NUMERICAL METHODS FOR PARTIAL DIFFERENTIAL EQUATIONS

ACS II – FALL 08

Max Gunzburger

INTRODUCTION

Partial differential equations

- Partial differential equations (PDEs) are equations containing partial derivatives

  - common examples

Poisson equation
$$-\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} = f(x, y)$$

heat equation
$$\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} = f(t, x)$$

wave equation
$$\frac{\partial^2 u}{\partial t^2} - \frac{\partial^2 u}{\partial x^2} = f(t, x)$$

- Linear PDEs $\Rightarrow$ the unknowns (dependent variables) appear linearly
  - the above three examples are linear PDEs
  - no powers of u appear in the PDE
  - no products of u and its derivatives appear in the PDE
  - no functions of u or of its derivatives appear in the PDE

- Nonlinear PDE $\Rightarrow$ a PDE that is not linear

power of u

$$-\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} + u^{3/2} = f(x,y)$$

function of u

$$-\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} + \sin(u) = f(x,y)$$

product of u and its derivatives

$$\begin{cases} \dfrac{\partial u}{\partial t} - \dfrac{\partial^2 u}{\partial x^2} + u\dfrac{\partial u}{\partial x} = f(t,x) \\ \dfrac{\partial u}{\partial t} - \dfrac{\partial}{\partial x}\left(u^2 \dfrac{\partial u}{\partial x}\right) = f(t,x) \end{cases}$$

- PDE problems are posed on a spatial domain (denoted here by Ω) in an Euclidean space

  – usually in one, two, or three dimensions but there are important applications for which PDEs are posed in higher dimensions

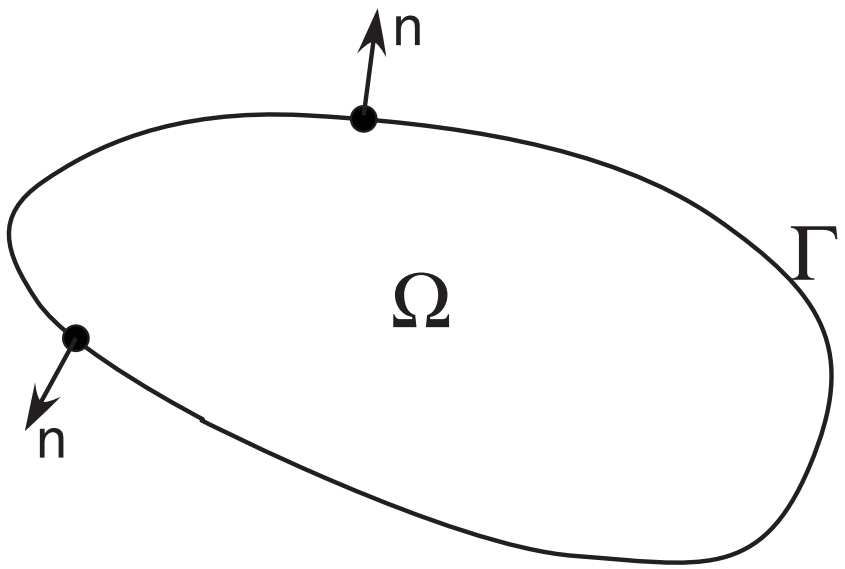
  – we denote by Γ the boundary of the domain Ω

    - the unit vector normal to the boundary and pointing away from Ω is denoted by $\mathbf{n}$

    - the normal derivative of a function u at a point on the boundary is denoted by

$$\frac{\partial u}{\partial n} = \mathbf{n} \cdot \nabla u = n_1 \frac{\partial u}{\partial x} + n_2 \frac{\partial u}{\partial y} + n_3 \frac{\partial u}{\partial z}$$

      where n_1, n_2, and n_3 respectively denote the x, y, and z components of $\mathbf{n}$



A bounded domain Ω, its boundary Γ, and the outward-pointing unit normal vector $\mathbf{n}$ at two points on the boundary

– domain nomenclature

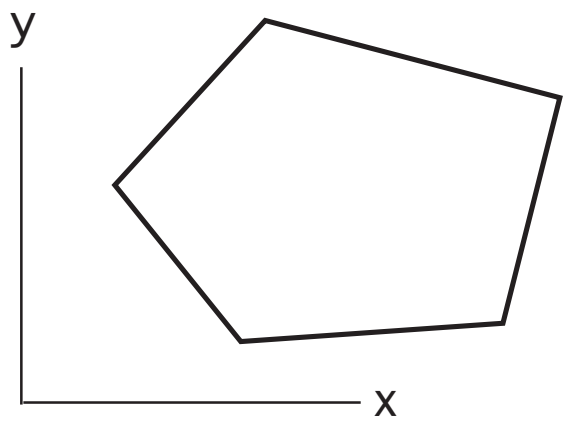
bounded domain $\Rightarrow$ can enclose Ω in a sphere of finite radius

unbounded domain $\Rightarrow$ a domain that is not a bounded domain
- a domain that has infinite extent in at least one direction

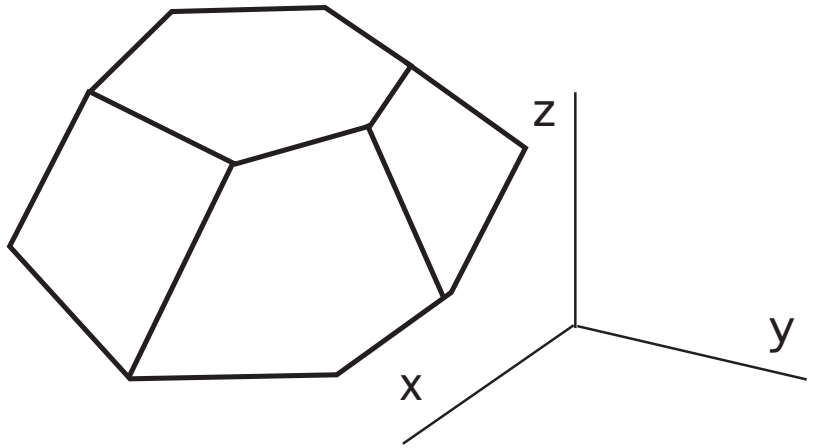
polygonal domain $\Rightarrow$ Ω has flat faces
- a polygon in 2D or a polytope in 3D

smooth domain $\Rightarrow$ the boundary Γ of Ω is (locally) the graph of a function having many derivatives
- in particular, Ω has no corners

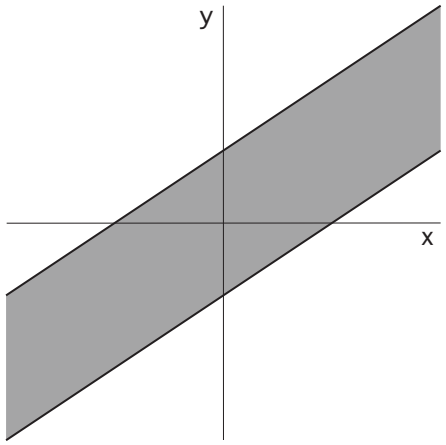
C^β domain $\Rightarrow$ the boundary Γ is (locally) the graph of a function having β derivatives



polygon in 2D



polyhedron in 3D



an unbounded domain

– Ω could be a manifold (surface) in Euclidean space

- a very important example is PDEs posed on the surface of a sphere,
  - e.g., the Earth

- PDE problems can additionally be posed over a time interval $[t_0, t_1]$, in which case we refer to them as time-dependent PDE problems

  - if the whole problem does not depend on time, we refer to it as a stationary or steady-state or time-independent problem

  – t_0 and t_1 are referred to as the initial and final times, respectively

  – most commonly, the interval is bounded, i.e., $-\infty < t_0 < t_1 < \infty$

    - there are problems of interest for which the interval $[t_0, t_1]$ is of infinite extent

  – without loss of generality, one usually chooses $t_0 = 0$ and $t_1 = T > 0$

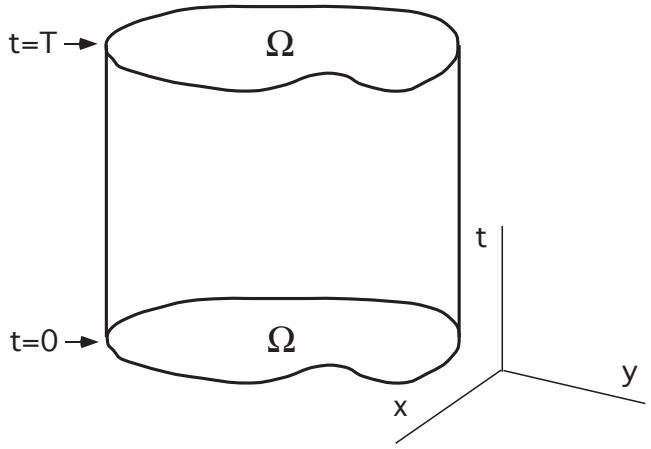
    - so that the time interval becomes $[0, T]$

