Finite element methods in scientific computing

Wolfgang Bangerth, Colorado State University

Lecture 3.9:

The ideas behind the finite element method

Part 1: Approximation

Two fundamental questions

Solving partial differential equations comes down to this:

Let's say we are given an equation such as

$$\begin{aligned} -\Delta u(x,y) &= f(x,y) && \text{in } \Omega \subset \mathbb{R}^2 \\ u(x,y) &= 0 && \text{on } \partial\Omega \end{aligned}$$

The solution is a **function *u(x,y)***.
To "know" *u(x,y)* means to know its value at *infinitely* many points *x,y*!

Two fundamental questions

The solution is a **function *u(x,y)***. To “know” *u(x,y)* means to know its value at *infinitely* many points *x,y*!

But:

– Computers can only store *finitely much data*
– Computations may only take *finitely many operations*

Consequence 1: In general, we can not solve PDEs *exactly*.

Consequence 2: The best we can do is *approximate* the solution somehow.

Two fundamental questions

Thus, “solving” PDEs comes down to the following two questions:

Question 1: What is a good way to *approximate* functions that requires only finitely much data/computation?

Question 2: How do we *find an approximation of the solution* of a PDE *without knowing the solution itself*?

Approximation

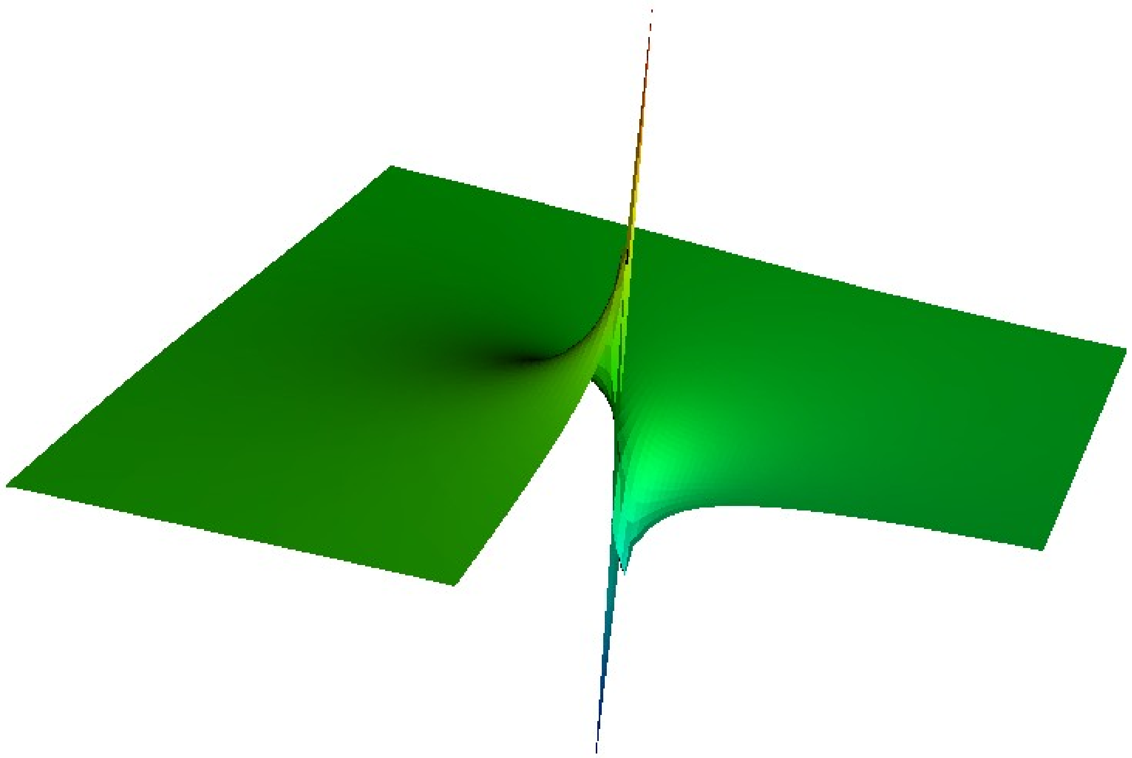
There are many ways to approximate functions:

1. (Finite) Fourier series
2. A global polynomial
3. Local (piecewise) polynomials defined on a subdivision (the "mesh") of the domain Ω
4. ...

For many good reasons, the finite element method uses option 3.

Why piecewise approximation?

