Numerical Solutions of Initial Value Problems --- Numerical Methods for ODEs

MATH 3014
Monday & Thursday 14:30-15:45
Instructor: **Dr. Luo Li**
https://www.fst.um.edu.mo/personal/liluo/math3014/

Department of Mathematics
Faculty of Science and Technology

- Finite difference methods for IVPs
- How to use Matlab programming to solve IVP
- The methods described here can be used as time discretization techniques for various applications.

Reference book:
Kendall Atkinson, Weimin Han, David Stewart, *Numerical Solution of Ordinary Differential Equations*, John Wiley & Sons, Inc.
https://onlinelibrary.wiley.com/doi/book/10.1002/9781118164495

A **first-order differential equation**

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

where $Y(t)$ is an unknown function that is being sought.

For example, given a function g, $Y'(t) = g(t)$ $\longrightarrow$ $Y(t) = \int g(s)\, ds + c$

with c an arbitrary integration constant. The constant c, and thus a particular solution, can be obtained by using the initial condition $Y(t_0) = Y_0$.

Example
$$\begin{cases} Y'(t) = \sin(t) \\ Y\left(\frac{\pi}{3}\right) = 2, \end{cases}$$

The general solution of the equation is $Y(t) = -\cos(t) + c.$

If we specify the condition $Y\left(\frac{\pi}{3}\right) = 2,$ then it is easy to find $c = 2.5$.

Example Using the method of integrating factors.

$$Y'(t) = \lambda Y(t) + g(t)$$

with λ a given constant. Multiplying the linear equation by the integrating factor $e^{-\lambda t}$, we can reformulate the equation as $\frac{d}{dt}\left(e^{-\lambda t}Y(t)\right) = e^{-\lambda t}g(t)$.

Integrating both sides from t_0 to t, we obtain

$$e^{-\lambda t}Y(t) = c + \int_{t_0}^{t} e^{-\lambda s}g(s)\,ds,$$

where

$$c = e^{-\lambda t_0}Y(t_0).$$

So the general solution

$$Y(t) = e^{\lambda t}\left[c + \int_{t_0}^{t} e^{-\lambda s}g(s)\,ds\right] = ce^{\lambda t} + \int_{t_0}^{t} e^{\lambda(t-s)}g(s)\,ds.$$

General Solvability Theory

A well-posed problem

1. The solution exists,
2. The solution is unique,
3. The solution is not sensitive to perturbations of the data.
   - initial condition
   - coefficients

For an IVP, the Lipschitz continuity can guarantee the well-posedness.

Theorem *Let D be an open connected set in $\mathbb{R}^2$, let $f(t,y)$ be a continuous function of t and y for all (t,y) in D, and let (t_0, Y_0) be an interior point of D. Assume that $f(t,y)$ satisfies the* Lipschitz condition

$$|f(t,y_1) - f(t,y_2)| \le K\,|y_1 - y_2| \qquad \text{all } (t,y_1), (t,y_2) \text{ in } D$$

for some $K \ge 0$. Then there is a unique function $Y(t)$ defined on an interval $[t_0 - \alpha, t_0 + \alpha]$ for some $\alpha > 0$, satisfying

$$Y'(t) = f(t, Y(t)), \qquad t_0 - \alpha \le t \le t_0 + \alpha,$$

$$Y(t_0) = Y_0.$$

For example, we can use $K = \max\limits_{(t,y)\in\overline{D}} \left| \dfrac{\partial f(t,y)}{\partial y} \right|$

Mean value theorem

provided this is finite.

Example Consider the initial value problem

$$Y'(t) = 2t[Y(t)]^2, \qquad Y(0) = 1.$$

Here

$$f(t, y) = 2ty^2, \qquad \frac{\partial f(t, y)}{\partial y} = 4ty,$$

and both of these functions are $\boxed{\text{continuous}}$ for all (t, y). Thus, by the theorem there is a unique solution to this initial value problem for t in a neighborhood of $t_0 = 0$. This solution is

$$Y(t) = \frac{1}{1 - t^2}, \qquad \boxed{t \neq \pm 1.}$$

This example illustrates that the continuity of $f(t, y)$ and $\partial f(t, y)/\partial y$ for all (t, y) $\boxed{\text{does not imply}}$ the existence of a solution $Y(t)$ for all t. ■

Stability of the Initial Value Problem

Stability means that a small perturbation in the initial value of the problem leads to a small change in the solution.

Make a small change in the initial value for the initial value problem,

$$Y_\epsilon'(t) = f(t, Y_\epsilon(t)), \qquad t_0 \le t \le b, \qquad Y_\epsilon(t_0) = Y_0 + \epsilon.$$

The original problem $\boxed{Y'(t) = f(t, Y(t)),}$ $\boxed{Y(t_0) = Y_0.}$

If for some $c > 0$ that is independent of ϵ,

$$\|Y_\epsilon - Y\|_\infty \equiv \max_{t_0 \le t \le b} |Y_\epsilon(t) - Y(t)| \le c\,\epsilon$$

then small changes in the initial value Y_0 will lead to small changes in the solution $Y(t)$ of the initial value problem.

$$\begin{cases} \|Y_\epsilon - Y\|_\infty \approx \epsilon & : \text{well–conditioned} \\ \|Y_\epsilon - Y\|_\infty \gg \epsilon & : \text{ill–conditioned} \end{cases}$$

Example The problem

$$Y'(t) = \lambda\left[Y(t) - 1\right], \qquad 0 \le t \le b, \qquad Y(0) = 1$$

has the solution

$$Y(t) = 1, \qquad 0 \le t \le b.$$

The perturbed problem

$$Y_\epsilon'(t) = \lambda[Y_\epsilon(t) - 1], \qquad 0 \le t \le b, \qquad Y_\epsilon(0) = 1 + \epsilon$$

has the solution

$$Y_\epsilon(t) = 1 + \epsilon e^{\lambda t}, \qquad 0 \le t \le b.$$

For the error, we obtain

$$Y(t) - Y_\epsilon(t) = -\epsilon e^{\lambda t},$$

$$\max_{0 \le t \le b} |Y(t) - Y_\epsilon(t)| = \begin{cases} |\epsilon|, & \lambda \le 0, \longrightarrow \text{well–conditioned} \\ |\epsilon|\, e^{\lambda b}, & \lambda \ge 0. \longrightarrow \text{ill–conditioned} \end{cases}$$

Why Numerical Methods?

- Many differential equations are too complicated to have solution formulas.
- Numerical methods provide a powerful alternative tool for solving the differential equation

Denote $Y(t)$ the true solution of the initial value problem with the initial value Y_0

$$\begin{cases} Y'(t) = f(t, Y(t)), & t_0 \le t \le b, \\ Y(t_0) = Y_0. \end{cases}$$

We aim to find an approximate solution $y(t)$ at a discrete set of nodes,

$$t_n = t_0 + nh, \qquad n = 0, 1, \ldots, N.$$

The following notations are all used for the approximate solution at the node points:

$$y(t_n) = y_h(t_n) = y_n, \qquad n = 0, 1, \ldots, N.$$

1.3 The Forward EULER'S Method

A forward difference approximation $Y'(t) \approx \frac{1}{h}[Y(t+h) - Y(t)]$.

Applying this to the initial value problem at $t = t_n$, $Y'(t_n) = f(t_n, Y(t_n))$,

we obtain

$$\frac{1}{h}[Y(t_{n+1}) - Y(t_n)] \approx f(t_n, Y(t_n)),$$

$$\boxed{Y(t_{n+1})} \approx Y(t_n) + hf(t_n, Y(t_n)).$$

Euler's method is defined by taking this to be exact:

