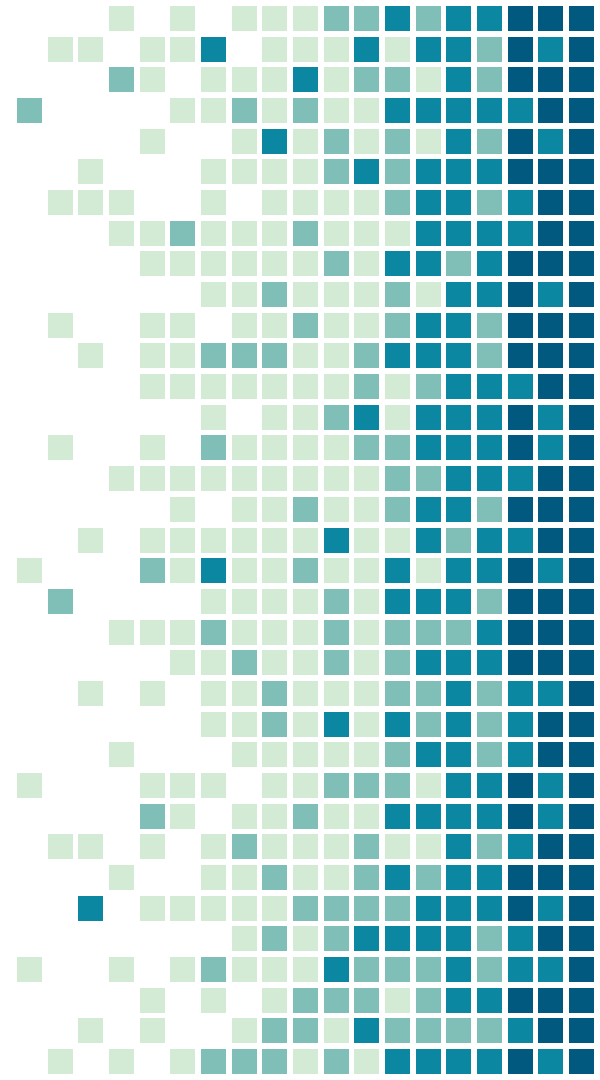




# GPU-accelerated

Fractional Differencing for Time Series Stationarity

Ritchie Ng, Jie Fu, Tat-Seng Chua





# Links

# Links

Github Repository:

[https://github.com/ritchieng/fractional\\_differencing\\_gpu](https://github.com/ritchieng/fractional_differencing_gpu)

Presentation:

[https://www.researchgate.net/publication/335159299\\_GFD\\_GPU\\_Fractional\\_Differencing\\_for\\_Rapid\\_Large-scale\\_Stationarizing\\_of\\_Time\\_Series\\_Data\\_while\\_Minimizing\\_Memory\\_Loss](https://www.researchgate.net/publication/335159299_GFD_GPU_Fractional_Differencing_for_Rapid_Large-scale_Stationarizing_of_Time_Series_Data_while_Minimizing_Memory_Loss)

# INTRO

Author, Collaborators and Supporters

# Main Author

## Ritchie Ng

- Chief AI Officer and Portfolio Manager, ensemblecap.ai
- Deep Learning Research Scholar in NExT++, NUS School of Computing
- NVIDIA Deep Learning Institute Instructor



