Introduction to Computational Physics

Lecture: Finite Difference Methods - 1

Andreas Adelmann

Paul Scherrer Institut, Villigen
E-mail: andreas.adelmann@psi.ch

https://moodle-app2.let.ethz.ch/course/view.php?id=18025

8.0.1 Basic Concepts in Error Estimation I

A *Errors in given input data* Input data can be the results of measurements which have been influenced by statistical/or and systematical errors. Format conversions for example 12-bit analog digital to double precision.

B *Rounding errors during the computation* A rounding error occurs whenever a number, for example π, is shortened ("rounded off") to a fixed number of digits, or when a decimal fraction is converted to the binary form used in the computer. The limitation of floating-point $\Rightarrow$ loss of information that. Two typical cases are

1. If the computer cannot handle numbers which have more than, say, s digits, then the exact calculations of two s-digit numbers cannot be used in subsequent calculations.
2. In a floating-point computation, if a relatively small term b is added to a, then some digits of b are "shifted out", and they will not have any effect on future quantities that depend on the value of $a + b$.

8.0.1 Basic Concepts in Error Estimation II

C *Truncation Errors* These are errors committed when a limiting process is truncated (broken off) before one has come to the limiting value. A truncation error occurs, for example, when

- infinite series is broken off after a finite number of terms
- a derivative is approximated with a difference quotient (although in this case the term discretization error is better).
- a nonlinear function is approximated with a linear function, as in Newton's method.

Observe the distinction between truncation error and rounding error

D *Simplifications in the Mathematical Model* In a mechanical problem one might assume that a string in a pendulum has zero mass. In many other types of problems it is advantageous to consider a given body to be homogeneously filled with matter, instead of being built of atoms. The effects of such sources of error are usually more difficult to estimate than the types A, B, and C from above.

8.0.1 Basic Concepts in Error Estimation III

E *Human & Machine Errors* In all numerical work, one must expect that clerical errors, errors in hand calculation, and misunderstandings will occur. One should even be aware that textbooks (!), tables, and formulas may contain errors. When one uses computers, one can expect errors in the program itself, typing errors in entering the data, operator errors, and pure machine errors.

Examples: Errors of type E do occur, sometimes with serious consequences.

- The first American Venus probe was lost due to a program fault caused by the inadvertent substitution of a statement in a Fortran program of the form `DO 3 I = 1.3` for one of the form `DO 3 I= 1,3`. Erroneously replacing the comma "," with a dot "." converts the intended loop statement into an assignment statement!

8.0.1 Basic Concepts in Error Estimation IV

- A hardware error that got much publicity surfaced in 1994, when it was found that the INTEL Pentium processor gave wrong results for division with floating-point numbers of certain patterns. This was discovered by A Edelman during research on prime numbers and later fixed.

From a different point of view, one may distinguish between controllable and uncontrollable (or unavoidable) error sources.

- Errors of type A and D are usually considered to be uncontrollable in the numerical treatment
- Errors of type C are usually controllable. For example, the number of iterations in the solution of an algebraic equation, or the step size in a simulation, can be chosen either directly or by setting a tolerance.

8.0.1 Basic Concepts in Error Estimation V

- The rounding error in the individual arithmetic operation (type B) is, in a computer, controllable only to a limited extent, mainly through the choice between single and double precision. A very important fact is, however, that it can often be controlled by appropriate rewriting of formulas or by other changes of the algorithm

An example we compute compounded interest. Consider depositing the amount c every day in an account with an interest rate i compounded daily. With the accumulated capital, the total at the end of the year equals

$$c[(1+x)^n - 1]/x, \text{with } x = \frac{1}{n} \ll 1,$$

and $n = 365$. Using this formula does not give accurate results. The reason is that a rounding error occurs in computing $(1+x) = 1 + \bar{x}$ and low order bits of x are lost. For example, if $i = 0.06$, then $i/n = 0.0001643836$; in decimal arithmetic using six digits when this is

8.0.1 Basic Concepts in Error Estimation VI

added to one we get $fl(1 + i/n) = 1.000164$, and thus four low order digits are lost.
The problem then is to accurately compute $(1 + x)^n = \exp(n \log(1 + x))$.

$$\log(1+x) = \begin{cases} x & \text{if } (1+x) = 1 \\ x\frac{\log(1+x)}{(1+x)-1} & \text{otherwise} \end{cases} \tag{1}$$

can be shown to yield accurate results when $x \in [0, 3/4]$ and the computed value of $\log(1 + x)$ equals the exact result rounded (Goldberg, p 12).
To check this formula we recall that the base e of the natural logarithm can be defined by the limit

$$e = \lim_{n\to\infty} \left(1 + \frac{1}{n}\right)^n$$

8.0.1 Basic Concepts in Error Estimation VII

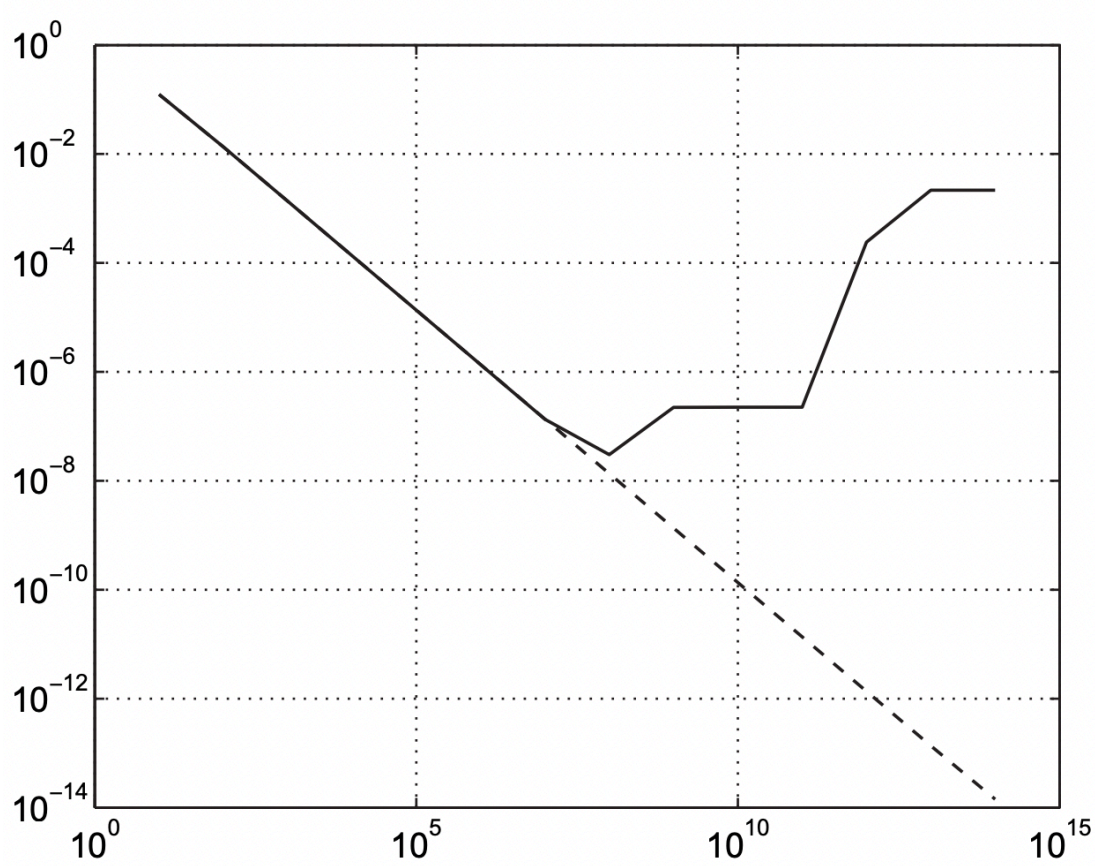


Figure: Computed values for $n = 10^p, p = 1 : 14$, of the sequences: solid line $|(1 + 1/n)^n - e|$; dashed line $|exp(nlog(1 + 1/n)) - e|$ using (1)

.

8.0.1 Basic Concepts in Error Estimation VIII

Remark (a fundamental insight from the above example can be expressed in the following way:)

mathematically equivalent *formulas or algorithms are not in general* **numerically equivalent**.

8.0.2 Very Basics of Error Propagation I

- in scientific computing the given input data are usually associated with uncertainties i.e. errors.
- the uncertainties/errors in the input will propagate and give rise to errors in the output. Here we will give a general feeling of how errors propagate. Error-propagation formulas are also of great interest in the planning and analysis of scientific experiments, see for references the script.
- note that rounding errors from each step in a calculation are also propagated to give errors in the final result.

8.0.2 Very Basics of Error Propagation II

We first consider two simple special cases of error propagation. For a sum of an arbitrary number of terms we get the following: in addition (and subtraction) a bound for the absolute errors in the result is given by the sum of the bounds for the absolute errors of the operands:

$$y = \sum_{i=1}^{n} x_i, \quad |\Delta y| = \sum_{i=1}^{n} |\Delta x_i|$$

In multiplication and division, an approximate bound for the relative error is obtained by adding the relative errors of the operands. More generally, for $y = x_1^{m_1}, x_2^{m_2}, \ldots, x_n^{m_n}$ (the relative error in a quantity is approximately equal to the absolute error in its natural logarithm)

$$\left|\frac{\Delta y}{y}\right| \lesssim \sum_{i=1}^{n} |m_i| \frac{\Delta x_i}{x_i}.$$

8.0.2 Very Basics of Error Propagation III

From this result a general formula for error propagation can be derived. Let the real-valued function $f = f(x_1, x_2, \ldots, x_n)$ be differentiable in a neighborhood of the point $x = (x_1, x_2, \ldots, x_n)$ with errors $\Delta x_1, \Delta x_2, \ldots, \Delta x_n$. Then it holds that

$$\Delta f \approx \sum_{i=1}^{n} \frac{\partial f}{\partial x_i} x_i. \tag{2}$$

The for the maximal error in $f(x_1, x_2, \ldots, x_n)$ we obtain the approximate upper bound

$$|\Delta f| \lesssim \sum_{i=1}^{n} \left| \frac{\partial f}{\partial x_i} \right| |x_i|. \tag{3}$$

where the partial derivatives are evaluated at x.

- to get a strict bound for , $|\Delta f|$ one should use in (3)

8.0.2 Very Basics of Error Propagation IV

- in most practical situations it suffices to calculate $|\partial f/\partial x_i|$ at x and then add a certain marginal amount (5 to 10 percent, say) for safety.
- if the Δx_i are large or if the derivatives have a large relative variation in the neighbourhood of x need the maximal values be used. (The latter situation occurs, for example, in a neighbourhood of an extremal point of $f(x)$.)

