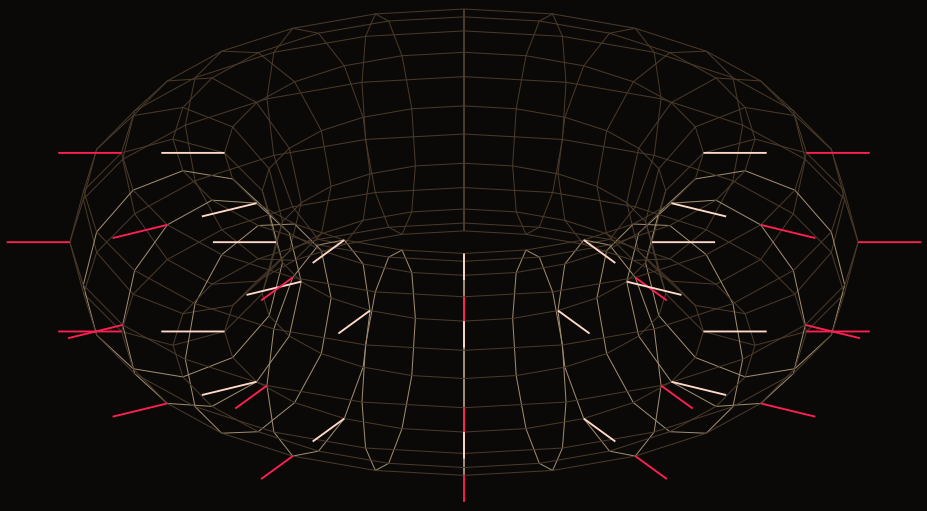


TORUS R = 8, R = 3 · DESIGN VELOCITY OF THE MAJOR RADIUS ON ITS OWN MESH

RED: OUTWARD · PALE: INWARD

Exact Gradients Through a BRep Kernel

Differentiating CAD parameters through tessellation, fillets, mass properties and contact

CAM PEDERSEN · MUNICIPAL ROBOTICS

Abstract

Gradient-based design needs the derivative of an objective with respect to the parameters a designer edits, such as a hole radius or a fillet radius. In a boundary-representation (BRep) kernel those parameters pass through Boolean operations, filleting and tessellation, which make discrete choices. We freeze the tessellation's topology at the current parameters and differentiate its node positions: surface nodes by evaluating the kernel's analytic surfaces on dual numbers, vertices and trim nodes by implicit differentiation of the surface equations, with tangency rows for fillet blends. The result is exact for the frozen mesh, extends to directional second derivatives and to a transposed (reverse) pass, and feeds mass properties and a rigid-body adjoint with penalty contact. A change of topology is reported as an error.

We validate against finite differences of the frozen plan and of fresh rebuilds that share no tessellation with the seam, and state where the method falls short: the derivative of the frozen mesh differs from the continuum by the polygonization error (6.7% for a fillet at 16 segments per circle, 0.11% at 128), a plan stays valid for parameter steps of at least 5.6×10^{-4} mm, and the same-plan checks found two errors in the seam on filleted, drilled parts, which we describe and have fixed.

1 Introduction

A mechanical designer works in parameters: the depth of a pocket, the radius of a bore, the radius of a fillet. An optimizer that serves that designer needs $dJ/d\theta$, the derivative of an objective J (a mass, an inertia, the outcome of a simulation) with respect to those parameters. Between θ and anything measurable sit Boolean operations, fillet construction and tessellation, each of which makes discrete decisions: which faces survive a Boolean, in which order they are sewn, how many segments approximate a circle. Commercial kernels do not generally expose parameter derivatives to the user, so designers typically fall back on finite differences of full rebuilds, at two rebuilds per parameter.

This is not a new problem, and our approach is not new in outline. Engineering Sketch Pad computes **tessellation sensitivities** on a parametric BRep: the motion of every grid point with respect to every design parameter, on a tessellation whose topology is kept consistent between the base and perturbed models, with sensitivities found on Nodes, then Edges, then Faces[1,2]. Those sensitivities are analytic for most construction operations and fall back to finite differences for others, filleting among them[1]. Banović et al. differentiate the Open CASCADE kernel itself by operator overloading[3,4]; algorithmic differentiation of that kind reaches every operation the kernel performs, fillets included, and can be nested for second derivatives, at the cost of differentiating the kernel's combinatorial code and taping its execution.

We describe the same idea, which we call the **seam**, in vcad, our open-source Rust BRep kernel, and extend it in four directions. None is new to differentiation in general. What is new is having them in a frozen-tessellation method that leaves the kernel's code undifferentiated.

- **Analytic fillet sensitivities without AD or FD.** Where Engineering Sketch Pad falls back to finite differences for fillets, a fillet radius moves every blend's radius and axis at once, and its blends meet their support faces tangentially, which makes the implicit system at their corner vertices rank-deficient. We add tangency rows that complete it (Section 3.3), so one fillet parameter, moving 20 surfaces, is differentiated without finite differences.
- **Second order on the frozen plan.** Directional second derivatives of mass properties, with the node accelerations of cones and tori (Section 4).
- **A transposed pass.** Node velocities are linear in per-surface seed values; transposing those maps gives the derivative for every parameter from one pass (Section 5).
- **Rigid-body dynamics.** Mass-property and contact cotangents from a discrete trajectory adjoint pulled back through the seam (Section 7).

We validate the derivatives against central finite differences of the frozen plan and, wherever a rebuild keeps its topology, of rebuilds tessellated from scratch, which share nothing with the seam's code (Section 8); the exceptions are listed there. We describe the two cases where the checks found errors in the seam. A script regenerates every measured value in this paper from a fresh clone (Section 10).

1.1 Scope and limits

- **The derivative is that of the frozen mesh.** The seam differentiates the mesh the kernel produces at θ_0, with segment counts and triangulation held fixed. It differs from the continuum shape derivative by the polygonization error, which falls as N^{-2} in the number of segments: for a filleted cube at 16 segments per full circle (four per quarter-blend) the fillet-radius derivative is 6.7% larger in magnitude than the continuum one, and 0.11% at 128 (Figure 8).
- **A plan is valid only near where it was captured.** Re-evaluating a plan matches every node to its capture-time position within 10^{-3} mm. On the five-parameter box of Section 8.4, every parameter can move by at least 5.6×10^{-4} mm, the largest step of our logarithmic grid that kept correspondence; the next grid point, 10^{-3} mm, lost it. The optimizer therefore captures a fresh plan at every iterate.
- **Segment counts can change between iterates.** vcad's Boolean rims are sag-adaptive: their arc count depends on the radius. Sweeping the box's hole radius from 1.6 to 2.4 mm changes the node count 20 times, and each change moves the volume by up to 2×10^{-6} of its value. Across iterates J is therefore piecewise smooth with small jumps. The optimizer does not reject a step because its node count differs from the last iterate's, because each iterate's gradient is taken on its own plan; the jumps show up as noise in the line search.
- **No derivatives across topology changes.** When a perturbation would change the topology (a blind hole breaking through, a fillet consuming a face), the seam detects it with a topology signature and per-node correspondence checks and returns an error, and the optimizer shrinks the step (Figure 1).
- **Seedings synthesized from the build function are not exact.** They are central differences of surface fields, accurate to $O(h^2)$, and they cost two rebuilds per parameter (Section 6).
- **Contact is phyz's differentiable penalty model**, a per-vertex spring–damper against a ground plane without friction, not a GJK/EPA collision pipeline[5]. The trajectory adjoint supports single-degree-of-freedom joints, open-loop control and objectives on the final state only.
- **Everything runs on CPU in double precision.**

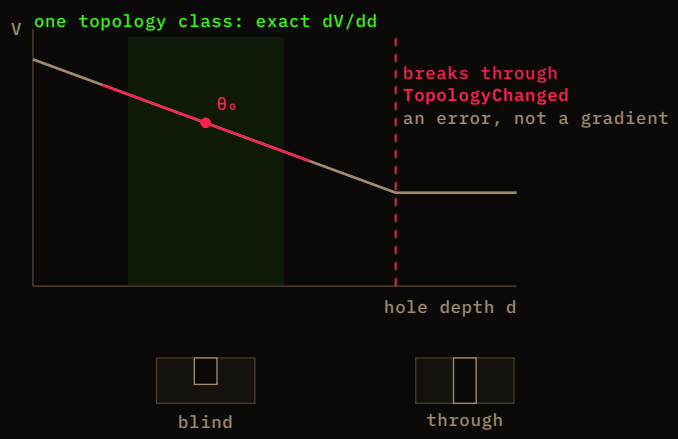


FIG 1 What the gradient is a gradient of. Within one topology class the frozen plan's volume is smooth in the parameter and the seam returns its derivative (red tangent). When a blind hole breaks through, the plan's topology signature changes and the seam returns an error; an optimizer treats that as a failed step and shrinks it. Schematic.

