Numerical Analysis II

Xiaojing Ye, Math & Stat, Georgia State University

Spring 2020
Section 1

Initial Value Problems for ODEs
We study numerical solution for initial value problem (IVP) of ordinary differential equations (ODE).

▶ A basic IVP:

\[
\frac{dy}{dt} = f(t, y), \quad \text{for } a \leq t \leq b
\]

with initial value \(y(a) = \alpha\).

Remark

▶ \(f\) is given and called the defining function of IVP.
▶ \(\alpha\) is given and called the initial value.
▶ \(y(t)\) is called the solution of the IVP if
  ▶ \(y(a) = \alpha\);
  ▶ \(y'(t) = f(t, y(t))\) for all \(t \in [a, b]\).
Example

The following is a basic IVP:

\[ y' = y - t^2 + 1, \quad t \in [0, 2], \text{ and } y(0) = 0.5 \]

- The defining function is \( f(t, y) = y - t^2 + 1 \).
- Initial value is \( y(0) = 0.5 \).
- The solution is \( y(t) = (t + 1)^2 - \frac{e^t}{2} \) because:
  - \( y(0) = (0 + 1)^2 - \frac{e^0}{2} = 1 - \frac{1}{2} = \frac{1}{2} \);
  - We can check that \( y'(t) = f(t, y(t)) \):
    \[
    y'(t) = 2(t + 1) - \frac{e^t}{2}
    \]
    \[
    f(t, y(t)) = y(t) - t^2 + 1 = (t + 1)^2 - \frac{e^t}{2} - t^2 + 1 = 2(t + 1) - \frac{e^t}{2}
    \]
IVP of ODE (cont.)

More general or complex cases:

- IVP of ODE system:

\[
\begin{align*}
\frac{dy_1}{dt} &= f_1(t, y_1, y_2, \ldots, y_n) \\
\frac{dy_2}{dt} &= f_2(t, y_1, y_2, \ldots, y_n) \\
&\vdots \\
\frac{dy_n}{dt} &= f_n(t, y_1, y_2, \ldots, y_n)
\end{align*}
\]

for \( a \leq t \leq b \)

with initial value \( y_1(a) = \alpha_1, \ldots, y_n(a) = \alpha_n \).

- High-order ODE:

\[ y^{(n)} = f(t, y, y', \ldots, y^{(n-1)}) \quad \text{for} \quad a \leq t \leq b \]

with initial value \( y(a) = \alpha_1, y'(a) = \alpha_2, \ldots, y^{(n-1)}(a) = \alpha_n \).
Why numerical solutions for IVP?

- ODEs have extensive applications in real-world: science, engineering, economics, finance, public health, etc.
- Analytic solution? Not with almost all ODEs.
- Fast improvement of computers.
Definition (Lipschitz functions)

A function $f(t, y)$ defined on $D = \{(t, y) : t \in \mathbb{R}^+, y \in \mathbb{R}\}$ is called **Lipschitz with respect to** $y$ if there exists a constant $L > 0$

$$|f(t, y_1) - f(t, y_2)| \leq L|y_1 - y_2|$$

for all $t \in \mathbb{R}^+$, and $y_1, y_2 \in \mathbb{R}$.

**Remark**

We also call $f$ is Lipschitz with respect to $y$ with constant $L$, or simply $f$ is $L$-Lipschitz with respect to $y$. 

Some basics about IVP

Example

*Function* $f(t, y) = t|y|$ *is Lipschitz with respect to* $y$ *on the set* $D := \{(t, y) | t \in [1, 2], y \in [-3, 4]\}$.

**Solution:** For any $t \in [1, 2]$ and $y_1, y_2 \in [-3, 4]$, we have

$$|f(t, y_1) - f(t, y_2)| = |t|y_1| - t|y_2| | \leq t|y_1 - y_2| \leq 2|y_1 - y_2|.$$

*So* $f(t, y) = t|y|$ *is Lipschitz with respect to* $y$ *with constant* $L = 2$. 
Some basics about IVP

Definition (Convex sets)

A set \( D \in \mathbb{R}^2 \) is **convex** if whenever \((t_1, y_1), (t_2, y_2) \in D\) there is \((1 - \lambda)(t_1, y_1) + \lambda(t_2, y_2) \in D\) for all \( \lambda \in [0, 1] \).
Some basics about IVP

Theorem
If $D \subset \mathbb{R}^2$ is convex, and $|\frac{\partial f}{\partial y}(t,y)| \leq L$ for all $(t,y) \in D$, then $f$ is Lipschitz with respect to $y$ with constant $L$.

Remark
This is a sufficient (but not necessary) condition for $f$ to be Lipschitz with respect to $y$. 
Some basics about IVP

Proof.
For any \((t, y_1), (t, y_2) \in D\), define function \(g\) by
\[
g(\lambda) = f(t, (1 - \lambda)y_1 + \lambda y_2)
\]
for \(\lambda \in [0, 1]\) (need convexity of \(D\)). Then we have
\[
g'(\lambda) = \partial_y f(t, (1 - \lambda)y_1 + \lambda y_2) \cdot (y_2 - y_1)
\]
So \(|g'(\lambda)| \leq L|y_2 - y_1|\). Then we have
\[
|g(1) - g(0)| = \left| \int_0^1 g'(\lambda) \, d\lambda \right| \leq L|y_2 - y_1| \left| \int_0^1 d\lambda \right| = L|y_2 - y_1|
\]
Note that \(g(0) = f(t, y_1)\) and \(g(1) = f(t, y_2)\). This completes the proof. \(\square\)
Some basics about IVP

Theorem
Suppose $D = [a, b] \times \mathbb{R}$, a function $f$ is continuous on $D$ and Lipschitz with respect to $y$, then the initial value problem $y' = f(t, y)$ for $t \in [a, b]$ with initial value $y(a) = \alpha$ has a unique solution $y(t)$ for $t \in [a, b]$.

Remark
This theorem says that there must be one and only one solution of the IVP, provided that the defining function $f$ of the IVP is continuous and Lipschitz with respect to $y$ on $D$. 
Some basics about IVP

Example

Show that \( y' = 1 + t \sin(ty) \) for \( t \in [0, 2] \) with \( y(0) = 0 \) has a unique solution.

Solution: First, we know \( f(t, y) = 1 + t \sin(ty) \) is continuous on \([0, 2] \times \mathbb{R}\). Second, we can see

\[
\left| \frac{\partial f}{\partial y} \right| = \left| t^2 \cos(ty) \right| \leq |t^2| \leq 4
\]

So \( f(t, y) \) is Lipschitz with respect to \( y \) (with constant 4). From theorem above, we know the IVP has a unique solution \( y(t) \) on \([0, 2]\).
Some basics about IVP

Definition (Well-posedness)
An IVP $y' = f(t, y)$ for $t \in [a, b]$ with $y(a) = \alpha$ is called well-posed if

- It has a unique solution $y(t)$;
- There exist $\epsilon_0 > 0$ and $k > 0$, such that $\forall \epsilon \in (0, \epsilon_0)$ and function $\delta(t)$, which is continuous and satisfies $|\delta(t)| < \epsilon$ for all $t \in [a, b]$, the perturbed problem $z' = f(t, z) + \delta(t)$ with initial value $z(a) = \alpha + \delta_0$ (where $|\delta_0| \leq \epsilon$) satisfies

  $$|z(t) - y(t)| < k\epsilon, \quad \forall t \in [a, b].$$

Remark
This theorem says that a small perturbation on defining function $f$ by $\delta(t)$ and initial value $y(a)$ by $\delta_0$ will only cause small change to original solution $y(t)$. 
Some basics about IVP

**Theorem**

Let \( D = [a, b] \times \mathbb{R} \). If \( f \) is continuous on \( D \) and Lipschitz with respect to \( y \), then the IVP is well-posed.

**Remark**

Again, a sufficient but not necessary condition for well-posedness of IVP.
Euler’s method

Given an IVP \( y' = f(t, y) \) for \( t \in [a, b] \) and \( y(a) = \alpha \), we want to compute \( y(t) \) on mesh points \( \{t_0, t_1, \ldots, t_N\} \) on \([a, b]\).

To this end, we partition \([a, b]\) into \( N \) equal segments: set \( h = \frac{b-a}{N} \), and define \( t_i = a + ih \) for \( i = 0, 1, \ldots, N \). Here \( h \) is called the step size.
Euler’s method

From Taylor’s theorem, we have

\[ y(t_{i+1}) = y(t_i) + y'(t_i)(t_{i+1} - t_i) + \frac{1}{2} y''(\xi_i)(t_{i+1} - t_i)^2 \]

for some \( \xi_i \in (t_i, t_{i+1}) \). Note that \( t_{i+1} - t_i = h \) and \( y'(t_i) = f(t_i, y(t_i)) \), we get

\[ y(t_{i+1}) \approx y(t_i) + hf(t, y(t_i)) \]

Denote \( w_i = y(t_i) \) for all \( i = 0, 1, \ldots, N \), we get the Euler’s method:

\[
\begin{align*}
w_0 &= \alpha \\
\begin{aligned}
\frac{dx}{dt} &= f(t, x) \\
\frac{dy}{dt} &= g(t, y) \\
\frac{dz}{dt} &= h(t, z)
\end{aligned}
\end{align*}
\]

\[
\begin{align*}
\frac{dx}{dt} &= f(t, x) \\
\frac{dy}{dt} &= g(t, y) \\
\frac{dz}{dt} &= h(t, z)
\end{align*}
\]

\[
\begin{align*}
\frac{dx}{dt} &= f(t, x) \\
\frac{dy}{dt} &= g(t, y) \\
\frac{dz}{dt} &= h(t, z)
\end{align*}
\]
Euler’s method

$y' = f(t, y), y(a) = \alpha$

Slope $y'(a) = f(a, \alpha)$

$y(b)$

$w_1$

$\alpha$

$t_0 = a, t_1, t_2, \ldots, t_N = b$

$t$

$w_2$

$w_1$

$\alpha$

$t_0 = a, t_1, t_2, \ldots, t_N = b$

$t$
Euler’s method

Example

*Use Euler’s method with* $h = 0.5$ *for IVP* $y' = y - t^2 + 1$ *for* $t \in [0, 2]$ *with initial value* $y(0) = 0.5$.

