Workshop Series Summer 2023

CAMBAM

Centre for Applied Mathematics
in Bioscience and Medicine

June 16th, 2023

Exploring Single Neuron Excitability with Mathematical and Computational Models

By Niklas Brake and Nils Koch

Numerical simulations

How do we go from math equations to plots of voltage dynamics?

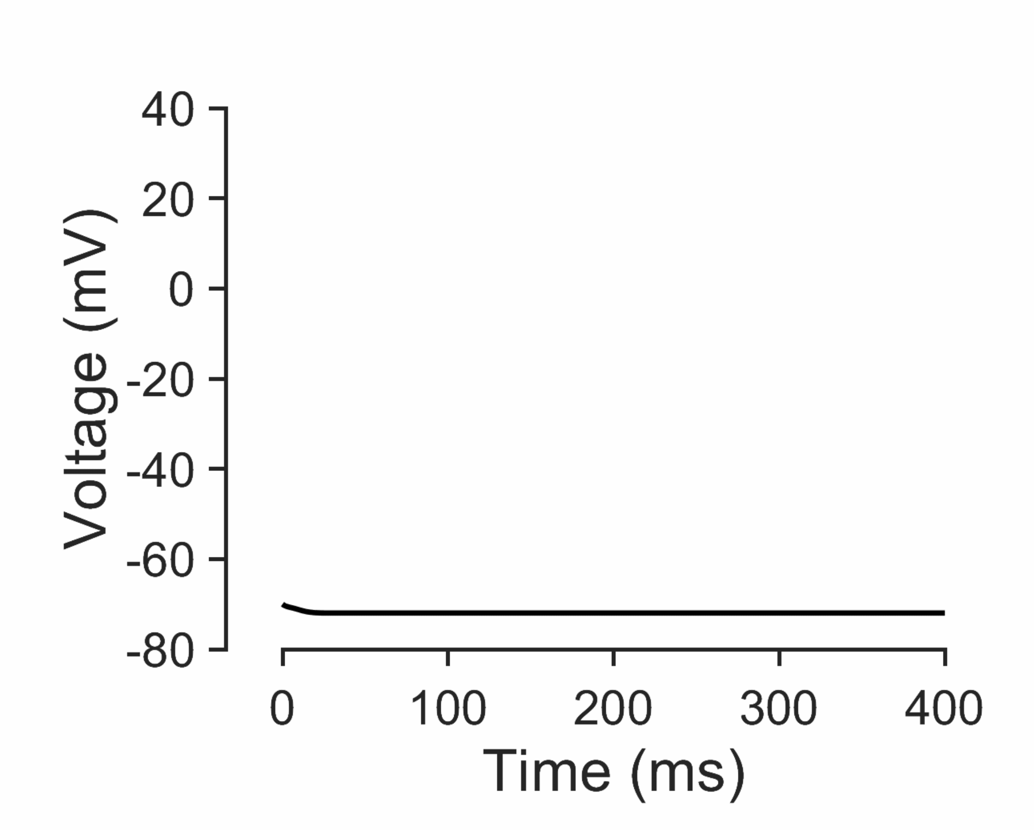
$$C_m \frac{dV}{dt} = I_{app} - \bar{g}_{Na}\boldsymbol{m}^3\boldsymbol{h}(\boldsymbol{V} - E_{Na}) - \bar{g}_K\boldsymbol{n}^4(\boldsymbol{V} - E_k) - g_L(\boldsymbol{V} - E_L)$$

$$\frac{d\boldsymbol{h}}{dt} = 0.07\exp\left(-\frac{\boldsymbol{V}}{20}\right)(1-\boldsymbol{h}) - \frac{1}{\exp\left(\frac{30-\boldsymbol{V}}{10}\right)+1}\boldsymbol{h}$$

$$\frac{d\boldsymbol{m}}{dt} = \frac{0.1(25-\boldsymbol{V})}{\exp\left(\frac{25-\boldsymbol{V}}{10}\right)-1}(1-\boldsymbol{m}) - 4\exp\left(-\frac{\boldsymbol{V}}{18}\right)\boldsymbol{m}$$

$$\frac{d\boldsymbol{n}}{dt} = \frac{0.01(10-\boldsymbol{V})}{\exp\left(\frac{10-\boldsymbol{V}}{10}\right)-1}(1-\boldsymbol{n}) - 0.125\exp\left(-\frac{\boldsymbol{V}}{80}\right)\boldsymbol{n}$$

