Interpolation

Part-2

Orthogonal Polynomials

- Orthogonal polynomials have many useful properties.
    - The elegant theory behind is beyond the scope of this class.
    - We will make use of some of those properties
    - Orthogonal polynomials are particularly convenient for least squares approximation of a given function
    - Orthogonal polynomials are useful for generating Gaussian quadrature rules
    - Can represent an arbitrary but smooth function efficiently

Example: *three-term recurrence* property makes generation and evaluation of polynomials very efficient

$$\boxed{p_{k+1}(t) = (\alpha_k t + \beta_k)p_k(t) - \gamma p_{k-1}(t)}$$

Orthogonal Polynomials

Inner product (generalization of dot product) of two polynomials *p* and *q* on an interval [*a*,*b*]

Inner product: $\langle p, q \rangle = \int_a^b p(t)q(t)w(t)dt$

where $w(t)$ is nonnegative weight function

Polynomials *p* and *q* are said to be orthogonal if $\langle p, q \rangle = 0$

A set of polynomials $\{p_i\}$ is said to be orthonormal if

$$\langle p_i, p_j \rangle = \begin{cases} 1 & \text{for} \quad i = j \\ 0 & \text{for} \quad i \neq j \end{cases}$$

Methods, such as Gram-Schmidt, process can be used for orthogonalization

Example: Legendre polynomials are derived by orthogonalization of monomial polynomials $(1, t, t^2, t^3, \dots)$

Orthogonal Polynomials

Recall $\langle p, q \rangle = \int_a^b p(t)q(t)w(t)dt$

Name	Symbol	Interval	Weight function w(t)
Legendre	P_k	[-1,1]	1
Chebyshev, 1st kind	T_k	[-1,1]	$(1-t^2)^{-1/2}$
Chebyshev, 1st kind	U_k	[-1,1]	$(1-t^2)^{1/2}$
Jacobi	J_k	[-1,1]	$(1-t)^\alpha(1+t)^\beta, \quad \alpha, \beta > 1$
Laguerre	L_k	[-1,1]	e^{-t}
Hermite	H_k	[-1,1]	e^{-t^2}

Chebyshev Polynomials

- Orthogonal polynomials
- Useful for interpolation of <u>continuous</u> functions
- Close to the best polynomial approximation to continuous function

Recall $\langle p, q \rangle = \int_a^b p(t)q(t)w(t)dt$

The kth Chebyshev polynomial of the 1st kind on the interval [-1,1]

