Help

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

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

#ifndef model_h_
#define model_h_

//model of a stochastic differential equation
//functions f_b, f_sigma for Euler scheme
//(the function near dt, dW_t in SDE respectively)

//functions exp_V0, f_1, f_2 for Ninomiya-Victoir Scheme
//f_1 and f_2 give a sequance of (exp_V1...exp_Vd) or (exp_Vd...exp_V1)

//functions f_esp, f_control for a variance reduction technique.
//f_esp() gives a value exact of control variable
//f_control() gives a control variable.

//parameters: T - maturity, K - strike, x0 - vector initial

class model
{
public:
  virtual std::vector<double> exp_V0(double, std::vector<double>) = 0;
  virtual std::vector<double> f_b(std::vector<double>, double) = 0;
  virtual std::vector<double> f_sigma(std::vector<double>, double) = 0;
  virtual std::vector<double> f_1(std::vector<double>, double, std::vector<doubl
  virtual std::vector<double> f_2(std::vector<double>, double, std::vector<doubl

  virtual double f_control(std::vector<double>) = 0;
  virtual double f_esp(double &) = 0;

  double T;
  std::vector<double> x0;
```

```
  double K;

  virtual ~model() {};

};

//we redefine operator +, * for vector

std::vector<double> operator+(std::vector<double> _x, std::vector<double> _y)
{
  int ndim = (_x.size() <= _y.size()) ? _x.size() : _y.size();
  std::vector<double> nres(ndim);

  for (int i = 0; i < ndim; i++)
    nres[i] = _x[i] + _y[i];

  return nres;
}

std::vector<double> operator*(std::vector<double> _x, double _a)
{
  int ndim = _x.size();
  std::vector<double> nres(ndim);

  for (int i = 0; i < ndim; i++)
    nres[i] = _a * _x[i];

  return nres;
}

std::vector<double> operator*(double _a, std::vector<double> _x)
{
  int ndim = _x.size();
  std::vector<double> nres(ndim);

  for (int i = 0; i < ndim; i++)
    nres[i] = _a * _x[i];

  return nres;
}
```

```
std::vector<double> operator*(std::vector<double> _x, std::vector<double> _y)
{
  int ndim = (_x.size() <= _y.size()) ? _x.size() : _y.size();
  std::vector<double> nres(ndim);

  for (int i = 0; i < ndim; i++)
    nres[i] = _x[i] * _y[i];

  return nres;
}


#endif

#endif //PremiaCurrentVersion
```