Finite volume methods

Stephen Millmore, Louisa Michael and Nikos Nikforakis

Laboratory for Scientific Computing, University of Cambridge

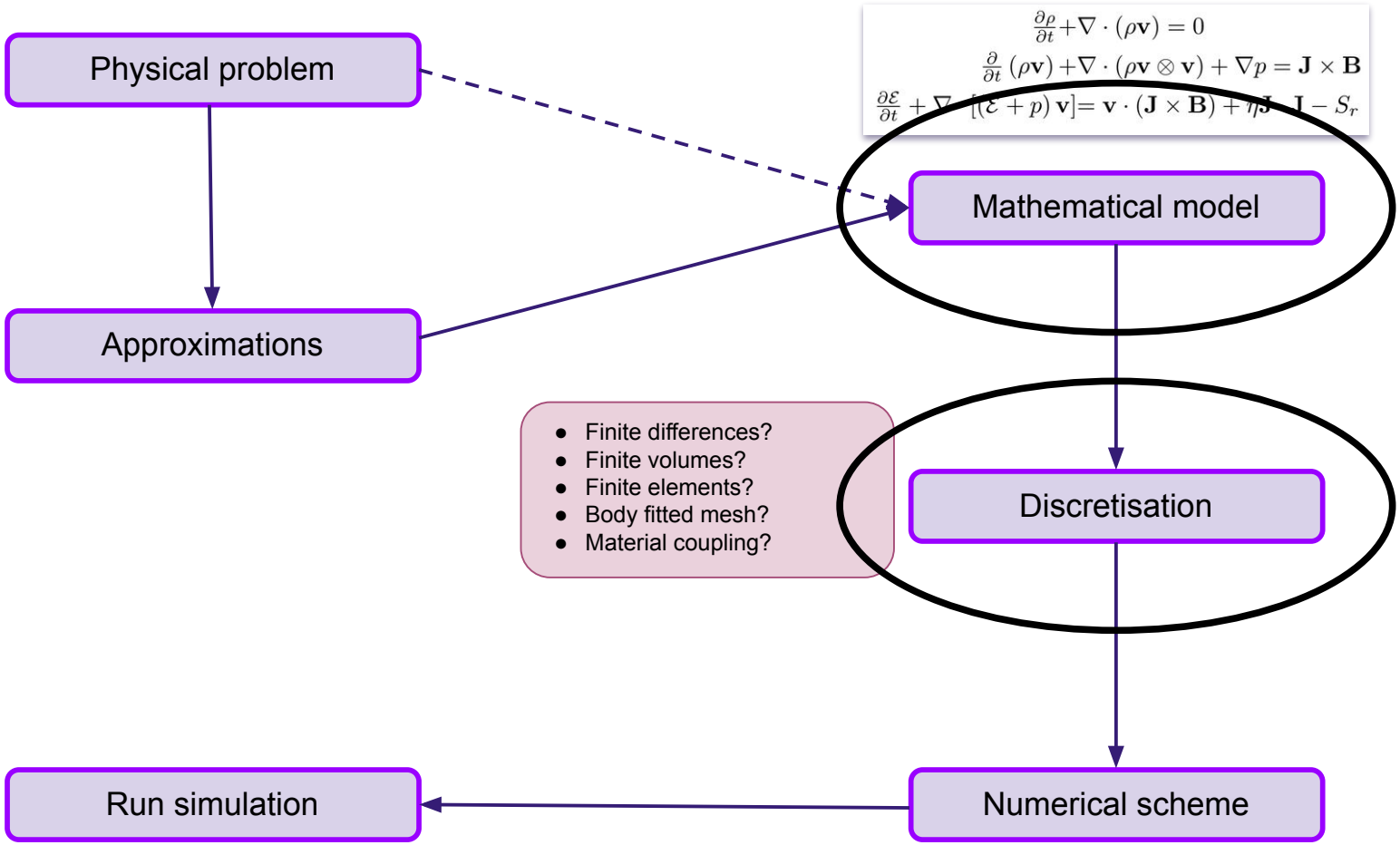
Outline

1. Finite volume methods
2. Centred schemes
3. Riemann problem-based schemes
4. Writing finite volume codes

Outline

1. Finite volume methods
2. Centred schemes
3. Riemann problem-based schemes
4. Writing finite volume codes

Finite volume methods



Why is conservation form important?

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \mathbf{f}(\mathbf{u}) = 0$$

- Over the first few lectures, we have often stated that conservation form is important
- For the advection equation, we didn't actually worry too much about writing our equation in this form (a direct result of $a = \text{const}$)
- However, for Burgers' equation, solving the conservation form is essential
- Recall, we the equation can be written in two different ways:

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\left(\frac{1}{2}u^2\right) = 0 \qquad \text{or} \qquad \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0$$

- Although these are mathematically the same equation, the discretised versions do not look the same

Why is conservation form important?

- Consider a first-order backwards difference

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\left(\frac{1}{2}u^2\right) = 0 \qquad \rightarrow \qquad u_i^{n+1} = u_i^n - \frac{\Delta t}{\Delta x}\left[\frac{1}{2}\left(u_i^n\right)^2 - \frac{1}{2}\left(u_{i-1}^n\right)^2\right]$$

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0 \qquad \rightarrow \qquad u_i^{n+1} = u_i^n - u_i^n\frac{\Delta t}{\Delta x}\left(u_i^n - u_{i-1}^n\right)$$

- Would you expect these to give the same results?
- Recall when we introduced conservation laws, we came up with an integral form that stated that the change in a variable, $\mathbf{u}$, could be determined by the amount of this variable that leaves or enters a domain

$$\int_a^b \mathbf{u}(t_2, x) - \mathbf{u}(t_1, x)\mathrm{d}x = -\int_{t_1}^{t_2} \mathbf{f}(\mathbf{u}(t, b)) - \mathbf{f}(\mathbf{u}(t, a))\mathrm{d}t$$

- The same behaviour can be seen in the discretised conservation law

Conservation and discrete representations

- We consider a domain discretised into $i \in [0, M]$ points

$$\sum_{i=0}^{M} u_i^{n+1} = \sum_{i=0}^{M} \left(u_i^n - \frac{\Delta t}{\Delta x} \left[\frac{1}{2} \left(u_i^n \right)^2 - \frac{1}{2} \left(u_{i-1}^n \right)^2 \right] \right) = \sum_{i=0}^{M} u_i^n - \frac{\Delta t}{\Delta x} \left[\frac{1}{2} \left(u_M^n \right)^2 - \frac{1}{2} \left(u_{-1}^n \right)^2 \right]$$

- The only change to the total amount of u in the domain is the contributions from the left-most and right-most terms
- The same is not true for the non-conservative equation

$$\sum_{i=0}^{M} u_i^{n+1} = \sum_{i=0}^{M} u_i^n - \frac{\Delta t}{\Delta x} \left[\sum_{i=0}^{M} \left(\left(u_i^n \right)^2 - u_i^n u_{i-1}^n \right) \right]$$

- In the limit $\Delta x \to 0$, we would hope that $u_i^n \to u_{i-1}^n$, and hence that the central terms would cancel, but they do not cancel in the discrete form
- What happens if there is a discontinuity?

