July 20, 2013

Chapter 10: Polynomial Interpolation

Uri M. Ascher and Chen Greif
Department of Computer Science
The University of British Columbia
{ascher,greif}@cs.ubc.ca

Slides for the book
A First Course in Numerical Methods (published by SIAM, 2011)
`http://www.ec-securehost.com/SIAM/CS07.html`

Goals of this chapter

- To motivate the need for interpolation of data and of functions;
- to derive three(!) different methods for computing a polynomial interpolant, each particularly suitable for certain circumstances;
- to derive error expressions for the polynomial interpolation process;
- to construct Chebyshev interpolants, which can provide very accurate approximations for complex functions using stable high degree polynomial interpolation at special points; and
- to consider cases where not only function values but also their derivative values are to be interpolated.

Outline

- Monomial basis
- Lagrange basis
- Newton basis and divided differences
- Interpolation error
- Chebyshev interpolation
- Interpolating also derivative values

Interpolating data

We are given a collection of **data samples** $\{(x_i, y_i)\}_{i=0}^{n}$.

- The $\{x_i\}_{i=0}^{n}$ are called the **abscissae** (singular: **abscissa**), the $\{y_i\}_{i=0}^{n}$ are called the **data values**.
- Want to find a function $v(x)$ which can be used to estimate sampled function for $x \neq x_i$. **Interpolation**: $v(x_i) = y_i, \quad i = 0, 1, \ldots, n$.
- Why?
  - We often get discrete data from sensors or computation, but want information as if the function were not discretely sampled.
  - May need to plot, differentiate or integrate data trend.
  - May require an economical approximation for the data.

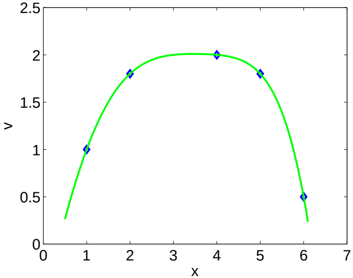
Interpolating data

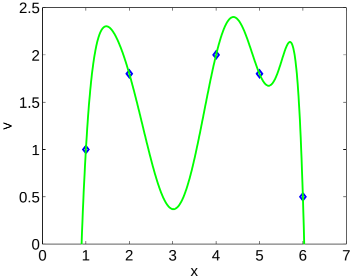
We are given a collection of **data samples** $\{(x_i, y_i)\}_{i=0}^n$.

- The $\{x_i\}_{i=0}^n$ are called the **abscissae** (singular: **abscissa**), the $\{y_i\}_{i=0}^n$ are called the **data values**.
- Want to find a function $v(x)$ which can be used to estimate sampled function for $x \neq x_i$. **Interpolation**: $v(x_i) = y_i, \quad i = 0, 1, \ldots, n$.
- Why?
  - We often get discrete data from sensors or computation, but want information as if the function were not discretely sampled.
  - May need to plot, differentiate or integrate data trend.
  - May require an economical approximation for the data.

Interpolating data wish list

- Want a reasonable looking interpolant. Example:





  The interplant in the left figure above looks, somehow, more reasonable than the right one.

- If possible, v should be inexpensive to construct.
- If possible, $v(x)$ should be inexpensive to evaluate for a given x.

Interpolating functions

A function $f(x)$ may be given on an interval, $a \leq x \leq b$, explicitly or implicitly. Want interpolant $v(x)$ such that

$$v(x_i) = f(x_i), \quad i = 0, 1, \ldots, n,$$

at points $x_i \in [a, b]$.

Same algorithms as for interpolating data apply, but (importantly) here we may be able to

- choose the abscissae x_i;
- estimate interpolation error.

Interpolating functions

A function $f(x)$ may be given on an interval, $a \le x \le b$, explicitly or implicitly. Want interpolant $v(x)$ such that

$$v(x_i) = f(x_i), \quad i = 0, 1, \ldots, n,$$

at points $x_i \in [a, b]$.

Same algorithms as for interpolating data apply, but (importantly) here we may be able to

- choose the abscissae x_i;
- estimate interpolation error.

Interpolation formulation

There are lots of ways to define a function $v(x)$ to interpolate the data: Polynomials, trigonometric, exponential, rational (fractions), wavelets/curvelets/ridgelets, radial basis functions, ...

Consider a **linear** combination of linearly independent **basis functions** $\{\phi_j(x)\}$

$$v(x) = c_0\phi_0(x) + c_1\phi_1(x) + \cdots + c_n\phi_n(x) = \sum_{j=0}^{n} c_j\phi_j(x)$$

where c_j are the **interpolation coefficients** or **interpolation weights**.
Then the interpolation conditions yield

$$\begin{pmatrix} \phi_0(x_0) & \phi_1(x_0) & \phi_2(x_0) & \cdots & \phi_n(x_0) \\ \phi_0(x_1) & \phi_1(x_1) & \phi_2(x_1) & \cdots & \phi_n(x_1) \\ \vdots & \vdots & \vdots & & \vdots \\ \phi_0(x_n) & \phi_1(x_n) & \phi_2(x_n) & \cdots & \phi_n(x_n) \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}.$$

