```
clc
clear
close all
set(0, 'DefaultLineLineWidth', 2);
set(0, 'defaultAxesFontSize', 12)
set(0, 'defaultAxesTickLabelInterpreter','latex');
set(0, 'defaultlegendInterpreter','latex')
```

The system

We will consider a LTI system of the form:

$$\begin{cases} \underset{n\,\mathrm{x}\,1}{x_{t+1}} = \underset{n\,\mathrm{x}\,n}{A} \cdot \underset{n\,\mathrm{x}\,1}{x_t} + \underset{n\,\mathrm{x}\,m}{B} \cdot \underset{m\,\mathrm{x}\,1}{u_t} + \underset{n\,\mathrm{x}\,l}{K} \cdot \underset{l\,\mathrm{x}\,1}{e_t} & (1) \\ \underset{l\,\mathrm{x}\,1}{y_t} = \underset{l\,\mathrm{x}\,n}{C} \cdot \underset{n\,\mathrm{x}\,1}{x_t} + \underset{l\,\mathrm{x}\,m}{D} \cdot \underset{m\,\mathrm{x}\,1}{u_t} + \underset{l\,\mathrm{x}\,1}{e_t} & (2) \end{cases}$$

where

- the inputs $u_t \in \mathbb{R}^m \;\; \Leftrightarrow \;\; m =$ number of inputs,
- the outputs $y_t \in \mathbb{R}^l \;\; \Leftrightarrow \;\; l =$ number of outputs,
- the states $x_t \in \mathbb{R}^n \;\; \Leftrightarrow n =$ number of states.
- the noise sequence e_t is supposed to be $\sim WN(0;\Sigma_e)$ with $\Sigma_e = \mathbb{E}[e_k e_k^T]$ and $\Sigma_{ij} = \mathbb{E}[e_i e_k^T] = 0 \text{ for } i \neq k$
- $N_s =$ number of samples

```
tol = 1e-10;                                % used to check null value
m = 3;
l = 2;
n = 2;
M = 4;
N = 5;
j = 100 *max(M,N);
Ns = M+N+j;
Ts = 1;                                     % sampling time
t = (0:Ns-1) * Ts;                          % time of sampling
A = [0.7  0; 0  0.3];                       % states matrix [n x n]
B = [-0.1  0.2  0.6; 0.9 -0.5 -0.4];        % inputs matrix [n x m]
C = [0.5  0.2; 0.6 -0.8];                   % outputs matrix [l x n]
D = [1 0 0; 0 0 0];                         % [l x m]
K = [1 0; 0 1];                             % [n x l]

rng(42);
sigma = 0.1;
e = sigma * randn(Ns, l);                   % e ~WN(0;σ^2 I_l)
x = zeros (Ns, n);
u = zeros(Ns,m);

% ==== Genera r(t): PRBS levels +-2, Ts= 0.05, full band ====
```

```
% r2 = idinput(N,Type,Band,Range) funzione del System Identification Toolbox
Ts = 0.05;                              % in [sec]
fs = 1/Ts;                              % 20 [Hz]
Range = [-2, 2];                        % levels of the PRBS
Band =  [ 0  1];                        % banda normalizzata (0-1 = full band fino a
Nyquist)

for c = 1:m
    u(:,c) = idinput(Ns,'prbs',Band,Range);
end
```

```
Warning: The PRBS signal delivered is the 509 first values of a full sequence of length 511.
Warning: The PRBS signal delivered is the 509 first values of a full sequence of length 511.
Warning: The PRBS signal delivered is the 509 first values of a full sequence of length 511.
```

```
y_clean = zeros(Ns, l, m);
sys = cell(m,1);
for c = 1:m
    U = u(:,c);                                     % input Matrix
    sys{c} = ss(A, B(:,c), C, D(:,c), 1);           % system state space model
    y_clean(:,:,c) = lsim(sys{c}, U, t);            % deterministic output for each
input
end
y_noisy = y_clean + e;                              % stochastic output for each input
yd = sum(y_clean, 3);                               % deterministic output as sum of
all inputs
ys = sum(y_noisy, 3);                               % stochastic output as sum of all
inputs
```