$C_m = 1\ \mu\text{F/cm}^2$	$g_L = 0.3\ \text{mS/cm}^2$	$E_L = 10\ \text{mV}$
$\boldsymbol{g_{Na}}(\boldsymbol{V}) = \bar{g}_{Na}\boldsymbol{m}^3(\boldsymbol{V})\boldsymbol{h}(\boldsymbol{V})$	$\bar{g}_{Na} = 120\ \text{mS/cm}^2$	$E_{Na} = 115\ \text{mV}$
$\boldsymbol{g_K}(\boldsymbol{V}) = \bar{g}_K\boldsymbol{n}^4(\boldsymbol{V})$	$\bar{g}_K = 36\ \text{mS/cm}^2$	$E_K = -12\ \text{mV}$



Numerical simulations: Euler's Method

$$\frac{dx}{dt} = \frac{x(t+\delta t) - x(t)}{\delta t}$$

as $\delta t \to 0$

Numerical simulations: Euler's Method

$$x(t+\delta t) = x(t) + \delta t \frac{dx}{dt}$$

as $\delta t \to 0$

This is exact as $\delta t \to 0$, but we can get an approximate solution for $0 < \delta t \ll 1$

Numerical simulations: Euler's Method

$$x(t+h) = x(t) + \dot{x}(t)h + O\left(h^2\right)$$

$$x(t+h) \approx x(t) + h\frac{dx}{dt}$$

This algorithm is recursive. One must specify the initial conditions $x(0)$

ALGORITHM euler

input: $t_{max}, f(t,x), x_0, h$

output: x

$N \leftarrow \mathrm{t_{max}}/h$

$x[0] \leftarrow x_0$

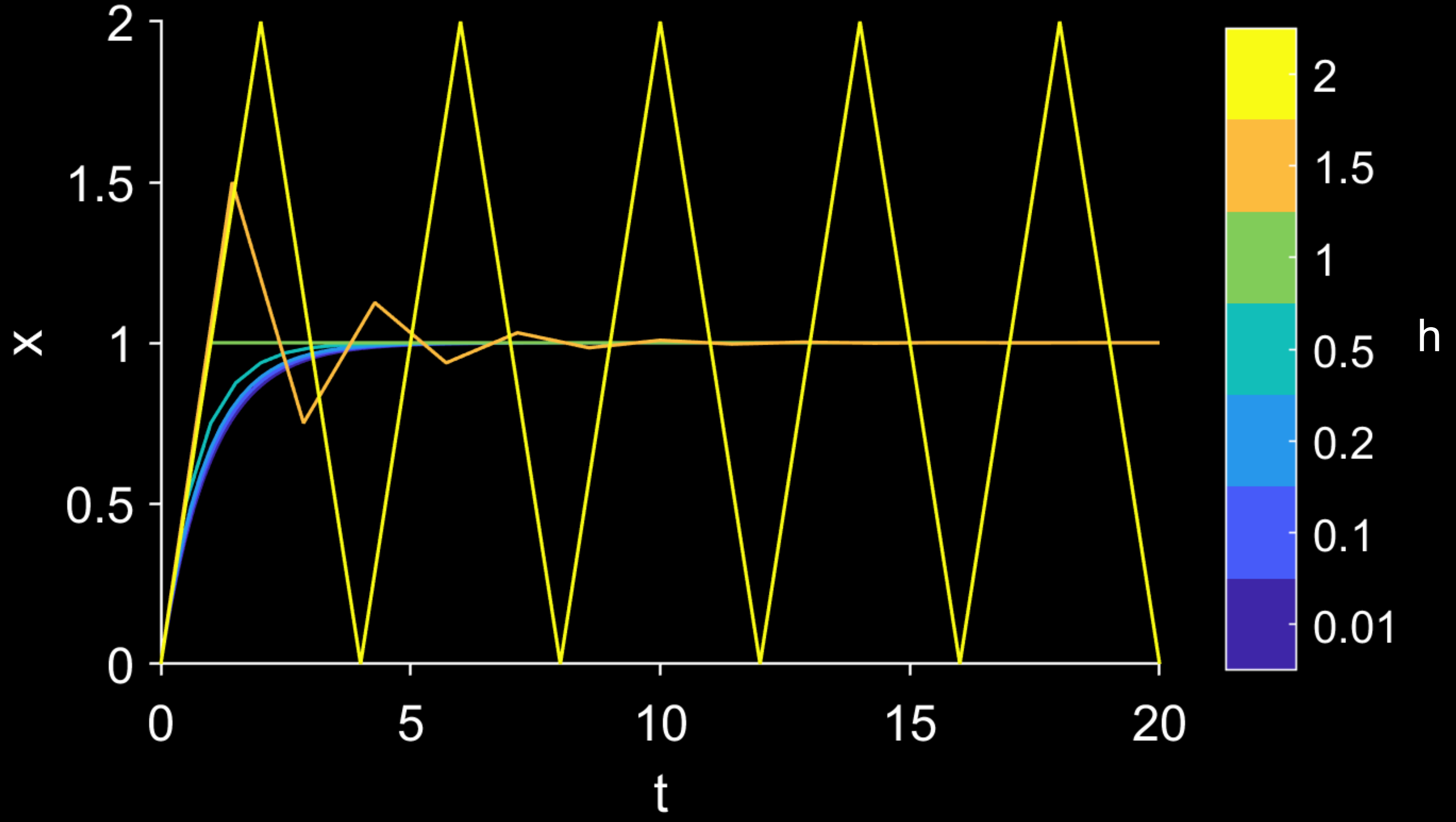
for i from 0 to $N-1$

$x[i+1] = x[i] + f(h \cdot i, x[i])h$

end

Numerical simulations: Euler's Method

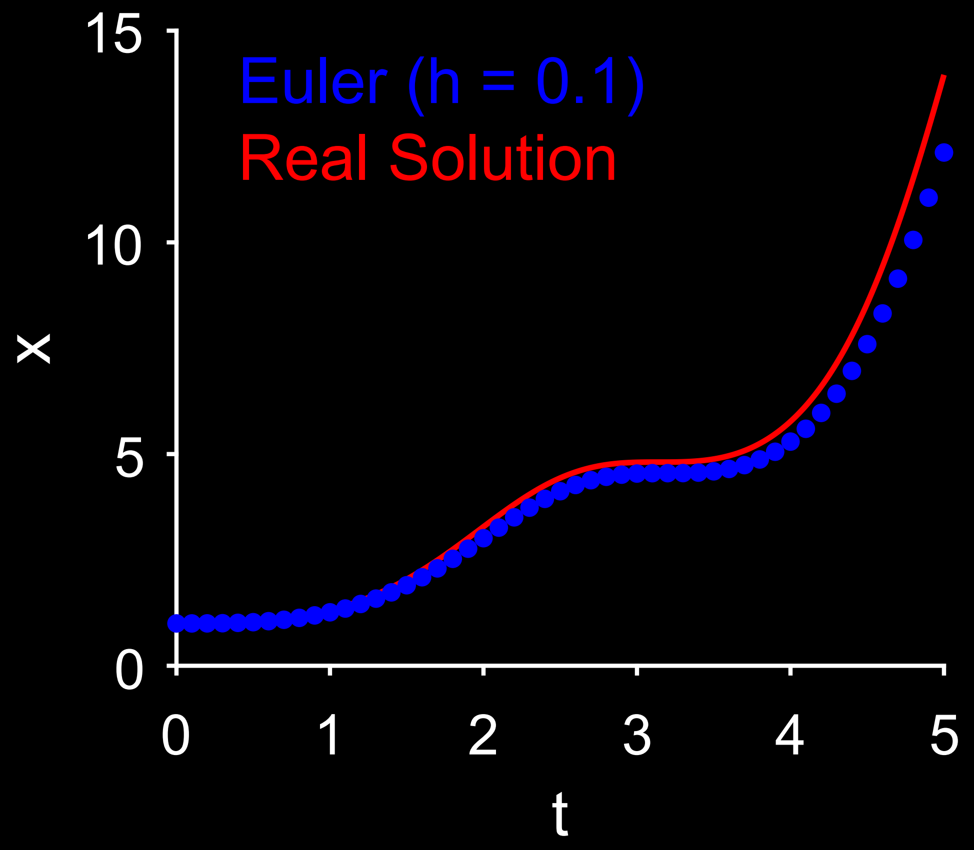
Using Euler's method with different step sizes, h, to simulate $\frac{dx}{dt} = 1 - x$