  – in general, the spatial domain Ω could depend on time

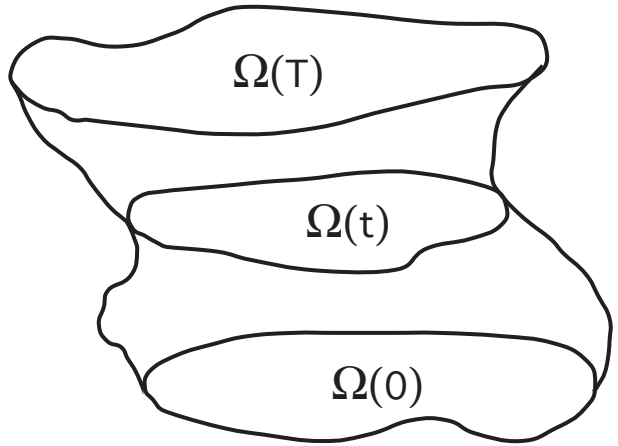
– if Ω does not depend on time, we say that the PDE is posed on a cylindrical space-time domain and denote that domain by $[0,T] \times \Omega$

- so we interpret $[0,T] \times \Omega$ to mean that the PDE problem is posed for

$$t \in [0,T] \qquad \text{and} \qquad \mathbf{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = (x,\, y,\, z)^T \in \Omega$$



space-time cylinder
spatial slices Ω are fixed in time



more general space time domain
spatial slices $\Omega(t)$ change in time

- PDE problems involve PDEs along with boundary conditions

  – these are conditions on the solution u imposed along the boundary Γ of the domain Ω

  – the most common boundary conditions are:

    for a given function $g_d(\mathbf{x}, t)$ or $g_n(\mathbf{x}, t)$, we require one of

    Dirichlet boundary condition $\Rightarrow$ $u(\mathbf{x}, t) = g_d(\mathbf{x}, t)$ for $\mathbf{x}$ on Γ

    Neumann boundary condition $\Rightarrow$ $\dfrac{\partial u}{\partial n}(\mathbf{x}, t) = g_n(\mathbf{x}, t)$ for $\mathbf{x}$ on Γ

  – it is possible, in fact, it is very common to impose the Dirichlet boundary condition on only part of the boundary and the Neumann boundary condition on another part

  – for some problems, boundary conditions are imposed on only part of the boundary and not on the whole boundary

    - so that on part of the boundary, no boundary condition is imposed

- If the problem is time dependent, PDE problems also involve initial conditions
  - these are conditions on the solution u imposed at $t = 0$
    - for a given function $u_0(\mathbf{x})$, we require

$$u(0, \mathbf{x}) = u_0(\mathbf{x}) \quad \text{for all } \mathbf{x} \in \Omega$$

  - much less commonly, there are problems for which one instead imposes a terminal condition at $t = T$

- If we only have a PDE and not the initial and boundary conditions, we cannot complete solve the PDE problem

- Boundary value problem $\Rightarrow$ PDE + boundary conditions

- Initial-boundary value problem $\Rightarrow$ PDE + boundary and initial conditions

- Example
  - consider the steady-state PDE (actually ODE)

$$-\frac{d^2u}{dx^2} = 2 \qquad \text{for } 0 < x < 1$$

so that we have $f(x) = 2$ and

$$\Omega = (0,1) \qquad \text{and} \qquad \Gamma = \{x = 0 \quad \text{and} \quad x = 1\}$$

  - the solution of the differential equation is easily obtained:

$$u(x) = -x^2 + ax + b \qquad \text{for } 0 < x < 1$$

for any constants a and b

  - we do not know what a and b are so that we have not completely solved the PDE problem

  - if we impose boundary conditions, we can determine a and b (sometimes)

if	$u(0) = 0$	and	$u(1) = 3$	$\Rightarrow$	$u(x) = -x^2 + 4x$	
if	$u(0) = 0$	and	$\frac{du}{dx}(1) = 3$	$\Rightarrow$	$u(x) = -x^2 + 5x$	
if	$\frac{du}{dx}(0) = 0$	and	$\frac{du}{dx}(1) = 3$	$\Rightarrow$	no solution exists	
if	$\frac{du}{dx}(0) = 0$			$\Rightarrow$	$u(x) = -x^2 + b$	for any b
if	$\frac{du}{dx}(0) = 5$	and	$\frac{du}{dx}(1) = 3$	$\Rightarrow$	$u(x) = -x^2 + 5x + b$	for any b

– in the first two cases, the boundary conditions help to completely determine $u(x)$

– in the third case, one cannot find any a and b such that the solution of the PDE satisfies both boundary conditions

– because in the third case we could not satisfy both boundary conditions, in the fourth case we threw one of them away

  - but now we have not yet completely determined $u(x)$

  - the constant b can be arbitrarily chosen

– in the fifth case, we kept both boundary conditions but changed the data (the right-hand sides) of the boundary conditions

  - now we can determine a solution

  - but it is not completely determined

    - the constant b can be arbitrarily chosen

– it turns out that for the PDE problem

$$-\frac{d^2u}{dx^2} = 2 \quad \text{in } (0,1)$$

- one can completely determine the solution if the Dirichlet boundary condition is specified at at least at one of the boundary points

- if the Neumann boundary condition is given at both boundary points, one can determine a solution if and only if the boundary data is such that

$$\frac{du}{dx}(0) - \frac{du}{dx}(1) = 2$$

but in this case, the solution is not completly determined

- such extra constraints on the data of a PDE problem are called compatibility conditions

- in order to completely determine the solution in the fourth case, an extra condition has to be given on the solution, e.g., if we also require u to satisfy

$$\int_0^1 u\,dx = 0$$

then we find that $b = 1/3$

- whenever we need extra constraints on the data to make sure a solution exists, we also need extra constraints on the solution in order to determine the solution completely

- if the Neumann boundary condition is given at both boundary points and the compatibility condition is not satisfied, then a solution cannot be determined

- To summarize, a PDE problem consists of
  - a domain in Euclidean space
  - a partial differential equation posed on that domain
  - boundary conditions posed on the boundary of that domain
  - initial conditions posed on that domain (if the problem is time dependent)
  - possibly compatibility conditions on the data
  - possibly constraints on the solution

- When a solution of a PDE problem can be determined, we say that a solution for the PDE problem exists

- When a solution of a PDE problem can be completely determined, we say that a solution for the PDE problem is unique

- When small changes to the data of a PDE problem result in small changes to the solution of a PDE problem, we say that the solution of the PDE problem depends continuously on the data

  – in the above example, a solution exists for the PDE problem

$$-\frac{d^2u}{dx^2} = 2 \text{ in } (0,1) \qquad \text{with} \qquad \frac{du}{dx}(0) = 5 \quad \text{and} \quad \frac{du}{dx}(1) = 3$$

  but not for the PDE problem

$$-\frac{d^2u}{dx^2} = 2 \text{ in } (0,1)$$

with

$$\frac{du}{dx}(0) = 5.000000000000000000000000000000000001 \quad \text{and} \quad \frac{du}{dx}(1) = 3$$

– clearly, the solution of the PDE problem

$$-\frac{d^2u}{dx^2} = 2 \text{ in } (0,1) \qquad \text{with} \qquad \frac{du}{dx}(0) = 5 \quad \text{and} \quad \frac{du}{dx}(1) = 3$$

do not depend continuously on the data

- a small change in the number 5 causes one to go from a problem for which a solutions exists to one for which a solution does not exist

– this is a very important example because it shows that roundoff error can change a problem from one for which a solution exists to one for which it does not

• Hadamard definition of a well-posed problem – a solution to the problem exists, is unique, and depends continuously on the data

- If such problematic behaviors can arise for the simple ODE example for which we can determine the exact solution, how can one know if they will occur for more complicated problems for which we do not know the exact solution so that we can only hope to use a computer to determine and approximate solution?

  - how does one know how many and what types of boundary and initial conditions one can impose?

  - how does one know if additional compatibility conditions on the data are needed?

  - how does one know if a solution exists?

  - how does one know if a solution is unique?

  - how does one know if a solution depends continuously on the data of the problem?

- Sometimes, physical intuition can answer some of these questions, but the best bet is to appeal to the theory of PDEs

- Knowing something about the properties of solutions of the PDEs is important to the design of numerical methods

  – existence of solutions – does the PDE problem have a solution?