Matrix input-output equations

$$\underset{lN \times j}{Y_f} = \underset{lN \times n}{\Gamma_N} \cdot \underset{n \times j}{X_f} + \underset{lN \times mN}{H_N} \cdot \underset{mN \times j}{U_f} + \underset{lN \times lN}{H_N^S} \cdot \underset{lN \times j}{E_f} \qquad (3)$$

$$\underset{lM \times j}{Y_p} = \underset{lM \times n}{\Gamma_M} \cdot \underset{n \times j}{X_p} + \underset{lM \times mM}{H_M} \cdot \underset{mM \times j}{U_p} + \underset{lM \times lM}{H_M^S} \cdot \underset{lM \times j}{E_p} \qquad (4)$$

$$\underset{mM \times j}{U_p} = \begin{pmatrix} u_1 & u_2 & \dots & u_j \\ u_2 & u_3 & \dots & u_{j+1} \\ \dots & \dots & \dots & \dots \\ u_M & u_{M+1} & \dots & u_{j+M-1} \end{pmatrix} \qquad (5)$$

$$\underset{mN \times j}{U_f} = \begin{pmatrix} u_{M+1} & u_{M+2} & \dots & u_{M+j} \\ u_{M+2} & u_{M+3} & \dots & u_{M+j+1} \\ \dots & \dots & \dots & \dots \\ u_{M+N} & u_{M+N+1} & \dots & u_{M+N+j} \end{pmatrix} \qquad (6)$$

In a similar way, we define the block Hankel matrices $\underset{lM \times j}{Y_p}$, $\underset{lN \times j}{Y_f}$, $\underset{lM \times j}{E_p}$, $\underset{lN \times j}{E_f}$,

j must be much larger (tipically 100 times) than N, M: $j \geq 100 \cdot \max(N, M)$ but $N_s = M + N + j$

$$\underset{(l+m)M \text{ x } j}{W_p} \triangleq \begin{pmatrix} Y_p \\ U_p \end{pmatrix}$$

The past and the future state sequences are definited as

$$\underset{n \text{ x } j}{X_p} = \begin{pmatrix} x_1 & x_2 & \dots & x_j \end{pmatrix}$$

$$\underset{n \text{ x } j}{X_f} = \begin{pmatrix} x_{M+1} & x_{M+2} & \dots & x_{M+j} \end{pmatrix}$$

$$\underset{lq \text{ x } n}{\Gamma_q} = \begin{pmatrix} C \\ CA \\ \dots \\ CA^{q-1} \end{pmatrix}, \quad \underset{lq \text{ x } mq}{H_q} = \begin{pmatrix} D & 0 & \dots & 0 \\ CB & D & \dots & 0 \\ \dots & \dots & \dots & \dots \\ CA^{q-2}B & CA^{q-3}B & \dots & D \end{pmatrix}, \quad \underset{lq \text{ x } lq}{H_q^S} = \begin{pmatrix} I_l & 0 & \dots & 0 \\ CK & I_l & \dots & 0 \\ \dots & \dots & \dots & \dots \\ CA^{q-2}K & CA^{q-3}K & \dots & I_l \end{pmatrix}$$

where $q \in \mathbb{N}_0$, Γ_q is the extended observability matrix, H_q and H_q^S are the Toeplitz matrices containing the impulse response of the system to the deterministic input u_k and to the stochastic input e_k respectively.

```
Up = zeros(m*M, j);
Uf = zeros(m*N, j);
Yp = zeros(l*M, j);
Yf = zeros(l*N, j);
for i = 1:j
    idp = i : i + M - 1;                        % index for past Hankel Matrices
    idf = M + i : M + i + N - 1;                % index for future Hankel Matrices
    Up(:,i) = reshape(u(idp,:)', m*M, 1);    % [mM x j] = [12 x 500]
    Yp(:,i) = reshape(ys(idp,:)', l*M, 1);  % [mN x j] = [15 x 500]
    Uf(:,i) = reshape(u(idf,:)', m*N, 1);    % [lM x j] = [ 8 x 500]
    Yf(:,i) = reshape(ys(idf,:)', l*N, 1);  % [lN x j] = [10 x 500]
end
```

