Getting Started with the FAμST Toolbox version 2.1 "Flexible Approximate Multi-Layer Sparse Transforms "

Adrien Leman; Nicolas Bellot

April 24, 2017

Contents

Chapter 1

Introduction

What is the FAμST toolbox ? FAμST is a C++ toolbox, useful to decompose or approximate any given dense matrix into a product of sparse matrices, in order to reduce its computational complexity (both for storage and manipulation). As an example, Figure 1.1 shows a dens matrix $\mathbf{A}$ and sparse matrices $\mathbf{S_j}$ such that $\mathbf{A} = \prod_{j=1}^{J} \mathbf{S_j}$.

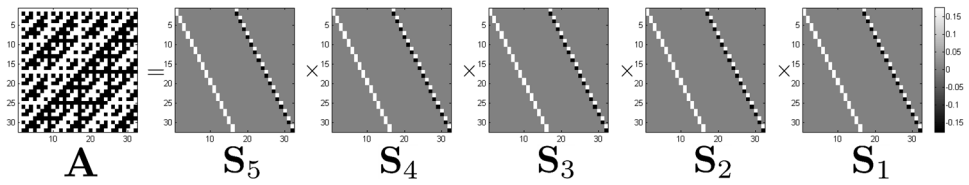


Figure 1.1: Brief presentation of FAμST

Why shall I use FAμST ? FAμST 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 [**?**]. For more information on the FAμST project, please visit the website of the project: http://faust.gforge.inria.fr.

Is my platform and compiler currently supported ? FAμST has been tested in various configurations, and is currently delivered with a Matlab wrapper. Figure 1.2 summarizes the configurations on which FAμST has been tested.

How should I use this document ? If your platform and configuration is currently supported according to Figure 1.2, please refer to the corresponding Install Chapter. By default we suggest the installation using the GCC or Clang compiler directly from a command terminal since it requires fewer externals components.

- for installation on UNIX (including Linux and Mac OS X) platform refer to Chapter 2;
- for installation on Windows refer to Chapter 3.

Chapter 4 shows quickly how to use this library and gives an example of application. Finally, a "work in progress" part is given Chapter 5 to give an overview of the roadmap for FAμST.

Config / Platform	Compiler	Matlab	Python	Integrated Development Environment (IDE)	
LINUX	GCC (Version < v5.0)	Matlab (V > 2013)	Python (v = 2.x)	Code ::Blocks (with GCC)	
MAC OS X	Clang	Matlab (V > 2013)	Python (v = 2.x)	Code ::Blocks (with Clang)	Xcode
WINDOWS	GCC (MinGW Version 4.9 & 5.3)	Matlab (V > 2013)	Python	Code ::Blocks (with GCC)	Visual Studio

Figure 1.2: Tested configurations: Platforms and IDE

How is the code of FAμSTstructured ? A brief outline of the code structure and status is given on Figure 1.3. The main C++ library called libfaust includes two components:

- the "FAμST matrix multiplication" provides efficient linear operators to efficiently manipulate your data, relying on efficient low-level routines from external libraries where needed;
- the "Factorization algorithms" is used to generate a FAμST core from a dense matrix.

Various wrappers are planned to call libfaust. In the current version of FAμST (Version 2.1), only the Matlab wrapper and the Python wrapper is implemented. Command-line and A||GO wrapper are planned for the next version of FAμST (see Section 5).

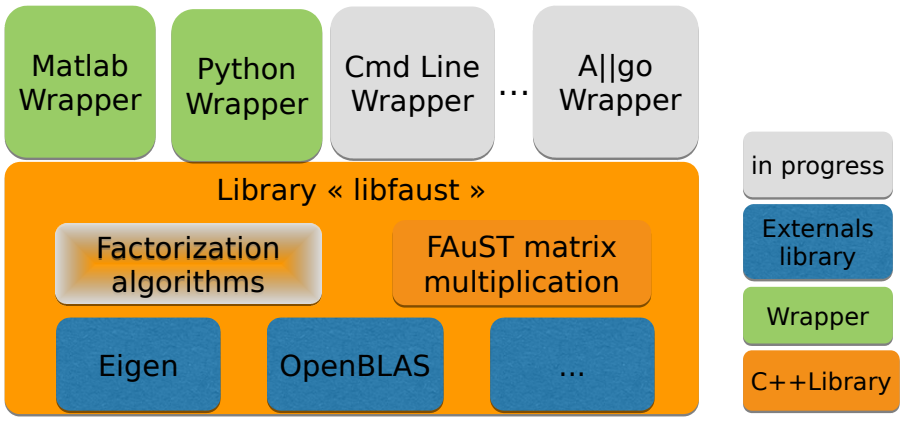


Figure 1.3: Brief structure of FAμST version 2.1.0

Known issues :

1. The installation on Windows using Visual Studio IDE has been tested with success in certain configurations, however **compilation problems have been encountered depending on the version of Windows and Visual Studio**. Unfortunately we cannot offer technical support but you may nevertheless report problems and/or suggestions on the mailing list http://lists.gforge.inria.fr/pipermail/faust-install/.

2. Compiling mex files with the **mex compiler** of Matlab requires a compatible third-party compiler. **Warning: With Matlab Release2016a, the mex compiler seems to only support up to GCC 4.7 (see** http://fr.mathworks.com/support/compilers/R2016a/index.html?sec=glnxa64 **for more detail)**.

3. The "Factorization algorithms" module represented on Figure 1.3 is still work in progress (see Section 5). A GPU implementation is also on its way, as well as more wrappers.