2 Background and related work

Parametric CAD sensitivities. Structural shape optimization has long used design velocities, the sensitivity of boundary points to design variables[6,7]. For CAD models these come from three routes. Finite differences of rebuilds with point correspondence are the most general and the most expensive; Robinson et al. and Agarwal et al. compute parametric design velocities this way for use with adjoint solvers[8,9]. Differentiating the kernel by operator overloading gives exact derivatives of the kernel's operations: Banović et al. and Mykhaskiv et al. differentiate Open CASCADE and couple it to a discrete adjoint CFD solver[3,4], and Xu et al. differentiate a NURBS-based design chain[10]. Engineering Sketch Pad differentiates its own constructive solid geometry process analytically where it can and uses finite differences elsewhere, and maps the result onto a consistent tessellation[1,2]. Our seam is closest to Engineering Sketch Pad: we leave the kernel's combinatorial code undifferentiated and differentiate the map from parameters to a tessellation of fixed topology. Within that frozen-tessellation approach we add analytic fillet sensitivities, second derivatives, a transposed pass, and the coupling to rigid-body dynamics with contact; operator-overloading AD of a kernel can supply fillet and second derivatives too, by differentiating the kernel's own code. Isogeometric analysis avoids the tessellation by computing on the CAD basis[11].

Differentiable CAD programs. Cascaval et al. write CAD programs in an automatically differentiable language for bidirectional editing, and do not optimize across changes of topology[12]. Michel and Boubekeur compute parameter updates for direct manipulation of procedural shapes[13]. Schulz et al. precompute a design space of CAD parameters for interactive optimization[14].

Differentiable rendering and other representations. Differentiable rasterizers[15,16] and path tracers[17] compute derivatives of images with respect to the vertices of a mesh of fixed connectivity, as ours is. Worchel and Alexa extract a mesh from parametric curves and surfaces for a differentiable mesh renderer and report

that naive differentiable tessellation is fragile[18]. Signed distance fields, learned[19] or analytic, are differentiable almost everywhere and can be rendered differentiably[20], and program synthesis recovers CSG trees from shapes[21,22]. These representations give up the BRep's exact faces, trims and fillets.

Differentiable physics and co-design. Differentiable simulators include rigid-body formulations with contact[23–25] and differentiable programming systems[26]. Robot co-design work optimizes morphology and control jointly[27,28], usually over parametric primitives rather than CAD models. Placing a solver or a simulation inside a differentiable program as one step with a known adjoint is standard[29]; our rollout coupling is such a step written by hand.

3 The differentiable seam

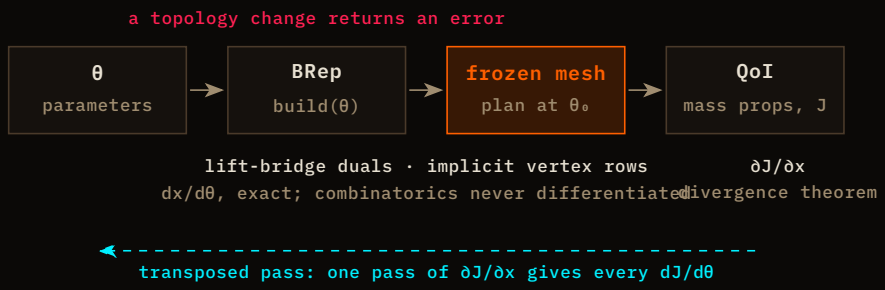


FIG 2 Where the chain is cut. The BRep build is never differentiated. At the current parameters a frozen plan records the tessellation; node velocities $dx/d\theta$ come from the kernel's analytic surfaces. Downstream quantities differentiate as smooth functions of node positions, and the transposed pass runs the same maps backwards.

3.1 Frozen plans

vcad represents solids as BReps whose faces carry analytic surfaces (plane, cylinder, cone, sphere, torus, bilinear patch, B-spline), with Boolean operations, filleting and a tessellator. Two properties of the kernel shaped the design. First, its Boolean and fillet code is order-nondeterministic: two builds of the same model can enumerate isomorphic topology in different orders, flip a blend cylinder's axis or rotate its reference direction, so traversal order cannot carry node identity from one build to another. Second, the stock mesh stores 32-bit positions, at which a central difference with $h = 10^{-6}$ is meaningless, so the seam emits its own 64-bit mesh.

Capturing a plan tessellates once at θ_0 and classifies every node:

- **Vertex nodes** coincide with a topology vertex. Their velocity comes from implicit differentiation (Section 3.3).
- **Surface nodes** are interior to a face, stored as fixed surface coordinates (u, v). Their position is $S(u, v; \varphi(\theta))$ for the surface's fields φ (offset, radius, axis, half-angle).
- **Trim nodes** lie on the intersection of two faces with non-parallel normals but are not topology vertices, for example a ring on a cylinder's cap. Their position is the Newton solution of $\{g_a(x) = 0, g_b(x) = 0, t \cdot (x - x_0) = 0\}$, with t the intersection tangent, so a trim node follows the moving trim whichever surface carries θ.

A plan carries a topology signature (a sorted multiset of per-face descriptors, invariant to isomorphic reordering) and each node's capture-time position. Re-evaluating at a nearby θ resolves vertices and faces geometrically, by nearest match after transporting the stored (u, v) through the rebuilt surface's frame. If any node lands more than 10^{-3} mm from where it should, evaluation fails with a lost-correspondence error; if the signature changes, it fails with a topology error.

- TopoVertex: implicit rows
- SurfaceUv: S(u, v; φ(θ)) on duals

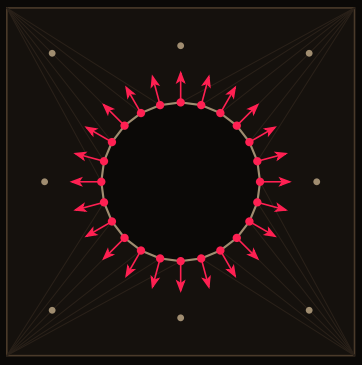
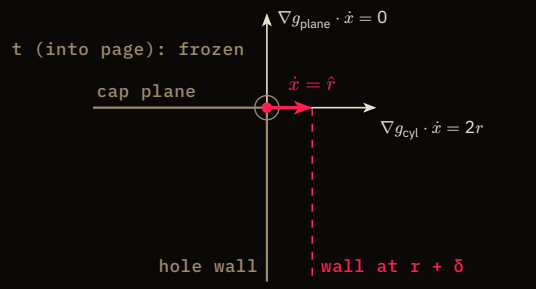


plate − cylinder(r), top cap
θ = r · needles are dx/dr



two rows + a frozen tangent: exactly radial

FIG 3 Node recipes on a Boolean result. Left: the top cap of a plate minus a cylinder, θ = hole radius. Rim vertices are topology vertices; their velocity comes from the implicit rows and is radial. Plane-interior samples lie on a surface that does not move and have zero velocity. Right: the rows at one rim vertex. The cap plane and the hole wall give two independent rows; the tangent direction is unconstrained and stays frozen, so the rim keeps its angular parameter.

3.2 Surface nodes

Every analytic surface in vcad is generic over a scalar type, so it can be evaluated on dual numbers $a + b\varepsilon$ with $\varepsilon^2 = 0$[30]. A parameter is connected to the surfaces it moves by a **seeding**: a map from surface index to field velocities (translate by v, grow the radius at rate $\dot{r}$, open the cone at rate $\dot{\alpha}$). Seeds on one surface add, which is what a fillet needs: when the fillet radius grows, each blend cylinder's radius grows at rate 1 while its axis retreats from the edge, and each corner sphere does the same in three coordinates. For a surface node, the dual part of $S(u, v; \varphi + \dot{\varphi}\varepsilon)$ is $dx/d\theta$.

