Project 1C: Using Optimization to Trade Stocks Quantitatively and Algorithmically

CEE 251L. Uncertainty, Design, and Optimization

MATLAB & Python

Professor Henri P. Gavin

Department of Civil & Environmental Engineering

Duke University

Ella Veysel & Felipe Jarmaillo

Interdisciplinary Engineering and Applied Sciences (IDEAS)

I. Introduction

This project involves the design, testing, and optimization of an automated stock trading algorithm aimed at maximizing cumulative returns. Using historical data, we will develop a strategy for systematically executing buy and sell decisions over a 200-day period of real stock closing prices. Once optimized, the algorithm will be evaluated on a subsequent 200-day out-of-sample dataset. Due to the high volatility and low autocorrelation of daily price changes, identifying short-term trends through raw prices can be challenging.

To mitigate noise and capture underlying momentum or reversal patterns, groups are encouraged to implement smoothing techniques, such as a weighted moving average (WMA) over the most recent (N) days, to estimate price trends. The smoothed price series and their derivatives (e.g., rate of change) can then serve as signals for trade execution.

ORSopt (Optimized Step-Size Random Search)

- Exploration-heavy random method
- Good for global search
- Non-deterministic unless you use rng(seed)
- Handles constraints with penalty methods

Summary:

We selected ORSopt for its ability to effectively explore complex, high-dimensional parameter spaces and locate global optima. While NMAopt is efficient for fine-tuning and local optimization, it struggles with global exploration due to its sensitivity to the dimensionality of data (NMA is a geometric method that uses n-sized simplex). SQPopt, although precise for smooth problems, is unreliable for our stochastic and rugged optimization landscape. ORSopt's adaptability, resilience to local minima, and robustness against non-smooth objective functions makes it the most appropriate choice for our optimization of the algorithmic trading strategy.

II. Financial Data Analysis

II.a. The Quality of a Stock

The data analysis in this project comes in two parts. The Q equation (quality), as designed by Professor Henri P. Gavin, was done through a mathematical investigation of how stocks behave and how we can attempt to quantify the decision of buying/selling a stock:

$$\mathcal{Q}_{d,s} = q_1 \; \dot{\bar{p}}_{d,s} + q_2 \; \ddot{\bar{p}}_{d,s} + q_3 \; v_{d,s}$$

It represents the decision to buy and sell stock based on the Q statistic, which we shall call "quality" Q. Stocks with higher quality are better candidates for purchase. Stocks with lower quality are better candidates for sale. A measure of the quality of stock s on day d (Q_d,s) could be a weighted sum of the fractional change in a stock price, the rate of the fractional change of a stock price, and the volatility of a stock price over a recent period of time.

where

- q_1, q_2, q_3 are weighting coefficients with values to be found via optimization.
- $\bar{p}_{d,s}$ is a weighted average of the price of stock s on day d,

$$\bar{p}_{d,s} = \sum_{i=1}^{N} w_i \; p_{(d-i),s} \tag{2}$$

where the weights w_i emphasize more recent data more significantly, and the sum of the weights is 1.

$$w_i = \exp\left[-\frac{5i}{N}\right] \; \left(\frac{1}{\sum_{i=1}^{N} \exp[-5i/N]}\right). \tag{3}$$

$\dot{\bar{p}}_{d,s}$ is the fractional change of a weighted average of the price of stock s on day d,

$$\dot{\bar{p}}_{d,s} = (\bar{p}_{d,s} - \bar{p}_{(d-2),s}) \; / \; (\bar{p}_{d,s} + \bar{p}_{(d-2),s}) \tag{4}$$

$\ddot{\bar{p}}_{d,s}$ is the rate of the fractional change of the weighted average of stock s on day d,

$$\ddot{\bar{p}}_{d,s} = (\dot{\bar{p}}_{d,s} - \dot{\bar{p}}_{(d-2),s}) \; / \; 2 \tag{5}$$

$v_{d,s}$ is the volatility of stock s on day d defined here as the standard deviation of the stock price over the previous N days, as a fraction of the weighted average stock price, $\bar{p}_{d,s}$,

$$v_{d,s} = \frac{1}{\bar{p}_{d,s}} \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} \left[p_{(d-i),s} - \left(\frac{1}{N} \sum_{j=1}^{N} p_{(d-j),s} \right) \right]^2} \; . \tag{6}$$

$v_{d,s}$ is like a running coefficient of variation over the last N days of data.

