

Piecewise Polynomial

Interpolation

A. DONEV, Spring 2021

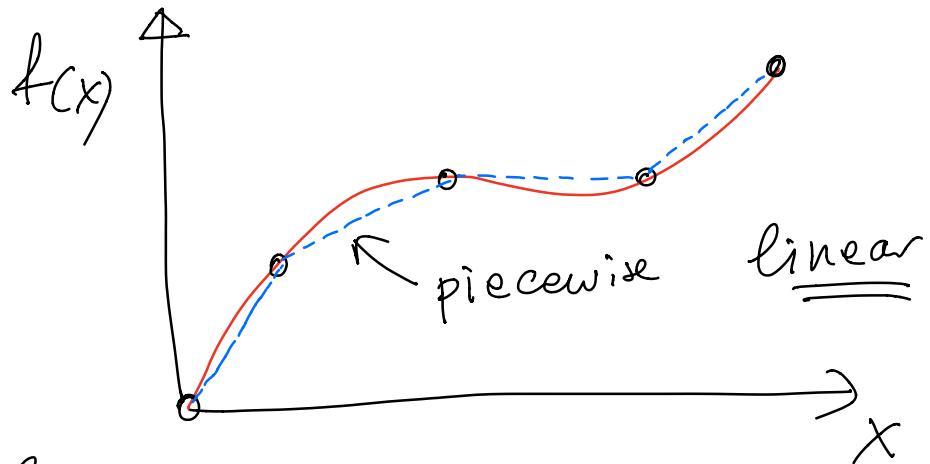
As we saw, polynomial interpolation on n equi-spaced nodes does not converge for large n .

While non-equispaced nodes can fix this, in many applications (e.g. engineering) we want to be able to use arbitrary grids.

Furthermore, we often need to work with non-smooth functions, for which large-degree polynomial approximants are inaccurate.

(1)

An alternative to global polynomial interpolation is to use piecewise (local) polynomial interpolation.



The key idea is very simple:

Break $[a, b]$ into disjoint sub-intervals. Eg. equal pieces

$$x_k = a + \frac{b-a}{n} \cdot k, \quad k=0, \dots, n+1$$

Approximate $f(x) \approx p_k(x)$ in each interval $[x_{i-1}, x_i]$ ②

with a low-order polynomial $P_k(x)$. Notice this is a different local polynomial in each interval.

E.g. piecewise linear

$$P_k(x) = f(x_{k-1}) \frac{x - x_k}{x_{k-1} - x_k} +$$

$$f(x_k) \frac{x - x_{k-1}}{x_k - x_{k-1}}$$

(Lagrange form of interpolant)

We can now piece together all of the $P_k(x)$ to form an approximation to $f(x)$ on $[a, b]$

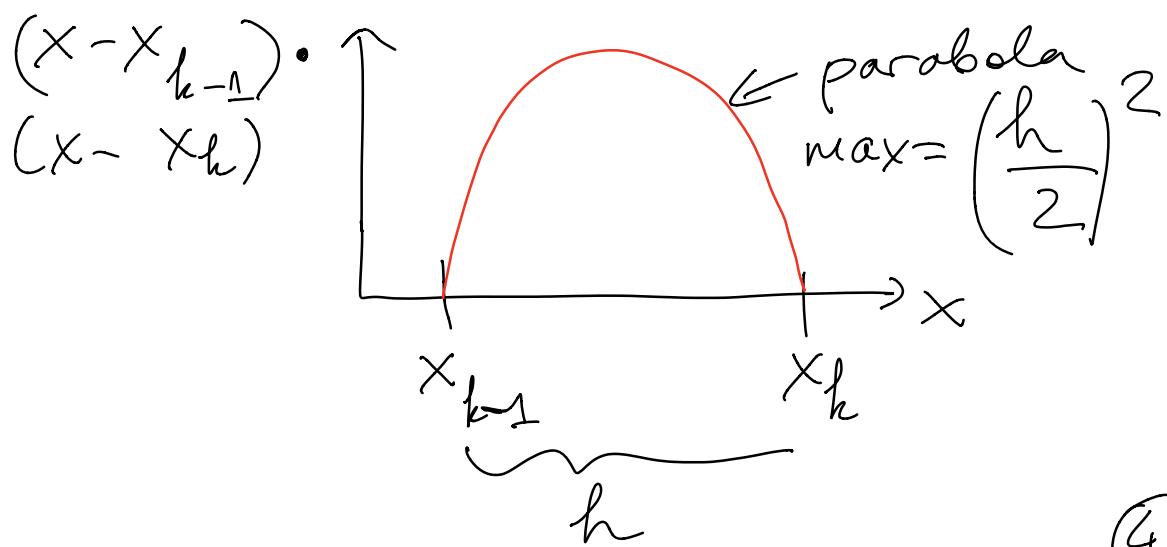
③

How accurate is this
piecewise linear approximation?

