ISDO Backend Strategy - Recommendations

Current Backend Performance Summary

Backend	Sign Agreement	Correlation	Mean Error	Status
MathInterferenceBackend	100%	1.0	~10^{-16}	✅ Reference
HadamardInterferenceBackend	100%	1.0	~10^{-16}	✅ Conceptual
TransitionInterferenceBackend	100%	1.0	~10^{-14}	✅ **USE THIS**
ISDOBPrimeInterferenceBackend	51%	-0.029	0.214	× Wrong observable
ISDOBPrimeV2InterferenceBackend	50%	0.078	0.863	× **Random performance**

Problem with B′ and B′ v2

Both attempt to avoid ancilla qubits but compute **quadratic observables**:

B′ (Original)

```
# Measures: <psi | U_chi^† Z^{⊗n} U_chi | psi>
# This is NOT Re⟨χ|ψ⟩
```

B′ v2 (Current)

```
# Measures: 2*|<chi|psi>|^2 - 1
# This is quadratic (fidelity-like), not linear ISDO
```

Why they fail:

- Single-state measurements give |amplitude|² (quadratic)
- Linear overlap Re⟨χ|ψ⟩ requires **interference between two states**
- You need TWO state paths to interfere (like Hadamard test provides)

Recommended Solution

Option 1: Use TransitionInterferenceBackend (BEST) ✅

This is **verified working** in your tests. Use it as your primary backend:

```
# In your training/inference code
from src.IQC.interference.circuit_backend_transition import
TransitionInterferenceBackend

# Initialize
backend = TransitionInterferenceBackend()

# In MemoryBank
memory_bank = MemoryBank(
    class_states=class_states,
    backend=backend  # Use this!
)
```

Advantages:

- ✅ Correct ISDO observable: Re⟨χ|ψ⟩
- ✅ Physically realizable (unitary operations)
- ✅ Verified numerically (10^{-14} error)
- ✅ Proper sign/phase sensitivity

Trade-offs:

- Requires ancilla qubit (total: n+1 qubits)
- More complex circuit than B' variants
- Higher gate count

Option 2: Keep MathInterferenceBackend for Simulation

For exact statevector simulation (no noise):

```
backend = MathInterferenceBackend()  # Fastest, most accurate
```

Use for:

- Development and debugging
- Hyperparameter tuning
- Quick experiments

Option 3: Hybrid Approach

```
class AdaptiveBackend(InterferenceBackend):
    def __init__(self, use_quantum=False):
        if use_quantum:
            self.backend = TransitionInterferenceBackend()
        else:
            self.backend = MathInterferenceBackend()
```

```
    def score(self, chi, psi):
        return self.backend.score(chi, psi)
```

What to Do with B' v2

Rename and Repurpose

Your B' v2 actually computes a **valid fidelity-based observable**:

```
class FidelityInterferenceBackend(InterferenceBackend):
    """
    Computes quadratic fidelity: 2*|<chi|psi>|^2 - 1

    This is NOT ISDO, but can be useful for:
    - Fidelity-based classification (like RFC)
    - Comparison experiments
    - Measuring state quality
    """

    def score(self, chi, psi) -> float:
        # Keep existing implementation
        # But document it computes fidelity, not ISDO
        ...
```

Use as Baseline Comparison

Compare ISDO (linear) vs Fidelity (quadratic) performance:

```
# Train with ISDO
isdo_backend = TransitionInterferenceBackend()
isdo_bank = MemoryBank(states, isdo_backend)
acc_isdo = train_and_evaluate(isdo_bank)

# Train with Fidelity
fidelity_backend = FidelityInterferenceBackend()  # Renamed B' v2
fidelity_bank = MemoryBank(states, fidelity_backend)
acc_fidelity = train_and_evaluate(fidelity_bank)

print(f"ISDO accuracy: {acc_isdo}")
print(f"Fidelity accuracy: {acc_fidelity}")
```

Implementation Plan

Step 1: Update Default Backend

```

```

```
# In src/IQC/memory/memory_bank.py
from ..interference.circuit_backend_transition import
TransitionInterferenceBackend

class MemoryBank:
    def __init__(self, class_states, backend=None):
        self.class_states = class_states
        # Default to correct ISDO backend
        self.backend = backend or TransitionInterferenceBackend()
```

Step 2: Update All Training Scripts

```
# In run_regime2.py, run_regime3a.py, etc.
from src.IQC.interference.circuit_backend_transition import
TransitionInterferenceBackend

backend = TransitionInterferenceBackend()
memory_bank = MemoryBank(class_states, backend)
```

Step 3: Add Comparative Experiments

```
# New file: src/experiments/iqc/compare_observables.py
def compare_isdo_vs_fidelity():
    """
    Compare ISDO (linear) vs Fidelity (quadratic) classifiers
    """
    # ISDO backend
    isdo_backend = TransitionInterferenceBackend()

    # Fidelity backend (renamed B' v2)
    fidelity_backend = FidelityInterferenceBackend()

    # Train both and compare
    ...
```

Step 4: Documentation

Update comments in all backend files to clarify:

- What observable each computes
- When to use each
- Performance characteristics

Why You Can't Fix B' v2 to Be Linear

Mathematical impossibility:

To get Re⟨χ|ψ⟩ from measurements, you need:

1. **Two interfering paths**: |ψ⟩ and |χ⟩ in superposition
2. **Phase preservation**: Maintain relative phase between paths
3. **Interference measurement**: Extract the real part of inner product

Single-state measurement scenarios (B′, B′ v2) give:

- $|\langle \text{basis}|\text{state}\rangle|^2$ - always quadratic
- No access to phase information
- No interference between different states

The ancilla qubit in Hadamard/Transition test is essential - it creates the superposition that enables interference.

Final Recommendations

1. ✅ **Primary backend**: `TransitionInterferenceBackend`
   - Use for all ISDO experiments
   - Already verified working
2. ✅ **Development backend**: `MathInterferenceBackend`
   - Fast prototyping
   - Exact reference
3. ✅ **Rename B′ v2**: `FidelityInterferenceBackend`
   - Document as fidelity-based
   - Use for comparison studies
4. × **Deprecate**: `ISDOBPrimeInterferenceBackend`
   - Wrong observable
   - Confusing naming
5. 📝 **Document**: Update all docstrings
   - Clarify what each backend computes
   - Explain trade-offs
   - Provide usage guidelines

Expected Performance After Fix

With `TransitionInterferenceBackend` as default:

- Sign agreement: 100% (vs current 50%)
- Correlation: 1.0 (vs current 0.078)
- Accurate ISDO scores
- Proper phase-sensitive classification

Your Regime 3-C results should improve significantly!