ENGINEERING TRIPOS PART IIA
Module 3A3: The equations of fluid flow and their numerical solution

Dr Anurag Agarwal (anurag.agarwal@eng.cam.ac.uk)

Handout 1

Some numerical basics

Before examining numerical techniques applicable to fluid flow, let us examine some simple equations, which we can solve by other (analytic) means so that we can check the results. These would allow us to learn the basics of numerical methods without going into the complexity of the fluid flow equations.

Consider the simple harmonic motion described by the equation

$$\frac{d^2y}{dt^2} + \Omega^2 y = 0 \tag{1}$$

subject to the boundary conditions:

$$y(0) = 0, \ \frac{dy}{dt}(0) = \Omega$$

We know that the analytical solution is given by $y(t) = \sin \Omega t$. Now let us try to solve Eq. (1) numerically. In order to do so we need a numerical scheme. The most popular is the finite-difference scheme.

Finite difference scheme

The natural way to represent a function of time numerically is to divide time into discrete steps each of length Δt, and to keep track of the values of y at the end of each time interval. Denote the value of y at $t = n\Delta t$ by y_n ($n = 0, 1, 2, \ldots$) as shown in Fig. 1. A numerical estimate of the derivative of y at time $n\Delta t$ can be found by considering the slope of the line joining successive points.

$$\frac{dy}{dt} \approx \frac{\Delta y}{\Delta t} = \frac{y_{n+1} - y_n}{\Delta t} \tag{2}$$

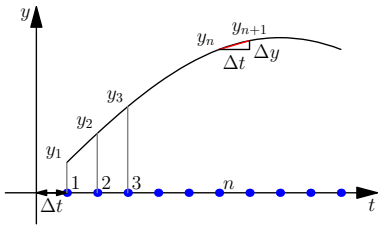


Figure 1: Discretisation to obtain the derivative. $y_1 = y(\Delta t)$, $y_2 = y(2\Delta t) \cdots y_n = y(n\Delta t)$.

One can think of this as approximating the function over the interval Δt by a simple function (in this case a straight line) and using this approximation as a means of estimating the derivative.

Eq. (2) represents a finite-difference formula. Finite difference formulae can be easily derived from Taylor series expansions:

$$y(t + \Delta t) = y(t) + \Delta t \frac{dy}{dt}\Big|_t + \frac{\Delta t^2}{2!}\frac{d^2y}{dt^2}\Big|_t + \frac{\Delta t^3}{3!}\frac{d^3y}{dt^3}\Big|_t + \cdots$$

Rearranging this equation gives where $O(\Delta t)$ indicates that the terms neglected are at most of order

$$\frac{dy}{dt}\Big|_t = \frac{y(t + \Delta t) - y(t)}{\Delta t} + O(\Delta t) \tag{3}$$

This expression is the same as the one we derived earlier, see Eq. (2).

Δt. It is the leading error term. For small Δt the error incurred in making this approximation would be small. The order of the leading error term gives the accuracy of the scheme: $O(\Delta t^n)$ indicates that the *order of accuracy* is n. Thus for the finite difference formula in Eq. (3) the order of accuracy is 1.

There are three *forms* of finite-difference schemes: forward, backward and central differences.

A *forward difference* scheme uses points at or forward of the point at which the derivative is desired, e.g. the formula we have just derived

$$\left.\frac{dy}{dt}\right|_n = \frac{y_{n+1} - y_n}{\Delta t}$$

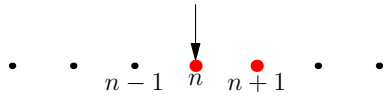


Figure 2: Forward difference scheme. Red dots indicate the grid points used to calculate the derivative at the location pointed by the arrow.

A *backward difference* scheme uses points behind the point at which the derivative is desired. A backward difference formula for the first derivative is given by[1]

$$\left.\frac{dy}{dt}\right|_n = \frac{y_n - y_{n-1}}{\Delta t}$$

[1] This is obtained by using the Taylor series in Eq. (5)

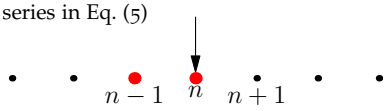


Figure 3: Backward difference scheme.

