OpenTURNS Modules

Régis LEBRUN

EADS Innovation Works

June 12th 2012

Outline

1. openturns-fftw
2. openturns-mixmod

Fast Fourier Transform

Description

In several places in OpenTURNS algorithms, one has to evaluate the sequence of complex numbers $(z_k)_{k=0,\dots,N-1}$ from a given sequence of complex numbers $(x_j)_{j=0,\dots,N-1}$ such that :

$$z_k = \sum_{j=0}^{N-1} x_j \exp\left(-2i\pi\frac{jk}{N}\right) \qquad (1)$$

The sequence $(z_k)_{k=0,\dots,N-1}$ is known as the Discrete Fourier Transform of the sequence $(x_j)_{j=0,\dots,N-1}$, and can be computed efficiently using the Fast Fourier Transform algorithm.
Conversely, one can have to compute the sequence $(x_j)_{j=0,\dots,N-1}$ given the sequence $(z_k)_{k=0,\dots,N-1}$ such that :

$$x_j = \frac{1}{N}\sum_{k=0}^{N-1} z_k \exp\left(2i\pi\frac{jk\pi}{N}\right) \qquad (2)$$

The sequence $(x_j)_{j=0,\dots,N-1}$ is known as the Inverse Discrete Fourier Transform of the sequence $(z_k)_{k=0,\dots,N-1}$, and can be computed efficiently using the Fast Fourier Transform algorithm too.

Fast Fourier Transform in OpenTURNS

Classes and algorithms

The main class to perform FFT in OpenTURNS is the `FFT` class, with a default implementation using the `KissFFT` class. This class is based on the kiss FFT library (http ://kissfft.sourceforge.net/) which is a reasonably fast free implementation of the Fast Fourier Transform compatible with OpenTURNS license.

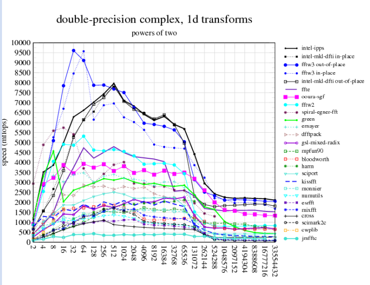
Where is it used in OpenTURNS ?

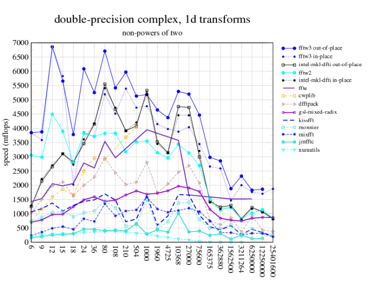
Two main classes benefit from an efficient implementation of discrete Fourier transforms : the `SpectralNormalProcess` and the `WelchFactory` classes, dedicated to the simulation and the estimation of vector-values normal processes described in the frequency space.
Two other classes benefit from the FFT algorithm : the `WhittleFactory` class dedicated to the estimation of 1D ARMA processes, and the multiplication operator of the `UniVariatePolynomial` class which may be used to perform convolution of discrete 1D distributions efficiently.

Efficient FFT computation through the use of FFTW

What is FFTW ?

From the website www.fftw.org :
"FFTW is a C subroutine library for computing the discrete Fourier transform (DFT) in one or more dimensions, of arbitrary input size, and of both real and complex data (as well as of even/odd data, i.e. the discrete cosine/sine transforms or DCT/DST). We believe that FFTW, which is free software, should become the FFT library of choice for most applications."





