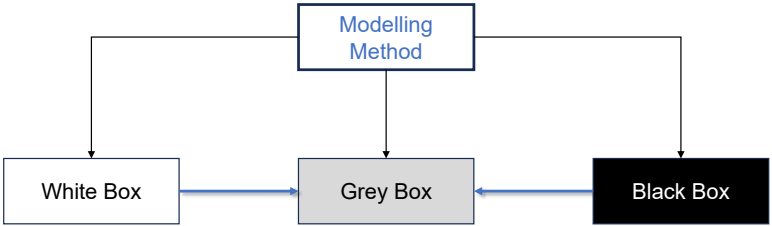

DTU

Industrial IoT for Digitization of Electronis Assets

Agenda

Model Types



White Box Models

- The structure of the model is known and is developed from fundamental laws of physics, thermodynamics, and heat transfer;
- The model parameters are well knownand used as inputs to the model.

Grey Box Models

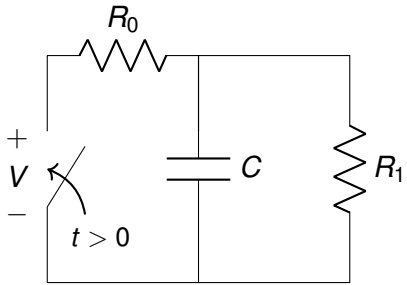
- Combination of black box and white box models.
- When data-driven models (black box) are partly supported by explicit models (equations) with a physical meaning.

Black Box Models

- Purely data-driven/statistical/empirical models
- Uses mathematical equations from statistics to map influential inputs to the outputs.
- The source of data: on-board monitored data from sensors, data simulated from simulation tools, surveyor standard data from public benchmark datasets.

Example of White Box Model

We know the physics, so we can write the equations:



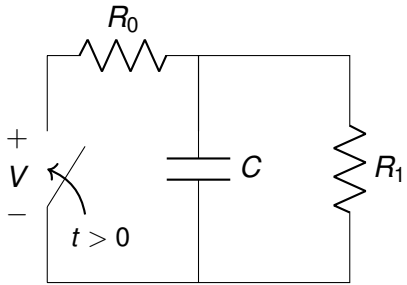
$$t \geq 0$$

$$I = \frac{1}{R_1} V_C + C \frac{d}{dt} V_C$$

$$V = R_0 I + V_C$$

Example of White Box Model

We know the physics, so we can write the equations:



$$\boldsymbol{t \geq 0}$$

$$I = \frac{1}{R_1} V_C + C \frac{d}{dt} V_C$$

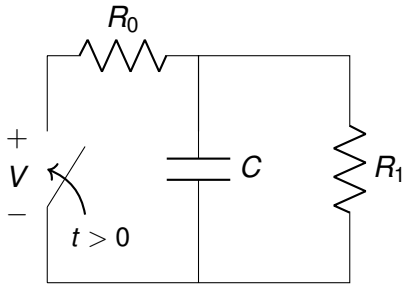
$$V = R_0 I + V_C$$

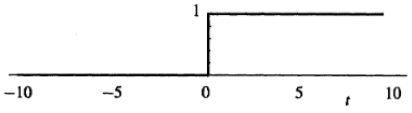
$$\boldsymbol{t \geq 0}$$

$$V + CR_1 \frac{d}{dt} V = (R_0 + R_1) I + CR_0 R_1 \frac{d}{dt} I$$

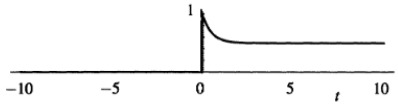
Example of White Box Model

The differential equation can be solved analitically, thus finding *I* has an exponential reponse





(a) Voltage step function.



(b) The current response.

Example of Grey Box Model

Stochastic Models

$$dx_t = f(x_t, u_t, t, \theta) + \sigma(u_t, t, \theta)d\omega$$
$$y_k = h(x_k, u_k, t_k, \theta) + e_k$$

Suitable for both linear, non-linear models. Used to represent systems influenced by both deterministic and stochastic components, accounting for random fluctuations in the system and uncertainties.
CTSM-R is a great R package from DTU Compure to fit *sde*.

