

BSPPFit Model Library — Overview

v0.145.0.349

Peter Felecan

October 11, 2026

Contents

1	Status	1
2	Architecture	3
2.1	File layout	3
2.1.1	Taxonomy ↔ wrapper mapping	4
2.2	Conventions	4
3	Dependencies	7
3.1	BOSL2 (Belfry OpenSCAD Library v2)	7
3.2	Shared library files	8
4	Sync rule for <code>dimensions.csv</code>	8
5	What this library is not	8
6	Where the rationale lives	9

1 Status

This library is the third Phase 1 deliverable of the **BSPPFit** project. It opened in iteration 0.9.0 as the transition from *taxonomy-only* to *taxonomy + first models*. As of iteration 0.15.0 the library has ten parametric fittings implemented end-to-end — one for each of the ten F-01 geometry classes, with the G-06 CapPlug class carrying two (Cap and Plug): Nipple (G-01 Straight), Elbow (G-02), Tee (G-03), Cross (G-04), Reducer (G-05), Cap and Plug (both G-06 CapPlug), Union (G-07), HexBush (G-08), HoseTail (G-09), and Manifold (G-10). At iteration 0.36.0 the Tee, Cross and Manifold were unified into the single Manifold class (Option B of the branch-fitting taxonomy review); the G-03 Tee and G-04 Cross *model files* are marked deprecated and remain callable for back-compatibility,

but the Manifold module is the model of record for every branch-fitting configuration. *All ten geometry classes now have at least one end-to-end parametric model*; the Phase 1 geometry-class coverage is complete. Variant work — the reducing tee and reducing cross landed in iteration 0.19.0, the Reducer reducing bushing (MxF) in 0.20.0, the Reducer FxF and FxM gender variants in 0.21.0, the female-thread hose tail in 0.22.0, the Union gender variants in 0.23.0, the gender-swapped hex bush in 0.24.0; with 0.24.0 the gender axis is complete. The form axis opened in 0.26.0 with the hexagonal-body reducer, continued in 0.27.0 with the hexagonal-body hose tail and in 0.28.0 with the shouldered hose-tail body-to-barbs transition, and closed in 0.29.0 with the alternative hose-tail barb profiles; with 0.29.0 the form axis is complete. Any later variant work is recorded in [FutureResearch.pdf](#).

Beside these ten production classes sits one *experimental* model, `bsppfit.scad` (since 0.93.0). It is a decorated super-class driven by a *port list*: a fitting is given as a list of ports, each a direction with a gender, size and open/closed termination, and a single code path emits the body — a hub with one threaded or socketed arm per port and a bore network joining the open ones. From that one path fall the cap (one closed port), pipe and elbow and reducer (two ports), tee (three) and cross (four), as well as fittings the dedicated models cannot express (arbitrary port angles, per-port gender and size mixed on one body). It is the working experiment behind the topological question of whether the whole catalogue has a single common ancestor (see [TopologyRigorous.pdf](#)). It is *not* the model of record for any trade category: whether it becomes the library's core (with the dedicated models as named conveniences over it) or remains a parallel exploration is a deliberately deferred architectural decision. What it subsumes cleanly is the catalogue's *topology* — including, since iteration 0.95.0, the multi-branch manifold, whose branch positions are carried as per-port offsets (branch position being a decoration, not topological data). The three decorations it does not carry as port parameters — a hex wrench grip, a hose barb, and a union's detachable three pieces — are solid-wall or assembly features; since iteration 0.98.0 they are supplied by three thin *sub-class* modules (`bsppfit_hexbush`, `bsppfit_hosetail`, `bsppfit_union`) that build the topological body with the same threaded-arm machinery and add the one extra feature. Since iteration 0.102.0 the cap and plug carry their commercial wrench head too, via `bsppfit_cap` and `bsppfit_plug` (hex or round head), so the last decoration the basic closed-port cap/plug lacked is now available — every dedicated model's decoration is reproduced. With these `bsppfit.scad` subsumes all eleven trade categories from one file, decorations included — the mono-file core toward which the library is converging (ideal cardinality one). The dedicated models are retained, for now, as the reference oracle against which the mono-file is tested. Angled branches that lean toward a run end (the Y-fan) have their run automatically extended to keep the run port fittable, via `bsppfit_yfan_ports` and the reach computed in `bsppfit_manifold_ports` (iteration 0.102.0).