3.3 Vertices and trims

A topology vertex lies on every surface incident to it. Each incident surface has an implicit form $g(x; \varphi) = 0$: for a plane $n \cdot (x - o)$, a cylinder $\|x_\perp\|^2 - r^2$, a sphere $\|x - c\|^2 - r^2$, and for a cone with apex a, axis d and half-angle α,

$$g = \|x_\perp\|^2 - \tan^2\alpha((x - a) \cdot d)^2.$$

Differentiating $g(x(\theta); \varphi(\theta)) = 0$ gives one linear row per surface,

$$\nabla g \cdot \dot{x} = -\frac{\partial g}{\partial \theta}.$$

We solve the rows by Gram–Schmidt, carrying the right-hand side. Independent rows fix components of $\dot{x}$, dependent rows must agree within tolerance (otherwise the solve fails), and directions no row constrains stay at zero, so a node on a circular rim

keeps its angular parameter. Three planes at a box corner give a full 3×3 solve; a plane and a cylinder at a hole rim give a rank-2 solve with a frozen tangent.

Two refinements were needed. Boolean sewing does not always pair the half-edges around a rim, so topological adjacency under-constrains some vertices, and the seam also collects rows by geometric incidence. And a fillet blend touches its support face tangentially, so the blend's row is parallel to the plane's and the system is rank-deficient exactly where the corner vertex slides along the face. The vertex must stay on the tangent curve; for a cylinder of axis a tangent to a plane of normal n we add the row $q \cdot \dot{x} = q \cdot \dot{v}_{\text{translate}}$ with $q = a \times n$. A sphere tangent to a plane has no axis; there the two rows use any two unit vectors q_1, q_2 orthogonal to n, which pins the tangent point's slide in both directions to the sphere's translation. With these rows the corner vertex of a filleted cube at (r, r, L) moves at $(1, 1, 0)$ per unit radius, as the closed form requires.

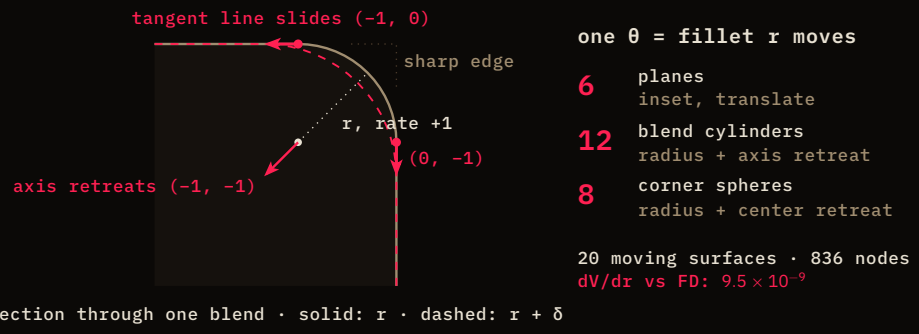


FIG 4 A fillet radius is a composite seed. A section through the blend on the edge at (L, L). Growing the radius by δ grows the blend cylinder's radius at rate 1 while its axis retreats at $(-1, -1)$; the two motions compose so that the tangent lines slide along the support faces at unit rate. (On the opposite edge, at the origin, every sign flips, which is the $(1, 1, 0)$ of the text.) On the filleted cube one parameter moves 20 surfaces.

4 Quantities of interest

4.1 Mass properties

Volume, mass, centroid and inertia tensor follow from signed-tetrahedron integrals over the closed mesh[31]. The integrals are written once over a generic scalar, so evaluating them at dual-number node positions gives every quantity and its derivative along θ in one pass. A downstream functional enters through the contraction

$$\frac{dJ}{d\theta} = \sum_i \frac{\partial J}{\partial x_i} \cdot \frac{dx_i}{d\theta},$$

with $\partial V / \partial x_i$ known in closed form from the divergence theorem.

4.2 Second order

For one direction in parameter space,

$$\frac{d^2V}{d\theta^2} = \sum_{ij} \frac{\partial^2 V}{\partial x_i \partial x_j} \dot{x}_i \dot{x}_j + \sum_i \frac{\partial V}{\partial x_i} \ddot{x}_i.$$

Evaluating the volume integral on nested duals $((x, \dot{x}), (\dot{x}, \ddot{x}))$ carries both terms; the work is in the node accelerations $\ddot{x}$. A surface node on a plane, cylinder or sphere is linear in its fields, so its acceleration is the first-order lift fed the field accelerations. Cone and torus points are not linear in their fields ($\tan\alpha$, the two torus radii), so their surface nodes are lifted on nested duals. Vertex and trim nodes solve the implicit system differentiated twice,

$$\nabla g \cdot \ddot{x} = -\left(\frac{\partial^2 g}{\partial \theta^2} + 2\dot{x}^\top \nabla_x \frac{\partial g}{\partial \theta} + \dot{x}^\top \nabla^2 g \, \dot{x} \right),$$

with the same row gradients as the first-order solve; cones and tori contribute closed-form curvature terms because their $\nabla^2 g$ is not constant. For least-squares objectives $J = \sum_q r_q^2$ we also form the Gauss–Newton Hessian $2J_r^\top J_r$, which drops the residual-curvature term and is exact at zero residual.

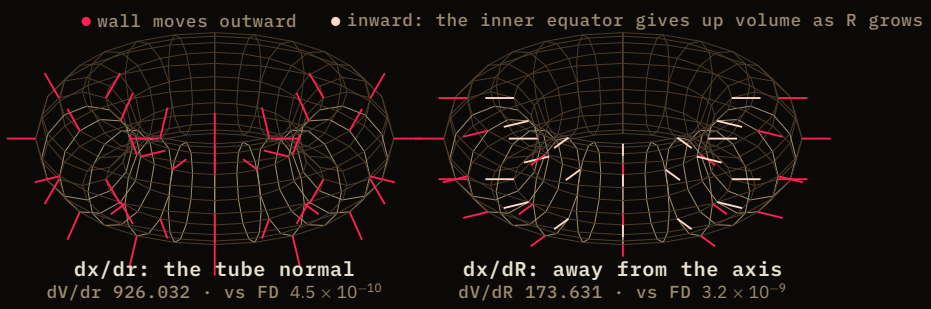


FIG 5 The torus's two design velocities on its own mesh, front half. Growing the minor radius moves every node along the tube normal. Growing the major radius moves every node away from the axis, so the wall moves outward on the outer equator and inward on the inner one, so the outward and inward motions largely cancel and dV/dR is the smaller derivative (continuum $2\pi^2 r^2$ against $4\pi^2 Rr$, a ratio $r/2R$). Values are the seam's, on the 24-segment mesh, about 2% below these continuum forms by the polygonization gap of Section 8.2 of a torus with $R = 8$, $r = 3$ mm.

5 The transposed pass and optimization

A seeding σ_k assigns to each surface s a vector of seed values: a translation in $\mathbb{R}^3$ and one rate per scalar field of that surface's kind. Each node's velocity is linear in the seed values of the surfaces that determine it: $\dot{x}_i = \sum_s A_{is}\sigma_k(s)$, where A_{is} is the Jacobian of node i's recipe with respect to surface s's fields at the frozen plan. The transposed pass contracts the transposes with $\partial J/\partial x$:

$$c_s = \sum_i A_{is}^\top \frac{\partial J}{\partial x_i}, \quad \frac{dJ}{d\theta_k} = \sum_s \langle c_s, \sigma_k(s) \rangle.$$

One pass computes every per-surface cotangent c_s; each parameter then costs a few dot products. No tape is recorded. The transposed vertex solve runs the same Gram–Schmidt elimination, carrying each coefficient as a linear functional of the rows, and each row's dependence on its seeds is read off by calling the forward row constructor with unit seeds. Forward and transposed passes therefore share the rows: their agreement checks the linear algebra, not the rows.

The optimizer is L-BFGS with memory 8[32], box projection and an Armijo line search on the projected step. Each iterate rebuilds the part and captures a fresh plan. A

trial step that fails with a topology or correspondence error counts as a failed step. A curvature pair is dropped when $s \cdot y \le 10^{-12}\|s\|\|y\|$ or when its line search had to shrink past such an error, and a direction that is not a descent direction clears the memory.