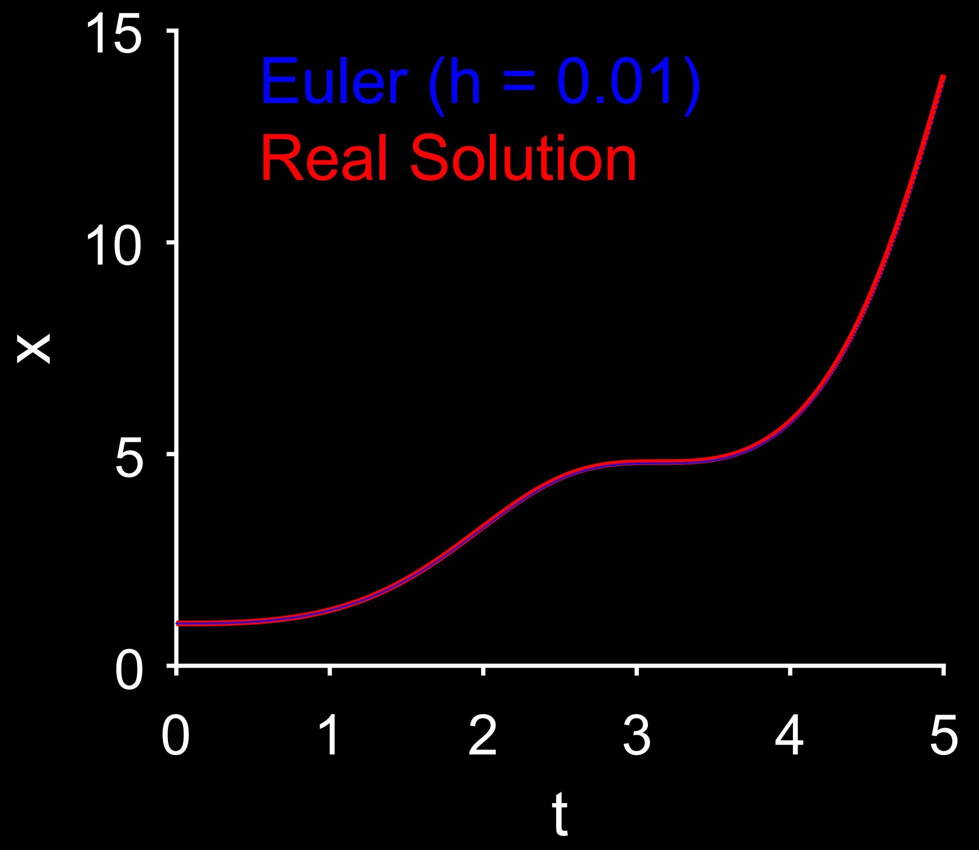


Numerical simulations: Euler's Method

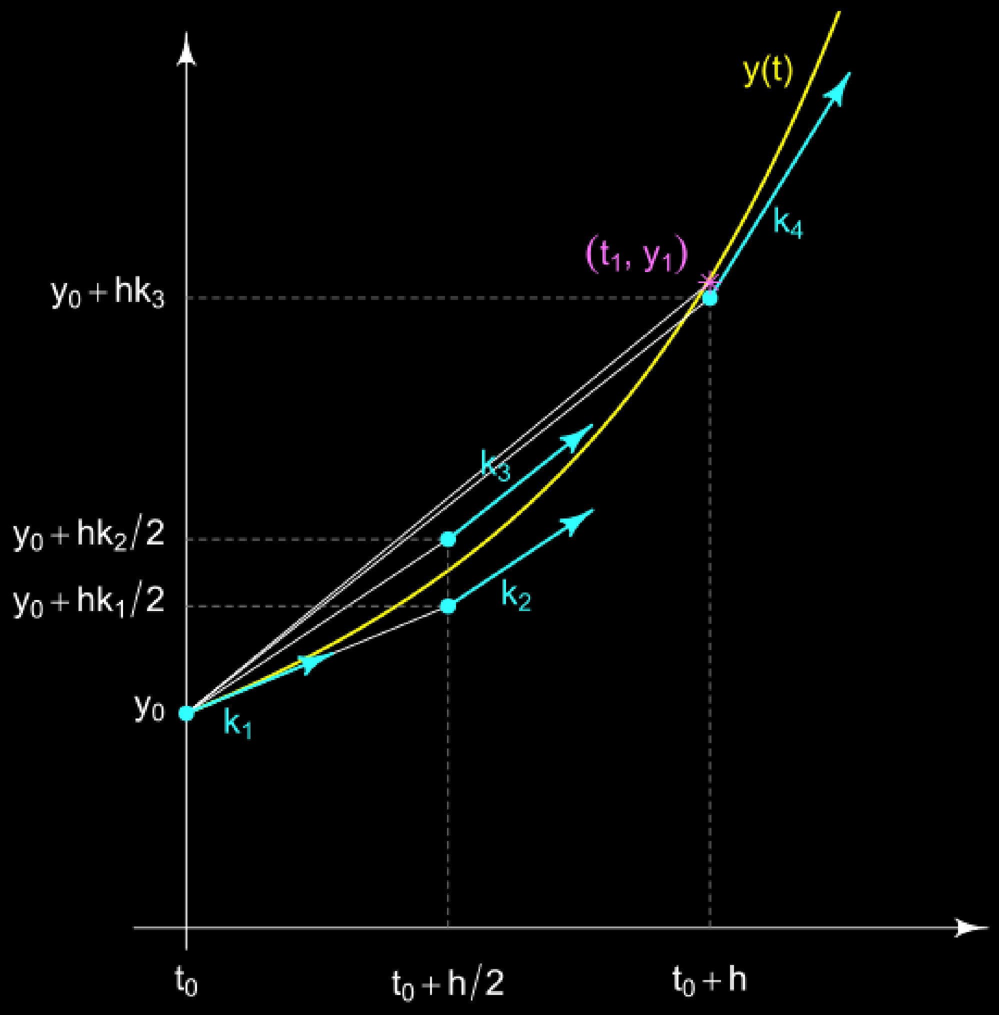
Error is not always obvious!

