Automatic Kinematic Classification and Collision Avoidance Algorithms for Panel Bending CAM (Salvagnini P4 Architecture)

1. Introduction and Architectural Context

The transition from Computer-Aided Design (CAD) to Computer-Aided Manufacturing (CAM) for automated panel benders represents one of the most complex challenges in computational geometry and industrial automation. While Phase 1 of the "StreamBend" project successfully established a static topological graph from STEP files—identifying faces, bend features, and the base reference face—Phase 2 demands a paradigm shift from static analysis to kinematic simulation. The objective is no longer merely to know *that* a bend exists, but to determine *how* that bend interacts with the machine's physical constraints during the forming process.

This report provides a comprehensive, expert-level analysis of the algorithmic logic required to implement Kinematic Directionality and Kinematic Validity engines for the Salvagnini P4 panel bending architecture. Unlike press brakes, which rely on air bending and often manual manipulation, the P4 system utilizes a constrained "folding" mechanism where the sheet metal blank rests horizontally on a manipulator table, clamped by a variable blankholder, while independent upper and lower blades actuate to fold the material.[1] This architecture imposes a rigorous dichotomy on bending operations: "Positive" bends (actuated by the lower blade moving up) and "Negative" bends (actuated by the upper blade moving down) possess fundamentally different risk profiles regarding collision and kinematic feasibility.

The central thesis of this analysis is that geometric validity does not imply kinematic safety. A bend may be mathematically possible in an unconstrained Euclidean space but physically impossible within the volumetric constraints of the P4's blankholder throat and manipulator operational envelope. Therefore, the CAM kernel must implement a two-stage validation pipeline: first, a deterministic classification of bend directionality using vector algebra relative to the base face normal; and second, a rigorous collision simulation that models the swept volume of the flange against the "Negative" (safe) and "Positive" (high-risk) zones of the machine tools.

1.1 The Fundamental Divergence: Press Brake vs. Panel Bender Kinematics

To understand the necessity for the complex vector logic detailed in this report, one must

appreciate the kinematic divergence between traditional press brakes and panel benders. In a press brake, the operator or robot holds the part, and the ram descends vertically. The collision constraints are primarily the tool profile and the backgauge. The part rotates in free space, typically upwards.

In the Salvagnini P4 architecture, the kinematics are inverted and constrained. The sheet is clamped horizontally by a manipulator (the "manipulator clamp") which moves in the X-Y plane and rotates (Theta/C-axis). The bending locus is fixed. The material to be bent (the flange) is cantilevered out from the manipulator. A "Blankholder" (ABA/MLA) descends to clamp the material against the counterblade table.

- **Kinematic Constraint A:** The bulk of the sheet (Base Face) is fixed to the XY plane ($Z = 0$).
- **Kinematic Constraint B:** The Blankholder occupies the volume directly *above* the sheet ($Z > Thickness$) and *behind* the bend line (towards the manipulator).
- **Kinematic Constraint C:** The Bending Blades oscillate. The Lower Blade moves $+Z$ to fold up; the Upper Blade moves $-Z$ to fold down.

This creates the "Collision Volume" problem. When a flange is bent "Up" (Positive), it enters the exact half-space ($Z > 0$) occupied by the Blankholder tool that is clamping the sheet. If the flange is taller than the tool's vertical clearance, or if it hooks "backwards" (a return bend) deeper than the tool's throat geometry, a catastrophic collision occurs.[2] Conversely, when a flange is bent "Down" (Negative), it moves into the machine bed or "pit," which generally offers significantly more clearance, although table collisions remain a boundary condition.

1.2 The Scope of Kinematic Directionality

Kinematic Directionality is the algorithmic process of assigning a machine-relative vector to a topological feature. In the static graph (Phase 1), a "Bend" is simply a cylindrical face connecting Face A and Face B. In the kinematic graph (Phase 2), this bend must be explicitly typed as BEND_UP or BEND_DOWN. This classification dictates:

1. **Blade Selection:** Which servo-hydraulic axis is energized (Lower vs. Upper).
2. **Safety Logic:** Which collision volume model (Blankholder vs. Table) is activated for interference checking.
3. **Sequence Optimization:** P4 machines often prioritize negative bends to avoid "trapping" the part or colliding with the blankholder in subsequent steps.

