Help

```
#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2007+2) //The "#els
#else

//reduction de variance!!!

#include <
href../../../common/math/highdim_solver/highdim_vector_h_src.pdfvector>
#include "
href../../../mod/hes1d/hes1d_pad/generator_h_src.pdfgenerator.h"
#include "
href../../../mod/hes1d/hes1d_pad/model_heston_h_src.pdfmodel_heston.h"
#include "pnl/pnl_cdf.h"

#ifndef function_heston_var_control_h_
#define function_heston_var_control_h_

//using namespace std;

//here we describe the functions of heston model
//for more information see the report

std::vector<double> model_heston::exp_V0(double s, std::vector<double> _x)
{

  double epsilon = DBL_EPSILON;

  std::vector<double> x(_x.size());
  double J = theta - beta * beta * 0.25 / alpha;
  double A = nu - 0.5 * _x[1];
  double mult = (std::abs(A) <= epsilon) ? s : (exp(A * s) - 1.) / A;

  x[0] = _x[0] * exp((nu - 0.5 * J) * s + ((_x[1] - J) * 0.5 / alpha) * (exp(-al
  x[1] = J + (_x[1] - J) * exp(-alpha * s);
  x[2] = _x[2] + _x[0] * mult;

  return x;
};
```

```
std::vector<double> model_heston_var_control::exp_V0(double s, std::vector<doubl
{
  double epsilon = DBL_EPSILON;

  std::vector<double> x = model_heston::exp_V0(s, _x);
  double y0 = x0[1];

  x[3] = _x[3] * exp(s * ((nu - 0.5 * theta) + 0.5 * (y0 - theta) * (exp(-alpha)
  x[4] = (std::abs(_x[3]) <= epsilon) ? _x[4] : _x[4] + s * log(_x[3]) - 0.5 * s

  return x;
};

std::vector<double> model_heston::exp_V1(double s, std::vector<double> _x)
{
  std::vector<double> x(_x.size());

  x[0] = _x[0] * exp(s * sqrt(std::abs(_x[1])));
  x[1] = _x[1];
  x[2] = _x[2];

  return x;
}

std::vector<double> model_heston_var_control::exp_V1(double s, std::vector<doubl
{
  double epsilon = DBL_EPSILON;

  std::vector<double> x = model_heston::exp_V1(s, _x);

  double y0 = x0[1];
  double sth = sqrt(std::abs(theta));
  double a = sqrt(std::abs(exp(-alpha) * (y0 - theta) + theta));
  double b = sqrt(std::abs(y0));

  if (std::abs(y0 - theta) <= epsilon)
    x[3] = _x[3] * exp(s * sth);
  else
    {
      double ntemp = std::abs((a - sth) * (b + sth) / ((a + sth) * (b - sth)));
      x[3] = _x[3] * exp((s / alpha) * (2.*(b - a) + sth * log(ntemp))); ////???
```

```
    }

  x[4] = _x[4];

  return x;
}

std::vector<double> model_heston::exp_V2(double s, std::vector<double> _x)
{
  double epsilon = DBL_EPSILON;

  std::vector<double> x(_x.size());

  x[0] = _x[0];
  x[1] = (std::abs(s) <= epsilon) ? _x[1] : (beta * s * 0.5 + sqrt(std::abs(_x[1
  x[2] = _x[2];

  return x;
}

std::vector<double> model_heston_var_control::exp_V2(double s, std::vector<doubl
{
  std::vector<double> x = model_heston::exp_V2(s, _x);

  x[3] = _x[3];
  x[4] = _x[4];

  return x;
}


std::vector<double> model_heston::f_b(std::vector<double> _x, double _t)
{
  std::vector<double> x(_x.size());

  x[0] = nu * _x[0];
  x[1] = alpha * (theta - _x[1]);
  x[2] = _x[0];

  return x;
}
```

```
std::vector<double> model_heston_var_control::f_b(std::vector<double> _x, double
{
  double epsilon = DBL_EPSILON;

  std::vector<double> x = model_heston::f_b(_x, _t);

  x[3] = nu * _x[3];
  x[4] = (std::abs(_x[3]) <= epsilon) ? 0. : log(std::abs(_x[3]));

  return x;
}


std::vector<double> model_heston::f_sigma(std::vector<double> _x, double _t)
{
  std::vector<double> x(_x.size());

  x[0] = _x[0] * sqrt(std::abs(_x[1]));
  x[1] = beta * sqrt(std::abs(_x[1]));
  x[2] = 0.;

  return x;
}

std::vector<double> model_heston_var_control::f_sigma(std::vector<double> _x, do
{
  std::vector<double> x = model_heston::f_sigma(_x, _t);

  double temp = exp(-alpha * _t) * (x0[1] - theta) + theta;

  x[3] = _x[3] * sqrt(std::abs(temp));
  x[4] = 0.;

  return x;
}

double model_heston_var_control::f_control(std::vector<double> _x)
{
  double x = exp((1. / T) * _x[4]);
  double epsilon = DBL_EPSILON;
```

```
  return (x - K > epsilon) ? x - K : 0.;
}

double model_heston_var_control::f_esp(double &_nvar)
{
  double epsilon = DBL_EPSILON;

  double z0 = x0[3];
  double y0 = x0[1];
  double one_a = 1. / alpha;

  double nsigma = theta * T * T * T / 3. + (y0 - theta) * 2.*one_a * (T * T / 2.
  double nmean = 0.;

  if (std::abs(nsigma) <= epsilon)
    {
      nmean = (z0 * exp(nu * T / 2.) - K > epsilon) ? z0 * exp(nu * T / 2.) - K
      _nvar = 0.;
    }
  else
    {
      double a = -(y0 - theta) * 0.5 * one_a * (1. + exp(-alpha * T) * one_a / T
      double b = (log(K / z0) - a) * T / sqrt(nsigma);
      nmean = z0 * exp(a + nsigma * 0.5 / (T * T)) * cdf_nor(b - sqrt(nsigma) /

      _nvar = z0 * z0 * exp(2.*a + 2.*nsigma / (T * T)) * cdf_nor(b - 2.*sqrt(ns
    }

  return nmean;
}

std::vector<double> model_heston::f_1(std::vector<double> _x, double _h, std::ve
{
  return exp_V1(_rv[0] * sqrt(_h), exp_V2(_rv[1] * sqrt(_h), _x));
}

std::vector<double> model_heston::f_2(std::vector<double> _x, double _h, std::ve
{
  return exp_V2(_rv[1] * sqrt(_h), exp_V1(_rv[0] * sqrt(_h), _x));
}
```

```
std::vector<double> model_heston_var_control::f_1(std::vector<double> _x, double
{
  return model_heston::f_1(_x, _nstep, _rv);
}

std::vector<double> model_heston_var_control::f_2(std::vector<double> _x, double
{
  return model_heston::f_2(_x, _nstep, _rv);
}

#endif

#endif //PremiaCurrentVersion
```