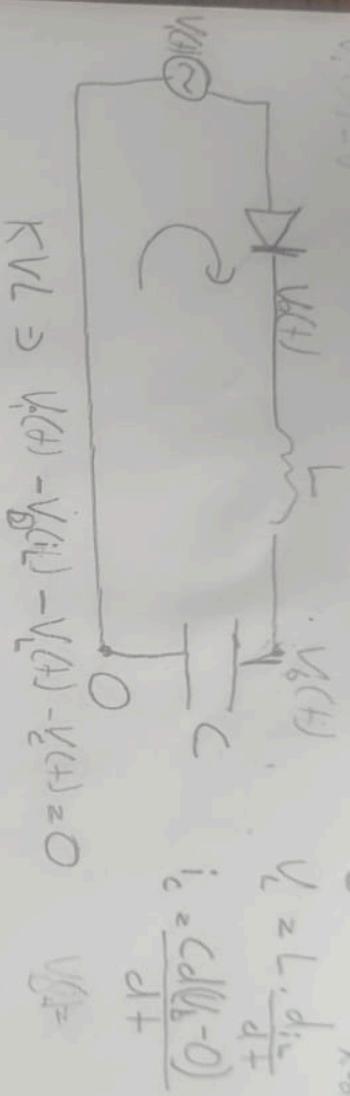


$x_1(t) = i_L(t)$ = current flowing through the inductor
 $x_2(t) = V_C(t)$ = the voltage across the capacitor

$$V_{DC} = V$$

$$V_2 \text{ Voltage drop from X-diode}$$



$$KVL \Rightarrow V_L(t) - V_C(t) - V_D(t) = 0$$

$$\frac{di_L}{dt} = \frac{1}{L} [V_L(t) - V_C(t) - V_D(t)] \Rightarrow \text{Inductor Current}$$

$$\frac{dV_b}{dt} = \frac{1}{C} i_L(t)$$

$$V_L(t) = \begin{cases} 0V, & 0 \leq t \leq 0.25 \\ 0.25V, & 0.25 \leq t \leq 0.5 \\ 0.25 + 0.05(t-0.5), & 0.5 \leq t \leq 1.5 \\ 1.5, & 1.5 \leq t \leq 2.0 \\ 1.5 + 0.05(t-1.5), & 2.0 \leq t \leq 2.25 \\ 2.0, & t \geq 2.25 \end{cases}$$

$$i_L(t) = \begin{cases} 0, & 0 \leq t \leq 0.25 \\ 0.25, & 0.25 \leq t \leq 0.5 \\ 0.25 + 0.05(t-0.5), & 0.5 \leq t \leq 1.5 \\ 1.5, & 1.5 \leq t \leq 2.0 \\ 1.5 + 0.05(t-1.5), & 2.0 \leq t \leq 2.25 \\ 2.0, & t \geq 2.25 \end{cases}$$

```

import numpy as np
import matplotlib.pyplot as plt

# =====
# CIRCUIT PARAMETERS AND INITIAL CONDITIONS (L AND C IS ASSUMED)
# =====
L = 10e-3      # Inductor value in Henries (10 mH)
C = 100e-6     # Capacitor value in Farads (100 μF)
Vi = 0.8        # Input voltage in Volts (constant step input)

# Initial conditions
iL0 = 0.0        # Initial inductor current in Amperes
vC0 = 0.0        # Initial capacitor voltage in Volts

# Simulation parameters
Tend = 0.5       # Total simulation time in seconds
h = 1e-6         # Time step in seconds (must be small for stability)

# =====
# X-DIODE MODEL (PIECEWISE LINEAR)
# =====
# From Figure 2, we use the following piecewise-linear approximation:
#
# vD(iD) = { 0                      if iD ≤ 0
#             { 0.25                  if 0 < iD ≤ 5 mA
#             { 0.25 + 0.035*(iD - 5)  if 5 < iD ≤ 15 mA
#             { 0.60 + 0.020*(iD - 15) if 15 < iD ≤ 20 mA
#             { 0.70 + 0.040*(iD - 20) if 20 < iD ≤ 25 mA
#
# where iD is in milliamperes and vD is in volts.
# This captures the non-uniform characteristic with flat and sloped
regions.

def v_diode(iD_amperes):
    """
    Returns diode voltage vD given diode current iD.
    Uses piecewise linear model based on the measured characteristic
from Fig. 2.

    Parameters:
        iD_amperes: diode current in Amperes

    Returns:
    """

