




On the Discretization Methods for Partial Differential Equations

ASST. PROF. ZAFEİRAKİS ZAFEİRAKOPOULOS

Mehmet Kocabaş – Caner Aslan – Ali Anıl
Apaydın – Uğur Eniş

Partial Differential Equations

- 
- Partial Differential Equations (PDEs)
 - Domain Decomposition Method
 - Finite Difference Method
 - Finite Element Method
 - Finite Volume Method
 - Conclusion

Partial Differential Equations

A **partial differential equation** (**PDE**) is an equation that involves an unknown function and its partial derivatives.

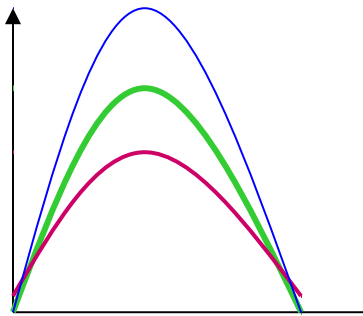
Example :

$$\frac{\partial^2 u(x, t)}{\partial x^2} = \frac{\partial u(x, t)}{\partial t}$$

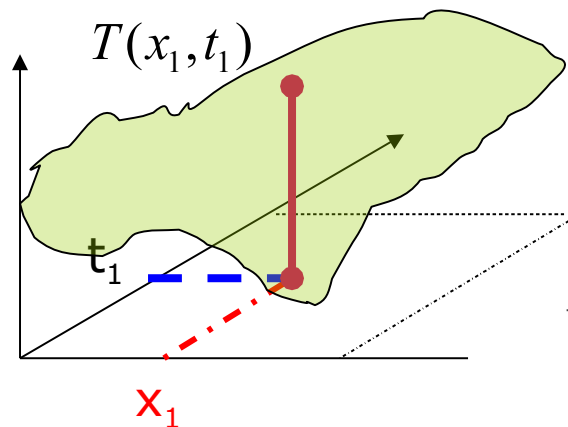
PDE involves two or more independent variable s
(in the example x and t are independent variable s)

Representing the Solution of a PDE (Two Independent Variables)

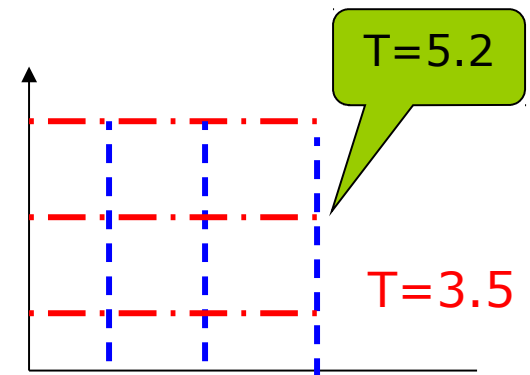
- Three main ways to represent the solution



Different curves are used for different values of one of the independent variable



Three dimensional plot of the function $T(x, t)$



The axis represent the independent variables. The value of the function is displayed at grid points


Examples of PDEs

PDEs are used to model many systems in many different fields of science and engineering.

Important Examples:

- Laplace Equation
- Heat Equation
- Wave Equation

Domain Decomposition Method

- 
- Definition of DDM
 - Background
 - Classifications

Definition of DDM

- Domain decomposition methods are iterative methods for the solution of linear or nonlinear systems that use ;
 - Explicit information about the geometry,
 - Discretization
 - Partial differential equation (PDE) that gave rise to the (non)linear system.

Background and Model Problems

- Domain decomposition algorithms may be applied to a variety of Partial differential equations.
- To simplify the presentation, however, we restrict attention to linear, second-order, elliptic PDEs,

$$Lu = f \text{ in } \Omega;$$

$$Bu = g \text{ on } \partial\Omega;$$

- The variationally form of the PDE may be written as: find $u \in V$ such that;

$$\mathbf{a(u, v) = F(v) \quad \forall u \in V}$$

(Here, $a(u, v)$ is a bilinear form, while $F(v)$ is linear in v)

Background

- If we use ϕ_i to denote the finite element basis functions and
- Let $V^h = \text{span}\{\phi_i\}$, then the finite-dimensional variationally (finite element) problem may be given as:

$$u^h = \sum_{i=1}^n u_i \phi_i \in V^h \text{ such that}$$

such that

$$a(u^h, v^h) = F(v^h) \quad \forall u^h \in V^h$$

This is equivalent to the linear system $Au = f$, where $A_{ji} = a(\phi_i, \phi_j)$,

$$u = \begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix} \quad \text{and} \quad f = \begin{pmatrix} F(\phi_1) \\ \vdots \\ F(\phi_n) \end{pmatrix}$$

The matrix A is **symmetric, positive definite** and hence defines an inner product of

$$(u, v)_A = u^T A v \quad \text{and a corresponding norm } \|u\|_A^2 = u^T A u.$$