**Solution:** We follow Euler’s method step-by-step:

$t_0 = 0 : \quad w_0 = y(0) = 0.5$

$t_1 = 0.5 : \quad w_1 = w_0 + hf(t_0, w_0) = 0.5 + 0.5 \times (0.5 - 0^2 + 1) = 1.25$

$t_2 = 1.0 : \quad w_2 = w_1 + hf(t_1, w_1) = 1.25 + 0.5 \times (1.25 - 0.5^2 + 1) = 2.25$

$t_3 = 1.5 : \quad w_3 = w_2 + hf(t_2, w_2) = 2.25 + 0.5 \times (2.25 - 1^2 + 1) = 3.375$

$t_4 = 2.0 : \quad w_4 = w_3 + hf(t_3, w_3) = 3.375 + 0.5 \times (3.375 - 1.5^2 + 1) = 4.4375$
Error bound of Euler’s method

Theorem
Suppose \( f(t, y) \) in an IVP is continuous on \( D = [a, b] \times \mathbb{R} \) and Lipschitz with respect to \( y \) with constant \( L \). If \( \exists M > 0 \) such that \( |y''(t)| \leq M \) (\( y(t) \) is the unique solution of the IVP), then for all \( i = 0, 1, \ldots, N \) there is

\[
|y(t_i) - w_i| \leq \frac{hM}{2L} \left( e^{L(t_i-a)} - 1 \right)
\]

Remark

- Numerical error depends on \( h \) (also called \( O(h) \) error).
- Also depends on \( M, L \) of \( f \).
- Error increases for larger \( t_i \).
Error bound of Euler’s method

**Proof.** Taking the difference of

\[ y(t_{i+1}) = y(t_i) + hf(t_i, y_i) + \frac{1}{2} y''(\xi_i)(t_{i+1} - t_i)^2 \]

\[ w_{i+1} = w_i + hf(t_i, w_i) \]

we get

\[ |y(t_{i+1}) - w_{i+1}| \leq |y(t_i) - w_i| + h|f(t_i, y_i) - f(t_i, w_i)| + \frac{Mh^2}{2} \]

\[ \leq |y(t_i) - w_i| + hL|y_i - w_i| + \frac{Mh^2}{2} \]

\[ = (1 + hL)|y_i - w_i| + \frac{Mh^2}{2} \]
Error bound of Euler’s method

Proof (cont).
Denote \( d_i = |y(t_i) - w_i| \), then we have

\[
d_{i+1} \leq (1 + hL)d_i + \frac{Mh^2}{2} = (1 + hL)\left(d_i + \frac{hM}{2L}\right) - \frac{hM}{2L}
\]

for all \( i = 0, 1, \ldots, N - 1 \). So we obtain

\[
d_{i+1} + \frac{hM}{2L} \leq (1 + hL)\left(d_i + \frac{hM}{2L}\right)
\]

\[
\leq (1 + hL)^2 \left(d_{i-1} + \frac{hM}{2L}\right)
\]

\[
\leq \cdots \leq (1 + hL)^{i+1} \left(d_0 + \frac{hM}{2L}\right)
\]

and hence \( d_i \leq (1 + hL)^i \cdot \frac{hM}{2L} - \frac{hM}{2L} \) (since \( d_0 = 0 \)).
Proof (cont).
Note that $1 + x \leq e^x$ for all $x > -1$, and hence $(1 + x)^a \leq e^{ax}$ if $a > 0$.
Based on this, we know $(1 + hL)^i \leq e^{ihL} = e^{L(t_i - a)}$ since $ih = t_i - a$. Therefore we get

$$d_i \leq e^{L(t_i - a)} \cdot \frac{hM}{2L} - \frac{hM}{2L} = \frac{hM}{2L} \left(e^{L(t_i - a)} - 1\right)$$

This completes the proof.
Error bound of Euler’s method

Example

Estimate the error of Euler’s method with $h = 0.2$ for IVP $y' = y - t^2 + 1$ for $t \in [0, 2]$ with initial value $y(0) = 0.5$.

Solution: We first note that $\frac{\partial f}{\partial y} = 1$, so $f$ is Lipschitz with respect to $y$ with constant $L = 1$. The IVP has solution $y(t) = (t - 1)^2 - \frac{e^t}{2}$ so $|y''(t)| = \left|\frac{e^t}{2} - 2\right| \leq \frac{e^2}{2} - 2 =: M$. By theorem above, the error of Euler’s method is

$$|y(t_i) - w_i| \leq \frac{hM}{2L} \left(e^{L(t_i-a)} - 1\right) = \frac{0.2(0.5e^2 - 2)}{2} \left(e^{t_i} - 1\right)$$
Error bound of Euler’s method

Example

Estimate the error of Euler’s method with \( h = 0.2 \) for IVP \( y' = y - t^2 + 1 \) for \( t \in [0, 2] \) with initial value \( y(0) = 0.5 \).

Solution: (cont)

| \( t_i \) | \( w_i \) | \( y_i = y(t_i) \) | \( |y_i - w_i| \) |
|---|---|---|---|
| 0.0 | 0.5000000 | 0.5000000 | 0.0000000 |
| 0.2 | 0.8000000 | 0.8292986 | 0.0292986 |
| 0.4 | 1.1520000 | 1.2140877 | 0.0620877 |
| 0.6 | 1.5504000 | 1.6489406 | 0.0985406 |
| 0.8 | 1.9884800 | 2.1272295 | 0.1387495 |
| 1.0 | 2.4581760 | 2.6408591 | 0.1826831 |
| 1.2 | 2.9498112 | 3.1799415 | 0.2301303 |
| 1.4 | 3.4517734 | 3.7324000 | 0.2806266 |
| 1.6 | 3.9501281 | 4.2834838 | 0.3333557 |
| 1.8 | 4.4281538 | 4.8151763 | 0.3870225 |
| 2.0 | 4.8657845 | 5.3054720 | 0.4396874 |

Note that the error grows slightly as the value of \( t \) increases. This controlled error growth is a consequence of the stability of Euler’s method, which implies that the error is expected to grow in no worse than a linear manner.

Maple has implemented Euler’s method as an option with the command \( \text{InitialValueProblem} \) within the \( \text{NumericalAnalysis} \) subpackage of the \( \text{Student} \) package. To use it for the problem in Example 1 first load the package and the differential equation.

```maple
with(Student[NumericalAnalysis]):
deq := diff(y(t), t) = y(t) - t^2 + 1;
C := InitialValueProblem(deq, y(0) = 0.5, t = 2, method = euler, numsteps = 10, output = information, digits = 8);
```

Maple produces the following output:

\[
\begin{bmatrix}
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array}
\end{bmatrix}
\]

Double clicking on the output brings up a table that gives the values of \( t_i \), actual solution values \( y_i = y(t_i) \), the Euler approximations \( w_i \), and the absolute errors \( |y_i - w_i| \). These agree with the values in Table 5.1.

To print the Maple table we can issue the commands

```maple
for k from 1 to 12 do
print(C[k,1],C[k,2],C[k,3],C[k,4]);
end do;
```

The options within the \( \text{InitialValueProblem} \) command are the specification of the first order differential equation to be solved, the initial condition, the final value of the independent variable, the choice of method, the number of steps used to determine that \( h = \frac{(2 - 0)}{10} \), the specification of form of the output, and the number of digits of rounding to be used in the computations. Other output options can specify a particular value of \( t \) or a plot of the solution.

Error Bounds for Euler’s Method

Although Euler’s method is not accurate enough to warrant its use in practice, it is sufficiently elementary to analyze the error that is produced from its application. The error analysis for Euler’s method shows that the error is bounded by \( Ch^2 \), where \( C \) is a constant and \( h \) is the step size. This bound is a consequence of the stability of Euler’s method and implies that the error grows in no worse than a quadratic manner.
Round-off error of Euler’s method

Due to round-off errors in computer, we instead obtain

\[
\begin{align*}
    u_0 &= \alpha + \delta_0 \\
    u_{i+1} &= u_i + hf(t_i, u_i) + \delta_i, \quad i = 0, 1, \ldots, N - 1
\end{align*}
\]

Suppose \( \exists \delta > 0 \) such that \( |\delta_i| \leq \delta \) for all \( i \), then we can show

\[
|y(t_i) - u_i| \leq \frac{1}{L} \left( \frac{hM}{2} + \frac{\delta}{h} \right) \left( e^{L(t_i-a)} - 1 \right) + \delta e^{L(t_i-a)}.
\]

Note that \( \frac{hM}{2} + \frac{\delta}{h} \) does not approach 0 as \( h \to 0 \). \( \frac{hM}{2} + \frac{\delta}{h} \) reaches minimum at \( h = \sqrt{\frac{2\delta}{M}} \) (often much smaller than what we choose in practice).
Higher-order Taylor’s method

Definition (Local truncation error)

We call the difference method

\[
\begin{cases}
  w_0 = \alpha \\
  w_{i+1} = w_i + h\phi(t_i, w_i), \\ i = 0, 1, \ldots, N - 1
\end{cases}
\]

**to have local truncation error**

\[
\tau_{i+1}(h) = \frac{y_{i+1} - (y_i + h\phi(t_i, y_i))}{h}
\]

where \( y_i := y(t_i) \).

Example

**Euler’s method has local truncation error**

\[
\tau_{i+1}(h) = \frac{y_{i+1} - (y_i + hf(t_i, y_i))}{h} = \frac{y_{i+1} - y_i}{h} - f(t_i, y_i)
\]
Higher-order Taylor’s method

Note that Euler’s method has local truncation error
\[ \tau_{i+1}(h) = \frac{y_{i+1} - y_i}{h} - f(t_i, y_i) = \frac{hy''(\xi_i)}{2} \text{ for some } \xi_i \in (t_i, t_{i+1}). \]
If \(|y''| \leq M\) we know \(|\tau_{i+1}(h)| \leq \frac{hM}{2} = O(h)\).

