

```

/*
 * Dirichlet.h
 *
 * For convenience and clarity, what might have been a single large
 * file Dirichlet.c has been organized as separate files:
 *
 * Dirichlet.c                provides the global organization,
 * Dirichlet_construction.c   does the actual computation of
 *                             the Dirichlet domain, and
 * Dirichlet_basepoint.c      moves the basepoint to a local maximum
 *                             of the injectivity radius function.
 *
 * Dirichlet_precision.c      makes an effort to keep roundoff error
 *                             under control whenever possible.
 *
 * Dirichlet_extras.c         adds all the "bells and whistles" to a
 *                             Dirichlet domain once it's computed.
 *                             It works out faces identifications,
 *                             determines ideal vertices, etc.
 *
 * Dirichlet_rotate.c         lets the UI spin the Dirichlet domain
 *                             in response to the user's mouse actions.
 *
 * This file (Dirichlet.h) provides the common declarations.
 */

#ifndef _Dirichlet_
#define _Dirichlet_

#include "kernel.h"

/*
 * Two O(3,1) matrices are considered equal if and only if each pair
 * of corresponding entries are equal to within MATRIX_EPSILON.
 *
 * Technical notes:
 *
 * (1) Originally (on a 680x0 Mac) I had MATRIX_EPSILON set to 1e-8,
 *     but it turned out that a product of two generators of an n-component
 *     circular chain complement was the identity to a precision of 2e-8.
 *     So I increased MATRIX_EPSILON. One doesn't want to make it too big,
 *     though, just in case the basepoint happens to start on a short
 *     geodesic of zero torsion.
 *
 * (2) When porting to other platforms (with lower floating point precision)
 *     this number (and probably many other constants) must be changed.
 */
#define MATRIX_EPSILON 1e-5

/*
 * The MatrixPair data structure stores an O3lMatrix and its inverse.
 */

typedef struct matrix_pair
{
    /*
     * m[0] and m[1] are the two matrices which are inverses of one another.
     */
    O3lMatrix      m[2];

    /*
     * height is the hyperbolic cosine of the distance either matrix
     * translates the origin (1, 0, 0, 0) of hyperbolic space.
     * height == m[0][0][0] == m[1][0][0].
     */
    double          height;

    /*
     * The left_ and right_child fields are used locally in
     * compute_all_products() in Dirichlet_compute.c to build a binary tree

```

```

    * of MatrixPairs. Normally MatrixPairs are kept on a doubly linked
    * list, using the prev and next fields. The next_subtree field is
    * used even more locally within tree-handling routines, to avoid doing
    * recursions on the system stack (for fear of stack/ heap collisions).
    */
    struct matrix_pair {
        *left_child,
        *right_child,
        *next_subtree;

    /*
     * Matrix pairs will be kept on doubly linked lists.
     */
    struct matrix_pair {
        *prev,
        *next;
    } MatrixPair;

/*
 * A MatrixPairList is a doubly linked list of MatrixPairs.
 * It typically includes the identity MatrixPair.
 */

typedef struct
{
    /*
     * begin and end are dummy nodes which serve to anchor
     * the doubly linked list. begin.prev and end.next
     * will always be NULL. The fields begin.m[], begin.dist,
     * end.m[] and end.dist are undefined and unused.
     */
    MatrixPair begin,
        end;
} MatrixPairList;

extern WEPolyhedron *compute_Dirichlet_domain(MatrixPairList *gen_list, double
vertex_epsilon);
extern void all_edges_counterclockwise(WFace *face, Boolean
redirect_neighbor_fields);
extern void redirect_edge(WEdge *edge, Boolean redirect_neighbor_fields);
extern void split_edge( WEdge *old_edge,
O3lVector cut_point,
Boolean set_Dirichlet_construction_fields);
extern FuncResult cut_face_if_necessary(WFace *face, Boolean
called_from_Dirichlet_construction);

extern void maximize_the_injectivity_radius(MatrixPairList *gen_list, Boolean *
basepoint_moved, DirichletInteractivity interactivity);
extern void conjugate_matrices(MatrixPairList *gen_list, double displacement[3]);
extern void free_matrix_pairs(MatrixPairList *gen_list);

extern void precise_o3l_product(O3lMatrix a, O3lMatrix b, O3lMatrix product);
extern void precise_generators(MatrixPairList *gen_list);

extern FuncResult Dirichlet_bells_and_whistles(WEPolyhedron *polyhedron);

#endif

```