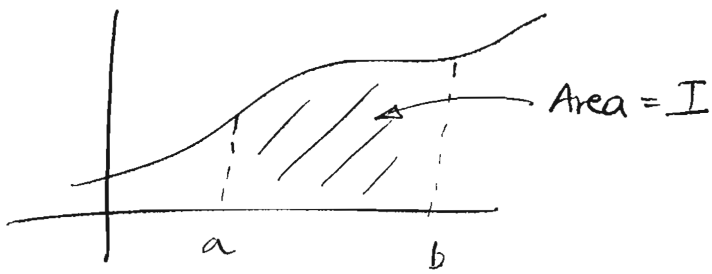
Numerical integration

We seek an algorithm to approximate the definite integral:

$$I = \int_a^b f(x)\,dx$$

(or, the area below the graph of $y=f(x)$)



Of course, in the fortuitous case where we know a function $F(x)$ (the anti-derivative of f), s.t. $F'(x) = f(x)$,

we can write:

$$\int_a^b f(x)\,dx = F(b) - F(a)$$

e.g $[\arctan(x)]' = \frac{1}{1+x^2}$, thus

$$\int_a^b \frac{dx}{1+x^2} = \arctan(b) - \arctan(a).$$

However, this is not a practical algorithm, since:

- The anti-derivative is not generally known.
- Often, the anti-derivative may be significantly more expensive to evaluate than $f(x)$ itself, (e.g. compare $f(x) = \frac{1}{1+x^2}$ (easy) with $F(x) = \arctan(x)$ (expensive)).

General methodology

- Subdivide the interval of integration using the $n+1$ points $\{x_i\}_{i=0}^{n}$, with

$$a = x_0 < x_1 < x_2 < \dots < x_{n-1} < x_n = b$$

- In each interval $[x_i, x_{i+1}]$ approximate $f(x)$ with some simpler function, say a polynomial $p^{(i)}(x)$, which is easy to integrate. Approximate

$$I_i = \int_{x_i}^{x_{i+1}} f(x)\,dx \approx \int_{x_i}^{x_{i+1}} p^{(i)}(x)\,dx.$$

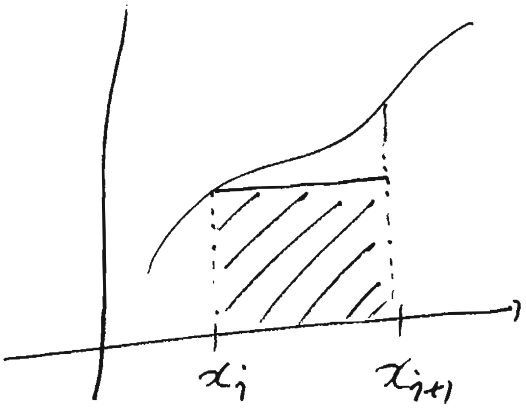
- Compute the integral $I = \int_a^b f(x)\,dx$

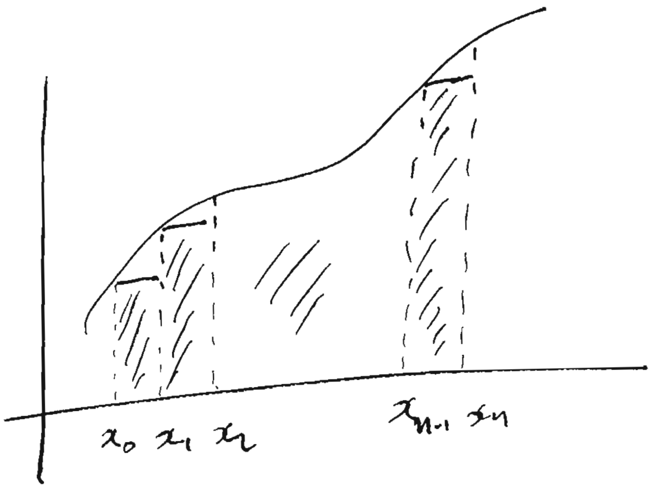
as $$I = \sum_{i=0}^{n-1} I_i \approx \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} p^{(i)}(x)\,dx.$$

Example The rectangle rule

At each interval $[x_i, x_{i+1}]$ use the approximation

$p^{(i)}(x) = f(x_i)$ (the left endpoint!)





Thus we approximate:

$$I_i = \int_{x_i}^{x_{i+1}} f(x)\,dx \approx \int_{x_i}^{x_{i+1}} f(x_i)\,dx = f(x_i)\,(x_{i+1} - x_i)$$

(We often present this rule on a single interval $[a,b]$, as

$$\int_a^b f(x)\,dx \approx f(a)\cdot(b-a)$$
)

In the case where $x_{i+1} - x_i = h = \text{const}$, we can write

$$I = \int_a^b f(x)dx = \sum_{i=0}^{n-1} I_i \cong \sum_{i=0}^{n-1} f(x_i) \cdot h \Rightarrow$$

$$\boxed{I \cong \frac{b-a}{n} \sum_{i=0}^{n-1} f(x_i)}$$

As in the case of interpolation, we can assess the error incurred by this approximation. There are 2 errors we actually focus on:

→ The local error $\left| \int_{x_i}^{x_{i+1}} (f(x) - p^{(i)}(x))\, dx \right|$ at each subinterval

→ The global error for the entire integral $\int_a^b f(x)\, dx$.

Numerical Integration

Our objective is to design an algorithm which produces an approximation of the definite integral

$$I = \int_a^b f(x)\,dx.$$

The reasons why an approximation would be sought instead of an analytic computation, include:

→ The anti-derivative of f may not be expressible using fundamental functions, or

→ An anti-derivative may be too expensive to evaluate.

The general methodology for such an approximation is:

→ Introduce points $[x_k]_{k=0}^{N}$ such that

$$a = x_0 < x_1 < x_2 < \cdots < x_{N-1} < x_N = b.$$

→ In every interval $[x_k, x_{k+1}]$, approximate $f(x)$ with a "simpler" function $p^{(k)}(x)$ which is trivial to integrate.

Approximate $I_k = \int_{x_k}^{x_{k+1}} f(x)\,dx \approx \int_{x_k}^{x_{k+1}} p^{(k)}(x)\,dx.$

After I_k has been approximated, an approximation for the entire $I = \int_a^b f(x)dx = \sum_k I_k \approx \sum_k \int_{x_k}^{x_{k+1}} p^{(k)}(x)dx$

is assembled.

Next, we will see certain popular choices for $p^{(k)}(x)$

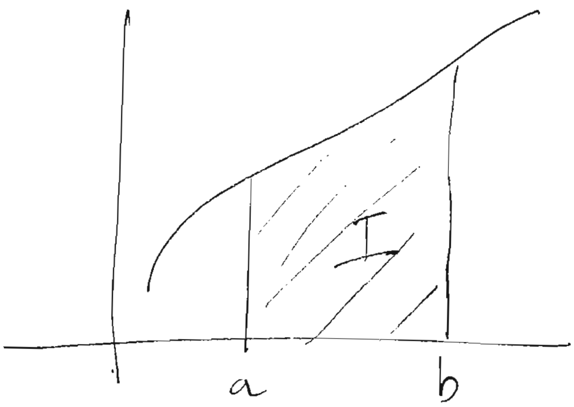
I. The rectangle rule

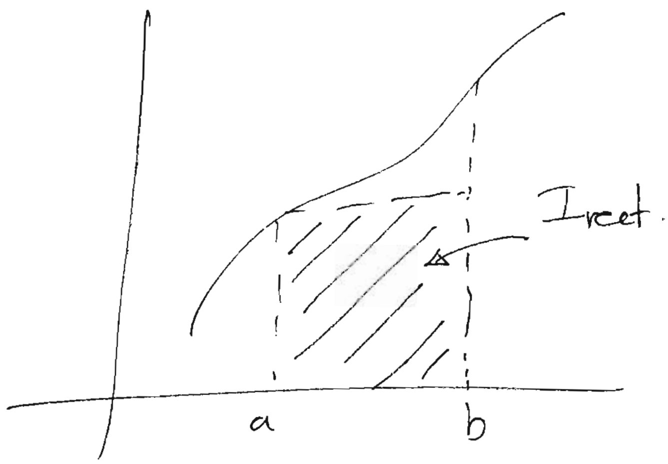
[First, we consider the entire interval $[a,b]$, without any partitioning, for simplicity of exposition].