**Question:** What if we use higher-order Taylor’s approximation?

\[ y(t_{i+1}) = y(t_i) + hy'(t_i) + \frac{h^2}{2} y''(t_i) + \cdots + \frac{h^n}{n!} y^{(n)}(t_i) + R \]

where \(R = \frac{h^{n+1}}{(n+1)!} y^{(n+1)}(\xi_i) \text{ for some } \xi_i \in (t_i, t_{i+1}). \)
Higher-order Taylor’s method

First note that we can always write $y^{(n)}$ using $f(t, y(t))$:

\begin{align*}
y'(t) &= f \\
y''(t) &= f' = \frac{\partial}{\partial t} f + (\frac{\partial}{\partial y} f) f \\
y'''(t) &= f'' = \frac{\partial^2}{\partial t^2} f + (\frac{\partial}{\partial t} \frac{\partial}{\partial y} f + (\frac{\partial}{\partial y}^2 f) f) f + \frac{\partial}{\partial y} f (\frac{\partial}{\partial t} f + (\frac{\partial}{\partial y} f) f) \\
&\quad \ldots \\
y^{(n)}(t) &= f^{(n-1)} = \ldots
\end{align*}

albeit it’s quickly getting very complicated.
Higher-order Taylor’s method

Now substitute them back to high-order Taylor’s approximation (ignore residual $R$)

$$y(t_{i+1}) = y(t_i) + hy'(t_i) + \frac{h^2}{2}y''(t_i) + \cdots + \frac{h^n}{n!}y^{(n)}(t_i)$$

$$= y(t_i) + hf + \frac{h^2}{2}f' + \cdots + \frac{h^n}{n!}f^{(n-1)}$$

We can get the $n$-th order Taylor’s method:

$$\begin{align*}
w_0 &= \alpha \\
w_{i+1} &= w_i + hT^{(n)}(t_i, w_i), \quad i = 0, 1, \ldots, N - 1
\end{align*}$$

where

$$T^{(n)}(t_i, w_i) = f(t_i, w_i) + \frac{h}{2}f'(t_i, w_i) + \cdots + \frac{h^{n-1}}{n!}f^{(n-1)}(t_i, w_i)$$
Higher-order Taylor’s method

- Euler’s method is the first order Taylor’s method.
- High-order Taylor’s method is more accurate than Euler’s method, but at much higher computational cost.
- Together with Hermite interpolating polynomials, it can be used to interpolate values not on mesh points more accurately.
Higher-order Taylor’s method

Theorem

If \( y(t) \in C^{n+1}[a, b] \), then the \( n \)-th order Taylor method has local truncation error \( O(h^n) \).
Runge-Kutta (RK) method

Runge-Kutta (RK) method attains high-order local truncation error \textit{without} expensive evaluations of derivatives of $f$. 
Runge-Kutta (RK) method

To derive RK method, first recall Taylor’s formula for two variables \((t, y)\):

\[
    f(t, y) = P_n(t, y) + R_n(t, y)
\]

where \(\partial_t^{n-k} \partial_y^k f = \frac{\partial^n f(t_0, y_0)}{\partial t^{n-k} \partial y^k} \) and

\[
    P_n(t, y) = f(t_0, y_0) + (\partial_t f \cdot (t - t_0) + \partial_y f \cdot (y - y_0)) \\
    + \frac{1}{2} \left( \partial_t^2 f \cdot (t - t_0)^2 + 2\partial_y \partial_t f \cdot (t - t_0)(y - y_0) + \partial_y^2 f \cdot (y - y_0)^2 \right) \\
    + \cdots + \frac{1}{n!} \sum_{k=0}^{n} \binom{n}{k} \partial_t^{n-k} \partial_y^k f \cdot (t - t_0)^{n-k} (y - y_0)^k
\]

\[
    R_n(t, y) = \frac{1}{(n+1)!} \sum_{k=0}^{n+1} \binom{n+1}{k} \partial_t^{n+1-k} \partial_y^k f(\xi, \mu) \cdot (t - t_0)^{n+1-k} (y - y_0)^k
\]
Runge-Kutta (RK) method

The second order Taylor’s method uses

\[ T^{(2)}(t, y) = f(t, y) + \frac{h}{2}f'(t, y) = f(t, y) + \frac{h}{2}(\partial_t f + \partial_y f \cdot f) \]

to get \( O(h^2) \) error. Suppose we use \( af(t + \alpha, y + \beta) \) (with some \( a, \alpha, \beta \) to be determined) to reach the same order of error. To that end, we first have

\[ af(t + \alpha, y + \beta) = a \left( f + \partial_t f \cdot \alpha + \partial_y f \cdot \beta + R \right) \]

where \( R = \frac{1}{2}(\partial_t^2 f(\xi, \mu) \cdot \alpha^2 + 2\partial_y \partial_t f(\xi, \mu) \cdot \alpha \beta + \partial_y^2 f(\xi, \mu) \cdot \beta^2) \).
Runge-Kutta (RK) method

Suppose we try to match the terms of these two formulas (ignore $R$):

\[
T^{(2)}(t, y) = f + \frac{h}{2} \partial_t f + \frac{hf}{2} \partial_y f
\]

\[
a f(t + \alpha, y + \beta) = af + a\alpha \partial_t f + a\beta \partial_y f
\]

then we have

\[
a = 1, \quad \alpha = \frac{h}{2}, \quad \beta = \frac{h}{2} f(t, y)
\]

So instead of $T^{(2)}(t, y)$, we use

\[
a f(t + \alpha, y + \beta) = f \left( t + \frac{h}{2}, y + \frac{h}{2} f(t, y) \right)
\]
Runge-Kutta (RK) method

Note that $R$ we ignored is

$$R = \frac{1}{2} \left( \partial_t^2 f(\xi, \mu) \cdot \left( \frac{h}{2} \right)^2 + 2\partial_y \partial_t f(\xi, \mu) \cdot \left( \frac{h}{2} \right)^2 f + \partial_y^2 f(\xi, \mu) \cdot \left( \frac{h}{2} \right)^2 f^2 \right)$$

which means $R = O(h^2)$.

Also note that

$$R = T^{(2)}(t, y) - f \left( t + \frac{h}{2}, y + \frac{h}{2} f(t, y) \right) = O(h^2)$$

and the error of $T^{(2)}(t, y)$ is of $O(h^2)$, we know $f \left( t + \frac{h}{2}, y + \frac{h}{2} f(t, y) \right)$ has error of $O(h^2)$. 
Runge-Kutta (RK) method

This is the **RK2 method** (Midpoint method):

\[
\begin{align*}
    w_0 &= \alpha \\
    w_{i+1} &= w_i + h \, f \left( t_i + \frac{h}{2}, w_i + \frac{h}{2} \, f(t_i, w_i) \right), \quad i = 0, 1, \ldots, N - 1.
\end{align*}
\]

**Remark**

*If we have \((t_i, w_i)\), we only need to evaluate \(f\) twice (i.e., compute \(k_1 = f(t_i, w_i)\) and \(k_2 = f(t_i + \frac{h}{2}, w_i + \frac{h}{2} \, k_1)\)) to get \(w_{i+1}\).*
Runge-Kutta (RK) method

We can also consider higher-order RK method by fitting

\[ T^{(3)}(t,y) = f(t,y) + \frac{h}{2} f'(t,y) + \frac{h}{6} f''(t,y) \]

with \( af(t, y) + bf(t + \alpha, y + \beta) \) (has 4 parameters \( a, b, \alpha, \beta \)).

Unfortunately we can’t make match to the \( \frac{hf''}{6} \) term of \( T^{(3)} \), which contains \( \frac{h^2}{6} f \cdot (\partial_y f)^2 \), by this way. But it leaves us open choices if we’re OK with \( O(h^2) \) error: let \( a = b = 1, \alpha = h, \beta = hf(t,y) \), then we get the **modified Euler’s method**:

\[
\begin{align*}
  w_0 &= \alpha \\
  w_{i+1} &= w_i + \frac{h}{2} \left( f(t_i, w_i) + f(t_{i+1}, w_i + hf(t_i, w_i)) \right), \quad i = 0, 1, \ldots, N - 1.
\end{align*}
\]

Also need evaluation of \( f \) twice in each step.
Example

*Use Midpoint method (RK2) and Modified Euler’s method with* $h = 0.2$ *to solve IVP* $y' = y - t^2 + 1$ *for* $t \in [0, 2]$ *and* $y(0) = 0.5$.

**Solution:** Apply the main steps in the two methods:

**Midpoint:**

$$w_{i+1} = w_i + h \left( f\left(t_i + \frac{h}{2}, w_i + \frac{h}{2} f(t_i, w_i)\right) \right)$$

**Modified Euler’s:**

$$w_{i+1} = w_i + \frac{h}{2} \left( f(t_i, w_i) + f(t_{i+1}, w_i + hf(t_i, w_i)) \right)$$
Runge-Kutta (RK) method

Example

Use Midpoint method (RK2) and Modified Euler’s method with \( h = 0.2 \) to solve IVP \( y' = y - t^2 + 1 \) for \( t \in [0, 2] \) and \( y(0) = 0.5 \).

Solution: (cont)

<table>
<thead>
<tr>
<th>( t_i )</th>
<th>( y(t_i) )</th>
<th>Midpoint Method</th>
<th>Error</th>
<th>Modified Euler Method</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.5000000</td>
<td>0.5000000</td>
<td>0</td>
<td>0.5000000</td>
<td>0</td>
</tr>
<tr>
<td>0.2</td>
<td>0.8292986</td>
<td>0.8280000</td>
<td>0.0012986</td>
<td>0.8260000</td>
<td>0.0032986</td>
</tr>
<tr>
<td>0.4</td>
<td>1.2140877</td>
<td>1.2113600</td>
<td>0.0027277</td>
<td>1.2069200</td>
<td>0.0071677</td>
</tr>
<tr>
<td>0.6</td>
<td>1.6489406</td>
<td>1.6446592</td>
<td>0.0042814</td>
<td>1.6372424</td>
<td>0.0116982</td>
</tr>
<tr>
<td>0.8</td>
<td>2.1272295</td>
<td>2.1212842</td>
<td>0.0059453</td>
<td>2.1102357</td>
<td>0.0169938</td>
</tr>
<tr>
<td>1.0</td>
<td>2.6408591</td>
<td>2.6331668</td>
<td>0.0076923</td>
<td>2.6176876</td>
<td>0.0231715</td>
</tr>
<tr>
<td>1.2</td>
<td>3.1799415</td>
<td>3.1704634</td>
<td>0.0094781</td>
<td>3.1495789</td>
<td>0.0303627</td>
</tr>
<tr>
<td>1.4</td>
<td>3.7324000</td>
<td>3.7211654</td>
<td>0.0112346</td>
<td>3.6936862</td>
<td>0.0387138</td>
</tr>
<tr>
<td>1.6</td>
<td>4.2834838</td>
<td>4.2706218</td>
<td>0.0128620</td>
<td>4.2350972</td>
<td>0.0483866</td>
</tr>
<tr>
<td>1.8</td>
<td>4.8151763</td>
<td>4.8009586</td>
<td>0.0142177</td>
<td>4.7556185</td>
<td>0.0595577</td>
</tr>
<tr>
<td>2.0</td>
<td>5.3054720</td>
<td>5.2903695</td>
<td>0.0151025</td>
<td>5.2330546</td>
<td>0.0724173</td>
</tr>
</tbody>
</table>

Midpoint (RK2) method is better than modified Euler’s method.
We can also consider higher-order RK method by fitting

$$T^{(3)}(t, y) = f(t, y) + \frac{h}{2}f'(t, y) + \frac{h^2}{6}f''(t, y)$$

with $af(t, y) + bf(t + \alpha_1, y + \delta_1(f(t + \alpha_2, y + \delta_2 f(t, y)))$ (has 6 parameters $a, b, \alpha_1, \alpha_2, \delta_1, \delta_2$) to reach $O(h^3)$ error.

