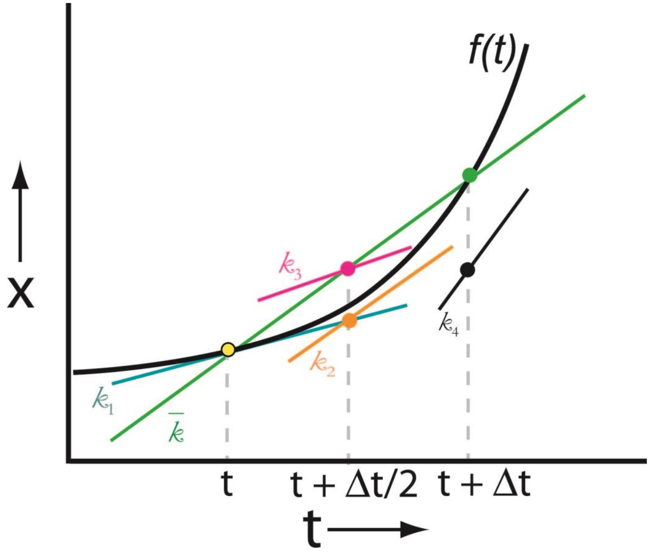


Numerical solution of ODE-2

Problem with Euler method

The curve representing the function is non-linear.
But we are doing linear extrapolation using the slope estimated at a time point

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

Estimate of X after Δt:

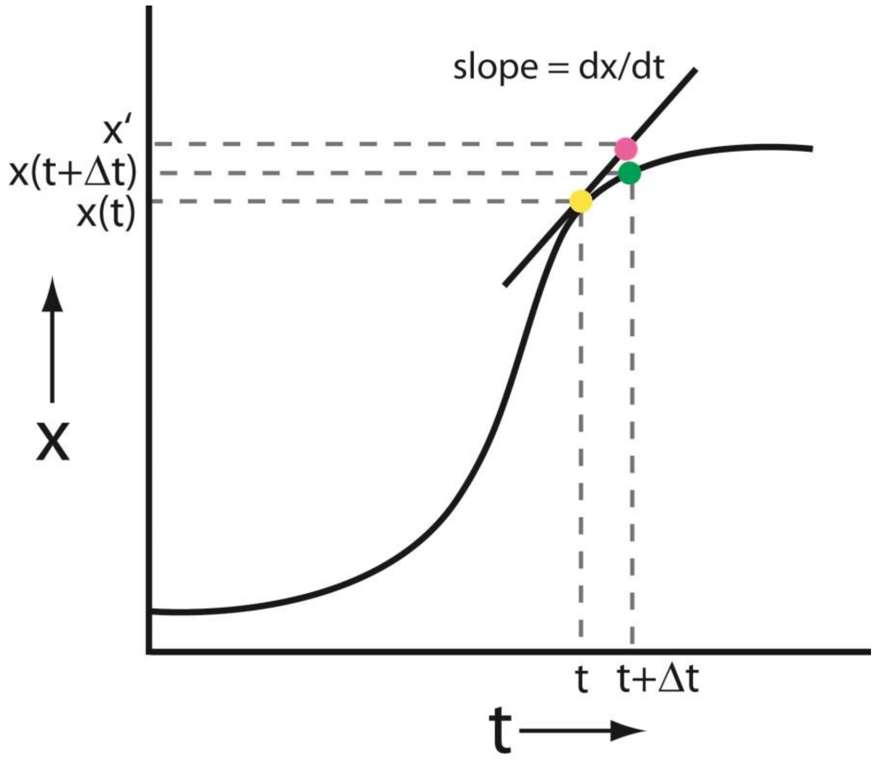
$$x(t+\Delta t) = x(t) + \Delta t . f(x,t)$$

In other words:

$$x(t+\Delta t) = x(t) + \Delta t .slope$$

Where, $slope \ = \frac{dx}{dt} = f(x,t)$

Can we have a better estimate of the slope?



