Renganathan Subramanian
Team: I_need_a_macbook

JPMC Derivative Modelling

Contents

1 Question 1

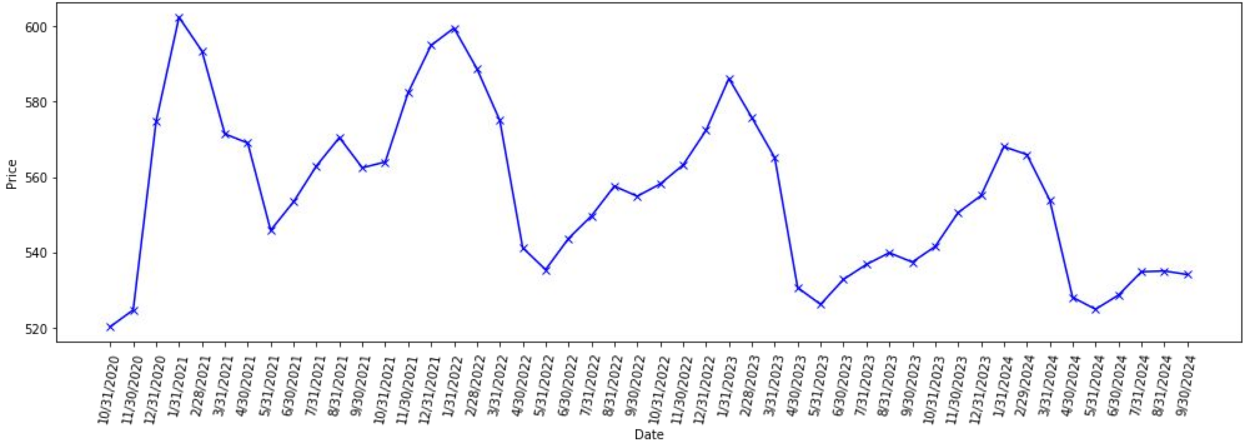
2 Question 4

3 Question 5

Question 1

1.1. Method Used
1.2. ‘Over’-Extrapolation
1.3. Other Methods

Introductory Analysis



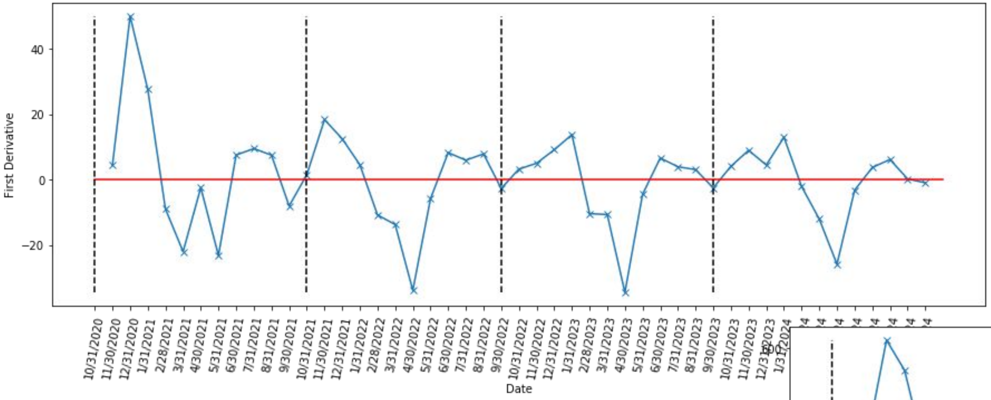
	Date	Price
0	10/31/2020	520.349403
1	11/30/2020	524.764215
2	12/31/2020	574.740259
3	1/31/2021	602.355246
4	2/28/2021	593.415544
5	3/31/2021	571.447877
6	4/30/2021	569.019001

