Secure State Estimation of Networked Systems Under Sparse Malicious Error Attacks

Stan Żak

School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47907
zak@purdue.edu

Acknowledgments

Results presented here were obtained in collaboration with

- Dr. Stefen Hui from San Diego State University

and with our former Ph.D. students:

- Dr. Badriah Alenezi from Kuwait University, and
- Dr. Mukai Zhang from Purdue University

Outline

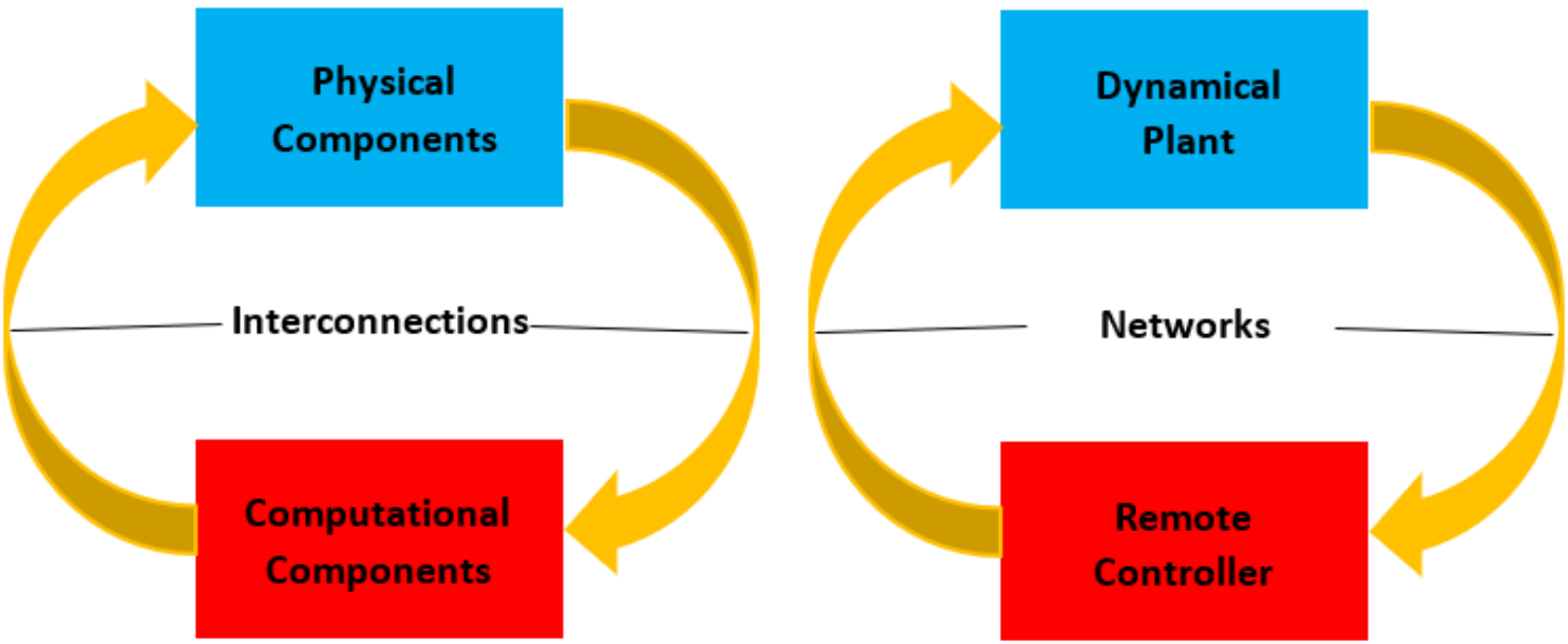
- Defining terms and background results
- Sparse vector recovery problem formulation
- Secure state estimation of networked control systems corrupted by unknown input and output spare errors
- Modeling sparse malicious packet drop attacks
- Observer Architecture
- Sparse malicious error attacks estimation

Defining terms and background results

Networked Control Systems

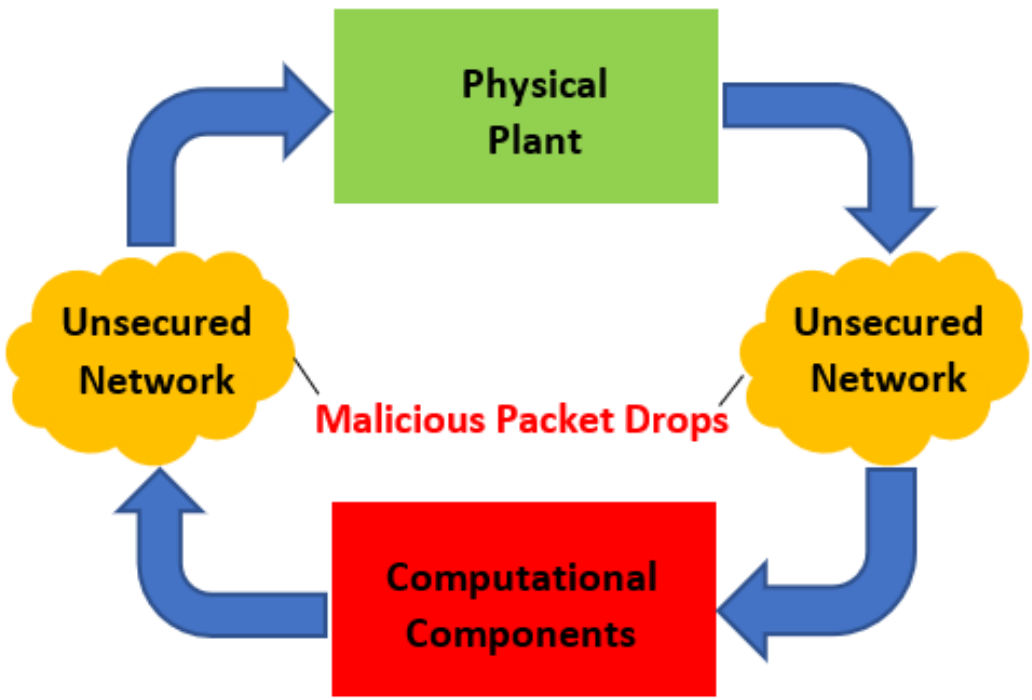
Definition

Networked Control Systems combine cyber and physical components, that is, they are combinations of the physical world with the virtual world of information processing



Challenges in a Networked Control System (NCS)

- NCS depends on wireless communication
- Major challenge in the NCS design—security
- For example, malicious packet drop attacks in the communication networks



Security Issues in the NCS Design

- Overcoming malicious packet drop attacks
- Secure state estimation of NCS when actuator and sensor measurements being corrupted by external malicious packet drop attacks
- Detecting and monitoring malicious attacks

Modeling sparse malicious packet drop attacks

- Estimating disturbances of the communication network such as noise, delays, and packet drops formulated as a sparse vector recovery problem
- Sparse $\boldsymbol{e}$—more zero entries than non-zero entries in the vector $\boldsymbol{e}$

Definition (Sparse vector recovery problem)

Estimate an unknown vector $\boldsymbol{x}$ in the linear system, $\boldsymbol{A}\boldsymbol{x} + \boldsymbol{e} = \boldsymbol{b}$, where the vector $\boldsymbol{b}$ and the matrix $\boldsymbol{A}$ are known and $\boldsymbol{e}$ models the unknown disturbances

Analysis of $\boldsymbol{Ax} + \boldsymbol{e} = \boldsymbol{b}$

Assumptions:

1. $\boldsymbol{b}$ and the full column rank matrix $\boldsymbol{A}$ are known;
2. Only a "small" number of entries of $\boldsymbol{b}$ corrupted by $\boldsymbol{e}$

Justifying the second assumption

Candes and Tao's observation: if the number of nonzero entries of the error vector is "large", then it is in general impossible to reconstruct $\boldsymbol{x}$ from $\boldsymbol{Ax} + \boldsymbol{e} = \boldsymbol{b}$ for a given $\boldsymbol{A}$ and $\boldsymbol{b}$

E. J. Candes and T. Tao, *Decoding by linear programming*, IEEE Transactions on Information Theory, Vol. 51, No. 12, pp. 4203–4215, 2005

Reconstructing $\boldsymbol{x}$ from $\boldsymbol{Ax} + \boldsymbol{e} = \boldsymbol{b}$ for a given $\boldsymbol{A}$ and $\boldsymbol{b}$

- Cannot have too many non-zero entries in $\boldsymbol{e}$ to reconstruct $\boldsymbol{x}$!
- Indeed, let $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and suppose $m = 2n$
- Consider two distinct fixed vectors $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$
- Suppose the vector $\boldsymbol{b} \in \mathbb{R}^m$ is constructed by setting n coefficients of $\boldsymbol{b}$ equal to those of $\boldsymbol{Ax}$ and n coefficients of $\boldsymbol{b}$ equal to those of $\boldsymbol{A}\hat{\boldsymbol{x}}$
- Then we have $\boldsymbol{b} = \boldsymbol{Ax} + \boldsymbol{e} = \boldsymbol{A}\hat{\boldsymbol{x}} + \hat{\boldsymbol{e}}$ for some $\boldsymbol{e}$ and $\hat{\boldsymbol{e}}$
- In sum, the maximum number of nonzero coefficients in $\boldsymbol{e}$ should be smaller than $n = m/2$ if we are to be able to reconstruct $\boldsymbol{x}$

Cannot have too many non-zero elements in $\boldsymbol{e}$ to recover $\boldsymbol{x}$ in $\boldsymbol{Ax} + \boldsymbol{e} = \boldsymbol{b}$

Example

$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \hat{\boldsymbol{x}} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Then two coefficients of $\boldsymbol{b}$ equal to those of $\boldsymbol{Ax}$ and two equal to those of $\boldsymbol{A}\hat{\boldsymbol{x}}$.

Example—Contd.

Solving the equations $\boldsymbol{e} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$ and $\hat{\boldsymbol{e}} = \boldsymbol{b} - \boldsymbol{A}\hat{\boldsymbol{x}}$, we obtain

$$\boldsymbol{e} = \begin{bmatrix} 0 \\ 0 \\ -1 \\ -1 \end{bmatrix} \quad \text{and} \quad \hat{\boldsymbol{e}} = \begin{bmatrix} -1 \\ -1 \\ 0 \\ 0 \end{bmatrix}.$$

Note that both $\boldsymbol{e}$ and $\hat{\boldsymbol{e}}$ have $m = n/2$ nonzero components

- We do not know if $\boldsymbol{e}$ or $\hat{\boldsymbol{e}}$ corrupts the system
- We cannot recover $\boldsymbol{x}$ in $\boldsymbol{A}\boldsymbol{x} + \boldsymbol{e} = \boldsymbol{b}$
- We have to have less than $m = n/2$ nonzero components in $\boldsymbol{e}$ to start talking about recovering $\boldsymbol{x}$ in $\boldsymbol{A}\boldsymbol{x} + \boldsymbol{e} = \boldsymbol{b}$

How to recover $\boldsymbol{x}$ from $\boldsymbol{Ax} + \boldsymbol{e} = \boldsymbol{b}$?

- Candes and Tao's† idea:
- We can recover $\boldsymbol{x}$ if we have $\boldsymbol{e}$
- Plan: Reconstruct $\boldsymbol{e}$ and then compute $\boldsymbol{x}$

† E. J. Candes and T. Tao, *Decoding by linear programming*, IEEE Transactions on Information Theory, Vol. 51, No. 12, pp. 4203–4215, 2005

Reconstructing $\boldsymbol{e}$ from $\boldsymbol{Ax} + \boldsymbol{e} = \boldsymbol{b}$?

