July 20, 2013

Chapter 11: Piecewise 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 explain what may go wrong with polynomial interpolation and how piecewise polynomials can fix it;
- to derive several important practical general interpolants;
- to derive error expressions for the piecewise polynomial interpolation process;
- to construct parametric interpolants, i.e., curves that are not necessarily functions $y = f(x)$; and
- *to consider interpolation in more than one space variable.

Outline

- Piecewise polynomials
- Local interpolants: C^0-linear and C^1-cubic
- Cubic spline interpolation
- Basis functions
- Parametric interpolation
- *Multidimensional interpolation

*advanced

What may go wrong with polynomial interpolation

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

- The error bound may not be small if $\frac{\|f^{(n+1)}\|}{(n+1)!}$ isn't.
- High order polynomials tend to oscillate "unreasonably".
- Data often suggest only piecewise smoothness, whereas polynomials are infinitely differentiable everywhere.
- No locality: changing any one data value may drastically alter the entire polynomial interpolant.

We will see in this chapter how all these concerns are addressed by applying the polynomial interpolation idea in a localized, piecewise manner.

What may go wrong with polynomial interpolation

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

- The error bound may not be small if $\frac{\|f^{(n+1)}\|}{(n+1)!}$ isn't.
- High order polynomials tend to oscillate "unreasonably".
- Data often suggest only piecewise smoothness, whereas polynomials are infinitely differentiable everywhere.
- No locality: changing any one data value may drastically alter the entire polynomial interpolant.

We will see in this chapter how all these concerns are addressed by applying the polynomial interpolation idea in a localized, piecewise manner.

Further major extensions of Chapter 10

There are important classes of interpolation problems where an extension of what we currently know about polynomial interpolation is required. Two of these are considered here:

- In a typical curve design application in CAD, there may not be a function $y = f(x)$ describing it. Rather, a parametric curve interplant is typically required.
- Everything hitherto has been in 1D (i.e., we have looked only at curves). Very often, extensions to surface interpolation in 2D and 3D ($v(x, y)$ and $v(x, y, z)$) are required.

Outline

- Piecewise polynomials
- Local interpolants: C^0-linear and C^1-cubic
- Cubic spline interpolation
- Basis functions
- Parametric interpolation
- *Multidimensional interpolation

Piecewise polynomial interpolation

- Divide given interval $[a, b]$ into subintervals

$$a = t_0 < t_1 < t_2 < \cdots < t_r = b$$

where t_i are the "breakpoints," or "knots".

- Construct interpolant

$$v(x) = s_i(x), \quad t_i \le x \le t_{i+1}, \ i = 0, \ldots, r-1$$

where each $s_i(x)$ is a polynomial of low degree (e.g., piecewise cubic: $m = 3$).

- As before $v(x)$ must satisfy the interpolation conditions. In addition, require a global smoothness property.

Piecewise polynomial: an illustration

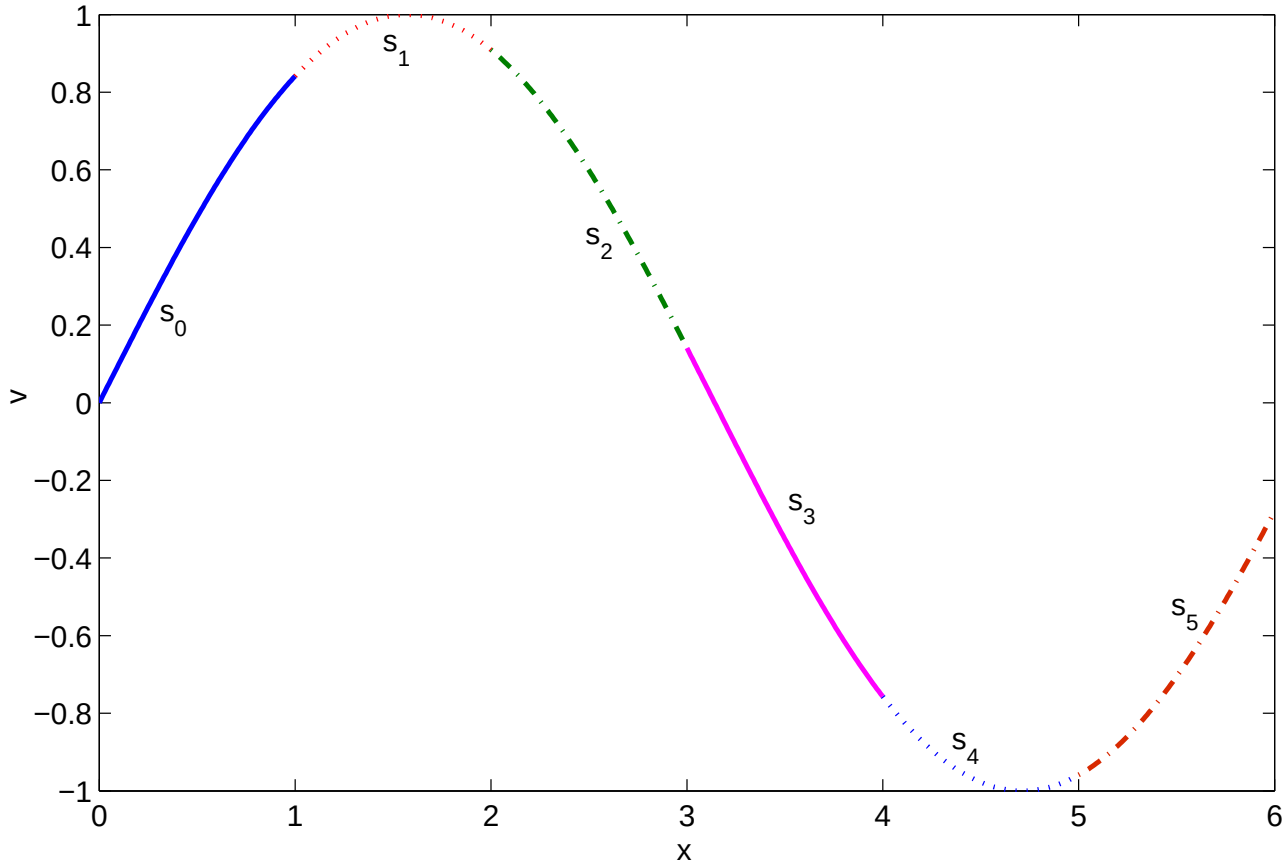


FIGURE: A piecewise polynomial function, with break points $t_i = i,\ i = 0, 1, \ldots, 6$.

t_i and x_i

Distinguish between abscissae x_i where data are specified

$$a \le x_0 \le x_1 \le \cdots \le x_{n-1} \le x_n \le b$$

and knots (break points) of the piecewise polynomial

$$a = t_0 < t_1 < \cdots < t_{r-1} < t_r = b$$

For future purposes, set

$$h = \max_{1 \le i \le r} (t_i - t_{i-1}).$$

Rely on h being small (while $b - a$ is not small) for quality approximation.

t_i and x_i

Distinguish between abscissae x_i where data are specified

$$a \le x_0 \le x_1 \le \cdots \le x_{n-1} \le x_n \le b$$

and knots (break points) of the piecewise polynomial

$$a = t_0 < t_1 < \cdots < t_{r-1} < t_r = b$$

For future purposes, set

$$h = \max_{1 \le i \le r} (t_i - t_{i-1}).$$

Rely on h being small (while $b - a$ is not small) for quality approximation.

Outline

- Piecewise polynomials
- Local interpolants: C^0-linear and C^1-cubic
- Cubic spline interpolation
- Basis functions
- Parametric interpolation
- *Multidimensional interpolation

Local interpolants: C^0-linear and C^1-cubic

- Local piecewise polynomial interpolants are simply constructed from local polynomial interpolation on each subinterval $[t_{i-1}, t_i]$.
- Some global smoothness conditions are automatically satisfied.
- Interpolation error bounds are easily derived from polynomial interpolation.
- The interpolant depends on the data only locally.
- Limitation: don't have control over how much global smoothness is easily achieved.

