Ordinary Differential Equations - ODEs

Differential eq$^{\underline{n}}$s are super important in physics,
e.g. Newton's 2^{nd} Law, Schrödinger eq$^{\underline{n}}$,
heat eq$^{\underline{n}}$ 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 eq$^{\underline{n}}$s

2) Boundary value problem: require knowledge of function

3) Eigenvalue problem: sol$^{\underline{n}}$ exists only for

Systems of ODEs and IVPs

E.g. Consider the simple harmonic oscillator

$$F = ma$$

Can rewrite this 2^{nd} order ODE as

$y_1 \equiv$ $\qquad\qquad$ $y_2 \equiv$

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

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

where $\vec{y} = \begin{pmatrix} \\ \\ \end{pmatrix}$ and $\vec{f} = \begin{pmatrix} \\ \\ \end{pmatrix}$

For SHO example

$y_1 =$ $f_1 =$

$y_2 =$ $f_2 =$

(Note: independent variable)

Formally, we can write sol$^{\underline{n}}$ 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

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

- equivalent to

then

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

or

or just

Can rewrite

which is just the approximation

Example: SHO

$\frac{dx}{dt} =$

$\frac{dv}{dt} =$

$\frac{d}{dt}\left(\quad \right) = \left(\quad \right)$

$\uparrow \qquad \uparrow$

$f_1 =$

$f_2 =$

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

$v(t+\Delta t) =$

or

$x_{i+1} =$

$v_{i+1} =$

show spring_euler.cpp results gnuplot spring.gnu

→ $\text{sol}^{\underline{n}}$ not good, as amplitude and energy grow in time

$E = \frac{1}{2}mv^2 + \frac{1}{2}kx^2$ 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 $N =$ steps to get from initial to final time,

total error is

Midpoint Method or Modified Euler

Recall centred difference $\dot{y}(t) = \dfrac{y(t+\Delta t) - y(t-\Delta t)}{2\Delta t} +$

or $\dot{y}(t) =$

or $\dot{y}(t+\frac{\Delta t}{2}) =$

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

$=$ ☆

- 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 $\Delta y =$

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

$=$

$=$

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

$x_{n+1} =$

$v_{n+1} =$

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

Midpoint

$x_{n+1} =$

$v_{n+1} =$

where $x_{mid} =$

$v_{mid} =$