Differential Equation

Differential equations are ubiquitous in physics – Newton's equation of motion, Maxwell relations in thermodynamics or Maxwell equations of E&M, Schrödinger equations and what not. Many of them can be solved analytically but a vast majority of them cannot be for even apparently simpler system like planetary orbits. Therefore, importance of learning a few numerical tricks to solve ordinary or partial differential equations can never be overstated. In this part of the course we will start with solving first order ODE and then moving to second order and finally PDE. The algorithms for solving ODE also vary depending on whether one is attempting initial value or boundary value problems. Here we will take a look at both.

First order ODE

A first order differential equation (ODE), with initial value(s) given, can generically be written as

$$\frac{dy}{dx} = f(y(x), x), \quad \text{with } y(x_0) = y_0. \tag{1}$$

The dy/dx is basically a tangent to the solution curve $y = y(x)$ at the point x and the initial condition states that at $x = x_0$, $y = y_0$. It is often possible to rewrite second-order differential equation in terms of two coupled first order ODE,

$$\frac{d^2x}{dt^2} = -\omega^2 x \quad \Rightarrow \quad v = \frac{dx}{dt} \text{ and } \frac{dv}{dt} = -\omega^2 x - \mu v \tag{2}$$

with two initial conditions being $x(t_0) = x_0$ and $v(t_0) = v_0$.

To solve the above first order ODEs given an initial condition, we will discuss the following methods,

1. Forward (explicit) and Backward (implicit) Euler's method
2. Predictor-Corrector method
3. Runge-Kutta 4th order

Forward Euler's method : Consider Taylor expansion of the function $y(x_0 + h)$ about x_0, where h is small,

$$y(x_0 + h) = y(x_0) + h \left.\frac{dy}{dx}\right|_{x_0} + \frac{h^2}{2!} \left.\frac{d^2y}{dx^2}\right| x_0 + \cdots \approx y(x_0) + h\, f(y(x_0), x_0) + \mathcal{O}(h^2) \tag{3}$$

If we consider $x_1 = x_0 + h$ to be a small h step away from x_0, then the Forward Euler's method gives the solution of our ODE (1) at x_1 as

$$y(x_1) = y(x_0) + h\, f(y(x_0), x_0) + \mathcal{O}(h^2) \tag{4}$$

In the next step, starting at $x = x_1$ we can use (4) to go to the next step $x_2 = x_1 + h$ and then to $x_3 = x_2 + h$ and so on. Therefore, at the n-th step the solution of the ODE (1) is

$$y(x_n + h) = y(x_n) + h\, f(y(x_n), x_n) \quad \text{or, equivalently } y_{n+1} = y_n + \kappa_1 \tag{5}$$

where $\kappa_1 = f(y(x_n), x_n)$. This forward Euler method depends on the tangent dy/dx calculated at earlier point x_n (*i.e.* beginning of the interval) to obtain the solution at the end of the interval y_{n+1}.

Hence, this method is also regarded as *explicit* Euler method. Why it is called *forward*? It becomes obvious if we look at (1) as discrete derivative,

$$\begin{aligned} \frac{dy}{dx} &= \lim_{\Delta x\to 0}\frac{\Delta y}{\Delta x} = \lim_{\Delta x\to 0}\frac{y(x+\Delta x)-y(x)}{\Delta x} = f(y(x),x) \\ \Rightarrow \qquad & y(x+\Delta x) \approx y(x) + \Delta x\, f(y(x),x). \end{aligned} \tag{6}$$

Euler method developed for first order ODE can be extended to N order ODE by reducing it to a system of coupled first order ODE, as in eqn. (2), which can generically be written as

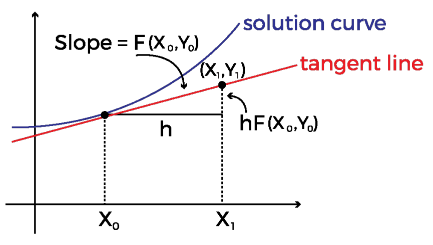
$$\frac{d^N y(x)}{dx^N} = f(x, y(x), y'(x), y''(x), \ldots y^{(N-1)\prime}(x), y^{N\prime}(x)) \tag{7}$$

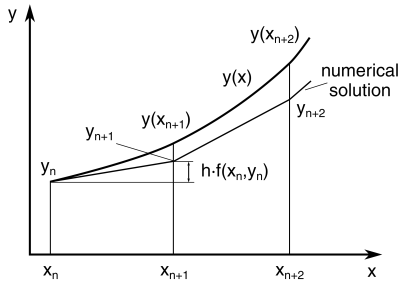
$$\mathbf{y}_{n+1} = \begin{pmatrix} y_{n+1} \\ y'_{n+1} \\ \vdots \\ y^{(N-1)\prime}_{n+1} \\ y^{N\prime}_{n+1} \end{pmatrix} = \begin{pmatrix} y_n + hy'_n \\ y'_n + hy''_n \\ \vdots \\ y_n^{(N-1)\prime} + hy_n^{N\prime} \\ y_n^{N\prime} + hf(x, y(x), y'(x), y''(x), \ldots y^{(N-1)\prime}(x), y^{N\prime}(x)) \end{pmatrix} \tag{8}$$

Local truncation error measures the error made in a single Euler step, which can be estimated by subtracting eqn. (4) from eqn. (3) (say for n=0)

$$y(x_0+h) - y(x_1) = \frac{1}{2}h^2 y''(x_0) + \mathcal{O}(h^3) \tag{9}$$

For small h, thus the local truncation error is proportional to h^2 which make Euler method less accurate for larger h and other higher order method like Runge-Kutta. Apart from accuracy and sluggishness *i.e.* slower convergence to the desired accuracy, the stability is more of a problem with forward Euler. It can easily veer away from the solution as demonstrated in the plots below,





Calcworkshop.com

But in all, forward Euler method often return fairly good approximation to the actual solution $y = y(x)$. To correct stability problem arising from tangent at the beginning of the interval, the next obvious step is the *backward* Euler where the tangent from the end of the interval is considered. Re-define $\Delta y/\Delta x$,

$$\begin{aligned} \frac{\Delta y}{\Delta x} &\approx \frac{y(x)-y(x-\Delta x)}{\Delta x} = f(y(x),x) \quad \Rightarrow \quad y(x) = y(x-\Delta x) + \Delta x\, f(y(x),x) \\ &\Rightarrow \quad y(x+\Delta x) = y(x) + f(y(x+\Delta x), x+\Delta x) \\ &\equiv \quad y(x+h) = y(x) + h\, f(y(x+h), x+h) \end{aligned} \tag{10}$$

The backward Euler thus $y(x+h)$ is determined from the tangent at $x+h$ which, rather strangely, implies that in order to calculate $y(x+h)$ one needs to know $y(x+h)$!! This apparent conflict is resolved by treating eqn. (10) as an algebraic (nonlinear) equation and then using Newton-Raphson to solve

$$y^{\mathrm{NR}}(x+h) = y(x) + h\, f(y^{\mathrm{NR}}(x+h), x+h) \tag{11}$$

The solution $y^{\mathrm{NR}}(x+h)$ is used to determine $y(x+h)$

$$y(x+h) = y(x) + h\, f(y^{\mathrm{NR}}(x+h), x+h). \tag{12}$$

The backward or implicit Euler method, in spite of having an extra step of Newton-Raphson, is advantageous because of better stability over forward method.