II.b. The Analysis of Data Provided

We tuned and optimized Professor Gavin's *exchange_analysis.m* algorithm to build a quantitative strategy that executes trading decisions.

Using **lognormal distribution** properties, **hypothesis testing**, **probability density functions, standard scores**, **ordinary least squares regression**, and **Gaussian smoothing** of parameter surfaces (this part was done in MATLAB), we made the final decision on the parameters and their optimal values.

II.b.c Our Jupyter Notebook (this is where our pre-model-building data is, see Appendix D).

Most of our subsequent data analysis (ETL and EDA), as well as the foundation steps that determine the modifications for the exchange_analysis.m, was conducted in a Jupyter notebook using Python. Our Jupyter Notebook can be found in **Appendix D**, with full data analysis provided with the code.

II.d. Reproducing Figure 2 for Stock Number 5

To study the method of weighted averages used in this project, Figure 2 of the assignment document must be reproduced for stock number 5. This will help analyze the effects of the window size N on these metrics.

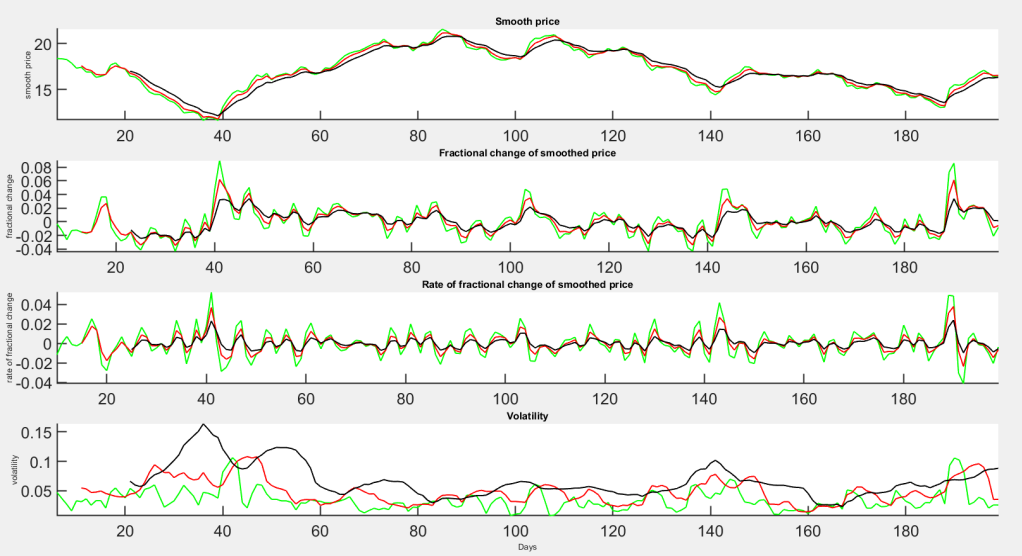


Figure 1: Effect of the number of trading days *N* involved in the weighted average of stock prices for Stock #5

on the smoothed price, the fractional change of smoothed price, the rate of fractional change of smoothed price, and volatility. The curves represent $N = 5$ (green), $N = 10$ (red), and $N = 20$ (black).

Observation: Increasing N leads to a smoother $\bar{p}$ but reduces responsiveness to short-term fluctuations. Higher N values dampen $\dot{\bar{p}}$ and $\ddot{\bar{p}}$ fluctuations, making them less sensitive to sudden changes. Volatility is reduced with a larger N since rapid price fluctuations are averaged out.