In practice, the trouble with formula (3) is that it often gives bounds which are too coarse. More realistic estimates are often obtained for the general case, which can be derived using probability theory:
Assume that the errors $\Delta x_1, \Delta x_2, \ldots, \Delta x_n$ are distributed with mean zero and standard deviations $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_{2n}$. Then the standard error ε of $f(x_1, x_2, \ldots, x_n)$ is given by the formula

$$\varepsilon \approx \left(\sum_{i=1}^{n} \left(\frac{\partial f}{\partial x_i} \right)^2 \varepsilon_i^2 \right)^{1/2}$$

8.0.2 Very Basics of Error Propagation V

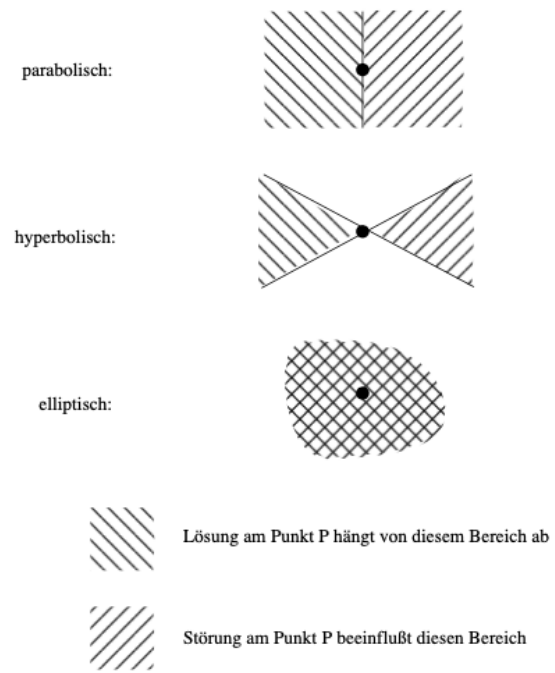
Remarks

- analysis of error propagation is more than just a means for judging the reliability of calculated results, it has an equally important function as a means for the planning of a calculation or scientific experiment.
- it can help in the choice of algorithm, and in making certain decisions during a calculation.
  - e.g. the choice of step length during a numerical integration.
  - number of iterations
- one can also shed some light on the degree to which it is advisable to obtain a new apparatus to improve the measurements
- measure how sensitive the output data are to small changes in the input data. In general, if `small` changes in the input data can result in `large` changes in the output data, we call the problem **ill-conditioned**; otherwise it is called **well- conditioned**. (The definition of large may differ from problem to problem depending on the accuracy of the data and the accuracy needed in the solution.)

8.1 Discretization in Space and Time I

We consider Partial Differential Equation (PDE), which can be classified in different groups

- parabolic: $D\frac{\partial^2\phi}{\partial^2 x} - \frac{\partial\phi}{\partial t} = 0$
- hyperbolic: $\frac{\partial^2\phi}{\partial^2 x} - \frac{1}{c}\frac{\partial^2\phi}{\partial^2 t} = 0$
- elliptic: $\nabla^2\phi = 0$



8.1 Discretization in Space and Time II

The canonical hyperbolic PDE is the wave equation:

$$\frac{\partial^2 \phi}{\partial t^2} = c^2 \frac{\partial^2 \phi}{\partial x^2} \tag{4}$$

The general solution to this is traveling waves in either direction:

$$\phi(x, t) = \alpha f_0(x - ct) + \beta g_0(x + ct) \tag{5}$$

Here f_0 and g_0 are set by the initial conditions, and the solution propagates f_0 to the right and g_0 to the left at a speed c.

8.1 Discretization in Space and Time III

The canonical elliptic PDE is the Poisson equation:

$$\nabla^2 \phi = f \tag{6}$$

Note that there is no time-variable here. This is a pure boundary value problem. The solution, ϕ is determined completely by the source, f, and the boundary conditions.
In contrast to the hyperbolic case, there is no propagation of information here. The potential, ϕ, is known instantaneously everywhere in the domain.

8.2 Linear Advection I

This simply propagates any initial profile to the right at the speed u. We will use linear advection as our model equation for numerical methods for hyperbolic PDEs.
Advection of concentration ϕ without diffusion or sources or sinks is

$$\frac{D\phi}{Dt} = \frac{\partial\phi}{\partial t} + \vec{u} \cdot \nabla\phi = 0 \tag{7}$$

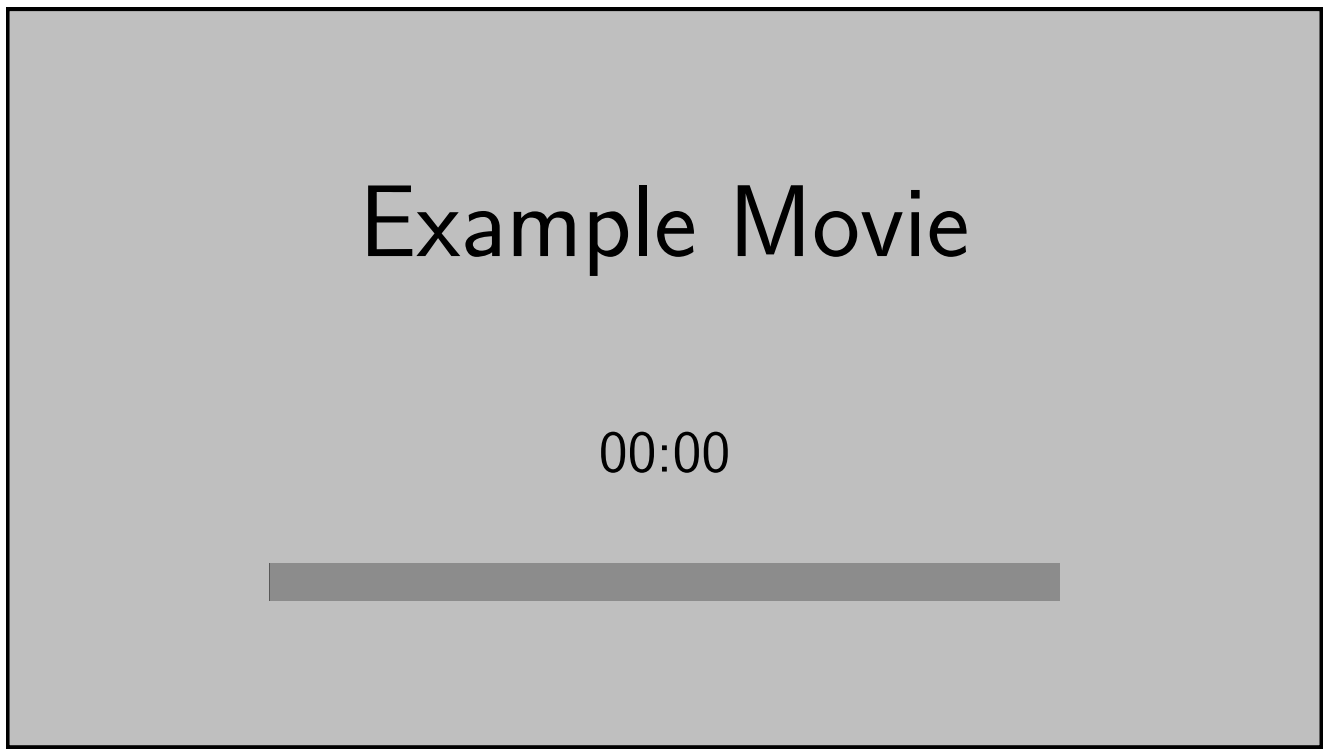
with D the Lagrangian derivative. Changes of ϕ are produced by the component of for example the wind in the same direction as gradients of ϕ. In one spatial dimension, x, with constant velocity, u, no diffusion and no sources or sinks, the linear advection equation for ϕ reads:

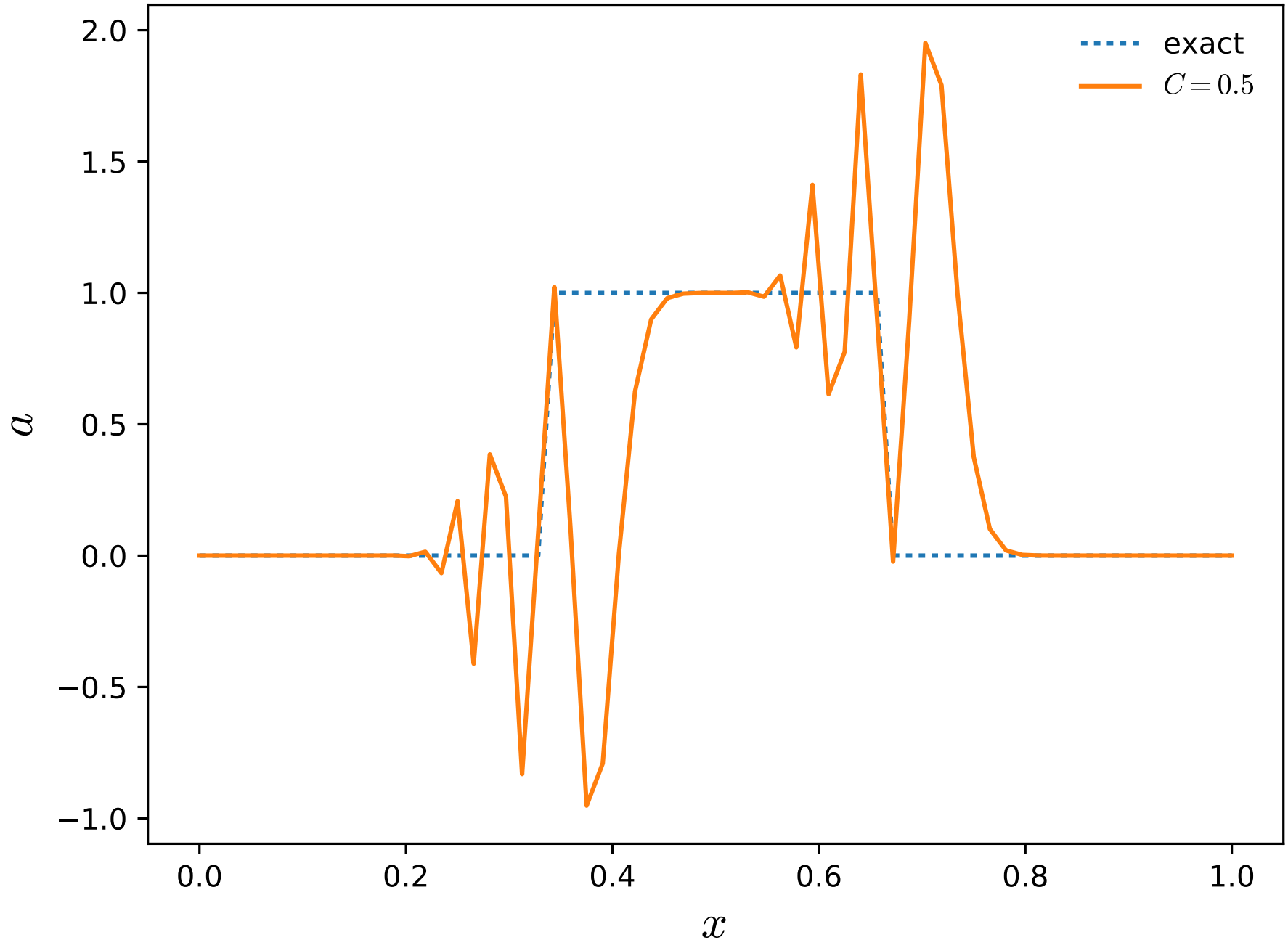
$$\frac{\partial\phi}{\partial t} + u\frac{\partial\phi}{\partial x} = 0. \tag{8}$$

