**STEP PROJECT REPORT**

**ABSTRACT**

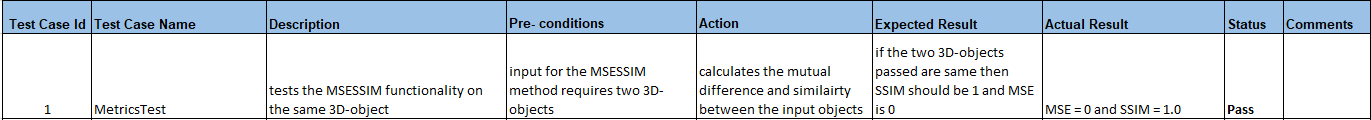
The goal of this STEP project is to create a testing framework to test the MATLAB code of computational imaging of microscopy. The computational imaging process in microscopy includes different stages: data collection from a microscope, data pre-processing, image reconstruction, and image post-processing. This report contains the details of testing framework created to test the project.

**INTRODUCTION**

Scientiﬁc applications perform numerical simulations of different natural phenomena in the ﬁled of computational physics and chemistry, mathematics, informatics, mechanics, bioinformatics, etc. They are primarily developed for the scientiﬁc research community and usually they are used for simulating some I/O and data extensive experiments [Segal 2005]. The performance of such simulation experiments requires powerful supercomputers, high performance, or Grid computing [Vecchiola et al. 2009].

A scientiﬁc application is a software applications which turns the object into mathematical models by simulating activities from the real world [Ziff Davis Publishing Holdings 1995]. Scientiﬁc applications are different from commercial application, especially in the process of testing and evaluation. They are developed for the speciﬁc research community and not evaluated by customers. Testing is usually performed by comparing the obtained results with theory or performed physical experiments. Also, scientists can decide for the correctness of the results visually, by comparing images.

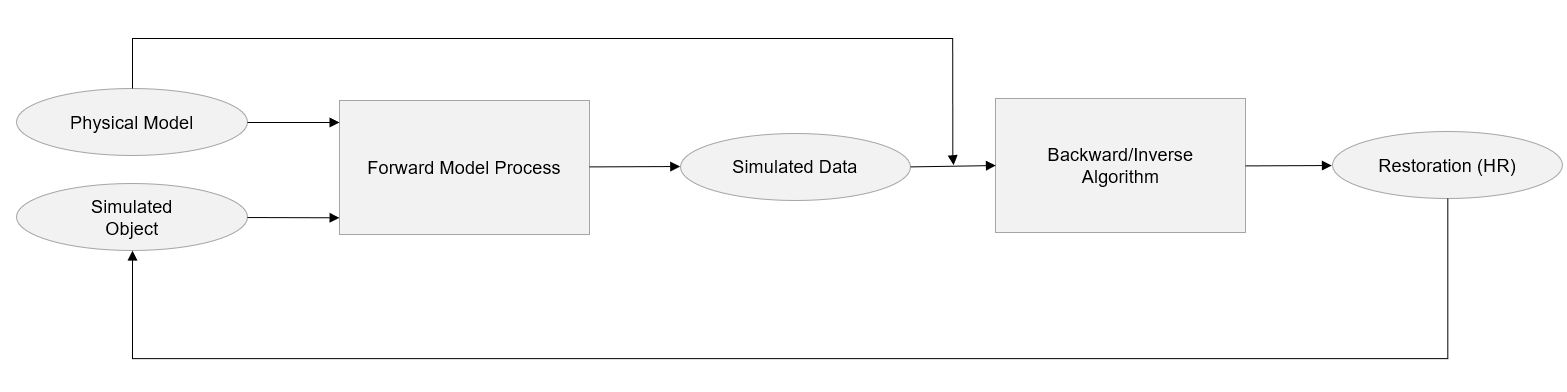
In this paper we include the software engineering practices in scientific application testing. For each functional requirement of the scientific application we design test cases, before testing the functionality. A test case definition consists of a unique identiﬁer, name, goal, preconditions, execution environment, expected and actual results, status (passed/failed) and history of changes as shown below:



**Table 1:** An Example of Test Case Definition

**APPLICATION**

A testing framework will be created for the scientific application called 3D-SIM (3D Structured Illumination Microscopy). Its goal is to make the collected microscopy images more clearer using computational imaging and obtain desired images with increased information content. To achieve this, we model the microscopy raw data to a simulated object and then apply algorithms and do a performance analysis.



**Figure 1:** SIM Integration Test Flow

**Physical Model** creates bead using the raw data settings for the 3-slit Tunable system.

**Simulated Object** is the original image from the microscopy that is constructed through computational imaging using the raw data settings.

**Forward Model process** takes physical model and simulated object as inputs to produce simulated data as output.

**Simulated Data** is used to test the model and given as input to the backward algorithm.

**Backward/Inverse Algorithm** uses the raw data settings and simulated data to generate multiple intermediate images and then generates the final restoration image.

**Restoration** object is compared with the simulated object and the process is repeated until the desired results are achieved.

Images collected by 3D-SIM are modeled as:

g = Hf ; where H is the forward model, g is the data of size [X, Y, Z] and f is the object of size [2X, 2Y, 2Z].

**Inverse problem:** given the data g, restore the object f.

**Optimization-based solution:** one can solve the above inverse problem by solving the optimization problem:

fr = H-1g := argminf (||Hf - Ug|| + R(f))

where fr is the restoration, U is the upsampling operator/function, R is the regularization, and H-1 is the inverse operator/function.

**TESTING FRAMEWORK**

A testing framework is created for the application 3D-SIM which creates computational Imaging for 3D- Microscopy. Using this framework, we are going to test whether the model can produce desired images which are better than collected microscopy images.

Before testing the required test cases need to be defined in similar format as mentioned in the Table 1.

For the microscopy imaging process deals with 2D and 3D images, the metrics mean square error (MSE) and the structural similarity index measure (SSIM) can be used in the tests to validate the results.

The testing framework should be created as defined below.

**Testing framework:** to test the optimization-based solution, one creates some simulated object fs, applies H to fs to obtain the simulated data gs, then uses optimization-based solution to obtain the restoration fr. One then checks whether fr is the same as fs. In other words, the input and output of this testing framework should be:

**Input:**

* Settings object: containing all the parameters of the inverse problem, i.e. in SIM, the parameters are wavelength, NA, etc.
* Dimensions [X, Y, Z]
* Forward function H
* Restoration function H-1
* Simulation function S to generate simulated data fs

**Output:**

* Relative error between the restoration fr and the simulated object fs: ||fr - fs||/||fs||.

The testing framework is independent of the programming language used. For convenience, we will build this testing framework in MATLAB and make sure the MATLAB codes work correctly for the flowchart in Figure 2 below. Then we will translate the MATLAB part of the reconstruction into high-performance languages, such as, Python or Julia on HPC clusters for much better performance. The correctness of the Python or Julia codes on HPC clusters can be guaranteed by the unit-tests using the MATLAB codes’ results since the MATLAB codes would have already been tested and verified.



**Figure 2:** Flowchart of testing framework for computational imaging in microscopy.

**EXPERIMENTS**

As part of the testing framework for the application 3D-SIM below changes were done in the code:

* **Settings** data structure object is created to include all the parameters that is needed for the inverse algorithm. It is created so that all the parameters can be easily used in different functions and the changes made in any code reflects in the object. Main purpose is other applications can also use the existing code functionality by just passing the Settings object.
* **InverseTest** contains complete functionality of the application which includes the forward model process and restoration process. “**TestInverseTest.m**” is created to test InverseTest.