Subject: Re: Added Eigen
Posted by forlano on Fri, 10 Aug 2012 06:42:56 GMT
View Forum Message <> Reply to Message

koldo wrote on Tue, 07 June 2011 07:37Hello all

Eigen library has been included to Bazaar. See here for details.

It includes matrix algebra and math algorithms.

In addition to the Eigen package, there is an Eigen_demo package with many demos from very simple ones to non-linear equations systems solving and optimization.

U++ Bazaar Eigen packages have been cooked by Honza (dolik.rce) and koldo (me).

Hello,

in the last week I needed a robust non linear fitting algorithm. I looked for it on the net and downloaded several. Unfortunately they do not compile on my windows machine or needed other libraries.
I was giving up when I realized it was already in my computer since one year... and the best one!
Thanks for providing this excellent package.

I have modified it for my need and of course it work as expected. Here my demo for a logistic fit:

```
#include <Core/Core.h>

using namespace Upp;

#include <plugin/Eigen/Eigen.h>
#include <plugin/Eigen/unsupported/Eigen/NonLinearOptimization>

using namespace Eigen;

// Generic functor
template<typename _Scalar, int nx = Dynamic, int ny = Dynamic>
struct Functor {
 typedef _Scalar Scalar;
 enum {
  InputsAtCompileTime = nx,
  ValuesAtCompileTime = ny
 };
 typedef Matrix<Scalar,InputsAtCompileTime,1> InputType;
 typedef Matrix<Scalar,ValuesAtCompileTime,1> ValueType;
 typedef Matrix<Scalar,ValuesAtCompileTime,InputsAtCompileTime> JacobianType;

 const int m_inputs, m_values;
```

```
 Functor() : m_inputs(InputsAtCompileTime), m_values(ValuesAtCompileTime) {}
 Functor(int inputs, int values) : m_inputs(inputs), m_values(values) {}

 int inputs() const {return m_inputs;}
 int values() const {return m_values;}

 // you should define that in the subclass :
 virtual  void operator() (const InputType& x, ValueType* v, JacobianType* _j=0) const {};
};

struct LogisticA_functor : Functor<double> {
 LogisticA_functor() : Functor<double>(3,15) {}
 static const double x[15];
 static const double y[15];
 int operator()(const VectorXd &b, VectorXd &fvec) const {
  ASSERT(b.size()==3);
  ASSERT(fvec.size()==15);
  for(int i=0; i<15; i++)
   fvec[i] = b[0] / (1.0 + b[1]*exp(-1.0 * b[2] * x[i])) - y[i];
  return 0;
 }
};
const double LogisticA_functor::x[15] = {-280.0, -240.0, -200.0, -160.0, -120.0, -80.0, -40.0, 0.0,
40.0, 80.0, 120.0, 160.0, 200.0, 240.0, 280.0};
const double LogisticA_functor::y[15] = {0.061276, 0.071429, 0.091574, 0.112821, 0.132959,
0.131597, 0.167887, 0.198380, 0.221380, 0.292292, 0.351831, 0.445803, 0.497754 ,
0.609337,0.632353};


void NonLinearOptimization() {

  VectorXd x(3);
  x << 5., 5., 0.01;  // Initial values

//first run
  LogisticA_functor functor;
  NumericalDiff<LogisticA_functor> numDiff(functor);
  LevenbergMarquardt<NumericalDiff<LogisticA_functor> > lm(numDiff);
  int ret = lm.minimize(x);
  if (ret == LevenbergMarquardtSpace::ImproperInputParameters ||
   ret == LevenbergMarquardtSpace::TooManyFunctionEvaluation)
   Cout() << "\nNo convergence!: " << ret;
  else {
     for (int i=0; i<3; i++) Cout() << "Parameter: "<< i << " = " << x[i] << "\n";
  }

//second run with new data of different length and same curve to fit
```

//
// x[13] = {-280.0, -240.0, -200.0, -160.0, -120.0, -80.0, -40.0, 0.0, 40.0, 80.0, 120.0, 160.0, 200.0};
// y[13] = {0.061276, 0.071429, 0.091574, 0.112821, 0.132959, 0.131597, 0.167887, 0.198380, 0.221380, 0.292292, 0.351831, 0.445803, 0.497754};
// ..... ??? .....

}

CONSOLE_APP_MAIN
{ NonLinearOptimization();
}

Now, and here comes the problems, I would like to use the same curve, with the same number of parameters, but with a NEW dataset [X,Y] of different size.
I can duplicate the code (new functor and new drive) and it should work. But I need to do it up to 16 different dataset and my way is very, very silly.

It should be a way to modify the template of the functor to permit to feed at request a data set [X,Y] of N values.
Unfortunately the template structure and the operator() scary me and I do not know where to put my hands.

Can I ask a more easy way to drive the same functor with different dataset leaving unchanged the fitting curve?

Thanks a lot for your patience.
Luigi