Discontinuities and shock capturing

$$u_{i+1}^{n+1} + u_i^{n+1} = u_i^n + u_{i+1}^n - \frac{\Delta t}{\Delta x}\left[(u_{i+1}^n)^2 - u_{i+1}^n u_i^n + (u_i^n)^2 - u_i^n u_{i-1}^n\right]$$

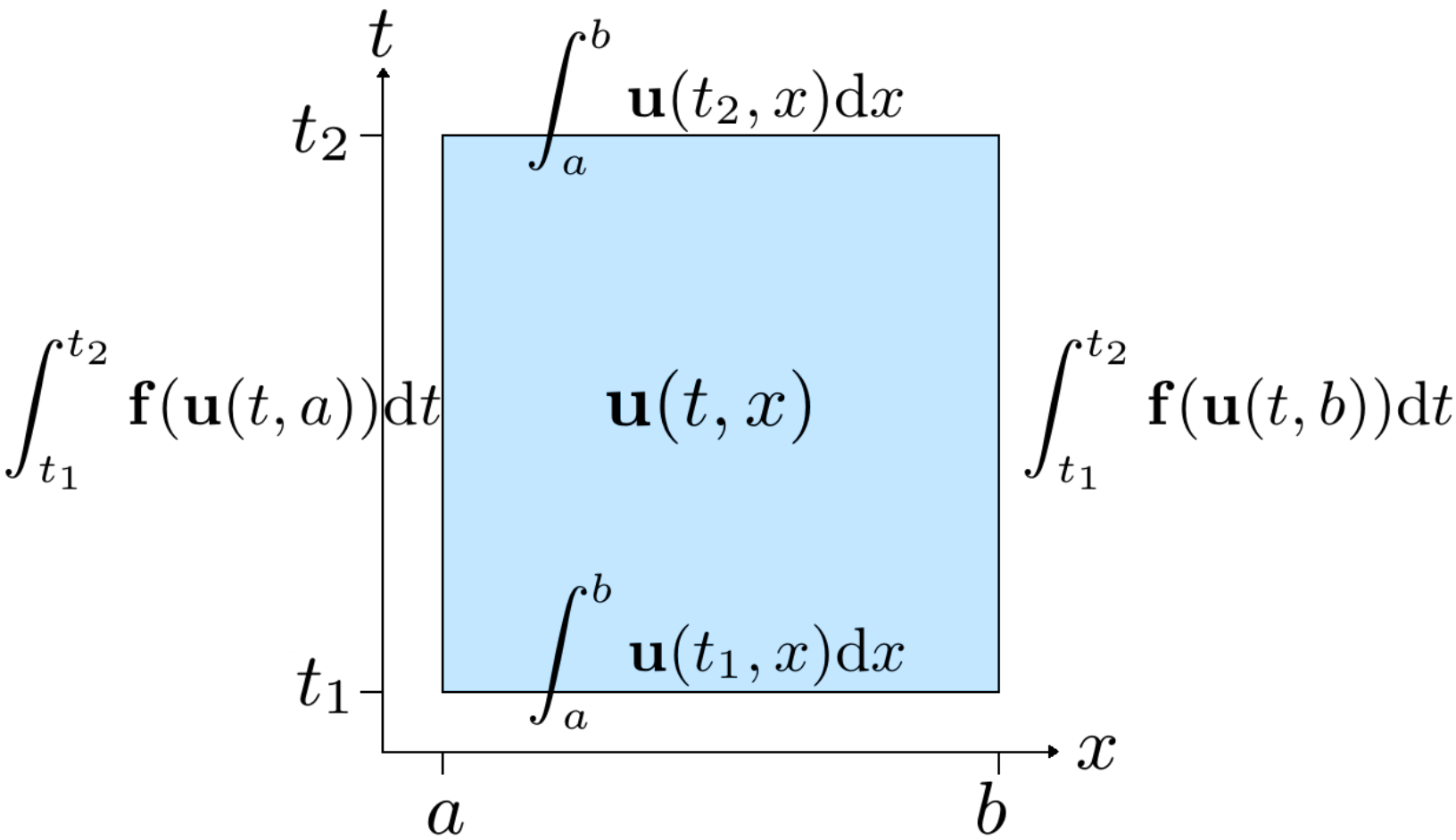
- Considering just two terms from the non-conservative summation, if there is a discontinuity, then these terms will not cancel, even in the limit of $\Delta x \to 0$
- If we consider a domain which is uniform, other than the discontinuity, then the gain (or loss) of u by using the non-conservative update would necessarily result in the discontinuity being incorrectly placed
- This can be shown more rigorously (see the textbooks of Laney and Leveque) - **non-conservative** methods **will** get the location of a shock wave **wrong**
- Similarly, the 'balance' of the terms in the **conservative** equation is achieved by getting the location of a discontinuity **correct**
- This property is known as **shock capturing**
- Note - there is no guarantee a shock capturing method will actually be stable though

Finite volume methods

- When considering conservation equations and related numerical methods, there has been a subtle change in how we consider the discretised equations
- When we talk about a change over a domain, with approximations to integral terms, we are considering the domain as a **volume** rather than a set of points
- In other words, we are no longer considering a finite difference discretisation, but a **finite volume** one
- There are a few changes in terminology that come with a move to finite volume methods:
  - **Cells** - instead of points, a discretised unit is a volume with a fixed width
  - **Fluxes** - instead of differences, within a given volume, material that leaves (or enters) the cell can be considered a flux through the "walls" of this cell

The four integral quantities

- Recall from our initial discussion of conservation laws, we considered four integrals



Finite volume method notation

- Within a finite volume method, discretised notation already introduced does not change
- A cell is still defined at a location x_i - this location is the mid-point of the cell, $x \in \left[x_i - \frac{1}{2}\Delta x, x_i + \frac{1}{2}\Delta x\right]$
- A common shorthand for the edges of a cell is $x_{i\pm 1/2} = x_i \pm \frac{1}{2}\Delta x$
- The quantity $\mathbf{u}_i^n$ is now the volume within cell i at time n - the quantity itself is an **integral average**

$$\mathbf{u}_i^n = \frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} \mathbf{u}\left(x, t^n\right) \mathrm{d}x$$

- The flux is the material that passes through the boundaries of the cell is now an integral over a time interval (still Δt)

$$\mathbf{f}_{i\pm 1/2}^n = \frac{1}{\Delta t}\int_{t^n}^{t^{n+1}} \mathbf{f}\left(\mathbf{u}\left(x_{i\pm 1/2}, t\right)\right) \mathrm{d}t$$

Finite volume update

- Using these definitions, the integral quantities at cell boundaries can be replaced with volume averages, e.g.

$$\int_{x_{i-1/2}}^{x_{i+1/2}} \mathbf{u}\left(x, t^n\right) \mathrm{d}x = \Delta x \mathbf{u}_i^n$$

- With a bit of rearranging, conservation gives us a reasonably familiar expression

$$\mathbf{u}_i^{n+1} = \mathbf{u}_i^n - \frac{\Delta t}{\Delta x}\left(\mathbf{f}_{i+1/2}^n - \mathbf{f}_{i-1/2}^n\right)$$

- Although this looks like a discretised form of the underlying equations, because $\mathbf{u}$ and $\mathbf{f}$ are integral averages, it is actually the exact solution for $\mathbf{u}^{n+1}$
- The only challenge - can we compute these integral averages on discrete space?

