$\alpha = \frac{1}{2}, \beta = 1$: The mid-point method (or second-order Runge-Kutta) has exactly the same stability properties as the Heun method for linear models. Second order accurate and weakly unstable.

$\alpha = 1, \beta = 1$: The forward-backward, Euler-backward or Matsuno method. The forward method is used to predict $\tilde{u}_{n+1}$ and then the result used in an explicit backward step. First order accurate, conditionally stable ($\Delta tf < 1$) and damping (maximized at $\Delta tf = 1/\sqrt{2}$).

### 4.9.1 Derivation of Runge-Kutta methods

We will now analyze the accuracy of the above two-stage schemes.

The Taylor series expansion for $u^{n+1}$ about $t_n$ is:

$$u^{n+1} = u^n + \Delta tu'(t_n) + \frac{1}{2!}\Delta t^2 u''(t_n) + \frac{1}{3!}\Delta t^3 u'''(t_n) + \ldots$$

Since $u'(t_n) = g(u^n, t_n)$ we can write:

$$u' = g$$

$$u'' = \partial_t g + u'\partial_u g = \partial_t g + g\partial_u g$$

$$u''' = \partial_t (\partial_t g + g\partial_u g) = \partial_t^2 g + 2g\partial_t u g + g^2\partial_{uu} g + \partial_u g\partial_t g + g\partial_u g^2$$

so that

$$u^{n+1} = u^n + \Delta t g + \frac{1}{2}\Delta t^2 (\partial_t g + g\partial_u g) + O(\Delta t^3) \quad (4.15)$$

Now we write the algorithm in a series of steps as follows:

$$
\begin{align*}
g_1 &= g(u^n, t_n) \\
u_1 &= u^n + \alpha \Delta tg_1 \\
g_2 &= g(u_1, t_n + \delta \Delta t) \\
u^{n+1} &= u^n + \gamma_1 \Delta tg_1 + \gamma_2 \Delta tg_2
\end{align*}
$$

where we have generalized the algorithm further than before by introducing the arbitrary parameters $\alpha, \delta, \gamma_1$ and $\gamma_2$. The objective now is to manipulate the last step into a form corresponding to (4.15). On inspecting the last step, we see that we need a Taylor expansion of $g_2$ which is:

$$
\begin{align*}
g_2 &= g(u^n + \alpha \Delta tg_1, t_n + \delta \Delta t) \\
&= g(u^n + \alpha \Delta tg_1, t_n) + \delta \Delta t \partial_t g(u^n + \alpha \Delta tg_1, t_n) + O(\Delta t^2) \\
&= g(u^n, t_n) + \alpha \Delta tg_1 \partial_u g(u^n, t_n) + \delta \Delta t \partial_t g(u^n, t_n) + O(\Delta t^2)
\end{align*}
$$
Substituting into the last step of the algorithm we get:

\[
u^{n+1} = u^n + \Delta t (\gamma_1 + \gamma_2) g + \Delta t^2 \gamma_2 (\alpha \partial_t g + \delta g \partial_u g) + O(\Delta t^3)\]

To make terms match with those in equation (4.15) we must choose:

\[
\begin{align*}
\gamma_1 + \gamma_2 &= 1 \\
\gamma_2 \alpha &= \frac{1}{2} \\
\gamma_2 \delta &= \frac{1}{2}
\end{align*}
\]

in which case the scheme is then of order \(O(\Delta t^2)\). These three equations in four unknowns can be solved in terms of just one parameter:

\[
\delta = \alpha \quad ; \quad \gamma_2 = \frac{1}{2\alpha} \quad ; \quad \gamma_1 = 1 - \frac{1}{2\alpha}
\]

The algorithm can now be written:

\[
\begin{align*}
g_1 &= g(u^n, t_n) \\
u_1 &= u^n + \alpha \Delta t g_1 \\
g_2 &= g(u_1, t_n + \alpha \Delta t) \\
u^{n+1} &= u^n + \left(1 - \frac{1}{2\alpha}\right) \Delta t g_1 + \frac{1}{2\alpha} \Delta t g_2
\end{align*}
\]

which corresponds to the two-stage method if we set \(\beta = \frac{1}{2\alpha}\) in equation (4.14). For the two-stage method we found that stability is conditional on \(\alpha \beta > \frac{1}{2}\) and that if \(\alpha \beta = \frac{1}{2}\) then the two-stage method was weakly unstable due to a \(O(\Delta t^3)\) term. This means that the second order accurate Runge-Kutta methods are weakly unstable.

