# H6: interpolation

## 6.1 introduction

**concept interpolation**

= fitting a function to data, where the function must match the given data exactly

> purposes:

- Plotting a smooth curve through discrete data points
- Reading between the lines of a table
- Differentiating or integrating tabular data
- Evaluating a mathematical function quickly and easily
- Replacing a complicated function by a simple one

**interpolation**

for a given set of datapoints $(t_i, y_i)$ , i=1,…,m

> an interpolant is chosen form the space of functions spanned by a suitable set:

**basis functions** $\phi_1(t), \ldots, \phi_n(t).$

the interpolating function **f** is therefore expressed as a lin. comb of these basis functions:

$$f(t) = \sum_{j=1}^{n} x_j \phi_j(t)$$

where the parameters $x_j$ are to be determined

> require that f interpolate the data $(t_i, y_i)$:

$$f(t_i) = \sum_{j=1}^{n} x_j \phi_j(t_i) = y_i$$

thus a system of linear equations in the form: **Ax = y**

> entries of **A** are given by $a_{ij} = \phi_j(t_i)$

**x** are $x_j$, which we want to determine

**y** are $y_i$, the data points

## 6.2 polynomial interpolation of discrete data

**set of polynomials $\mathbb{P}_k(k \geq 0)$**

= set of polynomials with a degree lower or equal than k

> $\mathbb{P}_k$ is a vector space with dimension k+1

### 6.2.1 monomial basis

**basis of monomial**

To interpolate n data points, choose k=n-1

> basis for $\mathbb{P}_{n-1}$ can be chosen to be the first n monomials:

$$\phi_j(t) = t^{j-1}$$

for which a given polynomial $p_{n-1} \in \mathbb{P}_{n-1}$ has the form

$$p_{n-1}(t) = x_1 + x_2 t + \cdots + x_n t^{n-1}$$

The system of equations we want to solve is

$$\mathbf{A}\mathbf{x} = \begin{bmatrix} 1 & t_1 & t_1^2 & \cdots & t_1^{n-1} \\ 1 & t_2 & t_2^2 & \cdots & t_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & t_n & t_n^2 & \cdots & t_n^{n-1} \end{bmatrix} \begin{bmatrix} x1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \mathbf{y}$$

> **A** is a Vandermonde matrix

**Horner's method**

When represented in the monomial basis, a polynomial:

$$p_{n-1}(t) = x_1 + x_2 t + x_3 t^2 + \cdots + x_n t^{n-1}$$

can be evaluated as:

$$p_{n-1}(t) = x_1 + t(x_2 + t(x_3 + t(\cdots(x_{n-1} + x_n t)\cdots)))$$

which requires only $n$ summations and $n$ additions.