First step of subspace identification problem

Using the QR decomposition:

$$\begin{pmatrix} \underset{(l+m)M \text{ x } j}{W_p} \\ \underset{mN \text{ x } j}{U_f} \\ \underset{lN \text{ x } j}{Y_f} \end{pmatrix} = \begin{pmatrix} \underset{(l+m)M \text{ x } (l+m)M}{R_{11}} & 0 & 0 \\ \underset{mN \text{ x } (l+m)M}{R_{21}} & \underset{mN \text{ x } mN}{R_{22}} & 0 \\ \underset{lN \text{ x } (l+m)M}{R_{31}} & \underset{lN \text{ x } mN}{R_{32}} & \underset{lN \text{ x } lN}{R_{33}} \end{pmatrix} \cdot \begin{pmatrix} Q_1^T \\ Q_2^T \\ Q_3^T \end{pmatrix} \qquad (9)$$

by posing:

$$\underset{lN \text{ x } ((l+m)M+mN)}{L} = (R_{31} \ \ R_{32}) \cdot \begin{pmatrix} R_{11} & 0 \\ R_{21} & R_{22} \end{pmatrix}^{\dagger}$$

We know that:

$$\underset{lN \text{ x } (l+m)M}{L_w} = L(:, 1 : M(m+l))$$

$$\underset{lN \text{ x } mN}{L_u} = L(:, M(m+l)+1 : end)$$

It appears that there is a minor inconsistency in the original segmentation of the matrix L. After careful examination, we found that adjusting the dimensions to correspond to the past window length M ensures consistency with the theoretical derivation and numerical implementation.

```
Wp = [Yp; Up];                              % [(l+m)M x j] = [20 x 500]
Z  = [Wp; Uf; Yf];                          % [(l+m)(M+N) x j] = [45 x 500]

[Q_tmp, R_tmp] = qr(Z');                    % Z' = Q_tmp * R_tmp
R = R_tmp';                                 % R inferior triangular matrix
Q = Q_tmp';                                 % Q ortogonal matrix

if norm(triu(R,1), 'fro') < tol
    fprintf('R is triangular: OK \n');
else
    fprintf ('R is not triangular: NOK\n');
end
```

```
R is triangular: OK
```

```
if norm(Q'*Q - eye(size(Q)), 'fro') < tol
    fprintf('Q is ortogonal: OK \n');
else
    fprintf('Q is not ortogonal: NOK\n');
end
```

```
Q is ortogonal: OK
```

```
nw = (l+m)*M;                               % Wp row number = 20
nu = m*N;                                   % Uf row number = 15
ny = l*N;                                   % Yf row number = 10
R11 = R(1:nw, 1:nw);                        % [20 x 20]
R12 = zeros(nw, nu);                        % [20 x 15]
R21 = R(nw+1:nw+nu, 1:nw);                  % [15 x 20]
R22 = R(nw+1:nw+nu, nw+1:nw+nu);            % [15 x 15]
R31 = R(nw+nu+1:end, 1:nw);                 % [10 x 20]
R32 = R(nw+nu+1:end, nw+1:nw+nu);           % [10 x 15]
R33 = R(nw+nu+1:end, nw+nu+1:nw+nu+ny);     % [10 x 10]
```

```
L = [R31 R32] * pinv([R11 R12; R21 R22]);   % [10 x 35]
Lw = L(:,1:M*(m+l));                        % [10 x 20] this is different to how it
is proposed in the paper
Lu = L(:,M*(m+l)+1:end);                    % [10 x 15]
```

