Introduction to Finite Difference Methods

Alejandro Cárdenas-Avendaño

Goals

- Fundamentals
  - Partial Differential Equations (PDEs)
  - Solution to a Partial Differential Equation
  - PDE Models
  - Classification of PDEs
  - Discrete Notation
- Finite Difference Method (FDM):
  - Details
  - General Concepts
    - Stability
    - Boundary Conditions
  - Taylor's Theorem
  - Simple Finite Difference Approximation to a Derivative
- Examples

Fundamentals

Partial Differential Equations

The following equation is an example of a PDE:

$$a(t,x,y)\frac{\partial U(t,x,y)}{\partial t}+b(t,x,y)\frac{\partial^3 U(t,x,y)}{\partial x^3}+c(t,x,y)\frac{\partial^2 U(t,x,y)}{\partial y^2}=g(t,x,y)$$

Where

- *t,x,y* are the independent variables.
- *a,b,c* and *g* are know functions of the independent variables.
- *U(t,x,y,)* is the dependent variable and is an unknown function of the independent variables.

We will use the following notation:

$$\frac{\partial U(t,x,y)}{\partial t}=U_t \qquad \frac{\partial^2 U(t,x,y)}{\partial y^2}=U_{yy}$$

Partial Differential Equations

The order of a PDE is the order of its highest derivative.

$$a(t,x,y)\frac{\partial U(t,x,y)}{\partial t}+b(t,x,y)\frac{\partial^3 U(t,x,y)}{\partial x^3}+c(t,x,y)\frac{\partial^2 U(t,x,y)}{\partial y^2}=g(t,x,y)$$

A PDE is linear if U and all its partial derivatives occur to the first power only and there are no products involving more than one of these terms.

$$a(t,x,y)\frac{\partial U(t,x,y)}{\partial t}*\left(\frac{\partial^3 U(t,x,y)}{\partial x^3}\right)+b(t,x,y)\frac{\partial^2 U(t,x,y)}{\partial y^2}=g(t,x,y)$$

No-Linear

The dimension of a PDE is the number of independent spatial variables it contains.

2D

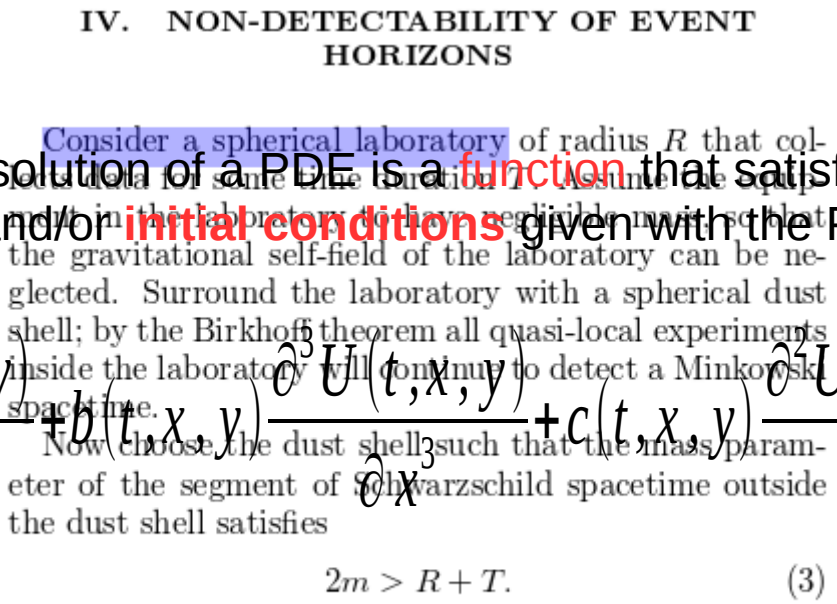
$$a(t,x,y)\frac{\partial U(t,x,y)}{\partial t}+b(t,x,y)\frac{\partial^3 U(t,x,y)}{\partial x^3}+c(t,x,y)\frac{\partial^2 U(t,x,y)}{\partial y^2}=g(t,x,y)$$

Solution to a Partial Differential Equation

Easy?

… Find U(t,x,y)

Analytical



IV. NON-DETECTABILITY OF EVENT HORIZONS

Consider a spherical laboratory of radius R that collects data for some time interval T. Assume the equipment in the laboratory do have negligible mass so that the gravitational self-field of the laboratory can be neglected. Surround the laboratory with a spherical dust shell; by the Birkhoff theorem all quasi-local experiments inside the laboratory will continue to detect a Minkowski spacetime.

Now choose the dust shell such that the mass parameter of the segment of Schwarzschild spacetime outside the dust shell satisfies

$$2m > R + T. \qquad (3)$$

An analytical (i.e. exact) solution of a PDE is a **function** that satisfies the PDE and also satisfies any **boundary** and/or **initial conditions** given with the PDE.

$$a(t,x,y)\frac{\partial U(t,x,y)}{\partial t}+b(t,x,y)\frac{\partial^3 U(t,x,y)}{\partial x^3}+c(t,x,y)\frac{\partial^2 U(t,x,y)}{\partial y^2}=g(t,x,y)$$

Most PDEs of interest **do not** have analytical solutions so a numerical procedure **must** be used to find an approximate solution.

arXiv:1407.7295

Solution to a Partial Differential Equation

KONRAD LORENZ
FUNDACIÓN UNIVERSITARIA

Analytical

"The purpose of this article is to get mathematicians interested in studying a number of partial differential equations (PDEs) that naturally arise in macroeconomics."