- Find a matrix $\boldsymbol{F} \in \mathbb{R}^{(m-n)\times m}$ such that $\boldsymbol{FA} = \boldsymbol{O}$
- Premultiply both sides of $\boldsymbol{Ax} + \boldsymbol{e} = \boldsymbol{b}$ by $\boldsymbol{F}$ to obtain, $\boldsymbol{FAx} + \boldsymbol{Fe} = \boldsymbol{Fb}$
- Let $\boldsymbol{z} = \boldsymbol{Fb}$
- Then, since $\boldsymbol{FAx} = \boldsymbol{0}$, we obtain

$$\boldsymbol{Fe} = \boldsymbol{z},$$

  where $\boldsymbol{z}$ is known
- Thus the original problem has been reduced to reconstructing the sparse error vector $\boldsymbol{e}$ from under-determined system of equations

Finding the sparsest solution to $\boldsymbol{Fe} = \boldsymbol{z}$

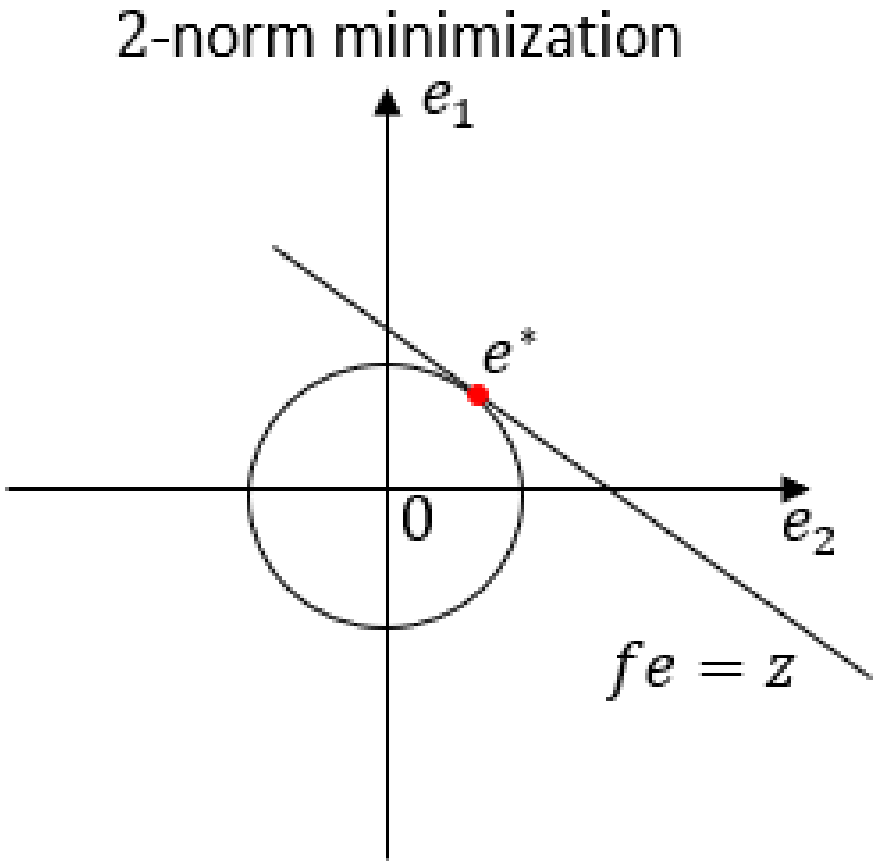
Definition (0-norm of a vector)

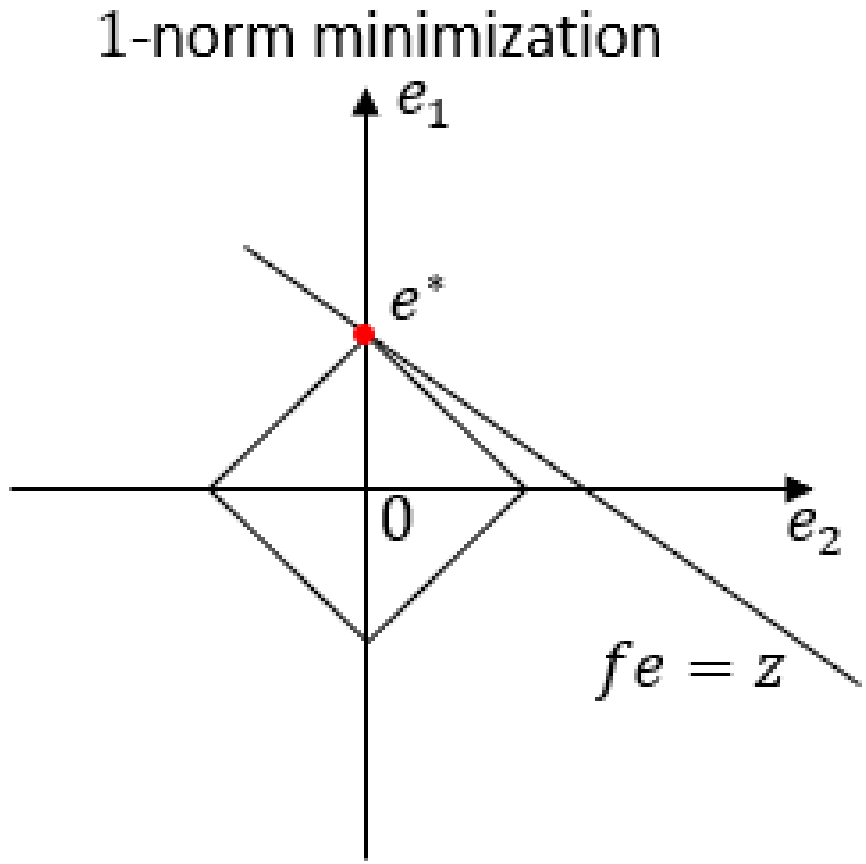
The 0-norm of a finite dimensional vector $\boldsymbol{x}$, denoted $\|\boldsymbol{x}\|_0$, is the number of nonzero entries in $\boldsymbol{x}$

Definition (Finding the sparsest solution problem)

$$\begin{aligned} &\min \|\boldsymbol{e}\|_0, \quad \boldsymbol{e} \in \mathbb{R}^m \\ &\text{subject to} \quad \boldsymbol{Fe} = \boldsymbol{z} \end{aligned}$$

Minimizing $\|\boldsymbol{e}\|$ subject to $\boldsymbol{Fe} = \boldsymbol{z}$





The minimal 1-norm solution is the sparsest

- D. L. Donoho and M. Elad, *For most large underdetermined systems of linear equations the minimal l_1-norm solution is also the sparsest solution*, SIAM Review, Vol. 56, No. 6, pp. 797–829, 2006
- Therefore, instead of minimizing $\|\boldsymbol{e}\|_0$, we consider an optimization problem where we minimize the 1-norm of a solution subject to the constraint, $\boldsymbol{F}\boldsymbol{e} = \boldsymbol{z}$

Finding the minimal 1-norm solution

- Since $\|\boldsymbol{e}\|_1 = \sum_{i=1}^{m} |e_i|$ is a convex function, we have a convex optimization problem,

$$\begin{aligned} &\min \|\boldsymbol{e}\|_1, \quad e \in \mathbb{R}^m \\ &\text{subject to} \quad \boldsymbol{F}\boldsymbol{e} = \boldsymbol{z} \end{aligned}$$

- Our objective: Find the unique solution $\boldsymbol{e}$ to the above problem
- Once we find $\boldsymbol{e}$, we can then recover $\boldsymbol{x}$

Sparse Vectors

Definition (i-sparse vector)

A vector $\boldsymbol{e}$ is i-sparse if it has at most i non-zero components, that is, $\|\boldsymbol{e}\|_0 \leq i$

Example

Let

$$\boldsymbol{e} = \begin{bmatrix} 0 \\ 0 \\ -1 \\ -1 \end{bmatrix}.$$

Then, $\|\boldsymbol{e}\|_0 = 2$

A Very Important Technical Result

- Consider an under-determined system, $\boldsymbol{F}\boldsymbol{e} = \boldsymbol{z}$, where $\boldsymbol{F}$ and $\boldsymbol{z}$ are given
- Let $\Sigma_i = \{\boldsymbol{e} : \|\boldsymbol{e}\|_0 \leq i\}$ be the set of all i-sparse vectors
- Let $\mathcal{N}(\boldsymbol{F})$ denote the null space of the matrix $\boldsymbol{F}$

Lemma

If $\Sigma_{2i} \cap \mathcal{N}(\boldsymbol{F}) = \{\boldsymbol{0}\}$, then any i-sparse solution of the under-determined system $\boldsymbol{F}\boldsymbol{e} = \boldsymbol{z}$ is unique

Proof of Lemma

- By contradiction: $S_1 \implies S_2 \iff \text{NOT}\,(S_1 \text{ AND NOT } S_2)$
- Suppose $\boldsymbol{e}^{(1)}$ and $\boldsymbol{e}^{(2)}$ are two different i-sparse solutions of the under-determined system $\boldsymbol{F}\boldsymbol{e} = \boldsymbol{z}$
- Then $\boldsymbol{F}(\boldsymbol{e}^{(1)} - \boldsymbol{e}^{(2)}) = \boldsymbol{0}$ and thus $\boldsymbol{e}^{(1)} - \boldsymbol{e}^{(2)} \in \mathcal{N}(F)$
- Since $\boldsymbol{e}^{(1)}$ and $\boldsymbol{e}^{(2)}$ are in Σ_i, we also have $\boldsymbol{e}^{(1)} - \boldsymbol{e}^{(2)} \in \Sigma_{2i}$
- Therefore $\boldsymbol{e}^{(1)} - \boldsymbol{e}^{(2)} \in \Sigma_{2i} \cap \mathcal{N}(F) = \{\boldsymbol{0}\}$
- It follows that we must have $\boldsymbol{e}^{(1)} = \boldsymbol{e}^{(2)}$, a contradiction, and thus an i-sparse solution of the under-determined system $\boldsymbol{F}\boldsymbol{e} = \boldsymbol{z}$ must be unique

□

The Spark of a Matrix

Definition

The spark of the matrix $\boldsymbol{F}$ is the smallest number of linearly dependent columns in $\boldsymbol{F}$, that is,

$$\text{spark}(\boldsymbol{F}) = \min\{\|\boldsymbol{d}\|_0 : \boldsymbol{F}\boldsymbol{d} = \boldsymbol{0}, \boldsymbol{d} \neq \boldsymbol{0}\}$$

Example

$$\text{spark} \begin{bmatrix} 1 & 1 & 3 & 0 \\ 1 & 1 & 2 & 0 \\ 1 & 1 & 4 & 0 \\ 0 & 1 & 3 & -1 \end{bmatrix} = 3$$

Indeed

- No zero column so no set of one columns linearly dependent
- No set of two columns that are linearly dependent
- There is a set of three columns that are linearly dependent; the first, the second, and the fourth columns are linearly dependent

Some Properties of the Spark of a Matrix

Let $\boldsymbol{A}$ be an $m \times n$ matrix, where $m \geq n$.

- Then, $\text{spark}(\boldsymbol{A}) = n + 1 \iff \text{rank}(\boldsymbol{A}) = n$, that is, the spark of $\boldsymbol{A}$ equals $n + 1$ if and only if $\boldsymbol{A}$ is a full column rank matrix
- $\text{spark}(\boldsymbol{A}) = 1 \iff \boldsymbol{A}$ has a zero column
- If $\text{spark}(\boldsymbol{A}) \neq n + 1$, then

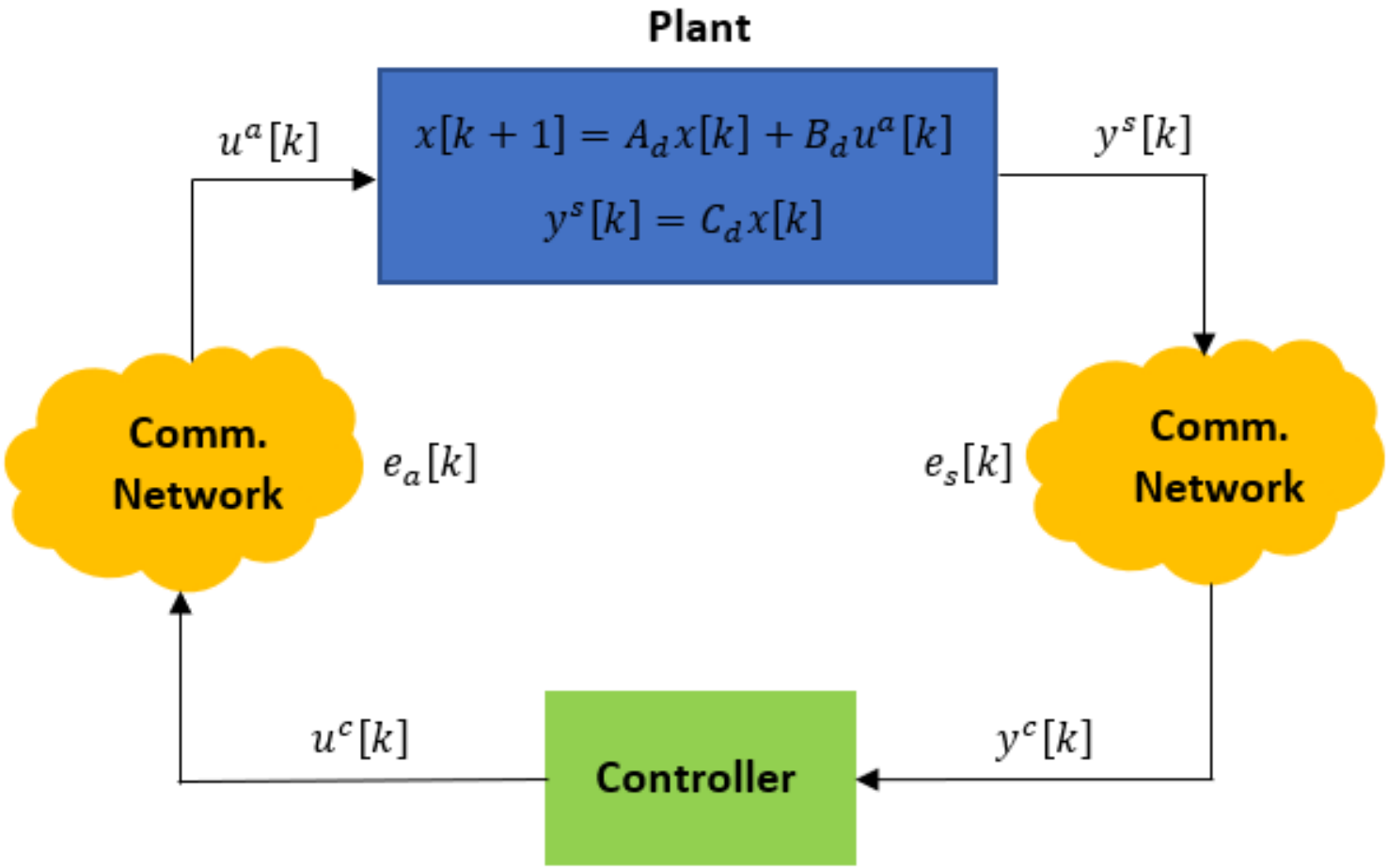
$$\text{spark}(\boldsymbol{A}) \leq \text{rank}(\boldsymbol{A}) + 1$$

