Time Series Analysis

Forecasting and Autoregressive Models

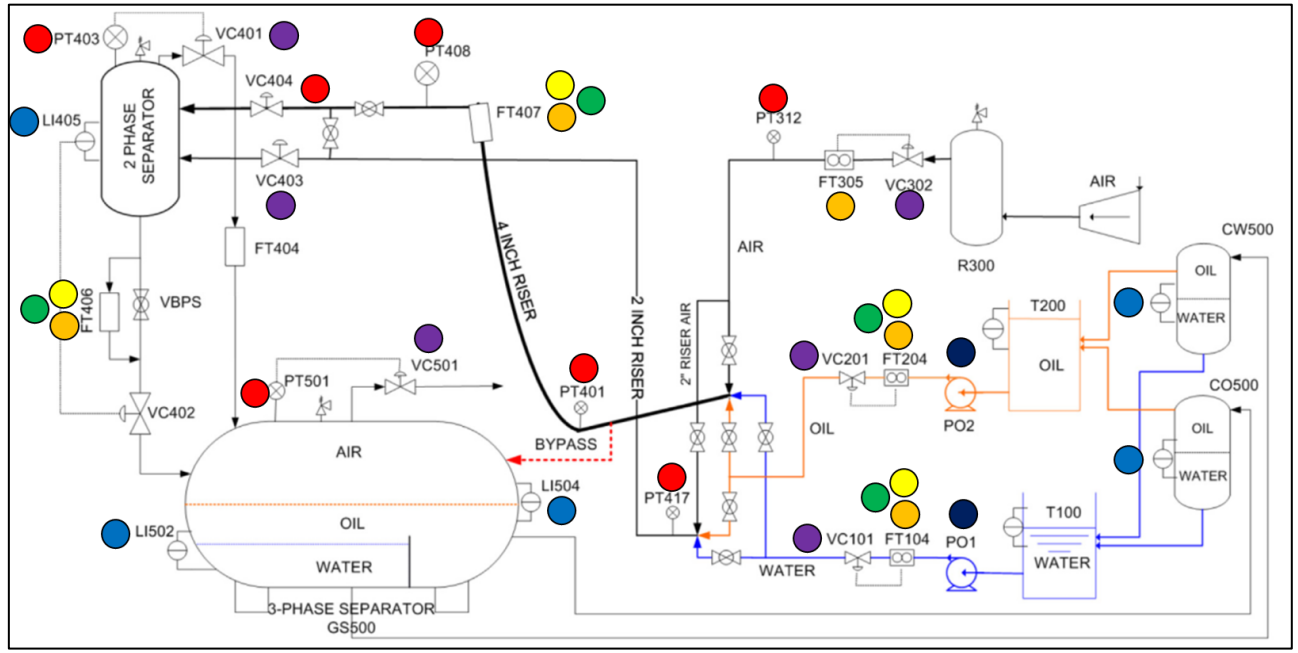
Assoc. Prof. Karl Ezra Pilario, Ph.D.
Process Systems Engineering Laboratory
Department of Chemical Engineering
University of the Philippines Diliman

Outline

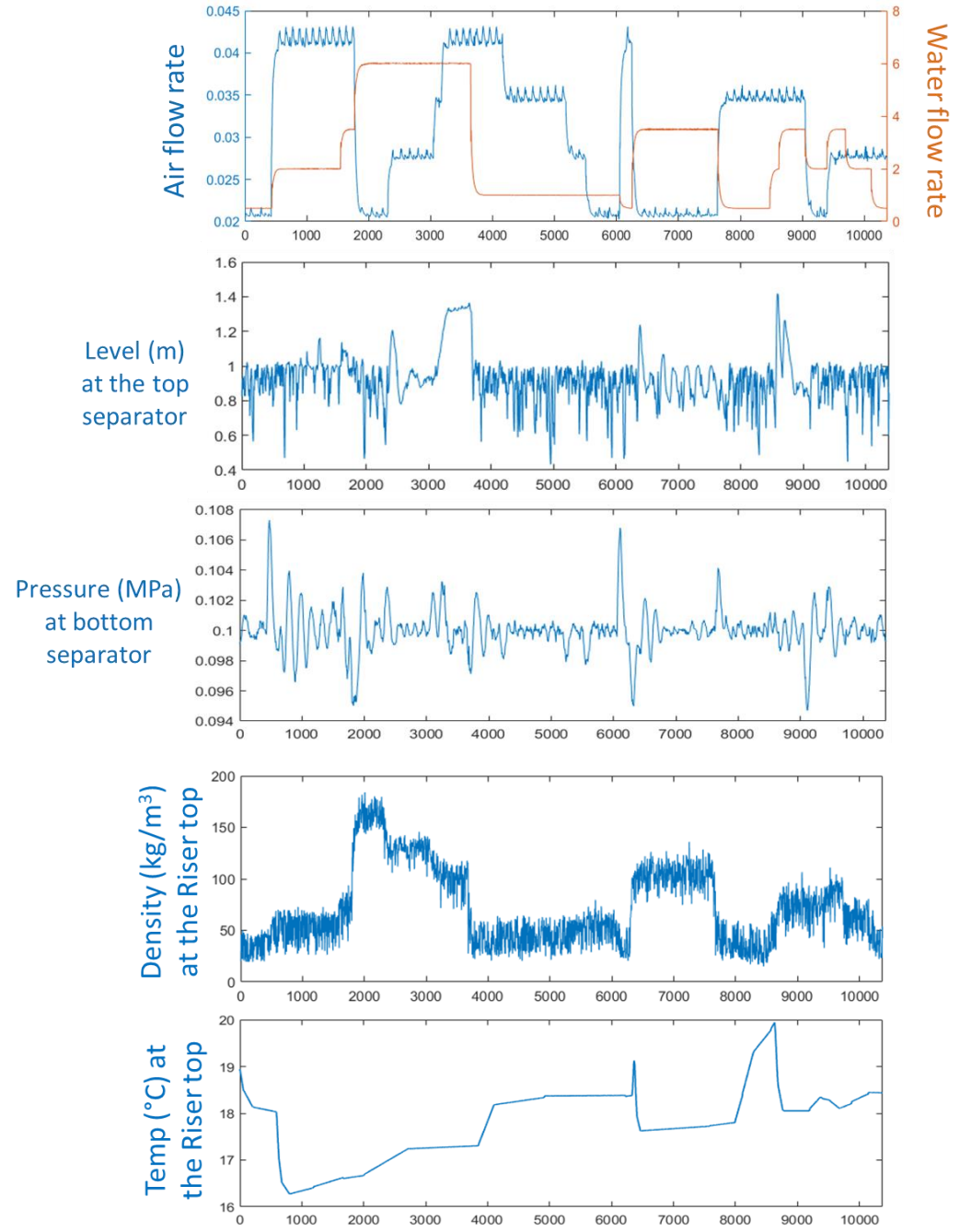
- Time Series
  - Univariate Time Series Models
    - Linear Regression (AR models)
    - Machine Learning
  - Multivariate Time Series Models
    - Vector AR models
    - State-space models

Time Series Data in ChemE

Plant data sets are multivariate time series data.



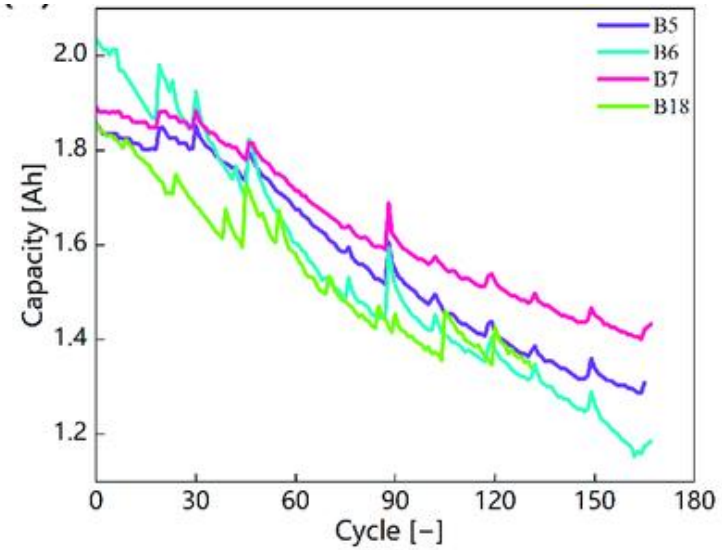
- Pressure
- Level
- Flow Rate
- Density
- Temperature
- Valve Position
- Pump current



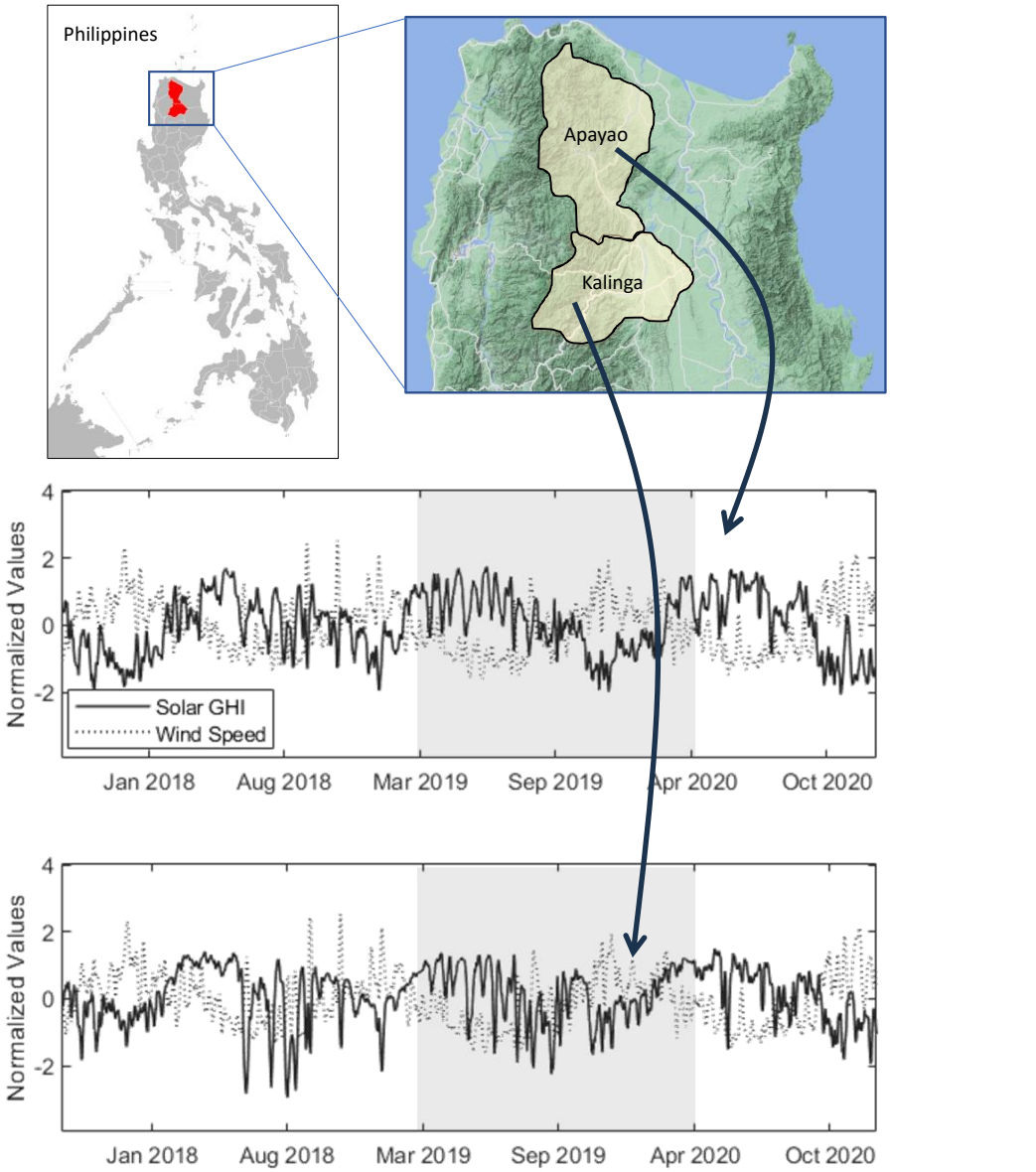
Time Series Data in ChemE

Time series data are also common in energy and environmental engineering.

