# CFD : Coursework 1

$$u_t + a u_x = 0$$

Discretise with (temporally and spatially) 2nd order-accurate schemes:

1. Leap Frog : $u_i^{n+1} = u_i^{n-1} - \sigma\left(u_{i+1}^n - u_{i-1}^n\right)$

2. Angled Derivative : $u_i^{n+1} = u_{i-1}^{n-1} + (1-2\sigma)\left(u_i^n - u_{i-1}^n\right)$

where $\sigma = a\Delta t/\Delta x$ is the Courant number.

## $\boxed{\text{Part 1}}$

$$x \in [-1, 1], \; \Delta x \Rightarrow \underline{x} = \begin{pmatrix} x_1 = -1 \\ x_2 = x_1 + \Delta x \\ \dots \\ x_{n-1} \\ x_n = 1 \end{pmatrix}, \; \underline{u}^t = \begin{pmatrix} u_1 \\ u_2 \\ \dots \\ u_{n-1} \\ u_n = u_1 \end{pmatrix}^t \in \mathbb{R}^n$$

* Periodic BCs

The initial velocity conditions are given for t = 0, and can be obtained for other times by adjusting the solution for the distance travelled by the wave:

$$\begin{cases} t = 0: & u(x,0) = \sin(2\pi x) = f(x) \\ t: & u(x,t) = f(x - at) = \sin(2\pi(x - at)) \end{cases}$$

```
function [u] = Set_IC1(u, x, t, a)  % I recognise that MATLAB doesn't Pass by Reference so no need to pass u here, but I like to do it to be reminded
    u(:) = sin(2*pi*(x - a*t));
end
```

b) An alternative to enforcing the exact solution at the second time step could be using a 2nd order predictor which only requires one previous time step (such as RK2 or Lax Wendroff)

Implementing the schemes, specifying the interior and boundary points separately:

```
function [u3] = LeapFrog(n, u1, u2, u3, c)  % I recognise that u3 isn't Passed by
                                            % Reference, but I like to do it to be reminded

    u3(1)      = u1(1)      - c * ( u2(2)      - u2(n-1));
    u3(2:n-2) = u1(2:n-2) - c * ( u2(3:n-1) - u2(1:n-3) );                 % interior
    u3(n-1)    = u1(n-1)    - c * ( u2(1)      - u2(n-2));

    u3(n) = u3(1);                                                        % periodic bcs

end

function [u3] = AngledDerivative(n, u1, u2, u3, c)

    u3(1)      = u1(n-1)    + (1 - 2*c)*( u2(1)      - u2(n-1));
    u3(2:n-1) = u1(1:n-2) + (1 - 2*c)*( u2(2:n-1) - u2(1:n-2));           % interior

    u3(n) = u3(1);                                                        % periodic bcs

end

function [u2] = LaxWendroff(n, u1, c)

    u2(1)      = 0.5 * c*(1 + c) * u1(n-1)    + (1 - c^2) * u1(1)      - 0.5*c*(1 - c) * u1(2);
    u2(2:n-2) = 0.5 * c*(1 + c) * u1(1:n-3) + (1 - c^2) * u1(2:n-2) - 0.5*c*(1 - c) * u1(3:n-1);
    u2(n-1)    = 0.5 * c*(1 + c) * u1(n-2)    + (1 - c^2) * u1(n-1)    - 0.5*c*(1 - c) * u1(1);

    u2(n) = u2(1);

end
```

With the numerical scheme and initial condition setter defined, the solver is defined as follows:

```
function [u] = Solver1(scheme, nt, nx, ts, xs, a, c)

    u = zeros(nx, nt);
    ut = zeros(nx, 1);

    for i = 1:nt
            if (i == 1) || (i == 2)
                ut = Set_IC1(ut, xs, ts(i), a);     % sets ics using exact solution
                u(:, i) = ut;

            else
                if scheme == "LeapFrog"
                    ut = LeapFrog(nx, u(:, i-2), u(:, i-1), ut, c);

                elseif scheme == "AngledDerivative"
                    ut = AngledDerivative(nx, u(:, i-2), u(:, i-1), ut, c);

                end

                u(:, i) = ut;
            end
    end
end
```

Note that two initial conditions are required for both the Leap Frog and Angled Derivative schemes

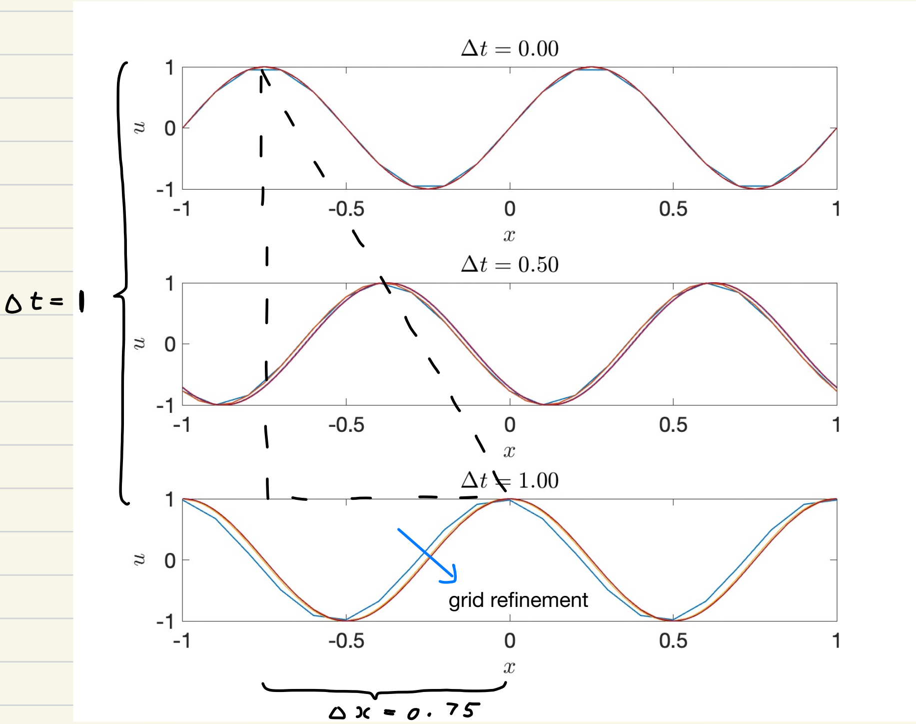
Now we may run our solver. Setting up with the problem parameters:

```
% scheme = "LeapFrog";
scheme = "AngledDerivative";

% Set-up
xd = [-1, 1];
Td = [0, 1];

a = 0.75;
c = 0.6;

dxs = [0.1, 0.05, 0.02, 0.01, 0.005, 0.002, 0.001];
dts = c * dxs / a;
```

And finally iterate for the solution:

```
fig = figure(1);
    labs = "x = " + dxs;

for ix = 1:length(dxs)
    dx = dxs(ix);
    xs = xd(1):dx:xd(2);
    nx = length(xs);

    dt = dts(ix);
    ts = Td(1):dt:Td(2);
    nt = length(ts);

    indt = [1, find(abs(ts - 0.5) == min(abs(ts - 0.5)), 1), find(abs(ts - 1.0) == min(abs(ts - 1.0)), 1)];

    u = Solver1(scheme, nt, nx, ts, xs, a, c);

    Save1(scheme, ix, dx, dt, a, c, ts, xs, u)

    for i = 1:length(indt)
        fig = Plot1(fig, i, ix, xs, u, indt, ts, xd, dx, dxs);
    end

end
hold off;
```

Visualising the (Angled Derivative) solution for various discretisation at t = {0, 0.5, 1}:

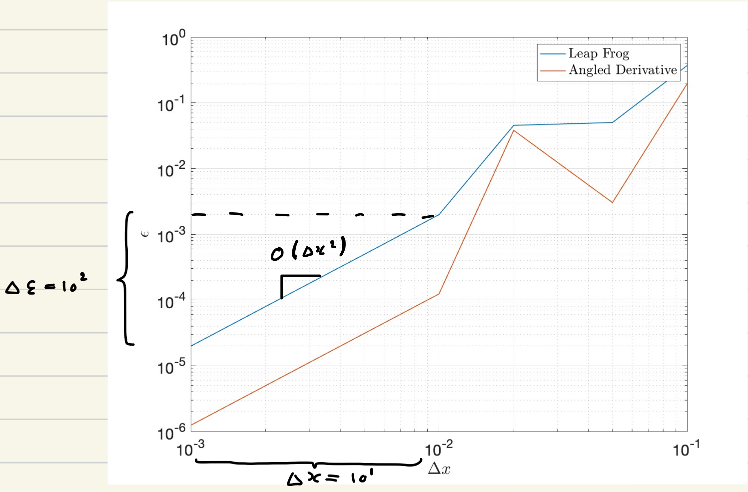



$\Delta x / \Delta t = 0.75 \equiv a$ $\therefore$ Good!

Now compute the error norm at T=1 and plot convergence:

```
eps = zeros(length(dxs), 2);

for ischeme = 1:2
    for igrid = 1:length(dxs)

        if ischeme == 1, fin = sprintf("CW_data/Q1/LF_dx%i", igrid); end
        if ischeme == 2, fin = sprintf("CW_data/Q1/AD_dx%i", igrid); end

        load(fin, 'xs', 'u')
        ut = u(:, end);

        u_exact = zeros(length(xs), 1);
        u_exact = Set_IC1(u_exact, xs, 1, a);

        eps(igrid, ischeme) = max(abs(u_exact - ut));

    end
end

figure(2)
    loglog(dxs, eps(:, 1), 'Color', [0      0.4470 0.7410], 'LineWidth', 1); hold on;
    loglog(dxs, eps(:, 2), 'Color', [0.8500 0.3250 0.0980], 'LineWidth', 1); hold off;
        xlabel('$\Delta x$', 'Interpreter', 'latex')
        ylabel('$\epsilon$', 'Interpreter', 'latex')
        legend(["Leap Frog", "Angled Derivative"], 'Interpreter', 'latex')
        grid on
        set(gca, 'FontSize', 18)
```

c) Inspecting for the convergence rates of the schemes, we find that both are 2nd order accurate:




d) The Lax equivalence theorem states that for a consistent FD method for a well-posed linear initial value problem, the method is convergent if and only if it is stable.

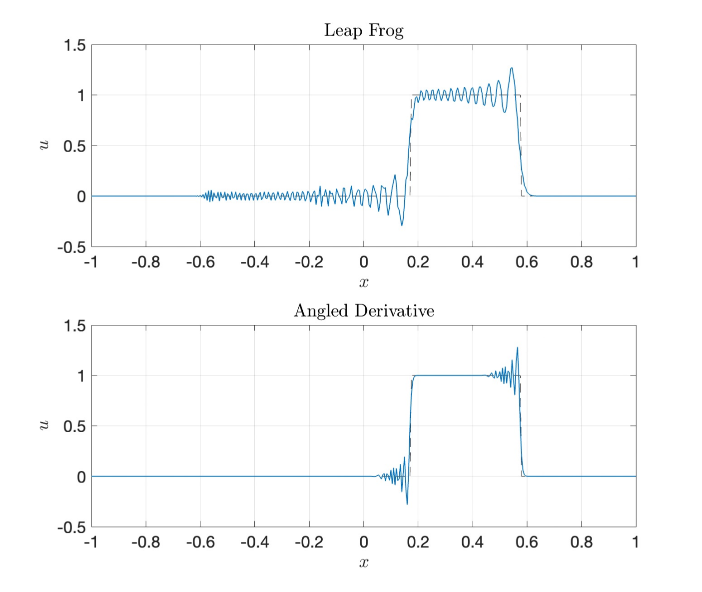
$\therefore$ Since this scheme is convergent and stable, consistency is implied

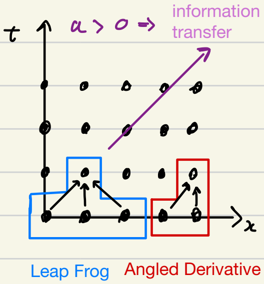
# Part 2

Introducing a new initial condition setter for the first two initial conditions (the first is a step while the second uses the 2nd order scheme which only requires one previous time step):

```
function [u] = Set_IC2(u, x, i, c, n)

    if i == 1
        u((x >= -0.2) & (x <= 0.2)) = 1;
    else
        u = LaxWendroff(n, u, c);
    end

end
```

Solving with both schemes we obtain the following:




Spurious behaviour in Leap Frog scheme may arise from the additional downstream point

The leap frog scheme is significantly more oscillatory and is over diffusive at the sharp corners (it has a flattening effect on the solution):

- Dispersion relates to the oscillations (frequency and phase differences)
- Diffusion relates to the difference in magnitude relative to the exact solution