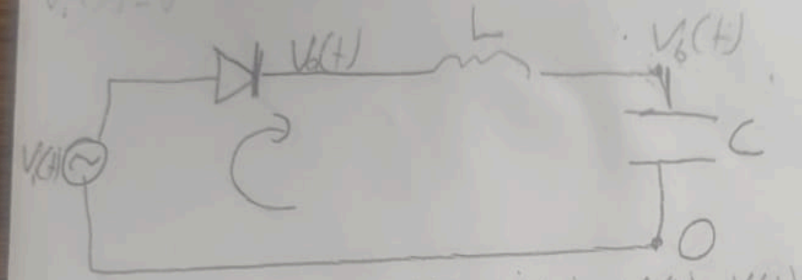
$x_1(t) = i_L(t)$ = Current flowing through the inductor

$x_2(t) = V_b(t)$ = the Voltage across the capacitor

$V_D$ = Voltage drop from X-diode

$V_L = L \cdot \frac{di_L}{dt}$

$i_c = \frac{C\,d(V_b - 0)}{dt}$




KVL $\Rightarrow V_i(t) - V_D(i_L) - V_L(t) - V_c(t) = 0$

$$\frac{di_L}{dt} = \frac{1}{L}\left[V_i(t) - V_b(t) - V_D(i_L)\right]$$ $\Rightarrow$ Inductor Current

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

If $i_L(t) \le 0$ And $(V_i(t) - V_b(t))$ $\widehat{0.25V}$

$\Downarrow$

$\frac{di_L}{dt} = 0$

$i_L(t) = 0$

$$V_D(i_L) = \begin{cases} 0\,V, & i_D \le 0 \\ 0.25\,V, & 0 < i_D \le 5 \\ 0.25 + 0.035(i_D - 5), & 5 < i_D \le 15 \\ 0.60 + 0.020(i_D - 15), & 15 < i_D \le 20 \\ 0.20 + 0.040(i_D - 20), & 20 < i_D < 25 \end{cases}$$

```python
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:
```

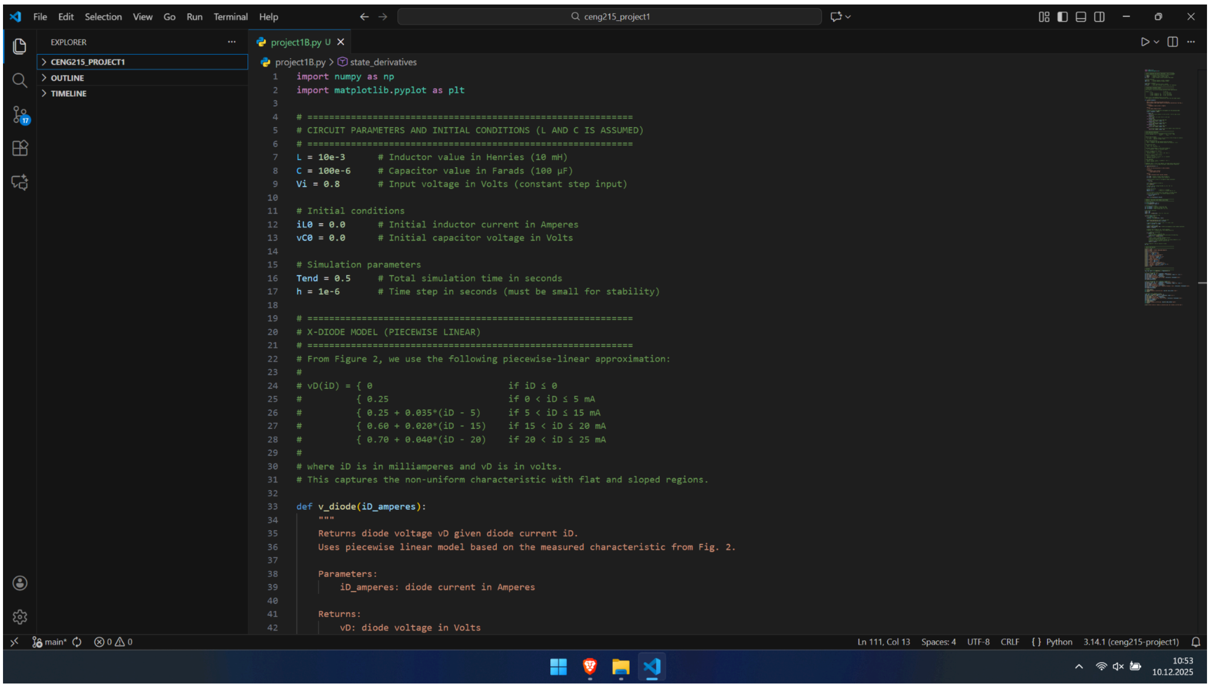
```
#   Vi(t) = vD + vL + vC
#   Vi(t) = vD + L*(diL/dt) + vC
#
# The diode voltage depends on the current through it:
#   vD = v_diode(iL)  [from our piecewise model]
#
# Inductor voltage-current relation:
#   vL = L * (diL/dt)
#   Therefore: diL/dt = vL/L = [Vi(t) - vD - vC] / L
#
# Capacitor current-voltage 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
```