The equilibrium can be characterized in terms of an HJB equation for the value function v and a Fokker–Planck equation for the density of households g. In a *stationary* equilibrium, the unknown functions v and g and the unknown scalar r satisfy the following system of coupled PDEs (stationary mean field game) on $(\underline{a}, \infty) \times (\underline{z}, \bar{z})$:

$$\frac{1}{2}\sigma^2(z)\partial_{zz}v + \mu(z)\partial_z v + (z + ra)\partial_a v + H(\partial_a v) - \rho v = 0, \tag{2.1}$$

$$-\frac{1}{2}\partial_{zz}(\sigma^2(z)g) + \partial_z(\mu(z)g) + \partial_a((z + ra)g) + \partial_a(\partial_p H(\partial_a v)g) = 0, \tag{2.2}$$

$$\int g(a, z)\, da\, dz = 1, \quad g \geq 0 \tag{2.3}$$

and

$$\int a g(a, z)\, da\, dz = 0, \tag{2.4}$$

where the Hamiltonian H is given by

$$H(p) = \max_{c \geq 0}(-pc + u(c)). \tag{2.5}$$

The function v satisfies a state constraint boundary condition at $a = \underline{a}$ and Neumann boundary conditions at $z = \underline{z}$ and $z = \bar{z}$.

In general, the boundary value problem including the Bellman equation (2.1) and the boundary condition has to be understood in the sense of viscosity (see Bardi & Capuzzo [24], Crandall *et al.* [25], Barles [26]), whereas the boundary problem with the Fokker–Planck equation (2.3) is set in the sense of distributions. An important issue is to check that (2.1) actually yields an optimal control (verification theorem): this is a direct application of Itô's formula if v is smooth enough; for general viscosity solutions, one may apply the results of Bouchard & Touzi [27] and Touzi [28] (this has not been done yet).

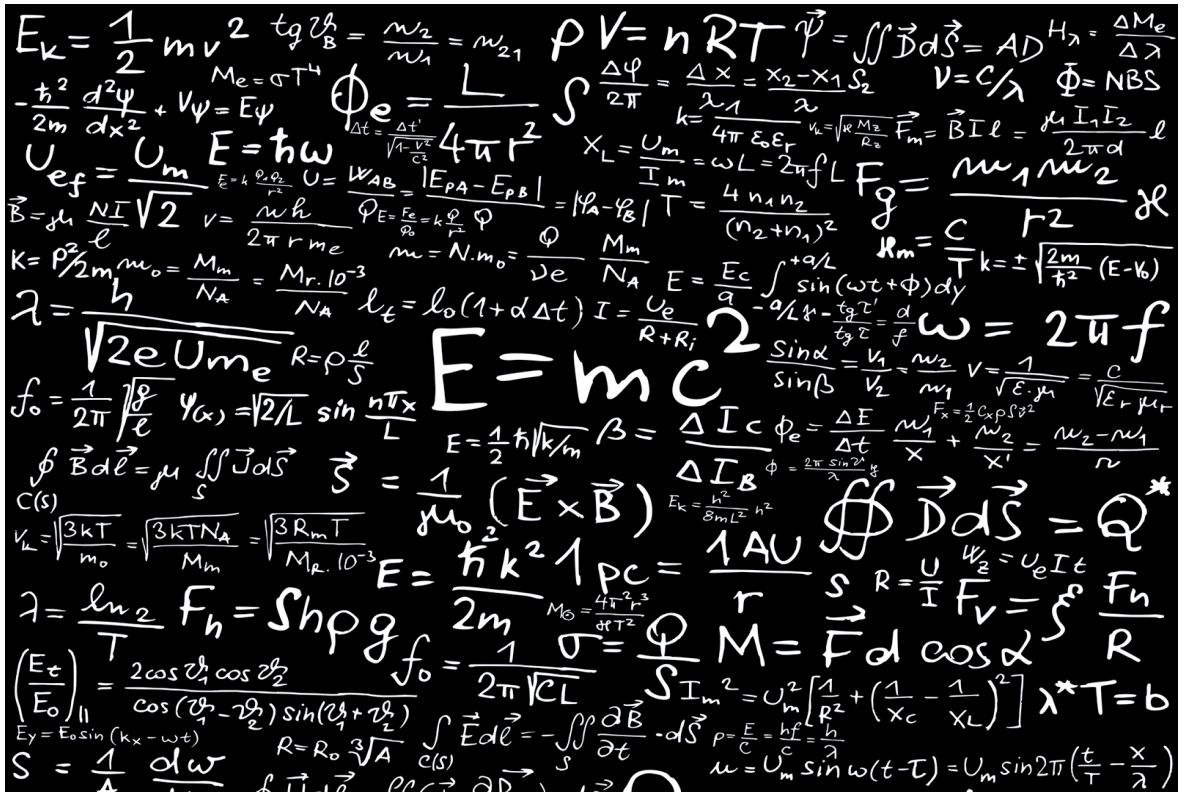
With well chosen initial and terminal conditions, solutions to the HJB equation (2.1) are expected to be smooth and we therefore look for such smooth solutions. If v is indeed smooth, the state constraint boundary condition can be shown to imply

$$(z + \underline{m})\lambda + H(\lambda) \geq (w + r\underline{a})\partial_a v(\underline{a}, z) + H(\partial_a v(\underline{a}, z)) \quad \forall \lambda \geq \partial_a v(\underline{a}, z)$$

AchdouY,BueraFJ,LasryJ-M,LionsP-L,MollB.2014 Partialdifferentialequationmodelsinmacroeconomics.Phil.Trans. R. Soc. A372:20130397.http://dx.doi.org/10.1098/rsta.2013.0397

Partial Differential Equations

PDEs describe many of the fundamental natural laws



- Conservation of mass
- Laplace's equation
- Maxwell's equations
- Navier–Stokes equations

Solution to a Partial Differential Equation

Numerical

The approximation is made at discrete values of the independent variables and the approximation scheme is implemented via a computer program.

Finite Difference Method

The FDM replaces all partial derivatives and other terms in the PDE by approximations. After some manipulation, a finite difference scheme (FDS) is created from which the approximate solution is obtained. The FDM depends fundamentally on Taylor's beautiful theorem (circa 1712!).

