Computational Methods

Interpolation

Interpolation

- Computing functions and solving equations (and systems of equations) are used solve problems on a known system model (represented by the system of equations)
  - Function calculations compute the output of the system
  - Solving equations computes parameter settings for a given output
- Interpolation is used to determine the system model from a number of data points
  - Estimates system equations from parameter/output data pairs

Interpolation

- Interpolation is aimed at determining *f(x)* from data points (x_i, y_i) such that
  - $f(x_i)=y_i$ (Interpolant fits the data points perfectly)
- Often additional constraints or requirements are imposed on the interpolant (interpolating function *f (x)*)
  - Desired slope
  - Continuity,
  - Smoothness
  - Convexity

Interpolation

- Interpolation is useful for a number of applications where only data points are given
  - Filling in unknown data points
  - Plotting smooth curves through data points
  - Determining equations for an unknown system
- Interpolation can also be used to simplify or compress information
  - Replacing a complicated function with a simpler approximation
  - Compressing complex data into a more compact form
- Interpolation is not for data with significant error
  - Approximation / optimization is more appropriate for this

Interpolation

- Generally there are an infinite number of interpolation functions for a set of data points
  - The choice of interpolation function should depend on the type and characteristics of the data
    - Monotonicity ? Convexity ?
    - Is data periodic ?
    - What behavior between data points ?
  - Choice of function can also be influenced by desired properties of the function
    - Will function be integrated or differentiated ?
    - Will function be used for equation solving ?
    - Is the result used for solving equations or visual inspection ?

Interpolation

- Commonly used families of interpolation functions
  - Polynomials
  - Piecewise polynomials
  - Trigonometric functions
  - Exponential functions
- Families of interpolation functions are spanned by a set of basis functions $\phi_i(x)$
  - Interpolating function can be computed as a linear combination of basis functions

$$f(x) = \sum_{i=1}^{n} \alpha_i \phi_i(x)$$

Interpolation

- The interpolation constraints can be defined
  $$f(x_j) = \sum_{i=1}^{n} \alpha_i \phi_i(x_j) = y_j$$
  - Constraints represent a system of linear equations
    $$A\vec{\alpha} = \vec{y} \quad , \quad a_{j,i} = \phi_i(x_j)$$
    - Solution to the linear system is the vector of coefficients
- Existence and uniqueness of interpolant depends on the number of points and basis functions
  - Too many data points means usually no interpolant exists
  - Too few data points means no unique solution exists
  - If there are as many data points as basis functions the system has a unique solution if A is not singular

Sensitivity and Conditioning

- Sensitivity of the parameters in the interpolation with respect to perturbations in the data depends on the sensitivity of the solution of the system of linear equations, *cond(A)*
  - Sensitivity depends on data points (and thus original function)
  - Sensitivity depends on the choice of basis functions

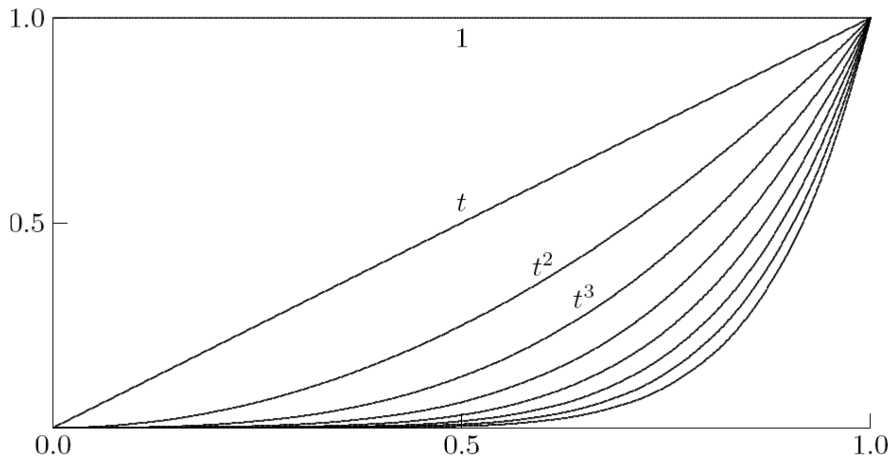
Polynomial Interpolation

- Polynomial Interpolation is the simplest and most common type of interpolation
  - Basis functions are polynomials
  - There is a unique polynomial of degree at most *n-1* that passes through *n* distinct data points
- There is a wide range of basis polynomials that can be used
  - All interpolating polynomials have to be identical independent of the basis chosen
    - Different polynomial bases might have different complexities for interpolation or prodice different rounding errors during calculation