III. Stock Trading Decision Rules

Core Inputs (parameters):

Parameter		Description
$p(1) = N$	*	Window size for exponential smoothing (velocity, acceleration, volatility)
$p(2:4) = q$	*	Coefficients for slope, curvature, and volatility in quality computation
$p(5) = fc$	*	Fraction of cash to invest (0–1)
$p(6) = B$	*	Initial buy threshold (quality-based)
$p(7) = S$	*	Initial sell threshold (quality-based)
$p(8) = W$	*	New: Window size for Weighted Moving Average (WMA) crossover rule

Decision Logic Enhancements (in addition to the Q "quality" score)

Modification 1. Weighted Moving Average (WMA) Crossover Indicator

The purpose of this enhancement is to improve signal confirmation using a trend-following logic. The algorithm's buy condition is to only consider buying stocks if they are above their WMA. Conversely, the algorithm's sell condition is to only sell if the stock is below its WMA. Finally, we parameterized W (WMA) by using an optimized random search process,

Modification 2. Pairs Trading Signal for stocks 6 & 11)

We implemented a z-score-based logic derived from the regression of logarithmic returns from stock 6 and those of stock 11. The mathematics and logic behind this modification, as well as the research done to implement it, are included in our Jupyter notebook.

Our **entry condition** is: if z-score < -1.3, buy both stocks (this is conditional on stocks not being owned and being above their WMA). Conversely, our **exit condition** is: if z-score > -0.25, sell both if currently held. Note that once our stock's z-score goes below the -0.25 threshold, it indicates that the spread between its correlated assets is converging, and therefore, a good moment to close our previously opened position (which was initiated while the z-score was away from the converging threshold). This ensures a mean-reversion style of entry/exit, while still obeying no-shorting constraints.

Modification 3. Risk Management via Drawdown Protection

- This third enhancement monitors the 3-day smoothed portfolio drawdown. It works in the following way:
- A trigger signal occurs if the drawdown exceeds 12% for 2 consecutive days, starting after day 48.
- If triggered, the algorithm responds to liquidate all positions and dynamically tighten thresholds (sets B = 0.01, S = -0.01) to re-enter conservatively.

Stock Quality Definition

Unchanged from the original Q definition.

$$quality = q1 * velocity + q2 * acceleration + q3 * volatility;$$

Portfolio and Visualization

$$Portfolio\ value = cash + stock\ holdings.$$

Drawdown is computed using a rolling max over the previous 3 days. This is to solve (debug) the conflict where any transaction spikes the drawdown, providing an unrealistic perspective of compromised value. (See Appendix A)

When the original *exchange_analysis.m* code transacts, the cash proceeds from selling a stock are realized on day *i*, but the investment change that replaces such stock is shown on day *i+1*. This results in a value for drawdown that is not accurate. Therefore, to account for this, we implemented a drawdown metric that allowed us to compute drawdown (compromised portfolio %) if such a

value continued to be compromised for at least 3 days, ensuring the realistic value was reflected (mathematically and visually).

One risk, however, with this measure is that if our portfolio value is compromised entirely in a single trading period (1 day), our drawdown will only reflect it on day i+2.

This trading algorithm integrates traditional quality-based momentum with WMA crossover confirmation, statistical arbitrage via pairs trading, and a risk-managed drawdown liquidation mechanism. The addition of the WMA window (W) as an optimizable parameter and real-time monitoring of drawdown thresholds introduces dynamic risk control and market adaptability.