Restatement of the Very Important Technical Result

Corollary:

spark($\boldsymbol{F}$) $> 2i$ is equivalent to $\Sigma_{2i} \cap \mathcal{N}(\boldsymbol{F}) = \{\mathbf{0}\}$. Therefore, spark($\boldsymbol{F}$) $> 2i$ implies that the i-sparse solution to $\boldsymbol{Fe} = \boldsymbol{z}$ is unique

Secure State Estimation of Networked Control Systems Corrupted by Unknown Input and Output Sparse Errors



Plant Design Model

$$\left.\begin{aligned} \boldsymbol{x}[k+1] &= \boldsymbol{A}\boldsymbol{x}[k] + \boldsymbol{B}\boldsymbol{u}^a[k] \\ \boldsymbol{y}^s[k] &= \boldsymbol{C}\boldsymbol{x}[k] \end{aligned}\right\}$$

where

- $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, $\boldsymbol{B} \in \mathbb{R}^{n\times m}$, $\boldsymbol{C} \in \mathbb{R}^{p\times n}$
- $\boldsymbol{B}$ full column rank, that is, $\operatorname{rank} \boldsymbol{B} = m$
- $\boldsymbol{u}^a[k] \in \mathbb{R}^m$—input received by actuators
- $\boldsymbol{y}^s[k] \in \mathbb{R}^p$—output measured by sensors

Malicious Attack Modeling

Modeling Malicious Attacks on Sensors

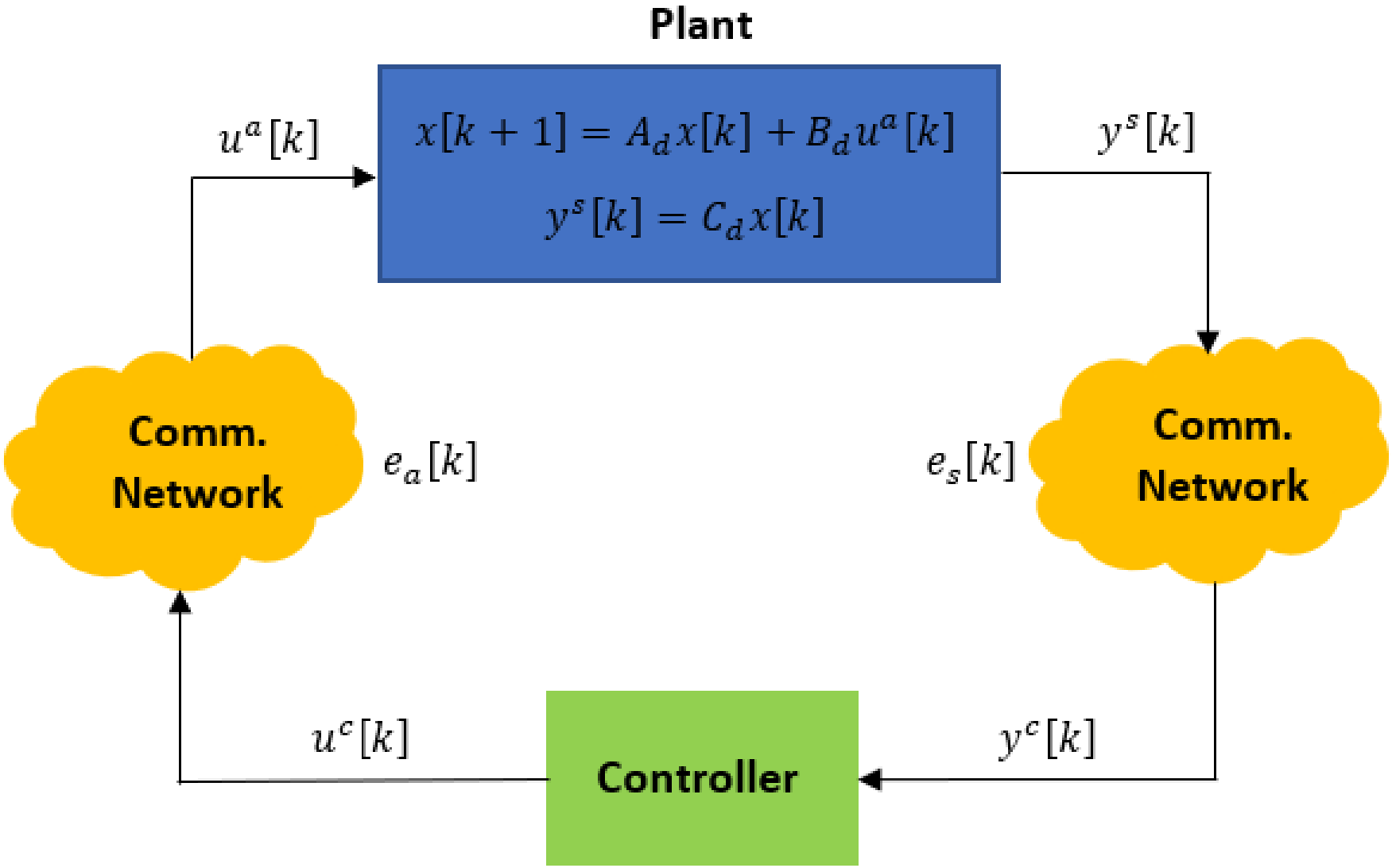
- Sensor measurements, $\boldsymbol{y}^s[k]$, are being sent to the controller through a communication network
- Malicious attacks cause packet drops in the communication network
- Malicious packet drops model:

$$\boldsymbol{\Gamma}(k) = \text{diag}\{\gamma_1(k), \gamma_2(k), \cdots, \gamma_p(k)\}$$

  where $\gamma_i(k), i = 1, \ldots, p$ are Boolean variables, $\gamma_i(k) = 1$ if the packet is correctly received; $\gamma_i(k) = 0$ if the packet is dropped
- Signal received by the controller:

$$\boldsymbol{y}^c[k] = \boldsymbol{\Gamma}(k)\boldsymbol{y}^s[k]$$

NCS


Plant
$x[k+1] = A_d x[k] + B_d u^a[k]$
$y^s[k] = C_d x[k]$
$u^a[k]$
$y^s[k]$
Comm. Network
$e_a[k]$
$e_s[k]$
Comm. Network
$u^c[k]$
Controller
$y^c[k]$

Modeling Malicious Attacks on Actuators

- The control signal is being sent to the plant through a communication network
- Malicious packet drops model:

$$\mathbf{\Lambda}(k) = \text{diag}\{\lambda_1(k), \lambda_2(k), \cdots, \lambda_m(k)\}$$

where $\lambda_i(k), i = 1, \ldots, m$ are Boolean variables, $\lambda_i(k) = 1$ if the packet is correctly received; $\lambda_i(k) = 0$ if the packet is dropped by the actuator

- Signal received by the actuator:

$$\boldsymbol{u}^a[k] = \mathbf{\Lambda}(k)\boldsymbol{u}^c[k]$$

Errors in communication between sensors and the controller

- Network communication errors in the communication flow from the sensor to the controller—$\boldsymbol{e}_s[k]$
- Hence,

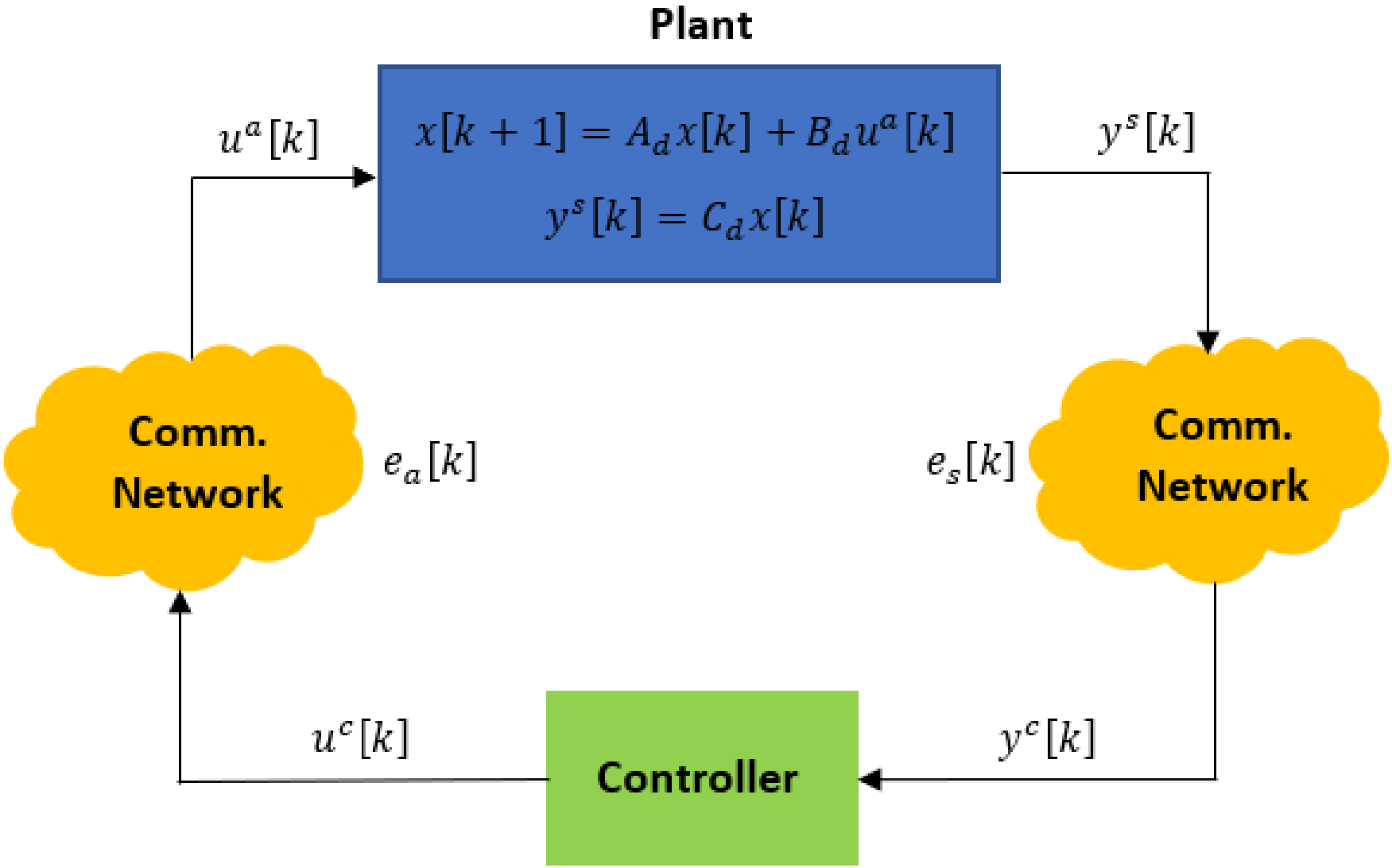
$$\boldsymbol{e}_s[k] = \boldsymbol{y}^c[k] - \boldsymbol{y}^s[k] \in \mathbb{R}^p$$

Errors in communication between the controller and actuators

- Errors in the communication between the controller to the actuator—$\boldsymbol{e}_a[k]$
- Hence,

$$\boldsymbol{e}_a[k] = \boldsymbol{u}^a[k] - \boldsymbol{u}^c[k] \in \mathbb{R}^m$$

NCS considered



NCS model

- Let $\overline{\boldsymbol{\Gamma}}(k) = \boldsymbol{\Gamma}(k) - \boldsymbol{I}_p \in \mathbb{R}^{p \times p}$ and $\overline{\boldsymbol{\Lambda}}(k) = \boldsymbol{\Lambda}(k) - \boldsymbol{I}_m \in \mathbb{R}^{m \times m}$
- Then

$$\boldsymbol{e}_s[k] = \overline{\boldsymbol{\Gamma}}(k)\boldsymbol{y}^s[k] \text{ and } \boldsymbol{e}_a[k] = \overline{\boldsymbol{\Lambda}}(k)\boldsymbol{u}^c[k]$$

- We analyze the case when malicious packet drops are sparse
- The system model under consideration

$$\left.\begin{aligned} \boldsymbol{x}[k+1] &= \boldsymbol{A}\boldsymbol{x}[k] + \boldsymbol{B}(\boldsymbol{u}^c[k] + \boldsymbol{e}_a[k]) \\ \boldsymbol{y}^c[k] &= \boldsymbol{C}\boldsymbol{x}[k] + \boldsymbol{e}_s[k] \end{aligned}\right\}$$

- **Objective**: obtain an estimate of the state $\boldsymbol{x}[k]$ of the NCS in the presence of malicious packet drops $\boldsymbol{e}_s[k]$ and $\boldsymbol{e}_a[k]$

