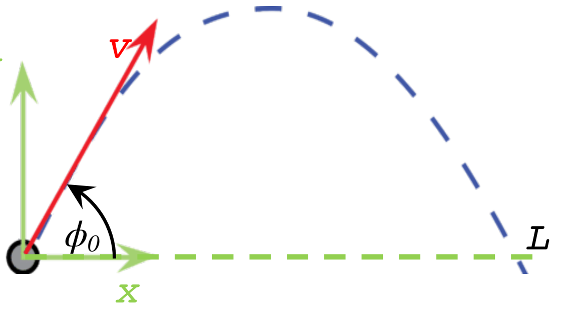
Projectile Motion with Air Resistance

- A projectile is fired from a cannon at the origin of the coordinates with initial velocity $v_0 = 18\ m/s$ in the attempt of hitting a target placed at the fixed distance of $L = 10\ m$ (horizontal distance from the cannon). The projectile motion can be modelled with the simple set of equations:

$$\begin{aligned} y' &= \tan\phi \\ v' &= -\frac{g}{v}\tan\phi - Cv\sec\phi \\ \phi' &= -\frac{g}{v^2} \end{aligned}$$



- Here a prime (e.g. $y' = dy/dx$) indicates derivative with respect to space, while y is the height of the projectile above the level of the cannon, v is the velocity of the projectile, and ϕ is the angle of the trajectory of the projectile with the horizontal.

- Also, $g = 9.81\ m/s^2$ is the gravitational acceleration, $C = 0.1\ m^{-1}$ accounts for air resistance.

- Find the initial shooting angles ϕ_0 and ϕ_1 (in degrees) and upload a single **.pdf** file containing:
  - A plot of the two solutions $y_0(x)$, $y_1(x)$ corresponding to the two shooting angles ϕ_0 and ϕ_1;
  - What are the corresponding angles in absence of air resistance ? [provide analytical derivation];
  - The C++ code used to find the solutions (excluding library files);

1D Heat Equation

- The heat equation is a partial differential equation describing the evolution of the temperature distribution in some region of space:

$$\frac{\partial T(x,t)}{\partial t} = \kappa \frac{\partial^2 T(x,t)}{\partial x^2}, \qquad x \in [x_b, x_e] \quad t \in [0, t_e]$$

where $\kappa = 1$ is the diffusion coefficient, `[`x_b`, `x_e`] = [-1,1]; `t_e` = 0.1.`

- The previous PDE is complemented by an **initial condition** specifying the temperature distribution at $t = 0$:

$$\boxed{T(x,0) \equiv T_0(x) = 1 + e^{-x^2/a^2} \qquad (a = 0.25)}$$

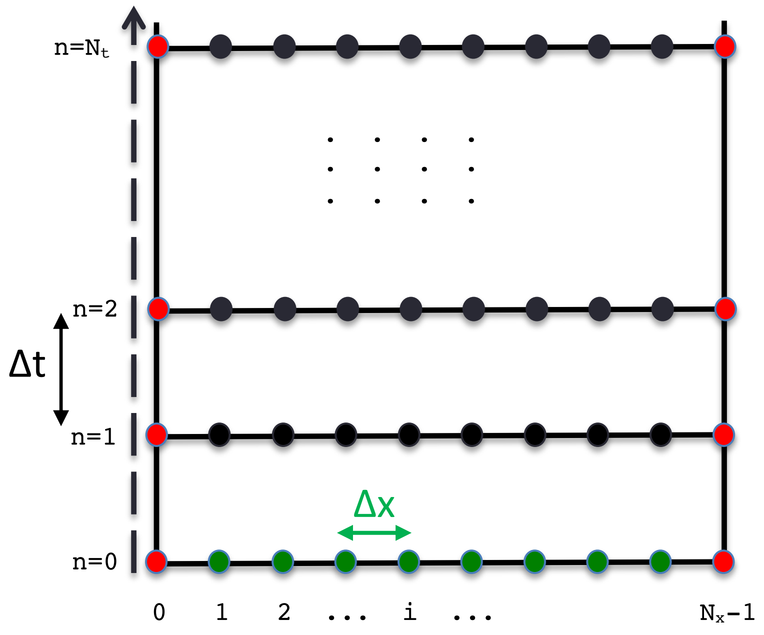
- Also, two boundary conditions, giving the temperature at $x = x_b$ and $x = x_e$ as a function of time must be assigned at each time step:

$$\boxed{T(x_b,t) = f_L(t) = 1\,, \quad T(x_e,t) = f_R(t) = 1}$$

- The discretization procedure replaces the domain $[x_b, x_e] \times [0, t_e]$ with a set of regularly-spaced mesh points (see figure):

$$x_i = x_b + i\Delta x\,, \quad t_n = n\Delta t$$

where $i = 0,\dots,N_x-1$ and $n = 0,\dots, N_t$ are the spatial and temporal indices.
- Set N_x`=256` as the number of spatial points (including boundary values) while N_t`=100` is the number of temporal steps.



Space-time discretization of the 1D diffusion equation. Green dots represents the initial condition, red points represents boundary values. Black points are the solution values to be found.
Notice that the values at $i=0$ and $i=N_x-1$ are known values and should **not** be evolved in time.

1D Heat Equation

- Using a *forward/backward approximation* to the *time derivative* and a ***central approximation*** to the ***spatial derivative*** to the original equation yields either the explicit / implicit Euler scheme:

$$\begin{aligned} T_i^{n+1} - T_i^n &= \alpha\left(T_{i-1}^n - 2T_i^n + T_{i+1}^n\right) &\equiv R_i^n \qquad &\text{(Forward Euler)} \\ T_i^{n+1} - T_i^n &= \alpha\left(T_{i-1}^{n+1} - 2T_i^{n+1} + T_{i+1}^{n+1}\right) &\equiv R_i^{n+1} \qquad &\text{(Backward Euler)} \end{aligned}$$

  where $\alpha = \kappa\Delta t/\Delta x^2$. is called the Courant (or CFL number).

- A linear combination of the two methods gives the well-known **theta-rule:**

$$\boxed{T_i^{n+1} - T_i^n = (1-\theta)R_i^n + \theta R_i^{n+1}}$$

- Where $\boldsymbol{\theta} = \mathbf{0}$ reproduces the forward (explicit) Euler scheme while $\boldsymbol{\theta} = \mathbf{1}$ yields the backward (implicit) Euler scheme.

- The forward explicit scheme has small stability range ($\alpha < ½$) but the implicit scheme is unconditionally stable for <u>*any*</u> Δt.

- The **theta-rule** can be cast in the form of a tridiagonal system of equations in the unknowns:

$$T_i^{n+1} \quad \text{for} \quad i = 1, ..., N_x - 2$$

- Find the coefficients of the tridiagonal matrix and solve the system at each time step $n=1,2,3\ldots N_t$ using the special value $\theta=1/2$ which corresponds to the popular 2nd order-accurate Crank-Nicholson scheme.

- Upload a single **.pdf** file containing:
  - The coefficients (possibly with the derivation) of the linear system of equations;
  - A plot of the solution(s) at the intermediate and final time steps ($t = 0.05$, $t=t_e=0.1$)
  - The C++ code (excluding library files)

    [Please do not use 2D arrays ! A single 1D array is sufficient]