Safety Analysis of Embedded Controllers Under Implementation Platform Timing Uncertainties

Sumit Gupta[1] Munna Kumar[2] Raushan Kumar[3] Saurav Kumar[4]

[1]Department of Physics, 200394

[2]Department of Electrical Engineering, 200608

[3]Department of Material Science and Engineering, 210830

[4]Department of Civil Engineering, 210950

EMBEDDED AND CYBER-PHYSICAL SYSTEMS

Introduction

- As embedded systems architectures become more complex and distributed, checking the safety of feedback control loops implemented on them becomes a crucial problem for emerging autonomous systems.

Introduction

- As embedded systems architectures become more complex and distributed, checking the safety of feedback control loops implemented on them becomes a crucial problem for emerging autonomous systems.
- In this presentation, we will be analyzing system behavior having timing uncertainties.

Introduction

- As embedded systems architectures become more complex and distributed, checking the safety of feedback control loops implemented on them becomes a crucial problem for emerging autonomous systems.
- In this presentation, we will be analyzing system behavior having timing uncertainties.
- There are different strategies for handling deadline misses or system overruns, all leading to a stable system.

Introduction

- What is deadline miss?

Introduction

- What is deadline miss?
  - Deadline miss in an embedded system occurs when a task does not complete its execution within a specific deadline.

Introduction

- What is deadline miss?
  - Deadline miss in an embedded system occurs when a task does not complete its execution within a specific deadline.
- Reason?

Introduction

- What is deadline miss?
  - Deadline miss in an embedded system occurs when a task does not complete its execution within a specific deadline.
- Reason?
  - Unexpected Events such as Hardware interruption, Network traffic

Introduction

- What is deadline miss?
  - Deadline miss in an embedded system occurs when a task does not complete its execution within a specific deadline.
- Reason?
  - Unexpected Events such as Hardware interruption, Network traffic
  - Incorrect task scheduling

Introduction

- What is deadline miss?
  - Deadline miss in an embedded system occurs when a task does not complete its execution within a specific deadline.
- Reason?
  - Unexpected Events such as Hardware interruption, Network traffic
  - Incorrect task scheduling
  - Resources used by others

Plant Model

- Now, we analyze linear time-invariant models and controllers.

Plant Model

- Now, we analyze linear time-invariant models and controllers.

$$\dot{x}(t) = A_c x(t) + B_c u(t)$$

Plant Model

- Now, we analyze linear time-invariant models and controllers.

$$\dot{x}(t) = A_c x(t) + B_c u(t)$$

$$y(t) = C_c x(t) + D_c u(t)$$

Plant Model

- Now, we analyze linear time-invariant models and controllers.

$$\dot{x}(t) = A_c x(t) + B_c u(t)$$

$$y(t) = C_c x(t) + D_c u(t)$$

$$x(t) = [x_1(t), x_2(t), \cdots, x_p(t)]$$

Plant Model

- Now, we analyze linear time-invariant models and controllers.

$$\dot{x}(t) = A_c x(t) + B_c u(t)$$

$$y(t) = C_c x(t) + D_c u(t)$$

$$x(t) = [x_1(t), x_2(t), \cdots, x_p(t)]$$

$$u(t) = [u_1(t), u_2(t), \cdots, u_r(t)]$$

Plant Model

- Now, we analyze linear time-invariant models and controllers.

$$\dot{x}(t) = A_c x(t) + B_c u(t)$$

$$y(t) = C_c x(t) + D_c u(t)$$

$$x(t) = [x_1(t), x_2(t), \cdots, x_p(t)]$$

$$u(t) = [u_1(t), u_2(t), \cdots, u_r(t)]$$

- A_c, B_c, C_c, D_c encodes the dynamics of the system.

Key Assumptions

- D_c is zero matrix

Key Assumptions

- D_c is zero matrix
- This means that the system is strictly proper.

Key Assumptions

- D_c is zero matrix
- This means that the system is strictly proper.
- C_c is the unit matrix of appropriate size. This means that state is measurable.

Key Assumptions

- D_c is zero matrix
- This means that the system is strictly proper.
- C_c is the unit matrix of appropriate size. This means that state is measurable.
- Now, we can represent our model as

$$\dot{x}(t) = A_c x(t) + B_c u(t)$$

and describe the system dynamics using only A_c and B_c

Discretizing

- After discretizing our model, the system becomes

$$x_{[k+1]} = A_d x_{[k]} + B_d u_{[k]}$$

Discretizing

- After discretizing our model, the system becomes

$$x_{[k+1]} = A_d x_{[k]} + B_d u_{[k]}$$

- K is sampling instant that means there is a distance of $\pi[s]$ between kth and (k+1)th instant.