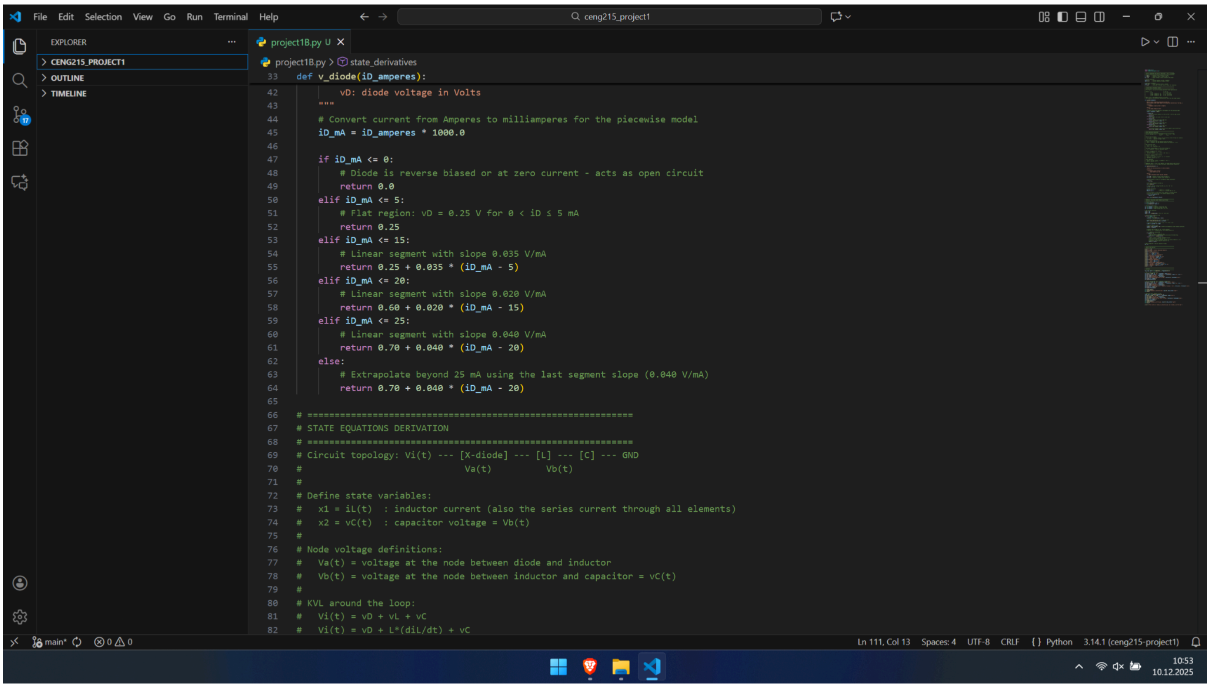
```
    vD = v_diode(iL)

    # 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
```