8.2 Linear Advection II

Remark

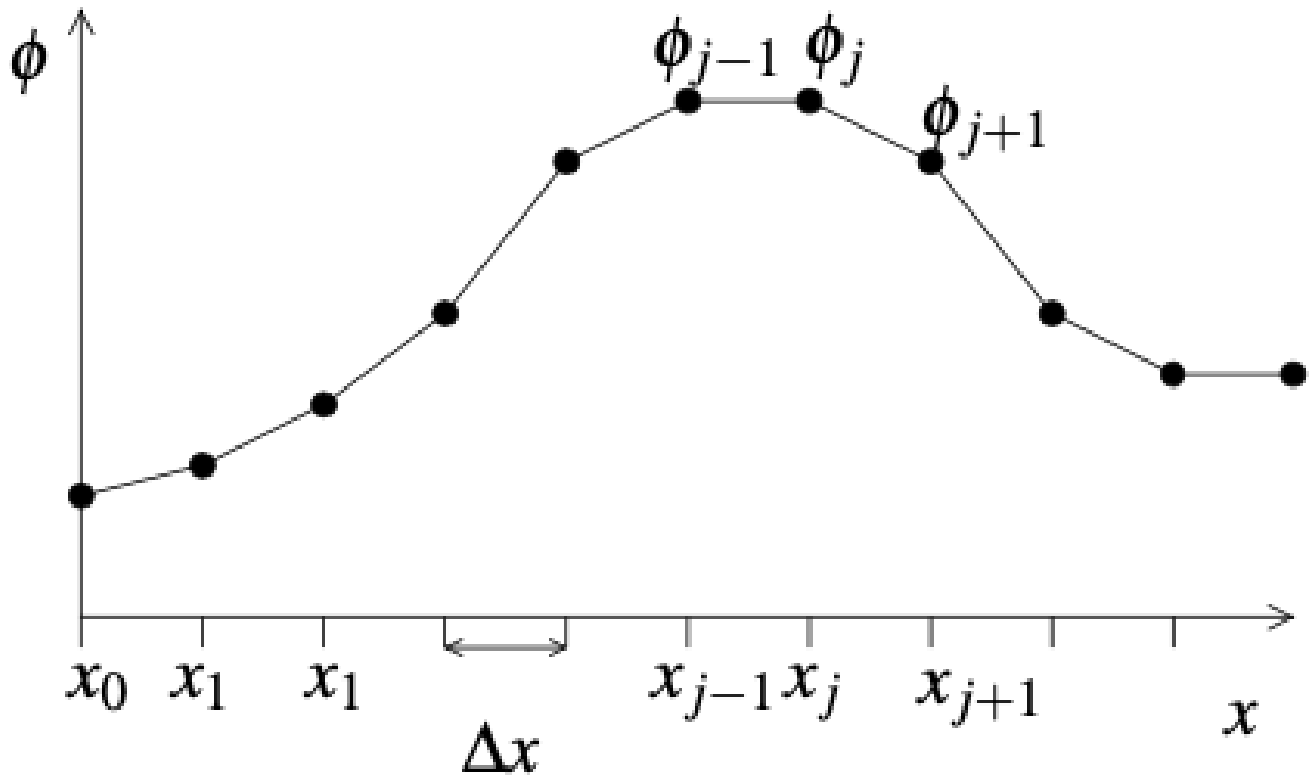
The exact solution of this equation is given by its initial value. Assuming that the initial value for eq. (8) is given as $\phi_0(x) = \phi(x, 0)$*, then the exact solution of eq. (8) is* $\phi_0(x - ut)$*.*




exact
$C = 0.5$
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
a
0.0
0.2
0.4
0.6
0.8
1.0
x

8.3 Forward in Time, Backward in Space (FTBS) I

Lets solve eq. 8 numerically.



- divide space into N equal intervals of size Δx with $x_j = j\Delta x$ for $j = 0, 1, \ldots, n_x$

8.3 Forward in Time, Backward in Space (FTBS) II

- divide time into time steps Δt with $t_n = n\Delta t$ for $n = 0, 1, 2, \ldots$
- define $\phi_j^{(n)} = \phi(x_j, t_n)$

Now we have to approximate the differential operator in space and time.

- forwards difference in time (Euler scheme): approximate $\partial\phi/\partial t$ at x_j, t_n by

$$\frac{\partial \phi}{\partial t} = \frac{\phi_j^{(n+1)} - \phi_j^{(n)}}{\Delta t} + \mathcal{O}(\Delta t)$$

- backward difference in space: approximate $\partial\phi/\partial x$ at x_j, t_n by

$$\frac{\partial \phi}{\partial x} = \frac{\phi_j^{(n)} - \phi_{j-1}^{(n)}}{\Delta x} + \mathcal{O}(\Delta x)$$

8.3 Forward in Time, Backward in Space (FTBS) III

Now substituting this into eq. 8 gives the FTBS discretization scheme:

$$\frac{\phi_j^{(n+1)} - \phi_j^{(n)}}{\Delta t} + u\frac{\phi_j^{(n)} - \phi_{j-1}^{(n)}}{\Delta x} + \mathcal{O}(\Delta x, \Delta t) = 0. \tag{9}$$

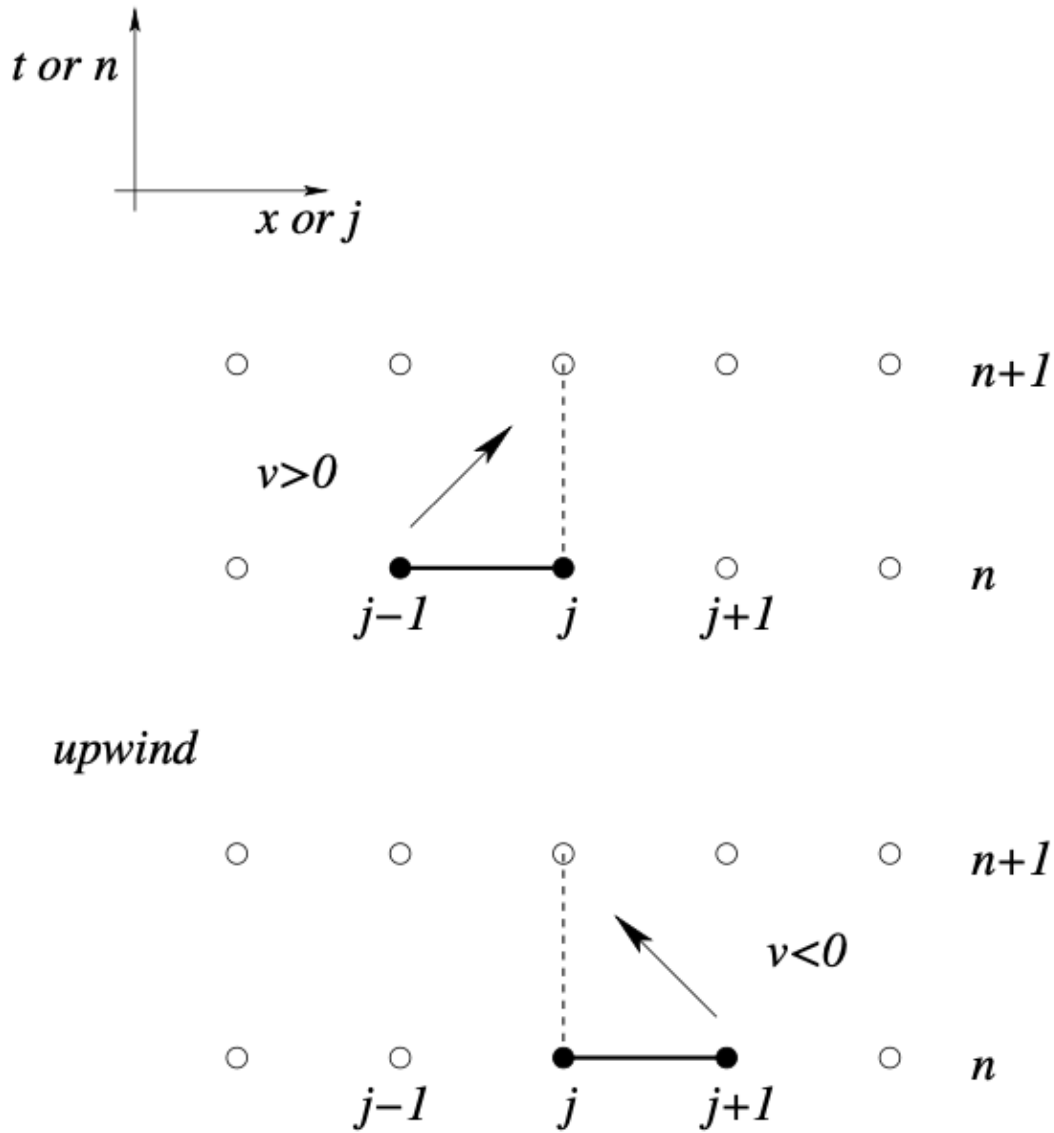
Now rearrange to have $\phi_j^{(n+1)}$ on the left hand side

$$\begin{aligned} \phi_j^{(n+1)} &= \phi_j^{(n)} - \frac{u\Delta t}{\Delta x}\left(\phi_j^{(n)} - \phi_{j-1}^{(n)}\right) + \mathcal{O}(\Delta x \Delta t, \Delta t^2) && (10) \\ &= \phi_j^{(n)} - c\left(\phi_j^{(n)} - \phi_{j-1}^{(n)}\right) + \mathcal{O}(\Delta x \Delta t, \Delta t^2). && (11) \end{aligned}$$