6 Seeding synthesis and documents

Writing a seeding by hand is error-prone on real parts. Synthesis derives it from the build function: it rebuilds at $\theta \pm h$ ($h = 10^{-4}$ in the units of the parameter in every result below), matches each base surface to its counterparts by frame-invariant geometric identity (a plane by its normal up to sign and its offset, a cylinder by its axis line and radius, a sphere by center and radius), and reads field velocities off central differences of the matched fields. Surface fields have no combinatorial structure, so the difference is accurate to $O(h^2)$, but the seeding, and every gradient computed with it, carries that error. It also costs two rebuilds per parameter: on the five-parameter box, synthesizing all five seedings takes 917 ms, slightly more than a finite-difference gradient of the whole objective (860 ms). Synthesis pays off only when a seeding is reused across many evaluations at one θ; deriving seedings symbolically from the feature tree would remove both costs and is not implemented.

A named parameter of a `.vcad` document is differentiated the same way: the parameter is bound, the document is evaluated to one BRep per part, the seeding is synthesized, and each solid part returns $d(V, m, c, I)/d\theta$. Evaluations at $\theta \pm h$ must produce the same number of parts, or the call fails.

7 Gradients through rigid-body rollouts

In a rollout of an articulated rigid-body system without contact, the geometry of body b enters the dynamics only through its ten mass-property scalars $p_b = (m, c, I)$. With contact, it also enters through the positions of the vertices that touch. The chain rule is

$$\frac{dJ}{d\theta} = \sum_b \frac{\partial J}{\partial p_b} \cdot \frac{dp_b}{d\theta} + \sum_b \sum_s \langle c_s^{(b)}, \sigma_k(s) \rangle, \quad c_s^{(b)} = \sum_i A_{is}^{\top} \frac{\partial J}{\partial x_i^{(b)}}.$$

$\partial J/\partial p_b$ comes from a discrete adjoint over semi-implicit Euler in the phyz simulator, written over Featherstone's articulated-body algorithm[33]; its within-step Jacobian columns are evaluated on dual numbers, one lane per column. $dp_b/d\theta$ is the seam's mass-property derivative. For contact, phyz applies a penalty force $f_z = \max(0, k\delta - cv_z)$ at every vertex of the body's frozen-plan mesh that lies below the ground, its adjoint returns $\partial J/\partial x$ on those vertices, and the transposed pass turns that into $dJ/d\theta$. A rollout the adjoint cannot express (state feedback, multi-DOF joints) falls back to central differences on the ten scalars, which needs about twenty re-simulations per body and no CAD rebuild.

8 Results

All results come from vcad's test suite at commit `a3ad5eced5` and from the experiments in `repro/` (Section 10), run in the release profile on one thread of a 12th Gen Intel(R) Core(TM) i9-12900K (rustc 1.98.0 (88d9e12ae 2026-08-18)). Each test asserts a tolerance, 10^{-6} for most first-order checks; the values below are what the tests measured. We use two finite-difference references. **Same-plan FD** rebuilds the part at $\theta \pm h$ and re-evaluates the frozen plan captured at θ_0; it checks the seam's derivative of its own plan evaluator. **Rebuild FD** tessellates each rebuild from scratch and shares no code with the seam beyond the kernel itself; it is the independent check, but it is noisy at small h, where node placements differ between rebuilds, and it carries truncation error at large h. Relative errors are $|a - b| \;/\; |b|$, with b the seam's value (or the closed form, where one is named). Rebuild FD is not possible for the irys benchmark (Section 8.8), and the cone's second derivative is checked against its closed form only.

8.1 Validation against finite differences

The models are: an extruded 4 × 3 mm rectangle, depth $d = 2$ mm; a 10 × 8 × 5 mm box minus a through-hole of radius $r = 2.5$ mm (32 circle segments requested; the Boolean's sag-adaptive rim refines the hole to 112); a cylinder of radius 5 mm and height 8 mm (24 segments); a 10 mm cube with every edge filleted at $r = 1.5$ mm (16 segments per full circle, 836 nodes); a cone frustum with base radius 5, top radius 3 and height 8 mm (apex at 20 mm, 24 segments), parameterized by half-angle and by height; and a torus with $R = 8$, $r = 3$ mm (24 segments).

TABLE 1 **First-order volume gradients.** Same-plan FD uses $h = 10^{-6}$. Node-wise is the worst relative error of $dx_i/d\theta$ over all mesh nodes against same-plan FD. Rebuild FD compares against the seam with a synthesized seeding, so that column mixes the seeding's $O(h^2)$ synthesis error with the FD's own error; [a] $h = 10^{-3}$, because the hole's sag-adaptive rim changes its segment count between $r \pm 10^{-2}$; [b] $h = 10^{-5}$ and 10^{-4}, below the cone's truncation error. Figure 7 shows every step size. `data/measured.tsv`.

MODEL	θ	DV/DΘ (SEAM)	SAME-PLAN FD	NODE-WISE	REBUILD FD, $h = 10^{-2}$
Extrusion	depth	12	8.3×10^{-12}	2.9×10^{-11}	2.9×10^{-11}
Box – cylinder	hole r	−78.499	6.2×10^{-10}	6.6×10^{-10}	1.4×10^{-10} [a]
Cylinder	height	77.646	2.9×10^{-9}	3×10^{-10}	3×10^{-10}
Filleted cube	fillet r	−72.807	9.5×10^{-9}	3.2×10^{-9}	2.3×10^{-5}
Filleted cube	edge a	293.665	8.2×10^{-10}	–	5×10^{-6}
Cone frustum	half-angle	3449.54	5.9×10^{-12}	9.6×10^{-11}	1.9×10^{-10} [b]
Cone frustum	height	27.952	4.7×10^{-9}	1.8×10^{-7}	4.1×10^{-10} [b]
Torus	minor r	926.03	4.5×10^{-10}	1.4×10^{-9}	2.1×10^{-9}
Torus	major R	173.63	3.2×10^{-9}	1.3×10^{-9}	3.6×10^{-8}

Every model passes both checks, and the rebuild FD, which is independent of the seam, confirms the same-plan results to the accuracy its noise and truncation allow. Where a discrete closed form exists the seam meets it to rounding: the hole's mesh cross-section is the inscribed 112-gon, so $dV/dr = -N \sin(2\pi/N) rt$, met to 3.6 $\times$ 10^{-16} the cone's half-angle derivative meets the prismatoid volume $V = (hk/3)(R_b^2 + R_t^2 + R_b R_t)$ with $k = N \sin(2\pi/N)/2$ to 1.3 $\times$ 10^{-16} the cylinder-height derivative meets the N-gon cap area to 1.8 $\times$ 10^{-16} and the extrusion's dV/dd is exactly the cross-section area. The mass-property derivatives behave the same way: dI_{zz}/dd of the extrusion meets its closed form to 5.7 $\times$ 10^{-16}, and the cylinder's inertia derivative with respect to radius meets same-plan FD to 8.7 $\times$ 10^{-11}.

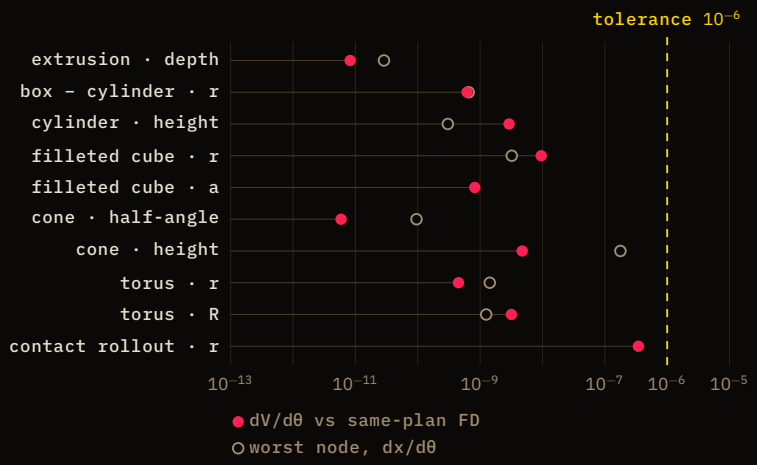


FIG 6 **Every first-order check on one axis.** Filled: the seam's $dV/d\theta$ against same-plan FD, except two rows: the contact rollout's $dJ/d\theta$ is against rebuild-and-resimulate FD, and the filleted cube's edge a is the transposed pass against the forward reference. Open: the worst node-wise $dx/d\theta$. All are below the 10^{-6} tolerance; the cone-height node-wise error (1.8 $\times$ 10^{-7}) and the contact rollout (3.5 $\times$ 10^{-7}) are less than an order below it, the first because the cap-rim nodes ride a sloped ruling and their FD difference is small, the second because its reference re-simulates a penalty contact. `data/measured.tsv`.

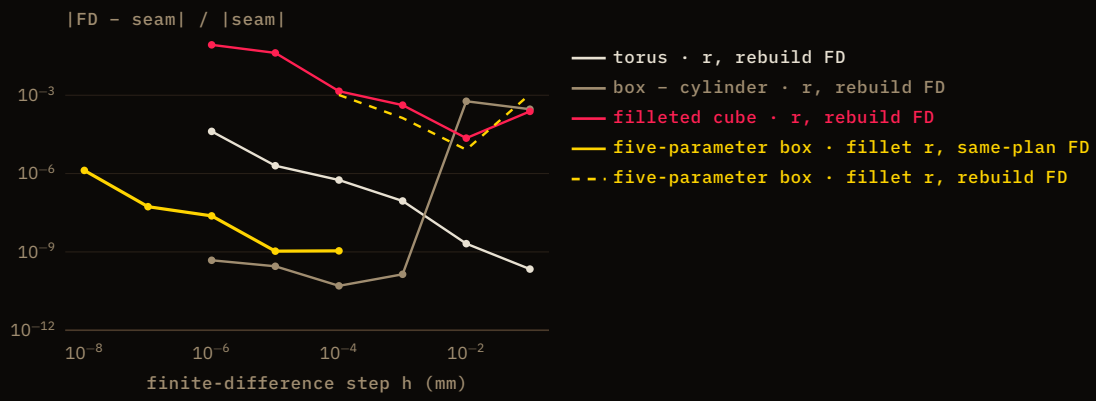


FIG 7 **Finite-difference error against step size.** Solid lines: rebuild FD for three models of Table 1. Each falls until the rebuilds' node placements stop agreeing (left) or truncation takes over (right), and at its best step it meets the seam. Yellow: same-plan FD of the fillet-radius component of the five-parameter box's objective, which has no rebuild noise and falls to roundoff; the dashed yellow line is rebuild FD for the same component. In an earlier kernel revision the yellow line stayed flat at about 10^{-4} across these steps, which is how a seam error was found (Section 8.4). `data/measured.tsv`.

8.2 The polygonization gap

The seam's derivative is that of the mesh, so it differs from the continuum derivative by the polygonization error. For the hole that error is the factor $\sin x/x$ with $x = 2\pi/N$, 5.2×10^{-4} at 112 segments. For the filleted cube it is larger, and larger in magnitude than the continuum value, because an inscribed polygon removes more material than the circular blend and removes it faster as r grows. At 16 segments per full circle (four per quarter-blend) the seam's $dV/dr = -72.807$ against the continuum −68.217, a gap of 6.7%; the gap falls as N^{-2} (Figure 8). Mass properties at four segments per quarter-blend are a coarse model of a filleted part, and the remedy is resolution, not a different derivative.

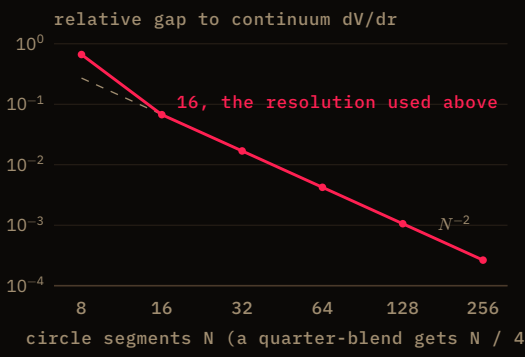


FIG 8 The fillet's polygonization gap. Relative difference between the seam's dV/dr on the filleted cube and the continuum derivative of $V(r) = a^3 + 6a^2r + 3\pi ar^2 + 4\pi r^3/3$, $a = L - 2r$, against the circle segment count. The dashed line is N^{-2}. At 128 segments the gap is 0.11%. `data/measured.tsv`.

8.3 Transposed pass, synthesis and second order

TABLE 2 Transposed pass, synthesized seedings and second derivatives. The transposed and forward passes share their rows, so their agreement checks the linear algebra only. Synthesized seedings carry the $O(h^2)$ error of the field differences. Second derivatives are checked against central differences of the seam's own first derivative on fresh rebuilds, at the step where that difference has converged. The Gauss–Newton comparison is on the five-parameter box near its optimum. `data/measured.tsv`.

CHECK	SEAM VALUE	REL. DIFFERENCE
Transposed vs forward, hole dV/dr	−78.4986	2.2×10^{-15}
Transposed vs forward, fillet dV/dr	−72.8072	2.9×10^{-15}
Synthesized vs hand seeding, fillet dV/dr	−72.8072	1.8×10^{-9}
Synthesized vs hand seeding, edge dV/da	293.665	7.5×10^{-10}
Synthesized, torus vs same-plan FD	926.03	3.1×10^{-10}
d²V/dr², hole vs closed form	−31.399	2×10^{-15}
d²V/dα², cone vs closed form	16385.3	2.2×10^{-16}
d²V/dr², torus minor vs rebuild FD of dV/dr, $h = 10^{-1}$	308.677	7.8×10^{-11}
d²V/dθ², torus with r = θ² vs chain rule	5556.2	2.1×10^{-15}
d²V/dr², filleted cube vs rebuild FD of dV/dr, $h = 10^{-1}$	−40.764	2.4×10^{-6}
Gauss–Newton vs full Hessian (Frobenius)	–	1.5×10^{-5}

For the torus the rebuild difference of dV/dr is dominated by rebuild noise at small h and converges onto the seam's second derivative as h grows (from 1.7×10^{-6} at $h = 10^{-4}$ to 7.8×10^{-11} at 10^{-1}); the 1.7×10^{-6} in vcad's own test is its smaller step. The filleted cube behaves the same way: the difference reaches −40.7642 at $h = 10^{-1}$ against the seam's −40.7643, and a five-point stencil on one frozen plan agrees with the seam to 2.5×10^{-7}. An earlier revision of the kernel failed this check by 10^{-3} at every step (Section 8.4).

8.4 Five parameters of a filleted, drilled box

The part is a box of dimensions (s_x, s_y, s_z) with every edge filleted at radius r and a centered through-hole of radius r_h: 6 planes, 12 blend cylinders, 8 corner spheres and the hole wall, 1046 mesh nodes at θ^* (16 segments per circle). The objective is the squared relative miss of five quantities (volume, the three centroid coordinates and I_{zz}) against their values at $\theta^* = (10, 12, 8, 1.8, 2.2)$ mm, starting from $\theta_0 = (8, 14, 6, 1.2, 1.6)$ mm. The five-by-five Jacobian of the quantities has determinant 35689 and smallest pivot 0.34, so the parameters are locally identifiable.

Two errors the finite differences caught. At θ_0, same-plan FD at $h = 10^{-6}$ agrees with the seam's gradient to 9.2×10^{-10} for s_x, 2.7×10^{-9} for s_z, 2.4×10^{-8} for the fillet radius and 5.3×10^{-12} for the hole radius, and rebuild FD of the fillet-radius component reaches 8.2×10^{-6} at $h = 10^{-2}$ (Figure 7). This was not so in the kernel revision we first measured (`d10f279`). In that run, which `repro.sh` does not regenerate and whose figures we quote, the fillet-radius component sat 1.4×10^{-4} from same-plan FD and the s_x component 2.4×10^{-5}, at every step from 10^{-8} to 10^{-6}, and the filleted cube's second derivative missed by 10^{-3}. An error that does not move with the step is not finite-difference noise, and both had causes in the seam. A Boolean rounds the centers of a corner sphere and its blend cylinder through single precision, so the two surfaces cross at about 3×10^{-4} rad instead of meeting tangentially; the tessellator classified their shared arc as a trim between two surfaces and tracked it with an ill-conditioned Newton solve. Such near-tangent pairs now take the sphere's surface rule, which removed the fillet-radius and second-derivative errors. Separately, a vertex on the hole's rim has only plane and cylinder rows, which left its slide along the rim at zero while the kernel moves it with the hole; volume cannot see that, moments can. Those directions now follow the vertex's curved surface. Both fixes are in the pinned commit, each with a regression test.

TABLE 3 Recovering θ* on the five-parameter box. Iterations and wall-clock time until $\max|\theta - \theta^*|$ first falls below the tolerance. All L-BFGS runs share one driver (memory 8, Armijo line search on rebuilt-from-scratch objective values, 200 iterations); only the gradient differs. Levenberg–Marquardt uses the seam's exact 5 × 5 Jacobian of the residuals. `data/measured.tsv`.

METHOD (SAME DRIVER)	TO 10^{-3} MM	TO 10^{-6} MM	FINAL ERROR
Levenberg–Marquardt, seam Jacobian	3 it · 0.5 s	4 it · 0.6 s	1.2×10^{-12}
L-BFGS, seam gradient	21 it · 2.5 s	28 it · 3.7 s	1.5×10^{-7}
L-BFGS, rebuild FD, $h = 10^{-4}$	20 it · 12.8 s	not reached	1.5×10^{-6}
L-BFGS, rebuild FD, $h = 10^{-6}$	not reached	not reached	4.2×10^{-1}
L-BFGS, rebuild FD, $h = 10^{-2}$	23 it · 16.2 s	not reached	7.6×10^{-4}

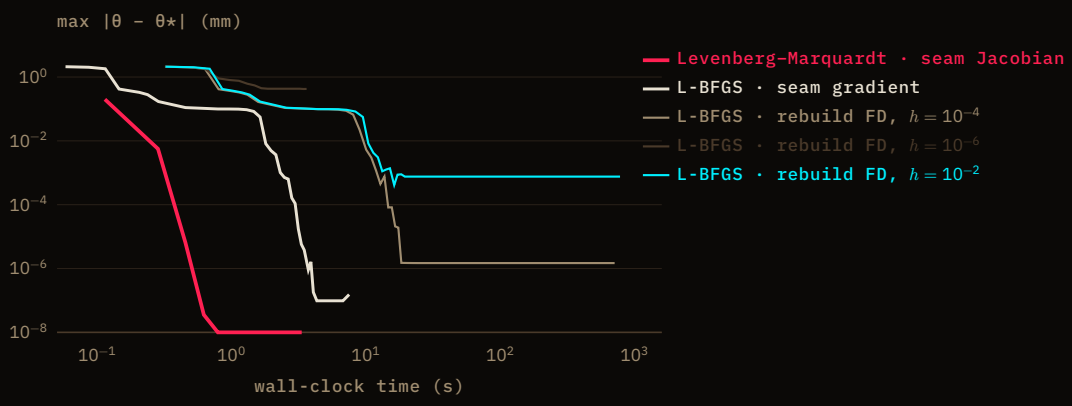


FIG 9 Parameter error against wall-clock time for the runs of Table 3. Every curve rebuilds the part at each objective evaluation; the gradient source is the only difference. `data/measured.tsv`.

What exact derivatives buy here is not only that L-BFGS converges: with rebuild FD at a well-chosen step ($h = 10^{-4}$) it reaches 10^{-3} in a similar number of iterations and stalls at 1.5×10^{-6} mm in this run (rebuild FD is not repeatable to 10^{-6}, because rebuilds do not place nodes identically, so where it stalls varies from run to run). They buy cost and robustness. A seam gradient (rebuild, capture, one pass and one transposed pass) takes 85 ms on this part against 860 ms for a rebuild-FD gradient, a factor of 10 at five parameters that grows with the parameter count; and FD at a too-small (10^{-6}) or too-large (10^{-2}) step ends 4.2×10^{-1} and 7.6×10^{-4} mm from θ^*. Because the residuals are square and vanish at θ^*, Gauss–Newton with the exact Jacobian is the natural method, and Levenberg–Marquardt reaches 10^{-6} in 4 iterations and 0.62 s. Its first iterates converge quadratically, as an exact Jacobian at zero residual should. Below about 10^{-8} mm the error shrinks only by a constant factor per iterate. At that scale the rebuilt objective is no longer smooth, because of the sag-adaptive jumps of Section 1.1 and single-precision rounding in the Boolean, and the damping never fully relaxes.

The rebuild dominates each iterate: 82 ms against 0.9 ms for the capture, 0.24 ms for one forward pass, 0.05 ms for the dual mass properties of that pass, and 0.44 ms for the transposed pass. The transposed pass costs about 1.8 forward passes on this part, so it pays from two parameters up. (On the filleted cube without the hole, Figure 12, it costs about three, because that part has fewer vertex rows to transpose relative to its surface nodes.)

8.5 A fillet radius and a flywheel

Projected gradient descent on the fillet radius of the 10 mm cube alone recovers the radius whose volume matches a target, $r^* = 2.2$ mm from $r_0 = 1$ mm, landing at 2.2 mm in 12 accepted iterates.

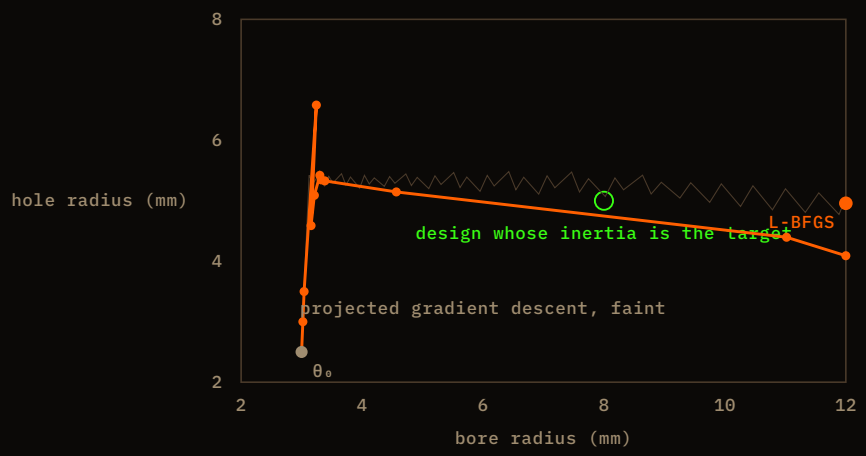


FIG 10 A flywheel design brief. A disc (radius 40 mm, thickness 10 mm) with a center bore and four lightening holes on a 25 mm bolt circle; θ = (bore radius, hole radius). Objective: mass over a reference mass plus 100 · the squared relative miss of a target spin inertia, the inertia of the design at (8, 5) mm. The bounds keep the holes inside the disc and clear of each other. L-BFGS (bold) stops after 11 iterations with the inertia within 1.1% of the target and 3.3% less mass than the (8, 5) design, the bore on its upper bound (the bound that keeps it clear of the lightening holes); projected gradient descent (faint) takes 77 steps along a zig-zag to a similar design. `data/measured.tsv`.

Each flywheel iterate takes seconds because the build does: five Booleans at 24 segments take 4.2 s, against 7 ms to capture its 2552-node plan.

8.6 A named document parameter

A `.vcad` document whose cylinder radius is bound to the named parameter `r` differentiates with no hand seeding: $dV/dr = 501.848$ against the N-gon closed form 501.848 (2.3×10^{-12}), and dI_{zz}/dr agrees with FD of document rebuilds to 4.1×10^{-11}. A document written in vcad's loon language and one whose parameter is declared with `defparam` reach the same closed form to 2.3×10^{-12} and 2.3×10^{-12}. These gradients use synthesized seedings and carry their $O(h^2)$ error.

8.7 Shape through dynamics and contact

TABLE 4 Gradients of rollouts with respect to a CAD radius: J in rad/s, rad and m; θ in mm. The first five rows use a cylinder (radius 10 mm, height 8 mm, density 2700 kg/m³) on one joint, θ its radius; the last uses a 20 mm cube filleted at $r = 2$ mm, θ the fillet radius. The reference rebuilds the CAD, re-tessellates, recomputes mass properties and re-simulates, at $h = 10^{-3}$ mm (cylinder) and 10^{-2} mm (filleted box). Contact: $k = 100$ N/m and $c = 0.5$ N·s/m per vertex. The adjoint's pendulum mounts the same cylinder on a revolute about the body x axis instead of rotating its inertia, so its value differs from the first pendulum's. `data/measured.tsv`.

ROLLOUT	DJ/DΘ	VS REBUILD-AND-RESIMULATE FD
Flywheel spin-up, $J = \omega(T)$, FD on p	−23.6544	5×10^{-8}
Gravity pendulum, $J = q(T)$, FD on p	−0.22795	8.3×10^{-10}
Flywheel, adjoint	−23.6544	5×10^{-8}
Pendulum on a body-x revolute, adjoint	0.19138	1.2×10^{-9}
Cylinder resting on its side, contact	9.82×10^{-4}	3.5×10^{-7}
Filleted box resting on a blend, contact	-4.639×10^{-4}	9.9×10^{-7}

The cylinder rows are a check of the chain on a body whose mass properties have closed forms. The last row exercises the seam through dynamics: a 20 mm cube with every edge filleted at 2 mm rests on one blended edge, held by a vertical prismatic joint, and J is its settled height. The fillet radius moves 20 surfaces. Growing it raises the lowest point of the blend by $\sqrt{2}-1$ per unit radius, so the surface channel alone would give $dJ/d\theta = -4.142 \times 10^{-4}$ m per mm; the gradient, -4.639×10^{-4}, adds the mass channel and the change in penetration depth of the blend vertices in contact, and matches a rebuild-and-resimulate FD to 9.9×10^{-7}. The gradient holds the set of contacting vertices fixed. A change of that set is a discontinuity, and no gradient sees it.

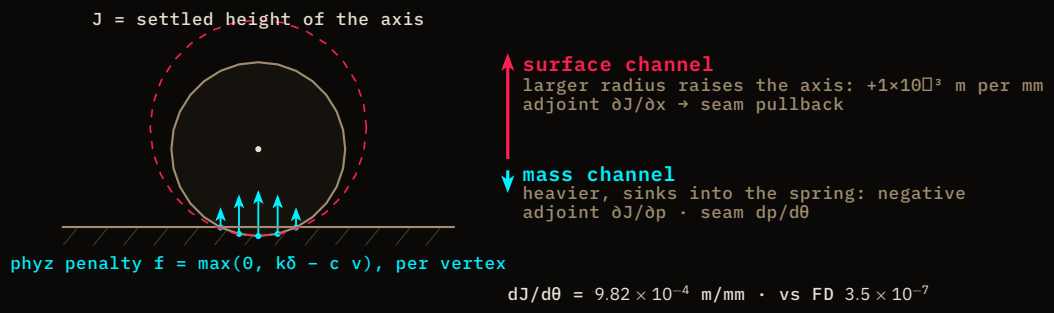


FIG 11 Shape through contact. The cylinder of Table 4 rests on its side on phyz's per-vertex penalty springs; J is its settled height. Growing the radius raises the body through the surface channel and sinks it through the mass channel. `data/measured.tsv`.

8.8 Pixels to CAD parameters

The transposed pass accepts any cotangent on the plan's nodes, including one from a renderer. With the irys differentiable path tracer[34] we pulled an image gradient back to a CAD dimension on a real part: the left shoulder-pitch output plate of the Rose robot arm, a 3 mm rounded plate with a 49 mm bearing bore and twelve 4.4 mm bolt holes. θ is the bore diameter; the plan has 4942 nodes at 32 circle segments and a synthesized seeding. irys renders the plan's mesh as it is against an empty background at 160 × 128 pixels, so the whole derivative comes from visibility edges, and its edge sampler returns $\partial\mathcal{L}/\partial x$ on every node for a fixed smooth image loss (100000 samples, 8 seeds).

TABLE 5 The rose-arm benchmark at irys `5e09686`, rose `832192f` and vcad `a3ad5ec`, regenerated by `repro/irys.sh`. The transposed pass's Monte Carlo standard error over the seeds is ±0.0226, which covers the seam's paths but not the render FD. At $h = 0.02$, 0.05 and 0.1 mm the render FD gives 40.353, 40.437 and 40.014, a spread of 1.1% with no trend in h, so quadrature noise sets its error. The transposed pass lies inside that spread. The render check is good to about 1%, and the seam checks above it to 10^{-5} or better.

PATH TO DL/DΘ	VALUE	VS TRANSPOSED PASS
Transposed pass	40.16864	–
Forward pass, $\sum \partial\mathcal{L}/\partial x \cdot \dot{x}$	40.16864	1.2×10^{-15}
Same-plan FD node velocities, $h = 10^{-5}$ mm	40.17028	4.3×10^{-5}
Render FD with nodes moved along FD velocities, $h = 0.1$ mm	40.0144	3.9×10^{-3}

A rebuild FD is not possible here. A rebuild at $\theta \pm h$ loses correspondence for $h \geq 2 \times 10^{-3}$ mm and changes topology at 0.05 mm, where the bore rim's sag-adaptive arc count drops by two vertices, while at $h = 5 \times 10^{-4}$ mm the render FD is dominated by quadrature noise. The last row therefore moves the plan's nodes along their same-plan FD velocities instead of rebuilding, which checks the renderer's derivative but not the seam's; the seam's part of the chain is checked by the third row.

8.9 Cost

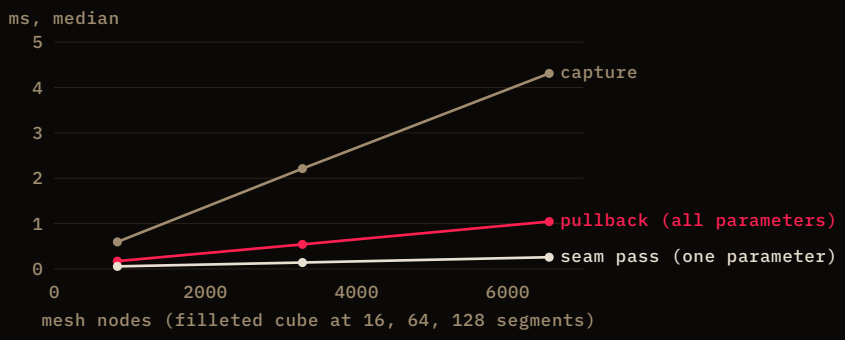


FIG 12 Per-call cost on the filleted cube at three blend resolutions, median. Capturing a plan, one forward pass and one transposed pass are all linear in the number of mesh nodes. `data/measured.tsv`.

9 Discussion

Why freeze the tessellation. Differentiating the kernel's code has to give some answer at every branch: which face a Boolean keeps, how a fillet orders its blends, how many segments a circle gets. Those answers are piecewise constant in θ, and in vcad they are also order-nondeterministic. Freezing the plan makes the discrete state explicit and checkable. The cost is the limits of Section 1.1: a plan is valid for steps of 5.6×10^{-4} mm on the five-parameter box, segment counts change between iterates, and the derivative is that of the mesh. For optimization this has been enough on every part we tried, because each iterate captures a new plan and the jumps between plans are of order 10^{-6} of the objective; for an objective that depends on fine surface detail, the polygonization gap would dominate before the jumps do.

What the finite differences show. On every plane, cylinder, cone and torus model, same-plan FD, rebuild FD and the seam agree. On filleted solids rebuild FD is noisy at small steps, because rebuilds place nodes differently, and has to be read at a step of 10^{-3} to 10^{-2} mm (Figure 7). At those steps it confirms the seam on the filleted cube and the filleted, drilled box. Same-plan FD is the sharper test, since it has no rebuild noise, and it is the one that exposed the two seam errors of Section 8.4: each showed as an error that stayed fixed while the step moved over two decades.

Toward an operator graph. The rollout coupling of Section 7 composes three steps by hand: the seam (parameters to node positions, with its transposed pass), mass properties (node positions to ten scalars per body) and the phyz adjoint (scalars and contact vertices to $\mathcal{J}$). Each is a step with a forward map and an exact vector–Jacobian product, which is the shape of an operator node in a differentiable program[29]. We are building that graph on the tang tensor compiler[35], where linear

solves and simulations become nodes with their own adjoints; none of that is evaluated here.

Remaining gaps. Cone-tangent-to-plane tangency rows are not implemented; fillets with asymmetric corners or three blends at a vertex fall back to an older pipeline that is not differentiated; the contact model is frictionless; and the method is demonstrated only on vcad. Nothing in it depends on vcad beyond analytic surfaces that evaluate on dual numbers and a tessellator that can classify its nodes.

10 Reproducibility

`repro/repro.sh` clones vcad at `a3ad5eced5a0fbfff8373e10d81beae201c69b3d` (its lockfile pins phyz, loon and tang) and applies `report.patch`, which adds `eprintln!` lines after statements in vcad's tests that already compute a measured quantity, so each test still asserts its own tolerance and prints the value it checks. It then copies in `paper_experiments.rs` and `paper_contact_fillet.rs`, the experiments added for this paper (rebuild FD, step sweeps, resolution study, optimizer baselines, costs, plan radius, segment-count jumps, second-order sweeps, the flywheel L-BFGS run and the filleted-box contact rollout), runs everything, runs `irys.sh` for the rose-arm benchmark, and writes `data/measured.tsv`. This document reads every measured value, table cell and plotted point from that file, with two exceptions stated where they occur: the figures quoted from the earlier kernel revision `d10f279` (Section 8.4), and the correspondence and topology thresholds of the irys rebuild probe, which come from a separate probe run (Section 8.8). Model construction counts (segments requested, the 112-gon rim) are inputs, not measurements. The run took place on 2026-11-10T16:01Z on Linux 6.8.0-147-generic x86_64.

11 Conclusion

Freezing a BRep's tessellation and differentiating its node positions gives derivatives of CAD parameters, fillet radii included, that are exact for the mesh, extend to second order and a transposed pass, and carry through to mass properties and rigid-body dynamics with contact. The approach follows Engineering Sketch Pad's tessellation sensitivities; within that approach our additions are analytic fillet sensitivities, second derivatives, the transposed pass and the dynamics coupling. Its limits are those of the frozen mesh, a polygonization gap that falls as N^{-2} and plans valid for small steps. Checking against finite differences of one frozen plan found two errors in the seam, both since fixed. vcad, the experiments and the data are open source.

References

1. Dannenhoffer, J. F. & Haimes, R. Design Sensitivity Calculations Directly on CAD-based Geometry. in *53rd AIAA Aerospace Sciences Meeting* (2015).
2. Dannenhoffer, J. F. & Haimes, R. Using Design-Parameter Sensitivities in Adjoint-Based Design Environments. in *55th AIAA Aerospace Sciences Meeting* (2017).

3. Banović, M. *et al.* Algorithmic differentiation of the Open CASCADE Technology CAD kernel and its coupling with an adjoint CFD solver. *Optimization Methods and Software* **33,** 813–828 (2018).
4. Mykhaskiv, O., Müller, J.-D. & others. NURBS-based and Parametric-based Shape Optimisation with Differentiated CAD Kernel. in *CAD'17 Conference* 415–419 (2017). doi:10.14733/cadconfP.2017.415-419.
5. Gilbert, E. G., Johnson, D. W. & Keerthi, S. S. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal on Robotics and Automation* **4,** 193–203 (1988).
6. Haftka, R. T. & Grandhi, R. V. Structural shape optimization—a survey. *Computer Methods in Applied Mechanics and Engineering* **57,** 91–106 (1986).
7. Sokołowski, J. & Zolésio, J.-P. *Introduction to Shape Optimization: Shape Sensitivity Analysis*. (Springer, 1992).
8. Robinson, T. T., Armstrong, C. G., Chua, H. S., Othmer, C. & Grahs, T. Optimizing parameterized CAD geometries using sensitivities based on adjoint functions. *Computer-Aided Design and Applications* **9,** 253–268 (2012).
9. Agarwal, D. *et al.* Parametric design velocity computation for CAD-based design optimization using adjoint methods. *Engineering with Computers* **34,** 225–239 (2018).
10. Xu, S., Jahn, W. & Müller, J.-D. CAD-based shape optimisation with CFD using a discrete adjoint. *International Journal for Numerical Methods in Fluids* **74,** 153–168 (2014).
11. Hughes, T. J. R., Cottrell, J. A. & Bazilevs, Y. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering* **194,** 4135–4195 (2005).
12. Cascaval, D. *et al.* Differentiable 3D CAD Programs for Bidirectional Editing. *Computer Graphics Forum* **41,** 309–323 (2022).
13. Michel, É. & Boubekeur, T. DAG Amendment for Inverse Control of Parametric Shapes. *ACM Transactions on Graphics* **40,** (2021).
14. Schulz, A. *et al.* Interactive Design Space Exploration and Optimization for CAD Models. *ACM Transactions on Graphics* **36,** (2017).
15. Kato, H., Ushiku, Y. & Harada, T. Neural 3D Mesh Renderer. in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2018).
16. Liu, S., Li, T., Chen, W. & Li, H. Soft Rasterizer: A Differentiable Renderer for Image-based 3D Reasoning. in *IEEE International Conference on Computer Vision (ICCV)* (2019).
17. Li, T.-M., Aittala, M., Durand, F. & Lehtinen, J. Differentiable Monte Carlo Ray Tracing through Edge Sampling. *ACM Transactions on Graphics* **37,** (2018).
18. Worchel, M. & Alexa, M. Differentiable Rendering of Parametric Geometry. *ACM Transactions on Graphics* **42,** 232:1–232:18 (2023).

19. Park, J. J., Florence, P., Straub, J., Newcombe, R. & Lovegrove, S. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2019).
20. Vicini, D., Speierer, S. & Jakob, W. Differentiable Signed Distance Function Rendering. *ACM Transactions on Graphics* **41,** (2022).
21. Sharma, G., Goyal, R., Liu, D., Kalogerakis, E. & Maji, S. CSGNet: Neural Shape Parser for Constructive Solid Geometry. in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2018).
22. Du, T. *et al.* InverseCSG: Automatic Conversion of 3D Models to CSG Trees. *ACM Transactions on Graphics* **37,** (2018).
23. Avila Belbute-Peres, F. de, Smith, K., Allen, K., Tenenbaum, J. & Kolter, J. Z. End-to-End Differentiable Physics for Learning and Control. in *Advances in Neural Information Processing Systems (NeurIPS)* (2018).
24. Geilinger, M. *et al.* ADD: Analytically Differentiable Dynamics for Multi-Body Systems with Frictional Contact. *ACM Transactions on Graphics* **39,** (2020).
25. Werling, K., Omens, D., Lee, J., Exarchos, I. & Liu, C. K. Fast and Feature-Complete Differentiable Physics for Articulated Rigid Bodies with Contact. in *Robotics: Science and Systems (RSS)* (2021).
26. Hu, Y. *et al.* DiffTaichi: Differentiable Programming for Physical Simulation. in *International Conference on Learning Representations (ICLR)* (2020).
27. Ha, S., Coros, S., Alspach, A., Kim, J. & Yamane, K. Joint Optimization of Robot Design and Motion Parameters using the Implicit Function Theorem. in *Robotics: Science and Systems (RSS)* (2017).
28. Xu, J. *et al.* An End-to-End Differentiable Framework for Contact-Aware Robot Design. in *Robotics: Science and Systems (RSS)* (2021).
29. Blondel, M. *et al.* Efficient and Modular Implicit Differentiation. in *Advances in Neural Information Processing Systems (NeurIPS)* (2022).
30. Griewank, A. & Walther, A. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. (SIAM, 2008).
31. Mirtich, B. Fast and Accurate Computation of Polyhedral Mass Properties. *Journal of Graphics Tools* **1,** 31–50 (1996).
32. Liu, D. C. & Nocedal, J. On the limited memory BFGS method for large scale optimization. *Mathematical Programming* **45,** 503–528 (1989).
33. Featherstone, R. *Rigid Body Dynamics Algorithms*. (Springer, 2008).
34. Municipal Robotics. irys: a differentiable path tracer. (2026).
35. Municipal Robotics. tang: a tensor compiler. (2026).