

Verification and Validation Report: SynthEddy

Phil Du (Software)
Nikita Holyev (Theory)
Dr. Spencer Smith (Supervisor)

July 10, 2026

1 Revision History

Date	Version	Notes
2024-04-15	1.0	Initial Commit

2 Symbols, Abbreviations and Acronyms

symbol	description
CFD	Computational Fluid Dynamics
DNS	Direct Numerical Simulation
LES	Large Eddy Simulation
MG	Module Guide
MIS	Module Interface Specification
NFR	Nonfunctional Requirement
SRS	Software Requirements Specification
VnV	Verification and Validation

Contents

1	Revision History	i
2	Symbols, Abbreviations and Acronyms	ii
3	Functional Requirements Evaluation	1
3.1	Verify Input [R1]	1
3.2	Generate Velocity Field [R2]	1
3.3	Interface with CFD Software [R3]	1
4	Nonfunctional Requirements Evaluation	1
4.1	Accuracy [NFR1]	1
4.2	Usability [NFR2]	1
4.3	Maintainability [NFR3]	2
4.4	Portability [NFR4]	2
4.5	Performance [NFR5]	2
5	Unit Testing	2
6	Changes Due to Testing	2
7	Automated Testing	3
8	Trace to Requirements	3
9	Trace to Modules	3
10	Code Coverage Metrics	4

List of Tables

1	Traceability Between Test Cases and Requirements, X indicated test passed and O indicates test designed but not yet performed	4
---	---	---

List of Figures

1	Coverage report	5
---	---------------------------	---

This document summarized the result of the verification and validation (VnV) process, as detailed in the [VnV Plan](#). The requirements that were being tested are detailed in the [SRS](#).

3 Functional Requirements Evaluation

3.1 Verify Input [R1]

This requirement is covered by system test in section 4.1.1 of the VnV Plan. The test cases are designed to test the input validation and error handling. All test cases are passed. Test file: [test_system_input.py](#) [VnV Plan 4.1.1]

3.2 Generate Velocity Field [R2]

This requirement is covered by system test in section 4.1.2 to 4.1.4 of the VnV Plan. The test cases are designed to test the calculation of meshgrid velocities in a field and test for the properties of the results. All test cases are passed. Test file: [test_system_eddy.py](#), [test_system_field.py](#) [VnV Plan 4.1.2, 4.1.3, 4.1.4]

3.3 Interface with CFD Software [R3]

This requirement cannot be tested at current stage of development. This functionality is not implemented yet.

4 Nonfunctional Requirements Evaluation

4.1 Accuracy [NFR1]

This requirement cannot be tested at current stage of development. Providing realistic eddy profile is not implemented yet. [VnV Plan 4.2.1]

4.2 Usability [NFR2]

This requirement is not yet tested as it shares implementation with R3.

4.3 Maintainability [NFR3]

The test for user to add a new shape function [VnV Plan 4.2.2] is not yet performed, but assuming it is, the result shall be presented as:

- test subjects: x
- average time taken: x minutes
- percentage that still passed all tests: x%

4.4 Portability [NFR4]

This requirement is tested by GitHub Actions running SynthEddy on Linux, MacOS, and Windows runner. All tests are passed for the current version. Test results can be found on the [GitHub Actions page](#). [VnV Plan 4.2.3]

4.5 Performance [NFR5]

With inputs specified in Section 4.2.4 of the VnV Plan (10 million eddies in a 1000^3 meshgrid), the program requires 24 GB of memory with estimated runtime of 50 minutes on an Intel i9-13900K processor.

Discussions with Nikita Holyev about the expected use case revealed that the original requirement of “minimal” memory usage is not feasible, due to the sheer size of the meshgrid in today’s DNS/LES CFD simulations.

5 Unit Testing

All unit tests covering every module developed for SynthEddy are passed. The coverage report is shown in Figure 1. The test results can be found on the [GitHub Actions page](#).

6 Changes Due to Testing

Other than bug fixes and minor improvements, the most significant change is direct result of performance testing and discussion with Nikita Holyev about potential real world use cases.

In the original MG and MIS design, each eddy is an abstract data type object created by the Eddy Module. The object stores eddy properties and methods to calculate velocities around it. This design is too slow for large scale use cases with many eddies (e.g. 10 million) and large meshgrid (e.g. 1000^3).

In the revised design, the Eddy Module is reduced to a library containing functions to calculate eddy velocity influences. The actual properties of all eddies are consolidated into the Flow Field Module as several arrays. This design allows NumPy to broadcast the calculation, greatly improving computing speed, with the trade-off of much increased memory usage.

To mitigate excessive memory usage and cut down unnecessary calculations between far away positions, a chunking approach is implemented to divide the entire meshgrid into smaller chunks, and calculate the velocity field for each chunk separately.

The combination of these changes allows SynthEddy to be usable in on regular personal computers with reasonable runtime and memory usage.

7 Automated Testing

GitHub Action is used for Continuous Integration (CI). System and unit tests (made with pytest) are run whenever a pull request is made to the main branch. The command used for testing is as follows, which runs all test that are not designated as “slow” (those that take too long to finish), and checks if the coverage is above 95%:

```
pytest --cov=src --cov-fail-under=95 -m "(unit or system) and not slow"
```

8 Trace to Requirements

Please see Table 1 for the traceability between system test cases and requirements. Note some tests are not yet performed.

9 Trace to Modules

All unit test are passed, which covers all modules. Please see VnV Plan Section 5.2 for the traceability between test cases and modules.

	R1	R2	R3	NFR1	NFR2	NFR3	NFR4	NFR5
VnV Plan 4.1.1	X							
VnV Plan 4.1.2		X						
VnV Plan 4.1.3		X						
VnV Plan 4.1.4		X						
VnV Plan 4.1.5			O		O			
VnV Plan 4.2.1				O				
VnV Plan 4.2.2						O		
VnV Plan 4.2.3							X	
VnV Plan 4.2.4								X

Table 1: Traceability Between Test Cases and Requirements, X indicated test passed and O indicates test designed but not yet performed

10 Code Coverage Metrics

Coverage is tested with `pytest-cov`, using the command shown in Section 7 that executes unit and system tests. An overall coverage of 98% is achieved, with coverage report is shown in Figure 1.

The only two modules that are not fully covered are `main` and `query`. The not covered lines are either related to exception handling with those exceptions already tested in lower level modules, or impossible to reach in automated testing (e.g. `plot pop-up`).

Coverage report: 98%				
coverage.py v7.4.4, created at 2024-04-14 16:14 -0400				
Module	statements	missing	excluded	coverage
src\main.py	59	2	0	97%
src\modules\eddy.py	15	0	2	100%
src\modules\eddy_profile.py	41	0	0	100%
src\modules\file_io.py	49	0	0	100%
src\modules\flow_field.py	186	0	4	100%
src\modules\query.py	76	6	0	92%
src\modules\shape_function.py	26	0	0	100%
src\modules\utils.py	17	0	0	100%
src\modules\visualize.py	27	0	0	100%
Total	496	8	6	98%
coverage.py v7.4.4, created at 2024-04-14 16:14 -0400				

Figure 1: Coverage report