The rectangle rule approximates $f(x) \approx f(a)$ (in $[a,b]$)

Thus $I = \int_a^b f(x)dx \approx \underbrace{\int_a^b f(a)\,dx}_{:= I_{rect}} = (b-a)\,f(a)$

Graphically:





In order to design a "composite" rectangle rule, we use the partition $a = x_0 < x_1 < \cdots < x_{N-1} < x_N = b$, and use the previous rule to approximate

$$I_k = \int_{x_k}^{x_{k+1}} f(x)\,dx \approx \underbrace{(x_{k+1} - x_k)\, f(x_k)}_{:= I_{k,rect}} = h_k\, f(x_k)$$

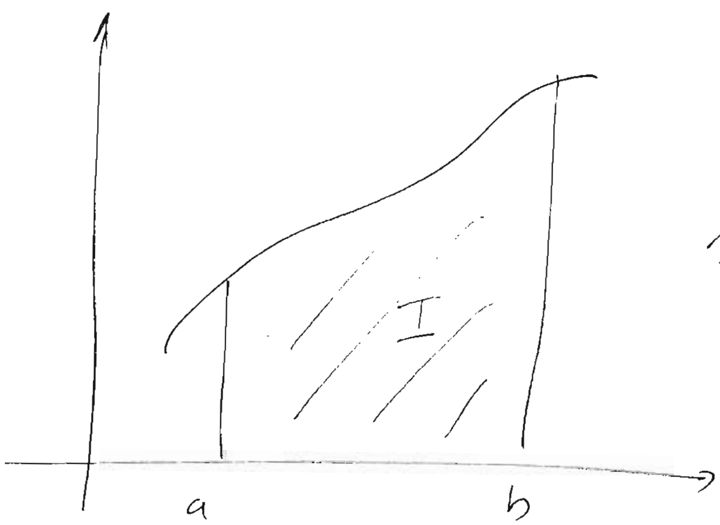
(where we defined $h_k := x_{k+1} - x_k$).

In the case where $h_0 = h_1 = h_2 = \cdots = h_{N-1} = h$ = const, we have

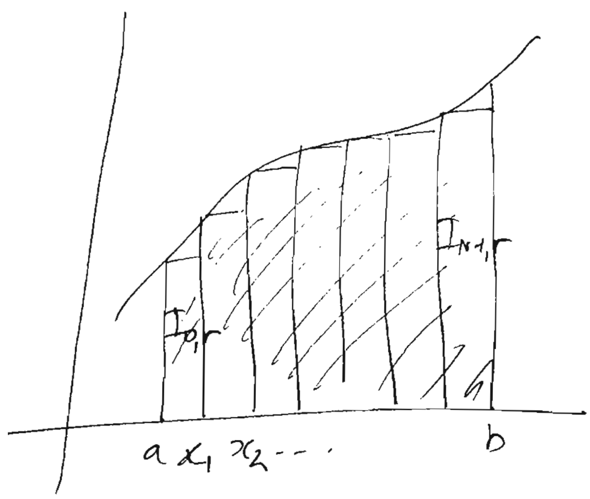
$$I = \int_a^b f(x)\,dx \approx \sum_{k=0}^{N-1} I_{k,rect} =$$

$$= \sum_{k=0}^{N-1} h \cdot f(x_k) = \underbrace{\frac{b-a}{N}}_{=h} \sum_{k=0}^{N-1} f(x_k)$$

Graphically:



$\approx$



As in the case of interpolation, we want 4/12/11 L

to be conscious about the error involved in this approximation. This, in fact, comes in 2 flavors:

→ The local error, is defined in each subinterval as:

$$e_k = \left| I_{k,\text{rule}} - I_{k,\text{analytic}} \right|.$$

For the rectangle rule, we have:

$$e_k = \left| \int_{x_k}^{x_{k+1}} f(x_k)\,dx - \int_{x_k}^{x_{k+1}} f(x)\,dx \right|$$

$$= \left| \int_{x_k}^{x_{k+1}} \left[f(x_k) - f(x) \right] dx \right| \qquad (1)$$

We shall seek to obtain an upper bound for the integral in eqn (1). Let us remember Taylor's formula, applied to $f(x)$ in the vicinity of x_k:

$$f(x) = f(x_k) + f'(c_k)(x - x_k), \quad \text{where } c_k \in (x_k, x_{k+1})$$

Thus

4/12/11

$$e_K = \left| - \int_{x_K}^{x_{K+1}} f'(c_K)(x-x_K)\,dx \right|$$

$$\leq \int_{x_K}^{x_{K+1}} |f'(c_K)|\,|x-x_K|\,dx$$

$$\leq \int_{x_K}^{x_{K+1}} \|f'\|_\infty \underbrace{|x-x_K|}_{\geq 0}\,dx = \|f'\|_\infty \int_{x_K}^{x_{K+1}} (x-x_K)\,dx$$

$$= \|f'\|_\infty \left[\frac{(x-x_K)^2}{2}\right]_{x_K}^{x_{K+1}} = \frac{1}{2}\|f'\|_\infty \frac{(x_{K+1}-x_K)^2}{2}$$

$$\Rightarrow \boxed{e_K \leq \frac{1}{2}\|f'\|_\infty \cdot h_K^2}$$

→ The global error is defined as:

$$e = |I_{rule} - I_{analytic}| = \left| \sum_{k=0}^{N-1} [I_{K,rule} - I_{K,analytic}] \right|$$

$$\leq \sum_{k=0}^{N-1} |I_{K,rule} - I_{K,analytic}| = \sum_{k=0}^{N-1} e_K$$

For example, if $h = \text{const}$, for the rectangle rule we have:

$$e \leq \sum_{K=0}^{N-1} e_K = N \cdot \frac{1}{2}\|f'\|_\infty \cdot h^2 \underset{Nh=b-a}{\Longrightarrow} \boxed{e_{global} \leq \frac{b-a}{2}\|f'\|_\infty h}$$

What we observe is that, for the rectangle rule:

Local error $= O(h^2)$

Global error $= O(h)$

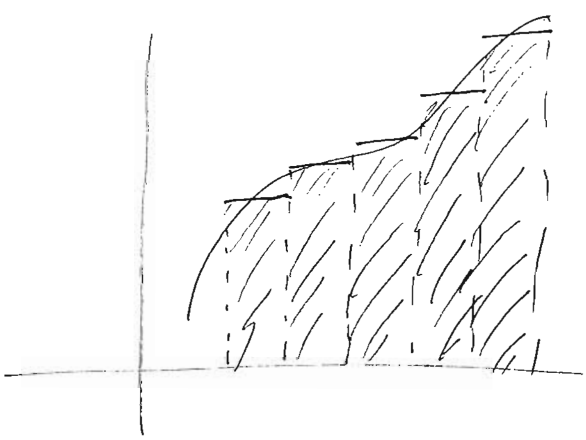
In general, we always get that if the local error is $O(h^{d+1})$ the global will be $O(h^d)$; additionally, in this case the numerical integration rule is called d-order accurate (e.g. rectangle rule is 1st order accurate)

