Introduction to Dynamical Systems

Part 2

Icahn School of Medicine at Mount Sinai

SBCNY

Outline

Euler's method for solving ODE systems

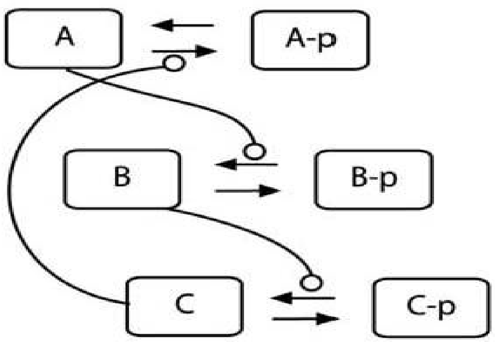
General concepts

Example with a single variable

Expanding Euler's method to multivariable systems

Biochemical signaling described by system of ODEs

In the last lecture, we discussed the following 3-component network



$$\frac{d[A]}{dt} = \frac{k_{p1}([A]_T - [A])}{[A]_T - [A] + K_{p1}} - \frac{k_{k1}[A][C]}{[A] + K_{k1}}$$

$$\frac{d[B]}{dt} = \frac{k_{p2}([B]_T - [B])[A]}{[B]_T - [B] + K_{p2}} - \frac{k_{k2}[B]}{[B] + K_{k2}}$$

$$\frac{d[C]}{dt} = \frac{k_{p3}([C]_T - [C])[B]}{[C]_T - [C] + K_{p3}} - \frac{k_{k3}[C]}{[C] + K_{k3}}$$

Mogilner *et al.*, *Developmental Cell* 11:279–287, 2006

Now we wish to address: how can we simulate this numerically?

Solving ordinary differential equations

Euler's method

$$\frac{dx}{dt} = f(x) \qquad x(t=0) = x_0$$

$$\frac{dx}{dt} \approx \frac{x(t+\Delta t) - x(t)}{\Delta t} \qquad \frac{x(t+\Delta t) - x(t)}{\Delta t} = f(x)$$

$$x(t + \Delta t) = x(t) + f(x) \cdot \Delta t$$

So, we start with $x(0)$, which is known

$$x(\Delta t) = x(0) + f(x(0)) \cdot \Delta t$$

Now $x(\Delta t)$ is known

$$x(2\Delta t) = x(\Delta t) + f(x(\Delta t)) \cdot \Delta t$$

$$x(3\Delta t) = x(2\Delta t) + f(x(2\Delta t)) \cdot \Delta t$$

etc.

Leonhard Euler
(1707-1783)

Remember from calculus

$$\frac{dx}{dt} = \lim_{\Delta t \to 0} \frac{x(t+\Delta t) - x(t)}{\Delta t}$$

Euler's method example

$$\frac{dx}{dt} = a - bx \qquad\qquad x(t=0) = c$$

Assume a=20, b=2, c=5

We can write simple MATLAB code to solve this numerically

```
a = 20 ;
b = 2 ;
c = 5 ;

dt    = 0.05 ;
tlast = 2 ;

iterations = round(tlast/dt) ;
xall = zeros(iterations,1) ;

x = c ;
for i = 1:iterations
  xall(i) = x ;
  dxdt = a - b*x ;
  x = x + dxdt*dt ;
end % of this time step

time = dt*(0:iterations-1)' ;
figure
plot(time,xall)
```


Euler's method example

What does this code actually do?

$\frac{dx}{dt} = a - bx$ a=20; b=2; x_0=5

x = 5

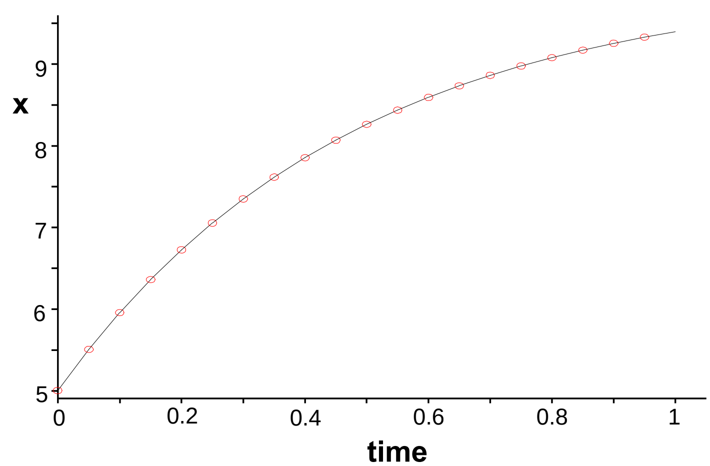
dx/dt = 20 – 2*5 = 10

x = 5 + 10*0.05 = 5.5

dx/dt = 20 – 2*5.5 = 9

x = 5.5 + 9*0.05 = 5.95

etc.