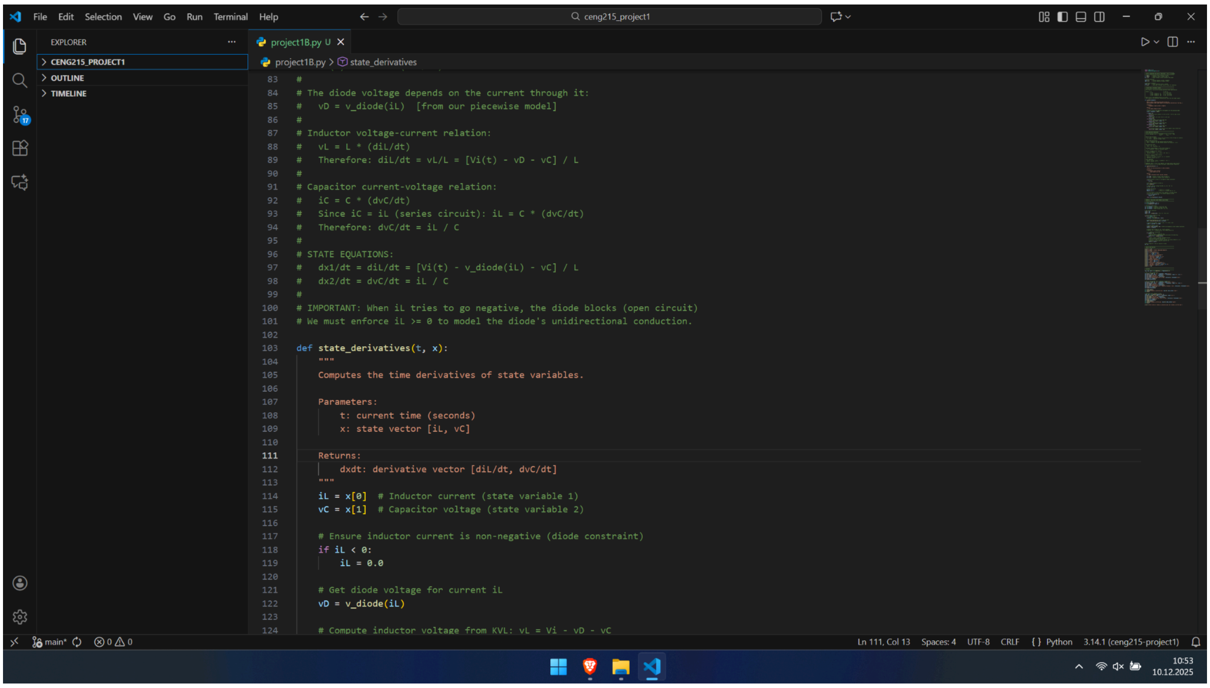
```
    # Extract updated states
    iL[k+1] = max(x_next[0], 0.0)  # Enforce non-negative current
(diode constraint)
    vC[k+1] = x_next[1]

    # 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')
```

