Help

```
#include "
href../../mod/bsnd_default/bsnd_default_h_src.pdfbsnd_default.h"
#include "
href../../common/chk_h_src.pdfchk.h"
#include "
href../../common/error_msg_h_src.pdferror_msg.h"
#include "
href../../mod/hes1d/hes1d_pad/model_h_src.pdfmodel.h"
#include "pnl/pnl_matrix.h"

extern char *path_sep;

static void set_Model_Size(void *model)
{
  TYPEMOD *pt = (TYPEMOD *)(model);

  int sz = pt->Size.Val.V_PINT;

  pnl_vect_resize(pt->S0.Val.V_PNLVECT, sz);
  pnl_vect_set_all(pt->S0.Val.V_PNLVECT, 100.);
  pnl_vect_resize(pt->Sigma.Val.V_PNLVECT, sz);
  pnl_vect_set_all(pt->Sigma.Val.V_PNLVECT, 0.2);
  pnl_vect_resize(pt->Divid.Val.V_PNLVECT, sz);
  pnl_vect_set_all(pt->Divid.Val.V_PNLVECT, 0.);
}

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

  if (model->init == 0)
    {
      model->init = 1;
      model->nvar = 0;
      pt->T.Vname = "Current Date";
      pt->T.Vtype = DATE;
      pt->T.Val.V_DATE = 0.;
      pt->T.Viter = ALLOW;
      model->nvar++;
```

```
pt->Size.Vname = "Model Size";
pt->Size.Vtype = PINT;
pt->Size.Val.V_PINT = 3;
pt->Size.Viter = FORBID;
pt->Size.setter = set_Model_Size;
model->nvar++;

pt->S0.Vname = "Spot";
pt->S0.Vtype = PNLVECT;
pt->S0.Val.V_PNLVECT= pnl_vect_create_from_double(pt->Size.Val.V_PINT, 100
pt->S0.Viter = FORBID;
model->nvar++;

pt->Sigma.Vname = "Volatility";
pt->Sigma.Vtype = PNLVECT;
pt->Sigma.Val.V_PNLVECT= pnl_vect_create_from_double(pt->Size.Val.V_PINT,
pt->Sigma.Viter = FORBID;
model->nvar++;

pt->Divid.Vname = "Annual Dividend Rate";
pt->Divid.Vtype = PNLVECT;
pt->Divid.Val.V_PNLVECT= pnl_vect_create_from_double(pt->Size.Val.V_PINT,
pt->Divid.Viter = FORBID;
model->nvar++;

pt->Rho.Vname = "Correlation";
pt->Rho.Vtype = RGDOUBLEM11;
pt->Rho.Val.V_RGDOUBLEM11 = 0.;
pt->Rho.Viter = ALLOW;
model->nvar++;

pt->R.Vname = "Annual Interest Rate";
pt->R.Vtype = DOUBLE;
pt->R.Val.V_DOUBLE = 5.0;
pt->R.Viter = ALLOW;
model->nvar++;

pt->Intensity.Vname = "Default Intensity";
pt->Intensity.Vtype = PDOUBLE;
pt->Intensity.Val.V_PDOUBLE = 0.03;
```

```
      pt->Intensity.Viter = ALLOW;
      model->nvar++;

      pt->Recovery.Vname = "Recovery Rate";
      pt->Recovery.Vtype = PDOUBLE;
      pt->Recovery.Val.V_PDOUBLE = 0.4;
      pt->Recovery.Viter = ALLOW;
      model->nvar++;

    }
  return OK;
}


/**
 * Check function for BSND
 * @param user:
 * @param pt_plan:
 * @param model: the model to be checked
 *
 * general model check function
 */
int MOD(Check)(int user, Planning *pt_plan, Model *model)
{
  VAR *var;
  void *pt = (model->TypeModel);
  int status = OK;
  int i, nvar = 0;
  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");

  nvar = model->nvar;
  var = ((VAR *) pt);
  for (i = 0; i < nvar; i++)
    {
      status += ChkVar(pt_plan, &(var[i]));
      if (var[i].Vtype == PNLVECT && var[i].Val.V_PNLVECT->size != ((BSND_DEFAUL
        status += 1;
    }
  return Valid(user, status, helpfile);
}


TYPEMOD BlackScholesndim_default;

MAKEMOD(BlackScholesndim_default);
```