Chebyshev Expansions: Background and Computing in the C++/Python Library ChebTools

Lucas Bouck and Ian Bell

George Mason University, Fairfax, VA, U.S.A. and National Institute of Standards and Technology, Boulder, CO, U.S.A

Summary

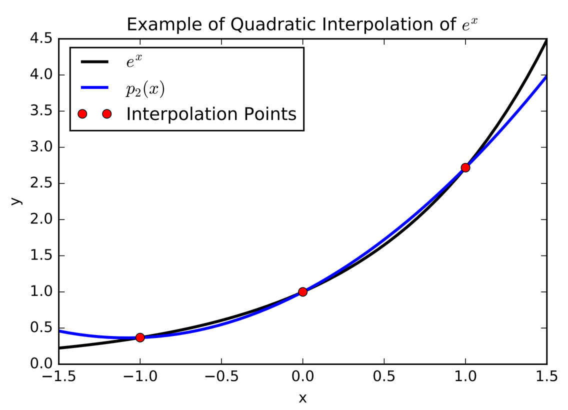
This poster has two goals:

1. Give a brief background of Chebyshev interpolation
   - Background on Polynomial Interpolation
   - Motivation and useful properties Chebyshev Polynomials
2. Introduce and demonstrate the capabilities of `ChebTools`
   - C++/Python library for working with Chebyshev expansions
   - In the early stages and are looking for help in growing the library

Polynomial Interpolation

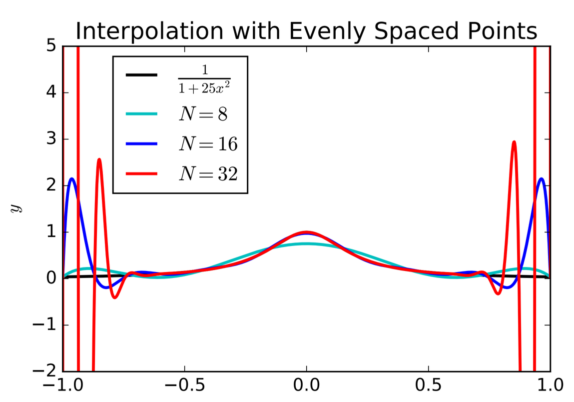
What is it

- Given points, $\{(x_i, y_i)\}_{i=0}^n$
- Want a degree n polynomial, p_n
- $p_n(x_i) = y_i$ for all i
- Interpolate a function f by setting $y_i = f(x_i)$
- Gives an approximation of a function with just addition and multiplication for operations



Need to be Careful [7, Thm. 6.1]

- Want to interpolate $f(x) = \frac{1}{1+25x^2}$ on $[-1, 1]$
- Use evenly spaced points,
- Polynomials actually diverge as $n \to \infty$



Motivation for Chebyshev Polynomials

Polynomial Interpolation Error [3, Thm. 3.1.1]

The error of polynomial interpolation for a function that is $C^{n+1}[a, b]$:

$$f(x) - p_n(x) = \frac{f^{n+1}(c)}{(n+1)!} \prod_{j=1}^{n+1} (x - x_j),$$

Motivating Fact [3, Thm. 3.3.4]

For $[-1, 1]$, the Chebyshev Roots

$$x_j = \cos\left(\frac{2j-1}{2(n+1)}\pi\right), 1 \le j \le n+1$$

minimize the term $\max_{x\in[a,b]} \left|\prod_{j=1}^{n+1}(x - x_j)\right|$

Chebyshev Polynomials

Definition

$$T_n(x) = \cos\left(n \arccos(x)\right), \ x \in [-1, 1]$$

Recurrence Relation

$$T_0(x) = 1 \quad T_1(x) = x$$
$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$$

Recurrence allows for fast evaluation of Chebyshev polynomial expansions

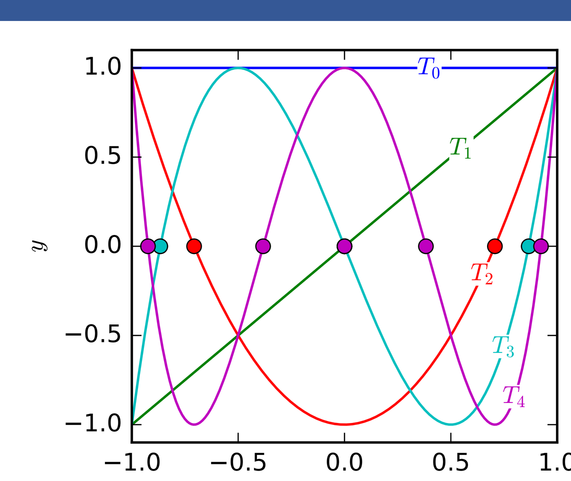


Figure: First 5 Chebyshev Polynomials

Useful Properties of Chebyshev Polynomials

Interpolation Convergence [11, Ch. 8]

