Black-Box Modeling of Nonlinear Dynamics Using Polynomial ARX

By Rus Alexandru, Pal Robert and Suciu Andrei

Problem Statement

Develop a ***nonlinear ARX*** model to approximate the dynamics of an ***unknown system*** with one input and noisy outputs using polynomial regression.

Objectives:

- **Configure Polynomial ARX Model**
- **Identify Model Parameters**
- **Evaluate Predictive and Simulative Accuracy**
- **Analyze Model Performance**

Approximator Structure

- Polynomial order ***m*** is incrementally tested to optimize model accuracy.
- Built from delayed inputs and outputs, creating a ***regression matrix***.

$$\varPhi = [1, y(k-1), \ldots, y(k-na), u(k-nk), \ldots, u(k-nk-nb+1), y(k-1)^2, \ldots]$$

- Parameters ***θ*** are estimated using linear regression.

$$\theta = (\varPhi^\top \varPhi)^{-1} \varPhi^\top y \qquad \hat{y} = \varPhi\theta$$

Approximator Structure

- ***MSE Evaluation:***

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

- Optimal Model Selection:
  - Explore combinations of ***na,nb,m***.
  - Select the configuration with the ***lowest validation*** MSE.
  - Save the identified parameters for ***final predictions***.

Key features:

- The ***monomial term generator*** efficiently creates polynomial expansions for delayed inputs (u) and outputs (y).

Visualization of Polynomial Terms for Degree m = 2

$$Constant\ Term : \Phi_1 = 1$$

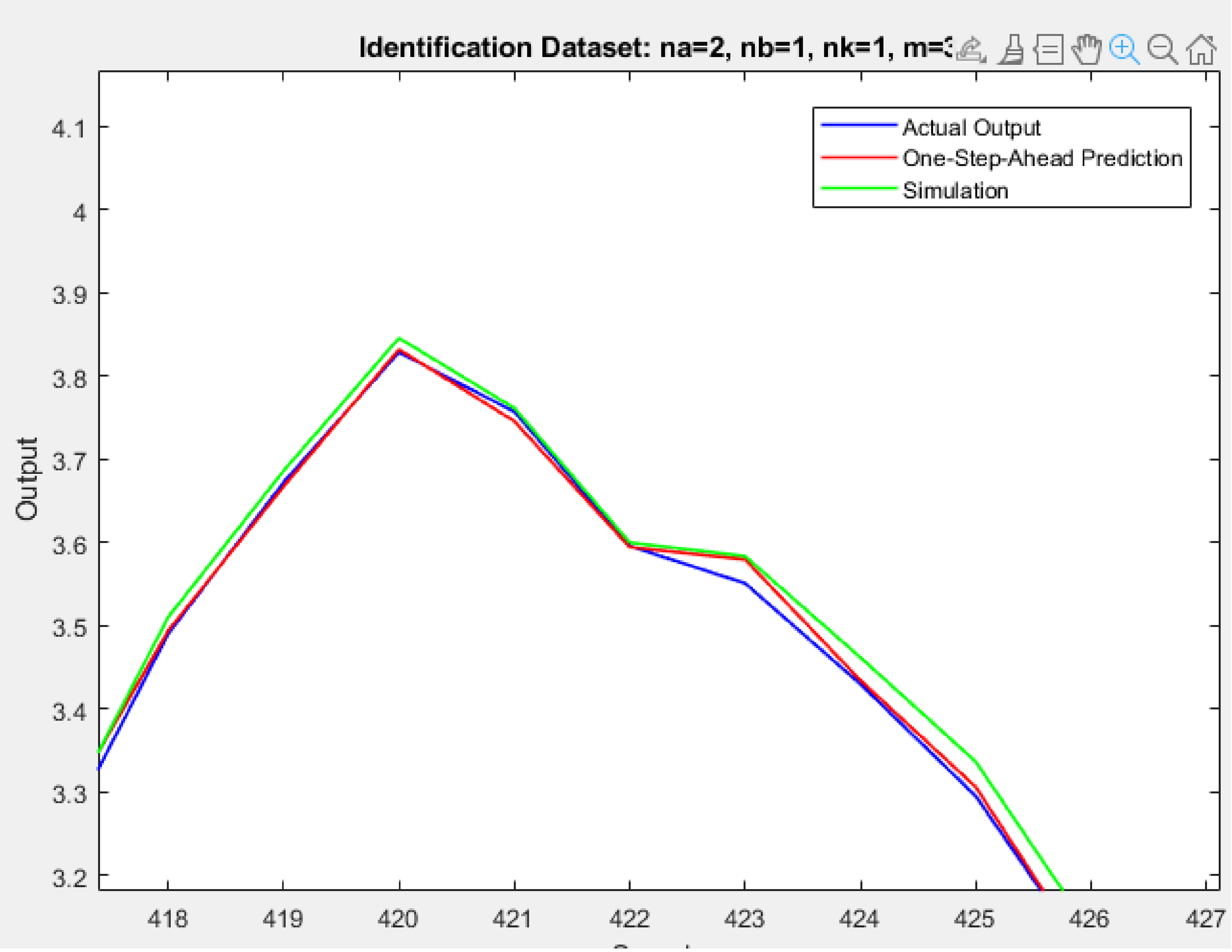
$$First - Degree\ Terms : \Phi_2 = y(k-1), y(k-2), u(k-1)$$