```

```

    vD: diode voltage in Volts
"""

# Convert current from Amperes to milliamperes for the piecewise
model

iD_mA = iD_amperes * 1000.0

if iD_mA <= 0:
    # Diode is reverse biased or at zero current - acts as open
circuit
    return 0.0
elif iD_mA <= 5:
    # Flat region: vD = 0.25 V for 0 < iD ≤ 5 mA
    return 0.25
elif iD_mA <= 15:
    # Linear segment with slope 0.035 V/mA
    return 0.25 + 0.035 * (iD_mA - 5)
elif iD_mA <= 20:
    # Linear segment with slope 0.020 V/mA
    return 0.60 + 0.020 * (iD_mA - 15)
elif iD_mA <= 25:
    # Linear segment with slope 0.040 V/mA
    return 0.70 + 0.040 * (iD_mA - 20)
else:
    # Extrapolate beyond 25 mA using the last segment slope (0.040
V/mA)
    return 0.70 + 0.040 * (iD_mA - 20)

# =====
# STATE EQUATIONS DERIVATION
# =====
# Circuit topology: Vi(t) --- [X-diode] --- [L] --- [C] --- GND
#                      Va(t)           Vb(t)
#
# Define state variables:
#   x1 = iL(t) : inductor current (also the series current through all
elements)
#   x2 = vC(t) : capacitor voltage = Vb(t)
#
# Node voltage definitions:
#   Va(t) = voltage at the node between diode and inductor
#   Vb(t) = voltage at the node between inductor and capacitor = vC(t)
#
# KVL around the loop:

```

```

#     Vi(t) = vD + vL + vC
#     Vi(t) = vD + L*(diL/dt) + vC
#
# The diode voltage depends on the current through it:
#     vD = v_diode(iL) [from our piecewise model]
#
# Inductor voltage-current relation:
#     vL = L * (diL/dt)
# Therefore: diL/dt = vL/L = [Vi(t) - vD - vC] / L
#
# Capacitor current-voltage relation:
#     iC = C * (dvC/dt)
# Since iC = iL (series circuit): iL = C * (dvC/dt)
# Therefore: dvC/dt = iL / C
#
# STATE EQUATIONS:
#     dx1/dt = diL/dt = [Vi(t) - v_diode(iL) - vC] / L
#     dx2/dt = dvC/dt = iL / C
#
# IMPORTANT: When iL tries to go negative, the diode blocks (open
circuit)
# We must enforce iL >= 0 to model the diode's unidirectional
conduction.

def state_derivatives(t, x):
    """
    Computes the time derivatives of state variables.

    Parameters:
        t: current time (seconds)
        x: state vector [iL, vC]

    Returns:
        dxdt: derivative vector [diL/dt, dvC/dt]
    """
    iL = x[0] # Inductor current (state variable 1)
    vC = x[1] # Capacitor voltage (state variable 2)

    # Ensure inductor current is non-negative (diode constraint)
    if iL < 0:
        iL = 0.0

    # Get diode voltage for current iL

```

```

vD = v_diode(iL)

# Compute inductor voltage from KVL: vL = Vi - vD - vC
vL = Vi - vD - vC

# State derivatives
diL_dt = vL / L                      # From vL = L * diL/dt
dvC_dt = iL / C                       # From iC = C * dvC/dt and iC = iL

# If the inductor current wants to go negative, the diode blocks
# In this case, set diL/dt = 0 to prevent negative current
if iL <= 0 and diL_dt < 0:
    diL_dt = 0.0

return np.array([diL_dt, dvC_dt])

# =====
# NUMERICAL SIMULATION USING FORWARD EULER METHOD
# =====

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

# Initialize state arrays
iL = np.zeros(N)  # Inductor current over time
vC = np.zeros(N)  # Capacitor voltage over time
Va = np.zeros(N)  # Node voltage Va(t) = Vi - vD

# Set initial conditions
iL[0] = iL0
vC[0] = vC0
Va[0] = Vi - v_diode(iL[0])  # Va = Vi - vD at t=0

# Forward Euler integration loop
for k in range(N - 1):
    # Current state vector
    x_current = np.array([iL[k], vC[k]])

    # Compute derivatives at current time and state
    dxdt = state_derivatives(t[k], x_current)

    # Forward Euler update: x[k+1] = x[k] + h * f(t[k], x[k])
    x_next = x_current + h * dxdt

