SymEngine User Guide

Preface
This user guide is for those who want to use the SymEngine library to perform series expansion calculations. Because SymEngine is still in development, the functionalities listed in this document are subject to change at any time. Please refer to the GitHub repository for the latest version of SymEngine (https://github.com/symengine/symengine). This does not show how to use SymEngine in other languages with wrappers (please refer to documentation on the GitHub wiki).

README.md: https://github.com/symengine/symengine/blob/master/README.md
SymEngine GitHub Wiki: https://github.com/symengine/symengine/wiki

Overview of the product
SymEngine is an open-source C++ symbolic manipulation library. While SymEngine can be used as a standalone library, it is intended to provide backend support for computer algebra systems written in other languages such as C, Python, Ruby, Julia and Haskell.

Installation or distribution
To install and build SymEngine on your machine, follow the instructions on this page corresponding to your computer's operating system: https://github.com/symengine/symengine/wiki/Building-SymEngine
It is not necessary to install the Piranha and Flint libraries to run SymEngine or the series expansion module specifically. Please refer to their respective documentation on how to install these libraries separately.

To use on SymEngine on an external C++ program, follow the instructions on this page: https://github.com/symengine/symengine/wiki/Using-SymEngine-from-a-Cpp-project

Functionality
A comprehensive list of methods and descriptions of the SymEngine series expansion module can be found in the SymEngine wiki page

Troubleshooting/Frequently Asked Questions
Q: What can I do with SymEngine’s series expansion?
The SymEngine series expansion module allows users of SymEngine to, given an expression, compute the series expansion of that expression in either one or several variables.

Q: What is the series expansion of an expression?
The series expansion of an expression is a polynomial approximation of that expression.

Q: How is SymEngine’s series expansion implemented?
SymEngine implements series expansion using the visitor design pattern. Expressions in SymEngine are represented as parse trees. Series expansion is performed by a separate SeriesVisitor class which recursively computes the series expansion of each node in the tree.

Q: What other things can SymEngine do?
Yes, SymEngine has other functionalities such as matrix operations, integration and differentiation of functions, and many more. This user guide does not cover these functionalities. Look at the SymEngine repository to find the appropriate test cases on how to utilize them.

Q: How can I contribute to SymEngine?
SymEngine provides many beginner contributor guides on how to get set up and where to get started: https://github.com/symengine/symengine/wiki. There are plenty of issues that need to be resolved for the series expansion module, also listed here: https://github.com/symengine/symengine/wiki/expression-series-expansion. The best way to get started is by looking at past pull requests/issues on SymEngine’s Github repository, asking specific questions on the SymEngine Gitter chat room (link in contact information), and attempting to solve current issues and bugs after getting comfortable with the library.

Contact Information
Charles Chen, cyuchen@ucdavis.edu
Iris Lui, iwlui@ucdavis.edu
James Stojic, jbstojic@ucdavis.edu
Matthew Luszczak, myluszczak@ucdavis.edu

Gitter: https://gitter.im/symengine/symengine
Mailing list: https://groups.google.com/forum/#!forum/symengine
Appendix

Fast-Series Expansion for SymPy and SymEngine
Requirements Document

Revision History:
5/31/16 - Update System Architecture UML
5/30/16 - Update names of univariable classes, appendices information
3/10/16 - Include System Architecture UML
3/7/16 - Add to requirements on univariable/multivariable use cases, to appendices resources
2/6/16 - Update Requirements deadlines
1/30/16 - Version 1

Introduction:
The series expansion of a function has a wide variety of applications, including the computation of limits, the solution of differential equations, the development of numerical methods, and the analysis of complex functions. Since many applications require expanding several functions into series many times, the speed of an implementation is particularly important.

We will significantly improve and expand the new polynomial-based series expansion modules in SymEngine. The current series expansion modules is missing some key components, such as mathematical functions bessel, gamma, and zeta and manipulation of full multivariable expressions. There are also many other issues that could potentially arise in the future. These functionalities are something we hope to implement and develop. We hope to coordinate with the current GitHub community of developers who are working on SymEngine. A long term goal (if possible) is to replace the default series module in SymPy with the new one.

Glossary:
Series: A series is a sum of an infinite number of terms.
Series expansion: The series expansion of a function is a series converging to a function on some interval in its domain.
Github: a Web-based Git repository hosting service. It offers all of the distributed revision control and source code management (SCM) functionality of Git as well as adding its own features.
Newton’s method: a root-finding algorithm that uses the first few terms of the Taylor series of a function f(x) in the vicinity of a suspected root. This method can be modified to provide a quick (second order) method for finding the solution to various types of equations.
Computer Algebra System (CAS): A computer program or environment for performing symbolic manipulation of mathematical expressions.

System Architecture Overview:
SymEngine is an open-source C++ library that can be used as a computer algebra system. Expressions in Symengine are represented as trees whose nodes are the symbols in the expression. Access to elements of an already existing expression is done through specialized visitor classes. References are handled using the Teuchos RCP smart pointer. All classes in SymEngine are derived from a single Basic class.