II. Midpoint rule: $f(x) \approx f\left(\frac{x_k + x_{k+1}}{2}\right)$

$$I = \int_a^b f(x)\,dx \approx (b-a)\, f\left(\frac{a+b}{2}\right).$$

Composite rule [h = constant]

$$I = \int_a^b f(x)\,dx \approx \sum_{k=0}^{N-1} \underbrace{(x_{k+1} - x_k)}_{= h = \frac{b-a}{N}} f\left(\frac{x_k + x_{k+1}}{2}\right) = \frac{b-a}{N} \sum_{k=0}^{N-1} f\left(\frac{x_k + x_{k+1}}{2}\right)$$



Local error analysis :

We use the (2nd order) Taylor's formula around the point $x_m = \frac{x_K + x_{K+1}}{2}$

$$f(x) = f(x_m) + f'(x_m)(x - x_m) + \frac{f''(c_K)}{2}(x - x_m)$$

$$c_K \in [x_K, x_{K+1}].$$

$$e_K = \left| \int_{x_K}^{x_{K+1}} [f(x_m) - f(x)]\, dx \right| =$$

$$= \left| f'(x_m) \int_{x_K}^{x_{K+1}} (x - x_m)\, dx + \int_{x_K}^{x_{K+1}} \frac{f''(c_K)}{2} (x - x_m)^2\, dx \right|$$

Note that $\int_{x_K}^{x_{K+1}} (x - x_m)\, dx = \left. \frac{(x - x_m)^2}{2} \right|_{x_K}^{x_{K+1}} = \frac{h_K^2}{8} - \frac{h_K^2}{8} = 0$

Thus :

$$e_K = \left| \frac{1}{2} \int_{x_K}^{x_{K+1}} f''(c_K)(x - x_m)^2\, dx \right| \le \frac{1}{2} \int_{x_K}^{x_{K+1}} |f''(c_K)|\, |x - x_m|^2\, dx$$

$$\le \frac{1}{2} \| f'' \|_\infty \int_{x_K}^{x_{K+1}} (x - x_m)^2\, dx = \frac{1}{2} \| f'' \|_\infty \left[\frac{(x - x_m)^3}{3} \right]_{x_K}^{x_{K+1}} =$$

$$= \frac{1}{2} \| f'' \|_\infty \left(\frac{h_k^3}{24} + \frac{h_k^3}{24} \right) \Rightarrow \boxed{e_k \leq \frac{1}{24} \| f'' \|_\infty h_k^3}$$

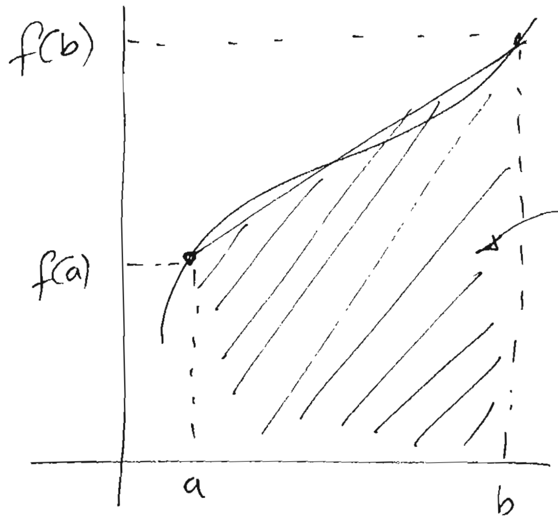
Global error :

$$e_{global} \leq \sum_{k=0}^{N-1} e_k \Rightarrow \boxed{e_k \leq \frac{b-a}{24} \| f'' \|_\infty \cdot h^2}$$

Thus, the midpoint rule is <u>2nd</u> order accurate

Trapezoidal rule

In this case f is approximated in $[a,b]$ with the straight line drawn between $(a, f(a))$ and $(b, f(b))$.



$$I_{trap} = \frac{f(a)+f(b)}{2} \cdot (b-a)$$

Thus in this case $I = \int_a^b f(x)dx \approx \underbrace{(b-a)\frac{f(a)+f(b)}{2}}_{=I_{trap}}$

To generate the corresponding composite rule, we write:

$$I_k = \int_{x_k}^{x_{k+1}} f(x)dx \approx (x_{k+1}-x_k)\frac{f(x_k)+f(x_{k+1})}{2} = h_k \frac{f(x_k)+f(x_{k+1})}{2} = I_{k,\text{trap}}$$

Thus $I = \sum_{k=0}^{N-1} I_k \approx \sum_{k=0}^{N-1} I_{k,\text{trap}} \overset{(h=\text{const})}{=} \sum_{k=0}^{N-1} h \frac{f(x_k)+f(x_{k+1})}{2}$

$$= \frac{b-a}{2N}\left[f(x_0)+2f(x_1)+2f(x_2)+\dots+2f(x_{N-2})+2f(x_{N-1})+f(x_N)\right]$$

Note that due to the simple formula for the trapezoidal area, we did not have to write the approximating polynomial $p(x) = f(a) + \frac{f(b)-f(a)}{b-a}(x-a)$ explicitly. Also, the result of integrating $\int_a^b p(x)\,dx$ results in a very simple formula $\left[(b-a)\frac{f(a)+f(b)}{2}\right]$, even "simpler" than the formula for p itself!

Local error analysis:

Estimating the local error can be somewhat delicate with the trapezoidal rule ... we will in this case use a formula from the theory of interpolating polynomials we saw before:

Thm If $p(x)$ is a n-degree polynomial, interpolating $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$, then for every $x \in [x_0, x_n]$ we have

$$f(x) - p(x) = \frac{f^{(n+1)}(c)}{(n+1)!}(x-x_0)(x-x_1)\cdots(x-x_n)$$

CAUTION c is not a constant; it depends on the particular x we chose in this theorem.

For the trapezoidal rule, we effectively use a linear $(n=1)$ interpolant, thus when $x \in [x_k, x_{k+1}]$:

$$f(x) - p^k(x) = \frac{f''(c_k)}{2}(x-x_k)(x-x_{k+1})$$

where $c_k \in (x_k, x_{k+1})$

$$e_k = \left| \int_{x_k}^{x_{k+1}} [f(x) - p^{(k)}(x)]\, dx \right| =$$

$$= \left| \int_{x_k}^{x_{k+1}} \frac{f''(c_k)}{2}(x-x_k)(x-x_{k+1})\, dx \right|$$

$$\leq \int_{x_k}^{x_{k+1}} \left| \frac{f''(c_k)}{2} \right| \left| (x-x_k)(x-x_{k+1}) \right| dx$$

$$\leq \frac{1}{2} \| f''(x) \|_\infty \int_{x_k}^{x_{k+1}} \left| (x-x_k)(x-x_{k+1}) \right| dx$$

The only reason we can meaningfully continue at this point, is to recognize that $(x-x_k)(x-x_{k+1}) \leq 0$ in $[x_k, x_{k+1}]$ thus $|(x-x_k)(x-x_{k+1})| = -(x-x_k)(x-x_{k+1})$ and remove in this way the absolute value in the integral above.