Monomial Basis

- The most obvious basis choice for polynomial interpolation are monomial basis functions

$$\phi_i(x) = x^{i-1}$$



- Interpolating polynomial takes the form

$$p_{n-1}(x) = \alpha_1 + \alpha_2 x + \alpha_3 x^2 + \cdots + \alpha_n x^{n-1}$$

Monomial Basis

- The interpolating polynomial can be computed by solving for the constraints given by the data points
  - Resulting linear system to resolve parameters is described by the Vandermonde matrix

$$A = \begin{pmatrix} 1 & x_1 & \cdots & x_1^{n-1} \\ 1 & x_2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \cdots & x_n^{n-1} \end{pmatrix}$$

  - Solution of the interpolation problem requires solving the linear system of equations
    - Interpolation with monomials takes $O(n^3)$ operations

Evaluating Monomial Interpolant

- To use the interpolating polynomial it's value has to be calculated

$$p_{n-1}(x) = \alpha_1 + \alpha_2 x + \alpha_3 x^2 + \cdots + \alpha_n x^{n-1}$$

  - This can be made more efficiently using Horner's nested evaluation scheme

$$p_{n-1}(x) = \alpha_1 + x(\alpha_2 + x(\alpha_3 + x(\cdots x(\alpha_{n-1} \_ \alpha_n x)\cdots)))$$

    - *O(n)* multiplications and additions

- Other operations such as differentiation are relatively easy using a monomial basis interpolant

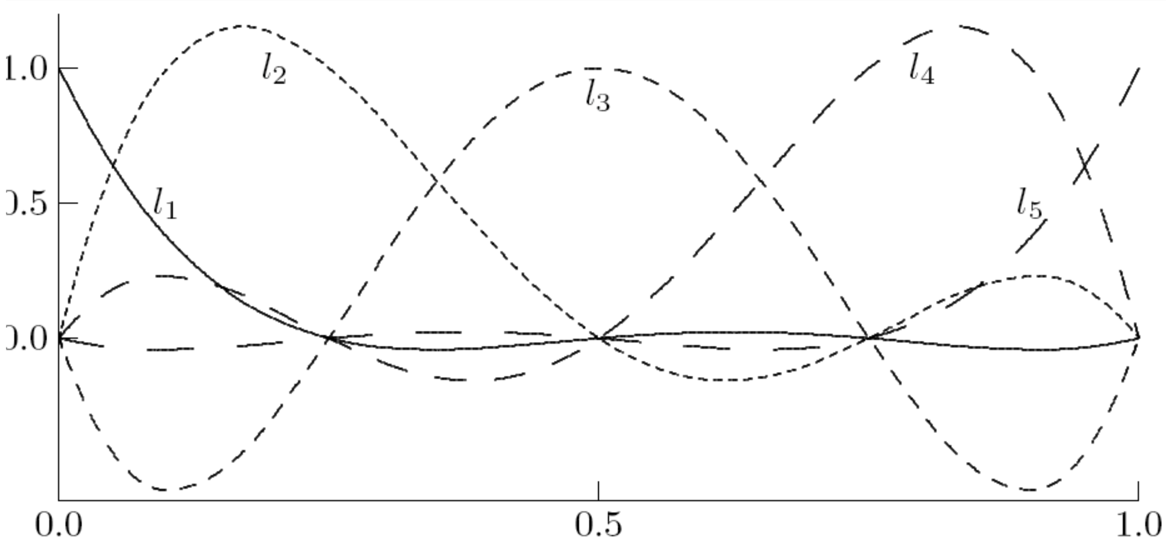
Monomial Basis

- Parameter solving for monomial basis becomes increasingly ill conditioned as the number of data points increases
  - Data point fitting is still precise
  - Weight parameters can only be determined imprecisely
- Conditioning can be improved by scaling the polynomial terms $$\phi_n(x) = \left(\frac{x - (\min_i x_i + \max_i x_i)/2}{(\max_i x_i - \min_i x_i)/2} \right)^{n-1}$$
- Choice of other polynomial basis can be even better and reduce complexity of interpolation

Lagrange Basis

- Lagrange basis functions are n-1th order polynomials

$$\phi_i(x) = \prod_{j=1, i\neq j}^{n} (x - x_j) \Big/ \prod_{j=1, i\neq j}^{n} (x_i - x_j)$$



Lagrange Basis

- For the Lagrange basis the linear system describing the data constraints becomes simple

$$\phi_i(x_j) = \begin{cases} 1 & if \quad i = j \\ 0 & otherwise \end{cases}$$

  - A is the identity matrix and therefor the Lagrange interpolant is easy to determine
  - Interpolating polynomial takes the form

