Project Title: System Verification and Validation Plan for SPDFM

S. Shayan Mousavi M.

December 19, 2020

1 Revision History

Date	Version	Notes
Oct 29 2020	1.0	First Draft

Contents

List of Tables

2 Symbols, Abbreviations and Acronyms

symbol	description
T	Test
VnV	verification and validation
SPDFM	Surface Plasmon Dynamics Finite Method
MNPBEM	Metallic NanoParticle Boundary Element Method

The complete table of symbols, abbreviations and acronyms can be found in the SRS document of the software.

This document provides the information on validation and verification plans implemented for the SPDFM software. In this regard, the general approaches and plans are initially discussed and afterwards specific test cases and approaches for validation and verification of functional and nonfunctional requirements (can be found in SRS) are reviewed. VnV plans here are a combination of manual (assigned to a member of the VnV team to assess) and automated testing approaches to evaluate the correctness of the information (whether input or output) or satisfaction of a goal in SPDFM.

3 General Information

3.1 Summary

The SPDFM software is a software for calculating plasmon-enhanced electric field and electric current in a meshed geometry. This software should be able to setup an optical source (given the related parameters) and study how electric field and current densities in the dielectric are affected by illumination of this light source. The calculations in this software are based on the newly established theory of surface plasmon oscillations, nonlocal hydrodynamic theory of surface plasmons (Hiremath et al., 2012).

3.2 Objectives

This document tries to address the most important areas of the SPDFM software that can act like bottle necks of the system and make sure these areas function properly. In this regard, following sections will discuss how these key aspects, which include setting up a light source, having properly meshed geometry, well defined dielectric environment, and theoretical formulation are verified. These areas are reflected within the functional and nonfunctional system requirements in Section 5 of the SRS document. In this software some Python libraries are implemented such as FEniCS. The validity of these libraries is accepted and will not be checked here.

3.3 Relevant Documentation

The relevant documentation for SPDFM, including the problem statement (Mousavi (2020a)), SRS (Mousavi (2020b)), SRS checklist (Smith (2020)),

VnV report, MG, and MIS can be found in the devoted GitHub repository to this software. For theoretical aspects Hiremath et al. (2012), Monk et al. (2003), and Maier (2007) are the major important resources used in this software.

4 Plan

4.1 Verification and Validation Team

The VnV team members in this work and their contributions are as below.

- **S. Shayan Mousavi M. (author):** reviewing all the documentations, providing test cases and their execution, verifying the theoretical aspects and their implementation.
- **Dr. Spencer Smith (CAS 741 instructor):** reviewing the design of the software, all documentations, and the documentation style.
- **S. Parsa Tayefeh Morsal (domain expert):** reviewing all documentations.
- **Siddharth (Sid) Shinde (secondary reviewer):** reviewing VnV document.
- **Gabriela Sánchez Díaz (secondary reviewer):** reviewing MG and MIS documents.
- **Dr. Gianluigi Botton (supervisor):** reviewing theoretical aspects and finite element method implemented in SPDFM.
- **Dr. Alexander Pofelski (field expert):** reviewing theoretical aspects, finite element method implemented in SPDFM, and all documentations.

4.2 SRS Verification Plan

The SPDFM shall be verified in the following ways:

Initial reviews from assigned members of the VnV team (Dr. Spencer Smith, S. Parsa Tayefeh Morsal, Naveen Ganesh Muralidharan, and Shayan Mousavi).

In this regard, the document shall be manually reviewed using the SRS checklist (Smith (2020)) upon its initial version.

Secondary review by the author (Shayan Mousavi). The SRS document shall be reviewed after receiving initial reviews, and completion of VnV document.

Final review by the author (Shayan Mousavi) and the instructor (Dr. Spencer Smith). The document shall be manually reviewed according to the SRS checklist (Smith (2020)) after MG and MIS development.

Review of theoretical aspects by the field experts (Dr. Gianluigi Botton, Dr. Alexander Pofelski, and Shayan Mousavi). Theories used in the SRS document shall be reviewed manually with respect to the governing relations in the realm of plasmonic physics (Maier (2007), Hiremath et al. (2012), Monk et al. (2003)).

Feedback received from interested contributors through issue tracker in GitHub platform will also be used to improve this document.

4.3 Design Verification Plan

The design shall be verified by ensuring that key aspects in SPDFM are, as listed in Section 3.2. In this regard, the system functional requirements shall be tested initially, as outlined in 5.1, and in following the nonfunctional requirements will be review, as outlined in 5.2.

4.4 Implementation Verification Plan

The implementation shall be verified in the following ways:

- Code Walkthrough: This process will be performed by the author (Shayan Mousavi) and the field expert (Alex Pofelski). Code walkthough in this work follows the procedure suggested in MIT website (mit.edu, 1997) and uses their code walkthrough checklist.

- System Tests: System tests will be carried out as listed in Section 5. These tests target functional and nonfunctional requirements listed in the SRS document. However, as there is an overlap between input

related functional requirements (R1 and R2) and SPDFM modular design, to avoid repetition, these requirements are only tested in Section 6. System tests are conducted either manually or automatically which is outlined for each test individual in section 5.

4.5 Automated Testing and Verification Tools

Automated testing of SPDFM is conducted using Pytest library in Python. These tests are performed by predetermining some user inputs and comparing some targeted parameters with their expected values automatically. Automated testings for each test case is separately outlined in Section 5.1 and Section 6.

4.6 Software Validation Plan

Software validation due to the lack of experimental data is beyond scope this work.

5 System Test Description

5.1 Tests for Functional Requirements

The subsections below are designed to cover R3 and R4 functional requirements of the system, which are listed in Section 5 of the SRS document. As modular design of SPDFM has modules that are directly responsible for data input, corresponding functional requirements (R1 and R2, Section 5 of the SRS document) are tested in the unit testing section of this document (Section 6) to avoid repetition. It also worth mentioning that as SPDFM is written in Python, some areas of the verification such as variable types (integer, float, string, etc.), and existence of input files with proper format in the given file path are automatically tested by Python and are beyond scope of this document to be discussed here.

For all the tests below, the tolerance for value equality is 10^{-5}. In this regard, if a test case states that a parameter should be equal to a specific value, it means that the aimed value, and value of the tested parameter should be within tolerance-level proximity of each other.

5.1.1 Light Source Calculation Verification (and/or Validation) Tests

Test R 3: Verifying light source setup

1. **Test id1:** Calculation of the electric field of the light source

   Control: Automated

   Initial State: N/A

   Input: Polarity, direction, frequency of a plane wave light source, and a meshed geometry are given using input files and input data indicated in Table 1.

   Output: Below outputs should be generated for each of the real and imaginary parts of the electric field separately.

   - Superimposed plot of light wave oscillation towards the light propagation axis (call this line L) calculated by SPDFM and calculated by python built in functions.
   - Difference between two calculated values at each point of the space that is located on the line L. line L is the line that parallel to the direction of the light source and passes the point (0, 0, 0) of the space.
   - Measuring execution time of the calculations.

   Test Case Derivation: This test evaluates how precise external FEM toolbox (FEniCS) in SPDFM calculates both real and imaginary parts of the electric field of the light source in the space. In this test two different frequencies are being studied; low frequency at 600 THz (visible range which is important for the future studies) and high frequency at 30000 for giveing visibility to oscillations of the electric field in the nanometer-scaled space are considered. All meshes are cubic geometries of 40 nm length but density of the mesh is sorted from low to high to study impact of mesh density on the precision of the calculated light source.

   How test will be performed: This test can be executed by running test_ls.py; this code is an auxiliary code which imports SPDFM mesh

Test Cases:	Test 1	Test 2
Input file	LS_t1.txt	LS_t2.txt
Polarity	0,1,0	0,1,0
Direction	1,0,0	1,0,0
Frequency (THz)	600	30000
Mesh Input	G_cube_10node.xml G_cube_10node_physical_region.xml G_cube_10node_facet_region.xml	G_cube_10node.xml G_cube_10node_physical_region.xml G_cube_10node_facet_region.xml
Test Cases:	Test 3	Test 4
Input file	LS_t1.txt	LS_t2.txt
Polarity	0,1,0	0,1,0
Direction	1,0,0	1,0,0
Frequency (THz)	600	30000
Mesh Input	G_cube_20node.xml G_cube_20node_physical_region.xml G_cube_20node_facet_region.xml	G_cube_20node.xml G_cube_20node_physical_region.xml G_cube_20node_facet_region.xml
Test Cases:	Test 5	Test 6
Input file	LS_t1.txt	LS_t2.txt
Polarity	0,1,0	0,1,0
Direction	1,0,0	1,0,0
Frequency (THz)	600	30000
Mesh Input	G_cube_40node.xml G_cube_40node_physical_region.xml G_cube_40node_facet_region.xml	G_cube_40node.xml G_cube_40node_physical_region.xml G_cube_40node_facet_region.xml

Table 1: Input data (files) for automated testing of the light source setup

input module, data structure module, and the function that sets up the light source. This code is not Pytest controlled and instead of pass/fail

results it provides the user with quantitative results. Shayan Mousavi is responsible for writing and execution of this test.

Test Cases:	Test 1	Test 2
Output files	testid2-1_Efield_real.pvd testid2-1_Efield_imag.pvd	testid2-2_Efield_real.pvd testid2-2_Efield_imag.pvd
Test Cases:	Test 3	Test 4
Output files	testid2-3_Efield_real.pvd testid2-3_Efield_imag.pvd	testid2-4_Efield_real.pvd testid2-4_Efield_imag.pvd
Test Cases:	Test 5	Test 6
Output files	testid2-5_Efield_real.pvd testid2-5_Efield_imag.pvd	testid2-6_Efield_real.pvd testid2-6_Efield_imag.pvd

Table 2: Output files for visual inspection of the light source setup

2. **Test id2:** Visual inspection of the electric field propagation of the light source

   Control: Manual

   Initial State: N/A

   Input: light polarity and direction, a frequency, and a meshed geometry are given using input files and input data indicated in Table 1.

   Output:

   - exported .pvd files containing the interpolated real and imaginary parts of the electric field of the light source in the entire space. These files are named as mentioned in Table 2.

   Test Case Derivation: This test visually evaluates the propagation of electric field in space. In this regard, the extracted pvd map of the electric field should be opened by a pvd reader (suggestion: Paraview software). If 3D colour map view is selected for illustration of the result (which is suggested) alternating domains are expected to be seen with domain width equal to wavelength of the light source. In this regard, for 600 THz source the wavelength is ∼499 nm, thus no alternation should be seen in real domain (due to the size of the mesh) and a dipole domain alternation should be observed in the result. The wavelength for the 30000 THz source is ∼10 nm, thus, alternating domains of the same

width is expected. However, user should observe a $\frac{\pi}{2}$ shift between real and imaginary components. Same colour domains are elongated towards polarity vector and colour alternation should happen towards the propagation vector (direction vector).

How test will be performed: The pvd maps can be obtained by executing test_visual_ls.py. Shayan Mousavi is responsible for writing and execution of this test.

Test R 4: Verifying calculated electric field and electric current density

1. **Test id3:** Plasmon enhanced electric field calculation compared to boundary element simulation

   Control: Manual

   Initial State: N/A

   Input: Input data is as listed in Table 3. These files, as is indicated in the Table 3, include all the required data for initiation of a SPDFM simulation. The meshed geometry in this test is a 20 nm in diameter nanoparticle embedded in the vacuum environment of diameter 40 nm. These 20 nm-sized particles are either simulated as an empty shell or as a particle with volume; see Table 3. Moreover, test cases here are divided into three separate sets of input with different number of nodes in the mesh.

   Output: Below outputs should be generated:

   - Simulation results are expected to be extracted as the pvd files and store Superimposed plot of electric field intensity vs. distance from the sphere surface for both MNPBEM simulated electric field and SPDFM. These plot will be provided for two directions one passing the centre of space point (0, 0, 0) and is parallel to (0, 1, 0) and the other one is parallel to (1, 0, 0).
   - Extracted data from

   Test Case Derivation: In this test excited electric field by a plane wave of 400 nm wavelength, around a 20 nm diameter sphere is calculated by SPDFM and MNPBEM toolbox. MNPBEM is a boundary element method (BEM) software for simulating plasmon activities is nanoparticles (Hohenester and Trügler, 2012). Although in MNPBEM, parameter determination is not as flexible as in SPDFM, and some discrepancies are expected due to the implementation of different theories (local quasi-static vs. non-local hydrodynamic) and different techniques (BEM vs FEM), MNPBEM result can still be compared with FEM simulations. This comparison should be closer when in FEM on the

Test Cases:	Test 1	Test 2
Input files	Input_t1.txt	Input_t1.txt
Polarity	0, 1, 0	
Direction	1, 0, 0	
Lambda(init) (nm)	400	
Lambda(fin) (nm)	500	
Steps	1	//
ε_0 (F/m)	8.85×10^{-12}	
μ_0 (H/m)	1.25×10^{-6}	
μ_1 (H/m)	1.25×10^{-6}	
γ (THz)	17.94	
Plasma freq. (THz)	2165	
β^2 (m^2/s^2)	1.16×10^{12}	
Input mesh Set1	G_shell_t1.xml	G_fill_t1.xml
(low mesh density)	G_shell_pr_t1.xml	G_fill_pr_t1.xml
	G_shell_fc_t1.xml	G_fill_fc_t1.xml
	177 nodes	178 nodes
Test Cases:	Test 3	Test 4
Input files	Input_t1.txt	Input_t1.txt
//	//	//
Input mesh Set2	G_shell_t2.xml	G_fill_t2.xml
(medium mesh density)	G_shell_pr_t2.xml	G_fill_pr_t2.xml
	G_shell_fc_t2.xml	G_fill_fc_t2.xml
	2306 nodes	1268 nodes
Test Cases:	Test 5	Test 6
Input files	Input_t1.txt	Input_t1.txt
//	//	//
Input mesh Set3	G_shell_t2.xml	G_fill_t2.xml
(high mesh density)	G_shell_pr_t3.xml	G_fill_pr_t3.xml
	G_shell_fc_t3.xml	G_fill_fc_t3.xml
	2306 nodes	2613

Table 3: Input data, required for SPDFM a complete FEM simulations used in Testid3.

boundary of the target particle is meshed instead of the whole volume (G_Shell_t2.xml). In this regard, for a 20 nm diameter sphere made of gold (parameters obtained from Grady et al. (2004)) that is only meshed on the surface, electric field intensity is compared between MNPBEM and SPDFM simulations in direction of light propagation (1, 0, 0) and polarity (0, 1, 0). To see how presence of the volume affects these results, a 20 nm fully meshed gold sphere is also compared with the MNPBEM results. As MNPBEM simulations use Gaussian units the output cannot be directly compared and here

How test will be performed: Executing test_efield_fem_mnpbem.py automatically initiates the test and simulations will be stored in Test Output folder. MNPBEM results are already simulated and results and the codes in MATLAB are provided in the MNPBEM folder. For comparing the MNPBEM results with SPDFM results, stored files should be opened with a .pvd viewer (ParaView is suggested). These results are analyzed in the VnV report document. Shayan Mousavi is responsible for writing and execution of this test.

2. **Test id4:** Plasmon enhanced electric current density testing

   # Implementation and execution of this test is beyond scope of the current version of this document and it is not possible to be delivered within the strict time window of CAS741. However, it is presented here to show the authors intention of conducting this test in the future.

   Control: Automated

   Initial State: N/A

   Input: Files regarding Tests 2, 4, and 6 in Table 3. These files include all the required data for initiation of a FEM simulation.

   Output: Below outputs shall be obtained:

   - Superimposed plots of current density calculated by SPDFM and Mie theorem solution for spherical particles (analytical solution) along x-axis (light propagation) and y-axis (light polarity).
   - Difference between two calculated values at each point along the mentioned axes.

   Test Case Derivation: According to Mie theorem the analytical solution for nonlocal hydrodynamic electromagnetic response of a spherical particle can be calculated.

   How test will be performed: Implementation is planned for the future.

5.2 Tests for Nonfunctional Requirements

Although it is believed that testing nonfunctional requirements in a short time-frame with limited number of users is not applicable and would not necessarily provide meaningful data, the plans for testing these areas are provided here. This plans show intention and approach of the author for verification of nonfunctional requirements, albeit in the future. The report reflecting results of tests mentioned below is beyond scope of the current draft of the VnV report but as soon as the enough data is gathered, the corresponding analysis will be released.

5.2.1 Usability

Test NR1: Capability of execution of the software

1. **Test id5:** Usability

   Type: Usability Survey

   Initial State: The system being used should already have Python3, and FEniCS toolbox installed on.

   Input/Condition: A usability survey with the questions listed in Section 7.2. For execution of a simulation, data provided in Test? is suggested to be used.

   Output/Result: Survey results

   How test will be performed: each participant shall install the software on a system and try to run a simulation. Respondents will be asked to rank their experience of installing and running a module. A final average grade of 3 will indicate that the users found the system to have average usability. The higher the score,the better the perception of usability. Shayan Mousavi and Alexander Pofelski shall be participate this test. This approach is suggested by Michalski (2019).

5.2.2 Maintainability

Test NR2: Maintainability and expandability of the software

1. **Test id6:** Maintainability

   Type: Maintainability Walkthrough

   Initial State: N/A

   Input/Condition: production version of SPDFM has been released.

   Output/Result: A graded report describing the maintainability of the repository

   How test will be performed: Dr. Alexander Pofelski shall check the repository for the following documentation: SRS, VnV Plan, MG, MIS, User Guide. He shall mark 1 point for each of the above documents. He shall read through each of the above documents and provide a grade

between 1 and 5 for clarity of the writing. A score of 1 represents a document that is hard to understand, and a score of 5 represents a document that is easy to understand. The user should also grade the traceability of each document. A score of 1 represents no links within the report, and a score of 5 represents many links between sections of the report. The user shall then divide the sum of the scores for all of the reports by 5. A final average grade of 3 will indicate that the users found the system to have average Maintainability. The higher the score,the better the perception of Maintainability. This approach is suggested by Michalski (2019).

5.3 Traceability Between Test Cases and Requirements

Table 4 shows the connection between functional and nonfunctional requirements and the tests provided in this document.

	R1	R2	R3	R4	NR1	NR2	NR3
Test1			X				
Test2			X				
Test3				X			
Test4				X			
Test5					X	X	
Test6							X
Test7		X					
Test8	X						
Test9	X						

Table 4: Traceability Matrix Showing the Connections Between Tests and Functional and Nonfunctional System Requirements

6 Unit Test Description

The modular design of SPDFM is introduced in MG document (MGS), and discussed in MIS document (MIS); according to these documents SPDFM is consist of eight modules. These modules are assigned to input the data,

input the mesh geometry, storing and organizing the data, calculate the electric field and electric current density, and output the data.

6.1 Unit Testing Scope

In the process of SPDFM verification, the modules that are the most emphasized on are the input (M5 and M6, MG) and output (M8, MG) modules. As the finite element solver module (M7, MG) uses an external finite element solver (FEniCS), and due to the fact that the obtained results from this module are separately verified within system verification section (Section 5), verifying M7 is beyond the unit testing scope. Moreover, hardware hiding module (M1, MG), SPDFM control module (M2, MG), data structure module (M3, MG), and output module (M8, MG) are not being tested here. About the data structure module (M3), it should be mentioned that as this module is being used in all other modules, it can be assumed that this module is being tested indirectly while others are verified. The reason that output module is not tested is that this module is majorly depends on the output module of the external FEM solver (FEniCS toolbox). Thus, as in this work it is assumed that FEniCS is verified, this module is exempted from testing.

6.2 Tests for Functional Requirements

6.2.1 Module 4: Constant parameters module (M4)

As the constant parameter module (M4, MIS and MG) is assigned to be a template object that holds the constants used in project, unit testing is performed by checking all the values stored in this module.

1. **Test id7**

   Type: Automatic

   Initial State: N/A

   Input: The constant values used in the SPDFM are stored in constants.txt.

   Output: PASS, if all the constant values in the template module is equal to (in the tolerance level distance of) the values stores in the constant.txt; FAIL, otherwise.

Test Cases:	Test 1	Test 2	Test 3
Input files	Input_t1.txt	Input_t2.txt	Input_t3.txt

Table 5: Input data (files) for input module (M5) unit testing

Test Case Derivation: The constant parameters used in the code should be equal to the constant parameters in Table 2 of the SRS document.

How test will be performed: By execution of test_const.py using pytest, all the values in the constant parameter object (template module M4, MG) will be tested.

6.2.2 Module 5: Input parameters modules (M5)

1. **Test id8**

Type: Automatic

Initial State: N/A

Input: In this test the input data is as stated in Table 5. constants.txt.

Output: Pass, if all the constant values in the template module is equal to (in the tolerance level distance of) the values stored in the input file; Fail, otherwise.

Test Case Derivation: This test only verifies if input module can properly store data in the data structure.

How test will be performed: By execution of test_inputparam.pyperformance of input module (M5, MG and MIS) will be tested.

6.2.3 Module 6: Input Mesh modules (M6)

1. **Test id9**

Type: Manual

Initial State: N/A

Input: In this test the inputs are mesh files listed in Table 3.

	M4	M5	M6	M7
Test1				X
Test2				X
Test3				X
Test4				X
Test5				
Test6				
Test7	X			
Test8		X		
Test9			X	

Table 6: Traceability Matrix Showing the Connections Between Tests and Modules. Modules that are not in the table are beyond scope of testing plan in this work

Output: N/A Test Case Derivation: This test aims to verify that mesh input module is capable of inputting a mesh and storing it in the data structure module and user can extract (plot) later. In this regard, test_inputmesh.py uses mesh input module to input and store a mesh then by plotting it examines if the mesh is properly restored or not.

How test will be performed: By executing test_inputmesh.py meshes in Table 3 will be input and stored using mesh input module and then they will be called and plotted. Shayan Mousavi is responsible of executing this test.

6.3 Tests for Nonfunctional Requirements

Unit testing the nonfunctional requirements in beyond scope and time-frame of this work.

6.4 Traceability Between Test Cases and Modules

The connections between tests and modules is indicated in Table 6.

References

Spdfm/docs/design/mg at master · shmouses/spdfm. https://github.com/shmouses/SPDFM/tree/master/docs/Design/MG. (Accessed on 12/16/2020).

Spdfm/docs/design/mis at master · shmouses/spdfm. https://github.com/shmouses/SPDFM/tree/master/docs/Design/MIS. (Accessed on 12/16/2020).

Nathaniel K Grady, Naomi J Halas, and Peter Nordlander. Influence of dielectric function properties on the optical response of plasmon resonant metallic nanoparticles. *Chemical Physics Letters*, 399(1-3):167–171, 2004.

Kirankumar R Hiremath, Lin Zschiedrich, and Frank Schmidt. Numerical solution of nonlocal hydrodynamic drude model for arbitrary shaped nano-plasmonic structures using nédélec finite elements. *Journal of Computational Physics*, 231(17):5890–5896, 2012.

Ulrich Hohenester and Andreas Trügler. Mnpbem–a matlab toolbox for the simulation of plasmonic nanoparticles. *Computer Physics Communications*, 183(2):370–381, 2012.

Stefan Alexander Maier. *Plasmonics: fundamentals and applications*. Springer Science & Business Media, 2007.

Peter Michalski. Latticeboltzmannsolvers/docs/vnvplan/systvnvplan at master · peter-michalski/latticeboltzmannsolvers. https://github.com/peter-michalski/LatticeBoltzmannSolvers/tree/master/docs/VnVPlan/SystVnVPlan, 2019. (Accessed on 10/29/2020).

mit.edu. Checklist for code walkthroughs (draft, version 1.2, 10/30/97). http://www.mit.edu/~mbarker/ideas/checkcode.html, 1997. (Accessed on 12/14/2020).

Peter Monk et al. *Finite element methods for Maxwell's equations*. Oxford University Press, 2003.

S. Shayan Mousavi. Spdfm/docs/problemstatement at master · shmouses/spdfm · github. https://github.com/shmouses/SPDFM/tree/master/docs/ProblemStatement, 2020a. (Accessed on 10/29/2020).

S. Shayan Mousavi. Spdfm/docs/srs at master · shmouses/spdfm · github. https://github.com/shmouses/SPDFM/tree/master/docs/SRS, 2020b. (Accessed on 10/29/2020).

Spencer Smith. Blankprojecttemplate/docs/srs/srs-checklist.pdf · master · w. spencer smith / cas741 · gitlab. https://gitlab.cas.mcmaster.ca/smiths/cas741/-/blob/master/BlankProjectTemplate/docs/SRS/SRS-Checklist.pdf, Sept 2020. (Accessed on 10/29/2020).

7 Appendix

This is where you can place additional information.

7.1 Symbolic Parameters

The definition of the test cases will call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

7.2 Usability Survey Questions?

Using the following rubric please rate the five statements found below (this rubric is suggested by Michalski (2019)):

1. The formatting of the input file was easy to understand.

1 - strongly disagree
2 - somewhat disagree
3 - neither agree nor disagree
4 - somewhat agree
5 - strongly agree

2. The location to place the input file was easy to find.

1 - strongly disagree
2 - somewhat disagree
3 - neither agree nor disagree
4 - somewhat agree
5 - strongly agree

3. Navigating to the correct module was straightforward.

1 - strongly disagree
2 - somewhat disagree
3 - neither agree nor disagree
4 - somewhat agree
5 - strongly agree

4. The MG and MIS of this product explain the modules well.

1 - strongly disagree
2 - somewhat disagree
3 - neither agree nor disagree

4 - somewhat agree
5 - strongly agree

5. I would recommend this product.

1 - strongly disagree
2 - somewhat disagree
3 - neither agree nor disagree
4 - somewhat agree
5 - strongly agree