Notation

$$\mathbf{u}_i^{n+1} = \mathbf{u}_i^n - \frac{\Delta t}{\Delta x}\left(\mathbf{f}_{i+1/2}^n - \mathbf{f}_{i-1/2}^n\right) \qquad u_i^{n+1} = u_i^n - \frac{\Delta t}{\Delta x}\left[\frac{1}{2}\left(u_i^n\right)^2 - \frac{1}{2}\left(u_{i-1}^n\right)^2\right]$$

$\mathbf{u}_i^{n+1}$ This quantity could be **either** a finite volume or a finite difference representation

- Either the text describing the method, or the other terms in the method, will make it clear

$\frac{1}{2}\left(u_i^n\right)^2$ Whether this is finite volume or finite difference depends on how u_n^n is computed

- If u_i^n is a finite volume quantity, this could be a finite volume flux

$\mathbf{f}_{i+1/2}^n$ This **must** be a finite volume quantity - variables cannot be defined at $x_{i+1/2}$ in finite differences

Finite volume update

$$\mathbf{u}_i^{n+1} = \mathbf{u}_i^n - \frac{\Delta t}{\Delta x}\left(\mathbf{f}_{i+1/2}^n - \mathbf{f}_{i-1/2}^n\right)$$

- Can we compute these integral averages on discrete space?
- Related question - do we need to know the distribution of $\mathbf{u}$ over a finite volume cell?
- If you know $\mathbf{u}_i^n$, you don't actually need to worry about how material is distributed, just how much is in your volume
- If you also know $\mathbf{f}_{i\pm1/2}^n$, then you'll know $\mathbf{u}_i^{n+1}$, just not how it is distributed
- The integral averages of the fluxes are the problem - can you know all behaviour between t and $t + \Delta t$ on a discrete space?

Finite volume update

- In fact, the fluxes are where the numerical approximations and discrete differences come in

$$\mathbf{u}_i^{n+1} = \mathbf{u}_i^n - \frac{\Delta t}{\Delta x}\left(\mathbf{f}_{i+1/2}^n - \mathbf{f}_{i-1/2}^n\right)$$

- The challenge of a numerical method is to calculate the flux, $\mathbf{f}_{i\pm1/2}^n$ in a manner that gives stable results
- If we can do this, because of the formulation, we know that we automatically have a conservative method
- Two possibilities for defining this flux:
  1. **Centred schemes**: Define flux in terms of integral averages, comparisons to finite difference schemes are clear here
  2. **Riemann problem-based schemes**: Solve a Riemann problem between two neighbouring cells to compute the flux
- Typically centred schemes are 'easier' but discontinuities are not captured as sharply

Flux notation

- Notation for numerical methods gets confusing at this point
- You have the **numerical flux**, $\mathbf{f}^n_{i\pm1/2}$
- This should **always** have the $i \pm 1/2$ subscript, though sometimes a time superscript is dropped
- This is always an integral average quantity, and, in most cases, approximated by a numerical method
- You also have the **flux function**, $\mathbf{f}$
- This should **never** have any subscript
- This is the flux function defined by your conservation law

Flux notation

- If you see something like $\mathbf{f}\left(\mathbf{u}_i^n\right)$, you take your approximation for $\mathbf{u}_i^n$ and plug it into the flux function
- For Burgers' equation

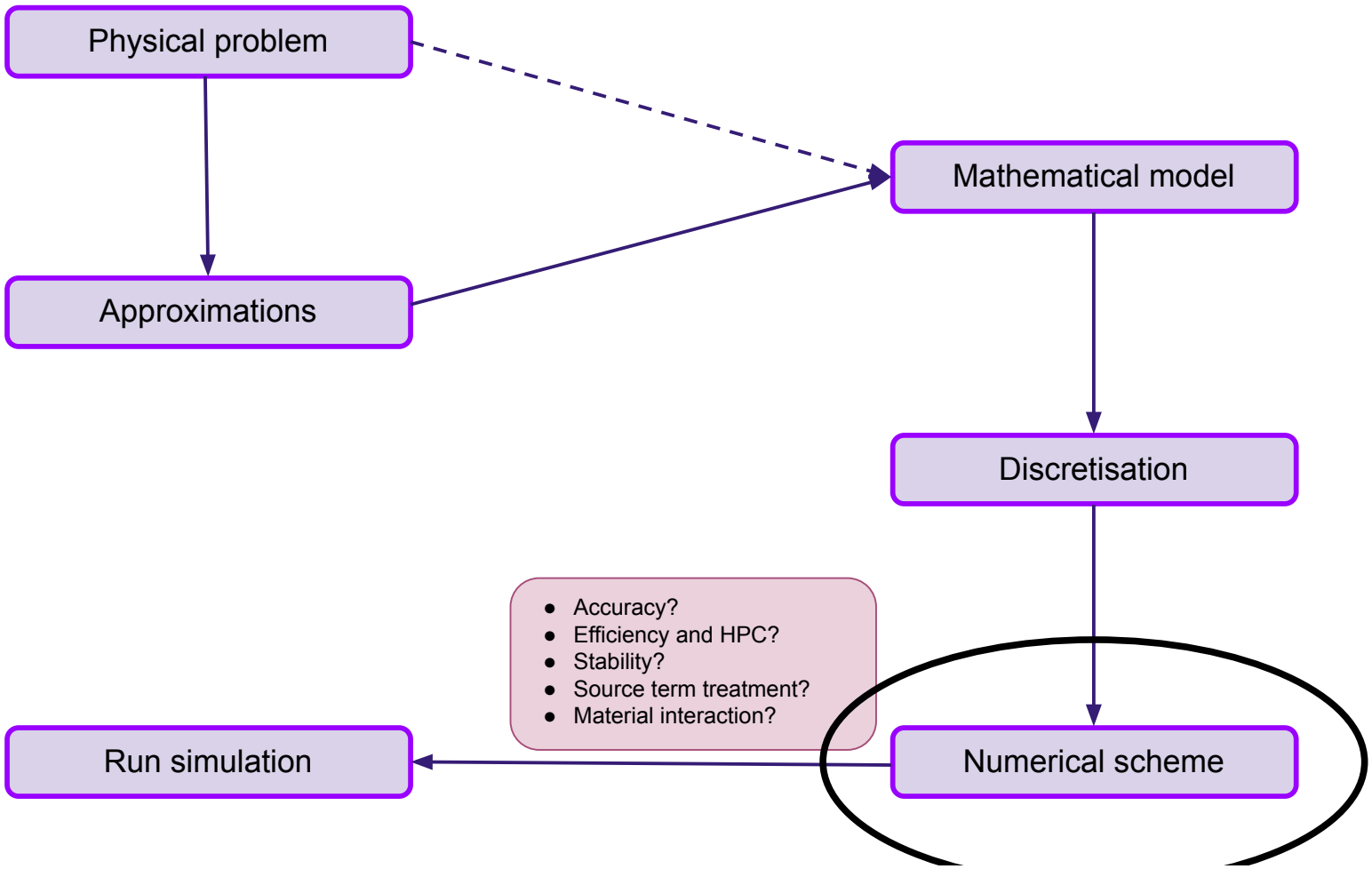
$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \mathbf{f}(\mathbf{u}) = 0, \qquad \frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\left(\frac{1}{2}u^2\right) = 0, \qquad f(u) = \frac{1}{2}u^2$$

- For Burgers' equation, $f\left(u_i^n\right) = \frac{1}{2}\left(u_i^n\right)^2$
- Distinguishing between flux functions and numerical fluxes can make reading numerical methods tricky
- It is almost universal to use $\mathbf{f}$ as the symbol for both - I won't change this

Outline

1. Finite volume methods
2. Centred schemes
3. Riemann problem-based schemes
4. Writing finite volume codes

Centred schemes



Finite difference scheme to finite volume scheme

- All of the finite difference schemes we have seen so far can be converted to finite volume schemes
- Although we expressed our schemes for the advection equation, once written in finite volume (conservation) form, the schemes will be applicable to any conservation equation
- Recall the advection equation can be written

$$\frac{\partial u}{\partial t} + \frac{\partial (au)}{\partial x} = 0$$

- In other words, $f(u) = au$
- This enables us to write finite difference schemes in terms of fluxes, we will start with the first-order upwind scheme (for $a > 0$)