Key Observations

- Seasonality
- Trend

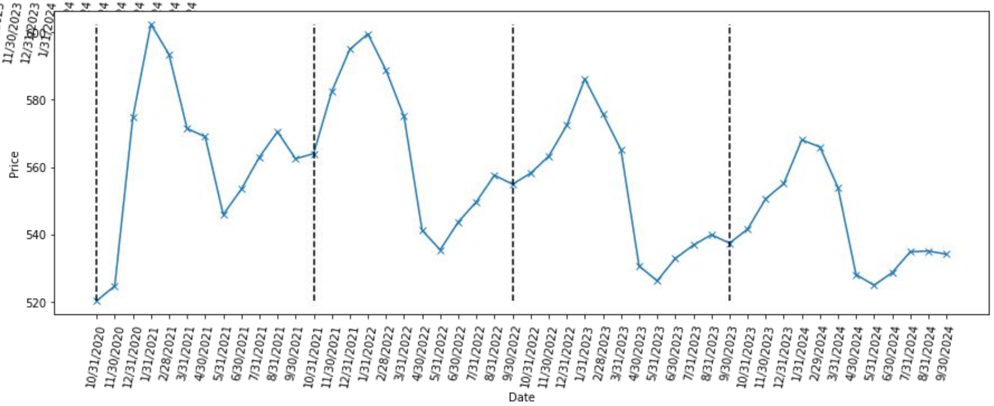
Identifying Seasonality

First Derivative



Identified Seasonality
= **12 months**

Data split into seasons

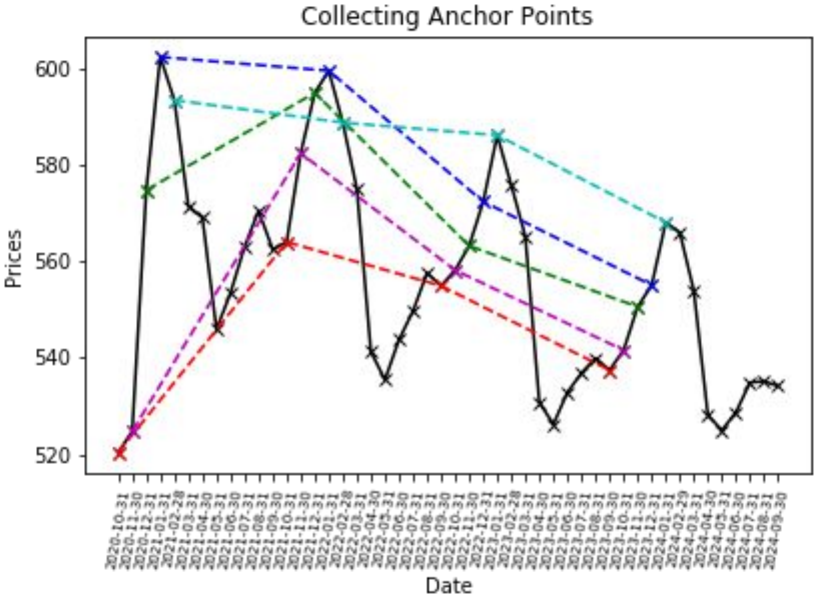


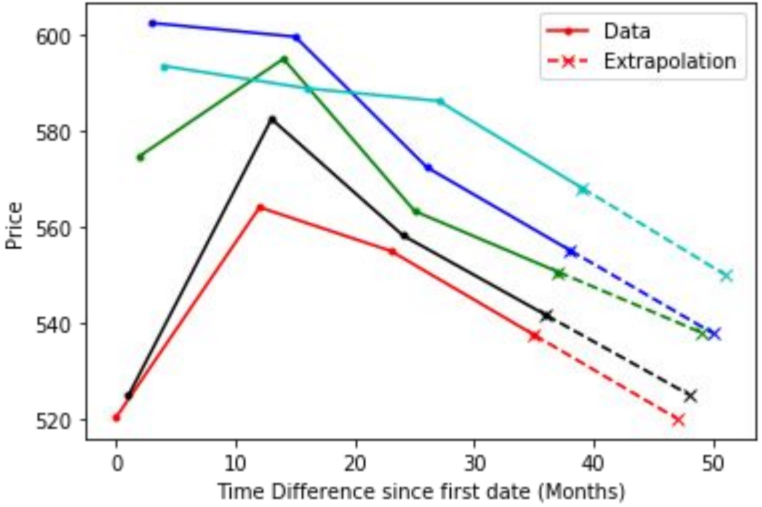
Method Adopted

- First Euler backward difference
- Identify zero difference

Extrapolation

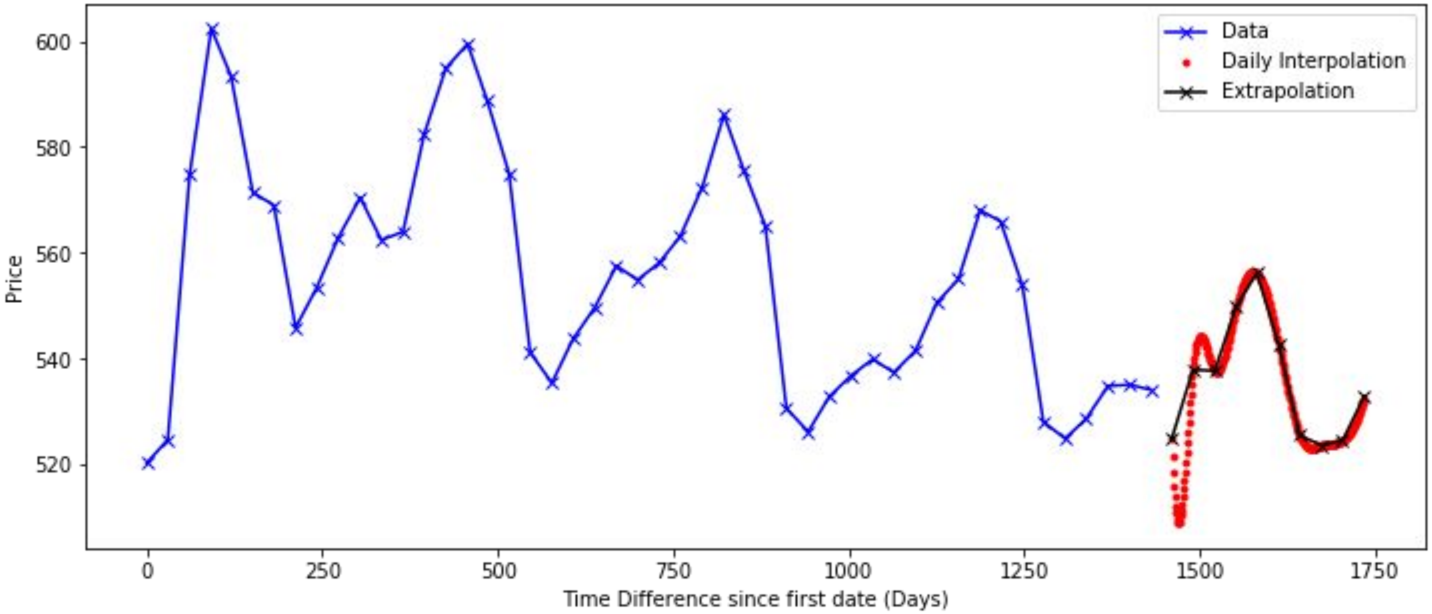
Treat group of anchor points as a dataset and extrapolate after 12 months