Per-iteration history — which fittings were added when, which conventions were ratified or reversed, which bugs were found and fixed — lives in [ResearchJournal.pdf](#) per the [Constitution](#)'s separation of *what it is* (this document) from *how it became this* (the journal).

2 Architecture

The architecture decisions are recorded in [Concept.org §Modelling Approach for the BSPP Model Library](#); per the [Constitution](#) the rationale (alternatives considered, libraries surveyed, trade-offs weighed) lives in [ResearchJournal.pdf](#) and not here.

2.1 File layout

LibraryOverview.org	this file
dimensions.csv	source of truth: BSPP per-size table
bsppfit.scad	the library: a port-list-driven fitting generator (since 0.93.0), together with its helpers --- numeric-size lookup, accessors, tolerance defaults, wall-thickness defaults, self-correcting helpers, cutaway half-cut. One code path emits every catalogue fitting --- nipple, pipe, elbow, reducer, cap, plug, union, hex bush, hose tail, manifold (subsuming tee and cross) --- and fittings beyond the catalogue (arbitrary port directions, per-port gender and size). A pure library (definitions only, no top-level geometry), safe to `include` into a host project (e.g. alongside BOSL2).
bsppfitTest.scad	exercise / render driver for the above: catalogue and usage galleries, the named-variant render dispatch, and the default test grid. `include`s bsppfit.scad; the render targets point here.
Renders/	per-model render output (PNG / tar /
-- Elbow/	MP4); created on demand by the
-- Nipple/	render targets, one subdirectory
-- ...	per model

Every project file lives at the project root, distinguished by suffix rather than by directory, since the whole source tree was flattened at iteration 0.136.0 (see that ResearchJournal entry for why); there is no longer a separate **Models/** or **Util/** subdirectory to describe.

Before iteration 0.143.0, this library was ten dedicated per-fitting files (**Nipple.scad**, **Pipe.scad**, **Elbow.scad**, **Reducer.scad**, **Cap.scad**, **Plug.scad**, **Union.scad**, **HexBush.scad**, **HoseTail.scad**, **Manifold.scad**) plus the **bsppfit.scad** mono-file wrapper, with the dedicated files kept specifically as the **equivalence-test** oracle the wrapper was checked against.

Once every topology’s wrapper was confirmed equivalent to its dedicated model (see the 2026-09-21 ResearchJournal entry), the dedicated files were retired: the comparison had nothing left to prove, and `bsppfit.scad` is now the library’s sole implementation, not an experimental alternative to one.

2.1.1 Taxonomy ↔ wrapper mapping

The taxonomy’s geometry facet catalogue is implemented entirely within `bsppfit.scad`, one named module per facet. Gaps in the G-numbering (G-03 Tee, G-04 Cross) correspond to taxonomy slots that are configurations of `bsppfit_manifold()` rather than distinct geometries (see the 0.36.0 unification).

G-facet	trade name	bsppfit.scad module	notes
G-01	Straight	<code>bsppfit_nipple()</code> , <code>bsppfit_pipe()</code>	nipple (MxM), adapter (MxF), socket (FxF); pipe: a length of pipe, gendered ports
G-02	Elbow	<code>bsppfit_elbow()</code>	90 degrees, per-port gender
G-03	Tee	<code>(bsppfit_manifold())</code>	one-sided 2-branch manifold
G-04	Cross	<code>(bsppfit_manifold())</code>	two-sided 1-branch-each manifold
G-05	Reducer	<code>bsppfit_reducer()</code>	concentric or eccentric
G-06	Cap / Plug	<code>bsppfit_cap()</code> , <code>bsppfit_plug()</code>	the one G-facet with two modules — a cap-and-plug catalogue pair
G-07	Union	<code>bsppfit_union()</code>	three-piece detachable joint
G-08	Hex bush	<code>bsppfit_hexbush()</code>	opposite-gender reducer
G-09	Hose tail	<code>bsppfit_hosetail()</code>	threaded one end, barbed the other
G-10	Manifold	<code>bsppfit_manifold()</code>	unified branch-fitting class

The mapping is intuitive: each module’s name names its fitting category. The G-numbers are documentation, not a navigational requirement; they appear in `Taxonomy.org` and as references in research-journal entries, but no workflow needs to recover them from the source.