Battery Degradation data sets



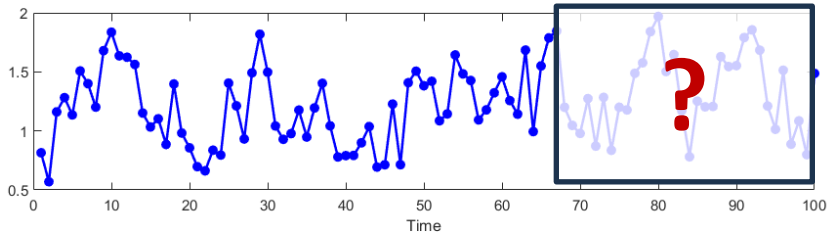
Weather variables: Solar irradiance / wind speeds



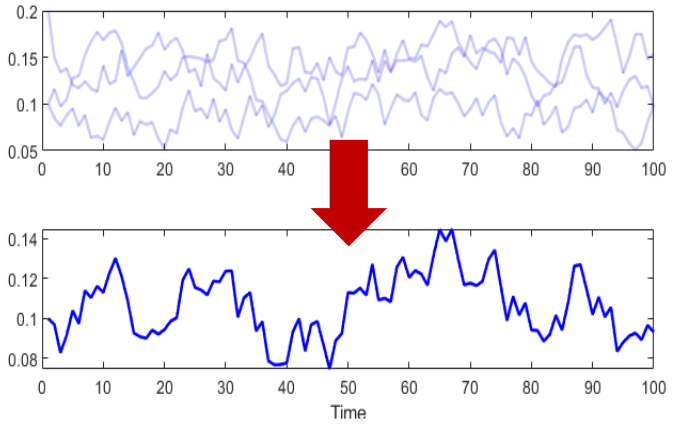
Time Series Data in ChemE

Machine Learning tasks can also apply for time series data.

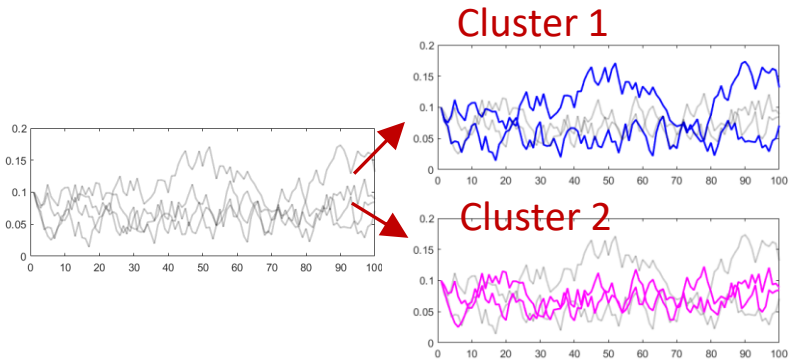
Regression (Forecasting)



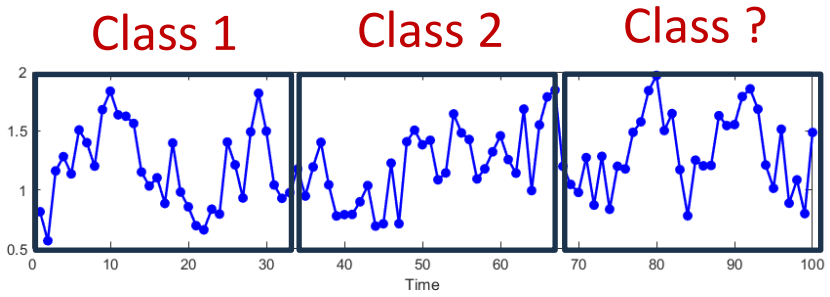
Dimensionality Reduction



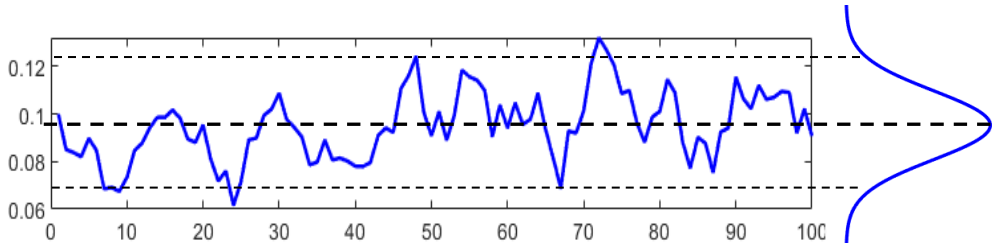
Clustering



Classification



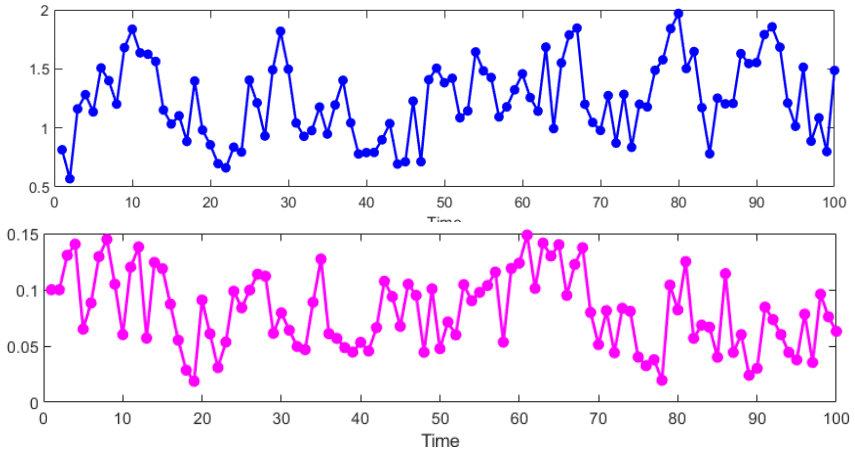
Density Estimation and Anomaly Detection



Time Series Data

- An ordered sequence of observations in time.

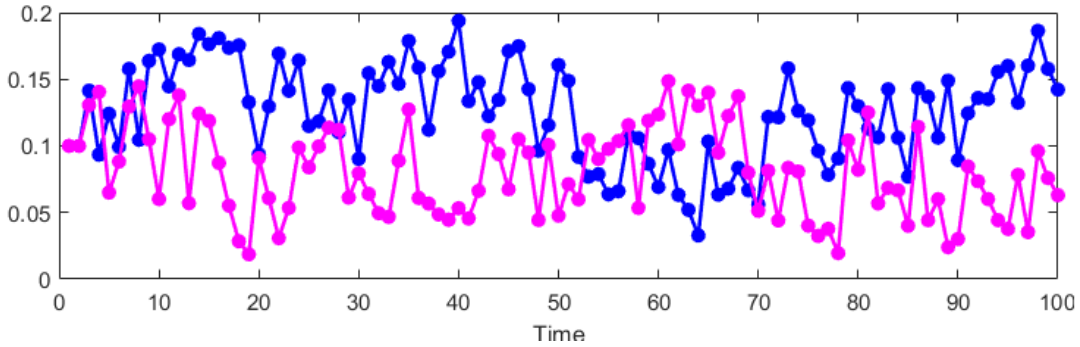
Univariate data



- The future value of a variable depends on its past values alone.
- Two variables can still be analyzed separately as univariate time series:

$$y_1(k) = f(y_1(k-1), y_1(k-2), \dots)$$
$$y_2(k) = f(y_2(k-1), y_2(k-2), \dots)$$

Multivariate data

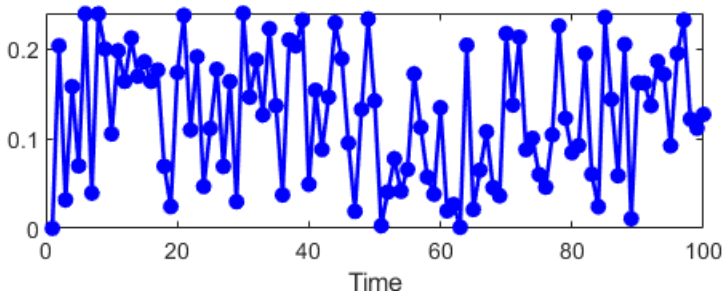


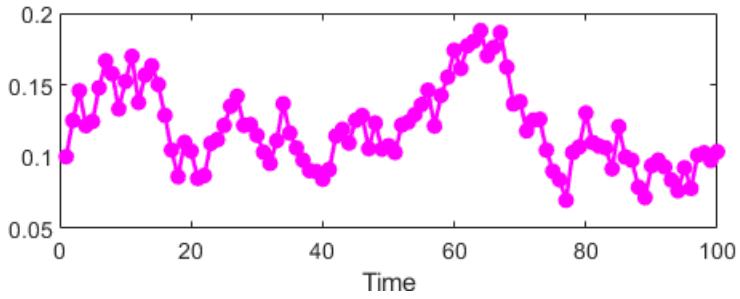
- The future value of a variable depends on its past values *as well as those of other variables*.
- Two variables analyzed as multivariate time series:

$$y_1(k) = f(y_1(k-1), y_2(k-1), \dots)$$
$$y_2(k) = f(y_1(k-1), y_2(k-1), \dots)$$

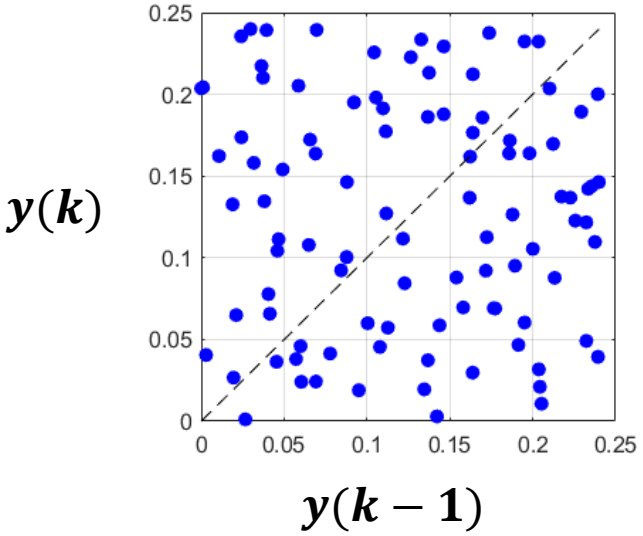
How do we know that future values depend on past values in a univariate time series?

