Help

```
#ifndef _MART_HPP
#define _MART_HPP

#include <string>
#include "pnl/pnl_matrix.h"
#include "pnl/pnl_basis.h"

#include "
href../../../../common/math/jlparser/include/jlparser/parser_h_src.pdfjlparser/p
#include "
href../../../../common/math/mcam/src/DupireModel_h_src.pdfModel.hpp"
#include "
href../../../../common/math/mcam/src/workspace_h_src.pdfworkspace.hpp"

namespace mcam {

class Martingale
{
public:
    std::string label; //!< Martingale type name.
    int size; //!< Size of the underlying brownian motion.
    int subdates; //!< number of time steps on the finer grid. It corresponds to
    int nbFreedom; //!< number of degrees of freedom in the martingale approxima
    PnlVect_Workspace Mval; //!< Vector of the values of the martingale at all t
    PnlMat_Workspace Mgrad; //!< Matrix of the gradient at all times. A line cor

    virtual  void print() const;
    Martingale();
    virtual ~Martingale();

    /**
     * Compute the martingale at each time step and its gradient w.r.t. alpha.
     * The variables Mval and Mgrad are filled in with these values.
     *
     * @param G The renormalized Brownian increments
     * @param alpha The coefficients of the decomposition
     */
    virtual void computePath(const PnlMat *G, const PnlVect *alpha) = 0;
    /**
```

```
     * Return the gradient of the martingale at time \ a t
     *
     * @param t time expressed in terms of ticks within the finer grid
     *
     * @return Grad(M_t)
     */
    PnlVect gradat(int t)
    {
        return pnl_vect_wrap_mat_row(Mgrad(), t);
    }

    /**
     * Return the value of the martingale at time \ a t
     *
     * @param t time expressed in terms of ticks within the finer grid
     *
     * @return M_t
     */
    double at(int t)
    {
        return GET(Mval(), t);
    }
    /**
     * Compute the first two moments of the martingale
     *
     * @param rng random number genrator
     * @param alpha coefficients of the decomposition
     * @param iterationsMC number of Monte Carlo iterations
     * @param t time expressed in terms of ticks within the finer grid
     * @param[out] one E[M_t]
     * @param[out] two E[M_t^2]
     */
    void martingale_moments(PnlRng_Workspace &rng, const PnlVect *alpha, int ite
};

Martingale *instantiate_martingale(Model *mod, const Param &P, bool prune);

}
#endif /* _MART_HPP */
```