The openturns-fftw module in practice

Installation (linux)

```
> svn export https ://svn.openturns.org/openturns-modules/openturns-fftw/trunk
> cd trunk && ./bootstrap
> ./configure -with-openturns=YOUR_OT_INSTALL_PATH && make && make dist
> YOUR_OT_INSTALL_PATH/bin/openturns-module -install otfftw-0.0.0.tar.gz
```

Basic usage

```python
from openturns import *
from otfftw import *

myFFTW = FFTW()
size = 8
x = NumericalComplexCollection(size)
for i in range(size) :
    x[i] = (i + 1.0) * (1.0 - 0.2j)
z = myFFTW.transform(x)
y = myFFTW.inverseTransform(z)
```

The openturns-fftw module in practice

Usage with SpectralNormalProcess

```
from openturns import *
from otfftw import *

timeGrid = RegularGrid(0.0, 0.01, 100)
process = SpectralNormalProcess(CauchyModel(), timeGrid)
myFFTW = FFTW()
process.setFFTAlgorithm(myFFTW)
```

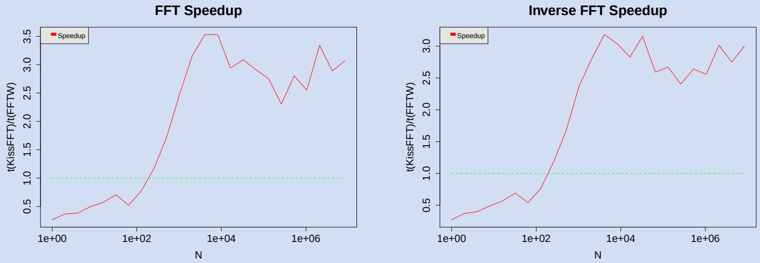
Usage with WelchFactory

```
from openturns import *
from otfftw import *

factory = WelchFactory()
myFFTW = FFTW()
factory.setFFTAlgorithm(myFFTW)
```

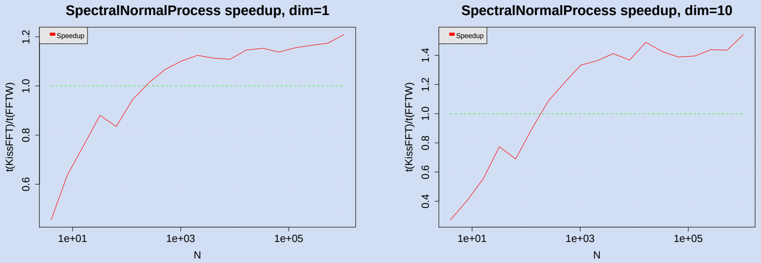
Performance

Basic usage : FFT and inverse FFT on a power of two data segment



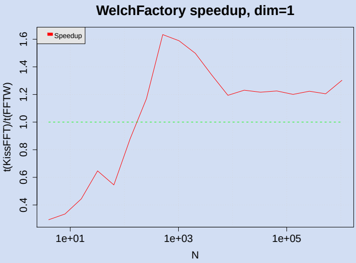
Performance

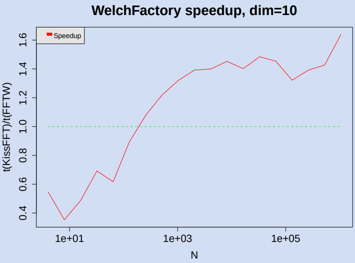
WelchFactory estimation, in dimension 1 and 10



Performance

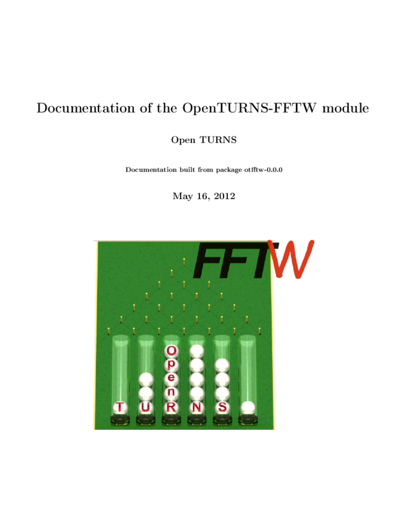
SpectralNormalProcess simulation, in dimension 1 and 10





Going further

Read the documentation !



