

[Help](#)

```
/* **** */
/* Written and (C) by David Pommier <david.pommier@gmail.com> */
/* **** */
/* INRIA 2009 */
/* **** */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "pnl/pnl_mathtools.h"
#include "pde_tools.h"
#include "gridsparse_functions.h"

static double mu = 0.5;
static double eta = 0.02;
static double Time_zero = 1.0;

double function_square(const PnlVect *X)
{
    int i;
    double prod = 1.0;
    for (i = 0; i < X->size; i++)
        prod *= GET(X, i) * (1.0 - GET(X, i));
    return prod;
}

double function_Test(const PnlVect *X)
{
    int i;
    double prod = 0.0;
    return 1.0;
    for (i = 0; i < X->size; i++)
        prod += GET(X, i) * (1.0 - GET(X, i));
    return prod;
}
```

```

double function_Test2(const PnlVect *X)
{
    int i;
    double prod = 1.0;
    for (i = 0; i < X->size; i++)
        prod *= sin(M_PI * GET(X, i));
    return prod;
}

double function_Test3(const PnlVect *X)
{
    int i = 0;
    double prod = 1.0;
    for (i = 0; i < X->size; i++)
        prod *= 1.0 / (sqrt(2 * M_PI * Time_zero) * eta) * exp(-pow(GET(X, i) - mu,
    return prod;
}

double function_Test4(const PnlVect *X)
{
    return MAX(0.5 - exp(GET(X, 0)), 0.0);
}

void Test_SpGrid_Poisson(void)
{
    PnlVect *V, *W, *W0;
    FILE *fic ;
    LaplacienSparseOp *Op = create_laplacien_sparse_operator();
    Op->G = grid_sparse_create01(2, 8);
    printf(">>>>> Test Sparse Grid On Poisson Problem \ n");
    initialise_laplacien_sparse_operator(Op);
    V = pnl_vect_create_from_zero(Op->G->size);
    W0 = pnl_vect_create_from_zero(Op->G->size);
    W = pnl_vect_create_from_zero(Op->G->size);
    GridSparse_apply_function(Op->G, V, function_Test2);
    Nodal_to_Hier(V, Op->G);
    pnl_vect_clone(W0, V);
    Hier_to_Nodal(W0, Op->G);
    GridSparse_Solve_Laplacien(Op, V, W);
}

```

```

Hier_to_Nodal(W, Op->G);
pnl_vect_axpby(0.0, W0, -1 * pow(M_PI, 2)*Op->G->dim, W);

fic = fopen("Data/Test_Graphe_n", "w");
if (fic == NULL)
{
    PNL_ERROR("Cannot open file", "GridSparse_fprint");
}
GridSparse_fprint(fic, Op->G, W);
fclose(fic);

fic = fopen("Data/Test_Graphe_n0", "w");
if (fic == NULL)
{
    PNL_ERROR("Cannot open file", "GridSparse_fprint");
}
GridSparse_fprint(fic, Op->G, W0);
fclose(fic);
pnl_vect_axpby(-1.0, W0, 1, W);
fic = fopen("Data/Test_Graphe_n2", "w");
if (fic == NULL)
{
    PNL_ERROR("Cannot open file", "GridSparse_fprint");
}
GridSparse_fprint(fic, Op->G, W);
fclose(fic);
printf(">> relativ error L2 = %7.4f\ n", pnl_vect_norm_two(W) / pnl_vect_norm_
laplacien_sparse_operator_free(&Op);
pnl_vect_free(&V);
pnl_vect_free(&W);
pnl_vect_free(&W0);
printf("end solve Laplacien on Sparse Grid \ n");
}

void Test_SpGrid_Heat(void)
{
    FILE *fic;
    int lev, dim;
    double delta_t = 5.0 * Time_zero;
    dim = 4;
    printf(">>>>> Test Sparse Grid On Heat Equation \ n");

```

```

for (lev = 4; lev <= 10; lev++)
{
    PnlVect *V, *W, *W0;
    HeatSparseOp *Op = create_heat_sparse_operator(eta);
    Op->G = grid_sparse_create01(dim, lev);
    Op->TG = premia_pde_time_homogen_grid(delta_t, 100);
    initialise_heat_sparse_operator(Op);
    V = pnl_vect_create_from_zero(Op->G->size);
    W0 = pnl_vect_create_from_zero(Op->G->size);
    W = pnl_vect_create_from_zero(Op->G->size);
    /* Initial condition, */
    GridSparse_apply_function(Op->G, V, function_Test3);
    Nodal_to_Hier(V, Op->G);
    GridSparse_Solve_heat(Op, V, W);
    Hier_to_Nodal(W, Op->G);
    /*pnl_vect_axpby(W,1,W0,0); */
    fic = fopen("Data/Test_Graphe_n", "w");
    if (fic == NULL)
    {
        PNL_ERROR("Cannot open file", "GridSparse_fprint");
    }
    GridSparse_fprint(fic, Op->G, W);
    fclose(fic);
    Time_zero += delta_t;
    GridSparse_apply_function(Op->G, W0, function_Test3);
    fic = fopen("Data/Test_Graphe_n0", "w");
    if (fic == NULL)
    {
        PNL_ERROR("Cannot open file", "GridSparse_fprint");
    }
    GridSparse_fprint(fic, Op->G, W0);
    fclose(fic);
    pnl_vect_axpby(-1.0, W0, 1.0, W);
    fic = fopen("Data/Test_Graphe_n2", "w");
    if (fic == NULL)
    {
        PNL_ERROR("Cannot open file", "GridSparse_fprint");
    }
    GridSparse_fprint(fic, Op->G, W);
    fclose(fic);
    printf(">> Size of the Sparse Grid in Dimension %d and level %d is %d -> R

```

```
Nodal_to_Hier(W0, Op->G);
/*pnl_vect_print(W0); */
fic = fopen("Data/Test_Graphe_nh", "w");
if (fic == NULL)
{
    PNL_ERROR("Cannot open file", "GridSparse_fprint");
}
GridSparse_fprint(fic, Op->G, W0);
fclose(fic);
heat_sparse_operator_free(&Op);
pnl_vect_free(&V);
pnl_vect_free(&W);
pnl_vect_free(&W0);
}
printf("End solve Heat on Sparse Grid \ n");
}

int main()
{
    Test_SpGrid_Poisson();
    Test_SpGrid_Heat();
}
```