Recall Euler’s method

Let’s for simplicity consider simple first order ODE (notice lack of vector)

\[ y' = f(x, y) \]

There is an exact way to write the solution

\[ y(x) = \int_{x_0}^{x} f(x, y) \, dx \]

However for small interval of \( x, x+h \) we assume that \( f(x, y) \) is constant

\[ \vec{y}(x_{i+1}) = \vec{y}(x_i + h) = \vec{y}(x_i) + \vec{f}(x_i, \vec{y}) h + \mathcal{O}(h) \]

The second-order Runge-Kutta method

Using multi-variable calculus and Taylor expansion, it can be shown

\[ \vec{y}(x_{i+1}) = \vec{y}(x_i + h) = \vec{y}(x_i) + \vec{C}_0 \vec{f}(x_i, \vec{y}) h + \vec{C}_1 \vec{f}(x_i + ph, \vec{y} + qh \vec{f}(x_i, \vec{y})) h + \mathcal{O}(h^3) \]

When

\[ C_0 + C_1 = 1, \quad C_1 p = 1/2, \quad C_1 q = 1/2 \]

There is a lot of possible choices of parameters \( C_0, C_1, p, \) and \( q \) which has no advantage over the others.

One of popular choices is \( C_0 = 0, \quad C_1 = 1, \quad p = 1/2, \) and \( q = 1/2 \) for

**Modified Euler’s method or midpoint method \((\mathcal{O}(h^3))\)**

\[
\begin{align*}
    k_1 &= h \vec{f}(x_i, \vec{y}_i) \\
    k_2 &= h \vec{f}(x_i + \frac{h}{2}, \vec{y}_i + \frac{1}{2} k_1) \\
    \vec{y}(x_i + h) &= \vec{y}_i + k_2
\end{align*}
\]

**The forth-order Runge-Kutta method**

**truncation error \( \mathcal{O}(h^5) \)**

\[
\begin{align*}
    k_1 &= h \vec{f}(x_i, \vec{y}_i) \\
    k_2 &= h \vec{f}(x_i + \frac{h}{2}, \vec{y}_i + \frac{1}{2} k_1) \\
    k_3 &= h \vec{f}(x_i + \frac{h}{2}, \vec{y}_i + \frac{1}{2} k_2) \\
    k_4 &= h \vec{f}(x_i + h, \vec{y}_i + k_3) \\
    \vec{y}(x_i + h) &= \vec{y}_i + \frac{1}{6} (k_1 + 2k_2 + 2k_3 + k_4)
\end{align*}
\]
Matlab ODEs solver

Have a look in help files for ODEs in particular

- **ode45** - adaptive explicit 4th order Runge-Kutta method (good default method)
- **ode23** - adaptive explicit 2nd order Runge-Kutta method
- **ode113** - “stiff” problem solver
- and other

Adaptive stands for no need to chose ‘h’, algorithm will do it by itself. But do remember the rule of not trusting computers. Also run **odeexamples** to see some of the demos for ODEs solvers