Documentation of the OpenTURNS-FFTW module

Open TURNS

Documentation built from package otfftw-0.0.0

May 16, 2012

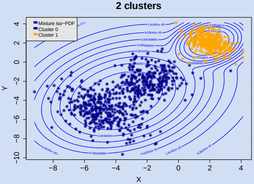
Mixture of normal distributions

Description

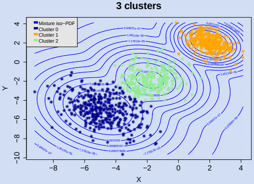
The first aim of the MixMod library (www.mixmod.org) is to estimate the parameters of a mixture of N multidimentional normal distributions from a multidimensional sample using likelihood maximization, the parameter N being provided by the user. The first version of the openturns-mixmod module was focus on this capability. Given this mixture, the Mixmod library is also able to classify the data into N clusters : each point gets a label that indicates its cluster. As a byproduct of the estimation, the Mixmod library also provides several information criteria that allow to select the best number of atoms to model te data. The new version of the openturns-module provides an interface to these functionalities.

Classification and model selection

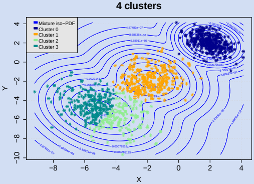
2D classification for probabilistic model selection



Comp. log-like=-4166.24



Comp. log-like=-3977.05



Comp. log-like=-4109.73

Mixture of experts

Application to meta-modeling

We want to approximate a function $f : \mathbb{R} \to \mathbb{R}$ which is known to be piecewise-smooth but with discontinuities. We adopt the following strategy :

1. We form a bidimensional database $(X_i, f(X_i))_{i=1,\dots,N_\ell}$
2. We iterate on the number of clusters k to build a bidimensional mixture :

$$p_k(x, y) = \sum_{j=0}^{k-1} w_j \phi_{\mu_j,\Sigma_j}(x, y)$$

   where ϕ_{μ_j,Σ_j} is the density of a bidimensional Gaussian vector with mean μ_j and covariance matrix Σ_j.
3. For each value of k, we partition the database into k clusters $\left((X_i^j, f(X_i^j))_{i=1,\dots,N_j}\right)_{j=0,\dots,k-1}$ such that $N_\ell = \sum_{j=0}^{k-1} N_j$

Mixture of experts

Application to meta-modeling, cont.

We want to approximate a function $f : \mathbb{R} \to \mathbb{R}$ which is known to be piecewise-smooth but with discontinuities. We adopt the following strategy :

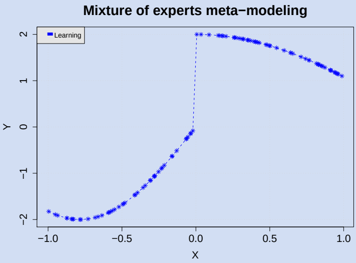
1. For each value of k, we partition the database into k clusters $\left((X_i^j, f(X_i^j))_{i=1,\dots,N_j}\right)_{j=0,\dots,k-1}$ such that $N_\ell = \sum_{j=0}^{k-1} N_j$
2. For each cluster $(X_i^j, f(X_i^j))_{i=1,\dots,N_j}$ we build a low dimensional polynomial chaos expansion meta-model $\hat{f}^j$ using the first marginal distribution of ϕ_{μ_j,Σ_j} as the input distribution
3. The meta-model $\hat{f}$ is computed using :

$$\forall x \in \mathbb{R},\ \hat{f}(x) = \hat{f}^{j^*} \text{ where } j^* = \mathrm{argmax}_j w_j \phi_{\mu_j,\Sigma_j}\left(x, \hat{f}^j(x)\right)$$

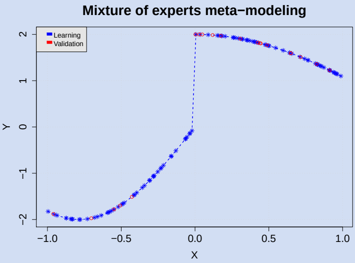
4. The optimal number of clusters k^* is selected using the L^2 error on a validation database $(\tilde{X}_i, f(\tilde{X}_i))_{i=1,\dots,N_V}$