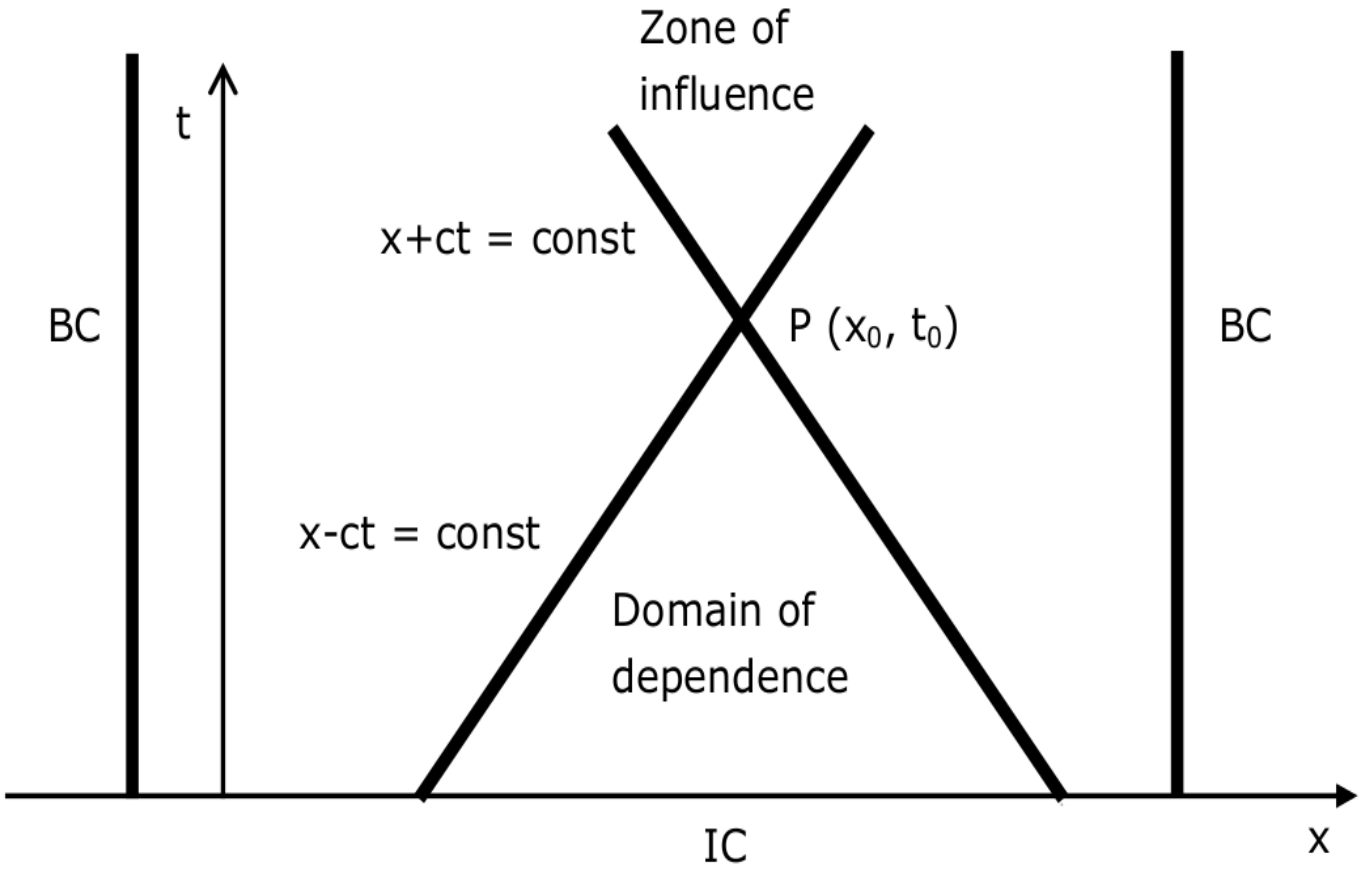
Classification of PDEs

Board I

Classification of PDEs

The differences between the types of PDEs can be illustrated by sketching their respective domains of dependence.

Hyperbolic case:

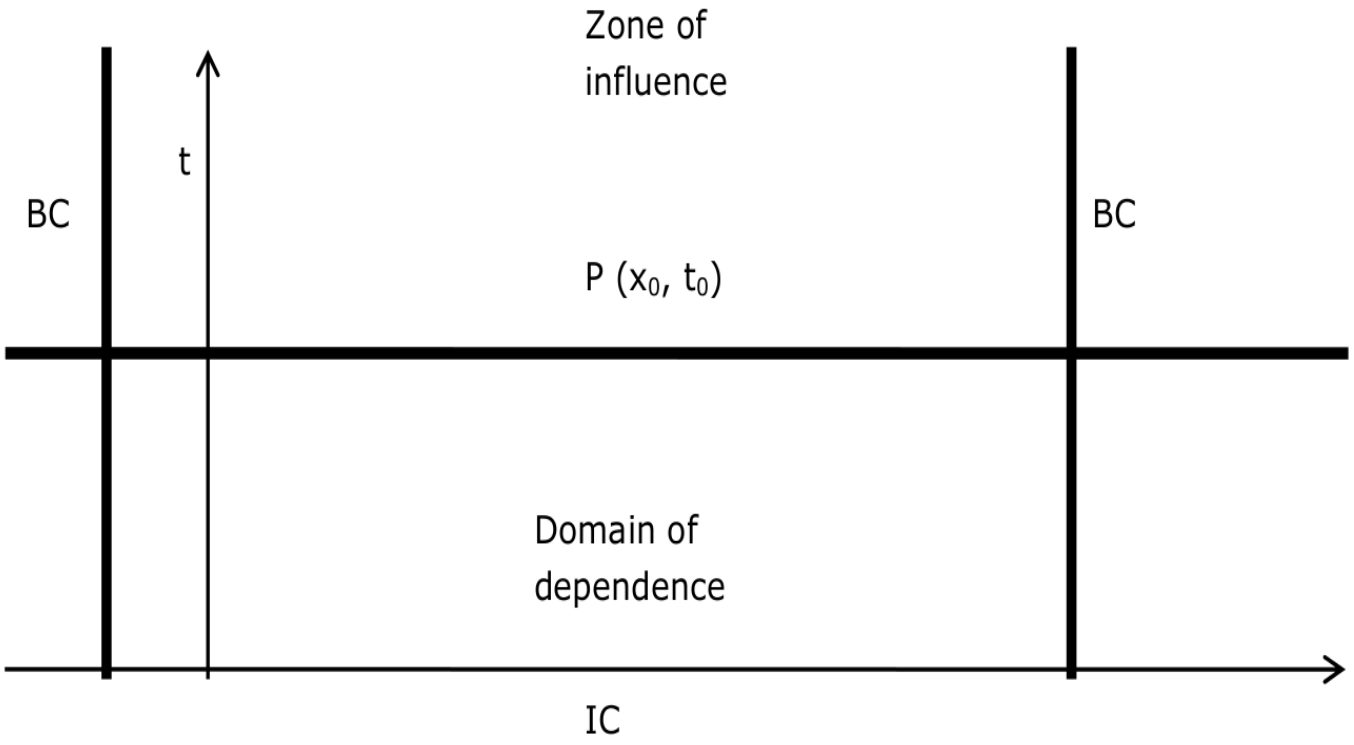


Point P (x_0,t_0) can only be influenced by points lying within the region bounded by the two characteristics x+ct = const and x-ct = const and $t < t_0$. This region is called the domain of dependence.

Classification of PDEs

The differences between the types of PDEs can be illustrated by sketching their respective domains of dependence.

Parabolic case:

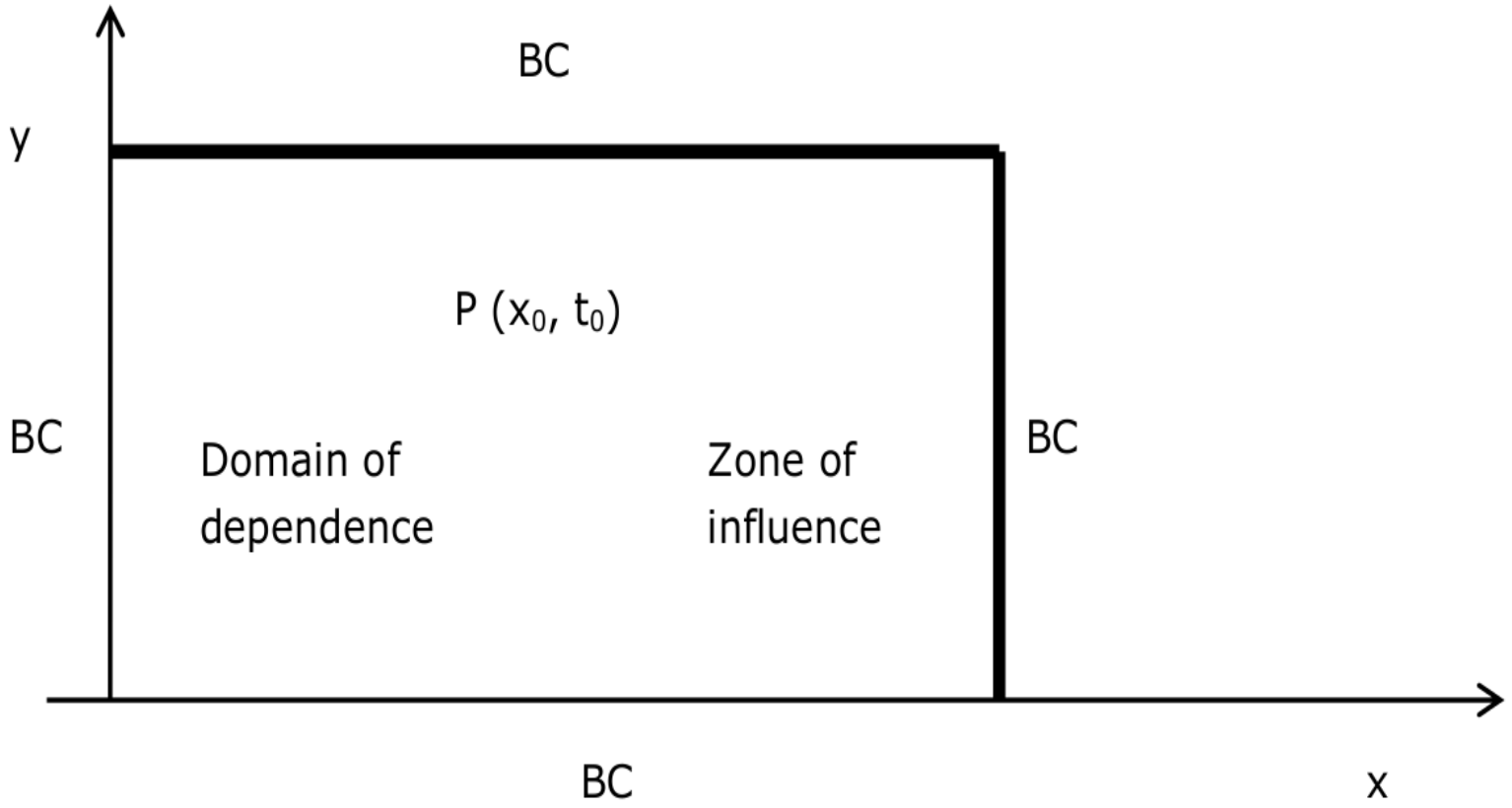


Information travels downstream (or forward in time) only and so the domain of dependence of point P (x_0,t_0) in this case is the region $t < t_0$ and the zone of influence is all points for which $t < t_0$

Classification of PDEs

The differences between the types of PDEs can be illustrated by sketching their respective domains of dependence.

