

## Help

```
#include "optype.h"
#include "var.h"
#include "method.h"
#include "test.h"
#include "timeinfo.h"
#include "error_msg.h"
#include "tools.h"
#ifndef SEEK_SET
#define SEEK_SET 0
#endif
#ifndef CLOCKS_PER_SEC
#include <unistd.h>
#define CLOCKS_PER_SEC _SC_CLK_TCK
#endif
extern char **error_msg;
extern char premiasrcdir[256];
extern char premiapersodir[256];
extern char *path_sep;
extern char PREMIA_OUT[256];
extern char GNUPLOT_DAT[256];
extern char TITLES_TEX[256];
extern char GNUPLOT_SCREEN_PLT[256];
extern char GNUPLOT_FILE_PLT[256];
extern char GNU_TEX[256];
extern char PREMIA_LOG[256];
extern char SESSION_LOG[256];
/*-----ITERATION_
-----*/
int Iterate(Planning *pt_plan,Iterator* pt_itera
tor,int count,char action,Model*pt_model,Option*pt
_option,Pricing *pt_pricing,PricingMethod* pt_
method,DynamicTest*pt_test,int user,TimeInfo *pt_time_
info)
{
    Iterator *next=pt_iterator+1;

    if (pt_iterator->Min.Vtype==END)
    { /*No iteration case*/
```

```

        (void)Action(action,pt_model,pt_option,pt_
pricing,pt_method,pt_test,user,pt_plan,pt_time_
info);

    }
else
{
    if (count>pt_iterator->StepNumber)
        return OK;
    else
    {
        if (count==0)
            *(pt_iterator->Location)=pt_it
erator->Min;

        if (next->Min.Vtype==END)
        {
            (void)ShowPlanning(VALUEONLY
TOFILE,pt_plan);

            (void)Action(action,pt_model
,pt_option,pt_pricing,pt_method,pt_test,VALUEONL
YTOFILE,pt_plan,pt_time_info);
            Fprintf(TOFILE,"{n");
        }
        else
            Iterate(pt_plan,next,0,action,
pt_model,pt_option,pt_pricing,pt_method,pt_test,u
ser,pt_time_info);

        (void)NextValue(count,pt_iterator)
;

        Iterate(pt_plan,pt_iterator,count+
1,action,pt_model,pt_option,pt_pricing,pt_
method,pt_test,user,pt_time_info);

    }
}

return OK;

```

```

}

void Action(char action, Model*pt_model, Option*pt_
    option, Pricing *pt_pricing, PricingMethod* pt_
    method,
        DynamicTest*pt_test, int user, Pl
    anning *pt_plan, TimeInfo *pt_time_info)
{
    int i, averaging, error=OK;
    double diff_time;
    clock_t start, finish;
    long j, number_of_runs;

    if ((pt_model->Check)(user, pt_plan, pt_model)==
        OK)
    {
        if ((pt_option->Check)(user, pt_plan, pt_opti
            on)==OK)
        {
            if ((pt_pricing->CheckMixing)(pt_
                option, pt_model)==OK)
            {
                if ((pt_method->Check)(user,
                    pt_plan, pt_method)==OK)
                {
                    if ((pt_time_info->Che
                        ck)(user, pt_plan, pt_time_info)==OK)
                    {
                        switch (action)
                        {
                            case 'p':

                                if (user!=
                                    NAMEONLYTOFILE)
                                {
                                    if (
                                        pt_time_info->Par[0].Val.V_INT==OK)
                                    {

```

```

averaging=pt_time_info->Par[1].Val.V_INT;

number_of_runs=pt_time_info->Par[2].Val.V_INT;

diff_time=0.;

for (i=0;i<averaging;i++)

{

    start=clock();

    for (j=0;j<number_of_runs;j++)

        error=(pt_method->Compute)(pt_option->
TypeOpt,pt_model->TypeModel,pt_method);

    finish=clock();

    diff_time+=((double)finish-(double)start)
/(double)CLOCKS_PER_SEC;

}

pt_time_info->Res[0].Val.V_DOUBLE=diff_time/(
double)averaging;

        }
        else

error=(pt_method->Compute)(pt_option->TypeOpt,pt_mod
el->TypeModel,pt_method);

        }

        if ((user=
=NAMEONLYTOFILE)|| (pt_plan->VarNumber>0))
        {
            Show
ResultTimeInfo(user,pt_plan,error,pt_time_info);

```

```

                                Show
ResultMethod(user,pt_plan,error,pt_method);
                                }
                                else
                                {
                                    Show
ResultTimeInfo(TOSCREEN,pt_plan,error,pt_time_info);
                                    Show
ResultTimeInfo(VALUEONLYTOFILE,pt_plan,error,pt_time_info);
                                    Show
ResultMethod(TOSCREEN,pt_plan,error,pt_method);
                                    Show
ResultMethod(VALUEONLYTOFILE,pt_plan,error,pt_method);
                                Fpri
ntf(TOFILE,"{n}");
                                }
                                break;

                                case 't':

                                    if ((pt_test->Check)(user,pt_plan,pt_test)==0)
                                    {

                                        if (
user!=NAMEONLYTOFILE)

error=(pt_test->Simul)(pt_option->TypeOpt,pt_model->TypeModel,pt_method,pt_test);

                                    Show
ResultTest(user,pt_plan,error,pt_test);

                                    }

                                break;

                                default :

```

```

                                break;
                            }
                        }
                    }
                }
            }

        return;
    }
    /*-----SELECTORS_
    -----*/
    int OutputFile(FILE **pt_file)
    {
        if ((*pt_file=fopen(PREMIA_OUT,"w"))==NULL)
        {
            Fprintf(TOSCREEN,"Unable to open output fi
            le{n");
            return 2;
        };

        return OK;
    }
    void InputMode(int *pt_user)
    {
        *pt_user=TOSCREEN;

        return;
    }
    char ChooseAction(void)
    {
        char msg='e';

        Fprintf(TOSCREEN,"{nPricing (p) or DynamicTest
        (t)?:{t");
        while ((msg!='t')&&(msg!='p'))
            scanf("%c",&msg);
        fflush(stdin);

```

```

    return msg;
}
int MoreAction(int *count)
{
    char msg='e';
    if (*count==(MAX_METHODS-1))
    {
        Fprintf(TOSCREEN, "{n Max Number of Methd
s Reached!{n");
        msg='n';
    } else
    {
        Fprintf(TOSCREEN, "{nMore Methods (ok:
Return, no: n)?:{t");

        while ((msg!='{n')&&(msg!='n'))
            scanf("%c",&msg);
        fflush(stdin);
    }
    if(msg=='n')
        return WRONG;
    else{
        (*count)++;
        return OK;
    }
}
int SelectModel(int user,Planning *pt_plan,Model
**listmodel,Model **mod)
{
    int choice=0, i=0;
    char fhelph_name[MAX_PATH_LEN]="";
    char msg,answer;

    if ((strlen(premiasrcdir)+strlen(path_sep)+str
len("mod_doc.pdf"))>=MAX_PATH_LEN)
    {
        Fprintf(TOSCREEN, "%s{n",error_msg[PATH_
TOO_LONG]);
        exit(WRONG);
    }

```

```

    }

    strcpy(fhhelp_name,"..");
    strcat(fhhelp_name,path_sep);
    strcat(fhhelp_name,"Src");
    strcat(fhhelp_name,path_sep);
    strcat(fhhelp_name,"mod_doc.pdf");

    Fprintf(TOSCREEN,"{n-----MOD
    EL CHOICE:{n{n}");

    while (listmodel[i]!=NULL)
    {
        Fprintf(TOSCREEN,"%s:{t%d{n",listmodel[i]->
        ID,i+1);
        i=i+1;
    }

    Fprintf(TOSCREEN,"{nChoice (h or any letter fo
    r help?):{t");
    do
    {
        msg = (char)tolower(fgetc(stdin));
        answer = msg;
        while( msg != '{n' && msg != EOF)
            msg = (char)fgetc(stdin);

        if (isalpha(answer) != 0)
        {
            choice=0;
#ifdef _WIN32
            _spawnlp(_P_WAIT,"AcroRd32.exe","_
            spawnlp",fhhelp_name,NULL);
#endif
#ifdef _WIN32
            _spawnlp(1,"acroread","_spawnlp",
            fhhelp_name,NULL);
#endif
            Fprintf(TOSCREEN,"{nNew value?:{t"
            );

```



```

    }
    else
        if(isdigit(answer) != 0)
        {
            choice = atoi(&answer);

            if ((choice<=0)|| (choice>i))
            {
                Fprintf(TOSCREEN, "{nBad
Choice: should range between 1 and %d ! New value?
:{t", i);
            }
        }

    } while ((choice<=0)|| (choice>i));
    Fprintf(TOSCREEN, "{n");

    if ((0<choice)&&(choice<=i))
    {
        *mod=listmodel[choice-1];
        return ((*mod)->Get)(user,pt_plan,*mod);
    }

    Fprintf(TOSCREEN, "Bad Choice!{n");

    return WRONG;
}
int SelectOption(int user,Planning *pt_plan,Fami
ly **L_listopt,Model* pt_model,Pricing **pricing,
Option **opt)
{
    int i,j,choice,k;
    Family* list;
    char family_name[MAX_CHAR_X3]="",dummy[MAX_CHA
R_X3]="";
    char msg,answer[3];

    do{
        Fprintf(TOSCREEN, "{n-----
OPTION CHOICE:{n{n");

```

```

i=0;
while (L_listopt[i] != NULL)
{
    if (MatchingPricing(user,pt_model,*(
L_listopt[i])[0],pricing)==0)
    {
        list=L_listopt[i];
        if ((strlen(premiasrcdir)+strlen
(path_sep))>=MAX_CHAR_X3)
        {
            Fprintf(TOSCREEN,"%s\n",
error_msg[PATH_TOO_LONG]);
            exit(WRONG);
        }
        strcpy(family_name,"..");
        strcat(family_name,path_sep);
        strcat(family_name,"Src");
        strcat(family_name,path_sep);
        if ((strlen("Opt")+2*strlen(pat
h_sep)+2*strlen((*list)[0]->ID)
+strlen("_doc.pdf"))>=
MAX_CHAR_X3)
        {
            Fprintf(TOSCREEN,"%s\n",
error_msg[PATH_TOO_LONG]);
            exit(WRONG);
        }

        strcpy(dummy,"opt");
        strcat(dummy,path_sep);
        strcat(dummy,(*list)[0]->ID);
        strcat(dummy,path_sep);
        strcat(dummy,(*list)[0]->ID);
        strcat(dummy,"_doc.pdf");
        for(k=0;k<(int)strlen(dummy);k+
+)
            dummy[k] = (char)tolower(dum
my[k]);
        if ((strlen(family_name)+strle
n(dummy))>=MAX_CHAR_X3))
        {

```

```

        Fprintf(TOSCREEN,"%s{n",
error_msg[PATH_TOO_LONG]);
        exit(WRONG);
    }

    strcat(family_name,dummy);
    Fprintf(TOSCREEN,"{nFamily{t%s{
n",(*list)[0]->ID);
    j=0;
    while ((*list)[j]!=NULL)
    {
        Fprintf(TOSCREEN,"%s:{t%
d{n",(*list)[j]->Name,j+1);
        j=j+1;
    }

    Fprintf(TOSCREEN,"{nChoice (0
for NextFamily, h for help)?:{t");
    do
    {
        k=0;
        msg=(char)tolower(fgetc(
stdin));
        answer[k++] = msg;
        while( (msg != '{n') && (
msg != EOF)){
            msg = (char)fgetc(stdi
n);
            if(k<=2)
                answer[k++]=msg;
        }
        answer[k--]='{0';
        if (isdigit(answer[0]) ==
0)
        {
            choice=j+1;
            if(answer[0] == 'h'
){
#ifdef _WIN32
                _spawnlp(_P_WAI
T,"AcroRd32.exe","_spawnlp",family_name,NULL);

```

```

#endif
#ifdef _WIN32
        _spawnlp(1,"acro
        read","_spawnlp",family_name,NULL);
#endif
        }
        Fprintf(TOSCREEN,"{
nNew value?:{t");

        } else {
        choice = atoi(answ
er);
        if ((choice<0)||
choice>j))
        {

                Fprintf(TOSCR
EEN,"{nBad Choice: should range between 1 and %d
! New value?:{t",j);
        }
        }
        } while ((choice<0)||
choice>j));

        Fprintf(TOSCREEN,"{n");

        if ((0<choice)&&(choice<=j))
        {
                *opt=(*list)[choice-1];
                return ((*opt)->Get)(use
r,pt_plan,*opt);
        }

        Fprintf(TOSCREEN,"{n");
        }
        i=i+1;/*choice=0*/
    }

    Fprintf(TOSCREEN,"No more families!{n");
} while(1);

```

```

    return WRONG;
}
int MatchingPricing(int user, Model *pt_model, Option
    *pt_option, Pricing **pricing)
{
    int i=-1;
    char dummy[MAX_CHAR_X3];

    if ((strlen(pt_model->ID)+1+strlen(pt_option->
        ID))>=MAX_CHAR_X3)
    {
        Fprintf(TOSCREEN, "%s\n", error_msg[PATH_
            TOO_LONG]);
        exit(WRONG);
    }

    strcpy(dummy, pt_model->ID);
    strcat(dummy, "_");
    strcat(dummy, pt_option->ID);
    do
    {
        i=i+1;
    }
    while ((strcmp(dummy, pricing[i]->ID)!=0) && (pr
        icing[i+1]!=NULL));

    if (strcmp(dummy, pricing[i]->ID)==0)
    {
        return OK;
    }
    return WRONG;
}
int SelectPricing(int user, Model *pt_model, Option
    *pt_option, Pricing **pricing, Pricing **result)
{
    int i=-1;
    char dummy[MAX_CHAR_X3];

    if ((strlen(pt_model->ID)+1+strlen(pt_option->
        ID))>=MAX_CHAR_X3)

```

```

    {
        Fprintf(TOSCREEN,"%s{\n",error_msg[PATH_
        TOO_LONG]);
        exit(WRONG);
    }

strcpy(dummy,pt_model->ID);
strcat(dummy,"_");
strcat(dummy,pt_option->ID);

do
{
    i=i+1;
}
while ((strcmp(dummy,pricing[i]->ID)!=0) && (pr
icing[i+1]!=NULL));

if (strcmp(dummy,pricing[i]->ID)==0)
{
    *result=pricing[i];
    return ((*result)->CheckMixing)(pt_option,
    pt_model) ;
}
Fprintf(TOSCREEN,"No choice available!{\n");

return WRONG;
}
int SelectMethod(int user,Planning *pt_plan,Pric
ing *pt_pricing, Option *opt,Model *mod,Pricing
Method **met)
{
    int i,isub,choice,sublist[MAX_MET],k;
    char msg,answer[3];

    PricingMethod** list;
    PricingMethod* dummy;

    Fprintf(TOSCREEN,"{\n-----
    METHOD CHOICE:{\n{\n");

    list=pt_pricing->Methods;