How do I pronounce FAμST ? We suggest to pronounce the library name as "FAUST", but you may also pronounce it "FAMUST" ;-)

1.1 New Features since last version

Here are presented somes new features that are incorporated in the current version (2.1) and weren't available in the former version.

Python wrapper The a beta version of the Python wrapper is now available. The list of components you musthave to use this wrapper is defined in the following section (5.1).

Matlab wrapper : complex scalar compatibility The Matlab wrapper is now compatible with complex scalar. This allows you to construct Faust from complex factors and multiply with complex matrices. Nevertheless, the transconjugate and conjugate operation are not yet implemented.

C++ code : improvement Some improvement have been made to improve the speed-up of a Faust. For instance, a factor of a Faust is no longer necessarily a sparse matrix.Now, it's a now generic matrix. This means that currently it can be a full-storage matrix or sparse one. The choice of the format that will represent the factor is made at execution time for more flexibility. Another advantage, this for future development, the class/interface of generic matrices can be inherited/implemented by more specific format (block-diagonal matrices, permutation matrices) or more efficient.

Chapter 2

Installation on a Unix platform

The FAμST project is available for Linux, MAC OS X and Windows platforms. The proposed toolbox provides a Matlab wrapper. **CMake** has been chosen to build the FAμST 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 FAμST tools on a Unix platform (both Linux and Mac OS).

1. Ensure that the **prerequisite components** listed in Section 2.1 are installed.
2. Refer to section 2.2 to download the FAμST package and set your terminal.
3. 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 Code::Blocks IDE**, refer to Section 2.4.
   - Basic installation using **the Xcode IDE** only for MAC OS X platform, refer to Section 2.5.

Section 2.6 describes the custom options available to build the FAμST toolbox, to propose a Custom - Advanced installation. For example, the optional configuration can be used to modify the install directory path, or to build in debug mode.

Throughout the document we provide examples as follows:

- Command lines you must enter in a terminal are displayed as:

```
> mkdir BUILD; ls .
```

- Messages resulting from such command lines appear without the leading '>':

```
example of return text in your current terminal.
```

- Command lines you must enter in a Matlab Command Window are displayed as:

```
>> y = A*x;
```

- Messages resulting from such command lines appear without the leading '»':

```
example of return text in Matlab Command Window.
```

2.1 Required components

This Section lists the required components you must install before to begin the FAμST installation.

- **Install CMake**. The minimum version required is Cmake version 3.0.2. Download binary distribution of CMake from their website https://cmake.org/download/.
- **Verify Cmake install** by typing in a command terminal :

```
> which cmake
```

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

```
/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 :

```
> which matlab
```

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

```
/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 :

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

You must obtain this kind of message :

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

But if you have an error message of the style :

```
No supported compiler or SDK was found. For options, visit
http:www.matworks.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 plateform `<ARCH>`.

2.2 Download FAuST Package & launch terminal

When prerequisities listed in precedent section 2.1 are checked, you can get the package FAμST.

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

2.3 Basic Build & Installation using Makefile

When you have done the step in section 2.2 (i.e download FAμST package and launch the terminal in the right directory), the FAμST 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 :

```
> mkdir build
> cd build
> cmake ..
> make
> sudo make install # run with administrator privilege
```

FAμST Toolbox should be installed. Now, refer to Quick-Start Chapter 4 to check the install and to try FAμST 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 :

```
> mkdir build
> cd build
> cmake .. -DCMAKE_INSTALL_PREFIX="<Your/Install/Dir>"
> make
> make install
```

FAμST Toolbox should be installed. Now, refer to Quick-Start Chapter 4 to check the install and to try FAμST 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 FAμST package and launch the terminal in the right directory), the FAμST 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 :

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

- Open the FAμST 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 FAμST project from the file **./build/FAUST.cbp** with Code::Blocks IDE **with administrator privilege**. For that, type in a terminal :

```
> sudo codeblocks
```

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

FAμST Toolbox should be installed. Now, refer to Quick-Start Chapter 4 to check the install and to try FAμST 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 :

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

- Open the FAμST 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.

FAμST Toolbox should be installed. Now, refer to Quick-Start Chapter 4 to check the install and to try FAμST toolbox.

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

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

FAμST install using the Xcode IDE concerns only MAC OS X environment.

When you have done the step in section 2.2 (i.e download FAμST package and launch the terminal in the right directory), the FAμST installation can start. This Build & Install section requires that you have the Xcode IDE 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 :

```
> mkdir build
> cd build
> cmake .. -G "Xcode"
```

- Open the FAμST 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 FAμST project from the file **./build/FAUST.xcodeproj** with Xcode IDE **with administrator privilege**. For that, type in a terminal:

```
> sudo Xcode
```

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

FAμST Toolbox should be installed. Now, refer to Quick-Start Chapter 4 to check the install and to try FAμST 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 :

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

- Open the FAμST 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.

FAμST Toolbox should be installed. Now, refer to Quick-Start Chapter 4 to check the install and to try FAμST toolbox.

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

2.6 Custom - Advanced Installation

The project FAμST can be configured with optional parameters, for example if you want to install FAμST 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 :