Discretizing

- After discretizing our model, the system becomes

$$x_{[k+1]} = A_d x_{[k]} + B_d u_{[k]}$$

- K is sampling instant that means there is a distance of $\pi[s]$ between kth and (k+1)th instant.
- A_d and B_d are counterpart of A_c and B_c for continuous system.

Controller Model

- As we know from our helicopter model example, the current control input is proportional to the previous state of the system.

Controller Model

- As we know from our helicopter model example, the current control input is proportional to the previous state of the system.
- $$u_{[k]} = Kx_{[k-1]}$$

Feedback interconnection

- Using $u_{[k]}$ from the previous equation,

$$x_{[k+1]} = A_d x_{[k]} + B_d K x_{[k-1]}$$

Feedback interconnection

- Using $u_{[k]}$ from the previous equation,

$$x_{[k+1]} = A_d x_{[k]} + B_d K x_{[k-1]}$$

- To analyze the closed-loop system, we define the current state as

$$\tilde{x}_{[k]} = [x_{[k]}, x_{[k-1]}]^T$$

Feedback interconnection

- Using $u_{[k]}$ from the previous equation,

$$x_{[k+1]} = A_d x_{[k]} + B_d K x_{[k-1]}$$

- To analyze the closed-loop system, we define the current state as

$$\tilde{x}_{[k]} = [x_{[k]}, x_{[k-1]}]^T$$

$$\tilde{x}_{[k+1]} = \begin{bmatrix} x_{[k+1]} \\ x_{[k]} \end{bmatrix} = \begin{bmatrix} A_d & B_d K \\ I_p & 0_{p\times p} \end{bmatrix} \begin{bmatrix} x_{[k]} \\ x_{[k-1]} \end{bmatrix}$$

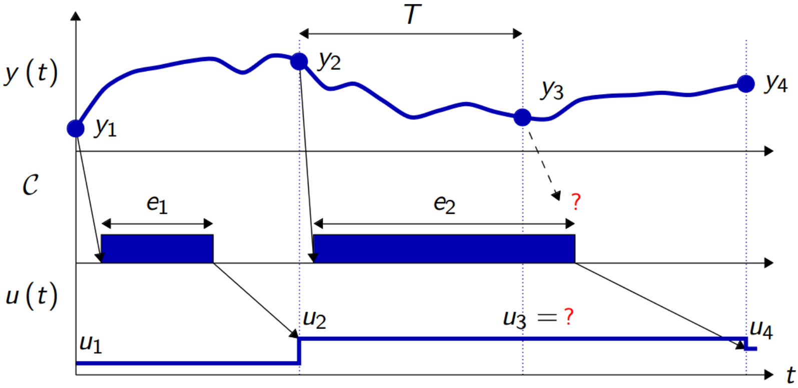
$$\tilde{x}_{[k+1]} = A\tilde{x}_{[k]}$$

Definitions

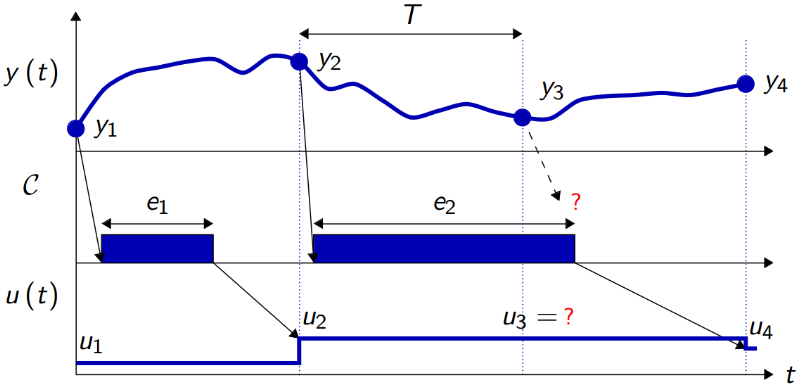
$$\tau \vdash \overline{\langle n \rangle}$$

This means that the maximum deadline misses the system can have is n.

Deadline Missed



Deadline Missed



- Now what?

Deadline Missed

- Control Input ($u_{[k]}$) = ?

Deadline Missed

- Control Input ($u_{[k]}$) = ?
- Missed Task = ?

