Precision Experiments in SDPB

Walter Landry
wlandry@caltech.edu

https://groups.google.com/forum/#!forum/bootstrap-collaboration-software

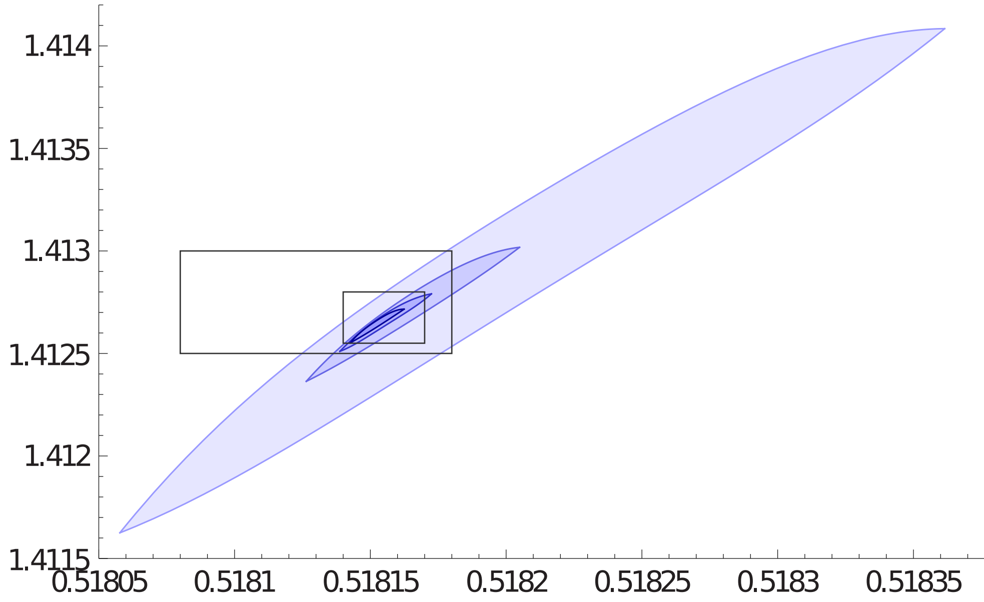
Caltech

The Bootstrap and Semidefinite Programs

- The conformal bootstrap can be formulated in terms of a semidefinite program.
- Semidefinite programs are generic math problems that occurs in many branchs of science and engineering.
- Existing, off-the shelf solver implementations exist in a variety of environments
  - Matlab, Mathematica, C, Python, ...

Why SDPB?

- Bootstrap calculations can require extreme numerical precision and computational resources.
  - Ising computations ran for weeks.
- SDPB is a solver optimized for bootstrapping.
  - Open-source
  - Arbitrary precision
  - Heavily parallelized



Very, Very, High Precision

- Unfortunately, bootstrap problems seem to require very, very high precision.
  - Ising bootstrap: 1216 bit mantissa
  - IEEE-754 Double precision: 53 bit mantissa
- Consequently, operations like addition and subtraction can take hundreds of times longer.
- They also use significantly more memory.

Why Such High Precision?

- I will be looking at a small stress tensor example. It seems non-trivial enough to be useful.
- You might expect to need only to resolve the error threshold: 10^{-30}
- In practice, we need much, much higher precision.

What Breaks?