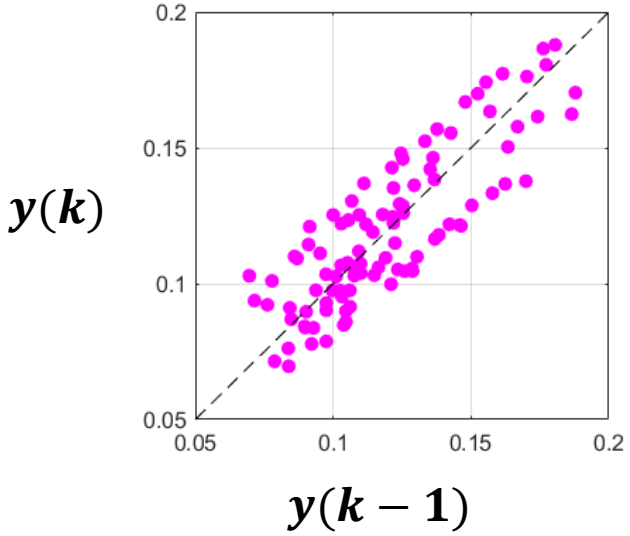
Here are two time-series data, one of them is a purely random process, the other has dependence.





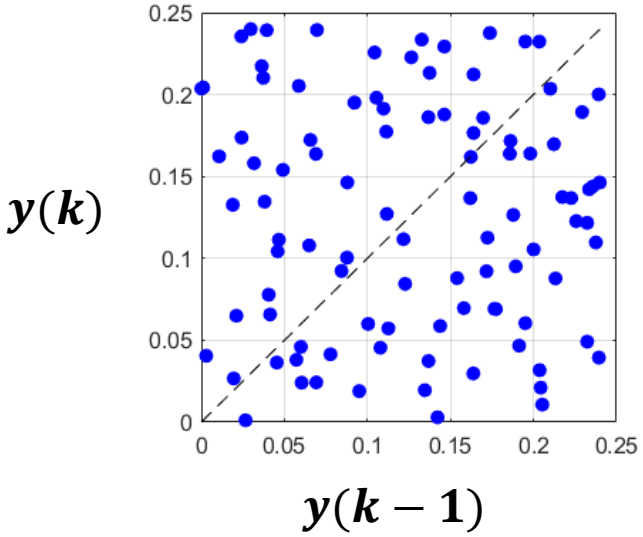
Plots of $y(k)$ versus $y(k-1)$ for all k:





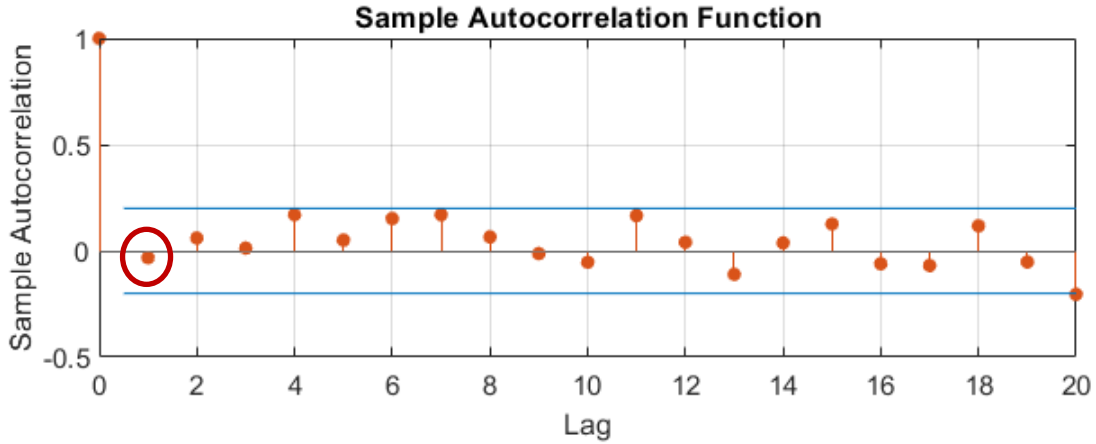
Here, there is **autocorrelation**.

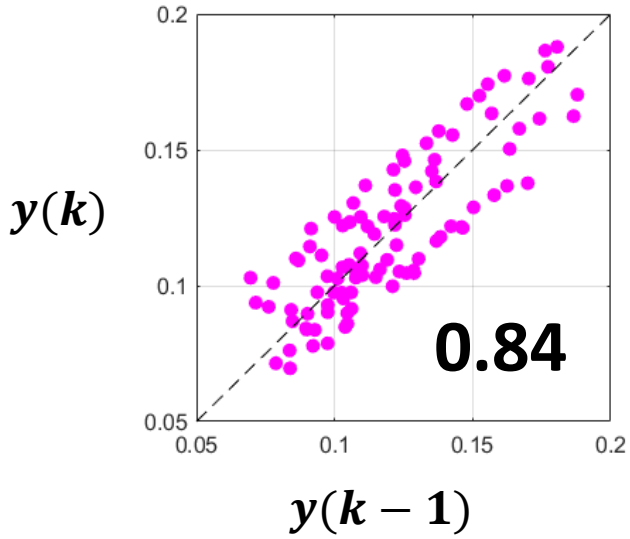
Autocorrelation Function (ACF)

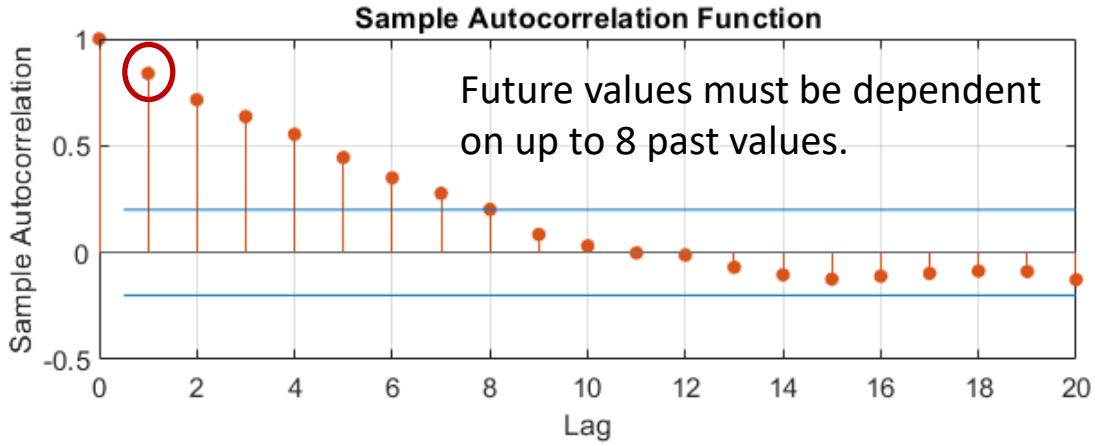


-0.03

We just analyzed the autocorrelation for 1 lag only.
How about other lags?



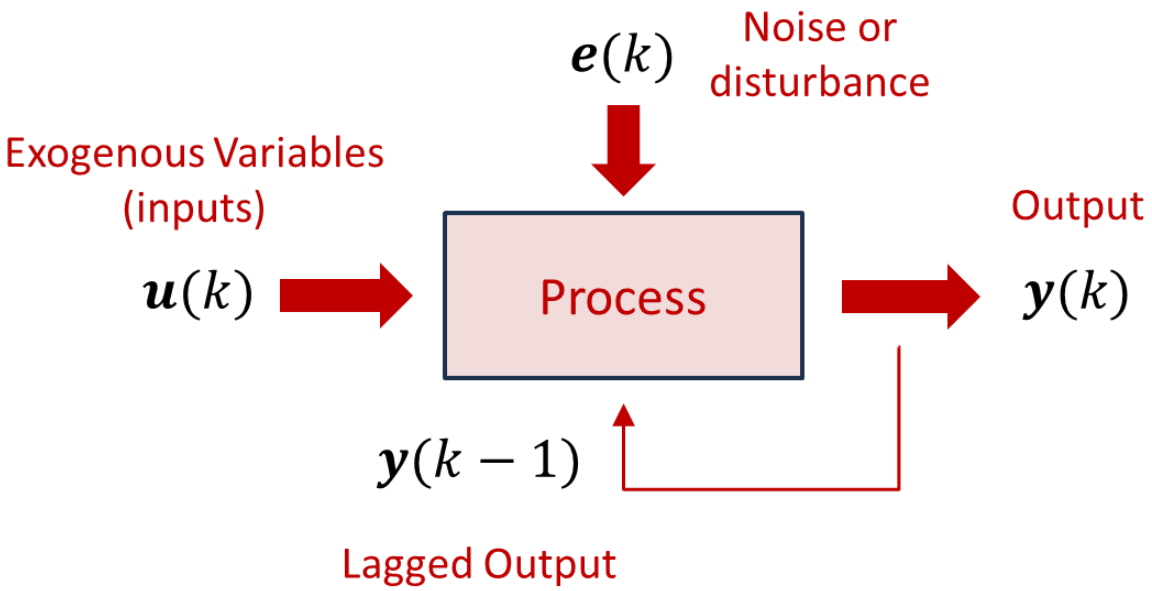




Autoregressive Models

In general, what are the factors that influence a future time series value?

$$\boldsymbol{y}(k) = \boldsymbol{f}\left(\begin{matrix} \boldsymbol{y}(k-1) \\ \boldsymbol{y}(k-2) \\ \vdots \end{matrix}, \begin{matrix} \boldsymbol{u}(k) \\ \boldsymbol{u}(k-1) \\ \boldsymbol{u}(k-2) \\ \vdots \end{matrix}, \begin{matrix} \boldsymbol{e}(k) \\ \boldsymbol{e}(k-1) \\ \boldsymbol{e}(k-2) \\ \vdots \end{matrix} \right)$$



AR (Autoregressive Model)

$AR(p)$

$$y(k) = a_0 + \sum_{i=1}^{p} a_i y(k-i) + e(k)$$

ARX (Autoregressive with Exogenous Input)

$ARX(p, q)$

$$y(k) = a_0 + \sum_{i=1}^{p} a_i y(k-i) + \sum_{j=1}^{q} b_j u(k-j) + e(k)$$

ARMA (Autoregressive, Moving Average)

$ARMA(p, q)$

$$y(k) = a_0 + \sum_{i=1}^{p} a_i y(k-i) + \sum_{j=1}^{q} c_j e(k-j) + e(k)$$

ARMAX (ARMA with Exogeneous Input) $ARMAX(p, q, r)$

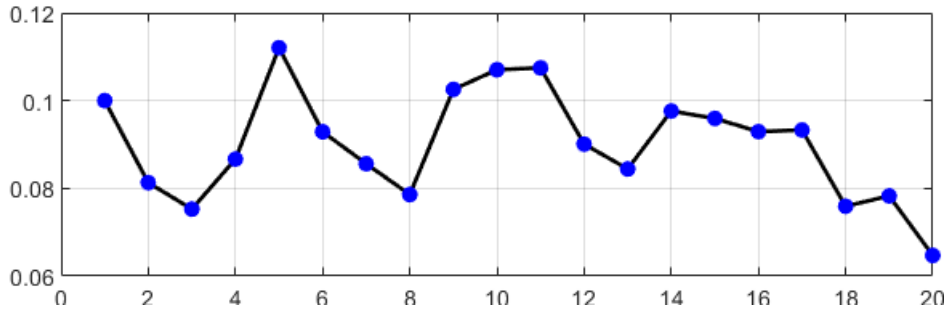
$$y(k) = a_0 + \sum_{i=1}^{p} a_i y(k-i) + \sum_{j=1}^{q} b_j u(k-j) + \sum_{l=1}^{r} c_l e(k-l) + e(k)$$

How to turn time series into tabular data?