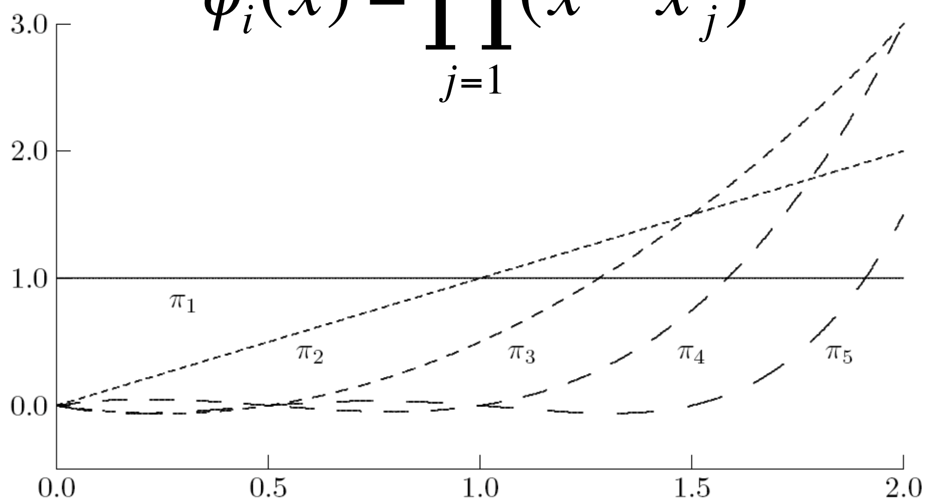
$$p_{n-1}(x) = y_1\phi_1(x) + y_2\phi_2(x) + \cdots + y_n\phi_n(x)$$

- Lagrange interpolant is difficult to evaluate, differentiate, integrate, etc.

Newton Basis

- Newton basis functions are i^{th} order polynomials

$$\phi_i(x) = \prod_{j=1}^{i-1} (x - x_j)$$



- Interpolating polynomial has the form

$$p_{n-1}(x) = \alpha_1 + \alpha_2(x - x_1) + \alpha_3(x - x_1)(x - x_2) + \cdots + \alpha_n(x - x_1)\cdots(x - x_{n-1})$$

Newton Basis

- For the Newton basis the linear system describing the data constraints is lower triangular

$$\phi_i(x_k) = \begin{cases} \prod_{j=1}^{i-1}(x_k - x_j) & if \quad k \geq i \\ 0 & otherwise \end{cases}$$

  - The interpolation problem can be solved by forward substitution in *$O(n^2)$* operations

- Polynomial evaluation can be made efficient in the same way as for monomial basis (Horner's method) and is easier to differentiate and integrate

Newton Basis

- Newton interpolation can be computed iteratively

$$p_n(x) = p_{n-1}(x) + \alpha_{n+1}\phi_{n+1}(x)$$

  - The coefficient is a function of the old polynomial without the additional data point and the data point

$$\alpha_{n+1} = \frac{y_{n+1} - p_{n-1}(x_{n+1})}{\phi_{n+1}(x_{n+1})}$$

  - Incremental construction starts with a constant polynomial representing a horizontal line through the first data point

$$p_0(x) = y_1$$

Newton Basis

- Newton interpolating functions can also be constructed incrementally using divided differences

$$d(x_i) = y_i$$

$$d(x_1, x_2, \ldots, x_k) = \frac{d(x_2, \ldots, x_k) - d(x_1, x_2, \ldots, x_{k-1})}{x_k - x_1}$$

  - The coefficients are defined in terms of the divided differences as

$$\alpha_n = d(x_1, \ldots, x_n)$$

  - Iterative interpolation takes $O(n^2)$ operations

Orthogonal Polynomials

- Orthogonal polynomials can be used as a basis for polynomial interpolation
  - Two polynomials are orthogonal if their inner product on a specified interval is 0

$$\langle p, q \rangle = \int_a^b p(x)q(x)w(x)dx = 0$$

  - A set of polynomials is orthogonal if any two distinct polynomials within it are orthogonal
- Orthogonal polynomials have useful properties
  - Three-term recurrence:

$$p_{k+1}(x) = (\beta_k x + \gamma_k)p_k(x) - \eta_k p_{k-1}(x)$$

Orthogonal Polynomials

- Legendre polynomials form an orthogonal basis for interpolation and are derived for equal weights of 1 and the base set of monomials over the interval [-1.1]

$$\phi_1(x) = 1 \ , \ \phi_2(x) = x \ , \ \phi_3(x) = (3x^2 - 1)/2$$

$$\phi_4(x) = (5x^3 - 3x)/2 \ , \ \phi_5(x) = (35x^4 - 30x^2 + 3)/8$$

$$\phi_{n+1}(x) = (2n+1)/(n+1)x\phi_n(x) - n/(n+1)\phi_{n-1}(x)$$

- Other weight functions yield other orhogonal polynomial bases
  - Chebyshev
  - Jacobi, …

