Computer Simulation of Systems/Circuits

CENG 215 Lecture Notes

1 States and State Variables (Quick Primer)

A *state* is the minimal set of variables whose values at time t_0 together with the input for $t \geq t_0$ determine the future of the system.

Energy-storing elements define natural state choices:

- Capacitor: $v_C(t)$ with stored energy $W_C = \frac{1}{2}Cv_C^2$,
- Inductor: $i_L(t)$ with stored energy $W_L = \frac{1}{2}Li_L^2$.

Hence in lumped RLC networks, a convenient state vector is $x = \begin{bmatrix} v_C \\ i_L \end{bmatrix}$ (for a single C and L).

General Series RLC (output v_C)

For the series R{L{C with source $v_s(t)$:

$$v_s = v_R + v_L + v_C = R\,i + L\,\dot{i} + v_C, \qquad i = C\,\dot{v}_C.$$

A *second-order* ODE for v_C follows:

$$LC\,\ddot{v}_C + RC\,\dot{v}_C + v_C = v_s(t). \tag{1}$$

A state-space form (one of many) is obtained by choosing $x_1 = v_C$ and $x_2 = i$:

$$\dot{x}_1 = \frac{1}{C}x_2, \qquad L\,\dot{x}_2 = -R\,x_2 - x_1 + v_s(t),$$

or in matrix form $\dot{x} = Ax + Bu$, $y = Cx$ with $u = v_s$, $y = v_C$.

2 Series RC as a First-Order Special Case

Setting $L = 0$ in (1) yields the familiar RC model:

$$RC\,\dot{v}_C + v_C = v_s(t) \iff \dot{v}_C = -\frac{1}{\tau}v_C + \frac{1}{\tau}v_s(t), \qquad \tau = RC. \tag{2}$$

Here the single state is $x(t) = v_C(t)$ and the state equation is

$$\dot{x}(t) = f\big(t, x(t), u(t)\big) = -\frac{1}{\tau}x(t) + \frac{1}{\tau}u(t), \quad u(t) = v_s(t).$$

3 Euler's (Forward) Method for Simulation

Given $\dot{x}(t) = f(t, x, u)$ and a time grid $t_k = t_0 + k\,h$ with step $h > 0$:

$$x_{k+1} = x_k + h\,f(t_k, x_k, u_k) \qquad \text{(Forward Euler)}. \tag{3}$$

Stability example (RC): for u bounded and $\dot{x} = -\frac{1}{\tau}x$ (homogeneous),

$$x_{k+1} = \big(1 - \tfrac{h}{\tau}\big)x_k \quad \Rightarrow \quad \text{stability requires } |1 - \tfrac{h}{\tau}| < 1 \iff 0 < h < 2\tau.$$

Smaller h improves accuracy and damping of numerical oscillations; for stiff or oscillatory second order RLC, consider smaller h or implicit/Runge{Kutta methods.

4 Driven RC: From Model to Simulation

We simulate (2) for three common inputs:

(i) Step: $u(t) = A\,u(t)$, (ii) Ramp: $u(t) = A\,t\,u(t)$, (iii) Sinusoid: $u(t) = A\sin(\omega t)$.

Analytic reference (closed-form solutions)

With $x(0) = x_0$:

$$\text{Step:} \quad x(t) = A + (x_0 - A)e^{-t/\tau}.$$
$$\text{Ramp:} \quad x(t) = A(t-\tau) + (x_0 + A\tau)\,e^{-t/\tau}.$$
$$\text{Sinusoid (steady state):} \quad x_{ss}(t) = \frac{A}{\sqrt{1+(\omega\tau)^2}}\sin\big(\omega t - \tan^{-1}(\omega\tau)\big).$$

