# System Verification and Validation Report for ROC: Software estimating the radius of convergence of a power series

John M Ernsthausen

December 25, 2020

## 1 Revision History

| Date | Version | Notes |
|---|---|---|
| 24 December 2020 | 1.0 | First submission |

## 2 Symbols, Abbreviations, and Acronyms

Symbols, abbreviations, and acronyms applicable to ROC are enumerated in Section 1 of the Software Requirements Document (SRS) (Ernsthausen, 2020c).

## Contents

In this document, we report the results of executing the verification and validation plan for ROC.

Our objective for ROC is correctness, accuracy, and timing. ROC is an idea for an academic research project. The impression from this software will guide future decisions about this idea as a viable research project.

### 2.1 Relevant Documentation

Relevant documentation is completed. All GitHub issues were fully considered and subsequently closed.

Relevant documentation includes the authors Software Requirements Specification (SRS) (Ernsthausen, 2020c), the authors Module Guide (MG) (Ernsthausen, 2020a), and the authors Module Interface Specification (MIS) (Ernsthausen, 2020b).

## 3 Verification and Validation Plan

Verification and Validation of ROC includes automated testing at the module level, and the system level. We achieved continuous integration through TravisCI.

### 3.1 Verification and Validation Team

The ROC team includes author John Ernsthausen, fellow students Leila Mousapour, Salah Gamal aly Hessien, Liz Hofer, and Xingzhi Liu as well as Professors Barak Shoshany, Spencer Smith, George Corliss, and Ned Nedialkov. The author appreciates the helpful comments and superior guidance on this project.

### 3.2 SRS Verification Plan

The SRS document Verification and Validation Plan for ROC was peer-reviewed by secondary reviewer Salah Gamal aly Hessien and Prof. Spencer Smith the course instructor.

The SRS document was published to GitHub. Defects were addressed with issues on the GitHub platform.

### 3.3 Design Verification Plan

The Design documents MG and MIS for ROC were peer-reviewed by domain expert Leila Mousapour and Prof. Spencer Smith the course instructor.

The MG and MIS documents were published to GitHub. Defects will be addressed with issues on the GitHub platform.

### 3.4 Implementation Verification Plan

Every effort was made to complete every aspect promised in this documentation.

## 4 Functional Requirements Evaluation

ROC does not have system functional requirements at this time.

## 5 Nonfunctional Requirements Evaluation

We considered Timing and Accuracy.

### 5.1 Timing

Gprof output roc/examples_cpp/gprof_output.txt on the Layne-Watson problem (Watson, 1979) shows that ROC did not consume a measureable amount of CPU time to compute $R_c$ for this problem.

### 5.2 Accuracy and Comparison to Existing Implementation

We exercised 3TA, 6TA, and TLA on the Layne-Watson problem (Watson, 1979), computing each analysis at each timestep. We discovered that fitting errors and truncation errors could be small (less than 1e-10), but one or the other of 3TA/6TA would align with TLA. This is not what the author expected. As a result, the decision logic proposed by Chang and Corliss (1982) is in question.

Careful research needs to be done to determine the correct indicators for which analysis 3TA/6TA/TLA to apply. As a result the nonfunctional accuracy requirement could not be fulfilled as I don't know the logic for this at this time.

The author plans to compare and contrast the $R_c$, $\mu$, fitting error, and truncation error among the three methods 3TA, 6TA, and TLA. These results will be compared to the Chang and Corliss (1982) existing implementation.

## 6 Unit Testing

Comparison with closed form solutions proved ROC to be accurate. These comparisons are among the unit tests. The assertions were chosen to be optimal to the number of digits reported, meaning that rounding the last digit down will cause the test to fail.

All the unit test pass.

## 7 Changes Due to Testing

I'm not sure what this is.

## 8 Automated Testing

All unit tests are automated.

Valgrind, gcov, and clang-format are automated via my top level Ruby script.

## 9 Trace to Requirements

Yes.

## 10 Trace to Modules

Yes.

## 11 Code Coverage Metrics

Visit my website to review gcov output.

## References

YF Chang and G Corliss. Solving ordinary differential equations using Taylor series. *ACM TOMS*, 8(2):114–144, 1982.

J.M. Ernsthausen. Module guide for ROC: Software estimating the radius of convergence of a power series. `https://github.com/JohnErnsthausen/roc/blob/master/docs/MG/MG.pdf`, 2020a.

J.M. Ernsthausen. Module interface specification for ROC: Software estimating the radius of convergence of a power series. `https://github.com/JohnErnsthausen/roc/blob/master/docs/MIS/MIS.pdf`, 2020b.

J.M. Ernsthausen. Software requirements specification for ROC: Software estimating the radius of convergence of a power series. `https://github.com/JohnErnsthausen/roc/blob/master/docs/SRS/SRS.pdf`, 2020c.

L.T. Watson. A globally convergent algorithm for computing fixed points of $c^2$ maps. *Appl. Math. Comput.*, 5:297–311, 1979.