$$\boxed{y_{n+1}} = y_n + hf(t_n, y_n), \qquad 0 \leq n \leq N-1.$$

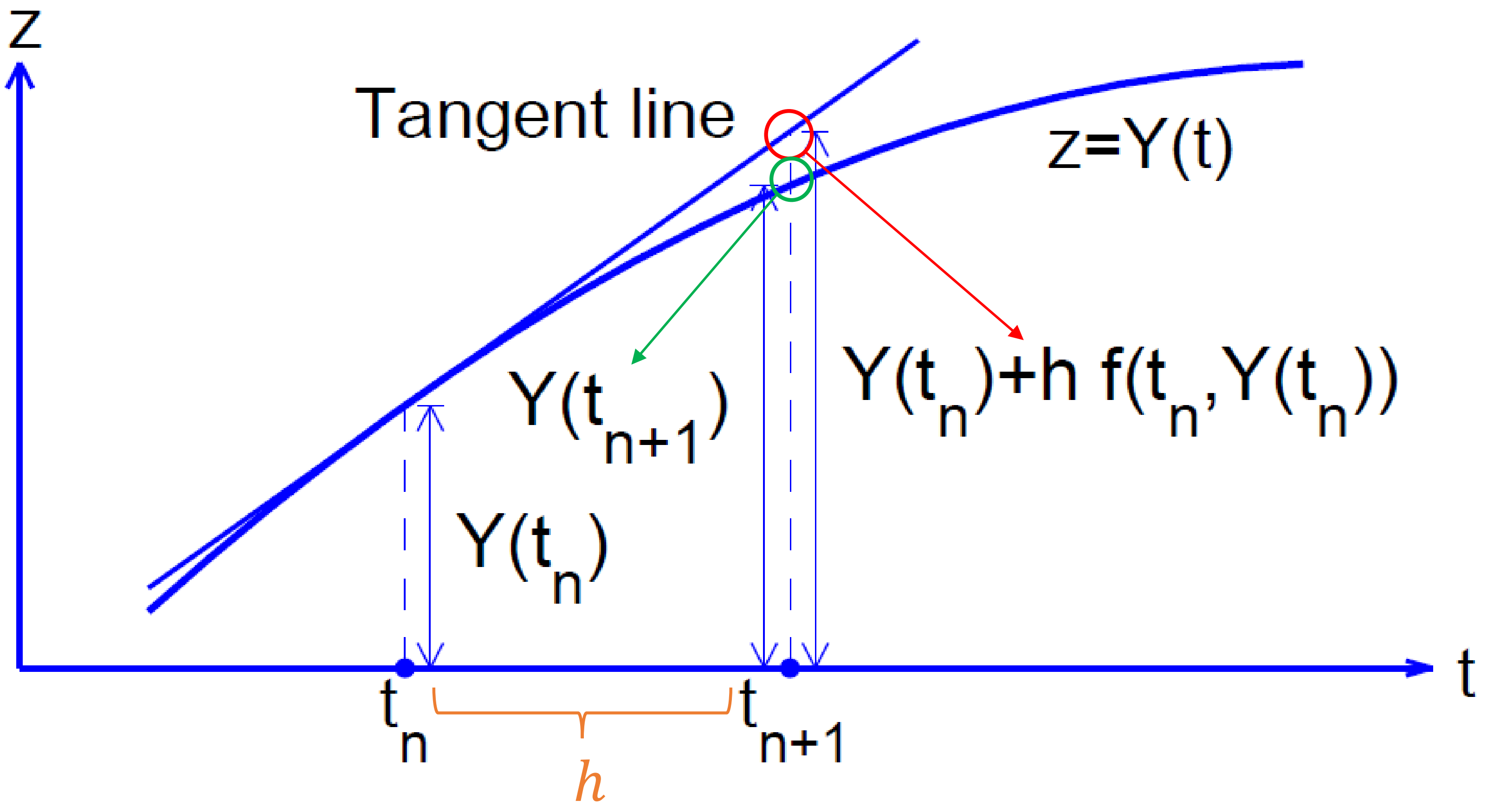


Figure: An illustration of Forward Euler's Method

The tangent line at t_n has slope $Y'(t_n) = f(t_n, Y(t_n))$.

Example Solve

$$Y'(t) = \frac{Y(t) + t^2 - 2}{t+1}, \qquad Y(0) = 2$$

whose true solution is

$$Y(t) = t^2 + 2t + 2 - 2(t+1)\log(t+1).$$

Euler's method for this differential equation is

$$y_{n+1} = y_n + \frac{h(y_n + t_n^2 - 2)}{t_n + 1}, \qquad n \geq 0$$

with $y_0 = 2$ and $t_n = nh$.

Matlab program for Forward Euler's Method $y_{n+1} = y_n + hf(t_n, y_n)$

```
function [t,y] = euler_for(t0,y0,t_end,h,fcn)
    % Solve the initial value problem
    % y' = f(t,y), t0 <= t <= b, y(t0)=y0

    n = fix((t_end-t0)/h)+1;
    t = linspace(t0, t0+(n-1)*h, n)';
    y = zeros(n, 1);

    y(1) = y0;

    for i = 2:n
        y(i) = y(i-1)+h*feval(fcn, t(i-1), y(i-1));
    end
```

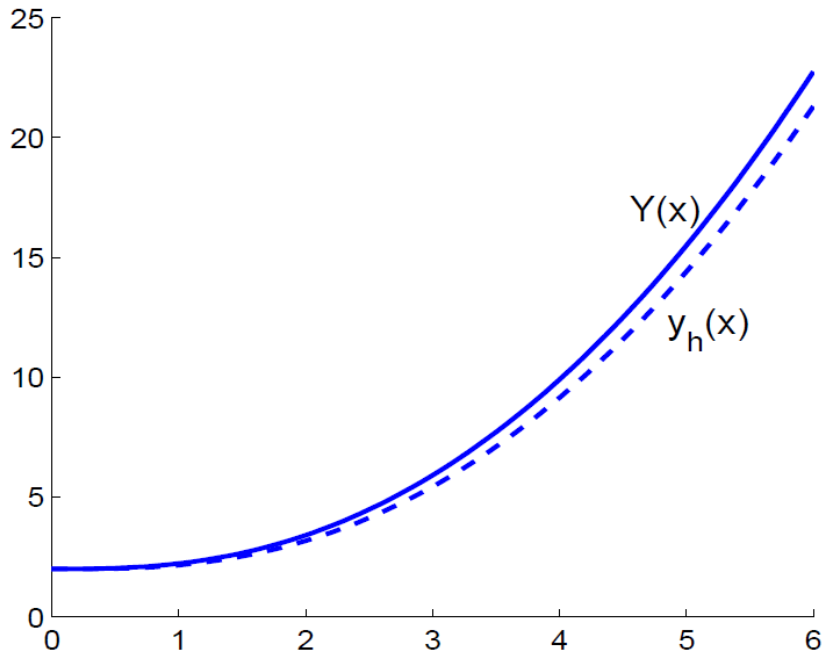
The routine returns two vectors

The right-side function of the differential equation.

The node points
t(j)=t0+(j−1)*h, j=1, 2, . . . , N

Approximated solution vector

h	t	$y_h(t)$	*Error*	*Relative Error*
0.2	1.0	2.1592	6.82e − 2	0.0306
	2.0	3.1697	2.39e − 1	0.0701
	3.0	5.4332	4.76e − 1	0.0805
	4.0	9.1411	7.65e − 1	0.129
	5.0	14.406	1.09	0.0703
	6.0	21.303	$\boxed{1.45}$	0.0637
0.1	1.0	2.1912	3.63e − 2	0.0163
	2.0	3.2841	1.24e − 1	0.0364
	3.0	5.6636	2.46e − 1	0.0416
	4.0	9.5125	3.93e − 1	0.0665
	5.0	14.939	5.60e − 1	0.0361
	6.0	22.013	$\boxed{7.44e-1}$	0.0327
0.05	1.0	2.2087	1.87e − 2	0.00840
	2.0	3.3449	6.34e − 2	0.0186
	3.0	5.7845	1.25e − 1	0.0212
	4.0	9.7061	1.99e − 1	0.0337
	5.0	15.214	2.84e − 1	0.0183
	6.0	22.381	$\boxed{3.76e-1}$	0.0165



Solution of Forward Euler's Method when $h = 0.2$.

1.4 Error Analysis of Euler's Method

- Assume that the initial value problem has a unique solution $Y(t)$ on $t_0 \leq t \leq b$
- Assume that the solution has a bounded second derivative $Y''(t)$ over this interval

$$Y(t_{n+1}) = Y(t_n) + hY'(t_n) + \tfrac{1}{2}h^2Y''(\xi_n)$$

for some $t_n \leq \xi_n \leq t_{n+1}$. Using the fact that $Y(t)$ satisfies the differential equation,

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

our Taylor approximation becomes

$$Y(t_{n+1}) = Y(t_n) + hf(t_n, Y(t_n)) + \tfrac{1}{2}h^2Y''(\xi_n).$$

The term

$$T_{n+1} = \tfrac{1}{2}h^2Y''(\xi_n)$$

is called the *truncation error* for Euler's method, and it is the error in the approximation

$$Y(t_{n+1}) \approx Y(t_n) + hf(t_n, Y(t_n)).$$

To analyze the error in Euler's method, subtract $y_{n+1} = y_n + hf(t_n, y_n)$

from $Y(t_{n+1}) = Y(t_n) + hf(t_n, Y(t_n)) + \frac{1}{2}h^2 Y''(\xi_n)$

we have
$$Y(t_{n+1}) - y_{n+1} = Y(t_n) - y_n + h[f(t_n, Y(t_n)) - f(t_n, y_n)] + \frac{1}{2}h^2 Y''(\xi_n).$$

The error in y_{n+1} consists of two parts:

(1) the truncation error T_{n+1}, newly introduced at step t_{n+1};

(2) the propagated error $Y(t_n) - y_n + h\boxed{[f(t_n, Y(t_n)) - f(t_n, y_n)]}$.

$$\boxed{f(t_n, Y(t_n)) - f(t_n, y_n) = \frac{\partial f(t_n, \zeta_n)}{\partial y}[Y(t_n) - y_n]}$$

Mean value theorem

for some ζ_n between $Y(t_n)$ and y_n. Let $e_k \equiv Y(t_k) - y_k$, $k \geq 0$,

$$e_{n+1} = \left[1 + h\frac{\partial f(t_n, \zeta_n)}{\partial y}\right] e_n + \frac{1}{2}h^2 Y''(\xi_n). \quad (*)$$

$$e_{n+1} = \left[1 + h\frac{\partial f(t_n, \zeta_n)}{\partial y}\right] e_n + \tfrac{1}{2}h^2 Y''(\xi_n).$$

Let us first consider a special case that the error in Euler's method. Consider using Euler's method to solve the problem

$$Y'(t) = 2t, \qquad Y(0) = 0,$$

whose true solution is $Y(t) = t^2$. Then, from the error formula (*), we have

$$e_{n+1} = e_n + h^2, \qquad e_0 = 0,$$

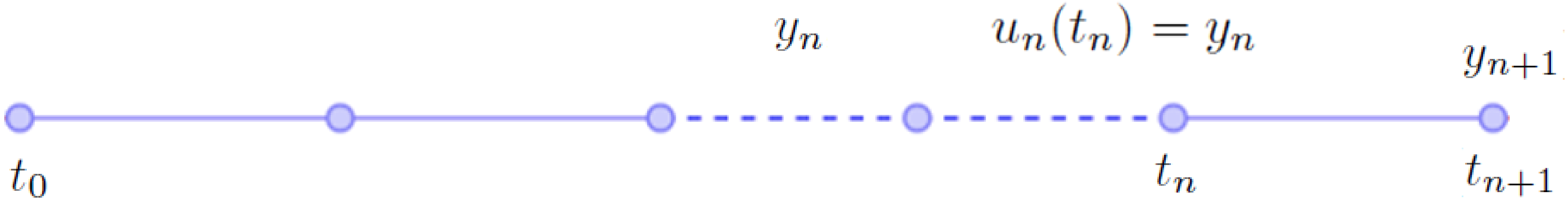
where we are assuming the initial value $y_0 = Y(0)$. This leads, by induction, to

$$e_n = nh^2, \qquad n \geq 0.$$

Since $nh = t_n$,

$$e_n = ht_n.$$

For each fixed t_n, the error at t_n is proportional to h. The truncation error is $\mathcal{O}(h^2)$, but the cumulative effect of these errors is a total error proportional to h.