| **6.2.2 Lagrange interpolation** | |
|---|---|
| Lagrange basis functions | for a given set of datapoints ($t_i$,$y_i$) , i=1,…,m these are for $\mathbb{P}_{n-1}$ given by:<br>$$l_j(t) = \frac{\prod_{k=1,k\neq j}^{n}(t-t_k)}{\prod_{k=1,k\neq j}^{n}(t_j-t_k)}$$<br>It can be seen that<br>• $l_j(t)$ is a polynomial of degree $n-1$<br>• $l_j(t_i) = \begin{cases} 1, & \text{if } i=j. \\ 0, & \text{if } i\neq j. \end{cases}$<br>thus, for this basis the matrix of the linear system **Ax=y** is the identity matrix **I**<br>> interpolating polynomial is then:<br>$$p_{n-1}(t) = y_1 l_1(t) + y_2 l_2(t) + \cdots + y_n l_n(t)$$<br>which is easy to construct. |
| **6.2.3 Newton interpolation** | |
| Newton basis functions | for a given set of datapoints ($t_i$,$y_i$) , i=1,…,m these are for $\mathbb{P}_{n-1}$ given by:<br>$$\pi_j(t) = \prod_{k=1}^{j-1}(t-t_k)$$<br>Note that we assign $\pi_j(t) = 1$ for $j = 1$<br>The interpolating polynomial then has the form<br>$$p_{n-1}(t) = x_1 + x_2(t-t_1) + x_3(t-t_1)(t-t_2) + \cdots x_n(t-t_1)\ldots(t-t_{n-1})$$<br>This the basis matrix **A** is lower triangular<br><br>Once we know the coefficients $\mathbf{x_i}$, the resulting polynomial can be evaluated using Horner<br>$$p_{n-1}(t) = x_1 + (t-t_1)\left[x_2 + (t-t_2)\left[x_3 + (t-t_3)\left[\cdots(x_{n-1} + x_n(t-t_{n-1}))\cdots\right]\right]\right]$$ |
| useful property of Newton interpolation | If $\mathbf{p_j}$(t) is a polynomial of degree j-1 which interpolates j data points<br>> for any $x_{j+1}$:<br>$$p_{j+1}(t) = p_j(t) + x_{j+1}\pi_{j+1}(t) \qquad (25$$<br>is a polynomial of degree $j$ that also interpolates the same $j$ points. The free parameter $x_{j+1}$ can be chosen so that $p_{j+1}(t)$ interpolates the new data points $y_{j+1}$ as<br>$$x_{j+1} = \frac{y_{j+1} - p_j(t_{j+1})}{\pi_{j+1}(t_{j+1})} \qquad (26$$<br>>> interpolant can be constructed incrementally |
| **6.2.4 polynomial interpolation of a continuous function** | |
| interpolants with high degree | - expensive to determine/evaluate<br>- polynomial of degree n has n-1 extrema<br>> many wiggles<br>> fluctuates wildly between data points<br>> doesn't approximate underlying function at all |
| > solution | interpolants of high degree: - converge to the function in the middle of the interval<br>- diverge away from the function at endpoints<br>>> bunch up the datapoints near the ends of the interval |
| ex: Chebysev points | function that bunches up endpoints on the interval [-1,1]:<br>$$t_i = \cos\left(\frac{(2i-1)\pi}{2k}\right), \qquad i = 1,\ldots,k$$ |

| 6.3 piecewise polynomial interpolation | |
|---|---|
| concept piecewise interpolation | divide the interval into subintervals<br>> interpolate a different polynomial to each subinterval<br>> points at which the interpolant changes are called *knots* |
| piecewise interpolation | For a set of ordered data points ($x_i$,$y_i$) on the interval [a,b]<br>> a function S is a spline interpolation of degree k if:<br>• $S$ is defined on the interval $[a, b]$<br>• the $r$th derivative of $S$ is continuous in the interval $[a, b]$ for $0 \leq r \leq k-1$<br>• $S$ is a polynomial of degree $\leq k$ in every subinterval between the knots<br><br>The simplest case is $k = 1$ for which $S_j$ is a straight line and the spline is a piecewise linear interpolation |
| **6.3.1 cubic spline interpolation** | |
| cubic spline interpolation | a spline with n knots has (n-1) piecewise polynomials of degree k<br>> there are (k+1)(n-1) free parameters<br><br>ex: linear polynomial, k=1 and 2 free parameters per polynomial<br>> has 2(n-1) free parameters and cubic spline has 4(n-1)<br>> • Interpolating the data requires $2(n-1)$ equations, because each of the $(n-1)$ polynomials must match the two data points at either end of the subinterval.<br>• Requiring the derivative to be continuous gives $(n-2)$ additional equations, because there are $(n-2)$ *interior* data points.<br>• Requiring that the second derivative is also continuous gives $(n-2)$ additional equations.<br>The spline of lowest degree that has sufficient variables to satisfy all these equations is a cubic spline: 4n-4 variables for 4n-6 conditions<br>> remaining two variables can be fixed in an number of ways:<br>• **clamped cubic spline**: Specifying the first derivative of the endpoints<br>• **natural spline**: forcing the second derivative to be zero at the endpoints<br>• **not-a-knot**: the third derivative of the spline is continuous at the one-but-outermost data points<br>• **periodic spline**: forcing equality of the first as well as second derivatives of the two outermost points (if the spline is to be periodic) |
| **6.4 software** | |
| *scipy.interpolate* | has the functions:<br>• `lagrange` for Lagrange interpolation<br>• `CubicSpline` for cubic spline interpolation (with many options)<br>• `interp1d` to interpolate 1 dimensional data (also capable of finding cubic splines, but with less options)<br>• `griddata` to interpolate multidimensional data |
| *interpld* | can do different kinds of cubicspline |