PH5720

In this unit you will learn the determination of the numerical solution of differential equations using different approaches. To reduce the computation cost and achieve accurate numerical solution with small error, it is always advisable to compute the differential equation with larger stability region. You will study to solve the first order differential equations. There are several examples in the slide. You can write small program and see the numerical solution of the differential equation. The unit will focus on the following subsection.

- Ordinary Differential Equations: Initial Value Problems
- Ordinary Differential Equations: Boundary value Problems
- Partial Differential Equations

Numerical methods for Partial differential equations

A **partial differential** equation, or PDE, is an equation involving partial derivatives of an unknown function with respect to **more than one independent variable**.

Independent variable typically include one or more space dimensions and possibly time dimension as well.

More dimensions **complicate problem formulation**, we can have pure initial value problem, pure boundary value problem, or mixture of both.

Equation and boundary data may be defined over irregular domain.

For simplicity, we will deal only with single PDEs with **only two independent variables** $(x, y)\ or\ (x, t)$ and corresponding partial derivatives.

$$u_t = \frac{\partial u}{\partial t} \qquad or \qquad u_{xy} = \frac{\partial^2 u}{\partial x \partial y}$$

Numerical methods for PDEs, Classification

Order of PDE is order of the highest-order partial derivative appearing in equation.

For examples,

First order

Transport equation in two independent variables.

Second order

Heat equation

Wave equation

Laplace equation

Numerical methods for PDEs, Classification

Second-order linear PDEs of general form

$$au_{xx} + bu_{xy} + cu_{yy} + du_x + eu_y + fu + g = 0$$

are classified by value of discriminant $b^2 - 4ac$

$b^2 - 4ac > 0$	hyperbolic	**Wave equation**
$b^2 - 4ac = 0$	parabolic	**Heat equation**
$b^2 - 4ac < 0$	elliptic	**Laplace equation**

Numerical methods for PDEs, Classification

Classification of more general PDEs is not so clean and simple, but roughly speaking

Hyperbolic PDEs describe time-dependent, conservative physical processes, such as convection, that are not evolving toward steady state.

Parabolic PDEs describe time-dependent, dissipative physical processes, such as diffusion, that are evolving toward steady state.

Elliptic PDEs describe processes that have already reached steady state, and hence are time-independent.

Numerical methods for PDEs

Time-dependent problems

Semidiscrete methods

Semidiscrete finite difference

Semidiscrete collocation method

Fully discrete methods

Implicit finite difference methods

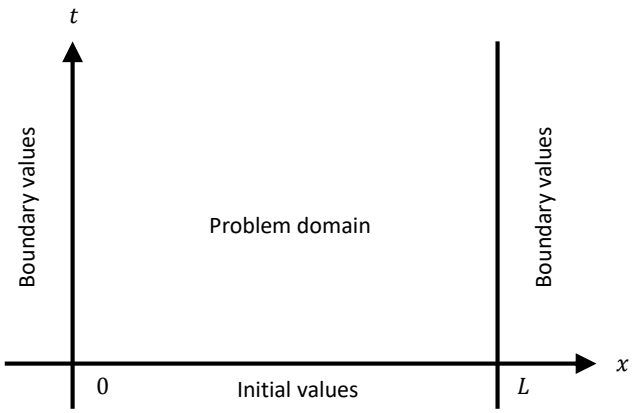
Time-independent problems

Finite difference methods

Finite element methods

Numerical methods for Time-dependent PDEs problems

Time-dependent PDEs usually involve both initial values and boundary values



Numerical methods for PDEs, Semidiscrete methods

One way to solve time-dependent PDE numerically is to discretize in space but leave the time variable continuous.

This approach results in a system of ODEs, which can then be solved by the methods previously discussed.

For example, consider the **heat equation**

$$u_t = cu_{xx} \qquad 0 \le x \le 1 \qquad t \ge 0$$

With initial conditions

$$u(0,x) = f(x) \qquad 0 \le x \le 1$$

And boundary conditions

$$u(t,0) = 0 \qquad u(t,1) = 0 \qquad t \ge 0$$

Numerical methods for PDEs, Semidiscrete finite difference methods

Define spatial mesh points $x_i = i\Delta x, \quad i = 0, 1, \cdots\cdots n+1$
where $\Delta x = \frac{1}{n+1}$

Replace derivatives u_{xx} by finite difference approximation

$$u_{xx}(t, x_i) \approx \frac{u(t, x_{i+1}) - 2u(t, x_i) + u(t, x_{i-1})}{(\Delta x)^2}$$

