Evolution of Geometric Sensitivity Derivatives from Computer Aided Design Models

William T. Jones*
NASA Langley Research Center, Hampton, VA 23681-2199
David Lazzara† and Robert Haimes‡
Massachusetts Institute of Technology, Cambridge, MA 02139

The generation of design parameter sensitivity derivatives is required for gradient-based optimization. Such sensitivity derivatives are elusive at best when working with geometry defined within the solid modeling context of Computer-Aided Design (CAD) systems. Solid modeling CAD systems are often proprietary and always complex, thereby necessitating ad hoc procedures to infer parameter sensitivity. A new perspective is presented that makes direct use of the hierarchical associativity of CAD features to trace their evolution and thereby track design parameter sensitivity. In contrast to ad hoc methods, this method provides a more concise procedure following the model design intent and determining the sensitivity of CAD geometry directly to its respective defining parameters.

Nomenclature

$\mathbb{R}$	Set of real numbers
$\mathbf{\Omega}$	Set of features
$\mathbf{E}$	Set of edges
$\mathbf{F}$	Set of faces
$\mathbf{L}$	Set of loops
$\mathbf{N}$	Set of nodes
$\mathbf{P}$	Set of parameters
$\mathbf{R}$	Set of discretized vertices
$\vec{\mathbf{e}}$	Edge position vector
$\vec{\mathbf{q}}$	Node position vector
$\vec{\mathbf{r}}$	Face position vector
n_Ω	Number of features
n_E	Number of edges
n_F	Number of faces
n_l	Number of faces/edges incident to node l
n_N	Number of nodes
n_p	Number of parameters
n_{L_j}	Number of loops for face F_j
t	Edge parameter
u	Tensor product surface parameter
v	Tensor product surface parameter

Subscripts

α	Left face index incident to edge
β	Right face index incident to edge
η	Face index indident to node
i	Range index
j	Face index
k	Edge index
l	Node index
w	Parameter index

Symbols

ϵ_0	Machine zero
ϵ_{tol}	Model tolerance
$\hat{\nabla}$	Edge gradient
μ	Edge *velocity*
∇	Surface gradient
ν	Surface *velocity*
$\|\ \ \|$	L_2 norm

Superscripts

$'$	Perturbed model instance

*Computer Engineer, Computational AeroSciences Branch, Senior Member AIAA
†PhD Candidate, Department of Aeronautics and Astronautics, Member AIAA
‡Principal Research Engineer, Department of Aeronautics and Astronautics, Senior Member AIAA

I. Introduction

GEOMETRY management is an essential component of any Multidisciplinary Design Analysis and Optimization (MDAO) environment, particularly if high-fidelity analysis tools are employed. When using a gradient-based optimization scheme, a key challenge is obtaining the necessary measure of the sensitivity of the geometric shape to the design parameters.

When tuning a given design variable (or set of variables) P to optimize a given output function L, gradient-based optimization requires the sensitivity of L to changes in P, $\partial L/\partial P$. The chain rule of differentiation can be applied to obtain $\partial L/\partial P$ as a composite of the individual components of the overall analysis. Therefore, each component of the analysis must provide the sensitivity of its output with respect to its inputs. For example, a three dimensional Computational Fluid Dynamics (CFD) analysis which produces some aerodynamic quantities of interest (e.g. lift, drag, etc.) must provide the sensitivity of these quantities with respect to the design variables of interest. Using the chain rule, the sensitivity of the aerodynamic quantities L to the design variable P can be viewed as,

$$\frac{\partial L}{\partial P} = \frac{\partial L}{\partial V}\frac{\partial V}{\partial B}\frac{\partial B}{\partial G}\frac{\partial G}{\partial P} \tag{1}$$

where the aerodynamic quantities L are computed on the input volume mesh V; V is constructed subject to a boundary mesh B; B depends on the surface geometry G; and finally G is defined by the design variables P.

Each of the analysis component derivatives can be constructed by differentiating the respective process (solver, mesh generator, etc.). Typically, this is accomplished by differentiating the respective tools by hand,[1] by using automatic differentiation tools,[2–6] or by the use of complex variables.[7] However, the geometric sensitivity derivatives, $\partial G/\partial P$ are elusive when the use of geometry defined by complex and often proprietary Computer-Aided Design (CAD) software is desired. With proprietary systems, software source code is unavailable and therefore differentiation of the source is not possible. Even when source is available, differentiation of said source proves impractical. For example, the very capable OpenCASCADE[8] software development platform for 3D CAD is available in open source. However, this object oriented library has at present more than 14,000 classes, making differentiation a daunting task. As a result, most research has focused on the development of ad hoc procedures to circumvent the problems associated with obtaining sensitivity derivatives from CAD software.

One family of approaches focuses on the construction of surrogate geometry representations that seek to approximate the model with newly defined parameters and known construction entities whereby the construction can be differentiated to produce sensitivity. Free form deformation methods[9,10] have been applied to a variety of shape design problems with success.[11,12] Commercial tools such as Sculptor^{TM}provide arbitrary shape deformation of the discrete analysis mesh to bypass the geometry model. While useful, these tools present new challenges such as: introducing a disconnect from the intended CAD design parameters; and the inability to realize the final design in a CAD environment (manufacturing, Product Lifecycle Management, etc).

Other techniques have employed finite-differencing of distinct CAD model instances to define sensitivity derivatives.[13–15] These methods are not only costly, but involve an assortment of challenges related to the proper definition of step size on finite precision computers. The cost is primarily associated with the potentially large number of instances of the geometry model that need to be generated. An instance of the geometry corresponding to the perturbation of each design parameter must be generated to serve in the finite-difference calculation of the sensitivity derivatives. Therefore, the cost of the sensitivity evaluation is directly proportional to the number of design variables. To mitigate the cost, a *farm* of CAD kernels can be run concurrently during the production of sensitivity derivatives.[13,15] One can imagine that this technique does not scale well as the number of design variables can largely outweigh the number of available CAD seats. More distressing is the selection of the appropriate design parameter perturbation step size. A step size for each parameter used in the finite-difference calculation must be chosen that will produce a small but significant change in the dependent geometry. As with any finite-difference calculation, the local truncation error is proportional to the step size so the step size must be small. However, the step size cannot be arbitrarily small as finite precision arithmetic will introduce error into the resulting sensitivity derivatives. To further complicate matters, a relevant step size must be chosen for each design parameter.

The finite-difference approach also poses a challenge in that the topology of the discretization (i.e. the number and connectivity of discrete vertices associated with each face and edge of the model) needs to remain unchanged in order for sensitivity information to be calculated at the vertices via divided difference. If, for example, model perturbations cause relative design motion of edges across a face and existing vertices on the face become occluded in the model interior, then a change in discretization topology occurs. This is often remedied by rescaling the position of vertices to adhere to the bounds of the newly defined face and preserve the discretization topology. However, it can be shown that such action is equivalent to artificially prescribing a *design velocity* to vertices which has no dependence on the perturbed driving parameter. Even worse is the situation when faces, edges or nodes cease to exist (or are added) after a model is regenerated. This often occurs when the geometry kernel enforces the underlying geometry to be closed for the new instance. Such a lack of consistency in the discrete representation of model instances presents yet another challenge to the finite-difference approach to sensitivity evaluation.

When a designer uses a feature-based CAD system to construct a geometry model, objects are represented as a collection of *features* which are constructed by following a hierarchical build recipe known as a feature-tree. In general the feature-tree is non-unique and is driven by a collection of parameters that size the model. The feature-tree along with the parameter space can be combined into the concept of a *master-model* representing an abstraction of the model. The CAD system processes the feature-tree and parameters within the master-model to define a specific *instance* of the model. It is the individual instance that is the subject of analysis while it is the master-model, along with its sensitivity to changes in the parameter space, that is the subject of optimization.

Each instance of the master-model results in a Boundary Representation (BRep). The BRep contains the geometric curves and surfaces that are functions of the parameters. The BRep also includes a model topology which collects the geometric entities into faces, loops, edges, and nodes and provides their connectivity information. The topology is directly related to both the design intent of the feature-tree and the construction methods of the underlying CAD system. Topological entities are driven explicitly by the design parameters or implicitly by construction operations. Therefore, analysis and optimization depends on information in the master-model parameter space as well as the BRep topology and gradient-based optimization requires the sensitivity of the model topology to the driving parameters.

To directly utilize the master-model within a gradient-based optimization context, one could imagine differentiating through the CAD system to directly obtain the geometric sensitivity to the parameters, but this is not possible for reasons cited above. However, an alternate approach is presented herein which makes direct use of the hierarchical associativity of the CAD features in a BRep to trace their evolution and thereby track sensitivity to design parameters. In the following sections we: detail the connectivity of the master-model; include a description of the infrastructure which supports the approach; describe the associativity of the driving parameters and feature components; discuss the development of sensitivity derivatives of the constituent components of a BRep; and provide example applications of the technique.

II. Master-Model Topological Connectivity

Consider a master-model containing a set of constructed features $\mathbf{\Omega} = \{\Omega_i\}$ $(i = 1, \ldots, n_\Omega)$ represented by a BRep topology consisting of a global listing of faces, edges, and nodes:

$$\begin{aligned} \text{Faces } \mathbf{F} &\rightarrow \{F_j\} \quad \text{for } j = 1, \ldots, n_F \\ \text{Edges } \mathbf{E} &\rightarrow \{E_k\} \quad \text{for } k = 1, \ldots, n_E \\ \text{Nodes } \mathbf{N} &\rightarrow \{N_l\} \quad \text{for } l = 1, \ldots, n_N, \end{aligned}$$

where the number of faces, edges, and nodes in the BRep are represented by n_F, n_E, and n_N respectively. Also, each face is bound by one or more loops of edges. Each face has a local loop set L_j with n_{L_j} loops numbered from 1 to n_{L_j}:

$$\text{Loops } \mathbf{L} \rightarrow \{L_{j,m}\} \quad \text{for } j = 1, \ldots, n_F \quad m = 1, \ldots, n_{L_j}.$$

Each feature contains a list of its constituent faces $\mathbf{F_\Omega} \subset \mathbf{F}$. For feature Ω_i, we define the associated topology tree in hierarchical fashion; however, sets of faces, edges, and nodes also contain connectivity.

When looping over the feature set $\mathbf{\Omega}$, the topology tree is traversed by looping over the face indices in $\mathbf{F_\Omega}$; each face branches to multiple loop indices in a set $\mathbf{L_F}$; each loop index also branches to multiple edge indices in a set $\mathbf{E_{L_F}} \subset \mathbf{E}$; all edge indices then branch to two node indices in a set $\mathbf{N_{E_{L_F}}} \subset \mathbf{N}$.

Looping over the sets $\mathbf{F}$, $\mathbf{E}$, or $\mathbf{N}$ yields connectivity information in a different manner. Each face $F_j \in \mathbf{F}$ points to a single feature index, $\Omega_{F_j} \in \mathbf{\Omega}$ which has a topology tree that expands as described for the feature set above. From the view of edges, the faces that intersect along edge E_k have indices stored in $\mathbf{F}_{E_k} \subset \mathbf{F}$ and the end-point node indices are contained in $\mathbf{N}_{E_k} \subset \mathbf{N}$ (both subsets have a cardinality of 2 as the BRep is assumed manifold). Finally, from the perspective of nodes, the faces and edges that are coincident to node N_l have indices stored in a set $\mathbf{F}_{\mathbf{N}_l} \subset \mathbf{F}$ and $\mathbf{E}_{\mathbf{N}_l} \subset \mathbf{E}$, respectively (the cardinalities of $\mathbf{F}_{\mathbf{N}_l}$ and $\mathbf{E}_{\mathbf{N}_l}$ are equal and represent the number of faces/edges incident to the node N_l which we denote n_l). In general each node in the BRep is constructed from a different number of intersecting faces/edges.

III. CAPRI

The current method is implemented within the Computational Analysis and Programming Interface (CAPRI).[16–18] CAPRI provides a vendor-neutral access point into CAD and can be used as the geometric foundation in any framework. This is accomplished by providing a unifying and simplifying Application Programming Interface (API) into CAD that also includes the ability to regenerate components via traversal of the feature-tree defined using the master-model concept.

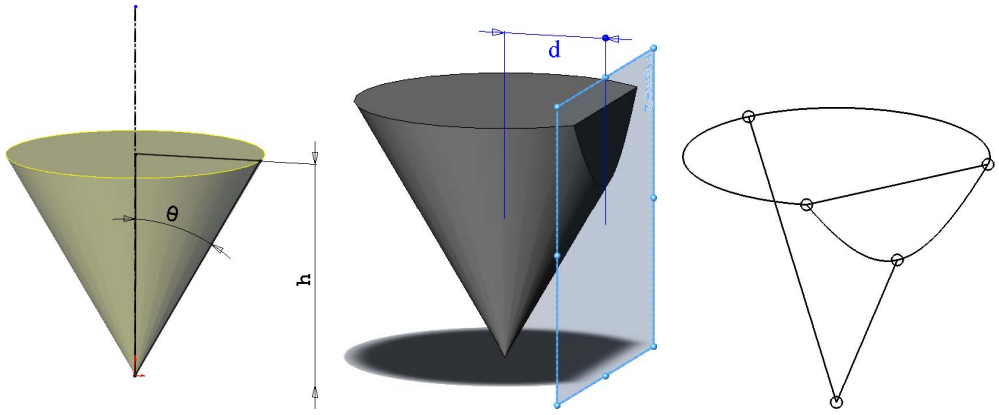
The CAPRI API offers a layer of abstraction from the specific methods of a given CAD kernel API while ultimately utilizing the original system to access the subject geometry. The API provides the operations that are common across the supported systems and provides for interrogation, data tagging, the creation of solid primitives, parameter modification, and model regeneration. By using the CAPRI API, an application is seamlessly integrated into all of the supported CAD systems without the need for software modification. CAPRI operations are generally restricted to manifold solid geometry, such as that defined by most modern CAD systems, and as such provides a closed topological description of the domain of interest. CAPRI also provides a closed tessellation[19,20] of the subject part that may be used to ensure physical consistency of the model. Most important to this work, CAPRI exposes the underlying feature-tree and associated parameter space defining the master-model in a concise neutral way. Therefore all of the geometric and topological information, as well as the parametric associativity, that are required for automated interrogation and regeneration of a CAD solid model is available to applications which employ the CAPRI API.

