ODES : LECTURES 11+12

What is an ODE / System of ODES?

- ODES are equation(s) for continuous function(s) of a single independent variable involving derivatives of the unknown function(s).

Normally written $\underline{y}' = \underline{f}(t, \underline{y})$ $\quad ' = \frac{d}{dt}$

$$\begin{bmatrix} \frac{dy_1}{dt} \\ \vdots \\ \frac{dy_n}{dt} \end{bmatrix} = \begin{bmatrix} f_1(t, y_1, y_2 \dots y_n) \\ \vdots \\ f_n(t, y_1, y_2 \dots y_n) \end{bmatrix}$$

NOTE: ~~they can also be~~ Higher order ODES can always be transformed into equivalent first order systems

EX $\frac{d^2 y}{dx^2} = yt$ Put $y = y_1$, $\frac{dy}{dx} = \frac{dy}{dx} = y_2$ $\Rightarrow \frac{dy_2}{dx} = y_1 t$, $\frac{dy_1}{dx} = y_2$

$$\Rightarrow \begin{bmatrix} y_1' \\ y_2' \end{bmatrix} = \begin{bmatrix} y_2 \\ y_1 t \end{bmatrix}$$

An ODE/Sys. of ODEs does not determine a unique solution (it only specifies y'). Usually, there are infinitely many solns.

To single out a particular soln we also have to specify a value, i.e. give (a) boundary condition(s).

The type of boundary conditions needed/desired define the problem as either:

1) An initial value problem (IVP)

or 2) A boundary value problem (BVP)

EX (IVP) $\quad y' = -y \quad$ with $y(t_0) = y_0$

$y'' = y^3 t \quad$ with $y(t_0) = y_0 \quad y'(t_0) = d_0$

EX (BVP) $\quad y'' = y^3 t \quad$ with $y(t_0) = y_0 \quad y(t_1) = y_1$

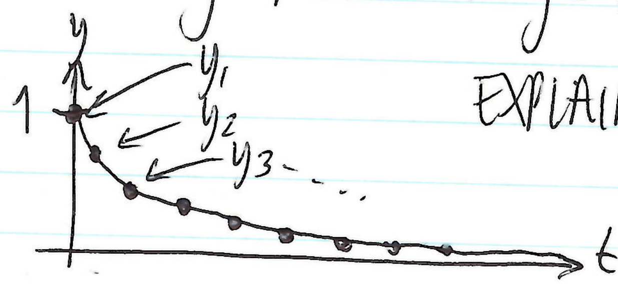
$y'' = -y^2 \quad$ with $y(t_0) = y_0 \quad y'(t_1) = d_1$

Initial Value problems

$$y' = \underline{f}(t, y) \quad \text{with} \quad \underline{y}(t_0) = y_0$$

A numerical soln. to this problem consists of finding approximate values of the soln at discrete points in time.

<u>EX</u> $y' = -y$ with $y(0) = 1 \Rightarrow y = \exp(-t)$.



EXPLAIN: h_k, t_k notation.

The central question is: how should we <u>accurately</u> and <u>quickly</u> compute these approximate values?

A: Discretise the derivative operator!

Different discretisations of $\frac{d}{dt}$ give rise to different <u>time-stepping schemes</u> with different properties.

Stable and unstable ODEs

If the members of a family move away from one another as t increases $\rightarrow$ unstable

—— " —— closer —"— $\rightarrow$ stable.

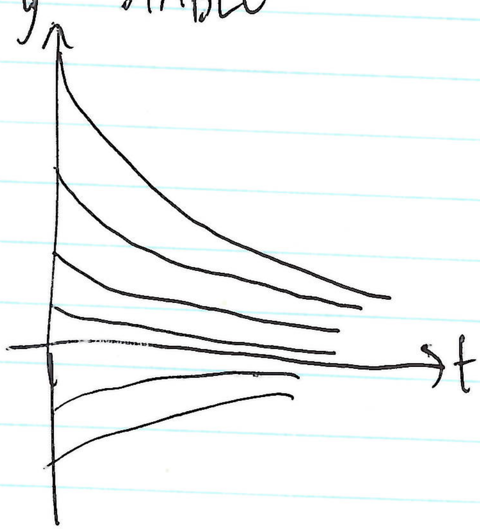
If neither $\rightarrow$ neutrally stable.