Second step of subspace identification problem

In this step we'll calculate the SVD of L_w.

$$\underset{lN \times M(m+l)}{L_w} = (U_1 \ \ U_2) \cdot \begin{pmatrix} S_1 & 0 \\ 0 & S_2 \end{pmatrix} \cdot \begin{pmatrix} V_1^T \\ V_2^T \end{pmatrix} \approx U_1 \ S_1 \ V_1^T \qquad (10)$$

where the rank $\hat{n}$ is determinated by inspecting the number of dominant singular value of S_1, This is an approximation of the order n of the system.

The system order was automatically selected based on a cumulative energy criterion applied to the singular values of the Hankel matrix. In accordance with common practice in subspace system identification, the estimated order $\hat{n}$ was chosen as the smallest integer such that the cumulative sum of the squared singular values accounts for at least 95% of the total energy. This approach is widely adopted in the literature, as it provides a robust and data-driven separation between the dominant system dynamics and noise-induced components.

$$\hat{n} = \min\{k : \frac{\sum_{i=1}^{k} \sigma_i^2}{\sum_{i=1}^{M(m+l)} \sigma_i^2} \geq \eta\} \quad \eta \in [0.95, 0.99]$$

Important is that, under the assumption that the number of columns in the data block Hankel matrices Y_f, U_f, W_p is infinite $(j = \infty)$ there exists a direct link between L_W and the observability matrix Γ_N and the state sequence X_f.

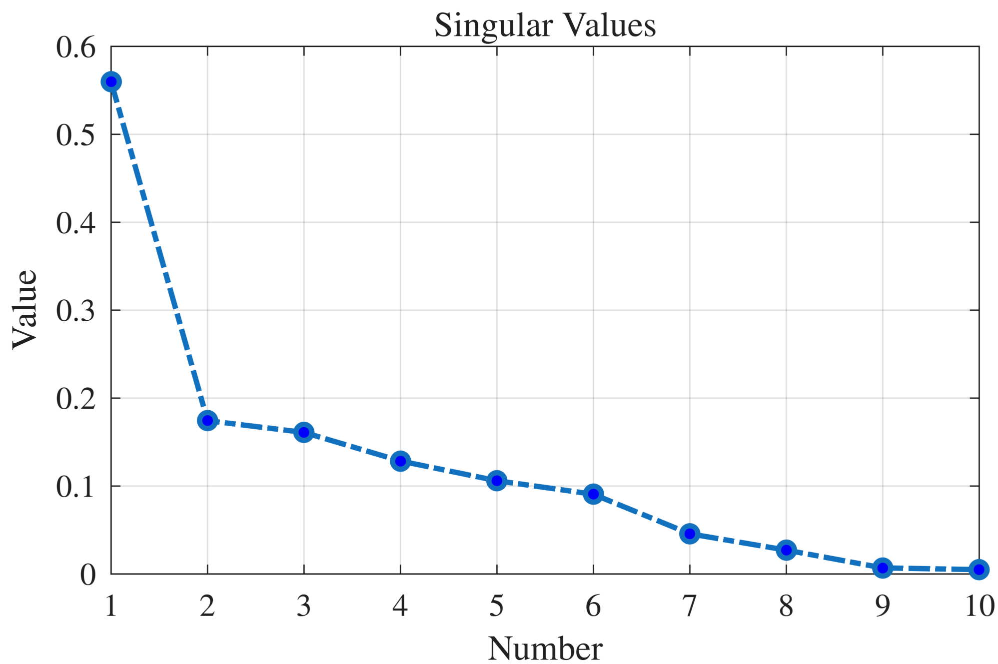
$$\underset{lN \times \hat{n}}{\Gamma_N} = \underset{lN \times \hat{n}}{U_1} \cdot \underset{\hat{n} \times \hat{n}}{S_1^{1/2}}$$

$$\underset{\hat{n} \times j}{\hat{X}_f} = \underset{\hat{n} \times \hat{n}}{S_1^{1/2}} \cdot \underset{\hat{n} \times M(m+l)}{V_1^T} \cdot \underset{M(m+l) \times j}{W_p}$$