Piecewise linear interpolation

Also known as "**broken line**" interpolation; the simplest continuous interpolant.

- Assuming $\{x_i\}_{i=0}^n$ are distinct and ordered, identify $t_i = x_i$, $y_i = f(x_i)$.
- Determine $s_i(x)$ as the straight line interpolant at x_i and x_{i+1}. Thus, $m = 1$

$$v(x) = s_i(x) = f(x_i) + f[x_i, x_{i+1}](x - x_i), \quad x_i \le x \le x_{i+1}, \quad 0 \le i \le n-1.$$

- Inherently, $v(x)$ is continuous but not smoother than that.
- No new minima and maxima are "invented" by $v(x)$ for $f(x)$.
- MATLAB's command `plot` uses this interplant by default.

Polynomial error estimate and bound

Recall:

- Assume $x \in [t_{i-1}, t_i]$ for some $i,\ 1 \le i \le n$. Consider polynomial interpolation of degree m with abscissae $x_0, x_1, \ldots, x_m \in [t_{i-1}, t_i]$.
- Then

$$|e_m(x)| \le \max_{\xi \in [t_{i-1}, t_i]} \frac{|f^{(m+1)}(\xi)|}{(m+1)!} \max_{\zeta \in [t_{i-1}, t_i]} \left| \prod_{j=0}^{m} (\zeta - x_j) \right|.$$

Error analysis for piecewise linear

- Linear polynomials: $m = 1,\ t_i = x_i$, so for $x_{i-1} \le x \le x_i$

$$\begin{aligned} |f(x) - v(x)| &\le \max_{\xi \in [x_{i-1}, x_i]} \frac{|f''(\xi)|}{2} \max_{\zeta \in [x_{i-1}, x_i]} |(\zeta - x_{i-1})(\zeta - x_i)| \\ &= \max_{\xi \in [x_{i-1}, x_i]} \frac{|f''(\xi)|}{2} \left| \frac{x_i - x_{i-1}}{2} \right|^2. \end{aligned}$$

- Recall notation $h = \max_i (t_i - t_{i-1})$. Then on the entire interval $[a, b]$,

$$\|f - v\|_\infty \le \frac{h^2}{8} \|f''\|_\infty.$$

- The error is $\mathcal{O}(h^2)$. If we have $n+1$ equi-distant data points, then $h = (b-a)/n$ and the error is $\mathcal{O}(\frac{1}{n^2})$.

Error analysis for piecewise linear

- Linear polynomials: $m = 1,\ t_i = x_i$, so for $x_{i-1} \le x \le x_i$

$$\begin{aligned} |f(x) - v(x)| &\le \max_{\xi \in [x_{i-1}, x_i]} \frac{|f''(\xi)|}{2} \max_{\zeta \in [x_{i-1}, x_i]} |(\zeta - x_{i-1})(\zeta - x_i)| \\ &= \max_{\xi \in [x_{i-1}, x_i]} \frac{|f''(\xi)|}{2} \left| \frac{x_i - x_{i-1}}{2} \right|^2. \end{aligned}$$

- Recall notation $h = \max_i (t_i - t_{i-1})$. Then on the entire interval $[a, b]$,

$$\|f - v\|_\infty \le \frac{h^2}{8} \|f''\|_\infty.$$

- The error is $\mathcal{O}(h^2)$. If we have $n+1$ equi-distant data points, then $h = (b-a)/n$ and the error is $\mathcal{O}(\frac{1}{n^2})$.

Error analysis for piecewise linear

- Linear polynomials: $m = 1,\ t_i = x_i$, so for $x_{i-1} \le x \le x_i$

$$\begin{aligned} |f(x) - v(x)| &\le \max_{\xi \in [x_{i-1}, x_i]} \frac{|f''(\xi)|}{2} \max_{\zeta \in [x_{i-1}, x_i]} |(\zeta - x_{i-1})(\zeta - x_i)| \\ &= \max_{\xi \in [x_{i-1}, x_i]} \frac{|f''(\xi)|}{2} \left| \frac{x_i - x_{i-1}}{2} \right|^2. \end{aligned}$$

- Recall notation $h = \max_i (t_i - t_{i-1})$. Then on the entire interval $[a, b]$,

$$\|f - v\|_\infty \le \frac{h^2}{8} \|f''\|_\infty.$$

- The error is $\mathcal{O}(h^2)$. If we have $n+1$ equi-distant data points, then $h = (b-a)/n$ and the error is $\mathcal{O}(\frac{1}{n^2})$.

Piecewise cubic interpolation

Most popular in applications other than plot.

- $$v(x) = s_i(x) = a_i + b_i(x - t_i) + c_i(x - t_i)^2 + d_i(x - t_i)^3,$$
  $$t_i \le x \le t_{i+1},\ i = 0, \ldots, r-1.$$

- We have $4r$ coefficients. Of these, $2r$ are used to satisfy interpolation of f values as in piecewise linear case. What about the remaining $2r$ coefficients?
- Use either to interpolate more data or to require higher degree of smoothness.
- The latter option leads to a C^1-cubic, which is a local piecewise polynomial interpolant.

Piecewise cubic interpolation

Most popular in applications other than plot.

- $$v(x) = s_i(x) = a_i + b_i(x - t_i) + c_i(x - t_i)^2 + d_i(x - t_i)^3,$$
  $$t_i \le x \le t_{i+1}, \ i = 0, \ldots, r-1.$$

- We have $4r$ coefficients. Of these, $2r$ are used to satisfy interpolation of f values as in piecewise linear case. What about the remaining $2r$ coefficients?
- Use either to interpolate more data or to require higher degree of smoothness.
- The latter option leads to a C^1-cubic, which is a local piecewise polynomial interpolant.

Piecewise cubic interpolation

Most popular in applications other than plot.

- $$v(x) = s_i(x) \quad = \quad a_i + b_i(x - t_i) + c_i(x - t_i)^2 + d_i(x - t_i)^3,$$
$$t_i \le x \le t_{i+1}, \ \ i = 0, \ldots, r-1.$$

- We have $4r$ coefficients. Of these, $2r$ are used to satisfy interpolation of f values as in piecewise linear case. What about the remaining $2r$ coefficients?
- Use either to interpolate more data or to require higher degree of smoothness.
- The latter option leads to a C^1-cubic, which is a local piecewise polynomial interpolant.

Piecewise cubic Hermite interpolation

Cleanest piecewise cubic.

- Assume values of $f'(t_i)$ are also provided. So

$$(x_0, x_1, x_2, \ldots, x_{n-1}, x_n) = (t_0, t_0, t_1, t_1, \ldots, t_r, t_r).$$

- Each cubic $s_i(x), \quad 0 \le i \le r-1$, satisfies

$$s_i(t_i) = f(t_i), \quad s_i(t_{i+1}) = f(t_{i+1}),$$
$$s_i'(t_i) = f'(t_i), \quad s_i'(t_{i+1}) = f'(t_{i+1}).$$

- Obtain $v(x) \in C^1[a,b]$. Construction entirely local!
- Error bound obtained directly from polynomial theory:

$$|f(x) - v(x)| \le \frac{h^4}{384} \max_{a \le \xi \le b} |f''''(\xi)|.$$

Piecewise cubic Hermite interpolation

Cleanest piecewise cubic.

