ITMO UNIVERSITY

**ITMO University**

# Numerical Integration of a Second-Order Linear ODE

**Course:** Simulation of Robotic Systems

**Author:** Mohammed Almaswary (517097)

**Date:** November 4, 2025

# Abstract

We study a damped second-order linear ordinary differential equation (ODE) with constant coefficients and constant forcing. An exact analytical solution is derived and compared against three numerical integrators implemented in the provided `Integrators.ipynb`: Explicit Euler, Implicit Euler, and classical Runge–Kutta of order 4 (RK4). Numerical results closely match the analytical solution; RK4 exhibits errors several orders of magnitude smaller than Euler methods for the chosen step size.


# 2. Problem Statement

We consider

$$a\,\ddot{x}(t) + b\,\dot{x}(t) + c\,x(t) = d, \qquad a = -1.14,\ b = -6.92,\ c = -7.82,\ d = -8.75. \tag{1}$$

The initial conditions are

$$x(0) = 0.1, \qquad \dot{x}(0) = 0.$$

The aim is to obtain an analytical solution of (1), and then approximate the same dynamics numerically using three different integrators and compare the results.

# 3. Analytical Solution

## 3.1 Normalized form

Dividing (1) by $a \neq 0$ gives

$$\ddot{x} + \alpha\,\dot{x} + \beta\,x = k, \quad \alpha = \frac{b}{a} \approx 6.0702,\ \beta = \frac{c}{a} \approx 6.8596,\ k = \frac{d}{a} \approx 7.6754.$$

## 3.2 Homogeneous solution

The homogeneous equation

$$\ddot{x} + \alpha\dot{x} + \beta x = 0$$

has characteristic equation

$$r^2 + \alpha r + \beta = 0 \ \Rightarrow\ r_{1,2} = \frac{-\alpha \pm \sqrt{\alpha^2 - 4\beta}}{2}.$$

Numerically,

$$r_1 \approx -4.5687, \qquad r_2 \approx -1.5014,$$

so the homogeneous solution is

$$x_h(t) = C_1 e^{r_1 t} + C_2 e^{r_2 t}.$$

## 3.3 Particular solution

For constant forcing $k$, a constant $x_p$ is sufficient:

$$x_p = \frac{k}{\beta} = \frac{d}{c} \approx \frac{-8.75}{-7.82} \approx 1.1189.$$

### 3.4 Apply initial conditions

The general solution is

$$x(t) = C_1 e^{r_1 t} + C_2 e^{r_2 t} + x_p.$$

Using $x(0) = 0.1$ and $\dot{x}(0) = 0$:

$$\begin{cases} C_1 + C_2 = 0.1 - x_p \approx -1.0189, \\ r_1 C_1 + r_2 C_2 = 0, \end{cases} \qquad \Rightarrow \quad C_1 \approx 0.4988, \;\; C_2 \approx -1.5177.$$

Hence

$$\boxed{x(t) \approx 0.4988\, e^{-4.5687t} - 1.5177\, e^{-1.5014t} + 1.1189},$$

and by differentiation

$$\boxed{\dot{x}(t) = 0.4988(-4.5687)e^{-4.5687t} - 1.5177(-1.5014)e^{-1.5014t}}.$$

# 4. Numerical Methods

## 4.1 First-order system

Define $y_1 = x$, $y_2 = \dot{x}$. Then

$$\dot{y}_1 = y_2, \qquad \dot{y}_2 = -\alpha y_2 - \beta y_1 + k = -6.0702\, y_2 - 6.8596\, y_1 + 7.6754.$$

In vector form: $\dot{\mathbf{y}} = f(t, \mathbf{y})$, where $\mathbf{y} = [y_1 \; y_2]^\top$.

## 4.2 Explicit Euler

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\, f(t_n, \mathbf{y}_n).$$

## 4.3 Implicit Euler

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\, f(t_{n+1}, \mathbf{y}_{n+1}),$$

which is solved in `Integrators.ipynb` by fixed-point iteration.

## 4.4 Classical RK4

$$\begin{aligned} k_1 &= f(t_n, \mathbf{y}_n), \\ k_2 &= f\big(t_n + \tfrac{h}{2}, \mathbf{y}_n + \tfrac{h}{2} k_1\big), \\ k_3 &= f\big(t_n + \tfrac{h}{2}, \mathbf{y}_n + \tfrac{h}{2} k_2\big), \\ k_4 &= f\big(t_n + h, \mathbf{y}_n + h k_3\big), \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + \frac{h}{6}\,(k_1 + 2k_2 + 2k_3 + k_4). \end{aligned}$$

# 5. Implementation and Parameters

The numerical experiments used the functions in `Integrators.ipynb`: `forward_euler`, `backward_euler`, and `runge_kutta4`. The dynamics were coded as

$$f(t, \mathbf{y}) = \begin{bmatrix} y_2 \\ -6.8596\, y_1 - 6.0702\, y_2 + 7.6754 \end{bmatrix}.$$

Simulation settings:

- Time interval: $t \in [0, 10]\,\mathrm{s}$.
- Step size: $h = 0.01\,\mathrm{s}$ (so $N = 1000$ steps).
- Initial state: $\mathbf{y}(0) = [0.1 \;\; 0]^\top$.

