```
   Help
#if defined(PremiaCurrentVersion) && PremiaCurrentVersion <
     (2008+2) //The "#else" part of the code will be freely av
    ailable after the (year of creation of this file + 2)
#else
#ifndef FD_SOLVER_COMMON_H
#define FD_SOLVER_COMMON_H

#include <stdio.h>

#include <math/highdim_solver/laspack/highdim_matrix.h>
#include <math/highdim_solver/laspack/qmatrix.h>
#include <math/highdim_solver/laspack/highdim_vector.h>
#include <math/highdim_solver/laspack/errhandl.h>

/////////////////////////////////////////////////////
//
// Implementation of a finite differences solver
// for PDEs.
//

#define FDSOLVERMAXDIM  20

typedef int FDBOOL;

#define TRUE 1
#define FALSE 0

struct _FDSolver;

struct _FDSolverVectorFiller;

typedef int (*FDSolverVectorFillerInit_t)(struct _FDSolver
    *,
                                          struct _FDSolverV
    ectorFiller *);

typedef int (*FDSolverVectorFillerNextElem_t)(struct _FDSol
    ver *,
                                              struct _FDSol
    verVectorFiller *,
                                          unsigned *,
    double *);

typedef int (*FDSolverVectorFillerFinish_t)(struct _FDSol
    ver *,
                                        struct _FDSol
    verVectorFiller *);

typedef void (*FDSolverVectorFillerFree_t)(struct _FDSol
    ver *,
                                       struct _FDSol
    verVectorFiller *);


typedef struct _FDSolverVectorFiller
{

  FDSolverVectorFillerInit_t init;
  FDSolverVectorFillerNextElem_t next_elem;
  FDSolverVectorFillerFinish_t finish;
  FDSolverVectorFillerFree_t free;

  void *data;

} FDSolverVectorFiller;


struct _FDSolverCoMatricesFiller;

typedef int (*FDSolverCoMatricesFillerInit_t)(
                        struct _FDSolver *,
                        struct _FDSolverCoMatricesFiller *
                                             );

typedef int (*FDSolverCoMatricesFillerNextRow_t)(
                        struct _FDSolver *,
                        struct _FDSolverCoMatricesFiller *,
                        unsigned *, unsigned *
                                                );

typedef int (*FDSolverCoMatricesFillerNextElem_t)(
                        struct _FDSolver *,
                        struct _FDSolverCoMatricesFiller *,
                        unsigned *, double *, unsigned *
                                                 );

typedef int (*FDSolverCoMatricesFillerFinish_t)(
                        struct _FDSolver *,
                        struct _FDSolverCoMatricesFiller *
                                               );

typedef void (*FDSolverCoMatricesFillerFree_t)(
                        struct _FDSolver *,
                        struct _FDSolverCoMatricesFiller *
                                              );

typedef struct _FDSolverCoMatricesFiller
{

  FDSolverCoMatricesFillerInit_t init;
  FDSolverCoMatricesFillerNextRow_t next_row;
  FDSolverCoMatricesFillerNextElem_t next_elem;
  FDSolverCoMatricesFillerFinish_t finish;
  FDSolverCoMatricesFillerFree_t free;

  void *data;

} FDSolverCoMatricesFiller;

#define FD_SLICE_WALKER_RESET(wd,idim,ifirst,isize)
           {
  do
           {
  {
           {
    unsigned _k;
           {

           {
    for(_k=0; _k < (idim); _k++)
           {
      (wd)->coord[_k] = (ifirst) ?
```

```
           {
                                ((unsigned *)(ifirst))[_k] : 0;
           {

           {
    (wd)->pl = (wd)->coord + (idim) - 1;
           {
    (wd)->ph = (wd)->coord;
           {
    (wd)->sh = isize;
           {
    (wd)->f = ifirst;
           {
    (wd)->first = ifirst;
           {
    (wd)->size = isize;
           {
    (wd)->dim = idim;
           {
  }
           {
  while(0)

#define FD_SLICE_WALKER_UPDATE(wd,state,notify)
           {
do {
           {
    unsigned _k;
           {
    int not;
           {
    if(notify!=NULL)
           {
      *((int *)(notify)) = 0;
           {

           {
    for(_k=0; _k < (wd)->dim; _k++)
           {
    {
           {
```

```
if((wd)->coord[_k] < ((wd)->first ?
     {
                        (wd)->first[_k] : 0) +
     {
                          (wd)->size[_k] - 1)
     {
{
     {
  (wd)->coord[_k]++;
     {

     {
  if(state!=NULL)
     {
  {
     {
    if((wd)->coord[_k] ==
     {
        (((wd)->first ? (wd)->first[_k] : 0) +
     {
           (wd)->size[_k] - 1))
     {
    {
     {
      ((unsigned *)state)[_k] = 2;
     {
      not = 1;
     {
    }
     {
    else
     {
    {
     {
      ((unsigned *)state)[_k] = 1;
     {
      not = 1;
     {
    }
     {
  }
```

```
       {
    break;
       {
  }
       {
  else
       {
  {
       {
    (wd)->coord[_k] = (wd)->first ? (wd)->first[_k] : 0
;      {

       {
    if(state!=NULL)
       {
    {
       {
      ((unsigned *)state)[_k] = 0;
       {
      not = 1;
       {
    }
       {
  }
       {
}
       {

       {
if(_k>0 && (wd)->pl <= (wd)->coord + _k)
       {
  (wd)->pl = (wd)->coord + _k - 1;
       {

       {
if(*((wd)->ph) != *((wd)->sh) - 1 &&
       {
    (wd)->coord[_k] ==
       {
      (((wd)->first ? (wd)->first[_k] : 0) +
       {
               (wd)->size[_k] - 1))
           {
    {
           {
      (wd)->ph = (wd)->coord + _k;
           {
      (wd)->sh = (wd)->size + _k;
           {

           {
      if((wd)->first)
           {
        (wd)->f = (wd)->first + _k;
           {
    }
           {
    if(notify!=NULL)
           {
      *((int *)(notify)) = not;
           {
  }
           {
  while (0)


#define FD_WALKER_RESET(s,wd) {
  FD_SLICE_WALKER_RESET(wd,(s)->dim,NULL,(s)->size)

#define FD_WALKER_UPDATE(wd) FD_SLICE_WALKER_UPDATE(wd,NUL
    L,NULL)

#define FD_WALKER_ON_BOUNDARY(wd)
           {
  (*((wd)->pl) == ((wd)->f ? *((wd)->f) : 0)
           {
   ||
           {
   *((wd)->ph) ==  ((wd)->f ? *((wd)->f) : 0) + *((wd)->sh)
     - 1)  {

typedef struct _FDSolverCoordWalkerData
```

```
{

  unsigned coord[FDSOLVERMAXDIM];
  unsigned *pl;
  unsigned *ph;
  unsigned *sh;
  unsigned *f;
  unsigned *first, *size, dim;

} FDSolverCoordWalkerData;


typedef struct _FDSolver
{
  // Common data
  unsigned dim;
  unsigned size[FDSOLVERMAXDIM];
  unsigned offsA[FDSOLVERMAXDIM];
  unsigned offsB[FDSOLVERMAXDIM];  // TODO: Is it useful?

  QMatrix Ac,An;
  Matrix Bc,Bn;
  Vector x1,x2,b1,b2;
  Vector *xc,*xn;
  Vector *bc,*bn;

  FDBOOL is_A_symmetric;
  FDBOOL is_fully_explicit;
  FDBOOL is_fully_implicit;

  // PDE state
  double t, deltaT;

  // PDE specific routines
  FDSolverVectorFiller *b_filler;    // Boundary condition

  // Problem-specific data
  void *data;

  // Internal data
  FDSolverCoordWalkerData xwd;
  unsigned xidx, bidx;
} FDSolver;


#endif

#endif //PremiaCurrentVersion
```

References