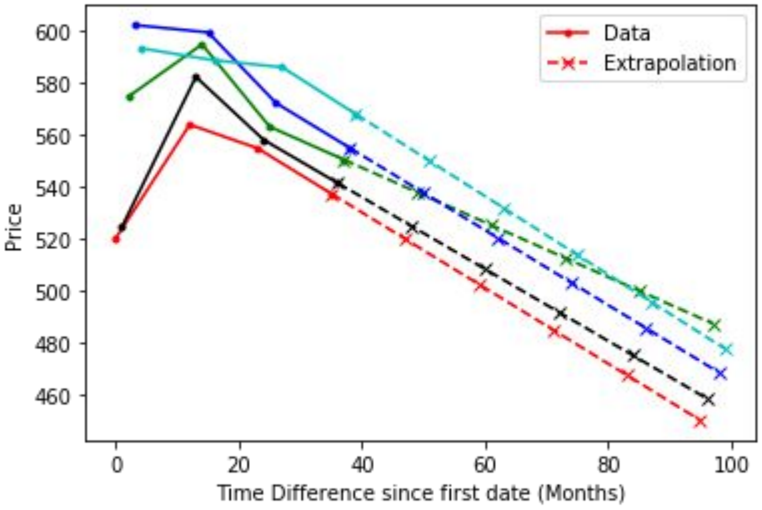


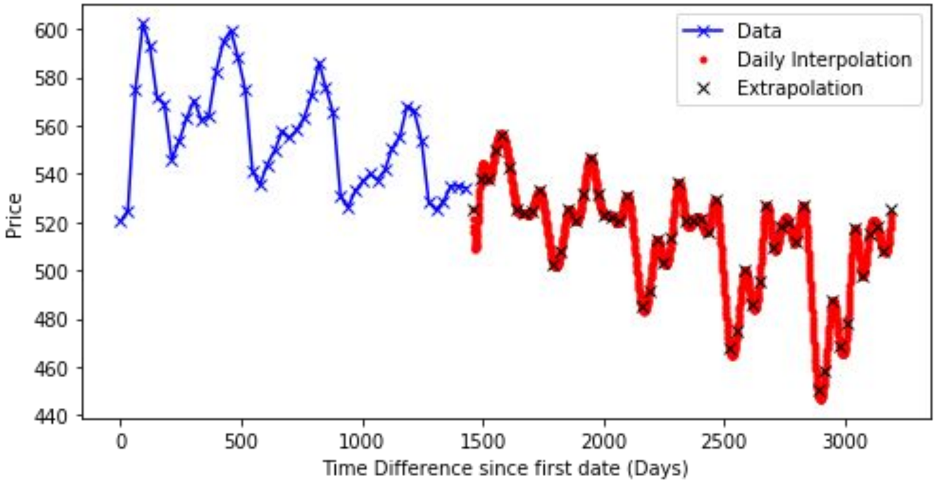
Interpolation

Use union of data and extrapolation for cubic spline interpolation



'Over'-Extrapolation





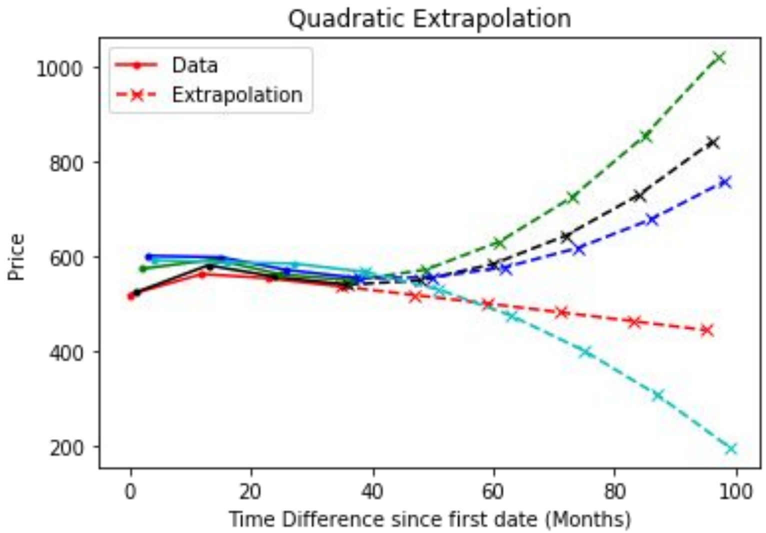
Problems

- Assume linear trend (Reality → Saturation)
- Observation → Decreasing variance; Prediction → Increasing Variance
- Within period pattern (unimodality) not maintained

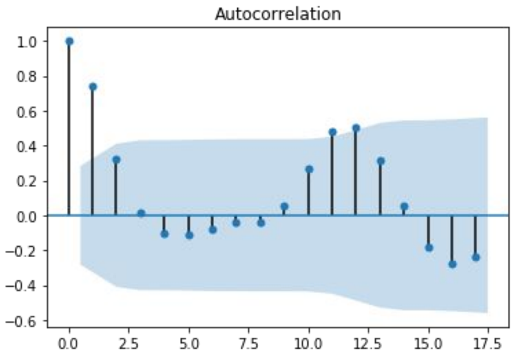
‘Over’-Extrapolation

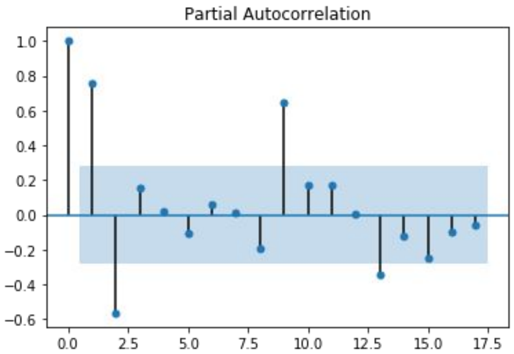
Limitations

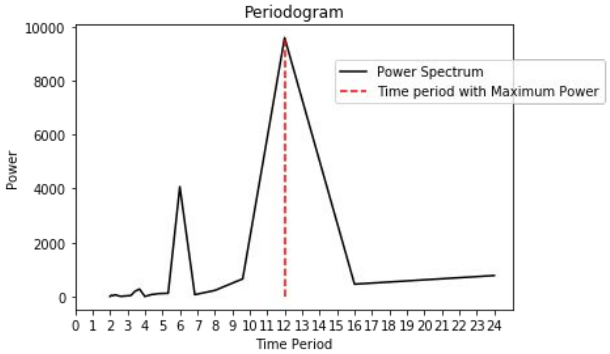
- Non-parametric approach → Not suited for outside distribution uages
- Linear too simple (but higher order overfits)
- Treating each group independently → Reduces data size, misses information present in ‘whole’
- Period calculation is visual-based