What if at some point t_{n+1} we discover that $Y(t_{n+1}) - y_{n+1}$ is too large?

Decreasing h from t_n to t_{n+1}? No!
Decreasing h from t_{n-1} to t_{n+1}? No!

We should decrease h from t_0 to t_{n+1}!

The error $Y(t_{n+1}) - y_{n+1}$ is called the **global error** or total error at t_{n+1}.

We next define the **local error** by introducing the following initial value problem:

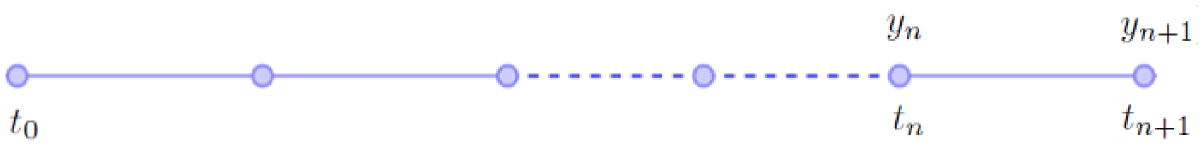
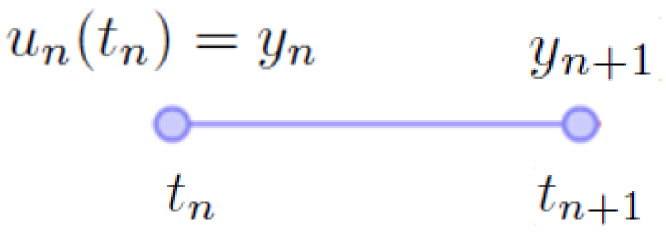
$$u_n'(t) = f(t, \boxed{u_n(t)}), \qquad t \geq t_n,$$
$$u_n(t_n) = y_n.$$

local solution

Assuming the solution y_n at t_n is the exact solution.

local error: $LE_{n+1} = u_n(t_{n+1}) - y_{n+1}.$

Relation with truncation error?

 Global initial value problem: from t_0 to t_{n+1} $Y'(t) = f(t, Y(t)),$ $Y(t_0) = Y_0.$	 Local initial value problem: from t_n to t_{n+1} $u_n'(t) = f(t, u_n(t))\,,$ $u_n(t_n) = y_n.$
Initial value: given value Y_0	Initial value: the numerical solution y_n
Numerical method to obtain y_{n+1} : Euler $y_{n+1} = y_n + hf(t_n, y_n)$	Numerical method to obtain y_{n+1} : Euler $y_{n+1} = y_n + hf(t_n, y_n)$
Global error $Y(t_{n+1}) - y_{n+1}$	Local error = Truncation error $u_n(t_{n+1}) - y_{n+1}$

For the initial value problem $\begin{aligned} Y'(t) &= f(t, Y(t)), \qquad t_0 \le t \le b, \\ Y(t_0) &= Y_0. \end{aligned}$ (**)

If there exists $K \geq 0$ such that $|f(t, y_1) - f(t, y_2)| \le K\, |y_1 - y_2|$ (***)
for $-\infty < y_1, y_2 < \infty$ and $t_0 \le t \le b$.

Theorem *Let $f(t, y)$ be a continuous function for $t_0 \le t \le b$ and $-\infty < y < \infty$, and further assume that $f(t, y)$ satisfies the* Lipschitz condition (***). *Assume that the solution $Y(t)$ of* (**) *has a continuous second derivative on $[t_0, b]$. Then the solution $\{y_h(t_n) \mid t_0 \le t_n \le b\}$ obtained by Euler's method satisfies*

$$\max_{t_0 \le t_n \le b} |Y(t_n) - y_h(t_n)| \le e^{(b-t_0)K}\, |e_0| + \left[\frac{e^{(b-t_0)K} - 1}{K}\right] \tau(h),$$

where

$$\tau(h) = \tfrac{1}{2} h \, \|Y''\|_\infty = \tfrac{1}{2} h \max_{t_0 \le t \le b} |Y''(t)|$$

and $e_0 = Y_0 - y_h(t_0)$.

If, in addition, we have

Initial error e_0

$$\boxed{|Y_0 - y_h(t_0)|} \le c_1 h \qquad as\ h \to 0$$

for some $c_1 \ge 0$ *(e.g., if* $Y_0 = y_0$ *for all* h*, then* $c_1 = 0$*), then there is a constant* $B \ge 0$ *for which*

$$\max_{t_0 \le t_n \le b} \boxed{|Y(t_n) - y_h(t_n)|} \le \boxed{Bh}$$

Final error e_n

In general, if we have $|Y(t_n) - y_h(t_n)| \le ch^p, \qquad t_0 \le t_n \le b$

for some constant $p \ge 0$, then we say that the numerical method is convergent with order p.

Proof:

Let $e_n = Y(t_n) - y(t_n), n \geq 0$. Let $N \equiv N(h)$ be the integer for which

$$t_N \leq b, \qquad t_{N+1} > b.$$

Define

$$\tau_n = \tfrac{1}{2} h Y''(\xi_n), \qquad 0 \leq n \leq N(h) - 1,$$

then
$$\max_{0 \leq n \leq N-1} |\tau_n| \leq \tau(h) = \tfrac{1}{2} h \, \|Y''\|_\infty$$

From
$$\boxed{Y(t_{n+1}) - y_{n+1}} = \boxed{Y(t_n) - y_n} + h[f(t_n, Y(t_n)) - f(t_n, y_n)] + \tfrac{1}{2}\boxed{h^2 Y''(\xi_n).}$$

we obtain
$$e_{n+1} = e_n + h\left[f(t_n, Y_n) - f(t_n, y_n)\right] + h\tau_n.$$

Taking bounds using $|f(t, y_1) - f(t, y_2)| \leq K\,|y_1 - y_2|$, we obtain

$$|e_{n+1}| \leq |e_n| + hK\,|Y_n - y_n| + h\,|\tau_n|\,,$$

$$|e_{n+1}| \leq (1 + hK)\,|e_n| + h\tau(h), \qquad 0 \leq n \leq N(h) - 1.$$

Apply this recursively to obtain

$$|e_n| \leq (1 + hK)^n\,|e_0| + \left[1 + (1 + hK) + \cdots + (1 + hK)^{n-1}\right] h\tau(h).$$

Using the formula for the sum of a finite geometric series,

$$1 + r + r^2 + \cdots + r^{n-1} = \frac{r^n - 1}{r - 1}, \qquad r \neq 1,$$

$\boxed{\leq e^{(b-t_0)K}}$

we obtain

$$|e_n| \leq \boxed{(1 + hK)^n}\,|e_0| + \left[\frac{\boxed{(1 + hK)^n} - 1}{K}\right] \tau(h).$$

Lemma *For any real* t,

$$1 + t \le e^t,$$

and for any $t \ge -1$, *any* $m \ge 0$,

$$0 \le (1+t)^m \le e^{mt}.$$

Proof. Using Taylor's theorem yields

$$e^t = 1 + t + \tfrac{1}{2}t^2 e^{\xi} \quad \text{with } \xi \text{ between } 0 \text{ and } t.$$

Using this lemma, we have

$$(1 + hK)^n \le e^{nhK} = e^{(t_n - t_0)K} \le e^{(b-t_0)K}$$

Substitute back to the formula, we obtain

$$\max_{t_0 \le t_n \le b} |Y(t_n) - y_h(t_n)| \le e^{(b-t_0)K}\,|e_0| + \left[\frac{e^{(b-t_0)K} - 1}{K}\right]\tau(h)$$

$$\boxed{\max_{t_0 \le t_n \le b} |Y(t_n) - y_h(t_n)| \le e^{(b-t_0)K} |e_0| + \left[\frac{e^{(b-t_0)K} - 1}{K}\right] \tau(h)}$$

If, in addition, $|Y_0 - y_h(t_0)| \le c_1 h$, there is a constant

$$B = c_1 e^{(b-t_0)K} + \frac{1}{2}\left[\frac{e^{(b-t_0)K} - 1}{K}\right] \|Y''\|_\infty$$

Such that

$$\max_{t_0 \le t_n \le b} |Y(t_n) - y_h(t_n)| \le Bh.$$

The procedure of the proof

1. Subtract the "Taylor expansion of the exact solution $Y(t_{n+1})$ at t_n" with the "numerical scheme of y_{n+1}".
2. Apply the Lipschitz condition to obtain the inequality between $|e_{n+1}|$ and $|e_n|$.
3. Apply the inequality recursively from n to 0.
4. Use some summation formulas to simplify the expression.
5. Use the Lemma to allow having $t_n - t_0 = nh$.

1.5 Numerical Stability

Define a numerical solution $\{z_n\}$

$$z_{n+1} = z_n + hf(t_n, z_n), \qquad n = 0, 1, \ldots, N(h) - 1$$

with $z_0 = y_0 + \epsilon$. We now compare the two numerical solutions $\{z_n\}$ and $\{y_n\}$ as $h \to 0$.

Let $e_n = z_n - y_n$, $n \geq 0$. Then $e_0 = \epsilon$, and subtracting $y_{n+1} = y_n + hf(t_n, y_n)$

we obtain $$e_{n+1} = e_n + h\left[f(t_n, z_n) - f(t_n, y_n)\right].$$

Lipschitz condition

Taking bounds using $\boxed{|f(t, y_1) - f(t, y_2)| \leq K\,|y_1 - y_2|}$, we have

$$|e_{n+1}| \leq |e_n| + hK\,|z_n - y_n| \quad \text{or} \quad |e_{n+1}| \leq (1 + hK)\,|e_n|$$

Apply this recursively to obtain

$$|e_n| \leq (1 + hK)^n\,|e_0|$$

Lemma *For any real* t,

$$1 + t \le e^t,$$

and for any $t \ge -1$, *any* $m \ge 0$,

$$0 \le (1+t)^m \le e^{mt}.$$

Using this lemma, we obtain

$$(1+hK)^n \le e^{nhK} = e^{(t_n - t_0)K} \le e^{(b-t_0)K},$$

substitute to $|e_n| \le (1+hK)^n |e_0|$, and note that $e_0 = \epsilon$, the following holds

$$\max_{0 \le n \le N(h)} |z_n - y_n| \le e^{(b-t_0)K} |\epsilon| .$$

Consequently, there is a constant $\widehat{c} \ge 0$, independent of h, such that

$$\max_{0 \le n \le N(h)} |z_n - y_n| \le \widehat{c} |\epsilon| .$$

Euler's method is a stable numerical method for the initial value problem if $hK \geq -1$.