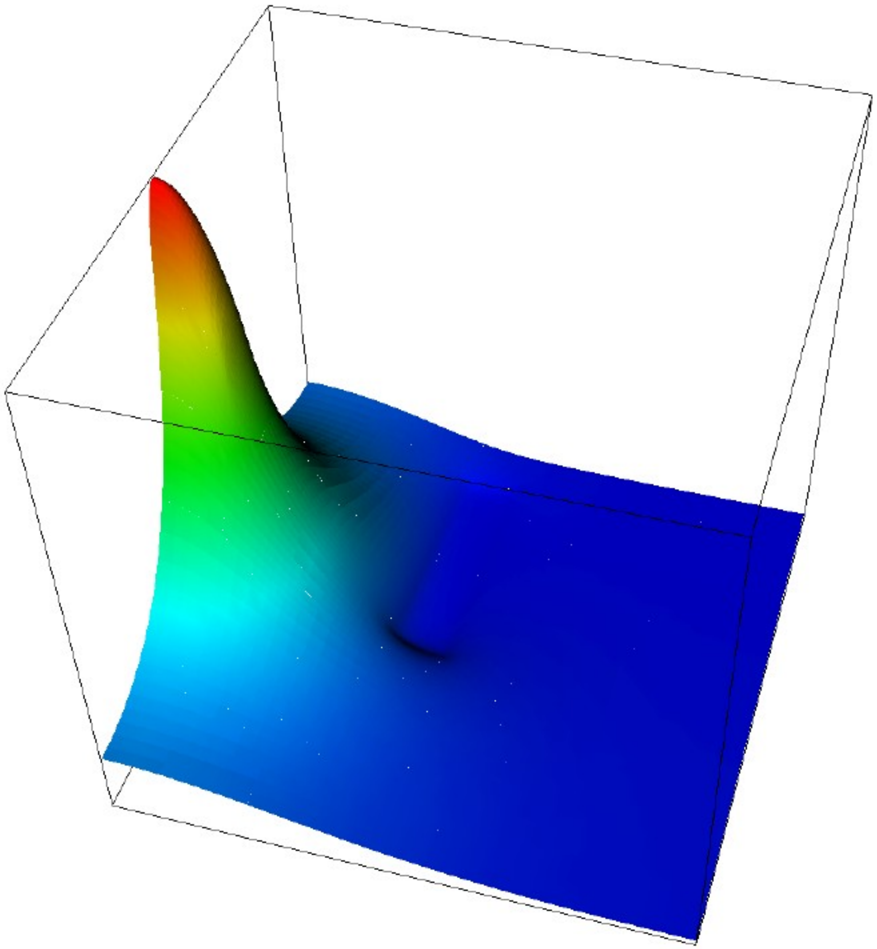
Here is a typical solution of a PDE:



(Pressure in Stokes flow in a domain with a corner.)

Why piecewise approximation?

Here is a typical solution of a PDE:



(Neutron flux density in the presence of an absorber.)

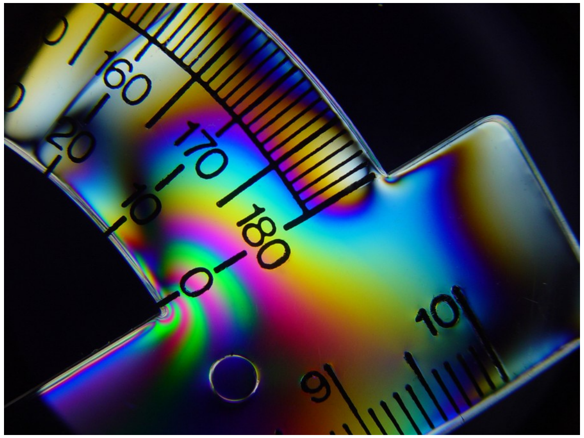
Why piecewise approximation?

Commonly found features of solutions of PDEs:

- Smooth in large parts of the domain
- Vary greatly in small parts of the domain
- May have kinks
- Singularities in corners

If you're an engineer, think about loads, displacements, and stresses:

- Displacements are continuous but not necessarily differentiable
- Stresses can have singularities



Source: Wikipedia

Why not Fourier approximation?

Non-smoothness and Fourier approximation:

Finite Fourier series of the form

$$u(x) \approx \sum_{k=1}^{N} U_k \sin(k x)$$

are good approximations only if *u(x)* is globally smooth.

Why not Fourier approximation?

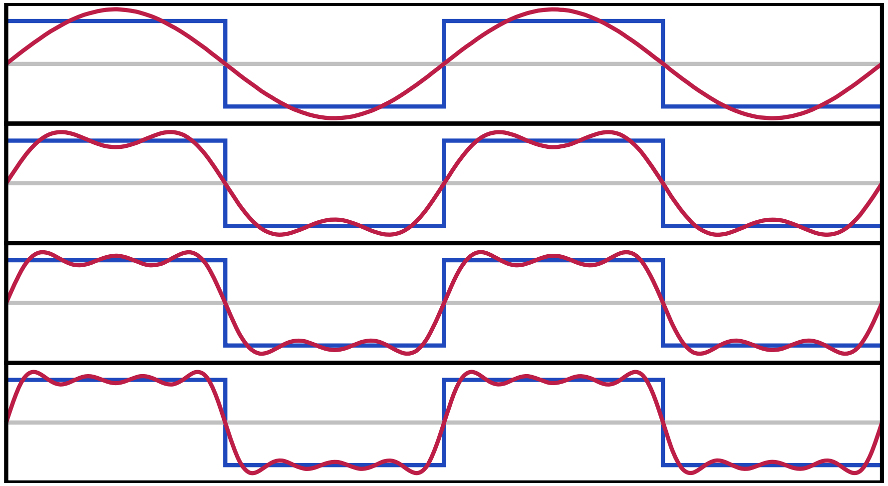
Non-smoothness and Fourier approximation:

Finite Fourier series of the form

$$u(x) \approx \sum_{k=1}^{N} U_k \sin(kx)$$

are good approximations only if *u(x)* is globally smooth.