- The first thing that breaks when reducing precision is when solving

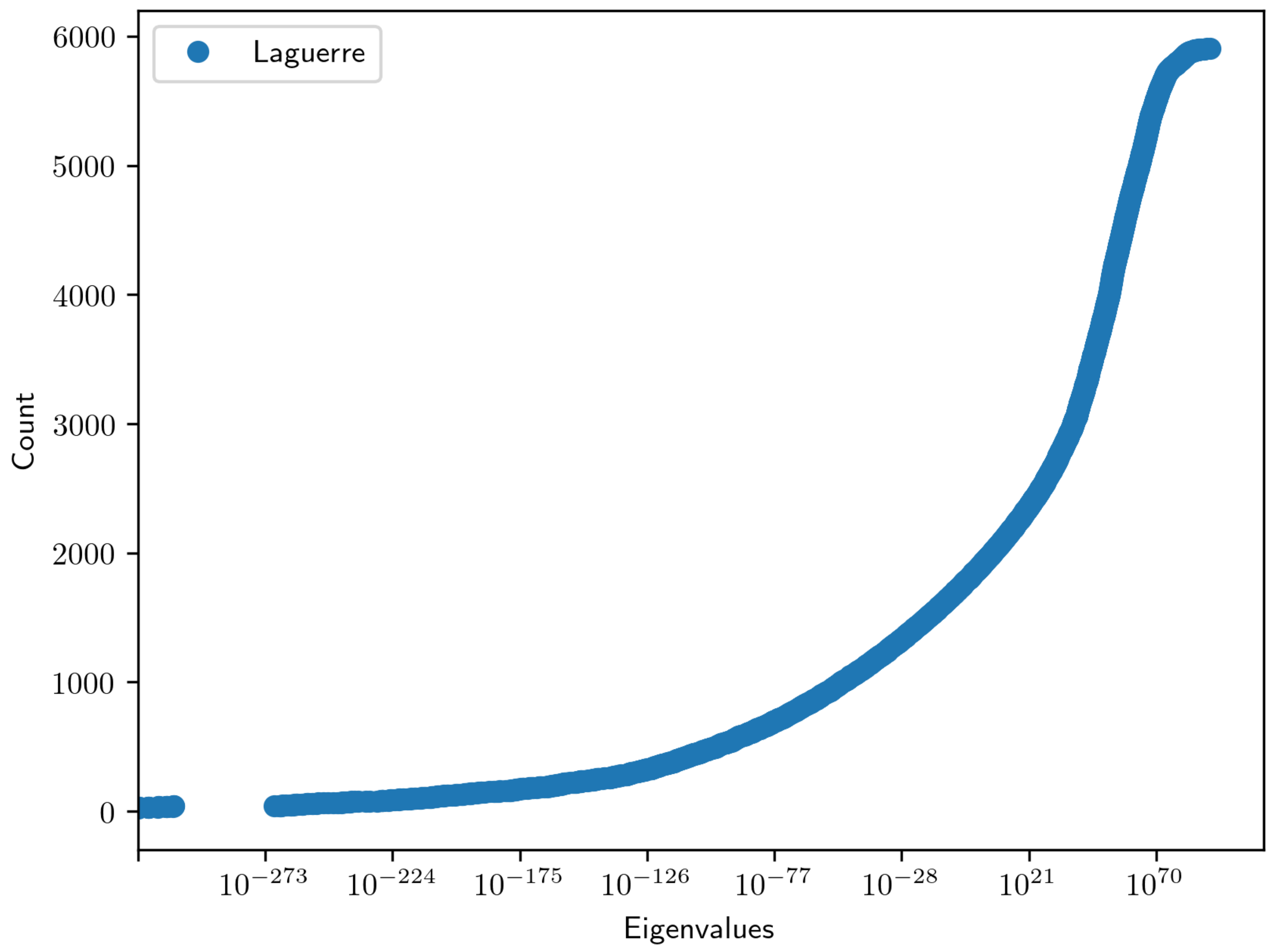
$$\begin{pmatrix} S & -B \\ B^T & 0 \end{pmatrix} \begin{pmatrix} dx \\ dy \end{pmatrix} = \begin{pmatrix} r_x \\ r_y \end{pmatrix}$$

- S has a block structure made up of symmetric positive-definite matrices.
- We use a Schur complement method, which involves inverting S first.

S is Ill-Conditioned

- When precision is low, S is no longer numerically positive.
- This is because S has a very bad condition number: 10^{180}
- This happens immediately, well before we do any real calculations.
- The condition number gets worse with more iterations, ending up at 10^{400} when SDPB finds a solution within tolerances.