- Assume values of $f'(t_i)$ are also provided. So

$$(x_0, x_1, x_2, \ldots, x_{n-1}, x_n) = (t_0, t_0, t_1, t_1, \ldots, t_r, t_r).$$

- Each cubic $s_i(x), \quad 0 \le i \le r-1$, satisfies

$$s_i(t_i) = f(t_i), \quad s_i(t_{i+1}) = f(t_{i+1}),$$
$$s_i'(t_i) = f'(t_i), \quad s_i'(t_{i+1}) = f'(t_{i+1}).$$

- Obtain $v(x) \in C^1[a,b]$. Construction entirely local!
- Error bound obtained directly from polynomial theory:

$$|f(x) - v(x)| \le \frac{h^4}{384} \max_{a \le \xi \le b} |f''''(\xi)|.$$

Piecewise cubic Hermite interpolation

Cleanest piecewise cubic.

- Assume values of $f'(t_i)$ are also provided. So

$$(x_0, x_1, x_2, \ldots, x_{n-1}, x_n) = (t_0, t_0, t_1, t_1, \ldots, t_r, t_r).$$

- Each cubic $s_i(x), \quad 0 \le i \le r-1$, satisfies

$$s_i(t_i) = f(t_i), \quad s_i(t_{i+1}) = f(t_{i+1}),$$
$$s_i'(t_i) = f'(t_i), \quad s_i'(t_{i+1}) = f'(t_{i+1}).$$

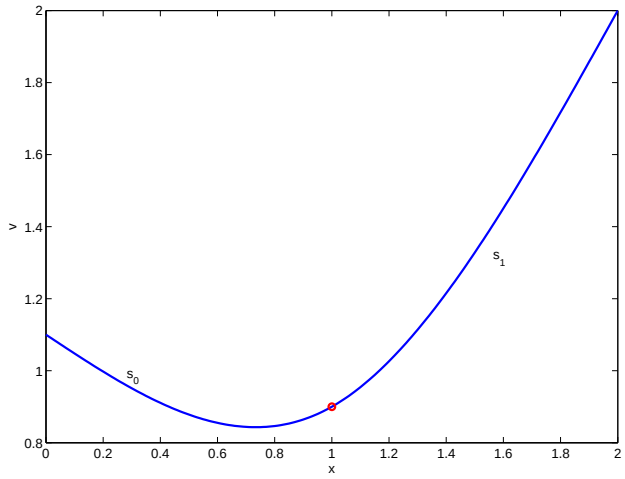
- Obtain $v(x) \in C^1[a,b]$. Construction entirely local!
- Error bound obtained directly from polynomial theory:

$$|f(x) - v(x)| \le \frac{h^4}{384} \max_{a \le \xi \le b} |f''''(\xi)|.$$

Outline

- Piecewise polynomials
- Local interpolants: C^0-linear and C^1-cubic
- Cubic spline interpolation
- Basis functions
- Parametric interpolation
- *Multidimensional interpolation

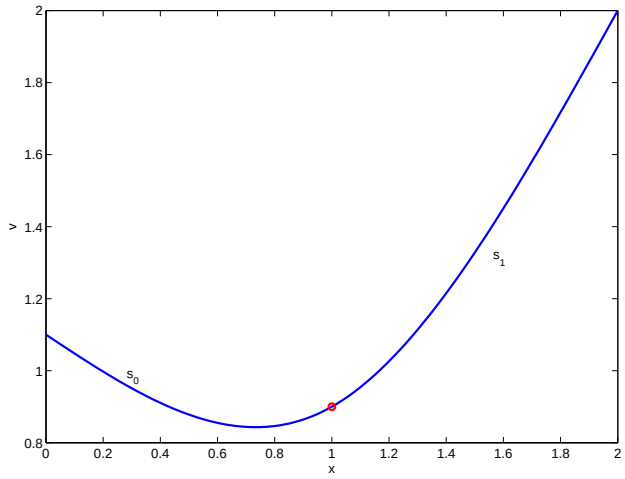
Cubic spline interpolation



Most popular general-purpose interpolant.

- We are given only function values, $\{x_i, f(x_i)\}, \quad x_0 < x_1 < \cdots < x_n$.
- Set $t_i = x_i, \ r = n$ and use $2r$ condition for C^0 interpolation as before.
- Use $2(r-1)$ additional coefficients to impose a global C^2 smoothness. (This is sufficient to create smooth-looking curves.)
- What to do with the remaining two coefficients? (Sometimes, freedom's just another word for something else to do!)

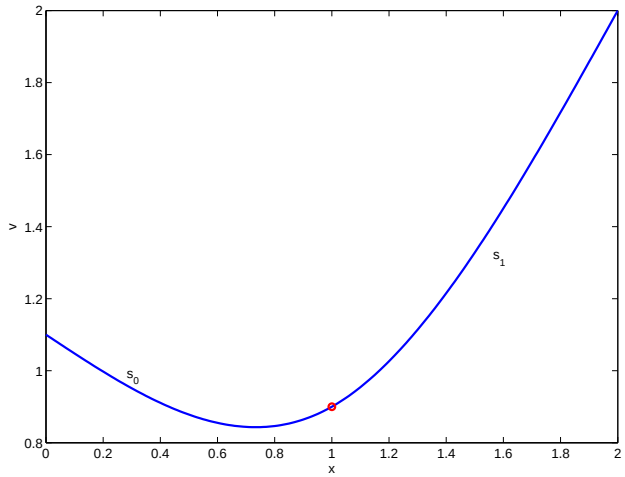
Cubic spline interpolation



Most popular general-purpose interpolant.

- We are given only function values, $\{x_i, f(x_i)\}, \quad x_0 < x_1 < \cdots < x_n$.
- Set $t_i = x_i, \ r = n$ and use $2r$ condition for C^0 interpolation as before.
- Use $2(r-1)$ additional coefficients to impose a global C^2 smoothness. (This is sufficient to create smooth-looking curves.)
- What to do with the remaining two coefficients? (Sometimes, freedom's just another word for something else to do!)

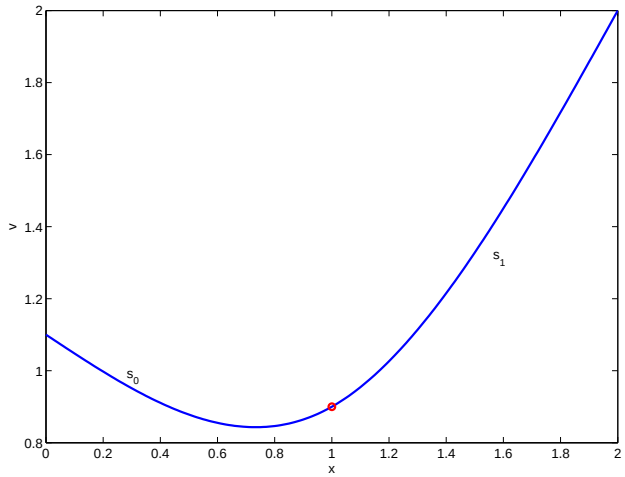
Cubic spline interpolation



Most popular general-purpose interpolant.

- We are given only function values, $\{x_i, f(x_i)\}, \quad x_0 < x_1 < \cdots < x_n$.
- Set $t_i = x_i, \ r = n$ and use $2r$ condition for C^0 interpolation as before.
- Use $2(r-1)$ additional coefficients to impose a global C^2 smoothness. (This is sufficient to create smooth-looking curves.)
- What to do with the remaining two coefficients?
  (Sometimes, freedom's just another word for something else to do!)

Cubic spline interpolation



Most popular general-purpose interpolant.

- We are given only function values, $\{x_i, f(x_i)\}, \quad x_0 < x_1 < \cdots < x_n$.
- Set $t_i = x_i, \ r = n$ and use $2r$ condition for C^0 interpolation as before.
- Use $2(r-1)$ additional coefficients to impose a global C^2 smoothness. (This is sufficient to create smooth-looking curves.)
- What to do with the remaining two coefficients? (Sometimes, freedom's just another word for something else to do!)