Since on $[x_{k-1}, x_k]$ the polynomial $p(x)$ is an interpolant, we can use the error estimate from last lecture:

$$f(x) - p_h(x) = \frac{f''(\xi(x))}{2} (x - x_{k-1})(x - x_k)$$

where $x, \xi(x) \in [x_{k-1}, x_k]$



(4)

where grid spacing

$$h = x_k - x_{k-1}$$

This gives us

$$|f(x) - P_h(x)| \leq \max_{x_{k-1} \leq x \leq x_k} |f''(x)| \frac{h^2}{8}$$

error $= O(h^2)$ "order of h^2 "

It means that if we reduce h by a factor of 2, the error reduces by a factor of $2^2 = 4$!

So we are guaranteed second-order convergence

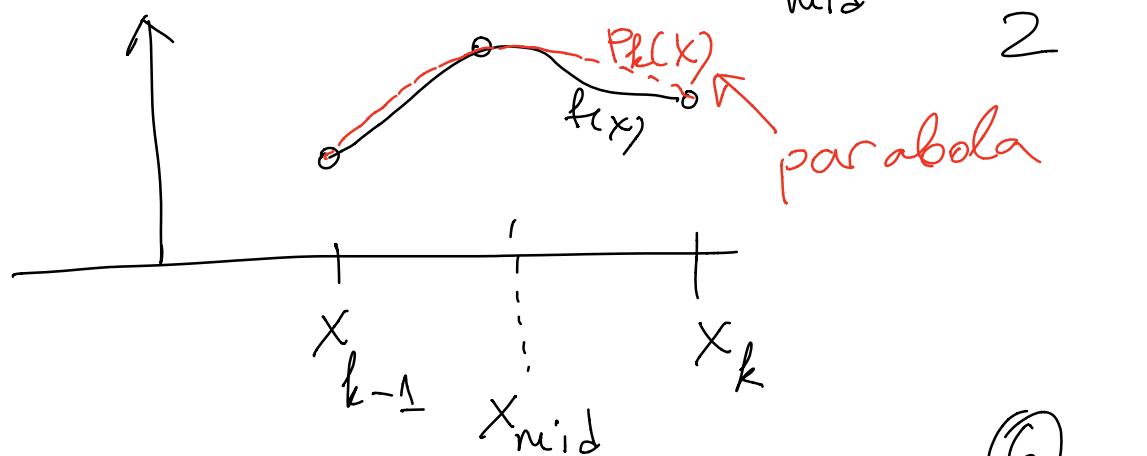
$$\left\| f - \underset{\infty}{\text{piecewise approx}} \right\| = O\left(\frac{1}{n^2}\right)$$

(5)

where n is the number of intervals / points used.

No Runge phenomenon any more; we get convergence \Rightarrow piecewise polynomial approximation is not very accurate but it is robust.

We can improve the order of accuracy by using a piecewise quadratic interpolant



⑥

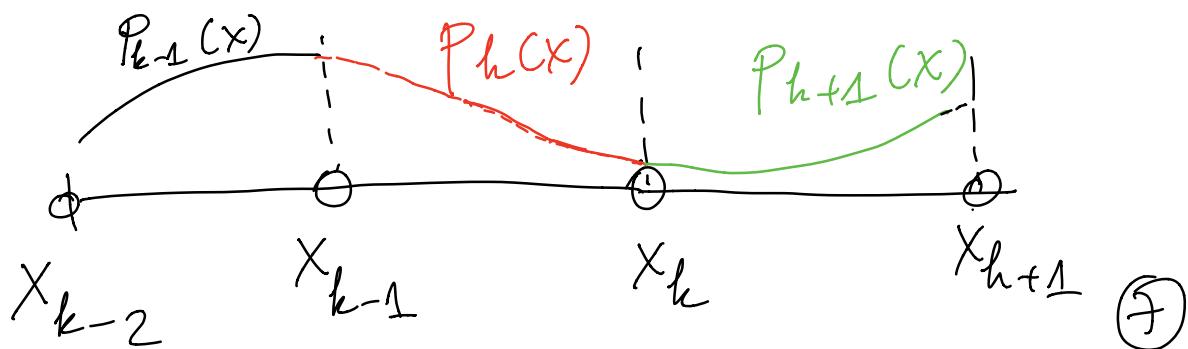
Now the error would be

$$|f(x) - P_h(x)| \sim |f'''(x)| h^3 \\ = O(h^3)$$

which is third-order accurate

Spline interpolants

Often we want to make sure that our approximant $P_h(x) \approx f(x)$ is smooth enough, for example, continuously differentiable.



