Finite element methods in scientific computing

Wolfgang Bangerth, Colorado State University

Lecture 3.92:

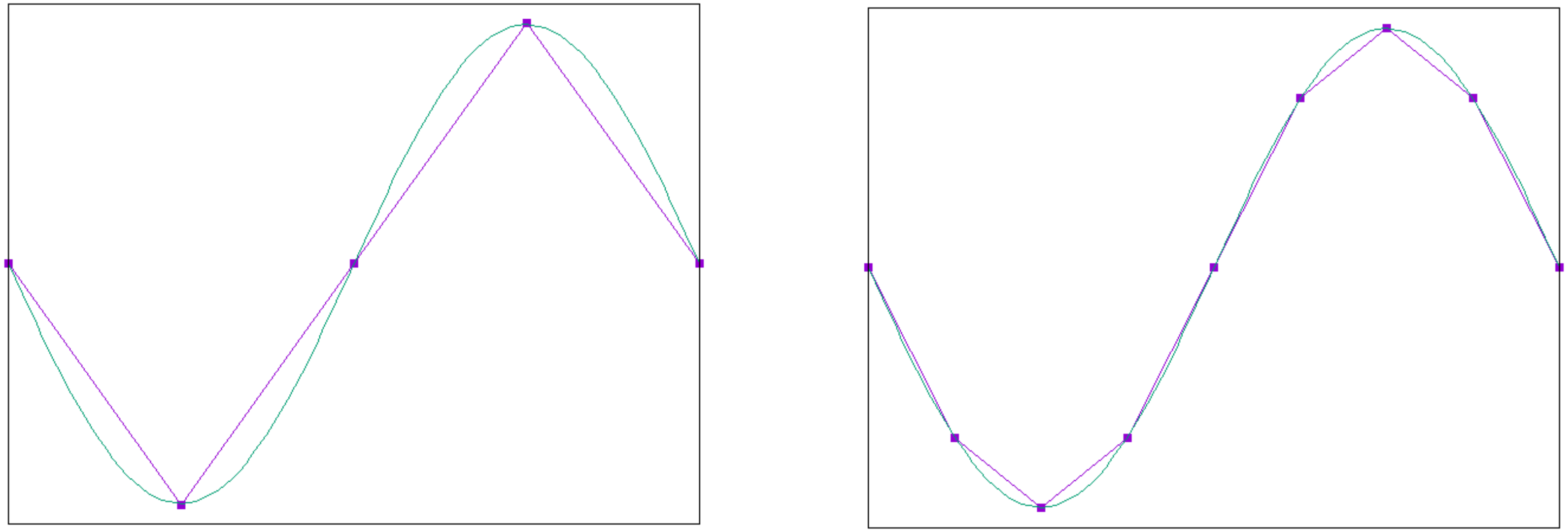
The ideas behind the finite element method

Part 3: Piecewise polynomial approximation in 2d/3d

Piecewise polynomial approximation in 1d

In 1d:

- Split the domain $\Omega \subset R$ of a function into intervals
- Define a piecewise function: linear on each interval, continuous at interval boundaries
- More sub-intervals → better approximation



Green: The function $f(x)$ we want to approximate.

Purple: The piecewise linear approximant $f_h(x)$.

Piecewise polynomial approximation in 2d

In 1d:

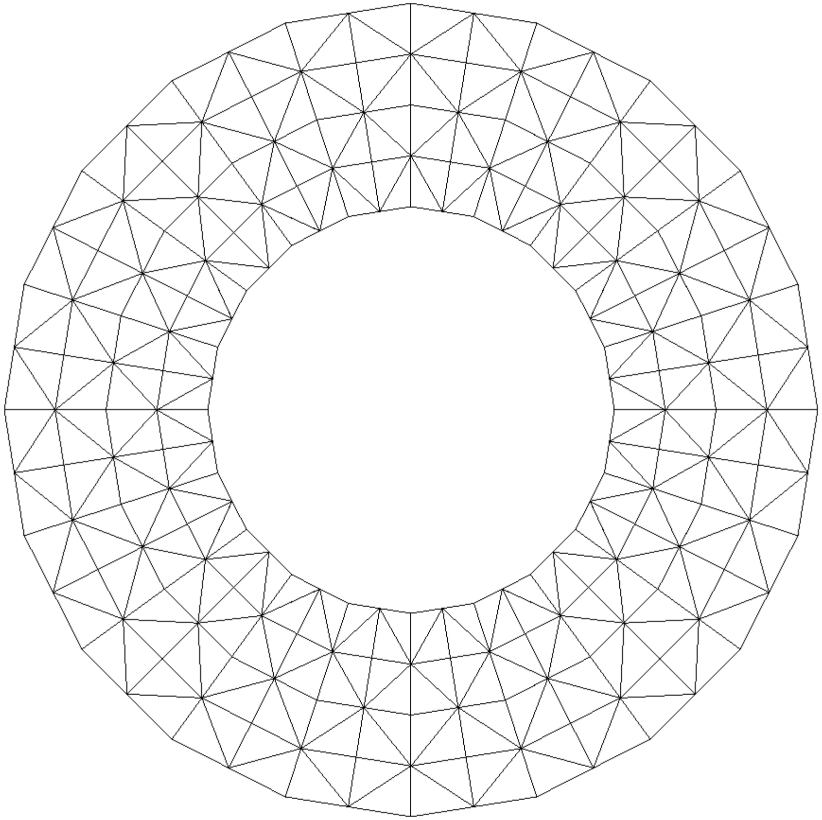
- Split the domain $\Omega \subset R$ of a function into intervals
- Define a piecewise function: linear on each interval, continuous at interval boundaries
- More sub-intervals → better approximation

In 2d:

- Split the domain $\Omega \subset R^2$ of a function into “cells”
- Define a piecewise function: linear on each cell, continuous at cell boundaries
- Smaller cells → better approximation

Piecewise polynomial approximation in 2d

Example with triangles:
Step 1: Subdivide the domain into triangular “cells”.

