# On the Discretization Methods for Partial Differential Equations

# Finite Element Method (FEM) , Finite Difference Method (FDM) , Finite Volume Method (FVM) , Domain Decomposition Method (DDM)

##### **Ali Anıl APAYDIN\*, Caner ASLAN\*, Mehmet KOCABAŞ\*, Uğur ENİŞ\*\***

\* GTU, Department of Computer Engineering

\*\*GTU, Department of Electronical Engineering

***Abstract*-** This report explores the discretization methods of the Partial Diffential Equations (PDE). The report includes descriptions and fundamental topics of the Finite Element Method (FEM), Finite Difference Method (FDM), Finite Volume Method (FVM) and Domain Decomposition Method (DDM).

1. Introductıon

The partial differential equation we will focus on is a scalar equation that represents the transport of a substance under the influence of advection by the air flow and mixing. The transport equation is frequently written in the advective form:

where T is the subtance transported, e.g. temperature, humidity or a pollutant concentration, is the velocity field presumed known, and α is the diffusion coefficient and which can represent either molecular diffusion or eddy mixing.

The velocity field cannot be arbitrary and must satisfy some sort of mass conservation equation. Here we will assume the flow to be incompressible so that its mass conservation equation reduces to:

The advective form can be interpreted as the time evolution of the T field along characteristic lines given by . It is thus closest to a Lagrangian description of the flow where one follows individual particles. In the Eulerian frame, however, another important issues is the conservation of the tracer T for long period of times. This stems not only from physical considerations but also for the need to account for the sources and sinks of T in long calculations. It is imperative that the discretization does not introduce spurios sources (this the prime imperative in climate models for example). A slightly different form of the equation called the conservative form can be derived and forms the starting point for the derivation of finite volume methods. Multiplying the continuity equation by *T*, adding it to the resultant equations to the advective form, and recalling that

We can derive the conservative form of the transport diffusion equation:

1. Fınıte Element Method (FEM)

Consider a single linear partial differential equation in http://reference.wolfram.com/language/FEMDocumentation/tutorial/Files/SolvingPDEwithFEM.en/1.png:

http://reference.wolfram.com/language/FEMDocumentation/tutorial/Files/SolvingPDEwithFEM.en/2.gif

The PDE is defined in http://reference.wolfram.com/language/FEMDocumentation/tutorial/Files/SolvingPDEwithFEM.en/3.png. Here http://reference.wolfram.com/language/FEMDocumentation/tutorial/Files/SolvingPDEwithFEM.en/4.png is the dependent variable for which a solution is search. The coefficients http://reference.wolfram.com/language/FEMDocumentation/tutorial/Files/SolvingPDEwithFEM.en/5.png, http://reference.wolfram.com/language/FEMDocumentation/tutorial/Files/SolvingPDEwithFEM.en/6.png, http://reference.wolfram.com/language/FEMDocumentation/tutorial/Files/SolvingPDEwithFEM.en/7.png and http://reference.wolfram.com/language/FEMDocumentation/tutorial/Files/SolvingPDEwithFEM.en/8.pngare scalars; http://reference.wolfram.com/language/FEMDocumentation/tutorial/Files/SolvingPDEwithFEM.en/9.png, http://reference.wolfram.com/language/FEMDocumentation/tutorial/Files/SolvingPDEwithFEM.en/10.png and http://reference.wolfram.com/language/FEMDocumentation/tutorial/Files/SolvingPDEwithFEM.en/11.png are vectors; and http://reference.wolfram.com/language/FEMDocumentation/tutorial/Files/SolvingPDEwithFEM.en/12.png is an http://reference.wolfram.com/language/FEMDocumentation/tutorial/Files/SolvingPDEwithFEM.en/13.png×http://reference.wolfram.com/language/FEMDocumentation/tutorial/Files/SolvingPDEwithFEM.en/14.png matrix.

What follows are some well-known PDEs and their corresponding coefficients. To illustrate the generality of (1), the components that are relevant to a specific equation are black, while the non-relevant components are gray.

The Laplace equation simply contains a diffusive term:

http://reference.wolfram.com/language/FEMDocumentation/tutorial/Files/SolvingPDEwithFEM.en/15.gif

To model Poisson's equation, only a small modification is needed; add a load term http://reference.wolfram.com/language/FEMDocumentation/tutorial/Files/SolvingPDEwithFEM.en/16.png:

http://reference.wolfram.com/language/FEMDocumentation/tutorial/Files/SolvingPDEwithFEM.en/17.gif

Helmholtz's equation adds a reaction term http://reference.wolfram.com/language/FEMDocumentation/tutorial/Files/SolvingPDEwithFEM.en/18.png:

http://reference.wolfram.com/language/FEMDocumentation/tutorial/Files/SolvingPDEwithFEM.en/19.gif

Convection-diffusion-reaction type equations are another common class of PDEs. Compared to the previous examples, these have an additional convection term http://reference.wolfram.com/language/FEMDocumentation/tutorial/Files/SolvingPDEwithFEM.en/20.png:

http://reference.wolfram.com/language/FEMDocumentation/tutorial/Files/SolvingPDEwithFEM.en/21.gif

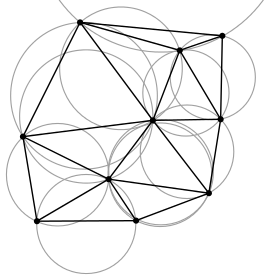
The PDEs considered so far are stationary, i.e. they have no time dependence. The heat equation adds time dependence to the Poisson equation. It has the following form:

http://reference.wolfram.com/language/FEMDocumentation/tutorial/Files/SolvingPDEwithFEM.en/22.gif

Similarly, the wave equation is given as:

http://reference.wolfram.com/language/FEMDocumentation/tutorial/Files/SolvingPDEwithFEM.en/23.gif