- The forward Euler's method is a first-order method. $\max_{t_0 \le t_n \le b} |Y(t_n) - y_h(t_n)| \le Bh.$

  when the step size h is smaller, the method is more accurate.
- A very small h decreases the efficiency of the numerical method.
- The forward Euler's method may not be stable when h is large.

Example

$$Y' = \lambda Y, \quad t > 0,$$
$$Y(0) = 1.$$

$\lambda < 0$ or λ is complex and with $\mathrm{Real}(\lambda) < 0$.

The true solution of the problem is

$$Y(t) = e^{\lambda t},$$

which decays exponentially in t since the parameter λ has a negative real part.

We would like the numerical solution satisfies

$$y_h(t_n) \to 0 \quad \text{as} \quad t_n \to \infty$$

The Euler method on the model problem

$$y_{n+1} = y_n + h\lambda\, y_n = (1 + h\lambda)\, y_n, \qquad n \geq 0, \quad y_0 = 1.$$

By an inductive argument, it is not difficult to find

$$y_n = (1 + h\lambda)^n, \qquad n \geq 0.$$

Note that for a fixed node point $t_n = n\,h \equiv \bar{t}$, as $n \to \infty$, we obtain

$$y_n = \left(1 + \frac{\lambda\bar{t}}{n}\right)^n \to e^{\lambda\bar{t}}.$$

We can see that $y_n \to 0$ as $n \to \infty$ if and only if

$$|1 + h\lambda| < 1 \quad \text{or} \quad \boxed{-2 < h\lambda < 0}$$

Region of absolute stability

Example Consider the model problem with $\lambda = -100$.

$$Y' = \lambda Y, \quad t > 0,$$
$$Y(0) = 1.$$

The true solution $Y(t) = e^{-100t}$ at $t = 0.2$ is 2.061×10^{-9}.

The forward Euler method will perform well only when $\boxed{h < 2 \times 100^{-1}} = 0.02.$

$$\boxed{-2 < h\lambda < 0}$$

h	$y_h(0.2)$
0.1	81
0.05	256
0.02	1
0.01	0
0.001	7.06e − 10

The Backward Euler Method

Absolutely stable: a numerical method is stable for any step size h.

i.e., $y_h(t_n) \to 0$ as $t_n \to \infty$ for $\begin{cases} Y' = \lambda Y, & t > 0, \\ Y(0) = 1. \end{cases}$

The backward Euler method has this property.

Forward difference approximation

$$Y'(t) \approx \frac{1}{h}\left[Y(t+h) - Y(t)\right] \Longrightarrow$$

The forward Euler's method

$$\begin{cases} y_{n+1} = y_n + hf\left(\boxed{t_n, y_n}\right), \\ y_0 = Y_0. \end{cases}$$

Backward difference approximation

$$Y'(t) \approx \frac{1}{h}\left[Y(t) - Y(t-h)\right] \Longrightarrow$$

The backward Euler method

$$\begin{cases} y_{n+1} = y_n + h\,f\left(\boxed{t_{n+1}, y_{n+1}}\right), \\ y_0 = Y_0. \end{cases}$$

Like the Euler method, the backward Euler method is of first-order accuracy.

The backward Euler's method for the model problem is absolutely stable:

$$\begin{cases} Y' = \lambda Y, & t > 0, \\ Y(0) = 1. \end{cases}$$

Applying the backward Euler's method,

$$y_{n+1} = y_n + h\lambda\, y_{n+1},$$

$$y_{n+1} = (1 - h\lambda)^{-1} y_n, \qquad n \geq 0.$$

Using this together with $y_0 = 1$, we obtain

$$y_n = (1 - h\lambda)^{-n}.$$

For any stepsize $h > 0$, we have $|1 - h\lambda| > 1$ and so $y_n \to 0$ as $n \to \infty$.

The forward Euler method

h	$y_h(0.2)$
0.1	81
0.05	256
0.02	1
0.01	0
0.001	7.06e − 10

The backward Euler method

h	$y_h(0.2)$
0.1	8.26e − 3
0.05	7.72e − 4
0.02	1.69e − 5
0.01	9.54e − 7
0.001	5.27e − 9

The backward Euler's method is an implicit method: y_{n+1} must be found by solving a root finding problem (usually, by solving a nonlinear algebraic equation).

$$y_{n+1} = y_n + h\, f(t_{n+1}, \boxed{y_{n+1}})$$

Lipschitz continuity assumption on $f(t, y)$
h is small enough
} The equation has a unique solution

Given an initial guess $y_{n+1}^{(0)} \approx y_{n+1}$, define $y_{n+1}^{(1)}$, $y_{n+1}^{(2)}$, etc., by

$$y_{n+1}^{(j+1)} = y_n + h\, f(t_{n+1}, y_{n+1}^{(j)}), \qquad j = 0, 1, 2, \ldots.$$

Will $y_{n+1}^{(j)}$ converge to y_{n+1}?

By subtraction, $y_{n+1} - y_{n+1}^{(j+1)} = h\,[f(t_{n+1}, y_{n+1}) - f(t_{n+1}, y_{n+1}^{(j)})]$,

$$y_{n+1} - y_{n+1}^{(j+1)} \approx h \cdot \boxed{\frac{\partial f(t_{n+1}, y_{n+1})}{\partial y}}\,[y_{n+1} - y_{n+1}^{(j)}].$$

Mean value theorem
&
h is small

If $\left. \begin{array}{l} \left| h \cdot \dfrac{\partial f(t_{n+1}, y_{n+1})}{\partial y} \right| < 1 \\ y_{n+1}^{(0)} \to y_{n+1} \end{array} \right\}$ the errors will converge to zero

The usual choice of the initial guess is based on the forward Euler method.

The Predictor Formula:

$$\overline{y}_{n+1} = y_n + hf(t_n, y_n),$$
$$y_{n+1} = y_n + h\, f(t_{n+1}, \overline{y}_{n+1}).$$

Or in combined form: $y_{n+1} = y_n + h\, f(t_{n+1},\, y_n + h\, f(t_n, y_n))$

- The scheme predicts the root of the implicit method.
- The scheme is usually sufficient to do the iteration once.
- The scheme is still of first-order accuracy.
- The scheme is no longer absolutely stable. i.e., try $\begin{array}{l} Y' = \lambda Y, \quad t > 0,\ \lambda < 0 \\ Y(0) = 1. \end{array}$

Matlab program for Backward Euler's Method

$$y_{n+1}^{(1)} = y_n + hf(t_n, y_n)$$

$$y_{n+1}^{(k+1)} = y_n + h\, f(t_{n+1}, y_{n+1}^{(k)})$$

```
function [t,y] = euler_back(t0,y0,t_end,h,fcn,tol)
% Initialize
n = fix((t_end-t0)/h)+1;
t = linspace(t0,t0+(n-1)*h,n)';
y = zeros(n,1);
y(1) = y0;
i = 2;

% advancing
while i <= n
    ...
    i = i+1;
end
```

```
% forward Euler estimate
yt1 = y(i-1)+h*feval(fcn,t(i-1),y(i-1));

% one-point iteration
count = 0;    diff = 1;
while diff > tol & count < 10
    yt2 = y(i-1) + h*feval(fcn,t(i),yt1);
    diff = abs(yt2-yt1);
    yt1 = yt2;
    count = count +1;
End

if count >= 10
    disp('Not converging after 10 steps at t = ')
    fprintf('%5.2f\n', t(i))
end
y(i) = yt2;
```

The Trapezoidal Method

Drawback of both the forward Euler method and the backward Euler method: only first-order accuracy

The Trapezoidal Method
- Has a higher convergence order
- Has the stability property for any step size h

To derive the Trapezoidal Method, we start from the trapezoidal rule for numerical integration

$a \leq \xi \leq b.$

$$\int_a^b g(s)\,ds = \frac{1}{2}(b-a)\left[g(a)+g(b)\right] - \frac{1}{12}(b-a)^3 g''(\boxed{\xi})$$

We integrate the differential equation $Y'(t) = f(t, Y(t))$ from t_n to t_{n+1}:

$$Y(t_{n+1}) = Y(t_n) + \int_{t_n}^{t_{n+1}} f(s, Y(s))\,ds.$$

Use the trapezoidal rule to approximate the integral:

$$Y(t_{n+1}) = Y(t_n) + \tfrac{1}{2}h\left[f(t_n, Y(t_n)) + f(t_{n+1}, Y(t_{n+1}))\right]$$
$$-\boxed{\tfrac{1}{12}h^3 Y^{(3)}(\boxed{\xi_n})} \qquad \boxed{t_n \le \xi_n \le t_{n+1}}$$

Truncation Error

By dropping the final error term and then equating both sides,

$$\begin{cases} y_{n+1} = y_n + \tfrac{1}{2}h\left[f(t_n, y_n) + f(t_{n+1}, y_{n+1})\right], \quad n \ge 0, \\ y_0 = Y_0. \end{cases}$$

The trapezoidal method

- is of second-order accuracy $\quad \max_{t_0 \le t_n \le b} |Y(t_n) - y_h(t_n)| \le \boxed{ch^2}$
- is absolutely stable i.e., try $\quad Y' = \lambda Y, \quad t > 0, \ \lambda < 0$, $\quad Y(0) = 1.$

The trapezoidal method is an implicit method

$$y_{n+1}^{(j+1)} = y_n + \frac{h}{2}[f(t_n, y_n) + f(t_{n+1}, y_{n+1}^{(j)})], \qquad j = 0, 1, 2, \ldots.$$