Here we also introduce the Courant number c as

$$c = \frac{u\Delta t}{\Delta x}.$$

8.3 Forward in Time, Backward in Space (FTBS) IV



8.3 Forward in Time, Backward in Space (FTBS) V

More in general, for a system of **linear hyperbolic equations** with state vector $\mathbf{\Phi}$ and flux-vector $\mathbf{F}$, the upwind scheme will take the form

$$\mathbf{\Phi}_j^{n+1} = \mathbf{\Phi}_j^n \pm \frac{\Delta t}{\Delta x}\left[\mathbf{F}_{j\mp 1}^n - \mathbf{F}_j^n\right] + \mathcal{O}(\Delta x \Delta t, \Delta t^2)$$

8.3.1 Order of Accuracy of FTBS I

Use Taylor series to derive the backward in space approximation in FTBS. The Taylor series of ϕ_{j-1} around ϕ_j is

$$\phi_{j-1} = \phi_j - \Delta x \phi_j^{'} + \frac{\Delta x^2}{2!}\phi_j^{''} + \mathcal{O}(\Delta x^3).$$

Now rearrange to find the backward in space approximation for $\partial\phi/\partial x = \phi_j^{'}$

$$\phi_j^{'} = \frac{\phi_j - \phi_{j-1}}{\Delta x} + \frac{\Delta x}{2!}\phi_j^{''} + \mathcal{O}(\Delta x^2).$$

Remark

- *the error is proportional to Δx i.e. first order accurate.*
- *the error behaves like $\phi^{''}$, which is the spatial term from the diffusion.*

8.4 Centred in Time, Centred in Space (CTCS) I

To solve nummerically:

$$\frac{\partial \phi}{\partial t} + u\frac{\partial \phi}{\partial x} = 0 \tag{12}$$

- approximate $\partial\phi/\partial t$ at x_j and t_n by a entered difference using ϕ_j^{n-1} and ϕ_j^{n+1}

$$\frac{\partial \phi_j^n}{\partial t} = \frac{\phi_j^{n+1} - \phi_j^{n-1}}{2\Delta t} + \mathcal{O}(\Delta t^2).$$

- approximate $\partial\phi/\partial x$ at x_j and t_n by a entered difference using ϕ_j^{n-1} and ϕ_j^{n+1}

8.4 Centred in Time, Centred in Space (CTCS) II

$$\frac{\partial \phi_j^n}{\partial x} = \frac{\phi_{j+1}^n - \phi_{j-1}^n}{2\Delta x} + \mathcal{O}(\Delta x^2).$$

Now we can substitute these two equations again into eq. 12 and rearrange to find ϕ_j^{n+1} and again using c

$$\phi_j^{n+1} = \phi_j^n - c\left(\phi_{j+1}^{n-1} - \phi_{j-1}^{n-1}\right) + \mathcal{O}(\Delta t^2, \Delta x^2).$$

Remark

This is a three-time-level formula (it involves values of ϕ at times t_{n-1}, t_n and t_{n+1}. To start the simulation, values of ϕ are needed at times t_0 and t_1. However, only $\phi(x, t_0)$ is available. Hence, an other scheme such as FTCS must be used to obtain $\phi^1 = (x, t_1)$

8.4.1 Order of Accuracy of CTCS I

Use Taylor series to derive the backward in space approximation in FTBS. The Taylor series of ϕ_{j-1} and ϕ_{j+1} around ϕ_j is

$$\phi_{j-1} = \phi_j - \Delta x \phi_j^{'} + \frac{\Delta x^2}{2!}\phi_j^{''} - \frac{\Delta x^3}{3!}\phi_j^{'''} + \mathcal{O}(\Delta x^4)$$

and

$$\phi_{j+1} = \phi_j + \Delta x \phi_j^{'} + \frac{\Delta x^2}{2!}\phi_j^{''} + \frac{\Delta x^3}{3!}\phi_j^{'''} + \mathcal{O}(\Delta x^4).$$

We again subtract to eliminate the largest unknown $\phi_j^{'''}$ and rearrange to find

$$\phi_j^{'} = \frac{\phi_{j+1} - \phi_{j-1}}{2\Delta x} + \mathcal{O}(\Delta x^2).$$

8.5 Implicit and Explicit Schemes I

- Explicit values at the next time-step are determined from values at the current (and previous) time-step. Abstract:

$$\phi^{n+1} = F(\phi^n)$$

- Implicit values at the next time-step are determined from values at the next time-step. Abstract: solve

$$G(\phi^n, \phi^{n+1}) = 0, \text{ to find } \phi^{n+1}.$$

Remark

- *implicit methods require an extra computation (harder to implement)*
- *implicit methods are used if the problems is stiff (explicit method would requires impractically small time steps)*
- *it takes much less computational time to use an implicit method with larger time steps*

8.6 Backward in Time, Centred in Space (BTCS) I

As an example of an implicit scheme we look at BTCS

$$\phi_j^{n+1} = \phi_j^n - \frac{c}{2}\left(\phi_{j+1}^{n+1} - \phi_{j-1}^{n+1}\right)$$

with periodic boundary conditions i.e. $\phi_0 = \phi_N$. We get the boundaries

$$\phi_0^{n+1} = \phi_0^n - \frac{c}{2}\left(\phi_1^{n+1} - \phi_{N-1}^{n+1}\right) \text{ and}$$

$$\phi_{N-1}^{n+1} = \phi_{N-1}^n - \frac{c}{2}\left(\phi_0^{n+1} - \phi_{N-2}^{n+1}\right).$$

The only change here is that the righthand side is evaluated at the new timelevel, $n+1$. Setting $C = c/2$, we can write this as a linear system with coupled equations:

$$-C\phi_{j-1}^{n+1} + \phi_j^{n+1} + -C\phi_{j+1}^{n+1} = \phi_j^n. \tag{13}$$

8.6 Backward in Time, Centred in Space (BTCS) II

If we use periodic boundary conditions, then point 0 and $N-1$ are identical, so we only need to update one of these. Taking $\phi_0^{n+1} = \phi_{N-1}^{n+1}$, our system in matrix form appears as:

$$\begin{pmatrix} 1 & C & 0 & 0 & 0 & & -C \\ -C & 1 & C & 0 & 0 & & 0 \\ & -C & 1 & C & 0 & & 0 \\ & & -C & 1 & C & & 0 \\ & & & \ddots & \ddots & & \\ 0 & & & & -C & 1 & C \\ C & 0 & \dots & & 0 & -C & 1 \end{pmatrix} \begin{pmatrix} \phi_0^{n+1} \\ \phi_1^{n+1} \\ \phi_2^{n+1} \\ \phi_3^{n+1} \\ \vdots \\ \phi_{N-2}^{n+1} \\ \phi_{N-1}^{n+1} \end{pmatrix} = \begin{pmatrix} \phi_0^{n} \\ \phi_1^{n} \\ \phi_2^{n} \\ \phi_3^{n} \\ \vdots \\ \phi_{N-2}^{n} \\ \phi_{N-1}^{n} \end{pmatrix} \tag{14}$$

8.6 Backward in Time, Centred in Space (BTCS) III

Remark

- *this requires a matrix solve—this makes implicit methods generally more expensive than explicit methods.*
- *however, stability analysis would show that this implicit discretization is stable for any choice of c. (But one must not confuse stability with accuracy—the most accurate solutions with this method will still have a small c.*
- *Also note that the form of the matrix will change depending on the choice of boundary conditions.*

8.7 Numerical Analysis of Advection Schemes I

Introduction & Definitions:

What is Numerical Analysis and why is it done?

- numerical analysis involves mathematically analyzing numerical methods in order to predict how they will behave when they are used.
- numerical analysis is important because
  - model development by trial and error is very time consuming
  - we cannot test our models for every possible situation.
  - ideally we need evidence that they will work for all situations
  - we gain insight into how numerical methods work and so how to design better ones

Some definitions:

- **Convergence:** A finite difference scheme is convergent if the solutions of the scheme converge to the solutions of the PDE as Δx and Δt tend to zero.

8.7 Numerical Analysis of Advection Schemes II

Introduction & Definitions:

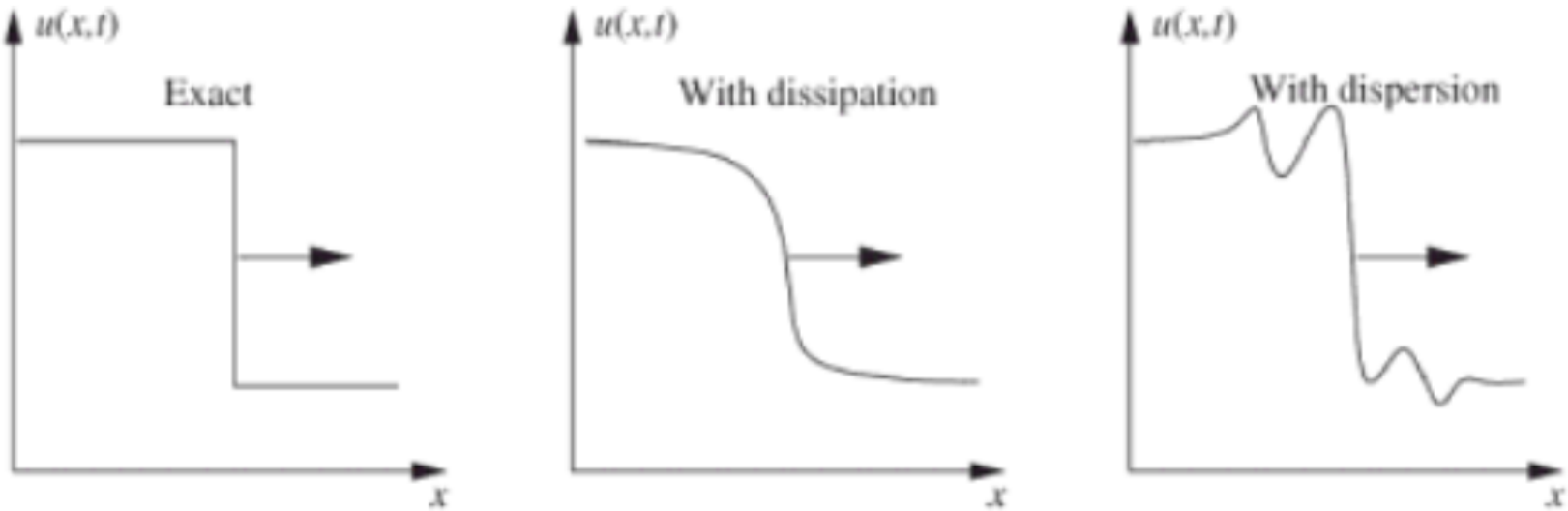
- **Consistency:** A finite difference scheme is consistent with a PDE if the errors in approximating all of the terms tend to zero as Δx and/or Δt tend to zero. (Terms of the finite difference scheme are typically analysed using Taylor series.)
- **Order of accuracy:** Error $\propto \Delta x^n$ – error is $\mathcal{O}(\Delta x^n)e$ – means scheme is nth order accurate. Errors of an nth order scheme converge to zero with order n.
- **Stability:** Errors do not tend to infinity for any number of time steps. Stability is typically proved using von-Neumann stability analysis.
  a) Conditionally stable - if stable only for a sufficiently small time-step
  b) Unconditionally stable - if stable for any time-step
  c) Unconditionally unstable - if unstable for any time-step
- **Conservation:** If, eg mass, energy, potential vorticity are conserved by the PDEs, are they conserved by the numerical scheme?