**Discretization of domain:** A Delaunay triangulation in the plane with circumcircles shown in [mathematics](https://en.wikipedia.org/wiki/Mathematics) and [computational geometry](https://en.wikipedia.org/wiki/Computational_geometry), a Delaunay triangulation (also known as a Delone triangulation) for a given set P of [discrete points](https://en.wikipedia.org/wiki/Isolated_point) in a plane is a [triangulation](https://en.wikipedia.org/wiki/Triangulation_(geometry)) DT(P) such that no point in P is inside the [circumcircle](file:///C:\Users\MAHREK\Desktop\ALGO%20Course\Docs\Circumcircle) of any [triangle](https://en.wikipedia.org/wiki/Triangle) in DT(P). Delaunay triangulations maximize the minimum angle of all the angles of the triangles in the triangulation; they tend to avoid [sliver triangles](https://en.wikipedia.org/wiki/Sliver_triangle). The triangulation is named after [Boris Delaunay](https://en.wikipedia.org/wiki/Boris_Delaunay) for his work on this topic from 1934.



For a set of points on the same line there is no Delaunay triangulation (the notion of triangulation is degenerate for this case). For four or more points on the same circle (e.g., the vertices of a rectangle) the Delaunay triangulation is not unique: each of the two possible triangulations that split the [quadrangle](https://en.wikipedia.org/wiki/Quadrilateral) into two triangles satisfies the "Delaunay condition", i.e., the requirement that the circumcircles of all triangles have empty interiors.

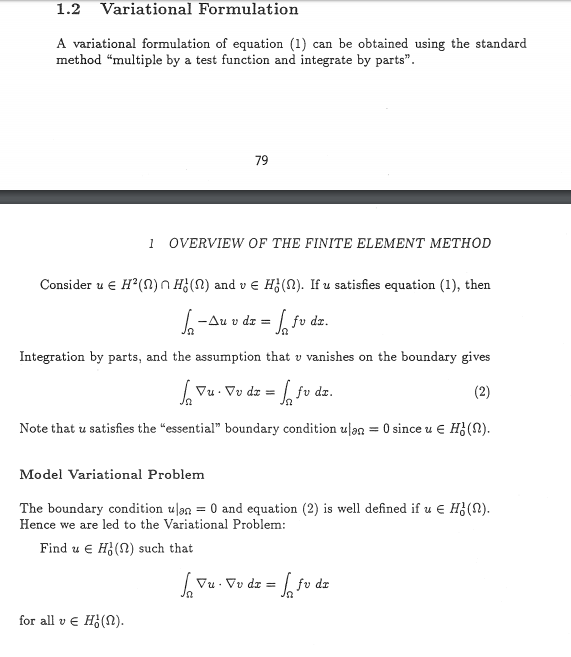
By considering circumscribed spheres, the notion of Delaunay triangulation extends to three and higher dimensions. Generalizations are possible to [metrics](https://en.wikipedia.org/wiki/Metric_(mathematics)) other than [Euclidean distance](https://en.wikipedia.org/wiki/Euclidean_distance). However, in these cases a Delaunay triangulation is not guaranteed to exist or be unique.

Each frame of the animation shows a Delaunay triangulation of the four points. Halfway through, the triangulating edge flips showing that the Delaunay triangulation maximizes the minimum angle, not the edge-length of the triangles.

Let *n* be the number of points and *d* the number of dimensions.

* The union of all simplices in the triangulation is the convex hull of the points.
* The Delaunay triangulation contains *O*(*n*⌈*d*/2⌉) simplices.
* In the plane (*d* = 2), if there are *b* vertices on the convex hull, then any triangulation of the points has at most 2*n*−2−*b* triangles, plus one exterior face (see [Euler characteristic](https://en.wikipedia.org/wiki/Euler_characteristic)).
* If points are distributed according to a [Poisson process](https://en.wikipedia.org/wiki/Poisson_process) in the plane with constant intensity, then each vertex has on average six surrounding triangles. More generally for the same process in *d* dimensions the average number of neighbors is a constant depending only on *d*.
* In the plane, the Delaunay triangulation maximizes the minimum angle. Compared to any other triangulation of the points, the smallest angle in the Delaunay triangulation is at least as large as the smallest angle in any other. However, the Delaunay triangulation does not necessarily minimize the maximum angle. The Delaunay triangulation also does not necessarily minimize the length of the edges.
* A circle circumscribing any Delaunay triangle does not contain any other input points in its interior.
* If a circle passing through two of the input points doesn't contain any other of them in its interior, then the segment connecting the two points is an edge of a Delaunay triangulation of the given points.
* Each triangle of the Delaunay triangulation of a set of points in *d*-dimensional spaces corresponds to a facet of [convex hull](https://en.wikipedia.org/wiki/Convex_hull) of the projection of the points onto a (*d* + 1)-dimensional [paraboloid](https://en.wikipedia.org/wiki/Paraboloid), and vice versa.
* The closest neighbor *b* to any point *p* is on an edge *bp* in the Delaunay triangulation since the [nearest neighbor graph](https://en.wikipedia.org/wiki/Nearest_neighbor_graph) is a subgraph of the Delaunay triangulation.
* The Delaunay triangulation is a [geometric spanner](https://en.wikipedia.org/wiki/Geometric_spanner): the shortest path between two vertices, along Delaunay edges, is known to be no longer than times the Euclidean distance between them.

**Deriving finite element equations:**



Assemble of finite elements: The assembling procedure from a local element matrix $ \mathbf{A^e}$ to a global matrix $ \mathbf{A^g}$ has the same routines for two- and three-dimensional structures. The dimension of the local matrix $ \mathbf{A^e}$ is always $ n k \times n k$, where$ n$ is the number of grid nodes on the finite element ($ n=3$ for triangles and $ n=4$ for tetrahedrons) and $ k$ is the number of unknown variables on a grid node. The dimension of the global matrix $ \mathbf{A^g}$ is always $ N k\times N k$, where $ N$ is the total number of grid nodes in the discretized domain.   
If it is assumed that there is only one sought variable $ \varphi$

$\displaystyle \mathbf{A^e}   \varphi^e = b^e, \qquad \mathbf{A^g}   \varphi^g = b^g$

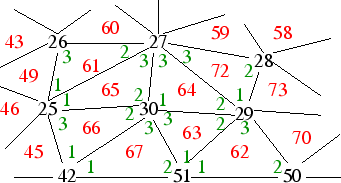
the dimension of $ \mathbf{A^e}$ is $ 3\times 3$ in the two-dimensional case and the dimension of $ \mathbf{A^g}$ is $ N\times N$.