Interpolation formulation

There are lots of ways to define a function $v(x)$ to interpolate the data:
Polynomials, trigonometric, exponential, rational (fractions), wavelets/curvelets/ridgelets, radial basis functions, ...
Consider a **linear** combination of linearly independent **basis functions** $\{\phi_j(x)\}$

$$v(x) = c_0\phi_0(x) + c_1\phi_1(x) + \cdots + c_n\phi_n(x) = \sum_{j=0}^{n} c_j\phi_j(x)$$

where c_j are the **interpolation coefficients** or **interpolation weights**.
Then the interpolation conditions yield

$$\begin{pmatrix} \phi_0(x_0) & \phi_1(x_0) & \phi_2(x_0) & \cdots & \phi_n(x_0) \\ \phi_0(x_1) & \phi_1(x_1) & \phi_2(x_1) & \cdots & \phi_n(x_1) \\ \vdots & \vdots & \vdots & & \vdots \\ \phi_0(x_n) & \phi_1(x_n) & \phi_2(x_n) & \cdots & \phi_n(x_n) \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}.$$

Interpolation formulation

There are lots of ways to define a function $v(x)$ to interpolate the data:
Polynomials, trigonometric, exponential, rational (fractions), wavelets/curvelets/ridgelets, radial basis functions, ...
Consider a **linear** combination of linearly independent **basis functions** $\{\phi_j(x)\}$

$$v(x) = c_0\phi_0(x) + c_1\phi_1(x) + \cdots + c_n\phi_n(x) = \sum_{j=0}^{n} c_j\phi_j(x)$$

where c_j are the **interpolation coefficients** or **interpolation weights**.
Then the interpolation conditions yield

$$\begin{pmatrix} \phi_0(x_0) & \phi_1(x_0) & \phi_2(x_0) & \cdots & \phi_n(x_0) \\ \phi_0(x_1) & \phi_1(x_1) & \phi_2(x_1) & \cdots & \phi_n(x_1) \\ \vdots & \vdots & \vdots & & \vdots \\ \phi_0(x_n) & \phi_1(x_n) & \phi_2(x_n) & \cdots & \phi_n(x_n) \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}.$$

Polynomial interpolation

- Special case: the functions $\phi_0(x), \phi_1(x), \ldots, \phi_n(x)$ form a basis for all polynomials of degree at most n.
- This is the simplest, most basic form of interpolation.
- Used as building block for other methods for interpolation, integration, solution of differential equations, etc.
- Our goal here is therefore to develop methods for polynomial interpolation, to be repeatedly used in later chapters (e.g., 11, 14, 15, 16).

Polynomial interpolation

- Special case: the functions $\phi_0(x), \phi_1(x), \ldots, \phi_n(x)$ form a basis for all polynomials of degree at most n.
- This is the simplest, most basic form of interpolation.
- Used as building block for other methods for interpolation, integration, solution of differential equations, etc.
- Our goal here is therefore to develop methods for polynomial interpolation, to be repeatedly used in later chapters (e.g., 11, 14, 15, 16).

Outline

- Monomial basis
- Lagrange basis
- Newton basis and divided differences
- Interpolation error
- Chebyshev interpolation
- Interpolating also derivative values

Monomial basis

$$v(x) = p(x) = p_n(x) = \sum_{j=0}^{n} c_j \phi_j(x).$$

Choose

$$\phi_j(x) = x^j.$$

Then

$$v(x) = p(x) = p_n(x) = \sum_{j=0}^{n} c_j x^j.$$

Example

$$\{(x_i, y_i)\} = \{(2,14), (6,24), (4,25), (7,15)\}$$

In particular, $n = 3$.

- Requires four basis functions: $\{\phi_j(x)\} = \{1, x, x^2, x^3\}$.
  The interpolant will be $p(x) = c_0 + c_1 x + c_2 x^2 + c_3 x^3$.
- Construct linear system

$$A = \begin{pmatrix} 1 & 2 & 4 & 8 \\ 1 & 6 & 36 & 216 \\ 1 & 4 & 16 & 64 \\ 1 & 7 & 49 & 343 \end{pmatrix} \quad \mathbf{y} = \begin{pmatrix} 14 \\ 24 \\ 25 \\ 15 \end{pmatrix}$$

- Solve c = `A\y`.
  We find $\mathbf{c} \approx (-0.267, 1.700, 2.767, 3.800)^T$.

Monomial basis assessment

- Simple!
- Matrix A is a *Vandermonde*: nonsingular. Hence uniqueness: there is precisely one interpolating polynomial.
- *Construction* cost is $\mathcal{O}(n^3)$ flops (high if n is large).
- *Evaluation* cost (per point x) using Horner's rule is $\mathcal{O}(n)$ flops (low).
- Coefficients c_j are not indicative of $f(x)$, and all change if one data value is modified.
- Potential stability difficulties if degree is large or abscissae spread apart.

Outline

- Monomial basis
- Lagrange basis
- Newton basis and divided differences
- Interpolation error
- Chebyshev interpolation
- Interpolating also derivative values

Lagrange form

In several ways, the opposite of monomials! Choose coefficients $c_j = y_j$.
For this define **Lagrange polynomials** $\phi_j(x) = L_j(x)$

$$\phi_j(x) = \frac{(x - x_0)\cdots(x - x_{j-1})(x - x_{j+1})\cdots(x - x_n)}{(x_j - x_0)\cdots(x_j - x_{j-1})(x_j - x_{j+1})\cdots(x_j - x_n)} = \prod_{\substack{i=0 \\ i\neq j}}^{n} \frac{(x - x_i)}{(x_j - x_i)}.$$

Then

$$\phi_j(x_i) = \begin{cases} 0 & i \neq j \\ 1 & i = j, \end{cases}$$

so

$$p(x) = \sum_{j=0}^{n} y_j \phi_j(x).$$

Example redux

$$\{(x_i, y_i)\} = \{(2,14),(6,24),(4,25),(7,15)\}$$

Use the Lagrange basis.