Eigenvalue Spectrum of S



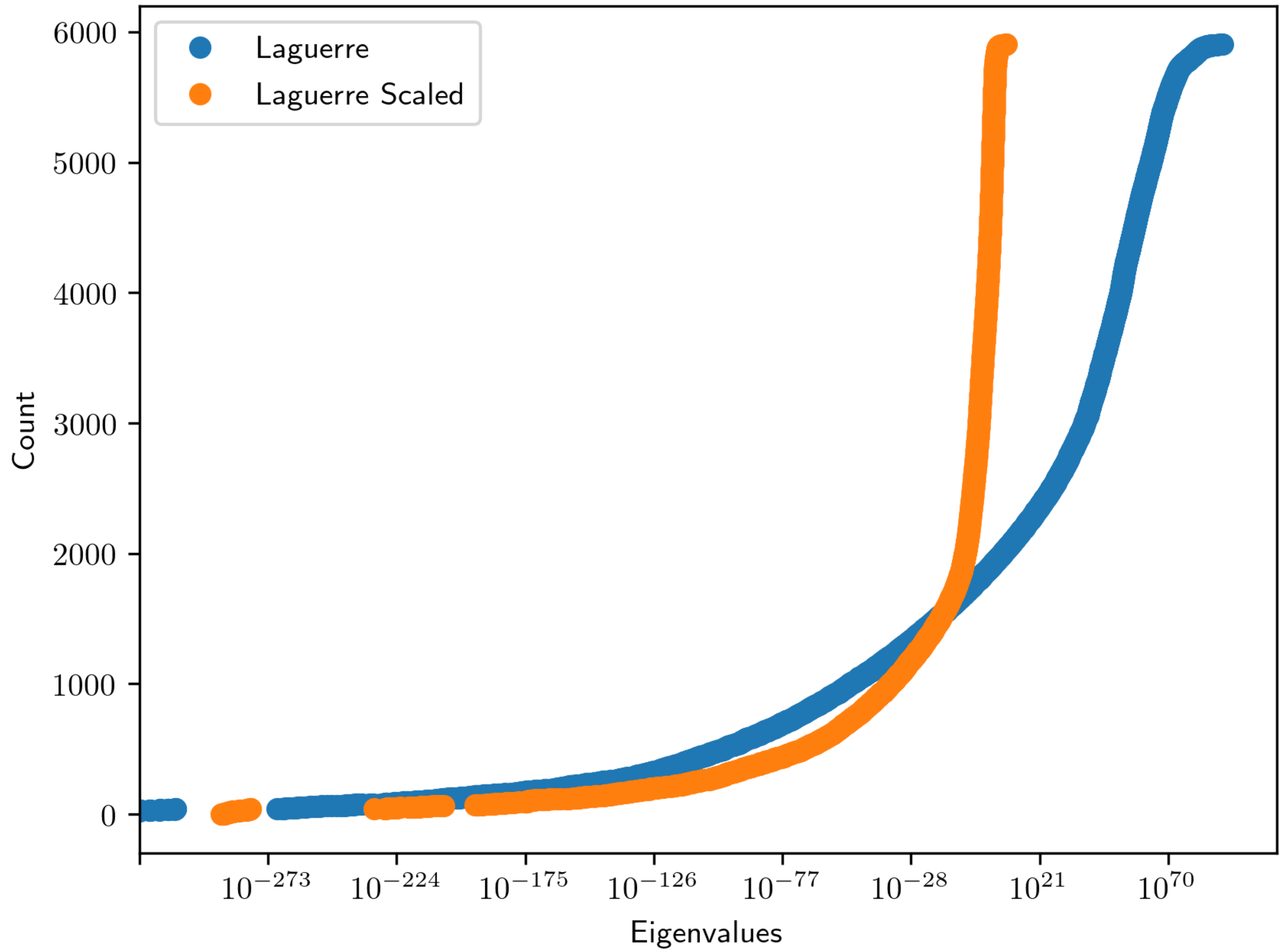
Eigenvalue Spectrum

- The eigenvalues smoothly vary from miniscule to gigantic. There is no natural break.
- This kind of structure is seen for all of the blocks. There is no individual bad block. Rather, they are all bad blocks.