Time Series-Based Approach

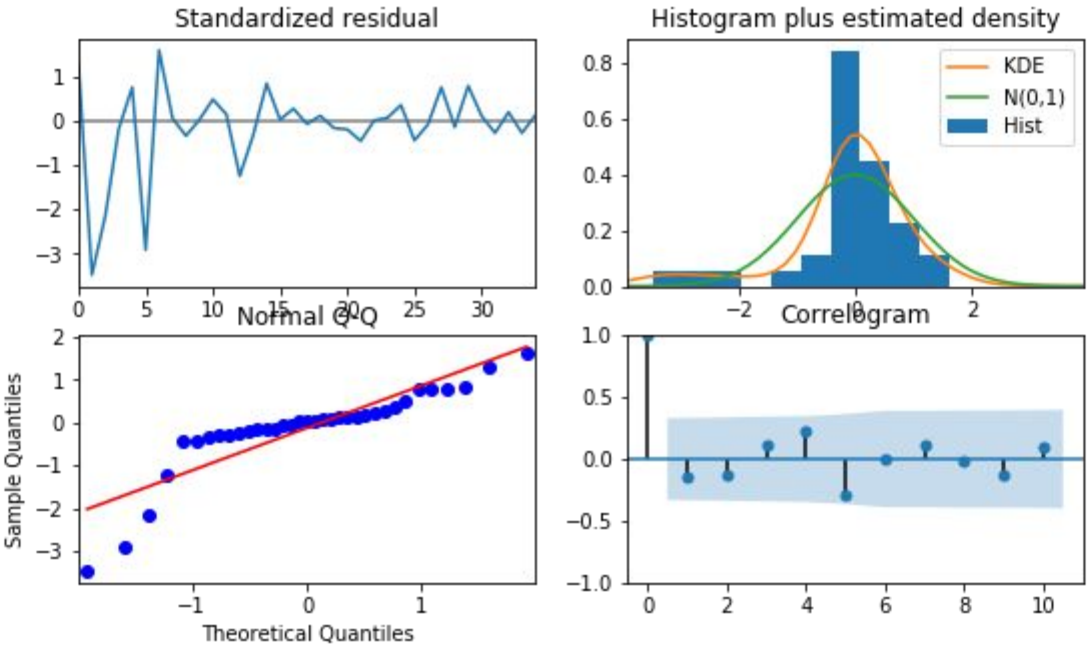






Seasonal Integrating effect present (Not trend)
Seasonality with period=**12 months** (Non visual)

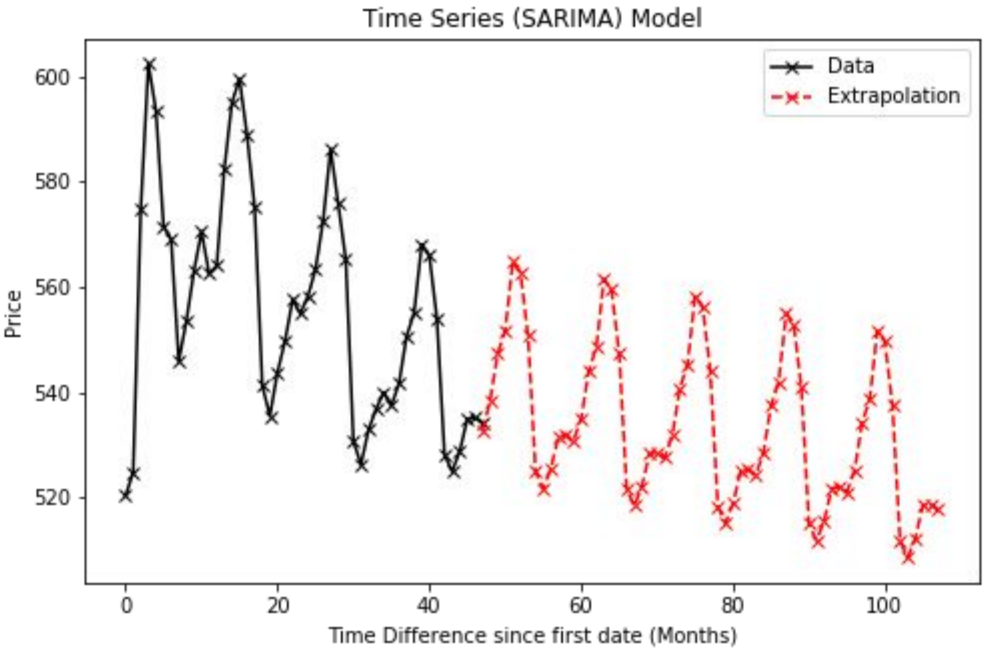
Time Series-Based Approach



SARIMA(0,1,0)(0,1,0)12

- Residuals pass ADF test
- Normal assumption of residuals failed

Time Series-Based Approach



Key Points

- Use all data points together
- Model-based extrapolation
- Statistical tests to verify goodness

What if Idiosyncratic Data?

- Any time series data can be modelled (reasonably) using either multiplicative or additive SARIMA models (under certain assumptions)
- Provides a standard way to **identify model parameters based on ACF, PACF, Periodograms** (thus, easy to tune for new data)
- Even if **seasonality absent**, can model with (p, d, q)(0, 0, 0)s
- Provides **measure of goodness** (under certain assumptions)

Question 4

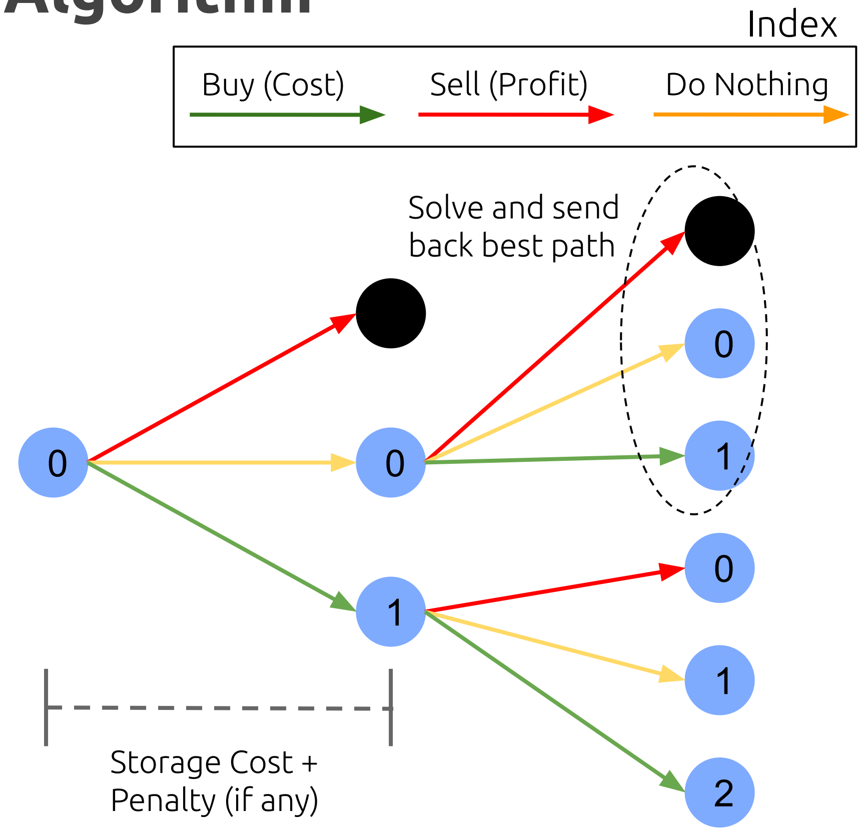
1.1. Naive Method
1.2. Improved Method
1.3. Volume Profile Interpretations
1.4. Parameter Effect Analysis