$$dx/dt = \sin(t)^2 x$$





Numerical simulations: Higher order Runge-Kutta Methods



Euler's Method (RK1)

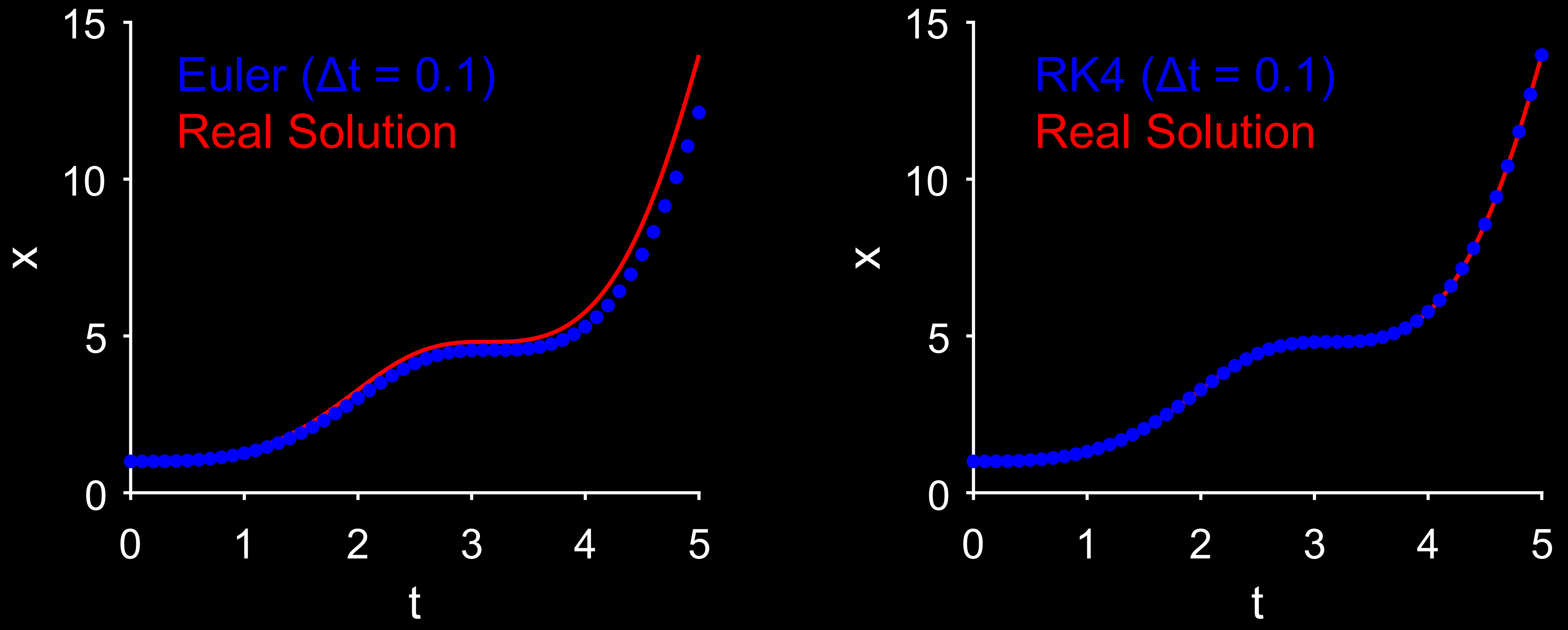
$$y_{n+1} = y_n + hk_1$$
$$k_1 = f(t_n, y_n)$$

4th Order Runge-Kutta

$$y_{n+1} = y_n + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4)$$
$$k_1 = f(t_n, y_n)$$
$$k_2 = f\left(t_n + \frac{h}{2}, y_n + h\frac{k_1}{2}\right)$$
$$k_3 = f\left(t_n + \frac{h}{2}, y_n + h\frac{k_2}{2}\right)$$
$$k_4 = f(t_n + h, y_n + hk_3)$$