4.1 Pseudo-code (Forward Euler)

```
Given R, C, tau=R*C, A, w, T_end, step h, initial x0
Define input u(t):
 STEP: u(t) = A
 RAMP: u(t) = A * t
 SINE: u(t) = A * sin(w * t)

Initialize:
t = 0
x = x0
Save (t, x)

Loop while t < T_end:
 u = input(t)
 f = -(1/tau) * x + (1/tau) * u
 x = x + h * f             # Forward Euler update
 t = t + h
 Save (t, x)

Optionally compute analytic x_ref(t) for comparison.
Plot u(t), x(t), (and x_ref(t) if computed).
```

4.2 Python code (NumPy + Matplotlib)

The script below simulates all three inputs, compares with analytic formulas, and saves figures:

- rc_step.png, rc_ramp.png, rc_sine.png

Listing 1: rc_sim.py Euler simulation of driven RC with step, ramp, and sinusoid

```python
import numpy as np
import matplotlib.pyplot as plt

# ---------- Parameters (edit freely) ----------
R = 1_000.0             # Ohms
C = 1e-4                # Farads
A = 5.0                 # Volts
w = 50.0                # rad/s (for sinusoid)
tau = R*C
x0 = 0.0                # initial capacitor voltage (V)
T_end = 5*tau           # total simulation time
h = 0.002*tau           # step size (keep 0 < h < 2*tau for Euler stability)
# ---------------------------------------------

# Time grid
```

```python
N = int(np.ceil(T_end/h)) + 1
t = np.linspace(0.0, N*h, N)
t = np.minimum(t, T_end)

def euler_sim(u_func):
        x = np.zeros_like(t)
        x[0] = x0
        for k in range(len(t)-1):
                u = u_func(t[k])
                f = -(1.0/tau)*x[k] + (1.0/tau)*u
                x[k+1] = x[k] + h*f
        return x

# Inputs
u_step  = lambda tt: A
u_ramp  = lambda tt: A*tt
u_sine  = lambda tt: A*np.sin(w*tt)

# Simulations
x_step  = euler_sim(u_step)
x_ramp  = euler_sim(u_ramp)
x_sine  = euler_sim(u_sine)

# Analytic references
x_step_ref = A + (x0 - A)*np.exp(-t/tau)
x_ramp_ref = A*(t - tau) + (x0 + A*tau)*np.exp(-t/tau)
# For sinusoid, plot steady state reference only:
x_sine_ss  = (A/np.sqrt(1+(w*tau)**2))*np.sin(w*t - np.arctan(w*tau))

# ----------- Plot helpers -----------
def plot_and_save(t, u, x_num, x_ref, title, fname, show_ref=True):
        plt.figure(figsize=(7.0,3.6))
        plt.plot(t, u, label='Input u(t)')
        plt.plot(t, x_num, label='Numerical x(t) (Euler)', linewidth=2)
        if show_ref and x_ref is not None:
                plt.plot(t, x_ref, '--', label='Analytic reference')
                plt.xlabel('t (s)')
                plt.ylabel('Voltage (V)')
                plt.grid(True, alpha=0.3)
                plt.title(title)
                plt.legend()
                plt.tight_layout()
                plt.savefig(fname, dpi=160)
        plt.close()

# Save figures
plot_and_save(t, A*np.ones_like(t), x_step, x_step_ref,
f'RC Step Response (tau={tau:.3e}s, h={h:.3e}s)', 'rc_step.png')

plot_and_save(t, A*t, x_ramp, x_ramp_ref,
f'RC Ramp Response (tau={tau:.3e}s, h={h:.3e}s)', 'rc_ramp.png')

plot_and_save(t, A*np.sin(w*t), x_sine, x_sine_ss,
f'RC Sinusoid Response (tau={tau:.3e}s, h={h:.3e}s)', 'rc_sine.png')
print("Saved: rc_step.png, rc_ramp.png, rc_sine.png")
```

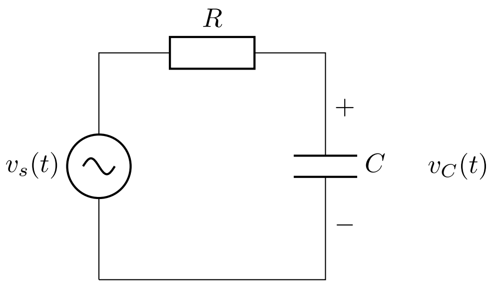

Figure 1: Series RC with capacitor voltage $v_C(t)$ as output.