```

```

# Extract updated states
iL[k+1] = max(x_next[0], 0.0) # Enforce non-negative current
(diode constraint)
vC[k+1] = x_next[1]

# Compute node voltage Va = Vi - vD for plotting
# Va[k+1] = Vi - v_diode(iL[k+1]) This looked wrong to me

# --- CORRECT Va CALCULATION ---
if iL[k+1] > 0:
    # CASE 1: Diode is CONDUCTING (ON)
    # The node Va is determined by the source minus the diode drop.
    Va[k+1] = Vi - v_diode(iL[k+1])
else:
    # CASE 2: Diode is BLOCKING (OFF/Open Circuit)
    # The diode disconnects the source.
    # Since iL = 0 and is constant, voltage drop across inductor vL
= 0.
    # Therefore, Va must equal Vb (which is vC).
    Va[k+1] = vC[k+1]

# Node voltage Vb is same as capacitor voltage
Vb = vC

# =====
# RESULTS AND ANALYSIS
# =====
print("=" * 60)
print("X-DIODE-L-C CIRCUIT SIMULATION RESULTS")
print("=" * 60)
print(f"Circuit parameters:")
print(f"  L = {L*1e3:.2f} mH")
print(f"  C = {C*1e6:.2f} μF")
print(f"  Vi = {Vi:.2f} V (constant)")
print(f"\nSimulation parameters:")
print(f"  Time step h = {h*1e6:.2f} μs")
print(f"  Total time = {Tend:.3f} s")
print(f"  Number of points = {N}")
print(f"\nInitial conditions:")
print(f"  iL(0) = {iL0:.3f} A")
print(f"  vC(0) = {vC0:.3f} V")
print(f"\nFinal values (steady state):")

```

```

print(f" iL(final) = {iL[-1]*1e3:.6f} mA")
print(f" Vb(final) = vC(final) = {vC[-1]:.6f} V")
print(f" Va(final) = {Va[-1]:.6f} V")
print("=" * 60)

# =====
# PLOTTING
# =====

fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 8))

# Plot 1: Node voltage Va(t)
ax1.plot(t * 1000, Va, 'b-', linewidth=2, label='Va(t)')
ax1.axhline(y=Vi, color='r', linestyle='--', linewidth=1, label=f'Vi = {Vi} V')
ax1.set_xlabel('Time (ms)', fontsize=12)
ax1.set_ylabel('Voltage (V)', fontsize=12)
ax1.set_title('Node Voltage Va(t) vs Time', fontsize=14,
fontweight='bold')
ax1.grid(True, alpha=0.3)
ax1.legend(fontsize=10)

# Plot 2: Node voltage Vb(t) = Capacitor voltage vC(t)
ax2.plot(t * 1000, Vb, 'g-', linewidth=2, label='Vb(t) = vC(t)')
ax2.axhline(y=Vi, color='r', linestyle='--', linewidth=1, label=f'Vi = {Vi} V')
ax2.set_xlabel('Time (ms)', fontsize=12)
ax2.set_ylabel('Voltage (V)', fontsize=12)
ax2.set_title('Node Voltage Vb(t) = Capacitor Voltage vs Time',
fontsize=14, fontweight='bold')
ax2.grid(True, alpha=0.3)
ax2.legend(fontsize=10)

plt.tight_layout()
plt.savefig('xdiode_lc_simulation.png', dpi=150, bbox_inches='tight')
plt.show()

# Additional plot: Inductor current
fig2, ax3 = plt.subplots(figsize=(10, 4))
ax3.plot(t * 1000, iL * 1000, 'r-', linewidth=2, label='iL(t)')
ax3.set_xlabel('Time (ms)', fontsize=12)
ax3.set_ylabel('Current (mA)', fontsize=12)
ax3.set_title('Inductor Current iL(t) vs Time', fontsize=14,
fontweight='bold')

```

```

ax3.grid(True, alpha=0.3)
ax3.legend(fontsize=10)
plt.tight_layout()
plt.savefig('xdiode_lc_current.png', dpi=150, bbox_inches='tight')
plt.show()

print("\nPlots saved as 'xdiode_lc_simulation.png' and
'xdiode_lc_current.png'")