Fourth-order Runge-Kutta method

The ODE $\dfrac{dx}{dt} = f(x,t)$

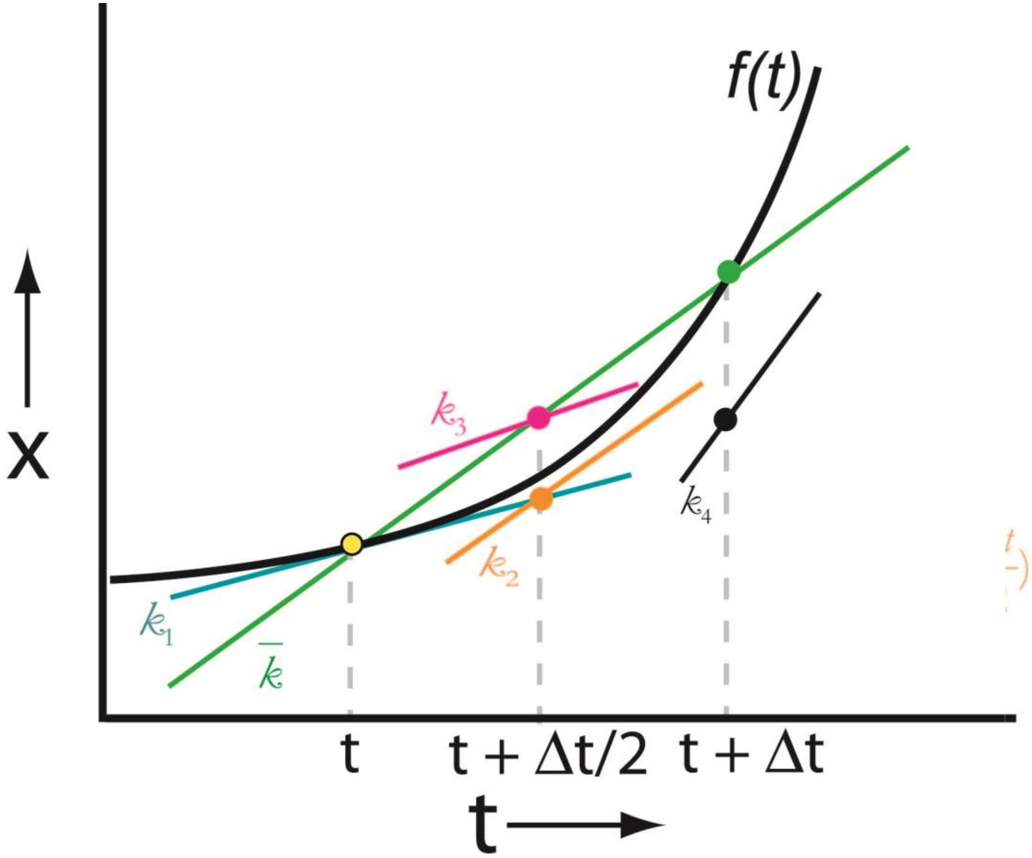
$$k_1 = \frac{dx}{dt} = f(x,t)$$

$$k_2 = \frac{dx}{dt} = f(x + k_1.\frac{\Delta t}{2}, t + \frac{\Delta t}{2})$$

$$k_3 = \frac{dx}{dt} = f(x + k_2.\frac{\Delta t}{2}, t + \frac{\Delta t}{2})$$

$$k_4 = \frac{dx}{dt} = f(x + k_3.\Delta t, t + \Delta t)$$

$$\bar{k} = \frac{1}{6}.(k_1 + 2.k_2 + 2k_3 + k_4)$$



Fourth-order Runge-Kutta method

The ODE $\frac{dx}{dt} = f(x,t)$

$$k_1 = \frac{dx}{dt} = f(x,t)$$

$$k_2 = \frac{dx}{dt} = f(x + k_1.\frac{\Delta t}{2}, t + \frac{\Delta t}{2})$$

$$k_3 = \frac{dx}{dt} = f(x + k_2.\frac{\Delta t}{2}, t + \frac{\Delta t}{2})$$

$$k_4 = \frac{dx}{dt} = f(x + k_3.\Delta t, t + \Delta t)$$

$$\bar{k} = \frac{1}{6}.(k_1 + 2.k_2 + 2k_3 + k_4)$$

$$x(t+\Delta t) = x(t) + \Delta t.slope$$

$$slope = \bar{k} = \frac{1}{6}(k_1 + 2.k_2 + 2k_3 + k_4)$$

$$x(t+\Delta t) = x(t) + \Delta t.\frac{1}{6}(k_1 + 2.k_2 + 2k_3 + k_4)$$

Application of Runge-Kutta method

$$\frac{dx}{dt} = r(1-x)x$$

$x_0 = 0.02$ $r = 0.5$ $\Delta t = 1$

T	X	k1	k2	k3	k4	k_mean	X(T+ delT)