```

```

i=0;isub=0;

dummy=*list;

while (dummy !=NULL)

{
    if ( (dummy->CheckOpt)(opt,mod)==OK)

        {
            Fprintf(TOSCREEN,"%s:{t%d{n", (pt_
pricing->Methods[i])->Name,isub+1);
            sublist[isub]=i;
            isub=isub+1;
        }

        i=i+1;
        list++;dummy=*list;
    }

list=pt_pricing->Methods;
if (isub==0)
{
    Fprintf(TOSCREEN,"No methods available!{n")
;
}
else
{
    Fprintf(TOSCREEN,"{nChoice?:{t");
    do
    {
        k=0;
        msg = (char)tolower(fgetc(stdin));
        answer[k++] = msg;
        while( (msg != '{n') && (msg != EO
F)){
            msg = (char)fgetc(stdin);
            if(k<=2)
                answer[k++]=msg;
        }
        answer[k--]='{0';

```

```

        if (isdigit(answer[0]) == 0) {
            Fprintf(TOSCREEN, "{nNew value?:
{t}");
            choice=0;
        }
        else{
            choice = atoi(answer);
            if ((choice<=0)|| (choice>isub))
                {
                    Fprintf(TOSCREEN, "{nBad
Choice: should range between 1 and %d ! New value?
:{t", isub);
                }
            }
        } while ((choice<=0)|| (choice>is
ub));
        Fprintf(TOSCREEN, "{n");

        if ((0<choice)&&(choice<=isub))
        {
            *met=*(list+sublist[choice-1]);
            return GetMethod(user, pt_plan, pt_
pricing, *met);
        }
        else
            Fprintf(TOSCREEN, "Bad choice!{n");
    }
    return WRONG;

}

int SelectTest(int user, Planning *pt_plan, Pricing
    *pt_pricing, Option *opt, Model *mod, DynamicTes
    t **test)

{

    int i, isub, choice, sublist[MAX_MET], k;

    char msg, answer[3];

```



```
DynamicTest** list;

DynamicTest* dummy;

if (pt_plan->Action=='t')
{

Fprintf(TOSCREEN,"{n_-----
    TEST CHOICE:{n{n"});

list=pt_pricing->Test;

i=0;isub=0;

dummy=*list;

while ( dummy !=NULL)

    {

Fprintf(TOSCREEN,"%s:{t%d{n",dummy->Name,isu
b+1);

sublist[isub]=i;

isub=isub+1;

i=i+1;

list++;dummy=*list;

}

list=pt_pricing->Test;
```

```
if (isub==0)

{

    Fprintf(TOSCREEN,"No test available!\n");

}

else

{

    Fprintf(TOSCREEN,"{nChoice?:{t}");

    do

{

    k=0;

    msg = tolower(fgetc(stdin));

    answer[k++] = msg;

    while( (msg != '{n}') && (msg != EOF)){

        msg = fgetc(stdin);

        if(k<=2)

            answer[k++]=msg;

    }

    answer[k--]='{0}';

    if (isdigit(answer[0]) == 0)

        Fprintf(TOSCREEN,"{nNew value?:{t}");

    else{
```

```
        choice = atoi(answer);

        if ((choice<=0)|| (choice>isub))

        {

Fprintf(TOSCREEN, "{nBad Choice: should range between 1 and %d ! New value?:{t", isub);

        }

    }

}    while ((choice<=0)|| (choice>isub));

Fprintf(TOSCREEN, "{n");


        if ((0<choice)&&(choice<=isub))

{

        *test=*(list+sublist[choice-1]);

        return GetTest(user,pt_plan,pt_pricing,opt,*test);

}

        else

Fprintf(TOSCREEN, "Bad choice!{n");

    }


return WRONG;
}
```

```

    else
        return OK;
}

/*-----GNU FILES_
-----*/
#undef MAX_CHAR
#define MAX_CHAR 240 /* 3* previous MAX_CHAR*/
#undef MAX_COLS
#define MAX_COLS 100 /* = max number of columns
to be plotted */
void BuildGnuStuff(Planning plan,Model* model,
    Option* option, Pricing* pricing, PricingMethod **
    method)
{

    FILE *data,*gnu_data,*gnu_screen,*gnu_file,*g
    nu_tex;
    FILE *metfile;
    char line[MAX_CHAR];
    char last_car,car_read='e', dummy[MAX_CHAR];
    char xaxis[MAX_CHAR],yaxis[MAX_CHAR];
    char *y_axis[MAX_COLS], *z_axis[MAX_COLS];
    char gnutitle[MAX_CHAR];
    char fig[100][9];
    int typegraph;
    int graphno=0,figno[MAX_METHODS];
    int icount, treatline=OK;
    char *nom;
    int i,index, nodieze,vt,vt2,par1[MAX_METHODS],
        par2[MAX_METHODS];
#define XY 2
#define XYZ 3

#define GOTODBLEDIEZE(Y) do {last_car=car_read;
    car_read=(char)fgetc(Y);} while ( ((last_car!='#')
        ||(car_read!='#'))&& (!feof(Y)) )
#define GOTODIEZE(Y) do {car_read=(char)fgetc(Y)
    ;} while ( (car_read!='#')&& ( !feof(Y)) )
#define MKSTRING(s,X,Y,Z) strcpy(s,X);strcat(s,Y)

```

```

;strcat(s,Z)

/* ===== Create a datafile in the Gnu direc
tory: NECESSARY ?? =====*/
data=fopen(PREMIA_OUT,"r");
fseek(data, 0L, SEEK_SET);
gnu_data=fopen(GNUPLOT_DAT,"w");
fseek(gnu_data, 0L, SEEK_SET);
while (!feof(data))
{
    car_read=(char)fgetc(data);
    if (car_read!=(char)EOF)
        fputc(car_read,gnu_data);
}
fclose(data);
fclose(gnu_data);
/* =====
===== */
/* ===== Split & Fill "
    premia.out" ===== */
data=fopen(PREMIA_OUT,"r");
fseek(data, 0L, SEEK_SET);

strcpy(line,"{t}");
for (i=0; i<=plan.NumberOfMethods; i++)
{
    sprintf(dummy,"%s%d%s",method[i]->Name+3,i,
".dat");
    metfile=fopen(dummy,"w");
    par1[i]=0;
    par2[i]=0;
    nodieze=0;
    /* first copy the '#' commented lines and
fill missing labels */
    do {
        if (i==0 && nodieze==4)
        {
            if (treatline==OK)
                fprintf(metfile,"%s{n",line)
;
            fgets(line,sizeof(line),data);

```

```

        line[strlen(line)-1]='{0';
        /*      strcpy(dummy,"#"); */
        strcpy(dummy,(plan.Par)[0].Location->Vname);
        if (CompareParameterNames(line,
dummy)!=OK)
        {
            fprintf(metfile,"%s{n", (
plan.Par)[0].Location->Vname);
            par1[i]=1;
            treatline=WRONG;
        }
        nodieze++;
    }
    else if (i==0 && nodieze==6 && plan.
VarNumber==2)
    {
        if (treatline==OK)
            fprintf(metfile,"%s{n",line)
;

        fgets(line,sizeof(line),data);
        line[strlen(line)-1]='{0';
        /*      strcpy(dummy,"#"); */
        strcpy(dummy,(plan.Par)[1].Location->Vname);
        if (CompareParameterNames(line,
dummy)!=OK)
        {
            fprintf(metfile,"%s{n", (
plan.Par)[1].Location->Vname);
            par2[i]=2;
            treatline=WRONG;
        }
        fprintf(metfile,"##{n");
        nodieze++;
    }
    else if (i>0 && nodieze==1)
    {
        if (treatline==OK)
            fprintf(metfile,"%s{n",line)
;

```

```

        fgets(line,sizeof(line),data);
        line[strlen(line)-1]='\0';
        /*      strcpy(dummy,"#");    */
        strcpy(dummy,(plan.Par)[0].Location->Vname);
        if (CompareParameterNames(line,
dummy)!=OK)
        {
            fprintf(metfile,"#%s\n",
plan.Par)[0].Location->Vname);
            par1[i]=1;
            treatline=WRONG;
        }
        nodieze++;
    }
    else if (i>0 && nodieze==2 && plan.
VarNumber==2)
    {
        if (treatline==OK)
            fprintf(metfile,"%s\n",line)
;
        fgets(line,sizeof(line),data);
        line[strlen(line)-1]='\0';
        /*      strcpy(dummy,"#");    */
/
        strcpy(dummy,(plan.Par)[1].Location->Vname);
        if (CompareParameterNames(line,
dummy)!=OK)
        {
            fprintf(metfile,"#%s\n",
plan.Par)[1].Location->Vname);
            par2[i]=2;
            treatline=WRONG;
        }
        nodieze++;
    }
    else
    {
        fprintf(metfile,"%s\n",line);
        fgets(line,sizeof(line),data);

```

```

        line[strlen(line)-1]='{0';
        treatline=OK;
    }
    if (line[0]=='#' && line[1]=='#') no
dieze++;
    } while(!feof(data) && (line[0]=='#' ||
line[0]=='{0'));

/* now copy the data corresponding to that
Pricing Method */
while (!feof(data) && line[0]!='#')
{
    if (par1[i]==1 && par2[i]==0)
    {
        vt=(plan.Par)[0].Min.Vtype;
        if ((vt==PDOUBLE)|| (vt==DA
TE)|| (vt==RGDOUBLE)|| (vt==RGDOUBLE12)){
            fprintf(metfile,"%lf{t %
s{n", (plan.Par)[0].Min.Val.V_DOUBLE,line);
            fprintf(metfile,"%lf{t %
s{n", (plan.Par)[0].Max.Val.V_DOUBLE,line);
        }
        if ((vt==INT2) || (vt==BOOL)
){
            fprintf(metfile,"%ld{t %
s{n", (plan.Par)[0].Min.Val.V_INT,line);
            fprintf(metfile,"%ld{t %
s{n", (plan.Par)[0].Max.Val.V_INT,line);
        }
        fgets(line,sizeof(line),data
);
        line[strlen(line)-1]='{0';
    }
    else if (par1[i]==0 && par2[i]==2)
    {
        vt=(plan.Par)[1].Min.Vtype;
        if ((vt==PDOUBLE)|| (vt==DA
TE)|| (vt==RGDOUBLE)|| (vt==RGDOUBLE12)){
            fprintf(metfile,"%s{t %lf
{n",line, (plan.Par)[1].Min.Val.V_DOUBLE);

```



```

        fprintf(metfile,"%s\t %lf\n",line,(plan.Par)[1].Max.Val.V_DOUBLE);
    }
    if ((vt==INT2) || (vt==BOOL)
){
        fprintf(metfile,"%s\t %ld\n",line,(plan.Par)[1].Min.Val.V_INT);
        fprintf(metfile,"%s\t %ld\n",line,(plan.Par)[1].Max.Val.V_INT);
    }
    fgets(line,sizeof(line),data);

    line[strlen(line)-1]='\0';
}
else if (par1[i]==1 && par2[i]==2)
{
    vt=(plan.Par)[0].Min.Vtype;
    vt2=(plan.Par)[1].Min.Vtype;
    if ((vt==PDOUBLE)|| (vt==DATE)|| (vt==RGDOUBLE)|| (vt==RGDOUBLE12))
        vt=DOUBLE;
    if ((vt==INT2) || (vt==BOOL)
)
        vt=INT;
    if ((vt2==PDOUBLE)|| (vt2==DATE)|| (vt2==RGDOUBLE)|| (vt2==RGDOUBLE12))
        vt2=DOUBLE;
    if ((vt2==INT2) || (vt2==BO
OL))
        vt2=INT;
    if (vt==DOUBLE && vt2==
DOUBLE)
    {
        fprintf(metfile,"%lf{
t%s\t%lf\n", (plan.Par)[0].Min.Val.V_DOUBLE,line,(
plan.Par)[1].Min.Val.V_DOUBLE);
        fprintf(metfile,"%lf{
t%s\t%lf\n", (plan.Par)[0].Min.Val.V_DOUBLE,line,(
plan.Par)[1].Max.Val.V_DOUBLE);
        fprintf(metfile,"%lf{
t%s\t%lf\n", (plan.Par)[0].Max.Val.V_DOUBLE,line,(

```

```

plan.Par)[1].Max.Val.V_DOUBLE);
        fprintf(metfile,"%lf{
t%s{t%lf{n", (plan.Par)[0].Max.Val.V_DOUBLE,line,(
plan.Par)[1].Max.Val.V_DOUBLE);
        }
        else if (vt==DOUBLE && vt2==
INT)
        {
            fprintf(metfile,"%lf{
t%s{t%ld{n", (plan.Par)[0].Min.Val.V_DOUBLE,line,(
plan.Par)[1].Min.Val.V_INT);
            fprintf(metfile,"%lf{
t%s{t%ld{n", (plan.Par)[0].Min.Val.V_DOUBLE,line,(
plan.Par)[1].Min.Val.V_INT);
            fprintf(metfile,"%lf{
t%s{t%ld{n", (plan.Par)[0].Max.Val.V_DOUBLE,line,(
plan.Par)[1].Min.Val.V_INT);
            fprintf(metfile,"%lf{
t%s{t%ld{n", (plan.Par)[0].Max.Val.V_DOUBLE,line,(
plan.Par)[1].Min.Val.V_INT);
        }
        else if (vt==INT && vt2==
DOUBLE)
        {
            fprintf(metfile,"%ld{
t%s{t%lf{n", (plan.Par)[0].Min.Val.V_INT,line,(pl
an.Par)[1].Min.Val.V_DOUBLE);
            fprintf(metfile,"%ld{
t%s{t%lf{n", (plan.Par)[0].Min.Val.V_INT,line,(pl
an.Par)[1].Min.Val.V_DOUBLE);
            fprintf(metfile,"%ld{
t%s{t%lf{n", (plan.Par)[0].Max.Val.V_INT,line,(pl
an.Par)[1].Min.Val.V_DOUBLE);
            fprintf(metfile,"%ld{
t%s{t%lf{n", (plan.Par)[0].Max.Val.V_INT,line,(pl
an.Par)[1].Min.Val.V_DOUBLE);
        }
        else if (vt==INT && vt2==
INT)
        {
            fprintf(metfile,"%ld{

```

```

t%s{t%ld{n", (plan.Par)[0].Min.Val.V_INT,line,(pl
an.Par)[1].Min.Val.V_INT);
        fprintf(metfile,"%ld{
t%s{t%ld{n", (plan.Par)[0].Min.Val.V_INT,line,(pl
an.Par)[1].Min.Val.V_INT);
        fprintf(metfile,"%ld{
t%s{t%ld{n", (plan.Par)[0].Max.Val.V_INT,line,(pl
an.Par)[1].Min.Val.V_INT);
        fprintf(metfile,"%ld{
t%s{t%ld{n", (plan.Par)[0].Max.Val.V_INT,line,(pl
an.Par)[1].Min.Val.V_INT);
    }
    fgets(line,sizeof(line),data
);
    line[strlen(line)-1]='{0';
    }
    else
    {
        fprintf(metfile,"%s{n",line)
;
        fgets(line,sizeof(line),data
);
        line[strlen(line)-1]='{0';
    }
}

    fclose(metfile);
}
fclose(data);
/* =====
===== */
/* Getting labels from Planning directly */
gnu_screen=fopen(GNUPLOT_SCREEN_PLT,"w");
gnu_file=fopen(GNUPLOT_FILE_PLT,"w");

if (plan.VarNumber==2) /*3D-graph*/
{
    typegraph=XYZ;
    strcpy(xaxis,(char *) (plan.Par)[0].Locatio
n);
    strcpy(yaxis,(char *) (plan.Par)[1].Locatio

