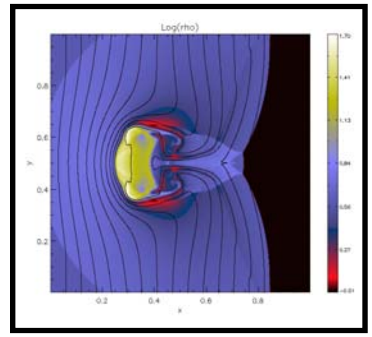
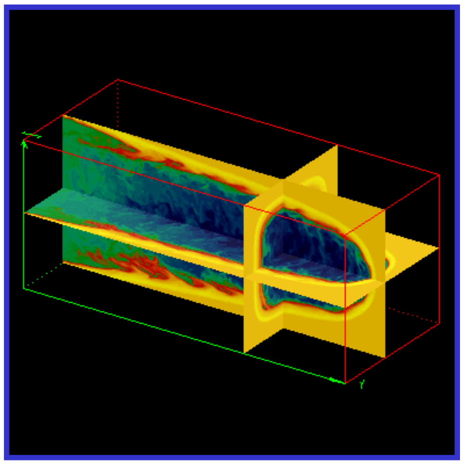
# Computational Astrophysics

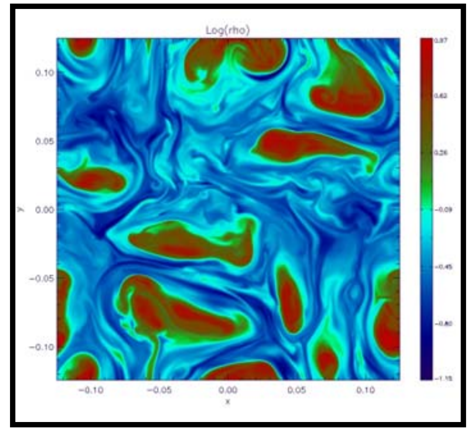


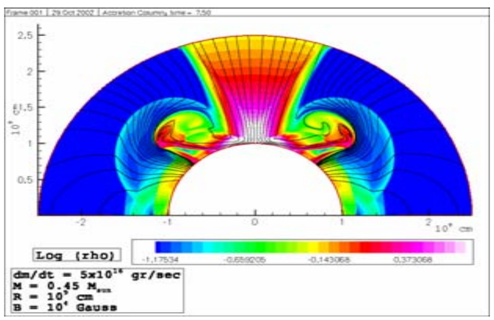



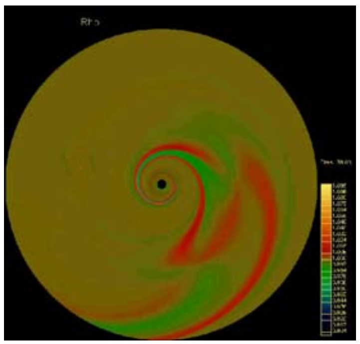
*- Andrea Mignone -*

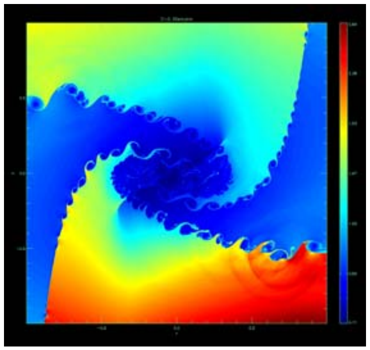
***Università di Torino***

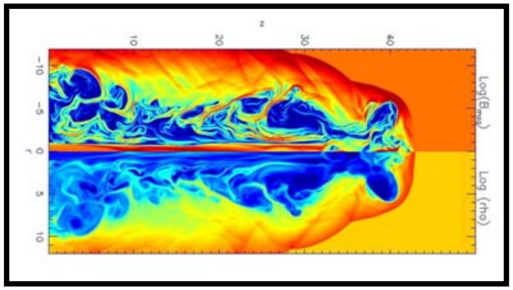
***INAF Osservatorio Astronomico di Torino***