$$Second - Degree\ Terms : \Phi_3 = y(k-1)^2, y(k-2)^2, u(k-1)^2, y(k-1)y(k-2), \\ y(k-1)u(k-1), y(k-2)u(k-1)$$

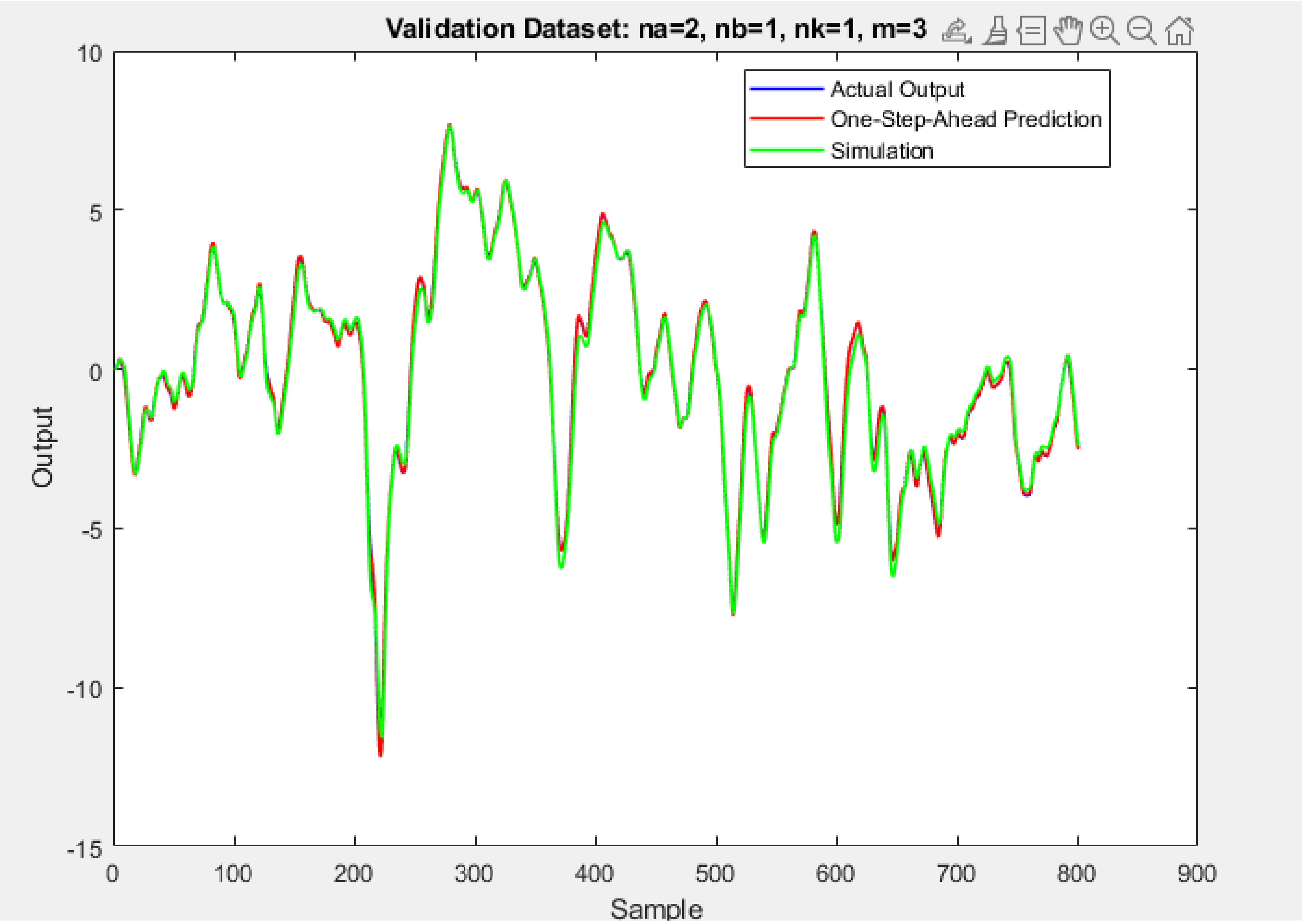
Tuning Results - MSE

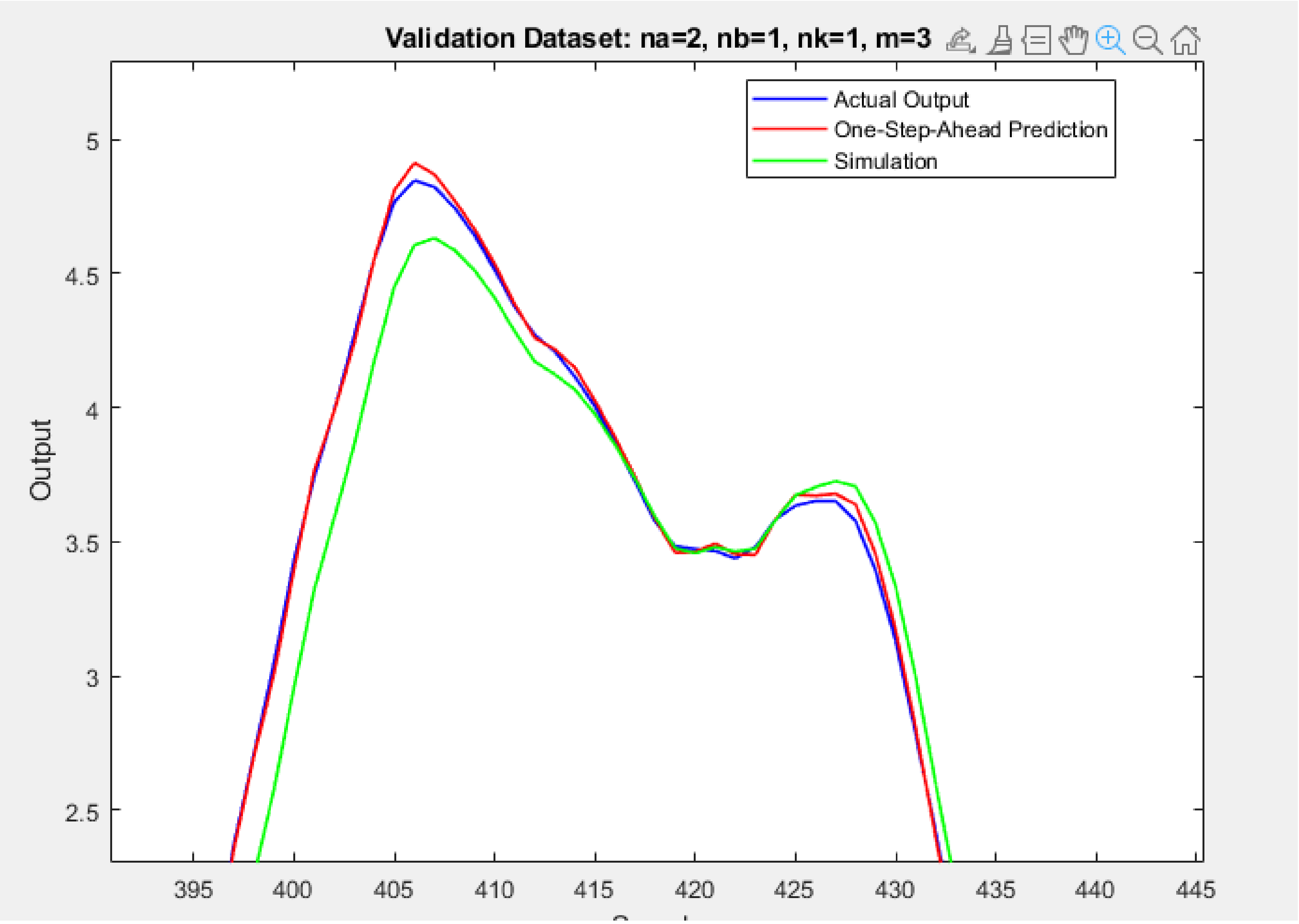
	na	nb	m	MSE y_{pred} (ID)	MSE y_{sim} (ID)	MSE y_{pred} (VAL)	MSE y_{sim} (VAL)
1	1.0000	1.0000	1.0000	0.0076	0.1633	0.0172	0.2763
2	1.0000	1.0000	2.0000	0.0033	0.0396	0.0104	0.1823
