Help

```
#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2008+2) //The "#els
#else
/*****************************************************
 *   CPS - A simple C PDE solver                     *
 *                                                   *
 *   Copyright (c) 2007,                             *
 *     Maya Briani       <m.briani@iac.rm.cnr.it>,   *
 *     Francesco Ferreri <francesco.ferreri@gmail.com>,  *
 *     Roberto Natalini  <r.natalini@iac.rm.cnr.it>, *
 *     Marco Papi        <m.papi@iac.rm.cnr.it>      *
 *                                                   *
 *****************************************************/
#include <math.h>
#include "cps_utils.h"
#include "cps_assertions.h"
#include "cps_stencil_operator.h"
#include "cps_stencil.h"
#include "cps_pde_term.h"

/*
   ACTUAL OPERATORS: this functions transparently
   implement discretization of single PDE terms,
   each operator creates a stencil with:
   - overall multiplying factor (coming from grid setup)
   - overall functional coefficient
   - terms mask, that is, the contribution of the stencil to
   matrixes for U(n+1) and U(n) according to position in each matrix
   (left: j-1, center: j, right: j+1)
*/

static stencil *sop_uxx(const pde_term *pterm, const grid *g)
{
  stencil *stemp;
  /* discretization scheme for Uxx terms:
     Uxx = (U(x+1,y) - 2U(x,y) + U(x-1,y))/(dx^2) */
  REQUIRE("pde_term_not_null", (pterm != NULL));
  REQUIRE("grid_not_null", (g != NULL));
  REQUIRE("valid_pde_term", pterm->type == UXX_TERM);
```

```
  STANDARD_CREATE(&stemp, stencil);
  stencil_set_function_factor(stemp, pterm->function_factor);
  stencil_set_factor(stemp, 1.0 / pow(g->delta[X_DIM], 2.0)); /* f = 1/dx^2 */

  stencil_set_weight(stemp, TIME_CUR, MODE_EXP, 1.0);
  stencil_set_weight(stemp, TIME_CUR, MODE_IMP, 0.5);
  stencil_set_weight(stemp, TIME_NXT, MODE_IMP, 0.5);

  stencil_set_value(stemp, XPY, 1.0);
  stencil_set_value(stemp, XY, -2.0);
  stencil_set_value(stemp, XMY, 1.0);

  return stemp;
}

static stencil *sop_uyy(const pde_term *pterm, const grid *g)
{
  stencil *stemp;

  /* discretization scheme for Uyy terms */
  REQUIRE("pde_term_not_null", (pterm != NULL));
  REQUIRE("grid_not_null", (g != NULL));
  REQUIRE("valid_pde_term", pterm->type == UYY_TERM);


  STANDARD_CREATE(&stemp, stencil);
  stencil_set_function_factor(stemp, pterm->function_factor);
  stencil_set_factor(stemp, 1.0 / pow(g->delta[Y_DIM], 2.0)); /* f = 1/dy^2 */
  stencil_set_weight(stemp, TIME_CUR, MODE_EXP, 1.0);
  stencil_set_weight(stemp, TIME_CUR, MODE_IMP, 0.5);
  stencil_set_weight(stemp, TIME_NXT, MODE_IMP, 0.5);

  stencil_set_value(stemp, XYP, 1.0);
  stencil_set_value(stemp, XY, -2.0);
  stencil_set_value(stemp, XYM, 1.0);

  return stemp;
}
```

```
static stencil *sop_uxy(const pde_term *pterm, const grid *g)
{

  stencil *stemp;
  /* discretization scheme for Uxy terms */
  REQUIRE("pde_term_not_null", (pterm != NULL));
  REQUIRE("grid_not_null", (g != NULL));
  REQUIRE("valid_pde_term", pterm->type == UXY_TERM);


  STANDARD_CREATE(&stemp, stencil);
  stencil_set_function_factor(stemp, pterm->function_factor);
  stencil_set_factor(stemp, 0.5 / (g->delta[X_DIM] * g->delta[Y_DIM]));
  stencil_set_weight(stemp, TIME_CUR, MODE_EXP, 1.0);
  stencil_set_weight(stemp, TIME_CUR, MODE_IMP, 0.5);
  stencil_set_weight(stemp, TIME_NXT, MODE_IMP, 0.5);

  stencil_set_value(stemp, XPYP , 1.0);
  stencil_set_value(stemp, XMYM, 1.0);
  stencil_set_value(stemp, XYP, -1.0);
  stencil_set_value(stemp, XPY, -1.0);
  stencil_set_value(stemp, XY, 2.0);