Handling Control Input

- Zero: $u_{[k+1]} = 0$

Handling Control Input

- Zero: $u_{[k+1]} = 0$
- Hold: $u_{[k+1]} = u_{[k]}$

Handling Task

- Kill: Kill the task like nothing happened.

Handling Task

- Kill: Kill the task like nothing happened.
- skip-next: Allow the task to continue and shift the next task to the next deadline.

Not Preffered

- Killing a task is bad.

Not Preffered

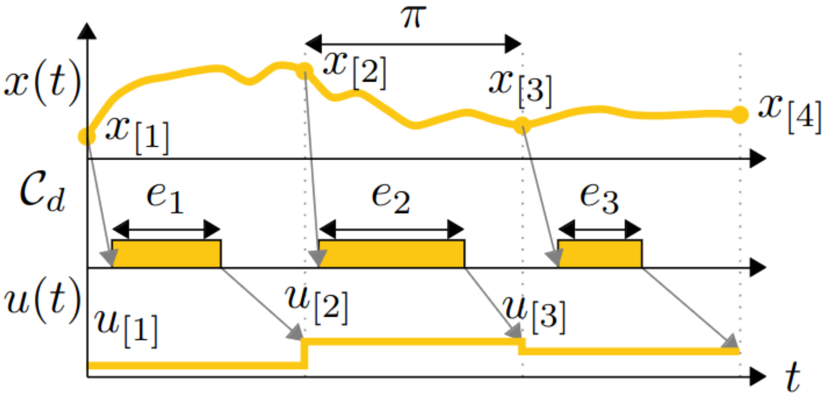
- Killing a task is bad.
  - Might have changed the internal state of the system.

Example

- All deadlines are met

Example

- All deadlines are met

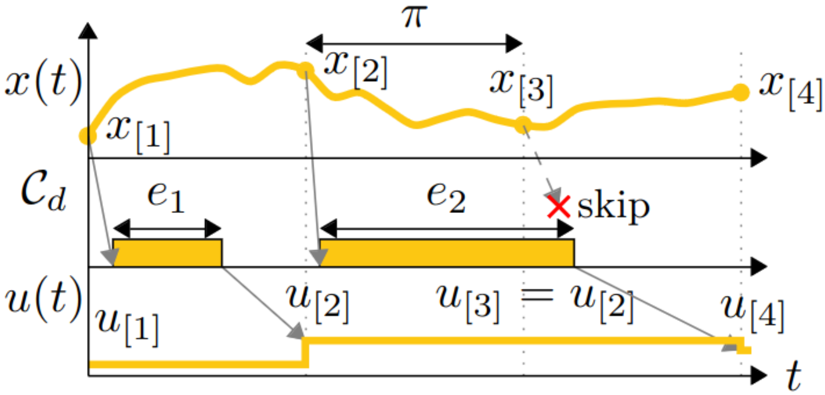


Example

- All deadlines are not met

Example

- All deadlines are not met
- Hold & Skip next



Stability in absence of deadline misses

$$x_{[k+1]} = Ax_{[k]}$$

Stability in absence of deadline misses

$$x_{[k+1]} = Ax_{[k]}$$

$$\Downarrow$$

$$x_{[k+1]} = A^{k+1}x_{[0]}$$

Stability in absence of deadline misses

$$x_{[k+1]} = Ax_{[k]}$$

$$\Downarrow$$

$$x_{[k+1]} = A^{k+1}x_{[0]}$$

- Linear Algebra says that the system is stable if all the eigenvalues of A have norm <1

Spectral Radius

- Spectral Radius:

$$\rho(A) = \max\{|\lambda_1|, \cdots, |\lambda_n|\}$$

Spectral Radius

- Spectral Radius:

$$\rho(A) = \max\{|\lambda_1|, \cdots, |\lambda_n|\}$$

- Another way of saying the system is stable is

$$\rho(A) < 1$$

Stability in presence of deadline misses

- The case is: 1001110.....1

Stability in presence of deadline misses

- The case is: 1001110.....1
- 1 represents a deadline hit, and 0 represents a deadline miss

Stability in presence of deadline misses

- The case is: 1001110.....1
- 1 represents a deadline hit, and 0 represents a deadline miss
- Say the transition matrix A during deadline hit is A_H and during deadline miss is A_M.