**nVIDIA®**



**ENSEMBLE CAPITAL**



**nVIDIA®**



**ENSEMBLE CAPITAL**

# Co-author

**Jie Fu**, Postdoc Quebec Artificial Intelligence Institute (Mila)

**Tat-Seng Chua**, KITHCT Chair Professor at the School of Computing, NUS



# Teaching Assistants

**Timothy**, Quantitative Engineer Goldman Sachs

**Si An**, AI Pod, Temasek Holdings



**TEMASEK  
HOLDINGS**

# STATIONARITY

Common Approaches and Pitfalls

## Achieving Stationarity

# Common Approaches

### Why

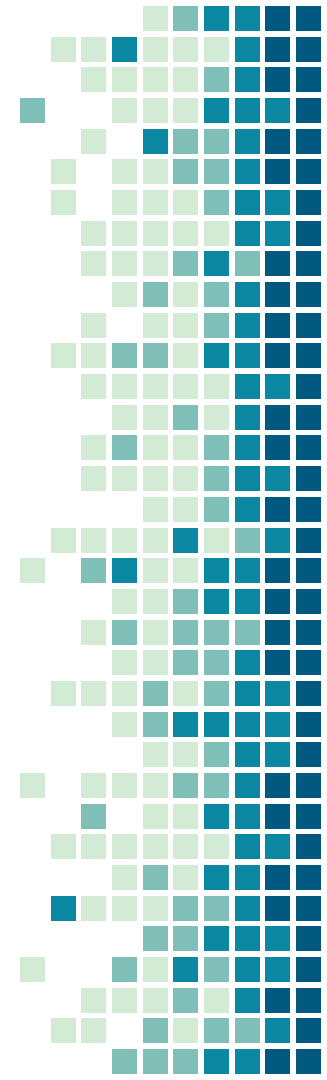
Typically we attempt to achieve some form of stationarity via a transformation on our time series.

### How

Common methods include integer differencing. For example to attempt to make S&P 500 time series stationary, we may take the one day difference yielding daily returns.

### Problem

However, integer differencing often removes too much memory in the time series. Often, we can achieve stationarity without losing too much memory via fractional differencing.

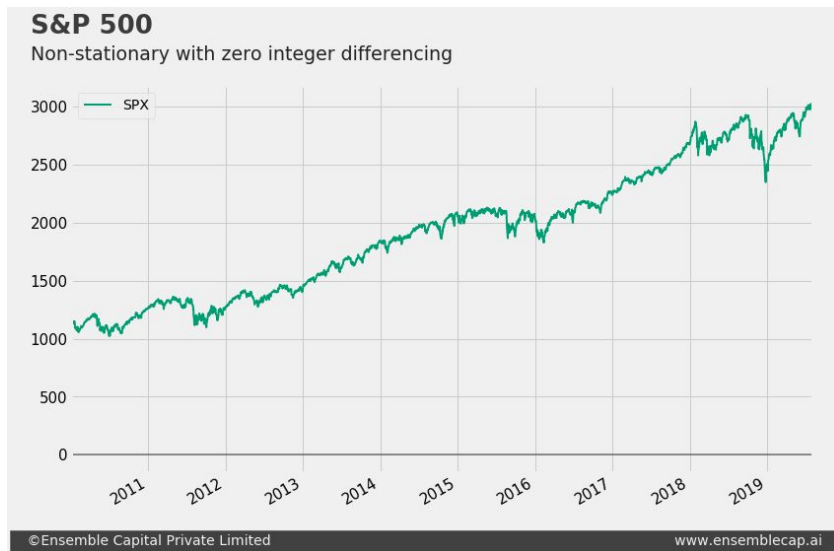




Achieving Stationarity

# Common Approaches

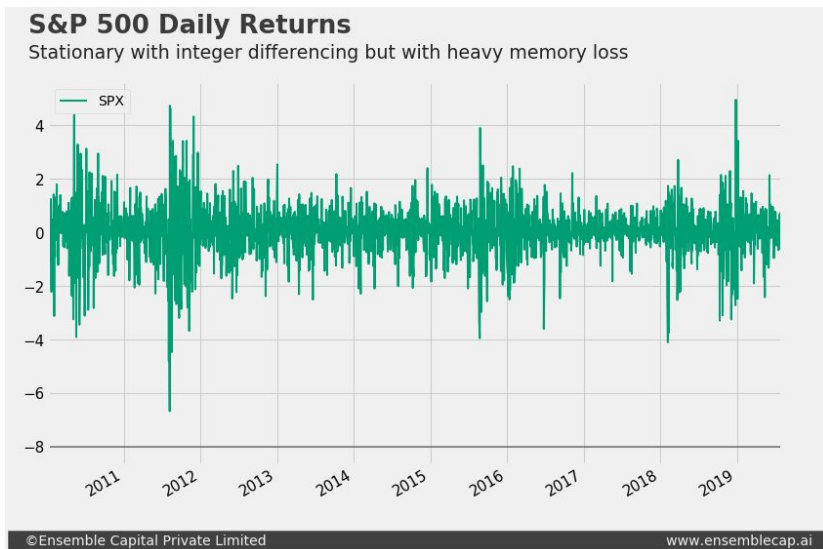
## S&P 500 Absolute Levels (Zero Differencing)



Achieving Stationarity

# Common Approaches

**S&P 500 Daily Returns (Integer Differencing,  $d=1$ )**



“Integer differencing unnecessarily removes too much memory while trying to make a time-series stationary. An alternative would be fractional differencing.

