Chapter 3.
Advanced Covering

Component: *Covering

The Advanced Covering functionality, in the ADM directory, is used to cover (i.e., fit a surface onto) circuits in solid or wire bodies. Example uses include end capping, post–translation corrections, and surface definition from curve data. Refer to Figure 3-1. Some of the advanced features of Advanced Covering are:

- Nonplanar, *n*–sided boundaries (must be plane–projectable)
- Position (G0) and tangent (G1) continuity with adjacent surfaces
- Boundary continuity can be individually specified on each edge
- Auxiliary point and curve constraints (G0 only)
- Post–covering gap reporting (diagnostics)
- Initial surface can be specified
- Existing surfaces can be re–covered

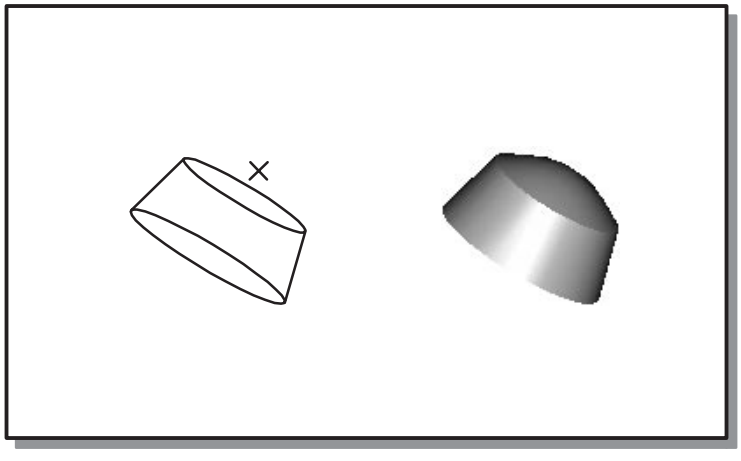

Figure 3-1. Covering with Advanced Covering

Advanced Covering Interface

Topic: *Covering

C++ Interface

The Advanced Covering C++ interface includes several functions and three classes: acovr_options, acovr_edge_constraint, and acovr_gap_report. The acovr_options object is used to define optional parameters that configure the covering problem.

The functions and classes are illustrated in the *C++ Advanced Covering Examples* section. Refer to the function and class reference templates for more information.

Scheme Interface

A Scheme interface is also supplied for learning and testing Advanced Covering in a Scheme environment, such as using Scheme AIDE. Refer to the Scheme extension reference templates for more information.

Some Advanced Covering Terminology

Topic: *Covering

This section explains some terms that are used in discussions of Advanced Covering.

Circuit

A *circuit* is a collection of edges that form a closed loop. For Advanced Covering, the loop must not be self–intersecting. The circuit edges can either lie on a collection of surfaces, or a collection of wires, but not some of each. Each edge in the circuit must be a wire or a sheet boundary; i.e., it must have only one coedge. Given an edge in an unambiguous (i.e. 2 and only 2 valid circuit edges at every vertex) circuit, api_adv_cover_circuit will automatically find the circuit to be covered. Advanced Covering cannot cover ambiguous circuits at this time.

In the case of re–covering, the circuit used will be the set of edges associated with the outer loop of the face.

G*n* Boundaries

The notation G*n* refers to smoothness while crossing a boundary. The larger the value of *n*, the greater the smoothness. A G0 boundary has position smoothness; i.e., there are no jumps in position. A G1 boundary has tangent smoothness; i.e., there are no jumps in tangency. Visually, tangency is slope; mathematically, it is the *xyz* first derivative.

For a surface, a boundary is (usually) a set of curves. For a curve, a boundary is always a set of points.

The junction between two boundary curves in a circuit must be at least G0 continuous; i.e., contain no position jumps.

For example, removing the top face from a cube gives an obvious circuit of 4 edges that has G0 continuity between the boundary curves. Removing the top face from a blended cube gives a circuit of eight edges that has G1 continuity between the boundary curves; i.e., no tangency (slope) jumps at curve joints. In this case, the boundary curves will not be G2—there will be jumps in the curvature at curve joints.