IV. Master-Model Associativity of Features and Driving Parameters

In order to determine the sensitivity of model topology to the parameters $\mathbf{P} = \{P_w\}$ $(w = 1, \ldots, n_P)$ in the master-model, it is necessary to determine the associativity between those driving parameters and the surfaces (trimmed to form faces) resulting from each feature. For example, Figure 1(a) contains a sketch with four entities (highlighted as thick black lines) used to create a revolve feature (shown with shaded faces). The entities in the sketch are: a vertical centerline reference; a vertical line segment; a horizontal line segment; and a diagonal line segment. The three line segments define a closed profile that defines the cone geometry when revolved about the vertical reference. The driving parameters for the line segments are the cone half-angle, θ, and height, h.

When a revolve geometry operator is applied to this sketch (thus creating a revolution feature), the CAD system's geometry kernel determines from the profile that a cone surface and a planar surface are the resulting geometry. The geometry kernel uses intersection algorithms to trim these surfaces according to the extent of the sketch entities and the cone BRep is created. Therefore, for the revolve feature in Figure 1(a) the planar surface is associated with the horizontal line segment and the cone surface is associated with the diagonal line segment. Furthermore, the angle and height parameters are associated with both surfaces through the line-surface associativity. When an additional planar cut-feature is introduced into the feature-tree, the model in Figure 1(b) results by regenerating the master-model. An additional distance parameter d drives the new planar surface. The resulting BRep topology for the new model is shown in Figure 1(c) (note that periodic cone surface is split resulting in an additional node and corresponding edge splits).

An additional associativity connection is made between the edges and nodes bounding each face to the underlying trimmed surface for that face. In the case of Figure 1, the angle and height parameters are associated with the cone and horizontal-plane surface and the distance parameter drives the vertical plane surface. Furthermore, the cone and planar surfaces in this example are defined by the geometry kernel using classical surface parameterizations:



(a) (b) (c)

Figure 1. (a) Sketch entities are piecewise continuous and driven by parameters, such as the cone angle and height in this case. The CAD geometry kernel processes the feature-tree containing the sketch entities and generates associated surfaces for the feature. (b) Adding a cut-operator with a planar face results in a new model instance and an additional parameter. (c) The final model topology shows edges (lines) and nodes (circles) bounding the geometry surfaces generated from these features.

$$\textbf{Vertical Plane:} \quad \vec{\mathbf{r}}_p = \vec{\mathbf{o}}_p + \begin{bmatrix} 0 \\ v_p - 1/2 \\ u_p \end{bmatrix}$$

$$\textbf{Cone:} \quad \vec{\mathbf{r}}_c = \vec{\mathbf{o}}_c + \begin{bmatrix} v_c \tan(\theta) \cos(u_c) \\ v_c \tan(\theta) \sin(u_c) \\ h v_c \end{bmatrix}$$

where $\vec{\mathbf{o}}_p = [d, 0, 0]^T$ is the relative plane origin and $\vec{\mathbf{o}}_c = [0, 0, 0]^T$ is the relative cone origin. We also presume a global reference frame with an origin at $\vec{\mathbf{o}}_0 = [0, 0, 0]^T$. In this example each surface is mapped to its own two-dimensional tensor product space with the corresponding coordinates u and v.

The driving parameters in the master-model must be associated to the corresponding parameters in the surface equations (in an automated fashion) resulting in a surface definition of the form $\vec{\mathbf{r}} = \vec{\mathbf{r}}(u, v; \mathbf{P_s})$ where $\mathbf{P_s} \subset \mathbf{P}$. Once this is established, a complete associativity chain results that links driving parameters from a sketch to the resulting construction of trimmed-surfaces in every feature of the model.

Unfortunately, across CAD systems there is no consistency in Features nor in the potential geometry emitted for the Feature. There are some CAD systems that produce a limited suite of surface types and have documented parameterizations, whereas there are others that require reverse engineering. Even though in practice there exists a finite number of surface types, this presents the potential for an extraordinary amount of work as CAD systems can have numerous features and each may need to be analyzed in order to generate $\vec{\mathbf{r}}(u, v; \mathbf{P_s})$. However, the problem is tractable and can be built up incrementally on an as-needed basis. We do not provide this exhaustive list of support here, but instead provide the basis upon which a proof-of-concept can be demonstrated.

Thus far we have the design parameters in the master-model associated with the surface parameters used to generate each feature face. Since the edges and nodes in the BRep topology essentially "sit on" each surface used to construct the model (within a tolerance), it is clear that the model topology sensitivity to the driving parameters will depend on the sensitivity of the underlying surfaces to those parameters. Therefore, symbolic differentiation of each surface $\vec{\mathbf{r}}(u, v; \mathbf{P_s})$ to these parameters is needed.

For implementation, we rewrite the known surface equations replacing the surface parameters with handles to the driving dimensions or other equations in the master-model. The new symbolic representation for these surface equations is then symbolically differentiated and evaluated. In the cone example, we require evaluation of the following non-zero symbolic derivatives:

$$\textbf{Vertical Plane} \left\{ \frac{\partial \vec{\mathbf{r}}_p}{\partial u_p} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad \frac{\partial \vec{\mathbf{r}}_p}{\partial v_p} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \frac{\partial \vec{\mathbf{r}}_p}{\partial d} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right.$$

$$\textbf{Cone} \left\{ \begin{array}{l} \dfrac{\partial \vec{\mathbf{r}}_c}{\partial u_c} = \begin{bmatrix} -v_c \tan(\theta) \sin(u_c) \\ v_c \tan(\theta) \cos(u_c) \\ 0 \end{bmatrix}, \quad \dfrac{\partial \vec{\mathbf{r}}_c}{\partial v_c} = \begin{bmatrix} \tan(\theta) \cos(u_c) \\ \tan(\theta) \sin(u_c) \\ h \end{bmatrix}, \quad \dfrac{\partial \vec{\mathbf{r}}_c}{\partial \theta} = \begin{bmatrix} v_c \sec^2(\theta) \cos(u_c) \\ v_c \sec^2(\theta) \sin(u_c) \\ 0 \end{bmatrix}, \\ \\ \dfrac{\partial \vec{\mathbf{r}}_c}{\partial h} = \begin{bmatrix} 0 \\ 0 \\ v_c \end{bmatrix} \end{array} \right.$$

These same steps are required for any surface representation used by the CAD geometry kernel to construct a given feature. Certainly the collection is large with numerous surface types, however, the process is repeatable. Even complicated surface types such as Non-Uniform Rational B-Splines (NURBS) can be handled in a similar manner by tracking associativity through their control points and knot vectors, for example.

V. Sensitivity of Model Topology

The model sensitivity to design parameters refers to the sensitivity of all faces, edges, and nodes in the BRep to a change in any driving parameter in the master-model. Since each face and edge is defined as continuous geometry, sensitivity information can be obtained by evaluating these entities at points anywhere in their domain; yet, in many applications the model topology is discretized, and sensitivity information is required at known discrete points on the faces and edges (nodes are discrete points by definition). In taking this view, the sensitivity method below is considered for sets of vertices on each topological entity even though it applies to any point evaluated in their domain.

We start the discussion of sensitivity generation with the sensitivity of faces of the BRep. Lower dimensional entities (edges and nodes) surprisingly introduce additional complications. Therefore, we will incrementally build up the discussion starting from the most straightforward entity.

V.A. Topology Parameterization

The CAPRI interface upon which the current work is based provides the ability to trace each face of the BRep back to its owning feature and also provides an association between parameters and features. Thus it is possible to also trace the face to its defining master-model parameters, $\mathbf{P}$.

We will consider a Euclidean space with origin $\vec{\mathbf{o}}_0$ and write the surface parameterization for face F_j using $\vec{\mathbf{r}}_j \in \mathbb{R}^3$ as

$$\vec{\mathbf{r}}_j = \vec{\mathbf{r}}_j(u_j, v_j; \mathbf{P}_j). \tag{2}$$

where u_j and v_j again represent the coordinates of the two-dimensional tensor product space mapping of face F_j.

Parameters are considered from two vantage points: the perspective of the specific entity derived from the parameter; and the perspective of the entire model. From the view of a face, we list the parameters used in generating the underlying surface of face F_j in the set $\mathbf{P}_j = \{P_i\}$ for $i = 1, \ldots, n_{P_j}$ (note that n_{P_j}, or the number of parameters used to generate F_j, is in general different for each face). From the perspective of the entire parameter space in the master-model, the collection of parameters used in the entire model will here be denoted as $\mathbf{P} = \mathbf{P}_1 \cup \mathbf{P}_2 \cup \ldots \cup \mathbf{P}_{n_F} = \{P_i\}$ for $i = 1, \ldots, n_P \leq \sum_{j=1}^{n_F} n_{P_j}$ (here we utilize an inequality because some surfaces may share common parameters, which are not repeated in $\mathbf{P}$). Each of the parameters in $\mathbf{P}$ may be stand-alone values that *drive* the model construction, although any parameter may instead be *driven* by a relation defined in the master-model equation set or by some equation defined external to the master-model (e.g. discipline-specific expression, assembly relations, etc.).

The parameterization for an edge is usually not directly obtained from the CAD geometry kernel because it is constructed by surface intersection algorithms (often as cubic B-splines). Therefore, by using $\vec{\mathbf{e}}_k \in \mathbb{R}^3$, edge E_k is assumed to have a parameterization

$$\vec{\mathbf{e}}_k = \vec{\mathbf{e}}_k(t_k; \mathbf{P_k}) \tag{3}$$

on the domain $T_k = [t_{k_{\min}}, t_{k_{\max}}] \in \mathbb{R}$ with coordinate $t_k \in T_k$. Since the edge is a construction from two intersecting surfaces, its driving parameter set is $\mathbf{P_k} = \mathbf{P}_\alpha \cup \mathbf{P}_\beta = \{P_i\}$ with $i = 1, \ldots, n_{P_k} \leq n_{P_\alpha} + n_{P_\beta}$, here $\mathbf{P}_\alpha$ and $\mathbf{P}_\beta$ are the parameter sets for faces F_α and F_β ($\alpha, \beta \in [1, n_F]$) which are incident to edge E_k. The value of n_{P_k} may be less than $n_{P_\alpha} + n_{P_\beta}$ when both surfaces share a common parameter.

Nodes, in general, are also generated without an output parameterization from the intersection algorithms of the geometry kernel. However, since the node is an artifact of intersecting surfaces, we can define a parameterization for node $\vec{\mathbf{q}}_l \in \mathbb{R}^3$ as

$$\vec{\mathbf{q}}_l = \vec{\mathbf{q}}_l\left(\vec{\mathbf{r}}_j; \mathbf{P}_l\right), \tag{4}$$

where the position vector $\vec{\mathbf{r}}_j$ follows from Eq. (2) for face $F_j \in \mathbf{F_{N_l}} \subset \mathbf{F}$ contributing to the node $\vec{\mathbf{q}}_l$. The subset of parameters that drive the node are $\mathbf{P}_l = \bigcup_{j=1}^{n_l} \mathbf{P}_j = \{P_i\}$ for $i = 1, \ldots, n_{P_l} \leq \sum_{j=1}^{n_l} n_{P_j}$. Here n_{P_l} represents the number of parameters which define node l and n_{P_j} represents the number of parameters defining face F_j.

V.B. Face Dependency on Design Parameters

The sensitivity of points on a face to parameters is the simplest to compute. Suppose face F_j is discretized by a set of vertices in the set $\mathbf{R}_j$. A subset of vertices are on the bounding edges of the face (designated as "points on an edge"), say $\mathbf{R}_{j,\text{edge}} \subset \mathbf{R}_j$, another subset of vertices, $\mathbf{R}_{j,\text{node}} \subset \mathbf{R}_j$, are at the intersection of bounding edges (these points are designated as "nodes" because they match nodes from the BRep topology description), while the remainder are on the face interior (we designate these as "points on the face"). In general, at any point $\vec{\mathbf{r}}_j \in \mathbf{R}_j$ evaluated on the interior, bounding edge, or node on a face we can write the derivative of the vertex position with respect to a parameter $P_i \in \mathbf{P}_j$ driving the face as

$$\begin{aligned} \frac{d}{dP_i}\left(\vec{\mathbf{r}}_j\right) &= \frac{d\vec{\mathbf{r}}_j}{dP_i} \\ &= \left(\frac{\partial \vec{\mathbf{r}}_j}{\partial u_j}\frac{\partial u_j}{\partial P_i} + \frac{\partial \vec{\mathbf{r}}_j}{\partial v_j}\frac{\partial v_j}{\partial P_i} + \frac{\partial \vec{\mathbf{r}}_j}{\partial P_i}\right). \end{aligned} \tag{5}$$

The terms $\frac{\partial \vec{\mathbf{r}}_j}{\partial u_j}$ and $\frac{\partial \vec{\mathbf{r}}_j}{\partial v_j}$ in Eq. (5) are readily obtained from the CAD geometry kernel as directional derivatives on the face. The $\frac{\partial \vec{\mathbf{r}}_j}{\partial P_i}$ term, however, is obtained solely by symbolically differentiating the surface definition with respect to P_i and evaluating. The two scalar multipliers, $\frac{\partial u_j}{\partial P_i}$ and $\frac{\partial v_j}{\partial P_i}$, are unknown in general. These terms correspond to a relative design motion in the domain of the face.

V.B.1. Points on the Face

For vertices that are not in any subset $\mathbf{R}_{j,\text{edge}}$ or $\mathbf{R}_{j,\text{node}}$ on face F_j, we find that Eq. (5) simplifies because $\frac{\partial u_j}{\partial P_i} = 0$ and $\frac{\partial v_j}{\partial P_i} = 0$, giving

$$\frac{d\vec{\mathbf{r}}_j}{dP_i} = \frac{\partial \vec{\mathbf{r}}_j}{\partial P_i}. \tag{6}$$

This implies, from a geometry perturbation perspective, that the points on the face interior show no relative design motion along the surface when a driving parameter P_i is perturbed. It also highlights, in the context of finite-differencing schemes, why rescaling vertices to fit within the bounds of the entity prescribes an artificial *design velocity* that does not reflect the true design motion of the underlying model. An exception to this occurs for faces constructed from a NURBS, bicubic spline, or other free-form representation of a surface, where parameters are often the coordinates of domain grid points themselves and relative motion is possible.