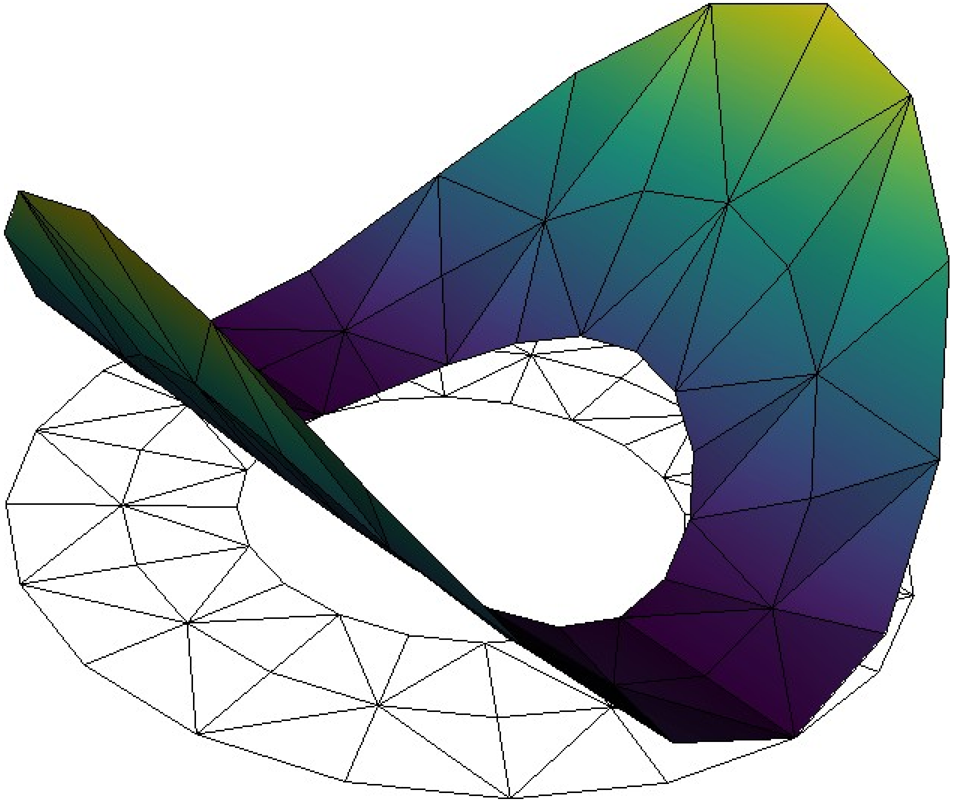


We call the collection of cells the “mesh”.

Piecewise polynomial approximation in 2d

Example with triangles:

Step 2: Represent functions as *piecewise polynomials*.

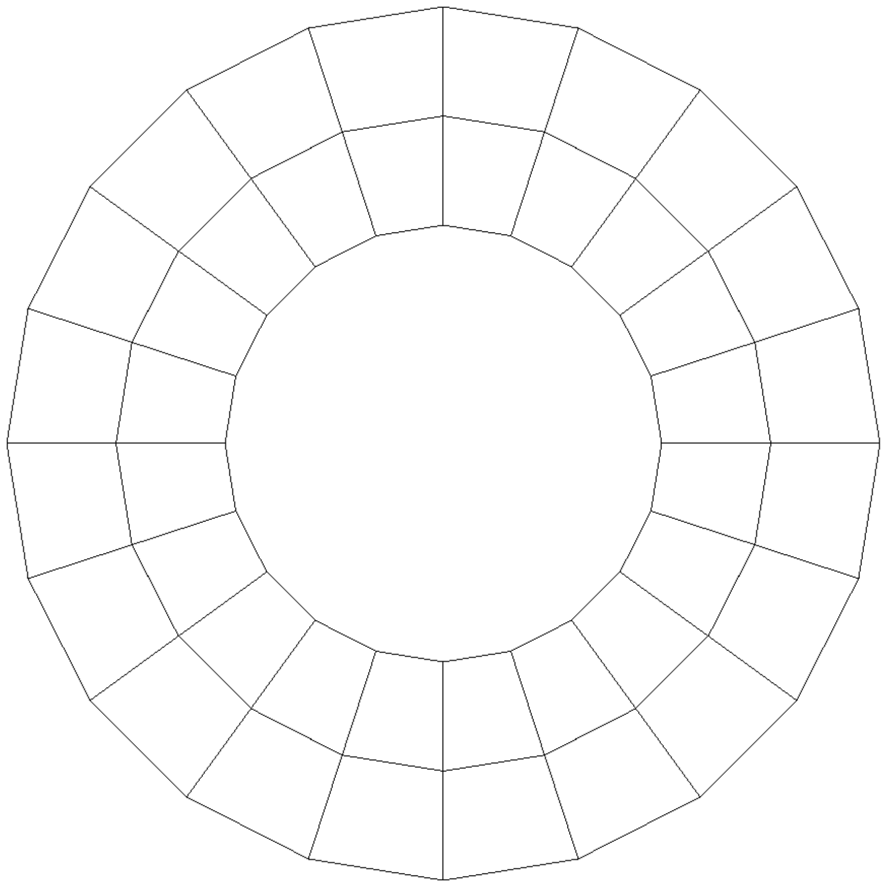


The function is linear on each triangle.

Piecewise polynomial approximation in 2d

Example with quadrilaterals:

Step 1: Subdivide the domain into *quadrilateral* “cells”.