The local matrix uses the local node indexes which are 1, 2, and 3 for every finite element. The global indexes for these grid nodes are different. There must exists a transformation $ T(\mathbf{A^e})$ which projects the local indexes $ k$ and $ l$ of the components $ A^e_{kl}$ to the global indexes $ i$ and $ j$. For example, the element 66 ,with its local nodes 1, 2, and 3 has the global nodes 42, 30, and 25 and the index transformation is

$\displaystyle 1 \to 42, \qquad 2 \to 30, \qquad \mathrm{and} \qquad 3 \to 25.$

This means that the components of the local matrix $ \mathbf{A^{66}}$from element 66 are transformed to the global matrix $ \mathbf{A^g}$ in the way

$\displaystyle A^{66}_{1,1} \to A^{g}_{42,42}, \qquad A^{66}_{1,2} \to A^{g}_{42...
...66}_{1,3} \to A^{g}_{42,25}, \qquad A^{66}_{2,1} \to A^{g}_{30,42}, \quad\ldots$

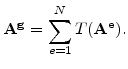


Part of a mesh with finite elements and grid nodes

Another important aspect is that a global grid node is shared by a number of different finite elements. This fact is taken into account during assembling of the global matrix by the so-called superposition principle. This means that the components $ A^g_{ij}$ of the global matrix are summed up from the contributions $ A^e_{kl}$ of the local element matrices. For example, the global grid node 30 is shared by the five elements 63, 64, 65, 66, and 67 , which all make a contribution to the node 30. So the global matrix components $ A^g_{30,j}$ are found with the help of index transformations by the way

$\displaystyle A^g_{30,25}= A^{65}_{2,1}+ A^{66}_{2,3}, \qquad A^g_{30,29}= A^{63}_{3,2}+ A^{64}_{1,2}, \quad\ldots$

The assembling of the global matrix from all $ N$ elements with the index transformation $ T(\mathbf{A^e})$ can be described in the form.



Sometimes there are more than one variable on the grid nodes. For example, with two variables the size of the local matrix $ \mathbf{A^e}$(two dimensions) is $ 6\times 6$ and $ 2N\times 2N$ for the global matrix $ \mathbf{A^g}$. If the variables are independent, the offset of the entries for the second variable is 3 in $ \mathbf{A^e}$ and $ N$ in $ \mathbf{A^g}$. For assembling the second variable from the element 66 the index transformation must be modified by adding the respective offset.

$\displaystyle A^{66}_{3+1,3+1} \to A^{g}_{N+42,N+42}, \quad A^{66}_{3+1,3+2} \to A^{g}_{N+42,N+30}, \quad A^{66}_{3+1,3+3} \to A^{g}_{N+42,N+25}, \ldots$

Boundry Conditions: Through the Dirichlet boundary conditions the values on the surface grid nodes are already fixed with the so-called Dirichlet value. Therefore, it is not necessary and even not allowed to recalculate the values on these grid nodes from the global equation system$ \mathbf{A^g}   \varphi^g = b^g$, because it is impossible to obtain the same Dirichlet values by solving the equation system. These surface grid nodes must be treated differently with the Dirichlet value. If on the global node $ i$ there is a Dirichlet boundary value $ \varphi_i = C_i$, the global equation system must be changed to

$\displaystyle \left[ \begin{array}{ccccccccc} a_{1,1} &a_{1,2} &\cdots &a_{1,i-...
... b_2 \vdots b_{i-1} C_{i} b_{i+1} \vdots b_{N} \end{array} \right]$

From the mathematical point of view the global equation system $ \mathbf{A^g}   \varphi^g = b^g$ has $ m$ pseudo-equations if there are $ m$ grid nodes with Dirichlet conditions, after setting all $ m$ rows and columns from Dirichlet grid nodes $ \varphi_i$ in $ \mathbf{A^g}$ to 0.

In practice it is more comfortable to multiplicate $ \mathbf{A^g}$ with a transformation matrix $ \mathbf{T_b}$

$\displaystyle \mathbf{T_b} \mathbf{A^g}   \varphi^g = b^g,$

which sets all rows and columns for the $ m$ Dirichlet grid nodes $ \varphi_i = C_i$ in $ \mathbf{A^g}$ to 0, instead of doing it componentwise by $ A^g_{ik}=0$ and $ A^g_{ki}=0$ for $ k=1\ldots N$. In the beginning $ \mathbf{T_b}$ is a unit matrix ($ T_{ii}=1$ and $ T_{ij}=0$), but for every Dirichlet grid node $ i$ the components $ T_{ii}$ are reset to $ T_{ii}=0$. Therefore, all $ m$ rows and columns in $ \mathbf{A^g}$ can easily be set to 0 at once with $ \mathbf{T_b}$.