If $\left\{ \begin{array}{l} \left| \frac{h}{2} \cdot \frac{\partial f(t_{n+1}, y_{n+1})}{\partial y} \right| < 1 \\ y_{n+1}^{(0)} \rightarrow y_{n+1} \end{array} \right\}$ the iteration will converge

The usual choice of the initial guess is based on the forward Euler method.

$$y_{n+1}^{(0)} = y_n + hf(t_n, y_n),$$

and if we accept $y_{n+1}^{(1)}$ as the value of y_{n+1}, then the resulting new scheme is called Heun's method

$$y_{n+1} = y_n + \frac{h}{2}\left[f(t_n, y_n) + f(t_{n+1}, y_n + h\,f(t_n, y_n))\right]$$

- The Heun method is of second-order accuracy.
- The Heun method it is no longer absolutely stable. i.e., try $Y' = \lambda Y, \quad t > 0,$ $Y(0) = 1.$

Forward Euler Method Not Absolutely Stable	
Backward Euler Method Absolutely Stable	**Trapezoidal method** Absolutely Stable
Backward Euler Method with Forward Euler as Predictor Not Absolutely Stable	**Trapezoidal Method with Forward Euler as Predictor (Heun's method)** Not Absolutely Stable

Matlab program for Trapezoidal Method

```
function [t,y] = trapezoidal (t0,y0,t_end,h,fcn,tol)
% Initialize
n = fix((t_end-t0)/h)+1;
t = linspace(t0,t0+(n-1)*h,n)';
y = zeros(n,1);
y(1) = y0;
i = 2;

% advancing
while i <= n
    ...
    i = i+1;
end
```

```
% forward Euler estimate
yt1 = y(i-1)+h*feval(fcn,t(i-1),y(i-1));

% one-point iteration
count = 0;    diff = 1;
while diff > tol & count < 10
    yt2 = y(i-1) +h*(feval(fcn,t(i-1),y(i-1))+
                     feval(fcn,t(i),yt1))/2;
    diff = abs(yt2-yt1);
    yt1 = yt2;
    count = count +1;
End

if count >= 10
    disp('Not converging after 10 steps at t = ')
    fprintf('%5.2f\n', t(i))
end
y(i) = yt2;
```

Example Consider the problem

$$Y'(t) = \lambda Y(t) + (1-\lambda)\cos(t) - (1+\lambda)\sin(t), \qquad Y(0) = 1,$$

whose true solution is $Y(t) = \sin(t) + \cos(t)$.

Forward Euler's method vs Backward Euler's method vs Trapezoidal method

A larger h is better, more efficient

Parameters: $h = \boxed{0.5,}\ 0.1,\ 0.01$

$\lambda = -1,\ -10,\ \boxed{-50}$

Stiff Differential Equation

Forward Euler's method

λ	t	Error $h=0.5$	Error $h=0.1$	Error $h=0.01$
-1	1	$-2.46\text{e}-1$	$-4.32\text{e}-2$	$-4.22\text{e}-3$
	2	$-2.55\text{e}-1$	$-4.64\text{e}-2$	$-4.55\text{e}-3$
	3	$-2.66\text{e}-2$	$-6.78\text{e}-3$	$-7.22\text{e}-4$
	4	$2.27\text{e}-1$	$3.91\text{e}-2$	$3.78\text{e}-3$
	5	$2.72\text{e}-1$	$4.91\text{e}-2$	$4.81\text{e}-3$
-10	1	$3.98\text{e}-1$	$-6.99\text{e}-3$	$-6.99\text{e}-4$
	2	$6.90\text{e}+0$	$-2.90\text{e}-3$	$-3.08\text{e}-4$
	3	$1.11\text{e}+2$	$3.86\text{e}-3$	$3.64\text{e}-4$
	4	$1.77\text{e}+3$	$7.07\text{e}-3$	$7.04\text{e}-4$
	5	$2.83\text{e}+4$	$3.78\text{e}-3$	$3.97\text{e}-4$
-50	1	$3.26\text{e}+0$	$1.06\text{e}+3$	$-1.39\text{e}-4$
	2	$1.88\text{e}+3$	$1.11\text{e}+9$	$-5.16\text{e}-5$
	3	$1.08\text{e}+6$	$1.17\text{e}+15$	$8.25\text{e}-5$
	4	$6.24\text{e}+8$	$1.23\text{e}+21$	$1.41\text{e}-4$
	5	$3.59\text{e}+11$	$1.28\text{e}+27$	$7.00\text{e}-5$

- The actual global error depends strongly on λ
- Unstable and rapid growth happen when $|\lambda|h$ is outside the stability region $|1+h\lambda|<1$
- The forward Euler scheme is of first-order accuracy

Backward Euler's method $h = 0.5$

t	Error $\lambda = -1$	Error $\lambda = -10$	Error $\lambda = -50$
2	$2.08\mathrm{e}-1$	$1.97\mathrm{e}-2$	$3.60\mathrm{e}-3$
4	$-1.63\mathrm{e}-1$	$-3.35\mathrm{e}-2$	$-6.94\mathrm{e}-3$
6	$-7.04\mathrm{e}-2$	$8.19\mathrm{e}-3$	$2.18\mathrm{e}-3$
8	$2.22\mathrm{e}-1$	$2.67\mathrm{e}-2$	$5.13\mathrm{e}-3$
10	$-1.14\mathrm{e}-1$	$-3.04\mathrm{e}-2$	$-6.45\mathrm{e}-3$

No stability problems!

Trapezoidal method $h = 0.5$

t	Error $\lambda = -1$	Error $\lambda = -10$	Error $\lambda = -50$
2	$-1.13\mathrm{e}-2$	$-2.78\mathrm{e}-3$	$-7.91\mathrm{e}-4$
4	$-1.43\mathrm{e}-2$	$-8.91\mathrm{e}-5$	$-8.91\mathrm{e}-5$
6	$2.02\mathrm{e}-2$	$2.77\mathrm{e}-3$	$4.72\mathrm{e}-4$
8	$-2.86\mathrm{e}-3$	$-2.22\mathrm{e}-3$	$-5.11\mathrm{e}-4$
10	$-1.79\mathrm{e}-2$	$-9.23\mathrm{e}-4$	$-1.56\mathrm{e}-4$

The backward Euler method and the trapezoidal method are therefore more desirable!

Higher Order Methods:
Taylor and Runger–Kutta Methods

Forward Euler's method

$$Y'(t) \approx \frac{1}{h}\left[Y(t+h) - Y(t)\right]$$

Linear Taylor polynomial approximation

$$Y(t_{n+1}) \approx Y(t_n) + hY'(t_n),$$

How about using higher-order Taylor approximations to improve the accuracy (or speed)?

Taylor methods

- Need higher-order derivatives
- Usually tedious and time-consuming

Runge–Kutta methods

- Use compositions of the right-side function to approximate the derivative
- Among the most popular methods in solving IVP

Example For the problem

$$\boxed{Y'(t)} = -Y(t) + 2\cos(t), \qquad Y(0) = 1,$$

whose true solution is $Y(t) = \sin(t) + \cos(t)$.

We use the quadratic Taylor approximation

$$Y(t_{n+1}) \approx Y(t_n) + h\boxed{Y'(t_n)} + \frac{1}{2}h^2\boxed{Y''(t_n)}.$$

Its truncation error is

$$T_{n+1}(Y) = \frac{1}{6}h^3 Y'''(\xi_n), \qquad \text{some } t_n \le \xi_n \le t_{n+1}.$$

Differentiat

$$\boxed{Y'(t) = -Y(t) + 2\cos(t)}$$

$$Y''(t) = -Y'(t) - 2\sin(t) = Y(t) - 2\cos(t) - 2\sin(t).$$

Substitute into the Taylor expansion, we have

$$\begin{aligned} Y(t_{n+1}) \approx{} & Y(t_n) + h[-Y(t_n) + 2\cos(t_n)] \\ & + \frac{1}{2}h^2[Y(t_n) - 2\cos(t_n) - 2\sin(t_n)]. \end{aligned}$$

By forcing equality, $$y_{n+1} = y_n + h[-y_n + 2\cos(t_n)] + \tfrac{1}{2}h^2[y_n - 2\cos(t_n) - 2\sin(t_n)], \qquad n \ge 0 \quad \text{with } y_0 = 1.$$

Results of the second-order Taylor method

h	t	$y_h(t)$	Error	Euler Error
0.1	2.0	0.492225829	9.25e − 4	−4.64e − 2
	4.0	−1.411659477	1.21e − 3	3.91e − 2
	6.0	0.682420081	−1.67e − 3	1.39e − 2
	8.0	0.843648978	2.09e − 4	−5.07e − 2
	10.0	−1.384588757	1.50e − 3	2.83e − 2
0.05	2.0	0.492919943	2.31e − 4	−2.30e − 2
	4.0	−1.410737402	2.91e − 4	1.92e − 2
	6.0	0.681162413	−4.08e − 4	6.97e − 3
	8.0	0.843801368	5.68e − 5	−2.50e − 2
	10.0	−1.383454154	3.62e − 4	1.39e − 2

In general, for the initial value problem

$$Y'(t) = f(t, Y(t)), \qquad t_0 \le t \le b, \qquad Y(t_0) = Y_0$$

Taylor method selects a Taylor approximation of order p

$$Y(t_{n+1}) \approx Y(t_n) + hY'(t_n) + \cdots + \frac{h^p}{p!} Y^{(p)}(t_n),$$

With the truncation error $T_{n+1}(Y) = \dfrac{h^{p+1}}{(p+1)!} Y^{(p+1)}(\xi_n), \qquad t_n \le \xi_n \le t_{n+1}.$

Find $Y''(t), \ldots, Y^{(p)}(t)$ by differentiating the differential equation successively, obtaining formulas that implicitly involve only t_n and $Y(t_n)$.

$$Y''(t) = f_t + f_y f,$$

$$Y^{(3)}(t) = f_{tt} + 2\, f_{ty} f + f_{yy} f^2 + f_y (f_t + f_y f),$$

See next page for the derivation

where

$$f_t = \frac{\partial f}{\partial t}, \quad f_y = \frac{\partial f}{\partial y}, \quad f_{ty} = \frac{\partial^2 f}{\partial t \partial y},$$

For higher derivatives, too complicate!!!

$$Y''(t) = \left(Y'(t)\right)' = \left(f(t,y)\right)' = f_t + f_y y_t = f_t + f_y f$$

$$\begin{aligned}
Y'''(t) &= \left(Y''(t)\right)' \\
&= \left(f_t + f_y f\right)' \\
&= \left(f_t(t,y)\right)' + \left(f_y(t,y) f(t,y)\right)' \\
&= f_{tt} + f_{ty} y_t + \left(f_y\right)' f + f_y f' \\
&= f_{tt} + f_{ty} f + \left(f_{yt} + f_{yy} y_t\right) f + f_y\left(f_t + f_y y_t\right) \\
&= f_{tt} + f_{ty} f + \left(f_{yt} + f_{yy} f\right) f + f_y\left(f_t + f_y f\right)
\end{aligned}$$

- Assume that $f_{ty} = f_{yt}$, substitute into the above formula, we can get the derivation on Slide 49 of Charpter 1.
- Note that f, f_t, f_y are also functions that depend on (t, y), and y depends on t, so we need to use chain rule for their derivatives w.r.t t.
- $Y'''(t)$ is already very complicate, so the Taylor method is not a good choice compared to the Runge-Kutta Method.

