Regime Switching in Quadratic Programming with CVXPY

Your Name

May 20, 2025

Abstract

This document presents a comprehensive guide to implementing regime-switching quadratic programming problems using CVXPY. We cover multiple formulations, from basic binary regime selection to asset-specific sign-based switching, with vectorized implementations and practical considerations.

1 Introduction

Regime-switching optimization allows different objective functions or constraints to be activated based on certain conditions. In portfolio optimization, this enables modeling different market conditions or asset-specific behaviors. We explore several formulations using mixed-integer quadratic programming (MIQP) in CVXPY.

2 Basic Regime Switching

2.1 Problem Formulation

Given two regimes with quadratic objectives:

$$\text{Regime 1: } x^T Q_1 x + c_1^T x$$
$$\text{Regime 2: } x^T Q_2 x + c_2^T x$$

We introduce a binary variable $z \in \{0, 1\}$ to select the regime.

2.2 CVXPY Implementation

```
import cvxpy as cp
import numpy as np

n = 3  # Problem dimension
```

```
x = cp.Variable(n)
z = cp.Variable(boolean=True)

# Parameters
Q1 = np.eye(n); Q2 = 2*np.eye(n)
c1 = np.array([1,2,3]); c2 = np.array([-1,-1,-1])
M = 1000  # Big-M constant

# Objectives
obj1 = cp.quad_form(x, Q1) + c1@x
obj2 = cp.quad_form(x, Q2) + c2@x

# Constraints
t = cp.Variable()
constraints = [
    t >= obj1 - M*z,
    t >= obj2 - M*(1-z),
    x >= 0,
    cp.sum(x) == 1
]

problem = cp.Problem(cp.Minimize(t), constraints)
problem.solve(solver=cp.GUROBI)
```

2.3 Key Insights

- The Big-M method enforces $t \geq \text{obj}_1$ when $z = 0$ and $t \geq \text{obj}_2$ when $z = 1$
- M must be large enough but not cause numerical instability
- Requires a MIQP-capable solver like Gurobi or CPLEX

3 Asset-Specific Regime Switching

3.1 Problem Formulation

For each asset i, we want:

$$\text{Objective} = \begin{cases} Q_{1,i} w_i^2 + c_{1,i} w_i & \text{if } w_i > 0 \\ Q_{2,i} w_i^2 + c_{2,i} w_i & \text{if } w_i \leq 0 \end{cases}$$

3.2 Vectorized Implementation

```
n = 4
W = cp.Variable(n)
z = cp.Variable(n, boolean=True)
```

```
t = cp.Variable(n)

# Parameters
Q1_diag = np.ones(n); Q2_diag = 5*np.ones(n)
c1 = np.array([0.2,0.1,0.15,0.05])
c2 = np.array([-0.1,-0.2,-0.1,-0.05])
M = 1e3; eps = 1e-4

# Constraints
constraints = [
    W >= -M*(1-z),        # If z=1 → W  0
    W <= -eps + M*z,      # If z=0 → W  -eps
    t >= Q1_diag*cp.square(W) - cp.multiply(c1,W) - M*(1-z),
    t >= Q2_diag*cp.square(W) - cp.multiply(c2,W) - M*z,
    cp.sum(W) == 1
]

prob = cp.Problem(cp.Minimize(cp.sum(t)), constraints)
prob.solve(solver=cp.GUROBI)
```

3.3 Key Features

- Each asset independently selects its regime based on weight sign
- Vectorized operations improve efficiency and readability
- ϵ creates numerical separation between regimes
- Allows both long ($w_i > 0$) and short ($w_i < 0$) positions

4 Advanced Topics

4.1 Multiple Regimes

Extend to k regimes using one-hot encoding:

$$\sum_{j=1}^{k} z_{i,j} = 1 \quad \forall i$$

where $z_{i,j} \in \{0, 1\}$ indicates whether asset i is in regime j.

4.2 Time-Varying Regimes

For portfolio weights $W \in \mathbb{R}^{n \times T}$:

```
Z = cp.Variable((n,T), boolean=True)
for t in range(T):
```

```
constraints += [
    W[:,t] >= -M*(1-Z[:,t]),
    W[:,t] <= -eps + M*Z[:,t]
]
```

5 Numerical Considerations

Parameter	Recommendation
Big-M (M)	10-1000 times typical variable scale
ϵ	10^{-4} to 10^{-8} depending on precision
Solver	Gurobi/CPLEX for MIQP, ECOS_BB for small problems

Table 1: Numerical parameters guidance

6 Conclusion

The presented methods enable flexible regime-switching optimization in CVXPY. Key takeaways:

- Binary variables encode regime selection
- Big-M constraints handle conditional objectives
- Vectorization improves performance
- Careful parameter tuning ensures reliability

7 Multi-Regime Switching Implementation

7.1 Problem Formulation

Consider four distinct regimes with:

- Regime 1: Low-risk, low-return (e.g., cash equivalents)
- Regime 2: Moderate-risk balanced portfolio
- Regime 3: High-growth equity focus
- Regime 4: Crisis mode (high penalty for risk)

The mathematical formulation becomes:

$$
\begin{aligned}
\min_{x,z} \quad & \sum_{k=1}^{4} z_k \left(\frac{1}{2} x^T Q_k x + c_k^T x + \phi_k \|x\|_1 \right) \\
\text{s.t.} \quad & \sum_{i=1}^{n} x_i = 1 \\
& \sum_{k=1}^{4} z_k = 1 \\
& z_k \in \{0,1\} \quad \forall k \in \{1,2,3,4\} \\
& A_k x \leq b_k \text{ when } z_k = 1
\end{aligned}
\tag{1}
$$