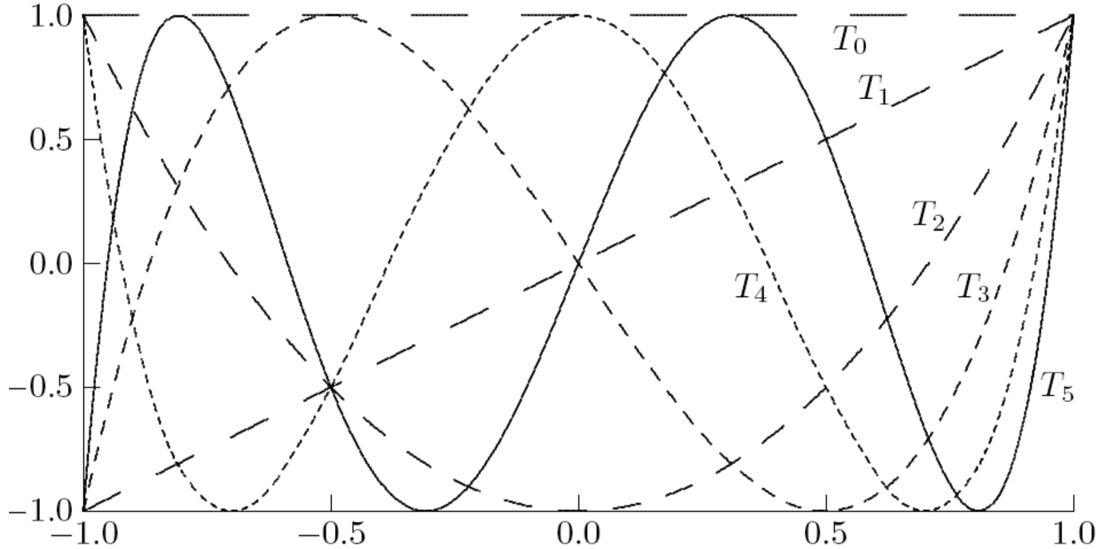
Chebyschev Polynomials

- Chebyschev basis is derived for weights of *$(1-x^2)^{-1/2}$* and the base set of monomials over the interval [-1.1]

$$\phi_n(x) = \cos(n \cdot \arccos(x))$$

$$\phi_1(x) = 1,\ \phi_2(x) = x,\ \phi_3(x) = 2x^2 - 1,\ \phi_4(x) = 4x^3 - 3x$$

$$\phi_{n+1}(x) = 2x\phi_n(x) - \phi_{n-1}(x)$$



Taylor Interpolation

- If a known function is to be interpolated, the Taylor series can be used to provide a polynomial interpolation

$$p_n(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2}(x-a)^2 + \cdots + \frac{f^{(n)}(a)}{n!}(x-a)^n$$

  - Can only be applied to a known function
  - Provides a good approximation in the neighborhood of a

Interpolation Error and Convergence

- To characterize an interpolation function we have to formalize interpolation error
  - Interpolation error is the difference between the original function and the interpolating function

$$f(x) - p(x)$$

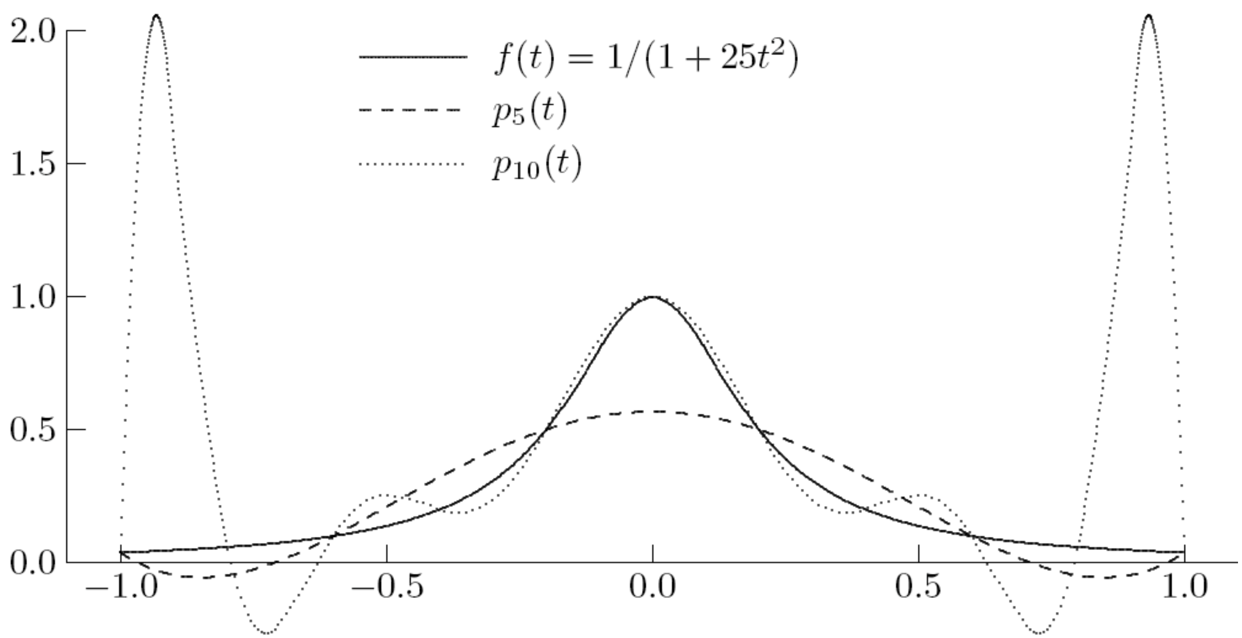
    - For interpolating polynomial of degree n-1 and the Taylor series