V.B.2. Points on an Edge

For faces defined by non-free-form surfaces, relative design motion may indeed be present for vertices $\vec{\mathbf{r}}_j \in \mathbf{R}_{j,\text{edge}}$ of face F_j. This can occur when perturbing the parameter P_i causes a displacement of an edge relative to the baseline surface after regeneration such as when the parameter change modifies an adjacent face. Thus, we can rewrite Eq. (5) to highlight the two terms that contribute to the total design motion of an edge:

$$\frac{d\vec{\mathbf{r}}_j}{dP_i} = \left(\frac{\partial \vec{\mathbf{r}}_j}{\partial P_i} + \nabla \vec{\mathbf{r}}_j \cdot \vec{\nu}\right) \tag{7}$$

where $\vec{\nu} = \left[\frac{\partial u_j}{\partial P_i}, \frac{\partial v_j}{\partial P_i}\right]^T$ and $\nabla \equiv \left[\frac{\partial}{\partial u_j}, \frac{\partial}{\partial v_j}\right]$. If we consider parameter P_i as continuous in $[P_{i_{\min}}, P_{i_{\max}}]$, then it is analogous to a "design time" over which the model follows a design trajectory of different instances. This implies a velocity ν in the $\mathbb{R}^2$ tensor product space that is projected into the space $\mathbb{R}^3$ by $\nabla \vec{\mathbf{r}}_j$. It is this velocity that contributes to relative design motion of an edge with respect to a surface. In addition, Eq. (7) can be analogously interpreted from a continuum mechanics perspective, where the edge is displaced first by a term denoting "time variation" of the surface, or design motion of the entire surface, and a second "convective" term representing a relative velocity, or design motion relative to the surface at the point $\vec{\mathbf{r}}_j$.

V.B.3. Point at a Node

Faces defined by non-free-form surfaces can also exhibit a relative design velocity at nodes as well. This occurs when a perturbation causes design motion of intersecting bounding edges on a face. We can apply Eq. (7) at the node position to determine the relative design velocity of the node with respect to the face.

V.C. Trim Curve Dependence on Driving Parameters

In deriving a method for edge sensitivity to driving parameters, we will consider the intersection trim curve E_k constructed from faces F_α and F_β $(\alpha, \beta \in [1, n_F], \alpha \neq \beta)$. We may proceed by considering any point along the trim curve domain or a set of vertices in the domain that results from a discretization of the trim curve. In taking the latter view, we suppose that F_α, F_β, and E_k are discretized, where the sets of vertices associated to each face are written as $\mathbf{R}_\alpha$ and $\mathbf{R}_\beta$. Along E_k we also have a set of vertices written as $\mathbf{R}_k$. At any point $\vec{\mathbf{e}}_k \in \mathbf{R}_k$, the derivative of the vertex position with respect to the parameter $P_i \in \mathbf{P}_k$ is

$$\begin{aligned}\frac{d}{dP_i}\left(\vec{\mathbf{e}}_k\right) &= \frac{d\vec{\mathbf{e}}_k}{dP_i} \\ &= \left(\frac{\partial \vec{\mathbf{e}}_k}{\partial t_k}\frac{\partial t_k}{\partial P_i} + \frac{\partial \vec{\mathbf{e}}_k}{\partial P_i}\right)\end{aligned} \tag{8}$$

The term $\frac{\partial \vec{\mathbf{e}}}{\partial t_k}$ in Eq. (8) is obtainable from the CAD geometry kernel as the local tangent to the trim curve. The remaining terms in Eq. (8) are unknown since the parameterization of the trim curve, which would contain the dependence on the driving parameters $\mathbf{P}_k$, is not provided as an output from intersection algorithms. This expression is defined over a 1D domain T_k and is similar to that for faces in Eq. (5) defined over a 2D tensor product space, thus the two terms in Eq. (8) correspond to different contributions for design motion of a trim curve. We can rewrite Eq. (8) as

$$\frac{d\vec{\mathbf{e}}_k}{dP_i} = \left(\frac{\partial \vec{\mathbf{e}}_k}{\partial P_i} + \hat{\nabla}\vec{\mathbf{e}}_k \cdot \mu\right) \tag{9}$$

where $\mu = [\frac{\partial t_k}{\partial P_i}]$ and $\hat{\nabla} = [\frac{\partial}{\partial t_k}]$. We again consider the driving parameter to be continuous over some range $[P_{i_{\min}}, P_{i_{\max}}]$ and act as a design time. In this 1D case, the "time varying" term $\frac{\partial \vec{\mathbf{e}}_k}{\partial P_i}$ refers to the design motion of the entire trim curve and the second "convective" term refers to the relative design motion with respect to the original curve (here the relative velocity μ is projected into the space $\mathbb{R}^3$ by $\hat{\nabla}\vec{\mathbf{e}}_k$). In other words, the unknown μ is seen as the dependence of position on the curve with respect to parameters because the domain coordinate t_k is usually expressed as a percentage of total curve length.

V.D. Node Dependence on Driving Parameters

Consider the node $N_l \in \mathbf{N}$ with coordinates $\vec{\mathbf{q}}_l$. Despite not having a parameterization, we know that the node will have no relative design velocity with respect to itself. Thus we can use Eqs. (7) and (9) as a template with zero relative velocity to specify

$$\frac{d\vec{\mathbf{q}}_l}{dP_i} = \frac{\partial \vec{\mathbf{q}}_l}{\partial P_i}. \tag{10}$$

From a geometric perspective, the total sensitivity of a node to a parameter is entirely defined by the "time varying" dependence of the node to design time over the continuous parameter range $[P_{i_{\min}}, P_{i_{\max}}]$.

V.E. Parameter Sensitivity on Edges with Minimum Velocity Method

From an analytic geometry perspective, the set of intersection for two surfaces will have the exact spatial coordinates as points on the trim curve, implying $\mathbf{R}_k = \mathbf{R}_\alpha \cap \mathbf{R}_\beta$ for vertices obtained after discretization of the face and edge. Another way of writing this for an element $\vec{\mathbf{e}}_k \in \mathbf{R}_k$ is $\vec{\mathbf{e}}_k \in \mathbf{R}_\alpha$ and $\vec{\mathbf{e}}_k \in \mathbf{R}_\beta$, which leads to the following equations that hold at the point $\vec{\mathbf{e}}_k \in \mathbf{R}_k$:

$$\begin{aligned} \vec{\mathbf{r}}_\alpha - \vec{\mathbf{r}}_\beta &= \mathbf{0} \\ \vec{\mathbf{r}}_\alpha - \vec{\mathbf{e}}_k &= \mathbf{0} \\ \vec{\mathbf{r}}_\beta - \vec{\mathbf{e}}_k &= \mathbf{0} \end{aligned} \tag{11}$$

where $\vec{\mathbf{r}}_\alpha$ and $\vec{\mathbf{r}}_\beta$ represent the closest points on edge k to the faces α and β respectively, $\vec{\mathbf{r}}_\alpha \in \mathbf{R}_\alpha$, and $\vec{\mathbf{r}}_\beta \in \mathbf{R}_\beta$. Although the three equations in Eq. (11) appear redundant, they represent the view of a point in Euclidean space $\mathbb{R}^3$ as determined by three different parameterizations: $\vec{\mathbf{r}}_\alpha(u_\alpha, v_\alpha; \mathbf{P}_\alpha)$, $\vec{\mathbf{r}}_\beta(u_\beta, v_\beta; \mathbf{P}_\beta)$ and $\vec{\mathbf{e}}_k(t_k; \mathbf{P}_k)$.

Due to the trimming algorithms undertaken by a geometry kernel, a BRep may identify a trim curve as the intersection of two surfaces even though the trim curve itself is not part of either surface (this occurs in consequence of using Newton's method in finding intersection points and tracing). This implies that each surface and trim curve are "intersecting" within a proximity tolerance (defined internal to the geometry kernel). Although this numerical geometry situation is different from the perspective of analytic geometry, the numerical implementation of trimming algorithms are susceptible to machine precision roundoff-error, ϵ_0. The proximity tolerance set within geometry kernels may be larger than machine precision as well (to improve computational efficiency of trimming algorithms) and is here denoted $\epsilon_{tol} \geq \epsilon_0$.

In the context of numerical geometry, our analytic constraint equations in Eq. (11) thus become

$$\begin{aligned} \vec{\mathbf{r}}_\alpha - \vec{\mathbf{r}}_\beta &= \vec{\epsilon}_1 \\ \vec{\mathbf{r}}_\alpha - \vec{\mathbf{e}}_k &= \vec{\epsilon}_2 \\ \vec{\mathbf{r}}_\beta - \vec{\mathbf{e}}_k &= \vec{\epsilon}_3. \end{aligned} \tag{12}$$

where $\vec{\epsilon}_1$, $\vec{\epsilon}_2$ and $\vec{\epsilon}_3$ are the offsets reached when the intersection search algorithm terminates its Newton method and $\| \vec{\epsilon}_1 \|, \| \vec{\epsilon}_2 \|, \| \vec{\epsilon}_3 \| \leq \epsilon_{tol}$. This indicates that the set of intersection points $\mathbf{R}_k$ has the property $\mathbf{R}_k \neq \mathbf{R}_\alpha \cap \mathbf{R}_\beta$. A BRep generated after surface trimming will define the faces $\mathbf{R}_\alpha$ and $\mathbf{R}_\beta$ (each with adjacent edges $\mathbf{R}_{\alpha,\text{edge}} \subset \mathbf{R}_\alpha$ and $\mathbf{R}_{\beta,\text{edge}} \subset \mathbf{R}_\beta$, respectively) along with the trim curve $\mathbf{R}_k$. These three BRep entities have a relative proximity that is within a ball B of radius ϵ_{tol} around each point in $\mathbf{R}_k$ for the nearest points in $\mathbf{R}_{\alpha,\text{edge}}$ and $\mathbf{R}_{\beta,\text{edge}}$. In this light, the constraint equations in Eq. (12) are similar, but not exactly redundant, and still represent the view of a surface-surface intersection from the perspective of three parameterizations. This approach ensures that all of the local information from the topology is used in deriving a sensitivity methodology. At a minimum, the first relation in Eq. (12) would suffice because the CAD geometry kernel employs information from the intersecting surfaces to construct the trim curve. Using all of the local topology information is consistent with the feature construction and allows for more insight into the edge displacement relative to its initial construction.

We further note that since Eq. (12) is not exact, we must obtain some point $\vec{\mathbf{r}}_\alpha$ at a (u_α, v_α) that minimizes $\| \vec{\mathbf{r}}_\alpha - \vec{\mathbf{e}}_k \|$; similarly, we must do the same on face F_β and determine some point $\vec{\mathbf{r}}_\beta$ at a (u_β, v_β)

that minimizes $\| \vec{\mathbf{r}}_\beta - \vec{\mathbf{e}}_k \|$. By providing the vertex coordinates of $\vec{\mathbf{e}}_k$, the CAD geometry kernel can return the points $\vec{\mathbf{r}}_\alpha$ and $\vec{\mathbf{r}}_\beta$ (including directional derivative information at these points) on faces F_α and F_β, respectively, that are nearest to $\vec{\mathbf{e}}_k$.

A variational analysis can be conducted to determine the sensitivity of vertices on faces and trim curves to those parameters $P_w \in \mathbf{P}$ which are also found in either $\mathbf{P}_\alpha$, $\mathbf{P}_\beta$, or both. We note here that it may be possible that $\epsilon_{tol} = \epsilon_{tol}(P_w)$, yet an expression for this could only be arbitrarily specified since access to the CAD geometry kernel source code is usually not available. Thus, we assume that $\epsilon_{tol} \neq \epsilon_{tol}(P_w)$, as if ϵ_{tol} were a constant value set in the geometry kernel. We consider a Taylor expansion around $\vec{\mathbf{r}}_\alpha$, $\vec{\mathbf{r}}_\beta$ and $\vec{\mathbf{e}}_k$ by writing

$$\begin{aligned}
\vec{\mathbf{r}}_\alpha(u_\alpha, v_\alpha; P_w + \Delta P_w)'_h &= \vec{\mathbf{r}}_\alpha(u_\alpha, v_\alpha; P_w) + \frac{d\vec{\mathbf{r}}_\alpha}{dP_w}\Delta P_w + \frac{1}{2}\frac{d^2\vec{\mathbf{r}}_\alpha}{dP_w^2}\Delta P_w^2 + \cdots \\
\vec{\mathbf{r}}_\beta(u_\beta, v_\beta; P_w + \Delta P_w)' &= \vec{\mathbf{r}}_\beta(u_\beta, v_\beta; P_w) + \frac{d\vec{\mathbf{r}}_\beta}{dP_w}\Delta P_w + \frac{1}{2}\frac{d^2\vec{\mathbf{r}}_\beta}{dP_w^2}\Delta P_w^2 + \cdots \\
\vec{\mathbf{e}}_k(t_k; P_w + \Delta P_w)' &= \vec{\mathbf{e}}_k(t_k; P_w) + \frac{d\vec{\mathbf{e}}_k}{dP_w}\Delta P_w + \frac{1}{2}\frac{d^2\vec{\mathbf{e}}_k}{dP_w^2}\Delta P_w^2 + \cdots .
\end{aligned} \tag{13}$$

For the purpose of analysis, we assume that the parameter perturbation, ΔP_w, is sufficiently small such that topology is preserved after regeneration with the new parameter value $P'_w = P_w + \Delta P_w$. The regenerated instance would also yield offset values $\vec{\epsilon}'_1$, $\vec{\epsilon}'_2$ and $\vec{\epsilon}'_3$ in Eq. (12). In this scenario, $\| \vec{\epsilon}'_1 - \vec{\epsilon}_1 \| < \epsilon_{tol}$, $\| \vec{\epsilon}'_2 - \vec{\epsilon}_2 \| < \epsilon_{tol}$ and $\| \vec{\epsilon}'_3 - \vec{\epsilon}_3 \| < \epsilon_{tol}$ are true, which from an implementation standpoint means we can set $\vec{\epsilon}'_1 = \vec{\epsilon}_1$, $\vec{\epsilon}'_2 = \vec{\epsilon}_2$ and $\vec{\epsilon}'_3 = \vec{\epsilon}_3$. This permits rewriting the first equation in Eq. (12) as

$$\begin{aligned}
\vec{\epsilon}'_1 &= \vec{\mathbf{r}}_\alpha(u_\alpha, v_\alpha; P_w + \Delta P_w)' - \vec{\mathbf{r}}_\beta(u_\beta, v_\beta; P_w + \Delta P_w)' \\
&= \underbrace{\vec{\mathbf{r}}_\alpha(u_\alpha, v_\alpha; P_w) - \vec{\mathbf{r}}_\beta(u_\beta, v_\beta; P_w)}_{\vec{\epsilon}_1} + \left(\frac{d\vec{\mathbf{r}}_\alpha}{dP_w} - \frac{d\vec{\mathbf{r}}_\beta}{dP_w}\right)\Delta P_w + \frac{1}{2}\left(\frac{d^2\vec{\mathbf{r}}_\alpha}{dP_w^2} - \frac{d^2\vec{\mathbf{r}}_\beta}{dP_w^2}\right)\Delta P_w^2 + \cdots \\
\mathbf{0} &= \left(\frac{d\vec{\mathbf{r}}_\alpha}{dP_w} - \frac{d\vec{\mathbf{r}}_\beta}{dP_w}\right)\Delta P_w + \frac{1}{2}\left(\frac{d^2\vec{\mathbf{r}}_\alpha}{dP_w^2} - \frac{d^2\vec{\mathbf{r}}_\beta}{dP_w^2}\right)\Delta P_w^2 + \cdots .
\end{aligned} \tag{14}$$