Transaction Cost for Realistic Simulations

Each time Ms shares of stock s are sold, the cash in hand, C, to purchase stocks is increased by the current value of the shares. A transaction cost of two dollars is paid for every stock sold or bought.

$$C_{d+1} = C_d + \sum_{s \in \text{sold}} p_{s,d} M_s - 2.00 \left(\sum_{s \in \text{sold}} 1 \right) - 2.00 \left(\sum_{s \in \text{bought}} 1 \right)$$

IV. Part 4 – Initial Guesses, Sensitivity Analysis, and Inferences (Report)

4.a. exchange_opt.m displaying initial guess and subsequent result:

```
x0 = [20, 0.40, 0.40, 0.2, 0.25, 10, -10, 20]';   % <-- added W = 10
LB  = [5, -1, -1,  0, 0.01, 0.001, -0.1, 4]';      % reasonable lower bound for W
UB  = [100, 1, 1,  1, 0.99, 0.1,    0,   30]';     % upper bound for W


% ORSopt requires func(x, consts) — wrap exchange_analysis accordingly
wrapped_fun = @(x, consts) exchange_analysis(x, 0);  % no consts used, just pass dummy

% ORSopt options: [verbosity, tol_x, tol_f, tol_g, max_evals, penalty, exponent, max_samples, f_cov, stop_if_feasible]
options = [1, 1e-3, 1e-3, 1e-3, 200, 100, 2, 1, 0.1, 0];  % customize if needed

% Dummy consts (not used in function, but ORSopt requires it)
consts = [];

% Call ORSopt
[x_opt, f_opt, g_opt, cvg_hst] = ORSopt(wrapped_fun, x0, LB, UB, options, consts);
```

```
-+-+-+-+-+-+-+-+-+-+- ORSopt -+-+-+-+-+-+-+-+-+-+-+-+-+
iteration               =    10     ***  feasible  ***
function evaluations    =    57  of    200  (28.5%)
e.t.a.                  =  8:51:43 PM
objective               =  -1.447e+03
variables               =   1.633e+01  2.575e-01  1.493e-01  2.416e-01  3.388e-01  9.462e-02 -9.158e-02  2.074e+0
max constraint          =  -1.000e+00 (1)
Convergence Criterion F =  4.6635e-04    tolF = 0.001000
Convergence Criterion X =  3.3941e-02    tolX = 0.001000
c.o.v. of f_a  =   0.000e+00
-+-+-+-+-+-+-+-+-+-+- ORSopt -+-+-+-+-+-+-+-+-+-+-+-+-+
*** Woo-Hoo!  Converged solution found!
*** convergence in design objective
*** Woo-Hoo!  Converged solution is feasible!
*** Completion  :  8:51:38 PM (00:00:01)
*** Objective   :  -1.447e+03
*** Variables   :
              x_init         x_lb     <     x_opt    <     x_ub
------------------------------------------------------------------
x(  1)      20.00000       5.00000       16.33354      100.00000
x(  2)       0.40000      -1.00000        0.25754        1.00000
x(  3)       0.40000      -1.00000        0.14929        1.00000
x(  4)       0.20000       0.00000        0.24159        1.00000
x(  5)       0.25000       0.01000        0.33878        0.99000
x(  6)       0.10000       0.00100        0.09462        0.10000
x(  7)      -0.10000      -0.10000       -0.09158        0.00000
x(  8)      20.00000       4.00000       20.74083       30.00000
 *** Constraints :
       g(  1) =     -1.00000
Optimal parameters:
   16.3335    0.2575    0.1493    0.2416    0.3388    0.0946   -0.0916   20.7408

Day 122: BUY signal (pairs logic) for stock 11 due to z = -1.42
```

4.b.

Optimizing with this algorithm minimizes the objective function; therefore, the value of the portfolio is negated, and the starting capital is $1000.00 USD. A value of $1700.00 USD represents a return of 70% (over training data).

