Technical Documentation: Adaptive Bayesian-Geometric Trading Algorithm

Executive Summary

The Adaptive Bayesian-Geometric Trading Algorithm is a sophisticated quantitative trading system that combines geometric pattern recognition with Bayesian probability theory to identify high-probability trading opportunities in financial markets. Designed specifically for the EURUSD currency pair on 5-minute timeframes, the system implements adaptive learning mechanisms that continuously improve performance based on market feedback.

1. Algorithm Basis & Theoretical Foundation

1.1 Core Philosophy

The algorithm operates on the principle that financial markets exhibit both mean-reverting and breakout behaviours around geometrically significant price levels. By quantifying the probability of each behaviour type, the system can adaptively position itself to capitalize on the most likely market movement.

1.2 Theoretical Underpinnings

Market Microstructure Theory:

- Price action clusters around psychologically and mathematically significant levels
- Support/resistance levels emerge from collective market participant behaviour
- Temporary inefficiencies create mean-reversion opportunities

Bayesian Probability Framework:

- Prior beliefs about market state are continuously updated with new evidence
- The system learns from its own trading outcomes to refine probability estimates
- Adaptive confidence thresholds prevent overfitting to recent market conditions

2. Mathematical Framework

2.1 Geometric Level Detection

Pivot Point Identification

python

```python
# Swing High/Low Detection
for i in range(lookback, len(high) - lookback):
    # Swing High: higher than lookback periods before and after
    if all(high.iloc[i] > high.iloc[i - j] for j in range(1, lookback + 1)) and \
       all(high.iloc[i] > high.iloc[i + j] for j in range(1, lookback + 1)):
        pivots_high.append((i, high.iloc[i]))
```

Level Clustering Algorithm

- Pivot points within `threshold × ATR` are clustered
- Cluster strength: `weight = 1.0 + (cluster_size × 0.2)`
- Fibonacci levels derived from major swing points with weighted significance

2.2 Bayesian Inference Engine

Feature Vector Construction

The system calculates four primary features:

1. **Normalized Distance (d)**: `distance_to_nearest_level / ATR`
2. **Clustering Strength (c)**: Sum of weights of levels within 0.5 ATR
3. **Momentum (mom)**: `(price_current - price_5periods_ago) / ATR`
4. **RSI**: 14-period Relative Strength Index

Likelihood Model

python

```python
def likelihood_model(self, features, state):
    d, c, rsi, mom, price_vs_level = features

    if state == 'revert':
        distance_factor = 1.0 / (1.0 + abs(d - optimal_distance))
        clustering_factor = min(2.0, c) / 2.0
        momentum_factor = 1.0 - (abs(rsi - 50) / 60)
```

```
        likelihood = distance_factor * 0.4 + clustering_factor * 0.4 + momentum_fa
ctor * 0.2
    else:  # breakout
        distance_factor = max(0, min(1.0, (d - 0.8) / 1.5))
        clustering_factor = 1.0 - (min(2.0, c) / 2.0)
        momentum_factor = 1.0 - (abs(rsi - 50) / 60)
        likelihood = distance_factor * 0.5 + clustering_factor * 0.3 + momentum_fa
ctor * 0.2
```

Bayesian Update Rule

```
text
```

```
Posterior = (Likelihood × Prior) / Evidence

Where:
Evidence = P(Data|Revert) × P(Revert) + P(Data|Breakout) × P(Breakout)
```

2.3 Adaptive Learning Mechanism

Performance-Based Threshold Adjustment

```
python
```

```
if recent_win_rate < 0.3:  # Poor performance
    self.min_confidence = min(0.85, self.min_confidence + 0.05)
    self.max_confidence = max(0.15, self.max_confidence - 0.05)
elif recent_win_rate > 0.6:  # Good performance
    self.min_confidence = max(0.70, self.min_confidence - 0.02)
    self.max_confidence = min(0.25, self.max_confidence + 0.02)
```

Prior Probability Adaptation

- Base prior adjusted based on EMA trend detection
- Recent performance influences prior confidence level
- Ranging markets receive higher mean-reversion priors

3. System Architecture

3.1 Component Overview

Component	Purpose	Key Features
GeometricEngine	Level detection & clustering	Pivot points, Fibonacci, ATR normalization
AdaptiveBayesianEngine	Probability calculation	Bayesian inference, adaptive learning
FixedTradeManager	Trade execution & management	Risk management, position synchronization
FixedLearningDataManager	Performance tracking	Trade recording, statistics calculation

3.2 Data Flow Pipeline

1. **Data Acquisition** → MT5 5-minute EURUSD data
2. **Feature Extraction** → Geometric levels + Technical indicators
3. **Probability Calculation** → Bayesian posterior estimation
4. **Signal Generation** → Adaptive threshold comparison
5. **Trade Execution** → Risk-managed position entry
6. **Performance Tracking** → Continuous learning feedback

3.3 Risk Management Framework

Position Sizing

```
python
```

```
risk_amount = account_balance × risk_per_trade
lot_size = risk_amount / (stop_distance × point_value × 100)
```

Stop-Loss & Take-Profit

- Stop-loss: 1.5× ATR or broker minimum
- Take-profit: 2× stop-loss distance (1:2 risk-reward)
- Dynamic breakeven: Activated at 0.8× ATR profit
- Trailing stops: Begin at 1.2× ATR profit