An alternative approach to the problem

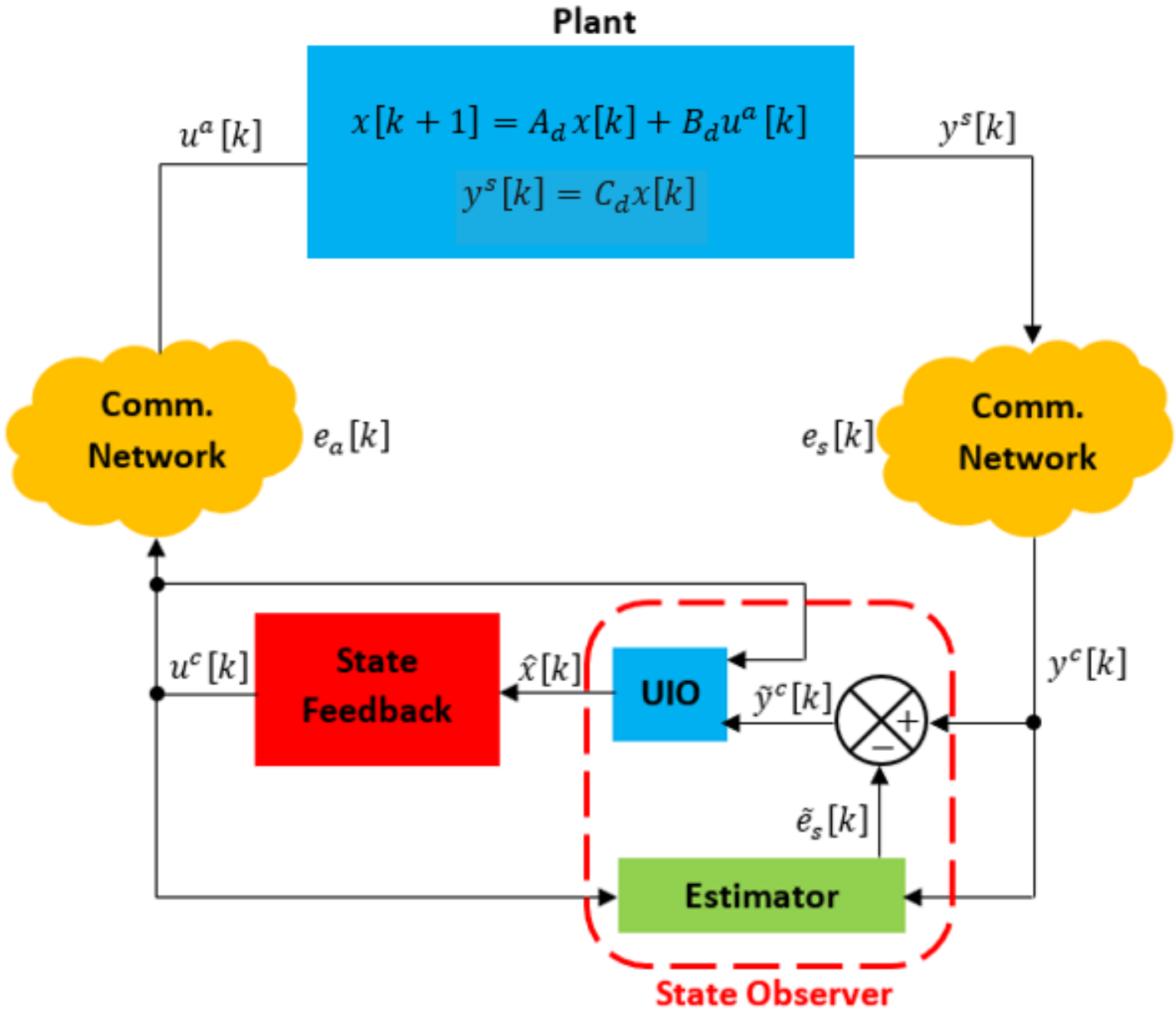
- Plant linear model

$$\left.\begin{aligned} \boldsymbol{x}[k+1] &= \boldsymbol{A}\boldsymbol{x}[k] + \boldsymbol{B}(\boldsymbol{u}^c[k] + \boldsymbol{e}_a[k]) \\ \boldsymbol{y}^c[k] &= \boldsymbol{C}\boldsymbol{x}[k] + \boldsymbol{e}_s[k] \end{aligned}\right\}$$

- Communication links subject to attacks
  - $\boldsymbol{e}_a[k]$—sparse attacks injected in the actuators
  - $\boldsymbol{e}_s[k]$—sparse attacks injected in the sensors
- **Objective**: correctly estimate the initial state

H. Fawzi, P. Tabuada, S. Diggavi, *Secure estimation and control for cyber-physical systems under adversarial attacks*, IEEE TAC, Vol. 59, No. 6, pp. 1454–1467, June 2014

State observer



State observer construction

- First, construct an estimator of network communication errors, $\boldsymbol{e}_s[k]$, in the signal flow from the sensor to the controller
- Use the estimation $\tilde{\boldsymbol{e}}_s[k]$ of $\boldsymbol{e}_s[k]$ to cancel out its effects
- Then, build **unknown input observer (UIO)** to estimate the plant state

Recovering error vector $\boldsymbol{e}_s[k]$

- Substitute $\boldsymbol{u}^a[k] = \boldsymbol{\Lambda}(k)\boldsymbol{u}^c[k]$ and $\boldsymbol{y}^c[k] = \boldsymbol{\Gamma}(k)\boldsymbol{y}^s[k]$ into

$$\left.\begin{array}{rcl} \boldsymbol{x}[k+1] & = & \boldsymbol{A}\boldsymbol{x}[k] + \boldsymbol{B}\boldsymbol{u}^a[k] \\ \boldsymbol{y}^s[k] & = & \boldsymbol{C}\boldsymbol{x}[k] \end{array}\right\}$$

  to obtain

$$\left.\begin{array}{rcl} \boldsymbol{x}[k+1] & = & \boldsymbol{A}\boldsymbol{x}[k] + \boldsymbol{B}\boldsymbol{\Lambda}(k)\boldsymbol{u}^c[k] \\ \boldsymbol{y}^c[k] & = & \boldsymbol{\Gamma}(k)\boldsymbol{C}\boldsymbol{x}[k] \end{array}\right\}$$

- The controller output $\boldsymbol{y}^c[k]$ and the controller input $\boldsymbol{u}^c[k]$ are known at all time
- Collect τ observations for the system

Notation

- For a given a vector $\boldsymbol{y}[k] \in \mathbb{R}^p$, $\tau \in \mathbb{N}$, the vector

$$\boldsymbol{y}^c|_{[k-\tau+1,k]}$$

  denotes the collection of τ samples of $\boldsymbol{y}[k]$
- That is,

$$\boldsymbol{y}^c|_{[k-\tau+1,k]} = \begin{bmatrix} \boldsymbol{y}^c[k-\tau+1] \\ \boldsymbol{y}^c[k-\tau+2] \\ \vdots \\ \boldsymbol{y}^c[k-1] \\ \boldsymbol{y}^c[k] \end{bmatrix}$$

Collecting observations

- Recall that $\overline{\boldsymbol{\Gamma}}(k) = \boldsymbol{\Gamma}(k) - \boldsymbol{I}_p \in \mathbb{R}^{p \times p}$
- One observation

$$\boldsymbol{y}^c|_{[k,k]} = \boldsymbol{\Gamma}(k)\boldsymbol{C}\boldsymbol{x}[k] = \boldsymbol{C}\boldsymbol{x}[k] + \overline{\boldsymbol{\Gamma}}(k)\boldsymbol{C}\boldsymbol{x}[k]$$

- Two observations starting from $\boldsymbol{x}[k-1]$
  - First observation at time $k-1$

$$\begin{aligned} \boldsymbol{y}^c[k-1] &= \boldsymbol{\Gamma}(k-1)\boldsymbol{C}\boldsymbol{x}[k-1] \\ &= \boldsymbol{C}\boldsymbol{x}[k-1] + \overline{\boldsymbol{\Gamma}}(k-1)\boldsymbol{C}\boldsymbol{x}[k-1] \end{aligned}$$

  - Second observation at time k

$$\begin{aligned} \boldsymbol{y}^c[k] &= \boldsymbol{\Gamma}(k)\boldsymbol{C}\boldsymbol{x}[k] = \boldsymbol{C}\boldsymbol{x}[k] + \overline{\boldsymbol{\Gamma}}(k)\boldsymbol{C}\boldsymbol{x}[k] \\ &= \boldsymbol{C}(\boldsymbol{A}\boldsymbol{x}[k-1] + \boldsymbol{B}\boldsymbol{\Lambda}(k-1)\boldsymbol{u}^c[k-1]) \\ &\quad + \overline{\boldsymbol{\Gamma}}(k)\boldsymbol{C}\boldsymbol{x}[k] \\ &= \boldsymbol{C}\boldsymbol{A}\boldsymbol{x}[k-1] + \overline{\boldsymbol{\Gamma}}(k)\boldsymbol{C}\boldsymbol{x}[k] \\ &\quad + \boldsymbol{C}\boldsymbol{B}\boldsymbol{\Lambda}(k-1)\boldsymbol{u}^c[k-1] \end{aligned}$$

Collecting two observations together

$$
\begin{aligned}
\boldsymbol{y}^c|_{[k-1,k]} &= \begin{bmatrix} \boldsymbol{y}^c[k-1] \\ \boldsymbol{y}^c[k] \end{bmatrix} \\
&= \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{CA} \end{bmatrix} \boldsymbol{x}[k-1] + \begin{bmatrix} \overline{\boldsymbol{\Gamma}}(k-1)\boldsymbol{C}\boldsymbol{x}[k-1] \\ \overline{\boldsymbol{\Gamma}}(k)\boldsymbol{C}\boldsymbol{x}[k] \end{bmatrix} \\
&\quad + \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{CB}\boldsymbol{\Lambda}(k-1)\boldsymbol{u}^c[k-1] \end{bmatrix}
\end{aligned}
$$

Collecting three observations together

$$
\begin{aligned}
\boldsymbol{y}^c|_{[k-2,k]} &= \begin{bmatrix} \boldsymbol{y}^c[k-2] \\ \boldsymbol{y}^c[k-1] \\ \boldsymbol{y}^c[k] \end{bmatrix} \\
&= \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{CA} \\ \boldsymbol{CA}^2 \end{bmatrix} \boldsymbol{x}[k-2] + \begin{bmatrix} \overline{\boldsymbol{\Gamma}}(k-2)\boldsymbol{C}\boldsymbol{x}[k-2] \\ \overline{\boldsymbol{\Gamma}}(k-1)\boldsymbol{C}\boldsymbol{x}[k-1] \\ \overline{\boldsymbol{\Gamma}}(k)\boldsymbol{C}\boldsymbol{x}[k] \end{bmatrix} \\
&\quad + \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{CB}\boldsymbol{\Lambda}(k-2)\boldsymbol{u}^c[k-2] \\ \boldsymbol{CB}\boldsymbol{\Lambda}(k-1)\boldsymbol{u}^c[k-1] + \boldsymbol{CAB}\boldsymbol{\Lambda}(k-2)\boldsymbol{u}^c[k-2] \end{bmatrix}
\end{aligned}
$$

Collecting τ observations

$$
\begin{aligned}
\boldsymbol{y}^c|_{[k-\tau+1,k]} &= \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{CA} \\ \vdots \\ \boldsymbol{CA}^{\tau-1} \end{bmatrix} \boldsymbol{x}[k-\tau+1] \\
&+ \begin{bmatrix} \overline{\boldsymbol{\Gamma}}(k-\tau+1)\boldsymbol{C}\boldsymbol{x}[k-\tau+1] \\ \overline{\boldsymbol{\Gamma}}(k-\tau+2)\boldsymbol{C}\boldsymbol{x}[k-\tau+2] \\ \vdots \\ \overline{\boldsymbol{\Gamma}}(k)\boldsymbol{C}\boldsymbol{x}[k] \end{bmatrix} \\
&+ \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{CB\Lambda}(k-\tau+1)\boldsymbol{u}^c[k-\tau+1] \\ \vdots \\ \Sigma_{i=1}^{\tau-1}\boldsymbol{CA}^{\tau-1-i}\boldsymbol{B\Lambda}(k-\tau+i)\boldsymbol{u}^c[k-\tau+i] \end{bmatrix}
\end{aligned}
$$

How many observations do we need?

- Consider a system model

$$\left.\begin{aligned} \boldsymbol{x}[k+1] &= \boldsymbol{A}\boldsymbol{x}[k] \\ \boldsymbol{y}^c[k] &= \boldsymbol{\Gamma}(k)\boldsymbol{C}\boldsymbol{x}[k] \end{aligned}\right\}$$

- Collect τ observations:

$$\begin{aligned} \boldsymbol{y}^c|_{[0,\tau-1]} &= \begin{bmatrix} \boldsymbol{\Gamma}(0)\boldsymbol{C} \\ \boldsymbol{\Gamma}(1)\boldsymbol{C}\boldsymbol{A} \\ \vdots \\ \boldsymbol{\Gamma}(\tau-1)\boldsymbol{C}\boldsymbol{A}^{\tau-1} \end{bmatrix} \boldsymbol{x}[0] \\ &= \text{diag}\{\ \boldsymbol{\Gamma}(0) \quad \boldsymbol{\Gamma}(1) \quad \cdots \quad \boldsymbol{\Gamma}(\tau-1)\ \}\mathcal{O}^{\tau-1}\boldsymbol{x}[0] \end{aligned}$$

where $\mathcal{O}^{\tau-1}$ is the τ-step observability matrix

More Notation

- For a given matrix $\boldsymbol{M} \in \mathbb{R}^{n \times m}$ and a set $\Xi \subseteq \{1, \ldots, n\}$, denote

$$\boldsymbol{M}_{\bar{\Xi}} \in \mathbb{R}^{(n-|\Xi|) \times m}$$

  the matrix obtained from $\boldsymbol{M}$ by removing the rows whose indices are contained in Ξ
- For a given matrix $\boldsymbol{M} \in \mathbb{R}^{n \times m}$ and a set $\Xi \subseteq \{1, \ldots, n\}$, denote

$$\boldsymbol{M}_{\Xi} \in \mathbb{R}^{|\Xi| \times m}$$

  the matrix obtained from $\boldsymbol{M}$ by removing the rows whose indices are **not** contained in Ξ

$\mathcal{O}^{\tau-1}$—τ-step observability matrix

$$\mathcal{O}^{\tau-1} = \begin{bmatrix} \boldsymbol{C}\boldsymbol{A}^{\tau-1} \\ \vdots \\ \boldsymbol{C}\boldsymbol{A} \\ \boldsymbol{C} \end{bmatrix}$$