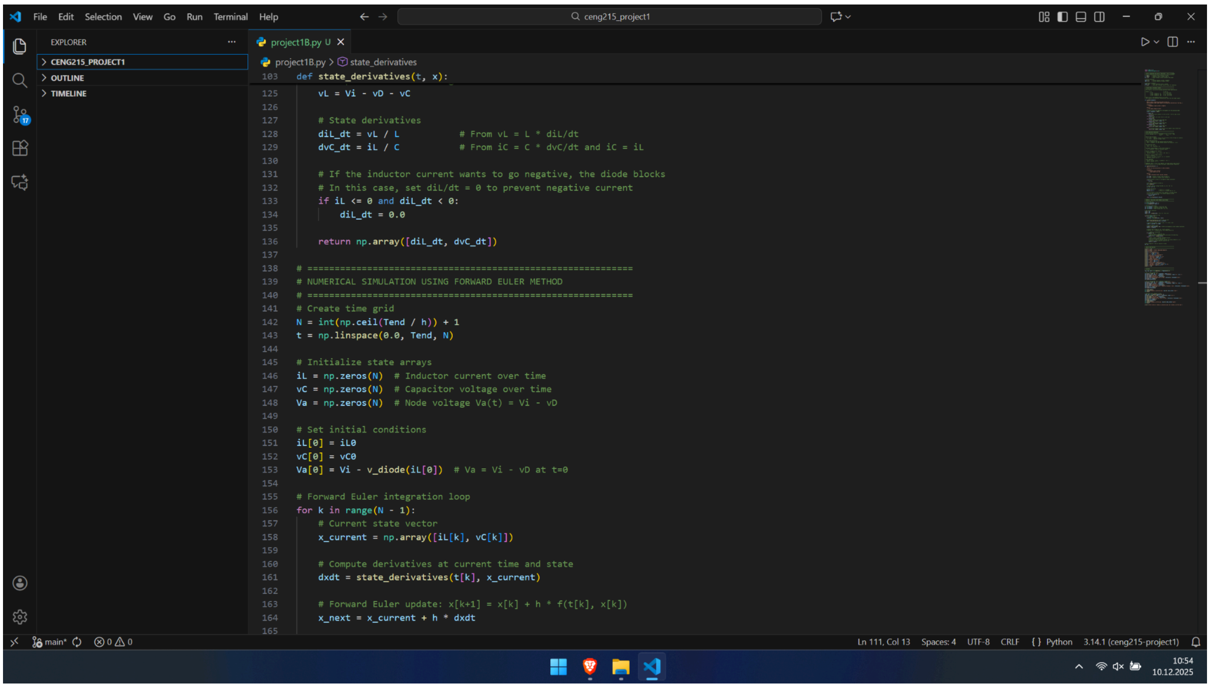
```
ax3.grid(True, alpha=0.3)
ax3.legend(fontsize=10)
plt.tight_layout()
plt.savefig('xdiode_lc_current.png', dpi=150, bbox_inches='tight')
plt.show()

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



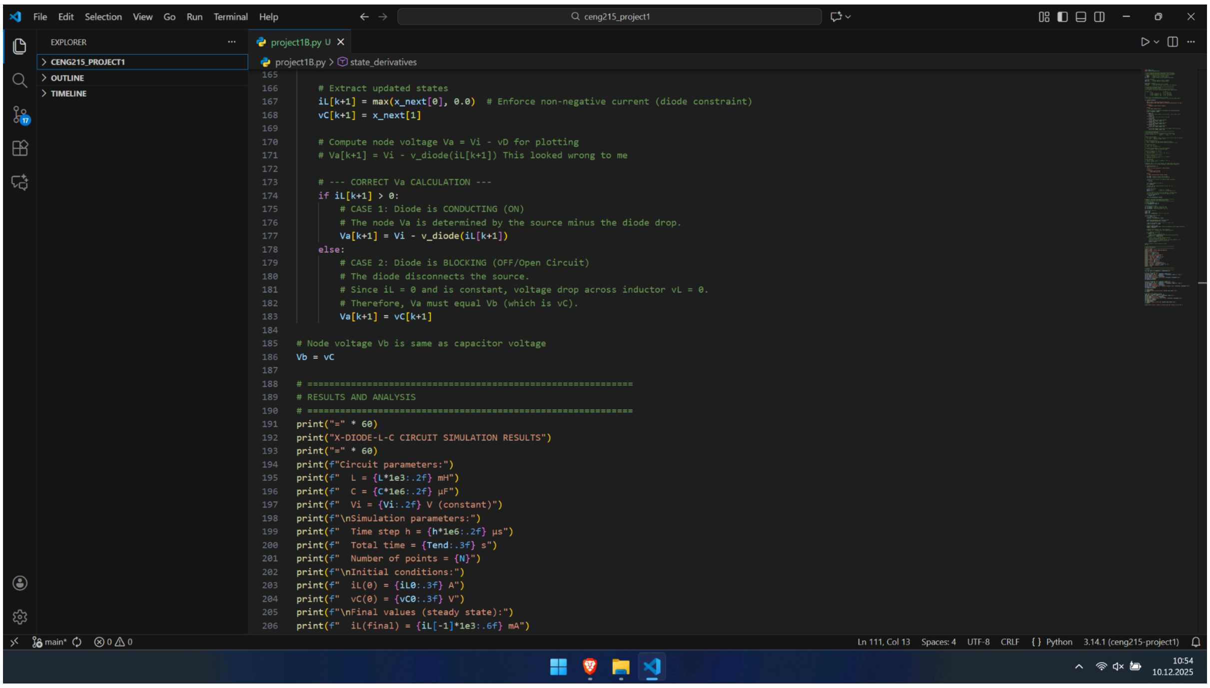





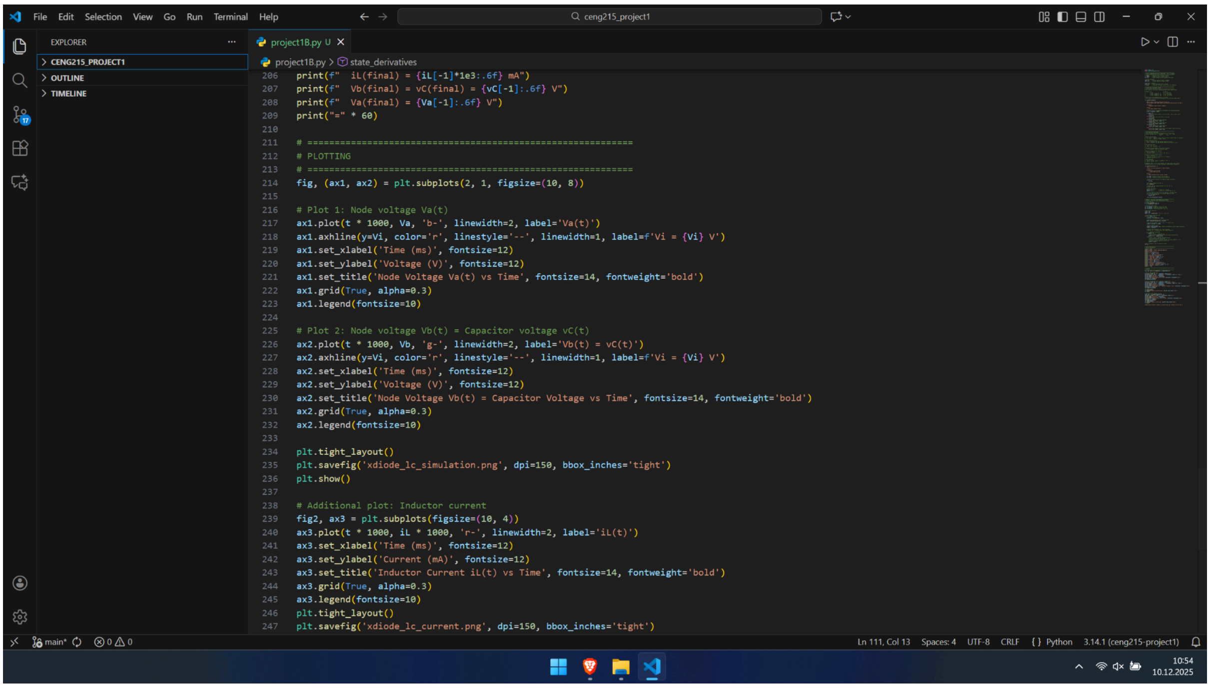
```python