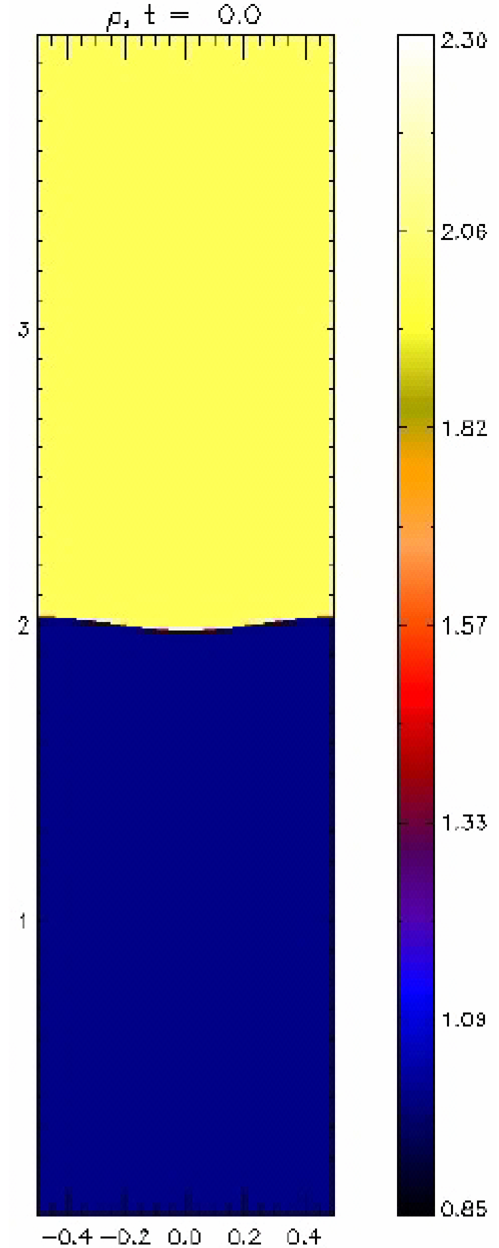
# Outline

1. Basic Discretization Methods
2. The Linear Advection Equation
3. Systems of Linear Advection Equations
4. Nonlinear Extension
5. Euler Equations
6. MHD Equations

# Example of Hyperbolic Systems:

Euler Equations of Fluid Dynamics

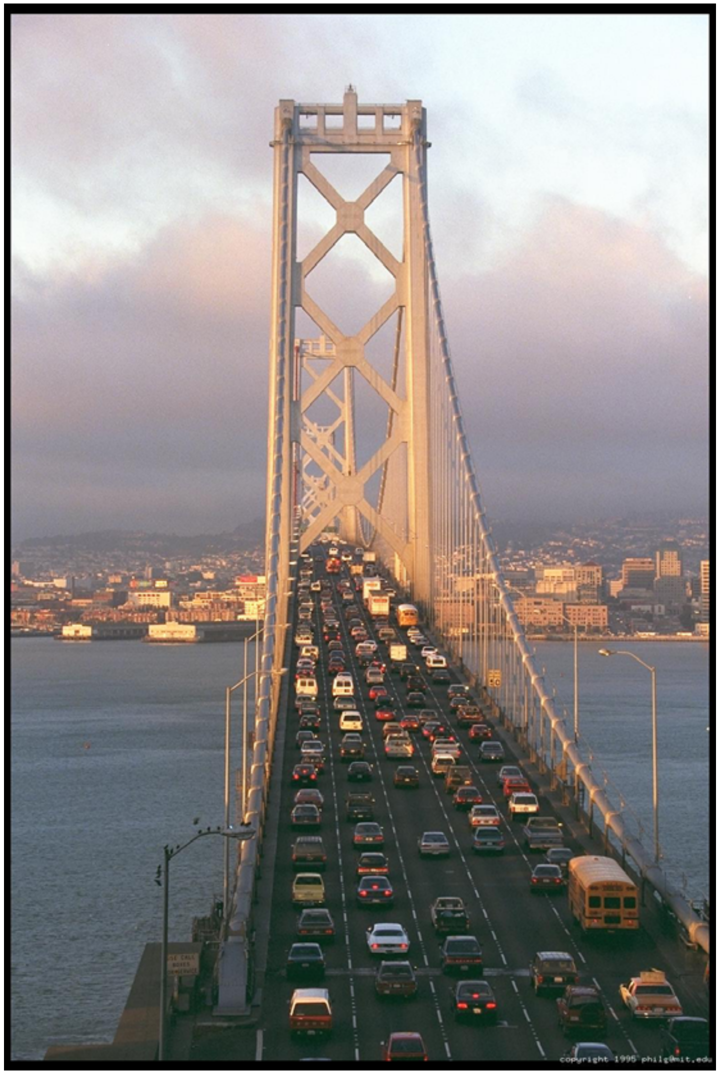
$$\frac{\partial}{\partial t}\begin{pmatrix}\rho \\ \rho \mathbf{v} \\ E\end{pmatrix} + \nabla \cdot \begin{pmatrix}\rho \mathbf{v} \\ \rho \mathbf{v}\mathbf{v}^{\mathbf{T}} + pI \\ (E+p)\mathbf{v}\end{pmatrix} = 0$$