- Four basis functions

$$\phi_0(x) = \frac{(x-6)(x-4)(x-7)}{(2-6)(2-4)(2-7)}, \quad \phi_1(x) = \frac{(x-2)(x-4)(x-7)}{(6-2)(6-4)(6-7)},$$
$$\phi_2(x) = \frac{(x-2)(x-6)(x-7)}{(4-2)(4-6)(4-7)}, \quad \phi_3(x) = \frac{(x-2)(x-6)(x-4)}{(7-2)(7-6)(7-4)}.$$

- Interpolant will be

$$\begin{aligned} p(x) =& 14\frac{(x-6)(x-4)(x-7)}{(-4)(-2)(-5)} + 24\frac{(x-2)(x-4)(x-7)}{(+4)(+2)(-1)} \\ &+ 25\frac{(x-2)(x-6)(x-7)}{(+2)(-2)(-3)} + 15\frac{(x-2)(x-6)(x-4)}{(+5)(+1)(+3)}. \end{aligned}$$

Ordering the operations

- *Construction*: **barycentric weights**.

$$w_0 = \frac{1}{(2-6)(2-4)(2-7)}, \; w_1 = \frac{1}{(6-2)(6-4)(6-7)},$$
$$w_2 = \frac{1}{(4-2)(4-6)(4-7)}, \; w_3 = \frac{1}{(7-2)(7-6)(7-4)}.$$

- *Evaluation*: at a point x define

$$\begin{aligned} \psi(x) &= (x-2)(x-6)(x-4)(x-7), \\ p(x) &= \psi(x)\left[\frac{14w_0}{x-2} + \frac{24w_1}{x-6} + \frac{25w_2}{x-4} + \frac{15w_3}{x-7}\right]. \end{aligned}$$

- In general

$$\psi(x) = \prod_{i=0}^{n}(x - x_i), \quad p(x) = \psi(x)\sum_{j=0}^{n}\frac{y_j w_j}{x - x_j}.$$

Ordering the operations

- *Construction*: **barycentric weights.**

$$w_0 = \frac{1}{(2-6)(2-4)(2-7)}, \; w_1 = \frac{1}{(6-2)(6-4)(6-7)},$$
$$w_2 = \frac{1}{(4-2)(4-6)(4-7)}, \; w_3 = \frac{1}{(7-2)(7-6)(7-4)}.$$

- *Evaluation*: at a point x define

$$\begin{aligned} \psi(x) &= (x-2)(x-6)(x-4)(x-7), \\ p(x) &= \psi(x)\left[\frac{14w_0}{x-2} + \frac{24w_1}{x-6} + \frac{25w_2}{x-4} + \frac{15w_3}{x-7}\right]. \end{aligned}$$

- In general

$$\psi(x) = \prod_{i=0}^{n}(x-x_i), \quad p(x) = \psi(x)\sum_{j=0}^{n}\frac{y_j w_j}{x-x_j}.$$

Lagrange basis assessment

- Not as simple as monomial basis.
- Matrix A is the *identity*: the coefficients are immediately obtained.
- *Construction* cost is $\mathcal{O}(n^2)$ flops (OK even if n is large).
- *Evaluation* cost for each x is $\mathcal{O}(n)$ flops (low but not lowest).
- Coefficients c_j indicative of data and useful for function manipulation such as integration and differentiation!
- Stable even if degree is large or abscissae spread apart!

Outline

- Monomial basis
- Lagrange basis
- Newton basis and divided differences
- Interpolation error
- Chebyshev interpolation
- Interpolating also derivative values

Newton form

Can we add a new data point without changing the entire interpolant?

- Need $n \to n+1$, easy to construct and evaluate.
- To this end require
  - New basis function cannot disturb prior interpolation: $\phi_j(x_i) = 0$ for $i < j$.
  - Old basis function does not need information about new data values: $\phi_j(x)$ is independent of (x_i, y_i) for $i > j$.
- Newton basis functions

$$\phi_j(x) = \prod_{i=0}^{j-1}(x - x_i), \quad j = 0, 1, \ldots, n.$$

- Leads to lower triangular matrix A .

Newton form

Can we add a new data point without changing the entire interpolant?

- Need $n \to n+1$, easy to construct and evaluate.
- To this end require
  - New basis function cannot disturb prior interpolation: $\phi_j(x_i) = 0$ for $i < j$.
  - Old basis function does not need information about new data values: $\phi_j(x)$ is independent of (x_i, y_i) for $i > j$.
- Newton basis functions

$$\phi_j(x) = \prod_{i=0}^{j-1} (x - x_i), \quad j = 0, 1, \ldots, n.$$

- Leads to lower triangular matrix A .

Newton form

Can we add a new data point without changing the entire interpolant?