This is <u>not</u> the case in general for higher-order polynomial interpolants, where we won't be able to remove the absolute value. (see Simpson's rule next).

We can verify that: 4/14/11

$$\int_{x_K}^{x_{K+1}} |(x-x_K)(x-x_{K+1})| = -\int_{x_K}^{x_{K+1}} (x-x_K)(x-x_{K+1}) = \frac{h_K^3}{6}$$

Putting everything together:

$$e_K \leq \frac{1}{2} \|f''\|_\infty \cdot \frac{h_K^3}{6} \Rightarrow \boxed{e_K \leq \frac{1}{12} \|f''\|_\infty \cdot h_K^3}$$

For the global error:

$$e \leq \sum_{K=0}^{N-1} e_K \leq \frac{N}{12} \|f''\|_\infty h_K^3 \xrightarrow{Nh=b-a} \boxed{e \leq \frac{b-a}{12} \|f''\| h^2}$$

Thus, trapezoidal rule is also 2nd order accurate.

SIMPSON's rule This is a slightly more complicated algorithm, but the accuracy gains are so attractive that it has become somewhat of a golden standard for numerical integration.

It is based on (piecewise) quadratic interpolation.

Specifically:

Consider 3 equally spaced x-values

x_1, $x_2 = x_1 + h$, $x_3 = x_1 + 2h$

with associated y-values $y_i = f(x_i)$ $i = 1,2,3$.

We will approximate $f(x)$ in $[x_1, x_3]$ with a <u>quadratic</u> $p(x) = c_2 x^2 + c_1 x + c_0$ that interpolates the 3 data points: $(x_1, y_1), (x_2, y_2), (x_3, y_3)$.

Using Lagrange interpolation:

$$l_1(x) = \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)} = \frac{(x-x_2)(x-x_3)}{(-h)(-2h)} = \frac{1}{2h^2}(x-x_2)(x-x_3)$$

$$l_2(x) = \frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)} = -\frac{1}{h^2}(x-x_1)(x-x_3)$$

$$l_3(x) = \frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)} = \frac{1}{2h^2}(x-x_1)(x-x_2)$$

And $p(x) = y_1 l_1(x) + y_2 l_2(x) + y_3 l_3(x)$

We then proceed to approximate

$$\underbrace{\int_{x_1}^{x_3} f(x)\,dx}_{I} \approx \underbrace{\int_{x_1}^{x_3} p(x)\,dx}_{I_{simp.}} = \sum_{i=1}^{3} y_i \int_{x_1}^{x_3} l_i(x)\,dx$$

After some easy (yet tedious) analytic integration using the previous formulas, we get:

$$\int_{x_1}^{x_3} l_1(x)\,dx = \frac{h}{3}, \quad \int_{x_1}^{x_3} l_2(x)\,dx = \frac{4h}{3}, \quad \int_{x_1}^{x_3} l_3(x)\,dx = \frac{h}{3}$$

Thus $$\int_{x_1}^{x_3} p(x)\,dx = \frac{h}{3}\left[f(x_1) + 4f(x_2) + f(x_3)\right].$$

This is Simpson's rule, and is commonly written as:

$$\int_a^b f(x)\,dx \approx \frac{\overbrace{b-a}^{=2h}}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right].$$

In order to define the respective composite rule, we use a partitioning:

$$a = x_0 < x_1 < x_2 < \cdots\cdots < x_{2N-1} < x_{2N} = b$$

this time we define each interval $D_k = [x_{2k}, x_{2(k+1)}]$, and

$$I_k = \int_{x_{2k}}^{x_{2k+2}} f(x)\,dx \quad \Rightarrow \quad I = \sum_{k=0}^{N-1} I_k.$$

Then: $I_{k,simp} = \frac{h}{3}\left[f(x_{2k}) + 4f(x_{2k+1}) + f(x_{2k+2})\right]$

and the composite rule $I_{simp} = \sum_{k=0}^{N-1} I_{k,simp}$ becomes:

$$I_{simp} = \frac{h}{3}\Big[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + 2f(x_4) + \cdots$$

$$\cdots + 2f(x_{2N-4}) + 4f(x_{2N-3}) + 2f(x_{2N-2}) + 4f(x_{2N-1}) + f(x_{2N})\Big]$$

In order to estimate the local error we again try to use the formula for the interpolation error of passing a quadratic $p^{(k)}(x)$ through $(x_{2k}, f(x_{2k}))$, $(x_{2k+1}, f(x_{2k+1}))$ and $(x_{2k+2}, f(x_{2k+2}))$.

$$f(x)-p^{(k)}(x) = \frac{f'''(c_k)}{3!}(x-x_{2k})(x-x_{2k+1})(x-x_{2k+2})$$

thus

$$e_k = \left| \int_{x_{2k}}^{x_{2k+2}} [f(x)-p(x)]\,dx \right|$$

$$\le \int_{x_{2k}}^{x_{2k+2}} \frac{|f'''(c_k)|}{3!} |(x-x_{2k})(x-x_{2k+1})(x-x_{2k+2})|\,dx$$

$$\le \frac{1}{6} \|f'''(x)\|_\infty \int_{x_{2k}}^{x_{2k+2}} \underbrace{|(x-x_{2k})(x-x_{2k+1})(x-x_{2k+2})|}_{(*)}\,dx$$

And with this we're at a dead end! We cannot simply remove the absolute value in the expression $(*)$ since it changes sign in $[x_{2k}, x_{2k+2}]$. Even if we break up this integral in sub-intervals, we will at best show that Simpson's rule is 3rd order accurate, where in fact it is even more, i.e. 4th order accurate!

To achieve our goal, we will use a different (and more general) type of analysis:

It is possible to show that:

<u>Thm</u> If an integration rule integrates <u>exactly</u> any polynomial up to degree $(d-1)$ then the global error is $O(h^d)$ or better, i.e. the rule is at least d-order accurate.

<u>Methodology</u> We will test Simpson's rule on monomials $f(x) = x^d$, $d = 0, 1, 2, \ldots$

- $f(x) = 1$: $I_{simp} = \frac{b-a}{6}[1+4+1] = (b-a) \equiv \int_a^b 1 \cdot dx$

CORRECT!

- $f(x) = x$: $I_{simp} = \frac{b-a}{6}\left[a+4\left(\frac{a+b}{2}\right)+b\right] =$

$$= \frac{b^2}{2} - \frac{a^2}{2} \equiv \int_a^b x\,dx \quad \text{CORRECT!}$$

- $f(x) = x^2$: $I_{simp} = \frac{b-a}{6}\left[a^2 + 4\left(\frac{a+b}{2}\right)^2 + b^2\right] =$

$$= \frac{b^3}{3} - \frac{a^3}{3} \equiv \int_a^b x^2\,dx \quad \text{CORRECT!}$$

- $f(x) = x^3$: $I_{simp} = \frac{b-a}{6}\left[a^3 + 4\left(\frac{a+b}{2}\right)^3 + b^3\right] =$

$$= \frac{b^4}{4} - \frac{a^4}{4} \equiv \int_a^b x^3\,dx \quad \text{CORRECT!}$$

$f(x) = x^4$: $I_{simp} = \frac{b-a}{6}\left[a^4 + 4\left(\frac{a+b}{2}\right)^4 + b^4\right]$

which is <u>not</u> equal to $\frac{b^5}{5} - \frac{a^5}{5} = \int_a^b x^4\,dx$

Thus Simpson's rule is <u>4th order accurate</u>

i.e. $e_{k,local} \leq O(h^5)$

$e_{global} \leq O(h^4)$

Assessing order of accuracy in integration rules

4/19/11 L1

<u>Theorem</u> If an integration rule integrates <u>exactly</u> any polynomial up to degree $(d-1)$, then the global error is $O(h^d)$ or better, i.e. the rule is at least d-order accurate.