# Example of Hyperbolic Systems:

Traffic Flow Equations

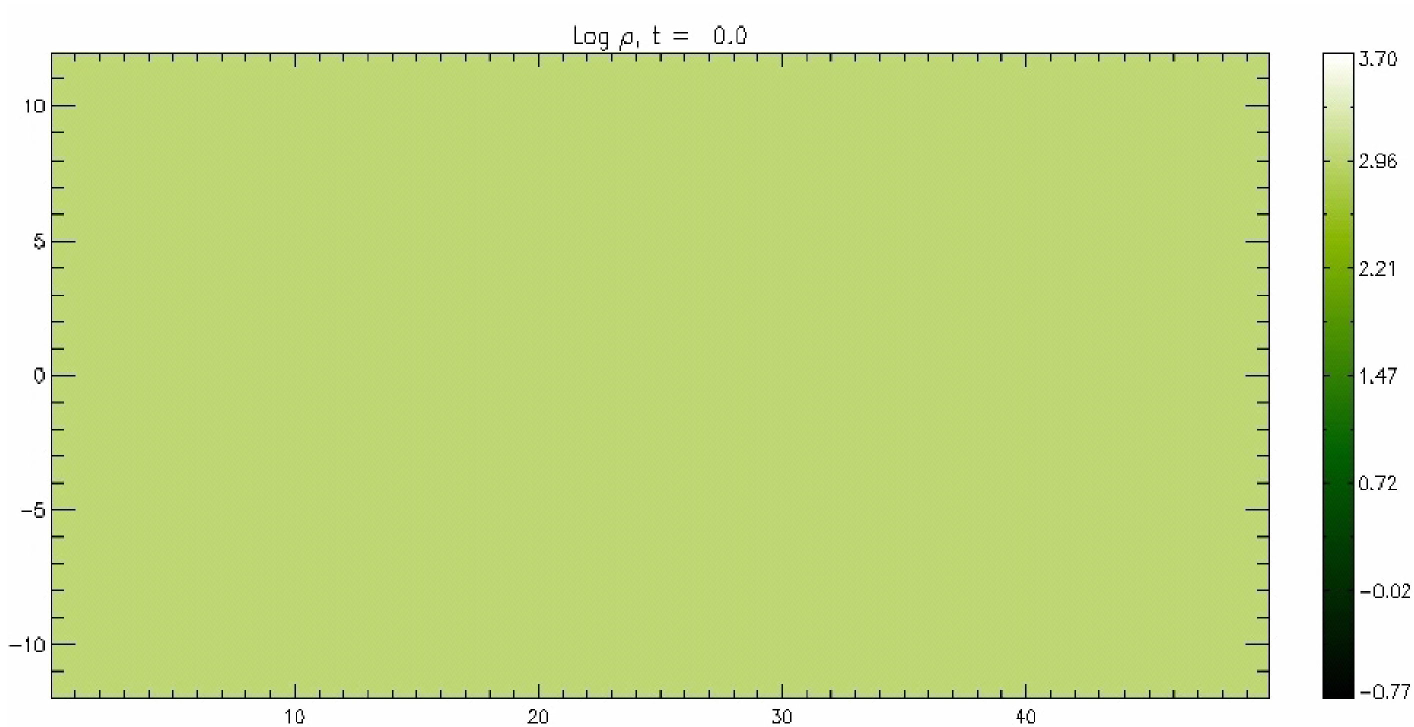
$$\frac{\partial q}{\partial t} + \frac{\partial}{\partial x}\left(u_{\max} q(1-q)\right) = 0$$



# Example of Hyperbolic Systems:

Relativistic Euler Equations

$$\boxed{\begin{aligned}\nabla_\mu(\rho u^\mu) &= 0\\ \nabla_\mu T^{\mu\nu} &= 0\end{aligned}}$$