Substitute these derivatives into the Taylor approximation and force it to be an equality, we have

$$y_{n+1} = y_n + hy'_n + \frac{h^2}{2}y''_n + \cdots + \frac{h^p}{p!}y_n^{(p)}$$

where $y'_n = f(t_n, y_n)\,, \qquad y''_n = (f_t + f_y f)\,(t_n, y_n)\,,$ etc.

If the solution $Y(t)$ and the derivative function $f(t, z)$ are sufficiently differentiable, the method satisfies

$$\max_{t_0 \le t_n \le b} |Y(t_n) - y_h(t_n)| \le ch^p \cdot \max_{t_0 \le t \le b} \left| Y^{(p+1)}(t) \right| .$$

The Taylor approximation of order p leads to a convergent numerical method with order of convergence p.

Taylor methods

- Need higher-order derivatives
- Usually tedious and time-consuming

Runge–Kutta methods

- Evaluate $f(t, y)$ at more points
- Retain the accuracy of the Taylor approximation
- Fairly easy to program

Runge–Kutta Methods

$$y_{n+1} = y_n + h\boxed{F(t_n, y_n; h)}, \qquad n \geq 0, \qquad y_0 = Y_0.$$

some kind of "average slope" of the solution

For methods of order 2,

$$F(t, y; h) = b_1 f(t, y) + b_2 f(t + \alpha h,\ y + \beta h f(t, y))$$

We determine the constants $\{\alpha, \beta, b_1, b_2\}$ so that the truncation error will satisfy

$$T_{n+1}(Y) \equiv Y(t_{n+1}) - [Y(t_n) + hF(t_n, Y(t_n); h)] = \boxed{\mathcal{O}(h^3)}$$

1 2

Truncation error for a 2-or method

To find the equations for the constants, we use Taylor expansions to compute the truncation error $T_{n+1}(Y)$.

To find the equations for the constants, we use Taylor expansions to compute the truncation error $T_{n+1}(Y)$. For the term $f(t+\alpha h, y+\beta h f(t,y))$, we first expand with respect to the second argument around y. Note that we need a remainder $\mathcal{O}(h^2)$:

$$f(t+\alpha h, y+\beta h f(t,y)) = f(t+\alpha h, y) + f_y(t+\alpha h, y)\beta h f(t,y) + \mathcal{O}(h^2).$$

We then expand the terms with respect to the t variable to obtain

$$f(t+\alpha h, y+\beta h f(t,y)) = f + f_t \alpha h + f_y \beta h f + \boxed{\mathcal{O}(h^2),}$$

A lot of things can be put here

① For the term $Y(t_{n+1})$

$$\begin{aligned} Y(t+h) &= Y + h\boxed{Y'} + \frac{h^2}{2}\boxed{Y''} + \mathcal{O}(h^3) \\ &= Y + hf + \frac{h^2}{2}(f_t + f_y f) + \mathcal{O}(h^3). \end{aligned}$$

$$\boxed{Y'(t) = f}$$

$$\boxed{Y''(t) = f_t + f_y f}$$

② For the term $f(t+\alpha h, y + \beta h f(t,y))$

We first expand $f(t+\alpha h, y+\beta h f(t,y))$ with respect to the second argument around y.

$$f(t+\alpha h, y+\beta h f(t,y)) = f(t+\alpha h, y) + f_y(t+\alpha h, y)\beta h f(t,y) + \mathcal{O}(h^2).$$

We then expand the terms with respect to the t variable to obtain

$$f(t+\alpha h, y+\beta h f(t,y)) = f + f_t \alpha h + f_y \beta h f + \mathcal{O}(h^2),$$

Then
$$
\begin{aligned}
T_{n+1}(Y) &= Y(t+h) - [Y(t) + h\,F(t,Y(t);h)] \\
&= Y + hf + \tfrac{1}{2}h^2(f_t + f_y f) \\
&\quad - [Y + \underline{hb_1 f + b_2 h\,(f + \alpha h f_t + \beta h f_y f)}] + \mathcal{O}(h^3) \\
&= h\,\underline{(1 - b_1 - b_2)}\,f + \tfrac{1}{2}h^2[\underline{(1 - 2\,b_2\alpha)}\,f_t \\
&\qquad + \underline{(1 - 2\,b_2\beta)}\,f_y f] + \mathcal{O}(h^3).
\end{aligned}
$$

The coefficients must satisfy the system

Underdetermined system
$$
\begin{cases}
1 - b_1 - b_2 = 0, \\
1 - 2\,b_2\alpha = 0, \\
1 - 2\,b_2\beta = 0.
\end{cases}
$$

By solving this system, we have

$$
b_2 \neq 0, \qquad b_1 = 1 - b_2, \qquad \alpha = \beta = \frac{1}{2b_2}.
$$

Thus there is a family of Runge–Kutta methods of order 2, depending on the choice of b_2. The three favorite choices are $b_2 = \frac{1}{2}, \frac{3}{4}$, and 1.

With $b_2 = \frac{1}{2}$, we obtain the numerical method

Forward Euler solution at t_{n+1}

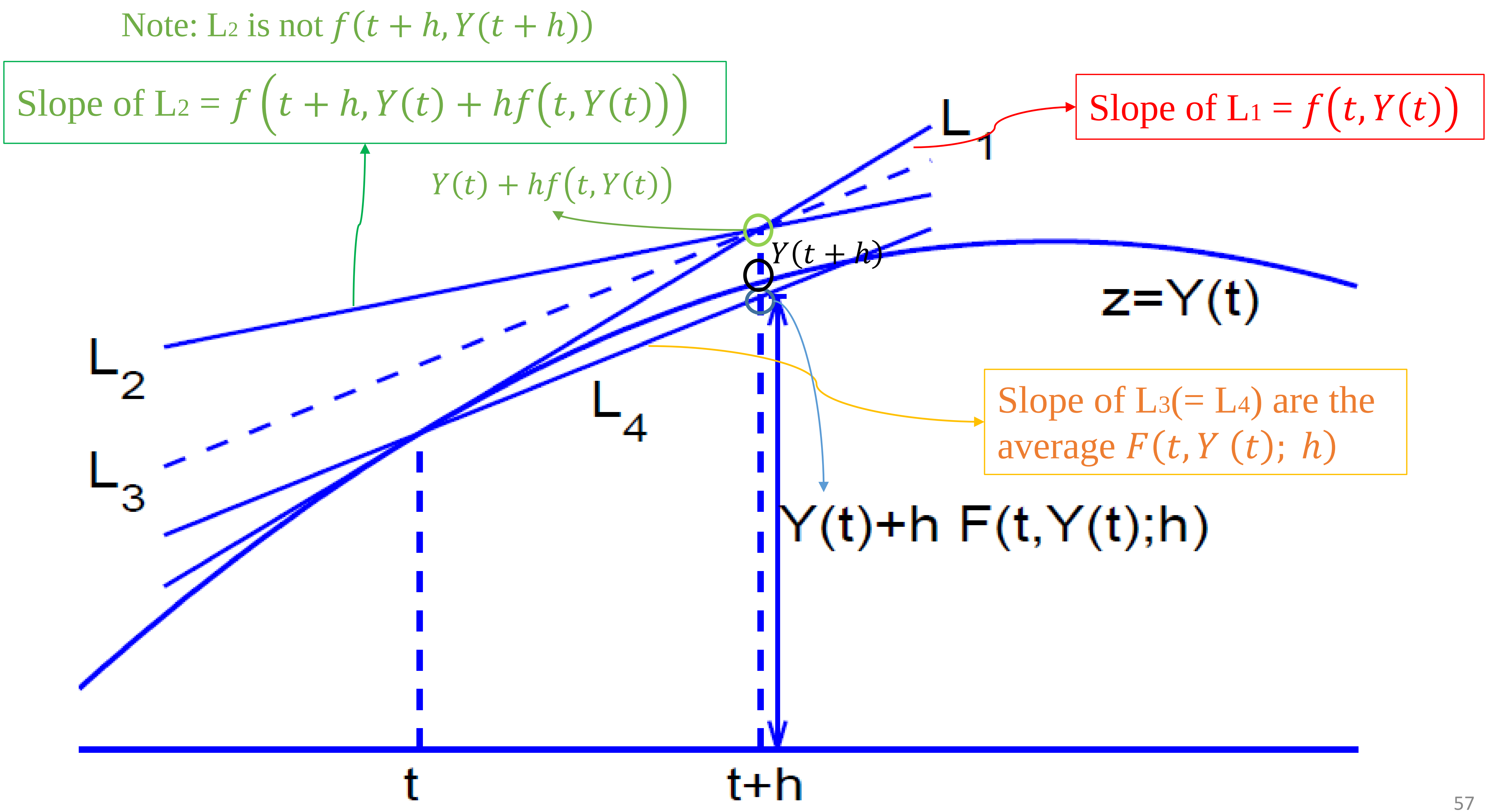
$$\underbrace{y_{n+1} = y_n + \frac{h}{2}[f(t_n, y_n) + f(t_n + h, \boxed{y_n + hf(t_n, y_n)})],}_{\text{Heun's method}} \qquad n \geq 0.$$

$$F(t_n, y_n; h) = \tfrac{1}{2}[f(t_n, y_n) + f(t_n + h, y_n + hf(t_n, y_n))]$$

is an "average" slope of the solution on the interval $[t_n, t_{n+1}]$.