The derivation of this directionality requires rigorous vector algebra, specifically leveraging the Normal Vector of the Base Face as the immutable "Truth" vector for the simulation.

2. Mathematical Framework: Vector Algebra for Bend Classification

The core algorithmic task is to classify the spatial relationship between the Base Face (fixed) and the Flange Face (moving) relative to the machine's global coordinate system. We assume the STEP file represents the final, folded state of the part. The goal is to reverse-calculate the kinematic trajectory.

2.1 Coordinate System Definition

We define the Global Machine Coordinate System ($\mathcal{M}$) such that:

- The **XY Plane** corresponds to the surface of the machine table (and the bottom surface of the sheet metal).
- The **Z Axis** is vertical. $+Z$ points towards the Blankholder/Upper Blade; $-Z$ points towards the machine bed.
- The **Base Face** of the topological graph is mechanically constrained to lie on the XY plane.

Consequently, the Normal Vector of the Base Face ($\mathbf{n}_{base}$), assuming it represents the *top* surface of the sheet, aligns with the Machine $+Z$ axis:

$$\mathbf{n}_{base} \approx \langle 0, 0, 1 \rangle$$

2.2 The Half-Space Classification Algorithm

The most robust method for determining bend direction is **Half-Space Classification**. This approach is superior to local edge winding methods because it is topologically invariant to complex edge cases (like holes near the bend line) and directly maps to the machine's Z-axis constraint.

We define the plane Π_{base} containing the Base Face using the Point-Normal form. Let $\mathbf{c}_{base}$ be the centroid (Center of Mass) of the Base Face, and $\mathbf{n}_{base}$ be its unit normal.

$$\Pi_{base}(\mathbf{p}) = (\mathbf{p} - \mathbf{c}_{base}) \cdot \mathbf{n}_{base} = 0$$

For the connected Flange Face, we calculate its centroid $\mathbf{c}_{flange}$. We then determine the signed distance D of this centroid from the Base Plane:

$$D = (\mathbf{c}_{flange} - \mathbf{c}_{base}) \cdot \mathbf{n}_{base}$$

This scalar value D provides the classification:

1. **Case:** $D > \epsilon$ **(Positive Half-Space)**

   The flange centroid lies in the direction of the normal. Since $\mathbf{n}_{base}$ points Up (towards the tool), the flange exists in the "Upper" world.
   - **Classification:** BEND_UP (Positive Bend).
   - **Actuator:** Lower Blade.
   - **Risk:** High (Blankholder Collision).
2. **Case:** $D < -\epsilon$ **(Negative Half-Space)**
   The flange centroid lies opposite to the normal. The flange exists in the "Lower" world.
   - **Classification:** BEND_DOWN (Negative Bend).
   - **Actuator:** Upper Blade.
   - **Risk:** Low (Table Collision).
3. **Case:** $|D| \leq \epsilon$ **(Coplanar)**
   The flange is coplanar with the base. This indicates a flat feature (no bend), a hem (180° bend compressed flat), or a parsing error.
   - **Secondary Logic:** Check the "Bend Feature" properties (radius, angle). If Angle $\approx 180^\circ$, it is a Hem. If Angle $\approx 0^\circ$, it is a continuous sheet.

2.3 Dot Product Derivation and Robustness

The foundation of this logic is the Dot Product ($\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}||\mathbf{b}| \cos\theta$). As established in vector calculus research [5], the dot product is a projection operator. By projecting the displacement vector $\mathbf{v}_{disp} = \mathbf{c}_{flange} - \mathbf{c}_{base}$ onto the base normal $\mathbf{n}_{base}$, we isolate the Z-component of the translation.

This method remains robust even for non-90° bends.

- **Acute Bend (e.g., 135° open angle, 45° internal):** The centroid still lifts significantly in Z. $D > 0$.
- **Obtuse Bend (e.g., 175° open angle, 5° internal):** The Z-lift is small. The tolerance ϵ must be chosen carefully based on machine precision (typically $\epsilon = 10^{-4}$ mm).
- **Complex Topology (Z-bends/Jogs):** If a flange has sub-flanges (e.g., Base -> Flange A -> Flange B), the classification is hierarchical. Flange A is bent relative to Base. Flange B is bent relative to Flange A. The machine actuates them sequentially. However, for

collision checking, the *entire sub-assembly* (Flange A + Flange B) is the "Moving Mass." The algorithm must compute the bounding box of the entire child tree, not just the immediate neighbor.

