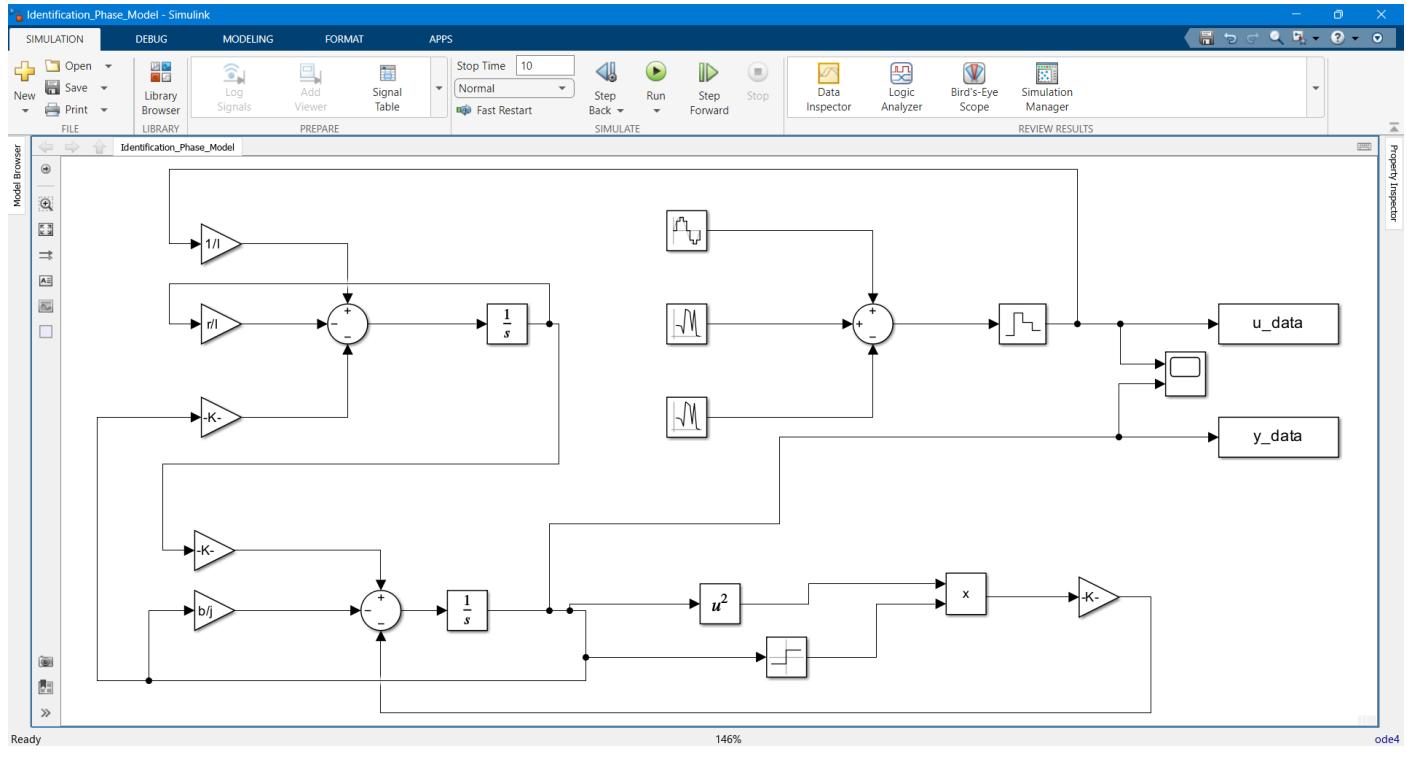