4. Applied Mathematics in Trading Decisions

4.1 Probability Thresholds

Mean-Reversion Signal:
`P(revert) > min_confidence` where `min_confidence ∈ [0.70, 0.85]`

Breakout Signal:
`P(revert) < max_confidence` where `max_confidence ∈ [0.15, 0.25]`

4.2 Distance Filtering

python

```python
# Minimum distance requirement (relaxed for 5M timeframe)
min_distance_atr = 0.03  # 3% of ATR
if distance < (atr × min_distance_atr):
    signal_filtered = True
```

4.3 Trend Integration

python

```python
# EMA-based trend detection
ema_20 = df['close'].ewm(span=20).mean().iloc[-1]
ema_50 = df['close'].ewm(span=50).mean().iloc[-1]

if abs(ema_20 - ema_50) < 0.0005:  # Ranging
    base_prior = 0.6
elif ema_20 > ema_50:  # Uptrend
    base_prior = 0.4
else:  # Downtrend
    base_prior = 0.45
```

5. Suggested Usage & Configuration

5.1 Optimal Market Conditions

- **Best Performance**: Ranging or mildly trending markets
- **Currency Pairs**: EURUSD (primary), other major FX pairs
- **Timeframes**: 5-minute (optimized), 15-minute (compatible)

- **Trading Sessions**: London and New York overlaps

5.2 Parameter Settings

Parameter	Recommended Value	Purpose
ATR Period	14	Volatility normalization
RSI Period	14	Momentum measurement
Pivot Lookback	5	Swing point detection
Max Consecutive Signals	2	Over-trading prevention
Trade Cooldown	300 seconds	Minimum between trades

5.3 Risk Parameters

- **Risk per Trade**: 0.5-2.0% of account balance
- **Maximum Open Trades**: 2 positions
- **Lot Size**: 0.01-1.00 lots (adaptive calculation)

6. Use Cases & Applications

6.1 Primary Use Case: Retail Algorithmic Trading

- Automated execution for individual traders
- Portfolio diversification through systematic approach
- Emotion-free trading discipline

6.2 Institutional Applications

- Alpha generation component in larger portfolios
- Market microstructure research tool
- Benchmark for manual trading strategies

6.3 Educational Value

- Practical implementation of Bayesian statistics
- Real-world geometric pattern recognition
- Adaptive system design principles

7. Performance Findings & Observations

7.1 Strengths Identified

Adaptive Learning:

- System automatically adjusts to changing market volatility
- Confidence thresholds evolve based on recent performance
- Prevents over-optimization to specific market regimes

Robust Risk Management:

- Multiple layers of position validation
- Dynamic stop-loss calculation based on current volatility
- Comprehensive trade synchronization with broker

Geometric Foundation:

- Identifies psychologically significant price levels
- Clustering algorithm reduces false signals from outlier pivots
- Fibonacci integration provides additional confluence

7.2 Limitations & Considerations

Market Regime Dependency:

- Performance varies with market volatility conditions
- Strong trending markets may generate fewer signals
- Requires periodic monitoring during fundamental events

Parameter Sensitivity:

- ATR multiplier settings impact signal frequency
- Confidence thresholds require careful calibration
- Distance filters affect trade opportunity count

Technical Requirements:

- Dependent on stable MT5 connectivity

- Requires continuous operation for learning adaptation
- Memory management for historical data storage

8. Conclusion & Future Enhancements

8.1 Key Achievements

The Adaptive Bayesian-Geometric Trading Algorithm successfully demonstrates:

1. **Theoretical Rigor**: Solid mathematical foundation combining geometry and probability
2. **Practical Implementation**: Robust production-ready trading system
3. **Adaptive Intelligence**: Self-improving mechanism based on performance feedback
4. **Risk-Aware Design**: Comprehensive position and money management

8.2 Empirical Validation

While extensive backtesting is recommended for specific deployment, the algorithm shows promise through:

- Logical signal generation based on multiple confluences
- Adaptive behaviour that responds to market conditions
- Conservative risk management protecting capital
- Transparent decision-making process

8.3 Recommended Enhancements

Short-term Improvements:

- Multi-timeframe geometric level confirmation
- Additional technical indicators for feature enrichment
- Enhanced broker connectivity error handling

Medium-term Developments:

- Machine learning integration for feature weight optimization
- Portfolio-level position sizing across correlated instruments
- Advanced regime detection for parameter auto-adjustment

Long-term Vision:

- Reinforcement learning for complete adaptive control

- Alternative data integration (sentiment, order flow)
- Cloud-based distributed operation for multiple instruments

8.4 Final Assessment

This algorithm represents a sophisticated approach to systematic trading that balances mathematical rigor with practical implementation concerns. The integration of geometric pattern recognition with Bayesian adaptive learning creates a system capable of evolving with market conditions while maintaining disciplined risk management. For traders seeking a methodical, probability-based approach to short-term FX trading, this algorithm provides a robust foundation that can be further enhanced based on individual requirements and market observations.

The system's design philosophy of continuous learning and adaptation makes it particularly valuable in the dynamic foreign exchange markets, where rigid systems often fail during regime changes. By prioritizing risk management and logical signal generation, the algorithm aims for consistent performance rather than explosive returns, making it suitable for capital preservation and steady growth objectives.