```

```

File Edit Selection View Go Run Terminal Help CENG215_PROJECT1 project18.py state_derivatives
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 # ****
5 # CIRCUIT PARAMETERS AND INITIAL CONDITIONS (L AND C IS ASSUMED)
6 #
7 L = 10e-3 # Inductor value in Henries (10 mH)
8 C = 100e-6 # Capacitor value in Farads (100 μF)
9 Vi = 0.8 # Input voltage in Volts (constant step input)
10
11 # Initial conditions
12 id = 0.0 # Initial inductor current in Amperes
13 vC0 = 0.0 # Initial capacitor voltage in Volts
14
15 # Simulation parameters
16 Tend = 0.5 # Total simulation time in seconds
17 h = 1e-6 # Time step in seconds (must be small for stability)
18
19 # ****
20 # X-DIODE MODEL (PIECEWISE LINEAR)
21 #
22 # From Figure 2, we use the following piecewise-linear approximation:
23 #
24 # vD(id) = { 0 if id <= 0
25 #             0.25 if 0 < id < 5 mA
26 #             { 0.25 + 0.035*(id - 5) if 5 < id < 15 mA
27 #             { 0.60 + 0.020*(id - 15) if 15 < id < 20 mA
28 #             { 0.70 + 0.040*(id - 20) if 20 < id < 25 mA
29 #
30 # where id is in milliamperes and vD is in volts.
31 # This captures the non-uniform characteristic with flat and sloped regions.
32
33 def v_diode(ID_amperes):
34     """
35     Returns diode voltage vD given diode current id.
36     Uses piecewise linear model based on the measured characteristic from Fig. 2.
37
38     Parameters:
39         id_amperes: diode current in Amperes
40
41     Returns:
42         vD: diode voltage in Volts
43
44     """
45     id_mA = ID_amperes * 1000.0
46
47     if id_mA <= 0:
48         # Diode is reverse biased or at zero current - acts as open circuit
49         return 0.0
50     elif id_mA < 5:
51         # Flat region: vD = 0.25 V for 0 < id ≤ 5 mA
52         return 0.25
53     elif id_mA < 15:
54         # Linear segment with slope 0.035 V/mA
55         return 0.25 + 0.035 * (id_mA - 5)
56     elif id_mA < 20:
57         # Linear segment with slope 0.020 V/mA
58         return 0.60 + 0.020 * (id_mA - 15)
59     elif id_mA < 25:
60         # Linear segment with slope 0.040 V/mA
61         return 0.70 + 0.040 * (id_mA - 20)
62     else:
63         # Extrapolate beyond 25 mA using the last segment slope (0.040 V/mA)
64         return 0.70 + 0.040 * (id_mA - 20)
65
66 # ****
67 # STATE EQUATIONS DERIVATION
68 #
69 # Circuit topology: Vi(t) --- (X-diode) --- [L] --- [C] --- GND
70 #                   |                         Va(t)                         Vb(t)
71 #
72 # Define state variables:
73 #   x1 = il(t) : inductor current (also the series current through all elements)
74 #   x2 = vc(t) : capacitor voltage = vC(t)
75 #
76 # Node voltage definitions:
77 #   Va(t) = voltage at the node between diode and inductor
78 #   Vb(t) = voltage at the node between inductor and capacitor = vC(t)
79 #
80 # KVL around the loop:
81 #   Vi(t) = vD + VL + VC
82 #   Vi(t) = vD + L*(di/dt) + VC

```

```

File Edit Selection View Go Run Terminal Help CENG215_PROJECT1 project18.py state_derivatives
33 def v_diode(ID_amperes):
34     """
35     Returns diode voltage vD given diode current id.
36     Uses piecewise linear model based on the measured characteristic from Fig. 2.
37
38     Parameters:
39         id_amperes: diode current in Amperes
40
41     Returns:
42         vD: diode voltage in Volts
43
44     """
45     id_mA = ID_amperes * 1000.0
46
47     if id_mA <= 0:
48         # Diode is reverse biased or at zero current - acts as open circuit
49         return 0.0
50     elif id_mA < 5:
51         # Flat region: vD = 0.25 V for 0 < id ≤ 5 mA
52         return 0.25
53     elif id_mA < 15:
54         # Linear segment with slope 0.035 V/mA
55         return 0.25 + 0.035 * (id_mA - 5)
56     elif id_mA < 20:
57         # Linear segment with slope 0.020 V/mA
58         return 0.60 + 0.020 * (id_mA - 15)
59     elif id_mA < 25:
60         # Linear segment with slope 0.040 V/mA
61         return 0.70 + 0.040 * (id_mA - 20)
62     else:
63         # Extrapolate beyond 25 mA using the last segment slope (0.040 V/mA)
64         return 0.70 + 0.040 * (id_mA - 20)
65
66 # ****
67 # STATE EQUATIONS DERIVATION
68 #
69 # Circuit topology: Vi(t) --- (X-diode) --- [L] --- [C] --- GND
70 #                   |                         Va(t)                         Vb(t)
71 #
72 # Define state variables:
73 #   x1 = il(t) : inductor current (also the series current through all elements)
74 #   x2 = vc(t) : capacitor voltage = vC(t)
75 #
76 # Node voltage definitions:
77 #   Va(t) = voltage at the node between diode and inductor
78 #   Vb(t) = voltage at the node between inductor and capacitor = vC(t)
79 #
80 # KVL around the loop:
81 #   Vi(t) = vD + VL + VC
82 #   Vi(t) = vD + L*(di/dt) + VC

```

```
project18.py x
project18.py > state_derivatives

83     #
84     # The diode voltage depends on the current through it:
85     # vD = v_diode(iL) [from our piecewise model]
86     #
87     # Inductor voltage-current relation:
88     # VL = L * (diL/dt)
89     # Therefore: diL/dt = VL/L = [Vi(t) - vD - vC] / L
90     #
91     # Capacitor current-voltage relation:
92     # iC = C * (dvC/dt)
93     # Since iC = iL (series circuit): iL = C * (dvC/dt)
94     # Therefore: dvC/dt = iL / C
95     #
96     # STATE EQUATIONS:
97     # dx1/dt = diL/dt = [Vi(t) - vD - vC] / L
98     # dx2/dt = dvC/dt = iL / C
99     #
100    # IMPORTANT: When iL tries to go negative, the diode blocks (open circuit).
101    # We must enforce iL >= 0 to model the diode's unidirectional conduction.
102
103    def state_derivatives(t, x):
104        """
105            Computes the time derivatives of state variables.
106
107            Parameters:
108                t: current time (seconds)
109                x: state vector [iL, vC]
110
111            Returns:
112            dxdt: derivative vector [diL/dt, dvC/dt]
113
114            iL = x[0] # Inductor current (state variable 1)
115            vC = x[1] # Capacitor voltage (state variable 2)
116
117            # Ensure inductor current is non-negative (diode constraint)
118            if iL < 0:
119                iL = 0.0
120
121            # Get diode voltage for current iL
122            vD = v_diode(iL)
123
124            # Compute inductor voltage from KVL: VL = Vi - vD - vC
```

Line 111, Col 13 Spaces: 4 UTF-8 CR/LF [] Python 3.14.1(ceng215-project1) 1053 10.12.2025

```
project18.py x
project18.py > state_derivatives
163    def state_derivatives(t, x):
164        vL = Vi - vD - vC
165
166        # State derivatives
167        diL_dt = vL / L           # From VL = L * diL/dt
168        dvC_dt = iL / C          # From iC = C * dvC/dt and iC = iL
169
170        # If the inductor current wants to go negative, the diode blocks
171        # In this case, set diL_dt = 0 to prevent negative current
172        if iL <= 0 and diL_dt < 0:
173            diL_dt = 0.0
174
175        return np.array([diL_dt, dvC_dt])
176
177    # **** NUMERICAL SIMULATION USING FORWARD EULER METHOD ****
178    # ****
179    # Create time grid
180    N = int(np.ceil(Tend / h)) + 1
181    t = np.linspace(0.0, Tend, N)
182
183    # Initialize state arrays
184    iL = np.zeros(N) # Inductor current over time
185    vC = np.zeros(N) # Capacitor voltage over time
186    Va = np.zeros(N) # Node voltage Va(t) = Vi - vD
187
188    # Set initial conditions
189    iL[0] = iL0
190    vC[0] = vC0
191    Va[0] = Vi - v_diode(iL[0]) # Va = Vi - vD at t=0
192
193    # Forward Euler integration loop
194    for k in range(N - 1):
195        # Current state vector
196        x_current = np.array([iL[k], vC[k]])
197
198        # Compute derivatives at current time and state
199        dxdt = state_derivatives(t[k], x_current)
200
201        # Forward Euler update: x[k+1] = x[k] + h * f(t[k], x[k])
202        x_next = x_current + h * dxdt
203
204        iL[k + 1] = x_next[0]
205        vC[k + 1] = x_next[1]
206        Va[k + 1] = Vi - v_diode(iL[k + 1])
```

Line 111, Col 13 Spaces: 4 UTF-8 CR/LF [] Python 3.14.1(ceng215-project1) 1054 10.12.2025

```

File Edit Selection View Go Run Terminal Help < - > Q ceng215_project1
project18.py > state_derivatives
165     # Extract updated states
166     il[k+1] = max(x.next[0], 0.0) # Enforce non-negative current (diode constraint)
167     vc[k+1] = x.next[1]
168
169     # Compute node voltage Va = Vi - v0 for plotting
170     # Va[k+1] = Vi - v_diode(il[k+1]) This looked wrong to me
171
172     # --- CORRECT Va CALCULATION ---
173     if il[k+1] > 0:
174         # CASE 1: Diode is CONDUCTING (ON)
175         # The node Va is determined by the source minus the diode drop.
176         Va[k+1] = Vi - v_diode(il[k+1])
177     else:
178         # CASE 2: Diode is BLOCKING (OFF/Open Circuit)
179         # The diode disconnects the source.
180         # Since il = 0 and is constant, voltage drop across inductor VL = 0.
181         # Therefore, Va must equal Vb (which is vc).
182         Va[k+1] = vc[k+1]
183
184     # Node voltage Vb is same as capacitor voltage
185     Vb = vc
186
187     # *****
188     # RESULTS AND ANALYSIS
189     # *****
190
191     print("=" * 60)
192     print("XDIODE-L-C CIRCUIT SIMULATION RESULTS")
193     print("=" * 60)
194     print("Circuit parameters:")
195     print(f" L = {L*1e3:.2f} μF")
196     print(f" C = {C*1e6:.2f} μF")
197     print(f" Vi = {Vi:.2f} V (constant)")
198     print("\nSimulation parameters:")
199     print(f" Time step h = {h*1e6:.2f} μs")
200     print(f" Total time = {Tend:.3f} s")
201     print(f" Number of points = {N}")
202     print("\nInitial conditions:")
203     print(f" il(0) = {il0:.3f} A")
204     print(f" vc(0) = {vc0:.3f} V")
205     print(f" Final values (steady state):")
206     print(f" il(final) = {il[-1]*1e3:.6f} mA")

```

In 111, Col 13 Spaces:4 UTF-8 CR/LF [] Python 3.14.1 (ceng215-project1) 1054 10.12.2025

```

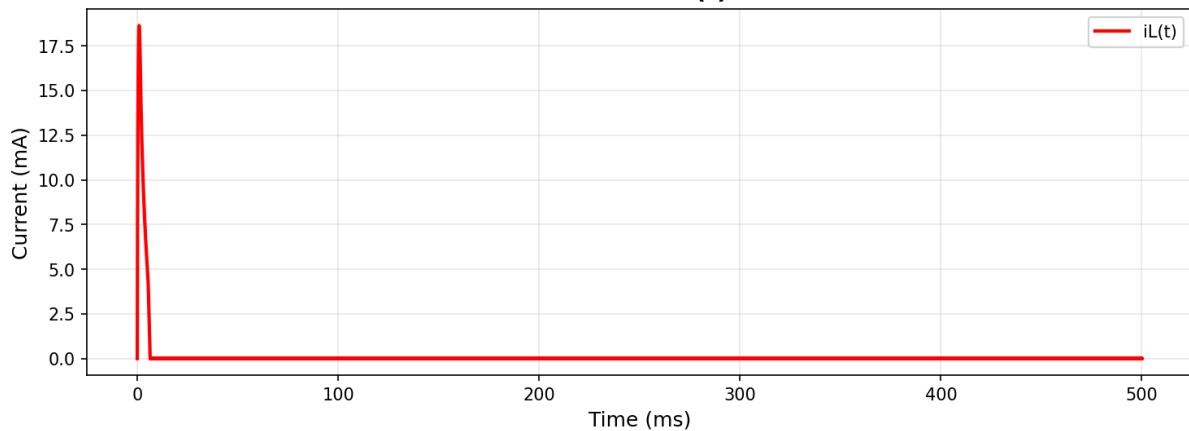
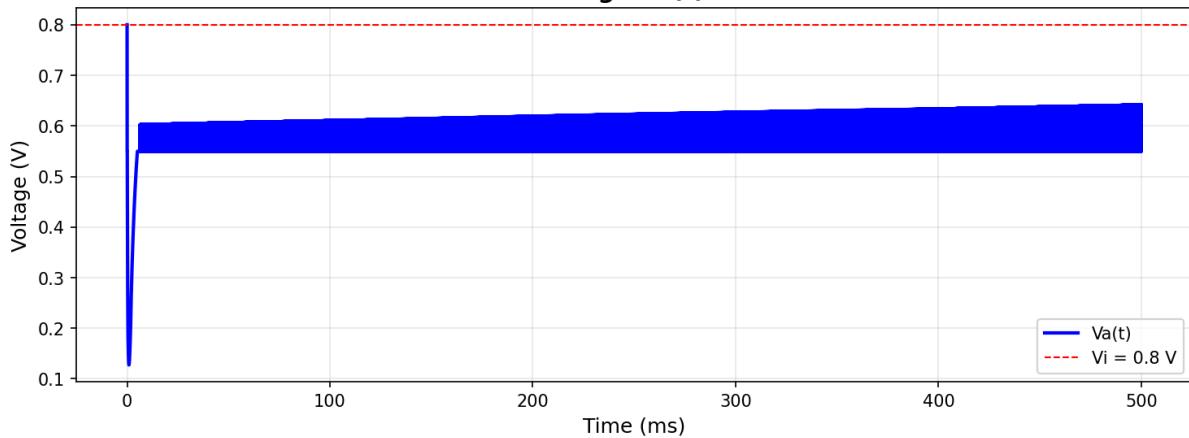
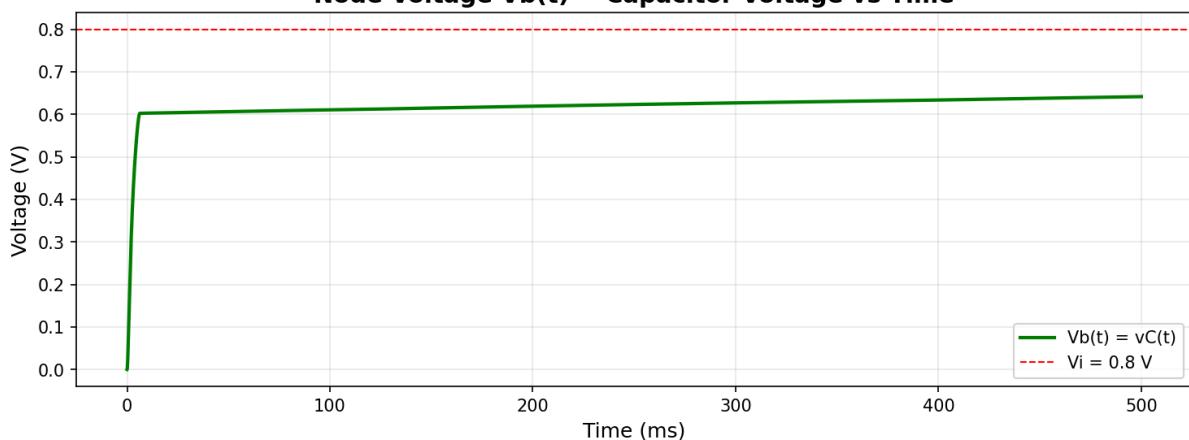
File Edit Selection View Go Run Terminal Help < - > Q ceng215_project1
project18.py > state_derivatives
206     print(f" il(final) = {il[-1]*1e3:.6f} mA")
207     print(f" Vb(final) = {vc[-1]:.6f} V")
208     print(f" Va(final) = {Va[-1]:.6f} V")
209     print("=" * 60)
210
211     # *****
212     # PLOTTING
213     # *****
214     fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 8))
215
216     # Plot 1: Node voltage Va(t)
217     ax1.plot(t * 1000, Va, 'b-', linewidth=2, label='Va(t)')
218     ax1.axhline(Va0, color='r', linestyle='--', linewidth=1, label=f'Vi = {Vi} V')
219     ax1.set_xlabel('Time (ms)', fontsize=12)
220     ax1.set_ylabel('Voltage (V)', fontsize=12)
221     ax1.set_title('Node Voltage Va(t) vs Time', fontsize=14, fontweight='bold')
222     ax1.grid(True, alpha=0.3)
223     ax1.legend(fontsize=10)
224
225     # Plot 2: Node voltage Vb(t) = Capacitor voltage vc(t)
226     ax2.plot(t * 1000, Vb, 'g-', linewidth=2, label='Vb(t) = vc(t)')
227     ax2.axhline(Vb0, color='r', linestyle='--', linewidth=1, label=f'Vi = {Vi} V')
228     ax2.set_xlabel('Time (ms)', fontsize=12)
229     ax2.set_ylabel('Voltage (V)', fontsize=12)
230     ax2.set_title('Node Voltage Vb(t) = Capacitor Voltage vs Time', fontsize=14, fontweight='bold')
231     ax2.grid(True, alpha=0.3)
232     ax2.legend(fontsize=10)
233
234     plt.tight_layout()
235     plt.savefig('xdiode_lc_simulation.png', dpi=150, bbox_inches='tight')
236     plt.show()
237
238     # Additional plot: Inductor current
239     fig2, ax3 = plt.subplots(figsize=(10, 4))
240     ax3.plot(t * 1000, il * 1000, 'r-', linewidth=2, label='il(t)')
241     ax3.set_xlabel('Time (ms)', fontsize=12)
242     ax3.set_ylabel('Current (mA)', fontsize=12)
243     ax3.set_title('Inductor Current il(t) vs Time', fontsize=14, fontweight='bold')
244     ax3.grid(True, alpha=0.3)
245     ax3.legend(fontsize=10)
246     plt.tight_layout()
247     plt.savefig('xdiode_lc_current.png', dpi=150, bbox_inches='tight')