```

```

    n);
}
else /*2D-graph*/
{
    typegraph=XY;
    strcpy(xaxis,(char *) (plan.Par)[0].Location);
}
for (index=0; index<=plan.NumberOfMethods; index++)
{
    sprintf(dummy,"%s%d%s",method[index]->Name+3,index,".dat");
    metfile=fopen(dummy,"r");
    switch (typegraph)
    {
        case XY:
            if (index==0){
                GOTODBLEDEZE(metfile);
                GOTODBLEDEZE(metfile);
            }
            GOTODBLEDEZE(metfile);
            GOTODBLEDEZE(metfile);
            GOTODBLEDEZE(metfile);
            if (index==0){
                do
                {
                    GOTODIEZE(metfile);
                    fscanf(metfile,"%s",dummy);
                } while(!feof(metfile));
            }
            y_axis[graphno]=(char *) malloc(MAX_CHAR);
            if(y_axis[graphno] == NULL){
                Fprintf(TOSCREEN,"allocation error - aborting");
                exit (0);
            }
            strcpy(y_axis[graphno++], dummy);
        } while(!feof(metfile));
    }
}

```

```

    }
    break;
case XYZ:
    if (index==0)
    {
        GOTODBLEDIEZE(metfile);
        GOTODBLEDIEZE(metfile);
    }
    GOTODBLEDIEZE(metfile);
    GOTODBLEDIEZE(metfile);
    GOTODBLEDIEZE(metfile);
    figno[index]=0;
    while(!feof(metfile))
    {
        GOTODIEZE(metfile);
        fscanf(metfile,"%s",dummy);
        z_axis[graphno]=(char *)mal
loc(MAX_CHAR);
        if(z_axis[graphno] == NULL){

            Fprintf(TOSCREEN,"allocat
ion error - aborting");
            exit (0);
        }
        strcpy(z_axis[graphno++],dum
my);
        figno[index]++;
    }
    break;
default:
    break;
}

fclose(metfile);
}
/* ===== START the Gnuplot scripts =====
= */
begin_gnu(gnu_screen);
fprintf(gnu_screen,"set terminal {n}");

begin_gnu(gnu_file);

```

```

fprintf(gnu_file,"set terminal latex {n}");

/*Completing the new Gnu files*/

strcpy(gnutitle,"Model:");
strcat(gnutitle,model->Name);
strcat(gnutitle,"/Option:");
strcat(gnutitle,option->Name);
/*
    strcat(gnutitle,"/PricingMethod:");
    strcat(gnutitle,method->Name);
*/
MKSTRING(dummy,"set title {\"",gnutitle,"\"{n}");
fprintf(gnu_screen,dummy);

MKSTRING(dummy,"set xlabel {\"",xaxis,"\"{n}");

fprintf(gnu_screen,dummy);
fprintf(gnu_file,dummy);
switch (typegraph)
{
case XY:
    strcpy(y_axis[graphno-1],"The End!");
    for(icount=0; icount<graphno-1; icount++)
    {
        sprintf(&fig[icount][0],"fig%d.tex",icount);
        fprintf(gnu_file,"set output {\"%s{",
        "{n",fig[icount]);
        MKSTRING(dummy,"set ylabel {\"",y_
axis[icount],"\"{n}");
        fprintf(gnu_screen,dummy);
        fprintf(gnu_file,dummy);
        free(y_axis[icount]);
        fprintf(gnu_screen,"plot ");
        fprintf(gnu_file,"plot ");
        for (index=0; index<plan.NumberOfM
ethods; index++)
        {
            sprintf(dummy,"%s%d%s",
method[index]->Name+3,index,".dat");

```

```

        fprintf(gnu_screen, "{%"s{" u
sing 1:%d lt %d,", dummy, icount+2, icount+index+1);
        fprintf(gnu_file, "{%"s{" usi
ng 1:%d lt %d,", dummy, icount+2, icount+index);
    }
    sprintf(dummy, "%s%d%s", method[plan
.NumberOfMethods]->Name+3, plan.NumberOfMethods, "
.dat");
    fprintf(gnu_screen, "{%"s{" using 1
:%d lt %d", dummy, icount+2, icount+plan.NumberOfM
ethods+1);
    fprintf(gnu_file, "{%"s{" using 1:%
d lt %d", dummy, icount+2, icount+plan.NumberOfMetho
ds);
    fprintf(gnu_screen, "{\npause -1 {"
Next graph: %s{"{n", y_axis[icount+1]);
    fprintf(gnu_file, "{n");
}
break;
case XYZ:
    MKSTRING(dummy, "set ylabel {"", yaxis, "{n"
);
    fprintf(gnu_screen, dummy);
    fprintf(gnu_file, dummy);
    fprintf(gnu_screen, "set parametric{n");
    fprintf(gnu_file, "set parametric{n");
    for (index=0; index<=plan.NumberOfMethods;
index++) {
        for(icount=0; icount<figno[index]-1;
icount++) {
            sprintf(&fig[icount][0], "fig%d.tex
", icount+index);
            fprintf(gnu_file, "set output {"s{"
{n", fig[icount+index]);
            MKSTRING(dummy, "set xlabel {"", z_
axis[icount+index], "{n");
            fprintf(gnu_screen, dummy);
            fprintf(gnu_file, dummy);
            free(z_axis[icount+(figno[index]-1
)*index]);
            sprintf(dummy, "%s%d%s", method[ind

```

```

ex]->Name+3,index,".dat");
    if (par2[index]!=0)
    {
        fprintf(gnu_screen,"splot {"
%s{"{t using 1:%d:%d lt %d{n",dummy,1+figno[ind
ex],icount+2,icount+index+1);
        fprintf(gnu_screen,"pause -1
{"NEXT : Plot{"{n");
        fprintf(gnu_file,"splot {"%
s{"{t using 1:%d:%d lt %d{n",dummy,1+figno[index]
,icount+2,icount+index+1);
    }
    else
    {
        fprintf(gnu_screen,"splot {"
%s{"{t using 1:2:%d lt %d{n",dummy,icount+3,icou
nt+index+1);
        fprintf(gnu_screen,"pause -1
{"NEXT : Plot{"{n");
        fprintf(gnu_file,"splot {"%
s{"{t using 1:2:%d lt %d{n",dummy,icount+3,icount
+index+1);
    }
}
}
break;
default:
    break;
}
fclose(gnu_screen);
fclose(gnu_file);

/* ===== Now make the LaTeX output
t file ===== */

if ( make_titles_file(TITLES_TEX,model,option,
pricing,method,plan.NumberOfMethods) == OK)
{
    gnu_tex=fopen(GNU_TEX,"w");

    MKSTRING(dummy,"mod/",model->ID,"/");

```



```

    strcat(dummy,pricing->ID);
    strcat(dummy,"/");
    nom=dummy;
    while(*nom) *nom++ = (char)tolower(*nom);
    fprintf(gnu_tex,"%%%s{s{n","../Src/",dummy);
my);
    for (i=0; i<=plan.NumberOfMethods; i++)
    {
        strcpy(dummy,method[i]->Name);
        nom=dummy;
        while(*nom) *nom++ = (char)tolower(*nom);
er(*nom);
        fprintf(gnu_tex,"%%%s{s{n",dummy);
    }
    strcpy(dummy,pricing->ID);
    strcat(dummy,"/");
    nom=dummy;
    while(*nom) *nom++ = (char)tolower(*nom);
    fprintf(gnu_tex,"%%%s{s{n",dummy);

begin_tex_file(gnu_tex);

for(icount=0; icount<graphno-1; icount++)
{
    fprintf(gnu_tex,"{{input{%s}{n{n",
TITLES_TEX);
    fprintf(gnu_tex,"{{vspace{1.5cm}{
n");
    fprintf(gnu_tex,"{{input{%s}{n",fig[icount]);
    if (icount<(graphno-2))
        fprintf(gnu_tex,"{{newpage{n");
    }
end_tex_file(gnu_tex);
fclose(gnu_tex);
}
/* =====
===== */

Fprintf(TOSCREEN,"{nGnuplot data file:{t{s{n",G

```

```

    NU PLOT_DAT);

    FurtherMsg();

    return;
}
int make_titles_file(char *name, Model* pt_model,
    Option* pt_option, Pricing *pt_pricing, Pricing
    Method **pt_method, int metno)
{
    FILE *titles_tex, *metfile;
    char doc_pdf_name[MAX_CHAR], line[MAX_CHAR], dummy
        my[MAX_CHAR];
    char *p;
    int i, index, lineno=0;
    titles_tex=fopen(name, "w");

    fprintf(titles_tex, "{{begin{alltt}}{n");

    /* ----- Write now parameters from each
        method-file to "titles_tex" ----- */

    i=0;
    for (index=0; index<=metno; index++)
    {
        sprintf(dummy, "%s%d%s", pt_method[index]->Name+3, index, ".dat");
        if( (metfile=fopen(dummy, "r")) == NULL)
        {
            Fprintf(TO_SCREEN, "Unable to open
the data file{n");
            return 2;
        }
        fseek(metfile, 0L, SEEK_SET);
        do
        {
            fgets(line, sizeof(line), metfile);
            line[strlen(line)-1]='{0';
            if (line[0]=='#' && line[1]=='#' &
& line[2]!='{0')

```

```

        {
            i++;
            switch (i)
            {
                case 1:
                    /* ----- Links for th
e MODEL ----- */
                    strcpy(doc_pdf_name, ".
./");
                    strcat(doc_pdf_name, pt
_model->ID);
                    strcat(doc_pdf_name, "_
doc.pdf");
                    p=doc_pdf_name;
                    while(*p) *p++ = (cha
r)tolower(*p);
                    fprintf(titles_tex, "{{
textcolor{darkblue}{Model:}}{href{%s}{%s}{n", doc_
pdf_name, pt_model->Name);
                    linenno++;
                    break;

                case 2:
                    /* ---- Links for th
e OPTION ----- */
                    strcpy(doc_pdf_name, ".
./.../.../opt/");
                    strcat(doc_pdf_name, pt
_option->ID);
                    strcat(doc_pdf_name, "/"
);
                    strcat(doc_pdf_name, pt
_option->Name);
                    strcat(doc_pdf_name, "_
doc.pdf");
                    p=doc_pdf_name;
                    while(*p) *p++ = (cha
r)tolower(*p);
                    fprintf(titles_tex, "{{
textcolor{darkblue}{Option:}}{href{%s}{%s}{n", doc_
pdf_name, pt_option->Name);

```

```

        lineno++;
        break;

        default:
            /* ----- Links for ea
ch PRICING METHOD ----- */
            strcpy(doc_pdf_name,pt
_method[index]->Name);
            strcat(doc_pdf_name,"_
doc.pdf");
            p=doc_pdf_name;
            while(*p) *p++ = (cha
r)tolower(*p);
            fprintf(titles_tex,"{{
textcolor{darkblue}{Pricing Method:}}{{href{%s}{%s}
{n",doc_pdf_name,pt_method[index]->Name);
            lineno++;
            break;
        }
    }
    else if (line[0] == '#' && line[1]
!='#')
    {
        fprintf(titles_tex,"%s{n",
line+1);
        lineno++;
        if(lineno==40){
            lineno=0;
            fprintf(titles_tex,"{{new
page{n");
        }
    }
    } while(!feof(metfile) && line[0]!='{
0');

    fclose(metfile);
}
fprintf(titles_tex,"{{end{alltt}{n");
fclose(titles_tex);

return OK;

```

```

}
void begin_tex_file(FILE *f_tex)
{
    fprintf(f_tex, "{{documentclass[12pt,a4paper]{ar
        ticle}}{n");
    fprintf(f_tex, "{{input premiamblegraph{n");
    fprintf(f_tex, "{{input premiadata{n");

    fprintf(f_tex, "{{begin{document}}{n");
    fprintf(f_tex, "{{bigus{n");
}
void end_tex_file(FILE *f_tex)
{
    fprintf(f_tex, "{n");
    fprintf(f_tex, "{{input premiaendnobib{n");
    fprintf(f_tex, "{{end{document}}{n");
}
void begin_gnu(FILE *fp)
{
    fprintf(fp, "set noclip points{n");
    fprintf(fp, "set clip one{n");
    fprintf(fp, "set noclip two{n");
    fprintf(fp, "set border{n");
    fprintf(fp, "set boxwidth{n");
    fprintf(fp, "set dummy x,y{n");
    fprintf(fp, "set format x {%c%c{"{n", '%', 'g');
    fprintf(fp, "set format y {%c%c{"{n", '%', 'g');
    fprintf(fp, "set format z {%c%c{"{n", '%', 'g');
    fprintf(fp, "set grid{n");
    fprintf(fp, "set key{n");
    fprintf(fp, "set nolabel{n");
    fprintf(fp, "set noarrow{n");
    fprintf(fp, "set nologscale{n");
    fprintf(fp, "set offsets 0, 0, 0, 0{n");
    fprintf(fp, "set nopolar{n");
    fprintf(fp, "set angles radians{n");
    fprintf(fp, "set data style lines{n");
    fprintf(fp, "##{n");
}
/*-----MSGS-----
    -----*/

```

```

void WellcomeMsg(int user)
{
    Fprintf(user,"%s","*****
    *****{n{tWELCOME TO PREMIA{n*****
    *****{n{n");
    return;
}
int NextSession(Planning* pt_plan,char action,
    int user)
{
    char msg='t';
    FILE *logfile;

    if ((pt_plan->VarNumber>0) || (action=='t'))
    {
        if( (logfile=fopen(PREMIA_LOG,"w")) == NUL
        L)
        {
            Fprintf(TOSCREEN,"Unable to open
            the log file{n");
            return 2;
        }

        fprintf(logfile,"%s","PREMIA{n");
        fclose(logfile);

    }
    Fprintf(user,"{nNext Session?(n next session,
    e to exit){n");

    while ((msg!='e')&&(msg!='n'))
        scanf("%c",&msg);
    if (msg=='e')
    {
        if( (logfile=fopen(SESSION_LOG,"w")) == NUL
        L)
        {
            Fprintf(TOSCREEN,"Unable to open
            the session file{n");
            return 2;
        }
    }
}

