Ordinary Differential Equations

Differential equations:

- dominated exact sciences for a long time
- are still the most important tool in physics and other discipline
- describe laws of nature in many areas

  classical mechanics

  electrodynamics

  quantum mechanics

  ....

Ordinary differential equations:
looking for an unknown functional dependence $y(x)$

given only: $y^{(n)} = f\left(x, y, y', y'', \ldots y^{(n-1)}\right)$

rename: $y_1 = y, y_2 = y', \ldots y_n = y^{(n-1)}$

equivalent:

$$\begin{aligned} y_1' &= y_2 \\ y_2' &= y_3 \\ &\ldots \\ &\ldots \\ y_n' &= f(x, y_1, y_2, \ldots, y_{n-1}) \end{aligned}$$

system of first order ODE !

Simplest case: first order, one-dimensional

$$y' = f(x, y)$$

Numerical Integration?

known: initial value $x_o,\, y(x_o) = y_0$

discretization: $x_n = x_o + n\, h$ (step size)

$$y_n \equiv y(x_n),\; y_n' = y'(x_n)$$

iteration from starting value, n=0,1,2,....

Basic idea: small steps using the Taylor expansion

$$y_{n\pm1} = y(x_n \pm h) = y_n \pm h y_n' + \frac{h^2}{2} y_n'' \pm \frac{h^3}{6} y_n''' + \mathcal{O}(h^4)$$

Euler method: $$y_{n+1} = y_n + h\, y_n' = y_n + h\, f(x_n, y_n)$$

error is quadratic in h, accumulates large deviations
from the correct results in practice (see pendulum!)
improvements: smaller h (limited success)
higher order approximations, e.g.
consider intermediate points

example: consider *mid-point* $x_{n+1/2} = x_n + \frac{h}{2}$
"look back- and forward"

$$y_n = y_{n+1/2} - \frac{h}{2} y_{n+1/2}' + \left(\frac{h}{2}\right)^2 \frac{1}{2} y_{n+1/2}'' + \mathcal{O}(h^3)$$

$$y_{n+1} = y_{n+1/2} + \frac{h}{2} y_{n+1/2}' + \left(\frac{h}{2}\right)^2 \frac{1}{2} y_{n+1/2}'' + \mathcal{O}(h^3)$$

$$y_{n+1} = y_n + h\, y_{n+1/2}' + \mathcal{O}(h^3)$$ (!!!)

approximation:

$$y_{n+1} = y_n + h\, y'_{n+1/2} + \mathcal{O}(h^3)$$

$$\begin{aligned} y'_{n+1/2} &= f(x_{n+1/2}, y_{n+1/2}) \\ &= f\left(x_{n+1/2}, y_n + h/2 f(x_n, y_n)\right) + \mathcal{O}(h^2) \end{aligned}$$

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf\left(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right) \\ y_{n+1} &= y_n + k_2 \end{aligned}$$

each step requires two evaluations of f

error is cubic in h !

second order Runge–Kutta method

“The” Runge Kutta method

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf\left(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right) \\ k_3 &= hf\left(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}\right) \\ k_4 &= hf\left(x_n + h, y_n + k_3\right) \\ y_{n+1} &= y_n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} \end{aligned}$$

order $\mathcal{O}(h^5)$

with four ‘function calls’

university of groningen

Matlab: ode45 (and others)

$$
\begin{aligned}
y_1' &= y_2 y_3 & y_1(0) &= 0 \\
y_2' &= -y_1 y_3 & y_2(0) &= 1 \\
y_3' &= -0.51 y_1 y_2 & y_3(0) &= 1
\end{aligned}
$$

example from
>> doc ode45

To simulate this system, create a function `rigid` containing the equations

```
function dy = rigid(t,y)
dy = zeros(3,1);    % a column vector
dy(1) = y(2) * y(3);
dy(2) = -y(1) * y(3);
dy(3) = -0.51 * y(1) * y(2);
```

change tolerances with “odeset”

```
options = odeset('RelTol',1e-4,'AbsTol',[1e-4 1e-4 1e-5]);
[T,Y] = ode45(@rigid,[0 12],[0 1 1],options);
```