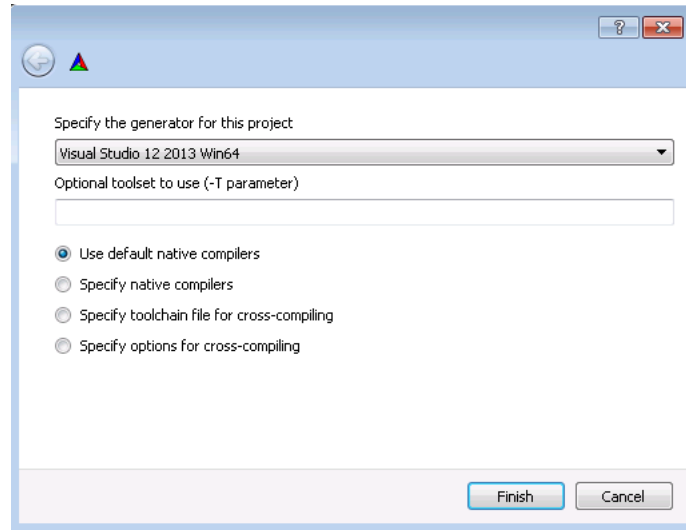


Figure 3.2: cmake GUI

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 3.4 what to do in such a case, but let us for the moment assume that everything ran smoothly. (see 3.3).