Example

$$\{(x_i, y_i)\} = \{(0, 1.1), (1, .9), (2, 2)\}$$

- $$v(x) = \begin{cases} s_0(x) = a_0 + b_0(x-0) + c_0(x-0)^2 + d_0(x-0)^3, & x < 1 \\ s_1(x) = a_1 + b_1(x-1) + c_1(x-1)^2 + d_1(x-1)^3, & x \geq 1 \end{cases}.$$

  Must determine the eight coefficients $a_0, b_0, c_0, d_0, a_1, b_1, c_1, d_1$.

- $$\begin{aligned} 1.1 &= v(0) = s_0(0) = a_0, \\ 0.9 &= v(1) = s_1(1) = a_1, \end{aligned}$$

  determine $a_0 = 1.1$ and $a_1 = 0.9$.

Example

$$\{(x_i, y_i)\} = \{(0, 1.1), (1, .9), (2, 2)\}$$

- $$v(x) = \begin{cases} s_0(x) = a_0 + b_0(x-0) + c_0(x-0)^2 + d_0(x-0)^3, & x < 1 \\ s_1(x) = a_1 + b_1(x-1) + c_1(x-1)^2 + d_1(x-1)^3, & x \geq 1 \end{cases}.$$

  Must determine the eight coefficients $a_0, b_0, c_0, d_0, a_1, b_1, c_1, d_1$.

- $$\begin{aligned} 1.1 &= v(0) = s_0(0) = a_0, \\ 0.9 &= v(1) = s_1(1) = a_1, \end{aligned}$$

  determine $a_0 = 1.1$ and $a_1 = 0.9$.

Example

$$\{(x_i, y_i)\} = \{(0, 1.1), (1, .9), (2, 2)\}$$

- $$v(x) = \begin{cases} s_0(x) = a_0 + b_0(x-0) + c_0(x-0)^2 + d_0(x-0)^3, & x < 1 \\ s_1(x) = a_1 + b_1(x-1) + c_1(x-1)^2 + d_1(x-1)^3, & x \geq 1 \end{cases}.$$

  Must determine the eight coefficients $a_0, b_0, c_0, d_0, a_1, b_1, c_1, d_1$.

- $$\begin{aligned} 1.1 &= v(0) = s_0(0) = a_0, \\ 0.9 &= v(1) = s_1(1) = a_1, \end{aligned}$$

  determine $a_0 = 1.1$ and $a_1 = 0.9$.

Example cont.

- Evaluating at right subinterval ends

$$\begin{aligned} 0.9 &= v(1) = s_0(1) = 1.1 + b_0 + c_0 + d_0, \\ 2.0 &= v(2) = s_1(2) = 0.9 + b_1 + c_1 + d_1. \end{aligned}$$

  Equating $s_0'(x_1) = s_1'(x_1)$ and $s_0''(x_1) = s_1''(x_1)$, where $x_1 = 1$, gives two more conditions

$$\begin{aligned} b_0 + 2c_0 + 3d_0 &= b_1, \\ 2c_0 + 6d_0 &= 2c_1. \end{aligned}$$

Obtain an underdetermined system of 4 linear equations for 6 unknowns. Need two more conditions!

Two additional conditions

- Free boundary, natural spline

  $$v''(x_0) = v''(x_n) = 0.$$

  Leads to lower accuracy of $\mathcal{O}(h^2)$.

- Clamped boundary, complete spline

  $$v'(x_0) = f'(x_0), \quad v'(x_n) = f'(x_n).$$

  Accuracy $\mathcal{O}(h^4)$, good for theory, but requires extra data.

- Not-a-knot:
  Ensure continuity of 3rd derivative at x_1, x_{n-1}. This effectively cancels these knots. It is the option used in practice, e.g., in MATLAB's function `spline`.

Two additional conditions

- Free boundary, natural spline

  $$v''(x_0) = v''(x_n) = 0.$$

  Leads to lower accuracy of $\mathcal{O}(h^2)$.
- Clamped boundary, complete spline

  $$v'(x_0) = f'(x_0), \quad v'(x_n) = f'(x_n).$$

  Accuracy $\mathcal{O}(h^4)$, good for theory, but requires extra data.
- Not-a-knot:
  Ensure continuity of 3rd derivative at x_1, x_{n-1}. This effectively cancels these knots. It is the option used in practice, e.g., in MATLAB's function `spline`.

Two additional conditions

- Free boundary, natural spline

  $$v''(x_0) = v''(x_n) = 0.$$

  Leads to lower accuracy of $\mathcal{O}(h^2)$.
- Clamped boundary, complete spline

  $$v'(x_0) = f'(x_0), \quad v'(x_n) = f'(x_n).$$

  Accuracy $\mathcal{O}(h^4)$, good for theory, but requires extra data.
- Not-a-knot:
  Ensure continuity of 3rd derivative at $x_1,\ x_{n-1}$. This effectively cancels these knots. It is the option used in practice, e.g., in MATLAB's function `spline`.

The Runge example using cubic spline interpolation

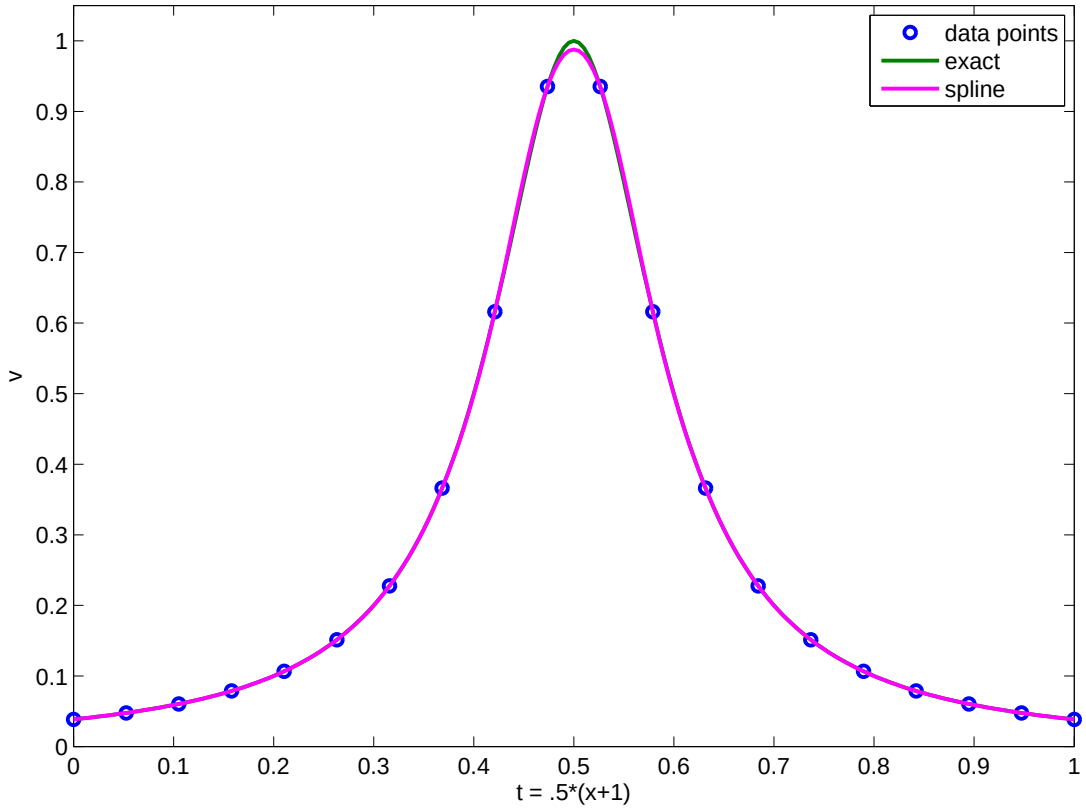


FIGURE: Not-a-knot cubic spline interpolation for the Runge Example at 20 equidistant points. The interval has been rescaled to be $[0, 1]$.

