Premia Interface
version 17

C. Martini, A.Zanette

March 11, 2015

Contents

1 Input-Output functionalities

All the functions which deal with the interface of Premiashould be in the file tools.c in Common . These functions, given the architecture of the kernel of Premia, allow to select a model, then an option, lastly the pricing method, from the arrays defined as `extern` variables in the main file premia.c (in Src).
The output of Premiaconsist in a Gnuplot command and data file, also in a bunch of LaTeX files which are intended to be processed by a companion bash script (either out2pdf.bashor makearchive.bash) in order to get a pdf file (automatically linked to the documentation web of pdf files for

makearchive.bash, out2pdf.bashis intended as a previewer of the result pdf files).

To view the Gnuplot graph you need...Gnuplot, to make use of the archiving facility you need LaTeX and a bash shell (see the file readme.txt).

2 The utilities of tools.c

2.1 Iterate(..)

```
int Iterate(Planning *pt_plan,Iterator* pt_iterator,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_iterator->Min;
      if (next->Min.Vtype==END)
        {
          (void)ShowPlanning(VALUEONLYTOFILE,pt_plan);
          (void)Action(action,pt_model,pt_option,pt_pricing,
            pt_method,pt_test,VALUEONLYTOFILE,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,user,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;
}
```

2.2 Action(..)

```
void Action(char action,Model*pt_model,Option*pt_option,Pricing
    *pt_pricing,PricingMethod* pt_method,
    DynamicTest*pt_test,int user,Planning *pt_plan,TimeInfo *pt_time_info)
 /*************************************************************************
 .Performs the calling of a PricingMethod or aDynamicTest depending on
  action ('p' or 't'),
  AFTER going though all the checks.
 .If user is NAMEONLYTOFILE, then it justs prints the names of the output
  variables (one per line) TOFILE.
 .Else: if pt_plan->VarNumber>0, ie in the iteration case, writes the
  output variable values TOFILE separated with a blank,
  otherwise (no iteration case), TOSCREEN.

 .No Return value.
 *************************************************************************/
{
  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_option)==OK)
    {
      if ((pt_pricing->CheckMixing)(pt_option,pt_model)==OK)
        {
          if ((pt_method->Check)(user,pt_plan,pt_method)==OK)
        {
```

```
if ((pt_time_info->Check)(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_model->TypeModel,pt_method);
      }
    if ((user==NAMEONLYTOFILE)||(pt_plan->VarNumber>0))
      {
        ShowResultTimeInfo(user,pt_plan,error,pt_time_info);
        ShowResultMethod(user,pt_plan,error,pt_method);
      }
    else
      {
        ShowResultTimeInfo(TOSCREEN,pt_plan,error,pt_time_info);
        ShowResultTimeInfo(VALUEONLYTOFILE,pt_plan,error,pt_time_info)
        ShowResultMethod(TOSCREEN,pt_plan,error,pt_method);
        ShowResultMethod(VALUEONLYTOFILE,pt_plan,error,pt_method);
        Fprintf(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);

                  ShowResultTest(user,pt_plan,error,pt_test);
                }

              break;

            default :
              break;
            }

        }

      }
    }
  }
}
return;
}
```

2.3 MoreAction(..)

```
int MoreAction(int *count)
{
  char msg='e';
  if (*count==(MAX_METHODS-1))
      {
        Fprintf(TOSCREEN,"\n Max Number of Methds 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;
  }
}
```

2.4 SelectModel(..)

```
int SelectModel(int user,Planning *pt_plan,Model **listmodel,Model
**mod)
/*************************************************************************
.Ask the user to select the model (**mod) in the list *listmodel,
AND performs the call to the corresponding GetModel function.
.Displays the field ID of the models in fact, which is set to
"TYPEMODEL"
 through the MAKEMODEL macro.

.Return Value: .That of GetModel: in fact the last lines of the
function are useless.
*************************************************************************/
{
  int choice=0, i=0;
  char fhelp_name[MAX_PATH_LEN]="";
  char msg,answer;

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

  strcpy(fhelp_name,"..");
  strcat(fhelp_name,path_sep);
  strcat(fhelp_name,"Src");
```

```
  strcat(fhelp_name,path_sep);
  strcat(fhelp_name,"mod_doc.pdf");

  Fprintf(TOSCREEN,"\n_____________________MODEL 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 for 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",fhelp_name,NULL);
#endif
#ifndef _WIN32
      _spawnlp(1,"acroread","_spawnlp",fhelp_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;
}
```