Example of Grey Box Model

Stochastic Models

$$dx_t = f(x_t, u_t, t, \theta) + \sigma(u_t, t, \theta)d\omega$$
$$y_k = h(x_k, u_k, t_k, \theta) + e_k$$

Suitable for both linear, non-linear models. Used to represent systems influenced by both deterministic and stochastic components, accounting for random fluctuations in the system and uncertainties.
CTSM-R is a great R package from DTU Compure to fit *sde*.

PINNs:Physics-informed neural networks

Gaining attention in recent years, as a class of machine learning techniques that combine data-driven neural networks with physical equations to solve complex problems in various fields, such as physics, and engineering, minimizing the discrepancy between the predictions made by the neural network and physical equations.

Example of Black Box Models

A black box model is a system or algorithm that makes predictions or decisions based solely on input and output data, without an interpretable framework that can explain the connection between the inputs and the outputs.

Figure: Caption for the Image

AutoRegressive Model (AR)

Test

Let's consider the simplest form of an AR model:

$$y_t = ay_{t-1} + \epsilon$$

Where y_{k-1} is the value of the time series at time $t-1$ and ϵ an error term.

General

$$y_k = a_1 y_{t-1} + a_2 y_{t-2} + \cdots + a_n y_{t-n} + \epsilon_t = \sum_{i=1}$$

AutoRegressive eXogenous Model (ARX)

Given an **LTI** system, with $\boldsymbol{y_t}$ the output signal *with exogenous* input $\boldsymbol{u_t}$ with delay k, an ARX model can be defined as follows:

$$y(t) = c + a_1 y(t-1) + \cdots + a_{n_a} y(t-n_a) + \\ + b_1 u(t-n_k) + \cdots + b_{n_b} u(t-n_k-n_b-1) + \epsilon(t)$$

Or in a more compact form:

$$y(t) = \varphi' \theta + \epsilon(t)$$

Where ϕ is the regressor vector, θ is the paramters vector to estimate, and (white noise process).

AutoRegressive eXogenous Model (ARX)

Test

Given an LTI system, with y_t the output signal with exogenous input u_t of delay k, an ARX model can be defined as follows:

$$\boxed{y(t) = \varphi'\theta + \epsilon(t)}$$

$$\begin{aligned}
\theta &= [a_1, \ldots, a_{n_a}\ b_1, \ldots, b_{n_b}]' \\
\varphi(t) &= [y(t-1), \ldots, y(t-n_a) \quad u(t-n_k) \ldots, u_{t-n_k-n_b+1}]' \\
\epsilon(t) &\sim \mathcal{N}(0, \sigma^2)
\end{aligned}$$

Paramters Estimation

Therefore, for a SISO (Single Input Single Output) time series, y_{t+1} can be written as:

$$\hat{y}(t+1|\theta) = \varphi(t+1)'\theta$$

- φ' is the regressor vector
- $\epsilon(t)$ is the error

- φ' is the regressor vector
- $\epsilon(t)$ is the error
- θ is the coefficients vector (to be estimated)

Parameters Estimation

We don't know θ yet, but we have collected a set Z^N of measured data.

$$Z^N = \{u(-n), y(-n) \dots u(N-1), y(N-1)\},\ n = max\{n_a, n_b + n_k - 1\}$$

Therefore, we use the least squared error to find the optimal paramters vector θ^* that minimize the prediction error between $\hat{y}(t+1|\theta)$ and $y(t)$, namely:

$$\theta^* = \arg\min_{\theta}\{\mathcal{L}(\theta, Z^N)\}$$

$$\mathcal{L}(\theta, Z^N) = \frac{1}{N}\sum_{k=0}^{N-1}(y(k) - \hat{y}(t|\theta))^2 = \frac{1}{N}\sum_{k=0}^{N-1}(y(k) - \varphi'(t)\theta)^2$$

Paramters Estimation

$\mathcal{L}(\theta, Z^N)$ is a quadratic function of θ. Therefore, θ^* can be found by zeroing The derivative of $\mathcal{L}$.