$$f(x) - p(x) = \frac{(x - x_1)(x - x_2)\cdots(x - x_n)}{n!} f^{(n)}(c)$$

- Convergence of interpolation implies that the error goes towards 0 as the number of data points is increased

Convergence

- Polynomial interpolation does not necessarily converge
  - Runge phenomenon for Monomial interpolation with uniformly spaced data points



Chebyshev Interpolation

- The choice of data points (here from within interval [-1,1]) influences the interpolation error
  - Data points can be chosen such as to minimize the maximum interpolation error for any point in an interval

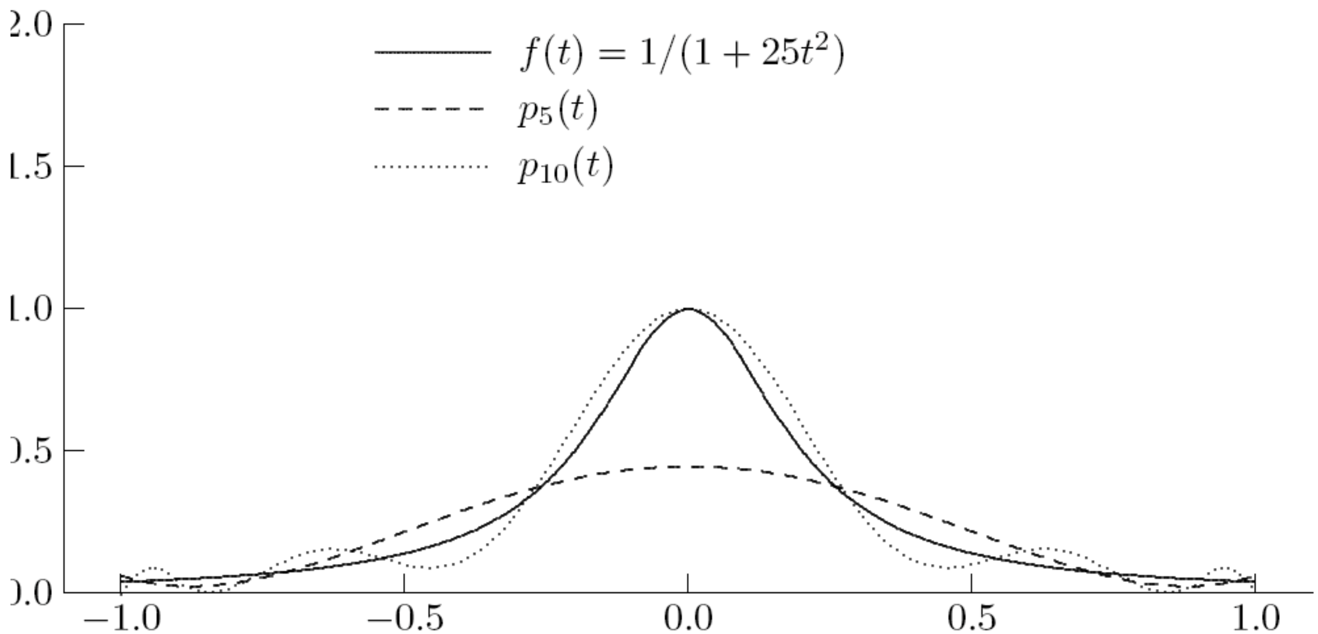
$$\arg\min_{(x_1 \cdots x_n)} \max_x \frac{(x-x_1)(x-x_2)\cdots(x-x_n)}{n!} f^{(n)}(c)$$

$$= \arg\min_{(x_1 \cdots x_n)} \max_x (x-x_1)(x-x_2)\cdots(x-x_n)$$

    - Leads to best convergence characteristics
  - Optimal choice for data points $x_i = \cos\frac{(2i-1)\pi}{2n}$
    - Error: $\frac{1}{2^{n-1}}$ and thus convergence