# Example of Hyperbolic Systems:

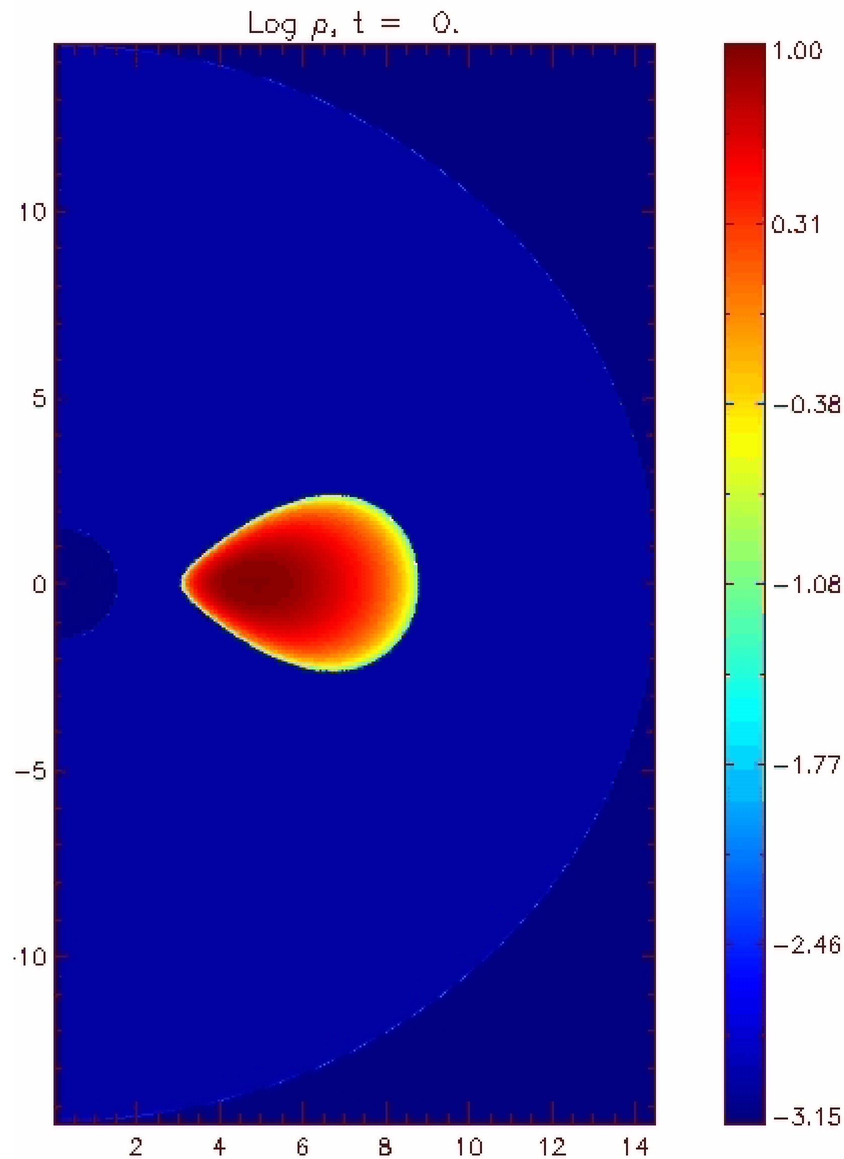
Magnetohydrodynamics (MHD) Equations

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0$$

$$\frac{\partial (\rho \mathbf{v})}{\partial t} + \nabla \cdot \left(\rho \mathbf{v}\mathbf{v}^{\mathbf{t}} + p\mathbf{I}\right) = (\nabla \times \mathbf{B}) \times \mathbf{B}$$

$$\frac{\partial E_{hd}}{\partial t} + \nabla \cdot [(E_{hd} + p)\mathbf{v}] = -(\mathbf{v} \times \mathbf{B}) \cdot (\nabla \times \mathbf{B})$$

$$\frac{\partial \mathbf{B}}{\partial t} - \nabla \times (\mathbf{v} \times \mathbf{B}) = 0$$



# 1 - Basic Discretization Methods

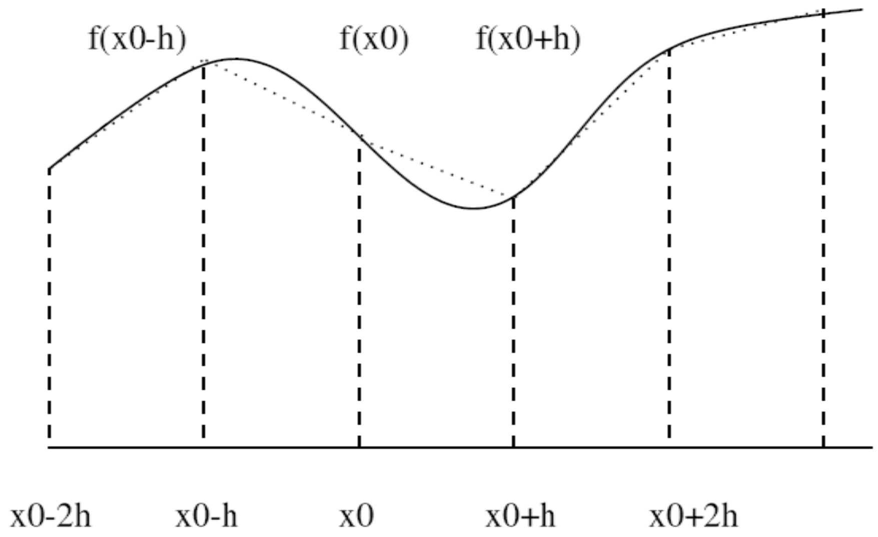
## Numerical Differentiation, Integration and ODE

*- Andrea Mignone -*

***Università di Torino***
***INAF Osservatorio Astronomico di Torino***

# Differentiation: Finite Difference Methods

- *Problem*:given a function known only at specific points (e.g. in space), equally spaced with step size h:

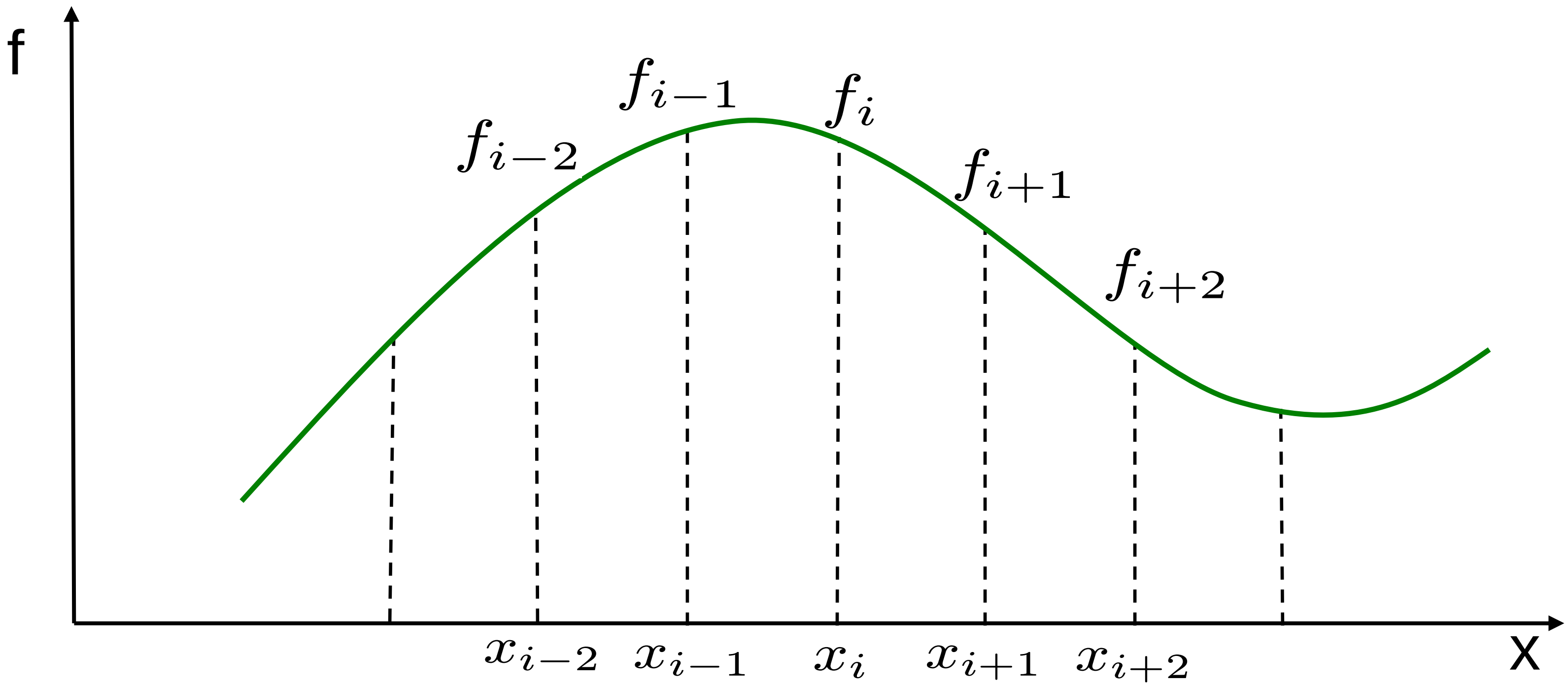



- → Need to compute the spatial derivative $\frac{df(x)}{dx}$

# Finite Difference Methods

- Convention: tabulated values of f(x) are given by

$$f(x) = f_i\,, \quad f(x+h) = f_{i+1}\,, \quad f(x+2h) = f_{i+2}\,, \ \ldots$$
$$f(x-h) = f_{i-1}\,, \quad f(x-2h) = f_{i-2}\,, \ \ldots$$



# Finite Difference: 1st derivative

- Recall the mathematical definition of the derivative:

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

- If we use a *Taylor expansion* for f(x):

$$f(x+h) \equiv f_{i+1} = f_i + h f_i' + \frac{h^2}{2} f_i'' + \frac{h^3}{6} f_i''' + \ldots \qquad (1)$$

$$f(x-h) \equiv f_{i-1} = f_i - h f_i' + \frac{h^2}{2} f_i'' - \frac{h^3}{6} f_i''' + \ldots \qquad (2)$$

- Solving for f'(x) →

$$f_i' \approx \frac{f_{i+1} - f_i}{h} - \frac{h}{2} f_i'' + \ldots$$

$$f_i' \approx \frac{f_i - f_{i-1}}{h} + \frac{h}{2} f_i'' + \ldots$$

# Finite Difference: 1st derivative

- Thus we can represent the derivative with
  - *Forward difference*: $$\boxed{f_i' \approx \frac{f_{i+1} - f_i}{h} - \frac{h}{2} f_i'' + \ldots}$$
  - *Backward difference*: $$\boxed{f_i' \approx \frac{f_i - f_{i-1}}{h} + \frac{h}{2} f_i'' + \ldots}$$
- *1st order* accurate: the dominant error ~h
- The term $\frac{h}{2} f''(x)$ is called *truncation error*
- Exact for lines, f(x) = a + bx

# Finite Difference: 1st derivative

- Can we do better than 1st order ? YES!! → Seek for a quadratic expression by using a 3-point stencil. Consider again Taylor expansion:

$$f_{i\pm1} = f_i \pm h f_i' + \frac{h^2}{2} f_i'' \pm \frac{h^3}{3!} f_i''' + \ldots$$

- Calculating both f(x+h) and f(x-h) and subtracting, one has

$$f_i' = \frac{f_{i+1} - f_{i-1}}{2h} - \frac{h^2}{6} f_i''' + O(h^3)$$

# Finite Difference: 1st derivative

- Thus we end up with the *centered derivative*:

$$\boxed{f_i' = \frac{f_{i+1} - f_{i-1}}{2h} - \frac{h^2}{6} f_i''' + O(h^3)}$$

- *Second-order* accurate: the dominant error goes like h^2.

- For a quadratic function f(x) = a + bx + cx^2 the centered derivative formula gives the *exact answer* b + 2cx.

