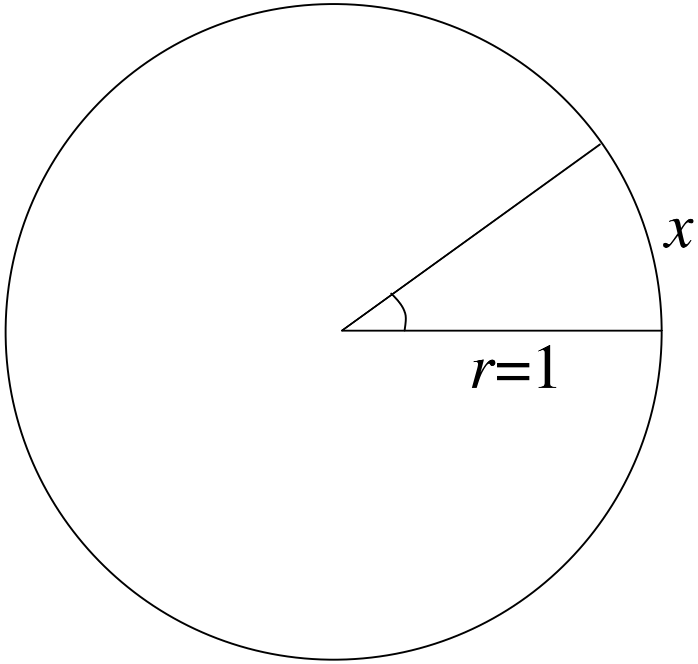
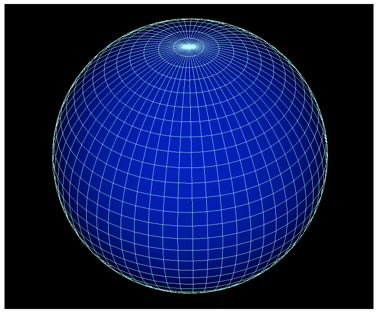
Finite Difference Method (FDM) :
An explanation through a simple 1-D problem

The diffusion equation on a circle



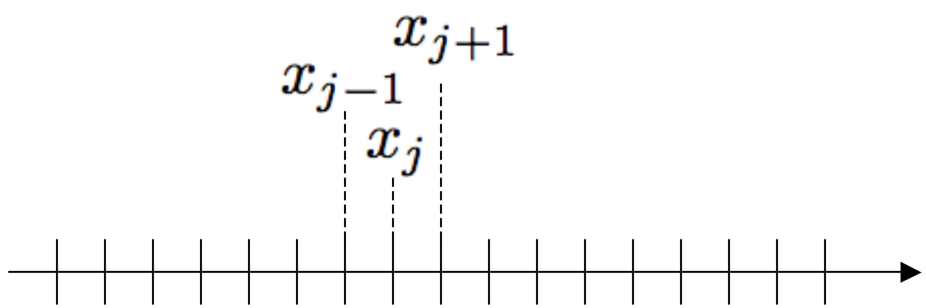
$$\frac{\partial \psi}{\partial t} = \frac{\partial^2 \psi}{\partial x^2}$$

$$0 \leq x < 2\pi$$



$$\boxed{\frac{\partial \psi}{\partial t} = \frac{\partial^2 \psi}{\partial x^2}}$$

FDM: Finite Difference Method



$$\frac{d\psi}{dx} = \frac{\psi_{j+1} - \psi_{j-1}}{2\Delta x} + O(\Delta x)^2$$

$$\frac{d^2\psi}{dx^2} = \frac{\psi_{j+1} - 2\psi_j + \psi_{j-1}}{(\Delta x)^2} + O(\Delta x)^2$$

$$\boxed{\frac{d\psi_j}{dt} = \frac{\psi_{j+1} - 2\psi_j + \psi_{j-1}}{(\Delta x)^2}}$$

$$\boxed{\frac{\partial \psi}{\partial t} = \frac{\partial^2 \psi}{\partial x^2}}$$

FDM: Finite Difference Method

$$\boxed{\frac{d\psi_j}{dt} = \frac{\psi_{j+1} - 2\psi_j + \psi_{j-1}}{(\Delta x)^2}}$$

$$\frac{d\psi_j}{dt} = f(\psi_1, \psi_2, \cdots, \psi_N)$$

==> Time integration.

