

Ordinary Differential Equations - ODEs

Differential eqⁿs are super important in physics,
e.g. Newton's 2nd Law, Schrödinger eqⁿ,
heat eqⁿ etc

ODE - single independent variable (time or position)

PDE - multiple independent variables (time and position, say,
or two spatial variables)

Three main types of ODE's

1) Initial value problem (IVP): time-dependent eqⁿs

2) Boundary value problem: require knowledge of
function

3) Eigenvalue problem: solⁿ exists only for

Systems of ODEs and IVPs

E.g. Consider the simple harmonic oscillator
 $F = ma$

Can rewrite this 2nd order ODE as

$$y_1 \equiv y_2 \equiv$$

$$-\frac{k}{m} x =$$

We get a system of equations

$$\dot{y}_1 =$$
$$\dot{y}_2 =$$

that are solved given initial () position
and velocity

In general, ODE of any order n ()
can be written as a

often written in vector form

$$\text{where } \vec{y} = \left(\begin{array}{c} \\ \\ \end{array} \right) \text{ and } \vec{f} = \left(\begin{array}{c} \\ \\ \end{array} \right)$$

For SHD example

$$y_1 =$$

$$y_2 =$$

$$f_1 =$$

$$f_2 =$$

(Note: independent variable

)

Formally, we can write solⁿ as

$$\vec{y}(t) =$$

Problem is RHS requires \vec{y} for all t , but that's

Euler Method

Consider discretization of indep. var.

$$\Delta t =$$

so that

$$\int_{t_i}^{t_{i+1}} dt' \vec{f}(\vec{y}(t'), t') \approx$$

- equivalent to

then

$$\vec{y}(t_{i+1}) =$$

or

or
just

Can rewrite

which is just the approximation

Example: SHO

$$\begin{aligned} \frac{dx}{dt} &= & \frac{d}{dt} \begin{pmatrix} \\ \end{pmatrix} &= \begin{pmatrix} \\ \end{pmatrix} & \begin{matrix} f_1 = \\ f_2 = \end{matrix} \\ \frac{dv}{dt} &= & \uparrow & & \uparrow \end{aligned}$$

Euler: $x(t+\Delta t) =$
 $v(t+\Delta t) =$

or

$$\begin{aligned} x_{i+1} &= \\ v_{i+1} &= \end{aligned}$$

show spring_euler.cpp results

gnuplot spring.gnu

→ solⁿ not good, as amplitude and energy grow in time

$$E = \frac{1}{2}mv^2 + \frac{1}{2}kx^2 \quad \text{should be}$$

$$\frac{dE}{dt} = \dot{E} =$$

$$\dot{E} = 0 \rightarrow$$

Truncation error - recall

$$\frac{dy_i}{dt} =$$

so for Euler we get

$$y_{i+1} =$$

For $N \approx$ steps to get from initial to final time,

total error is

Midpoint Method or Modified Euler

Recall centred difference $y'(t) = \frac{y(t+\Delta t) - y(t-\Delta t)}{2\Delta t} +$

or $y'(t) =$

or $y'(t + \frac{\Delta t}{2}) =$

solve for
 $y(t+\Delta t) =$

=

★

- gain an order of accuracy if we can evaluate f at the midpoint
- use Taylor expansion of f to estimate midpoint value

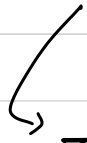
$$y(t+\Delta t) =$$

=

→ Need only a

★ $y(t+\Delta t) =$

try $y(t+\Delta t/2) =$



This is an Euler step

call $\Delta y =$

$f(y(t) + \Delta y/2, t + \Delta t/2) =$

=

=

It works!

Write scheme as

$\Delta y =$
 $y_{i+1} =$

E.g. SHO

$\dot{x} =$
 \dot{v}

} $k/m = 1$ in
spring-midpoint.cpp

Euler

$x_{n+1} =$

$v_{n+1} =$

Midpoint

$x_{n+1} =$

$v_{n+1} =$

Show code

- two extra lines, more stable
- still unstable at large times

where

$x_{mid} =$

$v_{mid} =$