For example, Heun’s choice is $a = \frac{1}{4}, b = \frac{3}{4}, \alpha_1 = \frac{2h}{3}, \alpha_2 = \frac{h}{3}, \delta_1 = \frac{2h}{3}f, \delta_2 = \frac{h}{3}f$.

Nevertheless, methods of order $O(h^3)$ are rarely used in practice.
4-th Order Runge-Kutta (RK4) method

Most commonly used is the **4-th order Runge-Kutta method (RK4)**: start with \( w_0 = \alpha \), and iteratively do

\[
\begin{align*}
k_1 &= f(t_i, w_i) \\
k_2 &= f(t_i + \frac{h}{2}, w_i + \frac{h}{2}k_1) \\
k_3 &= f(t_i + \frac{h}{2}, w_i + \frac{h}{2}k_2) \\
k_4 &= f(t_{i+1}, w_i + hk_3)
\end{align*}
\]

\[
\begin{align*}
w_{i+1} &= w_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)
\end{align*}
\]

Need to evaluate \( f \) for 4 times in each step. Reach error \( O(h^4) \).
4-th Order Runge-Kutta (RK4) method

Example

Use RK4 (with $h = 0.2$) to solve IVP $y' = y - t^2 + 1$ for $t \in [0, 2]$ and $y(0) = 0.5$.

Solution: With $h = 0.2$, we have $N = 10$ and $t_i = 0.2i$ for $i = 0, 1, \ldots, 10$. First set $w_0 = 0.5$, then the first iteration is

$$k_1 = f(t_0, w_0) = f(0, 0.5) = 0.5 - 0^2 + 1 = 1.5$$
$$k_2 = f(t_0 + \frac{h}{2}, w_0 + \frac{h}{2}k_1) = f(0.1, 0.5 + 0.1 \times 1.5) = 1.64$$
$$k_3 = f(t_0 + \frac{h}{2}, w_0 + \frac{h}{2}k_2) = f(0.1, 0.5 + 0.1 \times 1.64) = 1.654$$
$$k_4 = f(t_1, w_0 + hk_3) = f(0.2, 0.5 + 0.2 \times 1.654) = 1.7908$$
$$w_1 = w_0 + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) = 0.8292933$$

So $w_1$ is our RK4 approximation of $y(t_1) = y(0.2)$. 
4-th Order Runge-Kutta (RK4) method

Example

Use RK4 (with $h = 0.2$) to solve IVP $y' = y - t^2 + 1$ for $t \in [0, 2]$ and $y(0) = 0.5$.

Solution: (cont) Continue with $i = 1, 2, \ldots, 9$:

| $t_i$ | Exact $y_i = y(t_i)$ | Runge-Kutta Order Four $w_i$ | Error $|y_i - w_i|$ |
|-------|------------------|------------------|-----------------|
| 0.0   | 0.5000000        | 0.5000000        | 0               |
| 0.2   | 0.8292986        | 0.8292933        | 0.0000053       |
| 0.4   | 1.2140877        | 1.2140762        | 0.0000114       |
| 0.6   | 1.6489406        | 1.6489220        | 0.0000186       |
| 0.8   | 2.1272295        | 2.1272027        | 0.0000269       |
| 1.0   | 2.6408591        | 2.6408227        | 0.0000364       |
| 1.2   | 3.1799415        | 3.1798942        | 0.0000474       |
| 1.4   | 3.7324000        | 3.7323401        | 0.0000599       |
| 1.6   | 4.2834838        | 4.2834095        | 0.0000743       |
| 1.8   | 4.8151763        | 4.8150857        | 0.0000906       |
| 2.0   | 5.3054720        | 5.3053630        | 0.0001089       |
High-order Runge-Kutta method

Can we use even higher-order method to improve accuracy?

<table>
<thead>
<tr>
<th>#f eval</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5 ≤ n ≤ 7</th>
<th>8 ≤ n ≤ 9</th>
<th>n ≥ 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best error</td>
<td>(O(h^2))</td>
<td>(O(h^3))</td>
<td>(O(h^4))</td>
<td>(O(h^{n-1}))</td>
<td>(O(h^{n-2}))</td>
<td>(O(h^{n-3}))</td>
</tr>
</tbody>
</table>

So RK4 is the sweet spot.

Remark

*Note that RK4 requires 4 evaluations of \(f\) each step. So it would make sense only if it’s accuracy with step size \(4h\) is higher than Midpoint with \(2h\) or Euler’s with \(h\)!*
High-order Runge-Kutta method

Example

*Use RK4 (with \( h = 0.1 \)), Midpoint (with \( h = 0.05 \)), and Euler’s method (with \( h = 0.025 \)) to solve IVP* \( y' = y - t^2 + 1 \) for \( t \in [0, 0.5] \) *and* \( y(0) = 0.5 \).

Solution:

<table>
<thead>
<tr>
<th>( t_i )</th>
<th>Exact</th>
<th>Euler ( h = 0.025 )</th>
<th>Modified Euler ( h = 0.05 )</th>
<th>Runge-Kutta Order Four ( h = 0.1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.5000000</td>
<td>0.5000000</td>
<td>0.5000000</td>
<td>0.5000000</td>
</tr>
<tr>
<td>0.1</td>
<td>0.6574145</td>
<td>0.6554982</td>
<td>0.6573085</td>
<td>0.6574144</td>
</tr>
<tr>
<td>0.2</td>
<td>0.8292986</td>
<td>0.8253385</td>
<td>0.8290778</td>
<td>0.8292983</td>
</tr>
<tr>
<td>0.3</td>
<td>1.0150706</td>
<td>1.0089334</td>
<td>1.0147254</td>
<td>1.0150701</td>
</tr>
<tr>
<td>0.4</td>
<td>1.2140877</td>
<td>1.2056345</td>
<td>1.2136079</td>
<td>1.2140869</td>
</tr>
<tr>
<td>0.5</td>
<td>1.4256394</td>
<td>1.4147264</td>
<td>1.4250141</td>
<td>1.4256384</td>
</tr>
</tbody>
</table>

RK4 is better with same computation cost!
Can we control the error of Runge-Kutta method by using variable step sizes?

Let’s compare two difference methods with errors $O(h^n)$ and $O(h^{n+1})$ (say, RK4 and RK5) for fixed step size $h$, which have schemes below:

$$w_{i+1} = w_i + h\phi(t_i, w_i, h) \quad O(h^n)$$

$$\tilde{w}_{i+1} = \tilde{w}_i + h\tilde{\phi}(t_i, \tilde{w}_i, h) \quad O(h^{n+1})$$

Suppose $w_i \approx \tilde{w}_i \approx y(t_i) =: y_i$. Then for any given $\epsilon > 0$, we want to see how small $h$ should be for the $O(h^n)$ method so that its error $|\tau_{i+1}(h)| \leq \epsilon$?
Error control

We recall that the local truncation errors of these two methods are:

\[ \tau_{i+1}(h) = \frac{y_{i+1} - y_i}{h} - \phi(t_i, y_i, h) \approx O(h^n) \]

\[ \tilde{\tau}_{i+1}(h) = \frac{y_{i+1} - y_i}{h} - \tilde{\phi}(t_i, y_i, h) \approx O(h^{n+1}) \]

Given that \( w_i \approx \tilde{w}_i \approx y_i \) and \( O(h^{n+1}) \ll O(h^n) \) for small \( h \), we see

\[ \tau_{i+1}(h) \approx \tau_{i+1}(h) - \tilde{\tau}_{i+1}(h) = \tilde{\phi}(t_i, y_i, h) - \phi(t_i, y_i, h) \]

\[ \approx \tilde{\phi}(t_i, \tilde{w}_i, h) - \phi(t_i, w_i, h) = \frac{\tilde{w}_{i+1} - \tilde{w}_i}{h} - \frac{w_{i+1} - w_i}{h} \]

\[ \approx \frac{\tilde{w}_{i+1} - w_{i+1}}{h} \approx Kh^n \]

for some \( K > 0 \) independent of \( h \), since \( \tau_{i+1}(h) \approx O(h^n) \).
Error control

Suppose that we can scale $h$ by $q > 0$, such that

$$|\tau_{i+1}(qh)| \approx K(qh)^n = q^nK^h \approx q^n \frac{|\tilde{w}_{i+1} - w_{i+1}|}{h} \leq \epsilon$$

So we need $q$ to satisfy

$$q \leq \left(\frac{\epsilon h}{|\tilde{w}_{i+1} - w_{i+1}|}\right)^{1/n}$$

- $q < 1$: reject the initial $h$ and recalculate using $qh$.
- $q \geq 1$: accept computed value and use $qh$ for next step.
The Runge-Kutta-Fehlberg (RKF) method uses specific 4th-order and 5th-order RK schemes, which share some computed values and together only need 6 evaluation of \( f \), to estimate

\[
q = \left( \frac{\epsilon h}{2 |\tilde{w}_{i+1} - w_{i+1}|} \right)^{1/4} = 0.84 \left( \frac{\epsilon h}{|\tilde{w}_{i+1} - w_{i+1}|} \right)^{1/4}
\]

This \( q \) is used to tune step size so that error is always bounded by the prescribed \( \epsilon \).
**Multistep method**

**Definition**

Let \( m > 1 \) be an integer, then an \( m \)-step **multistep method** is given by the form of

\[
w_{i+1} = a_{m-1}w_i + a_{m-2}w_{i-1} + \cdots + a_0w_{i-m+1} \\
+ h \left[ b_m f(t_{i+1}, w_{i+1}) + b_{m-1} f(t_i, w_i) + \cdots + b_0 f(t_{i-m+1}, w_{i-m+1}) \right]
\]

for \( i = m - 1, m, \ldots, N - 1 \).