The Naive Algorithm

Recursive Backtrack Solution

VAL(i, S) = MAX
- VAL(i, S+G) - Buying Cost
- VAL(i, S-G) + Sell Value
- VAL(i, S)

+ - Storage/Penalty Costs



Shortcomings and Observations

Time Complexity: **O(3^n)**

Searching naive non-optimal paths

Overlapping searches

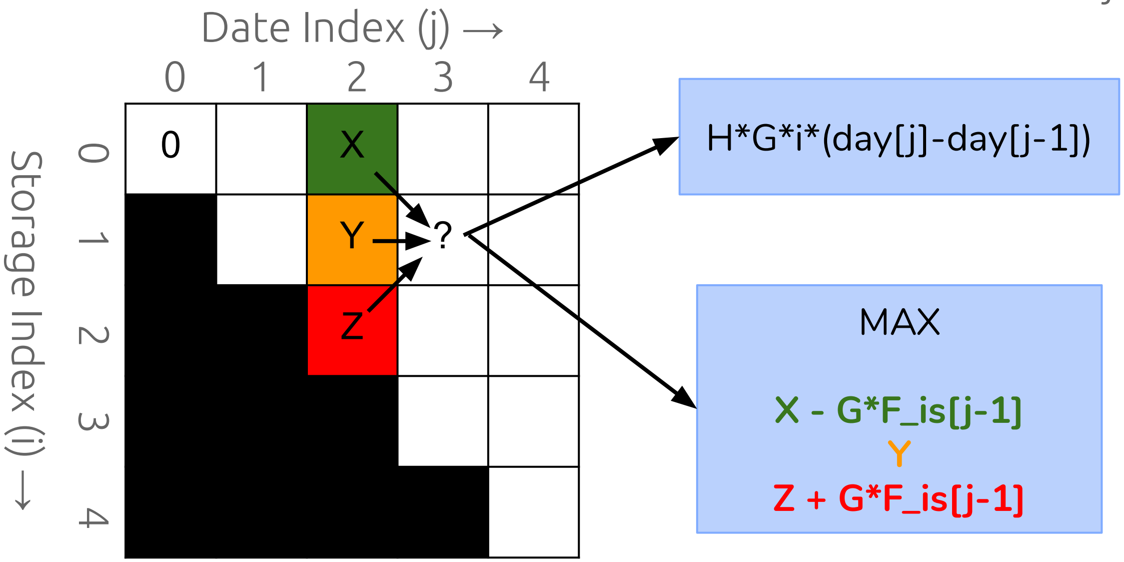
Useful Observations

- Future paths depend only on stock at the day and **not how we got there**
- **Maximum possible stock** on day i = i*(injection rate)
- Tail sub problems exist: **Opti(1 to i+k) = Opti(1 to i) + Opti(i to k)**