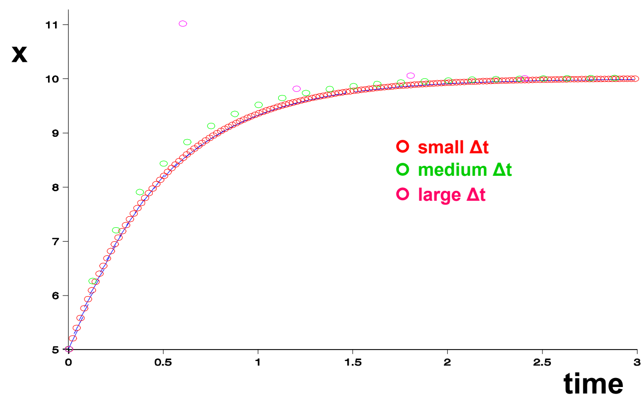
Euler's method example

$$\frac{dx}{dt} = a - bx \qquad x(t=0) = c \qquad \text{a=20; b=2; c=5}$$

This simple differential equation has an analytical solution

$$x(t) = a/b - (a/b - c)\cdot e^{-bt}$$

For small values of Δ*t*, the numerical solution is accurate



Notes on Euler's method

Extending this to systems of ODEs is straightforward

$$\frac{dx}{dt} = f(x,y,z) \qquad \frac{dy}{dt} = g(x,y,z) \qquad \frac{dz}{dt} = h(x,y,z)$$

$$\mathbf{v} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad \mathbf{v}(\Delta t) = \mathbf{v}(0) + \Delta t \cdot \begin{bmatrix} f(x,y,z) \\ g(x,y,z) \\ h(x,y,z) \end{bmatrix}$$

Solutions can become highly unstable if Δ*t* is too large

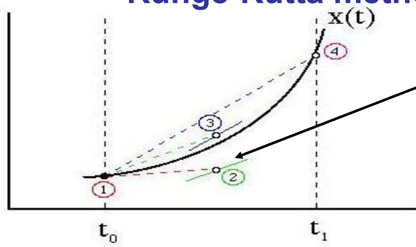
In MATLAB implementations, one of two things commonly happen:
1) variables achieve values such as 4.3 x 10^{78}
2) Strictly non-negative variables (concentrations) become negative

In over two centuries since Euler's death, applied mathematicians have devised more stable and accurate methods. These are the algorithms implemented by MATLAB's built in solvers (e.g. `ode23`, `ode15s`).

Other numerical methods for ODE systems

Euler died more than 200 years ago – since then, improvements to his algorithm have been made

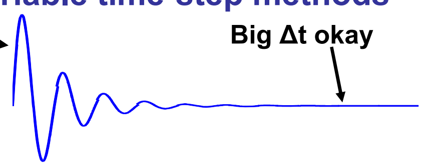
Runge-Kutta method



Also compute dx/dt between t_0 and t_1; use a weighted average

Variable time-step methods

Small Δt required



Big Δt okay

These algorithms are available in MATLAB as the built-in ODE solvers

Model structure to solve an ODE system

The Euler's method MATLAB script is structured as follows:

1) Define constants

2) Set time step, simulation time, etc.

3) Set initial conditions

4) A "for" loop to simulate evolution of time

   At each time step:

   write output if needed
   compute $\boxed{dX/dt}$
   compute *X* at the next time step

   X here refers to vector of ALL state variables

5) Plot and output results

Summary

Euler's method is the simplest and most straightforward algorithm for solving ODEs numerically

Euler's method is based on approximating the derivative for small values of Δt

$$\frac{dx}{dt} = \lim_{\Delta t \to 0} \frac{x(t+\Delta t) - x(t)}{\Delta t} \qquad \frac{dx}{dt} \approx \frac{x(t+\Delta t) - x(t)}{\Delta t} \text{ for small } \Delta t$$

Because Euler's method sometimes fails, more complex numerical algorithms are preferred for most models of biological interest

Self-assessment question

You are working with an array **`A`**, dimensions 100 x 4. You also have a vector **`time`**, dimensions 100 x 1. Each column in **`A`** represents a different variable measured in your experiment. Each row represents the corresponding time point in the vector **`time`**. You wish to write a **`for`** loop to plot 4 time courses in different colors. You paste the following lines into your command window:

```
colors = ‘krgb’ ;
for i=1:4
  plot(time,A(i))
end
```

This does not produce the desired result for 3 reasons. Why not?