```
> ccmake ..
```

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

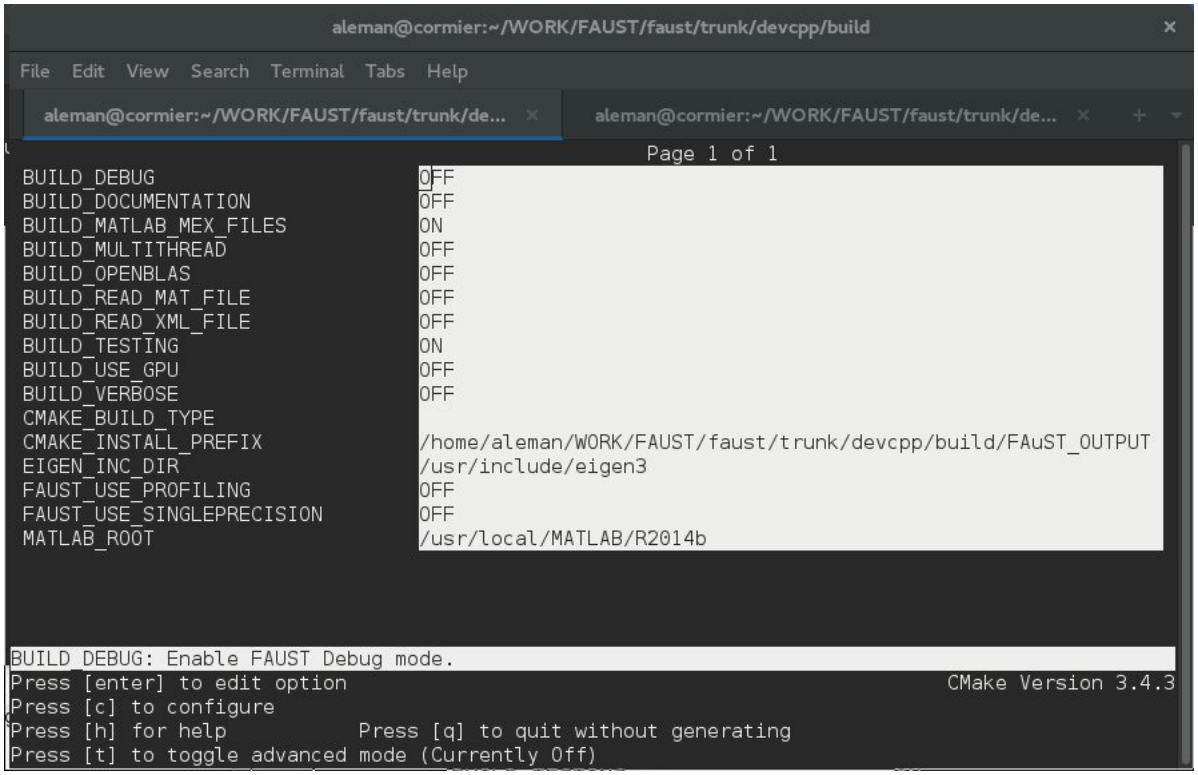


Figure 2.1: ccmake GUI

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 path for the FAμST library
- CMAKE_INSTALL_MATLAB_PREFIX : Install directory path for the Matlab wrapper

- 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 FAμST Debug mode (default value is OFF)
- BUILD_USE_GPU : Using both CPU and GPU process (default value is OFF) (refer to Section 5.3 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).

Chapter 3

Installation on a Windows platform

The FAμST project is available for Linux, MAC OS X and Windows platforms. The proposed toolbox provides a Matlab wrapper. **CMake** has been chosen to build the FAμST 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 FAμST tools on a Windows platform.

1. Ensure that the **prerequisite components** listed in Section 3.1 are installed.
2. **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).
3. **Process to the Basic installation** of FAμST by following the 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 FAμST 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.

Throughout the document we provide examples as follows:

- Command lines you must enter in a terminal are displayed as:

```
> mkdir BUILD; ls .
```

- Messages resulting from such command lines appear without the leading '>':

```
example of return text in your current terminal.
```

- Command lines you must enter in a Matlab Command Window are displayed as:

```
>> y = A*x;
```

- Messages resulting from such command lines appear without the leading '»':

```
example of return text in Matlab Command Window.
```

3.1 Required components

The installation of the FAμST 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 distribution of CMake from their website https://cmake.org/download/.
- **Verify CMake install** : Open a terminal and type the following command:

```
> 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 :

```
> where matlab
```

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

```
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 :

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

  You must obtain this kind of message :

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

  But if you have an error message of the style :

```
No supported compiler or SDK was found. For options, visit
http:www.matworks.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 plateform `<ARCH>`.

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

```
> where 7z
```

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

```
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 prerequisitie components listed in precedent section 3.1 are checked, you can get the FAμST package.

- **Download** the FAμST package on the website : http://faust.gforge.inria.fr/
- **Unzip** the FAμST package into your FAμST directory.

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

3.3 Install using GCC compiler

When prerequisite components listed in previous section 3.1 are checked, you must install **MinGW containing the C++ Compiler**. Then, the FAμST 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 compiler 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 :

```
> where gcc
```

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

```
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

```
> 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. 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 FAμST directory (NOTE: don't use any special character in your FAμST directory path, for example the character μ)
- Type the following commands :

```
> mkdir build
> cd build
> cmake -G "MinGW Makefiles" ..
> 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 FAμST directory
- Type the following commands :

```
> make install
```

FAμST Toolbox should be installed. Now, refer to Quick-Start Chapter 4 to check the install and to try FAμST toolbox.

3.3.2.2 Install without administrator privilege

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

```
> mkdir build
> cd build
> cmake -G "MinGW Makefiles" ..
        -DCMAKE_INSTALL_PREFIX="<Your/Install/Dir>"
> make
> make install
```

FAμST Toolbox should be installed. Now, refer to Quick-Start Chapter 4 to check the install and to try FAμST 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 code::Blocks IDE in the
`Settings menu -> Compiler...  -> Selected Compiler`
Then, the FAμST installation can be done.