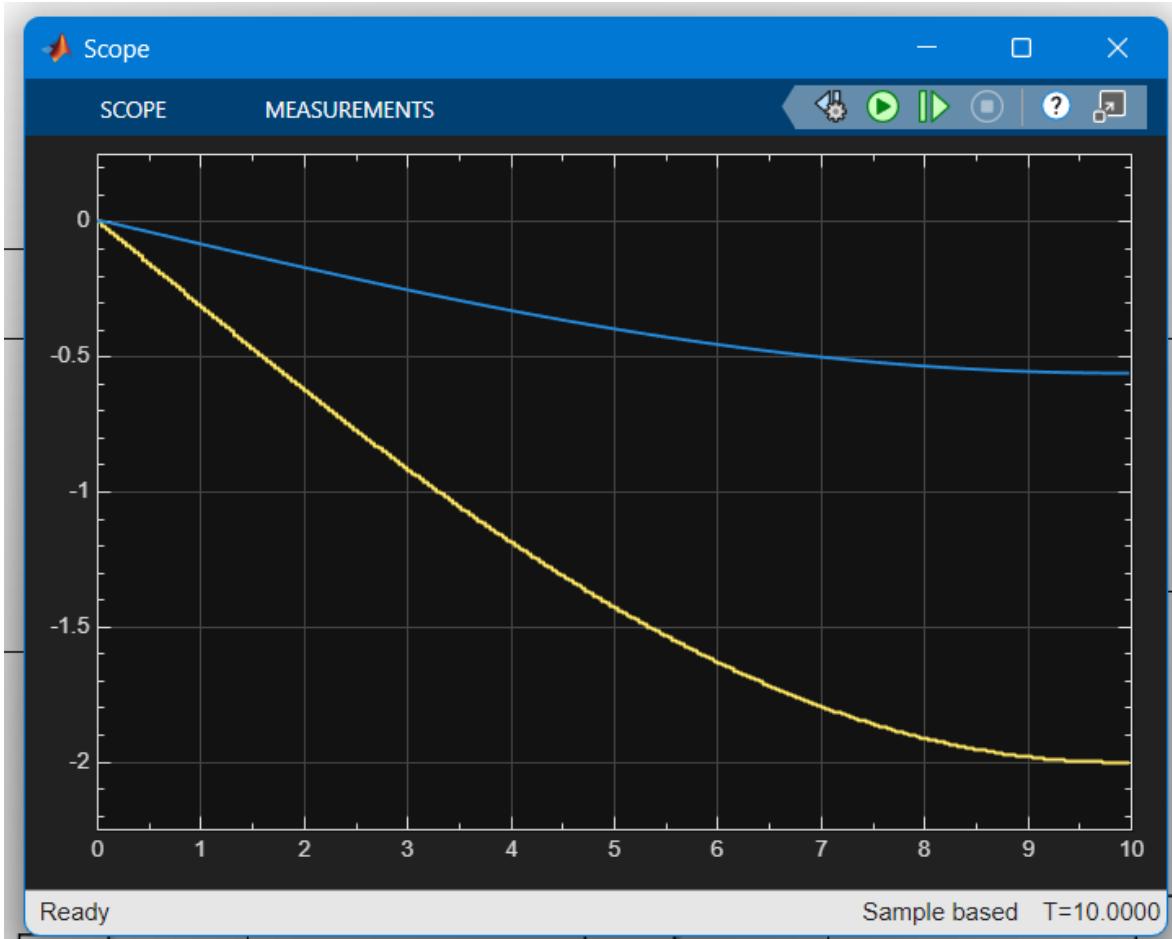