Example:



Source: Wikipedia

Why not Taylor approximation?

Global polynomial approximation:

Finite polynomial series of the form

$$u(x) \approx \sum_{k=0}^{N} U_k (x-x_0)^k$$

are good approximations only in a neighborhood.

Why not Taylor approximation?

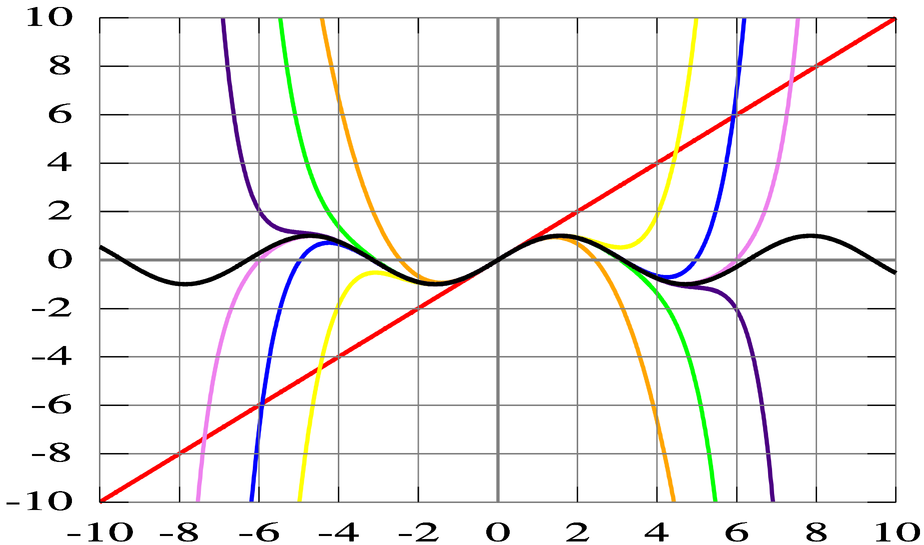
Global polynomial approximation:

Finite polynomial series of the form

$$u(x) \approx \sum_{k=0}^{N} U_k (x-x_0)^k$$

are good approximations only in a neighborhood.

Example:



Source: Wikipedia

Why not Taylor approximation?

Boundedness and polynomial approximation:

Finite polynomial series of the form

$$u(x) \approx \sum_{k=0}^{N} U_k (x-x_0)^k$$

are good approximations **only if *u(x)* is bounded**.

In fact (Weierstrass approximation theorem):

Every function *u(x)* that is bounded on an interval *(a,b)* can be approximated arbitrarily well by polynomials.

But: Functions with singularities are not bounded!

Why not Taylor approximation?

Non-smoothness and polynomial approximation:

Finite polynomial series of the form

$$u(x) \approx \sum_{k=0}^{N} U_k (x-x_0)^k$$

are good approximations **only if *u(x)* is globally smooth**.

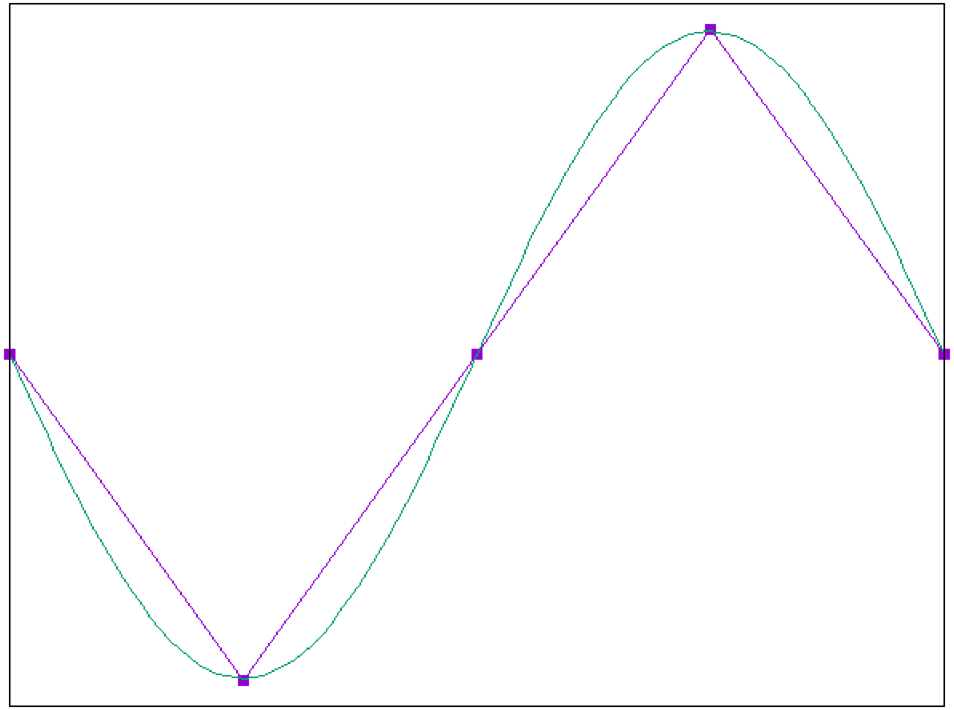
In fact:

In general, if *u(x)* or one of its derivatives have kinks, then polynomial approximations will be *globally* bad, not just where the kink is.