1. Fınıte Dıfference Method (FDM)

**A Brief History of the finite Difference Method:**

The finite difference method was invented by a Chinese scientist named Feng Kang in the late 1950’s. He proposed the finite difference method as a systematic numerical method

for solving partial differential equations that are applied to the computations of dam constructions. It is speculated that the same method was also independently invented in the West, named in the West the finite element method. It is now considered that the invention of the finite difference method is a milestone of computational mathematics.[2]

**Definition and Description of the Finite Difference Methods:**

In a simplistic way, a finite difference is defined as a mathematical expression of the form f(x+b)−f(x+a). If a finite difference is divided by (b-a), one gets an expression

similar to a differential quotient, except that it uses finite

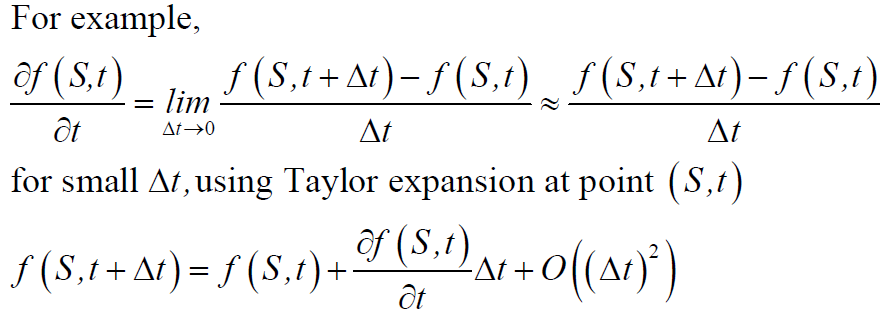
quantities instead of infinitesimal ones. The approximation of derivatives by finite differences plays a central role in finite difference methods for the numerical solution of partial differential equations.

In solving partial differential equations, the primary challenge is to create an equation which approximates the equation to be studied, but which is numerically stable, meaning

that errors in the input data and intermediate calculations do not accumulate and cause the resulting output to be meaningless.

**Finite Difference Approximations**

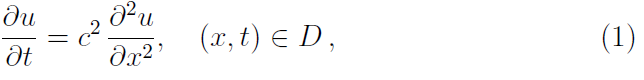
The basic idea of FDM is to replace the partial derivatives by approximations obtained by Taylor expansions near the point of interests.



The finite difference method applies three main difference schemes: The forward difference, the central difference, and the backward difference.

I can explain this schemes by an example :

**Diffusion Equations of One State Variable**



where t is a time variable, x is a state variable, and u(x, t) is an unknown function satisfying the equation.

To find well-defined solution, we need to impose the initial condition



and, if D = [a, b] x [0 , ∞] , the boundary conditions



where , , , are continuous functions.

If D = (-∞, ∞) x ( 0, ∞), we need to impose the boundary conditions



(4) implies u(x, t) does not grow too fast as |x| ∞.

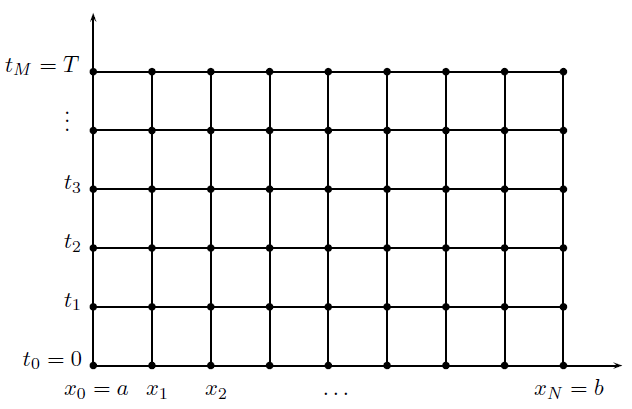
The diffusion equation (1) with the initial condition (2) and the boundary conditions (3) is well-posed, i.e. there exists a unique solution that depends continuously on , , .

**Grid Points :**

To find a numerical solution to equation (1) with finite difference methods, we first need to define a set of grid points in the domain D as follows:

Choose a state step size Δx = b−a / N (N is an integer) and a time step size Δt, draw a set of horizontal and vertical lines across D, and get all intersection points (xj , tn), or simply (j,n), where xj = a + j Δx, j = 0, . . . ,N, and tn = nΔt, n = 0, 1, . .

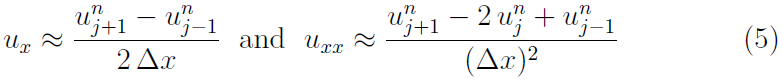
If D = [a, b]×[0, T] then choose Δt = T / M (M is an integer) and tn = nΔt, n = 0, . . . ,M.



**Finite Differences:**

The partial derivatives and

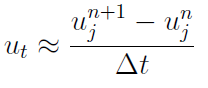
are always approximated by central difference quotients,i.e.



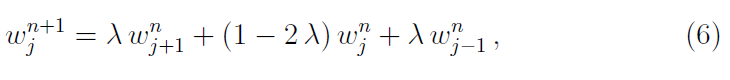
Depending on how ut is approximated, we have three basic schemes: explicit, implicit, and Crank–Nicolson schemes.

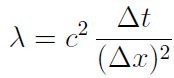
**Explicit Scheme**

If is approximated by a forward difference quotient

 at (j, n),

then the corresponding difference equation to (1) at grid point (j, n) is



where 

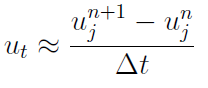
The initial condition is 

and the boundary conditions are 

The difference equations (6), j = 1, … , N – 1, can be solved explicitly.

**Implicit Scheme**

If is approximated by a backward difference quotient



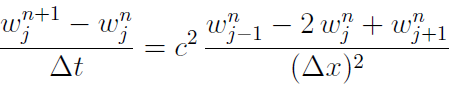
at (j, n+1),

then the corresponding difference equation to (1) at grid point (j, n+1) is



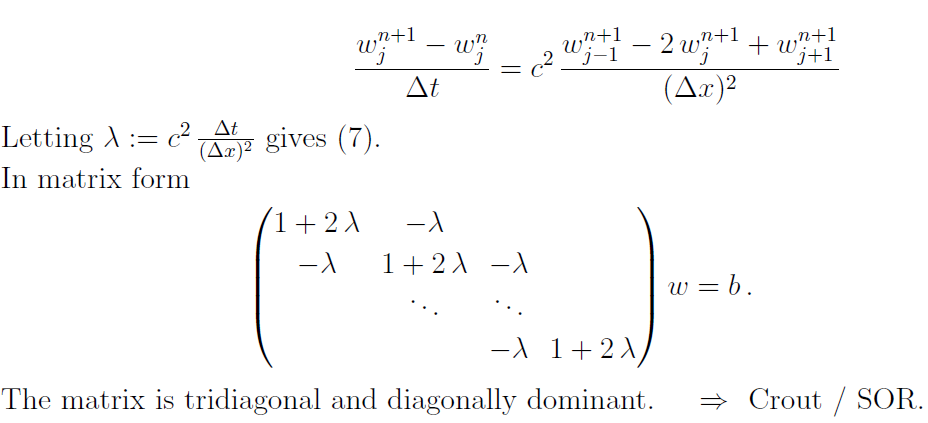
The difference equations (7), j = 1,2 … N – 1, together with the initial and boundary conditons as before, can be solved the Crout algorithm or the SOR algorithm.