Elliptic case:



Information travels in all directions at infinite speed so the solution at point $P(x_0,t_0)$ influences all points within the domain and vice versa.

Classification of PDEs

The type of PDE fundamentally influences the choice of solution strategy.

Time dependent hyperbolic problems and parabolic problems are solved numerically by time-marching methods which involves, as its name suggests, obtaining the numerical solution at a later time from that at an earlier time starting from given ICs.

Elliptic problems are solved numerically by so-called relaxation methods.

What do we need?

- Well-posedness
  - The solution exists
  - The solution is unique
  - The solution depends “continuously” on the initial and boundary conditions

Example:

$$u_t = u_x$$

Lets assume:

$$u(t,x) = f(t) * e^{ikx}$$

We should be REALLY careful about mathematical theory

Examples

- Find a solution to

$$u_t = u_x$$

Lets assume:

$$u(t,x) = f(t) * e^{ikx}$$

Norm of the solutions:

$$|u(t,x)| = 1$$

- Find a solution to

$$u_t = b u_{xx}$$

Lets assume:

$$u(t,x) = f(t) * e^{ikx}$$

$$|u(t,x)| = e^{bk^2 t}$$

Discrete Notation

We will use upper case U to denote the analytic (exact) solution of the PDE and lower case u to denote the numerical (approximate) solution.

Subscripts will denote discrete points in space and superscripts discrete levels in time.

e.g.

$$u_{i,j}^{n}$$

denotes the numerical solution at grid point (i, j) in a 2D region at time level n.

Checking Results

Verification:

The computer program implementing the scheme **must** be verified. This is a check to see if the program is doing what it is supposed to do. Comparing results from **pen and paper calculations** at a small number of points to equivalent computer output is a way to (partially) verify a program.

Validation:

Validation is really a **check** on whether the PDE is a good model for the real problem being studied. Validation means **comparing numerical results** with results from similar physical problems.

Taylor's Theorem

The finite difference method (FDM) works by replacing the region over which the independent variables in the PDE are defined by a finite grid (also called a *mesh*) of points at which the dependent variable is approximated.

The partial derivatives in the PDE at each grid point are approximated from neighbouring values by using Taylor's theorem.

Taylor's Theorem:

Let *U(x)* have *n* continuous derivatives over the interval *(a, b)*. Then for $a < x_o , x_o + h < b$,

$$U(x_0+h)=U(x_0)+hU_x(x_0)+h^2\frac{U_{xx}(x_0)}{2!}+\ldots+h^{n-1}\frac{U_{n-1}(x_0)}{(n-1)!}+O(h^n) \qquad \textbf{(1)}$$

where

$O(h^n)$ is an unknown error term

$U_x(x_0)$ is the derivative of U with respect to x evaluated at $x = x_o$.

"order h to the n"

Finite Difference Method

Corrections

Taylor's Theorem

Dirichlet and Neumann Conditions

Simple Finite Difference Approximation to a Derivative

Truncating (1) after the first derivative term gives,

$$U(x_0+h)=U(x_0)+hU_x(x_0)+O(h^2)$$

Rearranging gives,

$$U_x(x_0)=\frac{U(x_0+h)-U(x_0)}{h}-\frac{O(h^2)}{h}$$

Neglecting the *O(h)* term gives,

$$U_x(x_0)\approx\frac{U(x_0+h)-U(x_0)}{h}$$

The las Eq. is called a first order FD approximation to U x (x o) since the approximation error=O(h) which depends on the first power of h.

This approximation is called a forward FD approximation since we start at x_o and step forwards to the point x_o+h. *h* is called the **step size** (h > 0).

Simple Finite Difference Approximation to a Derivative

As an example we choose a simple function for *U*. Let

$$U(x) = x^3$$

We will find the first order forward FD approximation to U_x *(1)* using step size *h = 0.1*

Since

$$U_x(x_0) \approx \frac{U(x_0+h) - U(x_0)}{h}$$

Substituting for U gives

$$U_x(x_0) \approx \frac{(x_0+h)^3 - x_0^3}{h}$$

Replacing x_o by *1* and *h* by *0.1* gives,

$$U_x(1) \approx \frac{(1+0.1)^3 - 1^3}{0.1} = 3{,}311$$

What if h=0.05?

Lets do it by hand

Finite Difference Approximations

For simplicity we suppose that *U* is a function of **only two variables**, t and x.

We will approximate the **partial derivatives of U with respect to *x***.

As *t* is held constant U is effectively a function of the single variable *x* so we can use Taylor's formula (1) where the ordinary derivative terms are now partial derivatives and the arguments are *(t, x)* instead of *x*.

Finally we will replace the step size ***h*** by **Δx** (to indicate a change in x) so that (1) becomes,

$$U(t,x_0+\Delta x)=U(t,x_0)+\Delta x U_x(t,x_0)+\frac{\Delta x^2}{2!}U_{xx}(t,x_0)+\ldots+\frac{\Delta x^{n-1}}{(n-1)!}U_{n-1}(t,x_0)+O(\Delta x^n) \quad \textbf{(2)}$$

Truncating it to $O(\Delta x^2)$ gives,

$$U(t,x_0+\Delta x)=U(t,x_0)+\Delta x U_x(t,x_0)+O(\Delta x^2)$$

Now we derive some FD approximations to partial derivatives. Rearranging it gives,

$$U_x(t,x_0)=\frac{U(t,x_0+\Delta x)-U(t,x_0)}{\Delta x}-O(\Delta x)$$

Finite Difference Approximations

In numerical schemes for solving PDEs we are restricted to a grid of discrete *x* values, x_1, x_2 ,...,x_N , and discrete *t* levels t_0 , t_1 , ...

We will assume a **constant** grid spacing, Δx, in *x*, so that $x_{i+1} = x_i + \Delta x$.