Finite volume first-order upwind scheme

- Recall the first-order upwind scheme for the advection equation

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

- By writing this

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

  we can immediately identify $f_{i+1/2}^n = au_i^n = f(u_i^n)$

- Note, we have already seen how other equations can be written for this scheme when we wrote Burgers' equation as

$$u_i^{n+1} = u_i^n - \frac{\Delta t}{\Delta x}\left[\frac{1}{2}\left(u_i^n\right)^2 - \frac{1}{2}\left(u_{i-1}^n\right)^2\right]$$

- Here, $f_{i+1/2}^n = \frac{1}{2}\left(u_i^n\right)^2 = f(u_i^n)$

Finite volume Lax-Friedrichs scheme

- The Lax-Friedrichs scheme for the advection equation was

$$u_i^{n+1} = \frac{1}{2}\left(1 + a\frac{\Delta t}{\Delta x}\right) u_{i-1}^n + \frac{1}{2}\left(1 - a\frac{\Delta t}{\Delta x}\right) u_{i+1}^n$$

- It is not immediately obvious how to write this in conservative form (we explicitly removed the leading u_i^n term to make this method stable)
- We need to re-write this equation such that it we can identify two 'symmetric' flux terms
- We start by introducing a u_i^n term and rewriting

$$u_i^{n+1} = u_i^n - u_i^n + \frac{1}{2}u_{i-1}^n + \frac{1}{2}u_{i+1}^n - \frac{\Delta t}{2\Delta x}\left(au_{i+1}^n - au_{i-1}^n\right)$$

- We now use $au_i^n = f(u_i^n)$, and rewrite further to give

$$u_i^{n+1} = u_i^n - \frac{1}{2}\left(u_i^n - u_{i+1}^n\right) + \frac{1}{2}\left(u_{i-1}^n - u_i^n\right) - \frac{\Delta t}{\Delta x}\left[\frac{1}{2}\left(f(u_{i+1}^n) - f(u_i^n) + f(u_i^n) - f(u_{i-1}^n)\right)\right]$$

Finite volume Lax-Friedrichs scheme

$$u_i^{n+1} = u_i^n - \frac{1}{2}\left(u_i^n - u_{i+1}^n\right) + \frac{1}{2}\left(u_{i-1}^n - u_i^n\right) - \frac{\Delta t}{\Delta x}\left[\frac{1}{2}\left(f(u_{i+1}^n) + f(u_i^n) - f(u_i^n) - f(u_{i-1}^n)\right)\right]$$

- We are approaching a conservative formulation - the equation has been split into terms depending on (x_i, x_{i+1}) and similar looking terms depending on (x_{i-1}, x_i)

$$u_i^{n+1} = u_i^n - \frac{\Delta t}{\Delta x}\left[\frac{1}{2}\frac{\Delta x}{\Delta t}\left(u_i^n - u_{i+1}^n\right) + \frac{1}{2}\left(f(u_{i+1}^n) + f(u_i^n)\right)\right.$$
$$\left. - \frac{1}{2}\frac{\Delta x}{\Delta t}\left(u_{i-1}^n - u_i^n\right) - \frac{1}{2}\left(f(u_i^n) + f(u_{i-1}^n)\right)\right]$$

- Whilst we have made rather a mess of the previously one-line update scheme, we can now define the **Lax-Friedrichs flux**

$$f_{i+1/2}^n = \frac{1}{2}\frac{\Delta x}{\Delta t}\left(u_i^n - u_{i+1}^n\right) + \frac{1}{2}\left(f(u_{i+1}^n) + f(u_i^n)\right)$$

Another scheme - the Richtmyer flux

- When considering finite volume methods, the fact that data effectively exists everywhere gives new options for flux methods
- For example, we can use volume averages of cells x_i and x_{i+1} to compute $\mathbf{u}^n_{i+1/2}$
- We can even estimate this quantity at times between t^n and t^{n+1}
- Although this data cannot be stored permanently on our mesh, we can use it as part of a numerical method
- One example of this sort of method is the **Richtmyer flux** method
- The Richtmyer flux is defined as

$$\mathbf{f}^n_{i+1/2} = \mathbf{f}\left(\mathbf{u}^{n+1/2}_{i+1/2}\right)$$

- The quantity $\mathbf{u}^{n+1/2}_{i+1/2}$ is sometimes known as the half-time step update, and for the Richtmyer flux, has a single technique to calculate it (other techniques may exist, but may not be stable numerical methods)

Another scheme - the Richtmyer flux

$$\mathbf{f}_{i+1/2}^{n} = \mathbf{f}\left(\mathbf{u}_{i+1/2}^{n+1/2}\right)$$

- The half-time step update is straightforward; a Lax-Friedrichs update

$$\mathbf{u}_{i+1/2}^{n+1/2} = \frac{1}{2}\left(\mathbf{u}_{i}^{n} + \mathbf{u}_{i+1}^{n}\right) - \frac{1}{2}\frac{\Delta t}{\Delta x}\left(\mathbf{f}\left(\mathbf{u}_{i+1}^{n}\right) - \mathbf{f}\left(\mathbf{u}_{i}^{n}\right)\right)$$

- Note the difference between this version of the Lax-Friedrichs update and earlier ones is that it uses cells x_i and x_{i+1}, instead of x_{i-1} and x_{i+1}
- As a result, it is only stable for evolving the solution up to $\Delta t/2$
- However, since there is also a scaling of $\Delta x \to \Delta x/2$, it then looks identical
- It can be shown that using the Richtmyer flux is second-order accurate

What scheme should we use?

- So far we have seen second-order schemes (Richtmyer, Lax-Wendroff and Warming-Beam), all of which suffer from oscillations around discontinuities
- The Richtmyer flux will also cause oscillations
- We have also seen first-order schemes which are stable, but are **very** diffusive
- This is because, within the modified equation

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = A_1 \frac{\partial^2 u}{\partial x^2} + \dots$$

  the A_1 term was large
- Within this course, we shall consider how to obtain second-order accuracy without oscillations - first we want to consider some underlying first-order schemes on which these methods will be built
- For centred schemes, it is possible to reduce the size of A_1, whilst maintaining stability, by introducing a flux which is an average of a first- and second-order flux

First ORder CEntered scheme (FORCE)

$$\mathbf{u}_i^{n+1} = \mathbf{u}_i^n - \frac{\Delta t}{\Delta x}\left(\mathbf{f}_{i+1/2}^{\mathrm{FORCE}} - \mathbf{f}_{i-1/2}^{\mathrm{FORCE}}\right)$$

- The FORCE flux is simply the average of the Lax-Friedrichs and the Richtmyer fluxes

$$\mathbf{f}_{i+1/2}^{\mathrm{FORCE}} = \frac{1}{2}\left(\mathbf{f}_{i+1/2}^{\mathrm{LF}} + \mathbf{f}_{i+1/2}^{\mathrm{RI}}\right)$$

$$\mathbf{f}_{i+1/2}^{\mathrm{LF}} = \frac{1}{2}\frac{\Delta x}{\Delta t}\left(\mathbf{u}_i^n - \mathbf{u}_{i+1}^n\right) + \frac{1}{2}\left(\mathbf{f}(u_{i+1}^n) + \mathbf{f}(u_i^n)\right)$$

