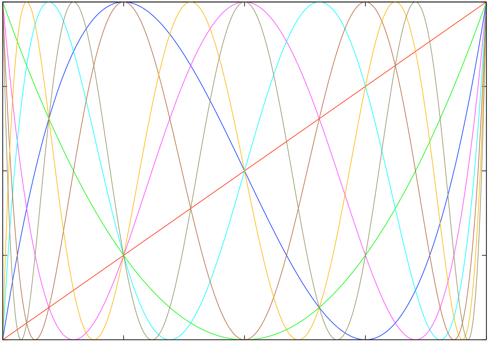
Chebyshev revisited

Björn von Sydow

Särö, February 4 2015.



General setting

The problem

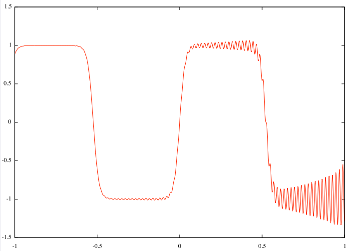
We consider real-valued functions on the standardized interval [-1,1]. Given f, we want to define a sequence of polynomials $\{P_n\}_{n=0}^{\infty}$, such that P_n converges uniformly to f as $n \to \infty$.

Motivation

We intend to use P_n instead of f for e.g.

- differentiation
- integration
- root-finding
- optimization
- ...

$f(x) = \tanh(5\sin(6x)) + 0.02e^{3x}\sin(300x)$



How can you find a polynomial approximant?

Idea 1: Taylor expansion

Use a Taylor polynomial around 0. For e.g. $f(x) = \sin(x)$, we get

$$P_1(x) = x$$
$$P_3(x) = x - x^3/6$$
$$P_5(x) = x - x^3/6 + x^5/120$$
$$\ldots$$

Idea 2: Interpolating polynomial

Given $n+1$ points

$$x_0 < x_1 < \ldots < x_n,$$

we can find interpolating P_n such that $P_n(x_k) = f(x_k)$ for all k.

For example, take $x_k = 2k/n - 1$.

Problems

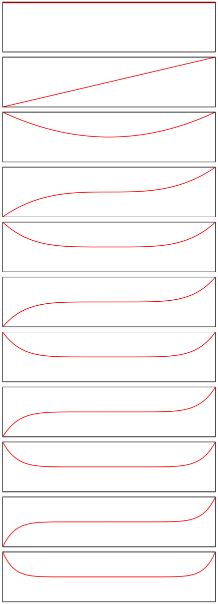
- Requires high order derivatives for good approximation.
- Uses only info at $x = 0$; need not converge in all [-1,1].

Problem

Will not converge throughout [-1,1] for most f.

Folklore: Polynomial interpolation is awful.

Monomials



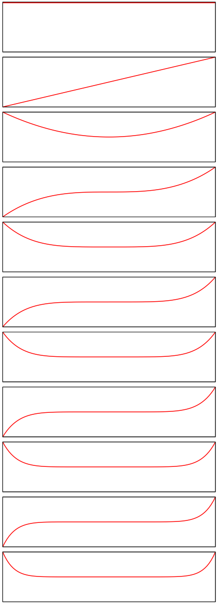
Monomials

To the left we have

$$M_n(x) = x^n,$$

for $n = 0, 1, \ldots 10.$

Monomials vs Chebyshev polynomials



Monomials

To the left we have

$$M_n(x) = x^n,$$

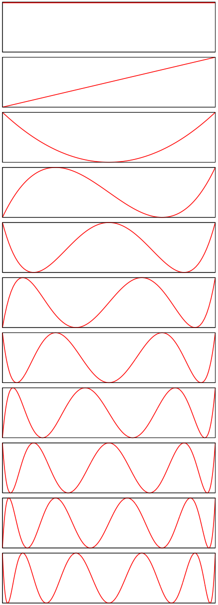
for $n = 0, 1, \dots 10.$

Chebyshev polynomials

To the right we have

$$T_n(x) = \cos(n \arccos x),$$

for $n = 0, 1, \dots 10.$



Are they polynomials at all?

