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 "cps_stencil.h"
#include "cps_stencil_pattern.h"
#include "cps_utils.h"
#include "cps_assertions.h"
#include "cps_types.h"

/* private functions */

/* public interface functions */

int stencil_create(stencil **s)
{

  STANDARD_CREATE(s, stencil);
  (*s)->factor = 0.0;
  (*s)->function_factor = NULL;
  stencil_set_weight((*s), TIME_CUR, MODE_IMP, 0.0);
  stencil_set_weight((*s), TIME_CUR, MODE_EXP, 0.0);
  stencil_set_weight((*s), TIME_NXT, MODE_IMP, 0.0);
  stencil_set_weight((*s), TIME_NXT, MODE_EXP, 0.0);
  return OK;
}

int stencil_destroy(stencil **s)
{

  STANDARD_DESTROY(s);
```

```
  return OK;
}

int stencil_set_factor(stencil *s, double fact)
{
  /* set constant factor */
  REQUIRE("stencil_not_null", s != NULL);

  s->factor = fact;
  return OK;
}

int stencil_set_function_factor(stencil *s, const function *f)
{
  /* set function factor for given stencil */
  REQUIRE("stencil_not_null", s != NULL);
  REQUIRE("function_not_null", f != NULL);

  s->function_factor = f;
  return OK;
}

int stencil_set_value(stencil *s, int space, double val)
{
  /* set a coefficient in given space position */
  REQUIRE("stencil_not_null", s != NULL);
  REQUIRE("valid_space_position", space >= XY && space < MAX_STENCIL_SIZE);

  s->value[space] = val;
  return OK;
}

int stencil_set_weight(stencil *s, int time, int mode, double w)
{
  /* set weight for given mode */
  REQUIRE("stencil_not_null", s != NULL);
  REQUIRE("valid_time", time == TIME_CUR || time == TIME_NXT);
  REQUIRE("valid_mode", mode == MODE_EXP || mode == MODE_IMP);

  s->weight[time][mode] = w;
```

```
  ENSURE("weight_set", s->weight[time][mode] == w);
  return OK;
}

int stencil_apply(stencil *s, const grid *grid, int time, int mode, const grid_n
{
  grid_node *neigh;
  stencil_application *sapp;
  int pos;
  double value;

  /* apply a stencil centered in node */
  REQUIRE("stencil_not_null", s != NULL);
  REQUIRE("grid_not_null", grid != NULL);
  REQUIRE("grid_node_not_null", node != NULL);
  REQUIRE("valid_time", time == TIME_CUR || time == TIME_NXT);
  REQUIRE("valid_mode", mode == MODE_EXP || mode == MODE_IMP);


  stencil_pattern_create(sptrn);

  for (pos = XY; pos <= XPYP; pos++)
    {
      grid_node_neighbour(grid, pos, node, &neigh);
      stencil_evaluate(s, time, mode, pos, node, &value);
      if (value != 0.0)
        {
          stencil_application_create(&sapp);
          sapp->value = value;
          sapp->position = pos;
          if (grid_node_is_boundary(neigh))
            {
              stencil_application_set_boundary(sapp);
              stencil_application_set_order(sapp, 0);
            }
          else if (grid_node_is_external(neigh))
            {
              stencil_application_set_external(sapp);
              stencil_application_set_order(sapp, 0);
            }
          else if (grid_node_is_internal(neigh))
```

```
            {
              stencil_application_set_internal(sapp);
              stencil_application_set_order(sapp, neigh->order);
            }
          stencil_pattern_put((*sptrn), pos, sapp);
        }
      grid_node_destroy(&neigh);
    }

  ENSURE("stencil_pattern_created", (*sptrn) != NULL);
  return OK;
}

int stencil_evaluate(stencil *s, int time, int mode, int s_pos, const grid_node
{
  /* evaluate stencil on given node */
  REQUIRE("stencil_not_null", s != NULL);
  REQUIRE("valid_position", s_pos >= XY && s_pos <= XPYP);
  REQUIRE("valid_time", time == TIME_CUR || time == TIME_NXT);
  REQUIRE("valid_mode", mode == MODE_EXP || mode == MODE_IMP);

  if (s->value[s_pos] != 0.0)
    {
      *result = s->weight[time][mode] * s->factor
                * cps_function_evaluate(s->function_factor, node) * s->value[s_p
    }
  else
    {
      *result = 0.0;
    }
  return OK;
}
/* end -- stencil.c */

#endif //PremiaCurrentVersion
```