Resilient System Against Packet Drops

Definition

The linear system

$$\left.\begin{aligned} \boldsymbol{x}[k+1] &= \boldsymbol{A}\boldsymbol{x}[k] \\ \boldsymbol{y}^c[k] &= \boldsymbol{\Gamma}(k)\boldsymbol{C}\boldsymbol{x}[k] \end{aligned}\right\}$$

is said to be resilient against d_s packet drops if there exists $\tau \in \mathbb{N}$ such that for any set $\Xi \subseteq \{1, \ldots, \xi\}$ with $|\Xi| \leq d_s$ the matrix $\mathcal{O}_{\bar{\Xi}}^{\tau\,-1}$ has full column rank

---

G. Fiore, Y. H. Chang, Q. Hu, M. D. Di Benedetto, and C. J. Tomlin, *Secure state estimation for Cyber Physical Systems with sparse malicious packet drops*, 2017 ACC, Sheraton Seattle Hotel, Seattle, May 24–26, pp. 1898–1903

Some Manipulations

- Let $\boldsymbol{U}^c[k] \in \mathbb{R}^{m \times m}$ be a diagonal matrix whose components consist of $\boldsymbol{u}^c[k]$
- Let vec $(\boldsymbol{\Lambda}(k)) \in \mathbb{R}^m$ represents vectorization of diagonal components of $\boldsymbol{\Lambda}(k)$
- Then,
$$\boldsymbol{\Lambda}(k)\boldsymbol{u}^c[k] = \boldsymbol{U}^c[k]\text{vec } (\boldsymbol{\Lambda}(k))$$
- Let $\boldsymbol{v}[k] = [\boldsymbol{0}^\top \ \cdots \ \Sigma_{i=1}^{\tau-1}(\boldsymbol{C}\boldsymbol{A}^{\tau-1-i}\boldsymbol{B}\boldsymbol{u}^c(k-\tau+i))^\top]^\top$
- Note that $\boldsymbol{v}[k]$ is known for all k and τ

M. Zhang, S. Hui, M. R. Bell, and S. H. Żak, Vector Recovery for a Linear System Corrupted by Unknown Sparse Error Vectors With Applications to Secure State Estimation, *IEEE Control Systems Letters*, Vol. 3, No. 4, pp. 895–900, October 2019

More Manipulations

- Let $\hat{\boldsymbol{y}}^c|_{[k-\tau+1,k]} = \boldsymbol{y}^c|_{[k-\tau+1,k]} - \boldsymbol{v}[k]$
- Then,

$$\boldsymbol{Y}[k] \triangleq \begin{bmatrix} \hat{\boldsymbol{y}}^c[k] \\ \hat{\boldsymbol{y}}^c[k-1] \\ \vdots \\ \hat{\boldsymbol{y}}^c[k-\tau+1] \end{bmatrix} = \begin{bmatrix} \boldsymbol{C}\boldsymbol{A}^{\tau-1} \\ \boldsymbol{C}\boldsymbol{A}^{\tau-2} \\ \vdots \\ \boldsymbol{C} \end{bmatrix} \boldsymbol{x}[k-\tau+1]$$

$$+ \boldsymbol{I}_{\tau p} \begin{bmatrix} \overline{\boldsymbol{\Gamma}}(k)\boldsymbol{C}\boldsymbol{x}[k] \\ \overline{\boldsymbol{\Gamma}}(k-1)\boldsymbol{C}\boldsymbol{x}[k-1] \\ \vdots \\ \overline{\boldsymbol{\Gamma}}(k-\tau+1)\boldsymbol{C}\boldsymbol{x}[k-\tau+1] \end{bmatrix} + \boldsymbol{F}[k] \begin{bmatrix} \mathrm{vec}(\overline{\boldsymbol{\Lambda}}(k-1)) \\ \mathrm{vec}(\overline{\boldsymbol{\Lambda}}(k-2)) \\ \vdots \\ \mathrm{vec}(\overline{\boldsymbol{\Lambda}}(k-\tau+1)) \end{bmatrix}$$

$$\triangleq \mathcal{O}^{\tau-1}\boldsymbol{x}[k-\tau+1] + \boldsymbol{I}_{\tau p}\boldsymbol{E}_s[k] + \boldsymbol{F}[k]\mathcal{V}[k]$$

Organizing Output Observations for Further Processing

- We have

$$\boldsymbol{Y}[k] = \mathcal{O}^{\tau-1}\boldsymbol{x}[k-\tau+1] + \boldsymbol{I}_{\tau p}\boldsymbol{E}_s[k] + \boldsymbol{F}[k]\mathcal{V}[k]$$

  where
  - $\mathcal{O}^{\tau-1} \in \mathbb{R}^{\tau p \times n}$
  - $\boldsymbol{Y}[k] \in \mathbb{R}^{\tau p}$
  - $\boldsymbol{F}[k] \in \mathbb{R}^{\tau p \times (\tau-1)m}$
  - $$\boldsymbol{F}[k] = \begin{bmatrix} \boldsymbol{CBU}^c[k-1] & \cdots & \boldsymbol{CA}^{\tau-2}\boldsymbol{BU}^c[k-\tau+1] \\ \vdots & \ddots & \vdots \\ \boldsymbol{O}_{p\times m} & \cdots & \boldsymbol{CBU}^c[k-\tau+1] \\ \boldsymbol{O}_{p\times m} & \cdots & \boldsymbol{O}_{p\times m} \end{bmatrix}$$

Organizing the Observations for Further Processing—Final Form

- Let $\boldsymbol{\Omega}[k] = [\boldsymbol{I}_{\tau p} \quad \boldsymbol{F}[k]]$
- Let $\boldsymbol{E}[k] = [\boldsymbol{E}_s^\top[k] \quad \mathcal{V}^\top[k]]^\top$
- Then

$$\boldsymbol{Y}[k] = \mathcal{O}^{\tau-1}\boldsymbol{x}[k-\tau+1] + \boldsymbol{\Omega}[k]\boldsymbol{E}[k]$$

  where $\boldsymbol{\Omega} \in \mathbb{R}^{\tau p \times [\tau p + (\tau-1)m]}$ and $\boldsymbol{E} \in \mathbb{R}^{\tau p + (\tau-1)m}$
- Objective: Recover $\boldsymbol{E}[k]$, a sparse vector

Background Results

- Let $\boldsymbol{M} \in \mathbb{R}^{m \times n}$, $\text{rank}\boldsymbol{M} = n$
- Take the qr-decomposition of $\boldsymbol{M}$ to obtain

$$\boldsymbol{M} = \boldsymbol{Q}\boldsymbol{R} = \begin{bmatrix} \boldsymbol{Q}_1 & \boldsymbol{Q}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{R}_1 \\ \boldsymbol{O} \end{bmatrix},$$

where $\boldsymbol{Q} \in \mathbb{R}^{m \times m}$ is orthogonal, $\boldsymbol{Q}_1 \in \mathbb{R}^{m \times n}$, $\boldsymbol{Q}_2 \in \mathbb{R}^{m \times (m-n)}$, and $\boldsymbol{R}_1 \in \mathbb{R}^{n \times n}$ is a full rank upper triangular matrix.

Lemma

Let $\boldsymbol{Q}_2$ be defined as above. Then $\boldsymbol{Q}_2^\top$ is a left annihilator of $\boldsymbol{M}$, that is, $\boldsymbol{Q}_2^\top \boldsymbol{M} = \boldsymbol{O}$

Constructing Left Annihilator of $\mathcal{O}^{\tau-1}$

- Take the qr decomposition of $\mathcal{O}^{\tau-1}$ to obtain

$$\mathcal{O}^{\tau-1} = \boldsymbol{Q}\boldsymbol{R} = \begin{bmatrix} \boldsymbol{Q}_1 & \boldsymbol{Q}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{R}_1 \\ \boldsymbol{O} \end{bmatrix}$$

  where
  - $\boldsymbol{Q} \in \mathbb{R}^{\tau p \times \tau p}$ is orthogonal
  - $\boldsymbol{Q}_1 \in \mathbb{R}^{\tau p \times n}$
  - $\boldsymbol{Q}_2 \in \mathbb{R}^{\tau p \times (\tau p - n)}$
  - $\boldsymbol{R}_1 \in \mathbb{R}^{n \times n}$ is a full rank upper triangular matrix
- By the Lemma, $\boldsymbol{Q}_2^\top$ is a left annihilator of $\mathcal{O}^{\tau-1}$, that is, $\boldsymbol{Q}_2^\top \mathcal{O}^{\tau-1} = \boldsymbol{O}$

$\boldsymbol{Q}_2^\top$ is a Left Annihilator of $\mathcal{O}^{\tau-1}$

- We have

$$\mathcal{O}^{\tau-1} = \boldsymbol{Q}\boldsymbol{R} = \begin{bmatrix} \boldsymbol{Q}_1 & \boldsymbol{Q}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{R}_1 \\ \boldsymbol{O} \end{bmatrix} = \boldsymbol{Q}_1\boldsymbol{R}_1$$

- Note that

$$\begin{aligned} \boldsymbol{Q}^\top\boldsymbol{Q} &= \begin{bmatrix} \boldsymbol{Q}_1^\top \\ \boldsymbol{Q}_2^\top \end{bmatrix} \begin{bmatrix} \boldsymbol{Q}_1 & \boldsymbol{Q}_2 \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{Q}_1^\top\boldsymbol{Q}_1 & \boldsymbol{Q}_1^\top\boldsymbol{Q}_2 \\ \boldsymbol{Q}_2^\top\boldsymbol{Q}_1 & \boldsymbol{Q}_2^\top\boldsymbol{Q}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{I}_n & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{I}_{\tau p-n} \end{bmatrix} \end{aligned}$$

- Hence, $\boldsymbol{Q}_2^\top\boldsymbol{Q}_1 = \boldsymbol{O}$
- Therefore

$$\boldsymbol{Q}_2^\top\mathcal{O}^{\tau-1} = \boldsymbol{Q}_2^\top(\boldsymbol{Q}_1\boldsymbol{R}_1) = \boldsymbol{O}$$

Quick Way to Compute Left Annihilator of $\mathcal{O}^{\tau-1}$

- Use MATLAB's `null` function to obtain

$$\boldsymbol{Q}_2^\top = \text{null}\left(\left(\mathcal{O}^{\tau-1}\right)^\top\right)^\top,$$

  where $\boldsymbol{Q}_2$ is right annihilator of $\left(\mathcal{O}^{\tau-1}\right)^\top$
- Indeed, since

$$\left(\mathcal{O}^{\tau-1}\right)^\top \boldsymbol{Q}_2 = \boldsymbol{O},$$

  then taking the transpose gives

$$\boldsymbol{Q}_2^\top \mathcal{O}^{\tau-1} = \boldsymbol{O}^\top$$

Constructing an Optimization Problem for $\boldsymbol{E}[k]$ Recovery

- Recall that $\boldsymbol{Q}_2^\top \mathcal{O}^{\tau-1} = \boldsymbol{O}$
- Pre-multiply

$$\boldsymbol{Y}[k] = \mathcal{O}^{\tau-1}\boldsymbol{x}[k-\tau+1] + \boldsymbol{\Omega}[k]\boldsymbol{E}[k]$$

  by $\boldsymbol{Q}_2^\top$
- We obtain, $\boldsymbol{Q}_2^\top \boldsymbol{Y}[k] = \boldsymbol{Q}_2^\top \boldsymbol{\Omega}[k]\boldsymbol{E}[k]$
- Let $\boldsymbol{Z}[k] = \boldsymbol{Q}_2^\top \boldsymbol{Y}[k]$ and $\boldsymbol{W}[k] = \boldsymbol{Q}_2^\top \boldsymbol{\Omega}[k]$
- Then

$$\boldsymbol{Z}[k] = \boldsymbol{W}[k]\boldsymbol{E}[k]$$

The Constraint in the Optimization Problem to Recover $\boldsymbol{E}[k]$

- We have
$$\boldsymbol{Z}[k] = \boldsymbol{W}[k]\boldsymbol{E}[k]$$
where
$\boldsymbol{Z}[k] \in \mathbb{R}^{\tau p - n}$ and $\boldsymbol{W}[k] \in \mathbb{R}^{(\tau p - n) \times [\tau p + (\tau - 1)m]}$
- Note that $\boldsymbol{W}[k]$ is full row rank
- That is,
$$\text{rank}(\boldsymbol{W}[k]) = \tau p - n$$
- This is because $\text{rank}(\boldsymbol{Q}_2^\top) = \tau p - n$, $\text{rank}(\boldsymbol{\Omega}[k]) = \tau p$
- Hence
$$\text{rank}(\boldsymbol{W}[k]) = \text{rank}(\boldsymbol{Q}_2^\top \boldsymbol{\Omega}[k]) = \text{rank}(\boldsymbol{Q}_2^\top)$$

Optimization Problem to Recover $\boldsymbol{E}[k]$

If $\boldsymbol{E}[k]$ is an i-sparse vector, the solution to

$$\boldsymbol{Z}[k] = \boldsymbol{W}[k]\boldsymbol{E}[k]$$

can be obtained by solving the optimization problem

$$\min \|\boldsymbol{E}[k]\|_0 \quad \text{subject to} \quad \boldsymbol{Z}[k] = \boldsymbol{W}[k]\boldsymbol{E}[k]$$