    - if it doesn't, it is silly to seek an approximate solution on a computer

  – uniqueness of solutions – does the PDE problem have more than one solution?

    - often, PDE problems have multiple solutions so that one may wonder which solution one is approximating on a computer

  – stability of solutions – are solutions of the PDE problem very sensitive to small changes in data?

    - if so, it may be difficult to get accurate approximations

- regularity of solutions – how may derivatives does a solutions of a PDE problem have?
  - the number of derivatives affects the accuracy of numerical methods
  - if a solution has few derivatives, it is wasteful to use higher-order approximations
    - one might as well use cheaper low-order approximations

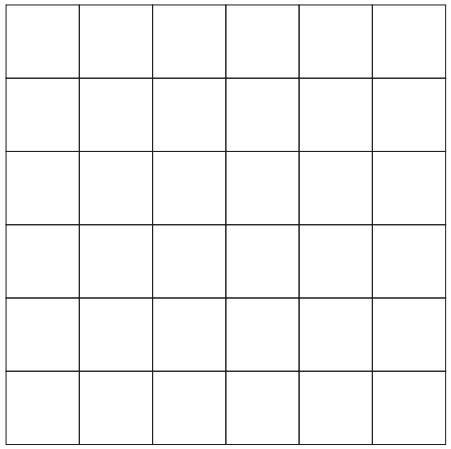
- Moral of the story: it is dangerous to try to use a computer to determine approximations of solutions of a PDE problem unless one knows something about the properties of those solutions

- Question: why do we need computers to solve PDEs?
  - even if you know a solution of a PDE problem exists, that does not mean that you can find it, i.e., that you can write down a formula for the solution
  - the best one can do is to find an approximate solution
  - the best and most general means for doing so is to use numerical methods

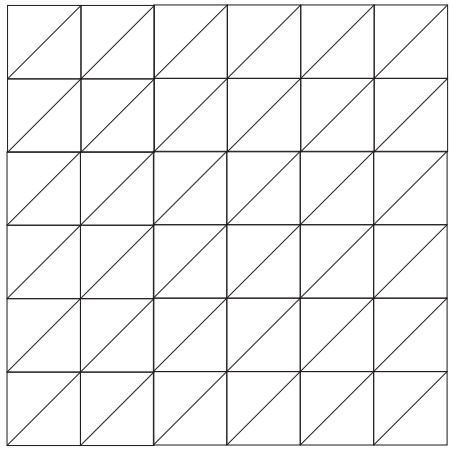
Discretization methods

turning PDE problems into problems you can put on a computer

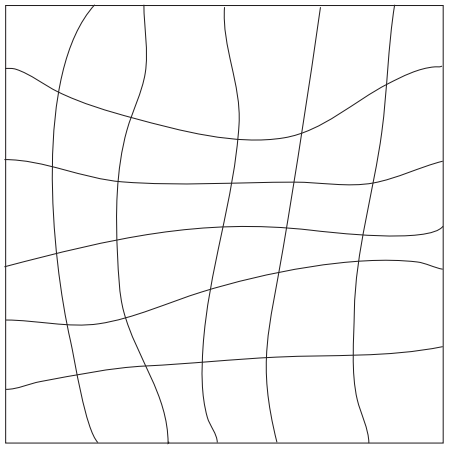
- Grids $\Rightarrow$ a sudivision of a domain into subdomains
  - covering property $\Rightarrow$ $\sum$ subdomains = domain $\Rightarrow$ no gaps
  - nonoverlapping property $\Rightarrow$ no two subdomains overlap
  - mesh = grid
  - sometimes it is said that one "triangulates a domain"
    - this means that one meshes a domain
      - used even when the subdivisions are not triangles
      - used even in dimensions higher than two