- Need $n \to n+1$, easy to construct and evaluate.
- To this end require
  - New basis function cannot disturb prior interpolation: $\phi_j(x_i) = 0$ for $i < j$.
  - Old basis function does not need information about new data values: $\phi_j(x)$ is independent of (x_i, y_i) for $i > j$.
- Newton basis functions

$$\phi_j(x) = \prod_{i=0}^{j-1}(x - x_i), \quad j = 0, 1, \ldots, n.$$

- Leads to lower triangular matrix A .

Example redux

$$\{(x_i, y_i)\} = \{(2, 14), (6, 24), (4, 25), (7, 15)\}$$

Use the Newton basis.

- Four basis functions

$$\phi_0(x) = 1, \quad \phi_1(x) = (x - 2),$$
$$\phi_2(x) = (x - 2)(x - 6), \quad \phi_3(x) = (x - 2)(x - 6)(x - 4).$$

- Construct linear system

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 4 & 0 & 0 \\ 1 & 2 & -4 & 0 \\ 1 & 5 & 5 & 15 \end{pmatrix}$$

and solve $A\mathbf{c} = \mathbf{y}$ to find $\mathbf{c} \approx (14, 2.5, -1.5, -0.2667)^T$.

- Note it is the same polynomial as before, just another form!

Example redux

$$\{(x_i, y_i)\} = \{(2,14), (6,24), (4,25), (7,15)\}$$

Use the Newton basis.

- Four basis functions

$$\phi_0(x) = 1, \quad \phi_1(x) = (x-2),$$
$$\phi_2(x) = (x-2)(x-6), \quad \phi_3(x) = (x-2)(x-6)(x-4).$$

- Construct linear system

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 4 & 0 & 0 \\ 1 & 2 & -4 & 0 \\ 1 & 5 & 5 & 15 \end{pmatrix}$$

  and solve $A\mathbf{c} = \mathbf{y}$ to find $\mathbf{c} \approx (14, 2.5, -1.5, -0.2667)^T$.

- Note it is the same polynomial as before, just another form!

Example redux

$$\{(x_i, y_i)\} = \{(2, 14), (6, 24), (4, 25), (7, 15)\}$$

Use the Newton basis.

- Four basis functions

$$\phi_0(x) = 1, \quad \phi_1(x) = (x - 2),$$
$$\phi_2(x) = (x - 2)(x - 6), \quad \phi_3(x) = (x - 2)(x - 6)(x - 4).$$

- Construct linear system

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 4 & 0 & 0 \\ 1 & 2 & -4 & 0 \\ 1 & 5 & 5 & 15 \end{pmatrix}$$

  and solve $A\mathbf{c} = \mathbf{y}$ to find $\mathbf{c} \approx (14, 2.5, -1.5, -0.2667)^T$.

- Note it is the same polynomial as before, just another form!

Divided differences

- An alternative method of determining the coefficients for a Newton basis interpolating polynomial.
- Used more often than solving a linear system.
- Makes it easier to add and delete data points.
- Divided differences have an interesting connection with function derivatives, and provide a tool with which we will analyze interpolation error .
- Divided differences are defined recursively

$$f[x_i] = y_i, \qquad f[x_i, \ldots, x_j] = \frac{f[x_{i+1}, \ldots, x_j] - f[x_i, \ldots, x_{j-1}]}{x_j - x_i}.$$

- The coefficients for Newton interpolation are just $c_j = f[x_0, \ldots, x_j]$ (the diagonal elements in the table).
- To add another data point ($n \to n+1$), just add another row to the table (assuming that the abscissae are distinct) .

Divided differences

- An alternative method of determining the coefficients for a Newton basis interpolating polynomial.
- Used more often than solving a linear system.
- Makes it easier to add and delete data points.
- Divided differences have an interesting connection with function derivatives, and provide a tool with which we will analyze interpolation error .
- Divided differences are defined recursively

$$f[x_i] = y_i, \qquad f[x_i, \ldots, x_j] = \frac{f[x_{i+1}, \ldots, x_j] - f[x_i, \ldots, x_{j-1}]}{x_j - x_i}.$$

- The coefficients for Newton interpolation are just $c_j = f[x_0, \ldots, x_j]$ (the diagonal elements in the table).
- To add another data point ($n \to n+1$), just add another row to the table (assuming that the abscissae are distinct) .

Divided differences

- An alternative method of determining the coefficients for a Newton basis interpolating polynomial.
- Used more often than solving a linear system.
- Makes it easier to add and delete data points.
- Divided differences have an interesting connection with function derivatives, and provide a tool with which we will analyze interpolation error .
- Divided differences are defined recursively

$$f[x_i] = y_i, \qquad f[x_i, \ldots, x_j] = \frac{f[x_{i+1}, \ldots, x_j] - f[x_i, \ldots, x_{j-1}]}{x_j - x_i}.$$

- The coefficients for Newton interpolation are just $c_j = f[x_0, \ldots, x_j]$ (the diagonal elements in the table).
- To add another data point ($n \to n+1$), just add another row to the table (assuming that the abscissae are distinct) .