- Ritchie Ng

# STATIONARITY

Fractional Differencing to Achieve  
Maximum Memory with Stationarity

Achieving Stationarity

# Fractional Differencing

## Why

Fractional differencing allows us to achieve stationarity while maintaining the maximum amount of memory compared to integer differencing.

## Where

This was originally introduced in 1981 in his paper “Fractional Differencing” by J. R. M. Hosking<sup>1</sup> and subsequent work by others concentrated on fast and efficient implementations for fractional differentiation for continuous stochastic processes.

Recently, fractional differencing was introduced for financial time series through the fixed window fractional differencing instead of the expanding window method by Marcos Lopez de Prado<sup>2</sup>.



Expanding Window

# Fractional Differencing

**How?**

## Step 1: Calculating Weights Array

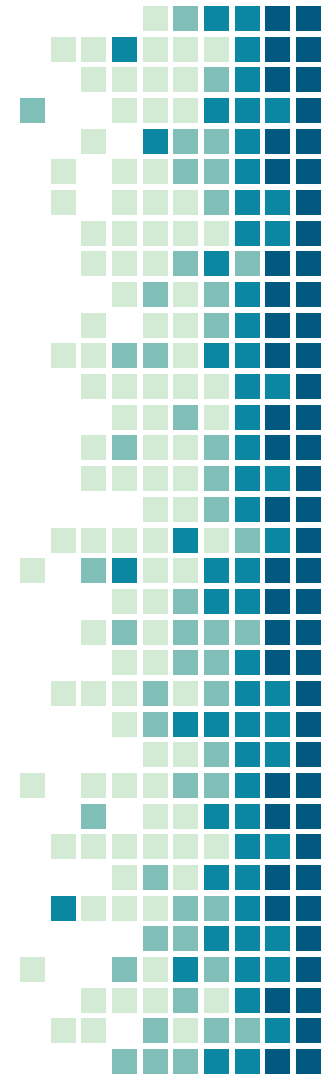
Essentially, independent of any time series, we can calculate the weights array via this iterative equation.

$$w_k = -w_{k-1} \frac{d - k + 1}{k}$$

w: weight at lag k

k: lag

d: fractional differencing value where 0 implies no differencing and above 1 implies integer differencing



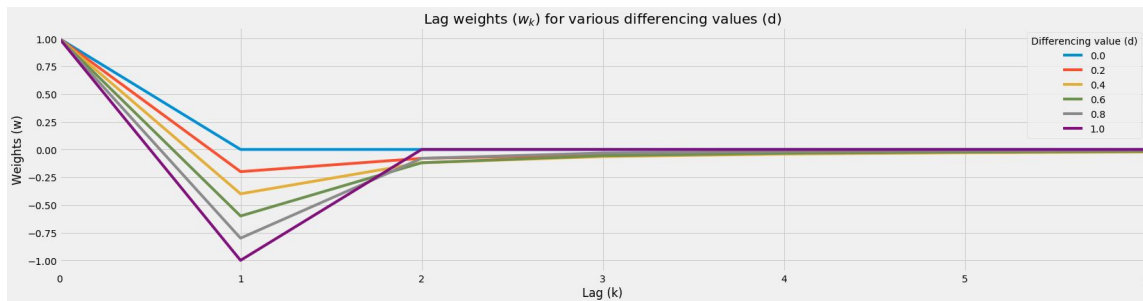
Expanding Window

# Fractional Differencing

How?

## Step 1: Calculating Weights Array

Essentially, independent of any time series, we can calculate the weights array via this iterative equation.