Explicit Method



Letting 

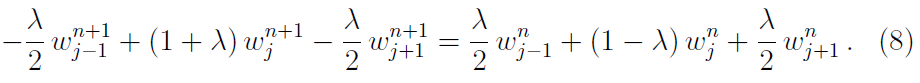
Implicit Method



**Crank–Nicolson Scheme.**

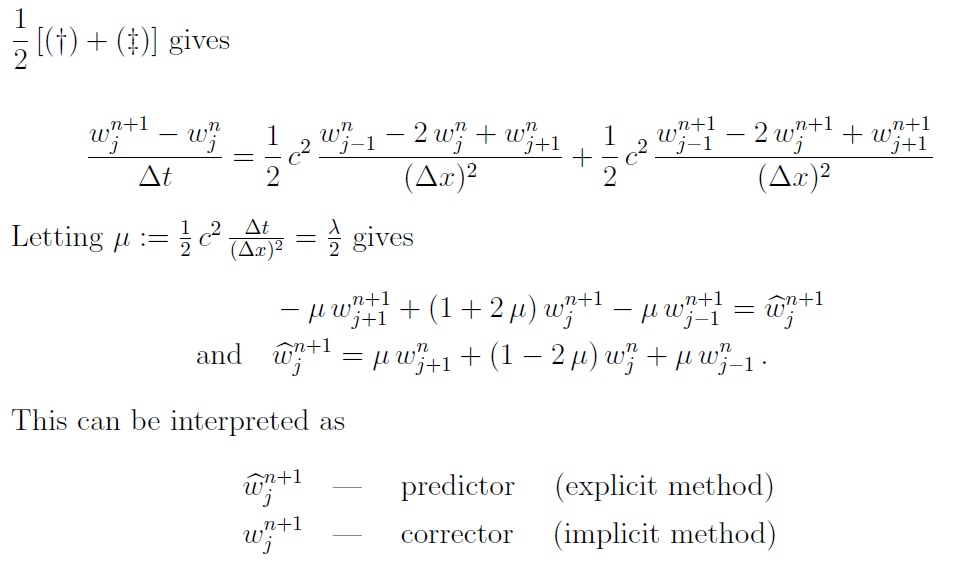
The Crank-Nicolson scheme is the average of the explicit scheme at ( j, n) and the implicit scheme at ( j, n+1).

The resulting difference equation is



The difference equations (8), j = 1,.. N – 1, together with the initial and boundary conditions as before, can be solved using Crout algorithm or SOR algorithm.

Crank-Nicolson



**Local Truncation Errors**

These are measures of the error by which the exact solution of a differential equation does not satisfy the difference equation at the grid points and are btained by substituting the exact solution of the continuous problem into the numerical scheme.

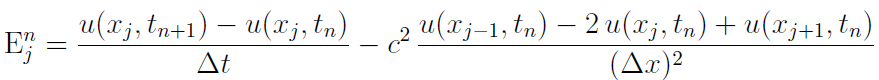
A necessary condition for the convergence of the numerical solutions to the continuous solution is that the local truncation error tends to zero as the step size goes to zero. In this case the method is said to be consistent.

It can be shown that all three methods are consistent.

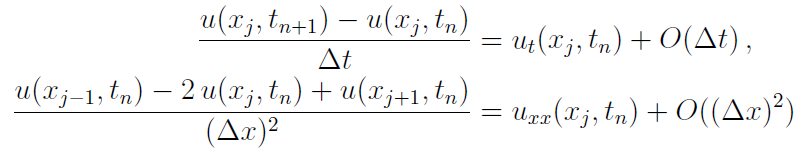
The explicit and implicit schemes have local truncation errors O (Δt, ()),

while that of the Crank–Nicolson scheme is O((), ()).

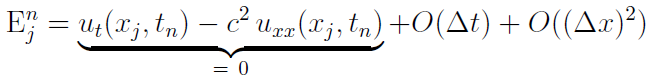
For the explicit scheme we get for the LTE at (j, n)



With the help of Taylor expansion at ( xj ,tn) we find that



Hence



**Numerical Stability**

There is really complicated analysis on the referenced article,

main idea behind this section is comparison between the approximated solution and the exact solution.

To make it simple, we can define two terms : 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 and approach zero.

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

For example , can be computed directly using:

* It can be unstable(errors are magnified)

To guarantee stability,

This means k is much smaller than h and this makes it slow.

**Observations so far**

There was really complex mathematic formulas about approximations , implicit , explicit and crank-nicolson schemes.

We have put algorithm implemantations of steady-state heat equation on Github page , the implematations are easier to understand than formulas. And their computations are at most .

And there is also another paper about Paralel Numerical Algorithms. [3]

Parallel computational complexity : O ()

Parallel communication complexity : O(n)

If we do matrix multiplication faster we can reduce the complexity of differential equations

1. Fınıte Volume Method (FVM)

The finite volume method is a discretization method which is well suited for the numerical simulation of various types (elliptic, parabolic or hyperbolic, for instance) of conservation laws; it has been extensively used in several engineering fields, such as fluid mechanics, heat and mass transfer or petroleum engineering. Some of the important features of the finite volume method are similar to those of the finite element  
method, it may be used on arbitrary geometries, using structured or unstructured meshes, and it leads to robust schemes.