- Suppose we want to interpolate $f : [-1, 1] \to \mathbb{R}$
- f is analytic $\implies$ geometric convergence
- Need a small degree expansion for machine precision in our example

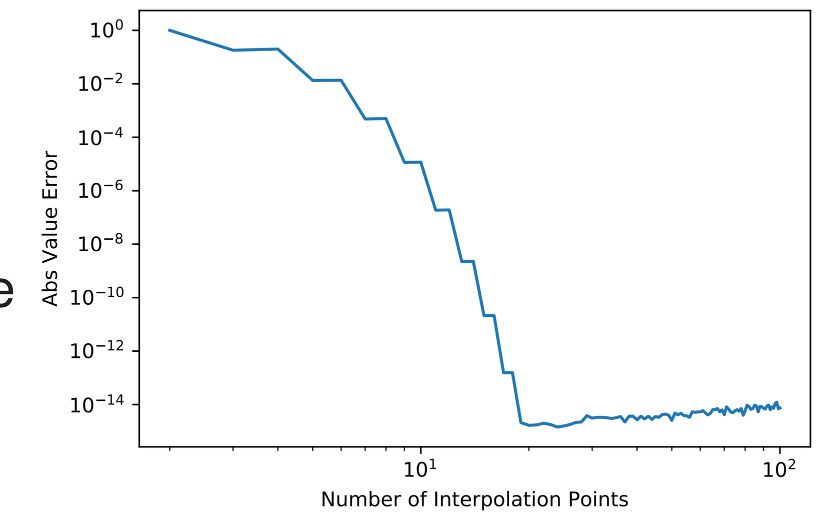


Figure: Error of Chebyshev Interpolation on $\sin(x)$ on $[0, 2\pi]$

Orthogonality

Chebyshev polynomials are orthogonal on $[-1, 1]$ with a weight

$$\int_{-1}^{1} \frac{T_n(x)T_m(x)}{\sqrt{1-x^2}} = 0 \quad \text{if } m \neq n$$

Discrete orthogonality relations allow for quick interpolation [7, Sec. 6.3].

ChebTools Introduction

- Written in C++11 with wrappers in Python using `pybind11` [6]
- Backend linear algebra with `Eigen` [5]

Chebyshev Expansion Object If our expansion looks like

$$\sum_{k=0}^{n} a_k T_k(x), \quad \text{on } [-1, 1]$$

then the object looks like

$$\texttt{xmin} = -1,\ \texttt{xmax} = 1,\ \texttt{coef} = [a_0, a_1, \dots, a_n]$$

Related Libraries

- `chebfun` [4]: Extensive MATLAB library by Nick Trefethen's group
- `ApproxFun.jl` [10]: Julia library by Alex Townsend
- `pychebfun`[9] and `chebpy`[8] for Python users

ChebTools Current Basic Capabilities

Generating Expansions

```
# arguments: (degree of expansion, function, xmin, xmax)
cheb_sin = ct.generate_Chebyshev_expansion(10, np.sin, 0, 2*np.pi)
```

Function Evaluation

```
# works with floats and numpy arrays
cheb_sin.y(3)
```

Arithmetic Operations

```
# addition by another expansion and a constant
# note that operators like +=, -, and -= are also supported
chebs_added = cheb_sin + cheb_cos + 1
# multiplication of two expansions and multiplication by a constant
chebs_sin2x = 2*cheb_sin *cheb_cos
```

Applying More Advanced Functions (from `pychebfun` [9])

```
# applying more complicated functions to a Chebyshev expansion object
sin_squared = cheb_sin.apply(lambda x: x**2)
```

Other Current Capabilities

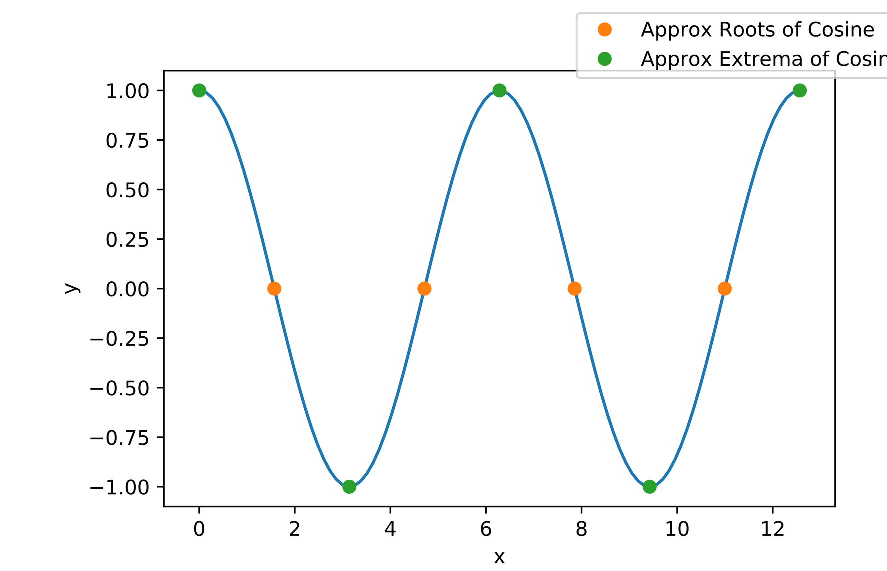
Root Finding, Derivatives, and Local Extrema

```
# generate expansion of cosine
long_cos = ct.generate_Chebyshev_expansion(25, np.cos, -.1, 4*np.pi+.1)

# finding the roots of the Chebyshev expansion
# the boolean argument being True means that we want just the roots in [
 xmin, xmax]
approx_roots = long_cos.real_roots(True)

# finding the local extrema of the Chebyshev expansion
approx_extrema = long_cos.deriv(1).real_roots(True)
```

- The method `real_roots` utilizes a companion matrix method [2]
- Faster method in the C++ library that avoids the companion matrix
- Has been used in [1]



Future Work

We are looking for help with development. We want to add additional capabilities to the library including:

- Integrating expansions
- 2-D capabilities
- Adaptive Interpolation
- Parallelization

More Info

JOSS Paper

Bell et al., (2018). ChebTools: C++11 (and Python) tools for working with Chebyshev expansions. *Journal of Open Source Software*, 3(22), 569. https://doi.org/10.21105/joss.00569

GitHub Link

https://github.com/usnistgov/ChebTools

Jupyter Notebook Example

https://github.com/usnistgov/ChebTools/blob/master/BattlesTrefethen.ipynb

References and Acknowledgements

[1] I.H. Bell and B.K. Alpert. Exceptionally reliable density-solving algorithms for multiparameter mixture models from chebyshev expansion rootfinding. *Fluid Phase Equilibria*, 2018.
[2] J.P. Boyd. Finding the zeros of a univariate equation: Proxy rootfinders, chebyshev interpolation, and the companion matrix. *SIAM Review*, 55(2):375–396, 2013.
[3] P. Davis. *Interpolation and Approximation.* Dover Publications, 1975.
[4] T. A Driscoll, N. Hale, and L. N. Trefethen. *Chebfun Guide.* Pafnuty Publications, 2014.
[5] B. Jacob and G. Guennebaud. Eigen, 2017. http://eigen.tuxfamily.org.
[6] W. Jakob, J. Rhinelander, and D. Moldovan. pybind11 – seamless operability between c++11 and python, 2016. https://github.com/pybind/pybind11.
[7] J.C. Mason and D.C. Handscomb. *Chebyshev Polynomials.* Chapman & Hall/CRC, 2003.
[8] M. Richardson. chebpy – a python implementation of chebfun, 2017. https://github.com/chebpy/chebpy.
[9] C. Swierczewski and O. Verdier. pychebfun – python implementation of chebfun, 2017. https://github.com/olivierverdier/pychebfun.
[10] A. Townsend. Approxfun.jl – julia package for function approximation, 2018. https://github.com/JuliaApproximation/ApproxFun.jl.
[11] L.N. Trefethen. *Approximation Theory and Approximation Practice (Other Titles in Applied Mathematics).* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2012.

This work was supported by the NIST SURF program.

Department of Mathematical Sciences, George Mason University, Fairfax, VA 22030 lbouck@gmu.edu