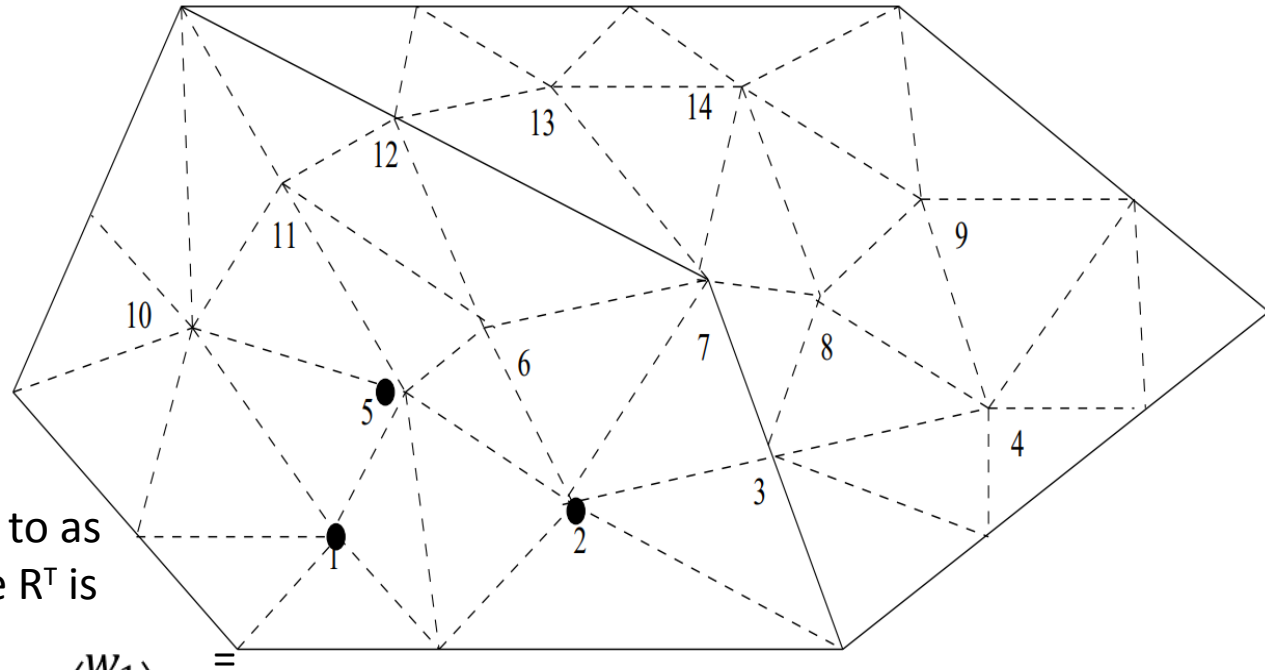
Classifications

- **Overlapping**
- **Nonoverlapping**

Classifications (Overlapping)

- ❑ Overlapping domain decomposition methods are efficient and flexible.
- ❑ It is also important that such methods are inherently suitable for parallel computing.

Classifications (Overlapping)



The matrix R is often referred to as the restriction operator, while R^T is the interpolation matrix.

$$R = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 1 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 1 & 0 & \dots \end{pmatrix} \Rightarrow \begin{pmatrix} w_1 \\ w_2 \\ 0 \\ 0 \\ w_3 \\ 0 \\ \dots \end{pmatrix} = R^T \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix}$$

Classifications (Overlapping)

Our “best” local correction is then defined by

$$\min_w \|u^* - (u^n + R^T w)\|_A^2$$

or, equivalently,

$$\min_w (u^* - (u^n + R^T w))^T A (u^* - (u^n + R^T w)).$$

(u^* is the exact solution to the linear system)

If we take the derivative with respect to the unknowns w ,

$$-RA(u^* - u^n - R^T w) = 0$$

or

$$RAR^T w = R(Au^* - Au^n).$$

Thus the “local” correction is given by

$$\begin{aligned} \text{Correction} &= R^T w \\ &= R^T (RAR^T)^{-1} R(f - Au^n). \end{aligned}$$

Where $Au = f$,

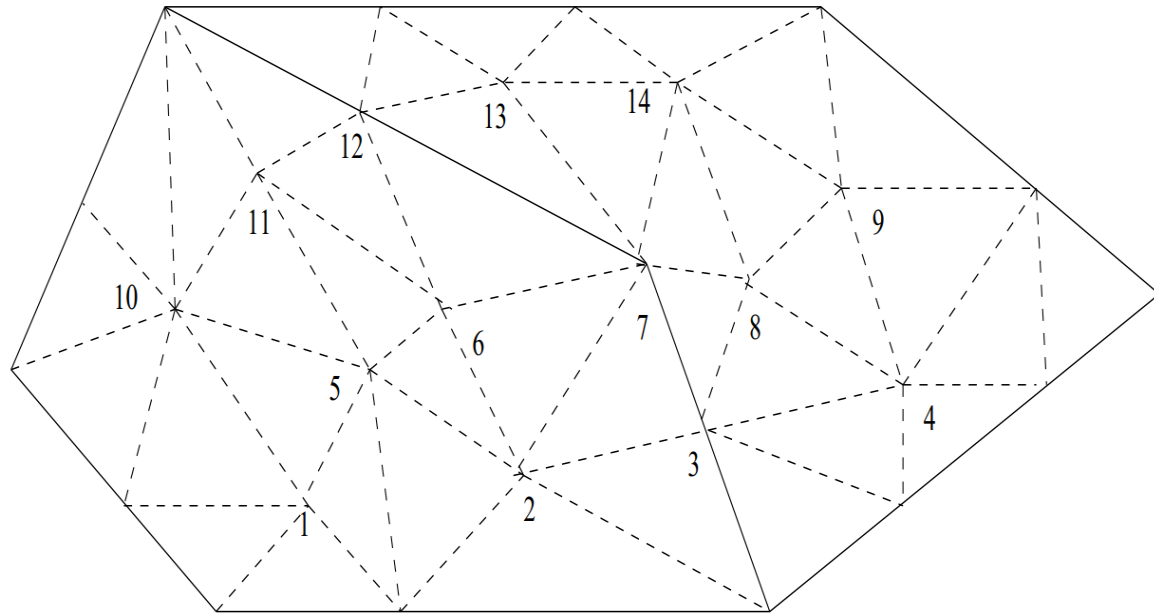
Classifications (Nonoverlapping)

- It may also be viewed as combining projections of the error onto subspaces of the solution space.

