Verification and Validation Report for SCEC (Solar Cooker Energy Calculator)

Deesha Patel

April 19, 2023

1 Revision History

Date	Version	Notes
April 15, 2023	1.0	Initial Version

2 Symbols, Abbreviations and Acronyms

symbol	description
T	Test
SRS	Software Requirement Specification
VnV	Verification and Validation Plan

All the units, symbols, and abbreviations in the SRS and VnV apply to this document as well.

Contents

List of Tables

This document provides a summary of the Verification and Validation (VnV) of SCEC System. The test cases listed in the VnV Plan were executed, and this document contains a summary of the results.

The whole system follows the paper (1). Section 3 and 4 summarize the results of the functional and non-functional requirements respectively. Subsequent sections summarize the test results in details.

3 Functional Requirements Evaluation

3.1 Verify Load Parameters

- **Test Case(s)**: test_load_params
- **Requirements**: R1: load-input
- **Type**: Automated
- **Result Summary**: All the test cases passed successfully.
- **Result Artifacts Location**: Test result can be found at following link and to re-generate the same result test user needs to run test_load_params.py.

3.2 Verify inputs

- **Test Case(s)**: test_invalid_input
- **Requirements**: R2: Verify-Input
- **Type**: Automated
- **Result Summary**: All the test cases passed successfully.
- **Result Artifacts Location**: Test result can be found at following link and to re-generate the same result test user needs to run test_invalid_input.py.

3.3 Verify Temperature Value

- **Test Case(s)**: test_temperature_calculation
- **Requirements**: R3: Calculate-temperature-reflector, R4: Calculate-temperature-fluid
- **Type**: Automated
- **Result Summary**: Among 4 test cases 1 test case is failed. Calculated temperature for normal, high and medium inputs works good. However, extreme high initial temperature makes the temperature of glass 1 negative.
- **Result Artifacts Location**: Test result can be found at following link and to re-generate the same result test user needs to run test_temperature_calculation.py.

3.4 Verify Energy Value

- **Test Case(s)**: test_energy
- **Requirements**: R5: Calculate-energy
- **Type**: Automated
- **Result Summary**: Among 4 test cases 1 test case is failed. Calculated energy for high, low and extreme low inputs works good. However, extreme high calculated temperature makes the energy negative.
- **Result Artifacts Location**: Test result can be found at following link and to re-generate the same result test user needs to run test_energy.py.

4 Nonfunctional Requirements Evaluation

4.1 Understandability

- **Test Case(s)**: test_id9
- **Requirements**: NFR1: Understandability

- **Type**: Automated
- **Result Summary**: Understandability test performed using 3 different tools: PyLint, Flake8 and PyFlakes.
  PyFlakes do not generate any warning for coding style.
  Flake8 contains some minor coding styles for line under-indented and line too long. These warning can be negligible.
  PyLint generate some issues related to the import using src folder. Makefile is not able to find the location of the imports without "from src" statement. Still, PyLint gives 8.64 rating out of 10.
- **Result Artifacts Location**: Test result for Flake8 and PyFlake are located at result_flake8.log and result_pylint.log file respectively.

4.2 Maintainability

As the author has not yet received any response according to Maintainability surveys reporting about this area is postponed to the future drafts of the present document.

4.3 Usability

As the author has not yet received any response according to usability surveys reporting about this area is postponed to the future drafts of the present document.

4.4 Portability

For achieving portability, we tested the same system in Linux machine. First we tried to run the code which eventually create a virtual environment and install dependencies. The log file for the same can be found at result_makefile_run.log. We also tested it for running test case on linux which can be found at result_makefile_test.log.

As system was already developed and tested on MacOs, this system is portable with MacOs.

We also tried to test the software with Windows machine. However, it was showing the error for not able to install scipy library on windows machine. The error can be found at windows_error.log

5 Unit Testing

5.1 Constant Value Module (M2)

- **Test Case(s)**: test_constant
- **Module**: test_constant.py
- **Type**: Automated
- **Result Summary**: All the test cases passed successfully.
- **Result Artifacts Location**: Test result can be found at following link and to re-generate the same result test user needs to run test_constant.py.

5.2 Temperature ODE Module (M8)

- **Test Case(s)**: test_temp_ode
- **Module**: calculation.py
- **Type**: Manual/ Automated
- **Result Summary**: All the values of the temperature are present in the print result.
- **Result Artifacts Location**: Test result can be found at following link and to re-generate the same result test user needs to run main.py by removing comment from print statement.

6 Changes Due to Testing

SCEC system is continuously improved and added new modules throughout the testing part.

7 Automated Testing

SCEC system uses PyTest for unit, functional and system test for automated testing. All the test cases for different modules are written on Github repositry and result associated with those modules are also store at Github repositry.

8 Trace to Requirements

Table 1 contains the mapping of requirements to test cases.

	R1	R2	R3	R4	R5	NFR1	NFR2	NFR3	NFR4
test-input-area-id1		X							
test-input-temperature-id2		X							
test-input-emittance-id3		X							
test-load-id4	X								
test-fluid-temp-id5			X	X					
test-fluid-energy-id6					X				
test-output-temp-id7			X	X					
test-output-file-id8			X	X					
test-id9						X			
test-id10							X		
test-id11								X	
test-id12									X
test-constant-id13			X	X	X				
test-temp-ode-id14			X	X					

Table 1: Tracebility Between Test Cases and Requirements

9 Trace to Modules

Table 2 contains the mapping of modules to test cases.

	calculation.py	constant.py	energy_calculation.py	load_params.py	main.py	parameters.py	plot.py
test-input-area-id1				X	X		
test-input-temperature-id2				X	X		
test-input-emittance-id3				X	X		
test-load-id4				X		X	
test-fluid-temp-id5	X	X		X			
test-fluid-energy-id6		X	X	X			
test-output-temp-id7					X		X
test-output-file-id8					X		X
test-id9							
test-id10							
test-id11							
test-id12							
test-constant-id13		X			X		
test-temp-ode-id14	X				X		

Table 2: Tracebility Between Test Cases and Modules

10 Code Coverage Metrics

The statement coverage is summerized in Table 3

Name	**Stmts**	**Miss**	**Cover**
calculation.py	29	0	100%
constant.py	20	0	100%
energy_calculation.py	6	0	100%
load_params.py	22	0	100%
main.py	15	0	100%
parameters.py	14	0	100%
plot.py	16	0	100%
Total	122	0	100%

Table 3: Tracebility Between Test Cases and Requirements

References

[1] Hilario Terres, Arturo Lizardi, Raymundo Lpez, Mabel Vaca, and Sandra Ch˚vez. Mathematical model to study solar cookers box-type with internal reflectors. *Energy Procedia*, 57:1583–1592, 2014. 2013 ISES Solar World Congress.