The analytical $x(t)$ and $\dot{x}(t)$ were evaluated on the same grid for comparison.

# 6. Results

Qualitatively, all three numerical methods reproduce the shape of the exact solution: the position $x(t)$ starts at 0.1, increases monotonically, and converges to the steady state $x_{\mathrm{ss}} = d/c \approx 1.1189$. The velocity $\dot{x}(t)$ starts at zero, exhibits a single peak, and decays smoothly back to zero as the system settles.

To quantify accuracy, the maximum absolute errors were computed as

$$\varepsilon_{\max}^{(x)} = \max_k |x_{\mathrm{num}}(t_k) - x_{\mathrm{an}}(t_k)|, \quad \varepsilon_{\max}^{(\dot{x})} = \max_k |\dot{x}_{\mathrm{num}}(t_k) - \dot{x}_{\mathrm{an}}(t_k)|.$$

| Method | $\max \lvert x - x_{\mathrm{an}} \rvert$ | $\max \lvert \dot{x} - \dot{x}_{\mathrm{an}} \rvert$ |
|---|---|---|
| Forward Euler | $3.300325 \times 10^{-3}$ | $1.566500 \times 10^{-2}$ |
| Backward Euler | $3.243652 \times 10^{-3}$ | $1.495076 \times 10^{-2}$ |
| RK4 | $6.766034 \times 10^{-9}$ | $3.138618 \times 10^{-8}$ |

# 7. Discussion

Figure 1 visualizes the analytical solution together with the numerical trajectories produced by the three integrators. The Explicit and Implicit Euler methods show small but noticeable differences around the peak of the velocity and in the phase portrait. The Forward Euler trajectory is slightly "outside", corresponding to a mild overshoot, whereas the Backward Euler trajectory is slightly more damped.

The RK4 solution lies almost exactly on top of the analytical curve in all three plots. This is consistent with the error table: Euler methods produce errors on the order of $10^{-3}$ to $10^{-2}$, while RK4 errors are on the order of $10^{-8}$, which is negligible for practical purposes at this time resolution.

The behaviour illustrates typical characteristics of first-order versus fourth-order schemes: Euler methods are easy to implement but relatively low accuracy, while RK4 requires more function evaluations per step but achieves much higher precision.

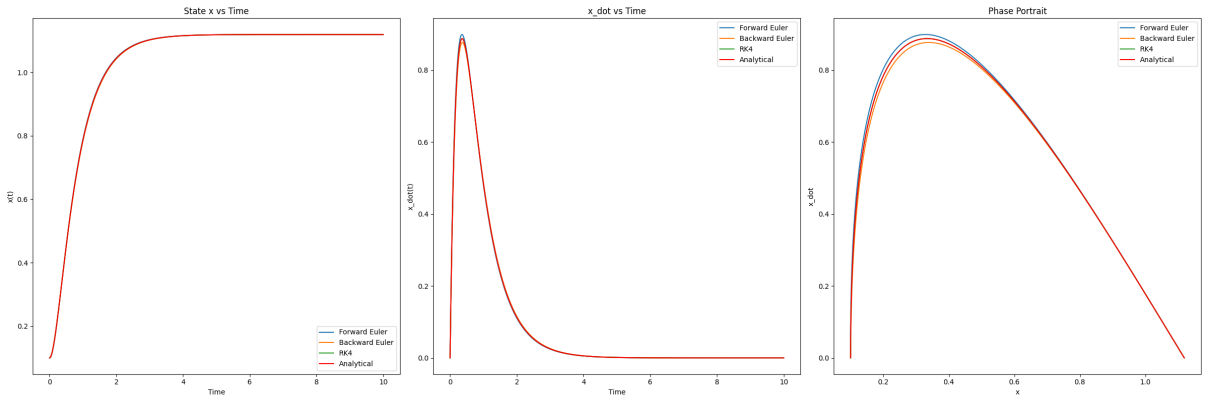


Figure 1: Analytical vs. numerical solutions: position (left), velocity (middle), and phase portrait (right).

# 8. Conclusion

In this work, a damped second-order linear ODE with constant coefficients and constant forcing was solved analytically and numerically. The analytical solution provided a reference for evaluating three time-integration schemes implemented in `Integrators.ipynb`: Explicit Euler, Implicit Euler, and RK4.

For the chosen step size $h = 0.01\,\mathrm{s}$, all methods were stable and correctly converged to the steady state. However, the accuracy differed significantly. Explicit and Implicit Euler methods produced maximum errors in the range of $10^{-3}$–$10^{-2}$, while the RK4 method achieved errors near $10^{-8}$ in both position and velocity, making its numerical trajectory visually indistinguishable from the exact solution.

Therefore, for this class of linear second-order systems, RK4 is strongly recommended when high accuracy is required, whereas Euler methods remain useful for quick, low-cost approximations or as building blocks in more complex schemes.