$$\frac{d}{d\theta}\mathcal{L}_N(\theta, Z^N) = \frac{2}{N}\sum_{k=0}^{N-1}\varphi(t)(y(k) - \varphi'(t)\theta) = 0$$

Thus, obtaining:

$$\sum_{k=0}^{N-1}\varphi(t)y(k) = \sum_{k=0}^{N-1}\varphi(t)\varphi'(k)$$

Finally, the θ^* can be derived:

Paramters Estimation

Thus, obtaining:

$$\sum_{k=0}^{N-1} \varphi(t) y(k) = \sum_{k=0}^{N-1} \varphi(t) \varphi'(k)$$

Finally, θ^* can be derived:

$$\theta^* = \left[\sum_{k=0}^{N-1} \varphi(t) \varphi'(t) \right]^{-1} \left[\sum_{k=0}^{N-1} \varphi(t) y(t) \right]$$

Generate SISO data

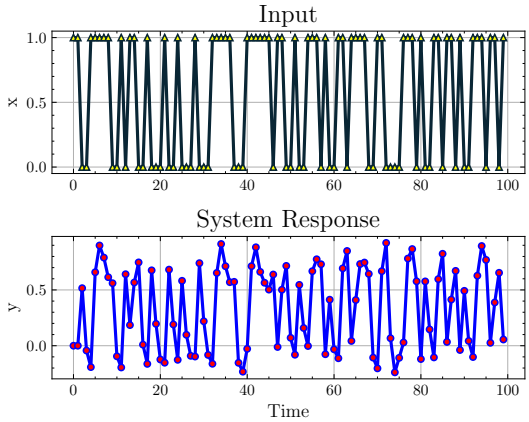
```python
def generate_siso_data(n, noise_level=0.1, a1=0.5, a2=-0.3, b1=0.7, b2=-0.2):
    """
    Generates synthetic Single Input Single Output (SISO) data for an ARX model with na=2 and
    :param n: Number of data points to generate.
    :param noise_level: Standard deviation of the noise.
    :param a1, a2: Coefficients for the autoregressive part of the model.
    :param b1, b2: Coefficients for the exogenous input part of the model.
    :return: Tuple of (y, u), where y is the target series and u is the exogenous series.
    """
    # Generating exogenous input (x) as a random signal
    x = np.random.randint(0, 2, size=n)

    # Generating the target series (y)
    y = np.zeros(n)
    for t in range(2, n):
        y[t] = a1 * y[t-1] + a2 * y[t-2] + b1 * x[t-1] + b2 * x[t-2] + np.random.normal(0, no

    return y, x
```

Visualize the data

```
y, x = generate_siso_data(n=100, noise_level=0.05)
```



Fit an ARX Model

```python
def fit_arx_model(y, u, na, nb):
    """
    Fits an ARX model to the given data and returns the coefficients of A, B, and the interce
    :param y: Target time series.
    :param u: Exogenous time series.
    :param na: Order of the autoregressive part.
    :param nb: Order of the exogenous input part.
    :return: Coefficients of A, B and the intercept of the ARX model.
    """
    max_lag = max(na, nb)
    features = []
    target = y[max_lag:]

    for i in range(max_lag, len(y)):
        lag_y = y[i-na:i] if na > 0 else []
        lag_u = u[i-nb:i] if nb > 0 else []
        row = np.concatenate([lag_y, lag_u])
        features.append(row)

    U = np.array(features)
    model = LinearRegression().fit(U, target)
    return model.coef_[:na], model.coef_[na:], model.intercept_
```

Estimated Paramters

```
na, nb = 2, 2
coefficients_a, coefficients_b, intercept = fit_arx_model_with_intercept(y, x, na, nb)
```

Regressors	Parameters
intercept	0.0035
$x(t-1)$	0.7112
$x(t-2)$	-0.2089
$y(t-1)$	0.5071
$y(t-2)$	-0.3255

Estimated Paramters

```
na, nb = 2, 2
coefficients_a, coefficients_b, intercept = fit_arx_model_with_intercept(y, x, na, nb)
```

Regressors	Parameters
intercept	0.0035
$x(t-1)$	0.7112
$x(t-2)$	-0.2089
$y(t-1)$	0.5071
$y(t-2)$	-0.3255

Are these parameters similar to the ones previously defined?