0	0.02	0.0098	0.01214	0.012695	0.015813	0.012547	0.032547
1	0.032547	0.015744	0.019393	0.02023	0.024996	0.019997	0.052545
2	0.052545	0.024892	0.030383	0.031574	0.038521	0.031221	0.083766
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
20	0.997772	0.001112	0.000835	0.000904	0.000661	0.000875	0.998647

At t = 0 and x = x_0 = 0.02

$$k_1 = \frac{dx}{dt} = r(1-x)x = 0.5\times(1-0.02)\times 0.02 = 0.0098$$

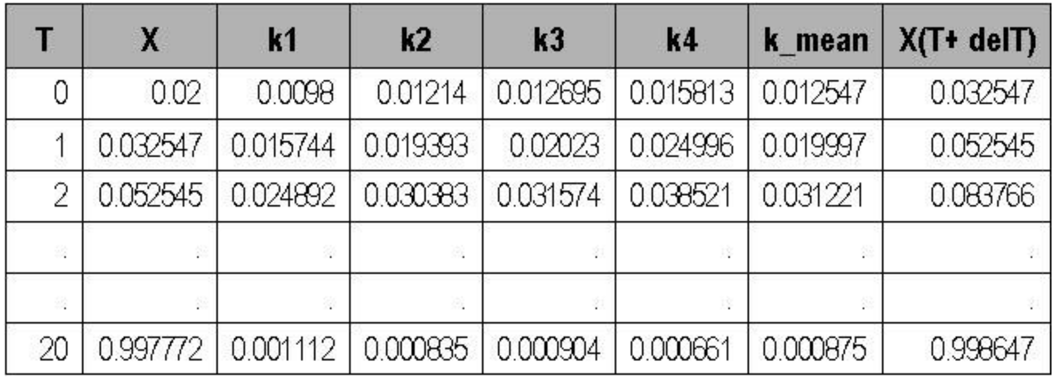
$$x(\frac{\Delta t}{2}) = x(\frac{1}{2}) = x(0) + k_1.\frac{\Delta t}{2} = 0.02 + 0.0098\times\frac{1}{2} = 0.0249$$

$$k_2 = \frac{dx}{dt} = r(1-x)x = 0.5\times(1-0.0249)\times 0.0249 = 0.01214$$

Application of Runge-Kutta method

$$\frac{dx}{dt} = r(1-x)x$$

$x_0 = 0.02$ $r = 0.5$ $\Delta t = 1$



T	X	k1	k2	k3	k4	k_mean	X(T+ delT)
0	0.02	0.0098	0.01214	0.012695	0.015813	0.012547	0.032547
1	0.032547	0.015744	0.019393	0.02023	0.024996	0.019997	0.052545
2	0.052545	0.024892	0.030383	0.031574	0.038521	0.031221	0.083766
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
20	0.997772	0.001112	0.000835	0.000904	0.000661	0.000875	0.998647