Why piecewise approximation?

Solution: Piecewise approximation!

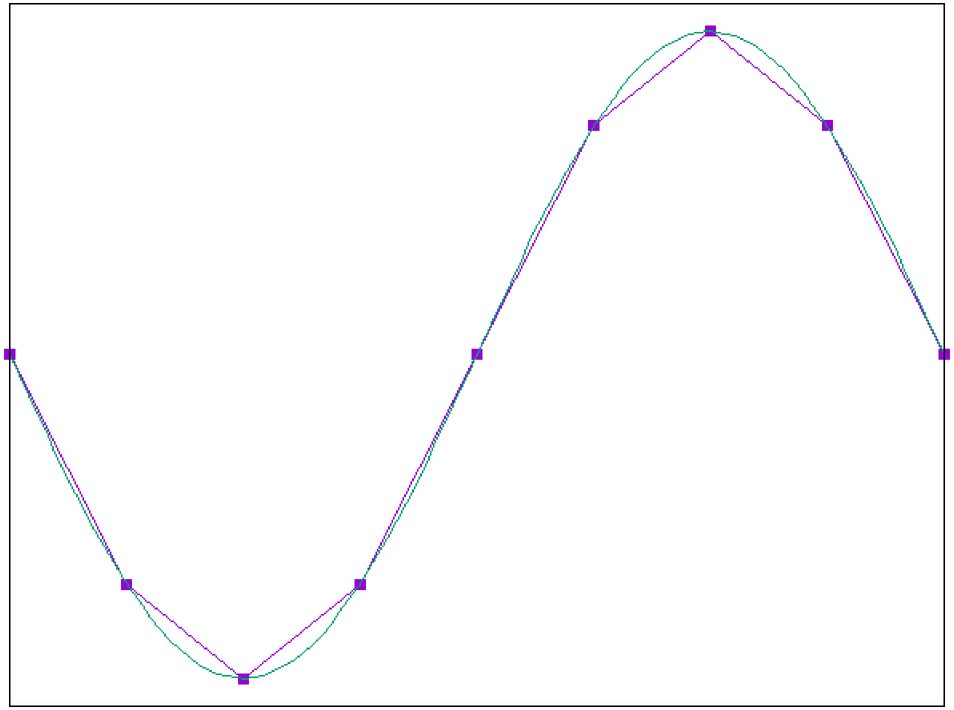
- Split the domain on which you want to approximate *u(x)* into small parts
- Approximate separately on each part



Why piecewise approximation?

Solution: Piecewise approximation!

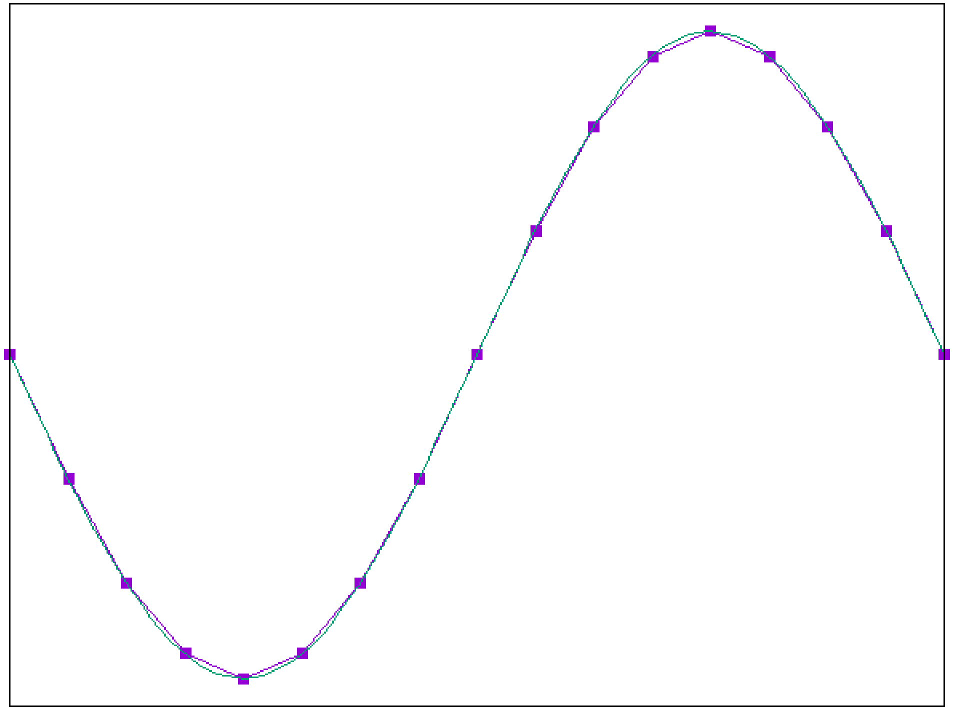
- Split the domain on which you want to approximate *u(x)* into small parts
- Approximate separately on each part



Why piecewise approximation?

Solution: Piecewise approximation!

