Modelling Biochemical Reaction Networks

Lecture 7: Numerical integration of ordinary differential equations

Marc R. Roussel

Department of Chemistry and Biochemistry

University of Lethbridge

Recommended reading

- Fall, Marland, Wagner and Tyson, section 1.4.1

Ordinary differential equation initial value problems

- An ordinary differential equation (ODE) is an equation of the form

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, t)$$

  $\mathbf{x}$ is a vector describing the state of a system (e.g. the concentrations) and $\mathbf{f}(\mathbf{x}, t)$ is a vector-valued function.
- In an initial value problem (IVP), we are provided with $\mathbf{x}$ at some initial time t_0 and want to get $\mathbf{x}$ for $t > t_0$.
- A solution of an ODE IVP is a function $\mathbf{x}(t)$ satisfying the equation and the initial condition.
- Very few ODEs have analytic solutions, so we generally need to use approximate numerical integration methods.

Numerical integration of ordinary differential equations

- Basic idea: Approximate the derivative as

$$\frac{d\mathbf{x}}{dt} \approx \frac{\Delta\mathbf{x}}{\Delta t} = \mathbf{f}(\mathbf{x}, t)$$

- The continuous solution $\mathbf{x}(t)$ is replaced by values of $\mathbf{x}$ at a discrete set of times: $(t_i, \mathbf{x}_i)$ $(i = 1, 2, \ldots)$ with $\mathbf{x}_0 = x(0)$.
- Need to decide
  1. what exactly we mean by $\Delta\mathbf{x}$;
  2. how to choose Δt; and
  3. at what $\mathbf{x}$ and t we're going to evaluate $\mathbf{x}$.

Euler's method

- Simplest possible method:

$$\Delta \mathbf{x} = \mathbf{x}_{i+1} - \mathbf{x}_i$$
$\Delta t = t_{i+1} - t_i$ is usually constant
$\mathbf{f}$ evaluated at $\mathbf{x}_i$ and t_i

$$\frac{\Delta x}{\Delta t} = \frac{\mathbf{x}_{i+1} - \mathbf{x}_i}{\Delta t} = \mathbf{f}(\mathbf{x}_i, t_i)$$
$$\therefore \mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{f}(\mathbf{x}_i, t_i)\Delta t$$

- `xppaut` example: $\frac{dx}{dt} = -x$, $x(0) = 1$

Runge-Kutta methods

- The problem with the Euler method is that it uses the derivative at a single point and extrapolates from there.
- Many numerical methods calculate the derivative using one or more intermediate points in order to obtain more refined estimates of the average derivative over one time step.
- Runge-Kutta methods impose the condition that the Taylor series of $\mathbf{x}(t+h)$ in powers of $h = \Delta t$ should match its approximation by the numerical method.
- Runge-Kutta methods involve multiple stages of computation in which we use rates computed at previous points to estimate the position of intermediate points.

Two-stage Runge-Kutta method

- For simplicity, consider a scalar differential equation.
- Take one step of size $h = \Delta t$.
- Use one intermediate point:
  - $t_{\text{intermed}} = t_i + ch$ where c is a coefficient between 0 and 1
  - Use Euler's method to estimate

$$x_{\text{intermed}} \approx x_i + chf(x_i, t_i)$$

- Blend the two derivatives at (t_i, x_i) and $(t_{\text{intermed}}, x_{\text{intermed}})$ to obtain the estimate of x_{i+1}:

$$x_{i+1} = x_i + h\left[a_1 f(x_i, t_i) + a_2 f\left(x_i + chf(x_i, t_i), t_i + ch\right)\right]$$

- We need to choose values for the coefficients a_1, a_2 and c.