$$k_2 = \frac{dx}{dt} = r(1-x)x = 0.5\times(1-0.0249)\times 0.0249 = 0.01214$$

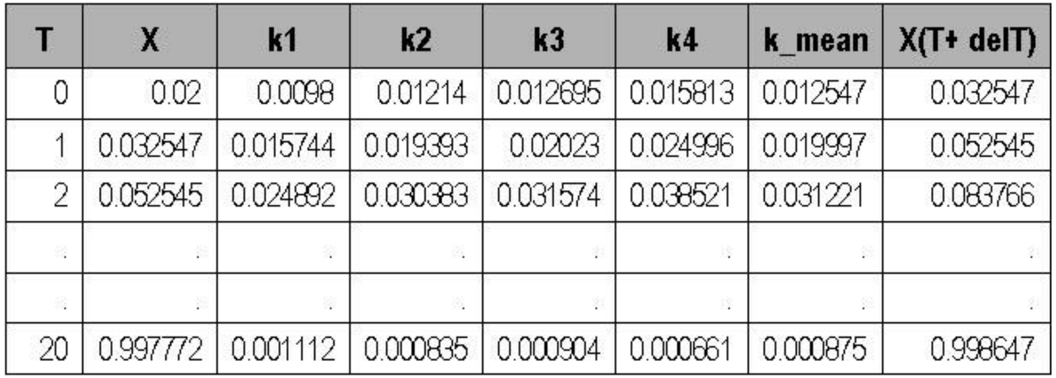
$$x(\frac{\Delta t}{2}) = x(\frac{1}{2}) = x(0) + k_2.\frac{\Delta t}{2} = 0.02 + 0.01214\times\frac{1}{2} = 0.02607$$

$$k_3 = \frac{dx}{dt} = r(1-x)x = 0.5\times(1-0.02607)\times 0.02607 = 0.012695$$

Application of Runge-Kutta method

$$\frac{dx}{dt} = r(1-x)x$$

$x_0 = 0.02$ $r = 0.5$ $\Delta t = 1$



T	X	k1	k2	k3	k4	k_mean	X(T+ delT)
0	0.02	0.0098	0.01214	0.012695	0.015813	0.012547	0.032547
1	0.032547	0.015744	0.019393	0.02023	0.024996	0.019997	0.052545
2	0.052545	0.024892	0.030383	0.031574	0.038521	0.031221	0.083766
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
20	0.997772	0.001112	0.000835	0.000904	0.000661	0.000875	0.998647

$$k_3 = \frac{dx}{dt} = r(1-x)x = 0.5\times(1-0.02607)\times 0.02607 = 0.012695$$

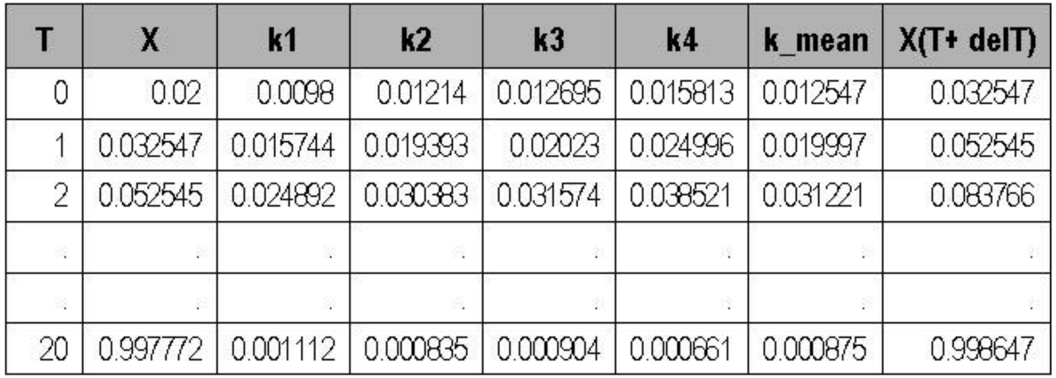
$$x(\Delta t) = x(1) = x(0) + k_3.\Delta t = 0.02 + 0.012695\times 1 = 0.032695$$

$$k_4 = \frac{dx}{dt} = r(1-x)x = 0.5\times(1-0.032695)\times 0.032695 = 0.015813$$