Numerical simulations: Higher order Runge-Kutta Methods

$$dx/dt = \sin(t)^2 x$$

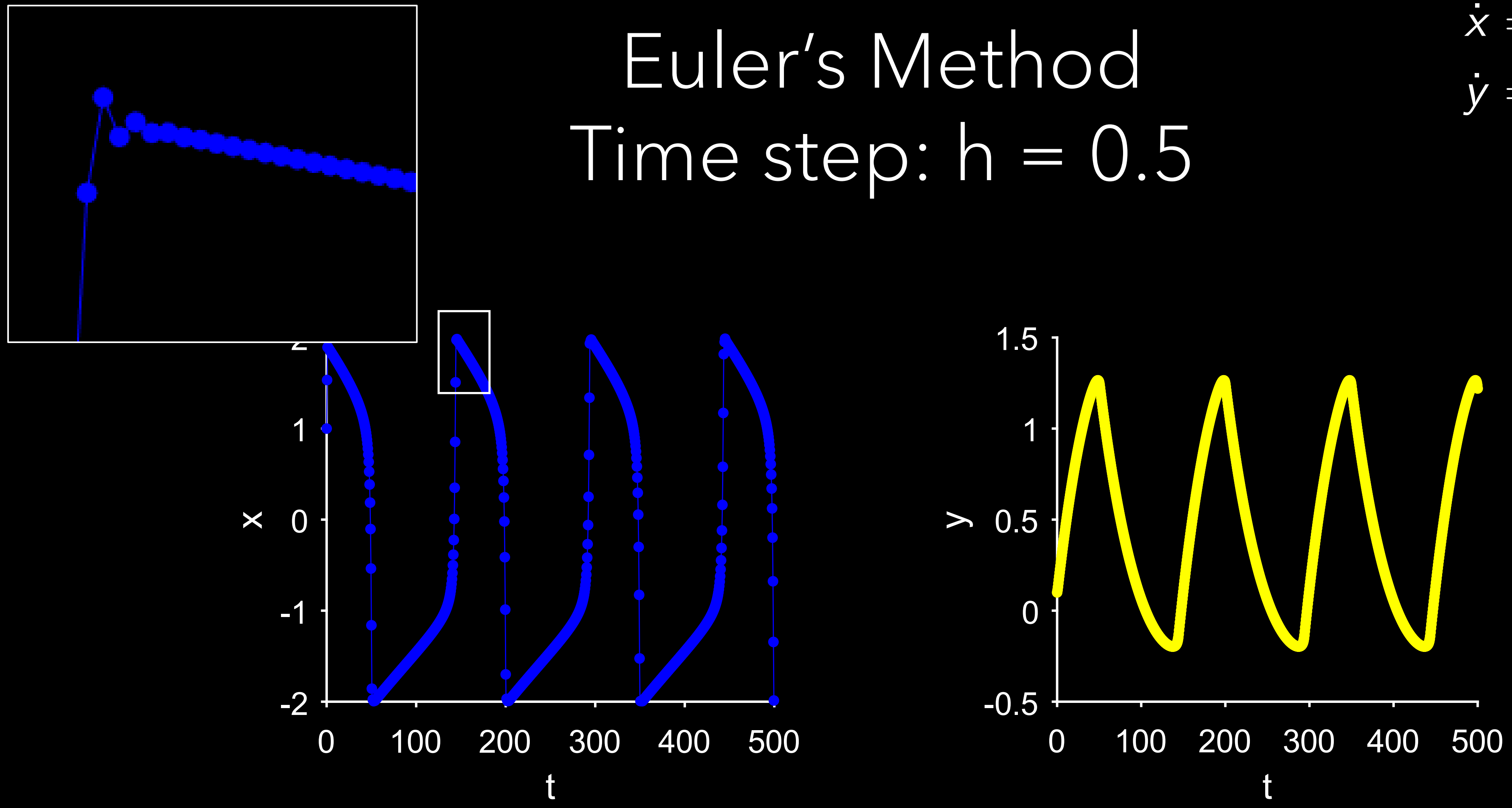


Numerical simulations: Computing Fitzhugh-Nagumo

$$\dot{x} = x - x^3/3 - y + I$$
$$\dot{y} = 1/\tau\,[x + a - by]$$