2.4 Implementing in OpenCASCADE

Implementation within the OpenCASCADE Technology (OCCT) framework requires careful handling of topological orientation. A TopoDS_Face has a geometric surface (e.g., Geom_Plane) and an orientation flag (TopAbs_FORWARD or TopAbs_REVERSED). The geometric normal of the underlying plane might point "Down" ($0, 0, -1$), but if the Face orientation is REVERSED, the *logical* material normal points "Up" ($0, 0, 1$).

Correct Normal Extraction Logic:

1. Retrieve the underlying surface: Handle(Geom_Surface) S = BRep_Tool::Surface(face).
2. Sample the geometric normal at the centroid UV: geom_n.
3. Check orientation: bool is_reversed = (face.Orientation() == TopAbs_REVERSED).
4. Compute Logic Normal: n_base = is_reversed? -geom_n : geom_n.

This ensures that the vector $\mathbf{n}_{base}$ always points *out* of the bulk material, which is critical for the "Material Side" vs. "Air Side" distinction essential for collision logic.[8]

3. Kinematic Validity and Collision Volume Logic

Having classified the bend direction, the system must now validate safety. The user prompt correctly notes: *"A unique constraint of Panel Benders... is that 'Negative Bends' (Down) are generally safer than 'Positive Bends' (Up)."* This asymmetry is a direct consequence of the P4's "Blankholder" architecture.

3.1 The Physics of the Blankholder (ABA/MLA)

The Blankholder is a massive steel assembly that descends to clamp the sheet. Unlike the "Upper Beam" of a press brake which is a single narrow edge, the Panel Bender blankholder is a volumetric tool with significant depth. It is composed of segments (ABA - Automatic Blankholder Adjustment) to accommodate different part lengths.[3]

The Collision Volume ($\mathcal{V}_{tool}$):

The space occupied by the tool can be modeled as a polyhedron defined by:

1. **Tool Clearance Height (H_{tool}):** The vertical distance from the sheet surface ($Z = 0$) to the tool holder mechanism. Typical values range from 127mm to 254mm depending on the model (e.g., P4-3126 vs P4-2120).[12]

2. **Throat Depth / Undercut (D_{throat}):** The horizontal relief behind the clamping tip. This allows for "Return Bends" (flanges that bend up and then inward back over the sheet). The tool is not a solid block; it has a "C" profile or undercut to allow material to curl inside it.
3. **Heel Width:** The dimension of the tool tip contacting the sheet.

Positive Bend Hazard: When the Lower Blade moves up, it pushes the flange into the $+Z$ half-space. The flange sweeps through a volume. If the flange is a simple 90° bend, it creates a vertical wall. If the wall height $H_{flange} > H_{tool}$, the flange hits the tool holder. *Crucially*, if the flange creates a "C" channel (bending back towards the base), the flange tip moves *horizontally* towards the tool body. If this horizontal intrusion exceeds D_{throat}, the flange collides with the back of the tool.[2]

Negative Bend Safety:

When the Upper Blade moves down, it pushes the flange into the $-Z$ half-space. The space below the table is the "bending pit." The Lower Blade retracts to clear the path. The only obstacles are the machine bed (at substantial depth, e.g., -500mm) and potentially the manipulator grippers if the bend is extremely close to the clamp. Generally, unless the flange is extraordinarily long ($> \text{Pit Depth}$), negative bends are collision-free relative to the tooling.

3.2 Computational Collision Simulation Strategies

The user asks: *"Does the CAM software simulate this? Does it use a simplified 'Bounding Box Check'... or a full mesh interference check?"*

High-performance CAM kernels utilize a **Hierarchical Collision Detection** strategy, moving from cheap heuristics to expensive mesh calculations only when necessary.[15]

Level 1: The Heuristic Bounding Box Check (AABB)

This is the "Short-Circuit" test. It is computationally $O(1)$ after the bounding box is computed.

- Compute the Axis-Aligned Bounding Box (AABB) of the flange in its *final folded state*.
- Extract Z_{max} (highest point) and Y_{min} (furthest point towards the tool).
- **Rule:** If $Z_{max} > H_{tool}$, flag "Potential Collision".
- **Critique:** This is conservative. A flange might be tall but narrow enough to fit in a gap between tool segments, or shaped such that the tall part is far away from the tool.