Result is system of ODEs

$$y_i'(t) = \frac{c}{(\Delta x)^2} \quad y_{i+1}(t) - 2y_i(t) + y_{i-1}(t) \quad i = 1, 2, \cdots\cdots\cdots, n$$

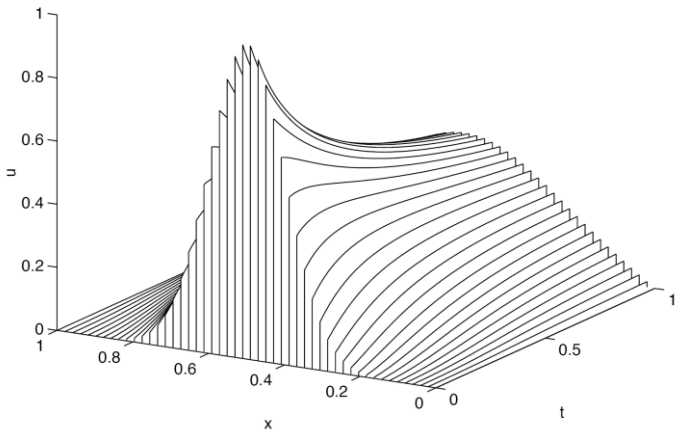
where $y_i(t) \approx u(t, x_i)$

From boundary conditions $y_0(t) = 0 = y_{n+1}(t)$
From initial conditions $y_i(0) = f(x_i), \quad i = 1, 2, \cdots\cdots\cdots, n$

We can therefore use ODE method to solve IVP for this system-this approach is called **Methods of lines**.

Numerical methods for PDEs, Method of lines

Method of lines uses ODE solver to compute cross-sections of solution surface over space-time plane, along a series of lines, each parallel to time axis and corresponding to discrete spatial mesh point.



Numerical methods for PDEs, Stiffness

Semidiscrete system of ODEs just derived can be written in matrix form

$$y' = \frac{c}{(\Delta x)^2} \begin{bmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \cdots & 0 \\ 0 & 1 & -2 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & -2 \end{bmatrix}$$

Jacobian matrix A of this system has eiegenvalues between $-4c/(\Delta x)^2$ and 0, which makes ODE very **stiff** as spatial mesh size Δx becomes small

This stiffness, which is typical of ODEs derived from PDEs, must be taken into account in choosing ODE method for solving semidiscrete system.

Numerical methods for PDEs, Semidiscrete collocation method

Spatial discretization to convert PDE into system of ODEs can also be done by spectral or finite elements approach.

Approximate solution is expressed as **linear combination of basis functions**, but with **time dependent coefficients**.

Thus, we seek solution of form

$$u(t,x) \approx v\big(t,x,\alpha(t)\big) = \sum_{j=1}^{n} \alpha_j(t)\, \phi_j(x)$$

where $\phi_j(x)$ are suitably chosen basis functions

If we **use collocation**, then we substitute this approximation into the PDE and require that the equation be satisfied exactly at a discrete set of **collocation points** x_i.

Numerical methods for PDEs, Semidiscrete collocation method

For **heat equation**, this yields system of ODEs

$$\sum_{j=1}^{n} \alpha_j'(t)\, \phi_j(x_i) = c \sum_{j=1}^{n} \alpha_j(t) \phi_j''(x_i)$$

whose solution is the **set of coefficient functions** $\alpha_i(t)$, that determine the approximate solution to PDE.

Implicit form of this is not the explicit form required by standard ODE methods, however, so we define the $n \times n$ matrices M and N by

$$m_{ij} = \phi_j(x_i) \qquad n_{ij} = \phi_j''(x_i)$$

Numerical methods for PDEs, Semidiscrete collocation method

Assuming the matrix M is **nonsingular**, we then obtain the system of ODEs

$$\alpha'(t) = cM^{-1}N\alpha(t)$$

which is in form suitable for solution with standard ODE software

As usual, the matrix M need not be **inverted explicitly**, but merely used to solve linear systems.

Initial condition for this ODE, can be obtained by requiring solution to satisfy given initial condition for the PDE **at mesh points** x_i.

Matrices involved in this method will be sparse if the basis functions are local, such as **B-splines**, which gives a finite element method.