Guide Points and Guide Curves

Guide points and *guide curves* are constraints that are not part of the covering boundary; that is, they are not part of the final model topology. Guide constraints can only constrain position (they are G0 constraints).

Position and Tangency Gaps

Gaps are jumps at a boundary. Position (abbreviated as “pos”) gaps are jumps in the *xyz* **coordinates**. Tangency (abbreviated as “tan”) are gaps in the slope, or *xyz* **derivatives**.

Flattening

The flattening parameter is used to control how G1 constraints affect the shape of the covering surface. As an example, consider slicing off the top of a cone and then performing a G1 cover to make a smooth ”nosecone”. When the flattening parameter is zero, the cross–sectional shape of the nosecone will be roughly parabolic. When the flattening is large (.4 to .7) the cross–section will be flat in the center, with sharp ”shoulders” near the outside to accommodate the tangency constraints.

Initial Projection Plane

The Advanced Covering algorithm begins with an Initial Projection Plane. The boundary to be covered is projected into this plane to determine which points in the covering surface should be associated with the corresponding boundary points. The initial plane also affects the shape induced by G1 constraints; G1 constraints tend to be satisfied by making the covering surface bulge in the direction of the normal to the initial plane. By overriding the choice of initial plane, users can adjust the shapes resulting from G1 covers. The initial projection plane can be replaced with an initial surface.

Initial Surface

Rather than an initial projection plane, users can specify an initial surface shape instead of an initial projection plane. The Advanced Covering algorithm will deform this surface onto the boundary geometry, while attempting to maintain its gross shape. Specifying an initial surface is also useful for covering boundary circuits which are not plane projectable or which would result in a ”double valued” covering surface (relative to the initial plane). Advanced Covering produces the best results when it starts from an initial surface which has roughly the same shape as the desired covering surface.

Basic Steps for Using Advanced Covering

Topic: *Covering

The basic steps in using the Advanced Covering are:

1. Find an edge in the circuit to be covered or a face to be re–covered.
2. Create an options object (acovr_options), which is used to specify non–default values for continuity constraints, number of B–spline spans, projection plane or surface, and flattening factor (G1 covers only).
3. Use acovr_options methods to set any non–default values for continuity constraints, number of B–spline spans, projection plane or surface, and flattening factor, as desired.
4. Call api_advanced_cover with the appropriate signature.

General Advanced Covering Algorithm

Topic: *Covering

The general algorithm used by Advanced Covering is:

1. The user specifies a “plane projectable” boundary circuit.
2. Advanced Covering calculates a “projection plane” that is used as a “first guess” for the shape of the covering surface. This choice of projection plane can be overridden through the acovr_options object.
3. In addition, the projection plane can be replaced by an arbitrary spline surface, using the set_initial_face method. An initial spline surface is created from the projection plane or if specified, the geometry of the initial_face. Knots are added to the initial surface to ensure there at least num_spans in each direction.
4. Boundary and guide geometry is projected onto the initial surface, and the covering surface is deformed so that the projected points are “pulled onto” the boundary and guide geometry. The covering surface is a B–spline, which will have the user–specified number of spans. More spans will result in smaller gaps, but decreased performance.
5. The covering surface is stitched back into the ACIS model. Any edges that have position gaps larger than SPAresabs are converted into tolerant edges.

Advanced Covering Limitations

Topic: *Covering

This section lists some of the Advanced Covering limitations.

Circuits

- Covering circuits must be projectable onto the initial surface (not applicable for re–cover).
- Each edge in the circuit must have a single coedge.
- A circuit is not allowed to branch.
- A circuit is not allowed to be self–intersecting.
- “Double–valued” covering surfaces cannot be created. That is, no two points on the covered surface can project to the same point on the initial surface. For example, in Figure 3-2, some points on the top of the surface project to the same points on the plane as points on the folded under “flaps” (a few sample points are highlighted). Another example of such an illegal circuit is the seam of a baseball—the “lobes” are folded under the top, requiring a double–valued covering surface. However, if a sphere were used as the initial surface in this case, the double–valued covering could be avoided.