Divided difference table

i	x_i	$f[x_i]$	$f[x_{i-1}, x_i]$	$f[x_{i-2}, x_{i-1}, x_i]$	$\cdots$	$f[x_{i-n}, \ldots, x_i]$
0	x_0	$f(x_0)$				
1	x_1	$f(x_1)$	$\frac{f[x_1]-f[x_0]}{x_1-x_0}$			
2	x_2	$f(x_2)$	$\frac{f[x_2]-f[x_1]}{x_2-x_1}$	$f[x_0, x_1, x_2]$		
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	
n	x_n	$f(x_n)$	$\frac{f[x_n]-f[x_{n-1}]}{x_n-x_{n-1}}$	$f[x_{n-2}, x_{n-1}, x_n]$	$\cdots$	$f[x_0, x_1, \ldots, x_n]$

The diagonal entries yield the coefficients $c_j = f[x_0, \ldots, x_j], \ j = 0, 1, \ldots, n$.

Basis comparison

Basis name	$\phi_j(x)$	construction cost	evaluation cost	selling feature
Monomial	x^j	$\frac{2}{3}n^3$	$2n$	simple
Lagrange	$L_j(x)$	n^2	$5n$	$c_j = y_j$ most stable
Newton	$\prod_{i=0}^{j-1}(x - x_i)$	$\frac{3}{2}n^2$	$2n$	adaptive

Outline

- Monomial basis
- Lagrange basis
- Newton basis and divided differences
- Interpolation error
- Chebyshev interpolation
- Interpolating also derivative values

Error expression

- Assume that f is the function to be interpolated and $y_i = f(x_i),\ i = 0, 1, 2, \ldots, n$. Denote interpolant by $p_n(x)$. For any evaluation point x, want to estimate error

$$e_n(x) = f(x) - p_n(x)$$

and see how it depends on the choice of n and the properties of f.

- Fixing $x \notin \{x_i\}_{i=0}^n$, pretend we are adding as new data point $(x, f(x))$.
- Using the properties of the Newton basis and divided differences,

$$f(x) = p_{n+1}(x) = p_n(x) + f[x_0, \ldots, x_n, x] \prod_{j=0}^{n} (x - x_j)$$

or, by rearranging,

$$e_n(x) = f(x) - p_n(x) = f[x_0, \ldots, x_n, x]\psi(x).$$

Error expression

- Assume that f is the function to be interpolated and $y_i = f(x_i),\ i = 0, 1, 2, \ldots, n$. Denote interpolant by $p_n(x)$. For any evaluation point x, want to estimate error

$$e_n(x) = f(x) - p_n(x)$$

and see how it depends on the choice of n and the properties of f.

- Fixing $x \notin \{x_i\}_{i=0}^{n}$, pretend we are adding as new data point $(x, f(x))$.
- Using the properties of the Newton basis and divided differences,

$$f(x) = p_{n+1}(x) = p_n(x) + f[x_0, \ldots, x_n, x] \prod_{j=0}^{n} (x - x_j)$$

or, by rearranging,

$$e_n(x) = f(x) - p_n(x) = f[x_0, \ldots, x_n, x]\psi(x).$$

Error estimate and bound

- Let $a = \min_i x_i,\ b = \max_i x_i$ and assume $x \in [a, b]$ (otherwise $p_n(x)$ is "extrapolating")
- Relationship between divided differences and derivatives:

$$\exists \xi \in [a, b] \quad \text{such that} \quad f[x_0, \ldots, x_n, x] = \frac{f^{(n+1)}(\xi)}{(n+1)!}.$$

- So take upper bounds to find

$$|e_n(x)| \le \max_{t \in [a,b]} \frac{|f^{(n+1)}(t)|}{(n+1)!} \max_{\mathbf{s} \in [a,b]} \left| \prod_{j=0}^{n} (s - x_j) \right| = \frac{\|f^{(n+1)}\|_\infty}{(n+1)!} \|\psi\|_\infty;$$

$$\|e_n\|_\infty \le \frac{\|f^{(n+1)}\|_\infty}{(n+1)!} (b-a)^{n+1}.$$

Example

Consider $\{(x_0, y_0),\ (x_1, y_1)\}$.
So $n = 1$, hence $n + 1 = 2$.

$$\|e_n\| \le \frac{1}{2}\|f''\| \max_s |(s - x_0)(s - x_1)|.$$

Max at $s = \frac{x_0+x_1}{2}$, so $\max_s |(s - x_0)(s - x_1)| = \frac{1}{4}(x_1 - x_0)^2$.
Thus

$$\|e_n\| \le \frac{1}{8}(x_1 - x_0)^2\|f''\|.$$

Example

Consider $\{(x_0, y_0),\ (x_1, y_1)\}$.
So $n = 1$, hence $n + 1 = 2$.

$$\|e_n\| \leq \frac{1}{2}\|f''\| \max_s |(s - x_0)(s - x_1)|.$$

Max at $s = \frac{x_0+x_1}{2}$, so $\max_s |(s - x_0)(s - x_1)| = \frac{1}{4}(x_1 - x_0)^2$.
Thus

$$\|e_n\| \leq \frac{1}{8}(x_1 - x_0)^2 \|f''\|.$$

Example

Consider $\{(x_0, y_0),\ (x_1, y_1)\}$.
So $n = 1$, hence $n + 1 = 2$.