Stability in presence of deadline misses

- The case is: 1001110.....1
- 1 represents a deadline hit, and 0 represents a deadline miss
- Say the transition matrix A during deadline hit is A_H and during deadline miss is A_M.
-

$$x_{[1]} = \qquad A_H x_{[0]}$$

Stability in presence of deadline misses

- The case is: 1001110.....1
- 1 represents a deadline hit, and 0 represents a deadline miss
- Say the transition matrix A during deadline hit is A_H and during deadline miss is A_M.
- $$x_{[1]} = A_H x_{[0]}$$
- $$x_{[2]} = A_M A_H x_{[0]}$$

Stability in presence of deadline misses

- The case is: 1001110.....1
- 1 represents a deadline hit, and 0 represents a deadline miss
- Say the transition matrix A during deadline hit is A_H and during deadline miss is A_M.
- $$x_{[1]} = A_H x_{[0]}$$
- $$x_{[2]} = A_M A_H x_{[0]}$$
- $$x_{[k+1]} = A_H \cdots A_M A_M A_H x_{[0]}$$

Stability in presence of deadline misses

- The case is: 1001110.....1
- 1 represents a deadline hit, and 0 represents a deadline miss
- Say the transition matrix A during deadline hit is A_H and during deadline miss is A_M.
- $$x_{[1]} = A_H x_{[0]}$$
- $$x_{[2]} = A_M A_H x_{[0]}$$
- $$x_{[k+1]} = A_H \cdots A_M A_M A_H x_{[0]}$$
- Again system is stable if
$$\rho(A_H \cdots A_M A_M A_H) < 1$$

Generalization

- The possible realisations of the system $\tau \vdash \overline{\langle n \rangle}$ can include

$$\{H, MH, \cdots, M^n H\}$$

Generalization

- The possible realisations of the system $\tau \vdash \overline{\langle n \rangle}$ can include

$$\{H, MH, \cdots, M^n H\}$$

- In terms of matrix

$$\Sigma = \{A_H, A_H A_M, A_H {A_M}^2 \cdots, A_H {A_M}^n\}$$

Generalization

- The possible realisations of the system $\tau \vdash \overline{\langle n \rangle}$ can include

$$\{H, MH, \cdots, M^n H\}$$

- In terms of matrix

$$\Sigma = \{A_H, A_H A_M, A_H {A_M}^2 \cdots, A_H {A_M}^n\}$$

- For stability

$$\rho(\Sigma) < 1$$

Further Generalization

- We can also have cases like

$$H\,\underbrace{MMM\cdots}_{\text{n-2 times}}\,H\cdots\underbrace{MMMMM....M}_{\text{n times}}\,H$$

Further Generalization

- We can also have cases like

$$H\,\underbrace{MMM\cdots}_{\text{n-2 times}}H\cdots\underbrace{MMMMM....M}_{\text{n times}}H$$

- We can have infinite such cases. How to compute Spectral radius now?

Further Generalization

- We can also have cases like

$$H\,\underbrace{MMM\cdots}_{\text{n-2 times}}\,H\cdots\underbrace{MMMMM....M}_{\text{n times}}\,H$$

- We can have infinite such cases. How to compute Spectral radius now?
- Well, computing the exact value is not possible. We use approximation methods.

Further Generalization

- We can also have cases like

$$H \underbrace{MMM\cdots}_{\text{n-2 times}} H \cdots \underbrace{MMMMM....M}_{\text{n times}} H$$

- We can have infinite such cases. How to compute Spectral radius now?
- Well, computing the exact value is not possible. We use approximation methods.
- The approximation method give a lower and upper bound of the spectral radius, and based on them, we comment on stability.

Further Generalization

- We can also have cases like

$$H \underbrace{MMM\cdots}_{\text{n-2 times}} H \cdots \underbrace{MMMMM....M}_{\text{n times}} H$$

- We can have infinite such cases. How to compute Spectral radius now?
- Well, computing the exact value is not possible. We use approximation methods.
- The approximation method give a lower and upper bound of the spectral radius, and based on them, we comment on stability.
- Used MATLAB JSR toolbox in our implementation.

Mathematical Formulation of A_H and A_M.