Scaling Rows

- We have the freedom to scale each row of B independently.
- This also implies a scaling of the bilinear bases.
- This is usually done by looking at the size of the underlying Laguerre polynomials.
- Instead, try scaling each row by the max(|B|) on that row.

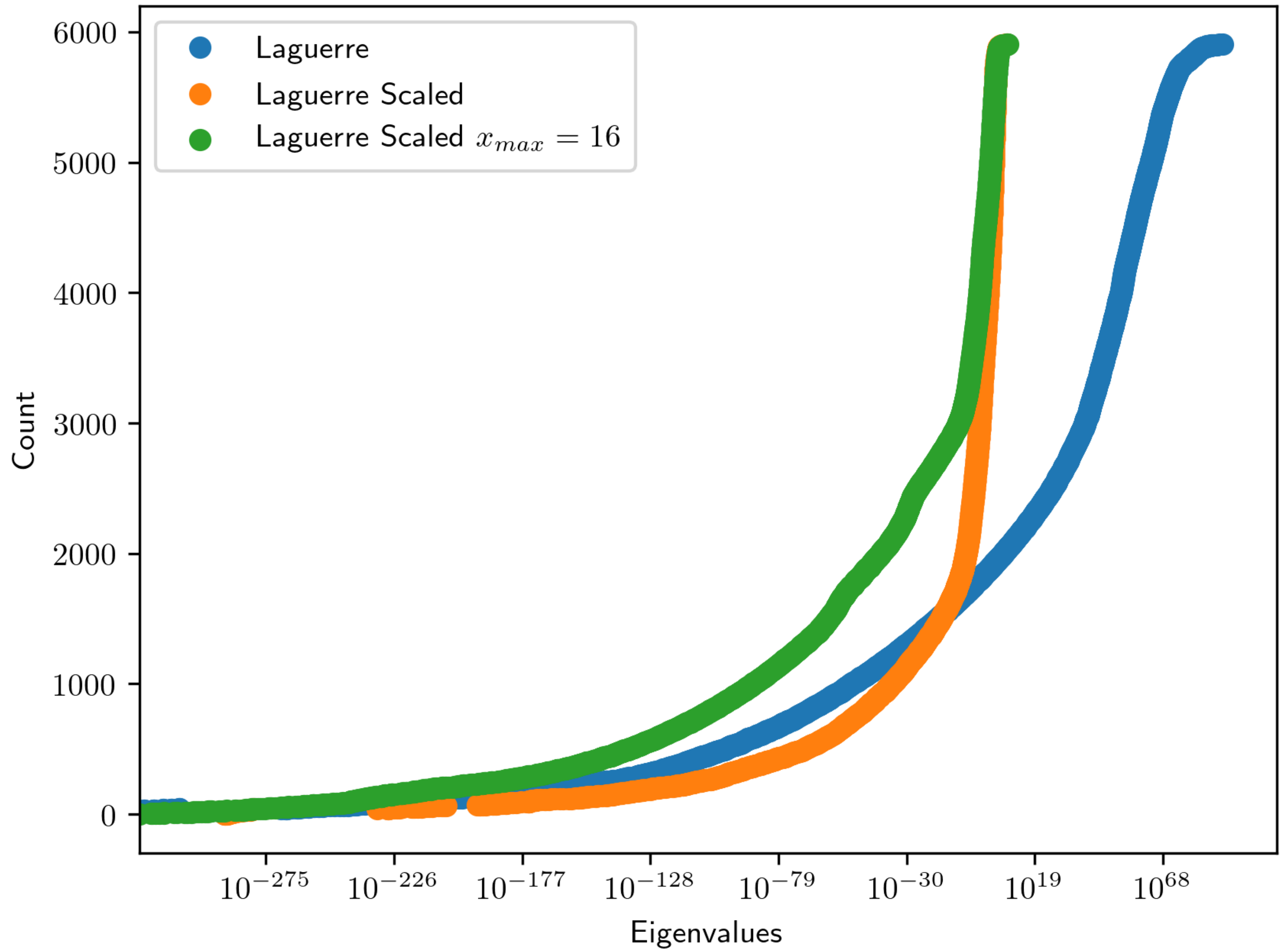
Eigenvalues with Row Scaling



Range of x

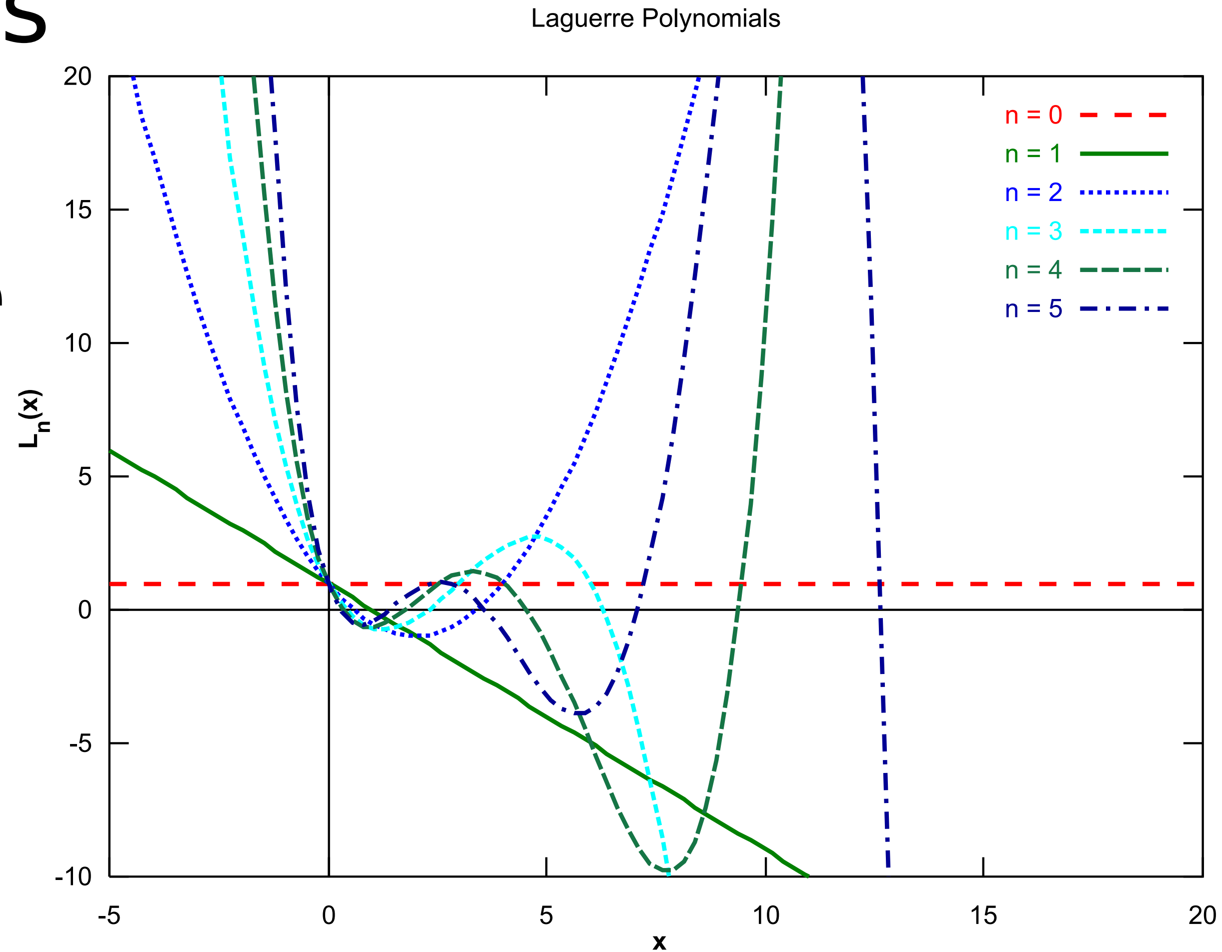
- The range of x (~77) is given by the largest root of the Nth Laguerre polynomial.
- For this problem, the largest root is at x=16, so try that instead.