4-step 4th-order Runge-Kutta Integration Method

$$\boxed{\frac{du(t)}{dt} = f(t, u(t))}$$

$$t_0 = t_n$$
$$u_0 = u(t_0)$$
$$df_1 = \Delta t\, f(t_0, u_0)$$

$$t_1 = t_0 + 0.5\,\Delta t$$
$$u_1 = u_0 + 0.5\, df_1$$
$$df_2 = \Delta t\, f(t_1, u_1)$$

$$t_2 = t_0 + 0.5\,\Delta t$$
$$u_2 = u_0 + 0.5\, df_2$$
$$df_3 = \Delta t\, f(t_2, u_2)$$

$$t_3 = t_0 + \Delta t$$
$$u_3 = u_0 + df_3$$
$$df_4 = f(t_3, u_3)$$

$$u_{n+1} = u_n + \tfrac{1}{6}(df_1 + 2\, df_2 + 2\, df_3 + df_4)$$

Error $O(\Delta t^5)$ for one step, $O(\Delta t^4)$ in total.

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

Combination of RK + FDM

$$u_0 = e^{ikx_j}$$

A test wave

$$df_1 = \frac{\kappa \Delta t}{(\Delta x)^2} \left\{ e^{ik(x_j + \Delta x)} - 2\, e^{ikx_j} + e^{ik(x_j - \Delta x)} \right\}$$

$$= -\frac{2\kappa \Delta t}{(\Delta x)^2} (1 - \cos k\Delta x)\, e^{ikx_j}$$

$$= -\alpha\, e^{ikx_j}$$

$$\alpha = \frac{2\kappa \Delta t}{(\Delta x)^2} (1 - \cos k\Delta x)$$

$$u_1 = u_0 + 0.5\, df_1$$

$$= \left(1 - \tfrac{\alpha}{2}\right) e^{ikx_j}$$

$$df_2 = -\frac{2\,\kappa \Delta t}{(\Delta x)^2} \left(1 - \tfrac{\alpha}{2}\right) (1 - \cos k\Delta x)\, e^{ikx_j}$$

$$= -\alpha \left(1 - \tfrac{\alpha}{2}\right) e^{ikx_j}$$

$$
\begin{aligned}
u_3 &= u_0 + 0.5\, df_2 \\
&= \left(1 - \tfrac{\alpha}{2} + \tfrac{\alpha^2}{4}\right)\, e^{ikx_j} \\
df_3 &= -\left(\alpha - \frac{\alpha^2}{2} + \frac{\alpha^3}{4}\right)\, e^{ikx_j} \\
u_4 &= u_0 + df_3 \\
&= \left(1 - \alpha + \frac{\alpha^2}{2} - \frac{\alpha^3}{4}\right)\, e^{ikx_j} \\
df_4 &= -\left(\alpha - \alpha^2 + \frac{\alpha^3}{2} - \frac{\alpha^4}{4}\right)\, e^{ikx_j} \\
u_{\mathrm{new}} &= u_0 + \tfrac{1}{6}\left(df_1 + 2\, df_2 + 2\, df_3 + df_4\right) \\
&= \left(1 - \alpha + \frac{\alpha^2}{2} - \frac{\alpha^3}{6} + \frac{\alpha^4}{24}\right)\, e^{ikx_j} \\
&= \left\{1 + \frac{(-\alpha)}{1!} + \frac{(-\alpha)^2}{2!} + \frac{(-\alpha)^3}{3!} + \frac{(-\alpha)^4}{4!}\right\}\, e^{ikx_j}
\end{aligned}
$$