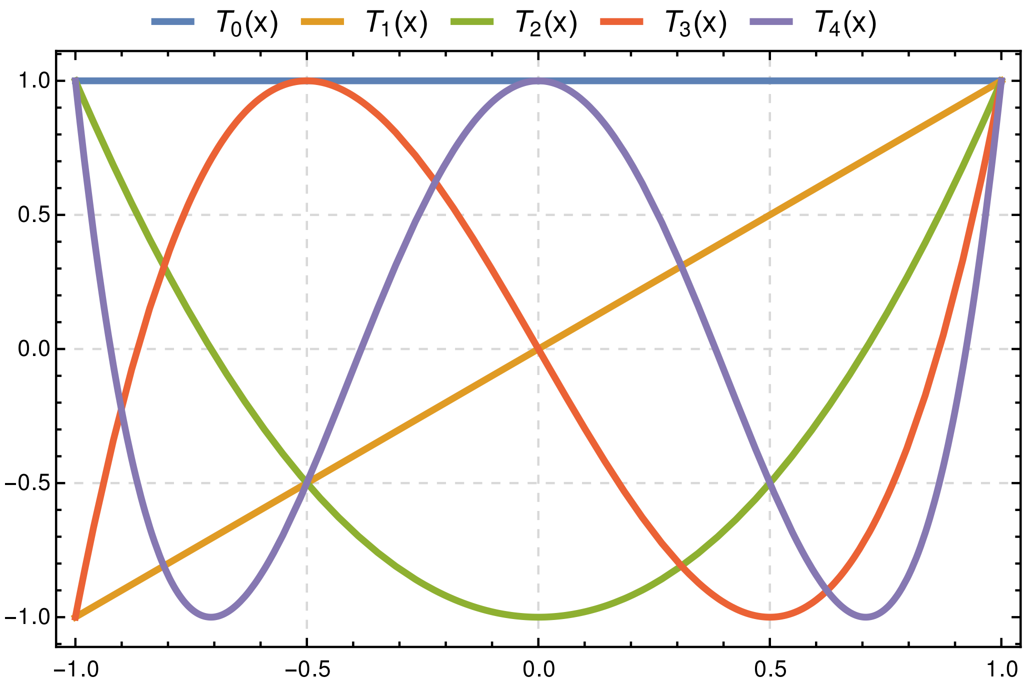
$$T_k(t) = \cos(k \arccos t)$$

With the three-term recurrence property

$$T_{k+1}(t) = 2\, t\, T_k(t) - T_{k-1}(t) \qquad k=0,1,2,\ldots$$

In-class exercise: Form the first six Chebyshev polynomials and plot them on the interval [-1,1]

T_o	1
T_1	t
T_2	
T_3	
T_4	
T_5	



Observations:

- Degree k polynomial contains only terms of even degree if k is even, odd terms if k is odd
- Highest degree term has a coefficient 2^{k-1}
- Equi-alternation or equi-oscillation: successive extrema are equal in magnitude and alternate in sign
    - Distribute error uniformly when used to approximate continuous functions
        - Maximum error over an interval is minimized if interpolation points are the extrema of a Chebyshev polynomial (Chebyshev points)

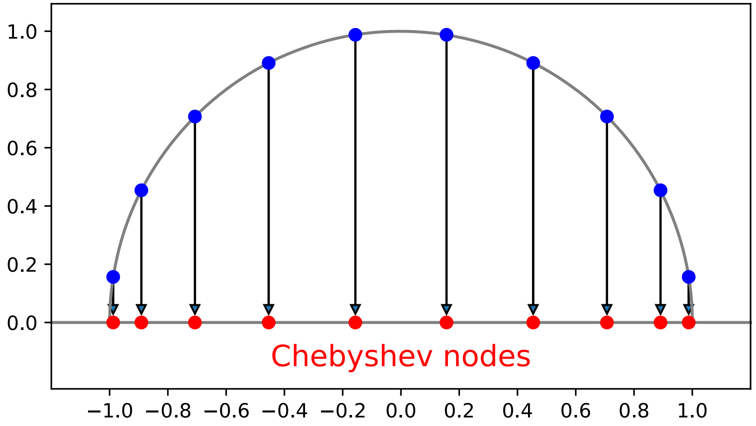
Chebyshev Points (Nodes)

k^{th} Chebyshev polynomial $\quad T_k(t) = \cos(k \arccos t)$

Its *k* zeros are given by $\quad t_i = \cos\left(\frac{(2i-1)\pi}{2k}\right), \qquad i = 1,2,\dots k$

Its *k+1* extrema (including end points) are given by $\quad t_i = \cos\left(\frac{i\pi}{k}\right), \qquad i = 0,1,2,\dots k$

Chebyshev points are the abscissas of points in the plane that are equally spaced around the unit circle



Interpolation of Continuous Functions

Let us assume the interpolating points (data) are obtained from a continuous function

Let us assume that we fit a polynomial to those points

How closely the interpolant approximates the given function between the points?

n points represented by a polynomial of *n-1* degree, the error is bounded as

$$\max|f(t) - p_{n-1}(t)| \leq \frac{Mh}{n}$$

Basis Splines (B-splines)

Goal is to represent an arbitrary spline with linear combination of basis functions

B-splines can be defined through recursion, convolution, or divided differences

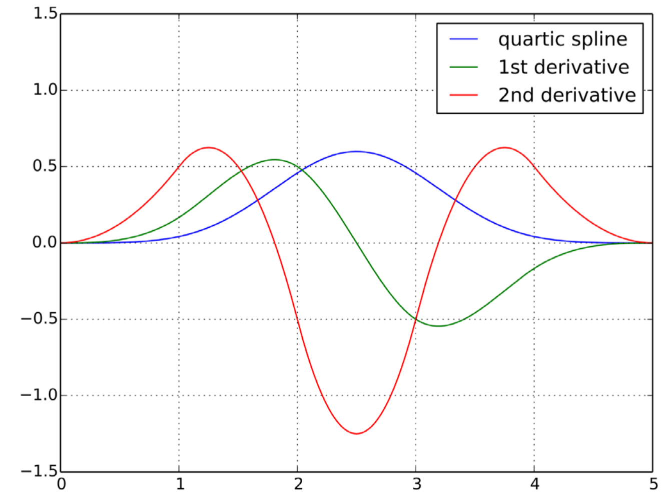
B-spline of degree *k* is *k-1* times continuously differentiable

B-splines (and Bézier curves) are used to create and manage complex shapes and surfaces using several points.

