2.2. Methods for Obtaining FD Expressions

There are several methods, and we will look at a few:

1) Taylor series expansion – the most common, but purely mathematical.

2) Polynomial fitting or interpolation – the most general ways. Taylor series method is a subset of this method. Interpolation takes us back to the M.O.C. and thus has a more physical interpretation.

3) Control volume approach – also called finite volume (FV) – we solve the equations in integral rather than differential form. Popular in engineering where complex geometries and coordinate transformations are involved. For Cartesian grids, simplest FV methods → FD.

We will look at only the first two approaches.

2.2.1. Taylor Series Expansion Method

Recall the definition of a derivative:

$$\left.\frac{\partial u}{\partial x}\right|_{x_0, y_0} = \lim_{\Delta x \to 0} \frac{u(x_0 + \Delta x, y_0) - u(x_0, y_0)}{\Delta x}$$

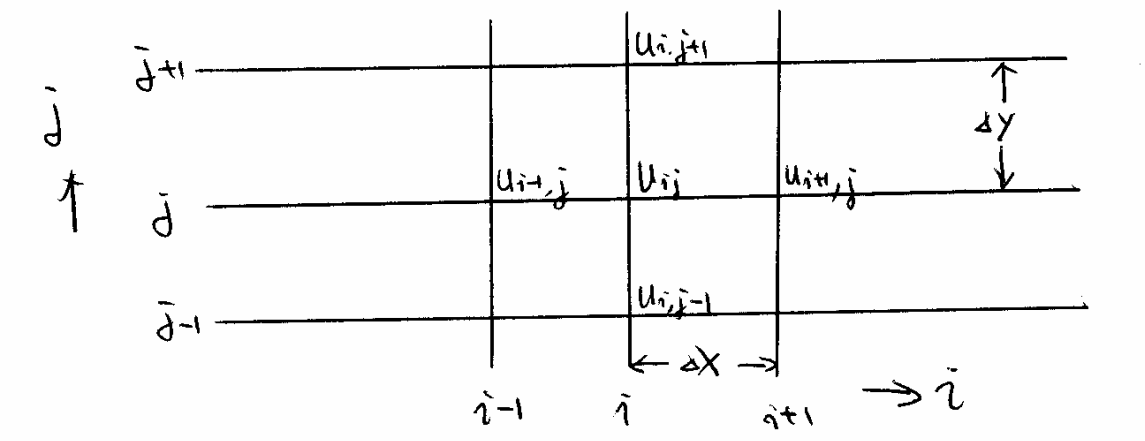
The Taylor series approach works backwards – want to approximate $\partial u/\partial x$ by a discrete difference, i.e., for finite Δx.

Given $u(x_0, y_0)$, we can write a Taylor series expansion for $u(x_0+\Delta x, y_0)$, as

$$u(x_0+\Delta x)=u(x_0)+\Delta x\left.\frac{\partial u}{\partial x}\right|_{x_0,y_0}+\frac{(\Delta x)^2}{2!}\left.\frac{\partial^2 u}{\partial x^2}\right|_{x_0,y_0}+\frac{(\Delta x)^3}{3!}\left.\frac{\partial^3 u}{\partial x^3}\right|_{x_0,y_0}+\ldots=\sum_{n=0}^{\infty}\frac{(\Delta x)^n}{n!}\frac{\partial^n u}{\partial x^n}$$

This expression is <u>exact</u> if we retain all terms!

The grid mesh or stencil looks like:

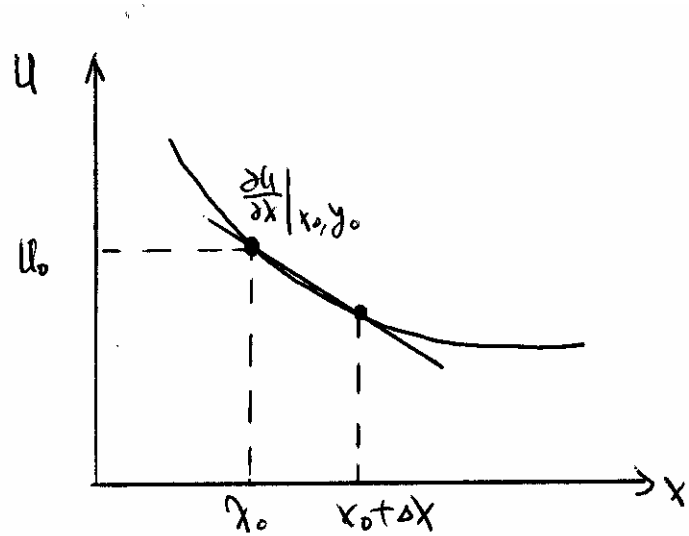


Here, $u_{i+1,j} = u(x_0+\Delta x, y_0)$
$u_{i,j+1} = u(x_0, y_0+\Delta y)$
etc.