By one step integration of 4-th order Runge-Kutta method,

$$u_{\text{new}} = \left\{ 1 + \frac{(-\alpha)}{1!} + \frac{(-\alpha)^2}{2!} + \frac{(-\alpha)^3}{3!} + \frac{(-\alpha)^4}{4!} \right\} e^{ikx_j}$$

$$\alpha = \frac{2\kappa\Delta t}{(\Delta x)^2}(1 - \cos k\Delta x)$$

When $k\Delta x << 1$ $\quad \alpha \sim \frac{\kappa\Delta t}{(\Delta x)^2}(k\Delta x)^2 = k^2\,\kappa\Delta t$

$$u_{\text{exact}} = e^{-k^2\,\kappa\Delta t}\,e^{ikx_j} = e^{-\alpha}e^{ikx_j}$$

Error in 1step $= O[(\Delta t)^5]$ $\Longrightarrow$ Error in total $= O[(\Delta t)^4]$

By one step integration of 4-th order Runge-Kutta method,

$$u_{\text{new}} = \left\{ 1 + \frac{(-\alpha)}{1!} + \frac{(-\alpha)^2}{2!} + \frac{(-\alpha)^3}{3!} + \frac{(-\alpha)^4}{4!} \right\} e^{ikx_j}$$

$$\alpha = \frac{2\kappa\Delta t}{(\Delta x)^2}(1 - \cos k\Delta x)$$

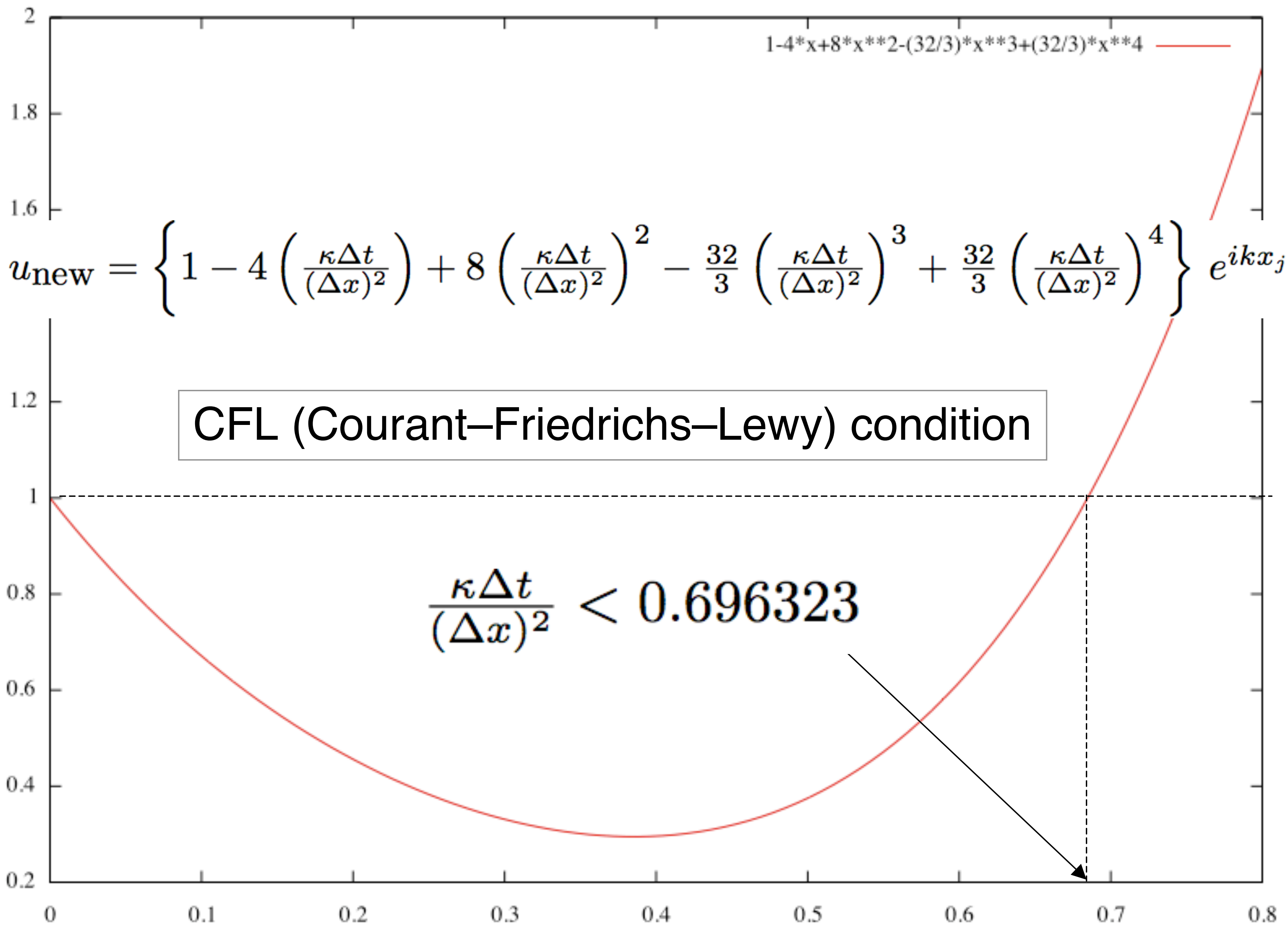
When $k\Delta x = \pi, \quad \alpha = \frac{4\kappa\Delta t}{(\Delta x)^2}$