Numerical methods for PDEs, Semidiscrete collocation method

Unlike the finite difference method, a **spectral or finite element** method does not produce approximate values of the solution u directly, but rather it **generates a representation of the approximate solution** as a **linear combination of basis functions**.

The basis functions depend only on the **spatial variable**, but the coefficients of the linear combination (given by the solution to the system of ODEs) are **time dependent**.

Thus, for any given time t, the corresponding **linear combination** of basis functions **generates a cross section** of the solution **surface parallel** to the spatial axis.

As with finite difference methods, systems of ODEs arising from semidiscretization of PDE by spectral or finite element methods tend to be **stiff**.

Numerical methods for PDEs, Fully discrete methods

Fully discrete methods, for PDEs are discretize in both space and time dimension.

In a fully discrete finite difference method, we

replace continuous domain of **equation by discrete mesh of points**.

replace all the **derivatives** in the PDE by **finite difference approximations**,

seek numerical solution as table of approximation values at selected points in space and time.

Numerical methods for PDEs, Fully discrete methods

In two dimension (one space and one time), resulting approximate solution values represent points on **solution surface** over problem domain in space –time plane.

Accuracy of approximate solution depends on step sizes in both space and time.

Replacement of all partial derivatives by finite differences results in system of algebraic equations for unknown solution at discrete set of sample points.

Discrete system may be **linear or non-linear**, depending on underlying PDE.

Numerical methods for PDEs, Fully discrete methods

With initial-value problem, solution is obtained by starting with initial values along boundary of problem domain and marching forward in time step by step, generating successive rows in solution table.

Time-stepping procedure may be **explicit or implicit**, depending on whether formula for solution values at next time step involves only past information.

We might expect to obtain **arbitrarily good accuracy** by taking **sufficiently small** step sizes in time and space.

Time and space **step sizes can not** always be chosen independently of each other.

Example : heat equation

Numerical methods for PDEs, Example : heat equation

Consider heat equation

$$u_t = cu_{xx} \qquad 0 \le x \le 1 \qquad t \ge 0$$

with initial boundary conditions

$$u(0,x) = f(x) \qquad u(t,0) = \alpha \qquad u(t,1) = \beta$$

Definite spatial mesh points $x_i = i\Delta x, \quad i = 0, 1, \cdots\cdots n+1$
Where

$$\Delta x = \frac{1}{n+1}$$

and temporal mesh points $t_k = k\Delta t,$ for suitably chosen Δt

Let u_i^k denote approximate solution at (t_k, x_i)

Numerical methods for PDEs, Example : heat equation

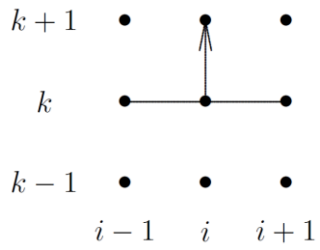
Consider heat equation $u_t = cu_{xx}$ $\quad 0 \le x \le 1 \quad t \ge 0$

Replacing u_t by forward difference in time and u_{xx} by centered difference in space, we obtain

$$\frac{u_i^{k+1} - u_i^k}{\Delta t} = c\frac{u_{i+1}^{k+1} - 2u_i^{k+1} + u_{i-1}^{k+1}}{(\Delta x)^2}$$

$$u_i^{k+1} = u_i^k + c\frac{\Delta t}{(\Delta x)^2}\left(u_{i+1}^{k+1} - 2u_i^{k+1} + u_{i-1}^{k+1}\right) \qquad i = 1, \cdots, n$$

Stencil : pattern of mesh points involved at each level



Numerical methods for PDEs, Example : heat equation

Consider heat equation $u_t = cu_{xx}$ $\quad 0 \le x \le 1 \quad t \ge 0$

Boundary condition gives us

$$u_0^k = \alpha \qquad u_{n+1}^k = \beta$$

for all k, and initial conditions provide starting values

$$u_i^0 = f(x_i) \qquad i = 1, \cdots, n$$

So we can march numerical solution forward in time using this **explicit** difference scheme.