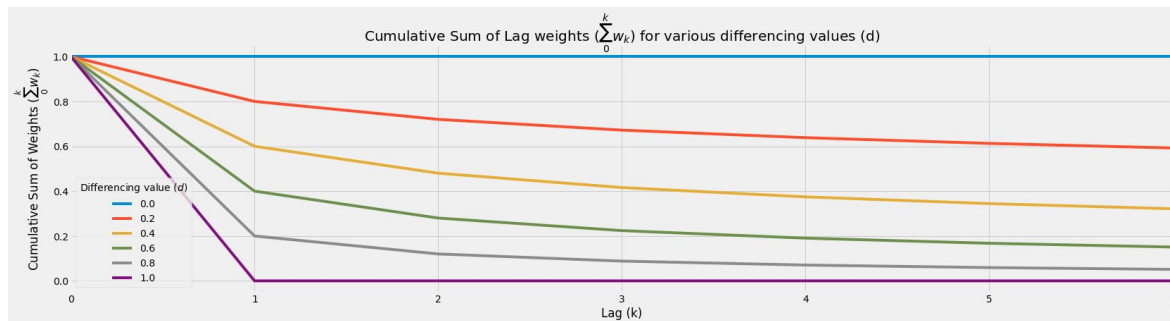
Expanding Window

# Fractional Differencing

How?

## Step 1: Calculating Weights Array

Essentially, independent of any time series, we can calculate the weights array via this iterative equation.





Expanding Window

# Fractional Differencing

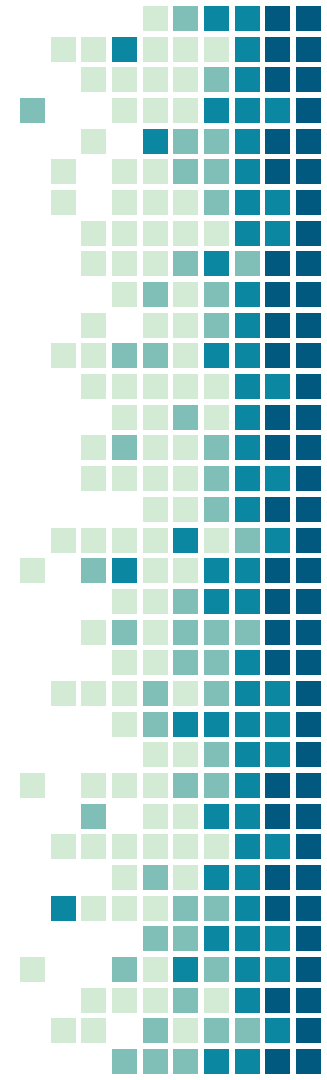
## How?

### Step 2: Rolling Dot Product of Weights Array and Time Series Array

When we take the dot product of the weights array and the time series array, we get a single value at lag  $k = 0$ . We do this for all lags  $k > 0$ , until we reach the beginning of the time series.

## Problem?

Notice how this is very computationally expensive as we even take parts of the weights array for our dot product where the values are extremely small? And this window keeps expanding as we move further down the time series timeline. The alternative to this is the fixed-window fractional differencing method.



Fixed Window

# Fractional Differencing

How?

## Step 1: Calculating Weights Array with Threshold

Essentially, independent of any time series, we can calculate the weights array via this iterative equation and put a floor to stop calculating when the weights are too small.

$$w_k = -w_{k-1} \frac{d - k + 1}{k}$$

$$w_k > \tau$$

w: weight at lag k

k: lag

d: fractional differencing value where 0 implies no differencing and above 1 implies integer differencing

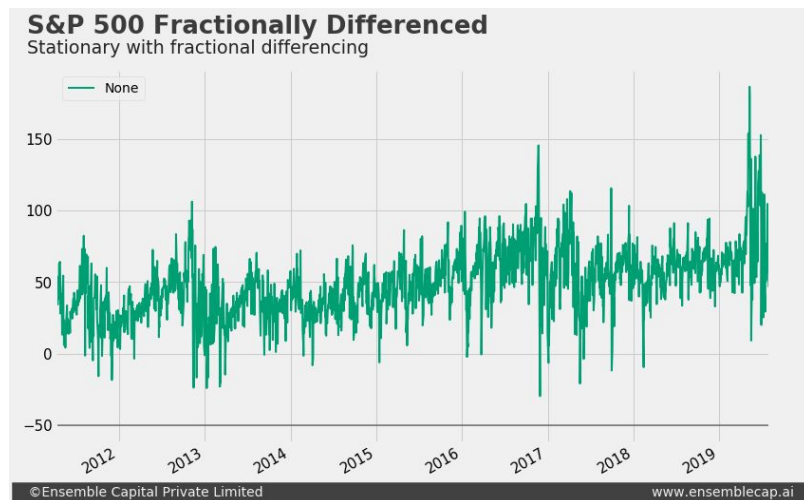
$\tau$ : the threshold to stop calculating

Fixed Window

# Fractional Differencing

## Example

Applying a fixed window fractional differencing on S&P 500, we get the following.