$$u_{\text{new}} = \left\{ 1 - 4\left(\frac{\kappa\Delta t}{(\Delta x)^2}\right) + 8\left(\frac{\kappa\Delta t}{(\Delta x)^2}\right)^2 - \frac{32}{3}\left(\frac{\kappa\Delta t}{(\Delta x)^2}\right)^3 + \frac{32}{3}\left(\frac{\kappa\Delta t}{(\Delta x)^2}\right)^4 \right\} e^{ikx_j}$$

$$u_{\text{new}} = \left\{ 1 - 4\left(\frac{\kappa \Delta t}{(\Delta x)^2}\right) + 8\left(\frac{\kappa \Delta t}{(\Delta x)^2}\right)^2 - \frac{32}{3}\left(\frac{\kappa \Delta t}{(\Delta x)^2}\right)^3 + \frac{32}{3}\left(\frac{\kappa \Delta t}{(\Delta x)^2}\right)^4 \right\} e^{ikx_j}$$

CFL (Courant–Friedrichs–Lewy) condition

$$\frac{\kappa \Delta t}{(\Delta x)^2} < 0.696323$$



Simple Numerical Simulation with Fortran90 Code

In sourcodes.tar.gz,
./src/DiffusionEquation/

A Sample Code in FDM

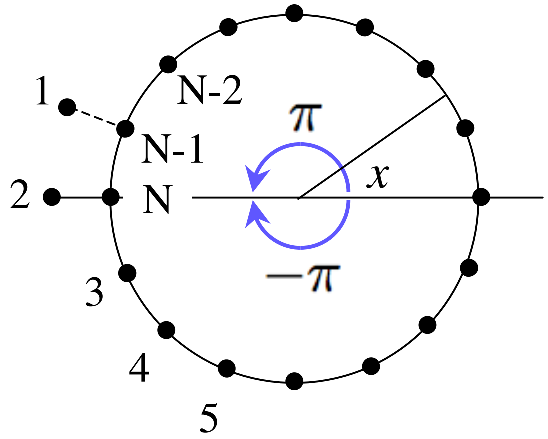
$$\boxed{\frac{\partial \psi}{\partial t} = \frac{\partial^2 \psi}{\partial x^2}}$$