Example:

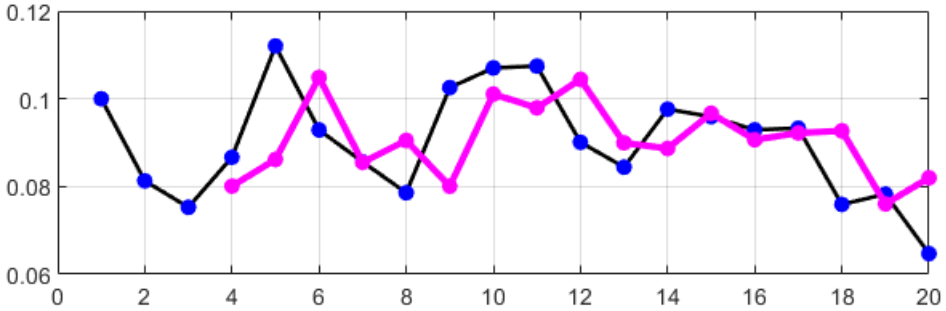
Train an AR(3) model on the following data.

$$y(k) = a_0 + a_1 y(k-1) + a_2 y(k-2) + a_3 y(k-3)$$



Result: Using linear regression,

a_0 = 0.2949
a_1 = -0.2139
a_2 = 0.9033



0.1000
0.0813
0.0753
0.0867
0.1120
0.0929
0.0856
0.0786
0.1025
0.1070
0.1075
0.0901
0.0844
0.0976
0.0959
0.0929
0.0933
0.0759
0.0783
0.0647

x			y
0.1000	0.0813	0.0753	0.0867
0.0813	0.0753	0.0867	**0.1120**
0.0753	0.0867	**0.1120**	0.0929
0.0867	**0.1120**	0.0929	0.0856
0.1120	0.0929	0.0856	0.0786
0.0929	0.0856	0.0786	0.1025
0.0856	0.0786	0.1025	0.1070
0.0786	0.1025	0.1070	0.1075
0.1025	0.1070	0.1075	0.0901
0.1070	0.1075	0.0901	0.0844
0.1075	0.0901	0.0844	0.0976
0.0901	0.0844	0.0976	0.0959
0.0844	0.0976	0.0959	0.0929
0.0976	0.0959	0.0929	0.0933
0.0959	0.0929	0.0933	0.0759
0.0929	0.0933	0.0759	0.0783
0.0933	0.0759	0.0783	0.0647

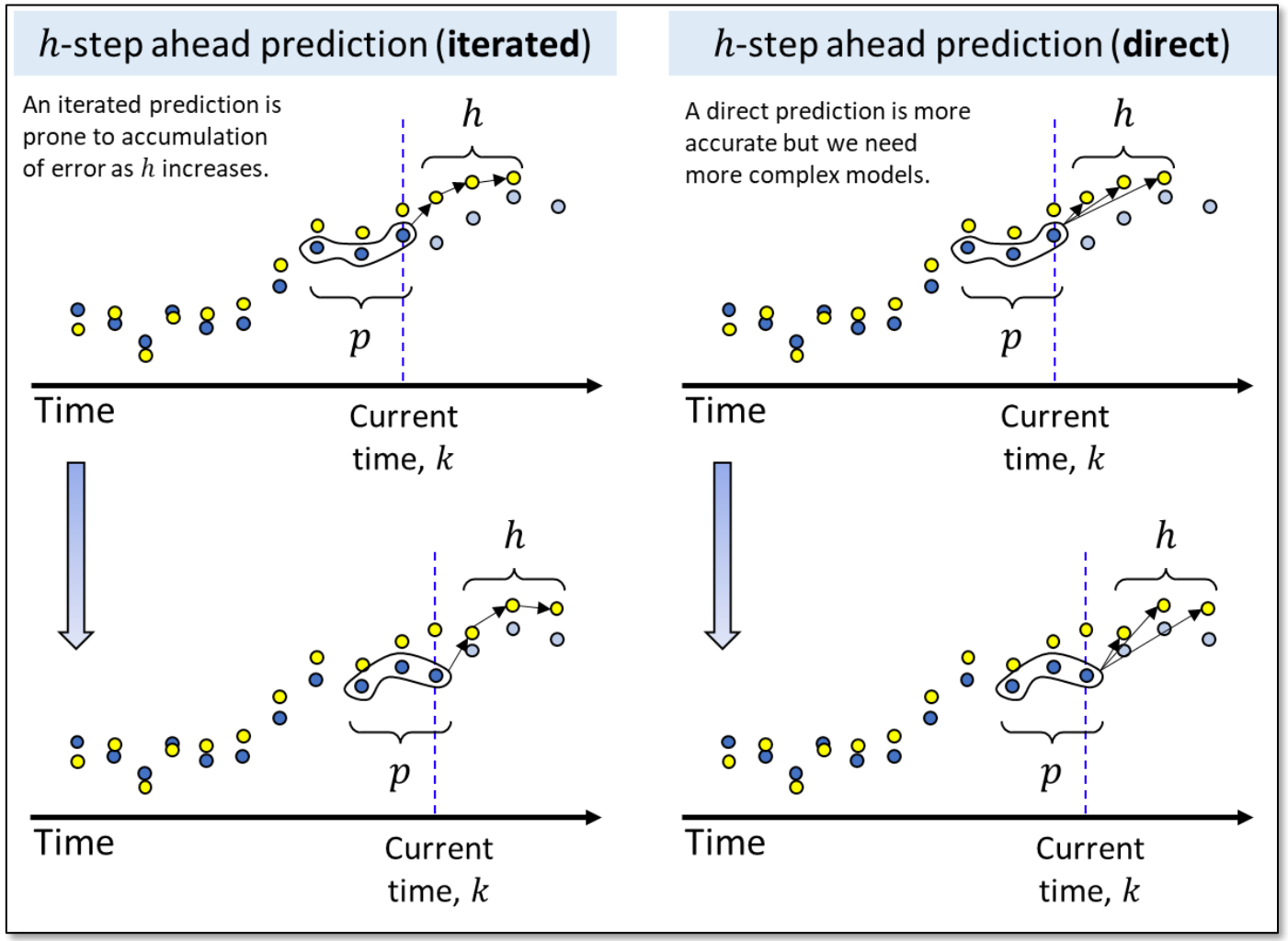
For an $AR(p)$ model, the data set transforms from N data points into $N - p$ data points.

H-step ahead prediction

x			$y(k)$	$y(k+1)$
0.1000	0.0813	0.0753	0.0867	?????
0.0813	0.0753	0.0867	**0.1120**	?????
0.0753	0.0867	**0.1120**	0.0929	?????
0.0867	**0.1120**	0.0929	0.0856	?????
0.1120	0.0929	0.0856	0.0786	?????
0.0929	0.0856	0.0786	0.1025	?????
0.0856	0.0786	0.1025	0.1070	?????
0.0786	0.1025	0.1070	0.1075	?????
0.1025	0.1070	0.1075	0.0901	?????
0.1070	0.1075	0.0901	0.0844	?????
0.1075	0.0901	0.0844	0.0976	?????
0.0901	0.0844	0.0976	0.0959	?????
0.0844	0.0976	0.0959	0.0929	?????
0.0976	0.0959	0.0929	0.0933	?????
0.0959	0.0929	0.0933	0.0759	?????
0.0929	0.0933	0.0759	0.0783	?????
0.0933	0.0759	0.0783	0.0647	?????

One-step ahead

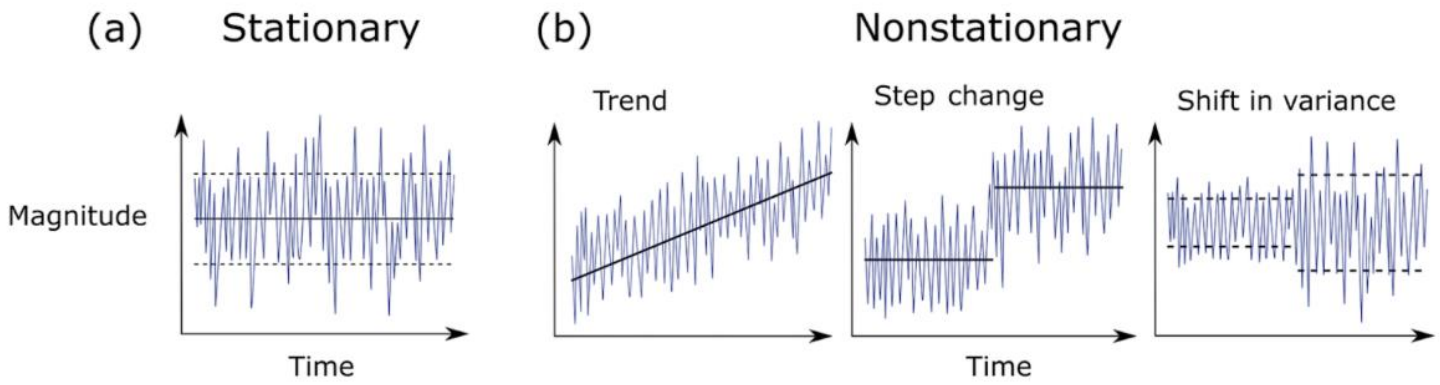
We have two ways to make an h-step ahead prediction:



ARIMA model

Stationary time series – statistical properties (e.g. distribution) of the time series are constant with time.
Non-stationary time series – statistical properties of the time series change with time.

* Stationarity can be checked using the Augmented Dickey-Fuller test.