Two-stage Runge-Kutta method

- Taylor expansion in powers of h of $x_{i+1} = x(t_i + h)$:

$$
\begin{aligned}
x(t_i + h) &= x(t_i) + hx'(t_i) + \frac{h^2}{2}x''(t_i) + O(h^3) \\
&= x_i + hf(x_i, t_i) + \frac{h^2}{2}\left.\frac{d}{dt}x'(t)\right|_{(t_i,x_i)} + O(h^3) \\
&= x_i + hf(x_i, t_i) + \frac{h^2}{2}\left.\frac{d}{dt}f(x,t)\right|_{(t_i,x_i)} + O(h^3) \\
&= x_i + hf(x_i, t_i) + \frac{h^2}{2}\left[\frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial t}\right]_{(t_i,x_i)} + O(h^3) \\
&= x_i + hf(x_i, t_i) + \frac{h^2}{2}\left[f(x_i, t_i)\left.\frac{\partial f}{\partial x}\right|_{(t_i,x_i)} + \left.\frac{\partial f}{\partial t}\right|_{(t_i,x_i)}\right] \\
&\quad + O(h^3)
\end{aligned}
$$

Two-stage Runge-Kutta method

- Taylor expansion in powers of h of the right-hand side (rhs) $x_i + h\left[a_1 f(x_i, t_i) + a_2 f\left(x_i + chf(x_i, t_i), t_i + ch\right)\right]$:

$$\begin{aligned}
\text{rhs} = x_i + h \Bigg\{ & a_1 f(x_i, t_i) \\
& + a_2 \left[f(x_i, t_i) + chf(x_i, t_i) \left.\frac{\partial f}{\partial x}\right|_{(t_i, x_i)} + ch \left.\frac{\partial f}{\partial t}\right|_{(t_i, x_i)} \right] \Bigg\} \\
& + O(h^3) \\
= x_i + h(a_1 + a_2) f(x_i, t_i) & + a_2 ch^2 \left[f(x_i, t_i) \left.\frac{\partial f}{\partial x}\right|_{(t_i, x_i)} + \left.\frac{\partial f}{\partial t}\right|_{(t_i, x_i)} \right] \\
& + O(h^3)
\end{aligned}$$

Two-stage Runge-Kutta method

- Now compare the two Taylor expansions:

$$x_i + hf(x_i, t_i) + \frac{h^2}{2}\left[f(x_i, t_i)\left.\frac{\partial f}{\partial x}\right|_{(t_i,x_i)} + \left.\frac{\partial f}{\partial t}\right|_{(t_i,x_i)}\right]$$

$$= x_i + h(a_1 + a_2)f(x_i, t_i) + a_2 ch^2\left[f(x_i, t_i)\left.\frac{\partial f}{\partial x}\right|_{(t_i,x_i)} + \left.\frac{\partial f}{\partial t}\right|_{(t_i,x_i)}\right]$$

- They are the same if $a_1 + a_2 = 1$ and $a_2 c = \frac{1}{2}$.
- Can write everything in terms of one parameter, say a_2: $a_1 = 1 - a_2$ and $c = \frac{1}{2a_2}$ $(a_2 \neq 0)$.

Two-stage Runge-Kutta method

- We get a family of two-stage (and second-order, i.e. with an error $O(h^3)$) Runge-Kutta methods parameterized by a_2:

$$x_{i+1} = x_i + h \left[(1 - a_2) f(x_i, t_i) + a_2 f\left(x_i + \frac{1}{2a_2} h f(x_i, t_i), t_i + \frac{1}{2a_2} h \right) \right]$$

- Examples:

  $a_2 = 1$: $x_{i+1} = x_i + h f\left(x_i + \frac{h}{2} f(x_i, t_i), t_i + \frac{h}{2} \right)$
  (Midpoint rule)

  $a_2 = \frac{1}{2}$: $x_{i+1} =$
  $x_i + h \left[\frac{1}{2} f(x_i, t_i) + \frac{1}{2} f\left(x_i + h f(x_i, t_i), t_i + h \right) \right]$
  (Improved Euler)

Two-stage Runge-Kutta method

Example: $\frac{dx}{dt} = x(1-x)$, $x(0) = 0.1$

(a version of the logistic equation)

- For this ODE, $f(x,t) = x(1-x)$.
- The general two-stage Runge-Kutta method is

$$x_{i+1} = x_i + h\left[(1-a_2)f(x_i,t_i) + a_2 f\left(x_i + \frac{1}{2a_2}hf(x_i,t_i), t_i + \frac{1}{2a_2}h\right)\right]$$

- This is a map, a rule for calculating x_{i+1} from x_i.
- Maps can be implemented in `xppaut`.