Initial Guess (x0)	Final Cost
[20, 0.4, 0.4, 0.2, 0.5, 10, -10, 20]	-885.1061
[20, 0.1, 0.1, 0.1, 0.5, 0.02, -0.02, 10]	-2414.2805
[20, 0.50, 0.25, 0.25, 0.25, 0.10, -0.10, 20]	-1575.9455
[20, 0.33, 0.33, 0.33, 0.25, 0.10, -0.10, 20]	-1711.323
[20, -0.40, 0.40, 0.20, 0.25, 0.10, -0.10, 20]	-1266.1315
[20, 0.40, -0.40, 0.20, 0.25, 0.10, -0.10, 20]	-1717.7738
[20, 0.40, 0.40, -0.20, 0.25, 0.10, -0.10, 20]	-1181.6737

Figure 2: Effect of Initial Guesses on Final Portfolio Value for Stock Trading Optimization. The

curves represent different initial guesses used for the parameters x_0 = [N, q1, q2, q3, fc, B, S, W]. Each line corresponds to a unique set of initial parameters as detailed in the analysis, demonstrating the sensitivity of final portfolio value to initial conditions.

Insights – The final cost is highly sensitive to the initial guess provided. For example:

- The best performing guess was [20, 0.1, 0.1, 0.1, 0.5, 0.02, -0.02, 10] with a final cost of -2414.2805.
- The worst performing guess was [20, 0.4, 0.4, 0.2, 0.5, 10, -10, 20] with a final cost of -885.1061.

This sensitivity suggests that the optimization landscape has many peaks and valleys, indicating a non-convex, non-smooth problem. The initial guess significantly affects the outcome, implying that relying on a single guess is insufficient. Robustness must be improved through sensitivity analysis.

4.c. Sensitivity Analysis, 5% Parameter Perturbation, and Interpretation

The optimal parameter values are robust to initial guesses, and the earnings outcome is consistently strong regardless of modifying starting values. This shows that although our optimal parameters landscape is full of non-differentiable peaks, the optimization algorithm and the stability of results are found on a robust global optimum.

To evaluate the sensitivity of the trading strategy to each individual parameter, we ran 8 separate simulations (shown above). In each run, we increased only one parameter at a time by +5% from its optimal value, while keeping all other parameters fixed at their optimal values. We will be using the initial guess that resulted in the best final cost (=-2414.2805). The most negative final cost means this initial guess gave the best performance, so this is the optimal baseline for the sensitivity analysis.

Parameter	Perturbed Value	Final Cost	Change in Earnings
N	21	-2296.1072	Significant improvement
q1	0.105	-1901.1273	Moderate improvement
q2	0.105	-2108.0856	Slight improvement
q3	0.105	-2132.3394	Slight improvement
fc	0.525	-1849.9291	Moderate improvement
B	0.021	-2050.8903	Slight improvement
S	-0.021	-2053.7712	Slight improvement
W	10.5	-2196.546	Significant improvement

Figure 2: Sensitivity Analysis Results - Final Costs for Each Parameter Perturbation

This table summarizes the results of the sensitivity analysis, where each parameter was perturbed by +5% from its original value. The resulting final costs and changes in earnings provide insight into the sensitivity of the trading strategy to individual parameters.

- The most sensitive parameters appear to be N and W since their perturbations resulted in the most significant changes in the final cost.
- Changes in q1 and fc also yielded noticeable improvements, while q2, q3, B, and S showed only slight improvements.
- Further testing with negative perturbations or combinations of perturbations could help refine the sensitivity analysis.

4.d.

When an optimal solution shows:

- **$q_1 < 0$**: Negative velocity coefficient indicates a preference for certain stocks that have decreased in value. This suggests a strategy aimed at **buying low** or taking advantage of

downward trends for future upward momentum.

- **q2 > 0**: Positive acceleration coefficient implies favoring stocks with increasing momentum. It shows that the algorithm is seeking **emerging trends or rapidly accelerating stocks**, indicating potential growth.

- **q3 > 0**: Positive volatility coefficient suggests a **tolerance or preference for more volatile stocks**, likely under the hypothesis that higher volatility could lead to higher returns.

Overall, this combination of parameters indicates an **investment strategy targeting emerging momentum in volatile environments**. It seeks opportunities in stocks experiencing rapid changes while avoiding crowded trades that have already moved too far.