Euler's Method

Time step: h = 0.5



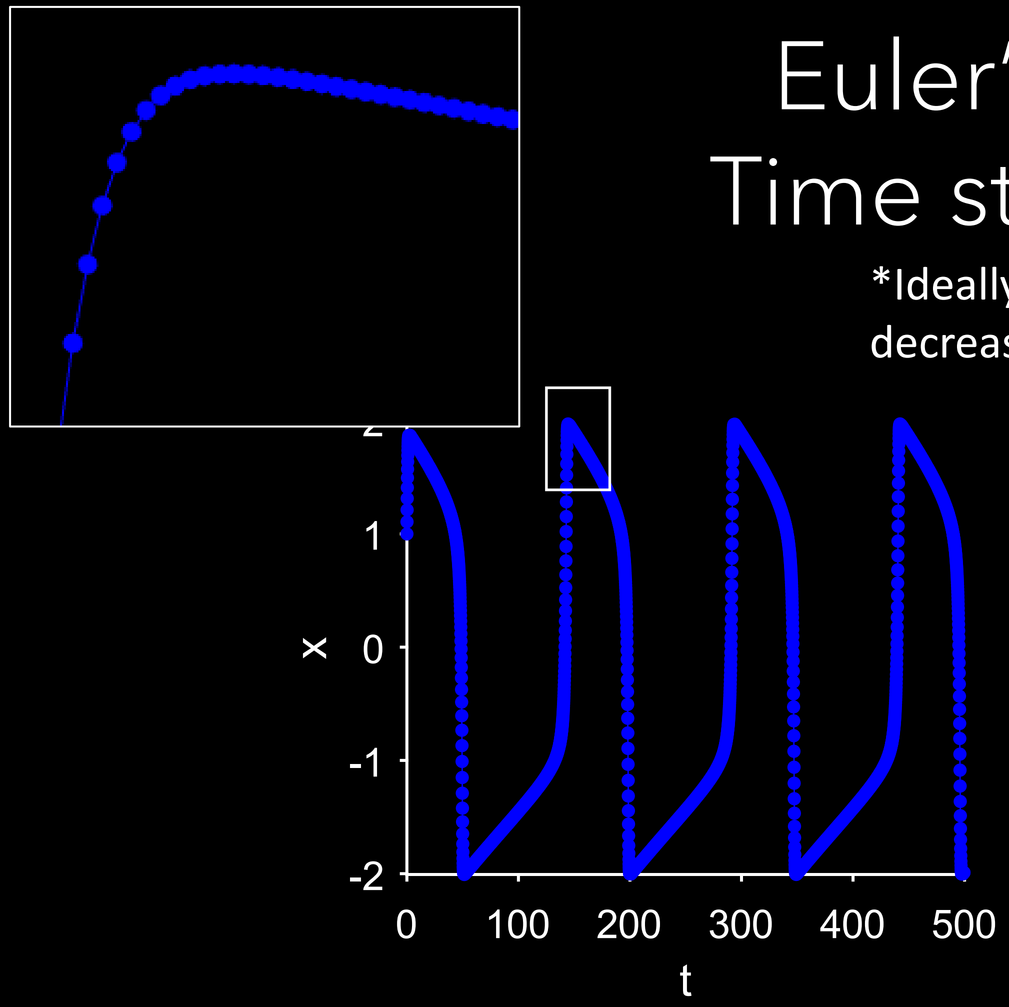
Numerical simulations: Computing Fitzhugh-Nagumo

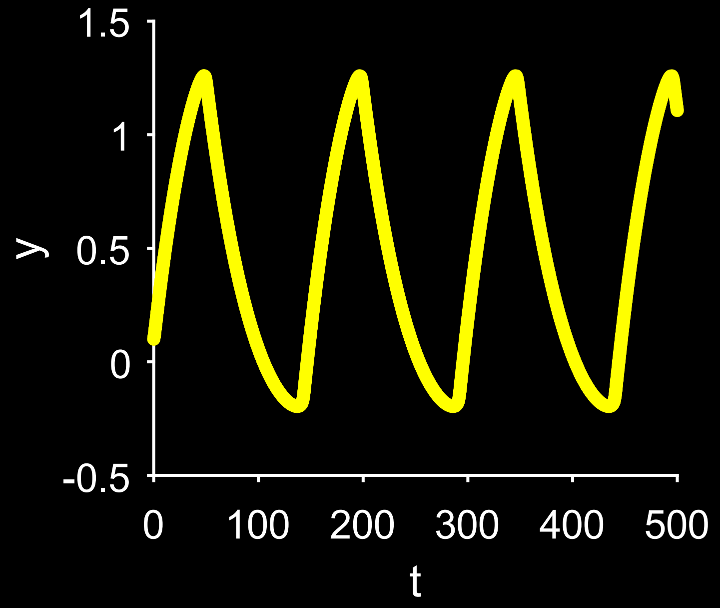
$$\dot{x} = x - x^3/3 - y + I$$
$$\dot{y} = 1/\tau\,[x + a - by]$$