Chebyshev Points

- Chebyshev points ensure convergence for polynomial interpolation
  - Runge function with monomial basis for Chebyshev points



Piecewise Polynomial Interpolation

- Fitting a single polynomial to a large number of data points requires a high-order polynomial
  - Very complex polynomial that introduces many oscillations between data points
- Piecewise polynomials can be used to form an interpolant from individual polynomials stretching between two neighboring data points
  - x values of data points are called knots and mark points where interpolant moves from one polynomial to the next

Piecewise Polynomial Interpolation

- Two data points can be interpolated with a wide range of polynomials
  - Piecewise linear interpolation
  - Piecewise quadratic interpolation
  - Piecewise cubic interpolation
- Resolves excessive oscillation between data points but has transition points at knots
  - Potentially not smooth
  - Potentially large number of parameters that have to be set

Piecewise Linear Interpolation

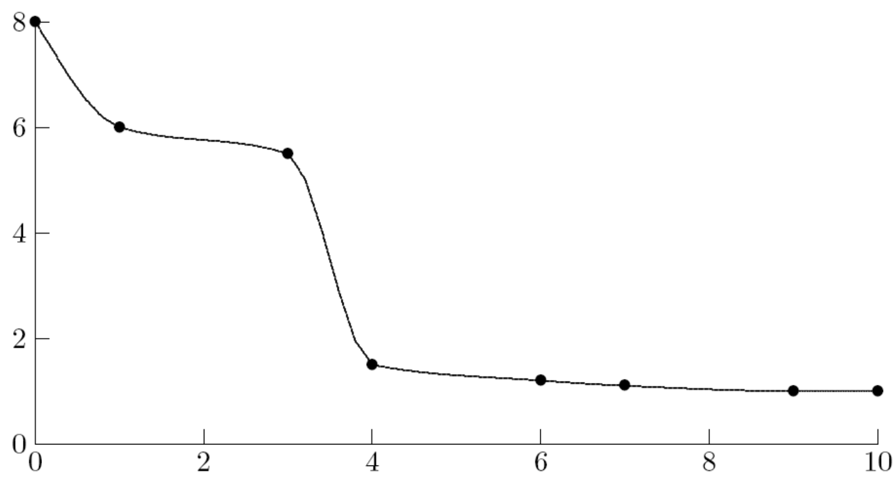
- Two consecutive data points are connected through lines
  - N data points are interpolated through n-1 lines.
    - Each line has 2 parameters (2(n-1) total parameters)
    - Each internal data point provides 2 equations while the boundary ones provide 1 each (2(n-2)+2=2(n-1) total equations)
  - Linear interpolation has a unique solution using only the data points
    - Interpolant is not smooth (not differentiable)

Piecewise Cubic Interpolation

- Two consecutive data points are connected through a third order polynomial
  - N data points are interpolated through n-1 third order polynomials.
    - Each polynomial has 4 parameters (4(n-1) total parameters)
    - Each internal data point provides 2 equations while the boundary ones provide 1 each (2(n-2)+2=2(n-1) equations)
  - Cubic interpolation has an extra 2(n-1) parameters that are not defined by the data points and can be used to impose additional characteristics
    - Differentiable at knots
    - Smoothness of function

Cubic Hermite Interpolation (Cspline)

- Hermite interpolation uses an additional constraint requiring continuous first derivative
  - Continuous first derivatives add n-2 equations
  - Hermite interpolation leaves n free parameters



Cubic Hermite Interpolation

- A particular Cubic Hermite interpolation can be constructed using a set of basis polynomials and desired slopes at the data points

$$p(x) = y_k(t^3 - 3t^2 + 1) + (x_{k+1} - x_k)m_k(t^3 - 2t^2 + t)$$
$$+ y_{k+1}(-2t^3 + 3t^2) + (x_{k+1} - x_k)m_{k+1}(t^3 - t^2)$$