# Finite Difference: 1st derivative

- A *fourth order* approximation can be obtained using a 5-point stencil:

$$f_{i\pm 2} = f_i \pm 2hf_i' + 2h^2 f_i'' \pm \frac{4h^3}{3} f_i''' + O(h^4)$$

- So that an expression for the 1st derivative can be found:

$$\boxed{f_i' = \frac{f_{i-2} - 8f_{i-1} + 8f_{i+1} - 2f_{i+2}}{12h} + O(h^4)}$$

# Finite Difference: 2nd derivative

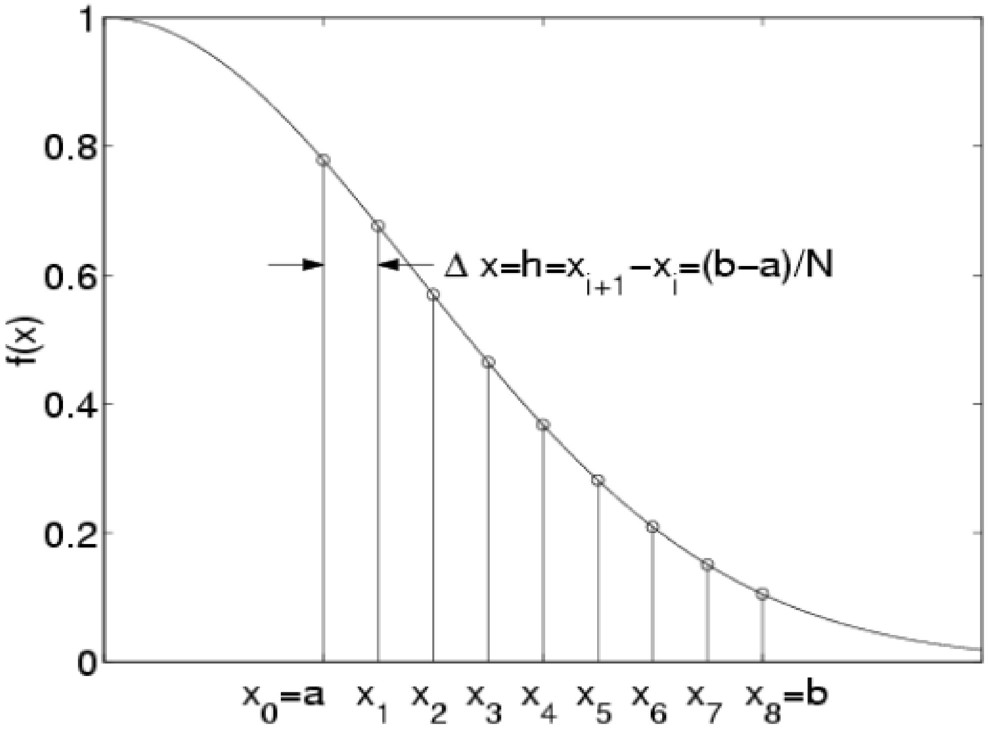
- Similar expressions may be found for derivatives of higher order;
- For example, the 2nd derivative can be expressed by:

$$f_i'' = \frac{f_{i+1} - 2f_i + f_{i-1}}{h^2} + O(h^2)$$

- Proof left as an *exercise.*

# Numerical Integration

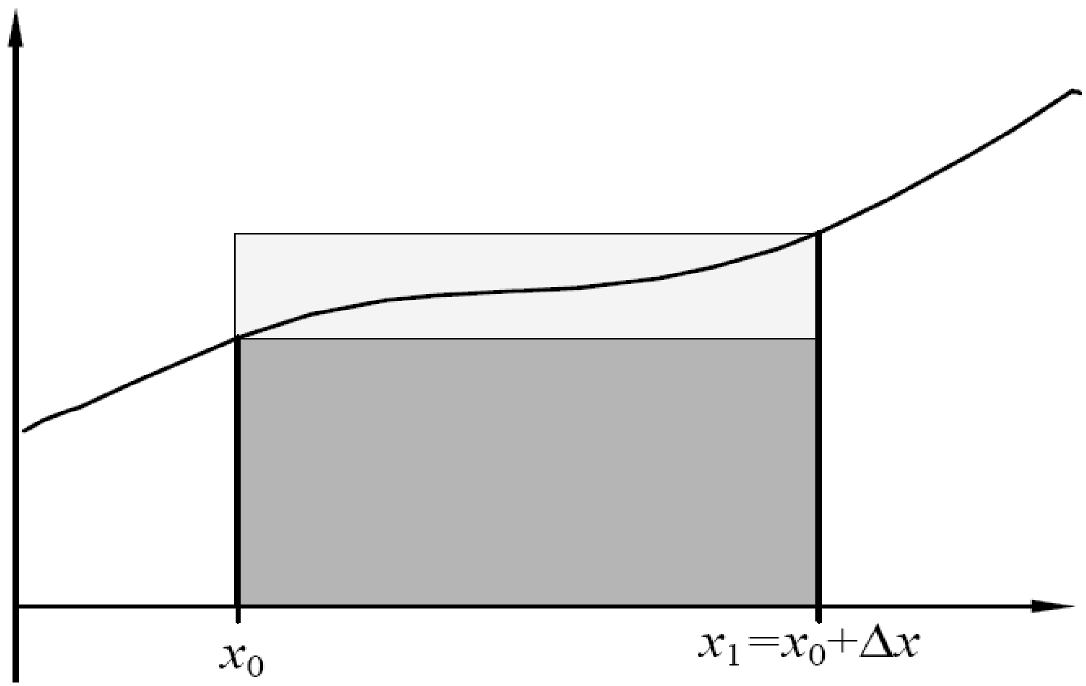
- Inverse *problem*: given a function known at specific points:




- → Want to evaluate $$\int_a^b f(x)dx$$

# Numerical Integration: Constant rule

- The simplest form is to assume f(x) constant over the interval being integrated:



# Numerical Integration: Constant rule

- Integrating the Taylor series:

$$\int_{x_i}^{x_{i+1}} f(x)dx = \int_{x_i}^{x_{i+1}} \left[f_i + f_i'(x - x_i) + \frac{f_i''}{2}(x - x_i)^2 + ...\right] dx$$

$$= hf_i + \frac{h^2}{2}f_i' + \frac{h^3}{6}f_i'' + O(h^4)$$

- keep only the first term:

$$\boxed{\int_{x_i}^{x_{i+1}} f(x)dx = hf_i + O(h^2)}$$

# Numerical Integration: Trapezoidal Rule:

- Using $f_i' = \dfrac{f_{i+1} - f_i}{h} + O(h)$ for the 1st derivative in the previous expression,

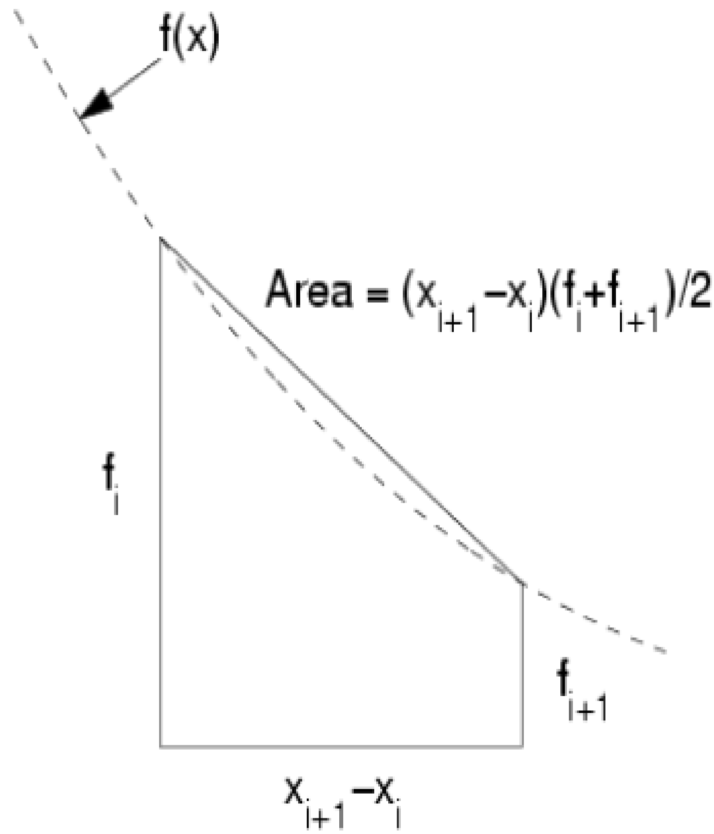
$$\int_{x_i}^{x_{i+1}} f(x)dx = hf_i + \frac{h^2}{2}f_i' + \frac{h^3}{6}f_i'' + ...$$

…one gets

$$\boxed{\int_{x_i}^{x_{i+1}} f(x)dx \approx \frac{h}{2}\left(f_i + f_{i+1}\right) + O(h^3)}$$

# Numerical Integration: Trapezoidal Rule:

- The trapezoidal rule approximates the integral of the function over the subinterval [xi, xi+1] as the area of the trapezoid created by the function values at fi  and fi+1



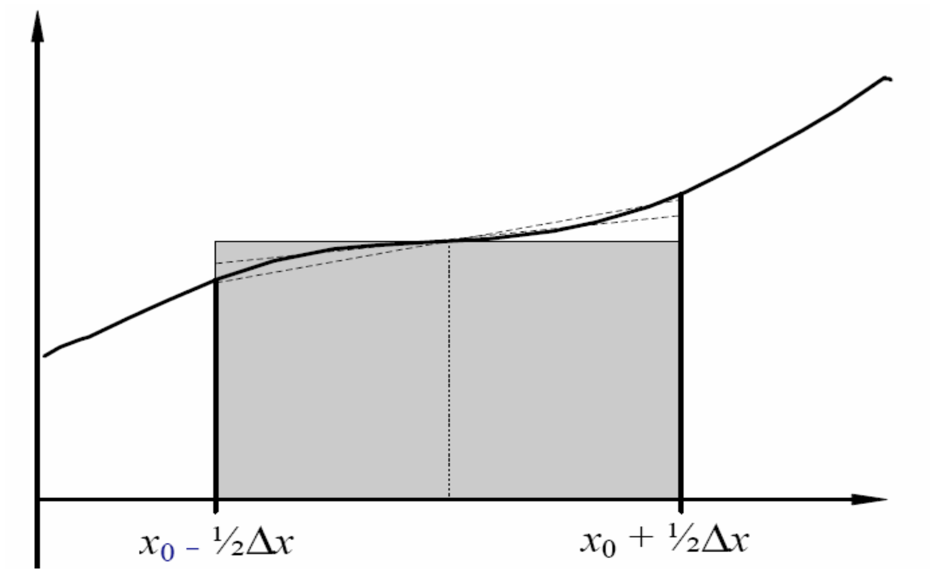
# Numerical Integration: Midpoint Rule

- A variant of the Trapezoidal rule is obtained by considering Taylor expansion around the *midpoint of the interval*:

$$\int_{x_{i-1/2}}^{x_{i+1/2}} f(x)dx = \int_{x_{i-1/2}}^{x_{i+1/2}} \left[ f_i + f_i'(x - x_i) + \frac{f_i''}{2}(x - x_i)^2 + \ldots \right] dx$$

$$= hf_i + O(h^3)$$

- The linear term cancels out!!