Here \( a_0, \ldots, a_{m-1}, b_0, \ldots, b_m \) are constants. Also \( w_0 = \alpha, w_1 = \alpha_1, \ldots, w_{m-1} = \alpha_{m-1} \) need to be given.

- \( b_m = 0 \): **Explicit** \( m \)-step method.
- \( b_m \neq 0 \): **Implicit** \( m \)-step method.
Multistep method

Definition

The local truncation error of the $m$-step multistep method above is defined by

$$
\tau_{i+1}(h) = \frac{y_{i+1} - (a_{m-1}y_i + \cdots + a_0y_{i-m+1})}{h}
$$

$$
- \left[ b_m f(t_{i+1}, y_{i+1}) + b_{m-1} f(t_i, y_i) + \cdots + b_0 f(t_{i-m+1}, y_{i-m+1}) \right]
$$

where $y_i := y(t_i)$. 
Adams-Bashforth Explicit method

Adams-Bashforth Two-Step Explicit method:

\[
\begin{aligned}
&\begin{cases}
  w_0 = \alpha, & w_1 = \alpha_1, \\
  w_{i+1} = w_i + \frac{h}{2} \left[ 3f(t_i, w_i) - f(t_{i-1}, w_{i-1}) \right]
\end{cases}
\end{aligned}
\]

for \( i = 1, \ldots, N - 1 \).

The local truncation error is

\[
\tau_{i+1}(h) = \frac{5}{12} y'''(\mu_i) h^2
\]

for some \( \mu_i \in (t_{i-1}, t_{i+1}) \).
We denote $y_i^{(k)} := y^{(k)}(t_i)$ for short. If $w_j = y_j$ for $j \leq i$, then

$$y_{i+1} = y_i + hy_i' + \frac{h^2}{2} y_i'' + \frac{h^3}{6} y'''(\xi_i), \quad \xi_i \in (t_i, t_{i+1})$$

$$w_{i+1} = y_i + hy_i' + \frac{h}{2} (y_i' - y_{i-1}'),$$

$$y_{i-1}' = y_i' - hy_i'' + \frac{h^2}{2} y'''(\eta_i), \quad \eta_i \in (t_{i-1}, t_i)$$

Plugging the equations above into the formula of local truncation error:

$$\tau_{i+1}(h) = \frac{y_{i+1} - w_{i+1}}{h} = \left( \frac{1}{6} y'''(\xi_i) + \frac{1}{4} y'''(\eta_i) \right) h^2 = \frac{5}{12} y'''(\mu_i) h^2$$

for some $\mu_i \in (t_{i-1}, t_{i+1})$, where in the last equality we used the intermediate value theorem and $y \in C^3$ (so $y'''$ is continuous) to obtain $y'''(\mu_i) = \frac{1}{6} y'''(\xi_i) + \frac{1}{4} y'''(\eta_i)$, which is between $y'''(\xi_i)$ and $y'''(\eta_i)$. 
Adams-Bashforth Explicit method

Adams-Bashforth Three-Step Explicit method:

\[
\begin{align*}
    w_0 &= \alpha, \quad w_1 = \alpha_1, \quad w_2 = \alpha_2, \\
    w_{i+1} &= w_i + \frac{h}{12} \left[ 23f(t_i, w_i) - 16f(t_{i-1}, w_{i-1}) + 5f(t_{i-2}, w_{i-2}) \right]
\end{align*}
\]

for \( i = 2, \ldots, N - 1 \).

The local truncation error is

\[
\tau_{i+1}(h) = \frac{3}{8} y^{(4)}(\mu_i) h^3
\]

for some \( \mu_i \in (t_{i-2}, t_{i+1}) \).
Adams-Bashforth Explicit method

Adams-Bashforth Four-Step Explicit method:

\[
\begin{aligned}
&w_0 = \alpha, \quad w_1 = \alpha_1, \quad w_2 = \alpha_2, \quad w_3 = \alpha_3 \\
&w_{i+1} = w_i + \frac{h}{24} \left[ 55f(t_i, w_i) - 59f(t_{i-1}, w_{i-1}) + 37f(t_{i-2}, w_{i-2}) - 9f(t_{i-3}, w_{i-3}) \right]
\end{aligned}
\]

for \( i = 3, \ldots, N - 1 \).

The local truncation error is

\[
\tau_{i+1}(h) = \frac{251}{720} y^{(5)}(\mu_i) h^4
\]

for some \( \mu_i \in (t_{i-3}, t_{i+1}) \).
Adams-Bashforth Explicit method

Adams-Bashforth Five-Step Explicit method:

\[
\begin{cases}
  w_0 = \alpha, & w_1 = \alpha_1, & w_2 = \alpha_2, & w_3 = \alpha_3, & w_4 = \alpha_4 \\
  w_{i+1} = w_i + \frac{h}{720} \left[ 1901f(t_i, w_i) - 2774f(t_{i-1}, w_{i-1}) + 2616f(t_{i-2}, w_{i-2}) \\
  - 1274f(t_{i-3}, w_{i-3}) + 251f(t_{i-4}, w_{i-4}) \right]
\end{cases}
\]

for \( i = 4, \ldots, N - 1. \)

The local truncation error is

\[
\tau_{i+1}(h) = \frac{95}{288} y^{(6)}(\mu_i) h^5
\]

for some \( \mu_i \in (t_{i-4}, t_{i+1}). \)
Adams-Moulton Implicit method

Adams-Moulton Two-Step Implicit method:

\[
\begin{aligned}
&\begin{cases}
  w_0 = \alpha, & w_1 = \alpha_1, \\
  w_{i+1} = w_i + \frac{h}{12}[5f(t_{i+1}, w_{i+1}) + 8f(t_i, w_i) - f(t_{i-1}, w_{i-1})]
\end{cases}
\end{aligned}
\]

for \( i = 1, \ldots, N - 1 \).

The local truncation error is

\[
\tau_{i+1}(h) = -\frac{1}{24}y^{(4)}(\mu_i)h^3
\]

for some \( \mu_i \in (t_{i-1}, t_{i+1}) \).
Adams-Moulton Implicit method

Adams-Moulton Three-Step Implicit method:

\[
\begin{align*}
    w_0 &= \alpha, \quad w_1 = \alpha_1, \quad w_2 = \alpha_2 \\& \\
    w_{i+1} &= w_i + \frac{h}{24} [9f(t_{i+1}, w_{i+1}) + 19f(t_i, w_i) - 5f(t_{i-1}, w_{i-1}) + f(t_{i-2}, w_{i-2})]
\end{align*}
\]

for \( i = 2, \ldots, N - 1 \).

The local truncation error is

\[
\tau_{i+1}(h) = -\frac{19}{720} y^{(5)}(\mu_i) h^4
\]

for some \( \mu_i \in (t_{i-2}, t_{i+1}) \).
Adams-Moulton Implicit method

Adams-Moulton Four-Step Implicit method:

\[
\begin{align*}
\{ & w_0 = \alpha, \quad w_1 = \alpha_1, \quad w_2 = \alpha_2, \quad w_3 = \alpha_3 \\
& w_{i+1} = w_i + \frac{h}{720} \left[ 251 f(t_{i+1}, w_{i+1}) + 646 f(t_i, w_i) - 264 f(t_{i-1}, w_{i-1}) \\
& \quad + 106 f(t_{i-2}, w_{i-2}) - 19 f(t_{i-3}, w_{i-3}) \right]
\end{align*}
\]

for \( i = 3, \ldots, N - 1. \)

The local truncation error is

\[
\tau_{i+1}(h) = -\frac{3}{160} y^{(6)}(\mu_i) h^5
\]

for some \( \mu_i \in (t_{i-3}, t_{i+1}). \)
Steps to develop multistep methods

- Construct interpolating polynomial $P(t)$ (e.g., Newton’s backward difference method) using previously computed $(t_{i-m+1}, w_{i-m+1}), \ldots, (t_i, w_i)$.
- Approximate $y(t_{i+1})$ based on

$$y(t_{i+1}) = y(t_i) + \int_{t_i}^{t_{i+1}} y'(t) \, dt = y(t_i) + \int_{t_i}^{t_{i+1}} f(t, y(t)) \, dt \approx y(t_i) + \int_{t_i}^{t_{i+1}} f(t, P(t)) \, dt$$

and construct difference method:

$$w_{i+1} = w_i + h\phi(t_i, \ldots, t_{i-m+1}, w_i, \ldots, w_{i-m+1})$$
Explicit vs. Implicit

- Implicit methods are generally more accurate than the explicit ones (e.g., Adams-Moulton three-step implicit method is even more accurate than Adams-Bashforth four-step explicit method).

- Implicit methods require solving for $w_{i+1}$ from

\[ w_{i+1} = \cdots + \frac{h}{XXX} f(t_{i+1}, w_{i+1}) + \cdots \]

which can be difficult or even impossible.

- There could be multiple solutions of $w_{i+1}$ when solving the equation above in implicit methods.
Predictor-Corrector method

Due to the aforementioned issues, implicit methods are often cast in “predictor-corrector” form in practice.

In each step $i$:

- **Prediction:** Compute $w_{i+1}$ using an explicit method $\phi$ to get $w_{i+1,p}$ using
  \[
  w_{i+1,p} = w_i + h\phi(t_i, w_i, \ldots, t_{i-m+1}, w_{i-m+1})
  \]

- **Correction:** Substitute $w_{i+1}$ by $w_{i+1,p}$ in the implicit method $\tilde{\phi}$ and compute $w_{i+1}$ using
  \[
  w_{i+1} = w_i + h\tilde{\phi}(t_{i+1}, w_{i+1,p}, t_i, w_i, \ldots, t_{i-m+1}, w_{i-m+1})
  \]
Predictor-Corrector method

Example

Use the Adams-Bashforth 4-step explicit method and Adams-Moulton 3-step implicit method to form the Adams 4th-order Predictor-Corrector method.