8.7 Numerical Analysis of Advection Schemes III

Introduction & Definitions:

- **Boundedness:** If the initial conditions are bounded between values a and b then a bounded solution will remain bounded between a and b for all time.
- **Monotonicicity:** Monotone schemes do not generate new extrema or amplify existing extrema. If the initial conditions are monotonic then they will remain monotonoic after the action of a monotonic numerical scheme.

8.8 Dissipation & Dispersion I



- Numerical diffusion (also known as numerical dissipation) is the tendency of a space discretization operator to smooth out sharp corners, gradients, or discontinuities. For example, instead of having a sharp interface, such as a square wave, over 1 cell, it will spread over a few cells.

8.8 Dissipation & Dispersion II

- Numerical dispersion results from the dependence of wave speed on wavelength. For example, for any wavelength, the exact solution can be represented by an approximate Fourier superposition with harmonics of many different wavelengths. Dispersion makes the components travel at different speeds, because of the varying wavelengths, so that over time the harmonics separate from the true solution.

8.9 Lax-Equivalence Theorem I

The Lax equivalence theorem is the fundamental theorem in the analysis of finite difference methods for the numerical solution of partial differential equations. It states that for a consistent finite difference method for a well-posed linear initial value problem, the method is convergent if and only if it is stable:

$$\text{consistency} + \text{stability} \iff \text{convergence}$$

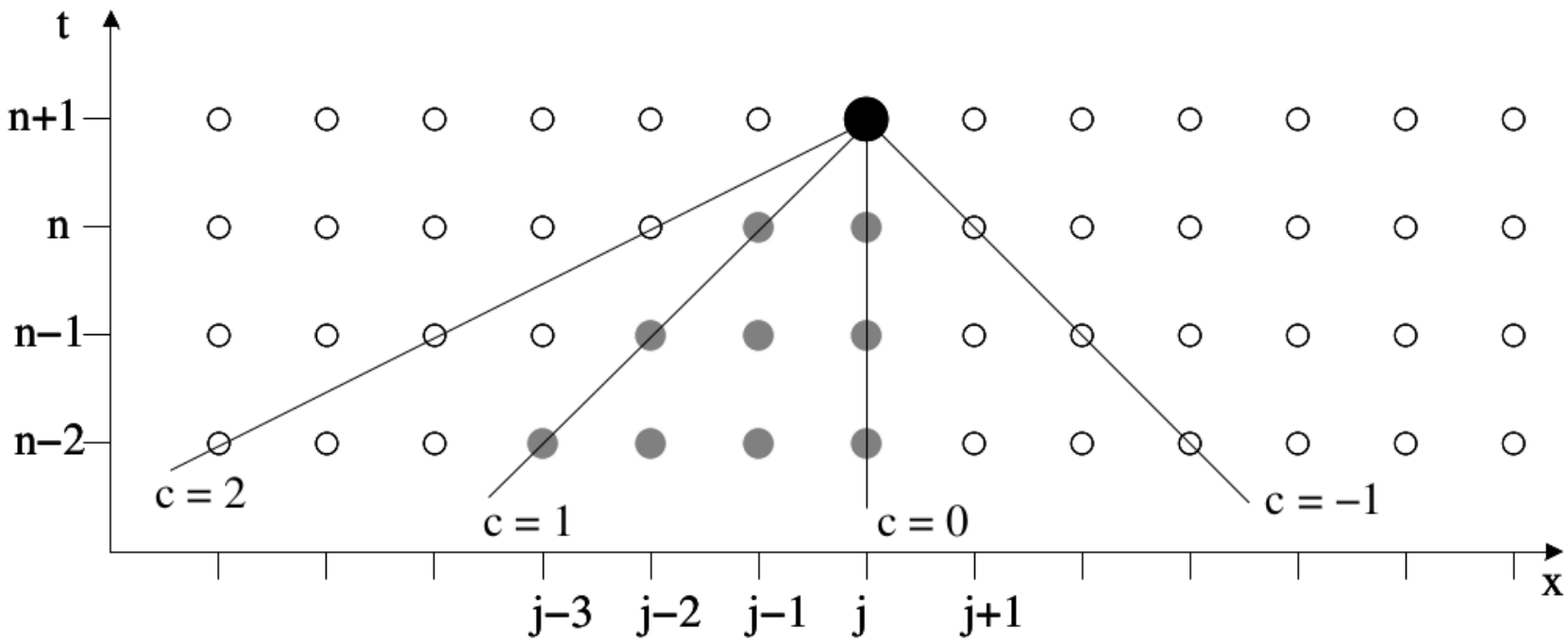
So if you can show that the finite difference approximations for each of the terms is at least first-order accurate and you can show that the scheme is stable, then you know that solutions to the finite difference scheme will converge to solutions of the PDE. This is why we study order of accuracy and stability.

8.10 Domain of Dependence (DoD) I

FTBS & CTCS

The Domain of Dependence (DoD) of the solution of a PDE at position x and at time t is the set of points at a previous time that influence **the solution** at position x and at time t. Again we look at

$$\phi_t + u\phi_x = 0 \text{ with solution } \phi(x,t) = (x - ut, 0)$$

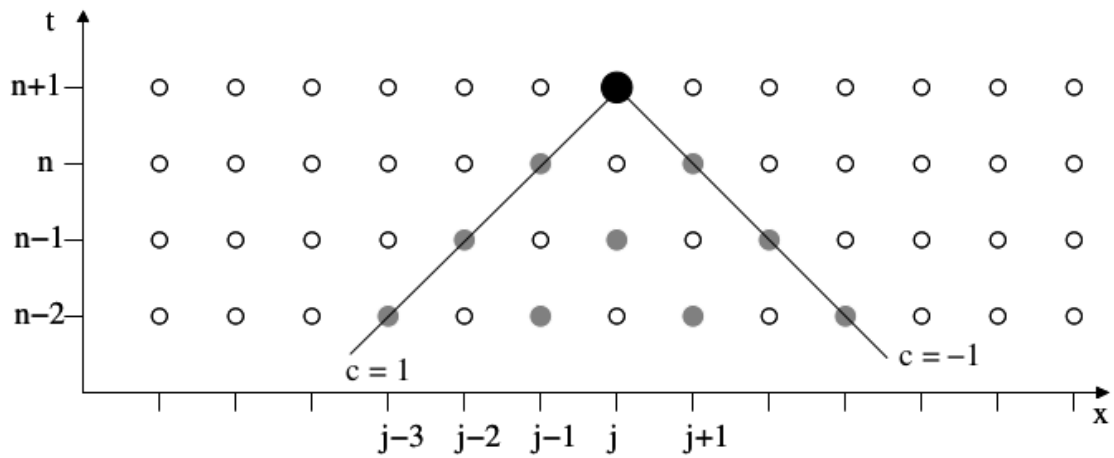


8.10 Domain of Dependence (DoD) II

FTBS & CTCS

The Domain of Dependence (DoD) of the solution of a PDE at position x and at time t is the set of points at a previous time that influence **the solution** at position x and at time t. Again we look at

$$\phi_t + u\phi_x = 0 \text{ with solution } \phi(x,t) = (x - ut, 0)$$

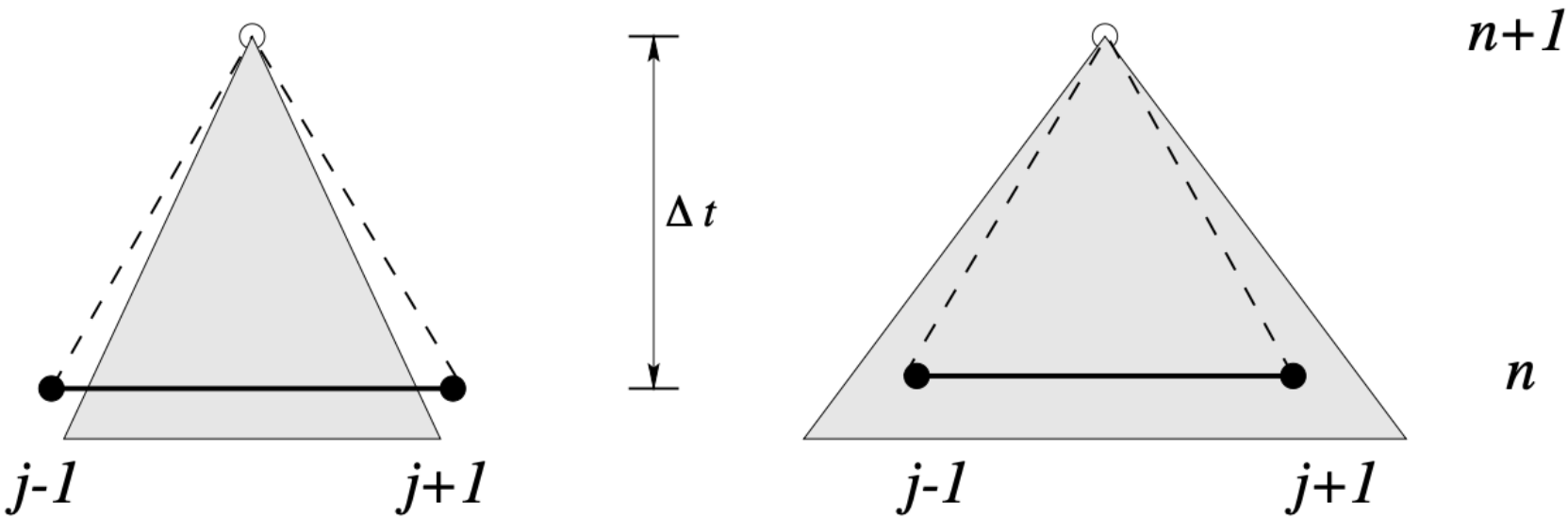


Interpretation:

8.10 Domain of Dependence (DoD) III

FTBS & CTCS

- Schematic diagram of Courant stable and unstable choices of time-steps Δt
- The two dashed lines limit the numerical domain of dependence of the solution at x_j^{n+1}, while the shaded area represents the physical domain of dependence
- Stability is achieved when the first one is larger than the second one.