```

In 111, Col 13 Spaces:4 UTF-8 CR/LF [] Python 3.14.1 (ceng215-project1) 1054 10.12.2025

```
File Edit Selection View Go Run Terminal Help
project18.py U
> CENG215 PROJECT1
> OUTLINE
> TIMELINE
EXPLORER
project18.py > state derivatives
210
211 # =====
212 # PLOTTING
213 # =====
214 fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 8))
215
216 # Plot 1: Node voltage V_a(t)
217 ax1.plot(t * 1000, Va, 'b', linewidth=2, label='Va(t)')
218 ax1.axhline(y=Vi, color='r', linestyle='--', linewidth=1, label=f'Vi = {Vi} V')
219 ax1.set_xlabel('Time (ms)', fontsize=12)
220 ax1.set_ylabel('Voltage (V)', fontsize=12)
221 ax1.set_title('Node Voltage Va(t) vs Time', fontsize=14, fontweight='bold')
222 ax1.grid(True, alpha=0.3)
223 ax1.legend(fontsize=10)
224
225 # Plot 2: Node voltage V_b(t) = Capacitor voltage vC(t)
226 ax2.plot(t * 1000, Vb, 'g', linewidth=2, label='Vb(t) = vC(t)')
227 ax2.axhline(y=Vi, color='r', linestyle='--', linewidth=1, label=f'Vi = {Vi} V')
228 ax2.set_xlabel('Time (ms)', fontsize=12)
229 ax2.set_ylabel('Voltage (V)', fontsize=12)
230 ax2.set_title('Node Voltage Vb(t) = Capacitor Voltage vs Time', fontsize=14, fontweight='bold')
231 ax2.grid(True, alpha=0.3)
232 ax2.legend(fontsize=10)
233
234 plt.tight_layout()
235 plt.savefig('xdiode_lc_simulation.png', dpi=150, bbox_inches='tight')
236 plt.show()
237
238 # Additional plot: Inductor current
239 fig2, ax3 = plt.subplots(figsize=(10, 4))
240 ax3.plot(t * 1000, iL * 1000, 'r', linewidth=2, label='iL(t)')
241 ax3.set_xlabel('Time (ms)', fontsize=12)
242 ax3.set_ylabel('Current (mA)', fontsize=12)
243 ax3.set_title('Inductor Current iL(t) vs Time', fontsize=14, fontweight='bold')
244 ax3.grid(True, alpha=0.3)
245 ax3.legend(fontsize=10)
246 plt.tight_layout()
247 plt.savefig('xdiode_lc_current.png', dpi=150, bbox_inches='tight')
248 plt.show()
249
250 print("\nPlots saved as 'xdiode_lc_simulation.png' and 'xdiode_lc_current.png'")
```

In 111, Col 13 Spaces:4 UTF-8 CR/LF {} Python 3.14.1(ceng215-project1) 1054 10.12.2025

Inductor Current $i_L(t)$ vs Time**Node Voltage $V_a(t)$ vs Time****Node Voltage $V_b(t) = v_C(t)$ vs Time**

```
=====
X-DIODE-L-C CIRCUIT SIMULATION RESULTS
=====
Circuit parameters:
L = 10.00 mH
C = 100.00 µF
Vi = 0.80 V (constant)

Simulation parameters:
Time step h = 1.00 µs
Total time = 0.500 s
Number of points = 500001

Initial conditions:
iL(0) = 0.000 A
vC(0) = 0.000 V

Final values (steady state):
iL(final) = 0.000000 mA
Vb(final) = vC(final) = 0.641602 V
Va(final) = 0.641602 V
=====

Plots saved as 'xdiode_lc_simulation.png' and 'xdiode_lc_current.png'
```