Eigenvalues with a Smaller Range



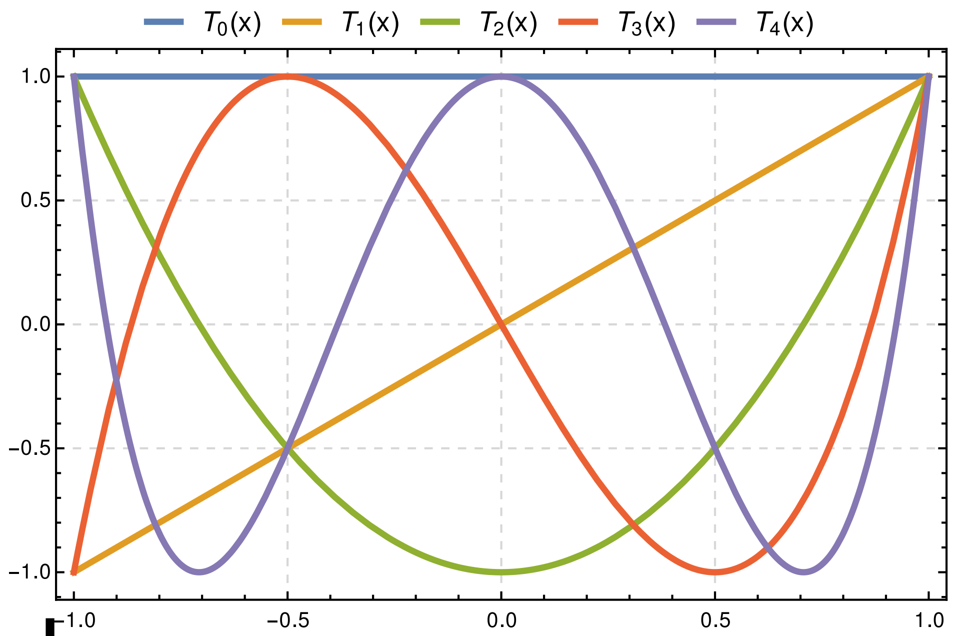
Bad Basis

- By default, we evaluate functions at the roots of Laguerre polynomials.
- Laguerre polynomials mimic exponentials, but the functions we are approximating are not as badly behaved over the domain.