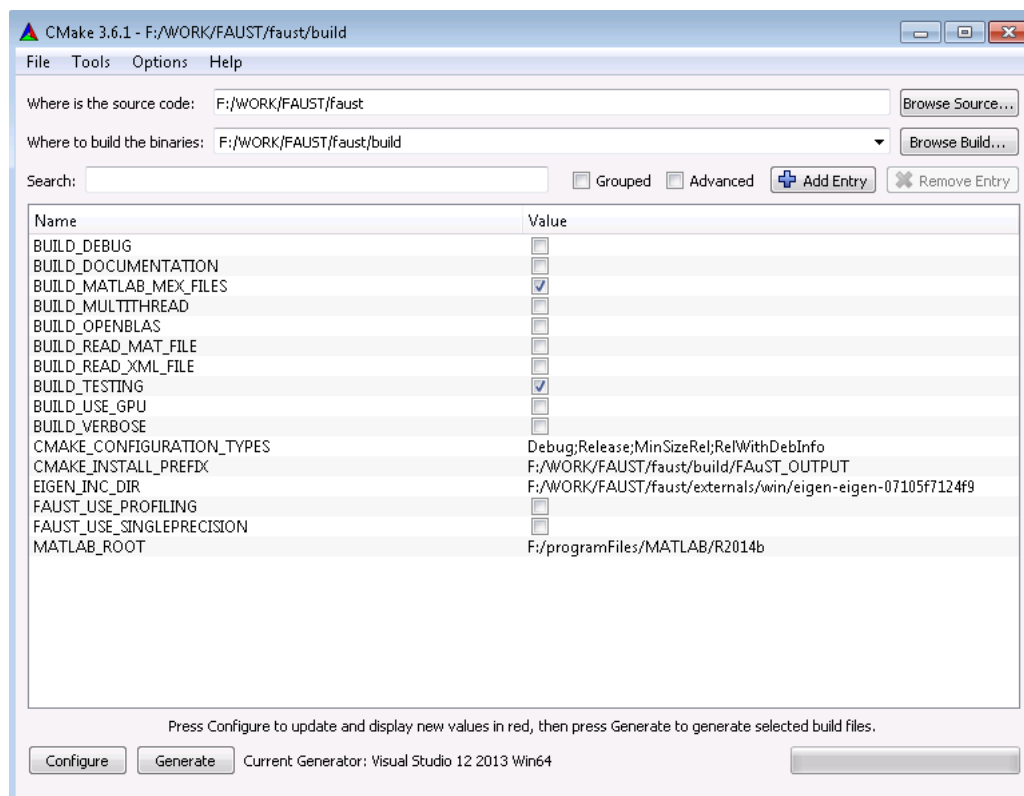


Figure 3.3: cmake GUI

Chapter 4

QuickStart

A Matlab wrapper is delivered with the FA μ ST C++ library. It provides a user friendly new class of matrix **Faust** efficient for the multiplication with matlab built-in dense matrix class. In order to use matlab wrapper after the installation of Faust, launch Matlab. In the Matlab terminal, set your working directory to `/"HOMEDIR"/Documents/MATLAB/Faust` and set the Matlab path by typing the following commands :

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

4.1 Use a faust from a saved one

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

```
1 >> quick_start
```

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

```
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 % Warning : A(i,j)=3 will not modify A, writing is not allowed

```

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).

```

1 %% speed-up multiplication
2 nb_mult=100;
3 time_full=0;
4 time_faust=0;
5
6 for i=1:nb_mult
7     tic
8     y=A_full*x1;
9     time_full=time_full+toc;
10
11     tic
12     y=A*x1;
13     time_faust=time_faust+toc;
14 end
15
16 disp('multiplication SPEED-UP using Faust');
17 disp(['Faust is ' num2str(time_full/time_faust) ' faster than a
    full matrix']);

```

A **Faust** object can be constructed from several ways.

4.2 Other ways to build a Faust

A Faust can be initialized from a matrix or from a cell-array storing its factors.

4.2.1 build a Faust from a matrix and parameters

First, you can build a Faust from a cell-array of matlab matrix (sparse or dense) representing its factors.

The following example shows how to build a random faust of size 5x3 with 3 factors :

```
1 >> nb_factor = 3;
2 >> dim1 = 5;
3 >> dim2 = 3;
4 >>
5 >> % list of factors
6 >> factors = cell(nb_factor);
7 >>
8 >> factors{1}=sprand(dim1,dim2,0.1); % first factor is rectangular
   and sparse
9 >> for i=2:nb_factor
10 >>     factors{i}=rand(dim2,dim2); % second is sparse
11 >> end
12 >>
13 >> % build the faust
14 >> A=Faust(factors);
```

An optional multiplying scalar argument can be taken into account :

```
1 >> % multiplicative scalar
2 >> lambda = 3.5
3 >> % build the faust
4 >> A=Faust(factors,lambda);
```

This functionality allows you to build a Faust from the factorization algorithm : **mexHierarchical_fact** or **mexPalm4MSA** :

