Help

```
/*
 * Writen by David Pommier <david.pommier@gmail.com>
 * INRIA 2009
 */

#include <stdlib.h>
#include <math.h>
#include <stdio.h>
#include "pnl/pnl_vector.h"
#include "gridsparse_constructor.h"
#include "sparse_grid_constructor.h"

/* Here we suppose that father relation is construct and we
     construct other relations. */
static void GridSparse_compute_relation(GridSparse *G)
{
  int Nei,cur_index,dir,LorR;
  int PF[3]={0,0,0};
  int PS[3]={0,0,0};
  int Tab_Size[3]={G->dim,G->size,2};
  G->Ind_Son=pnl_hmat_int_create_from_int(3,Tab_Size,0);
  G->Ind_Neigh=pnl_hmat_int_create_from_int(3,Tab_Size,0);
  /* Compute son rules : */
  for(dir=0;dir<G->dim;dir++)
    {
      PF[0]=dir;
      PS[0]=dir;
      for(cur_index=0;cur_index<G->size;cur_index++)
        {
          PS[1]=cur_index;
          /* If Points even then Son(Father(i,dir,Right),
    dir,Left)= i. */
          PS[2]=(pnl_mat_int_get(G->Points,cur_index,dir)%2
    ==0)?1:0;
          PF[1]=pnl_hmat_int_get(G->Ind_Father,PS);
          PF[2]=1-PS[2];
          pnl_hmat_int_set(G->Ind_Son,PF,cur_index);
        }
    }
  /* End compute son rules, now computes neighbour rules  *
```

```
    /
  for(dir=0;dir<G->dim;dir++)
    {
      PF[0]=dir;
      PS[0]=dir;
      for(cur_index=0;cur_index<G->size;cur_index++)
        {
          PS[1]=cur_index;
          for(LorR=0;LorR<2;LorR++)
            {
              PS[2]=LorR;
              if (pnl_hmat_int_get(G->Ind_Son,PS)==0)
                {
                  Nei=pnl_hmat_int_get(G->Ind_Father,PS);
                  pnl_hmat_int_set(G->Ind_Neigh,PS,Nei);
                  if (!(Nei==0))
                    {
                      PF[1]=Nei;
                      PF[2]=1-PS[2];
                      pnl_hmat_int_set(G->Ind_Neigh,PF,cur_
    index);
                    };
                }
            }
        }
    }
  /* We don't have need of son relation so it is delete */
  pnl_hmat_int_free(&(G->Ind_Son));
}

/* To debug */
void GridSparse_check_relation(GridSparse *G)
{
  PnlVectInt * Current=pnl_vect_int_create(0);
  int cur_index,dir;
  int PF[3]={0,0,0};
  int PS[3]={0,0,0};
  /* Test father rules : */
  for(dir=0;dir<G->dim;dir++)
    {
      PF[0]=dir;
```

```
    for(cur_index=0;cur_index<G->size;cur_index++)
      {
        printf("test father of point %u >> ",cur_index);
        PF[1]=cur_index;
        pnl_mat_int_get_row(Current,G->Points,PF[1]);
        pnl_vect_int_print(Current);
        PF[2]=0;
        printf("left father >>");
        pnl_mat_int_get_row(Current,G->Points,pnl_hmat_
int_get(G->Ind_Father,PF));
        pnl_vect_int_print(Current);
        printf("right father >>");
        PF[2]=1-PS[2];
        pnl_mat_int_get_row(Current,G->Points,pnl_hmat_
int_get(G->Ind_Father,PF));
        pnl_vect_int_print(Current);
      }
  }
/* End compute son rules, now computes neighbour rules  *
  /
for(dir=0;dir<G->dim;dir++)
  {
    PF[0]=dir;
    for(cur_index=0;cur_index<G->size;cur_index++)
      {
        printf("test neighbour of point %u -> ",cur_ind
ex);
        PF[1]=cur_index;
        pnl_mat_int_get_row(Current,G->Points,PF[1]);
        pnl_vect_int_print(Current);
        PF[2]=0;
        printf("left neighbour -> ");
        pnl_mat_int_get_row(Current,G->Points,pnl_hmat_
int_get(G->Ind_Neigh,PF));
        pnl_vect_int_print(Current);
        printf("right neighbour -> ");
        PF[2]=1-PS[2];
        pnl_mat_int_get_row(Current,G->Points,pnl_hmat_
int_get(G->Ind_Neigh,PF));
        pnl_vect_int_print(Current);
      }
```

```
    }
  pnl_vect_int_free(&Current);
}


GridSparse *grid_sparse_create01(int dim, int lev)
{
  GridSparse * G = malloc(sizeof(GridSparse));
  create_grid_sparse_cpp(dim,lev,G);
  /*printf(">> Size of the Sparse Grid in Dimension %d and
    level %d is %d {n>> Note that we use Homogen Dirichlet Bounda
    ry conditions{n",dim,lev,G->size); */
  GridSparse_compute_relation(G);
  /*GridSparse_check_relation(G);*/
  G->Bnd=premia_pde_dim_boundary_create_from_int(G->dim);
  return G;
}

GridSparse *grid_sparse_create(const PnlVect * X0,const Pn
    lVect * X1,int lev)
{
  GridSparse * G = malloc(sizeof(GridSparse));
  create_grid_sparse_cpp(X0->size,lev,G);
  /*printf(">> Size of the Sparse Grid in Dimension %d and
    level %d is %d {n>> Note that we use Homogen Dirichlet Bounda
    ry conditions{n",X0->size,lev,G->size); */
  GridSparse_compute_relation(G);
  G->Bnd=premia_pde_dim_boundary_create(X0,X1);
  return G;
}

void GridSparse_free(GridSparse **G)
{
  pnl_hmat_int_free(& (*G)->Ind_Father);
  pnl_hmat_int_free(& (*G)->Ind_Son);
  pnl_hmat_int_free(& (*G)->Ind_Neigh);
  /*  PnlMatInt  * Ind_Next;   // Give Index of Next [Dim
    ension][Points] */
  pnl_vect_int_free(& (*G)->size_in_level);
  pnl_mat_int_free(& (*G)-> Points);
```

```
  premia_pde_dim_boundary_free(& (*G)->Bnd);
  free(*G);
  *G=NULL;
}
```

References