```

```

        fprintf(logfile,"%s","PREMIA{n");
        fclose(logfile);
    }
    return (msg=='e');
}
void FurtherMsg(void)
{
    Fprintf(TOSCREEN,"%s","*****
    *****
    {n");
#ifdef _WIN32
    Fprintf(TOSCREEN,"%s %s{n","1. To view the resu
    lts on the screen execute:{tgnuplot (or wgnuplot)
    ",GNUPLOT_SCREEN_PLT);
    Fprintf(TOSCREEN,"%s","2. To preview a PDF outp
    ut file on this run enter: {n");
    Fprintf(TOSCREEN,"%s","          make output; ac
    roread gnupremia.pdf{n");
    Fprintf(TOSCREEN,"%s","3. To archive (PDF) this
    last run: {n");
    Fprintf(TOSCREEN,"%s","          csh premia_ar
    chive{n");
#endif
#ifdef _WIN32
    Fprintf(TOSCREEN,"%s %s{n","1. To view the resu
    lts on the screen execute:{tgnuplot (or wgnuplot)
    ",GNUPLOT_SCREEN_PLT);
    Fprintf(TOSCREEN,"%s"," (WinEdt: nothing to
    do) {n");
    Fprintf(TOSCREEN,"%s","2. To preview a PDF outp
    ut file on this run execute: {n");
    Fprintf(TOSCREEN,"%s","          out2pdf.bat (
    WinEdt: Ctr+F11) {n");
    Fprintf(TOSCREEN,"%s","3. To archive (PDF) this
    last run: {n");
    Fprintf(TOSCREEN,"%s","          archivepdf.bat
    (WinEdt: Ctr+F12 {n");
#endif
    Fprintf(TOSCREEN,"%s","*****
    *****

```

```

        {n});
    return;
}
/* Desallocation of memory needed in Test->Res */
void FreeTest(DynamicTest *test)
{
    int i=0;
    while (test->Res[i].Vtype!=END)
    {
        if (test->Res[i].Vtype==DOUBLEARRAY)
            free(test->Res[i].Val.V_DOUBLEARRAY->
                array);
        i++;
    }
    return;
}

/* A buildgnustuf for Test output */
void BuildGnuStuffTest(Model* model,Option* option, Pricing* pricing, PricingMethod* method,Dyna
micTest* test)
{
    FILE *gnu_screen,*gnu_file,*gnu_tex,*titles_tex
        , *output;
    int typegraph=0, write, k;
    char doc_pdf_name[MAX_CHAR], *p, *nom, dummy[
        MAX_CHAR], line[MAX_CHAR];

    if (strcmp(method->Name,"TR_PatryMartini")==0)
        typegraph=5;
    else
    if (strcmp(option->ID,"STD")==0)
        typegraph=0;
    else if (strcmp(option->ID,"LIM")==0)
        typegraph=1;
    else if (strcmp(option->ID,"DOUBLIM")==0)
        typegraph=2;
    else if (strcmp(option->ID,"PAD")==0)
        typegraph=3;
    else if (strcmp(option->ID,"STD2D")==0)
        typegraph=4;

```



```

/* GNU TO FILE */

if ((gnu_file=fopen(GNUPLOT_FILE_PLT,"w"))==
    NULL)
{
    Fprintf(TOSCREEN,"Unable to create the G
    nutofile file{n");
    return;
}
begin_gnu(gnu_file);
fprintf(gnu_file,"set terminal latex {n");

fprintf(gnu_file,"set size 1.,1.{nset origin 0
    .,0.{n");

switch (typegraph)
{
case 0: /*STD*/
    {
        fprintf(gnu_file,"set xlabel {"Time{"
        {n");
        /*Stockmin and PLmin*/
        fprintf(gnu_file,"set output {"fig0.
        tex{"{n");
        fprintf(gnu_file,"set title {"Stock'
        s trajectory to obtain the minimal PL.{n");
        fprintf(gnu_file,"set ylabel {"Stock{
        {n");
        fprintf(gnu_file,"plot {"%s{"{t using
        1:2 notitle ,{
        {"%s{"{t using 8:9 title {"Spot Target{" with po
        ints,{
        {"%s{"{t using 10:11 title {"Exercice Time{" wit
        h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT);
        fprintf(gnu_file,"set output {"fig1.
        tex{"{n");
        fprintf(gnu_file,"set title {"PL's
        trajectory.{n");
        fprintf(gnu_file,"set ylabel {"PL{"{

```

```

n");
    fprintf(gnu_file,"plot {"%s{"{t using
1:5 notitle{n",PREMIA_OUT);
    /*Stockmax and PLmax*/
    fprintf(gnu_file,"set output {"fig2.
tex{"{n");
    fprintf(gnu_file,"set title {"Stock'
s trajectory to obtain the maximal PL.{"{n");
    fprintf(gnu_file,"set ylabel {"Stock{
"{n");
    fprintf(gnu_file,"plot {"%s{"{t using
1:3 notitle ,{
{"%s{"{t using 8:9 title {"Spot Target{" with po
ints,{
{"%s{"{t using 10:11 title {"Exercice Time{" wit
h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT);
    fprintf(gnu_file,"set output {"fig3.
tex{"{n");
    fprintf(gnu_file,"set title {"PL's
trajectory.{"{n");
    fprintf(gnu_file,"set ylabel {"PL{"{
n");
    fprintf(gnu_file,"plot {"%s{"{t using
1:6 notitle{n",PREMIA_OUT);
    /*Stockmin and PLmin*/
    fprintf(gnu_file,"set output {"fig4.
tex{"{n");
    fprintf(gnu_file,"set title {"Stock'
s trajectory to obtain the median PL.{"{n");
    fprintf(gnu_file,"set ylabel {"Stock{
"{n");
    fprintf(gnu_file,"plot {"%s{"{t using
1:4 notitle ,{
{"%s{"{t using 8:9 title {"Spot Target{" with po
ints,{
{"%s{"{t using 10:11 title {"Exercice Time{" wit
h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT);
    fprintf(gnu_file,"set output {"fig5.
tex{"{n");
    fprintf(gnu_file,"set title {"PL's
trajectory.{"{n");

```

```

        fprintf(gnu_file,"set ylabel {"PL{"{
n"});
        fprintf(gnu_file,"plot {"%s{"{t using
1:7 notitle{n",PREMIA_OUT);
    }
    break;
case 1: /*LIM*/
    {
        fprintf(gnu_file,"set xlabel {"Time{"{
n"});
        /*Stockmin and PLmin*/
        fprintf(gnu_file,"set output {"fig0.
tex{"{n"});
        fprintf(gnu_file,"set title {"Stock'
s trajectory to obtain the minimal PL.{"{n"});
        fprintf(gnu_file,"set ylabel {"Stock{"{
n"});
        fprintf(gnu_file,"plot {"%s{"{t using
1:2 notitle ,{
{"%s{"{t using 1:14 title {"Limit{"{
{"%s{"{t using 15:16 title {"Spot Target{" with
points,{
{"%s{"{t using 17:18 title {"Exercice Time{" wit
h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT,PREM
IA_OUT);
        fprintf(gnu_file,"set output {"fig1.
tex{"{n"});
        fprintf(gnu_file,"set title {"PL's
trajectory.{"{n"});
        fprintf(gnu_file,"set ylabel {"PL{"{
n"});
        fprintf(gnu_file,"plot {"%s{"{t using
1:5 notitle{n",PREMIA_OUT);
        /*Stockmax and PLmax*/
        fprintf(gnu_file,"set output {"fig2.
tex{"{n"});
        fprintf(gnu_file,"set title {"Stock'
s trajectory to obtain the maximal PL.{"{n"});
        fprintf(gnu_file,"set ylabel {"Stock{"{
n"});
        fprintf(gnu_file,"plot {"%s{"{t using

```

```

        1:3 notitle ,{
{"%s{"{t using 1:14 title {"Limit{"{
{"%s{"{t using 15:16 title {"Spot Target{" with
points,{
{"%s{"{t using 17:18 title {"Exercice Time{" wit
h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT,PREM
IA_OUT);
        fprintf(gnu_file,"set output {"fig3.
tex{"{n");
        fprintf(gnu_file,"set title {"PL's
trajectory.{"{n");
        fprintf(gnu_file,"set ylabel {"PL{"{
n");
        fprintf(gnu_file,"plot {"%s{"{t using
1:6 notitle{n",PREMIA_OUT);
        /*Stockmedian and PLmedian*/
        fprintf(gnu_file,"set output {"fig4.
tex{"{n");
        fprintf(gnu_file,"set title {"Stock'
s trajectory to obtain the median PL.{"{n");
        fprintf(gnu_file,"set ylabel {"Stock{
{n");
        fprintf(gnu_file,"plot {"%s{"{t using
1:4 notitle ,{
{"%s{"{t using 1:14 title {"Limit{"{
{"%s{"{t using 15:16 title {"Spot Target{" with
points,{
{"%s{"{t using 17:18 title {"Exercice Time{" wit
h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT,PREM
IA_OUT);
        fprintf(gnu_file,"set output {"fig5.
tex{"{n");
        fprintf(gnu_file,"set title {"PL's
trajectory.{"{n");
        fprintf(gnu_file,"set ylabel {"PL{"{
n");
        fprintf(gnu_file,"plot {"%s{"{t using
1:7 notitle{n",PREMIA_OUT);
        /*Stockminbreached and PLminbreached*
/
        fprintf(gnu_file,"set output {"fig6.

```

```

tex{"{n}");
    fprintf(gnu_file,"set title {"Stock'
s trajectory to obtain the minimal PL after havi
ng reached the limit.{"{n}");
    fprintf(gnu_file,"set ylabel {"Stock{
"{n}");
    fprintf(gnu_file,"plot {"%s{"{t using
1:8 notitle ,{
{"%s{"{t using 1:14 title {"Limit{"{
{"%s{"{t using 15:16 title {"Spot Target{" with
points,{
{"%s{"{t using 17:18 title {"Exercice Time{" wit
h points{"n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT,PREM
IA_OUT);
    fprintf(gnu_file,"set output {"fig7.
tex{"{n}");
    fprintf(gnu_file,"set title {"PL's
trajectory.{"{n}");
    fprintf(gnu_file,"set ylabel {"PL{"{
n}");
    fprintf(gnu_file,"plot {"%s{"{t using
1:11 notitle{"n",PREMIA_OUT);
    /*Stockmaxbreached and PLmaxbreached*
/
    fprintf(gnu_file,"set output {"fig8.
tex{"{n}");
    fprintf(gnu_file,"set title {"Stock'
s trajectory to obtain the maximal PL after havi
ng reached the limit.{"{n}");
    fprintf(gnu_file,"set ylabel {"Stock{
"{n}");
    fprintf(gnu_file,"plot {"%s{"{t using
1:9 notitle ,{
{"%s{"{t using 1:14 title {"Limit{"{
{"%s{"{t using 15:16 title {"Spot Target{" with
points,{
{"%s{"{t using 17:18 title {"Exercice Time{" wit
h points{"n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT,PREM
IA_OUT);
    fprintf(gnu_file,"set output {"fig9.
tex{"{n}");

```

```

        fprintf(gnu_file,"set title {"PL's
trajectory.{"{n"});
        fprintf(gnu_file,"set ylabel {"PL{"{
n"});
        fprintf(gnu_file,"plot {"%s{"{t using
1:12 notitle{n",PREMIA_OUT);
        /*Stockmedianbreached and PLmedianbre
ached*/
        fprintf(gnu_file,"set output {"fig10.
tex{"{n"});
        fprintf(gnu_file,"set title {"Stock'
s trajectory to obtain the median PL after having
reached the limit.{"{n"});
        fprintf(gnu_file,"set ylabel {"Stock{
n"});
        fprintf(gnu_file,"plot {"%s{"{t using
1:10 notitle ,{
{"%s{"{t using 1:14 title {"Limit{"{
{"%s{"{t using 15:16 title {"Spot Target{" with
points,{
{"%s{"{t using 17:18 title {"Exercice Time{" wit
h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT,PREM
IA_OUT);
        fprintf(gnu_file,"set output {"fig11.
tex{"{n"});
        fprintf(gnu_file,"set title {"PL's
trajectory.{"{n"});
        fprintf(gnu_file,"set ylabel {"PL{"{
n"});
        fprintf(gnu_file,"plot {"%s{"{t using
1:13 notitle{n",PREMIA_OUT);
    }
    break;

case 2: /*DOUBLIM*/
{
    fprintf(gnu_file,"set xlabel {"Time{"
n"});
    /*Stockmin and PLmin*/
    fprintf(gnu_file,"set output {"fig0.
tex{"{n"});

```

```

        fprintf(gnu_file,"set title {"Stock'
s trajectory to obtain the minimal PL.{n"});
        fprintf(gnu_file,"set ylabel {"Stock{
n"});
        fprintf(gnu_file,"plot {"%s"{t using
1:2 notitle ,{
{"%s"{t using 1:14 title {"LowerLimit{",{
{"%s"{t using 1:15 title {"UpperLimit{",{
{"%s"{t using 16:17 title {"Spot Target{" with
points,{
{"%s"{t using 18:19 title {"Exercice Time{" wit
h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT,PREM
IA_OUT,PREMIA_OUT});
        fprintf(gnu_file,"set output {"fig1.
tex{n"});
        fprintf(gnu_file,"set title {"PL's
trajectory.{n"});
        fprintf(gnu_file,"set ylabel {"PL{"{
n"});
        fprintf(gnu_file,"plot {"%s"{t using
1:5 notitle{n",PREMIA_OUT});
        /*Stockmax and PLmax*/
        fprintf(gnu_file,"set output {"fig2.
tex{n"});
        fprintf(gnu_file,"set title {"Stock'
s trajectory to obtain the maximal PL.{n"});
        fprintf(gnu_file,"set ylabel {"Stock{
n"});
        fprintf(gnu_file,"plot {"%s"{t using
1:3 notitle ,{
{"%s"{t using 1:14 title {"LowerLimit{",{
{"%s"{t using 1:15 title {"UpperLimit{",{
{"%s"{t using 16:17 title {"Spot Target{" with
points,{
{"%s"{t using 18:19 title {"Exercice Time{" wit
h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT,PREM
IA_OUT,PREMIA_OUT});
        fprintf(gnu_file,"set output {"fig3.
tex{n"});
        fprintf(gnu_file,"set title {"PL's
trajectory.{n"});

```

```

        fprintf(gnu_file,"set ylabel {"PL{"{
n"});
        fprintf(gnu_file,"plot {"%s{"{t using
1:6 notitle{n",PREMIA_OUT);
        /*Stockmedian and PLmedian*/
        fprintf(gnu_file,"set output {"fig4.
tex{"{n"});
        fprintf(gnu_file,"set title {"Stock'
s trajectory to obtain the median PL.{"{n"});
        fprintf(gnu_file,"set ylabel {"Stock{
n"});
        fprintf(gnu_file,"plot {"%s{"{t using
1:4 notitle ,{
{"%s{"{t using 1:14 title {"LowerLimit{"{
{"%s{"{t using 1:15 title {"UpperLimit{"{
{"%s{"{t using 16:17 title {"Spot Target{" with
points,{
{"%s{"{t using 18:19 title {"Exercice Time{" wit
h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT,PREM
IA_OUT,PREMIA_OUT);
        fprintf(gnu_file,"set output {"fig5.
tex{"{n"});
        fprintf(gnu_file,"set title {"PL's
trajectory.{"{n"});
        fprintf(gnu_file,"set ylabel {"PL{"{
n"});
        fprintf(gnu_file,"plot {"%s{"{t using
1:7 notitle{n",PREMIA_OUT);
        /*Stockminbreached and PLminbreached*
/
        fprintf(gnu_file,"set output {"fig6.
tex{"{n"});
        fprintf(gnu_file,"set title {"Stock'
s trajectory to obtain the minimal PL after havi
ng reached the limit.{"{n"});
        fprintf(gnu_file,"set ylabel {"Stock{
n"});
        fprintf(gnu_file,"plot {"%s{"{t using
1:8 notitle ,{
{"%s{"{t using 1:14 title {"LowerLimit{"{
{"%s{"{t using 1:15 title {"UpperLimit{"{