We'll see an example of a central difference scheme below. In order to discretise Eq. (1) we need a finite difference formula for the second derivative, d^2y/dt^2. This can be obtained by considering two Taylor series:

$$y(t+\Delta t) = y(t) + \Delta t \left.\frac{dy}{dt}\right|_t + \frac{\Delta t^2}{2!}\left.\frac{d^2y}{dt^2}\right|_t + \frac{\Delta t^3}{3!}\left.\frac{d^3y}{dt^3}\right|_t + \cdots \quad (4)$$

$$y(t-\Delta t) = y(t) - \Delta t \left.\frac{dy}{dt}\right|_t + \frac{\Delta t^2}{2!}\left.\frac{d^2y}{dt^2}\right|_t - \frac{\Delta t^3}{3!}\left.\frac{d^3y}{dt^3}\right|_t + \cdots \quad (5)$$

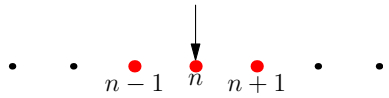


Figure 4: Central difference scheme.

Adding the two series gives

$$\frac{d^2y}{dt^2} = \frac{y(t+\Delta t) - 2y(t) + y(t-\Delta t)}{\Delta t^2} + O(\Delta t^2) \quad (6)$$

The finite-difference formula for the second derivative is obtained by neglecting terms of order Δt^2. Thus the order of accuracy is 2. Note that this is a *central difference* formula as points both forward and behind are used in a symmetric way to calculate the derivative at a point.

Substituting this discrete form of the second derivative (Eq. (6)) in Eq. (1), we get

$$\frac{y(t+\Delta t) - 2y(t) + y(t-\Delta t)}{\Delta t^2} + \Omega^2 y(t) = 0$$

or, in terms of our grid points

$$\frac{y_{n+1} - 2y_n + y_{n-1}}{\Delta t^2} + \Omega^2 y_n = 0$$

The difference equation can be recast as

$$y_{n+1} = (2 - \Omega^2 \Delta t^2) y_n - y_{n-1} \quad (7)$$

To solve this we start with $n = 1$ and march forward in n. For $n = 1$ we need two initial conditions y_0 and y_1. When these are known, y_2 and all succeeding values can be generated in turn.

The initial conditions are $y(0) = 0$ and $dy/dt(0) = \Omega$, which become

$$y_0 = 0 \qquad \text{and} \qquad \frac{y_1 - y_0}{\Delta t} = \Omega$$

Thus, $y_1 = \Omega \Delta t$.

Let us find the solution for the case of $\Omega = 1$. Let us set the time step Δt to 0.1. Then from Eq. (7)

$$y_2 = (2 - \Omega^2 \Delta t^2) y_1 - y_0 = 0.199$$

Subsequent values in time can be easily generated by marching forwards in time. The solution is shown in Fig. 5

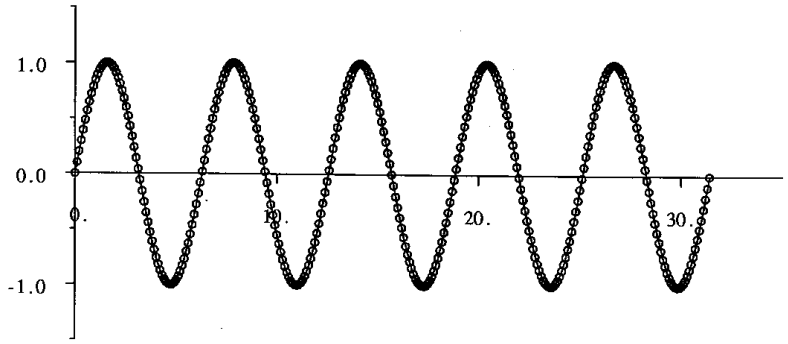


Figure 5: $\Delta t = 0.1$

The circles are generated from the difference equation, while the solid line is the analytic, exact solution. It is difficult to see the error.

We have followed this procedure for a trivial equation. It applies without modification to much more complicated equations which cannot be solved analytically. Turning the differential equation into a difference equation is just as easy for complicated equations. E.g.

$$\frac{d^2y}{dt^2} + e^{-t} \sin y^2 = t^2$$

$$\frac{d^2y}{dt^2} - e^{-n\Delta t} \sin y_n^2 = n^2 \Delta t^2$$

This implies,

$$y_{n+1} = \Delta t^2 (n^2 \Delta t^2 - e^{-n\Delta t} \sin y_n^2) + 2y_n - y_{n-1}$$

Like for the simple-harmonic oscillator this equation can be solved given the initial conditions.

