```
   Help
#include "bsnd.h"
#include "chk.h"
#include "error_msg.h"
#include "model.h"
#include "pnl/pnl_matrix.h"

extern char* path_sep;

static int adjust_compact_vector_size(VAR *v, int size,
    double default_value)
{
  PnlVectCompact *vc = v->Val.V_PNLVECTCOMPACT;

  if (vc==NULL)
    {
      if ((v->Val.V_PNLVECTCOMPACT=
           pnl_vect_compact_create (size, default_value))==
    NULL)
        return MEMORY_ALLOCATION_FAILURE;
      else
        return OK;
    }

  if ( vc->size == size ) return OK;
  return pnl_vect_compact_resize (vc, size, default_value);
}

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

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

  adjust_compact_vector_size(&pt->S0, sz, 100.);
  adjust_compact_vector_size(&pt->Sigma, sz, 0.2);
  adjust_compact_vector_size(&pt->Divid, sz, 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=PNLVECTCOMPACT;
    pt->S0.Val.V_PNLVECTCOMPACT=NULL;
    pt->S0.Viter=FORBID;
    model->nvar++;

    pt->Sigma.Vname = "Volatility";
    pt->Sigma.Vtype=PNLVECTCOMPACT;
    pt->Sigma.Val.V_PNLVECTCOMPACT=NULL;
    pt->Sigma.Viter=FORBID;
    model->nvar++;

    pt->Divid.Vname = "Annual Dividend Rate";
    pt->Divid.Vtype=PNLVECTCOMPACT;
    pt->Divid.Val.V_PNLVECTCOMPACT=NULL;
    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++;

      adjust_compact_vector_size(&pt->S0, pt->Size.Val.V_PI
    NT, 100.);
      adjust_compact_vector_size(&pt->Sigma, pt->Size.Val.
    V_PINT, 0.2);
      adjust_compact_vector_size(&pt->Divid, pt->Size.Val.
    V_PINT, 0.);
    }
  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->si
    ze != ((BSND*)pt)->Size.Val.V_PINT)
        status += 1;
    }
  return Valid(user,status,helpfile);
}


TYPEMOD BlackScholesndim;

MAKEMOD(BlackScholesndim);
```

References