3.3.3.1 Install with administrator privilege

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

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

- Open the FAμST 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 FAμST 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.

FAμST Toolbox should be installed. Now, refer to Quick-Start Chapter 4 to check the install and to try FAμST 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 FAμST directory (NOTE: don't use any special character in your FAμST directory path, for example the character μ)
- Type the following commands :

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

- Open the FAμST 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.

FAμST Toolbox should be installed. Now, refer to Quick-Start Chapter 4 to check the install and to try FAμST 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 FAμST 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 FAμST 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 FAμST 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 FAμST directory (NOTE: don't use any special character in your FAμST directory path, for example the character μ)
- Type the following commands :

```
> mkdir build
> cd build
> cmake ..
```

- Open the FAμST 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 FAμST 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.

FAμST Toolbox should be installed. Now, refer to Quick-Start Chapter 4 to check the install and to try FAμST 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 FAμST directory (NOTE: don't use any special character in your FAμST directory path, for example the character μ)
- Type the following commands :

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

- Open the FAμST 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.

FAμST Toolbox should be installed. Now, refer to Quick-Start Chapter 4 to check the install and to try FAμST 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 FAμST directory (NOTE: don't use any special character in your FAμST directory path, for example the character μ)
- Type the following commands :

```
> mkdir build
> cd build
> cmake ..
> cmake --build . --config "Release"
> cmake --build . --config "Release"
```

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

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

FAμST Toolbox should be installed. Now, refer to Quick-Start Chapter 4 to check the install and to try FAμST 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 FAμST directory (NOTE: don't use any special character in your FAμST directory path, for example the character μ)
- Type the following commands :

```
> mkdir build
> cd build
> cmake .. -DCMAKE_INSTALL_PREFIX="<Your/Install/Dir>"
> cmake --build . --config "Release"
> cmake --build . --config "Release"
> cmake --build . --config "Release" --target "install"
```

FAμST Toolbox should be installed. Now, refer to Quick-Start Chapter 4 to check the install and to try FAμST toolbox.

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

3.5 Custom - Advanced Installation

The project FAμST can be configured with optional parameters, for example if you want to install FAμST 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:

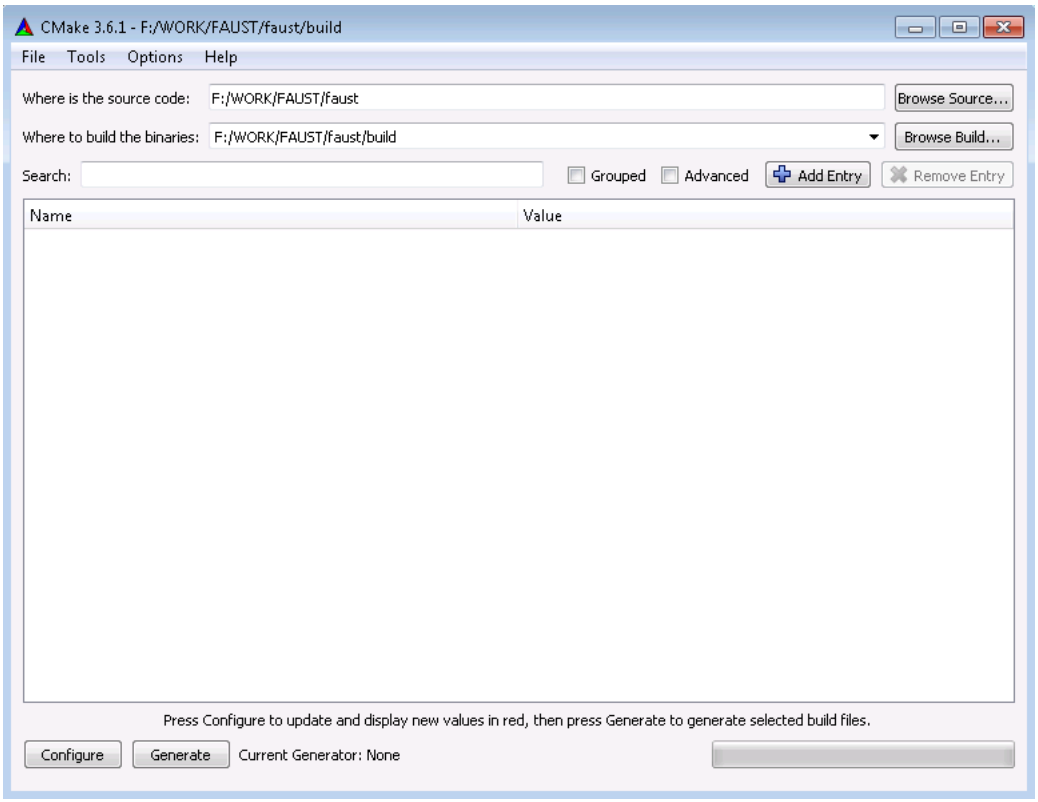


Figure 3.1: cmake GUI

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).

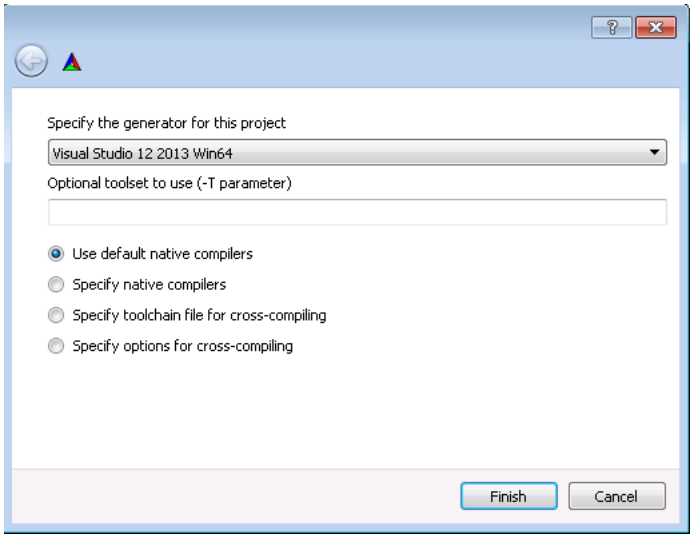


Figure 3.2: cmake GUI

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 appears 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).