The **local truncation error** of this scheme is $\mathcal{O}(\Delta t) + \mathcal{O}\left((\Delta x)^2\right)$, so scheme is **first-order accurate in time** and **second-order accurate in space.**

Example : wave equation

Numerical methods for PDEs, Example : wave equation

Consider wave equation

$$u_{tt} = cu_{xx} \qquad 0 \le x \le 1 \qquad t \ge 0$$

with initial boundary conditions

$$u(0,x) = f(x) \qquad u_t(0,x) = g(x) \qquad 0 \le x \le 1$$

$$u(t,0) = \alpha \qquad u(t,1) = \beta \qquad t \ge 0$$

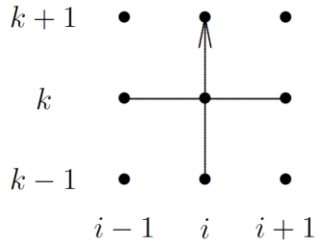
Numerical methods for PDEs, Example : wave equation

Consider wave equation $u_{tt} = cu_{xx}$ $0 \le x \le 1$ $t \ge 0$

With mesh points defined as before, using centered difference formulas for both u_{tt} and u_{xx} gives finite difference scheme

$$\frac{u_i^{k+1} - 2u_i^k + u_i^{k-1}}{(\Delta t)^2} = c\frac{u_{i+1}^k - 2u_i^k + u_{i-1}^k}{(\Delta x)^2}$$

$$u_i^{k+1} = 2u_i^k - u_i^{k-1} + c\left(\frac{\Delta t}{\Delta x}\right)^2 \left(u_{i+1}^k - 2u_i^k + u_{i-1}^k\right) \qquad i = 1,\cdots,n$$



Numerical methods for PDEs, Example : wave equation

Consider wave equation $u_{tt} = cu_{xx}$ $0 \le x \le 1$ $t \ge 0$

With mesh points defined as before, using centered difference formulas for both u_{tt} and u_{xx} gives finite difference scheme

$$u_i^{k+1} = 2u_i^k - u_i^{k-1} + c\left(\frac{\Delta t}{\Delta x}\right)^2 \left(u_{i+1}^k - 2u_i^k + u_{i-1}^k\right) \qquad i = 1,\cdots,n$$

Using data at two levels in time requires additional storage

We also need u_i^0 and u_i^1 to get started, which can be obtained from initial conditions

$$u_i^0 = f(x_i), \qquad u_i^1 = f(x_i) + (\Delta t)g(x_i)$$

where latter uses forward difference approximation to initial condition

$$u_i(0,x) = g(x)$$

Numerical methods for PDEs, Stability

Unlike method of lines, where time step is chosen automatically by ODE solver, user must choose **time step** Δt in **fully discrete method**, taking into account both **accuracy and stability** requirements.

For example, the fully discrete finite difference scheme for the heat equation is simply **Euler's** method applied to the semidiscrete system of ODEs derived for heat equation given previously.

We saw that **Jacobian** matrix of semidiscrete system has eigenvalues between $-4c/(\Delta x)^2$ and 0, so stability region for Euler's method requires time step to satisfy

$$\Delta t \leq \frac{(\Delta x)^2}{2c}$$

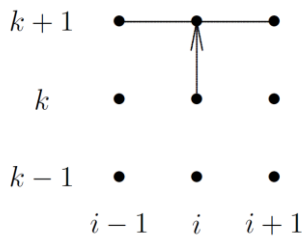
Severe restriction on **time step** can make explicit methods relatively **inefficient**.

Numerical methods for PDEs, Implicit finite difference methods

For ODEs we saw that implicit methods are **stable** for much greater range of **step sizes**, and same is true of **implicit methods** for PDEs.

Applying **backward Euler method** to semidiscrete system for heat equation gives implicit finite difference scheme.

$$u_i^{k+1} = u_i^k + c\frac{\Delta t}{(\Delta x)^2}\left(u_{i+1}^{k+1} - 2u_i^{k+1} + u_{i-1}^{k+1}\right) \qquad i = 1, \cdots, n$$



Numerical methods for PDEs, Implicit finite difference methods

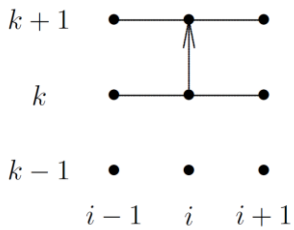
This scheme inherits **unconditional stability** of backward Euler method, which means there is **no stability restriction** on relative sizes of ***Δt & Δx***.

But **first-order accuracy** in time still severely **limits time step**.

Applying trapezoid method to semidiscrete system of ODEs for heat equation yields **implicit Crank-Nicolson** method.

$$u_i^{k+1} = u_i^k + c\frac{\Delta t}{2(\Delta x)^2}\left(u_{i+1}^{k+1} - 2u_i^{k+1} + u_{i-1}^{k+1} + u_{i+1}^k - 2u_i^k + u_{i-1}^k\right)$$

Method is unconditionally stable and **second-order accurate** in time.