Mixture of experts

2D classification for 1D discontinuous meta modeling



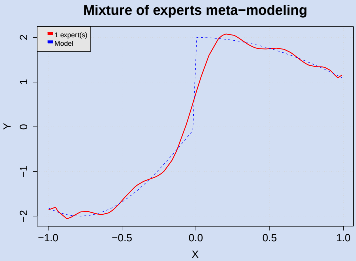
Learning database
100 points



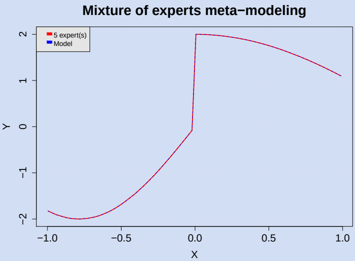
Validation database
20 additional points

Mixture of experts

2D classification for 1D discontinuous meta modeling, cont.



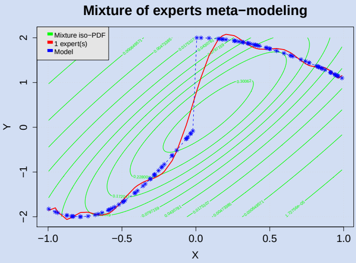
1 expert : global PCE of degree 24
L^2 validation error=1.4172



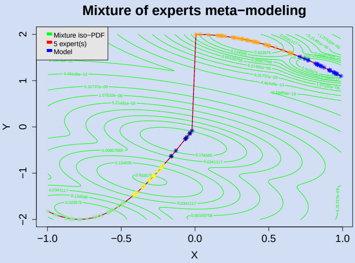
12 experts : local PCE of degree 2
L^2 validation error=$1.7635\,10^{-7}$

Mixture of experts

2D classification for 1D discontinuous meta modeling, cont.



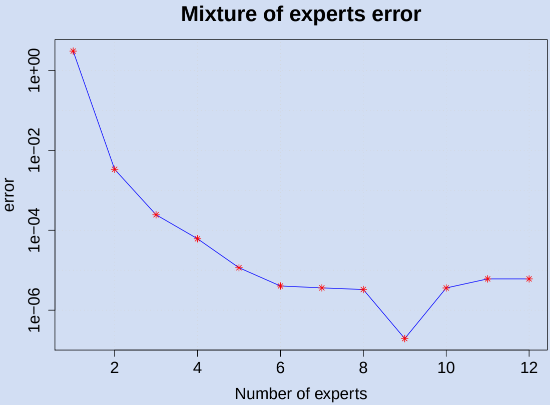
1 expert : global PCE of degree 16
L^2 validation error=3.0400



5 experts : local PCE of degree 2
L^2 validation error=$1.1555\,10^{-5}$

Mixture of experts

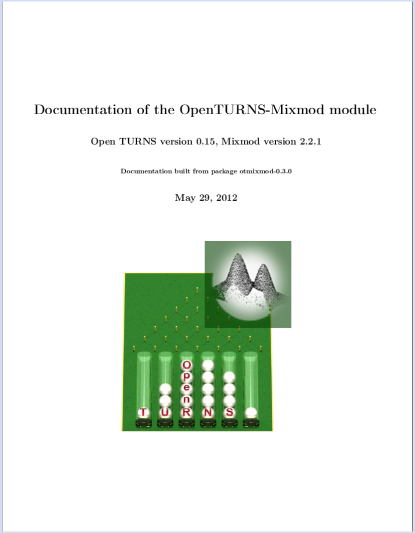
2D classification for 1D discontinuous meta modeling, cont.



Evolution of the validation error

Going further

Read the documentation !



Documentation of the OpenTURNS-Mixmod module

Open TURNS version 0.15, Mixmod version 2.2.1

Documentation built from package otmixmod-0.3.0

May 29, 2012

Conclusion

Any question ?