Some first-year calculus

Recall

$$\cos(\alpha + \beta) = \cos\alpha\cos\beta - \sin\alpha\sin\beta,$$
$$\cos(\alpha - \beta) = \cos\alpha\cos\beta + \sin\alpha\sin\beta.$$

Add them and put $\alpha = n \arccos\, x$ and $\beta = \arccos\, x$ to get

$$T_{n+1}(x) + T_{n-1}(x) = 2xT_n(x)$$

Recursion scheme

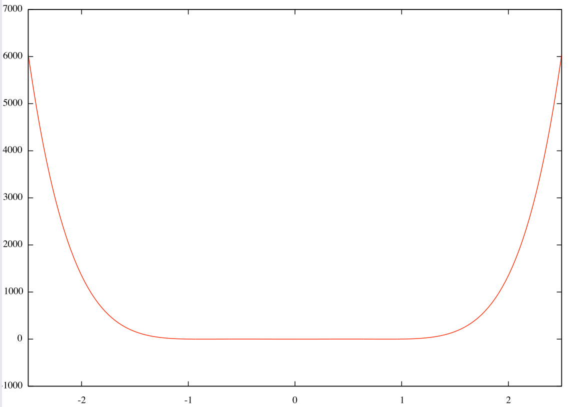
$$T_0(x) = 1,$$
$$T_1(x) = x,$$
$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x),\ \ n \geq 2$$

Conclusion

$T_n(x)$ is a polynomial of degree n with leading coefficient 2^{n-1} (for $n > 0$) and all n zeros in $[-1, 1]$.

The bigger picture

T_6 plotted over $[-2.5, 2.5]$.



Conclusion

Chebyshev polynomials are very useful inside [-1,1].

Chebyshev approximation

The problem

We intend to approximate f by $\sum_{k=1}^{n} c_k T_k$. How to define $\{c_k\}$?

Inspiration: Fourier analysis

Let $g(\theta) = f(\cos\theta), 0 \leq \theta \leq \pi$. We can expand g in a cosine series

$$g(\theta) = \frac{a_0}{2} + \sum_{k=0}^{\infty} a_k \cos(kx)$$

where

$$a_k = \int_0^{\pi} g(\theta)\cos(k\theta)d\theta.$$

with an extensive theory.

Back to our setting

$$\begin{aligned} f(x) &= g(\arccos\, x) \\ &= \frac{a_0}{2}T_0(x) + \sum_{k=0}^{\infty} a_k T_k(x). \end{aligned}$$

This is fine, but

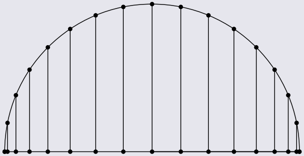
$$\begin{aligned} a_k &= \int_0^{\pi} g(\theta)\cos(k\theta)d\theta \\ &= \int_{-1}^{1} \frac{f(x)T_k(x)}{\sqrt{1-x^2}}dx. \end{aligned}$$

How do we compute these?

Chebyshev interpolation

Key insight 1

Use interpolation, but choose $\{x_k\}$ as the Chebyshev points: $x_k = \cos \frac{k\pi}{n}, k = 0, \ldots n.$



Key insight 2

Represent P_n in the Chebyshev basis, i.e. as $P_n = \sum_{k=0}^{n} c_k T_k$.

We need to compute $\{c_k\}$ so that $P_n(x_k) = f(x_k)$ for all k.

Convergence

- If $f \in C^k([-1, 1])$, then $||f - P_n||_\infty = O(n^{-k})$.
- If f is analytic, then $||f - P_n||_\infty = O(\rho^{-n})$ for some $\rho > 1$.

Computation

- $[c_0, c_1, \ldots c_n]$ is the inverse DCT of $[f(x_0), f(x_1), \ldots f(x_n)]$.
- There are simple, linear, stable recursions for computing
  - $P_n(x)$ (Clenshaw's algorithm).
  - $P_n'(x)$.
  - $\int^x P_n(t)dt$.

Misconceptions

Quotations from numeric analysis textbooks

- *Polynomial interpolants rarely converge [...]. Polynomial interpolation is a bad idea. (1989)*
- *In this section we consider examples which warn us of the limitations of using interpolation polynomials [...]. (1996)*
- *The surprising state of affairs is that for most continuous functions, the quantity* $||f - p_n||_\infty$ *will not converge to 0. (2002)*
- *By their very nature, polynomials of a very high degree do not constitute reasonable models for real-life phenomena, from the approximation and from the handling point-of-view. (2004)*
- *The oscillatory nature of high degree polynomials [...] restricts their use. (2005)*
- *In addition to the inherent instability [...] there are also classes of functions that are not suitable for [...] interpolation. (2011)*

Root-finding

An essential tool

We use the roots of the Chebyshev approximant in several important ways, e.g.:

- for finding minima and maxima we need roots of the derivative.
- for splitting into low-order pieces when approximating the absolute value of an expression we need the roots.