The very satisfactory agreement obtained in the above figure is far from the whole story. For instance, if we increase Δt from 0.1 to 0.5 the solution is not very accurate as shown in Fig. 6. This is understandable because we expect the solution to be less accurate

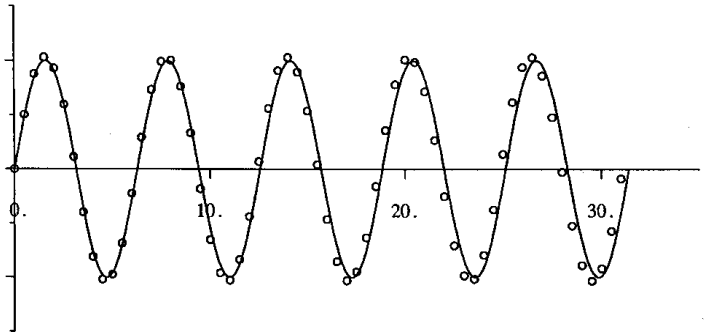


Figure 6: $\Delta t = 0.5$

for larger Δt. But if we increase Δt to 2.000001 then the solution becomes unstable (!) as can be seen in Fig. **??**. So one cannot

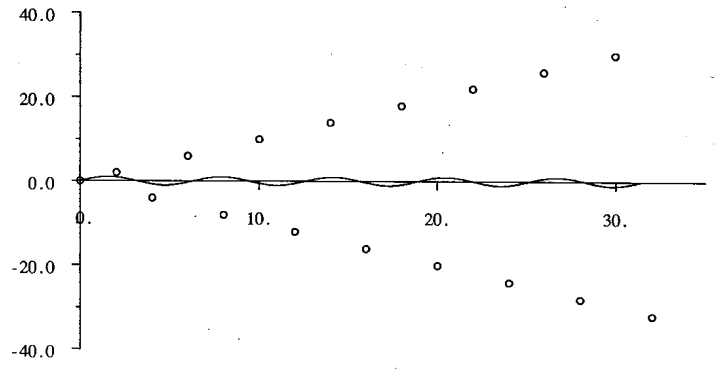


Figure 7: $\Delta t = 2.000001$.

blindly discretise and solve an equation. We need to know about the stability characteristics of the scheme. We will come back to this later.

Heat conduction equation

Let us consider a similar exercise for the scalar diffusion equation (e.g. heat conduction equation in a bar, with the temperatures of the ends held constant).

$$\frac{\partial u}{\partial t} = \nu \frac{\partial^2 u}{\partial x^2}$$

with boundary conditions $u(0) = u(L) = 0$. At time $t = 0$ the initial value of u is given by

$$u(x,0) = \begin{cases} 10\frac{x}{L} & 0 < x < \frac{L}{2} \\ 10\left(1 - \frac{x}{L}\right) & \frac{L}{2} < x < L \end{cases} \tag{8}$$

This time we discretise space *and* time and denote our approximation to u at $x = i\Delta x$ $(1 \leq i \leq N_x)$ and $t = n\Delta t$. If we use a central difference approximation for $\partial^2 u/\partial x^2$ and a forward difference

scheme for $\partial u/\partial t$, we have:

$$\frac{\partial u}{\partial t} \approx \frac{u_i^{n+1} - u_i^n}{\Delta t}$$

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

Therefore the difference equation for the heat equation becomes

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \nu \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} \tag{9}$$

Using Taylor series expansion we can show that

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{\partial u}{\partial t}\Big|_{i,n} + \frac{\Delta t}{2!}\frac{\partial^2 u}{\partial t^2}\Big|_{i,n} + \frac{\Delta t^2}{3!}\frac{\partial^3 u}{\partial t^3}\Big|_{i,n} + \cdots = \frac{\partial u}{\partial t} + O(\Delta t)$$

$$\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} = \frac{\partial^2 u}{\partial x^2}\Big|_{i,n} + 2\frac{\Delta x^2}{4!}\frac{\partial^4 u}{\partial x^4}\Big|_{i,n} + \cdots = \frac{\partial^2 u}{\partial x^2} + O(\Delta x^2)$$

So that the truncation error is now $O(\Delta t, \Delta x^2)$, so our approximation is first order accurate in time and second order accurate in space. Eq. (9) can be rearranged to give

$$u_i^{n+1} = u_i^n + \frac{\nu \Delta t}{\Delta x^2}\left(u_{i+1}^n - 2u_i^n + u_{i-1}^n\right) \quad \text{for } 1 < i < N_x$$

Note that we do not solve this equation for $i = 1$ or N_x. This is because we implement the boundary conditions at these points.