Assumptions for the Optimization Problem to Recover $\boldsymbol{E}[k]$

- Assume that over the time interval $[k-\tau+1, k]$ there are at most i_s malicious packet drops from the sensor to the controller and at most i_a malicious packet drops from the controller to the actuator
- Assume that $\boldsymbol{E}[k]$ is i-sparse
- Hence,

$$i = \|\boldsymbol{E}[k]\|_0 = \|\boldsymbol{E}_s[k]\|_0 + \|\boldsymbol{E}_a[k]\|_0 \leq i_s + i_a$$

Existence of the Solution to the Optimization Problem to Recover $\boldsymbol{E}[k]$

Lemma

If the solution $\boldsymbol{E}[k]$ to $\boldsymbol{Z}[k] = \boldsymbol{W}[k]\boldsymbol{E}[k]$ is i-sparse and $(\tau p - n) \geq 2(i_s + i_a)$ and all subsets of $2(i_s + i_a)$ columns of $\boldsymbol{W}[k]$ are full rank, then $\boldsymbol{E}[k]$ is unique

---

G. Fiore, Y. H. Chang, Q. Hu, M. D. Di Benedetto, C. J. Tomlin, *Secure state estimation for Cyber Physical Systems with sparse malicious packet drops*, 2017 ACC, Sheraton Seattle Hotel, Seattle, May 24–26, pp. 1898–1903

Approximating the Optimization Problem to Recover $\boldsymbol{E}[k]$

We approximate the the 0-norm optimization problem

$$\min \|\boldsymbol{E}[k]\|_0 \quad \text{subject to} \quad \boldsymbol{Z}[k] = \boldsymbol{W}[k]\boldsymbol{E}[k]$$

with the 1-norm optimization problem

$$\min \|\boldsymbol{E}[k]\|_1 \quad \text{subject to} \quad \boldsymbol{Z}[k] = \boldsymbol{W}[k]\boldsymbol{E}[k]$$

D. L. Donoho and M. Elad, *For most large under-determined systems of linear equations the minimal l_1-norm solution is also the sparsest solution*, SIAM Review, Vol. 56, No. 6, pp. 797–829, 2006

Converting the 1-Norm Optimization Into a Linear Programming Problem

- $\min \|\boldsymbol{E}[k]\|_1 = \min(\sum_{i=1}^{q} |E_i[k]|)$
- Let E_i^+, E_i^- be such that $|E_i| = E_i^+ + E_i^-$, $E_i = E_i^+ - E_i^-$, and $E_i^+ E_i^- = 0$
- Then we obtain

$$\begin{array}{ll} \min & (E_1^+ + E_1^-) + (E_2^+ + E_2^-) + \cdots + (E_q^+ + E_q^-) \\ \text{subject to} & \boldsymbol{W}(\boldsymbol{E}^+ - \boldsymbol{E}^-) = \boldsymbol{Z} \\ & \boldsymbol{E}^+, \boldsymbol{E}^- \geq 0, \end{array}$$

where $\boldsymbol{E}^+ = [E_1^+ \quad \cdots \quad E_q^+]^\top$, $\boldsymbol{E}^- = [E_1^- \quad \cdots \quad E_q^-]^\top$, and $q = \tau p + (\tau - 1)m$

Linear Programming Program in Standard Form

- Let $\boldsymbol{x}_{lp} = [\boldsymbol{E}^{+\top} \quad \boldsymbol{E}^{-\top}]^\top$
- Let $\boldsymbol{c} = [1 \quad \cdots \quad 1]^\top \in \mathbb{R}^{2q}$
- Let $\boldsymbol{A}_{lp} = [\boldsymbol{W} \quad -\boldsymbol{W}]$
- Then we have

$$\begin{aligned} \min \quad & \boldsymbol{c}^\top \boldsymbol{x}_{lp} \\ \text{subject to} \quad & \boldsymbol{A}_{lp}\boldsymbol{x}_{lp} = \boldsymbol{Z} \\ & \boldsymbol{x}_{lp} \geq 0 \end{aligned}$$

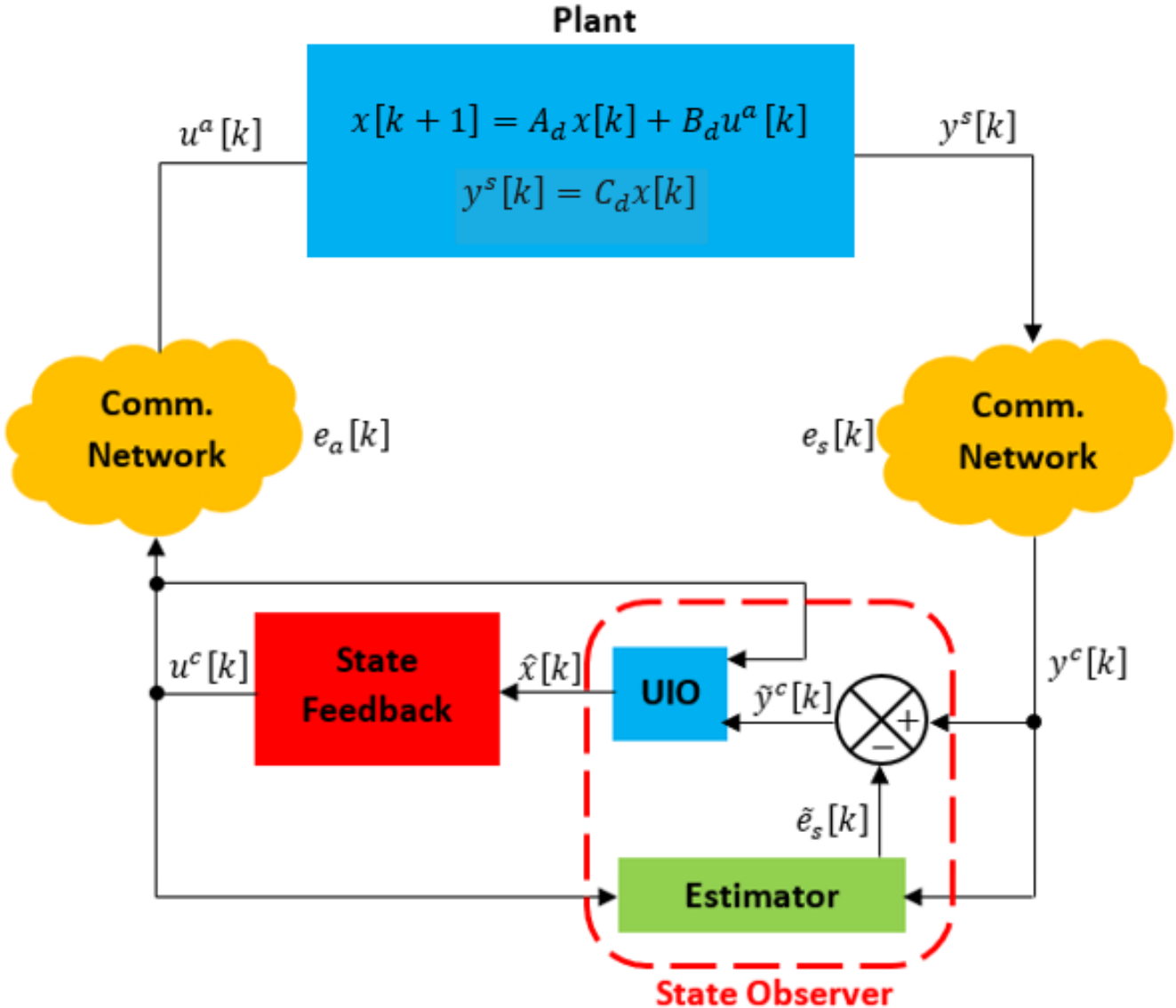
- The above linear programming problem can be solved using standard methods

Recovering Output Sensor Error $\boldsymbol{e}_s[k]$

1. Choose τ such that $(\tau p - n) \geq 2(i_s + i_a)$
2. Compute $\boldsymbol{Y}[k]$ and matrices $\mathcal{O}^{\tau-1}$ and $\boldsymbol{\Omega}[k]$
3. Find left annihilator, $\boldsymbol{Q}_2^\top$ of $\mathcal{O}^{\tau-1}$
4. Construct the optimization problem, where $\boldsymbol{Z}[k] = \boldsymbol{Q}_2^\top \boldsymbol{Y}[k]$, $\boldsymbol{W}[k] = \boldsymbol{Q}_2^\top \boldsymbol{\Omega}[k]$
5. Solve optimization problem for $\tilde{\boldsymbol{E}}[k]$
6. Compute $\tilde{\boldsymbol{e}}_s[k]$ that approximates $\boldsymbol{e}_s[k]$

Observer Architecture

State Observer = UIO + Output Sensor Error Estimator



Combined output sensor error estimator and UIO

- Construct an estimator of output sensor error $\boldsymbol{e}_s[k]$ to obtain its estimate denoted $\tilde{\boldsymbol{e}}_s[k]$
- Subtract $\tilde{\boldsymbol{e}}_s[k]$ from $\boldsymbol{y}^c[k]$ to obtain

$$\begin{aligned} \tilde{\boldsymbol{y}}^c[k] &= \boldsymbol{y}^c[k] - \tilde{\boldsymbol{e}}_s[k] \\ &= \boldsymbol{y}^s[k] + \boldsymbol{e}_s[k] - \tilde{\boldsymbol{e}}_s[k] \end{aligned}$$

- To proceed, assume $\tilde{\boldsymbol{y}}^c[k] = \boldsymbol{y}^s[k]$
- We obtain

$$\left.\begin{aligned} \boldsymbol{x}[k+1] &= \boldsymbol{A}\boldsymbol{x}[k] + \boldsymbol{B}(\boldsymbol{u}^c[k] + \boldsymbol{e}_a[k]) \\ \tilde{\boldsymbol{y}}^c[k] &= \boldsymbol{C}\boldsymbol{x}[k] \end{aligned}\right\}$$

Constructing UIO

- First decompose the state $\boldsymbol{x}[k]$ as

$$\begin{aligned}\boldsymbol{x}[k] &= \boldsymbol{x}[k] - \boldsymbol{M}\tilde{\boldsymbol{y}}^c[k] + \boldsymbol{M}\tilde{\boldsymbol{y}}^c[k] \\ &= \boldsymbol{x}[k] - \boldsymbol{MC}\boldsymbol{x}[k] + \boldsymbol{MC}\boldsymbol{x}[k] \\ &= (\boldsymbol{I} - \boldsymbol{MC})\boldsymbol{x}[k] + \boldsymbol{M}\tilde{\boldsymbol{y}}^c[k]\end{aligned}$$

  where $\boldsymbol{M} \in \mathbb{R}^{n\times p}$ is a parameter matrix to be constructed

- Let $\boldsymbol{z}[k] = (\boldsymbol{I} - \boldsymbol{MC})\boldsymbol{x}[k]$
- Then

$$\begin{aligned}\boldsymbol{z}[k+1] &= (\boldsymbol{I} - \boldsymbol{MC})\boldsymbol{x}[k+1] \\ &= (\boldsymbol{I} - \boldsymbol{MC})(\boldsymbol{A}\boldsymbol{x}[k] + \boldsymbol{B}\boldsymbol{u}^c[k] + \boldsymbol{B}\boldsymbol{e}_a[k]) \\ &= (\boldsymbol{I} - \boldsymbol{MC})(\boldsymbol{A}\boldsymbol{x}[k] + \boldsymbol{B}\boldsymbol{u}^c[k]) + (\boldsymbol{I} - \boldsymbol{MC})\boldsymbol{B}\boldsymbol{e}_a[k]\end{aligned}$$

Constructing UIO

- First decompose the state $\boldsymbol{x}[k]$ as

$$\begin{aligned}\boldsymbol{x}[k] &= \boldsymbol{x}[k] - \boldsymbol{M}\tilde{\boldsymbol{y}}^c[k] + \boldsymbol{M}\tilde{\boldsymbol{y}}^c[k] \\ &= \boldsymbol{x}[k] - \boldsymbol{MC}\boldsymbol{x}[k] + \boldsymbol{MC}\boldsymbol{x}[k] \\ &= (\boldsymbol{I} - \boldsymbol{MC})\boldsymbol{x}[k] + \boldsymbol{M}\tilde{\boldsymbol{y}}^c[k]\end{aligned}$$

  where $\boldsymbol{M} \in \mathbb{R}^{n\times p}$ is a parameter matrix to be constructed

- Let $\boldsymbol{z}[k] = (\boldsymbol{I} - \boldsymbol{MC})\boldsymbol{x}[k]$
- Then

$$\begin{aligned}\boldsymbol{z}[k+1] &= (\boldsymbol{I} - \boldsymbol{MC})\boldsymbol{x}[k+1] \\ &= (\boldsymbol{I} - \boldsymbol{MC})(\boldsymbol{A}\boldsymbol{x}[k] + \boldsymbol{B}\boldsymbol{u}^c[k] + \boldsymbol{B}\boldsymbol{e}_a[k]) \\ &= (\boldsymbol{I} - \boldsymbol{MC})(\boldsymbol{A}\boldsymbol{x}[k] + \boldsymbol{B}\boldsymbol{u}^c[k]) + (\boldsymbol{I} - \boldsymbol{MC})\boldsymbol{B}\boldsymbol{e}_a[k]\end{aligned}$$