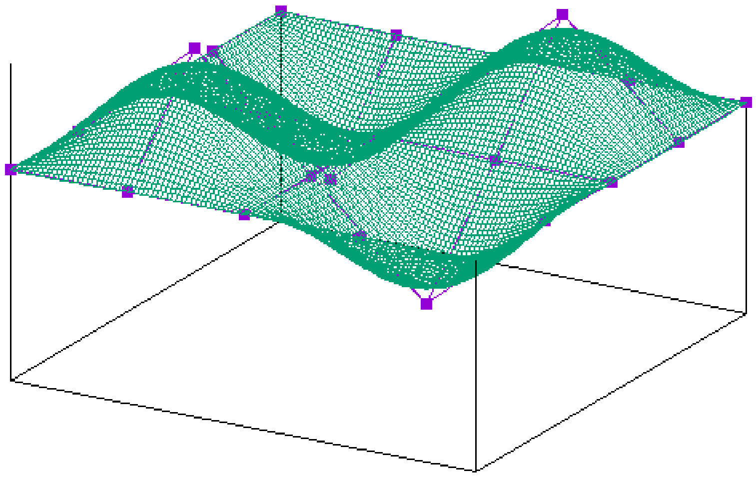
- Split the domain on which you want to approximate *u(x)* into small parts
- Approximate separately on each part



Why piecewise approximation?

Solution: Piecewise approximation!

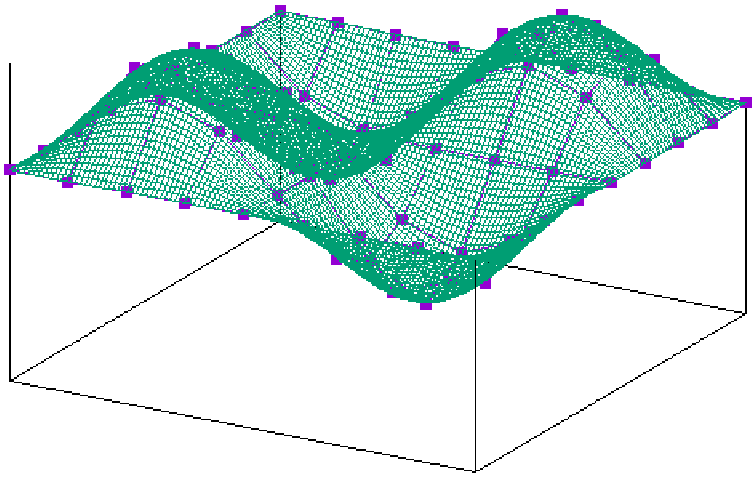
- Split the domain on which you want to approximate *u(x)* into small parts
- Approximate separately on each part



Why piecewise approximation?

Solution: Piecewise approximation!

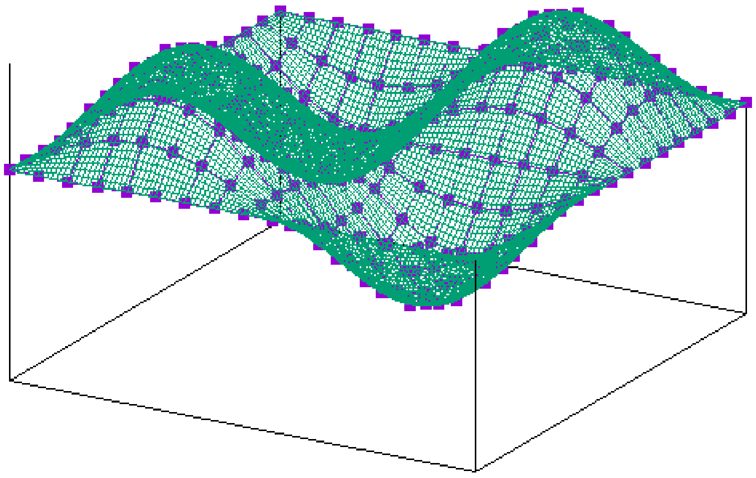
- Split the domain on which you want to approximate *u(x)* into small parts
- Approximate separately on each part



Why piecewise approximation?

Solution: Piecewise approximation!

- Split the domain on which you want to approximate *u(x)* into small parts
- Approximate separately on each part



Why piecewise approximation?

Solution: Piecewise approximation!

- Split the domain on which you want to approximate *u(x)* into small parts
- Approximate separately on each part

Advantages:

- We can use low-order approximations on each part:

Small intervals → good approximation.

Insensitive to singularities.

Easy and stable to parameterize, evaluate.

Why piecewise approximation?

Solution: Piecewise approximation!

- Split the domain on which you want to approximate *u(x)* into small parts
- Approximate separately on each part

Advantages:

- Resolving kinks and singularities can be done *locally:*

We can approximate the solution on one interval independently of the solution elsewhere.

Why piecewise approximation?

Solution: Piecewise approximation!