The finite volume method is locally conservative because it is based on a “balance” approach: a local balance is written on each discretization cell which is often called “control volume”; by the divergence formula, an integral formulation of the fluxes over the boundary of the control volume is then obtained. The fluxes on the boundary are discretized with respect to the discrete unknowns.

1. Cell-Centered Finite Volume Method

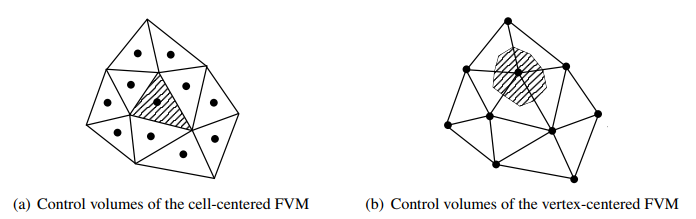
Let be a triangular or Cartesian grid of . We choose the finite dimensional space

Then , the number of elements of . We also choose . To complete the discretization, we need to assign the boundary flux of each element.

This can be done in a finite difference fashion. For example, for an interior side (an edge in 2-D and a face in 3-D) shared by two elements and , we can define

where the normal vector ne is the outward unit normal vector of in , i.e. pointing from to and are points in each element such that the line segment connecting and is orthogonal to the side . By the symmetry, for rectangles or cubes is the mass center of . For simplex, should be the circumcenters which imposes restriction on the triangulation. When the mesh is a uniform rectangular grid, it is reduced to the cell centered finite difference method;

Another approach to discretize the boundary flux is through mixed finite element methods. The gradient operator is understood as . Optimal error estimate can  
be easily obtained by using that of mixed finite element methods.



1. Vertex-Centered Finite Volume Method

Let be a polygon and let be a triangular grid of . Denoted by be the linear finite element spaces of based on :

where is the linear polynomial space on . We shall choose . The dimension is the number of interior vertices of .

The control volume will be given by another mesh satisfying

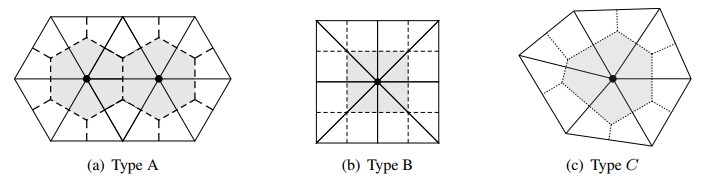
and to reflect to the Dirichlet boundary condition, we set

The element of is not necessary to be polygons. But for practical reasons, are chosen as polygons such that the boundary integral is easy to evaluate.

Given a triangulation , one construction of is given as follows: for each triangle , select a point . The point can coincides with middle points of edges.

There are three common choices of :

* Type A: is the barycenter of .
* Type B: is the middle point of the longest edge.
* Type C: is the circumcenter of .



1. Domaın Decomposition Method (DDM)

Domain decomposition methods are iterative methods for the solution of linear or nonlinear systems that use explicit information about the geometry, discretization, and/or partial differential equation (PDE) that gave rise to the (non)linear system.

Domain decomposition methods are parallel, potentially fast, robust algorithms for the solution of the linear (or nonlinear) equations that arise from discretization’s of partial differential equations(PDEs). Some of the motivations for the use of these methods include;

* Potential for efficient parallelization using data locality,
* Ability to deal with PDEs on complicated physical geometries,
* Ability to deal with PDEs that exhibit different behaviour on different parts of the domain (heterogeneous operators), and
* Superior convergence properties of the iterative method even on sequential machines.

The linear (or nonlinear) systems that arise from the discretization of PDEs inherit many algebraic properties from properties of the underlying PDE. By understanding and using these relationships it is possible to derive fast linear (and nonlinear) solvers.

For linear problems it is customary to view domain decomposition methods as preconditioners for Krylov subspace methods such as the conjugate gradient method or GMRES; see, for instance, the chapter in this volume, Linear System Solvers: Sparse Iterative Methods, by Chan and Van der Vorst. Rather than discuss the theory of preconditioners we simply state that for the solution of the linear system

the application of a preconditioner B should approximate the action of A-1 well and should be inexpensive (and parallel) to apply.

The most comprehensive reference to domain decomposition methods is Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations, by Smith, Bjorstad, and Gropp, (1995). A recent eighty-page survey of domain decomposition methods may be found in Chan and Mathew (1994). Two other limited surveys are on implementation issues by Gropp and Smith (1995)8 and non-self-adjoint problems by Cai (1994). The proceedings of the first seven international conferences on domain decomposition methods contain reports on a wide variety of applications and techniques (Chan, Glowinski, et al., 1989, 19909,10; Keyes, Chan, et al., 199211; Glowinski, Golub, et al., 198812; Glowinski, Kuznetsov, et al., 199113; Quarteroni, Periaux, et al., 1994, Keyes and Xu, 1995)14.

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 in Ω

Bu = g on Ω

A wide variety of discretization’s may be applied. Again, to simplify notations, we assume that the problem is to be discretized with conforming finite elements; finite differences, spectral methods, or finite volume methods may also be used. Domain decomposition methods may be applied equally well on structured or unstructured grids.

The variationally form of the PDE may be written as: find V such that;

a (u, v) = F(v) V (1)

Here, a (u, v) is a bilinear form, while F (v) is linear in v. The space V is a suitable function space.

For instance, for the homogeneous Dirichlet boundary value problem

,

while

If we use to denote the finite element basis functions and let , then the finite-dimensional variationally (finite element) problem may be given as: find such that

.