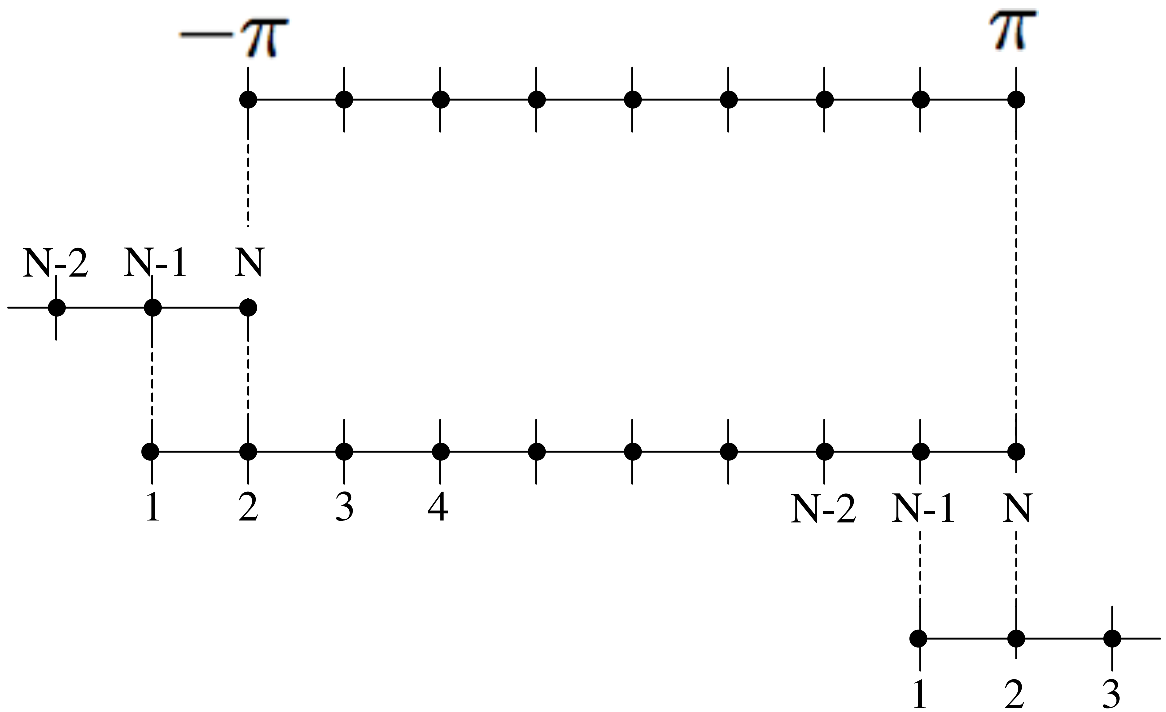
$$\boxed{\frac{d\psi_j}{dt} = \frac{\psi_{j+1} - 2\psi_j + \psi_{j-1}}{(\Delta x)^2}}$$

$$\frac{d\psi_j}{dt} = f(\psi_1, \psi_2, \cdots, \psi_N)$$

==> 4-step (4th order) Runge-Kutta method

Periodic boundary condition






```
subroutine iBoundary_condition(psi)
   real(DP), dimension(nx), intent(inout) :: psi

   psi(1)     = psi(nx-1)
   psi(nx)    = psi(2)

 end subroutine iBoundary_condition
```

Now let's see the code: main.f90

```
do nloop = 1 , nloop_max

   dpsi01(:) = rk4__step('1st',dt,dx,psi)
   call iBoundary_condition(dpsi01)

   dpsi02(:) = rk4__step('2nd',dt,dx,psi,dpsi01)
   call iBoundary_condition(dpsi02)

   dpsi03(:) = rk4__step('3rd',dt,dx,psi,dpsi02)
   call iBoundary_condition(dpsi03)

   dpsi04(:) = rk4__step('4th',dt,dx,psi,dpsi03)
   call iBoundary_condition(dpsi04)

   time = time + dt
   psi(:) = psi(:) + ONE_SIXTH*(dpsi01(:)      &
                            +2*dpsi02(:)      &
                            +2*dpsi03(:)      &
                              +dpsi04(:))

end do
```

Runge-Kutta step (rk.f90)

```
function rk4__step(nth,dt,dx,psi,dpsi_prev)      &
                                     result(dpsi_new)

    character(len=3), intent(in)          :: nth
    real(DP), intent(in)                  :: dt
    real(DP), intent(in)                  :: dx
    real(DP), dimension(:), intent(in)    :: psi
    real(DP), dimension(size(psi,dim=1)),    &
                    intent(in), optional :: dpsi_prev
    real(DP), dimension(size(psi,dim=1)) :: dpsi_new
    real(DP), dimension(size(psi,dim=1)) :: psi_
```