Obtaining the cubic spline

- Form and solve tridiagonal system of size $n(\pm 1)$.
- Construction cost is still linear in n, but approximation is not entirely local. Still, much better than polynomial interpolation.

Interpolant	Local?	Order	Smooth?	Selling features
Piecewise constant	yes	1	bounded	Accommodates discontinuous f
Broken line	yes	2	C^0	Simple, max and min at data values
Piecewise cubic Hermite	yes	4	C^1	Elegant and accurate
Spline (not-a-knot)	not quite	4	C^2	Accurate, smooth, requires only f data

Obtaining the cubic spline

- Form and solve tridiagonal system of size $n(\pm 1)$.
- Construction cost is still linear in n, but approximation is not entirely local. Still, much better than polynomial interpolation.

Interpolant	Local?	Order	Smooth?	Selling features
Piecewise constant	yes	1	bounded	Accommodates discontinuous f
Broken line	yes	2	C^0	Simple, max and min at data values
Piecewise cubic Hermite	yes	4	C^1	Elegant and accurate
Spline (not-a-knot)	not quite	4	C^2	Accurate, smooth, requires only f data

Outline

- Piecewise polynomials
- Local interpolants: C^0-linear and C^1-cubic
- Cubic spline interpolation
- Basis functions
- Parametric interpolation
- *Multidimensional interpolation

Basis functions

But what about basis functions?
For polynomial interpolation we saw three different choices of basis functions $\{\phi_j\}$ such that $v(x) = \sum_{j=1}^{n} c_j \phi_j(x)$. For piecewise polynomial interpolation, so far, we have seen none. Are there corresponding basis functions for piecewise polynomials? and if yes, are they good for anything?

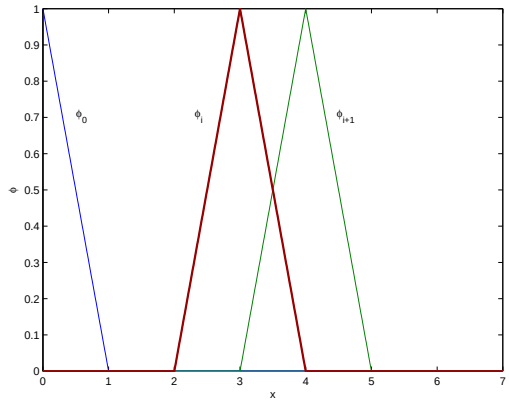
- Yes, there are such basis functions, and they are important.
- Choose them so as to have local support (i.e., each is nonzero only over a few subintervals).
- Good for interpolating functions that are given implicitly, for instance as solution of a differential equation.

Basis functions

But what about basis functions?
For polynomial interpolation we saw three different choices of basis functions $\{\phi_j\}$ such that $v(x) = \sum_{j=1}^{n} c_j \phi_j(x)$. For piecewise polynomial interpolation, so far, we have seen none. Are there corresponding basis functions for piecewise polynomials? and if yes, are they good for anything?

- Yes, there are such basis functions, and they are important.
- Choose them so as to have local support (i.e., each is nonzero only over a few subintervals).
- Good for interpolating functions that are given implicitly, for instance as solution of a differential equation.

Hat functions



- Basis functions for piecewise linear interpolation.
- Local support: $\phi_j(x) = 0$ outside subinterval (t_{j-1}, t_{j+1}).
- Nodal basis functions: $\phi_j(x_i) = \delta_{i,j}$.
- They are all obtained by scaling and translation of one mother hat function

$$\phi(z) = \begin{cases} 1+z, & -1 \le z < 0, \\ 1-z, & 0 \le z < 1, \\ 0, & \text{otherwise.} \end{cases}$$

Hermite cubic basis functions

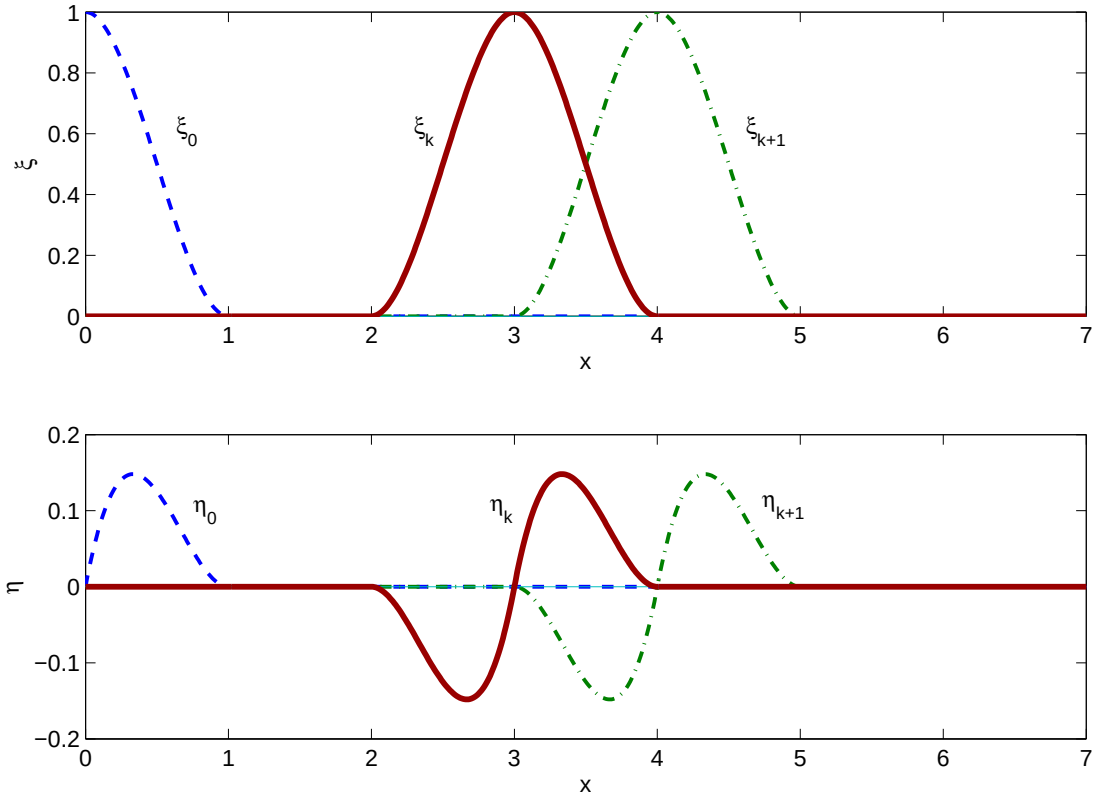


FIGURE: Basis functions for piecewise Hermite polynomials.

Two mother functions.

B-splines

A local basis methodology for different piecewise polynomial schemes, including C^2-cubics.

