

Lagrangian-Based Trading Algorithm

All Juice Capital

May 2025

Abstract

This paper delves into the development and theoretical basis of a trading algorithm that utilizes the Lagrangian formulation to analyze and predict price movements within financial markets. By applying principles of kinetic and potential energy, this algorithm aims to identify optimal entry points for trades, with a particular focus on mean reversion and momentum.

1 An Exploration of Lagrangian Systems

1.1 The Lagrangian: A Synthesis of Energy Concepts

The Lagrangian (L) is a critical component of this algorithm, combining kinetic energy (KE) and potential energy (PE) to describe the state of the price system.

$$L = KE - PE$$

Where the kinetic energy is defined as:

$$KE = \frac{1}{2} \cdot m \cdot v^2$$

Here, m is a mass-like constant, adjusted to reflect the volatility of the market and velocity v as the rate of change of price over time. Then, we define the rate of change of price v as:

$$v = \frac{\text{currentPrice} - \text{previousPrice}}{\text{timeElapsed}}$$

1.2 Modeling Mean Reversion with Potential Energy

The concept of potential energy (PE) is used to model the tendency of prices to revert to their mean, or average.

$$PE = k \cdot (\text{price} - \text{meanPrice})^2$$

Here, k is a constant that determines the strength of this mean reversion. A higher k value indicates a stronger pull towards the mean.

1.3 Calculating the Lagrangian

Let's assume $k = 10.0$ and $m = 1.0$.

```
// Function to calculate the Lagrangian
double CalculateLagrangian(double velocity, double price, double meanPrice)
{
    double kineticEnergy = 0.5 * m * MathPow(velocity, 2);
    double potentialEnergy = CalculatePotentialEnergy(price, meanPrice);
    return kineticEnergy - potentialEnergy;
}
```

1.4 Calculating price velocity and price acceleration

```
// Function to calculate velocity and acceleration of the price
void CalculatePriceDynamics(double &velocity, double &acceleration)
{
    static double previousPrice = 0.0;
    static double previousVelocity = 0.0;

    // Get current price (close of the current bar)
    double currentPrice = iClose(_Symbol, PERIOD_CURRENT, 0);

    // Calculate velocity (rate of change of price)
    velocity = (currentPrice - previousPrice) / PeriodSeconds();
    velocity *= 1e5; // Scale up to match realistic price movements

    // Calculate acceleration (rate of change of velocity)
    acceleration = (velocity - previousVelocity) / PeriodSeconds();

    // Update the previous price and velocity for the next calculation
    previousPrice = currentPrice;
    previousVelocity = velocity;
}
```

1.5 Buy Conditions: Optimal Entry Points

The trading algorithm uses the calculated Lagrangian to identify optimal entry points for buy orders. These orders are triggered when the Lagrangian is positive, indicating upward momentum, and the velocity exceeds a predefined threshold.

```
// Buy condition
double lagrangianThreshold = -500.0; // Allow slightly negative values
if (lagrangian > lagrangianThreshold && velocity > 0.0001)
{
    // Open buy order
}
```

- $L > 0$: Indicates that the kinetic energy exceeds the potential energy, suggesting upward momentum.
- $v > 0.0001$: Ensures that the velocity is sufficiently high to confirm momentum.