8.11 Courant-Friedrichs-Lewy (CFL) criterion I

The DoD of the numerical solution should include the DoD of the original PDE.

- the CFL criterion is necessary but not sufficient
- for linear advection, the domain of dependence of the differential equation at $(j\Delta x, n\Delta t)$ is the straight line of slope $1/u$ through $(j\Delta x, n\Delta t)$ for $t \leq n\Delta t$ in the (x, t) plane

The CFL condition then has the following form:

$$C = \frac{u\,\Delta t}{\Delta x} \leq C_{\max}$$

while $C_{\max}$ is dependent on the on the solution scheme.

Remark

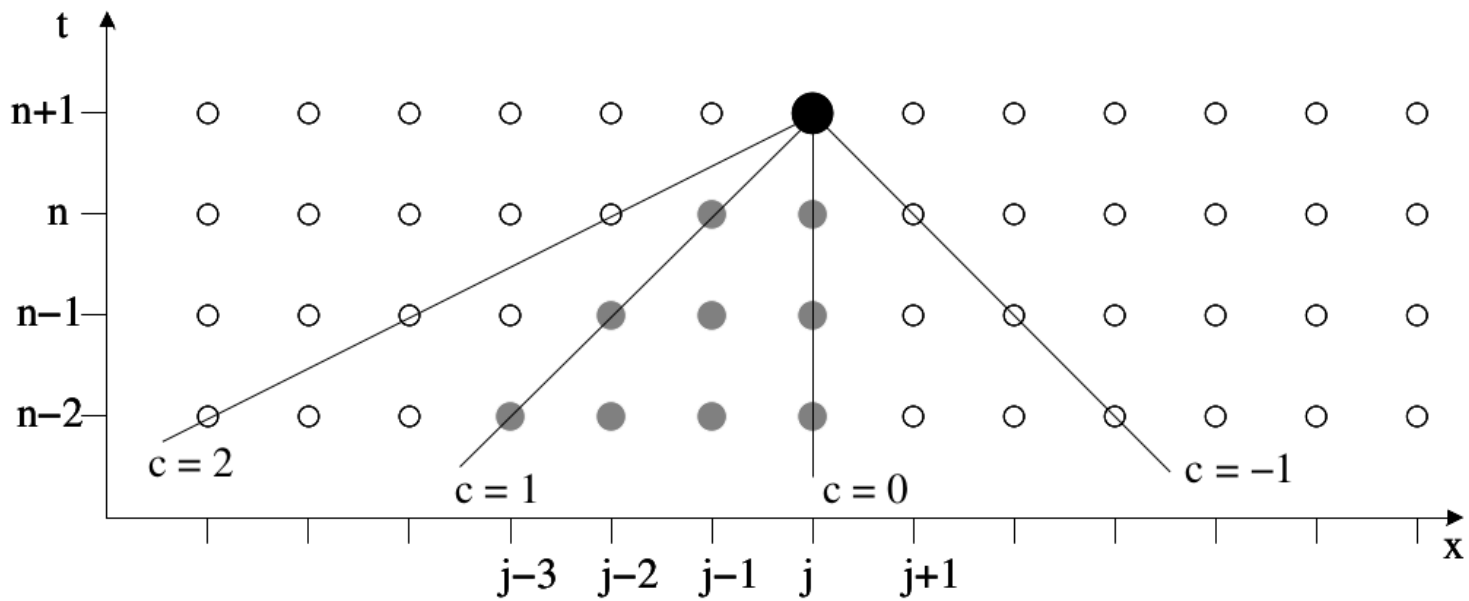
If an explicit (time-marching) solver is used then typically $C_{\max} = 1$. Implicit (matrix) solvers are usually less sensitive to numerical instability and so larger values of $C_{\max}$ may be tolerated.

8.12 Domain of Dependence of the FTBS Scheme I

The FTBS scheme reads:

$$\phi_j^{(n+1)} = \phi_j^{(n)} - c\left(\phi_j^{(n)} - \phi_{j-1}^{(n)}\right) \tag{15}$$

Clearly $\phi_j^{(n)}$ depends on $\phi_j^{(n-1)}$ and $\phi_{j-1}^{(n-1)}$. Subsequently these depend on $\phi_j^{(n-2)}$ and $\phi_{j-2}^{(n-2)}$. Lets set $c = -1, 0, 1, 2$ and draw DoD using the PDF.



8.12 Domain of Dependence of the FTBS Scheme II

- $c = 0$ stable (straight line)
- $c = 1$ stable captured by the numerical scheme
- $c = 2, -1$ unstable outside of the numerical DoD

The numerical DoD contains the physical DoD for

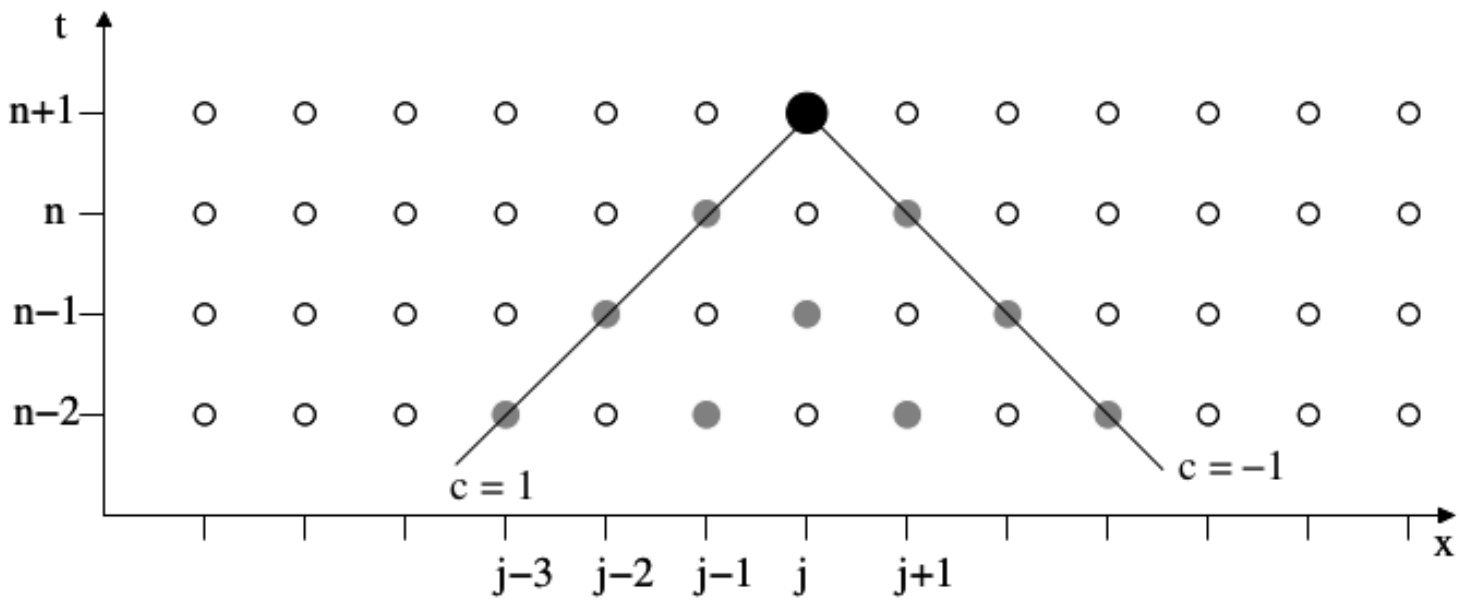
$$0 \leq c \leq 1.$$

The FTBS is unstable otherwise.

8.13 Domain of Dependence of the CTCS Scheme I

The CTCS scheme reads:

$$\phi_j^{(n+1)} = \phi_j^{(n-1)} - c\left(\phi_{j+1}^{(n)} - \phi_{j-1}^{(n)}\right) \tag{16}$$



8.13 Domain of Dependence of the CTCS Scheme II

Except at the initial time, the solution is found on two sets of points that are not coupled. The solution can oscillate between two unrelated solutions. This is a manifestation of the computational mode of CTCS. The domain of dependence and the CFL criterion can tell us when some schemes are unstable. How about proving stability?

The numerical DoD contains the physical DoD for

$$-1 \leq c \leq 1.$$

The CTCS is unstable otherwise.

8.14 Von-Neumann Stability Analysis I

Assume that the solution at time level $n+1$ can be represented as an amplification factor, A, multiplied by the solution at time-level n:

$$\phi^{(n+1)} = A\phi^{(n)}, \quad A \in \mathbb{C}, \quad |A|^2 = AA^\dagger$$

The amplification factor will give use the following information:

- $|A|^2 < 1, \quad \forall k, \Delta x$: stable and damping
- $|A|^2 = 1, \quad \forall k, \Delta x$: neutral stable
- $|A|^2 > 1, \quad \forall k, \Delta x$: unstable and amplyfying

Remark

For linear advection $A \in \mathbb{C}$ since the solution changes location every timestep time-step.

8.14 Von-Neumann Stability Analysis II

Remember the solution of a PDE in one spatial dimension can be expressed as a sum of Fourier modes

$$\phi = \sum_{k=-\infty}^{\infty} A_k e^{ikx}$$

expressing the wave number with k. We will now proceed by on a uniform grid $x = j\Delta x$ and substitute

$$\phi_j^n = A^n e^{ikj\Delta x}$$

into the equation(s) for a linear numerical scheme.