More precisely the system $\underline{y}' = \underline{f}(t, \underline{y})$
is unstable if $J_{ij} = \frac{\partial f_i}{\partial x_j}$ has any eigvals > 0
stable if —— " —— has only eigvals < 0

EX

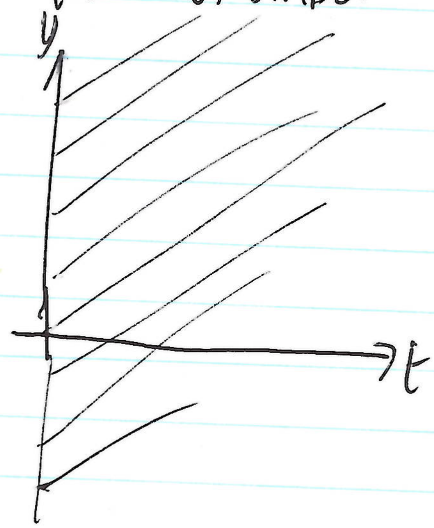
$y' = -y$

STABLE



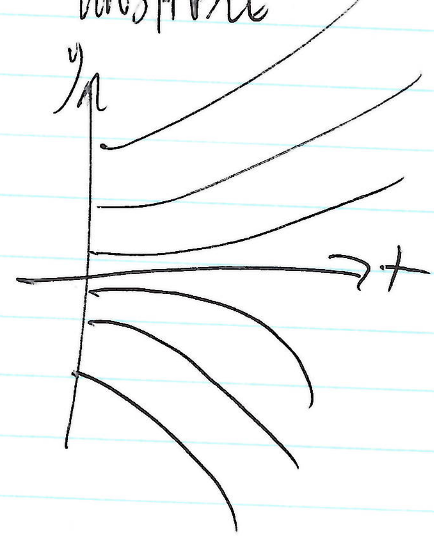
$y' = 1$

NEUTRALLY STABLE



$y' = y$

UNSTABLE



Euler's Method

$y(t+h)$ - is the object we wish to find given $y(t)$.

Expand $y(t+h)$ as a Taylor Series about $y(t) \Rightarrow$

$$y(t+h) = y(t) + y'(t)h + \frac{y''(t)}{2}h^2 + O(h^3).$$

If we truncate at $O(h^2)$ we have the following approximation:

$$y'(t) \approx \frac{y(t+h) - y(t)}{h}$$

In the discrete setting

$$\boxed{y_{i+1} = y_i + h_i f(t_i, y_i).}$$

NOTE: In general $y_i \neq y|_{t=t_i}$ ← The method is approximate.

Accuracy and Stability

Like all other numerical procedures ~~[illegible]~~ time stepping methods suffer from 2 sources of error

1) Rounding
2) Truncation.

We can't do much about 1), 2) we can!

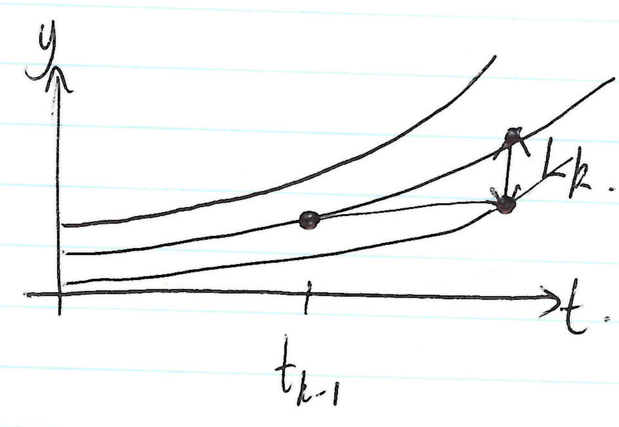
Truncation errors

(a) Local truncation error. (the error made in making one time step).

$$L_k = y_k - u_{k-1}(t_k).$$

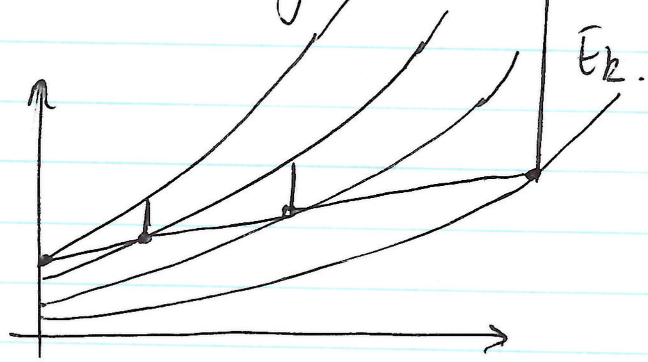
y_k: the computed soln at time t_k.