Relying solely on AABB generates false positives.

Level 2: The 2D Profile Projection (Ray Casting / Shadow)

Panel bending is fundamentally an extrusion process (2.5D). The most effective collision logic projects the 3D flange geometry onto the 2D plane perpendicular to the Bend Axis (the $X - Z$ plane in local bend coordinates).

- **Method:**
  1. Project the Flange Mesh onto the section plane to create a 2D Polygon P_{flange} .
  2. Define the Tool Profile as a static 2D Polygon P_{tool} (incorporating the throat/undercut geometry).
  3. Perform a **Polygon Intersection Test** ($P_{flange} \cap P_{tool}$).
- **Result:** This accurately catches "Throat collisions" where a return bend hooks into the tool, which an AABB check might miss or mischaracterize. It is computationally efficient ($O(V)$ where V is vertices in the profile) and is the industry standard for profile validation.

Level 3: Full Mesh Boolean (Complex Tools)

Full 3D Mesh-Mesh interference (Boolean Intersection) is reserved for cases involving complex ABA configurations where the tool width varies along the bend line (e.g., using "horns" or gaps to clear a flange). If the flange is taller than H_{tool} but narrower than a gap between segments, 2D profile projection fails (it assumes infinite width). The 3D check verifies if the flange fits *between* the solids.

3.3 The "Dead Zone" and ABA Logic

A unique feature of the P4 is the ABA (Automatic Blankholder Adjustment). The tool length is programmable.

- **Kinematic Logic:** If a flange is classified as BEND_UP, the CAM checks if Flange_Width < (Sheet_Edge - Dead_Zone).
- If the flange is located at the corners of the sheet, the ABA can retract to width exactly matching the base face, minimizing collision risk.
- However, if the flange is in the center of the sheet (e.g., a window cutout bend), the tool *must* span across it, making the collision constraints absolute.

4. Algorithmic Implementation

The following sections translate the theoretical physics into concrete pseudo-code and implementation guides for the Python/OpenCASCADE environment.

4.1 Data Structures

We require a robust structure to hold the machine capabilities, often loaded from a machine configuration file (XML/JSON).

C++

```
// Machine constraints based on P4 model (e.g., P4-3126)
struct MachineSpecs {
    float max_positive_bend_height; // e.g., 254.0 mm
    float max_throat_depth;         // e.g., 45.0 mm (Undercut limit)
    float min_negative_clearance;   // e.g., -500.0 mm (Pit depth)
    float tool_segment_gap;         // Minimum width for horn clearance
};

enum BendDirection {
    BEND_UP,      // Positive: Actuate Lower Blade
    BEND_DOWN,    // Negative: Actuate Upper Blade
    BEND_NEUTRAL, // Coplanar/Hem
    BEND_ERROR    // Topology failure
};
```

4.2 Vector Algebra Implementation (Python/OCC Context)

This function determines the directionality. It assumes the BaseFace normal is oriented towards positive Z (Up).

Python

```
def classify_bend_direction(base_face, flange_face):
    """
    Classifies the bend direction relative to the Base Face Normal.
    Returns: BEND_UP, BEND_DOWN, or BEND_NEUTRAL
    """
    # 1. Extract Properties using BRepGProp
    props_base = BRepGProp_Face(base_face)
    props_flange = BRepGProp_Face(flange_face)
```

```
    c_base = props_base.CentreOfMass()
    c_flange = props_flange.CentreOfMass()

    # 2. Extract Normal (Handling Orientation)
    # Get geometric normal from surface
    surf_base = BRep_Tool.Surface(base_face)
    # Assuming planar, get normal at UV(0,0) or center
    geom_n = surf_base.Normal(0, 0)

    # Apply Topological Orientation Correction
    # If Face is REVERSED, the material is on the "other" side
    if base_face.Orientation() == TopAbs_REVERSED:
        n_base = -1 * geom_n
    else:
        n_base = geom_n

    # 3. Calculate Displacement Vector
    v_disp = c_flange - c_base

    # 4. Dot Product Projection (Half-Space Test)
    # Project displacement onto the Normal vector
    d = v_disp.Dot(n_base)

    # 5. Threshold Classification
    EPSILON = 1e-4 # Tolerance for floating point errors

    if d > EPSILON:
        return BEND_UP
    elif d < -EPSILON:
        return BEND_DOWN
    else:
        return BEND_NEUTRAL
```

