Temporal integrators can be classified along several lines:

- one-step (multistage)
- multi-step single stage
- multi-step multi stage

- explicit
- implicit
- mixed implicit-explicit
Runge-Kutta Schemes

\[ w'(t) = F(t, w(t)), \quad t > 0 \]

One-step RK with \( s \) stages

\[
\begin{align*}
\begin{cases}
\quad w^{n+1} = w^n + \sum_{i=1}^{s} b_i \cdot F(t_n + c_i \bar{t}, w^n, i) \\
\quad w^{n, i}, \quad i = 1, \ldots, s \quad \text{intermediate stage value} \\
\quad w^{n, i} = w^n + \bar{t} \sum_{j=1}^{s} \alpha_{ij} \cdot F(t_n + c_j \bar{t}, w_{n, j}) \\
\quad \text{The coefficients} \quad \alpha_{ij}, \quad b_i, \quad \text{and} \\
\quad c_i = \sum_{j=1}^{s} \alpha_{ij}
\end{cases}
\end{align*}
\]

The method
These coefficients are usually collected in a Butcher-array (Tableau):

\[
\begin{array}{c|c}
\mathbf{c} & \mathbf{A} \\
\hline
\mathbf{b}^T & \end{array}
\Rightarrow
\begin{cases}
\text{Explicit if } \mathbf{A} \text{ is lower triangular} \\
\alpha_{ij} = 0 \text{ for } j \geq i
\end{cases}
\]

In CFD, due to high cost of linear solvers, usually \( \mathbf{A} \) is at most diagonally implicit (so each stage requires a single linear solve independently of the other stages).
Order conditions

\[
\begin{align*}
\text{p} = 1 : & \quad b^T c = 1 \\
\text{p} = 2 : & \quad b^T c^2 = 1/2 \\
\text{p} = 3 : & \quad b^T c^3 = 1/3 \\
& \quad b^T A c = 1/6
\end{align*}
\]

The method also has a stage order \( q \)

\[
q = \min_i \left\{ q_i : i, p > q \right\}
\]

\[
W(t_n + c_i \tau) - W_{n,i} = O(\tau^{q + 1})
\]

\[
\text{If } W(t_n) = W_*
\]
Explicit trapezoidal rule

\[
\begin{array}{c|cc}
0 & 1 & 1 \\
\hline
1/2 & 1/2 & 1/2 \\
\end{array}
\]  
(we already studied it)

Explicit midpoint rule:

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

\[
\begin{align*}
W^{n+1} &= W^n + \frac{1}{2} F^{n+1/2} \\
F^{n+1/2} &= F(t_n + \frac{1}{2}, W^{n+1/2}) \\
W^{n+1/2} &= W^n + \frac{1}{2} F^n
\end{align*}
\]

Note that linearly these two are equivalent.

One can also write implicit versions
Rosenbrock methods

Purely implicit methods require nonlinear solves. An alternative is to limit to linear systems only by linearizing $F$ around $(t^n, w^n)$ (or some related point). This gives linearly implicit or Rosenbrock schemes.

\[
\begin{align*}
W'(t) &= F(W(t)) \quad \text{(autonomous)} \\
W^{n+1} &= W^n + \sum_{i=1}^{s} b_i k_i \\
k_i &= \frac{1}{i} F\left(W^n + \sum_{j=1}^{s} a_{ij} k_j\right) + i A \sum_{j=1}^{s} f_{ij} k_j
\end{align*}
\]
where \( A = A^* = \frac{\partial F(w^n)}{\partial w} \) (Jacobian) 

Example: Advection term in Navier-Stokes is \((\nu \cdot D)u\). But one can approximate with \((\nu^n \cdot D)u\), giving a linear system.

Rosenbrock methods are related to fully nonlinear methods if Newton's method is used to solve the nonlinear systems.

Often \( f_i = f_e = \text{const.} \)
\[
I - z f_e A = M = \text{const.}
\]
Examples

\[
\begin{align*}
w^{n+1} &= w^n + k_1 \\
k_1 &= F(w^n) + \frac{1}{2} A w^T k_1
\end{align*}
\]

Second order

\[
A = \frac{\partial F(w^n)}{\partial w} + O(\tau) \leq \text{one can approximate}
\]

Order Reduction

Higher-order RK methods sometimes exhibit a reduction in the order of (global) accuracy when there are certain inhomogeneous boundary conditions.

In fact, the same applies for stiff problems (analysis is hard).
Conclusion of analysis (see Hundsdorfer Verwer)

Any explicit or implicit RK method of classical order \( p \geq 3 \) and stage order \( q < p-1 \) may suffer an order reduction to order \( q+1 \leq p \) for stiff problems or when boundary conditions are present (\( h \) plays the role of stiffness).

There are some mitigation techniques, but the point here is to watch out with high-order methods.
Linear Multistep methods

\[ \frac{1}{h} \sum_{j=0}^{k} \beta_j w^{n+j} = \frac{1}{2} \sum_{j=0}^{p} \beta_j F(t_{n+j}, w^{n+j}) \]

\[ h = \text{number of past values used (memory)} \]

\[ \beta_h = 0 = \text{explicit} \]

\[ \alpha_h = 1 \]

Note that this is as efficient as a single stage of RK.