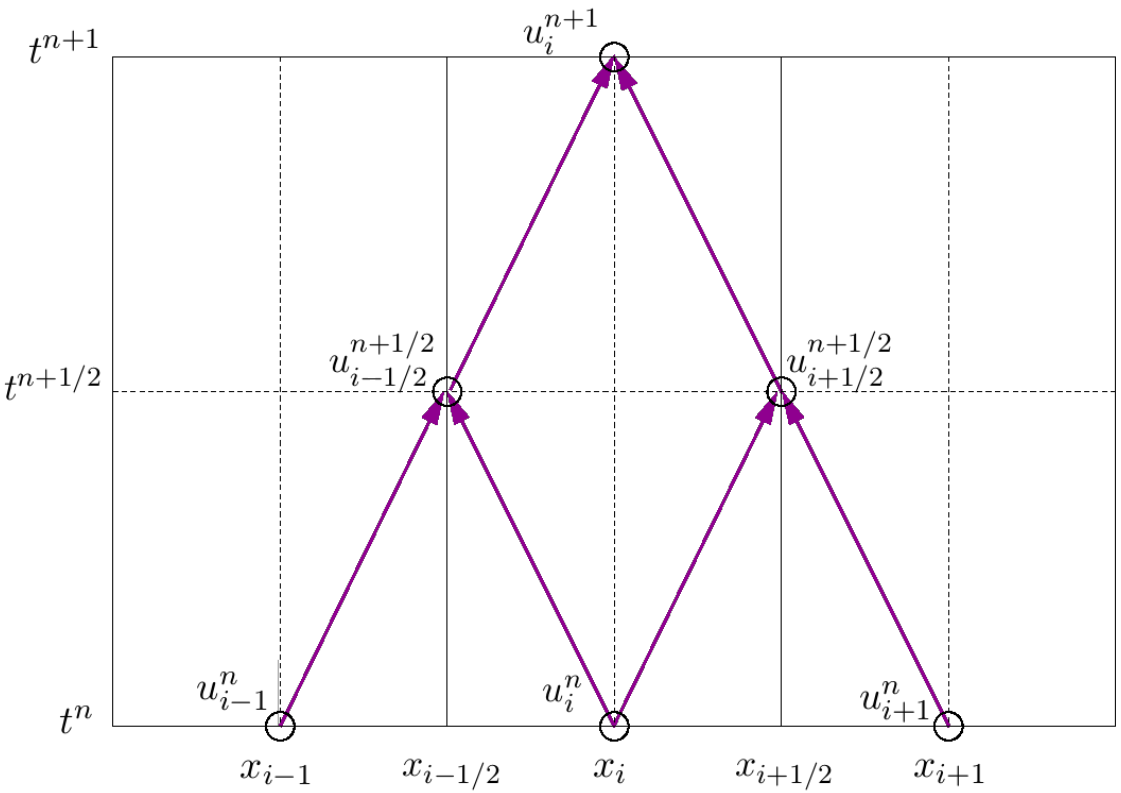
$$\mathbf{f}_{i+1/2}^{\mathrm{RI}} = \mathbf{f}\left(\mathbf{u}_{i+1/2}^{n+1/2}\right)$$

$$\mathbf{u}_{i+1/2}^{n+1/2} = \frac{1}{2}\left(\mathbf{u}_i^n + \mathbf{u}_{i+1}^n\right) - \frac{1}{2}\frac{\Delta t}{\Delta x}\left(\mathbf{f}\left(\mathbf{u}_{i+1}^n\right) - \mathbf{f}\left(\mathbf{u}_i^n\right)\right)$$

- Why does this average work?

Deriving the FORCE flux

- Perhaps unsurprisingly, stable numerical methods are rarely created by combining two others and hoping for the best...
- Instead, it considers updating to time t^{n+1} in steps



Deriving the FORCE flux

- We need to be able to obtain solutions at time $t^{n+1/2}$ at positions $x_{i\pm1/2}$
- We can once again revisit the Lax-Friedrichs update

$$\mathbf{u}_{i-1/2}^{n+1/2} = \frac{1}{2}\left(\mathbf{u}_{i-1}^{n} + \mathbf{u}_{i}^{n}\right) - \frac{1}{2}\frac{\Delta t}{\Delta x}\left(\mathbf{f}\left(\mathbf{u}_{i}^{n}\right) - \mathbf{f}\left(\mathbf{u}_{i-1}^{n}\right)\right)$$

$$\mathbf{u}_{i+1/2}^{n+1/2} = \frac{1}{2}\left(\mathbf{u}_{i}^{n} + \mathbf{u}_{i+1}^{n}\right) - \frac{1}{2}\frac{\Delta t}{\Delta x}\left(\mathbf{f}\left(\mathbf{u}_{i+1}^{n}\right) - \mathbf{f}\left(\mathbf{u}_{i}^{n}\right)\right)$$

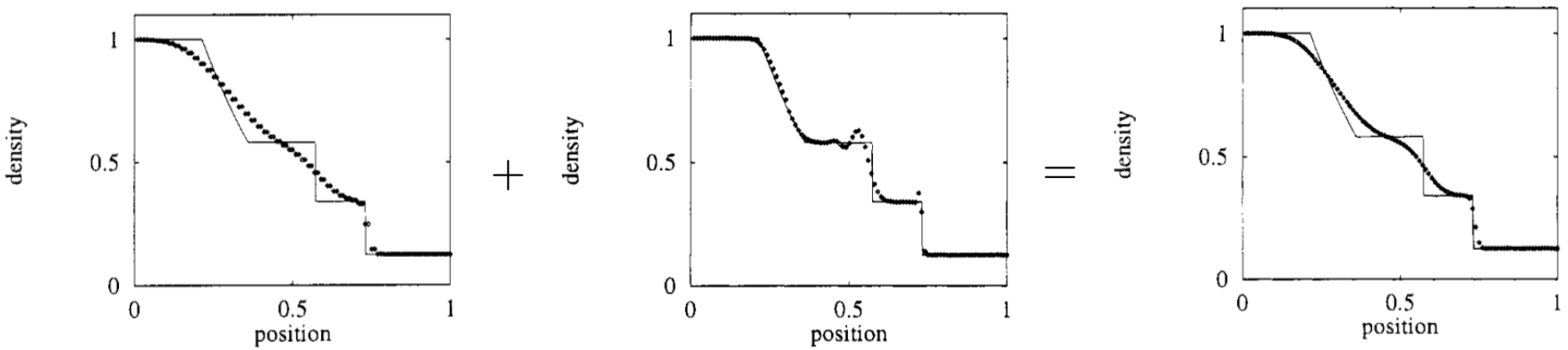
- The Richtmyer flux we saw previously then used these values to compute fluxes in the conservative update formula
- Instead, we could use **another** Lax-Friedrichs update

$$\mathbf{u}_{i}^{n+1} = \frac{1}{2}\left(\mathbf{u}_{i-1/2}^{n+1/2} + \mathbf{u}_{i+1/2}^{n+1/2}\right) - \frac{1}{2}\frac{\Delta t}{\Delta x}\left(\mathbf{f}\left(\mathbf{u}_{i+1/2}^{n+1/2}\right) - \mathbf{f}\left(\mathbf{u}_{i-1/2}^{n+1/2}\right)\right)$$

Combining two fluxes

- Mathematically, this two-step update can be rewritten to obtain conservation law-form fluxes (as we did for the original Lax-Friedrichs flux)
- Doing this gives us the result

$$\mathbf{f}_{i+1/2}^{\mathrm{FORCE}} = \frac{1}{2}\left(\mathbf{f}_{i+1/2}^{\mathrm{LF}} + \mathbf{f}_{i+1/2}^{\mathrm{RI}}\right)$$