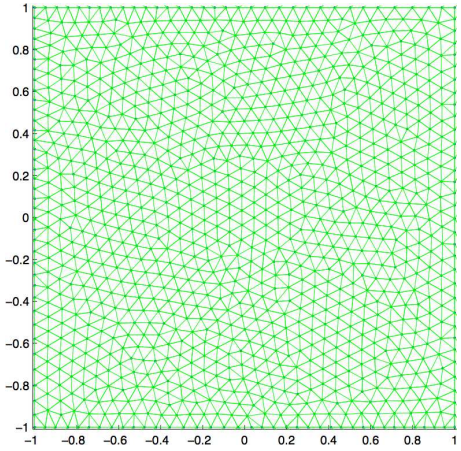
uniform rectangular



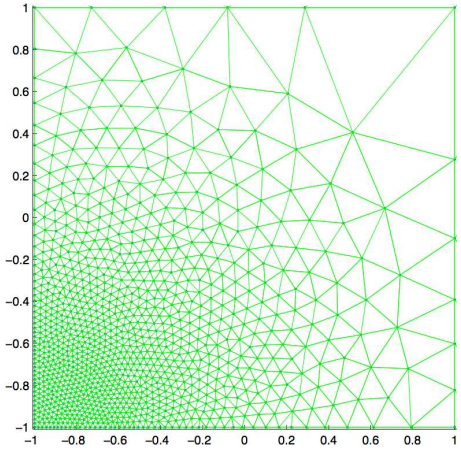
uniform triangular



nonuniform structured

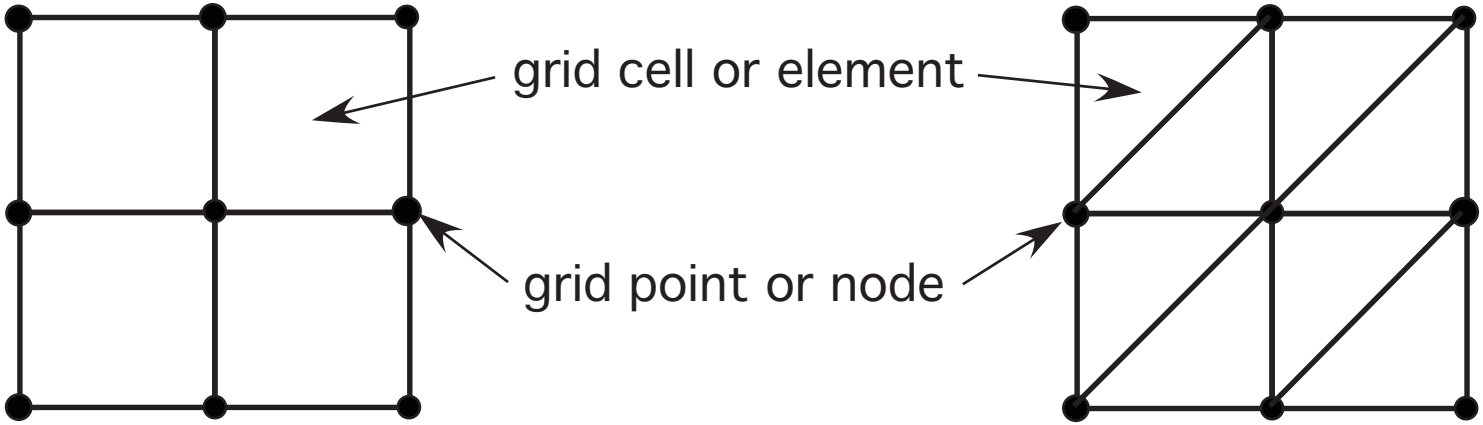


uniform unstructured

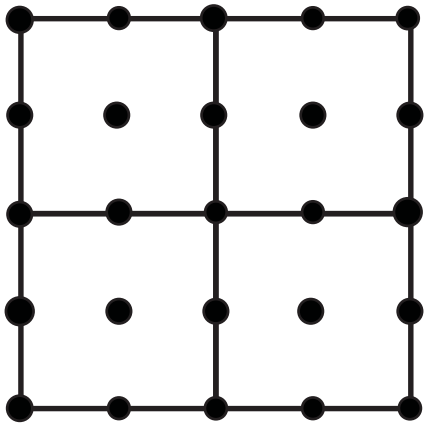


nonuniform unstructured

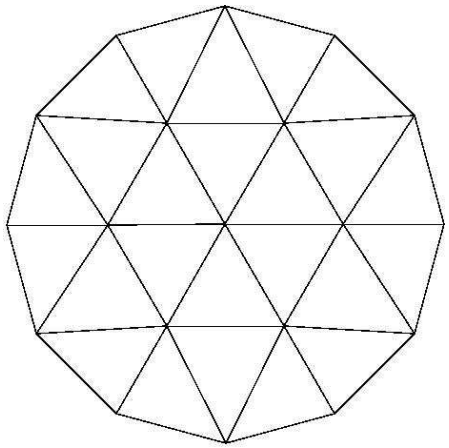
Grid nomenclature



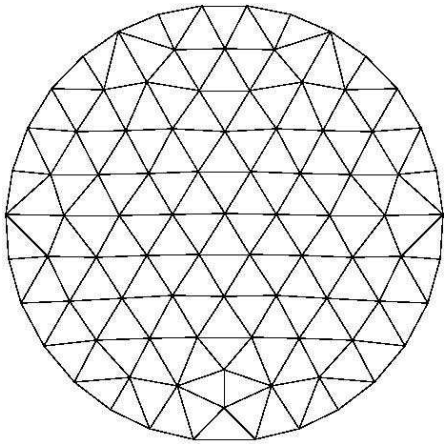
More grid nomenclature



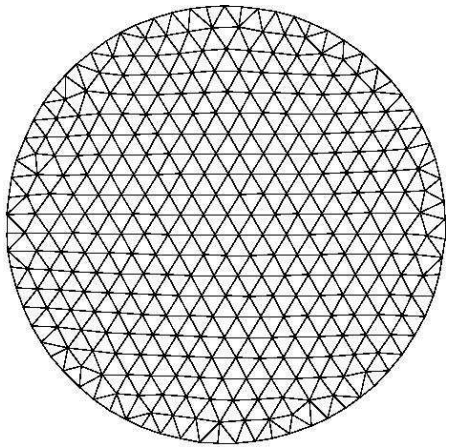
Nodes can be placed at points other than the vertices of the elements, e.g., at edge and interior points



coarse grid

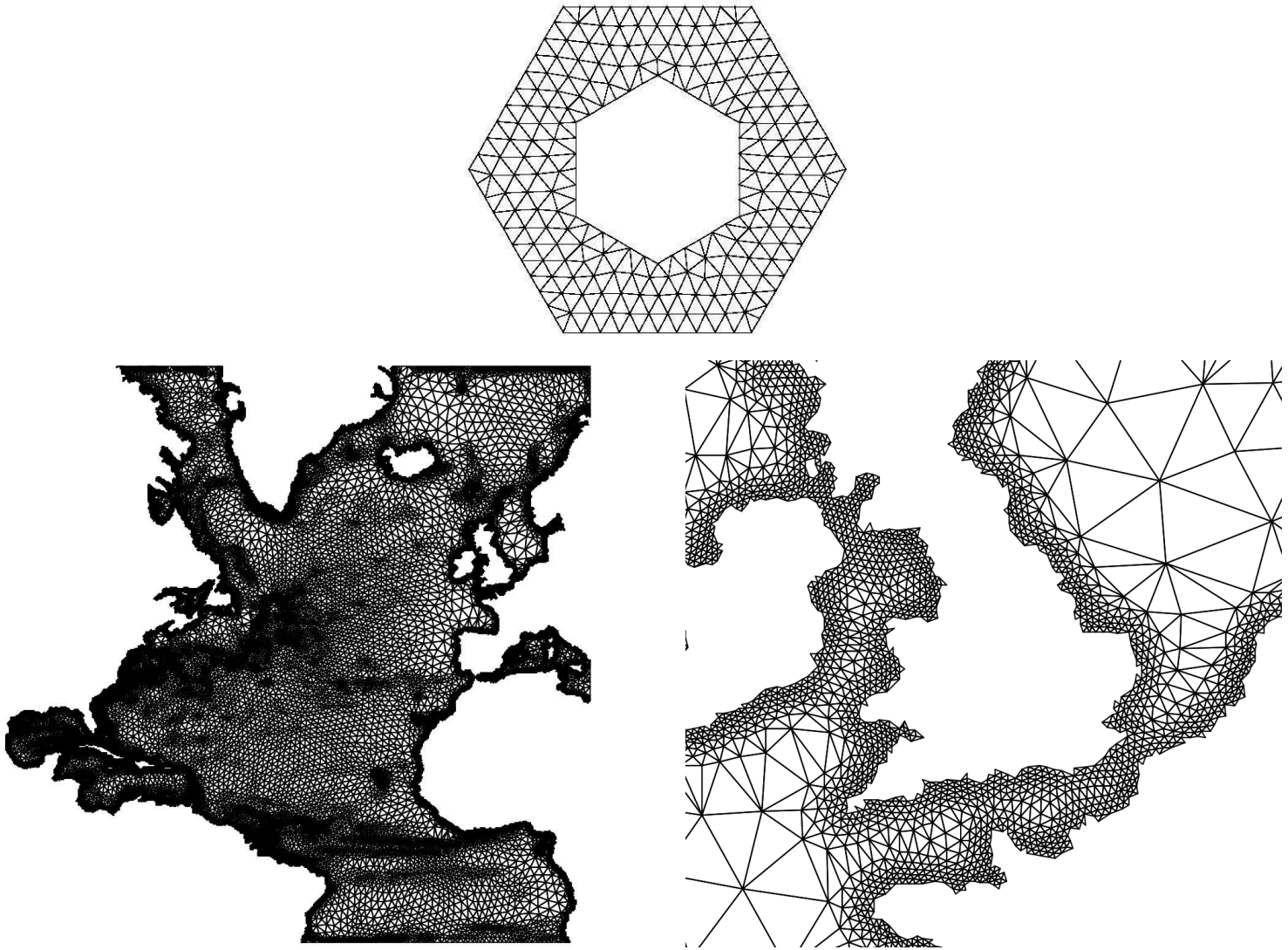


medium grid

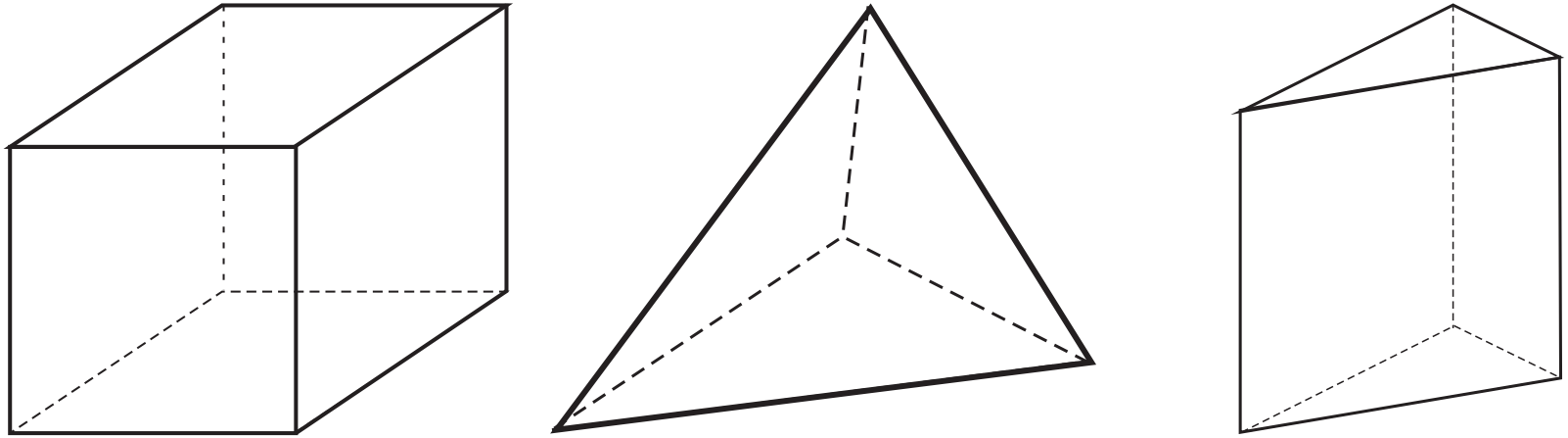


fine grid

More grid nomenclature



Grids on more general domains



hexahedral tetrahedral prismatic

Some possible grid cells in three dimensions

– a d-simplex is a polytope in d dimensions having $d+1$ vertices

1-simplex = interval 2-simplex = triangle

3-simplex = tetrahedron

- why are they important?