But one must store \( w \), \( w \), \( w \), \( w \), \( w \).
these methods are not self-starting. Typically a single-step method is used to initialize.

Local order of consistency: $P$

\[
\sum_{j=0}^{k-1} \lambda_j w(t_{n+j}) + \lambda_k w_{n+k} = \sum_{j=0}^{n+k} \beta_j F + \sum_{k=1}^{n+k} \beta_k F
\]

\[w(t_{n+k}) - w_\ast = 0 \left( \tau^{p+1} \right)\]

Plug exact values for past steps and do Taylor series analysis.

Note: Starting values $w_1, \ldots, w_{k-1}$ must also be computed with convergence order $p$. 
Examples: Leap-frog method

\[ W^{n+2} - W^n = 2 \Delta t \cdot F(t_{n+1}, W^{n+1}) \]

(for wave equations)

second-order

Adams-Bashforth methods

2-step:

\[ W^{n+2} - W^{n+1} = \frac{3}{2} \Delta t \cdot F - \frac{\Delta t}{2} \cdot F^n \]

often used to treat advection!

3-step:

\[ W^{n+3} - W^{n+2} = \frac{23}{12} \Delta t \cdot F^{n+2} - \frac{16}{2} \Delta t \cdot F^{n+1} + \frac{5}{12} \Delta t \cdot F^n \]
Backward Differentiation (BDF)

\[ \beta_1 = 1, \quad \beta_j = 0 \text{ otherwise} \]

\[ \frac{3}{2} \, W^{n+2} - 2 \, W^{n+1} + \frac{1}{2} \, W^n = -2 \, F^{n+2} \]

(implicit)

These have very good stability properties for stiff problems. The best approach often is to use a mixed implicit-explicit approach, e.g.

\[ \text{IMEX RK} \]

(implicit-explicit Runge-Kutta)

or splitting approaches
For example:

\[ u_t + a u_x = d u_{xx} \]

\[ u_t + g(n) = L u \]

\( (L \text{ - linear, } g \text{ - nonlinear or difficult due to linear algebra}) \)

Treat \( g(n) \) using Adams–Bashforth \( n-2 \)
but \( L u \) using implicit midpoint (Crank–Nicolson):

\[
\frac{u^{n+1} - u^n}{\Delta t} + \left[ \frac{3}{2} T g^n - \frac{1}{2} T g^{n-1} \right] = L \frac{u^n + u^{n+1}}{2}
\]

\( u_t \) is advection
\( d u_{xx} \) is diffusion

Doing this for higher order is tricky.
Monotonicity

\[ W'(t) = A w(t) \]

- \( a_{ij} \geq 0 \) for \( i \neq j \)
- \( a_{ii} \geq - \alpha, \ x > 0 \)

\( A \) has no eigenvalue on positive real axes

Take a linear one-step method

\[ w^{n+1} = R (\tau A) w^n \]

Stability function

(see old lectures)
Theorem \[ R(\tau A) > 0 \quad \text{iff} \quad \lambda \leq \lambda \]

where \( \lambda \) is the largest \( \lambda \) for which \( R(\tau A) \) and all of its derivatives are positive on \([-\lambda, 0] \).

So here \( \lambda \) is a monotonicity limit on the timestep, similar to stability limits but distinct.

Theorem: Any unconditionally positive method \((\lambda = \infty)\) has order 1.
So one cannot do better than Backward Euler if we want to take very large time steps and preserve non-negativity!

Often we can construct spatial discretizations such that:
- forward Euler is monotone under a condition \( \lambda \tau \leq 1 \) (\( \lambda \tau = 1 \)).

This knowledge can then be used to construct higher-order methods that preserve this property.
Non-linear Positivity

\[ w' = F(t, w) \]

Assume forward Euler is positive:
\[
\begin{align*}
\varrho + \varphi F(t, w) & \geq 0 \\
\text{if } \varrho > 0 \quad \text{and} \quad \varphi \leq 1
\end{align*}
\]

then one can prove that
diagonally-implicit RK methods where
the final update is a convex
combination of forward and
backward Euler steps is also
positive if
\[
\varphi \leq s
\]
where \( s = O(1) \) depends on method
Such RK methods are called Strong Stability Preserving (SSP) methods (Osher & Shu 1988) (generalizes to TVD or in fact any convex functional).

Explicit Trapezoidal Rule (RK2)

\[
\begin{cases}
    w^* = w^n + \tau F^n \\
    w^{**} = w^* + \tau F^*
\end{cases}
\]

First Euler:

\[
w^{n+1} = \frac{1}{2} (w^n + w^{**})
\]

Second Euler:

\[
w^{n+1} = w^n + \frac{1}{2} (\tau F^n + \tau F^*)
\]
RK3 \{ TVD \{ SSP \} \} scheme (explicit)

\[
\begin{align*}
    w^* &= w^n + \frac{3}{2} F^n \quad \text{(first Euler)} \\
    w^{**} &= \frac{3}{4} w^n + \frac{1}{4} \left[ w^* + 2 F^* \right] \quad \text{(convex combo)} \\
    w^{n+1} &= \frac{1}{3} w^n + \frac{2}{3} \left[ w^{**} + 2 F^{**} \right] \quad \text{(convex combo)}
\end{align*}
\]

This scheme is SSP, and its stability function includes a portion of the imaginary axes. Also third-order. Great for advection!