Help

```
#ifndef _WIENER_HPP
#define _WIENER_HPP

#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/Martingale_h_src.pdfMartingale.hpp"

namespace mcam {
/**
 *  @class Wiener.
 *  @brief Representation of Brownian martingales as stochastic integrals
 *
 *  This class is used to deal with martingales, which are defined as stochastic
 *  different possible integrands (one derived class per integrand family)
 *
 *  \ f[ \ int_0^t \ sum_{i=1}^n \ alpha_i f_i(W) dW \ f]
 *
 *  where n depends on the degree of the approximation and the number of increme
 *  The functions f_i's are basically PnlBases.
 */
class Wiener: public Martingale
{
public:

    int order; //!< order of the chaos expansion
    PnlBasis *Basis;
    Model *mod;

    Wiener();
    Wiener(const Param &P, Model *_mod);
    virtual ~Wiener();
    void print() const;

    // Be aware that all the dates are computed w.r.t the finer grid
    virtual void computePath(const PnlMat *G, const PnlVect *alpha);
```

```
protected:
    PnlMat_Workspace work;
};


/**
 * @class Trigo.
 * @brief Stochastic integrals with trigonometric integrands
 */
class Trigo: public Wiener
{
public:
    Trigo();
    Trigo(const Param &P, Model *_mod);
    virtual ~Trigo();
};

/**
 * @class Trigo.
 * @brief Stochastic integrals with polynomial integrands
 */
class Pol: public Wiener
{
public:
    Pol();
    Pol(const Param &P, Model *_mod);
    virtual void computePath(const PnlMat *G, const PnlVect *alpha);
    virtual ~Pol();
};
}
#endif /* _WIENER_HPP */
```