ARIMA (Autoregressive Integrated Moving Average)

- The appropriate linear model for non-stationary time series.
- Perform *differencing* first, then train an ARMA:

$$y'(k) = y(k) - y(k-1)$$
$$y''(k) = y'(k) - y'(k-1)$$

ARMA $$y(k) = a_0 + \sum_{i=1}^{p} a_i y(k-i) + \sum_{j=1}^{q} c_j e(k-j) + e(k)$$

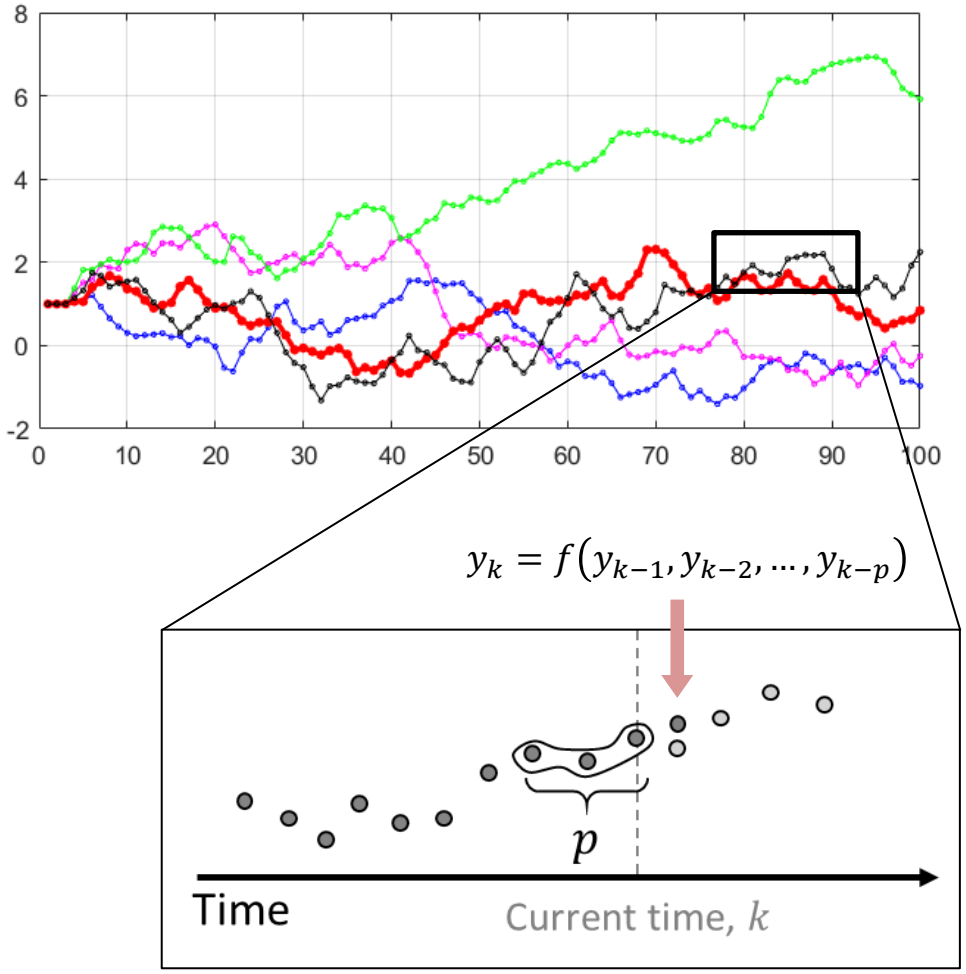
ARIMA $$y'(k) = a_0 + \sum_{i=1}^{p} a_i y'(k-i) + \sum_{j=1}^{q} c_j e(k-j) + e(k)$$

Outline

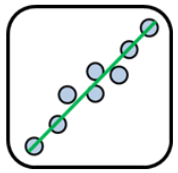
- Time Series
  - Univariate Time Series Models
    - Linear Regression (AR models)
    - **Machine Learning**
  - Multivariate Time Series Models
    - Vector AR models
    - State-space models

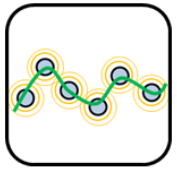
Machine Learning in Forecasting

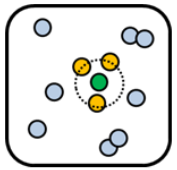
Instead of using linear models, we can use any ML model to train $y = f(x)$.
This will turn ML models into *autoregressive* ML models.

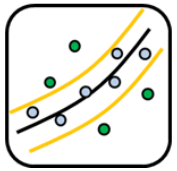


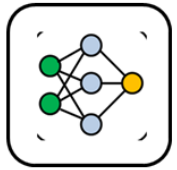
x			y
0.1000	0.0813	0.0753	0.0867
0.0813	0.0753	0.0867	**0.1120**
0.0753	0.0867	**0.1120**	0.0929
0.0867	**0.1120**	0.0929	0.0856
0.1120	0.0929	0.0856	0.0786
0.0929	0.0856	0.0786	0.1025
0.0856	0.0786	0.1025	0.1070
0.0786	0.1025	0.1070	0.1075
0.1025	0.1070	0.1075	0.0901
0.1070	0.1075	0.0901	0.0844
0.1075	0.0901	0.0844	0.0976
0.0901	0.0844	0.0976	0.0959
0.0844	0.0976	0.0959	0.0929
0.0976	0.0959	0.0929	0.0933
0.0959	0.0929	0.0933	0.0759
0.0929	0.0933	0.0759	0.0783
0.0933	0.0759	0.0783	0.0647

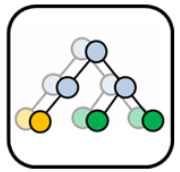

Linear Regression

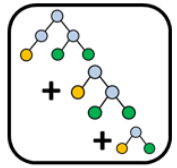

KRR
Kernel Ridge Regression


kNN
K-Nearest Neighbors


SVR
Support Vector Regression


MLP
Multi-layer Perceptron


RF
Random Forest


GBR
Gradient Boosting Regression

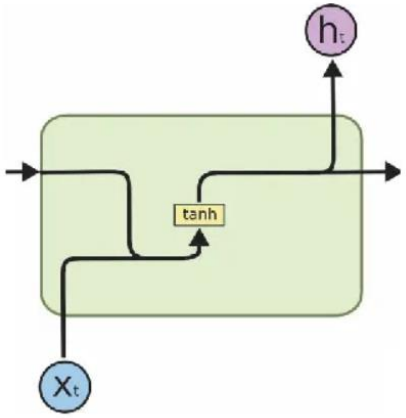

GPR
Gaussian Process Regression

Machine Learning: RNN vs GRU vs LSTM

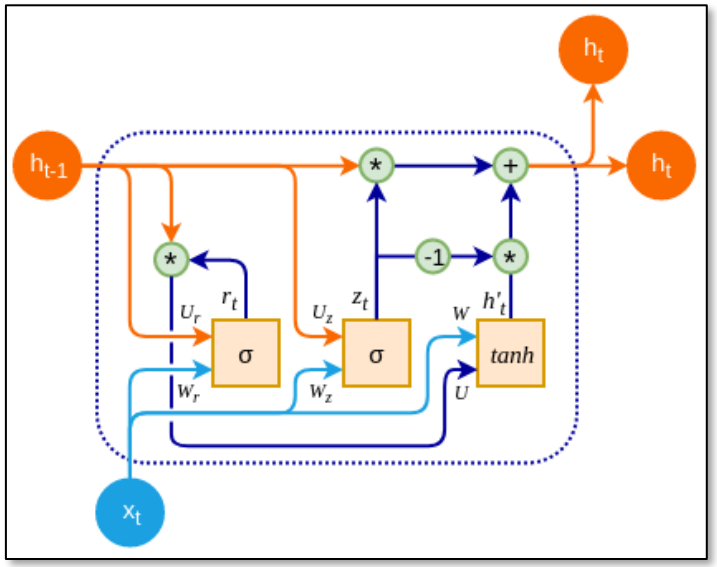
Deep learning models for time series:

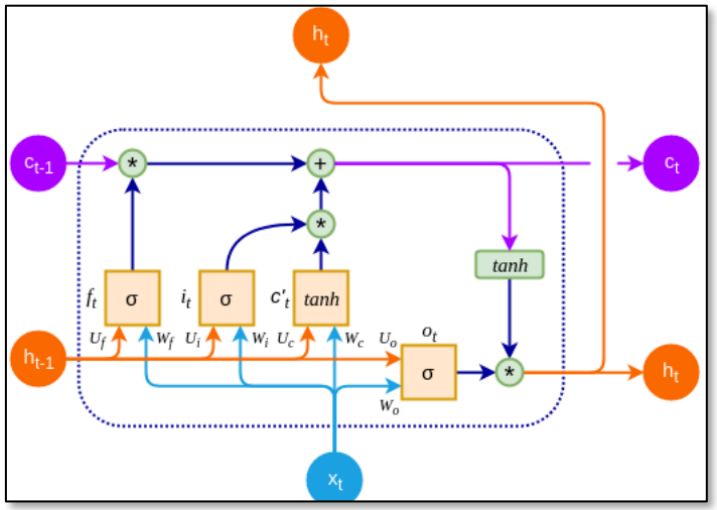
- Recurrent Neural Nets (RNNs)
- Gated Recurrent Units (GRUs)
- Long Short-term Memory (LSTMs)

Recurrent Neural Nets (RNNs)



- Contains recurrent units.
- Trained by Backpropagation Through Time (BPTT)
- Suffers from the *Vanishing Gradient* problem.
  - The partial derivative of the loss function approaches zero when there are many layers.





GRU

- Solves the *Vanishing Gradient* problem using gates, which are neural networks, σ, themselves.

- **Update Gate, z_t:** decides whether to update the state h_{t-1} with the new h'_t.
- **Reset Gate, r_t:** decides how much memory from the past h_{t-1} influences the present.

LSTM

- Similar to GRU but with added features.

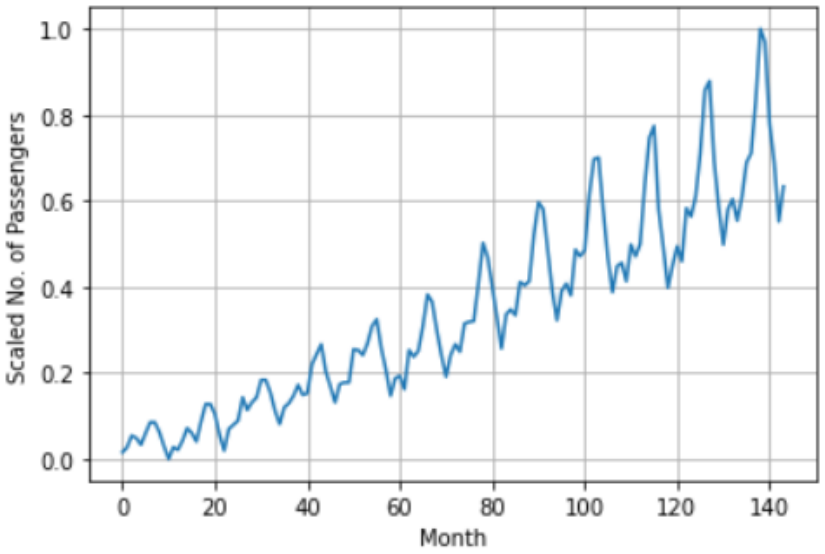
- **Forget Gate, f_t:** decides how much the previous state h_{t-1} is remembered now.
- **Input Gate, i_t:** decides how much new information is retained.
- **Output Gate, o_t:** decides how the next hidden state is computed.
- **Cell state, c_t:** the key variable in LSTM that learns long-term dependencies.

Source: Vasilev (2019). Advanced Deep Learning with Python. Packt Publishing.