# ANN Control of DC Motor with Nonlinear Pump Load

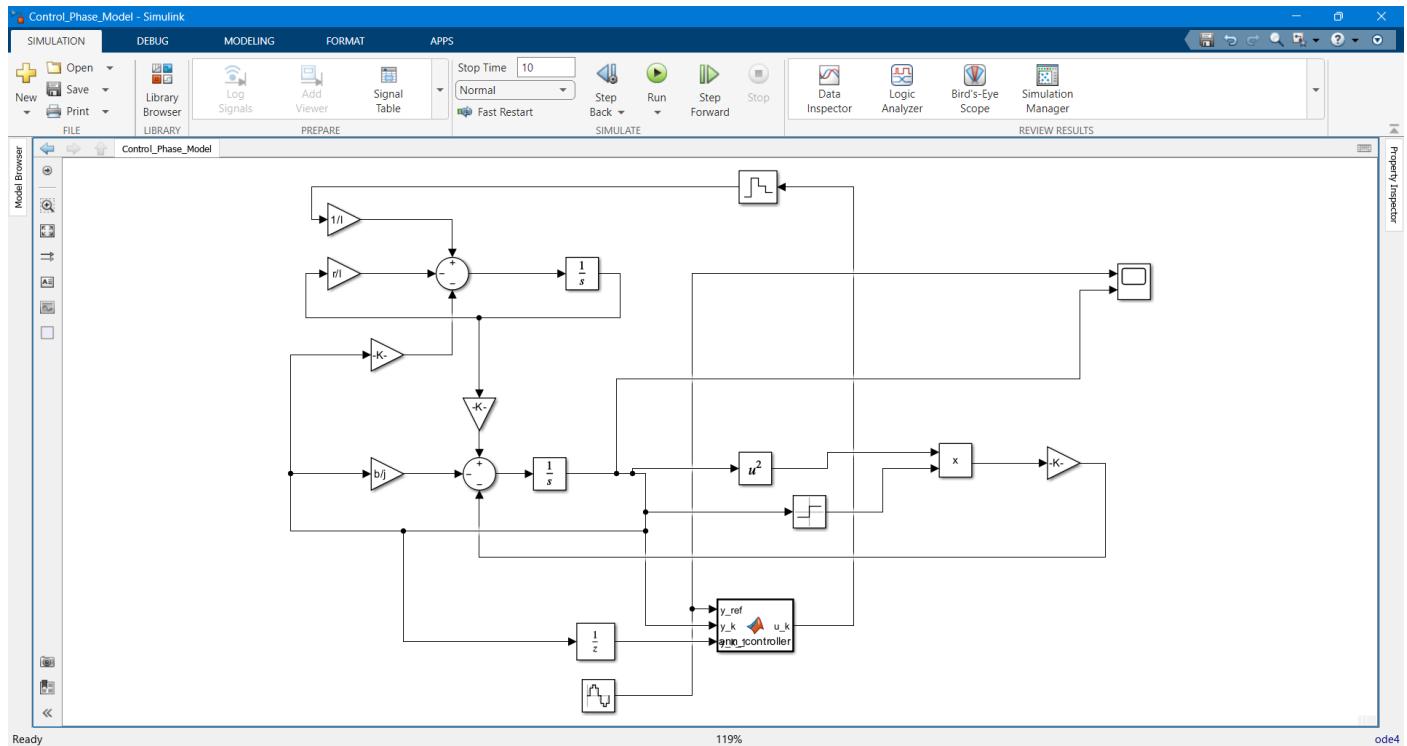
## Identification phase model and scope



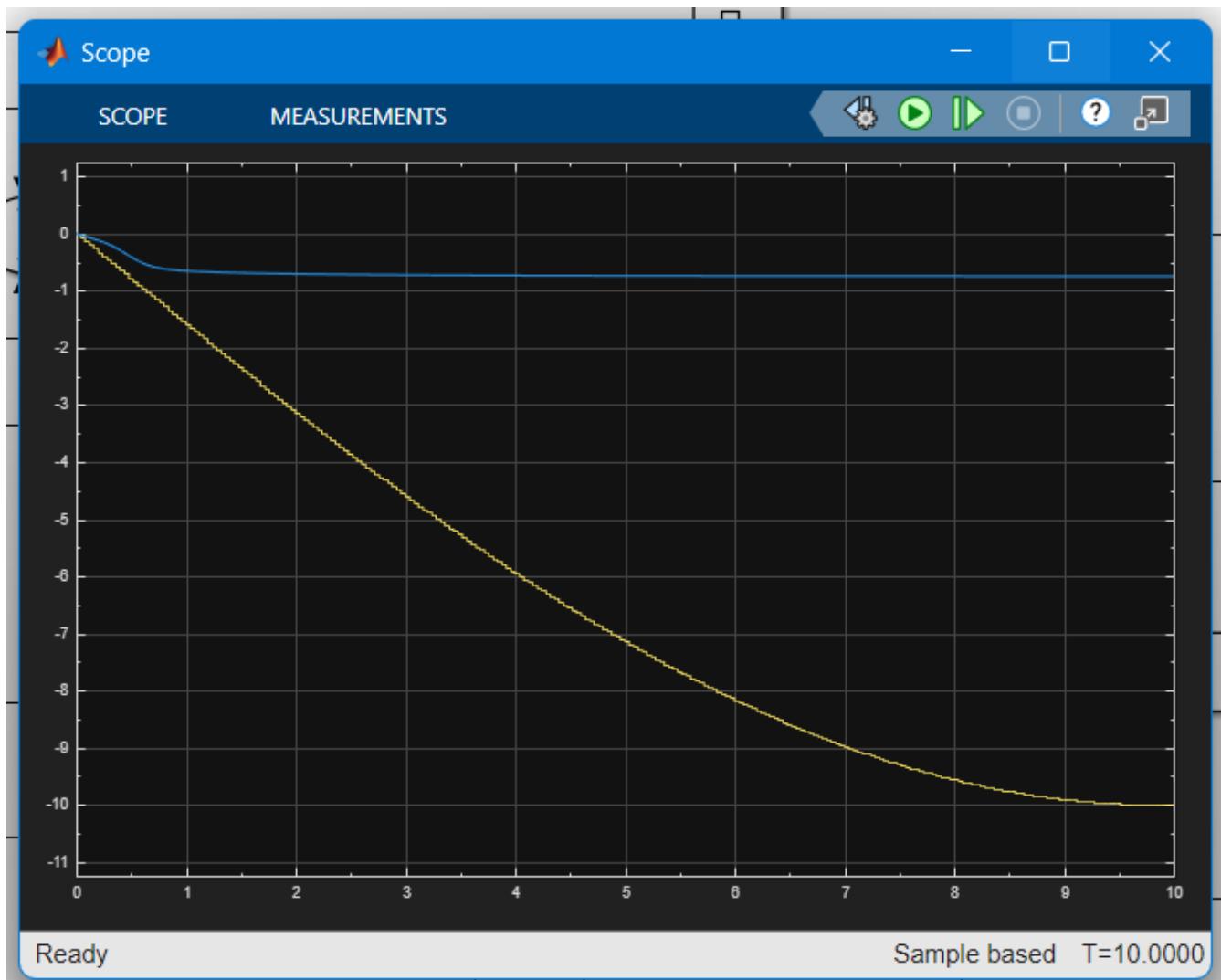
scope



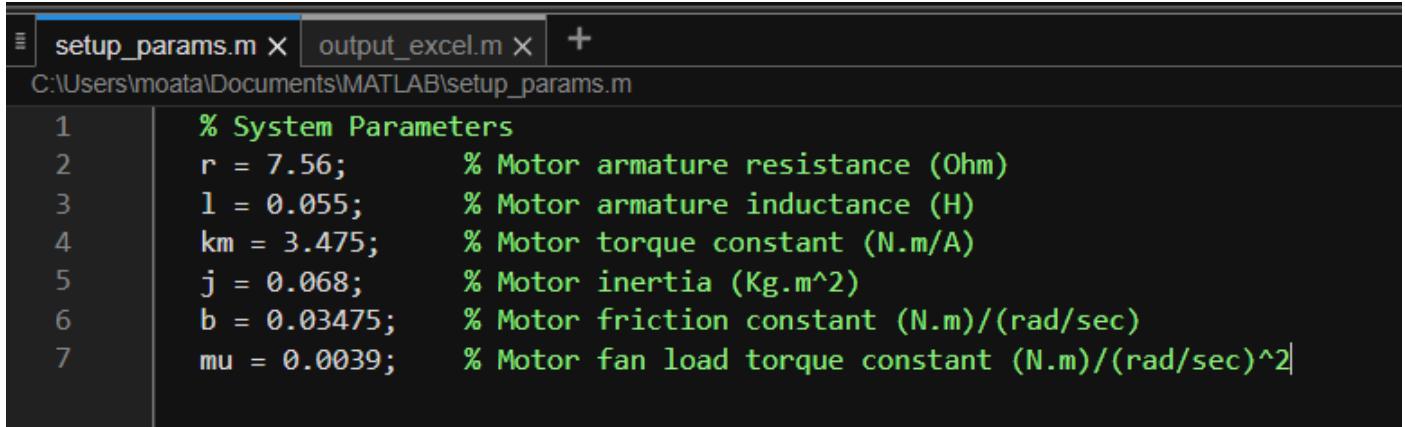
## Control phase model and scope



scope