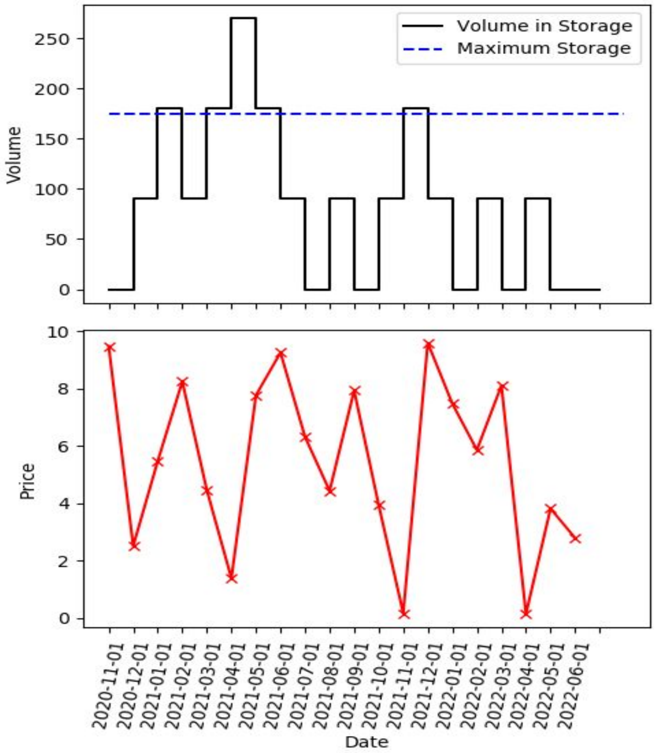
Improved Algorithm

Dynamic Programming with Tabulation

DP[i, j] holds optimal value of trade which leaves stock G*j on day i



Results



Time Complexity: **$O(n^2)$**

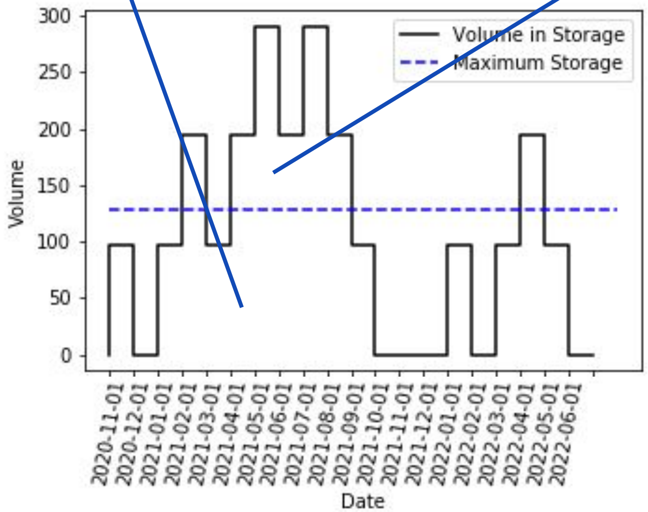
Retain only promising paths on any day

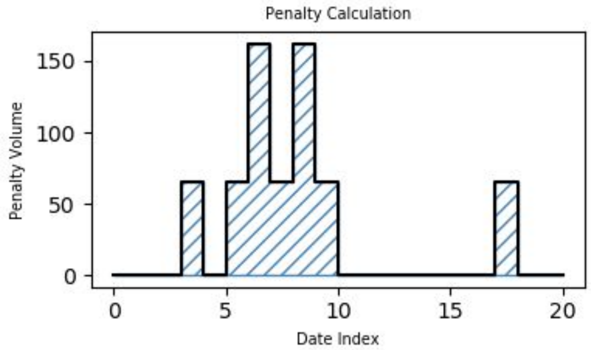
Overlapping subproblems combined

Final Value from Volume Profile

Total Storage Cost

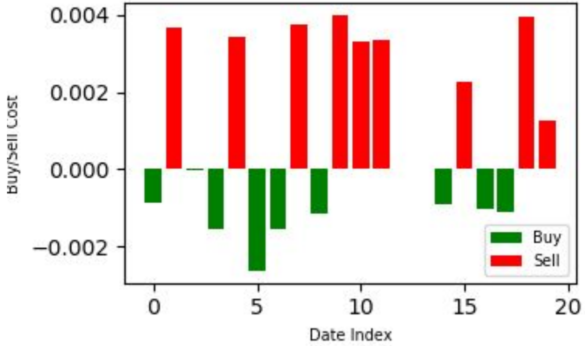
Area under curve
x
Storage Cost





Total Penalty

Area under curve
x
Penalty Cost



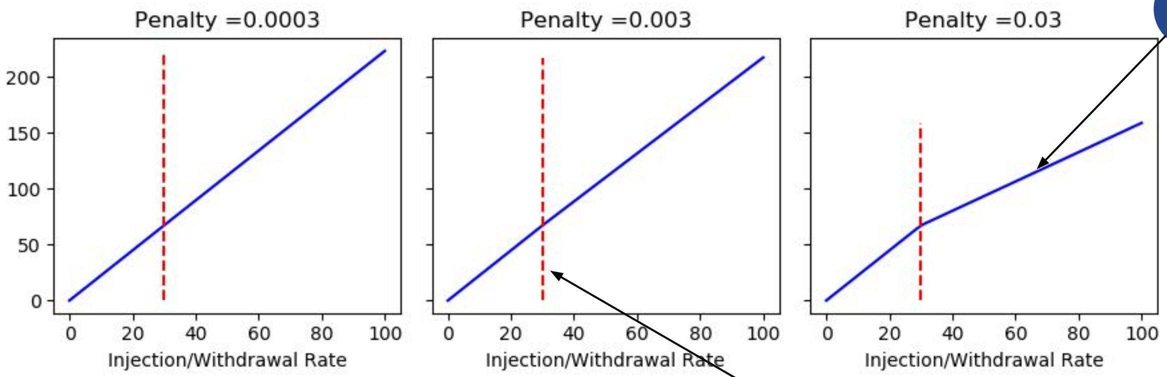
Buy/Sell

Injection Rate
x
(-**Prices**[Vol Inc Points]
+**Prices**[Vol Dec Points])

Effect of Parameters

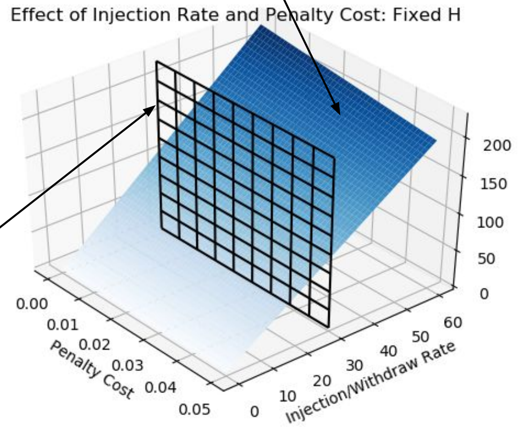
Parameters	Intuitive Effect on Value
Storage Cost	Negative effect
Injection/Withdraw Rate	Positive effect
Max Volume	Positive effect
Excess Penalty Cost	Negative Effect

Penalty, Injection Rate, Max Volume



2

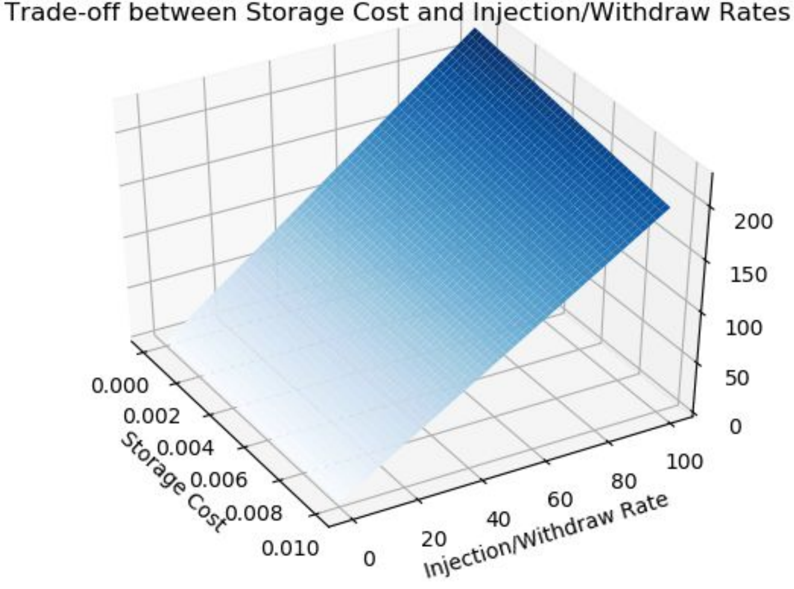
Beyond that, increase in **Penalty** reduces the positive effect of **Injection Rate**



1

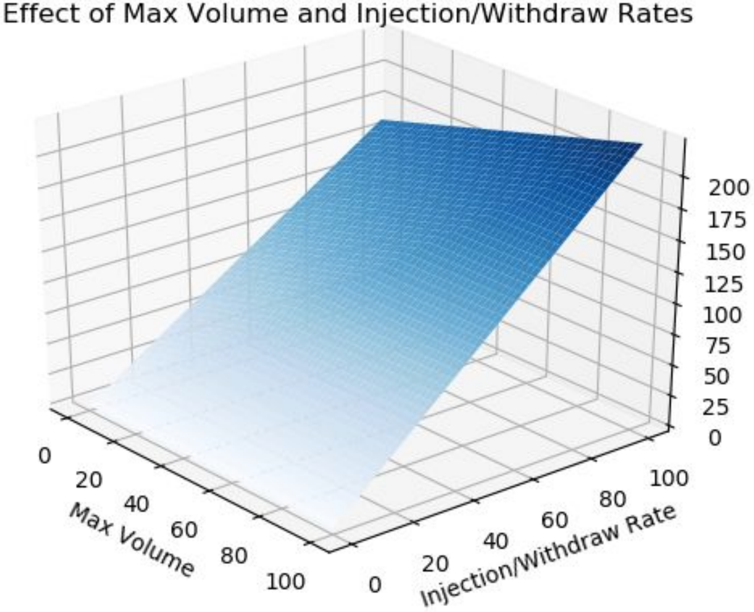
Penalty Cost has no impact on **Value** till **Injection Rate** becomes comparable to **Max Volume**