Now we can march forwards in time. We know the initial conditions ($n = 0$) for all i. Use this to generate u_i^{n+1} for $1 < i < N_x$. Obtain u_i^{n+1} for $i = 1$ and N_x by using the boundary conditions.

The code in Matlab is shown below

```
nx=101;
for i=1:nx
    x(i) = (i-1)*1./(nx-1);
    if i < nx/2
        u(i) = 10*x(i);
    else
        u(i) = 10*(1.-x(i));
    end
end
hold on;
plot(x,u,'r','LineWidth',1.5);

al = .49;

nt = 2000;
nplot=100;
for n=1:nt
```

```
    for i=2:nx-1
        un(i) = u(i)+al*(u(i+1)-2*u(i)+u(i-1));
    end
    un(1) = 0;
    un(nx) = 0;
    if rem(n,nplot)==0
        plot(x,un,'b','Linewidth',1.5);
    end
    u = un;
end
hold off;
```

The solution is shown below for two different values of time-steps. Just a minor difference in time step makes a big difference in the solution. Again we need to know something about the stability of the scheme.

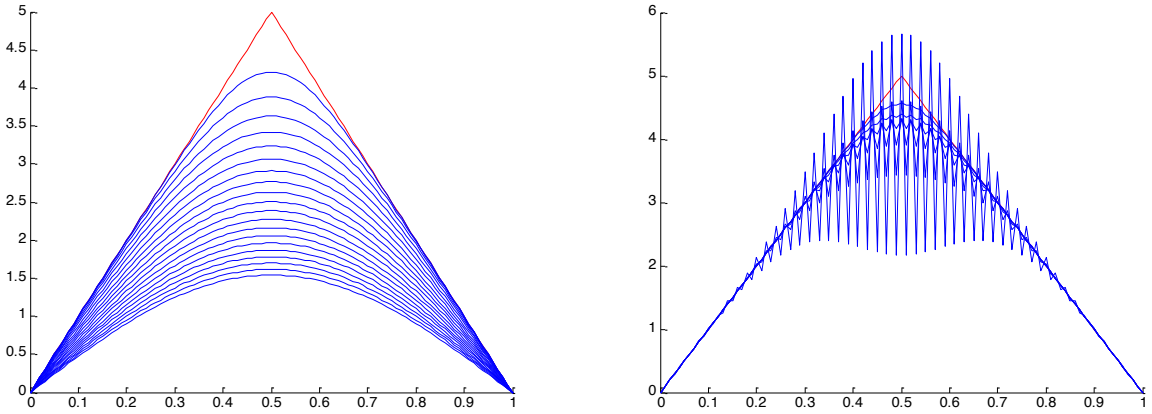


Figure 8: Solution to the heat equation. Red: initial condition. The succession of blue curves represent the solution as we march forwards in time. Left: $\frac{\nu\Delta t}{\Delta x^2} = .49$, Right: $\frac{\nu\Delta t}{\Delta x^2} = .51$

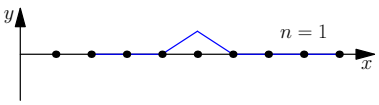
Stability analysis

Let us investigate the stability of the solution. It is clear from the pictures that the mechanism of breakdown is for disturbances that alternate in sign at alternate grid points (nodes). We can analyse this by imagining the finite difference solution to be composed of two parts, the exact steady solution $\bar{u}_i^n$ and a perturbation ϵ_i^n,

$$u_i^n = \bar{u}_i^n + \epsilon_i^n$$

We substitute this into the finite difference equations and follow the growth or decay of ϵ. This approach leads, in general, to strong physical insight into how the numerical model mimics the underlying physics of the governing differential equation.