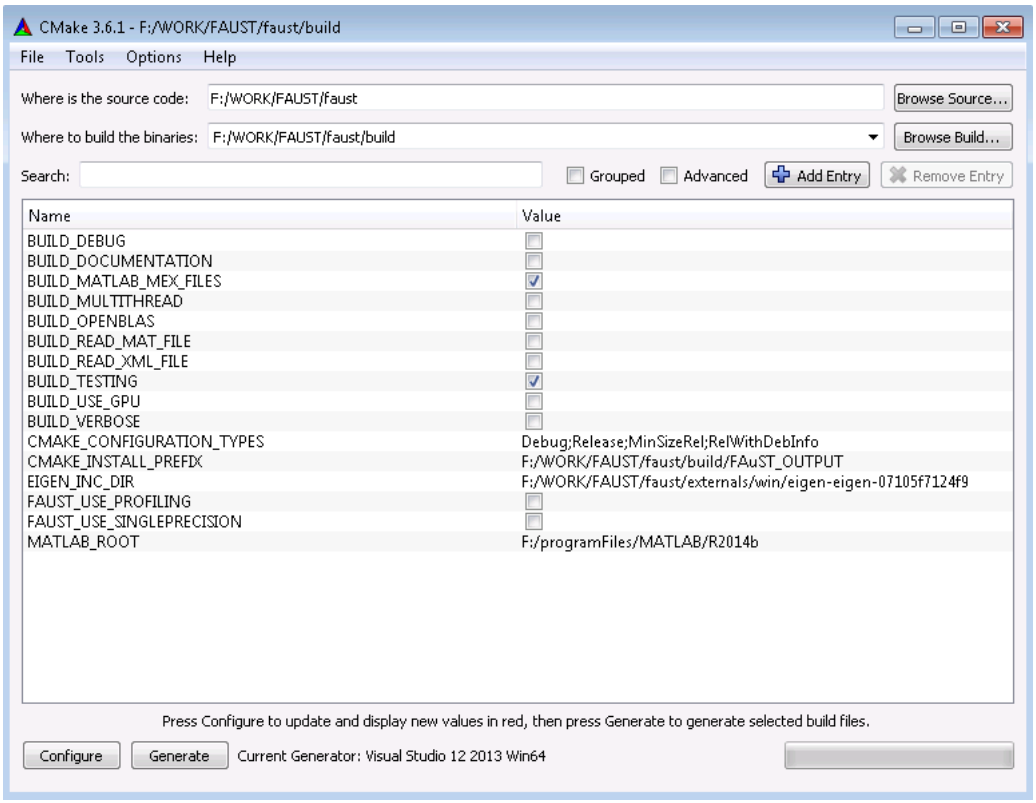


Figure 3.3: cmake GUI

Here is the list of available options:

- CMAKE_INSTALL_PREFIX : Install directory path for the FAμST library
- CMAKE_INSTALL_MATLAB_PREFIX : Install directory path for the Matlab wrapper
- 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 FAμST Debug mode (default value is OFF)
- BUILD_USE_GPU : Using both CPU and GPU process (default value is OFF) (refer to Annex 5.3 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 FAμST C++ library. It provides a user friendly new class of **FAμST** matrices for efficient multiplication with Matlab built-in dense matrix class. This chapter presents some demos in order to get accustomed to the **FAμST** class and illustrates the advantages of **FAμST** class on some examples.

1. **Configure your Matlab path** (refer to Section 4.1).
2. You can run and take a look at various Matlab demos :
   - **Use a FAμST from a saved one (Section 4.2)** presents the general functionality of a FAμST matrix
   - **Construct a FAμST from its factors (Section 4.3)**
   - **Brain Sources Localization (Section 4.4)** illustrates the speed-up induced by FAμST in a medical imaging application

4.1 Configure Matlab path

In order to use Matlab wrapper, follow the instructions :

1. **Install** FAμST tool (see Chapter 2 in case of Unix install or Chapter 3 in case of Windows install)
2. **Launch** Matlab.
3. **Set the working directory** of the Matlab Command Window to your FAμST install directory FAuST_INSTALL_DIR by typing :

```
1 >> cd <FAuST_INSTALL_DIR>
```

   By default, FAuST_INSTALL_DIR is inside the Matlab user directory. So, depending on your OS and configuration, you must replace the string <FAuST_INSTALL_DIR> in the above Matlab command line by this path :

   - **Windows** : `<ROOT>\Users\<USERNAME>\Documents\MATLAB\faust`
   - **Linux** : `/home/<USERNAME>/Documents/MATLAB/faust`
   - **Mac OS X** : `/Users/<USERNAME>/Documents/MATLAB/faust`

   You can also configure the FAuST_INSTALL_DIR to whatever you like (cf. CMake variable CMAKE_INSTALL_MATLAB_PREFIX section 2.6 for Unix or section 3.5 for Windows users)

4. **Configure** the Matlab path by typing the following commands :

```
>> setup_Faust
```