V. Optimization and Modifications

We propose several strategic enhancements to improve risk management and trading logic. First, we aim to optimize the maximum drawdown by triggering a portfolio liquidation and altering strategy logic when a critical threshold is breached. Additionally, we can leverage the correlation of returns to anticipate movements, either reinforcing signals for highly correlated pairs or applying inverse correlation logic. If correlation-based signals underperform, we transition to a pairs trading approach. Lastly, we will explore global minima in strategy performance per stock and optimize quality weights (q1, q2, q3) to identify robust parameter regions across different assets.

Recommended Set of Parameters:

param = [27.0235; -0.6170; 0.3793; 0.3103; 0.7015; 0.0418; -0.0086; 4.6282];

param = [N ; q1 ; q2 ; q3 ; fc ; B ; S ; WMA];

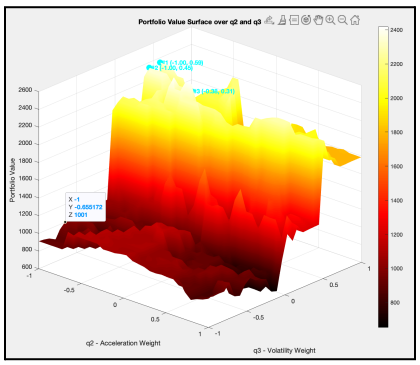
Trials after localizing robust parameters:

Top 3 Robust q2/q3 combos (smoothed):

#2: q2 = -0.3793, q3 = 0.2414, Value = 2227.89

#3: q2 = -0.4483, q3 = 0.3103, Value = 2216.10

#4: q2 = -0.3793, q3 = 0.3103, Value = 2332.39



We recognized that the parameter space is high-dimensional and cannot be fully visualized. To better understand the optimization landscape, we fixed the most optimal value of q_1 (based on several optimization trials) and plotted portfolio value (z-axis) across a grid of q_2 and q_3. While multiple peaks indicated the potential for high returns, we focused on identifying a smooth and robust global optimum to avoid overfitting. The resulting surface, visualized in both a lightly smoothed and a more smoothed version using a Gaussian filter, represents the optimization landscape of parameter values.

This script is called *quality_surface_plot*, and it performs a **2D grid search** over quality weights q_2 (acceleration) and q_3 (volatility) to evaluate their impact on **portfolio performance** using a fixed trading strategy. For each parameter pair, it computes the **portfolio value** via *exchange_analysis.m,* storing results in a surface matrix. The surface is then **smoothed with a Gaussian filter** to reduce local noise and emphasize general trends. A Gaussian filter smooths data by averaging neighboring values with weights from a Gaussian distribution, reducing noise while preserving overall structure. A **robustness-adjusted value** is computed by penalizing local volatility using a standard deviation-based metric. Finally, the top 3 most **robust and high-performing (q_2, q_3) combinations** are identified and visualized on a 3D surface plot to aid in selecting stable, high-return strategy parameters.

The changes our group made and executed in *exchange_analysis.m* are discussed above, and the script is provided in the appendix. The maximum amount of money we were able to make was the following:

```
maxSharesOwned =

  122.9816

Final Portfolio Value: 2411.43
```

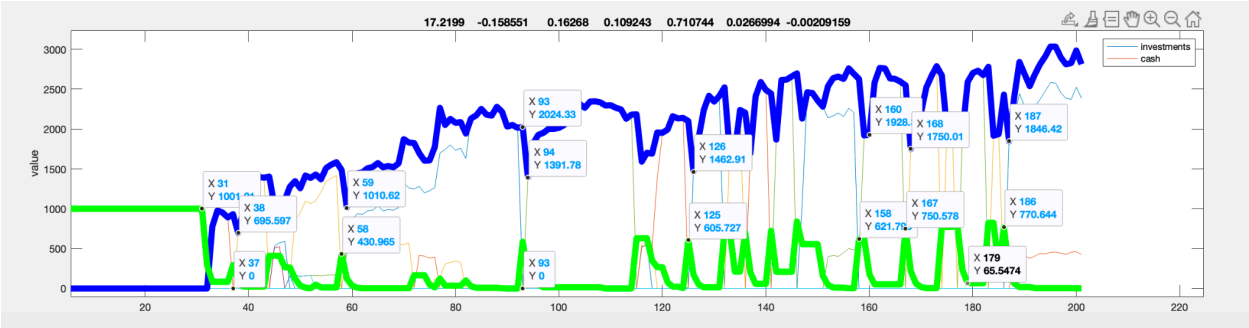
The final portfolio value of $2411.43 represents a net gain of $1411.43 over the initial investment of $1000, achieved over 200 days. This equates to a return of approximately 141.14% over the given trading period.

