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

Adrien Leman; Nicolas Bellot

October 18, 2016

Contents

Chapter 1

Introduction

Presentation: FAμST 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 $\mathbf{A} = \prod_{j=1}^{J} \mathbf{S_j}$.

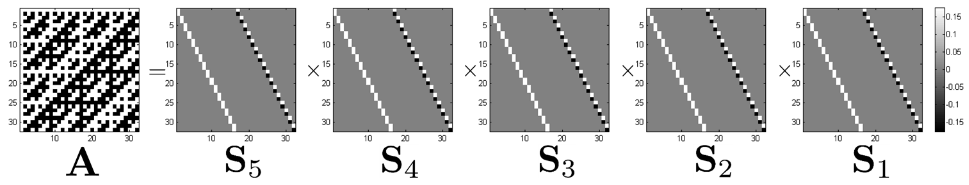


Figure 1.1: presentation

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.[1]. The FAμST 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`.

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

Chapter 2

Installation on Unix platform

The FAμST 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 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 Unix platform (both Linux and Mac OS).

Firstly, please ensure that the **prerequisites components** listed in Section 2.1 are installed. Secondly, refer to section 2.2 to download FAμST package and setting your terminal. Then refer to the appropriate section following the use (or not) of an IDE (Integrated Development Environment):

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

Finally in Section 2.6, the configure options available to build the FAμST 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 FAμST installation.

- **Install CMake**. Visit the website `https://cmake.org/` to process to the installation.
- **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** (`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).

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 Faust 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
```

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
```

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

2.4 Basic Build & Install using Code Block IDE

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

For more detail about `cmake` commands, 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.

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

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

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

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

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.
- 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.

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.

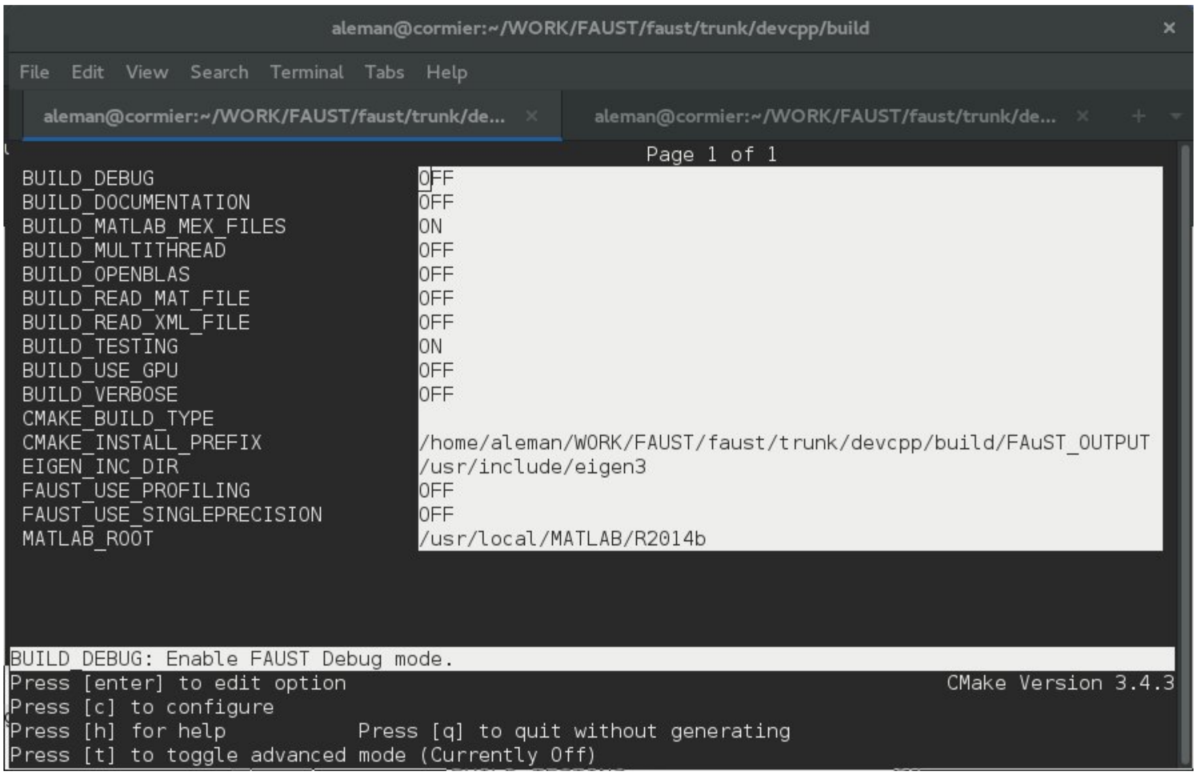
- In Xcode IDE, select **install** target and build the project.

For more detail about `cmake` commands, 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. When using the `ccmake` command in your build directory, the Cmake GUI appears in the console (see fig. 2.1).


aleman@cormier:~/WORK/FAUST/faust/trunk/devcpp/build
File Edit View Search Terminal Tabs Help
aleman@cormier:~/WORK/FAUST/faust/trunk/de...
aleman@cormier:~/WORK/FAUST/faust/trunk/de...

```
                                                  Page 1 of 1
 BUILD_DEBUG                      OFF
 BUILD_DOCUMENTATION              OFF
 BUILD_MATLAB_MEX_FILES           ON
 BUILD_MULTITHREAD                OFF
 BUILD_OPENBLAS                   OFF
 BUILD_READ_MAT_FILE              OFF
 BUILD_READ_XML_FILE              OFF
 BUILD_TESTING                    ON
 BUILD_USE_GPU                    OFF
 BUILD_VERBOSE                    OFF
 CMAKE_BUILD_TYPE
 CMAKE_INSTALL_PREFIX             /home/aleman/WORK/FAUST/faust/trunk/devcpp/build/FAuST_OUTPUT
 EIGEN_INC_DIR                    /usr/include/eigen3
 FAUST_USE_PROFILING              OFF
 FAUST_USE_SINGLEPRECISION        OFF
 MATLAB_ROOT                      /usr/local/MATLAB/R2014b



BUILD_DEBUG: Enable FAUST Debug mode.
Press [enter] to edit option                                  CMake Version 3.4.3
Press [c] to configure
Press [h] for help             Press [q] to quit without generating
Press [t] to toggle advanced mode (Currently Off)
```

Figure 2.1: ccmake GUI

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

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

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

- CMAKE_INSTALL_PREFIX : Install directory
- BUILD_TESTING : Enable the ctest option (default value is ON)
- BUILD_DOCUMENTATION : Generating the doxygen documentation (default value is OFF)
- BUILD_MULTITHREAD : Enable multithread with OpenMP Multithreading (default value is OFF)
- BUILD_VERBOSE : Enable verbose option when compile (-v) (default value is OFF)
- BUILD_DEBUG : Enable FAμST 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 FAμST 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 FAμST 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 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 3.3) or from Microsoft Visual Studio (Section 3.4.
- Then, **process to Basic installation** of FAμST tool by following instructions given in the appropriate section, depending to the kind of compiler (MinGW or Microsoft Visual) and the use (or not) of an IDE (Integrated Development Environment).

Section 3.5 describes the configure options available to build the FAμST 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 FAμST 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:

```
> 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:

```
C:\Program Files\MATLAB
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

```
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 :

```
> where matlab
```

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

```
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 :

```
> where 7z
```

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

```
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 Download Faust Package

When prerequisities listed in precedent section 3.1 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.

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

3.3 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 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 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 :

  ```
  > where gcc
  ```

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

  ```
  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

  ```
  > 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.

3.3.2 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 FAUST 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
```

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 FAUST 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
```

3.3.3 Basic build & Install using Code::Blocks IDE

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

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

Then, the 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 FAUST 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 target and build 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 and build the project. 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 FAUST 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 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.

3.4 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 3.4.1). 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 FAUST 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 target and generate the project three times.
- 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 and generate the project. 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 FAUST 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 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.

3.4.3 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

- Open a command terminal
- Set the current directory to your FAμST directory (NOTE: don't use any special character in your FAUST 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**

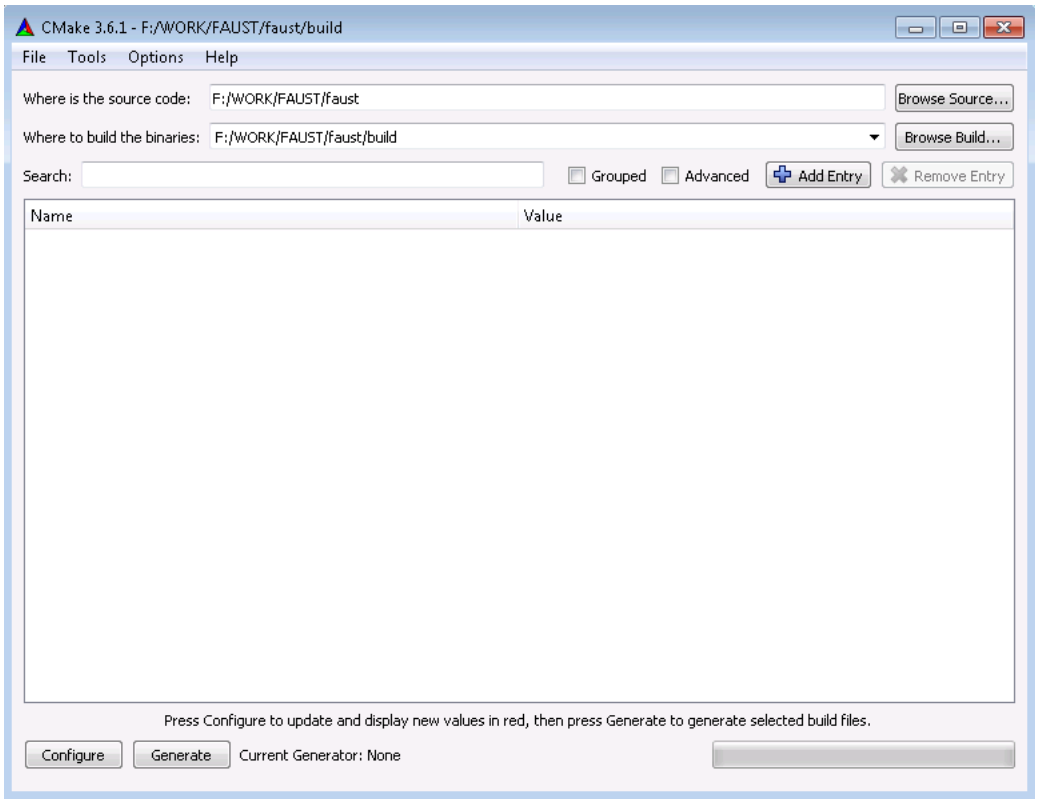
```
> cmake --build . --config "Release" --target "install"
```

3.4.3.2 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 FAμST directory (NOTE: don't use any special character in your FAUST 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"
```



CMake 3.6.1 - F:/WORK/FAUST/faust/build

Where is the source code: F:/WORK/FAUST/faust

Where to build the binaries: F:/WORK/FAUST/faust/build

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: None

Figure 3.1: cmake GUI

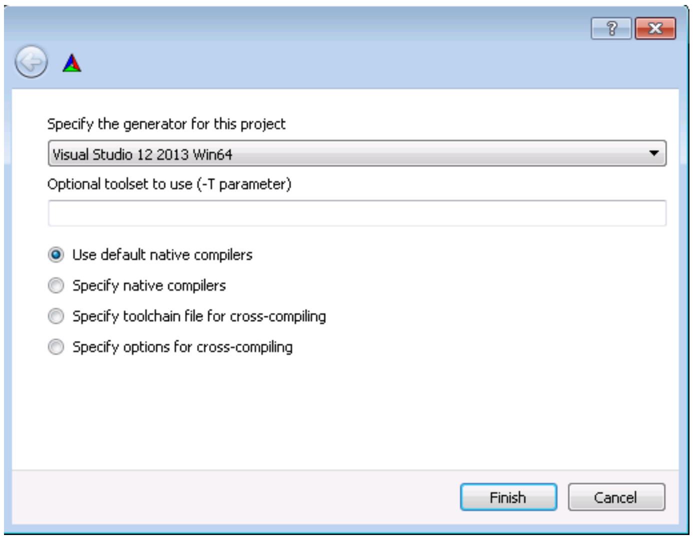
3.5 Custom - Advanced Installation

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

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).

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).



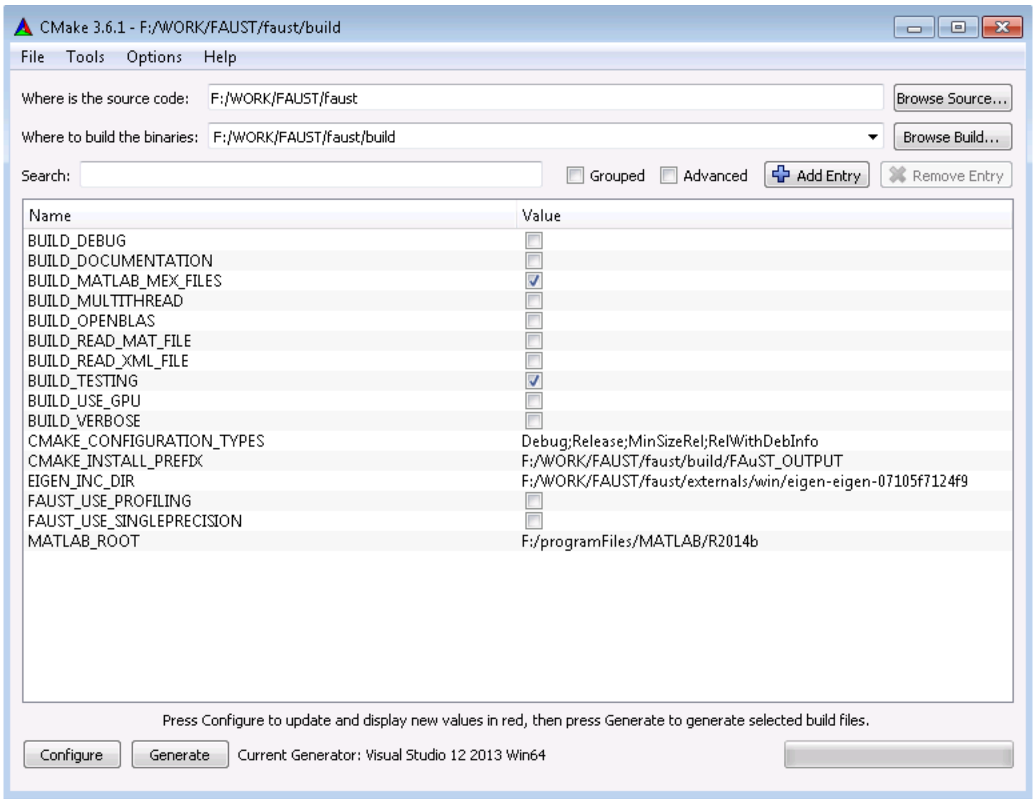
Specify the generator for this project

Visual Studio 12 2013 Win64

Optional toolset to use (-T parameter)

- [✓ Use default native compilers]
- Specify native compilers
- Specify toolchain file for cross-compiling
- Specify options for cross-compiling

Figure 3.2: cmake GUI



CMake 3.6.1 - F:/WORK/FAUST/faust/build

Where is the source code: F:/WORK/FAUST/faust

Where to build the binaries: F:/WORK/FAUST/faust/build

Name	Value
BUILD_DEBUG	☐
BUILD_DOCUMENTATION	☐
BUILD_MATLAB_MEX_FILES	✓
BUILD_MULTITHREAD	☐
BUILD_OPENBLAS	☐
BUILD_READ_MAT_FILE	☐
BUILD_READ_XML_FILE	☐
BUILD_TESTING	✓
BUILD_USE_GPU	☐
BUILD_VERBOSE	☐
CMAKE_CONFIGURATION_TYPES	Debug;Release;MinSizeRel;RelWithDebInfo
CMAKE_INSTALL_PREFIX	F:/WORK/FAUST/faust/build/FAuST_OUTPUT
EIGEN_INC_DIR	F:/WORK/FAUST/faust/externals/win/eigen-eigen-07105f7124f9
FAUST_USE_PROFILING	☐
FAUST_USE_SINGLEPRECISION	☐
MATLAB_ROOT	F:/programFiles/MATLAB/R2014b

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Visual Studio 12 2013 Win64

Figure 3.3: cmake GUI

Chapter 4

QuickStart

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

4.1 Configure Matlab path

In order to use matlab wrapper, follow the instructions :

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

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

4.2 Use a faust from a saved one

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

```
>> quick_start
```

`quick_start.m` script is located in following path :
`<HOMEDIR>/Documents/MATLAB/faust/demo/Quick_start/quick_start.m`
In this script, first of all, a Faust of size 4000x5000 is loaded from a previous one that is saved into a matfile located in :
`<HOMEDIR>/Documents/MATLAB/faust/demo/Quick_start/faust_quick_start.mat`

```
% loading a Faust A from saved-one
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.

```
 % get the size of the faust
 [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 coefficient i,j and slicing for reading purpose
 coeff=A(3,4);
 col_2=A(:,2);
 submatrix_A=A(3:5,2:3);
 submatrix_A=A(2:end,3:end-1);
 % 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).

4.3 Construct a Faust from a given matrix

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

```
>> factorise_matrix
```

`factorise_matrix.m` script is located in following path :
`<HOMEDIR>/Documents/MATLAB/faust/demo/Quick_start/factorise_matrix.m`

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

```
% 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 factorisation from :

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

```
% 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 Faust **Faust_A**

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

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

4.4 Construct a Faust from its factor

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

```
>> construct_Faust_from_factors
```

This following example shows how to build a faust 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);
```

Chapter 5

Example

5.1 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 :
`/<HOMEDIR>/Documents/MATLAB/Faust/demo/Brain_source_localization/BSL.m`)
`Fig_BSL.m` displays the Figure 5.1. illustrating the speed-up using a FAμST. (see the script file:
`/<HOMEDIR>/Documents/MATLAB/Faust/demo/Brain_source_localization/Fig_BSL.m`)

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

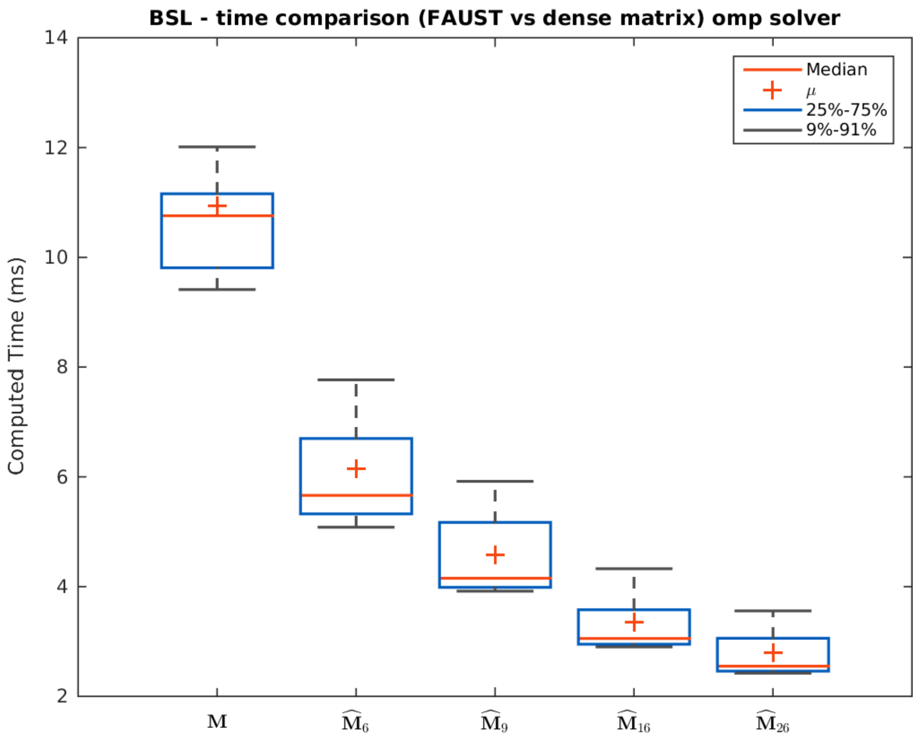


Figure 5.1: Performance of FAuST tool on Brain Source Localization experiment

Chapter 6

Annexes

6.1 Required packages

Here is a list of packages used in the 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.
- Library **xml2** `http://xmlsoft.org`
- Library **matio** `https://sourceforge.net/projects/matio`

6.2 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/compilers/R2016a/index.html?sec=glnxa64` for more detail). To temporally change your version of gcc compiler, you can modify the environment variable CC an CXX. For that, export your CC and CXX variables corresponding to gcc and g++ binaries path :

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

```
> 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++
```

An other way to change the version of compiler use by mex function from matlab command is :

```
> mex -setup
```

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 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 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 Add environment variable on Windows platform

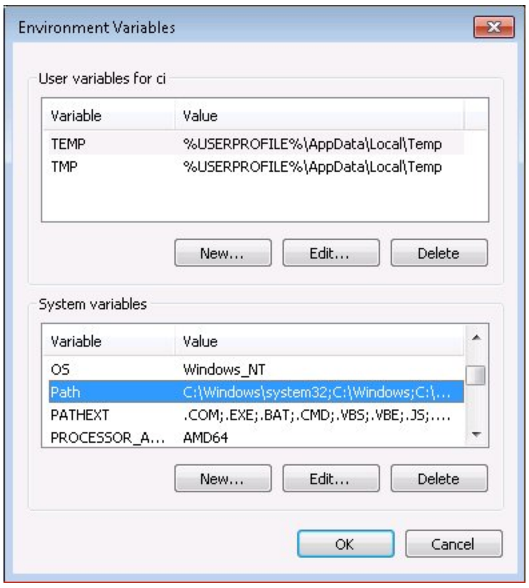
Here is the steps to add an environment variable in Windows 7.

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
```



Environment Variables

User variables for ci

Variable	Value
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	%USERPROFILE%\AppData\Local\Temp

System variables

Variable	Value
OS	Windows_NT
Path	C:\Windows\system32;C:\Windows;C:\...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;....
PROCESSOR_A...	AMD64

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.