2.2 Conventions

- *Pipe semantics with cap/plug exception* (added 0.9.2, since the 0.9.1 **Nipple** was a threaded rod with no fluid path): every BSPP fitting must have a *defined fluid behaviour*. For *pass-through* fittings (G-01 straight, G-02 elbow, G-03 tee, G-04 cross, G-05 reducer, G-07 union, G-08 hex bushing) this is a through-bore from one port to each other port, since the second P in BSPP stands for *Pipe*. For *terminating* fittings (G-06 cap and plug) this is the *absence* of a through-bore: their role is to close a port. For *transitional* fittings (G-09 hose tail) this is a through-bore on the threaded side that opens into the barb-side ID. Models that omit the through-bore for visualisation purposes only (e.g. via `bore=0` on a Nipple) must mark themselves clearly as *not real fittings* in their parameter documentation.

- *Bore heuristic verification per fitting* (added 0.9.2): `bsppfit.scad`'s `bsppfit_default_bore(size)` returns `0.7 * minor_od`, a commercial-fitting heuristic that matches brass G1/2 nipples in the supplier-catalogue mining of 0.8.8. Each new fitting must verify the heuristic against trade catalogues for that fitting's commercial range, and adapt where necessary. Reasons the factor may differ for other fittings: elbows have flow-direction changes; reducers have two different bores by definition; hose tails have a barb-side ID that depends on hose ID, not thread minor diameter; caps and plugs have no bore. The 0.7 factor is the Nipple default; it is *not* a universal rule.
- *Wall-thickness convention* (added 0.9.7): `bsppfit.scad` defines `BSPPFIT_MIN_WALL = 1.5 mm`, the default minimum thickness of material between an internal void (thread cavity, through-bore, port opening) and the outer surface, or between two adjacent internal voids. The 0.9.7 helper `bsppfit_safe_wall(~requested_od, thread_od, min_wall = BSPPFIT_MIN_WALL)~` returns the larger of `requested_od` and `thread_od + 2*min_wall`; the companion `bsppfit_warn_wall_raised` module echoes a warning whenever the helper raises a user-supplied value. Rationale: at 0.4 mm FDM nozzle, three extrusion passes give 1.2 mm; 1.5 mm adds a safety margin. SLA printing tolerates ~0.6 mm; users can override at the call site for non-FDM workflows.

Per-fitting overrides recorded for the cases that have surfaced so far:

- *Nipple* uses `min_wall = 0.5` in `bsppfit_safe_middle_od` because the through-bore is small relative to the major OD and the wall in the threaded region is naturally several mm. The 0.5 is a safety floor on the `middle_od` raise, not the actual wall thickness.
- *Cap, Plug* use `min_wall = 1.5` matching `BSPPFIT_MIN_WALL` because the internal thread cavity (Cap) or the head-vs-thread seam (Plug) is the structurally tight feature.
- *HexBush* uses `BSPPFIT_MIN_WALL` on the `hex_od` computation against the LARGER thread's `major_od` (the binding constraint when `size_male >= size_female`).
- *Manifold-clean default with opt-out* (added 0.9.2): each model exposes a `manifold_safe` parameter that defaults to `true`. When `true`, geometry choices prioritise reliability under OpenSCAD's Manifold backend over visual authenticity — e.g. for the nipple wrapper the threaded ends use `blunt_start=false` which produces simpler bevel-ended threads. When `false`, the more authentic default-BOSL2 thread terminations are used at the cost of possible `PolySet -> Manifold conversion failed: NotManifold` warnings (the Manifold backend's repair pass usually succeeds, but the geometry is fragile). The default is conservative; users who want the prettier render can opt in case-by-case.
- *OpenSCAD code style* (refined in 0.9.3, see also [DevelopmentStandard.org §BSPP Model Library — OpenSCAD Style Addenda](https://developmentstandard.org/BSPP-Model-Library-OpenSCAD-Style-Addenda/)):