#
# The diode voltage depends on the current through it:
#   vD = v_diode(iL)  [from our piecewise model]
#
# Inductor voltage-current relation:
#   vL = L * (diL/dt)
#   Therefore: diL/dt = vL/L = [Vi(t) - vD - vC] / L
#
# Capacitor current-voltage relation:
#   iC = C * (dvC/dt)
#   Since 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
```



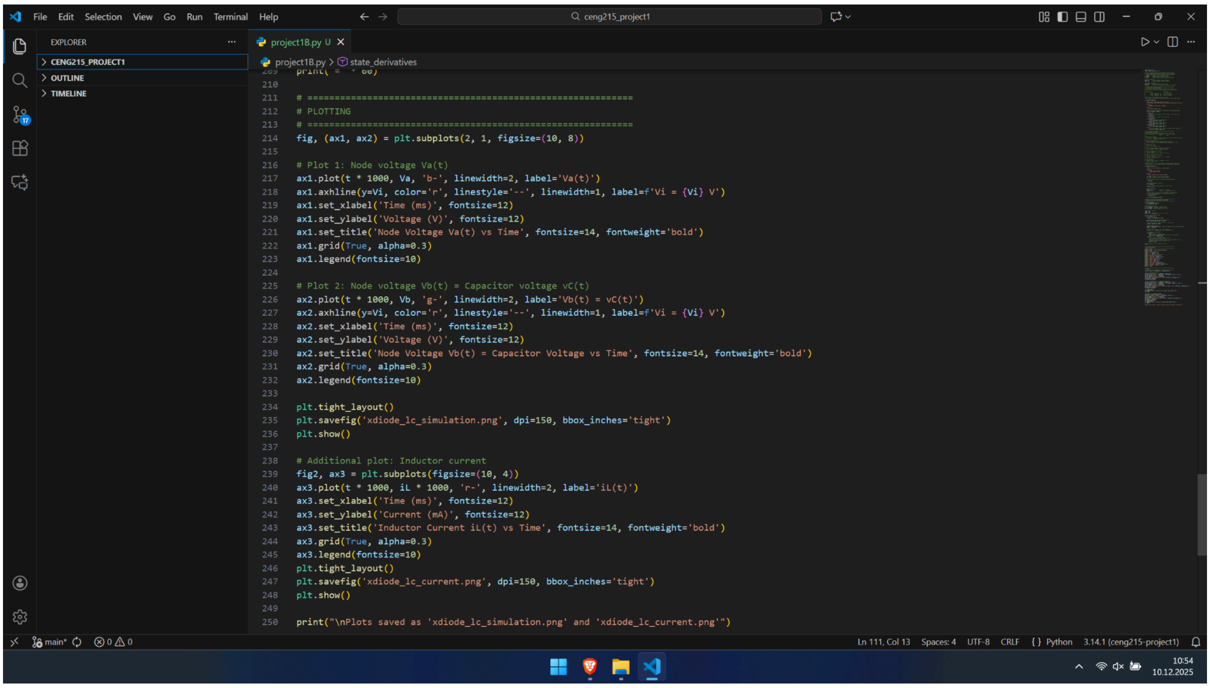
```python
def state_derivatives(t, x):
    vL = Vi - vD - vC