## M-files

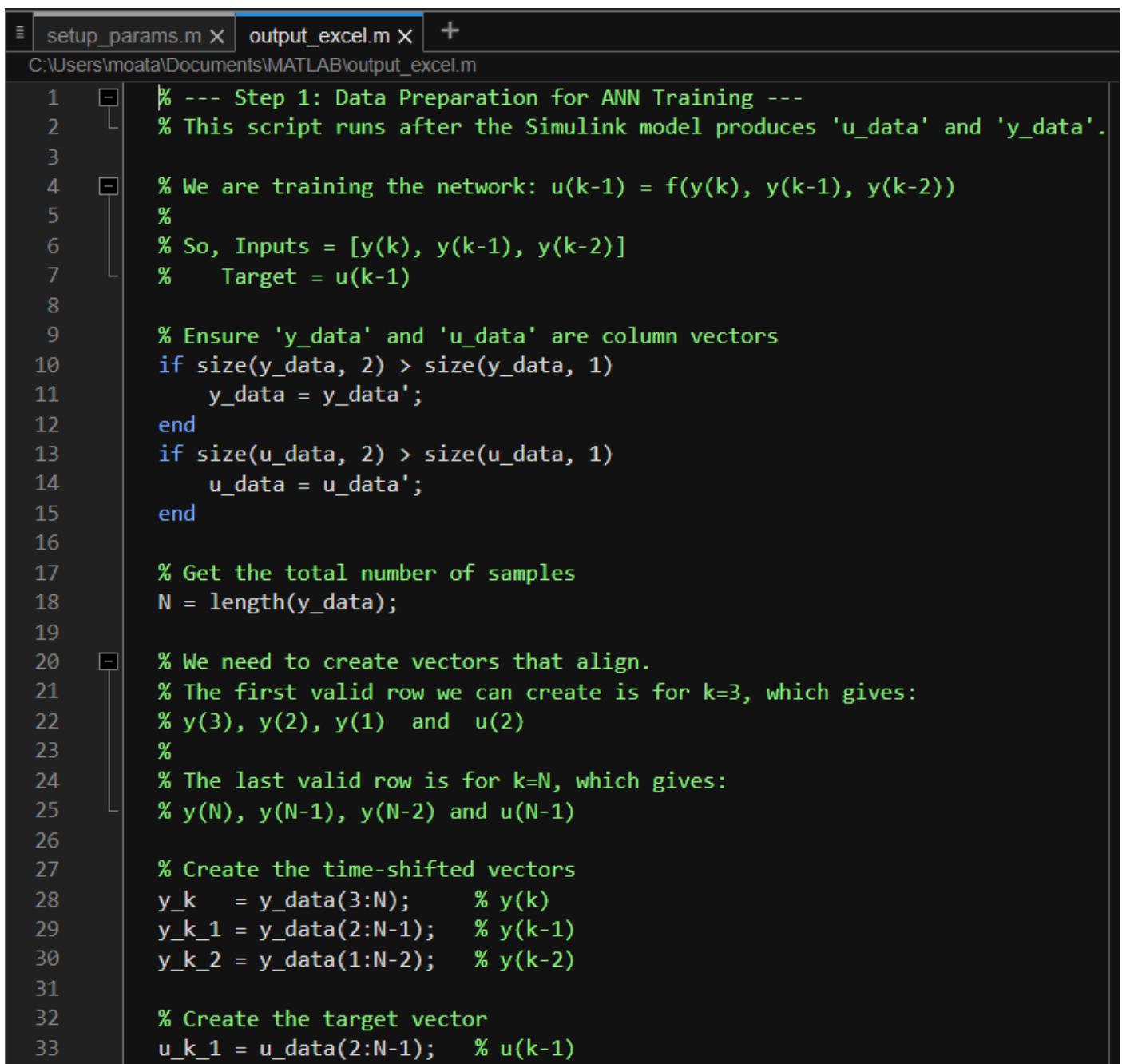


The screenshot shows the MATLAB Editor window with the file `setup_params.m` open. The code defines various motor parameters:

```

1 % System Parameters
2 r = 7.56;           % Motor armature resistance (Ohm)
3 l = 0.055;          % Motor armature inductance (H)
4 km = 3.475;         % Motor torque constant (N.m/A)
5 j = 0.068;          % Motor inertia (Kg.m^2)
6 b = 0.03475;        % Motor friction constant (N.m)/(rad/sec)
7 mu = 0.0039;        % Motor fan load torque constant (N.m)/(rad/sec)^2