$$\|e_n\| \le \frac{1}{2}\|f''\| \max_s |(s - x_0)(s - x_1)|.$$

Max at $s = \frac{x_0+x_1}{2}$, so $\max_s |(s - x_0)(s - x_1)| = \frac{1}{4}(x_1 - x_0)^2$.
Thus

$$\|e_n\| \le \frac{1}{8}(x_1 - x_0)^2\|f''\|.$$

Example

Consider $\{(x_0, y_0),\ (x_1, y_1)\}$.
So $n = 1$, hence $n + 1 = 2$.

$$\|e_n\| \leq \frac{1}{2}\|f''\| \max_s |(s - x_0)(s - x_1)|.$$

Max at $s = \frac{x_0 + x_1}{2}$, so $\max_s |(s - x_0)(s - x_1)| = \frac{1}{4}(x_1 - x_0)^2$.
Thus

$$\|e_n\| \leq \frac{1}{8}(x_1 - x_0)^2\|f''\|.$$

Outline

- Monomial basis
- Lagrange basis
- Newton basis and divided differences
- Interpolation error
- Chebyshev interpolation
- Interpolating also derivative values

Chebyshev interpolation

Can we minimize error bound? i.e., the right hand side of

$$\max_{a \le x \le b} |e_n(x)| \le \max_{t\in[a,b]} \frac{|f^{(n+1)}(t)|}{(n+1)!} \max_{\mathbf{s}\in[a,b]} \left| \prod_{j=0}^{n} (s - x_j) \right|$$

- Assume we can evaluate $f(x)$ at *any* $n+1$ points x_i. What should those be?
- Knowing nothing more about the interpolated function $f(x)$, choose the abscissae x_i attempting to minimize $\max_{\mathbf{s}\in[a,b]} \left| \prod_{j=0}^{n} (s - x_j) \right|$.
- This leads to **Chebyshev points**: over $a = -1,\ b = 1$ they are

$$x_i = \cos\left(\frac{2i+1}{2(n+1)}\pi\right), \quad i = 0, \ldots, n.$$

Chebyshev interpolation

Can we minimize error bound? i.e., the right hand side of

$$\max_{a \le x \le b} |e_n(x)| \le \max_{t \in [a,b]} \frac{|f^{(n+1)}(t)|}{(n+1)!} \max_{\mathbf{s} \in [a,b]} \left| \prod_{j=0}^{n} (s - x_j) \right|$$

- Assume we can evaluate $f(x)$ at *any* $n+1$ points x_i. What should those be?
- Knowing nothing more about the interpolated function $f(x)$, choose the abscissae x_i attempting to minimize $\max_{\mathbf{s} \in [a,b]} \left| \prod_{j=0}^{n} (s - x_j) \right|$.
- This leads to **Chebyshev points**: over $a = -1, \ b = 1$ they are

$$x_i = \cos\left(\frac{2i+1}{2(n+1)}\pi\right), \quad i = 0, \ldots, n.$$

Chebyshev interpolation

Can we minimize error bound? i.e., the right hand side of

$$\max_{a \le x \le b} |e_n(x)| \le \max_{t \in [a,b]} \frac{|f^{(n+1)}(t)|}{(n+1)!} \max_{\mathbf{s} \in [a,b]} \left| \prod_{j=0}^{n} (s - x_j) \right|$$

- Assume we can evaluate $f(x)$ at *any* $n+1$ points x_i. What should those be?
- Knowing nothing more about the interpolated function $f(x)$, choose the abscissae x_i attempting to minimize $\max_{\mathbf{s} \in [a,b]} \left| \prod_{j=0}^{n} (s - x_j) \right|$.
- This leads to **Chebyshev points**: over $a = -1, \ b = 1$ they are

$$x_i = \cos\left(\frac{2i+1}{2(n+1)}\pi\right), \quad i = 0, \ldots, n.$$

Chebyshev points

- These points solve the **min-max** problem

$$\beta = \min_{x_0, x_1, \ldots, x_n} \max_{-1 \le x \le 1} |(x - x_0)(x - x_1) \cdots (x - x_n)|,$$

yielding the value $\beta = 2^{-n}$.

- This leads to the Chebyshev interpolation error bound

$$\max_{-1 \le x \le 1} |f(x) - p_n(x)| \le \frac{1}{2^n (n+1)!} \max_{-1 \le t \le 1} |f^{(n+1)}(t)|.$$

- For a general interval $[a, b]$, scale and translate $[-1, 1]$ onto $[a, b]$

$$x = a + \frac{b-a}{2}(t + 1), \quad t \in [-1, 1].$$

Thus, shift and scale the Chebyshev points by

$$x_i \longleftarrow a + \frac{b-a}{2}(x_i + 1), \quad i = 0, \ldots, n.$$

Chebyshev points

- These points solve the **min-max** problem

$$\beta = \min_{x_0,x_1,\ldots,x_n} \max_{-1\le x\le 1} |(x-x_0)(x-x_1)\cdots(x-x_n)|,$$

  yielding the value $\beta = 2^{-n}$.