Constructing UIO

- First decompose the state $\boldsymbol{x}[k]$ as

$$\begin{aligned}\boldsymbol{x}[k] &= \boldsymbol{x}[k] - \boldsymbol{M}\tilde{\boldsymbol{y}}^c[k] + \boldsymbol{M}\tilde{\boldsymbol{y}}^c[k] \\ &= \boldsymbol{x}[k] - \boldsymbol{MC}\boldsymbol{x}[k] + \boldsymbol{MC}\boldsymbol{x}[k] \\ &= (\boldsymbol{I} - \boldsymbol{MC})\boldsymbol{x}[k] + \boldsymbol{M}\tilde{\boldsymbol{y}}^c[k]\end{aligned}$$

  where $\boldsymbol{M} \in \mathbb{R}^{n\times p}$ is a parameter matrix to be constructed

- Let $\boldsymbol{z}[k] = (\boldsymbol{I} - \boldsymbol{MC})\boldsymbol{x}[k]$
- Then

$$\begin{aligned}\boldsymbol{z}[k+1] &= (\boldsymbol{I} - \boldsymbol{MC})\boldsymbol{x}[k+1] \\ &= (\boldsymbol{I} - \boldsymbol{MC})(\boldsymbol{A}\boldsymbol{x}[k] + \boldsymbol{B}\boldsymbol{u}^c[k] + \boldsymbol{B}\boldsymbol{e}_a[k]) \\ &= (\boldsymbol{I} - \boldsymbol{MC})(\boldsymbol{A}\boldsymbol{x}[k] + \boldsymbol{B}\boldsymbol{u}^c[k]) + (\boldsymbol{I} - \boldsymbol{MC})\boldsymbol{B}\boldsymbol{e}_a[k]\end{aligned}$$

Constructing UIO

- First decompose the state $\boldsymbol{x}[k]$ as

$$\begin{aligned}\boldsymbol{x}[k] &= \boldsymbol{x}[k] - \boldsymbol{M}\tilde{\boldsymbol{y}}^c[k] + \boldsymbol{M}\tilde{\boldsymbol{y}}^c[k] \\ &= \boldsymbol{x}[k] - \boldsymbol{M}\boldsymbol{C}\boldsymbol{x}[k] + \boldsymbol{M}\boldsymbol{C}\boldsymbol{x}[k] \\ &= (\boldsymbol{I} - \boldsymbol{M}\boldsymbol{C})\boldsymbol{x}[k] + \boldsymbol{M}\tilde{\boldsymbol{y}}^c[k]\end{aligned}$$

  where $\boldsymbol{M} \in \mathbb{R}^{n\times p}$ is a parameter matrix to be constructed
- Let $\boldsymbol{z}[k] = (\boldsymbol{I} - \boldsymbol{M}\boldsymbol{C})\boldsymbol{x}[k]$
- Then

$$\begin{aligned}\boldsymbol{z}[k+1] &= (\boldsymbol{I} - \boldsymbol{M}\boldsymbol{C})\boldsymbol{x}[k+1] \\ &= (\boldsymbol{I} - \boldsymbol{M}\boldsymbol{C})(\boldsymbol{A}\boldsymbol{x}[k] + \boldsymbol{B}\boldsymbol{u}^c[k] + \boldsymbol{B}\boldsymbol{e}_a[k]) \\ &= (\boldsymbol{I} - \boldsymbol{M}\boldsymbol{C})(\boldsymbol{A}\boldsymbol{x}[k] + \boldsymbol{B}\boldsymbol{u}^c[k]) + (\boldsymbol{I} - \boldsymbol{M}\boldsymbol{C})\boldsymbol{B}\boldsymbol{e}_a[k]\end{aligned}$$

Open-Loop UIO

- We have

$$\boldsymbol{z}[k+1] = (\boldsymbol{I} - \boldsymbol{MC})(\boldsymbol{Ax}[k] + \boldsymbol{Bu}^c[k]) + (\boldsymbol{I} - \boldsymbol{MC})\boldsymbol{Be}_a[k]$$

- Select $\boldsymbol{M}$ such that $(\boldsymbol{I} - \boldsymbol{MC})\boldsymbol{B} = \boldsymbol{O}$
- Then, $\boldsymbol{z}[k+1] = (\boldsymbol{I} - \boldsymbol{MC})(\boldsymbol{Ax}[k] + \boldsymbol{Bu}^c[k])$
- Substituting $\boldsymbol{x}[k] = \boldsymbol{z}[k] + \boldsymbol{M}\tilde{\boldsymbol{y}}^c[k]$ gives

$$\boldsymbol{z}[k+1] = (\boldsymbol{I} - \boldsymbol{MC})(\boldsymbol{Az}[k] + \boldsymbol{AM}\tilde{\boldsymbol{y}}^c[k] + \boldsymbol{Bu}^c[k])$$

- State estimate: $\hat{\boldsymbol{x}}[k] = \boldsymbol{z}[k] + \boldsymbol{M}\tilde{\boldsymbol{y}}^c[k]$
- State observation error:

$$\boldsymbol{e}[k+1] = \boldsymbol{x}[k+1] - \hat{\boldsymbol{x}}[k+1] = (\boldsymbol{I} - \boldsymbol{MC})\boldsymbol{Ae}[k]$$

Open-Loop Observer Error Analysis

- UIO observer

$$\begin{aligned} \boldsymbol{z}[k+1] &= (\boldsymbol{I}-\boldsymbol{MC})(\boldsymbol{Az}[k]+\boldsymbol{AM}\tilde{\boldsymbol{y}}^c[k]+\boldsymbol{Bu}^c[k]) \\ \hat{\boldsymbol{x}}[k] &= \boldsymbol{z}[k]+\boldsymbol{M}\tilde{\boldsymbol{y}}^c[k] \end{aligned}$$

- Observation error dynamics

$$\begin{aligned} \boldsymbol{e}[k+1] &= \boldsymbol{x}[k+1]-\hat{\boldsymbol{x}}[k+1] \\ &= \boldsymbol{Ax}[k]+\boldsymbol{B}(\boldsymbol{u}^c[k]+\boldsymbol{e}_a[k])-\boldsymbol{z}[k+1] \\ &\quad -\boldsymbol{M}\tilde{\boldsymbol{y}}^c[k+1] \\ &= \boldsymbol{Ax}[k]+\boldsymbol{B}(\boldsymbol{u}^c[k]+\boldsymbol{e}_a[k])-\boldsymbol{z}[k+1] \\ &\quad -\boldsymbol{MCx}[k+1] \\ &= \boldsymbol{Ax}[k]+\boldsymbol{B}(\boldsymbol{u}^c[k]+\boldsymbol{e}_a[k])-\boldsymbol{z}[k+1] \\ &\quad -\boldsymbol{MCAx}[k]-\boldsymbol{MCBu}^c[k]-\boldsymbol{MCBe}_a[k]) \end{aligned}$$

Open-Loop Observation Error

- Select $\boldsymbol{M}$ so that $(\boldsymbol{I} - \boldsymbol{MC})\boldsymbol{B} = \boldsymbol{O}$ gives

$$\begin{aligned}
\boldsymbol{e}[k+1] &= \boldsymbol{A}\boldsymbol{x}[k] + \boldsymbol{B}(\boldsymbol{u}^c[k] + \boldsymbol{e}_a[k]) - \boldsymbol{z}[k+1] \\
&\quad -\boldsymbol{MCA}\boldsymbol{x}[k] - \boldsymbol{MCB}\boldsymbol{u}^c[k] - \boldsymbol{MCB}\boldsymbol{e}_a[k]) \\
&= (\boldsymbol{I} - \boldsymbol{MC})(\boldsymbol{A}\boldsymbol{x}[k] + \boldsymbol{B}\boldsymbol{u}^c[k]) \\
&\quad +(\boldsymbol{I} - \boldsymbol{MC})\boldsymbol{B}\boldsymbol{e}_a[k] \\
&\quad -\boldsymbol{z}[k+1] \\
&= (\boldsymbol{I} - \boldsymbol{MC})(\boldsymbol{A}\boldsymbol{x}[k] + \boldsymbol{B}\boldsymbol{u}^c[k]) \\
&\quad -(\boldsymbol{I} - \boldsymbol{MC})(\boldsymbol{A}\hat{\boldsymbol{x}}[k] + \boldsymbol{B}\boldsymbol{u}^c[k]) \\
&= (\boldsymbol{I} - \boldsymbol{MC})\boldsymbol{A}\boldsymbol{e}[k]
\end{aligned}$$

- Open-loop UIO impractical—no control of the observation error dynamics

Closed-Loop UIO Error Dynamics

- Close the loop by adding the term $\boldsymbol{L}(\tilde{\boldsymbol{y}}^c[k] - \hat{\boldsymbol{y}})$ to the UIO, where $\hat{\boldsymbol{y}} = \boldsymbol{C}\hat{\boldsymbol{x}}$ is the UIO output
- Closed-loop UIO observation error

$$\begin{aligned} \boldsymbol{e}[k+1] &= (\boldsymbol{I} - \boldsymbol{MC})\boldsymbol{A}\boldsymbol{e}[k] - \boldsymbol{L}(\tilde{\boldsymbol{y}}^c[k] - \hat{\boldsymbol{y}}) \\ &= (\boldsymbol{I} - \boldsymbol{MC})\boldsymbol{A}\boldsymbol{e}[k] - \boldsymbol{L}(\boldsymbol{C}\boldsymbol{x}[k] - \boldsymbol{C}\hat{\boldsymbol{x}}) \\ &= (\boldsymbol{I} - \boldsymbol{MC})\boldsymbol{A}\boldsymbol{e}[k] - \boldsymbol{LC}\boldsymbol{e}[k] \end{aligned}$$

- Let $\boldsymbol{A}_1 = (\boldsymbol{I} - \boldsymbol{MC})\boldsymbol{A}$, then we have

$$\boldsymbol{e}[k+1] = (\boldsymbol{A}_1 - \boldsymbol{LC})\boldsymbol{e}[k]$$

- Closed-loop UIO—use the observer gain $\boldsymbol{L}$ to control the observation error dynamics

Closed-Loop UIO

- Compute $\boldsymbol{M}$ such that

$$(\boldsymbol{I} - \boldsymbol{MC})\boldsymbol{B} = \boldsymbol{O}$$

- The UIO

$$\begin{aligned} \boldsymbol{z}[k+1] &= (\boldsymbol{I} - \boldsymbol{MC})\left(\boldsymbol{Az}[k] + \boldsymbol{AM}\tilde{\boldsymbol{y}}^c[k] + \boldsymbol{Bu}^c[k]\right) \\ &\quad + \boldsymbol{L}(\tilde{\boldsymbol{y}}^c[k] - \hat{\boldsymbol{y}}[k]) \\ \hat{\boldsymbol{x}}[k] &= \boldsymbol{z}[k] + \boldsymbol{M}\tilde{\boldsymbol{y}}^c[k] \end{aligned}$$

  where

$$\hat{\boldsymbol{y}}[k] = \boldsymbol{C}\hat{\boldsymbol{x}}[k]$$

Solving $(\boldsymbol{I} - \boldsymbol{MC})\boldsymbol{B} = \boldsymbol{O}$ for $\boldsymbol{M}$

- $(\boldsymbol{I} - \boldsymbol{MC})\boldsymbol{B} = \boldsymbol{O} \iff \boldsymbol{MCB} = \boldsymbol{B}$
- $\boldsymbol{MCB} = \boldsymbol{B}$ implies

$$\mathrm{rank}(\boldsymbol{MCB}) = \mathrm{rank}\boldsymbol{B}$$

- On the other hand,

$$\mathrm{rank}(\boldsymbol{MCB}) \leq \mathrm{rank}(\boldsymbol{CB}) \leq \mathrm{rank}(\boldsymbol{B})$$

- Hence, a necessary and sufficient condition for solvability of $(\boldsymbol{I} - \boldsymbol{MC})\boldsymbol{B} = \boldsymbol{O}$ is

$$\mathrm{rank}(\boldsymbol{CB}) = \mathrm{rank}(\boldsymbol{B})$$

Summary of UIO Construction

Theorem

Let $\boldsymbol{A}_1 = (\boldsymbol{I} - \boldsymbol{MC})\boldsymbol{A}$ *and* $\boldsymbol{T} = \boldsymbol{PL}$. *If*

1. $(\boldsymbol{I} - \boldsymbol{MC})\boldsymbol{B} = \boldsymbol{O}$,
2. *there exists* $\boldsymbol{P} = \boldsymbol{P}^\top \succ 0$ *such that*

$$\begin{bmatrix} -\boldsymbol{P} & \boldsymbol{A}_1^\top \boldsymbol{P} - \boldsymbol{C}^\top \boldsymbol{T}^\top \\ \boldsymbol{P}\boldsymbol{A}_1 - \boldsymbol{T}\boldsymbol{C} & -\boldsymbol{P} \end{bmatrix} \prec 0,$$

then the UIO exists

Theorem Discussion

- $(\boldsymbol{A}_1 - \boldsymbol{LC})$ Schur stable $\iff$ there exists $\boldsymbol{P} = \boldsymbol{P}^\top \succ 0$ such that

$$(\boldsymbol{A}_1 - \boldsymbol{LC})^\top \boldsymbol{P}(\boldsymbol{A}_1 - \boldsymbol{LC}) - \boldsymbol{P} \prec 0$$

- Substitute $\boldsymbol{P} = \boldsymbol{P}\boldsymbol{P}^{-1}\boldsymbol{P}$ into the Lyapunov matrix inequality to obtain

$$(\boldsymbol{A}_1 - \boldsymbol{LC})^\top \boldsymbol{P}\boldsymbol{P}^{-1}\boldsymbol{P}(\boldsymbol{A}_1 - \boldsymbol{LC}) - \boldsymbol{P} \prec 0$$

  which is equivalent to the LMI condition of the Theorem by taking the Schur complement