## Fixed Window

# Fractional Differencing

### ADF Tests: S&P 500 (2012-2019)

Comparing the three ADF test with constant order only included in the regression :  
no differencing, integer differencing ( $d=1$ ) and fractional differencing ( $d=0.5$ ,  $\tau = 5e-5$ ).

	No Differencing	Integer Differencing	Fractional Differencing
t Statistic	-0.11	-11.12	-4.38
Critical Values	1%: -3.43 5%: -2.86 10%: -2.57		

**Important:** there are other ways to check for stationarity like Kwiatkowski–Phillips–Schmidt–Shin (KPSS) tests for trend stationarity, Phillips–Perron test for higher order autocorrelation and Augmented Dickey–Fuller test (ADF) with linear/quadratic trend order to include in the regression. But they are not covered as it is not the main point. The point is to show how we can minimize memory loss while reaching stationarity with fractional differencing.

Achieving Stationarity

# Fractional Differencing

## Derivation of Fractional Differencing Weights Formula

(refer to Hosking<sup>1</sup> paper)



“ Existing CPU-based implementations are inefficient for running fractional differencing on many large-scale time-series. GPU-based implementations provide an avenue to adapt to this century’s big data requirements.

- Ritchie Ng

# PERFORMANCE

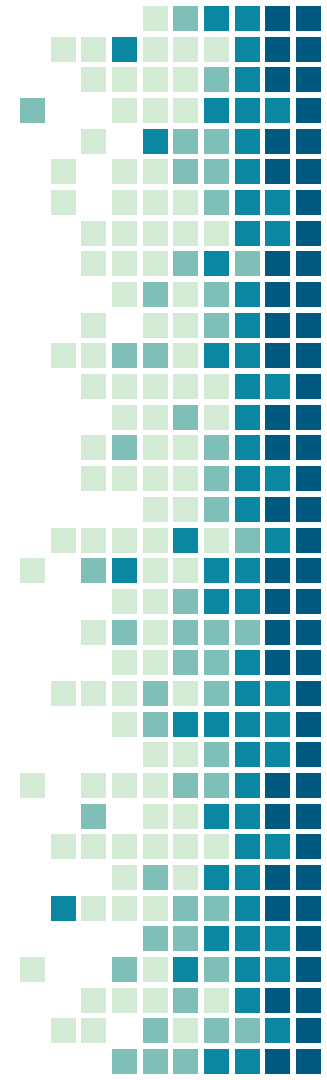
GPU vs CPU Implementation

Fixed Window Fractional Differencing

# Improvements

## Improvement Indicators

- 6x-7x speed-up on a dataset of S&P 500 (2000-2019)





Fixed Window Fractional Differencing

# Improvements

	Mean (seconds)	Standard Deviation (seconds)
Google Colab: 2x vCPUs	2.9105	0.0265
Google Colab: 1x T4 GPU	0.3704	0.0448
GCP 4x vCPUs, 15 GB RAM	2.0105	0.0551
GCP 8x vCPUs, 15 GB RAM	2.0774	0.0419
GCP 1x Tesla V100 GPU	0.3111	0.0080

Fixed Window Fractional Differencing

# Improvements

	Speed-up
Speed-up Colab 1x T4 vs Colab 2vCPUs	7.8581x
Speed-up GCP 1x V100 vs GCP 4vCPUs	6.4636x
Speed-up GCP 1x V100 vs GCP 8vCPUs	6.6786x

# References

# References

Hosking, J. R. M. Fractional Differencing. Biometrika 68, no. 1 (1981): 165–76.

Marcos Lopez de Prado. 2018. Advances in Financial Machine Learning (1st ed.). Wiley Publishing.



*Corporate: [ritchie@ensemblecap.ai](mailto:ritchie@ensemblecap.ai)*

*Academic: [ritchieng@u.nus.edu](mailto:ritchieng@u.nus.edu)*

# CONTACT