A table of our final optimized design parameter values:

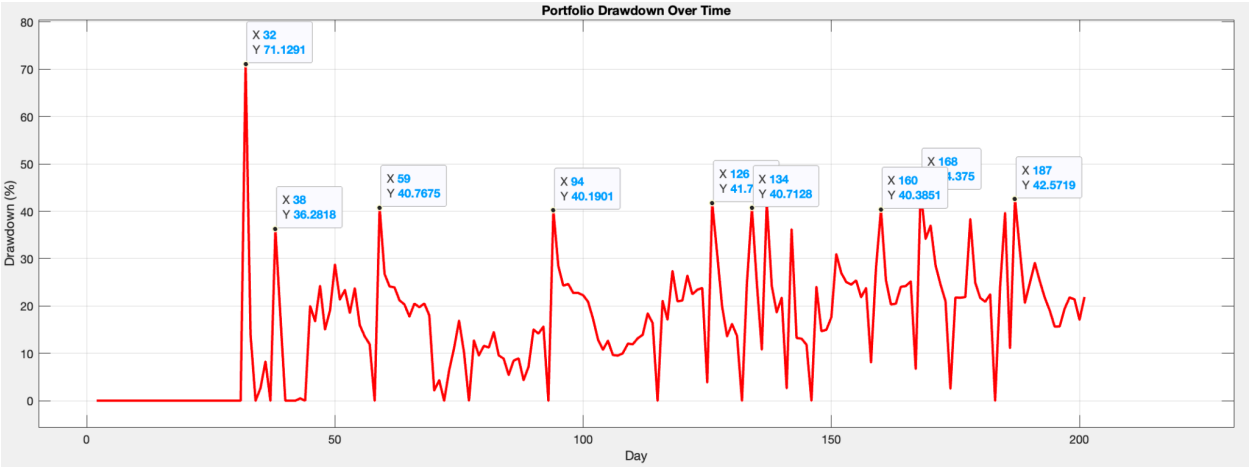
Appendices

Appendix A: Problem with Algorithmic Calculation of Drawdown per Day and how we fixed it.

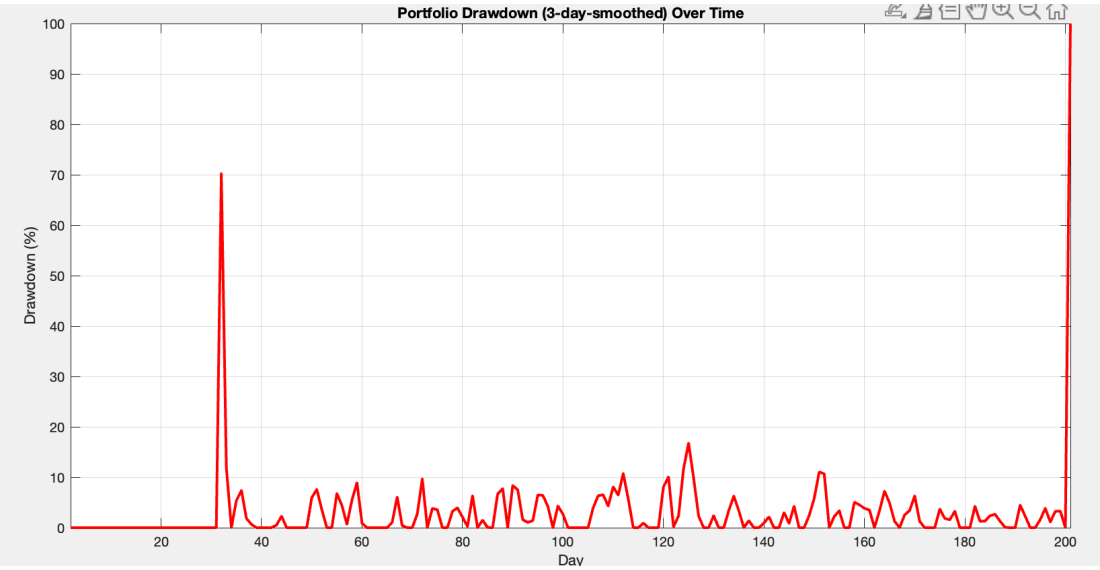
The cash component of transactions occurs on day i, the investment component is reflected in day i+1.