Classifications (Nonoverlapping)

We partition the unknowns into three sets:

- First domain (denoted by $\overline{u_1}$ and containing nodes 1, 2, 5, 6, 10, 11)
- Second domain, and those on the interface between the two domains (denoted by $\overline{u_3}$ and containing 3, 7, 12).



Then the linear system may be written as

$$\begin{pmatrix} A_1 & 0 & A_{13} \\ 0 & A_2 & A_{23} \\ A_{31} & A_{32} & A_3 \end{pmatrix} \begin{pmatrix} \overline{u_1} \\ \overline{u_2} \\ \overline{u_3} \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix} \text{ where } A_3 = A_3^{(1)} + A_3^{(2)}$$

Classifications

(Nonoverlapping)

- The key point is that nodes in the first subdomain are completely decoupled from nodes in the second subdomain.
- Once the unknowns in Ω_1 and Ω_2 have been eliminated, the resulting Schur complement system is given by

$$\begin{aligned} S\bar{u}_3 &= (S^{(1)} + S^{(2)}) \bar{u}_3 \\ &= (A_3^{(1)} + A_{31}A_1^{-1}A_{13} + A_3^{(2)} - A_{32}A_2^{-1}A_{23}) \bar{u}_3 \\ &= f_3 - A_{31}A_1^{-1}f_1 - A_{32}A_2^{-1}f_2 \\ &= g_3. \end{aligned}$$

Classifications (Nonoverlapping)

- ❑ Substructuring is an efficient, parallel direct method for the solution of linear systems arising from discretizations of PDEs based on Schur complements.
- ❑ In this particular example the elimination is done in three levels.
 - Level 1: The four groups 1, 2 and 3, 4 and 7 and 8, 9, 10 are eliminated in parallel.
 - Level 2: The two groups 5, 6 and 11 are eliminated in parallel.
 - Level 3: The final Schur complement involving 12, 13, 14 is eliminated.

Classifications (Nonoverlapping)

- **Once the factorization is complete, the unknowns may be calculated.**

- Level 3: Solve for u_{12} , u_{13} , u_{14} :
- Level 2: Solve for u_5 , u_6 and u_{11} in parallel.
- Level 1: Solve for u_1 , u_2 and u_3 , u_4 and u_7 and u_8 , u_9 , u_{10} in parallel.

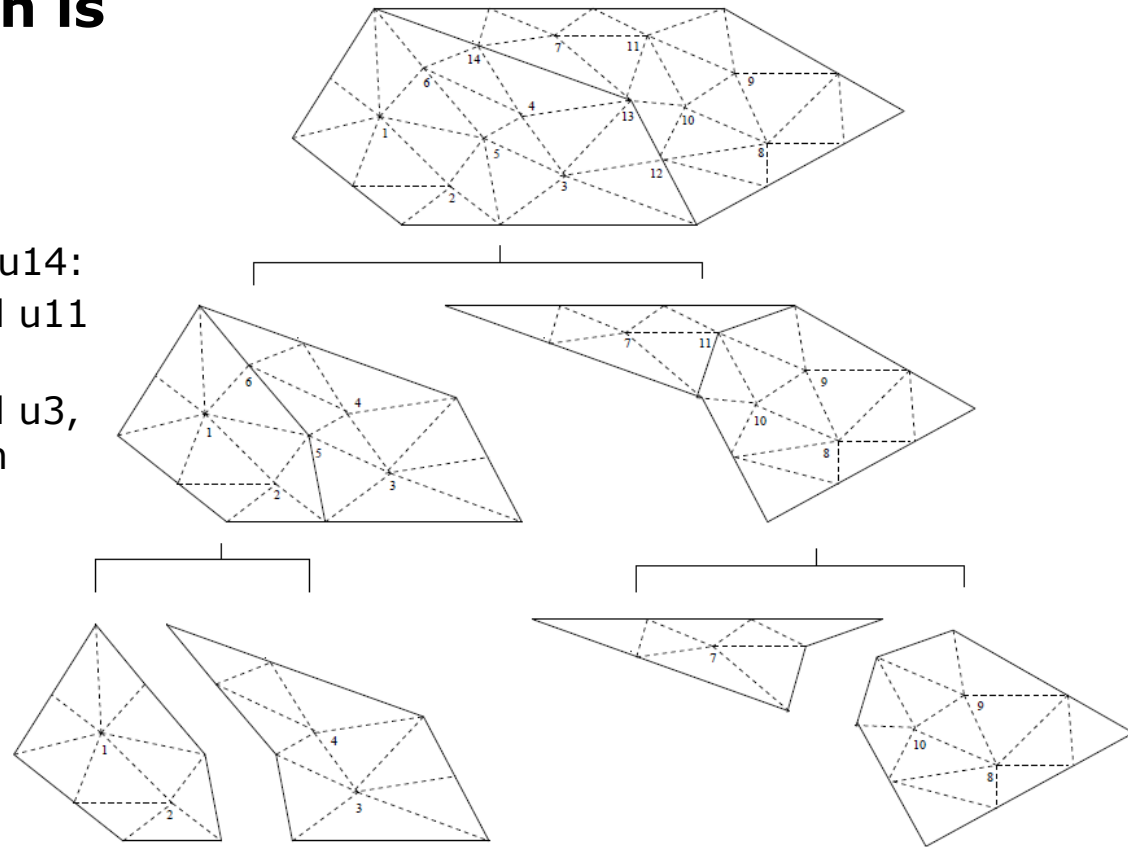



Figure 4: Example of substructuring

Finite Difference Methods

- 
- Brief History
 - Definition of FDM
 - Heat Equation
 - Solution Methods
 - Computational Overview

Brief History of FDM

- ❑ It was invented by a Chinese scientist named Feng Kang in the late 1950's
- ❑ Applied on computations of dam constructions
- ❑ It was also independently invented in the West, named in the West the finite element method

Definition

- Finite difference is defined as a math expression in the form of $f(x+b) - f(x+a)$.
- If a finite difference is divided by $(b-a)$
- This expression is similar to a differential quotient
- It uses finite quantities instead of infinitesimal ones.

Where it comes from ?

- ❏ To use FDM we need to use Taylor Series
- ❑ Forward Taylor series expansion;

$$f(x + \Delta x) = f(x) + f'(x)\Delta x + \frac{f''(x)\Delta x^2}{2!} + \dots$$

- ❑ Backward Taylor series expansion;

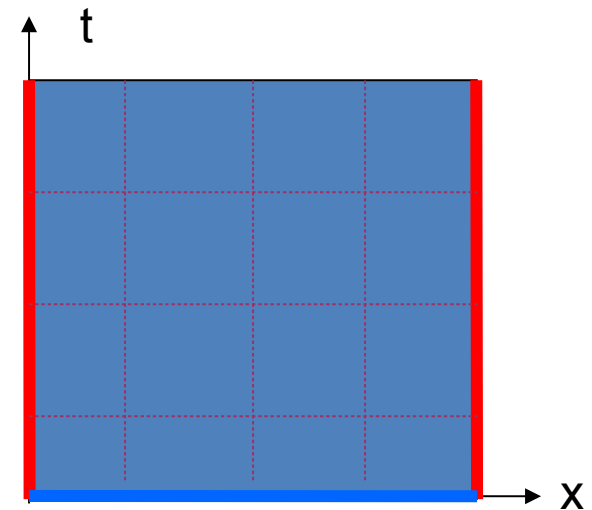
$$f(x - \Delta x) = f(x) - f'(x)\Delta x + \frac{f''(x)\Delta x^2}{2!} - \dots$$

- ❑ Add two equations and truncate after second derivative terms, we get;

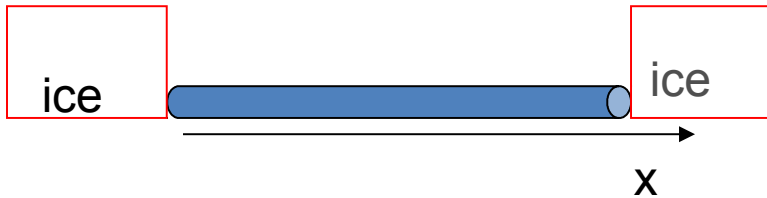
$$f''(x) = \frac{f(x - \Delta x) - 2f(x) + f(x + \Delta x)}{\Delta x^2}$$

Grid Representation

- Divide the interval x into sub-intervals, each of width h
- Divide the interval t into sub-intervals, each of width k
- A grid of points is used for the finite difference solution
- $T_{i,j}$ represents $T(x_i, t_j)$
- Replace the derivatives by finite-difference formulas



Heat Equation



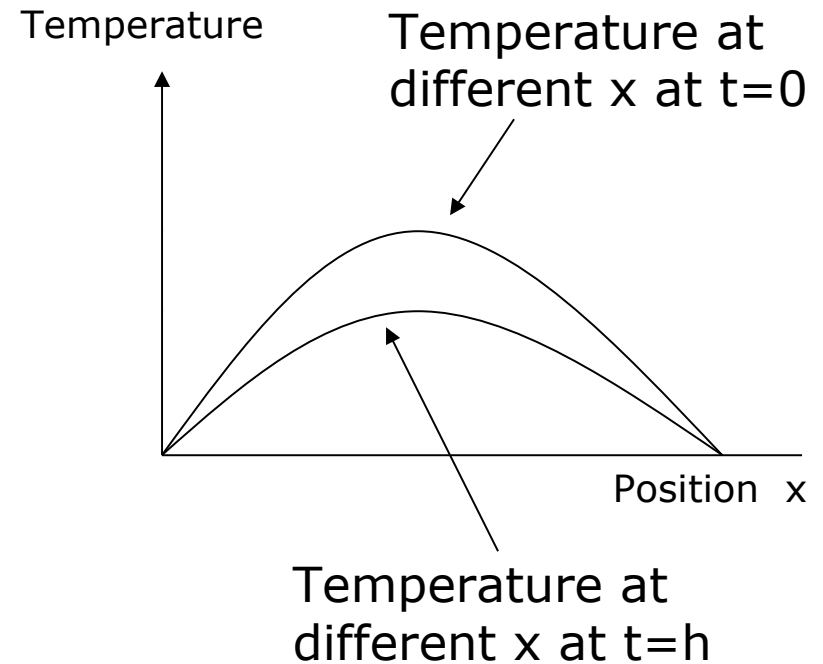
Thin metal rod insulated everywhere except at the edges. At $t = 0$ the rod is placed in ice

$$\frac{\partial^2 T(x,t)}{\partial x^2} - \frac{\partial T(x,t)}{\partial t} = 0$$

$$T(0,t) = T(1,t) = 0$$

$$T(x,0) = \sin(\pi x)$$

Different curve is used for each value of t



Boundary conditions

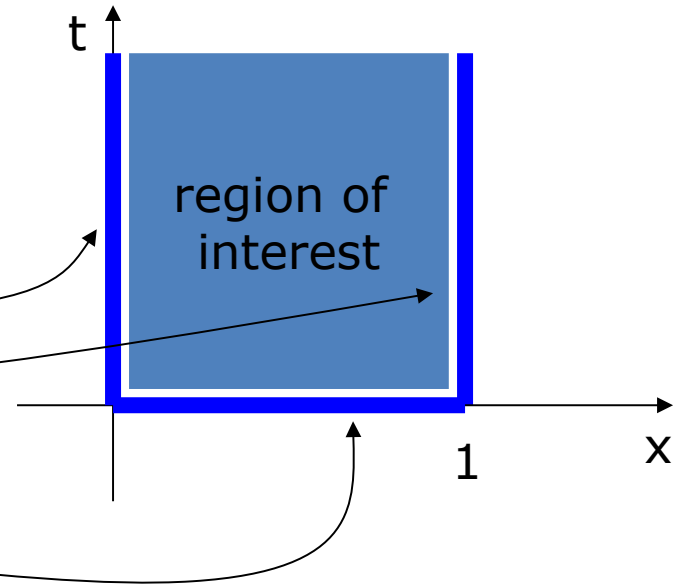
- To uniquely specify a solution to any PDE, a set of boundary conditions are needed.
- Both regular and irregular boundaries are possible.

Heat Equation : $\alpha \frac{\partial^2 u(x,t)}{\partial x^2} - \frac{\partial u(x,t)}{\partial t} = 0$

$u(0,t) = 0$

$u(1,t) = 0$

$u(x,0) = \sin(\pi x)$



Solution of the Heat Equation

- Two solutions to the Heat Equation will be presented:

1. Explicit Method: FTCS (Forward differencing in Time and Central differencing in Space at time level n)

Simple, Stability Problems.

2. Implicit Method: Crank-Nicolson

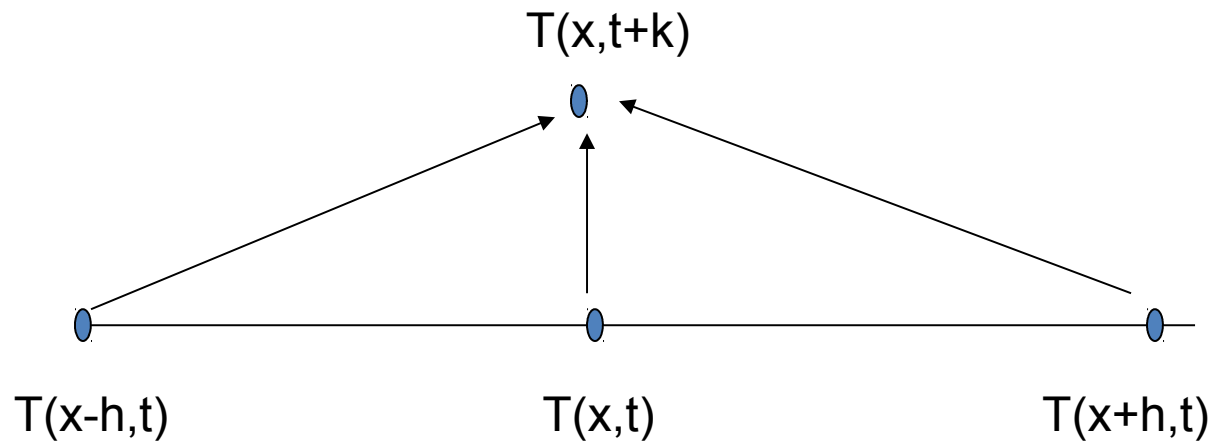
Involves the solution of a Tridiagonal system of equations, Stable.

Explicit Method

How Do We Compute?

$$T(x, t + k) = \lambda T(x - h, t) + (1 - 2\lambda) T(x, t) + \lambda T(x + h, t)$$

means



Explicit Method

How Do We Compute?

$$T(x, t + k) = \lambda T(x - h, t) + (1 - 2\lambda) T(x, t) + \lambda T(x + h, t)$$

means

There is only one unknown (*at new time level **t+k***) on the left hand side (LHS) of the difference equation

And it is computed in terms of all other terms on the RHS which are known (*at previous time level **t***).

Convergence and Stability

$T(x, t + k)$ can be computed directly using :

$$T(x, t + k) = \lambda T(x - h, t) + (1 - 2\lambda) T(x, t) + \lambda T(x + h, t)$$

Can be unstable (errors are magnified)

To guarantee stability, $(1 - 2\lambda) \geq 0 \Rightarrow \lambda \leq \frac{1}{2} \Rightarrow k \leq \frac{h^2}{2}$

This means that k is much smaller than h

This makes it slow.

Convergence and Stability of the Solution

□ Convergence

The solutions converge means that the solution obtained using the finite difference method approaches the true solution as the steps Δx and Δt approach zero.

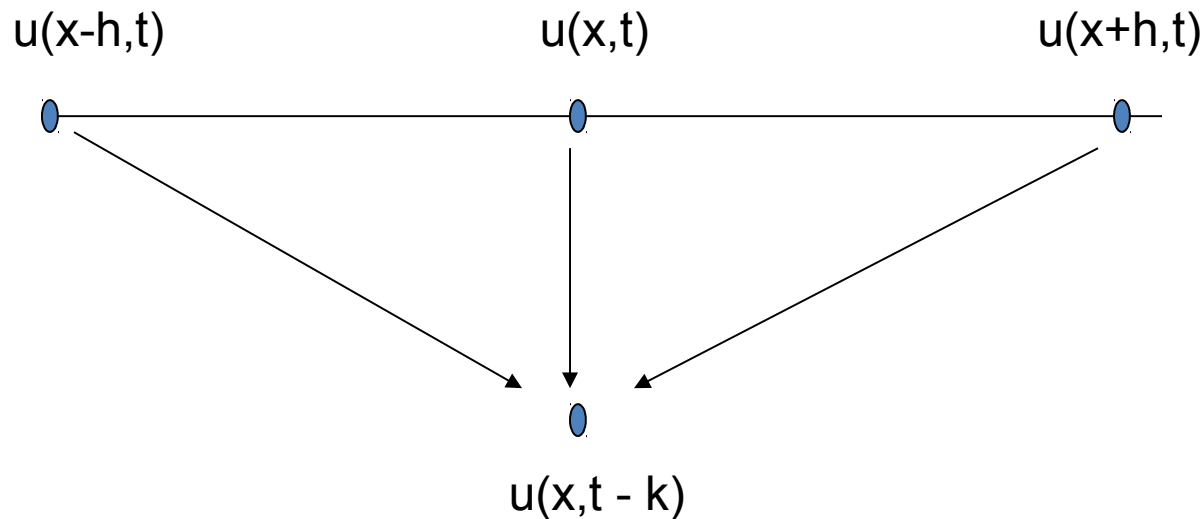
□ Stability:

An algorithm is stable if the errors at each stage of the computation are not magnified as the computation progresses.

Crank-Nicolson Method(Implicit)

Define $\lambda = \frac{k}{h^2}$ then Heat equation becomes :

$$-\lambda u(x-h, t) + (1 + 2\lambda) u(x, t) - \lambda u(x+h, t) = u(x, t-k)$$



Crank-Nicolson Method(Implicit)

Define $\lambda = \frac{k}{h^2}$ then Heat equation becomes :

$$-\lambda u(x-h, t) + (1 + 2\lambda) u(x, t) - \lambda u(x+h, t) = u(x, t-k)$$

There are multiple unknowns at **$t+k$** time level on the LHS of the equation
And the terms on RHS are known ones at **t** time level

Crank-Nicolson Method

The equation :

$$-\lambda u(x-h, t) + (1+2\lambda) u(x, t) - \lambda u(x+h, t) = u(x, t-k)$$

can be rewritten as :

$$-\lambda u_{i-1,j} + (1+2\lambda) u_{i,j} - \lambda u_{i+1,j} = u_{i,j-1}$$

and can be expanded as a system of equations (fix $j = 1$) :

$$-\lambda u_{0,1} + (1+2\lambda) u_{1,1} - \lambda u_{2,1} = u_{1,0}$$

$$-\lambda u_{1,1} + (1+2\lambda) u_{2,1} - \lambda u_{3,1} = u_{2,0}$$

$$-\lambda u_{2,1} + (1+2\lambda) u_{3,1} - \lambda u_{4,1} = u_{3,0}$$

$$-\lambda u_{3,1} + (1+2\lambda) u_{4,1} - \lambda u_{5,1} = u_{4,0}$$

Crank-Nicolson Method

The solution of the tridiagonal system produces :

The temperature values $u_{1,1}, u_{2,1}, u_{3,1}$, and $u_{4,1}$ at $t = t_0 + k$

To compute the temperature values at $t = t_0 + 2k$

Solve a second tridiagonal system of equations ($j = 2$)

$$\begin{bmatrix} 1+2\lambda & -\lambda & & \\ -\lambda & 1+2\lambda & -\lambda & \\ & -\lambda & 1+2\lambda & -\lambda \\ & & -\lambda & 1+2\lambda \end{bmatrix} \begin{bmatrix} u_{1,2} \\ u_{2,2} \\ u_{3,2} \\ u_{4,2} \end{bmatrix} = \begin{bmatrix} u_{1,1} + \lambda u_{0,2} \\ u_{2,1} \\ u_{3,1} \\ u_{4,1} + \lambda u_{5,2} \end{bmatrix}$$

To compute $u_{1,2}, u_{2,2}, u_{3,2}$, and $u_{4,2}$

Repeat the above step to compute temperature values at $t_0 + 3k$, etc.

Remarks

The Explicit Method:

- One needs to select small k to ensure **stability**.
- Computation per point is very simple but many points are needed.

Cranks Nicolson:

- Requires the solution of a **Tridiagonal** system.
- **Unconditionally stable**

Computational Overview (Explicit - FTCS)

```
% --- Define constants and initial condition
L = ...           % length of domain in x direction
tmax = ...        % end time
nx = ...          % number of nodes in x direction
nt = ...          % number of time steps
dx = L/(nx-1);
dt = tmax/(nt-1);
r = alpha*dt/dx^2;    r2 = 1 - 2*r;

% --- Loop over time steps
t = 0
u = ...           % initial condition
for m=1:nt
    uold = u;      % prepare for next step
    t = t + dt;
    for i=2:nx-1
        u(i) = r*uold(i-1) + r2*uold(i) + r*uold(i+1);
    end
end
```

Implementation of the FTCS scheme. u and $uold$ variables are vectors. $O(n^2)$

Computational Overview (Implicit - TDMA)

```
program TDMA
implicit doubleprecision(a-h,o-z)
parameter (nd = 100)
doubleprecision A(nd), B(nd), C(nd), D(nd), X(nd), P(0:nd), Q(0:nd)
```

```
A(1) = 0
C(n) = 0
```

n is the number of
unknowns

```
forward elimination
do i = 1, n
    denom = B(i) + A(i)*P(i-1)
    P(i) = -C(i) /denom
    Q(i) = (D(i) - A(i)*Q(i-1)) /denom
enddo
```

```
back substitution
do i = n, 1, -1
    X(i) = P(i)*X(i+1) + Q(i)
enddo
stop
end
```

Implementation of the Tridiagonal Matrix Algorithm. $O(n)$

Computational Overview


(Implicit - CrankNicolson)

```
% --- Coefficients of the tridiagonal system
a = (-alpha/dx^2)*ones(nx,1);      % subdiagonal a: coefficients of phi(i-1)
c = a;                             % superdiagonal c: coefficients of phi(i+1)
b = (1/dt)*ones(nx,1) - 2*a;       % diagonal b: coefficients of phi(i)
b(1) = 1;      c(1) = 0;           % Fix coefficients of boundary nodes
b(end) = 1;    a(end) = 0;
[e,f] = tridiagLU(a,b,c);          % Get LU factorization of coefficient matrix

% --- Loop over time steps
for m=2:nt
    d = U(:,m-1)/dt;                % update right hand side
    d(1) = u0;  d(end) = uL;        % overwrite BC values
    U(:,m) = tridiagLUSolve(d,a,e,f,U(:,m-1)); % solve the system
end
```