$\Rightarrow$ any polytope can be divided into simplices

– the grid size is a measure of the size of the grid

- for simple uniform Cartesian grids, we have the grid sizes h_x and h_y and let the grid size be defined by

$$h = \max\{h_x, h_y\}$$

- for nonuniform Cartesian grids, we define the grid size

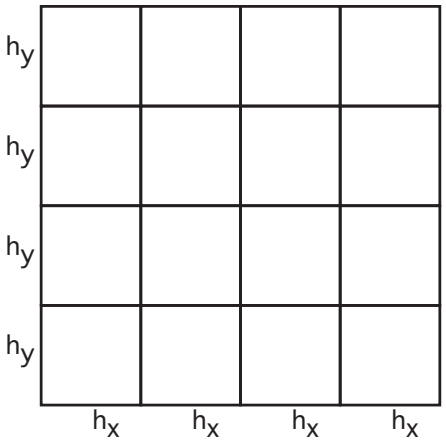
$$h = \max_{i,j}\{h_{x_i}, h_{y_j}\}$$

- for general grids, we define the grid size

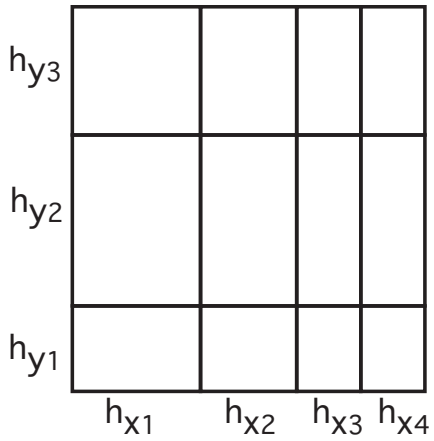
$$h = \text{maximum of all the cell diameters}$$

where the diameter of a cell is the diameter of the smallest sphere that encloses the cell

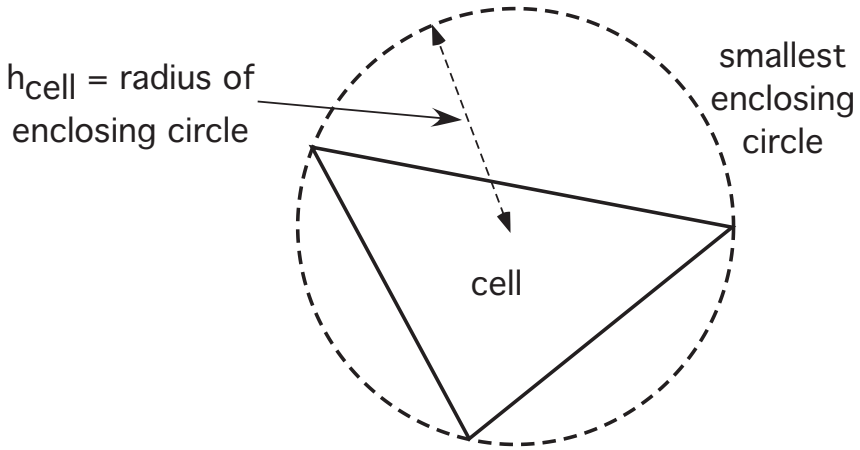
- instead of the cell diameter, the longest side of any cell is also used as a definition of h



uniform Cartesian grid



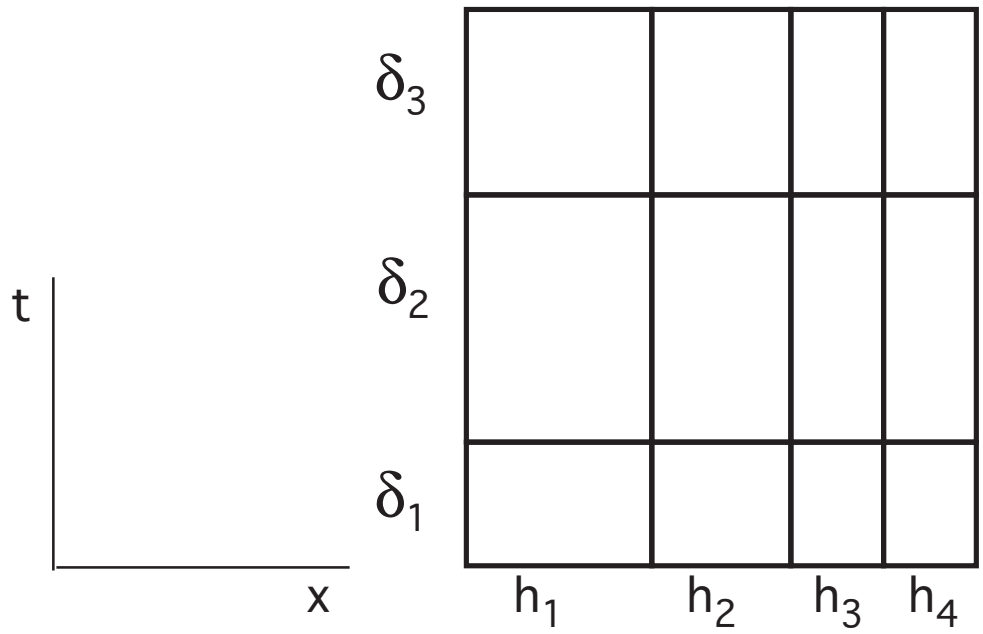
nonuniform Cartesian grid



Cell diameter h_{cell} for a triangle

– we also have have grid sizes in time which are usually called time steps

- the grid size in time is denote by δ and is given by $\delta = \max_n \delta_n$

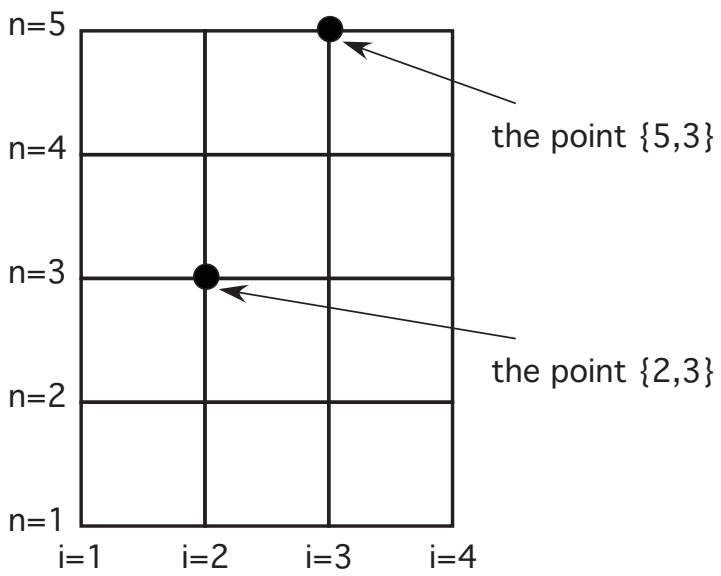


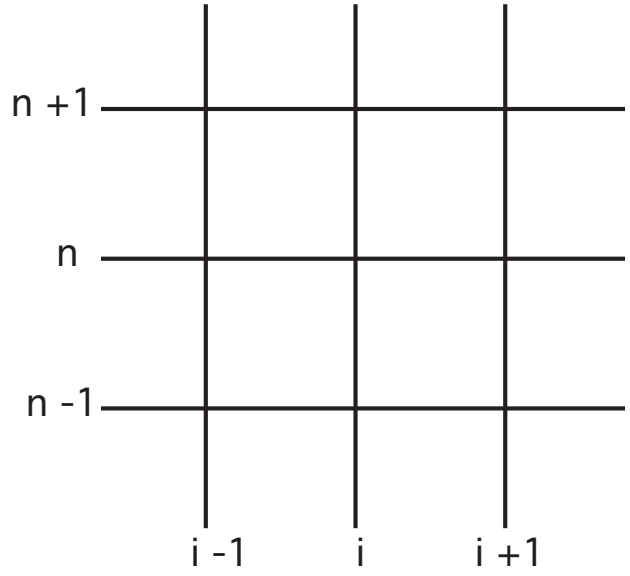
Grid sizes in space and time

- Grid-based discretization methods
  - finite difference methods
    - basic step: approximating differential operators
  - finite element methods
    - basic step: approximating functions
  - finite volume methods
    - somewhere in between finite difference and finite element methods

- There are other discretization methods that are not grid based
  - spectral methods are the best known example
    - polynomials are used to approximate the solution of the PDE
    - usually limited to simple domains, e.g., in 2D, rectangles or ellipses

- Finite difference methods – first thing one does is to replace derivatives by difference quotients





Left: lexicographical numbering of a uniform Cartesian grid
Right: typical grid point $\{n, i\}$ in a uniform Cartesian grid and its neighbors

– let $u_{n,i}$ denote an approximation to $u(t_n, x_i)$

– then

$$\frac{\partial u}{\partial x}(t_n, x_i) \approx \frac{u_{n,i+1} - u_{n,i-1}}{2h} \qquad \frac{\partial u}{\partial t}(t_n, x_i) \approx \frac{u_{n+1,i} - u_{n,i}}{\delta}$$

– the discrete system (the equation you solve on a computer) are obtained by approximating (differential) operators

– the approximate solution of the PDE is a grid function, i.e., a function defined at the nodes of a grid

- after replacing operators by difference quotients, one is left with the task of determining $u_{n,i}$ at all the grid points

- Finite element methods – first thing one does is write down a formula for the approximation of the solution
  - let
    $$\{\phi_1(x,y), \phi_2(x,y), \ldots, \phi_J(x,y)\} = \{\phi_j(x,y)\}_{j=1}^J$$
    denote a finite dimensional basis (a set of linearly independent functions)
  - then we set
    $$u^h(x,y) = \sum_{j=1}^{J} c_j \phi_j(x,y)$$
    and view $u^h(x,y) \approx u(x,y)$
  - one is left with the task of determining the coefficients c_j for $j = 1, \ldots, J$
  - one approximates the solution (not operators)
  - the choice of basis depends on the expected properties of the solution
    - this is one place where PDE theory comes in handy

– where does a grid come into finite element methods?