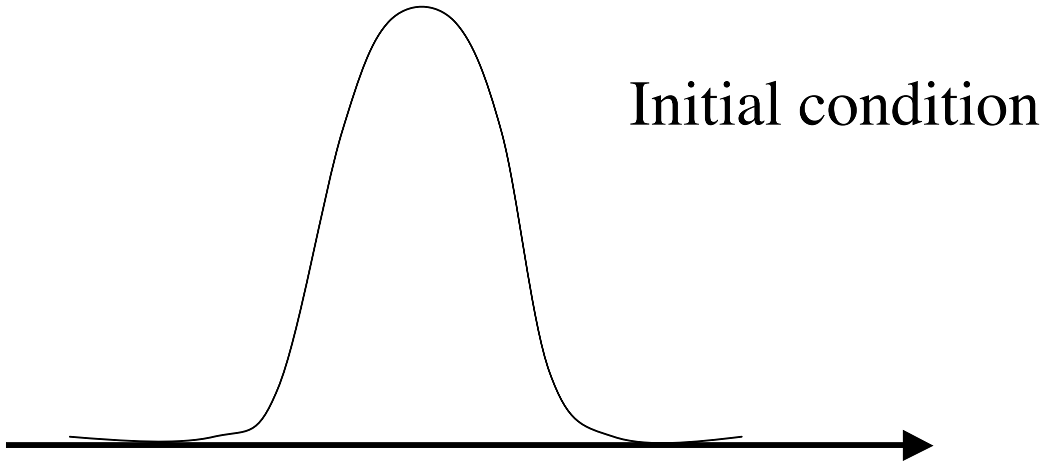
```
  select case (nth)
  case ('1st')
    dpsi_new(:) = dt*diffusion_equation(size(psi,dim=1), &
                                        dx,psi)
  case ('2nd')
    psi_(:) = psi(:) + dpsi_prev(:)*0.5_DP
    dpsi_new(:) = dt*diffusion_equation(size(psi,dim=1), &
                                        dx,psi_)
  case ('3rd')
    psi_(:) = psi(:) + dpsi_prev(:)*0.5_DP
    dpsi_new(:) = dt*diffusion_equation(size(psi,dim=1), &
                                        dx,psi_)
  case ('4th')
    psi_(:) = psi(:) + dpsi_prev(:)
    dpsi_new(:) = dt*diffusion_equation(size(psi,dim=1), &
                                        dx,psi_)
  end select

end function rk4__step
```

diffusion_equation (rk.f90)

```
function diffusion_equation(nx,dx,psi)
   integer, intent(in)                        :: nx
   real(DP), intent(in)                       :: dx
   real(DP), dimension(nx), intent(in)        :: psi
   real(DP), dimension(nx)                    :: diffusion_equation

   integer :: i
   real(DP) :: dx2

   dx2 = namelist__double('Diffusion_coeff')/(dx**2)

   do i = 2 , nx-1
      diffusion_equation(i) = dx2*(psi(i+1)-2*psi(i)+psi(i-1))
   end do

 end function diffusion_equation
```