Another choice is to use $b_2 = 1$, resulting in the numerical method

$$y_{n+1} = y_n + hf\left(t_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}hf(t_n, y_n)\right).$$



Note: L_2 is not $f(t+h, Y(t+h))$
Slope of $L_2 = f\left(t+h, Y(t) + hf\left(t, Y(t)\right)\right)$
Slope of $L_1 = f\left(t, Y(t)\right)$
$Y(t) + hf\left(t, Y(t)\right)$
L_1
$Y(t+h)$
$z=Y(t)$
L_2
L_3
L_4
Slope of $L_3 (= L_4)$ are the average $F(t, Y(t);\ h)$
$Y(t)+h\,F(t,Y(t);h)$
t
$t+h$

Example For the problem

$$Y'(t) = -Y(t) + 2\cos(t), \qquad Y(0) = 1,$$

whose true solution is $Y(t) = \sin(t) + \cos(t)$.

Results of the 2-or Runge–Kutta method

h	t	$y_h(t)$	Error
0.1	2.0	0.491215673	1.93e − 3
	4.0	−1.407898629	−2.55e − 3
	6.0	0.680696723	5.81e − 5
	8.0	0.841376339	2.48e − 3
	10.0	−1.380966579	$\boxed{-2.13e-3}$
0.05	2.0	0.492682499	4.68e − 4
	4.0	−1.409821234	−6.25e − 4
	6.0	0.680734664	2.01e − 5
	8.0	0.843254396	6.04e − 4
	10.0	−1.382569379	$\boxed{-5.23e-4}$

2-or Taylor method

Error	Euler Error
9.25e − 4	−4.64e − 2
1.21e − 3	3.91e − 2
−1.67e − 3	1.39e − 2
2.09e − 4	−5.07e − 2
1.50e − 3	2.83e − 2
2.31e − 4	−2.30e − 2
2.91e − 4	1.92e − 2
−4.08e − 4	6.97e − 3
5.68e − 5	−2.50e − 2
3.62e − 4	1.39e − 2

A General Framework for Explicit Runge–Kutta Methods

An explicit Runge–Kutta formula with *s* stages has the following form:

$$
\begin{aligned}
z_1 &= y_n, \\
z_2 &= y_n + h a_{2,1} f(t_n, z_1), \\
z_3 &= y_n + h\left[a_{3,1} f(t_n, z_1) + a_{3,2} f(t_n + c_2 h, z_2)\right], \\
&\vdots \\
z_s &= y_n + h\left[a_{s,1} f(t_n, z_1) + a_{s,2} f(t_n + c_2 h, z_2)\right. \\
&\qquad \left. + \cdots + a_{s,s-1} f(t_n + c_{s-1} h, z_{s-1})\right],
\end{aligned}
$$

$$
\begin{aligned}
y_{n+1} = y_n + h\,&\left[b_1 f(t_n, z_1) + b_2 f(t_n + c_2 h, z_2)\right. \\
&\left. + \cdots + b_{s-1} f(t_n + c_{s-1} h, z_{s-1}) + b_s f(t_n + c_s h, z_s)\right].
\end{aligned}
$$

More succinctly

$$z_i = y_n + h \sum_{j=1}^{i-1} a_{i,j} f(t_n + c_j h, z_j)\,, \qquad i = 1, \ldots, s,$$

$$y_{n+1} = y_n + h \sum_{j=1}^{s} b_j f(t_n + c_j h, z_j)\,.$$

The coefficients are often displayed in a table called a **Butcher tableau**

$$\begin{array}{c|ccccc}
0 = c_1 & & & & & \\
c_2 & a_{2,1} & & & & \\
c_3 & a_{3,1} & a_{3,2} & & & \\
\vdots & \vdots & & \ddots & & \\
c_s & a_{s,1} & a_{s,2} & \cdots & a_{s,s-1} & \\
\hline
 & b_1 & b_2 & \cdots & b_{s-1} & b_s
\end{array}$$

The coefficients $\{c_i\}$ and $\{a_{i,j}\}$ are usually assumed to satisfy the conditions

$$\sum_{j=1}^{i-1} a_{i,j} = c_i, \qquad i = 2, \ldots, s.$$

Example

Heun's method

$$y_{n+1} = y_n + \frac{h}{2}[f(t_n, y_n) + f(t_n + h, y_n + hf(t_n, y_n))]$$

$$\begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & 1/2 & 1/2 \end{array}$$

Fourth-order RK method

$$\begin{aligned} z_1 &= y_n, \\ z_2 &= y_n + \tfrac{1}{2}h\, f\left(t_n, z_1\right), \\ z_3 &= y_n + \tfrac{1}{2}h\, f\left(t_n + \tfrac{1}{2}h, z_2\right), \\ z_4 &= y_n + h\, f\left(t_n + \tfrac{1}{2}h, z_3\right), \end{aligned}$$

$$\begin{aligned} y_{n+1} = y_n + \tfrac{1}{6}h \Big[& f\left(t_n, z_1\right) + 2f\left(t_n + \tfrac{1}{2}h, z_2\right) \\ & + 2f\left(t_n + \tfrac{1}{2}h, z_3\right) + f\left(t_n + h, z_4\right)\Big]. \end{aligned}$$

$$\begin{array}{c|cccc} 0 & & & & \\ 1/2 & 1/2 & & & \\ 1/2 & 0 & 1/2 & & \\ 1 & 0 & 0 & 1 & \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array}$$

Convergence of the Runge-Kutta Method

We want to examine the convergence of the Runge-Kutta method

$$y_{n+1} = y_n + hF(t_n, y_n; h), \qquad n \geq 0, \qquad y_0 = Y_0$$

We want the truncation error

$$\tau_n(Y) = \frac{Y(t_{n+1}) - Y(t_n)}{h} - F(t_n, Y(t_n), h; f) \quad \to 0$$

we require that

$$F(t, Y(t), h; f) \to Y'(t) = f(t, Y(t)) \qquad \text{as } h \to 0.$$

Accordingly, define

$$\delta(h) = \sup_{\substack{t_0 \leq t \leq b \\ -\infty < y < \infty}} |f(t, y) - F(t, y, h; f)|,$$

consistency condition

and assume

①

$$\boxed{\delta(h) \to 0 \qquad \text{as } h \to 0.}$$

We also need a Lipschitz condition on F

② $$\boxed{|F(t, y, h; f) - F(t, z, h; f)| \leq L\,|y - z|}$$

for all $t_0 \leq t \leq b$, $-\infty < y, z < \infty$, and all small $h > 0$.

Theorem Assume that the Runge–Kutta method satisfies the Lipschitz condition. Then, for the initial value problem, the solution $\{y_n\}$ satisfies

$$\max_{t_0 \leq t_n \leq b} |Y(t_n) - y_n| \leq e^{(b-t_0)L}\,|Y_0 - y_0| + \left[\frac{e^{(b-t_0)L} - 1}{L}\right]\tau(h),$$

where

$$\tau(h) \equiv \max_{t_0 \leq t_n \leq b} |\tau_n(Y)|\,.$$

If the consistency condition is satisfied, then the numerical solution $\{y_n\}$ converges to Y(t).

$$h\tau_n(Y) = Y(t_{n+1}) - Y(t_n) - hF(t_n, Y(t_n), h; f)$$

Taylor expansion

$$= \underline{hY'(t_n)} + \frac{h^2}{2}Y''(\xi_n) \underline{- hF(t_n, Y(t_n), h; f)},$$

$$h\,|\tau_n(Y)| \leq h\delta(h) + \frac{h^2}{2}\,\|Y''\|_\infty\,,$$

$$\tau(h) \leq \delta(h) + \frac{1}{2}h\,\|Y''\|_\infty\,.$$

Thus $\tau(h) \to 0$ as $h \to 0$

Corollary If the Runge–Kutta method has a truncation error $T_n(Y) = \mathcal{O}(h^{m+1})$, then the error in the convergence of $\{y_n\}$ *to* $Y(t)$ *on* $[t_0, b]$ *is* $\mathcal{O}(h^m)$.

Example Consider the problem

$$Y' = \frac{1}{1+x^2} - 2Y^2, \qquad Y(0) = 0$$

with the solution $Y = x/(1+x^2)$. The stepsizes are $h = 0.25$ and $2h = 0.5$.

Fourth-order Runge-Kutta method

$$z_1 = y_n,$$

$$z_2 = y_n + \tfrac{1}{2}h\,f\left(t_n, z_1\right),$$

$$z_3 = y_n + \tfrac{1}{2}h\,f\left(t_n + \tfrac{1}{2}h, z_2\right),$$

$$z_4 = y_n + h\,f\left(t_n + \tfrac{1}{2}h, z_3\right),$$

$$y_{n+1} = y_n + \tfrac{1}{6}h\left[f\left(t_n, z_1\right) + 2f\left(t_n + \tfrac{1}{2}h, z_2\right) + 2f\left(t_n + \tfrac{1}{2}h, z_3\right) + f\left(t_n + h, z_4\right)\right].$$

Results of fourth-order Runge-Kutta method

x	$y_h(x)$	$Y(x) - y_h(x)$	$Y(x) - y_{2h}(x)$	Ratio
2.0	0.39995699	4.3e − 5	1.0e − 3	24
4.0	0.23529159	2.5e − 6	7.0e − 5	28
6.0	0.16216179	3.7e − 7	1.2e − 5	32
8.0	0.12307683	9.2e − 8	3.4e − 6	36
10.0	0.09900987	3.1e − 8	1.3e − 6	41

Not accurate enough

Runge–Kutta–Fehlberg Methods

- Currently most popular Runge–Kutta methods (Matlab code ode45.m).
- Simultaneously computes by using two methods of different orders
- The two methods share most of the function evaluations of f at each step from t_n to t_{n+1}.