It (Van Overschee and De Moor 1996) is proven that $\hat{X}_f$ is a Kalman filter estimate of the state sequence X_f

```
singvals = svd(Lw);
figure;
nsv = length(singvals);
plot(1:nsv, singvals, '-.o', 'MarkerFaceColor','b', 'MarkerSize',6);
grid on;
xlabel('Number','Interpreter','latex');
ylabel('Value','Interpreter','latex');
```

```
title('Singular Values','Interpreter','latex');
xlim([1 nsv]);
xticks(1:1:nsv);
```




```
eta = 0.95;                              % thereshold (95%)
energy = cumsum(singvals.^2) / sum(singvals.^2);
hat_n = find(energy >= eta, 1, 'first')
```

```
hat_n =
5
```

```
[U_,S_,V_] = svd(Lw);
U1 = U_(:, 1:hat_n);                     % [10 x 3]
S1 = S_(1:hat_n, 1:hat_n);               % [3 x 3]
V1 = V_(:, 1:hat_n);                     % [20 x 3]
Gamma_N = U1 * sqrt(S1);                 % [10 x 3]
hat_Xf = sqrt(S1) * V1' * Wp;            % [3 x 500]
```

Checking results consistency

Solve the following least square problem for the unknown parameters L_w and L_u

$$\min_{L_w,\ L_u} \left\| Y_f - (L_w \ \ L_u) \cdot \begin{pmatrix} W_p \\ U_f \end{pmatrix} \right\|_F^2 = \min_{L_w,\ L_u} ||Y_f - L_w \cdot Wp - L_u \cdot U_f||_F^2$$

This is a linear least squares having as closed solution:

$$\underset{lN \times (M(l+m)+mN)}{L} = [\underset{lN \times M(m+l)}{L_w} \quad \underset{lN \times mN}{L_u}] = \underset{lN \times j}{Y_f} \cdot \underset{j \times (M(l+m)+mN)}{\begin{bmatrix} W_p \\ U_f \end{bmatrix}^{\dagger}}$$

```
L_ls = Yf * pinv([Wp; Uf]);
Lw_ls = L_ls(:,1:M*(m+l));                              % [10 x 20]
Lu_ls = L_ls(:,M*(m+l)+1:end);                          % [10 x 15]
err_w = norm(Lw - Lw_ls, 'fro') / norm(Lw, 'fro');
err_u = norm(Lu - Lu_ls, 'fro') / norm(Lu, 'fro');

if err_w < tol
    fprintf('QR+SVD and closed-form LS yield the same Lw matrix up to numerical 
accuracy: OK\n');
else
    fprintf('The Lw matrix obtained via QR+SVD does not coincide with the closed-
form LS solution within numerical precision: NOK\n');
end
```

```
QR+SVD and closed-form LS yield the same Lw matrix up to numerical accuracy: OK
```

```

if err_u < tol
    fprintf('QR+SVD and closed-form LS yield the same Lu matrix up to numerical 
accuracy: OK\n');
else
    fprintf('The Lu matrix obtained via QR+SVD does not coincide with the closed-
form LS solution within numerical precision: NOK\n');
end
```

```
QR+SVD and closed-form LS yield the same Lu matrix up to numerical accuracy: OK
```

The SPC prediction coincides exactly with the Least Squares solution, confirming that both approaches yield the same predicted output.

