Ordinary Differential equations continued

Eugeniy E. Mikhailov

The College of William & Mary

Lecture 20

Recall the Euler's method

We are solving

$$\vec{y}' = \vec{f}(x, \vec{y})$$

There is the exact way to write the solution

$$\vec{y}(x) = \vec{y_0} + \int_{x_0}^{x} \vec{f}(x, \vec{y}) dx$$

The Euler's method assumes that the $\vec{f}(x, \vec{y})$ is constant over a small interval of $(x, x + h)$

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

The second-order Runge-Kutta method

Using the multi-variable calculus and the Taylor expansion

$$\vec{y}(x_{i+1}) = \vec{y}(x_i + h) =$$
$$= \vec{y}(x_i) + C_0 \vec{f}(x_i, \vec{y}_i) h + C_1 \vec{f}(x_i + ph, \vec{y}_i + qh\vec{f}(x_i, \vec{y}_i)) h + \mathcal{O}(h^3)$$

where $C_0 + C_1 = 1, \quad C_1 p = 1/2, \quad C_1 q = 1/2$ see[1].
There are a lot of possible choices of parameters C_0, C_1, p, and q.
One choice generally has no advantage over another.
One intuitive choice is $C_0 = 0$, $C_1 = 1$, $p = 1/2$, and $q = 1/2$ gives

Modified Euler's method or midpoint method (error $\mathcal{O}(h^3)$)

$$k_1 = h\vec{f}(x_i, \vec{y}_i)$$
$$k_2 = h\vec{f}(x_i + \frac{h}{2}, \vec{y}_i + \frac{1}{2}k_1)$$
$$\vec{y}(x_i + h) = \vec{y}_i + k_2$$

[1] Holistic Numerical Methods

The forth-order Runge-Kutta method

Higher order expansion leads to another possible choice

The forth-order Runge-Kutta method with truncation error $\mathcal{O}(h^5)$

$$
\begin{aligned}
k_1 &= h\vec{f}(x_i, \vec{y}_i) \\
k_2 &= h\vec{f}(x_i + \frac{h}{2}, \vec{y}_i + \frac{1}{2}k_1) \\
k_3 &= h\vec{f}(x_i + \frac{h}{2}, \vec{y}_i + \frac{1}{2}k_2) \\
k_4 &= h\vec{f}(x_i + h, \vec{y}_i + k_3) \\
\vec{y}(x_i + h) &= \vec{y}_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)
\end{aligned}
$$

Matlab built-in ODEs solvers

Have a look in help files for ODEs. In particular, pay attention to

- `ode45` - adaptive explicit 4th order Runge-Kutta method (good default method)
- `ode23` - adaptive explicit 2nd order Runge-Kutta method
- `ode113` - "stiff" problem solver
- and others

Adaptive stands for no need to choose 'h', the algorithm will do it by itself. However, remember the rule about not trusting a computer's choice.

Run `odeexamples` to see some of the demos for ODEs solvers