If we solve for $\partial u/\partial x$ from the Taylor series, we have

$$\left.\frac{\partial u}{\partial x}\right|_{x_0,y_0}=\frac{u(x_0+\Delta x,y_0)-u(x_0,y_0)}{\Delta x}-\frac{\Delta x}{2!}\left.\frac{\partial^2 u}{\partial x^2}\right|_{x_0,y_0}-\frac{(\Delta x)^2}{3!}\left.\frac{\partial^3 u}{\partial x^3}\right|_{x_0,y_0}+\ldots \quad (1)$$

The first term on the RHS is simply the slope of the function $u(x,y)$, using the current and the points to the right.



Therefore, $\left.\dfrac{\partial u}{\partial x}\right|_{x_0,y_0} = \dfrac{u(x_0+\Delta x, y_0) - u(x_0, y_0)}{\Delta x} + \mathrm{O}(\Delta x)$.

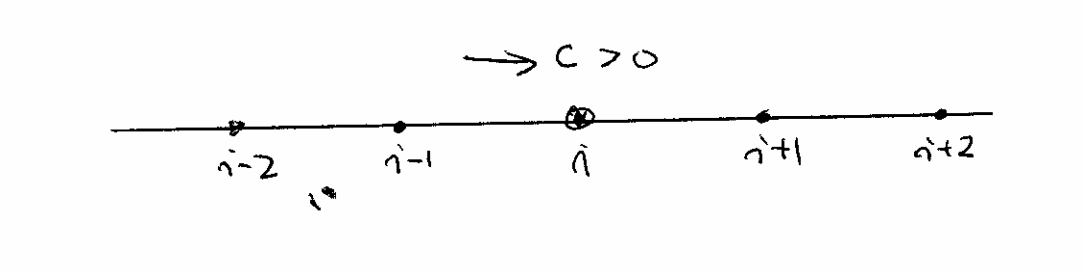
One can also use the value at $x_0 - \Delta x$ instead to get

$$\left.\frac{\partial u}{\partial x}\right|_{x_0,y_0} = \frac{u(x_0, y_0) - u(x_0 - \Delta x, y_0)}{\Delta x} + \frac{\Delta x}{2!}\left.\frac{\partial^2 u}{\partial x^2}\right|_{x_0,y_0} - \frac{(\Delta x)^2}{3!}\left.\frac{\partial^3 u}{\partial x^3}\right|_{x_0,y_0} + \ldots \quad (2)$$

Both (1) and (2) provide an expression for $\partial u/\partial x$, but <u>numerically</u> the answers will be different.

(1) is called a <u>forward difference</u>
(2) is called a <u>backward difference</u>

Consider a 1-D example:



If we have the equation

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0, \quad \text{where } c > 0,$$

we might want to use

$$\frac{\partial u}{\partial t} + c\frac{u_i - u_{i-1}}{\Delta x} \approx 0,$$

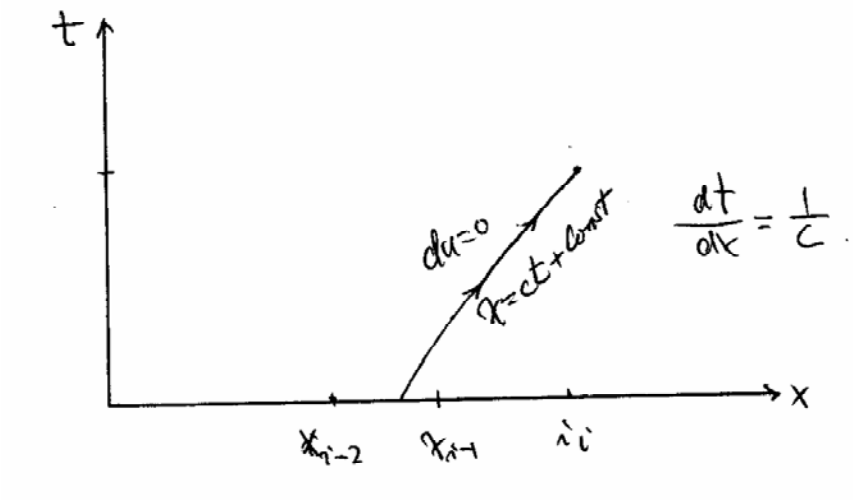
which is called <u>Upwind or Upstream Difference</u>. Alternatively, in