Evaluating the last equation for a point, (t_n , x_i), on the grid gives,

$$U_x(t_n, x_i) = \frac{U(t_n, x_{i+1}) - U(t_n, x_i)}{\Delta x} - O(\Delta x)$$

We will use the common *subscript/superscript* notation

$$U_i^n = U(t_n, x_i)$$

so that dropping the O(Δx) error term,

$$U_x(t_n, x_i) \approx \frac{U_{i+1}^n - U_i^n}{\Delta x}$$

Finite Difference Approximations

We now derive **another FD approximation** to $U_x\,(t_n\,,\,x_i)$. Replacing Δx by $-\Delta x$

$$U(t,x_0-\Delta x)=U(t,x_0)-\Delta x U_x(t,x_0)+O(\Delta x^2)$$

Evaluating it at $(t_n\,,\,x_i\,)$ and rearranging as previously gives,

$$U_x(t_n,x_i)\approx\frac{U_i^n-U_{i-1}^n}{\Delta x}$$

And it is the **first order backward** difference approximation to $U_x\,(\,t_n\,,\,x_i)$.

Our first two FD approximations are first order in *x* but we can increase the order (and so make the approximation more **accurate**) by taking more terms in the Taylor series as follows.

Truncating to $O(\Delta x^3)$, then replacing Δx by $-\Delta x$ and subtracting this new expression from (2) and evaluating at $(t_n\,,x_i)$ gives, after some algebra,

$$U_x(t_n,x_i)\approx\frac{U_{i+1}^n-U_{i-1}^n}{2\Delta x}$$

And is called the **second order central difference** FD approximation to $U_x\,(\,t_n\,,\,x_i)$.

Finite Difference Approximations

Many PDEs of interest contain second order (and higher) partial derivatives so we need to derive approximations to them.

We will restrict our attention to second order *unmixed* partial derivatives i.e. U_{xx} .

Truncating (2) to $O(\Delta x^4)$ gives

$$U(t,x_0+\Delta x)=U(t,x_0)+\Delta x U_x(t,x_0)+\frac{\Delta x^2}{2!}U_{xx}(t,x_0)+\frac{\Delta x^3}{3!}U_{xxx}(t,x_0)+O(\Delta x^4) \quad \textbf{(3)}$$

Replacing Δx by $-\Delta x$ gives

$$U(t,x_0-\Delta x)=U(t,x_0)-\Delta x U_x(t,x_0)+\frac{\Delta x^2}{2!}U_{xx}(t,x_0)-\frac{\Delta x^3}{3!}U_{xxx}(t,x_0)+O(\Delta x^4) \quad \textbf{(4)}$$

Adding (3) and (4) gives

$$U(t,x_0+\Delta x)+U(t,x_0-\Delta x)=2U(t,x_0)+\Delta x^2 U_{xx}(t,x_0)+O(\Delta x^4)$$

Evaluating at (t_n, x_i)

$$U_{i+1}^n+U_{i-1}^n=2U_i^n+\Delta x^2 U_{xx}(t_n,x_i)+O(\Delta x^4)$$

Finite Difference Approximations

Rearranging it and dropping the $O(\Delta x^2)$ error term gives

$$U_{xx}(t_n, x_i) \approx \frac{U_{i+1}^n - 2U_i^n + U_{i-1}^n}{\Delta x^2} \qquad \textbf{(5)}$$

And it is the second order symmetric difference FD approximation to $U_{xx}(t_n,x_i)$

partial derivative	finite difference approximation	type	order
$\frac{\partial U}{\partial x} = U_x$	$\frac{U_{i+1}^n - U_i^n}{\Delta x}$	forward	first in x
$\frac{\partial U}{\partial x} = U_x$	$\frac{U_i^n - U_{i-1}^n}{\Delta x}$	backward	first in x
$\frac{\partial U}{\partial x} = U_x$	$\frac{U_{i+1}^n - U_{i-1}^n}{2\Delta x}$	central	second in x
$\frac{\partial^2 U}{\partial x^2} = U_{xx}$	$\frac{U_{i+1}^n - 2U_i^n + U_{i-1}^n}{\Delta x^2}$	symmetric	second in x

Finite Difference Approximations

Approximations to partial derivatives with respect to *t* are derived in a similar manner

partial derivative	finite difference approximation	type	order
$\frac{\partial U}{\partial t} = U_t$	$\frac{U_i^{n+1} - U_i^n}{\Delta t}$	forward	first in t
$\frac{\partial U}{\partial t} = U_t$	$\frac{U_i^{n} - U_i^{n-1}}{\Delta t}$	backward	first in t
$\frac{\partial U}{\partial t} = U_t$	$\frac{U_i^{n+1} - U_i^{n-1}}{2\Delta t}$	central	second in t
$\frac{\partial^2 U}{\partial t^2} = U_{tt}$	$\frac{U_i^{n+1} - 2U_i^{n} + U_i^{n-1}}{\Delta t^2}$	symmetric	second in t

Example

The 1D linear advection equation is

$$U_t + vU_x = 0$$

where the independent variables are *t* (time) and *x* (space). x is restricted to the finite interval *[p, q]* which is called the computational domain.

v is a constant and the dependent variable, U = U(t,x).

Let the initial conditions be,

Known!

$$U(0,x) = f(x) \qquad p \leqslant x \leqslant q$$

A solution is a function U = U(t, x) which satisfies the PDE at all points *x* in the computational domain and all times *t* and the initial conditions.