```



The screenshot shows the MATLAB Editor window with the file `output_excel.m` open. The code performs data preparation for ANN training, specifically handling time-shifted inputs and target vectors:

```

1 % --- Step 1: Data Preparation for ANN Training ---
2 % This script runs after the Simulink model produces 'u_data' and 'y_data'.
3
4 % We are training the network: u(k-1) = f(y(k), y(k-1), y(k-2))
5 %
6 % So, Inputs = [y(k), y(k-1), y(k-2)]
7 % Target = u(k-1)
8
9 % Ensure 'y_data' and 'u_data' are column vectors
10 if size(y_data, 2) > size(y_data, 1)
11     y_data = y_data';
12 end
13 if size(u_data, 2) > size(u_data, 1)
14     u_data = u_data';
15 end
16
17 % Get the total number of samples
18 N = length(y_data);
19
20 % We need to create vectors that align.
21 % The first valid row we can create is for k=3, which gives:
22 % y(3), y(2), y(1) and u(2)
23 %
24 % The last valid row is for k=N, which gives:
25 % y(N), y(N-1), y(N-2) and u(N-1)
26
27 % Create the time-shifted vectors
28 y_k    = y_data(3:N);      % y(k)
29 y_k_1  = y_data(2:N-1);    % y(k-1)
30 y_k_2  = y_data(1:N-2);    % y(k-2)
31
32 % Create the target vector
33 u_k_1 = u_data(2:N-1);    % u(k-1)

```

```

setup_params.m x output_excel.m x +
C:\Users\moata\Documents\MATLAB\output_excel.m

34
35 % Check for length consistency (this should match)
36 % We have N-2 samples for training
37 if length(y_k) ~= length(u_k_1)
38     error('Vector lengths do not match. Check data simulation.');
39 end
40
41 % Separate into final inputs and targets
42 ann_inputs = [y_k, y_k_1, y_k_2]; % Columns [y(k), y(k-1), y(k-2)]
43 ann_targets = u_k_1; % Column [u(k-1)]
44
45 disp('ANN training data created successfully.');
46 disp('Input matrix size (ann_inputs):');
47 disp(size(ann_inputs));
48 disp('Target vector size (ann_targets):');
49 disp(size(ann_targets));
50
51
52 % --- Step 2: Export Data to Excel ---
53 % Combine inputs and targets for easy export
54 output_table = array2table([ann_inputs, ann_targets], ...
55     'VariableNames', {'y_k', 'y_k_1', 'y_k_2', 'u_k_1'});
56
57 filename = 'ann_training_data.xlsx';
58 writetable(output_table, filename);
59
60 fprintf('Training data successfully exported to %s\n', filename);

```

## Python Scripts

```

train_ann_identifier.py x
train_ann_identifier.py > ...
1 import pandas as pd
2 import numpy as np
3 from sklearn.neural_network import MLPRegressor
4 from sklearn.preprocessing import StandardScaler
5 from sklearn.model_selection import train_test_split
6 from sklearn.metrics import mean_squared_error
7 import joblib
8 import warnings
9
10 # Suppress warnings
11 warnings.filterwarnings('ignore')
12
13 print("--- 1. Loading and Preparing Data ---")
14
15 # Load the data from Excel
16 data = pd.read_excel('ann_training_data.xlsx')
17
18 # Define features (X) and target (y)
19 features = ['y_k', 'y_k_1', 'y_k_2']
20 target = 'u_k_1'
21 X = data[features]
22 y = data[target]
23
24 print("--- 2. Scaling the Data ---")
25 scaler = StandardScaler()
26
27 # --- THIS IS THE FIX ---
28 # Fit the scaler on the NumPy array (.values), not the DataFrame (X)
29 # This trains the scaler WITHOUT feature names.
30 X_scaled = scaler.fit_transform(X.values)
31 #
32
33 # Save the new scaler
34 scaler_filename = 'ann_identifier_scaler.pkl'
35 joblib.dump(scaler, scaler_filename)
36 print(f'Data scaler (trained on NumPy) saved to '{scaler_filename}'')