- This leads to the Chebyshev interpolation error bound

$$\max_{-1\le x\le 1} |f(x)-p_n(x)| \le \frac{1}{2^n(n+1)!} \max_{-1\le t\le 1} |f^{(n+1)}(t)|.$$

- For a general interval $[a,b]$, scale and translate $[-1,1]$ onto $[a,b]$

$$x = a + \frac{b-a}{2}(t+1), \quad t\in[-1,1].$$

  Thus, shift and scale the Chebyshev points by

$$x_i \longleftarrow a + \frac{b-a}{2}(x_i+1), \quad i=0,\ldots,n.$$

Chebyshev points

- These points solve the **min-max** problem

$$\beta = \min_{x_0, x_1, \ldots, x_n} \max_{-1 \le x \le 1} |(x - x_0)(x - x_1) \cdots (x - x_n)|,$$

  yielding the value $\beta = 2^{-n}$.

- This leads to the Chebyshev interpolation error bound

$$\max_{-1 \le x \le 1} |f(x) - p_n(x)| \le \frac{1}{2^n (n+1)!} \max_{-1 \le t \le 1} |f^{(n+1)}(t)|.$$

- For a general interval $[a, b]$, scale and translate $[-1, 1]$ onto $[a, b]$

$$x = a + \frac{b - a}{2}(t + 1), \quad t \in [-1, 1].$$

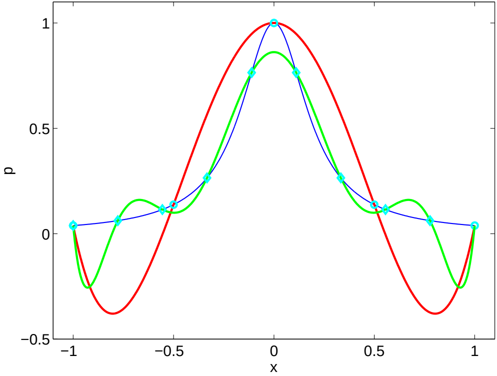
  Thus, shift and scale the Chebyshev points by

$$x_i \longleftarrow a + \frac{b - a}{2}(x_i + 1), \quad i = 0, \ldots, n.$$

Runge example

$$f(x) = \frac{1}{1+25x^2}, \quad -1 \leq x \leq 1.$$

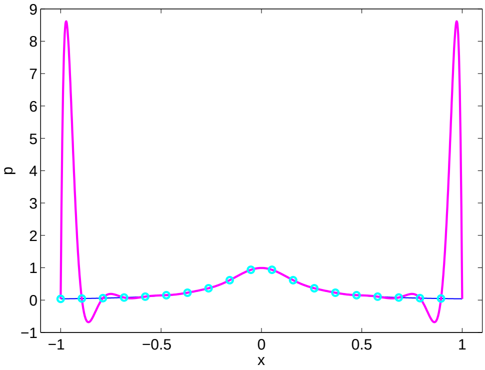
Equi-distant points, $n = 4, 9$.



Runge example

$$f(x) = \frac{1}{1+25x^2}, \quad -1 \le x \le 1.$$

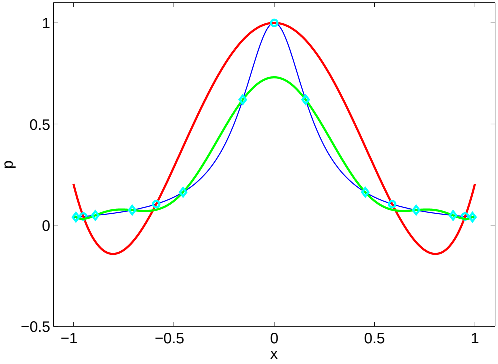
Equi-distant points, $n = 19$.



Runge example

$$f(x) = \frac{1}{1+25x^2}, \quad -1 \le x \le 1.$$

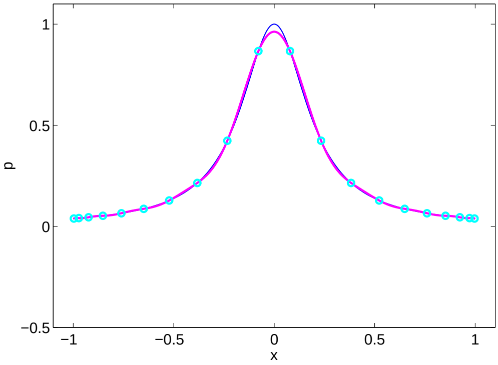
Chebyshev points, $n = 4, 9$.



Runge example

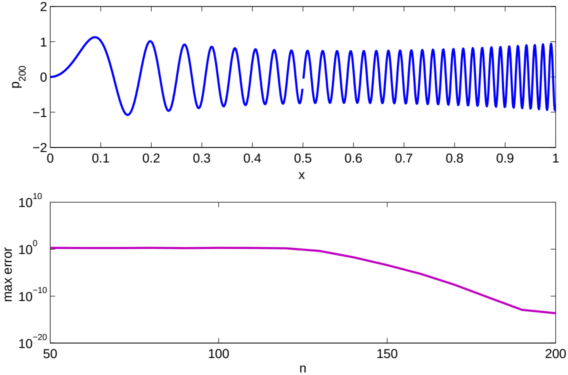
$$f(x) = \frac{1}{1+25x^2}, \quad -1 \le x \le 1.$$

Chebyshev points, $n = 19$.



More difficult example

$$f(x) = e^{3x}\sin(200x^2)/(1+20x^2), \quad 0 \le x \le 1.$$



Error does not change much at first as n increases, but then it decreases very rapidly: spectral accuracy.