Euler's Method

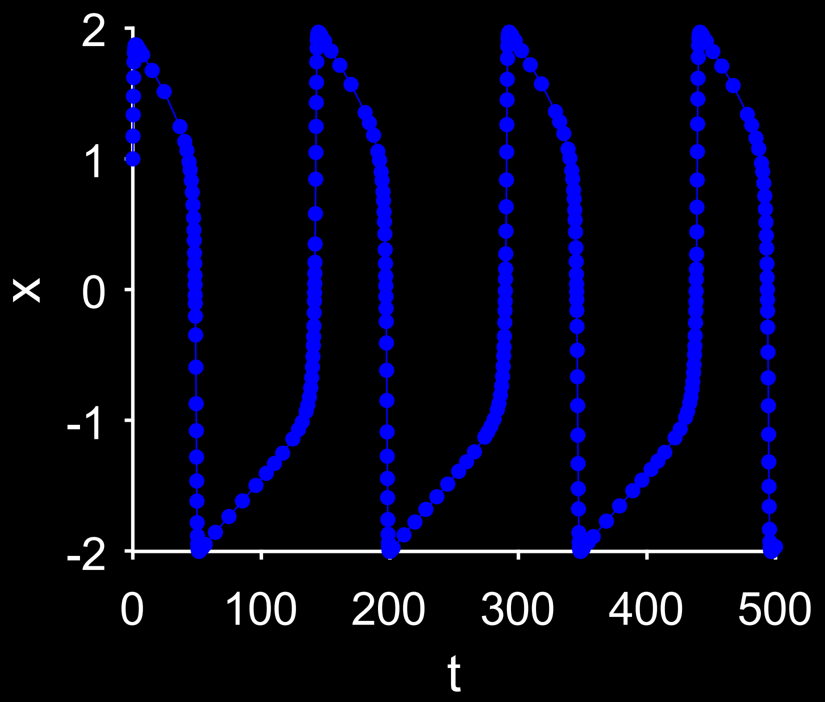
Time step: h = 0.1

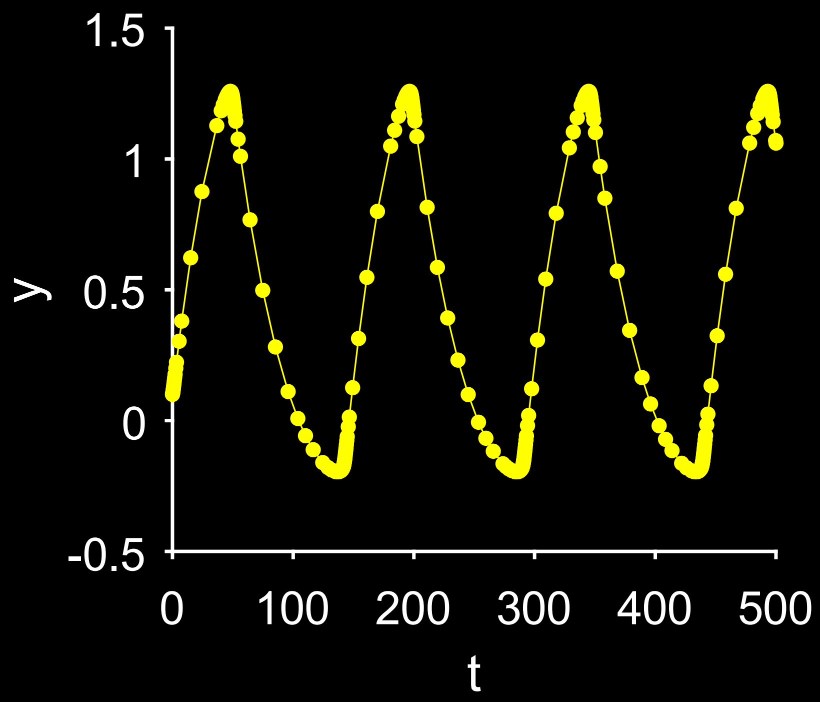
*Ideally, we would only decrease h at the cusps.





Numerical simulations: Adaptive step size





Instead of specifying timestep, we specify ***error tolerance*** either as

- Relative Error Tolerance, e.g., simulation should always be within 0.1% of actual solution (as above)
- Absolute Error Tolerance, e.g., simulation should always be within 0.05 of actual solution

Be mindful that simulation output will not necessarily be equally spaced in time

Numerical simulations: Conclusion

Euler's Method: easy to implement quickly. May need very small step size to simulate equations accurately.

Input specifications: (1) step size

Runge-Kutta: Generalization of Euler's Method. More accurate and allows for larger step size but performs more calculations at every iteration.

Input specifications: (1) step size (2) May need to specify the "order" (4^{th} order is most common, 1^{st} order is equivalent to Eurler's method).

Adaptive Time Step: Finds optimal step size for each iteration but therefore requires more calculations each iteration. Can be faster and use less memory if optimal step size changes a lot.

Input specifications: Either a relative or absolute error tolerance (or both)

Depending on the library being used, there may be default options and these choice may therefore be hidden. However, these options are still there behind the scenes, and you can specify them if your simulations aren't looking right.