3	1.0000	1.0000	3.0000	0.0027	0.0290	0.0100	0.1847
4	1.0000	2.0000	1.0000	0.0076	0.1626	0.0171	0.2770
5	1.0000	2.0000	2.0000	0.0033	0.0404	0.0105	0.1901
6	1.0000	2.0000	3.0000	0.0025	0.0287	0.0104	0.1949
7	1.0000	3.0000	1.0000	0.0078	0.1632	0.0171	0.2775
8	1.0000	3.0000	2.0000	0.0034	0.0416	0.0105	0.1953
9	1.0000	3.0000	3.0000	0.0027	0.0297	0.0105	0.2018
10	1.0000	4.0000	1.0000	0.0087	0.1688	0.0171	0.2775
11	1.0000	4.0000	2.0000	0.0043	0.0466	0.0104	0.2001
12	1.0000	4.0000	3.0000	0.0035	0.0344	0.0106	0.2062
13	2.0000	1.0000	1.0000	0.0065	0.1826	0.0133	0.2799
14	2.0000	1.0000	2.0000	0.0019	0.0179	0.0052	0.0833
15	2.0000	1.0000	3.0000	0.0014	0.0147	0.0050	0.0825
16	2.0000	2.0000	1.0000	0.0031	0.1775	0.0041	0.2783
17	2.0000	2.0000	2.0000	0.0017	NaN	9.7510	NaN
18	2.0000	2.0000	3.0000	0.0013	NaN	25.3965	NaN
19	2.0000	3.0000	1.0000	0.0032	0.1900	0.0041	0.2779