- *Allman braces, everywhere.* Every compound statement uses curly braces in Allman style (opening brace alone on its own line, aligned under the introducing statement). No exceptions: single-child modifier chains (`translate`, `rotate`, `scale` etc.) also enclose their child in braces. The 0.9.2 addendum that exempted single-child modifier chains is reverted in 0.9.3 per project-author direction in favour of a uniform rule.
- *Hybrid identifier rule.* Public-facing identifiers (module names callable from outside the library, exported function names, module parameters, top-level configuration constants) use `snake_case` (or `UPPERCASE_SNAKE_CASE` for constants, `PascalCase` for modules); this matches BOSL2's convention so library code reads cleanly alongside BOSL2 calls. Internal identifiers (local variables, private modules/functions prefixed `_`, internal-only constants, loop iterators) use Hungarian notation per the project's standing rule (`g~/~l~/~p~/~i` scope prefixes).
- *Spaces around operators, including = in named arguments.* `cyl(d = 10, h = 20)` rather than `cyl(d=10, h=20)`. Same rule applies to function and module parameter defaults: `factor = BSPPFIT_DEFAULT_BORE_FACTOR`.
- *Parameter defaults that depend on other parameters use `undef` + ternary in the body, not the native parameter-list form.* The 0.9.3 attempt to use `thread_length = bsppfit_useful_thread_length(size)` in the parameter list fails at runtime: OpenSCAD does NOT bind earlier parameters for use in later default expressions. Use `thread_length = undef` in the parameter list and `is_undef(thread_length) ? bsppfit_useful_thread_length(size) : thread_length` in the body. Defaults that are literals or top-level constants (e.g. `tol_thread = BSPPFIT_TOL_THREAD`, `middle_shape = "hex"`) still use the native parameter-list form. The choice is per-parameter.
- *Wrapper + underscore-implementation* pair per file: `Nipple()` is the public API; `_nipple()` takes resolved-and-validated parameters and is the implementation. Callers should never invoke `_nipple()` directly because parameter resolution may evolve.
- *Numeric size parameter:* the primary `size` parameter is a fractional-inch number (`1/8`, `1/2`, `1+1/4`, `2`, etc.), matching BOSL2's `bspp_threaded_rod()` convention. The "G..."-prefixed string form (e.g. `G1/2`) is kept as the human-readable *designator* in `dimensions.csv` and is returned by `bsppfit_designator()` for warning messages and gallery labels — but it is *not* accepted as a model-call parameter as of 0.9.1. Rationale: the "G" prefix is redundant inside a BSPP- only library, and the numeric form composes more cleanly with BOSL2.
- *BOSL2 attachment system:* all public modules accept `anchor`, `spin`, `orient` and pass them to BOSL2's `attachable()`. Models compose with BOSL2's positioning primitives without manual coordinate tracking.
- *Multi-named tolerances:* `BSPPFIT_TOL_THREAD`, `BSPPFIT_TOL_GASKET`, `BSPPFIT_TOL_FACE`,

`BSPPFIT_TOL_PIPE` as separate constants in `bsppfit.scad`. BOSL2's `$slop` remains the printer-specific global handle and is orthogonal.