The algorithm for forward method is absolutely straight forward,

1. Choose an h and take the first step from x_0 to $x_1 = x_0 + h$ using (4) to solve $y(x_1)$.
2. Use x_1 to solve $y(x_2)$, x_2 to solve $y(x_3)$ and so on till you reach your desired end point x_N.
3. A plot of $\{x_i, y(x_i)\}$ will give you an approximate solution $y = y(x)$ curve, smaller the h better the solution.

Predictor-Corrector method : As the name implies, the method first *predicts* a slope at the end point of the interval and then corrects it. The method predicts the $y(x_n + h)$ using forward Euler (5), say $y^p(x_n + h)$, and then use it to estimate the slope at $x_n + h$ which is $f(y^p(x_n + h), x_n + h)$. Taking the average of the two slopes at the beginning and end of the interval under consideration, $f(y(x_n), x_n)$ and $f(y^p(x_n + h), x_n + h)$ respectively, one obtains the corrected value $y^c(x_n + h)$,

$$\begin{aligned}
y^p(x_n+h) &= y(x_n) + h\, f(y(x_n), x_n) + \mathcal{O}(h^2) \\
y^c(x_n+h) &= y(x_n) + \frac{h}{2}\Big[f(y(x_n), x_n) + f(y^p(x_n+h), x_n+h)\Big] \\
&= y(x_n) + \frac{1}{2}\Big(\kappa_1 + \kappa_2\Big) \qquad (13) \\
\text{where, } \kappa_1 &= h\, f(y(x_n), x_n) \ \text{ and } \ \kappa_2 = h\, f(y^p(x_n+h), x_n+h)
\end{aligned}$$

There are several variants of predictor-corrector method depending on how often the corrector method is applied. For instance, instead of evaluating the function $f(y(x_n), x_n)$ at every step, the predicted function $f(y^p(x_n + h), x_n + h)$ can be used in its place in the next interval.

$$\begin{aligned}
y^p(x_n+h) &= y(x_n) + h\, f(y(x_n), x_n) \\
y^c(x_n+h) &= y(x_n) + \frac{h}{2}\Big[f(y^p(x_n), x_n) + f(y^p(x_n+h), x_n+h)\Big] \qquad (14)
\end{aligned}$$

The algorithm for the simplest Predictor-Corrector method (also known as Heun's method) is almost like forward Euler,

1. Compute the slope at x_n and define $\kappa_1 = h\, f(y(x_n), x_n)$.
2. Calculate the predicted $y^p(x_n + h) = y(x_n) + \kappa_1$.
3. Use y^p compute to $\kappa_2 = h\, f(y^p(x_n + h), x_n + h) = h\, f(y(x_n) + \kappa_1, x_n + h)$.
4. Obtain the corrected $y^c(x_n + h) = y(x_n) + (\kappa_1 + \kappa_2)/2$, which is the equation (13).
5. Go to step 1 and continue iterating through the steps till the end interval x_N is reached.

Runge-Kutta method : It is by far the most popular method of solving ODE and, unless specified otherwise, it is always assumed RK has been used to solve an ODE. Runge-Kutta (RK) methods are based as usual on Taylor expansion but gives in general better algorithm for solutions of an ODE for the same step size and stability. It is, in fact, a generalization of the Euler and PC methods. It involves use of numerical integration for computing $y(x_n + h)$. Consider the following

$$\begin{aligned}
\frac{dy}{dx} &= f(y(x), x) \\
\int_{y_n}^{y_{n+h}} dy &= \int_{x_n}^{x_n+h} f(y(x), x)\, dx \\
y(x_n + h) &= y(x_n) + \int_{x_n}^{x_n+h} f(y(x), x)\, dx \qquad (15)
\end{aligned}$$

To numerically estimate the integral in (15), we can use any of Midpoint, Trapezoidal or Simpson rule for numerical integration. Let us begin with using the Midpoint rule, where

$$\begin{aligned}
\overline{x}_n &= \frac{(x_n + h) + x_n}{2} = x_n + \frac{h}{2} \\
y(x_n + h) &= y(x_n) + h\, f(y(x_n + h/2), x_n + h/2) + \mathcal{O}(h^3) \qquad (16)
\end{aligned}$$

But since $y(x_n + h/2)$ is not known, so the next approximation involves forward Euler's method to compute it,

$$y(x_n + h/2) = y(x_n) + \frac{h}{2}\, f(y(x_n), x_n) \qquad (17)$$

Therefore, *second order Runge-Kutta method (RK2)* goes as,

$$\begin{aligned}
\kappa_1 &= h\, f(y(x_n), x_n) \qquad (18) \\
\kappa_2 &= h\, f(y(x_n) + \kappa_1/2, x_n + h/2) \qquad (19) \\
y(x_n + h) &\approx y(x_n) + \kappa_2 + \mathcal{O}(h^3) \qquad (20)
\end{aligned}$$

So the difference between the previous one-step methods is the addition of an intermediate half-step method (17). The order of error follows from the maximum error bound of the Midpoint method.

Similarly, we can use the Trapezoidal rule but it simply reproduce the Predictor-Corrector formula with the same $\mathcal{O}(h^3)$ error. Integral in the RHS of (15) can be estimated using the Trapezoidal rule,

$$\int_{x_n}^{x_n+h} f(y(x), x)\, dx = \frac{h}{2}\, [f(y(x_n + h), x_n + h) + f(y(x_n), x_n)] \qquad (21)$$

which is essentially eqn. (13).

The next obvious step is using the Simpson rule to develop the *fourth order Runge-Kutta (RK4)* which is thus far the most popular method for solving ODE. Unless stated or asked differently, it will always be assumed that you are using RK4 to solve an ODE. The integral in the expression (15) using Simpson rule can be written as,

$$\begin{aligned}
y(x_n + h) &= y(x_n) + \frac{h}{6}\Big[f(y(x_n), x_n) + 4f(y(x_n + h/2), x_n + h/2) + f(y(x_n + h), x_n + h)\Big] \\
&= y(x_n) + \frac{h}{6}\Big[f(y(x_n), x_n) + 2f(y(x_n + h/2), x_n + h/2) + \\
&\qquad 2f(y(x_n + h/2), x_n + h/2) + f(y(x_n + h), x_n + h)\Big] \qquad (22)
\end{aligned}$$

Basically, we split up the expression for slopes at interval midpoint $f(y(x_n + h/2), x_n + h/2)$ into two – one *predicts* the tangent at the mid interval and the later *corrects* it. Now, we define the following

$$\begin{aligned}
\kappa_1 &= h\, f(y(x_n), x_n) && (23)\\
\kappa_2 &= h\, f(y(x_n) + \kappa_1/2, x_n + h/2) && (24)\\
\kappa_3 &= h\, f(y(x_n) + \kappa_2/2, x_n + h/2) && (25)\\
\kappa_4 &= h\, f(y(x_n) + \kappa_3, x_n + h) && (26)
\end{aligned}$$

which when combined, we finally arrive at the RK4 solution

$$y(x_n + h) = y(x_n) + \frac{1}{6}\Big(\kappa_1 + 2\kappa_2 + 2\kappa_3 + \kappa_4\Big) + \mathcal{O}(h^5) \qquad (27)$$

Once again the $\mathcal{O}(h^5)$ error in RK4 estimate of $y(x_n + h)$ follows from the maximum error bound of Simpson integration rule.