B-splines are extensively used in shape optimization

In computer aided design (CAD), computer aided manufacturing (CAM), and computer graphics, a powerful extension of B-splines is non-uniform rational B-splines (NURBS).

B-splines



$B_i^4(t)$

Linear functions: $$v_i^k(t) = \frac{t - t_i}{t_{i+k} - t_i}$$

B-spline of degree zero: $$B_i^o(t) = \begin{cases} 1 & \text{if } t_i \leq t \leq t_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

B-spline of degree *k* $$B_i^k(t) = v_i^k(t)\, B_i^{k-1}(t) + \left(1 - v_{i+1}^k(t)\right)\, B_{i+1}^{k-1}(t)$$

$B_i^k(t)$ is a piecewise polynomial of degree *k*

When data value at a given knot changes, coefficients of only a few basis functions change, whereas in standard polynomial, the entire interpolant is affected

Horner's Rule

$$p(x) = c_1 + c_2 x + \cdots c_n x^{n-1}$$

An efficient way to evaluate such polynomials is through nested multiplication instead of calling the function pow(x,n)

$$p(x) = \Big(\big((c_n x + c_{n-1})x + c_{n-2}\big)x + \cdots + c_2\Big)x + \ c_1$$

```
"""
    horner(c,x)

Evaluate a polynomial whose coefficients are given in ascending
order in `c`, at the point `x`, using Horner's rule.
"""
function horner(c,x)
    n = length(c)
    y = c[n]
    for k in n-1:-1:1
        y = x*y + c[k]
    end
    return y
end
```

Multi-dimensional Interpolation

What if the function we want to interpolate represents a surface?

Example: Consider a photograph with many pixels.

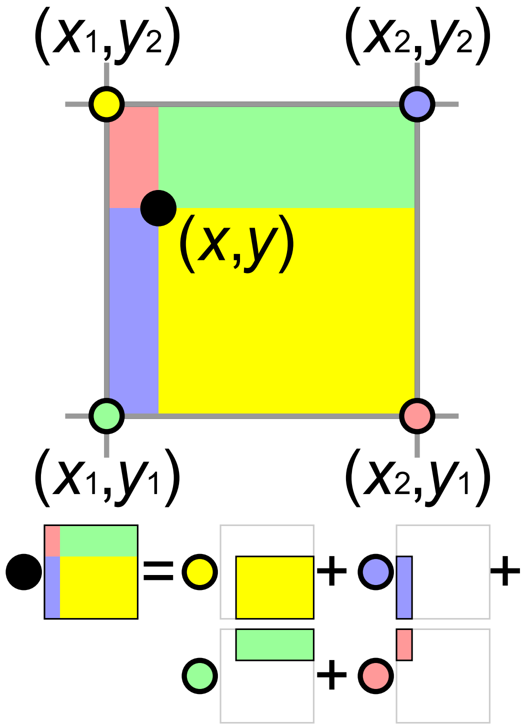
$$p(x_i, y_i) = f_i, \qquad\qquad i = 0,1,2, \dots n$$

$$p(x, y) = \sum_{k=0}^{N_x} \sum_{l=0}^{N_y} c_k \phi_k(x) \psi_l(y)$$

$\phi(x)$ & $\psi\,(y)$ are one-dimensional basis functions

Bilinear Interpolation

$$p(x, y) = c_0 + c_1 x + c_2 y + c_3 xy$$



Can be calculated through different ways

1. Polynomial fit
2. Repeated linear interpolation
3. Weighted mean

Multivariate Interpolation

- Trilinear interpolation
  - *n*-linear interpolation for any dimensions
- Nearest neighbor
- Kriging
- Inverse distance weighting
- Radial basis functions
- Delaunay triangulation
  - Construct a triangulation for the scattered data
  - Construct a piecewise polynomial interpolant for each triangle
- and several other methods …