Numerical methods for PDEs, Implicit finite difference methods

Much greater stability of **implicit finite difference methods** enables them to take much **larger time steps** than are permissible with explicit methods, but they **require more work per step**, since system of equations must be solved at each step.

For both the **backward Euler** and **Crank-Nicolson** methods for the heat equation in one space dimension, this linear system is tridiagonal, and thus both the work and the storage required are modest.

In **higher dimensions** the matrix of the linear system does not have such a simple form, but it is still very sparse, with non-zeros in a very regular pattern.

Numerical methods for PDEs, Convergence

In order for approximate solution to converge to true solution of PDE as step sizes in time and space jointly go to zero, following two conditions must hold.

Consistency: the local truncation error goes to zero

Stability: the approximate solution at any fixed time t remains bounded

Lax Equivalence Theorem says that for a well-posed linear PDE, consistency and stability are together necessary and sufficient condition for convergence.

Neither consistency nor stability condition alone is sufficient to guarantee convergence.

Numerical methods for PDEs, stability

Consistency is usually fairly easy to verify using a **Taylor series expansion**.

Analysing **stability** is more challenging and several methods are available

Matrix method, is based on the locations of the eigenvalues of the matrix representation of the finite difference scheme, as we saw with Euler's method for the heat equation

Fourier method, in which complex exponential representation of the solution error is substituted into the difference equation and then analyzed for growth or decay.

Domains of dependence, in which domain of dependence of the PDE and difference scheme are compared.

Numerical methods for PDEs, CFL condition

Domain of dependence of PDE is portion of problem domain that influences solution at given point, which depends on characteristics of PDE.

Domain of dependence of difference scheme is set of all other mesh points that affect approximate solution at the given mesh point.

CFL condition (named for its originators, **Courant**, **Friedrichs**, and **Lewy**): necessary condition for explicit finite difference scheme for hyperbolic PDE to be stable is that for each mesh point domain of dependence of PDE must lie within the domain of dependence of the finite difference scheme.

Example : wave equation

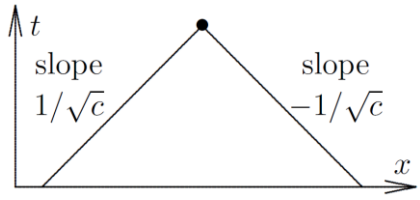
Numerical methods for PDEs, Example : wave equation

Consider the explicit finite difference scheme for the wave equation

$$u_{tt} = cu_{xx} \qquad 0 \le x \le 1 \qquad t \ge 0$$

Characteristics of the wave equation are the straight lines in the (t, x) plane along which either $(x + \sqrt{c}\, t)$ or $(x - \sqrt{c}\, t)$ is constant.

Domain of dependence of the wave equation for given point is triangle with apex at the given point and with sides of slope $1/\sqrt{c}$ and $-1/\sqrt{c}$

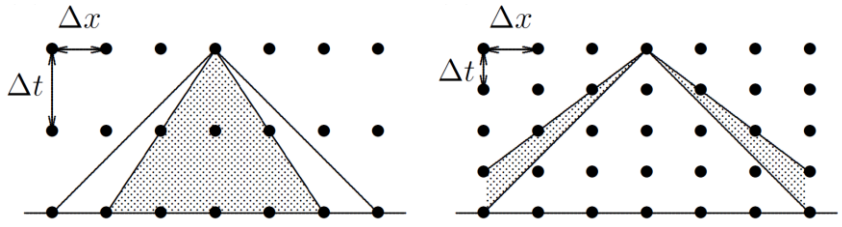


Numerical methods for PDEs, Example : wave equation

CFL condition implies, step sizes must satisfy

$$\Delta t \leq \frac{\Delta x}{\sqrt{c}}$$

For this particular finite difference scheme



Numerical methods for PDEs, Time-independent problems

We next consider time-independent, elliptic PDFs in two space dimensions, such as **Helmholtz equation**

$$u_{xx} + u_{yy} + \lambda u = f(x, y)$$

Important special cases

Poisson equation : $\lambda = 0$
Laplace equation : $\lambda = 0$ and $f = 0$

For simplicity, we will consider this equation on unit square

Numerous possibilities for boundary conditions specified along each side of square

Dirichlet : u is specified
Neumenn : u_x or u_y is specified
Mixed : combination of these are specified

Numerical methods for PDEs, Finite difference methods

Finite difference methods for elliptic boundary value problems proceed as before:

Define a **discrete mesh** of points within domain of equation.