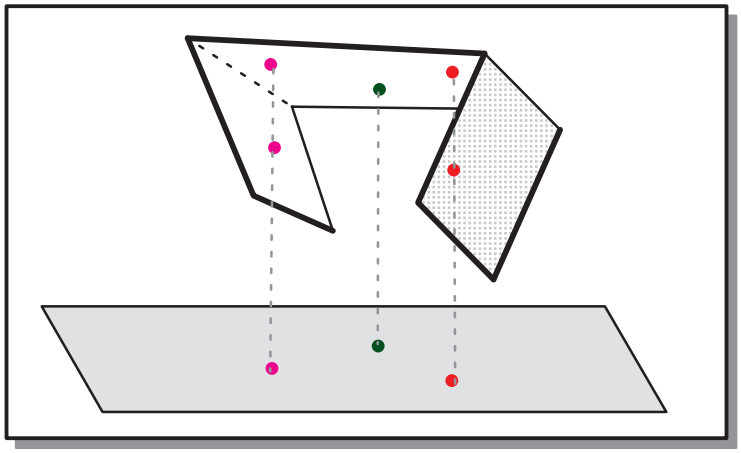

Figure 3-2. Illegal Double–Valued Covering Surface

Figure 3-3 illustrates a single–valued covering surface. The side “flaps” flare outward, so no two points on the covering surface project to the same point on the plane.

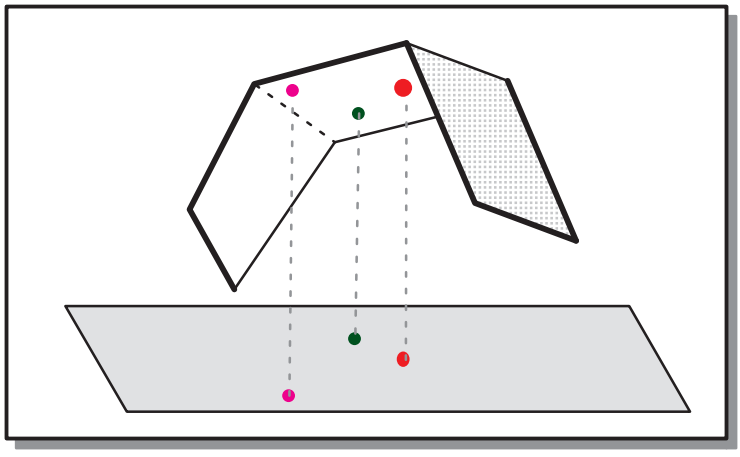

Figure 3-3. Legal Single–Valued Covering Surface

- A circuit used to specify a cover must not force the surface to be discontinuous. For example, performing a G1 cover of the circuit formed by removing the top face from a cube would force the covering surface to have a tangent discontinuity at the corners, so this operation is not allowed.

  This limitation does not apply if the discontinuities are coplanar with the adjacent surface. For example, performing a G1 cover of a star shape (with many sharp corners) cut out of the center of an existing (smooth) surface is allowed, because the discontinuities are compatible with the smooth covering surface.

Negative, Zero, or Near–Zero Draft Angle

Negative, zero, or near–zero draft angle bodies can be covered with a G0 boundary, but G1 boundary constraints may require tuning of the flattening parameter and/or initial plane. An example of this is the G1 cover of the top of a cyliner; the flattening parameter specifies how high the resulting “dome” will be. In general, specifying G1 boundary constraints which will result in a significantly different surface than the corresponding G0 cover can result in unaesthetic shapes.

Boundary Constraints G0 or G1 Only

Boundary constraints can enforce at most G1 continuity with adjacent surfaces—no higher continuity is allowed.

Guide Constraints G0 Only

Guide constraints can only specify G0 continuity. In contrast, boundary constraints may specify G1 continuity.

