Help

```
#ifndef _REGRESSION_HPP
#define _REGRESSION_HPP

#include "pnl/pnl_matrix.h"
#include "pnl/pnl_basis.h"
#include "
href../../../../common/math/mcam/src/DupireModel_h_src.pdfModel.hpp"
#include <iostream>
#include <string>

namespace mcam {
class Regression
{
public:
    std::string label; //!< Martingale type name.
    int size; //!< Size of the brownian motion.
    int samples; //!< NUmber of independent samples
    int currentDate; //!< Date at which we compute the regression


    Regression();
    virtual ~Regression();

    /**
     * Compute the coefficients of the expansion of Y
     *
     * @param[out] p_alpha contains the coefficients on output. Must be initiali
     * @param p_Y vector of size samples containing some drawings of the variabl
     * @param p_Z vector of size samples containing the payoffs. We only keep pa
     */
    virtual void computeCoefficients(PnlVect *p_alpha, PnlVect *p_Y, PnlVect *p_

    /**
     * Compute the value of the regressor on a given path.
     *
     * @param pathNumber the index of the path to consider.
     * @param alpha the coefficients of the expansion
     */
    virtual double computeRegressorValue(int pathNumber, const PnlVect *alpha) c
```

```
    /**
     * Set the current date and create the regressor functions accordiingly
     *
     * @param p_currentDate the current date
     */
    virtual void setCurrentDate(int p_currentDate) = 0;

protected:
    PnlMat **regressors;

};

class RegressionPol : public Regression
{
public:
    int degree; //!< degree of the chaos expansion
    PnlBasis *basis;

    ~RegressionPol();
    RegressionPol(int p_size, int p_degree, PnlMat **p_regressors, int p_samples
    void setReduced(Model *p_mod);

    /**
     * Compute the coefficients of the expansion of Y
     *
     * @param[out] p_alpha contains the coefficients on output. Must be initiali
     * @param p_Y vector of size samples containing some drawings of the variabl
     * @param p_Z vector of size samples containing the payoffs. We only keep pa
     */
    virtual void computeCoefficients(PnlVect *p_alpha, PnlVect *p_Y, PnlVect *p_

    /**
     * Compute the value of the regressor on a given path.
     *
     * @param pathNumber the index of the path to consider.
     * @param alpha the coefficients of the expansion
     */
    virtual double computeRegressorValue(int pathNumber, const PnlVect *alpha) c

    /**
```

```
     * Set the current date and create the regressor functions accordiingly
     *
     * @param p_currentDate the current date
     */
    virtual void setCurrentDate(int p_currentDate);
};

class RegressionChaos : public Regression
{
public:
    int degree; //!< degree of the chaos expansion
    PnlBasis *basis;

    ~RegressionChaos();
    RegressionChaos(int p_size, int p_degree, PnlMat **p_regressors, int p_sampl

    /**
     * Compute the coefficients of the expansion of Y
     *
     * @param[out] p_alpha contains the coefficients on output. Must be initiali
     * @param p_Y vector of size samples containing some drawings of the variabl
     * @param p_Z vector of size samples containing the payoffs. We only keep pa
     */
    virtual void computeCoefficients(PnlVect *p_alpha, PnlVect *p_Y, PnlVect *p_

    /**
     * Compute the value of the regressor on a given path.
     *
     * @param pathNumber the index of the path to consider.
     * @param alpha the coefficients of the expansion
     */
    virtual double computeRegressorValue(int pathNumber, const PnlVect *alpha) c

    /**
     * Set the current date and create the regressor functions accordiingly
     *
     * @param p_currentDate the current date
     */
    virtual void setCurrentDate(int p_currentDate);
};
```

```
class RegressionChaosReduced : public Regression
{
public:
    int degree; //!< degree of the chaos expansion
    PnlBasis *basis;

    ~RegressionChaosReduced();
    RegressionChaosReduced(int p_size, int p_degree, PnlMat **p_regressors, int

    /**
     * Compute the coefficients of the expansion of Y
     *
     * @param[out] p_alpha contains the coefficients on output. Must be initiali
     * @param p_Y vector of size samples containing some drawings of the variabl
     * @param p_Z vector of size samples containing the payoffs. We only keep pa
     */
    virtual void computeCoefficients(PnlVect *p_alpha, PnlVect *p_Y, PnlVect *p_

    /**
     * Compute the value of the regressor on a given path.
     *
     * @param pathNumber the index of the path to consider.
     * @param alpha the coefficients of the expansion
     */
    virtual double computeRegressorValue(int pathNumber, const PnlVect *alpha) c

    /**
     * Set the current date and create the regressor functions accordiingly
     *
     * @param p_currentDate the current date
     */
    virtual void setCurrentDate(int p_currentDate);
};
}
#endif /* _REGRESSION_HPP */
```