$$\frac{\partial u}{\partial t} + c\frac{u_{i+1} - u_i}{\Delta x} \approx 0,$$

<u>Downstream Difference</u> is used.

Upstream difference is better than downstream difference for this problem, because for this pure advection problem, signals move from upstream (left) to downstream (right). The value of u at point i at a future time should be influenced by the values of u upstream, not downstream.

Think of it in terms of characteristics:



No information is coming from the $+x$ direction. This of course depends on the sign of c. If $c < 0$, then upstream means the left side of the current point.

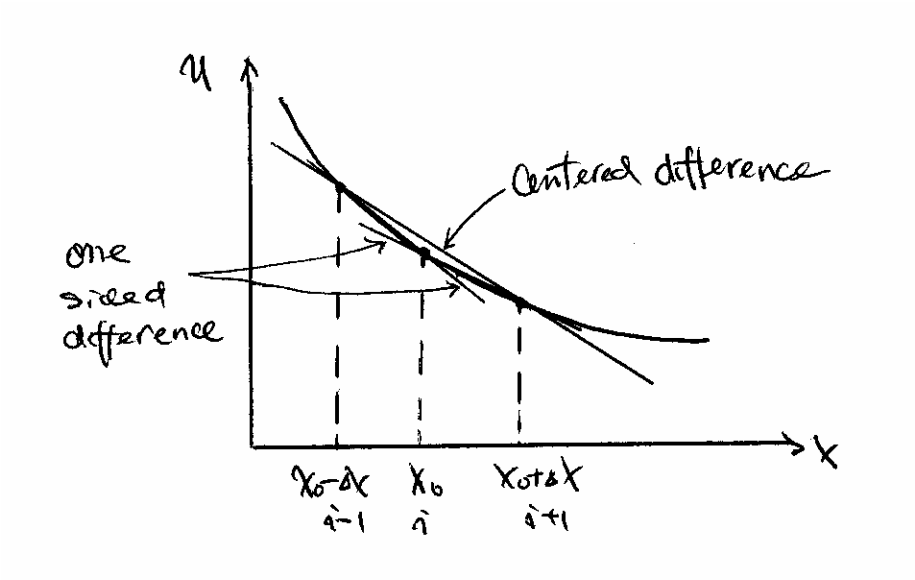
Note that upstream or <u>upstream-biased</u> schemes are usually better choices in CFD (obviously we are talking about hyperbolic equations that represent propagations!).

We will see later, that 1st-order downstream difference is absolutely <u>unstable</u> for the above problem.

We can get another discrete approximation to $\partial u/\partial x$ by adding (1) and (2):

$$\left.\frac{\partial u}{\partial x}\right|_{x_0,y_0} = \frac{u_{i+1}-u_{i-1}}{2\Delta x} - \frac{(\Delta x)^2}{3!}\left.\frac{\partial^3 u}{\partial x^3}\right|_{x_0,y_0} + \ldots \qquad (3)$$

This is called <u>Centered Difference</u>. Note, it doesn't even use value of *u* at the current point *i*. It approximates the <u>slope</u> using two neighboring points:



Note that this will not be accurate if *u* has <u>very sharp gradient</u> (e.g., has a shape of $\wedge$). Note also that the extra terms in (1), (2) and (3) are different – they control the order of accuracy of the scheme.

We can build many different approximations to the derivatives through <u>linear combinations</u> of various T.S. expansions.

For <u>high-order derivatives</u>, we follow the same approach.