$u_{k-1}(t_k)$: the real (exact) solution curve that passes through (y_{k-1}, t_{k-1}).

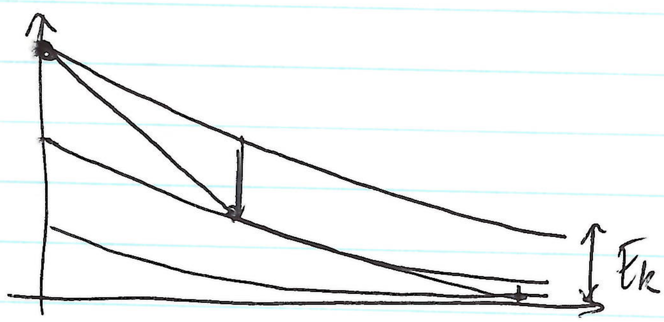


(b) Global truncation error:- Difference between computed soln. and true soln. determined by initial data.

$$E_k = y_k - u_0(t_k)$$



NOTE: $E_k \neq \sum_{i=1}^{k} L_i$



Having a small E_k is what we really care about. However all we can control is L_k.

<u>Accuracy</u>: A scheme is said to be of order p if

$$L_k = O(h_k^{p+1}).$$

EX What is the accuracy of forward Euler?

$$y(t+h) = y(t) + y'(t)h + O(h^2)$$
$$= y(t) + f(y,t)h + O(h^2).$$

Set $t = t_k$ and $h = h_k$ $\Rightarrow$

$$y(t_{k+1}) = y(t_k) + f(y(t_k), t_k)\, h_k + O(h_k^2). \quad ①$$

Recall Euler's forward step

$$y_{k+1} = y_k + h_k f(t_k, y_k) \quad ②$$

Look at the difference between ① and ② $\Rightarrow$

$$②-① \Rightarrow \underbrace{y_{k+1} - y(t_{k+1})}_{E_k} = [y_k - y(t_k)] + h_k[f(y_k,t_k) - f(y(t_k),t_k)] + O(h_k^2).$$

If we had $y_k = y(t_k)$ and

$$f(y_k,t_k) = f(y(t_k),t_k)$$

we get the local error $L_k = O(h_k^2)$.

Thus Euler Forward is "first order".

Stability of a num. meth.

Distinct from the concept of stability of the ODE being solved!

A method is stable if small perturbations lead to small changes in the computed solns.

Unstable if small ~~change~~ perturbation leads to big changes.

~~Ex~~ One way to assess stability is to look at the model equation $y' = \lambda y$ with $y(0) = y_0$

$$\Rightarrow \quad y = y_0 \exp(-\lambda t).$$

Ex: Stability for forward Euler.

If we choose a fixed step-size h, Euler forward says

$$y_{k+1} = y_k + y_k \lambda h.$$
$$= y_k (1 + \lambda h).$$

Thus

$$y_k = (1 + \lambda h)^k y_0$$

The exact soln $\searrow 0$ as $t \nearrow \infty$. As does the numerical method, provided $\lambda < 0$. ~~Thus forward Euler~~ ~~is stable if also~~

This says that the errors do not grow provided

$$|1+\lambda h| < 1 \Rightarrow \underbrace{\lambda h \in (0,-2)}.$$

$\lambda h < -2$

$h > -2/\lambda$.

In this interval F Euler stable otherwise unstable.

Unstable if $\lambda > 0$ ← the equation itself is unstable

Unstable if $h > \frac{-2}{\lambda}$ ← the equation is stable, but the time step too large. The integration is unstable.

Backward Euler

$$y_{k+1} = y_k + f(t_{k+1}, y_{k+1}) h_k.$$

Very similar to F Euler, but one important difference. ~~We want to evaluate $f(y_{k+1}, t_{k+1})$ before we know y_{k+1}~~

Can no longer simply evaluate the RHS. In fact, this is a system of equations to solve for y_{k+1}.

This makes Backward E. <u>implicit</u> whereas Forward E. <u>explicit</u>.