2.5 SelectOption(..)

```
int SelectOption(int user,Planning *pt_plan,Family
**L_listopt,Model* pt_model,Pricing **pricing,Option **opt)
/*************************************************************************
.Ask the user to select the option (**opt) once *pt_model is
chosen: .**pricing should be the array of &Pricing available in
PREMIA. .**L_listopt should be the array of option families
available in PREMIA. Then the function tests trough
MatchingPricing if there is a *pricing corresponding to the
selected model and the current
 option family and if it is the case runs across the options of the family
 and displays the field name of the options.
AND performs the call to the corresponding GetOption function, or
goes to
 the next family if the choice 0 is made.

.Return Value: .That of GetOption or WRONG in case no option has
been selected.
*************************************************************************/
{
  int i,j,choice,k;
  Family* list;
  char family_name[MAX_CHAR_X3]="",dummy[MAX_CHAR_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(path_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(dummy[k]);

    if ((strlen(family_name)+strlen(dummy)>=MAX_CHAR_X3))
      {
    Fprintf(TOSCREEN,"%s\n",error_msg[PATH_TOO_LONG]);
    exit(WRONG);
      }

    strcat(family_name,dummy);
    /*printf("%s\n",family_name);*/
    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(stdin);
          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_WAIT,"AcroRd32.exe",
               "_spawnlp",family_name,NULL);
#endif
#ifndef _WIN32
              _spawnlp(1,"acroread","_spawnlp",
               family_name,NULL);
#endif
            }
            Fprintf(TOSCREEN,"\nNew value?:\t");
          }
          else {
            choice = atoi(answer);
            if ((choice<0)||(choice>j))
              {
            Fprintf(TOSCREEN,"\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)(user,pt_plan,*opt);;
          }
        Fprintf(TOSCREEN,"\n");
      }

    i=i+1;/*choice=0*/
      }
    Fprintf(TOSCREEN,"No more families!\n");

  } while(1);

  return WRONG;
}
```

2.6 MatchingPricing(..)

```
int MatchingPricing(int user,Model *pt_model,Option
*pt_option,Pricing **pricing)
/*****************************************************************************
.Checks wether there is a *pricing corresponding to Model and
Option* by concatenating the ID of model and option* and comparing
to that of the
 elements of **pricing.
.**pricing should be the array of &Pricing available in PREMIA.

.Return Value: .OK if success
 WRONG otherwise.
*****************************************************************************/
{
  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) && (pricing[i+1]!=NULL));

  if (strcmp(dummy,pricing[i]->ID)==0)
    {
      return OK;
    }
  return WRONG;
}
```

2.7 SelectPricing(..)

```
int SelectPricing(int user,Model *pt_model,Option
*pt_option,Pricing **pricing,Pricing **result)
/*****************************************************************************
.Selects the pricing (**result) corresponding to Option and
Model,THEREFORE
 ASSUMING THERE IS UNICITY, by concatenating the ID of model and option*
 and comparing to that of the elements of **pricing,
 AND calls the CHK_MIXING function.
.**pricing should be the array of &Pricing available in PREMIA.

.Return Value: .That of CHK_MIXING
 WRONG if there is no pricing available for this couple Option, Model.
*****************************************************************************/
{
  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) && (pricing[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;
}
```

2.8 SelectMethod(..)

```
int SelectMethod(int user,Planning *pt_plan,Pricing *pt_pricing,
    Option *opt,Model *mod,PricingMethod **met) {
/*****************************************************************************
.Asks the user to select he PricingMethod (**met) from the list in
the
 field Methods of  pt_pricing whose field CHK_OPT does not reject *opt,
 AND calls the GetMethod function.

.Return Value: .That of GetMethod WRONG if there is no method
available or if a stupid choice is made.
*****************************************************************************/
  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 != EOF)){
      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>isub));
    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;
}
```

2.9 BuildGnuStuff() and BuildGnuStuffTest()

```
void BuildGnuStuff()
/*****************************************************************************
.Builds from the file PREMIA_OUT a Gnuplot command file
(GNUPLOT_PLT)
 and datafile (GNUPLOT_DAT) (version 3.5 or greater),
 with setting the required title, xaxis, yaxis,...
 also prepares for 2D or 3d graph automatically.
 BUT ASSUMES IT IS EITHER A 2D OR 3D shot, ie this corresponds to
MAX_ITER=3 in optype.h. .GNUPLOT_PLT/DAT are set in config.h

.No Return Value.
*****************************************************************************/
{ /*We send the reader to the files tools.c, also for
BuildGnuStuffTest()}  */ }
```