Application of Runge-Kutta method

$$\frac{dx}{dt} = r(1-x)x$$

$x_0 = 0.02$ $r = 0.5$ $\Delta t = 1$



T	X	k1	k2	k3	k4	k_mean	X(T+ delT)
0	0.02	0.0098	0.01214	0.012695	0.015813	0.012547	0.032547
1	0.032547	0.015744	0.019393	0.02023	0.024996	0.019997	0.052545
2	0.052545	0.024892	0.030383	0.031574	0.038521	0.031221	0.083766
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
20	0.997772	0.001112	0.000835	0.000904	0.000661	0.000875	0.998647

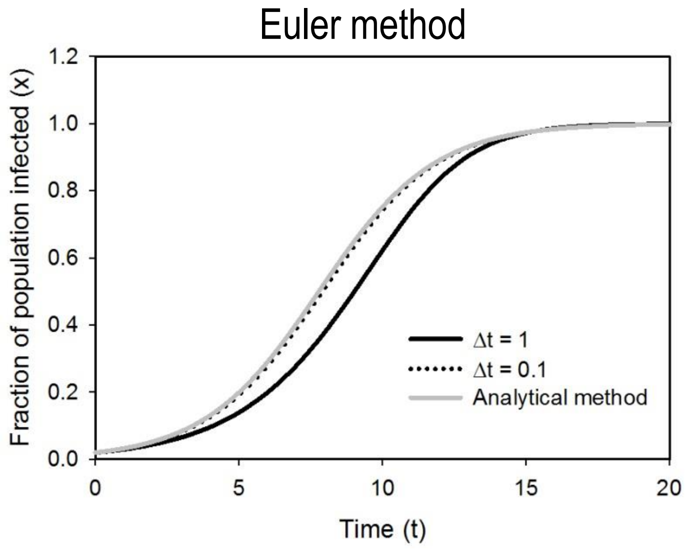
$$\bar{k} = \frac{1}{6}(k1 + 2k_2 + 2k_3 + k4) = \frac{1}{6} \times (0.098 + 2 \times 0.01214 + 2 \times 0.012695 + 0.015813) = 0.012547$$

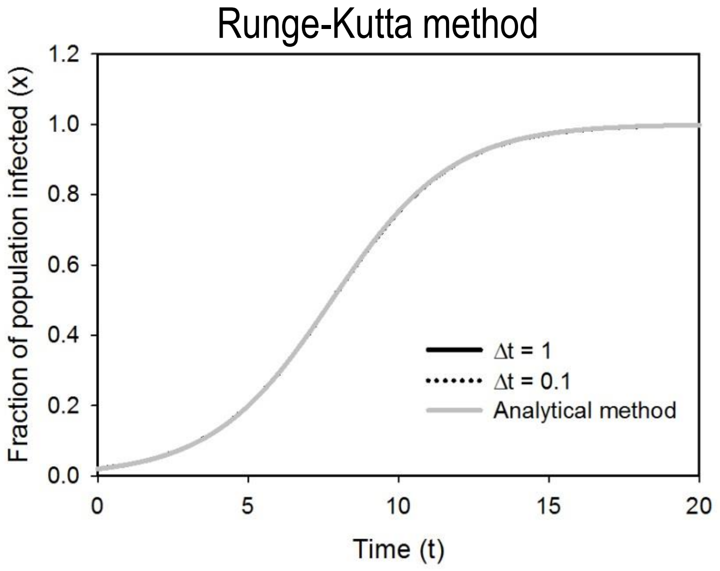
$$x(\Delta t) = x(1) = x(0) + \bar{k}.\Delta t = 0.02 + 0.012547 \times 1 = 0.032547$$