We want to require:

$$P'_k(x_k) = P'_{k+1}(x_k) \dots (*)$$

and if we want a twice continuously-differentiable approximation then also

$$P''_k(x_k) = P''_{k+1}(x_k) \dots (**)$$

in addition to the usual interpolation condition

$$f(x_k) = P_k(x_k) = P_{k+1}(x_k) \dots (D)$$

Let's assume that in each interval we use a quadratic polynomial $P_h(x)$.

Can we satisfy both (D) and (*) ?

⑧

On each interval we have
3 unknown coefficients, so

$$\# \text{ unknowns} = 3^n$$

On each endpoint of an
interval we have

2 equations (\square)

(one at each endpoint)

and at each interior node

(*) adds one equation \Rightarrow

$$\# \text{ equations} = 2 \cdot n +$$

$$(n-1) = 3^n - 1$$

equations.

So if we add one more
equation we could solve for
 3^n unknowns

⑨

Such quadratic spline interpolation is rarely used. Instead, more common & popular is cubic spline interpolation.

Here $P_k(x)$ is cubic so there are 4 unknown coefficients in each sub-interval, or a total of $4n$ unknowns.

If we add also condition (***) in addition to (**) and (**), we get 1 more equation at each node, so

$$\# \text{ of unknowns} = 4n$$

$$\begin{aligned}\# \text{ of equations} &= 3n-1 + n-1 \\ &= 4n-2\end{aligned}$$

(10)

and if we add two more equations we could find a unique solution.

A common choice is to require

$$P_1''(x_0) = P_n''(x_n) = 0 \quad (\text{***})$$

for a natural cubic spline.

to get $4n$ unknowns &
 $4n$ equations.

Summary: unique!

The natural cubic spline
is the C^2 (twice cont. diff.)
piecewise cubic interpolant
that has (approximately)
minimal overall curvature.

(11)

How do we find the cubic spline interpolant?

Since $P_h(x)$ is cubic,

$P_h''(x)$ is linear in x .

Denote $\tau_k = P_k''(x_k)$. Then

$P_k''(x)$ must be equal to τ_k at nodes, and be linear in each sub-interval \Rightarrow
 $P_h''(x)$ is the piecewise linear interpolant.

$$P_k''(x) = \tau_{i-1} \frac{x - x_k}{x_{k-1} - x_k} + \tau_i \frac{x - x_{k-1}}{x_k - x_{k-1}}$$

in $[x_{k-1}, x_k]$

(12)

Simplify algebra (not required)
 by assuming equi-spaced nodes,
 and integrate twice to get

$$P_k(x) = \frac{1}{h} z_{k-1} \frac{(x_k - x)^3}{6} +$$

$$\frac{1}{h} z_k \frac{(x - x_{k-1})^3}{6}$$

$$+ C_k (x - x_{k-1}) + D_k$$

↑
 unknown coefficients /
 integration constants

But we have the conditions:

(D) at x_{k-1} :

$$\frac{1}{h} \frac{h^3}{6} z_{k-1} + D_k = f_{k-1} = f(x_{k-1})$$

(13)

$$\Rightarrow D_k = f_{k-1} - \frac{h^2}{6} z_{k-1}$$

(D) at x_k :

$$\frac{h^2}{6} z_k + C_k h + D_k = f_k$$

$$\Rightarrow C_k = \frac{1}{h} \left[(f_k - f_{k-1}) + \frac{h^2}{6} (z_{k-1} - z_k) \right]$$

Now we have a formula for P_k in terms of the unknown z_k 's. To determine z_k 's, we use (*) and (***) at each interior node, and (****) at the two end points ($z_0 = z_n = 0$)

to obtain a tridiagonal
linear system for the unknown's

$$\begin{bmatrix} 2/3 & 1/6 & & \\ 1/6 & \ddots & \ddots & \\ & \ddots & \ddots & 1/6 \\ & & 1/6 & 2/3 \end{bmatrix} \begin{bmatrix} t_1 \\ \vdots \\ t_{n-1} \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_{n-2} \end{bmatrix}$$

symmetric

$$b_k = \frac{1}{h^2} (f_{k+1} - 2f_k + f_{k-1})$$

In a worksheet / homework
you will learn that

$$b_k \approx f''(x_k)$$

which makes sense because
 $t_h \approx f''(x_h)$ is a second
derivative

(15)

Also in homework you obtained an $O(n)$ simple algorithm to do LU factorization of tridiagonal matrices. So we can compute the cubic spline fast even for millions of points (e.g. computer graphics) without a problem.

In Matlab, use spline to compute spline approximant, and fppval to evaluate it

Cubic splines approximate $f(x)$ to third order, $f'(x)$ to second order, and $f''(x)$ to first order.