- Split the domain on which you want to approximate *u(x)* into small parts
- Approximate separately on each part

Advantages:

- We can increase the "resolution" where necessary:

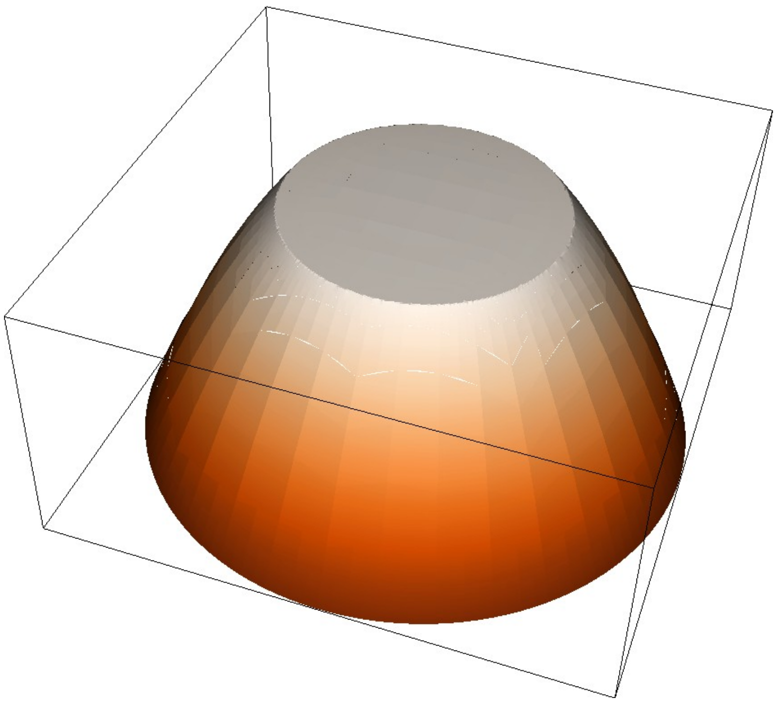
This is "(*h*-)adaptive mesh refinement" (AMR).

It is a way to make computations *cheaper.*

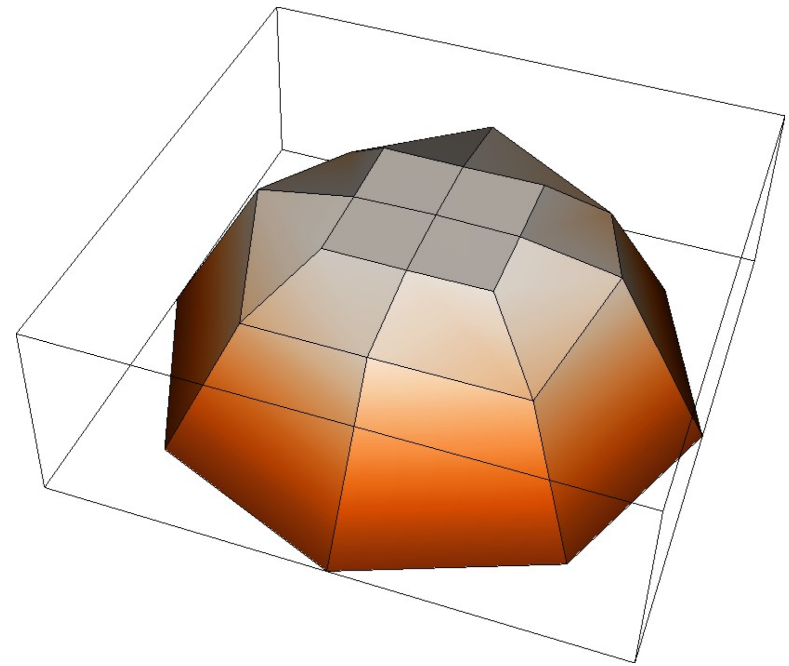
See lectures 15 and following.

Why piecewise approximation?

Step-6: An example in "adaptive mesh refinement":



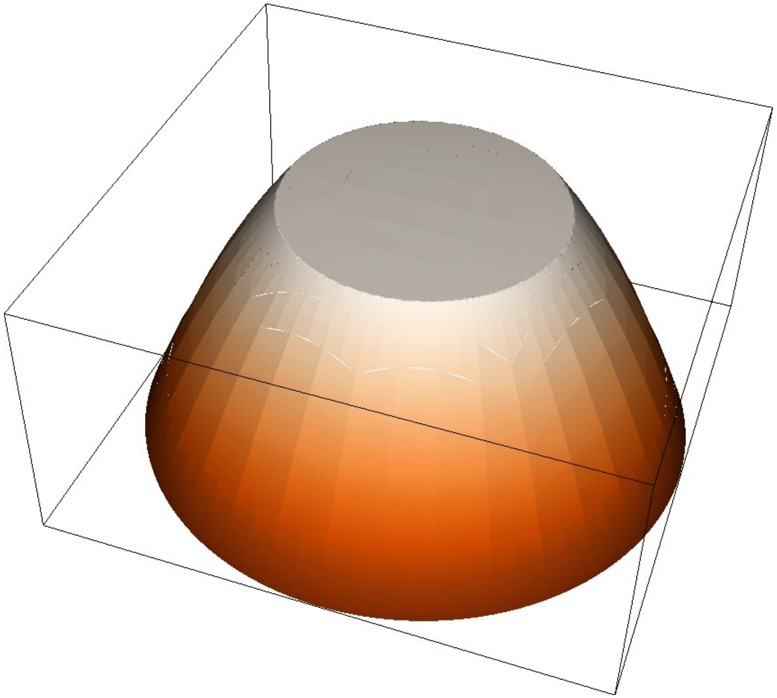
The "exact" solution.