Effects of Injection Rate



3 Competing effect between **Storage Cost** and **Injection Rate**

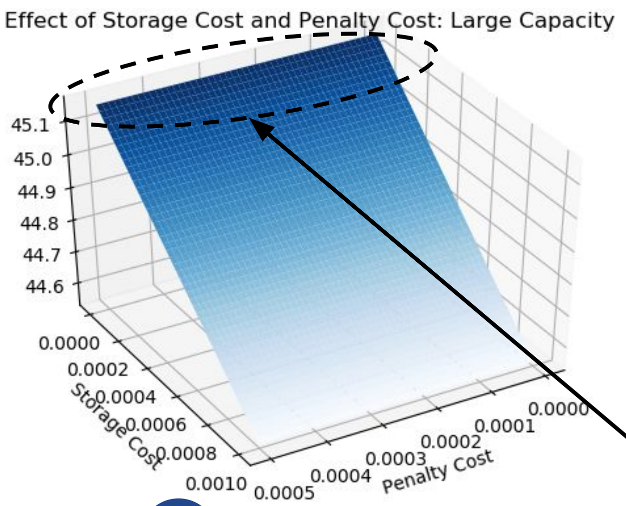
4 Low **Max Volume** offsets the positive effect of **Injection Rate**

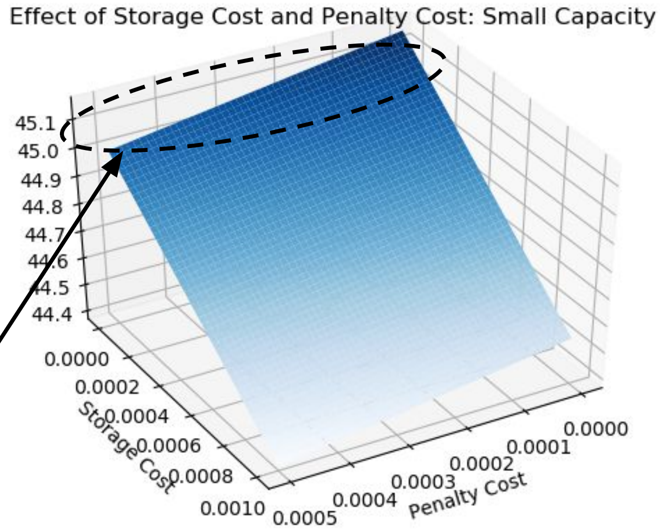


Effects of Penalty Cost and Storage Cost

5

Increasing Storage Cost always has a negative effect on **Value**





6

For large **Storage Capacity Penalty Cost** has no effect on **Value**. For small **Storage Capacities,** it has a negative effect

Conclusion: Effect of Parameters

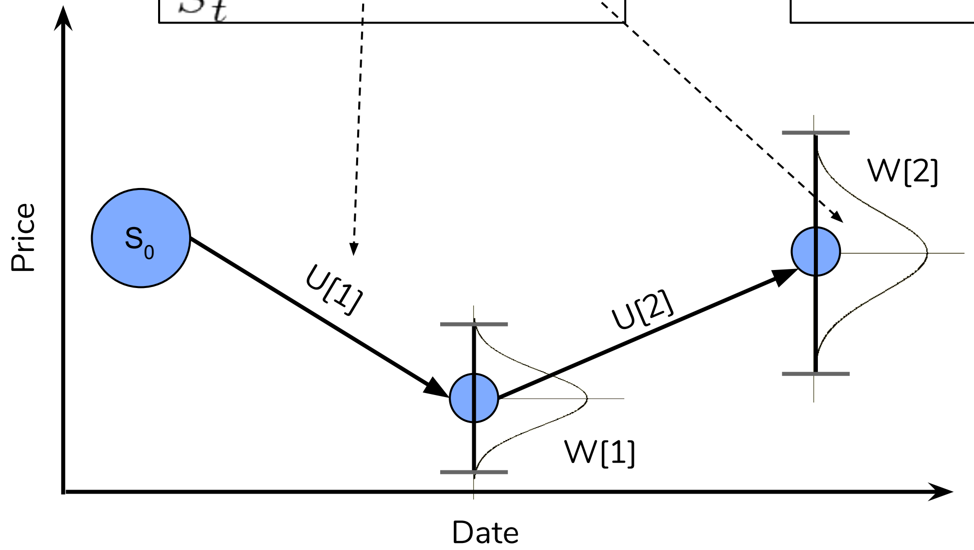
Parameters	Analyzed Effect on Value
Storage Cost	Negative effect
Injection/Withdraw Rate	Positive effect till a threshold (determined by **Max Volume**), Reduced Positive effect after that, Competing effect with **Storage Cost**
Max Volume	No effect if it is large compared to **Injection Rate.** Reduces **Value** on reduction when comparable to **Injection Rate**
Excess Penalty Cost	No effect for large **Max Volume**/ small **Injection rates**, Negative effect for small **Max Volume**/ large **Injection Rates**

Question 5

1.1. Stochastic Evolution
1.2. Monte Carlo Solution
1.3. Naive Optimization
1.4. Improved Optimization

Stochastic Price Evolution

$$\frac{dS_t}{S_t} = \mu_t dt + \sigma dW \longrightarrow S_T = S_0 * \exp\left(\left(\left(\int_0^T \mu_t dt\right) - \frac{\sigma^2 T}{2} + \sigma W_T\right)\right)$$