Evaluate in simulation the tracking performance

The output predicted is calculated as:

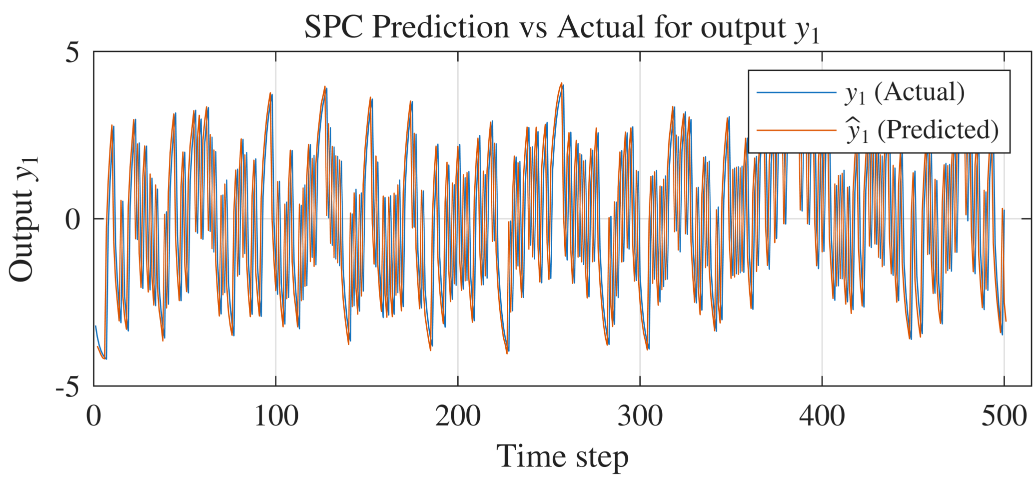
$$\underset{lN \times j}{\widehat{Y}_f} = \underset{lN \times (l+m)M}{L_w} \cdot \underset{(l+m)M \times j}{W_p} + \underset{lN \times mN}{L_u} \cdot \underset{mN \times j}{U_f}$$

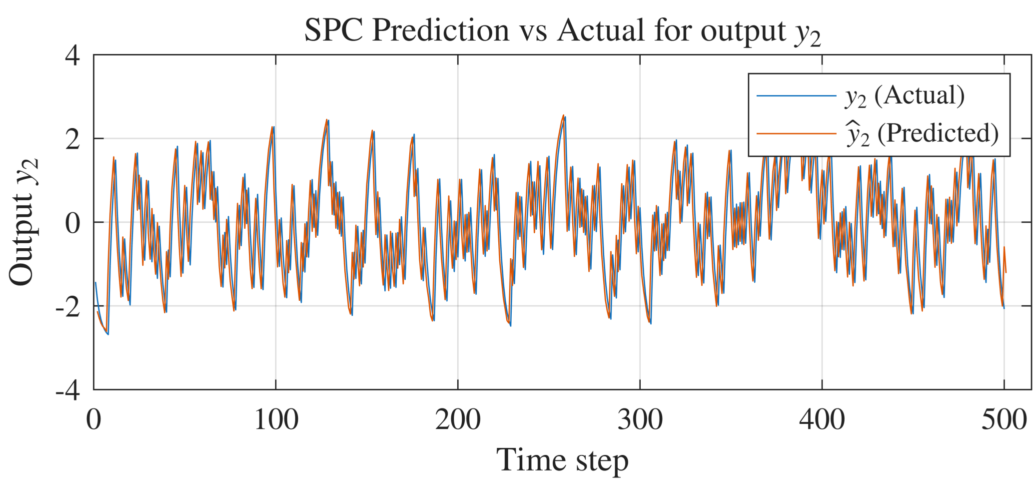
Only the first block of the predicted horizon is plotted because it represents the immediate next-step prediction. The real output is shifted by one step to align with this one-step-ahead prediction.

```
hat_Yf = Lw * Wp + Lu * Uf;
hat_y= hat_Yf(1:l, :);  % l x Ns

figure('Position', [100, 100, 1200, 1000]);
for i = 1:2
    subplot(2,1,i);
    % the prediction is 1 step delayed so, shift the y
    plot(1:j, yd(3:j+2,i),'LineWidth', 0.5,'DisplayName','$y_{' + string(i) + '}$
(Actual)');
    hold on;
```

```
    plot(2:j+1, hat_y(i,1:j).','LineWidth', .5,'DisplayName',['$\hat y_{' 
num2str(i) '}$ (Predicted)']);
    grid on;
    xlabel('Time step','Interpreter','latex');
    ylabel(['Output $y_' num2str(i) '$'],'Interpreter','latex');
    title(['SPC Prediction vs Actual for output $y_' num2str(i) 
'$'],'Interpreter','latex');
    legend('Interpreter','latex');
    xlim([0 515])
end
```