We can generalized the above RK4 solution to n-coupled first order ODE.

$$\left.\begin{aligned}
\tfrac{dy_1}{dx} &= f_1(y_1, y_2, \ldots, y_n, x)\\
\tfrac{dy_2}{dx} &= f_2(y_1, y_2, \ldots, y_n, x)\\
&\vdots\\
\tfrac{dy_n}{dx} &= f_n(y_1, y_2, \ldots, y_n, x)
\end{aligned}\right\} \Rightarrow \quad \frac{d\vec{y}}{dx} = \vec{f}(\vec{y}, x) \qquad (28)$$

where $\vec{y} = (y_1, y_2, \ldots, y_n)$ and $\vec{f} = (f_1, f_2, \ldots, f_n)$. The vector sign simply implies collection of variables. In such case the RK4 equations take the forms,

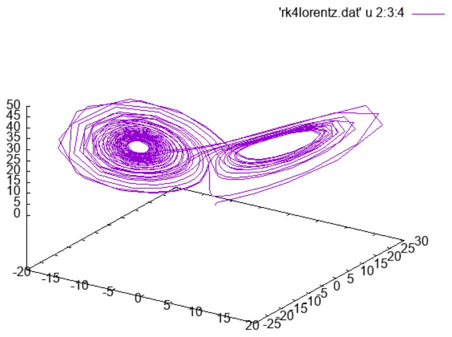
$$\begin{aligned}
\vec{\kappa}_1 &= h\, \vec{f}(\vec{y}_i, x_i)\\
\vec{\kappa}_2 &= h\, \vec{f}(\vec{y}_i + \vec{\kappa}_1/2, x_i + h/2)\\
\vec{\kappa}_3 &= h\, \vec{f}(\vec{y}_i + \vec{\kappa}_2/2, x_i + h/2)\\
\vec{\kappa}_4 &= h\, \vec{f}(\vec{y}_i + \vec{\kappa}_3, x_i + h)\\
\vec{y}_{i+h} &= \vec{y}_i + \frac{1}{6}\Big[\vec{\kappa}_1 + 2\vec{\kappa}_2 + 2\vec{\kappa}_3 + \vec{\kappa}_4\Big] && (29)
\end{aligned}$$

where $\vec{y}_i$ are the values at the i-th interval boundary. The above set of equations (29) have to be read only in terms of components.

There can be just a set of coupled first oder differential equations that are not necessarily obtained by reducing higher order differential equations. For instance, the *Lorentz equations*,

$$\frac{dx}{dt} = \sigma\,(y - x), \quad \frac{dy}{dt} = x\,(\rho - z) \quad \text{and} \quad \frac{dz}{dt} = xy - \beta z \qquad (30)$$

The above equations relate the properties of two dimensional fluid layer uniformly warmed from below and cooled from above. The σ, β and ρ are three parameters whose certain values give rise to chaotic behavior. For $\sigma = 10,\ \rho = 28,\ \beta = 8/3$, the 3-dim plot of the solution shows the famous *Lorentz attractor.*



Symplectic Integrators

When solving Hamiltonian dynamics two important criteria to satisfy are time-reversibility and energy conservation. The RK4 integration scheme does not conserve energy and allow the system to drift significantly over time. Another important problem is *time reversibility* – it is important since it guarantees conservation of energy and certain other conserved quantity. The two most popular integration schemes satisfying these two requirements are less-used *semi-implicit Euler* and widely used *Verlet* or *velocity Verlet* and *Leapfrog integration.* Of these, Leapfrog has the best time reversible property.

Semi-implicit Euler can be applied to a pair of first order differential equations of the form

$$\frac{dx}{dt} = f(t,v) \quad \text{and} \quad \frac{dv}{dt} = g(t,x) \quad \text{with } x(t_0) = x_0,\ v(t_0) = v_0 \tag{31}$$

which are derived from the Hamiltonian $H = T(t,v) + V(t,x)$ of a conservative system. The semi-implicit Euler produces an approximate solution by

$$\begin{aligned} v_{n+1} &= v_n + g\left(t, x_n\right)\,\Delta t + \mathcal{O}(\Delta t) \\ x_{n+1} &= x_n + f\left(t, v_{n+1}\right)\,\Delta t + \mathcal{O}(\Delta t) \end{aligned} \tag{32}$$

Semi-implicit Euler has the same problem with the standard Euler's like $\mathcal{O}(\Delta t)$ error and stability.

Verlet method is used to integrate Newton's equations of motion for trajectory calculations. It has good numerical stability and computational cost is about semi Euler method. Newton's EoM for conservative physical system,

$$\begin{aligned} M\ddot{\mathbf{x}}(t) = -\nabla V\left(\mathbf{x}(t)\right) = \mathbf{F}\left(\mathbf{x}(t)\right) \quad \Rightarrow \\ \ddot{\mathbf{x}}(t) = \mathbf{A}\left(\mathbf{x}(t)\right), \quad \text{where,} \quad \mathbf{x}(t_0) = x_0,\ \dot{\mathbf{x}}(t_0) = v_0,\ \mathbf{A}(\mathbf{x}(t)) = \mathbf{F}(\mathbf{x}(t))/M \end{aligned} \tag{33}$$

If we take the approximate numerical solution $\mathbf{x}_n \approx \mathbf{x}(t_n)$ at the time $t_n = t_0 + n\Delta t$ with step size $\Delta t > 0$, then

$$\begin{aligned} \frac{\Delta^2 \mathbf{x}_n}{\Delta t^2} = \frac{\mathbf{x}_{n+1} + \mathbf{x}_{n-1} - 2\mathbf{x}_n}{\Delta t^2} = \mathbf{A}(\mathbf{x}_n) \\ \mathbf{x}_{n+1} = 2\mathbf{x}_n - \mathbf{x}_{n-1} + \mathbf{a}_n \Delta t^2, \quad \text{where} \quad \mathbf{a}_n = \mathbf{A}(\mathbf{x}_n) \end{aligned} \tag{34}$$

Algorithm consists of the following steps

1. At time t_0, the integration begins with

$$\mathbf{x}_1 = \mathbf{x}_0 + \mathbf{v}_0 \Delta t + \frac{1}{2}\mathbf{A}(x_0)\Delta t^2 \tag{35}$$

2. Then subsequently

$$\mathbf{x}_{n+1} = 2\mathbf{x}_n - \mathbf{x}_{n-1} + \mathbf{A}(x_n)\Delta t^2 \tag{36}$$

The velocities do not appear explicitly in basic Verlet equation but they may be necessary for calculations like kinetic energy. The reason being velocity at t cannot be calculated unless the position is known at $t + \Delta t$,

$$\mathbf{v}(t) = \frac{\mathbf{x}(t+\Delta t) - \mathbf{x}(t-\Delta t)}{2\Delta t} + \mathcal{O}\left(\Delta t^2\right) \tag{37}$$

Thus velocity term is one step behind the position. We can, however, avoid this problem at the cost of accuracy,

$$\mathbf{v}(t) = \frac{\mathbf{x}(t+\Delta t) - \mathbf{x}(t)}{\Delta t} + \mathcal{O}\left(\Delta t\right) \tag{38}$$

But a more commonly used algorithm is *velocity Verlet*, the standard implementation of which is,

1. Calculate the velocity at half-time step $\Delta t/2$

$$\mathbf{v}(t + \Delta t/2) = \mathbf{x}(t) + \mathbf{v}(t)\Delta t + \frac{1}{2}\mathbf{a}(t)\Delta t \tag{39}$$

where the acceleration $\mathbf{a}(t)$ is defined in eqn. (34).