8.14.1 Von-Neumann Stability Analysis of FTBS I

The FTBS reads

$$\phi_j^{(n+1)} = \phi_j^{(n)} - c\left(\phi_j^{(n)} - \phi_{j-1}^{(n)}\right)$$

and now substituting $A^n e^{ikj\Delta x}$ for ϕ_j^n we get

$$A^{n+1} e^{ikj\Delta x} = A^n e^{ikj\Delta x} - cA^n\left(e^{ikj\Delta x} - e^{ik(j-1)\Delta x}\right)$$

Now powers of A cancel and we find in terms of c and $k\Delta x$

$$A = 1 - c(1 - e^{-ik\Delta x})$$

Substituting now $e^{-ik\Delta x} = \cos k\Delta x - i \sin k\Delta x$ and calculating $|A|^2$ we obtain finally

$$|A|^2 = 1 - 2c(1-c)(1 - \cos k\Delta x)$$

For what Δt or c is FTBS stable?

8.14.1 Von-Neumann Stability Analysis of FTBS II

$$
\begin{aligned}
|A| \leq 1 \quad &\Longleftrightarrow \quad |A|^2 - 1 \leq 0 \\
&\Longleftrightarrow \quad -2c(1-c)(1-\cos k\Delta x) \leq 0 \\
&\Longleftrightarrow \quad c(1-c)(1-\cos k\Delta x) \leq 0
\end{aligned}
$$

We know that $1 - \cos k\Delta x \geq 0$ for all $k\Delta x$ so FTBS is stable when

$$c(1-c) \geq 0 \iff 0 \leq c \leq 1.$$

We have proved that FTBS is unstable if $u < 0$ or if $\frac{u\Delta t}{\Delta x} > 1$.

Remark

Solution is depending on u!

8.14.2 Von-Neumann Stability Analysis of CTCS I

The CTCS reads

$$\phi_j^{(n+1)} = \phi_j^{(n-1)} - c\left(\phi_{j+1}^{(n)} - \phi_{j-1}^{(n)}\right)$$

Following the previous avenue we get

$$\Rightarrow A = -ic\sin k\Delta \pm \sqrt{1 - c^2 \sin^2 k\Delta x}$$

- case 1

$$\begin{aligned} |c| \leq 1 \quad &\Rightarrow \quad c^2 \sin^2 k\Delta x \leq 1 \\ &\Rightarrow \quad |A|^2 = 1 \\ &\Rightarrow \quad \text{Solution stable and not damping} \end{aligned}$$

8.14.2 Von-Neumann Stability Analysis of CTCS II

- case 2

$$
\begin{aligned}
|c| > 1 \quad &\Rightarrow \quad c^2 \sin^2 k\Delta x > 1 \text{ for some } k\Delta x \\
&\Rightarrow \quad |A|^2 = \left(c \sin k\Delta \pm \sqrt{c^2 \sin^2 k\Delta x - 1} \right)^2 \\
&\Rightarrow \quad \text{At least one of the roots has } |A| > 1 \\
&\Rightarrow \quad \text{the solution is unstable}
\end{aligned}
$$

- the CTCS is conditionally stable
- it is stable for $|c| \leq 1$

8.14.2 Von-Neumann Stability Analysis of CTCS III

Remark

Regardless of c, there are always two possible values of A. This means that there will always be two possible solutions;

- *a realistic solution (the physical mode)*
- *the un-realistic, oscillating solution*

The latter one is also called the spurious computational mode. This will contaminate the solution and behave in an un-realistic way.

8.15 Conservation I

For a solution ϕ. advected with velocity u: $(\phi_t + u\phi_x)$ we show that the total 'mass' of ϕ is conserved. Define the total mass to be

$$M = \int_0^1 \phi dx \tag{17}$$

and assume that ϕ has periodic boundary condition such that $\phi(0,t) = \phi(1,t)\ \forall t$.

$$\begin{aligned} \frac{dM}{dt} &= \frac{d}{dt}\left(\int_0^1 \phi dx\right) = \int_0^1 \frac{d\phi}{dt} dx \\ &= -\int_0^1 u\frac{d\phi}{dx} dx = -u\int_0^1 d\phi = -u\phi|_0^1 = 0 \quad \square \end{aligned}$$

Now lets us insert the FTBS scheme for advection:

$$\phi_j^{(n+1)} = \phi_j^{(n)} - c\left(\phi_j^{(n)} - \phi_{j-1}^{(n)}\right)$$

8.15 Conservation II

Idea: calculate $M^{(n+1)}$ from $M^{(n)}$

$$
\begin{aligned}
M^{(n+1)} &= \sum_{j=1}^{N} \Delta x \phi_j^{(n+1)} = \Delta x \sum_{j=1}^{N} \left[\phi_j^{(n)} - c \left(\phi_j^{(n)} - \phi_{j-1}^{(n)} \right) \right] \\
&= M^{(m)} - \Delta x \cdot c \left(\sum_{j=1}^{N} \phi_j^{(n)} - \sum_{j=1}^{N} \phi_{j-1}^{(n)} \right) \\
&= M^{(m)} - \Delta x \cdot c \left(\sum_{j=1}^{N} \phi_j^{(n)} - \sum_{j=0}^{N-1} \phi_j^{(n)} \right) \\
&= M^{(m)} - \Delta x \cdot c \left(\phi_N^{(n)} - \phi_0^{(n)} \right) = M^{(n)} \quad \square
\end{aligned}
$$

Exercice: show that the variance of ϕ is also conserved.

8.16 Waves, Dispersion and Dispersion Errors I

Inaccurate dispersion of numerical methods is a common source of errors. This is why we look now at dispersion and dispersion errors of numerical methods.
A travelling wave (in 1D) can be described by the equation

$$y(x,t) = A\sin(kx - \omega t). \tag{18}$$

where $y(x,t)$ is the height of the wave at position x, at time t, A is the amplitude of the wave k is the wavenumber (number of whole waves between 0 and 2π) ω is the angular wave frequency - the number of complete oscillations in time 2π at a fixed point
Question: How do variations in k and ω influence the wave and its propagation according to the equation 18?
Demo: $a_1\sin(xk_1 - t\omega_1) + a_2\sin(xk_2 - t\omega_2)$

8.16 Waves, Dispersion and Dispersion Errors II

Definition (Dispersion)

Dispersion occurs when waves of different frequencies propagate at different velocities.

Remark

Hence, for a wave function to propagate without changing shape, all of the Fourier modes must propagate at the same velocities.

8.17 Phase Velocity of Linear Advection I

Consider

$$\phi_t + u\phi_x = 0$$

and its analytic solution

$$\phi(x,t) = \phi(x - ut, 0)$$

under the action of the linear advection equation, waves propagate with phase speed u. (linear advection is non-dispersive since u does not depend on k). We know

- if we multiply a single Fourier mode e^{ikx} by $e^{-iku\Delta t}$ then the mode will move a distance $u\Delta t$
- the amplification factor A for exact linear advection is $A = e^{-iku\Delta t}$

Therefore if a numerical method has an amplification factor $e^{-i\alpha}$ for wavenumber k then the phase velocity of the numerical waves if wavenumber k will be $\alpha/(k\Delta t)$

8.18 Phase Velocity and dispersion Errors of CTCS I

The amplification factor of CTCS is

$$A = ic \sin \Delta x k \pm \sqrt{1 - c^2 \sin^2 k\Delta x}$$

where $c = u\Delta t/\Delta x$. Therefore we can find the phase-speeds of the numerical waves relative to the analytic waves:

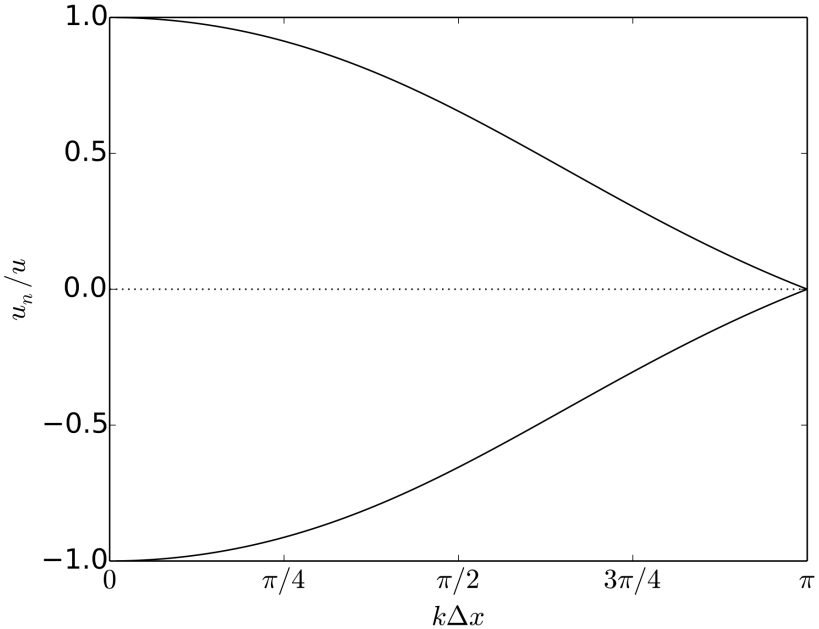
$$\frac{u_n}{u} = \frac{1}{uk\Delta t} \tan^{-1} \frac{c \sin k\Delta x}{\pm\sqrt{1 - c^2 \sin^2 k\Delta x}}$$