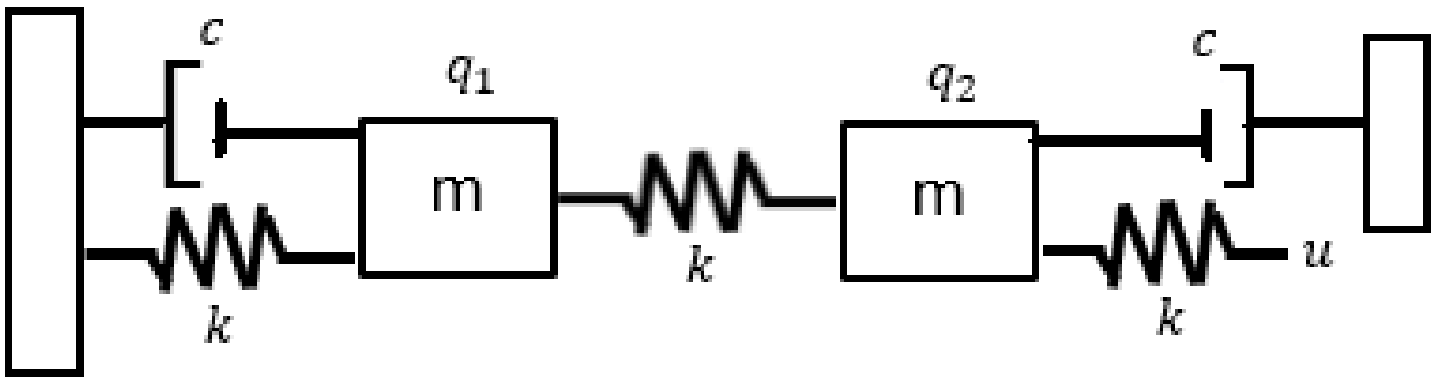
Combined Estimator-UIO Design

1. Design an estimator $\tilde{\boldsymbol{e}}_s[k]$ of $\boldsymbol{e}_s[k]$
2. Design the UIO performing the following steps:
   - Check if $\text{rank}(\boldsymbol{CB}) = \text{rank}(\boldsymbol{B})$ is satisfied. If the condition is not satisfied, STOP
   - Solve $(\boldsymbol{I} - \boldsymbol{MC})\boldsymbol{B} = \boldsymbol{O}$ to obtain

     $$\boldsymbol{M} = \boldsymbol{B}\left((\boldsymbol{CB})^{\dagger} + \boldsymbol{H}_0(\boldsymbol{I}_p - (\boldsymbol{CB})(\boldsymbol{CB})^{\dagger})\right),$$

     where the superscript $\dagger$ denotes the Moore-Penrose pseudo-inverse and $\boldsymbol{H}_0$ is a design parameter matrix
   - Solve for $\boldsymbol{P}$ and $\boldsymbol{T}$
   - If $\boldsymbol{P} = \boldsymbol{P}^{\top} \succ 0$, UIO exists
   - Compute $\boldsymbol{L}_1 = \boldsymbol{P}^{-1}\boldsymbol{T}$

Numerical Example



Coupled mass-spring-damper system

Modeling

- State-space model:

$$\boldsymbol{x} = \begin{bmatrix} q_1 \\ q_2 \\ \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}, \ \boldsymbol{A}_c = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{-2k}{m} & \frac{k}{m} & \frac{-c}{m} & 0 \\ \frac{k}{m} & \frac{-2k}{m} & 0 & \frac{-c}{m} \end{bmatrix}, \ \boldsymbol{B}_c = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{k}{m} \end{bmatrix},$$

$$\boldsymbol{C}_c = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

- Parameters: $m = 5$ kg, $k = 5$ N/m, and $c = 10$ N·s/m
- Discretize: sampling time $T_s = 0.1$ s

Discrete-Time (DT) Model of the Coupled Mass-Spring-damper System

$$\boldsymbol{A} = \begin{bmatrix} 0.9907 & 0.0047 & 0.0903 & 0.0002 \\ 0.0047 & 0.9907 & 0.0002 & 0.0903 \\ -0.1805 & 0.0900 & 0.8100 & 0.0044 \\ 0.0900 & -0.1805 & 0.0044 & 0.8100 \end{bmatrix},$$

$$\boldsymbol{B} = \begin{bmatrix} 0 \\ 0.0047 \\ 0.0002 \\ 0.0903 \end{bmatrix}, \boldsymbol{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Computing the $\boldsymbol{M}$ and $\boldsymbol{L}$ Matrices of the UIO

$$\boldsymbol{M} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0.0033 \\ 0.0017 & 199.66 \end{bmatrix}, \quad \boldsymbol{L} = \begin{bmatrix} 0.9936 & 0.0039 \\ -0.0008 & -0.0000 \\ -0.1551 & 0.0581 \\ -0.0159 & -19.3226 \end{bmatrix}$$

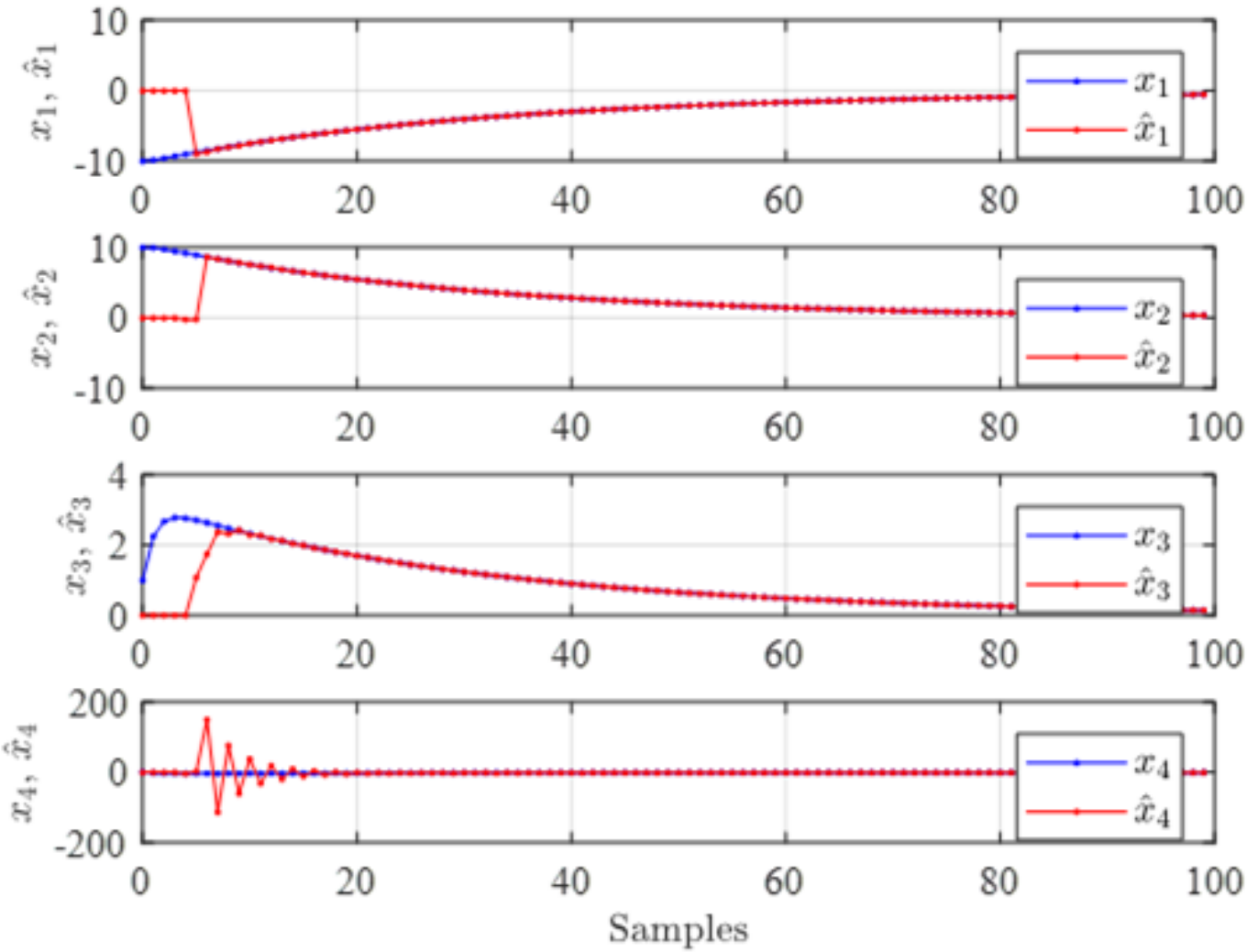
Simulation Parameters

- Number of observations $\tau = 10$ selected so that the condition of the Lemma is satisfied, that is,

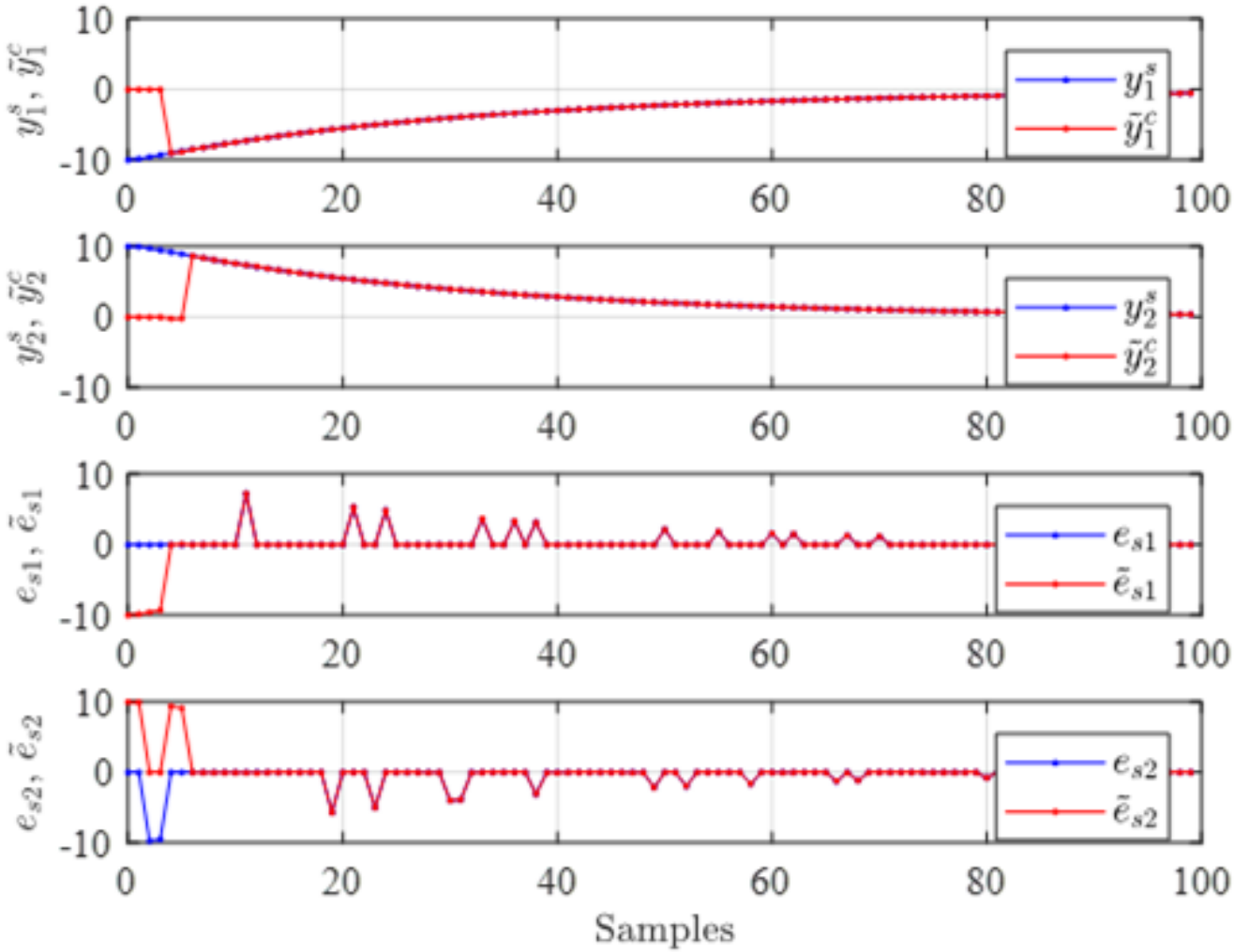
$$(\tau p - n) \geq 2(i_s + i_a)$$

- Controller, $\boldsymbol{u}^c[k] = -\boldsymbol{K}_d \hat{\boldsymbol{x}}[k]$, where $\boldsymbol{K}_d = \begin{bmatrix} 0.1381 & 0.2677 & 0.0651 & 0.3401 \end{bmatrix}$ is the feedback gain calculated using discrete-time LQR
- Zero initial conditions on the plant input and its output, that is, $\boldsymbol{u}^c[-1] = \cdots = \boldsymbol{u}^c[1-\tau] = 0$ and $\boldsymbol{y}^c[-1] = \cdots = \boldsymbol{y}^c[1-\tau] = \mathbf{0}$

State estimates with 15% output transmission packet drops and 5% input transmission packet drops



Output recovery and output recovery errors with 15% output transmission packet drops and 5% input transmission packet drops



Recap

Recap: Networked Control System (NCS) Security

- Networked Control Systems depend on wireless communication—a major challenge in the NCS design is their security
- Actuators and sensor measurements exposed to malicious attacks in communication networks
- Methods of detecting sparse malicious packet drop attacks in the communication networks proposed
- Limitations of the proposed methods—malicious attacks assumed to be sparse