# Alternatives for the High-Level Interface to the Data Assimilation Library

Claire Mouton

# Alternatives for the High-Level Interface to the Data Assimilation Library

Claire Mouton - claire.mouton@inria.fr

November 2008

## Part I
# Requirements

The various solutions to interface the library should be evaluated bearing in mind the following constraints:

- The maintenance cost should be minimum. This implies an almost automatic building of the interface to include any change in the C++ code.
- The interface should give access to all functions, objects and methods of the C++ code. Note that a library defines a large number of functions and objects.

The high-level language can be either Python or Matlab or both. Elements to compare Python and Matlab can be found here: http://mail.python.org/pipermail/python-list/2003-January/827579.html and from page 54 to page 80 there: http://web.bryant.edu/~bblais/bryant/numerical_computing/python_matlab.pdf.

## Part II
# C++ to Python Interface

## 1 SWIG

SWIG (Simplified Wrapper and Interface Generator) is an open source software tool used to connect programs or libraries written in C/C++ with eighteen different languages such as Python and Octave (http://www.swig.org/). The current version is SWIG-1.3.35 released in April 2008. The aim is to achieve the connection between the C++ code

and Python with minimal effort: a small number of directives are added to the program header files and within an interface file, and the SWIG tool then creates the source code which provides the "glue" between C/C++ and Python.

SWIG provides wrapping support for almost all of ANSI C++ such as:

- Overloaded functions and methods,
- Overloaded operators,
- Templates,
- C++ library support for strings and the STL.

## 2 Other Existing Tools

**SIP** (based on SWIG) is a tool that makes it easy to create Python bindings for C and C++ libraries (http://www.riverbankcomputing.com/software/sip/intro). It was used to build PyQt (the Python bindings for the Qt toolkit) and PyKDE. C++ templates are supported. It does not appear to be specifically tied to Qt, but does not seem to have gained general acceptance with the rest of the Python Community.

**Boost.Python** is a C++ library which enables seamless inter-operability between C++ and the Python programming language (http://www.boost.org/doc/libs/1_37_0/libs/python/doc/index.html). It is designed to wrap C++ interfaces as unobtrusively as possible. It is a very rich wrapper interface including support for:

- Function overloading,
- C++ to Python exception translation,
- Default arguments,
- Exporting C++ iterators as Python iterators.

It seems that Boost.Python wrappers can be generated automatically with Py++ (http://www.language-binding.net/pyplusplus/pyplusplus.html). It prevents from updating the code generator script every time the source code is changed.

**Weave** allows the inclusion of C/C++ within Python code and is useful in accelerating Python code (http://www.scipy.org/Weave). Weave is a subpackage of SciPy. It is a way to optimize time critical code.

**Instant** is a Python module that allows for instant inlining of C and C++ code in Python (http://www.fenics.org/wiki/Instant). It is a small Python module built on top of SWIG.

**Examples using these tools** can be found there: http://www.suttoncourtenay.org.uk/duncan/accu/integratingpython.html.

# Part III
# C++ to Matlab Interface

## 3 Direct Interface

### 3.1 Swig for Matlab

SWIG-1.1, the second SWIG version released in 1997, only covers Tcl, Perl and Python. It also presents a Matlab module (http://garr.dl.sourceforge.net/sourceforge/swig/swig1.1p5.tar.gz). This is a highly experimental language module that allows SWIG to create native Matlab cMEX wrappers. That is, one can take a C library and turn it into a working Matlab module.
*Limitations* — This module has only been tested with Matlab 5.0 under Windows NT-4.0. No support for C arrays or mapping between Matlab matrices and arrays has been implemented. It is not developed any more. It remains undocumented and essentially untested.

SwigMatlabPlus is a simple extension of this SWIG module (http://alumni.media.mit.edu/~sbasu/code/swigmatlabplus/), under MIT license. This allows to take C++ code and export it to Matlab. This gives access to C++ objects, member variables, and methods from Matlab. A support for vectors/matrice has been added.
*Limitations* — This is only available for Visual C++.

In May 2008, a Matlab branch was created in the SWIG public repository (http://swig.svn.sourceforge.net/viewvc/swig/branches/matlab-branch/). This branch aims at experimenting with making Octave module generate Matlab MEX-files.
*Limitations* — Development on this branch did not get far at all. However, the work to port the Octave backend to Matlab MEX-files does not require an outrageous amount of work. If one is interested in moving it forward, the author of the Octave backend (Xavier Delacour) would be willing to provide guidance and explanation.

### 3.2 MEX-files

MEX is a built-in utility that enables to call C, C++ or Fortran code in Matlab by compiling the code into a Matlab Executable called a MEX-file (http://www.mathworks.com/support/compilers/interface.html). MEX-files are dynamically linked subroutines that are called as regular Matlab functions.
*Limitations* — MEX-files have a specific syntax. The application `main()` has to be replaced with a special gateway function (called "MEXFunction") to pass inputs and outputs to and from Matlab. In C++, function argument checking is done at compile time. In Matlab and with MEX-files, any number or type of arguments can be passed to a MEX-function, which is responsible for argument checking to be written manually. It does not match with the low maintenance cost requirement.

### 3.3 Matwrap

Matwrap is a wrapper generator for matrix languages such as Matlab, Octave or Tela (http://lnc.usc.edu/~holt/matwrap/), available free of charge under the terms of the artistic license (http://www.perl.com/language/misc/Artistic.html). It is a Perl script that generates wrapper functions automatically from .h files.
*Limitations* — It was developed for Matlab version 5 and it has not been adapted to the latest version. Furthermore, it was compiled with gcc 2.7.2, and has not been tested on other compilers. It does not support template programming, function overloading, operator definition, function and member function pointers.

## 4 Interface via Octave

An interface to Matlab can be built through Octave.

The first step is to build an interface to Octave via SWIG (see section 1).

The second step is to convert the code from Octave to Matlab. To do so, Oct2Mat can be applied to the files (http://users.powernet.co.uk/kienzle/octave/oct2mat.tar.gz).
This is an awk script supporting the main differences between Octave and Matlab.
*Limitations* — It was created for and tested with Matlab 4. Oct2Mat needs the implementation of some features to be complete, such as:

- Converting `function [x y]=...` to `function [x, y]=...`,
- Converting `_y` to `y_`,
- Converting `x=y=Z;` to `y=Z; x=y;`,

- Variable argument/return lists,
- Solving confusion with transpose operator when appearing on a line containing a string,
- Handling Octave builtin variables.

## 5 Matlab through Python

**PyMat** exposes the Matlab engine interface allowing Python programs to start, close, and communicate with a Matlab engine session. In addition, the package allows transferring matrices to and from a Matlab workspace. These matrices can be specified as NumPy arrays, allowing a blend between the mathematical capabilities of NumPy and those of Matlab. For instance, this allows to plot NumPy arrays in a Matlab plot window. For further information: http://pymat.sourceforge.net/.

**Mlabwrap** (based on PyMat) is a high-level Python-to-Matlab bridge: it makes Matlab look like a normal Python library. Beta version under BSD License or MIT License available from http://mlabwrap.sourceforge.net/.