```



```

{"%s{"{t using 16:17 title {"Spot Target{" with
  points,{
{"%s{"{t using 18:19 title {"Exercice Time{" wit
  h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT,PREM
  IA_OUT,PREMIA_OUT);
    fprintf(gnu_file,"set output {"fig7.
tex{"{n");
    fprintf(gnu_file,"set title {"PL's
trajectory.{"{n");
    fprintf(gnu_file,"set ylabel {"PL{"{
n");
    fprintf(gnu_file,"plot {"%s{"{t using
  1:11 notitle{n",PREMIA_OUT);
    /*Stockmaxbreached and PLmaxbreached*
/
    fprintf(gnu_file,"set output {"fig8.
tex{"{n");
    fprintf(gnu_file,"set title {"Stock'
s trajectory to obtain the maximal PL after havi
ng reached the limit.{"{n");
    fprintf(gnu_file,"set ylabel {"Stock{
{n");
    fprintf(gnu_file,"plot {"%s{"{t using
  1:9 notitle ,{
{"%s{"{t using 1:14 title {"LowerLimit{"},{
{"%s{"{t using 1:15 title {"UpperLimit{"},{
{"%s{"{t using 16:17 title {"Spot Target{" with
  points,{
{"%s{"{t using 18:19 title {"Exercice Time{" wit
  h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT,PREM
  IA_OUT,PREMIA_OUT);
    fprintf(gnu_file,"set output {"fig9.
tex{"{n");
    fprintf(gnu_file,"set title {"PL's
trajectory.{"{n");
    fprintf(gnu_file,"set ylabel {"PL{"{
n");
    fprintf(gnu_file,"plot {"%s{"{t using
  1:12 notitle{n",PREMIA_OUT);
    /*Stockmedianbreached and PLmedianbre
ached*/

```

```

        fprintf(gnu_file,"set output {"fig10.
tex{"{n");
        fprintf(gnu_file,"set title {"Stock'
s trajectory to obtain the median PL after having
reached the limit.{n");
        fprintf(gnu_file,"set ylabel {"Stock{
{n");
        fprintf(gnu_file,"plot {"%s{"{t using
1:10 notitle ,{
{"%s{"{t using 1:14 title {"LowerLimit{",{
{"%s{"{t using 1:15 title {"UpperLimit{",{
{"%s{"{t using 16:17 title {"Spot Target{" with
points,{
{"%s{"{t using 18:19 title {"Exercice Time{" wit
h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT,PREM
IA_OUT,PREMIA_OUT);
        fprintf(gnu_file,"set output {"fig11.
tex{"{n");
        fprintf(gnu_file,"set title {"PL's
trajectory.{n");
        fprintf(gnu_file,"set ylabel {"PL{"{
n");
        fprintf(gnu_file,"plot {"%s{"{t using
1:13 notitle{n",PREMIA_OUT);
    }
    break;
case 3: /*PAD*/
{
    fprintf(gnu_file,"set xlabel {"Time{"
{n");
    fprintf(gnu_file,"set size 1.,.7{n");
    /*Stockmin and PLmin*/
    fprintf(gnu_file,"set output {"fig0.
tex{"{n");
    fprintf(gnu_file,"set title {"Stock'
s trajectory to obtain the minimal PL.{n");
    fprintf(gnu_file,"set ylabel {"Stock{
{n");
    fprintf(gnu_file,"plot {"%s{"{t using
1:2 notitle ,{
{"%s{"{t using 11:12 title {"Spot Target{" with

```

```

    points,{
{"%s{"{t using 13:14 title {"Exercice Time{" wit
    h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT);
        fprintf(gnu_file,"set output {"fig2.
tex{"{n");
        fprintf(gnu_file,"set title {"PL's
trajectory.{"{n");
        fprintf(gnu_file,"set ylabel {"PL{"{
n");
        fprintf(gnu_file,"plot {"%s{"{t using
    1:5 notitle{n",PREMIA_OUT);
        fprintf(gnu_file,"set output {"fig1.
tex{"{n");
        fprintf(gnu_file,"set title {"PathDep
's trajectory.{"{n");
        fprintf(gnu_file,"set ylabel {"PathD
ep{"{n");
        fprintf(gnu_file,"plot {"%s{"{t using
    1:8 notitle{n",PREMIA_OUT);
        /*Stockmax and PLmax*/
        fprintf(gnu_file,"set output {"fig3.
tex{"{n");
        fprintf(gnu_file,"set title {"Stock'
s trajectory to obtain the maximal PL.{"{n");
        fprintf(gnu_file,"set ylabel {"Stock{
{n");
        fprintf(gnu_file,"plot {"%s{"{t using
    1:3 notitle ,{
{"%s{"{t using 11:12 title {"Spot Target{" with
    points,{
{"%s{"{t using 13:14 title {"Exercice Time{" wit
    h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT);
        fprintf(gnu_file,"set output {"fig5.
tex{"{n");
        fprintf(gnu_file,"set title {"PL's
trajectory.{"{n");
        fprintf(gnu_file,"set ylabel {"PL{"{
n");
        fprintf(gnu_file,"plot {"%s{"{t using
    1:6 notitle{n",PREMIA_OUT);
        fprintf(gnu_file,"set output {"fig4.

```

```

tex{"{n}");
    fprintf(gnu_file,"set title {"PathDep
's trajectory.{n}");
    fprintf(gnu_file,"set ylabel {"PathD
ep{"{n}");
    fprintf(gnu_file,"plot {"%s{"{t using
1:9 notitle{n",PREMIA_OUT);
    /*Stockmin and PLmin*/
    fprintf(gnu_file,"set output {"fig6.
tex{"{n}");
    fprintf(gnu_file,"set title {"Stock'
s trajectory to obtain the median PL.{n}");
    fprintf(gnu_file,"set ylabel {"Stock{
n}");
    fprintf(gnu_file,"plot {"%s{"{t using
1:4 notitle ,{
{"%s{"{t using 11:12 title {"Spot Target{" with
points,{
{"%s{"{t using 13:14 title {"Exercice Time{" wit
h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT);
    fprintf(gnu_file,"set output {"fig8.
tex{"{n}");
    fprintf(gnu_file,"set title {"PL's
trajectory.{n}");
    fprintf(gnu_file,"set ylabel {"PL{"{
n}");
    fprintf(gnu_file,"plot {"%s{"{t using
1:7 notitle{n",PREMIA_OUT);
    fprintf(gnu_file,"set output {"fig7.
tex{"{n}");
    fprintf(gnu_file,"set title {"PathDep
's trajectory.{n}");
    fprintf(gnu_file,"set ylabel {"PathD
ep{"{n}");
    fprintf(gnu_file,"plot {"%s{"{t using
1:10 notitle{n",PREMIA_OUT);
    fprintf(gnu_file,"set size 1.,1.{n}");
}
break;
case 4: /*STD2D*/
{

```

```

        fprintf(gnu_file,"set xlabel {"Time{"
{n");
        fprintf(gnu_file,"set size 1.,.7{n");
        /*Stockmin and PLmin*/
        fprintf(gnu_file,"set output {"fig0.
tex{"{n");
        fprintf(gnu_file,"set title {"First
Stock's trajectory to obtain the minimal PL.{n");
;
        fprintf(gnu_file,"set ylabel {"Stock1
{n");
        fprintf(gnu_file,"plot {"%s{"t using
1:2 notitle , {"%s{"t using 11:12 title {"Spot
Target{" with points,{
{"%s{"t using 15:16 title {"Exercice Time{" wit
h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT);
        fprintf(gnu_file,"set output {"fig2.
tex{"{n");
        fprintf(gnu_file,"set title {"PL's
trajectory.{n");
        fprintf(gnu_file,"set ylabel {"PL{"{
n");
        fprintf(gnu_file,"plot {"%s{"t using
1:5 notitle{n",PREMIA_OUT);
        fprintf(gnu_file,"set output {"fig1.
tex{"{n");
        fprintf(gnu_file,"set title {"Second
Stock's trajectory.{n");
        fprintf(gnu_file,"set ylabel {"Stock2
{n");
        fprintf(gnu_file,"plot {"%s{"t using
1:8 notitle, {"%s{"t using 13:14 title {"Spot
Target{" with points{n",PREMIA_OUT,PREMIA_OUT);
        /*Stockmax and PLmax*/
        fprintf(gnu_file,"set output {"fig3.
tex{"{n");
        fprintf(gnu_file,"set title {"First
Stock's trajectory to obtain the maximal PL.{n");
;
        fprintf(gnu_file,"set ylabel {"Stock1
{n");

```

```

        fprintf(gnu_file,"plot {%s{%t using
1:3 notitle , {%s{%t using 11:12 title {"Spot
Target{" with points,{
{%s{%t using 15:16 title {"Exercice Time{" wit
h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT);
        fprintf(gnu_file,"set output {"fig5.
tex{"{n");
        fprintf(gnu_file,"set title {"PL's
trajectory.{"{n");
        fprintf(gnu_file,"set ylabel {"PL{"{
n");
        fprintf(gnu_file,"plot {%s{%t using
1:6 notitle{n",PREMIA_OUT);
        fprintf(gnu_file,"set output {"fig4.
tex{"{n");
        fprintf(gnu_file,"set title {"Second
Stock's trajectory.{"{n");
        fprintf(gnu_file,"set ylabel {"Stock2
{"{n");
        fprintf(gnu_file,"plot {%s{%t using
1:9 notitle, {%s{%t using 13:14 title {"Spot
Target{" with points{n",PREMIA_OUT,PREMIA_OUT);
        /*Stockmin and PLmin*/
        fprintf(gnu_file,"set output {"fig6.
tex{"{n");
        fprintf(gnu_file,"set title {"First
Stock's trajectory to obtain the median PL.{"{n");
        fprintf(gnu_file,"set ylabel {"Stock1
{"{n");
        fprintf(gnu_file,"plot {%s{%t using
1:4 notitle , {%s{%t using 11:12 title {"Spot
Target{" with points,{
{%s{%t using 15:16 title {"Exercice Time{" wit
h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT);
        fprintf(gnu_file,"set output {"fig8.
tex{"{n");
        fprintf(gnu_file,"set title {"PL's
trajectory.{"{n");
        fprintf(gnu_file,"set ylabel {"PL{"{
n");
        fprintf(gnu_file,"plot {%s{%t using

```

```

1:7 notitle{\n",PREMIA_OUT);
    fprintf(gnu_file,"set output {"fig7.
tex{"{\n");
    fprintf(gnu_file,"set title {"Second
Stock's trajectory.{"{\n");
    fprintf(gnu_file,"set ylabel {"Stock2
{"{\n");
    fprintf(gnu_file,"plot {"%s{"{t using
1:10 notitle, {"%s{"{t using 13:14 title {"Spot
Target{" with points{\n",PREMIA_OUT,PREMIA_OUT);
    fprintf(gnu_file,"set size 1.,1.{n");
}
break;
                                case 5: /*TR_Patry*/
{

    fprintf(gnu_file,"set xlabel {"Time{"{\n
});

/*Stockmin and PLmin*/

    fprintf(gnu_file,"set output {"fig0.tex{
{"{\n");

    fprintf(gnu_file,"set title {"Stock's
trajectory to obtain the minimal PL.{"{\n");

    fprintf(gnu_file,"set ylabel {"Stock{"{
n");

    fprintf(gnu_file,"plot {"%s{"{t using 1:
2 notitle ,{
{"%s{"{t using 11:12 title {"Hedge
times{" with points{\n",PREMIA_OUT,PREMIA_OUT);

    fprintf(gnu_file,"set output {"fig1.tex{
{"{\n");

```

```

    fprintf(gnu_file,"set title {"PL's traj
ectory.{n"});

    fprintf(gnu_file,"set ylabel {"PL{n"});

    fprintf(gnu_file,"plot {"%s"{t using 1:
5 notitle{n",PREMIA_OUT});

    fprintf(gnu_file,"set output {"fig2.tex{
n"});

    fprintf(gnu_file,"set title {"Delta's
trajectory.{n"});

    fprintf(gnu_file,"set ylabel {"Delta{n"});

    fprintf(gnu_file,"plot {"%s"{t using 1:
8 notitle{n",PREMIA_OUT});

/*Stockmax and PLmax*/

    fprintf(gnu_file,"set output {"fig3.tex{
n"});

    fprintf(gnu_file,"set title {"Stock's
trajectory to obtain the maximal PL.{n"});

    fprintf(gnu_file,"set ylabel {"Stock{n"});

    fprintf(gnu_file,"plot {"%s"{t using 1:
3 notitle ,{
    {"%s"{t using 13:14 title {"Hedge
times{" with points{n",PREMIA_OUT,PREMIA_OUT});

```



```

    fprintf(gnu_file,"set output {"fig4.tex{
"{n"});

    fprintf(gnu_file,"set title {"PL's traj
ectory.{"{n"});

    fprintf(gnu_file,"set ylabel {"PL{"{n"});

    fprintf(gnu_file,"plot {"%s{"{t using 1:
6 notitle{n",PREMIA_OUT});

    fprintf(gnu_file,"set output {"fig5.tex{
"{n"});

    fprintf(gnu_file,"set title {"Delta's
trajectory.{"{n"});

    fprintf(gnu_file,"set ylabel {"Delta{"{
n"});

    fprintf(gnu_file,"plot {"%s{"{t using 1:
9 notitle{n",PREMIA_OUT});

    /*Stockmin and PLmin*/

    fprintf(gnu_file,"set output {"fig6.tex{
"{n"});

    fprintf(gnu_file,"set title {"Stock's
trajectory to obtain the mean PL.{"{n"});

    fprintf(gnu_file,"set ylabel {"Stock{"{
n"});

    fprintf(gnu_file,"plot {"%s{"{t using 1:
4 notitle ,{
{"%s{"{t using 15:16 title {"Hedge
times{" with points{n",PREMIA_OUT,PREMIA_OUT});

```

```

    fprintf(gnu_file,"set output {"fig7.tex{
"{n"});

    fprintf(gnu_file,"set title {"PL's traj
ectory.{"{n"});

    fprintf(gnu_file,"set ylabel {"PL{"{n"});

    fprintf(gnu_file,"plot {"%s{"{t using 1:
7 notitle{n",PREMIA_OUT});

    fprintf(gnu_file,"set output {"fig8.tex{
"{n"});

    fprintf(gnu_file,"set title {"Delta's
trajectory.{"{n"});

    fprintf(gnu_file,"set ylabel {"Delta{"{
n"});

    fprintf(gnu_file,"plot {"%s{"{t using 1:
10 notitle{n",PREMIA_OUT});

    }

    break;
default:
    break;
}
fclose(gnu_file);
/* TEX FILE */

if ((gnu_tex=fopen(GNU_TEX,"w"))==NULL)
{
    Fprintf(TOSCREEN,"Unable to create the G
nutex file{n");
    return;
}