Define six intermediate slopes in $[t_n, t_{n+1}]$ by

$$v_0 = f(t_n, y_n),$$

$$v_i = f\left(t_n + \alpha_i h, y_n + h\sum_{j=0}^{i-1}\beta_{ij}v_j\right), \quad i = 1, 2, 3, 4, 5.$$

Then the fourth- and fifth-order formulas are given by

$$y_{n+1} = y_n + h\sum_{i=0}^{4}\gamma_i v_i,$$

$$\hat{y}_{n+1} = y_n + h\sum_{i=0}^{5}\delta_i v_i.$$

i	0	1	2	3	4	5
γ_i	$\frac{25}{216}$	0	$\frac{1408}{2565}$	$\frac{2197}{4104}$	$-\frac{1}{5}$	
δ_i	$\frac{16}{135}$	0	$\frac{6656}{12825}$	$\frac{28561}{56430}$	$-\frac{9}{50}$	$\frac{2}{55}$

Example Consider the problem

$$Y' = \frac{1}{1+x^2} - 2Y^2, \qquad Y(0) = 0$$

with the solution $Y = x/(1+x^2)$. The stepsizes are $h = 0.25$ and $2h = 0.5$.

Results of fourth-order of Fehlberg method

h	t	$y_h(t)$	$Y(t) - y_h(t)$
0.25	2.0	0.493156301	$-5.71e-6$
	4.0	-1.410449823	$3.71e-6$
	6.0	0.680752304	$2.48e-6$
	8.0	0.843864007	$-5.79e-6$
	10.0	-1.383094975	$2.34e-6$
0.125	2.0	0.493150889	$-2.99e-7$
	4.0	-1.410446334	$2.17e-7$
	6.0	0.680754675	$1.14e-7$
	8.0	0.843858525	$-3.12e-7$
	10.0	-1.383092786	$1.46e-7$

The s-stage explicit Runge–Kutta method

$$z_i = y_n + h \sum_{j=1}^{\boxed{i-1}} a_{i,j} f(t_n + c_j h, z_j)\,,$$

$$y_{n+1} = y_n + h \sum_{j=1}^{s} b_j f(t_n + c_j h, z_j)\,.$$

$$\begin{array}{c|ccccc}
0 = c_1 & & & & & \\
c_2 & a_{2,1} & & & & \\
c_3 & a_{3,1} & a_{3,2} & & & \\
\vdots & \vdots & & \ddots & & \\
c_s & a_{s,1} & a_{s,2} & \cdots & a_{s,s-1} & \\
\hline
 & b_1 & b_2 & \cdots & b_{s-1} & b_s
\end{array}$$

The s-stage implicit Runge–Kutta method

$$z_i = y_n + h \sum_{j=1}^{\boxed{s}} a_{i,j} f(t_n + c_j h, z_j)\,,$$

$$y_{n+1} = y_n + h \sum_{j=1}^{s} b_j f(t_n + c_j h, z_j)\,.$$

The equations form a simultaneous system of s nonlinear equations for the *s* unknowns $z_1, \ldots, z_s$:

$$\begin{array}{c|ccc}
c_1 & a_{1,1} & \cdots & a_{1,s} \\
c_2 & a_{2,1} & \cdots & a_{2,s} \\
\vdots & \vdots & & \vdots \\
c_s & a_{s,1} & \cdots & a_{s,s} \\
\hline
 & b_1 & \cdots & b_s
\end{array}$$

Stability!

How to derive implicit methods? --- Integral methods

Integrating the equation $Y'(t) = f(t, Y(t))$ over the interval $[t_n, t]$,

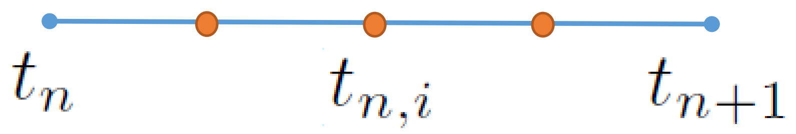
$$\int_{t_n}^{t} Y'(r)\, dr = \int_{t_n}^{t} f(r, Y(r))\, dr,$$

$$Y(t) = Y(t_n) + \int_{t_n}^{t} f(r, Y(r))\, dr.$$

Approximate the equation

- replacing $Y(t_n)$ with y_n
- replacing the integrand with a polynomial interpolant of it

Let $p(r)$ be the unique polynomial of degree $< s$ that interpolates $f(r, Y(r))$ at the node points $\{t_{n,i} \equiv t_n + \tau_i h : i = 1, \ldots, s\}$ on $[t_n, t_{n+1}]$; $0 \le \tau_1 < \cdots < \tau_s \le 1$.



The integral equation is then approximated by

$$Y(t) \approx y_n + \int_{t_n}^{t} p(r)\,dr \quad \text{where} \quad \begin{cases} p(r) = \sum_{j=1}^{s} f(t_{n,j}, Y(t_{n,j}))l_j(r). \\ l_i(x) = \prod_{j \neq i} \left(\frac{x - x_j}{x_i - x_j} \right), \qquad i = 0, 1, \ldots, n. \end{cases}$$

Forcing equality in the expression and let $\{y_{n,j}\}$ denote the approximate values to be determined by solving the nonlinear system

$$y_{n,i} = y_n + \sum_{j=1}^{\boxed{s}} f(t_{n,j}, y_{n,j}) \int_{t_n}^{t_{n,i}} l_j(r)\,dr, \qquad i = 1, \ldots, s.$$

If $\tau_s = 1$, then we define $y_{n+1} = y_{n,s}$. Otherwise, we define

$$y_{n+1} = y_n + \sum_{j=1}^{s} f(t_{n,j}, y_{n,j}) \int_{t_n}^{t_{n+1}} l_j(r)\,dr.$$

Two-point Collocation Methods (implicit RK)

Let $0 \leq \tau_1 < \tau_2 \leq 1$, and recall that $t_{n,1} = t_n + h\tau_1$ and $t_{n,2} = t_n + h\tau_2$. Then the interpolation polynomial is

$$p(r) = \frac{1}{h(\tau_2 - \tau_1)} \left[(t_{n+1} - r) f(t_{n,1}, Y(t_{n,1})) + (r - t_n) f(t_{n,2}, Y(t_{n,2})) \right].$$

$$y_{n,i} = y_n + \sum_{j=1}^{s} f(t_{n,j}, y_{n,j}) \boxed{\int_{t_n}^{t_{n,i}} l_j(r)\, dr}$$

$$z_i = y_n + \boxed{h} \sum_{j=1}^{s} \boxed{a_{i,j}} f(t_n + c_j h, z_j)$$

Implicit RK formula

$$\begin{array}{c|cc}
\tau_1 & (\tau_2^2 - [\tau_2 - \tau_1]^2)/(2[\tau_2 - \tau_1]) & -\tau_1^2/(2[\tau_2 - \tau_1]) \\
\tau_2 & \tau_2^2/(2[\tau_2 - \tau_1]) & ([\tau_2 - \tau_1]^2 - \tau_1^2)/(2[\tau_2 - \tau_1]) \\
\hline
 & (\tau_2^2 - [1 - \tau_2]^2)/(2[\tau_2 - \tau_1]) & ([1 - \tau_1]^2 - \tau_1^2)/(2[\tau_2 - \tau_1])
\end{array}$$

Implicit RK method → Trapezoidal method

$$
\begin{array}{c|cc}
\tau_1 & \left(\tau_2^2 - [\tau_2 - \tau_1]^2\right) / \left(2\,[\tau_2 - \tau_1]\right) & -\tau_1^2 / \left(2\,[\tau_2 - \tau_1]\right) \\
\tau_2 & \tau_2^2 / \left(2\,[\tau_2 - \tau_1]\right) & \left([\tau_2 - \tau_1]^2 - \tau_1^2\right) / \left(2\,[\tau_2 - \tau_1]\right) \\
\hline
 & \left(\tau_2^2 - [1 - \tau_2]^2\right) / \left(2\,[\tau_2 - \tau_1]\right) & \left([1 - \tau_1]^2 - \tau_1^2\right) / \left(2\,[\tau_2 - \tau_1]\right)
\end{array}
$$

when $\tau_1 = 0$ and $\tau_2 = 1$

$$
\begin{aligned}
y_{n,1} &= y_n, \\
y_{n,2} &= y_n + \tfrac{1}{2} h \left[f(t_n, y_{n,1}) + f(t_{n+1}, y_{n,2}) \right].
\end{aligned}
$$

Substituting from the first equation into the second equation and using $y_{n+1} = y_{n,2}$, we have

$$
y_{n+1} = y_n + \tfrac{1}{2} h \left[f(t_n, y_n) + f(t_{n+1}, y_{n+1}) \right],
$$

which is simply the trapezoidal method.

Another choice is to use $\tau_1 = \frac{1}{2} - \frac{1}{6}\sqrt{3}, \qquad \tau_2 = \frac{1}{2} + \frac{1}{6}\sqrt{3}.$

The Butcher tableau is

$$\begin{array}{c|cc} \left(3-\sqrt{3}\right)/6 & 1/4 & \left(3-2\sqrt{3}\right)/12 \\ \left(3+\sqrt{3}\right)/6 & \left(3+2\sqrt{3}\right)/12 & 1/4 \\ \hline & 1/2 & 1/2 \end{array}$$

The associated nonlinear system is

$$\boxed{\begin{aligned} y_{n,i} &= y_n + \sum_{j=1}^{2} a_{i,j} f(t_n + \tau_j h, y_{n,j}), \qquad i = 1, 2, \\ y_{n+1} &= y_n + \frac{h}{2}\left[f(t_{n+1}, y_{n,1}) + f(t_{n+1}, y_{n,2})\right]. \end{aligned}}$$

Two stage Gauss method

- Truncation error for this method has size $\mathcal{O}(h^5)$.
- The convergence is $\mathcal{O}(h^4)$