Cannot Mix Wire and Solid Edges

Boundary constraints come from circuits. Circuits are made up of edges from solid or wire bodies. However, within a single covering surface specification, all circuit edges must come from either a solid or a wire body, but not a mix of both.

C++ Advanced Covering Examples

Topic: *Covering, *Examples

This section contains *snippets* of C++ code to illustrate how to use Advanced Covering. These snippets are designed to be very simple starting points for developing your own code—they are not complete, compilable and runnable programs.

G0 Boundary with Default Settings

Topic: *Covering, *Examples

This example illustrates covering with G0 continuity on all boundary edges. Because G0 is the default continuity used, an options object (acovr_options) does not need to be passed to the API. In this example, the steps are:

1. Pick an edge on the circuit
2. Cover the circuit

```
#include ”kernel/geomhusk/ckoutcom.hxx”     // check_outcome
#include ”admhusk/acover/acovrapi.hxx”      // Advanced Covering interface

class FACE;                                 // ACIS ENTITY FACE

// ACIS initialization code & includes go here

outcome out(0);

out = api_initialize_deformable_modeling();
check_outcome(out);

FACE* my_face = NULL;

// my_edge is a pointer to an EDGE contained in the circuit we wish to cover

// cover the circuit using defaults
out = api_acovr_circuit(my_face, my_edge);
check_outcome(out);

// my_edge may now be invalid since
// boundary EDGEs are usually replaced with TEDGEs because of gaps

// my_face points to the new covering FACE.
// my_face has been stitched to the BODY on which the circuit lies.
```

```
// Code to save, etc. to the body goes here

out = api_terminate_deformable_modeling();
check_outcome(out);

// ACIS terminate code goes here
```

G1 Boundary with Options

Topic: *Covering, *Examples

This example illustrates how to use an acovr_options object to change the continuity on the boundary edges from G0 (default) to G1. The continuity values for boundary edges are specified in an acovr_edge_constraint object. The default continuity is changed by creating a G1 constraint object and setting its value in the acovr_options object.

Note *The* set_default_constraint *method copies values into the* acovr_options *object; it does not create an alias to the* acovr_edge_constraint *that is passed in.*

```
#include "kernel/geomhusk/ckoutcom.hxx"     // check_outcome
#include "admhusk/acover/acovrapi.hxx"      // Advanced Covering interface

class FACE;                                 // ACIS ENTITY FACE

// ACIS initialization code & includes go here

outcome out(0);

out = api_initialize_deformable_modeling();
check_outcome(out);

FACE* my_face = NULL;

// my_edge is a pointer to an EDGE contained in the circuit we wish to cover

// create options for G1 boundary
acovr_options my_options;               // create options with default values
acovr_edge_constraint my_constraint;    // create default constraint
my_constraint.set_continuity(acovr_G1); // override default constraint
                                        // continuity

// override default options; this COPIES values, does not alias
my_options.set_default_constraint(my_constraint);

// cover with options
out = api_advanced_cover(my_face, my_edge, &my_options);
check_outcome(out);

// my_edge may now be invalid since
// boundary EDGEs are usually replaced with TEDGEs because of gaps

// Code to save, etc. to the body goes here
```

```
out = api_terminate_deformable_modeling();
check_outcome(out);

// ACIS terminate code goes here
```

G1 Re–Cover with Options

Topic: *Covering, *Examples

This example illustrates how to re–cover an existing face, using an acovr_options object to change the continuity on the boundary edges from G0 (default) to G1. The continuity values for boundary edges are specified in a acovr_edge_constraint object. The default continuity is changed by creating a G1 constraint object and setting its value in the acovr_options object. Note that the current shape of the face has no effect on the shape obtained by re–covering, unless the initial_face parameter is set in the acovr_options object.