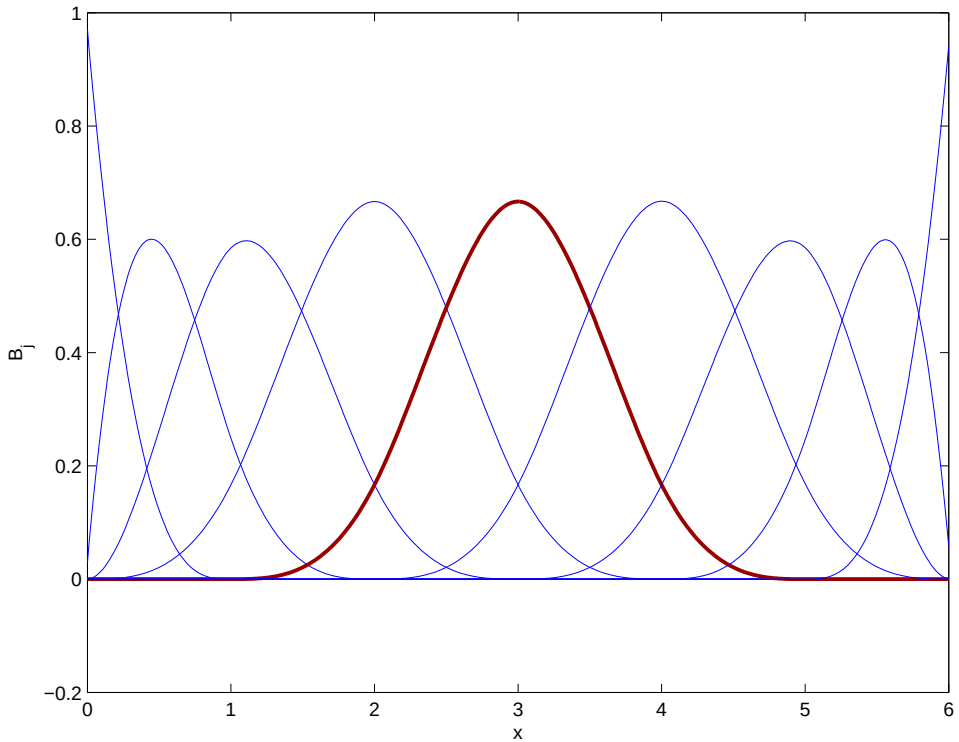


FIGURE: B-spline basis for the C^2 cubic spline.

Outline

- Piecewise polynomials
- Local interpolants: C^0-linear and C^1-cubic
- Cubic spline interpolation
- Basis functions
- Parametric interpolation
- *Multidimensional interpolation

Parametric interpolation

Generally, the given data $\{(x_i, y_i)\}_{i=0}^n$ may not correspond to a uni-valued function $y = f(x)$.

- Parametrize both abscissa x_i and ordinate y_i by same parameter τ:

$$(\tau_0, x_0), (\tau_1, x_1), (\tau_2, x_2), \ldots, (\tau_n, x_n)$$
$$(\tau_0, y_0), (\tau_1, y_1), (\tau_2, y_2), \ldots, (\tau_n, y_n).$$

- Simplest parametric choice is good: set $\tau_i = i/n, \quad i = 0, 1, \ldots, n.$
- Interpolate as before, twice:

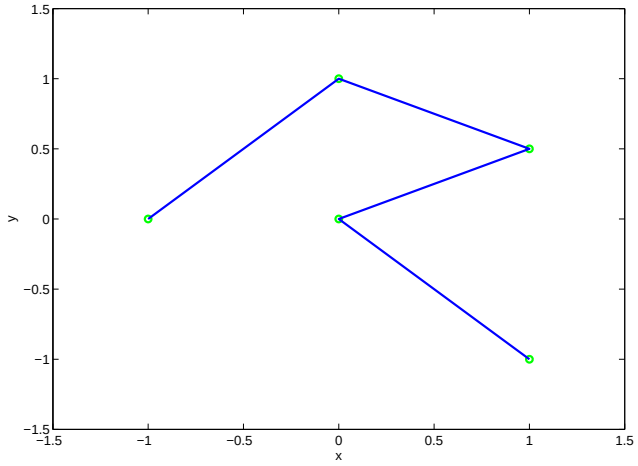
$$X(\tau_i) = x_i, \qquad Y(\tau_i) = y_i, \quad i = 0, 1, \ldots, n.$$

- The resulting parametric curve is

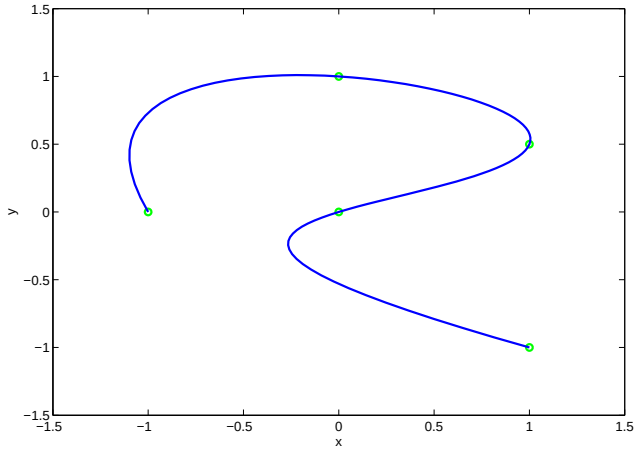
$$C : (X(\tau), Y(\tau)).$$

Example: two parameteric interpolants

1. Broken-line interpolation (implemented in MATLAB's `plot`)



2. Parametric polynomial interpolation of same data



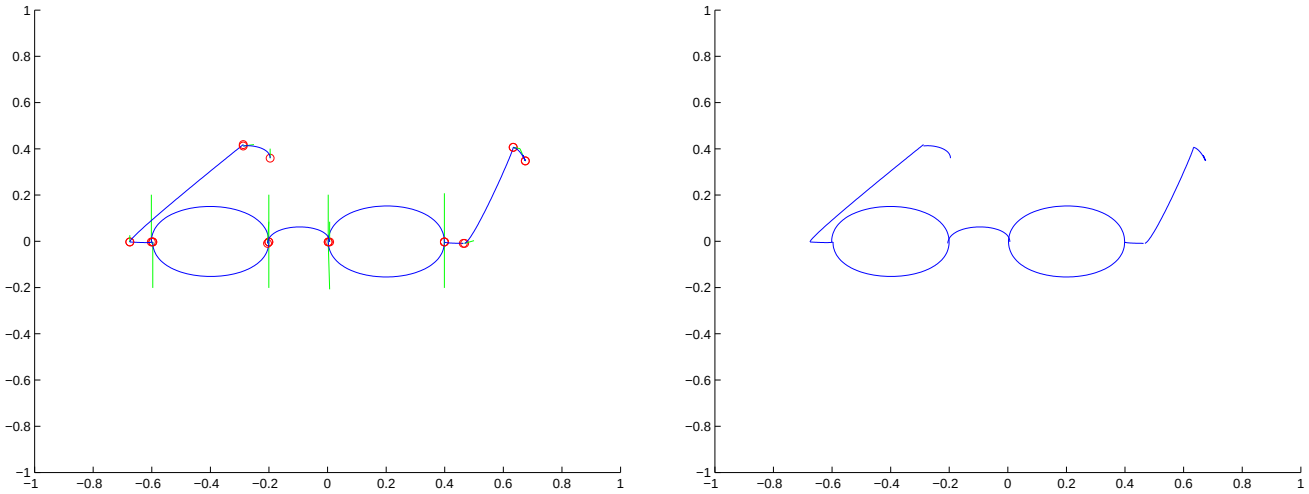
Designing curves

- Use Hermite cubics to design curves.
- Given data points $(x_0, y_0), \quad (x_1, y_1)$, specify **guidepoints** $(x_0 + \alpha_0, y_0 + \beta_0), \quad (x_1 - \alpha_1, y_1 - \beta_1)$. Then, the software constructs the two Hermite cubics $X(\tau), \quad Y(\tau)$ satisfying

$$X(0) = x_0, X(1) = x_1, X'(0) = 3\alpha_0, X'(1) = 3\alpha_1,$$
$$Y(0) = y_0, Y(1) = y_1, Y'(0) = 3\beta_0, Y'(1) = 3\beta_1.$$

- Bezier polynomials

Example: glasses design



(a) with guidepoints displayed (green)

(b) guidepoints hidden

FIGURE: A simple curve design using 11 Bezier polynomials.

