

[Help](#)

```
#ifndef _DIGITAL_H
#define _DIGITAL_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

/** implements a digital option
 */
class DigitalFinalOption : public BaseOption
{
public:
    DigitalFinalOption();
    DigitalFinalOption(const Param &ParamTab);
    ~DigitalFinalOption();
    void print() const;

    /** the barriers can be different for each component */
    PnlVect *lower_barrier; /*!< the asset should be above this
                             barrier */
    PnlVect *upper_barrier; /*!< the asset should remain below this
                             barrier */

    double payoff(const PnlMat *path_val);
};

/** implements a digital option combined with a put/call option
 */
class DigitalBasketOption : public BaseOption
{
public:
    DigitalBasketOption();
    DigitalBasketOption(const Param &ParamTab);
    ~DigitalBasketOption();
    void print() const;
```

```

double K;
PnlVect *lambda;
/** the barriers can be different for each component */
PnlVect *lower_barrier; /*!< the asset should be above this
                           barrier */
PnlVect *upper_barrier; /*!< the asset should remain below this
                           barrier */

double payoff(const PnlMat *path_val);

};

/** implements a digital option
 */
class DigitalOption : public BaseOption
{
public:
    DigitalOption();
    DigitalOption(const Param &ParamTab);
    ~DigitalOption();
    void print() const;

    /** the barriers can be different for each component */
    PnlVect *lower_barrier; /*!< the asset should be above this
                              barrier */
    PnlVect *upper_barrier; /*!< the asset should remain below this
                              barrier */

    double payoff(const PnlMat *path_val);

};

/** implements a barrier basket option
 */
class BarrierOption : public BaseOption
{
public:
    BarrierOption();
    BarrierOption(const Param &ParamTab);
    ~BarrierOption();

```

```

void print() const;

/** the barriers can be different for each component */
PnlVect *lower_barrier; /*!< the asset should be above this
                           barrier */
PnlVect *upper_barrier; /*!< the asset should remain below this
                           barrier */

double K;
PnlVect *lambda;

double payoff(const PnlMat *path_val);

};

#endif

```