Note *The* set_default_constraint *method copies values into the* acovr_options *object; it does not create an alias to the* acovr_edge_constraint *that is passed in.*

```
#include "kernel/geomhusk/ckoutcom.hxx"     // check_outcome
#include "admhusk/acover/acovrapi.hxx"      // Advanced Covering interface

class FACE;                                 // ACIS ENTITY FACE

// ACIS initialization code & includes go here

outcome out(0);

out = api_initialize_deformable_modeling();
check_outcome(out);

// my_face is a pointer to the FACE that we wish to re–cover

// create options for G1 boundary
acovr_options my_options;               // create options with default values
acovr_edge_constraint my_constraint;    // create default constraint
my_constraint.set_continuity(acovr_G1); // override default constraint
                                        // continuity

// override default options; this COPIES values, does not alias
my_options.set_default_constraint(my_constraint);

// re–cover with options
out = api_advanced_cover(my_face, &my_options);
check_outcome(out);

// Code to save, etc. to the body goes here

out = api_terminate_deformable_modeling();
check_outcome(out);

// ACIS terminate code goes here
```

G1 Snapping

Topic: *Covering, *Examples

This example illustrates how to snap an existing face to be G1 continuous with its neighbors. This is done by re–covering the face with G1 continuity constraints on the boundaries, and the initial_face parameter in the acovr_options set to itself. The Advanced Covering algorithm will impose G1 continuity on the boundaries, while attempting to maintain the shape of the (initial) face.

Note *The* set_default_constraint *method copies values into the* acovr_options *object; it does not create an alias to the* acovr_edge_constraint *that is passed in.*

```
#include "kernel/geomhusk/ckoutcom.hxx"     // check_outcome
#include "admhusk/acover/acovrapi.hxx"      // Advanced Covering interface

class FACE;                                 // ACIS ENTITY FACE

// ACIS initialization code & includes go here

outcome out(0);

out = api_initialize_deformable_modeling();
check_outcome(out);

// my_face is a pointer to the FACE that we wish to re–cover

// create options for G1 boundary
acovr_options my_options;               // create options with default values
acovr_edge_constraint my_constraint;    // create default constraint
my_constraint.set_continuity(acovr_G1); // override default constraint
                                        // continuity

// override default options; this COPIES values, does not alias
my_options.set_default_constraint(my_constraint);

// use my_face as initial_face so that the interior shape is maintained
my_options.set_initial_face(my_face);

// for a face that is already near–tangent, use small flattening
// for large (60–90 degree) rotations use medium flattening (.4)
// for extreme (>90 degree) rotations use large flattening (.7)
// Tuning flattening will change the resulting shape
my_options.set_flattening(.1);          // assume face was near–tangent

// re–cover with options
out = api_advanced_cover(my_face, &my_options); // replaces geometry of
my_face
check_outcome(out);

// Code to save, etc. to the body goes here

out = api_terminate_deformable_modeling();
check_outcome(out);

// ACIS terminate code goes here
```

G1 Boundary with Symmetry Axis

Topic: *Covering, *Examples

This example illustrates how to use an acovr_options object to change the initial projection plane. The normal vector for the initial plane is specified in an acovr_options object. In this example, we dome the end–cap of a sliced cylinder.

Note *The* set_default_constraint *and* set_planet_normal *methods copy values into the* acovr_options *object; they do not create aliases to the* acovr_edge_constraint *and* SPAunit_vector *objects that are passed in.*

```
#include ″kernel/geomhusk/ckoutcom.hxx″     // check_outcome
#include ″admhusk/acover/acovrapi.hxx″      // Advanced Covering interface

class FACE;                                 // ACIS ENTITY FACE

// ACIS initialization code & includes go here

outcome out(0);

out = api_initialize_deformable_modeling();
check_outcome(out);

FACE* my_face = NULL;

// Initial model is a cylinder around z–axis, which has been sliced at an
// angle

// my_face is a pointer to the (elliptical) face created by the slice

// create options for G1 boundary
acovr_options my_options;               // create options with default values
acovr_edge_constraint my_constraint;    // create default constraint
my_constraint.set_continuity(acovr_G1); // override default constraint
                                        // continuity

// override default options; this COPIES values, does not alias
my_options.set_default_constraint(my_constraint);

// create unit vector in z direction
SPAunit_vector zdir(0., 0., 1.);

// and use it to specify initial projection plane (COPIES, does not alias)
my_options.set_plane_normal(zdir);

// use medium flattening since this is large (90 degree) change
my_options.set_flattening(.4);

// re–cover with options
out = api_advanced_cover(my_face, &my_options);
check_outcome(out);

// my_edge may now be invalid since boundary EDGEs are usually replaced with
// TEDGEs because of gaps
```

```
// Code to save, etc. to the body goes here

out = api_terminate_deformable_modeling();
check_outcome(out);

// ACIS terminate code goes here
```