2. Next, calculate the position at $t + \Delta t$,

$$\mathbf{x}(t+\Delta t) = \mathbf{x}(t) + \mathbf{v}(t + \Delta t/2)\Delta t \tag{40}$$

3. Derive $\mathbf{a}(t+\Delta t)$ from potential *i.e.* force as defined in eqn. (33)

$$\mathbf{a}(t+\Delta t) = \mathbf{A}(x(t+\Delta t)) = \mathbf{F}(x(t+\Delta t))/M \tag{41}$$

assuming $\mathbf{a}(t)$ depends only on position $\mathbf{x}(t)$ and not on velocity.

4. Finally calculate velocity at the end of the interval $v(t+\Delta t)$,

$$\mathbf{v}(t+\Delta t) = \mathbf{v}(t) + \frac{1}{2}\Big(\mathbf{a}(t) + \mathbf{a}(t+\Delta t)\Big) \tag{42}$$

Velocity Verlet is thus one order magnitude better than semi-implicit Euler.

Leapfrog integration is a second order method like Verlet and in contrast to Euler integration, which is only first order, yet requires the same number of function evaluations per step. It is also stable against oscillatory motion which is why it is often the integrator of choice in *molecular dynamics* simulation. Leapfrog integration is equivalent to updating positions $x(t)$ and velocities $v(t) = \dot{x}(t)$ at different interleaved time points, staggered in such a way that they leapfrog over each other. The Hamilton equations of motion are,

$$\dot{x} = \frac{\partial H(x,\pi)}{\partial \pi} \quad \text{and} \quad \dot{\pi} = -\frac{\partial H(x,\pi)}{\partial x} \tag{43}$$

Taylor expand $x(t+\Delta t)$ and $\pi(t+\Delta t)$ (using $\ddot{x}(t) = \dot{\pi}(t)$),

$$\begin{aligned} x(t+\Delta t) &= x(t) + \dot{x}(t)\,\Delta t + \frac{\Delta t^2}{2}\ddot{x}(t) + \mathcal{O}(\Delta t^3) \;=\; x(t) + \pi(t)\,\Delta t + \dot{\pi}\,\frac{\Delta t^2}{2} \\ &= x(t) + \left[\pi(t) - \frac{\partial H}{\partial x}\frac{\Delta t}{2}\right]\Delta t \;\equiv\; x(t) + \pi\left(\frac{\Delta t}{2}\right)\Delta t \end{aligned} \tag{44}$$

$$\pi(t+\Delta t) = \pi(t) + \dot{\pi}(t)\,\Delta t + \frac{\Delta t^2}{2}\ddot{\pi}(t) + \mathcal{O}(\Delta t^3) \tag{45}$$

$$\text{where,}\quad \dot{x} = \pi, \quad \text{and} \quad \dot{\pi} = -\frac{\partial H}{\partial x} = -\frac{\partial V}{\partial x} = F \tag{46}$$

where F is the force in eqn. (33). The $\ddot{\pi}$ is estimated to leading order as,

$$\ddot{\pi}(t) = \frac{\partial}{\partial t}\frac{\partial H}{\partial x} = \frac{1}{\Delta t}\left(\frac{\partial H}{\partial x(t+\Delta t)} - \frac{\partial H}{\partial x(t)}\right) + \mathcal{O}(\Delta t) \tag{47}$$

Using the above expression (47) in eqns. (44) and (45), we get

$$\begin{aligned} x(t+\Delta t) &= x(t) + \Delta t\left(\pi(t) - \frac{\Delta t}{2}\frac{\partial H}{\partial x(t)}\right) + \mathcal{O}(\Delta t^3) \\ \left(\pi(t+\Delta t) - \frac{\Delta t}{2}\frac{\partial H}{\partial x(t+\Delta t)}\right) &= \left(\pi(t) - \frac{\Delta t}{2}\frac{\partial H}{\partial x(t)}\right) - \frac{\partial H}{\partial x(t+\Delta t)}\Delta t + \mathcal{O}(\Delta t^3) \end{aligned} \tag{48}$$

The quantities in parentheses are the momentum evaluated at the midpoint of the time intervals. The discretised equations of motion (44, 45) thus take the form,

$$\begin{aligned} x(t+\Delta t) &= x(t) + \pi\left(t + \frac{\Delta t}{2}\right)\Delta t \\ \pi\left(t + \frac{3}{2}\Delta t\right) &= \pi\left(t + \frac{\Delta}{2}\right) - \frac{\partial H}{\partial x(t+\Delta t)}\Delta t = \pi\left(t + \frac{\Delta}{2}\right) + F(t+\Delta t)\Delta t \end{aligned} \tag{49}$$

This is the *Leapfrog* scheme to integrate the Hamilton equations of motion.

1. The integration begins with taking a half-step in time for momentum

$$\pi(t_0 + \Delta t/2) = \pi(t_0) + F(t_0)\,(\Delta t/2) \tag{50}$$

2. Subsequently, full time step in both position and momentum for the length of the trajectory $\tau - \Delta t$ with one interval less.

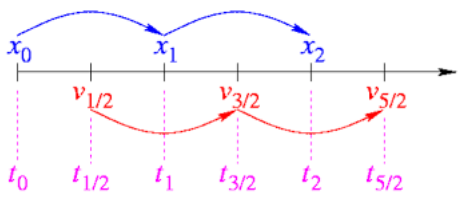
$$\begin{aligned} x(t+\Delta t) &= x(t) + \pi(t+\Delta t/2)\,\Delta t \\ \pi(t+3\Delta t/2) &= \pi(t+\Delta t/2) + F(t+\Delta t)\,\Delta t \end{aligned} \tag{51}$$

3. One final full step for x takes it to the end of the trajectory.

$$x(\tau) = x(\tau - \Delta t) + \pi(\tau - \Delta t/2)\,\Delta t \tag{52}$$

4. One final half-step for π takes it to the end of the trajectory.

$$\pi(\tau) = \pi(\tau - \Delta t/2) + F(\tau - \Delta t/2)\,(\Delta t/2) \tag{53}$$



Boundary value problem

Many problems in physics are, in fact, boundary value problems. For instance the Laplace equation in electrostatics or, more famously, Schrödinger equations. In boundary value problems, we have conditions specified at two different space (and/or time) points. We can have either

Dirichlet condition	:	$y(x_0) = Y_0$ and $y(x_N) = Y_n$
Neumann condition	:	$y'(x_0) = Y'_0$ and $y'(x_N) = Y'_N$
Mixed condition	:	$y(x_0) = Y_0$ and $y'(x_N) = Y'_N$ or the other way round.

Here two methods will be discussed, the most commonly used *shooting method* and the one employed in solving partial differential equations.

Shooting method : The strategy to numerically solve boundary value problem is to reduce the second order ODE to a system of first order initial value ODE.

$$\frac{d^2y}{dx^2} = f(x, y, y') \quad \text{where } a \le x \le b \text{ and } y(a) = \alpha,\ y(b) = \beta \tag{54}$$

$$\Rightarrow \quad \frac{dy}{dx} = z \quad \text{with } y(a) = \alpha \tag{55}$$

$$\frac{dz}{dx} = \frac{d^2y}{dx^2} = f(x, y, z) \quad \text{with } z(a) = \zeta_h \tag{56}$$

where $z(a) = \zeta_h$ *i.e.* slope at $x = a$ is a guess. The next step involves solving (55) and (56) by Euler/RK2/RK4 using the initial values $y(a) = \alpha$ and $z(a) = \zeta_h$. The solution obtained at the end point x_N is compared with the boundary condition $y(b) = \beta$. If $y_{\zeta_h}(b) = \beta$ within tolerance then the ODE is solved.

