Level 2: State-Space Averaging

From Basic 2-State to Advanced 4-Topology Averaging

Why Averaging? Understanding the Time-Scale Separation

Problem: Switches operate at high frequency (f_{sw} = 50 kHz), but we want to control low-frequency dynamics (AC line = 50/60 Hz, control bandwidth ~kHz).

Solution: State-space averaging "smooths out" rapid switching ripple, revealing the slower underlying dynamics that are relevant for control design.

Foundation: Standard SEPIC Averaging (2 States)

A standard SEPIC has switch ON (duration $D \cdot T_s$) and OFF (duration $(1-D) \cdot T_s$)

```
Classical averaging formula:
A_avg = D·A_1 + (1-D)·A_2
B_avg = D·B_1 + (1-D)·B_2

where D is the duty cycle (0 < D < 1)
```

Physical meaning: The averaged model represents the "time-weighted" behavior over one switching period, eliminating high-frequency ripple while preserving low-frequency dynamics.

Extension: Two-Phase Interleaved System (4 Topologies)

With two independent switches (S1, S2), we have four possible combinations:
11 (both ON), **10** (S1 ON, S2 OFF), **01** (S1 OFF, S2 ON), **00** (both OFF)

```
Understanding Topology Weights:
• k = topology index ∈ {11, 10, 01, 00} (binary representation of switch states S1S2)
• w_k = "duration weight" = fraction of switching period T_s spent in topology k
Example: If w_11 = 0.3, the system spends 30% of each cycle with both switches ON
• Why needed? Each topology has different energy transfer characteristics (storage vs. transfer). Knowing how long we spend in each mode lets us calculate the average behavior.
Constraint: w_11 + w_10 + w_01 + w_00 = 1
(System must be in exactly one of the four topologies at all times)
```

Timing Diagram Visualization:

The numbers below each diagram show **time markers** within one switching period T_s. They indicate when each topology begins/ends, measured from time 0 at the start of the period.

```
Non-overlapping (d1=0.3, d2=0.4, sum=0.7 < 1):
T_s: |----S1 ON----|----S2 ON-----|-----BOTH OFF-----|
     0            d1·T_s        (d1+d2)·T_s            T_s
     |←  w_10=0.3  →|←   w_01=0.4   →|←     w_00=0.3     →|

Time markers explained:
• 0: Start of period
• d1·T_s: S1 turns OFF at time = (duty cycle d1) × (period T_s)
• (d1+d2)·T_s: S2 turns OFF
• T_s: End of period (next cycle begins)
```

```
Overlapping (d1=0.6, d2=0.7, sum=1.3 ≥ 1):
T_s: |--S1-only--|---BOTH ON----|--S2-only--|
0=(1-d2)·T_s  (d1+d2-1)·T_s   (1-d1)·T_s      T_s
     |← w_10=0.3 →|←   w_11=0.3   →|← w_01=0.4 →|

Time markers explained:
• 0: S1 turns ON first
• (1-d2)·T_s: S2 turns OFF, leaving only S1 ON
• (1-d1)·T_s: Both ON briefly before S1 turns OFF
• (d1+d2-1)·T_s: S1 turns OFF, only S2 ON remains
• T_s: End of period
```

Duty Cycle Weight Derivation:

Key insight: Each weight w_k = (duration of topology k) / T_s

```
Non-overlapping case (d1 + d2 < 1): Switches never ON simultaneously, so w_11 = 0.
• S1 is ON alone for duration = d1·T_s → w_10 = d1
• S2 is ON alone for duration = d2·T_s → w_01 = d2
• Both OFF for remaining time: T_s - d1·T_s - d2·T_s = (1-d1-d2)·T_s → w_00 = 1-d1-d2
Check: w_10 + w_01 + w_00 = d1 + d2 + (1-d1-d2) = 1 ✓
```

```
Overlapping case (d1 + d2 ≥ 1): Switches have overlapping ON times. Assume S1 turns ON at t=0 and stays ON for d1·T_s. S2 turns ON at t=0 and stays ON for d2·T_s.
• S1-only period: From t=d2·T_s (when S2 turns OFF) to t=d1·T_s (when S1 turns OFF)
Duration = d1·T_s - d2·T_s = (d1-d2)·T_s
But we need this in normalized form: Since d2 > 1-d1, rearranging gives: w_10 = 1 - d2
• Both ON period: Overlap from t=0 to min(d1·T_s, d2·T_s)
When d1+d2 ≥ 1, overlap duration = d1·T_s + d2·T_s - T_s = (d1+d2-1)·T_s → w_11 = d1+d2-1
• S2-only period: Symmetric to S1-only w_01 = 1 - d1
• Both OFF: Since duty cycles overlap, no time remains with both OFF → w_00 = 0
Check: w_11 + w_10 + w_01 = (d1+d2-1) + (1-d2) + (1-d1) = 1 ✓
```

Final Averaged Model

```
State Matrix (time-weighted average):
A_avg(d1,d2) = w_11·A_11 + w_10·A_10 + w_01·A_01 + w_00·A_00

Input Matrix (with AC polarity handling):
B_in,avg(d1,d2,s) = s·(w_11·B_11 + w_10·B_10 + w_01·B_01 + w_00·B_00)

Complete averaged system:
ẋ = A_avg(d1,d2)·x + B_in,avg(d1,d2,s)·V_in + f_CPL(vC0)
```

Understanding the 's' variable:

$s = \text{sign}(V_{in}) \in \{+1, -1\}$ handles AC input polarity

Why do we need it? This converter is a **PFC (Power Factor Correction) rectifier** operating directly from AC mains:
• **Positive half-cycle ($V_{in} > 0$):** s = +1 → Current flows through L1, L2 (active), L3, L4 inactive
• **Negative half-cycle ($V_{in} < 0$):** s = -1 → Current flows through L3, L4 (active), L1, L2 inactive
The **bridgeless topology** uses different inductor sets for each AC polarity, eliminating the need for a diode bridge rectifier (reducing losses by ~1-2%). The sign variable ensures the averaged model correctly represents energy flow regardless of AC polarity.

Key Insight: This 4-topology averaging generalizes the classical 2-state method. It captures the complex interleaved operation while maintaining the same fundamental principle: time-weighted averaging over a switching period smooths out high-frequency ripple (50 kHz) to reveal low-frequency control dynamics.