Implementation of the Crank – Nicolson : $O(n^2)$

Finite Volume Method

- 
- Definition of FVM and Features
 - Flavors of FVM
 - FVM Formulation – Cell Centered Method

Finite Volume Method

- Elliptic, parabolic or hyperbolic, for instance
- Used in several engineering fields, such as fluid mechanics, heat and mass transfer or petroleum engineering
- Some of the important features of FVM are similar to those of the FEM

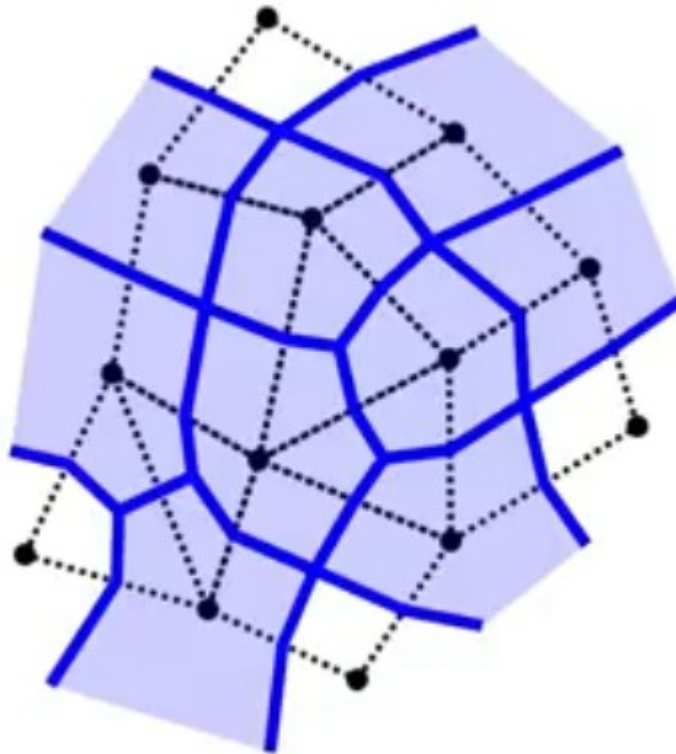
Finite Volume Method

- Extensible to unstructured mesh / multiple dimensions
- Physical intuition
- Conservative locally as well as globally
- A lot of Fluid dynamics and Heat Transfer software written in FVM

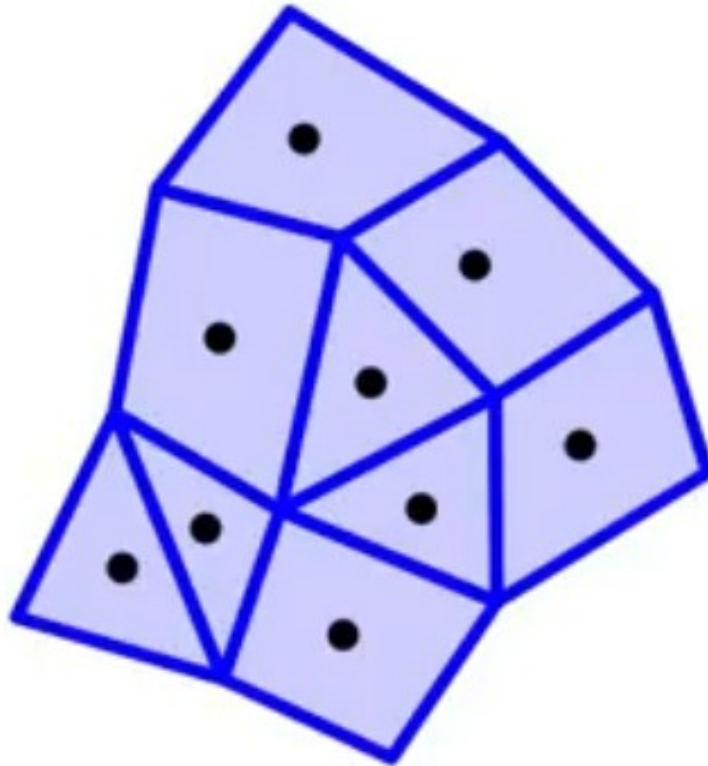
Two flavors of FVM

- Vertex Centered
- Cell Centered

Vertex Centered



Cell Centered



Finite Volume Formulation

$$\frac{\partial \phi}{\partial t} + \frac{\partial f}{\partial x} + \frac{\partial g}{\partial y} = s$$

Integrating over the cell area:

$$\iint_{\Omega} \frac{\partial \phi}{\partial t} dA + \iint_{\Omega} \left(\frac{\partial f}{\partial x} + \frac{\partial g}{\partial y} \right) dA = \iint_{\Omega} s dA$$