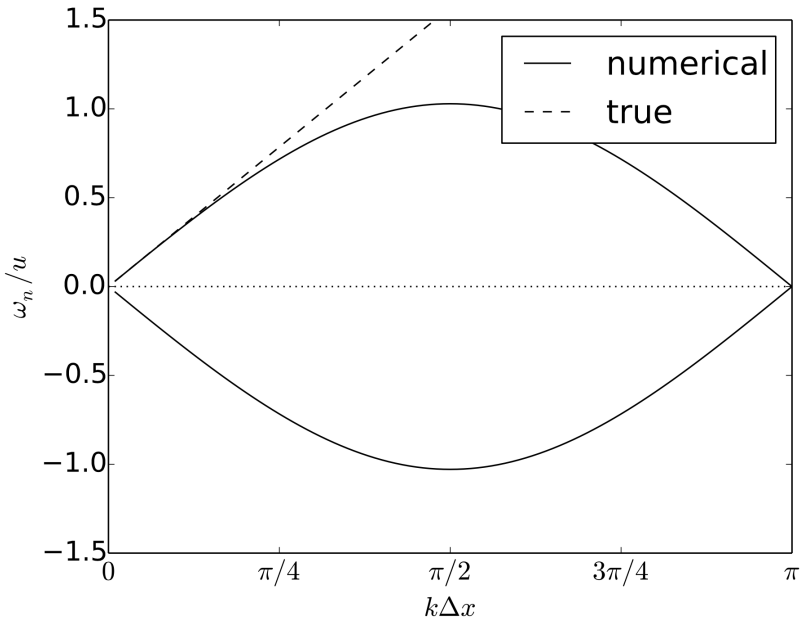
with $u\Delta t = c\Delta x$ and $\sin \gamma = c \sin k\Delta x$

$$\frac{u_n}{u} = \pm \frac{\gamma}{ck\Delta x}.$$

There are two possible phase-speeds for each mode. These can be plotted against $k\Delta x$ to find out how waves propagate when advected by CTCS. A plot of u_n/u against $k\Delta x$ (or $\omega = ku$ against k) is called a dispersion relation. The dispersion relation for CTCS for $c = 0.4$ is given below:

8.18 Phase Velocity and dispersion Errors of CTCS II



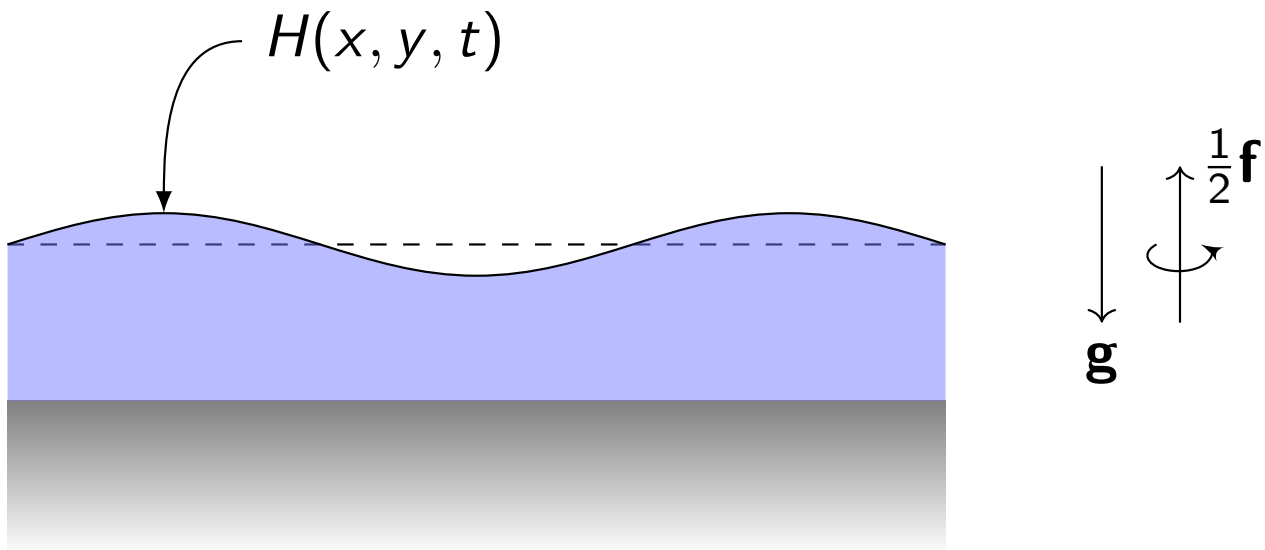


- Two solutions: - physics ($u_n/u > 0$), computational mode ($u_n/u < 0$)
- all waves propagate too slowly
- For the physical mode, when waves are well resolved (small k or small Δx), they propagate at nearly the correct speed (this is why it is called the physical mode)

8.19 Shallow water equations I

We are now going to look at the shallow water equations, where we have some water shallow relative to its horizontal extent. This is actually quite a good approximation for the ocean — while the Atlantic is around 4 km deep, it is several thousand kilometers across. So the ratio is just like a piece of paper.

Suppose we have a shallow layer of depth $z = h(x, y)$ with $p = p_0$ on $z = h$.



8.19 Shallow water equations II

- starting point are the incompressible Navier-Stokes equation Eq. (8.6) in the script together with the condition of incompressibility $\nabla \cdot \mathbf{v} = 0$.
- we assume for a moment that both the surface of the water as well as the ground are flat, we can choose our coordinates such that the surface of the water is at $z = 0$ and the ground at $z = -H_0$.

Now, also including potential waves, we can decompose the total depth of the fluid

$$H(x, y, t) = H_0 + H'(x, y, t), \tag{19}$$

where H' is the elevation of the fluid relative to the average depth H_0. Furthermore, we assume that $H' \ll H_0$. Neglecting any viscosity effects ($\eta = 0$) but including the gravitational force $\mathbf{F} = (0, 0, g)$, the single

8.19 Shallow water equations III

components of the incompressible Navier-Stokes equation can be written as

$$\partial_t v_x + v_x \partial_x v_x + v_y \partial_y v_x + v_z \partial_z v_x = -\frac{1}{\rho} \partial_x p \tag{20}$$

$$\partial_t v_y + v_x \partial_x v_y + v_y \partial_y v_y + v_z \partial_z v_y = -\frac{1}{\rho} \partial_y p \tag{21}$$

$$\partial_t v_z + v_x \partial_x v_z + v_y \partial_y v_z + v_z \partial_z v_z = -\frac{1}{\rho} \partial_z p + g \tag{22}$$

$$\partial_x v_x + \partial_y v_y + \partial_z v_z = 0. \tag{23}$$

Incompressibility Equation (details see script) Integrating Eq. (23) over depth yields

$$0 = \int_{-H_0}^{H'} \nabla \cdot \mathbf{v} dz$$

8.19 Shallow water equations IV

since H' depends on the x and y coordinates. Due to the fact that there is no fluid above H' and below $-H_0$, we cannot have any normal flow to the two surfaces and get the additional boundary conditions

$$\begin{aligned} 0 &= v_z|_{z=-H_0} \\ 0 &= [-\partial_t H' + v_x \partial_x H' + v_y \partial_y H' - v_z]|_{z=H'}. \end{aligned}$$

Plugging these into the previous equation results in

$$0 = \partial_x(H \langle v_x \rangle_z) + \partial_y(H \langle v_y \rangle_z) + \partial_t H',$$

where $\langle \cdot \rangle_z = \int_{-H_0}^{H'} \cdot dz$ is the depth average. In the case $H' \ll H_0$, we finally find

$$0 = H_0(\partial_x \langle v_x \rangle_z + \partial_y \langle v_y \rangle_z) + \partial_t H'.$$

z-Momentum Equation (details see script) Next, we consider Eq. (22). Since the water is shallow, we expect that the change in z-velocity is

8.19 Shallow water equations V

small. In particular, we assume that the complete left-hand side is much smaller than the right-hand side and neglect it completely. This results in

$$\partial_z p = \rho g,$$

i.e. that the pressure distribution is *hydrostatic*. Consequently, for the x and y directions we find

$$\partial_x p = \rho g \partial_x H' \tag{24}$$
$$\partial_y p = \rho g \partial_y H'. \tag{25}$$

x-Momentum Equation (details see script) In the limit that $v_x \partial_x v_x$, $v_y \partial_y v_x$ and $v_z \partial_z v_x$ are small, we can integrate Eq. (20) over depth and find

$$\partial_t (H \langle v_x \rangle_z) = - \int_{-H_0}^{H'} \frac{1}{\rho} \partial_x p dz.$$

8.19 Shallow water equations VI

Since $\partial_x p = \rho g \partial_x H'$ [Eq. (24)],

$$\partial_t (H \langle v_x \rangle_z) = -gH\partial_x H'. \tag{26}$$

Again assuming that $H' \ll H_0$, we arrive at

$$\partial_t \langle v_x \rangle_z = -g\partial_x H'. \tag{27}$$

***y*-Momentum Equation (details see script)** Analogously as for the *x*-momentum equation, we arrive at

$$\partial_t \langle v_y \rangle_z = -g\partial_y H'. \tag{28}$$

8.20 Discretization Schemes I

The *Shallow Water Equations* consist of Eq. (62), Eq. (27) and Eq. (28). For convenience we change the notation and introduce $u = \langle v_x \rangle_z$, $v = \langle v_y \rangle_z$ and $\eta = H'$ such that we can write

$$\partial_t \eta = -H_0(\partial_x u + \partial_y v) \tag{29}$$
$$\partial_t u = -g\partial_x \eta \tag{30}$$
$$\partial_t v = -g\partial_y \eta. \tag{31}$$

Restricting ourselves further to the one-dimensional case results in

$$\partial_t \eta = -H_0 \partial_x u \tag{32}$$
$$\partial_t u = -g\partial_x \eta. \tag{33}$$

Again we will see how to solve them numerically using finite difference schemes. As both equations depend on each other, it is quite natural to solve them using forward-backward time-stepping (here, forward for u and backward for η). Spatially, we consider two different situations

8.20 Discretization Schemes II

- A-Grid (unstaggered), centered in space:

$$\frac{\eta_j^n - \eta_j^{n-1}}{\Delta t} = -H_0 \frac{u_{j+1}^n - u_{j-1}^n}{\Delta x} \tag{34}$$

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = -g \frac{\eta_{j+1}^n - \eta_{j-1}^n}{\Delta x} \tag{35}$$

- C-Grid (staggered), centered in space:

$$\frac{\eta_j^n - \eta_j^{n-1}}{\Delta t} = -H_0 \frac{u_{j+\frac{1}{2}}^n - u_{j-\frac{1}{2}}^n}{\Delta x} \tag{36}$$

$$\frac{u_{j+\frac{1}{2}}^{n+1} - u_{j+\frac{1}{2}}^n}{\Delta t} = -g \frac{\eta_{j+1}^n - \eta_{j-1}^n}{\Delta x} \tag{37}$$

Here, we introduce two spatial grids shifted by $1/2$ relative to each other: one for η (integer), one for u (integer + $1/2$).

8.20 Discretization Schemes III

In both situations, we define the same Courant number $c = \sqrt{gH_0}\frac{\Delta t}{\Delta x}$ and doing a von Neumann stability analysis reveals that the scheme

- for the unstaggered grid is stable for $c \leq 2$
- while for the staggered grid stability is only given for $c \leq 1$.