Consider $\partial^2 u/\partial x^2$. The Taylor series gives

$$u(x+\Delta x) = u(x) + \Delta x \frac{\partial u}{\partial x} + \frac{(\Delta x)^2}{2!}\frac{\partial^2 u}{\partial x^2} + \frac{(\Delta x)^3}{3!}\frac{\partial^3 u}{\partial x^3} + \ldots \qquad (4)$$

We can solve for $\frac{\partial^2 u}{\partial x^2}$ from (4), but we don't want to have the unknown $\frac{\partial u}{\partial x}$.
We can replace it with one of the earlier approximations to it, or make use of the following:

$$u(x-\Delta x) = u(x) - \Delta x \frac{\partial u}{\partial x} + \frac{(\Delta x)^2}{2!}\frac{\partial^2 u}{\partial x^2} - \frac{(\Delta x)^3}{3!}\frac{\partial^3 u}{\partial x^3} + \ldots \qquad (5)$$

and add (4) and (5) and solve for $\frac{\partial^2 u}{\partial x^2}$:

$$\frac{\partial^2 u}{\partial x^2} = \frac{u(x+\Delta x) - 2u(x) + u(x-\Delta x)}{(\Delta x)^2} + \mathrm{O}(\Delta x^2)$$

which is a centered difference for $\frac{\partial^2 u}{\partial x^2}$.

Truncation Error

The high-order terms (H.O.T.) of $O(\Delta x^n)$ are called the Truncation Error, and are a measure of the error associated with representing a PDE by a truncated T.S. – we can't retain all terms, so we neglect the terms of order $O(\Delta x^2)$ and above, for example.

$$\text{PDE} = \text{FDE} + \tau$$

where τ is the Truncation Error.

If we change the nature of the H.O.T., by using a different approximation, the accuracy of the F.D. expression will also change.

It is important to understand the impact of τ, and this is the subject of one of our computer problems.

From the above, it's clear that we want $\tau \rightarrow 0$ when PDE = FDE. Otherwise, we have a problem (consistency)!

Let $\Delta x \rightarrow 0$, then τ should $\rightarrow 0$ => our discrete system approaches continuum and our FDE $\rightarrow$ PDE.

Order of the F.D. Approximation.

The power to which the leading discrete interval in τ is raised is called the Order of the F.D. Approximation.

Example: $\dfrac{u_i - u_{i-1}}{\Delta x}$ is first-order accurate in space

because $\tau = \frac{\Delta x}{2!}\frac{\partial^2 u}{\partial x^2} + \ldots = O(\Delta x^1)$

$\frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2}$ is second order accurate.

Comments on τ:

(1) If there are several independent variables, each has a truncation error, e.g., $O(\Delta x^2 + \Delta t)$, we say it's first order in time and second order in space.

(2) The <u>accuracy</u> of a scheme also depends on the <u>local</u> properties of the function, it may be much less than the <u>formal</u> or <u>theoretical</u> order near sharp gradients. Recall that Taylor series are valid only for <u>smooth and continuous functions</u>.

(3) Typically, we prefer <u>higher-order scheme</u> because τ is smaller for small Δ. However, the <u>order</u> is not <u>necessarily</u> related to <u>accuracy</u> because, <u>for a given Δx</u>, a first-order scheme may give more accurate results because the coefficient is smaller. What the order does tells us is how the error will <u>change</u> as we change the resolution. Higher-order τ decreases faster when we decreases Δx.

e.g., $\tau = \kappa (\Delta x)^3$ versus $\tau = \kappa \Delta x$.

Remember that τ includes κ, which might be $\frac{\partial^4 u}{\partial x^4}$, etc and it matters.

(4) Truncation error is cumulative – adds up per time step.

(5) Truncation error is usually much larger than <u>machine round-off error</u>. Also for most problems, τ (space) >> τ (time), because solution often evolves smoothly in time but have rapid changes in space. Time step size is often not as large as we wish because of stability constraint. For meteorological models, we usually want to increase time step size and decrease grid spacing.

General Method for Deriving FD Expressions

Let's write a generalized form of the Taylor series as

$$u_{i+1} = \sum_{m=0}^{\infty} \left(\frac{\partial^m u}{\partial x^m} \right)_i \frac{(\Delta x)^m}{m!}$$

Now, suppose we want to use a 3-point stencil at i-1, i, i+1. Then, we can write a generic expression (PDE = FDE + τ) as:

$$\frac{\partial u}{\partial x} = a u_{i-1} + b u_i + c u_{i+1} + \mathrm{O}(\Delta x)^m$$

where a, b and c are unknown constants to be determined and m is the order of the approximation. (General rule: If F.D. spans n points, you can derive an n-1 order F.D. scheme).