Zero&Kill

Hit

$$x_{[k+1]} = A_d\, x_{[k]} + B_d\, u_{[k]}.$$

$$u_{[k+1]} = K\, x_{[k]}$$

$$\begin{bmatrix} x_{[k+1]} \\ u_{[k+1]} \end{bmatrix} = \underbrace{\begin{bmatrix} A_d & B_d \\ K & 0_{r\times r} \end{bmatrix}}_{A_H} \begin{bmatrix} x_{[k]} \\ u_{[k]} \end{bmatrix}$$

Miss

$$x_{[k+1]} = A_d\, x_{[k]} + B_d\, u_{[k]}.$$

$$u_{[k+1]} = 0$$

$$\begin{bmatrix} x_{[k+1]} \\ u_{[k+1]} \end{bmatrix} = \underbrace{\begin{bmatrix} A_d & B_d \\ 0_{r\times r} & 0_{r\times r} \end{bmatrix}}_{A_M} \begin{bmatrix} x_{[k]} \\ u_{[k]} \end{bmatrix}$$

When the controller hits the deadline. Kill implies that in case of a deadline miss there is an abort of what the control task has been computing up to its deadline, which means there is no need to take into account its (partial) behaviour. In case of a deadline miss in the k-th iteration, the control signal u[k+1] is set to zero. With n maximum consecutive misses, we can then compute all the matrices in Σ = {$A_H A_M^i$ i ∈ Z ≥, i ≤ n} 8 and then compute the upper bound on ρ(Σ).

Zero&Skip-Next

The difference between Zero&Kill and Zero&Skip-Next lays in the freshness of the measurements that are used for the computation of the control signal when the task τ hits its deadline.

HIT

$$\begin{bmatrix} x_{[k+1]} \\ x_{[k]} \\ \dots \\ x_{[k-n+1]} \\ u_{[k+1]} \end{bmatrix} = \underbrace{\begin{bmatrix} A_d & 0_{p\times(n\cdot p)} & B_d \\ & I_{n\cdot p} & 0_{(n\cdot p)\times(p+r)} \\ K & 0_{r\times(n\cdot p)} & 0_{r\times r} \end{bmatrix}}_{A_H} \begin{bmatrix} x_{[k]} \\ x_{[k-1]} \\ \dots \\ x_{[k-n]} \\ u_{[k]} \end{bmatrix}.$$

In fact, when the deadline is hit (not after a miss) there is no use of the previous values of the state.

Zero&Skip-Next

When a miss occurs, the control signal is set to zero. Substitute the value of K with a zero matrix of appropriate size, i.e., 0r×p to obtain the Am matrix.

$$\begin{bmatrix} x_{[k+1]} \\ x_{[k]} \\ \cdots \\ x_{[k-n+1]} \\ u_{[k+1]} \end{bmatrix} = \underbrace{\begin{bmatrix} A_d & 0_{p\times(n\cdot p)} & B_d \\ & I_{n\cdot p} & 0_{(n\cdot p)\times(p+r)} \\ & 0_{r\times(n+1)\cdot p+r} & \end{bmatrix}}_{A_M} \begin{bmatrix} x_{[k]} \\ x_{[k-1]} \\ \cdots \\ x_{[k-n]} \\ u_{[k]} \end{bmatrix}.$$

a hit that follows a certain number of misses (up to n) has a different matrix with respect to AH. We denote with A_{R_i} the matrix that represents the evolution of the closed-loop system when a recovery happens after i deadlines were missed.

Zero&Skip-Next

For i = 1, i.e., with one deadline miss, we can write

$$\begin{bmatrix} x_{[k+1]} \\ x_{[k]} \\ \cdots \\ x_{[k-n+1]} \\ u_{[k+1]} \end{bmatrix} = \underbrace{\begin{bmatrix} A_d & 0_{p\times(n\cdot p)} & & B_d \\ & I_{n\cdot p} & & 0_{(n\cdot p)\times(p+r)} \\ 0_{r\times p} & K & 0_{r\times(n-1)\cdot p} & 0_{r\times r} \end{bmatrix}}_{A_{R_1}} \begin{bmatrix} x_{[k]} \\ x_{[k-1]} \\ \cdots \\ x_{[k-n]} \\ u_{[k]} \end{bmatrix}.$$

Consistently with our treatise, $A_{R_0} = A_H$. We can then compute the set Σ as

Σ = { $A_{R_i} A_M^i$ | i ∈ Z ≥, i ≤ n} and then compute the upper bound on ρ(Σ) using the computed set of matrices.

Zero&Queue(1)

The behaviour of the combination of the zero policy and the queue(1) strategy vary depending on the possibility of the queued job to complete before the deadline or not.