  stencil_set_value(stemp, XMY, -1.0);
  stencil_set_value(stemp, XYM, -1.0);
  return stemp;
}

static stencil *sop_ux(const pde_term *pterm, const grid *g)
{
  /* discretization scheme for Ux terms:
     Ux = (U(x+1,y) - U(x-1,y))/(2*dx) */
  stencil *stemp;

  REQUIRE("pde_term_not_null", (pterm != NULL));
  REQUIRE("grid_not_null", (g != NULL));
  REQUIRE("valid_pde_term", pterm->type == UX_TERM);


  STANDARD_CREATE(&stemp, stencil);
  stencil_set_function_factor(stemp, pterm->function_factor);
  stencil_set_factor(stemp, 0.5 / (g->delta[X_DIM])); /* f = 1/(2 * dx) */
```

```
  stencil_set_weight(stemp, TIME_CUR, MODE_EXP, 1.0);
  stencil_set_weight(stemp, TIME_CUR, MODE_IMP, 0.5);
  stencil_set_weight(stemp, TIME_NXT, MODE_IMP, 0.5);

  stencil_set_value(stemp, XPY, 1.0);
  stencil_set_value(stemp, XMY, -1.0);

  return stemp;
}

static stencil *sop_uy(const pde_term *pterm, const grid *g)
{
  stencil *stemp;
  /* discretization scheme for Uy terms */
  REQUIRE("pde_term_not_null", (pterm != NULL));
  REQUIRE("grid_not_null", (g != NULL));
  REQUIRE("valid_pde_term", pterm->type == UY_TERM);



  STANDARD_CREATE(&stemp, stencil);
  stencil_set_function_factor(stemp, pterm->function_factor);
  stencil_set_factor(stemp, 0.5 / (g->delta[Y_DIM])); /* f = 1/(2 * dy) */
  stencil_set_weight(stemp, TIME_CUR, MODE_EXP, 1.0);
  stencil_set_weight(stemp, TIME_CUR, MODE_IMP, 0.5);
  stencil_set_weight(stemp, TIME_NXT, MODE_IMP, 0.5);

  stencil_set_value(stemp, XYP, 1.0);
  stencil_set_value(stemp, XYM, -1.0);

  return stemp;
}

static stencil *sop_u(const pde_term *pterm, const grid *g)
{

  stencil *stemp;
  /* discretization scheme for U terms */
  REQUIRE("pde_term_not_null", (pterm != NULL));
  REQUIRE("grid_not_null", (g != NULL));
  REQUIRE("valid_pde_term", pterm->type == U_TERM);
```

```
  STANDARD_CREATE(&stemp, stencil);
  stencil_set_factor(stemp, 1.0);
  stencil_set_function_factor(stemp, pterm->function_factor);
  stencil_set_weight(stemp, TIME_CUR, MODE_EXP, 1.0);
  stencil_set_weight(stemp, TIME_CUR, MODE_IMP, 0.5);
  stencil_set_weight(stemp, TIME_NXT, MODE_IMP, 0.5);

  stencil_set_value(stemp, XY, 1.0);

  return stemp;
}



/*
   PUBLIC INTERFACE METHODS:
   operation which are visible to client objects (ehm, structures!)
*/

int stencil_operator_create(stencil_operator **s, int type)
{
  /* create a stencil_operator with given operator function */
  STANDARD_CREATE(s, stencil_operator);
  (*s)->type = type;
  switch (type)
    {
    case STENCIL_OP_UXX:
      (*s)->apply = sop_uxx;
      break;
    case STENCIL_OP_UYY:
      (*s)->apply = sop_uyy;
      break;
    case STENCIL_OP_UXY:
      (*s)->apply = sop_uxy;
      break;
    case STENCIL_OP_UX:
      (*s)->apply = sop_ux;
      break;
```

```
    case STENCIL_OP_UY:
      (*s)->apply = sop_uy;
      break;
    case STENCIL_OP_U:
      (*s)->apply = sop_u;
      break;
    }
  return OK;
}

int stencil_operator_destroy(stencil_operator **s)
{
  /* destroy a stencil operator */
  STANDARD_DESTROY(s);
  return OK;
}

int stencil_operator_apply(stencil_operator *sop, const pde_term *pterm, const g
{
  /* apply a stencil_operator to a term, producing a stencil in term itself */
  REQUIRE("stencil_operator_not_null", (sop != NULL));
  REQUIRE("pde_term_not_null", (pterm != NULL));

  sop->applied_stencil = sop->apply(pterm, g);
  sop->is_applied = 1;

  ENSURE("correctly_applied", IMPLIES(sop->is_applied, sop->applied_stencil != N
  return OK;
}
/* end -- stencil_operator.c */

#endif //PremiaCurrentVersion
```