You must obtain the following message in your Matlab Command Window :

```
Welcome to the Matlab wrapper of the FAuST C++ toolbox.
FAuST root directory is <FAuST_INSTALL_DIR>
Adding path <FAuST_INSTALL_DIR> and all its subdirectories
To get started with the FAuST Toolbox : launch quick_start
or run_all_demo.m
```

4.2 Use a FAuST from a saved one

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

```
>> quick_start
```

The `quick_start.m` script is located in the following path :
`<FAuST_INSTALL_DIR>/demo/Quick_start/quick_start.m`
In this script :

1. A FAμST of size 4000x5000 is loaded from a MAT-file located in :
`<FAuST_INSTALL_DIR>/demo/Quick_start/faust_quick_start.mat`

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

2. A list of overloaded Matlab function shows that a FAμST is almost handled as a normal Matlab builtin matrix. The exception is that one cannot *assign* a value to a given entry.

```
[dim1,dim2] = size(A)

%  transpose a faust
A_trans = A'

% multiplication by A
x1 = rand(dim2,1);
y1 = A*x1;

% multiplication by A'
x2 = rand(dim1,5);
y2 = A'*x2;


% get the 2-norm (spectral norm) of the faust A
norm_A = norm(A) % equivalent to norm(A,2);

```

```
% convert Faust to full matrix
A_full=full(A);

% get the number of non-zeros coefficient
nz = nnz(A)

% READING coefficient
coeff=A(3,4)
col_2=A(:,2);
submatrix_A=A(3:5,2:3)
submatrix_A=A(end-5:end,4980:end-1)

% WARNING : WRITING coefficient NOT ALLOWED
% A(i,j)=3; will throw an error with this message :
%            'function not implemented for Faust class'
```

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

```
%% speed-up multiplication
nb_mult=100;
time_full=0;
time_faust=0;

for i=1:nb_mult
   tic
      y=A_full*x1;
   time_full=time_full+toc;

   tic
      y=A*x1;
   time_faust=time_faust+toc;
end

disp('multiplication SPEED-UP using Faust');
disp(['Faust is ' num2str(time_full/time_faust) ' faster than
   a full matrix']);
```

4.3 Construct a FAuST from its factors

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

```
>> construct_Faust_from_factors
```

The following example shows how to build a FAμST from a cell-array representing its factors.

```
% number of row of the Faust
```

```
dim1=300;
% number of column of the Faust
dim2=100;
% number of factor of the Faust
nb_factor=4;
% density of each factor
density_factor=0.1;

%cell-array representing its factors
factors = cell(1,nb_factor);

% 1st factor is a rectangular sparse matrix of density equal to
   density_factor
factors{1} = sprand(dim1,dim2,density_factor);

% all the others factor are square sparse matrix of density equal
   to density_factor
for i=2:nb_factor
  factors{i} = sprand(dim2,dim2,density_factor);
end

%% construct the Faust
A_faust=Faust(factors);


% a multiplicative scalar can be taken into account to construct
   the Faust
lambda=2;

% B_faust=lambda*A_faust
B_faust=Faust(factors,lambda);
```

If you want to build a FAμST from a given matrix, you can go to the section Work in Progress 5.2. Be aware that this not yet stable but under development.

4.4 Brain Sources Localization

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

```
>> BSL
>> Fig_BSL
```

`BSL.m` runs the experiment (see the script file : `<FAuST_INSTALL_DIR>/demo/Brain_source_localization/BSL.m`)
`Fig_BSL.m` displays Figure 4.1. illustrating the speed-up using a FAμST. (see the script file: `<FAuST_INSTALL_DIR>/demo/Brain_source_localization/Fig_BSL.m`)

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

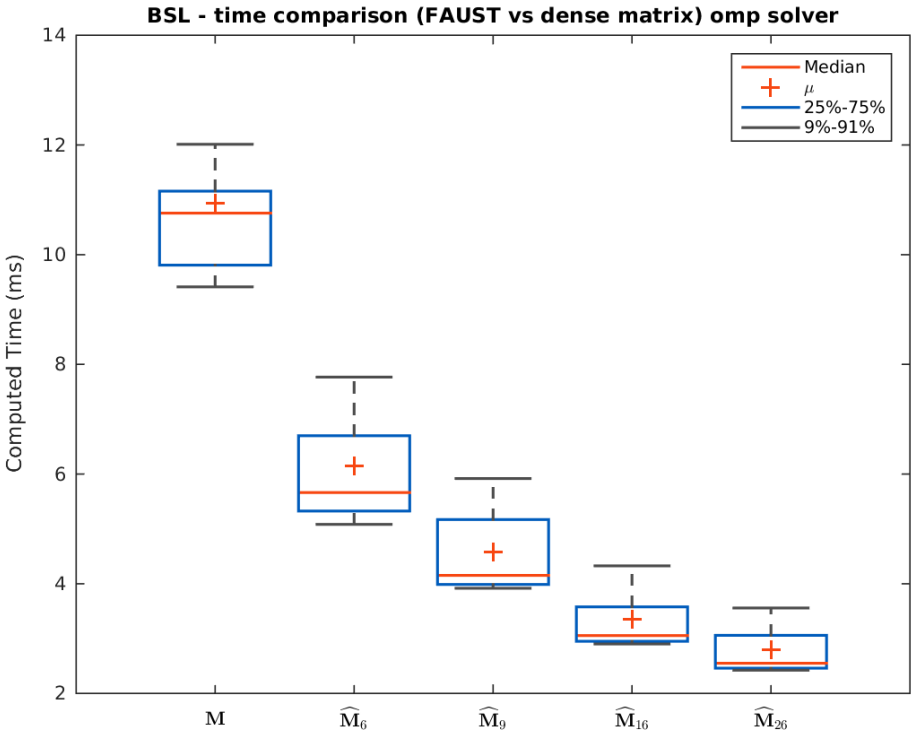


Figure 4.1: Performance of FAμST tool on Brain Source Localization experiment

Chapter 5

Work in Progress

Various features are still under development and not yet stable. But here is an overview of a roadmap for features that we plan to improve/integrate in an upcoming release :