The second equation in Eq. (12) can then be rewritten as

$$\begin{aligned}
\vec{\epsilon}'_2 &= \vec{\mathbf{r}}_\alpha(u_\alpha, v_\alpha; P_w + \Delta P_w)' - \vec{\mathbf{e}}_k(t_k; P_w + \Delta P_w)' \\
&= \underbrace{\vec{\mathbf{r}}_\alpha(u_\alpha, v_\alpha; P_w) - \vec{\mathbf{e}}_k(t_k; P_w)}_{\vec{\epsilon}_2} + \left(\frac{d\vec{\mathbf{r}}_\alpha}{dP_w} - \frac{d\vec{\mathbf{e}}_k}{dP_w}\right)\Delta P_w + \frac{1}{2}\left(\frac{d^2\vec{\mathbf{r}}_\alpha}{dP_w^2} - \frac{d^2\vec{\mathbf{e}}_k}{dP_w^2}\right)\Delta P_w^2 + \cdots \\
\mathbf{0} &= \left(\frac{d\vec{\mathbf{r}}_\alpha}{dP_w} - \frac{d\vec{\mathbf{e}}_k}{dP_w}\right)\Delta P_w + \frac{1}{2}\left(\frac{d^2\vec{\mathbf{r}}_\alpha}{dP_w^2} - \frac{d^2\vec{\mathbf{e}}_k}{dP_w^2}\right)\Delta P_w^2 + \cdots ,
\end{aligned} \tag{15}$$

with the final equation in Eq. (12) becoming

$$\begin{aligned}
\vec{\epsilon}'_3 &= \vec{\mathbf{r}}_\beta(u_\beta, v_\beta; P_w + \Delta P_w)' - \vec{\mathbf{e}}_k(t_k; P_w + \Delta P_w)' \\
&= \underbrace{\vec{\mathbf{r}}_\beta(u_\beta, v_\beta; P_w) - \vec{\mathbf{e}}_k(t_k; P_w)}_{\vec{\epsilon}_3} + \left(\frac{d\vec{\mathbf{r}}_\beta}{dP_w} - \frac{d\vec{\mathbf{e}}_k}{dP_w}\right)\Delta P_w + \frac{1}{2}\left(\frac{d^2\vec{\mathbf{r}}_\beta}{dP_w^2} - \frac{d^2\vec{\mathbf{e}}_k}{dP_w^2}\right)\Delta P_w^2 + \cdots \\
\mathbf{0} &= \left(\frac{d\vec{\mathbf{r}}_\beta}{dP_w} - \frac{d\vec{\mathbf{e}}_k}{dP_w}\right)\Delta P_w + \frac{1}{2}\left(\frac{d^2\vec{\mathbf{r}}_\beta}{dP_w^2} - \frac{d^2\vec{\mathbf{e}}_k}{dP_w^2}\right)\Delta P_w^2 + \cdots .
\end{aligned} \tag{16}$$

In order for Eqs. (14), (15) and (16) to hold to first-order in ΔP_w, we must have

$$\begin{aligned}\frac{d\vec{\mathbf{r}}_\alpha}{dP_w} - \frac{d\vec{\mathbf{r}}_\beta}{dP_w} &= \mathbf{0} \\ \frac{d\vec{\mathbf{r}}_\alpha}{dP_w} - \frac{d\vec{\mathbf{e}}_k}{dP_w} &= \mathbf{0} \\ \frac{d\vec{\mathbf{r}}_\beta}{dP_w} - \frac{d\vec{\mathbf{e}}_k}{dP_w} &= \mathbf{0},\end{aligned} \tag{17}$$

where each term can be expanded as

$$\begin{aligned}\frac{d\vec{\mathbf{r}}_\alpha}{dP_w} &= \left(\frac{\partial\vec{\mathbf{r}}_\alpha}{\partial u_\alpha}\frac{\partial u_\alpha}{\partial P_w} + \frac{\partial\vec{\mathbf{r}}_\alpha}{\partial v_\alpha}\frac{\partial v_\alpha}{\partial P_w} + \frac{\partial\vec{\mathbf{r}}_\alpha}{\partial P_w}\right) = \left(\frac{\partial\vec{\mathbf{r}}_\alpha}{\partial P_w} + \nabla\vec{\mathbf{r}}_\alpha \cdot \nu_\alpha\right) \\ \frac{d\vec{\mathbf{r}}_\beta}{dP_w} &= \left(\frac{\partial\vec{\mathbf{r}}_\beta}{\partial u_\beta}\frac{\partial u_\beta}{\partial P_w} + \frac{\partial\vec{\mathbf{r}}_\beta}{\partial v_\beta}\frac{\partial v_\beta}{\partial P_w} + \frac{\partial\vec{\mathbf{r}}_\beta}{\partial P_w}\right) = \left(\frac{\partial\vec{\mathbf{r}}_\beta}{\partial P_w} + \nabla\vec{\mathbf{r}}_\beta \cdot \nu_\beta\right) \\ \frac{d\vec{\mathbf{e}}_k}{dP_w} &= \left(\frac{\partial\vec{\mathbf{e}}_k}{\partial t_k}\frac{\partial t_k}{\partial P_w} + \frac{\partial\vec{\mathbf{e}}_k}{\partial P_w}\right) = \left(\frac{\partial\vec{\mathbf{e}}_k}{\partial P_w} + \hat{\nabla}\vec{\mathbf{e}}_k \cdot \mu_k\right).\end{aligned} \tag{18}$$

We can then rewrite Eq. (17) using Eq. (18) to form a system of equations,

$$\begin{bmatrix} \nabla\vec{\mathbf{r}}_\alpha & -\nabla\vec{\mathbf{r}}_\beta & \mathbf{0} & \mathbf{0} \\ \nabla\vec{\mathbf{r}}_\alpha & \mathbf{0} & -\hat{\nabla}\vec{\mathbf{e}}_k & -\mathbf{I} \\ \mathbf{0} & \nabla\vec{\mathbf{r}}_\beta & -\hat{\nabla}\vec{\mathbf{e}}_k & -\mathbf{I} \end{bmatrix} \begin{bmatrix} \nu_\alpha \\ \nu_\beta \\ \mu_k \\ \frac{\partial\vec{\mathbf{e}}_k}{\partial P_w} \end{bmatrix} = \begin{bmatrix} \frac{\partial\vec{\mathbf{r}}_\beta}{\partial P_w} - \frac{\partial\vec{\mathbf{r}}_\alpha}{\partial P_w} \\ -\frac{\partial\vec{\mathbf{r}}_\alpha}{\partial P_w} \\ -\frac{\partial\vec{\mathbf{r}}_\beta}{\partial P_w} \end{bmatrix}, \tag{19}$$

which expands to

$$\begin{bmatrix} \frac{\partial\vec{\mathbf{r}}_\alpha}{\partial u_\alpha} & \frac{\partial\vec{\mathbf{r}}_\alpha}{\partial v_\alpha} & -\frac{\partial\vec{\mathbf{r}}_\beta}{\partial u_\beta} & -\frac{\partial\vec{\mathbf{r}}_\beta}{\partial v_\beta} & \mathbf{0} & \mathbf{0} \\ \frac{\partial\vec{\mathbf{r}}_\alpha}{\partial u_\alpha} & \frac{\partial\vec{\mathbf{r}}_\alpha}{\partial v_\alpha} & \mathbf{0} & \mathbf{0} & \frac{\partial\vec{\mathbf{e}}_k}{\partial t_k} & -\mathbf{I} \\ \mathbf{0} & \mathbf{0} & \frac{\partial\vec{\mathbf{r}}_\beta}{\partial u_\beta} & \frac{\partial\vec{\mathbf{r}}_\beta}{\partial v_\beta} & -\frac{\partial\vec{\mathbf{e}}_k}{\partial t_k} & -\mathbf{I} \end{bmatrix} \begin{bmatrix} \partial u_\alpha/\partial P_w \\ \partial v_\alpha/\partial P_w \\ \partial u_\beta/\partial P_w \\ \partial v_\beta/\partial P_w \\ \partial t_k/\partial P_w \\ \partial\vec{\mathbf{e}}_k/\partial P_w \end{bmatrix} = \begin{bmatrix} \frac{\partial\vec{\mathbf{r}}_\beta}{\partial P_w} - \frac{\partial\vec{\mathbf{r}}_\alpha}{\partial P_w} \\ -\frac{\partial\vec{\mathbf{r}}_\alpha}{\partial P_w} \\ -\frac{\partial\vec{\mathbf{r}}_\beta}{\partial P_w} \end{bmatrix}, \tag{20}$$

where $\mathbf{I}$ is the identity matrix. The system in Eq. (20) is over-determined and can be solved in a least-squares sense as $\mathbf{A}\mathbf{x} = \mathbf{b}$ where the least-squares solution, $\mathbf{x}^*$, is obtained by minimizing $R_{min} = \| \mathbf{A}\mathbf{x}^* - \mathbf{b} \|$. Since there is no guarantee that R_{min} will have order of magnitude ϵ_0, substituting the components of $\mathbf{x}^*$ back into Eq. (18) may result in violations of Eq. (17) on the order of $\mathcal{O}(R_{min})$:

$$\begin{aligned}\frac{d\vec{\mathbf{r}}_\alpha}{dP_w} - \frac{d\vec{\mathbf{r}}_\beta}{dP_w} &= \mathcal{O}(R_{min}) \neq 0 \\ \frac{d\vec{\mathbf{r}}_\alpha}{dP_w} - \frac{d\vec{\mathbf{e}}_k}{dP_w} &= \mathcal{O}(R_{min}) \neq 0 \\ \frac{d\vec{\mathbf{r}}_\beta}{dP_w} - \frac{d\vec{\mathbf{e}}_k}{dP_w} &= \mathcal{O}(R_{min}) \neq 0,\end{aligned}$$

This situation also implies that the vertex at $\vec{\mathbf{e}}_k$ will need to be assigned one of three possible sensitivity values:

$$\frac{d\vec{\mathbf{r}}_\alpha}{dP_w} \neq \frac{d\vec{\mathbf{r}}_\beta}{dP_w} \neq \frac{d\vec{\mathbf{e}}_k}{dP_w}.$$

In order to remedy this, the system in Eq. (20) is augmented with additional constraint equations in order to provide a single sensitivity result, $\vec{\psi}$, at vertex $\vec{\mathbf{e}}_k$:

$$
\begin{aligned}
\vec{\psi} &= \left(\frac{\partial \vec{\mathbf{r}}_\alpha}{\partial u_\alpha} \frac{\partial u_\alpha}{\partial P_w} + \frac{\partial \vec{\mathbf{r}}_\alpha}{\partial v_\alpha} \frac{\partial v_\alpha}{\partial P_w} + \frac{\partial \vec{\mathbf{r}}_\alpha}{\partial P_w} \right) \\
\vec{\psi} &= \left(\frac{\partial \vec{\mathbf{r}}_\beta}{\partial u_\beta} \frac{\partial u_\beta}{\partial P_w} + \frac{\partial \vec{\mathbf{r}}_\beta}{\partial v_\beta} \frac{\partial v_\beta}{\partial P_w} + \frac{\partial \vec{\mathbf{r}}_\beta}{\partial P_w} \right) \\
\vec{\psi} &= \left(\frac{\partial \vec{\mathbf{e}}_k}{\partial t} \frac{\partial t}{\partial P_w} + \frac{\partial \vec{\mathbf{e}}_k}{\partial P_w} \right).
\end{aligned} \tag{21}
$$

By including these additional constraints into the overdetermined system and augmenting $\mathbf{x}$ with $\vec{\psi}$, we obtain the new system

$$
\begin{bmatrix}
\nabla \vec{\mathbf{r}}_\alpha & -\nabla \vec{\mathbf{r}}_\beta & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\nabla \vec{\mathbf{r}}_\alpha & \mathbf{0} & -\hat{\nabla} \vec{\mathbf{e}}_k & -\mathbf{I} & \mathbf{0} \\
\mathbf{0} & \nabla \vec{\mathbf{r}}_\beta & -\hat{\nabla} \vec{\mathbf{e}}_k & -\mathbf{I} & \mathbf{0} \\
-\nabla \vec{\mathbf{r}}_\alpha & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \\
\mathbf{0} & -\nabla \vec{\mathbf{r}}_\beta & \mathbf{0} & \mathbf{0} & \mathbf{I} \\
\mathbf{0} & \mathbf{0} & -\hat{\nabla} \vec{\mathbf{e}}_k & -\mathbf{I} & \mathbf{I}
\end{bmatrix}
\begin{bmatrix}
\nu_\alpha \\ \nu_\beta \\ \mu_k \\ \dfrac{\partial \vec{\mathbf{e}}_k}{\partial P_w} \\ \vec{\psi}
\end{bmatrix}
=
\begin{bmatrix}
\dfrac{\partial \vec{\mathbf{r}}_\beta}{\partial P_w} - \dfrac{\partial \vec{\mathbf{r}}_\alpha}{\partial P_w} \\
-\dfrac{\partial \vec{\mathbf{r}}_\alpha}{\partial P_w} \\
-\dfrac{\partial \vec{\mathbf{r}}_\beta}{\partial P_w} \\
\dfrac{\partial \vec{\mathbf{r}}_\alpha}{\partial P_w} \\
\dfrac{\partial \vec{\mathbf{r}}_\beta}{\partial P_w} \\
\mathbf{0}
\end{bmatrix}. \tag{22}
$$

We can then write Eq. (22) in a block structure form with

$$
\underbrace{\left[\begin{array}{c|c} \mathbf{A} & \mathbf{0} \\ \hline \mathbf{G} & \mathbf{A}_I \end{array}\right]}_{\mathbb{A}}
\underbrace{\left[\begin{array}{c} \mathbf{x} \\ \hline \vec{\psi} \end{array}\right]}_{\mathbb{X}}
=
\underbrace{\left[\begin{array}{c} \mathbf{b} \\ \hline \mathbf{b}_G \end{array}\right]}_{\mathbb{B}}, \tag{23}
$$

where $\mathbf{A}$, $\mathbf{x}$ and $\mathbf{b}$ are defined as in Eq. (20) and

$$
\mathbf{G} = \begin{bmatrix}
-\frac{d\vec{\mathbf{r}}_\alpha}{du_\alpha} & -\frac{d\vec{\mathbf{r}}_\alpha}{dv_\alpha} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & -\frac{d\vec{\mathbf{r}}_\beta}{du_\beta} & -\frac{d\vec{\mathbf{r}}_\beta}{dv_\beta} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\frac{d\vec{\mathbf{e}}_k}{dt_k} & -\mathbf{I}
\end{bmatrix},
$$

$$
\mathbf{A}_I = \begin{bmatrix} \mathbf{I} \\ \mathbf{I} \\ \mathbf{I} \end{bmatrix},
$$

$$
\mathbf{b}_G = \begin{bmatrix} \frac{\partial \vec{\mathbf{r}}_\alpha}{\partial P_w} \\ \frac{\partial \vec{\mathbf{r}}_\beta}{\partial P_w} \\ \mathbf{0} \end{bmatrix}.
$$

The system in Eq. (23) remains over-determined and has the form $\mathbb{A}\mathbb{X} = \mathbb{B}$, which can be solved in a least-squares sense to give $\tilde{R}_{min} = \| \mathbb{A}\mathbb{X}^* - \mathbb{B} \|$ with solution $\mathbb{X}^*$. When $\tilde{R}_{min} > \epsilon_0$, the resulting sensitivity vector $\vec{\psi}$ is equivalent to a weighted linear combination of the vectors $\frac{d\vec{\mathbf{r}}_\alpha}{dP_w} \neq \frac{d\vec{\mathbf{r}}_\beta}{dP_w} \neq \frac{d\vec{\mathbf{e}}_k}{dP_w}$, which are obtained by back-substituting the components ν_α, ν_β, μ_k and $\frac{\partial \vec{\mathbf{e}}_k}{\partial P_w}$ of $\mathbb{X}^*$ into Eq. (18). In this case, we can write

$$
\vec{\psi} = \lambda_\alpha \frac{d\vec{\mathbf{r}}_\alpha}{dP_w} + \lambda_\beta \frac{d\vec{\mathbf{r}}_\beta}{dP_w} + \lambda_f \frac{d\vec{\mathbf{e}}_k}{dP_w}
$$

and determine the weights λ_α, λ_β and λ_f by setting up a 3×3 system. On the other hand, if the solution to the least-squares problem gives $\mathcal{O}(\tilde{R}_{min}) \approx \epsilon_0$, then the weights become $\lambda_\alpha, \lambda_\beta, \lambda_f \approx \frac{1}{3}$ and

$$\vec{\psi} \approx \frac{d\vec{\mathbf{r}}_\alpha}{dP_w} \approx \frac{d\vec{\mathbf{r}}_\beta}{dP_w} \approx \frac{d\vec{\mathbf{e}}_k}{dP_w}$$

to within machine-precision.

In order to solve the system $\mathbb{AX} = \mathbb{B}$ in a least-squares sense, it is possible to use the normal equations, QR decomposition, or the singular value decomposition (SVD). The truncated SVD is known to be the most robust approach when $\mathbb{A}$ is rank-deficient or singular, which can be the case when constructing $\mathbb{A}$ with geometry information at points on trim curves.

When $\mathbb{A}$ has full rank and is non-singular, the solution $\mathbb{X}^*$ is unique; however, when either of these conditions are not satisfied the solution is non-unique because adding any vector projected in the null space of $\mathbb{A}$ to $\mathbb{X}$ will also satisfy $\mathbb{AX} = \mathbb{B}$. In this case, a unique solution $\mathbb{X}^*_{\text{min}}$ is defined to have minimum norm $\| \mathbb{X}^*_{\text{min}} \|$ by projecting the zero-vector into the null space. This implies that each element of $\mathbb{X}^*_{\text{min}}$ has a norm that is at most $\mathcal{O}(\| \mathbb{X}^*_{\text{min}} \|)$ in magnitude. Since $\mathbb{X}$ consists of relative velocity terms ν_α, ν_β and μ_k, the unique minimum norm solution also provides an upper bound to the relative velocity magnitudes.

The minimum norm solution to $\mathbb{AX} = \mathbb{B}$ implies that the sensitivity $\vec{\psi}$ has an upper-bound norm as a result of relative velocities having a bounded norm. However, this result is the *lowest* norm sensitivity possible within the infinite solutions in the null space of $\mathbb{A}$. From a geometry perturbation perspective, vertex $\vec{\mathbf{e}}_k$ is perturbed to a new location $\vec{\mathbf{e}}'_k = \vec{\mathbf{e}}_k + \vec{\psi}\Delta P_w$ (to first order) with the smallest displacement relative to each baseline intersecting face and trim curve. Therefore, in this approach the direction of design motion (i.e. design velocity, or sensitivity of the vertex) of the trim curve is occurring along a "minimum energy" trajectory (compared to other possible solutions in the null space) in the face and edge domains, where specific energy can be written in the domain space of F_α, F_β and E_k as

$$\begin{aligned} F_\alpha\text{:} \quad & \tfrac{1}{2} \| \nu_\alpha \|^2 \\ F_\beta\text{:} \quad & \tfrac{1}{2} \| \nu_\beta \|^2 \\ E_k\text{:} \quad & \tfrac{1}{2}\mu_k^2 \ . \end{aligned}$$

An additional feature to this minimum velocity approach is that the resulting design velocity vector contains a component that lies in the direction of the null space of $\mathbb{A}$. This component direction is actually the tangent vector direction at the vertex $\vec{\mathbf{e}}_k$ on the trim curve E_k. This can be shown by writing the tangent at $\vec{\mathbf{e}}_k$ in a conventional manner using the local directional derivatives of F_α and F_β at the intersection and substituting the results into Eq. (17). It is clear that the resulting system has a null space which contains the direction of the trim curve tangent at $\vec{\mathbf{e}}_k$. This leads to the conclusion, seen both theoretically and in practice, that the null space direction acts as the locus of points swept by all possible design velocity vectors that can be attributed to a perturbed point on an edge; in other words, any perturbation direction chosen for a point on an edge will cause design motion to a position on the null space vector, which is parallel to the tangent vector at the point of the unperturbed edge and separated by a distance proportional to the perturbation magnitude. The infinite design velocity solutions correlate to the ambiguity associated with point tracking in finite-differencing, where the new t value for a point on a perturbed curve is uncertain in comparison to its t value on the unperturbed curve. The minimal velocity approach prescribes a unique design velocity that is driven by the constructed model geometry.

V.F. Parameter Sensitivity on Nodes with Minimum Velocity Method

Node topological entities may be treated in a similar fashion to edges. After discretizing a model, a vertex on an edge has at most two intersecting faces (F_α and F_β) that contribute to its position. However, the vertex at a node has multiple intersecting edges (and faces) contributing to its position as shown in Figure 2. This intersection node is created by collapsing the end-points of intersecting trim curves to a single point, as seen in Figure 3. Such end-points are only used if they are within some tolerance sphere of each other, as specified within the CAD system.

When the parameters that drive intersecting surfaces are perturbed, the perturbations may be such that the trim curve end-points move in tandem and maintain a relative spacing within a tolerance sphere of each other. This circumstance is illustrated in Figure 4. In some instances, the intersecting surfaces may also be constructed with a design intent that preserves the single intersection node, thus disallowing the trim curve end-points from moving outside of a tolerance sphere in proximity of each other, as shown in Figure

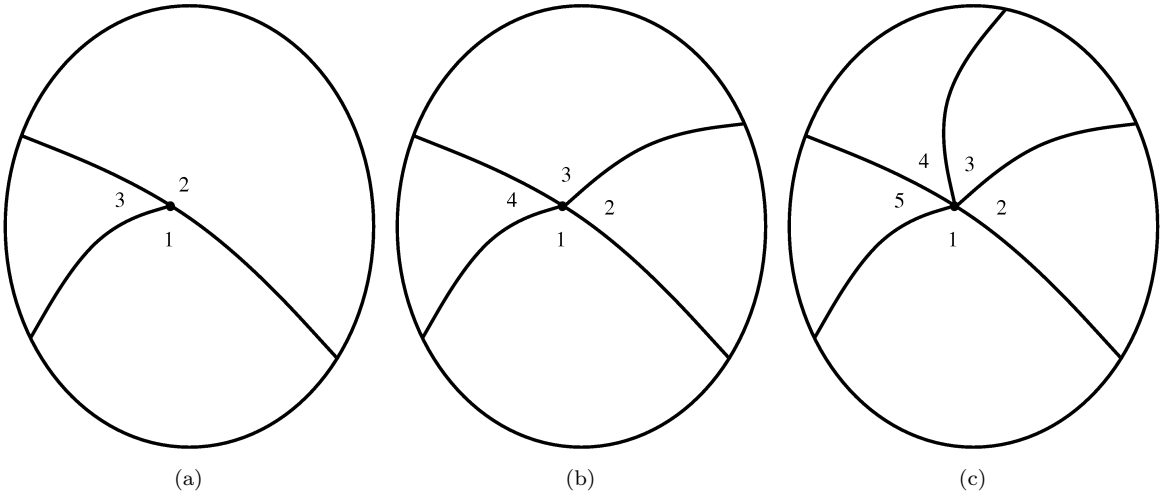


Figure 2. Illustration of (a) three, (b) four, and (c) five surfaces intersecting at a node.

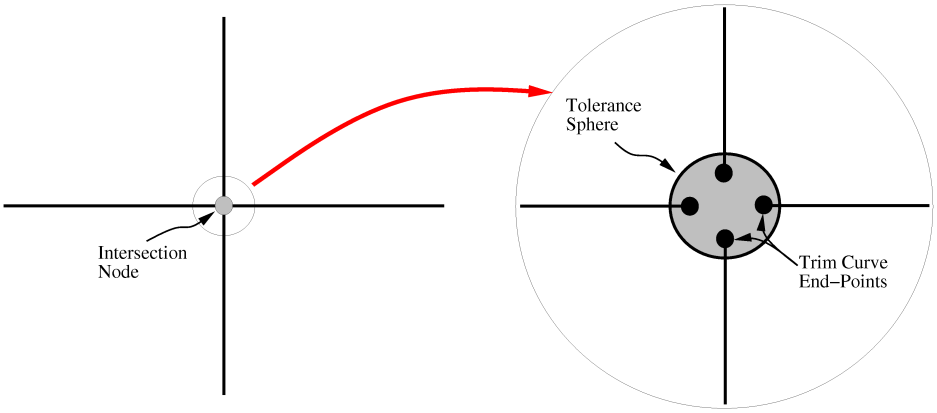


Figure 3. Illustration of the node topology representation for a BRep.

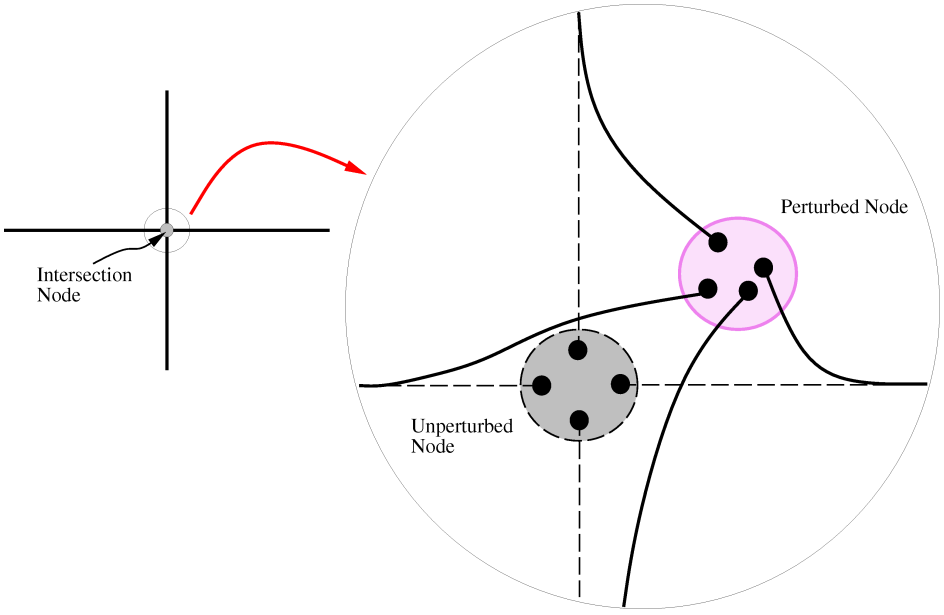


Figure 4. Node perturbation where no topology changes occur.

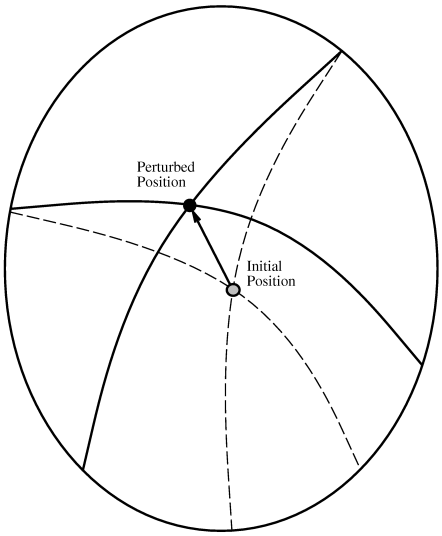


Figure 5. When perturbing intersecting surfaces, a single intersection node is maintained if the trim curve end-points move in tandem to remain within a tolerance sphere of each other.

5. These types of perturbation conserve the original topology as no *new* intersection nodes, edges, or faces are created.

If perturbations cause the new trim curve end-points to be enclosed by a sphere greater than the tolerance sphere defined in the CAD system, new intersection nodes and edges are created. This scenario is depicted in Figure 6. As a result of two end-points being perturbed more than a tolerance sphere away from the two remaining end-points, two new nodes and edges, shown in blue, are created to augment the initial topology (thus ensuring "watertightness"). This scenario also reflects another inherent difficulty associated with finite-differencing, where attempts to circumvent changes in the model topology (rescaling to accommodate differencing) are handled with an artificial design velocity at points in the discretization that have no dependence on the perturbed parameter. Such an approach leads to errors in the sensitivity calculation when the differenced model geometry instances do not share the same model topology.

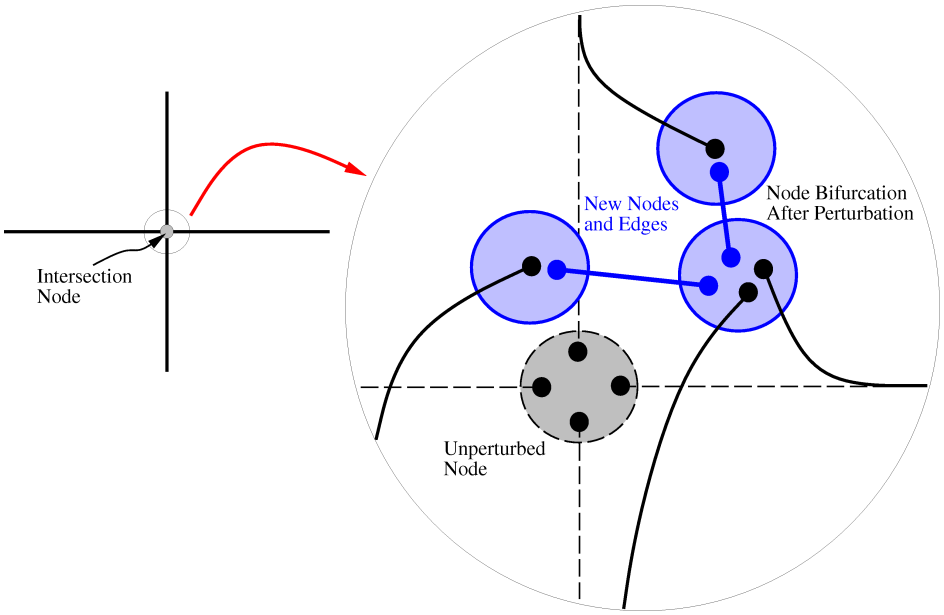


Figure 6. Node perturbation resulting in a topology changes.

Since the CAD system constructs surfaces, trim curves, and intersection nodes in a non-transparent manner to the user, it is generally unclear how the trim curve end-points are *actually* oriented prior to being

collapsed into a single intersection node. This inhibits the ability to track the end-points for sensitivity analysis. In order to conduct sensitivity analysis at intersection nodes, a simplifying assumption is that a node perturbation does not result in a node bifurcation and local topology is conserved as in Figures 4 and 5.

To determine the sensitivity of a node to driving parameters using the minimum velocity method, we begin by extending Eq. (12) for all combinations of intersecting faces. Collecting the intersecting face indices associated with node N_l in the set $\mathbf{F_{N_l}}$, we represent the number of faces (and therefore edges) incident to node N_l as n_l. Recalling that Eq. (12) is not exact, we must obtain, for each face $F_\eta \in \mathbf{F_{N_l}}$ ($\eta = 1, \ldots, n_l$), the point $\vec{\mathbf{r}}_\eta$ that minimizes $\| \vec{\mathbf{r}}_\eta - \vec{\mathbf{q}}_l \|$. These points are collected in $\mathbf{R_{N_l}}$. We also consider that the CAD geometry kernel determines the intersection of edges within a tolerance ball of radius ϵ_{tol} at the node as shown in Figure 3. The intersection of faces at the node also occurs with the same tolerance, thus we have for $\vec{\mathbf{r}}_\eta \in \mathbf{R_{N_l}}$

$$\left\{ \begin{array}{c} \vec{\mathbf{r}}_1 - \vec{\mathbf{r}}_2 = \vec{\epsilon}_{1,2} \\ \vdots \\ \vec{\mathbf{r}}_1 - \vec{\mathbf{r}}_{n_l-1} = \vec{\epsilon}_{1,n_l-1} \\ \vec{\mathbf{r}}_1 - \vec{\mathbf{r}}_{n_l} = \vec{\epsilon}_{1,n_l} \end{array} \right\}, \quad \left\{ \begin{array}{c} \vec{\mathbf{r}}_2 - \vec{\mathbf{r}}_3 = \vec{\epsilon}_{2,3} \\ \vdots \\ \vec{\mathbf{r}}_2 - \vec{\mathbf{r}}_{n_l-1} = \vec{\epsilon}_{2,n_l-1} \\ \vec{\mathbf{r}}_2 - \vec{\mathbf{r}}_{n_l} = \vec{\epsilon}_{2,n_l} \end{array} \right\}, \quad \ldots \tag{24}$$

$$\left\{ \begin{array}{c} \vec{\mathbf{r}}_{n_l-2} - \vec{\mathbf{r}}_{n_l-1} = \vec{\epsilon}_{n_l-2,n_l-1} \\ \vec{\mathbf{r}}_{n_l-2} - \vec{\mathbf{r}}_{n_l} = \vec{\epsilon}_{n_l-2,n_l} \end{array} \right\}, \quad \left\{ \vec{\mathbf{r}}_{n_l-1} - \vec{\mathbf{r}}_{n_l} = \vec{\epsilon}_{n_l-1,n_l} \right\}$$

where each equation corresponds to the perspective of each face in $\mathbf{F_N}$ to all other intersecting faces. Again, these equations are similar to Eq. (12), but the number of equations is greater due to the fact that the node is generally incident to more than two faces. We also note, as in the edge case, the possibility for an offset between the intersecting faces at the node, where

$$\left\{ \begin{array}{c} \| \vec{\epsilon}_{1,2} \| \leq \epsilon_{tol} \\ \vdots \\ \| \vec{\epsilon}_{1,n_l-1} \| \leq \epsilon_{tol} \\ \| \vec{\epsilon}_{1,n_l} \| \leq \epsilon_{tol} \end{array} \right\}, \quad \left\{ \begin{array}{c} \| \vec{\epsilon}_{2,3} \| \leq \epsilon_{tol} \\ \vdots \\ \| \vec{\epsilon}_{2,n_l-1} \| \leq \epsilon_{tol} \\ \| \vec{\epsilon}_{2,n_l} \| \leq \epsilon_{tol} \end{array} \right\}, \quad \ldots$$

$$\left\{ \begin{array}{c} \| \vec{\epsilon}_{n_l-2,n_l-1} \| \leq \epsilon_{tol} \\ \| \vec{\epsilon}_{n_l-2,n_l} \| \leq \epsilon_{tol} \end{array} \right\}, \quad \left\{ \| \vec{\epsilon}_{n_l-1,n_l} \| \leq \epsilon_{tol} \right\}.$$

We can also add the perspective of each face to the node itself as

$$\left\{ \begin{array}{c} \vec{\mathbf{r}}_1 - \vec{\mathbf{q}}_l = \vec{\epsilon}_{N,1} \\ \vec{\mathbf{r}}_2 - \vec{\mathbf{q}}_l = \vec{\epsilon}_{N,2} \\ \vdots \\ \vec{\mathbf{r}}_{n_l} - \vec{\mathbf{q}}_l = \vec{\epsilon}_{N,n_l} \end{array} \right\}, \tag{25}$$

with similar offsets written as

$$\begin{array}{c} \| \vec{\epsilon}_{N,1} \| \leq \epsilon_{tol} \\ \| \vec{\epsilon}_{N,2} \| \leq \epsilon_{tol} \\ \vdots \\ \| \vec{\epsilon}_{N,n_l} \| \leq \epsilon_{tol}. \end{array}$$

At this point we recognize that the CAD geometry kernel provides no parameterization for $\vec{\mathbf{q}}_l$ and continue with a variational analysis similar to the edge case. We first write a Taylor expansion around $\vec{\mathbf{q}}_l$ and $\vec{\mathbf{r}}_\eta$ for

$\eta = 1, \ldots, n_l$ as

$$\vec{\mathbf{q}}'_l = \vec{\mathbf{q}}_l + \frac{d\vec{\mathbf{q}}_l}{dP_w}\Delta P_w + \frac{1}{2}\frac{d^2\vec{\mathbf{q}}_l}{dP_w^2}\Delta P_w^2 + \cdots \tag{26}$$

$$\vec{\mathbf{r}}_\eta(u_\eta, v_\eta; P_w + \Delta P_w)' = \vec{\mathbf{r}}_\eta(u_\eta, v_\eta; P_w) + \frac{d\vec{\mathbf{r}}_\eta}{dP_w}\Delta P_w + \frac{1}{2}\frac{d^2\vec{\mathbf{r}}_\eta}{dP_w^2}\Delta P_w^2 + \cdots \tag{27}$$

We also carry the assumption that the parameter perturbation, ΔP_w, is infinitesimally small to preserve topology after regeneration with the new parameter $P'_w = P_w + \Delta P_w$. The new model instance would also yield the offset vectors $\vec{\epsilon}'_{1,2}$, $\vec{\epsilon}'_{N,1}$, etc., such that $\| \vec{\epsilon}'_{1,2} - \vec{\epsilon}_{1,2} \| \leq \epsilon_{tol}$, $\| \vec{\epsilon}'_{N,1} - \vec{\epsilon}_{N,1} \| \leq \epsilon_{tol}$, etc., as well. From an implementation view, we can then set $\vec{\epsilon}'_{1,2} = \vec{\epsilon}_{1,2}$, $\vec{\epsilon}'_{N,1} = \vec{\epsilon}_{N,1}$, etc. We take the first equation in Eq. (24) as representative of how to rewrite all face-to-face node constraint equations:

$$\begin{aligned}
\vec{\epsilon}'_{1,2} &= \vec{\mathbf{r}}_1(u_1, v_1; P_w + \Delta P_w)' - \vec{\mathbf{r}}_2(u_2, v_2; P_w + \Delta P_w)' \\
&= \underbrace{\vec{\mathbf{r}}_1(u_1, v_1; P_w) - \vec{\mathbf{r}}_2(u_2, v_2; P_w)}_{\vec{\epsilon}_{1,2}} + \left(\frac{d\vec{\mathbf{r}}_1}{dP_w} - \frac{d\vec{\mathbf{r}}_2}{dP_w}\right)\Delta P_w + \frac{1}{2}\left(\frac{d^2\vec{\mathbf{r}}_1}{dP_w^2} - \frac{d^2\vec{\mathbf{r}}_2}{dP_w^2}\right)\Delta P_w^2 + \cdots \\
\mathbf{0} &= \left(\frac{d\vec{\mathbf{r}}_1}{dP_w} - \frac{d\vec{\mathbf{r}}_2}{dP_w}\right)\Delta P_w + \frac{1}{2}\left(\frac{d^2\vec{\mathbf{r}}_1}{dP_w^2} - \frac{d^2\vec{\mathbf{r}}_2}{dP_w^2}\right)\Delta P_w^2 + \cdots .
\end{aligned} \tag{28}$$

We also look at the first equation in Eq. (25) as representative of how to rewrite the face-to-node constraint equations:

$$\begin{aligned}
\vec{\epsilon}'_{N,1} &= \vec{\mathbf{r}}_1(u_1, v_1; P_w + \Delta P_w)' - \vec{\mathbf{q}}'_l \\
&= \underbrace{\vec{\mathbf{r}}_1(u_1, v_1; P_w) - \vec{\mathbf{q}}_l}_{\vec{\epsilon}_{N,1}} + \left(\frac{d\vec{\mathbf{r}}_1}{dP_w} - \frac{d\vec{\mathbf{q}}_l}{dP_w}\right)\Delta P_w + \frac{1}{2}\left(\frac{d^2\vec{\mathbf{r}}_1}{dP_w^2} - \frac{d^2\vec{\mathbf{q}}_l}{dP_w^2}\right)\Delta P_w^2 + \cdots \\
\mathbf{0} &= \left(\frac{d\vec{\mathbf{r}}_1}{dP_w} - \frac{d\vec{\mathbf{q}}_l}{dP_w}\right)\Delta P_w + \frac{1}{2}\left(\frac{d^2\vec{\mathbf{r}}_1}{dP_w^2} - \frac{d^2\vec{\mathbf{q}}_l}{dP_w^2}\right)\Delta P_w^2 + \cdots .
\end{aligned} \tag{29}$$

In order for Eqs. (28) and (29) to hold to first-order in ΔP_w, the first term in parenthesis must equal 0, which allows us to write the following for each node constraint equation:

$$\left\{ \begin{array}{c} \frac{d\vec{\mathbf{r}}_1}{dP_w} - \frac{d\vec{\mathbf{r}}_2}{dP_w} = \mathbf{0} \\ \vdots \\ \frac{d\vec{\mathbf{r}}_1}{dP_w} - \frac{d\vec{\mathbf{r}}_{n_l-1}}{dP_w} = \mathbf{0} \\ \frac{d\vec{\mathbf{r}}_1}{dP_w} - \frac{d\vec{\mathbf{r}}_{n_l}}{dP_w} = \mathbf{0} \end{array} \right\}, \quad \left\{ \begin{array}{c} \frac{d\vec{\mathbf{r}}_2}{dP_w} - \frac{d\vec{\mathbf{r}}_3}{dP_w} = \mathbf{0} \\ \vdots \\ \frac{d\vec{\mathbf{r}}_2}{dP_w} - \frac{d\vec{\mathbf{r}}_{n_l-1}}{dP_w} = \mathbf{0} \\ \frac{d\vec{\mathbf{r}}_2}{dP_w} - \frac{d\vec{\mathbf{r}}_{n_l}}{dP_w} = \mathbf{0} \end{array} \right\}, \quad \cdots$$

$$\left\{ \begin{array}{c} \frac{d\vec{\mathbf{r}}_{n_l-2}}{dP_w} - \frac{d\vec{\mathbf{r}}_{n_l-1}}{dP_w} = \mathbf{0} \\ \frac{d\vec{\mathbf{r}}_{n_l-2}}{dP_w} - \frac{d\vec{\mathbf{r}}_{n_l}}{dP_w} = \mathbf{0} \end{array} \right\}, \quad \left\{ \frac{d\vec{\mathbf{r}}_{n_l-1}}{dP_w} - \frac{d\vec{\mathbf{r}}_{n_l}}{dP_w} = \mathbf{0} \right\} \tag{30}$$

and

$$\left\{ \begin{array}{c} \frac{d\vec{\mathbf{r}}_1}{dP_w} - \frac{d\vec{\mathbf{q}}_l}{dP_w} = \mathbf{0} \\ \frac{d\vec{\mathbf{r}}_2}{dP_w} - \frac{d\vec{\mathbf{q}}_l}{dP_w} = \mathbf{0} \\ \vdots \\ \frac{d\vec{\mathbf{r}}_{n_l}}{dP_w} - \frac{d\vec{\mathbf{q}}_l}{dP_w} = \mathbf{0} \end{array} \right\}. \tag{31}$$

We recall further that each $\frac{d\vec{\mathbf{r}}_\eta}{dP_w}$ can be expanded to

$$\frac{d\vec{\mathbf{r}}_\eta}{dP_w} = \left(\frac{\partial \vec{\mathbf{r}}_\eta}{\partial u_\eta}\frac{\partial u_\eta}{\partial P_w} + \frac{\partial \vec{\mathbf{r}}_\eta}{\partial v_\eta}\frac{\partial v_\eta}{\partial P_w} + \frac{\partial \vec{\mathbf{r}}_\eta}{\partial P_w} \right) = \left(\frac{\partial \vec{\mathbf{r}}_\eta}{\partial P_w} + \nabla \vec{\mathbf{r}}_\eta \cdot \nu_\eta \right).$$

Without needing to augment the equation set further to determine $\frac{d\vec{\mathbf{r}}_\eta}{dP_w}$, we can then write a linear system as

$$\underbrace{\left[\begin{array}{cccccc}
\nabla\vec{\mathbf{r}}_1 & -\nabla\vec{\mathbf{r}}_2 & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\
\nabla\vec{\mathbf{r}}_1 & \mathbf{0} & -\nabla\vec{\mathbf{r}}_3 & \mathbf{0} & \cdots & \mathbf{0} \\
\vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\
\nabla\vec{\mathbf{r}}_1 & \mathbf{0} & \cdots & \cdots & \nabla\vec{\mathbf{r}}_{n_l} & \mathbf{0} \\
\hline
\mathbf{0} & \nabla\vec{\mathbf{r}}_2 & -\nabla\vec{\mathbf{r}}_3 & \mathbf{0} & \cdots & \mathbf{0} \\
\mathbf{0} & \nabla\vec{\mathbf{r}}_2 & \mathbf{0} & -\nabla\vec{\mathbf{r}}_4 & \cdots & \mathbf{0} \\
\vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\
\mathbf{0} & \nabla\vec{\mathbf{r}}_2 & \mathbf{0} & \cdots & \nabla\vec{\mathbf{r}}_{n_l} & \mathbf{0} \\
\hline
\vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\
\hline
\mathbf{0} & \cdots & \mathbf{0} & \nabla\vec{\mathbf{r}}_{n_l-1} & -\nabla\vec{\mathbf{r}}_{n_l} & \mathbf{0} \\
\hline
-\nabla\vec{\mathbf{r}}_1 & \mathbf{0} & \cdots & \cdots & \mathbf{0} & \mathbf{I} \\
\mathbf{0} & -\nabla\vec{\mathbf{r}}_2 & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{I} \\
\vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\
\mathbf{0} & \cdots & \cdots & \mathbf{0} & -\nabla\vec{\mathbf{r}}_{n_l} & \mathbf{I}
\end{array}\right]}_{\mathbb{A}_N}
\underbrace{\left[\begin{array}{c}
\nu_1 \\ \nu_2 \\ \nu_3 \\ \vdots \\ \nu_{n_l} \\ \frac{d\vec{\mathbf{q}}_l}{dP_w}
\end{array}\right]}_{\mathbb{X}_N}
=
\underbrace{\left[\begin{array}{c}
\frac{\partial\vec{\mathbf{r}}_2}{\partial P_w} - \frac{\partial\vec{\mathbf{r}}_1}{\partial P_w} \\
\frac{\partial\vec{\mathbf{r}}_3}{\partial P_w} - \frac{\partial\vec{\mathbf{r}}_1}{\partial P_w} \\
\vdots \\
\frac{\partial\vec{\mathbf{r}}_{n_l}}{\partial P_w} - \frac{\partial\vec{\mathbf{r}}_j}{\partial P_w} \\
\hline
\frac{\partial\vec{\mathbf{r}}_3}{\partial P_w} - \frac{\partial\vec{\mathbf{r}}_2}{\partial P_w} \\
\frac{\partial\vec{\mathbf{r}}_4}{\partial P_w} - \frac{\partial\vec{\mathbf{r}}_2}{\partial P_w} \\
\vdots \\
\frac{\partial\vec{\mathbf{r}}_{n_l}}{\partial P_w} - \frac{\partial\vec{\mathbf{r}}_2}{\partial P_w} \\
\hline
\vdots \\
\hline
\frac{\partial\vec{\mathbf{r}}_{n_l}}{\partial P_w} - \frac{\partial\vec{\mathbf{r}}_{n_l-1}}{\partial P_w} \\
\hline
\frac{\partial\vec{\mathbf{r}}_1}{\partial P_w} \\
\frac{\partial\vec{\mathbf{r}}_2}{\partial P_w} \\
\vdots \\
\frac{\partial\vec{\mathbf{r}}_{n_l}}{\partial P_w}
\end{array}\right]}_{\mathbb{B}_N}. \tag{32}$$

Since $\mathbb{A}_N\mathbb{X}_N = \mathbb{B}_N$ is an over-determined system, it is also solved in a least-squares sense using the truncated SVD approach. It is likely that $\mathbb{A}_N$ may be rank deficient as well; in these situations, the only unique solution we can use is $\mathbb{X}_F^*$, which minimizes $\| \mathbb{A}_N\mathbb{X}_N^* - \mathbb{B}_N \|$ while having a minimum value for $\| \mathbb{X}_N^* \|$. As in the edge case, the geometric perspective for this solution implies a minimum relative velocity embodied in $\frac{d\vec{\mathbf{r}}_\eta}{dP_w}$ for the design motion of the node.

V.G. Simplification of the Minimum Velocity Method for Edges

Having developed the minimum velocity method for nodes, we can simplify the edge formulation such that a single algorithm can be used for both edges and nodes. We utilize the first equation in Eq. (12) for the edge sensitivity calculation and generate a simplified form of Eq. (22) as

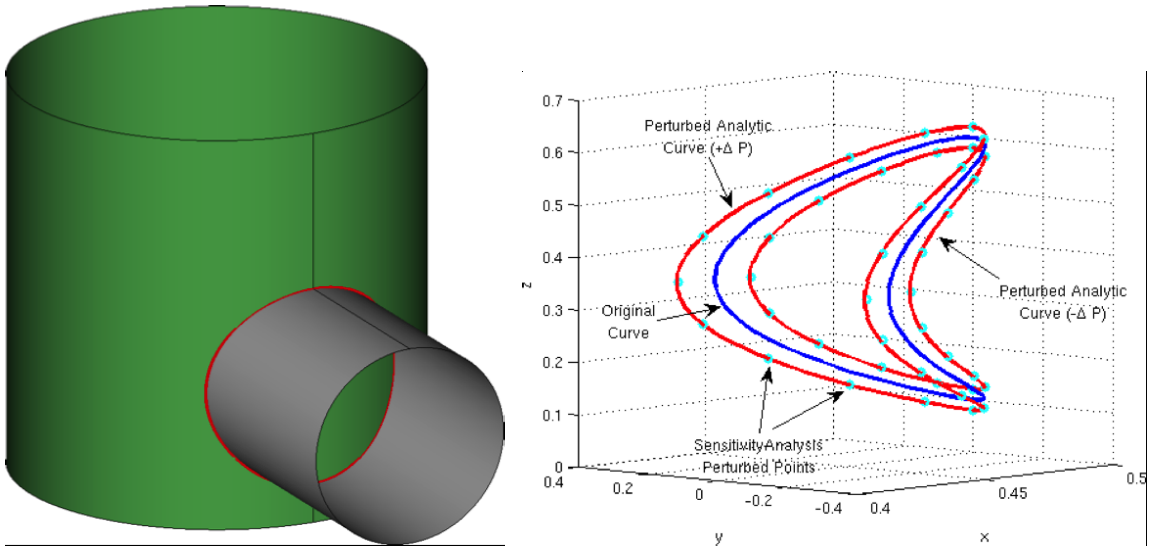
$$\left[\begin{array}{ccc}
\nabla\vec{\mathbf{r}}_\alpha & -\nabla\vec{\mathbf{r}}_\beta & \mathbf{0} \\
-\nabla\vec{\mathbf{r}}_\alpha & \mathbf{0} & \mathbf{I} \\
\mathbf{0} & -\nabla\vec{\mathbf{r}}_\beta & \mathbf{I}
\end{array}\right]
\left[\begin{array}{c}
\nu_\alpha \\ \nu_\beta \\ \vec{\psi}
\end{array}\right]
=
\left[\begin{array}{c}
\frac{\partial\vec{\mathbf{r}}_\beta}{\partial P_w} - \frac{\partial\vec{\mathbf{r}}_\alpha}{\partial P_w} \\
\frac{\partial\vec{\mathbf{r}}_\alpha}{\partial P_w} \\
\frac{\partial\vec{\mathbf{r}}_\beta}{\partial P_w}
\end{array}\right]. \tag{33}$$

This approach simplifies the implementation because Eq. (33) is generated by applying the same algorithm for the node system in Eq. (32) to a two-face "node" (i.e. a location along the edge is considered a "node" incident to two faces). A single algorithm then serves to calculate the sensitivity of edges and nodes, albeit the solution to Eq. (33) will not exactly match the solution to Eq. (22). The difference in the minimum velocity solutions is seen in the slightly different null space for each system; the solution from Eq. (33) will likely have a smaller magnitude than that obtained from Eq. (22) with the tradeoff that less geometry

information is taken into account. The subsequent examples utilize this simplified implementation for edge sensitivities.

VI. Examples

Consider first the model of two intersecting cylinders shown in Figure 7(a) constructed using the SolidWorks CAD system. The model is parameterized such that the diameter of the small cylinder can be changed. Modification of this parameter (denoted P) does not alter the surface of the larger cylinder as there is no such dependency in the feature-tree. However, the intersection curve of the two cylinders (shown in red in Figure 7(a)) is indirectly dependent on the small cylinder diameter. The response of the intersection curve to a change in the small cylinder diameter is shown in 7(b). The baseline intersection curve is rendered in blue. As the small cylinder diameter is perturbed by $\pm\Delta P$, the new intersection curves generated by master-model regeneration are shown in red. Section V.E, with the exception that Eq. (33) is substituted for Eq. (22), is used to compute the edge sensitivity derivatives at several discrete points along the original intersection. First order approximations to the perturbed points are then computed using the sensitivity derivatives, $\pm\Delta P$, and the corresponding points of the original curve. These points are plotted as the cyan points in the figure. Their approximation is in good agreement with the perturbed analytic curve.



(a) (b)

Figure 7. Union of two Cylinders (a) geometry with intersection edge in red, (b) Edge perturbation comparison.

Next we consider the model of the small cam shown in Figure 8. This model was constructed using Pro/ENGINEER and is parameterized to allow specification of the camshaft diameter. Modification of the camshaft diameter will result in a change in the diameter of the hole through the center of the cam. The model is oriented such that the longitudinal axis of the camshaft, and therefore the center hole of the cam, is parallel to the Z axis. The center hole is extruded along its axis with a fixed thickness such that a pin, parallel to the Y axis, can be added to secure the cam to the camshaft. An additional parameter defines the diameter of the pin. The cam hole is keyed to further secure the cam to the camshaft.

The methods described in Section V are applied to the cam model. When the parameter which defines the diameter of the camshaft is modified, the model association forces a regeneration of the cylindrical surface defining the hole through the center of the cam. As a result, there is a non-zero sensitivity of the hole surface to the camshaft diameter. This sensitivity is computed analytically and shown in Figure 9. Note that portions of the cam are translucently rendered, thus exposing the affected surfaces of the cam center hole. Figure 9(a) shows the sensitivity contours of the X coordinate to camshaft diameter while Figure 9(b) shows the same sensitivity of the Y coordinate. Symmetry about the XZ and YZ planes respectively is apparent and expected due to the model orientation. The sensitivity of the planar sides of the key slot, the pin hole, and the sides of the cam is identically zero since these surfaces have no dependency on the camshaft diameter. Therefore color contours on these surface are not shown.

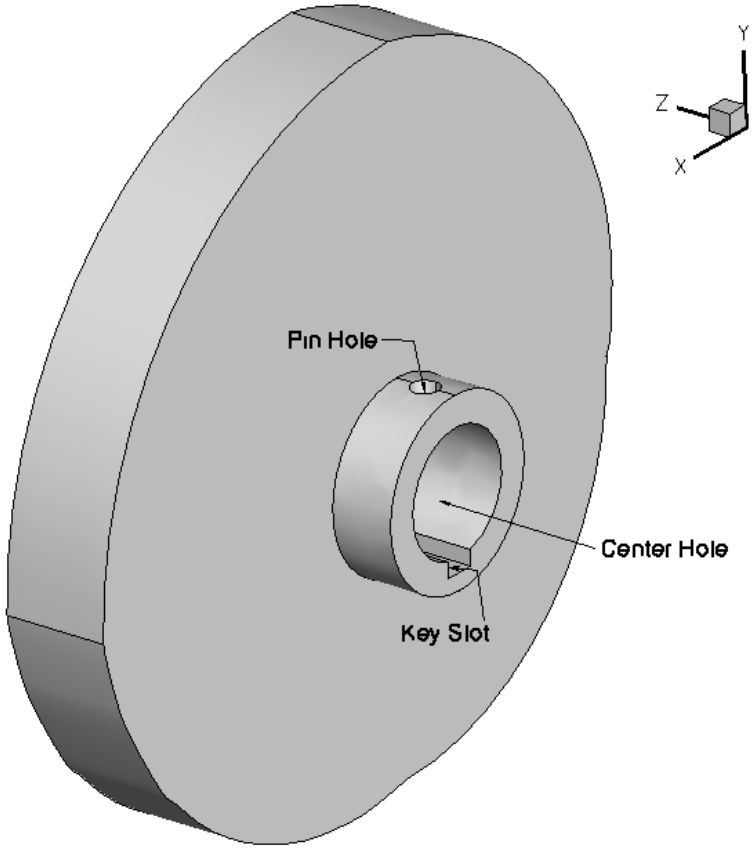
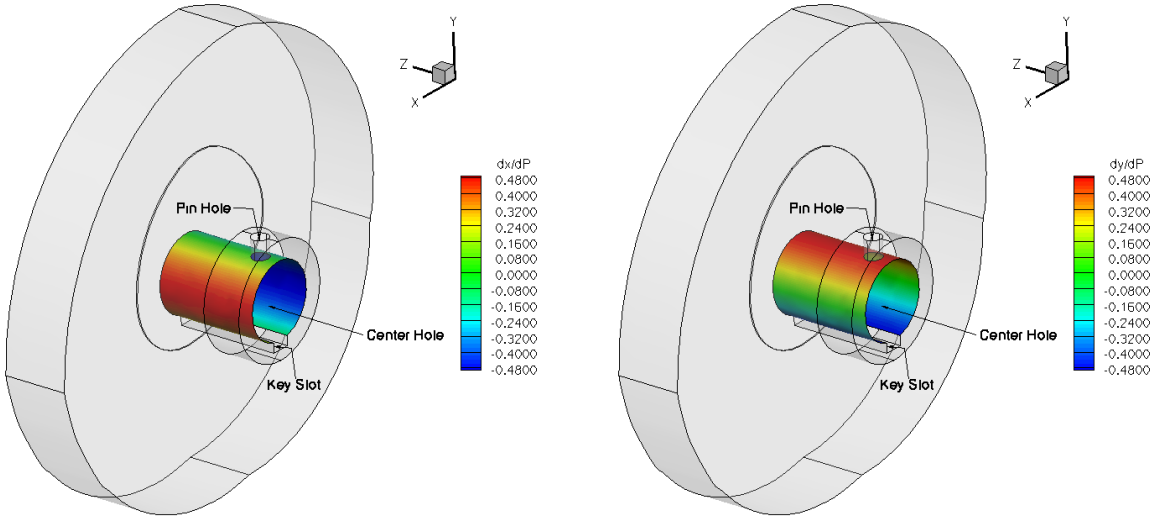


Figure 8. Simple cam part with parameterized shaft diameter.



(a) (b)

Figure 9. Sensitivity of camshaft cutout (a) $\frac{\partial x}{\partial P}$, (b) $\frac{\partial y}{\partial P}$.

In order to assess the sensitivity of edges and nodes, we also examine a change to the pin hole diameter. The topology of the pin hole is show in Figure 10 as a reference for the location of the edges and nodes under examination. The methods of Sections V.E, V.F, and V.G are applied to generate the respective sensitivity derivatives. The computed sensitivity derivatives are tabulated in Tables 1 and 2 along with their finite-difference (FD) counterparts. The finite-difference values result from a perturbation of 0.0001 to the nominal pin hole diameter of 0.338. Perturbations of 0.001 and 0.00001 showed no appreciable change in the FD values thus further indicating difficulty in the choice of step size.

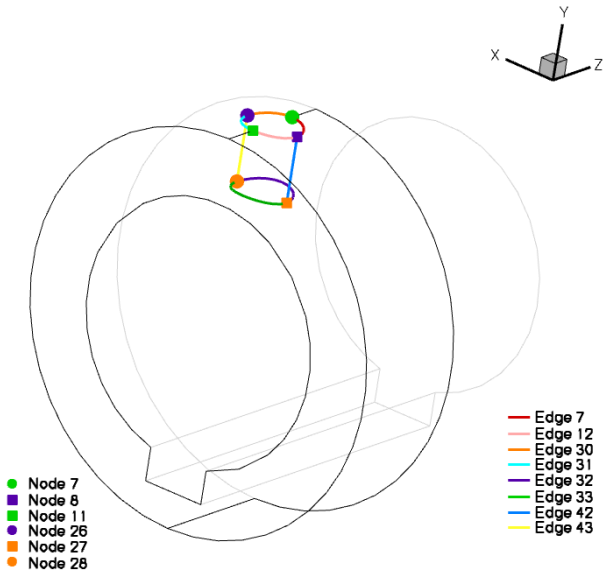


Figure 10. Cam model topology showing Edges and Nodes.

The significant differences are underlined in the tables. Note that when derivatives are $\mathcal{O}(\epsilon_0)$ they are considered equal and not highlighted. Edge sensitivity derivatives are evaluated at their midpoint for brevity. Edges 7, 12, 30, 31, 32, and 33 define the intersection of the pin hole with the outer and inner cylindrical surfaces of the center hole extrusion. Edges 42 and 43 connect the inner and outer intersections along the pin hole length.

Table 1. Validation of edge sensitivity derivatives evaluated at $t = 0.5 * (t_{\min} + t_{\max})$ for the Pro/ENGINEER Cam Example

	Analytic			Finite Difference*		
Edge	$\frac{\partial x}{\partial d}$	$\frac{\partial y}{\partial d}$	$\frac{\partial z}{\partial d}$	$\frac{\partial x}{\partial d}$	$\frac{\partial y}{\partial d}$	$\frac{\partial z}{\partial d}$
7	-3.482710e-01	-3.655012e-02	3.587600e-01	-3.483801e-01	-3.656156e-02	3.586547e-01
12	-3.482757e-01	-3.655111e-02	-3.587555e-01	-3.486479e-01	-3.659017e-02	-3.583959e-01
30	3.482978e-01	-3.655579e-02	3.587340e-01	3.486181e-01	-3.658941e-02	3.584245e-01
31	3.483032e-01	-3.655693e-02	-3.587288e-01	3.483408e-01	-3.656087e-02	-3.586925e-01
32	-1.658267e-05	-1.265307e-10	5.000000e-01	5.069418e-04	3.868017e-09	5.000000e-01
33	1.658267e-05	-1.265308e-10	-5.000000e-01	-5.069418e-04	3.868017e-09	-5.000000e-01
42	-5.000000e-01	0.000000e+00	6.123032e-17	-5.000000e-01	-9.580699e-02	0.000000e+00
43	5.000000e-01	0.000000e+00	-6.123032e-17	5.000000e-01	-9.580699e-02	0.000000e+00

* Underlined digits indicated significant differences.

The analytic sensitivity derivatives in Table 1 exhibit the expected symmetry of the model construction. For example, edges 7 and 12 are symmetric about the XY plane. These edges are defined by the intersection of the pin hole and a cylindrical surface aligned with the Z axis. Therefore, as the pin hole diameter is

increased, their midpoints will move along a design motion vector with a component in the $-X$ direction. As the midpoints will also remain on the Z aligned cylinder, an increase in the pin hole diameter will induce a $-Y$ component to the motion vector. However, due to the XY symmetry of the edges and orientation of the constituent geometry, the motion vector for the midpoint of edge 7 will have a $+Z$ component whereas that of the midpoint of edge 12 will have a $-Z$ component as the pin hole diameter increases. Therefore, the sensitivity derivatives at the midpoint of edges 7 and 12 with respect to a change in pin hole diameter exhibit the same symmetry about the XY plane as do the edge shapes themselves. Similar observations can be made for other edge pairs.

The analytic sensitivities in Table 1 show the expected symmetry to more significant digits than do the respective FD approximations. The argument could be made that the analytic values are more accurate because an arbitrary step-size was selected for the FD calculations. As stated above, it is commonly known that an appropriate step-size must be selected for each parameter; in this case, we point out that although the selected step-size may be considered "small" from the standpoint of a designer accustomed to the order-of-magnitude of values generally considered for the parameter, such conclusions will often be insufficient for a finite-difference calculation. Truncation error or lack of precision will lead to ambiguity in the FD result because we generally do not know what the "true" design velocity should be. For example, edges 42 and 43 have FD results that do not agree in the Y-component with the analytic sensitivities; in this case the point should have a design velocity with a zero-valued Y-component, as seen in the analytic result, because the model design implies a design motion normal to the YZ plane. It may be possible to obtain greater precision agreement between the two methods by determining an appropriate step-size; however, the results will not match exactly because the FD method implies evaluation of the perturbed points at the same t value of the baseline edge, whereas the minimal velocity method allows for relative design motion to different t values on the perturbed edge.

Table 2. Validation of node sensitivity derivatives for the pin surfaces of the Pro/ENGINEER Cam Example

	Analytic			Finite Difference		
Node	$\frac{\partial x}{\partial d}$	$\frac{\partial y}{\partial d}$	$\frac{\partial z}{\partial d}$	$\frac{\partial x}{\partial d}$	$\frac{\partial y}{\partial d}$	$\frac{\partial z}{\partial d}$
7	-6.727954e-16	-2.700634e-16	5.000000e-01	0.000000e+00	0.000000e+00	5.000000e-01
8	-5.000000e-01	-7.597324e-02	6.260072e-17	-5.000000e-01	-7.597324e-02	-3.330669e-12
11	-1.264327e-14	-2.594608e-17	-5.000000e-01	-2.465190e-28	0.000000e+00	-5.000000e-01
26	5.000000e-01	-7.597324e-02	-5.862222e-17	5.000000e-01	-7.597324e-02	0.000000e+00
27	-5.000000e-01	-1.156407e-01	6.583285e-17	-5.000000e-01	-1.156407e-01	-5.551115e-12
28	5.000000e-01	-1.156407e-01	-5.543323e-17	5.000000e-01	-1.156407e-01	-2.220446e-12

When comparing the analytic versus FD sensitivity derivatives of the nodes of the pin hole feature, Table 2 shows excellent agreement. In fact, the respective sensitivities agree to seven significant digits for all values which are not $\mathcal{O}(\epsilon_0)$ thus adding to the confidence in the methods of Section V.F.

VII. Conclusion

A method for computing analytic geometric sensitivity derivatives by tracing the evolution of features which constitute a BRep CAD model was presented. The method presented offers the construction of sensitivity derivatives that are directly tied to the master-model parameters of the subject CAD model and as such reflect the original design intent. The method presented was analytic and therefore avoids problems associated with finite-difference methods. No access to the source code of the underlying modeler was required. As no modification or differencing of the modeler source was needed, the problem of sensitivity derivative computation remains tractable. Sensitivity derivatives were calculated for all components of the model topology allowing for the indirect associativity of edges and nodes which may bound multiple geometric entities. The sensitivity of edges and nodes was computed via a minimum velocity approach that does not require any assumed directionality of perturbation. Examples were provided to demonstrate and help validate the method. Quantification of the minimum velocity method for select edges and nodes of a cam model was provided through comparison to finite-difference computations of the respective sensitivity

derivatives.

Acknowledgments

The authors would like to acknowledge the Supersonics Project of the NASA Fundamental Aeronautics Program for supporting this work and Mark Drela for providing additional perspectives in discussions on the geometry sensitivity problem.

References

[1]Nielsen, E. J., Diskin, B., and Yamaleev, N. K., "Discrete Adjoint-Based Design Optimization of Unsteady Turbulent Flows on Dynamic Unstructured Grids," *AIAA Journal*, Vol. 48, No. 6, June 2010, pp. 1195–1206.

[2]Bischof, C., Carle, A., Khademi, P., and Mauer, A., "ADIFOR 2.0: Automatic Differentiation of Fortran 77 Programs," *IEEE Computational Science and Engineering*, Vol. 3, No. 3, 1996, pp. 18–32.

[3]Bischof, C., Roh, L., and Mauer, A., "ADIC–an Extensible Automatic Differentiation Tool for ANSI-C," *Software–Practice and Experience*, Vol. 27, No. 12, 1997, pp. 1427–1456.

[4]Bischof, C. H., Jones, W. T., Mauer, A., and Samareh-Abolhassani, J., "Experiences with the Application of the ADIC Automatic Differentiation tool to the CSCMDO 3-D Volume Grid Generation Code," *In Proceedings of the 34th AIAA Aerospace Sciences Meeting, American Institute of Aeronautics and Astronomics*, 1996, AIAA 1996-0716.

[5]Gumbert, C. R., Hou, G. J. W., and Newman, P. A., "Simultaneous Aerodynamic Analysis and Design Optimization (SAADO) for a 3-D Flexible Wing," *In Proceedings of the 39th AIAA Aerospace Sciences Meeting, American Institute of Aeronautics and Astronomics*, 2001, AIAA 2001-1107.

[6]Smith, R. E., Bloor, M. I. G., Wilson, M. J., and Thomas, A. M., "Rapid Airplane Parametric Input Design (RAPID)," *In Proceedings of the 12th AIAA Computational Fluid Dynamics Conference, San Diego, American Institute of Aeronautics and Astronautics*, 1995, AIAA 1995-1687.

[7]Anderson, W. K., Newman, J. C., Whitfield, D. L., and Nielsen, E. J., "Sensitivity Analysis for Navier-Stokes Equations on Unstructured Meshes Using Complex Variables," *AIAA Journal*, Vol. 49, No. 1, January 2001, pp. 56–63, AIAA Journal 0001-1452.

[8]"OpenCASCADE," http://www.opencascade.org, July 2010, OPEN CASCADE, S.A.S.

[9]Samareh, J. A., "A Novel Shape Parameterization Approach," NASA TM-1999-209116, May 1999.

[10]Samareh, J. A., "Aerodynamic Shape Optimization Based On Free-Form Deformation," *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Albany, New York, Aug 2004, AIAA 2004-4630.

[11]Nielsen, E. J. and Park, M. A., "Using an Adjoint Approach to Eliminate Mesh Sensitivities in Computational Design," *AIAA Journal*, Vol. 44, No. 5, May 2006, pp. 948–953.

[12]Nielsen, E. J., Lee-Rausch, E. M., and Jones, W. T., "Adjoint-Based Design of Rotors in a Noninertial Reference Frame," *Journal of Aircraft*, Vol. 47, No. 2, March-April 2010, pp. 638–646, Journal of Aircraft 0021-8669.

[13]Alonso, J., Martins, J., Reuther, J., Haimes, R., and Crawford, C., "High-Fidelity Aero-Structural Design Using a Parametric CAD-Based Model," 16th AIAA Computational Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics, Orlando, Florida, June 2003, AIAA 2003-3429.

[14]Choi, S., Alonso, J. J., Kroo, I. M., and Wintzer, M., "Multi-Fidelity Design Optimization of Low-Boom Supersonic Business Jets," AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, American Institute of Aeronautics and Astronautics, Albany, New York, 30 August - 1 September 2004, AIAA 2004-4371.

[15]Nemec, M., Aftosmis, M. J., and Pulliam, T. H., "CAD-Based Aerodynamic Design of Complex Configurations Using a Cartesian Method," 42nd AIAA Aerospace Sciences Meeting and Exhibit, American Institute of Aeronautics and Astronautics, Reno, Nevada, 5-8 January 2004, AIAA 2004-113.

[16]Haimes, R. and Follen, G., "Computational Analysis PRogramming Interface," Proceedings of the 6th International Conference on Numerical Grid Generation in Computational Field Simulations, July 1998.

[17]Crawford, C. A. and Haimes, R., "Synthesizing an MDO Architecture in CAD," 42nd AIAA Aerospace Sciences Meeting and Exhibit, American Institute of Aeronautics and Astronautics, Reno, Nevada, 5-8 January 2004, AIAA 2004-281.

[18]Haimes, R. and Merchant, A., "Direct CAD Access for Analysis and Design," Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems, EUROGEN 2005, FLM, Munich, 2005.

[19]Aftosmis, M. J., Delanaye, M., and Haimes, R., "Automatic Generation of CFD-Ready Surface Triangulations from CAD Geometry," 37th AIAA Aerospace Sciences Meeting and Exhibit, American Institute of Aeronautics and Astronautics, Reno, Nevada, 11-14 January 1999, AIAA 99-0776.

[20]Haimes, R. and Aftomis, M. J., "On Generating High Quality Watertight Triangulations Directly from CAD," *Proceedings of the 8th International Conference on Numerical Grid Generation in Computational Field Simulations*, University of Greenwich, United Kingdom, July 2002.