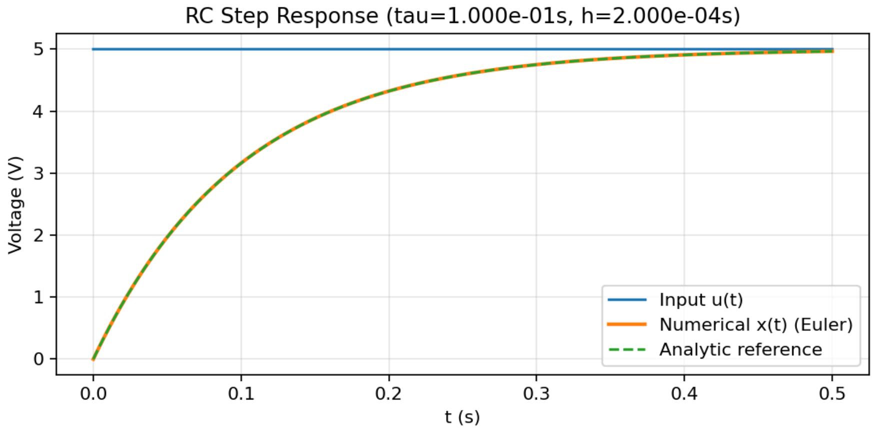

Figure 2: Euler vs Analytic: Step input.

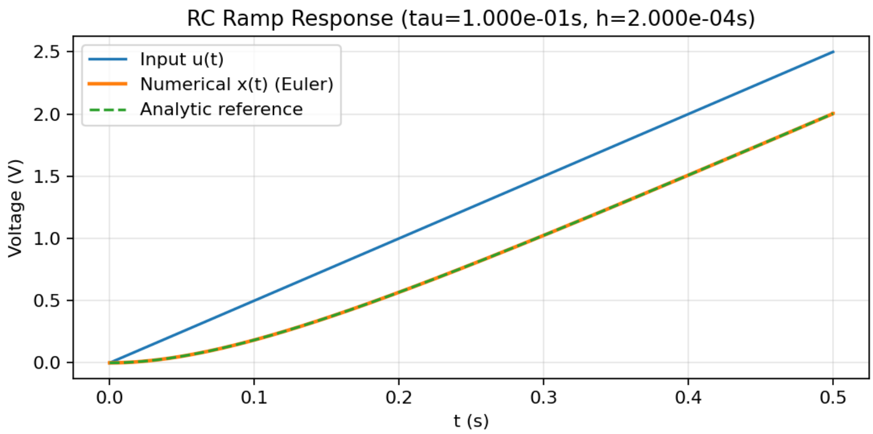

Figure 3: Euler vs Analytic: Ramp input.

4.3 Circuit schematic (for the driven RC)

5 Simulation Plots

Step, Ramp, and Sinusoid Results

6 Practical Guidance

- `Step size (`h`): For RC with` $\tau = RC$`, keep` $0 < h < 2\tau$ `for forward Euler stability; usually` $h \leq 0.05\tau$ `gives good accuracy.`
- `Inputs: If` $u(t)$ `is discontinuous (step), smaller` h `around the jump captures the corner more accurately.`