We can start from the set Σ used for the Zero&Skip-Next combination and add to the set all the matrices $A_H A_M^i$, that take into account the possibility that the queued job completed before its deadline. We should also include in the set Σ the matrices A_{R_i} alone, as it could happen that a queued job doesn't terminate in the period it was started in. We then obtain

Σ = { $A_H A_M^i, A_{R_i}, A_{R_i} A_M^i$ | i ∈ Z ≥, i ≤ n}, and we can use the set to compute the upper bound on the joint spectral radius ρ(Σ).

Hold&Kill.

The hold and kill strategy aborts the task but applies the previously computed control signal to the plant. We can write the evolution of the system when a deadline is hit as

$$\begin{bmatrix} x_{[k+1]} \\ u_{[k+1]} \end{bmatrix} = \underbrace{\begin{bmatrix} A_d & B_d \\ K & 0_{r\times r} \end{bmatrix}}_{A_H} \begin{bmatrix} x_{[k]} \\ u_{[k]} \end{bmatrix},$$

When a deadline is missed, we compute the system evolution as

$$\begin{bmatrix} x_{[k+1]} \\ u_{[k+1]} \end{bmatrix} = \underbrace{\begin{bmatrix} A_d & B_d \\ 0_{r\times r} & I_r \end{bmatrix}}_{A_M} \begin{bmatrix} x_{[k]} \\ u_{[k]} \end{bmatrix},$$

if we assume there can be a maximum of n consecutive deadline misses, we can then compute all the matrices in Σ = { $A_H A_M^i$ | i ∈ Z ≥, i ≤ n} and then compute the upper bound on ρ(Σ).

Hold&Skip-Next

In order to analyse the combination of hold and skip-next we need to augment the state vector as we did for the zero&skip-next handling strategy. We obtain the following expression for the closed-loop system when we hit a deadline,

$$\begin{bmatrix} x_{[k+1]} \\ x_{[k]} \\ \cdots \\ x_{[k-n+1]} \\ u_{[k+1]} \end{bmatrix} = \underbrace{\begin{bmatrix} A_d & 0_{p\times(n\cdot p)} & B_d \\ & I_{n\cdot p} & 0_{(n\cdot p)\times(p+r)} \\ K & 0_{r\times(n\cdot p)} & 0_{r\times r} \end{bmatrix}}_{A_H} \begin{bmatrix} x_{[k]} \\ x_{[k-1]} \\ \cdots \\ x_{[k-n]} \\ u_{[k]} \end{bmatrix}.$$

Hold&Skip-Next

When we miss a deadline we use the old control value

$$\begin{bmatrix} x_{[k+1]} \\ x_{[k]} \\ \dots \\ x_{[k-n+1]} \\ u_{[k+1]} \end{bmatrix} = \underbrace{\begin{bmatrix} A_d & 0_{p\times(n\cdot p)} & B_d \\ & I_{n\cdot p} & 0_{(n\cdot p)\times(p+r)} \\ 0_{r\times(n+1)\cdot p} & & I_r \end{bmatrix}}_{A_M} \begin{bmatrix} x_{[k]} \\ x_{[k-1]} \\ \dots \\ x_{[k-n]} \\ u_{[k]} \end{bmatrix}.$$

For one deadline miss we obtain AR1 as

$$\begin{bmatrix} x_{[k+1]} \\ x_{[k]} \\ \dots \\ x_{[k-n+1]} \\ u_{[k+1]} \end{bmatrix} = \underbrace{\begin{bmatrix} A_d & & 0_{p\times(n\cdot p)} & B_d \\ & & I_{n\cdot p} & 0_{(n\cdot p)\times(p+r)} \\ 0_{r\times p} & K & 0_{r\times(n-1)\cdot p} & 0_{r\times r} \end{bmatrix}}_{A_{R_1}} \begin{bmatrix} x_{[k]} \\ x_{[k-1]} \\ \dots \\ x_{[k-n]} \\ u_{[k]} \end{bmatrix},$$

Again, $A_{R_0} = A_H$ and we can define the set Σ as Σ = { $A_{R_i} A_M^i$ | i ∈ Z ≥, i ≤ n}. With this, we can compute the upper bound on ρ(Σ).

Hold&Queue(1).

To analyse the hold&queue(1) strategy, we follow the same principles used for the zero&queue(1) strategy. We start from the hold&skip-next matrices and determine

Σ = { $A_H A_M^i$, A_{R_i}, $A_{R_i} A_M^i$ | i ∈ Z ≥, i ≤ n}.

Experimental Validation

- A few examples of how the analysis.
- Some results obtained with an unstable second-order system.
- Verify the stability of a permanent magnet synchronous motor for an automotive electric steering application.