Methodology

* Test the integration rule on <u>monomials</u> of degree $0, 1, 2, \ldots$, i.e. on $f(x) = 1$, $f(x) = x$, $f(x) = x^2$, ...
* If $f(x) = x^d$ is the 1st test function that is <u>not</u> integrated exactly, the order of accuracy is equal to <u>d</u>

<u>Example</u> Trapezoidal rule $\quad I = \int_a^b f(x)\,dx \approx \frac{f(a)+f(b)}{2}(b-a)$

$f(x) = 1$: $I_{trap} = \frac{1+1}{2} \cdot (b-a) = b - a \quad$ (= exact)

$f(x) = x$: $I_{trap} = \frac{a+b}{2}(b-a) = \frac{b^2}{2} - \frac{a^2}{2} \quad$ (= exact)

$f(x) = x^2$ $\quad I_{trap} = \frac{a^2+b^2}{2}(b-a)$ <u>not exact</u> $\quad \left(\int_a^b x^2 dx = \frac{b^3}{3} - \frac{a^3}{3} \right)$

Thus, rule is <u>2nd</u> order accurate.

Initial value problems for 1st order differential equations

In this last part of our class we will turn our attention to differential equation problems, of the form:

Find the function $y(t) : [t_0, +\infty) \to \mathbb{R}$ that satisfies the "ordinary differential equation (ODE)":

$$y'(t) = f(t, y(t)) \quad \text{(for a certain function } f\text{)}$$

and $y(t_0) = y_0$ (This is called an initial value problem)

example

→ The velocity of a vehicle over the time interval $[0,5]$ satisfies $v(t) = t(t+1)$. At time $t=0$, the vehicle starts from position $x(0) = 5$. What is $x(t)$, $t \in [0,5]$?

Ans Given by IVP
$$x'(t) = t(t+1)$$
$$x(0) = 5$$

(Since $x'(t) = v(t)$)

ex 2: The concentration $y(t)$ of a chemical species is given by:

$$y'(t) = y\,(t^2+1)$$

$$y(0) = 1$$

(Here $f(t,y) = y\,(t^2+1)$).

Of course in certain cases we can solve this differential equation exactly, e.g. in the last example:

$$y'(t) = y\,(t^2+1) \Rightarrow \frac{y'}{y} = t^2+1$$

$$\Rightarrow \int_{t_0=0}^{t} \frac{y'}{y}\,d\tau = \int_{t_0=0}^{t} (\tau^2+1)\,d\tau \Rightarrow$$

$$\Rightarrow [\ln y]_{t_0=0}^{t} = \left[\frac{\tau^3}{3}+\tau\right]_0^t \Rightarrow \ln y(t) - \underbrace{\cancel{\ln y(0)}}_{=0} = \frac{t^3}{3}+t$$

$$\Rightarrow y(t) = e^{\left(\frac{t^3}{3}+t\right)}$$

However we do not want to depend in our ability to solve the O.D.E. exactly, since: 4/19/11

→ An exact solution may not be analytically expressible in closed form

→ The exact solution may be too complicated and (very important:)

→ The function $f(t,y)$ may not be available as a formula; e.g. it could result from a black-box computer program.

Solution: APPROXIMATE the solution to the differential equation

General methodology ("1-step methods")

- Consider discrete points in time
  $t_0 < t_1 < t_2 < \dots < t_k < \dots$
  If we set $\Delta t_k = t_{k+1} - t_k$ and $\Delta t_k = \Delta t = \text{const}$,
  then $t_k = t_0 + k \cdot \Delta t$.
- Use the notation $y_k = y(t_k)$.

- Use the values t_k, y_k AND the ODE.
$y'(t) = f(t, y)$ to approximate y_{k+1}.

<u>Method</u>:

$$y'(t) = f(t,y)$$

$$\Rightarrow \int_{t_k}^{t_{k+1}} y'(\tau)\, d\tau = \int_{t_k}^{t_{k+1}} f(\tau, y)\, d\tau$$

$$\Rightarrow \underbrace{y(t_{k+1})}_{:= y_{k+1}} - \underbrace{y(t_k)}_{:= y_k} = \underbrace{\int_{t_k}^{t_{k+1}} f(\tau, y)\, d\tau}_{\text{Approximate using integration rule!}}$$

Thus

$$\begin{matrix} y_0 & \longrightarrow & y_1 & \longrightarrow & y_2 & \longrightarrow & y_3 \cdots \\ t_0 & & t_1 & & t_2 & & \end{matrix}$$

For example, if we approximate this integral with the rectangle rule $\int_a^b f dx \cong f(a)(b-a)$, we get 4/19/11

$$y_{k+1} - y_k = \int_{t_k}^{t_{k+1}} f(\tau, y)\, d\tau \approx f(t, y)(t_{k+1} - t_k)$$

$$\Rightarrow \boxed{y_{k+1} = y_k + \Delta t f(t_k, y_k)}$$

Forward Euler method, or
Euler's method, or
Explicit Euler's method

Easy to evaluate: Plug in $t_n, y_k \longrightarrow$ obtain y_{n+1}.

Now if we had used the "right-sided" rectangle rule

$\int_a^b f dx \approx f(b)(b-a)$, we would obtain:

$$y_{k+1} - y_k = \int_{t_k}^{t_{k+1}} f(\tau, y)\, d\tau \approx f(t_{k+1}, y_{k+1})\, \Delta t$$

$$\Rightarrow \boxed{y_{k+1} = y_k + \Delta t f(t_{k+1}, y_{k+1})}$$

Backward Euler method, or
Implicit Euler method.

Note: We need to solve a (possibly nonlinear) equation to obtain y_{k+1} (y_{n+1} is not isolated in this equation).

One more variant: trapezoidal rule

$$\int_a^b f\,dx = \frac{f(a)+f(b)}{2} \cdot (b-a)$$

$$y_{k+1} - y_k = \int_{t_k}^{t_{k+1}} f(\tau, y)\,d\tau \approx \frac{f(t_k, y_k) + f(t_{k+1}, y_{k+1})}{2} \cdot \Delta t$$

$$\Rightarrow \boxed{y_{k+1} = y_k + \frac{\Delta t}{2}\left\{ f(t_k, y_k) + f(t_{k+1}, y_{k+1}) \right\}}$$

example: $y'(t) = -ty^2$ using trap. rule

$$y_{k+1} = y_k + \frac{\Delta t}{2}\left\{ -t_k y_k^2 - t_{k+1} y_{k+1}^2 \right\}$$

Let: $t_k = 0.9$ $y_k = 1$

$\Delta t = 0.1$

$$y_{k+1} = 1 + 0.05\left\{ -0.9 - 1 \cdot y_{k+1}^2 \right\}$$

$\Rightarrow 0.05 y_{k+1}^2 + y_{k+1} + 1.045 = 0 \Rightarrow$ solve quadratic to get y_{k+1}