- *Male-thread clearance*: `BSPPFIT_UNDSIZE` (default 0) in `bsppfit.scad`, exposed as an `undsize` parameter on every public module that has a male thread (all but `Cap`, which is female-only). It is BOSL2's own thread-diameter reduction — mm off the diameter, pitch and thread depth unchanged — applied to the male threads only and handed straight to BOSL2's `bspp_threaded_rod()`. It is the male counterpart of BOSL2's `$slop`, which acts on internal threads only, for heterogeneous fits such as a printed male into a rigid metal female. It is *not* wired to `tol_thread`: the two are distinct physical allowances (the female bore opening versus the male diameter shrinking). Default 0 leaves the male thread nominal; bench-measured values are in the DevelopmentStandard Print Tuning Log (PETG male into brass G1/2 female: 0.10 mm). Until iteration 0.144.0 this was a radial parameter named `male_slop`, half the value.
- *Self-correcting middle-diameter*: an undersized `middle_od` is silently raised to the safe minimum and a warning is `=echo()`ed. Per the 0.8.3 architecture decision (Makrokaba pattern); the alternative was an assertion failure that the user has to debug.
- *Test-area toggle* at the bottom of each model file: comment-toggle pattern from doommeister, lets the file be opened in OpenSCAD with a default preview without adding a separate test driver.
- *Gallery module* per file: `Nipple_Gallery()` renders all supported sizes side by side, used to produce the per-fitting reference render.
- *Render-dispatch module* per file (added 0.18.0): `<Model>_Render(variant)` maps a string variant name to the module call that produces it, and is the file's active top-level statement. The variant is read from a top-level `render` knob that OpenSCAD's `-D` option sets, so the render pipeline can target a named variation — the Union's exploded view, any model's gallery — without a per-variant copy of the file. Every dispatched variant recognises at least `"default"` and `"gallery"`; the union variants also include `"exploded"`, `"tail-a"`, `"tail-b"`, `"nut"`, `"mxf"`, `"fxf"`. An unknown variant fails loudly via an assert. See [DevelopmentStandard.org §Render Dispatch Module](#) and [§Render variant selection](#).

3 Dependencies

3.1 BOSL2 (Belfry OpenSCAD Library v2)

The library depends on BOSL2 for the threading layer (`bspp_threaded_rod`, `npt_threaded_rod`) and for the attachment system. Pin policy and install procedure are recorded in [Develop-](#)

[mentStandard.pdf](#) §BOSL2 BSPP Module.

To open any model file in OpenSCAD, BOSL2's `std.scad` and `threading.scad` must be on the include path. The model files declare their own dependencies via `include` statements at the top per BOSL2 convention (specialised modules included separately).

3.2 Shared library files

`bsppfit.scad` is a single self-contained file. It includes BOSL2 (`std.scad` and `threading.scad`) and carries its general helpers in the same file (the size tables and lookups, tolerance defaults, wall-thickness guards and the cutaway half-cut), so a host project needs a single `include <bsppfit.scad>`:

```
include <BOSL2/std.scad>
include <BOSL2/threading.scad>
bosl_required([2, 0, 767]);
```

Every thread site calls BOSL2's own `bspp_threaded_rod()`; the male-thread clearance (`undersize`) reaches it as the parameter of the same name, which is why BOSL2 2.0.767 or newer is required and enforced at load time. Until iteration 0.144.0 the library carried a diverted copy of that module, `bsppfit_threading.scad`; it was retired once BOSL2 gained the parameter (see [ResearchJournal](#), [iteration 0.144.0](#)).

4 Sync rule for `dimensions.csv`

`dimensions.csv` is the canonical record. The `BSPP_TABLE` constant in `bsppfit.scad` mirrors it manually. When the CSV changes, `BSPP_TABLE` must be updated to match.

A generator script (`csv2scad.py` or similar) is queued as a TODO in [FutureResearch.pdf](#); until then, manual sync is the practice.

5 What this library is not

- *Not a general pipe-modelling library.* The scope is BSPP fittings; BSPT and NPT are documented in the taxonomy as adjacent classes for confusion-avoidance, not modelled here.
- *Not a CAD package.* The output is parametric OpenSCAD geometry suitable for 3D printing or visualisation, not engineering analysis.
- *Not a wrapper around BOSL2.* The library composes with BOSL2; it is not a parallel implementation of the same threading geometry. If integration upstream into BOSL2 becomes settled, the library shape is compatible (per the 0.8.5 architectural goal).

6 Where the rationale lives

Every architectural decision in this library has a one-line rationale in `Concept.org`'s table and a detailed entry (alternatives, sources, decision-day context) in `ResearchJournal.org`. Readers who want *what* should consult Concept; readers who want *why* should follow the journal cross-references.