G0 Boundary with Guide Points

Topic: *Covering, *Examples

This example illustrates covering with guide points. Guide geometry is specified by passing EDGE (curve) or VERTEX (point) pointers to the acovr_options object. Any guide edges passed in must be contained in an ACIS body. Guide geometry can be specified singly (using the set_guide method), or in an ENTITY_LIST (using set_guides).

```
#include ”kernel/geomhusk/ckoutcom.hxx”     // check_outcome
#include ”kernel/kerndata/lists/lists.hxx”  // class ENTITY_LIST;
#include ”admhusk/acover/acovrapi.hxx”      // Advanced Covering interface

class FACE;                                 // ACIS ENTITY FACE

// ACIS initialization code & includes go here

outcome out(0);

out = api_initialize_deformable_modeling();
check_outcome(out);

FACE* my_face = NULL;

// my_edge is a pointer to an EDGE contained in the circuit we wish to cover

// create options for guide points
// my_guides is an ENTITY_LIST containing VERTEXes
// my_tol is a double, which overrides the default gap tolerance between the
// covering surface and the vertex, if desired

acovr_options my_options;   // create options with default values
my_options.set_guides (my_guides);

// cover with options
out = api_advanced_cover(my_face, my_edge, &my_options);
check_outcome(out);

// my_edge may now be invalid since
// boundary EDGEs are usually replaced with TEDGEs because of gaps

// Code to save, etc. to the body goes here

out = api_terminate_deformable_modeling();
check_outcome(out);

// ACIS terminate code goes here
```

Mixed Continuity Boundary

Topic: *Covering, *Examples

This example illustrates covering with a mixed G0–G1 boundary. Users may want to set different continuities on different edges in a circuit. For example, blending a single edge of a cube and then removing the blend results in a circuit that should be covered with a G0 constraint on two of the edges and a G1 constraint on the other two edges. This is achieved by overriding the default constraint for specific edges using the set_boundary_constraint(s) method of acovr_options. The continuity level can simply be set to either acovr_G0 or acovr_G1 using the enumerated type acovr_continuity_level.

```
#include ″kernel/geomhusk/ckoutcom.hxx″     // check_outcome
#include ″admhusk/acover/acovrapi.hxx″      // Advanced Covering interface

class FACE;                                 // ACIS ENTITY FACE

// ACIS initialization code & includes go here

outcome out(0);

out = api_initialize_deformable_modeling();
check_outcome(out);

FACE* my_face = NULL;

// my_edge is a pointer to an EDGE contained in the circuit we wish to cover

// create options for mixed G0–G1 boundary
// my_G0_edge will have a G0 boundary, all others in my_circuit will have a
// G1 boundary
acovr_options my_options;               // create options with default values
acovr_edge_constraint my_constraint;    // create default constraint
my_constraint.set_continuity(acovr_G1); // override default constraint
                                        // continuity
my_options.set_default_constraint(my_constraint);   // override default
                                                    // options

// create G0 constraint object to override G1 default
acovr_edge_constraint G0_constraint;    // create constraint object
G0_constraint.set_continuity(acovr_G0); // set continuity to G0

// override default constraint continuity for only my_G0_edge
my_options.set_boundary_constraint(my_G0_edge,G0_constraint);

// cover with options
out = api_advanced_acover(my_face, my_edge, &my_options);
check_outcome(out);

// my_edge may now be invalid since
// boundary EDGEs are usually replaced with TEDGEs because of gaps

// my_face points to the new covering FACE.
// my_face has been stitched to the BODY on which the circuit lies.
```

```
// Code to save, etc. to the body goes here

out = api_terminate_deformable_modeling();
check_outcome(out);

// ACIS terminate code goes here
```