Outline

- Monomial basis
- Lagrange basis
- Newton basis and divided differences
- Interpolation error
- Chebyshev interpolation
- Interpolating also derivative values

Interpolating also derivative values

Example: quadratic interpolant for

$$f(0) = 1.5, \quad f'(0) = 1, \quad f(20) = 0.$$

Hence

$$\{(x_i, y_i)\} = \{(0, 1.5), (0, 1), (20, 0)\}.$$

- Using the simplest basis, monomial:

  $$p(x) = c_0 + c_1 x + c_2 x^2.$$

  Then

  $$\begin{aligned} 1.5 = p(0) &= c_0, \\ 1 = p'(0) &= c_1, \\ 0 = p(20) &= c_0 + 20c_1 + 400c_2. \end{aligned}$$

  Hence $c_2 = (-1.5 - 20)/400 = -\frac{21.5}{400}$ and

  $$p(x) = 1.5 + x - \frac{21.5}{400} x^2.$$

Interpolating also derivative values

Example: quadratic interpolant for

$$f(0) = 1.5, \quad f'(0) = 1, \quad f(20) = 0.$$

Hence

$$\{(x_i, y_i)\} = \{(0, 1.5), (0, 1), (20, 0)\}.$$

- Using the simplest basis, monomial:

  $$p(x) = c_0 + c_1 x + c_2 x^2.$$

  Then

  $$\begin{aligned} 1.5 = p(0) &= c_0, \\ 1 = p'(0) &= c_1, \\ 0 = p(20) &= c_0 + 20c_1 + 400c_2. \end{aligned}$$

  Hence $c_2 = (-1.5 - 20)/400 = -\frac{21.5}{400}$ and

  $$p(x) = 1.5 + x - \frac{21.5}{400} x^2.$$

Hermite cubic

An important building block in computer aided design (CAD).

$$\{(x_i, y_i)\} = \{(t_0, f(t_0)), (t_0, f'(t_0)), (t_1, f(t_1)), (t_1, f'(t_1))\}.$$

- Using the simplest basis, monomial:

$$p(x) = c_0 + c_1 x + c_2 x^2 + c_3 x^3.$$

- Then form linear equations

$$\begin{aligned} c_0 + c_1 t_0 + c_2 t_0^2 + c_3 t_0^3 &= f(t_0), \quad & c_1 + 2c_2 t_0 + 3c_3 t_0^2 &= f'(t_0), \\ c_0 + c_1 t_1 + c_2 t_1^2 + c_3 t_1^3 &= f(t_1), \quad & c_1 + 2c_2 t_1 + 3c_3 t_1^2 &= f'(t_1), \end{aligned}$$

and solve for the coefficients c_j.

Hermite cubic

An important building block in computer aided design (CAD).

$$\{(x_i, y_i)\} = \{(t_0, f(t_0)), (t_0, f'(t_0)), (t_1, f(t_1)), (t_1, f'(t_1))\}.$$

- Using the simplest basis, monomial:

$$p(x) = c_0 + c_1 x + c_2 x^2 + c_3 x^3.$$

- Then form linear equations

$$c_0 + c_1 t_0 + c_2 t_0^2 + c_3 t_0^3 = f(t_0), \quad c_1 + 2c_2 t_0 + 3c_3 t_0^2 = f'(t_0),$$
$$c_0 + c_1 t_1 + c_2 t_1^2 + c_3 t_1^3 = f(t_1), \quad c_1 + 2c_2 t_1 + 3c_3 t_1^2 = f'(t_1),$$

  and solve for the coefficients c_j.

General algorithm

1. *Construction*: Given data $\{(x_i, y_i)\}_{i=0}^n$, where the abscissae are not necessarily distinct;

$$
\begin{aligned}
&\text{for } j = 0, 1, \ldots, n \\
&\quad \text{for } l = 0, 1, \ldots, j \\
&\qquad \gamma_{j,l} = \begin{cases} \frac{\gamma_{j,l-1} - \gamma_{j-1,l-1}}{x_j - x_{j-l}} & \text{if } x_j \neq x_{j-l}, \\ \frac{f^{(l)}(x_j)}{l!} & \text{otherwise} \end{cases}
\end{aligned}
$$

2. *Evaluation*: Given an evaluation point x,

$$
\begin{aligned}
&p = \gamma_{n,n} \\
&\text{for } j = n-1, n-2, \ldots, 0, \\
&\quad p = p\,(x - x_j) + \gamma_{j,j}
\end{aligned}
$$

Example

Given five data values

t_i	$f(t_i)$	$f'(t_i)$	$f''(t_i)$
8.3	17.564921	3.116256	0.120482
8.6	18.505155	3.151762	

set up

$$(x_0, x_1, x_2, x_3, x_4) = \left(\underbrace{8.3, 8.3, 8.3}_{m_0=2}, \underbrace{8.6, 8.6}_{m_1=1}\right),$$

Obtain

x_i	$f[\cdot]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$	$f[\cdot,\cdot,\cdot,\cdot]$	$f[\cdot,\cdot,\cdot,\cdot,\cdot]$
8.3	17.564921				
8.3	17.564921	<u>3.116256</u>			
8.3	17.564921	<u>3.116256</u>	<u>0.060241</u>		
8.6	18.505155	3.130780	0.048413	-0.039426	
8.6	18.505155	<u>3.151762</u>	0.069400	0.071756	0.370604