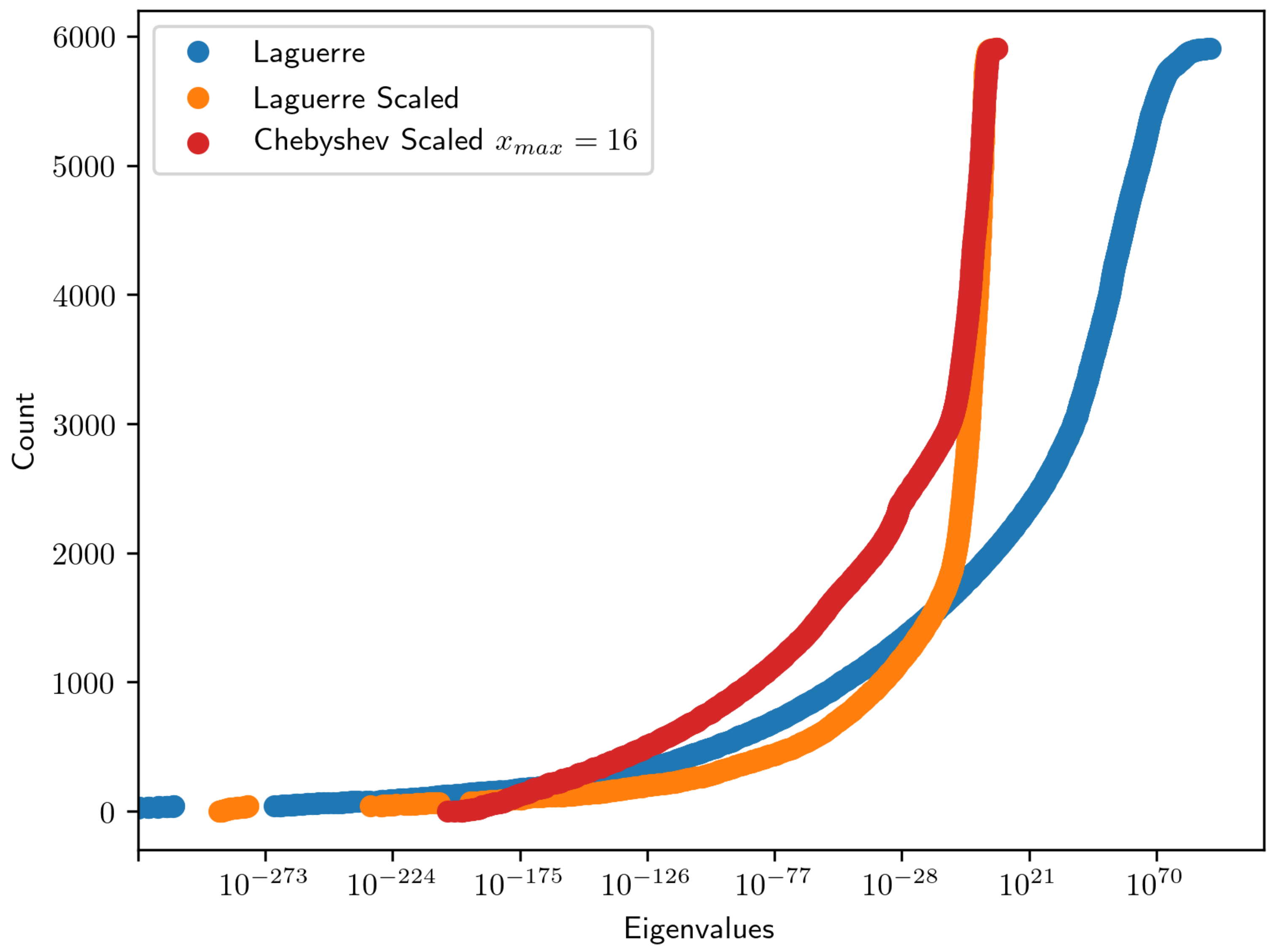


Chebyshev Polynomials

- Chebyshev polynomials are ***very*** well behaved in their domain.
- We tried mapping the Chebyshev roots to the same interval.
- This dramatically improved the initial condition number for S: 10^4