With initial value \( w_0 = \alpha \), suppose we first generate \( w_1, w_2, w_3 \) using RK4 method. Then for \( i = 3, 4, \ldots, N - 1 \):

- Use Adams-Bashforth 4-step explicit method to get a predictor \( w_{i+1,p} \):

\[
w_{i+1,p} = w_i + \frac{h}{24} \left[ 55f(t_i, w_i) - 59f(t_{i-1}, w_{i-1}) + 37f(t_{i-2}, w_{i-2}) - 9f(t_{i-3}, w_{i-3}) \right]
\]

- Use Adams-Moulton 3-step implicit method to get a corrector \( w_{i+1} \):

\[
w_{i+1} = w_i + \frac{h}{24} \left[ 9f(t_{i+1}, w_{i+1,p}) + 19f(t_i, w_i) - 5f(t_{i-1}, w_{i-1}) + f(t_{i-2}, w_{i-2}) \right]
\]
Predictor-Corrector method

Example

*Use Adams Predictor-Corrector Method with $h = 0.2$ to solve IVP* 

$y' = y - t^2 + 1$ for $t \in [0, 2]$ and $y(0) = 0.5$.

<table>
<thead>
<tr>
<th>$t_i$</th>
<th>$y_i = y(t_i)$</th>
<th>$w_i$</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.50000000</td>
<td>0.50000000</td>
<td>0</td>
</tr>
<tr>
<td>0.2</td>
<td>0.8292986</td>
<td>0.8292933</td>
<td>0.0000053</td>
</tr>
<tr>
<td>0.4</td>
<td>1.2140877</td>
<td>1.2140762</td>
<td>0.0000114</td>
</tr>
<tr>
<td>0.6</td>
<td>1.6489406</td>
<td>1.6489220</td>
<td>0.0000186</td>
</tr>
<tr>
<td>0.8</td>
<td>2.1272295</td>
<td>2.1272056</td>
<td>0.0000239</td>
</tr>
<tr>
<td>1.0</td>
<td>2.6408591</td>
<td>2.6408286</td>
<td>0.0000305</td>
</tr>
<tr>
<td>1.2</td>
<td>3.1799415</td>
<td>3.1799026</td>
<td>0.0000389</td>
</tr>
<tr>
<td>1.4</td>
<td>3.7324000</td>
<td>3.7323505</td>
<td>0.0000495</td>
</tr>
<tr>
<td>1.6</td>
<td>4.2834838</td>
<td>4.2834208</td>
<td>0.0000630</td>
</tr>
<tr>
<td>1.8</td>
<td>4.8151763</td>
<td>4.8150964</td>
<td>0.0000799</td>
</tr>
<tr>
<td>2.0</td>
<td>5.3054720</td>
<td>5.3053707</td>
<td>0.0001013</td>
</tr>
</tbody>
</table>
We can also use Milne’s 3-step explicit method and Simpson’s 2-step implicit method below:

\[
\begin{align*}
  w_{i+1,p} &= w_{i-3} + \frac{4h}{3} \left[ 2f(t_i, w_i) - f(t_{i-1}, w_{i-1}) + 2f(t_{i-2}, w_{i-2}) \right] \\
  w_{i+1} &= w_{i-1} + \frac{h}{3} \left[ f(t_{i+1}, w_{i+1,p}) + 4f(t_i, w_i) + f(t_{i-1}, w_{i-1}) \right]
\end{align*}
\]

This method is \(O(h^4)\) and generally has better accuracy than Adams PC method. However it is more likely to be vulnerable to round-off error.
Predictor-Corrector method

- PC methods have comparable accuracy as RK4, but often require only 2 evaluations of $f$ in each step.
- Need to store values of $f$ for several previous steps.
- Sometimes are more restrictive on step size $h$, e.g., in the stiff differential equation case later.
Variable step-size multistep method

Now let’s take a closer look at the errors of the multistep methods. Denote $y_i := y(t_i)$.

The Adams-Bashforth 4-step explicit method has error

$$\tau_{i+1}(h) = \frac{251}{720} y^{(5)}(\mu_i) h^4$$

The Adams-Moulton 3-step implicit method has error

$$\tilde{\tau}_{i+1}(h) = -\frac{19}{720} y^{(5)}(\tilde{\mu}_i) h^4$$

where $\mu_i \in (t_{i-3}, t_{i+1})$ and $\tilde{\mu}_i \in (t_{i-2}, t_{i+1})$.

Question: Can we find a way to scale step size $h$ so the error is under control?
Consider the their local truncation errors:

\[ y_{i+1} - w_{i+1,p} = \frac{251}{720} y^{(5)}(\mu_i) h^5 \]

\[ y_{i+1} - w_{i+1} = -\frac{19}{720} y^{(5)}(\tilde{\mu}_i) h^5 \]

Assume \( y^{(5)}(\mu_i) \approx y^{(5)}(\tilde{\mu}_i) \), we take their difference to get

\[ w_{i+1} - w_{i+1,p} = \frac{1}{720} (19 + 251) y^{(5)}(\mu_i) h^5 \approx \frac{3}{8} y^{(5)}(\mu_i) h^5 \]

So the error of Adams-Moulton (corrector step) is

\[ \tilde{\tau}_{i+1}(h) = \frac{|y_{i+1} - w_{i+1}|}{h} \approx \frac{19 |w_{i+1} - w_{i+1,p}|}{270h} = Kh^4 \]

where \( K \) is independent of \( h \) since \( \tilde{\tau}_{i+1}(h) = O(h^4) \).
Variable step-size multistep method

If we want to keep error under a prescribed $\epsilon$, then we need to find $q > 0$ such that with step size $qh$, there is

$$\tilde{\tau}_{i+1}(qh) = \frac{|y(t_i + qh) - w_{i+1}|}{qh} \approx \frac{19q^4|w_{i+1} - w_{i+1,p}|}{270h} < \epsilon$$

This implies that

$$q < \left( \frac{270h\epsilon}{19|w_{i+1} - w_{i+1,p}|} \right)^{1/4} \approx 2 \left( \frac{h\epsilon}{|w_{i+1} - w_{i+1,p}|} \right)^{1/4}$$

To be conservative, we may replace 2 by 1.5 above.

In practice, we tune $q$ (as less as possible) such that the estimated error is between $(\epsilon/10, \epsilon)$.
System of differential equations

The IVP for a system of ODE has form

\[
\begin{align*}
\frac{du_1}{dt} &= f_1(t, u_1, u_2, \ldots, u_m) \\
\frac{du_2}{dt} &= f_2(t, u_1, u_2, \ldots, u_m) \\
&\vdots \\
\frac{du_m}{dt} &= f_m(t, u_1, u_2, \ldots, u_m)
\end{align*}
\]

for \( a \leq t \leq b \)

with initial value \( u_1(a) = \alpha_1, \ldots, u_m(a) = \alpha_m \).

Definition

A set of functions \( u_1(t), \ldots, u_m(t) \) is a solution of the IVP above if they satisfy both the system of ODEs and the initial values.
System of differential equations

In this case, we will solve for $u_1(t), \ldots, u_m(t)$ which are interdependent according to the ODE system.
System of differential equations

Definition

A function $f$ is called **Lipschitz** with respect to $u_1, \ldots, u_m$ on $D := [a, b] \times \mathbb{R}^m$ if there exists $L > 0$ s.t.

$$|f(t, u_1, \ldots, u_m) - f(t, z_1, \ldots, z_m)| \leq L \sum_{j=1}^{m} |u_j - z_j|$$

for all $(t, u_1, \ldots, u_m), (t, z_1, \ldots, z_m) \in D.$
Theorem
If \( f \in C^1(D) \) and \( \frac{\partial f}{\partial u_j} \leq L \) for all \( j \), then \( f \) is Lipschitz with respect to \( u = (u_1, \ldots, u_m) \) on \( D \).

Proof.
Note that \( D \) is convex. For any \((t, u_1, \ldots, u_m), (t, z_1, \ldots, z_m) \in D\), define

\[
g(\lambda) = f(t, (1 - \lambda)u_1 + \lambda z_1, \ldots, (1 - \lambda)u_m + \lambda z_m)
\]

for all \( \lambda \in [0, 1] \). Then from \( |g(1) - g(0)| \leq \int_0^1 |g'(\lambda)| \, d\lambda \) and the definition of \( g \), the conclusion follows. \( \square \)
System of differential equations

Theorem

If \( f \in C^1(D) \) and is Lipschitz with respect to \( u = (u_1, \ldots, u_m) \), then the IVP with \( f \) as defining function has a unique solution.
System of differential equations

Now let’s use vector notations below

\[ a = (\alpha_1, \ldots, \alpha_m) \]
\[ y = (y_1, \ldots, y_m) \]
\[ w = (w_1, \ldots, w_m) \]
\[ f(t, w) = (f_1(t, w), \ldots, f_m(t, w)) \]

Then the IVP of ODE system can be written as

\[ y' = f(t, y), \quad t \in [a, b] \]

with initial value \( y(a) = a \). So the difference methods developed above, such as RK4, still apply.
System of differential equations

Example

Use RK4 (with $h = 0.1$) to solve IVP for ODE system

\[
\begin{align*}
    l_1'(t) &= f_1(t, l_1, l_2) = -4l_1 + 3l_2 + 6 \\
    l_2'(t) &= f_2(t, l_1, l_2) = -2.4l_1 + 1.6l_2 + 3.6
\end{align*}
\]

with initial value $l_1(0) = l_2(0) = 0$.

Solution: The exact solution is

\[
\begin{align*}
    l_1(t) &= -3.375e^{-2t} + 1.875e^{-0.4t} + 1.5 \\
    l_2(t) &= -2.25e^{-2t} + 2.25e^{-0.4t}
\end{align*}
\]

for all $t \geq 0$. 
System of differential equations

Example

*Use RK4 (with \( h = 0.1 \)) to solve IVP for ODE system*

\[
\begin{align*}
I'_1(t) &= f_1(t, I_1, I_2) = -4I_1 + 3I_2 + 6 \\
I'_2(t) &= f_2(t, I_1, I_2) = -2.4I_1 + 1.6I_2 + 3.6
\end{align*}
\]

with initial value \( I_1(0) = I_2(0) = 0 \).