4.3 Kinematic Validity Logic (IsBendDirectionSafe)

This function implements the "Collision Volume" logic, prioritizing the rigorous check for Positive bends.

```
    Python
```

```
def is_bend_direction_safe(bend_dir, flange_shape, machine_specs):
    """
    Validates if the bend can be performed without collision.
    """

    # CASE 1: NEGATIVE BEND (DOWN)
    # ----------------------------
    # Constraint: Machine Table / Pit Depth
    if bend_dir == BEND_DOWN:
        # Get global bounding box of the flange in its final state
        bbox = get_bounding_box(flange_shape)
        min_z = bbox.MinZ()

        # Check against pit depth (e.g., -500mm)
        if min_z < machine_specs.min_negative_clearance:
            log_collision(f"Flange extends to {min_z}, exceeds pit depth")
            return False

        # Check Manipulator Interference (Simplified)
        # If flange folds down near the grippers (Center of Base Face)
        # This usually requires a geometric distance check against the Clamp Mesh
        return True # Generally Safe

    # CASE 2: POSITIVE BEND (UP)
    # --------------------------
    # Constraint: Blankholder Tool Volume (Height + Throat)
    elif bend_dir == BEND_UP:
        bbox = get_bounding_box(flange_shape)

        # CHECK A: Vertical Height Clearance (AABB)
        # Does the flange hit the tool holder mechanism?
        max_z = bbox.MaxZ()
        if max_z > machine_specs.max_positive_bend_height:
            log_collision(f"Flange height {max_z} exceeds tool clearance
{machine_specs.max_positive_bend_height}")
            # Potential mitigation: Check if flange fits in tool gaps (ABA)
            # if check_gap_clearance(flange_shape, machine_specs): return True
            return False

        # CHECK B: Throat Depth / Undercut (Profile Check)
        # Does the flange curve back and hit the tool face?
        # We need the horizontal distance from the Bend Line to the
        # furthest point of the flange *towards* the Base Face.
```

```
        # 1. Project flange to 2D Profile on XZ plane (perp to bend axis)
        distance_to_bend_line = calculate_intrusion_depth(flange_shape)

        if distance_to_bend_line > machine_specs.max_throat_depth:
            log_collision(f"Flange intrusion {distance_to_bend_line} exceeds throat depth {machine_specs.max_throat_depth}")
            return False

        return True

    return False # Unknown direction
```

4.4 Advanced Simulation: The Swept Volume

While the AABB check on the *final* geometry catches static collisions, the physical reality is a continuous motion. The flange sweeps through an arc.

- **The Apex Path:** The tip of the flange describes a circular arc centered on the bend axis.
- **Collision Condition:** If *any point* on this arc intersects the collision volume, it is a crash.
- **Implementation:** Instead of checking just the final mesh, the software constructs a "Swept Solid" using BRepPrimAPI_MakePrism (linear sweep) or BRepPrimAPI_MakeRevol (rotational sweep) in OpenCASCADE.[8] This swept solid represents the *total volume* occupied by the flange during the bend. The intersection of Swept_Solid AND Tool_Solid must be empty. This is the gold standard for high-fidelity CAM verification.

5. Strategic Integration and Future Considerations

5.1 Material Effects: Springback and Radius

The collision logic must account for material properties. A bend programmed for 90° might require an overbend to 92° or 95° to account for springback (Elastic Recovery), especially in high-tensile steel (UTS 660 N/mm2).[13]

- **Impact:** The "Collision Volume" must be calculated at the **Overbend Angle**, not the nominal angle. A flange might clear the tool at 90° but collide at the 95° required to achieve it. The CAM logic must calculate the *Physical Required Angle* based on the Material Library before running the collision check.

5.2 Sequence Optimization

The IsBendDirectionSafe function returns a boolean. If a bend is unsafe as BEND_UP, the CAM system creates a constraint: **"Must Bend Negative"**.

However, if the part geometry requires a Positive bend (based on the fold pattern), and it

collides, the system must trigger **Automatic Blankholder Adjustment (ABA)**.

- **Logic:** Can we narrow the tool width?
- **Logic:** Can we perform an auxiliary bend (pre-bend) to change the clearance?
- **Logic:** Can we rotate the sheet 180° and perform the bend as a Negative bend from the other side?