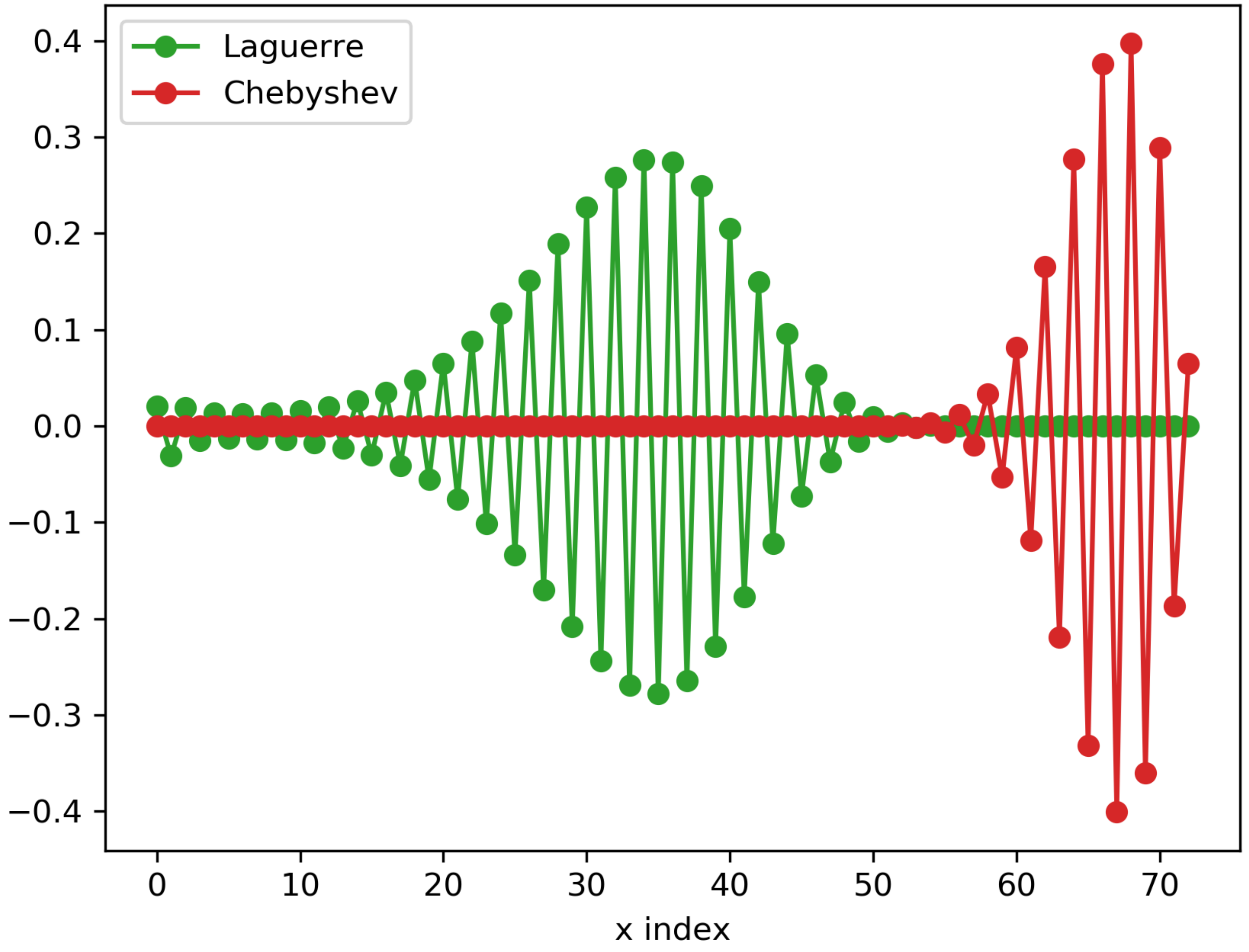
Chebyshev vs Laguerre



Chebyshev is better

- The gap threshold of 10^{-30} was achieved in fewer iterations: 300 rather than 500.
- This means the condition number only grew to 10^{300}.
  - So we can use lower precision 1280 -> 768
- The overall speedup is a factor of 3.

Eigenvectors of Smallest Eigenvalue



Eigenvector Structure in x

- With Laguerre zeros as a basis, the eigenvectors of small eigenvalues are scattered all over.
- With Chebyshev zeros as a basis, the eigenvectors of small eigenvalues are concentrated on the right (large x). As the eigenvalues get larger, the eigenvectors start sampling more on the left (small x).