The Journal of Open Source Software

ChebTools: C++11 (and Python) tools for working with Chebyshev expansions

Ian H. Bell[1], **Bradley Alpert**[1], **and Lucas Bouck**[1,2]

1 National Institute of Standards and Technology, Boulder, CO, USA **2** George Mason University, Fairfax, VA, USA

DOI: 10.21105/joss.00569

Software

- Review
- Repository
- Archive

Submitted: 29 January 2018
Published: 12 February 2018

Summary

Chebyshev-basis expansions, and more broadly, orthogonal polynomial expansions, are commonly used as numerical approximations of continuous functions on closed domains (Boyd 2013,Mason and Handscomb (2003),Battles and Trefethen (2004)). One of the most successful projects that makes use of the Chebyshev expansions is the `chebfun` library (Driscoll, Hale, and Trefethen 2014) for MATLAB. Other similar libraries are `pychebfun`[1], `chebpy`[2], and `Approxfun`[3]. Our library `ChebTools` fills a similar niche as that of `chebfun` – working with Chebyshev expansions.

The primary motivation for the development of `ChebTools` is the need for a highly optimized and fast C++11 library for working with Chebyshev expansions in order to do one-dimensional rootfinding from nonlinear functions of multiple variables that arise out of thermodynamic modeling (equations of state of multiple state variables). A manuscript on this topic is forthcoming that builds off the tools developed in `ChebTools`.

Internally, the header-only library `Eigen`[4] is used to carry out all the matrix operations, allowing for behind-the-scenes vectorization without any user intervention. Thus the library is also very computationally efficient.

A short list of the capabilities of `ChebTools` is as follows:

- Construct a Chebyshev expansion approximation of any one-dimensional function in an arbitrary closed domain.
- Apply numerical operators on expansions : addition, subtraction, multiplication, arbitrary mathematical functions.
- Find all roots of the function.
- Calculate the derivative of the expansion.

While C++11 allows for the development of very computationally-efficient code, users often prefer a higher-level interface. As such, a comprehensive one-to-one Python wrapper of `ChebTools` was developed through the use of the pybind11[5] library(Jakob, Rhinelander, and Moldovan (2016)). This library offers the capability to natively integrate C++11 and Python - it is, for instance, trivial to pass Python functions to C++ functions accepting C++11 `std::function` (for use as a callback, or here, as the function sampled to generate the expansion).

We provide a `jupyter` notebook (Pérez and Granger 2007) mirroring much of the example code from Battles and Trefethen (2004) and `pychebfun`. Furthermore, a binder[6]

[1]https://github.com/pychebfun
[2]https://github.com/chebpy/chebpy
[3]https://github.com/JuliaApproximation/ApproxFun.jl
[4]http://eigen.tuxfamily.org
[5]https://github.com/pybind/pybind11
[6]https://mybinder.org/

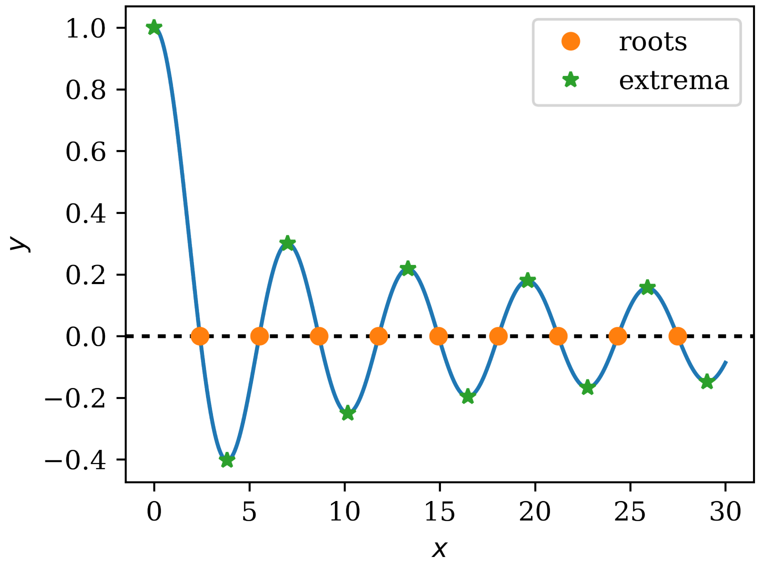

Figure 1: Roots and extrema of the 0-th Bessel function

environment has been configured such that ChebTools can be explored in an online jupyter notebook without installing anything on the user's computer.

In this Python code block, we demonstrate finding the roots and extrema of the 0-th Bessel function in the closed domain [0, 30]:

```
import scipy.special
import ChebTools
# Only keep the roots that are in [-1,1] in scaled coordinates
only_in_domain = True
# The 0-th Bessel function (for code concision)
def J0(x): return scipy.special.jn(0,x)
# Make a 200-th order expansion of the 0-th Bessel function in [0,30]
f = ChebTools.generate_Chebyshev_expansion(200, J0, 0, 30)
# Roots of the function
rts = f.real_roots(only_in_domain)
# Extrema of the function (roots of the derivative, where dy/dx =0)
extrema = f.deriv(1).real_roots(only_in_domain)
```

A graphical representation of the roots and extrema of the 0-th Bessel function in the closed domain [0, 30] is shown in the `jupyter` notebook in the repository and is also displayed here:

Disclaimer

Contribution of the National Institute of Standards and Technology, not subject to copyright in the U.S. Commercial equipment, instruments, or materials are identified only in order to adequately specify certain procedures. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the products identified are necessarily the best available for the purpose.

References

Battles, Z., and L.N. Trefethen. 2004. "An Extension of MATLAB to Continuous Functions and Operators." *SIAM J. Sci. Comput.* 25 (5):1743–70. https://doi.org/10.1137/S1064827503430126.

Boyd, J.P. 2013. "Finding the Zeros of a Univariate Equation: Proxy Rootfinders, Chebyshev Interpolation, and the Companion Matrix." *SIAM Review* 55 (2):375–96. https://doi.org/10.1137/110838297.

Driscoll, T.A., N. Hale, and L.N. Trefethen. 2014. "Chebfun Guide." Oxford: Pafnuty Publications.

Jakob, W., J. Rhinelander, and D. Moldovan. 2016. "Pybind11 – Seamless Operability Between C++11 and Python."

Mason, J.C., and D.C. Handscomb. 2003. *Chebyshev Polynomials*. Boca Raton: Chapman & Hall.

Pérez, F., and B. E. Granger. 2007. "IPython: A System for Interactive Scientific Computing." *Computing in Science and Engineering* 9 (3). IEEE Computer Society:21–29. https://doi.org/10.1109/MCSE.2007.53.