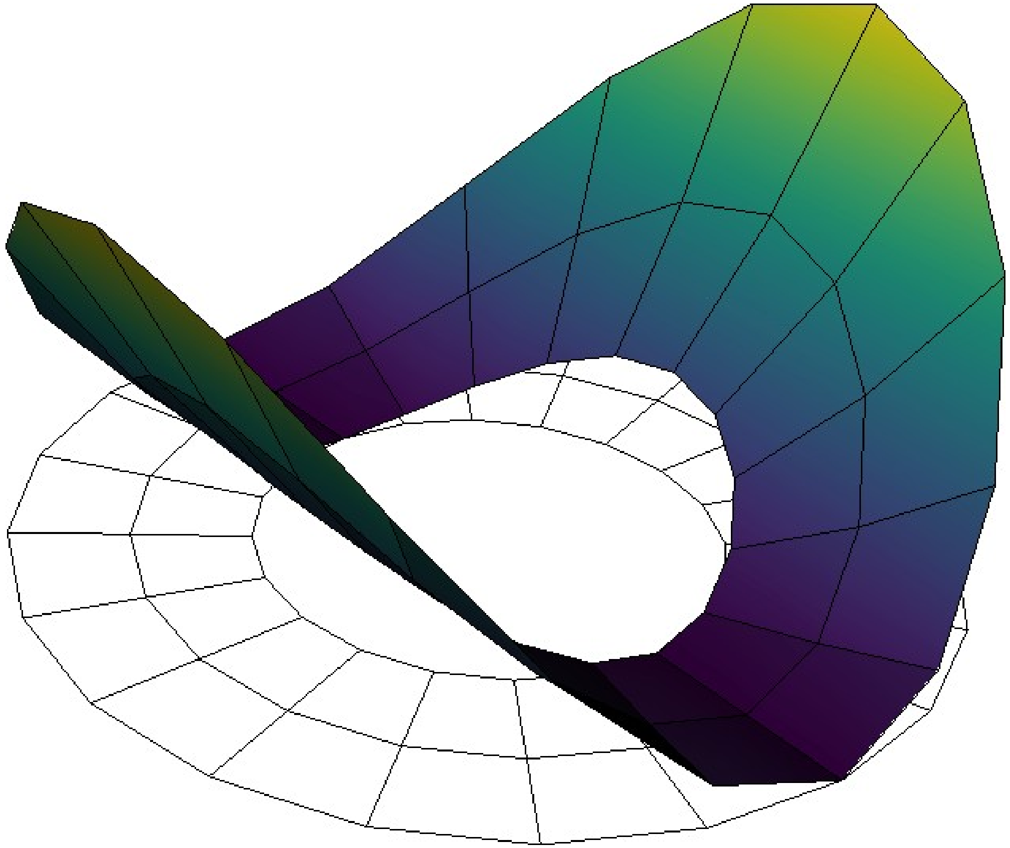


We call the collection of cells the “mesh”.

Piecewise polynomial approximation in 2d

Example with quadrilaterals:

Step 2: Represent functions as *piecewise polynomials*.



The function is bi-linear on each quadrilateral.

Piecewise polynomial approximation in 3d

In 2d:

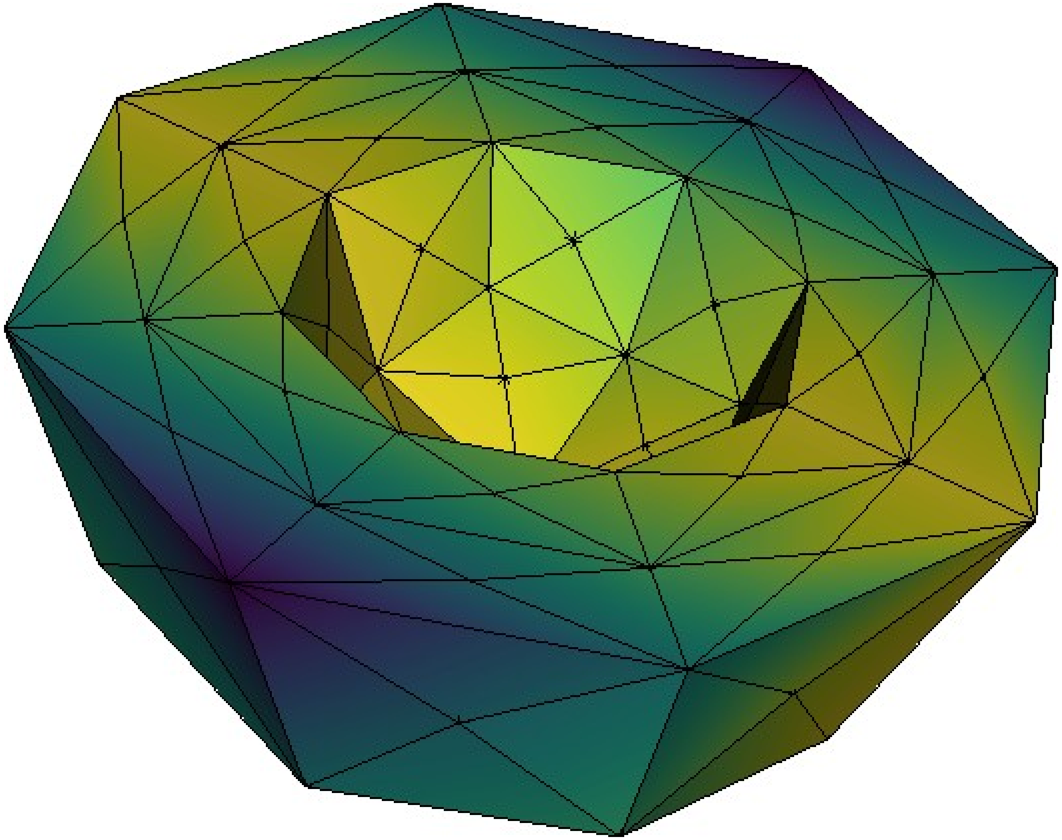
- Split the domain $\Omega \subset R^2$ of a function into "cells"
- Cells can be triangles or quadrilaterals
- Define a piecewise function: linear/polynomial on each cell, continuous at cell boundaries

In 3d:

- Split the domain $\Omega \subset R^3$ of a function into "cells"
- Cells can be tetrahedra, hexahedra, pyramids, prisms
- Define a piecewise function: linear/polynomial on each cell, continuous at cell boundaries