But suppose it is not and $y_{\zeta_h}(b) > \beta$ *i.e.* you shoot higher. Change the guess initial value to ζ_l and the system is solved again as above. The choice should be such (assuming it does not land bang on the solution) that $y_{\zeta_l}(b) < \beta$ *i.e.* you shoot lower. So the target is some where in between. The choice of the slope would be such that the actual slope at initial point is $\zeta_l < z(a) < \zeta_h$.
Use Lagrange's interpolation formula to choose the next $z(a) = \zeta$, which is in between ζ_l and ζ_h,

$$\frac{\zeta - \zeta_l}{\zeta_h - \zeta_l} = \frac{y(b) - y_{\zeta_l}(b)}{y_{\zeta_h}(b) - y_{\zeta_l}(b)} \quad \Rightarrow \quad \zeta = \zeta_l + \frac{\zeta_h - \zeta_l}{y_{\zeta_h}(b) - y_{\zeta_l}(b)} \Big(y(b) - y_{\zeta_l}(b) \Big) \tag{57}$$

The $z(a) = \zeta$ is our new guess and chances are this choice will lead us to the solution of the ODE *i.e.* $y_\zeta(b) \approx \beta$. If not, go through the above procedure until $y_\zeta(b)$ converges to β reasonably well.

Let us study the following example,

$$\frac{d^2y}{dx^2} = 2y \quad \text{with } y(x = 0.0) = \alpha = 1.2,\ \ y(x = 1.0) = \beta = 0.9,\ \ h = 0.02. \tag{58}$$

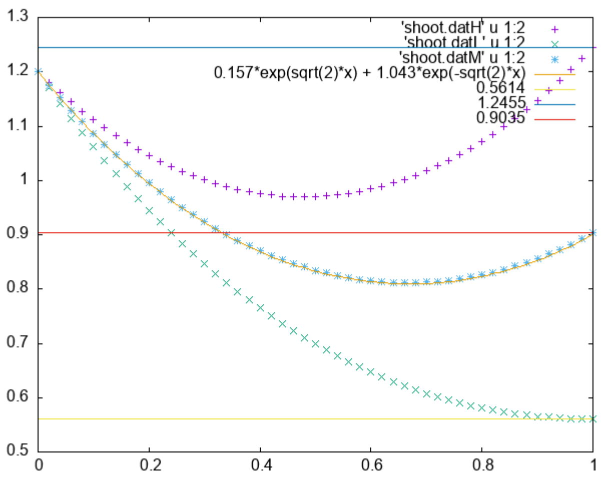
The solution of the above ODE (58) is

$$y(x) = c_1\, e^{\sqrt{2}\,x} + C_2\, e^{-\sqrt{2}\,x}, \quad \text{where } C_1 = 0.157,\ C_2 = 1.043 \tag{59}$$

Suppose we start with $z(x = 0.0) = -1.5$, using RK4 we obtain $y(x = 1.0) = 0.5614$ which is less than $\beta = 0.9$. So $\zeta_l = -1.5$ and $y_{\zeta_l}(1.0) = 0.5614$. Next we try (presently the choice is deliberate) $z(x = 0.0) = -1.0 = \zeta_h$ which yields $y_{\zeta_h}(x = 1.0) = 1.2455 > \beta = 0.9$. Using Lagrange's linear interpolating formula (57) we get,

$$\zeta = -1.5 + \frac{-1.0 - (-1.5)}{1.2455 - 0.5614} \times \Big(0.9 - 0.5614 \Big) = -1.2525 \tag{60}$$

Using $z(x = 0.0) = -1.25$, we obtain $y_\zeta(x = 1.0) = 0.9035 \approx \beta = 0.9$. Thus we have solved our ODE in (58). A graphical view of the process is shown below.



In practice, you may not be so lucky to guess ζ_h and ζ_l appropriately, so you may end up going through multiple iterations of the above method.

Finite element method : This method depends on numerical approximation of derivatives, which obviously comes from Taylor series.

$$y(x+h) = y(x) + hy'(x) + \frac{h^2}{2!}f''(x) + \frac{h^3}{3!}y'''(x) + \cdots$$

$$y(x-h) = y(x) - hy'(x) + \frac{h^2}{2!}f''(x) - \frac{h^3}{3!}y'''(x) + \cdots \tag{61}$$

$$\Rightarrow \quad y'(x) = \frac{y(x+h) - y(x-h)}{2h} + \mathcal{O}(h^2)$$

$$y''(x) = \frac{y(x+h) + y(x-h) - 2y(x)}{h^2} + \mathcal{O}(h^2) \tag{62}$$

Apart from these, a second order method is required for x on the boundaries. For a boundary point on the left and right require additional Taylor series,

$$y(x+2h) = y(x) + 2hy'(x) + 2h^2y''(x) + \frac{4h^3}{3}y'''(x) + \cdots$$

$$y(x-2h) = y(x) - 2hy'(x) + 2h^2y''(x) - \frac{4h^3}{3}y'''(x) + \cdots \tag{63}$$

Combining the Taylor series for $y(x \pm h)$ and $y(x \pm 2h)$ in (61, 63), we get

$$y'(x) = \frac{-3y(x) + 4y(x+h) - y(x+2h)}{2h} + \mathcal{O}(h^2)$$

$$y'(x) = \frac{3y(x) - 4y(x-h) + y(x-2h)}{2h} + \mathcal{O}(h^2) \tag{64}$$

$$y''(x) = \frac{16y(x+h) + 16y(x-h) - 30y(x) - f(x-2h) - f(x+2h)}{12h^2} + \mathcal{O}(h^5) \tag{65}$$

The expression (65) follows from eliminating $\mathcal{O}(h^4)$ terms from $y(x \pm 2h)$. Aside the x at the boundaries, the equation (54) in the interior grid points gets the following form

$$y(x+h) - 2y(x) + y(x-h) = h^2\, f(x,y)) \tag{66}$$

with the boundary conditions $y(a) = \alpha$ and $y(b) = y(a + Nh) = \beta$ where $Nh = b - a$. We therefore have a linear system of equations to solve. A new notation will be adopted below to make consecutive steps clear. The first and the N-th equations are given by

$$\begin{aligned}
& y(a) + y(a + 2h) - 2y(a + h) = h^2 f(a + h) \\
\rightarrow \quad & y(a + 2h) - 2y(a + h) = h^2 f(a + h) - \alpha \\
\Rightarrow \quad & y_2 - 2y_1 = h^2 f_1 - \alpha \\
\text{or,} \quad & 2y_1 - y_2 = -h^2 f_1 + \alpha
\end{aligned} \tag{67}$$

where, $y_i = y(a + ih),\ i = 1, 2, \ldots, N - 1$. In this notation, the final step is

$$-y_{N-2} + 2y_{N-1} = -h^2 f_{N-1} + \beta \tag{68}$$

The subsequent intermediate equations are,

$$\begin{aligned}
-y_1 + 2y_2 - y_3 &= -h^2 f_2 \\
-y_2 + 2y_3 - y_4 &= -h^2 f_3 \\
\vdots &= \vdots
\end{aligned} \tag{69}$$

The corresponding matrix form is

$$\begin{pmatrix} 2 & -1 & 0 & 0 & \cdots & & 0 \\ -1 & 2 & -1 & 0 & \cdots & & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \cdots & -1 & 2 & -1 \\ 0 & \cdots & \cdots & & & -1 & 2 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{N-2} \\ y_{N-1} \end{pmatrix} = \begin{pmatrix} -h^2 f_1 + \alpha \\ -h^2 f_2 \\ \vdots \\ -h^2 f_{N-2} \\ -h^2 f_{N-1} + \beta \end{pmatrix} \tag{70}$$

The linear system is *tridiagonal*, *symmetric* and *positive definite*. The positive definite property is only ensured if we use $y'' = -f$ form. It requires $\mathcal{O}(N)$ operations to solve. Standard method to solve for such a tridiagonal system $\mathbf{A} \cdot \mathbf{y} = \mathbf{f}$ is Cholesky decomposition $\mathbf{A} = \mathbf{L}\,\mathbf{L}^T$ subsequently using *forward* and *backward substitution*.