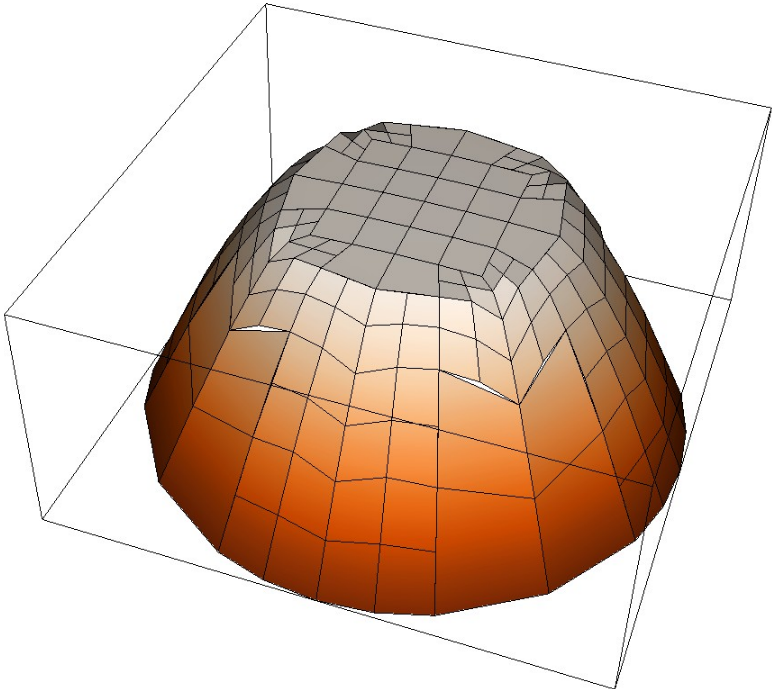
Approximation on a "coarse mesh".

Why piecewise approximation?

Step-6: An example in "adaptive mesh refinement":



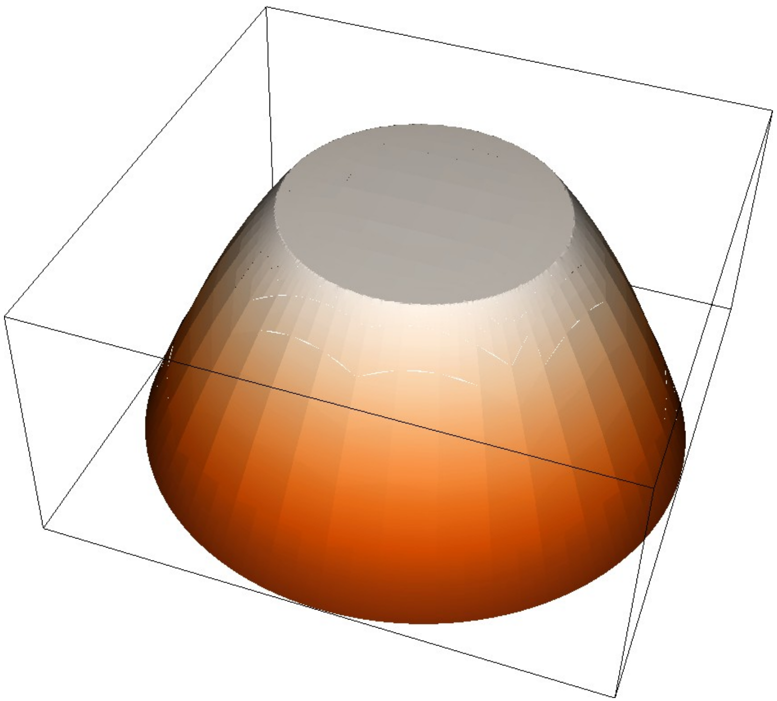
The "exact" solution.



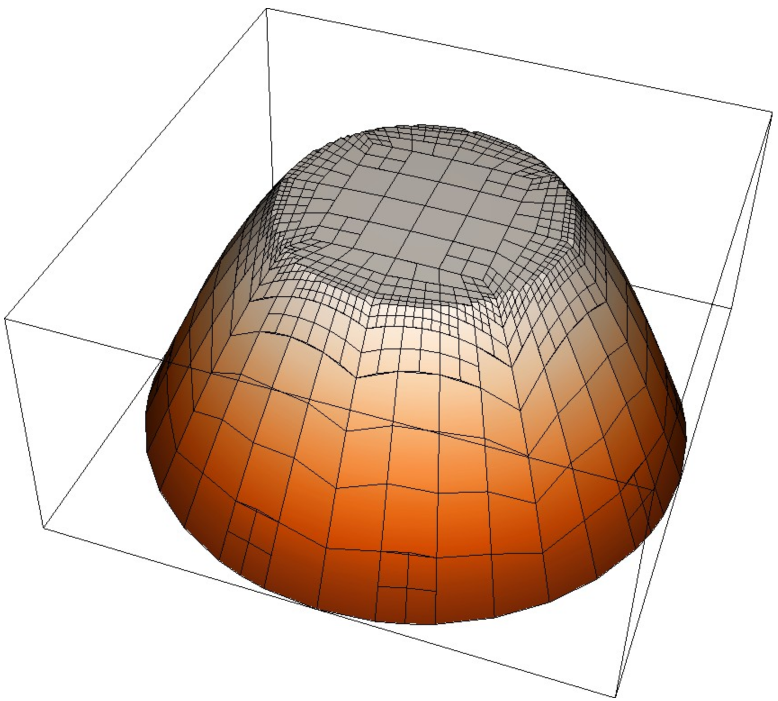
Approximation on a "somewhat refined" mesh.