```

```

/* Comments for LaTeX: needed for archival pur
   poses */
fprintf(gnu_tex,"%%Test{n");
MKSTRING(dummy,"mod/",model->ID,"/");
strcat(dummy,pricing->ID);
strcat(dummy,"/");
nom=dummy;
while(*nom) *nom++ = (char)tolower(*nom);
fprintf(gnu_tex,"%%%s{s{n","../Src/",dummy);
strcpy(dummy,method->Name);
nom=dummy;
while(*nom) *nom++ = (char)tolower(*nom);
fprintf(gnu_tex,"%%%s{s{n",dummy);

strcpy(dummy,pricing->ID);
strcat(dummy,"/");
nom=dummy;
while(*nom) *nom++ = (char)tolower(*nom);
fprintf(gnu_tex,"%%%s{s{n",dummy);

strcpy(dummy,test->Name);
/*strcpy(dummy,pricing->Test->Name);*/
nom=dummy;
while(*nom) *nom++ = (char)tolower(*nom);
fprintf(gnu_tex,"%%%s{s{n",dummy);
begin_tex_file(gnu_tex);
/* first page */
titles_tex=fopen(TITLES_TEX,"w");
fprintf(titles_tex,"{{begin{alltt}{n");

/* ----- Make links for the MODEL --
   ----- */
strcpy(doc_pdf_name,"../");
strcat(doc_pdf_name,model->ID);
strcat(doc_pdf_name,"_doc.pdf");
p=doc_pdf_name;
while(*p) *p++ = (char)tolower(*p);
fprintf(titles_tex,"{{textcolor{darkblue}{Mod
   el:}{{href{%s}{%s}{n",doc_pdf_name,model->Name);

/* ----- Make links for the OPTION

```

```

----- */

strcpy(doc_pdf_name,"../../../opt/");
strcat(doc_pdf_name,option->ID);
strcat(doc_pdf_name,"/");
strcat(doc_pdf_name,option->Name);
strcat(doc_pdf_name,"_doc.pdf");
p=doc_pdf_name;
while(*p) *p++ = (char)tolower(*p);
fprintf(titles_tex,"{{textcolor{darkblue}{Opti
on:}}{{href{%s}{%s}}{n",doc_pdf_name,option->Name);

/* ----- Make links for the PRICING
METHOD ----- */

strcpy(doc_pdf_name,method->Name);
strcat(doc_pdf_name,"_doc.pdf");
p=doc_pdf_name;
while(*p) *p++ = (char)tolower(*p);
fprintf(titles_tex,"{{textcolor{darkblue}{Pric
ing Method:}}{{href{%s}{%s}}{n",doc_pdf_name,method-
>Name);

/* ----- Make links for the Dynamic
Test ----- */
strcpy(doc_pdf_name,test->Name);
/*strcpy(doc_pdf_name,pricing->Test->Name);*/
strcat(doc_pdf_name,"_doc.pdf");
p=doc_pdf_name;
while(*p) *p++ = (char)tolower(*p);
fprintf(titles_tex,"{{textcolor{darkblue}{Dyna
mic Test:}}{{href{%s}{%s}}{n",doc_pdf_name,test->Na
me);

/* fprintf(titles_tex,"{{textcolor{darkblue}{
Dynamic Test:}}{{href{%s}{%s}}{n",doc_pdf_name,pric
ing->Test->Name);*/
/* ----- Write now the parameters ----
-----*/
if ((output=fopen(PREMIA_OUT,"r"))==NULL)

```

```

    {
        Fprintf(TOSCREEN,"Unable to open the da
ta file{n");
        return;
    }

fseek(output,0L,SEEK_SET);
do
{
    write=0;
    fgets(line,sizeof(line),output);
    line[strlen(line)-1]='{0';
    if (line[0]=='{0')
        fprintf(titles_tex,"{n");
    if (line[0]=='#' && line[1]!='#')
    {
        for (k=0; k<=(int)(strlen(line)-1)
;k++)
        {
            if (line[k]==':')
                write=1;
        }
        if (write==1)
            fprintf(titles_tex,"%s{n",line+
1);
    }
} while (!feof(output));

fclose(output);
fprintf(titles_tex,"{{end{alltt}}{n");
fclose(titles_tex);
fprintf(gnu_tex,"{{input{%s}}{n",TITLES_TEX);
fprintf(gnu_tex,"{{newpage{n");
if ((typegraph==0)|| (typegraph==1)|| (typegrap
h==2)) /* STD or LIM or DOUBLIM */
{
    /* second page*/
    fprintf(gnu_tex,"{{input{fig0.tex}}{n{{{
{n");
    fprintf(gnu_tex,"{{vspace{2.5cm}}{n");
    fprintf(gnu_tex,"{{input{fig1.tex}}{n");

```

```

        fprintf(gnu_tex, "{{newpage{n}}");
        /* third page*/
        fprintf(gnu_tex, "{{input{fig2.tex}{n}}{{{{
{n}}");
        fprintf(gnu_tex, "{{vspace{2.5cm}{n}}");
        fprintf(gnu_tex, "{{input{fig3.tex}{n}}");
        fprintf(gnu_tex, "{{newpage{n}}");
        /*fourth page*/
        fprintf(gnu_tex, "{{input{fig4.tex}{n}}{{{{
{n}}");
        fprintf(gnu_tex, "{{vspace{2.5cm}{n}}");
        fprintf(gnu_tex, "{{input{fig5.tex}{n}}");
    }
if (typegraph==5) /*for trpatry*/    {

    /* second page*/

    fprintf(gnu_tex, "{{input{fig0.tex}{n}}{{{{{{{{n}}
    );

    fprintf(gnu_tex, "{{vspace{2.5cm}{n}}");

    fprintf(gnu_tex, "{{input{fig1.tex}{n}}{{{{{{{{n}}
    );

    fprintf(gnu_tex, "{{vspace{2.5cm}{n}}");

    fprintf(gnu_tex, "{{input{fig2.tex}{n}}");

    fprintf(gnu_tex, "{{newpage{n}}");

    /* third page*/

    fprintf(gnu_tex, "{{input{fig3.tex}{n}}{{{{{{{{n}}
    );

    fprintf(gnu_tex, "{{vspace{2.5cm}{n}}");

    fprintf(gnu_tex, "{{input{fig4.tex}{n}}{{{{{{{{n}}
    );

```

```

fprintf(gnu_tex, "\\vspace{2.5cm}\\n");

fprintf(gnu_tex, "\\input{fig5.tex}\\n");

fprintf(gnu_tex, "\\newpage\\n");

/*fourth page*/

fprintf(gnu_tex, "\\input{fig6.tex}\\n\\{\\{\\{\\{\\n"
);

fprintf(gnu_tex, "\\vspace{2.5cm}\\n");

fprintf(gnu_tex, "\\input{fig7.tex}\\n\\{\\{\\{\\{\\n"
);

fprintf(gnu_tex, "\\vspace{2.5cm}\\n");

fprintf(gnu_tex, "\\input{fig8.tex}\\n");

}

if ((typegraph==1)|| (typegraph==2)) /* LIM or
DOUBLIM */
{
    /* fifth page*/
    fprintf(gnu_tex, "\\newpage\\n");
    fprintf(gnu_tex, "\\input{fig6.tex}\\n\\{\\{\\{\\{\\n");
    fprintf(gnu_tex, "\\vspace{2.5cm}\\n");
    fprintf(gnu_tex, "\\input{fig7.tex}\\n");
    fprintf(gnu_tex, "\\newpage\\n");
    /* sixth page*/
    fprintf(gnu_tex, "\\input{fig8.tex}\\n\\{\\{\\{\\{\\n");
    fprintf(gnu_tex, "\\vspace{2.5cm}\\n");
    fprintf(gnu_tex, "\\input{fig9.tex}\\n");
    fprintf(gnu_tex, "\\newpage\\n");
    /* seventh page*/
    fprintf(gnu_tex, "\\input{fig10.tex}\\n\\{\\{\\{\\{\\n");

```

```

    {\n");
    fprintf(gnu_tex, "{{vspace{2.5cm}}\n");
    fprintf(gnu_tex, "{{input{fig11.tex}}\n");
}
if ((typegraph==3)|| (typegraph==4)) /* PAD or
STD2D */
{
    /* second page*/
    fprintf(gnu_tex, "{{input{fig0.tex}}\n{{{
\n");
    fprintf(gnu_tex, "{{vspace{1.5cm}}\n");
    fprintf(gnu_tex, "{{input{fig1.tex}}\n{{{
\n");
    fprintf(gnu_tex, "{{vspace{1.5cm}}\n");
    fprintf(gnu_tex, "{{input{fig2.tex}}\n");
    fprintf(gnu_tex, "{{newpage\n");
    /* third page*/
    fprintf(gnu_tex, "{{input{fig3.tex}}\n{{{
\n");
    fprintf(gnu_tex, "{{vspace{1.5cm}}\n");
    fprintf(gnu_tex, "{{input{fig4.tex}}\n{{{
\n");
    fprintf(gnu_tex, "{{vspace{1.5cm}}\n");
    fprintf(gnu_tex, "{{input{fig5.tex}}\n");
    fprintf(gnu_tex, "{{newpage\n");
    /*fourth page*/
    fprintf(gnu_tex, "{{input{fig6.tex}}\n{{{
\n");
    fprintf(gnu_tex, "{{vspace{1.5cm}}\n");
    fprintf(gnu_tex, "{{input{fig7.tex}}\n{{{
\n");
    fprintf(gnu_tex, "{{vspace{1.5cm}}\n");
    fprintf(gnu_tex, "{{input{fig8.tex}}\n");
}
end_tex_file(gnu_tex);
fclose(gnu_tex);
/* GNU TO SCREEN */
if ((gnu_screen=fopen(GNUPLOT_SCREEN_PLT, "w"))
==NULL)
{
    Fprintf(TOSCREEN, "Unable to create the G

```



```

    nutoscreen file{n");
    return;
}

begin_gnu(gnu_screen);
fprintf(gnu_screen,"set terminal{n");
switch (typegraph)
{
case 0: /*STD*/
    {
        fprintf(gnu_screen,"set xlabel {"
Time{"{n");
        /*Stockmin and PLmin*/
        fprintf(gnu_screen,"set size 1.,1.{ns
et origin 0.,0.{nset multiplot{n");
        fprintf(gnu_screen,"set size 1.,.5{ns
et origin 0.,.5{n");
        fprintf(gnu_screen,"set title {"Stock
's trajectory to obtain the minimal PL.{n");
        fprintf(gnu_screen,"set ylabel {"Sto
ck{"{n");
        fprintf(gnu_screen,"plot {"%s{"{t usi
ng 1:2 notitle ,{
{"%s{"{t using 8:9 title {"Spot Target{" with po
ints,{
{"%s{"{t using 10:11 title {"Exercice Time{" wit
h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT);
        fprintf(gnu_screen,"set size 1.,.5{ns
et origin 0.,0.{n");
        fprintf(gnu_screen,"set title {"PL's
trajectory.{n");
        fprintf(gnu_screen,"set ylabel {"PL{"
{n");
        fprintf(gnu_screen,"plot {"%s{"{t usi
ng 1:5 notitle{n",PREMIA_OUT);
        fprintf(gnu_screen,"set nomultiplot{
n");
        fprintf(gnu_screen,"pause -1 {"NEXT :
PLmax{"{n");
        /*Stockmax and PLmax*/
        fprintf(gnu_screen,"set size 1.,1.{ns

```

```

et origin 0.,0.{nset multiplot{n");
    fprintf(gnu_screen,"set size 1.,.5{ns
et origin 0.,.5{n");
    fprintf(gnu_screen,"set title {"Stock
's trajectory to obtain the maximal PL.{n");
    fprintf(gnu_screen,"set ylabel {"Sto
ck{n");
    fprintf(gnu_screen,"plot {"%s{"{t usi
ng 1:3 notitle ,{
{"%s{"{t using 8:9 title {"Spot Target{" with po
ints,{
{"%s{"{t using 10:11 title {"Exercice Time{" wit
h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT);
    fprintf(gnu_screen,"set size 1.,.5{ns
et origin 0.,0.{n");
    fprintf(gnu_screen,"set title {"PL's
trajectory.{n");
    fprintf(gnu_screen,"set ylabel {"PL{"
{n");
    fprintf(gnu_screen,"plot {"%s{"{t usi
ng 1:6 notitle{n",PREMIA_OUT);
    fprintf(gnu_screen,"set nomultiplot{
n");

    fprintf(gnu_screen,"pause -1 {"NEXT :
PLmedian{n");
    /*Stockmedian and PLmedian*/
    fprintf(gnu_screen,"set size 1.,1.{ns
et origin 0.,0.{nset multiplot{n");
    fprintf(gnu_screen,"set size 1.,.5{ns
et origin 0.,.5{n");
    fprintf(gnu_screen,"set title {"Stock
's trajectory to obtain the median PL.{n");
    fprintf(gnu_screen,"set ylabel {"Sto
ck{n");
    fprintf(gnu_screen,"plot {"%s{"{t usi
ng 1:4 notitle ,{
{"%s{"{t using 8:9 title {"Spot Target{" with po
ints,{
{"%s{"{t using 10:11 title {"Exercice Time{" wit
h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT);

```

```

        fprintf(gnu_screen,"set size 1.,.5{ns
et origin 0.,0.{n");
        fprintf(gnu_screen,"set title {"PL's
trajectory.{n");
        fprintf(gnu_screen,"set ylabel {"PL{"
n");
        fprintf(gnu_screen,"plot {"%s{"{t usi
ng 1:7 notitle{n",PREMIA_OUT);
        fprintf(gnu_screen,"set nomultiplot{
n");

        fprintf(gnu_screen,"pause -1 {"NEXT :
EXIT{"{n");
    }
    break;
case 1: /*LIM*/
{
    fprintf(gnu_screen,"set xlabel {"
Time{"{n");
    /*Stockmin and PLmin*/
    fprintf(gnu_screen,"set size 1.,1.{ns
et origin 0.,0.{nset multiplot{n");
    fprintf(gnu_screen,"set size 1.,.5{ns
et origin 0.,.5{n");
    fprintf(gnu_screen,"set title {"Stock
's trajectory to obtain the minimal PL.{n");
    fprintf(gnu_screen,"set ylabel {"Sto
ck{"{n");
    fprintf(gnu_file,"plot {"%s{"{t using
    1:2 notitle ,{
{"%s{"{t using 1:14 title {"Limit{"{
{"%s{"{t using 15:16 title {"Spot Target{" with
points,{
{"%s{"{t using 17:18 title {"Exercice Time{" wit
h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT,PREM
IA_OUT);
    fprintf(gnu_screen,"set size 1.,.5{ns
et origin 0.,0.{n");
    fprintf(gnu_screen,"set title {"PL's
trajectory.{n");
    fprintf(gnu_screen,"set ylabel {"PL{"

