WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
Català
Deutsch
Español
فارسی
Français
हिन्दी
Italiano
Nederlands
日本語
Norsk nynorsk
Polski
Português
Русский
Suomi
Svenska
Türkçe
Українська
中文
Edit links

Finite difference method

From Wikipedia, the free encyclopedia

Not to be confused with "finite difference method based on variation principle", the first name of finite element method[citation needed].

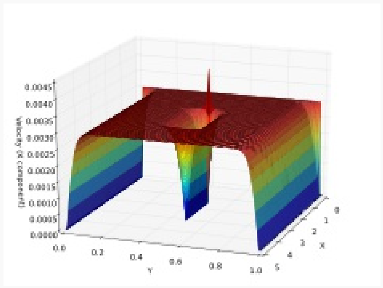
This article has multiple issues. Please help **improve it** or discuss these issues on the **talk page**. [hide]

- This article's **lead section may not adequately summarize key points of its contents**. *(April 2015)*
- This article **may be too technical for most readers to understand**. *(April 2015)*

In mathematics, **finite-difference methods** (FDM) are numerical methods for solving differential equations by approximating them with difference equations, in which finite differences approximate the derivatives. FDMs are thus discretization methods.

Today, FDMs are the dominant approach to numerical solutions of partial differential equations.[1]

Differential equations



Navier–Stokes differential equations used to simulate airflow around an obstruction.

Scope [show]	
Classification	
Types	[show]
Relation to processes	[show]
Solution	
General topics	[show]
Solution methods	[show]
People	[show]

v · t · e

Contents [hide]

Derivation from Taylor's polynomial [edit]

First, assuming the function whose derivatives are to be approximated is properly-behaved, by Taylor's theorem, we can create a Taylor Series expansion

$$f(x_0+h)=f(x_0)+\frac{f'(x_0)}{1!}h+\frac{f^{(2)}(x_0)}{2!}h^2+\cdots+\frac{f^{(n)}(x_0)}{n!}h^n+R_n(x),$$

where *n*! denotes the factorial of *n*, and $R_n(x)$ is a remainder term, denoting the difference between the Taylor polynomial of degree *n* and the original function. We will derive an approximation for the first derivative of the function "f" by first truncating the Taylor polynomial:

$$f(x_0+h)=f(x_0)+f'(x_0)h+R_1(x),$$

Setting, x_0=a we have,

$$f(a+h)=f(a)+f'(a)h+R_1(x),$$

Dividing across by *h* gives:

$$\frac{f(a+h)}{h}=\frac{f(a)}{h}+f'(a)+\frac{R_1(x)}{h}$$

Solving for f'(a):

$$f'(a)=\frac{f(a+h)-f(a)}{h}-\frac{R_1(x)}{h}$$

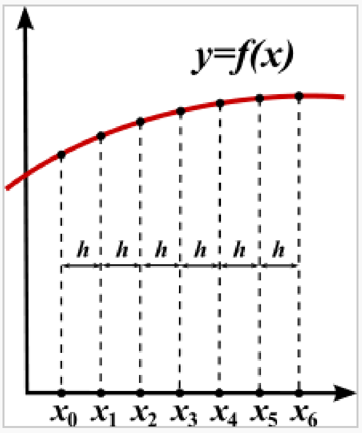
Assuming that $R_1(x)$ is sufficiently small, the approximation of the first derivative of "f" is:

$$f'(a) \approx \frac{f(a+h) - f(a)}{h}.$$

Accuracy and order [edit]

See also: Finite difference coefficient

The error in a method's solution is defined as the difference between the approximation and the exact analytical solution. The two sources of error in finite difference methods are round-off error, the loss of precision due to computer rounding of decimal quantities, and truncation error or discretization error, the difference between the exact solution of the finite difference equation and the exact quantity assuming perfect arithmetic (that is, assuming no round-off).



The finite difference method relies on discretizing a function on a grid.

To use a finite difference method to approximate the solution to a problem, one must first discretize the problem's domain. This is usually done by dividing the domain into a uniform grid (see image to the right). Note that this means that finite-difference methods produce sets of discrete numerical approximations to the derivative, often in a "time-stepping" manner.