Step by Step

Step 1: Spatial Discretization

The computational domain contains an infinite number of x values so first we must replace them by a finite set. This process is called spatial discretization

For simplicity the computational domain is replaced by a grid of **N** *equally spaced grid points.* Starting with the first grid point at *x=p* and ending with the last grid point at *x=q*, the constant grid spacing, Δx, is,

$$\Delta x = \frac{q-p}{N-1}$$

The values of *x* in the discretized computational domain are indexed by subscripts to give,

$$x_1 = p, x_2 = p + \Delta x, \ldots, x_i = p + (i-1)\Delta x, \ldots, x_N = p + (N-1)\Delta x = q$$

So the grid spacing is constant,

$$x_{i+1} = x_i + \Delta x$$

Fixing t at t = t_n we approximate the spatial partial derivative, U_x at each point (t_n,x_i) using the forward difference formula

$$U_t + vU_x = 0 \quad \longrightarrow \quad U_t + v\frac{U_{i+1}^n - U_i^n}{\Delta x} = 0$$

semi-discrete form since only the spatial derivative has been discretized

Step 2: Time Discretization

Fixing x at $x=x_i$ we approximate the temporal partial derivative, U_t at each point (t_n, x_i) using the first order forward difference formula

$$U_t \approx \frac{U_i^{(n+1)} - U_i^n}{\Delta t}$$

and

$$U_t + v\frac{U_{i+1}^n - U_i^n}{\Delta x} = 0 \quad \longrightarrow \quad \frac{U_i^{(n+1)} - U_i^n}{\Delta t} + v\frac{U_{i+1}^n - U_i^n}{\Delta x} = 0$$

which rearranges to give

$$U_i^{(n+1)} = U_i^{(n)} - \frac{v\Delta t}{\Delta x}\left(U_{i+1}^{(n)} - U_i^{(n)}\right)$$

It is an example of a FDS to approximate the solution of the PDE. Is a so-called time-marching scheme which enables U values at time level n+1 to be approximated from U values at the previous time level n.

Since all U values are only known exactly at the initial time level is rewritten as

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

:$u(t_n,x_i)$ is a numerical approximation to $U(t_n,x_i)$

Step 2: Time Discretization

Remarks

- $u(0,x_i) = U(0,x_i)$ but this will **not** be true in general for later times.

- **u** values on the right hand side of are all at time t_n whereas on the left hand side **u** values are all at the **next** timed level $t_n + \Delta t = t_{n+1}$
- It is an example of a time-marching scheme in that (known) data for each grid point at time t_n is used to find data at each grid point at the future time $t_n + \Delta t$. This is called an **iteration** of the scheme. After an iteration of the scheme all *u* values at each grid point are known at time $t_n + \Delta t$.

- These new values can be used as known data for another iteration of the scheme to give data for each grid point at the next time level.

- This process can be repeated until the required future time is attained.

- The errors in approximating the spatial and temporal derivatives which are used are $O(\Delta x)$ and $O(\Delta t)$ respectively and so it is said to be (formally) first order in space (*x*) and first order in time (*t*).

- The grid spacing, Δx, was determined by choosing the number of grid points, N. A larger N gives a smaller Δx and a (hopefully) more accurate solution as spatial derivatives are more accurately approximated. However as N increases compute time increases so there is a trade off between **accuracy and speed**.

Step 2: Time Discretization

Remarks

- The time step, Δt, is for the moment, chosen arbitrarily. However a smaller time step will mean that more iterations are needed to reach a stated future time which will obviously increase the compute time.

- In addition, since the result of each iteration is an approximation to the required solution, more iterations could cause the build up of more error.

$$u_i^{(n+1)} = u_i^{(n)} - \frac{v\,\Delta t}{\Delta x}\left(u_{i+1}^{(n)} - u_i^{(n)}\right)$$

is said to be an *explicit method* since the value of *u* at the next time level is given by an explicit formula for each grid point.

MATHEMICAL WARNING:

It does not work for v>0...

KONRAD LORENZ
FUNDACIÓN UNIVERSITARIA

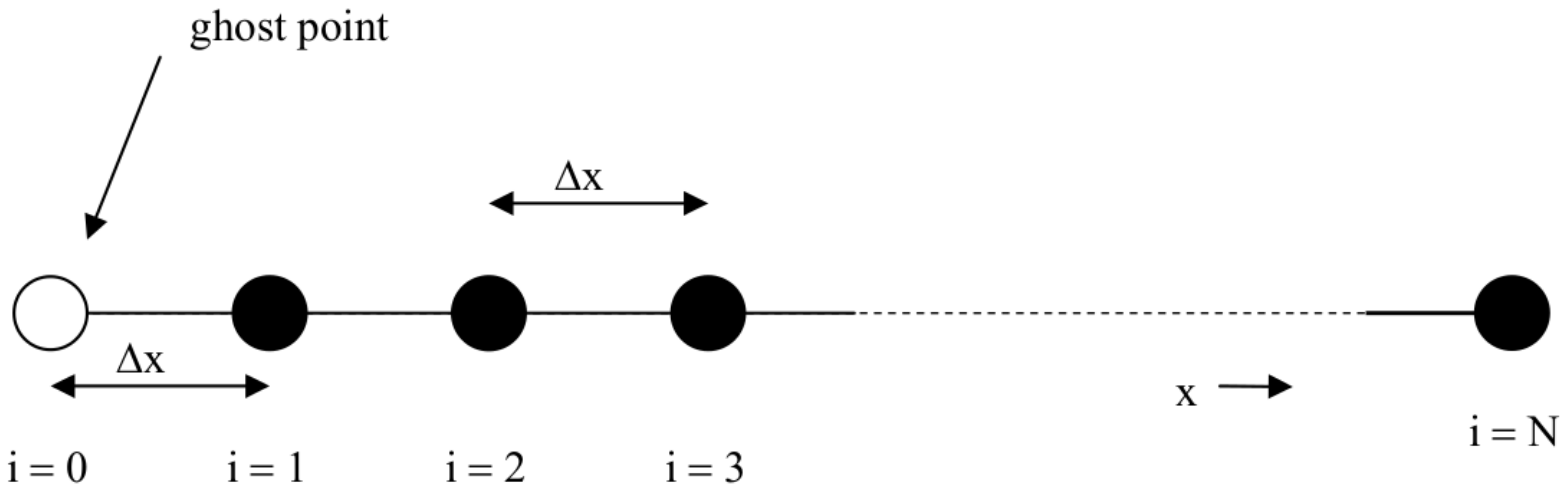
PEN AND PAPER CALCULATION

Example

Board II

Boundary Conditions

When solving a PDE using a finite difference (FD) scheme we may need to specify ghost grid points and associated ghost values for the dependent variable at these points.