  - we will see how later

Solving the discrete system

after discretization by any discretization method, one has to solve the resulting discrete system in order to determine the approximate solution

---

- For finite difference methods, one solves for the grid function u_{jn}
- For finite element methods, one solves for the coefficients c_j
- For linear PDEs, the discrete system is often a system of linear algebraic equations
  - direct solvers are mostly limited (because of storage and CPU costs) to problems in one and two dimensions
    - Cholesky factorization for symmetric, positive definite linear systems
    - Gauss elimination for general linear systems
    - super LU is the most efficient implementation of direct methods

– iterative solvers

- Gauss-Seidel and conjugate gradients methods for symmetric, positive definite linear systems
- GMRES, multigrid, and bi-cg methods for general systems

- For nonlinear problems, one has to linearize the equations before applying a linear system solver

  – use Newton's method, quasi-Newton methods, nonlinear conjugate gradient methods, . . .

  – all of these methods are iterative in character, so that a code would be structured as follows

  outer loop – `nonlinear iteration`
  inner loop – `linear solve` (may be direct or iterative)

  – thus, for each iteration of, e.g., Newton's method, one has to solve a linear system

What do we want to know about an approximation method?

- Accuracy – how close is the approximate solution to the exact solution of the PDE?
  - since we do not know the exact solution, the best we can do is to estimate the difference between the two solutions
    - error estimate = usually an upper bound for the difference between the approximate and exact solutions

- Efficiency – how much does it costs (memory and CPU) to obtain the approximate solution?
  - CPU costs are usually given in terms of operation counts
    - e.g., how many multiplications, additions, divisions are needed to solve the problem
    - sometimes these counts can be determined precisely, and at other times, they can only be estimated

- Robustness – how sensitive is the method to changes in the data for the problem?
  - this is related to the notion of the stability of the discretization method

- For all three properties, knowledge of properties of the exact solution of the PDE again comes in handy

Errors in steady-state problems

- Truncation error
  - suppose the PDE is written abstractly as

$$Lu = f$$

    where

    L denotes the partial differential operator

    f denotes given data

    u the solution to be determined

  - suppose the discretized PDE is written abstractly as

$$L_h u^h = f_h$$

    where

    L_h denotes the discretized partial differential operator

    f_h denotes an approximation to the given data

    u^h the approximate solution to be determined

– then

$$\text{truncation error} = \tau^h = L_h u - f_h$$

i.e., the truncation error is what is left after one plugs in the exact solution of the PDE into the discretized PDE

– example:

- consider the PDE (actually, it's an ODE)

$$Lu = -\frac{d^2u}{dx^2} = f(x)$$

- consider the discretized PDE [recall that $u_i^h \approx u(x_i)$] using a uniform grid

$$L_h u^h = -\frac{u_{i+1}^h - 2u_i^h + u_{i-1}^h}{h^2} = f(x_i)$$

where we choosen $f_h = f(x_i)$

- then, the truncation error is given by

$$\tau^h = L_h u - f_h = -\frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} - f(x_i)$$

- we do not know the exact solution so what good is this formula?

- suppose we know (from PDE theory) that the exact solution of the PDE has four continuous derivatives

- we then have Taylor series with remainder

$$u(x_{i\pm1}) = u(x_i \pm h) = u(x_i) \pm h\frac{du}{dx}(x_i) + \frac{h^2}{2}\frac{d^2u}{dx^2}(x_i) \pm \frac{h^3}{6}\frac{d^3u}{dx^3}(x_i) + \frac{h^4}{24}\frac{d^4u}{dx^4}(x_i \pm h\theta_\pm)$$

for some $0 \leq \theta_\pm \leq 1$

- substituting these Taylor series into the the truncation error formula we obtain

$$
\begin{aligned}
\tau^h &= -\frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} - f(x_i) \\
&= \frac{1}{h^2}\Bigg\{ -\Bigg[u(x_i) + h\frac{du}{dx}(x_i) + \frac{h^2}{2}\frac{d^2u}{dx^2}(x_i) \\
&\qquad + \frac{h^3}{6}\frac{d^3u}{dx^3}(x_i) + \frac{h^4}{24}\frac{d^4u}{dx^4}(x_i + h\theta_+)\Bigg] \\
&\qquad + 2u(x_i) \\
&\qquad -\Bigg[u(x_i) - h\frac{du}{dx}(x_i) + \frac{h^2}{2}\frac{d^2u}{dx^2}(x_i) \\
&\qquad - \frac{h^3}{6}\frac{d^3u}{dx^3}(x_i) + \frac{h^4}{24}\frac{d^4u}{dx^4}(x_i - h\theta_-)\Bigg]\Bigg\} \\
&\qquad - f(x_i)
\end{aligned}
$$

- this simplifies to

$$\tau^h = -\frac{d^2u}{dx^2}(x_i) \quad + \quad O(h^2) \quad - f(x_i)$$

where $O(h^\alpha)$ means that we have a term that goes to zero as h tends to zero as the α power of h, i.e.,

$$\epsilon = O(h^\alpha) \qquad \Rightarrow \qquad \lim_{h \to 0} \frac{\epsilon}{h^\alpha} < \infty$$

or

$$|\epsilon| \approx C_t h^\alpha$$

for some constant C_t whose value is independent of h

- but $u(x)$ is the exact solution of the PDE so it satisfies

$$-\frac{d^2u}{dx^2}(x) \quad - \quad f(x) = 0$$

for any x, including $x = x_i$

- we then have obtained an estimate for the truncation error

$$\tau^h = O(h^2) \qquad \text{or} \qquad |\tau^h| \approx C_t h^2$$

– what happens if the exact solution does not possess 4 continuous derivatives?

- if u has 3 continuous derivatives but not 4,
  we have to truncate the Taylor series earlier

$$u(x_{i\pm1}) = u(x_i \pm h) = u(x_i) \pm h\frac{du}{dx}(x_i) + \frac{h^2}{2}\frac{d^2u}{dx^2}(x_i) \pm \frac{h^3}{6}\frac{d^3u}{dx^3}(x_i \pm h\theta_\pm)$$

  which leads a truncation error estimate

$$\tau^h = O(h) \qquad \text{or} \qquad |\tau^h| \approx C_t h$$

  - because h^2 is much smaller than h as $h \to 0$, we see that
    the truncation error is worse when the exact solution
    has 3 derivatives but not 4

- if u has 2 continuous derivatives but not 3,

  we have to truncate the Taylor series even earlier

$$u(x_{i\pm1}) = u(x_i \pm h) = u(x_i) \;\pm\; h\frac{du}{dx}(x_i) \;+\; \frac{h^2}{2}\frac{d^2u}{dx^2}(x_i \pm h\theta_\pm)$$

  which leads a truncation error estimate

$$\tau^h = O(h^0) = O(1) \qquad \text{or} \qquad |\tau^h| \approx C_t$$

  - now the truncation error does not go to zero as the $h \to 0$

- For problems in two and three dimensions, one has to use multivariate Taylor series to determine similar estimates for the truncation error

  – e.g., in two dimensions, we have

$$u(x_{i\pm1}, y_{j\pm1}) = u(x_i, y_j) \pm h_x \frac{\partial u}{\partial x}(x_i, y_j) \pm h_y \frac{\partial u}{\partial y}(x_i, y_j)$$
$$+\frac{h_x^2}{2}\frac{\partial^2 u}{\partial x^2}(x_i, y_j) + h_x h_y \frac{\partial^2 u}{\partial x \partial y}(x_i, y_j) + \frac{h_y^2}{2}\frac{\partial^2 u}{\partial y^2}(x_i, y_j) + \cdots$$

  where h_x and h_y are spatial grid sizes in x and y, respectively

  – clearly, this gets to be rather tedious rather quickly

    - in 2D, there are 4 different third derivatives

    - in 3D, there are 3 different first derivatives, 6 different second derivatives, and 10 different third derivatives