- **Factorization algorithms :** (cf. Section 5.2)
  A beta version is available in this release, implementing in C++ what what delivered in plain Matlab in version 1.0.0. Refer to Section 5.2) for more information on how to play with these algorithms, which C++ implementation and API are not yet stabilized.
- **Graphics Processing Unit (GPU) :** (cf. Section 5.3)
  A GPU version of the code is under development, it already shows some time savings compared to the CPU code for certain computations, but is not enough user-friendly and not easy to install to be in this release. The most advanced development is currently on Linux, you can play with it following the steps described in Section 5.3. Report any problem and/or suggestion to the mailing list http://lists.gforge.inria.fr/pipermail/faust-install/.
- **Python wrapper :** A Python wrapper is also planned, it will used the Cython module.
- **Command line wrapper :** A command line wrapper is also envisioned. It will integrated the externals Libraries "XML" and "MATIO" to manipulate the data (configuration and matrix).
- **A||GO wrapper :** A wrapper for the A||GO platform of web services is also planned to propose a demonstration on the website https://allgo.inria.fr/.
- **Image denoising experiment :** (cf [**?**, chapter VI])
  This experiment was in the previous release (FAμST version 1.0, pure Matlab implementation) but is not in the current one because our C++ wrapper is not yet compatible with the sparse decomposition algorithm used in this experiment. Please use version 1.0.0 to reproduce this image denoising experiment.

5.1 Python wrapper : beta version

This section describes the different step to install the Python wrapper and to get acustomed to its behaviour. The Python wrapper has only been tested on Unix system, not Windows.

Throughout this section we provide examples as follows:

- Command lines you must enter in a terminal are displayed as:

```
1 > mkdir BUILD; ls .
```

- Messages resulting from such command lines appear without the leading '>':

```
example of return text in your current terminal.
```

- Python code is displayed like this :

```
# this is a commentary
y = A*x;
```

5.1.1 Required Components

This Section lists the required components you must install before to begin the FAμST installation.

- **Install Python**. The wrapper has only been tested with Python version 2 (2.7.x) (https://cmake.org/download/). To install Python 2, type the following command :

```
> pip install python2
```

  But Python 2 is a native package on many Unix system.

- **Install Numpy** Numpy is a Python package useful linear algebra (http://www.numpy.org/) It is often delivered with Python but in case you haven't it,you can install it by typing the following command :

```
> pip install numpy
```

  Check the version of the Numpy package, by typing the following command :

```
> pip freeze | grep 'numpy'
```

  Warning : version of Numpy older than 1.9.2 may be not compatible with the Python wrapper

- **Install Cython** Cython is a Python package, that allows to interface C/C++ with Python. (http://cython.org/)

  It is often delivered with Python but in case you haven't it,you can install it by typing the following command :

```
> pip install cython
```

5.1.2 Install

The installation is very similar to the installation of the Matlab wrapper.

- **Type the following commands :**

```
> mkdir build
> cd build
> cmake .. -DBUILD_WRAPPER_PYTHON=ON
> make
> sudo make install # run with administrator privilege
```

- By default, The Python wrapper is installed in the directory : `<HOMEDIR>\Documents\PYTHON\faust`.

  But you can choose your own install directory by setting the CMAKE variable **CMAKE_INSTALL_PYTHON_PREFIX**.

  Type the following command :

```
> mkdir build
> cd build
> cmake .. -DBUILD_WRAPPER_PYTHON=ON -
   DCMAKE_INTALL_PYTHON_PREFIX="YourPath"
> make
> sudo make install # run with administrator privilege
```

5.1.3 Quickstart

This subsection presents the quickstart demo to get acustomed to the Python wrapper. Now, you can go to the install directory of the Python wrapper which is `<HOMEDIR>\Documents\PYTHON\faust` by default.

- In a terminal, type the following command :

```
    > cd ~/Documents/PYTHON/faust/

```

- You can run the Python script `quickstart.py` by typing the following command :

```
    > python quickstart.py

```

In this script

1. We import the FaustPy package

```
#import the FaustPy package
import FaustPy;
```

2. We create a Faust from a list of factors represented as Numpy matrices

```
# create a Faust named F from its factors
A = FaustPy.Faust(list_factor)
```

3. A list of overloaded Numpy operation shows that a Faust is handled as a normal Numpy matrix.

```
# get the size of the Faust
print "dimension of the Faust : ", A.shape

# transpose a Faust
A_trans = A.transpose()

# multiplication a Numpy matrix by a Faust
x = np.random.randint(int_max, size=(dim2,1))
y = A * x

# convert a faust to numpy matrix
A_numpy = A.todense()

# slicing
coeff = A[0,0]
col_2nd = A[:,1];
submatrix_A = A[3:5,2:3]
```

4. It performs a little time comparison between multiplication by a FAμST or its Numpy equivalent matrix.

```
# get the size of the Faust
print "dimension of the Faust : ", A.shape

# transpose a Faust
A_trans = A.transpose()

# multiplication a Numpy matrix by a Faust
x = np.random.randint(int_max, size=(dim2,1))
y = A * x

# convert a faust to numpy matrix
A_numpy = A.todense()

# slicing
coeff = A[0,0]
col_2nd = A[:,1];
submatrix_A = A[3:5,2:3]
```