An expression of general interest is the local truncation error of a method. Typically expressed using Big-O notation, local truncation error refers to the error from a single application of a method. That is, it is the quantity $f'(x_i) - f'_i$ if $f'(x_i)$ refers to the exact value and f'_i to the numerical approximation. The remainder term of a Taylor polynomial is convenient for analyzing the local truncation error. Using the Lagrange form of the remainder from the Taylor polynomial for $f(x_0 + h)$, which is

$$R_n(x_0+h) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(h)^{n+1},$$ where
$$x_0 < \xi < x_0 + h,$$

the dominant term of the local truncation error can be discovered. For example, again using the forward-difference formula for the first derivative, knowing that $f(x_i) = f(x_0 + ih)$,

$$f(x_0+ih) = f(x_0) + f'(x_0)ih + \frac{f''(\xi)}{2!}(ih)^2,$$

and with some algebraic manipulation, this leads to

$$\frac{f(x_0+ih) - f(x_0)}{ih} = f'(x_0) + \frac{f''(\xi)}{2!}ih,$$

and further noting that the quantity on the left is the approximation from the finite difference method and that the quantity on the right is the exact quantity of interest plus a remainder, clearly that remainder is the local truncation error. A final expression of this example and its order is:

$$\frac{f(x_0+ih) - f(x_0)}{ih} = f'(x_0) + O(h).$$

This means that, in this case, the local truncation error is proportional to the step sizes. The quality and duration of simulated FDM solution depends on the discretization equation selection and the step sizes (time and space steps). The data quality and simulation duration increase significantly with smaller step size.[2] Therefore, a reasonable balance between data quality and simulation duration is necessary for practical usage. Large time steps are favourable to increase simulation speed in many practice, however too large time steps may create instabilities and affecting the data quality.[3][4]

The von Neumann method (Fourier stability analysis) is usually applied to determine the numerical model stability.[3][4][5][6]

Example: ordinary differential equation [edit]

For example, consider the ordinary differential equation

$$u'(x) = 3u(x) + 2.$$

The Euler method for solving this equation uses the finite difference quotient

$$\frac{u(x+h)-u(x)}{h} \approx u'(x)$$

to approximate the differential equation by first substituting in for u'(x) then applying a little algebra (multiplying both sides by h, and then adding u(x) to both sides) to get

$$u(x+h) = u(x) + h(3u(x)+2).$$

The last equation is a finite-difference equation, and solving this equation gives an approximate solution to the differential equation.

Example: The heat equation [edit]

Consider the normalized heat equation in one dimension, with homogeneous Dirichlet boundary conditions

$$U_t = U_{xx}$$
$$U(0,t) = U(1,t) = 0 \text{ (boundary condition)}$$
$$U(x,0) = U_0(x) \text{ (initial condition)}$$

One way to numerically solve this equation is to approximate all the derivatives by finite differences. We partition the domain in space using a mesh $x_0, ..., x_J$ and in time using a mesh $t_0, ..., t_N$. We assume a uniform partition both in space and in time, so the difference between two consecutive space points will be *h* and between two consecutive time points will be *k*. The points

$$u(x_j, t_n) = u_j^n$$

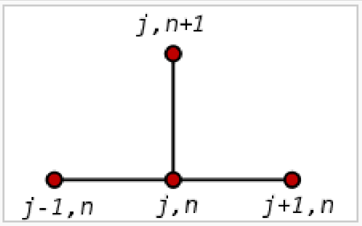
will represent the numerical approximation of $u(x_j, t_n)$.

Explicit method [edit]

Using a forward difference **at time** t_n and a second-order central difference for the space derivative at position x_j (FTCS) we get the recurrence equation:

$$\frac{u_j^{n+1} - u_j^n}{k} = \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{h^2}.$$

This is an explicit method for solving the one-dimensional heat equation.



The stencil for the most common explicit method for the heat equation.

We can obtain u_j^{n+1} from the other values this way:

$$u_j^{n+1} = (1-2r)u_j^n + ru_{j-1}^n + ru_{j+1}^n$$

where $r = k/h^2.$

So, with this recurrence relation, and knowing the values at time *n*, one can obtain the corresponding values at time *n*+1. u_0^n and u_J^n must be replaced by the boundary conditions, in this example they are both 0.

This explicit method is known to be numerically stable and convergent whenever $r \le 1/2$.[7] The numerical errors are proportional to the time step and the square of the space step:

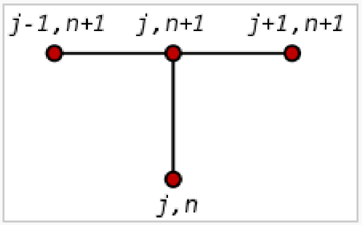
$$\Delta u = O(k) + O(h^2)$$

Implicit method [edit]

If we use the backward difference **at time** t_{n+1} and a second-order central difference for the space derivative at position x_j (The Backward Time, Centered Space Method "BTCS") we get the recurrence equation:

$$\frac{u_j^{n+1} - u_j^n}{k} = \frac{u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}}{h^2}.$$

This is an implicit method for solving the one-dimensional heat equation.



The implicit method stencil.