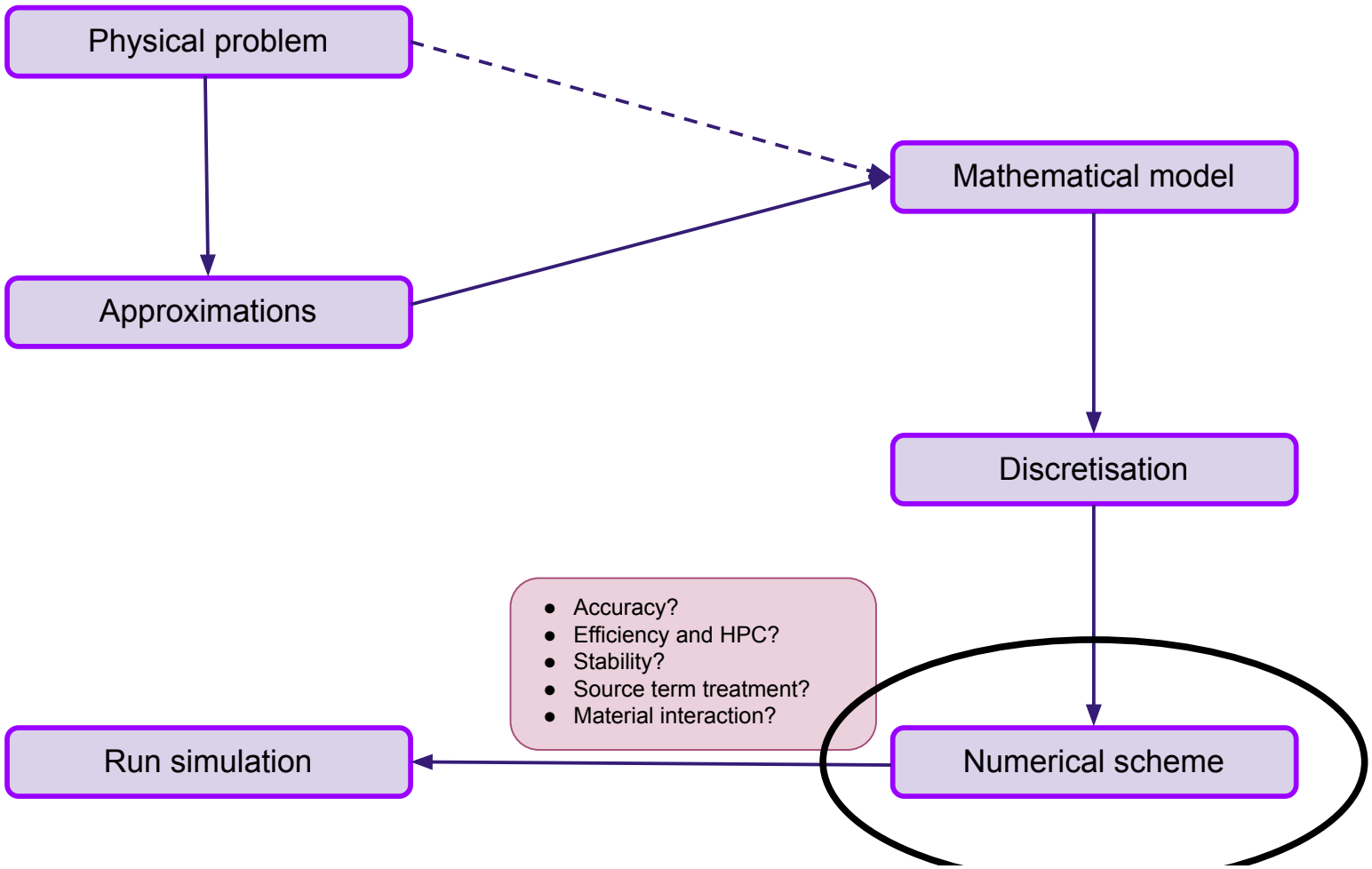


- Note that Toro originally used a different approach to derive this flux (with the same outcome)

Outline

1. Finite volume methods
2. Centred schemes
3. Riemann problem-based schemes
4. Writing finite volume codes

Riemann problem-based schemes



Riemann problems in numerical methods

- We will now consider how Riemann problems fit into numerical schemes
- Riemann problem-based methods are able to provide solutions with less diffusion and dissipation of sharp features than centred schemes
- To define a method, we need to consider two things:
  1. How do we define a Riemann problem from our finite volume approximation?
  2. How do we use the Riemann problem solution to define a flux?
- This also assumes that we know how to solve a Riemann problem for our system of equations
- For the scalar equations we have considered so far, this is straightforward, for more complex systems of equations, this can be a challenge - we will consider these cases later

Behaviour at cell vertices

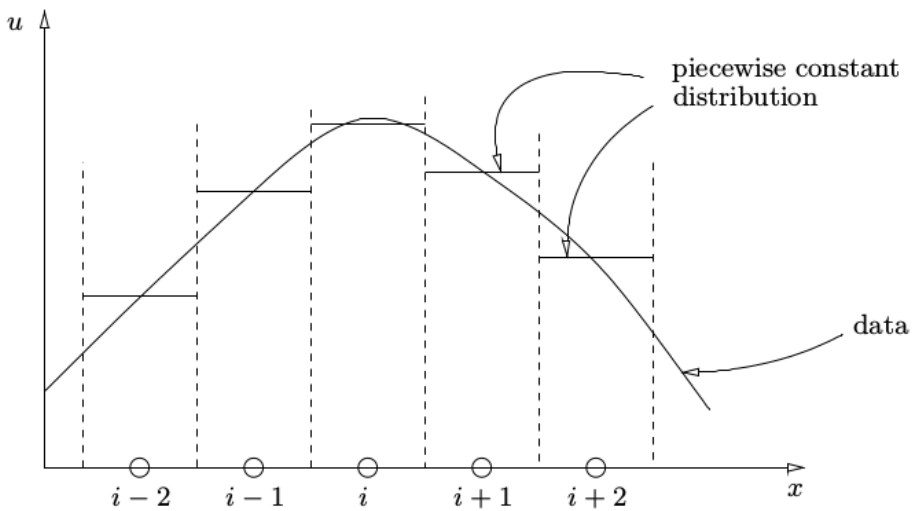
- We stated that we were interested in the solution **along** $x_{i+1/2}$ between t^n and t^{n+1}
- If we assume that at t^n data is constant (and known) in cell $\mathbf{u}_i^n$ and also in $\mathbf{u}_{i+1}^n$, though not the same constant, then we have a Riemann problem centred on $x_{i+1/2}$
- If we know the solution to this Riemann problem, the we know the solution at $x_{i+1/2}$ for time $t > t^n$
- However, recall that, when we considered the characteristics of a conservation law, we found that these were straight lines originating from our initial centre point
- This tells us that the solution of the Riemann problem between cells x_i and x_{i+1} at $x_{i+1/2}$ is constant for $t > t^n$, either by being one side of a shock wave, or in the middle of a rarefaction
- This means we know the we know the flux across $x_{i+1/2}$, since $\mathbf{f}_{i+1/2}^n = \mathbf{f}\left(\mathbf{u}(x_{i+1/2}, t)\right)$, and for $t > t^n$, this is the flux function evaluated for the solution to the Riemann problem

Behaviour at cell vertices

- For the piecewise-constant data we considered, the solution along the line $x_{i+1/2}$ can can be calculated exactly
- In other words, for this case, we are recovering the exact solution to the PDE
- **However**, would we normally expect piecewise-constant data in each cell?
- In general, we only know the integral average of data within a cell, not how it is distributed within the cell
- For a Riemann problem-based method, we have to make some assumption as to how this material is distributed, i.e. what values are present at cell vertices
- This is where **discretisation errors**, and order of accuracy, exist in Riemann problem-based methods
- If fact, the assumption of piecewise-constant data is a first-order accurate representation of data within a finite volume cell

