Minimizer Interface (MI)

```
int main(int argc, char **argv){
  std::string whichMinimizer("all");
  double p0=-10., p1=10., p2=1., p3=-0.01, sigma_smear=3;

  // Generate data distribution
  boost::shared_ptr<MIData> myFit(new PolyFit(p0, p1, p2, p3, sigma_smear));

  //-------------------------Minimizer IF -----------------------------------------
  std::vector<boost::shared_ptr<MIBase>> myMinimizerList;

  // Add minimizers
  if (whichMinimizer=="Geneva")
    myMinimizerList.push_back(boost::shared_ptr<MIBase> (new MIGeneva(myFit)));
  else if (whichMinimizer=="Minuit")
    myMinimizerList.push_back(boost::shared_ptr<MIBase> (new MIMinuit(myFit)));
  else if (whichMinimizer=="all") {
    myMinimizerList.push_back(boost::shared_ptr<MIBase> (new MIGeneva(myFit)));
    myMinimizerList.push_back(boost::shared_ptr<MIBase> (new MIMinuit(myFit)));
  }else{
   std::cout << "Minimizer/t" << whichMinimizer << "\tdoesn't exist" << std::endl;
   return 0;
  }

  // Initiate parameters
  double val[4], min[4], max[4], err[4];
  val[0] = -11; max[0] = 0; min[0] = -20; err[0] = 3;
  val[1] = 9.8; max[1] = 15; min[1] = 5; err[1] = 2;
  val[2] = 1.1; max[2] = 1.5; min[2] = 0.5; err[2] = 0.3;
  val[3] = -0.008; max[3] = 0.; min[3] = -0.02; err[3] = 0.005;

  // Loop over minimizers (at the moment this means: Geneva, Minuit or Geneva then Minuit)
  for(unsigned int Nmin=0; Nmin<myMinimizerList.size(); Nmin++){
    // Pointer to one of the used minimizers
    boost::shared_ptr<MIBase> minimizer = myMinimizerList[Nmin];
    // Do the actual minimization
    double genResult = minimizer->exec(4, val,  min, max, err);

    std::cout << "Minimizer " << Nmin << "\t final par :\t" << genResult << std::endl;
    std::cout << "final a:\t" << val[0] << " +- " << err[0] << std::endl;
    std::cout << "final b:\t" << val[1] << " +- " << err[1] << std::endl;
    std::cout << "final c:\t" << val[2] << " +- " << err[2] << std::endl;
    std::cout << "final d:\t" << val[3] << " +- " << err[3] << std::endl;
    std::cout << "Done ..." << std::endl << std::endl;
  }

  // Plot results
  myFit->drawGraph(val[0],val[1],val[2],val[3]);
  return 0;
}
```