In a computational region (which may be 1, 2 or 3D) ghost points and associated values occur at or adjacent to the boundaries of the region. Conditions leading to the prescription of ghost values are called boundary conditions.

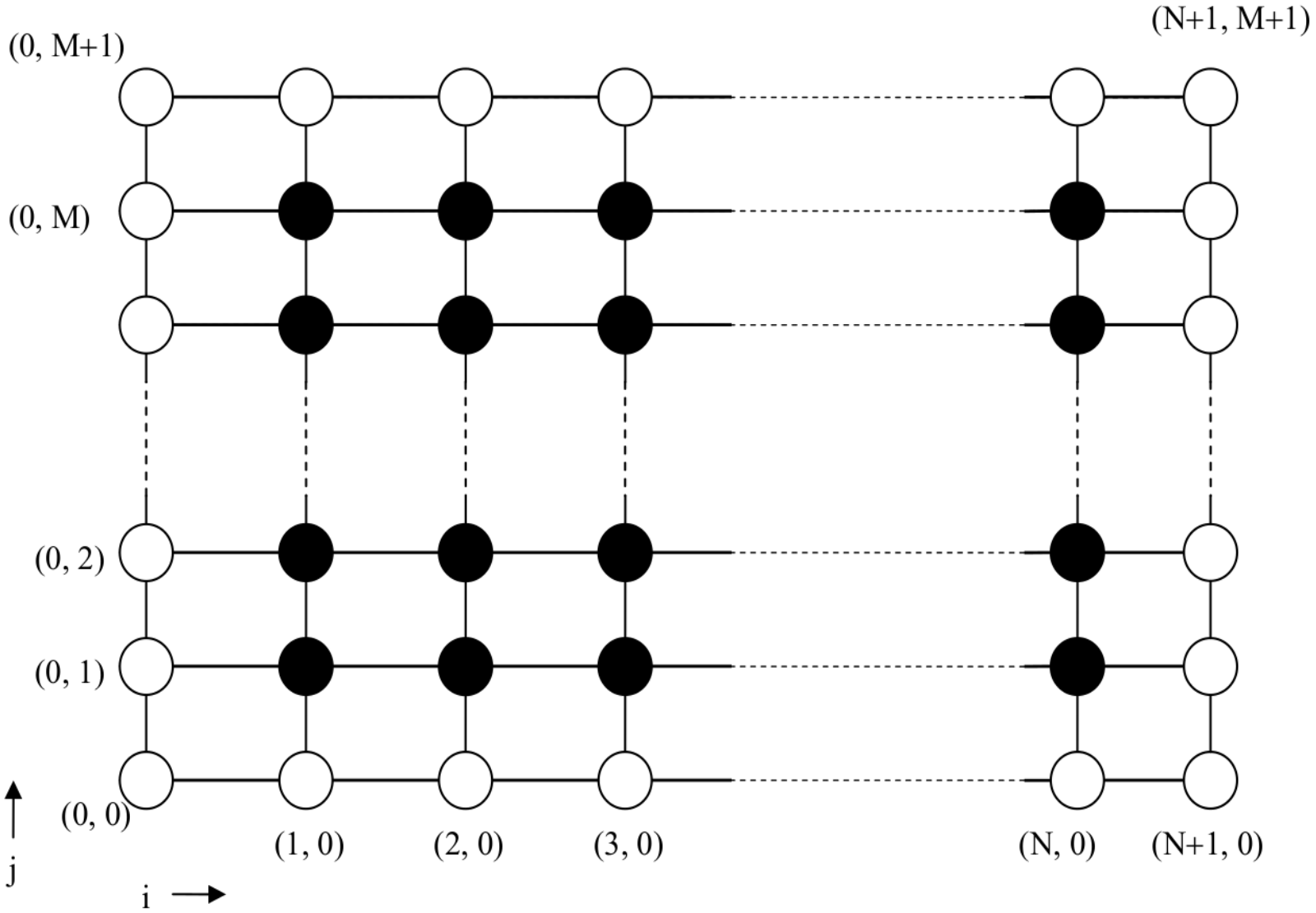
Be careful with the indexation of each language:

Python

C++

Matlab

Boundary Conditions



Definition and Properties of Order

Definition of ***O(h^n)***:

We say that a function *f(h)* "is order h ti the n" when:

$$\lim_{h \to 0} \frac{f(h)}{h^n} = C$$

Where C is a **non-zero** constant.

Examples:

$$137h^4 + 24h^3 - 2h = O(h)$$

$$137h^7 = O(h^7)$$

Definition and Properties of Order

If f(h) = ***O(h^n)*** then, for **small** h, we get:

$$\frac{f(h)}{h^n} \approx C \quad \longrightarrow \quad f(h) \approx C h^n$$

Where C is a **non-zero** constant.

So for small **h**, an error which is O(h^n) is proportional to h^n.

Example:

If the error is O(h^3) then it is proportional to h^3 which means that halving **h** reduces the error by a factor of 2^3 = **8**.

Properties of Order

- Scaling a function by a **constant** doesn't change its order. In particular f(h) and -f(h) have the same order.
- The order of the sum of two functions of different orders is the **smaller** of the orders of the two functions

$$O(h)+O(h^8)=O(h)$$

- Dividing a function by h^n reduces its order by n.
- The order of the product of two functions is the sum of their orders

$$O(h)O(h^8)=O(h^9)$$

References

D. M. Causon, C. G. Mingham, Introductory Finite Difference Methods for PDEs Ventus Publishing ApS, ISBN 978-87-7681-642-1 (2010).