Using our Taylor series for u_{i-1}, u_i and u_{i+1}, we can write

$$
\begin{aligned}
&au_{i-1} + bu_i + cu_{i+1} = \\
&\qquad a(u_i - \Delta x \frac{\partial u}{\partial x} + \frac{(\Delta x)^2}{2!}\frac{\partial^2 u}{\partial x^2} - \frac{(\Delta x)^3}{3!}\frac{\partial^3 u}{\partial x^3} + \ldots) \\
&\qquad + bu_i \\
&\qquad c(u_i + \Delta x \frac{\partial u}{\partial x} + \frac{(\Delta x)^2}{2!}\frac{\partial^2 u}{\partial x^2} + \frac{(\Delta x)^3}{3!}\frac{\partial^3 u}{\partial x^3} + \ldots) \\
&= (a+b+c)u_i + (-a+c)\Delta x \frac{\partial u}{\partial x} + (a+c)\frac{(\Delta x)^2}{2!}\frac{\partial^2 u}{\partial x^2} + (-a+c)\frac{(\Delta x)^3}{3!}\frac{\partial^3 u}{\partial x^3} + \ldots (6)
\end{aligned}
$$

Since we want (6) to have the form of $\frac{\partial u}{\partial x} + O(\Delta x)^m$, therefore we set

$(a + b + c) = 0$,
$(-a + c)\Delta x = 1$
$a + c = 0$.

From them we can find $b = 0$, $c = -a = 1/(2\Delta x)$, therefore

$$\frac{\partial u}{\partial x} = \frac{u_{i+1} - u_{i-1}}{2\Delta x} + O(\Delta x)^2$$

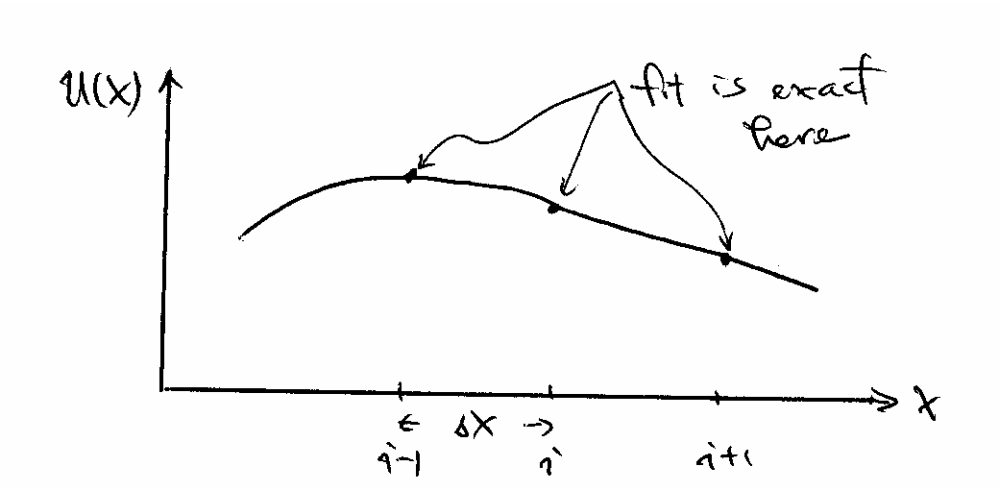
which is a second-order <u>centered difference scheme</u>.

This method can be used in a general manner for symmetric or non-symmetric difference formula. You just pick which points you want to use.

2.2.2. Polynomial Fitting

This is the second, most general method for generating finite difference expression. Here, we assume that the solution to the PDE can be approximated by a polynomial, and that the values at the mesh points are exact. We thus differentiate the polynomial to obtain expression for various derivatives.

Assume that $u(x) = a\,x^2 + b\,x + c$:



Goal: Find a, b and c. Note that the grid spacing need not be uniform.