```

```

train_ann_identifier.py X
train_ann_identifier.py > ...
23
24 print(" --- 2. Scaling the Data ---")
25 scaler = StandardScaler()
26
27 # --- THIS IS THE FIX ---
28 # Fit the scaler on the NumPy array (.values), not the DataFrame (x)
29 # This trains the scaler WITHOUT feature names.
30 X_scaled = scaler.fit_transform(x.values)
31 #
32
33 # Save the new scaler
34 scaler_filename = 'ann_identifier_scaler.pkl'
35 joblib.dump(scaler, scaler_filename)
36 print(f"Data scaler (trained on NumPy) saved to '{scaler_filename}'")
37
38
39 print("\n--- 3. Building and Training ANN Identifier ---")
40 ann_identifier = MLPRegressor(
41     hidden_layer_sizes=(10,),
42     activation='logistic',
43     solver='adam',
44     max_iter=1000,
45     random_state=42,
46     verbose=True
47 )
48
49 print("Starting ANN training...")
50 ann_identifier.fit(X_scaled, y)
51 print("Training complete.")
52
53 # Save the new model
54 model_filename = 'ann_identifier_model.pkl'
55 joblib.dump(ann_identifier, model_filename)
56 print(f"\nTrained ANN model saved to '{model_filename}'")

```

\* Control\_Phase\_Model/MATLAB Function - Simulink

SIMULATION DEBUG MODELING APPS FUNCTION

IO Go To Refactor Update Model Stop Time: 10 Normal Step Run Step Stop

Find Find CODE COMPILER

PREPARE NAVIGATE SIMULATE

MATLAB Function

Model Browser

```

function u_k = ann_controller(y_ref, y_k, y_k_1)
% This function implements the ANN Controller
% Based on the equation: u(k) = f(y_ref(k), y(k), y(k-1))

% --- Declare Python functions as extrinsic ---
coder.extrinsic('py.importlib.import_module');
coder.extrinsic('py.joblib.load');
coder.extrinsic('py.numpy.array');
coder.extrinsic('py.numpy.reshape');
coder.extrinsic('py.tuple');
coder.extrinsic('int32'); % For creating the reshape tuple
% ----

% --- Initialize output to define its size and type ---
u_k = 0.0;
% ----

% Use persistent variables
persistent scaler model reshaper_tuple;

% Load the model on the first step
if isempty(model)
    py.importlib.import_module('joblib');
    py.importlib.import_module('numpy');

    scaler = py.joblib.load('ann_identifier_scaler.pkl');
    model = py.joblib.load('ann_identifier_model.pkl');

    % Create the constant reshape tuple (1, -1)
    reshaper_tuple = py.tuple([int32(1), int32(-1)]);

    disp('Python ANN model and scaler loaded into Simulink.');
end

% --- Step 1. Format Inputs ---
X_live = [y_ref, y_k, y_k_1];

```

Control\_Phase\_Model/MATLAB Function - Simulink

SIMULATION DEBUG MODELING APPS FUNCTION

Edit Data Go To Refactor Update Model Stop Time 10 Normal Step Back Run Step Forward PREPARE NAVIGATE CODE COMPILE Fast Restart SIMULATE

Model Browser

```
23 py.importlib.import_module('joblib'),
24 py.importlib.import_module('numpy');
25
26 scaler = py.joblib.load('ann_identifier_scaler.pkl');
27 model = py.joblib.load('ann_identifier_model.pkl');
28
29 % Create the constant reshape tuple (1, -1)
30 reshaper_tuple = py.tuple([int32(1), int32(-1)]);
31
32 disp('Python ANN model and scaler loaded into Simulink.');
33 end
34
35 % --- Step 1. Format Inputs ---
36 X_live = [y_ref, y_k, y_k_1];
37
38 % Convert to a 1D Python (Numpy) array
39 X_py_1d = py.numpy.array(X_live);
40
41 % Reshape to 2D numpy array [1, 3]
42 X_py = py.numpy.reshape(X_py_1d, reshaper_tuple);
43
44 % --- Step 2. Scale the Inputs ---
45 % This will now work, as the scaler expects a NumPy array
46 X_scaled = feval('transform', scaler, X_py);
47
48 % --- Step 3. Predict the Control Signal ---
49 u_k_py = feval('predict', model, X_scaled);
50
51 % --- Step 4. Return the Output ---
52 u_k = double(u_k_py);
53
54 end
```

معتز أحمد سمير عبدالعاطي 2305223