This is equivalent to the linear system

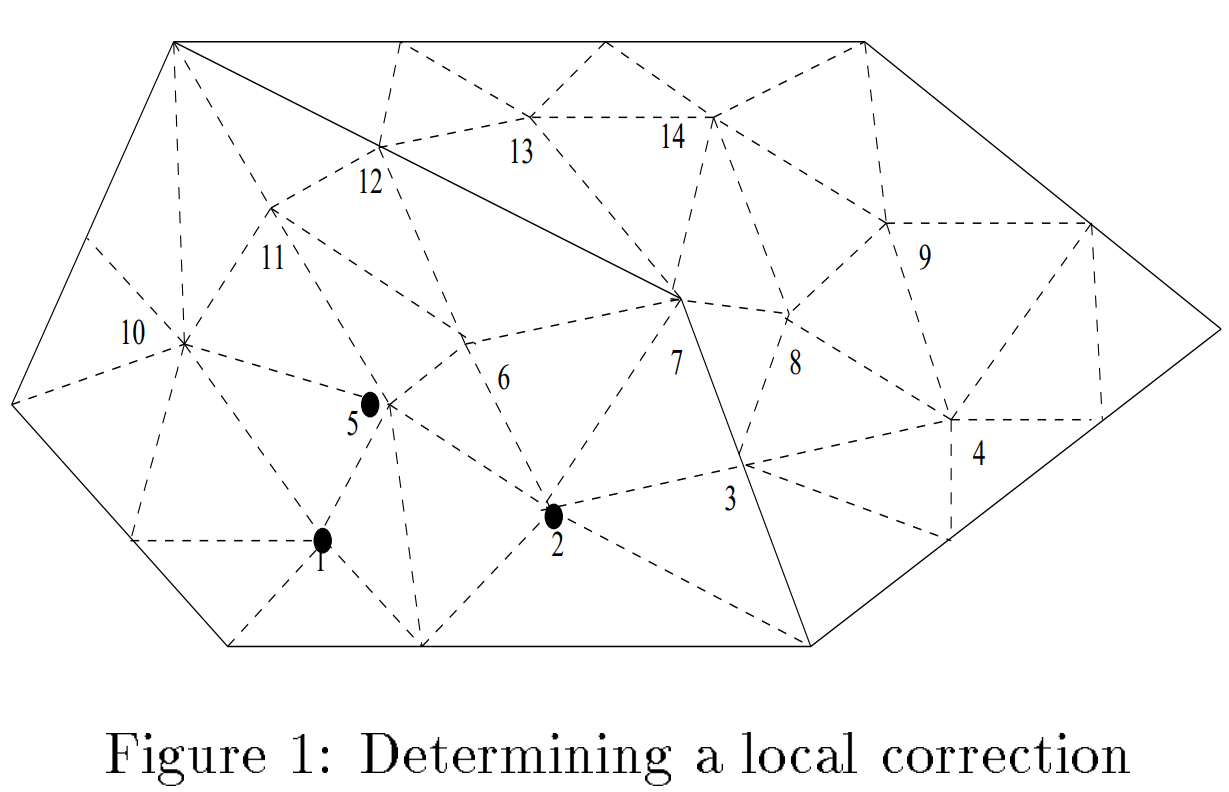
where

To motivate the design of domain decomposition methods, we  
consider the important special case when the operator L is self-adjoint and uniformly elliptic. In this case, the matrix A is symmetric, positive definite and hence defines an inner product and a corresponding norm

.

**3. Overlapping Methods**

Domain decomposition methods can be broadly classified as either overlapping or nonoverlapping methods. In this section we introduce the key ideas behind overlapping methods.



**Figure 1: Determining a local correction**

Consider the domain as depicted in Figure 1, and assume that  
a second-order, self-adjoint, uniformly elliptic PDE has been discretized by using piecewise linear finite elements on the given grid. If an approximate solution u is known, how may one improve the given solution by adjusting the values on the indicated nodes? To quantify this question, we need to introduce some notation. Let R denote the matrix that when applied to the vector u returns only those values associated with the indicated nodes. For instance, for the nodes 1, 2, and 5, the matrix R is given by

The transpose of R simply inserts the given values into the larger array:

=

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

Our “best" local correction is then defined by

or, equivalently,

Here is the exact solution to the linear system. If we take the derivative with respect to the unknowns w,