Applying the polynomial to those three points gives

$$u_{i-1} = a\,x^2_{i-1} + b\,x_{i-1} + c$$

$$u_i = a\,x^2_i + b\,x_i + c$$

$$u_{i+1} = a\,x^2_{i+1} + b\,x_{i+1} + c$$

Solve for *a*, *b*, *c*, we obtain

$$u(x) = u_{i-1}\left[\frac{(x-x_i)(x-x_{i+1})}{(x_{i-1}-x_i)(x_{i-1}-x_{i+1})}\right] + u_i\left[\frac{(x-x_{i-1})(x-x_{i+1})}{(x_i-x_{i-1})(x_i-x_{i+1})}\right]$$
$$+ u_{i+1}\left[\frac{(x-x_i)(x-x_{i-1})}{(x_{i+1}-x_i)(x_{i+1}-x_{i-1})}\right]$$

Note: $u(x_i) = u_i$ for i, i+1, i-1 (verify yourself).

The above formula is often called a <u>Lagrange Interpolation Polynomial</u> and can be generalized to any order.

If the <u>true</u> solution u is a polynomial having the same degree as that used in the interpolation, then the F.D. expression will be <u>exact</u> … as will be the derivatives.

Now, let's compute the derivative $\left.\frac{\partial u}{\partial x}\right|_{x_i}$ by differentiating the expression for $u(x)$.

First, let us <u>define</u> $\Delta x \equiv x_{i+1} - x_i = x_i - x_{i-1}$ => $2\,\Delta x = x_{i+1} - x_{i-1}$, therefore

$$\frac{\partial u}{\partial x} = \frac{u_{i+1} - u_{i-1}}{2\Delta x}$$

$$\frac{\partial^2 u}{\partial x^2} = \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2}$$

$\frac{\partial^3 u}{\partial x^3} = 0$ because we used a 2nd order polynomial. To obtain FD formulation for third order derivative, we have to use 3rd- or higher- order polynomial.

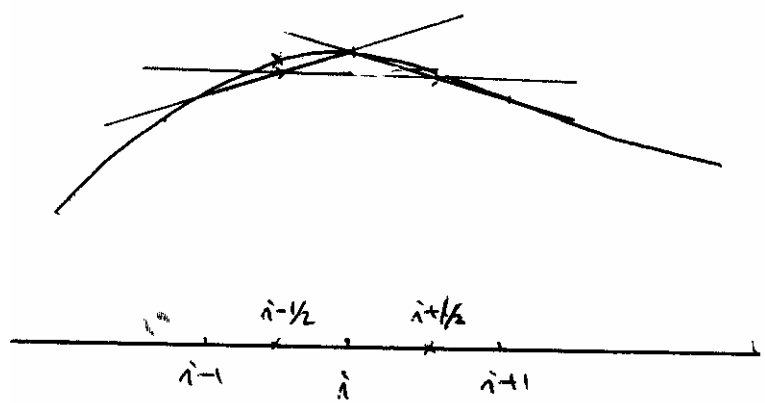
The 2nd order case can be interpreted physically as the derivative of derivative, or the difference between two slopes

$$\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}\right) = \frac{\partial^2 u}{\partial x^2}$$

or

$$\frac{\frac{u_{i+1} - u_i}{\Delta x} - \frac{u_i - u_{i-1}}{\Delta x}}{\Delta x}$$

i.e., the difference between the slope centered at i+1/2 and slope centered at i-1/2.



- It turns out the a 2nd-order polynomial is usually adequate

- Higher-order approximations tend to introduce <u>noises</u> into the solution because the higher-order derivatives may be large, particularly near sharp gradients.

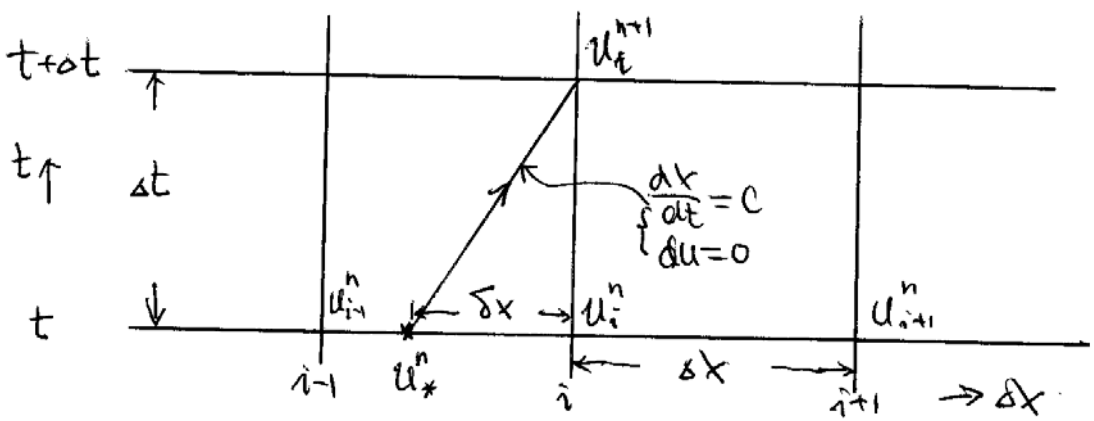
- Beyond 2nd-order, the polynomial method is <u>not</u> identical to the Taylor series method.

