Documentation: Integrating SASHIMI-SI with ArgoLOOM

Internal Development Note

February 8, 2026

Purpose of this Document

This note documents a **successful end-to-end integration** of the cosmological subhalo simulation code SASHIMI-SI with the orchestration framework ArgoLOOM.

- The goal is reproducibility for collaborators.
- This is a **success-case guide**: it shows what worked.
- It provides both minimal steps (to get running) and explanatory context.

Reading modes:

- **Fast setup:** Follow code blocks only.
- **More details:** Read explanatory text.

—

1 Assumptions

We assume:

- SASHIMI-SI runs correctly standalone.
- ArgoLOOM runs correctly standalone.
- Python version tested: **3.11**. Other versions were not tested.

—

2 Environment Setup

2.1 Conda Environment

```
conda create -n sashimi-si python=3.11 -y
conda activate sashimi-si
```

—

2.2 Core Dependencies

```
pip install numpy==1.26.4
pip install scipy==1.10.1
pip install numexpr
pip install tqdm
pip install matplotlib
pip install jupyter
pip install openai
pip install sentence-transformers

# FAISS (may require conda-forge)
conda install -c conda-forge faiss-cpu -y
```

Notes:

- Mixed `pip` + `conda` install was used successfully.
- FAISS installation can vary by system.
- Python 3.11 compatibility was verified; other versions not tested.

—

3 Directory Structure and Code Edits

A representative working layout on the local machine is shown below. This is provided as a guideline for organizing the integration between `SASHIMI-SI` and `ArgoLOOM`.

```
/Users/<username>/Documents/

        cosmowork/
                sashimi_si_project/
                        argoloom_sashimi_adapter.py
                        repo/sashimi-si/
                        runs/

        GitHub/
                ArgoLOOM/
                        argo-loom.py
                        tools/cosmo/
                           API_SASHIMI_subhalos.py
```

Replace `<username>` with your machine username where necessary. Absolute paths in scripts should be updated accordingly.

Repository Edits

All edits made to link `SASHIMI-SI` with `ArgoLOOM` are available in the public repository:

https://github.com/sdbakshi13/ArgoLOOM/tree/main

Only the following components were modified or added:

- `argo-loom.py`
  - Addition of the `sashimi_subhalos` tool schema.
  - Tool execution block for dispatching SASHIMI runs.
- `tools/cosmo/API_SASHIMI_subhalos.py`
  - New tool wrapper to interface ArgoLOOM with the SASHIMI adapter.
  - Handles parameter passing and JSON output formatting.

No other core ArgoLOOM modules were modified. This ensures minimal intrusion into the base orchestration framework.

—

4 Conceptual Architecture (Skippable)

Skip if you only want setup instructions.

Pipeline flow:

ArgoLOOM → Tool Call → SASHIMI Adapter → Catalog → Summary JSON

Artifacts produced:

- `catalog_raw.npz`
- `catalog_SIDM_survivors.npz`
- `summary.json`

—

5 SASHIMI Adapter

Location:

```
cosmowork/sashimi_si_project/argoloom_sashimi_adapter.
    py
```

Purpose:

- Runs SASHIMI.
- Filters surviving subhalos.
- Writes catalogs + JSON summary.
- Prints JSON to stdout for ArgoLOOM ingestion.

(Use the working adapter script from development.)

—

6 ArgoLOOM Tool Wrapper

Create:

```
ArgoLOOM/tools/cosmo/API_SASHIMI_subhalos.py
```

This script:

- Reads parameters from env variable: `ARGOLOOM_SASHIMI_PARAMS_JSON`
- Calls the adapter.
- Returns JSON output.

—

7 Edits to `argo-loom.py`

7.1 Add Tool Schema

Add to `build_tools_schema()`:

```
{
        "type": "function",
        "function": {
                "name": "sashimi_subhalos",
                "description": "Run SASHIMI-SI subhalo
                   catalog generation.",
                "parameters": {
                        "type": "object",
                        "properties": {
                                "M0": {"type":"number
                                   "},
                                "redshift":{"type":"
                                   number"},
                                "logmamin":{"type":"
                                   integer"},
                                "sigma0_m":{"type":"
                                   number"},
                                "w":{"type":"number"}
                        },
                        "required":["M0","sigma0_m","w
                           "]
}}}
```

—

7.2 Add Tool Execution Block

Inside `# ---- EXECUTE TOOLS ----`:

```
elif fname == "sashimi_subhalos":
try:
result = run_tool_sashimi_subhalos(fargs)
except Exception as e:
result = {"error": str(e)}

messages.append({
        "role": "tool",
```

```
                "tool_call_id": tc.id,
                "name": "sashimi_subhalos",
                "content": json.dumps(result),
            })
```

—

8 Running the Agent

```
conda activate sashimi-si

cd /Users/<username>/Documents/GitHub/ArgoLOOM

python argo-loom.py --model gpt-4o
```

Example prompt:

```
Run sashimi subhalos for host mass 1e12 Msun at z=0
with sigma0_m=147.1 and w=24.33.
```

—

9 Expected Output

- Tool call executed.
- SASHIMI progress printed.
- Agent returns summary:

```
N_filtered = 1353384
Quantiles [1,10,50,90,99] = [...]
```

—

10 Cross-Checks

```
python - <<'PY'
import json
p=".../summary.json"
print(json.load(open(p)))
PY
```

—

11 Troubleshooting

Common issues encountered:

- `ModuleNotFoundError: numexpr`
- Missing `scipy.integrate.simps`
- Missing `openai`

- Missing `sentence_transformers`
- FAISS installation issues
- API quota errors

> **Practical advice:** Copy the error into ChatGPT and ask for resolution. This was highly effective during development.

—

12 Conclusion

We have documented a successful integration of:

- Cosmological SIDM simulation (`SASHIMI-SI`)
- Agentic orchestration framework (`ArgoLOOM`)

This enables automated generation, filtering, and reasoning over subhalo populations within the ArgoLOOM pipeline.

—