Another example:

$$\left.\begin{aligned} y'(t) &= -2y(t) \\ y(0) &= 1 \end{aligned}\right\} \quad \text{exact solution } y(t) = e^{-2t}$$

Using Forward Euler:

$$y_{k+1} = y_k + \Delta t\, f(t_k, y_k)$$

$$= y_k - 2\Delta t\, y_n = (1 - 2\Delta t) y_k$$

Thus
$$y_1 = (1-2\Delta t) y_0$$
$$y_2 = (1-2\Delta t) y_1 = (1-2\Delta t)^2 y_0$$
$$\vdots$$
$$y_n = (1-2\Delta t)^k y_0$$

How does this behave when $\Delta t \to 0$?

$$(1-2\Delta t)^k = \left[\left(1 + \frac{1}{-\frac{1}{2\Delta t}}\right)^{-\frac{1}{2\Delta t}}\right]^{-2k\Delta t}$$

Using $\lim \left(1 + \frac{1}{x}\right)^x = e \Rightarrow \lim_{\Delta t \to 0} (1-2\Delta t)^k = e^{-2k\Delta t} = e^{-2t_n}$

Thus, when $\Delta t \to 0$ $y_n \to e^{-2t_n}$ (compare with exact solution $y(t) = e^{-2t}$).

Initial value problems for 1st order ODE's

Problem statement: Find $y(t)$: $[t_0, +\infty)$

s.t

$$y'(t) = f(t, y)$$
$$y(t_0) = y_0$$

Method (1-step algorithms)

→ Set $t_k = t_0 + k \cdot \Delta t$

→ Define $y_k := y(t_k)$

→ Iteratively approximate

$$y_{k+1} = y_k + \Delta t\, f(t_k, y_k)$$
(Forward Euler)

Explicit

$$y_{k+1} = y_k + \Delta t\, (t_{k+1}, y_{k+1})$$
(Backward Euler)

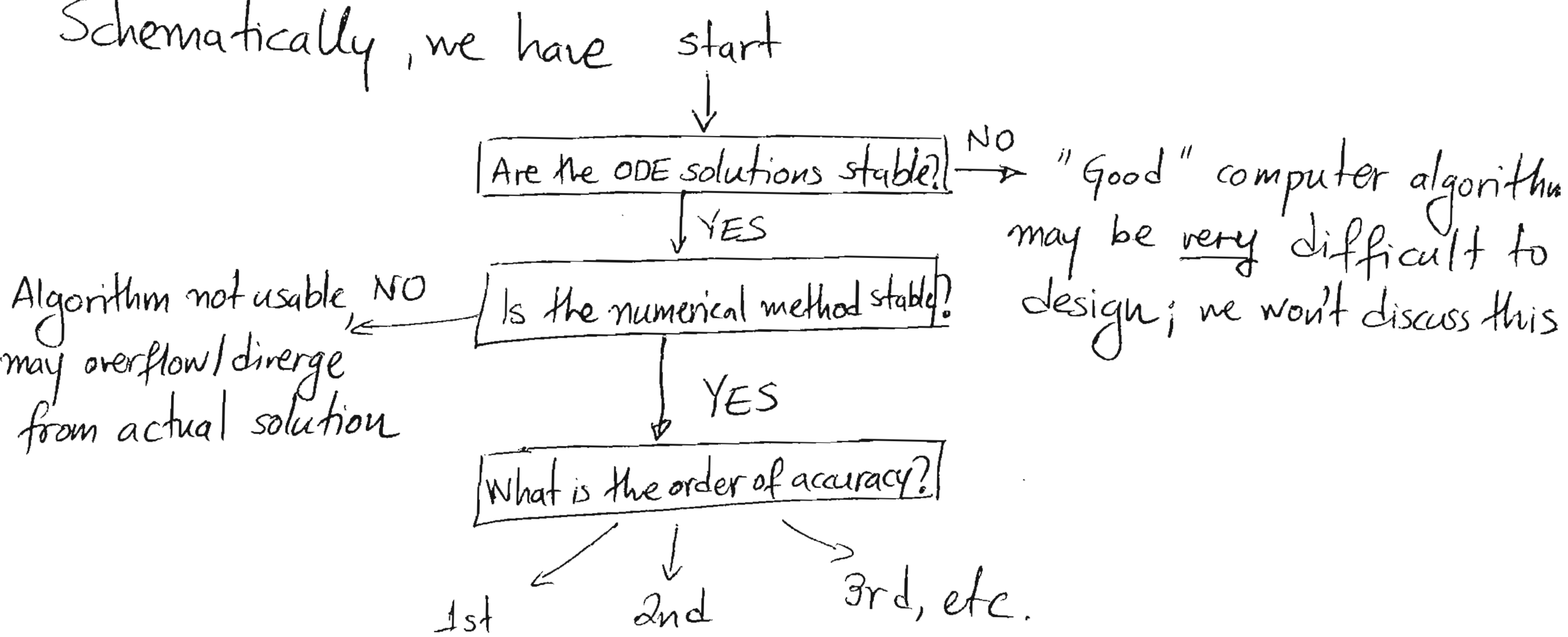
$$y_{k+1} = y_k + \frac{\Delta t}{2}\left[f(t_k, y_k) + f(t_{k+1}, y_{k+1})\right]$$
(Trapezoidal)

Implicit (need to solve an equation for y_{k+1}).

Before we use one of these algorithms in practice we need to examine their limitations, and ensure they are usable for a specific problem. We look at the following

Properties of the ODE itself	Properties of the numerical method
→ Are the solutions to the ODE stable?	→ Is the numerical method stable? Under what conditions?
	→ What is the accuracy of the method?

Schematically, we have



Stability of solutions to ODE

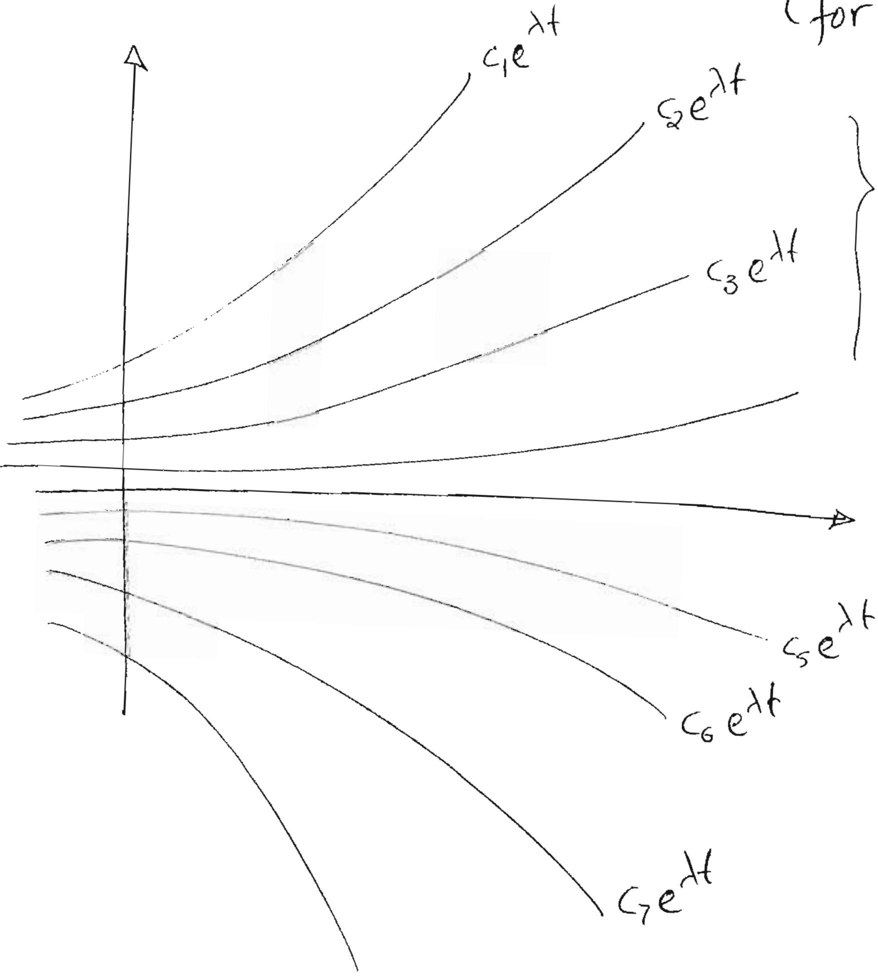
4/21/11 L3

We formulated an IVP as:

$$y'(t) = f(t, y) \longleftarrow \text{ODE}$$
$$y(t_0) = y_0 \longleftarrow \text{initial condition}$$

Under normal circumstances we expect this to have a unique solution; however if we omit the initial condition, we get an entire <u>family</u> of solutions to the ODE.

e.g. $y'(t) = \lambda y(t) \Rightarrow$ Exact solution $y(t) = ce^{\lambda t}$ (for any arbitrary $c \in \mathbb{R}$)



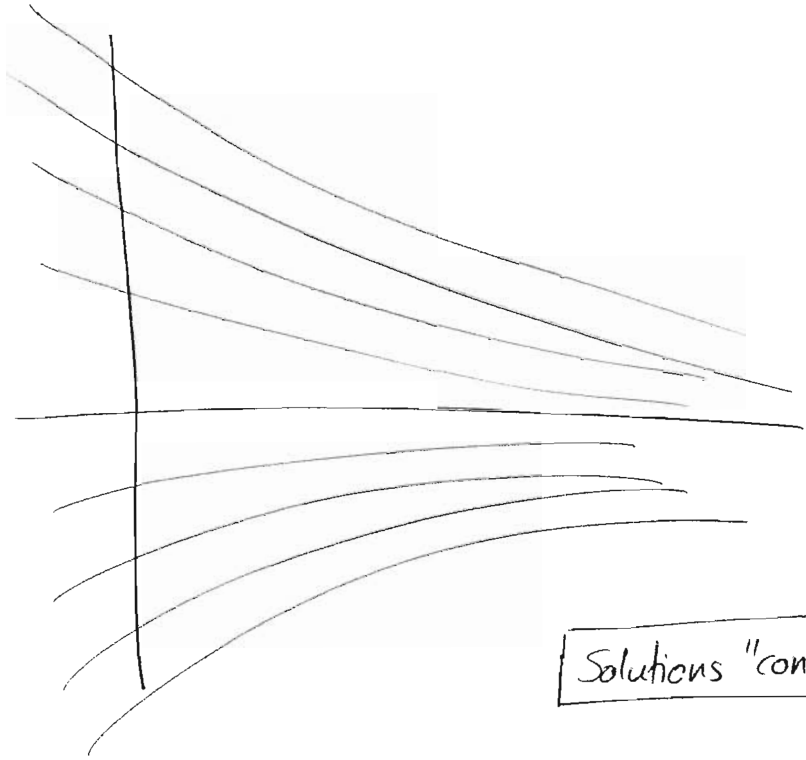
positive c_i's

Case $\lambda > 0$

$\boxed{\text{Solutions "diverge"}}$

Negative c_i's.

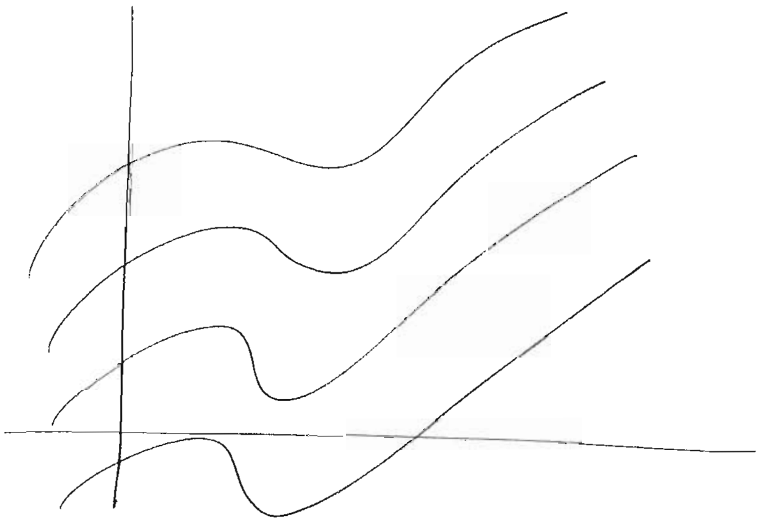
Case $\lambda < 0$



$\boxed{\text{Solutions "converge"}}$

- $y'(t) = f(t)$ (f is a function of t alone, not y)

Exact solution: $y(t) = \int_{t_0}^{t} f(\tau)\,d\tau + C$, $C \in \mathbb{R}$



$\boxed{\text{Solutions stay at fixed distance apart}}$

Definition

- An ODE is said to have <u>stable</u> solutions if the distance between any 2 solutions y & $\hat{y}$ remains bounded, i.e $|y(t)-\hat{y}(t)| \leq \text{const} \quad \forall t \geq t_0$

  (Strictly speaking, we must also be able to make this constant arbitrarily small, by bringing the initial values y_0 & $\hat{y}_0$ closer together).

- If we additionally have that for any 2 solutions $y(t)$ & $\hat{y}(t)$ we have $\lim_{t\to\infty} |y(t)-\hat{y}(t)| = 0$, the ODE has <u>asymptotically stable</u> solutions.

  <u>Note</u>: If the ODE is <u>asymptotically stable</u> then, it is <u>stable</u>, too.

- Otherwise (i.e. when solutions diverge away from one another) the ODE is said to have <u>unstable</u> solutions.

The ODE $\boxed{y'(t) = \lambda y(t)}$ is called the 4/21/11 LG

<u>model</u> 1st order ODE and is extremely useful as an example in the analysis of stability, etc. We have:

- When <u>$\lambda < 0$</u> the solutions to the model ODE are <u>asymptotically stable</u> (converge towards one another)
- When $\lambda > 0$ the solutions are <u>unstable</u> (diverge away)
- When $\lambda = 0$ the solutions are <u>stable</u> although not asymptotically stable (they stay within bounded distance).