Partial Differential Equation

As we all know that a bunch of equations in physics are partial differential equations – Maxwell equations, Laplace and Poisson equations, wave equations, Schrödinger equation, diffusion equation and so on. A general linear partial differential equation in 2-dimension *i.e.* second-order in two independent variables reads,

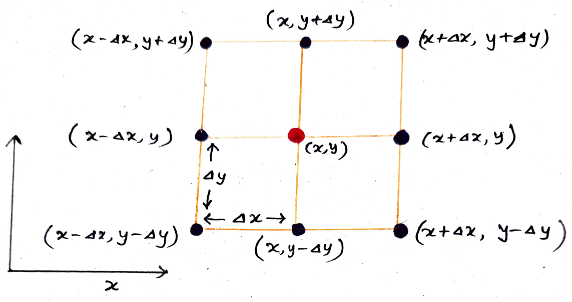
$$A(x, y)\,\frac{\partial^2 \phi}{\partial x^2} + B(x, y)\,\frac{\partial^2 \phi}{\partial x \partial y} + C(x, y)\,\frac{\partial^2 \phi}{\partial y^2} = F\left(x, y, \phi, \frac{\partial \phi}{\partial x}, \frac{\partial \phi}{\partial y}\right) \tag{71}$$

Depending on the coefficients A, B, C, the partial differential equations are classified as

1. *elliptic* if $B^2 - 4AC < 0$
2. *parabolic* if $B^2 - 4AC = 0$
3. *hyperbolic* if $B^2 - 4AC > 0$

For instance, $B = C = 0$ is the $1+1$ dimensional diffusion equation which corresponds to parabolic differential equation, whereas $B = 0$ and $AC < 0$ is $2+1$ dimensional wave equation is an elliptic partial differential equation. `What would be the class(es) of Maxwell's equations?` Note, that we will often denote $\partial^2 u/\partial x^2 = u_{xx}$ etc.

One of the most common method employed to solve PDE is *finite difference* which converts ODE or PDE into a system of linear equations. Subsequently, these linear equations can be solved iteratively. It is further divided into two categories – (i) explicit methods and (ii) implicit methods. For simplicity consider only two degrees of freedom, x and y, and the function being $u(x, y)$



1. *Forward difference* is involved in the explicit methods
$$\left.\frac{\partial u}{\partial x}\right|_y = \frac{u(x+\Delta x, y) - u(x, y)}{\Delta x} + \mathcal{O}(\Delta x) \quad \text{and} \quad \left.\frac{\partial u}{\partial y}\right|_x = \frac{u(x, y+\Delta y) - u(x, y)}{\Delta y} + \mathcal{O}(\Delta y) \quad (72)$$

2. *Backward difference* is involved in the implicit methods
$$\left.\frac{\partial u}{\partial x}\right|_y = \frac{u(x, y) - u(x-\Delta x, y)}{\Delta x} + \mathcal{O}(\Delta x) \quad \text{and} \quad \left.\frac{\partial u}{\partial y}\right|_x = \frac{u(x, y) - u(x, y-\Delta y)}{\Delta y} + \mathcal{O}(\Delta y) \quad (73)$$

3. *Symmetric difference* is $\mathcal{O}(\Delta x^2, y^2)$ corrected,
$$\left.\frac{\partial u}{\partial x}\right|_y = \frac{u(x+\Delta x, y) - u(x-\Delta x, y)}{\Delta x} + \mathcal{O}(\Delta x^2) \quad \text{and similarly for } y \qquad (74)$$
Although it appears preferable, it has surprising instability problem.

4. Second order partial derivative is rather straight forward and is, by definition, $\mathcal{O}(\Delta x^2, y^2)$ corrected. For u_{xx},
$$\left.\frac{\partial^2 u}{\partial x^2}\right|_y = \frac{u(x+\Delta x, y) + u(x-\Delta x, y) - 2u(x, y)}{\Delta x^2} + \mathcal{O}(\Delta x^2) \qquad (75)$$

The partial derivatives, both first and second order, defined above require the point itself and at most two nearest neighbor points. It forms a 3-point *stencil.* We can also have 5-point *stencil* made up of the point itself together with its four nearest neighbors, (the second coordinate y is not shown explicitly in the first two expressions for brevity)

$$u_x = \frac{-u(x+2\Delta x) + 8u(x+\Delta x) - 8u(x-\Delta x) + u(x-2\Delta x)}{12\Delta x} + \mathcal{O}(\Delta x^4) \qquad (76)$$

$$u_{xx} = \frac{-u(x+2\Delta x) + 16u(x+\Delta x) - 30u(x) + 16u(x-\Delta x) - u(x-2\Delta x)}{12\Delta x^2} + \mathcal{O}(\Delta x^5) \qquad (77)$$

$$u_{xx} + u_{yy} = \frac{u(x+\Delta x, y) + u(x-\Delta x, y) + u(x, y+\Delta y) + u(x, y-\Delta y) - 4u(x, y)}{\Delta x^2} + \mathcal{O}(\Delta x^2) \qquad (78)$$

This 5-point stencil is often used to approximate Laplacian of a function of two variables.

First consider the explicit scheme in $1+1$-dimensional diffusion or heat equation,

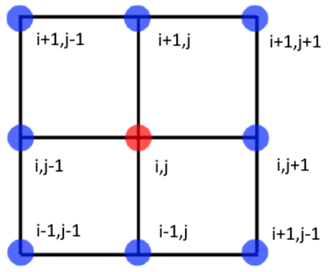
$$\frac{\partial^2 u(x,t)}{\partial x^2} = \frac{\partial u(x,t)}{\partial t} \quad \equiv \quad u_{xx} = u_t \tag{79}$$

with the initial conditions at $t=0$ and boundary conditions at later time $t \geq 0$,

$$\begin{aligned} u(x,0) &= g(x) \quad \text{for } \; 0 < x < L & (80)\\ u(0,t) &= a(t) \quad \text{for } \; t \geq 0 & (81)\\ u(L,t) &= b(t) \quad \text{for } \; t \geq 0 & (82)\end{aligned}$$

In discretised setup, let Δx and Δt be the step lengths in space x-direction and time t-direction respectively. The position after i steps and time at j step are given by

$$\begin{cases} x_i = i\Delta x & 0 \leq i \leq n+1 \\ t_j = j\Delta t & j \geq 0 \end{cases} \tag{83}$$