Godunov's method

- Godunov is recognised as the first person do document the use of Riemann problems to solve PDEs
- The method we have just considered is **Godunov's method**



1. Assume that your integral averages $\mathbf{u}_i^n$ are piecewise constant
2. Compute the Riemann problem solution between each neighbouring $\mathbf{u}_i^n$ and $\mathbf{u}_{i+1}^n$
3. This gives $\mathbf{u}_{i+1/2}(t)$ for some $t > t^n$
4. The flux is then $\mathbf{f}_{i+1/2}^n = \mathbf{f}(\mathbf{u}_{i+1/2})$

Return to time steps

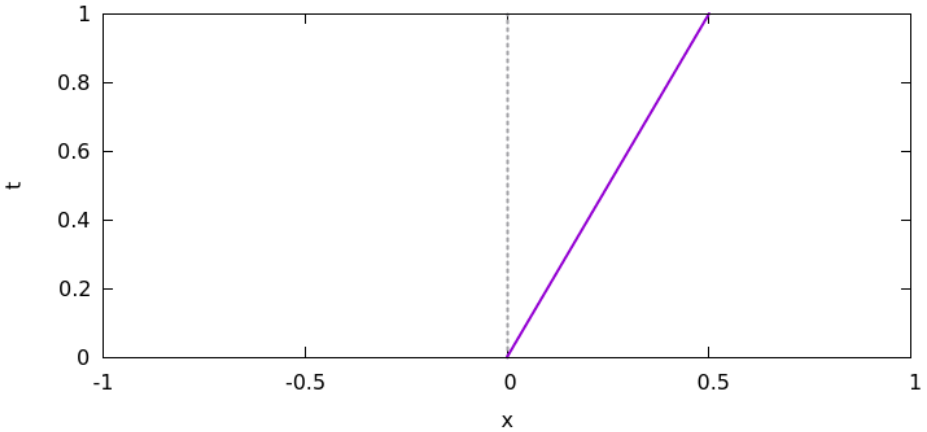
- We just made the statement "This gives $\mathbf{u}_{i+1/2}(t)$ for some $t > t^n$"
- The idea of Godunov's method is to solve a Riemann problem at **every** cell boundary
- This solution will generate wave behaviour (e.g. for Burgers' equation, either a rarefaction or a shock)
- Clearly, the solution originating from $x_{i+1/2}$ will, at some point, cross the cell boundary into $x_{i-1/2}$ or $x_{i+3/2}$ (or both)
- At this point, we have **another Riemann problem** using information from cell x_{i-1}
- However, the stability of our numerical method required that information from cells outside the stencil of the method was not used
- In other words, our time step is chosen such that a wave from cell $x_{i-1/2}$ cannot meet the boundary of $x_{i+1/2}$

Godunov's method for the advection equation

- Recall the solution to the Riemann problem for the advection equation was a discontinuity along the line $t \propto x/a$

- For a cell boundary, the solution along $x_{i+1/2}$ for $t > t^n$ is then

$$u_{i+1/2}(t) = \begin{cases} u_i^n & a > 0 \\ u_{i+1}^n & a < 0 \end{cases}$$



- Assuming $a > 0$, the Godunov flux is then given by

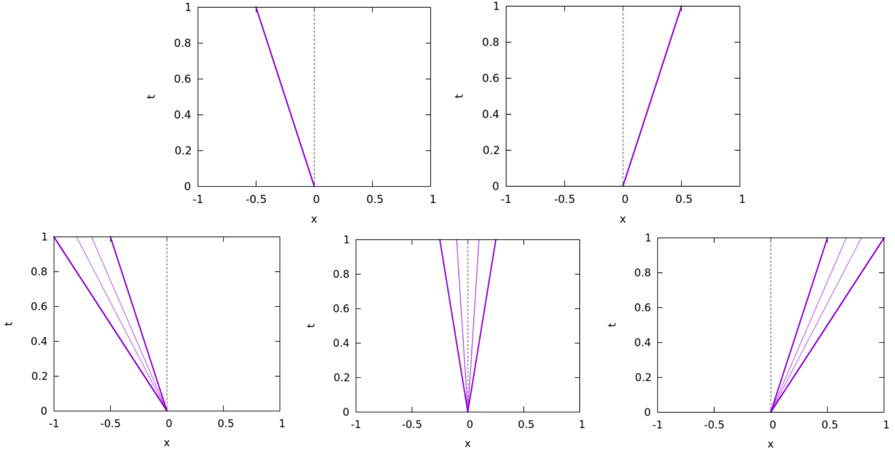
$$f^{\text{God}}_{i-1/2} = f(u_{i-1/2}) = a u_{i-1}, \qquad f^{\text{God}}_{i+1/2} = f(u_{i+1/2}) = a u_i$$

- Note - for the advection equation, the Godunov scheme **is** the upwind scheme, i.e. if we have $a > 0$, then

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

Riemann problem solutions for Burgers' equation

- Five possible solutions



Riemann problem solutions for Burgers' equation

- We are assuming we are solving Burgers' equation for a Riemann problem at $x_{i+1/2}$, between constant states u_i and u_{i+1}
- In order to know the solution to Burgers' equation at a cell boundary, we need to know what **type** of wave is present, and in what **direction(s)** it moves
- For this, we need to know the wave speeds; the shock speed for a shock, and the two outermost rarefaction wave speeds (sometimes called the **head** and **tail** of the rarefaction)
- Recall that rarefactions are caused when characteristics **diverge**, and shocks are caused when they **meet**
- For shocks, the shock speed (slope in the $x-t$ diagram) was given by

$$S = \frac{1}{2}\left(u_i^n + u_{i+1}^n\right)$$

- For Burgers' equation, the characteristics have slope u, hence a rarefaction is bounded by lines with slope $1/u_i$ and $1/u_{i+1}$

Riemann problem solutions for Burgers' equation

$$u_{i+1/2} = \begin{cases} u_i^n & u_i^n > u_{i+1}^n, \quad s > 0 \\ u_{i+1}^n & u_i^n > u_{i+1}^n, \quad s < 0 \\ \\ u_i^n & u_i^n < u_{i+1}^n, \quad u_i^n > 0 \\ 0 & u_i^n < u_{i+1}^n, \quad u_i^n \leq 0 \leq u_{i+1}^n \\ u_{i+1}^n & u_i^n < u_{i+1}^n, \quad u_{i+1}^n < 0 \end{cases}$$