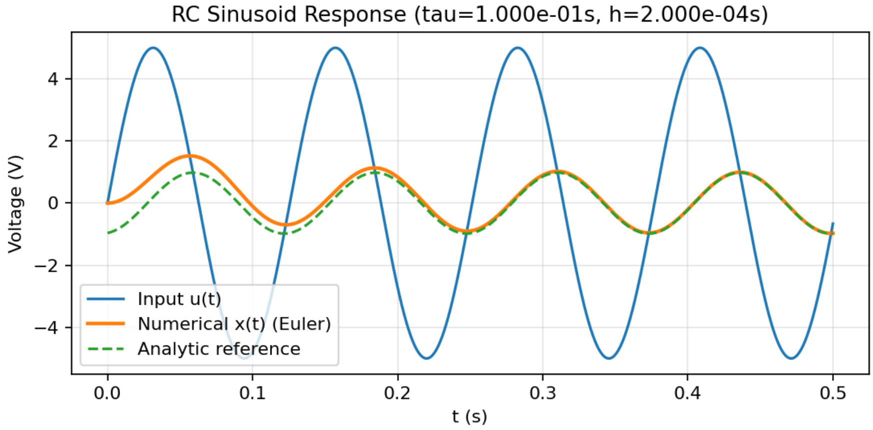

Figure 4: Euler vs Analytic (steady state reference shown): Sinusoid input.

- RLC systems: Oscillatory or stiff cases may require smaller h or higher-order/implicit schemes (e.g., RK4, Trapezoidal, BDF).
- Verification: Always compare with a known analytic solution (when available) to calibrate h and validate your simulator.

7 Nonlinear RC Simulation with a Quadratic i–v Device

In the previous sections, we considered a linear resistor in series with a capacitor and derived the state equation

$$\dot{x}(t) = -\frac{1}{\tau}x(t) + \frac{1}{\tau}v_s(t), \qquad x(t) = v_C(t).$$

We now replace the resistor by a nonlinear two--terminal device whose current--voltage characteristic is

$$i = 0.01\, v^2,$$

where v is the voltage across the nonlinear element. We keep the series connection with the capacitor, and the source $v_s(t)$ remains the input.

Nonlinear Device Model and State Equation

Let $v_C(t)$ be the capacitor voltage (state) and $i(t)$ the series current. The circuit is

$$v_s(t) = v_{\mathrm{NL}}(t) + v_C(t),$$

where v_{NL} is the voltage across the nonlinear device. The device law is

$$i(t) = 0.01\, v_{\mathrm{NL}}^2(t),$$

and the capacitor current relation is

$$i(t) = C\,\dot{v}_C(t).$$

Because the elements are in series, the same current flows through both, so

$$C\,\dot{v}_C(t) = 0.01\, v_{\mathrm{NL}}^2(t).$$

Using $v_{\mathrm{NL}}(t) = v_s(t) - v_C(t)$, we obtain the scalar nonlinear state equation

$$\dot{v}_C(t) = \frac{0.01}{C}\left(v_s(t) - v_C(t)\right)^2.$$

Defining $x(t) = v_C(t)$ gives

$$\boxed{\dot{x}(t) = f(t,x) = \frac{0.01}{C}\left(v_s(t) - x(t)\right)^2.}$$

For a step input $v_s(t) = A\,u(t)$, the state equation becomes

$$\dot{x}(t) = \frac{0.01}{C}\,\bigl(A - x(t)\bigr)^2, \qquad x(0) = x_0.$$

There is no simple closed{form exponential solution as in the linear RC case; instead we will integrate this nonlinear equation numerically.

Forward Euler Update for the Nonlinear RC

Given a time grid $t_k = kh$ with step size $h > 0$, and state samples $x_k \approx x(t_k)$, Forward Euler applied to

$$\dot{x}(t) = \frac{0.01}{C}\,\bigl(v_s(t) - x(t)\bigr)^2$$

yields

$$x_{k+1} = x_k + h\,\frac{0.01}{C}\,\bigl(v_s(t_k) - x_k\bigr)^2.$$

For a step input $v_s(t) = A\,u(t)$ (so $v_s(t_k) = A$ for $t_k > 0$),

$$\boxed{x_{k+1} = x_k + h\,\frac{0.01}{C}\,\bigl(A - x_k\bigr)^2.}$$

As in the linear case, h must be chosen sufficiently small to obtain a stable and accurate numerical solution. Because the right{hand side grows with $(A-x)^2$, too large a step size can cause numerical blow{up.