Unstable Second-Order System

$$\dot{x}(t) = \underbrace{\begin{bmatrix} 10 & 0 \\ -2 & -1 \end{bmatrix}}_{A_c} x(t) + \underbrace{\begin{bmatrix} 5 & 1 \\ 4 & 10 \end{bmatrix}}_{B_c} u(t),$$

A_c is a lower triangular matrix

- analyse the following continuous-time linear time-invariant open-loop system, where both the state and the input vector are composed of two variables.
- the poles (eigen values) of the system are 10 and −1.
- Since one pole is in the left half plane is a stable and a right half plane is a unstable , the system has one unstable mode.

Now we need for control to stabilise the system.

An optimal linear quadratic regulator [26] is designed for this system, assuming that the controller execution is instantaneous and there is no one-step delay actuation.

$$K = \begin{bmatrix} -4.7393 & 0.2430 \\ 0.2277 & -0.8620 \end{bmatrix}.$$

check the stability of the closed loop system when the controller is executed with one step delay.

select a sampling period of 10 ms and discretise the system, we get

$$x_{[k+1]} = \underbrace{\begin{bmatrix} 1.1053 & 0.0000 \\ -0.0209 & 0.9900 \end{bmatrix}}_{A_d} x_{[k]} + \underbrace{\begin{bmatrix} 0.0526 & 0.0105 \\ 0.0393 & 0.0994 \end{bmatrix}}_{B_d} u_{[k]}.$$

A_d is also lower triangular.

The poles of the open-loop system are the numbers indicated in the main diagonal.

Since one of them is outside the unit circle, the discretised version of the continuous-time system is also unstable and needs control.

Stability Results for the Unstable Second-Order System.

	Misses	Stability	Lower Bound	Upper Bound
Zero&Kill	1	✓	0.961037	0.961975
	2	✗	1.071911	1.071915
Zero&Skip-Next	1	✓	0.914298	0.920769
	2	✗	1.059819	1.059822
Zero&Queue(1)	1	✓	0.961037	0.964287
	2	✗	1.071911	1.071915

Stability Results for the Unstable Second-Order System.

	Misses	Stability	Lower Bound	Upper Bound
Hold&Kill	1	✓	0.891089	0.891090
	2	✓	0.891089	0.891090
	3	✓	0.891089	0.891098
	4	✓	0.891089	0.891251
	5	✓	0.891089	0.935272
	6	✗	0.891089	1.004593
	7	✗	0.961344	1.083038
	8	✗	1.065537	1.172249
Hold&Skip-Next	1	✓	0.891089	0.891090
	2	✓	0.914556	0.944458
	3	✗	1.076507	1.091171
Hold&Queue(1)	1	✗	1.347066	1.370827

Electric Steering Application.

- Verify the stability of a permanent magnet synchronous motor for an automotive electric steering application in the presence of deadline misses.
- Present a standard model for the motor and a proportional and integral (PI) controller for setpoint tracking, written in the state-feedback form.

When modeling the motor, our system state is x(t) = [id(t), iq(t)]T where id(t) and iq(t) represent respectively the currents in the d and q coordinates over time.

The model of the motor can be written as

$$\dot{x}(t) = \begin{bmatrix} -R/L_d & L_q\,\omega_{el}/L_d \\ -L_d\,\omega_{el}/L_q & -R/L_q \end{bmatrix} x(t) + \begin{bmatrix} 1/L_d & 0 \\ 0 & 1/L_q \end{bmatrix} u(t).$$

Here, Ld [ht] and Lq [ht] denote respectively the inductance in the d and q direction, R [Ohm] is the winding resistance, and ωel [rad/s] is the frequency of the rotor-induced voltage.

R = 0.025 [Ohm], ωel = 6283.2 [rad/s], Ld = 0.0001 [ht],

Lq = 0.00012 [ht]

Tustin's method introduces frequency distortion, the method is what is currently applied in our industrial case study. Similar results can be obtained with the exact matrix exponential, changing the discretisation command parameters in the Matlab code.

using Tustin's method, and a sampling period of 10μs, obtaining

$$x_{[k+1]} = \underbrace{\begin{bmatrix} 0.996 & 0.075 \\ -0.052 & 0.996 \end{bmatrix}}_{A_{d,\mathbf{base}}} x_{[k]} + \underbrace{\begin{bmatrix} 0.100 & 0.003 \\ -0.003 & 0.083 \end{bmatrix}}_{B_{d,\mathbf{base}}} u_{[k]}.$$

the eigenvalues of Ad,base are 0.9957 ± 0.0626i and their absolute value is 0.9977, meaning that the open-loop system is stable (even without control).