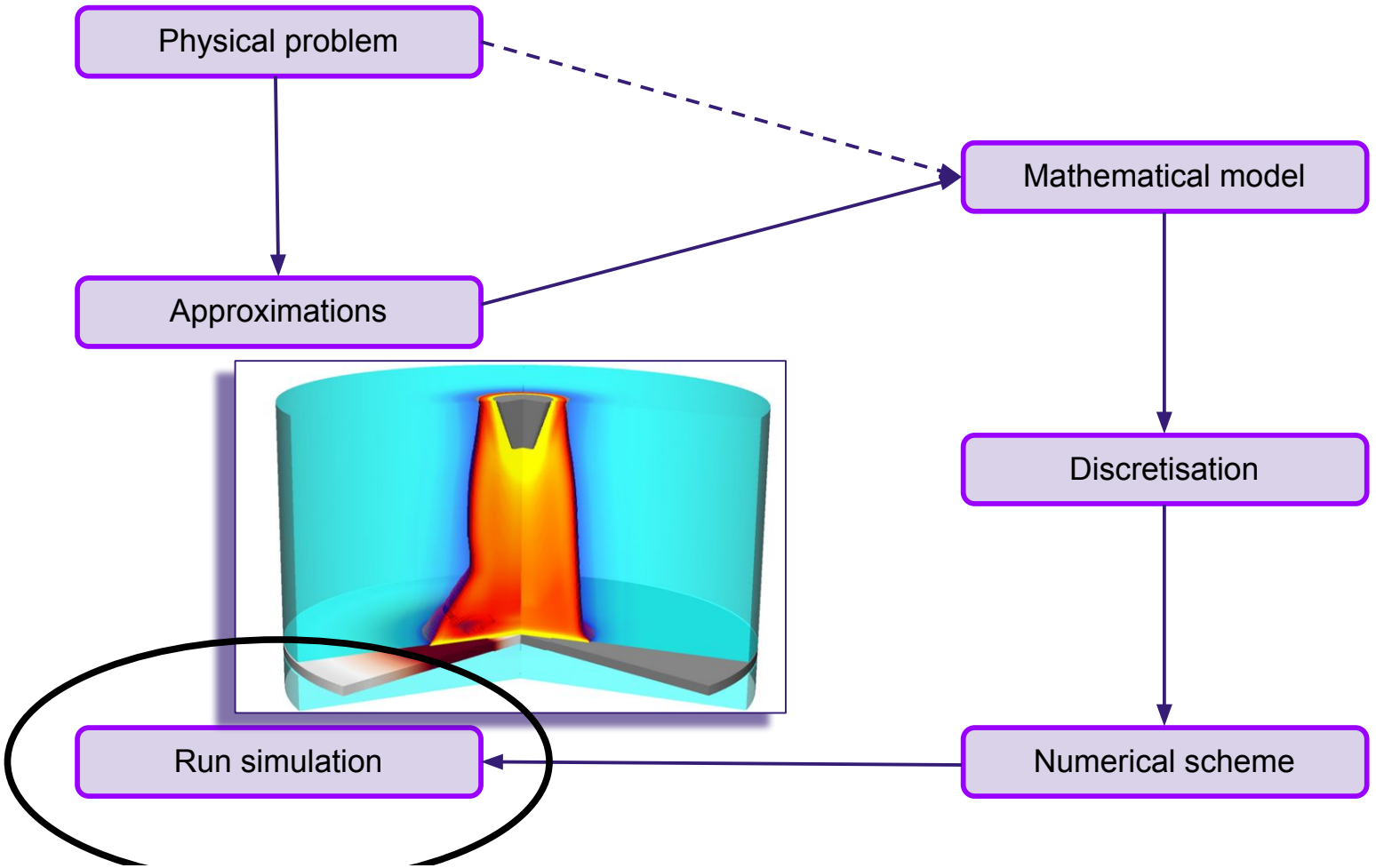
- Note that in the case of the rarefaction covering $x_{i+1/2}$, we know the characteristics are straight lines, and $u = 0$ is a characteristic along $x = \text{const}$
- Also note in the case $u_i^n = u_{i+1}^n$ then the wave speed must be u_i^n (we have parallel characteristics in the Riemann problem solution)
- The Godunov scheme is then

$$u_i^{n+1} = u_i^n - \frac{\Delta t}{\Delta x}\left(f(u_{i+1/2}) - f(u_{i-1/2})\right) = u_i^n - \frac{\Delta t}{\Delta x}\left(\frac{1}{2}(u_{i+1/2})^2 - \frac{1}{2}(u_{i-1/2})^2\right)$$

Outline

1. Finite volume methods
2. Centred schemes
3. Riemann problem-based schemes
4. Writing finite volume codes

Finite volume codes



Constructing a finite volume code

- The code you wrote for your advection equation does not need to change much
- There are a few cosmetic changes, you may wish to refer to `nPoints` as `nCells` in these codes
- One of the more subtle changes is that `x0` and `x1` now refer to **cell boundaries**, which changes how you compute x values
- Consider (again) setting initial data as a sine wave, and assuming our vector has size `nCells + 2`

```
for(int i = 0; i < u.size(); i++) {
  // x_0 is at point i=1
  double x = x0 + (i-0.5) * dx;
  u[i] = sin(x); // Is this a finite volume representation?
}
```

- Other, more important changes, are the choice of time step, and the definition of a **flux array**

Initial data for finite volume methods

- The initial data shown on the previous slide was computed using a finite difference approximation

```
u[i] = sin(x);
```

- In reality, the value u_i^0 should be given by

$$u_i^0 = \int_{x_{i-1/2}}^{x_{i+1/2}} \sin x \mathrm{d}x$$

- For this initial data, this is straightforward, but what if the integral cannot be evaluated exactly (e.g. a Gaussian)?
- From a computational perspective, initial data is only calculated once, it doesn't matter too much if it is an expensive procedure (e.g. an iterative solver)
- However, $u_i^0 = u(x_i)$ is a **second-order accurate** approximation, for first-order schemes (and second-order), these errors will not dominate the solution
- In reality, initial data is rarely so well defined (e.g. contains error bars)

Choosing a time step for non-linear problems

- Recall that for the advection equation (and for a stable numerical method) the maximum time step was given by

$$\Delta t = C\frac{\Delta x}{|a|}$$

- Here, a, the advection coefficient, was also referred to as the **wave speed**
- For Burgers' equation (and other non-linear equations), the wave speed is not constant, either in time or space
- We need a single time step for every cell, and this must be such that every cell has a stable update
- This means we replace the advection coefficient with the **maximum** wave speed over the domain

$$\Delta t = C\frac{\Delta x}{a_{\max}}$$

Defining the maximum wave speed for Burgers' equation

- We have already stated the possible wave speeds for Burgers' equation at a cell boundary $x_{i+1/2}$; they were needed for solving the Riemann problem

$$a_{i+1/2} = S = \frac{1}{2}\left(u_i^n + u_{i+1}^n\right) \quad \text{or} \quad a_{i+1/2} = \max\left(\left|u_i^n\right|, \left|u_{i+1}^n\right|\right)$$

- One way to compute the maximum wave speed is then to take the maximum value across all these boundary cells

$$a_{\max} = \max_i \left(\left|a_{i+1/2}\right|\right), \quad i \in [0, M]$$

- Note, this must include **every** cell boundary, even those where left or right state is defined by boundary conditions
- An equivalent statement, and far less cumbersome to implement is

$$a_{\max} = \max_i \left(\left|u_i^n\right|\right)$$

Defining a flux array

- When using finite volume methods, it is often convenient to place the inter-cell fluxes in their own array (or vector for scalar equations)
- This is because $f_{i+1/2} = f_{(i+1)-1/2}$, and we can avoid calculating this flux twice, once for cell i and once for $i+1$
- Although this may not seem too important for simple PDEs such as Burgers' equation, for more complex equations, flux calculation can be computationally expensive
- The flux array stores values at cell boundaries, if we have `nCells` in our domain, how big does the flux vector have to be?

Updating the data - finite volume methods

```
double t = tStart;
do {
  // Compute the stable time step for this iteration
  dt = computeTimeStep(u);  // You need to define this function
  t = t + dt;
  //You may want to manually reduce dt if this would overshoot tStop

  //Apply boundary conditions
  u[0] = <choice of boundary>;
  u[nCells+1] = <choice of boundary>;

  for(int i = 0; i < nCells+1; i++) { //Define the fluxes
    // flux[i] corresponds to cell i+1/2 // You need to define this
    flux[i] = getFlux(u[i],u[i+1]);
  }

  for(int i = 1; i < nCells+1; i++) { //Update the data
    uPlus1[i] = u[i] - (dt/dx) * (flux[i] - flux[i-1]);
  }
  // Now replace u with the updated data for the next time step
  u = uPlus1;
} while (t < tStop);
```