$$\kappa \frac{\psi_{j+1} - 2\psi_j + \psi_{j-1}}{(\Delta x)^2}$$

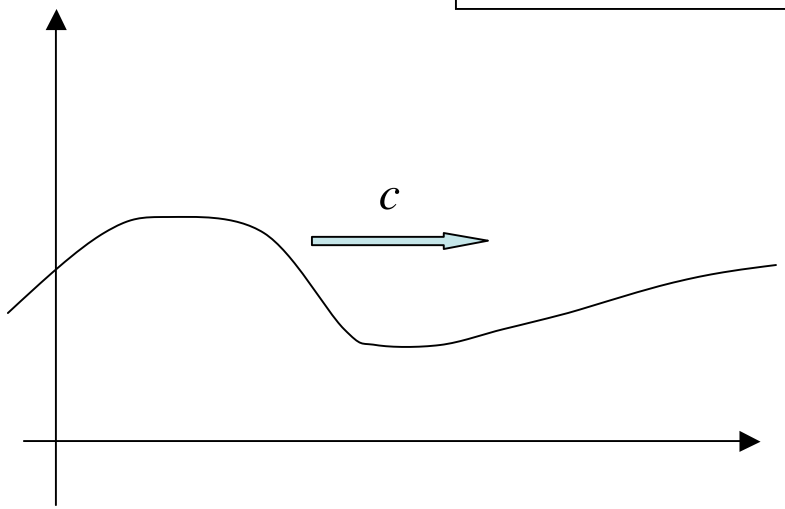
Let's run the code



Other equations by FDM (nonlinear terms)

Burgers' equation

$$\frac{\partial \psi}{\partial t} = -\psi \frac{\partial \psi}{\partial x} + \nu \frac{\partial^2 \psi}{\partial x^2}$$



$$\frac{\partial \psi}{\partial t} = -c \frac{\partial \psi}{\partial x}$$

Solution:

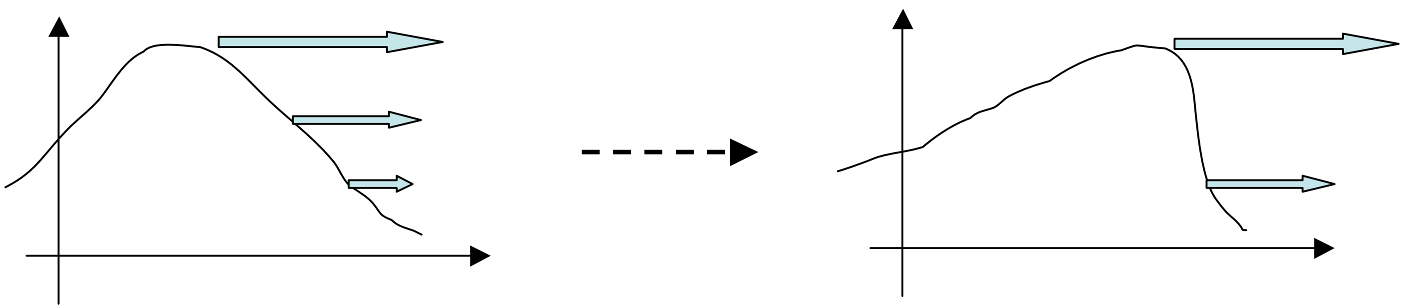
$$\psi(x,t) = f(x - ct)$$

Burgers' equation

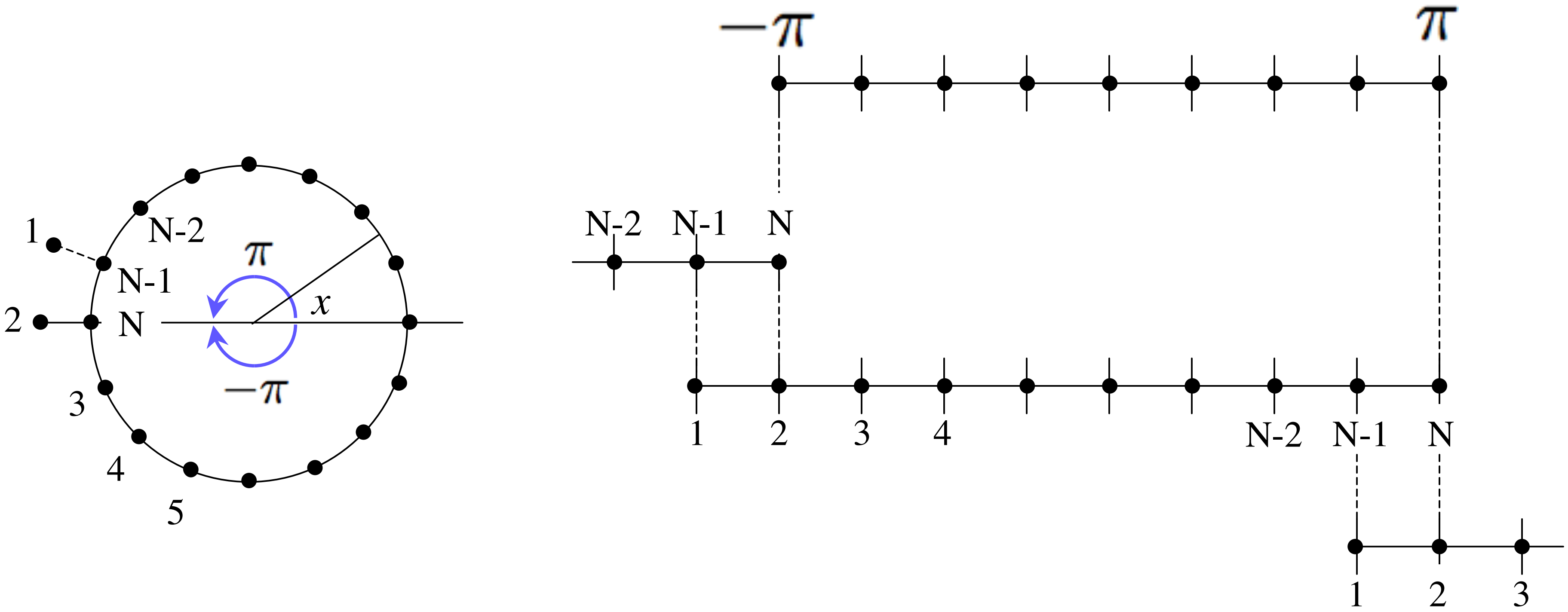
$$\frac{\partial \psi}{\partial t} = -\psi \frac{\partial \psi}{\partial x} + \nu \frac{\partial^2 \psi}{\partial x^2}$$

Diffusion term

$$\frac{\partial \psi}{\partial t} = -\psi \frac{\partial \psi}{\partial x}$$



Burgers' equation by FDM



$$\frac{d\psi_j}{dt} = -\psi_j \frac{\psi_{j+1} - \psi_{j-1}}{2\Delta x} + \nu \frac{\psi_{j+1} - 2\psi_j + \psi_{j-1}}{(\Delta x)^2}$$

Runge-Kutta 1st step (rk4.f90)

```
select case (nth)
   case ('1st')
      dpsi_new(:) = dt*burgers_equation(size(psi,dim=1),dx,psi)
   case ('2nd')
      psi_(:) = psi(:) + dpsi_prev(:)*0.5_DP
      dpsi_new(:) = dt*burgers_equation(size(psi,dim=1),dx,psi_)
   case ('3rd')
      psi_(:) = psi(:) + dpsi_prev(:)*0.5_DP
      dpsi_new(:) = dt*burgers_equation(size(psi,dim=1),dx,psi_)
   case ('4th')
      psi_(:) = psi(:) + dpsi_prev(:)
      dpsi_new(:) = dt*burgers_equation(size(psi,dim=1),dx,psi_)

end select
```

burgers_equation (rk4.f90)

```
function burgers_equation(nx,dx,psi)
  integer, intent(in)                        :: nx
  real(DP), intent(in)                       :: dx
  real(DP), dimension(nx), intent(in) :: psi
  real(DP), dimension(nx)                    :: burgers_equation

  integer :: i
  real(DP) :: dx1, dx2

  dx1 = 1.0_DP / (2*dx)
  dx2 = namelist__double('Diffusion_coeff')/(dx**2)

  do i = 2 , nx-1
    burgers_equation(i) = - psi(i)*dx1*(psi(i+1)-psi(i-1)) &
                 + dx2*(psi(i+1)-2*psi(i)+psi(i-1))
  end do

end function burgers_equation
```

$$\frac{d\psi_j}{dt} = -\psi_j \frac{\psi_{j+1} - \psi_{j-1}}{2\Delta x} + \nu \frac{\psi_{j+1} - 2\psi_j + \psi_{j-1}}{(\Delta x)^2}$$

Let’s run the code