7.2 CVXPY Implementation

```
import cvxpy as cp
import numpy as np

# Problem dimensions
n = 10  # Number of assets
K = 4   # Number of regimes
M = 1e5 # Big-M constant
eps = 1e-4 # Numerical epsilon

# Variables
x = cp.Variable(n)  # Portfolio weights
Z = cp.Variable(K, boolean=True)  # One-hot regime vector

# Regime-specific parameters
Q = [None]*K
c = [None]*K
phi = [0.001, 0.005, 0.01, 0.05]  # Regime-specific penalties

# Construct regime parameters
for k in range(K):
    # Diagonal risk matrices with increasing risk
    Q[k] = (k+1)*np.eye(n) + 0.1*np.random.rand(n,n)
    Q[k] = (Q[k] + Q[k].T)/2  # Ensure symmetry

    # Return vectors with varying profiles
    if k == 0:      # Cash regime
        c[k] = 0.02*np.ones(n)
    elif k == 1:    # Balanced
        c[k] = 0.05*np.random.rand(n)
    elif k == 2:    # Growth
        c[k] = 0.1*np.random.rand(n)
```

```
    else:           # Crisis
        c[k] = -0.1*np.ones(n)  # Negative expected returns

# Construct objective terms
t = cp.Variable()  # Auxiliary variable
obj_terms = []
for k in range(K):
    reg_obj = 0.5*cp.quad_form(x, Q[k]) + c[k]@x + phi[k]*cp.norm1(x)
    obj_terms.append(reg_obj - M*(1-Z[k]))

constraints = [
    t >= cp.max(cp.hstack(obj_terms)),  # Takes the active regime
    cp.sum(Z) == 1,                     # Exactly one regime active
    cp.sum(x) == 1,                     # Fully invested
    x >= -1, x <= 1                     # Leverage constraints
]

# Additional regime-specific constraints
for k in range(K):
    if k == 0:  # Cash regime constraints
        constraints += [x <= Z[k] + 0.2*(1-Z[k])]  # Max 20% in non-cash
    elif k == 3:  # Crisis regime constraints
        constraints += [cp.norm(x, 1) <= 1.5 - 0.5*Z[k]]  # Reduce leverage

# Solve the problem
prob = cp.Problem(cp.Minimize(t), constraints)
prob.solve(solver=cp.GUROBI, verbose=True)

# Print results
print("Optimal weights:", x.value)
print("Active regime:", np.argmax(Z.value))
print("Regime probabilities:", Z.value)
```

7.3 Key Features

- **One-Hot Encoding**: Uses $Z \in \{0,1\}^4$ with $\sum Z_k = 1$ to ensure exactly one active regime
- **Regime-Specific Parameters**:
  - Different risk matrices Q_k
  - Varying return vectors c_k
  - Regime-dependent penalty terms ϕ_k
- **Big-M Formulation**: Only the active regime's objective contributes meaningfully

- **Regime Constraints**: Conditional constraints activated based on Z_k

7.4 Mathematical Analysis

The formulation ensures:

1. Convexity is maintained within each regime
2. The Big-M value dominates all possible objective values:

$$M > \max_{x,k} \left| \frac{1}{2} x^T Q_k x + c_k^T x \right| \quad \forall x \text{ feasible} \tag{2}$$

3. The problem remains MIQP-representable:

$$\begin{aligned} \min_{x,z,t} \quad & t \\ \text{s.t.} \quad & t \geq \frac{1}{2} x^T Q_k x + c_k^T x - M(1 - z_k) \quad \forall k \\ & \sum z_k = 1 \\ & z_k \in \{0, 1\} \quad \forall k \end{aligned} \tag{3}$$

7.5 Performance Optimization

To improve computational efficiency:

```
# Warm-start heuristic
def get_initial_guess(Q, c):
    """Solve relaxed problems to initialize regimes"""
    x_init = np.zeros((K, n))
    for k in range(K):
        prob = cp.Problem(
            cp.Minimize(0.5*cp.quad_form(x, Q[k]) + c[k]@x),
            [cp.sum(x) == 1, x >= -1, x <= 1]
        )
        prob.solve()
        x_init[k] = x.value
    return x_init

x_init = get_initial_guess(Q, c)
x.value = np.mean(x_init, axis=0)  # Warm-start with average
```

7.6 Regime Interpretation

7.7 Extension to Probabilistic Regimes

For stochastic regime probabilities p_k:

$$\min_x \sum_{k=1}^{4} p_k \left(\frac{1}{2} x^T Q_k x + c_k^T x \right) \tag{4}$$

Regime	Risk Profile	Typical Holdings	Activation Condition
1	Low (Q_1)	Cash, short-term bonds	High volatility
2	Moderate (Q_2)	Balanced portfolio	Normal markets
3	High (Q_3)	Growth equities	Bull markets
4	Severe (Q_4)	Inverse ETFs, puts	Crisis indicators

Table 2: Regime characteristics and typical allocations

Implemented via:

```
p = np.array([0.1, 0.6, 0.25, 0.05])  # Regime probabilities
prob = cp.Problem(
    cp.Minimize(sum(p[k]*(0.5*cp.quad_form(x,Q[k]) + c[k]@x) for k in range(K))),
    constraints
)
```