- Error of the approximate solution

$$e^h = u - u^h$$

  - note that because, by definition, the approximate solution satisfies $L_h u^h - f_h = 0$,

$$\begin{aligned} \tau^h &= L_h u - f_h = L_h u - f_h - (L_h u^h - f_h) \\ &= L^h(u - u^h) = L^h e^h \end{aligned}$$

    - the truncation error $\tau^h = L^h(u - u^h)$ tells us something about how well we approximate the operator L

    - the error $e^h = u - u^h$ tells us how well we approximate the solution u

      - this is of more interest than the truncation error

    - so, we would like to get an estimate for the error $e^h = u - u^h$ itself

- Estimating the error

  – suppose we have we have an estimate for the truncation error, i.e., we know that

  $$|\tau^h| \approx C_t h^\alpha$$

  for some $\alpha \geq 0$

  – then, since $L^h e^h = \tau^h$, we have that

  $$e^h = L_h^{-1} \tau^h$$

  where L_h^{-1} denotes the inverse of the discrete operator L_h

  - note that L_h^{-1} is the solution operator, i.e., we have that

  $$u^h = L_h^{-1} f_h$$

  - so, in some way, in order to determine the approximate solution u^h, we have to deal with L_h^{-1}

  – then, for some appropriate norm $\|\cdot\|$,

  $$\|e^h\| \approx C_t \|L_h^{-1}\| h^\alpha$$

– it is tempting, but wrong, to think that

if $\alpha > 0$ (so that the truncation error $\tau^h \longrightarrow 0$ as $h \longrightarrow 0$),

then

necessarily the error $e^h \longrightarrow 0$ as $h \longrightarrow 0$ as well

- we have to worry about $\|L_h^{-1}\|$

– suppose we can determine that $\|L_h^{-1}\| \approx C_\ell h^{-\beta}$ for some constants $C_\ell > 0$ and $\beta \geq 0$

- we then have that

$$\|e^h\| \approx C_\ell C_t h^{\alpha-\beta}$$

– if $\beta = 0$, then

$$\|e^h\| \approx C_e h^\alpha$$

with $C_e = C_\ell C_t$, the same as the truncation error

– if $0 < \beta < \alpha$, then

$$\alpha - \beta > 0$$

and we still have $e^h \to 0$ as $h \to 0$

– if $\beta = \alpha$, then

$$\|e^h\| \approx C_e$$

so that $e^h \not\to 0$ as $h \to 0$

– if $\beta > \alpha$, then

$$\|e^h\| \to \infty \text{ as } h \to 0$$

- Some definitions

  – a method is called consistent if the truncation error $\tau^h \longrightarrow 0$ as $h \longrightarrow 0$

    - in the example, we had that $\tau^h \approx C_t h^2$

      $\Rightarrow$ the discretization method is consistent

  – a method is called stable if $\|L_h^{-1}\| \approx C_\ell$ where $C_e ll > 0$ is a constant independent of h

    - it can be shown that the solution operator in the example satisfies this inequality

      $\Rightarrow$ the discretization method is stable

  – a method is called convergent if $\|e_h\| = \|u - u^h\| \longrightarrow 0$ as $h \longrightarrow 0$

    - because in the example we have $\tau^h \approx C_t h^2$ and $\|L_h^{-1}\| \approx C_\ell$, we have that $\|e_h\| \approx \|L_h^{-1}\||\tau^h| \leq C_t C_\ell h^2$

      $\Rightarrow$ the discretization method is convergent

– if it can be shown that $\|u - u^h\| \approx C_e h^\alpha$ for some constant C_e, then we say that the method is α-order accurate or that the convergence rate of the method is α

- in the example we have $\|e_h\| \approx C_t C_\ell h^2$

$\Rightarrow$ the discretization method is 2nd-order accurate ($C_e = C_t C_\ell$)

• Lax equivalence theorem – if a method is consistent, then it is convergent if and only if it is stable

– consistency ($\tau^h \to 0$) + stability ($\|L_h^{-1}\| \approx C_\ell$) implies convergence

- we have that

$$u - u^h = e^h = L_h^{-1}\tau^h$$

so that

$$\|e^h\| = \|L_h^{-1}\tau^h\| \leq \|L_h^{-1}\|\|\tau^h\| \leq C_\ell\|\tau^h\|$$

so that $e^h \to 0$ if $\tau^h \to 0$

- consistency ($\tau^h \longrightarrow 0$) + convergence ($\|e^h\| \longrightarrow 0$) implies stability
  - harder to prove

- Directly showing that a method is convergent is sometimes difficult
  - this is especially true for finite difference methods
  - the Lax equivalence theorem is a useful tool to prove the convergence of a method
    - directly showing that a method is consistent is usually easy
      - just use Taylor series to determine the truncation error τ^h
    - showing that method is stable is not so easy, but is often possible

- The Lax equivalence theorem says nothing about the convergence of a method that is not consistent
  - in fact, it is possible for a non-consistent method to be convergent
    - this happens often with finite element methods
  - in such cases, one cannot use the Lax equivalence theorem to prove convergence
    - one must prove convergence directly
- Proving the convergence of finite element methods is greatly facilitated by the fact that one can remove oneself from the grid
  - because finite element approximations are functions that are defined at every point (not just at the nodes), one can do the convergence analysis in function spaces that a lot is known about
    - one can bring to bear powerful results from functional analysis

– in contrast, for finite difference methods, approximations are defined at the nodes so we have to deal with grid functions

  - the functional analysis of grid functions is not nearly so well developed

  - as a result, compared to finite element methods, it is usually much more difficult to directly prove the convergence or to determine convergence rates of finite difference methods

Errors in time-dependent problems

- We have two grid sizes in the problem
  - the spatial grid size h
  - the temporal grid size δ

- As a result, the truncation error τ^h will take the form

$$\tau^h \approx C_t(h^\alpha + \delta^\beta)$$

  for some positive numbers α and β

  - we have to use multivariate Taylor series to determine the estimate for the truncation error

  - for more complicated problems, e.g., nonlinear problems, the truncation error could even take the form

$$\tau^h \approx C_t(h^\alpha + \delta^\beta + h^{\widehat{\alpha}}\delta^{\widehat{\beta}})$$

    where $\widehat{\alpha}$ and $\widehat{\beta}$ are two additional positive numbers

- Estimates for the error in the solution $e^h = u - u^h$ will take similar forms

- With the mostly obvious notational changes, everything else discussed about errors for time-independent problems holds for time-dependent problems

Determining convergence rates on a computer

- We have that the error is defined by

$$e^h = u - u^h$$

where u is the exact solution of the PDE problem
and u^h is the approximate solution of the PDE problem

- Measuring the error for finite difference methods (we will do this for finite element methods later)

  – recall that in a finite difference method, the approximate solution u^h is a grid function

    - thus measures of the error should involve comparing the exact and approximate solutions at the nodes

– max or ℓ_∞ norm or the error

$$\|e^h\|_\infty = \max_{\text{grid points}} |u(\text{grid point}) - u_{\text{grid points}}|$$

so that for a 2D problem on a Cartesian grid, we would have

$$\|e^h\|_\infty = \max_{i,j} |u(x_i, y_j) - u_{i,j}|$$

– root mean square or ℓ^2 norm

$$\|e^h\|_2 = \left(\sum_{\text{grid points}} |u(\text{grid point}) - u_{\text{grid points}}|^2 \right)^{1/2}$$

so that for a 2D problem on a Cartesian grid, we would have

$$\|e^h\|_2 = \left(\sum_{i,j} |u(x_i, y_j) - u_{i,j}|^2 \right)^{1/2}$$

- We assume that, for a steady state problem, the error $e^h = u - u^h$ satisfies

$$\|e^h\| \approx C_\ell h^\alpha$$

for some numbers $C_\ell > 0$ and $\alpha > 0$

- Our task is to determine (an approximation) for α, the rate of convergence

- Suppose that for two grid sizes h_1 and h_2, we could determine $\|e^{h_1}\|$ and $\|e^{h_2}\|$

  – then, according to our assumption, we have that

$$\|e^{h_1}\| \approx C_\ell h_1^\alpha \qquad \text{and} \qquad \|e^{h_2}\| \approx C_\ell h_2^\alpha$$

  – dividing these two relations we get

$$\frac{\|e^{h_2}\|}{\|e^{h_1}\|} \approx \frac{C_\ell h_2^\alpha}{C_\ell h_1^\alpha} = \frac{h_2^\alpha}{h_1^\alpha} = \left(\frac{h_2}{h_1}\right)^\alpha$$