Runge-Kutta method is more accurate

$$\frac{dx}{dt} = r(1-x)x$$

$x_0 = 0.02$ $r = 0.5$



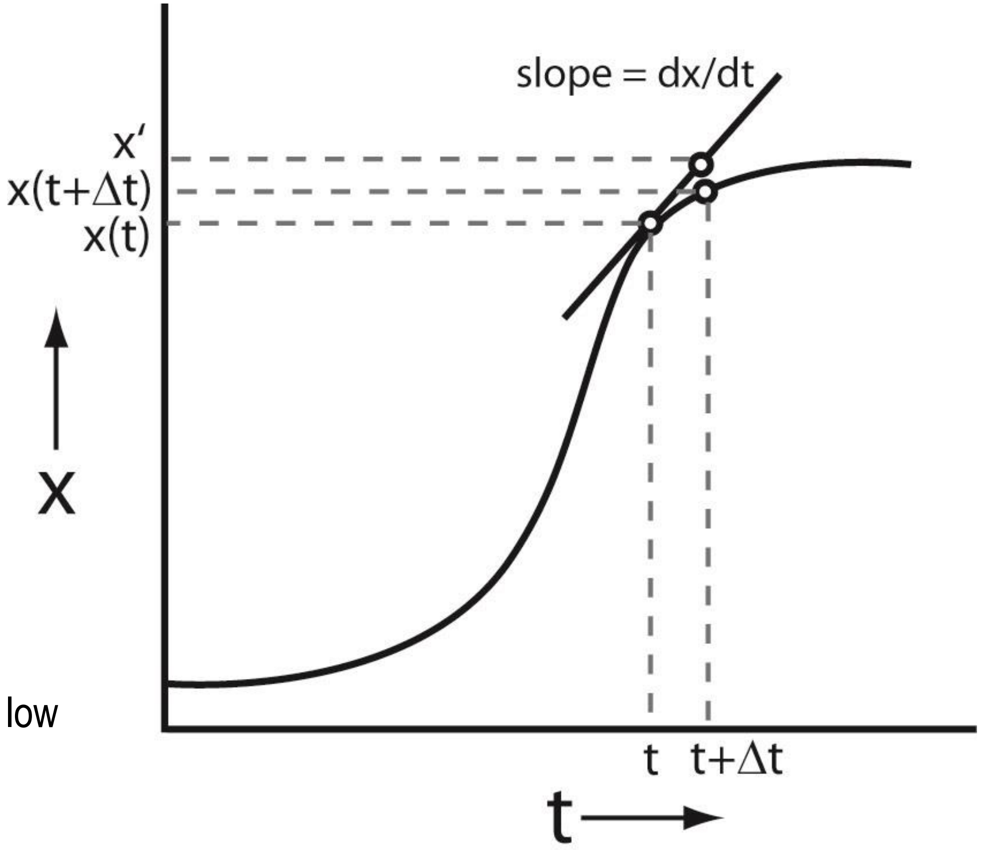


Adaptive time-step size

Issues with step size:

1. Smaller time step, Δt, gives smaller error; but increases time for computation.
2. Constant Δt throughout the integration does not help

Optimum solution: Δt varies at each step and keeps very low uniform error



Estimating error at each step

At each time point, use two good methods, one more accurate than the other. Calculate the difference in estimation by these two methods. That is local error.

The ODE, $\frac{dx}{dt} = f(x,t)$

1. Compute $x(t+\Delta t)$ using method 1. Call this x_1
2. Compute $x(t+\Delta t)$ using method 2 which is more accurate than method 1. Call this x_2
3. The local error, $\varepsilon = |x_2 - x_1|$

Change size of time-step based on local error

1. Compute $x(t+\Delta t)$ using method 1. Call this x_1

2. Compute $x(t+\Delta t)$ using method 2 which is more accurate than method 1. Call this x_2

3. The local error, $\varepsilon = \left| x_2 - x_1 \right|$

4. If $\varepsilon > (\text{tolerance value})_{\max}$, reject the calculation.

   Half the step size (i.e change Δt to Δt /2) and do the calculation again

5. Otherwise, take the result of the more accurate method (method 2).

   If $\varepsilon < (\text{tolerance value})_{\min}$

   Double the step size (i.e change Δt to 2.Δt) and do the calculations at next time point.

   Otherwise keep the same step size and do the calculations at next time point.

Key points:

1. Family of Runge-kutta algorithms are better for numerical solution of ODEs
2. These methods use average multiple slopes to increase accuracy in extrapolation
3. Most modern algorithms use adaptive time step size.
4. Adaptive time step size increases accuracy and decreases time of computation.
5. Majority of these algorithms are implemented in various programming languages and also available in plug-and-play software.