2.2.3. Interpolation and Characteristics

Interpolation is at the heart of virtually all numerical techniques. e.g., 1-D advection equation:

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0 \qquad (c > 0 \text{ and constant})$$

Let's draw a space-time diagram, using discrete points in space and time:



Recall that dx/dt = c and du=0 along the characteristic curves.

Goal: Determine $u_i^{n+1} = f(u_i^n, u_{i-1}^n, u_{i+1}^n, etc)$.

We recognize u_i^{n+1} = value of u along characteristic curve at the previous time level, i.e., u_*^n, i.e.,

$$u_i^{n+1} = u_*^n .$$

In general, u^* won't coincide with grid points but we know only values at the grid points. So we need <u>interpolation</u> from the grid points to *.

We are free to use as many points as we want, more points → higher order.

Indeed, the interpolation used determines the <u>order of accuracy</u> of the solution.

With simple linear interpolation, we get (see figure)

$$u^* = \frac{\delta x}{\Delta x} u_{i-1}^n + \left(1 - \frac{\delta x}{\Delta x}\right) u_i^n .$$

Now, in one time step Δt, a particle moves a distance δx at speed c, i.e.,

$$\delta x = c\, \Delta t.$$

Substituting, we have

$$u_*^n = u_i^{n+1} = \frac{c\Delta t}{\Delta x} u_{i-1}^n + u_i^n - \frac{c\Delta t}{\Delta x} u_i^n$$

or

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + c\frac{u_i^n - u_{i-1}^n}{\Delta x} = 0$$

This is a familiar first-order upstream method using a forward-in-time scheme!

As we will see later, this scheme is <u>strongly damping</u>, therefore inaccurate. This is because linear interpolation always under-estimate the peak values in the solution.

This scheme yields exact solution if c = constant and $\Delta t = \Delta x/c$. For general problems this condition is hard to meet, however.

If we new include point i+1 in our interpolation, we end up fitting a parabola to the points and obtain

$$u_*^n = u_i^{n+1} = u_i^n - \frac{\alpha}{2}(u_{i+1}^n - u_{i-1}^n) + \frac{\alpha^2}{2}(u_{i+1}^n - 2u_i^n + u_{i-1}^n)$$

where $\alpha = \dfrac{c\Delta t}{\Delta x}$.

This scheme is known as the Lax-Wendroff or the Crowley scheme in 1-D (see Tremback et al 1987 for example of using high-order interpolation to obtain high-order Crowley schemes).

Notice that the last term on R.H.S. is an approximation to a second-order derivative which represents a diffusion process. This term is not present in the original advection equation. It is needed, however, to keep the scheme stable (by damping unstable modes). Without this term, the scheme becomes forward-in-time and centered-in-space. We will see later in the section of stability analysis that such as scheme is absolutely unstable (i.e., the numerical solution will grow without bound – in computer problems, you will quickly run into floating-point overflow error).

Discussion on Stability Condition / Constraint on Time Step Size

It turns out, as we will show later, that we can only use interpolation not extrapolation at time n for numerical stability.

The criterion of interpolation is that

$$\left|\frac{c\Delta t}{\Delta x}\right| < 1, \text{ i.e.,}$$

$$|\delta x| < |\Delta x|$$

or particle cannot move more than one grid interval per time step.

This is known as the stability constraint, and we can write it as

$$\Delta t < \Delta x / c.$$

It is a limit on the time step size in terms of Δx and c. All explicit time integration schemes have stability limitations (all of which are not the same). The faster signal moves, the smaller is the time step size that can be used. Half $\Delta x \rightarrow$ half Δt. In 3D and for a fixed domain size, doubling resolution in all three dimensions increases the total computation by a factor $(2)^4 = 16$!