Introduce a point disturbance at point i as shown in the figure into the finite difference scheme at $n = 1$,



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

we can work out the solution at subsequent timesteps by marching forward in time. Let $\alpha = \nu\Delta t/\Delta x^2$. Let us see what happens to the solution for different values of α.

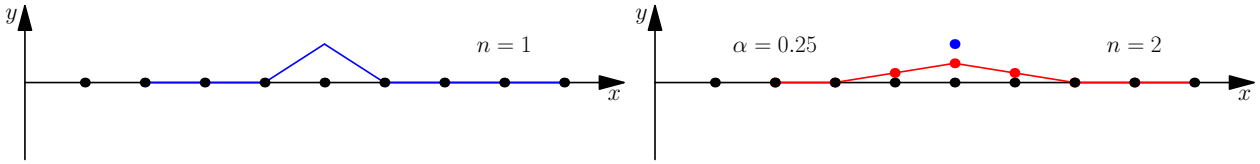


Figure 9: $\alpha = 0.25$

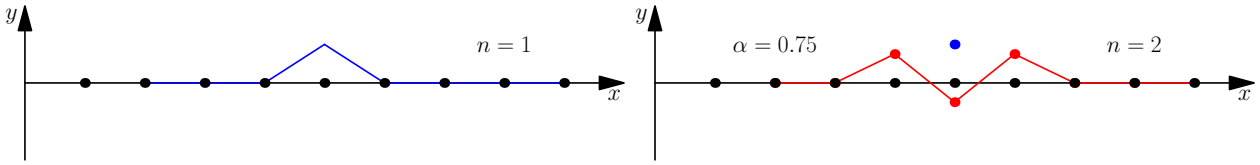


Figure 10: $\alpha = 0.75$

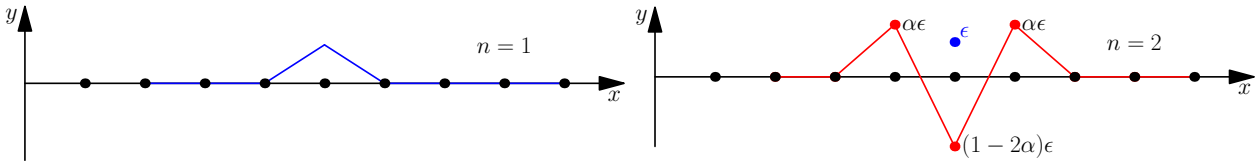
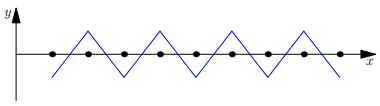


Figure 11: $\alpha = 1.5$

For $\alpha = 0.25$ the disturbance diffuses as can be seen in Fig. 9. If $\alpha > 0.5$ then the perturbation starts to "wiggle" as it spreads out. If $\alpha > 1$, then the "first" disturbed point has a magnitude which grows with increasing time (i.e. the method is unstable).

A general method to investigate instability is to introduce a "sawtooth wiggle" disturbance (where alternate points have the value $\pm\epsilon$), thus giving a grid-to-grid oscillation.



Introducing this disturbance into the finite difference equation gives:

$$u_i^{n+1} = \epsilon + \alpha(-\epsilon - 2\epsilon - \epsilon)$$

$$u_i^{n+1} = \underbrace{u_i^n}_{\epsilon} + \alpha\left(\underbrace{u_{i+1}^n}_{-\epsilon} - 2\underbrace{u_i^n}_{\epsilon} + \underbrace{u_{i-1}^n}_{-\epsilon}\right)$$

$$\implies \frac{u_i^{n+1}}{\epsilon} = 1 - 4\alpha$$

For stability,

$$\frac{|u_i^{n+1}|}{\epsilon} \leq 1 \implies -1 \leq 1 - 4\alpha \leq 1$$

Hence,

$$\alpha = \frac{\nu\Delta t}{\Delta x^2} \leq \frac{1}{2}$$

Giving the stability boundary consistent with the figures. This shows that for a given ν and Δx, the stable choice of Δt is restricted. This stability bound is very restrictive, since if we halve the mesh spacing (to improve accuracy for example) then Δt is reduced by a factor of four. We must therefore perform four times as many time marching steps, each costing twice as much, to get the same physical time - an increase in cost of a factor 8!