Subspace Identification based Implementation of MPC

The aim of MPC is to construct a controller that minimizes a performance criterion J, defined as:

$$J = (\hat{y}_f^* - r_f^*)^T \ Q \ (\hat{y}_f^* - r_f^*) + (u_f^*)^T R u_f^*$$

where

- ν is the simulation duration

- $$\underset{m\nu \text{ x } 1}{u_f^*} = \begin{pmatrix} u_1^* \\ u_2^* \\ \dots \\ u_\nu^* \end{pmatrix} \in \mathbb{R}^{Mm}$$ is constructed by vertically stacking the future input vectors at each future time step for all input components.

- $$\underset{l\nu \text{ x } 1}{\hat{y}_f^*} = \begin{pmatrix} \hat{y}_1^* \\ \hat{y}_2^* \\ \dots \\ \hat{y}_\nu^* \end{pmatrix} \in \mathbb{R}^{Ml}$$ is constructed by vertically stacking the predicted output vectors at each future time step for all output components.

- $$\underset{l\nu \text{ x } 1}{r_f^*} = \begin{pmatrix} r_1^* \\ r_2^* \\ \dots \\ r_\nu^* \end{pmatrix} \in \mathbb{R}^{Ml}$$ is constructed by vertically stacking the reference output vectors at each future time step for all output components.

- $$\underset{Nl \text{ x } Nl}{Q^*} = \begin{bmatrix} Q_1 & 0 & .. & 0 \\ 0 & Q_2 & .. & 0 \\ .. & .. & .. & .. \\ 0 & 0 & .. & Q_N \end{bmatrix}$$ and $$\underset{Nm \text{ x } Nm}{R^*} = \begin{bmatrix} R_1 & 0 & .. & 0 \\ 0 & R_2 & .. & 0 \\ .. & .. & .. & .. \\ 0 & 0 & .. & R_N \end{bmatrix}$$ are user-defined weight matrices.

This section of the algorithm must be done on a running system, unlike the previous one that could be done "offline".

```
Q_star = kron(eye(N),diag([1, 2]));                         %[10x10]
R_star = kron(eye(N),diag([.01, 0.5,.02]));                 %[15x15]

% Simulation setup
nu = 200;                           % simulation duration
u_star = zeros(nu, m);              % buffer to save inputs
y_star = zeros(nu, l);              % buffer to save measured outputs
x_star = zeros(n,1);                % system states
r_star = zeros(nu + N, l);          % reference output
r_star(20:end, 1) = 1.0;            % we set reference values arbitrarily
r_star(50:end, 2) = -1.0;
```

Recursive algorithm control steps

The algorithm uses a backwards horizon of M data points to estimate the forward horizon of N data points, repeating iteratively at each time step.

From subspace identification (but in this case not using block Hankel matrices), we know that :

$$\underset{lN \times 1}{\hat{y}_f^*} = \underset{lN \times (l+m)M}{L_w} \cdot \underset{(l+m)M \times 1}{w_p^*} + \underset{lN \times mN}{L_u} \cdot \underset{mN \times 1}{u_f^*}$$

with

$$\underset{(l+m)M \times 1}{w_p^*} \triangleq \begin{pmatrix} y_p^* \\ u_p^* \end{pmatrix}, \quad \underset{lM \times 1}{y_p^*} = \begin{pmatrix} y_{-M+1}^* \\ y_{-M+2}^* \\ \dots \\ y_0^* \end{pmatrix}, \quad \underset{mM \times 1}{u_p^*} = \begin{pmatrix} u_{-M+1}^* \\ u_{-M+2}^* \\ \dots \\ u_0^* \end{pmatrix}$$

being the last known values of the inputs and the outputs.