Replace the derivatives in the PDE by finite difference approximations.

Seek a numerical solution at all of the mesh points.

Unlike time-dependent problems, solution is not produced by marching forward step by step in time.

Approximate solution is determined **at all mesh points** simultaneously by solving single system of algebraic equations.

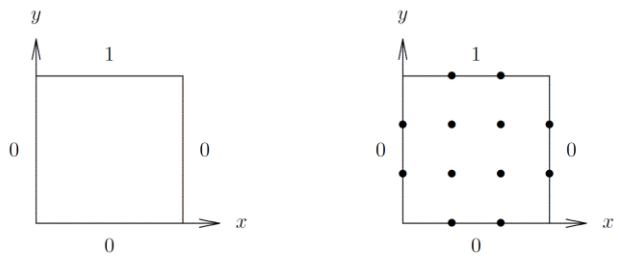
Example: Laplace equation

Numerical methods for PDEs, Example: Laplace equation

Consider Laplace equation

$$u_{xx} + u_{yy} = 0$$

On unit square with boundary conditions shown below left



Define a discrete mesh in domain, including boundaries, as shown above in the right.

Numerical methods for PDEs, Example: Laplace equation

Interior grid points where we will compute approximate solution are given by

$$(x_i, x_j) = (ih, jh) \qquad i, j = 1, \cdots\cdots, n$$

where in example $n = 2$ and $h = 1/(n+1) = 1/3$

Next we replace derivatives by centered difference approximation at each interior mesh point to obtain the finite difference equations

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2} = 0$$

where $u_{i,j}$ is approximation to true solution $u(x_i, y_j)$ for $i, j = 1, \cdots\cdots, n$, and represent one of given boundary values if i *or* j is 0 or $n+1$

Numerical methods for PDEs, Example: Laplace equation

Simplifying and writing out the resulting four equations explicitly, we obtain

$$4u_{1,1} - u_{0,1} - u_{2,1} - u_{1,0} - u_{1,2} = 0$$

$$4u_{2,1} - u_{1,1} - u_{3,1} - u_{2,0} - u_{2,2} = 0$$

$$4u_{1,2} - u_{0,2} - u_{2,2} - u_{1,1} - u_{1,3} = 0$$

$$4u_{2,2} - u_{1,2} - u_{3,2} - u_{2,1} - u_{2,3} = 0$$

Numerical methods for PDEs, Example: Laplace equation

Writing previous equations in matrix form gives

$$A_x = \begin{pmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{pmatrix} \begin{bmatrix} u_{1,1} \\ u_{2,1} \\ u_{1,2} \\ u_{2,2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = b$$

System of equations can be solved for unknowns $u_{i,j}$ either by Cholesky factorization or by an iterative method, yielding the solution.

$$x = \begin{bmatrix} u_{1,1} \\ u_{2,1} \\ u_{1,2} \\ u_{2,2} \end{bmatrix} = \begin{bmatrix} 0.125 \\ 0.125 \\ 0.375 \\ 0.375 \end{bmatrix}$$

Numerical methods for PDEs, Example: Laplace equation

In a practical problem, the mesh size h would need to be much smaller and resulting linear system would be much larger

Matrix would be very sparse, however, since each equation would still involve at most only five of the variables, thereby saving substantially on work and storage.

Numerical methods for PDEs, Finite element methods

Finite element methods are also applicable to boundary value problems for PDEs as well.

Conceptually, there is no change in going from one dimension to two or three dimensions

Solution is still represented as a linear combination of basis functions.

Some criterion (e.g., Galerkin) is applied to derive a system of equations that determines the coefficients of the linear combination.

The main practical difference is that instead of subintervals in one dimension, the elements become triangles or quadrilaterals in two dimensions, and tetrahedra or hexahedra in three dimensions.

Numerical methods for PDEs, Finite element methods

Basis functions typically used are bilinear or bicubic functions in two dimensions and trilinear or tricubic functions in three dimensions, analogous to piecewise linear “hat” function or piecewise cubics in one dimension.

Increasing in dimensionality means that the linear system to be solved is much larger, but it is still sparse due to local support of the basis functions.

Finite element methods for PDEs are extremely flexible and powerful, but a detailed treatment of them is beyond the scope of this course.