Outline

- Piecewise polynomials
- Local interpolants: C^0-linear and C^1-cubic
- Cubic spline interpolation
- Basis functions
- Parametric interpolation
- *Multidimensional interpolation

Multidimensional interpolation

- A large number of additional issues and methods arise when extending from one dimension to several. We only touch briefly upon a few items.
- For instance, consider $v(x,y) = \sum_{j=0}^{n} c_j \phi_j(x,y)$ satisfying

$$v(x_i, y_i) = f_i, \quad i = 0, 1, \ldots, n.$$

- Is the data scattered or located on a grid?
- What is the purpose of the interpolating function? – where is it expected to be evaluated and how often?
- Special cases:
  - Orderly interpolation on a rectangular mesh, e.g. bilinear interpolation
  - Triangle meshes, surface meshes, and finite elements
  - Scattered data and radial basis functions

Multidimensional interpolation

- A large number of additional issues and methods arise when extending from one dimension to several. We only touch briefly upon a few items.
- For instance, consider $v(x,y) = \sum_{j=0}^{n} c_j \phi_j(x,y)$ satisfying

$$v(x_i, y_i) = f_i, \quad i = 0, 1, \ldots, n.$$

- Is the data scattered or located on a grid?
- What is the purpose of the interpolating function? – where is it expected to be evaluated and how often?
- Special cases:
  - Orderly interpolation on a rectangular mesh, e.g. bilinear interpolation
  - Triangle meshes, surface meshes, and finite elements
  - Scattered data and radial basis functions

Multidimensional interpolation

- A large number of additional issues and methods arise when extending from one dimension to several. We only touch briefly upon a few items.
- For instance, consider $v(x,y) = \sum_{j=0}^{n} c_j \phi_j(x,y)$ satisfying

$$v(x_i, y_i) = f_i, \quad i = 0, 1, \ldots, n.$$

- Is the data scattered or located on a grid?
- What is the purpose of the interpolating function? – where is it expected to be evaluated and how often?
- Special cases:
  - Orderly interpolation on a rectangular mesh, e.g. bilinear interpolation
  - Triangle meshes, surface meshes, and finite elements
  - Scattered data and radial basis functions

Bilinear interpolation

- Assume that there are abscissae in x and y separately, e.g., $x_0 < x_1 < \cdots < x_{N_x}$ and $y_0 < y_1 < \cdots < y_{N_y}$, such that the $n+1 = (N_x+1)(N_y+1)$ data points are given as $(x_i, y_j, f_{i,j})$.
- So, the interpolation problem is to find $v(x,y)$ satisfying

$$v(x_i, y_j) = f_{i,j}, \quad i = 0, 1, \ldots, N_x, \quad j = 0, 1, \ldots, N_y.$$

- In this special case we can think of seeking $v(x,y)$ in the tensor product form

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

where ϕ_k and ψ_l are **one-dimensional basis functions**.

- In the simple case of linear in each direction, can use hat basis functions for ϕ_k and ψ_l, and obtain a local bilinear interplant in $x \times y$.

Bilinear interpolation at square corners

- Since piecewise bilinear interpolation is local, can concentrate one bilinear polynomial on a square.
- So, consider data at the four corners of the unit square, and the interpolant

$$v(x,y) = c_0 + c_1 x + c_2 y + c_3 xy, \quad 0 \le x \le 1,\ 0 \le y \le 1.$$

- Substituting the data directly gives

$$\begin{aligned} & & c_0 &= f_{0,0}, \\ c_0 + c_1 = f_{1,0} \ &\Rightarrow & c_1 &= f_{1,0} - f_{0,0}, \\ c_0 + c_2 = f_{0,1} \ &\Rightarrow & c_2 &= f_{0,1} - f_{0,0}, \\ c_0 + c_1 + c_2 + c_3 = f_{1,1} \ &\Rightarrow & c_3 &= f_{1,1} + f_{0,0} - f_{0,1} - f_{1,0}. \end{aligned}$$

Interpolating at mid edges and center of square

- Occasionally, the data are placed at other locations in the square.
- The resulting bilinear interpolation formulas are still simple:

$$
\begin{aligned}
v(.5,0) &= c_0 + .5c_1 = \frac{1}{2}(f_{0,0} + f_{1,0}),\\
v(0,.5) &= c_0 + .5c_2 = \frac{1}{2}(f_{0,0} + f_{0,1}),\\
v(.5,1) &= c_0 + c_2 + .5(c_1 + c_3) = \frac{1}{2}(f_{0,1} + f_{1,1}),\\
v(1,.5) &= c_0 + c_1 + .5(c_2 + c_3) = \frac{1}{2}(f_{1,0} + f_{1,1}),\\
v(.5,.5) &= c_0 + .5c_1 + .5c_2 + .25c_3 = \frac{1}{4}(f_{0,0} + f_{1,0} + f_{0,1} + f_{1,1}).
\end{aligned}
$$

Bilinear interpolation: an example

- Simplest: given values vc on a coarse $N/2 \times N/2$ square mesh (N even), interpolate into a doubly fine array v on an $N \times N$ mesh that contains the coarse mesh. Assume these meshes cover the interior of a square and $v = 0$ on the square's boundary.
- Here is a MATLAB script:

```
v = zeros(N,N);
v(2:2:N-1,2:2:N-1) = vc; % inject given values to fine mesh
vz = [zeros(1,N);v;zeros(1,N)]; % embed v with a ring of 0s
vz = [zeros(N+2,1),vz,zeros(N+2,1)]; % complete 0 ring
% Interpolate along x edges, then y edges, then in cell middles
v(1:2:N,2:2:N-1) = .5*(vz(1:2:N,3:2:N)+vz(3:2:N+2,3:2:N));
v(2:2:N-1,1:2:N) = .5*(vz(3:2:N,1:2:N)+vz(3:2:N,3:2:N+2));
v(1:2:N,1:2:N) = .25*(vz(1:2:N,1:2:N)+vz(1:2:N,3:2:N+2)+...
                      vz(3:2:N+2,3:2:N+2)+vz(3:2:N+2,1:2:N));
```

Triangle mesh

- Domains in 2D often have a general shape that is not easily described as a union of a few rectangles.
- In such a case it is often easier to approximate the domain by a union of triangles. See example in the following slide.
- On the unit triangle with vertices $(0,0),\ (1,0),\ (0,1)$, suppose we are given values $f_{0,0},\ f_{1,0},\ f_{0,1}$ to interpolate.
- Then the linear interplant

$$v(x,y) = c_0 + c_1 x + c_2 y$$

  is **linear**, not bilinear! (no quadratic term xy).
- The coefficients $c_0,\ c_1$ and c_2 are easily determined by the three interpolation conditions.
- On a triangle mesh a piecewise linear interplant is globally continuous, because of interpolation of two common vertex values at adjacent triangles.

Triangle mesh

- Domains in 2D often have a general shape that is not easily described as a union of a few rectangles.
- In such a case it is often easier to approximate the domain by a union of triangles. See example in the following slide.
- On the unit triangle with vertices $(0,0),\ (1,0),\ (0,1)$, suppose we are given values $f_{0,0},\ f_{1,0},\ f_{0,1}$ to interpolate.
- Then the linear interplant

$$v(x,y) = c_0 + c_1 x + c_2 y$$

  is **linear**, not bilinear! (no quadratic term xy).
- The coefficients $c_0,\ c_1$ and c_2 are easily determined by the three interpolation conditions.
- On a triangle mesh a piecewise linear interplant is globally continuous, because of interpolation of two common vertex values at adjacent triangles.

Triangle mesh examples: (i) for a non-square domain in 2D plane, (ii) describing a surface in 3D

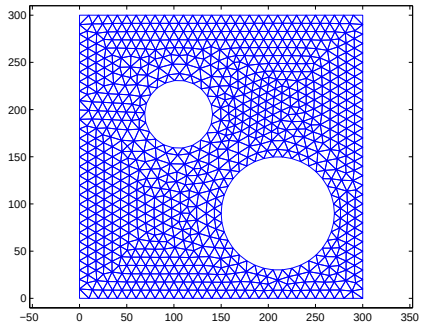


FIGURE: Triangle mesh in the plane. This one is MATLAB's data set `trimesh2d`.