Piecewise polynomial approximation in 3d

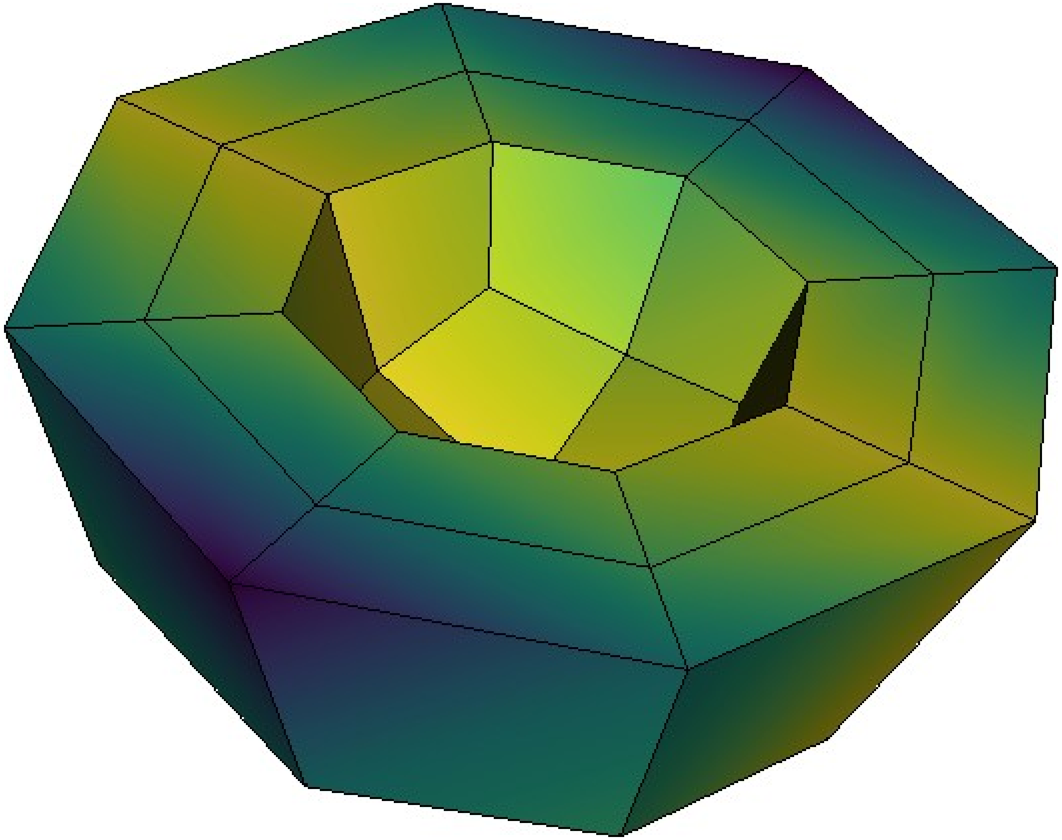
Example with tetrahedra:
Subdivide the domain into cells



We call the collection of cells the “mesh”.

Piecewise polynomial approximation in 3d

Example with hexahedra:
Subdivide the domain into cells



We call the collection of cells the “mesh”.

Piecewise polynomial approximation

"Higher order":

- The examples above used piecewise linear approximation
- Easily extendable to higher order:
  - piecewise quadratic
  - piecewise cubic
  - …
- On each interval/cell, the approximation is linear/quadratic/cubic/…

Notation: We say that the approximation is

- $P_1/P_2/...$: pw. linear/quadratic on triangles/tetrahedra
- $Q_1/Q_2/...$: pw. linear/quadratic on quads/hexahedra
- In 1d, $P_p=Q_p$

Piecewise polynomial approximation

Regardless of dimension and choice of cell:

Theorem:

- *f* is a function defined on a domain $\Omega \subset R^2$.
- $f_{h,p}$ is a piecewise function that interpolates *f*, defined on a mesh that subdivides Ω.
- h=the diameter of the largest cell of the mesh
- p=the polynomial degree used on the cells

Then:

$$\|f-f_{h,p}\| := \left(\int_\Omega |f(x)-f_{h,p}(x)|^2\right)^{1/2} \leq \frac{C_1(f,p,\Omega)}{p!} h^{p+1}$$

$$\|\nabla f-\nabla f_{h,p}\| := \left(\int_\Omega |\nabla f(x)-\nabla f_{h,p}(x)|^2\right)^{1/2} \leq \frac{C_2(f,p,\Omega)}{p!} h^{p}$$

Piecewise polynomial approximation

A note on terminology:

- We split the domain $\Omega\subset R^d$ of a function into "cells"
- In 1d: Cells are intervals
- In 2d: Cells can be triangles or quadrilaterals
- In 3d: Cells can be tetrahedra, hexahedra, pyramids, prisms

- The collection of cells is called the "mesh"

- We also use the term "triangulation"
  – even in 1d and 3d
  – even in 2d if we use quadrilaterals

Piecewise polynomial approximation

Summary:

- Subdivision of the domain into a mesh of cells + defining polynomial approximations on each cell generalizes the 1d construction

- As before, if we make cells small (*h* small), then the interpolation error decreases.

- Any function (within the class we care about) can be arbitrarily well approximated if we just put in enough computational work!

Finite element methods in scientific computing

Wolfgang Bangerth, Colorado State University