Python Simulation Code (Step Input)

The script below simulates the nonlinear RC circuit with step input $v_s(t) = A\,u(t)$ using Forward Euler, and plots both the input voltage and the capacitor voltage.

Listing 2: Euler simulation of nonlinear RC with quadratic i-v device

```
import numpy as np
import matplotlib.pyplot as plt

# ----- Parameters (edit freely) -----
C = 1e-4            # Farads
A = 5.0             # Step amplitude (Volts)
x0 = 0.0            # Initial capacitor voltage (V)
Tend = 0.3          # Total simulation time (s)
h = 1e-5            # Time step (s) - choose small for stability
# ------------------------------------

# Time grid
N = int(np.ceil(Tend / h)) + 1
t = np.linspace(0.0, Tend, N)

def vsource(tt):
        # Step input v_s(t) = A u(t)
        return A

def f(tval, xval):
        v = vsource(tval)
        return 0.01 / C * (v - xval)**2

# Forward Euler simulation
x = np.zeros_like(t)
x[0] = x0
for k in range(len(t) - 1):
        x[k+1] = x[k] + h * f(t[k], x[k])

# Build input waveform for plotting
u = np.array([vsource(tt) for tt in t])

# Plot results
```

```
        plt.figure(figsize=(7.0, 3.6))
        plt.plot(t, u, label="Input v_s(t)")
        plt.plot(t, x, label="Capacitor voltage v_C(t)", linewidth=2)
        plt.xlabel("t (s)")
        plt.ylabel("Voltage (V)")
        plt.grid(True, alpha=0.3)
        plt.title("Nonlinear RC: step response with i = 0.01 v^2")
        plt.legend()
        plt.tight_layout()
        plt.show()
```

Remarks

- The right{hand side is always nonnegative for a step with $A > 0$, so $x(t)$ is monotonically increasing as long as $x(t) < A$.
- Unlike the linear RC, the approach to the input level is not exponential; the dynamics slow down as $x(t) \to A$ because the current becomes small.
- The same code structure can be reused for other nonlinear devices by simply changing the function f(tval, xval) to implement the desired i{v relation.

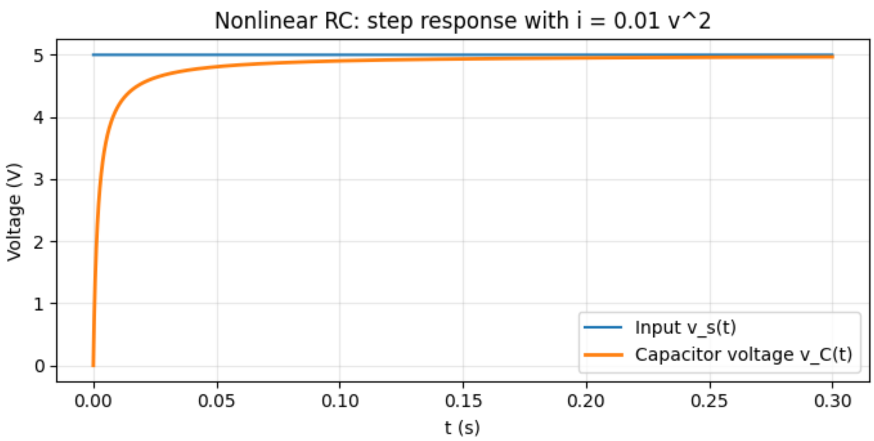

Figure 5: Simulation results: Non-linear device in series with capacitor