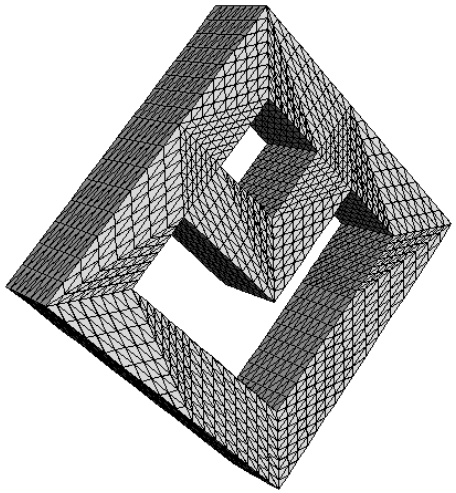


FIGURE: Triangle surface mesh.

Radial basis function (RBF)

- Let $|\mathbf{z}|$ denote the ℓ_2-norm (length) of a point $\mathbf{z}$ in 2D or 3D.
- Given data $(\mathbf{x}_i, f_i),\ i = 0, 1, \ldots, n$, the values $\phi(|\mathbf{x} - \mathbf{x}_j|)$ of a radial basis function ϕ measure the distances from a point $\mathbf{x}$ to the data abscissae. Thus, $\phi(\mathbf{z})$ depends only on the radial distance $|\mathbf{z}|$.

- Simplest version: look for an interpolant

$$v(\mathbf{x}) = \sum_{j=0}^{n} w_j \phi(|\mathbf{x} - \mathbf{x}_j|),$$

where the weights w_j are determined by the interpolation conditions

$$f_i = v(\mathbf{x}_i) = \sum_{j=0}^{n} w_j \phi(|\mathbf{x}_i - \mathbf{x}_j|), \quad i = 0, 1, \ldots, n.$$

- Solve these $n+1$ linear equations for the $n+1$ unknown weights w_j once. Then, construct the interpolant for any given $\mathbf{x}$ using the formula for $v(\mathbf{x})$.

Radial basis function (RBF)

- Let $|\mathbf{z}|$ denote the ℓ_2-norm (length) of a point $\mathbf{z}$ in 2D or 3D.
- Given data $(\mathbf{x}_i, f_i),\ i = 0, 1, \ldots, n$, the values $\phi(|\mathbf{x} - \mathbf{x}_j|)$ of a radial basis function ϕ measure the distances from a point $\mathbf{x}$ to the data abscissae. Thus, $\phi(\mathbf{z})$ depends only on the radial distance $|\mathbf{z}|$.

- Simplest version: look for an interpolant

$$v(\mathbf{x}) = \sum_{j=0}^{n} w_j \phi(|\mathbf{x} - \mathbf{x}_j|),$$

where the weights w_j are determined by the interpolation conditions

$$f_i = v(\mathbf{x}_i) = \sum_{j=0}^{n} w_j \phi(|\mathbf{x}_i - \mathbf{x}_j|), \quad i = 0, 1, \ldots, n.$$

- Solve these $n+1$ linear equations for the $n+1$ unknown weights w_j once. Then, construct the interpolant for any given $\mathbf{x}$ using the formula for $v(\mathbf{x})$.

RBF choices

- Note that there is only one ϕ in

$$v(\mathbf{x}) = \sum_{j=0}^{n} w_j \phi(|\mathbf{x} - \mathbf{x}_j|).$$

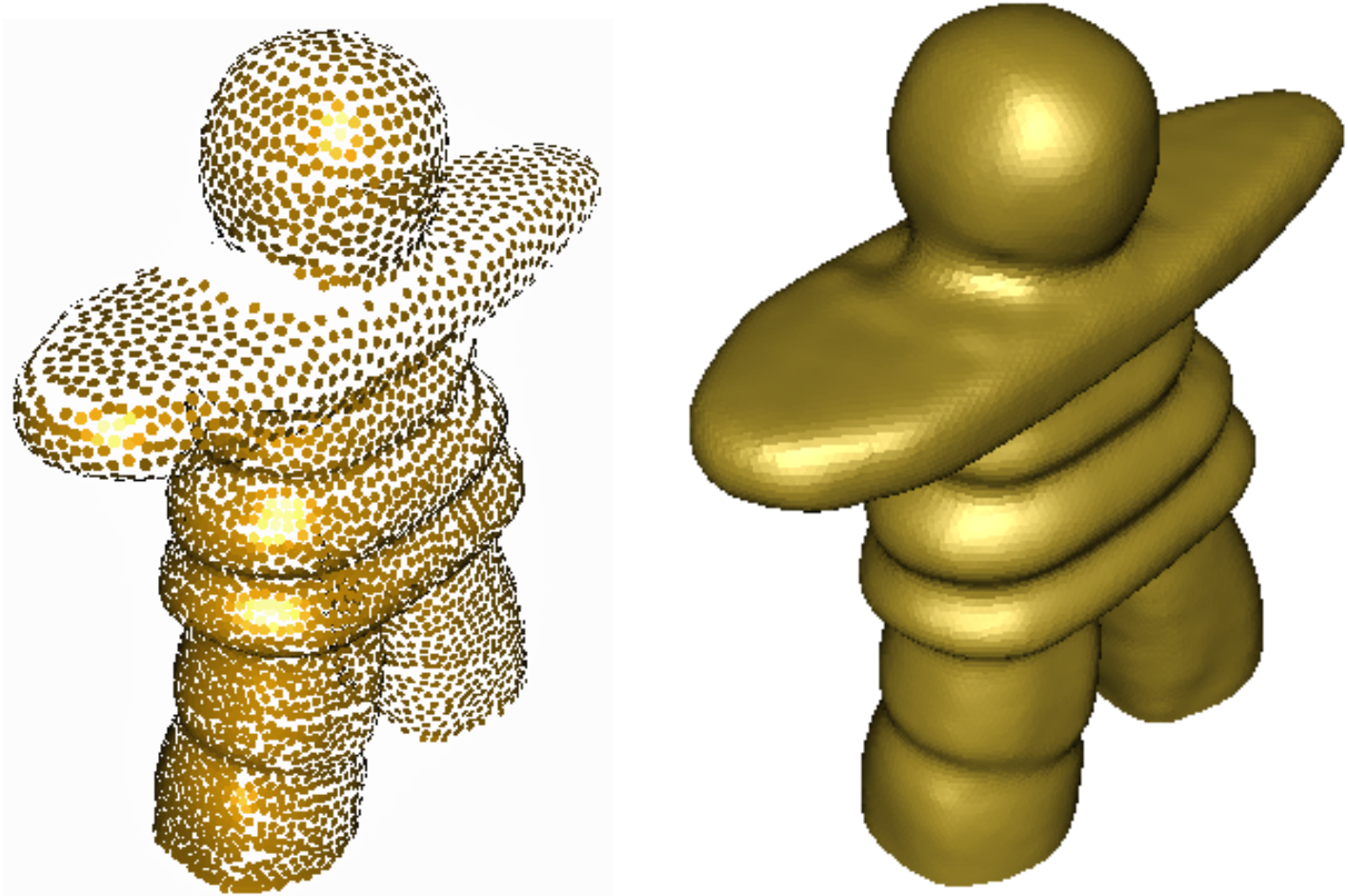
- A good choice for ϕ in 2D is the **multiquadric**

$$\phi(r) = \sqrt{r^2 + c^2},$$

  where c is a user-determined parameter.
- In 3D the **biharmonic spline**

$$\phi(r) = r$$

  plus an extra linear polynomial has been recommended.
- There are other choices in both 2D and 3D.



(a) consolidated point cloud (b) RBF surface

FIGURE: RBF interpolation of an upsampling of a consolidated point cloud.