5.2 Construct a FAuST from a given matrix

Please ensure that you have configured your Matlab environment (cf. Section 4.1). Then, to see an example of building a FAμST from a matrix, you can run `factorize_matrix.m` in the Matlab Command Window by typing :

```
>> factorize_matrix
```

`factorize_matrix.m` script is located in the following path :
`<FAuST_INSTALL_DIR>/demo/Quick_start/factorize_matrix.m`

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

```
% number of row of the matrix
dim1 = 100;
% number of column of the matrix
dim2 = 200;
% matrix to factorise
A = rand(dim1,dim2);
```

we generate the parameters of the factorization from :

- The dimension of A (**dim1** and **dim2**),
- **nb_factor:** the desired number of factors of the FAμST,
- **rcg:** the targeted *Relative Complexity Gain*, which represents the theoretical memory gain and multiplication speed-up of the FAμST compared to the initial matrix .

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

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

Then we factorize the matrix **A** into a FAμST **Faust_A**

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

5.3 Installing FAuST to enable GPU acceleration

As a beta version, the FAμST toolbox integrates optional GPU (Graphics Processing Unit) acceleration to improve its time performance.

Warning: Currently, this optional GPU install has only be implemented and tested on a Linux machine. There is no guarantee that the installation and the use will be effective for every system.

- **Install** the CUDA Toolkit from NVIDIA website:
  https://developer.nvidia.com/cuda-downloads).

- **Install** the drivers for NVIDIA from NVIDIA website: http://www.nvidia.fr/Download/index.aspx.
- **Verify install** of GPU tools by typing in a terminal :

```
> which nvcc
```

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

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

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

- **Verify install of GPU library** by typing in a terminal:

```
> echo CUDADIR
```

You must obtain the path of your cuda directory like

```
/usr/local/cuda-7.5
```

If not, export CUDADIR (in your .bashrc file for example).

```
export CUDADIR=/usr/local/cuda-7.5
```

When prerequisities listed in Section 2.1 are checked, you can get the package FAμST.

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

```
> mkdir build
> cd build
> cmake -DBUILD_USE_GPU="ON" ..
> make
> sudo make install # run with administrator privilege
```

The FAμST Toolbox should be installed. Now, refer to Quick-Start Chapter 4 to check the install and to try FAμST toolbox **using GPU process**.

Chapter 6

Annexes

6.1 Required packages

Here is a list of packages used in the FAμST 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 GotoBLAS2 1.13 BSD version.

6.2 Compatibility between MATLAB and GCC compiler

If your gcc is too recent, you have 2 choices available, make a little modification to your installed Matlab or install a older version of gcc. 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=glnxa64 for more detail).

6.2.1 slightly modify Matlab installation

As it is explained in this MathWorks forum https://fr.mathworks.com/matlabcentral/newsreader/view_thread/255846, you can change the /usr/local/matlabRXXXXx/sys/os/glnx64/libstdc++.so.6,

- For example, under UNIX system, you can type the following command (with adapting the path to your config):

```
> cd /usr/local/matlabRXXXXx/sys/os/glnx64/
> ln -s /usr/lib/libstdc++.so.6 libstdc++.so.6
```

6.2.2 install older 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.

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

```
> which gcc
> which g++
```

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

```
# export version of gcc
export CC=/usr/lib64/ccache/gcc
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:

```
-- Configuring done
-- Generating done
-- 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 FAμST library,
`~/Documents/MATLAB/faust/` for the wrapper matlab.
You must have administrator privilege because the library file `libfaust.a` is copied in a 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 FAμST 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 a 32 bit, Matlab must be installed in 32 bit. You can check your environment variable called PROCESSOR_ARCHITECTURE by typing :

```
> set
```

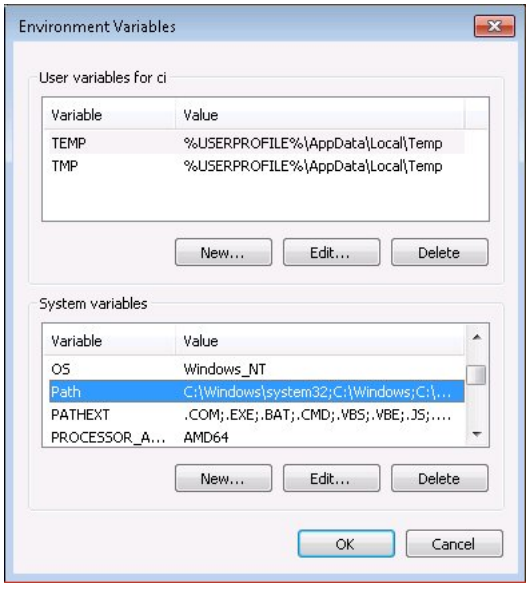
Check in the list the variable PROCESSOR_ARCHITECTURE.
PROCESSOR_ARCHITECTURE=AMD64 correspond to 64bit, PROCESSOR_ARCHITECTURE=x86 correspond to 32 bit.

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.

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



6.7 FAuST Install on MAC OS X platform, using Xcode from terminal command

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

```
> mkdir build
> cd build
> cmake .. -G "Xcode"
# list all target of the project
> xcodebuild -list -project FAUST.xcodeproj
# Build the targets
> xcodebuild -configuration "Release" -target "ALL_BUILD" build
# performs the "make install"
> xcodebuild -configuration "Release" -target "install" build
```