Observe, this creates unrealistic peaks for the value of our drawdown (i.e the percent of our portfolio compromised any given day.



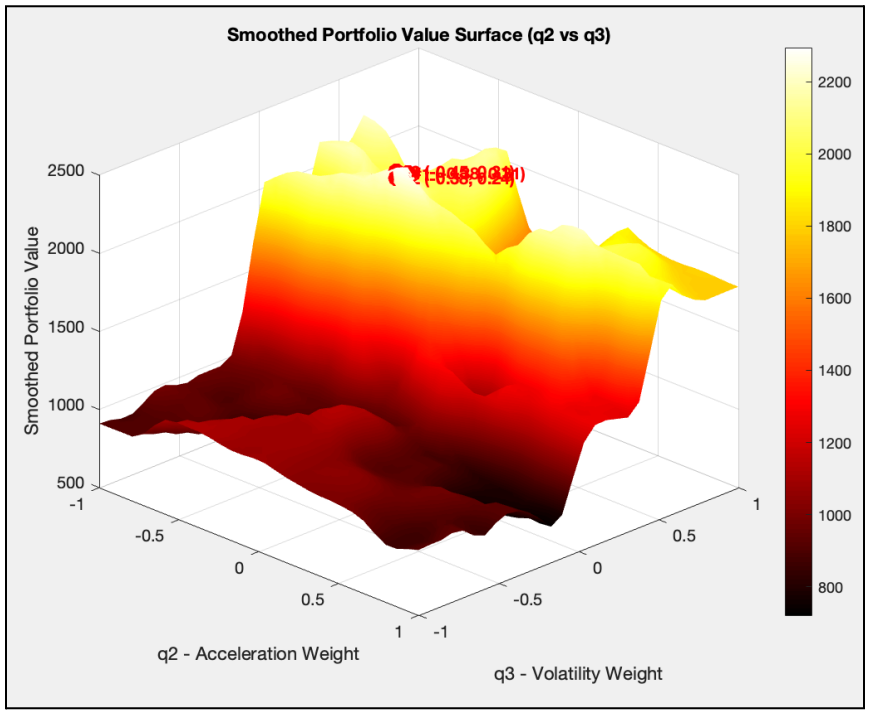
We fixed this by plotting the drawdown of every simulation as smoothed over a two-day period. Therefore, any drawdown percent must have lasted for at least n=2 periods (days) to be reflected on our graph. This allowed us to implement a risk management measure that triggers an order to liquidate our portfolio if more than 25% of it is compromised.



Final Strategy (Optimized Parameters and Added Logic) Portfolio Drawdown. Note, the algorithm doesn't make any trades until day 30, where it has learned enough information required for its parameters. The peaks on this day and the final day (200) are caused to to entering investments initially and liquidation, respectively.

Appendix B: Surface Plot (smoothed) to find Robust Global Minima (Maxima)

Smoothed using Gaussian Filter:



Non-smoothed: