

Help

```
#ifndef _MOUNTAIN_H
#define _MOUNTAIN_H

#include "pnl/pnl_matrix.h"
#include "
href../../../../../common/math/jlparser/include/jlparser/parser_h_src.pdfjlparser/p
#include "
href../../../../../common/math/ImportanceSampling_jl/src/Option_h_src.pdfmath/Impor

/** Describe Atlas options.
 *
 * remove the <tt>nb_worst</tt> worst performances and
 * <tt>nb_best</tt> best performances and pay the average
 * performance of the remaining basket */
class AtlasOption : public BaseOption
{
public:
    AtlasOption();
    AtlasOption(const Param &ParamTab);
    ~AtlasOption();
    void print() const;
    int nb_worst;
    int nb_best;
    double payoff(const PnlMat *path_val);

private:
    PnlVect *final;

};

/** Describes Altiplano options
 *
 * If none of the stocks went lower than <tt>nominal</tt> %
 * of its spot between <tt>start_window</tt> and
 * <tt>end_window</tt> (expressed in % of the maturity) it
 * pays
 * 
$$1 + \left( \frac{1}{N} \sum_{i=1}^N \frac{S_{T^i}}{S_0^i} - 1 \right)_+$$

```

```

* \ f]
*/
class AltiplanoOption : public BaseOption
{
public:
    AltiplanoOption();
    AltiplanoOption(const Param &ParamTab);
    ~AltiplanoOption()
    {
    };
    void print() const;

    double nominal;
    /* barrier is expressed as a fraction of the spot */
    double barrier;
    /* the barrier is only effective between start_window and
    * end_window */
    double start_window, end_window;

    double payoff(const PnlMat *path_val);
};

/** Describes an Everest option
*
* Pays at maturity time
* \ f[
*  $1 + \min_i \left( \frac{S_{T^i}}{S_{0^i}} \right)$  \ right)
* \ f]
*/
class EverestOption : public BaseOption
{
public:
    EverestOption();
    EverestOption(const Param &ParamTab);
    ~EverestOption();
    void print() const;
    double payoff(const PnlMat *path_val);

private:

```

```
PnlVect *final;  
  
};  
  
#endif
```