### 4.9.2 Higher order Runge-Kutta

Derivation of higher order Runge-Kutta methods uses the same technique. However, the pages of algebra entailed in find the coefficients are unrevealing. Instead, we supply the “Maple” code to illustrate how to obtain the coefficients:

\[
> n:=3;
\]
alias( G=g(t,u(t)), Gt=D[1](g)(t,u(t)), Gu=D[2](g)(t,u(t)),
    Gtt=D[1,1](g)(t,u(t)), Gtu=D[1,2](g)(t,u(t)), Guu=D[2,2](g)(t,u(t)) );
D(u):=t->g(t,u(t));
TaylorExpr:=(mtaylor(u(t+h),h,n+1)-u(t))/h;
g1:=mtaylor( g(t,u(t)) ,h,n);
g2:=mtaylor( g(t+beta[1]*h,u(t)+h*alpha[1]*g1) ,h,n);
g3:=mtaylor( g(t+beta[2]*h,u(t)+h*alpha[2,1]*g1+h*alpha[2,2]*g2) ,h,n);
RungaKuttaExpr:=( gamma[1]*g1+gamma[2]*g2+gamma[3]*g3 );
eq:=-simplify(RungaKuttaExpr-TaylorExpr);
eqns:={coeffs(eq,[h,G,Gt,Gu,Gtt,Gtu,Guu])};
indets(eqns);
solve(eqns,indets(eqns));

Extending the above script to fourth order involves adding the necessary
definitions for \( u_3 \) and \( g_4 \). The most common fourth order method is:

\[
\begin{align*}
g_1 &= g(u^n, t_n) \\
g_2 &= g(u^n + \frac{1}{2} \Delta t g_1, t_n + \frac{1}{2} \Delta t) \\
g_3 &= g(u^n + \frac{1}{2} \Delta t g_2, t_n + \frac{1}{2} \Delta t) \\
g_4 &= g(u^n + \Delta t g_3, t_n + \Delta t) \\

u^{n+1} &= u^n + \frac{1}{6} \Delta t (g_1 + 2g_2 + 2g_3 + g_4)
\end{align*}
\]

and is widely used. It is both accurate and near neutrally stable. Higher than
fourth order Runge-Kutta methods exist and can be found in textbooks but
are rarely used in models of the ocean or atmosphere.

### 4.10 Side-by-side comparison

A simple P-Z model is

\[
\begin{align*}
N &= N_t - P - Z \\
\partial_t P &= \frac{uPN}{N+N_o} - gZP \\
\partial_t Z &= agZP - dZ
\end{align*}
\]  

(4.16)

where \( N_t = 5, N_o = 0.1, u = 0.03, g = 0.2, a = 0.4 \) and \( d = 0.08 \) are all
constants.
A slightly different model has a wider separation of inherent time-scales and behaves more non-linearly:

\[ N = N_t - P - Z \]
\[ \partial_t P = \frac{uPN}{N + N_o} - \frac{gZP}{P + P_o} \]
\[ \partial_t Z = \frac{agZP}{P + P_o} - dZ \]  (4.17)

where \( N_t = 5, N_o = 0.1, P_o = 0.5, u = 0.01, g = 0.1, a = 1 \) and \( d = 0.08 \) are all constants.
Figure 4.17: Solutions to the P-Z model (equations 4.16) obtained using a “small” $\Delta t = 1$ and the largest “stable” $\Delta t$ for each scheme.
Figure 4.18: Solutions to the P-Z model (equations 4.17) obtained using a “small” $\Delta t = 1$ and the largest “stable” $\Delta t$ for each scheme.