```

```

{n});
    fprintf(gnu_screen,"plot {%s"{"t usi
ng 1:5 notitle{n",PREMIA_OUT);
    fprintf(gnu_screen,"set nomultiplot{
n");

    fprintf(gnu_screen,"pause -1 {"NEXT :
PLmax{"{n");
    /*Stockmax and PLmax*/
    fprintf(gnu_screen,"set size 1.,1.{ns
et origin 0.,0.{nset multiplot{n");
    fprintf(gnu_screen,"set size 1.,.5{ns
et origin 0.,.5{n");
    fprintf(gnu_screen,"set title {"Stock
's trajectory to obtain the maximal PL.{"{n");
    fprintf(gnu_screen,"set ylabel {"Sto
ck{"{n");
    fprintf(gnu_file,"plot {%s"{"t using
1:3 notitle ,{
{"%s"{"t using 1:14 title {"Limit{"{
{"%s"{"t using 15:16 title {"Spot Target{" with
points,{
{"%s"{"t using 17:18 title {"Exercice Time{" wit
h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT,PREM
IA_OUT);
    fprintf(gnu_screen,"set size 1.,.5{ns
et origin 0.,0.{n");
    fprintf(gnu_screen,"set title {"PL's
trajectory.{"{n");
    fprintf(gnu_screen,"set ylabel {"PL{"
{n");
    fprintf(gnu_screen,"plot {%s"{"t usi
ng 1:6 notitle{n",PREMIA_OUT);
    fprintf(gnu_screen,"set nomultiplot{
n");

    fprintf(gnu_screen,"pause -1 {"NEXT :
PLmedian{"{n");
    /*Stockmedian and PLmedian*/
    fprintf(gnu_screen,"set size 1.,1.{ns
et origin 0.,0.{nset multiplot{n");

```

```

        fprintf(gnu_screen,"set size 1.,.5{ns
et origin 0.,.5{n");
        fprintf(gnu_screen,"set title {"Stock
's trajectory to obtain the median PL.{n");
        fprintf(gnu_screen,"set ylabel {"Sto
ck{n");
        fprintf(gnu_file,"plot {"%s{"{t using
1:4 notitle ,{
{"%s{"{t using 1:14 title {"Limit{",{
{"%s{"{t using 15:16 title {"Spot Target{" with
points,{
{"%s{"{t using 17:18 title {"Exercice Time{" wit
h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT,PREM
IA_OUT);
        fprintf(gnu_screen,"set size 1.,.5{ns
et origin 0.,0.{n");
        fprintf(gnu_screen,"set title {"PL's
trajectory.{n");
        fprintf(gnu_screen,"set ylabel {"PL{"
{n");
        fprintf(gnu_screen,"plot {"%s{"{t usi
ng 1:7 notitle{n",PREMIA_OUT);
        fprintf(gnu_screen,"set nomultiplot{
n");

        fprintf(gnu_screen,"pause -1 {"NEXT :
PLminbreached{n");
        /*Stockminbreached and PLminbreached*
/
        fprintf(gnu_screen,"set size 1.,1.{ns
et origin 0.,0.{nset multiplot{n");
        fprintf(gnu_screen,"set size 1.,.5{ns
et origin 0.,.5{n");
        fprintf(gnu_screen,"set title {"Stock
's trajectory to obtain the minimal PL after ha
ving reached the limit.{n");
        fprintf(gnu_screen,"set ylabel {"Sto
ck{n");
        fprintf(gnu_file,"plot {"%s{"{t using
1:8 notitle ,{
{"%s{"{t using 1:14 title {"Limit{",{

```

```

{"%s{"{t using 15:16 title {"Spot Target{" with
  points,{
{"%s{"{t using 17:18 title {"Exercice Time{" wit
  h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT,PREM
  IA_OUT);
    fprintf(gnu_screen,"set size 1.,.5{ns
  et origin 0.,0.{n");
    fprintf(gnu_screen,"set title {"PL's
  trajectory.{n");
    fprintf(gnu_screen,"set ylabel {"PL{"
  {n");
    fprintf(gnu_screen,"plot {"%s{"{t usi
  ng 1:11 notitle{n",PREMIA_OUT);
    fprintf(gnu_screen,"set nomultiplot{
  n");

    fprintf(gnu_screen,"pause -1 {"NEXT :
  PLmaxbreached{"{n");
    /*Stockmaxbreached and PLmaxbreached*
  /
    fprintf(gnu_screen,"set size 1.,1.{ns
  et origin 0.,0.{nset multiplot{n");
    fprintf(gnu_screen,"set size 1.,.5{ns
  et origin 0.,.5{n");
    fprintf(gnu_screen,"set title {"Stock
  's trajectory to obtain the maximal PL after ha
  ving reached the limit.{n");
    fprintf(gnu_screen,"set ylabel {"Sto
  ck{"{n");
    fprintf(gnu_file,"plot {"%s{"{t using
    1:9 notitle ,{
{"%s{"{t using 1:14 title {"Limit{",{
{"%s{"{t using 15:16 title {"Spot Target{" with
  points,{
{"%s{"{t using 17:18 title {"Exercice Time{" wit
  h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT,PREM
  IA_OUT);
    fprintf(gnu_screen,"set size 1.,.5{ns
  et origin 0.,0.{n");
    fprintf(gnu_screen,"set title {"PL's
  trajectory.{n");

```

```

        fprintf(gnu_screen,"set ylabel {"PL{"
{n");
        fprintf(gnu_screen,"plot {"%s{"{t usi
ng 1:12 notitle{n",PREMIA_OUT);
        fprintf(gnu_screen,"set nomultiplot{
n");

        fprintf(gnu_screen,"pause -1 {"NEXT :
PLmedianbreached{"{n");
        /*Stockmedianbreached and PLmedianbre
ached*/
        fprintf(gnu_screen,"set size 1.,1.{ns
et origin 0.,0.{nset multiplot{n");
        fprintf(gnu_screen,"set size 1.,.5{ns
et origin 0.,.5{n");
        fprintf(gnu_screen,"set title {"Stock
's trajectory to obtain the median PL after havi
ng reached the limit."{n");
        fprintf(gnu_screen,"set ylabel {"Sto
ck{"{n");
        fprintf(gnu_file,"plot {"%s{"{t using
1:10 notitle ,{
{"%s{"{t using 1:14 title {"Limit","{
{"%s{"{t using 15:16 title {"Spot Target{" with
points,{
{"%s{"{t using 17:18 title {"Exercice Time{" wit
h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT,PREM
IA_OUT);
        fprintf(gnu_screen,"set size 1.,.5{ns
et origin 0.,0.{n");
        fprintf(gnu_screen,"set title {"PL's
trajectory."{n");
        fprintf(gnu_screen,"set ylabel {"PL{"
{n");
        fprintf(gnu_screen,"plot {"%s{"{t usi
ng 1:13 notitle{n",PREMIA_OUT);
        fprintf(gnu_screen,"set nomultiplot{
n");

        fprintf(gnu_screen,"pause -1 {"NEXT :
EXIT{"{n");

```

```

    }
    break;

case 2: /*DOUBLIM*/
    {
        fprintf(gnu_screen,"set xlabel {"
Time{"{n"});
        /*Stockmin and PLmin*/
        fprintf(gnu_screen,"set size 1.,1.{ns
et origin 0.,0.{nset multiplot{n"});
        fprintf(gnu_screen,"set size 1.,.5{ns
et origin 0.,.5{n"});
        fprintf(gnu_screen,"set title {"Stock
's trajectory to obtain the minimal PL.{"{n"});
        fprintf(gnu_screen,"set ylabel {"Sto
ck{"{n"});
        fprintf(gnu_screen,"plot {"%s{"{t usi
ng 1:2 notitle ,{
{"%s{"{t using 1:14 title {"LowerLimit{"{
{"%s{"{t using 1:15 title {"UpperLimit{"{
{"%s{"{t using 16:17 title {"Spot Target{" with
points,{
{"%s{"{t using 18:19 title {"Exercice Time{" wit
h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT,PREM
IA_OUT,PREMIA_OUT);
        fprintf(gnu_screen,"set size 1.,.5{ns
et origin 0.,0.{n"});
        fprintf(gnu_screen,"set title {"PL's
trajectory.{"{n"});
        fprintf(gnu_screen,"set ylabel {"PL{"
{n"});
        fprintf(gnu_screen,"plot {"%s{"{t usi
ng 1:5 notitle{n",PREMIA_OUT);
        fprintf(gnu_screen,"set nomultiplot{
n"});

        fprintf(gnu_screen,"pause -1 {"NEXT :
PLmax{"{n"});
        /*Stockmax and PLmax*/
        fprintf(gnu_screen,"set size 1.,1.{ns
et origin 0.,0.{nset multiplot{n"});

```



```

        fprintf(gnu_screen,"set size 1.,.5{ns
et origin 0.,.5{n");
        fprintf(gnu_screen,"set title {"Stock
's trajectory to obtain the maximal PL.{n");
        fprintf(gnu_screen,"set ylabel {"Sto
ck{n");
        fprintf(gnu_screen,"plot {"%s{"{t usi
ng 1:3 notitle ,{
{"%s{"{t using 1:14 title {"LowerLimit{",{
{"%s{"{t using 1:15 title {"UpperLimit{",{
{"%s{"{t using 16:17 title {"Spot Target{" with
points,{
{"%s{"{t using 18:19 title {"Exercice Time{" wit
h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT,PREM
IA_OUT,PREMIA_OUT);
        fprintf(gnu_screen,"set size 1.,.5{ns
et origin 0.,0.{n");
        fprintf(gnu_screen,"set title {"PL's
trajectory.{n");
        fprintf(gnu_screen,"set ylabel {"PL{"
{n");
        fprintf(gnu_screen,"plot {"%s{"{t usi
ng 1:6 notitle{n",PREMIA_OUT);
        fprintf(gnu_screen,"set nomultiplot{
n");

        fprintf(gnu_screen,"pause -1 {"NEXT :
PLmedian{n");
        /*Stockmedian and PLmedian*/
        fprintf(gnu_screen,"set size 1.,1.{ns
et origin 0.,0.{nset multiplot{n");
        fprintf(gnu_screen,"set size 1.,.5{ns
et origin 0.,.5{n");
        fprintf(gnu_screen,"set title {"Stock
's trajectory to obtain the median PL.{n");
        fprintf(gnu_screen,"set ylabel {"Sto
ck{n");
        fprintf(gnu_screen,"plot {"%s{"{t usi
ng 1:4 notitle ,{
{"%s{"{t using 1:14 title {"LowerLimit{",{
{"%s{"{t using 1:15 title {"UpperLimit{",{

```

```

{"%s{"{t using 16:17 title {"Spot Target{" with
  points,{
{"%s{"{t using 18:19 title {"Exercice Time{" wit
  h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT,PREM
  IA_OUT,PREMIA_OUT);
    fprintf(gnu_screen,"set size 1.,.5{ns
et origin 0.,0.{n");
    fprintf(gnu_screen,"set title {"PL's
trajectory.{n");
    fprintf(gnu_screen,"set ylabel {"PL{"
{n");
    fprintf(gnu_screen,"plot {"%s{"{t usi
ng 1:7 notitle{n",PREMIA_OUT);
    fprintf(gnu_screen,"set nomultiplot{
n");

    fprintf(gnu_screen,"pause -1 {"NEXT :
PLminbreached{"{n");
    /*Stockminbreached and PLminbreached*
/
    fprintf(gnu_screen,"set size 1.,1.{ns
et origin 0.,0.{nset multiplot{n");
    fprintf(gnu_screen,"set size 1.,.5{ns
et origin 0.,.5{n");
    fprintf(gnu_screen,"set title {"Stock
's trajectory to obtain the minimal PL after ha
ving reached the limit.{n");
    fprintf(gnu_screen,"set ylabel {"Sto
ck{"{n");
    fprintf(gnu_screen,"plot {"%s{"{t usi
ng 1:8 notitle ,{
{"%s{"{t using 1:14 title {"LowerLimit{",{
{"%s{"{t using 1:15 title {"UpperLimit{",{
{"%s{"{t using 16:17 title {"Spot Target{" with
  points,{
{"%s{"{t using 18:19 title {"Exercice Time{" wit
  h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT,PREM
  IA_OUT,PREMIA_OUT);
    fprintf(gnu_screen,"set size 1.,.5{ns
et origin 0.,0.{n");
    fprintf(gnu_screen,"set title {"PL's

```

```

trajectory.{"{n}";
    fprintf(gnu_screen,"set ylabel {"PL{"
{n}");
    fprintf(gnu_screen,"plot {"%s{"{t usi
ng 1:11 notitle{n",PREMIA_OUT);
    fprintf(gnu_screen,"set nomultiplot{
n}");

    fprintf(gnu_screen,"pause -1 {"NEXT :
PLmaxbreached{"{n}";
    /*Stockmaxbreached and PLmaxbreached*
/
    fprintf(gnu_screen,"set size 1.,1.{ns
et origin 0.,0.{nset multiplot{n}");
    fprintf(gnu_screen,"set size 1.,.5{ns
et origin 0.,.5{n}");
    fprintf(gnu_screen,"set title {"Stock
's trajectory to obtain the maximal PL after ha
ving reached the limit."{n}");
    fprintf(gnu_screen,"set ylabel {"Sto
ck{"{n}");
    fprintf(gnu_screen,"plot {"%s{"{t usi
ng 1:9 notitle ,{
{"%s{"{t using 1:14 title {"LowerLimit{",{
{"%s{"{t using 1:15 title {"UpperLimit{",{
{"%s{"{t using 16:17 title {"Spot Target{" with
points,{
{"%s{"{t using 18:19 title {"Exercice Time{" wit
h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT,PREM
IA_OUT,PREMIA_OUT);
    fprintf(gnu_screen,"set size 1.,.5{ns
et origin 0.,0.{n}");
    fprintf(gnu_screen,"set title {"PL's
trajectory."{n}");
    fprintf(gnu_screen,"set ylabel {"PL{"
{n}");
    fprintf(gnu_screen,"plot {"%s{"{t usi
ng 1:12 notitle{n",PREMIA_OUT);
    fprintf(gnu_screen,"set nomultiplot{
n}");

```

```

        fprintf(gnu_screen,"pause -1 {"NEXT :
        PLmedianbreached{"{n}");
        /*Stockmedianbreached and PLmedianbreached*/
        fprintf(gnu_screen,"set size 1.,1.{ns
        et origin 0.,0.{nset multiplot{n}");
        fprintf(gnu_screen,"set size 1.,.5{ns
        et origin 0.,.5{n}");
        fprintf(gnu_screen,"set title {"Stock
        's trajectory to obtain the median PL after havi
        ng reached the limit.{n}");
        fprintf(gnu_screen,"set ylabel {"Sto
        ck{n}");
        fprintf(gnu_screen,"plot {"%s{"{t usi
        ng 1:10 notitle ,{
        {"%s{"{t using 1:14 title {"LowerLimit{",{
        {"%s{"{t using 1:15 title {"UpperLimit{",{
        {"%s{"{t using 16:17 title {"Spot Target{" with
        points,{
        {"%s{"{t using 18:19 title {"Exercice Time{" wit
        h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT,PREM
        IA_OUT,PREMIA_OUT);
        fprintf(gnu_screen,"set size 1.,.5{ns
        et origin 0.,0.{n}");
        fprintf(gnu_screen,"set title {"PL's
        trajectory.{n}");
        fprintf(gnu_screen,"set ylabel {"PL{"
        {n}");
        fprintf(gnu_screen,"plot {"%s{"{t usi
        ng 1:13 notitle{n",PREMIA_OUT);
        fprintf(gnu_screen,"set nomultiplot{
        n}");

        fprintf(gnu_screen,"pause -1 {"NEXT :
        EXIT{"{n}");
        }
        break;

case 3: /*PAD*/
{
        fprintf(gnu_screen,"set xlabel{n}");