Tuning Results - MSE Identification



Tuning Results - MSE Validation





Overall Conclusion

- The chosen nonlinear ARX model achieves low MSE, balancing accuracy and simplicity.
- The polynomial term generator ensures efficient and flexible model construction.
- Validation results confirm strong generalization and reliable performance on unseen data.

```
load('iddata-07.mat');
u = id.u;
y = id.y;
Ts = id.Ts;
u_val = val.u;
y_val = val.y;
N_val = length(y_val);
na_max = 4;
nb_max = 4;
m_max  = 3;
nk = 1;
N = length(y);
results = [];
for na = 1:na_max
    for nb = 1:nb_max
        for m = 1:m_max
            Phi = [];
            for k = max(na, nb + nk):N
                delayed_y = zeros(1, na);
                for j = 1:na
                    if (k - j) > 0
                        delayed_y(j) = y(k - j);
                    end
                end
                delayed_u = zeros(1, nb);
                for j = 1:nb
                    if (k - nk - j + 1) > 0
                        delayed_u(j) = u(k - nk - j + 1);
                    end
                end
                delayed_vars = [delayed_y, delayed_u];
                poly_terms = [1, monomial_terms(delayed_vars, m)];
                Phi = [Phi; poly_terms];
            end
            y_regress = y(max(na, nb + nk):end);
            theta = Phi \ y_regress;
            y_pred = zeros(N,1);
            for k = max(na, nb + nk):N
                delayed_y = zeros(1, na);
                for j = 1:na
                    if (k - j) > 0
                        delayed_y(j) = y(k - j);
                    end
                end
                delayed_u = zeros(1, nb);
                for j = 1:nb
                    if (k - nk - j + 1) > 0
                        delayed_u(j) = u(k - nk - j + 1);
                    end
                end
```

```
            delayed_vars = [delayed_y, delayed_u];
            poly_terms = [1, monomial_terms(delayed_vars, m)];
            y_pred(k) = poly_terms * theta;
        end
        y_sim = zeros(N,1);
        for k = max(na, nb + nk):N
            delayed_y = zeros(1, na);
            for j = 1:na
                if (k - j) > 0
                    delayed_y(j) = y_sim(k - j);
                end
            end
            delayed_u = zeros(1, nb);
            for j = 1:nb
                if (k - nk - j + 1) > 0
                    delayed_u(j) = u(k - nk - j + 1);
                end
            end
            delayed_vars = [delayed_y, delayed_u];
            poly_terms = [1, monomial_terms(delayed_vars, m)];
            y_sim(k) = poly_terms * theta;
        end
        mse_pred = mean((y - y_pred).^2);
        mse_sim = mean((y - y_sim).^2);
        y_pred_val = zeros(N_val,1);
        for k = max(na, nb + nk):N_val
            delayed_y = zeros(1, na);
            for j = 1:na
                if (k - j) > 0
                    delayed_y(j) = y_val(k - j);
                end
            end
            delayed_u = zeros(1, nb);
            for j = 1:nb
                if (k - nk - j + 1) > 0
                    delayed_u(j) = u_val(k - nk - j + 1);
                end
            end
            delayed_vars = [delayed_y, delayed_u];
            poly_terms = [1, monomial_terms(delayed_vars, m)];
            y_pred_val(k) = poly_terms * theta;
        end
        y_sim_val = zeros(N_val,1);
        for k = max(na, nb + nk):N_val
            delayed_y = zeros(1, na);
            for j = 1:na
                if (k - j) > 0
                    delayed_y(j) = y_sim_val(k - j);
                end
            end
```

```
                delayed_u = zeros(1, nb);
                for j = 1:nb
                    if (k - nk - j + 1) > 0
                        delayed_u(j) = u_val(k - nk - j + 1);
                    end
                end
                delayed_vars = [delayed_y, delayed_u];
                poly_terms = [1, monomial_terms(delayed_vars, m)];