Machine Learning: RNN vs GRU vs LSTM

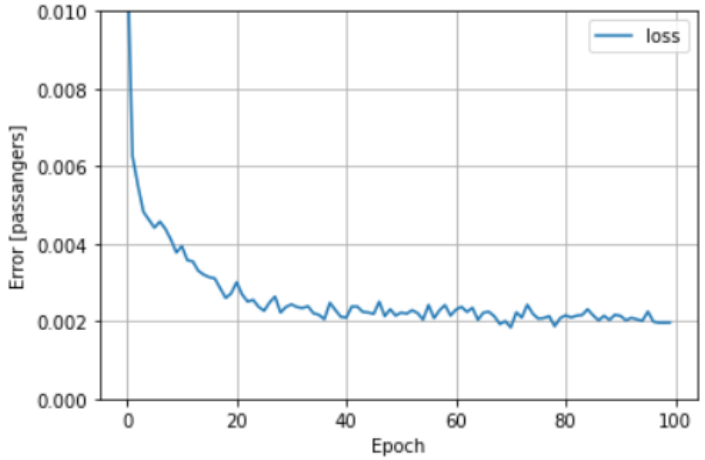
Example:

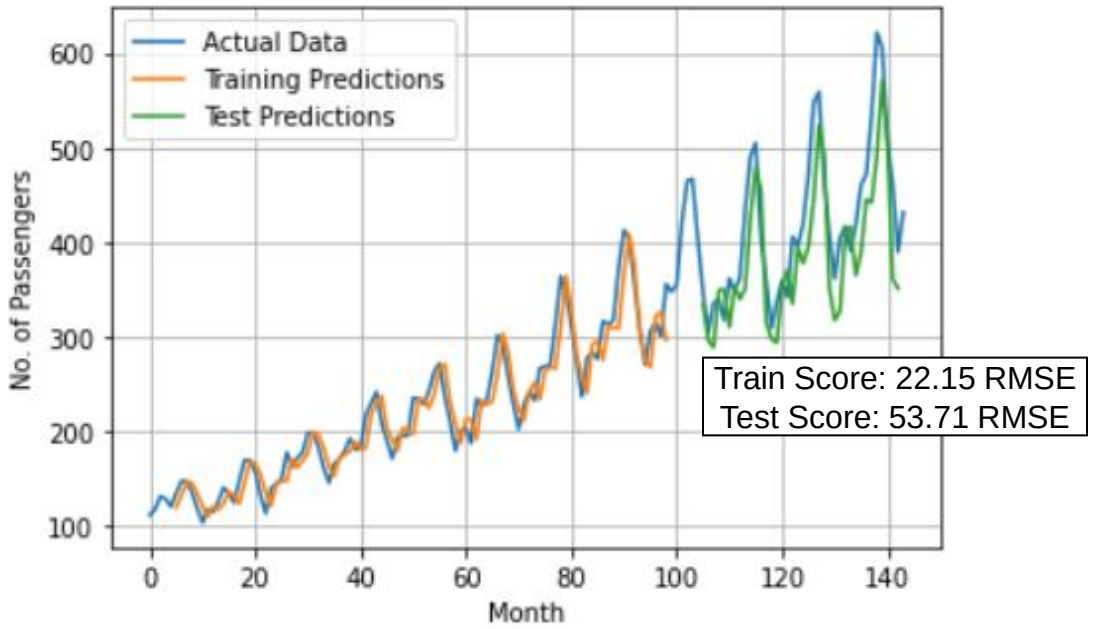
Train an LSTM-ANN for forecasting the last 30% of the data in the Airline Passengers Data Set. Use a single LSTM layer followed by a fully connected layer with 8 hidden neurons.




```
In [4]: model = Sequential()
        model.add(LSTM(8, input_shape=(1, look_back)))
        model.add(Dense(8))
        model.add(Dense(1))
        model.summary()
```