We can obtain u_j^{n+1} from solving a system of linear equations:

$$(1+2r)u_j^{n+1} - ru_{j-1}^{n+1} - ru_{j+1}^{n+1} = u_j^n$$

The scheme is always numerically stable and convergent but usually more numerically intensive than the explicit method as it requires solving a system of numerical equations on each time step. The errors are linear over the time step and quadratic over the space step:

$$\Delta u = O(k) + O(h^2).$$

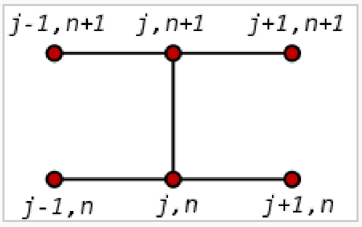
Crank–Nicolson method [edit]

Finally if we use the central difference at time $t_{n+1/2}$ and a second-order central difference for the space derivative at position x_j ("CTCS") we get the recurrence equation:

$$\frac{u_j^{n+1} - u_j^n}{k} = \frac{1}{2}\left(\frac{u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}}{h^2} + \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{h^2}\right).$$

This formula is known as the Crank–Nicolson method.

We can obtain u_j^{n+1} from solving a system of linear equations:



The Crank–Nicolson stencil.

$$(2+2r)u_j^{n+1} - ru_{j-1}^{n+1} - ru_{j+1}^{n+1} = (2-2r)u_j^n + ru_{j-1}^n + ru_{j+1}^n$$

The scheme is always numerically stable and convergent but usually more numerically intensive as it requires solving a system of numerical equations on each time step. The errors are quadratic over both the time step and the space step:

$$\Delta u = O(k^2) + O(h^2).$$

Usually the Crank–Nicolson scheme is the most accurate scheme for small time steps. The explicit scheme is the least accurate and can be unstable, but is also the easiest to implement and the least numerically intensive. The implicit scheme works the best for large time steps.

See also [edit]

- Finite element method
- Finite difference
- Finite difference time domain
- Stencil (numerical analysis)
- Finite difference coefficients
- Five-point stencil
- Lax–Richtmyer theorem
- Finite difference methods for option pricing
- Upwind differencing scheme for convection
- Central differencing scheme

v· t· e **Numerical partial differential equations by method** [show]

References [edit]

1. ^ Christian Grossmann; Hans-G. Roos; Martin Stynes (2007). *Numerical Treatment of Partial Differential Equations*. Springer Science & Business Media. p. 23. ISBN 978-3-540-71584-9.
2. ^ Arieh Iserlas (2008). *A first course in the numerical analysis of differential equations*. Cambridge University Press. p. 23. ISBN 9780521734905.
3. ^ *a* *b* Hoffman JD; Frankel S (2001). *Numerical methods for engineers and scientists*. CRC Press, Boca Raton.
4. ^ *a* *b* Jaluria Y; Atluri S (1994). "Computational heat transfer". *Computational Mechanics* **14**: 385–386. doi:10.1007/BF00377593.
5. ^ Majumdar P (2005). *Computational methods for heat and mass transfer* (1st ed.). Taylor and Francis, New York.
6. ^ Smith GD (1985). *Numerical solution of partial differential equations: finite difference methods* (3rd ed.). Oxford University Press.
7. ^ Crank, J. *The Mathematics of Diffusion*. 2nd Edition, Oxford, 1975, p. 143.

- K.W. Morton and D.F. Mayers, *Numerical Solution of Partial Differential Equations, An Introduction*. Cambridge University Press, 2005.
- Autar Kaw and E. Eric Kalu, *Numerical Methods with Applications*, (2008) [1]. Contains a brief,

engineering-oriented introduction to FDM (for ODEs) in Chapter 08.07.

- John Strikwerda (2004). *Finite Difference Schemes and Partial Differential Equations* (2nd ed.). SIAM. ISBN 978-0-89871-639-9.
- Smith, G. D. (1985), *Numerical Solution of Partial Differential Equations: Finite Difference Methods, 3rd ed.*, Oxford University Press
- Peter Olver (2013). *Introduction to Partial Differential Equations*. Springer. Chapter 5: Finite differences. ISBN 978-3-319-02099-0..
- Randall J. LeVeque, *Finite Difference Methods for Ordinary and Partial Differential Equations*, SIAM, 2007.

External links [edit]

- List of Internet Resources for the Finite Difference Method for PDEs

Various lectures and lecture notes [edit]

- Finite-Difference Method in Electromagnetics (see and listen to lecture 9)
- Lecture Notes Shih-Hung Chen, National Central University
- Finite Difference Method for Boundary Value Problems
- Numerical Methods for time-dependent Partial Differential Equations

Authority control	NDL: 00569934

Categories: Finite differences | Numerical differential equations

This page was last modified on 2 September 2015, at 11:19.

Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. By using this site, you agree to the Terms of Use and Privacy Policy. Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.

Privacy policy About Wikipedia Disclaimers Contact Wikipedia Developers Mobile view