    # 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

```

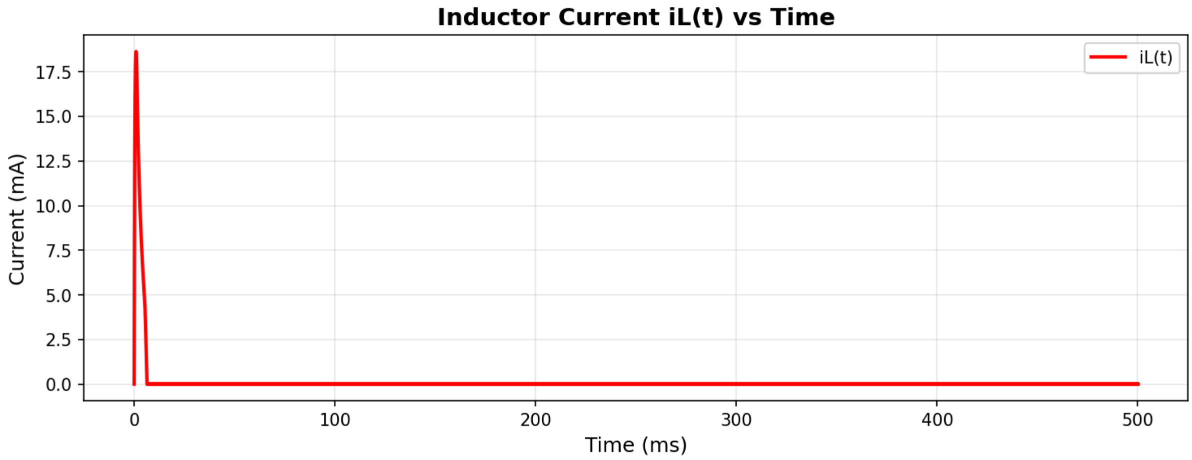


```python
    # Extract updated states
    iL[k+1] = max(x_next[0], 0.0)  # Enforce non-negative current (diode constraint)
    vC[k+1] = x_next[1]