Currently, there are eight classes in the library related to series: SeriesBase, in series.h and series.cpp, UnivariateSeries in series_generic.h and series_generic_cpp, URatPSeriesPiranha and UPSeriesPiranha in series_piranha.h and series_piranha.cpp, and URatPSeriesFlint in series_flint.h and series_flint.cpp. In the future, we intend to replace the URatPSeriesPiranha, UPSeriesPiranha and URatPSeriesFlint, classes with our own.
**Requirements:** (Deadlines subject to change)

1. As a user of the library, I should be able to use the SymEngine library to quickly find the series expansion of any expression composed of the atoms of a SymEngine expression. (Week 9)
2. As a user of the library, I should be able to use the SymEngine library to efficiently manipulate a series. (Week 7)
3. As a user, I want to be able to efficiently and correctly calculate the series expansion and other series operations according to the standard mathematical laws. (Week 9)
4. As a user and developer, I should be able to verify both the algorithms and implementation used. Toward this end, I should be able to view readable code conforming to the project’s style guide. This code should contain well-written comments explaining the details both of the algorithm and its implementation. (Continuous)
5. As a user and developer, I should be able to improve my understanding of SymEngine by reading the appropriate documentation. (Continuous, we will document as we go.)
6. As a user and developer, I should be able to use SymEngine’s series module without relying on external libraries. (Week 9)
7. As a user and developer, when different algorithms have varying advantages and disadvantages, I should be able to manually choose the algorithm best suited to my particular situation (Continuous).
8. As a user and developer, I should be able to measure the performance of series expansion and manipulation by testing the code against prepared benchmarks. (Week 10)
9. As a user, I should be able to expand a function in multiple variables into a multivariate series. (Week 15)
10. As a C, Python, Ruby, or Julia programmer or SAGE user, I should be able to access the full functionality of SymEngine with respect to series through the appropriate wrappers (Week 17).

**User cases for univariable polynomial classes** (UIntPoly, UExprPoly, UExprDict):

- As a User, I should be able to...
  1. … construct a univariable polynomial with numeric and symbolic coefficients given a sparse or dense representation.
  2. … perform addition, negation, subtraction, or multiplication between two univariable polynomials.
  3. … verify the correctness of the implementations of the above operations through tests integrated into the SymEngine code.
  4. … measure efficiency of the algorithm with benchmarks.

**Acceptance Test:** Code passes automatic continuous integration tests (TravisCI and Appveyor); code passes other automated tests which we will have to add; code is approved by the client and the SymEngine project members and is merged into repository.

**User cases for univariable series** (UnivariateSeries):

1. … construct a univariable series with univariable polynomials with precision.
2. ... perform addition, negation, subtraction, or multiplication between two univariable series.
3. ... verify the correctness of the implementations of the above operations through tests integrated into the SymEngine code.
4. ... measure efficiency of the algorithm with benchmarks.

Acceptance Test: Code passes automatic continuous integration tests (TravisCI and Appveyor); code passes other automated tests which we will have to add; code is approved by the client and the SymEngine project members and is merged into repository.

User cases for multivariable polynomial classes (MultivariateIntPolynomial, MultivariatePolynomial, MultivariablePolynomialExpr):
   As a User, I should be able to...
1. ... create, negate a multivariable polynomial, or add, subtract, or multiply two multivariable polynomials.
2. ... verify the correctness of the implementations of the above operations through tests integrated into the SymEngine code.
3. Measure the efficiency of the above implementations with ready-made benchmarks
4. ... add, subtract, or multiply a multivariable polynomial with univariable polynomial.
5. ... add, subtract, or multiply two univariable polynomials with different variables
6. ... measure efficiency of the algorithm with benchmarks.

Acceptance Test: Code passes automatic continuous integration tests (TravisCI and Appveyor); code passes other automated tests which we will have to add; code is approved by the client and the SymEngine project members and is merged into repository.

User cases for multivariable series:
   As a User, I should be able to ...
1. ... create, negate a multivariable series, or add, subtract, or multiply two multivariable series.
2. ... find the multivariable series expansion of a legal SymEngine expression using a series_visitor object.
3. ... verify the correctness of the implementations of the above operations through tests integrated into the SymEngine code.
4. ... measure the efficiency of the above implementations with ready-made benchmarks
5. ... add, subtract, or multiply a multivariable series with univariable series.
6. ... add, subtract, or multiply two univariable series with different variables
7. ... measure efficiency of the algorithm with benchmarks.

Acceptance Test: Code passes automatic continuous integration tests (TravisCI and Appveyor); code passes other automated tests which we will have to add; code is approved by the client and the SymEngine project members and is merged into repository.
Prototyping Code and Test Cases:
For a comprehensive list of pull requests, benchmark tests, and description of progress of this project please look at https://github.com/symengine/symengine/wiki/UC-Davis-Series-Expansion-project

Appendices:
CMake
Piranha: https://github.com/bluescarni/piranha
Flint: http://www.flintlib.org/
Programming Languages: C++
Teuchos RCP
SymEngine main repository: https://github.com/symengine/symengine
GitHub SymEngine UC Davis Series Expansion project wiki page: https://github.com/symengine/symengine/wiki/UC-Davis-Series-Expansion-project
SymEngine Google Group: http://groups.google.com/group/symengine
SymEngine Gitter Chat Room: https://gitter.im/symengine/symengine