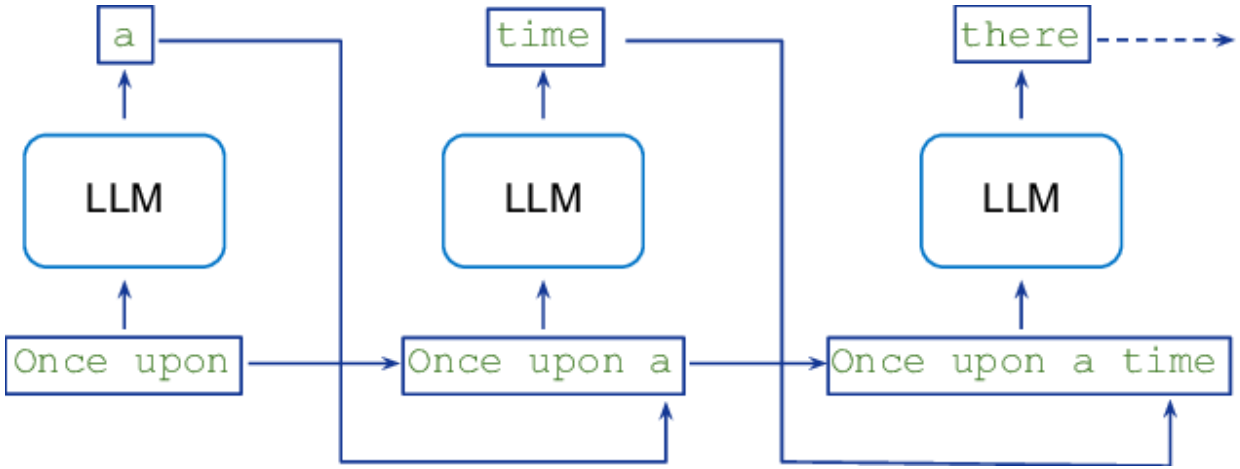
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 8)                 448
_________________________________________________________________
dense (Dense)                (None, 8)                 72
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 9
=================================================================
Total params: 529
Trainable params: 529
Non-trainable params: 0
_________________________________________________________________
```





Large Language Models (LLMs)

Side Note: Today's LLMs (like ChatGPT) are autoregressive LLMs.
LLM = Large Language Models



"Autoregressive LLMs are doomed. They cannot be made factual, non-toxic, etc.

They are not controllable."

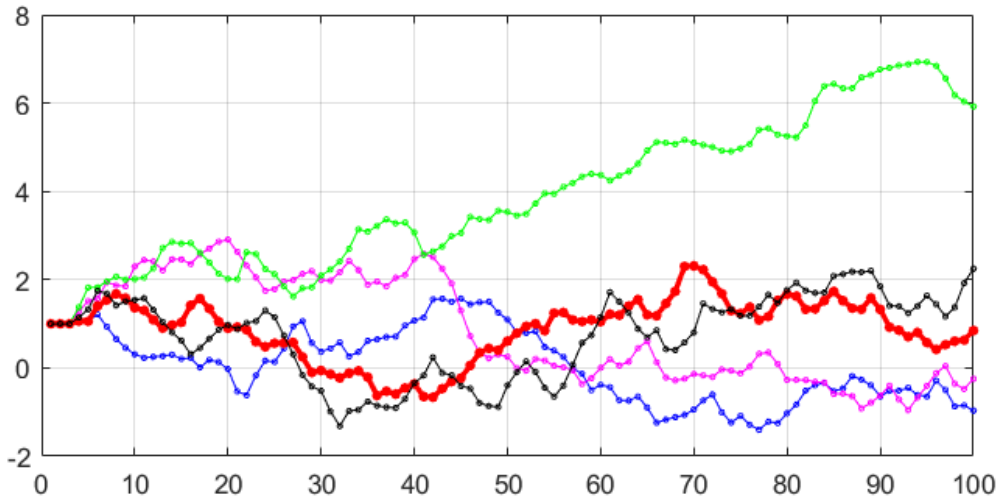
- Yann LeCun (2023)
Chief AI Scientist, Facebook AI Research
Turing Award Winner

Outline

- Time Series
  - Univariate Time Series Models
    - Linear Regression (AR models)
    - Machine Learning
  - **Multivariate Time Series Models**
    - Vector AR models
    - State-space models

Vector Autoregressive (VAR) Models

For multivariate time series, AR, ARX, and ARMA models become VAR, VARX, VARMA. We can now model *joint dynamic behavior*.



Note: The multivariate ARIMA model is called a co-integration model or a vector error correction (VEC) model.

AR (Autoregressive Model) $AR(p)$

$$y(k) = a_0 + \sum_{i=1}^{p} a_i y(k-i) + e(k)$$

VAR (Vector Autoregressive Model) $VAR(p)$

$VAR(p=1)$

$$\begin{bmatrix} y_1(k) \\ y_2(k) \\ y_3(k) \\ \vdots \end{bmatrix} = \begin{bmatrix} a_{0,0} \\ a_{0,1} \\ a_{0,2} \\ \vdots \end{bmatrix} + \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots \\ a_{21} & a_{22} & a_{33} & \cdots \\ a_{31} & a_{32} & a_{33} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} y_1(k-1) \\ y_2(k-1) \\ y_3(k-1) \\ \vdots \end{bmatrix} + \begin{bmatrix} e_1(k) \\ e_2(k) \\ e_3(k) \\ \vdots \end{bmatrix}$$

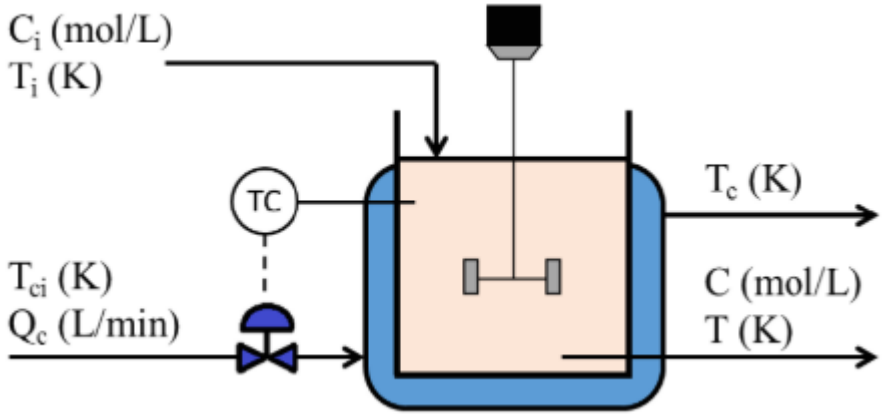
A more compact way to write the VAR(p) model:

$$\boldsymbol{y}_k = \boldsymbol{a}_0 + \sum_{i=1}^{p} \boldsymbol{\Phi}_i \boldsymbol{y}_{k-i} + \boldsymbol{e}_k$$

Vector Autoregressive (VAR) Models

Example:

Simulated Continuous Stirred Tank Reactor (CSTR)



Fault scenarios:

1. Sensor Drift
2. Fouling

https://uk.mathworks.com/matlabcentral/fileexchange/66189-feedback-controlled-cstr-process-for-fault-simulation