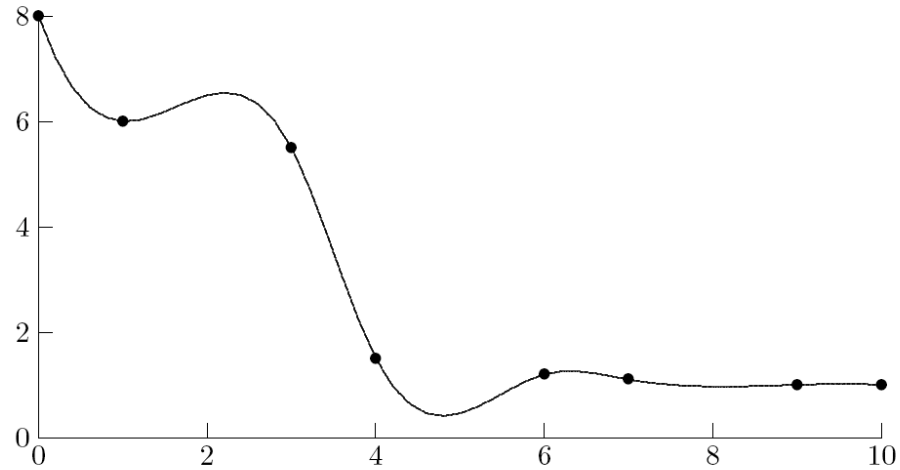
$$t = \frac{x - x_k}{x_{k+1} - x_k}$$

  - Multiple ways exist to pick the slopes
    - Finite Differences:

$$m_k = \frac{y_{k+1} - y_k}{2(x_{k+1} - x_k)} + \frac{y_k - y_{k-1}}{2(x_k - x_{k-1})}$$

Smooth Cubic Spline Interpolation

- Spline interpolation uses an additional constraint requiring that the polynomial of degree n is n-1 times continuously differentiable
  - For Cubic Splines this adds n-2 equations for the first and n-2 equations for the second derivative leaving 2 free parameters



Cubic Spline Interpolation

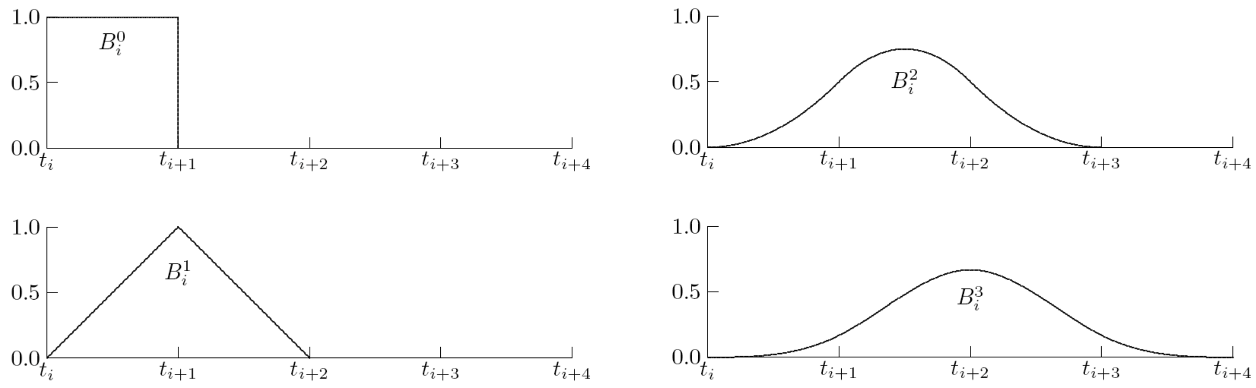
- The final 2 parameters can be determined to ensure additional properties
  - Set derivative at first and last knot
  - Force second derivative to be 0 at the end points
    - Natural spline
  - Force two consecutive splines to be the same (effectively removing one knot)
  - Set derivatives and second derivatives to be the same at end points
    - Useful for periodic functions

B-Splines

- B-Splines form a basis for a family of spline functions with useful properties
  - Spline functions can be defined recursively

$$\phi_{0,i}(x) = \begin{cases} 1 & x_i \le x \le x_{i+1} \\ 0 & otherwise \end{cases}$$

$$\phi_{k,i}(x) = \frac{x - x_i}{x_{i+k} - x_i}\phi_{k-1,i}(x) + \left(1 - \frac{x - x_{i+1}}{x_{(i+1)+k} - x_{i+1}}\right)\phi_{k-1,i+1}(x)$$

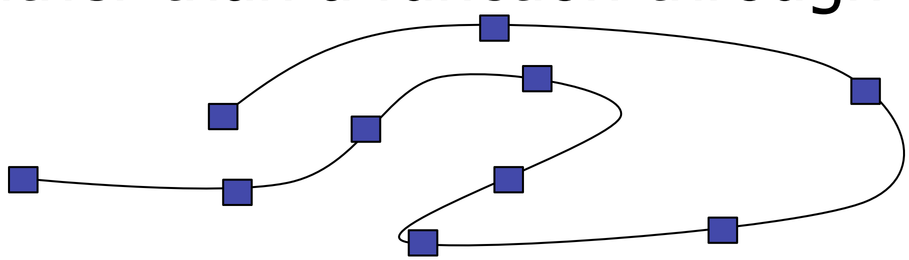


B-Splines

- B-Splines provide a number of properties that are useful for piecewise interpolation
  - Set of basis functions located at different data points allows for an efficient formulation of the complete interpolation function
  - Linear systems matrix for solving coefficients is banded and nonsingular
    - Can be solved efficiently
  - Operations on interpolant can be performed efficiently