SysIdentPy: A Python for System identification

SysIdentPy is a Python library designed for linear and non-linear system identification.

- FROLS $\rightarrow$ Forward Regression Orthogonal Least Squares.
- Particularly effective in nonlinear system.
- Itertively adding new terms to the model, creating a pool of candidate terms (linear, nonlinear) based on the input data.
- The orthogonalization process multicollinearity.

SysIdentPy: A Python for System identification

In this example we test this library on our SISO system to fit an ARX fit, manually selecting the order of the model.

```
from sysidentpy.model_structure_selection import FROLS
from sysidentpy.basis_function._basis_function import Polynomial

basis_function = Polynomial(degree=1)

model = FROLS(
    order_selection=False,
    n_terms=5,
    extended_least_squares=False,
    ylag=2,
    xlag=2,
    estimator="least_squares",
    basis_function=basis_function,
)

model.fit(X=x.reshape(-1,1), y=y.reshape(-1,1))
```

Estimated Coeff. Our Models

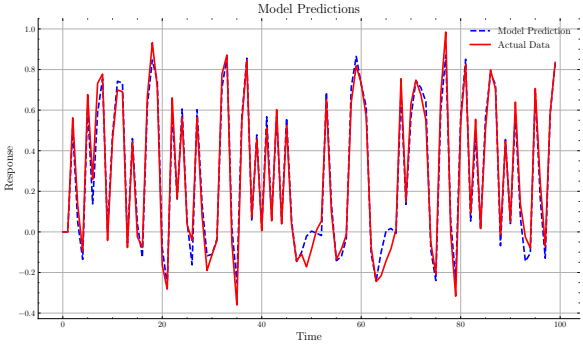
Regressors	Parameters
intercept	0.0035
$x(t-1)$	0.7112
$x(t-2)$	-0.2089
$y(t-1)$	0.5071
$y(t-2)$	-0.3255

Estimated Coeff. FROLS

Regressors	Parameters
intercept	0.0115
$x(t-1)$	0.7034
$x(t-2)$	-0.1962
$y(t-1)$	0.4883
$y(t-2)$	-0.3367

Forecast using SysIdentPy

```
yhat = model.predict(X=x.reshape(-1,1), y=y.reshape(-1,1)).squeeze()
```



RMSE: Root Mean Squared Error

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{1}$$

```
from sklearn.metrics import mean_squared_error
MSE = mean_squared_error(y, yhat) # Mean Squared Error
RMSE = np.sqrt(MSE) #Root Mean Squared Error
```

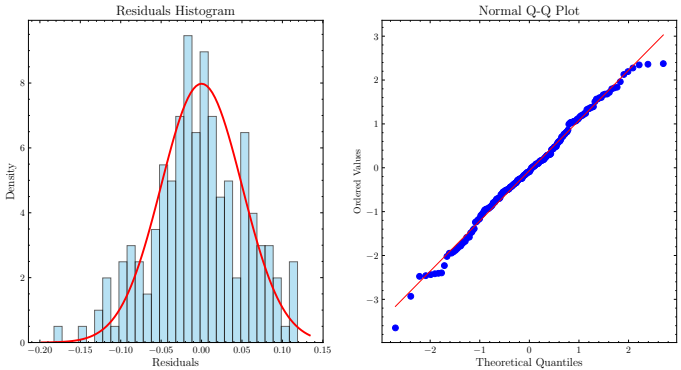
Do you rember the assumtion on $\epsilon(t)$?

We previously assumed that the residual of the model, namely $(y - \hat{y})$ are distributed as $\epsilon(t) \sim \mathcal{N}(0, \sigma^2)$.

In our example we have introduce a distrurbance modeled as gaussian noise with $\sigma = 0.01$ If the model is good enough, we error will coincide with the gaussian noise we introduced.

Model Validation

Shapiro-Wilk test confirmed that the residuals are white noise and the Q-Q Plot below shows that the distribution of the residual is compatible with the noise introduced in our data $\sim \mathcal{N}(0, \sigma^2 = 0.05)$.



Do you rember the assumtion on $\epsilon(t)$?