Specifying B–Spline Weight

Topic: *Covering, *Examples

This example illustrates using the num_spans parameter to change the number of control points in the covering surface. Fewer cotrol points can lead to wrinkling or large gaps in the covering surface, while surfaces with many control points suffer performance degradation. In this case, we will assume that the user would prefer a light–weight (fewer control points) surface.

```
#include ”kernel/geomhusk/ckoutcom.hxx”     // check_outcome
#include ”admhusk/acover/acovrapi.hxx”      // Advanced Covering interface
#include <stdio.h>                          // printf

class FACE;                                 // ACIS ENTITY FACE

// ACIS initialization code & includes go here

outcome out(0);

out = api_initialize_deformable_modeling();
check_outcome(out);

FACE* my_face = NULL;

// my_edge is a pointer to an EDGE contained in the circuit we wish to cover

// create options object to specify num_spans
acovr_options my_options;       // create options with default values
my_constraint.set_num_spans(10);// performance begins to degrade at num_spans
                                // ~ 20–25, num_spans>45 can take many seconds

// cover with options
out = api_advanced_cover(my_face, my_edge, &my_options);
check_outcome(out);

// my_edge may now be invalid since
// boundary EDGEs are usually replaced with TEDGEs because of gaps

// my_face points to the new covering FACE
// my_face has been stitched to the BODY on which the circuit lies

// Code to save, etc. to the body goes here

out = api_terminate_deformable_modeling();
check_outcome(out);

// ACIS terminate code goes here
```

Gap Reporting

Topic: *Covering, *Examples

This example illustrates gap reporting to check how well the boundary constraints were satisfied. The report_max_gap method of the acovr_options object examines each boundary edge and calculates the maximum constrained gap of each type (position and tangent) among the edges. G0 constraints only report position gaps, while G1 constraints report both position and tangency gaps. The maximum gaps for position and tangency are stored in an acovr_gap_report object. This object can then be queried for the information.

```
#include ”kernel/geomhusk/ckoutcom.hxx”     // check_outcome
#include ”admhusk/acover/acovrapi.hxx”      // Advanced Covering interface
#include <stdio.h>                          // printf

class FACE;                                 // ACIS ENTITY FACE

// ACIS initialization code & includes go here

outcome out(0);

out = api_initialize_deformable_modeling();
check_outcome(out);

FACE* my_face = NULL;

// my_edge is a pointer to an EDGE contained in the circuit we wish to cover

// create options for G1 boundary
acovr_options my_options;       // create options with default values
acovr_edge_constraint my_constraint;    // create default constraint
my_constraint.set_continuity(acovr_G1); // override default constraint
                                        // continuity
my_options.set_default_constraint(my_constraint);   // override default
                                                    // options

// cover with options
out = api_advanced_cover(my_face, my_edge, &my_options);
check_outcome(out);

// check boundary gap compliance
acovr_gap_report gr;
my_options.report_max_gap(gr);

// maximum position gap over all boundary edges
printf(”boundary position gap %f \n”, gr.get_pos_gap());

// maximum tangent gap over G1–constrained edges only
printf(”boundary position gap %f \n”, gr.get_tan_gap());

// my_edge may now be invalid since
// boundary EDGEs are usually replaced with TEDGEs because of gaps

// my_face points to the new covering FACE
// my_face has been stitched to the BODY on which the circuit lies
```

```
// Code to save, etc. to the body goes here

out = api_terminate_deformable_modeling();
check_outcome(out);

// ACIS terminate code goes here
```