Unfortunately this is often thought to be problematic:

- *Our main object in this chapter has been to focus attention on the severe inherent limitations of all numerical algorithms for finding the zeros of polynomials.* (Wilkinson 1963).
- *Speaking for myself I regard it as the most traumatic experience in my career as a numerical analyst.* (Wilkinson 1984, The perfidious polynomial.)

Root-finding, cont'd

Colleague matrices

Given $P_n = \sum_{k=0}^{n} c_k T_k$, we define the colleague matrix

$$\begin{pmatrix} 0 & 1 & & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & & \\ & \frac{1}{2} & 0 & \frac{1}{2} & & \\ & & \ddots & \ddots & \ddots & \\ & & & & & \frac{1}{2} \\ & & & & \frac{1}{2} & 0 \end{pmatrix} - \frac{1}{2c_n} \begin{pmatrix} & & & & \\ & & & & \\ & & & & \\ & & & & \\ c_0 & c_1 & c_2 & \dots & c_{n-1} \end{pmatrix}.$$

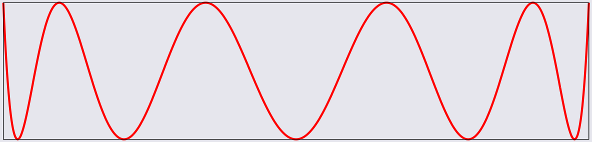
Simple fact

The zeros of P_n are the eigenvalues of its colleague matrix.
Fast and stable algorithms exist (e.g., QR algorithm).

Best approximation

Equioscillation theorem

A continuous function f on $[-1, 1]$ has a unique best approximation P_n^* (i.e. a unique polynomial minimizing $||f - P_n||_\infty$).
The error term $f - P_n^*$ attains it maximal absolute value at least $n - 2$ times with alternating signs.



Almost as good

Let C_n be the Chebyshev interpolant of degree n. Then

$$||f - C_n||_\infty < \left(2 + \frac{2}{\pi}\log(n+1)\right) ||f - P_n^*||_\infty.$$

Quadrature

Gaussian quadrature

The problem: For given n, find $\{x_k\}_{k=0}^{n}$ and weights $\{\lambda_k\}_{k=0}^{n}$ such that

$$\int_{-1}^{1} f(x)dx = \sum_{k=0}^{n} \lambda_k f(x_k)$$

for all polynomials f of degree $\leq 2n+1$.

The solution: Gaussian quadrature. The x_k are the zeros of the Legendre polynomial of degree $n+1$.

Common wisdom: Mostly of theoretical interest. Most software uses adaptive Newton-Cotes rules with Richardson extrapolation.

Recent development

Linear, stable algorithms for computing abscissas and weights, so high-order Gaussian quadrature is feasible and competitive.

Clenshaw-Curtis quadrature

Gauss is optimal but Clenshaw-Curtis is better

Fix x_k to be the Chebyshev point $\cos \frac{k\pi}{n}$ and determine the weights to make the integral exact for polynomials of degree $\leq n + 1$.
A simple computation of $\int_{-1}^{1} T_k(x)dx$ gives

$$\int_{-1}^{1} \sum_{k=0}^{n} c_k T_k(x) dx = \sum_{k=0,\, k \text{ even}}^{n} \frac{2c_k}{1 - k^2}.$$

Note that this is exact. So, if we have approximated f to machine precision, the integral will also be to machine precision.

Is Haskell suitable for this?

Basic ingredients

We need

- efficient vectors; `Data.Vector.Storable` seem fine.
- state-of-the-art FFT algorithms; access to FFTW via the Haskell FFI is fine.
- state-of-the-art eigenvalue computation; access to LAPACK via Haskell FFI is fine.

Given this infrastructure, Haskell and the interactive environment `ghci` is excellent.

Is a Haskell implementation of Chebfun interesting?

- The applied math community is certainly happy with Matlab.
- Implementing an inferior version of a publicly available package is dubious.

\+ But I have basic funding and am free to do what I think is fun!

Further reading

- Nick Trefethen: Six myths about polynomial interpolation and quadrature. Summer Lecture at the Royal Society 2011. Easily found by googling. Compulsory reading!
- Nick Trefethen: Approximation Theory and Approximation Practice. SIAM Press 2013.