The discrete derivatives are written in standard way,

$$u_t \approx \frac{u(x,t+\Delta t)-u(x,t)}{\Delta t} \equiv \frac{u(x_i,t_j+\Delta t)-u(x_i,t_j)}{\Delta t} = \frac{u_{i,j+1}-u_{i,j}}{\Delta t} \tag{84}$$

$$\begin{aligned} u_{xx} &\approx \frac{u(x+\Delta x,t)+u(x-\Delta x,t)-2u(x,t)}{\Delta x^2} \\ &\equiv \frac{u(x_i+\Delta x,t_j)+u(x_i-\Delta x,t_j)-2u(x_i,t_j)}{\Delta x^2} = \frac{u_{i+1,j}+u_{i-1,j}-2u_{i,j}}{\Delta x^2} & (85)\end{aligned}$$

Defining $\alpha = \Delta t/\Delta x^2$ and putting back everything in (79) results in the explicit scheme

$$\begin{aligned} \frac{u_{i,j+1}-u_{i,j}}{\Delta t} &= \frac{u_{i+1,j}+u_{i-1,j}-2u_{i,j}}{\Delta x^2} \\ u_{i,j+1} &= = \alpha\,(u_{i+1,j}+u_{i-1,j}) + (1-2\alpha)\,u_{i,j} & (86)\end{aligned}$$

Since we have all the discretised initial values (80), $u_{i,0} = g(x_i)$, after one time step we get $u_{i,1}$,

$$u_{i,1} = \alpha\,(u_{i+1,0}+u_{i-1,0}) + (1-2\alpha)\,u_{i,0} = \alpha\,(g(x_{i+1},0)+g(x_{i-1,0})) + (1-2\alpha)\,g(x_{i,0}) \tag{87}$$

For simplicity and without loss of generality, consider $a(t) = b(t) = 0$ which implies $u_{0,j} = u_{L=n+1,j} = 0$ where we have divided the interval $[0,L]$ in n parts. Then a vector V_j at the time $t_j = j\Delta t$ is defined as,

$$V_j = \begin{pmatrix} u_{1,j} \\ u_{2,j} \\ \vdots \\ u_{n,j} \end{pmatrix} \tag{88}$$

This amounts to solving V_{j+1} as,

$$V_{j+1} = \mathbf{A}\, V_j \quad \text{where } \mathbf{A} = \begin{pmatrix} 1-2\alpha & \alpha & 0 & 0 & \dots \\ \alpha & 1-2\alpha & \alpha & 0 & \cdots \\ 0 & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \alpha & 1-2\alpha \end{pmatrix} \tag{89}$$

which in turn implies that

$$V_{j+1} = \mathbf{A} V_j = \mathbf{A}^2 V_{j-1} = \cdots = \mathbf{A}^{j+1} V_0 \tag{90}$$

where V_0 is the initial vector at time $t = 0$ given by the initial values $g(x)$. Therefore, the steps involved are pretty straight forward

1. Initialize $u_{i,0} = g(x_i)$ for all x_i sites at time $j = 0$ with whatever the function $g(x)$ is and construct V_0. Remember that $u_{0,0} = u_{n+1,0} = 0$ for $a(t) = b(t) = 0$. Otherwise, these two will also enter the vector V_0.

2. Evolve the system in time using matrix-vector multiplication in (89) till the end of time, say $u_{i,T}$. This is your solution.

Although the explicit scheme is easy to implement, it has a weak stability condition given by $\Delta t/\Delta x^2 \leq 0.5$ (which will not be discussed in this course). However, very similar to the problem of explicit forward Euler, we can mostly overcome the stability problem using implicit scheme *i.e.* by taking backward derivative in time direction keeping the second derivative as in (85) unchanged,

$$\begin{aligned} u_t &\approx \frac{u_{i,j} - u_{i,j-1}}{\Delta t} && (91) \\ u_{i,j-1} &= -\alpha\,(u_{i+1,j} + u_{i-1,j}) + (1+2\alpha)\, u_{i,j} && (92) \end{aligned}$$

This leads to redefining the matrix $\mathbf{A}$ and subsequent iterative solution steps,

$$\begin{aligned} V_{j-1} &= \mathbf{A}\, V_j \quad \text{where } \mathbf{A} = \begin{pmatrix} 1+2\alpha & -\alpha & 0 & 0 & \dots \\ -\alpha & 1+2\alpha & -\alpha & 0 & \cdots \\ 0 & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & -\alpha & 1+2\alpha \end{pmatrix} && (93) \\ V_j &= \mathbf{A}^{-1} V_{j-1} = \mathbf{A}^{-2} V_{j-2} = \cdots = \mathbf{A}^{-j} V_0 && (94) \end{aligned}$$

The above evolution equation has much better stability property. But what about taking symmetric derivative instead of asymmetric as in (84) or (91). *Du Fort* and *Frankel* method uses

$$u_t \approx \frac{u_{i,j+1} - u_{i,j-1}}{2\Delta t} \tag{95}$$

This reduces the diffusion equation to

$$\begin{aligned} \frac{u_{i,j+1} - u_{i,j-1}}{2\Delta t} &= \frac{u_{i+1,j} + u_{i-1,j} - 2u_{i,j}}{\Delta x^2} \\ u_{i,j+1} &= = u_{i,j-1} + 2\alpha\,(u_{i+1,j} + u_{i-1,j} - 2u_{i,j}) && (96) \end{aligned}$$

Further, $u_{i,j}$ is replaced by the average $(u_{i,j+1} + u_{i,j-1})/2$ which yields,

$$\begin{aligned} u_{i,j+1} &= = u_{i,j-1} + 2\alpha\,(u_{i+1,j} + u_{i-1,j} - [u_{i,j+1} + u_{i,j-1}]) \\ u_{i,j+1} &= \frac{1-2\alpha}{1+2\alpha} u_{i,j-1} + \frac{2\alpha}{1+2\alpha}\,(u_{i+1,j} + u_{i-1,j}) && (97) \end{aligned}$$

The one complication it has is the appearance of the term $u_{i,j-1}$. For $t = 0$ *i.e.* $j = 0$, the field $u_{i,-1}$ has to be specified possibly using one of the boundary conditions. Otherwise, there will two time step iterations, one at $j = 0$ and the other for $j \neq 0$. Besides, Du Fort method does not provide any additional numerical advantages over implicit method of backward derivative.

There is yet another scheme called *Crank-Nicolson*, where the u_{xx} is approximated by the average of u_{xx} at two different time slices $j-1$ and j (or, alternatively j and $j+1$),

$$u_{xx} = \frac{1}{2\Delta x^2}\Big[(u_{i+1,j-1} + u_{i-1,j-1} - 2u_{i,j-1}) + (u_{i+1,j} + u_{i-1,j} - 2u_{i,j}) \Big] \tag{98}$$

Substituting this in he heat equation $u_{xx} = u_t$ along with backward discretised u_t gives us

$$\begin{aligned}
& \frac{u_{i,j} - u_{i,j-1}}{\Delta t} = \frac{1}{2\Delta x^2}\Big[(u_{i+1,j-1} + u_{i-1,j-1} - 2u_{i,j-1}) + (u_{i+1,j} + u_{i-1,j} - 2u_{i,j}) \Big] \\
& -\alpha\,(u_{i+1,j} + u_{i-1,j}) + 2(1+\alpha)u_{i,j} = \alpha\,(u_{i+1,j-1} + u_{i-1,j-1}) + 2(1-\alpha)u_{i,j-1} \qquad (99) \\
\Rightarrow \quad & \Big(2\mathbb{I} + \alpha\mathbf{B}\Big)\, V_j = \Big(2\mathbb{I} - \alpha\mathbf{B}\Big)\, V_{j-1} \\
\text{or,} \quad & Vj = \Big(2\mathbb{I} + \alpha\mathbf{B}\Big)^{-1}\Big(2\mathbb{I} - \alpha\mathbf{B}\Big)\, V_{j-1} \qquad (100)
\end{aligned}$$

$$\text{where,} \qquad \mathbf{B} = \begin{pmatrix} 2 & -1 & 0 & 0 & \cdots \\ -1 & 2 & -1 & 0 & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & -1 & 2 \end{pmatrix} \tag{101}$$

The Crank-Nicolson method is, therefore, a combination of explicit and implicit schemes. The method is known to converge for all values of α and, hence, very useful where coarse graining in any of the space or time direction is needed.