Input

Output

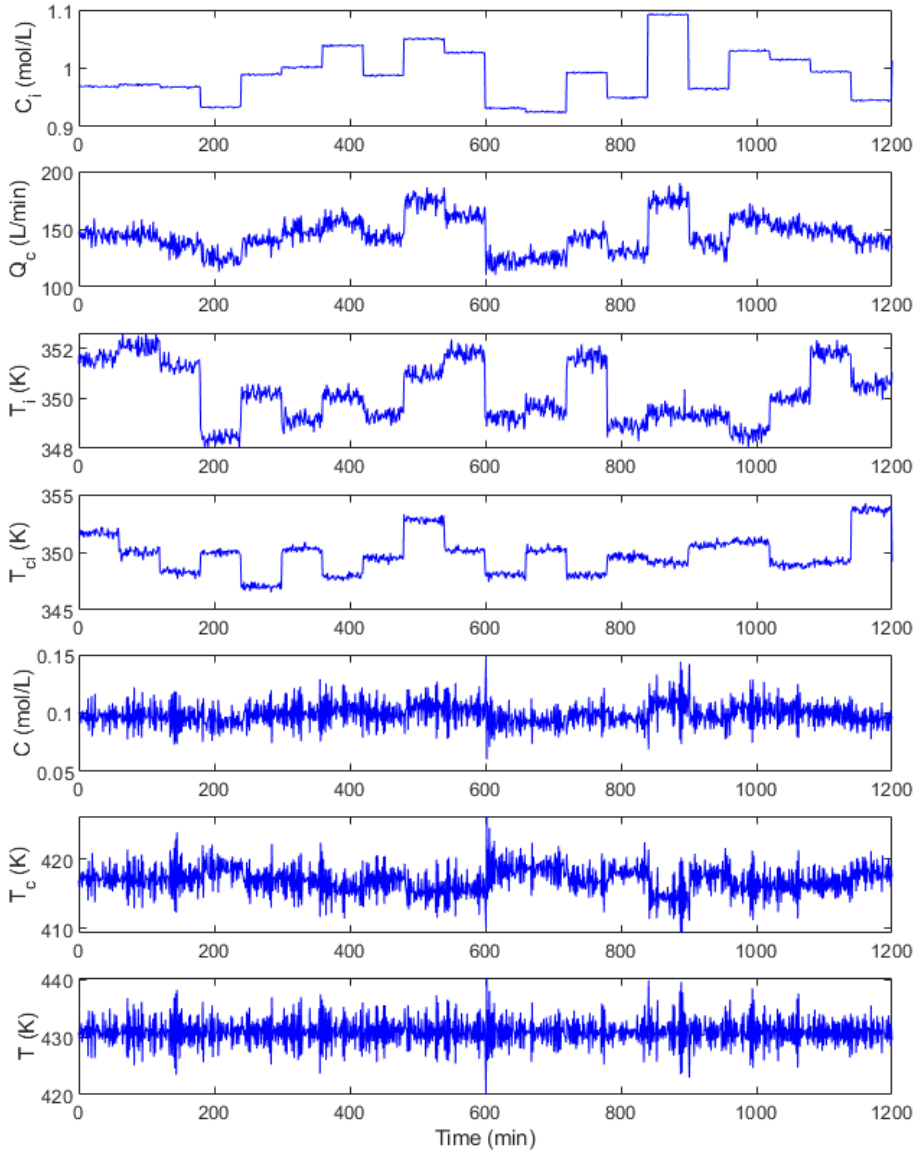
Input

Input

Output

Output

Output



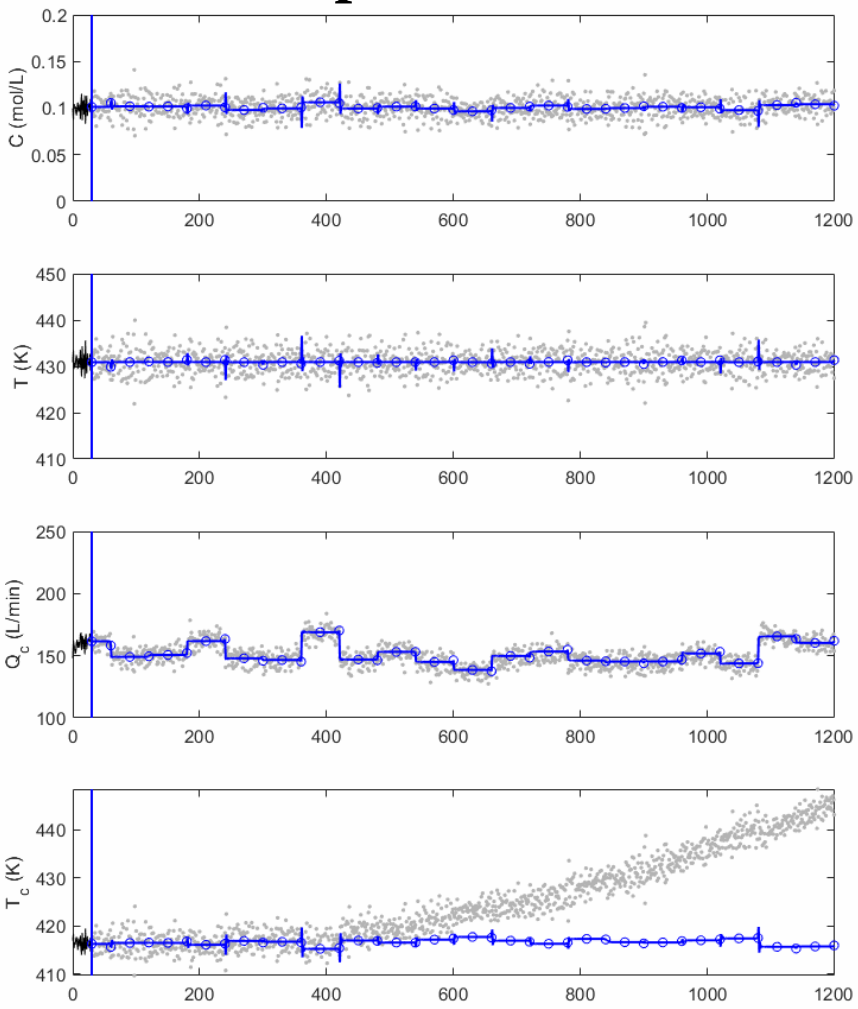
Vector Autoregressive (VAR) Models

Example:

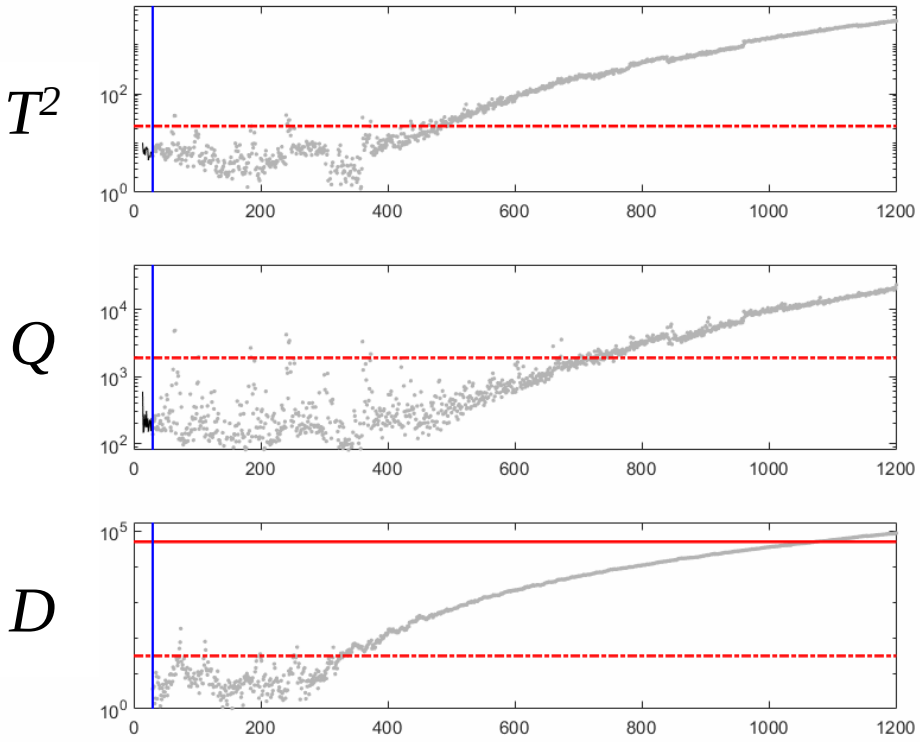
Sensor Drift in T_c

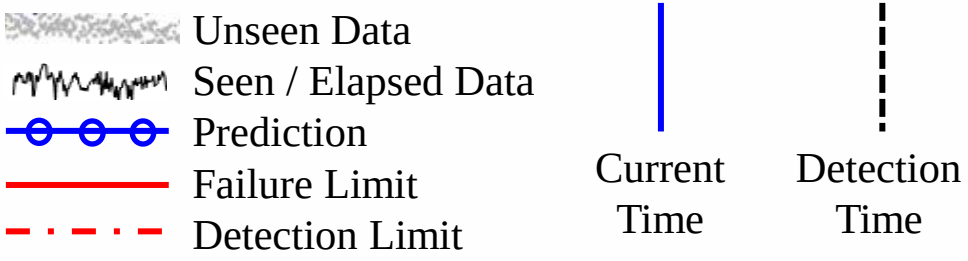
FAULTY VARIABLE

Output Variables

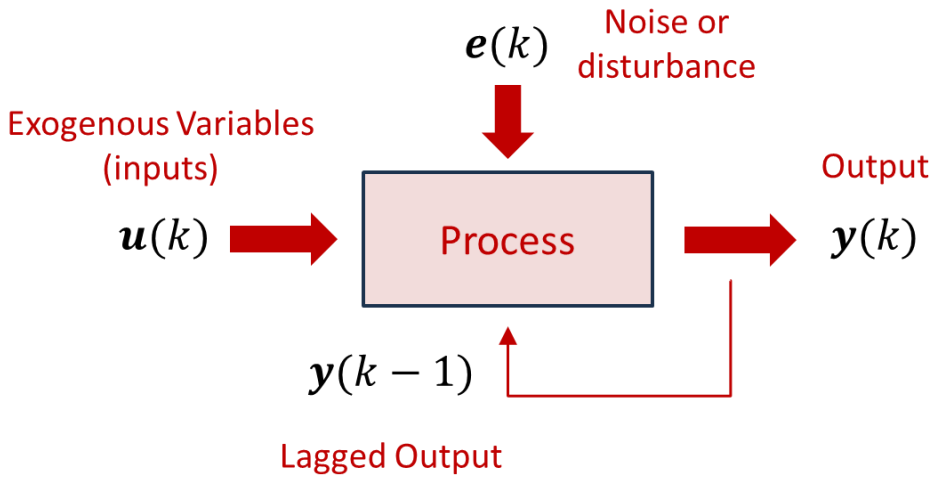


Monitoring Charts





State-space Models



So far, all our models involve only:

$u(k)$ = exogeneous inputs; those we can manipulate

$y(k)$ = outputs; those we can physically measure

$e(k)$ = output noise

But for processes with hidden dynamics, we also need:

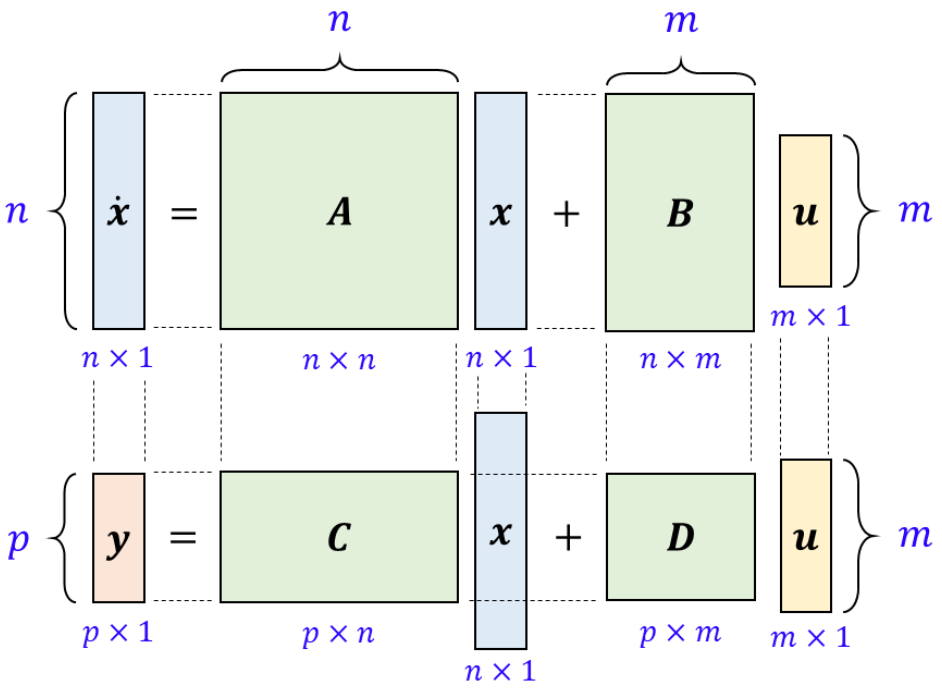
$x(k)$ = states; hidden variables that may or may not be measured physically

The linear time-invariant state-space model is equivalent to a VARMAX (vector autoregressive, moving average model with exogeneous inputs) plus state variables.

State-space Model

A linear time-invariant (**LTI**) state-space model can be written as:

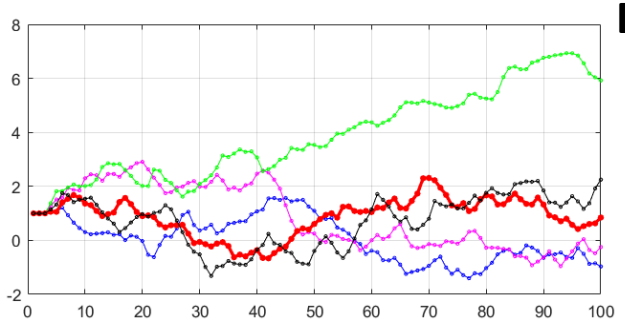
$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$



State-space Models

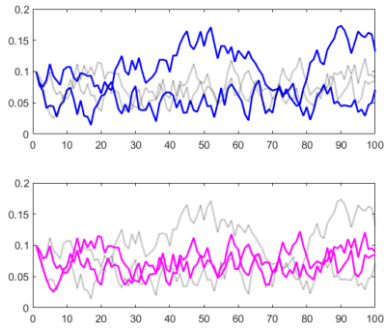
How to identify state-space models from time series data?

Subspace Identification Methods



Dimensionality Reduction

Estimate the states first

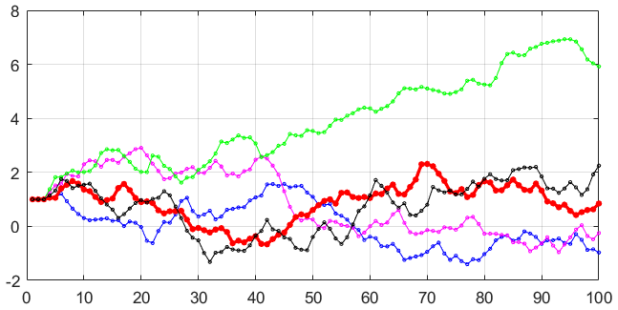


$\boldsymbol{x}(k)$

Regression

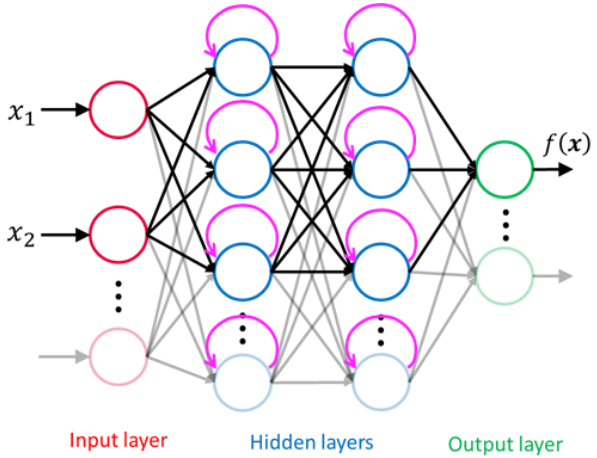
$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}$$
$$\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u}$$

Prediction Error Methods



Regression

The architecture must contain internal states.



What about high-frequency time series?

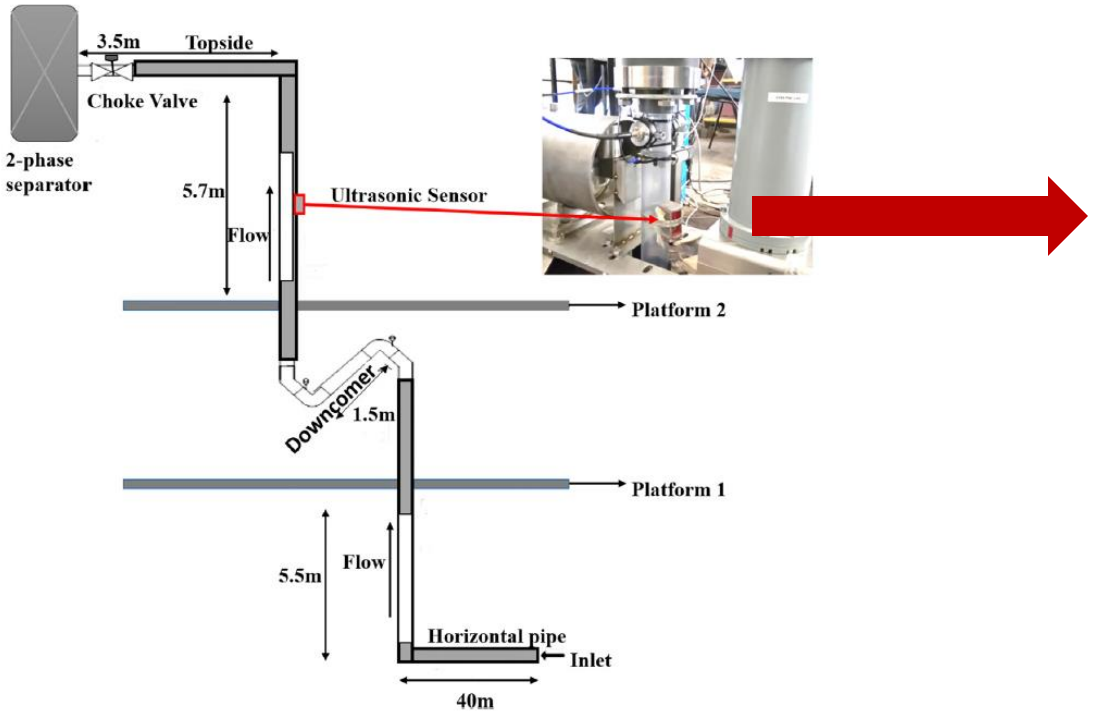
High-frequency data (e.g. 1 kHz) are also common in engineering:

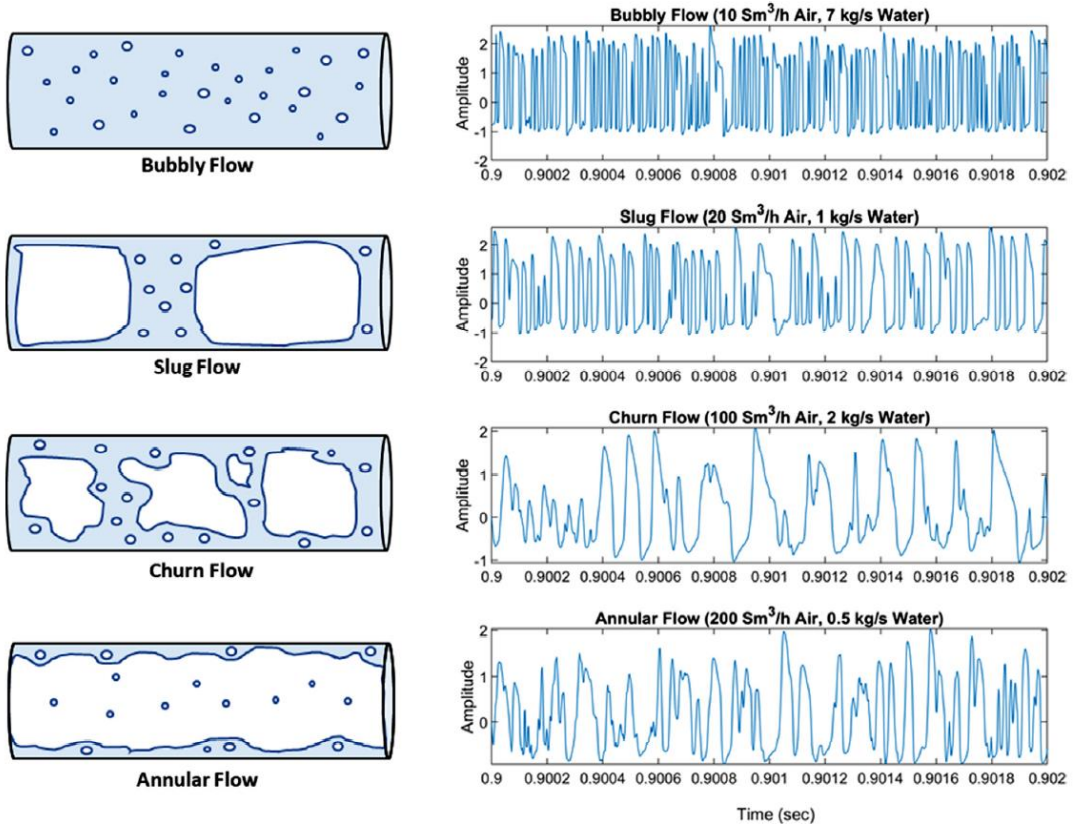
- Vibration data from pumps, pipes, buildings
- Flow measuring devices such as pressure, ultrasonic data

In these cases, it does not make sense to predict each data point.

These can be handled by signal decomposition techniques:

- Power Spectral Density
- Discrete / Continuous Wavelet Transform
- Empirical Mode Decomposition





Outline

- Time Series
  - Univariate Time Series Models
    - Linear Regression (AR models)
    - Machine Learning
  - Multivariate Time Series Models
    - Vector AR models
    - State-space models