This dependency chain transforms the "Safety Check" into a "Process Planner" that actively solves for a valid manufacturing strategy.

6. Conclusion

The reverse-engineering of Salvagnini P4 CAM logic necessitates a rigorous application of vector algebra and computational geometry. By establishing the Base Face Normal as the kinematic Z-axis, we can deterministically classify bends using the Half-Space Dot Product method.

The safety of these operations is defined by the asymmetric volume of the machine. The Blankholder tool imposes strict Height and Throat Depth constraints on Positive bends, necessitating a multi-stage collision detection algorithm that prioritizes 2D profile projection over simple bounding boxes to accurately model undercut clearances.

The pseudo-code and algorithmic strategies detailed in this report provide the blueprint for the "Kinematic Validity Engine," enabling the automated transition from static STEP topology to safe, executable P4 machine code. This ensures that the generated CAM instructions respect the complex physical reality of the panel bending process, preventing costly collisions and enabling autonomous "Batch One" production.

Works cited

1. Panel Bender -P4 - MachineryHost, accessed February 1, 2026, https://f.machineryhost.com/71b2fab0316bfa7fa24ce0a0e5460bad/Brochure%20-%20Panel%20Bender%20-P4.pdf
2. Universal Sheet Metal Panel Bending Machine - Salvagnini, accessed February 1, 2026, https://www.salvagninigroup.com/en-US/products/panel-benders
3. Panel Bending Tools - JEELIX, accessed February 1, 2026, https://www.jeelix.com/panel-bending-tools/
4. Range of automatic panel benders, accessed February 1, 2026, https://irp.cdn-website.com/c8f65e13/files/uploaded/Salvagnini%20P4%20US%20IMPERIAL.pdf
5. Chapter 1 Units and Vectors: Tools for Physics - Tennessee Tech University, accessed February 1, 2026, https://www2.tntech.edu/leap/murdock/books/v1chap1.pdf
6. Elementary Differential Geometry: Curves and Surfaces Martin Raussen - FD intranet, accessed February 1, 2026,

https://intranet.fd.cvut.cz/department/k611/pedagog/K611GM_A_soubory/GMliteratura_soubory/DifGeo_Raussen_160.pdf
7. PLASTICITY CHARACTERIZATION OF SHEET METALS USING TORSIONAL IN-PLANE SHEAR TESTING - UNH Scholars Repository, accessed February 1, 2026, https://scholars.unh.edu/cgi/viewcontent.cgi?article=2954&context=thesis
8. Modeling Algorithms - Open CASCADE Technology, accessed February 1, 2026, https://dev.opencascade.org/doc/overview/html/occt_user_guides__modeling_algos.html
9. High level understanding of OpenCascade Boundary Representation, accessed February 1, 2026, https://dev.opencascade.org/content/high-level-understanding-opencascade-boundary-representation
10. p4-benders.pdf - Empire Machinery, accessed February 1, 2026, https://www.empire-machinery.com/wp-content/uploads/2021/06/p4-benders.pdf
11. Salvagnini P2 User Manual | PDF - Scribd, accessed February 1, 2026, https://www.scribd.com/document/761106673/Salvagnini-P2-User-Manual
12. Bending? Easy, with Salvagnini!, accessed February 1, 2026, https://www.salvagninigroup.com/en-INT/campaigns/panel-benders-range
13. Automatic Sheet Metal Bending Machine - Salvagnini P4, accessed February 1, 2026, https://www.salvagninigroup.com/en-US/products/panel-benders/P4
14. Lean & Compact Panel Bender - Salvagnini P2, accessed February 1, 2026, https://www.salvagninigroup.com/en-US/products/panel-benders/P2
15. Collision detection during planning for sheet metal bending by bounding volume hierarchy approaches | Request PDF - ResearchGate, accessed February 1, 2026, https://www.researchgate.net/publication/324742687_Collision_detection_during_planning_for_sheet_metal_bending_by_bounding_volume_hierarchy_approaches
16. BVH – Knowledge and References - Taylor & Francis, accessed February 1, 2026, https://taylorandfrancis.com/knowledge/Engineering_and_technology/Computer_science/BVH/
17. Sheet metal bending calculation basics - The Fabricator, accessed February 1, 2026, https://www.thefabricator.com/thefabricator/article/bending/sheet-metal-bending-calculation-basics