Vesicle Simulation Framework

Overview

This project simulates two-phase vesicle shapes governed by curvature elasticity and volume/area constraints.

Each solution is obtained by solving a coupled two-region boundary-value problem (BVP) in the Laplace–Beltrami form using MATLAB's bvp6c solver.

The simulation framework explores a 2-D parameter space in spontaneous curvatures $\left(H_0^{(1)}, H_0^{(2)}\right)$ while keeping global physical parameters fixed for each run (area ratio, moduli, etc.).

Solutions are stored in a content-addressed database (SimResults/) and organized through a persistent catalog (catalog.mat).

High-Level Design Components

User script (driver)

Defines a simulation configuration and launches a task driver.

Key Files: run_initial_sweep.m

Simulation configuration

Specifies physical parameters (`sim.MP`), thresholds (`sim.TH`), and solver settings (`sim.SP`).

Key Files: `sim_config()`

Task driver

Performs adaptive quadtree exploration in H_0 space, coordinating catalog I/O and solver calls.

Key Files: `src/sim_explore_H0_quad_tree.m`

Bootstrap & cleanup

Initialize and teardown environment, import seeds, manage paths.

Key Files: `bootstrap.m`, `cleanup.m`

Solver

Computes a solution at a single parameter point. Performs warm-starts, continuation, and quality gating.

Key Files: `src/utils/solveAtParams.m`

Continuation

Smoothly walks a prior solution toward a new target H_0.

Key Files: `src/utils/continuation_towards_H0.m`

Scheduler (quadtree)

Chooses which cell and parameter point to solve next.

Key Files: `src/utils/processQuadtree.m`

Catalog utilities

Manage a persistent MAT-based catalog of all results.

Key Files: `catalog_load.m`, `catalog_append.m`, `catalog_save.m`, `catalog_view_wide.m`

Seed importer

Registers initial shape files from `InitialShapes/` as seeds for warm starts.

Key Files: `bootstrap.m` → `import_initial_shapes_into_catalog()`

Warm-start selector

Finds nearest solved or seeded shape to initialize new solves.

Key Files: `src/utils/pickWarmStart.m`

Execution Flow

1. Environment setup

Running:

```
bootstrap();
```

- Resets MATLAB's path, adds `src/`, `bvp6c-solver/`, and `InitialShapes/`.
- Ensures `SimResults/` exists.
- Imports any new `InitialShapes/SIM_Node_*.mat` files into the catalog as seed entries.

2. Simulation configuration

The user config function (`sim_config`) defines three nested structures:

```
SP: Simulation parameters
MaxIters, ModelVersion, LogToFile

TH: Thresholds & solver knobs
BCmax, DEmaxHard, rMin, delta, opts

MP: Physical parameters
A, V, KA, KB, KG, (→ aS, bS via computePhaseScales)
```

Each simulation run keeps `MP` fixed while exploring $\left(H_0^{(1)}, H_0^{(2)}\right)$.

3. Task driver

`sim_explore_H0_quad_tree` orchestrates the run:

Initialize or resume from previous cache (`cache.mat`) and catalog (`catalog.mat`).

Quadtree scheduling:

- `processQuadtree` proposes the next $\left(H_0^{(1)}, H_0^{(2)}\right)$ to solve based on solved corners.

Hash generation:

- A SHA-256 hash computed from
- `{ModelVersion, H0_1, H0_2, A, V, KA, KB, KG}`
- defines a unique key and file name in `SimResults/hashed_results/`.

Solve or skip:

- If `<hash>.mat` exists → skip (already solved).
- Otherwise → call `solveAtParams`.

Save result:

- Save to `hashed_results/<hash>.mat`.
- Append to `catalog.mat`.
- Update `cache.mat` to track quadtree progress.
- Repeat until all cells are solved or `MaxIters` reached.

4. Solver stage

`solveAtParams(params, sim, warm)`:

- Builds the parameter bundle:

```
Par = {H0, A, V, KA, KB, KG, aS, bS, delta}
```

- Uses warm start from `pickWarmStart` if available.
- Optionally applies `continuation_towards_H0` to precondition the solution.
- Executes an attempt ladder of increasing tolerance looseness and delta scaling: Baseline → δ/2 → δ/4 → looser tolerances.
- Accepts a solution if all gates pass:

```
BCmax ≤ TH.BCmax
DEmax ≤ TH.DEmaxHard
rMin  ≥ TH.rMin
```

- Records metadata `{label, E, P, BCmax, DEmax, mesh}` and returns to the driver.

5. Catalog management

All solved data is tracked in `SimResults/catalog.mat` (MAT-only):

Column	Type	Description
`hash`	string	SHA-256 key
`timestamp`	datetime(UTC)	Creation/update time
`entry`	cell → struct	Contains: • `params`: full parameter set (H0 + physics) • `meta`: metrics & provenance

Seeds added by `bootstrap` have `entry.meta.type = "seed"`.