Splines for Computer Graphics and Multiple Dimensions

- In Computer Graphics it is often desired to fit a curve rather than a function through data points



  - A curve through data points d_i in n dimensions can be represented as a function through the same points in $n+1$ dimensions
  - Interpolation is represented as n interpolation functions (one for each dimension) over a free parameter t that usually represents the distance of the data points

$$t_1 = 0 \quad , \quad t_{i+1} = t_i + \left\| d_{i+1} - d_i \right\|$$

Splines in Multiple Dimensions

- All interpolation methods covered can be used to interpolate data points in multiple dimensions
  - One interpolation per dimension, picking one dimension or an auxiliary dimension as the common basis
    - Interpolation in multiple dimensions results in a system of equations with one function per dimension (if an auxiliary parameter is used for the interpolation)
  - k^{th} order Bézier curves are a frequently used spline technique where *k+1* points are used to define two points to interpolate through and two directions for the curve through these points
    - Used to describe scalable fonts
      - Type 1 and 3 fonts: Cubic Bézier curves
      - True type fonts: Quadratic Bézier curves

Trigonometric Interpolation

- Fourier Interpolation represents a way to interpolate periodic data using sine and cosine functions as a basis.

$$\phi_i(x) = \alpha_i \sin((i-1)x) + \beta_i \cos((i-1)x)$$

  - Data points have to be scaled in *x* to be between *–π* and *π* and to not fall on the boundaries (e.g. through

    $t=\pi\,(-1 + 2(x-x_{min}+1/(2n))/(x_{max}-x_{min}+1/n))$

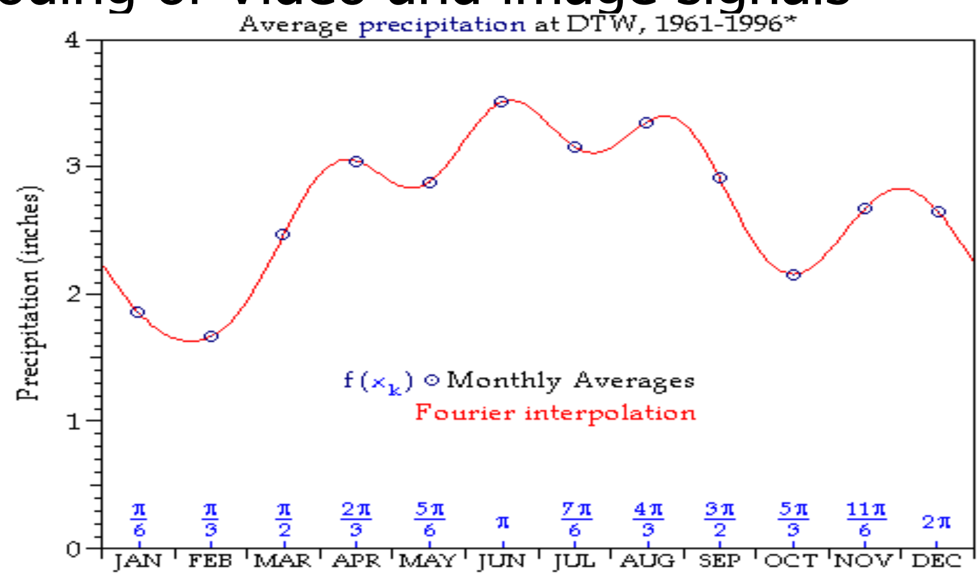
  - Interpolation of 2N (or 2N+1) data points requires the first N+1 basis functions

$$f(x) = \sum_{i=1}^{N+1} \alpha_i \sin((i-1)x) + \beta_i \cos((i-1)x)$$

    - Coefficients can be solved for evenly spaced points efficiently in *O (N log N)* using FFT

Fourier Interpolation

- Fourier Interpolation is very effective and efficient for periodic data since the function repeats identically outside the defined region.
  - Encoding of audio signals
  - Encoding of Video and image signals



Interpolation

- Interpolation can be used to derive a system of equations from a set of data points
  - Interpolation requires data points to be matched precisely
    - Complexity of interpolant has to be high enough to allow interpolation
  - Interpolation is appropriate only if there is no substantial noise in the data points
    - Interpolation not only models data but also noise in the data
- Interpolation provides an efficient way to derive approximations to unknown systems equations from a set of data points
  - It should still be known what is being modeled to pick the appropriate function form