    # 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")
```

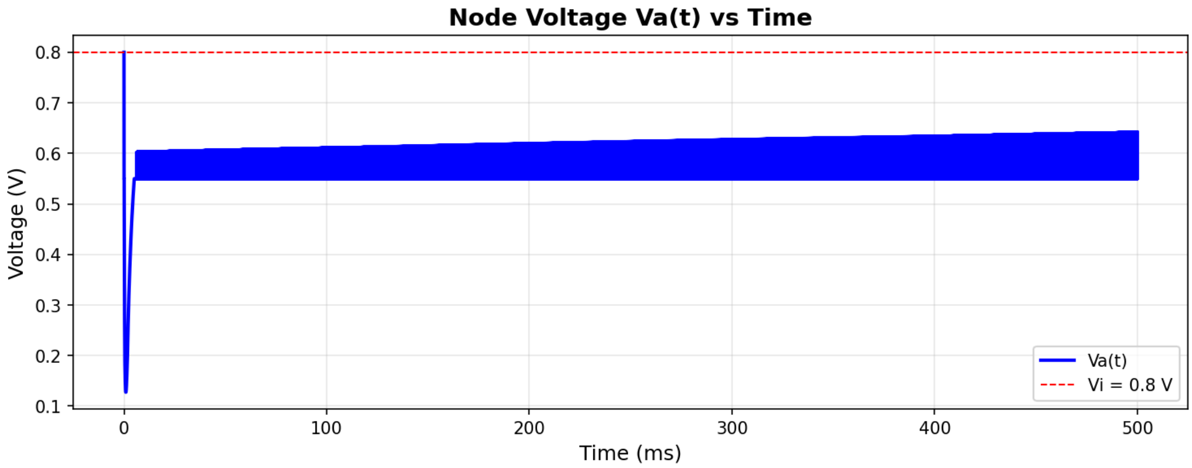


```python
print(f"  iL(final) = {iL[-1]*1e3:.6f} mA")
print(f"  Vb(final) = vC(final) = {vC[-1]:.6f} V")
print(f"  Va(final) = {Va[-1]:.6f} V")
print("=" * 60)

# ============================================================
# 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')
```

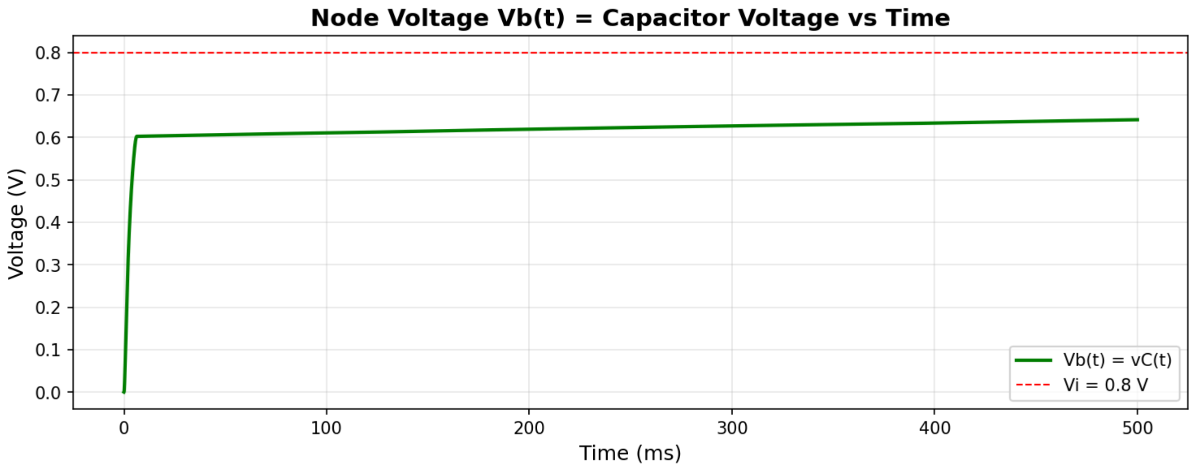

File Edit Selection View Go Run Terminal Help
ceng215_project1
EXPLORER
CENG215_PROJECT1
OUTLINE
TIMELINE
project1B.py U
project1B.py > state_derivatives
# ============================================================
# PLOTTING
# ============================================================
fig, (ax1, 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'")
main*
Ln 111, Col 13 Spaces: 4 UTF-8 CRLF { } Python 3.14.1 (ceng215-project1)
10:54
10.12.2025



Inductor Current iL(t) vs Time
iL(t)
Current (mA)
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0
100
200
300
400
500
Time (ms)

Node Voltage Va(t) vs Time
Va(t)
Vi = 0.8 V
Voltage (V)
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
100
200
300
400
500
Time (ms)

Node Voltage Vb(t) = Capacitor Voltage vs Time
Vb(t) = vC(t)
Vi = 0.8 V
Voltage (V)
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
100
200
300
400
500
Time (ms)

```
============================================================
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'
```