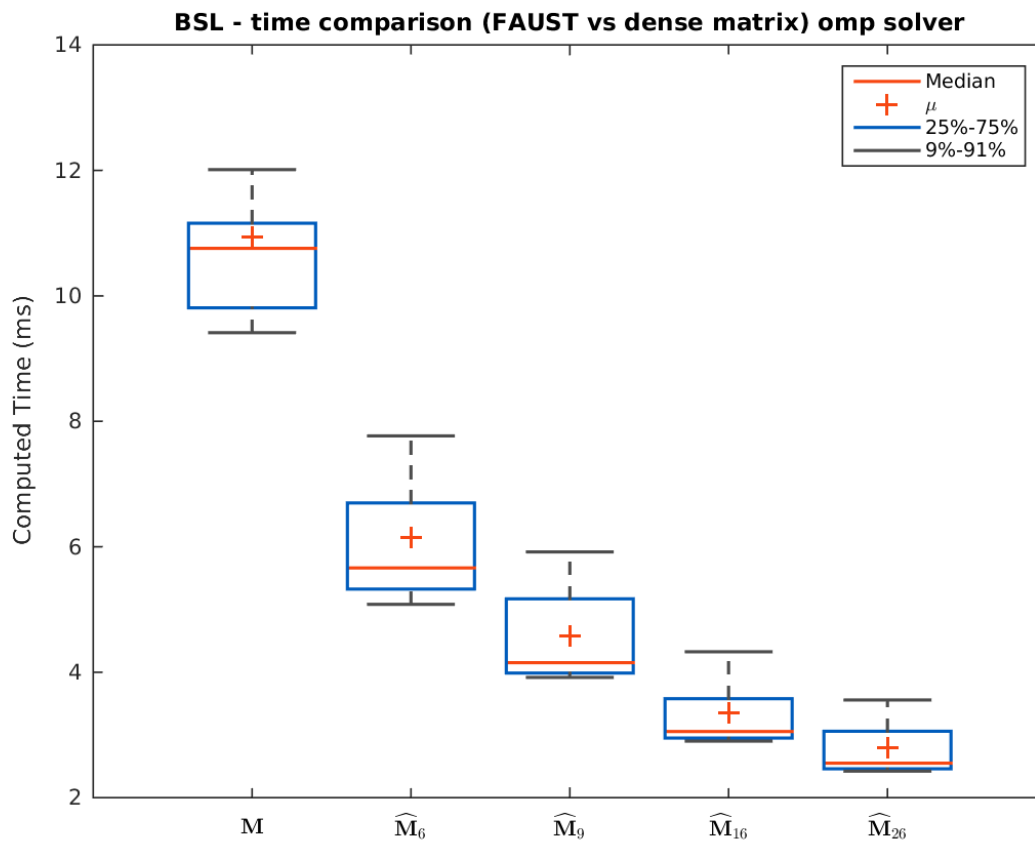
```
1 >> % factorization step
2 >> [lambda,factors]=mexHierarchical_fact(params);
3 >> % build the faust corresponding to the factorization
4 >> A=Faust(factors,lambda);
```

Chapter 5

Example

5.1 Brain Sources Localization

An experience of Brain Source Localization using several gain matrices including FAuSTs and several solvers is provided. You can execute the matlab script `demo/Brain_source_localization/BSL.m` to run this experiment and `demo/Brain_source_localization/Fig_BSL.m` to display the following pictures illustrating the speed-up using a Fajust.



Chapter 6

Annexes

6.1 Required packages

Here is a list of packages used in the FA μ 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>
- Library openBlas <http://www.openblas.net>
- Library xml2 <http://xmlsoft.org>
- Library matio <https://sourceforge.net/projects/matio>

6.2 Compatibility between MATLAB and compiler gcc

Compatibility between MATLAB and compiler gcc 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/compiler/R2016a/index.html?sec=glxa64> for more detail). To temporally change your version of gcc compiler, you can modify the environment variable CC an CXX:

- **export 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++
```

You can change the version of compiler use by mex function from matlab command:

```
1 > mex -setup
```

6.3 Further information about Build & Install process

Further information: 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>
```

Further information: The command **make** will compile the build files.

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

/usr/local/lib/libfaust.a for the FA μ 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 μ 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/>

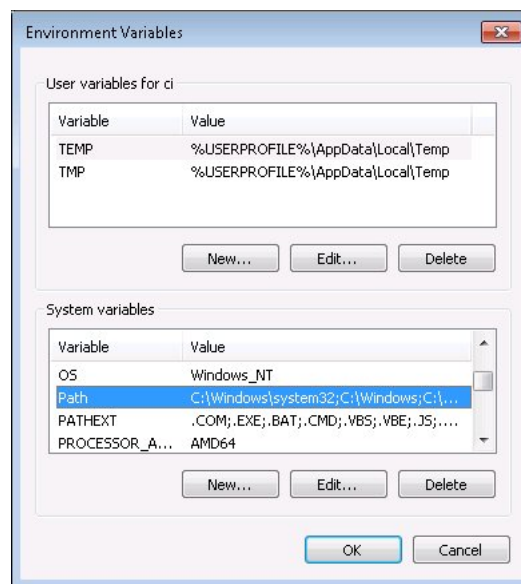
6.5 Add environment variable on Windows platform

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



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.