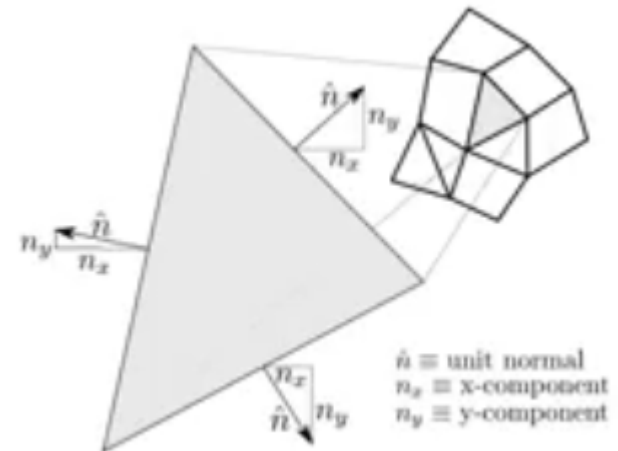
Using average and Gauss-Divergence:

$$A \frac{\partial \bar{\phi}}{\partial t} + \oint_{\Gamma} (f n_x + g n_y) dL = \bar{s} A$$

$$A \frac{\partial \bar{\phi}}{\partial t} + \sum_{\text{faces}} \int_{\Gamma_{\text{face}}} (f n_x + g n_y) dL = \bar{s} A$$

$$A \frac{\partial \bar{\phi}}{\partial t} + \sum_{\text{faces}} (\bar{f} n_x + \bar{g} n_y) \Delta L = \bar{s} A$$

$$\text{where, } \bar{\phi} = \frac{1}{A} \iint_{\Omega} \phi dA; \quad \bar{s} = \frac{1}{A} \iint_{\Omega} s dA$$



Finite Volume Formulation

2D advection equation with uniform velocity field

$$\frac{\partial \phi}{\partial t} + a_x \frac{\partial \phi}{\partial x} + a_y \frac{\partial \phi}{\partial y} = 0; \quad \phi = \phi(x, y, t);$$

- Initial Conditions

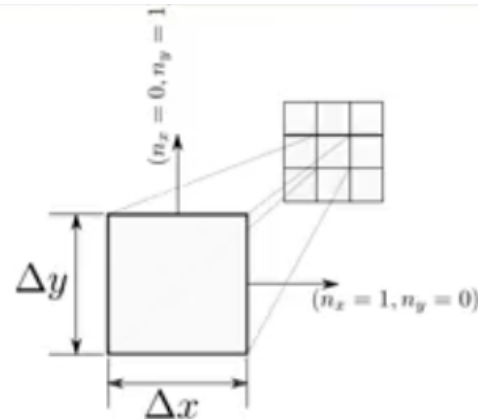
$$\phi(x, y, t = 0) = \phi_0$$

- Boundary Conditions

Finite Volume Formulation

$$\frac{\partial \phi}{\partial t} + a_x \frac{\partial \phi}{\partial x} + a_y \frac{\partial \phi}{\partial y} = 0$$

$$\frac{\partial \phi}{\partial t} + \frac{\partial f}{\partial x} + \frac{\partial g}{\partial y} = 0; \quad f = a_x \phi, \quad g = a_y \phi$$



Integrating over cell area:

$$\iint_{\Omega} \frac{\partial \phi}{\partial t} dx dy + \iint_{\Omega} \left(\frac{\partial f}{\partial x} + \frac{\partial g}{\partial y} \right) dx dy = 0$$

$$\Delta x \Delta y \frac{\partial \bar{\phi}}{\partial t} + \oint_{\Gamma} (f n_x + g n_y) dL = 0 \quad (\text{using Gauss-Divergence})$$

$$\text{where, } \bar{\phi} = \frac{1}{\Delta x \Delta y} \iint_{\Omega} \phi dx dy$$

Finite Volume Formulation

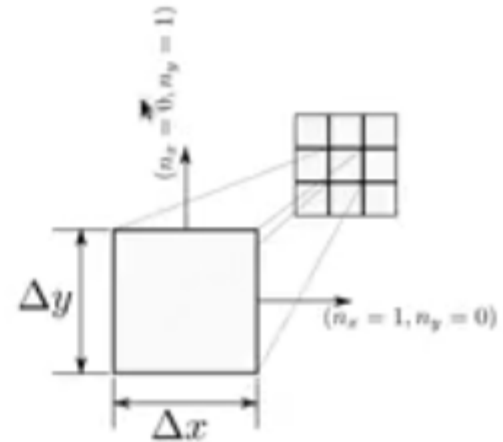
$$\Delta x \Delta y \frac{\partial \bar{\phi}}{\partial t} + \oint_{\Gamma} (f n_x + g n_y) dL = 0$$

$$\Delta x \Delta y \frac{\partial \bar{\phi}}{\partial t} = - \oint_{\Gamma} \begin{cases} f n_x dL & \text{for vertical faces} \\ g n_y dL & \text{for horizontal faces} \end{cases}$$

$$\Delta x \Delta y \frac{d\bar{\phi}}{dt} = - \begin{cases} \sum \bar{f} n_x \Delta y & \text{for vertical faces} \\ \sum \bar{g} n_y \Delta x & \text{for horizontal faces} \end{cases}$$

$$\frac{d\bar{\phi}}{dt} = - \frac{1}{\Delta x \Delta y} \begin{cases} (\bar{f}_{i+1/2} - \bar{f}_{i-1/2}) \Delta y \\ (\bar{g}_{j+1/2} - \bar{g}_{j-1/2}) \Delta x \end{cases}$$

$$\boxed{\frac{d\bar{\phi}}{dt} = - \frac{1}{\Delta x} (\bar{f}_{i+1/2} - \bar{f}_{i-1/2}) - \frac{1}{\Delta y} (\bar{g}_{j+1/2} - \bar{g}_{j-1/2})}$$



Conclusion

- Partial Differential Equations (PDE)
- Descriptions and fundamental topics of the **FEM, FDM, FVM** and **DDM**
- We **not only** read the some description papers but also **tried to understand what fundamentals topics** of them.
- We did **not mention any of the mechanics or industrial concepts**
- Because as computer engineer graduate students many of us **knows about their importance** in the field.
- Also we want to present **some pure math and its beauty** to our colleagues.