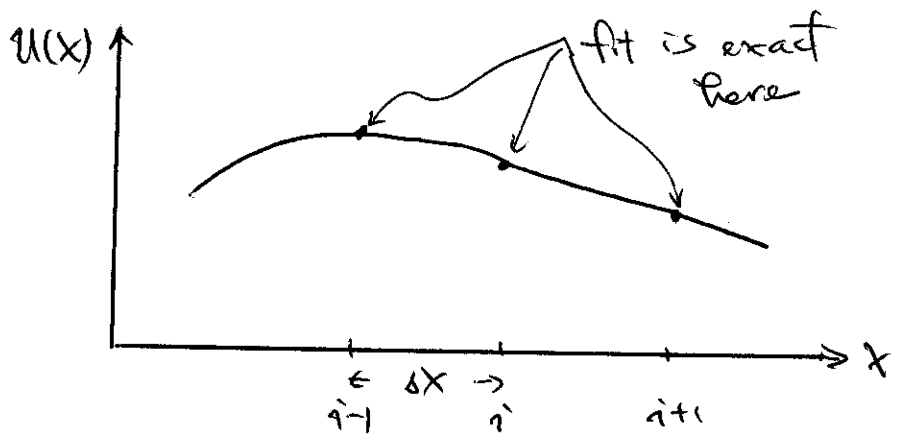
2.2. Polynomial Fitting

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 at 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 Lagrange Interpolation Polynomial and can be generalized to any order.

If the true solution u is a polynomial having the same degree as that used in the interpolation, then the F.D. expression will be exact … 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 define $\Delta x \equiv x_{i+1} - x_i = x_i - x_{i-1}$ => $2\Delta x = x_{i+1} - x_{i-1}$, 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.

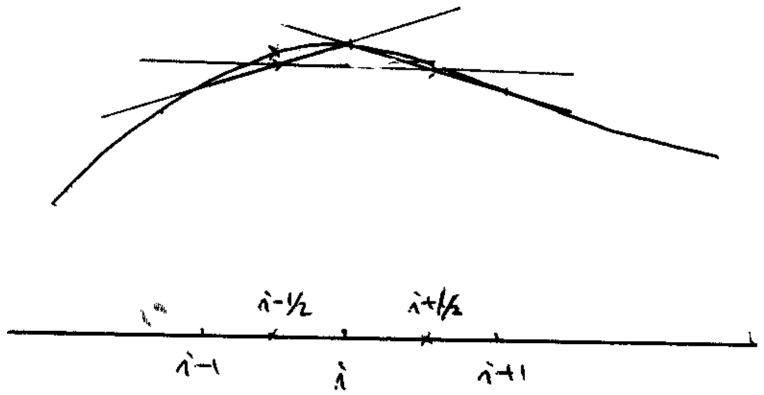
The 2nd order case can be interpreted physically as the derivative of the derivative, or the difference between two slopes

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

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 approximation tend to introduce noises into the solution because the higher-order derivatives may be large, particularly near sharp gradients.

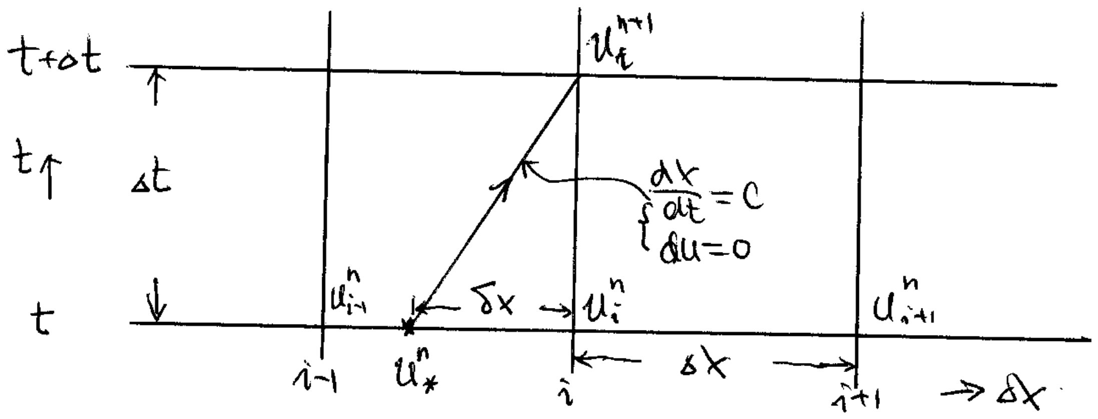
- Beyond 2nd-order, the polynomial method is not identical to the Taylor series method.

2.3. Interpolation and Characteristics

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

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

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 interpolation 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 order of accuracy of the solution.

With simple linear interpolation, we get (see figure)

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

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 problem 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

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

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

This scheme is known as the Lax-Wendroff or the Crowley scheme in 1-D.

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 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 $(2)^4 = 16$!