– taking the $\log$ of both sides (can use natural or base-10 logarithms), we get

$$\log\left(\frac{\|e^{h_2}\|}{\|e^{h_1}\|}\right) \approx \alpha \log\left(\frac{h_2}{h_1}\right)$$

or

$$\alpha \approx \frac{\log\left(\frac{\|e^{h_2}\|}{\|e^{h_1}\|}\right)}{\log\left(\frac{h_2}{h_1}\right)}$$

– for example, if $h = h_1 = 2h_2$, we have

$$\alpha \approx \frac{\log\left(\frac{\|e^{h/2}\|}{\|e^{h}\|}\right)}{-\log 2}$$

– for example,

- if we believe that a method is second-order accurate, i.e.,

$$\alpha = 2 \qquad \text{and} \qquad e^h \approx C_\ell h^2$$

then we would expect that

$$e^{h/2} = C_\ell \left(\frac{h}{2}\right)^2 = \frac{1}{4} C_\ell h^2 \approx \frac{1}{4} e^h$$

- if the computations really show that $\|e^{h/2}\| = \frac{1}{4}\|e^h\|$ then, from the computations, we determine that

$$\alpha \approx \frac{-\log 4}{-\log 2} = \frac{\log 2^2}{-\log 2} = \frac{2\log 2}{\log 2} = 2$$

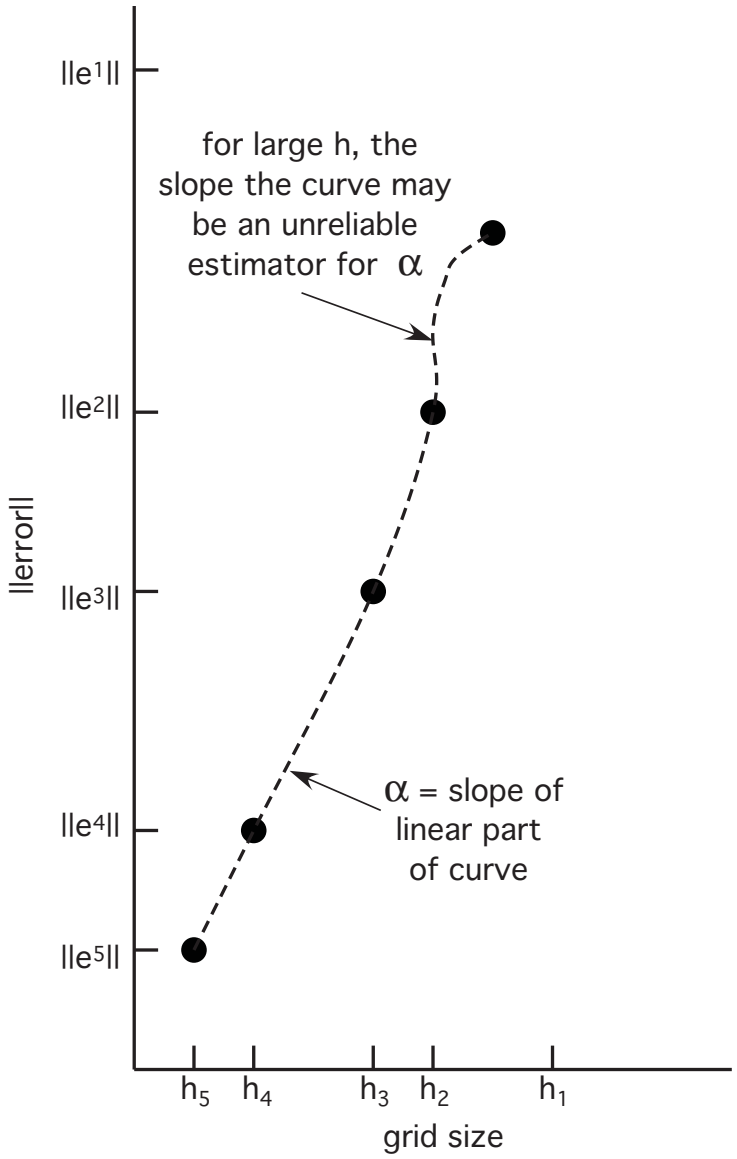
and we have verified our belief

- if it turns out that the approximation to α determined from the computations is less than 2, then our belief is not confirmed

- one must make sure that the grids sizes we use are "small enough"; estimates for the error hold "as $h \to 0$" so that for big grid sizes, we cannot expect that the error will follow the expected pattern $e^h \approx Ch^\alpha$

– usually, one calculates with several qrids with sizes $\{h_1, h_2, \ldots, h_K\}$

- one then obtains the corresponding errors $\{e^{h_1}, e^{h_2}, \ldots, e^{h_K}\}$
- one then uses these results to find and approximation for α, by e.g., either of
    - linear regression
    - graphically using a log-log plot



Log-log plot of error vs. grid size

– problem with all this: we do not know the exact solution so how do we determine the error $e^h = (u - u^h)$?

• Method of manufactured solutions – we do things backwards

– to test for the accuracy of a method,

- instead of

specifying the data for the PDE problem and then finding the exact solution (which may be impossible to do),

we

specify an exact solution and then determine the data (this is easy to do)

- then

we use the manufactured data to determine an approximate solution,

pretending we do not know the exact solution

– example: Poisson problem

$$\begin{cases} -\dfrac{\partial^2 u}{\partial x^2} - \dfrac{\partial^2 u}{\partial y^2} = f(x,y) & \text{for } x, y \text{ in } \Omega \\ u = g(x,y) & \text{for } x, y \text{ on } \Gamma \end{cases}$$

- in the problem we really want to solve,

  - we would be given $f(x,y)$ and $g(x,y)$

  - then, we would like to determine the exact solution $u(x,y)$ for x, y in Ω

  - this usually impossible to do

  - this is precisely why one is interested in finding approximate solutions in the first place

- to test the accuracy of a method, suppose instead one sets, i.e, manufactures the solution,

$$u_{man}(x,y) = e^x \cos 3y$$

- then, one easily finds that (by substitution into the PDE problem instead of solving the PDE problem)

$$\begin{cases} f_{man}(x,y) = -e^x \cos 3y + 9e^x \cos 3y = 8e^x \cos 3y \\ g_{man}(x,y) = e^x \cos 3y \end{cases}$$

- then one finds the approximate solution for the problem

$$\begin{cases} -\dfrac{\partial^2 u_{man}}{\partial x^2} - \dfrac{\partial^2 u_{man}}{\partial y^2} = f_{man}(x,y) = 8e^x \cos 3y & \text{for } x, y \text{ in } \Omega \\ u = g_{man}(x,y) = e^x \cos 3y & \text{for } x, y \text{ on } \Gamma \end{cases}$$

pretending that one does not know that $u_{man}(x,y) = e^x \cos 3y$

- in this way we have
    - the exact solution u_{man} that we manufactured
    - an approximate solution u^h_{man} that we determined on the computer using an approximation method

  so that we can now compute the error $e^h = u_{man} - u^h_{man}$ for the manufactured solution problem

– how does one pick the manufactured solution u_{man}?

- it is important to choose u_{man} so that it has the features one expects of the exact solution u of the PDE problem we really want to solve
- PDE theory comes in handy here to tell us something about the exact solution u of the PDE problem we really want to solve
- for example, if PDE theory says that we can expect the exact solution u of the problem we really want to solve has only 2 derivatives but not not 3, then we should manufacture a solution u_{man} that also only has 2 derivatives

- Comparing to very fine grid solutions
  - determine an approximate solution $u^{h_{\text{very fine}}}$ using a very fine grid, i.e., a very small grid size $h_{\text{very fine}}$
  - assume that that fine grid solution is a very good approximation $u^{h_{\text{very fine}}}$ to the exact solution u
  - compute approximate solutions for two grid sizes h_1 and h_2 such that $h_1 \gg h_{\text{very fine}}$ and $h_1 \gg h_{\text{very fine}}$
    - of course, can use several grid sizes $\{h_1, \ldots, h_K\}$
  - then proceed as in the manufacuted solution case, except that now $u^{h_{\text{very fine}}}$ is used as a surrogate for the unknown exact solution u
  - advantages of this approach
    - do not need to create a manufactured solution
    - can use the actual data (e.g., f and g) of the problem one really wants to solve

– disadvantages of this approach

- sometimes, because of limitation on available computational resources, one cannot use a fine enough grid size $\widehat{h}$ to obtain a reliable very good approximation $u^{h_{\textit{very fine}}}$ to the exact solution u

- in general, results are not so reliable