To achieve zero steady-state error, we would like to design our controller in the PI form, using a term that is proportional to the error between the actual current vector and the setpoint vector and a term that is proportional to the integral of the error.

After this transformation, we denote the new system input as

$$v_{[k]} = [u_{[k]}^{\mathrm{T}}, w_{[k]}^{\mathrm{T}}]^{\mathrm{T}}$$

where w[k] denotes a vector that contains the desired values for the currents id and iq at time k.

The model the system as:

$$s_{[k+1]} = \underbrace{\begin{bmatrix} A_{d,\text{base}} & 0_{2\times 2} & 0_{2\times 2} \\ -I_2 & 0_{2\times 2} & 0_{2\times 2} \\ 0_{2\times 2} & I_2 & 0_{2\times 2} \end{bmatrix}}_{A_d} s_{[k]} + \underbrace{\begin{bmatrix} B_{d,\text{base}} & 0_{2\times 2} \\ 0_{2\times 2} & I_2 \\ 0_{2\times 2} & 0_{2\times 2} \end{bmatrix}}_{B_d} v_{[k]}.$$

We design our PI controller as:

$$K = \begin{bmatrix} 0_{2\times 2} & \overbrace{\begin{bmatrix} 5 & 0 \\ 1 & 7 \end{bmatrix}}^{K_1} & \overbrace{\begin{bmatrix} -4 & 0 \\ -3 & 7 \end{bmatrix}}^{K_2} \\ 0_{2\times 2} & 0_{2\times 2} & 0_{2\times 2} \end{bmatrix}.$$

Stability Results for the Electric Steering Application

	Misses	Stability	Lower Bound	Upper Bound
Zero&Kill	∞	✓	0.997713	0.997713
Zero&Skip-Next	1	✓	0.892575	0.892575
	2	✓	0.892575	0.892575
	3	✓	0.892575	0.892575
	4	✓	0.892575	0.892575
	5	✓	0.892575	0.892575
	6	✓	0.892575	0.892575
	7	✓	0.892575	0.892575
	8	✓	0.900922	0.900922
	9	✓	0.912902	0.912902
	10	✓	0.938565	0.938565
	11	✓	0.940823	0.940823
	12	✓	0.942610	0.942610
	13	✓	0.951092	0.951092
	14	✓	0.962846	0.962846
	15	✓	0.973776	0.973776
	16	✓	0.983954	0.983954
	17	✓	0.993436	0.993436
	18	✗	1.002273	1.002273
Zero&Queue(1)	1	✓	0.925966	0.925966
	2	✗	1.001620	1.001620
Hold&Kill	1	✓	0.892575	0.892575
	2	✓	0.938332	0.938332
	3	✗	1.073542	1.073542
Hold&Skip-Next	1	✓	0.968574	0.968574
	2	✗	1.107390	1.107390
Hold&Queue(1)	1	✗	1.423968	1.423968

Contributions

Name	Roll No.	Contribution (%)
Sumit Gupta	200394	25
Munna Kumar	200608	25
Raushan Kumar	210830	25
Saurav Kumar	210950	25

Implementation Theory

The state equation is

$$x_{[k+1]} = \begin{bmatrix} 1.1053 & 0.0000 \\ -0.0209 & 0.9900 \end{bmatrix} x_{[k]} + \begin{bmatrix} 0.0526 & 0.0105 \\ 0.0393 & 0.0994 \end{bmatrix} u_{[k]}$$

with

$$u_{[k]} = \begin{bmatrix} -4.7393 & 0.2430 \\ 0.2277 & -0.8620 \end{bmatrix} x_{[k-1]}$$

Implementation Result

Approach	Misses	Stability	Lower Bound	Upper Bound
Zero&Kill	1	✓	0.9613	0.9613
	2	✗	1.0286	1.1217
Zero& Skip Next	1	✓	0.9144	0.9144
	2	✗	0.9929	1.1808
Hold& Skip Next	1	✓	0.7846	0.7846
	2	✓	0.7446	0.8121
	⋮	⋮	⋮	⋮
	20	✓	0.6872	0.9994
	21	✗	0.6868	1.0049
Hold& Skip Next	1	✓	0.7585	0.7585
	2	✗	0.8471	1.0075

(** Our implementation might differ from original because of the different algorithms used for computing spectral radius.)