```
   Help
#include "mer1d.h"

extern char* path_sep;
extern char **error_msg;

int MOD(Get)(int user,Planning *pt_plan,Model *
    model)
  {
  TYPEMOD* pt=(TYPEMOD*)(model->TypeModel);

  MOD(Init)(model);

  if (user==TOSCREEN)
    if  ((model->Show)(user,pt_plan,model))
      do
      {
        Fprintf(TOSCREEN,"____________________
    __________Model:%s{n",model->Name);

        ScanVar(pt_plan,user,&(pt->T));
        ScanVar(pt_plan,user,&(pt->S0));
        ScanVar(pt_plan,user,&(pt->Mu));
        ScanVar(pt_plan,user,&(pt->Sigma));

        ScanVar(pt_plan,user,&(pt->Divid));

        ScanVar(pt_plan,user,&(pt->R));
        ScanVar(pt_plan,user,&(pt->Lambda));
        ScanVar(pt_plan,user,&(pt->Mean));
        ScanVar(pt_plan,user,&(pt->Variance))
    ;
      }
      while ((model->Show)(user,pt_plan,model)
    );

  return ((model->Show)(TOSCREENANDFILE,pt_plan,
    model));
  }

int MOD(Show)(int user,Planning *pt_plan,Model *
```

```
    model)
  {
  TYPEMOD* pt=(TYPEMOD*)(model->TypeModel);
  VAR r_bs,divid_bs;

  Fprintf(user,"##Model:%s{n",model->Name);

  PrintVar(pt_plan,user,&(pt->T));
  PrintVar(pt_plan,user,&(pt->S0));
  PrintVar(pt_plan,user,&(pt->Mu));
  PrintVar(pt_plan,user,&(pt->Sigma));

  PrintVar(pt_plan,user,&(pt->Divid));
  divid_bs.Val.V_DOUBLE=log(1+pt->Divid.Val.V_
    DOUBLE/100);
  divid_bs.Vtype=DOUBLE;
  strcpy(divid_bs.Vname,"-->Instantaneous Divid
    end Rate");
  divid_bs.Viter=FORBID;
        PrintVar(pt_plan,user,&divid_bs);

  PrintVar(pt_plan,user,&(pt->R));
  r_bs.Vtype=DOUBLE;
  r_bs.Val.V_DOUBLE=log(1+pt->R.Val.V_DOUBLE/100
    );
  r_bs.Vtype=DOUBLE;
  strcpy(r_bs.Vname,"-->Instantaneous Interest
    Rate");
  r_bs.Viter=FORBID;
        PrintVar(pt_plan,user,&r_bs);


  PrintVar(pt_plan,user,&(pt->Lambda));
  PrintVar(pt_plan,user,&(pt->Mean));
  PrintVar(pt_plan,user,&(pt->Variance));
  return (model->Check)(user,pt_plan,model);
  }

int MOD(Check)(int user,Planning *pt_plan,Model *
    model)
  {
```

```
  TYPEMOD* pt=(TYPEMOD*)(model->TypeModel);
  int status=OK;
  char helpfile[MAX_PATH_LEN]="";

  if ((2*strlen(model->ID)+strlen("{{mod{{") +
    strlen("{{")
    +strlen("_doc.pdf"))>=MAX_PATH_LEN)
  {
    Fprintf(TOSCREEN,"%s{n",error_msg[PATH_TOO_
    LONG]);
        exit(WRONG);
  }

  strcpy(helpfile,path_sep);
  strcat(helpfile,"mod");
  strcat(helpfile,path_sep);

  strcat(helpfile,model->ID);
  strcat(helpfile,path_sep);

  strcat(helpfile,model->ID);
  strcat(helpfile,"_doc.pdf");

  status+=ChkVar(pt_plan,&(pt->T));
  status+=ChkVar(pt_plan,&(pt->S0));
  status+=ChkVar(pt_plan,&(pt->Mu));
  status+=ChkVar(pt_plan,&(pt->Sigma));
  status+=ChkVar(pt_plan,&(pt->Divid));

  status+=ChkVar(pt_plan,&(pt->R));
  status+=ChkVar(pt_plan,&(pt->Lambda));
  status+=ChkVar(pt_plan,&(pt->Mean));
  status+=ChkVar(pt_plan,&(pt->Variance));

  return Valid(user,status,helpfile);
  }


int MOD(Init)(Model *model)
  {
  TYPEMOD* pt=(TYPEMOD*)(model->TypeModel);
```

```
static int first=1;

if (first)
  {
  strcpy(pt->T.Vname,"Current Date");
  pt->T.Vtype=DATE;
  pt->T.Val.V_DATE=0.;
  pt->T.Viter=ALLOW;

  strcpy(pt->S0.Vname,"Spot");
  pt->S0.Vtype=PDOUBLE;
  pt->S0.Val.V_PDOUBLE=100.;
  pt->S0.Viter=ALLOW;

  strcpy(pt->Mu.Vname,"Trend");
  pt->Mu.Vtype=DOUBLE;
  pt->Mu.Val.V_DOUBLE=0.;
  pt->Mu.Viter=ALLOW;

  strcpy(pt->Sigma.Vname,"Volatility");
  pt->Sigma.Vtype=PDOUBLE;
  pt->Sigma.Val.V_PDOUBLE=0.2;
  pt->Sigma.Viter=ALLOW;

  strcpy(pt->Divid.Vname,"Annual Dividend Rat
  e");
  pt->Divid.Vtype=DOUBLE;
  pt->Divid.Val.V_DOUBLE=0.;
  pt->Divid.Viter=ALLOW;

  strcpy(pt->R.Vname,"Annual Interest Rate");
  pt->R.Vtype=DOUBLE;
  pt->R.Val.V_DOUBLE=10.;
  pt->R.Viter=ALLOW;

  strcpy(pt->Lambda.Vname,"Lambda");
  pt->Lambda.Vtype=DOUBLE;
  pt->Lambda.Val.V_DOUBLE=0.1;
  pt->Lambda.Viter=ALLOW;

  strcpy(pt->Mean.Vname,"Mean");
```

```
    pt->Mean.Vtype=DOUBLE;
    pt->Mean.Val.V_DOUBLE=0.;
    pt->Mean.Viter=ALLOW;

    strcpy(pt->Variance.Vname,"Variance");
    pt->Variance.Vtype=DOUBLE;
    pt->Variance.Val.V_DOUBLE=0.16;
    pt->Variance.Viter=ALLOW;


    first=0;
    }

  return OK;
  }


TYPEMOD Merton1dim;

MAKEMOD(Merton1dim);
```

References