                y_sim_val(k) = poly_terms * theta;
            end
            mse_pred_val = mean((y_val - y_pred_val).^2);
            mse_sim_val = mean((y_val - y_sim_val).^2);
            results = [results; na, nb, m, mse_pred, mse_sim, mse_pred_val,
mse_sim_val];
        end
    end
end
fig = uifigure('Name', 'MSE Results', 'Position', [100, 100, 1000, 400]);
uitable(fig, 'Data', results, ...
    'ColumnName', {'na','nb','m','MSE y_{pred} (ID)','MSE y_{sim}  (ID)','MSE
y_{pred} (VAL)','MSE y_{sim}  (VAL)'}, ...
    'Position', [25, 50, 950, 300], 'FontSize', 12);
na = 2; nb = 1; m = 3;
Phi = [];
for k = max(na, nb + nk):N
    delayed_y = zeros(1, na);
    for j = 1:na
        if (k - j) > 0
            delayed_y(j) = y(k - j);
        end
    end
    delayed_u = zeros(1, nb);
    for j = 1:nb
        if (k - nk - j + 1) > 0
            delayed_u(j) = u(k - nk - j + 1);
        end
    end
    delayed_vars = [delayed_y, delayed_u];
    poly_terms = [1, monomial_terms(delayed_vars, m)];
    Phi = [Phi; poly_terms];
end
y_regress = y(max(na, nb + nk):end);
theta = Phi \ y_regress;
y_pred = zeros(N,1);
for k = max(na, nb + nk):N
    delayed_y = zeros(1, na);
    for j = 1:na
        if (k - j) > 0
            delayed_y(j) = y(k - j);
        end
```

```
    end
    delayed_u = zeros(1, nb);
    for j = 1:nb
        if (k - nk - j + 1) > 0
            delayed_u(j) = u(k - nk - j + 1);
        end
    end
    delayed_vars = [delayed_y, delayed_u];
    poly_terms = [1, monomial_terms(delayed_vars, m)];
    y_pred(k) = poly_terms * theta;
end
y_sim = zeros(N,1);
for k = max(na, nb + nk):N
    delayed_y = zeros(1, na);
    for j = 1:na
        if (k - j) > 0
            delayed_y(j) = y_sim(k - j);
        end
    end
    delayed_u = zeros(1, nb);
    for j = 1:nb
        if (k - nk - j + 1) > 0
            delayed_u(j) = u(k - nk - j + 1);
        end
    end
    delayed_vars = [delayed_y, delayed_u];
    poly_terms = [1, monomial_terms(delayed_vars, m)];
    y_sim(k) = poly_terms * theta;
end
u_val = val.u;
y_val = val.y;
N_val = length(y_val);
y_pred_val = zeros(N_val,1);
for k = max(na, nb + nk):N_val
    delayed_y = zeros(1, na);
    for j = 1:na
        if (k - j) > 0
            delayed_y(j) = y_val(k - j);
        end
    end
    delayed_u = zeros(1, nb);
    for j = 1:nb
        if (k - nk - j + 1) > 0
            delayed_u(j) = u_val(k - nk - j + 1);
        end
    end
    delayed_vars = [delayed_y, delayed_u];
    poly_terms = [1, monomial_terms(delayed_vars, m)];
    y_pred_val(k) = poly_terms * theta;
end
```

```
y_sim_val = zeros(N_val,1);
for k = max(na, nb + nk):N_val
    delayed_y = zeros(1, na);
    for j = 1:na
        if (k - j) > 0
            delayed_y(j) = y_sim_val(k - j);
        end
    end
    delayed_u = zeros(1, nb);
    for j = 1:nb
        if (k - nk - j + 1) > 0
            delayed_u(j) = u_val(k - nk - j + 1);
        end
    end
    delayed_vars = [delayed_y, delayed_u];
    poly_terms = [1, monomial_terms(delayed_vars, m)];
    y_sim_val(k) = poly_terms * theta;
end
figure('Name','Identification Results','NumberTitle','off');
plot(y,'b','LineWidth',1.2); hold on; plot(y_pred,'r','LineWidth',1.2);
plot(y_sim,'g','LineWidth',1.2);
legend('Actual Output','One-Step-Ahead Prediction','Simulation','Location','Best');
title(sprintf('Identification Dataset: na=%d, nb=%d, nk=%d, m=%d',na,nb,nk,m));
xlabel('Sample'); ylabel('Output');
figure('Name','Validation Results','NumberTitle','off');
plot(y_val,'b','LineWidth',1.2); hold on; plot(y_pred_val,'r','LineWidth',1.2);
plot(y_sim_val,'g','LineWidth',1.2);
legend('Actual Output','One-Step-Ahead Prediction','Simulation','Location','Best');
title(sprintf('Validation Dataset: na=%d, nb=%d, nk=%d, m=%d',na,nb,nk,m));
xlabel('Sample'); ylabel('Output');

function terms = monomial_terms(vars, degree)
num_vars = length(vars);
terms = [];
for d = 1:degree
    for i = 1:num_vars
        terms = [terms, vars(i)^d];
    end
end
end
```