For a more general ODE $y' = f(t,y)$ the criteria are:

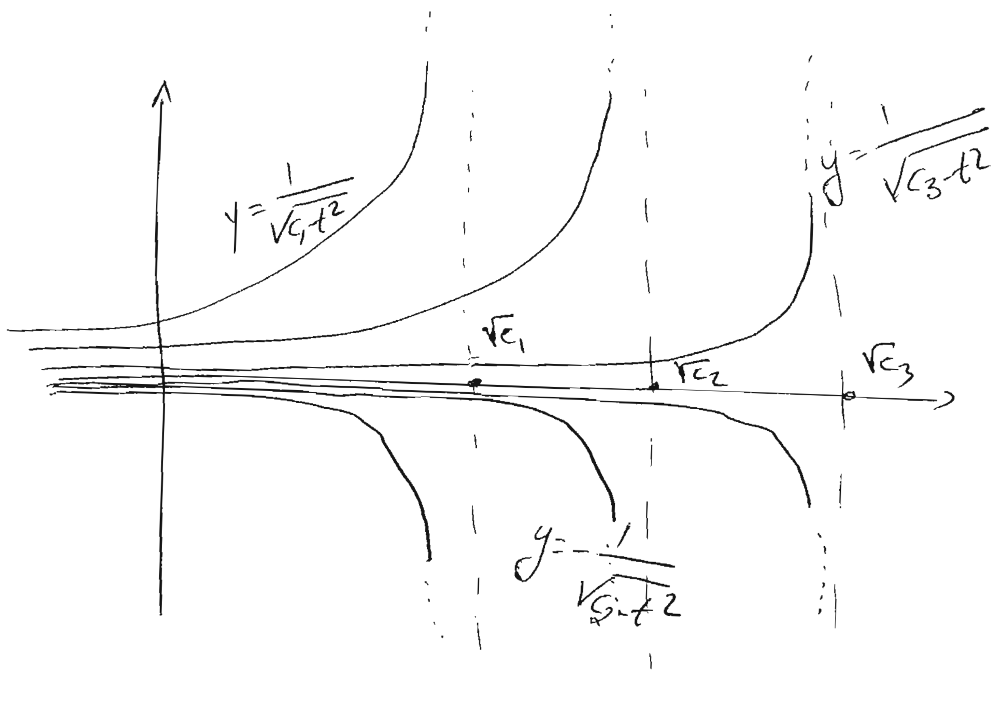
- If $\frac{\partial f}{\partial y}(t,y) < 0$ for all t & y, the solutions are <u>asymptotically stable</u>
- If $\frac{\partial f}{\partial y}(t,y) \leq 0$ $\forall t,y$, the solutions are <u>stable</u> (but not necessarily asymptotically stable).
- If $\frac{\partial f}{\partial y}(t,y)$ is positive or changes sign $\Rightarrow$ we cannot conclude stability with certainty.

Why do we ideally want ODEs with stable solutions?

→ Errors (approximation, truncation, roundoff) tend to move us away from the "intended" solution to an IVP, and onto another function from the family of solutions to the ODE. If the solution is stable (or even better, asymptotically stable) then the error remains bounded (or diminishes, for asymptotic stability) over time.

→ ODEs with unstable solutions are prone to developing problematic behaviors. For example, different solutions may become <u>undefined</u> after a certain (solution-dependent) point in time.

e.g. $y'(t) = ty^3 \Rightarrow$ Exact solution : $y(t) = \pm \dfrac{1}{\sqrt{c-t^2}}$



Designing a "usable" algorithm for approximating solutions to an unstable ODE is highly nontrivial, and we will not address it in CS412! So, we will continue under the premise that the ODE in question is stable.

→ Sometimes, even if the ODE is stable, an approximation method may diverge/overflow! e.g.

$y' = \lambda y \quad \lambda < 0 \quad$ (Exact solution $y(t) = y_0 e^{\lambda(t-t_0)}$)

Using Forward Euler:

$$y_{k+1} = y_k + \Delta t \lambda y_k = (1 + \lambda \Delta t) y_k$$

$$\Rightarrow \quad y_k = (1 + \lambda \Delta t)^k y_0$$

When $\lambda < 0$ the exact solution satisfies:

$$y(t) = y_0 e^{\lambda(t-t_0)} \xrightarrow[t \to \infty]{} 0$$

However, for the approximate solution

$$y_k = (1+\lambda\Delta t)^k y_0 \xrightarrow[t\to\infty]{} \begin{cases} \text{Converges to } 0, \text{ if } |1+\lambda\Delta t|<1 \\ \text{Diverges to } \pm\infty, \text{ if } |1+\lambda\Delta t|>1 \\ \text{Oscillates, if } |1+\lambda\Delta t| = 1. \end{cases}$$

<u>Def</u> A numerical method is called <u>stable</u>, when if applied to an ODE with stable solutions, exhibits the same asymptotic behavior with the exact solution when $t\to\infty$.

In our case, the proper asymptotic behavior is $y_k \xrightarrow[t\to\infty]{} 0$, which is only guaranteed when

$$|1+\lambda\Delta t| < \quad \Leftrightarrow \quad -1 < 1+\lambda\Delta t < 1$$

$$\Leftrightarrow \quad -2 < \underbrace{\lambda\Delta t < 0}_{\substack{\text{always true} \\ \text{since } \lambda<0}} \quad \Leftrightarrow \quad -2 < -|\lambda|\,\Delta t$$

$$\Leftrightarrow \quad \boxed{\Delta t < \frac{2}{|\lambda|}}$$

Stability condition for Forward Euler!

(All implicit methods have <u>some</u> condition for stability)

What about Backward Euler? Again we test on the model stable ODE $y' = \lambda y$, $\lambda < 0$

$$y_{k+1} = y_k + \Delta t\, f(t_{k+1}, y_{k+1})$$

$$y_{k+1} = y_k + \Delta t \cdot \lambda\, y_{k+1}$$

$$(1 - \lambda \Delta t)\, y_{k+1} = y_k \Rightarrow y_{k+1} = \frac{1}{1 - \lambda \Delta t}\, y_k$$

$$\Rightarrow \boxed{y_k = \left(\frac{1}{1 - \lambda \Delta t}\right)^k y_0}$$

Here, in order to have $y_k \xrightarrow[k \to \infty]{} 0$, we need

$$\left|\frac{1}{1 - \lambda \Delta t}\right| < 1 \Leftrightarrow |1 - \lambda \Delta t| > 1$$ Always true! since $\lambda < 0$.

Thus, Backward Euler is unconditionally stable!

Similarly, for trapezoidal rule:

(on model ODE $y' = \lambda y,\ \lambda < 0$)

$$y_{k+1} = y_k + \frac{\Delta t}{2}\left[f(t_k, y_k) + f(t_{k+1}, y_{k+1})\right]$$

$$\Rightarrow y_{k+1} = y_k + \frac{\Delta t}{2}\left[\lambda y_k + \lambda y_{k+1}\right]$$

$$\Rightarrow \left(1 - \frac{\lambda \Delta t}{2}\right) y_{k+1} = \left(1 + \frac{\lambda \Delta t}{2}\right) y_k$$

$$\Rightarrow \quad y_k = \left[\frac{\left(1 + \frac{\lambda \Delta t}{2}\right)}{\left(1 - \frac{\lambda \Delta t}{2}\right)}\right]^k y_0$$

For stability we need:

$$\left|\frac{1 + \frac{\lambda \Delta t}{2}}{1 - \frac{\lambda \Delta t}{2}}\right| < 1 \iff \left|1 + \frac{\lambda \Delta t}{2}\right| < \underbrace{\left|1 - \frac{\lambda \Delta t}{2}\right|}_{>0} \iff$$

$$\iff \left|1 + \frac{\lambda \Delta t}{2}\right| < 1 - \frac{\lambda \Delta t}{2} \iff \underbrace{-1 + \frac{\lambda \Delta t}{2} < 1}_{\text{Always true}} + \underbrace{\frac{\lambda \Delta t}{2} < 1}_{\substack{\text{Always true,} \\ \text{for } \lambda < 0}} - \frac{\lambda \Delta t}{2}$$

Thus, the trapezoidal rule is also <u>unconditionally stable</u>!