or

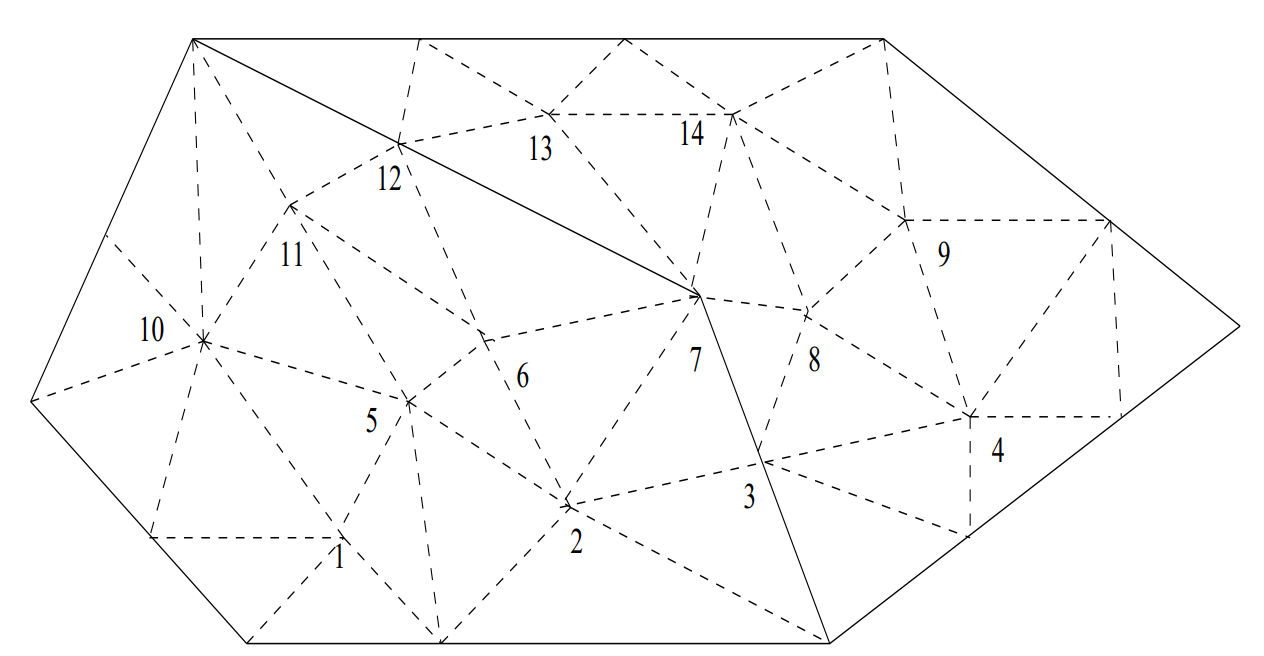
Thus the “local” correction is given by

Correction =

The matrix RART is simply the sub block of A associated with the given nodes. Hence, the process of extracting out a logical block from the matrix A and solving the reduced linear system with respect to these unknowns is a projection of the error in the A norm. It turns out that, in addition, on the PDE side the computational process maybe viewed as a projection of the error onto a subspace of the finite element space Vh. In the more general case when A is not symmetric, positive definite, the corrections are no longer orthogonal projections of the error. However, they may still have certain desirable qualities.

**4. Nonoverlapping Methods**

The nonoverlapping domain decomposition methods may also be viewed as combining projections of the error onto subspaces of the solution space. However, there is also a simple linear algebra interpretation, based on a reduced linear system, that we will adopt for this survey.



**Figure 2: Two subdomains and an interface**

Consider a domain divided into two nonoverlapping regions as depicted in Figure 3. We partition the unknowns into three sets: those in the first domain (denoted by and containing nodes 1, 2, 5, 6, 10, 11), those in in the second domain, and those on the interface between the two domains (denoted by and containing 3, 7, 12). Then the linear system may be written as

Where

contains contributions from both sides of the interface. The key point is that nodes in the first subdomain are completely decoupled from nodes in the second subdomain. Once the unknowns in and have been eliminated, the resulting Schur complement system is given by

This is easily generalized to any number of subdomains.

1. SUMMARY

We believe that that general notion of computer scientists and engineers are being good at math is a common misconception. So we tried hard to overcome our fallacies. We not only red the some description papers and books of that discretization methods but also tried to understand what fundementals topics of them and some previous numerical analysis experiments of the authors and a very fun to report 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.

We have investigated some discretization methods that FEM, FDM, FVM, DDM and have tried to descript as easily as we can. And we tried to implement some of that methods in Python but we could not report the run analysis.

References

[1] Parallel Numerical Methods for Differential Equations : Carlos A. de Moura

[2] By Ebenezer Ampadu A Masters Project Submitted to the Faculty

Of WORCESTER POLYTECHNIC INSTITUTE In partial fulfillment of the requirements for the Degree of Professional Master of Science In Financial Mathematics May 2007

[3] New York University Math6911, S08, HM ZHU , Chapter 5

[4] MSc Course in Mathematics and Finance Imperial College London, 2010-11 , Mark Davis

[5] San Jose State University Department of Mechanical and Aerospace Engineering ME 130 Applied Engineering Analysis Instructor: Tai-Ran Hsu, Ph.D. , Chapter 9

[6] http://en.wikipedia.org/wiki/Derivative\_(finance)

[7]Finite Volume Methods. Eymard, Galloueet and Herbin. Handbook of Numerical Analysis, January 2003.

[8]Finite volume schemes as non-conforming Petrov-Galerkin approximations  
of primal-dual mixed formulations, Angermann, L. ,Report 181, Institüt für Angewandte Mathematik, Universtat Erlangen-Nurnberg, 1996.  
[9]Connection between finite volume and mixed finite element methods, Baranger, J., J.-F. Maitre and F. Oudin, Model. Math. Anal. Numer., 30, 3, 4, 444-465, 1996  
[10]Introduction to the Finite Volumes Method. Application to the Shallow Water Equations, Miguel, Marques, 2008

[11] Finite element methods for flow problems, John Wiley & Sons, Chichester, England. Donea, J. & Huerta, A. (2003)

[12] Finite Volume Methods, Chen, 2009

[13] On the finite volume element method for general self-adjoint elliptic problems J. Hu1ang and S. Xi, SIAM J. Numer. Anal., 35(5):1762–1774, 1998

[14] Smith, Barry F., Bjrstad, Petter, and Gropp, William, 1995. Domain Decomposition: Paral lel Multilevel Methods for El liptic Partial Di erential Equations, Cambridge University Press.

[15] Chan, Tony, Glowinski, Roland, Periaux, Jacques, and Widlund, Olof, editors, 1989. Domain Decomposition Methods, SIAM, Philadelphia, PA.

[16] Chan, Tony, Glowinski, Roland, Periaux, Jacques, and Widlund, Olof, editors, 1990. Third International Symposium on Domain Decomposition Methods for Partial Differential Equations, SIAM, Philadelphia, PA.

[17] Keyes, David E., Chan, Tony F., Meurant, Gerard A., Scroggs, Jeffrey S., and Voigt, Robert G., editors, 1992. Fifth International Symposium on Domain Decomposition Methods for Partial Differential Equations, SIAM, Philadelphi, PA.

[18] Glowinski, Roland, Golub, Gene H., Meurant, Gerard A., and Periaux, Jacques, editors, 1988. Domain Decomposition Methods for Partial Differential Equations, SIAM, Philadelphia, PA.

[19] Glowinski, Roland, Kuznetsov, Yuri A., Meurant, Gerard A., Periaux, Jacques, and Widlund, Olof, editors, 1991. Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations, SIAM, Philadelphia, PA.

[20] Quarteroni, Al o, Periaux, Jacques, Kuznetsov, Yuri A., and Widlund, Olof B., editors, 1994. Domain Decomposition Methods in Science and Engineering, AMS, Providence, RI.