3 The file premia.c

```
#include "optype.h"
```

```
#include "var.h"

#include "method.h"
#include "test.h"
#include "timeinfo.h"

#include "tools.h"
#include "random.h"

extern Model BS1D_model;
extern Model BS2D_model;
Model *models[]= {
  &BS1D_model,
  &BS2D_model,
  NULL
};

extern Family STD_family;
extern Family LIM_family;
extern Family LIMDISC_family;
extern Family DOUBLIM_family;
extern Family PAD_family;
extern Family STD2D_family;
Family *families[]= {
  &STD_family,
  &LIM_family,
  &LIMDISC_family,
  &DOUBLIM_family,
  &PAD_family,
  &STD2D_family,
  NULL
};

extern  Pricing BS1D_STD_pricing;
extern  Pricing BS1D_LIM_pricing;
extern  Pricing BS1D_LIMDISC_pricing;
extern  Pricing BS1D_PAD_pricing;
extern  Pricing BS1D_DOUBLIM_pricing;
extern  Pricing BS2D_STD2D_pricing;
Pricing *pricings[]= {
  &BS1D_STD_pricing,
```

```
  &BS1D_LIM_pricing,
  &BS1D_LIMDISC_pricing,
  &BS1D_PAD_pricing,
  &BS1D_DOUBLIM_pricing,
  &BS2D_STD2D_pricing,
  NULL
};

extern TimeInfo computation_time_info;

FILE *out_stream;
char premiasrcdir[256];
char premiapersodir[256];
#ifdef _WIN32 const char *path_sep = "\\";
#endif
#ifndef _WIN32 const char *path_sep = "/";
#endif

char PREMIA_OUT[256]="";
char GNUPLOT_DAT[256]="";
char TITLES_TEX[256]="";
char GNUPLOT_SCREEN_PLT[256]="";
char GNUPLOT_FILE_PLT[256]="";
char GNU_TEX[256]="";
char PREMIA_LOG[256]="";
char SESSION_LOG[256]="";

int main() {
  Planning          plan;
  Model*            pt_model;
  Option*           pt_option;
  Pricing*          pt_pricing;
  PricingMethod*    pt_method;
  PricingMethod*    pt_methods_available[MAX_METHODS];
  char             *tmp,*tmp1;
  int user;

  /*    char action;*/
     tmp=getenv("PREMIADIR");
     if ((strlen(tmp)+strlen(path_sep)+strlen("Src"))>=256)
        return WRONG;
```

```
      strcpy(premiasrcdir,tmp);
      strcat(premiasrcdir,path_sep);
      strcat(premiasrcdir,"Src");

      tmp1 = getenv("PREMIA_HOME_DIR");
      strcpy(premiapersodir,tmp1);

      sprintf(PREMIA_OUT,"premia.out");
      sprintf(GNUPLOT_DAT,"%s%sgnupremia.dat",
         premiapersodir,path_sep);
      sprintf(TITLES_TEX,"titles.tex");
      sprintf(GNUPLOT_SCREEN_PLT,"%s%sgnutoscreen.plt",
         premiapersodir,path_sep);
      sprintf(GNUPLOT_FILE_PLT,"%s%sgnutofile.plt",
         premiapersodir,path_sep);
      sprintf(GNU_TEX,"%s%sgnupremia.tex",
         premiapersodir,path_sep);
      sprintf(PREMIA_LOG,"%s%spremia.log",
         premiapersodir,path_sep);
      sprintf(SESSION_LOG,"%s%ssession.log",
         premiapersodir,path_sep);

(void)InputMode(&user);
(void)WellcomeMsg(user);
if ((InitErrorMsg()==OK)&&(InitVar()==OK)&&(InitMc()==OK))
{
do
{
(void)ResetPlanning(&plan);
plan.Action=ChooseAction();
if (OutputFile(&out_stream)==OK)
 {
 if (SelectModel(user,&plan,models,&pt_model)==OK)
  {
  if (SelectOption(user,&plan,families,pt_model,
    pricings,&pt_option)==OK)
   {
   if (SelectPricing(user,pt_model,pt_option,
    pricings,&pt_pricing)==OK)
    {
    while(1)
```

```
     {
     if (SelectMethod(user,&plan,pt_pricing,
        pt_option,pt_model,&pt_method)==OK)
        {
        if (GetTimeInfo(user,&plan,
         &computation_time_info)==OK)
         {
        /*action=ChooseAction();*/
        if ((plan.Action=='p')||(( plan.Action=='t')
         &&(GetTest(user,&plan,pt_pricing,pt_option,
          pt_pricing->Test)==OK))   )
         {
        (void)ShowPlanning(NAMEONLYTOFILE,&plan);
        (void)Action(plan.Action,pt_model,pt_option,
         pt_pricing,pt_method,pt_pricing->Test,
         NAMEONLYTOFILE,&plan,&computation_time_info);

        Fprintf(TOSCREEN,"\nComputing...\n");
        Iterate(&plan,&(plan.Par[0]),0,plan.Action,pt_model,
         pt_option,pt_pricing,pt_method,pt_pricing->Test,
         TOFILE,&computation_time_info);
        pt_methods_available[plan.NumberOfMethods]=pt_method;
        if (plan.Action=='t' || MoreAction(&(plan.NumberOfMethods))==WRONG)break
        }
    }
}
else break;
}
}
}
}
}

fclose(out_stream);

if ((plan.Action=='p') && (plan.VarNumber>0))
 (void)BuildGnuStuff(plan,pt_model,pt_option,pt_pricing,pt_methods_available);

if (plan.Action=='t')
 {
 (void)FreeTest(pt_pricing->Test);
```

```
 (void)BuildGnuStuffTest(pt_model,pt_option,pt_pricing,pt_method);
 }
}while (NextSession(&plan,plan.Action,user)==OK);
}

(void)ExitVar();
(void)FreeErrorMsg();

 return OK;
}
```