# Numerical Integration: Simpson Rule

- By using the expression for the second derivative in the mid-point rule, one obatins the *Simpson rule*

$$\int_{x_{i-1/2}}^{x_{i+1/2}} f(x)dx = \frac{h}{6}\left(f_{i-1/2} + 4f_i + f_{i+1/2}\right) + O(h^5)$$

- Based on a quadratic approximation to the function in the desired range.

# Numerical Integration: Solving Ordinary Differential Equations (ODE)

- Numerical quadrature can be used to solve ordinary differential equations, .e.g.

$$\frac{dy}{dx} = f(x,y) \quad \Longrightarrow \quad y_{i+1} - y_i = \int_{x_i}^{x_{i+1}} f(x,y)dx$$

with initial condition $y_i \quad \text{at} \quad x = x_i$

- The integrand depends on x and y as well;
- For example, with the constant rule, one has

$$\boxed{y_{i+1} - y_i = \int_{x_i}^{x_{i+1}} f(x,y)dx \approx f(x_i, y_i)\Delta x + O(\Delta x^2)}$$

This is called the explicit *Euler method.* It is only 1st order accurate.

# Numerical Integration: Solving ODE

- Higher accuracy can be achieved using, for example, the trapezoidal rule:

$$y_{i+1} - y_i = \int_{x_i}^{x_{i+1}} f(x,y)dx \approx \frac{\Delta x}{2}\left[f(x_i,y_i) + f(x_{i+1},y_{i+1})\right] + O(\Delta x^3)$$

- Problem: the unknown yi+1 appears on both side of the equation!!!
- Use an estimate (predictor) for yi+1 with Euler method:

$$y^*_{i+1} = y_i + f(x_i,y_i)\Delta x + O(\Delta x^2)$$

$$y_{i+1} = y_i + \frac{\Delta x}{2}\left[f(x_i,y_i) + f(x_{i+1},y^*_{i+1}) + O(\Delta x^2)\right] + O(\Delta x^3)$$

called the explicit *2nd order Runge-Kutta or Heun's method.* It is 2nd order accurate. The error is O(h^3).

# An Example

- As a simple example, we try to integrate the following ODE

$$\frac{dy}{dx} = f(x, y) \quad \text{in} \quad x \in [0, 10]$$

- with

$$f(x, y) = -\frac{y}{2} + 4e^{-x/2}\cos(4x)$$

- and initial condition

$$y = 0 \quad \text{at} \quad x = 0$$

- We use both 1st order Euler method and the 2nd order *Heun's method.*

# Fortran Code

```fortran
      program ode
      implicit none
      integer NX, i
      parameter (NX = 50)
      double precision xbeg, ybeg, xend, dx
      double precision x, y, yp, f0, f

      xbeg = 0.0
      ybeg = 0.0
      xend = 10.0
      dx = (xend - xbeg)/NX

c ** initial conditions **

      x = xbeg
      y = ybeg
      write (*,*) x, y, y, exp(-0.5*x)*sin(4.0*x)

c ** solve ODE with Heun method **

      do 10 i = 1, NX
        f0 = f(x,y)
        yp = y + dx*f0
        x  = x + dx
        y  = y + 0.5*dx*(f0 + f(x, yp))
        write (*,*) x, yp, y, exp(-0.5*x)*sin(4.0*x)
10    continue
      end

c **  Your right hand side  **

      double precision function f(x,y)
      double precision x,y
      f = -0.5*y + 4.0*exp(-0.5*x)*cos(4.0*x)
      return
      end
```

# C Code

```c
#include<stdio.h>
#include<math.h>
#define NX   50
double f(double, double);
int main()
{
  int i;
  double xbeg, ybeg, xend, dx;
  double x, y, yp, f0;

  xbeg = ybeg = 0.0;
  xend = 10.0;
  dx = (xend - xbeg)/(double)NX;

/*  Set initial conditions */

  x = xbeg; y = ybeg;
  printf ("%f %f %f %f \n",x,y,y,exp(-0.5*x)*sin(4.0*x));

  for (i = 1; i <= NX; i++){
    f0 = f(x,y);
    yp = y + dx*f0;
    x += dx;
    y += 0.5*dx*(f0 + f(x,yp));
    printf ("%f %f %f %f \n",x,yp,y,exp(-0.5*x)*sin(4.0*x));
  }
  return(0);
}

/* ** Your right hand side ** */

double f(double x, double y)
{
  return (-0.5*y + 4.0*exp(-0.5*x)*cos(4.0*x));
}
```

# An Example

- Analytical solution $$y(x) = e^{-x/2}\sin(4x)$$