```

```

/*Stockmin and PLmin*/
    fprintf(gnu_screen,"set size 1.,1.{ns
et origin 0.,0.{nset multiplot{n");
    fprintf(gnu_screen,"set size 1.,.4{ns
et origin 0.,.6{n");
    fprintf(gnu_screen,"set title {"Stock
's trajectory to obtain the minimal PL.{n");
    fprintf(gnu_screen,"set ylabel {"Sto
ck{n");
    fprintf(gnu_screen,"plot {"%s{"{t usi
ng 1:2 notitle ,{
{"%s{"{t using 11:12 title {"Spot Target{" with
points,{
{"%s{"{t using 13:14 title {"Exercice Time{" wit
h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT);
    fprintf(gnu_screen,"set size 1.,.4{ns
et origin 0.,.3{n");
    fprintf(gnu_screen,"set title {"PathD
ep's trajectory.{n");
    fprintf(gnu_screen,"set ylabel {"Pat
hDep{n");
    fprintf(gnu_screen,"plot {"%s{"{t usi
ng 1:8 notitle{n",PREMIA_OUT);

    fprintf(gnu_screen,"set xlabel {"
Time{" ,+2{n");
    fprintf(gnu_screen,"set size 1.,.4{ns
et origin 0.,0.{n");
    fprintf(gnu_screen,"set title {"PL's
trajectory.{n");
    fprintf(gnu_screen,"set ylabel {"PL{"
{n");
    fprintf(gnu_screen,"plot {"%s{"{t usi
ng 1:5 notitle{n",PREMIA_OUT);
    fprintf(gnu_screen,"set nomultiplot{
n");
    fprintf(gnu_screen,"set xlabel{n");
    fprintf(gnu_screen,"pause -1 {"NEXT :
PLmax{n");
/*Stockmax and PLmax*/
    fprintf(gnu_screen,"set size 1.,1.{ns

```

```

et origin 0.,0.{nset multiplot{n");
    fprintf(gnu_screen,"set size 1.,.4{ns
et origin 0.,.6{n");
    fprintf(gnu_screen,"set title {"Stock
's trajectory to obtain the maximal PL.{n");
    fprintf(gnu_screen,"set ylabel {"Sto
ck{n");
    fprintf(gnu_screen,"plot {"%s{"{t usi
ng 1:3 notitle ,{
{"%s{"{t using 11:12 title {"Spot Target{" with
points,{
{"%s{"{t using 13:14 title {"Exercice Time{" wit
h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT);
    fprintf(gnu_screen,"set size 1.,.4{ns
et origin 0.,.3{n");
    fprintf(gnu_screen,"set title {"PathD
ep's trajectory.{n");
    fprintf(gnu_screen,"set ylabel {"Pat
hDep{n");
    fprintf(gnu_screen,"plot {"%s{"{t usi
ng 1:9 notitle{n",PREMIA_OUT);

    fprintf(gnu_screen,"set xlabel {"
Time{" ,+2{n");
    fprintf(gnu_screen,"set size 1.,.4{ns
et origin 0.,0.{n");
    fprintf(gnu_screen,"set title {"PL's
trajectory.{n");
    fprintf(gnu_screen,"set ylabel {"PL{"
{n");
    fprintf(gnu_screen,"plot {"%s{"{t usi
ng 1:6 notitle{n",PREMIA_OUT);
    fprintf(gnu_screen,"set nomultiplot{
n");
    fprintf(gnu_screen,"set xlabel{n");

    fprintf(gnu_screen,"pause -1 {"NEXT :
PLmedian{n");
    /*Stockmedian and PLmedian*/
    fprintf(gnu_screen,"set size 1.,1.{ns
et origin 0.,0.{nset multiplot{n");

```

```

        fprintf(gnu_screen,"set size 1.,.4{ns
et origin 0.,.6{n");
        fprintf(gnu_screen,"set title {"Stock
's trajectory to obtain the median PL.{n");
        fprintf(gnu_screen,"set ylabel {"Sto
ck{n");
        fprintf(gnu_screen,"plot {"%s{"{t usi
ng 1:4 notitle ,{
{"%s{"{t using 11:12 title {"Spot Target{" with
points,{
{"%s{"{t using 13:14 title {"Exercice Time{" wit
h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT);
        fprintf(gnu_screen,"set size 1.,.4{ns
et origin 0.,.3{n");
        fprintf(gnu_screen,"set title {"PathD
ep's trajectory.{n");
        fprintf(gnu_screen,"set ylabel {"Pat
hDep{n");
        fprintf(gnu_screen,"plot {"%s{"{t usi
ng 1:10 notitle{n",PREMIA_OUT);

        fprintf(gnu_screen,"set xlabel {"
Time{" ,+2{n");
        fprintf(gnu_screen,"set size 1.,.4{ns
et origin 0.,0.{n");
        fprintf(gnu_screen,"set title {"PL's
trajectory.{n");
        fprintf(gnu_screen,"set ylabel {"PL{"
{n");
        fprintf(gnu_screen,"plot {"%s{"{t usi
ng 1:7 notitle{n",PREMIA_OUT);
        fprintf(gnu_screen,"set nomultiplot{
n");
        fprintf(gnu_screen,"set xlabel{n");

        fprintf(gnu_screen,"pause -1 {"NEXT :
EXIT{n");
    }
    break;
case 4: /*STD2D*/
    {

```

```

        fprintf(gnu_screen,"set xlabel{n");
        /*Stockmin and PLmin*/
        fprintf(gnu_screen,"set size 1.,1.{ns
et origin 0.,0.{nset multiplot{n");
        fprintf(gnu_screen,"set size 1.,.4{ns
et origin 0.,.6{n");
        fprintf(gnu_screen,"set title {"Firs
t Stock's trajectory to obtain the minimal PL.{
",-1{n");
        fprintf(gnu_screen,"set ylabel {"Sto
ck1{"{n");
        fprintf(gnu_screen,"plot {"%s{"{t usi
ng 1:2 notitle ,{
{"%s{"{t using 11:12 title {"Spot Target{" with
points,{
{"%s{"{t using 15:16 title {"Exercice Time{" wit
h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT);
        fprintf(gnu_screen,"set size 1.,.4{ns
et origin 0.,.3{n");
        fprintf(gnu_screen,"set title {"Secon
d Stock's trajectory.{
",-1{n");
        fprintf(gnu_screen,"set ylabel {"Sto
ck2{"{n");
        fprintf(gnu_screen,"plot {"%s{"{t usi
ng 1:8 notitle , {"%s{"{t using 13:14 title {"Sp
ot Target{" with points{n",PREMIA_OUT,PREMIA_OUT)
;

        fprintf(gnu_screen,"set xlabel {"
Time{" ,+2{n");
        fprintf(gnu_screen,"set size 1.,.4{ns
et origin 0.,0.{n");
        fprintf(gnu_screen,"set title {"PL's
trajectory.{
",-1{n");
        fprintf(gnu_screen,"set ylabel {"PL{"
{n");
        fprintf(gnu_screen,"plot {"%s{"{t usi
ng 1:5 notitle{n",PREMIA_OUT);
        fprintf(gnu_screen,"set nomultiplot{
n");
        fprintf(gnu_screen,"set xlabel{n");

```



```

        fprintf(gnu_screen,"pause -1 {"NEXT :
PLmax{"{n"});
        /*Stockmax and PLmax*/
        fprintf(gnu_screen,"set size 1.,1.{ns
et origin 0.,0.{nset multiplot{n"});
        fprintf(gnu_screen,"set size 1.,.4{ns
et origin 0.,.6{n"});
        fprintf(gnu_screen,"set title {"Firs
t Stock's trajectory to obtain the minimal PL.{"{
n"});
        fprintf(gnu_screen,"set ylabel {"Sto
ck1{"{n"});
        fprintf(gnu_screen,"plot {"%s{"{t usi
ng 1:3 notitle ,{
{"%s{"{t using 11:12 title {"Spot Target{" with
points,{
{"%s{"{t using 15:16 title {"Exercice Time{" wit
h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT});
        fprintf(gnu_screen,"set size 1.,.4{ns
et origin 0.,.3{n"});
        fprintf(gnu_screen,"set title {"Secon
d Stock's trajectory.{"{n"});
        fprintf(gnu_screen,"set ylabel {"Sto
ck2{"{n"});
        fprintf(gnu_screen,"plot {"%s{"{t usi
ng 1:9 notitle , {"%s{"{t using 13:14 title {"Sp
ot Target{" with points{n",PREMIA_OUT,PREMIA_OUT)
;

        fprintf(gnu_screen,"set xlabel {"
Time{" ,+2{n"});
        fprintf(gnu_screen,"set size 1.,.4{ns
et origin 0.,0.{n"});
        fprintf(gnu_screen,"set title {"PL's
trajectory.{"{n"});
        fprintf(gnu_screen,"set ylabel {"PL{"
{n"});
        fprintf(gnu_screen,"plot {"%s{"{t usi
ng 1:6 notitle{n",PREMIA_OUT});
        fprintf(gnu_screen,"set nomultiplot{
n"});

```

```

    fprintf(gnu_screen,"set xlabel{n");

    fprintf(gnu_screen,"pause -1 {"NEXT :
    PLmedian{"{n");
    /*Stockmedian and PLmedian*/
    fprintf(gnu_screen,"set size 1.,1.{ns
    et origin 0.,0.{nset multiplot{n");
    fprintf(gnu_screen,"set size 1.,.4{ns
    et origin 0.,.6{n");
    fprintf(gnu_screen,"set title {"Firs
    t Stock's trajectory to obtain the minimal PL.{"{
    n");
    fprintf(gnu_screen,"set ylabel {"Sto
    ck1{"{n");
    fprintf(gnu_screen,"plot {"%s{"{t usi
    ng 1:4 notitle ,{
    {"%s{"{t using 11:12 title {"Spot Target{" with
    points,{
    {"%s{"{t using 15:16 title {"Exercice Time{" wit
    h points{n",PREMIA_OUT,PREMIA_OUT,PREMIA_OUT);
    fprintf(gnu_screen,"set size 1.,.4{ns
    et origin 0.,.3{n");
    fprintf(gnu_screen,"set title {"Secon
    d Stock's trajectory.{"{n");
    fprintf(gnu_screen,"set ylabel {"Sto
    ck2{"{n");
    fprintf(gnu_screen,"plot {"%s{"{t usi
    ng 1:10 notitle , {"%s{"{t using 13:14 title {"Sp
    ot Target{" with points{n",PREMIA_OUT,PREMIA_OUT)
    ;

    fprintf(gnu_screen,"set xlabel {"
    Time{" ,+2{n");
    fprintf(gnu_screen,"set size 1.,.4{ns
    et origin 0.,0.{n");
    fprintf(gnu_screen,"set title {"PL's
    trajectory.{"{n");
    fprintf(gnu_screen,"set ylabel {"PL{"
    {n");
    fprintf(gnu_screen,"plot {"%s{"{t usi
    ng 1:7 notitle{n",PREMIA_OUT);

```

```

        fprintf(gnu_screen,"set nomultiplot{
n");
        fprintf(gnu_screen,"set xlabel{n");

        fprintf(gnu_screen,"pause -1 {"NEXT :
EXIT{"{n");
    }
    break;
case 5: /*TR_Patry*/

    {

        fprintf(gnu_screen,"set xlabel {"Time{"{
n");


        /*Stockmin and PLmin*/

        fprintf(gnu_screen,"set size 1.,1.{nset
origin 0.,0.{nset multiplot{n");

        fprintf(gnu_screen,"set size 1.,.5{nset
origin 0.,.5{n");

        fprintf(gnu_screen,"set title {"Stock's
trajectory to obtain the minimal PL.{"{n");

        fprintf(gnu_screen,"set ylabel {"Stock{"
{n");

        fprintf(gnu_screen,"plot {"%s{"{t using
1:2 notitle ,{
{"%s{"{t using 8:9 title {"Hedge
times{" with points{n",PREMIA_OUT,PREMIA_OUT);

        fprintf(gnu_screen,"set size 1.,.5{nset
origin 0.,0.{n");

```

```

    fprintf(gnu_screen,"set title {"PL's
trajectory.{"{n}");

    fprintf(gnu_screen,"set ylabel {"PL{"{n"
});

    fprintf(gnu_screen,"plot {"%s{"{t using
1:5 notitle{n",PREMIA_OUT);

    fprintf(gnu_screen,"set nomultiplot{n");

    fprintf(gnu_screen,"pause -1 {"NEXT : PL
max{"{n}");

    /*Stockmax and PLmax*/

    fprintf(gnu_screen,"set size 1.,1.{nset
origin 0.,0.{nset multiplot{n");

    fprintf(gnu_screen,"set size 1.,.5{nset
origin 0.,.5{n");

    fprintf(gnu_screen,"set title {"Stock's
trajectory to obtain the maximal PL.{"{n});

    fprintf(gnu_screen,"set ylabel {"Stock{"
{n});

    fprintf(gnu_screen,"plot {"%s{"{t using
1:3 notitle ,{
{"%s{"{t using 10:11 title {"Hedge
times{" with points{n",PREMIA_OUT,PREMIA_OUT);

    fprintf(gnu_screen,"set size 1.,.5{nset

```

```

origin 0.,0.{n});

    fprintf(gnu_screen,"set title {"PL's
trajectory.{n");

    fprintf(gnu_screen,"set ylabel {"PL{"{n"
);

    fprintf(gnu_screen,"plot {"%s{"{t using
1:6 notitle{n",PREMIA_OUT);

    fprintf(gnu_screen,"set nomultiplot{n");

    fprintf(gnu_screen,"pause -1 {"NEXT : PL
mean{"{n");

/*Stockmean and PLmean*/

    fprintf(gnu_screen,"set size 1.,1.{nset
origin 0.,0.{nset multiplot{n");

    fprintf(gnu_screen,"set size 1.,.5{nset
origin 0.,.5{n");

    fprintf(gnu_screen,"set title {"Stock's
trajectory to obtain the mean PL.{n");

    fprintf(gnu_screen,"set ylabel {"Stock{"
{n");

    fprintf(gnu_screen,"plot {"%s{"{t using
1:4 notitle ,{
    {"%s{"{t using 12:13 title {"Hedge
times{" with points{n",PREMIA_OUT,PREMIA_OUT);

```

```

    fprintf(gnu_screen,"set size 1.,.5{\\nset
origin 0.,0.{\\n");

    fprintf(gnu_screen,"set title {\\nPL's
trajectory.{\\n");

    fprintf(gnu_screen,"set ylabel {\\nPL{\\n"
);

    fprintf(gnu_screen,"plot {\\n%s{\\nt using
1:7 notitle{\\n",PREMIA_OUT);

    fprintf(gnu_screen,"set nomultiplot{\\n");

    fprintf(gnu_screen,"pause -1 {\\nNEXT :
EXIT{\\n");

}

    break;
default:
    break;
}
fclose(gnu_screen);
return;
}

```

## References