We can define the SPC control law as

$$\underset{mN \times 1}{u_f^*} = \left[\left(\underset{mN \times mN}{R^*} + \underset{mN \times lN}{L_u^T} \cdot \underset{lN \times lN}{Q^*} \cdot \underset{lN \times mN}{L_u} \right)^{-1} \underset{mN \times lN}{L_u^T} \cdot \underset{lN \times lN}{Q^*} \right] \cdot \left(\underset{lN \times 1}{r_f^*} - \underset{lN \times (l+m)M}{L_w} \cdot \underset{(l+m)M \times 1}{w_p^*} \right)$$

it can also be seen as

$$\underset{mN \times 1}{u_f^*} = \underset{mN \times lN}{\text{gain}} \cdot \underset{lN \times 1}{\epsilon}$$

And then calculate iteratively the next controlled input.

```
% to save computational time, let's compute the system gain outside of the
% simulation loop
gain=pinv(R_star+Lu'*Q_star*Lu)*(Lu'*Q_star); % we used pinv for stability

% SIMULATION START
for k = (M + 1) : nu

    % SPC step 4: build wp
    yp_star = reshape(y_star(k-M:k-1, :).', [], 1);   % [M*l × 1]
    up_star = reshape(u_star(k-M:k-1, :).', [], 1);   % [M*m × 1]
    wp_star = [yp_star; up_star];

    % build 1rf
    ref_block = r_star(k : k+N-1, :);
    rf_star = reshape(ref_block', [], 1);
    epsilon=rf_star-Lw*wp_star;

    % SPC step 5: compute u at time step k using control law
    u_future_seq = gain * epsilon;
    u_k = u_future_seq(1:m);

    % compute y and x at time step k, to be used in the next step
    x_next = A * x_star + B * u_k;
    y_k_measured = C * x_star + D * u_k;
    x_star = x_next;
```

```
    u_star(k, :) = u_k';
    y_star(k, :) = y_k_measured';

end
```

Plotting control results

```
% ------------------------------
% Plot results - scalable version
% ------------------------------

clf

% --- Outputs ---
subplot(2,1,1); hold on;

time_y = 1:nu;           % tutti gli step simulati
num_outputs = 1;         % numero di output
colors = lines(num_outputs);

for i = 1:num_outputs
    plot(time_y, y_star(:,i), 'LineWidth',1.5, 'Color', colors(i,:), ...
        'DisplayName', ['$y_' num2str(i) '$']);
end

% plot reference (same size as y_star)
r_plot = r_star(1:nu, :);
for i = 1:num_outputs
    plot(time_y, r_plot(:,i), '--', 'LineWidth',1.2, 'Color', colors(i,:), ...
        'DisplayName', ['$r_' num2str(i) '$']);
end

grid on
legend('Interpreter','latex','Location','best')
title('Controlled outputs','Interpreter','latex')
xlabel('Time step','Interpreter','latex')
ylabel('Output','Interpreter','latex')

% --- Inputs ---
subplot(2,1,2); hold on;

time_u = (M+1):nu;       % solo step dove u è calcolato
colors = lines(m);      % nuova mappa colori per gli input

for i = 1:m
    plot(time_u, u_star(time_u,i), 'LineWidth',1 * (4-i), 'Color', colors(i,:), ...
        'DisplayName', ['$u_' num2str(i) '$']);
end
```

```
grid on
legend('Interpreter','latex','Location','best')
title('Control inputs','Interpreter','latex')
xlabel('Time step','Interpreter','latex')
ylabel('Input','Interpreter','latex')
```