All later solves have numerical results and metrics.

Warm-Start Hierarchy

When solving at a new H0 point, `pickWarmStart` searches:

1. **Nearest solved neighbor** with identical physics in the catalog. Loads its `.mat` file and returns its BVP structure as `warm.result.sol`.
2. **Seed entry** from `InitialShapes/` matching `(A,V,KA,KB,KG)`. Loads its `Version(1).Solution` for use as the starting state.
3. **Fallback analytic/legacy seed** via `initialGuessFromFile`.

If the warm-start's H0 differs significantly from the target (>0.15 units), `solveAtParams` calls `continuation_towards_H0` to bridge the gap in small steps.

Content-Addressed Storage Layout

```
SimResults/
├─ hashed_results/
│   ├─ <hash>.mat          % result, meta structs
├─ cache.mat               % quadtree & scheduler state
├─ catalog.mat             % persistent catalog table
├─ OPfile.txt              % optional run log
```

All results are immutable: rerunning the solver with identical parameters produces the same hash and simply updates the catalog timestamp.

Adaptive Quadtree Exploration

The quadtree scheduler partitions the H_0 plane into cells.

Each cell stores corner solutions (if solved) and tests uniformity based on energy and pressure:

```
1. Pop a cell from the queue.
2. For each corner, check if its parameters exist in the catalog (using lookup_in_catalog).
3. If a corner is unsolved → return it as a new task.
4. Once all corners are solved, run uniformTest:
    * If the solution is uniform → finalize cell.
    * Otherwise → subdivide into four children and enqueue.
```

This process continues until the target resolution or MaxIters is reached.

Physics Summary

Symbol	Meaning	Appears in
A	area ratio (α/β phases)	`computePhaseScales`, `sim.MP`
V	volume fraction	`BendV_Lag_EIGp_BC_impl`
$K^{(1)}, K^{(2)}, K_G$	elastic moduli	DE/BC implementation
$\left(H_0^{(1)}, H_0^{(2)}\right)$	spontaneous curvatures	exploration plane
a_S, b_S	arc-length scalings (from A)	DE equations
δ	small buffer near poles for Taylor expansion	DE equations
`opts`	solver tolerances	numerical controls only

Catalog Entry Example

```
entry.params =
    struct with fields:
        H0_1: 0.5
        H0_2: -0.3
        A: 0.75
        V: 0.72
        KA: 1
        KB: 1
        KG: 0
        aS: 0.75/pi
        bS: (0.75 - 1)*pi

entry.meta =
    struct with fields:
        label: 2
        E: 3.147
        P: 0.061
        BCmax: 2.8e-7
        DEmax: 1.9e-2
        mesh: 167
        version: "BVP-v3.1"
        hash: "<SHA256>"
```

Run Lifecycle

Initialize environment

```
bootstrap();
```

Run simulation

```
sim = sim_config();
sim_explore_H0_quad_tree(sim);
```

Cleanup

```
cleanup();
```

Inspect results

```
T = catalog_load('SimResults');
W = catalog_view_wide('SimResults');
```

Key Advantages of the Refactor

- **Single Source of Truth:** All results in one unified MAT catalog; no CSV duplicates.
- **Idempotent Execution:** Content-hashing prevents re-solving existing parameter points.
- **Flexible Configuration:** Physics and numerical settings are cleanly separated.
- **Warm-Start Intelligence:** Automatic reuse of nearest solved or seed shapes; continuation ensures stability far from seeds.
- **Adaptive Refinement:** Quadtree exploration automatically concentrates computation in regions of high variation.
- **Modular Design:** Each major function is standalone and easily testable.

File Dependency Diagram

```
run_initial_sweep.m
├─ bootstrap.m
│   └─ import_initial_shapes_into_catalog.m
├─ sim_config()
├─ sim_explore_H0_quad_tree.m
│   ├─ processQuadtree.m
│   │   └─ lookup_in_catalog() …
│   ├─ pickWarmStart.m
│   ├─ solveAtParams.m
│   │   ├─ continuation_towards_H0.m
│   │   ├─ BendV_Lag_EIGp_DE_impl.m
│   │   ├─ BendV_Lag_EIGp_BC_impl.m
│   │   ├─ bc_diagnostics.m
│   │   ├─ de_residual.m
│   │   └─ labelFromSolution.m
│   ├─ catalog_append.m / catalog_load.m / catalog_save.m
│   └─ computePhaseScales.m
└─ cleanup.m
```

Extensibility

Future enhancements can include:

- **Parallel scheduling** for independent cells.
- **Phase-aware continuation** (bias by label).
- **Result-based pruning** (stop refining when energy variation < ε).
- **JSON/CSV export** for external analysis.
- **Visualization tools** for the quadtree and phase diagram.