**Solution:** (cont) The result by RK4 is

| \( t_j \) | \( w_{1,j} \) | \( w_{2,j} \) | \( |I_1(t_j) - w_{1,j}| \) | \( |I_2(t_j) - w_{2,j}| \) |
|-----------|-----------|-----------|-----------------|-----------------|
| 0.0       | 0         | 0         | 0               | 0               |
| 0.1       | 0.5382550 | 0.3196263 | \( 0.8285 \times 10^{-5} \) | \( 0.5803 \times 10^{-5} \) |
| 0.2       | 0.9684983 | 0.5687817 | \( 0.1514 \times 10^{-4} \) | \( 0.9596 \times 10^{-5} \) |
| 0.3       | 1.310717  | 0.7607328 | \( 0.1907 \times 10^{-4} \) | \( 0.1216 \times 10^{-4} \) |
| 0.4       | 1.581263  | 0.9063208 | \( 0.2098 \times 10^{-4} \) | \( 0.1311 \times 10^{-4} \) |
| 0.5       | 1.793505  | 1.014402  | \( 0.2193 \times 10^{-4} \) | \( 0.1240 \times 10^{-4} \) |
A general IVP for $m$th-order ODE is

$$y^{(m)} = f(t, y, y', \ldots, y^{(m-1)}), \quad t \in [a, b]$$

with initial value $y(a) = \alpha_1, y'(a) = \alpha_2, \ldots, y^{(m-1)}(a) = \alpha_m$.

**Definition**

A function $y(t)$ is a solution of IVP for the $m$th-order ODE above if $y(t)$ satisfies the differential equation for $t \in [a, b]$ and all initial value conditions at $t = a$. 
High-order ordinary differential equations

We can define a set of functions $u_1, \ldots, u_m$ s.t.

$$u_1(t) = y(t), \quad u_2(t) = y'(t), \quad \ldots, \quad u_m(t) = y^{(m-1)}(t)$$

Then we can convert the $m$th-order ODE to a system of first-order ODEs (total of $m$ coupled ODEs):

$$\begin{cases} 
    u'_1 = u_2 \\
    u'_2 = u_3 \\
    \vdots \\
    u'_m = f(t, u_1, u_2, \ldots, u_m)
\end{cases} \quad \text{for } a \leq t \leq b$$

with initial values $u_1(a) = \alpha_1, \ldots, u_m(a) = \alpha_m$. 
High-order ordinary differential equations

Example

Use RK4 (with $h = 0.1$) to solve IVP for ODE system

$$y'' - 2y' + 2y = e^{2t} \sin t, \quad t \in [0, 1]$$

with initial value $y(0) = -0.4$, $y'(0) = -0.6$.

Solution: The exact solution is

$$y(t) = u_1(t) = 0.2e^{2t}(\sin t - 2 \cos t).$$

Also $u_2(t) = y'(t) = u'_1(t)$ but we don’t need it.
High-order ordinary differential equations

Example

Use RK4 (with \( h = 0.1 \)) to solve IVP for ODE system

\[
y'' - 2y' + 2y = e^{2t} \sin t, \quad t \in [0, 1]
\]

with initial value \( y(0) = -0.4, y'(0) = -0.6 \).

Solution: (cont) The result by RK4 is

| \( t_j \) | \( y(t_j) = u_1(t_j) \) | \( w_{1,j} \) | \( y'(t_j) = u_2(t_j) \) | \( w_{2,j} \) | \( |y(t_j) - w_{1,j}| \) | \( |y'(t_j) - w_{2,j}| \) |
|---|---|---|---|---|---|---|
| 0.0 | -0.40000000 | -0.40000000 | -0.60000000 | -0.60000000 | 0 | 0 |
| 0.1 | -0.46173297 | -0.46173334 | -0.63163040 | -0.63163124 | 3.7 \times 10^{-7} | 7.75 \times 10^{-7} |
| 0.2 | -0.52555905 | -0.52555988 | -0.64014789 | -0.64014895 | 8.3 \times 10^{-7} | 1.01 \times 10^{-6} |
| 0.3 | -0.58860005 | -0.58860144 | -0.61366300 | -0.61366381 | 1.39 \times 10^{-6} | 8.34 \times 10^{-7} |
| 0.4 | -0.64661028 | -0.64661231 | -0.53658210 | -0.53658203 | 2.03 \times 10^{-6} | 1.79 \times 10^{-7} |
| 0.5 | -0.69356395 | -0.69356666 | -0.38873950 | -0.38873810 | 2.71 \times 10^{-6} | 5.96 \times 10^{-7} |
| 0.6 | -0.72114849 | -0.72115190 | -0.14438340 | -0.14438087 | 3.41 \times 10^{-6} | 7.75 \times 10^{-7} |
| 0.7 | -0.71814890 | -0.71815295 | 0.22899170 | 0.22899702 | 4.05 \times 10^{-6} | 2.03 \times 10^{-6} |
| 0.8 | -0.66970677 | -0.66971133 | 0.77198150 | 0.77199180 | 4.56 \times 10^{-6} | 5.30 \times 10^{-6} |
| 0.9 | -0.55643814 | -0.55644290 | 1.53476405 | 1.53478150 | 4.76 \times 10^{-6} | 9.54 \times 10^{-6} |
| 1.0 | -0.35339436 | -0.35339886 | 2.57874105 | 2.57876630 | 4.50 \times 10^{-6} | 1.34 \times 10^{-5} |
A brief summary

The difference methods we developed above, e.g., Euler’s, midpoints, RK4, multistep explicit/implicit, predictor-corrector methods, are

- based on step-by-step derivation and easy to understand;
- widely used in many practical problems;
- fundamental to more advanced and complex techniques.
Stability of difference methods

Definition (Consistency)

A difference method is called **consistent** if

$$\lim_{h \to 0} \left( \max_{1 \leq i \leq N} \tau_i(h) \right) = 0$$

where $\tau_i(h)$ is the local truncation error of the method.

Remark

Since local truncation error $\tau_i(h)$ is defined assuming previous $w_i = y_i$, it does not take error accumulation into account. So the consistency definition above only considers how good $\phi(t, w_i, h)$ in the difference method is.
Stability of difference methods

For any step size $h > 0$, the difference method $w_{i+1} = w_i + h\phi(t_i, w_i, h)$ can generate a sequence of $w_i$ which depend on $h$. We call them $\{w_i(h)\}_i$. Note that $w_i$ gradually accumulate errors as $i = 1, 2, \ldots, N$.

Definition (Convergent)

A difference method is called **convergent** if

$$\lim_{h \to 0} \left( \max_{1 \leq i \leq N} |y_i - w_i(h)| \right) = 0$$
Stability of difference methods

Example

Show that Euler’s method is convergent.

Solution: We have showed before that for fixed \( h > 0 \) there is

\[
|y(t_i) - w_i| \leq \frac{hM}{2L} \left( e^{L(t_i-a)} - 1 \right) \leq \frac{hM}{2L} \left( e^{L(b-a)} - 1 \right)
\]

for all \( i = 0, \ldots, N \). Therefore we have

\[
\max_{1 \leq i \leq N} |y(t_i) - w_i| \leq \frac{hM}{2L} \left( e^{L(b-a)} - 1 \right) \rightarrow 0
\]

as \( h \rightarrow 0 \). Therefore \( \lim_{h \rightarrow 0} (\max_{1 \leq i \leq N} |y(t_i) - w_i|) = 0 \).
Definition

A numerical method is called **stable** if its results depend on the initial data continuously.
Stability of difference methods

Theorem
For a given IVP \( y' = f(t, y) \), \( t \in [a, b] \) with \( y(a) = \alpha \), consider a difference method \( w_{i+1} = w_i + h\phi(t_i, w_i, h) \) with \( w_0 = \alpha \). If there exists \( h_0 > 0 \) such that \( \phi \) is continuous on \([a, b] \times \mathbb{R} \times [0, h_0]\), and \( \phi \) is \( L \)-Lipschitz with respect to \( w \), then

- The difference method is stable.
- The difference method is convergent if and only if it is consistent (i.e., \( \phi(t, y, 0) = f(t, y) \)).
- If there exists bound \( \tau(h) \) such that \( |\tau_i(h)| \leq \tau(h) \) for all \( i = 1, \ldots, N \), then \( |y(t_i) - w_i| \leq \tau(h)e^{L(t_i-a)/L} \).
Proof.
Let $h$ be fixed, then $w_i(\alpha)$ generated by the difference method are functions of $\alpha$. For any two values $\alpha$, $\hat{\alpha}$, there is

$$|w_{i+1}(\alpha) - w_{i+1}(\hat{\alpha})| = |(w_i(\alpha) - h\phi(t_i, w_i(\alpha), h)) - (w_i(\hat{\alpha}) - h\phi(t_i, w_i(\hat{\alpha}), h))|$$

$$\leq |w_i(\alpha) - w_i(\hat{\alpha})| + h|\phi(t_i, w_i(\alpha), h) - \phi(t_i, w_i(\hat{\alpha}), h)|$$

$$\leq |w_i(\alpha) - w_i(\hat{\alpha})| + hL|w_i(\alpha) - w_i(\hat{\alpha})|$$

$$= (1 + hL)|w_i(\alpha) - w_i(\hat{\alpha})|$$

$$\leq \cdots$$

$$\leq (1 + hL)^{i+1}|w_0(\alpha) - w_0(\hat{\alpha})|$$

$$= (1 + hL)^{i+1}|\alpha - \hat{\alpha}|$$

$$\leq (1 + hL)^N|\alpha - \hat{\alpha}|$$

Therefore $w_i(\alpha)$ is Lipschitz with respect to $\alpha$ (with constant at most $(1 + hL)^N$), and hence is continuous with respect to $\alpha$. We omit the proofs for the other two assertions here. \qed
Stability of difference method

Example

Use the result of Theorem above to show that the Modified Euler’s method is stable.