Laplace's and Poisson's equations : In 2-dimension the Laplace's equation reads,

$$\nabla^2 u(x,y) = u_{xx} + u_{yy} = 0 \tag{102}$$

with boundary conditions $u(x,y) = g(x,y)$,

$$\begin{array}{lcl} u_{i,0} = g_{i,0} & \text{and} & u_{i,L} = g_{i,L} \\ u_{0,j} = g_{0,j} & \text{and} & u_{L,j} = g_{L,j} \end{array} \tag{103}$$

We have to choose square grid of length $L \times L$ with equally many steps in both direction,

$$h = \Delta x = \Delta y = \frac{L}{n+1} \tag{104}$$

The discretised versions of u_{xx} and u_{yy} are

$$\begin{aligned}
u_{xx} &\approx \frac{u(x+h,y) + u(x-h,y) - 2u(x,y)}{h^2} = \frac{u_{i+1,j} + u_{i-1,j} + 2u_{i,j}}{h^2} && (105) \\
u_{yy} &\approx \frac{u(x,y+h) + u(x,y-h) - 2u(x,y)}{h^2} = \frac{u_{i,j+1} + u_{i,j-1} + 2u_{i,j}}{h^2} && (106) \\
0 &= \frac{u_{i+1,j} + u_{i-1,j} + 2u_{i,j}}{h^2} + \frac{u_{i,j+1} + u_{i,j-1} + 2u_{i,j}}{h^2} \\
\Rightarrow \quad u_{i,j} &= \frac{1}{4}\Big[u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} \Big] && (107)
\end{aligned}$$

where the solution $u_{i,j}$ is basically the average of nearest neighbors. It is called the *standard 5-point calculation*. So the numerical solution of Laplace's equation is even more simpler than heat equation. Solving Poisson's equation involves only small complication,

$$\nabla^2 u(x,y) = -\rho(x,y) \quad \Rightarrow \quad u_{i,j} = \frac{1}{4}\Big[u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}\Big] + \frac{h^2}{4}\,\rho_{i,j} \tag{108}$$

Either of Laplace's or Poisson's cannot be solved by starting at one boundary and work your way across the grid to the other because the scheme requires knowledge of u at all of the neighboring points. The evolution matrix $\mathbf{A}$ is $L^2 \times L^2$ sparse matrix, hence the solving calls for iterative schemes like *Jacobi* or *Gauss-Seidel* methods.

As an example take $L_x = L_y = L = 3$, the boundaries being at $(i,j) = (0,0)$ and $(L_x+1, L_y+1) \equiv (4,4)$. Away from the boundaries, the solutions are required for 9-points $(i,j) = (1,1)$, $(1,2), \ldots (3,3)$. The vector form of the solutions is,

$$\mathbf{V}^T = \Big(u_{11}\; u_{12}\; u_{13}\; u_{21}\; u_{22}\; u_{23}\; u_{31}\; u_{32}\; u_{33}\Big) \tag{109}$$

The matrix system of $\mathbf{V}$ using eqn. (108),

$$\begin{aligned}
i,j = 1,1 \quad &: \quad 4u_{11} - u_{21} - u_{01} - u_{12} - u_{10} = h^2\rho_{11} \\
& \qquad 4u_{11} - u_{12} - u_{21} - (u_{01} + u_{10}) = h^2\rho_{11} \\
i,j = 1,2 \quad &: \quad 4u_{12} - u_{21} - u_{02} - u_{13} - u_{11} = h^2\rho_{12} \\
& \qquad -u_{11} + 4u_{12} - u_{13} - u_{21} - (u_{02}) = h^2\rho_{12} \\
\vdots \quad & \quad \vdots \\
i,i = 2,2 \quad &: \quad 4u_{22} - u_{32} - u_{12} - u_{23} - u_{21} = h^2\rho_{22} \\
& \qquad -u_{12} - u_{21} + 4u_{22} - u_{23} - u_{32} = h^2\rho_{22} \\
\vdots \quad & \quad \vdots
\end{aligned} \tag{110}$$

$$\mathbf{A} = \begin{pmatrix}
4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
-1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\
-1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\
0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\
0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\
0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4
\end{pmatrix} \tag{111}$$

$$\mathbf{f} = \begin{pmatrix}
h^2\rho_{11} + g_{01} + g_{10} \\
h^2\rho_{12} + g_{02} \\
h^2\rho_{13} + g_{14} + g_{03} \\
h^2\rho_{21} + g_{20} \\
h^2\rho_{22} \\
h^2\rho_{23} + g_{24} \\
h^2\rho_{31} + g_{30} + g_{41} \\
h^2\rho_{32} + g_{42} \\
h^2\rho_{33} + g_{34} + g_{43}
\end{pmatrix} \tag{112}$$

Wave equation : The $1+1$ dimensional wave equation is

$$\frac{\partial^2 u}{\partial x^2} = \frac{\partial^2 u}{\partial t^2} \quad \Rightarrow \quad u_{xx} = u_{tt} \tag{113}$$

Choosing different spacing in space and time direction, Δx and Δt, as in diffusion equation, resulting in

$$u_{i,j+1} = 2u_{i,j} - u_{i,j-1} + \frac{\Delta t^2}{\Delta x^2}\Big[u_{i+1,j} + u_{i-1,j} - 2u_{i,j}\Big] \tag{114}$$

Once again solving this is straight forward. A complication will arise if one of the boundary condition is Neumann type *i.e.* say

$$\left.\frac{\partial u}{\partial t}\right|_{t=0} = 0 \quad \text{for} \;\; x \in [0, L] \tag{115}$$

This will modify the equation for the first time step

$$\begin{aligned} u_t \quad &\approx \quad \frac{u(x_i, t_j + \Delta t) - u(x_i, t_j - \Delta t)}{2\Delta t} \quad \Rightarrow \quad u_t = \frac{u_{i,j+1} - u_{i,j-1}}{2\Delta t} = 0 \\ \Rightarrow \quad u_{i,1} \quad &= \quad u_{i,0} + \frac{\Delta t^2}{2\Delta x^2}\Big[u_{i+1,0} + u_{i-1,0} - 2u_{i,0}\Big] \end{aligned} \tag{116}$$

In the above step, we have used the fact that at $t = 0, u_t = 0$ implies $u_{i,0+1} = u_{i,0-1}$ and substituted it in (114). Using two equations (114) and (116) for evolution can be avoided if we supply the $u_{i,-1}$ using

$$u_{i,-1} = u_{i,0} + \frac{\Delta t^2}{2\Delta x^2}\Big[u_{i+1,0} + u_{i-1,0} - 2u_{i,0}\Big] \tag{117}$$

in our setup of boundary condition.