Why piecewise approximation?

Step-6: An example in “adaptive mesh refinement”:



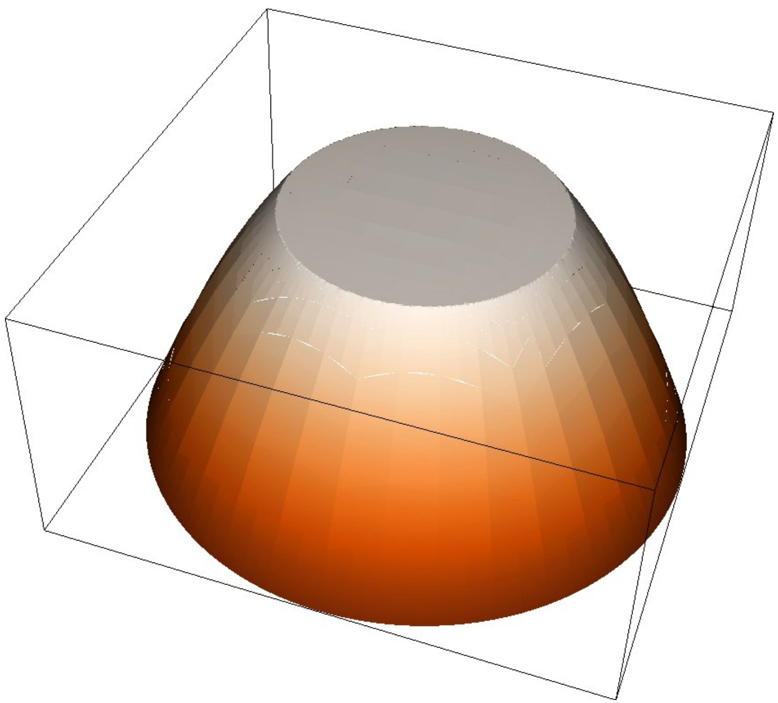
The “exact” solution.



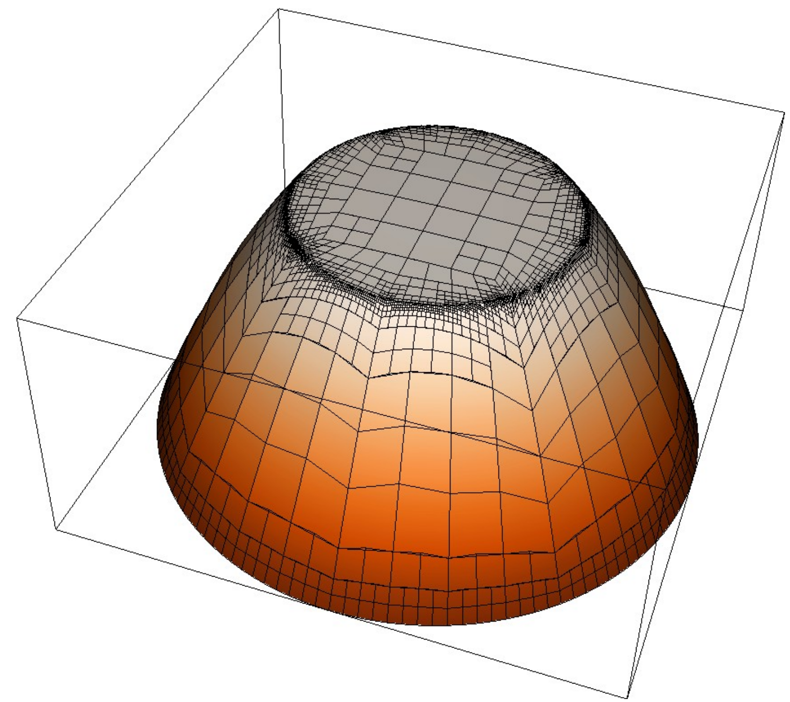
Approximation on a “decent mesh”.

Why piecewise approximation?

Step-6: An example in “adaptive mesh refinement”:



The “exact” solution.



Approximation on a “fine mesh”.

Why piecewise approximation?

Solution: Piecewise approximation!

- Split the domain on which you want to approximate *u(x)* into small parts
- Approximate separately on each part

Advantages:

- We can increase the polynomial degree where the solution is smooth:

This is "*p*-adaptive mesh refinement".

It is a way to make things *more accurate.*

Why piecewise approximation?

Take-away message:

- There are many ways to approximate functions
- For PDEs, the most appropriate way is:

Piecewise polynomial approximation on a subdivision (the "mesh") of the domain Ω.

Reasons:

- Accurate & stable
- Flexible: Can do *h*- and *p*-adaptive mesh refinement
- Relatively easy to represent in data structures

Finite element methods in scientific computing

Wolfgang Bangerth, Colorado State University