Solution: Recall the Modified Euler’s method is given by

\[
    w_{i+1} = w_i + \frac{h}{2} \left( f(t_i, w_i) + f(t_{i+1}, w_i + hf(t_i, w_i)) \right)
\]

So we have \( \phi(t, w, h) = \frac{1}{2} (f(t, w) + f(t + h, w + hf(t, w))) \).

Now we want to show \( \phi \) is continuous in \( (t, w, h) \), and Lipschitz with respect to \( w \).
Solution: (cont) It is obvious that $\phi$ is continuous in $(t, w, h)$ since $f(t, w)$ is continuous. Fix $t$ and $h$. For any $w, \bar{w} \in \mathbb{R}$, there is

$$|\phi(t, w, h) - \phi(t, \bar{w}, h)| \leq \frac{1}{2}|f(t, w) - f(t, \bar{w})|$$

$$+ \frac{1}{2}|f(t + h, w + hf(t, w)) - f(t + h, \bar{w} + hf(t, \bar{w}))|$$

$$\leq \frac{L}{2}|w - \bar{w}| + \frac{L}{2}|(w + hf(t, w)) - (\bar{w} + hf(t, \bar{w}))|$$

$$\leq L|w - \bar{w}| + \frac{Lh}{2}|f(t, w) - f(t, \bar{w})|$$

$$\leq L|w - \bar{w}| + \frac{L^2 h}{2}|w - \bar{w}|$$

$$= (L + \frac{L^2 h}{2})|w - \bar{w}|$$

So $\phi$ is Lipschitz with respect to $w$. By first part of Theorem above, the Modified Euler’s method is stable.
Stability of multistep difference method

Definition
Suppose a multistep difference method given by

$$w_{i+1} = a_{m-1}w_i + a_{m-2}w_{i-1} + \cdots + a_0w_{i-m+1} + hF(t_i, h, w_{i+1}, \ldots, w_{i-m+1})$$

Then we call the following the characteristic polynomial of the method:

$$\lambda^m - (a_{m-1}\lambda^{m-1} + \cdots + a_1\lambda + a_0)$$

Definition
A difference method is said to satisfy the root condition if all the $m$ roots $\lambda_1, \ldots, \lambda_m$ of its characteristic polynomial have magnitudes $\leq 1$, and all of those which have magnitude $= 1$ are single roots.
Stability of multistep difference method

Definition

- A difference method that satisfies root condition is called **strongly stable** if the only root with magnitude 1 is $\lambda = 1$.
- A difference method that satisfies root condition is called **weakly stable** if there are multiple roots with magnitude 1.
- A difference method that does not satisfy root condition is called **unstable**.
Stability of multistep difference method

Theorem

- A difference method is stable if and only if it satisfies the root condition.

- If a difference method is consistent, then it is stable if and only if it is convergent.
Example

Show that the Adams-Bashforth 4-step explicit method is strongly stable.

Solution: Recall that the method is given by

\[ w_{i+1} = w_i + \frac{h}{24} \left[ 55f(t_i, w_i) - 59f(t_{i-1}, w_{i-1}) + 37f(t_{i-2}, w_{i-2}) - 9f(t_{i-3}, w_{i-3}) \right] \]

So the characteristic polynomial is simply \( \lambda^4 - \lambda^3 = \lambda^3(\lambda - 1) \), which only has one root \( \lambda = 1 \) with magnitude 1. So the method is strongly stable.
Stability of multistep difference method

Example

Show that the Milne’s 4-step explicit method is weakly stable but not strongly stable.

Solution: Recall that the method is given by

\[ w_{i+1} = w_{i-3} + \frac{4h}{3} \left[ 2f(t_i, w_i) - f(t_{i-1}, w_{i-1}) + 2f(t_{i-2}, w_{i-2}) \right] \]

So the characteristic polynomial is simply \( \lambda^4 - 1 \), which have roots \( \lambda = \pm 1, \pm i \). So the method is weakly stable but not strongly stable.

Remark

This is the reason we chose Adams-Bashforth-Moulton PC rather than Milne-Simpsons PC since the former is strongly stable and likely to be more robust.
Stiff differential equations have $e^{-ct}$ terms ($c > 0$ large) in their solutions. These terms $\to 0$ quickly, but their derivatives (of form $c^ne^{-ct}$) do not, especially at small $t$.

Recall that difference methods have errors proportional to the derivatives, and hence they may be inaccurate for stiff ODEs.
Stiff differential equations

Example

Use RK4 to solve the IVP for a system of two ODEs:

\[
\begin{align*}
    u'_1 &= 9u_1 + 24u_2 + 5\cos t - \frac{1}{3}\sin t \\
    u'_2 &= -24u_1 - 51u_2 - 9\cos t + \frac{1}{3}\sin t
\end{align*}
\]

with initial values \( u_1(0) = 4/3 \) and \( u_2(0) = 2/3 \).

Solution: The exact solution is

\[
\begin{align*}
    u_1(t) &= 2e^{-3t} - e^{-39t} + \frac{1}{3}\cos t \\
    u_2(t) &= -e^{-3t} + 2e^{-39t} - \frac{1}{3}\cos t
\end{align*}
\]

for all \( t \geq 0 \).
Stiff differential equations

Solution: (cont) When we apply RK4 to this stiff ODE, we obtain

<table>
<thead>
<tr>
<th>$t$</th>
<th>$u_1(t)$</th>
<th>$w_1(t)$</th>
<th>$u_1(t)$</th>
<th>$w_1(t)$</th>
<th>$u_2(t)$</th>
<th>$w_2(t)$</th>
<th>$w_2(t)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$h = 0.05$</td>
<td>$h = 0.1$</td>
<td></td>
<td></td>
<td>$h = 0.05$</td>
<td>$h = 0.1$</td>
</tr>
<tr>
<td>0.1</td>
<td>1.793061</td>
<td>1.712219</td>
<td>–2.645169</td>
<td>–1.032001</td>
<td>–0.8703152</td>
<td>7.844527</td>
<td></td>
</tr>
<tr>
<td>0.2</td>
<td>1.423901</td>
<td>1.414070</td>
<td>–18.45158</td>
<td>–0.8746809</td>
<td>–0.8550148</td>
<td>38.87631</td>
<td></td>
</tr>
<tr>
<td>0.3</td>
<td>1.131575</td>
<td>1.130523</td>
<td>–87.47221</td>
<td>–0.7249984</td>
<td>–0.7228910</td>
<td>176.4828</td>
<td></td>
</tr>
<tr>
<td>0.4</td>
<td>0.9094086</td>
<td>0.9092763</td>
<td>–934.0722</td>
<td>–0.6082141</td>
<td>–0.6079475</td>
<td>789.3540</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>0.7387877</td>
<td>9.7387506</td>
<td>–1760.016</td>
<td>–0.5156575</td>
<td>–0.5155810</td>
<td>3520.00</td>
<td></td>
</tr>
<tr>
<td>0.6</td>
<td>0.6057094</td>
<td>0.6056833</td>
<td>–7848.550</td>
<td>–0.4404108</td>
<td>–0.4403558</td>
<td>15697.84</td>
<td></td>
</tr>
<tr>
<td>0.7</td>
<td>0.4998603</td>
<td>0.4998361</td>
<td>–34989.63</td>
<td>–0.3774038</td>
<td>–0.3773540</td>
<td>69979.87</td>
<td></td>
</tr>
<tr>
<td>0.8</td>
<td>0.4136714</td>
<td>0.4136490</td>
<td>–155979.4</td>
<td>–0.3229535</td>
<td>–0.3229078</td>
<td>311959.5</td>
<td></td>
</tr>
<tr>
<td>0.9</td>
<td>0.3416143</td>
<td>0.3415939</td>
<td>–695332.0</td>
<td>–0.2744088</td>
<td>–0.2743673</td>
<td>1390664.</td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>0.2796748</td>
<td>0.2796568</td>
<td>–3099671.</td>
<td>–0.2298877</td>
<td>–0.2298511</td>
<td>6199352.</td>
<td></td>
</tr>
</tbody>
</table>

which can blow up for larger step size $h$. 
Now let’s use a simple example to see why this happens: consider an IVP \( y' = \lambda y, \ t \geq 0, \) and \( y(0) = \alpha. \) Here \( \lambda < 0. \) We know the problem has solution \( y(t) = \alpha e^{\lambda t}. \)

Suppose we apply Euler’s method, which is

\[
\begin{align*}
    w_{i+1} &= w_i + hf(t_i, w_i) = w_i + h\lambda w_i = (1 + \lambda h)w_i \\
    &= \cdots = (1 + \lambda h)^{i+1}w_0 = (1 + \lambda h)^{i+1}\alpha
\end{align*}
\]

Therefore we simply have \( w_i = (1 + \lambda h)^i\alpha. \) So the error is

\[
|y(t_i) - w_i| = |\alpha e^{\lambda h} - (1 + \lambda h)^i\alpha| = |e^{\lambda h} - (1 + \lambda h)^i||\alpha|
\]

In order for the error not to blow up, we need at least \(|1 + \lambda h| < 1,\) which yields \( h < \frac{2}{|\lambda|}. \) So \( h \) needs to be sufficiently small for large \( \lambda. \)
Similar issue occurs for other one-step methods, which for this IVP can be written as $w_{i+1} = Q(\lambda h)w_i = \cdots = (Q(\lambda h))^{i+1} \alpha$. For the solution not to blow up, we need $|Q(\lambda h)| < 1$.

For example, in $n$th-order Taylor’s method, we need

$$|Q(\lambda h)| = \left|1 + \lambda h + \frac{\lambda^2 h^2}{2} + \cdots + \frac{\lambda^n h^n}{n!}\right| < 1$$

which requires $h$ to be very small.

The same issue occurs for multistep methods too.
A remedy of stiff ODE is using implicit method, e.g., the implicit Trapezoid method:

\[ w_{i+1} = w_i + \frac{h}{2} \left( f(t_{i+1}, w_{i+1}) + f(t_i, w_i) \right) \]

In each step, we need to solve for \( w_{i+1} \) from the equation above. Namely, we need to solve for the root of \( F(w) \):

\[ F(w) := w - w_i - \frac{h}{2} (f(t_{i+1}, w) + f(t_i, w_i)) = 0 \]

We can use fixed point iteration or Newton’s method to solve \( F(x) = 0 \).