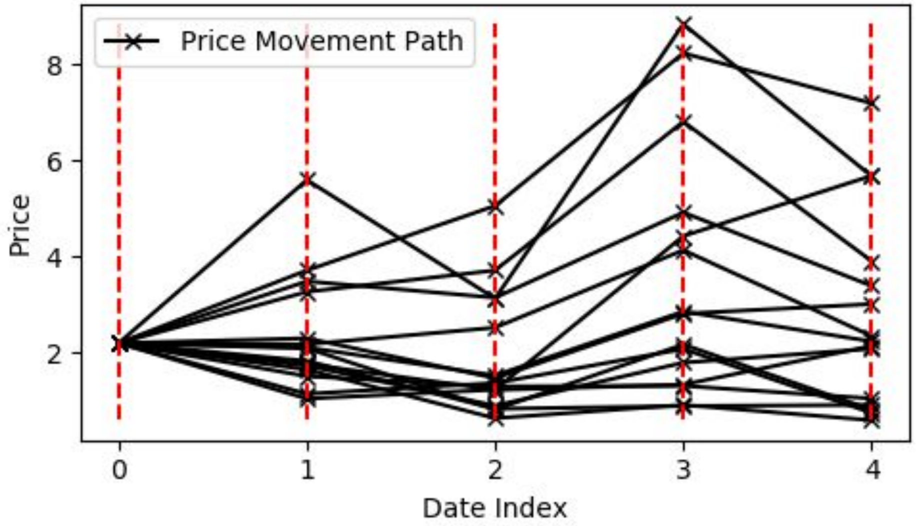
Drift: Deterministic Movement
Variation: Random Movement

- Uncertainty increases with time
- Drift piecewise constant

Monte Carlo Solution

Simulate multiple possible paths

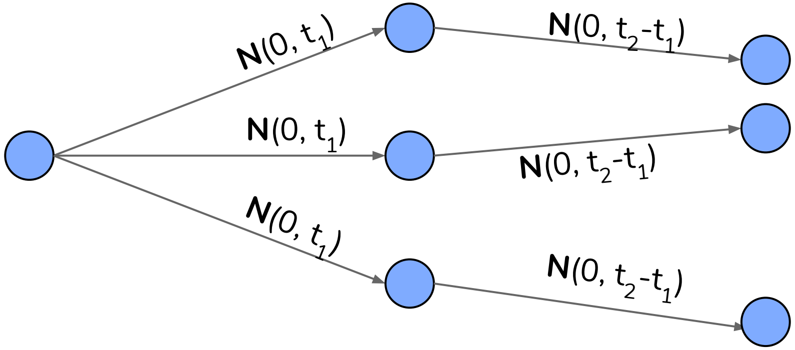
- Paths clustered around more likely price values
- Paths spreads out with time (more uncertainty)
- Use **sample mean** to get **best estimate** of price on any day



Monte Carlo Solution

Simulating a path

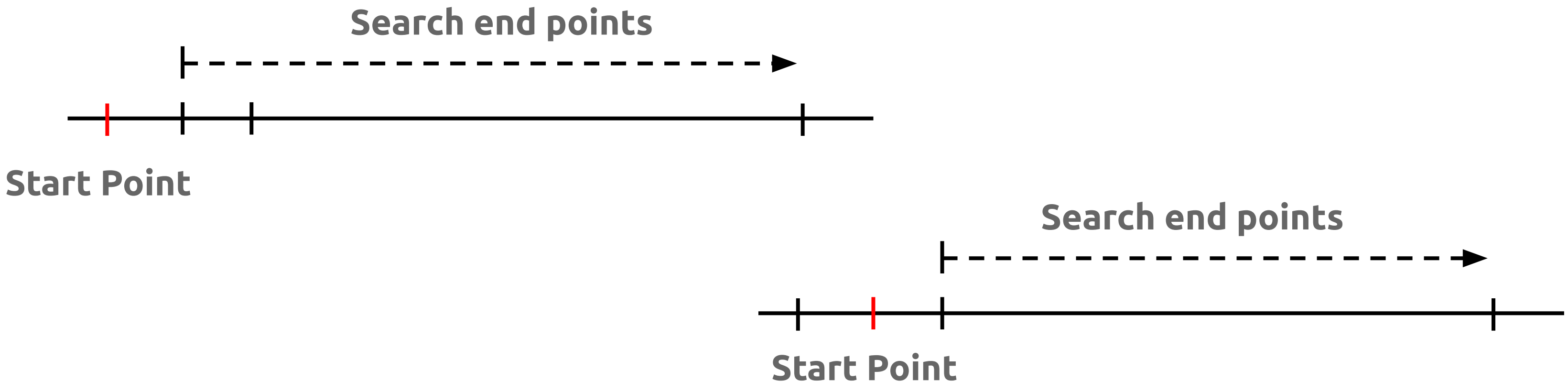
- Gaussian random variables → Characterised by mean and variance only
- $X_1 \sim N(0, \boldsymbol{\sigma_1}^2)$ and $X_2 \sim N(0, \boldsymbol{\sigma_2}^2) \Rightarrow X_1 + X_2 \sim N(0, \boldsymbol{\sigma_1}^2 + \boldsymbol{\sigma_2}^2)$ (if independent)



Expected Price Movement for Test Case: 2.8208, 2.3958, 3.4331, 2.9091

Naive Optimization

Search all combinations of buy and sell dates
Time Complexity: $O(n^2)$

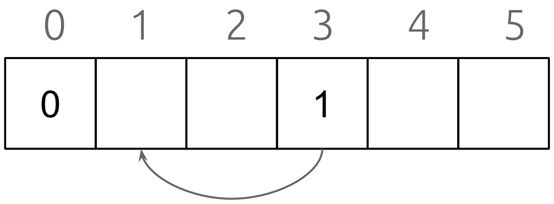


Improved Algorithm

Insights

- Storage cost ramps up linearly with time
- Use dynamic programming with tabulation

DP[i] holds the best point in the past to buy if we need to sell on day i



If we need to sell at day 3, buying on day 1 gives maximum value

The best value for the problem is the
max[opt[sell at 0], opt[sell at 1], ... opt[sell at N]]

Improved Algorithm

- Storage cost of all points < i-1 increase by the **same amount** when sold at i instead of i-1
- Therefore best buy point is either the best buy point of i-1 or i-1 itself

dp[i] = max{val[sell at i-1], val[sell at dp[i-1]]}

0	1	2	3	4	5
0	x	y	z	?	

The best buy-point for 4 (?) can either be 3 or the best buy-point of 3 (z)

Time Complexity = **O(n)**

Value of test case = **33.03175**

Thank You!

Any Questions?