Documentation of the project FAμST "Flexible Approximate Multi-Layer Sparse Transform"

Adrien Leman; Nicolas Bellot

September 8, 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). 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 in Le Magoarou [?]. 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: First chapter 2 explains how to install the library FAμST for UNIX platform and second chapter 3 corresponds to the Windows installation. The chapter 4 shows quickly how to used this library and finally an example is proposed in chapter 5.

Chapter 2

Installation on Unix platform

The FAμST project is based on an C++ library available for both UNIX and Windows environments. CMake has been choose 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 the Unix platform (both Linux and Mac OS). First section 2.1 presents the basic installation of FAμST and second section 2.2 corresponds to the advanced installation.

2.1 Getting Started

2.1.1 Required tools

- CMake : (tested with version 3.4.3 cf. website `https://cmake.org/`)
- Matlab : (tested with version 2014 and 2015) 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). Please adjust your version of GCC compiler in order to run the installation properly. You must too have matlab in your environment PATH. If not please add.
- *OPTIONAL* The use of GPU process in FAUST project required the drivers for NVIDIA and CUDA install. You must have nvcc in your environment PATH. If not please add.

Please export following variable:

- CC with gcc (example: `"export CC='/usr/lib64/ccache/gcc'"`)
- CXX with g++ (example: `"export CXX=/usr/lib64/ccache/g++`

2.1.2 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 directory "./externals").

- Library eigen `http://eigen.tuxfamily.org`
- Library openBlas `http://www.openblas.net`
- Library xml2 `http://xmlsoft.org`
- Library matio `https://sourceforge.net/projects/matio`

2.1.3 Basic build and installation

When prerequisities listed in precedent sections are checked, the FAμST installation can be done :

- Download the FAμST package on the website : `http://faust.gforge.inria.fr/`
- Open a command terminal
- Place you in your local 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 ..
make
sudo make install % run with administrator privilege
```

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:  <./build>
```

The command `make` will compile the build files.

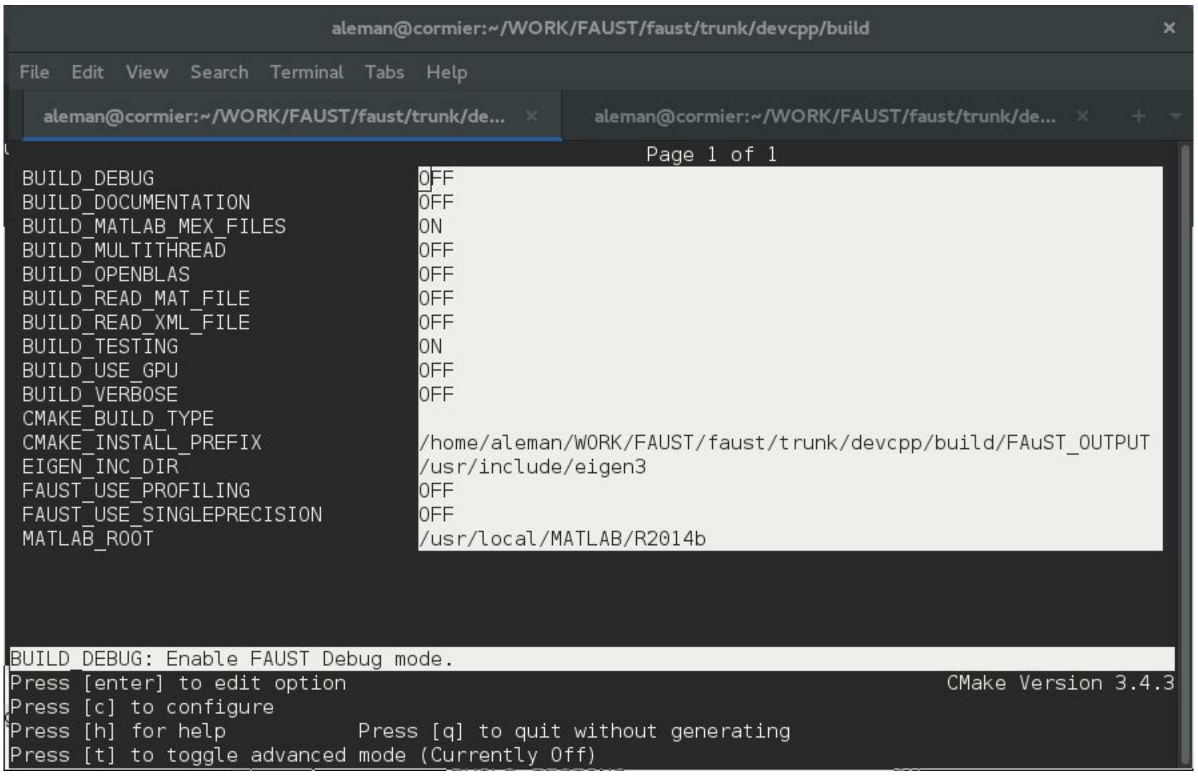
The command `sudo make install` will install the library and others components in default directory. For that, you must have administrator privilege because the library file `libfaust.a` is copied in your root path directory. If you don't have administrator privilege, you can make a local install using the following option (see 2.2):

```
cmake ..  -DCMAKE_INSTALL_PREFIX="<Home/Local/Project/Output>"
```

2.2 Custom - Advanced Installation

The project FAμST can be configured with optional parameters, for example if you want to install FAμST 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 selected 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 press [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: $cmake\ ..\ \ -D < BUILD\_NAME >=< ON/OFF >$

- 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 FAUST 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).

2.3 Build using Code Block

progress...

2.4 Build FAuST using Xcode on MAC OS

The Faust tools can be generated using XCode on MAC OS. To do this, please follow this instructions :

- Download the FAμST package on the website : `http://faust.gforge.inria.fr/`
- Open a command terminal
- Place you in your local 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 "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
```

NOTE: You can manually generated the target using visual studio from file `FAUST.sln`.

Chapter 3

Installation on Windows platform

The FAμST project is based on an C++ library available for both UNIX and Windows environments. CMake has been choose 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 the Windows platform. First section 3.1 presents the basic installation of FAμST and second section 3.2 corresponds to the advanced installation.

3.1 Getting Started

3.1.1 Required tools

The installation of the FAμST tools depends on other components to be installed in order to run properly.

1. **Install CMake** for building the FAμST tools. From `https://cmake.org/download/`, download Binary distributions correspond to your environment (in our case cmake-3.6.1-win64-x64.zip). The directory of binary must be add to the environment PATH of your system if you want to use the cmake command line tool.

2. **Install Matlab** if not already done (MATLAB R2015b in our case). The builder of the FAμST tools automatically checks your Matlab root directory if your `matlab.exe` is present in your environment Path and/or if your Matlab installation has been performed in a default directory like `"C:/Program Files/MATLAB/<R2015b>/bin/matlab.exe"` or `"C:/Program Files (x86)/MATLAB/<R2015b>/bin/matlab.exe"`. In case of several versions of Matlab installed in your system, you can force the directory of your preferred version of Matlab using the following system variable :
`MATLAB_EXE_DIR="C:/Program Files/MATLAB/<R2015b>/bin/matlab.exe"`

   Note for the case of using the compiler MinGW : In Matlab, you must install MinGW version 4.9.2 from MATLAB 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. It is easy to create. Current this latest step, an environment variable called MW_MINGW64_LOC is automatically generated.

3. **Install C++ Compiler:** Both **Microsoft visual C++** and **MinGW "Minimalist GNU for Windows"** compiler have been tested. If you are friendly with Unix tools and command line terminal, preferd **MinGW "Minimalist GNU for Windows"** C++ compiler. Otherwise, if you are more familiar with the graphical user interface, prefer the

Microsoft visual C++ compiler. The version of the C++ compiler must be coherent with the version of your Matlab version. In this documentation, the version of our C++ compiler corresponds to Matlab 2014 and 2015. If you use an other version of Matlab, please refer to the Mathworks website `http://fr.mathworks.com/support/compilers/<R20XXa>`.

For **Microsoft visual C++** installation :

- Download and install Microsoft Visual C++ 2013 professional from `https://www.microsoft.com/en-US/download/details.aspx?id=44916`

For **MinGW** installation :

- Download Mingw in `https://sourceforge.net/projects/mingw/files/latest/download?source=files`
- Launch install file and choose MINGW version 4.9.2 for mexFunction compatibility
- The directory of binary must be add to the environment PATH.
- *Note for make tool :* In a terminal command, type `make`. if it doesn't exist, 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

3.1.2 Required packages

Here is a list of packages used in the FAμST project. There are nothing to do because the installation of this packages are automatically done (see the directory "./externals").

- **Eigen** is a C++ template library for linear algebra: matrices, vectors, numerical solvers, and related algorithms. (see `http://eigen.tuxfamily.org`)

3.1.3 Basic build and Installation

When prerequisites listed in previous section 3.1.1 are checked, the FAμST installation can be done. First download the FAμST package on the website `http://faust.gforge.inria.fr/`.

Depending to your C++ compiler (MinGW or Visual studio), refer to the right part and follow the given instructions.

Using MinGW compiler:

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

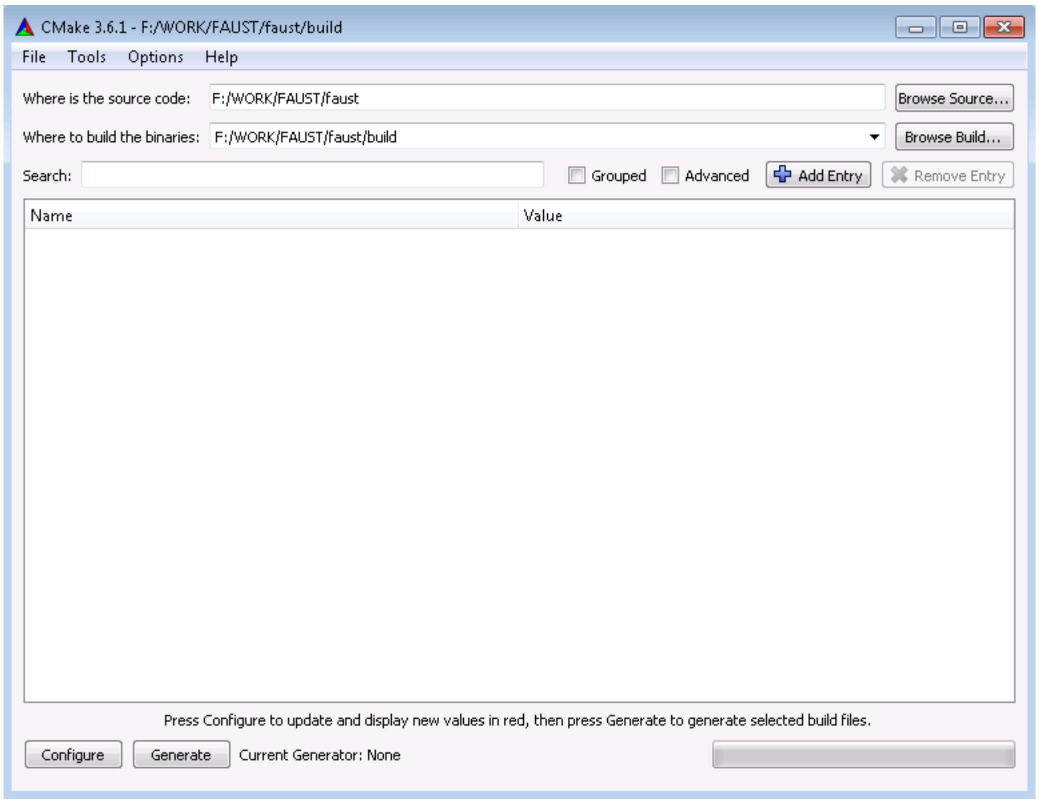
```
mkdir build
cd build
cmake -G "MinGW Makefiles" ..
make
make install % run with administrator privilege
```

NOTE: You can generated the CodeBlocks project with the following command : cmake -G "CodeBlocks - MinGW Makefiles" ..

Using Visual Studio compiler:

In the case of **Microsoft Visual Studio 2013 compiler using the Graphical Users Interfaces**:

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


CMake 3.6.1 - F:/WORK/FAUST/faust/build
File Tools Options Help
Where is the source code: F:/WORK/FAUST/faust
Browse Source...
Where to build the binaries: F:/WORK/FAUST/faust/build
Browse Build...
Search:
Grouped
Advanced
Add Entry
Remove Entry
Name
Value
Configure
Generate

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

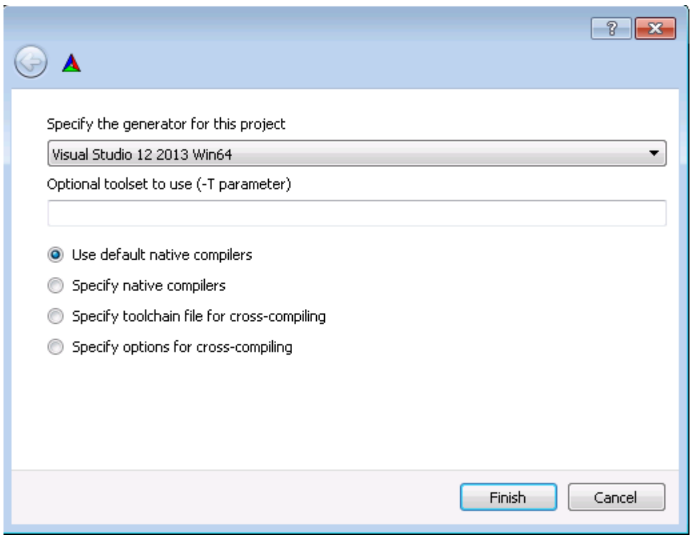
Current Generator: None

Figure 3.1: cmake GUI

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



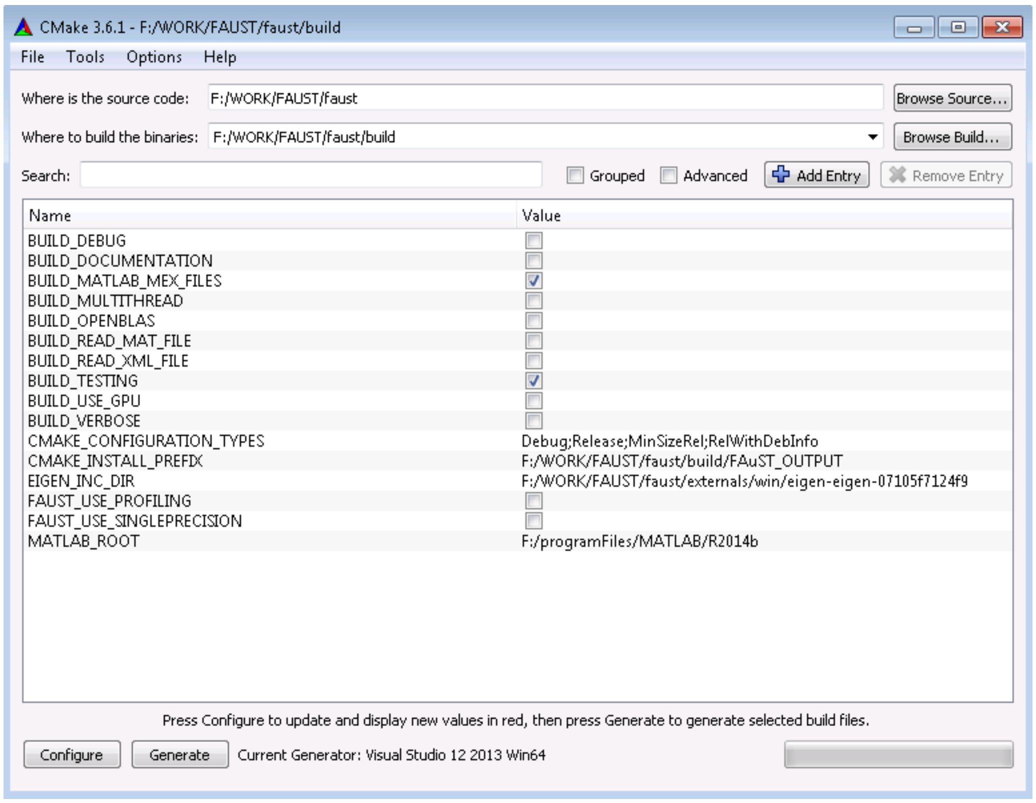
Specify the generator for this project

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

4. Once the build system configured then generated, you have to actually build FAUST, using Visual Studio.
5. Open file "faust.sln" with visual studio
6. Click right on Target ALL_BUILD and select generated
7. Click right on Target INSTALL and select generated
8. Click right on Target CTEST and select generated

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

- Open a command terminal
- Place you in your local 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" --target "install"
```

3.2 Custom - Advanced Installation

progress...

Chapter 4

QuickStart

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

As much as possible, a **Faust** object is handled as a normal matlab matrix, here is listing of matlab builtin function that can be applied to a faust A :

```
>> % considering A is a Faust of size 10x3
>>
>> % get the size of the faust
>> [dim1,dim2] = size(A);
>>
>> %  transpose a faust
>> A_trans = A';
>>
>> % multiplication by A
>> x1 = rand(3,3);
>> y1 = A*x1;
>>
>> % multiplication by A'
>> x2 = rand(10,5);
>> y2 = A'*x2;
>>
>>
>> % get the 2-norm (spectral norm) of the faust A
>> norm_A = norm(A); % equivalent to norm(A,2);
>>
>> % get the coefficient i,j and slicing for reading purpose
>> coeff=A(i,j);
>> 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
>>
>> % get the number of non-zeros coefficient
>> nz = nnz(A);
```

4.1 construct a faust

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

4.1.1 construct a faust from a cell-array

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

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

```
>> nb_factor = 3;
>> dim1 = 5;
>> dim2 = 3;
>>
>> % list of factors
>> factors = cell(nb_factor);
>>
>> factors{1}=rand(dim1,dim2); % first factor is rectangular
>> for i=2:nb_factor
>>      factors{i}=rand(dim2,dim2);
>> end
>>
>> % build the faust
>> A=Faust(factors);
```

An optional multiplying scalar argument can be taken into account :

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

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

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

4.1.2 construct a faust from a saved one

You can also build a faust from a previously one which is saved into a mat file :

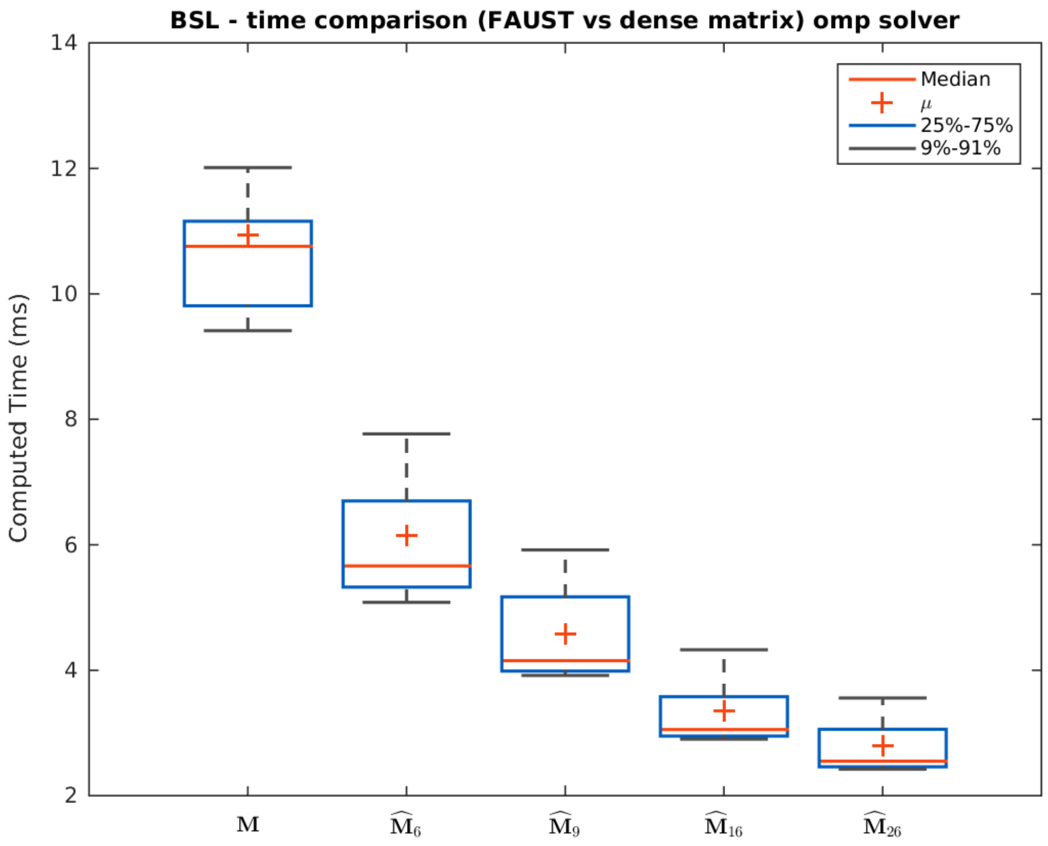
```
>> %  save the faust A into the file faust.mat
>> [lambda,factors]=save(A,'faust.mat');
>> % create a new faust from the file faust.mat
>> A_loaded=Faust('faust.mat');
```

Chapter 5

Example

5.1 Brain Sources Localization

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


BSL - time comparison (FAUST vs dense matrix) omp solver
Computed Time (ms)
Median
μ
M
$\widehat{M}_6$
$\widehat{M}_9$
$\widehat{M}_{16}$
$\widehat{M}_{26}$